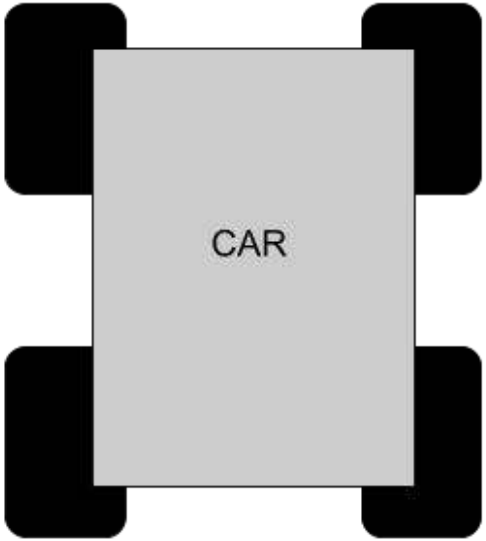
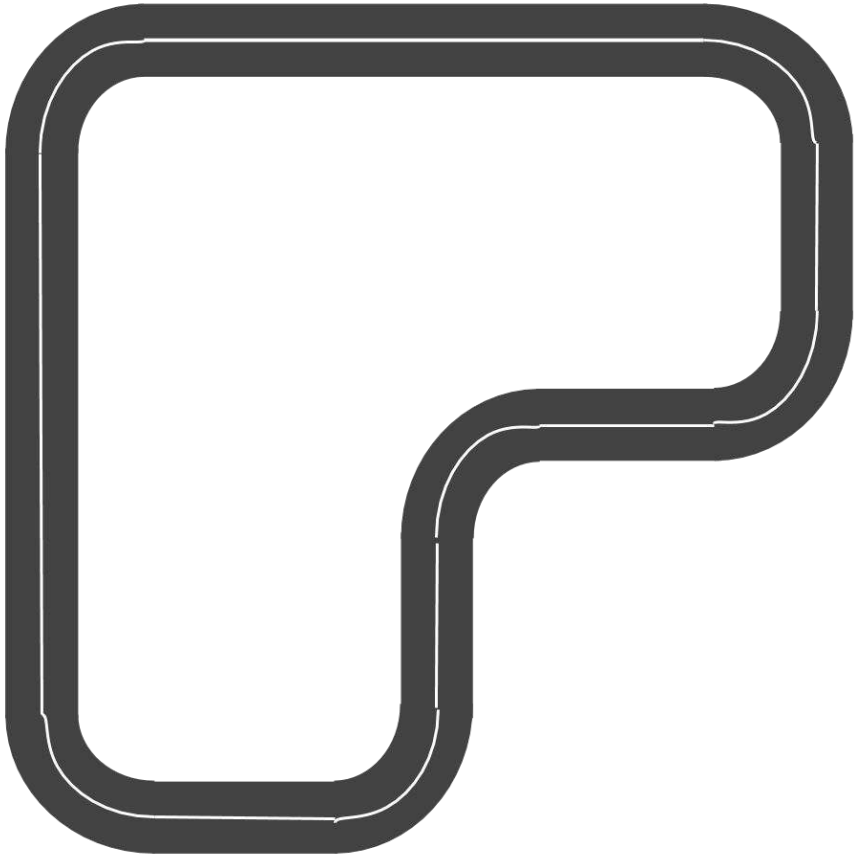
Localization Module

Abhinav & Ritika & Zuyong

What is localization?





Real World Applications

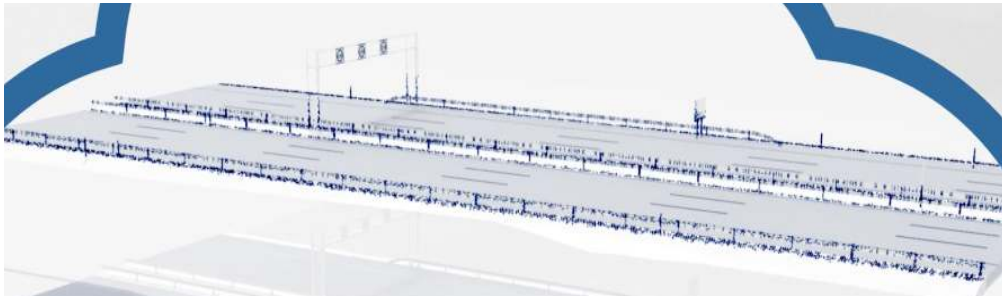
Autonomous Driving

Current Application

1) LiDAR
2) Cameras and other surround sensors
3) Satellite Navigation
4) Inertial Sensors

Road Signature

Using the features detected by the roadside a map layer is created.



Our Goals

1) Resourceful
2) Reliable
3) Real Time
4) Reconstructable

Our Implementation

Need:

1) Roadside signature equivalent
2) Cheaper sensors for small vehicle
3) Reliable detection of the features on the road

JSON

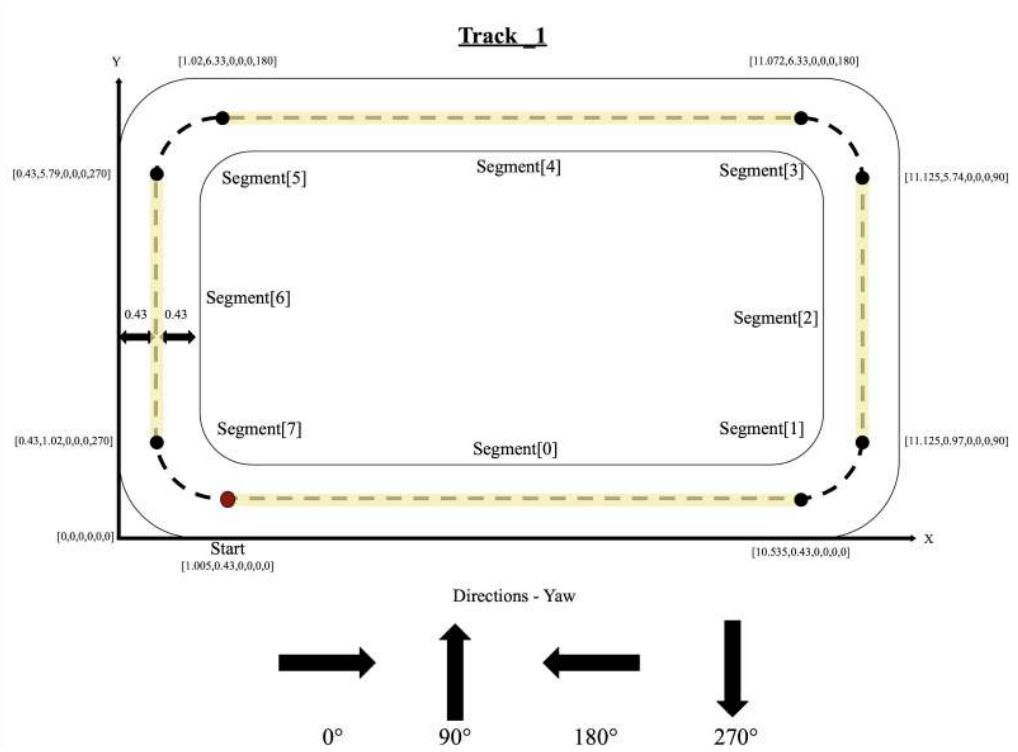
Implementation

Features with the car:

Config matrix

The track has 2 types of segments:

Straight lines & Turns



Feature Format (Config Matrix)

- 4 x 4 matrix

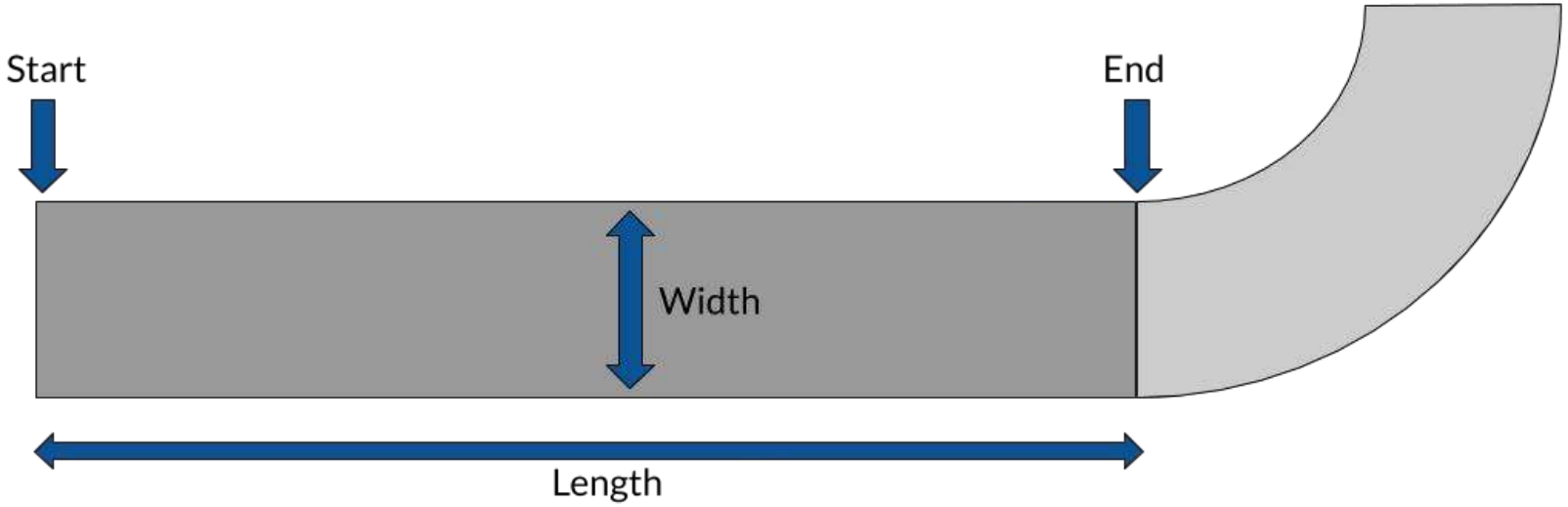
$$g = \begin{bmatrix} R & t \\ \mathbf{0} & 1 \end{bmatrix} \in \mathbb{R}^{4\times 4}$$

$$\text{where } R \in \mathbb{R}^{3\times 3},\ t \in \mathbb{R}^{3\times 1}, \mathbf{0} \text{ is a zero vector of size } 3$$

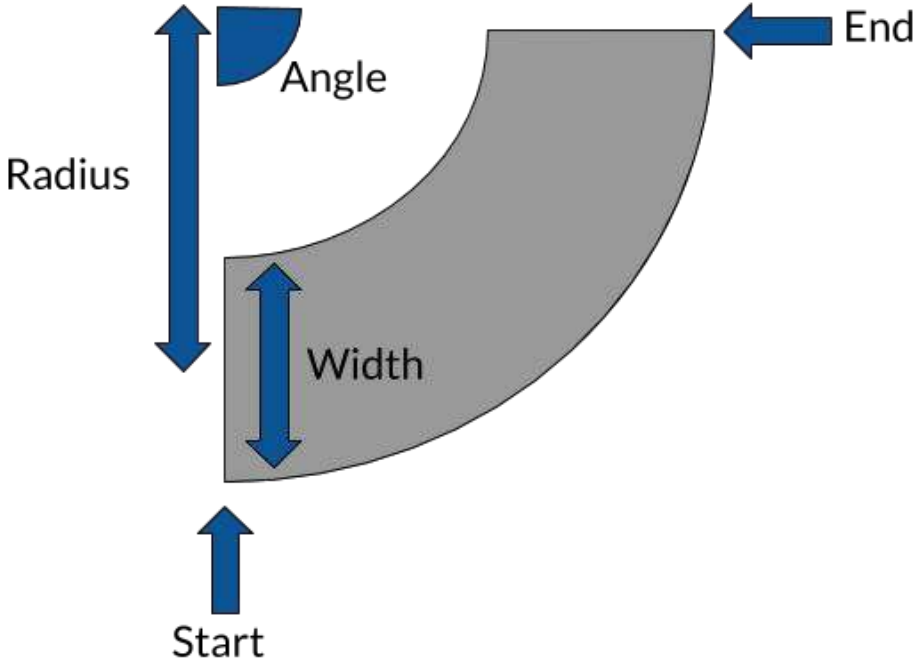
  - R is the rotation matrix
  - t is the translation vector

- Always sent as this matrix

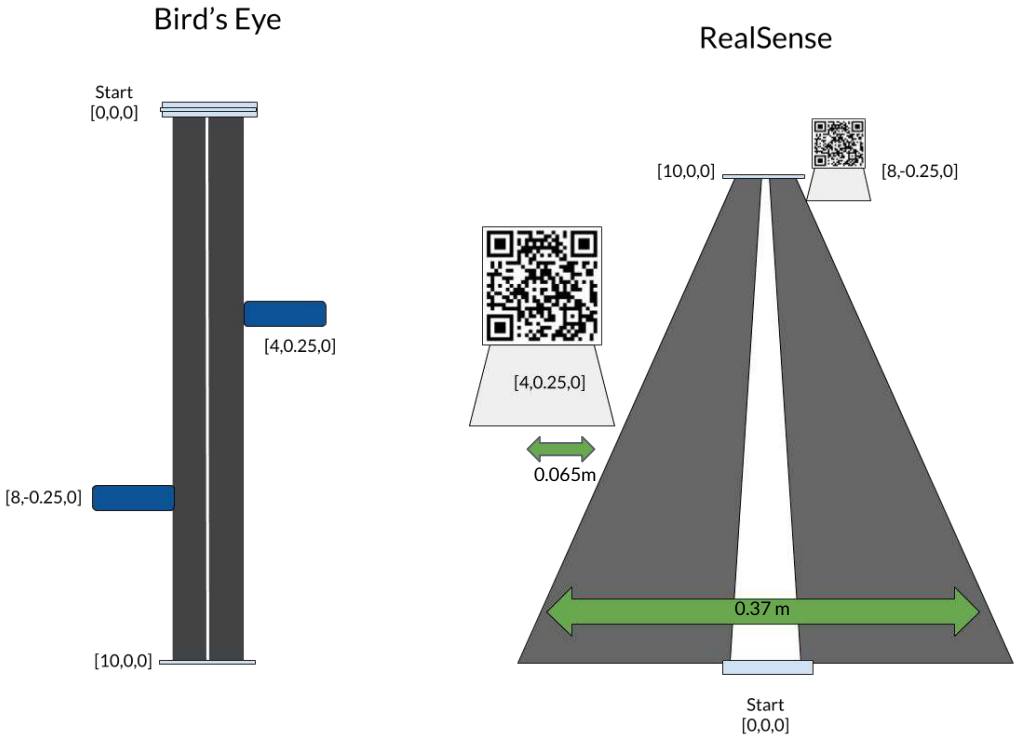
Track - Lines



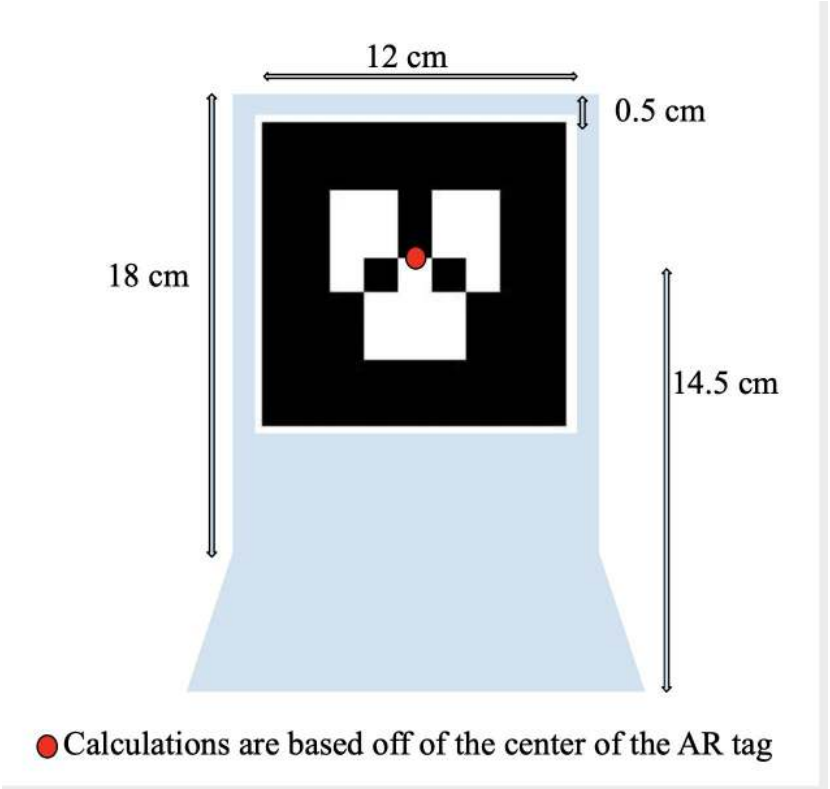
Track - Turns



AR Markers


Bird's Eye
Start
[0,0,0]
[4,0.25,0]
[8,-0.25,0]
[10,0,0]
RealSense
[10,0,0]
[8,-0.25,0]
[4,0.25,0]
0.065m
0.37 m
Start
[0,0,0]

AR Marker Stand Setup



JSON Map Standard

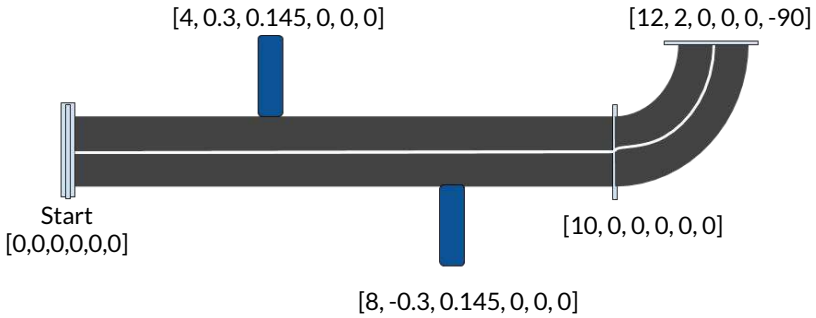

```
{
	"Comment": "",
	"AR parameters": {
			"Comment": "",
			"Width": 12,
			"Margin": 0.5,
			"Dimension": 6,
			"Size": 250
		},
	"Segments": [{
		"Angle": 0,
		"Radius": 0,
		"Length": 10,
		"Width": 0.5,
		"Start": [0,0,0,0,0,0],
		"End": [10,0,0,0,0,0],
		"AR Id": [1, 2]
	}],
	"AR tags": [{
			"Id": 1,
			"Location": [4, 0.3, 0.145,0,0,0],
			"Segment": 0
		}]
}
```


Consists of 4 parts:

1) Comment
   a) Explanation on how to understand the information in the JSON
2) AR parameters
   a) Comment - how to understand the components
   b) Width - in cm of the AR tag
   c) Margin - white space in cm surrounding the AR marker
   d) Dimension - bits of the AR Marker
   e) Size - number of markers
3) Segments
   a) Angle - angle of the segment (0 - straight segments)
   b) Radius - radius of a turn (0 - straight segments)
   c) Length - length of the segment (meters)
   d) Width - width of the track in the segment (meters)
   e) Start - [x, y, z,φ,ψ,Θ] global position of the start
   f) End - [x, y, z, φ,ψ,Θ] global position of the end
   g) AR Id - array with Ids for the AR markers found on segment
4) AR tags
   a) Id - Id of the AR markers
   b) Location - [x, y, z, φ,ψ,Θ] global position of tag
   c) Segment - index of the segment that the AR marker is found on

JSON Map Standard

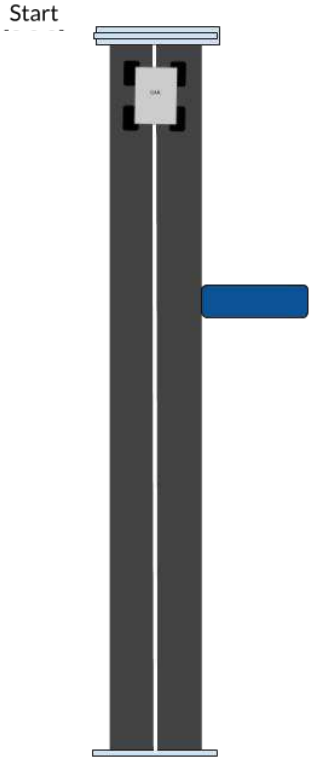
```
"Segments": [{
                "Angle": 0,
                "Radius": 0,
                "Length": 10,
                "Width": 0.5,
                "Start": [0,0,0,0,0,0],
                "End": [10,0,0,0,0,0],
                "AR Id": [1,2]
        },{
                "Angle": 90,
                "Radius": 2.82,
                "Length": 4.429656,
                "Width": 0.5,
                "Start": [10,0,0,0,0,0],
                "End": [12,2,0,0,0,-90],
                "AR Id": [3]
        }],
```



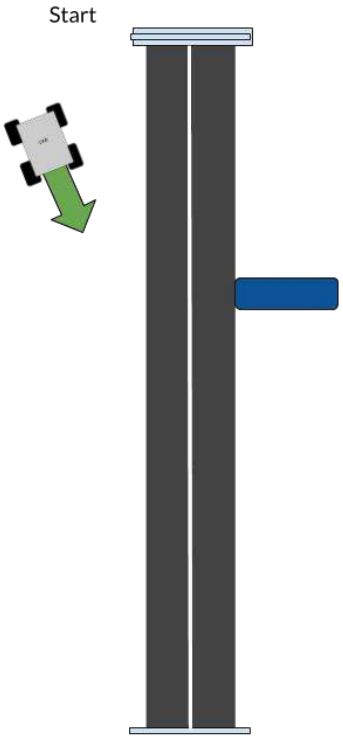
Localization Class

- Input: JSON map and the D435i image of the current frame
- run_threaded function:
  - Input: image array from the RealSense camera
  - Output: current localization within the map based on QR codes within the image.
- Situation that can be handled
  - One QR code seen
  - Multiple QR codes seen
  - No QR codes are seen
    - image keypoint stitching to find our change in position
- Things to do
  - Increase robustness of keypoint stitching

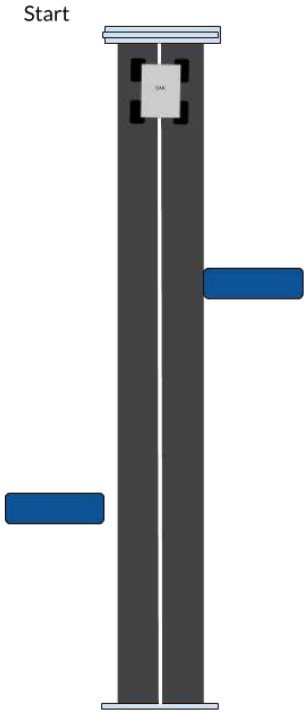
Localization Conditions



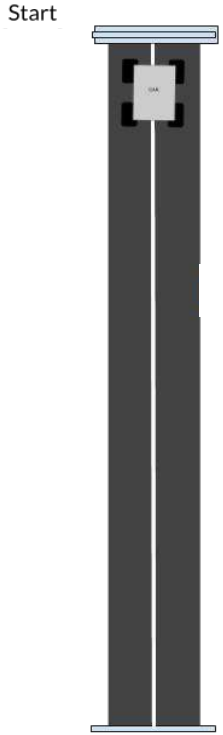
One AR Tag and On Track



One AR Tag and Off Track



Multiple AR Tags



No AR Tags

Configuration Matrix (one ar marker)

The car's configuration in the world frame is:

$$g_w^{car} = g_w^{AR} g_{AR}^{car}$$

The configuration of AR marker in the world frame is:

$$g_w^{AR} = \begin{bmatrix} R_z(\psi)R_y(\theta)R_x(\phi) \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} & t \\ \mathbf{0} & 1 \end{bmatrix}$$

- Assume extrinsic rotations
- Multiplying the matrix to convert the world coordinates to ar markers local coordinates

The configuration of car in the ar marker coordinates is computed by aruco

Configuration Matrix (multiple ar markers)

Let's say we have multiple config matrices calculated as in the last side:

$$g_w^{AR_1}, g_w^{AR_2}, g_w^{AR_3}$$

We can separated these into rotation matrices and translation vectors:

$$r_w^{AR_1}, r_w^{AR_2}, r_w^{AR_3} \qquad t_w^{AR_1}, t_w^{AR_2}, t_w^{AR_3}$$

We can calculate the weighted translation with linear interpolation as:

$$t_w^{AR} = (\frac{t_w^{AR_1}}{dist_{AR_1}} + \frac{t_w^{AR_2}}{dist_{AR_2}} + \frac{t_w^{AR_3}}{dist_{AR_3}})/(\frac{1}{dist_{AR_1}} + \frac{1}{dist_{AR_2}} + \frac{1}{dist_{AR_3}})$$

Via spherical interpolation, we can also calculate rotation interpolation r_w^{AR}

Configuration Matrix (no ar marker)

- Feature detection and extraction using FAST and BRIEF
- Feature matching using brute force and hamming distance
- Compute 3D points for matches

$$z_{3d}\begin{bmatrix} x_{2d} \\ y_{2d} \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & t_x \\ 0 & f_y & t_y \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} x_{3d} \\ y_{3d} \\ z_{3d} \end{bmatrix} \implies \begin{bmatrix} x_{3d} \\ y_{3d} \\ z_{3d} \end{bmatrix} = z_{3d}\begin{bmatrix} f_x & 0 & t_x \\ 0 & f_y & t_y \\ 0 & 0 & 1 \end{bmatrix}^{-1}\begin{bmatrix} x_{2d} \\ y_{2d} \\ 1 \end{bmatrix}$$

- Using Ransac to find the rigid body transformation
  - Compute the rigid body transformation using only 3 pairs of points, http://nghiaho.com/?page_id=671
  - Compute the overall accuracy and choose the highest accuracy
  - Use all the inlier points corresponding to the high accuracy to compute a new transformation
  - If number of inliers is smaller than the threshold, no valid rigid body transformation will be returned
- Estimate current configuration $g_w^{curr} = g_w^{prev} g_{prev}^{curr}$

Current Attributes and Functions

Class Localization

Attributes:

- `json_in` : str
  - Will be the JSON file location for the map
- `map` : dictionary
  - calls and saves information from json.loads(json_in)
- `qr_loc` : array of QR codes from JSON str

Methods:

`__init__`(str json_in, camera_mtx, dist_coeffs)

`run_threaded`(D435i image):
- Threaded func that calls get_position
- Return config detected

`get_position`(D435i image)
- Config matrix represent the global position of the car on the map
- Returns config matrix, detected
- Returns previous config matrix if not QR detected

`avg_positions`(list of config matrices, one for each QR code)
- Returns average weighted by distance of the global interpolated translation and global interpolated rotation

Implementation

- CV2 aruco functions to extract the QR codes and get the rotation and translation information
- Reference the global location and the angle of each detected QR code
- Determine the global interpolated translation and global interpolated rotation.
  - Weighted average for multiple QR codes based on the distance from the QR code
- Return the global config matrix (R and T) of the car

Runs independent of IMU and other functions of the car.