Localization using poles & signs detected by a lidar

UTC - ARS4 - Estimation for robotic navigation

Victor Demessance, Galia Fabiola Cornejo Urquieta, Fadi Ben Ali

I. System modeling

A. State vector

In order to represent the pose of the vehicule in the 2D space, we can use the following state vector $\mathbf{x}_t$:

$$X = \begin{bmatrix} x \\ y \\ \theta \\ v \\ \omega \end{bmatrix}$$

with :

- x, y : Coordinates of the vehicule (ENU working frame).
- θ : Heading of the vehicule (ENU working frame).
- v : Longitudinal speed.
- ω : Angular speed.

B. Dynamic space state model

The evolution of the system can be define as follows :

$$X_{t+1} = f(X_t, u_t) + q_t$$

with :

- $\mathbf{u}_t = [v_t, \omega_t]^T$: speed inputs.
- $\mathbf{q}_t \sim \mathcal{N}(0, Q)$: model noise.

Thanks to the Euler method, that uses the derivatives definition to define the value of the next iteration of the discrete space state, we have :

$$X_{k+1} = X(t) + \Delta.\dot{X}(t),$$

Using basic definitions of geometry and automatic control, we can define the nonlinear function f such that (assuming constant speeds between 2 samples) :

$$\begin{aligned} x_{t+1} &= x_t + v_t \Delta t \cos(\theta_t), \\ y_{t+1} &= y_t + v_t \Delta t \sin(\theta_t), \\ \theta_{t+1} &= \theta_t + \omega_t \Delta t, \\ v_{t+1} &= v_t, \\ \omega_{t+1} &= \omega_t. \end{aligned}$$

Consequently, we have

$$f(X_t, u_t) = \begin{bmatrix} x_t + v_t \Delta t \cos(\theta_t), \\ y_t + v_t \Delta t \sin(\theta_t), \\ \theta_t + \omega_t \Delta t, \\ v_t, \\ \omega_t. \end{bmatrix}.$$

II. Observation Model

We can define observation as the reception of GNSS and Lidar information. In the model, we describe it as :

- GNSS (x_{GNSS}, y_{GNSS}, θ_{GNSS})

  $$\begin{aligned} x_{\text{GNSS}} &= x \\ y_{\text{GNSS}} &= y \\ \theta_{\text{GNSS}} &= \theta \end{aligned}$$

  with $\sigma_x^2 = \sigma_y^2 = 0.2$ and $\sigma_\theta^2 = 0.01$

- Lidar (x_{lidar}, y_{lidar}, x_{map}, y_{map})

  To process LiDAR observation, we make a change of variables in polar coordinates to compute the estimated observation:

  $$\hat{\rho}^i = \sqrt{(x^i_{map} - x_t)^2 + (y^i_{map} - y_t)^2}$$

  $$\hat{\lambda}^i = \arctan 2(y^i_{map} - y_t, x^i_{map} - x_t) - \theta_t$$

  And so we can compare it with the real observation in polar coordinates

  $$\rho^i = \sqrt{(x_{lidar})^2 + (y_{lidar})^2}$$

  $$\lambda^i = \arctan 2(y_{lidar}, x_{lidar})$$

The global observation model equation can be finally written as follows (defining M as the map informations):

$$\mathbf{z}_t = g(X_t, M),$$

with

$$\mathbf{z}_t = \begin{bmatrix} x_{\text{GNSS}} \\ y_{\text{GNSS}} \\ \theta_{\text{GNSS}} \\ \rho^i \\ \lambda^i \end{bmatrix}, \quad g(X_t, M) = \begin{bmatrix} g_{\text{GNSS}} \\ g_{\text{LiDAR}} \end{bmatrix} = \begin{bmatrix} x_t \\ y_t \\ \theta_t \\ \hat{\rho}^i \\ \hat{\lambda}^i \end{bmatrix}.$$

As Lidar obervations are received at a higher frequency than GNSS observations (10Hz vs. 1Hz), they will enable us to update our state estimation more frequently.

III. Simulation Implementation

A. Extended Kalman Filter

Filter is implemented following system and observation modelling sections. Once implemented, we seek to tune it so that it has 95% consistency. We also have to check if the estimation error is not biased. We do consistency test using the reference values for the variables x, y and heading (h). We do the test for each of them. The results are shown in the following table:

Variable	Mean Error	Max Error	MSE	Consistency
x	0.10644	1.9729	0.15641	0.99853
y	-0.056811	2.3455	0.12408	0.98387

The initial state for the simulation refered to the first position of the GNSS sensor. We have

$$x_0 = \begin{bmatrix} \text{gnss(1).x} \\ \text{gnss(1).y} \\ \text{gnss(1).heading} \\ \text{v(1)} \\ \text{omega(1)} \end{bmatrix}$$

Initial covariance matrix P is set to

$$P = \begin{bmatrix} 0.1 & 0 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 & 0 \\ 0 & 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0.5 \end{bmatrix}$$

Initial process noise matrix Q is set to

$$Q = \begin{bmatrix} 0.1 & 0 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 & 0 \\ 0 & 0 & 0.7 & 0 & 0 \\ 0 & 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0 & 0.5 \end{bmatrix}$$

$$R_g = \begin{bmatrix} 0.1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.01 \end{bmatrix}$$

$$R_l = \begin{bmatrix} 0.7 & 0 \\ 0 & 0.7 \end{bmatrix}$$

Note that we trust a lot more the GNSS measures than the lidar measures, given the fact that in real data, detection algorithms may fall to detect poles and traffic signs and may also detect some false positives, so we expect to have a really noisy data. For the simulation, we decided to separate the two observation models and implement 2 separate Kalman filters. In this way, we can correct the state with two types of independent observations.

To apply EKF, we need to compute the jacobians of the observation and evolution models. We have :

$$F = \frac{\partial f}{\partial X} = \begin{bmatrix} 1 & 0 & -v\Delta t \sin(\theta) & \Delta t \cos(\theta) & 0 \\ 0 & 1 & v\Delta t \cos(\theta) & \Delta t \sin(\theta) & 0 \\ 0 & 0 & 1 & 0 & \Delta t \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

$$C_{\text{GNSS}} = \frac{\partial g_{\text{GNSS}}}{\partial X} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}.$$

$$C_{\text{LiDAR}} = \begin{bmatrix} -\frac{\Delta X}{r} & -\frac{\Delta Y}{r} & 0 & 0 & 0 \\ \frac{\Delta Y}{r^2} & -\frac{\Delta X}{r^2} & -1 & 0 & 0 \end{bmatrix}$$

With

$$\Delta_X = x_{map} - x_t$$
$$\Delta_Y = y_{map} - y_t$$
$$r = \sqrt{\Delta_X^2 + \Delta_Y^2}$$

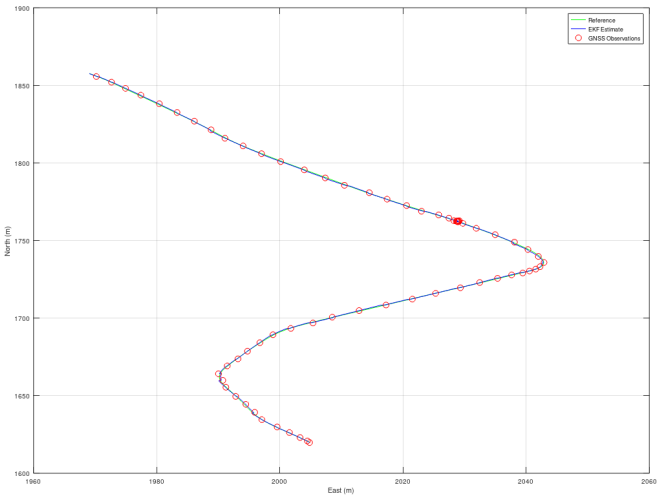

Fig. 1. EKF Localization with GNSS and LiDAR observations

B. Unscented Kalman Filter

Following the same protocole we have the following results for the unscented kalman filter:

Variable	Mean Error	Max Error	MSE	Consistency
x	-0.03998	0.51297	0.028646	1.0000
y	-0.16454	0.61987	0.055269	0.99413
h	-0.0082228	0.042492	0.00010691	1.0000

The initial state for the simulation refered to the first position of the GNSS sensor. We have

$$x_0 = \begin{bmatrix} \text{gnss(1).x} \\ \text{gnss(1).y} \\ \text{gnss(1).heading} \\ \text{v(1)} \\ \text{omega(1)} \end{bmatrix}$$

Initial covariance matrix P is set to

$$P = \begin{bmatrix} 0.5 & 0 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0 & 0 \\ 0 & 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0.5 \end{bmatrix}$$

Initial process noise matrix Q is set to

$$Q = \begin{bmatrix} 0.1 & 0 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 & 0 \\ 0 & 0 & 0.001 & 0 & 0 \\ 0 & 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0 & 0.001 \end{bmatrix}$$

$$R_g = \begin{bmatrix} 0.2 & 0 & 0 \\ 0 & 0.2 & 0 \\ 0 & 0 & 0.01 \end{bmatrix}$$

$$R_l = \begin{bmatrix} 0.3 & 0 \\ 0 & 0.3 \end{bmatrix}$$

Note that we trust more the GNSS measures than the lidar measures, just as the previous case.

In order to set up the UKF filter, we need to define our sigma points. In our case, th such that :

$$\mathbf{X}_i = x \pm \sqrt{(n+\lambda)\mathbf{P}}, \quad i = 1, \ldots, 2n$$

with

- $\lambda = \alpha^2(n+\kappa) - n$
- n the state vector dimension
- α, β, κ parameters defining the spread of sigma points

Then, we propagate sigma points $\mathbf{X}_i$ through the dynamic model:

$$\mathbf{X}_i^{\text{pred}} = f(\mathbf{X}_i, u_t)$$

and we can compute the statistical moments of the model like in the UKF definition. The rest is simply the application of the UKF filter's discounting and prediction formulas. You can see the results on the Figure n°2.

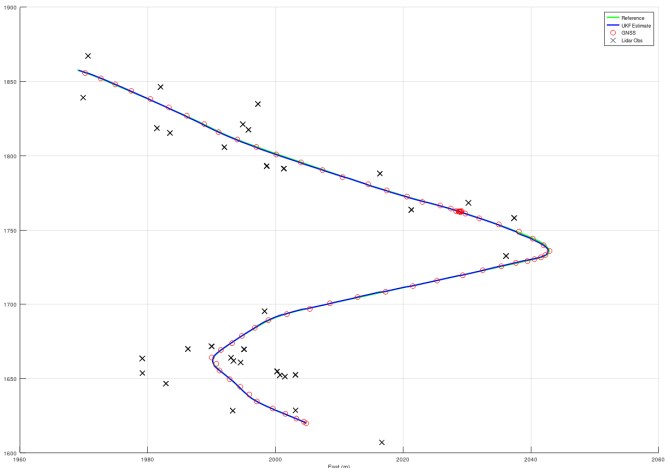

Fig. 2. UKF Localization with GNSS and LiDAR observations

IV. Real Implementation

A. Multi sensor data fusion

To be able to use the LiDAR observations in real time, we need to create a data association algorithm that will link the observation to the global pole position in the ENU frame. We create a NN association based on the global poles map (Figure n°3) and the observed poles converted in the ENU frame.

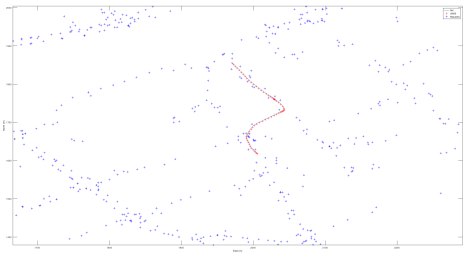

Fig. 3. Global Poles Map superposed on the Reference Trajectory (with GNSS obs)

For each pole detected by the LiDAR, its coordinates in the vehicle frame $(x_{\text{lidar}}, y_{\text{lidar}})$ are transformed into the global frame $(x^M_{\text{lidar}}, y^M_{\text{lidar}})$ using the current estimated robot state (x_t, y_t, θ_t). For this process, we use the **cartesian coordinates** and we compute :

$$\begin{bmatrix} x^M_{\text{lidar}} \\ y^M_{\text{lidar}} \end{bmatrix} = \begin{bmatrix} x_t \\ y_t \end{bmatrix} + R(\theta_t) \begin{bmatrix} x_{\text{lidar}} \\ y_{\text{lidar}} \end{bmatrix}$$

That can be reformulate as :

$$x^M_{\text{lidar}} = x_t + \cos(\theta_t) \cdot x_{\text{lidar}} - \sin(\theta_t) \cdot y_{\text{lidar}}$$
$$y^M_{\text{lidar}} = y_t + \sin(\theta_t) \cdot x_{\text{lidar}} + \cos(\theta_t) \cdot y_{\text{lidar}}$$

After that, we compute the Euclidean distance between its position $(x^M_{\text{lidar}}, y^M_{\text{lidar}})$ and all the poles in the map $(x_{\text{map}}, y_{\text{map}})$

$$\mathrm{d}_j = \sqrt{(x^j_{\text{map}} - x^M_{\text{lidar}})^2 + (y^j_{\text{map}} - y^M_{\text{lidar}})^2}$$

where j iterates over all the poles in the map.

In the case of a NN association, we basically link the map pole with the smallest distance to the detected pole as the corresponding association (Figure n°4):

$$j^* = \arg\min_j \mathrm{d}_j$$

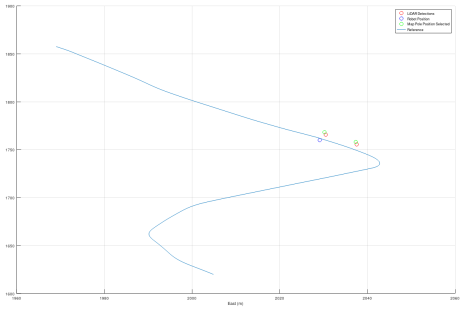

Fig. 4. NN process on LiDAR observation with selected linked map poles

B. Extended Kalman Filter

The implementation of the EKF is the same than in the simulation part, we just need to apply the NN algorithm before the process of LiDAR observation.

C. Unscented Kalman Filter

Development of the UKF filter for real data is carried out in the same way as for simulated data, using the nearest neighbor data association process accordingly.

V. Filter comparison

To evaluate the two filters we use reference values and compute for variables x, y and h.
Unfortunately, we didn't have time to finalize the development of the UKF filter for real data. As a result, the comparison table is only available for simulated data.

TABLE I
FILTER COMPARISON

SIMULATION						
	Mean Error		**Max Error**		**MSE**	
var	EKF	UKF	EKF	UKF	EKF	UKF
x	0.10644	-0.03998	1.9729	0.51297	0.15641	0.028646
y	-0.056811	-0.16454	2.3455	0.61987	0.12408	0.055269
h	-1.2708	-0.0082228	6.3689	0.042492	2.567	0.00010691

VI. Conclusions

According to the results in the previous table, we can see that the UKF filter works better than the EKF. Particularly, the estimate of the variable h made by the unscented Kalman filter is impressive compared to the estimate made by the filter. This is something we expected, since the filter propagates the sigma points through the non-linear functions and therefore does not make linear approximations.

We conclude also that, in general, for the initial values, we assume that the lidar data is less reliable than the gnss and therefore it has more noise (in fact, for the real data, the lidar may miss some poles or detect some false positives).