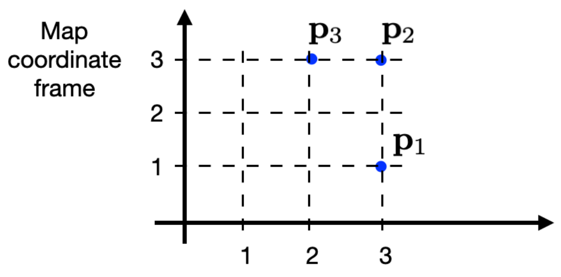
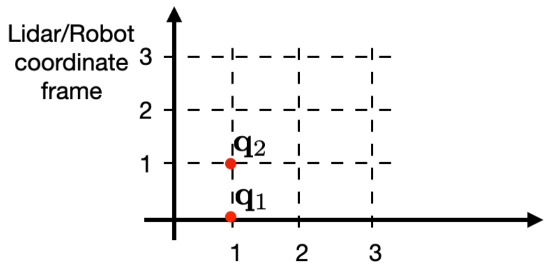
Exam from ARO (resit) **Name:** ____________________
Number of submitted sheets: ____________________

June 27, 2022

1. **Pose from known correspondences:** Consider the problem of robot localization in 2D world from known marker positions. Three markers $i = 1, \ldots, 3$ are known to be located at positions $\mathbf{p}_1 = [3,1]^\top, \mathbf{p}_2 = [3,3]^\top, \mathbf{p}_3 = [2,3]^\top$ in the "Map coordinate frame". The robot (which is at unknown position $\mathbf{t}^*$ in map c.f.) detects markers and measures its positions $\mathbf{q}_i$ in its own "Lidar/Robot coordinate frame". Only two markers are detected: marker 1 at position $\mathbf{q}_1 = [1,0]^\top$ in the Lidar/Robot c.f. and marker 2 at position $\mathbf{q}_2 = [1,1]^\top$ (marker 3 is not detected). Each measurement is assumed to have zero-mean Gaussian noise with the same diagonal covariance Σ and measurements are assumed to be independent on each other. Only *2D translation* of the robot is considered (any rotation is technically impossible).





a) Suggest a suitable optimization problem that estimates the robot pose $\mathbf{t}^*$?

b) What is the globally optimal solution of the above suggested problem?

$\mathbf{t}^* =$

2. **2D localization from unknown correspondences:** Let us assume that everything is the same as in the previous example: sensor has Gaussian noise, same marker positions and measurements, and only 2D translation is considered. The only difference is that correspondences are unknown, i.e. you measured the two markers $\mathbf{q}_1, \mathbf{q}_2$ but the markers are indistinguishable from each other, therefore you need to figure out also the correspondences between markers and measurements.

   a) Use ICP algorithm without outlier rejection step, that is initialized at position $\mathbf{t} = [1, 1]^\top$, to estimate its translation $\mathbf{t}^*$
   Hint:

   $\mathbf{t}^* =$

3. **Extended Kalman filter:** Let us assume a simplified Turtlebot robot operating on 2D world. Its state $\mathbf{x}_t = [x_t, y_t, \theta_t]^\top$ in time t is described by its x, y-coordinates in the plane and its yaw angle θ_t (i.e. heading of the robot). In each time instance, the robot is controlled in two steps: (i) the robot is turned by amount $\mathbf{u}_t$ (i.e. its yaw angle is updated to $\theta_t + \mathbf{u}_t$), (ii) once turning is finished it moves forward by constant distance d. Its state-transition probability is non-linear system with Gaussian noise:

$$p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t) = \mathcal{N}_{\mathbf{x}_t}(g(\mathbf{x}_{t-1}, \mathbf{u}_t),\ \mathbf{R}_t),$$

where

$$g(\mathbf{x}_{t-1}, \mathbf{u}_t) = \begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix} = \begin{bmatrix} x_{t-1} + d\cos(\theta_t + \mathbf{u}_t) \\ y_{t-1} + d\sin(\theta_t + \mathbf{u}_t) \\ \theta_t + \mathbf{u}_t \end{bmatrix}$$

and $\mathbf{R}_t$ is identity matrix.

a) Linearize the system around the mean $\mu_{t-1} = \begin{bmatrix} 1 \\ 0 \\ \pi/2 \end{bmatrix}$ of its belief from time t-1.

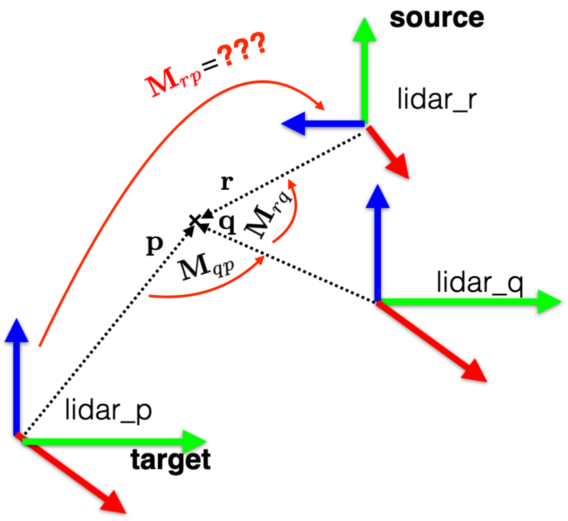
Hint: The linearization is $g(\mathbf{x}_{t-1}, \mathbf{u}_t) \approx g(\boldsymbol{\mu}_{t-1}, \mathbf{u}_t) + \mathbf{G}_t(\mathbf{x}_{t-1} - \boldsymbol{\mu}_{t-1})$, where $\mathbf{G}_t = \frac{\partial g(\mathbf{x}=\boldsymbol{\mu}_{t-1}, \mathbf{u}=\mathbf{u}_t)}{\partial \mathbf{x}}$ is Jacobian of g in point μ_{t-1}.

$\mathbf{G}_t =$

$g(\mathbf{x}_{t-1}, \mathbf{u}_t) \approx$

b) What kind of probability distribution is the belief $\overline{\mathrm{bel}}(\mathbf{x}_t)$ after the prediction step of the EKF (i.e. with the linearized model)? Justify your answer by 1 sentence.

4. **Transformations in homogeneous coordinates:** Lidar_r measures point **r** in its own coordinate frame. The color encodes axes as follows: x-axis = red, y-axis=green, z-axis=blue. The translation of the lidar_r in the lidar_q's coordinate frame is $\mathbf{t} = [0, 5, 5]$, rotation is $+90°$ around x-axis. The translation of the lidar_q in the lidar_p's coordinate frame is $\mathbf{t} = [0, 5, 5]$, and lidar_q and lidar_p have the same orientation of axis (their mutual rotation is roll=pitch=yaw=0°).



a) What is the homogeneous transformation between the lidar_p and lidar_r?

$\mathbf{M}_{rp} =$

b) Given the point $\mathbf{r} = [1, 2, 3]^\top$ in the lidar_r's coordinate frame, what are its coordinates **p** in the lidar_p's coordinate frame ?

$\mathbf{p} =$

5. **Project 3D point to camera:** Consider a perspective camera with the following intrinsic camera parameters K, camera rotation matrix R that corresponds to the rotation of 90° around x-axis and 0° around y-axis and z-axis, and translation vector t:

$$\mathbf{K} = \begin{bmatrix} 300 & 0 & 300 \\ 0 & 300 & 150 \\ 0 & 0 & 1 \end{bmatrix}, \mathbf{t} = \begin{bmatrix} -2 \\ 0 \\ -1 \end{bmatrix},$$

a) Project point

$$\mathbf{q} = \begin{bmatrix} 2 \\ 2 \\ 0 \end{bmatrix}$$

into the camera. What are pixel coordinates $\mathbf{u} \in \mathcal{R}^2$ of the projection?

$u_1 =$

$u_2 =$

6. Describe what is completeness of path planning algorithms.

7. Describe what is probabilistic completeness of path planning algorithms.

8. Describe (using text/sketch/pseudocode .. what you prefer) how sPRM method works. What is input and output of the method? How do you use it to find path between two given configurations? What you can say about optimality of the solution? Is the planner complete/probabilistic complete?

9. Describe (using text/sketch/pseudocode . . . what you prefer) how planning using Vertical cell decomposition works. If the method consists of multiple steps, make clear what is their order. What are pros/cons of the method? For what systems (robots) is the method suitable? What kind of map/obstacle/robot geometry representation is required for it? What is the time complexity of the method?

10. Describe (using text/sketch/pseudocode . . . what you prefer) how planning via Rapidly-exploring Random Tree works. If the method consists of multiple steps, make clear what is their order. What are pros/cons of the method?

11. What distribution of random samples is used in the basic sampling-based planners? In which space are the samples generated?

12. A mad scientist designed a new sampling-based planner to find a path from $q_{init} \in \mathcal{C}_{free}$ to goal $q_{goal} \in \mathcal{C}_{free}$, where $\mathcal{C}_{free} \subseteq \mathcal{C}$ is the free region of the configuration space $\mathcal{C}$. Consider L is the classic python array.

```
$L = [q_{init}]$; // array of configurations
**while** *True* **do**
    $q$ = L[-1];
    $q_{rand}$ = random configuration from $\mathcal{C}$;
    $q_{new}$ is any configuration on the line-segment from $q$ to $q_{rand}$;
    $L$.append( $q_{new}$ );
    **if** $q_{new}$ *is close enough to* $q_{goal}$ **then**
        return $L$
    **end**
**end**
```

Analyze this planner and try to tell as much as you can about it. Find all possible pros/cons of the method.

Hints:

- Draw the first 4 steps of the method.
- Is the planner complete/probabilistic complete?
- For what kind of robots and maps it works? (show example)

- Draw example of robot+map combination, where it returns a valid path.
- Try to find combination of robot+map where the method may fail or cannot work at all.
- Is the resulting path always/sometimes/never optimal?
- Assume a path returned by this method. What can you say about clearance of the path?

13. You have n configurations in a set $S = \{q_1, \dots, q_n\}$, $q_i \in \mathcal{C}$, and a query point $q \in \mathcal{C}$. How do you find nearest neighbor from S to q efficiently? Assume a classic 3D Euclidean metric. Describe/illustrate the method and if it uses a special data structure, describe it. What is the complexity of finding the nearest neighbor?