ELE6209A - Navigation Systems

Project Report

Ahmad zaydan, 1959838
Aurey tsemo, 1701797

presented to
Jerome Le Ny

April 30, 2022

Contents

List of Figures

1 Introduction

To move in a space in complete safety requires knowledge of the whole environment. More precisely, the reasonably precise knowledge of the position and the shape of the obstacles present in the environment allows to avoid them. For a robot, the knowledge of the environment allows to better plan its movements. In the case where the environment is unknown, it becomes difficult for a robot to move safely. A solution to this problem would be to allow the robot to estimate the shape and position of obstacles around it in order to avoid them. This means mapping and locating obstacles in an environment. Since these two problems cannot be dissociated, it is a problem of simultaneous localization and mapping (SLAM: Simultaneous Localisation And Mapping). The problem to be solved would be simpler if it was only a question of identifying an obstacle with a simple shape. But in our case, the space can contain several obstacles of different shapes. In order to map the space, the robot is equipped with a laser range finder (LRF). The measurements acquired by the LRF range finder will allow to estimate the shape (or a part of it) of an obstacle, as well as its location. The series of consecutive measurements from several LRF scans corresponding to an obstacle is called an aggregate. The algorithm developed during this project allows to identify the shape and the location of an obstacle by least squares and to segment the received measurements into aggregates (in case several obstacles have been scanned).

In the context of solving SLAM problems, the processing of data collected from various sensors is an aspect critical to the system's performance. This work based on the memoirs of Sophie Chaudonneret - "*Segmentation, localisation et cartographie avec primitives géométriques 2D*" [1] - implements feature extraction via non-linear least-squares from LIDAR scans using object segmentation by dual Kalman filters.

In this report, a description of the problem to be solved will be presented. Then will follow the algorithm of estimation of the shape of the obstacle by the least squares and finally the algorithm of segmentation of the measurements by a double extended Kalman filter. Results for these implementations are then presented and discussed.

2 System description

2.1 Preliminary analysis

We present once more an overview of the system before diving into more details. The data received from the LRF consists of planar-range (2D) data. We suppose that the position and orientation of the robot in the 2D environment are known and represented as follows:

$$\mathbf{x}^p(x) = [x^p(t), y^p(t), \phi^p(t)]$$

Each scan from the LRF consists of a constant number I of scanner-object distance measurements $\tilde{d}_i^k$, scans are received at a known time interval t_k and classified by low to high scan angle values θ_i^k. We model $\tilde{d}_i^k$ as a stochastic variable which includes the white noise σ_d^2. This is illustrated in the following figure:

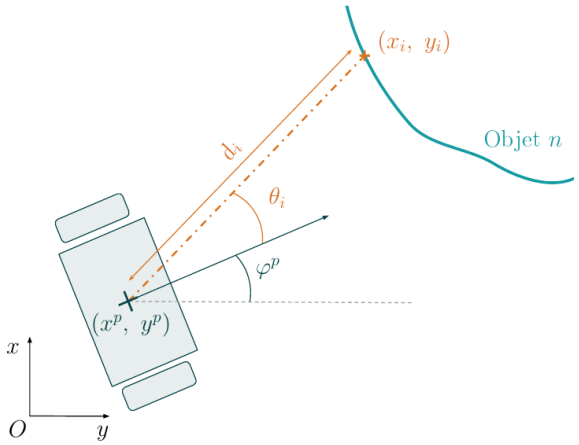


Figure 1: Representation of the system [1]

The identification and segregation of objects will be done via a fusion algorithm that tries to match point aggregates with super-ellipses. Super-ellipses are parameterized by the following pseudo function:

$$F_s(\mathbf{x}_i, \mathbf{p}) = \left(\left(\frac{(x - x_c)\cos\phi + (y - y_c)\sin\phi}{a} \right)^{\frac{2}{\epsilon}} + \left(\frac{(x - x_c)\sin\phi + (y - y_c)\cos\phi}{b} \right)^{\frac{2}{\epsilon}} \right)^{\epsilon}$$

where (x_c, y_c) is the position of the center of the super-ellipse, ϕ the orientation of the ellipse, (a, b) the size of the ellipse and ϵ the shape of the ellipse.

The fusion algorithm does segmentation using a double EKF architecture. Identification is done using a modified least-squares method. The two cascading filters are considered to be "flexible" and "strict". This corresponds to their propensity for accepting different ranges of values. This architecture allows easier initialization of new aggregates. The first "flexible" EKF defines if new values are outliers or correspond to a new entity. The second "strict" filter refines the model and detects interruptions. An interruption is detected when the "strict" criterion is not met, the aggregate is sealed and stored and the next values are passed to the "flexible" filter. Aggregates are initialized using a least-squares approximation method in order to obtain initial $\mathbf{p}$ parameter values.

2.2 Detailed analysis

2.2.1 Fitting

The modified nonlinear least-squares method is used to solve the problem of fitting an aggregate to the previously defined implicit functions. The basis is an iterative Gauss-Newton gradient descent as is often used for non-linear functions. The first variation is using the Levenberg-Marquardt which is a trust region method allowing for dynamically adjusting the step size of each iteration according to the performance of the cost function [2]. The initial associated cost function is therefore defined as:

$$\mathcal{F}_s = \left(\left(\frac{|\cos(\phi)\ (x - x_c) + \sin(\phi)\ (y - y_c)|}{|a|} \right)^{2/\varepsilon} + \left(\frac{|\cos(\phi)\ (y - y_c) - \sin(\phi)\ (x - x_c)|}{|b|} \right)^{2/\varepsilon} \right)^{\varepsilon} \tag{1}$$

with the following cost function initially defined as:

$$\Phi(\mathbf{x}) = \frac{1}{2} \sum_{i}^{I} r_i^2 = \frac{1}{2} \sum_{i}^{I} (\mathcal{F}_s - 1)^2 \tag{2}$$

However this yields oversized ellipses while the preference in this context is to estimate the smallest ellipse possible. We therefore modify this function with a cost constraint on size as follows:

$$r_i = \sqrt{ab}\,(\mathcal{F}_s - 1) \tag{3}$$

With a moving robot filling a point cloud, we get the additional constraint on auto-occlusion. This constraint is added to the cost function under the G_Δ term as it is decomposed in multiple parts. The final cost function becomes:

$$r_i = \alpha_1 G_S + \alpha_2 G_F + \alpha_3 G_\Delta \tag{4}$$

with α predefined scalars and the terms defined as follows:

$$\begin{aligned} G_S &= ab \\ G_F &= \mathcal{F}_s \\ G_\Delta = \frac{1}{2}(1 + \tanh(\alpha_0 D))D &= [\cos(\phi^p + \theta_i) \quad sin(\phi^p + \theta_i)]^T \cdot \Delta F \end{aligned} \tag{5}$$

The trust region iterative Levenberg-Marquardt style algorithm requires initial values for the estimations. These initial values are quickly obtained using PCA (Principal Component Analysis). In simple terms, the basic relation between points is quantified by determining the principal axes in the point cloud. The covariance matrix M is computed then decomposed into singular values, yielding $M = VDV^T$. We therefore yield our initial estimations using these matrices with:

$$\begin{aligned} a_0 &= \sqrt{2D_{11}} \\ b_0 &= \sqrt{2D_{22}} \\ \phi_0 &= \arctan(V_{21}, V_{11}) \end{aligned} \tag{6}$$

PCA alone yields very good estimates for these three values when points all around the shape of the ellipse are given. In our context of partial visibility, these initial parameters allow our iterative algorithm to center on the real values.

For performance specifications, the aim is to reproduce the performance seen in the memoirs used as a reference for this work. This means that the least-squares algorithm used should return DOP and initial parameter estimates appropriate to the segmentation filter. For segmentation it comes down to being able to separate and assign distinct shapes successfully in various scenes as seen in the memoirs.

2.2.2 Segmentation

Segmentation in the double EKF architecture is tightly coupled with the estimation models. The covariance matrix Σ_0^+ associated with every state is calculated using the DOP measure of uncertainty on the least-squares models calculating said states. In order to efficiently map out the environment, the system needs to include several models. In our case, we account for super-ellipses as defined previously. Belonging to a model or another is therefore determined via testing against the flexible criteria of all models such as there being as many flexible EKF initializations as there are models. Steps to determine belonging are therefore summarized as follows:

1. Test measurement against all models, if they all classify it as an outlier than go to the next step.
2. If a single model initializes an aggregate with the measurement then the coming values in the aggregate are only tested against this model according to the strict EKF criterion.
3. If multiple models are able to initialize an aggregate using the measurement then multiple strict EKF process the coming values in the aggregate. In that case, final belonging to a model is determined by selecting the one with minimal compatibility error.

Testing against models is coupled with testing against established maximum angle and distance criteria between q selected consecutive measurements. This fully defines measurement segmentation in the double EKF architecture as described in the following figure:

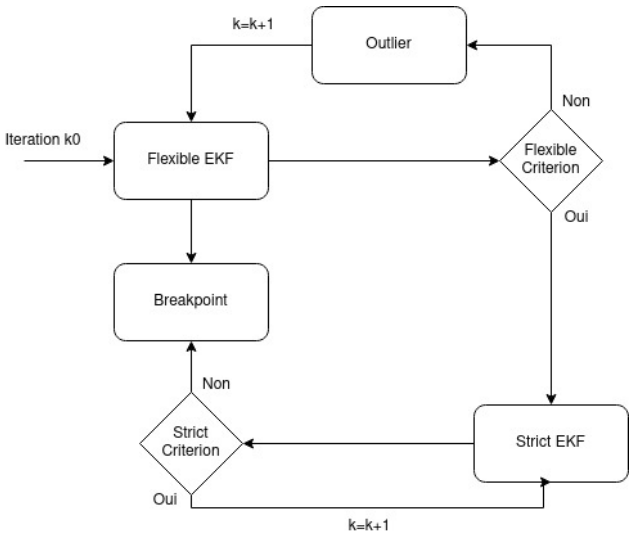


Figure 2: Dual EKF architecture

3 Work performed

All of the functions referenced in the coming sections can be found in the project repository: [https://github.com/ahzay/SEGMPARAM/].

3.1 Testing data collection

The first step was establishing a simulation environment for data collection. This was done in *CoppeliaSim* controlled using the Matlab API. A basic connection between these two programs was first established, commands tested and validated and required functionality was achieved using the online documentation [3]. Mainly the initialization of the environment, the configuration of the LIDAR device, the controlled movement of said device as well as streaming the data fetched by the simulated LIDAR. Super-ellipses then had to be generated and imported into the 3D environment. This is done in the *genSEMesh.m* function in the repository, by generating an *alphashape* using points, extracting the facets then writing them to the *stl* format. Once our super-quadrics based on the ellipse functions could be generated, they were then imported into test scenes. A program was fitted to connect with the API and place an arbitrary number of imported shapes at wanted locations then move the "robot" along a freehand drawn path by the user in order to collect data. The program can be found in the attached repository link as *simSetup.m* and a short demo video is available at the following link : [Coppelia environment with Matlab API].

3.2 Least-squares approximation

3.2.1 Least-squares calculations

With the basic environment set up the following step was implementing the least squares approximation. As stated previously, the chosen approximation layers different levels of complexity and is initially based upon the Gauss-Newton gradient method for non-linear functions. In order to facilitate implementation, we began by working with a simple single-variable case of optimizing the radius of a circle to fit noised data using Gauss-Newton and an explicitly defined circle function. With this implementation successful, we moved onto the same methods but for a non-centred circle, adding two variables to the optimization but still using the explicit function. In order to extend the tests and add more variables, attempts were made to use the Gauss-Newton method on an ellipse with 5 optimization variables, the methods could not be brought to converge. The implementations were then adapted to the trust region method, the structure inspired from [4]. As previously they were first implemented on simpler problems, notably a noised uncentered circle for three optimization variables and an implicit function, a noised centered ellipse for three optimization variables and an implicit function, a noised centered and rotated ellipse for four optimization variables and an implicit function and finally the noised uncentered and rotated ellipse for six optimization variables and an implicit function which is the cumulation and goal of this section.

The 6-variable least squares was thus implemented with a trust region method based on levenberg marquardt (L-M). The initial cost function was simply chosen as $r = \mathcal{F}_s$ (2). As stated in section 2.2.2, two more complex algorithms are implemented and tested with the cost function being the only difference between them. We first quickly define the algorithm and equations involved. After defining the initial parameters using PCA as described in the previous section and with measurements $\mathbf{m}$ at iteration k, we define the following functions:

$$\begin{aligned} J_k(\mathbf{m}, \mathbf{p}_k) &= \frac{\partial r}{\partial \mathbf{p}} \\ H_k(\mathbf{m}, \mathbf{p}_k) &= J_k^T J_k \\ g_k(\mathbf{m}, \mathbf{p}_k) &= J^T r \end{aligned} \tag{7}$$

At each iteration a vector d of the same size as $\mathbf{p}$ defines the step by which the parameters are modified before re-evaluating the cost function and determining if the modification reduced the cost. d is defined as follows using the gradient defined in (7), this allows d to push the parameters towards cost function minimas:

$$d_k = -uH_k^{-1}g_k \tag{8}$$

if the chosen push d is beneficial (next cost is less than the last) then it is retained, if not then it is discarded and the parameters are not updated. We control down the evolution of our algorithm using the factor u which is divided every time a d is retained and multiplied when it is discarded. This ensures that the algorithm slows down and does not miss a minima when moving in the correct direction. It also pushes it away from the parameters at the current iteration if they are deemed to be far from the minima. Using these conditions at every change of the parameters the J, r, H, g variables associated with their respective functions are updated. The complete algorithm used is therefore described as follows:

Algorithm 1 Identification algorithm

$M \leftarrow \mathrm{COV}(x, y)$ ▷ Initial values by PCA
$[V, D,] \leftarrow \mathrm{SVD}(M)$;
$[a_0, b_0, \phi_0] \leftarrow \mathrm{INIT}(V, D)$
$[x_c0, y_c0] \leftarrow \mathrm{MEAN}(x, y)$
$\epsilon_0 \leftarrow 10$
$B \leftarrow [a_0, b_0, \epsilon_0, \phi_0, x_c0, y_c0]$
$J \leftarrow \mathrm{J}(B)$ ▷ Now initialize algorithm
$r \leftarrow \mathrm{R}(B)$
$H \leftarrow J'J$ ▷ This is an approximation of the Hessian
$d \leftarrow [1; 1; 1; 1; 1; 1]$
$n \leftarrow 1$
while 1 **do**
  if $d' * d \leq 1e - 15$ **then** break
  end if
  while $n \leftarrow n + 1$ **do**
    $d \leftarrow -uH \setminus g$
    $r \leftarrow \mathrm{R}(B + d)$
    $cost(n) \leftarrow \mathrm{COST}(r)$; ▷ Evaluating the step
    if $cost(n) \leq cost(n - 1)$ **then** ▷ if d_k accepted mu is decreased
      $B \leftarrow B + d$
      $u \leftarrow u/5$
      $J \leftarrow \mathrm{J}(B)$
      $r \leftarrow \mathrm{R}(B)$
      $H \leftarrow J'J$
      $g \leftarrow J'r$
      break
    else
      $u \leftarrow 1.5u$
    end if
  end while
end while

3.2.2 Least-squares results and discussion

The concrete implementation of this algorithm highly depends on the chosen cost function. For the initial least-squares implementation, the partial derivatives were all analytically calculated using *Matlab*'s symbolic toolbox solver. When implementing the Kalman filter these same derivatives were re-calculated by hand and validated using the symbolic toolbox results. Implementations with all three cost functions (see eqs. 2,3 and 4) can be found in the linked repository under the names *NL-LS.mlx*, *NLAM-LS.mlx* and *NLCN-LS.mlx* respectively.

After testing, the chosen cost function is the one (3) rewritten below:

$$r_i = \sqrt{ab}\,(\mathcal{F}_s - 1)$$

as it gives an ideal middle ground between complexity and performance. The last cost function (eq. 3) which accounts for occlusion tended to diverge often (randomly with the gaussian noise) while the chosen function is simple and stable. Moreover, when using real data from the simulation, no occlusion was observed as the laser always picked up a slight curvature on the edges even if very close to the object, this is demonstrated in figure 3.a.

We present some results of the L-M least-squares using the chosen cost function on artificially created and noised ellipses in the figures 3.b,3.c and 3.d below. For the final segmentation function, we use manually calculated derivatives (detailed in the coming section) *drd*.m* as well as a final faster manual implementation of the least-squares *LM_ LS.m*

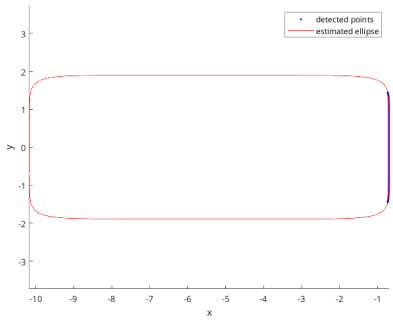


Figure 3.a

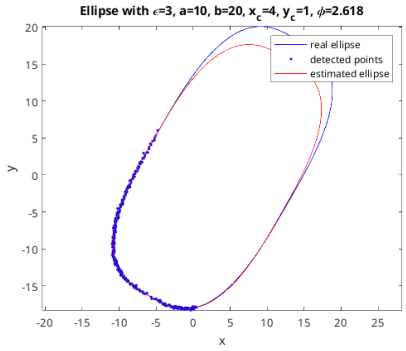


Figure 3.b

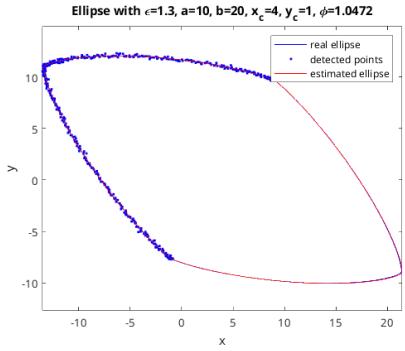


Figure 3.c

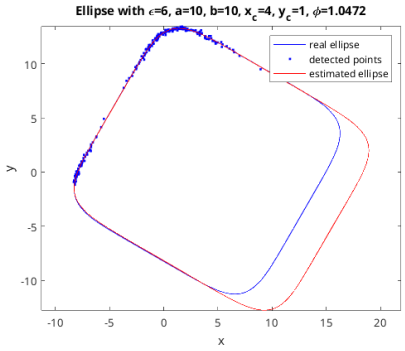


Figure 3.d

Figure 3: L-M least-squares with area minimization using real and artificial data

3.3 Segmentation

3.3.1 Segmentation calculations

As stated earlier, the separation of the measurements of the same object into aggregates in done using a dual extended Kalman filter architecture. One of the advantages of the LIDAR is that the data collected from the scans are consecutive and ordered according to the angle of capture. This way the points belonging to the same objects are automatically grouped together.

In order to segment the measurements obtained from the scan, it is necessary to determine the dynamic of the system. We remind that the super-ellipse implicit function $F\left(x^p, m_i, p\right)$ is represented by the expression (9):

$$F\left(x^p, m_i, p\right) = \left(\left(\frac{(x^p + d_i\cos(\theta_i + \psi^p) - x_c)\cos\phi + (y^p + d_i\sin(\theta_i + \psi^p) - y_c)\sin\phi}{a}\right)^{\frac{2}{\epsilon}} + \left(\frac{(x^p + d_i\cos(\theta_i + \psi^p) - x_c)\sin\phi - (y^p + d_i\sin(\theta_i + \psi^p) - y_c)\cos\phi}{b}\right)^{\frac{2}{\epsilon}}\right)^{\epsilon} - 1 \tag{9}$$

With $P = [x_c\ y_c\ \phi\ a\ b\ \epsilon]^T$ the state vector of the filter with:

- $(x_c,\ y_c)$ the super-ellipse coordinate center
- ϕ the super-ellipse orientation
- a, b the super-ellipse area
- ϵ the super-ellipse curvature

Since a scan is very fast, the environment is considered to be static in the duration it takes for the LIDAR to do a full scan. This eliminates any consideration we might have to take concerning movement in the environment. With these considerations, the cinematic model of the obstacles is described with the expression ((10)):

$$\dot{P} = 0_{6\times 1} \tag{10}$$

The state of the filter allows us to determine the value of the implicit function ((9)). This implicit function can therefore be considered the filter observation function.

$$z = F\left(x^p, m_i, p\right) \tag{11}$$

The inputs of the Kalman filter are the measurements $\left(\tilde{m}_i = \left[\tilde{d}_i\ \tilde{\theta}_i\right]^T\right)$. As they are noised, they can be modelled with (12) where $\nu_i = [\nu_{d,i}\ \nu_{\theta,i}]^T \sim \mathcal{N}(0, V_i)$ is the measurement noise of a LIDAR scan with covariance matrix $V_i = \begin{bmatrix} \sigma_d^2 & 0 \\ 0 & 0 \end{bmatrix}$.

$$\tilde{m}_i = m_i + \nu_i \tag{12}$$

The implicit function (9) can therefore be written:

$$F\left(x^p, m_i, p\right) = F\left(x^p, \tilde{m}_i - \nu_i, p\right) \tag{13}$$
$$= \left(\left(\frac{\left(x^p + (\tilde{d}_i - \nu_{d,i})\cos\left((\tilde{\theta}_i - \nu_{\theta,i}) + \psi^p\right) - x_c\right)\cos\phi + \left(y^p + (\tilde{d}_i - \nu_{d,i})\sin((\tilde{\theta}_i - \nu_{\theta,i}) + \psi^p) - y_c\right)\sin\phi}{a}\right)^{\frac{2}{\epsilon}} + \left(\frac{\left(x^p + (\tilde{d}_i - \nu_{d,i})\cos((\tilde{\theta}_i - \nu_{\theta,i}) + \psi^p) - x_c\right)\sin\phi - \left(y^p + (\tilde{d}_i - \nu_{d,i})\sin((\tilde{\theta}_i - \nu_{\theta,i}) + \psi^p) - y_c\right)\cos\phi}{b}\right)^{\frac{2}{\epsilon}}\right)^{\epsilon} - 1$$

The implicit function F being non-linear, it is therefore necessary to use an extended filter. In that case, the observation (11) must be linearized as follows:

$$z = H\left(x^p, m_i, \hat{p}\right)\hat{P} + J\left(x^p, m_i, \hat{p}\right)\rho \tag{14}$$

Where ρ is the measurement noise with covariance $R = \begin{bmatrix} \sigma_d^2 & 0 \\ 0 & 0 \end{bmatrix}$ and with

$$H\left(x^p, \tilde{m}_i, \hat{p}\right) = \left.\frac{\partial F}{\partial P}\right|_{(x^p, \tilde{m}_i, \hat{p})} \qquad J\left(x^p, \tilde{m}_i, \hat{p}\right) = \left.\frac{\partial F}{\partial \nu_i}\right|_{(x^p, \tilde{m}_i, \hat{p})} \tag{15}$$

We set:

$$F\left(x^p, \tilde{m}_i - \nu_i, p\right) = \left(\left(F_1\left(x^p, \tilde{m}_i - \nu_i, p\right)\right)^{\frac{2}{\epsilon}} + \left(F_2\left(x^p, \tilde{m}_i - \nu_i, p\right)\right)^{\frac{2}{\epsilon}}\right)^{\epsilon} - 1 \tag{16}$$

with

$$F_1\left(x^p, \tilde{m}_i - \nu_i, p\right) = \\ = \frac{\left(x^p + (\tilde{d}_i - \nu_{d,i})\cos\left((\tilde{\theta}_i - \nu_{\theta,i}) + \psi^p\right) - x_c\right)\cos\phi + \left(y^p + (\tilde{d}_i - \nu_{d,i})\sin((\tilde{\theta}_i - \nu_{\theta,i}) + \psi^p) - y_c\right)\sin\phi}{a}$$

$$F_2\left(x^p, \tilde{m}_i - \nu_i, p\right) = \\ \frac{\left(x^p + (\tilde{d}_i - \nu_{d,i})\cos((\tilde{\theta}_i - \nu_{\theta,i}) + \psi^p) - x_c\right)\sin\phi - \left(y^p + (\tilde{d}_i - \nu_{d,i})\sin((\tilde{\theta}_i - \nu_{\theta,i}) + \psi^p) - y_c\right)\cos\phi}{b}$$

and

$$\beta = \{x_c,\ y_c,\ \phi,\ a,\ b,\ \nu_{d,i},\ \nu_{\theta,i}\} \tag{17}$$

we obtain the general expression

$$\frac{\partial F\left(x^p, \tilde{m}_i - \nu_i, p\right)}{\partial \beta} = 2\left(\frac{\partial F_1}{\partial \beta}.F_1.\left(F_1^2\right)^{\frac{1}{\epsilon}-1} + \frac{\partial F_2}{\partial \beta}.F_2.\left(F_2^2\right)^{\frac{1}{\epsilon}-1}\right)^{\epsilon-1} \tag{18}$$

The derivatives of F_1 and F_2 as a function of various expressions of β (17), we obtain

$$\frac{\partial F_1}{\partial x_c} = -\frac{\cos(\phi)}{a} \qquad \frac{\partial F_2}{\partial x_c} = -\frac{\sin(\phi)}{b} \tag{19}$$

$$\frac{\partial F_1}{\partial y_c} = -\frac{\sin(\phi)}{a} \qquad \frac{\partial F_2}{\partial y_c} = \frac{\cos(\phi)}{b} \tag{20}$$

$$\frac{\partial F_1}{\partial \phi} = \frac{-\left(x^p + (\tilde{d}_i - \nu_{d,i})\cos\left((\tilde{\theta}_i - \nu_{\theta,i}) + \psi^p\right) - x_c\right)\sin\phi + \left(y^p + (\tilde{d}_i - \nu_{d,i})\sin((\tilde{\theta}_i - \nu_{\theta,i}) + \psi^p) - y_c\right)\cos\phi}{a} \tag{21}$$

$$\frac{\partial F_2}{\partial \phi} = \frac{\left(x^p + (\tilde{d}_i - \nu_{d,i})\cos\left((\tilde{\theta}_i - \nu_{\theta,i}) + \psi^p\right) - x_c\right)\cos\phi + \left(y^p + (\tilde{d}_i - \nu_{d,i})\sin((\tilde{\theta}_i - \nu_{\theta,i}) + \psi^p) - y_c\right)\sin\phi}{b}$$

$$\frac{\partial F_1}{\partial a} = -\frac{\left(x^p + (\tilde{d}_i - \nu_{d,i})\cos\left((\tilde{\theta}_i - \nu_{\theta,i}) + \psi^p\right) - x_c\right)\cos\phi + \left(y^p + (\tilde{d}_i - \nu_{d,i})\sin((\tilde{\theta}_i - \nu_{\theta,i}) + \psi^p) - y_c\right)\sin\phi}{a^2} \tag{22}$$

$$\frac{\partial F_2}{\partial a} = 0$$

$$\frac{\partial F_1}{\partial b} = 0 \tag{23}$$

$$\frac{\partial F_2}{\partial b} = -\frac{\left(x^p + (\tilde{d}_i - \nu_{d,i})\cos\left((\tilde{\theta}_i - \nu_{\theta,i}) + \psi^p\right) - x_c\right)\sin\phi - \left(y^p + (\tilde{d}_i - \nu_{d,i})\sin((\tilde{\theta}_i - \nu_{\theta,i}) + \psi^p) - y_c\right)\cos\phi}{b^2}$$

$$\frac{\partial F_1}{\partial \nu_{d,i}} = -\frac{\cos\left((\tilde{\theta}_i - \nu_{\theta,i}) + \psi^p\right)\cos\phi + \sin\left((\tilde{\theta}_i - \nu_{\theta,i}) + \psi^p\right)\sin\phi}{a} \tag{24}$$

$$\frac{\partial F_2}{\partial \nu_{d,i}} = \frac{-\cos\left((\tilde{\theta}_i - \nu_{\theta,i}) + \psi^p\right)\sin\phi + \sin\left((\tilde{\theta}_i - \nu_{\theta,i}) + \psi^p\right)\cos\phi}{b}$$

$$\frac{\partial F_1}{\partial \nu_{\theta,i}} = \frac{(\tilde{d}_i - \nu_{d,i})\sin\left((\tilde{\theta}_i - \nu_{\theta,i}) + \psi^p\right)\cos\phi - (\tilde{d}_i - \nu_{d,i})\cos\left((\tilde{\theta}_i - \nu_{\theta,i}) + \psi^p\right)\sin\phi}{a} \tag{25}$$

$$\frac{\partial F_2}{\partial \nu_{\theta,i}} = \frac{(\tilde{d}_i - \nu_{d,i})\cos\left((\tilde{\theta}_i - \nu_{\theta,i}) + \psi^p\right)\sin\phi + (\tilde{d}_i - \nu_{d,i})\cos\left((\tilde{\theta}_i - \nu_{\theta,i}) + \psi^p\right)\cos\phi}{b}$$

The expressions (15) can be rewritten in the form:

$$H\left(x^p, \tilde{m}_i, \hat{p}\right) = \left[\frac{\partial F}{\partial x_c} \quad \frac{\partial F}{\partial y_c} \quad \frac{\partial F}{\partial \phi} \quad \frac{\partial F}{\partial a} \quad \frac{\partial F}{\partial b} \quad \frac{\partial F}{\partial \epsilon}\right]_{\left(x^p, \tilde{m}_i, \hat{p}\right)} \tag{26}$$

$$J\left(x^p, \tilde{m}_i, \hat{p}\right) = \left[\frac{\partial F}{\partial \nu_{d,i}} \quad \frac{\partial F}{\partial \nu_{\theta,i}}\right]_{\left(x^p, \tilde{m}_i, \hat{p}\right)} \tag{27}$$

The discretisation of the equations (10) and (14) allow us to enable the following estimated model:

$$P_{i+1} = P_i + \omega_i \tag{28}$$
$$z_i = H_i P_i + J_i \rho_i \tag{29}$$

with

$$H_i = H\left(x^p, m_i, \hat{p}_k\right) \qquad J_i = J\left(x^p, m_i, \hat{p}_k\right)$$

and $\omega_i \sim \mathcal{N}(0, Q)$, a white gaussian noise term. The Kalman model depends on two principles: propagation and observation. The propagation model of the state is obtained with (28) while the observation model is obtained with (29). The propagation model is:

$$\hat{P}_i^- = \hat{P}_{i+1}^+$$
$$\Sigma_i^- = \Sigma_{i+1}^+ + Q$$

While the measurement update of the filter can be expressed as:

$$r_i = -F\left(x^p, \tilde{m}_i, \hat{P}_i^-\right)$$
$$S_i = H_i \Sigma_i^- H_i^T + J_i V_i J_i^T$$
$$K_i = \Sigma_i^- H_i^T S_i^{-1}$$
$$\hat{P}_i^+ = \hat{P}_i^- + K_i r_i$$
$$\Sigma_i^+ = \left(I - K_i H_i\right) \Sigma_i^-$$

where r_i is the innovation S_i and the uncertainty associated with the innovation

The initialisation of the filter is done via the least-squares parametrization described previously. The least squares method begins after a chosen number N of measurements from the laser. The Kalman algorithm is therefore the following, the implementation can be found in the function *project.m*. We have the expression

Algorithm 2 Extended Kalman filter algorithm

Initialisation of $\hat{P}_0^+$, Σ_0^+
for $i = 1 : I$ **do**
  $\hat{P}_i^- \leftarrow \hat{P}_{i+1}^+$
  $\Sigma_i^- \leftarrow \Sigma_{i+1}^+ + Q$
  $r_i \leftarrow -F\left(x^p, \tilde{m}_i, \hat{P}_i^-\right)$
  $S_i \leftarrow H_i \Sigma_i^- H_i^T + J_i V_i J_i^T$
  if $F\left(x_p, \tilde{m}_i, \hat{P}_i^-\right)^2 S_i^{-1} > \tau$ **then**
    Save the aggregate breakpoint
    Re-initialization of the state P_0^+,Σ_0^+
  else
    Update
    $K_i \leftarrow \Sigma_i^- H_i^T S_i^{-1}$
    $\hat{P}_i^+ \leftarrow \hat{P}_i^- + K_i r_i$
    $\Sigma_i^+ \leftarrow \left(I - K_i H_i\right) \Sigma_i^-$
  end if
end for

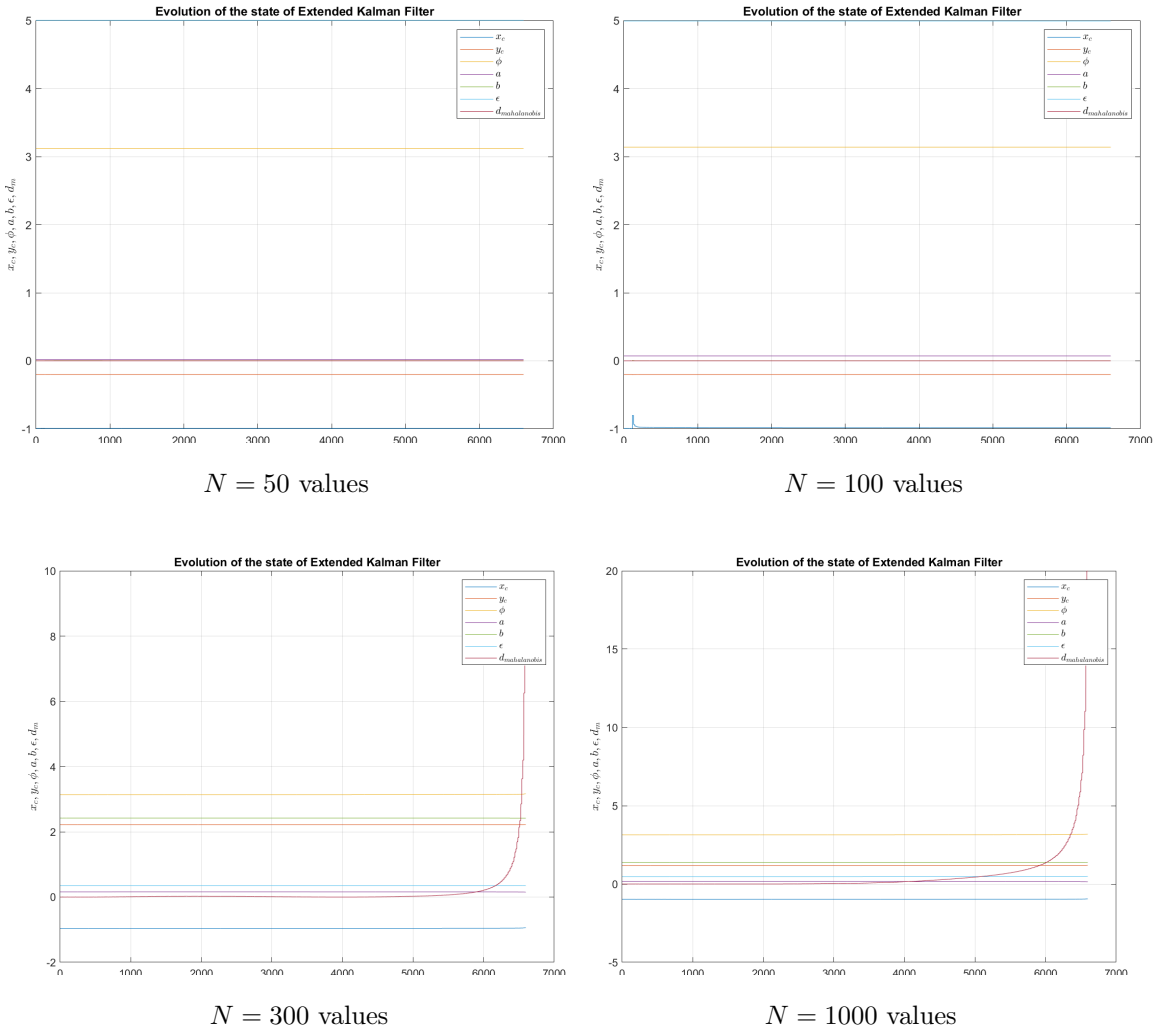
$F\left(x_p, \tilde{m}_i, \hat{P}_i^-\right)^2 S_i^{-1}$ called *the Mahalanobis distance* and its associated parameter τ. During the scan of the environment by the laser, when the laser meets a zone with no obstacle, the *the Mahalanobis distance* becomes very big. This allows us to associate big values of this distance with values that are either aberrant or break points (end of an aggregate).

In order to distinguish break points from aberrant values we use a dual EKF architecture briefly described previously. At each iteration (measurement), a first filter evaluates the validity of the measurement. This more flexible filter allows to determine the beginning of an aggregate. If the measurement if not valid according to the flexible criterion, it becomes aberrant and the filter iterates to the next measurement. If the measurement is validated according to the flexible criterion, it is transmitted to a filter with a more strict criterion. The objective

of this second filter is to determine if the measures belong to the same aggregate. Therefore, if the strict criterion is validated it is added to the aggregate and the filter iterates to the next measurement. If the strict criterion is not validated, the measurement is considered to belong to another aggregate and is marked as a break point. The first flexible filter thus inputs said measurement and opens a new aggregate if it corresponds to its criterion.

The flexible and strict filters are both identical, with the only difference between them being the tolerance τ which defines its flexibility. The bigger this parameter, the more flexible the filter becomes, and vice-versa. This algorithm was previously presented in the figure 2. One of the main difficulties was settings the different τ values for each filter. In order to define it, we observe the behavior of the filters with a scan of the environment containing a single object. We therefore present the covariance and state evolution of the filters initialized via least-squares on measurements of a single aggregate.

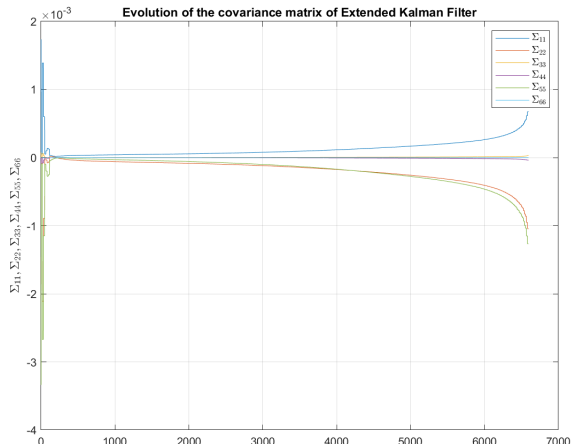
3.3.2 Segmentation results and discussion



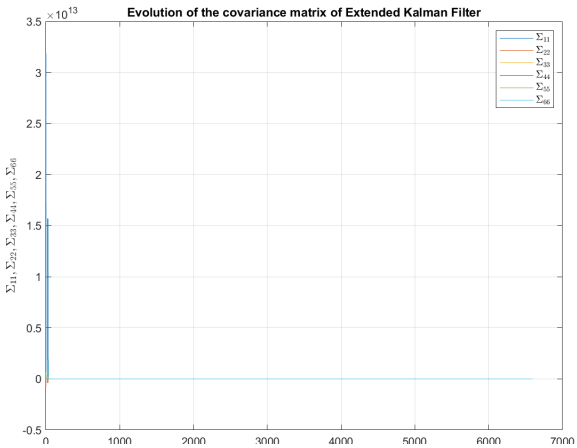
$N = 50$ values $N = 100$ values

$N = 300$ values $N = 1000$ values

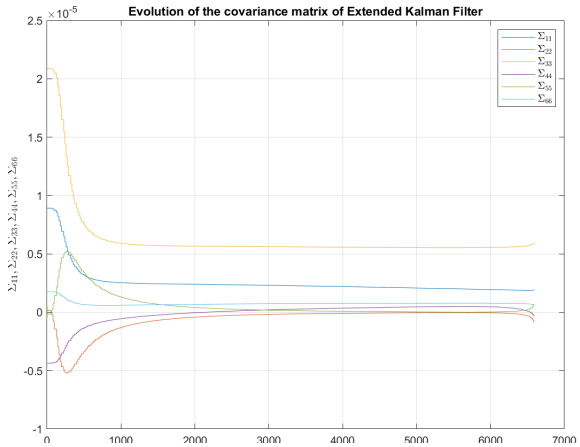
Figure 4: Evolution of the filter state



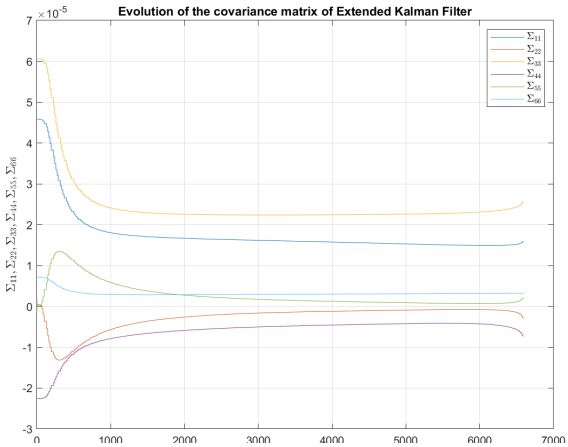
$N = 50$ values



$N = 100$ values

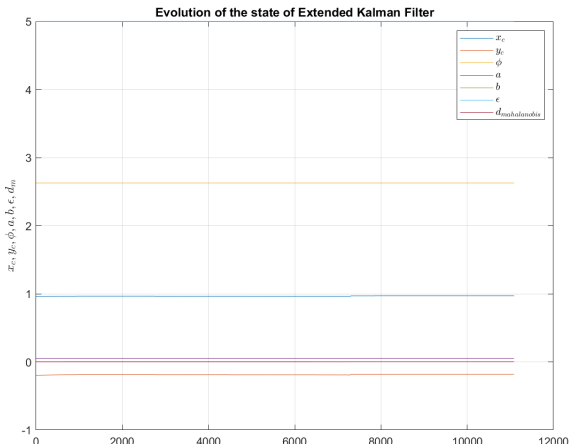


$N = 300$ values

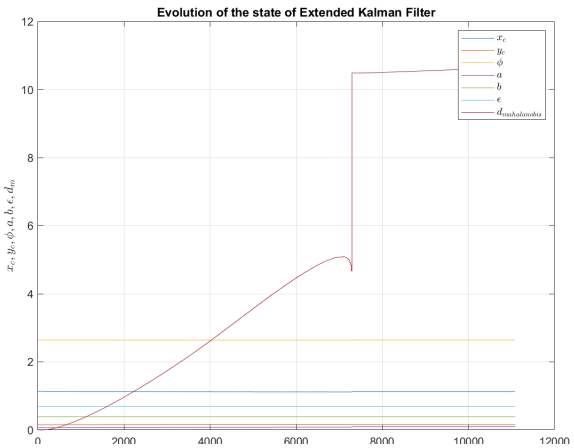


$N = 1000$ values

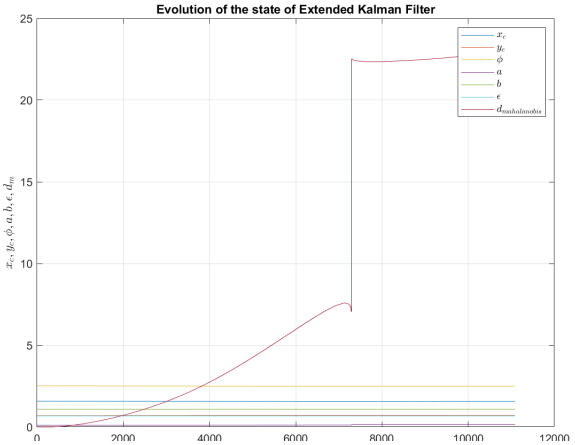
Figure 5: Evolution of the state error covariance



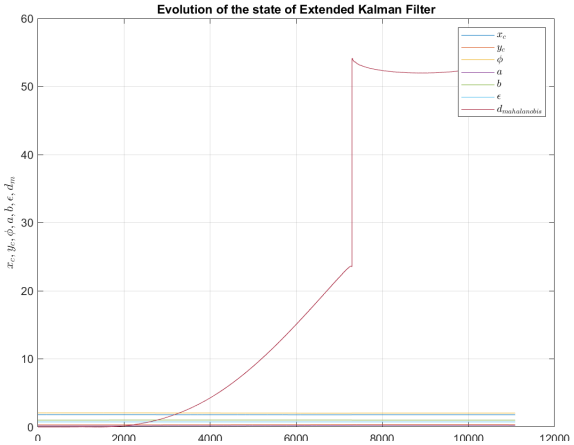
$N = 50$ values



$N = 100$ values

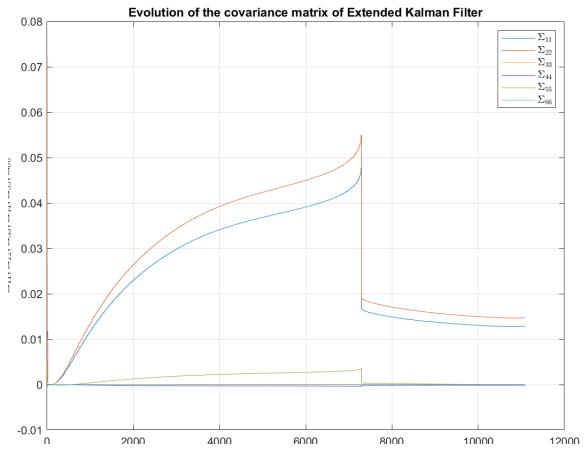


$N = 300$ values

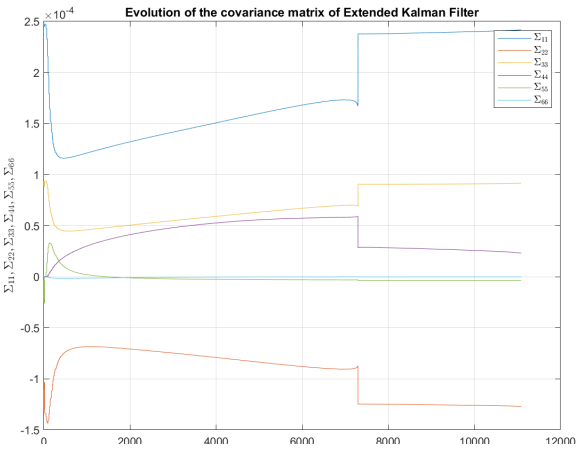


$N = 1000$ values

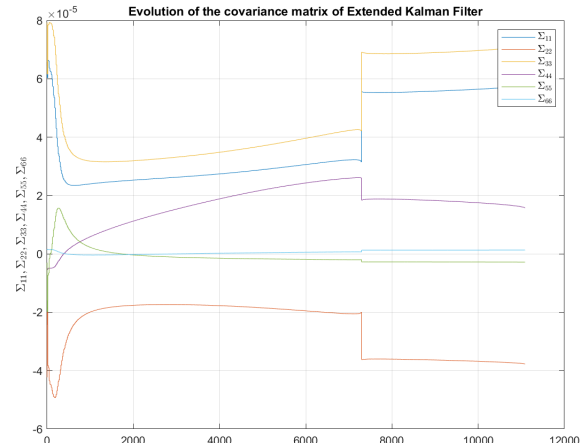
Figure 6: Evolution of the filter state with two obstacles



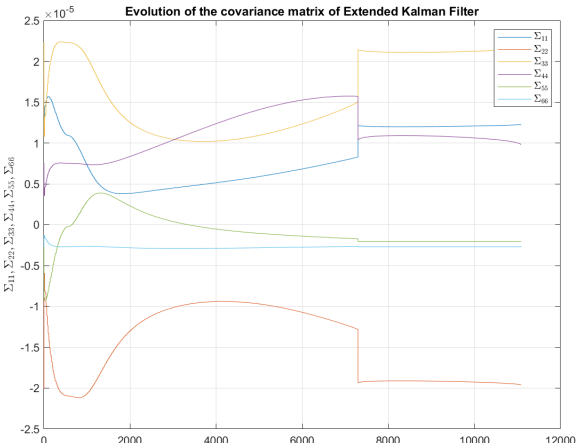
$N = 50$ values



$N = 100$ values



$N = 300$ values



$N = 1000$ values

Figure 7: Evolution of the state error covariance with two objects

We observe on figures 4 and 5 that when initializing the filter with less than 100 measurements, the filter does not allow us to differentiate measurements. From around 300 measurements, the filter starts converging which allows us to properly evaluate and aggregate the measurements of the scan. The extended filter was then used on measurements taken in an environment with two obstacles. The state error covariance and state evolution are presented in 6 and 7. On these figures, it is immediately visible that the *Mahalanobis distance* shoots up as the measurements are evaluated. This change corresponds to $k = 7296$, the measurement corresponding to the second obstacle. With these and similar tests in other environment, we conclude that an optimal *Mahalanobis distance* is 10 for the strict filter and 60 for the flexible filter.

We test the final system on an environment with more objects in figure 8. For closer objects in figure 9 we tune the strict criterion to 3 in order to attain better separation. However since the filter operates on distance and angle measurements, the position of the sensor greatly influences how successful the segmentation is in this case. Thus with closer objects despite the tuned criterion, segmentation is not successful from some positions as seen in figure 10. The real mesh of the objects is attached in annex (figures 11, 12 and 13).

It is clear that the system could also benefit from several improvements. We notably notice an interesting phenomenon with the least squares approximation: when the sensor is almost aligned with a side the far out measurements on that side become more spaced out. This causes the approximation to over estimate the size of the ellipse along that dimension. This could be dealt with using a method similar to the gradient-based cost function least-squares. A simple but potentially effective improvement for segmentation would be the implantation of adaptive criteria for the strict and flexible filters. The system would take into account the angle and position of the robot in respect to the current aggregate and adapt the τ range of the next predicted valid point according to the laser scan resolution.

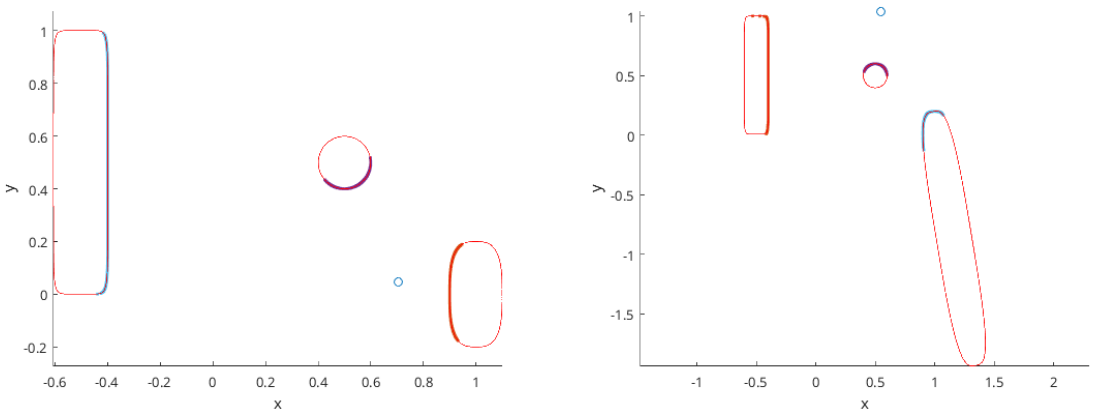


Figure 8: Three object segmentation and least-squares parametrization, far objects

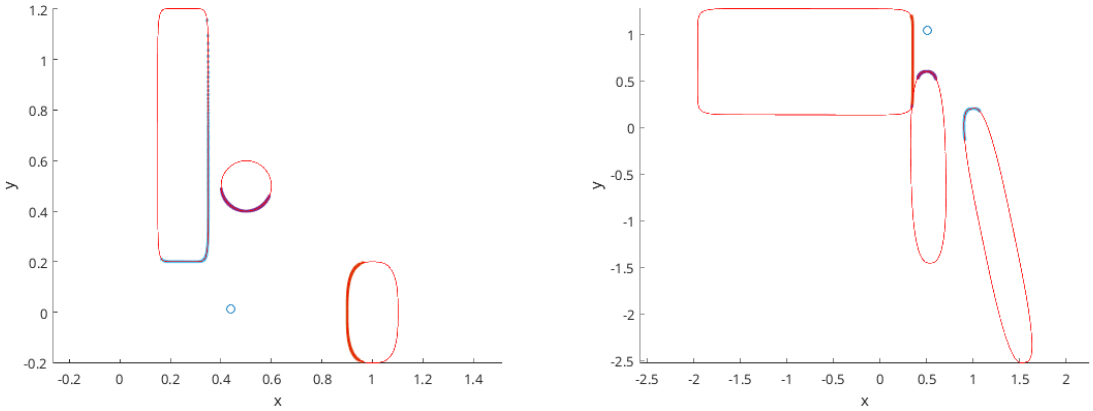


Figure 9: Three object segmentation and least-squares parametrization, close objects

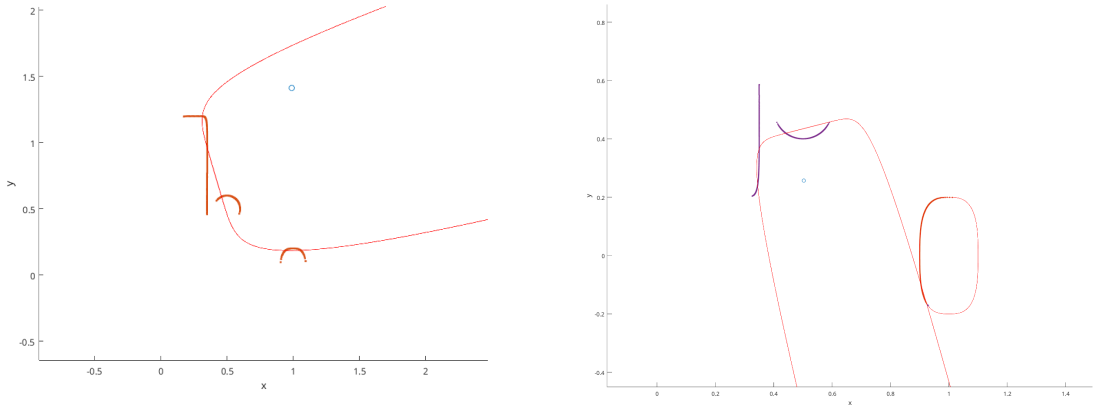


Figure 10: Unsuccessful segmentation

4 Conclusion

In the context of this project we have implemented two main modules of a navigation system. The first is a non-linear least-squares approximation function based around a trust region numerical method in order to fit point aggregates to a super-ellipse function. The second was the implementation of a dual-EKF system for segmentation using the least-squares approximation to initialize the aggregates. In future work, it would be interesting to solve the problems discussed in the previous section. It would also be interesting to implement the system on real-time hardware and test its applications in the real world.

5 Bibliography

1. S. Chaudonneret, "Segmentation, localisation et cartographie avec primitives géométriques 2D," M.S. thesis, DGE , Polytechnique, Montréal, Canada, 2021. [Accessed on: Feb. 14, 2022].

2. T. F. Coleman and Y. Li, "An Interior Trust Region Approach for Nonlinear Minimization Subject to Bounds," SIAM Journal on Optimization, vol. 6, no. 2. Society for Industrial Applied Mathematics (SIAM), pp. 418–445, May 1996. doi: 10.1137/0806023.

3. "Regular API reference," regular API reference. [Online]. Available: https://www.coppeliarobotics.com/helpFiles/en/ap [Accessed on: Mar. 28, 2022].

4. "Nonlinear least-squares fitting," Nonlinear Least-Squares Fitting - GSL 2.7 documentation. [Online]. Available: https://www.gnu.org/software/gsl/doc/html/nls.html. [Accessed on: Mar. 28, 2022].

6 Annex

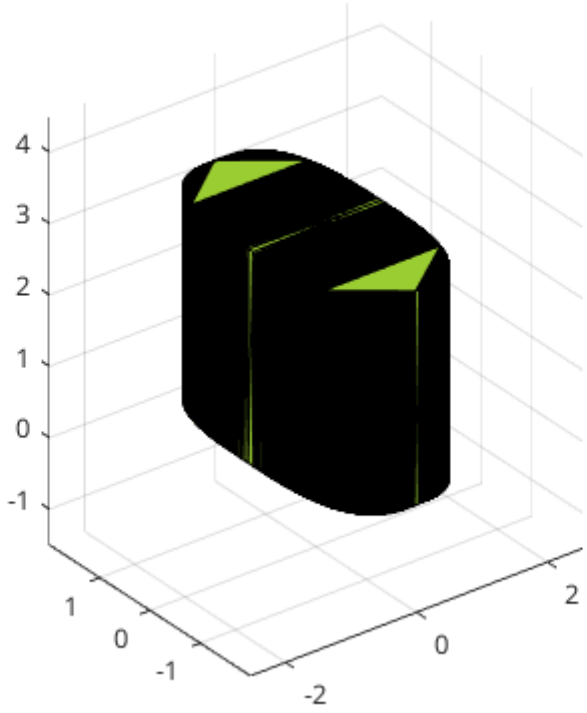


Figure 11: Obstacle 1

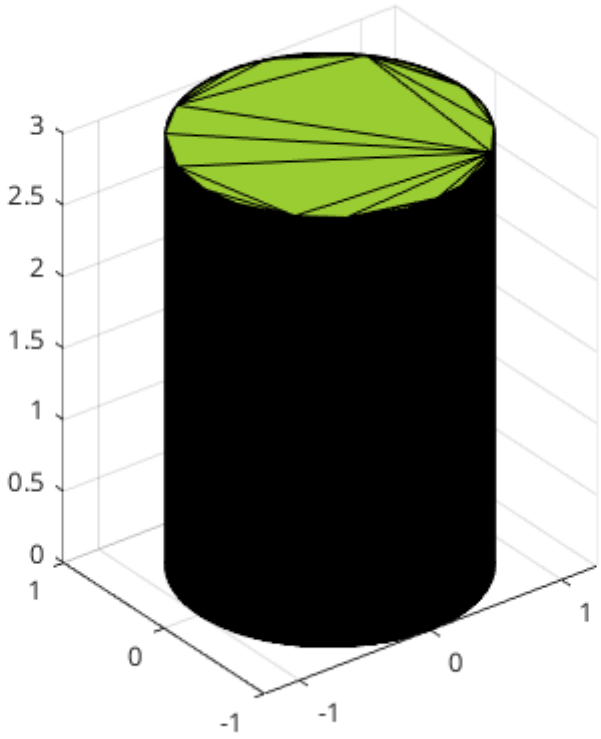


Figure 12: Obstacle 2

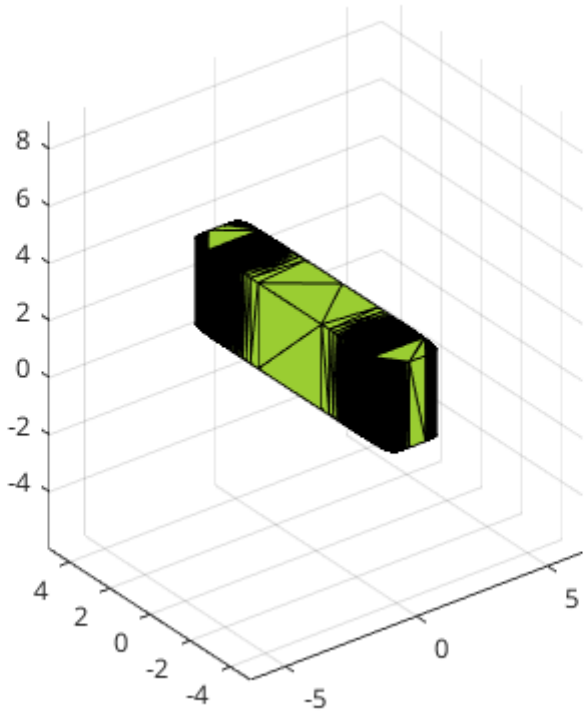


Figure 13: Obstacle 3