Environment perception, mapping and motion planning for Autonomous mobile robots

Jonas Ulmen[1], Roshan Sathyanarayana Shenoy[2], Nagasai Pavanswaroop Ainapurapu[2], and Shrenik Lalani[2]

***Abstract*—An autonomous mobile robot (AMR) must able to navigate through environment autonomously while performing goal-oriented tasks. Even in the static environment an autonomous mobile robot has to face unprecedented challenges. Dynamic environment introduces lots of new challenges also. An AMR should fulfill tasks such as object detection, localisation, mapping and motion planning without human intervention that too while performing predefined tasks. Motion planning is one of the most challenging tasks performed by an autonomous robot. While planning its trajectory, an AMR must consider not only obstacles but also dynamic objects such as humans or other mobile robots. AMR must maintain safe distance from obstacles to avoid any damage. In this paper various algorithms for trajectory planning are discussed. Configuration space is one of the most important concepts when we consider motion planning. Firstly, basic motion planning approaches such as linear and circular trajectories are discussed. It is followed by deterministic algorithms such as A* and sampling-based algorithms such as Rapidly-Exploring Random Trees (RRT). Potential Field Methods and optimized path planning using Genetic Algorithm are also discussed.**

***Index Terms*—IEEE, IEEEtran, journal, LaTeX, paper, template.**

.

I. Introduction

Knowing the position of underwater robots and obtaining maps of the surrounding environment is essential for a variety of robot tasks, from gathering geo-referenced data to autonomous navigation and exploration. Simultaneous Localization and Mapping (SLAM) offers a framework to incrementally build a map while a robot moves through an unknown area and to use that map to localize the robot simultaneously. A typical implementation of SLAM in the underwater environment involves the use of dead-reckoning, acoustic sensors and cameras[]. In the last few years, the use of cameras as the primary sensor for SLAM has increased. This branch of SLAM is also referred to as visual SLAM (vSLAM) which mainly focuses on estimating the pose of the camera from partially overlapping images from different viewpoints and creates a map of images or a cloud of points. Visual SLAM can be categorized based on how the images are processed in direct algorithms, where complete image intensities are processed, and feature-based, where only certain key-points of the image are computed.[]

Motion planning aims to determine the course of action/path while moving from an initial position to a final position avoiding obstacles. The main objective of motion planning of a mobile robot is to explore an effective and efficient path so that a robot moves in a prescribed route within a specified framework. Continuously changing environment, multiple agents, and dynamic obstacles make the problem of motion planning more difficult. Applications of mobile robots are increasing in multiple domains and it encourages researchers to carry out vast research work in the field of motion planning.

Path planning is subset of motion planning which is purely geometric matter as it refers to the generation of a path without a specified time law, while motion planning assign a time law to the geometric path and also considers kinematic and dynamic constrains of a robot [1].

II. Feature Detection and Description

Feature detection is the process of computing the abstraction of the image information and making a local decision at every image point to see if there is an image feature of the given type existing in that point. Feature detection and image matching have been two important problems in machine vision and robotics, and their applications continue to grow in various fields. An ideal feature detection technique should be robust to image transformations such as rotation, scale, illumination, noise and affine transformations. In addition, ideal features must be highly distinctive, such that a single feature to be correctly matched with high probability [2], [3]

A feature descriptor is an algorithm which takes an image and outputs feature descriptors/feature vectors. Feature descriptors encode interesting information into a series of numbers and act as a sort of numerical “fingerprint” that can be used to differentiate one feature from another.

Ideally, this information would be invariant under image transformation, so we can find the feature again even if the image is transformed in some way. After detecting interest point we go on to compute a descriptor for every one of them. Descriptors can be categorized into two classes:

- Local Descriptor: It is a compact representation of a point’s local neighborhood. Local descriptors try to resemble shape and appearance only in a local neighborhood around a point and thus are very suitable for representing it in terms of matching.
- Global Descriptor: A global descriptor describes the whole image. They are generally not very robust as a

[1]Jonas Ulmen is with the Department of Electrical and Computer Engineering, Technische Universität Kaiserslautern, Germany, ulmen@rhrk.uni-kl.de

[2]Roshan Sathyanarayana Shenoy, Nagasai Pavanswaroop Ainapurapu, and Shrenik Lalani are with the Department of Mechanical and Process Engineering, Technische Universität Kaiserslautern, Germany, roshan@rhrk.uni-kl.de, nagas@rhrk.uni-kl.de, and lalani@rhrk.uni-kl.de

Manuscript received April 19, 2005; revised August 26, 2015.

change in part of the image may cause it to fail as it will affect the resulting descriptor.

A. Computer Vision Background

Computer vision is the field of computer science that focuses on replicating parts of the complexity of the human vision system and enabling computers to identify and process objects in images and videos in the same way that humans do. Until recently, computer vision only worked in limited capacity. Thanks to advances in artificial intelligence and innovations in deep learning and neural networks, the field has been able to take great leaps in recent years and has been able to surpass humans in some tasks related to detecting and labeling objects. One of the driving factors behind the growth of computer vision is the amount of data we generate today that is then used to train and make computer vision better.

B. Machine Learning vs Deep Learning

Machine Learning: Machine Learning is the practice of using algorithm to break up data, learn from them and then use this learning to make some prediction about certain things. We are not doing any hard-coding with some specific set of instruction to accomplice any task, instead machine is trained with huge amount of data which give an ability to trained model so that it can perform specific task, i.e. Machine-learning programs, in a sense, adjust themselves in response to the data they're exposed to.

Machine learning is closely related to (and often overlaps with) computational statistics, which also focuses on prediction-making through the use of computers. It has strong ties to mathematical optimization. The "learning" part of machine learning means that ML algorithms attempt to optimize along a certain dimension; i.e. they usually try to minimize error or maximize the likelihood of their predictions being true.

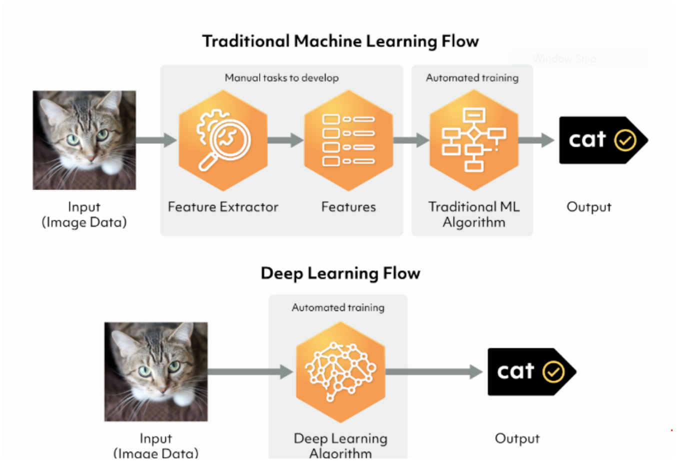

Fig. 1. Machine Learning Vs Deep Learning Flow

Deep Learning is subset of Machine Learning. Deep learning model involves feeding a computer system lot of data, which it can use to make decision about other data. Deep learning works in same way as human brain make conclusion with respect to any scenario.

1) Traditional Vision:

We now describe some techniques used in traditional vision. The idea behind each of these techniques is to formulate some way of representing the image by encoding the existence of various features. These features can be corners, colorschemes, texture of image, etc.

- Scale Invariant Feature Transform (SIFT):
  The SIFT algorithm deals with the problem that certain image features like edges and corners are not scale-invariant. In other words, there are times when a corner looks like a corner, but looks like a completely different item when the image is blown up by a few factors. The SIFT algorithm uses a series of mathematical approximations to learn a representation of the image that is scale-invariant.

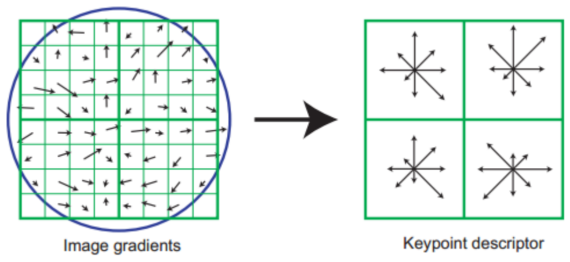

Fig. 2. Scale-Invariant Feature Transform (SIFT)

  In effect, it tries to standardize all images (if the image is blown up, SIFT shrinks it; if theimage is shrunk, SIFT enlarges it). This corresponds to the idea that if some feature (say a corner) can be detected in an image using some square-window of dimension σ across the pixels, then we would if the image was scaled to be larger, we would need a larger dimension $k\sigma$ to capture the same corner (see figure 1). The mathematical ideas of SIFT are skipped, but the general idea is that SIFT standardizes the scale of the image then detects important key features. The existence of these features are subsequently encoded into a vector used to represent the image.

- Speeded Up Robust Feature (SURF):
  The problem with SIFT is that the algorithm itself uses a series of approximations using difference of Gaussians for standardizing the scale. Unfortunately, this approximation scheme is slow. SURF is simply a speeded-up version of SIFT. SURF works by finding a quick and dirty approximation to the difference of Gaussians using a technique called box blur. A box blur is the average value of all the images values in a given rectangle and it can be computed efficiently. SURF algorithms have detection techniques similar to SIFT algorithms. The difference is that SURF algorithms simplify scale-space extrema detection by constructing the scale space via distribution changes instead of using Difference of Gaussian (DoG) filter. To approximate the Laplacian of Gaussian, SURF uses a box filter representation

Simplified scale-space extrema detection in SIFT algorithms accelerates feature extraction speed, so they are several times faster than SIFT algorithms. SURF relies on integral images for image convolutions to reduce computation time.
The descriptor describes a distribution of Haar-wavelet responses within the interest point neighborhood.

- Feature Extraction
- Orientation and size assignment
- Descriptor generation

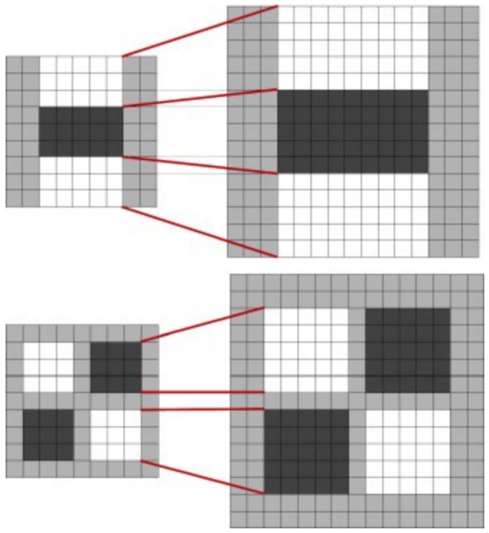
Fig. 3. Speeded Up Robust Feature (SURF)

The image descriptor is generated by measuring an image gradient. SURF algorithms that rely on image descriptor are robust against different image transformations and disturbance in the images by occlusions. Despite reduced time for feature computation and matching, they have difficulty in providing real-time object recognition in resource-constrained embedded system environments.

- Features from Accelerated Segment Test (FAST):
Corners in an input image have distinctive features that clearly distinguish them from surrounding pixels. Reliable detection and tracking of corners in images are possible even when the images have geometric deformations. Thus, most object recognition algorithms utilize corner information to extract features. The Harris corner detector used in the SIFT method has good performance but it is not effective for real-time object recognition due to its long computation time.

FAST corner detector is 10 times faster than the Harris corner detector without degrading performance. It finds corners by examining a circle of sixteen pixels around the corner candidate. This candidate is detected as corner if the intensities of a certain number of contiguous pixels are all above or all below the intensity of the center pixel by some threshold. The extracted interest points lie on distinctive, high-contrast regions of the image.

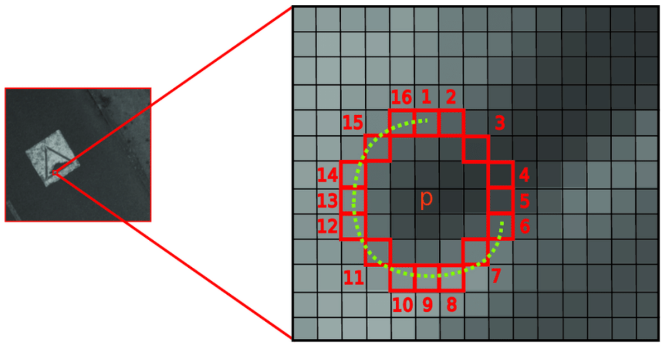

Fig. 4. FAST Corner Detector

III. Deep Learning based descriptors

IV. Background and Related Work

Fabric manipulation is an active area of robotics research [4], [5], [6], [7]. Over the past decade, the research has primarily been focused on three different categories: perception-based manipulation, learning-based algorithms in the real world, and learning-based algorithms in simulation which are then transferred to real robots.

Previous articles by Sundaresan et al.[8], which leverages dense object descriptors [9] to learn visual correspondences for rope using synthetic depth data in simulation. These correspondences are then used to learn new rope manipulation tasks such as rearrangement or knot tying given a single task demonstration. Similar visual correspondence learning methods are also effective for learning correspondences between different cable configurations using taskagnostic RGB data collected entirely in simulation and can be used to recognize cables in the terrain. Precisely, given a user demonstration of the task from a given initial configuration,the approach leverages the learned visual correspondences to perform the same task from different initial configurations by computing geometrically equivalent actions using the correspondences. This approach has a number of appealing properties. First, visual correspondences can be learned purely in simulation without task-specific data and widely applied to a variety of real cables with no further training. Second, training in simulation enables sufficient data variety through domain randomization, making it possible to learn correspondences that generalize to cables with different colors, shapes, and configurations. Third, since perception and control are decoupled, the same perception module can be used on different robots with no additional training.

A. *Traditional Vision-Based Algorithms for Cable Recognition*

Much of the prior work on perception-based deformable object manipulation relies on traditional image processing and vision techniques to estimate the state . This state estimation is then used to define geometric controllers which bring the object into some desired configuration. However, due to limitations in these traditional vision algorithms, most prior work makes specific assumptions on the fabric's initial configurations or requires more complex robotic manipulators to bring the fabric into a desired starting configuration. For example, Miller et al.[10] demonstrate a robust folding pipeline for clothing by fitting a polygonal contour to the fabric and designing a geometric controller on top of it, but assume that the initial state of the fabric is flat. Sun et al.[11], [12] perform effective fabric smoothing by estimating the wrinkles in the fabric, but condition on a nearflat starting fabric. Other work relies on ”vertically smoothing” fabrics using gravity [13], [14], [15], [16], [17] to standardize the initial configuration and to expose fabric corners before attempting the task, which is difficult for large fabrics or single-armed robots.

B. Learning-Based Algorithms in the Real World

More recent approaches have transitioned to end to end learning of fabric manipulation directly on a real system, but these approaches have struggled to generalize to a variety of fabrics and tasks due to the high volume of training data required. For example, Ebert et al.[18] use model-based reinforcement learning to learn fabric manipulation policies which generalize to many tasks, but require several days of continuous data collection on a real physical system and perform relatively low precision tasks. Jia et al.[19], [20] show impressive collaborative human-robot cloth folding under the assumption that fabric has already been grasped and is in a particular starting configuration, and Schulman et al.[21] demonstrate deformable object manipulation while requiring task-specific kinesthetic demonstrations. In follow-up work, Lee et al.[22] consider many of the same tasks as in this paper anddemonstrate that policies can be learned to fold fabric using reinforcement learning with only one hour of experience on a real robot. In contrast, we learn entirely in simulation and decouple perception from control, making it easier to generalize to different fabric colors and shapes and flexibly deploy the learned policies on different robots.

C. Sim-to-Real Learning-Based Algorithms

Due to the recent success of sim-to-real transfer [23], [24] many recent papers leverage simulation to collect large amounts of training data which is used to learn fabric manipulation policies. Seita et al.[25], [26] and Wu et al.[27] followed-up on the smoothing task from [11] by generalizing to a wider range of initial fabric states using imitation learning (DAgger [28]), and reinforcement learning (Soft Actor-Critic [29]) respectively, but still tailor policies specifically for smoothing. Similarly, Matas et al.[30] learn fabric folding policies by using deep reinforcement learning augmented with task-specific demonstrations. These works use simulation to optimize fabric manipulation policies for specific tasks. In follow-up and concurrent work, Hoque et al.[31] and Yan et al.[32] use simulation to train fabric manipulation policies using modelbased reinforcement learning for multiple tasks. In contrast, we leverage simulation to learn visual representations of fabric to capture its geometric structure without task-specific data or a model of the environment and then use this representation to design intuitive controllers for several tasks from different starting configurations.

V. Dense Object Descriptors

The current research problem deals with navigation of a mobile robot over a terrain containing 1D object such as cables. Deformable objects can be challenging to model due to the lack of high-fidelity analytical models and large configuration spaces. Moreover, end-end recognition based on learning from images and interaction on a physical level requires a considerable amount of time and can fail to be generalize across tasks. These challenges can be addressed by considering the previous research carried out on dense object descriptors for robot manipulation. This facilitates the design of interpretable and transferable geometric policies built on top of the learned representations, decoupling visual reasoning and control. The current article reviews approaches that learns point-pair correspondences between initial and goal rope configurations, which implicitly encodes geometric structure, entirely in simulation from synthetic depth images.

We learn visual representations for cables by using dense object descriptors [9], which were shown to enable task oriented manipulation of various rigid and slightly deformable objects [8].

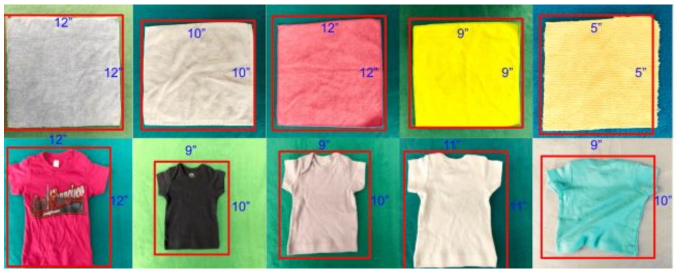

Fig. 5. Fabric specifications used in experiments [8]

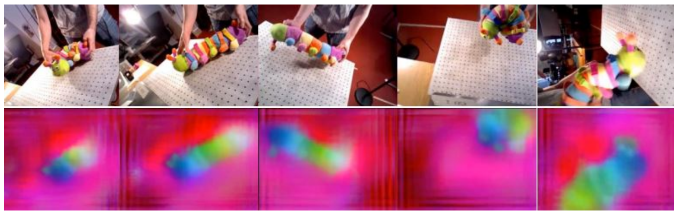

Fig. 6. Example of learned descriptors of a soft caterpillar toy for robot manipulation [9]

These approaches use a deep neural network to learn a representation which encourages corresponding pixels in images of an object in different configurations to have similar representations in embedding space.

Such descriptors can be used to design geometrically structured manipulation policies for grasping [9], assembly [34], or for learning from demonstrations [35]. Sundaresan et al.[8] extend this idea to manipulation of ropes, and demonstrate that deformation-invariant dense object descriptors can be learned for rope using synthetic depth data in simulation and then

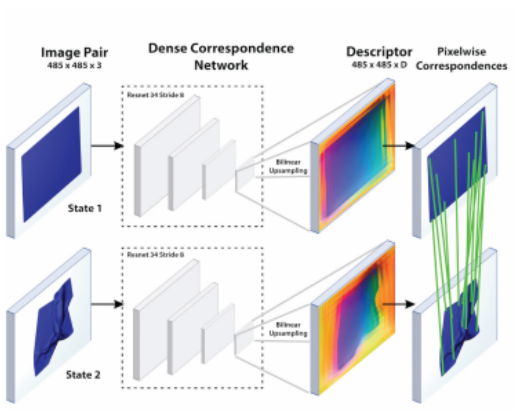

Fig. 7. Example of a pipeline for training dense object nets and then using them for robot fabric manipulation. Left: Training dense object network on pairs of simulated fabric images to learn pixel-wise correspondences using a pixelwise contrastive loss. Right: Using the learned descriptors for policy optimization. Correspondence to map a reference action to a new fabric configuration. For example, image of a wrinkled fabric in "State 2," and using descriptors to figure out the action needed to smooth the fabric from "State 2" to "State 1.[33]

transferred to a real physical system. Sundaresan et al.[8] then use the learned descriptors imitate offline demonstrations of various rope manipulation tasks. In this work, we apply the techniques from [9] to learn descriptors which capture geometric correspondence across different fabric configurations from synthetic RGB images and use them for 2D fabric manipulation.

VI. Methodology

In[9] mapping to a descriptor image using a full resolution RGB image has been performed using a dense object net. The learned deep descriptor mapping is defined by the function $f(\cdot)$. For an RGB image I, we have

$$f(I) : \mathbb{R}^{H \times W \times 3} \rightarrow \mathbb{R}^{H \times W \times D} \tag{1}$$

for some dimension D, $D = 3$, with some larger values, and occasionally with $D = 2$.

If we are trying to map a full resolution RGB image I to some other "space", we would want to decrease the size of the data, but as per [9] the same height and width of the image needs to be maintained in order to get the pixel correspondences.

The function $f(I)$ maps each pixel in the original, three-channel image I, to a D -dimensional vector. The authors generally use $D = 3$, and compared to larger values of D, using $D = 3$ has the advantage in that descriptors can be visualized easily; it means the image is effectively another $H \times W \times 3$ -dimensional image, so upon some normalization (such as to convert values into [0,255]) it can be visualized as a normal color image.

The data and the loss formulation for training f? consists of data tuples with four elements: two images I_a and I_b, then two pixels on the images, u_a and u_b, respectively. Each pixel is therefore a 2-D vector in $\mathbb{R}^2$; in practice, each value in u_a or u_b can be rounded to the nearest pixel integer. We write $f\left(I_a\right)\left(u_a\right)$ for the channel values at pixel location u_a in descriptor image $f\left(I_a\right)$. For example, if f is the identity function and I_a a pure white image, then $f\left(I_a\right)\left(u_a\right) = [255, 255, 255]$ for all possible values of u_a, because a white pixel value corresponds to 255 in all three channels. There are two loss functions that add up to one loss function for the given image pair:

$$\mathcal{L}_{\text{matches }}\left(I_a, I_b\right) = \frac{1}{N_{\text{matches}}} \sum_{N_{\text{mathes}}} \left\|f\left(I_a\right)\left(u_a\right) - f\left(I_b\right)\left(u_b\right)\right\|_2^2 \tag{2}$$

and

$$\begin{aligned} \mathcal{L}_{\text{non-matches }}\left(I_a, I_b\right) = \frac{1}{N_{\text{non-matches}}} \sum_{N_{\text{non-matches}}} \max \Big\{0, M \\ - \left\|f\left(I_a\right)\right. \\ \left.\left(u_a\right) - f\left(I_b\right)\left(u_b\right)\right\|_2^2 \Big\} \end{aligned} \tag{3}$$

The final loss for $\left(I_a, I_b\right)$ is simply the sum of the two above:

$$\mathcal{L}\left(I_a, I_b\right) = \mathcal{L}_{\text{matches }}\left(I_a, I_b\right) + \mathcal{L}_{\text{non-matches }}\left(I_a, I_b\right) \tag{4}$$

Minimizing the loss will encourage f to map pixels such that they are close in descriptor space with respect to Euclidean distance if they are matches, and far away — by at least some target margin M — if they are non-matches. The target margin has been most likely borrowed from the famous hinge loss (or "max margin" loss) that is used for training Support Vector Machine classifiers

VII. Navigation

ONE of the most challenging skills expected of a mobile robot is navigation. Success in navigation requires success in the four navigation building blocks. Perception: the robot must perceive its sensors to extract meaningful data; localization: the robot must evaluate its location in the environment; cognition: the robot must decide how to achieve its goals; and motion control: the robot must modulate its motor outputs to achieve the desired trajectory [36].

A. Challenges to the localization problem

Sensor noises and sensor aliasing are the major challenges to the localization issue. Sensor noise causes a constraint on the accuracy of sensor readings in the same state of the environment and, thus, on the amount of usable useful bits from each sensor read. Usually, the origin of sensor noise problems is that the representation of the robot does not capture any environmental features [36]. Just one example of the apparent noise in a vision-based sensor device is illumination dependence. All of the additional sources of noise are image jitter, signal gain, blooming and blurring, potentially reducing the useful quality of a color video image. The solution is to take several readings into account, to increase the overall information content of the robot's inputs, using temporal fusion or multi-sensor fusion.Sensor aliasing is a phenomenon rarely experienced by humans. In each unique local state, the human sensory system, particularly the visual system, tends to receive unique inputs. Any other position looks different. The influence of this distinctive mapping is only evident when one considers conditions where it does not hold. For instance, the visual system sees only black when traveling through completely dark unknown, one's localization system quickly degrades [36]. The issue raised by sensor aliasing to navigation is that the amount of information is normally inadequate to identify the location of the robot from a single percept reading, even with noise-free sensors. The robot programmer must therefore employ techniques that base the localization of the robot on a series of readings and, therefore, sufficient knowledge to recover the location of the robot over time [36].

B. Simultaneous localization and mapping

SLAM outlines the importance of obtaining a spatial map of a mobile robot world when locating the robot relative to this model simultaneously. In the pursuit of creating fully autonomous mobile robots, the SLAM problem is widely regarded as one of the most critical issues. It still presents great challenges, despite considerable progress in this area. We currently have robust methods that are static, standardized, and of limited size for mapping environments. This remains largely an area of active research to map unstructured, complex, or large-scale environments [37]. Categorisation of the Problem of SLAM: SLAM problems are differentiated according to a variety of different characteristics. The type of aspects outlined by making the underlying assumptions clear are found in most major research papers. We have already come across one such distinction: full versus online. Full SLAM estimates the entire path, whereas Online SLAM seeks to recover only the most recent pose. The following are other prevalent distinctions [37].

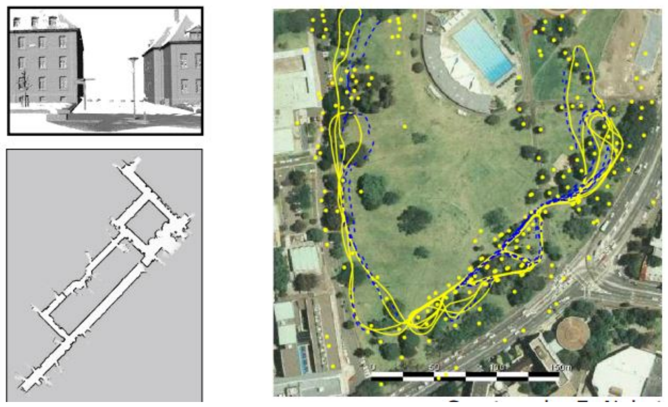

Fig. 8. Volumetric vs. feature-based SLAM

1) Volumetric Versus Feature-Based: Volumetric Versus Feature-Based: The map is sampled at a resolution high enough to allow for photorealistic environmental reconstruction in volumetric SLAM. The volumetric SLAM map is typically very high-dimensional, so it can be quite involved in the computation. Feature-based SLAM removes the sensor stream from sparse features. The map is then composed only of characteristics. Our example of a point landmark is an example of feature-based SLAM. Feature-based SLAM techniques appear to be more effective, but due to the fact that the extraction of features discards information in the sensor measurements, their results may be inferior to volumetric SLAM [37].

2) Topological Versus Geometric Maps: Topological Versus Geometric Maps: Some mapping techniques only retrieve a qualitative environmental definition that characterizes the relationship between fundamental locations. These techniques are referred to as topological ones. A topological map could be described over a number of different locations and a series of qualitative relationships between these locations (e.g., place A is adjacent to place B). Metric SLAM methods provide the relationship of such locations with metric knowledge. Topological approaches have fallen out of style in recent years, despite ample evidence that humans also use topological data for navigation [37].

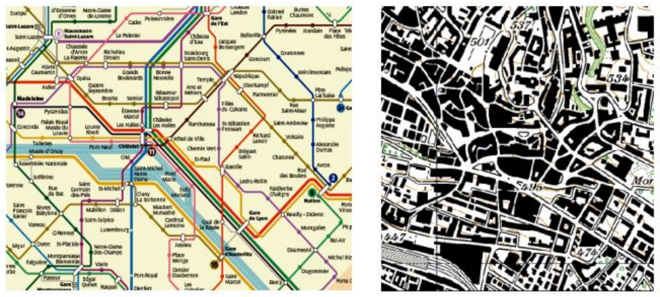

Fig. 9. Topologic vs. geometric maps

3) Known Versus Unknown Correspondence: The issue of correspondence is the problem of linking the identity of things sensed to other things sensed. We have assumed in the landmark example of the Figure 7 that the identification of landmarks is known. Some SLAM algorithms, although others do not, make such an assumption. Special mechanisms for estimating the correspondence of measured characteristics to previously observed landmarks on the map are given by algorithms which do not make this assumption. It is understood that the problem of estimating the correspondence as a data association problem, and one of SLAM's most challenging issues [37].

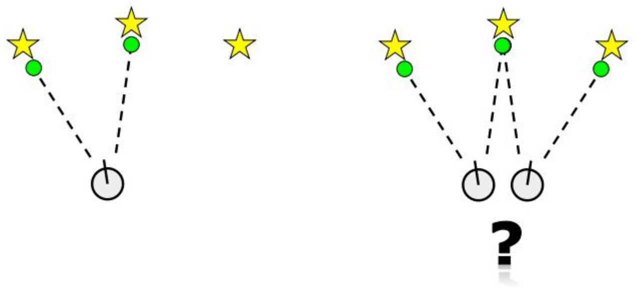

Fig. 10. Known vs. unknown correspondence

4) Static Versus Dynamic: Algorithms with static SLAM presume that the world does not change over time. Dynamic approaches allow for environmental adjustments. Static conditions are assumed by the vast majority of the SLAM literature; dynamic results are mostly viewed only as measurement outliers. Methods that motivate motion in the environment are more involved, but in most applications they tend to be more stable from which most others are derived [38].

Fig. 11. Static vs. dynamic environments

5) Small Versus Large Uncertainty: The degree of position uncertainty that they can handle distinguishes SLAM issues. The simplest SLAM algorithms only allow for small errors in the estimation of the position. In cases where a robot goes down a path that does not intersect itself and then returns along the same path, they are good. The same location can be approached from different directions in several conditions. A large amount of uncertainty can arise from the robot here. This problem is known as the loop-closing problem. The uncertainty can be considerable when closing a loop. A main feature of modern-day SLAM algorithms is the ability to close loops. If the robot can sense information about its location in any absolute coordinate frame, for example, with the use of a satellite-based global positioning system (GPS) receiver, the uncertainty can be minimized from which most others are derived [38].

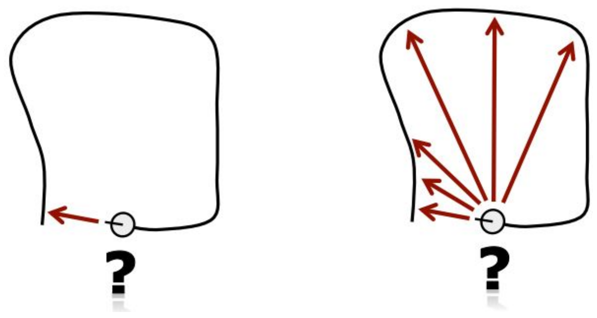

Fig. 12. Small vs. large uncertainty

6) Active Versus Passive: In passive SLAM algorithms, some other entity controls the robot, and the SLAM algorithm is purely observing. The vast majority of algorithms are of this type; they give the robot designer the freedom to implement arbitrary motion controllers, and pursue arbitrary motion objectives. In active SLAM, the robot actively explores its environment in the pursuit of an accuratemap. Active SLAM methods tend to yieldmore accuratemaps in less time, but they constrain the robot motion. There exist hybrid techniques in which the SLAM algorithm controls only the pointing direction of the robot's sensors, but not the motion direction from which most others are derived [38].

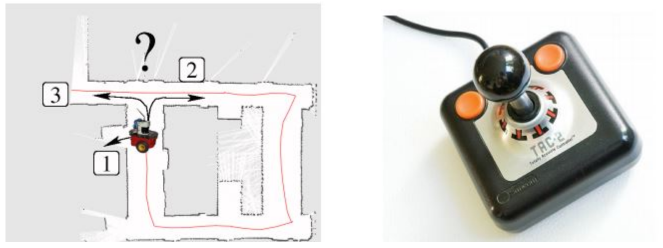

Fig. 13. Active vs. passive SLAM

Fig. 14. Single-robot vs. multi-robot SLAM

7) Single-Robot Versus Multirobot SLAM: Most SLAM problems are defined for a single-robot platform, although recently the problem of multirobot exploration has gained in popularity. Multirobot SLAM problems come in many flavors. In some, robots are able to observe each other, while in others robots are told their relative initial locations. Multi-robot SLAM problems are also distinguished by the type of communication allowed between the different robots. More realistic are setups in which only nearby robots can communicate, and the communication is subject to latency and bandwidth limitations from which most others are derived [38].

C. The Three Main SLAM Paradigms

There exists three basic SLAM paradigms. All the others are derived from these. This section reviews three basic SLAM paradigms, from which most others are derived [38]. The first, known as extended Kalman filter (EKF) SLAM, is historically the earliest but has recently become slightly unpopular due to its limiting computational properties. The second, which is based on graphical representations, successfully applies sparse nonlinear optimization methods to the SLAM problem, and has become the main paradigm for solving the full SLAM problem. The third and final method uses nonparametric statistical filtering techniques known as particle filters. It is a popular method for online SLAM, and provides a fresh new solution to the data association problem in SLAM[38].

1) Extended Kalman Filters: To operate a mobile robot, as explained above, information from several sources mostly needs to be combined. Different types of sensors, however, have different resolutions and error degrees. As a consequence, data coming from trustworthy sources may be more relevant or bear more weight than less credible sources. A general way to determine the data from sources that are more or less accurate and what weights must be given to each source's data is by applying the measurements to a weighted pounder. This approach is best known as the Kalman filter and is one of the most frequently used sensory fusion techniques in mobile robotics applications [39]. In Figure 15, a Kalman filter is illustrated where the blocks represent the measurements, devices, and the environment. This filter is used when the system to be modeled fails for having a nonlinear Gaussian noise distribution. While the errors are approximately Gaussian, the Kalman filter can be used nevertheless but will probably not be optimal[39].

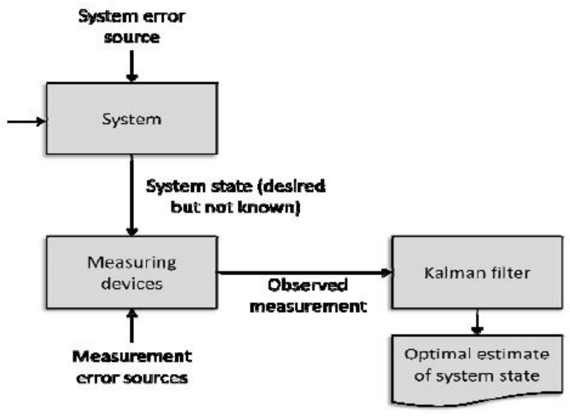
System error source
System
System state (desired but not known)
Measuring devices
Observed measurement
Kalman filter
Measurement error sources
Optimal estimate of system state

Fig. 15. Typical Kalman filter application

An extended Kalman Filter (EKF) is used for nonlinear systems. This includes the linearization of the plant, and the linearization of the measurement, if necessary. The high order terms of the Taylor expansion are also cancelled. In the Kalman filter family, the current linearized error propagation will result in large errors and inconsistency in the problem of simultaneous localization and mapping (SLAM). The use of iteration in the EKF and the sigma point Kalman filter are one technique to mitigate this condition (SPKF) [39].

$$p\left(x_t, m \mid Z_T, U_T\right) \sim N\left(\mu_t, \Sigma_t\right) \tag{5}$$

The high-dimensional vector μ_t contains the robot's best estimate of its own location and the location of the features in the environment. The most frequently used approaches for resolving a SLAM issue were addressed by Thrun et al.[38]. These techniques include the extended Kalman filter, the sparse optimization technique based on graphs, and the particle filter. The earliest SLAM algorithm relative to others is the one based on the Extended Kalman filter of SLAM. In the EKF, the covariance matrix is quadratic in nature and this takes time and space. On the other hand, the benefit of the sparse optimization technique based on graphs is that it can be scaled to a much higher dimensional map compared to EKF[38]. The update time of the graphical method is also constant and the necessary amount of memory is linear. The particle filter method can be viewed via the FastSLAM algorithm method. This approach has the benefit of being inexpensive in terms of computing. This can also be used as a filter and requires linear-logarithmic time for its update point. The data associations can be easily sampled by FastSLAM. The downside is that as the evaluation over maps runs in the loop, the number of particles to evaluate becomes very high [38].

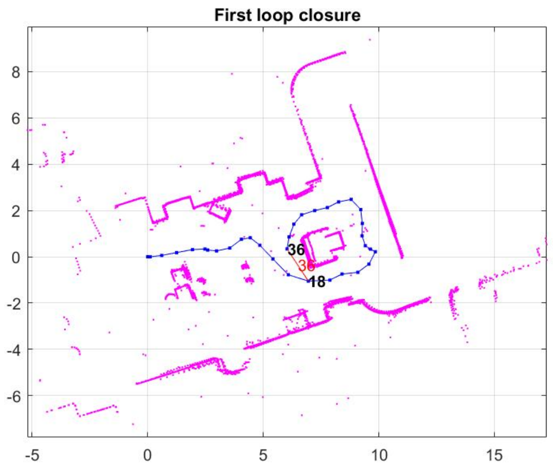

Fig. 16. EFK SLAM

The work carried out by the scientific community on this part of research is divided into two distinct forms of frameworks. First, works using landmarks will be displayed. Most of these papers include vision sensors and methods of triangulation. And second, laser sensor-based methods are shown. These two structures reflect the attempt to enhance the solution to the problem of localization, and their significance is individually highlighted.

Landmark and triangulation method: As a navigation technique, landmark methods and signal triangulation rely on the detection of environmental features or objects. Features and objects must be known a priori, or dynamically derived. Environmental characteristics are classified into four types:

1) Active beacons fixed at known positions and actively transmitting ultrasonic, IR (infrared) or RF (radio frequency) signals to determine the absolute position of the robot in the direction of incidence receiving.
2) Artificial landmarks that are specially built objects or markers positioned in known environmental locations.
3) Natural landmarks that are distinctive characteristics of the environment and can be extracted by sensors.
4) Environmental models that are created from prior environmental information and can be used to match new sensor observations.

Natural landmark-based navigation is versatile among the environmental features discussed since no specific artificial landmarks are required, but may not function well when landmarks are scarce or a priority is not known to the environment. Today, the advent of visual sensors has led to a trend towards the use of digital cameras to capture information as the key sensor. The simultaneous process of camera localization and mapping is usually referred to as visual SLAM and resolved with EKF. The fundamental strength of EKF in solving the problem of SLAM lies in its iterative approach to estimation determination. Building an expanded map of its surrounding area where the robots move through some waypoints. The map is progressively built by presenting it as an increase in the projected states, which are nothing but a list of environmental positions of the characteristics (or landmarks) along with the robot's location. In order to solve the localization problem, the robot position and the positions of observed stationary landmarks therefor can be calculated (for example line segments) [40]. The observation-update step requires that each time an observation is made, all the landmarks and the joint covariance matrix be modified. This implies that with the number of landmarks on a map, the scale of the calculation extends quadratically. In addition, vision-based methods present many issues with occlusions, activity in real time, and changes to the environment. Consequently, only when it moves in their vicinity will the robot sense the presence of the tags. As a result, the importance of combining this information with data obtained from other sensors (e.g. odometry) is observed [40].

Laser range finder method: The localization system based on the laser scanner and retro-reflective landmarks is a promising absolute positioning technique in terms of performance and cost. The laser actively emits a signal and records its echo, the output signal being a light beam. Lasers provide much more focused beams than other sensors like sonars. This is crucial when hitting a smooth surface at an angle.[41] use sensor fusion between an omnidirectional camera and a 3D laser range finder (LRF). This approach takes advantage of the metric information provided by the LRF and combines it with the omnidirectional vision. Then camera extracts the vertical lines in the environment and using a scan matching technique, solves the SLAM problem. However, the authors do not consider occlusions and illumination changes [41]. In [42], the EKF is used to localize the mobile robot with a LRF sensor in an environment demarcated with line segments. Simulating the kinematic model of the robot performs a prediction step. A method for estimating the covariances of the line parameters based on classic least squares (LSQ) is proposed. This method is compared with the method resulting from the orthogonal LSQ in terms of computational complexity. The results of a comparison show that the use of classic LSQ instead of orthogonal LSQ reduces the number of computations in a localization algorithm that is a part of a SLAM. In the input noise covariance matrix of the EKF the standard deviation of each angular speed of robot wheels is calculated as being proportional to the angular speed of the robot wheels. A correction step is performed minimizing the difference between the matched line segments from the local and global maps [19]. If the overlapping rate between the most similar local and global line segments is below the threshold, the line segments are paired. The line covariances of parameters, which arise from the LRF distance-measurement error, comprise the output noise covariance matrix of the EKF. Line segments were chosen because they require a smaller amount of computer memory in comparison with the occupancy grids method [42]. Traditionally, many nonlinear Bayesian estimation problems are solved using the EKF. But when the dynamic models and measurements are highly nonlinear and the measurement noise is not Gaussian, linearized methods may not always be a good approach [43]. Popular alternatives to Gaussian techniques are nonparametric filters. Nonparametric filters do not rely on a fixed functional form of the posterior, such as Gaussians. Instead, they approximate posteriors by a finite number of values, each roughly corresponding to a region in state space.

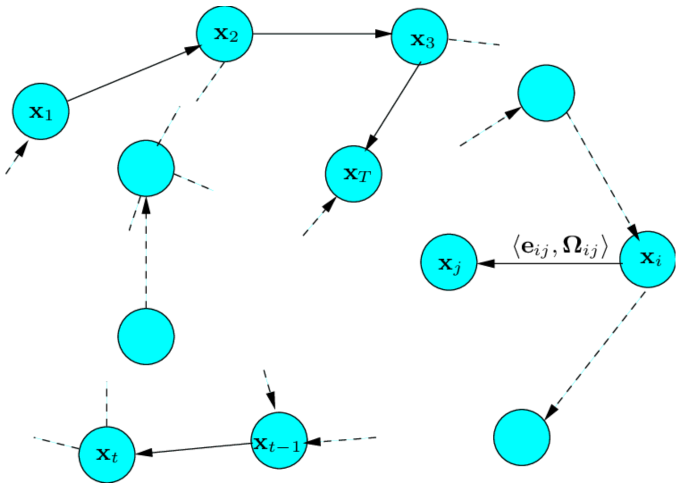

Fig. 17. Illustration of the graph construction.

2) Graph-Based Optimization Techniques: Graphical SLAM methods have the advantage that they scale to much higher-dimensional maps than EKF SLAM. The key limiting factor in EKF SLAM is the covariance matrix, which takes space (and update time) quadratic in the size of the map. No such constraint exists in graphical methods [44]. The update time of the graph is constant, and the amount of memory required is linear (under some mild assumptions). Performing the optimization can be expensive, however. Technically, finding the optimal data association is suspected to be an NP-hard problem, although in practice the number of plausible assignments is usually small [45]. The continuous optimization of the log-likelihood function depends among other things on the number and size of loops in the map. A graph-based SLAM approach constructs a simplified estimation problem by abstracting the raw sensor measurements. These raw measurements are replaced by the edges in the graph which can then be seen as "virtual measurements". More in detail an edge between two nodes is labeled with a probability distribution over the relative locations of the two poses, conditioned to their mutual measurements. In general, the observation model p(zt — xt,mt) is multi-modal and therefore the Gaussian assumption does not hold. This means that a single observation zt might result in multiple potential edges connecting different poses in the graph and the graph connectivity needs itself to be described as a probability distribution. Directly dealing with this multi-modality in the estimation process would lead to a combinatorial explosion of the complexity. As a result of that, most practical approaches restrict the estimate to the most likely topology [45].

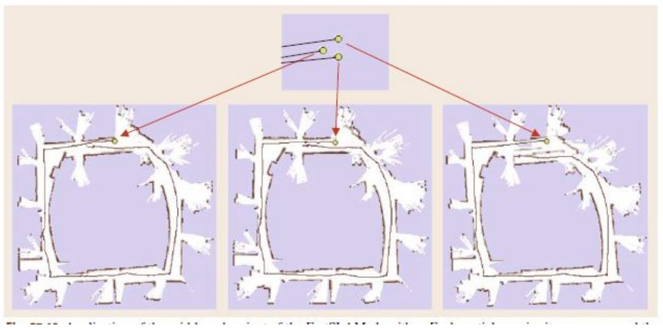
Fig. 18. Application of the grid-based variant of the FastSLAM algorithm. Each particle carries its own map, and the importance weights of the particles are computed based on the likelihood of the measurements given the particle's own map

3) Particle Filters: Particle filters (PF) are sequential Monte Carlo methods under the Bayesian estimation framework and have been widely used in many fields such as signal processing, target tracking, mobile robot localization, image processing, and various economics applications. The key idea is to represent the next probability density function (PDF) of the state variables by a set of random samples or particles with associated weights, and compute estimates based on these samples and weights. PF can estimate the system states sufficiently when the number of particles (estimations of the state vectors which evolve in parallel) is large [46]. These errors are hard to model using a linear estimator such as the Kalman filter because of their high inherent nonlinearity and randomness. However, this method has not yet become popular in the industry because implementation details are missing in the available research literature, and because its computational complexity has to be handled in real-time applications. The first method discussed it is the triangulation by WiFi (IEEE 802.11 WLAN), which consists in identifying access points in the environment. One advantage in using WiFi technology is its frequent use in indoor environments [47].

WiFi: According to WiFi-alliance, over 700 million people use WiFi and there are about 800 million new WiFi devices every year. This freely available wireless infrastructure prompted many researchers to develop WiFi-based positioning systems for indoor environments. Three main approaches for WiFi-based positioning exist: time-based, angle-based, and signal-strength-based approaches. Often times, however, there are no available WiFi access points and it is necessary to find a new manner of identifying the environment. Omnidirectional cameras represent a cheap solution and many features of the environment can be extracted from an image [47].

Omnidirectional cameras and laser range finder: According to [48] two methodologies have been prevalent in live motion and structure estimation from a single moving video camera:

1) Filtering approaches that fuse measurements from all images sequentially by updating probability distributions over features and camera parameters.
2) Bundle adjustment (BA) methods that perform batch optimization over selected images from the live stream, such as sliding window, or in particular spatially distributed keyframes.

[49] presented a methodology to build incremental topological maps. They used omnidirectional images both in robot mapping and localization. These solutions can be categorized into two main groups: feature-based and appearance-based solutions. In the first approach, a number of significant points or regions are extracted from each omnidirectional image and each point is described using an invariant descriptor[49]. All the experiments have been carried out with a set of omnidirectional images captured by a catadioptric system mounted on the mobile platform. Each scene is first filtered to avoid lighting dependence and then is described through a Fourier-based signature that presents a good performance in terms of amount of memory and processing time. In that work, the authors have evaluated the influence of the descriptor in the localization by varying the number of possible associations [49]. The system is able to estimate the position of the robot in the case of an unknown initial position and it is able to track the position of the robot while moving. In the evaluated methods, as they increase the number of particles in the system, the average of localization decreases rapidly [50]. Also, it is possible to correct the weighting of the particles by combining a physical system of forces with a Gaussian weight. Approaches before the present, do not represent all the techniques used in the visual framework.There exist other methods with more than a camera like stereo vision. [50] solve the SLAM problem with an observation model that consider both the 3D positions and the SIFT descriptor of the lankmarks. One advantage of stereo vision is the measure of the depth and therefore the possibility of realice a probabilistic model for visual odometry. In the next section a compilation of parallel techniques is presented, many of which are focused on reducing the number of the computations [50].

D. Relation of Paradigms

In the field of SLAM, the three paradigms mentioned cover the vast majority of work. EKF SLAM comes with a computational barrier that presents significant scaling limitations, as mentioned. EKF SLAM's most promising extensions are based on the creation of local submaps, but the resulting algorithms imitates the graph-based approach in several aspects. Graph-based approaches solve the full problem of SLAM, and are thus not online by default. They derive their intuition from the observation that a sparse graph of soft constraints can model SLAM, where each restriction corresponds either to a motion or a measurement case. Graph-based SLAM has become the method of choice for constructing large-scale maps offline due to the availability of highly efficient optimization methods for sparse nonlinear optimization problems. Searching for data correlations is very easily integrated into the basic mathematical structure, and there are a range of search techniques for relevant communication findings. For the online SLAM problem, there are also extensions of the graph-based SLAM. These help to erase from the graph the old robot positions. Some of the problems arising from the natural interface similarities in the map, which plagued the EKF, are bypassed by particle filter methods. The individual landmarks in the map become independent by sampling from robot poses, and thus are decorrelated. As a consequence, via a sampled robot pose, and several local, independent Gaussians for its landmarks, FastSLAM can represent the posterior. There are a number of advantages to the particle representation of FastSLAM. Fast-SLAM can be used as a filter computer-wise, and its update requires linear-logarithmic time where quadratic time was required by EKF. In addition, FastSLAM will sample over data association, making it a prime SLAM method with unidentified data association. On the negative side, particularly for robots seeking to map multiple nested loops, the number of required particles may grow very high. This is addressed by FastSLAM extensions that use occupancy grid maps instead of Gaussian landmarks, and demonstrated state-of-the-art examples in the construction of large maps.

VIII. Trajectory planning

Geometric path, the kinmematic and dynamic constraints of the manipulator/wheeled robot are taken as inputs to a motion planning algorithm which in turn returns trajectory of joints or robot, expressed as a sequence of values of position, velocity and acceleration.

In [51], author has mentioned approaches such as linear and circular trajectories and De Casteljau's algorithm for solving bezier curves. Also, different path planning approaches can be classified mainly into three groups: deterministic algorithms, probabilistic algorithms, and heuristic algoeithms [52]. According to [44], motion planning methods can be classified as complete methods, grid methods, sampling methods, and virtual potential fields. It is desired that a path planning algorithm generates optimal path in minimum time period. In order to achieve optimal path within minimum time, in [53], the authors have propose path planning algorithm using Genetic Algorithm. An approach to generate optimized local path using 3d point cloud generated by a stereo camera is proposed in [54]. In [1], the authors have classified motion planning algorithms according to:

- minimum execution time
- minimum energy (or actuator effort)
- minimum jerk

IX. Configuration Space and Simple Motion Planning Approaches

Motion planning can be done offline or online. In offline mode, the path planning task starts from a point where the robot has a complete knowledge about: it's environment with obstacles (given map or global perception through camera for example), its initial position, and its final goal within this environment. The offline path planing task simply connect the initial position to the final position, then the agent will execute the generated path. In online mode, path planning is carried out in parallel while moving towards the goal, and perceiving (locally or globally) the environment including its changes with the help of various sensors. Offline path planning is generally used for static environment (or slowly changing) and only when global map of the environment is initially available (given or built). And thus, this approach can ensure global optimum path (in terms of safety, shortest path, time, energy). But for dynamic environments, online path planning must be used since the path must be updated according to environment changes. This approach is also necessary for motion in incompletely known environment map, since the agent is discovering its map while moving and needs to update the path according to any new knowledge.

A. Configuration Space

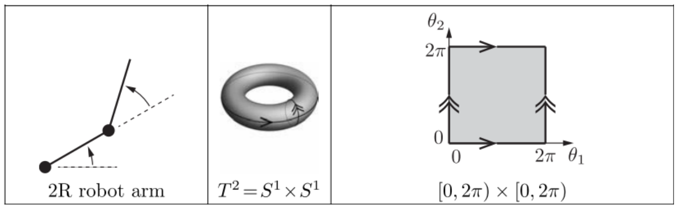

Fig. 19. C-space of 2R robot arm [44]

For planning the motion configuration space (C-space) is one of the most important things. "The configuration of a robot is a complete specification of the position of every point of the robot. The minimum number n of real-valued coordinates needed to represent the configuration is the number of degrees of freedom (dof) of the robot. The n-dimensional space containing all possible configurations of the robot is called the configuration space (C-space). The configuration of a robot is represented by a point in its C-space" [44]. Fig. 16 depicts the C-space of 2R robot arm.

With the help of C-space a planning problem can be abstracted in a unified way [55]. [55] states that a robot with complex geometry shape is mapped to a single point in the C-space. The number of degree of freedom of a robot system is the dimension of the C-space, or minimum number of parameters needed to specify a configuration [55]. Fig. 17 shows that how the obstacle region can be traced and complex geometry shape of robot is converted to a single point by sliding the robot around obstacles. Hence, Motion planning for the robot becomes equivalent to motion planning for a point in C-space [55].

B. Linear and Circular Trajectories

According to [51], with the help of standard drive kinematics straight line movements and driving along a circular path with radius bigger than a minimal radius is possible. Also, based on vehicle parameters the radius (r) can be calculated [51]. As shown in Fig. 18, $\overrightarrow{x_1}$, and $\overrightarrow{x_2}$ for minimal radius r to pass $\overrightarrow{p_2}$ on a circular path can be found when $\overrightarrow{p_1}$, $\overrightarrow{p_2}$, $\overrightarrow{p_3}$ and r are known [51].

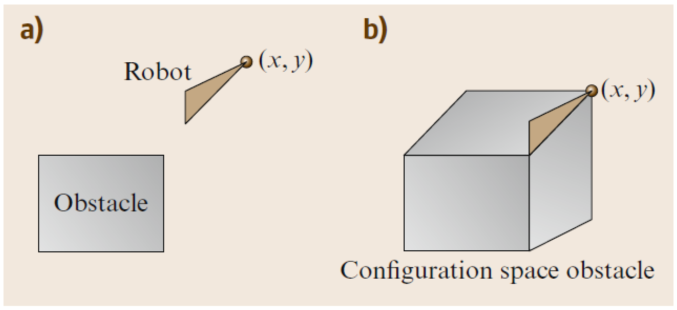

Fig. 20. (a). a triangular robot moves in a workspace with a single rectangular obstacle. (b) The C-space obstacle [55]

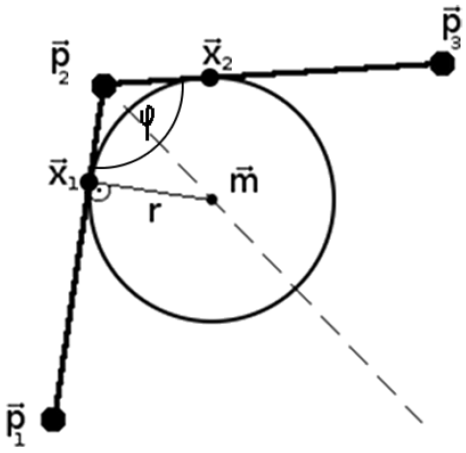

Fig. 21. Linear and Circular Trajectory [51]

C. Bezier Curves

Given $n+1$ control points, polynomial curve of n degree can be solved [51]. Also, Bezier curve for a specific parameter t can be easily evaluated with De Casteljau's algorithm [51].
Formulas:
Initial points set to control points: $\vec{c}_i^{\,0} = P_i$
Evaluation using recurrent relation: $\vec{c}_i^{\,k+1} = (1-t)\vec{c}_i^{\,k} + t\vec{c}_{i+1}^{\,k}$

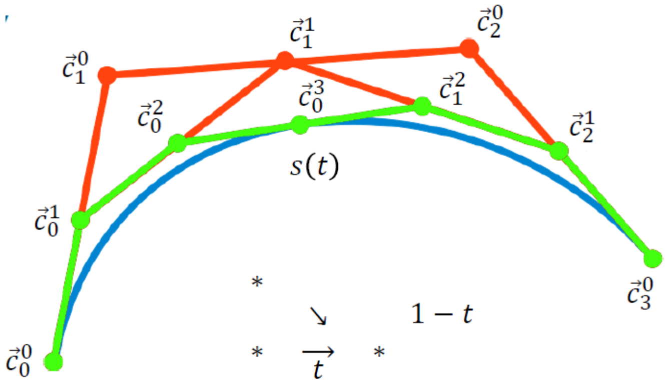

Fig. 22. Initial points and recurrent points while solving Bezier curves with De Casteljau'S algorithm [?]

X. Deterministic Algorithms

A. A^* and Dijkstra's Algorithm

A common approach for PP consists of representing the configuration space of the robot as a weighted directed graph G=(V,E), where V is a set of possible robot states (locations), and E is a set of edges which represent transitions between these states. The weight of each edge corresponds to the cost of transition between the two states (locations) [52]. Once the free space is represented as a graph, a motion plan can be found by searching the graph for a path from the start to the goal [44].

One of the most powerful and popular graph search algorithms is A^* search [44]. Consider a graph with multiple nodes. We want to reach the target node from the starting node as quickly as possible. A^* algorithm picks at each step the node according to a value 'f' which is equal to the sum of 'g' and 'h'. At each step it picks the node having the lowest 'f', and process it.
$f(n) = g(n) + h(n)$
where n is the previous node on the path,
g(n) is the cost of the path from the start node n,
h(n) is a heuristic that estimates the cost of the cheapest path from n to the target node.

To overcome limitations of A^* algorithm, in [45], the authors came up with enhanced A^* algorithm by adding a new parameter namely number of turnings (p(n)) the robot makes during its traverse. The authors has designed a grid map to represent an unknown environment containing unknown static obstacles and identifying the quadrant in which the goal is situated. The authors state that since the goal is present in any one of the four quadrants only, the quadrant is identified and the movement of the robot to that quadrant is directed and hence the robot's path could be optimized. [45]

If in A^* algorithm, heuristic h(n) is always estimated as zero, then this variant is called Dijkstra's algorithm [44]. Dijkstra's algorithm is also guaranteed to find a minimum-cost path but on many problems it runs more slowly than A^* owing to the lack of a heuristic look-ahead function to help guide the search [45].

B. Visibility Graph

One approach to complete path planning is based on representing the complex high-dimensional space C_{free} by one dimensional roadmap R [44]. According to [44] roadmap R should have following properties:

- **Reachability**: From every point q $\in C_{\text{free}}$, a free path to a point $q^{'} \in$ R can be found trivially (e.g., a straight-line path).
- **Connectivity**: For each connected component of C_{free}, there is one connected component R.

[44] further states that constructing a roadmap of C_{free} is complex in general. By considering the nodes in an environment as vertices of a graph, the set of all feasible (collision-free) connections between two vertices is called the "Visibility Graph" [52]. According to the authors, to find a visibility graph with vertices as close as possible to obstacles, the outer edges

of obstacles can be considered as vertices and the feasible connections between them as edges of the graph and after Dijkstra's algorithm can be employed to find the shortest path between start and destination nodes. [52]

In [44], the authors have proposed Adaptive Roadmap Approach which combines ability of some algorithms to produce short paths with the characteristic of other algorithms that route robots safely away from obstacles and a minimum safe distance is introduced to the algorithm in order to modify it based on the complexity of the map and the certainty of the robot's position. Proposed algorithm in [?] generates a set of waypoints, through which the robot can navigate without colliding with obstacles.

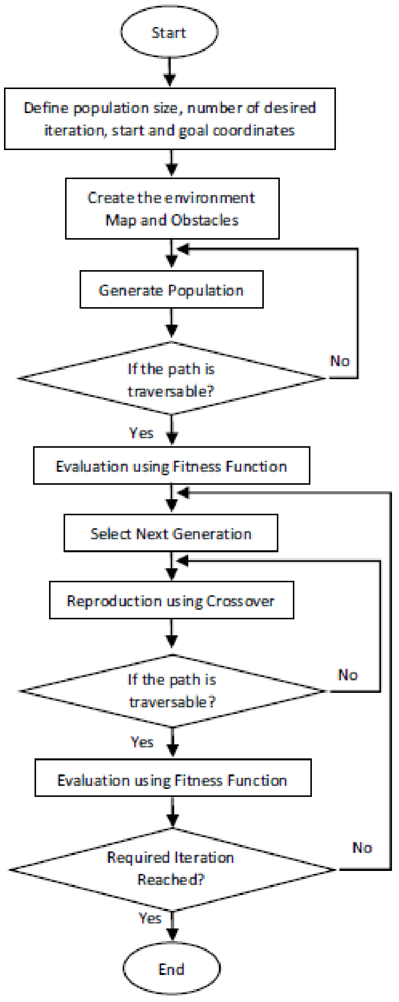
Start
Define population size, number of desired iteration, start and goal coordinates
Create the environment Map and Obstacles
Generate Population
If the path is traversable?
No
Yes
Evaluation using Fitness Function
Select Next Generation
Reproduction using Crossover
If the path is traversable?
No
Yes
Evaluation using Fitness Function
Required Iteration Reached?
No
Yes
End

Fig. 23. The workflow of the proposed algorithm [53]

XI. Sampling Based Algorithms

A. Rapidly-Exploring Random Trees

In [56], the authors state that the Rapidly-exploring Random Tree (RRT) is an efficient data structure and sampling scheme to quickly search high-dimensional spaces that have algebric constraints (arising from obstacles) and differential constraints (arising from dynamics). According to [56], the key idea of RRT is to bias the exploration toward unexplored portions of the space by sampling points in the state space, and incrementally 'pulling' the search tree toward them. Further details and algorithm for RRT can be found in [56]. In [57], an improvement in RRT algorithm is made so that the algorithm also satisfies the kinematics of the a differential driven robot.

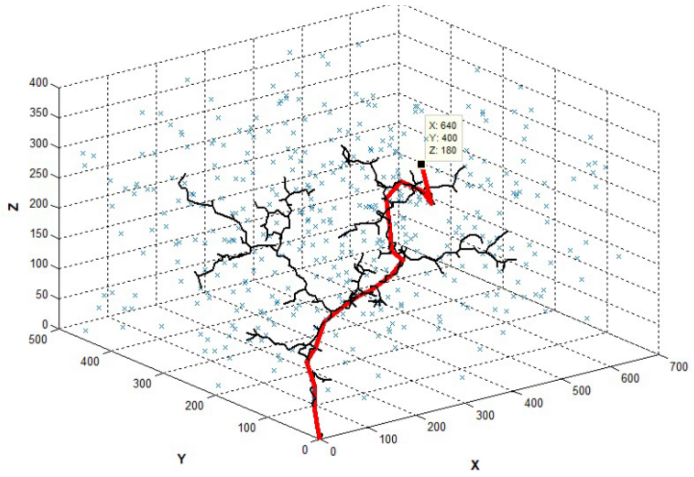

Fig. 24. Basic RRT algorithm [57]

XII. Other Algorithms

A. Potential Field Methods

According to [1], C-space can also be discretized in a dense and regular grid of configurations. As this grid is generally very wide, [1] states that this approach requires very powerful heuristic for the path research and potential field method can be a possible solution. In Potential Field Approach a manipulator or robot moves in a field of force in such a way that the position to be reached is an attractive pole for a robot and obstacles are repulsive surfaces for a robot [58]. To overcome the problem of local minima, due to which a robot runs into a dead end, [?] came up with the Virtual Force Field Method. According to [?], proposed algorithm allows real-time obstacle avoidance with fast mobile robots.

B. Optimized Path Planning Using Genetic Algorithm

[53] emphasize optimal path determination between the initial point and the defined goal position in the minimum possible time. [53] proposed the utilization of Genetic Algorithm to find optimal path utilizing minimum possible time. Fig. 20 depicts the flow chart of the proposed algorithm.

XIII. Conclusion

References

[1] A. Gasparetto, P. Boscarioal, A. Lanzutti, and R. Vidoni, "Trajectory planning in robotics," *Mathematics in Computer Science*, vol. 6, no. 3, pp. 269–279, 2012.
[2] B. Moghaddam, C. Nastar, and A. Pentland, "A bayesian similarity measure for deformable image matching," *Image and Vision computing*, vol. 19, no. 5, pp. 235–244, 2001.
[3] B. Shan, "A novel image correlation matching approach." *Journal of Multimedia*, vol. 5, no. 3, 2010.

[4] J. Borras, G. Alenya, and C. Torras, "A grasping-centered analysis for cloth manipulation," *arXiv preprint arXiv:1906.08202*, 2019.

[5] R. Jangir, G. Alenyà, and C. Torras, "Dynamic cloth manipulation with deep reinforcement learning," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 4630–4636.

[6] O. Kroemer, S. Niekum, and G. Konidaris, "A review of robot learning for manipulation: Challenges, representations, and algorithms," *arXiv preprint arXiv:1907.03146*, 2019.

[7] J. Sanchez, J.-A. Corrales, B.-C. Bouzgarrou, and Y. Mezouar, "Robotic manipulation and sensing of deformable objects in domestic and industrial applications: a survey," *The International Journal of Robotics Research*, vol. 37, no. 7, pp. 688–716, 2018.

[8] P. Sundaresan, J. Grannen, B. Thananjeyan, A. Balakrishna, M. Laskey, K. Stone, J. E. Gonzalez, and K. Goldberg, "Learning rope manipulation policies using dense object descriptors trained on synthetic depth data," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 9411–9418.

[9] P. R. Florence, L. Manuelli, and R. Tedrake, "Dense object nets: Learning dense visual object descriptors by and for robotic manipulation," *arXiv preprint arXiv:1806.08756*, 2018.

[10] S. Miller, J. Van Den Berg, M. Fritz, T. Darrell, K. Goldberg, and P. Abbeel, "A geometric approach to robotic laundry folding," *The International Journal of Robotics Research*, vol. 31, no. 2, pp. 249–267, 2012.

[11] L. Sun, G. Aragon-Camarasa, P. Cockshott, S. Rogers, and J. P. Siebert, "A heuristic-based approach for flattening wrinkled clothes," in *Conference Towards Autonomous Robotic Systems*. Springer, 2013, pp. 148–160.

[12] L. Sun, G. Aragon-Camarasa, S. Rogers, and J. P. Siebert, "Accurate garment surface analysis using an active stereo robot head with application to dual-arm flattening," in *2015 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2015, pp. 185–192.

[13] A. Doumanoglou, A. Kargakos, T.-K. Kim, and S. Malassiotis, "Autonomous active recognition and unfolding of clothes using random decision forests and probabilistic planning," in *2014 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2014, pp. 987–993.

[14] Y. Kita, T. Ueshiba, E. S. Neo, and N. Kita, "A method for handling a specific part of clothing by dual arms," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2009, pp. 4180–4185.

[15] Y. Kita, T. Ueshiba, E. S. Neo, and N. Kita, "Clothes state recognition using 3d observed data," in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 1220–1225.

[16] J. Maitin-Shepard, M. Cusumano-Towner, J. Lei, and P. Abbeel, "Cloth grasp point detection based on multiple-view geometric cues with application to robotic towel folding," in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 2308–2315.

[17] F. Osawa, H. Seki, and Y. Kamiya, "Unfolding of massive laundry and classification types by dual manipulator," *Journal of Advanced Computational Intelligence and Intelligent Informatics*, vol. 11, no. 5, pp. 457–463, 2007.

[18] F. Ebert, C. Finn, S. Dasari, A. Xie, A. Lee, and S. Levine, "Visual foresight: Model-based deep reinforcement learning for vision-based robotic control," *arXiv preprint arXiv:1812.00568*, 2018.

[19] B. Jia, Z. Hu, J. Pan, and D. Manocha, "Manipulating highly deformable materials using a visual feedback dictionary," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 239–246.

[20] B. Jia, Z. Pan, Z. Hu, J. Pan, and D. Manocha, "Cloth manipulation using random-forest-based imitation learning," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 2086–2093, 2019.

[21] J. Schulman, J. Ho, C. Lee, and P. Abbeel, "Learning from demonstrations through the use of non-rigid registration," in *Robotics Research*. Springer, 2016, pp. 339–354.

[22] R. Lee, D. Ward, A. Cosgun, V. Dasagi, P. Corke, and J. Leitner, "Learning arbitrary-goal fabric folding with one hour of real robot experience," *arXiv preprint arXiv:2010.03209*, 2020.

[23] F. Sadeghi and S. Levine, "Cad2rl: Real single-image flight without a single real image," *arXiv preprint arXiv:1611.04201*, 2016.

[24] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2017, pp. 23–30.

[25] D. Seita, A. Ganapathi, R. Hoque, M. Hwang, E. Cen, A. K. Tanwani, A. Balakrishna, B. Thananjeyan, J. Ichnowski, N. Jamali, *et al.*, "Deep imitation learning of sequential fabric smoothing policies," *arXiv preprint arXiv:1910.04854*, 2019.

[26] D. Seita, N. Jamali, M. Laskey, A. K. Tanwani, R. Berenstein, P. Baskaran, S. Iba, J. Canny, and K. Goldberg, "Deep transfer learning of pick points on fabric for robot bed-making," *arXiv preprint arXiv:1809.09810*, 2018.

[27] Y. Wu, W. Yan, T. Kurutach, L. Pinto, and P. Abbeel, "Learning to manipulate deformable objects without demonstrations," *arXiv preprint arXiv:1910.13439*, 2019.

[28] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2011, pp. 627–635.

[29] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International Conference on Machine Learning*. PMLR, 2018, pp. 1861–1870.

[30] J. Matas, S. James, and A. J. Davison, "Sim-to-real reinforcement learning for deformable object manipulation," in *Conference on Robot Learning*. PMLR, 2018, pp. 734–743.

[31] R. Hoque, D. Seita, A. Balakrishna, A. Ganapathi, A. K. Tanwani, N. Jamali, K. Yamane, S. Iba, and K. Goldberg, "Visuospatial foresight for multi-step, multi-task fabric manipulation," *arXiv preprint arXiv:2003.09044*, 2020.

[32] W. Yan, A. Vangipuram, P. Abbeel, and L. Pinto, "Learning predictive representations for deformable objects using contrastive estimation," *arXiv preprint arXiv:2003.05436*, 2020.

[33] A. Ganapathi, P. Sundaresan, B. Thananjeyan, A. Balakrishna, D. Seita, J. Grannen, M. Hwang, R. Hoque, J. E. Gonzalez, N. Jamali, *et al.*, "Learning to smooth and fold real fabric using dense object descriptors trained on synthetic color images," *arXiv preprint arXiv:2003.12698*, 2020.

[34] K. Zakka, A. Zeng, J. Lee, and S. Song, "Form2fit: Learning shape priors for generalizable assembly from disassembly," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 9404–9410.

[35] P. Florence, L. Manuelli, and R. Tedrake, "Self-supervised correspondence in visuomotor policy learning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 492–499, 2019.

[36] B. Siciliano and O. Khatib, *Springer Handbook of Robotics*. Berlin, Heidelberg: Springer-Verlag, 2007.

[37] S. Malagon-Soldara, M. Toledano-Ayala, G. Soto-Zarazua, R. Carrillo-Serrano, and E. Rivas-Araiza, "Mobile robot localization: A review of probabilistic map-based techniques," *IAES International Journal of Robotics and Automation (IJRA)*, vol. 4, 03 2015.

[38] S. Thrun and J. J. Leonard, *Simultaneous Localization and Mapping*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 871–889.

[39] K. Shojaei and A. M. Shahri, "Experimental study of iterated kalman filters for simultaneous localization and mapping of autonomous mobile robots," *Journal of Intelligent & Robotic Systems*, vol. 63, no. 3, pp. 575–594, 2011.

[40] K. H. Yu, M. C. Lee, J. H. Heo, and Y. G. Moon, "Localization algorithm using a virtual label for a mobile robot in indoor and outdoor environments," *Artificial Life and Robotics*, vol. 16, no. 3, pp. 361–365, 2011.

[41] L. Teslić, I. Škrjanc, and G. Klančar, "Ekf-based localization of a wheeled mobile robot in structured environments," *Journal of Intelligent & Robotic Systems*, vol. 62, no. 2, pp. 187–203, 2011.

[42] L. Teslić, I. Škrjanc, and G. Klančar, "Using a lrf sensor in the kalman-filtering-based localization of a mobile robot," *ISA transactions*, vol. 49, no. 1, pp. 145–153, 2010.

[43] A. Llarena, J. Savage, A. Kuri, and B. Escalante-Ramírez, "Odometry-based viterbi localization with artificial neural networks and laser range finders for mobile robots," *Journal of Intelligent & Robotic Systems*, vol. 66, no. 1, pp. 75–109, 2012.

[44] K. M. Lynch and F. C. Park, *Motion Planning*. Cambridge University Press, 2017, pp. 353–402.

[45] P. Sudhakara and V. Ganapathy, "Trajectory planning of a mobile robot using enhanced a-star algorithm," *Indian Journal of Science and Technology*, vol. 9, pp. 1–10, 2018.

[46] M. Boccadoro, F. Martinelli, and S. Pagnottelli, "Constrained and quantized kalman filtering for an rfid robot localization problem," *Autonomous Robots*, vol. 29, no. 3, pp. 235–251, 2010.

[47] F. Serratosa, R. Alquézar, and N. Amézquita, "A probabilistic integrated object recognition and tracking framework," *Expert Systems With Applications*, vol. 39, no. 8, pp. 7302–7318, 2012.

[48] H. Strasdat, J. Montiel, and A. J. Davison, “Visual slam: Why filter?” *Image and Vision Computing*, vol. 30, no. 2, pp. 65–77, 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0262885612000248

[49] L. Payá, L. Fernández, A. Gil, and O. Reinoso, “Map building and monte carlo localization using global appearance of omnidirectional images,” *Sensors*, vol. 10, no. 12, pp. 11 468–11 497, 2010. [Online]. Available: https://www.mdpi.com/1424-8220/10/12/11468

[50] F. Moreno, J. Blanco, and J. Gonzalez, “Stereo vision specific models for particle filter-based slam,” *Robotics and Autonomous Systems*, vol. 57, no. 9, pp. 955–970, 2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889009000438

[51] K. Berns. Autonomous mobile robots. Technische Universität Kaiserslautern. [Online]. Available: agrosy.informatik.uni-kl.de/en/teaching/offered-courses/vorlesungen/autonome-mobile-roboter/autonomous-mobile-robots-20/

[52] E. Khanmirza, M. Haghbeigi, M. Nazarahari, and S. Doostie, “Comparative study of deterministic and probabilistic mobile robot path planning algorithms,” *International Conference on Robotics and Mechatronics (IcRoM 2017)*, pp. 534–539, 2017.

[53] M. Samadi and M. F. Othman, “Global path planning for autonomous mobile robot using genetic algorithm,” *International Conference on Signal-Image Technology and Internet-Based Systems*, pp. 726–730, 2013.

[54] H. Yang, J. Zhang, and S. Chen, “Stereo camera based real-time local path-planning for mobile robots,” *Chinese Conference on Pattern Recognition*, pp. 345–354, 2014.

[55] K. E. Lydla and L. M. Steven, *Motion Planning*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 139–158.

[56] L. M. LaValle and J. K. Kuffner, “Rapidly-exploring random trees: Progress and prospects,” *Algorithmic and computational robotics: New directions*, 2000.

[57] A. M. Varghese and V. R. Jisha, “Motion planning and control of an autonomous mobile robots,” *International CET Conference on Control, Communication, and Computing (IC4)*, pp. 17–21, 2018.

[58] O. Khatib, “Real-time obstacle avoidance for manipulators and mobile robots,” *International Conference on Robotics and Automation*, pp. 500–505, 1985.