electronics

MDPI

Review

Deep Learning for Visual SLAM: The State-of-the-Art and Future Trends

Margarita N. Favorskaya

Department of Informatics and Computer Engineering, Reshetnev Siberian State University of Science and Technology, 660037 Krasnoyarsk, Russia; favorskaya@sibsau.ru

Abstract: Visual Simultaneous Localization and Mapping (VSLAM) has been a hot topic of research since the 1990s, first based on traditional computer vision and recognition techniques and later on deep learning models. Although the implementation of VSLAM methods is far from perfect and complete, recent research in deep learning has yielded promising results for applications such as autonomous driving and navigation, service robots, virtual and augmented reality, and pose estimation. The pipeline of traditional VSLAM methods based on classical image processing algorithms consists of six main steps, including initialization (data acquisition), feature extraction, feature matching, pose estimation, map construction, and loop closure. Since 2017, deep learning has changed this approach from individual steps to implementation as a whole. Currently, three ways are developing with varying degrees of integration of deep learning into traditional VSLAM systems: (1) adding auxiliary modules based on deep learning, (2) replacing the original modules of traditional VSLAM with deep learning modules, and (3) replacing the traditional VSLAM system with end-to-end deep neural networks. The first way is the most elaborate and includes multiple algorithms. The other two are in the early stages of development due to complex requirements and criteria. The available datasets with multi-modal data are also of interest. The discussed challenges, advantages, and disadvantages underlie future VSLAM trends, guiding subsequent directions of research.

Keywords: visual SLAM; deep learning; robotics; pose estimation; map construction; visual odometry

Citation: Favorskaya, M.N. Deep Learning for Visual SLAM: The State-of-the-Art and Future Trends. *Electronics* **2023**, *12*, 2006. https://doi.org/10.3390/electronics12092006

Academic Editors: Beiwen Li and KC Santosh

Received: 21 February 2023
Revised: 31 March 2023
Accepted: 23 April 2023
Published: 26 April 2023

1. Introduction

In recent decades, Simultaneous Localization and Mapping (SLAM) has been one of the scientific areas deserving active study in various autonomous robotic systems. SLAM involves creating or updating a map of an unknown environment and simultaneously tracking the location of the robot in it. This approach enables the robot not only to create a map of the travelled path, but also to accurately identify the scenes already visited. The benefits of progressively improved VSLAM algorithms are already being used in autonomous driving and navigation in different environments [1–3], service robots [4], virtual and augmented reality [5,6], and pose estimation [7,8].

SLAM has been a subject of research in robotics since the 1980s, focusing on the use of Sound Navigation and Ranging (SONAR) sensors, 2D laser scanners, and Light Detection and Ranging (LiDAR) scanners as primary sensors. Since the 2000s, thanks to improvements in hardware and advances in computer vision (particularly "Structure from Motion"), the SLAM approach has become visual SLAM based on low-cost video cameras. Since then, the use of VSLAM in real-time systems has gradually become a reality. Thus, VSLAM refers to those SLAM systems which use cameras (monocular, stereo, or RGB-D) as the main input sensors that capture visual information about unknown objects and environments. VSLAM technologies can be formally represented as visual-only, visual-inertial, or RGB-D-based. The visual-only VSLAM systems use monocular or stereo cameras and process 2D images and are considered the most studied and cheapest systems, despite the fact that stereo cameras are inferior in cost to monocular cameras. The main

benefit of the stereo camera application is to provide real information about the depth of the scene and pose in indoor and/or outdoor environments. It is currently the most researched field in VSLAM. Thus, MonoSLAM algorithm utilized an extended Kalman filter for visual measurement to estimate the ego-motion and the 3D coordinates of feature points in a scene [9]. In addition to MonoSLAM, many advanced but similar algorithms have been developed, such as parallel tracking and mapping [10], collaborative visual SLAM [11], monocular/stereo visual ORB-SLAM [12], and robust to fast camera motions RKSLAM [13], among others. Visual-inertial VSLAM systems provide rich information about the angular velocity, acceleration, and the magnetic field around the devices, which allows us to accurately assess the position of the sensors in dynamic scenes. However, a fusion of visual data and inertial measurements is a real problem, far from its reasonable implementation at the algorithmic and software levels. RGB-D sensors, including a monocular RGB camera and a depth sensor, simultaneously generate a color image and a dense depth map that help significantly in pose estimation and mapping. Thus, the RGB-D-based VSLAM architecture is simplified. However, this approach is only suitable for indoor environments due to the limitations of the depth sensor and requires large amounts of memory and power consumption. Sometimes other distance sensors are used instead of RGB-D sensors, such as SONAR sensors (for underwater UAVs), 2D laser scanners, or LiDAR scanners. In this case, data fusion is also necessary.

VSLAM is based on the principals of Visual Odometry (VO) and loop closure and includes six steps consisting of initialization (data acquisition), feature extraction, feature matching, pose estimation, map construction, and loop closure. Methods for simultaneous estimation of camera pose and scene structure from video are divided into feature-based methods and direct methods. For a long time, feature-based methods prevailed in VO. They include image input, feature extraction and matching, and tracking and mapping. Thus, the recent ORB-SLAM3 system, which analyzes information captured by monocular, stereo, and RGB-D cameras, as well as IMU (Inertial Measurement Unit) sensors, is more versatile, accurate, and robust than previous ones [14]. Direct methods treat pose estimation as a non-linear optimization problem, resulting in higher accuracy if the photometric calibration of the camera has been handled well. They minimize photometric error by iteratively optimizing the initial motion. Semi-direct methods take advantage of both approaches to achieve higher accuracy and efficiency [15]. Initially, semi-direct methods apply the principle of epipolar line constraint to match the features on the epipolar line. They then minimize the re-projection error and thereby solve the problem of pose estimation. However, direct and semi-direct methods require high-quality images and are sensitive to photometric changes.

The first major part of SLAM is VO, which estimates the pose of an agent (vehicle, human, or robot) using video sequences acquired from one or multiple cameras. The core of VO is camera pose estimation, the so-called real-time ego-motion estimation. Another task of VO is to track the position of moving agents based on a captured video sequence. VO is an inexpensive and alternative odometry technique compared to local techniques such as wheel odometry, SONAR localization systems, LiDAR sensors, Inertial Navigation Systems (INS), and global techniques such as GPS/GNSS or the low-cost Ultra-WideBand (UWB) positioning technology. At the same time, in practice, conventional video cameras sometimes cannot provide reliable data with poor rotation, low parallax movement, or complex illumination. Reliability can be improved by integrating information from multiple devices [16,17]. Another way is to use RGB-D sensors applied not only for analysis of the static, but also the dynamic environment [18]. There are numerous VSLAM methods focused on the application on a single platform. Since the 2010s, applications of a team of agents (drones, robots, etc.) have been of great of interest [11]. Such systems are called collaborative VSLAMs and have recently attracted many researchers.

Map construction followed by refinement and loop closure is the second major part of SLAM. Map construction is an essential requirement for locating a mobile robot or an unmanned autonomous vehicle in the indoor/outdoor environment to perform path planning tasks. For further optimization and recognition of previously visited places, a loop closure system is used. Thus, identical scenes can be easily recognized by the mobile agent.

We can mention three ways to integrate deep learning models into traditional VSLAM systems:

1. Adding auxiliary modules based on deep learning.
2. Replacing the original modules of a traditional VSLAM system with deep learning modules.
3. Replacing the traditional VSLAM system with end-to-end deep neural networks.

Since 2017, many outstanding surveys have been published in the field of VSLAM. Some of them are presented in Table 1. However, there are not many surveys in the field of VSLAM that include a partial discussion of deep learning. Moreover, VSLAM deep learning applications are discussed only in [19], where 42% of references were published in the last 5 years. Moreover, at present, virtual and augmented reality methods do not explicitly use deep learning models. Therefore, a detailed discussion of the latest deep learning models for VSLAM will fill a gap in systematic surveys and provoke further research.

Table 1. A summary of recent surveys in the field of VSLAM.

Title and Reference	Year	Deep Learning
Keyframe-based monocular SLAM: design, survey, and future directions [20]	2017	No
Survey and evaluation of monocular visual-inertial SLAM algorithms for augmented reality [21]	2019	No
Collaborative visual SLAM for multiple agents: A brief survey [22]	2019	No
A state-of-the-art review on mobile robotics tasks using artificial intelligence and visual data [23]	2021	Yes
SLAM; definition and evolution [24]	2021	Yes
Role of deep learning in loop closure detection for visual and LiDAR SLAM: A survey [25]	2021	Yes
A review of visual SLAM methods for autonomous driving vehicles [26]	2022	No
Advances in visual simultaneous localisation and mapping techniques for autonomous vehicles: A review [27]	2022	Yes
A survey of state-of-the-art on visual SLAM [28]	2022	Yes
Visual SLAM algorithms and their application for AR, mapping, localization and wayfinding [29]	2022	No
A comprehensive survey of visual SLAM algorithms [30]	2022	Yes
An overview on visual SLAM: From tradition to semantic [31]	2022	Yes
Overview of deep learning application on visual SLAM [19]	2022	Yes
Perception and navigation in autonomous systems in the era of learning: A survey [32]	2022	Yes
Approaches, challenges, and applications for deep visual odometry: Toward complicated and emerging areas [33]	2022	Yes
In-depth review of augmented reality: Tracking technologies, development tools, AR displays, collaborative AR, and security concerns [34]	2023	No
Augmented reality-based guidance in product assembly and maintenance/repair perspective: A state of the art review on challenges and opportunities [35]	2023	No
Automated guided vehicles and autonomous mobile robots for recognition and tracking in civil engineering [36]	2023	Yes

The contribution of this survey is as follows.

1. A brief description of historical development and some problem statements of deep-based VSLAM tasks are presented. Although the historical evaluation of SLAM systems is divided into two main periods: the past (1985–1999) and the present (2001–2023), we introduce a different interpretation in terms of the development of deep models—since 2017.
2. We provide a new and complete classification and overview of the recent VSLAM methods based on three ways to integrate deep learning into traditional VSLAM systems: (1) adding auxiliary modules based on deep learning, (2) replacing the original modules of a traditional VSLAM system with deep learning modules, and (3) replacing the traditional VSLAM system with end-to-end deep neural networks. These three ways have different degrees of elaborateness due to a short period of development.
3. Description of multi-modal VSLAM datasets suitable for supervised training and testing will help to select the most suitable datasets in terms of intra-cross and inter-cross validation. Most VSLAM datasets use real data obtained from multi-modal sensors. However, several datasets include simulation data using third party software tools.
4. Critical analysis of advantages and disadvantages provides further research on the integration of deep learning into VSLAM methods applied in many practical fields.

Structure diagram for the rest of the survey is depicted in Figure 1.

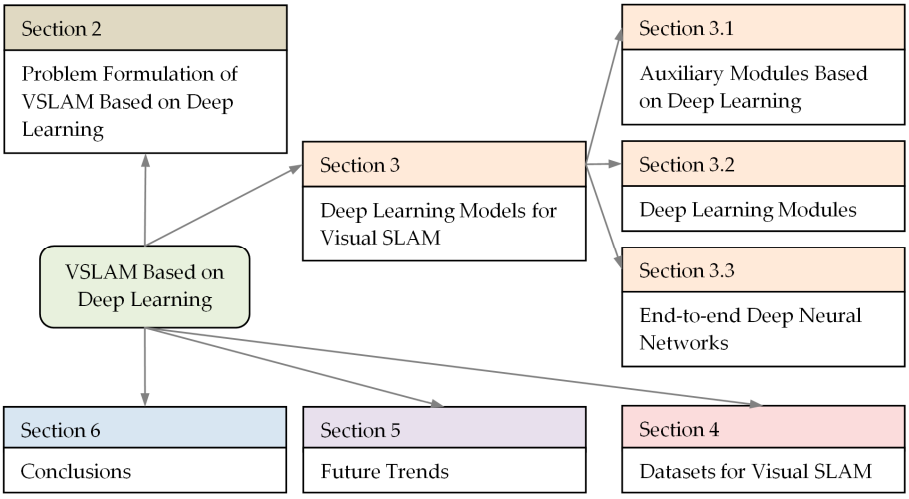

Figure 1. Structure diagram for the rest of this survey.

The rest of this survey is organized as follows. Some examples of problem formulations of VSLAM based on deep learning are discussed in Section 2. Deep learning models for VSLAM are presented in Section 3. Section 4 includes a brief technical summary of VSLAM datasets. Section 5 provides discussion and future trends. Section 6 concludes the survey.

2. Problem Formulation of VSLAM Based on Deep Learning

VSLAM is a rapidly evolving branch of SLAM based on computer vision paradigms. VSLAM components cover all the challenges of traditional SLAM and include data association (feature extraction, feature tracking, and motion tracking), pose estimation, map construction, map refinement, and loop closure. The algorithmic interpretation of this sequence is highly dependent on the types of SLAM sensors such as monocular, stereo, or RGB-D. Monocular SLAM methods were the earliest methods that estimated the location of objects in the environment, indoor or outdoor, as well as the position of the camera by direct (pixel-based), feature-based, or semi-direct (hybrid) methods. The basic stereo SLAM algorithms produce a dense colored point cloud that is more accurate than standard LiDAR SLAM algorithms. RGB-D SLAM methods are superior to traditional VSLAM methods, providing more depth range. They extract features using both 2D and 3D measurements,

improving interaction between frames. However, there are long distance limitations caused by the physical nature of RGB-D sensors.

Several problem statements in the traditional formulation of the VSLAM problem can be found in the literature, for example, feature extraction and matching [27], Extended Kalman Filter (EKF)-SLAM and its modifications [37], pose estimation of the omnidirectional robot [38,39], loop closure [40], etc. Due to the fact that deep learning does not support traditional computing paradigms, the formulation of VSLAM problems based on deep learning has fundamentally changed. Moreover, deep learning techniques have opened up new possibilities for VSLAM solutions, including depth prediction [41], object detection [42], and semantic/instance segmentation [43]. Let us consider some problem formulations for adding auxiliary modules based on deep learning (Example 1), replacing the original modules of a traditional VSLAM system with deep learning modules (Example 2) and replacing the traditional VSLAM system with end-to-end deep neural networks (Example 3).

Example 1. *Feature-based VSLAM systems use keypoints and descriptors to find a match between two consecutive frames. Many hand-crafted features were offered in the 2010s. A real-time VSLAM method called ORB-SLAM was presented in [12]. This method was based on the ORB (Oriented FAST and Rotated BRIEF) keypoint detector, which measures the orientation of a corner using the intensities of its surroundings. The main advantages of the ORB detector are achieving the highest feature extraction rate and a certain invariance to rotation and scale, which made it possible to develop the SOTA algorithm based on hand-crafted features in SLAM—ORB-SLAM2 [44]. The ORB-SLAM2 system includes three modes consisting of monocular, stereo, and RGB-D. Deep learning has fundamentally changed the approach to feature extraction. Thus, auxiliary modules based on deep learning began to develop.*

Zhu et al. [45] focused on improving the quality of feature matching for a SLAM interpolation engine using a deep network. They estimated the re-projection error using the 3D map point $\mathbf{P}_i = [X_i, Y_i, Z_i]^{\mathrm{T}} \in \mathbb{R}^3$ in world coordinates relative to a keypoint $\mathbf{p}_{t-1} \in \mathbb{R}^2$ in the previous frame and a matching keypoint $\mathbf{p'}_t \in \mathbb{R}^2$ in the current frame. The transformation matrix

$$\mathbf{M}_t = \begin{bmatrix} \mathbf{R} & \mathbf{T} \\ 0 & 1 \end{bmatrix} \tag{1}$$

recalculated the world coordinates to the current camera coordinates. This transformation matrix was the target matrix for optimization. Here, $\mathbf{R}$ is the rotation matrix and $\mathbf{T}$ is the translation vector. $\mathbf{p}_t$ denotes the coordinates of projecting $\mathbf{P'} = \mathbf{M}_t\mathbf{P}_i$ (the current camera coordinate) and is calculated as

$$\mathbf{p}_t = \mathbf{K}\ \mathbf{M}_t\ \mathbf{P}_i \tag{2}$$

where $\mathbf{K} \in \mathbb{R}^{3\times3}$ is the camera intrinsic parameters matrix.

However, usually $\mathbf{p}_{t-1}$ and $\mathbf{p}'_t$ do not represent the same visual object, since the number of features can be limited. A better feature correspondence means a smaller distance between $\mathbf{p}'_t$ and $\mathbf{u}_{gt}$, where $\mathbf{u}_{gt}$ is the ground truth of $\mathbf{p}_{t-1}$ in the current frame. Estimating $\mathbf{p}'_t$ can be achieved by n feature matching to approximate its ground truth $\mathbf{u}_{gt}$:

$$\mathbf{T}^* = \underset{\mathbf{M}_t}{\operatorname{argmin}} \frac{1}{2} \sum_{i=1}^{n} \left\| \mathbf{u}_{gt} + \Delta\mathbf{u} - \mathbf{p}_t \right\|, \tag{3}$$

where $\Delta\mathbf{u}$ is the error between $\mathbf{p'}_t$ and its ground truth $\mathbf{u}_{gt}$.

The proposed InterpolationSLAM network was trained to optimize Equation (3). Such manipulations with keypoints made it possible to build one interpolated frame between the previous and current frames to obtain more accurate velocity estimates.

Example 2. *One promising approach is scene reconstruction and ego-motion estimation from a sequence of unstructured frames. The unsupervised neural network proposed in [46] was trained to*

match the spatio-temporal 3D dependencies of unlabeled monocular frames, minimizing photometric loss and depth distortion loss. Unsupervised learning does not require high quality ground truth labels in depth images and camera poses. At each step, initial depth D_t, as well as camera rotation R_t and translation T_t, are estimated by a pair of consecutive frames I_t, I_{t+1}. A 3D matching model is then designed, where the depth projects onto a point cloud, and an optical flow corresponding to the camera pose is generated. To this end, the model minimized the depth warped loss L_D, considering forward-backward consistency constraints:

$$L_D = \sum_{x,y} \left\| D_t(x,y) - \hat{D}_t(x',y') \right\|_2, \tag{4}$$

where $D_t(x, y)$ is the inferred depth value from frame pair $I_t \rightarrow I_{t+1}$ at pixel (x, y) and $\hat{D}_t(x', y')$ is the warped depth value from frame pair $I_{t+1} \rightarrow I_t$ at pixel (x', y'). Pixel coordinates (x', y') are calculated as follows:

$$x' = x + U_t(x,y), \qquad y' = y + V_t(x,y), \tag{5}$$

where $U_t(x, y)$ and $V_t(x, y)$ are the optical flow between the two consecutive frames I_t and I_{t+1} at pixel (x, y) along OX and OY axes, respectively.

A depth smoothness loss L_S is calculated using image gradients to preserve sharp image details:

$$L_S = \sum_{x,y} \nabla D_t(x,y) \cdot \left(e^{-|\nabla(I(x,y))|} \right)^T, \tag{6}$$

where ∇ is the differential operator, $|\cdot|$ is the element-wise absolute value, and T is the transpose of weighted image gradient.

The 3D point cloud in the scene $\mathbf{P}_t = (X_t, Y_t, Z_t)$ is created using a pinhole camera model:

$$\mathbf{P}_t = \begin{bmatrix} X_t \\ Y_t \\ Z_t \end{bmatrix} = \frac{D_t}{f} \begin{bmatrix} \frac{x_t}{w} - c_x \\ \frac{y_t}{h} - c_y \\ f \end{bmatrix}, \tag{7}$$

where x_t and y_t are the column and row pixel coordinates in frame I_t, respectively, w and h are the numbers of pixels in column and row, respectively, while c_x, c_y, and f are the camera intrinsic parameters, which are taken from the camera documentation or are assigned empirically.

It should be noted that during 3D reconstruction tasks from the point clouds using a neural network, holes may appear. The problem of repairing holes is solved by generating hole regions within the boundaries of triangle patch.

The pinhole camera model in frame I_{t+1} is calculated through rotation and translation in frame I_t:

$$\mathbf{P}_{t+1} = R_t \mathbf{P}_t + T_t \tag{8}$$

and then projects a 3D point onto the image plane using the camera intrinsic parameters.

The corresponding 3D point $\mathbf{P}_{t+1} = (X_{t+1}, Y_{t+1}, Z_{t+1})$ in frame I_{t+1} is projected onto the image plane as a row and column position (x_{t+1}, y_{t+1}) using Equation (7). The rigid optical flow is calculated by simple subtraction of pixels, and the virtual frame method was used to estimate the optical flow of moving visual objects (so called non-rigid optical flow).

For estimating ego-motion, a pose sub-network, into which the depth and monocular images are fed, is trained using a photometric loss L_{ph} based on the structural similarity index SSIM:

$$\begin{aligned} L_{ph}^{t} &= \eta \tfrac{1-SSIM(I_t, I'_t)}{2} + (1-\eta)\|I_t - I'_t\|_2 \\ L_{ph}^{t+1} &= \eta \tfrac{1-SSIM(I_{t+1}, I'_{t+1})}{2} + (1-\eta)\|I_{t+1} - I'_{t+1}\|_2 \end{aligned} \tag{9}$$

where η is assumed to be 0.80 by cross validation while I'_t and I'_{t+1} are the synthesized frames from I_t and I_{t+1} frames, respectively, using 3D rotation expressions:

$$R_t^x(\alpha) = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad R_t^y(\beta) = \begin{pmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{pmatrix} \quad R_t^z(\gamma) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{pmatrix} \tag{10}$$

where α, β, and γ are the angles of rotation about the OX, OY, and OZ axes, respectively. The two fully connected layers predict a set of parameters including translation parameters T_t and sinα, sinβ and sinγ. The last three parameters must be in the interval [–1, 1], for when the Softplus activation function is applied.

Example 3. *In [47], a self-supervised monocular VO method was proposed. A deep network called PU-PoseNet evaluated the prediction-update poses of visual objects. It should be noted that the end-to-end self-supervised monocular VO framework is usually based on the idea of Kalman filter prediction-update and can reconstruct the original image using the outputs of the networks. Such networks are trained using photometric error. First, the PU-PoseNet network estimates the relative pose between two consecutive frames I_{t-1} and I_t. Second, the DepthNet network predicts the depth map D_t of the current frame. Third, the outputs of PU-PoseNet and DepthNet networks are used for reconstructing the current frame I_t from the previous frame I_{t-1}. The overall VO network is trained using both the reconstructed image from PU-PoseNet and the depth map from DepthNet according to the loss function:*

$$L = L_p + \lambda_d L_d + L_{VAE} + \lambda_s L_s, \tag{11}$$

where L_p is the photometric loss considering weighted movement based on long-term pose consistency, L_d is the depth consistency constraint loss, L_{VAE} is the VAE (Variatonal Auto-Encoder) loss, L_s is the depth smoothness loss, and λ_d and λ_s are the weighted coefficients for the depth consistency constraint loss and the depth smoothness loss, respectively.

In this case, the photometric loss function has the form:

$$L_p(I_t, \hat{I}_t) = \frac{\alpha_0}{2}(1 - SSIM(I_t, \hat{I}_t)) + \alpha_1 \left\| I_t - \hat{I}_t \right\|_1 + \alpha_2 \left\| I_t - \hat{I}_t \right\|_2, \tag{12}$$

where $SSIM(I_t, \hat{I}_t)$ is the structured similarity between the current frame I_t and the reconstructed frame $\hat{I}_t$ while α_0, α_1, and α_2 are hyper-parameters.

The depth consistency error depends on the depth of moving objects and the camera movement. For the static objects in the scene, the depth consistency error, similar to the photometric error, is calculated using the Euclidean transformation applied to the sampled frame. The VAE loss determines the noises robustness of the VO by assuming that the random vector satisfies a multivariate Gaussian distribution. The depth smoothness loss improves the representation of geometric details in the scene and is defined as follows:

$$L_s = |\partial_x D_t^*| e^{-|\partial_x I_t|} + |\partial_y D_t^*| e^{-|\partial_y I_t|}, \tag{13}$$

where D_t^* is the disparity and ∂ is the partial derivative.

To eliminate occlusions, the mask can be constructed to remove data about moving or occluded objects. The estimated depth values of static objects in the scene may be infinite thus reducing the accuracy of pose estimation. To solve this problem, a mask of stationary pixels is automatically created to exclude stationary pixels from the training process of the VO network.

We have given just a few examples of the VSLAM problem formulation using deep learning models. However, one can see a specific approach to the use of deep neural networks in VSLAM tasks.

3. Deep Learning Models for Visual SLAM

The VSLAM framework includes many fundamental topics that are traditionally considered to be different computer vision tasks. The traditional geometric methods can be divided into filtering-based VSLAM methods, which were widely studied initially due to their low computational cost, and optimization-based VSLAM methods showing higher accuracy. Although traditional geometric VSLAM methods have achieved amazing results in environmental mapping and localization, they have several disadvantages related to sensitivity to changing lighting/weather/seasonal conditions, scale non-invariance, and so on. Some of these shortcomings have been overcome with deep learning paradigms, but not all.

The active use of deep learning and reinforcement learning in VSLAM can be traced back to the mid-2010s. This work was carried out in three main areas: the development of auxiliary deep-based modules, the development of original deep learning modules, and the construction of end-to-end deep neural networks. These main directions are considered in Sections 3.1–3.3, respectively.

3.1. Auxiliary Modules Based on Deep Learning

Table 2 provides a summary of the recent auxiliary modules based on deep learning since 2018. The information is sorted by the "Main Subject" attribute including feature extraction, semantic segmentation, pose estimation, map construction, and loop closure and the "Year" attribute. This field in deep learning for VSLAM is most numerous. As can be seen from Table 2, most deep learning models use a supervised ("SP") learning strategy, denoted by a "+" sign. The unsupervised learning strategy is labeled as "US". The description of the datasets is presented in Section 4.

Hereinafter, a brief discussion of the publications presented in Table 2 is given in terms of deep learning paradigms. Please note that sometimes the authors do not specify the title of their deep neural models.

Feature extraction

Feng et al. [48] developed an end-to-end deep network called 2D3D-MatchNet for jointly matching descriptors based on 2D and 3D keypoints extracted from an image and a point cloud. The main idea was to use LiDAR data to build more accurate 3D reference maps for pose estimation. 2D3D-MatchNet had a triplet-like deep network architecture that evaluated the similarity between a given pair of image patches and the local point cloud volume. Image patches centered on the 2D image keypoints were fed into one of the branches based on the VGG-16 model. The output was a descriptor of the 2D image keypoints. The other two branches with shared weights learned the descriptor of the 3D point cloud keypoints. The positive and negative local volume point clouds were fed into these branches. The network was trained using the triplet loss function estimated by a Euclidian metric. The pose of the camera was computed from the supposed set of 2D-3D correspondences. The authors created a dataset with a huge collection of 2D-3D image patches with 3D point cloud volume correspondences called the Oxford 2D-3D Patches dataset.

Table 2. A summary of recent auxiliary modules based on deep learning.

Method	Year	Main Subject	Data	Learning Strategy		Dataset
				SP	US	
2D3D-MatchNet [48]	2019	Feature extraction	Monocular, LiDAR data	+		Oxford 2D-3D Patches Dataset
SP-Flow [49]	2020	Feature extraction	Monocular, Stereo, Depth		+	KITTI Visual Odometry, TUM RGB-D
LIFT-SLAM [50]	2021	Feature extraction	Monocular, Intertial data	+		KITTI Visual Odometry, EuRoc MAV

Table 2. *Cont.*

Method	Year	Main Subject	Data	Learning Strategy		Dataset
				SP	US	
[51]	2018	Semantic segmentation	Monocular	+		CARLA simulator
[52]	2019	Semantic segmentation	Monocular	+		KITTI Visual Odometry, TUM-mono
ObjectFusion [53]	2019	Semantic segmentation	Monocular, Depth	+		Own dataset
Deep SAFT [54]	2020	Semantic segmentation	Monocular	+		TUM RGB-D, ICL-NUIM
EF-Razor [55]	2020	Semantic segmentation	Monocular, Depth	+		TUM RGB-D
RoomSLAM [56]	2020	Semantic segmentation	Monocular	+		MIT Stata Center, TUM RGB-D,
USS-SLAM [57]	2020	Semantic segmentation	Monocular		+	Pascal VOC, SBD, COCO
[58]	2022	Semantic segmentation	Monocular, Depth		+	Virtual KITTI 2, KITTI Visual Odometry, Extended CMU Seasons, RobotCar Seasons
[59]	2020	Semantic segmentation	Monocular, Inetrtial	+		ADVIO
[60]	2022	Semantic segmentation	Monocular			TUM RGB-D
[45]	2022	Pose estimation	Monocular	+		KITTI Visual Odometry, TUM RGB-D, own dataset
[46]	2022	Pose estimation	Monocular		+	KITTI Visual Odometry
ObjectFusion [61]	2022	Pose estimation	Monocular, Depth	+		SceneNet RGB-D, ScanNet
Cowan-GGR [62]	2022	Pose estimation	Monocular	+		KITTI Visual Odometry, MidAir, Synthetic images
TransPoseNet [63]	2023	Pose estimation	Monocular, Depth		+	RGB-D 7-Scenes
ORGPoseNet, ORGMapNet [8]	2023	Pose estimation	Monocular	+		RGB-D 7-Scenes, RIO10, Oxford RobotCar
LKN [64]	2019	Map construction	Monocular	+		KITTI Visual Odometry, ApolloScape
DRM-SLAM [65]	2020	Map construction	Monocular, Depth	+		NYU RGB-D V2, TUM RGB-D, ICL-NUIM
Mask-RCNN [66]	2020	Map construction	Monocular	+		Own dataset
[3]	2021	Map construction	Stereo	+		Own agricultural dataset
[67]	2020	Loop closure	Monocular	+	+	City Centre, KITTI Visual Odometry, Gardens Point Walking
Triplet Loss [68]	2021	Loop closure	Monocular	+		TUM2, City Centre

Table 2. *Cont.*

Method	Year	Main Subject	Data	Learning Strategy		Dataset
				SP	US	
[69]	2022	Loop closure	Monocular	+		KITTI Visual Odometry, Oxford RobotCar
PlaceNet [70]	2023	Loop closure	Monocular	+		CityScape, subset of ADE20K

An approach suggested by Qin et al. [49] can be connected to feature-based methods of VO. The authors developed a keypoint extraction network called SP-Flow, which combined a self-supervised framework and the Lucas–Kanade (LK) optical flow algorithm. The ORB feature extractor was replaced with SP-Flow, which was easy to train and, at that time, did not require labeled images for network training. Optical flow was used to match feature points. The SP-Flow network involved six conventional convolution layers, a middle convolution layer, and a pixel shuffle layer. The output represented a probability map with keypoints extracted. The originality is a progressive training process that consists of three steps: keypoint pre-training, keypoint self-labeling, and joint training. A similar approach to detecting and matching keypoints in the underwater environment has been demonstrated in [71]. For this, a very simple network architecture such as LeNet-5 was used.

Bruno and Colombini [50] proposed to use the Learned Invariant Feature Transform (LIFT) module [72] in the traditional VSLAM concept using ORB-SLAM. The network architecture included three main CNN-based modules called Detector, Orientation Estimator, and Descriptor. The network processed image patches. Since the original LIFT was trained on photo-tourism datasets, the authors retrained it on VO datasets.

Semantic segmentation

The main VSLAM feature-based methods use rich image features and robust descriptors to match feature correspondences in different images or frames. However, such fixed feature presentation suffers from a performance penalty in environments with low-textured regions, low-structured areas, or motion blur. Deep learning models do not need handcrafted feature extraction. Thus, the traditional VSLAM task relative to feature extraction has been transformed into semantic segmentation.

Kaneko et al. [51] combined the results of VSLAM and semantic segmentation using the DeepLab v2 model in order to improve VSLAM. The improvement concerned the removal of feature points of dynamic objects; in other words, the authors proposed to select only reliable correspondences of stationary objects. The system was trained on scenes of an urban environment with numerous vehicles, pedestrians, etc. using the CARLA simulator [73]. The authors empirically found that the "Car" class (moving objects) and the "Sky" class (distant area) worsened the performance of VSLAM. Thus, a mask based on the semantic segmentation of these areas did not allow processing feature points in these masked areas.

The Faster R-CNN model with semantic filtering was applied to solve the outlier problem in RANSAC (RANdom SAmple Consensus) based on F-matrix calculations [52]. Instead of feature points, stable semantic regions were extracted from the two images in order to increase the accuracy of the RANSAC method. Such a semantic filter was implemented in the ORB-SLAM system.

The original problem statement was given in [53], when both 3D detection and semantic segmentation were performed by CNN for general object detection, and then the SLAM system improved these results. First, CNN detected 2D objects in frames and built a local target map. Second, the local target map was fused with the SLAM results in order to update the global target map as a 3D surfel model (surfels are simple surface elements). Third, the global target map was projected in the current 2D frame. Modified RANSAC was used to remove the outliers. The designed system included three main components: a

CNN-based object detection module (Faster R-CNN), a surfel-based SLAM module, and a Fusion-Update module.

Xu at al. [54] proposed an online Scene Adaptive Feature Transform (SAFT) module called Deep-SAFT to replace the corresponding feature extraction module in the SLAM system, such as ORB-SLAM2. A learning-based descriptor was implemented in the Siamese network followed by the decision network. The grayscale image patches are fed into the two Siamese branches that extract two 256-dimensional descriptors while the next decision network computes their similarity score. The Deep-SAFT module was embedded into the ORB-SLAM2 (RGB-D) system, providing the threads for modified tracking and online learning, plus additional ones such as local mapping and loop closure used in ORB-SLAM2 with minor enhancements.

An effective method for processing the edge-features in SLAM called Edge-Feature Razor (EF-Razor) was proposed in [55]. YOLOv3 [28] was used to accurately identify the position of the object. At first, all edge-features detected by YOLOv3 are considered unstable. EF-Razor then implements stricter criteria for keypoint pairs, removing unstable ones. Thus, EF-Razor can work in cases where an object is occasionally lost.

RoomSLAM presented in [56] was built on traditional VSLAM back-end optimization, including block of measurements, front-end part with object detector, wall detector, robot pose prediction, and room detection/creation, as well as data association and back-end part with sub-graph (room) optimization and loop closure detection/correction. Deep learning has only been used for object detection in the form of a one-stage YOLOv3 detector pre-trained on the COCO dataset. RoomSLAM was evaluated with two datasets: the TUM RGBD dataset and the MIT Stata Center dataset.

Jin et al. [57] proposed a semantic SLAM framework with Unsupervised Semantic Segmentation (USS-SLAM) in dynamic environments. The USS-SLAM framework ran four threads in parallel: tracking thread, local mapping thread, loop closing thread, and semantic map generation thread. The tracking thread based on the ORB-SLAM2 system filtered out feature points belonging to dynamic objects, combining a semantic segmentation model with a multi-view geometry method. To improve the performance of semantic segmentation, an unsupervised adversarial learning method was used. It should be noted that the main advantage of unsupervised learning is the application of unlabeled per-pixel ground truths. Moreover, these authors suggested adversarial transfer learning in the multi-level feature spaces to transfer more information at different levels of the semantic segmentation model. Keyframes were used as input data.

A long-term visual localization method under changing environments was proposed in [58]. This method used semantic segmentation to create an invariant scene representation. Additional deep information made it possible to clarify the difference in appearance between images due to environmental changes. The authors also used the automatic synthesis dataset to reduce computational costs in building the depth maps and specifying semantic labels in a real scene. The trained model had significant generalization ability due to domain adaptation strategy. The model was trained on the vKITTI2 and KITTI VO datasets and then tested on the Extended CMU Seasons and RobotCar Seasons datasets with better results than other SOTA visual localization models.

Zhao et al. [59] proposed a real-time semantic visual-inertial SLAM system for dynamic environments. This system utilized the pixel-wise results of semantic segmentation based on DeepLabv3+ [74]. A monocular Visual-Inertial System (VINS) in combination with RGB sequences made it possible to achieve real-time trajectory estimation. Feature tracking and extraction modules have been integrated into the front-end of the developed SLAM system. For experiments, the authors used the ADVIO dataset containing dynamic indoor and outdoor scenes.

The same idea with using ORB-SLAM2 and a parallel semantic thread based on the lightweight YOLOv5 detector in a dynamic environment was developed in [60]. For the tracking thread of static feature points, the optimization of the homography matrix with the

removal of noise points was implemented using the RANSAC algorithm. The optimized optical flow mask made it possible to remove dynamic characteristic points.

Pose estimation

Zhu et al. [45] presented an interpolation network in order to improve the performance of feature-based VSLAM systems. Both hand-crafted and deep learning features were verified. The authors integrated their interpolation network into the ORB-SLAM2, SP-SLAM [75], and DSO (Direct Sparse Odometry) [76] frameworks. Since the interpolation network slows down the pose transformation, such an approach can be recommended for a better initial pose estimation based on the assumption of constant velocity.

Song et al. [46] proposed a multi-task approach for estimating optical flow, pose, and scene depth based on spatio–temporal 3D dependencies matching. They designed sub-network models for these tasks using warped depth maps and frames. To estimate object motion and ego-motion, a self-supervised iteration model was also developed, in which rigid flow was optimally converted to optical flow using the virtual frame method, taking into account dynamic regions.

Zou et al. [61] developed the ObjectFusion system, which estimated the camera pose in each frame and incrementally built up 3D surface reconstructions of object instances in a scene. A typical encoder-decoder CNN created instance segmentation masks using RGB-D frame. The distance value of the surface points (depth) and the projection silhouette were used for the object shape and pose inference based on the ResNet50 model. The camera pose was then estimated using hybrid camera tracking based on both the deep implicit object representation and sparsely sampled map points. The SceneNet RGB-D dataset and the ScanNet dataset were used for training.

In order to overcome the well-known challenges of dynamic environment, limitations of aerial datasets for training, and embedded hardware constraints, Mumuni et al. [62] proposed a confidence-weighted adaptive network with geometric-guided refinement called Cowan-GGR. They designed and tested a lightweight real-time CNN for the UAV system, which included three deep models. DepthNet estimated the depth of the scene using a single monocular image. At the same time, EgoMNet and OFNet predicted the camera pose and optical flow, respectively, using two adjacent frames.

Li et al. [63] developed a fast system for indoor localization under low illumination or night conditions. The processing consisted of four steps, including initial pose estimation, search of referenced Point ClouDs (PCDs), point cloud generation based on keypoints, and geometry alignment using Iterated Closest Points (ICP). The proposed TransPoseNet used visual and depth images to initialize the pose. Pose initialization and pose refinement were performed using deep learning pose regression and keypoint geometry alignment, respectively. The proposed TransPoseNet jointly performed self-supervised depth mapping and transformer-based pose regression. In other words, Transformer and CNN were combined into a single neural network for pose regression. This architecture made it possible to extract both local and global features in depth images. For pose refinement, a method based on the ICP algorithm was developed, which included three steps: sub-map selection, ICP-based geometry alignment, and pose refinement across multiple results. The authors empirically showed that their method outperforms the traditional keypoints detectors such as SIFT and SURF using the 7-Scenes dataset, which contains a collection of RGB-D frames.

An Object Relation Graph (ORG) has been proposed in [8]. ORG incorporated deep multi-layer GNNs to exploit the semantic connections and relative spatial clues of the objects. First, the Fully Convolutional One-Stage (FCOS) network [77] detected the objects of interest and labeled them with bounding boxes. Second, the detected objects and the corresponding features were integrated with ORG. A four-layer GNN structure extracted multi-level object features. Edges based on the updated features were dynamically built in each GNN layer and then concatenated and fed into the fully connected layer. The developed ORG module was embedded into PoseNet [78] and MapNet [79]. The resulting

networks called ORGPoseNet and ORGMapNet were tested on several datasets including 7-Scenes, RIO10, and Oxford RobotCar.

Map construction

Although the monocular VSLAM methods can accurately track ego-motion and camera poses, the constructed 3D maps are extremely sparse. Even the stereo matching application fails in the textureless regions of raw frames.

Zhao et al. [64] proposed an ego-motion estimation system through current measurements, suitable for building 3D maps in urban environments. They designed a monocular VO system based on a hybrid Learning Kalman Network (LKN). On the one hand, this model demonstrated non-linearity. The learning observation network included the FlowNet2 model as an optical flow network and a network for ego-motion estimation. On the other hand, the Kalman probabilistic mechanism using LSTM (Long-Short Term Memory) cells allowed them to update the state. LKN provided powerful constraints for trajectory filtering in the spatio-temporal domain. The LKN-VO system has been successfully integrated with dense 3D mapping.

In [65], a dense reconstruction method was developed using sparse depth samples and predicted dense depth maps. The ORB-SLAM model generated sparse depth samples, while the CNN model created depth maps. The ResNet-101 model was chosen as the backbone with some modifications. Keyframes in the form of single-view color images were fed to the inputs of ResNet-101. Due to the fact that the sparse depth map created by ORB-SLAM and the dense depth map obtained from CNN were not on the same scale, a robust RANSAC-based least square method was employed to determine the optimal scale factor. However, the depth estimation module and the depth fusion module were implemented separately, not as an end-to-end framework.

In [66], a Mask Regional CNN (Mask RCNN) model, enhanced with a feature pyramid network, improved a 3D semantic map of an indoor environment using keyframes. The authors created their own dataset with 21 types of objects commonly found in laboratory and home scenes and applied data augmentation.

Some specific tasks, such as large-scale agricultural tasks, are not solved only by the basic SLAM application. Thus, the main contribution of [3] was the development of a global mapping framework suitable for fruit picking tasks based on stereo vision and large-scale SLAM methods. For this purpose, the EfficientDet-D3 network [80] was applied, providing progress in orchard picking tasks in various environmental conditions and limited computational costs.

Loop closure

It is well-known that scene recognition allows a robot to navigate in an already visited scene. The recognition of such visited scenes is carried out by the loop closure detection module, which performs self-localization and reduces the drift in the map due to the movement of the robot.

In [67], one deep neural network accelerated loop closure detection, while another deep neural network detected moving objects such as pedestrians, vehicles, bikes, animals, etc. to eliminate their influence on loop closure detection. The proposed system for loop closure detection used a super dictionary and deep learning features extracted in parallel streams.

The framework of triplet loss based metric learning for a SLAM system was proposed in [68]. In this study, the ResNet_V1_50 has been introduced to make the feature vector more expressive. Keyframes were converted to feature vectors and their similarity was calculated using Euclidean distance. Feature vectors were utilized to determine if a closed loop was forming.

Duan et al. [69] considered the loop closure detection problem in a special problem statement. As is well-known, loop closure detection involves two steps: matching keyframes in the current scene and searching for a transformation between keyframes for the correct trajectory. The trajectory correction is then usually achieved by pose graph optimization. The authors embedded keyframe descriptors in the pose graphs in such

a way that each node of the pose graph was a keyframe with 6-DoF pose attributes and an encoded image. Thus, the local map based on the pose graph has become a sparse image. Deep Feature Matching (DFM) [81] was used to solve the two steps mentioned above simultaneously on a large data scale. The proposed method was evaluated on the KITTI dataset and the Oxford RobotCar dataset.

Osman et al. [70] developed a plug-and-play model called PlaceNet that detects loop closure. The main idea was to learn the multi-scale deep auto-encoder network in a way that avoids tracking dynamic objects. The introduced PlaceNet had a U-Net architecture trained on scaled input images. The CityScapes dataset and a subset of the ADE20K dataset were used for training, while the City Center dataset, the KITTI vision benchmark suite, the Nordland dataset, the Gardens Point (GP) dataset, and TUM-SLAM dataset were applied for testing the designed model.

3.2. Deep Learning Modules

Another way is to replace the original modules of a traditional VSLAM system with deep learning modules. A summary of several methods, not numerous, is shown in Table 3. The acronyms "SP" and "US" stand supervised and unsupervised learning, respectively. The learning strategy used is indicated by a "+" sign.

Table 3. A summary of recent original deep learning modules.

Method	Year	Main Subject	Data	Learning Strategy		Dataset
				SP	US	
[82]	2017	Camera relocalization	Monocular	+		RGB-D 7-Scenes
DistanceNet [83]	2019	Distance estimation	Monocular	+		KITTI Visual Odometry
DDL-SLAM [84]	2020	Object segmentation, Background inpainting	Monocular, Depth	+		TUM RGB-D, PASCAL VOC
PSPNet-SLAM [85]	2020	Object segmentation	Monocular, Depth	+		TUM RGB-D
[86]	2022	Path planning	Monocular		+	Own dataset
DEM [87]	2020	Scene reconstruction	Monocular, Depth	+		NYU-Depth-v2, KITTI Visual Odometry

Wu et al. [82] solved the problem of camera relocalization with a single monocular image using CNNs. To avoid problems with periodicity of angle values, the authors used the Euler method based on the 6D orientation vector of a rigid body. Data augmentation helped to reduce the sparsity of poses during training. Finally, the authors developed and tested the BranchNet architecture with the idea that the orientation and translation vectors were predicted by different branches compared to a single branch of the PoseNet architecture.

Kreuzig et al. [83] proposed an end-to-end many-to-one traveled distance estimator based on CNN and Recurrent Neural Network (RNN). The developed DistanceNet estimated the traveled distance of the ego-vehicle between the first and last image of a set of consecutive frames. First, CNN extracted geometric features of several frames that had semantic meaning. Second, the obtained features were fed to the RNN to estimate the distance. FlowNetC was used as a feature extractor that estimated the optical flow between two images. The correlation layer of FlowNetC provided an increase in the size of the functional channel up to 1024. The RNN acquired pairs of images over multiple time steps and estimated the distance between the first and last given images. This allowed them to estimate the distance traveled between consecutive frames.

A robust RGB-D SLAM system for dynamic scenarios was developed in [84]. The Dynamic Deep Learning SLAM (DDL-SLAM) provided dynamic object segmentation and background inpainting capabilities for the ORB-SLAM2 model. DDL-SLAM adopts Deformable U-Net [88] to implement pixel-wise semantic segmentation trained on the PASCAL VOC dataset. Multi-view geometry has been added to the system to improve segmentation of dynamic objects and search for new dynamic objects that were static most of the time. Tracking and mapping were implemented based on ORB-SLAM2. For background inpainting, the last 15 previous keyframes were selected for projection into the dynamic parts of the current frame. A very close approach can be found in [85] while based on a pyramid scene parsing network [89]. The PSPNet semantic segmentation network filtered the feature points by optical flow, detecting a priori dynamic points to eliminate them, while stable static feature points were used to build semantic maps. This network has been integrated with the ORB-SLAM2 system.

Footstep planning for the indoor navigation of humanoid robots was discussed in [86]. The authors proposed the GAN-based architecture for building an accurate path (even the narrow path) for planning the footsteps of a humanoid robot. The GAN model for generating the optimal path was implemented in Robot Operating System (ROS). The aim was to solve the motion planning problem of a humanoid robot in an unknown environment between obstacles. The GAN-based footstep planner included a generator for creating a high-resolution map from an input map containing random goal and obstacle points and a discriminator, which classified the generated maps as real or fake. The generator had a typical U-Net architecture, while the discriminator included two branches (for real and fake inputs) of a Siamese-like architecture. The system was tested on its own dataset and demonstrated an accuracy of approximately 93% of the generated footsteps using the GAN-based path planner.

The Depth Estimation Model (DEM) based on an encoder-decoder for the SLAM applications was proposed in [87]. The capabilities of DEM were tested with three modalities: RGB images, sparse depth, and RGB-D data. The ResNet-50 model and the modified DPN-92 model were employed as the encoder for processing the RGB images and the depth images, respectively. A decoder generated pixel-wise depth images of size 228×304, learned by the transposed convolution and convolution layers. The DEM module was successfully incorporated in the SLAM system.

3.3. End-to-End Deep Neural Networks

Recently, deep learning has been used to solve problems of detection, localization, classification, and control of objects, eliminating the shortcomings of geometric solutions in SLAM. With more efficient and higher-level feature extraction, deep learning solutions become better than conventional monocular VO/VIO or VSLAM/VISLAM solutions. The conventional VO systems require an accurate solution at each step, including feature detection, feature matching, camera calibration, local optimization, etc., while deep-based SLAM systems do not need a camera calibration step due to how the correspondences between sensor data and target values are learned during the training phase. Thus, deep-based SLAM methods are related to direct VSLAM/VO methods [33]. Only in recent years have end-to-end deep neural networks been developed for SLAM applications.

The third way is to use end-to-end deep networks instead of a traditional VSLAM system. A summary of several methods ordered by the criterion "Main Subject" is presented in Table 4. The acronyms "SP", "US" and "RL" stand supervised, unsupervised and reinforcement learning, respectively. The learning strategy used is indicated by a "+" sign.

Lan proposed to integrate VSLAM with CNN, called vSLAM-CNN (vCNN) for surgical applications [90]. It should be noted that SLAM technology has been widely applied in robots and unmanned vehicles, but rarely in the analysis of surgical videos. The vCNN model generated region predictions using constant localization from 3D maps and provided deep learning models to capture objects with bounding boxes. The Mask R-CNN model

was chosen as a backbone for object classification, and a recurrent neural network with LSTM cells predicted the workflows.

An end-to-end monocular VO system based on self-supervised learning has been proposed in [47]. The authors used the Kalman filter to predict and update their data and they developed a prediction update mechanism that took into account information from previous pose measurements and applied a novel training strategy. To reduce the effect of local motion, a motion-weighted photometric loss was formulated based on the constraints of long-term posture constancy. The architecture of PU-PoseNet (Prediction-Update Pose Estimation Network) included several networks for pose measurement, weighted fusion, pose predict, and pose update. The KITTI odometry dataset was used for experiments.

In [91], the UAV pose estimation was improved by reducing the noise of the inertial data using the Savitzky-Golay filter and extracting visual-inertial features based on the Inception-v3 model, which were fed to the Gaussian process regression unit. The inertial image features have been enhanced with optical flow. The proposed method was tested on the EuRoC dataset and its own dataset.

Table 4. A summary of recent end-to-end deep neural networks.

Method	Year	Main Subject	Data	Learning Strategy		Dataset
				SP	US/RL	
vCNN [90]	2022	Sub-VSLAM	Monocular	+		M2CAI 2016 Challenge
PU-PoseNet [47]	2022	Pose estimation	Monocular		+	KITTI Visual Odometry
[91]	2022	Pose estimation	Monocular, Inertial	+		EuRoC, own dataset
VIOLearner [92]	2018	Trajectory estimation	Monocular, Depth, Inertial		+	KITTI Visual Odometry
DeepMLE [93]	2022	Depth estimator	Monocular, Depth	+		KITTI Visual Odometry, Virtual KITTI 2, DeMoN
PoseConvGRU [94]	2020	Ego-motion estimation	Monocular	+		KITTI Visual Odometry, Malaga 2013
DeepAVO [95]	2022	Ego-motion estimation	Monocular	+		KITTI Visual Odometry, Malaga, ApolloScape, own dataset
DeepVO [96]	2017	Visual odometry	Monocular	+		KITTI Visual Odometry
UnDeeopVO [97]	2018	Visual odometry	Monocular, Depth, Stereo		+	KITTI Visual Odometry
HVIOnet [98]	2022	Visual–inertial odometry	Monocular, Intertial	+		EuRoC, ROS-based simulation dataset
SelfVIO [99]	2022	Visual–inertial odometry	Monocular		+	KITTI Visual Odometry, EuRoC, Cityscapes
[100]	2021	Loop closure	Monocular	+		Own dataset
MGRL [101]	2021	Visual navigation	Monocular		+	AI2-THOR framework
VGF-Net [102]	2021	Drone navigation	Monocular, Depth, GPS	+		Own dataset

Shamwell et al. [92] developed an unsupervised deep neural network called VIO-Learner for absolute trajectory estimation. VIOLearner estimated scaled ego-motion based on RGB-D imagery and inertial data as input. VIOLearner received the RGB-D source image, target RGB image, IMU data from consecutive frames, and intrinsic camera parameters from calibration matrix. VIOLearner performed multi-scale projections and online error

correction. At each scale, the network computed the Jacobians of the re-projection error associated with the grid. The authors argued that the unsupervised VIOLearner model performed online error correction by combining uncalibrated and weakly time-synchronized multi-modal data from different frames to improved VO estimates.

Xiao et al. [93] formulated the two-view Structure from Motion (SfM) problem as a Maximum Likelihood Estimation (MLE) problem and solved it with a proposed framework called DeepMLE. They developed an end-to-end DNN that iteratively searched for the optimal estimation to maximize the likelihood of using the gradient-like data to improve robustness and generalization capability. DeepMLE had three parts: correlation volume calculation, uncertainty prediction, and iterative solver. At each iteration, the correlation module evaluated the pixel correspondences between target and source images. The prediction module predicted the uncertainty parameters for each pixel under the influence of illumination, occlusions, moving objects, and noise. The likelihood map and gradient information were then computed based on the correlation map and uncertainty parameters. Finally, the deep iterative solver updated the depth and pose. The model was tested on three datasets with indoor and outdoor scenes under various meteorological conditions.

Zhai et al. [94] introduced a long-term recurrent CNN called PoseConvGRU. The PoseConvGRU model is a fully end-to-end network that encodes geometric features for camera position estimation. This recurrent CNN extracted the geometric relationship features from two adjacent frames in a video sequence, fed the feature maps to the stacked ConvGRU module, and then built a relative pose regression function. The backbone of PoseConvGRU was FlowNetSimple [103] without a decoder part. The proposed model did not use dense optical flow, which made it faster and less computationally expensive. As a result, it demonstrated better results than the VISO2-M model and the ORB-SLAM-M model.

For an accurate assessment of ego-motion, Zhu et al. [95] proposed a framework for learning monocular VO in an end-to-end way. The proposed lightweight DeepAVO model based on frame-to-frame analysis yielded four parallel CNNs processing four quadrants of the optical flow. Convolutional Block Attention Module (CBAM) [104] was implemented as a dual attention mechanism that recalibrated the feature map in the channel and spatial domains to find essential information and suppress useless information.

One of the first end-to-end frameworks for monocular VO was proposed in [96]. The framework was based on deep RNNs, which have demonstrated the ability to generalize to completely new environments using geometric feature representation and capturing sequential dependence and complex motion dynamics. The main advantage of this architecture was feature extraction with CNN and sequential VO modeling with RNN as a simultaneous process.

The UnDeepVO architecture proposed in [97] was fundamentally different from the basic version. The unsupervised UnDeepVO model, which was scale invariant, estimated the 6-DoF pose and depth of view of a monocular camera using CNNs. The pose estimator as a VGG-based model predicted the 6-DoF transformation between two consecutive monocular images. The depth estimator using the encoder-decoder architecture generated dense depth maps. The UnDeepVO model was trained in an unsupervised way with spatial and temporal image losses. The spatial losses were constructed from the geometric constraints between stereo images and summarized three types of consistency losses: photometric, disparity, and pose. The temporal image losses minimized camera motion errors using two consecutive monocular images. The proposed UnDeepVO model was evaluated on the KITTI Visual Odometry dataset and compared with other SLAM-based models.

Aslan et al. [98] designed an end-to-end deep learning architecture to predict Unmanned Aerial System (UAS) position. In this system, inertial data were integrated with images, and their processing was carried out in three steps. The first step used CNN to extract spatial features from input images. In the second step, temporal features were extracted from the IMU data using a Bidirectional Long Short Term Memory (BiLSTM) network. In the third step, the UAS position was estimated by fusing both types of fea-

tures, visual and inertial, using a BiLSTM-based model. The proposed method was tested on the public EuRoC (European Robotics Challenge) dataset and the ROS-based simulation dataset.

Almalioglu et al. [99] proposed an end-to-end trainable deep visual–inertial architecture for pose estimation and depth mapping. A self-supervised deep learning-based VIO model called SelfVIO involved modules for depth generation, visual odometry, inertial odometry, visual–inertial fusion, spatial transformer, and target discrimination. Unlabeled image sequences as well as raw IMU measurements were fed to the deep network. The parameters of relative translation and rotation in consecutive frames were estimated with 6-DoF motion, while a depth image was created using a disparity map.

In [100], an end-to-end deep neural network was developed to evaluate the overlap between two underwater scenes to solve the Visual Loop Detection (VLD) problem for autonomous underwater vehicles. A Siamese CNN extracted the global image descriptors, and then the loop quantifier compared the outputs of the two branches. Various cases of loop representations were tested. The autonomous underwater vehicle was equipped with a bottom-looking camera and quantified how much the frames overlapped.

The deep reinforcement learning visual navigation method was proposed in [101]. The Markov network modeled the abstract map, and a knowledge graph initialized its structure, reducing the difficulty of model learning. The end-to-end learning process trained by a reinforcement learning method was utilized to update the abstract map. In addition to visual features, the proposed MGRL (Reinforcement Learning method combined with a Markov network and Graph) derived new graph relational features that measured the relative distance between an observation and a target, providing a global view of the environment through probability inference. This model was tested on the AI2-THOR dataset and in the physical environment.

Liu et al. [102] developed VGF-Net to fuse visual and geometric information for simultaneous drone navigation and height mapping. The goal was to create a better representation of outdoor/indoor scenes based on visual-geometric features extracted from RGB image sequences and an initial rough height map. In addition, a directional attention model selected essential object relationships using the object boundaries in the 2.5D height map and the extracted 3D keypoints. The standard SLAM module was used to refine the height map and 3D keypoints. Thus, deep networks can be used not only as internal modules of SLAM, but SLAM can also be part of the network.

4. Datasets for Visual SLAM

VSLAM methods based on deep learning models require large datasets with ground truth for optimizing parameters during supervised learning. However, to collect complex data including color, depth, LiDAR, inertial, GPS, and some specific data, developers need expensive equipment maintained on moving platforms such as cars, UAVs, underwater drones, etc. This problem limited the number of datasets in 2012–2014, and it is only since 2016 that large datasets have emerged for training various VSLAM methods based on deep learning. Sometimes simulated scenes, such as those created in Unity 3D, are used as datasets instead of real indoor and/or outdoor environments.

The ranking of applied datasets used in publications mentioned above is as follows: the KITTI Vision Benchmark Suite (28%), the TUM RGB-D SLAM Dataset and Benchmark (13%), own datasets (12%), the RGB-D dataset 7-Scenes (4.4%), the EuRoC MAV dataset (4.4%), the ICL-NUIM dataset (3%), the NYU RGB-D V2 dataset (3%), the Oxford RobotCar dataset (3%), the Malaga dataset (3%), the CityScapes dataset (3%), and the ApolloScape dataset (3%), among others. An ordered summary of commonly used and most interesting datasets from Table 5 is presented below.

Table 5. A summary of VSLAM datasets.

Dataset	Year	Subject	Sensor	Short Description
The KITTI Visual Odometry dataset	2012	22 stereo sequences	Color camera, GPS, LiDAR scanner	The dataset contains image pairs captured in outdoor scenes with ground truth labeling
The vKITTI2 dataset	2015	5 sequences	Unity game engine	The dataset is a synthetic dataset for training and testing autonomous driving models
The TUM RGB-D dataset	2012	39 sequences	Kinect sensor (near-infrared laser, infrared camera, color camera)	The data was recorded as color and depth frames with ground truth trajectories
The NYU RGB-D V2 dataset	2012	464 indoor scenes	Kinect RGB-D sensor	The dataset includes physical scenes for segmenting the visible areas of objects
The RGB-D Dataset 7-Scenes	2013	7 scenes	Kinect RGB-D sensor	The dataset allows observers to evaluate dense tracking and mapping, as well as relocalization
The Malaga dataset	2013	15 outdoor scenes	Stereo camera, 5 laser scanners, IMU, GPS	The dataset presents high-resolution stereo images over a 36.8 km trajectory
The ICL-NUIM dataset	2014	2 scenes	Kinect sensor	The dataset includes labeled scenes with a living room and an office
The EuRoC MAV dataset	2016	11 indoor scenes	Stereo camera in grey space, IMU	The dataset consists of stereo images synchronized with IMU measurements
The Oxford RobotCar dataset	2016	Over 130 scenes	6 cameras, LiDAR, GPS, INS	The dataset contains long trajectories in outdoor scenes with complex weather conditions
The Cytiscapes dataset	2016	50 urban scenes	Stereo camera, GPS	The dataset contains complex street scenes from 50 different cities
The AI2-THOR dataset	2017	120 indoor scenes	Physical simulation in Unity 3D	The dataset consists of photorealistic 3D indoor scenes with AI agent navigation
The ScanNet dataset	2017	707 scenes	Kinect sensor	The dataset is a set of annotated 3D indoor reconstructions
The ApolloScape dataset	2018	2 long sequences	Camera, Stereo camera, LiDAR, GPS, IMU	The dataset consists of varying conditions and traffic densities with complex scenarios
The MidAir dataset	2019	79 min of drone flight	3 RGB cameras, accelerometer, gyroscope, GPS	The dataset provides a large amount of synchronized data corresponding to flight records
The RIO10 dataset	2020	10 sequences	Mobile phone	The dataset provides RGB and depth images with semantic maps for reference

Created at the Karlsruhe Institute of Technology and Toyota Technological Institute at Chicago, the history of the KITTI Vision Benchmark Suite began in 2012 with the stereo, flow, and odometry benchmarks [105]. At that time, the raw data were collected [106]. Every year this benchmark has expanded its content, reaching the evaluation procedure for Tracking and MOTS (Multi-Object Tracking and Segmentation) in 2021 [107]. The

KITTI dataset includes such benchmarks as Stereo, Flow, SceneFlow, Depth, Odometry, Object, Tracking, Road, Semantics, and Raw data. For VSLAM, the odometry benchmark is suitable [108]. The KITTI Visual Odometry benchmark includes 22 stereo sequences with and without ground truth trajectories.

Virtual KITTI 2 (vKITTI2) is a more photo-realistic version with better quality related to the original virtual KITTI dataset [109]. It exploits recent improvements of the Unity game engine and provides new data such as stereo images and scene flow [110]. While Virtual KITTI included five driving video sequences cloned from the original KITTI dataset, Virtual KITTI 2 consists of the same five sequence clones as Virtual KITTI, but has such new features as increased photorealism, stereo cameras, and additional ground truth labeling.

The TUM RGB-D SLAM Dataset and Benchmark was created at the Technical University of Munich [111] in 2012. This dataset helps evaluate visual SLAM and odometry systems on RGB-D data containing both RGB-D and ground-truth data [112]. The data was recorded at full frame rate with resolution 640 $\times$ 480 by a Microsoft Kinect sensor in two different indoor scenes. Each sequence consists of color and depth images and also includes a ground truth trajectory obtained from the motion capture system. All data was calibrated and synchronized in time. To obtain ground-truth trajectories, a high-precision motion capture system was used, containing eight high-speed tracking cameras. The developers also proposed an evaluation criterion for measuring the quality of the camera trajectory of visual SLAM systems.

The RGB-D dataset 7-Scenes presented in 2013 by Microsoft is an indoor dataset, which included seven scenes captured by a Kinect RGB-D sensor [113]. This dataset is a collection of RGB-D frames from each of seven scenes. Each scene is a specific office room with many textureless regions. The ground truth camera poses are obtained by applying the KinectFusion system [114]. The dataset may be used for evaluation of dense tracking and mapping, as well as relocalization methods. Each sequence contains 500–1000 frames, and each frame consists of three files with color, depth, and pose information. Each scene has a full description for evaluation and dense reconstruction. However, the RGB and depth camera have not been calibrated.

The EuRoC MAV dataset is provided by Swiss Federal Institute of Technology and Autonomous Systems Lab. This visual-inertial data were collected using Micro Aerial Vehicle (MAV) on-board devices [115]. The EuRoC (European Robotics Challenge) MAV dataset [116] publicly shared by ETH is often used in VIO applications. Grayscale images were acquired from a stereo camera, while the simultaneous accelerometer and gyroscope data were captured by the on-board IMU sensor. For VIO calibration, the intrinsic parameters of the camera and IMU sensor were included in the dataset. The dataset was collected in an outdoor environment with various obstacles which contained light, medium, and complex scenes depending on the MAV speed, brightness, and blur.

The ICL-NUIM dataset from Imperial College, London, UK was collected for comparative analysis of RGB-D, VO, and SLAM algorithms [117]. Both scenes with living and office rooms are labeled with ground truth data [118]. The 3D ground truth surfaces, depth map, and camera poses were created for the living room. Such data can be used not only to build the camera trajectory but also to reconstruct the scene. The office room does not have a 3D model and only provides trajectory data. An additional synthetic dataset contains the images obtained from ray-traced 3D models. All data is compatible with the evaluation tools available for the TUM RGB-D dataset.

The NYU RGB-D V2 is a dataset for understanding indoor scenes [119]. Its previous version called the NYU Depth V1 dataset contained a limited number of scenes and scene types plus unlabeled and labeled frames [120]. The NYU RGB-D V2 dataset involves 464 different indoor scenes belonging to 26 scene classes (total 1449 RGB-D images) that were captured from commercial and residential buildings in three cities in the USA. A dense per-pixel labeling in each image was performed using the Amazon Mechanical Turk tool. For scene understanding, each instance in the scene was labeled with a unique label. Thus, the dataset contains 35,064 objects, belonging to 894 classes. Each of the 1449 RGB-D

images was manually annotated with a predefined vector describing relationships between objects. There are also 407,024 unlabeled frames.

The Oxford RobotCar dataset contains over 1000 km of recorded videos captured from six on-board vehicle cameras, as well as LiDAR, GPS, and INS objective data [121]. This dataset contains much longer trajectories with complex dynamic outdoor environments and meteorological conditions, including rain, snow, direct sunlight, and night, [122]. In total, the dataset contains over 20 TB of imagery.

The Malaga dataset was collected in an urban environment using a special car, which had eight sensors, including one stereo camera, five laser scanners, one IMU, and one GPS receiver [123]. The first online version appeared in 2013. In 2013, this dataset was a collection of high-resolution stereo images captured during a 36.8 km trajectory. Raw (unprocessed) images had a resolution of 1024 $\times$ 768. A single sequence had a duration of 6–93 min [124]. The developers have created a video index (as an additional tool) available online to select relevant segments from the dataset. In addition to the images, a 3D point cloud reconstruction of the scenes was created as a video sequence based on laser scanners and GPS data.

The Cityscapes dataset involves highly complex street scenes that were recorded in fifty cities, mostly in Germany, with thirty classes grouped into eight categories: flat, construction, nature, vehicle, sky, object, human, and void [125]. 5000 of these images have high quality pixel-level annotations. 20,000 additional images have coarse annotations to evaluate methods for processing weakly labeled data [126]. Semantic labeling is available at the pixel-level and instance-level. The Cityscapes dataset significantly exceeds previous datasets in terms of volume, annotation quality, and scene complexity. The densely annotated images can be used for training, validation, and testing, while the annotated images only serve for additional training. Each of the training, validation, and testing sets was ordered according to the territory size, the geographic direction across parallels and meridians, and the season period.

The ApolloScape dataset includes many subsets, such as Scene Parsing, Car Instance, Lane Segmentation, Self-Localization, Trajectory, Detection/Tracking, Stereo, and Inpainting [127]. Thus, in April 2018, the Scene Parsing dataset contains 140 K frames at a resolution of 3384 $\times$ 2710 with corresponding pixel-level annotations and pose information, as well as depth maps for static background. It is expected that this dataset will include 1 MB frames with corresponding pixel-level annotations and pose information. Herewith, the number of moving objects on average ranges from tens to more than a hundred [128]. Each image is also labeled with centimeter-accurate pose information, and the background point cloud has millimeter-accuracy. The 25 classes were divided into five groups: movable object, surface, infrastructure, nature, and void. A labeling pipeline includes 3D labeling and 2D labeling for handling static background/objects and moving objects, respectively, using CNN. The dataset is divided into training, validation, and testing subsets. The semantic annotations for test images are not provided.

The ScanNet dataset is a large RGB-D video dataset. It contains over 707 unique indoor environments, annotated with 3D camera poses, surface reconstructions, and instance-level segmentations [129]. The surface reconstruction and semantic annotation were performed automatically using a scalable RGB-D system. A CAD model was also provided as a subset of scans [130].

The Mid-Air dataset was created at the Montefiore Institute [131] as a multi-purpose synthetic dataset containing low altitude drone flights data. The dataset consists of 79 min of records that were manually extracted from over 5 h of flight records [132]. These records are divided into 54 separate trajectories of the same length. Since each trajectory has been rendered multiple times for different weather conditions, Mid-Air contains over 420,000 separate training frames. Data received from multi-modal vision sensors and navigation sensors were synchronized. Multi-modal vision sensors provide RGB images, stereo disparity, scene depth, and even object semantics.

The AI2-THOR dataset is an open-source interactive environment for embodied AI [133]. THOR means The House of Interactions for visual AI research. The AI2-THOR dataset consists of photo-realistic 3D indoor scenes with AI agent navigation to complete tasks [134]. It can be used in many fields such as planning, imitation learning, learning by interaction, visual question answering, and so on. This dataset visualizes 120 rooms, including kitchens, bedrooms, bathrooms, and living rooms, with over 2000 unique objects. Based on Unity 3D, the AI2-THOR dataset enables physical simulation for objects and scenes.

The RIO10 dataset [135] includes 10 RGB-D video sequences of indoor scenes captured by a mobile phone. Each scene was captured several times throughout the year. The ground truth camera poses were obtained using an off-line package setup system. The dataset also includes semantic maps. However, the application of this dataset to the relocalization task is very difficult due to dynamic objects, blur, and various lighting conditions captured by a mobile phone.

It should be noted that many datasets do not have a standalone website but are located on the GitHub website.

5. Discussion and Future Trends

This survey presents the current advances in the field of deep-based VSLAM methods since 2017, focusing on two aspects. First, high-quality studies show how deep learning paradigms help to solve VSLAM tasks and even change traditional VSLAM problem statements. Second, the new approaches proposed in the articles mentioned above open up great perspectives for future investigations. Every year, deep learning models improve their performance and demonstrate new capabilities for solving more and more complex problems. Obviously, the implementation of deep learning methods in VSLAM is currently far from desirable, but the first steps are very promising.

Recently, three ways to develop deep learning-based VSLAM software components encompassing auxiliary modules, original deep learning modules, and end-to-end deep neural networks have been identified with different degrees of implementation. A way to develop auxiliary deep-based modules introduces most of the published studies including feature extraction [48–50], semantic segmentation [51–60], pose estimation [8,45,46,61–63], map construction [3,64–66], and loop closure [67–70]. It should be noted that deep neural networks extract low-level features from images by converting them to high-level featureslayer by layer. Thus, deep learning "changes" the term "feature extraction" from conventional keypoints extraction to complex tasks, such as matching keypoints of a 2D image and 3D LiDAR points [48], keypoints extraction from an optical flow [49], extraction of image patches using the famous ORB-SLAM algorithm [50], etc. Semantic segmentation seems to be a more explored area, with semantic filtering [51,52], object detection followed by semantic segmentation in static and dynamic environments [55–57], and scene representation [58,59] being the main approaches. Deep learning-based pose estimation is a wide area of study in many scientific fields, but only a few approaches have been implemented in VSLAM systems related to VO tasks [8,45,62], ego-motion of camera [46,61], and low illumination conditions [63]. Currently, map construction is not well explored by deep learning paradigms and is presented by several attempts to incorporate optical flow networks, RNNs, and stereo vision into validated SLAM systems. Better results are achieved by combining depth, LiDAR, and optical data. Auxiliary modules in the loop closure eliminate the influence of moving objects in the scene [67,70] and extract keyframes for searching the correct trajectory [68,69].

There are several studies devoted to the development of original deep learning modules for camera relocalization [82], distance estimation [83], object segmentation [84,85], path planning [86], and scene reconstruction [87]. The architecture of original deep learning modules becomes more complex when multiple deep neural network models are used in serial or parallel pipelines with RNNs or GANs. It was shown in [84] that the accuracy and robustness of the proposed DDL-SLAM model outperforms the indicators of the ORB-

SLAM2 model in highly-dynamic scenarios. At the same time, the DDL-SLAM model has several limitations in real-time performance and scene inpainting.

Obviously, the development of end-to-end deep neural networks is the most promising approach for VSLAM systems due to self-supervised learning and reinforcement learning as the basis of high adaptive ability to a real dynamic environment. Interesting experimental results were obtained in VO/VIO [96–99] and ego-motion tasks [94,95]. Sometimes traditional methods such as Kalman filter [47] or the Savitzky-Golay filter [91] are combined with end-to-end deep models that provide improved results. This is the so-called hybrid approach. Some end-to-end deep neural networks have original applications, for example, in surgery [90], UAV pose estimation [91,98], autonomous underwater vehicles [100], drone navigation, and height mapping [102], etc.

It should be noted that the implementation of deep learning in VSALM systems is a very complex process. However, impressive results have recently been achieved. Thus, the Absolute Trajectory Error (ATE) metrics with and without auxiliary deep modules improve values by 50 times [57], and depth reconstruction estimates in terms of time and accuracy are better using the DRM-SLAM model [65], as well as precision–recall results for different datasets in the loop closure problem [70].

Application of deep models in VSLAM partially changes the problem statements, challenges, and directions of research. For example, scene mapping tends to move from 2D to 3D analysis, from sparse to dense maps, and from topological to semantic representation. Let us summarize the challenges and future trends in VSLAM in theoretical and practical aspects.

1. **Complex Types of Unsupervised Learning**: Currently, supervised learning prevails with the need to have access to large labeled datasets. Unsupervised learning is more preferable from a practical point of view. However, it provides less precision, which is of great importance for many near-photogrammetric VSLAM tasks. It is evident that meta-learning, reinforcement learning, and life-long learning, as well as the development of efficient architectures, will make it possible to compensate for the inherent properties of unsupervised learning.
2. **Robustness in Challenging Scenes**: The robustness of algorithms plays an important role in practical applications. Deep learning models well-trained on datasets need to be robust to sensor noise, lighting, weather conditions, and complex scenarios when used in real environments. Thus, the ego-motion and the motion of objects must be accurately estimated. At the same time, the performance of some sensors, such as LiDAR and RADAR sensors, is low in extreme meteorological conditions. Conventional VSLAM systems based on stable visual landmarks are of limited use and currently cannot be considered an effective solution. Another problem is failure in situations with fast movement. Some algorithms are tested on multiple datasets, including their own datasets. However, the problem of generalization is far from its sufficient solution.
3. **Real-time Deployment**: Real-time implementation is one of the main factors positively influencing the practical development of VSALM for autonomous systems. At present, most deep learning architectures tend to use complex modules, which increase computational calculations and costs. Nevertheless, it is difficult to expect in the near future that light-weight neural solutions will be acceptable for such complex problems as VSLAM provides. Thus, we can talk about a huge gap between simulation environments and real scenes.
4. **Multi-task and Multi-modal Architectures**: Deep networks that use multi-task and multi-modal paradigms are attracting particular attention. Although datasets typically provide multi-modal data from sensors such as color camera, stereo camera, event-based camera, IMU, LiDAR, etc., deep learning networks cannot fuse or process all types of data directly due to their inherent limitations. Data fusion remains an open issue not only for VSLAM problems, but for many others.

5. **Dynamic Environment**: The first VSLAM and traditional SLAM methods focused on a static environment and failed in a dynamic environment. Only recently developed end-to-end deep networks can be implemented as real time systems in a dynamic environment, primarily due to the high performance of trackers with recognition functions. Moreover, some solutions involve not only semantic but also instance segmentation. Dynamic scenarios with dynamic objects are typical environments for autonomous systems (road networks, highways, etc.).
6. **Navigation Control**: Path planning and navigation control are the core modules for autonomous vehicles that affect safety. Many VSLAM algorithms lack an efficient control technique for navigation. Navigation control is highly dependent on the state of the environment and the performance of the algorithms. This issue requires future work for fully autonomous vehicles.

6. Conclusions

VSLAM as a rapidly developing scientific field attracts a lot of attention from researchers who develop and/or use deep learning models. The latest deep learning techniques objectively improve VSLAM steps including data processing, pose estimation, trajectory estimation, mapping, and loop closure. This detailed overview describes three main ways of interaction. VSLAM methods with auxiliary modules based on deep learning prevail, since it is a relatively simple introduction of different deep neural networks. Replacing the original modules of traditional VSLAM with deep learning modules is the least numerous group of methods due to the difficulty of merging traditional VSLAM methods and deep learning techniques. Replacing a traditional VSLAM system with end-to-end deep neural networks is the most perspective way, suggesting new paradigms in VSLAM. However, deep models have fundamental limitations. For example, they cannot process inertial data in combination with color, depth, and LiDAR data. This means that extensive research needs to be undertaken in the future. Generally speaking, deep learning models offer opportunities for efficient processing of visual data in real time, but at the same time have limitations in data fusion obtained from different types of sensors at the current stage of technology development.

Another open issue is a preferable type of learning—supervised or unsupervised. At the same time, reinforcement learning, perhaps the most acceptable for VSLAM, is in its early use. This survey does not address these theoretical issues, as well as the issues of inter-cross and intra-cross validation. A few datasets collected since 2012 are also discussed. It should be noted that most datasets evolve from year to year, providing researchers with large multi-modal data. The complexity and diversity of data is increasing. Finally, challenges and future trends in VSLAM based on deep learning are summarized, indicating promising research directions.

Funding: This research received no external funding.

Data Availability Statement: No new data were created in this study. Data sharing is not applicable to this article.

Conflicts of Interest: The author declares no conflict of interest.

References

1. Palomeras, N.; Carreras, M.; Andrade-Cetto, J. Active SLAM for autonomous underwater exploration. *Remote Sens.* **2019**, *11*, 2827. [CrossRef]
2. Fang, B.; Mei, G.; Yuan, X.; Wang, L.; Wang, Z.; Wang, J. Visual SLAM for robot navigation in healthcare facility. *Pattern Recognit.* **2021**, *113*, 107822. [CrossRef] [PubMed]
3. Chen, M.; Tang, Y.; Zou, X.; Huang, Z.; Zhou, H.; Chen, S. 3D global mapping of large-scale unstructured orchard integrating eye-in-hand stereo vision and SLAM. *Comput. Electron. Agric.* **2021**, *187*, 106237. [CrossRef]
4. Ouyang, M.; Shi, X.; Wang, Y.; Tian, Y.; Shen, Y.; Wang, D.; Wang, P.; Cao, Z. A collaborative visual SLAM framework for service robots. In Proceedings of the 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2021), Prague, Czech Republic, 27 September–1 October 2021; pp. 8679–8685. [CrossRef]

5. Kuo, C.-Y.; Huang, C.-C.; Tsai, C.-H.; Shi, Y.-S.; Smith, S. Development of an immersive SLAM-based VR system for teleoperation of a mobile manipulator in an unknown environment. *Comput. Ind.* **2021**, *132*, 103502. [CrossRef]
6. Li, W.; Wang, J.; Liu, M.; Zhao, S. Real-time occlusion handling for augmented reality assistance assembly systems with monocular images. *J. Manuf. Syst.* **2022**, *62*, 561–574. [CrossRef]
7. Sucar, E.; Liu, S.; Ortiz, J.; Davison, A.J. iMAP: Implicit mapping and positioning in real-time. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, QC, Canada, 11–17 October 2021; pp. 6229–6238.
8. Qia, C.; Xiang, Z.; Wang, X.; Chen, S.; Fan, Y.; Zhao, X. Objects matter: Learning object relation graph for robust absolute pose. *Neurocomputing* **2023**, *521*, 11–26. [CrossRef]
9. Davison, A.J.; Reid, I.D.; Molton, N.D.; Stasse, O. MonoSLAM: Real-time single camera SLAM. *IEEE Trans. Pattern Anal. Mach. Intell.* **2007**, *29*, 1052–1067. [CrossRef]
10. Klein, G.; Murray, D. Parallel tracking and mapping for small AR workspaces. In Proceeding of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, Nara, Japan, 13–16 November 2007; pp. 1–10. [CrossRef]
11. Zou, D.P.; Tan, P. CoSLAM: Collaborative visual SLAM in dynamic environments. *IEEE Trans. Pattern Anal. Mach. Intell.* **2013**, *35*, 354–366. [CrossRef]
12. Mur-Artal, R.; Montiel, J.M.M.; Tardos, J.D. ORB-SLAM: A versatile and accurate monocular SLAM system. *IEEE Trans. Robot.* **2015**, *31*, 1147–1163. [CrossRef]
13. Liu, H.M.; Zhang, G.F.; Bao, H.J. Robust keyframe-based monocular SLAM for augmented reality. In Proceedings of the 2016 IEEE International Symposium on Mixed and Augmented Reality (ISMAR-Adjunct), Merida, Mexico, 19–23 September 2016; pp. 1–10. [CrossRef]
14. Campos, C.; Elvira, R.; Rodrıguez, J.J.G.; Montiel, J.M.; Tardos, J.D. ORB-SLAM3: An accurate open-source library for visual, visual–inertial, and multimap SLAM. *IEEE Trans. Robot.* **2021**, *37*, 1874–1890. [CrossRef]
15. Forster, C.; Zhang, Z.C.; Gassner, M.; Werlberger, M.; Scaramuzza, D. SVO: Semi-direct visual odometry for monocular and multicamera systems. *IEEE Trans. Robot.* **2017**, *33*, 249–265. [CrossRef]
16. Qin, T.; Li, P.L.; Shen, S.J. VINS-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Trans. Robot.* **2018**, *34*, 1004–1020. [CrossRef]
17. Zou, D.P.; Wu, Y.X.; Pei, L.; Ling, H.B.; Yu, W.X. StructVIO: Visual-inertial odometry with structural regularity of man-made environments. *IEEE Trans. Robot.* **2019**, *35*, 999–1013. [CrossRef]
18. Sun, Y.; Liu, M.; Meng, M.Q.-H. Motion removal for reliable RGB-D SLAM in dynamic environments. *Robot. Auton. Syst.* **2018**, *108*, 115–128. [CrossRef]
19. Li, S.; Zhang, D.; Xian, Y.; Li, B.; Zhang, T.; Zhong, C. Overview of deep learning application on visual SLAM. *Displays* **2022**, *74*, 102298. [CrossRef]
20. Younes, G.; Asmar, D.; Shammas, E.; Zelek, J. Keyframe-based monocular SLAM: Design, survey, and future directions. *Robot. Auton. Syst.* **2017**, *98*, 67–88. [CrossRef]
21. Li, J.; Yang, B.; Chen, D.; Wang, N.; Zhang, G.; Bao, H. Survey and evaluation of monocular visual-inertial SLAM algorithms for augmented reality. *Virtual Real. Intell. Hardw.* **2019**, *1*, 386–410. [CrossRef]
22. Zou, D.; Tan, P.; Yu, W. Collaborative visual SLAM for multiple agents: A brief survey. *Virtual Real. Intell. Hardw.* **2019**, *1*, 461–482. [CrossRef]
23. Cebollada, S.; Payá, L.; Flores, M.; Peidró, A.; Reinoso, O. A state-of-the-art review on mobile robotics tasks using artificial intelligence and visual data. *Expert Syst. Appl.* **2021**, *167*, 114195. [CrossRef]
24. Taheri, H.; Xia, Z.C. SLAM; definition and evolution. *Eng. Appl. Artif. Intell.* **2021**, *97*, 104032. [CrossRef]
25. Arshad, S.; Kim, G.-W. Role of deep learning in loop closure detection for visual and LiDAR SLAM: A survey. *Sensors* **2021**, *21*, 1243. [CrossRef] [PubMed]
26. Cheng, J.; Zhang, L.; Chen, Q.; Hu, X.; Cai, J. A review of visual SLAM methods for autonomous driving vehicles. *Eng. Appl. Artif. Intell.* **2022**, *114*, 104992. [CrossRef]
27. Bala, J.A.; Adeshina, S.A.; Aibinu, A.M. Advances in visual simultaneous localisation and mapping techniques for autonomous vehicles: A review. *Sensors* **2022**, *22*, 8943. [CrossRef] [PubMed]
28. Kazerouni, I.A.; Fitzgerald, L.; Dooly, G.; Toal, D. A survey of state-of-the-art on visual SLAM. *Expert Syst. Appl.* **2022**, *205*, 117734. [CrossRef]
29. Theodorou, C.; Velisavljevic, V.; Dyo, V.; Nonyelu, F. Visual SLAM algorithms and their application for AR, mapping, localization and wayfinding. *Array* **2022**, *15*, 100222. [CrossRef]
30. Macario Barros, A.; Michel, M.; Moline, Y.; Corre, G.; Carrel, F. A comprehensive survey of visual SLAM algorithms. *Robotics* **2022**, *11*, 24. [CrossRef]
31. Chen, W.; Shang, G.; Ji, A.; Zhou, C.; Wang, X.; Xu, C.; Li, Z.; Hu, K. An overview on visual SLAM: From tradition to semantic. *Remote Sens.* **2022**, *14*, 3010. [CrossRef]
32. Tang, Y.; Zhao, C.; Wang, J.; Zhang, C.; Sun, Q.; Zheng, W.X.; Du, W.; Qian, F.; Kurths, J. Perception and navigation in autonomous systems in the era of learning: A survey. *IEEE Trans. Neural Netw. Learn. Syst.* 2022, *in press*. [CrossRef]
33. Wang, K.; Ma, S.; Chen, J.; Ren, F.; Lu, J. Approaches, challenges, and applications for deep visual odometry: Toward complicated and emerging areas. *IEEE Trans. Cogn. Dev. Syst.* **2022**, *14*, 35–49. [CrossRef]

34. Syed, T.A.; Siddiqui, M.S.; Abdullah, H.B.; Jan, S.; Namoun, A.; Alzahrani, A.; Nadeem, A.; Alkhodre, A.B. In-depth review of augmented reality: Tracking technologies, development tools, AR displays, collaborative AR, and security concerns. *Sensors* **2023**, *23*, 146. [CrossRef]
35. Eswaran, M.; Gulivindala, A.K.; Inkulu, A.K.; Bahubalendruni, R.M.V.A. Augmented reality-based guidance in product assembly and maintenance/repair perspective: A state of the art review on challenges and opportunities. *Expert Syst. Appl.* **2023**, *213*, 118983. [CrossRef]
36. Zhang, J.; Yang, X.; Wang, W.; Guan, J.; Ding, L.; Lee, V.C.S. Automated guided vehicles and autonomous mobile robots for recognition and tracking in civil engineering. *Autom. Constr.* **2023**, *146*, 104699. [CrossRef]
37. Martinelli, F.; Mattogno, S.; Romanelli, F. A resilient solution to Range-Only SLAM based on a decoupled landmark range and bearing reconstruction. *Robot. Auton. Syst.* **2023**, *160*, 104324. [CrossRef]
38. Ila, V.; Porta, J.M.; Andrade-Cetto, J. Amortized constant time state estimation in Pose SLAM and hierarchical SLAM using a mixed Kalman-information filter. *Robot. Auton. Syst.* **2011**, *59*, 310–318. [CrossRef]
39. Bonetto, E.; Goldschmid, P.; Pabst, M.; Black, M.J.; Ahmad, A. iRotate: Active visual SLAM for omnidirectional robots. *Robot. Auton. Syst.* **2022**, *154*, 104102. [CrossRef]
40. Xie, H.; Chen, W.; Wang, J. Hierarchical forest based fast online loop closure for low-latency consistent visual-inertial SLAM. *Robot. Auton. Syst.* **2022**, *151*, 104035. [CrossRef]
41. Lee, S.J.; Choi, H.; Hwang, S.S. Real-time depth estimation using recurrent CNN with sparse depth cues for SLAM system. *Int. J. Control Autom. Syst.* **2020**, *18*, 206–216. [CrossRef]
42. Soares, J.C.V.; Gattass, M.; Meggiolaro, M.A. Crowd-SLAM: Visual SLAM towards crowded environments using object detection. *J. Intell. Robot. Syst.* **2021**, *102*, 50. [CrossRef]
43. Liu, Y.; Miura, J. RDS-SLAM: Real-time dynamic SLAM using semantic segmentation methods. *IEEE Access* **2021**, *9*, 23772–23785. [CrossRef]
44. Mur-Artal, R.; Tardós, J.D. ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras. *IEEE Trans. Robot.* **2017**, *33*, 1255–1262. [CrossRef]
45. Zhu, Z.; Wang, J.; Xu, M.; Lin, S.; Chen, Z. InterpolationSLAM: An effective visual SLAM system based on interpolation network. *Eng. Appl. Artif. Intell.* **2022**, *115*, 105333. [CrossRef]
46. Song, C.; Niu, M.; Liu, Z.; Cheng, J.; Wang, P.; Li, H.; Hao, L. Spatial-temporal 3D dependency matching with self-supervised deep learning for monocular visual sensing. *Neurocomputing* **2022**, *481*, 11–21. [CrossRef]
47. Xiu, H.; Liang, Y.; Zeng, H.; Li, Q.; Liu, H.; Fan, B.; Li, C. Robust self-supervised monocular visual odometry based on prediction-update pose estimation network. *Eng. Appl. Artif. Intell.* **2022**, *116*, 105481. [CrossRef]
48. Feng, M.; Hu, S.; Ang, M.H.; Lee, G.H. 2D3D-MatchNet: Learning to match keypoints across 2D image and 3D point cloud. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA 2019), Montreal, QC, Canada, 20–24 May 2019; pp. 4790–4796.
49. Qin, Z.; Yin, M.; Li, G.; Yang, F. SP-Flow: Self-supervised optical flow correspondence point prediction for real-time SLAM. *Comput. Aided Geom. Des.* **2020**, *82*, 101928. [CrossRef]
50. Bruno, H.M.S.; Colombini, E.L. LIFT-SLAM: A deep-learning feature-based monocular visual SLAM method. *Neurocomputing* **2021**, *455*, 97–110. [CrossRef]
51. Kaneko, M.; Iwami, K.; Ogawa, T.; Yamasaki, T.; Aizawa, K. Mask-SLAM: Robust feature-based monocular SLAM by masking using semantic segmentation. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW 2018), Salt Lake City, UT, USA, 18–22 June 2018; pp. 371–379.
52. Shao, C.; Zhang, C.; Fang, Z.; Yang, G. A deep learning-based semantic filter for RANSAC-based fundamental matrix calculation and the ORB-SLAM system. *IEEE Access* **2019**, *8*, 3212–3223. [CrossRef]
53. Tian, G.; Liu, L.; Ri, J.H.; Liu, Y.; Sun, Y. ObjectFusion: An object detection and segmentation framework with RGB-D SLAM and convolutional neural networks. *Neurocomputing* **2019**, *345*, 3–14. [CrossRef]
54. Xu, L.; Feng, C.; Kamat, V.R.; Menassa, C.C. A scene-adaptive descriptor for visual SLAM-based locating applications in built environments. *Autom. Constr.* **2020**, *112*, 103067. [CrossRef]
55. Liu, W.; Mo, Y.; Jiao, J.; Deng, Z. EF-Razor: An effective edge-feature processing method in visual SLAM. *IEEE Access* **2020**, *8*, 140798–140805. [CrossRef]
56. Rusli, I.; Trilaksono, B.R.; Adiprawita, W. RoomSLAM: Simultaneous localization and mapping with objects and indoor layout structure. *IEEE Access* **2020**, *8*, 196992–197004. [CrossRef]
57. Jin, S.; Chen, L.; Sun, R.; McLoone, S. A novel vSLAM framework with unsupervised semantic segmentation based on adversarial transfer learning. *Appl. Soft Comput. J.* **2020**, *90*, 106153. [CrossRef]
58. Wu, J.; Shi, Q.; Lu, Q.; Liu, X.; Zhu, X.; Lin, Z. Learning invariant semantic representation for long-term robust visual localization. *Eng. Appl. Artif. Intell.* **2022**, *111*, 104793. [CrossRef]
59. Zhao, X.; Wang, C.; Ang, M.H. Real-time visual-inertial localization using semantic segmentation towards dynamic environments. *IEEE Access* **2020**, *8*, 155047–155059. [CrossRef]
60. Su, P.; Luo, S.; Huang, X. Real-time dynamic SLAM algorithm based on deep learning. *IEEE Access* **2022**, *10*, 87754–87766. [CrossRef]

61. Zou, Z.-X.; Huang, S.-S.; Mu, T.-J.; Wang, Y.-P. ObjectFusion: Accurate object-level SLAM with neural object priors. *Graph. Model.* **2022**, *123*, 101165. [CrossRef]
62. Mumuni, F.; Mumuni, A.; Amuzuvi, C.K. Deep learning of monocular depth, optical flow and ego-motion with geometric guidance for UAV navigation in dynamic environments. *Mach. Learn. Appl.* **2022**, *10*, 100416. [CrossRef]
63. Li, Q.; Cao, R.; Zhu, J.; Fu, H.; Zhou, B.; Fang, X.; Jia, S.; Zhang, S.; Liu, K.; Li, Q. Learn then match: A fast coarse-to-fine depth image-based indoor localization framework for dark environments via deep learning and keypoint-based geometry alignment. *ISPRS J. Photogramm. Remote Sens.* **2023**, *195*, 169–177. [CrossRef]
64. Zhao, C.; Sun, L.; Yan, Z.; Neumann, G.; Duckett, T.; Stolkin, R. Learning Kalman Network: A deep monocular visual odometry for on-road driving. *Robot. Auton. Syst.* **2019**, *121*, 103234. [CrossRef]
65. Ye, X.; Ji, X.; Sun, B.; Chen, S.; Wang, Z.; Li, H. DRM-SLAM: Towards dense reconstruction of monocular SLAM with scene depth fusion. *Neurocomputing* **2020**, *396*, 76–91. [CrossRef]
66. Tao, C.; Gao, Z.; Yan, J.; Li, C.; Cui, G. Indoor 3D semantic robot VSLAM based on mask regional convolutional neural network. *IEEE Access* **2020**, *8*, 52906. [CrossRef]
67. Memon, A.R.; Wang, H.; Hussain, A. Loop closure detection using supervised and unsupervised deep neural networks for monocular SLAM systems. *Robot. Auton. Syst.* **2020**, *126*, 103470. [CrossRef]
68. Chang, J.; Dong, N.; Li, D.; Qin, M. Triplet loss based metric learning for closed loop detection in VSLAM system. *Expert Syst. Appl.* **2021**, *185*, 115646. [CrossRef]
69. Duan, R.; Feng, Y.; Wen, C.-Y. Deep pose graph-matching-based loop closure detection for semantic visual SLAM. *Sustainability* **2022**, *14*, 11864. [CrossRef]
70. Osman, H.; Darwish, N.; Bayoumi, A. PlaceNet: A multi-scale semantic-aware model for visual loop closure. *Eng. Appl. Artif. Intell.* **2023**, *119*, 105797. [CrossRef]
71. Leonardi, M.; Fiori, L.; Stahl, A. Deep learning based keypoint rejection system for underwater visual ego-motion estimation. *IFAC-Pap.* **2020**, *53*, 9471–9477. [CrossRef]
72. Yi, K.M.; Trulls, E.; Lepetit, V.; Fua, P. LIFT: Learned invariant feature transform. In *Computer Vision–ECCV 2016*; LNCS; Leibe, B., Matas, J., Sebe, N., Welling, M., Eds.; Springer: Cham, Switzerland, 2016; Volume 9910, pp. 467–483. [CrossRef]
73. Dosovitskiy, A.; Ros, G.; Codevilla, F.; Lopez, A.; Koltun, V. CARLA: An open urban driving simulator. In Proceedings of the 1st Annual Conference on Robot Learning, Mountain View, CA, USA, 13–15 November 2017; pp. 1–16.
74. Chen, L.C.; Zhu, Y.; Papandreou, G.; Schroff, F.; Adam, H. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Computer Vision–ECCV 2018*; LNCS; Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y., Eds.; Springer: Cham, Switzerland, 2018; Volume 11211, pp. 833–851. [CrossRef]
75. Deng, C.; Qiu, K.; Xiong, R.; Zhou, C. Comparative study of deep learning based features in SLAM. In Proceedings of the 2019 4th Asia-Pacific Conference on Intelligent Robot Systems (ACIRS 2019), Nagoya, Japan, 13–15 July 2019; pp. 250–254. [CrossRef]
76. Engel, J.; Koltun, V.; Cremers, D. Direct sparse odometry. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *40*, 611–625. [CrossRef]
77. Tian, Z.; Shen, C.; Chen, H.; He, T. FCOS: Fully convolutional one-stage object detection. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Republic of Korea, 27 October–2 November 2019; pp. 9627–9636.
78. Kendall, A.; Grimes, M.; Cipolla, R. PoseNet: A convolutional network for realtime 6-DoF camera relocalization. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 2938–2946.
79. Brahmbhatt, S.; Gu, J.; Kim, K.; Hays, J.; Kautz, J. MapNet: Geometry-aware learning of maps for camera localization. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 12616–12625.
80. Tan, M.; Pang, R.; Le, Q.V. EfficientDet: Scalable and efficient object detection. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2020), Seattle, WA, USA, 13–19 June 2020; pp. 10781–10790.
81. Efe, U.; Ince, K.G.; Alatan, A. DFM: A performance baseline for deep feature matching. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2021) Workshops, Nashville, TN, USA, 20–25 June 2021; pp. 4284–4293.
82. Wu, J.; Ma, L.; Hu, X. Delving deeper into convolutional neural networks for camera relocalization. In Proceedings of the IEEE International Conference on Robotics and Automation, Singapore, 29 May–3 June 2017; pp. 5644–5651. [CrossRef]
83. Kreuzig, R.; Ochs, M.; Mester, R. DistanceNet: Estimating traveled distance from monocular images using a recurrent convolutional neural network. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW 2019), Long Beach, CA, USA, 16–17 June 2019; pp. 1–9.
84. Ai, Y.; Rui, T.; Lu, M.; Fu, L.; Liu, S.; Wang, S. DDL-SLAM: A robust RGB-D SLAM in dynamic environments combined with deep learning. *IEEE Access* **2020**, *8*, 162335–162342. [CrossRef]
85. Han, S.; Xi, Z. Dynamic scene semantics SLAM based on semantic segmentation. *IEEE Access* **2020**, *8*, 43563–43570. [CrossRef]
86. Mishra, P.; Jain, U.; Choudhury, S.; Singh, S.; Pandey, A.; Sharma, A.; Singh, R.; Pathak, V.K.; Saxena, K.K.; Gehlot, A. Footstep planning of humanoid robot in ROS environment using Generative Adversarial Networks (GANs) deep learning. *Robot. Auton. Syst.* **2022**, *158*, 104269. [CrossRef]
87. Tu, X.; Xu, C.; Liu, S.; Xie, G.; Huang, J.; Li, R.; Yuan, J. Learning depth for scene reconstruction using an encoder-decoder model. *IEEE Access* **2020**, *8*, 89300–89317. [CrossRef]

88. Jin, Q.; Meng, Z.; Pham, T.D.; Chen, Q.; Wei, L.; Su, R. DUNet: A deformable network for retinal vessel segmentation. *Knowl. Based Syst.* **2019**, *178*, 149–162. [CrossRef]
89. Zhao, H.; Shi, J.; Qi, X.; Wangh, X.; Jia, J. Pyramid scene parsing network. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW 2017), Honolulu, HI, USA, 21–26 July 2017; pp. 2881–2890.
90. Lan, E. A novel deep learning architecture by integrating visual simultaneous localization and mapping (VSLAM) into CNN for real-time surgical video analysis. In Proceedings of the 2022 IEEE 19th International Symposium on Biomedical Imaging (ISBI 2022), Kolkata, India, 28–31 March 2022; pp. 1–5. [CrossRef]
91. Aslan, M.F.; Durdu, A.; Sabanci, K. Visual-Inertial Image-Odometry Network (VIIONet): A Gaussian process regression-based deep architecture proposal for UAV pose estimation. *Measurement* **2022**, *194*, 111030. [CrossRef]
92. Shamwell, J.E.; Leung, S.; Nothwang, W.D. Vision-aided absolute trajectory estimation using an unsupervised deep network with online error correction. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2018), Madrid, Spain, 1–5 October 2018; pp. 1–9. [CrossRef]
93. Xiao, Y.; Li, L.; Li, X.; Yao, J. DeepMLE: A robust deep maximum likelihood estimator for two-view structure from motion. In Proceedings of the 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2022), Kyoto, Japan, 23–27 October 2022; pp. 1–8.
94. Zhai, G.; Liu, L.; Zhang, L.; Liu, Y.; Jiang, Y. PoseConvGRU: A monocular approach for visual ego-motion estimation by learning. *Pattern Recognit.* **2020**, *102*, 107187. [CrossRef]
95. Zhu, R.; Yang, M.; Liu, W.; Song, R.; Yan, B.; Xiao, Z. DeepAVO: Efficient pose refining with feature distilling for deep visual odometry. *Neurocomputing* **2022**, *467*, 22–35. [CrossRef]
96. Wang, S.; Clark, R.; Wen, H.; Trigoni, N. DeepVO: Towards end-to-end visual odometry with deep recurrent convolutional neural networks. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation, Singapore, 29 May–3 June 2017; pp. 2043–2050. [CrossRef]
97. Li, R.; Wang, S.; Long, Z.; Gu, D. UnDeepVO: Monocular visual odometry through unsupervised deep learning. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation, Brisbane, Australia, 21–25 May 2018; pp. 7286–7291. [CrossRef]
98. Aslan, M.F.; Durdu, A.; Yusefi, A.; Yilmaz, A. HVIOnet: A deep learning based hybrid visual–inertial odometry. *Neural Netw.* **2022**, *155*, 461–474. [CrossRef]
99. Almalioglu, Y.; Turan, M.; Saputra, M.R.U.; de Gusmão, P.P.B.; Markham, A.; Trigoni, N. SelfVIO: Self-supervised deep monocular visual–inertial odometry and depth estimation. *Neural Netw.* **2022**, *150*, 119–136. [CrossRef]
100. Burguera, A. Lightweight underwater visual loop detection and classification using a Siamese convolutional neural network. *IFAC Pap.* **2021**, *54*, 410–415. [CrossRef]
101. Lu, Y.; Chen, Y.; Zhao, D.; Li, D. MGRL: Graph neural network based inference in a Markov network with reinforcement learning for visual navigation. *Neurocomputing* **2021**, *421*, 140–150. [CrossRef]
102. Liu, Y.; Xie, K.; Huang, H. VGF-Net: Visual-geometric fusion learning for simultaneous drone navigation and height mapping. *Graph. Model.* **2021**, *116*, 101108. [CrossRef]
103. Dosovitskiy, A.; Fischer, P.; Ilg, E.; Hausser, P.; Hazirbas, C.; Golkov, V.; van der Smagt, P.; Cremers, D.; Brox, T. FlowNet: Learning optical flow with convolutional networks. In Proceedings of the IEEE International Conference on Computer Vision, Washington, DC, USA, 7–13 December 2015; pp. 2758–2766. [CrossRef]
104. Woo, S.; Park, J.; Lee, J.-Y.; Kweon, I.S. CBAM: Convolutional block attention module. In *Computer Vision–ECCV 2018*; LNCS; Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y., Eds.; Springer: Cham, Switzerland, 2018; Volume 11211, pp. 3–19. [CrossRef]
105. Geiger, A.; Lenz, P.; Urtasun, R. Are we ready for autonomous driving? The KITTI vision benchmark suite. In Proceedings of the 2012 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2012), Providence, RI, USA, 16–21 June 2012; pp. 3354–3361. [CrossRef]
106. Geiger, A.; Lenz, P.; Stiller, C.; Urtasun, R. Vision meets robotics: The KITTI dataset. *Int. J. Robot. Res.* **2013**, *32*, 1231–1237. [CrossRef]
107. Voigtlaender, P.; Krause, M.; Osep, A.; Luiten, J.; Sekar, B.B.G.; Geiger, A.; Leibe, B. MOTS: Multi-object tracking and segmentation. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2019), Long Beach, CA, USA, 15–20 June 2019; pp. 7942–7951.
108. The KITTI Vision Benchmark Suite. Available online: https://www.cvlibs.net/datasets/kitti/index.php (accessed on 25 January 2023).
109. Gaidon, A.; Wang, Q.; Cabon, Y.; Vig, E. VirtualWorlds as proxy for multi-object tracking analysis. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 4340–4349. [CrossRef]
110. Virtual KITTI 2 Dataset. Available online: https://europe.naverlabs.com/research/computer-vision/proxy-virtual-worlds-vkitti-2 (accessed on 12 February 2023).
111. Sturm, J.; Engelhard, N.; Endres, F.; Burgard, W.; Cremers, D. A benchmark for the evaluation of RGB-D SLAM systems. In Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, Vilamoura, Algarve, Portugal, 7–12 October 2012; pp. 573–580. [CrossRef]
112. RGB-D SLAM Dataset and Benchmark. Available online: https://vision.in.tum.de/data/datasets/rgbd-dataset (accessed on 12 February 2023).

113. Shotton, J.; Glocker, B.; Zach, C.; Izadi, S.; Criminisi, A.; Fitzgibbon, A. Scene coordinate regression forests for camera relocalization in RGB-D images. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, 23–28 June 2013; pp. 2930–2937. [CrossRef]
114. RGB-D Dataset 7-Scenes. Available online: https://www.microsoft.com/en-us/research/project/rgb-d-dataset-7-scenes (accessed on 12 February 2023).
115. EuRoC MAV Dataset. Available online: https://mldta.com/dataset/euroc-mav-dataset (accessed on 12 February 2023).
116. Burri, M.; Nikolic, J.; Gohl, P.; Schneider, T.; Rehder, J.; Omari, S.; Achtelik, M.; Siegwart, R. The EuRoC micro aerial vehicle datasets. *Int. J. Robot. Res.* **2016**, *35*, 1157–1163. [CrossRef]
117. VaFRIC (Variable Frame-Rate Imperial College) Dataset. Available online: https://www.doc.ic.ac.uk/~ahanda/VaFRIC/iclnuim.html (accessed on 12 February 2023).
118. Handa, A.; Whelan, T.; McDonald, J.B.; Davison, A.J. A benchmark for RGB-D visual odometry, 3D Reconstruction and SLAM. In Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA 2014), Hong Kong, China, 31 May–5 June 2014; pp. 1–9. [CrossRef]
119. Silberman, N.; Hoiem, D.; Kohli, P.; Fergus, R. Indoor segmentation and support inference from RGBD images. In *Computer Vision–ECCV 2012*; LNCS; Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C., Eds.; Springer: Berlin/Heidelberg, Germany, 2012; Volume 7576, pp. 746–760. [CrossRef]
120. NYU Depth Dataset V2. Available online: https://cs.nyu.edu/~silberman/datasets/nyu_depth_v2.html (accessed on 12 February 2023).
121. Maddern, W.; Pascoe, G.; Linegar, C.; Newman, P. 1 Year, 1000km: The Oxford RobotCar dataset. *Int. J. Robot. Res.* **2017**, *36*, 3–15. [CrossRef]
122. Oxford RobotCar Dataset. Available online: https://robotcar-dataset.robots.ox.ac.uk (accessed on 12 February 2023).
123. The Malaga Stereo and Laser Urban Data Set. Available online: https://www.mrpt.org/MalagaUrbanDataset (accessed on 17 February 2023).
124. Blanco-Claraco, J.-L.; Moreno-Dueñas, F.-Á.; González-Jiménez, J. The Malaga urban dataset: High-rate stereo and LiDAR in a realistic urban scenario. *Int. J. Robot. Res.* **2014**, *33*, 207–214. [CrossRef]
125. Cordts, M.; Omran, M.; Ramos, S.; Rehfeld, T.; Enzweiler, M.; Benenson, R.; Franke, U.; Roth, S.; Schiele, B. The Cityscapes dataset for semantic urban scene understanding. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2016), Las Vegas, NV, USA, 27–30 June 2016; pp. 3213–3223. [CrossRef]
126. The CityScapes Dataset. Available online: https://www.cityscapes-dataset.com (accessed on 17 February 2023).
127. ApolloScapes Dataset. Available online: http://apolloscape.auto/self_localization.html (accessed on 17 February 2023).
128. Huang, X.; Cheng, X.; Geng, Q.; Cao, B.; Zhou, D.; Wang, P.; Lin, Y.; Yang, R. The ApolloScape dataset for autonomous driving. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW 2018), Salt Lake City, UT, USA, 18–22 June 2018; pp. 1067–1073. [CrossRef]
129. ScanNet. Available online: http://www.scan-net.org (accessed on 12 February 2023).
130. Dai, A.; Chang, A.X.; Savva, M.; Halber, M.; Funkhouser, T.A.; Nießner, M. ScanNet: Richly-annotated 3D reconstructions of indoor scenes. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2017), Honolulu, HI, USA, 21–26 July 2017; pp. 2432–2443. [CrossRef]
131. Mid-Air. Available online: https://midair.ulg.ac.be (accessed on 12 February 2023).
132. Fonder, M.; Van Droogenbroeck, M. Mid-Air: A multi-modal dataset for extremely low altitude drone flights. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, Long Beach, CA, USA, 16–17 June 2019; pp. 1–10. [CrossRef]
133. AI2-THOR. Available online: https://ai2thor.allenai.org (accessed on 12 February 2023).
134. Kolve, E.; Mottaghi, R.; Han, W.; VanderBilt, E.; Weihs, L.; Herrasti, A.; Gordon, D.; Zhu, Y.; Gupta, A.; Farhadi, A. AI2-THOR: An interactive 3D environment for visual AI. *arXiv* **2022**, arXiv:1712.05474v4.
135. Wald, J.; Sattler, T.; Golodetz, S.; Cavallari, T.; Tombari, F. Beyond controlled environments: 3D camera re-localization in changing indoor scenes. In *Computer Vision–ECCV 2020*; LNCS; Vedaldi, A., Bischof, H., Brox, T., Frahm, J.M., Eds.; Springer: Cham, Switzerland, 2020; Volume 12352, pp. 467–487. [CrossRef]