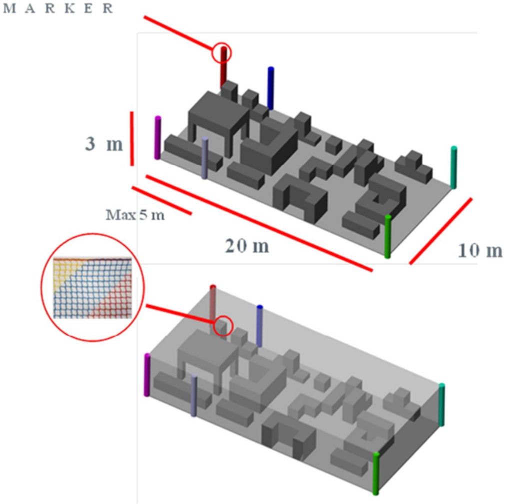
A ROS implementation of a 6-DoF EKF for indoor drone Visual SLAM

Diego Emanuel Avila

15-12-2020

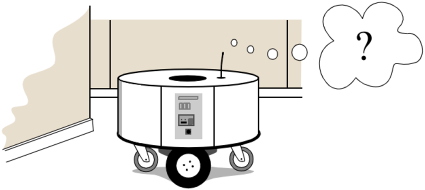
Table of contents

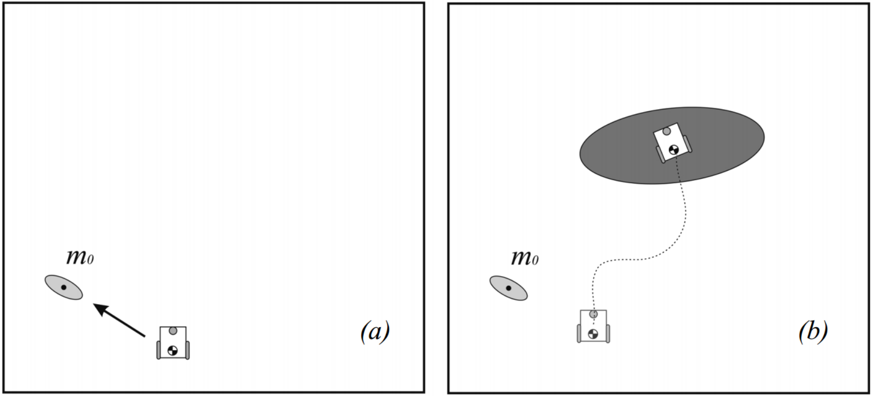
- Indoor environment
- Fully autonomous drone
- No GNSS nor Laser devices
- Inertial devices, range sensors, cameras and speed sensors are allowed
- Obstacles have at most 3mts height with passages of at least 1mt
- Landmarks: colored poles and QR markers
- 2-phase competition: inspection and path-follow

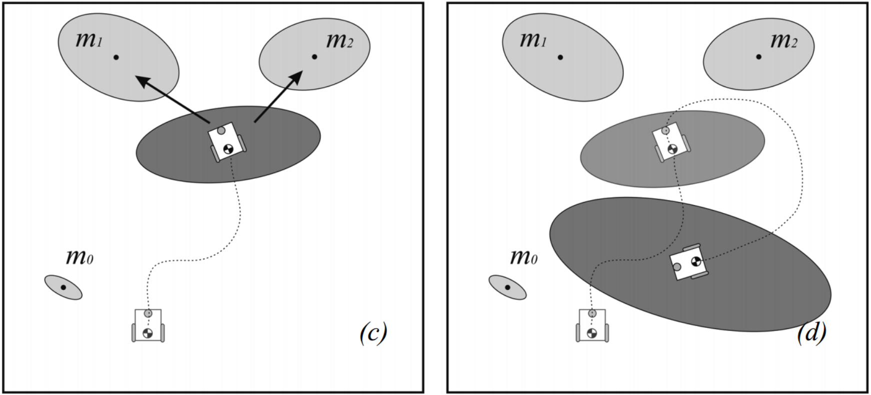

MARKER
3 m
Max 5 m
20 m
10 m

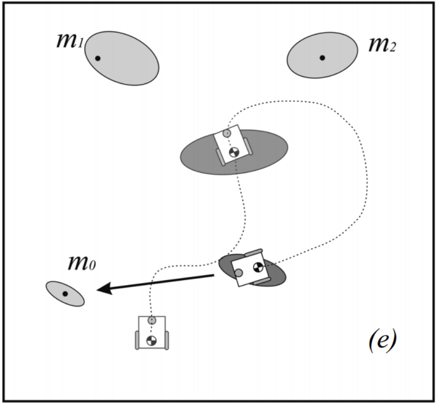
Simultaneous Localization & Mapping

- Localization: where the robot is
- Mapping: build a map of the environment
- SLAM: localize itself while building a map




m_0
(a)
m_0
(b)


m_1
m_2
m_0
(c)
m_1
m_2
m_0
(d)


m_1
m_2
m_0
(e)

EKF-SLAM

- Motion and observation processes are usually not linear
- Motion model: $g\left(u_t, \mu_{t-1}\right)$
- Observation model: $h\left(\hat{\mu}_t\right)$
- EKF uses Taylor expansion to linearize the processes
- Builds feature-based maps
- The full-state (map) is
  $\mu = \begin{bmatrix} q_r & m_0 & \dots & m_{n-1} \end{bmatrix}^T$
- As in Kalman Filter, the process comprises 2 steps: prediction and correction

Input: μ_{t-1}, Σ_{t-1}, u_t, z_t

```
1  μ̂_t = g(μ_{t−1}, u_t)
2  G_t = computeJacobian(g)
3  Σ̂_t = G_t Σ_{t−1} G_t^T + R_t
4  foreach landmark observation z_t^i do
5      if landmark i has not being seen before then
6          addLandmarkToStateVector(z_t^i)
7      end
8      H_t^i = computeJacobian(h^i)
9      v^i = z_t^i − h^i(μ̂_t)
10     S = H_t^i Σ̂_t H_t^{iT} + Q_t
11     K_t^i = Σ̂_t H_t^{iT} S^{−1}
12     μ_t = μ̂_t + K_t^i (v^i)
13     Σ_t = (I − K_t^i H_t^i) Σ̂_t
14 end
15 return μ_t, Σ_t
```

- Lines 1-4 are the prediction step
- Lines 4-14 are the correction step
- If the landmark has not being seen before, the algorithm adds it in lines 5-7

New landmarks

- Map is not always available
- Inverse sensor model: $h^{-1}(q_r, z)$
- The state vector is extended with h^{-1}
- The covariance matrix is extended as well:

$$Y_x = \frac{\partial y}{\partial x} = \begin{bmatrix} \frac{\partial x}{\partial x} \\ \frac{\partial h^{-1}}{\partial x} \end{bmatrix} = \begin{bmatrix} I \\ G_x \end{bmatrix}$$

$$Y_z = \frac{\partial y}{\partial z} = \begin{bmatrix} \frac{\partial x}{\partial z} \\ \frac{\partial h^{-1}}{\partial z} \end{bmatrix} = \begin{bmatrix} 0 \\ G_z \end{bmatrix}$$

$$\hat{\Sigma} = \begin{bmatrix} \Sigma & \Sigma G_x^T \\ G_x \Sigma & G_x \Sigma G_x^T + G_z Q_t G_z^T \end{bmatrix}$$

Normalized Estimation Error Squared

- Defined as:

$$NEES = (\boldsymbol{x} - \hat{\boldsymbol{x}})\, \Sigma^{-1}\, (\boldsymbol{x} - \hat{\boldsymbol{x}}) \leqslant \chi^2_{d,1-\alpha}$$

- Can be performed to check the consistency of the filter
- Consistency of the filter is maintained by using observations that satisfy the test

$$\begin{aligned} v^i &= z^i_t - h^i\left(\hat{\mu}_t\right) \\ S &= H^i_t \hat{\Sigma}_t H^{iT}_t + Q_t \\ D^2_i &= v^i S^{-1} v^i \leqslant \chi^2_{d,1-\alpha} \end{aligned}$$

- MAVROS node interfaces with the flight controller, and provides the control signal
- Control signal is composed by linear and angular velocities

$$u = \begin{bmatrix} v_x^b & v_y^b & v_z^b & \omega_x^b & \omega_y^b & \omega_z^b & \phi^b & \theta^b & \psi^b \end{bmatrix}^T$$

$$v^w = \mathbf{T} * \begin{bmatrix} v_x^b \\ v_y^b \\ v_z^b \end{bmatrix}$$

Characteristics

- PixHawk 4 Flight Controller
- 8 range finder
- 4 monocular cameras
- 2 stereo cameras: 1 pointing down, 1 pointing forward
- 1 optical-flow device
- Accelerometer + gyroscope
- Magnetometer
- Barometer

- Given the linear velocities transformation v^w, the motion model can be summarized as follow

$$g\left(\mu_{t-1}, u_t\right) = \begin{cases} \begin{bmatrix} \mu_{t-1,x^w} \\ \mu_{t-1,y^w} \\ \mu_{t-1,z^w} \end{bmatrix} + \Delta t * v^w \\ \mu_{t-1,\psi} + \Delta t * \omega_z^b \end{cases}$$

- However, this does not consider the noise

- Noise in the motion process is assumed to be $\mathcal{N}(0, R_t)$
- The noise covariance can be decomposed as $R_t = N * U * N^T$
  - N is the Jacobian of the acceleration with respect to the state vector
  - U is the average acceleration of the drone

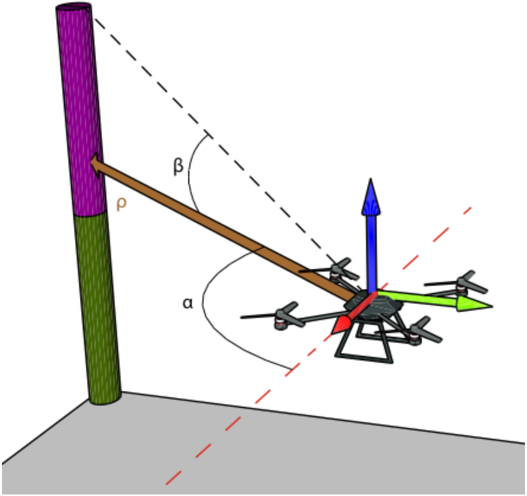
- 2 types of landmarks: Poles & Markers
  - Poles' observations is Range and Bearing
  - Markers' observations is the pose with respect to the camera reference frame
- Additionally, height correction is done using range sensor and 3D obstacle map
- Observation for landmark i is $z_i = h_i\left(x_t\right) + \mathcal{N}\left(0, Q_t\right)$

Poles

- Range and bearing information
  - ρ: distance
  - β: altitude
  - α: azimuth
- Observation model:

$$\begin{bmatrix} p^b_{i,x} \\ p^b_{i,y} \\ p^b_{i,z} \end{bmatrix} = \boldsymbol{T}_{\boldsymbol{r}}^{-1} \begin{bmatrix} p^w_{i,x} \\ p^w_{i,y} \\ p^w_{i,z} \end{bmatrix}$$

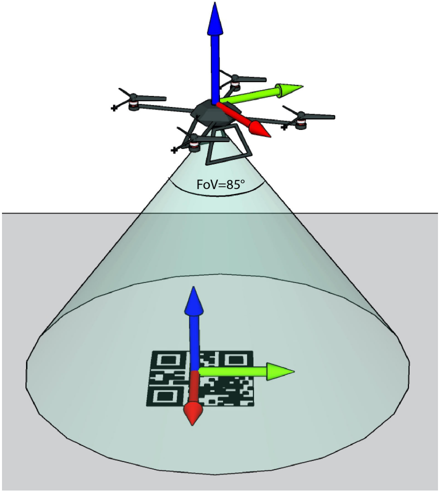
$$h_i(\hat{\mu}_t) = \begin{bmatrix} p_{i,\rho} \\ p_{i,\alpha} \\ p_{i,\beta} \end{bmatrix} = \begin{bmatrix} \sqrt{p^2_{i,x^b} + p^2_{i,y^b}} \\ \mathrm{atan2}\left(p^b_{i,y}, p^b_{i,x}\right) \\ \mathrm{atan2}\left(p^b_{i,z}, p^b_{i,\rho}\right) \end{bmatrix}$$


β
ρ
α

Markers

- Markers are unknown first, and added to the map once the drone sees them
- Observations is composed by its position with respect to the camera reference frame
- Observation model is then:

$$h_i(\hat{\mu}_t) = \begin{bmatrix} m^c_{i,x} \\ m^c_{i,y} \\ m^c_{i,z} \\ m^c_{i,\phi} \\ m^c_{i,\theta} \\ m^c_{i,\psi} \end{bmatrix} = (\boldsymbol{T_r} * \boldsymbol{T_c})^{-1} * \boldsymbol{T_m}$$


FoV=85°

What about adding new markers?

- Inverse observation model should project the observation from the camera reference frame to the world reference frame
- Inverse observation model is then:

$$h_i^{-1}(\hat{\mu}_t) = \begin{bmatrix} m^w_{i,x} \\ m^w_{i,y} \\ m^w_{i,z} \\ m^w_{i,\phi} \\ m^w_{i,\theta} \\ m^w_{i,\psi} \end{bmatrix} = \boldsymbol{T_r} * \boldsymbol{T_c} * \boldsymbol{T^c_m}$$

- And extend the state vector:

$$\mu_t = \left[\begin{array}{c|cccccc} \mu_t & m^w_{i,x} & m^w_{i,y} & m^w_{i,z} & m^w_{i,\phi} & m^w_{i,\theta} & m^w_{i,\psi} \end{array}\right]^T$$

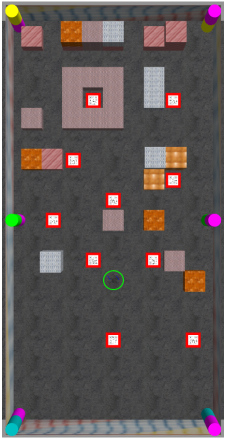
and state covariance matrix

Height

- The observations to correct the height are composed by the range sensor + Octomap measurement
- The height observation is $z = voxel_z + range_{distance}$
- Hence, the observation model is simply:

$$h\left(\hat{\mu}_t\right) = \hat{\mu}_z + bias$$

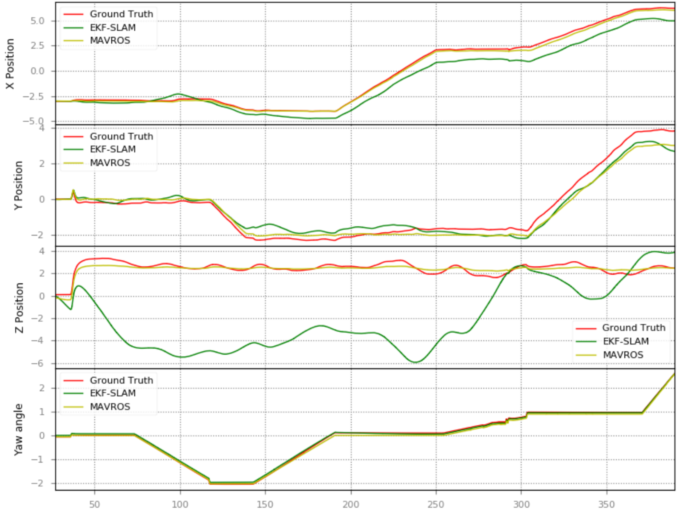
- Obstacles of at most 2mts height
- 6 known colored poles
- 10 unknown markers
- 10mts $\times$ 20mts
- Delimited by net-like walls



All the experiments were done in a simulated environment.
The aim of the experiments were to evaluate:

- the importance of known poles and known and unknown markers in the localization and mapping process
- the accuracy of markers' pose estimation
- the importance of the range and Octomap measurements in the height correction
- the importance of the NEES test to evaluate measurements and the filter's consistency

Odometry only

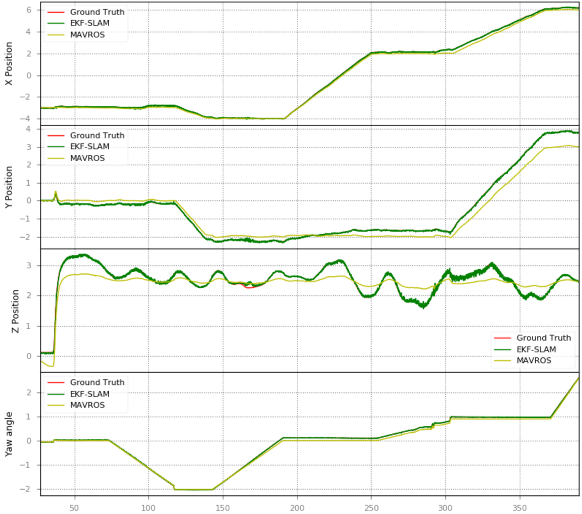


The importance of poles

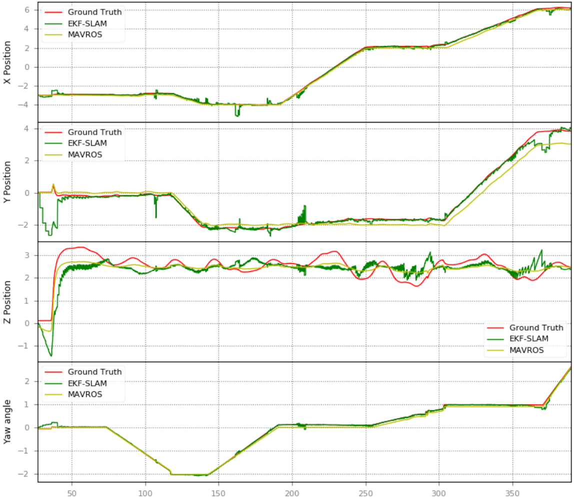
2 experiments were carried out:

1. using *perfect observations* of poles, had the objective of showing the perfect localization compared with the MAVROS localization and the ground truth.
2. using *real observations* taken from the ROS bag, had the objective of understanding if the proposed implementation works as well as with the perfect observations of the previous experiment

Perfect observation of poles



Real observation of poles

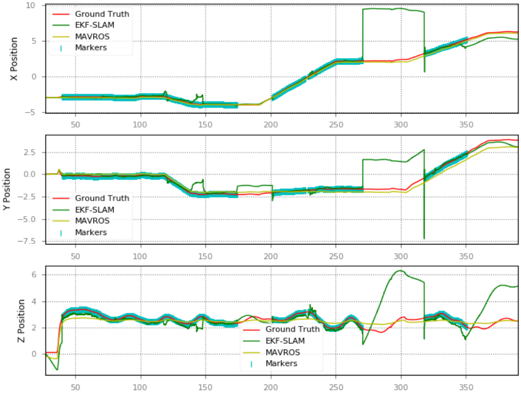


The importance of markers

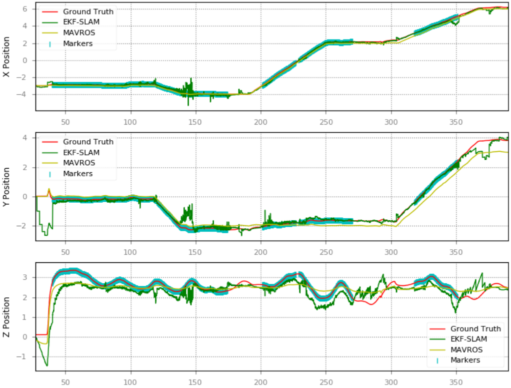
3 experiments were carried out:

1. The drone follows the path and estimates its pose based on known markers only. It had the objective of understanding the importance of markers in the localization process
2. Similar to the previous one, but it uses markers and poles to localize. However, the main difference with the previous experiments lies on the absence or not of a perfect map. It had the objective of understanding the importance of both poles and markers in the SLAM situation.
3. Measures the distance between the true position of markers and the one estimated by the algorithm in a context of *real observations*, both for markers and poles.

Localization using only markers



SLAM with poles and markers



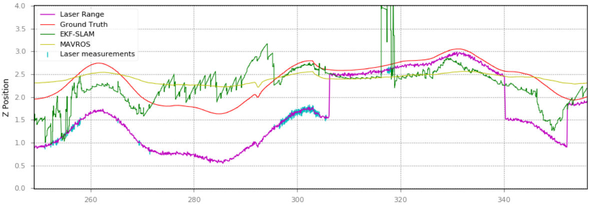
Distance between real and estimated poses.

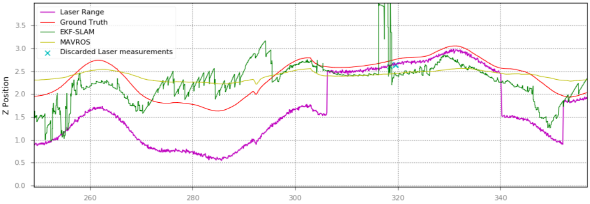
Marker ID	Euclidean distance	ϕ	θ	ψ
0	0.076	0.089	0.142	0.034
1	0.186	0.407	0.145	0.052
4	0.119	0.153	0.132	0.055
5	0.155	0.024	0.057	0.013
Max.	0.186	0.407	0.145	0.055
Min.	0.076	0.024	0.057	0.013
Avg.	0.134	0.168	0.119	0.039

Height estimation

- 2 types of observations: Octomap + range sensor
- Implementation only updates height iff Octomap information is available
- 1 experiment was carried out with the objective of understanding the influence of both observations in the height update.
- Real landmarks observations were used.

Experiments & Results - Experiments C



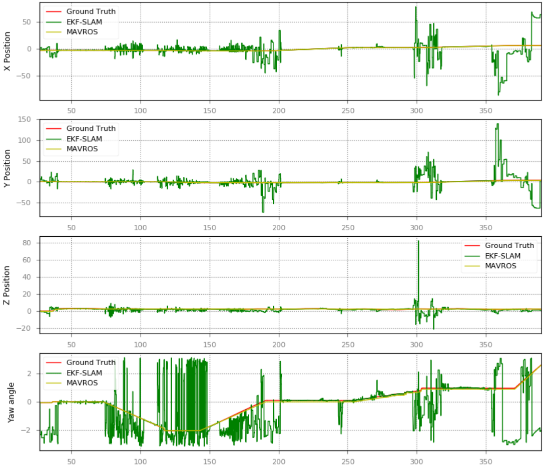


NEES test influence

The experiments were carried out with the objective of understanding

1. the importance of using (or not) the NEES test
2. the importance of the χ^2 value

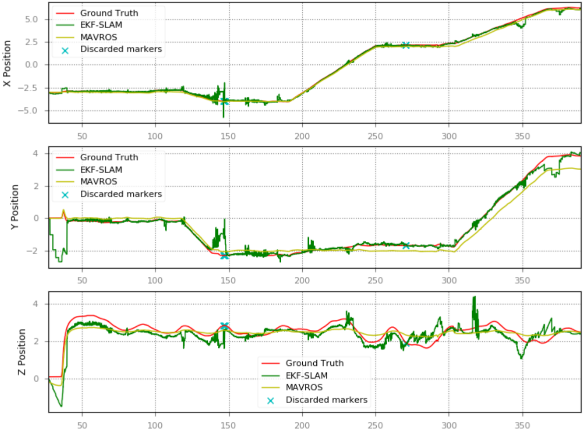
What happen when no NEES test is used?



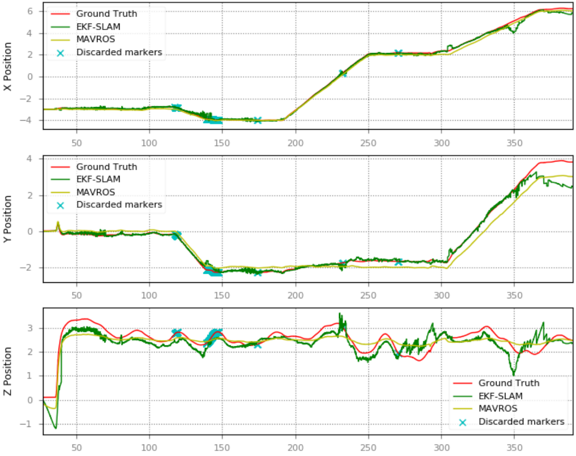
What about the χ^2_α threshold?

- So far, was used a threshold that corresponds to 95% of valid measurements. This means that we assume that the 95% of the observations are valid.
- This corresponds to $\alpha = 0.05$, but this value is the standard one.
- What happen with other confidence levels?
- Moreover, different thresholds can be set for different landmarks

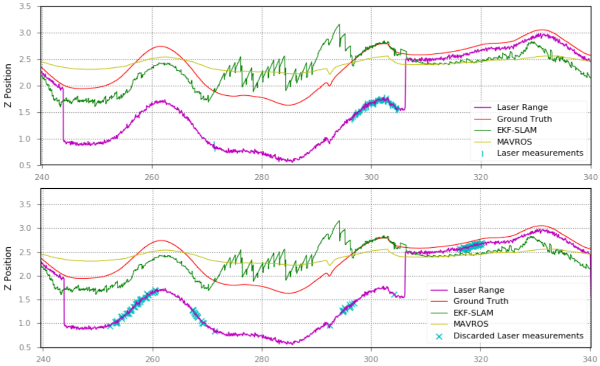
Discarded markers observations when $\alpha = 0.05$



Discarded markers observations when $\alpha = 0.9$



Height correction when $\alpha = 0.9$



Conclusions

- The first 2 experiments showed the importance of poles and markers in the localization process.
- The markers' pose estimation in an SLAM situation is good enough for the competition context.
- The height estimation can be corrected using a combination of Octomap and range sensor.
- The completeness and correctness of the Octomap is fundamental for good results.
- NEES is fundamental for the consistency of the filter.
- Lower confidence of valid measurements implies discarding truly bad measurements, but on the other hand it can discard measurements that can be useful to correct the drone's pose.
- The value of α becomes a parameter of the system.

Future work

- Deploy and test the proposed implementation in the real drone.
- Fine-tune the noise covariance matrices.
- Extensive experimentation with Octomap and range sensor for height update.
- Camera self calibration procedure to improve the Z position estimation using poles and markers
- Compare the current algorithm with other algorithms like Error-State EKF-SLAM, or UKF-SLAM, and evaluate their performance with the current implementation as baseline

Thank you!

Questions?