Chapter 1

Related work

1.1 Introduction

Lorem ipsum dolor sit amet, consectetur adipisci elit, sed eiusmod tempor incidunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrum exercitationem ullam corporis suscipit laboriosam, nisi ut aliquid ex ea commodi consequatur. Quis aute iure reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint obcaecat cupiditat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

1.2 Related work in scan matching

Given two sets of 2D data, the *scan matching* problem consists in finding a traslation T and a rotation R_ϕ that maximize the overlapping between the two sets. The scan matching algorithms usually work using data furnished by 2D range sensors and differ in their behaviour depending on the availability of an initial guess of solution.
A further classification of scan matching algorithms can be made on the basis of their assumption about the presence of noise in sensor's measurements and of features in surrounding environment. Feature-based algorithms, in particular, gained a great success because of their computational effectiveness, but it must be underlined that the extraction of features may produce a loss of information.

When ambient contains features that are invariant to rotation and traslation, they can be easily extracted from scans and can be used to find a solution in a linear time.
If the extraction of features is not simple because of the structure of environment, then it is possible to use algorithms belonging to ICP family, which base their operation on a two step procedure:

- find an initial heuristic set of correspondences between points in two scans
- find a roto-traslation that more or less satisfies the set of correspondences

The two steps are repeated until the error falls below a certain threshold. The convergence is possible if scan are produced in two robot's positions that are close to each other.

1.2.1 Scan matching using the Hough transform

In a 2009 article by Censi, Iocchi and Grisetti, the problem of scan matching is afforded by the use of *Hough transform (HT)* and is, then, moved to the Hough domain. If we have an *input space* $\boldsymbol{\mathcal{S}}$, the HT maps an input $i(\mathbf{s})$ (with $\mathbf{s} \in \boldsymbol{\mathcal{S}}$) to an output function $\mathrm{HT}\{i\}(\mathbf{p})$, with $\mathbf{p}$ belonging to a *parameter space* $\boldsymbol{\mathcal{P}}$.
We can define the HT of input $i(\mathbf{s})$ as

$$\mathrm{HT}[\boldsymbol{\mathcal{F}}, i](\mathbf{p}) = \int_{\boldsymbol{\mathcal{F}}_\mathbf{p}} i(\mathbf{s}) d\mathbf{s} \tag{1.1}$$

A possible choice for $\boldsymbol{\mathcal{P}}$ is the the set of representations of lines in $\mathbb{R}^2$. If we recall the polar representation of lines

$$x\cos\theta + y\sin\theta = \rho \tag{1.2}$$

in which θ expresses the direction of line's normal vectors and ρ its distance from origin, we can select a family of sets to apply HT:

$$\boldsymbol{\mathcal{F}}_{(\theta,\rho)} = \{(x,y) | x\cos\theta + \sin\theta = \rho\} \tag{1.3}$$

On the other side, the input space is composed by points $P = \{p_j\}$, as they are returned by a range sensor. So we have that

$$\mathbf{s} := (x,y) \in \boldsymbol{\mathcal{S}} := \mathbb{R}^2 \tag{1.4}$$

and

$$i(\mathbf{s}) = \sum_j \delta(\mathbf{s} - \mathbf{p_j}) \tag{1.5}$$

where δ is Dirac's impulse distribution.

HT has two fundamental properties:

- $\mathrm{HT}(\theta, \rho)$ is 2π-periodic
- the two points (θ, ρ) and $(\theta + \pi, -\rho)$ in $\boldsymbol{\mathcal{P}}$ represent the same line in $\mathbb{R}^2$

Let's suppose that $i(\mathbf{s})$ and $i'(\mathbf{s})$ are two inputs, such that

$$i'(\mathbf{s}) = i(R_\phi \cdot \mathbf{s} + T) \tag{1.6}$$

Then, within parameter space $\boldsymbol{\mathcal{P}}$:

$$\mathrm{HT}'(\rho, \theta) = \mathrm{HT}(\theta + \phi, \rho + (\cos\theta \quad \sin\theta)T) \tag{1.7}$$

Computing the HT passes through a discrete approximation of it, called *Discrete Hough Transform (DHT)*, that allows to redefine our family of sets as:

$$\boldsymbol{\mathcal{D}}_{(\theta,\rho)} = \{(x,y) |\ \rho \leq x\cos\theta + y\sin\theta < \rho + \Delta\rho\} \tag{1.8}$$

For a complete definition of the scan matching algorithm based on HT, we need to define the so called *Hough spectrum*. For such purpose, we need a funcional g that is invariant to traslation. Theoretically speaking, this means that, given a function f:

$$f'(\tau) = f(\tau + \alpha) \Rightarrow g[f] = g[f'] \tag{1.9}$$

So, it is possible to define the Hough Spectrum as

$$\mathrm{HS}_g[i](\theta) := g[\mathrm{HT}[i](\theta, \cdot)] \tag{1.10}$$

An interesting property of spectrum is its *invariance*. If we define again two inputs $i(\mathbf{s})$ and $i'(\mathbf{s})$ as in (), the following statement is valid:

$$\mathrm{HS}_a[i](\theta) = \mathrm{HS}_a[i'](\theta + \phi). \tag{1.11}$$

This property, in case of the use of DHT in place of HT, holds only if the quantity $(\cos\theta \quad \sin\theta)T$ is a multiple of $\Delta\rho$.

The set of possible choices for g is wide and a natural one is the energy of the sequence in discrete domain:

$$g[f] = \sum_i f_i^2 \tag{1.12}$$

Since the property illustrated in () holds, it is possible to estimate ϕ by correlating the discrete spectra of sensor data $\mathrm{DHS}^S(\theta)$ and of reference data $\mathrm{DHS}^R(\theta)$. Then, local maxima of correlation are ordered; beginning with the greatest maximum, proceeding up to a chosen number, a new hypothesis for ϕ is formulated.
We can suppose that, if Φ is the domain in which we search for the value of ϕ, such domain can have an upper and lower bound, in such a way that $\Phi = [-\phi_U, \phi_U]$. If this assumption cannot be made, then it can be simply set $\Phi = [-\pi, \pi]$.
The correlation between the two spectra can be obtained using the relation

$$\mathrm{corr}_{DHS}(\phi) = \sum_{\theta \in \Theta} \mathrm{DHS}^S(\theta) \cdot \mathrm{DHS}^R(\theta - \phi) \tag{1.13}$$

The hypotheses $\phi_1, \ldots, \phi_k$ are retrieved from local maxima of correlation.

The scan matching problem is almost solved; what remains to be done is the estimation of traslation T. Let's assume $\phi = 0$ (this can be obtained by shifting the columns of DHT^S and DHT^R by the ϕ previously computed). We can choose an arbitrary $\theta = \hat{\theta}$. According to (), we can derive that

$$\mathrm{DHT}^R(\hat{\theta}, \rho) = \mathrm{DHT}^S(\hat{\theta}, \rho + (\cos\hat{\theta} \quad \sin\hat{\theta})T) \tag{1.14}$$

By correlating the columns of DHT^R and DHT^S, the projection of T onto the direction $\hat{\theta}$, $(\cos\hat{\theta} \quad \sin\hat{\theta})T = d(\hat{\theta})$, can be retrieved. Similarly to ϕ, a bound $[-|T|_U, |T|_U]$ for searching in the domain of $d(\hat{\theta})$ can be set.
Considering two different values for θ and the maximum of correlation, a linear system can be built to find T. Different pairs (θ_1, θ_2) furnish different results for T, that can be used as different hypotheses or combined to obtain an over-constrained system to be solved with least square estimation:

$$i = 1 \ldots n: \quad (\cos\theta_i \quad \sin\theta_i)T = d(\theta_1) \tag{1.15}$$

The elements of set $\{\theta_1, \ldots, \theta_n\}$ can be chosen among the maxima of sensor's DHS, i.e.

$$\{\theta_1, \ldots, \theta_n\} = \text{localMaxOf}\{\text{DHS}^S(\phi)\} \tag{1.16}$$

Finally, using the concepts described until this moment, it is possible to create a scan matching procedure, whose main steps can be summarized in the following:

- Compute DHT and DHS for reference and sensor data.
- Use local maxima of spectra cross-correlation to make hypotheses about ϕ.
- For each hypothesis produced at previous step, correlate columns of DHT to get linear constraints on T.
- Proceed following one of two possible paths:
  - combine linear constraints to get multiple hypotheses about T and order solutions on the basis of a likelihood function
  - produce a dense output by accumulating the results of correlation

1.2.2 Scan matching in polygonal environments

Many of the known matching methodologies require a good estimation of robot's initial pose which, when it is close to actual one, allows to limit the search space. The pose estimation and its update exploit a Gaussian model; thus, pose itself is accurately computed and updates can be calculated using Kalman filters. In particular, the extended Kalman filter models robot's pose as a Gaussian distribution $l(t) \sim N(\mu_l, \Sigma_l)$, using $\mu_l = (x, y, \alpha)^T$ as mean value and a 3×3 matrix for covariance Σ_L.
When the robot moves a distance δ and rotates by an angle θ, such motion can be described as $a \sim N((\delta, \theta)^T, \Sigma_a)$ and robot's pose is updated according to the following relations:

$$\begin{aligned} \mu_l &:= E(F(l,a)) = \begin{pmatrix} x + \delta\cos(\alpha) \\ y + \delta\sin(\alpha) \\ \alpha + \delta \end{pmatrix} \\ \Sigma_l &:= \nabla F_l \Sigma_l \nabla F_l^T + \nabla F_a \Sigma_a \nabla F_a^T \end{aligned} \tag{1.17}$$

where E is the expected value of function F and ∇F_l and ∇F_a are its Jacobians with respect to l and a.

From scan matching a pose update $s \sim N(\mu_s, \Sigma_s)$ is obtained, so the robot pose can be updated:

$$\begin{aligned} \mu_l &:= (\Sigma_l^{-1} + \Sigma_s^{-1})^{-1} \cdot (\Sigma_l^{'-1}\mu_l + \Sigma_s^{-1}\mu_s) \\ \Sigma_l &:= (\Sigma_l^{-1} + \Sigma_s^{-1})^{-1} \end{aligned} \tag{1.18}$$

Anyway, the success of Kalman filtering depends mainly on how the scan matching algorithm can estimate robot's pose. That is why, Gutmann, Weigel and Nebel, in 1999, proposed a new scan matching algorithm that exploits the polygonal structure of environment.
The procedure, known as *LineMatch*, is described by Algorithm 1. The main tasks performed by algorithm are extracting

Algorithm 1 LINEMATCH (M, S, P)

```
1:  Input : model lines M, scan lines S, pairs P
2:  Output : set of position hypotheses H
3:  if |P| = |S| then
4:      H := P
5:  else
6:      H := ∅
7:      s := SelectScanline(S, P)
8:     for all m ∈ M do
9:        if VerifyMatch(M, S, P ∪ {(m, s)}) then
10:           H := H ∪ {LINEMATCH(M, S, P ∪{(m, s)})}
11: return H
```

lines from current scan and comparing them with the ones contained in a *a priori* map. In particular, function *SelectScanline* picks a a new line to be matched, while *VerifyMatch* checks that a new pair *(m, s)* of matched lines is compatible with the pairs alreary accepted in previous iterations of procedure. Finally, the algorithm returns a set of accepted pairs,

that can be translated into positions from which robot may have taken scans. Proper constraints on lines' lengths and roto-traslations can allow algorithm to be more effective; this implies that it is not necessary to have a good estimate of robot's initial pose, since robot is able to recover from initial error.

Once the matching activities are completed, the most plausible position is selected using odometry (a closest neighborhood policy will fit for the purpose) and is used to update robot's pose, by Kalman filters. A good choice for initial mean and covariance values may be

$$\begin{aligned} \mu_l &:= \begin{pmatrix} 0 & 0 & 0 \end{pmatrix}^T \\ \Sigma_l &:= \begin{pmatrix} \infty & 0 & 0 \\ 0 & \infty & 0 \\ 0 & 0 & \infty \end{pmatrix} \end{aligned} \tag{1.19}$$

1.3 Selecting features to track

A scan matching algorithm, whatever its inner functionalities, can work properly only if the choice of features to extract has been made correctly, in order to maximize matching result.
Shi and Tomasi, introduced in 1994, a method for features extraction based on the concept of *dissimilarity*, a measure of changes occurred to features between first frame and current one. When dissimilarity is too high, features should be ignored.

Considering frames acquired by a moving camera, all patterns that can be extracted vary in terms of intensity. Such changes are included in the definition of *motion*:

$$I(x,y,t+\tau) = I(x-\xi(x,y,t,\tau), y-\eta(x,y,t,\tau)). \tag{1.20}$$

What previous equation states is that an image taken at the instant of time $t+\tau$ can be obtained from one taken at time t by applying a motion, whose entity $\delta = (\xi, \eta)$ is called *displacement* of point $\boldsymbol{x}$= *(x, y)*.

If we speak in terms of windows, we can define the *affine motion field* as

$$\delta = D\boldsymbol{x} + d \tag{1.21}$$

with

$$D = \begin{pmatrix} d_{xx} & d_{xy} \\ d_{yx} & d_{yy} \end{pmatrix} \tag{1.22}$$

a deformation matrix; $\boldsymbol{d}$ is the traslation of a feature window's center.
By measuring the coordinates $\boldsymbol{x}$ with respect to the center of window, we get that a point $\boldsymbol{x}$ in the first image I moves to a point $A\boldsymbol{x}+d$ in second image J, with $A = \mathbf{1} + D$ and $\mathbf{1}$ is a 2×2 identity matrix.
Thus, the tracking problem between two images I and J can be formulated in terms of finding the six parameters contained in D and $\boldsymbol{d}$. A model that uses small windows is preferable, since it causes less problems in terms of deph discontinuity; in such condition, the matrix D, that is harder to estimate because of the small variation of motion, can be assumed equal to zero, so

$$\delta = \boldsymbol{d}.$$

The equation () is not satisfied completely, then finding the six unknown motion parameters means to find A and $\boldsymbol{d}$ that minimize dissimilarity

$$\varepsilon = \int \int_W [J(A\boldsymbol{x} + \boldsymbol{d} - I(\boldsymbol{x})]^2 w(\boldsymbol{x}) d\boldsymbol{x} \tag{1.23}$$

with W the feature window and $w(\boldsymbol{x})$ a weighting function (in the simpleste case, it can be set to 1. In the other cases, it can be a Gaussian-like function).

When we have a simple traslation, matrix A must coincide with identity matrix. To minimize the value in equation (), it is possible to diffirentiate with respect to unknown parameters in D and $\boldsymbol{d}$ and then set result to zero. The obtained system can linearized by a truncation in Taylor expansion:

$$J(A\boldsymbol{x} + \boldsymbol{d}) = J(\boldsymbol{x}) + g^T(\boldsymbol{u}). \tag{1.24}$$

This leads to a 6×6 system:

$$T\boldsymbol{z} = \boldsymbol{a} \tag{1.25}$$

with

$$\boldsymbol{z}^T = [d_{xx} \quad d_{yx} \quad d_{xy} \quad d_{yy} \quad d_x \quad d_y]$$

The system is completed by an error vector

$$\boldsymbol{a} = \int\int_W [I(\boldsymbol{x}) - J(\boldsymbol{x})] \begin{bmatrix} xg_x \\ xg_y \\ yg_x \\ yg_y \\ g_x \\ g_y \end{bmatrix} w \, d\boldsymbol{x} \tag{1.26}$$

and a 6×6 matrix

$$T = \int\int_W \begin{bmatrix} U & V \\ V^T & Z \end{bmatrix} w \, d\boldsymbol{x} \tag{1.27}$$

with

$$U = \begin{bmatrix} x^2 g_x^2 & x^2 g_x g_y & xyg_x^2 & xyg_x g_y \\ x^2 g_x g_y & x^2 g_y^2 & xyg_x g_y & xyg_y^2 \\ xyg_x^2 & xyg_x g_y & y^2 g_x^2 & y^2 g_x g_y \\ xyg_x g_y & xyg_y^2 & y^2 g_x g_y & y^2 g_y^2 \end{bmatrix}$$

$$V^T = \begin{bmatrix} xg_x^2 & xg_x g_y & yg_x^2 & yg_x g_y \\ xg_x g_y & xg_y^2 & yg_x g_y & yg_y^2 \end{bmatrix} \tag{1.28}$$

$$Z = \begin{bmatrix} g_x^2 & g_x g_y \\ g_x g_y & g_y^2 \end{bmatrix}$$

To be sure to have a good tracking, it is necessary to have a small motion between adjacent frames; this causes D to be small, too, so it can be set to zero. In this situation, determining deformation parameters can be pointless and dangerous as well, since it may lead to solutions with tiny displacements. Indeed, D and $\boldsymbol{d}$ are linked to each other through matrix V and an error in D produces an error in $\boldsymbol{d}$, too.
Then, when only $\boldsymbol{d}$ needs to be sought, it is sufficient to solve smaller system

$$Z\boldsymbol{d} = \boldsymbol{e} \tag{1.29}$$

with $\boldsymbol{e}$ containing the last two elements in $\boldsymbol{a}$.
When the two frames to be compared are the first and current ones, a simple traslation model is not suitable and full system should be solved.

Considering a certain frame, not all its parts contain motion information and not all features are good for tracking, so a selection policy is necessary. A window can be tracked from frame to frame if the matrix Z is well-conditioned; thus, its eigenvalues must be large and cannot differ by too many orders of magnitude. Two small eigenvalues denote a constant intensity within window; a small and large eigenvalues indicate a unidirectional texture pattern; two large eigenvalues may correspond to corners and other easily trackable features.
In a few words, a window is accepted if, given the two eigenvalues λ_1 and λ_2 of Z we have

$$\min(\lambda_1, \lambda_2) > \lambda \tag{1.30}$$

with λ a chosen threshold.

Chapter 2

Basic concepts

2.1 Introduction

This chapter introduces to the main concepts on which this thesis is based. First of all, we will concentrate ourselves on describing the concept of sensors and their importance in self-perception of robots. Sensors, indeed, are very useful to allow a robot to localize itself and have an idea of surrounding environment; anyway, to make all this possible, it is necessary to design them in such a way to satisfy a series of requirements. Sensors, although their immense utility, furnish noisy measurements with which localization algorithms must deal. In this chapter, we will describe one particular kind of sensors, the ones based on the *time of flight (TOF)* method.

The main tecnique used by robots to estimate their current position and orientation is *odometry*, which is based on data returned by robot's encoders. We will analyse the two types of errors it introduces and how they have been experimentally measured.

Data furnished by sensors is often huge, so it is important to extract only relevant parts of it, in order to reduce time in computation. We will describe how such parts can be extracted and which properties we expect them to have.

Finally, we will propose a process based on least squares minimization, that allows to estimate a robot's state (in terms of position and orientation), exploiting the error between measured and expected state value.

2.2 Robot sensors

Robots can be considered, among human artefacts, as the ones closest to the concept of a living being, with their capabilities to sense world and change their own choices on the basis of what they perceive in the surrounding environment. Without a sensing system, robots could be just able to repeat the same task and wouldn't be able to adapt their decisions to an environment continously changing around them. Thanks to sensors, robots can operate in places that are too dangerous for humans and interact with each other.

When designing a sensor, there is a small set of requirements that should be kept into account:

- a good field of view, to satisfy the needings of the greatest number of applications
- a good minimum and maximum range of detection
- the largest accuracy and resolution
- a great ability to detect objects in the environment, dealing with phenomena such as reflection and noise due to ambient interference
- high update frequency, to have real time data at disposal as fast as possible
- generated data should be concise and easy to interpret
- low costs in designing, construction and maintenance
- low power consumption
- small size and weight

2.2.1 Range sensors

In many cases, a robot is required to build a map of its surrounding environment and for such goal a measurement of range between itself and obstacles is needed. There exists a class of range sensors widely used for such purpose based on the so called *time of flight (TOF)* method; an ultrasonic, RF or optical source generates an energy wave that propagates into ambient and the distance between robot and an object is computed as the product of the velocity of wave and the time required to travel the round-trip distance, according to relation

$$d = vt \tag{2.1}$$

where d is the round-trip distance, v is the speed of propagation of energy wave and t is the elapsed time. The time interval t in previous equations is the one that elapses while the energy wave reaches the obstacle and gets back to robot; thus, to retrieve the real range between robot and target, t must be divided by two.

The signal emitted by robot reaches an object and walks back on the same way to the robot, where it will be detected by a receiver, that can be located close to emitter or integrated with emitter itself. As it is evident in ( ), the range to an object can be easily retrieved and no further knowledge about object's nature is necessary.

Similarly to all kinds of sensors, even TOF systems are subject to measurements errors, which can be summarized in the following short list:

- **variations in the speed of propagation of emitted signal**
- **uncertainty in determining when reflected signal was detected**
  Such uncertainties depend on the fact that different surfaces reflect signal differently and this may cause time for detection of returned signal to rise. When a threshold on detection is established, more reflective objects are perceived as closer.
- **uncertanty in determining the exact round-trip time** t
  It is due to problems correlated to inner timing circuitry.
- **interaction between originated signal and surfaces**
  When source signal hits a surface, only part of it is reflected and perceived by the robot. The remaining part is propagated into ambient or passes through the hit object. Thus, it may happen that no signal is received by robot, particularly if emitting angle exceeds a certain threshold. The part of signal that is not sent back to robot, may be spread to envirnonment and perceived by another robot's sensors.

2.3 Odometry

Odometry is probably the most used method to estimate the position of a robot at any instant in time. The reasons of its success must be found in good accuracy, low cost and high sampling rates. Anyway, the way odometry computes robot's position (by integrating motion information across time) leads to errors; in particular, errors in estimating robot's orientation lead to large position errors, which grow proportionally with the distance afforded by robot.
Despite limitations due to errors, odometry gained a great success in mobile robots field, for at the least four reasons:

- other kinds of measurements can be itegrated to obtain a better accuracy
- in cases in whick landmarks are used for helping to estimate robot's position, their number can be smaller, thanks to odometry's presence
- odometry allows to assume that robot is stuck in a certain pose for enough time to perceive all the landmarks in a given area and compute possible matchings with landmarks detected previously
- in some circumstances, because of the absence of external references (e.g. landmarks) or the impossibility to place any of them in a hostile environment, odometry can be the only way to estimate robot's position

Odometry exploits the measurements obtained from encoders mounted on robot's wheels to extimate its new pose. Let's suppose that, in a certain time interval, robot's wheels have pulses N_L and N_R. Let's denote with c_m the conversion factor that allows to convert pulses into a linear displacement. We can suppose that

$$c_m = \frac{\pi D_n}{nC_e} \tag{2.2}$$

where:

- D_m is the nominal wheel's diameter
- C_e is the encoder resolution (pulses per revolution)

- n is the reduction gear ratio between motor and drive wheel

The incremental linear displacements $\Delta U_{L,i}$ and $\Delta U_{R,i}$ for the two wheels can be computed as:

$$\begin{aligned} \Delta U_{L,i} &= c_m N_L \\ \Delta U_{R,i} &= c_m N_R \end{aligned} \tag{2.3}$$

We can now compute the displacement for robot's centerpoint ΔU_i (that will be taken as a reference for robot's position) and change in orientation $\Delta\theta_i$:

$$\begin{aligned} \Delta U_i &= \frac{\Delta U_R + \Delta U_L}{2} \\ \Delta\theta_i &= \frac{\Delta U_R - \Delta U_L}{b} \end{aligned} \tag{2.4}$$

where b is the distance between the two points where wheels touch the ground.

Thus, the new orientation of robot at time i is:

$$\theta_i = \theta_{i-1} + \Delta\theta_i \tag{2.5}$$

The new position, at the same instant of time, of its centerpoint will be:

$$\begin{aligned} x_i &= x_{i-1} + \Delta U_i \cos\theta_i \\ y_i &= y_{i-1} + \Delta U_i \sin\theta_i \end{aligned} \tag{2.6}$$

2.3.1 Systematic and Non-Systematic errors in odometry

As explained previously, the odometry exploits three simple equations to estimate current position of robot. All of them, are based on the assumption that any encoder's measurement can be translated into a linear diplacement. Anyway, such assumption may lose its validity in certain circumstances; for instance, if a wheel slips, its encoder will register a measurement that is converted in a linear displacement, even if robot didn't move actually.
Slippage is only one of the possible causes of errors in odometry; they can me summarized and divided into the following two categories:

Systematic errors

- wheels with different diameters
- wheelbase different from nominal one
- wheels misalignment
- finite encoder resolution
- finite encoder sampling rate

Non-Systematic errors

- travelling on uneven floors
- travelling over objects
- slippage

Both errors categories are very difficult to manage; the first ones, in fact, are constantly present and accumulate over time; the latter, are unpredictable and robot must be able to react promptly when they manifest.

Across time, algorithms managing robot's position have thought it as surrounded by an error ellipse, denoting an area where robot may be in a certain moment. Ellipses grow with distance travelled, until an absolute measurement of position reduces ellipse's size. This method based on ellipses can be used to contain systematic errors only, since non systematic ones are unpredictable.

Estimating correctly the extent of odometric errors avoids further problems such as a wrong calibration of mobile platforms. In 1995, Borenstein and Feng developed a method based on a model which considers two erros to be dominant:

- the error due to different wheels' diameters $E_d = \frac{D_R}{D_L}$, with D_R and D_L, respectively, the actual diameters of right and left wheel
- the error due to uncertainty about real wheelbase $E_b = \frac{b_{actual}}{b_{nominal}}$, with b the wheelbase of robot

2.3.2 Measuring Systematic errors

Before describing Borenstein and Feng's method to estimate systematic errors, it is useful to analyse Borenstein and Koren's method (1987).
Let's suppose that robot starts its path from an initial position $<x_0, y_0, \theta_0>$ and has to move on a 4x4 meter square path. The robot is programmed to return to its initial position at the end of the path, but because of systematic errors, this will not happen accurately and a series of errors will be accumulated (*return position errors*):

$$\begin{aligned} \varepsilon_x &= x_{abs} - x_{calc} \\ \varepsilon_y &= y_{abs} - y_{calc} \\ \varepsilon_\theta &= \theta_{abs} - \theta_{calc} \end{aligned} \tag{2.7}$$

We have denoted with the *abs* subscript the absolute position and orientation of the robot and with *calc* subscript the position and orientation of robot according to odometry.

This kind of experiment is not suitable for testing odometry for differential drive platforms. That is why Borenstein and Feng introduced their *bidirectional square-path* experiment. This time, the robot is required to walk the path both in clockwise and counterclockwise direction. The errors E_d and E_b could compensate each other when robot travelled on one direction only; now, travelling on both direction, the two errors sum up.

If we let the robot travel on both directions n times (usually, $n = 5$), at the end of each run robot will accumulate a pose error as explained in (). We can compute the center of gravity of these errors according to the following relations:

$$\begin{aligned} x_{cg}^{cw} &= \frac{1}{n}\sum_{i=1}^{n} \varepsilon_{x,i}^{cw} \\ y_{cg}^{cw} &= \frac{1}{n}\sum_{i=1}^{n} \varepsilon_{y,i}^{cw} \\ x_{cg}^{ccw} &= \frac{1}{n}\sum_{i=1}^{n} \varepsilon_{x,i}^{ccw} \\ y_{cg}^{ccw} &= \frac{1}{n}\sum_{i=1}^{n} \varepsilon_{y,i}^{ccw} \end{aligned} \tag{2.8}$$

The two absolute offsets of centers of gravity from origin are:

$$\begin{aligned} r_{cg}^{cw} &= \sqrt{(x_{cg}^{cw})^2 + (y_{cg}^{cw})^2} \\ r_{cg}^{cw} &= \sqrt{(x_{cg}^{ccw})^2 + (y_{cg}^{ccw})^2} \end{aligned} \tag{2.9}$$

Finally, a measure of systematic odometric error can be obtained as

$$E_{sys} = \max(r_{cg}^{cw}, r_{cg}^{ccw}). \tag{2.10}$$

The orientation error ε_θ is not considered in E_{sys}, since each orientation error translates into a position error.

2.3.3 Measuring non-Systematic errors

The square path test that was used to estimate systematic error, can be implemented again for non-systematic case, adding to the path some artificial bumps. Since return errors are sensible to the positions of bumps, this time error ε_θ will be used. We can, then, compute an *average absolute orientation error*:

$$\varepsilon_{\theta,avg}^{nonsys} = \frac{1}{n}\sum_{i=1}^{n} |\varepsilon_{\theta,i,cw}^{nonsys} - \varepsilon_{\theta,i,cw}^{sys}| + \frac{1}{n}\sum_{i=1}^{n} |\varepsilon_{\theta,i,ccw}^{nonsys} - \varepsilon_{\theta,i,ccw}^{sys}| \tag{2.11}$$

The use of absolute values in previous equations is needed to avoid that two return orientation errors with opposite signs compensate each other. Thus, if after first run we have $\varepsilon_\theta = 1^\circ$ and after second one we have $\varepsilon_\theta = -1^\circ$, we will not erroneously derive that $\varepsilon_{avg}^{nonsys} = 0^\circ$.

2.4 Extracting features

In many different fields (machine learning and pattern recognition, just to mention a pair) the amount of data to manage (for example, the measurements returned by robot sensor) is too huge to be used in toto, so it becomes of great importance to extract only the necessary information, removing, moreover, all the redundancies that could make computation unfeasible. Thus., it is convenient to design processes that are able to extract information that is representative

of data and allows to overcome the problem of using data in its totality. Such processes perform what is called *features extraction*, i.e. they retrieve (and sometimes compute) elements (*features*) that are distinctive of data, with which it is easier to operate; features are required to be informative, discriminative and indipendent. For some uses, features are even required to mantain some properties across space and time.
Usually, features have a numerical nature, but sometimes they can present a different aspect (in some cases they can be strings of characters or histograms).

Before designing an extraction process, features' structure needs to be planned, in order to respect the goal of good representation of data and to obtain elements with which working will be easy and effective. The choice about structure and computation of features depends on their final use and algorithms' purposes; here is a short list of common choices made in different fields:

- histograms of the distribution of black pixels in characters recognition
- phonemes in speech recognition
- repetitive words or text structures in spam detection, inside email inboxes
- edges and corners in computer vision

As explained in previous section, data retrieved from odometry are often inaccurate and they can lead to errors in robot's position estimation; features, after their extraction from sensors' data, can add a precious information allowing to reduce the errors and gain a better estimate of robot position in time. Indeed, each feature carries information about its reciprocal position with respect to robot and it can be used to improve localization methods
It is necessary to underline that even a well designed process of extraction can return a number of features that is too huge to be handled; so, it becomes crucial, in such cases, to be aware about which features are really necessary and which can be ignored. This mainly depends on the context in which robot operates, the perception system and the way features are represented.

2.4.1 Extracting lines as features

When receiving data from a ranger sensor (for instance, a laser range), one needs to operate with pairs (ρ_i, θ_i), which denote the position of i-th point in space, in polar coordinates, in robot's reference frame. Retrieving features from such information can be performed in two different ways:

- use a subset of points as features
- extract lines from the sequence of points, representing the part of environment seen by sensor during last scan

For the purpose of this thesis, we will consider the second option.

Across the years, many algorithms have been developed for extracting lines, starting from a laser's sequence of points, each with its pros and cons. Here, we will describe the so-called *split and merge algorithm*, which operates using a divide et impera logic, starting with an initial set of points, progressively divided into subsets (on the basis of a check condition), until satisfying the conditions for determining the existence of a line (the subset under inspection has not enough points to be divided into two subsets and/or no point satisfies check condition to proceed to another division). The main steps of procedure are described by Algorithm 1. The second step of algorithm can be performed using any fitting process, even

Algorithm 2 Split and merge algorithm

1: Initial: put all points in a set s_1 and insert s_1 into an initially empty list $\mathcal{L}$
2: Get a set s_i from $\mathcal{L}$ and fit a line through its points
3: Find point P having the greatest distance d_P from line
4: If d_P is less than a threshold, go back to step 2
5: Otherwise, split s_i in P into two subsets s_{i1} and s_{i2}. Replace s_i with s_{i1} and s_{i2} in $\mathcal{L}$. Go back to step 2
6: If all sets in $\mathcal{L}$ have been checked, merge collinear lines

by connecting the first and last point in the sequence s_i. Each computed line is represented by its two extremes.

2.5 Least squares estimation

Let $\mathbf{x}$ be a vector variable indicating the state of the system and let's suppose to have a series of functions

$$\{f_i(\mathbf{x})\}_{i=1:n}$$

which, given $\mathbf{x}$, predict its measurement.
Let's suppose, moreover, to have a series of real measurements $\mathbf{z_i}$. The goal is to extimate the state $\mathbf{x}$ that best fits the

measurements $\mathbf{z_{i:n}}$.
We can, then, determine an error between each real measurement and the relative predicted one:

$$\mathbf{e}_i = \mathbf{z}_i - f_i(\mathbf{x}) \tag{2.12}$$

This error is assumed to be zero mean and normally distributed with information matrix Ω_i.
If we compute the squared form of error, we notice that it is a scalar and depends only on the state of the system:

$$e_i(\mathbf{x}) = \mathbf{e}_i(\mathbf{x})^T \Omega_i \mathbf{e}_i(\mathbf{x}) \tag{2.13}$$

Then, the problem of finding the state that best fits the real measurements can be reduced to find the state $\mathbf{x}^*$ such that

$$\begin{aligned} \mathbf{x*} &= \underset{\mathbf{x}}{\operatorname{argmin}} \; F(\mathbf{x}) \\ &= \underset{\mathbf{x}}{\operatorname{argmin}} \sum_i e_i(\mathbf{x}) \\ &= \underset{\mathbf{x}}{\operatorname{argmin}} \sum_i \mathbf{e}_i^T(\mathbf{x}) \Omega_i \mathbf{e}_i(\mathbf{x}) \end{aligned} \tag{2.14}$$

A general solution to the problem is to derive the global error function and find its nulls, but this would result in a complex solution which doesn't admit closed forms. So, proceeding with numerical approaches is a more practical path.

If a good initial guess is at disposal and the measurement functions are smooth in the neighborhood of minima, we can solve problem using iterative (local) linearizations. The iterative process is composed of three steps:

- linearize the problem around current initial guess
- solve a linear system
- compute a set of increments to be applied to current estimate in order to get closer to minima

The first step is linearizing the problem and this can be done using a Taylor expansion:

$$\begin{aligned} \mathbf{e}_i(\mathbf{x}+\Delta\mathbf{x}) &\simeq \mathbf{e}_i(\mathbf{x}) + \frac{\partial \mathbf{e}_i}{\partial \mathbf{x}} \Delta\mathbf{x} \\ &= \mathbf{e}_i + \mathbf{J}_i \Delta\mathbf{x} \end{aligned} \tag{2.15}$$

We can now place the Taylor expansion in the squared error:

$$\begin{aligned} & e_i(\mathbf{x}+\Delta\mathbf{x}) \simeq \\ & \mathbf{e}_i^T \Omega_i \mathbf{e}_i + \mathbf{e}_i^T \Omega_i \mathbf{J}_i \Delta\mathbf{x} + \Delta\mathbf{x}^T \mathbf{J}_i^T \Omega_i \mathbf{e}_i + \Delta\mathbf{x}^T \mathbf{J}_i^T \Omega_i \mathbf{J}_i \Delta\mathbf{x} \\ & = \underbrace{\mathbf{e}_i \Omega_i \mathbf{e}_i}_{c_i} + 2 \underbrace{\mathbf{e}_i^T \Omega_i \mathbf{J}_i}_{b_i^T} \Delta\mathbf{x} + \Delta\mathbf{x}^T \underbrace{\mathbf{J}_i^T \Omega_i \mathbf{J}_i}_{\mathbf{H}_i} \Delta\mathbf{x} \\ & = c_i + 2\mathbf{b}_i^T \Delta\mathbf{x} + \Delta\mathbf{x}^T \mathbf{H}_i \Delta\mathbf{x} \end{aligned} \tag{2.16}$$

The global error is the sum of the squared errors, so, from (3) and (5), we have:

$$\begin{aligned} F(\mathbf{x}+\Delta\mathbf{x}) &\simeq \sum_i (c_i + 2\mathbf{b}_i^T \Delta\mathbf{x} + \Delta\mathbf{x}^T \mathbf{H}_i \Delta\mathbf{x}) \\ &= \sum_i c_i + 2(\sum_i \mathbf{b}_i^T)\Delta\mathbf{x} + \Delta\mathbf{x}^T (\sum_i \mathbf{H}_i)\Delta\mathbf{x} \end{aligned} \tag{2.17}$$

where

$$\mathbf{b}^T = \sum_i \mathbf{e}_i^T \Omega_i \mathbf{J}_i \tag{2.18}$$

$$\mathbf{H} = \sum_i \mathbf{J}_i^T \Omega_i \mathbf{J}_i \tag{2.19}$$

The expression of global error we have just obtained is quadratic in $\Delta\mathbf{x}$ and can be minimized; for this purpose, we can compute its derivative with repect to $\Delta\mathbf{x}$ in the neighborhood of current solution $\mathbf{x}$:

$$\frac{\partial F(\mathbf{x}+\Delta\mathbf{x})}{\partial \Delta\mathbf{x}} \simeq 2\mathbf{b} + 2\mathbf{H}\Delta\mathbf{x} \tag{2.20}$$

with the optimal solution reached for

$$\Delta\mathbf{x*} = -\mathbf{H}^{-1}\mathbf{b} \tag{2.21}$$

Thus, an iterative algorithm can be designed; it will execute the following steps:

- linearize the system around current guess $\mathbf{x}$ and compute $\mathbf{e}_i(\mathbf{x}+\Delta\mathbf{x})$ as described in () and ()
- compute $\mathbf{b}^T$ and $\mathbf{H}$ as described in () and ()
- solve the system to get a new optimal increment $\Delta\mathbf{x}$*, as indicated in ()
- update the previous estimate: $\mathbf{x} \leftarrow \mathbf{x}+\Delta\mathbf{x}$*