Intell Ind Syst (2015) 1:289–311
DOI 10.1007/s40903-015-0032-7

ORIGINAL PAPER

An Overview to Visual Odometry and Visual SLAM: Applications to Mobile Robotics

Khalid Yousif[1] · **Alireza Bab-Hadiashar**[1] · **Reza Hoseinnezhad**[1]

Received: 4 July 2015 / Revised: 30 October 2015 / Accepted: 1 November 2015 / Published online: 13 November 2015

Abstract This paper is intended to pave the way for new researchers in the field of robotics and autonomous systems, particularly those who are interested in robot localization and mapping. We discuss the fundamentals of robot navigation requirements and provide a review of the state of the art techniques that form the bases of established solutions for mobile robots localization and mapping. The topics we discuss range from basic localization techniques such as wheel odometry and dead reckoning, to the more advance Visual Odometry (VO) and Simultaneous Localization and Mapping (SLAM) techniques. We discuss VO in both monocular and stereo vision systems using feature matching/tracking and optical flow techniques. We discuss and compare the basics of most common SLAM methods such as the Extended Kalman Filter SLAM (EKF-SLAM), Particle Filter and the most recent RGB-D SLAM. We also provide techniques that form the building blocks to those methods such as feature extraction (i.e. SIFT, SURF, FAST), feature matching, outlier removal and data association techniques.

Keywords Visual SLAM · Visual Odometry · Navigation · RGB-D · Autonomous · Mapping

✉ Khalid Yousif
s3362555@student.rmit.edu.au

Alireza Bab-Hadiashar
alireza.bab-hadiashar@rmit.edu.au

Reza Hoseinnezhad
reza.hoseinnezhad@rmit.edu.au

[1] School of Aerospace, Mechanical and Manufacturing Engineering, RMIT University, Melbourne, Australia

Introduction

In the past few decades, the area of mobile robotics and autonomous systems has attracted substantial attention from researchers all over the world, resulting in major advances and breakthroughs. Currently, mobile robots are able to perform complex tasks autonomously, whereas in the past, human input and interaction was a necessity. Mobile robotics has applications in various fields such as military, medical, space, entertainment and domestic appliances fields. In those applications, mobile robots are expected to perform complicated tasks that require navigation in complex and dynamic indoor and outdoor environments without any human input. In order to autonomously navigate, path plan and perform these tasks efficiently and safely, the robot needs to be able to localize itself in its environment. As a result, the localization problem has been studied in detail and various techniques have been proposed to solve the localization problem.

The simplest form of localization is to use wheel odometry methods that rely upon wheel encoders to measure the amount of rotation of robots wheels. In those methods, wheel rotation measurements are incrementally used in conjunction with the robot's motion model to find the robot's current location with respect to a global reference coordinate system. The wheel odometry method has some major limitations. Firstly, it is limited to wheeled ground vehicles and secondly, since the localization is incremental (based on the previous estimated location), measurement errors are accumulated over time and cause the estimated robot pose to drift from its actual location. There are a number of error sources in wheel odometry methods, the most significant being wheel slippage in uneven terrain or slippery floors.

To overcome those limitations, other localization strategies such using inertial measurement units (IMUs), GPS, LASER odometry and most recently Visual Odometry

(VO) [88] and Simultaneous Localization and Mapping (SLAM) [36,38] methods have been proposed. VO is the process of estimating the egomotion of an agent (e.g., vehicle, human, and robot) using only the input of a single or multiple cameras attached to it [101]. Whereas SLAM is a process in which a robot is required to localize itself in an unknown environment and build a map of this environment at the same time without any prior information with the aid of external sensors (or a single sensor). Although VO does not solve the drift problem, researchers have shown that VO methods perform significantly better than wheel odometry and dead reckoning techniques [54] while the cost of cameras is much lower compared to accurate IMUs and LASER scanners. The main difference between VO and SLAM is that VO mainly focuses on local consistency and aims to incrementally estimate the path of the camera/robot pose after pose, and possibly performing local optimization. Whereas SLAM aims to obtain a globally consistent estimate of the camera/robot trajectory and map. Global consistency is achieved by realizing that a previously mapped area has been re-visited (loop closure) and this information is used to reduce the drift in the estimates. Figure 1 shows an overview of VO and SLAM systems.

This paper extends on the past surveys of visual odometry [45,101]. The main difference between this paper and the aforementioned tutorials is that we aim to provide the fundamental frameworks and methodologies used for visual SLAM in addition to VO implementations. The paper also includes significant developments in the area of Visual SLAM such as those that devise using RGB-D sensors for dense 3D reconstruction of the environment.

In the next section, we will discuss the related work in this area. In "Localization" and "Mapping" we will discuss the localization and mapping problems respectively. The SLAM problem will be presented in "Simultaneous Localization and Mapping" section and the fundamental components in VO and V-SLAM will be discussed in "Fundamental Components in V-SLAM and VO" section. In "RGB-D SLAM" section we will present the RGB-D SLAM approach for solving the V-SLAM problem. Finally we will conclude this paper in "Conclusion" section.

Related Work

Visual Odometry

VO is defined as the process of estimating the robot's motion (translation and rotation with respect to a reference frame) by observing a sequence of images of its environment. VO is a particular case of a technique known as Structure From Motion (SFM) that tackles the problem of 3D reconstruction of both the structure of the environment and camera poses from sequentially ordered or unordered image sets [101]. SFM's final refinement and global optimization step of both the camera poses and the structure is computationally expensive and usually performed off-line. However, the estimation of the camera poses in VO is required to be conducted in real-time. In recent years, many VO methods have been proposed which those can be divided into monocular [19] and stereo camera methods [79]. These methods are then further divided into feature matching (matching features over a number of frames) [108], feature tracking [31] (matching features in adjacent frames) and optical flow techniques [118] (based on the intensity of all pixels or specific regions in sequential images).

The problem of estimating a robot's ego-motion by observing a sequence of images started in the 1980s by Moravec [82] at Stanford University. Moravec used a form of stereo vision (a single camera sliding on a rail) in a move and stop fashion where a robot would stop to extract image features (corners) in the first image, the camera would then slide on a rail in a perpendicular direction with respect to the robot's motion, and repeat the process until a total of 9 images are captured. Features were matched between the 9 images using Normalized Cross Correlation (NCC) and used those to reconstruct the 3D structure. The camera motion transformation was then obtained by aligning the reconstructed 3D points observed from different locations. Matthies and Shafer [79] later extended the above work by deriving an error model using 3D Gaussian distributions as opposed to the scalar model used in Moravec's method. Other notable works

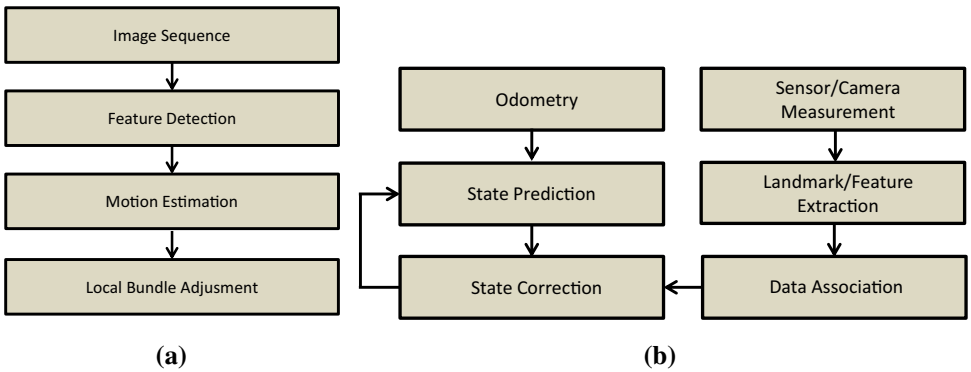


Fig. 1 A block diagram showing the main components of a: **a** VO and **b** filter based SLAM system

related to stereo VO were also appeared in literature. For instance, in [90] a maximum likelihood ego-motion method for modeling the error was presented for accurate localization of a rover over long distances while [73] described a method for outdoor rover localization that relied on raw image data instead of geometric data for motion estimation.

The term "Visual Odometry" was first introduced by Nister et al. [88] for it's similarity to the concept of wheel odometry. They proposed pioneering methods for obtaining camera motion from visual input in both monocular and stereo systems. They focused on the problem of estimating the camera motion in the presence of outliers (false feature matches) and proposed an outlier rejection scheme using RANSAC [44]. Nister et al. were also the first to track features across all frames instead of matching features in consecutive frames. This has the benefit of avoiding feature drift during cross-correlation based tracking [101]. They also proposed a RANSAC based motion estimation using the 3D to 2D re-projection error (see "Motion Estimation" section) instead of using the Euclidean distance error between 3D points. Using 3D to 2D re-projection errors were shown to give better estimates when compared to the 3D to 3D errors [56].

VO is most famous for its application in the ongoing robotic space mission on Mars [22] that started in 2003 involving two rovers that were sent to explore the geology and surface of the planet. Other research involving VO was performed by Scaramuzza and Siegwart [102] where they focus on VO for ground vehicles in an outdoor environment using a monocular omni-directional camera and fuse the motion estimates gained by two following approaches. In the first approach, they extracted SIFT [77] features and used RANSAC for outlier removal while in the second approach an appearance based method, which was originally proposed by [27], was used for the vehicle pose estimation. Appearance based techniques are generally able to accurately handle outdoor open spaces efficiently and robustly whilst avoiding error prone feature extraction and matching techniques [6].

Kaess et al. [67] proposed a stereo VO system for outdoor navigation in which the sparse flow obtained by feature matching was separated into flow based on close features and flow based on distant features.The rationale for the separation is that small changes in camera translations do not visibly influence points that are far away. The distant points were used to recover the rotation transformation (using a two-point RANSAC) while close points were used to recover the translation using a one-point RANSAC [67]. Alcantarilla et al. [3] integrated the visual odometery information gained from the flow separation method in their EKF-SLAM motion model to improve the accuracy of the localization and mapping.

We have so far discussed the history of the VO problem and mentioned some of the pioneering work in this area that mainly focused on monocular VO, stereo VO, motion estimation, error modeling, appearance based, and feature based techniques. In the next section we will present a brief history of and the related works to the visual SLAM problem.

Visual SLAM

As we described in the introduction section, SLAM is a way for a robot to localize itself in an unknown environment, while incrementally constructing a map of its surroundings. SLAM has been extensively studied in the past couple of decades [48,66,91] resulting in many different solutions using different sensors, including sonar sensors [71], IR sensors [1] and LASER scanners [26]. Recently there has been an increased interest in visual based SLAM also known as V-SLAM because of the rich visual information available from passive low-cost video sensors compared to LASER scanners. However, the trade off is a higher computational cost and the requirement for more sophisticated algorithms for processing the images and extracting the necessary information. Due to recent advances in CPU and GPU technologies, the real time implementation of the required complex algorithms are no longer an insurmountable problem. Indeed, variety of solutions using different visual sensors including monocular [28], stereo [78], omni-directional [70], time of flight (TOF) [103] and combined color and depth (RGB-D) cameras [56] have been proposed.

One of the pioneering V-SLAM solutions was proposed by Davison et al. [28]. They employed a single monocular camera and constructed a map by extracting sparse features of the environment using a Shi and Tomasi operator [103] and matching new features to those already observed using a normalized sum-of-squared difference correlation. The use of single monocular camera meant that the absolute scale of structures could not be obtained and the camera had to be calibrated. Furthermore, since an Extended Kalman Filter (EKF) was used for state estimation, only a limited number of features were extracted and tracked to manage the computational cost of the EKF. Se et al. [56] proposed a vision based method for mobile robot localization and mapping using the SIFT [77] for feature extraction.

A particular case of V-SLAM, known as cv-SLAM (Ceiling Vision SLAM), was studied by pointing the camera upwards towards the ceiling. The advantages of cv-SLAM when compared to the frontal view V-SLAM are: less interactions with moving obstacles and occlusions, steady observation of features and the fact that ceilings are usually highly textured and rich with visual information. Jeong et al. [63,64] were the first to propose the cv-SLAM method, where they employed a single monocular camera that was pointed upwards towards the ceiling. Corner features on the ceiling were extracted using a Harris corner detector [53]. A landmark orientation estimation technique was then used to align and match the currently observed and

previously stored landmarks using a NCC method. Other researchers [23,24,58,59] followed suit and produced various studies on cv-SLAM using different techniques for feature extraction and data association (DA).

Most Recently, there has been an increasing interest in dense 3D reconstruction of the environment as opposed to the sparse 2D and 3D SLAM problems. Newcombe and Davison [85] were successful in obtaining a dense 3D model of the environment in real time using a single monocular camera. However, their method is limited to small and highly textured environments. Henry et al. [56] were the first to implement an RGB-D mapping approach that employed an RGB-D camera (i.e. Microsoft Kinect). They used this information to obtain a dense 3D reconstructed environment and estimated the 6 degree of freedom (6DOF) camera pose. They extracted Features From Accelerated Segment Test (FAST) [96] features in each frame, match them with the features from the previous frame using the Calonder descriptors [12] and performed a RANSAC alignment step which obtains a subset of feature matches (inliers) that correspond to a consistent rigid transformation [56]. This transformation was used as an initial guess in the Iterative Closest Point (ICP) [12] algorithm which refined the transformation obtained by RANSAC. Sparse Bundle Adjustment (SBA) [76] was also applied in order to obtain a globally consistent map and loop closure was detected by matching the current frame to previously collected key-frames. Similarly Endres et al. [41] proposed an RGB-D-SLAM method that uses SIFT, SURF [10] and ORB [97] feature descriptors in place of FAST features. They also used a pose-graph optimization technique instead of Bundle Adjustment for global optimization. Du et al. [34] implemented an RGB-D SLAM system that incorporates online user interaction and feedback allowing the system to recover from registration failures which may be caused by fast camera movement. Audras et al. [5] proposed an appearance based RGB-D SLAM which avoids the error prone feature extraction and feature matching steps.

Newcombe et al. [61,84] proposed a depth only mapping algorithm using an RGB-D camera. They developed an ICP variant method that matches the current measurement to the full growing surface model instead of matching sequential frames. They also segmented outliers (such as moving humans) and divided the scene into foreground and background. This allowed a user to interact with the scene without deteriorating the accuracy of the estimated transformations and demonstrated the usability of their method in a number of augmented reality applications. The major downside to their approach is that it is limited to small environments. Whelan et al. [113] outlined this problem and proposed an extension to KinectFusion that enabled the approach to map larger environments. This was achieved by allowing the region of the environment that is mapped by the KinectFusion algorithm to vary dynamically.

Bachrach et al. [8] proposed a VO and SLAM system for unmanned air vehicles (UAVs) using an RGB-D camera that relied on extracting FAST features from sequential preprocessed images at different pyramid levels, followed by an initial rotation estimation that limited the size of the search window for feature matching. The matching was performed by finding the mutual lowest sum of squared difference (SSD) score between the descriptor vectors (80-byte descriptor consisting of the brightness values of the 9×9 pixel patch around the feature and omitting the bottom right pixel). A greedy algorithm was also applied to refine the matches and obtain the inlier set which were then used to estimate the motion between frames. In order to reduce drift in the motion estimates, they suggested matching the current frame to a selected key-frame instead of matching consecutive frames. Hu et al. [57] outlined the problem of not having enough depth information in large rooms due to the limitations of RGB-D cameras. They proposed a switching algorithm that heuristically chooses between an RGB-D mapping approach similar to Henry et al.'s [55] method and a 8-point RANSAC monocular SLAM, based on the availability of depth information. The two maps were merged using a 3D Iterative Sparse Local Submap Joining Filter (I-SLSJF). Yousif et al. outlined the problem of mapping environments containing limited texture, as such, they proposed a method [116,117] that solves this issue by only using depth information for registration using a ranked order statistics based sampling scheme that is able to extract useful points for registration by directly analyzing each point's neighborhood and the orientations of their normal vectors. Kerl et al. [69] recently proposed a method that uses both photometric and geometric information for registration. In their implementation, they use all the points for registration and optimize both intensity and depth errors. Another recent method proposed by Keller et al. [68] allows for 3D reconstruction of dynamic environments by automatically detecting dynamic changes in the scene. Dryanovski et al. [33] proposed a fast visual odometry and mapping method that extracts features from RGB-D images and aligns those with a persistent model, instead of frame to frame registration techniques. By doing so, they avoid using dense data, feature descriptor vectors, RANSAC alignment, or keyframe-based bundle adjustment (BA). As such, they are able to achieve an average performance rate of 60Hz using a single thread and without the aid of a GPU.

In the above section, we discussed various approaches to solving the V-SLAM problem. Earlier methods generally focused on sparse 2D and 3D SLAM methods due to limitations of available computational resources. Most recently, the interest has shifted towards dense 3D reconstruction of the environment due to technological advances and availability of efficient optimization methods. In the next section, we will discuss the localization problem and present the different localization techniques based on the VO approach.

Localization

Visual Odometry

VO is the process of estimating the camera's relative motion by analyzing a sequence of camera images. Similar to wheel odometry, estimates obtained by VO are associated with errors that accumulate over time [39]. However, VO has been shown to produce localization estimates that are much more accurate and reliable over longer periods of time compared to wheel odometry [54]. VO is also not affected by wheel slippage usually caused by uneven terrain.

Motion Estimation

In general, there are three commonly used VO motion estimation techniques called: 3D to 3D, 3D to 2D and 2D to 2D methods. Figure 2 illustrates an example of the VO problem. The motion estimation techniques are outlined here.

3D to 3D Motion Estimation In this approach, the motion is estimated by triangulating 3D feature points observed in a sequence of images. The transformation between the camera frames is then estimated by minimizing the 3D Euclidean distance between the corresponding 3D points as shown below.

$$\mathbf{T} = \underset{\boldsymbol{T}}{\operatorname{argmin}} \sum_i |\mathbf{X_i} - \mathbf{T}\acute{\mathbf{X}}_\mathbf{i}|^2 \tag{1}$$

In the above equation, $\mathbf{T}$ is the estimated transformation between two consecutive frames, $\mathbf{X}$ is the 3D feature point observed by the current frame F_k, $\acute{\mathbf{X}}$ is the corresponding 3D feature point in the previous frame F_{k-1} and i is the minimum number of feature pairs required to constrain the transformation. The minimum number of required points depends on the systems' DOF and the type of modeling used. Although using more points means more computation, better accuracy is achieved by including more points than the minimum number required.

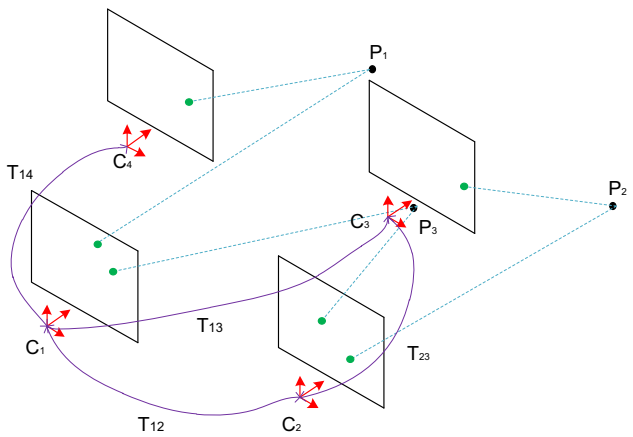


Fig. 2 An example of a monocular VO system. The relative poses T_{nm} between cameras viewing the same 3D point are computed by matching the corresponding points in the 2D image. If the points' 3D location is known, a 3D to 3D or 3D to 2D method may be used. The global poses C_n are computed by concatenating the relative transformations with respect to a reference frame (can be set to the initial frame)

3D to 2D Motion Estimation This method is similar to the previous approach but here the 2D re-projection error is minimized to find the required transformation. The cost function for this method is as follows:

$$\mathbf{T} = \underset{\boldsymbol{T}}{\operatorname{argmin}} \sum_i |\mathbf{z} - f(\mathbf{T}, \acute{\mathbf{X}}_\mathbf{i})|^2 \tag{2}$$

where T is the estimated transformation between two consecutive frames, $\mathbf{z}$ is the observed feature point in the current frame F_k, $f(\mathbf{T}, \acute{X}_i)$ is the re-projection function of it's corresponding 3D feature point in the previous frame F_{k-1} after applying a transformation $\mathbf{T}$ and i is the number of feature pairs. Again, the minimum number of points required varies based on the number of constraints in the system.

2D to 2D Motion Estimation The 3D to 3D and 3D to 2D approaches are only possible when 3D data are available. This is not always the case, for instance, estimating the relative transformation between the first two calibrated monocular frames where points have not been triangulated yet. In this case, the epipolar geometry is exploited to estimate this transformation. An example of the epipolar geometry is illustrated in Fig. 3. The figure shows two cameras, separated by a rotation and a translation, viewing the same 3D point. Each camera captures a 2D image of the 3D world. This conversion from 3D to 2D is referred to as a perspective projection and is described in more detail in "Camera Modeling and Calibration" section. The epipolar constraint used in this approach is written as:

$$\mathbf{q}'^{\top}\mathbf{E}\mathbf{q} = 0 \tag{3}$$

where $\mathbf{q}$ and $\mathbf{q}'$ are the corresponding homogeneous image points in two consecutive frames and $\mathbf{E}$ is the essential matrix given by:

$$\mathbf{E} = [\mathbf{t}]_\times \mathbf{R} \tag{4}$$

where R is the rotation matrix, $\mathbf{t}$ is the translation matrix given by:

$$\mathbf{t} = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$$

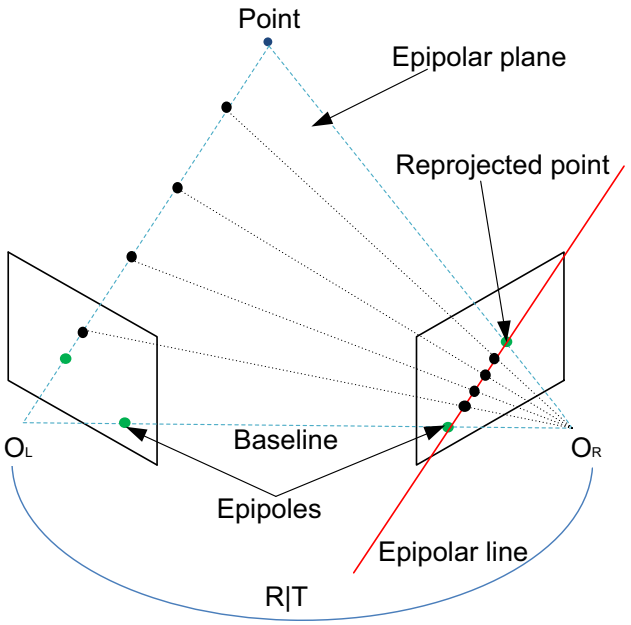


Fig. 3 Illustration of the epipolar geometry. The two cameras are indicated by their centres O_L and O_R and image planes. The camera centers, 3D point, and its re-projections on the images lie in a common plane. An image point back-projects to a ray in 3D space. This ray is projected back as a line in the second image called the epipolar line (shown in *red*). The 3D point lies on this ray, so the image of the 3D point in the second view must lie on the epipolar line. The pose between O_L and O_R can be obtained using the essential matrix **E**, which is the algebraic representation of epipolar geometry for known calibration

and $[\mathbf{t}]_\times$ is the skew symmetric matrix given by:

$$[\mathbf{t}]_\times = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & t_y \\ -t_y & t_x & 0 \end{bmatrix}.$$

Full descriptions of different ways to solve the motion estimation using the above approach are provided by [75, 86, 119].

Stereo Vision Versus Monocular Vision

Stereo Visual Odometry In stereo vision, 3D information is reconstructed by triangulation in a single time-step by simultaneously observing the features in the left and right images that are spatially separated by a known baseline distance. In Stereo VO, motion is estimated by observing features in two successive frames (in both right and left images). The following steps outline a common procedure for stereo VO using a 3D to 2D motion estimation:

1. Extract and match features in the right frame $F_{R(I)}$ and left frame $F_{L(I)}$ at time I, reconstruct points in 3D by triangulation.
2. Match these features with their corresponding features in the next frames $F_{R(I+1)}$ and $F_{L(I+1)}$.
3. Estimate the transformation that gives the minimum sum of square differences (SSD) between the observed features in one of the camera images (left or right) and the re-projected 3D points that were reconstructed in the previous frame after applying the transformation to the current frame (see Eq. 2).
4. Use a RANSAC type refinement step (see "Refining the Transformation Using ICP" section) to recalculate the transformation based on inlier points only.
5. Concatenate the obtained transformation with previously estimated global transformation.
6. Repeat from 1 at each time step.

Monocular Visual Odometry For the reconstruction of feature points in 3D via triangulation, they are required to be observed in successive frames (time separated frames). In monocular VO, feature points need to be observed in at least three different frames (observe features in the first frame, re-observed and triangulate into 3D points in the second frame, and calculate the transformation in the third frame). A major issue in monocular VO is the scale ambiguity problem. Unlike stereo vision systems where the transformation (rotation and translation) between the first two camera frames can be obtained, the transformation between the first two consecutive frames in monocular vision is not fully known (scale is unknown) and is usually set to a predefined value. Therefore, the scale of the reconstructed 3D points and following transformations are relative to the initial predefined scale between the first two frames. The global scale cannot be obtained unless additional information about the 3D structure or the initial transformation is available. It has been shown [89] that the required information can be collected using other sensors such as IMU's, wheel encoders or GPS. The procedure for monocular VO is similar to stereo VO (described in "Stereo Vision Versus Monocular Vision" section). However, unlike stereo VO, the triangulation of feature points occurs at different times (sequential frames).

A possible procedure for Monocular VO using the 3D to 2D motion estimation is described in the following steps:

1. Extract features in the first frame F_I at time step I and assign descriptors to them.
2. Extract features in the next frame F_{I+1} and assign descriptors to them.
3. Match features between the two consecutive frames. Estimate a transformation (with predefined scale) between the first two frames using a 5-point algorithm [86] and triangulate the corresponding points using this transformation (3D points will be up to the assumed scale).
4. Extract features in the following frame F_{I+2}, match them with the previously extracted features from the previous frame.

5. Use a RANSAC to refine the matches and estimate the transformation that gives the minimum sum of square differences (SSD) between the observed features in the current frame F_{I+2} and the re-projected 3D points that were reconstructed from the two previous frames after applying the transformation (see Eq. 2). This process is called Perspective N Points (PnP) algorithm [74].
6. Triangulate the matched feature pairs between F_{I+1} and F_{I+2} into 3D points using the estimated transformation.
7. Set $I = I + 1$, repeat from step 4 for every iteration.

Visual Odometry Based on Optical Flow Methods

Optical flow calculation is used as a surrogate measurement of the local image motion. The optical flow field is calculated by analyzing the projected spatio-temporal patterns of moving objects in an image plane and its value at a pixel specifies how much that pixel has moved in sequential images. Optical flow measures the relative motion between objects and the viewer [47] and can be useful in estimating the motion of a mobile robot or a camera relative to its environment.

Optical flow calculation is based on the *Intensity Coherence* assumption which states that the image brightness of a point projected on two successive images is (strictest assumption) constant or (weakest assumption) nearly constant [7]. This assumption leads to the well-known optical flow constraint:

$$\frac{\partial I}{\partial x}\mathbf{v_x} + \frac{\partial I}{\partial y}\mathbf{v_y} + \frac{\partial I}{\partial t} = 0 \tag{5}$$

where $\mathbf{v_x}$ and $\mathbf{v_y}$ are the x and y optical flow components. A number of algorithms to solve the optical flow problem using motion constraint equations have been proposed (see [115] for a list of current approaches).

Having calculated the 2D displacements (u, v) for every pixel, the 3D camera motion can be fully recovered. Irani et al. [60] described an equation for retrieving the 6DOF motion parameters of a camera which consist of three translational (T_x, T_y, T_z) and three rotation components $(\Omega_x, \Omega_y, \Omega_z)$. Their approach is based on solving the following equations:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \frac{-f_c T_x + x T_z}{Z} + \frac{xy}{f_c}\Omega_x - \left(\frac{x^2}{f_c} + f_c\right)\Omega_y + y\Omega_z \\ \frac{-f_c T_y + x T_z}{Z} - \frac{xy}{f_c}\Omega_y + \left(\frac{y^2}{f_c} + f_c\right)\Omega_x + x\Omega_z \end{bmatrix} \tag{6}$$

where f_c is the camera's focal length and (x, y) are the image coordinates of the 3D point (X, Y, Z).

Assuming that the depth Z is known, there are six unknowns and a minimum of three points are required to fully constrain the transformation. However, in many situations, additional constraints are imposed such as moving on a flat plane, therefore reducing both the of DOF and the minimum number of points required to constrain the transformation.

Mapping

In most real-world robotics applications, maps of the environment in which the mobile robot is required to localize and navigate are not available. Therefore, in order to achieve true autonomy, generating a map representation of the environment is one of the important competencies of autonomous vehicles [109]. In general, mapping the environment is considered to be a challenging task. The most commonly used mapping representations are as follows.

Metric Maps

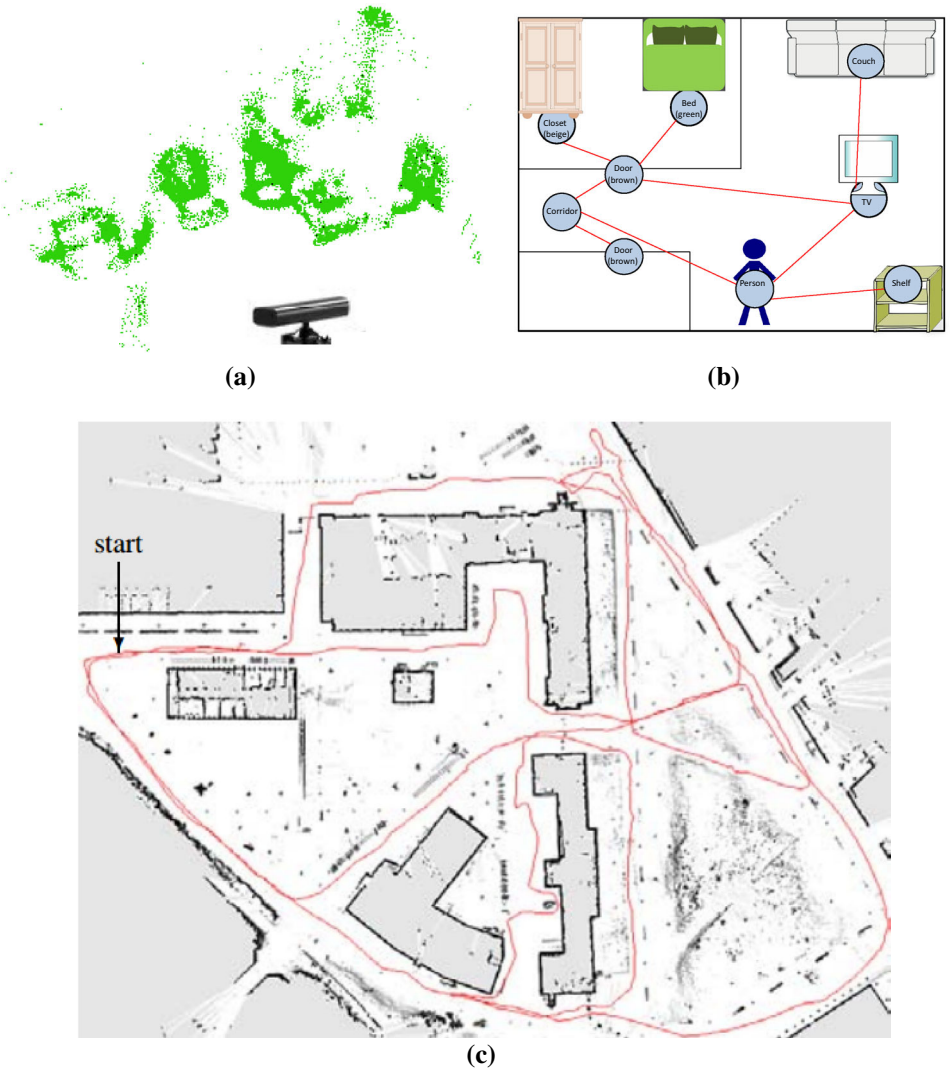
In metric maps, the environment is represented in terms of geometric relations between the objects and a fixed reference frame [92]. The most common forms of metric maps are:

Feature Maps

Feature maps [21] represent the environment in a form of sparse geometric shapes such as points and straight lines. Each feature is described by a set of parameters such as its location and geometric shape. Localization in such environments is performed by observing and detecting features and comparing those with the map features that have already been stored. Since this approach uses a limited number of sparse objects to represent a map, its computation cost can be kept relatively low and map management algorithms are good solutions for current applications. The major weakness in feature map representation is its sensitivity to false data association [9] (a measurement that is incorrectly associated to a feature in the map). This is particularly evident in data association techniques that do not take the correlation between stored features into account. DA solutions to this problem have been proposed such as [83] and those are discussed in more detail in "Data Association" section.

Occupancy Grids

Occupancy Grid maps [14,40] are represented by an array of cells in which each cell (or pixel in an image) represents a region of the environment. Unlike feature maps which are concerned about the geometric shape or type of the objects, occupancy grids are only concerned about the occupancy probability of each cell. This probability value ranges between 0 (not occupied) and 1 (occupied). In occupancy grid mapping, DA between the observed measurements and the stored map is performed by similarity based techniques such as cross correlation [9]. One of the major advantages



(a) **(b)** **(c)**

Fig. 4 An example of a different mapping techniques. **a** Feature map. **b** Topological map. **c** Occupancy grid map [51]

of this representation is its usefulness in path planning and exploration algorithms in which the occupancy probability information can reduce the complexity of the path planning task. The major drawback of this method is its computational complexity especially for large environments. A trade-off between accuracy and computational cost can be achieved by reducing the resolution of the map where every cell would represent a larger region.

Topological Maps

In contrast to metric maps which are concerned about the geometric relations between places or landmarks, topological maps are only concerned about adjacency information between objects [35] and avoid metric information as far as possible [92]. Topological maps are usually represented by a graph in which nodes define places or landmarks and contain distinctive information about them and connecting arcs that manifest adjacency information between the connected nodes. Topological maps are particularly useful in representing a large environment in an abstract form where only necessary information are held. This information includes high level features such as objects, doors, humans and other semantic representation of the environments. Figure 4 illustrates a simple example of a topological map. One of the major advantages of topological maps is its usefulness in high-level path planning methods within graph data structures such as finding the shortest path. DA is performed by comparing the information obtained from sensor measurements to the distinctive information held at each node. For example, DA can be performed using place recognition approaches such as using a visual dictionary [4] or other high level feature matching approaches. Additional constraints between nodes may be added when re-observing places and detecting loop closures. One of the main weaknesses of topological maps is the difficulty in ensuring reliable navigation

between different places without the aid of some form of a metric location measure [9]. Methods such as follow the right or left wall are adequate in many applications such as navigating in a static indoor environment (for example all doors are closed). However, relying only on qualitative information may not be sufficient for navigation in dynamic and cluttered environments. Another major weakness is the issue of detecting false DA where the robot fails to recognize a previously observed place (may be due to small variations of the place) or associating a location with an incorrect place. In such cases, the topological sequence is broken and the robot's location information would become inaccurate [9].

Hybrid Maps (Metric + Topological)

In general, metric maps result in more accurate localization, whereas topological maps results in an abstract representation of the environment which is more useful for path planning methods. The functionalities of those representations are complementary [92] and the combination of metric (quantitative information) and qualitative information have been used to improve navigation and DA [93,110].

Simultaneous Localization and Mapping

In the previous sections, we described the localization and mapping problems separately. SLAM is an attempt to solve both of those problems at the same time. SLAM approaches have been categorized into filtering approaches (such as EKF-SLAM [106] and particle filter based SLAM [81]) and smoothing approaches (such as GraphSLAM [111], RGB-D SLAM [56], Smoothing and Mapping [30]). Filtering approaches are concerned with solving the on-line SLAM problem in which only the current robot state and the map are estimated by incorporating sensor measurements as they become available. Whereas smoothing approaches address the full SLAM problem in which the posterior is estimated over the entire path along with the map, and is generally solved using a least square (LS) error minimization technique [49].

Extended Kalman Filter Based SLAM

The EKF is arguably the most common technique for state estimation and is based on the Bayes filter for the filtering and prediction of non linear functions in which their linear approximations are obtained using a 1st order Taylor series expansion. EKF is also based on the assumption that the initial posterior has a Gaussian distribution and by applying the linear transformations, all estimated states would also have Gaussian distributions.

The EKF-SLAM is divided into two main stages: prediction and update (correction). In the prediction step, the future position of the robot is estimated (predicted) based on the robot's current position and the control input that is applied to change the robot's position from time step k to $k+1$ (usually obtained by dead reckoning techniques), while taking into account the process noise and uncertainties. The general motion model equation can be formulated as:

$$\mathbf{X_{k+1}} = f(\mathbf{X_k}, \mathbf{U_k}) + \mathbf{W_k} \tag{7}$$

where $\mathbf{X_{k+1}}$ is an estimate of the robot's future position, $f(\mathbf{X_k}, \mathbf{U_k})$ is a function of the current estimate of the robot's position X_k and the control input that is applied to change the robot's position from time step k to $k+1$ and $\mathbf{W_k}$ is the process uncertainty that is assumed to be uncorrelated zero mean Gaussian noise with covariance $\mathbf{Q}$.

The general measurement (observation) model can be formulated as:

$$\mathbf{z} = h(\mathbf{X_k}) + \mathbf{V_k} \tag{8}$$

where $\mathbf{z}$ is the noisy measurement, $h(\bar{\mathbf{X}}_\mathbf{k})$ is the observation function and $\mathbf{V_k}$ is the measurement noise and is assumed to be uncorrelated zero mean Gaussian noise with covariance $\mathbf{R}$.

EKF-SLAM has the advantage of being easy to implement and is more computationally efficient than other filters such as the particle filter [109]. However, EKF-SLAM major limitation is its linear approximation of non-linear functions using the Taylor expansion which can result in inaccurate estimates in some cases. Another limitation is caused by the Gaussian density assumption which means that the EKF-SLAM cannot handle the multi-modal densities associated with global localization (localizing the robot without knowledge of its initial position).

FastSLAM 1.0

The FastSLAM 1.0 [81] is popular SLAM technique based on the Rao-Blackwellized particle filter [32]. As it is described in [109] the FastSLAM 1.0 consists of three main stages: Sampling (drawing M particles), updating and re-sampling (based on an importance factor). For simplicity, in our explanations, we assume known DA (i.e. the correspondence between measurements and landmarks are known). We also assume that only one landmark is observed at each point in time. Another important assumption in the FastSLAM 1.0 method is that features are conditionally independent given the robot pose. In other words, in contrast to the EKF-SLAM that uses a single EKF to estimate the robot pose and the map, FastSLAM 1.0 uses separate EKFs (consisting of a mean $\mu^{\mathbf{k}}_{\mathbf{j,t}}$ and a covariance $\mathbf{\Sigma}^{\mathbf{k}}_{\mathbf{j,t}}$) for each feature and each

particle. This means the total number of EKFs is $M \times N$ where N is the total number of features observed. We also assume that the motion model used here is the same one described in the EKF-SLAM section, and the observations are also based on the range-bearing sensor measurements (although this method can also be applied to vision and other sensors). Let us denote the full SLAM posterior as $\mathbf{Y_t}$ which consists of $[\mathbf{x_t^{[k]}}, (\mu_{\mathbf{1,t}}^{\mathbf{k}}, \mathbf{\Sigma_{1,t}^k}), \ldots, (\mu_{\mathbf{j,t}}^{\mathbf{k}}, \mathbf{\Sigma_{j,t}^k})]$ for $k = 1 : M$ particles and $j = 1 : N$ features, $x_t^{[k]}$ is the pose of the kth particle at time t. In the following section we describe the FastSLAM 1.0 procedure in detail.

Sample

The first step is to retrieve M $(k = 1 : M)$ particles and their features $[\mathbf{x_{t-1}^{[k]}}, (\mu_{\mathbf{1,t-1}}^{\mathbf{k}}, \mathbf{\Sigma_{1,t-1}^k}), \ldots, (\mu_{\mathbf{j,t-1}}^{\mathbf{k}}, \mathbf{\Sigma_{j,t-1}^k})]$ from the previous posterior $\mathbf{Y_{t-1}}$, where $\mathbf{x_{t-1}^{[k]}} = [x_{x(t-1)}^{[k]} \quad x_{y(t-1)}^{[k]} \quad \theta_{t-1}^{[k]}]^T$ is the kth particle's pose at time t. This follows by sampling a new pose for each particle using a particular motion model.

Observe new features

The next step is to observe features and add new ones to each state. In the case that a feature has not been observed previously, that feature's mean $\mu_{j,t}^k$ and covariance $\Sigma_{j,t}^k$ are added to the state vector of all particles using the following equations:

$$\left[\mu_{\mathbf{j,t}}^{\mathbf{k}}\right] = \begin{bmatrix} \mu_{x,j,t}^k \\ \mu_{y,j,t}^k \end{bmatrix} = \begin{bmatrix} x_{xt}^{[k]} + r\cos(\theta_t^{[k]} + \phi) \\ x_{yt}^{[k]} + r\sin(\theta_t^{[k]} + \phi) \end{bmatrix} \tag{9}$$

where r and ϕ are the range and bearing measurements respectively. To find the Jacobian of (9) with respect to the range and bearing, we have:

$$\mathbf{H_m} = \begin{bmatrix} \cos(\theta_t^{[k]} + \phi) & -r\sin(\theta_t^{[k]} + \phi) \\ \sin(\theta_t^{[k]} + \phi) & r\cos(\theta_t^{[k]} + \phi) \end{bmatrix}. \tag{10}$$

The covariance matrix of the new feature can also be calculated by:

$$\mathbf{\Sigma_{j,t}^k} = \mathbf{H_m Q_t H_m^T} \tag{11}$$

where $\mathbf{Q_t}$ is the measurement noise covariance matrix. In the case that only new features are observed, a default importance weight is assigned to the particle $w^{[k]} = p_0$.

Update

If the robot re-observes a feature (assuming known data-association), an EKF update step is required to compute the importance factor for each particle. As part of the EKF update, we first need to calculate the measurement prediction based on each particle's predicted location using:

$$\hat{\mathbf{z}} = \begin{bmatrix} \sqrt{\left(\mu_{x,j,t}^k - x_{xt}^{[k]}\right)^2 + \left(\mu_{y,j,t}^k - x_{yt}^{[k]}\right)^2} \\ \arctan\left(\frac{\mu_{y,j,t}^k - x_{yt}^{[k]}}{\mu_{x,j,t}^k - x_{xt}^{[k]}}\right) - \theta_t^{[k]} \end{bmatrix}. \tag{12}$$

Next we need to calculate the Jacobian of (12) with respect to the feature location using:

$$\mathbf{H} = \begin{bmatrix} \frac{\mu_{x,j,t}^k - x_{xt}^{[k]}}{r} & \frac{\mu_{y,j,t}^k - x_{yt}^{[k]}}{r} \\ -\left(\frac{\mu_{y,j,t}^k - x_{yt}^{[k]}}{r^2}\right) & \frac{\mu_{x,j,t}^k - x_{xt}^{[k]}}{r^2} \end{bmatrix} \tag{13}$$

$$r = \sqrt{(\mu_{x,j,t}^k - x_{xt}^{[k]})^2 + (\mu_{y,j,t}^k - x_{yt}^{[k]})^2}. \tag{14}$$

We then need to calculate the measurement covariance and the Kalman gain by:

$$\mathbf{Q} = \mathbf{H\Sigma_{j,t-1}^k H^T} + \mathbf{Q_t} \tag{15}$$

$$\mathbf{K} = \mathbf{\Sigma_{j,t-1}^k H^T Q^{-1}} \tag{16}$$

and update the mean and covariance of the re-observed feature by:

$$\mu_{\mathbf{j,t}}^{\mathbf{k}} = \mu_{\mathbf{j,t-1}}^{\mathbf{k}} + \mathbf{K}(\mathbf{z_t} - \hat{\mathbf{z}}) \tag{17}$$

$$\mathbf{\Sigma_{j,t}^k}(\mathbf{I} - \mathbf{KH})\mathbf{\Sigma_{j,t-1}^k}. \tag{18}$$

Finally, the importance factor $w^{[k]}$, which is based on the ratio between the target distribution and the proposal distribution, is calculated using:

$$w^{[k]} = |2\pi\mathbf{Q}|^{\frac{-1}{2}} \exp\left(-\frac{\mathbf{1}}{\mathbf{2}}(\mathbf{z_t} - \hat{\mathbf{z}})^{\mathbf{T}}\mathbf{Q^{-1}}(\mathbf{z_t} - \hat{\mathbf{z}})\right). \tag{19}$$

A detailed derivation of $w^{[k]}$ is provided by [109]. For all the other features that have not been observed, their estimated location remains unchanged:

$$\mu_{\mathbf{j,t}}^{\mathbf{k}} = \mu_{\mathbf{j,t-1}}^{\mathbf{k}} \tag{20}$$

$$\mathbf{\Sigma_{j,t}^k} = \mathbf{\Sigma_{j,t-1}^k}. \tag{21}$$

The previous steps are repeated for all particles.

Re-sample

The final part of the FastSLAM algorithm is to find the posterior Y_t by completing the following steps:

1. Initialize $\mathbf{Y_t} = 0$ and $counter = 1$.

2. Draw random particle k with a probability $\propto w^{[k]}$.
3. Add $[\mathbf{x_t^{[k]}}, (\mu_{\mathbf{1,t}}^{\mathbf{k}}, \mathbf{\Sigma}_{\mathbf{1,t}}^{\mathbf{k}}), \ldots, (\mu_{\mathbf{j,t}}^{\mathbf{k}}, \mathbf{\Sigma}_{\mathbf{j,t}}^{\mathbf{k}})]$ to $\mathbf{Y_t}$.
4. Increment *counter* by 1. If $counter = M$ terminate. Otherwise, repeat from 2.

Other Well-Known Nonlinear Filtering Methods

When the motion and observation models are highly non-linear, the EKF can result in a poor performance. This is because the covariance is propagated through linearization of the underlying non-linear model. The Unscented Kalman Filter (UKF) [65] is a special case of a family of filters called Sigma-Point Kalman Filters (SPKF), and address's the above issue by using a deterministic sampling technique known as the unscented transform in which a minimal set of sample points, called "sigma points" around the mean are carefully deterministically selected. These points are then propagated through the non-linear functions, from which the mean and covariance of the estimate are then recovered to obtain a Gaussian approximation of the posterior distribution. This results in a filter that more accurately captures the true mean and covariance in comparison to the EKF [65]. Since no linearization is required in the propagation of the mean and covariance, the Jacobians of the system and measurement model do not need to be calculated, making the Unscented Kalman Filter an attractive solution and suitable for application consisting of black-box models where analytical expressions of the system dynamics are either unavailable or not easily linearized [94]. Another extension to the EKF is the Iterated EKF (IEKF) [11] method which attempts to improve upon EKF, by using the current estimate of the state vector to linearize the measurement equation in an iterative mode until convergence to a stable solution.

Monte Carlo localization (MCL) [29] is another well known filtering method for robot localization utilizing a particle filter (similar to FastSLAM). Assuming that the map of the environment is provided, the method estimates the pose of a robot as it moves and senses the environment. The algorithm uses a particle filter to represent the distribution of likely states, with each particle representing a possible state, i.e. a hypothesis of where the robot is [109]. The algorithm typically starts with a uniform random distribution of particles over the configuration space, meaning the robot has no information about its location is and assumes it is equally likely to be at any point in space. When the robot moves, it shifts the particles by predicting its new state after the motion. When sensor information is obtained, the particles are re-sampled based on recursive Bayesian estimation, i.e. how well the actual sensed data correlate with the predicted state. Ultimately, the particles should converge towards the actual position of the robot [109]. Particle filter (PF) based estimation techniques, In contrast to Kalman filtering based techniques, are able to represent multi-modal distributions and thus can re-localize a robot in the kidnapped situation (when the localization estimation fails and a robot is lost). The particle filter has some similarities with the UKF in that it transforms a set of points via known nonlinear equations and combines the results to estimate the posterior distribution. However, in the particle filter the points are selected randomly, whereas in the UKF the points are chosen based on a particular method. As such, the number of points used in a particle filter generally needs to be much greater than the number of points in a UKF, in an attempt to propagate an accurate and possibly non-Gaussian distribution of the state [94]. Another difference between the two filters is that the estimation error in a UKF does not converge to zero, whereas the estimation error in a particle filter converges to zero as the number of particles approaches infinity (at the expense of computational complexity) [105].

Another method that is suitable for problems with a large number of variables is the Ensemble Kalman Filter [43] (EnKF). EnKFs represent the distribution of the system state using a collection of state vectors, called an ensemble, and replace the covariance matrix by the sample covariance computed from the ensemble. EnKFs are related to the particle filters such that a particle is identical to an ensemble member. The EnKF differs from the particle filter in that although the EnKF uses full non-linear dynamics to propagate the errors, the EnKF assumes that all probability distributions involved are Gaussian. In addition, EnKFs generally evolve utilizing a small number of ensembles (typically 100 ensembles), as such, making it viable solution for real time applications. The main difference between the EnKF and the UKF is that in EnKF, the samples are selected heuristically, whereas samples are selected deterministically in the UKF (and Sigma-Point Kalman Filters generally).

Distributed Filtering Methods

The aforementioned filtering methods are based on a single centralized filter in which the entire system state must be reconfigured when the feature points change, a problem that is particularly evident when mapping dynamic environments. This causes an exponential growth in computational computation and difficulties to find potential faults. In order to tackle those limitations, Won et al. proposed a visual SLAM method based on using a distributed particle filter [114]. As opposed to the centralized particle filter, the distribute SLAM system divides the filter to feature point blocks and landmark block. Their simulation results showed that the distributed SLAM system has a similar estimation accuracy and requires only one-fifth of the computational time when compared to the centralized particle filter. Rigatos et al. [95] outline the problem of multiple autonomous systems (unmanned surface vessels) tracking and perusing a target (another vessel). They proposed a solution for the problem of distributed con-

trol of cooperating the unmanned surface vessels (USVs). The distributed control aims at achieving the synchronized convergence of the USVs towards the target and at maintaining the cohesion of the group, while also avoiding collisions at the same time. The authors also propose new nonlinear distributed filtering approach called "Derivative-free distributed nonlinear Kalman Filter". The filter is comprised of fusing states estimates of the target's motion which are provided by local nonlinear Kalman filters performed by the individual vessels. They showed in their simulation evaluation that their method was both faster and more accurate than other well known distributed filtering techniques such as the Unscented Information Filter (UIF) and Extended Information Filter (EIF). Complete mapping of the environment becomes very difficult when environment is dynamic or stochastic as in the case of moving obstacles. For such problems the secure autonomous navigation of the robot can be assured with the use of motion planning algorithms such as the one described above [95], in which the collisions risk is minimized.

Data Fusion Using Filtering Methods

Data fusion is the process of combing information from a number of different sources/sensors in order to provide a robust and complete description of an environment or process of interest, such that the resulting information has less uncertainty than would be possible when those sources were used individually. Data fusion methods play a very important role in autonomous systems and mobile robotics. In principle, automated data fusion processes allow information to be combined to provide knowledge of sufficient richness and integrity that decisions may be formulated and executed autonomously [37]. The Kalman filter has a number of features which make it ideally suited to dealing with complex multi-sensor estimation and data fusion problems. In particular, the explicit description of process and observations allows a wide variety of different sensor models to be included within the basic method. In addition, the consistent use of statistical measures of uncertainty makes it possible to quantitatively evaluate the role of each sensor towards the overall performance of the system. The linear recursive nature of the algorithm ensures that its application is simple and efficient. As such, Kalman filters (and EKFs for nonlinear applications) have become very common tools for solving various data fusion problems [37]. Other filters that have been previously discussed may also be similarly used for data fusion such as in [20], where they outline the problem of the unreliability of Global Positioning Systems in particular situations. As such, propose a solution that utilizes a Particle Filter for fusing data from multiple sources in order to compensate for the loss of information provided by the GPS.

Fundamental Components in V-SLAM and VO

Camera Modeling and Calibration

Perspective Projection

A camera model is a function which maps our 3D world onto a 2D image plane and is designed to closely model a real-world physical camera. There are many camera models of varying complexity. In this paper, we will explain the basic and most common model: the perspective camera model. Perspective is the property that objects that are far away appear smaller than closer objects, which is the case with human vision and most real world cameras. The most common model for perspective projection is the pinhole camera model which assumes that the image is formed by the intersecting light rays from the objects passing through the center of the projection with the image plane. An illustration of the perspective projection is shown in Fig. 5. The pinhole perspective projection equation can be written as:

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = KX = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{22}$$

where u and v are the 2D image coordinates of a 3D point with coordinates X, Y and Z, after it is projected onto the image plane. λ is a depth factor, K is the intrinsic calibration matrix and contains the intrinsic parameters: f_x and f_y are the focal lengths in the x and y directions, and c_x and c_y are the 2D coordinates of the projection center. Note: when the field of view of the camera is larger than 45°, the effects

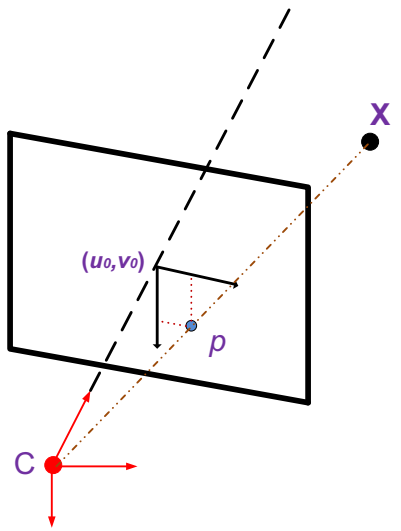


Fig. 5 Illustration of the perspective projection camera model. Light rays from point **X** intersect the image plane at point p when passing through the center of the projection

of the radial distortion may become noticeable and can be modeled using a second or higher order polynomial [101].

Intrinsic Camera Calibration

Camera calibration is the process of finding the quantities internal to the camera (intrinsic parameters) that affect the imaging process such as the image center, focal length and lens distortion parameters. Precise camera calibration is required due to having camera production errors and lower quality lenses and is very important for 3D interpretation of the image, reconstruction of world models and the robot's interaction with the world. The most popular method uses a planar checkerboard pattern with known 3D geometry. The user is required to take several images of the checkerboard at varying poses and covering the field of view of the camera as much as possible. The parameters are estimated by solving a LS minimization problem where the input data are 3D locations of the square corners on the checkerboard pattern and their corresponding 2D image coordinates. There are a number of open source software available for estimating the camera parameters such as the MATLAB camera calibration toolbox [15] and [100] and the C/C++ OpenCV calibration toolbox [16].

Feature Extraction and Matching

Vision sensors have attracted a lot of interest from researchers over the years as they provide images that are rich in information. In most cases, raw images need to be processed in order to extract the useful information. Features that are of interest range from simple point features such as corners to more elaborate features such as edges and blobs and even complex objects such as doorways and windows. Feature tracking is the process of finding the correspondences between such features in adjacent frames and is useful when small variations in motions occur between two frames. Conversely, feature matching is the process of individually extracting features and matching them over multiple frames. Feature matching is particularly useful when significant changes in the appearance of the features occur due to observing those over longer sequences [46]. In the following sections, we will briefly describe the most common feature, also called keypoint, region of interest (ROI) or point of interest (POI), extraction techniques that are used in mobile robotics applications.

Harris Corner Detector

The corner detection method described by [53] and illustrated in Fig. 6b, is based on Moravec's corner detector [82] in which a uniform region is defined to have no change in image intensities between adjacent regions in all directions, an edge which has a significant variation in directions normal to the edge and a corner by a point where image intensities have a large variation between adjacent regions in all directions. This variation in image intensities is obtained by calculating and analyzing an approximation to the SSD between adjacent regions [53].

FAST Corners

Features From Accelerated Segment Test (FAST) [96] is a corner detector based on the SUSAN method [107] in which a circle with a circumference of 16 pixels is placed around the center pixel. The brightness value of each pixel in this circle is compared to the center pixel. A region is defined as uniform, an edge or a corner based on the percentage of neighboring pixels with similar intensities to the center pixel. FAST is known to be one of the most computationally efficient feature extraction methods [96]. However it is very sensitive to noise. FAST features are illustrated in Fig. 6a.

SIFT Features

Scale Invariant Feature Transform (SIFT) is recognized by many as one of the most robust feature extraction techniques currently available. SIFT is a blob detector in which a blob can be defined as an image pattern that differs from its immediate neighborhood in terms of intensity, color and texture [104]. The main reason behind SIFT success is that it is invariant to rotation, scale, illumination and viewpoint change. Not only is SIFT a feature detector but also a feature descriptor, therefore giving each feature a distinctive measure which improves the accuracy and repeatability of feature matching between corresponding features in different images. The main downside of this method is that it is computationally expensive. SIFT features are extracted by analyzing the Difference of Gaussian (DoG) between images at different scales. A feature descriptor is computed by dividing the region around the keypoint into a 4×4 subregions. In each subregion, an orientation histogram of eight bins is constructed. This information is then stored in $4 \times 4 \times 8 = 128$ byte description vector. An example of SIFT features is illustrated in Fig. 6d. The combination of keypoint location, scale selection and the feature descriptor makes SIFT a very robust and repeatable technique for matching corresponding features in different images even under variations such as rotation, illumination and the 3D viewpoint.

SURF Features

Speeded Up Robust Features (SURF) [10] is a feature (blob) detector and descriptor that is largely inspired by SIFT. In contrast to SIFT which uses the DoG to approximate the Laplacian of Gaussian (LoG), SURF employs box filters (to

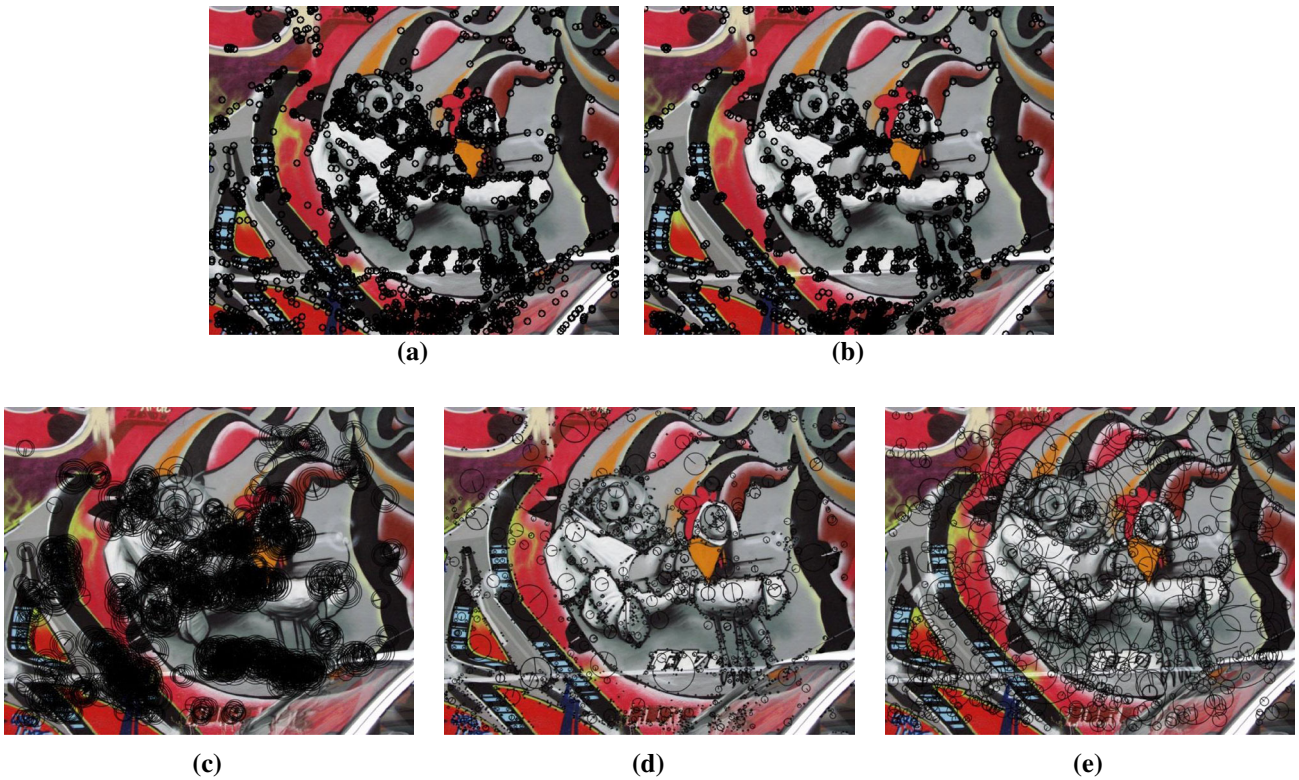


Fig. 6 Comparison of different features extraction methods using an image obtained from the Oxford dataset [80]: **a** FAST, **b** HARRIS, **c** ORB, **d** SIFT, **e** SURF. The size of the circle corresponds to the scale and the *line* corresponds to the orientation (direction of major change in intensity). Note that FAST and HARRIS are not scale and rotation invariant, as such, they are illustrated by *small circles* (single scale) and no orientation

approximate the second order Gaussian) and image integrals for the calculating the image convolution in order to approximate the Hessian matrix. To find the keypoint and its scale, analysis of the determinant of the approximated Hessian matrix is performed to find the local maximum across all scales. A robust descriptor which has similar properties to the SIFT descriptor but with less computational complexity is also introduced. For orientation assignment of the interest point Haar-wavelet responses are summed up in a circular region around the keypoint and the region around the keypoint is then divided into $n \times n$ subregions in which Haar-wavelet responses are computed and weighted with a Gaussian kernel for obtaining the descriptor vector. SURF outperforms SIFT in terms of time and computational efficiency. However, SIFT slightly outperforms SURF in terms of robustness. Surf features can be seen in Fig. 6e.

BRIEF Descriptor

Binary Robust Independent Elementary Features (BRIEF) is an efficient feature point descriptor proposed by Calonder et al. [18]. BRIEF relies on effectively classifying image patches based on a relatively small number of pairwise intensity comparisons. Based on this comparison, binary strings are created which define a region that surrounds a keypoint. BRIEF is a feature descriptor only and needs to be used in conjunction with a feature detector such as Harris corner detector, FAST, SIFT or SURF. Matching between descriptors can be achieved by finding the nearest neighbor using the efficient Hamming distance. Recently, a feature detector and descriptor named "ORB" [97] (See Fig. 6c) has been proposed which is based on the FAST feature detector and BRIEF descriptor.

3D Descriptors

3D descriptors are useful when 3D information is available (point clouds). Some of the most well-known 3D Descriptor methods are Point Feature Histogram (PFH) [98], Fast Point Feature Histogram (FPFH) [99], 3D Shape Context (3DSC) [46], Unique Shape Context (USC) [112] and Signatures of Histograms of Orientations (SHOT) [112]. The PFH captures information of the geometry surrounding every point by analyzing the difference between the directions of the normals in its local vicinity. The PFH does not only pair the query keypoint with its neighbors, but also the neighbors with themselves. As such, the PFH is computationally demanding and is not suitable for real-time applications.

The FPFH extends the PFH by only considering the direct connections between the query keypoint and its surrounding neighbors. To make up for the loss of extra connections between neighbors, an additional step after all histograms have been computed is added: the sub-PFHs of a point's neighbors are merged with its own and is weighted according to the distance between those. This provides point surface information of points as far away as 2 times the radius used. The 3D Shape Context is a descriptor that works by constructing a structure (sphere) centered at the each point, using the given search radius. The "north pole" of that sphere is pointed in the same direction as the normal at that point. Then, the sphere is divided in 3D regions (regions vary in volume in the radial direction as they are logarithmically spaced so they are smaller towards the center). A descriptor is computed by counting the number of points in each 3D region. The count is weighted by the volume of the bin and the local point density (number of points around the current neighbor). As such, the descriptor is resolution invariant to some extent. In order to account for rotation variances, the support sphere is rotated around the normal N times and the process is repeated for each, giving a total of N descriptors for a point. This procedure is computationally expensive, as such, the USC descriptor extends the 3DSC by defining a local reference frame that provides a unique orientation at each point. This procedure reduces the size of the descriptor when compared to 3DSC, since computing multiple descriptors to account for orientations is not required. SHOT descriptor is similar to 3DSC and USC in that it encodes surface information within a spherical support structure. The sphere around each keypoint is divided into 32 bins, and for each bin, a descriptor containing one variable is computed. This variable is the cosine of the angle between the normal of the keypoint and the neighboring point inside the bin. Finally, the descriptor is obtained by augmenting local histograms. Similiar to USC, SHOT descriptors are also rotation invariant.

Data Association

DA is defined as the process of associating a measurement (or feature) to its corresponding previously extracted feature. DA is one of the most important aspects of SLAM and VO. In SLAM, correct DA between the current measurements and previously stored features is vital for the SLAM convergence and obtaining a consistent map. False DA will cause the robot to think it is at a location where it is not, therefore diverging the SLAM solution. In VO, correct data association between features in successive frames is required to obtain an accurate estimation of the transformation between frames. We will describe the most common methods used for DA.

Data Association Based on Visual Similarity

Sum of Square Differences SSD is a similarity measure that uses the squared differences between corresponding pixels in two images. SSD has the following formula:

$$SSD = \sum_{i=-n}^{n} \sum_{j=-n}^{n} ((I_1(u_1+i, v_1+j) - I_2(u_2+i, v_2+j))^2 \tag{23}$$

where I_1 and I_2 are the first and second image patches centered at (u_1, v_1) and (u_2, v_2) respectively.

Another similarity measure similar to SSD is the Sum of Absolute Differences (SAD) in which the absolute differences between corresponding pixels is used instead of the squared difference. SSD is a more robust similarity measure compared to SAD. However, it is less computationally efficient.

Normalized Cross Correlation Normalized Cross Correlation (NCC) is one of the most common and accurate techniques for finding the similarity between two image patches. The technique works by summing the product of intensities of the corresponding pixels in two image patches and normalizing this summation by a factor based on the intensity of the pixels. NCC has the following formula:

$$NCC = \frac{\sum_{i=-n}^{n} \sum_{j=-n}^{n} I_1(u_1+i, v_1+j) \cdot I_2(u_2+i, v_2+j)}{\sqrt{\left(\sum_{i=-n}^{n} \sum_{j=-n}^{n} I_1(u_1+i, v_1+j)^2\right) \cdot \left(\sum_{i=-n}^{n} \sum_{j=-n}^{n} I_2(u_2+i, v_2+j)^2\right)}} \tag{24}$$

The higher the NCC value, the more similarity between the compared image patches. Note: Because of the normalization, NCC outperforms both SAD and SSD in cases where there is a variation in the brightness levels between the compared patches. However, NCC is a slower method since it involves more complex operations including multiplications, divisions and square root operations.

RANSAC Refinement

The above steps for feature matching and DA usually result in false matches (outliers) mainly due to noisy sensors and lightning and view point variation. The Random Sampling Consensus (RANSAC) is the most common tool used for

estimating a set of parameters of a mathematical model from a set of observed data which contains outliers. An example of the refined matches using RANSAC can be seen in Fig. 7d–f. The RANSAC method for refining the matches between two frames generally consists of the following steps:

1. Randomly select the minimum number of matched features (e.g. points in one frame and their matches in the next) required to estimate the transformation (i.e. three points for a 6DOF system using the 3D to 2D reprojection error estimation method).
2. Estimate the transformation (rotation and translation parameters) using the selected points.
3. Apply the obtained transformation to the remaining points.
4. Find the distance (l^2 distance is commonly used) D between the transformed points and their corresponding matches.
5. Pairs with D less than a predefined threshold τ are considered inliers.
6. Count the total number of inliers obtained by this transformation.
7. Repeat the previous steps n times.
8. Transformation with the highest number of inliers is assumed to be the correct transformation.
9. Re-estimate the transformation using LS applied to all the inliers.

RANSAC is a non-deterministic algorithm in the sense that it produces a reasonable result only with a certain probability, with this probability increasing as more iterations are allowed. A number of RANSAC extensions have been proposed over the years. Chum et al. [25] proposed to guide the sampling procedure if priori information regarding the input data is known, i.e. whether a sample is likely to be an inlier or an outlier. The proposed approach is called PROgressive SAmple Consensus (PROSAC). Similarly, SAC-IA [99] proposed the selection of points based on those with the most similar feature histograms. Another example is the prerejection RANSAC algorithm [17]. This method adds an additional verification step to the standard RANSAC algorithm which eliminates some false matches by analyzing their geometry.

Loop Closure

Loop closure (also known as cycle closure) detection is the final refinement step and is vital for obtaining a globally consistent SLAM solution especially when localizing and mapping over long periods of time. Loop closure is the process of observing the same scene by non-adjacent frames and adding a constraint between them, therefore considerably reducing the accumulated drift in the pose estimate. To illustrate this, imagine a case where a robot moves in a closed looped corridor while it continuously observes features of its environment. By the time the robot goes back to its initial position, the error in the robot's position estimate has accumulated and is offset by a finite distance from the actual position. Now assume that the robot realizes it is observing the initial scene, therefore it adds a constraint between the current frame and the initial frame (based on the features location estimates in the scene) and hence, reduces the overall drift in position and map estimates.

The most basic form of loop closure detection is to match the current frame to all the previous frames using feature matching techniques. This approach is computationally very expensive (computational expense increases linearly with the number of estimates [42]) due to the fact that the number of frames are increased over time and matching the current frame with all the previous frames is not suitable for real-time applications. A solution to this problem is to define key frames (a subset of all the previous frames) and compare the current frame with the key frames, only. The simplest form of key frame selection is to select every nth frame. Kerl et al. [69] proposed a key-frame selection method based on a differential entropy measure. Other solutions such as the one used by [56] is to perform a RANSAC alignment step (similar to the one described in "RANSAC Refinement" section) in between the current frame and a key frame. A new key frame is selected only when the number of inliers detected by the RANSAC alignment step is less than a predefined threshold (making it a new scene). In order to improve the computational efficiency and avoid the RANSAC step between each frame and the key frame, they proposed adding a key frame when the accumulated rotation or translation exceeds a predefined threshold (10 ° in rotation or 25 cm in translation [8]). In order to filter the loop closure frame candidates, they used a place recognition approach based on the vocabulary tree described by Nister and Stewenius [87] in which the feature descriptors of the candidate key frames are hierarchically quantized and are represented by a "bag of visual words" (B.O.W). Another well-known approach that is similar to B.O.W is the Vector of Locally Augmented Descriptors (VLAD) [62]. As opposed to B.O.W which constructs a histogram for each image simply based on counting the number of occurrences of the visual words in the image, VLAD constructs a descriptor by summing the residuals between each image descriptor and the associated visual word resulting in a more robust visual similarity recognition approach [62]. Alternatively, instead of selecting key frames or matching to all the previous frames, Endres et al. [41] also presented a loop closure method that matched the current frame to 20 frames consisting of the three most recent frames and uniformly sampled previous frames, therefor resulting in a more computationally efficient approach.

RGB-D SLAM

RGB-D SLAM (or RGB-D mapping) is a V-SLAM method that uses RGB-D sensors for localization and mapping. RGB-D sensors are ones that provide depth information in addition to the color information. Although RGB-D sensors have been popularized by the Microsoft Kinect and Asus Xtion PRO when they were released in 2010, older RGB-D SLAM solutions had already existed. For example, Biber et al. [13] proposed a SLAM method that uses a combination of a laser scanner and a panoramic camera. The more recent RGB-D cameras such as the Kinect are based on the structured light approach which capture RGB color images and provide pixel depth information. Localization and mapping using RGB-D cameras have become an active area of research mainly due to the low cost of these cameras which provide very useful information for mapping. The aim of RGB-D SLAM is to obtain a dense 3D representation of the environment while keeping track of the camera pose at the same time. Using both color and depth information has a number of advantages over using either color or depth. Some of those reasons are outlined below:

1. Associating each pixel with a depth value provides a dense and colored point cloud. This allows for the visualization of dense 3D reconstructed environments.
2. RGB-D sensors provide metric information, thus overcoming the scale ambiguity problem in image based SLAM systems.
3. Some environments contain limited texture. as such, the availability of depth information allows the SLAM system to fall back on depth only approaches such as ICP.
4. Other environments may contain limited structure. In those cases, the system can fall back on using RGB information for matching and registration.

In the following sections, we will describe the basic steps that constitute the RGB-D SLAM method. Although there is an intensity based RGB-D SLAM approach [5], the main approach is based on using features and we will describe this approach in detail (similar to [34,41,56]) in the following sections. A system overview of a typical RGBD-SLAM system is shown in Fig. 8.

Feature Extraction, Matching and Refinement

The first step is to extract and match features across sequential images. We discussed feature extraction techniques such as SIFT, SURF and ORB in "Feature Extraction and Matching" section in which points of interest such as edges and corners are detected in the images. The depth information provided by the RGB-D camera are then used to obtain these points in 3D (point clouds). Matching extracted features between sequential frames is then performed using an appropriate DA technique. If the descriptors are appearance based (based on the intensity value of pixels in the neighborhood of a feature), a similarity measure such as NCC or SSD is required. Matching between SIFT and SURF features is performed by a distance based approach such as the euclidean distance nearest neighbor. The Hamming distance is usually used to match between features with BRIEF descriptors. Matching features using SIFT, SURF and BRIEF descriptors can be seen in Fig. 7a, b and c respectively. The next step is to estimate the camera's relative transformation (with respect to the previous location) using the feature matches between sequential frames. This is a typical VO problem and can be solved using one of the motion estimation methods discussed in "Motion Estimation" section. The most common method is to estimate the motion based on the 3D–2D re-projection error. In an ideal case where all features are matched correctly between frames, this overdetermined system can be solved by a LS based optimization method. However, because of noise, illumination changes and other factors, not all matches are correct (as illustrated in Fig. 7a–c) and estimating the transformation in the presence of these false matches (outliers) is required. As we explained earlier, RANSAC is the most common tool used for estimating a set of parameters in the presence of outliers. The refined matches can be seen in Fig. 7d–f.

Refining the Transformation Using ICP

The ICP is a common technique in computer vision for aligning 3D points obtained by different frames. ICP iteratively aligns the closest points (correspondences found using a fast nearest neighbor approach) in two clouds of points until convergence (change in the estimated transformation becomes small between iterations or maximum number of iterations is reached).

ICP has been shown to be effective when the two clouds are already nearly aligned [56]. As such, a good initial estimate would help the convergence of the ICP. In situations in which a bad estimate is used, convergence could occur at an incorrect local minimum, producing a poor estimate of the actual transformation. In RGB-D SLAM, the RANSAC transformation estimate can be used as the initial guess in the ICP algorithm. In cases where RANSAC fails to obtain a reliable transformation (number of inliers are less than a threshold) which may be caused by the low number of extracted features from a texture-less scene, the transformation is fully estimated using the ICP algorithm and using an initial predefined transformation (usually set to the identity matrix).

Global Transformation

The previous steps provide an estimate of the camera motion transformation between two frames. In order to obtain a

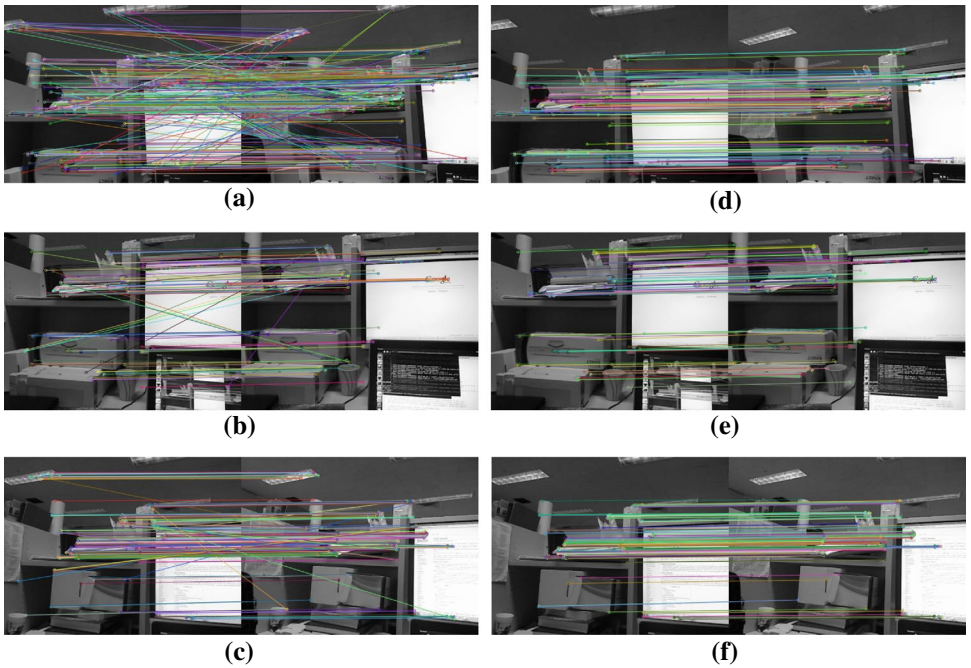


Fig. 7 Features matching methods: **a** SIFT matches, **b** SURF matches, **c** BRIEF matches, **d–f** refined matches using RANSAC

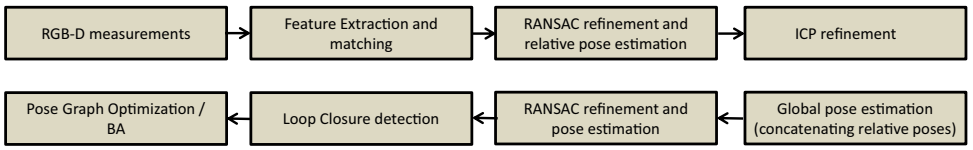


Fig. 8 A block diagram showing the main components of a typical RGBD-SLAM system

global estimate with respect to a reference frame, all the transformations up to the current time need to be concatenated. Let us denote the transformation between two frames as:

$$\mathbf{T_{k,k-1}} = \begin{bmatrix} \mathbf{R_{k,k-1}} & \mathbf{t_{k,k-1}} \\ 0 & 1 \end{bmatrix} \tag{25}$$

where $\mathbf{R_{k,k-1}} \in SO(3)$ is the rotation matrix and $\mathbf{t_{k,k-1}} \in \Re^{3\times 1}$ is the translation vector between frames taken at time-steps k and $k-1$ respectively where $k = 1 \ldots n$. The global estimate $\mathbf{G_n}$ can be calculated using the following formula:

$$\mathbf{G_n} = \mathbf{G_{n-1}} \mathbf{T_{n,n-1}} \tag{26}$$

where $\mathbf{G_{n-1}}$ is the previous global transformation with respect to an initial reference frame $\mathbf{G_0}$ at $k = 0$ and $\mathbf{T_{n,n-1}}$ is the transformation between the current frame and the previous frame.

Global Optimization

The previous steps (except reconstructing the 3D map) describe a typical VO problem which is concerned with obtaining a locally consistent estimate of the camera trajectory. In order to obtain a globally consistent estimate (in order to reduce the drift), a global refinement is required. The common approaches for this requirement are as follows.

Pose Graph Optimization

Pose graph optimization is a global optimization method used in SLAM and VO problems that are represented by a graph model. The graph model consists of nodes that correspond to the camera poses and edges which represent the constraints between the nodes, described by rigid-body transformations. The pose graph model allows the formation of constraints between non-adjacent poses by observing the same features from different locations. The goal of this optimization is to find the arrangement of poses that best satisfies those constraints. Generally, this is a non-linear LS optimization problem which can be described as:

$$\mathbf{C}^* = \underset{C}{\operatorname{argmin}} \sum_{\mathbf{i,j}} ||\mathbf{C_i} - \mathbf{T_{i,j}} \mathbf{C_j}||^2 \tag{27}$$

where $\mathbf{C}^* = \mathbf{C_1^*} \ldots \mathbf{C_n^*}$ is a vector containing all the optimized camera poses (x, y and z values). There are a number of approaches for solving the above optimization problem and we will briefly describe common ones here.

1. *GraphSLAM* GraphSLAM [111] is a full-SLAM problem which means that a posterior (robot poses and map) is calculated over the entire path $x_{1:t}$ along with the map, instead of just the current pose calculation in online-SLAM [109] (such as the EKF-SLAM). It works by representing the SLAM posterior by a graphical network which leads to a sum of non-linear quadratic constraints and optimizing the graph by taking all the constraints into account using standard optimization techniques. The result is a maximum likelihood map and a corresponding set of robot poses [109].
2. *TORO* Tree-based Network Optimizer [52] is a graph optimization method that is based on Olsen's method [94] for solving the optimization problem using a stochastic gradient descent (an optimization technique used for finding the local minimum based on the observation that a function decreases fastest in the direction of the negative gradient or it's approximation). In this approach, Olsen's method is extended by introducing a tree-based parametrization for the nodes in the graph which results in a faster convergence and better computational efficiency. This optimization technique is used by [55] in their RGB-D SLAM method.
3. *HOG-MAN* Hierarchical Optimizations on Manifolds for Online 2D and 3D mapping [50] is an efficient pose-graph optimization approach in which the problem can be modeled based on different levels of abstraction. Instead of targeting the optimization of the full problem (all the nodes of the graph), this method only optimizes a simplified problem which contains all the required relevant information and is constructed incrementally. This is performed by updating only the parts of the map that are required for DA. This approach has the advantage of reducing the computational complexity of the SLAM problem while preserving the global consistency [50].
4. g^2o General Framework for Graph Optimization [72] is an open-source framework for graph-based optimization that is applicable to both SLAM and BA problems. It includes a number of linear solvers such as the preconditioned conjugate gradient (PCG), CHOLMOD and CSparse. g^2o allows the user to specifically re-define the error function to be minimized and choose the method for solving the linearized problem. This technique is used by [41] in their RGB-D SLAM optimization step.
5. *Ceres* The Google Ceres Solver [2] a powerful open source C++ library for modeling and solving large, complicated optimization problems. Ceres applications range from rather simple fitting curves and robust regression to the more complicated pose graph optimization and BA problems. Ceres's biggest advantage is its simplicity and user friendliness as it had been designed so that the user can easily build and modify the objective function (desired cost function), one term at a time. And to do so without worrying about how the solver is going to deal with the resulting changes in the sparsity/structure of the underlying problem [2]. Ceres offers numerous options to the user, such using a robust cost functions to reduce the influence of outliers. In addition Ceres supports many solvers such as Levenberg–Marquardt, Powells Dogleg, and Subspace dogleg methods in which the user may choose from depending on the size, sparsity structure, time and memory budgets, and solution quality requirements.

Global/Local Bundle Adjustment Optimization

BA is another global optimization method similar to pose-graph optimization and is commonly used in computer vision applications. BA jointly optimizes the camera pose and the 3D structure parameters that are viewed and matched over multiple frames by minimizing a cost function. BA is usually called Global Bundle Adjustment (GBA) when taking all frames into consideration, and Local Bundle Adjustment (LBA) in which the optimization is applied over a number of fixed frames (a window). In general, the cost function to be minimized can be formulated as [45]:

$$\arg\min_{X^i, C_k} \sum_{i,k} ||\mathbf{p_k^i} - g(\mathbf{X^i}, \mathbf{C_k})||^2 \tag{28}$$

where $\mathbf{X^i}$ is the ith 3D point, $\mathbf{p_k^i}$ is its corresponding image point observed in the kth frame and $g(\mathbf{X^i}, \mathbf{C_k})$ is the re-projection function according to the camera pose $\mathbf{C_k}$.

GBA generally results in a more accurate optimization when compared to LBA since it takes all previous frames into account in the optimization. However, optimizing over a window limits the number of parameters that are solved in the optimization and therefore LBA is more suitable for real-time applications in which the computational resources are limited.

Loop Closure for RGB-D SLAM

Similar to other types of SLAM, as we discussed in "Loop Closure" section, a loop closure step is required to obtain a globally consistent map and reduce the drift.

Map Representation

The previous global optimization and loop closure steps produces a globally consistent trajectory (i.e. each node/frame is associated with an optimized global pose). In order to build the map, the points are projected to 3D using the following equations:

$$X = (u - c_x) * Z/f_x \quad (29)$$

$$Y = (v - c_y) * Z/f_y \quad (30)$$

where (u, v) is the image coordinate of an extracted visual feature and (X, Y, Z) is its projected 3D coordinate in the camera optical frame. Z is obtained from the depth image which is provided by the RGB-D sensor. f_x and f_y are the focal lengths in the horizontal and vertical axises respectively and (c_x, c_y) is the 2D coordinate of the camera optical center. After applying this projection, 3D points are then transformed to a common reference frame using the optimized trajectory.

Conclusion

In this paper, a extensive theoretical review of solutions for the VO and visual SLAM (V-SLAM) problems is presented. We initially outlined the history of the research undertaken in those areas and discussed the localization and mapping problems, separately. The SLAM problem was discussed and the basic framework for solving this problem was presented. We also discussed the fundamental techniques used in both VO and V-SLAM such as feature extraction and DA methods. We finally presented the recently popularized RGB-D SLAM approach for solving the V-SLAM problem. The RGB-D SLAM approach consists of combining visual odometry methods for finding the relative transformation between frames, and SLAM methods for global optimization and loop closure. The paper paves the way for new researchers in this area to have clear understanding of the subtleties of different solutions and would help practitioners to choose the best available approach for these tasks.

Acknowledgments The authors would like to acknowledge the financial support provided by the Australian Postgraduate Award (APA).

References

1. Abrate, F., Bona, B., Indri, M.: Experimental EKF-based slam for mini-rovers with IR sensors only. In: Preceedings of Third European Conference on Mobile Robots (2007)
2. Agarwal, S., Mierle, K.: Others: Ceres solver. http://ceres-solver.org
3. Alcantarilla, P., Bergasa, L., Dellaert, F.: Visual odometry priors for robust ekf-slam. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 3501–3506 (2010)
4. Angeli, A., Doncieux, S., Meyer, J.A., Filliat, D.: Visual topological slam and global localization. In: IEEE International Conference on Robotics and Automation, 2009 (ICRA'09), pp. 4300–4305 (2009)
5. Audras, C., Comport, A., Meilland, M., Rives, P.: Real-time dense appearance-based slam for RGB-D sensors. In: Australasian Conference on Robotics and Automation (2011)
6. Audras, C., Comport, A., Meilland, M., Rives, P.: Real-time dense RGB-D localisation and mapping. In: Australian Conference on Robotics and Automation (ACRA). Monash University, Australia (2011)
7. Bab-Hadiashar, A., Suter, D.: Robust optic flow computation. Int. J. Comput. Vis. **29**(1), 59–77 (1998)
8. Bachrach, A., Prentice, S., He, R., Henry, P., Huang, A., Krainin, M., Maturana, D., Fox, D., Roy, N.: Estimation, planning, and mapping for autonomous flight using an rgb-d camera in gps-denied environments. Int. J. Robot. Res. **31**(11), 1320–1343 (2012)
9. Bailey, T.: Mobile robot localisation and mapping in extensive outdoor environments. PhD Thesis, The University of Sydney (2002)
10. Bay, H., Tuytelaars, T., Van Gool, L.: Surf: speeded up robust features. In: ECCV on Computer Vision 2006, pp. 404–417 (2006)
11. Bell, B.M., Cathey, F.W.: The iterated Kalman filter update as a Gauss–Newton method. IEEE Trans. Autom. Control **38**(2), 294–297 (1993)
12. Besl, P., McKay, N.: A method for registration of 3-D shapes. IEEE Trans. Pattern Anal. Mach. Intell. **14**(2), 239–256 (1992)
13. Biber, P., Andreasson, H., Duckett, T., Schilling, A.: 3D modeling of indoor environments by a mobile robot with a laser scanner and panoramic camera. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, 2004 (IROS 2004), Proceedings, vol. 4, pp. 3430–3435 (2004)
14. Borenstein, J., Koren, Y.: The vector field histogram-fast obstacle avoidance for mobile robots. IEEE Trans. Robot. Autom. **7**(3), 278–288 (1991)
15. Bouguet, J.Y.: Camera calibration toolbox for Matlab. http://www.vision.caltech.edu/bouguetj/calib.doc/index.html
16. Bradski, G.: The OpenCV Library. Dr. Dobb's Journal of Software Tools (2000)
17. Buch, A.G., Kraft, D., Kämäräinen, J.K., Petersen, H.G., Krüger, N.: Pose estimation using local structure-specific shape and appearance context. In: IEEE International Conference on Robotics and Automation, 2013 (ICRA)
18. Calonder, M., Lepetit, V., Strecha, C., Fua, P.: Brief: Binary robust independent elementary features. In: ECCV on Computer Vision 2010, pp. 778–792 (2010)
19. Campbell, J., Sukthankar, R., Nourbakhsh, I., Pahwa, A.: A robust visual odometry and precipice detection system using consumergrade monocular vision. In: Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA 2005), pp. 3421–3427 (2005)
20. Caron, F., Davy, M., Duflos, E., Vanheeghe, P.: Particle filtering for multisensor data fusion with switching observation models: application to land vehicle positioning. IEEE Trans. Signal Process. **55**(6), 2703–2719 (2007)
21. Chatila, R., Laumond, J.: Position referencing and consistent world modeling for mobile robots. In: IEEE International Conference on Robotics and Automation, Proceedings, vol. 2, pp. 138–145 (1985)
22. Cheng, Y., Maimone, M., Matthies, L.: Visual odometry on the mars exploration rovers. In: 2005 IEEE International Conference on Systems, Man and Cybernetics, vol. 1, pp. 903–910 (2005)
23. Choi, H., Kim, D., Hwang, J., Kim, E., Kim, Y.: Cv-slam using ceiling boundary. In: 2010 the 5th IEEE Conference on Industrial Electronics and Applications (ICIEA), pp. 228–233 (2010)
24. Choi, H., Kim, D., Hwang, J., Park, C., Kim, E.: Efficient simultaneous localization and mapping based on ceiling-view: ceiling boundary feature map approach. Adv. Robot. **26**(5–6), 653–671 (2012)
25. Chum, O., Matas, J.: Matching with PROSAC-progressive sample consensus. In: IEEE Computer Society Conference on Computer

Vision and Pattern Recognition, 2005 (CVPR 2005), vol. 1, pp. 220–226 (2005)
26. Cole, D., Newman, P.: Using laser range data for 3D slam in outdoor environments. In: IEEE International Conference on Robotics and Automation, 2006 (ICRA 2006), Proceedings, pp. 1556–1563 (2006). doi:10.1109/ROBOT.2006.1641929
27. Comport, A., Malis, E., Rives, P.: Accurate quadrifocal tracking for robust 3D visual odometry. In: 2007 IEEE International Conference on Robotics and Automation, pp. 40–45 (2007)
28. Davison, A.: Real-time simultaneous localisation and mapping with a single camera. In: Ninth IEEE International Conference on Computer Vision, 2003, Proceedings, pp. 1403–1410 (2003)
29. Dellaert, F., Fox, D., Burgard, W., Thrun, S.: Monte carlo localization for mobile robots. In: Proceedings of IEEE International Conference on Robotics and Automation, 1999, vol. 2, pp. 1322–1328 (1999)
30. Dellaert, F., Kaess, M.: Square root SAM: simultaneous localization and mapping via square root information smoothing. Int. J. Robot. Res. **25**(12), 1181–1203 (2006)
31. Dornhege, C., Kleiner, A.: Visual odometry for tracked vehicles. In: Proceedings of the IEEE International Workshop on Safety, Security and Rescue Robotics (SSRR) (2006)
32. Doucet, A., De Freitas, N., Murphy, K., Russell, S.: Rao–Blackwellised particle filtering for dynamic bayesian networks. In: Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence, pp. 176–183. Morgan Kaufmann Publishers, San Francisco (2000)
33. Dryanovski, I., Valenti, R.G., Xiao, J.: Fast visual odometry and mapping from RGB-D data. In: 2013 IEEE International Conference on Robotics and Automation (ICRA), pp. 2305–2310 (2013)
34. Du, H., Henry, P., Ren, X., Cheng, M., Goldman, D., Seitz, S., Fox, D.: Interactive 3D modeling of indoor environments with a consumer depth camera. In: Proceedings of the 13th International Conference on Ubiquitous Computing, pp. 75–84. ACM, New York (2011)
35. Dudek, G., Jenkin, M., Milios, E., Wilkes, D.: Robotic exploration as graph construction. IEEE Trans. Robot. Autom. **7**(6), 859–865 (1991)
36. Durrant-Whyte, H., Bailey, T.: Simultaneous localization and mapping: part I. IEEE Robot. Autom. Mag. **13**(2), 99–110 (2006)
37. Durrant-Whyte, H., Henderson, T.C.: Multisensor data fusion. In: Springer Handbook of Robotics, pp. 585–610. Springer, Heidelberg (2008)
38. Durrant-Whyte, H., Rye, D., Nebot, E.: Localization of autonomous guided vehicles. In: Proceedings of the IEEE International Conference on Robotics and Automation, vol. 7, pp. 613–625 (1996)
39. Dutton, B., Maloney, E.: Dutton's navigation & piloting. Naval Institute Press, Annapolis (1978). http://books.google.com.au/books?id=-A5UAAAAMAAJ
40. Elfes, A.: Occupancy grids: a stochastic spatial representation for active robot perception. In: Proceedings of the Sixth Conference on Uncertainty in AI, vol. 2929 (1990)
41. Endres, F., Hess, J., Engelhard, N., Sturm, J., Cremers, D., Burgard, W.: An evaluation of the RGB-D slam system. In: 2012 IEEE International Conference on Robotics and Automation (ICRA), pp. 1691–1696 (2012)
42. Endres, F., Hess, J., Sturm, J., Cremers, D., Burgard, W.: 3-D mapping with an RGB-D camera. In: IEEE Transactions on Robotics (2014)
43. Evensen, G.: The ensemble Kalman filter: theoretical formulation and practical implementation. Ocean Dyn. **53**(4), 343–367 (2003)
44. Fischler, M., Bolles, R.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun. ACM **24**(6), 381–395 (1981)
45. Fraundorfer, F., Scaramuzza, D.: Visual odometry: part II: matching, robustness, optimization, and applications. IEEE Robot. Autom. Mag. **19**(2), 78–90 (2012)
46. Frome, A., Huber, D., Kolluri, R., Bülow, T., Malik, J.: Recognizing objects in range data using regional point descriptors. In: ECCV on Computer Vision 2004, pp. 224–237. Springer, Heidelberg (2004)
47. Gibson, J.: The Senses Considered as Perceptual Systems. Houghton Mifflin Co, Boston (1966)
48. Grisetti, G., Grzonka, S., Stachniss, C., Pfaff, P., Burgard, W.: Efficient estimation of accurate maximum likelihood maps in 3D. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, 2007 (IROS 2007), pp. 3472–3478 (2007)
49. Grisetti, G., Kummerle, R., Stachniss, C., Burgard, W.: A tutorial on graph-based slam. IEEE Intell. Transp. Syst. Mag. **2**(4), 31–43 (2010)
50. Grisetti, G., Kummerle, R., Stachniss, C., Frese, U., Hertzberg, C.: Hierarchical optimization on manifolds for online 2D and 3D mapping. In: 2010 IEEE International Conference on Robotics and Automation (ICRA), pp. 273–278 (2010)
51. Grisetti, G., Stachniss, C., Burgard, W.: Improving grid-based slam with Rao-Blackwellized particle filters by adaptive proposals and selective resampling. In: Robotics and Automation, 2005 (ICRA 2005), pp. 2432–2437 (2005)
52. Grisetti, G., Stachniss, C., Grzonka, S., Burgard, W.: A tree parameterization for efficiently computing maximum likelihood maps using gradient descent. In: Proceedings of Robotics: Science and Systems (RSS) (2007)
53. Harris, C., Stephens, M.: A combined corner and edge detector. In: Alvey Vision Conference, vol. 15, p. 50, Manchester (1988)
54. Helmick, D., Cheng, Y., Clouse, D., Matthies, L., Roumeliotis, S.: Path following using visual odometry for a Mars Rover in high-slip environments. In: Proceedings of 2004 IEEE Aerospace Conference, vol. 2, pp. 772–789 (2004). doi:10.1109/AERO.2004.1367679
55. Henry, P., Krainin, M., Herbst, E., Ren, X., Fox, D.: Rgb-d mapping: using depth cameras for dense 3D modeling of indoor environments. In: The 12th International Symposium on Experimental Robotics (ISER) (2010)
56. Henry, P., Krainin, M., Herbst, E., Ren, X., Fox, D.: RGB-D mapping: using kinect-style depth cameras for dense 3D modeling of indoor environments. Int. J. Robot. Res. **31**(5), 647–663 (2012)
57. Hu, G., Huang, S., Zhao, L., Alempijevic, A., Dissanayake, G.: A robust RGB-D slam algorithm. In: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1714–1719 (2012)
58. Hwang, S., Song, J.: Stable monocular slam with indistinguishable features on estimated ceiling plane using upward camera. In: International Conference on Control, Automation and Systems, 2008 (ICCAS 2008), pp. 704–709 (2008)
59. Hwang, S., Song, J., Kim, M.: Robust extraction of arbitrary-shaped features in ceiling for upward-looking camera-based slam. World Cong. **18**, 8165–8170 (2011)
60. Irani, M., Rousso, B., Peleg, S.: Recovery of ego-motion using region alignment. IEEE Trans. Pattern Anal. Mach. Intell. **19**(3), 268–272 (1997)
61. Izadi, S., Kim, D., Hilliges, O., Molyneaux, D., Newcombe, R., Kohli, P., Shotton, J., Hodges, S., Freeman, D., Davison, A., et al.: Kinectfusion: real-time 3D reconstruction and interaction using a moving depth camera. In: Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology, pp. 559–568. ACM, New York (2011)
62. Jégou, H., Perronnin, F., Douze, M., Sanchez, J., Perez, P., Schmid, C.: Aggregating local image descriptors into compact codes. IEEE Trans. Pattern Anal. Mach. Intell. **34**(9), 1704–1716 (2012)

63. Jeong, W., Lee, K.: CV-slam: a new ceiling vision-based slam technique. In: 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2005 (IROS 2005), pp. 3195–3200 (2005)
64. Jeong, W., Lee, K.: Visual slam with line and corner features. In: 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2570–2575 (2006)
65. Julier, S.J., Uhlmann, J.K.: New extension of the Kalman filter to nonlinear systems. In: AeroSense'97. International Society for Optics and Photonics, pp. 182–193 (1997)
66. Kaess, M., Ranganathan, A., Dellaert, F.: ISAM: incremental smoothing and mapping. IEEE Trans. Robot. **24**(6), 1365–1378 (2008)
67. Kaess, M., Ni, K., Dellaert, F.: Flow separation for fast and robust stereo odometry. In: IEEE International Conference on Robotics and Automation, 2009 (ICRA'09), pp. 3539–3544 (2009)
68. Keller, M., Lefloch, D., Lambers, M., Izadi, S., Weyrich, T., Kolb, A.: Real-time 3D reconstruction in dynamic scenes using point-based fusion. In: 2013 International Conference on 3DTV-Conference, pp. 1–8 (2013)
69. Kerl, C., Sturm, J., Cremers, D.: Dense visual slam for RGB-D cameras. In: 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2100–2106 (2013)
70. Kim, S., Oh, S.: Slam in indoor environments using omni-directional vertical and horizontal line features. J. Intell. Robot. Syst. **51**(1), 31–43 (2008)
71. Kleeman, L.: Advanced sonar and odometry error modeling for simultaneous localisation and map building. In: Proceedings of 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2003 (IROS 2003), vol. 1, pp. 699–704 (2003)
72. Kummerle, R., Grisetti, G., Strasdat, H., Konolige, K., Burgard, W.: G2o a general framework for graph optimization. In: 2011 IEEE International Conference on Robotics and Automation (ICRA), pp. 3607–3613 (2011)
73. Lacroix, S., Mallet, A., Chatila, R., Gallo, L.: Rover self localization in planetary-like environments. In: Artificial Intelligence, Robotics and Automation in Space, pp. 440–433 (1999)
74. Lepetit, V., Moreno-Noguer, F., Fua, P.: EPNP: an accurate O(n) solution to the PnP problem. Int. J. Comput. Vis. **81**(2), 155–166 (2009)
75. Longuet-Higgins, H.: A computer algorithm for reconstructing a scene from two projections. In: Fischler, M.A., Fischler, O. (eds.) Readings in Computer Vision: Issues, Problems, Principles, and Paradigms, pp. 61–62. Morgan Kaufmann, San Francisco (1987)
76. Lourakis, M., Argyros, A.: SBA: a software package for generic sparse bundle adjustment. ACM Trans. Math. Softw. (TOMS) **36**(1), 2 (2009)
77. Lowe, D.: Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vis. **60**(2), 91–110 (2004)
78. Mahon, I., Williams, S., Pizarro, O., Johnson-Roberson, M.: Efficient view-based slam using visual loop closures. IEEE Trans. Robot. **24**(5), 1002–1014 (2008). doi:10.1109/TRO.2008.2004888
79. Matthies, L.: Error modeling in stereo navigation. IEEE J. Robot. Autom. **3**, 239–250 (1987)
80. Mikolajczyk, K., Schmid, C.: A performance evaluation of local descriptors. IEEE Trans. Pattern Anal. Mach. Intell. **27**(10), 1615–1630 (2005)
81. Montemerlo, M., Thrun, S., Koller, D., Wegbreit, B., et al.: Fastslam: a factored solution to the simultaneous localization and mapping problem. In: Proceedings of the National Conference on Artificial Intelligence, pp. 593–598. AAAI Press/MIT Press, Menlo Park/Cambridge (1999/2002)
82. Moravec, H.: Obstacle avoidance and navigation in the real world by a seeing robot rover. In: Technical Report. CMU-RI-TR-80-03, Robotics Institute, Carnegie Mellon University & Doctoral Dissertation, Stanford University, CMU-RI-TR-80-03 (1980)
83. Neira, J., Tardós, J.: Data association in stochastic mapping using the joint compatibility test. IEEE Trans. Robot. Autom. **17**(6), 890–897 (2001)
84. Newcombe, R., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A., Kohli, P., Shotton, J., Hodges, S., Fitzgibbon, A.: Kinectfusion: real-time dense surface mapping and tracking. In: 2011 10th IEEE International Symposium on Mixed and Augmented Reality (ISMAR), pp. 127–136 (2011)
85. Newcombe, R.A., Davison, A.J.: Live dense reconstruction with a single moving camera. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 1498–1505 (2010)
86. Nister, D.: An efficient solution to the five-point relative pose problem. In: Proceedings of 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 2, Part II, p. 195
87. Nister, D., Stewenius, H.: Scalable recognition with a vocabulary tree. In: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 2, pp. 2161–2168 (2006)
88. Nistér, D., Naroditsky, O., Bergen, J.: Visual odometry. In: Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2004), vol. 1, pp. I–652 (2004)
89. Nützi, G., Weiss, S., Scaramuzza, D., Siegwart, R.: Fusion of IMU and vision for absolute scale estimation in monocular slam. J. Intell. Robot. Syst. **61**(1), 287–299 (2011)
90. Olson, C., Matthies, L., Schoppers, M., Maimone, M.: Rover navigation using stereo ego-motion. Robot. Auton. Syst. **43**(4), 215–229 (2003)
91. Olson, E., Leonard, J., Teller, S.: Fast iterative alignment of pose graphs with poor initial estimates. In: Proceedings of IEEE International Conference on Robotics and Automation (ICRA 2006), pp. 2262–2269 (2006)
92. Panzier, S., Pascucci, F., Setola, R., Ulivi, G.: A low cost vision based localization system for mobile robots. Target **4**, 5 (2001)
93. Panzieri, S., Pascucci, F., Santinelli, I., Ulivi, G.: Merging topological data into Kalman based slam. In: World Automation Congress, 2004, Proceedings, vol. 15, pp. 57–62 (2004)
94. Rigatos, G.: Modelling and Control for Intelligent Industrial Systems: Adaptive Algorithms in Robotics and Industrial Engineering, vol. 7. Springer, New York (2011)
95. Rigatos, G., Siano, P., Raffo, G.: Distributed control of unmanned surface vessels using the derivative-free nonlinear Kalman filter. Intell. Ind. Syst. **1**, 99–126 (2015)
96. Rosten, E., Drummond, T.: Machine learning for high-speed corner detection. In: ECCV on Computer Vision 2006, pp. 430–443 (2006)
97. Rublee, E., Rabaud, V., Konolige, K., Bradski, G.: ORB: an efficient alternative to sift or surf. In: 2011 IEEE International Conference on Computer Vision (ICCV), pp. 2564–2571 (2011)
98. Rusu, R.B.: Semantic 3D object maps for everyday manipulation in human living environments. KI-Künstliche Intelligenz **24**(4), 345–348 (2010)
99. Rusu, R.B., Blodow, N., Beetz, M.: Fast point feature histograms (FPFH) for 3D registration. In: IEEE International Conference on Robotics and Automation, 2009 (ICRA'09), pp. 3212–3217 (2009)
100. Scaramuzza, D.: Ocamcalib toolbox: Omnidirectional camera calibration toolbox for Matlab. https://sites.google.com/site/scarabotix/ocamcalib-toolbox
101. Scaramuzza, D., Fraundorfer, F.: Visual odometry: part I—the first 30 years and fundamentals. IEEE Robot. Autom. Mag. **18**(4), 80–92 (2011)
102. Scaramuzza, D., Siegwart, R.: Appearance-guided monocular omnidirectional visual odometry for outdoor ground vehicles.

IEEE Trans. Robot. **24**(5), 1015–1026 (2008). doi:10.1109/TRO.2008.2004490
103. Shi, J., Tomasi, C.: Good features to track. In: Proceedings of 1994 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1994 (CVPR'94), pp. 593–600 (1994)
104. Siegwart, R., Nourbakhsh, I., Scaramuzza, D., Siegwart, R., Nourbakhsh, I., Scaramuzza, D.: Introduction to Autonomous Mobile Robots, 2nd edn. MIT Press, Cambridge (2011)
105. Simon, D.: Optimal State Estimation: Kalman, H-infinity, and Nonlinear Approaches. Wiley, New York (2006)
106. Smith, R., Self, M., Cheeseman, P.: Estimating uncertain spatial relationships in robotics. Auton. Robot Veh. **1**, 167–193 (1990)
107. Smith, S., Brady, J.: Susana new approach to low level image processing. Int. J. Comput. Vis. **23**(1), 45–78 (1997)
108. Talukder, A., Goldberg, S., Matthies, L., Ansar, A.: Real-time detection of moving objects in a dynamic scene from moving robotic vehicles. In: 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2003 (IROS 2003), Proceedings, vol. 2, pp. 1308–1313 (2003). doi:10.1109/IROS.2003.1248826
109. Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics. MIT Press, Cambridge (2005)
110. Thrun, S., Gutmann, J., Fox, D., Burgard, W., Kuipers, B., et al.: Integrating topological and metric maps for mobile robot navigation: A statistical approach. In: Proceedings of the National Conference on Artificial Intelligence, pp. 989–996. Wiley, New York (1998)
111. Thrun, S., Montemerlo, M.: The graph slam algorithm with applications to large-scale mapping of urban structures. Int. J. Robot. Res. **25**(5–6), 403–429 (2006)
112. Tombari, F., Salti, S., Di Stefano, L.: Unique shape context for 3D data description. In: Proceedings of the ACM Workshop on 3D Object Retrieval, pp. 57–62. ACM, New York (2010)
113. Whelan, T., Kaess, M., Fallon, M., Johannsson, H., Leonard, J., McDonald, J.: Kintinuous: Spatially extended kinectfusion. In: RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras (2012)
114. Won, D.H., Chun, S., Sung, S., Kang, T., Lee, Y.J.: Improving mobile robot navigation performance using vision based slam and distributed filters. In: International Conference on Control, Automation and Systems, 2008 (ICCAS 2008), pp. 186–191 (2008)
115. Xu, L., Jia, J., Matsushita, Y.: Motion detail preserving optical flow estimation. IEEE Trans. Pattern Anal. Mach. Intell. **34**(9), 1744–1757 (2012)
116. Yousif, K., Bab-Hadiashar, A., Hoseinnezhad, R.: Real-time RGB-D registration and mapping in texture-less environments using ranked order statistics. In: 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2014), pp. 2654–2660 (2014)
117. Yousif, K., Bab-Hadiashar, A., Hoseinnezhad, R.: 3D slam in texture-less environments using rank order statistics. In: Robotica (Accepted, September 2015)
118. Zhang, T., Liu, X., Kühnlenz, K., Buss, M.: Visual odometry for the autonomous city explorer. In: Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'09), pp. 3513–3518. IEEE Press, Piscataway (2009). http://dl.acm.org/citation.cfm?id=1733023.1733309
119. Zhang, Z.: Determining the epipolar geometry and its uncertainty: a review. Int. J. Comput. Vis. **27**(2), 161–195 (1998)