**MTRN4010 / ADVANCED AUTONOMOUS SYSTEMS T1.2022**

**Week 9 tutorial problems. Estimating unknown variables, via Optimization**

**Aim of this tutorial / Hands on tutorial:** This set of problems aims to help in getting experience in exploiting optimization for estimating unknown variables.

……

**Question 1.**

Consider the problem described in Project 1.Part D. Propose a cost function for being minimized using an optimizer such as Matlab fminsearch.

a) Assume the hypothetical case in which you want to exploit the measurements of ranges to four OOIs whose associated landmarks are known.
Consider the following data:

| | OOIs | | Landmarks | |
|---|---|---|---|---|
| | x | y | x | y |
| 1 | 5.02 | -2.44 | 8.96 | 5.18 |
| 2 | 8.00 | -5.22 | 12.88 | 3.97 |
| 3 | 8.99 | -14.72 | 17.91 | -4.14 |
| 4 | 12.00 | -12.50 | 19.65 | -0.84 |

(this data is defined and used in the example code, so that you do not need to copy the values from here)

In which the column named OOI specifies the positions, in the LiDAR's CF, of certain detected OOIs, and the column named AL contains the positions of their associated landmarks (expressed in GCF).

Define a cost function based on the items specified in the table. Base the cost function on measurements of range and bearing, inferred from the positions of the OOIs specified in the column OOI. The cost function is intended the be used to estimate the vehicle's pose from which the OOIs were detected.

Additional necessary data: the LiDAR sensors is installed at pose (Lx;0;0) in the vehicle's coordinate frame, being Lx=0.45m.

**Question 2.**

Implement the cost function proposed in question1. Minimize it by using an optimizer, such as **fminsearch**.

Solution: It will converge to the actual vehicle's pose `[3;5;0.45];( in (m,m,radian))`

However, it will not match it perfectly as the OOIs positions are affected by noise.

**Question 3.**

Implement, as a Matlab function, a cost function appropriate for the following specifications:

In place of considering ranges and bearing, propose a cost function which exploits discrepancy between the positions of the associated landmarks and the positions of the OOIs in the GCF, to be used to infer the vehicle's pose. The positions are expressed in Cartesian representation (not in terms of ranges and bearings)

Exploit the fact that the detected OOI's positions, if expressed in the GCF, do depend on the vehicle's pose.

```
% pseudo code of required cost function
function Cost= MyCostFunction(  Local_OOIs_positions, Landmarks_positions_GCF, LiDAR_pose)
OOIs_positions_in_GCF= Local2Global ( Local_OOIs_positions , LiDAR_pose);
% expresss, in GCF,  OOIs' positions

Distances = SomeProperNorm(OOIs_positions - Landmarks_positions_GCF );
% evaluate distance of each OOI and its associated landmark
Cost= CombineDistances(Distances);
%combine the list of distances, in a proper way, to define a valid cost function
% A possible implementation can be found in file "Q3Tu9().m"

end;
```

Note that in this case, all the equations which we use in defining the cost function, correspond to the same type of physical variable (position), which means no scaling factors are needed to compensate for the engineering units being used. That matter may have been an issue Part D

End to tutorial.