A Comprehensive Review on Autonomous Navigation

SAEID NAHAVANDI, ROOHALLAH ALIZADEHSANI, DARIUS NAHAVANDI, SHADY MOHAMED, NAVID MOHAJER, MOHAMMAD ROKONUZZAMAN, and IBRAHIM HOSSAIN, Institute for Intelligent Systems Research and Innovation (IISRI), Deakin University, Australia

The field of autonomous mobile robots has undergone dramatic advancements over the past decades. Despite achieving important milestones, several challenges are yet to be addressed. Aggregating the achievements of the robotic community as survey papers is vital to keep the track of current state-of-the-art and the challenges that must be tackled in the future. This paper tries to provide a comprehensive review of autonomous mobile robots covering topics such as sensor types, mobile robot platforms, simulation tools, path planning and following, sensor fusion methods, obstacle avoidance, and SLAM. The urge to present a survey paper is twofold. First, autonomous navigation field evolves fast so writing survey papers regularly is crucial to keep the research community well-aware of the current status of this field. Second, deep learning methods have revolutionized many fields including autonomous navigation. Therefore, it is necessary to give an appropriate treatment of the role of deep learning in autonomous navigation as well which is covered in this paper. Future works and research gaps will also be discussed.

1 INTRODUCTION

Robotics have impacted our lives in many ways. Backed up by technological advancements, robots have found their way in several application domains such as medical [44, 83, 177, 264], military [122, 233, 300], industrial [21, 39, 135, 174, 305], space [42, 193, 197, 222, 291], agricultural [89, 99, 181, 195, 207, 311], etc. Adding autonomous navigation ability to robot platforms boosts their performance significantly since they can reach wherever they are needed on their own. This motivation has driven researchers to push the autonomous navigation technology to its limits. Given the rich and fast-evolving literature on the topic of autonomous navigation, it is necessary to prepare literature surveys on regular basis. This way, experienced researchers, as well as newcomers, can get an insight into the current state-of-the-art in autonomous navigation which is the inspiration for this survey.

The journey toward autonomous mobile robots has been started with development of first-ever general-purpose mobile robot named Shakey [225]. Developed by Artificial Intelligence Center of Stanford Research Institute, Shakey had the ability to reason about its own actions. Another notable mobile robot is CART [31, 88] which was capable of line

Authors' address: Saeid Nahavandi, saeid.nahavandi@deakin.edu.au; Roohallah Alizadehsani, r.alizadehsani@deakin.edu.au; Darius Nahavandi, darius.nahavandi@deakin.edu.au; Shady Mohamed, shady.mohamed@deakin.edu.au; Navid Mohajer, n.mohajer@deakin.edu.au; Mohammad Rokonuzzaman, m.pappu@deakin.edu.au; Ibrahim Hossain, i.hossain@deakin.edu.au, Institute for Intelligent Systems Research and Innovation (IISRI), Deakin University, 75 Pigdons road, Geelong, Victoria, Australia, 3216.

following using a camera and a radio control link. The required processing was done on a mainframe computer which was connected to CART via the aforementioned radio link. There have also been attempts at the development of mobile robots for space missions. For example, Lunokhod [23, 148] was developed by Soviet Union for moon exploration or the United States developed Viking Mars landers (VL-1, and VL-2) for measurement of atmospheric temperature, wind speed and direction, and pressure [130] as well as Mars rover Sojourner [202].

Autonomous navigation has made constant progress over the years. Autonomous mobile robots were developed to drive [25, 84, 150, 227, 237], operate as medical assistance in hospitals [226], explore active volcanoes (Dante I and II) [45, 169], etc. The progress has even been boosted by holding international competitions focused on autonomous mobile robots. The DARPA grand challenge held in 2004 [18] was a turning point in the field of autonomous mobile robots. In this competition, autonomous cars were tasked to navigate a 240km off-road path without human intervention. Although none of the contenders managed to reach the finish line, valuable lessons were learned from the experienced failures. The challenge was repeated a year later and this time, five of the autonomous vehicles reached the finish line; among them, Stanley [282] from Stanford University claimed first place. In 2006, yet another grand challenge was held but this time in urban environments [19]. The objective was to navigate through an urban area while complying with traffic regulations and avoiding collisions with obstacles. The winner of this contest was vehicle Boss of team Tartan [289] developed in a collaborative effort by Carnegie Mellon University and General Motors.

Researchers have also worked on multi-robot setups. Swarm-bots project [117] was focused on self-organizing swarm of simple robots to achieve complex tasks. Swarm of autonomous underwater vehicles for performing stealthy underwater missions has also been investigated yielding a method called Suave (Swarm Underwater Autonomous Vehicle localization) [184]. This method reduces the frequency of going to water surface for obtaining position information from external sources such as satellites. This way, the probability of being compromised for the stealthy underwater mission is reduced. In 2010, there has also been an international contest called MAGIC (Multi Autonomous Ground-Robotic International Challenge) [36] devoted to group of autonomous robots working together.

Obviously, multiple survey papers have already been published on the subject of autonomous mobile robots. To highlight the differences between our survey and existing ones, table 1 has been prepared in which existing surveys have been sorted according to their publication year. As can be seen in table 1, some of the topics are missing in previous surveys but are covered in this paper. For example, Niloy et al. [224] have only focused on indoor autonomous navigation. The survey by Pol and Murugan [236] is also limited to indoor navigation except that human presence in the environment has been considered.

Mohanty and Parhi [205] have primarily focused on requirements of global and local path planning and navigation. In Guzel's survey [120], the primary topic is mapless, vision-based navigation but some map-based methods have been reviewed as well. However, SLAM and several other important topics are absent. Injarapu and Gawre [137] have only considered path planning and legacy methods for obstacle avoidance. Pandey et al. [230] have divided mobile robot navigation algorithms into three groups deterministic, non-deterministic, and evolutionary algorithms. Similar to some other surveys, Pandey et al. divide navigation into global and local [223]. Again, some topics such as SLAM and modern obstacle avoidance methods are missing from [230]. Victerpaul et al. [290] have special emphasis on path planning approaches and have also covered related topics such as navigation and legacy obstacle avoidance. Tzafestas [286] has covered most of the topics but lacks the review of modern obstacle avoidance methods, simulation tools, etc. Alatise and Hancke [32] have covered multiple topics such as sensor types, autonomous mobile robots challenges, classical obstacle avoidance methods, sensor fusion, etc.; however, other important topics such as SLAM and newer obstacle avoidance methods based on reinforcement learning (RL) or deep learning (DL) have not been covered.

Table 1. Comparison between our survey and previous ones.

		2013	2013	2015	2017	2017	2017	2018	2020	2021	2021	2021	2021	-
Year		[205]	[120]	[236]	[137]	[230]	[290]	[286]	[32]	[250]	[232]	[224]	[307]	ours
Sensor types									✓			✓	✓	✓
Mobile robot platform types				✓					✓			✓		✓
Simulation tools														✓
Path planning			✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓
Path following						✓	✓	✓				✓	✓	✓
Sensor fusion								✓	✓		✓	✓	✓	✓
Obstacle avoidance	Legacy	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	RL/DL					✓					✓			✓
Navigation	Map-based	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓
	Mapless	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓
SLAM								✓			✓	✓	✓	✓

Ibanez et al. [250] have reviewed path planning methods with special focus on path planners for ground vehicles which can be extended to other platforms such as autonomous boats. However, SLAM, modern obstacle avoidance, sensor fusion, etc. are missing from [250]. Panigrahi and Bisoy [232] have not investigated path following/planning, simulation tools, etc. Zghair and Al-Araji [307] have investigated the evolution of control systems of mobile robots for the past decade. Again, modern obstacle avoidance methods and simulation tools are missing from their survey.

Based on the shortcomings of the previous surveys mentioned above, this survey has been prepared to cover all the topics listed in table 1. We have striven to:

- give a comprehensive treatment on legacy as well as modern obstacle avoidance methods (based on RL and DL)
- review the most notable SLAM methods
- introduce well-known robotic simulators and whether they can be linked to robotic operating system (ROS) or not [1]
- types of mobile robot platforms and their characteristics based on their operational environment
- Simple and concise review of famous sensor fusion approaches such as Kalman filter [144] and its extensions as well as particle filter [114]
- review open source SLAM datasets

The rest of this survey is structured as follows. Section 2 is devoted to different types of sensors used for autonomous navigation. Section 3 reviews different types of mobile robot platforms and their characteristics. Common software for robotic simulations is reviewed in section 4. Path planning and path following approaches are reviewed in sections 5 and 6, respectively. Sensor fusion methods are reviewed in section 7. Obstacle avoidance is investigated in section 8. Navigation and SLAM methods are reviewed in sections 9 and 10, respectively. The survey is closed with future works and conclusion in sections 11 and 12, respectively.

2 SENSORS

Acting autonomously is not possible without the ability to sense the surrounding environment [249]. Any autonomous robot must be equipped with a perception module which is combination of hardware (sensors) and software components. This section is devoted to various types of sensors that are commonly used in autonomous mobile robots.
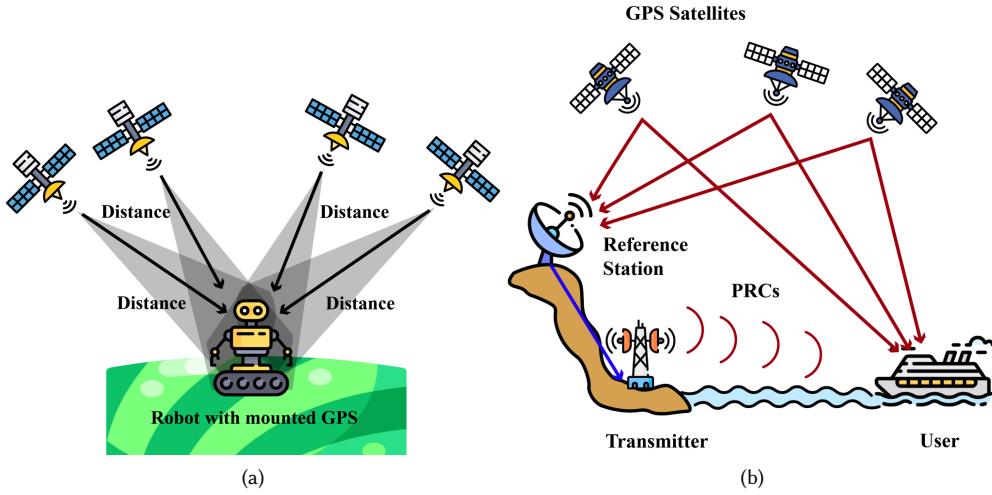
Fig. 1. Illustration of (a) robot localization using GPS receiver and (b) DGPS localization and error correction using reference station.

2.1 Absolution positioning sensors

Mobile robots need to move in their operational environment in order to carry out their objectives at different positions in the environment. For example, a rescue robot needs to navigate toward the wounded victim in order to perform the rescue operation. An important prerequisite to autonomous navigation is the self-localization ability of the robot. Usually, localization is performed with respect to some reference frame in the environment.

Global positioning system (GPS) receiver is one of the common sensors that provides absolute positioning in terms of longitude and latitude as well as time data anywhere on or near the Earth based on information received from low Earth orbiting satellites. The receiver uses the difference between the reception time of satellite signal and its broadcast time to compute its distance from the satellite. As shown in figure 1a, having access to information from at least four satellites, the receiver can provide positioning information accurate to a few meters for outdoor navigation. The drawback of GPS is that localization data becomes unreliable if unobstructed line of sight to at least four satellites is unavailable. In urban areas, signal outages may be caused by tall buildings or mountains [2]. Additionally GPS signals are not available inside indoor environments.

The localization error of GPS can be reduced by using differential GPS (DGPS). Basically, DGPS relies on multiple fixed beacons with known locations on the Earth. These beacons compare their known positions with the ones computed based on received satellite signals and broadcast correction information as shown in figure 1b. GPS receivers can reduce their positioning error using corrective signals broadcast by these beacons. The drawback of DGPS is that the quality of correction degrades as we move away from base/reference station [46].

2.2 Obstacle detection sensors

In this section, different types of sensors that are commonly used for obstacle detection and avoidance are introduced.
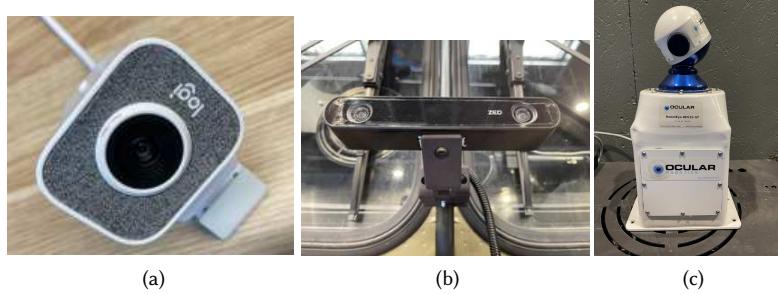
Fig. 2. Different types of cameras: (a) mono, (b) stereo/RGBD, and (c) Stabilized vision system

2.2.1 Visual sensors. Claiming that a robotic system has autonomous navigation capability is almost impossible without equipping the robot with means of obstacle detection and avoidance. One can think of variety of sensor types when it comes to obstacle detection. Monocular (figure 2a) and stereo/RGBD cameras (figure 2b) provide feature-rich visual data at a reasonable price. The emergence of powerful vision algorithms based on DL and relatively affordable edge devices with reasonable processing power such as Jetson boards [3] has made visual sensors an ideal choice for performing various vision tasks including obstacle detection. During the robot movement, the onboard cameras may be subject to vibration making captured images blurry. This issue is addressed by stabilized vision systems [26] an example of which is shown in figure 2c. Stabilized vision systems also improve the tracking capability.

2.2.2 Range sensors. One of the most convenient solutions for obstacle detection is using range sensors which measure time-of-flight. Radar (Radio Detection And Ranging) relies on radio waves to detect distance and angle to objects and/or their velocity. A typical Radar has been depicted in figure 3. LiDAR (light detection and ranging) does similar thing to Radar but instead of using radio waves, it uses pulsed laser light. LiDAR sensors are capable of measuring range data with higher accuracy compared to Radar. The downside of these sensors is their high price. Moreover, laser scanners cannot sense glass obstacles and they cannot be used underwater due to laser disruption by water [245]. Typical LiDARs have been shown in figure 3.

Ultrasonic sensors also provide range data but with much less precision compared to LiDAR and RADAR. The total range of ultrasonic sensor is lower (about seven meters) than LiDAR and RADAR. The lower precision of ultrasonic sensor is due to the fact that its radiation pattern is bat-like. Therefore, after receiving the reflection of an emitted signal that hits an obstacle, the position of the sensed obstacle can only be estimated vaguely within the region covered by the emitted wave. This is in contrast to LiDAR and RADAR that provide much more accurate data about the obstacle position. A typical ultrasonic sensor has been shown in figure 3.

2.3 Relative positioning sensors

In section 2.1, absolute positioning sensors were reviewed. However, there are scenarios that absolution positioning is not possible which is the motivation for using relative positioning sensors such as inertial measurement unit (IMU) and encoder. IMU is composed of three accelerometers and three gyroscopes along $\{x, y, z\}$ axes. When IMU is mounted on the robot, the accelerometers and gyroscopes measure the robot acceleration and rotation rate along the three axes, respectively. Using the well-established mathematics of an inertial navigation system (INS) [283], it is possible to turn IMU readings into useful 3D position and 3D orientation for the robot. Due to successive integration of IMU data, the
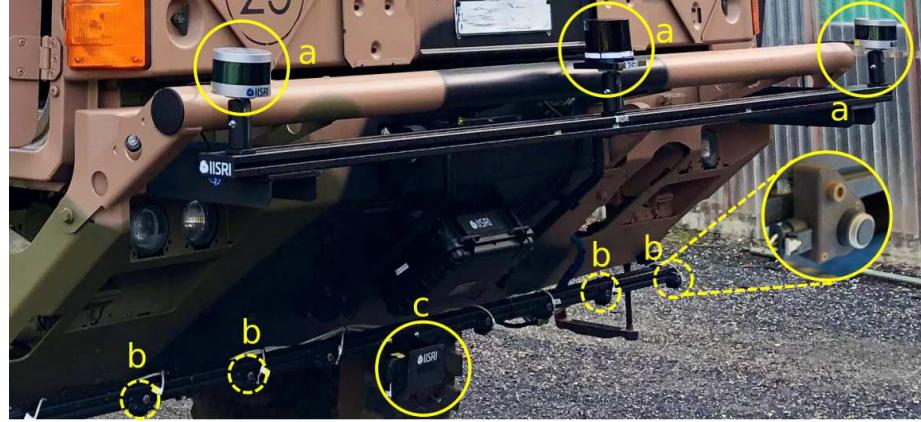
Fig. 3. Different types of range sensors: (a) LiDAR, (b) Ultrasonic, (c) RADAR.

estimated robot pose is subject to accumulated error which can be reduced by fusing INS data with other sensors such as GPS and camera(s) (see section 7.4).

Another sensor that can be used for relative positioning in ground robots is rotary encoder. Mounting encoder on a robot wheel, it is possible to count the number of rounds the wheel has completed. Knowing the perimeter of the wheel, the amount of robot displacement can be estimated. The drawback of using encoders is that wheel slippage leads to erroneous displacement estimates.

3 MOBILE ROBOT PLATFORMS

Over the years, different types of mobile robot platforms have been designed and developed. Mobile robots may be ground [41, 55, 253, 288], aerial [24, 129, 200] or underwater [68] vehicles.

3.1 Ground vehicles

Ground vehicles come in variety of sizes and shapes depending on the objectives they have been designed for. Apart from structural differences, suite of sensors mounted on these robots depends on their operational environment. In indoor environments, GPS is not accessible but the environment is usually feature-rich due to surrounding walls, doors, furniture, etc. Thus, range sensors can be used to carry out scan matching [165] in order to achieve accurate localization. Using stereo cameras is also an affordable solution to achieve depth data. Ring of ultrasonic sensors around the robot chassis is also a viable solution for obstacle detection [156] or localization [211, 212] in indoor environments. An illustration of indoor environment has been shown in figure 4. Regardless of being indoor or outdoor, INS can be used for localization which relies on IMU data. Indoor environments usually have smooth floors so wheel slippage is not likely and rotary encoders are quite suitable for estimating the robot displacement.

For outdoor environments, GPS can aid the absolute positioning provided that clear line-of-sight between the GPS receiver and at least four satellites is available. Such conditions may be violated in urban environments (figure 5a) due to signal blockage by tall buildings. However, urban environments are well-structured in nature. This property can be exploited to improve localization accuracy by performing scan matching on LiDAR data [302]. Even if GPS signal is available, scan matching can still lead to better localization accuracy [175]. An alternate approach is visual place
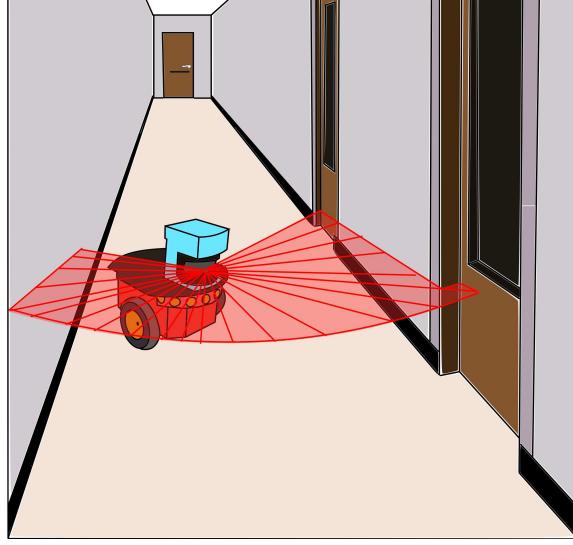
Fig. 4. Indoor environment illustration.

recognition (VPR) [63, 194, 284] which plays a vital role in reducing localization error by recognizing the places that have been visited previously.

In some outdoor applications (e.g. military), the mobile robot encounters offroad environments which are harsh, unstructured, and sometimes impossible to traverse. Offroad environments push the mechanical [30] and autonomous capabilities [141] of mobile robots to their limits. As can be seen in figure 5b, the offroad robots must be equipped with large wheels, long travel suspension, and preferably all-wheel drive ability to be able to traverse different types of lands such as muddy, rocky, snowy, etc. Apart from robot mechanical requirements for offroad environments, several challenges must be addressed by autonomous software modules of the robot to achieve successful navigation:

(1) Given that offroad environments do not follow any specific structure [277], autonomous navigation cannot make any assumptions about them beforehand.
(2) Some offroad environments like deserts are featureless which hinders localization of the robot due to lack of recognizable landmarks. For example in deserts, even the terrain shape may vary due to movement of sands by blowing winds.
(3) The offroad environments are hazardous due to deep valleys, rough rivers, volcanoes, etc. Therefore, it is crucial to take into account impassable areas during path planning to enforce the robot safety.
(4) In offroad environments, some of the obstacles may be passable like bushes while others are impassable like mountains. The ability to detect passable obstacles is vital to shorten the length of the planned path by moving through the passable obstacles.
(5) Autonomous navigation relies on the robot sensor readings to make decisions. However, the field of view of sensors may be limited due to dense vegetation which hinders efficient localization.
(6) Depending on the offroad environment, the robot motion controller parameters must be tuned differently [54]. For example, the amount of force applied to the robot wheels in snowy environments is different from dirt lands.
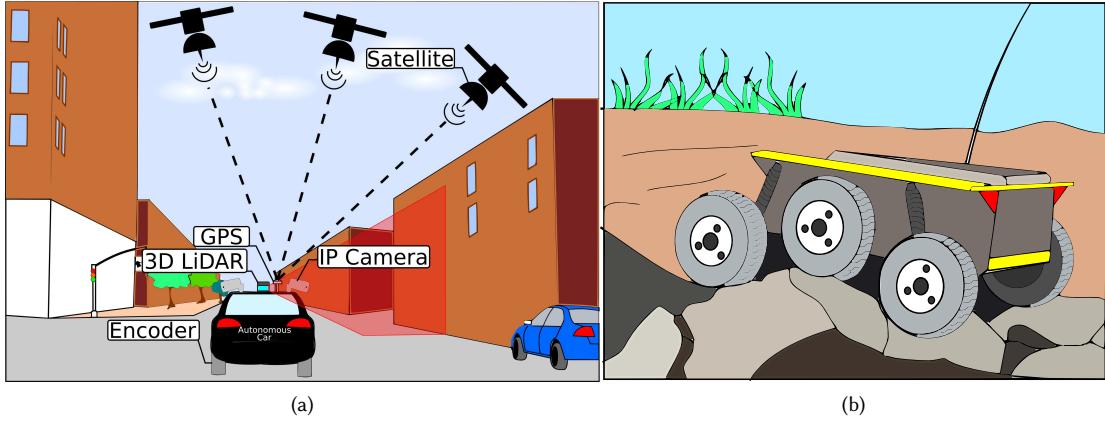
Fig. 5. Ground robots for outdoor scenarios: (a) structured urban environment, (b) unstructured offroad environment.

3.2 Unmanned aerial vehicles

While controlling ground vehicles are simpler than unmanned aerial vehicles (UAVs), their movement is limited to traversable ground. Aerial vehicles are not bound to such constraints and they can reach the desired location in less time compared to ground vehicles. However, aerial vehicles have some limitations as well. For example, the flight duration may be short, especially for low-grade UAVs. Additionally, failure in control module may lead to a disastrous crash damaging the UAV. Another drawback is limited payload. UAVs can carry limited load which means the set of hardware (e.g. sensors and processing units) mountable on them must be chosen wisely. Given that IMUs are usually small and lightweight, using them to implement INS for efficient localization of UAVs is quite popular. As shown in figure 6a, UAVs may be deployed to variety of outdoor environments e.g. urban or unstructured and offroad. To keep the INS error bounded, GPS sensor can be mounted on UAVs [28, 75, 221, 308]. Using UAVs in indoor environments is also possible. The only difference to outdoor scenarios is unavailability of GPS signals. Instead of GPS, INS data can be fused with LiDAR [180], mix of laser data and vision sensors [146], Ultra Wideband (UWB) [266], mix of UWB and 3D LiDAR [178]. Even seamless switch between indoor/outdoor navigation modes for UAVs has been attempted [60]. Moreover, Hall effect sensor data has been used to aid the velocity update of INS [306]. Hall effect sensor detects magnetic filed presence and magnitude by exploiting Hall effect. According to this effect, a magnetic field perpendicular to current in an electrical conductor produces voltage difference across the electrical conductor such that it is transverse to the electric current and the magnetic field.

3.3 Underwater vehicles

AUVs can be used in various applications such as retrieving black boxes of airplanes [139, 244] crashed in oceans/seas, enforcing security of ports and harbors by detection and disposal of explosives and mines [29, 241], and infrastructure maintenance for oil and gas industries [57, 138, 297, 309]. However, underwater autonomous navigation (figure 6b) is challenging due to unavailability of GPS [112] and limited visibility. Failing to register the current position of the robot using visual features leads to inaccurate localization which in turn hinders autonomous navigation [239]. Using DR and INS to localize AUVs has a long history [172]. However, these methods suffer from accumulated error due to

noisy sensor data, ocean currents, and Earth's gravity in underwater environments [112]. To deal with the accumulated error, Geophysical Navigation (GN) [243] can be used in which AUV sensor readings are matched with geophysical map of the underwater environment. Unfortunately, GN suffers from some drawbacks such as requiring geophysical map of the environment before the navigation starts and computational complexity of matching sensor data with the map. As an alternative approach to GN, acoustic ranging systems [95] may be used but they rely on complex infrastructure and their cost of deployment is high. Researchers have explored new alternatives for AUV localization such as optical technologies [53]. Yet again, development of these technologies has been slowed down due to tough conditions of underwater environments. In the presence of proper lighting conditions, visual processing systems can improve localization accuracy significantly [112]. For example, Mehdi et al. [199] have fused IMU and depth data extracted from stereo camera system to plot the map of the underwater environment using SLAM implemented in ROS. The map is used for path planning and autonomous navigation.

3.3.1   AUV suitable sensors. The first sensor to perform localization using INS underwater is IMU. To deal with INS accumulated error, fusing IMU data with other sensors is crucial. Doppler velocity loggers (DVL)[20] is a sonar system for measuring motion underwater that can determine the speed and the direction of AUV movement. Regarding absolute positioning, GPS cannot be used underwater since its electromagnetic signals decay quickly. Acoustic signaling is an alternative solution since acoustic signals decay very slowly underwater [126]. The downside of this method is its reliance on baseline stations deployed in the navigation environment. The baseline batteries need frequent charging in order to stay functional and respond to interrogator device mounted on AUV. In addition, baseline deployment and recovery is time-consuming and costly which makes acoustic navigation impractical in large-scale environments [158]. To address this problem, MIT researchers have developed underwater backscatter localization (UBL) [110] which is a battery-free pinpointing system. Instead of emitting acoustic signals which require battery consumption, UBL reflects the modulated signals from its surrounding environment. This way, position information is provided without needing any battery-operated device. UBL takes the role of underwater GPS.

Family of sonar sensors such as forward looking sonars and side scan sonar [151] is widely used for obstacle detection and localization underwater. Off-the-shelf underwater camera systems [196] are also very common for extracting depth data based on stereo vision.

4   SIMULATION TOOLS

To conduct research in the field of robotics, a suitable robotic platform is needed which is a combination of software and hardware components. During research process, various experiments are needed to be carried out. It is quite likely for these experiments to fail due to software and/or hardware module malfunction. Sometimes, these failures may damage the robot. On the other hand, performing experiments on real robots is tedious and time consuming. An alternative approach is to implement and test the robot software in simulation. This way, researchers will not have to worry about accidentally damaging robots and no hardware is needed to test an initial idea. Resetting the simulation scenario is also much easier than resetting the robot status in real world. In table 2, the characteristics of some of main robotic simulators have been provided [4].

While testing new ideas in simulation is beneficial, it is not the ultimate goal of robotic projects. Sooner or later, the simulated project must be evaluated on real robots. The drawback of using simulators is that for modeling the physical laws of the real world, some compromises may have been made. Therefore, the project tested in simulation may need some refinements and manual tuning to be ready for deployment on a real robot. Nevertheless, putting the
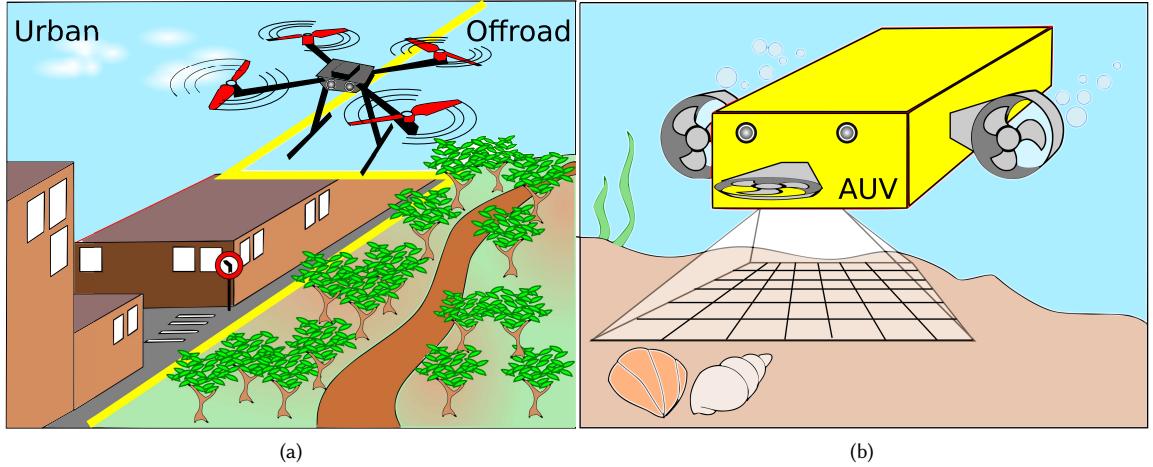
Fig. 6. Outdoor (a) aerial and (b) under-water environments and their respective robots.

time and effort into developing robotic projects in simulation before deployment on real robots is definitely worth it since it reduces development time and cost significantly. Among the existing simulators, some like Gazebo rely on high quality physics engines such as open dynamic engine (ODE) [5] to reduce the gap between simulated and real robots drastically. That is why many well known robotic companies provide their customers with simulation tools of their commercial robots in the form of software packages relying on ROS and Gazebo. ROS is a set of tools and software libraries specifically designed to facilitate robot application development. Using ROS, development time of robotic projects is drastically reduced due to the possibility to use high quality of-the-shelf software packages that implement famous algorithms and interface with various sensors (hardware abstraction). Some examples of well-known robots with descent simulation in ROS and Gazebo include but are not limited to Fetch robotics [6], PR2 robot [7], KUKA iiwa [8], and humanoid Nao robot [9].

5   PATH PLANNING

After percepting the environment using robot onboard sensors, it is necessary to plan a feasible path in order to reach the desired target starting from the current position. The choice of the path planning algorithm depends on whether the environment map is available or not. In case the map is known, the choice of the path planner depends on the map representation. For occupancy grid maps, D* [267] and its extensions such as focused D* [268] and D* Lite [163] are suitable choices. D* and its extensions are all considered as incremental search algorithms. D* assumes that unseen parts of the environment are free of obstacles. Based on this assumption, D* uses the map to compute the shortest path from the start to the goal. In the event of observing new obstacles, the map is updated with their information and a new shortest path is computed if necessary. This trend repeats until the goal is reached or it is concluded that the goal is indeed unreachable. Considering that observing new obstacles during navigation is quite likely, recomputing the shortest path must be fast which is what D* and its extensions are famous for. It is worth noting that focused D* is a combination of A* [125] and the ideas from the original D* algorithm; and D* Lite has been built based on life long

Table 2. Characteristics comparison of main robotic simulators

Simulator	Supported OS	Physics engine	ROS support	Supported			
				Robots	Actuators	Tools	Sensors
Gazebo[10]	Linux	ODE, Bullet, Simbody, DART	ROS1, ROS2	Mobile, humanoid, industrial	Revolute, prismatic, screw and spherical joints	Grippers	Camera, distance & proximity sensors, laser, force sensors
RoboDK	Linux, Mac OS, Windows, Android	None	-	Industrial robots	Possibility to create rotated and linear axis	Grippers, weld tools, spindle, grinding, polishing, tool changer, ...	Lasers, cameras, laser tracker
Webots[11]	Linux, Mac OS, Windows	Proprietary, ODE-based	ROS1, ROS2	Mobile, humanoid, industrial	Brake, connector, display, emitter, linear/rotational motor, muscle, pen, propeller, speaker, track (conveyor belt or tank robots)	Grippers	Accelerometer, camera, compass, distance sensor, GPS, gyroscope, lidar, position sensor, receiver, touch sensor
CoppeliaSim [12]	Linux, Mac OS, Windows	ODE, Bullet, Vortex, Newton	ROS1, ROS2	Mobile, humanoid, industrial	Revolute, prismatic & spherical motors	Pens, paint gun, welding torch, grippers	Vision, force sensors, proximity sensor, accelerometer, gyroscope, lasers, LiDARs
OpenRave	Linux, Windows	ODE, Bullet	ROS1	Mobile, humanoid, industrial	Revolute & linear joints	Grippers	Cameras, ray-casting laser
Unity	Linux, Mac OS, Windows	NVidia PhysX, Box2D, other open-source physics engines	ROS1, ROS2	Mobile, humanoid, industrial	Revolute & prismatic joints	Grippers	Cameras

planning A* (LPA*)[164] and another incremental search method made of combining A* ideas and dynamic SWSF-FP [240].

5.1 DMP-based path planning

One of the popular approaches for trajectory planning and control signals generation in manipulator robots is dynamic movement primitive (DMP) [136, 252]. DMP is a highly adaptable mathematical model capable of representing complex trajectories without the need for manual tuning of its parameters. Roughly speaking, a DMP is a second order nonlinear dynamical system capable of imitating complex trajectories provided that its parameters are set appropriately. DMPs are either discrete or rhythmic which are suitable for generation of aperiodic and periodic trajectories, respectively. Discrete DMPs are suitable for path following since planned paths are not periodic trajectories. The crux of a discrete DMP is a point-attractor system:

$$\ddot{y} = \alpha_y \left[\beta_y (g - y) - \dot{y} \right], \tag{1}$$

where y, $\dot{y}$, and $\ddot{y}$ are the system position, velocity, and acceleration, respectively. Moreover, g is the goal point and $\alpha_y > 0$ and $\beta_y > 0$ are gain coefficients. Careful inspection of equation 1 reveals that it is just a PD controller whose objective is to reach g given the current state y. To specify the shape of the trajectory between y and g, equation 1 is augmented with a special function $f(x)$ known as the forcing function[252]:

$$f(x) = \frac{\sum_{i=1}^{N} \psi_i w_i}{\sum_{j=1}^{N} \psi_j}(g - y_0)x, \tag{2}$$

where ψ_i is the ith Gaussian basis function, w_i is the ith weight parameter, y_0 is the initial position of the system and x is state variable of another dynamical system known as the canonical system. By setting the parameter set $\{w_i, i = 1, ..., N\}$ of the forcing function, any arbitrary trajectory can be produced using DMP. The canonical system

$$\zeta \dot{x} = -\alpha_x x, \tag{3}$$

controls the life time of the DMP. In equation 3, α_x is a constant which is set to $-\log(0.01)$ for ensuring 99% convergence of the DMP after its life time ζ is passed.

Despite the fact that DMP is primarily designed for motion planning of robotic manipulators, it has also been used for path planning in mobile robots. For example, Jiang et al. [140] have carried out mobile robot path planning using DMPs. A desired path is first planned and trajectories are extracted from it to be used as training samples for adjustment of DMP parameters. Thanks to the DMP high adaptability, Jiang et al. were able to achieve smooth trajectories for the mobile robot even when start and goal positions were altered.

6 PATH FOLLOWING

To execute planned paths, mobile robots must be equipped with the necessary means to follow them. Over the years, various path following methods have been developed. Some of these methods are as simple as pure pursuit while others take a more involved approach by relying on dynamical systems or deep learning. Some of these path following methods are reviewed in this section.

6.1 Pure pursuit algorithm

Pure pursuit [72] is a path tracking algorithm with respectable history as it has been used in various projects such as the Terragator [62], NavLab[278], and NavLabs2[73]. In a nutshell, pure pursuit algorithm computes the robot angular velocity while assuming constant linear velocity. The robot is moved from its current position to some look-ahead point in front of it. After reaching the specified look-ahead point, it is moved further on the path which is to be followed.

The algorithm computes an arc joining the robot current position to the goal position. The goal point is one look-ahead distance from the current position of the robot. In figure 7, the robot's local coordinate frame as well as the goal point (x_g, y_g) have been shown. The objective is to compute the curvature γ of the arc joining the robot current position to the goal position assuming chord length (look-ahead distance) of 1. The goal point is enforced to be on the robot path. Computation of the curvature value is fairly straightforward. Looking at figure 7, one can write the following equations:

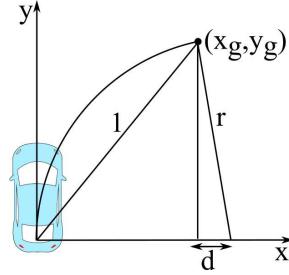$$x^2 + y^2 = 1^2 \tag{4}$$

$$x + d = r \tag{5}$$

Fig. 7. Geometry of the pure pursuit algorithm: 1 is the chord length (look-ahead distance), r is the radius of the arc joining robot position to the goal point, (x_g, y_g) is the goal point

where equation 4 follows from Pythagorean theorem in the left bigger triangle and equation 5 is due to the fact that sum of x and d is equal to the radius of curvature r. Using equations 4 and 5, the curvature γ can be computed as:

$$d = r - x$$

$$(r - x)^2 + y^2 = r^2 \implies r^2 - 2rx + x^2 + y^2 = r^2 \implies 2rx = l^2 \implies r = \frac{l^2}{2x} \implies \gamma = \frac{2x}{l^2} \tag{6}$$

6.2 DMP-based path following

As mentioned in section 5, DMP is a popular tool that generates smooth trajectories for manipulator robots. However, it has also been used for path following of mobile robots. Sharma el al.[260] have implemented a DMP-based path tracking approach for nonholonomic robots which is capable of avoiding an obstacle on the path. Two DMPs (to represent movement along x and y axes) were used and forcing function of each of the DMPs was parameterized using radial basis functions (RBFNs). Thanks to the adaptive nature of DMP, the proposed approach can preserve the shape of the planned trajectory and reach the goal even if the start or goal position is altered. Moreover, the proposed approach is able to suggest both trajectories passing both sides of the obstacle. This is particularly useful when one of the two paths is not traversable for some reason.

6.3 DL and RL-based path following

Inspired by interesting learning results achieved by RL, it has been extended to handle problems with continuous state and action spaces. This trend has been reinforced by borrowing ideas from DL giving birth to robust deep RL (DRL) methods capable of controlling robots with continuous and high dimensional states and action spaces. For example, Cheng et al. [66] applied DRL for path following and obstacle avoidance. The path following is realized by designing appropriate state and reward functions which are central to successful training of any RL agent. To realize obstacle avoidance, the aforementioned reward and state functions are modified to take into account distance and angle to detect obstacles. The RL algorithm used in this work was DDPG [182]. The proposed approach has been evaluated in simulation.

7 SENSOR FUSION METHODS

Effective localization of mobile robots is a challenging task. There are multiple sources of uncertainty that the localization module has to deal with in order to estimate robot location with reasonable accuracy at each instance in time. The simplest form of localization is dead reckoning (DR) [67]. In this method, equations of motion of the mobile robot

are devised based on robot structural properties such as wheel circumference, onboard sensors e.g. IMU, magnetome-ter, and rotary encoders. Knowing number of wheel rotations (using rotary encoder) and direction of movement, it is possible to estimate the robot position and heading at each instance in time. However, sensor readings are always noisy so estimated positions are prone to errors. The situation gets worse by stochastic phenomena of the surrounding environment. For example, in case of wheel slippage, the rotary encoder reports wheel movement, however, due to slippage the robot may not have move at all leading to incorrect position estimation. To deal with localization error, it is common practice to fuse readings from multiple sensors to get a better position estimation. In the remainder of this section, some of the most famous filtering algorithms that are used in mobile robot localization are reviewed.

7.1 Kalman filter

Kalman filter (KF) is an algorithm for estimation of unknown variables based on a sequence of possibly noisy observa-tions (measurements) collected over time. The estimated values will be more precise compared to the values estimated based on a single observation. KFs do not keep history of past observations. Only the previous state is kept so memory requirements are light. KFs computations are fast enough to be used in real-time applications such as robot localiza-tion. KF models the unknown current state vector x_{k-1} with a Gaussian distribution $\mathcal{N}(\hat{x}_{k-1}, P_{k-1})$ in which $k-1$ is the current time step, $\hat{x}_{k-1}$ and P_{k-1} are the mean and covariance matrix of the distribution. The hat accent of $\hat{x}_{k-1}$ emphasizes that mean is not necessarily equal to the true current state x_{k-1}. The motivation for this representation is that the true state x_{k-1} is not known but some states are more likely than others and the elements of the state vector may be correlated. These two properties are captured by the Gaussian distribution covariance matrix. KF cycles through prediction and correction phases. In the prediction phase, the linear equations governing the robot motion are represented as matrix F_k which can be multiplied with the current state estimate $\hat{x}_{k-1}$ to predict the next state $\hat{x}_k$. However, the effect of external control commands (or forces) such as turning the wheel or stopping must be taken into account as well. In KF terminology, the control vector is denoted as u_k. How u_k affects the next state is represented by matrix B_k. The final equation for predicting next state using KF reads as:

$$\hat{x}_k = F_k \hat{x}_{k-1} + B_k u_k. \tag{7}$$

It is also necessary to update the covariance matrix P_k. This is done using matrix F_k. However, the uncertainty due to external events such as robot wheel slippage or wind force (for aerial robots) must be considered as well. KF models external uncertainty by mapping each point from the current Gaussian $\mathcal{N}(\hat{x}_{k-1}, P_{k-1})$ to multiple new points which will be part of the next Gaussian distribution $\mathcal{N}(\hat{x}_k, P_k)$. The external uncertainty is captured by adding covariance matrix Q_k to the update equation of P_k yielding:

$$P_k = F_k P_{k-1} F_k^T + Q_k, \tag{8}$$

which means the new estimate of P_k is computed from the current estimate P_{k-1} and additional environment uncer-tainty represented by Q_k. The prediction process is graphically illustrated in figure 8a. In the correction phase, the expected observations (i.e. expected sensor readings) are computed by multiplying matrix H_k with the new estimates $\{\hat{x}_k, P_k\}$ (figure 8b). Actual observations are also obtained from the onboard sensors.

Therefore, the overlap of Gaussian distributions $\mathcal{N}(H_k \hat{x}_k, H_k P_k H_k^T)$ and $\mathcal{N}(z_k, R_k)$ can be computed to obtain the corrected observation distribution with mean $H_k \hat{x}_k'$ as shown in figure 8c. After some manipulation, the H_k in front

Fig. 8. Kalman filter phases for 2D state vector: (a) prediction phase, (b) expected observation computation based on predicted state $\hat{x}_k$, (c) correction phase.

of $H_k \hat{x}'_k$ can be dropped to get the final equations for correcting the prediction ($\{\hat{x}_k, P_k\}$):

$$\hat{x}'_k = \hat{x}_k + K_k \left(z_k - H_k \hat{x}_k\right),$$

$$P'_k = P_k - K_k H_k P_k,$$

$$K_k = P_k H_k^T \left(H_k P_k H_k^T + R_k\right)^{-1},$$

where R_k is the covariance matrix describing sensors uncertainty and K_k is known as the Kalman gain.

7.2 Extended Kalman filter

KF is a powerful state estimator even in the presence of noisy sensor readings. However, to be able to use KF, the state propagation equations must be linear. This requirement rarely holds in real-world applications. Even in the case of a simple differential drive robot [69], the kinematics equations are nonlinear. That is where the extended Kalman filter (EKF) [295] comes in. The main idea behind EKF is to linearize nonlinear state propagation equations using Jacobian (matrix of partial derivatives) so that the propagation equations can be expressed as equation 7. To this end, partial derivatives of the next state $\hat{x}_k$ are computed with respect to the current state $\hat{x}_{k-1}$ yielding matrix F_k and partial derivatives of the next state $\hat{x}_k$ with respect to the control vector u_k are computed yielding matrix B_k:

$$F_k = \begin{bmatrix} \frac{\partial \hat{x}_{k,1}}{\partial \hat{x}_{k-1,1}} & \cdots & \frac{\partial \hat{x}_{k,1}}{\partial \hat{x}_{k-1,n}} \\ \vdots & \ddots & \vdots \\ \frac{\partial \hat{x}_{k,n}}{\partial \hat{x}_{k-1,1}} & \cdots & \frac{\partial \hat{x}_{k,n}}{\partial \hat{x}_{k-1,n}} \end{bmatrix}, \quad B_k = \begin{bmatrix} \frac{\partial \hat{x}_{k,1}}{\partial u_{k,1}} & \cdots & \frac{\partial \hat{x}_{k,1}}{\partial u_{k,m}} \\ \vdots & \ddots & \vdots \\ \frac{\partial \hat{x}_{k,n}}{\partial u_{k,1}} & \cdots & \frac{\partial \hat{x}_{k,n}}{\partial u_{k,m}} \end{bmatrix},$$

where state vectors are assumed to be n-dimensional. Following a similar pattern, nonlinear equations for mapping predicted next state $\hat{x}_k$ to predicted observations $\hat{z}$ are linearized using partial derivatives of observations (sensor readings) with respect to $\hat{x}_k$:

$$H_k = \begin{bmatrix} \frac{\partial \hat{z}_{k,1}}{\partial \hat{x}_{k,1}} & \cdots & \frac{\partial \hat{z}_{k,1}}{\partial \hat{x}_{k,n}} \\ \vdots & \ddots & \vdots \\ \frac{\partial \hat{z}_{k,h}}{\partial \hat{x}_{k,1}} & \cdots & \frac{\partial \hat{z}_{k,h}}{\partial \hat{x}_{k,n}} \end{bmatrix}, \tag{9}$$

where observation vector is assumed to be h-dimensional.

7.3 Unscented Kalman Filter

As mentioned in section 7.2, EKF tackles the nonlinear system equations using their first order approximations. In practice, this linearization method may lead to a posterior distribution with corrupted mean and covariance. This issue has been addressed by Unsented Kalman Filter (UKF) [293] via taking a derivative-free approach based on deterministic sampling. Instead of employing a Gaussian, UKF represents the state distribution by carefully choosing a set of sample points around the distribution mean. These points are known as the sigma points $\{\sigma_i^{(k-1)}, i = 1, ..., 2n + 1\}$ where n is the dimension of the state vector and $k - 1$ refers to the current time step. As pointed out in [293], the sigma points are selected as:

$$\sigma_0^{(k-1)} = \hat{x}_{k-1},$$
$$\sigma_i^{(k-1)} = \hat{x}_{k-1} + \left(\sqrt{(n + \lambda)P_{k-1}}\right)_i, \quad i = 1, ..., n,$$
$$\sigma_i^{(k-1)} = \hat{x}_{k-1} - \left(\sqrt{(n + \lambda)P_{k-1}}\right)_{i-n}, \quad i = n + 1, ..., 2n$$

where λ is a scaling parameter and $\left(\sqrt{(n + \lambda)P_{k-1}}\right)_i$ refers to the i-th column of matrix square root of $(n + \lambda)P_{k-1}$. The square root may be computed e.g. by lower triangular Cholesky factorization.

Similar to KF, UKF workflow includes prediction and correction phases followed by a new phase for sigma points selection. Assuming that the next state $\hat{x}_k$ is calculated as nonlinear mapping $f(.)$ of the current state $\hat{x}_{k-1}$, in the prediction phase, the sigma points undergo unscented transform [142] by being fed to the nonlinear function $f(.)$:

$$\sigma_i^{(k)} = f(\sigma_i^{(k-1)}), \quad i = 1, ..., 2n + 1.$$

(a) True mapping   (b) EKF prediction   (c) UKF prediction
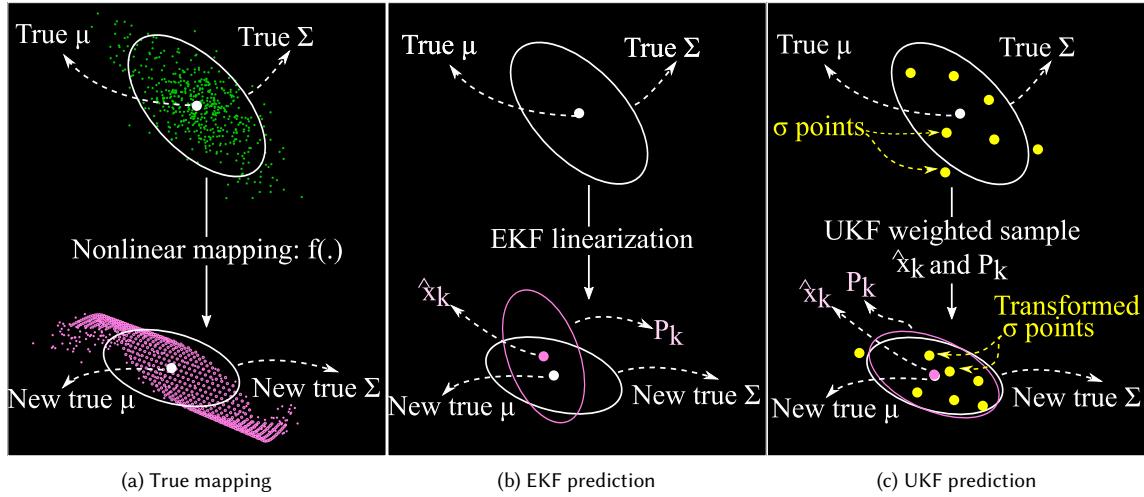
Fig. 9. Illustrating the state propagation mechanisms of EKF and UKF compared to the true mapping.

The weighted average of the new sigma points are then used to estimate the mean and covariance of the predicted state:

$$\hat{x}_k \simeq \sum_{i=1}^{2n+1} w_i^{(m)} \sigma_i^{(k)},$$

$$P_k \simeq \sum_{i=1}^{2n+1} w_i^{(c)} \left(\sigma_i^{(k)} - \hat{x}_k\right)\left(\sigma_i^{(k)} - \hat{x}_k\right)^T.$$

The transformed sigma points are then used to calculate the mean and covariance of the new state Gaussian distribution. The estimated mean and covariance are accurate up to the second order of Taylor series expansion [293].

The difference between EKF and UKF state propagation mechanism has been depicted in figure 9. Given the current distribution with mean μ and covariance Σ, in the absence of noise, applying nonlinear mapping $f(.)$ yields the new mean and covariance (figure 9a). The drawback of EKF linearization is apparent in figure 9b. The error introduced by the first order linearization has led to mismatch between true ($\mathcal{N}(\mu_{New}, \Sigma_{New})$) and predicted ($\mathcal{N}(\hat{x}_k, P_k)$) distributions. The distribution mismatch has been addressed by UKF in figure 9c. To this end, multiple sigma points sampled from the current distribution have been transformed based on $f(.)$. The transformed points are then used to calculate the new mean and covariance. Such an approach is more flexible and reliable compared to EKF first order linearization.

7.4 Multi-State Constraint Kalman Filter

Inertial navigation system (INS) is the backbone of autonomous navigation that can be applied on aerial [109, 209, 251], ground [56, 61, 186], and underwater vehicles [43, 127, 128]. Relying primarily on IMU, INS provides very cost-efficient navigation solution. However, INS localization accuracy degrades over time due to accumulative error. INS employs numerical integration over gyroscope and accelerometer readings to estimate the position and orientation of the platform it is mounted on. Applying numerical integration in successive time steps causes the aforementioned accumulated

error. That is why INS is usually paired with auxiliary sensor(s) e.g. GPS, magnetometer, camera, etc. for error correction. The choice of the auxiliary sensors depends on the characteristics of the environment (e.g. indoor/outdoor, in the air/on the ground/underwater).

Visual sensors such as cameras can be used in variety of environments and they are available at reasonable prices. That is the motivation behind Multi-State Constraint Kalman Filter (MSCKF) which is an EKF-based method designed to implement visually-aided inertial navigation in real-time [213]. The main difference between MSCKF and ordinary EKF is the fact that MSCKF uses a novel measurement model capable of capturing geometric constraints that arise when a static feature appears in multiple cameras with different poses. Thanks to this new measurement model, MSCKF is able to perform localization which is accurate up to linearization error of EKF only using visual sensor(s). Moreover, MSCKF computational complexity scales linearly with the number of features which is remarkable. One of the notable applications of MSCKF is visual-inertial navigation system OpenVINS implemented by Geneva et al. [109] which utilizes a VI-sensor (visual-inertial sensor) similar to the one shown in figure 10. The VI-sensor consists of two cameras and one 3-axis IMU which are hardware-synchronized.

The goal of MSCKF is 3D pose estimation of reference frame $\{I\}$ of an IMU strapped to a mobile (robot) platform with respect to some global reference frame $\{G\}$. To this end, the IMU state is written as [213]:

$$X_{IMU} = \left[{}_G^I \bar{q}^T, b_g^T, {}^G v_I^T, b_a^T, {}^G p_I^T \right]^T, \tag{10}$$

where the rotation from global frame $\{G\}$ to IMU frame $\{I\}$ is expressed as a unit quaternion ${}_G^I \bar{q}^T$. Moreover, ${}^G v_I^T$ and ${}^G p_I^T$ represent the IMU position and velocity with respect to $\{G\}$, and the 3×1 vectors b_g and b_a express the gyroscope and accelerometer bias affecting their measurements. Based on equation 10, the error state of the IMU reads as [213]:

$$\tilde{X}_{IMU} = \left[\delta\theta_I^T, \tilde{b}_g^T, {}^G \tilde{v}_I^T, \tilde{b}_a^T, {}^G \tilde{p}_I^T \right]^T, \tag{11}$$

where the additive error definition (i.e. difference between true value and estimated value) is used to compute error for position, velocity, and biases but the orientation error is described by the error quaternion $\delta\bar{q}$ such that the relation $\bar{q} = \delta\bar{q} \otimes \hat{\bar{q}}$ holds in which $\otimes$ stands for quaternion multiplication.

As the objective of MSCKF is an accurate estimation of IMU pose and orientation, the IMU state (equation 10) is used as part of the EKF state vector [213]:

$$\hat{X}_k = \left[\hat{X}_{IMU_k}^T, {}_G^{C_1} \hat{\bar{q}}^T, {}^G \hat{p}_{C_1}^T, ..., {}_G^{C_N} \hat{\bar{q}}^T, {}^G \hat{p}_{C_N}^T \right]^T, \tag{12}$$

where subscript k in $\hat{X}_k$ denotes the EKF state at time step k. In addition to the IMU state, equation 12 contains N pairs of $\{ {}_G^{C_i} \hat{\bar{q}}^T, {}^G \hat{p}_{C_i}^T \}$ items where ${}_G^{C_i} \hat{\bar{q}}^T$ is the quaternion expressing the rotation from the i-th camera frame $\{C_i\}$ to the global frame and ${}^G \hat{p}_{C_i}^T$ denotes the origin of frame $\{C_i\}$ expressed in the global frame. The aforementioned pairs corresponding to N cameras are considered in the EKF state to establish geometric constraints induced by observation of static features from multiple cameras poses. Looking at equation 12, the EKF error-state vector reads as [213]:

$$\tilde{X}_k = \left[\tilde{X}_{IMU_k}^T, \delta\theta_{C_1}^T, {}^G \tilde{p}_{C_1}^T, ..., \delta\theta_{C_N}^T, {}^G \tilde{p}_{C_N}^T \right]^T, \tag{13}$$

where $\{ \delta\theta_{C_i}^T, {}^G \tilde{p}_{C_i}^T \}$ expresses the i-th camera attitude and position error.

Recall that EKF consists of prediction and correction phases. In the prediction phase, the IMU state (i.e. position and attitude of frame $\{I\}$) is propagated according to the set of equations that will be described below.

The first equation expresses how quaternion $_G^I\hat{\bar{q}}$ evolves over time [213]:

$$_G^I\dot{\hat{\bar{q}}} = \frac{1}{2}\Omega(\hat{\omega})_G^I\hat{\bar{q}},\tag{14}$$

where $_G^I\dot{\hat{\bar{q}}}$ is the derivative of quaternion $\bar{q}$ at time step t and matrix $\Omega(\hat{\omega})$ is computed as [213]:

$$\Omega(\hat{\omega}) = \begin{bmatrix} -\lfloor\hat{\omega}\times\rfloor & \hat{\omega} \\ -\hat{\omega}^T & 0 \end{bmatrix}.\tag{15}$$

In equation 15, $\hat{\omega} = \omega_m - \hat{b}_g - C(_G^I\hat{\bar{q}})\omega_G$ where ω_m is the gyroscope sensor reading from which $\hat{b}_g$ and $C(_G^I\hat{\bar{q}})\omega_G$ are subtracted to cancel the effect of gyroscope bias and angular rate ω_G caused by planet's rotation, respectively. Note that the term $C(_G^I\hat{\bar{q}})$ represents the rotational matrix corresponding to quaternion $_G^I\hat{\bar{q}}$. The other component of matrix $\Omega(\hat{\omega})$ (equation 15) is $-\lfloor\hat{\omega}\times\rfloor$ which is known as the skew symmetric matrix of angular velocity vector $\hat{\omega} = [\hat{\omega}_x, \hat{\omega}_y, \hat{\omega}_z]^T$ [283]. The second and third equations of IMU state propagation express the evolution of gyroscope and accelerometer biases ($\dot{\hat{b}}_g, \dot{\hat{b}}_a$) over time [213]:

$$\dot{\hat{b}}_g = \mathbb{E}\left[n_g\right] = [0,0,0]^T, \; \dot{\hat{b}}_a = \mathbb{E}\left[n_a\right] = [0,0,0]^T,\tag{16}$$

where n_g and n_a are the gyroscope and accelerometer biases, respectively and they are modeled as zero-mean white Gaussian noise processes. That is why expectation of n_g and n_a are zero vectors as shown in equation 16. The fourth equation related to IMU state propagation reads as [213]:

$$^G\dot{\hat{v}}_I = C(_G^I\hat{\bar{q}})^T\hat{a} - 2\lfloor\omega_G\times\rfloor^G\hat{v}_I - \lfloor\omega_G\times\rfloor^2 \, ^G\hat{p}_I + G_g,\tag{17}$$

where $^G\dot{\hat{v}}_I$ is the derivative of IMU frame velocity which expresses the IMU frame acceleration with respect to frame $\{G\}$. Moreover, $\hat{a} = a_m - \hat{b}_a$ where a_m is the accelerometer sensor reading from which the accelerometer bias $\hat{b}_a$ is subtracted. The term $\lfloor\omega_G\times\rfloor$ in equation 17 is the skew symmetric matrix corresponding to planet's rotation ω_G and as mentioned before, $^G\hat{v}_I$ and $^G\hat{p}_I$ represent the IMU frame velocity and position with respect to frame $\{G\}$ and G_g stands for gravitational acceleration expressed in frame $\{I\}$. The last equation for IMU state propagation establishes the relation between the IMU velocity and position [213]:

$$^G\dot{\hat{p}}_I = {}^G\hat{v}_I.\tag{18}$$

To sum up, the set of equations 14-17 form the basis of the prediction phase of MSCKF. Considering that MSCKF is based on EKF, it is necessary to linearize equations 14-17:

$$\dot{\tilde{X}}_{IMU} = F\tilde{X}_{IMU} + G_{n_{IMU}},\tag{19}$$

where matrices F and $G_{n_{IMU}}$ are the results of linearizing equations 14-17 (see [213] for further details) and $n_{IMU} = \left[n_g^T, n_{wg}^T, n_a^T, n_{wa}^T\right]^T$ is the system noise consisting of gyroscope and accelerometer noise densities (n_g, n_a) and random walks (n_{wg}, n_{wa}). The covariance matrix Q_{IMU} corresponding to the system noise n_{IMU} is computed during the calibration process offline. Kalibr [22, 102, 103, 198, 229, 242] is one of the famous calibration tools implemented in ROS which can be used to determine IMU noise density and random walk parameters as well as camera intrinsic and extrinsic parameters. Moreover, Kalibr offers spatial (coordinate transform) and temporal (time synchronization) calibration of an IMU with respect to a mono/stereo camera system.
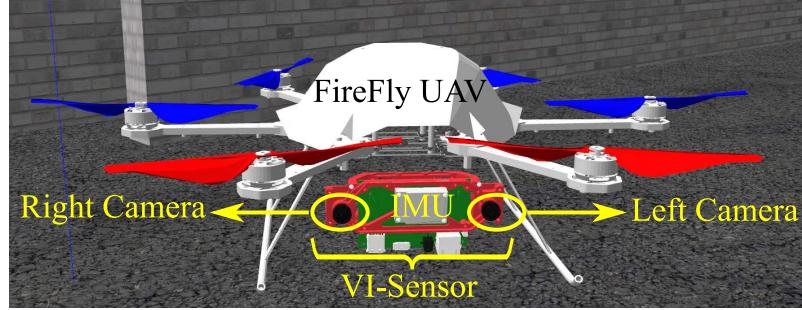
Fig. 10. Simulated Astec Firefly UAV equipped with VI-sensor: the image is captured while running Rotors_Simulator [27, 104] in Gazebo.

As mentioned before, MSCKF exploits the presence of 3D features in multiple camera poses. To this end, the EKF state vector is augmented with the camera poses from which the 3D features are observed. The camera poses are determined based on the current estimate of the IMU pose. The EKF covariance matrix is updated as well.

To implement the correction phase of the EKF, a new measurement model is used. Consider a single feature f_j which has been observed in M_j camera poses. Then according to the measurement model, each of these M_j observations is expressed as [213]:

$$z_i^{(j)} = \frac{1}{C_i Z_j} \begin{bmatrix} C_i X_j \\ C_i Y_j \end{bmatrix} + n_i^{(j)}, \quad i \in S_j, \tag{20}$$

where $n_i^{(j)}$ is a 2×1 vector expressing image noise and $^{C_i}p_{f_j} = \left[{}^{C_i}X_j, {}^{C_i}Y_j, {}^{C_i}Z_j \right]$ is the 3D position of the feature f_j in i-th camera pose frame $\{C_i\}$. To determine $^{C_i}p_{f_j}$, the 3D position of the feature f_j in global frame $(^{G}p_{f_j})$ is needed which is unknown. To estimate $^{G}p_{f_j}$, lease-squares minimization is employed. The estimated vector $^{G}\hat{p}_{f_j}$ is used to compute estimated 3D position of the feature f_j in frame $\{C_i\}$ (i.e. $^{C_i}p_{f_j}$) using the transformation from frame $\{G\}$ to frame $\{C_i\}$. Using $^{C_i}p_{f_j}$ and equation 20, the measurement model output (i.e. $\hat{z}_i^{(j)}$) can be computed. The difference between $\hat{z}_i^{(j)}$ and the measurement obtained from the environment $(z_i^{(j)})$ forms the measurement residual $r_i^{(j)}$ which is linearized to comply with EKF requirements. Gathering all residuals $\{r_i^{(j)}, i = 1, ..., M_j\}$ corresponding to feature f_j forms an equation enforcing constraint between all the camera poses from which feature f_j has been observed. The constraint equation can then be used in the update (correction) phase of the EKF.

7.5   Particle filter

Particle filter (PF) is one of the popular sample-based approaches for optimal state estimation in nonlinear and non-Gaussian scenarios. Contrary to the EKF, PF does not use local linearization which leads to more flexibility and representation power. However, more flexibility of PF comes at the expense of higher computational complexity. The goal of PF is estimation of state vector x_k at each time step k. PF requires process and measurement models. The process model $f(\hat{x}_{k-1}, u_k, v_{k-1})$ can be nonlinear and represents the mapping from current state $\hat{x}_{k-1}$ to the next state $\hat{x}_k$. Apart from the state vector $\hat{x}_{k-1}$, the function $f(.)$ receives deterministic control input u_k and process noise v_{k-1} which is independently and identically distributed across different time steps and captures the process model uncertainties. The

measurement model $z_k = h_k(\hat{x}_k, u_k, \zeta_k)$ establishes the relation between measurements z_k and state $\hat{x}_k$. The measurement noise is represented by ζ_k which is independently and identically distributed. In case of having multiple sensors, each sensor will have its own measurement model.

As pointed out in [119], PF expresses the measurement update of the next state as a probability density function (PDF) conditioned on the past observations $z_{1:k-1}$:

$$p(\hat{x}_k|z_{1:k-1}) = \int p(\hat{x}_k|\hat{x}_{k-1})p(\hat{x}_{k-1}|z_{1:k-1})d\hat{x}_{k-1}, \tag{21}$$

in which the probability of being at state $\hat{x}_k$ depends on sequence of applied control signals $u_{1:k} = \{u_i, i = 1, ..., k\}$ (not included in equation (21) for brevity) and sequence of obtained measurements $z_{1:k-1} = \{z_i, i = 1, ..., k - 1\}$ in the past $k - 1$ time steps. The probability $p(\hat{x}_{k-1}|z_{1:k-1})$ of equation (21) can be represented as:

$$p(\hat{x}_{k-1}|z_{1:k-1}) = \frac{p(z_{k-1}|\hat{x}_{k-1})p(\hat{x}_{k-1}|z_{-1:k-2})}{p(z_{k-1}|z_{-1:k-2})}, \tag{22}$$

where $z_{-1:k-2}$ refers to the set of last $k-1$ observations starting from time step $k-2$. Analytical solution of the integral in equation (21) is only possible under heavy restrictive assumptions that rarely hold in practice [92]. Therefore, instead of computing this integral, mean and covariance of the next state posterior (after measurement update) are approximated using equation (22) and weighted sums over N sampled points $\{x_k^i, i = 1, ..., N\}$:

$$\hat{x}_k = \sum_{i=1}^{N} w_k^i x_k^i. \tag{23}$$

In equation (23), the set of points $\{x_k^i, i = 1, ..., N\}$ are known as particles. During the initialization step, the particles are sampled randomly from the initial distribution on the system state. At each time step, in the prediction phase, the particles are propagated according to the process model $f(\hat{x}_{k-1}, u_k, v_{k-1})$ and in the correction phase (i.e. measurement update), they are weighted according to the measurement model and the obtained measurements [119]:

$$w_k^{(i)} = w_{k-1}^{(i)} p(z_k|x_k^{(i)}) = w_{k-1}^{(i)} p_{\zeta_k}\left(z_k - h(x_k^{(i)})\right), \tag{24}$$

where in equation (24), $p_{\zeta_k}(.)$ is the PDF of measurement noise.

Like any other method, PF has its own advantages and disadvantages. Following a sample-based approach, PF is capable of representing any state distribution with arbitrary shape. To make this point clear, consider the localization scenarios of a mobile robot as shown in figure 11. In figure 11a, the robot can only sense its distance to landmark L_1. Without having access to the landmark angle with respect to the robot, after PF update step, the particles will be spread around the landmark forming a circular shape. Adding a second landmark L_2 to the scenario restricts the possible region for robot position leading to a bimodal distribution of the particles (figure 11b).

Using PF, process and measurement models are no longer required to be linear which has made PF very popular in the robotics community. The disadvantage of using samples-based representation is that the target distribution will not be captured with reasonable accuracy if the number of particles is not sufficient. On the other hand, as the number of particles is increased, so will the computational complexity. PF performance may degrade due to loss of diversity of particle samples. Methods like resample-move algorithm [111], regularization [217], Rao-Blackwellisation [185], and multiple Monte-Carlo [185] can be utilized to deal with particles loss of diversity issue. PF has seen multiple variants such as sampling importance resampling (SIR) [38], auxiliary sampling importance resampling (ASIR) [234], and regularized particle filter (RPF) [85].
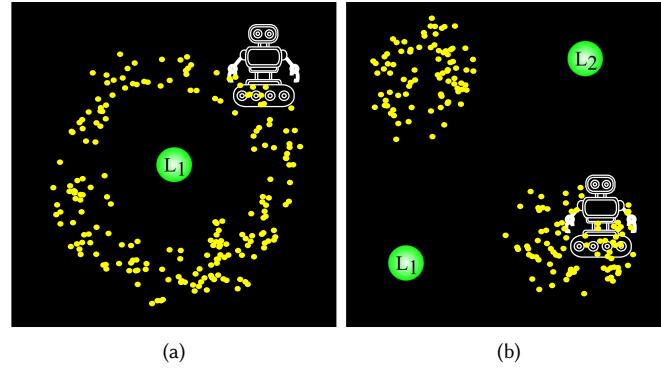
(a)	(b)

Fig. 11. Illustration of particles distribution with (a) one landmark L_1, (b) two landmarks $\{L_1, L_2\}$.

8 OBSTACLE AVOIDANCE

As mobile robots navigate through their operational environment, they have to deal with variety of obstacles in order to reach their destination. Therefore, the ability to avoid obstacles during navigation is a crucial module in autonomous systems. The location of obstacles may be known as part of the environment map but oftentimes the map is not known beforehand. Thus, it is highly desirable for autonomous mobile robots to be able to detect and avoid obstacles in real-time during navigation. Several obstacle avoidance methods exist in the literature which are reviewed in this section.

8.1 Legacy methods

The history of obstacle avoidance methods is almost as old as the emergence of mobile robots which is not surprising because without obstacle avoidance, mobility is not of much use. In this subsection, the classic methods of obstacle avoidance are reviewed.

8.1.1 Family of Bug algorithms. One of the simplest sets of obstacle avoidance methods is the family of Bug algorithms. As the name implies, these algorithms mimic the behavior of bugs upon encountering an obstacle on their way. The family of Bug algorithms began with the proposal of Bug1 and Bug2 [189]. The obstacle avoidance mechanisms of Bug1 and Bug2 have been depicted in figure 12a and 12b. According to figure 12a, whenever Bug1 bumps into an obstacle, it completes a cycle around it and in the second cycle, departs from the obstacle and continues its path toward the goal until the goal is reached or a new obstacle is detected. As shown in figure 12b, Bug2 assumes a hypothetical line from the start point to the goal point and circumvents any obstacle that intersects this line. Clearly, Bug2 is more efficient in terms of traveled distance compared to Bug1. Bug2+ [35] is an improved version of Bug2. In Bug2+ the condition for switching from obstacle-boundary-following to motion-to-goal has been revised in order to reduce the length of the path traveled from the start to the goal.

Another algorithm from the Bug family is called CBUG [105] which considers an ellipse with area equal to A_0 such that the start and the goal points form the ellipse's focus points. Within this ellipse, Bug1 is executed in an attempt to reach the goal. If the goal is not reachable in the current ellipse, its area is increasd to $2^i A_0$ where i is the iteration count. Within the new ellipse Bug1 is executed again. This trend is repeated until the goal is reached or it is concluded that the goal is not reachable. As shown in [105], CBUG is an optimal online planner with constant memory requirement.

Bug2+ was originally designed to work with unknown environments but it has been combined with A* to exploit the map of the environment in order to reduce the length of the path to be traveled. The resulting method is an anytime planner called ABUG [34]. The advantage of anytime planners is that they plan a sub-optimal path quickly and revise the path during the navigation. The term anytime planner means that the planning process can be interrupted at any time and a valid path is available. The more time is given to the planner the better the planned path will be. In the case of ABUG, whenever an obstacle is encountered, Bug2+ can circumvent it either from the its left or right side. Given the map of the environment, all of the obstacles on the path is known beforehand. That is where A* comes in. For each obstacle, a binary node is created corresponding to going around the left or right side of the obstacle. Set of the obstacle form a binary search tree. Running A* on the tree gives us the optimal path from the start to the goal.

Perhaps one of the most important Bug algorithms is TangetBug [145] based on which other methods such as WedgeBug [171] and CautiousBug [192] have been proposed. WedgeBug uses stereo vision for obstacle detection and its energy consumption, computational complexity, and memory usage has been minimized to meet the limited resources of mars rovers. The name WedgeBug is due to the fact that this algorithm is capable of navigation using the limited ($30°$ to $45°$) wedge-shaped field-of-view of stereo vision. This algorithm inspects a portion of the environment and if necessary, rotates its stereo vision system to inspect regions in the vicinity of the already observed region. This way, the limited field-of-view is dealt with. WedgeBug has been further modified to give birth to RoverBug [170] used in Rock7 mars rover [171].

TangentBug utilizes range sensors such as laser scanners to detect obstacles before actually bumping into them. This way, the traveled path toward the goal is reduced significantly. Each detected obstacle is presented by its two endpoints. For example, in figure 12c, the robot is surrounded by four obstacles that have been represented by the set of six endpoints $\{O_i, i = 1, ..., 6\}$. TangentBug has two primary behavior modes namely motion-to-goal (M2G) and wall-following (WF). Initially, the behavior is set to M2G. In this mode, the robot takes the direct route toward goal G. In case an obstacle interferes with the direct route, the robot computes the heuristic distance for the two endpoints of the obstacle as depicted in figure 12d. The heuristic distance $H(.,.)$ is computed as

$$H(x, G) = d(x, O_*) + d(O_*, G) \tag{25}$$

where x is the robot current position, O_* is one of the endpoints of the detected obstacle and $d(x, O_*)$ is the Euclidean distance between robot position x and endpoint O_*. After computing $H(x, G)$ for O_s and O_e, the endpoint with minimum distance is chosen for bypassing the obstacle. M2G behavior is continued until the goal is reached or $d(x, G)$ starts to increase in which case the behavior is switched to WF. During WF, the robot follows the obstacle boundary until one of the conditions below is satisfied:

- The goal is reached → the algorithm halts with success.
- The robot completes a full cycle around the obstacle → the algorithm halts with failure since the goal is not reachable.
- The equation

$$d_{Reach} < d_{Followed} \tag{26}$$

holds → the algorithm switches back to M2G behavior. $d_{Followed}$ is the distance between the goal G and the point on the blocking obstacle boundary which is closest to G. Additionally, $d_{Reach} = d(v_{leave}, G)$ where v_{leave} is a node on the local tangent graph such that equation 26 holds.
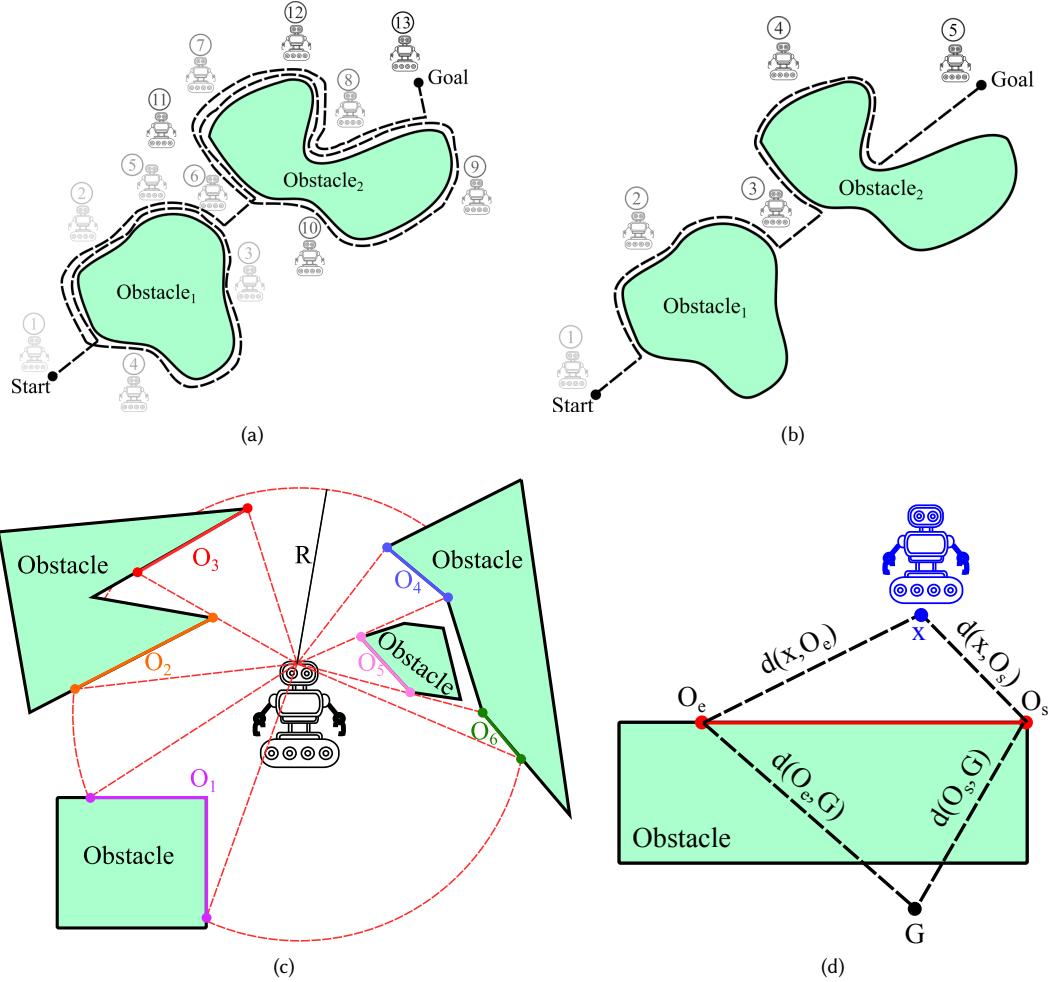
Fig. 12. Illustration of Bug algorithms: (a) Bug1 behavior, (b) Bug2 behavior, (c) local tangent graph used by TangentBug, (d) heuristic distance computation by TangentBug.

Another method that has been proposed with practical limitations in mind is I-Bug [274]. This method has been designed to deal with inaccurate localization and noisy sensor readings. It is assumed that the goal point emits a signal which is detectable by a special sensor onboard the robot. As the robot gets closer to the goal, the signal strength increases and as it moves away from the goal, the signal strength drops. The signal strength measurement sensor is the only accurate sensor onboard the robot. In addition to this sensor, the robot is also equipped with a sensor that determines whether the robot is facing the goal or not. I-Bug is able to reach the goal although the robot does not know its position and obstacle positions in the environment. The family of Bug algorithm has even been extended for handling moving obstacles. The resulting algorith is called distance histogram Bug (DH-Bug) [314].
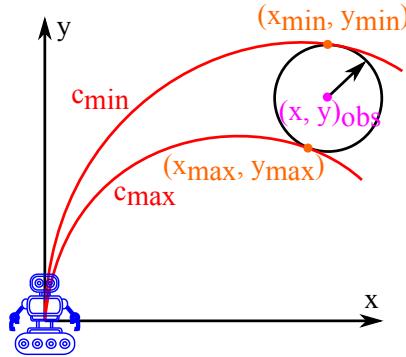
Fig. 13. CVM computing valid curvatures for bypassing the blocking of the obstacle.

8.1.2 Curvature velocity method. Bug algorithms are simple and easy to implement but they do not take into account the kinematics and/or dynamic constraints of the robot. On the contrary, curvature velocity method (CVM) [262] considers the robot kinematics constraints and even dynamic constraints to some extent.

To perform obstacle avoidance, CVM transforms the physical constraints imposed by environment blocking obstacles as well as acceleration and speed limitations of the robot to the velocity space. Assuming that the robot only moves along circle arcs with curvature $c = \omega/v$, the velocity space consists of rotational (ω) and translational velocity (v). CVM maps blocking obstacles from Cartesian grid to velocity space. The mapping is done based on the distance between the robot and the obstacles. After mapping, the range of v and ω that must be filtered out due to the blocking obstacles can be determined.

Naturally, obstacle avoidance methods must be real-time. To satisfy this requirement, CVM models the obstacles with a circular shape (figure 13). Such modeling may lead to poor representation for the shape of some of the obstacles. This is the main drawback of CVM.

8.1.3 Potential field. Potential field [153] was originally designed for control and obstacle avoidance of manipulator arms. However, due to its simplicity and effectiveness, potential field has been adopted by mobile robots community as well. Potential field models the robot as a point that is moved in the environment under influence of various attractive and repulsive potential fields. The goal is the source of attraction while the obstacles act as repulsive forces. In this setup, the total force $F(q)$ applied to the robot is the sum of repulsive (F_{rep}) and attractive (F_{att}) forces [153]:

$$F(q) = F_{att}(q) + F_{rep}(q). \tag{27}$$

The attractive and repulsive forces are computed by taking the gradient from the attractive and repulsive potential functions. Therefore, it is necessary to define these functions such that they are differentiable:

$$U_{att}(q) = \frac{1}{2}k_{att}\rho_{goal}^2(q), \tag{28}$$

$$U_{rep}(q) = \begin{cases} \frac{1}{2}k_{rep}\left(\frac{1}{\rho(q)} - \frac{1}{\rho_0}\right)^2, \ if \ \rho(q) < \rho_0 \\ 0, \ if\rho(q) \geq \rho_0 \end{cases} \tag{29}$$

In equation 28, the attractive potential field is defined where k_{att} is the scaling factor and $\rho_{goal}(q)$ is the Euclidean distance between point q and goal q_{goal}. Looking at equation 28, it is clear that as the robot gets closer to the goal, the strength of the attractive field is reduced. This way when the robot reaches the goal, it is stopped.

In equation 29, the repulsive potential field is defined where k_{rep} is the scaling factor and $\rho(q)$ is the minimum distance between the robot and the obstacle the repulsive field of which is to be computed for the robot. Additionally, in equation 29, ρ_0 is a threshold value. If the robot distance to the obstacle is less than ρ_0, it is affected by the repulsive field of the obstacle and otherwise, the robot is not affected by the field. As $\rho(q)$ is in the denominator of equation 29, the more the robot gets close to an obstacle, the stronger the repulsive field of the obstacle gets.

8.1.4   Dynamic window approach. Similar to CVM, dynamic window approach (DWA) [100] takes into account robot kinematics. To this end, DWA searches in velocity space which consists of all possible pairs of values of linear and angular velocities (v, ω). Another similarity between DWA and CVM is the assumption that the robot movement is only limited to circular arcs expressed as (v, ω). DWA forms a dynamic window in the velocity space such that the current linear and angular velocities of the robot will be at the center of that window. The dimensions of the window are determined based on the acceleration capabilities of the robot and the control cycle time. The infeasible velocity tuples (v, ω) within the dynamic window are pruned. The selection criterion of the tuples is that by using them, the robot will be able to stop before colliding with obstacles. The remaining tuples after the pruning are called admissible tuples. An objective function is applied on admissible tuples to determine the final movement direction for avoiding the obstacles and reaching the goal. The objective function reads as [100]:

$$O = a.heading(v, \omega) + b.velocity(v, \omega) + c.dist(v, \omega) \tag{30}$$

where heading measures the progress made toward the goal, velocity encourages fast forward motion, and dist is the distance between the robot and the closes obstacle during navigation.

8.1.5   Vector field histogram. Virtual force field (VFF) [51] has been proposed to meet the obstacle avoidance requirements of relatively fast moving robots. VFF is capable of working with unknown environments having different configurations of obstacles using a ring of cheap sonar sensors around the robot chassis. VFF models the sensed obstacles as a 2D Cartesian histogram called certainty grid. The value in each cell of the grid represents the certainty (likelihood) that the cell is occupied with an obstacle. The grid is initialized with all zeros indicating that there are no cells occupied with obstacles.Each time a sonar reading is available, the certainty value of the appropriate histogram cell is incremented. The appropriate cell is determined based on the orientation of the sensor acoustic axis and measured range. During navigation, robot sonar readings are accumulated in the appropriate grid cells which leads to a reasonable estimation of the obstacle locations in the environment even though the sonar readings are not accurate. The process of accumulating sonar readings in the grid has been illustrated in figure 14a. As can be seen, the certainty values of the cells close to the obstacle have been increased due to successive sonar readings.

After populating the certainty grid with reasonable values, a rectangular sub-region of the grid (called active region) is considered with the robot at its center. The obstacles within the active region are considered for obstacle avoidance using the potential field (section 8.1.3) method.

The main drawback of VFF is the inability to bypass certain obstacles configuration. For example, while moving down a hallway, the repulsive force exerted on the robot due to the left and right walls paralyzes VFF obstacle avoidance. The reason is the sudden reduction of the certainty grid to direction and magnitude of the repulsive force vector F_r in a single step [52]. This issue has been addressed by vector field histogram (VFH)[52] method using a two-step reduction
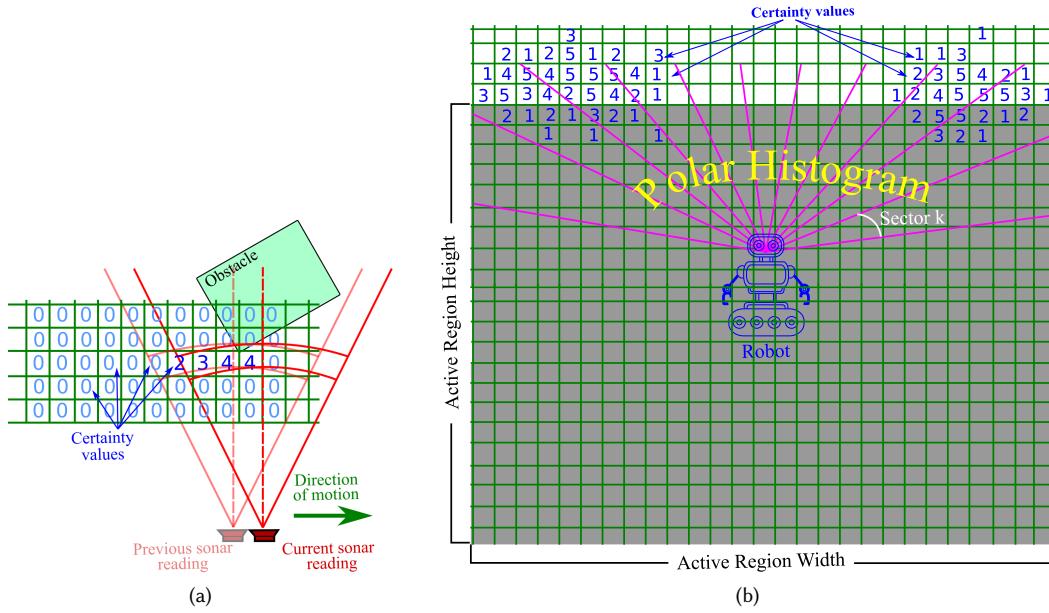
Fig. 14. (a) VFF certainty grid update process using successive sonar readings, (b) illustration of how certainty grid is mapped to polar histogram by VFH method.

procedure. After populating the certainty grid, it is reduced to a 1D polar histogram. To this end, the active region (the gray region in figure 14b) is partitioned into multiple sectors with angle α. To compute the obstacle density for each sector k, for each certainty grid cell (i, j), an obstacle vector with magnitude $m_{i,j}$ and direction $\beta_{i,j}$ is computed [52]:

$$\beta_{i,j} = \tan^{-1}\left(\frac{y_i - y_r}{x_i - x_r}\right), \tag{31}$$

$$m_{i,j} = \left(c_{i,j}^*\right)^2 \left(a - bd_{i,j}\right), \tag{32}$$

where (x_i, y_i) is the coordinates of cell (i, j) and $c_{i,j}$ is its certainty value, (x_r, y_r) is the robot current position, a and b are positive constants, and $d_{i,j}$ is the distance between cell (i, j) and the robot position. Based on the direction component $\beta_{i,j}$, the sector k within which cell (i, j) resides is decided [52]:

$$k = \lfloor \frac{\beta_{i,j}}{\alpha} \rfloor.$$

After determining the appropriate sector for all of the certainty cells, the obstacle density for each sector k is computed [52]:

$$h_k = \sum_{i,j} m_{i,j}. \tag{33}$$

Using equation 33, the polar histogram is formed with horizontal axis showing $[0, 2\pi]$ degrees and vertical axis showing the obstacle density values for sectors at different angles. To determine the safe portions of the active region for robot movement, a threshold is applied on the polar histogram vertical axis. Sectors with obstacle density below the threshold are considered safe. VFH has been further extended to VFH+[287] to improve robot safety during navigation.

8.2 Obstacle avoidance using RL and DL

Over the years, computers have evolved dramatically reducing the computational burden for mobile robot platforms. This has paved the road toward using more powerful methods like DL and reinforcement learning which have higher processing power requirements. In this subsection, the state-of-the-art obstacle avoidance methods are reviewed.

To be able to avoid environmental obstacles, the distance between the robot and the obstacle is required. LiDAR sensors provide high precision range data but they are costly. Much cheaper alternatives are stereo cameras that provide depth data via disparity map computed between the two cameras. However, acquiring correct range data depends on appropriate calibration of stereo cameras and the estimated range is limited to a few meters (which is much less than LiDAR). To avoid the calibration process, one can use monocular cameras at the cost of losing depth data. However, recent advancements in DL have made it possible to estimate depth using only a single camera. To this end, Wenzel et al. [296] utilized generative adversarial networks (GAN) [113] to convert RGB images to their corresponding depth images. This approach not only makes depth estimation from a single camera possible but also bridges the gap between simulation and real-world applications. The depth estimation is made possible thanks to the powerful feature extraction of DL methods. Bridging the gap between simulation and real-world is realized due to the fact that GAN does not care about the source of RGB images and it can estimate the depth maps for simulated as well as real images. The authors trained deep reinforcement learning agents such as deep Q networks (DQN) [203] and proximal policy optimization (PPO) [256] to generate appropriate control commands for input depth images. History of utilizing RL for obstacle avoidance precedes the birth of the infamous DQN approach. As an example, instead of relying on a lookup table, Huang et al. [134] used a NN to estimate the Q function of a Q-learning agent.

In addition to RL, DL-based obstacle avoidance modules can be trained in supervised manner as well. Liu et al. [183] implemented indoor obstacle avoidance by feeding raw images to a DL model based on AlexNet[168] and using the model output as steering commands. Based on the input image, the robot is steered left, right and commanded to go straight. The training of the DL model has been done using data collected during robot motion controlled by a human operator. To facilitate the implementation, ROS has been used.

9 NAVIGATION METHODS

Autonomous navigation methods are either map-based or mapless. In map-based approaches, the environment map is known before the navigation starts but it is not the case for mapless methods. Obviously having the environment map facilitates efficient path planning. However, in many scenarios, the environment map is not known beforehand which has led to the development of mapless navigation and simultaneous localization and mapping (SLAM) reviewed in section 10.

9.1 Map-based navigation

The quality of map-based navigation is directly affected by how the environment map is represented and how accurate it is. Map representations are either metric or topological. In metric maps, the location of detected obstacles, landmarks, robot, etc. are expressed with respect to a certain reference frame [201]. A typical metric map corresponding to environment in figure 15a has been illustrated in figure 15b in the form of a 2D grid. As can be seen, in the grid map, the detected obstacles, robot position, etc. are represented with respect to some specific XY reference frame. Each cell of this grid represents the probability of being occupied with an obstacle. Grid cells with darker black are more likely to be occupied by obstacles. Probabilistic occupancy grids can be implemented using multiple approaches. Bayesian
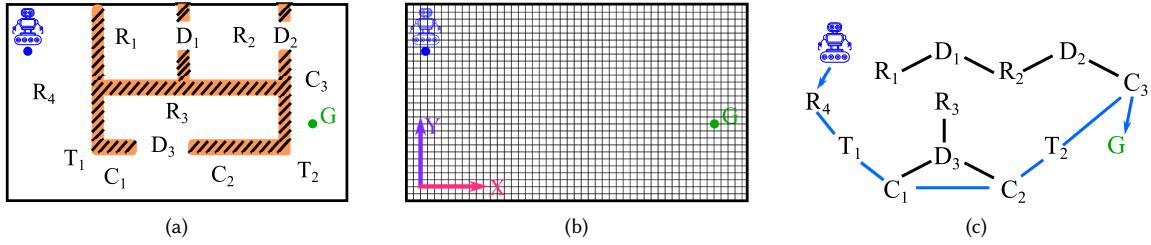
Fig. 15. Types of map representation: (a) actual environment, (b) metric map, (c) topological map.

occupancy grid (BOG) [210] relies on Baye's rule to estimate the probability of being occupied for each cell of the grid map. BOG has been used in some of the mapping approaches [91, 133]. Other methods to implement occupancy grid are based on reflection map [281] and expectation maximization [280].

Naturally, when range sensors are available, expressing the environment map as a 2D or 3D grid is useful [116, 123, 285]. However, the memory requirement for storing map of large scale environments may be prohibitive[98] since the memory size needed to store the map increases as range data are obtained and processed [98]. Several attempts have been made to keep the memory requirement of grid maps manageable [90, 133, 152].

Apart from range sensors, camera sensors provide rich visual features as well. These features can be used to implement landmark maps [208, 265] which are effective for scenarios that locally distinguishable features are present in the environment. As an example, consider the environment in figure 15a again and assume that there are distinguishable features in different regions of this environment. Using these features, it is possible to generate a topological map similar to the one in figure 15c. Each node of the topological map represents a particular region in the environment based on the aforementioned distinguishable visual features. The regions that are accessible from one another are connected by edges in the topological map. For example R_4 (room 4) is accessible via T_1 (turn 1) which is shown in the topological map (15c) with an edge connecting R_4 to T_1. Treating the topological map as an undirected graph, off-the-shelf graph path planning algorithms can be used to find the shortest path between robot position and the goal position (denoted by G in figure 15c). In real-world applications, it is likely that the environment map is only partially known. To carry out autonomous navigation under such circumstances, Sgorbissa and Renato proposed a hybrid navigation approach to exploit a possibly incomplete map of the environemnt and local perceptions during robot motion in order to achieve safe and collision-free navigation [257].

9.2 Mapless

Autonomous mapless navigation is the act of controlling a mobile robot toward the desired destination without having access to the environment map. The problem of mapless navigation has been tackled several times. As pointed out in [120], mapless vision-based techniques for indoor robot navigation can be categorized as:

- methods based on optical flow: in a nutshell, the motion of visual features in an image sequence is defined as optical flow. One of the mapless navigation methods based on optical flow is robee [48]. In this method, vision system of a bee is mimicked using stereo cameras to navigate through the environment without colliding with obstacles. Optical flow has also been used to implement corridor following behavior [86] as well as a feature-based navigation system [276].

- appearance-based matching: in this approach, certain templates and images of the environment are memorized. The memorized data are then associated with controlling commands to guide the robot toward desired destination [81]. The main challenge of appearance-based methods is finding appropriate methods to express environment in terms of prominent features as well as defining matching criteria for recognizing certain regions of the environment [50]. Visual memory [107] is one of the appearance-based methods in which a sequence of images are captured during robot motion. The captured images are associated with appropriate motions required for reaching a specific destination. The images, their associated motions and a template matching algorithm can be used to construct the robot route. Another appearance-based approach is multidimensional histograms that are extracted based on statistical analysis (related to color, texture, and edge density) of captured images [254, 312]. The recognition of places is done via matching histograms of stored places and histograms computed for the current position of the robot.
- object recognition: appearance-based approaches are primarily focused on memorizing their surrounding environment. However, symbolic navigation can be used instead [154, 155]. In this approach, the detected obstacle are stored in a 2D grid map called the S-map. Using this map, path planning is done and robot is given symbolic commands such as "go to the main exit" or "go to the corner in front of you" [120]
- feature-based navigation methods: in these methods, the motion of the robot is determined by extracting features from image sequence and tracking the relative displacement of those features during robot movement. An example of feature-based navigation methods is funnel lane [65] in which a robot is tasked to follow a previously traversed path by matching image feed from its mono camera with the previously recorded image sequence called visual path. Funnel lane method is fully qualitative and does not need environment map of any kind. This method has been extended to sloped funnel lanes [149] to address issues such as ambiguity between translation and rotation in certain scenarios and achieve better maneuverability.

RL has also been utilized to implement map-less autonomous navigation in highly dynamic environments where quick and adaptive action selection is required. As an example, Fan et al. [96] optimized a mapless navigation policy capable of robot control in highly dynamic environments such as pedestrian crowds. The optimization of the policy as a local planner was done using PPO and the policy was expressed as a CNN with four hidden layers.

10   SLAM

Simultaneous Localization And Mapping (SLAM) can be thought as a concept rather than a specific algorithm [245]. The term was coined after the original work of Hugh Durrant-Whyte and John J. Leonard [173] whose research was founded based on earlier work by Smith, Self and Cheeseman [265]. In a nutshell, SLAM objective is to localize a mobile robot in an unknown environment and at the same time estimate the environment map in terms of observed environment landmarks. Roughly speaking, the main components of SLAM are landmark extraction, data association, state estimation, and state and landmark update.

The high level steps of robot pose prediction and correction during SLAM have been depicted in figure 16. As shown in figure 16a, the robot measures its distance to three landmarks (L_1, L_2, L_3) and then moves to a new location. Using robot odometry model, the new location is predicted which has been shown as a robot figure with solid lines in figure 16b. Once again, the distance to the three landmarks is checked (figure 16c). Based on the obtained measurements, the robot pose is corrected leading to orange robot in figure 16d. As can be seen, the corrected pose is much closer to the ground truth pose (cyan robot in figure 16d). Sensors like IMU provide valuable information about the robot motion
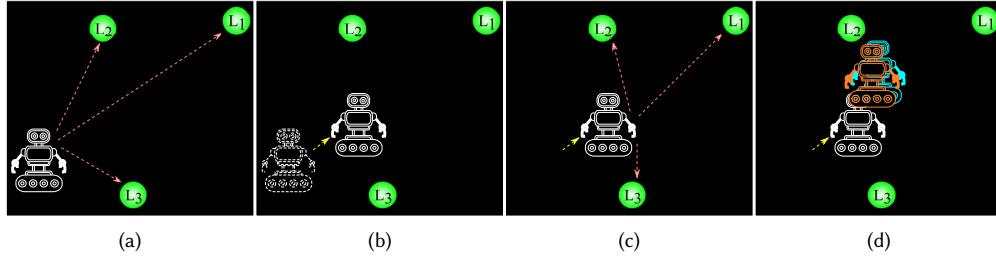
Fig. 16. Illustration of mobile robot localization using SLAM and detected landmarks: (a) distance measurement to three landmarks $\{L_1, L_2, L_3\}$, (b) robot pose prediction after movement (dotted robot: old pose, solid robot: new predicted pose), (c) sensing landmarks again, (d) pose correction based on measurements in part c (white robot: predicted pose before correction, orange robot: pose after correction, cyan robot: true pose).

and play vital role in the prediction step of the SLAM [147]. On the other hand, measured range data or visual features obtained from cameras are used in the update step of the SLAM in order to reduce the prediction error [204].

SLAM solutions can be categorized into two broad classes of filtering (online SLAM) and smoothing (full SLAM) methods [33] which are briefly introduced below.

Smoothing approaches rely on least-square optimization to estimate the robot full trajectory using all of the sensor measurements collected so far. Therefore, the memory consumption is high however, it is still possible to implement full SLAM in real-time [270]. Estimatating the robot full trajectory $\hat{x}_{1:K} = \{\hat{x}_k, k = 1, ..., K\}$ (with length K) given the sequence of control data $u_{1:K} = \{u_k, k = 1, ..., K\}$ obtained from odometry sensors and observations $z_{1:K} = \{z_k, k = 1, ..., K\}$ obtained from the environment is what separates full SLAM from online SLAM [166]. In full SLAM, the following posterior probability must be estimated [115]:

$$p(\hat{x}_{1:K}|u_{1:K}, z_{1:K}, x_0) \tag{34}$$

where x_0 is the initial state of the robot at the start of the navigation. To estimate the posterior probability in equation (34), one has to deal with intractable high dimensional state spaces unless certain conditions such as static world and Markov property are assumed. Taking these two assumptions into account, Graph SLAM was proposed by Lu and Millios [188] which uses a graph-based representation to capture the underlying structure of the SLAM problem. Each graph node consists of a robot poseThe graph edges represent spatial constraints between nodes which are based on the collected observations z_k or odometry data u_k.

Graph SLAM decomposes the problem into two parts: (a) graph construction (known as front-end) based on raw measurements and (b) finding the most likely configuration of the robot poses (known as back-end) such that the set of constraints imposed by the graph edges is satisfied [115]. A typical pose-graph has been depicted in figure 17a in which Ω_{ij} is the information matrix and $e_{ij} = z_{ij} - \hat{z}(x_i, x_j)$ (the error function) is the difference between actual observation z_{ij} gathered by the robot and the predicted observation $\hat{z}_{ij}(x_i, x_j)$ which represents x_j in the coordinate frame of x_i. For more clarity, the connection between the two poses x_i and x_j due to their mutual observation z_{ij} has been depicted in figure 17b. As can be seen, the uncertainty of observing z_{ij} is captured by Ω_{ij}.

In SLAM, data association is the matching of measurements obtained from the environment with the existing map which is vital for successful SLAM implementation [313]. Data association is usually handled by the SLAM front-end. To achieve a consistent estimate of the prior over robot trajectory $p(\hat{x}_{1:K}|z_{1:K}, u_{1:K})$, interleaving of the front-end
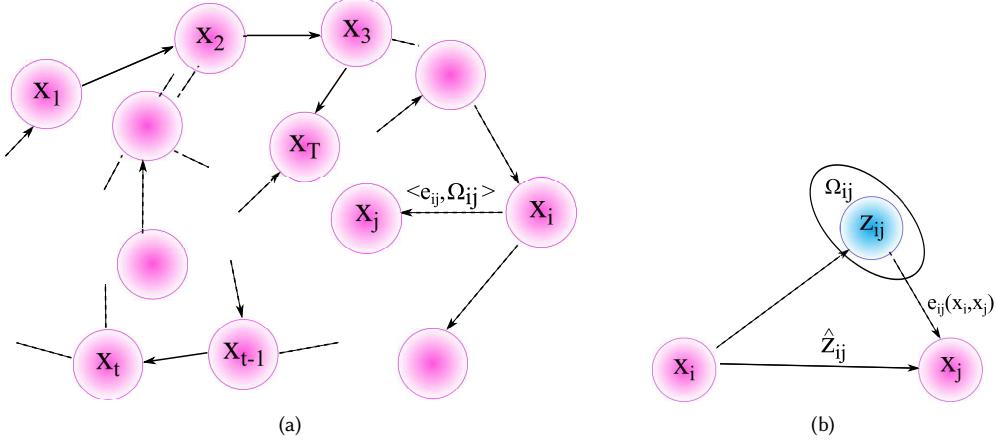
Fig. 17. (a) a typical pose-graph representation for Graph SLAM: nodes represent robot pose and edges express spatial constraint between nodes due to obtained measurements, (b) nodes x_i and x_j connection due to mutual observation z_{ij}.

and the back-end is crucial. The data association problem has been tackled using different approaches based on joint compatibility branch and bound [220], spectral clustering [275], and backtracking [124].

The second category of SLAM methods is online approaches which have lower memory consumption compared to full SLAM methods. This stems from the fact that online methods only rely on the previous pose estimate and sensor readings. However, lower memory consumption comes at the cost of decreased accuracy compared to the full SLAM approaches.

Apart from being online or full, SLAM methods can be also categorized based on the sensor(s) they use for collecting observations from the environment. Methods that use range sensors as their source of observation and methods that use camera sensors are called LiDAR SLAM and visual SLAM, respectively. Sections 10.1 and 10.2 are devoted to reviewing some of well-known methods belonging to LiDAR and visual SLAM and they are summarized in table 3.

10.1 LiDAR SLAM

Indoor environments are heavily structured and having obstacles within 60 meters from the robot is quite likely. Such conditions are ideal for using laser scanners to sense and map the environment. SLAM approaches based on LiDAR sensors are known as LiDAR SLAM. Despite the ability to provide high precision data, LiDAR sensors are usually accompanied by hefty price tags making LiDAR SLAM an expensive solution.

The SLAM literature consists of various LiDAR-based SLAM approaches such as Hector slam [165], keyframe-based 3D LiDAR SLAM [118], Google Cartographer [131], GMapping [116], and TinySLAM [269]. Traditionally, LiDAR sensors used rotary mechanisms to scan the surrounding environments. However, recent developments have led to low power and low cost LiDAR sensors based on Micro-Electro-Mechanical Systems MEMS technology. MEMS LiDARs are available at much more affordable prices which makes them ideal choices for the robotic researchers. However, lower cost of MEMS LiDAR comes at the price of shorter range (about nine meters) and smaller field of view (FOV) which makes LiDAR SLAM challenging. Fusing MEMS LiDAR data with other data sources such as IMU can improve

the SLAM performance [292]. It has been shown that utilizing fast point cloud registration can further improve the localization and mapping quality [294]. While using LiDAR sensors provide range data with outstanding accuracy, their relatively costly price tag is the motivation for using cheaper alternatives such as ultrasonic sensors [71, 76, 143, 301].

10.2 Visual SLAM

Another type of SLAM methods is visual SLAM which is primarily focused on mono/stereo camera(s). Visual SLAM must not be confused with visual odometry (VO) [304]. Visual SLAM is all about preparing the global map of the explored environment whereas VO is used to estimate the robot pose locally.

One natural approach to implement visual SLAM is extracting features from camera images and use them to determine the scene geometry and camera pose. Feature-based methods are either based on filtering like the EKF-based visual-inertial odometry [179] or based on keyframes [159]. At first glance, feature-based approaches seem very intuitive. However, these methods are limited to the information that are captured by the type of the features extracted. As pointed out in [93], feature-based methods will have difficulty with man-made environments. Some edge-based [87, 161] and region-based [70] have been proposed to tackle the aforementioned limitation. Among feature-based methods, Parallel Tracking And Mapping (PTAM) [160–162] computational complexity is low enough to be runnable on a mobile phone using its camera as the source of visual data. PTAM uses FAST-10 corner detector [246] for feature extraction. ORB-SLAM [214] is based on ORB features [248] capable of handling small/large and indoor/outdoor environments. The use of ORB features has made ORB-SLAM runnable in real-time without requiring a graphical processing unit (GPU). ORB-SLAM has received two important extensions resulting in second [215] and third [58] versions of this method. The second version provides support for stereo and RGBD cameras. The first version was limited to pin-hole camera model but the third version (based on first and second versions and ORB-SLAM-VI [216]) works with any camera type assuming that the camera module is provided. OV^{2}SLAM [97] also offers a feature-based approach which is accurate, robust, and real-time outperforming ORB-SLAM.

It is also possible to take a direct (feature-less) approach in which all of the information in the image intensities are used. Using the direct approach, loss of prominent information due to limited representation power of features is avoided altogether. One of the direct methods is Large-Scale Direct Monocular SLAM (LSD-SLAM) [93] capable of building large-scale and consistent environment 3D maps in real-time. Direct Sparse Mapping (DSM) [315] is based on photometric bundle adjustment (PBA) that introduces the idea of map reuse in direct methods. DSM showed the importance of mid-term data association.

While majority of SLAM methods are either visual or LiDAR-based, Multi-cue Direct SLAM (MD-SLAM) [82] is the only open source approach that supports LiDAR and RGBD sensors in a unified manner.

10.3 Loop closure techniques

An efficient SLAM module must be capable of recognizing previously visited places in order to reduce the localization and mapping error that has been accumulated during multiple time steps. Such capability is called loop closure detection. Traditionally, feature descriptors such as SIFT [187] and SURF [47] were used to loop closure detection. However, these methods are computationally inefficient for real-time applications. Researchers have developed loop closure mechanisms [74, 106, 271] based on bag-of-words (BoWs) approach to remedy the high computational load of descriptors such as SIFT and SURF. Bag-of-words refers to a dictionary of visual features extracted from the environment and clustered to form the so-called words. Despite being fast, BoWs rely on fixed visual features that must be known beforehand. It comes as no surprise that BoWs will not perform well in unexplored environments. To make

Table 3. Summary of some of well known SLAM methods.

Method	Sensor					SLAM approach		Loop closure detection
	Radar	LiDAR	Visual			Online	Full	
			Mono	Stereo	RGBD			
Hector SLAM [165]		✓				✓		
LSD-SLAM [93, 94]			✓				✓	✓
ORB-SLAM [214]			✓				✓	✓
ORB-SLAM2 [215]			✓	✓	✓		✓	✓
ORB-SLAM3 [58]			✓	✓	✓		✓	✓
ORB-SLAM-VI [216]			✓	✓	✓		✓	✓
Mono-SLAM [77–79, 206]			✓			✓		
PTAM [160–162]			✓				✓	
DSM [315]			✓				✓	
OV^{2}SLAM [97]			✓	✓			✓	✓
RT-SLAM [247]			✓	✓		✓		✓
Google Cartographer [131]		✓					✓	✓
GMapping [116]		✓				✓		✓
TinySLAM [269]		✓				✓		✓
RadarSLAM [132]	✓						✓	✓
MD-SLAM [82]		✓			✓		✓	✓

BoWs adaptable to new regions (not explored before), it may be created in an online step [106]. However, the memory requirements of the online approach are high. Following the significant achievements of DL in various domains, it has been adopted by the robotics community [219, 238, 261, 310]. A notable example of reinforcing SLAM with DL is DXSLAM [13, 176] which uses CNN to extract salient features for boosting SLAM performance.

10.4 Public SLAM datasets

Conducting research on SLAM methods requires access to multiple sensors and mobile robot platforms. Apart from the cost of preparing the aforementioned hardware, it may be difficult for individual researchers to put together a SLAM-capable hardware setup. An alternative approach is development and evaluation of SLAM methods on public datasets obtained from real robots. Yin and Berger [303] have reviewed 27 publicly available datasets that have been collected on public roads. Some of these datasets are suited for evaluation of SLAM approaches. The summary of these datasets is available in table 4. As can be seen, existing datasets cover variety of environments being indoor, and outdoor, having different weather and lighting conditions, etc. Recently, semantic SLAM has gained popularity which is the inspiration for preparing synthetic datasets suitable for semantic SLAM. SSS-dataset (ShanghaiTech Semantic SLAM) [59] is an example of such datasets which provides accurate ground truth for the scene composition and individual object shapes and poses.

11 FUTURE WORK

In this section, the future research direction for autonomous mobile robots is presented.

Table 4. Summary of public SLAM datasets.

Dataset	Indoor	Urban	Suburb	Rural	Attributes	Loop closure	Sensor data
AMUSE [167]		✓			Weather: {snowy, water & snow on lens}	✓	Omni-directional multi-camera, height sensor, IMU, velocity sensor, GPS
CCSAD [121]		✓			Recorded in developing countries, variable lighting, tunnel at night, narrow roads		Stereo camera, IMU, GPS
Cheddar Gorge [263]				✓	Weather: {dry, sunny, clear, cold}	✓	Stereo/mono camera, mono IR camera, Velodyne 64 LiDAR, high-end GPS/IMU, low-end IMU, wheel encoders, laser tracker for sensor pose measurement
CMU Visual Localization Dataset [40]		✓	✓		Versatile weather & light conditions	✓	2 Mono cameras, 4 Sick LiDARs, GPS, IMU
ETH3D [255]	✓	✓			Suitable for evaluating visual inertial mono, stereo, and RGB-D-SLAM [14], accompanying source code [15], synchronized global shutter cameras, modeling rolling shutter effect not required	✓	Stereo/mono/RGBD cameras, IMU, Faro Focus X 330 laser scanner
Ford [231]		✓			Weather condition: sunny, time-registered data, accompanied with utility source codes	✓	GPS, IMU, Velodyne 3D LiDAR, 2 push-broom forward looking Riegl LiDARs, omnidirectional camera
2D Cartographer Backpack Deutsches Museum [17]	✓				ROS .bag format, intended for 2D SLAM	✓	IMU, two 2D LiDAR (horizontal and vertical)
KITTI [108]		✓		✓	grayscale/color images, ground truth provided based on GPS and laser scanner	✓	2 Mono grayscale cameras, 2 mono color cameras, Velodyne 64 LiDAR, GPS, IMU
Malaga [49]		✓	✓		Direct sun, accompanied with C++ sample codes for parsing raw log files	✓	Stereo camera, 3 Hokuyo UTM-30LX LiDAR, 2 Sick LiDARs, GPS, IMU
Oxford [191]		✓			Different lighting/weather conditions, accompanied with Matlab, Python tools	✓	Stereo camera, 3 monocular color cameras, 2 Sick 2D LiDARs, Sick 3D LiDAR, GPS, INS

11.1 Mobile robots deployment in real-world

One of the ultimate goals of pursuing research on autonomous mobile robots is their deployment in manufacturing, hospitals, warehousing, etc. Implementing a multi-robot setup is quite natural to complete tasks faster[218]. Deciding on the operations that must be carried out in centralized or decentralized manner is an important future direction [101]. It is also necessary to support multi-robot setup with efficient decision making about the set of tools that must be mounted on the robots to carry out specific tasks [101].

Autonomous navigation has also been used to develop autonomous ships with different levels of autonomy. As future works, it is important to establish cooperative behavior between maritime autonomous surface ships (MASS) which have autonomy levels of D3 and D4. D3 autonomy is remote control of ships without having seafarers onboard while D4 refers to fully autonomous ships [228]. It is also critical to analyze safety in human-machine interaction setup [157].

11.2 SLAM

Although there is multitude of works in SLAM literature, due to complex and challenging nature of SLAM, there are still important research gaps for future work. For efficient SLAM implementation, it is crucial to keep balance between exploration (visiting new places) and exploitation (revisiting known places for uncertainty reduction) [279] which is addressed by active SLAM. Future works [235] for active SLAM consist of active spatial perception, proposing evaluation metrics to facilitate comparison of methods, and performing benchmarks to move toward real-world applications.

SLAM methods commonly assume that the world is static. However, real world is full of moving objects which contribute to failure of feature recognition causing tracking failure in SLAM [80]. As future work, DL and spectral methods [298, 299] can be used to achieve more robust feature recognition and tracking [190]. It may be possible to detect and exclude dynamic objects from map construction process of SLAM. Currently, dynamic SLAM methods such as [272, 273] exist to deal with dynamicity of the environment. Using semantic information to devise semantic SLAM is an ongoing field of research for addressing dynamic SLAM [190]. Additionally, using semantic SLAM to take the autonomy level of robots to the next level is challenging future task [64].

While DL has shown some merits for loop closure detection, due to its limitations, the true loop detection is still unaddressed [37]. The limitation is due to sensitivity of vision-based loop closure detection methods to different illumination conditions. LiDAR-based loop does not suffer from such limitations however its performance is degraded due to weather conditions such as rain. Coming up with hybrid methods, exploiting the strengths and covering the weaknesses of both LiDAR-based and DL-based methods seems to be a necessary future direction [37].

To deploy visual SLAM methods in real world applications, they must be executable on embedded platforms such as smartphones which have limited processing power. Therefore, one of the important future works for SLAM is to keep the balance between real-time execution and accuracy [64].

11.3 Autonomous exploration in unseen environment

Being able to navigate and explore unseen environments autonomously is highly desirable. To this end, Shah et al. exploited RL to propose a method called RECON (Rapid Exploration Controllers for Outcome-driven Navigation) [258] which was trained on visual dataset [16] collected during 18 months from diverse real-world environments including different seasons and lighting conditions. Two primary components of RECON are uncertainty-aware and context

conditioned goal representation as well as topological map representation. Nodes of the map represent egocentric observations and the edges are distances between nodes.

Extending RECON, Shah and Levine [259] focused on long-range autonomous navigation through unseen outdoor environments without requiring accurate map and/or sensor data. The advantage of their method called ViKiNG (Vision-Based Kilometer-Scale Navigation with Geographic Hints) is twofold. First, ViKiNG manages to reach the given destination using inaccurate data (GPS coordinates, satellite maps and schematic roadmaps) as a heuristic during planning. Second, this method does not build geometric maps and only relies on topological representation of the environment. This way, the issue of building and storing environment dense map for long distances is not encountered on robots with lightweight processing/storage platforms. The authors have trained ViKiNG using trajectories that are only 80 meters long but have managed to reach goals up to three kilometers away. As pointed out by Shah and Levine, one of the important future works is devising models similar to ViKiNG that are capable of accepting inputs such as textual instructions and paper maps.

12 CONCLUSION

Autonomous navigation has come a long way but the journey still continues. Different sensor combinations and mathematical approaches have been tried leading to variety of approaches; each having advantages and disadvantages of their own. In this paper, we strove to present a comprehensive review on different aspects of autonomous mobile robots including localization methods, obstacle avoidance, sensor types, SLAM, etc. We also tried to highlight the role of DL in reinforcing obstacle avoidance and SLAM approaches. The common simulation tools and their properties were reviewed. Additionally, publicly available datasets (containing real sensor data) for implementing SLAM without having an actual robot were introduced. Finally, the possible directions for future research were pointed out.

ACKNOWLEDGMENTS

REFERENCES

[1] [n. d.]. url = https://www.ros.org/. ([n. d.]). accessed February 27, 2022.

[2] [n. d.]. url = https://en.wikipedia.org/wiki/Global_Positioning_System. ([n. d.]). accessed February 27, 2022.

[3] [n. d.]. url = https://developer.nvidia.com/buy-jetson. ([n. d.]). accessed February 27, 2022.

[4] [n. d.]. url = https://learn.e.ros4.pro/en/robotic_simulators/comparison/. ([n. d.]). accessed February 27, 2022.

[5] [n. d.]. url = https://www.ode.org/. ([n. d.]). accessed February 27, 2022.

[6] [n. d.]. url = https://docs.fetchrobotics.com/gazebo.html. ([n. d.]). accessed February 27, 2022.

[7] [n. d.]. url = http://wiki.ros.org/Robots/PR2#Hardware_Drivers_and_Simulation. ([n. d.]). accessed February 27, 2022.

[8] [n. d.]. url = https://github.com/IFL-CAMP/iiwa_stack. ([n. d.]). accessed February 27, 2022.

[9] [n. d.]. url = http://wiki.ros.org/nao. ([n. d.]). accessed February 27, 2022.

[10] [n. d.]. url = https://gazebosim.org/home. ([n. d.]). accessed February 27, 2022.

[11] [n. d.]. url = https://cyberbotics.com/. ([n. d.]). accessed February 27, 2022.

[12] [n. d.]. url = https://www.coppeliarobotics.com/. ([n. d.]). accessed February 27, 2022.

[13] [n. d.]. url = https://github.com/ivipsourcecode/dxslam. ([n. d.]). accessed February 27, 2022.

[14] [n. d.]. url = https://www.eth3d.net/slam_overview. ([n. d.]). accessed February 27, 2022.

[15] [n. d.]. url = https://github.com/ETH3D. ([n. d.]). accessed February 27, 2022.

[16] [n. d.]. url = sites.google.com/view/recon-robot. ([n. d.]). accessed February 27, 2022.

[17] [n. d.]. 2D Cartographer Backpack - Deutsches Museum. url = https://google-cartographer-ros.readthedocs.io/en/latest/data.html. ([n. d.]). accessed February 27, 2022.

[18] [n. d.]. DARPA Grand Challenge (2004). url = https://en.wikipedia.org/wiki/DARPA_Grand_Challenge_(2004). ([n. d.]). accessed February 27, 2022.

[19] [n. d.]. DARPA Grand Challenge (2007). url = https://en.wikipedia.org/wiki/DARPA_Grand_Challenge_(2007). ([n. d.]). accessed February 27, 2022.

[20] [n. d.]. DVLs - Doppler Velocity Logs. url = https://geo-matching.com/dvls-doppler-velocity-logs. ([n. d.]). accessed February 27, 2022.

[21] [n. d.]. Industrial robots from KUKA. url = https://www.kuka.com/en-de/products/robot-systems/industrial-robots. ([n. d.]). accessed February 27, 2022.

[22] [n. d.]. Kalibr. url = https://github.com/ethz-asl/kalibr. ([n. d.]). accessed February 27, 2022.

[23] [n. d.]. Luna - Exploring the Moon. url = http://www.orbitalfocus.uk/Diaries/Luna/Luna17.php. ([n. d.]). accessed February 27, 2022.

[24] [n. d.]. Parrot drones. url = https://www.parrot.com/us/drones. ([n. d.]). accessed February 27, 2022.

[25] [n. d.]. Prof. Schmidhuber's highlights of robot car history. url = https://people.idsia.ch/ juergen/robotcars.html. ([n. d.]). accessed February 27, 2022.

[26] [n. d.]. RobotEye REV25-ST. url = https://www.ocularrobotics.com/products/camera-pointing-devices/rev25-st/. ([n. d.]). accessed July 14, 2022.

[27] [n. d.]. Rotors_simulator. url = https://github.com/ethz-asl/rotors_simulator. ([n. d.]). accessed February 27, 2022.

[28] Nemra Abdelkrim, Nabil Aouf, Antonios Tsourdos, and Brian White. 2008. Robust nonlinear filtering for INS/GPS UAV localization. In *2008 16th Mediterranean Conference on Control and Automation*. 695–702. https://doi.org/10.1109/MED.2008.4602149

[29] Nuno Abreu and Aníbal Matos. 2014. Minehunting mission planning for autonomous underwater systems using evolutionary algorithms. *Unmanned Systems* 2, 04 (2014), 323–349.

[30] Evan Ackerman. 2017. Carnegie Mellon Solves 12-Year-Old DARPA Grand Challenge Mystery. url = https://spectrum.ieee.org/cmu-solves-12-year-old-darpa-grand-challenge-mystery. (2017). accessed February 27, 2022.

[31] James Lowell Adams. 1961. *Remote control with long transmission delays*. Stanford University.

[32] Mary B Alatise and Gerhard P Hancke. 2020. A review on challenges of autonomous mobile robot and sensor fusion methods. *IEEE Access* 8 (2020), 39830–39846.

[33] Bashar Alsadik and Samer Karam. 2021. The simultaneous localization and mapping (SLAM)-An overview. *Surveying and Geospatial Engineering Journal* 2, 01 (2021), 01–12.

[34] Javier Antich, Alberto Ortiz, and Javier Mínguez. 2009. A bug-inspired algorithm for efficient anytime path planning. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 5407–5413.

[35] Javier Antich, Alberto Ortiz, and J Minguez. 2009. Bug2+: Details and formal proofs. *Technical Report A-1-2009, University of the Balearic Islands* (2009).

[36] Richard Aplin, Ben Cazzolato, Steven Grainger, and Chris Madden. 2010. *1025: MAGIC 2010 Multi Autonomous Ground International Challenge. Volume I*. Technical Report. ADELAIDE UNIV (AUSTRALIA).

[37] Saba Arshad and Gon-Woo Kim. 2021. Role of deep learning in loop closure detection for visual and lidar slam: A survey. *Sensors* 21, 4 (2021), 1243.

[38] M Sanjeev Arulampalam, Simon Maskell, Neil Gordon, and Tim Clapp. 2002. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Transactions on signal processing* 50, 2 (2002), 174–188.

[39] Farah Bader and Shahin Rahimifard. 2020. A methodology for the selection of industrial robots in food handling. *Innovative Food Science & Emerging Technologies* 64 (2020), 102379.

[40] Hernán Badino, Daniel Huber, and Takeo Kanade. 2011. Visual topometric localization. In *2011 IEEE Intelligent vehicles symposium (IV)*. IEEE, 794–799.

[41] James Andrew Bagnell, David Bradley, David Silver, Boris Sofman, and Anthony Stentz. 2010. Learning for autonomous navigation. *IEEE Robotics & Automation Magazine* 17, 2 (2010), 74–84.

[42] Max Bajracharya, Mark W Maimone, and Daniel Helmick. 2008. Autonomy for mars rovers: Past, present, and future. *Computer* 41, 12 (2008), 44–50.

[43] Jianhua Bao, Daoliang Li, Xi Qiao, and Thomas Rauschenbach. 2020. Integrated navigation for autonomous underwater vehicles in aquaculture: A review. *Information processing in agriculture* 7, 1 (2020), 139–151.

[44] Lavinia Barducci, Giovanni Pittiglio, Joseph C Norton, Keith L Obstein, and Pietro Valdastri. 2019. Adaptive dynamic control for magnetically actuated medical robots. *IEEE robotics and automation letters* 4, 4 (2019), 3633–3640.

[45] John E Bares and David S Wettergreen. 1999. Dante II: Technical description, results, and lessons learned. *The International Journal of robotics research* 18, 7 (1999), 621–649.

[46] Simon P Barr, Peter F Swaszek, Richard J Hartnett, and Gregory W Johnson. 2013. Performance of Multi-Beacon DGPS. In *Proceedings of the International Technical Meeting of The Institute of Navigation*. ION, 359–373.

[47] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. 2006. SURF: Speeded Up Robust Features. In *Computer Vision – ECCV 2006*, Alevs Leonardis, Horst Bischof, and Axel Pinz (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 404–417.

[48] Alexandre Bernardino and José Santos-Victor. 1998. Visual behaviours for binocular tracking. *Robotics and Autonomous Systems* 25, 3-4 (1998), 137–146.

[49] Jose-Luis Blanco-Claraco, Francisco-Angel Moreno-Duenas, and Javier Gonzalez-Jimenez. 2014. The Malaga urban dataset: High-rate stereo and LiDAR in a realistic urban scenario. *The International Journal of Robotics Research* 33, 2 (2014), 207–214.

[50] Francisco Bonin-Font, Alberto Ortiz, and Gabriel Oliver. 2008. Visual navigation for mobile robots: A survey. *Journal of intelligent and robotic systems* 53, 3 (2008), 263–296.

[51] Johann Borenstein and Yoram Koren. 1989. Real-time obstacle avoidance for fast mobile robots. *IEEE Transactions on systems, Man, and Cybernetics* 19, 5 (1989), 1179–1187.

[52] Johann Borenstein, Yoram Koren, et al. 1991. The vector field histogram-fast obstacle avoidance for mobile robots. *IEEE transactions on robotics and automation* 7, 3 (1991), 278–288.

[53] Josep Bosch, Nuno Gracias, Pere Ridao, Klemen Istenič, and David Ribas. 2016. Close-range tracking of underwater vehicles using light beacons. *Sensors* 16, 4 (2016), 429.

[54] David M Bradley and J Andrew Bagnell. 2010. *Domain adaptation for mobile robot navigation.* Technical Report. Number CMU-RI-TR.

[55] Deborah Braid, Alberto Broggi, and Gary Schmiedel. 2006. The TerraMax autonomous vehicle. *Journal of Field Robotics* 23, 9 (2006), 693–708.

[56] Xiaobo Cai, Houtse Hsu, Hua Chai, Leixiang Ding, and Yong Wang. 2019. Multi-antenna GNSS and INS integrated position and attitude determination without base station for land vehicles. *The Journal of Navigation* 72, 2 (2019), 342–358.

[57] O Calvo, A Sousa, J Bibiloni, H Curti, G Acosta, and A Rozenfeld. 2009. Low-cost autonomous underwater vehicle for underwater acoustic inspections. *Journal of Maritime Research* 6, 2 (2009), 37–52.

[58] Carlos Campos, Richard Elvira, Juan J Gómez Rodríguez, José MM Montiel, and Juan D Tardós. 2021. Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam. *IEEE Transactions on Robotics* 37, 6 (2021), 1874–1890.

[59] Yuchen Cao, Lan Hu, and Laurent Kneip. 2020. Representations and Benchmarking of Modern Visual SLAM Systems. *Sensors* 20, 9 (2020), 2572.

[60] Álvaro Cebrian, Andrea Bellés, Carla Martin, Aitor Salas, Javier Fernández, Javier Arribas, Jordi Vilà-Valls, and Monica Navarro. 2019. Low-cost hybrid GNSS/UWB/INS integration for seamless indoor/outdoor UAV navigation. In *Proceedings of the 32nd International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2019)*. 2680–2691.

[61] Hee-Won Chae, Ji-Hoon Choi, and Jae-Bok Song. 2020. Robust and autonomous stereo visual-inertial navigation for non-holonomic mobile robots. *IEEE Transactions on Vehicular Technology* 69, 9 (2020), 9613–9623.

[62] Lee Champeny-Bares, Syd Coppersmith, and Kevin Dowling. 1991. *The terregator mobile robot.* Technical Report. CARNEGIE-MELLON UNIV PITTSBURGH PA ROBOTICS INST.

[63] Chao Chen, Xinhao Liu, Xuchu Xu, Yiming Li, Li Ding, Ruoyu Wang, and Chen Feng. 2022. Self-Supervised Visual Place Recognition by Mining Temporal and Feature Neighborhoods. *arXiv preprint arXiv:2208.09315* (2022).

[64] Weifeng Chen, Guangtao Shang, Aihong Ji, Chengjun Zhou, Xiyang Wang, Chonghui Xu, Zhenxiong Li, and Kai Hu. 2022. An Overview on Visual SLAM: From Tradition to Semantic. *Remote Sensing* 14, 13 (2022), 3010.

[65] Zhichao Chen and Stanley T Birchfield. 2009. Qualitative vision-based path following. *IEEE Transactions on Robotics* 25, 3 (2009), 749–754.

[66] Xiuquan Cheng, Shaobo Zhang, Sizhu Cheng, Qinxiang Xia, and Junhao Zhang. 2022. Path-Following and Obstacle Avoidance Control of Non-holonomic Wheeled Mobile Robot Based on Deep Reinforcement Learning. *Applied Sciences* 12, 14 (2022), 6874.

[67] Howie Choset, Kevin M Lynch, Seth Hutchinson, George A Kantor, and Wolfram Burgard. 2005. *Principles of robot motion: theory, algorithms, and implementations.* MIT press.

[68] Robert Codd-Downey, Michael Jenkin, and Katherine Allison. 2017. Milton: An open hardware underwater autonomous vehicle. In *2017 IEEE International Conference on Information and Automation (ICIA)*. IEEE, 30–34.

[69] compiled from Dudek and Computational Principles of Mobile Robotics Jenkin. [n. d.]. CS W4733 NOTES - Differential Drive Robots. url = https://www.cs.columbia.edu/ allen/F19/NOTES/icckinematics.pdf. ([n. d.]). accessed February 27, 2022.

[70] Alejo Concha and Javier Civera. 2014. Using superpixels in monocular SLAM. In *2014 IEEE international conference on robotics and automation (ICRA)*. IEEE, 365–372.

[71] Giuseppe Cotugno, Luigi D'Alfonso, Walter Lucia, Pietro Muraca, and Paolo Pugliese. 2013. Extended and Unscented Kalman Filters for mobile robot localization and environment reconstruction. In *21st Mediterranean Conference on Control and Automation*. IEEE, 19–26.

[72] R Craig Coulter. 1992. *Implementation of the pure pursuit path tracking algorithm.* Technical Report. Carnegie-Mellon UNIV Pittsburgh PA Robotics INST.

[73] R Craig Coulter and George G Mueller. 1994. *A Reconfiguration Study for the NavLab II Mobile Robot.* Technical Report. CARNEGIE-MELLON UNIV PITTSBURGH PA ROBOTICS INST.

[74] Mark Cummins and Paul Newman. 2008. FAB-MAP: Probabilistic localization and mapping in the space of appearance. *The International Journal of Robotics Research* 27, 6 (2008), 647–665.

[75] André Luís da Silva and José Jaime da Cruz. 2016. Fuzzy adaptive extended Kalman filter for UAV INS/GPS data fusion. *Journal of the Brazilian Society of Mechanical Sciences and Engineering* 38, 6 (2016), 1671–1688.

[76] Luigi D'Alfonso, Antonio Grano, Pietro Muraca, and Paolo Pugliese. 2013. A polynomial based SLAM algorithm for mobile robots using ultrasonic sensors-Experimental results. In *2013 16th International Conference on Advanced Robotics (ICAR)*. IEEE, 1–6.

[77] Andrew J Davison. 2003. Real-time simultaneous localisation and mapping with a single camera. In *Computer Vision, IEEE International Conference on*, Vol. 3. IEEE Computer Society, 1403–1403.

[78] Andrew J Davison, Yolanda Gonzalez Cid, and Nobuyuki Kita. 2004. Real-time 3D SLAM with wide-angle vision. *IFAC Proceedings Volumes* 37, 8 (2004), 868–873.

[79] Andrew J Davison, Ian D Reid, Nicholas D Molton, and Olivier Stasse. 2007. MonoSLAM: Real-time single camera SLAM. *IEEE transactions on pattern analysis and machine intelligence* 29, 6 (2007), 1052–1067.

[80] Xinke Deng, Zixu Zhang, Avishai Sintov, Jing Huang, and Timothy Bretl. 2018. Feature-constrained active visual SLAM for mobile robot navigation. In *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 7233–7238.

[81] Guilherme N DeSouza and Avinash C Kak. 2002. Vision for mobile robot navigation: A survey. *IEEE transactions on pattern analysis and machine intelligence* 24, 2 (2002), 237–267.

[82] Luca Di Giammarino, Leonardo Brizi, Tiziano Guadagnino, Cyrill Stachniss, and Giorgio Grisetti. 2022. MD-SLAM: Multi-cue Direct SLAM. *arXiv preprint arXiv:2203.13237* (2022).

[83] Antonio Di Lallo, Robin Murphy, Axel Krieger, Junxi Zhu, Russell H Taylor, and Hao Su. 2021. Medical robots for infectious diseases: lessons and challenges from the COVID-19 pandemic. *IEEE Robotics & Automation Magazine* 28, 1 (2021), 18–27.

[84] Ernst Dieter Dickmanns. 2007. *Dynamic vision for perception and control of motion.* Springer Science & Business Media.

[85] Arnaud Doucet, Nando De Freitas, Neil James Gordon, et al. 2001. *Sequential Monte Carlo methods in practice.* Vol. 1. Springer.

[86] Andrew P Duchon, Leslie Pack Kaelbling, and William H Warren. 1998. Ecological robotics. *Adaptive Behavior* 6, 3-4 (1998), 473–507.

[87] Ethan Eade and Tom Drummond. 2009. Edge landmarks in monocular SLAM. *Image and Vision Computing* 27, 5 (2009), 588–596.

[88] Les Earnest. 2012. Stanford Cart. url = https://web.stanford.edu/ learnest/sail/oldcart.html. (2012). accessed February 27, 2022.

[89] Yael Edan. 1995. Design of an autonomous agricultural robot. *Applied Intelligence* 5, 1 (1995), 41–50.

[90] Erik Einhorn, Christof Schröter, and Horst-Michael Gross. 2010. Building 2d and 3d adaptiveresolution occupancy maps using nd-trees. *Proceeding 55th Int. Scientific Colloquiium, Ilmenau, Germany. Verlag ISLE* (2010), 306–311.

[91] Alberto Elfes. 1989. Using occupancy grids for mobile robot perception and navigation. *Computer* 22, 6 (1989), 46–57.

[92] Jos Elfring, Elena Torta, and René van de Molengraft. 2021. Particle filters: A hands-on tutorial. *Sensors* 21, 2 (2021), 438.

[93] Jakob Engel, Thomas Schöps, and Daniel Cremers. 2014. LSD-SLAM: Large-scale direct monocular SLAM. In *European conference on computer vision.* Springer, 834–849.

[94] Jakob Engel, Jörg Stückler, and Daniel Cremers. 2015. Large-scale direct SLAM with stereo cameras. In *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 1935–1942.

[95] Maurice F Fallon, Michael Kaess, Hordur Johannsson, and John J Leonard. 2011. Efficient AUV navigation fusing acoustic ranging and side-scan sonar. In *2011 IEEE International Conference on Robotics and Automation.* IEEE, 2398–2405.

[96] Tingxiang Fan, Xinjing Cheng, Jia Pan, Dinesh Manocha, and Ruigang Yang. 2018. Crowdmove: Autonomous mapless navigation in crowded scenarios. *arXiv preprint arXiv:1807.07870* (2018).

[97] Maxime Ferrera, Alexandre Eudes, Julien Moras, Martial Sanfourche, and Guy Le Besnerais. 2021. OV2SLAM: A Fully Online and Versatile Visual SLAM for Real-Time Applications. *IEEE Robotics and Automation Letters* 6, 2 (2021), 1399–1406.

[98] Alex Fisher, Ricardo Cannizzaro, Madeleine Cochrane, Chatura Nagahawatte, and Jennifer L Palmer. 2021. ColMap: A memory-efficient occupancy grid mapping framework. *Robotics and Autonomous Systems* 142 (2021), 103755.

[99] Mario M Foglia and Giulio Reina. 2006. Agricultural robot for radicchio harvesting. *Journal of Field Robotics* 23, 6-7 (2006), 363–377.

[100] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. 1997. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine* 4, 1 (1997), 23–33.

[101] Giuseppe Fragapane, Rene De Koster, Fabio Sgarbossa, and Jan Ola Strandhagen. 2021. Planning and control of autonomous mobile robots for intralogistics: Literature review and research agenda. *European Journal of Operational Research* 294, 2 (2021), 405–426.

[102] Paul Furgale, Timothy D Barfoot, and Gabe Sibley. 2012. Continuous-time batch estimation using temporal basis functions. In *2012 IEEE International Conference on Robotics and Automation.* IEEE, 2088–2095.

[103] Paul Furgale, Joern Rehder, and Roland Siegwart. 2013. Unified temporal and spatial calibration for multi-sensor systems. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems.* IEEE, 1280–1286.

[104] Fadri Furrer, Michael Burri, Markus Achtelik, and Roland Siegwart. 2016. *Robot Operating System (ROS): The Complete Reference (Volume 1).* Springer International Publishing, Cham, Chapter RotorS—A Modular Gazebo MAV Simulator Framework, 595–625. https://doi.org/10.1007/978-3-319-26054-9_23

[105] Yoav Gabriely and Elon Rimon. 2008. Cbug: A quadratically competitive mobile robot navigation algorithm. *IEEE Transactions on Robotics* 24, 6 (2008), 1451–1457.

[106] Emilio Garcia-Fidalgo and Alberto Ortiz. 2018. ibow-lcd: An appearance-based loop-closure detection approach using incremental bags of binary words. *IEEE Robotics and Automation Letters* 3, 4 (2018), 3051–3057.

[107] Philippe Gaussier, Cédric Joulain, Stéphane Zrehen, Jean-Paul Banquet, and Arnaud Revel. 1997. Visual navigation in an open environment without map. In *Proceedings of the 1997 IEEE/RSJ International Conference on Intelligent Robot and Systems. Innovative Robotics for Real-World Applications. IROS'97*, Vol. 2. IEEE, 545–550.

[108] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. 2013. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research* 32, 11 (2013), 1231–1237.

[109] Patrick Geneva, Kevin Eckenhoff, Woosik Lee, Yulin Yang, and Guoquan Huang. 2020. Openvins: A research platform for visual-inertial estimation. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 4666–4672.

[110] Reza Ghaffarivardavagh, Sayed Saad Afzal, Osvy Rodriguez, and Fadel Adib. 2020. Underwater backscatter localization: Toward a battery-free underwater GPS. In *Proceedings of the 19th ACM Workshop on Hot Topics in Networks*. 125–131.

[111] Walter R Gilks and Carlo Berzuini. 2001. Following a moving target—Monte Carlo inference for dynamic Bayesian models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 63, 1 (2001), 127–146.

[112] Josué González-García, Alfonso Gómez-Espinosa, Enrique Cuan-Urquizo, Luis Govinda García-Valdovinos, Tomás Salgado-Jiménez, and Jesus Arturo Escobedo Cabello. 2020. Autonomous underwater vehicles: Localization, navigation, and communication for collaborative missions. *Applied sciences* 10, 4 (2020), 1256.

[113] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. *Advances in neural information processing systems* 27 (2014).

[114] Neil J Gordon, David J Salmond, and Adrian FM Smith. 1993. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. In *IEE Proceedings F-radar and signal processing*, Vol. 140. IET, 107–113.

[115] Giorgio Grisetti, Rainer Kümmerle, Cyrill Stachniss, and Wolfram Burgard. 2010. A tutorial on graph-based SLAM. *IEEE Intelligent Transportation Systems Magazine* 2, 4 (2010), 31–43.

[116] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. 2007. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE transactions on Robotics* 23, 1 (2007), 34–46.

[117] R. Groß, M. Bonani, F. Mondada, and M. Dorigo. 2006. Autonomous Self-assembly in a Swarm-bot. In *Proc. of the 3rd Int. Symp. on Autonomous Minirobots for Research and Edutainment (AMiRE 2005)*, K. Murase, K. Sekiyama, N. Kubota, T. Naniwa, and J. Sitte (Eds.). Springer, Berlin, Germany, 314–322.

[118] Shiyi Guo, Zheng Rong, Shuo Wang, and Yihong Wu. 2022. A LiDAR SLAM with PCA-based Feature Extraction and Two-stage Matching. *IEEE Transactions on Instrumentation and Measurement* (2022).

[119] Fredrik Gustafsson, Fredrik Gunnarsson, Niclas Bergman, Urban Forssell, Jonas Jansson, Rickard Karlsson, and P-J Nordlund. 2002. Particle filters for positioning, navigation, and tracking. *IEEE Transactions on signal processing* 50, 2 (2002), 425–437.

[120] Mehmet Serdar Güzel. 2013. Autonomous vehicle navigation using vision and mapless strategies: a survey. *Advances in Mechanical Engineering* 5 (2013), 234747.

[121] Roberto Guzmán, Jean-Bernard Hayet, and Reinhard Klette. 2015. Towards ubiquitous autonomous driving: The CCSAD dataset. In *International Conference on Computer Analysis of Images and Patterns*. Springer, 582–593.

[122] QP Ha, L Yen, and C Balaguer. 2019. Robotic autonomous systems for earthmoving in military applications. *Automation in Construction* 107 (2019), 102934.

[123] Dirk Hahnel, Wolfram Burgard, Dieter Fox, and Sebastian Thrun. 2003. An efficient FastSLAM algorithm for generating maps of large-scale cyclic environments from raw laser range measurements. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*, Vol. 1. IEEE, 206–211.

[124] Dirk Hähnel, Sebastian Thrun, Ben Wegbreit, and Wolfram Burgard. 2005. Towards lazy data association in SLAM. In *Robotics Research. The Eleventh International Symposium*. Springer, 421–431.

[125] Peter E Hart, Nils J Nilsson, and Bertram Raphael. 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics* 4, 2 (1968), 100–107.

[126] Bo He, Hongjin Zhang, Chao Li, Shujing Zhang, Yan Liang, and Tianhong Yan. 2011. Autonomous navigation for autonomous underwater vehicles based on information filters and active sensing. *Sensors* 11, 11 (2011), 10958–10980.

[127] Oyvind Hegrenaes and Einar Berglund. 2009. Doppler water-track aided inertial navigation for autonomous underwater vehicle. In *OCEANS 2009-EUROPE*. IEEE, 1–10.

[128] Oyvind Hegrenas, Einar Berglund, and Oddvar Hallingstad. 2008. Model-aided inertial navigation for underwater vehicles. In *2008 IEEE International Conference on Robotics and Automation*. IEEE, 1069–1076.

[129] Lionel Heng, Dominik Honegger, Gim Hee Lee, Lorenz Meier, Petri Tanskanen, Friedrich Fraundorfer, and Marc Pollefeys. 2014. Autonomous visual mapping and exploration with a micro aerial vehicle. *Journal of Field Robotics* 31, 4 (2014), 654–675.

[130] SL Hess, RM Henry, Conway B Leovy, JA Ryan, and James E Tillman. 1977. Meteorological results from the surface of Mars: Viking 1 and 2. *Journal of Geophysical Research* 82, 28 (1977), 4559–4574.

[131] Wolfgang Hess, Damon Kohler, Holger Rapp, and Daniel Andor. 2016. Real-time loop closure in 2D LIDAR SLAM. In *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 1271–1278.

[132] Ziyang Hong, Yvan Petillot, Andrew Wallace, and Sen Wang. 2022. RadarSLAM: A robust simultaneous localization and mapping system for all weather conditions. *The International Journal of Robotics Research* (2022), 02783649221080483.

[133] A Hornung, KM Wurm, M Bennewitz, C Stachniss, and W Burgard. 2013. An efficient probabilistic 3d mapping framework based on octrees armin hornung. *Autonomous Robots Journal. Springer* (2013).

[134] Bing-Qiang Huang, Guang-Yi Cao, and Min Guo. 2005. Reinforcement learning neural network to the problem of autonomous mobile robot obstacle avoidance. In *2005 International conference on machine learning and cybernetics*, Vol. 1. IEEE, 85–89.

[135] Hoai Nam Huynh, Hamed Assadi, Edouard Rivière-Lorphèvre, Olivier Verlinden, and Keivan Ahmadi. 2020. Modelling the dynamics of industrial robots for milling operations. *Robotics and Computer-Integrated Manufacturing* 61 (2020), 101852.

[136] Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. 2013. Dynamical movement primitives: learning attractor models for motor behaviors. *Neural computation* 25, 2 (2013), 328–373.

[137] Anantha Sai Hari Haran V Injarapu and Suresh Kumar Gawre. 2017. A survey of autonomous mobile robot path planning approaches. In *2017 International conference on recent innovations in signal processing and embedded systems (RISE)*. IEEE, 624–628.

[138] Marco Jacobi and Divas Karimanzira. 2013. Underwater pipeline and cable inspection using autonomous underwater vehicles. In *2013 MTS/IEEE OCEANS - Bergen*. 1–6. https://doi.org/10.1109/OCEANS-Bergen.2013.6608089

[139] Xu Jian, Chen Xiaoyuan, Song Xiaoping, and Li Hang. 2015. Target recognition and location based on binocular vision system of UUV. In *2015 34th Chinese Control Conference (CCC)*. IEEE, 3959–3963.

[140] Minghao Jiang, Yang Chen, Wenlei Zheng, Huaiyu Wu, and Lei Cheng. 2016. Mobile robot path planning based on dynamic movement primitives. In *2016 IEEE International Conference on Information and Automation (ICIA)*. IEEE, 980–985.

[141] Eagle Jones, Brian Fulkerson, Emilio Frazzoli, Deepak Kumar, Robert Walters, Jim Radford, and Richard Mason. 2006. Autonomous off-road driving in the DARPA Grand Challenge. In *Proceedings of IEEE/ION PLANS 2006*. 366–371.

[142] Simon J Julier and Jeffrey K Uhlmann. 1997. New extension of the Kalman filter to nonlinear systems. In *Signal processing, sensor fusion, and target recognition VI*, Vol. 3068. International Society for Optics and Photonics, 182–193.

[143] Sungyoung Jung, Jungmin Kim, and Sungshin Kim. 2009. Simultaneous localization and mapping of a wheel-based autonomous vehicle with ultrasonic sensors. *Artificial Life and Robotics* 14, 2 (2009), 186–190.

[144] Rudolph Emil Kalman. 1960. A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME–Journal of Basic Engineering* 82, Series D (1960), 35–45.

[145] Ishay Kamon, Elon Rimon, and Ehud Rivlin. 1998. Tangentbug: A range-sensor-based navigation algorithm. *The International Journal of Robotics Research* 17, 9 (1998), 934–953.

[146] Mohamed Kara Mohamed, Sourav Patra, and Alexander Lanzon. 2011. Designing simple indoor navigation system for UAVs. In *2011 19th Mediterranean Conference on Control & Automation (MED)*. 1223–1228. https://doi.org/10.1109/MED.2011.5983054

[147] S Karam, V Lehtola, and G Vosselman. 2019. INTEGRATING A LOW-COST MEMS IMU INTO A LASER-BASED SLAM FOR INDOOR MOBILE MAPPING. *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences* (2019).

[148] Simon Kassel. 1971. *Lunokhod-1 Soviet lunar surface vehicle.* Technical Report. RAND CORP SANTA MONICA CA.

[149] Mohamad Mahdi Kassir, Maziar Palhang, and Mohammad Reza Ahmadzadeh. 2020. Qualitative vision-based navigation based on sloped funnel lane concept. *Intelligent Service Robotics* 13, 2 (2020), 235–250.

[150] DM Keirsey, JS Mitchell, DW Payton, DY Tseng, and VS Wong. 1987. *Autonomous Land Vehicle (ALV) Planning and Navigation System.* Technical Report. HUGHES RESEARCH LABS MALIBU CA.

[151] W.H. Key. 2000. Side scan sonar technology. In *OCEANS 2000 MTS/IEEE Conference and Exhibition. Conference Proceedings (Cat. No.00CH37158)*, Vol. 2. 1029–1033 vol.2. https://doi.org/10.1109/OCEANS.2000.881735

[152] Sheraz Khan, Athanasios Dometios, Chris Verginis, Costas Tzafestas, Dirk Wollherr, and Martin Buss. 2014. RMAP: a rectangular cuboid approximation framework for 3D environment mapping. *Autonomous Robots* 37, 3 (2014), 261–277.

[153] Oussama Khatib. 1986. Real-time obstacle avoidance for manipulators and mobile robots. In *Autonomous robot vehicles*. Springer, 396–404.

[154] Dongsung Kim and Ramakant Nevatia. 1998. Recognition and localization of generic objects for indoor navigation using functionality. *Image and Vision Computing* 16, 11 (1998), 729–743.

[155] Dongsung Kim and Ramakant Nevatia. 1999. Symbolic navigation with a generic map. *Autonomous Robots* 6, 1 (1999), 69–88.

[156] Sungbok Kim and Hyunbin Kim. 2012. Simple and complex obstacle detection using an overlapped ultrasonic sensor ring. In *2012 12th International Conference on Control, Automation and Systems*. IEEE, 2148–2152.

[157] Tae-eun Kim, Lokukaluge Prasad Perera, Magne-Petter Sollid, Bjørn-Morten Batalden, and Are Kristoffer Sydnes. 2022. Safety challenges related to autonomous ships in mixed navigational environments. *WMU Journal of Maritime Affairs* (2022), 1–18.

[158] James C Kinsey, Ryan M Eustice, and Louis L Whitcomb. 2006. A survey of underwater vehicle navigation: Recent advances and new challenges. In *IFAC conference of manoeuvering and control of marine craft*, Vol. 88. Lisbon, 1–12.

[159] Georg Klein and David Murray. 2007. Parallel Tracking and Mapping for Small AR Workspaces. In *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*. 225–234. https://doi.org/10.1109/ISMAR.2007.4538852

[160] Georg Klein and David Murray. 2007. Parallel tracking and mapping for small AR workspaces. In *2007 6th IEEE and ACM international symposium on mixed and augmented reality*. IEEE, 225–234.

[161] Georg Klein and David Murray. 2008. Improving the agility of keyframe-based SLAM. In *European conference on computer vision*. Springer, 802–815.

[162] Georg Klein and David Murray. 2009. Parallel tracking and mapping on a camera phone. In *2009 8th IEEE International Symposium on Mixed and Augmented Reality*. IEEE, 83–86.

[163] Sven Koenig and Maxim Likhachev. 2005. Fast replanning for navigation in unknown terrain. *IEEE Transactions on Robotics* 21, 3 (2005), 354–363.

[164] Sven Koenig, Maxim Likhachev, and David Furcy. 2004. Lifelong planning A∗. *Artificial Intelligence* 155, 1-2 (2004), 93–146.

[165] Stefan Kohlbrecher, Johannes Meyer, Thorsten Graber, Karen Petersen, Uwe Klingauf, and Oskar von Stryk. 2013. Hector open source modules for autonomous mapping and navigation with rescue robots. In *Robot Soccer World Cup*. Springer, 624–631.

[166] Mehmet Korkmaz, Nihat Yılmaz, and Akif Durdu. 2016. Comparison of the SLAM algorithms: Hangar experiments. In *MATEC Web of Conferences*, Vol. 42. EDP Sciences, 03009.

[167] Philipp Koschorrek, Tommaso Piccini, Per Oberg, Michael Felsberg, Lars Nielsen, and Rudolf Mester. 2013. A multi-sensor traffic scene dataset with omnidirectional video. In *Ground Truth - What is a good dataset? CVPR Workshop 2013*.

[168] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2017. Imagenet classification with deep convolutional neural networks. *Commun. ACM* 60, 6 (2017), 84–90.

[169] Eric Krotkov, John Bares, Lalitesh Katragadda, Reid Simmons, and Red Whittaker. 1994. Lunar rover technology demonstrations with Dante and Ratler. In *JPL, Third International Symposium on Artificial Intelligence, Robotics, and Automation for Space 1994*.

[170] Sharon Laubach and Joel Burdick. 2000. RoverBug: Long range navigation for mars rovers. In *Experimental Robotics VI*. Springer, 339–348.

[171] Sharon L Laubach and Joel W Burdick. 1999. An autonomous sensor-based path-planner for planetary microrovers. In *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No. 99CH36288C)*, Vol. 1. IEEE, 347–354.

[172] John J Leonard, Andrew A Bennett, Christopher M Smith, Hans Jacob, and S Feder. 1998. Autonomous underwater vehicle navigation. In *MIT Marine Robotics Laboratory Technical Memorandum*. Citeseer.

[173] John J Leonard and Hugh F Durrant-Whyte. 1991. Mobile robot localization by tracking geometric beacons. *IEEE Transactions on robotics and Automation* 7, 3 (1991), 376–382.

[174] Sergey Levine, Nolan Wagener, and Pieter Abbeel. 2015. Learning Contact-Rich Manipulation Skills with Guided Policy Search. *CoRR* abs/1501.05611 (2015). arXiv:1501.05611 http://arxiv.org/abs/1501.05611

[175] Jesse Levinson, Michael Montemerlo, and Sebastian Thrun. 2007. Map-based precision vehicle localization in urban environments.. In *Robotics: science and systems*, Vol. 4. Citeseer, 1.

[176] Dongjiang Li, Xuesong Shi, Qiwei Long, Shenghui Liu, Wei Yang, Fangshi Wang, Qi Wei, and Fei Qiao. 2020. DXSLAM: A robust and efficient visual SLAM system with deep features. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 4958–4965.

[177] Jing Li, Erfan Shojaei Barjuei, Gastone Ciuti, Yang Hao, Peisen Zhang, Arianna Menciassi, Qiang Huang, and Paolo Dario. 2018. Magnetically-driven medical robots: An analytical magnetic model for endoscopic capsules design. *Journal of Magnetism and Magnetic Materials* 452 (2018), 278–287.

[178] Kang Li, Can Wang, Sheng Huang, Guoyuan Liang, Xinyu Wu, and Yubin Liao. 2016. Self-positioning for UAV indoor navigation based on 3D laser scanner, UWB and INS. In *2016 IEEE International Conference on Information and Automation (ICIA)*. 498–503. https://doi.org/10.1109/ICInfA.2016.7831874

[179] Mingyang Li and Anastasios I Mourikis. 2013. High-precision, consistent EKF-based visual-inertial odometry. *The International Journal of Robotics Research* 32, 6 (2013), 690–711.

[180] Rongbing Li, Jianye Liu, Ling Zhang, and Yijun Hang. 2014. LIDAR/MEMS IMU integrated navigation (SLAM) method for a small UAV in indoor environments. In *2014 DGON Inertial Sensors and Systems (ISS)*. 1–15. https://doi.org/10.1109/InertialSensors.2014.7049479

[181] Zhang Libin, Yang Qinghua, Bao Guanjun, Wang Yan, Qi Liyong, Gao Feng, and Xu Fang. 2008. Overview of research on agricultural robot in China. *International Journal of Agricultural and Biological Engineering* 1, 1 (2008), 12–21.

[182] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* (2015).

[183] Canglong Liu, Bin Zheng, Chunyang Wang, Yongting Zhao, Shun Fu, and Haochen Li. 2017. CNN-based vision model for obstacle avoidance of mobile robot. In *MATEC Web of Conferences*, Vol. 139. EDP Sciences, 00007.

[184] Jun Liu, Zhaohui Wang, Zheng Peng, Jun-Hong Cui, and Lance Fiondella. 2014. Suave: Swarm underwater autonomous vehicle localization. In *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*. IEEE, 64–72.

[185] Jun S Liu and Rong Chen. 1998. Sequential Monte Carlo methods for dynamic systems. *Journal of the American statistical association* 93, 443 (1998), 1032–1044.

[186] Jorge Lobo, Paulo Lucas, Jorge Dias, and A Traca De Almeida. 1995. Inertial navigation system for mobile land vehicles. In *1995 Proceedings of the IEEE International Symposium on Industrial Electronics*, Vol. 2. IEEE, 843–848.

[187] David G. Lowe. 2004. Distinctive Image Features from Scale-Invariant Keypoints. *Int. J. Comput. Vision* 60, 2 (nov 2004), 91–110. https://doi.org/10.1023/B:VISI.0000029664.99615.94

[188] Feng Lu and Evangelos Milios. 1997. Globally consistent range scan alignment for environment mapping. *Autonomous robots* 4, 4 (1997), 333–349.

[189] Vladimir Lumelsky and Alexander Stepanov. 1986. Dynamic path planning for a mobile automaton with limited information on the environment. *IEEE transactions on Automatic control* 31, 11 (1986), 1058–1063.

[190] Andréa Macario Barros, Maugan Michel, Yoann Moline, Gwenolé Corre, and Frédérick Carrel. 2022. A comprehensive survey of visual slam algorithms. *Robotics* 11, 1 (2022), 24.

[191] Will Maddern, Geoffrey Pascoe, Chris Linegar, and Paul Newman. 2017. 1 year, 1000 km: The Oxford RobotCar dataset. *The International Journal of Robotics Research* 36, 1 (2017), 3–15.

[192] Evgeni Magid and Ehud Rivlin. 2004. CAUTIOUSBUG: A competitive algorithm for sensory-based robot navigation. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, Vol. 3. IEEE, 2757–2762.

[193] Mark Maimone, Andrew Johnson, Yang Cheng, Reg Willson, and Larry Matthies. 2006. Autonomous navigation results from the Mars Exploration Rover (MER) mission. In *Experimental robotics IX*. Springer, 3–13.

[194] András L Majdik, Damiano Verda, Yves Albers-Schoenberg, and Davide Scaramuzza. 2015. Air-ground matching: Appearance-based GPS-denied urban localization of micro aerial vehicles. *Journal of Field Robotics* 32, 7 (2015), 1015–1039.

[195] Flavio BP Malavazi, Remy Guyonneau, Jean-Baptiste Fasquel, Sebastien Lagrange, and Franck Mercier. 2018. LiDAR-only based navigation algorithm for an autonomous agricultural robot. *Computers and electronics in agriculture* 154 (2018), 71–79.

[196] Miquel Massot-Campos and Gabriel Oliver-Codina. 2015. Optical sensors and methods for underwater 3D reconstruction. *Sensors* 15, 12 (2015), 31525–31557.

[197] Michel Maurette. 2003. Mars rover autonomous navigation. *Autonomous Robots* 14, 2 (2003), 199–208.

[198] Jérôme Maye, Paul Furgale, and Roland Siegwart. 2013. Self-supervised calibration for robotic systems. In *2013 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 473–480.

[199] Syed Atif Mehdi, Suleman Mazhar, and Francesco Maurelli. 2021. Autonomous navigation of low cost underwater vehicle for shallow freshwater applications. In *OCEANS 2021: San Diego–Porto*. IEEE, 1–4.

[200] Lorenz Meier, Petri Tanskanen, Lionel Heng, Gim Hee Lee, Friedrich Fraundorfer, and Marc Pollefeys. 2012. PIXHAWK: A micro aerial vehicle design for autonomous flight using onboard computer vision. *Autonomous Robots* 33, 1 (2012), 21–39.

[201] Jean-Arcady Meyer and David Filliat. 2003. Map-based navigation in mobile robots:: Ii. a review of map-learning and path-planning strategies. *Cognitive Systems Research* 4, 4 (2003), 283–317.

[202] Andrew H Mishkin, Jack C Morrison, Tam T Nguyen, Henry W Stone, Brian K Cooper, and Brian H Wilcox. 1998. Experiences with operations and autonomy of the mars pathfinder microrover. In *1998 IEEE aerospace conference proceedings (Cat. No. 98TH8339)*, Vol. 2. IEEE, 337–351.

[203] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *nature* 518, 7540 (2015), 529–533.

[204] Sherif AS Mohamed, Mohammad-Hashem Haghbayan, Tomi Westerlund, Jukka Heikkonen, Hannu Tenhunen, and Juha Plosila. 2019. A survey on odometry for autonomous navigation systems. *IEEE Access* 7 (2019), 97466–97486.

[205] Prases K Mohanty and Dayal R Parhi. 2013. Controlling the motion of an autonomous mobile robot using various techniques: a review. *Journal of Advance Mechanical Engineering* 1, 1 (2013), 24–39.

[206] Nicholas Molton, Andrew J Davison, and Ian Reid. 2004. Locally planar patch features for real-time structure from motion.. In *Bmvc*. 1–10.

[207] M. Monta, N. Kondo, and Y. Shibano. 1995. Agricultural robot in grape production system. In *Proceedings of 1995 IEEE International Conference on Robotics and Automation*, Vol. 3. 2504–2509 vol.3. https://doi.org/10.1109/ROBOT.1995.525635

[208] Michael Montemerlo, Sebastian Thrun, Daphne Koller, Ben Wegbreit, et al. 2002. FastSLAM: A factored solution to the simultaneous localization and mapping problem. *Aaai/iaai* 593598 (2002).

[209] Joshua J Morales and Zaher M Kassas. 2021. Tightly coupled inertial navigation system with signals of opportunity aiding. *IEEE Trans. Aerospace Electron. Systems* 57, 3 (2021), 1930–1948.

[210] Hans Moravec and Alberto Elfes. 1985. High resolution maps from wide angle sonar. In *Proceedings. 1985 IEEE international conference on robotics and automation*, Vol. 2. IEEE, 116–121.

[211] L Moreno, JM Armingol, A De La Escalera, and MA Salichs. 1999. Global integration of ultrasonic sensors information in mobile robot localization. In *Proceedings of the Ninth International Conference on Advanced Robotics*. Tokyo, Japan, 283–288.

[212] Luis Moreno, Jose M Armingol, Santiago Garrido, Arturo De La Escalera, and Miguel A Salichs. 2002. A genetic algorithm for mobile robot localization using ultrasonic sensors. *Journal of Intelligent and Robotic Systems* 34, 2 (2002), 135–154.

[213] Anastasios I Mourikis, Stergios I Roumeliotis, et al. 2007. A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation.. In *ICRA*, Vol. 2. 6.

[214] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. 2015. ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE transactions on robotics* 31, 5 (2015), 1147–1163.

[215] Raul Mur-Artal and Juan D Tardós. 2017. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE transactions on robotics* 33, 5 (2017), 1255–1262.

[216] Raúl Mur-Artal and Juan D Tardós. 2017. Visual-inertial monocular SLAM with map reuse. *IEEE Robotics and Automation Letters* 2, 2 (2017), 796–803.

[217] Christian Musso, Nadia Oudjane, and Francois Le Gland. 2001. Improving regularised particle filters. In *Sequential Monte Carlo methods in practice*. Springer, 247–271.

[218] Saeid Nahavandi, Shady Mohamed, Ibrahim Hossain, Darius Nahavandi, Syed Moshfeq Salaken, Mohammad Rokonuzzaman, Rachael Ayoub, and Robin Smith. 2022. Autonomous Convoying: A Survey on Current Research and Development. *IEEE Access* (2022).

[219] Tayyab Naseer, Michael Ruhnke, Cyrill Stachniss, Luciano Spinello, and Wolfram Burgard. 2015. Robust visual SLAM across seasons. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2529–2535.

[220] Jose Neira and Juan D Tardos. 2001. Data association in stochastic mapping using the joint compatibility test. *IEEE Transactions on robotics and automation* 17, 6 (2001), 890–897.

[221] Abdelkrim Nemra and Nabil Aouf. 2010. Robust INS/GPS sensor fusion for UAV localization using SDRE nonlinear filtering. *IEEE Sensors Journal* 10, 4 (2010), 789–798.

[222] Issa AD Nesnas, Lorraine M Fesq, and Richard A Volpe. 2021. Autonomy for Space Robots: Past, Present, and Future. *Current Robotics Reports* 2, 3 (2021), 251–263.

[223] Jianjun Ni, Liuying Wu, Xinnan Fan, and Simon X Yang. 2016. Bioinspired intelligent algorithm and its applications for mobile robot control: a survey. *Computational intelligence and neuroscience* 2016 (2016).

[224] Md AK Niloy, Anika Shama, Ripon K Chakrabortty, Michael J Ryan, Faisal R Badal, Zinat Tasneem, Md H Ahamed, Sumaya I Moyeen, Sajal K Das, Md F Ali, et al. 2021. Critical design and control issues of indoor autonomous mobile robots: A review. *IEEE Access* 9 (2021), 35338–35370.

[225] Nils J Nilsson et al. 1984. Shakey the robot. (1984).

[226] Lisa Nocks. 2007. *The robot: the life story of a technology.* Greenwood Publishing Group.

[227] Alex Oagana. 2013. A Short History of Mercedes-Benz Autonomous Driving Technology. url = https://www.autoevolution.com/news/a-short-history-of-mercedes-benz-autonomous-driving-technology-68148.html. (2013). accessed February 27, 2022.

[228] International Maritime Organization. 2018. Working Group Report in 100th Session of IMO Maritime Safety Committee for the Regulatory Scoping Exercise for the Use of Maritime Autonomous Surface Ships (MASS). (2018).

[229] Luc Oth, Paul Furgale, Laurent Kneip, and Roland Siegwart. 2013. Rolling shutter camera calibration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1360–1367.

[230] Anish Pandey, Shalini Pandey, and DR Parhi. 2017. Mobile robot navigation and obstacle avoidance techniques: A review. *Int Rob Auto J* 2, 3 (2017), 00022.

[231] Gaurav Pandey, James R. McBride, and Ryan M. Eustice. 2011. Ford campus vision and lidar data set. *International Journal of Robotics Research* 30, 13 (2011), 1543–1552.

[232] Prabin Kumar Panigrahi and Sukant Kishoro Bisoy. 2021. Localization strategies for autonomous mobile robots: A review. *Journal of King Saud University-Computer and Information Sciences* (2021).

[233] Deepak Patil, Munsaf Ansari, Dilisha Tendulkar, Ritesh Bhatlekar, Vijaykumar Naik Pawar, and Shailendra Aswale. 2020. A survey on autonomous military service robot. In *2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE)*. IEEE, 1–7.

[234] Michael K Pitt and Neil Shephard. 1999. Filtering via simulation: Auxiliary particle filters. *Journal of the American statistical association* 94, 446 (1999), 590–599.

[235] Julio A Placed, Jared Strader, Henry Carrillo, Nikolay Atanasov, Vadim Indelman, Luca Carlone, and José A Castellanos. 2022. A Survey on Active Simultaneous Localization and Mapping: State of the Art and New Frontiers. *arXiv preprint arXiv:2207.00254* (2022).

[236] Rahul Shivaji Pol and M Murugan. 2015. A review on indoor human aware autonomous mobile robot navigation through a dynamic environment survey of different path planning algorithm and methods. In *2015 International conference on industrial instrumentation and control (ICIC)*. IEEE, 1339–1344.

[237] DA Pomerleau. 1994. Defense and civilian applications of the alvinn robot driving system. In *Proceedings of 1994 Government Microcircuit Applications Conference*. 358–362.

[238] Hao Qin, May Huang, Jian Cao, and Xing Zhang. 2018. Loop closure detection in SLAM by combining visual CNN features and submaps. In *2018 4th International Conference on Control, Automation and Robotics (ICCAR)*. IEEE, 426–430.

[239] Jiangying Qin, Ke Yang, Ming Li, Jiageng Zhong, and Hanqi Zhang. 2022. Real-Time Positioning and Tracking for Vision-Based Unmanned Underwater Vehicles. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences* 46 (2022), 163–168.

[240] Ganesan Ramalingam and Thomas Reps. 1996. An incremental algorithm for a generalization of the shortest-path problem. *Journal of Algorithms* 21, 2 (1996), 267–305.

[241] Scott Reed, Jon Wood, and Chris Haworth. 2010. The detection and disposal of IED devices within harbor regions using AUVs, smart ROVs and data processing/fusion technology. In *2010 International WaterSide Security Conference*. IEEE, 1–7.

[242] Joern Rehder, Janosch Nikolic, Thomas Schneider, Timo Hinzmann, and Roland Siegwart. 2016. Extending kalibr: Calibrating the extrinsics of multiple IMUs and of individual axes. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 4304–4311.

[243] Hugh Rice, Steven Kelmenson, and Louis Mendelsohn. 2004. Geophysical navigation technologies and applications. In *PLANS 2004. Position Location and Navigation Symposium (IEEE Cat. No. 04CH37556)*. IEEE, 618–624.

[244] Pere Ridao, Marc Carreras, David Ribas, Pedro J Sanz, and Gabriel Oliver. 2015. Intervention AUVs: the next challenge. *Annual Reviews in Control* 40 (2015), 227–241.

[245] Søren Riisgaard and Morten Rufus Blas. 2003. SLAM for Dummies. *A Tutorial Approach to Simultaneous Localization and Mapping* 22, 1-127 (2003), 126.

[246] Edward Rosten and Tom Drummond. 2006. Machine learning for high-speed corner detection. In *European conference on computer vision*. Springer, 430–443.

[247] Cyril Roussillon, Aurelien Gonzalez, Joan Sola, Jean-Marie Codol, Nicolas Mansard, Simon Lacroix, and Michel Devy. 2011. RT-SLAM: a generic and real-time visual SLAM implementation. In *International Conference on Computer Vision Systems*. Springer, 31–40.

[248] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. 2011. ORB: An efficient alternative to SIFT or SURF. In *2011 International conference on computer vision*. Ieee, 2564–2571.

[249] Stuart J. Russell and Peter Norvig. 2009. *Artificial Intelligence: a modern approach* (3 ed.). Pearson.

[250] José Ricardo Sánchez-Ibáñez, Carlos J Pérez-del Pulgar, and Alfonso García-Cerezo. 2021. Path Planning for Autonomous Mobile Robots: A Review. *Sensors* 21, 23 (2021), 7898.

[251] Thomas Sayre-McCord, Winter Guerra, Amado Antonini, Jasper Arneberg, Austin Brown, Guilherme Cavalheiro, Yajun Fang, Alex Gorodetsky, Dave McCoy, Sebastian Quilter, et al. 2018. Visual-inertial navigation algorithm development using photorealistic camera simulation in the loop.

In *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2566–2573.

[252] Stefan Schaal. 2006. Dynamic movement primitives-a framework for motor control in humans and humanoid robotics. In *Adaptive motion of animals and machines*. Springer, 261–280.

[253] H Schäfer and K Berns. 2006. Ravon-an autonomous vehicle for risky intervention and surveillance. In *International Workshop on Robotics for risky intervention and environmental surveillance-RISE*. Citeseer.

[254] Bernt Schiele and James L Crowley. 1996. Object recognition using multidimensional receptive field histograms. In *European Conference on Computer Vision*. Springer, 610–619.

[255] Thomas Schops, Torsten Sattler, and Marc Pollefeys. 2019. Bad slam: Bundle adjusted direct rgb-d slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 134–144.

[256] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[257] Antonio Sgorbissa and Renato Zaccaria. 2012. Planning and obstacle avoidance in mobile robotics. *Robotics and Autonomous Systems* 60, 4 (2012), 628–638.

[258] Dhruv Shah, Benjamin Eysenbach, Nicholas Rhinehart, and Sergey Levine. 2021. Rapid exploration for open-world navigation with latent goal models. *arXiv preprint arXiv:2104.05859* (2021).

[259] Dhruv Shah and Sergey Levine. 2022. ViKiNG: Vision-Based Kilometer-Scale Navigation with Geographic Hints. *arXiv preprint arXiv:2202.11271* (2022).

[260] Radhe Shyam Sharma, Santosh Shukla, Hamad Karki, Amit Shukla, Laxmidhar Behera, and Venkatesh K.S. 2019. DMP Based Trajectory Tracking for a Nonholonomic Mobile Robot With Automatic Goal Adaptation and Obstacle Avoidance. In *2019 International Conference on Robotics and Automation (ICRA)*. 8613–8619. https://doi.org/10.1109/ICRA.2019.8793911

[261] Dong-Won Shin and Yo-Sung Ho. 2018. Loop closure detection in simultaneous localization and mapping using learning based local patch descriptor. *Electronic Imaging* 2018, 17 (2018), 284–1.

[262] Reid Simmons. 1996. The curvature-velocity method for local obstacle avoidance. In *Proceedings of IEEE international conference on robotics and automation*, Vol. 4. IEEE, 3375–3382.

[263] R Simpson, J Cullip, and J Revell. 2011. The cheddar gorge data set. *BAE Systems (Operations) Limited, UK, Tech. Rep.* (2011).

[264] Metin Sitti. 2018. Miniature soft robots—road to the clinic. *Nature Reviews Materials* 3, 6 (2018), 74–75.

[265] Randall Smith, Matthew Self, and Peter Cheeseman. 1990. Estimating uncertain spatial relationships in robotics. In *Autonomous robot vehicles*. Springer, 167–193.

[266] Yang Song and Li-Ta Hsu. 2021. Tightly coupled integrated navigation system via factor graph for UAV indoor localization. *Aerospace Science and Technology* 108 (2021), 106370.

[267] Anthony Stentz. 1997. Optimal and efficient path planning for partially known environments. In *Intelligent unmanned ground vehicles*. Springer, 203–220.

[268] Anthony Stentz et al. 1995. The focussed D* algorithm for real-time replanning. In *IJCAI*, Vol. 95. 1652–1659.

[269] Bruno Steux and Oussama El Hamzaoui. 2010. tinySLAM: A SLAM algorithm in less than 200 lines C-language program. In *2010 11th International Conference on Control Automation Robotics & Vision*. IEEE, 1975–1979.

[270] Hauke Strasdat, J Montiel, and Andrew J Davison. 2010. Scale drift-aware large scale monocular SLAM. *Robotics: Science and Systems VI* 2, 3 (2010), 7.

[271] Elena Stumm, Christopher Mei, Simon Lacroix, Juan Nieto, Marco Hutter, and Roland Siegwart. 2016. Robust visual place recognition with graph kernels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4535–4544.

[272] Yuxiang Sun, Ming Liu, and Max Q-H Meng. 2017. Improving RGB-D SLAM in dynamic environments: A motion removal approach. *Robotics and Autonomous Systems* 89 (2017), 110–122.

[273] Yuxiang Sun, Ming Liu, and Max Q-H Meng. 2018. Motion removal for reliable RGB-D SLAM in dynamic environments. *Robotics and Autonomous Systems* 108 (2018), 115–128.

[274] Kamilah Taylor and Steven M LaValle. 2009. I-Bug: An intensity-based bug algorithm. In *2009 IEEE International Conference on Robotics and Automation*. IEEE, 3981–3986.

[275] S Teller, E Olson, and J Leonard. 2006. Fast iterative optimization of pose graphs with poor inital estimates. In *Proc. of IEEE International Conference on Robotics and Automation*. 2262–2269.

[276] Selim Temizer. 2001. *Optical flow based local navigation*. Master's thesis. Massachusetts Institute of Technology.

[277] Rohan Thakker, Nikhilesh Alatur, David D Fan, Jesus Tordesillas, Michael Paton, Kyohei Otsu, Olivier Toupet, and Ali-akbar Agha-mohammadi. 2020. Autonomous off-road navigation over extreme terrains with perceptually-challenging conditions. In *International Symposium on Experimental Robotics*. Springer, 161–173.

[278] Charles Thorpe, Martial Hebert, Takeo Kanade, and Steven Shafer. 1989. Vision and Navigation for the Carnegie Mellon Navlab. In *High Precision Navigation*, Klaus Linkwitz and Ulrich Hangleiter (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 97–122.

[279] Sebastian Thrun. 1995. Exploration in active learning. *Handbook of Brain Science and Neural Networks* (1995), 381–384.

[280] Sebastian Thrun. 2001. Learning occupancy grids with forward models. In *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No. 01CH37180)*, Vol. 3. IEEE, 1676–1681.

[281] Sebastian Thrun. 2002. Probabilistic robotics. *Commun. ACM* 45, 3 (2002), 52–57.

[282] Sebastian Thrun, Mike Montemerlo, Hendrik Dahlkamp, David Stavens, Andrei Aron, James Diebel, Philip Fong, John Gale, Morgan Halpenny, Gabriel Hoffmann, et al. 2006. Stanley: The robot that won the DARPA Grand Challenge. *Journal of field Robotics* 23, 9 (2006), 661–692.

[283] David Titterton, John L Weston, and John Weston. 2004. *Strapdown inertial navigation technology*. Vol. 17. IET.

[284] Mihnea-Alexandru Tomita, Mubariz Zaffar, Michael Milford, Klaus D. McDonald-Maier, and Shoaib Ehsan. 2021. Sequence-Based Filtering for Visual Route-Based Navigation: Analysing the Benefits, Trade-offs and Design Choices. *CoRR* abs/2103.01994 (2021). arXiv:2103.01994 https://arxiv.org/abs/2103.01994

[285] Rudolph Triebel, Patrick Pfaff, and Wolfram Burgard. 2006. Multi-level surface maps for outdoor terrain mapping and loop closing. In *2006 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2276–2282.

[286] Spyros G Tzafestas. 2018. Mobile robot control and navigation: A global overview. *Journal of Intelligent & Robotic Systems* 91, 1 (2018), 35–58.

[287] Iwan Ulrich and Johann Borenstein. 1998. VFH+: Reliable obstacle avoidance for fast mobile robots. In *Proceedings. 1998 IEEE international conference on robotics and automation (Cat. No. 98CH36146)*, Vol. 2. IEEE, 1572–1577.

[288] Chris Urmson, Joshua Anhalt, Drew Bagnell, Christopher Baker, Robert Bittner, MN Clark, John Dolan, Dave Duggins, Tugrul Galatali, Chris Geyer, et al. 2008. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of field Robotics* 25, 8 (2008), 425–466.

[289] Chris Urmson, J Andrew Bagnell, Christopher Baker, Martial Hebert, Alonzo Kelly, Raj Rajkumar, Paul E Rybski, Sebastian Scherer, Reid Simmons, Sanjiv Singh, et al. 2007. Tartan racing: A multi-modal approach to the darpa urban challenge. (2007).

[290] P Victerpaul, D Saravanan, S Janakiraman, and J Pradeep. 2017. Path planning of autonomous mobile robots: A survey and comparison. *Journal of Advanced Research in Dynamical and Control Systems* 9, 12 (2017), 1535–1565.

[291] Richard Volpe, J Balaram, Timothy Ohm, and Robert Ivlev. 1996. The rocky 7 mars rover prototype. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS'96*, Vol. 3. IEEE, 1558–1564.

[292] Arjan Vonk. 2022. Graph Based LiDAR-Inertial Lo-calization with a Low Power Solid State LiDAR. (2022).

[293] Eric A Wan, Rudolph Van Der Merwe, and Simon Haykin. 2001. The unscented Kalman filter. *Kalman filtering and neural networks* 5, 2007 (2001), 221–280.

[294] Qingshan Wang, Jun Zhang, Yuansheng Liu, and Xinchen Zhang. 2022. High-Precision and Fast LiDAR Odometry and Mapping Algorithm. *Journal of Advanced Computational Intelligence and Intelligent Informatics* 26, 2 (2022), 206–216.

[295] Greg Welch, Gary Bishop, et al. 1995. An introduction to the Kalman filter. (1995).

[296] Patrick Wenzel, Torsten Schön, Laura Leal-Taixé, and Daniel Cremers. 2021. Vision-based mobile robotics obstacle avoidance with deep reinforcement learning. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 14360–14366.

[297] Xianbo Xiang, Bruno Jouvencel, and Olivier Parodi. 2010. Coordinated formation control of multiple autonomous underwater vehicles for pipeline inspection. *International Journal of Advanced Robotic Systems* 7, 1 (2010), 3.

[298] Qingwen Xu, Arturo Gomez Chavez, Heiko Bülow, Andreas Birk, and Sören Schwertfeger. 2019. Improved fourier mellin invariant for robust rotation estimation with omni-cameras. In *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE, 320–324.

[299] Qingwen Xu, Haofei Kuang, Laurent Kneip, and Sören Schwertfeger. 2021. Rethinking the Fourier-Mellin Transform: Multiple Depths in the Camera's View. *Remote Sensing* 13, 5 (2021), 1000.

[300] Brian M. Yamauchi. 2004. PackBot: a versatile platform for military robotics. In *Unmanned Ground Vehicle Technology VI*, Grant R. Gerhart, Chuck M. Shoemaker, and Douglas W. Gage (Eds.), Vol. 5422. International Society for Optics and Photonics, SPIE, 228 – 237. https://doi.org/10.1117/12.538328

[301] Po Yang. 2012. Efficient particle filter algorithm for ultrasonic sensor-based 2D range-only simultaneous localisation and mapping application. *IET Wireless Sensor Systems* 2, 4 (2012), 394–401.

[302] Qifeng Yang, Daokui Qu, Fang Xu, Fengshan Zou, Guojian He, and Mingze Sun. 2018. Mobile robot motion control and autonomous navigation in GPS-denied outdoor environments using 3D laser scanning. *Assembly Automation* (2018).

[303] Hang Yin and Christian Berger. 2017. When to use what data set for your self-driving car algorithm: An overview of publicly available driving datasets. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 1–8.

[304] Khalid Yousif, Alireza Bab-Hadiashar, and Reza Hoseinnezhad. 2015. An overview to visual odometry and visual SLAM: Applications to mobile robotics. *Intelligent Industrial Systems* 1, 4 (2015), 289–311.

[305] Mingxin Yu, Lin Shao, Zhehuan Chen, Tianhao Wu, Qingnan Fan, Kaichun Mo, and Hao Dong. 2021. RoboAssembly: Learning Generalizable Furniture Assembly Policy in a Novel Multi-robot Contact-rich Simulation Environment. *arXiv preprint arXiv:2112.10143* (2021).

[306] Shady Zahran, Adel M. Moussa, Abu B. Sesay, and Naser El-Sheimy. 2019. A New Velocity Meter Based on Hall Effect Sensors for UAV Indoor Navigation. *IEEE Sensors Journal* 19, 8 (2019), 3067–3076. https://doi.org/10.1109/JSEN.2018.2890094

[307] Noor Abdul Khaleq Zghair and Ahmed S Al-Araji. 2021. A one decade survey of autonomous mobile robot systems. *International Journal of Electrical and Computer Engineering* 11, 6 (2021), 4891.

[308] Guohao Zhang and Li-Ta Hsu. 2018. Intelligent GNSS/INS integrated navigation system for a commercial UAV flight control system. *Aerospace science and technology* 80 (2018), 368–380.

[309] Hongwei Zhang, Shitong Zhang, Yanhui Wang, Yuhong Liu, Yanan Yang, Tian Zhou, and Hongyu Bian. 2021. Subsea pipeline leak inspection by autonomous underwater vehicle. *Applied Ocean Research* 107 (2021), 102321.

[310] Xiwu Zhang, Yan Su, and Xinhua Zhu. 2017. Loop closure detection for visual SLAM systems using convolutional neural network. In *2017 23rd International Conference on Automation and Computing (ICAC)*. IEEE, 1–6.

[311] Zhongzhong Zhang, Erkan Kayacan, Benjamin Thompson, and Girish Chowdhary. 2020. High precision control and deep learning-based corn stand counting algorithms for agricultural robot. *Autonomous Robots* 44, 7 (2020), 1289–1302.

[312] Chao Zhou, Yucheng Wei, and Tieniu Tan. 2003. Mobile robot self-localization based on global visual appearance features. In *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, Vol. 1. 1271–1276 vol.1. https://doi.org/10.1109/ROBOT.2003.1241767

[313] Wu Zhou, E Shiju, Zhenxin Cao, and Ying Dong. 2016. Review of slam data association study. In *Proceedings of the 2016 International Conference on Sensor Network and Computer Engineering, Xian, China*. 8–10.

[314] Yi Zhu, Tao Zhang, Jingyan Song, and Xiaqin Li. 2012. A new bug-type navigation algorithm for mobile robots in unknown environments containing moving obstacles. *Industrial Robot: An International Journal* 39, 1 (2012), 27–39.

[315] Jon Zubizarreta, Iker Aguinaga, and Jose Maria Martinez Montiel. 2020. Direct sparse mapping. *IEEE Transactions on Robotics* 36, 4 (2020), 1363–1370.