

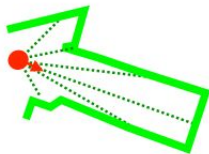


# Introduction to Graph-based SLAM

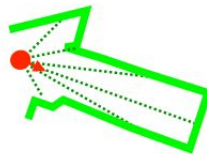
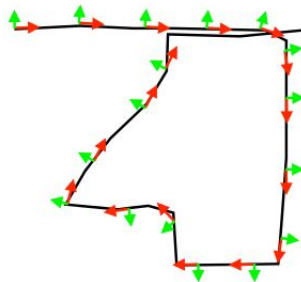
---

Hongrui Zheng (hongruiz@seas.upenn.edu)

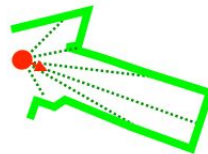
# Problem Setting



**Localization:** given a *map*, use *sensor data* to estimate the current *pose* of the robot



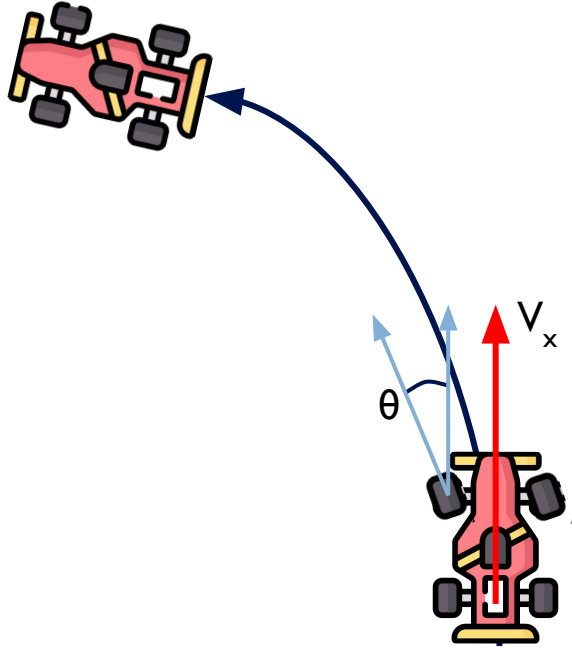
**Mapping:** given robot pose at each time (*trajectory*), use *sensor data* to build map



**Simultaneous Localization and Mapping (SLAM):**

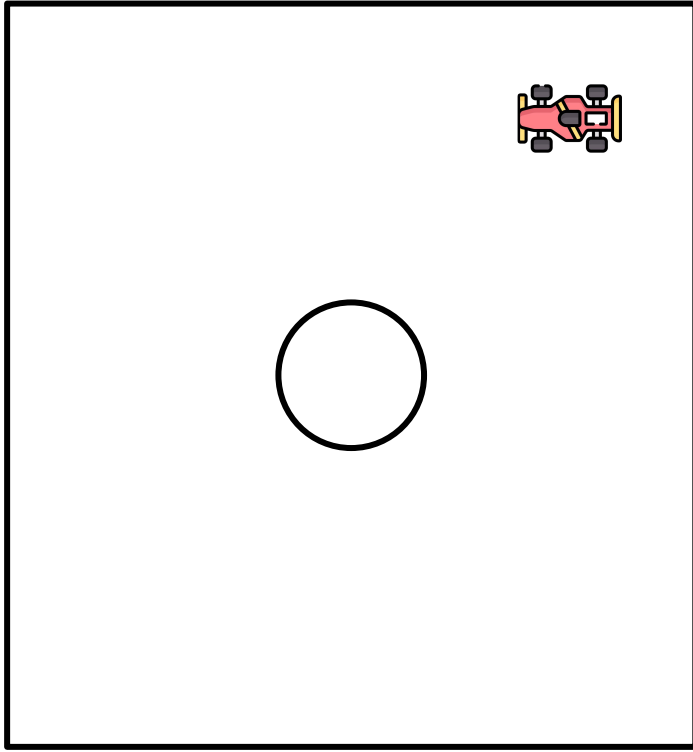
use sensor data to build map and estimate robot trajectory

# Dead Reckoning

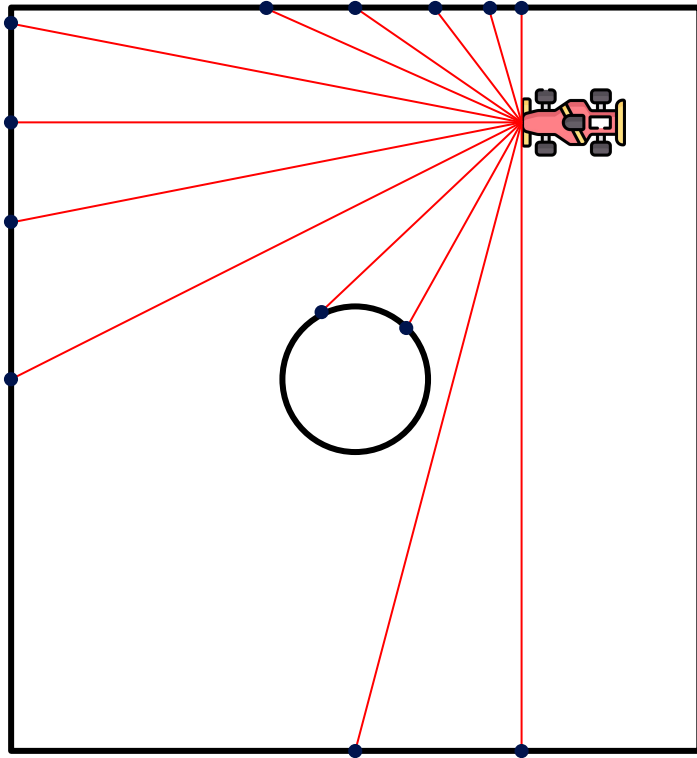


- Calculating current position of a moving robot based on a previously determined position.
- In our case, the VESC determines the current ERPM of the motor, and converts it into the vehicle's longitudinal velocity. Then, the servo angle is used to determine the current steering angle.
- Using a basic kinematics model, the relative position of the car can be determined. This is odometry.

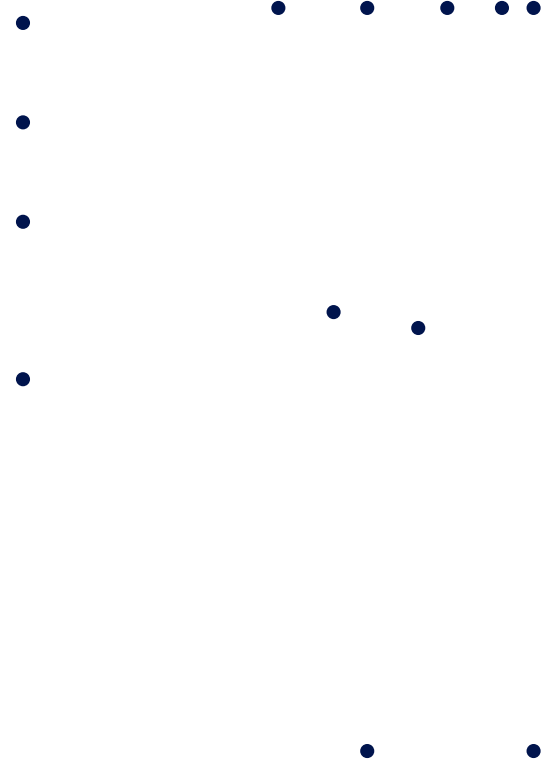
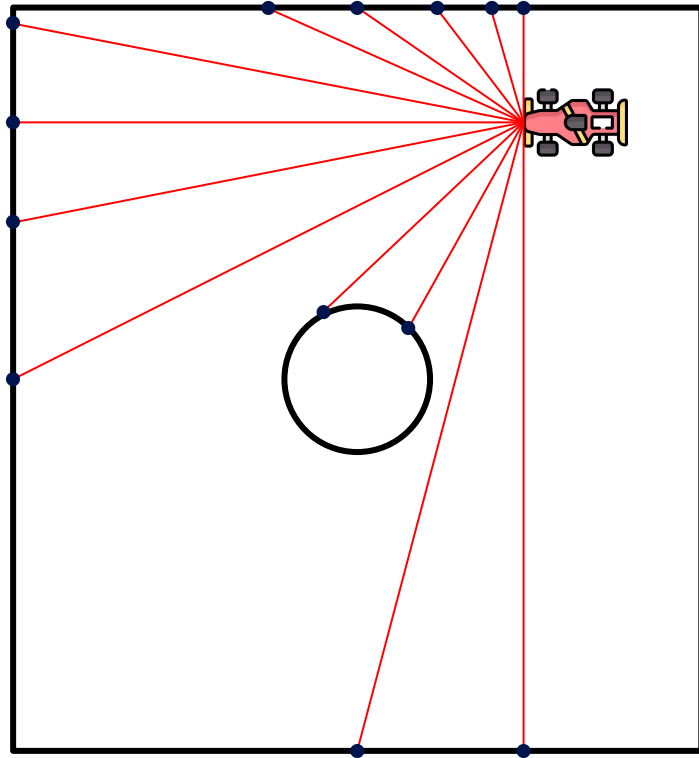
# Sensing the environment



# Sensing the environment

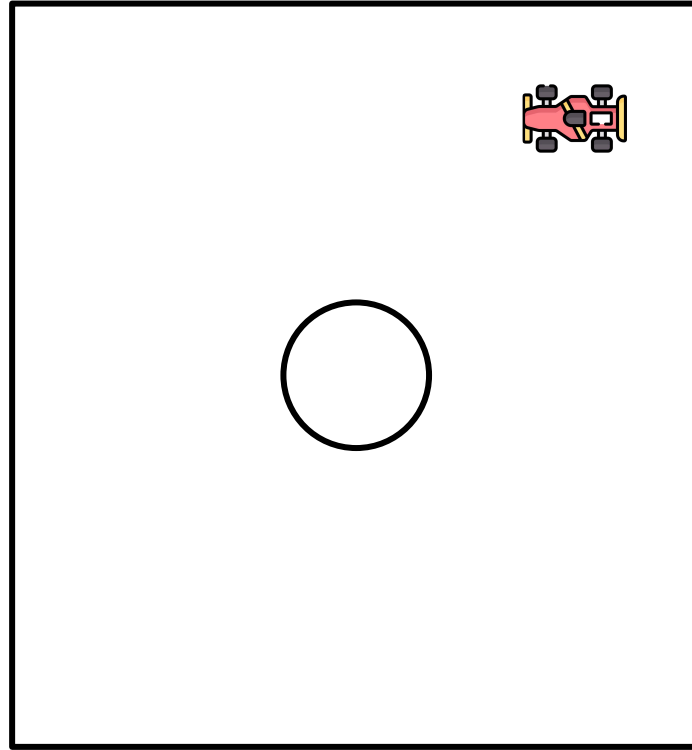


# Sensing the environment



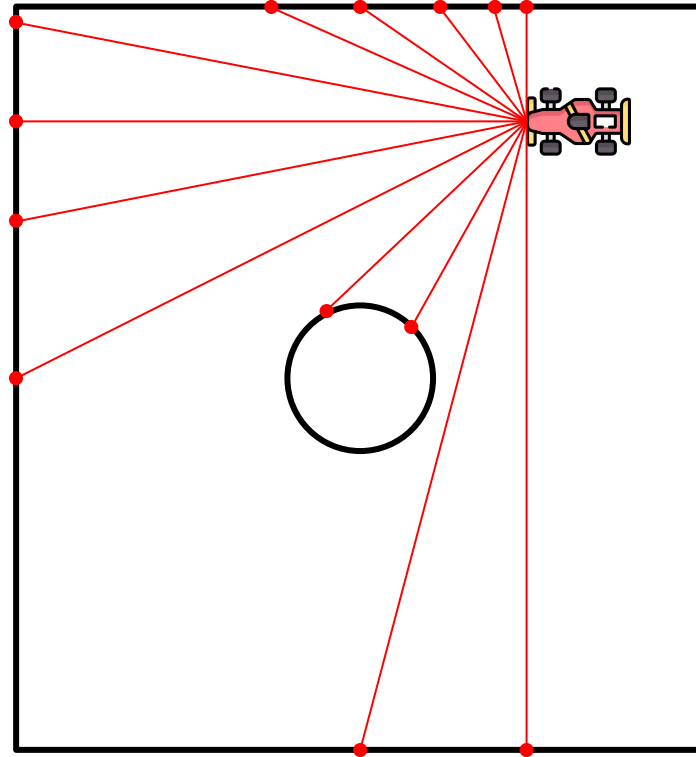
Alright, let's try to make a map.

# In an ideal world...

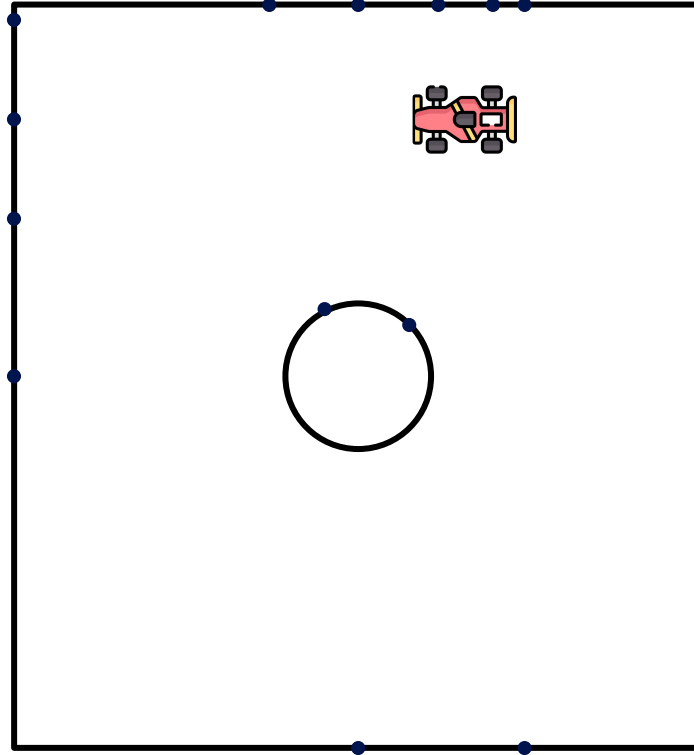




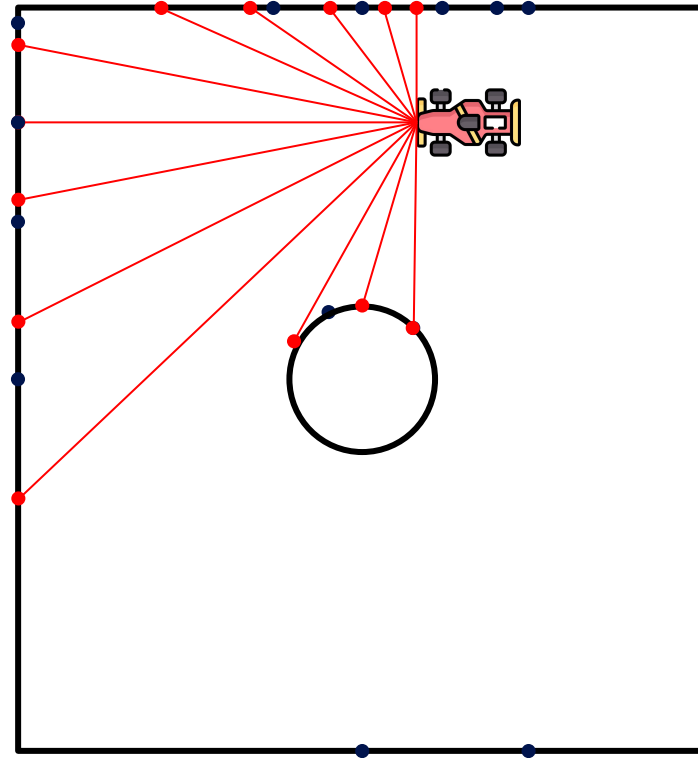
# In an ideal world...



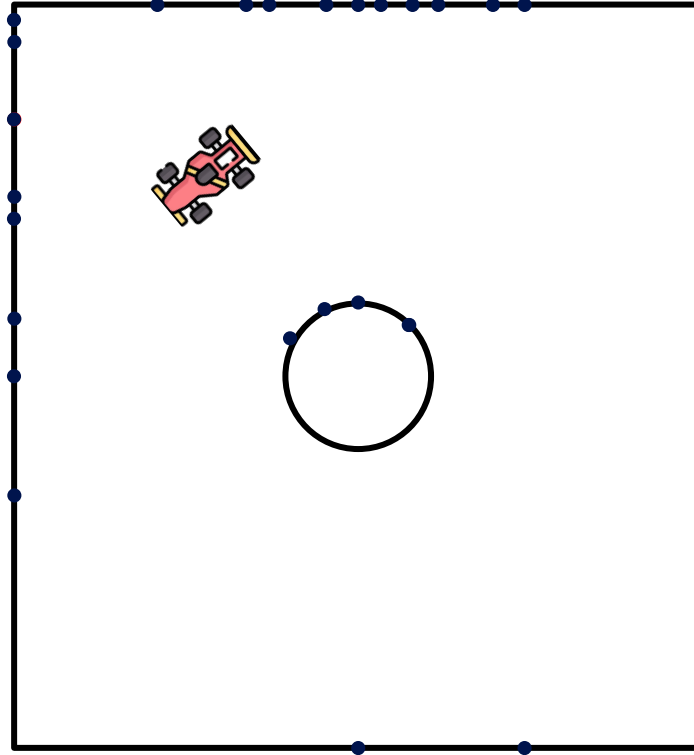
# In an ideal world...



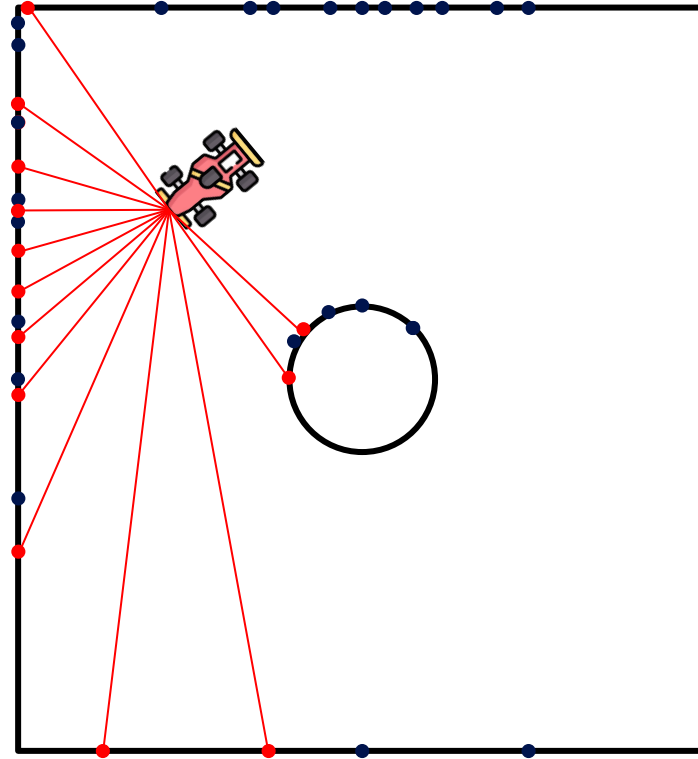
# In an ideal world...



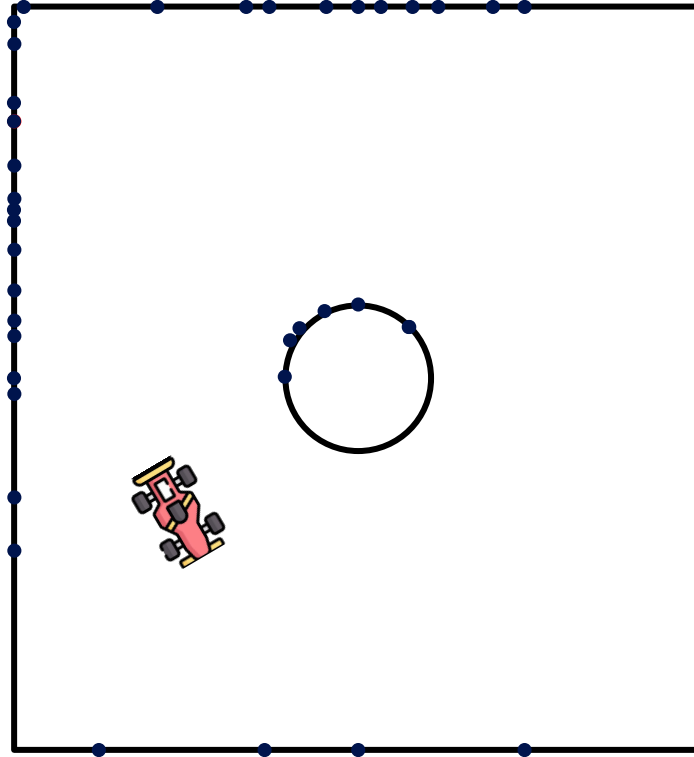
# In an ideal world...



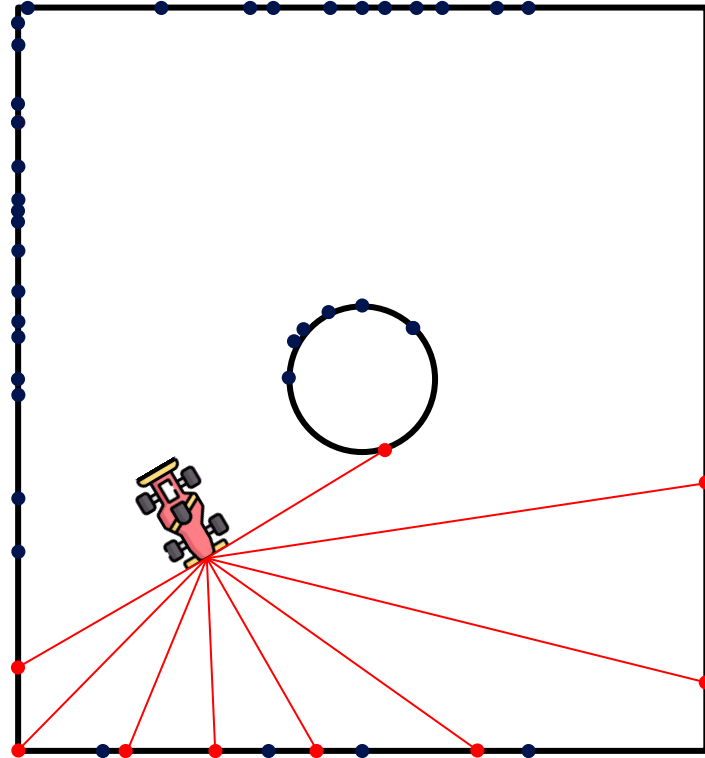
# In an ideal world...



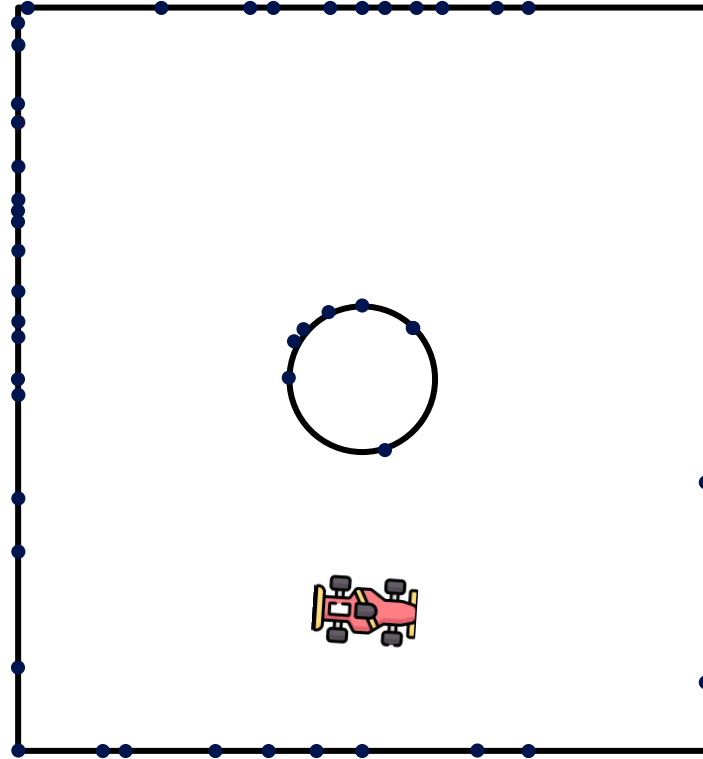
# In an ideal world...



# In an ideal world...

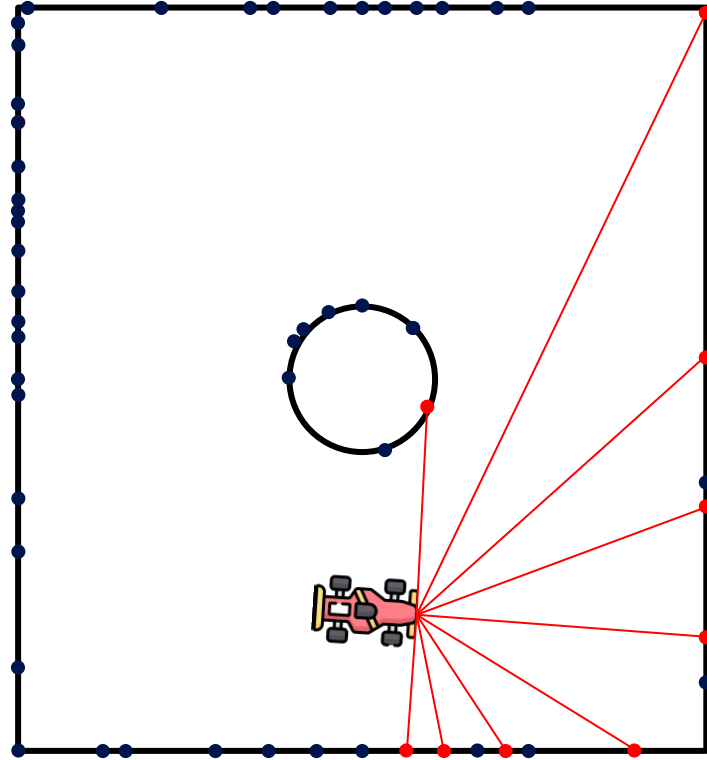


# In an ideal world...

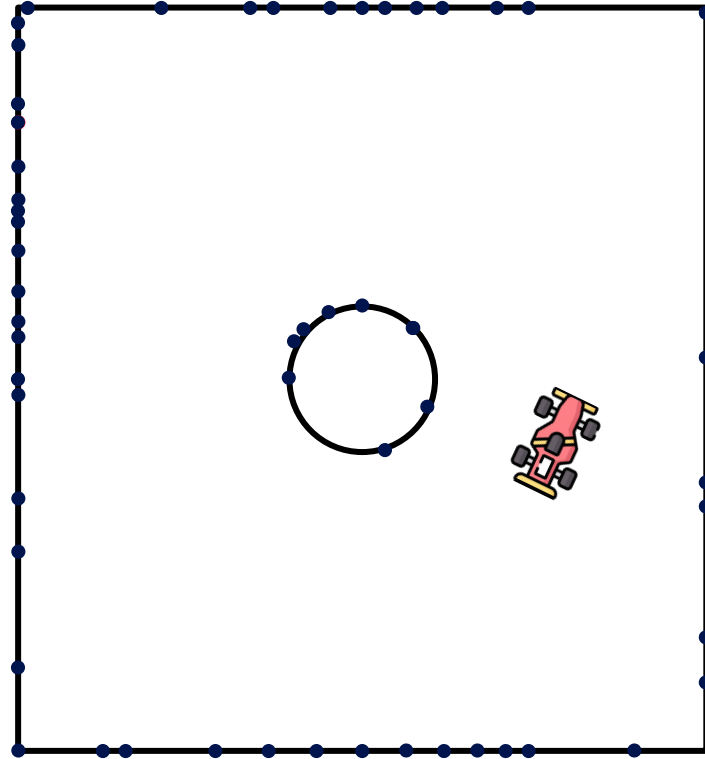




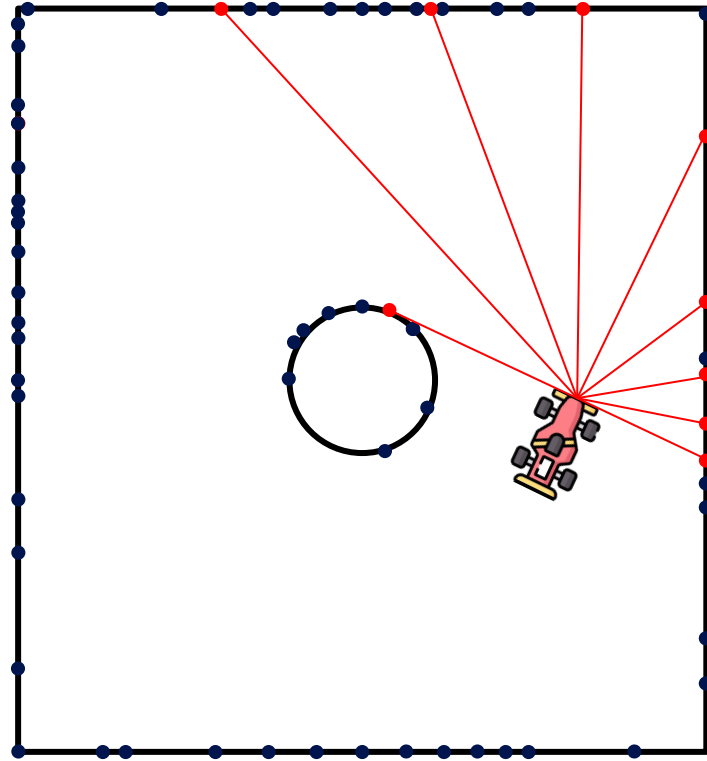
# In an ideal world...



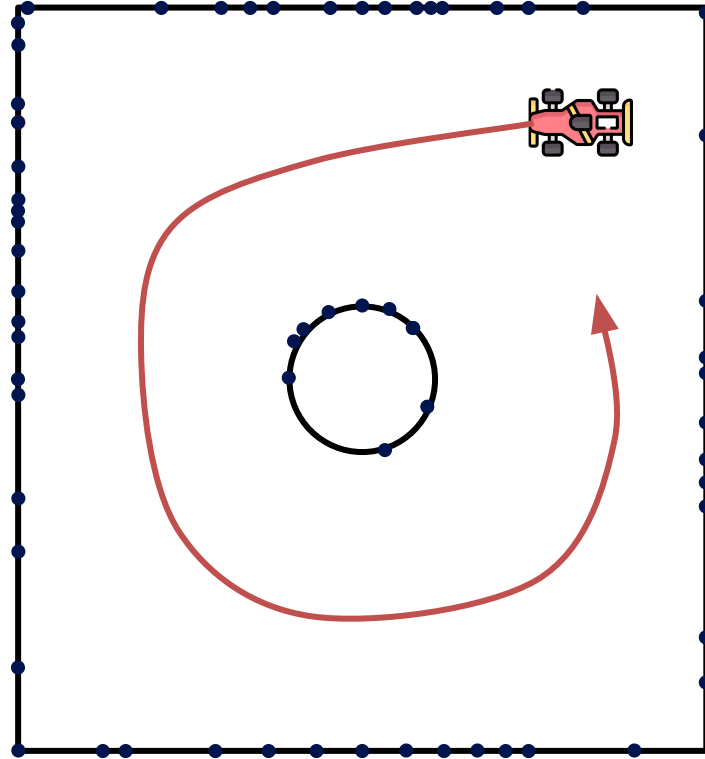
# In an ideal world...



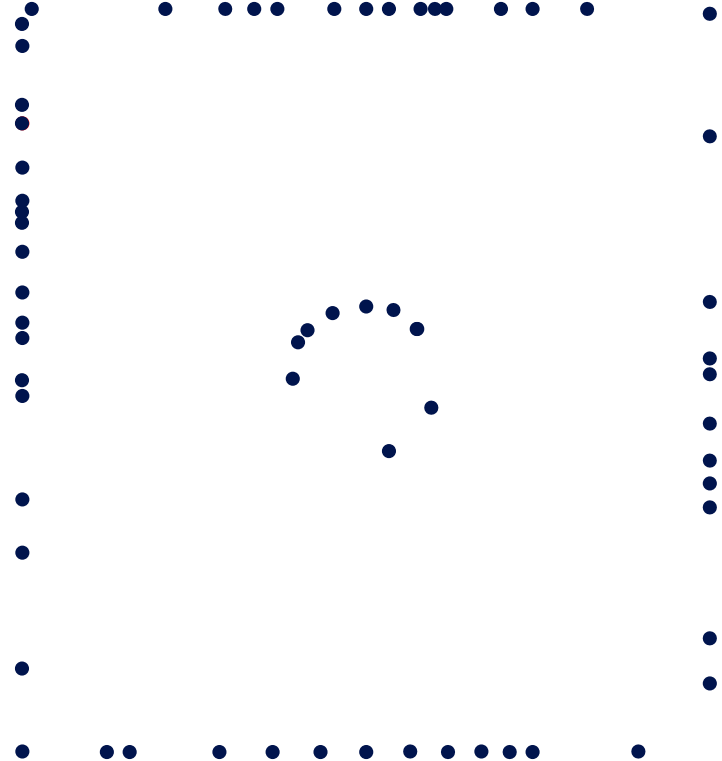
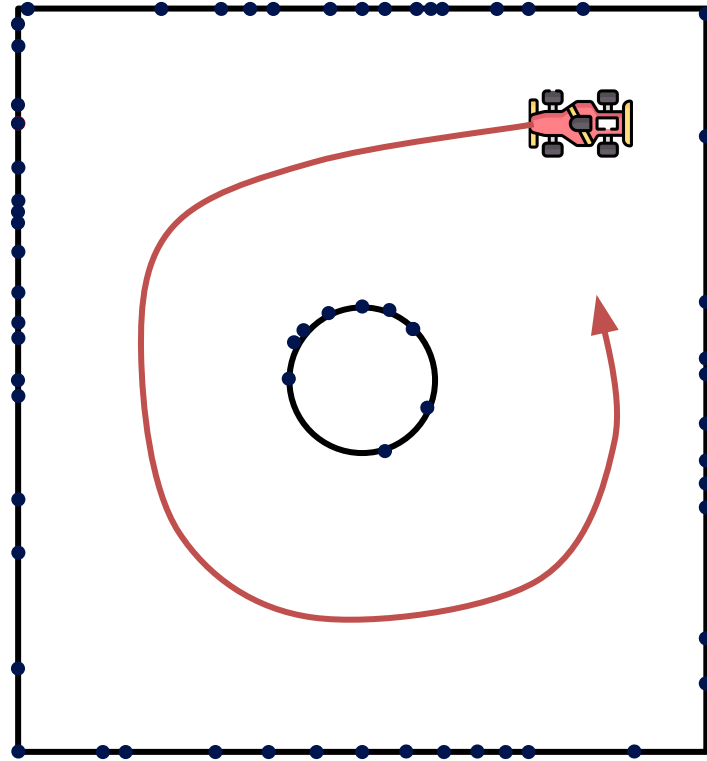
# In an ideal world...



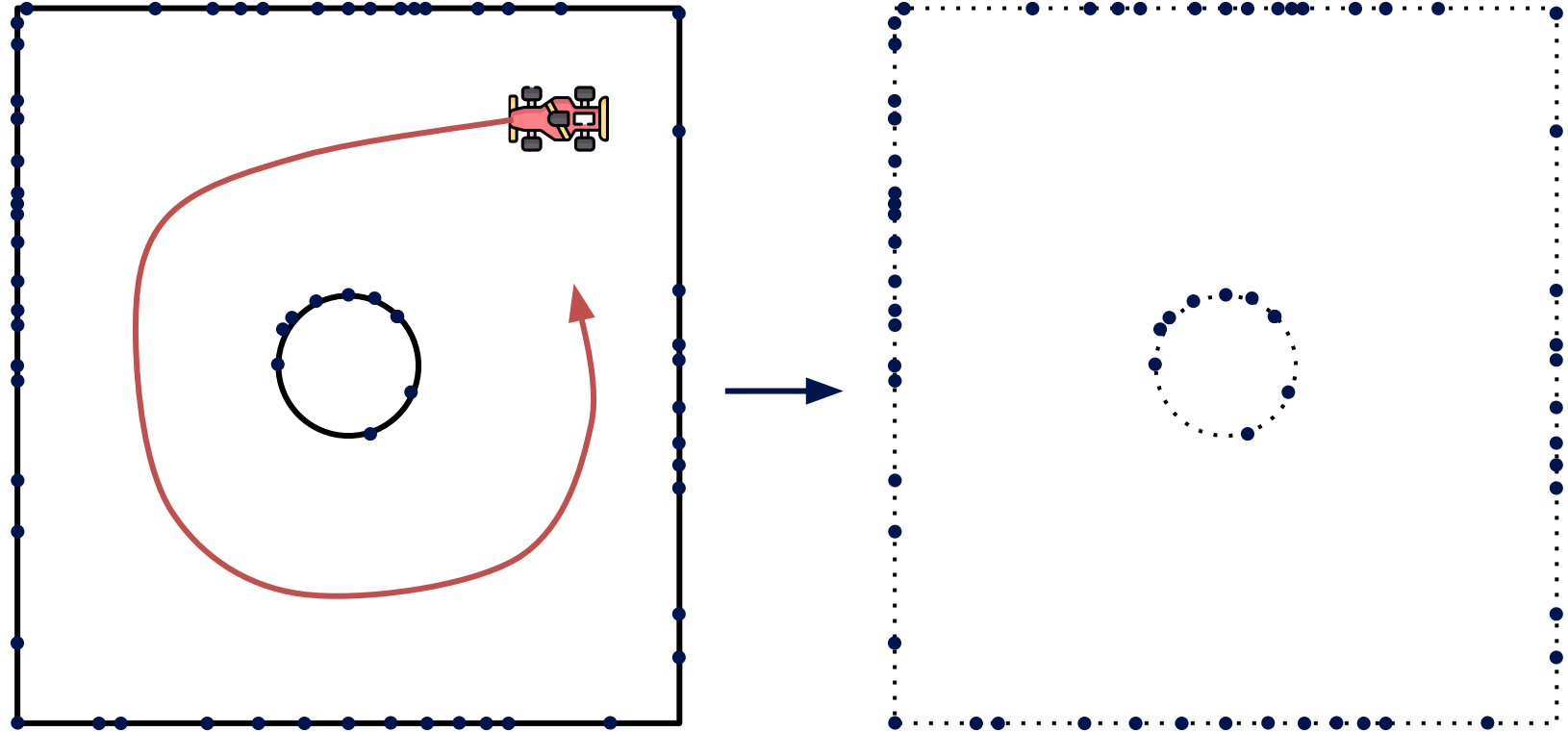
# In an ideal world...



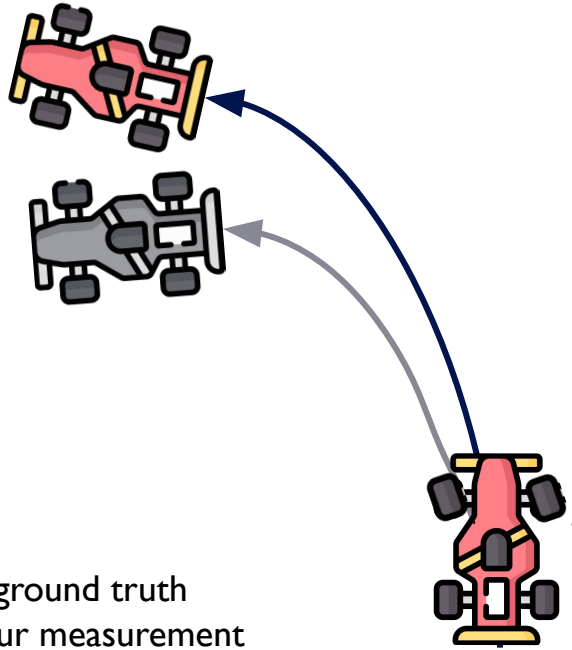
# In an ideal world...



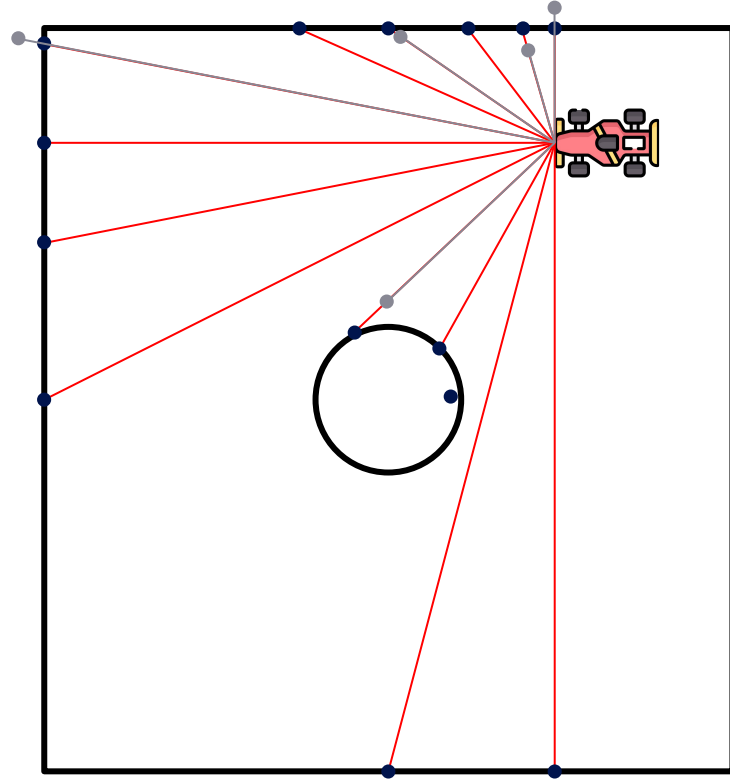
# In an ideal world...



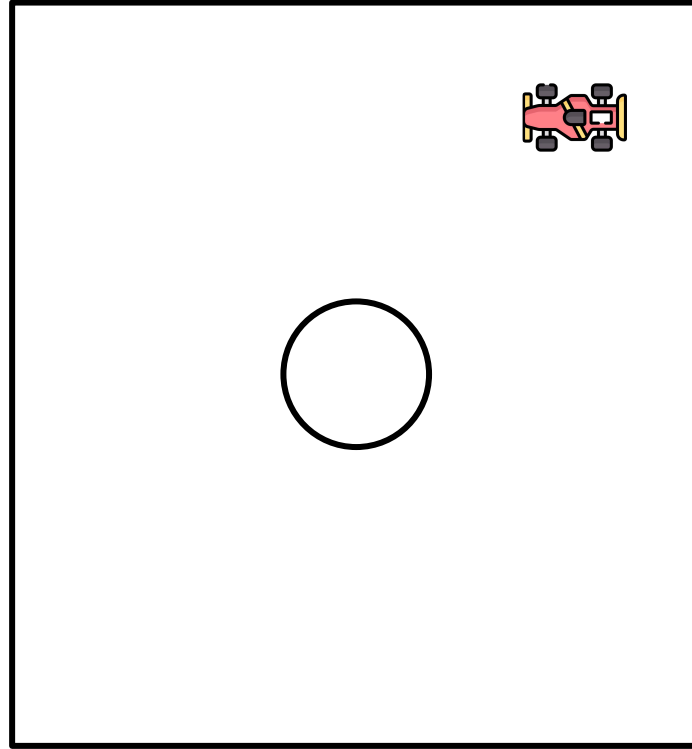
# But in the real world...



Color: ground truth  
Gray: our measurement

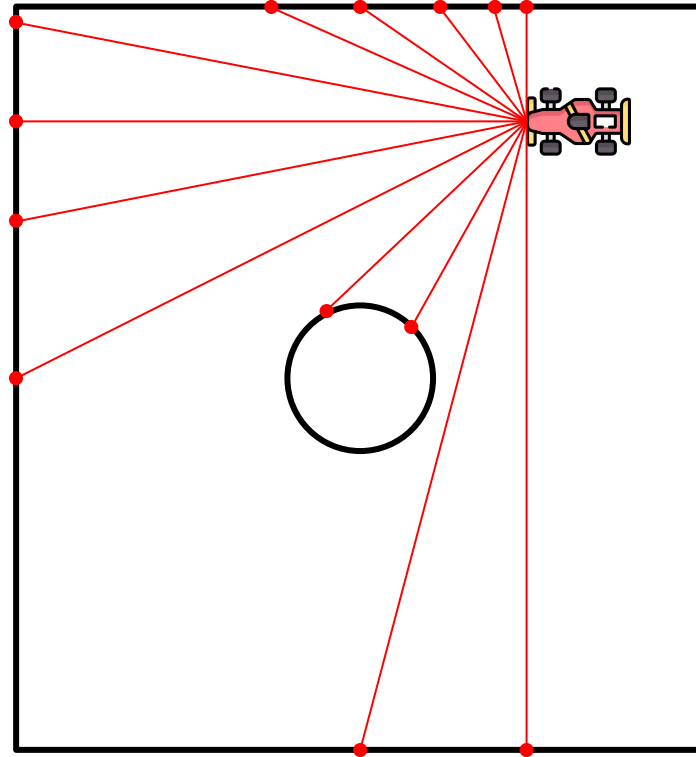


# But in the real world...

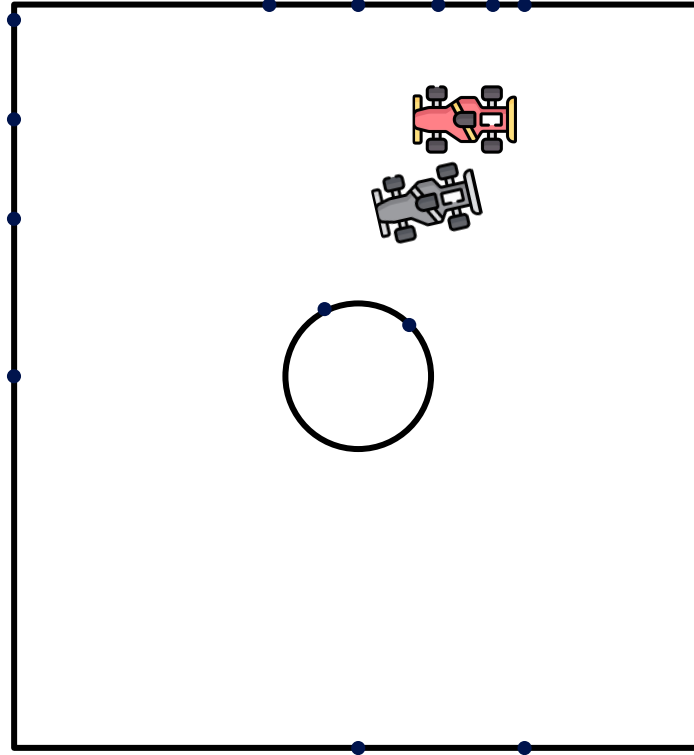




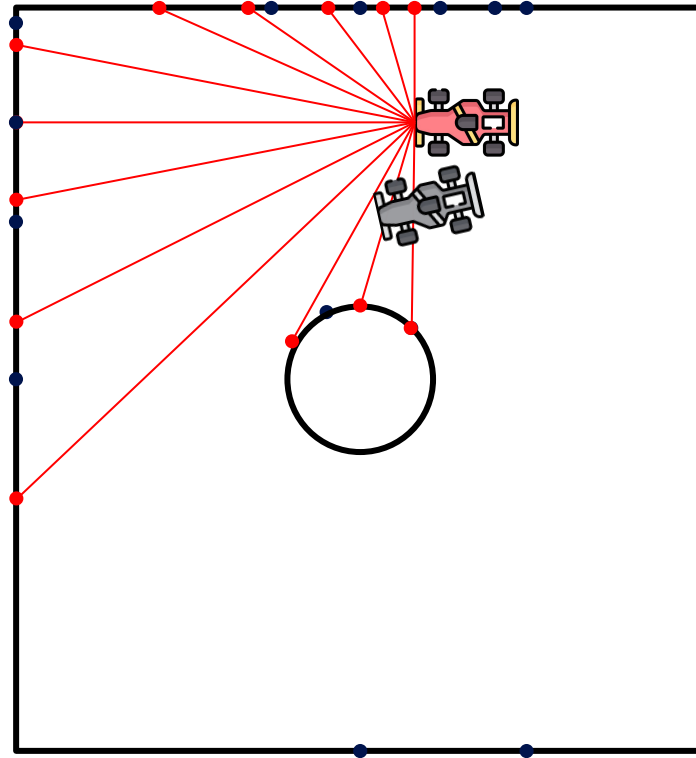
# But in the real world...



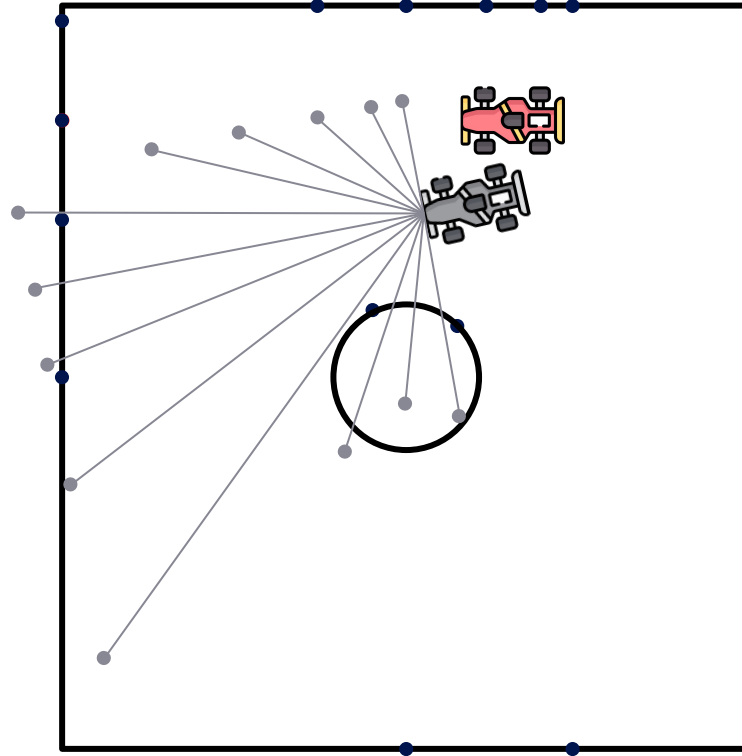
# But in the real world...



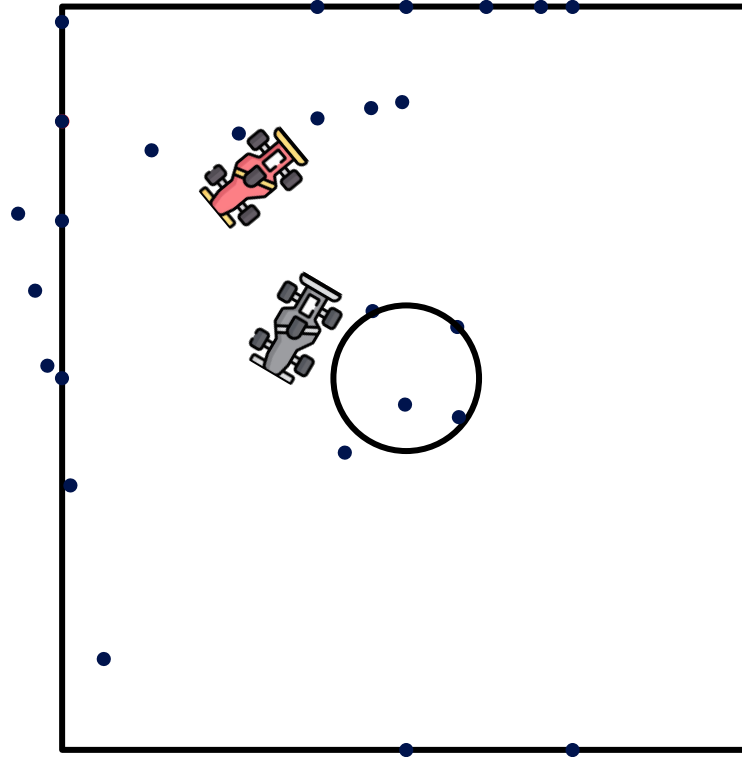
# But in the real world...



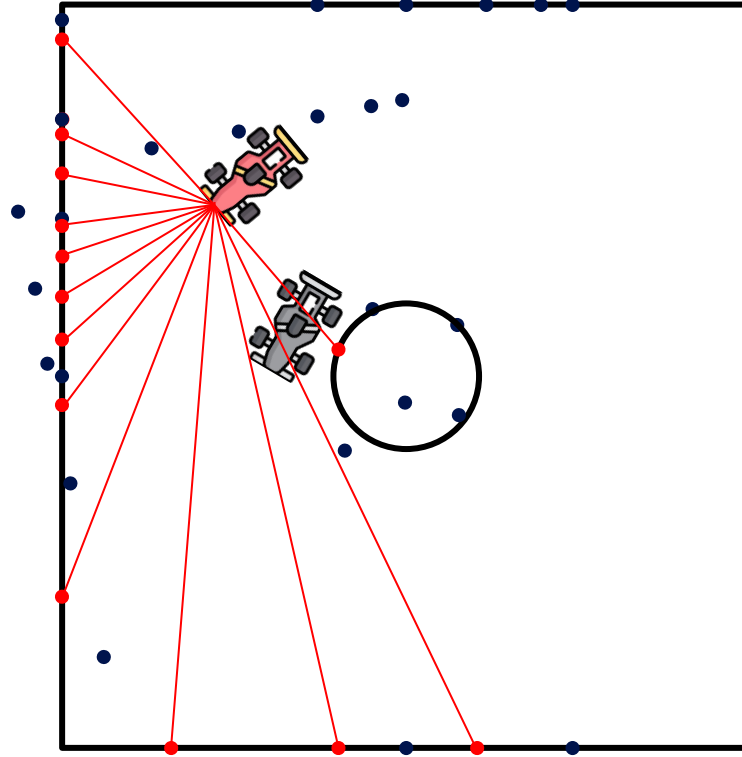
# But in the real world...



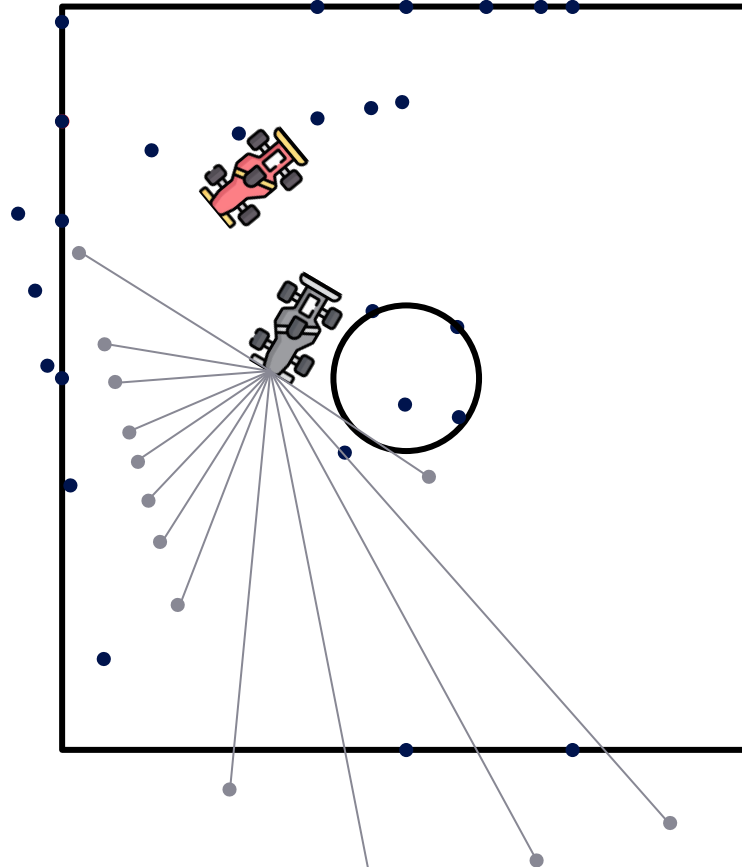
# But in the real world...



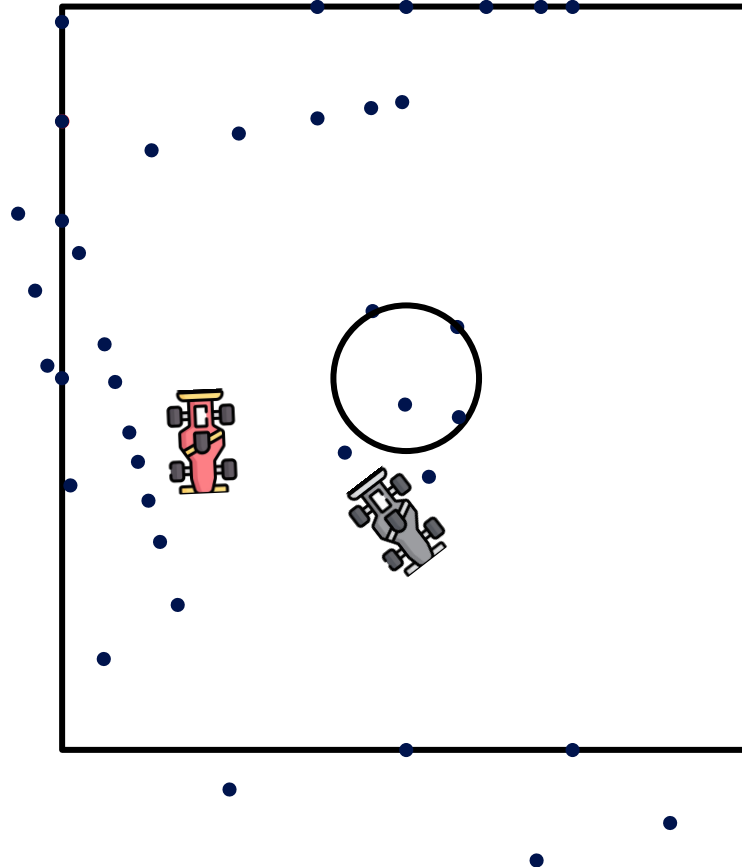
# But in the real world...



# But in the real world...

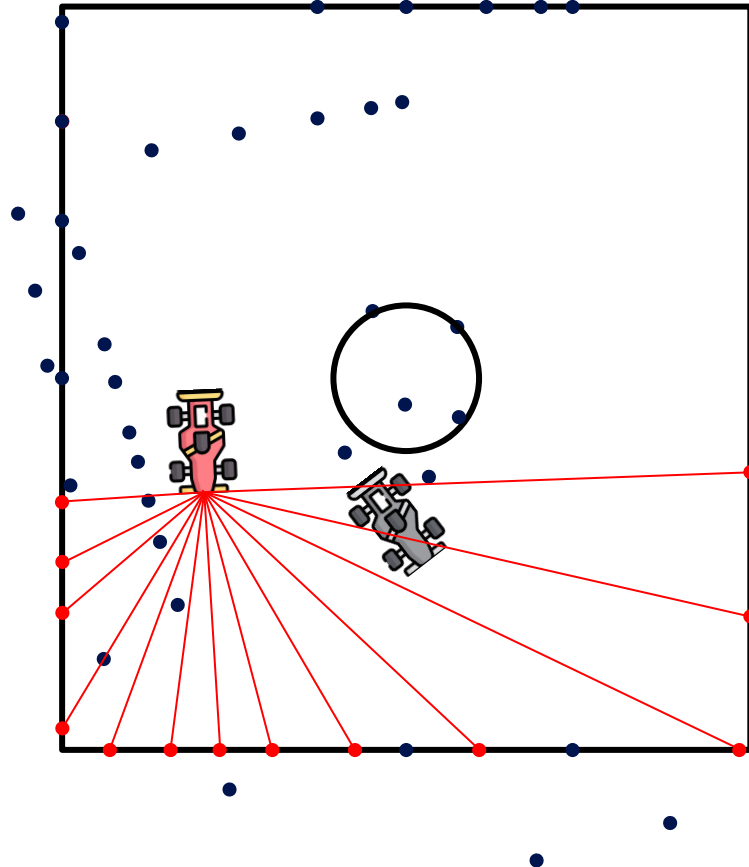


# But in the real world...

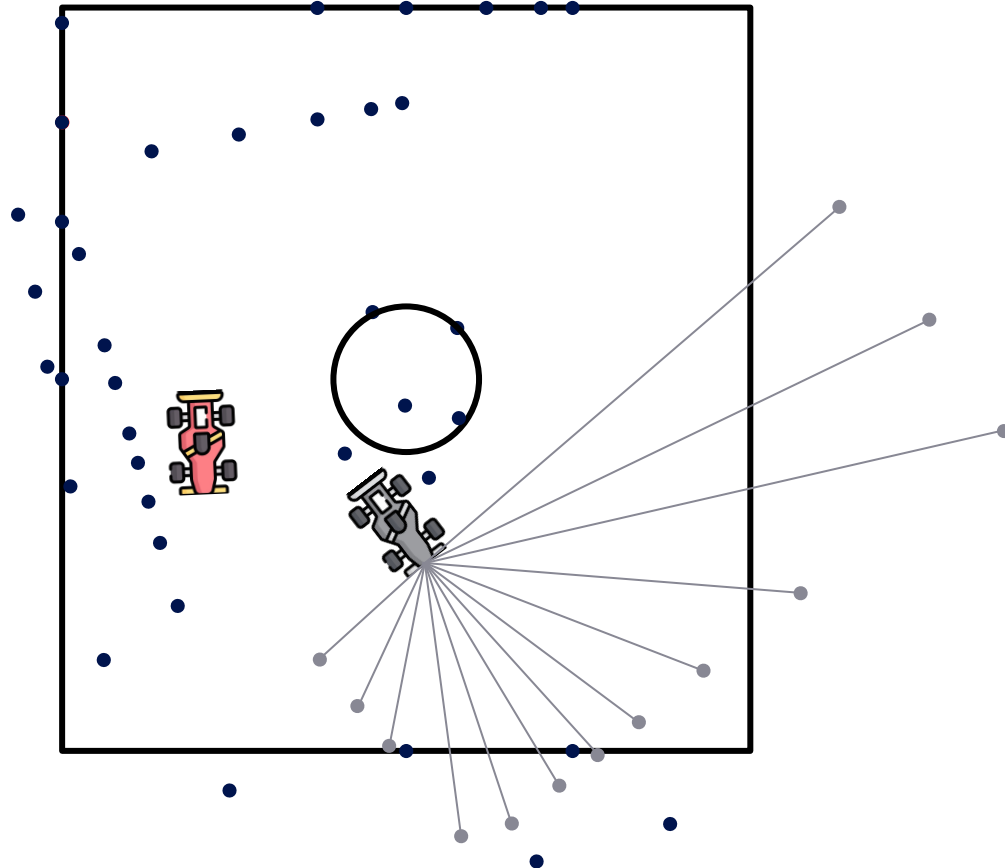




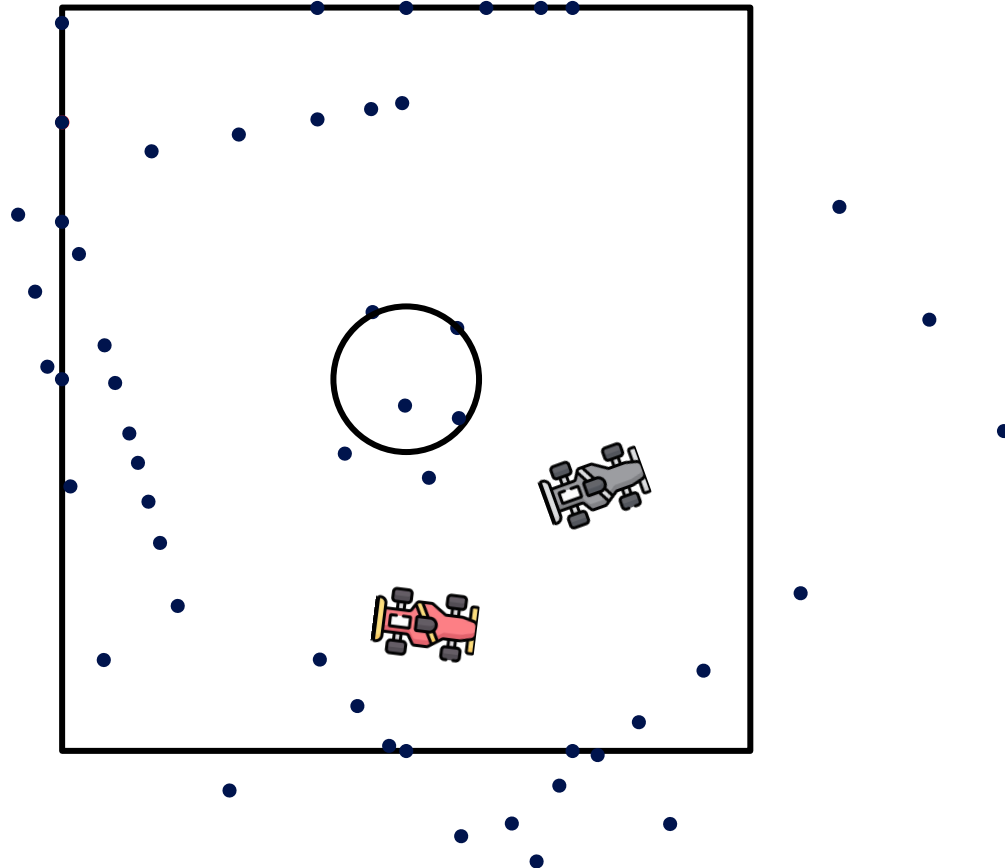
# But in the real world...



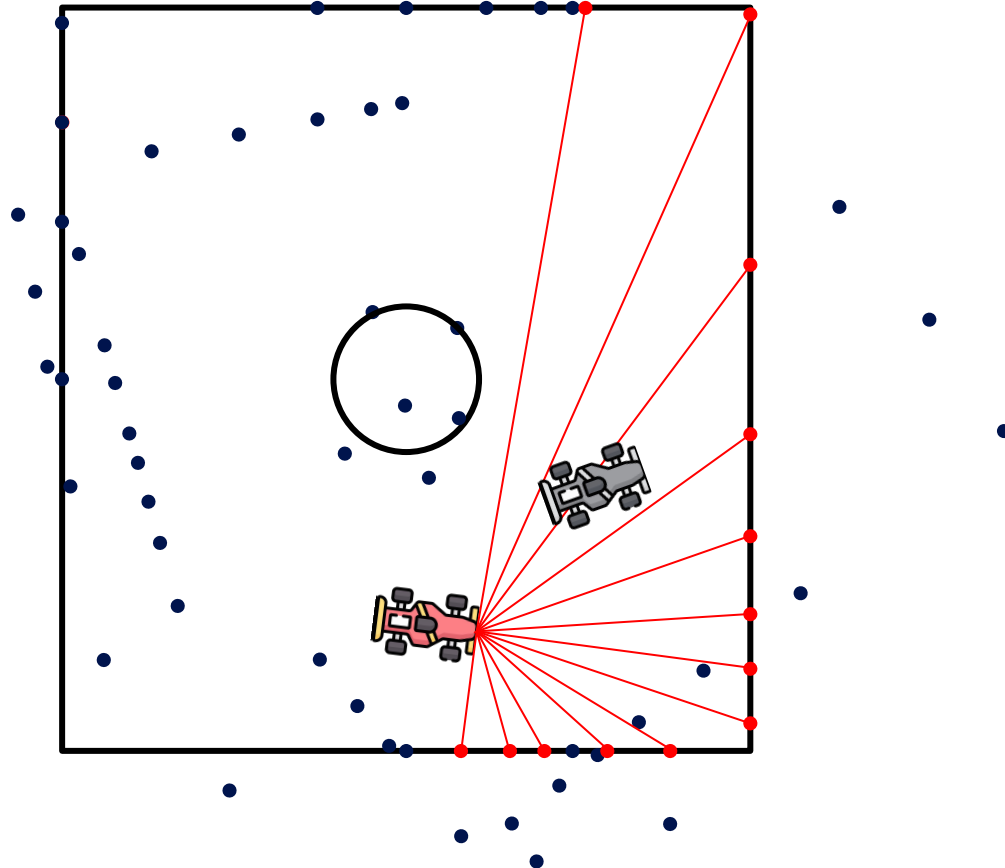
# But in the real world...



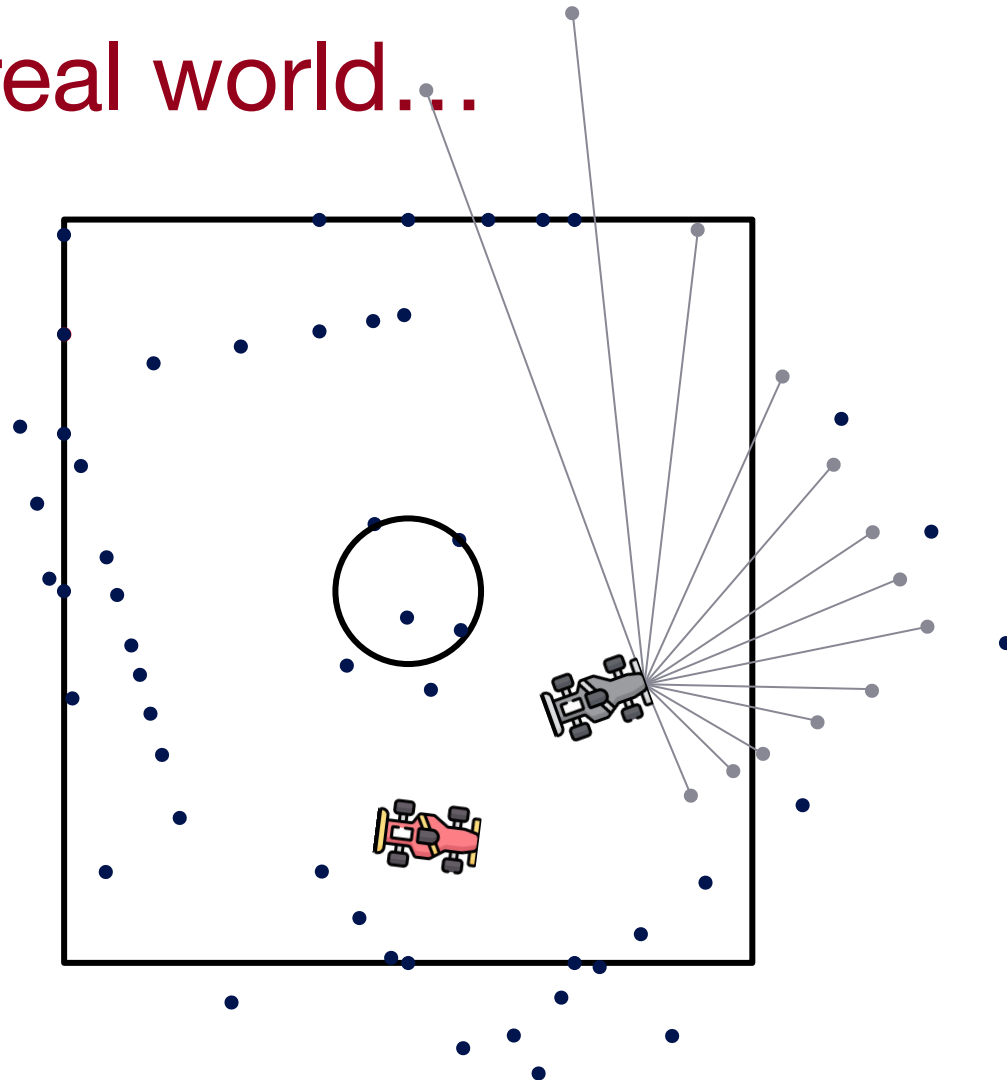
# But in the real world...



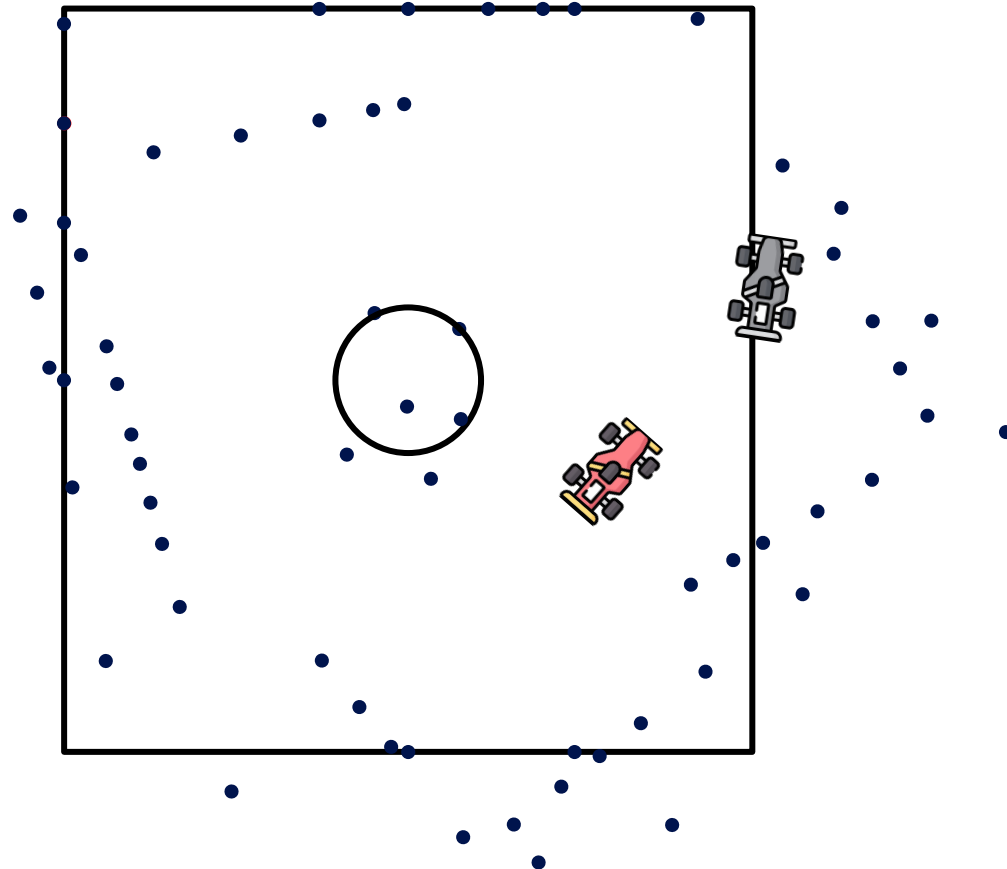
# But in the real world...



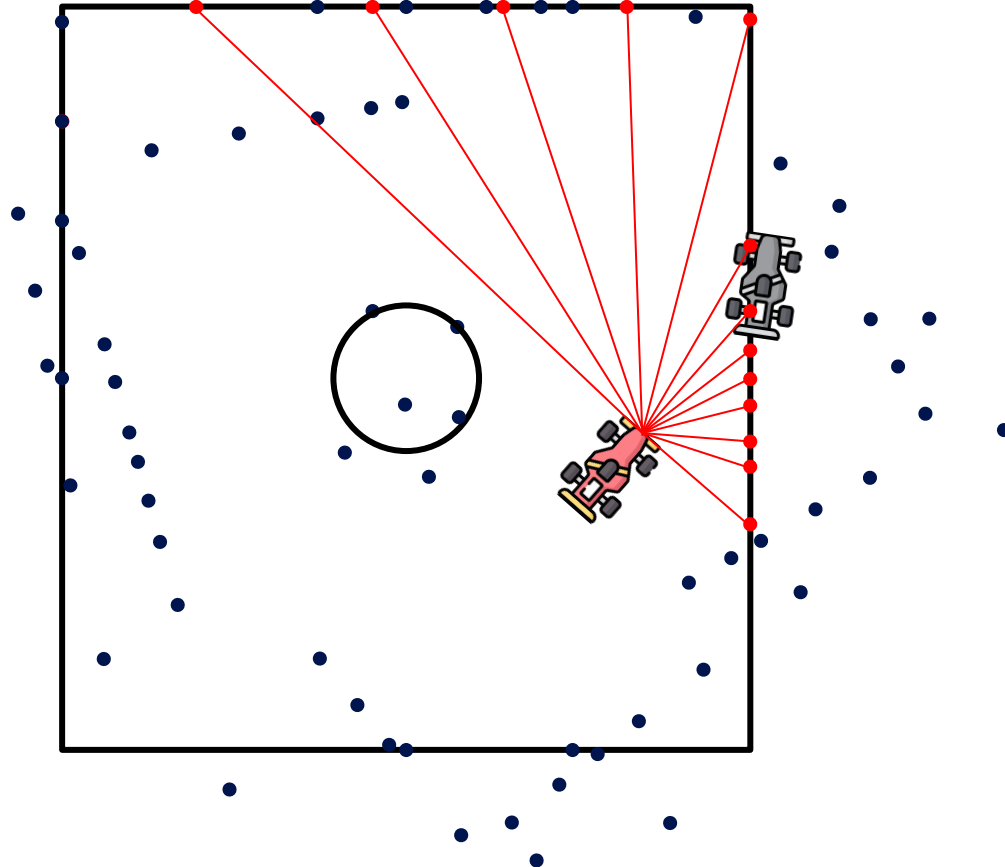
# But in the real world...



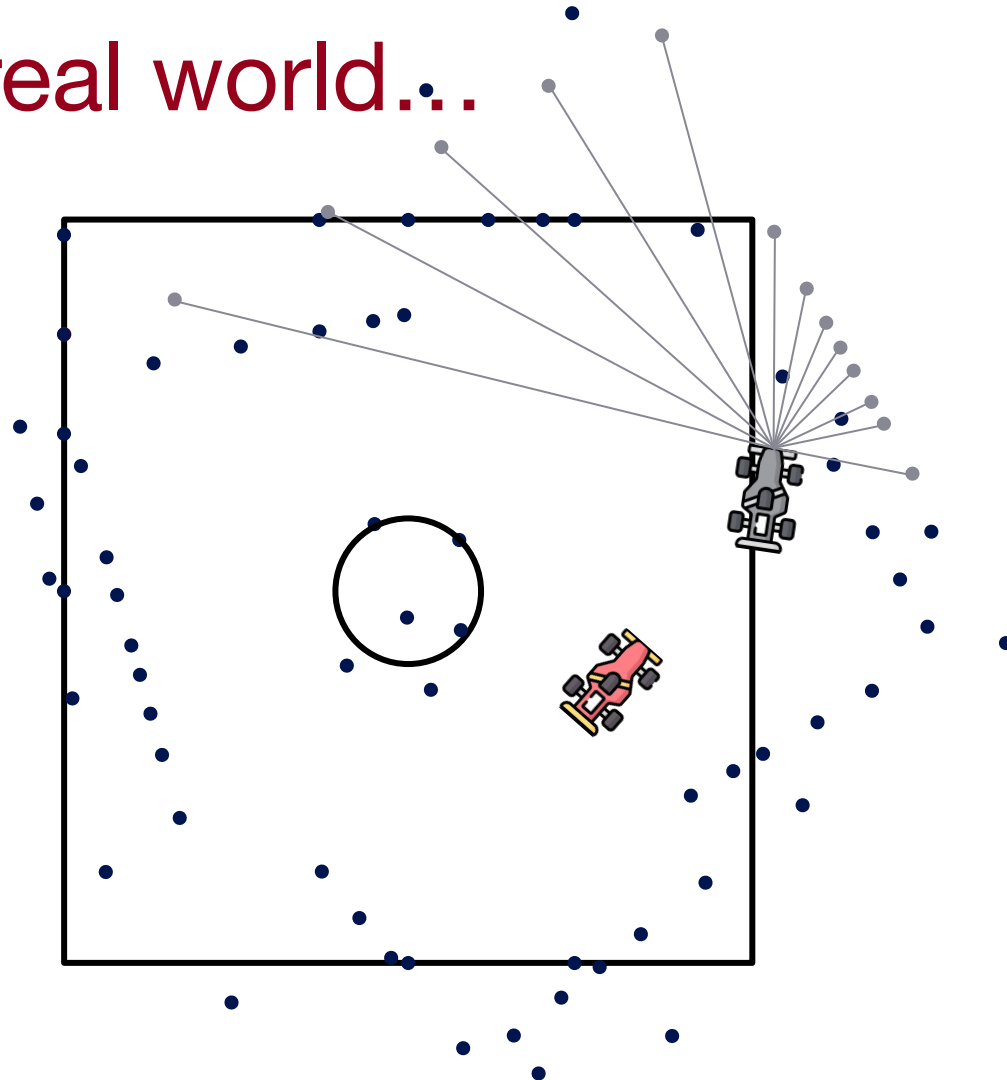
# But in the real world...



# But in the real world...

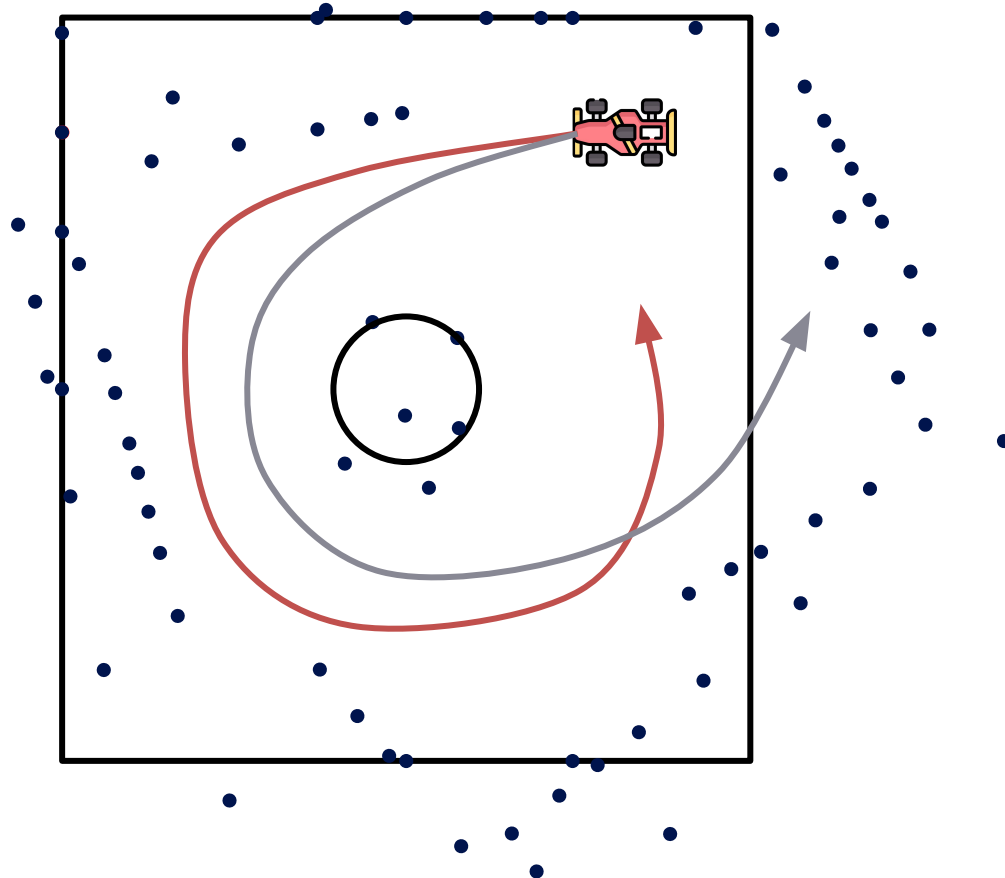


# But in the real world...

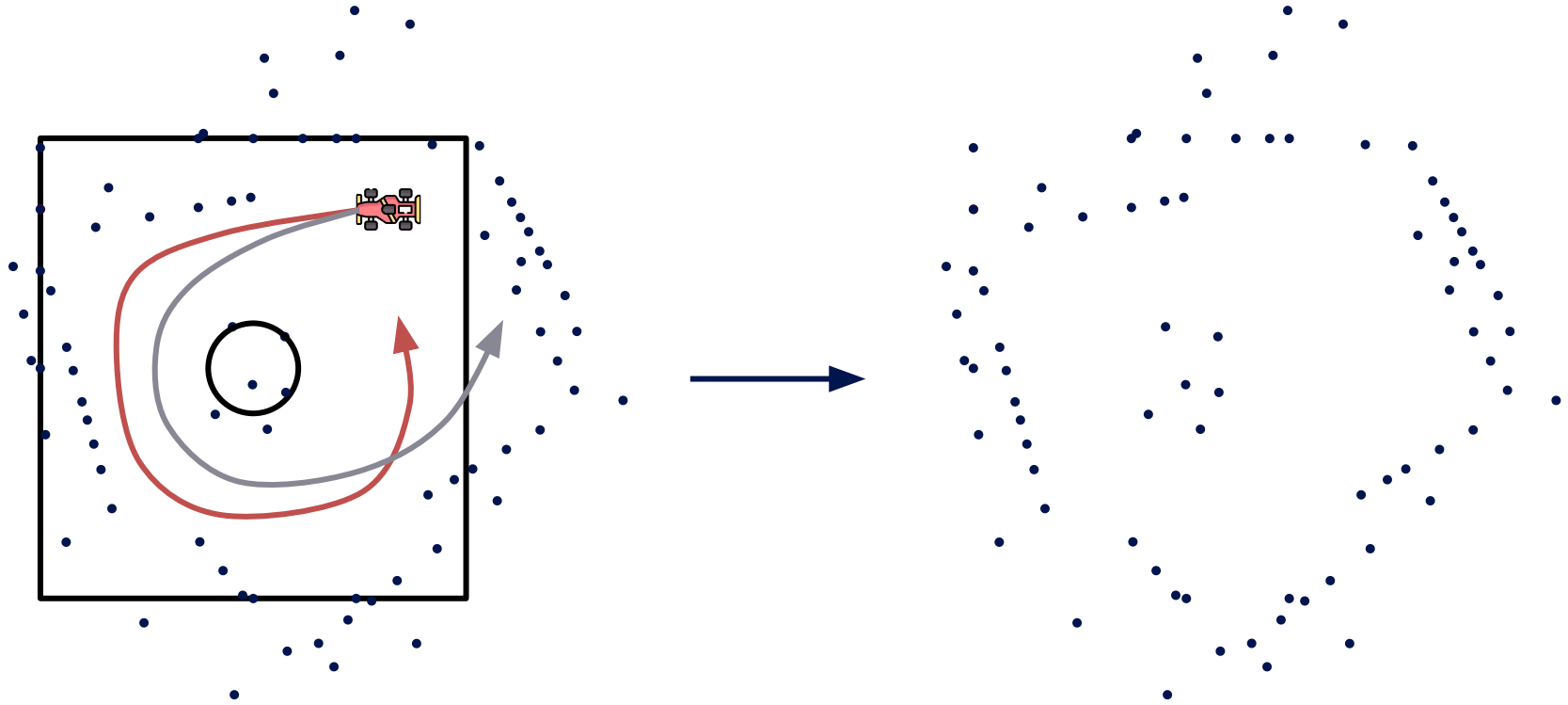




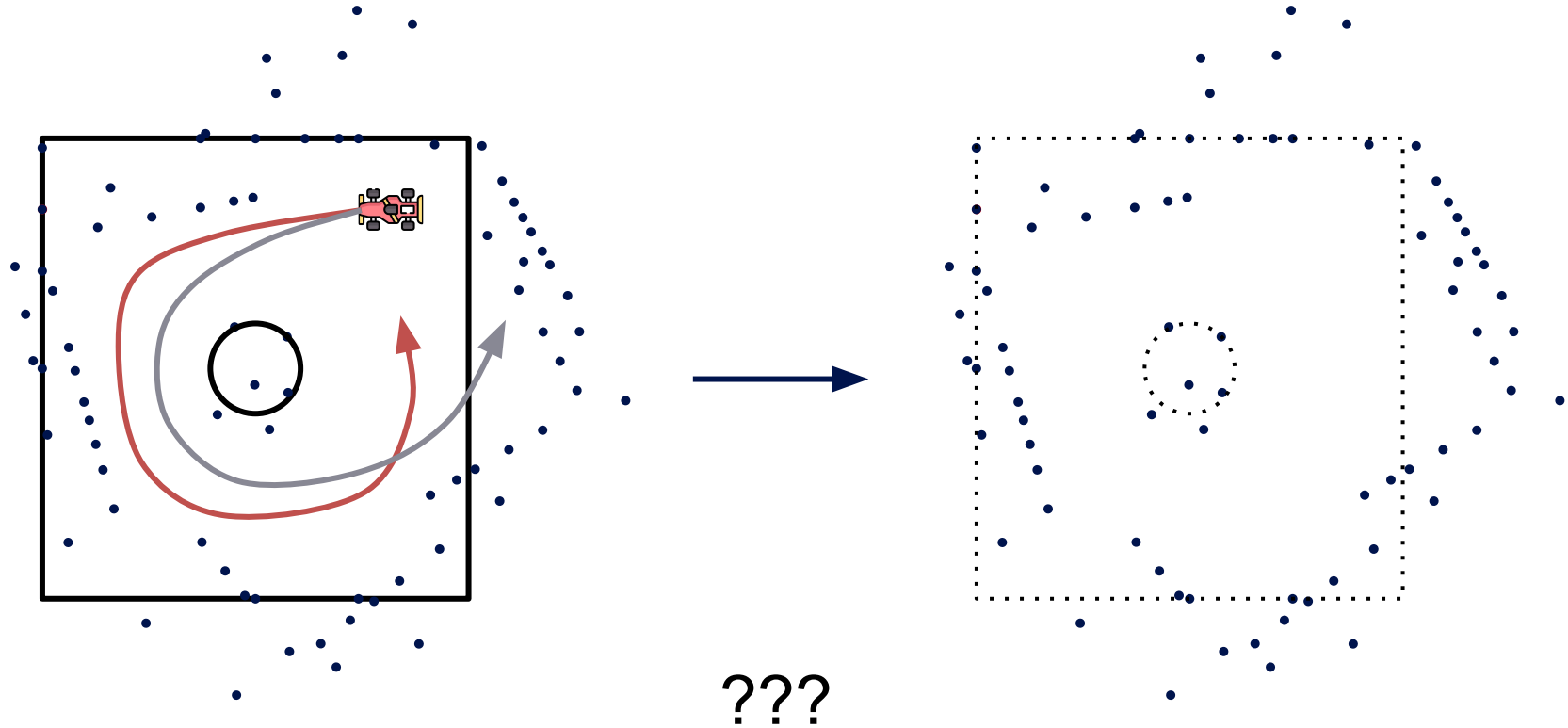
# But in the real world...



# But in the real world...



# But in the real world...

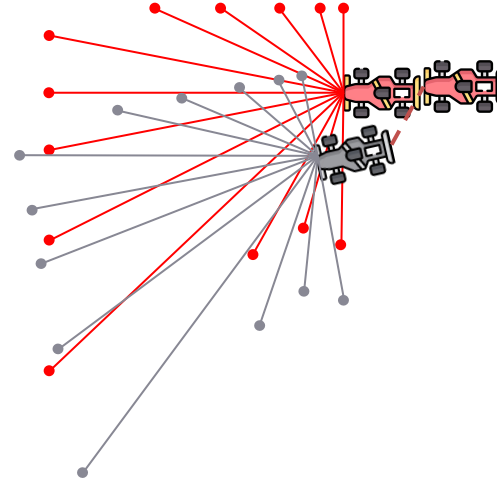
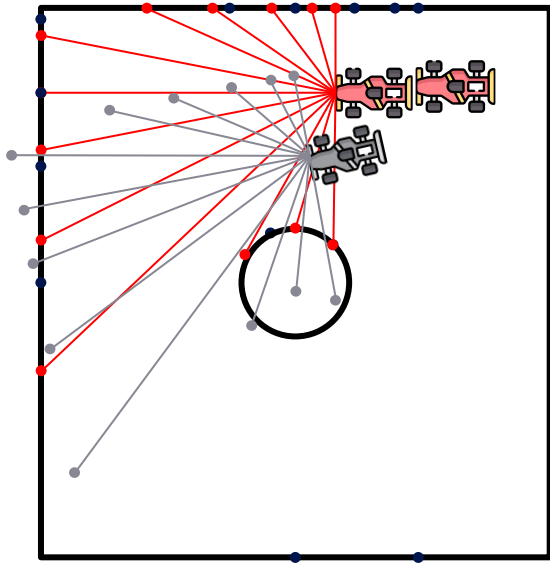


# SLAM overview

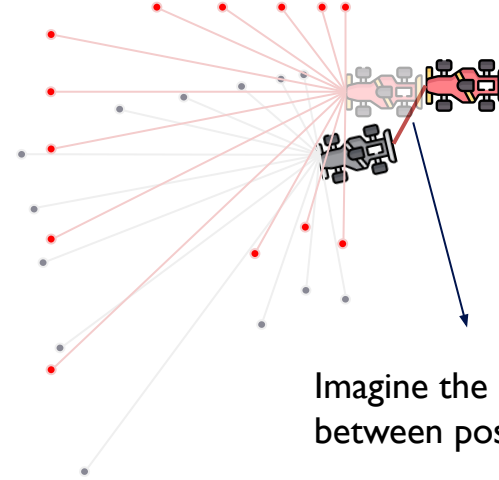
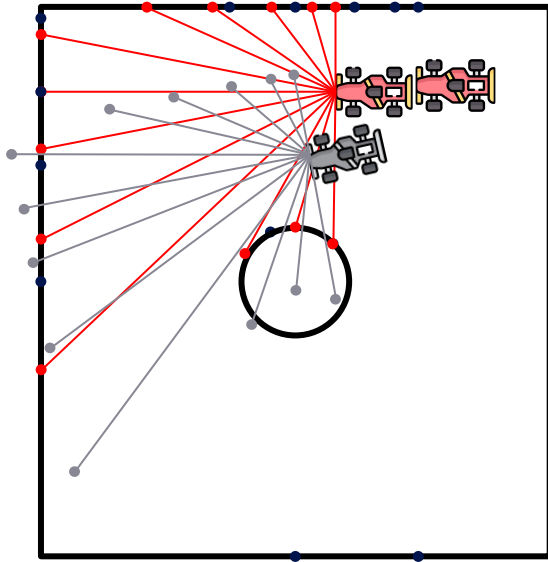
Generally 2 different approaches:

- Filtering-based
  - Use EKF or Particle Filter for state estimation and build the map based on that estimation
  - Examples are GMapping, HectorSLAM
- Graph-based
  - Use pose graphs to store constraints between pose estimations
  - Use optimization for loop-closure to correct for pose estimations
  - Examples are KartoSLAM and Cartographer
  - We'll be using slam\_toolbox, also graph-based

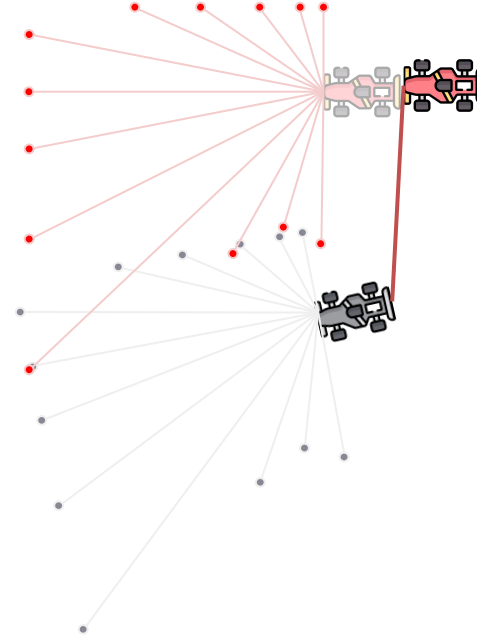
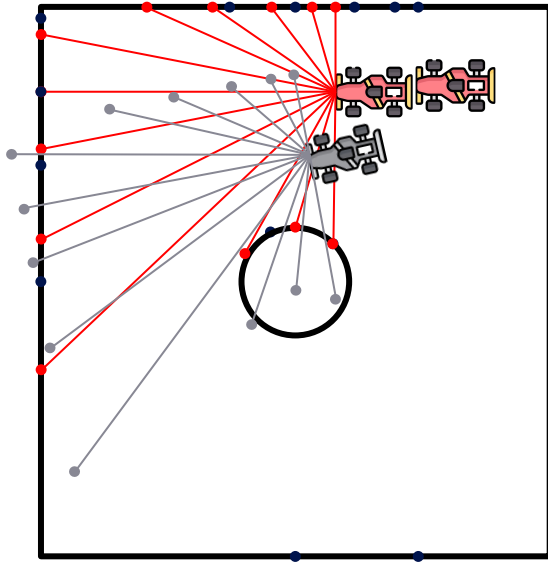
# Pose Graphs: Edge Constraints



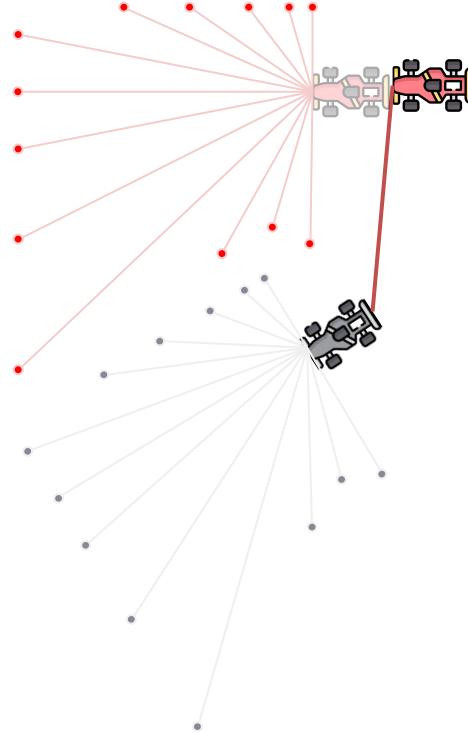
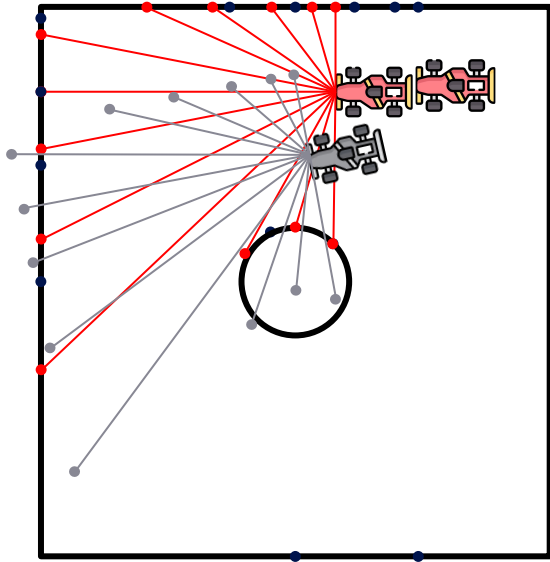
# Pose Graphs: Edge Constraints



# Pose Graphs: Edge Constraints

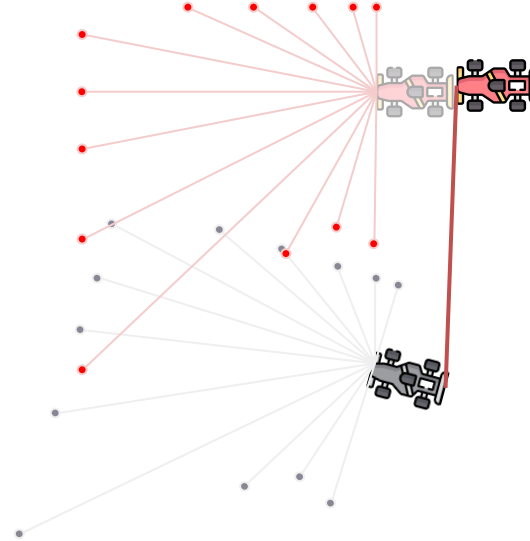
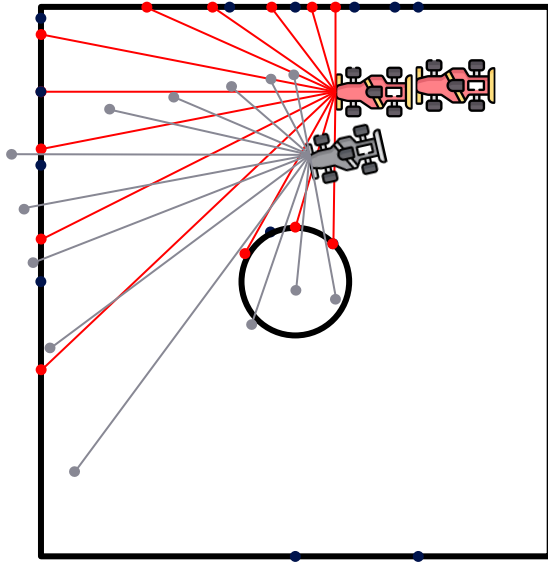


# Pose Graphs: Edge Constraints

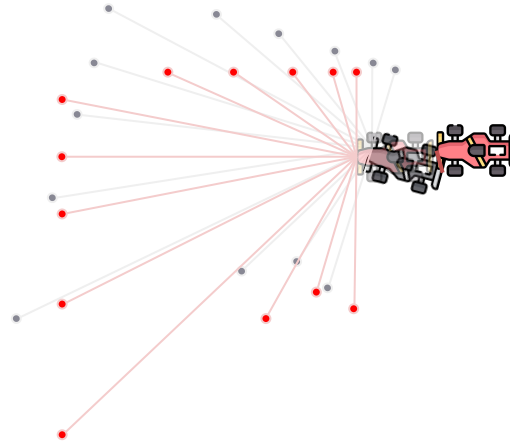
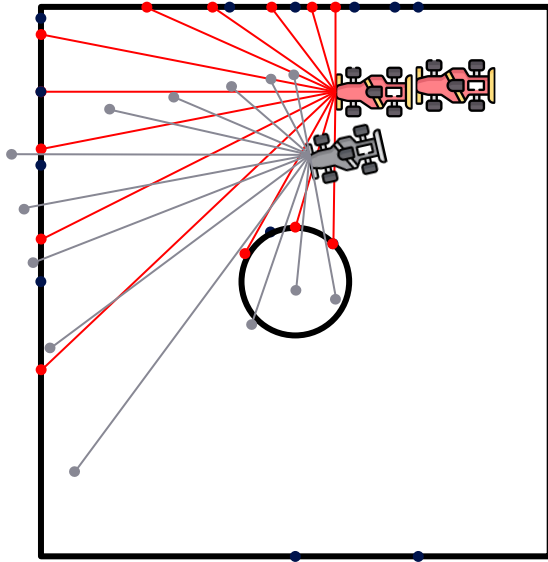




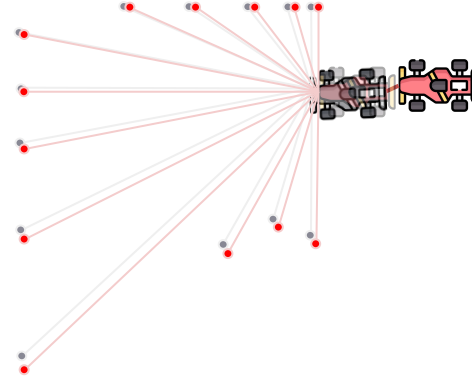
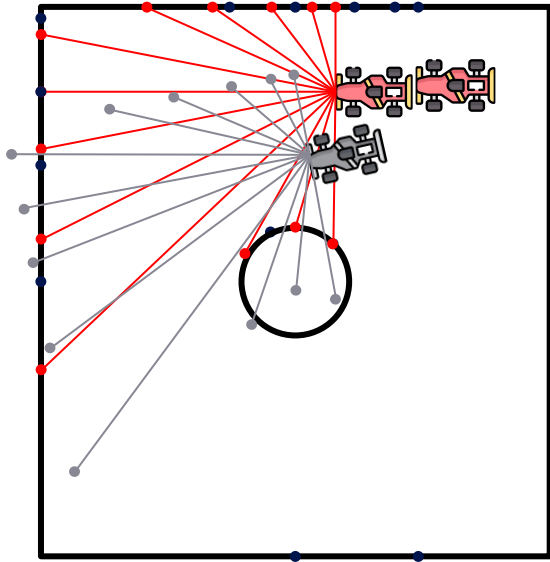
# Pose Graphs: Edge Constraints



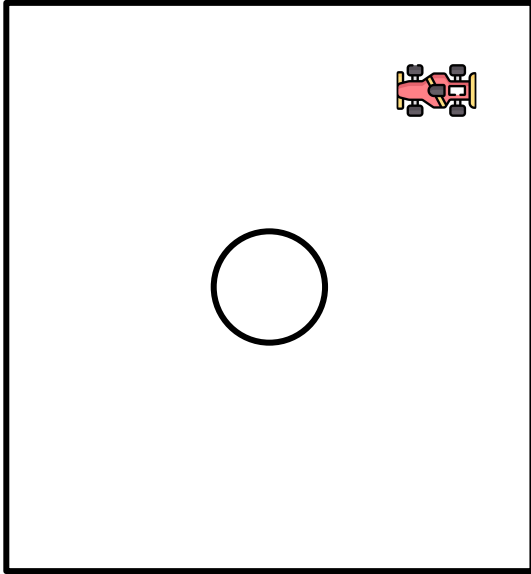
# Pose Graphs: Edge Constraints



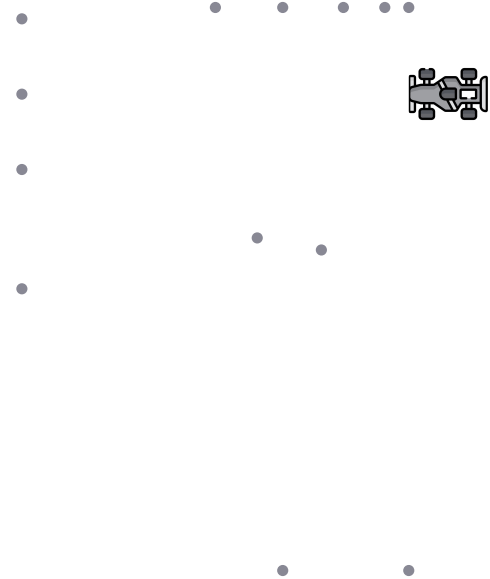
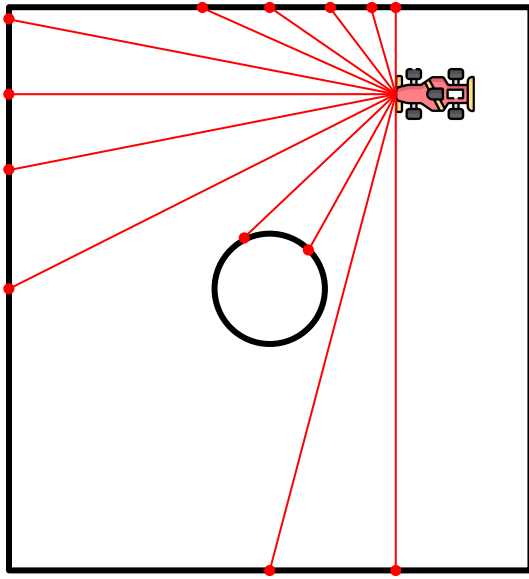
# Pose Graphs: Edge Constraints



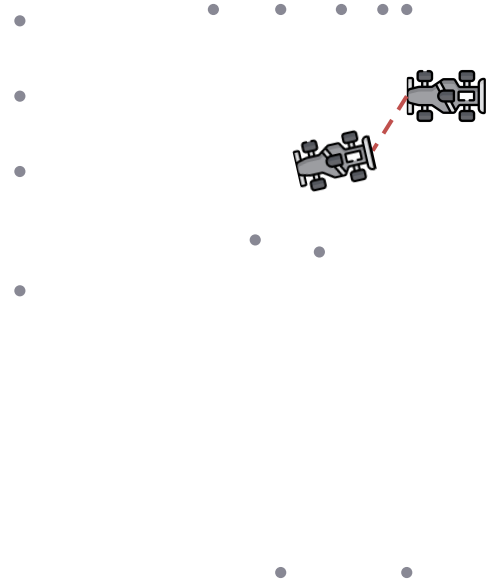
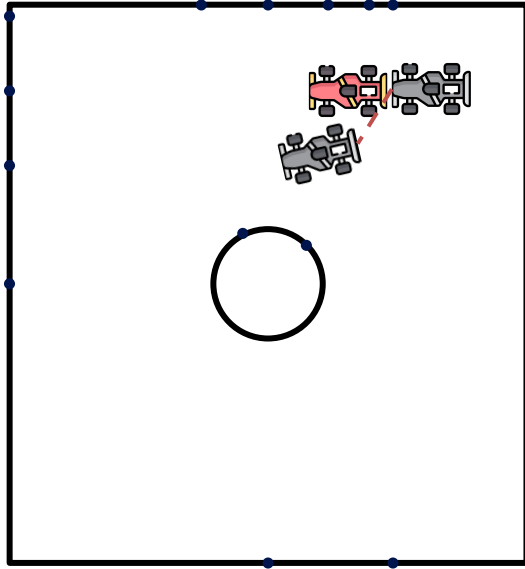
# Pose Graphs



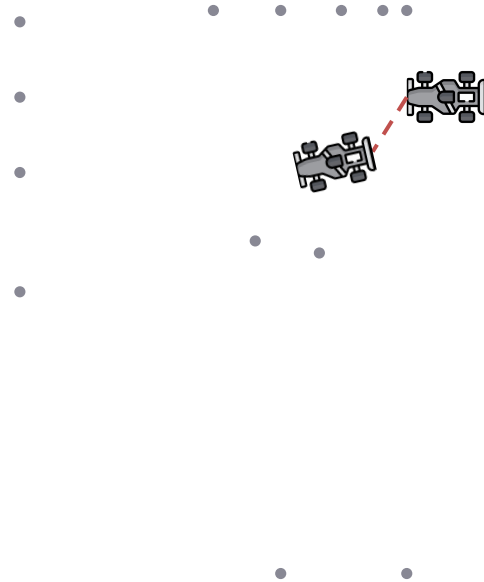
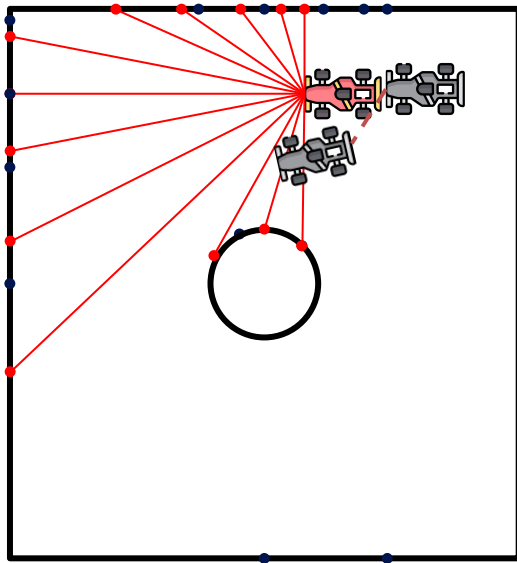
# Pose Graphs



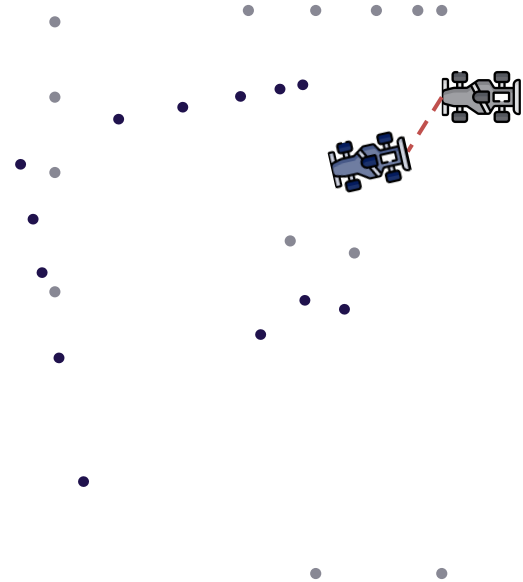
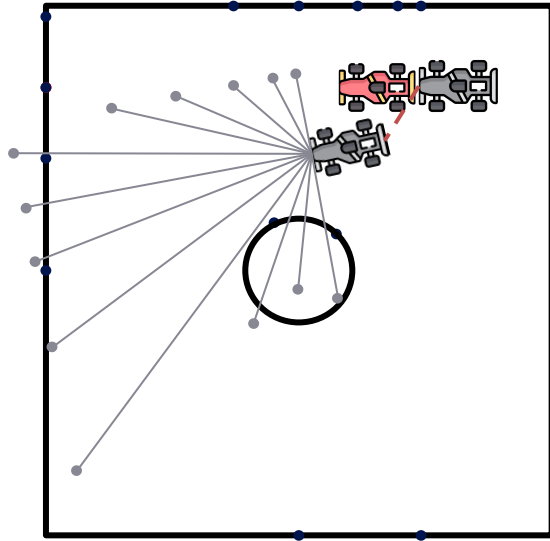
# Pose Graphs



# Pose Graphs

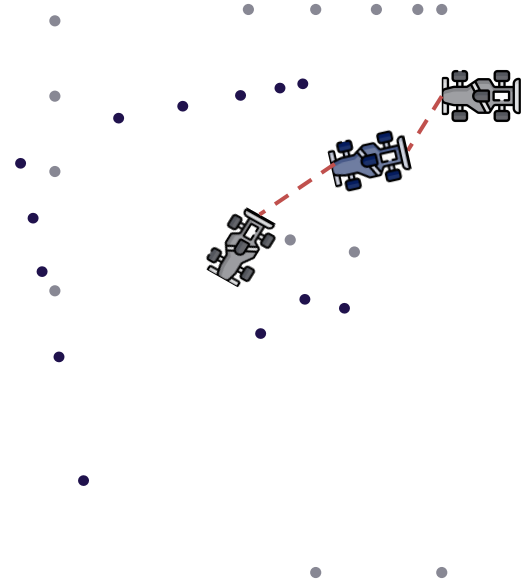
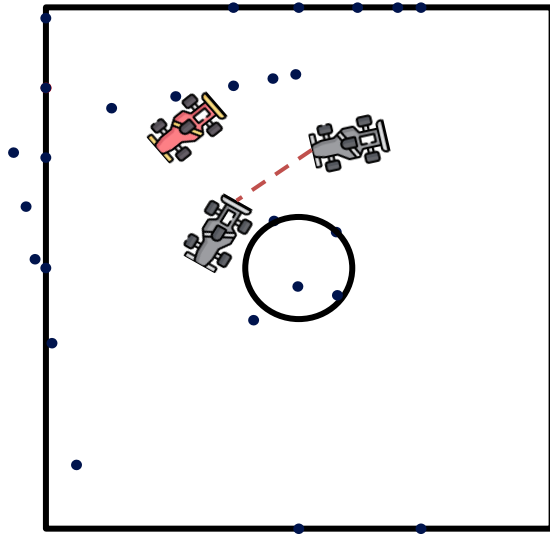


# Pose Graphs

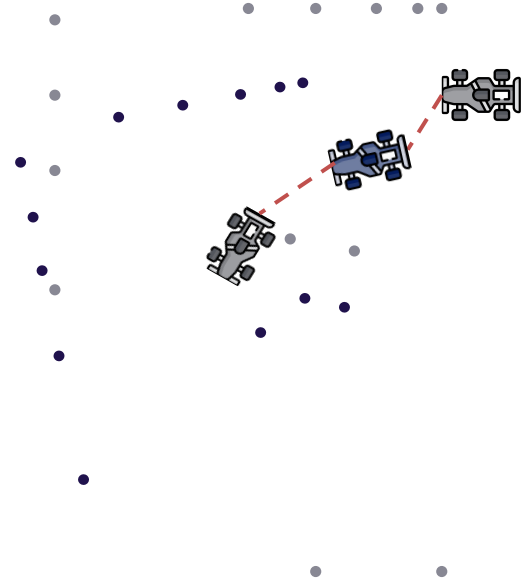
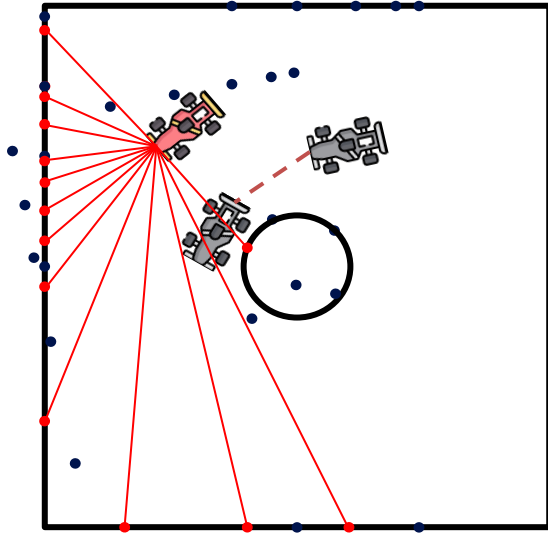




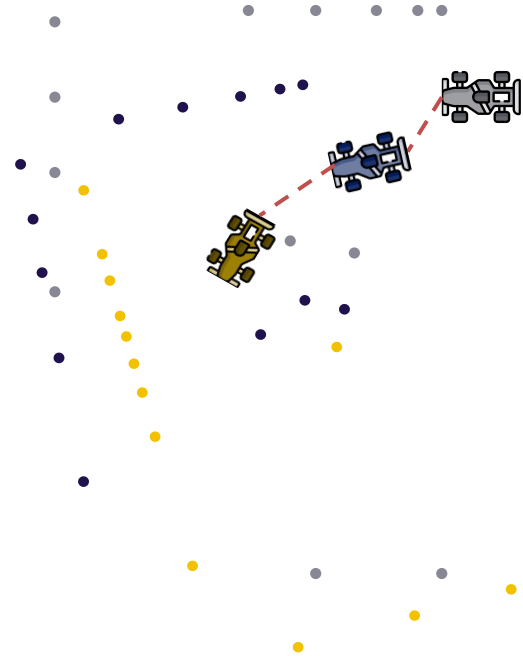
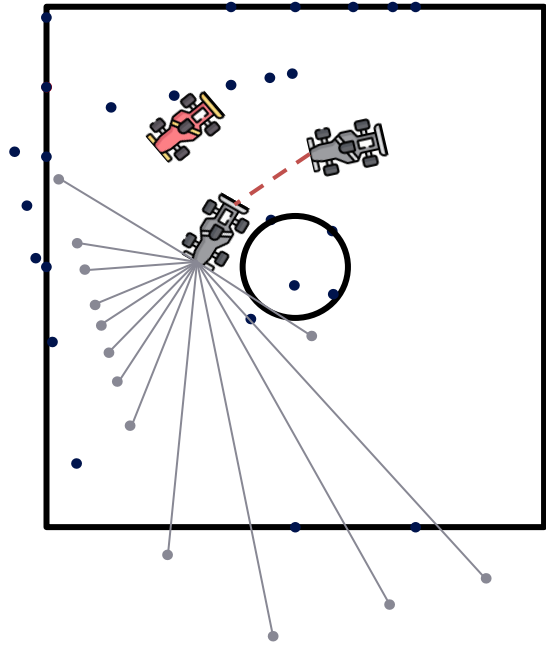
# Pose Graphs



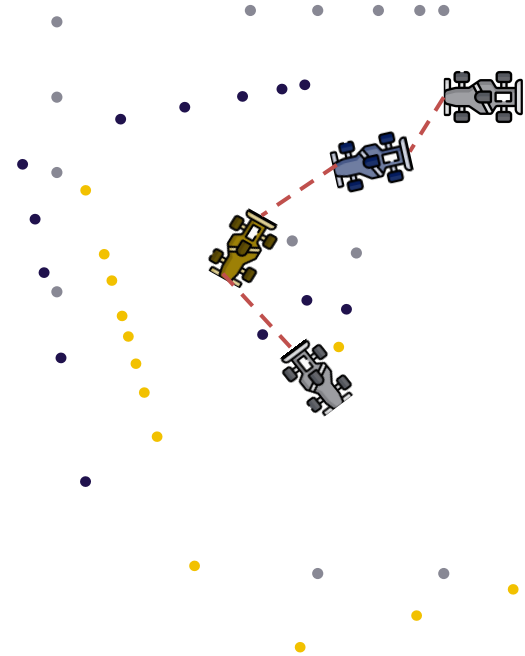
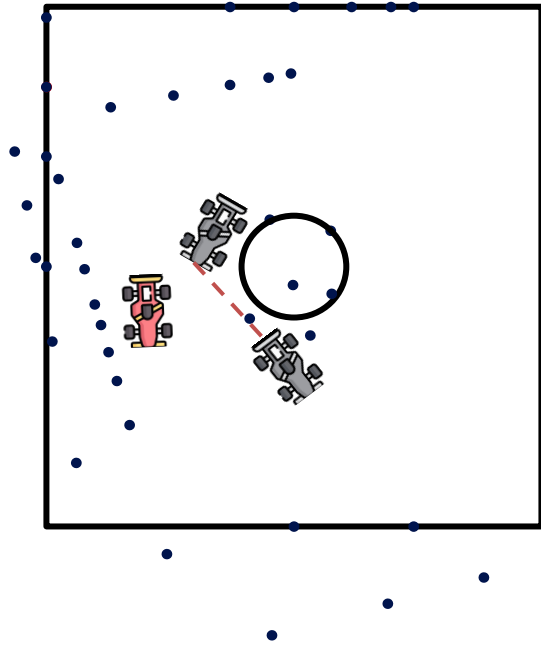
# Pose Graphs



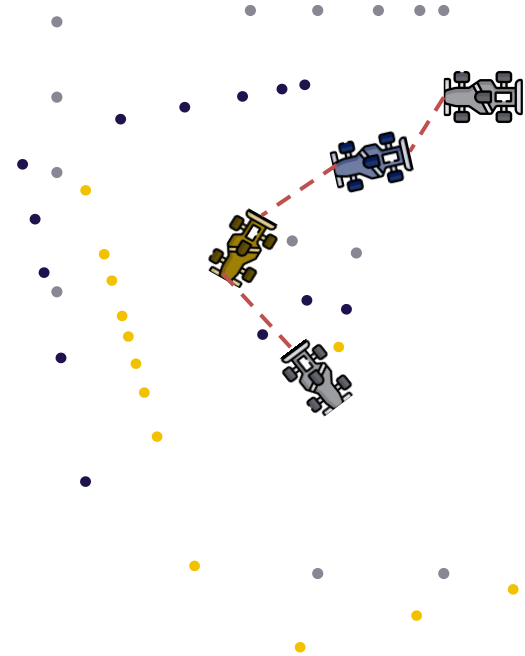
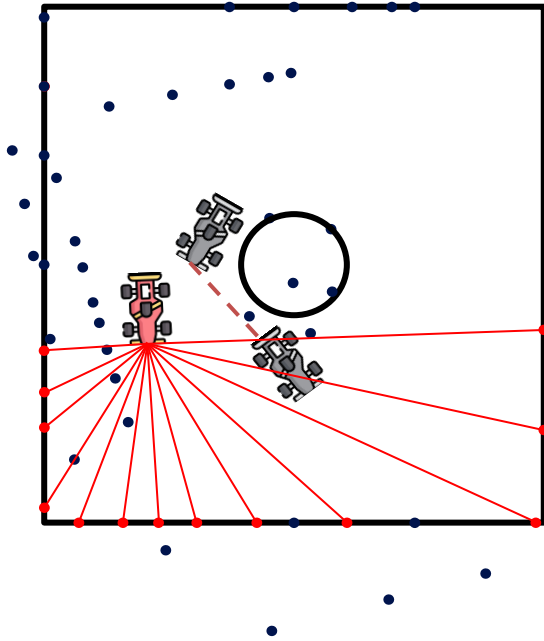
# Pose Graphs



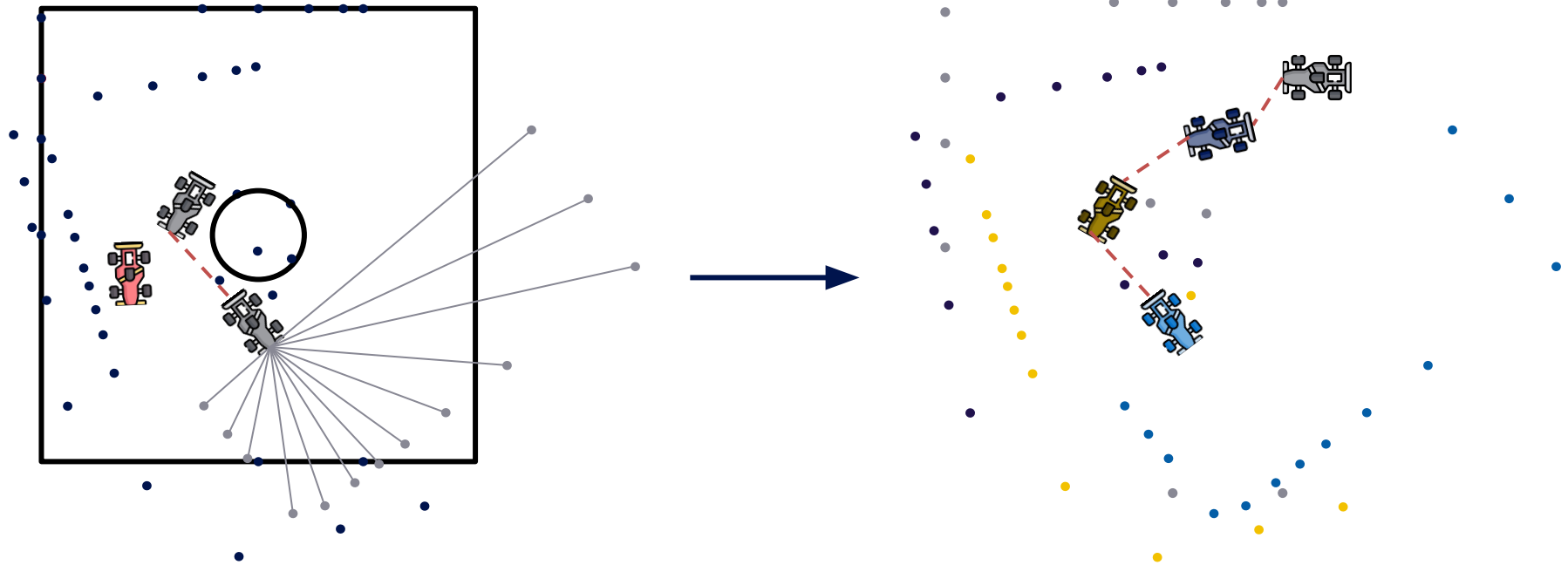
# Pose Graphs



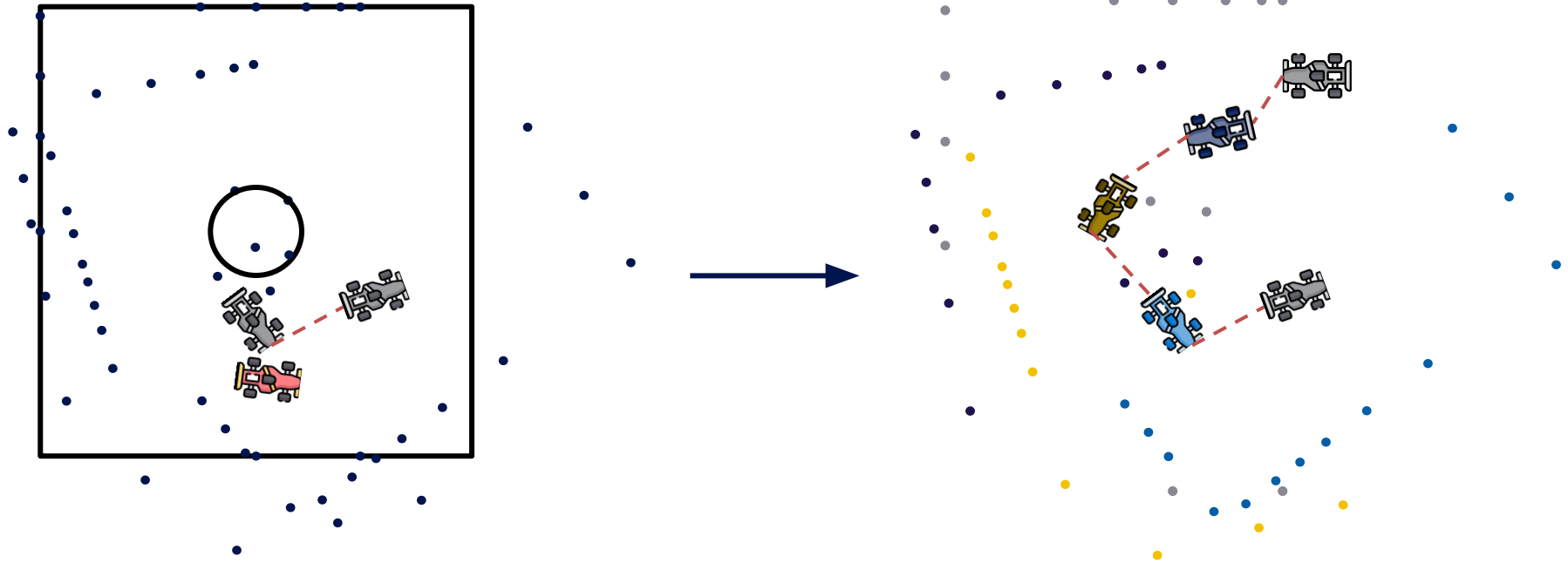
# Pose Graphs



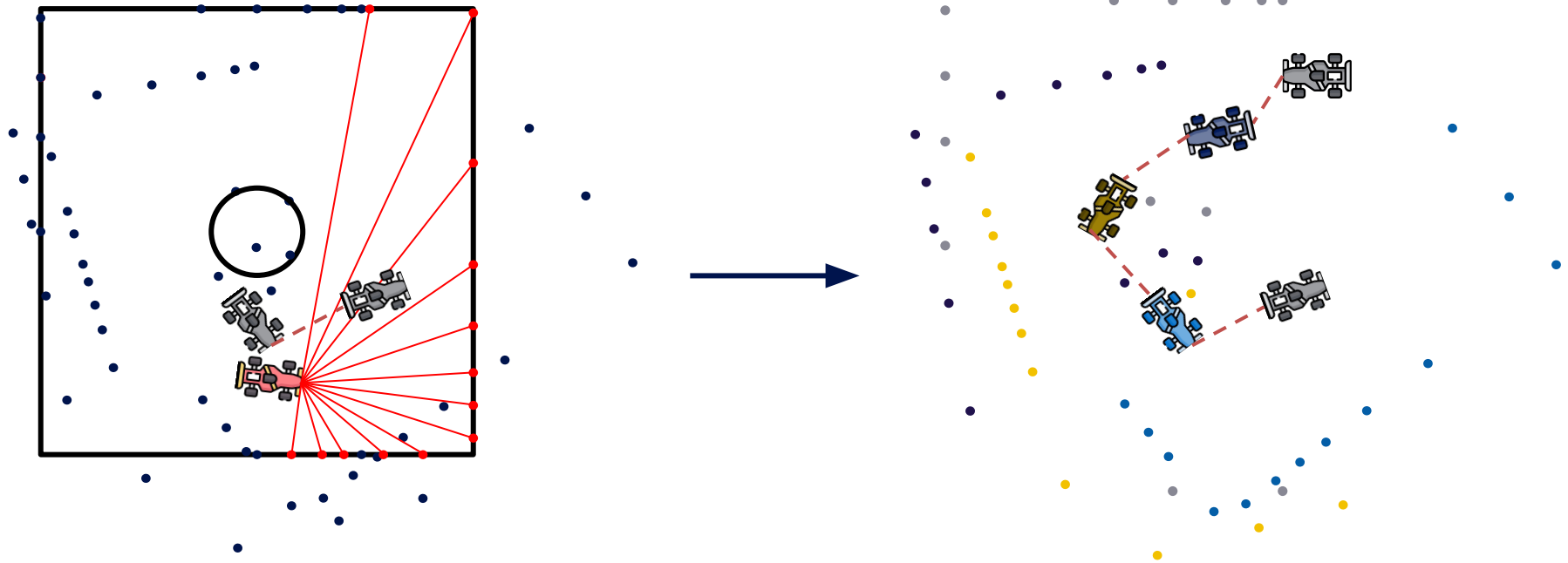
# Pose Graphs



# Pose Graphs

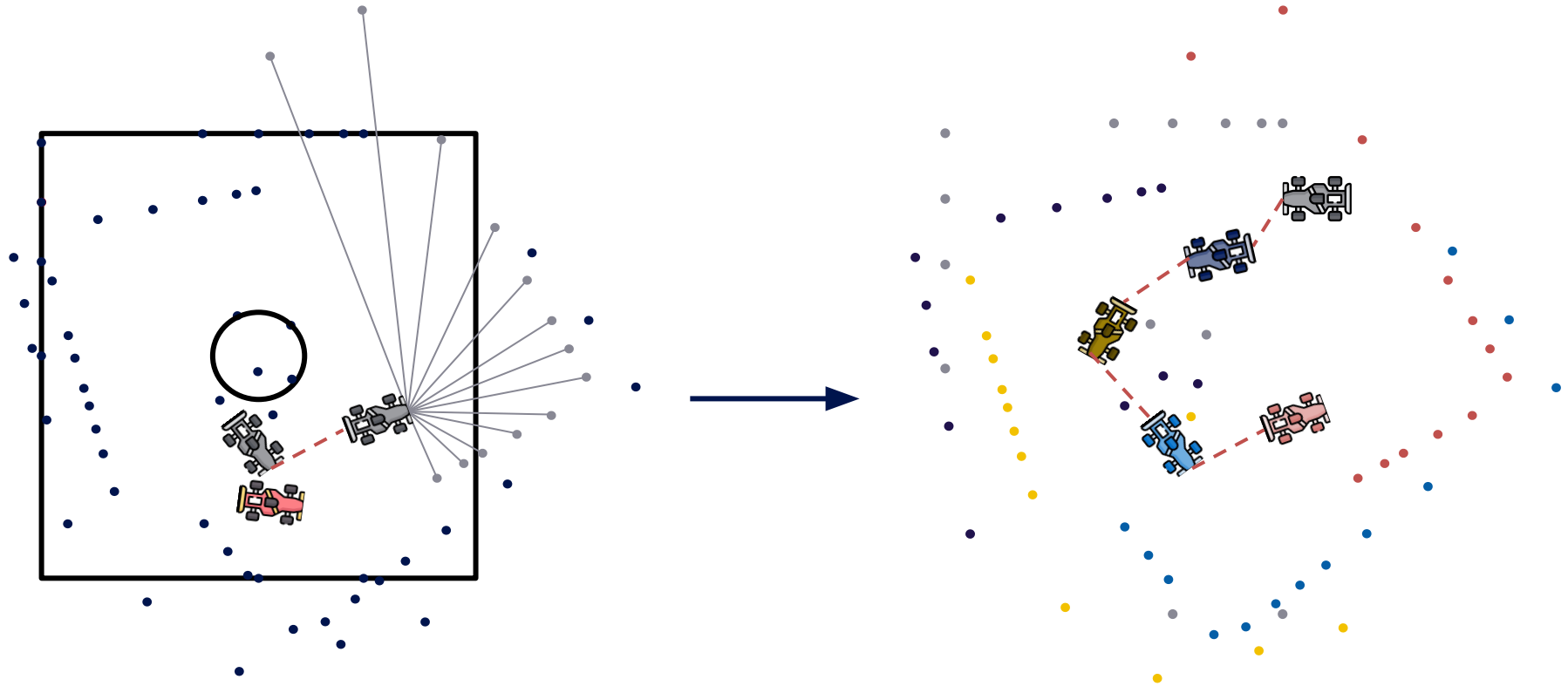


# Pose Graphs

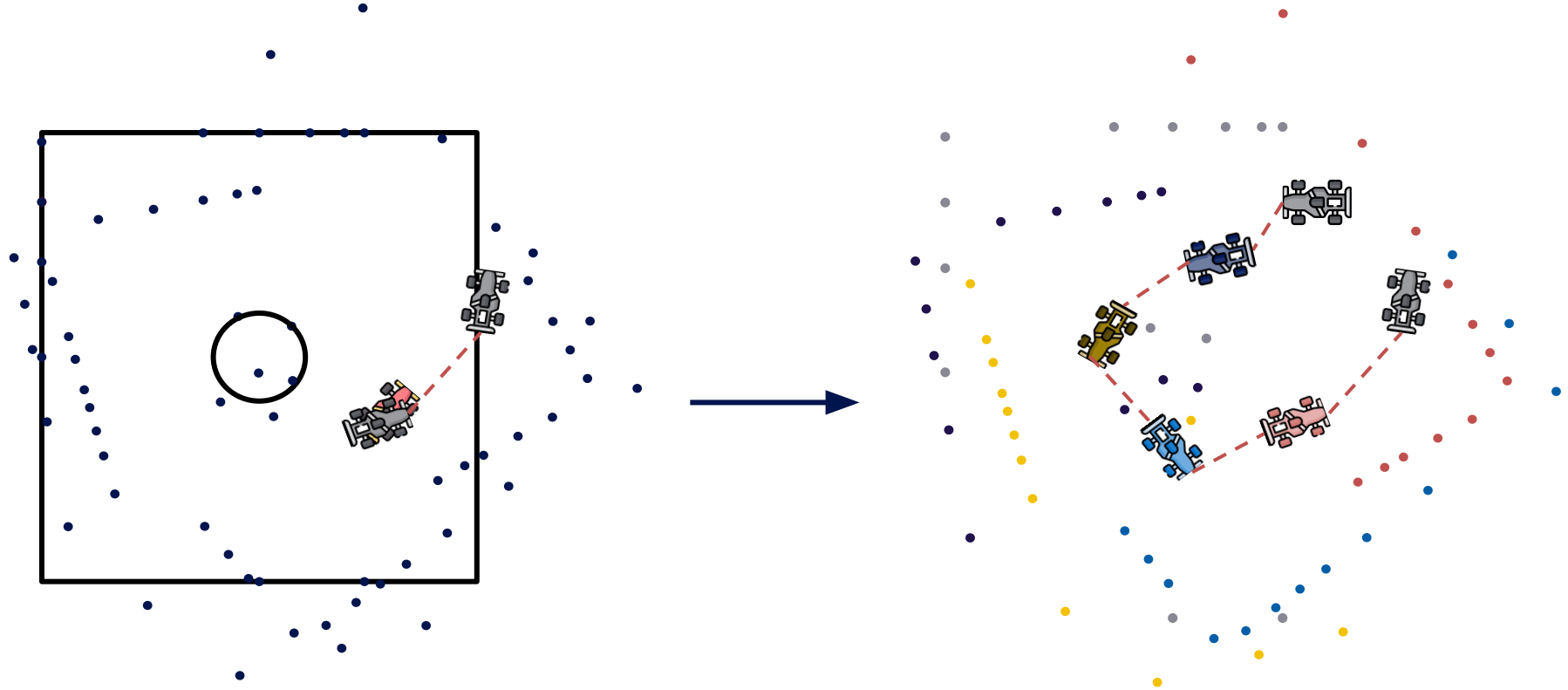




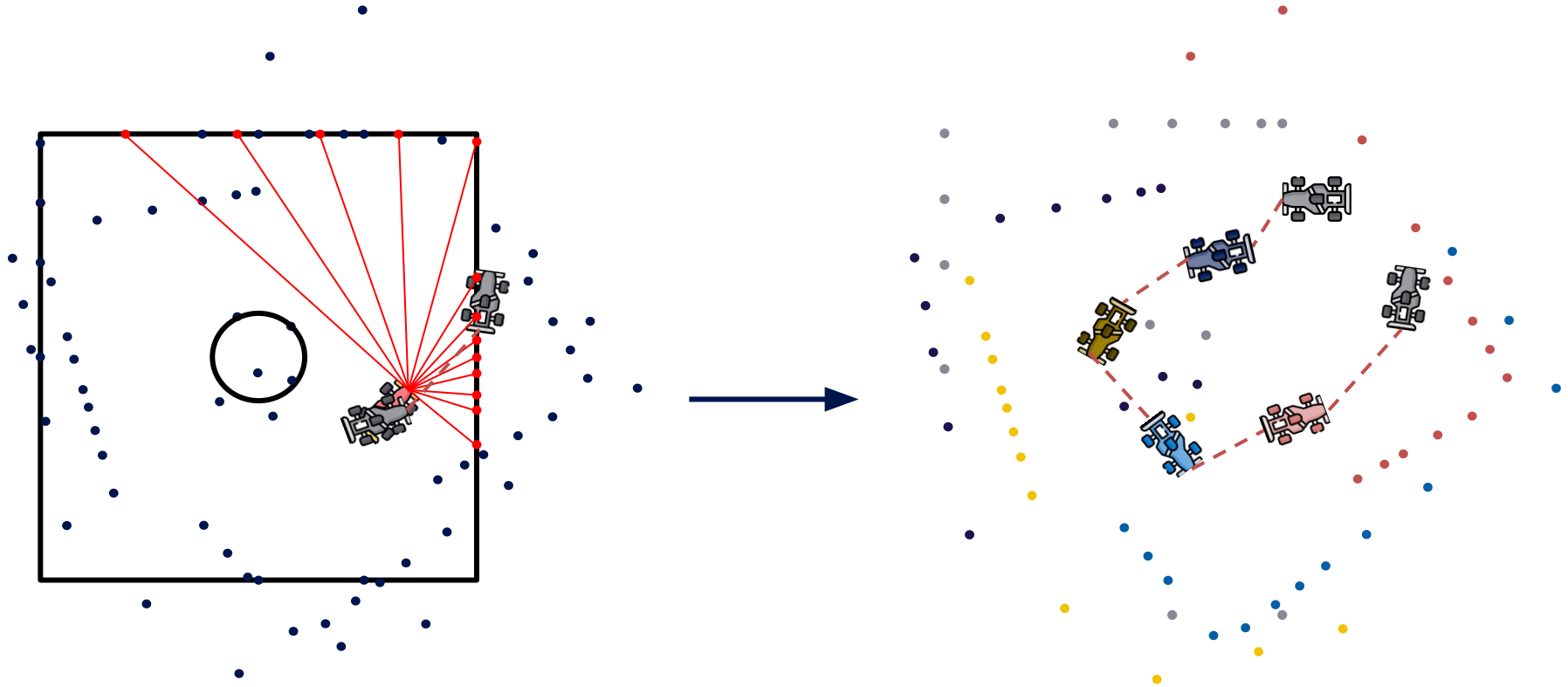
# Pose Graphs



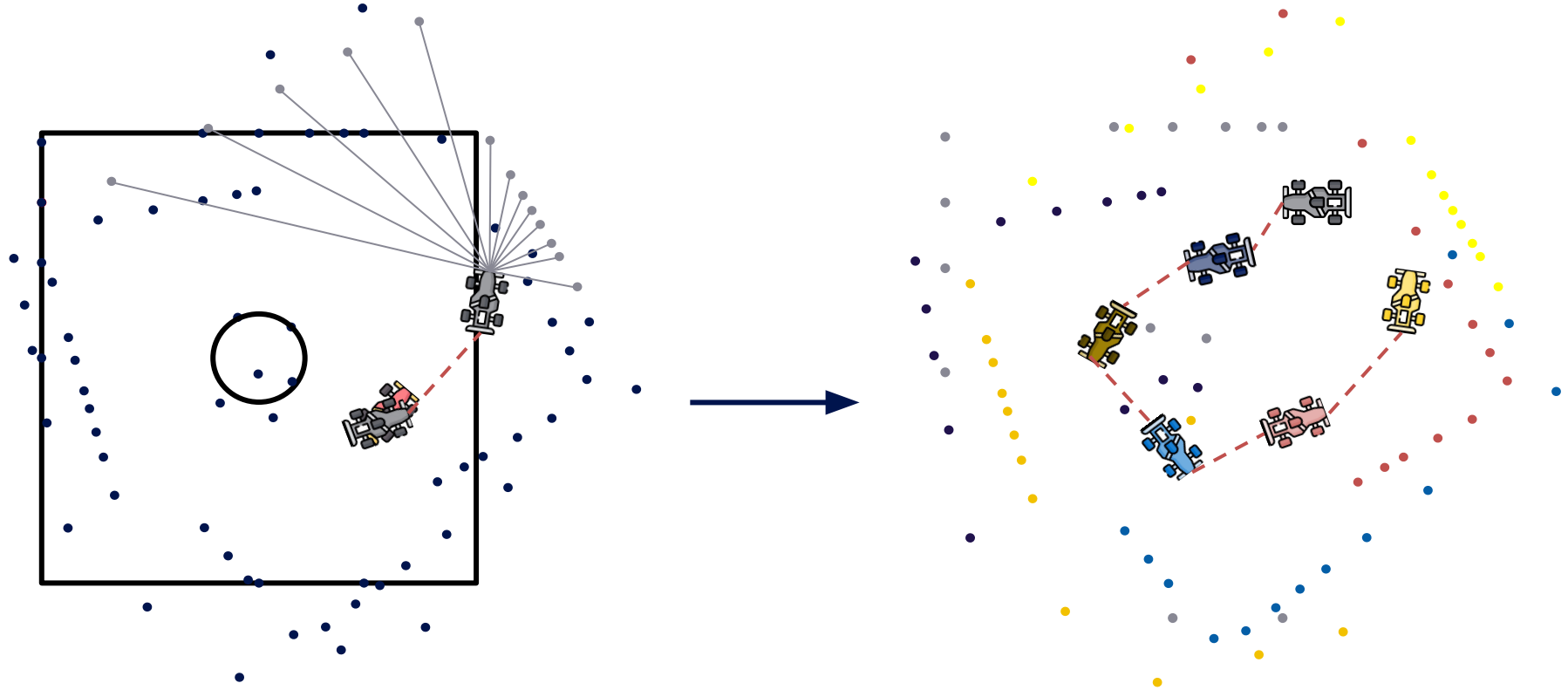
# Pose Graphs



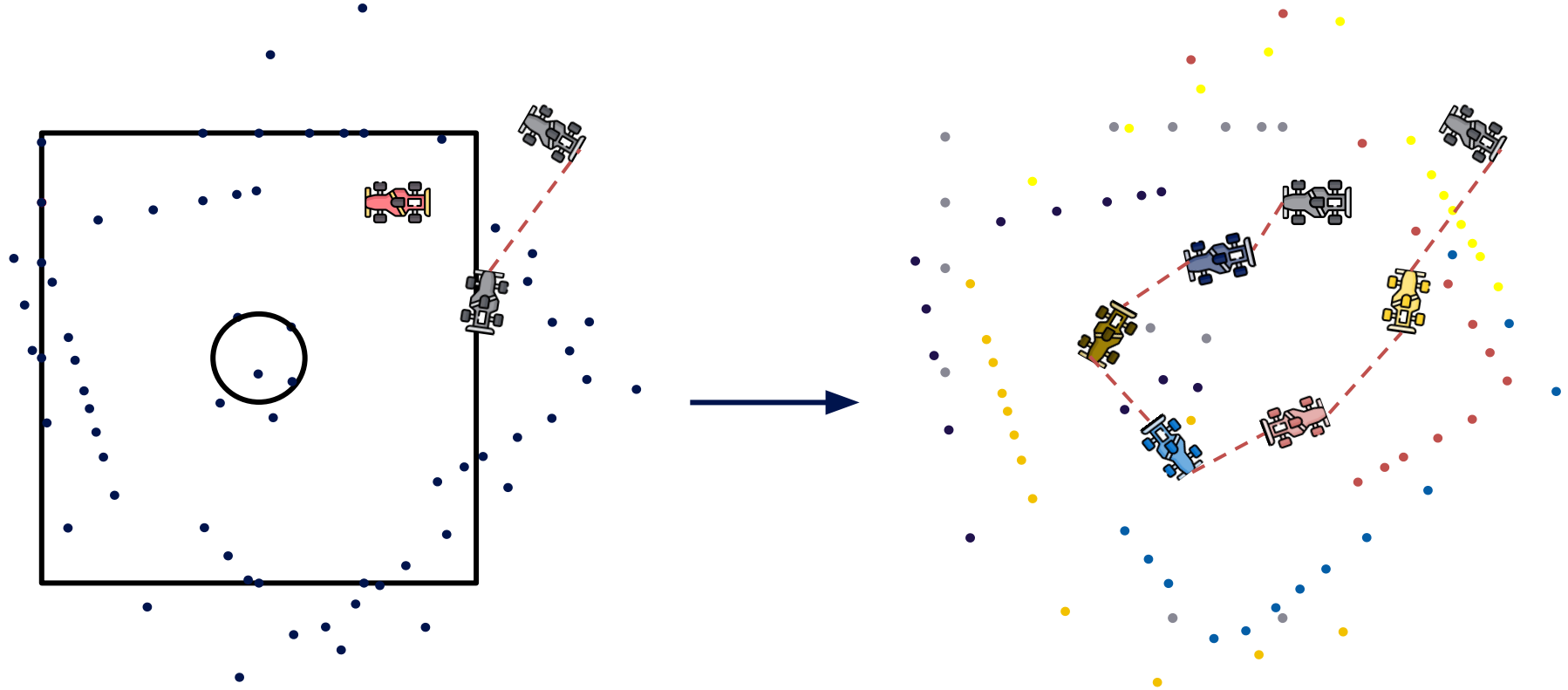
# Pose Graphs



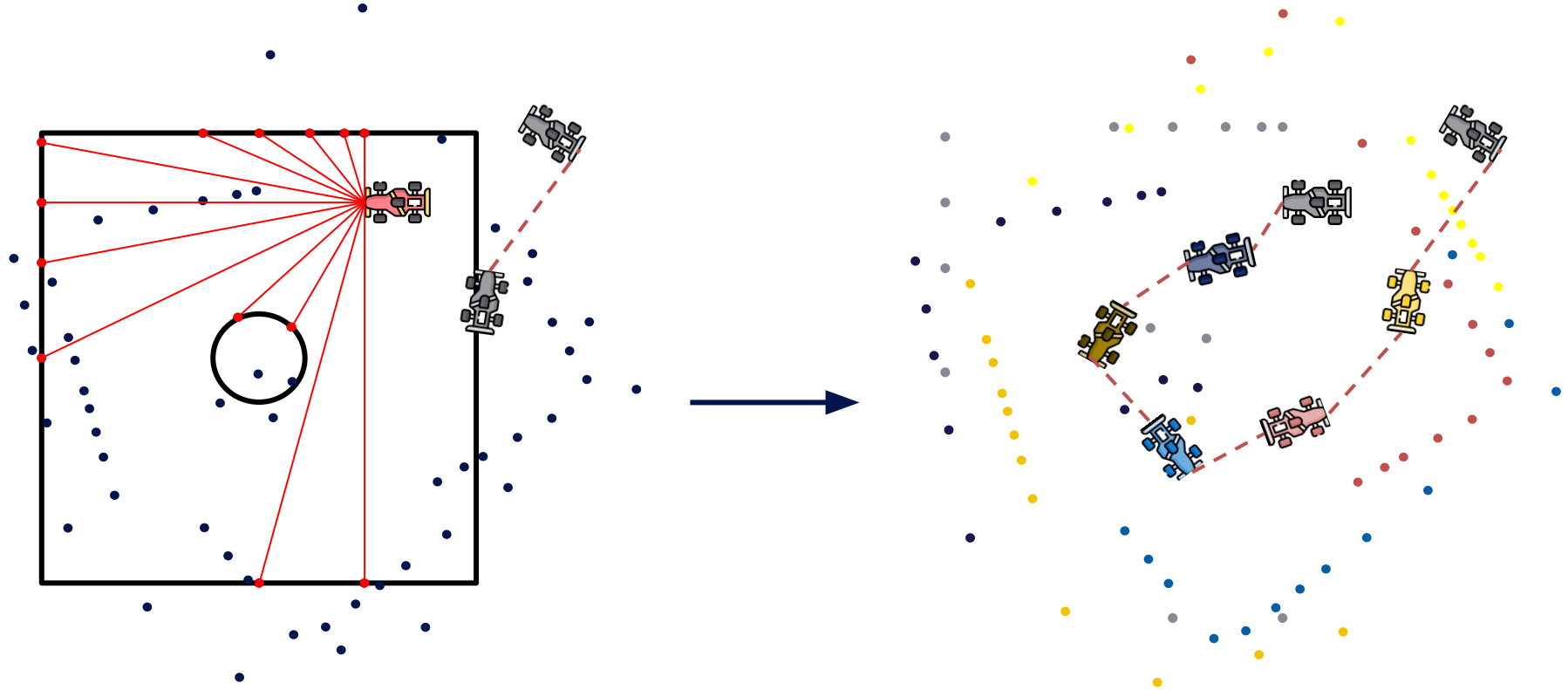
# Pose Graphs



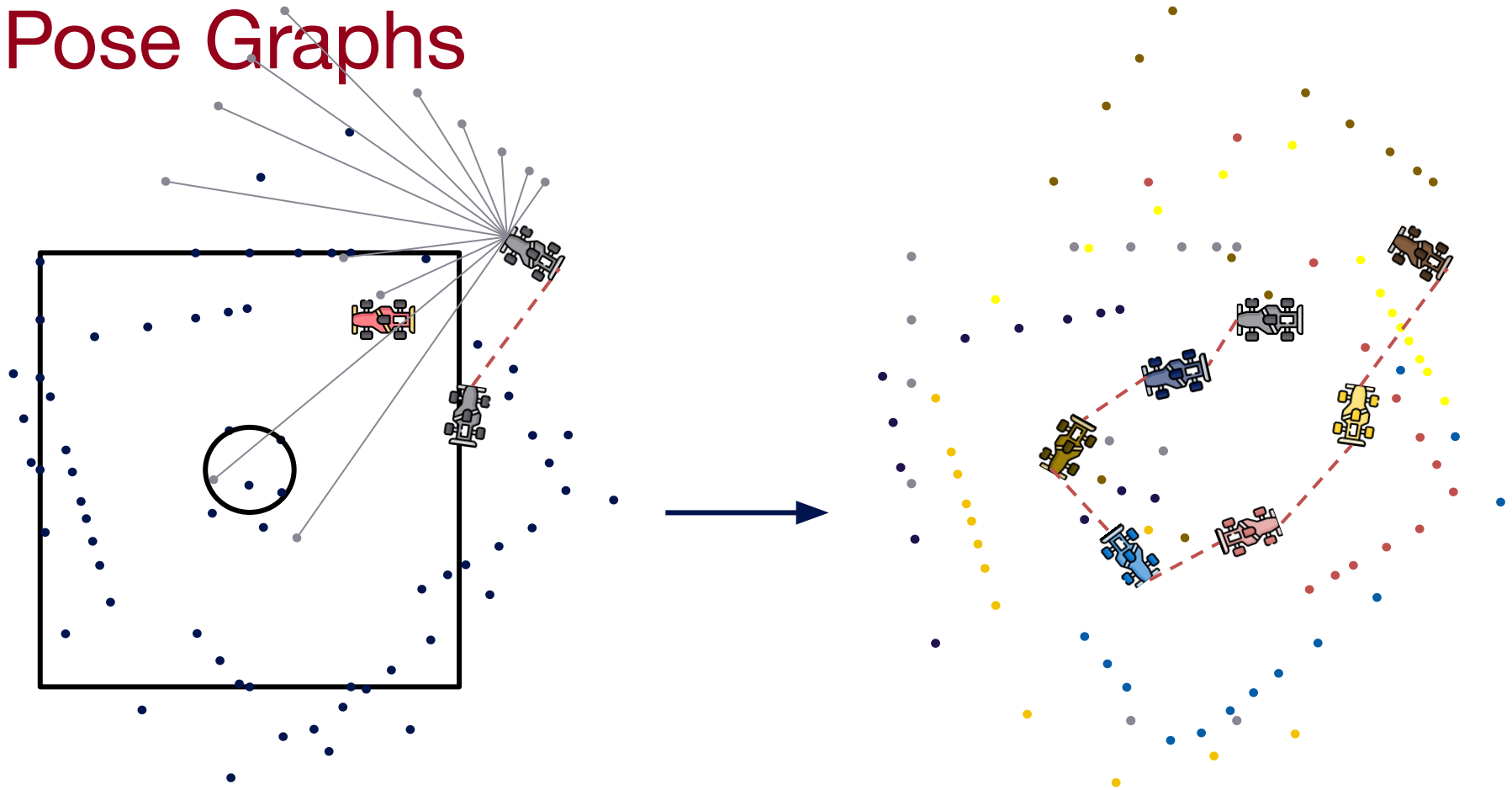
# Pose Graphs



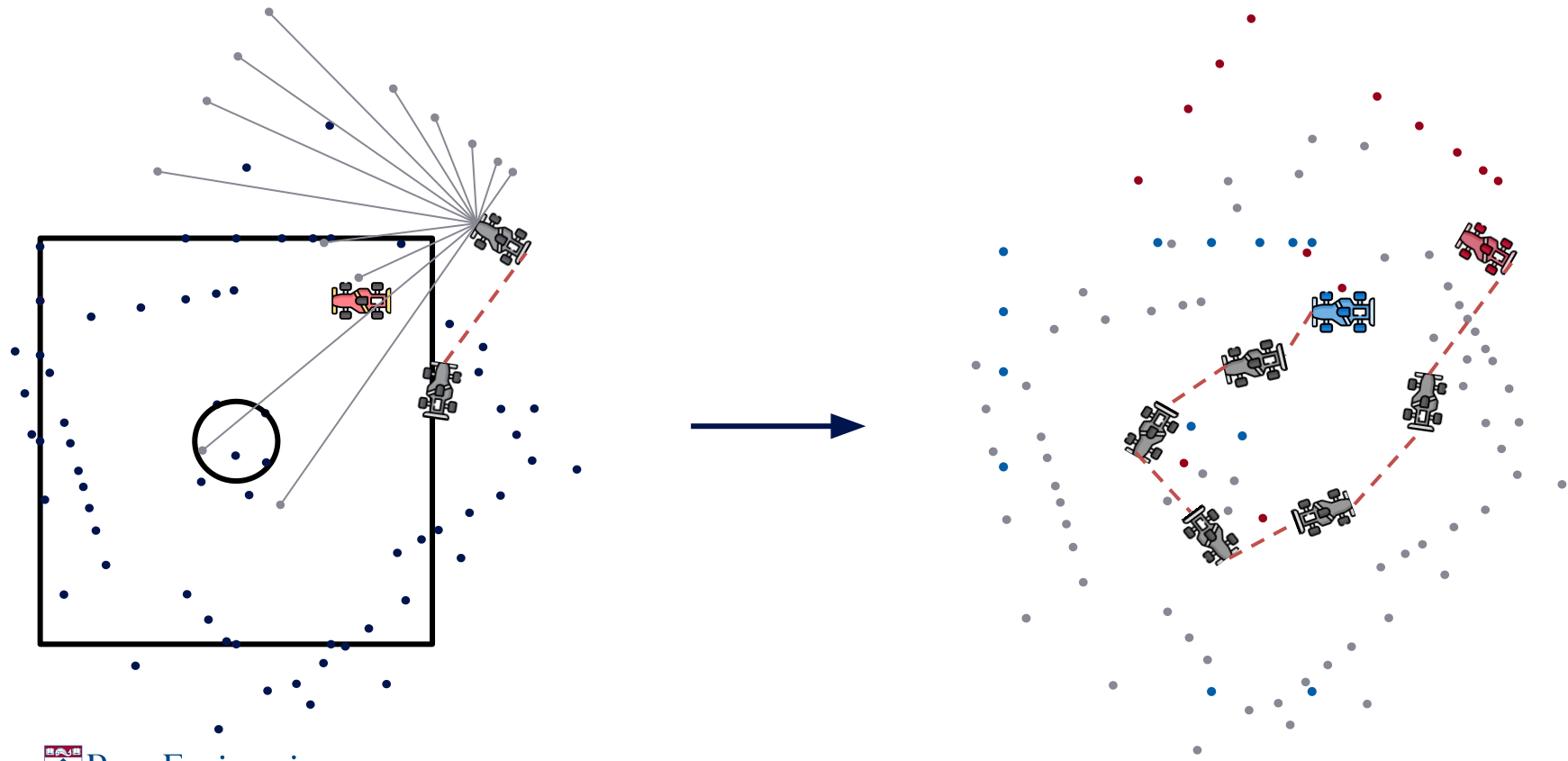
# Pose Graphs



# Pose Graphs

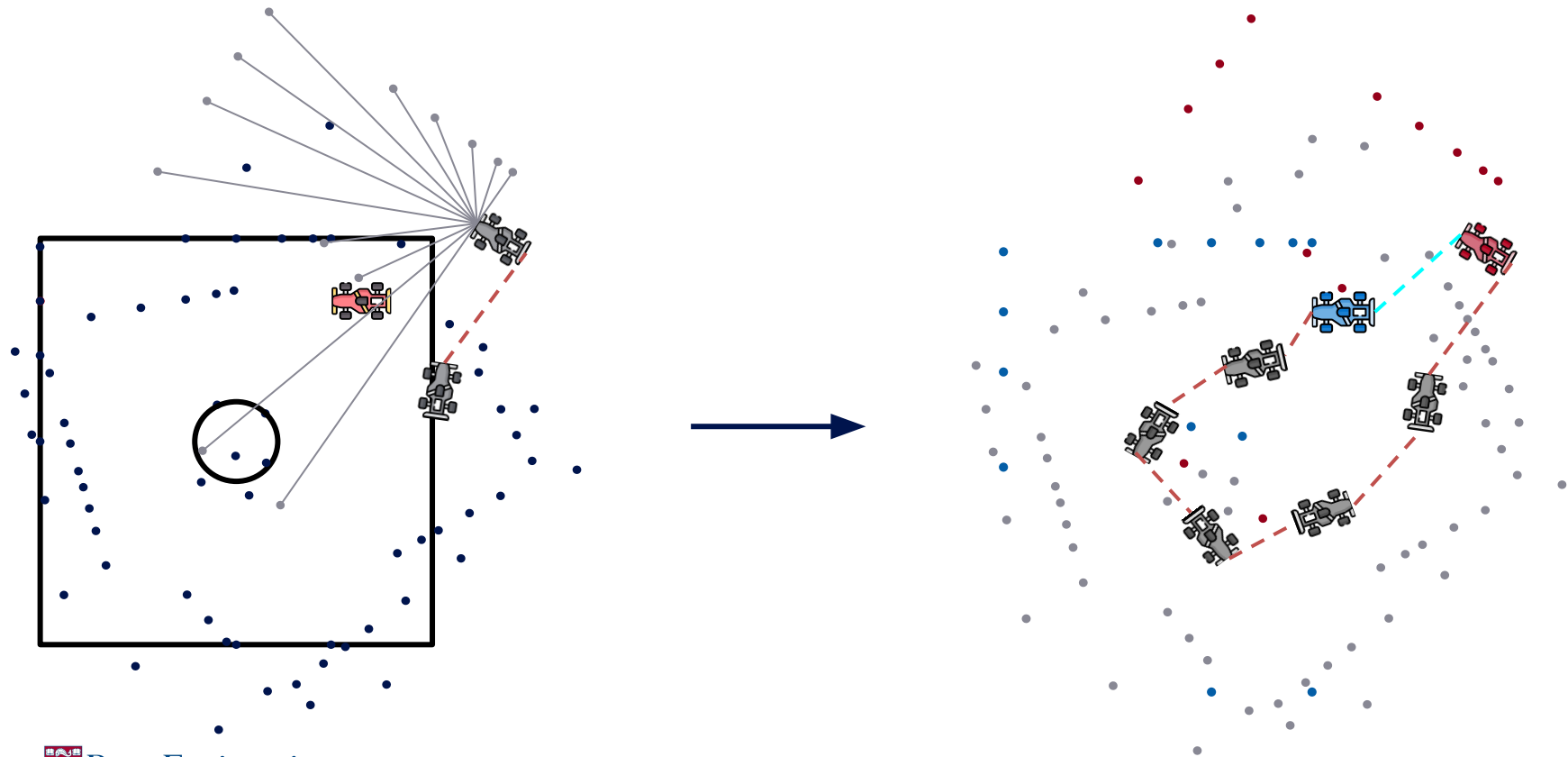


# Pose Graph Optimization: Loop closure

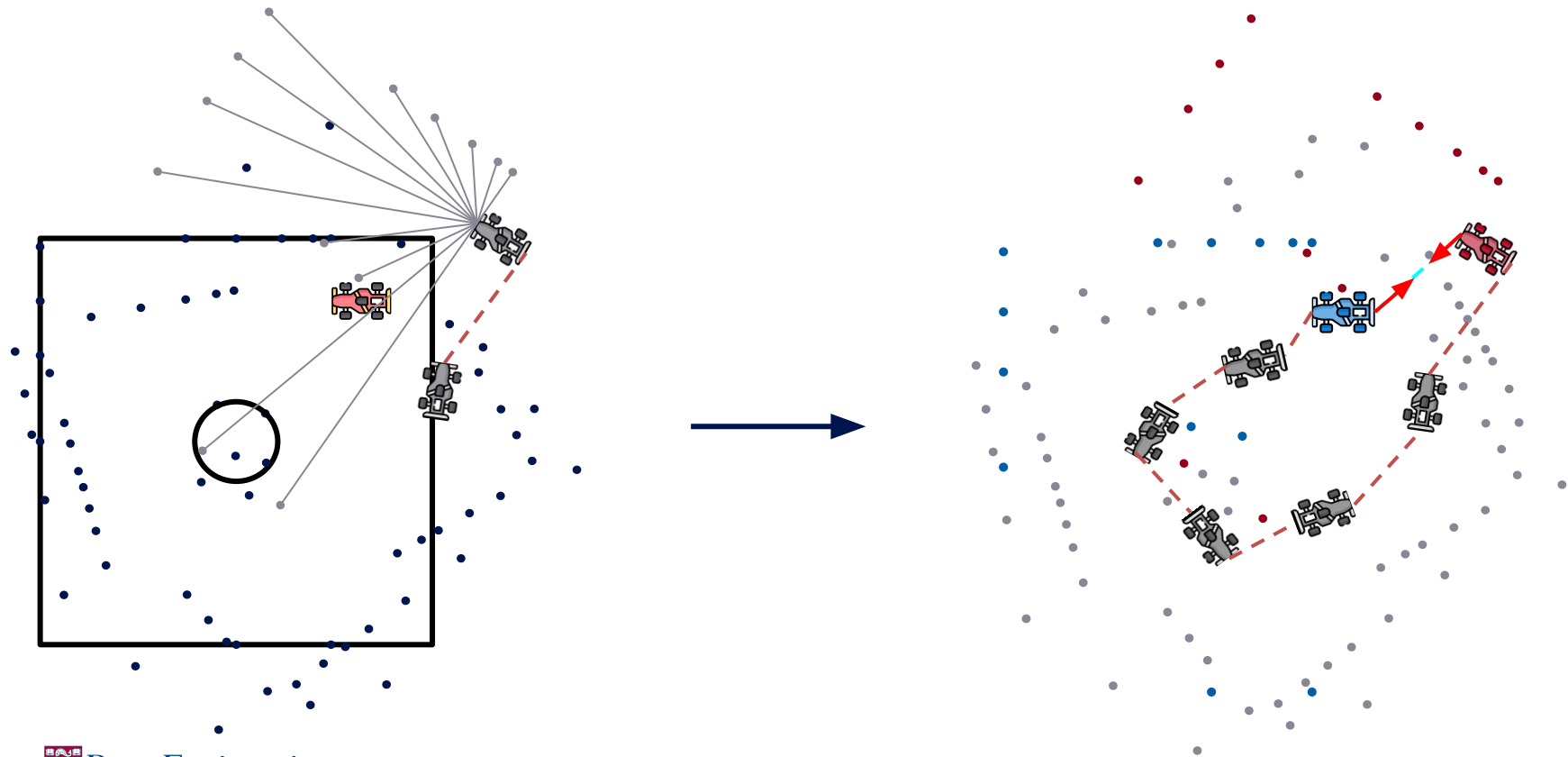




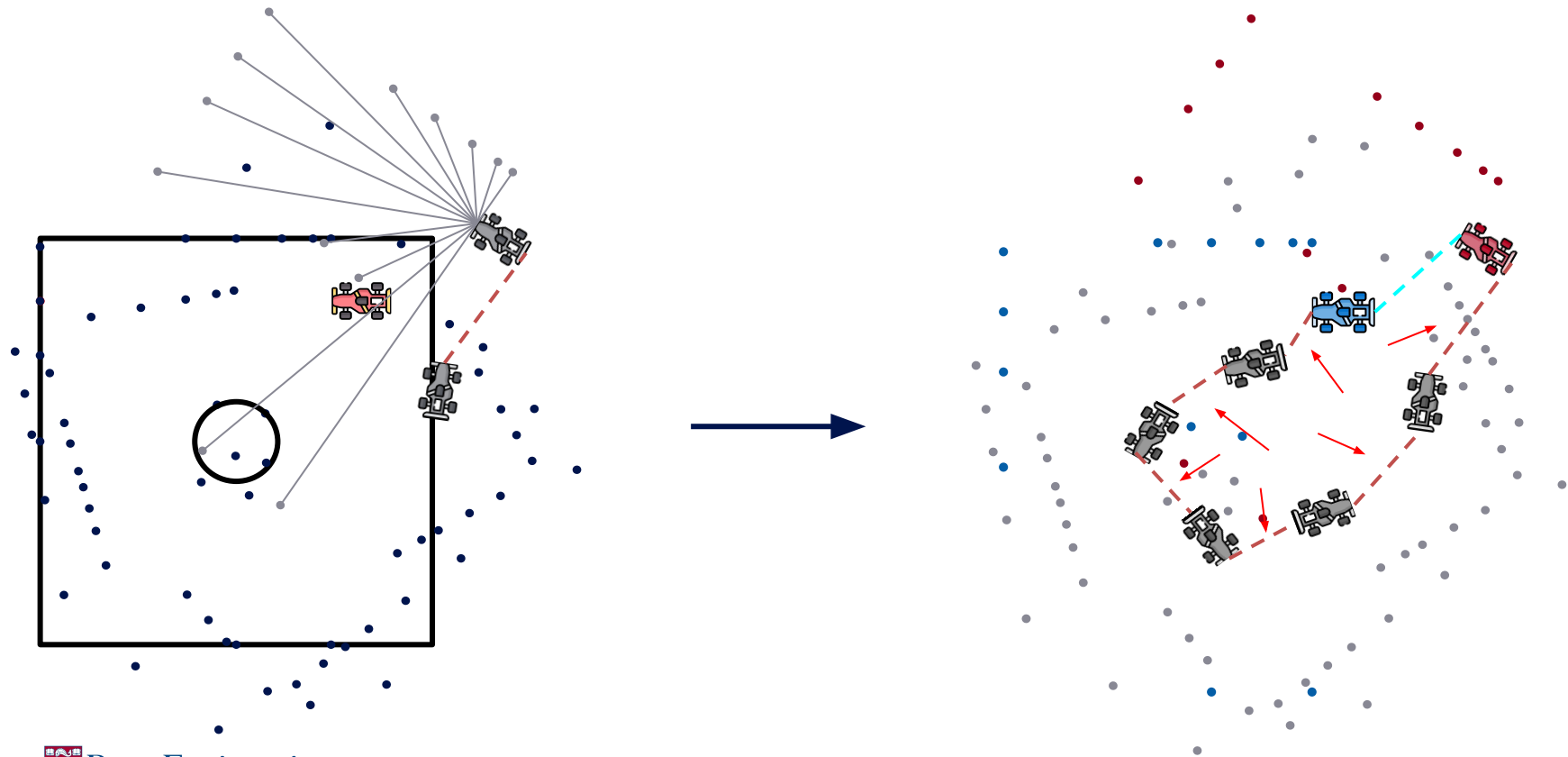
# Pose Graph Optimization: Loop closure



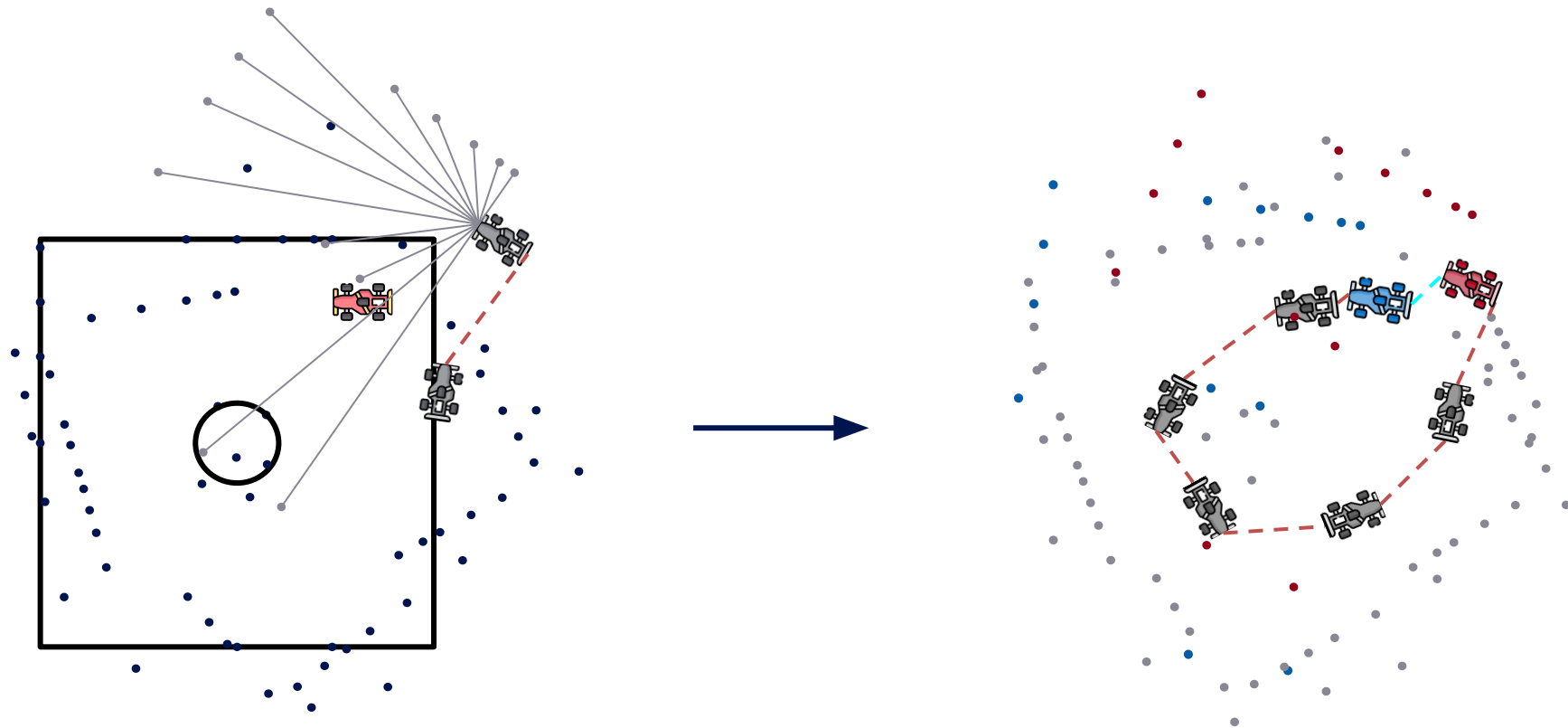
# Pose Graph Optimization: Loop closure



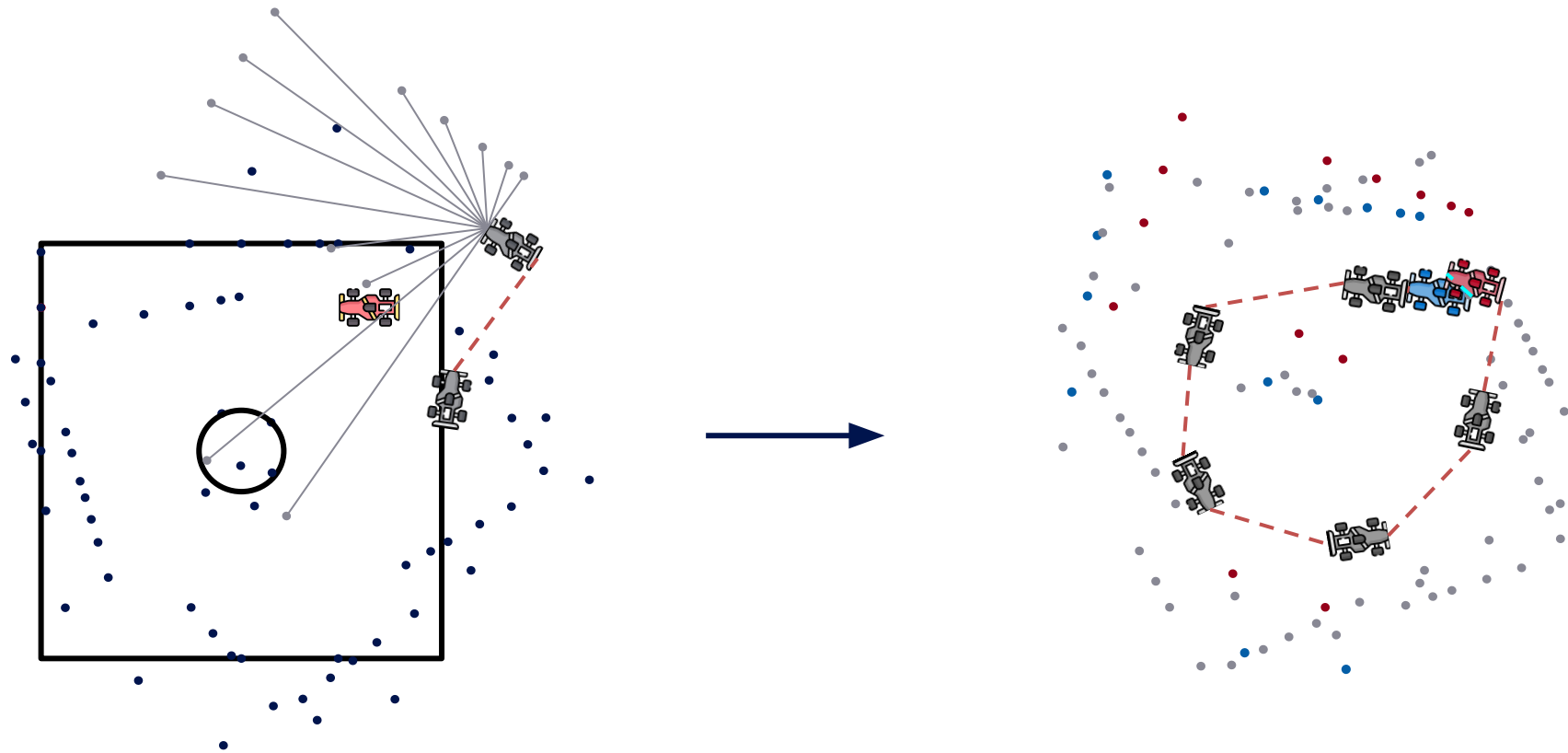
# Pose Graph Optimization: Loop closure



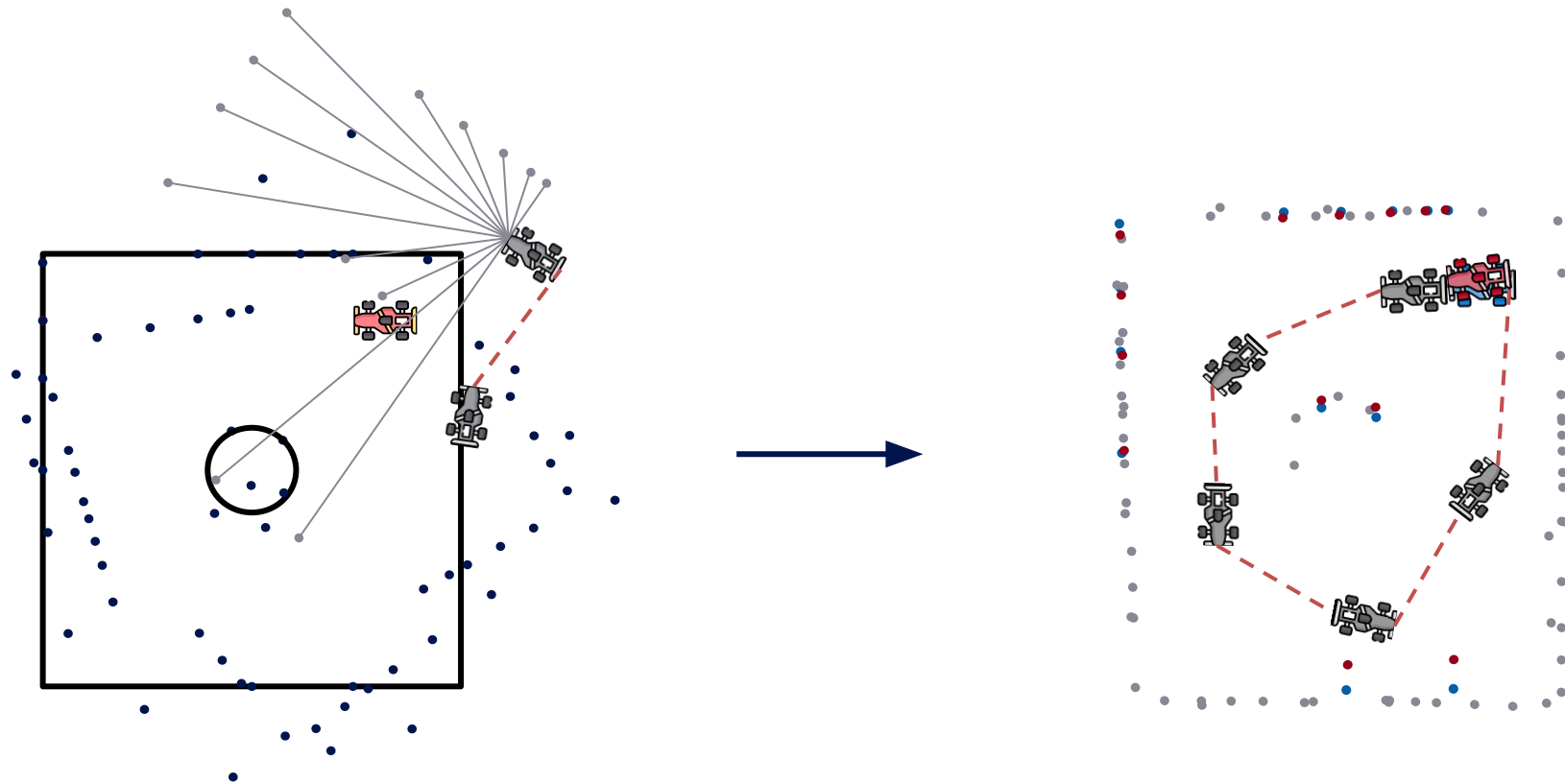
# Pose Graph Optimization: Loop closure



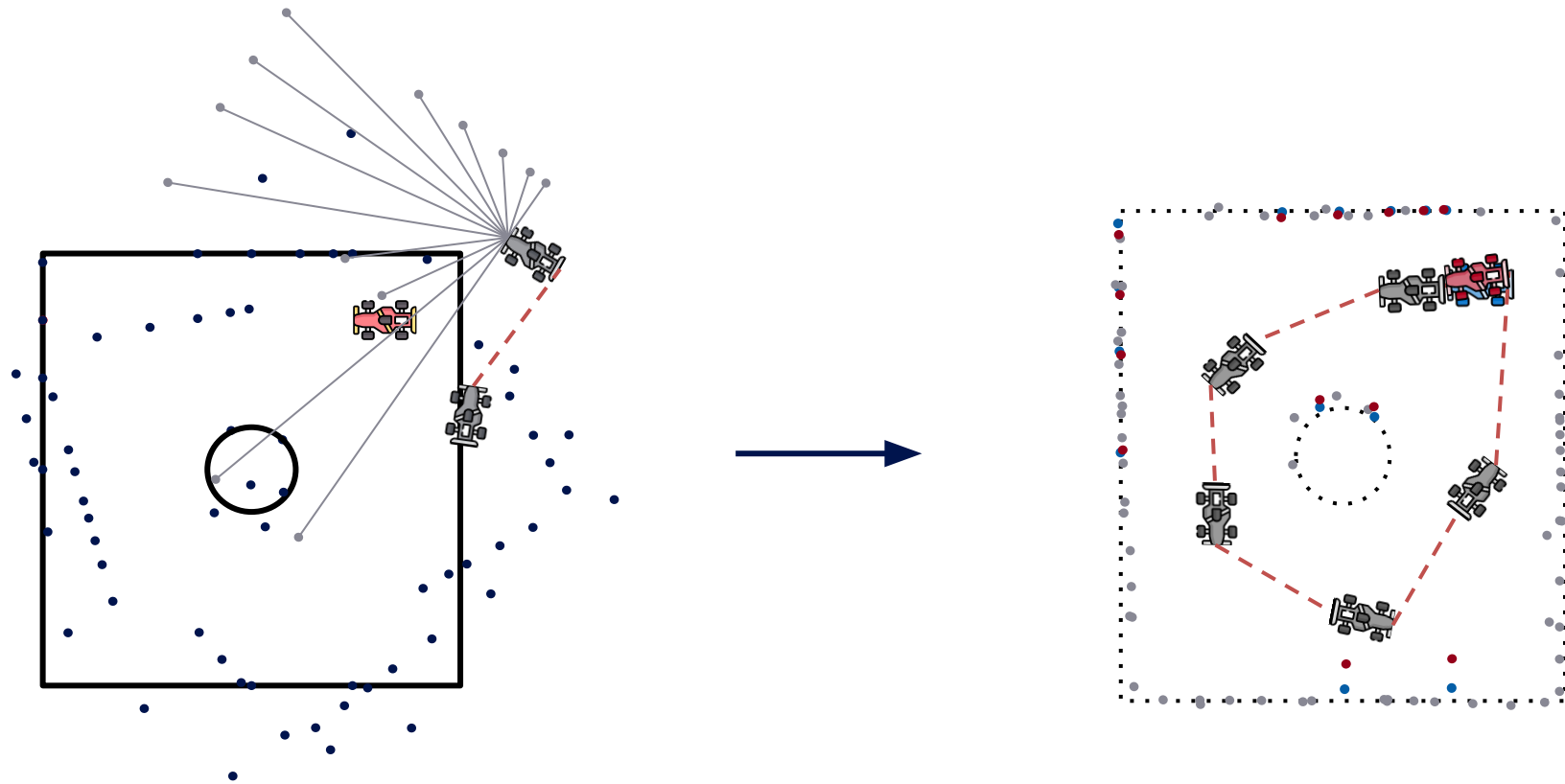
# Pose Graph Optimization: Loop closure



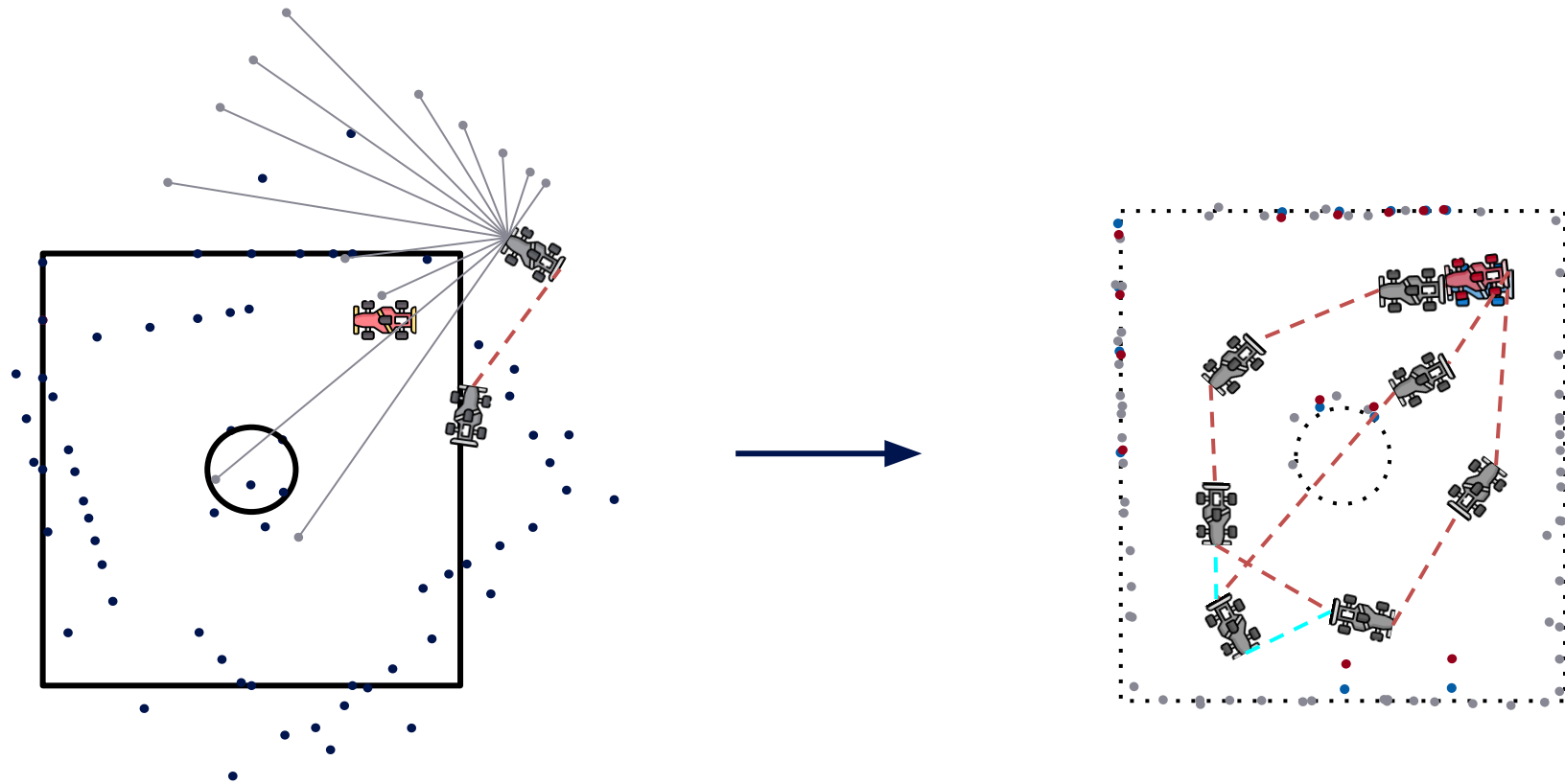
# Pose Graph Optimization: Loop closure



# Pose Graph Optimization: Loop closure

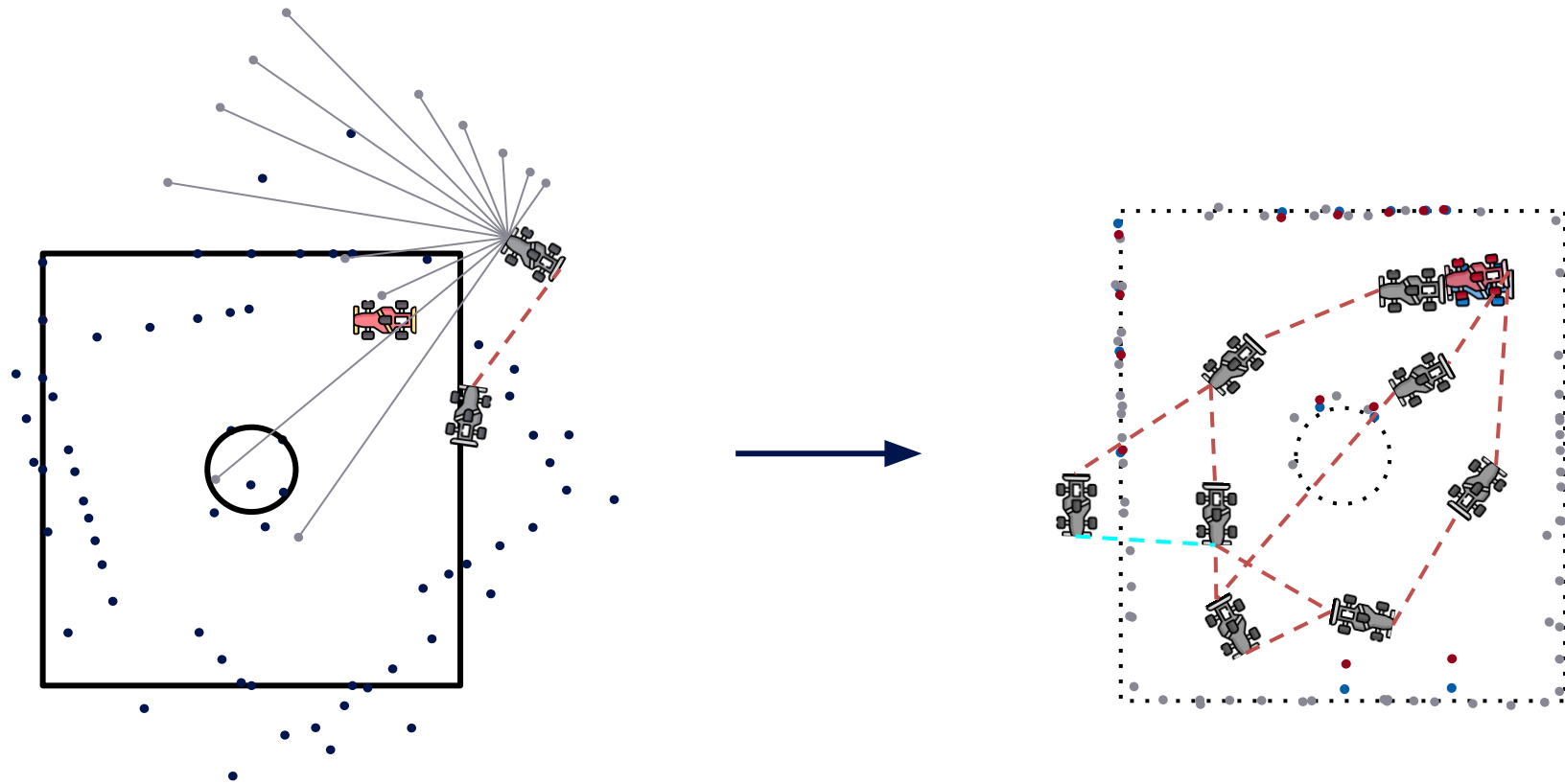


# Pose Graph Optimization: Loop closure

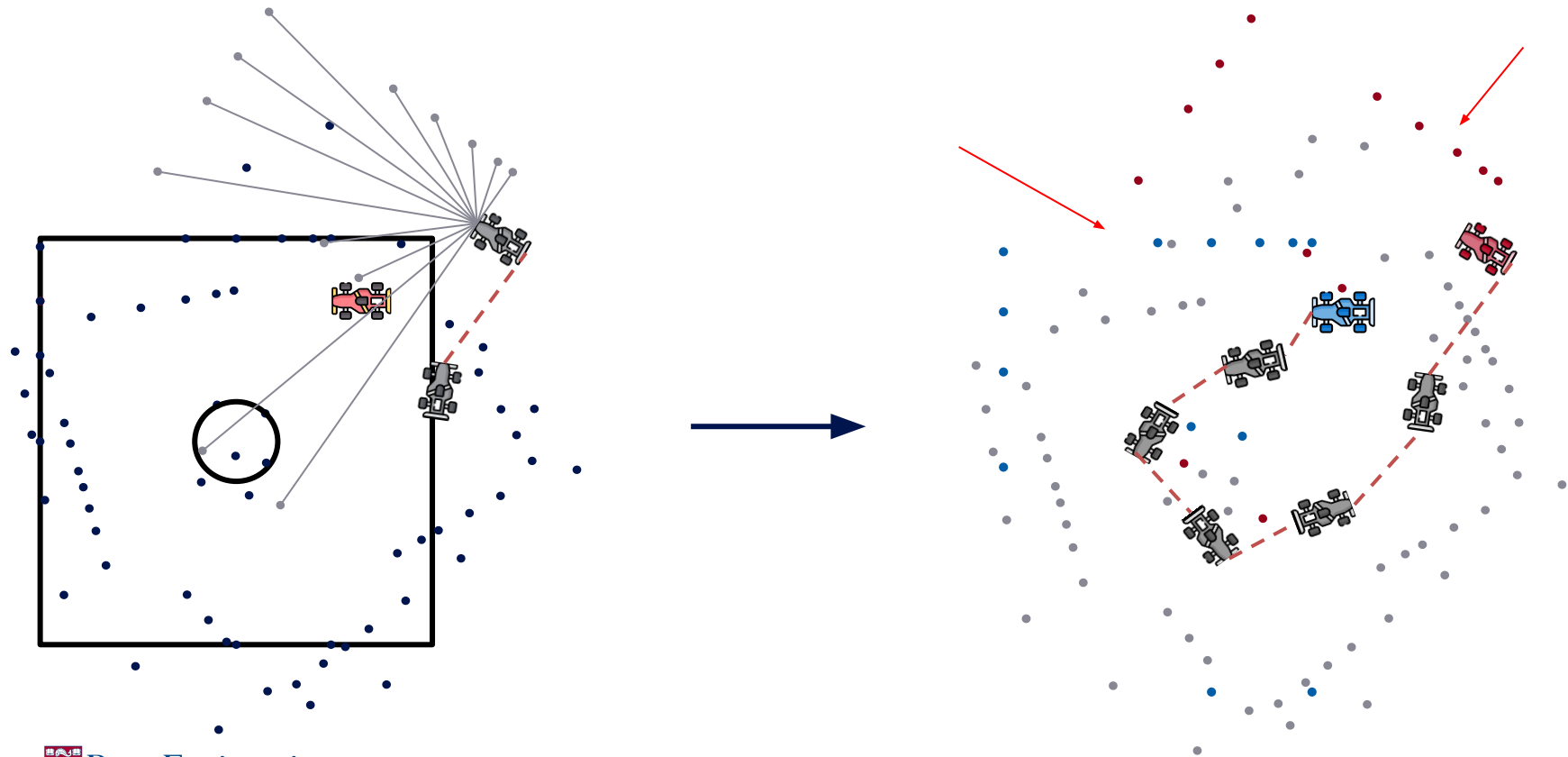




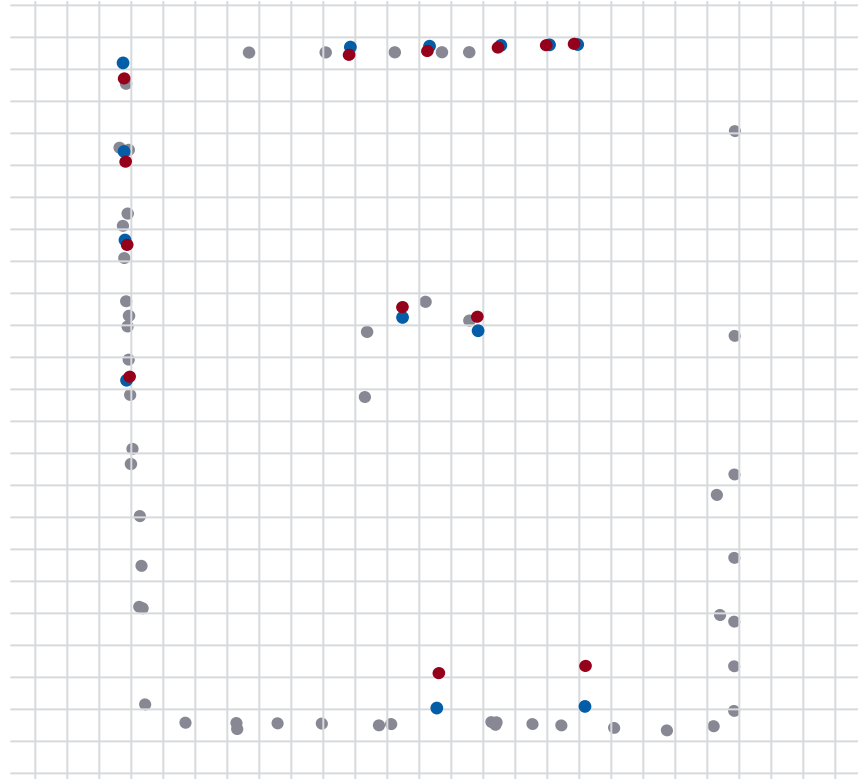
# Pose Graph Optimization: Loop closure



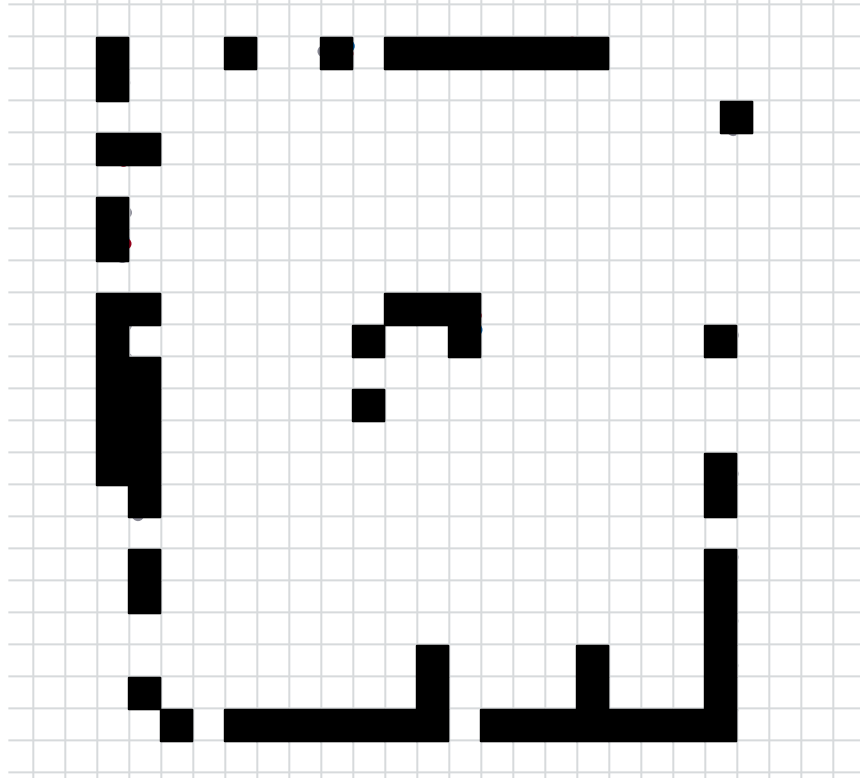
# Pose Graph Optimization: Loop closure



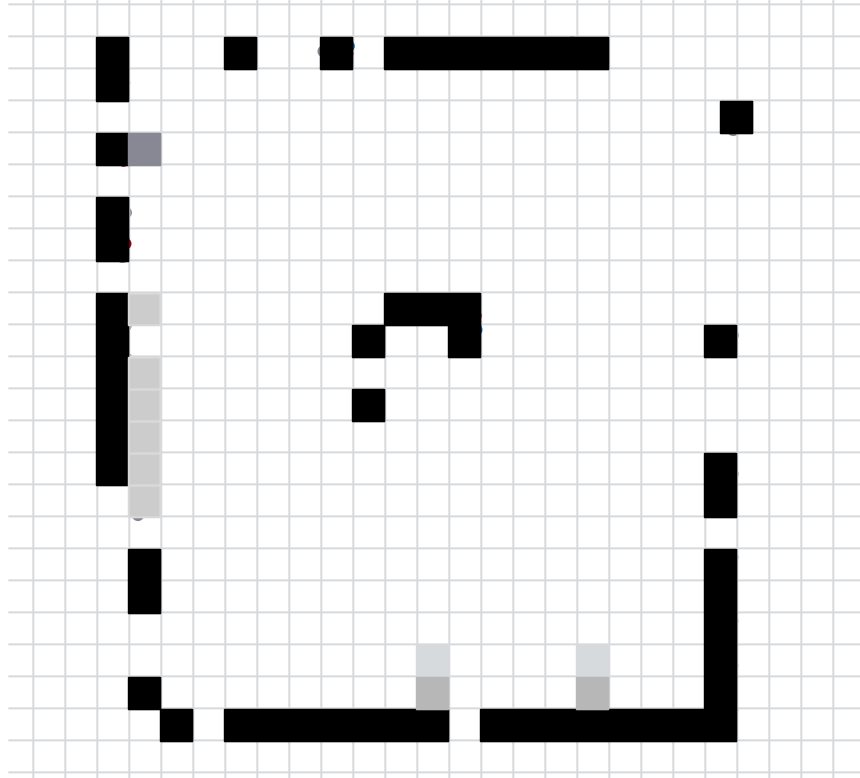
# How to store data as a map?



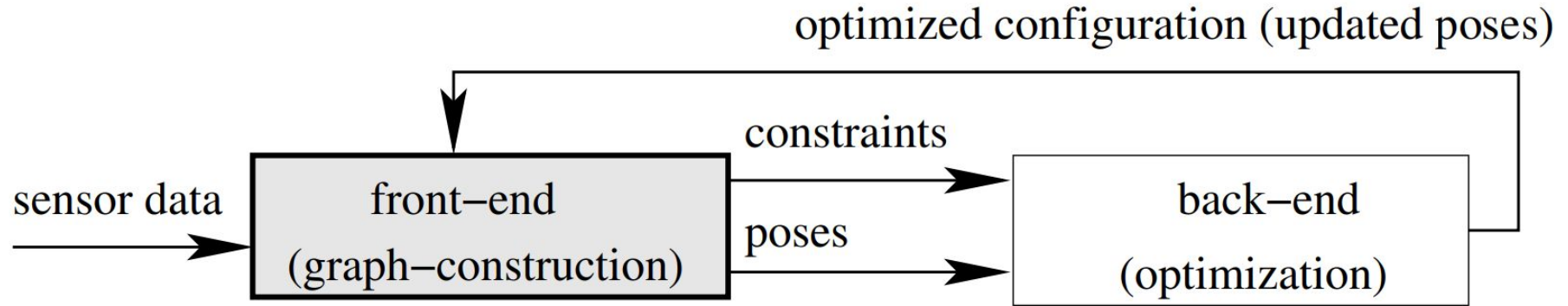
# How to store data as a map?



# How to store data as a map?



# Summary



# Final Notes

Graph-based SLAM is sensor agnostic. Only requires 2 types of sensors:

- One that can measure **relative** transformations between poses (IMU, odometry, wheel encoders, etc.). This sensor is used to build constraints between nodes.
- One that can measure **absolute** observation of the environment. (LiDARs, cameras, etc.). This sensor is used for loop closure

# Final Notes

Graph-based SLAM is also scan (sensor) matching method agnostic.

- ICP (what we learned in scan matching lecture)
- Scan-to-scan
- Scan-to-map
- Map-to-map
- Feature-based
- RANSAC for outlier rejection
- Bundle Adjustment
- **Correlative Scan Matching**



# Correlative Scan Matching

- PL-ICP/ICP explicitly computes correspondence between two scans for a transformation.
- Susceptible to local minima; poor initial estimates lead to incorrect data associations.
- **Correlative Scan Matching searches for transformations (without computing correspondences) that projects one set of scans on top of a reference map.**
- The reference map is generally implemented as a look-up table that assigns a cost for every projected query point.

# Probabilistic View of Scan Matching

- Previously we viewed scan matching as finding an absolute solution. Now we try to take a look at the same problem in a Bayesian POV.
- Let's say the robot moves from  $x_{i-1}$  to  $x_i$ , with some motion  $u$  and observation  $z$  depending on environment model  $m$  and robot's position.
- Our goal is to find the posterior  $p(x_i|x_{i-1},u,m,z)$ . With Bayes' rule:
$$p(x_i|x_{i-1}, u, m, z) \propto p(z|x_i, m)p(x_i|x_{i-1}, u)$$
- On the RHS, the first term is the *observation* model. And the second term is the *motion* model.
- Motion model is easy to make a guess (multivariate Gaussian, and estimate with our odometry). Our goal is to find the approximate the observation model.

# Probabilistic View of Scan Matching

- First assumption we make is that individual scan points are independently observed, thus:

$$p(z|x_i, m) = \prod_j p(z_j|x_i, m)$$

- For 2D lidar scans, each of these distributions will be based on which surface of the map  $m$  would be visible at  $x_i$ .
- Could use ray marching to find this, but expensive. (we actually use ray marching in our particle filter on the gpu for the same observation model).
- **Correlative Scan Matching** uses a 2D look up table for this. It contains the log probabilities of a lidar observation at each  $(x, y)$  position in the world.

# Correlative Scan Matching

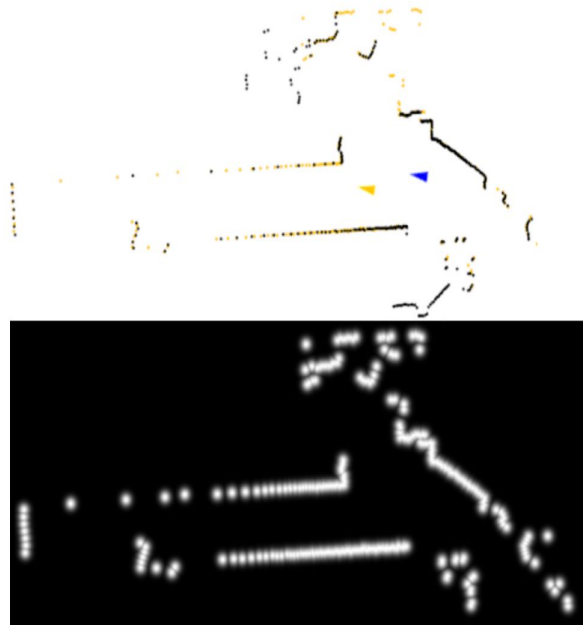


Image from olson2009icra

- Candidate transformations are scored according to a cost map (left figure).
- Rendered into the cost map with a blurring kernel to approximate uncertainty in scans.
- Candidate transformations are generated from plausible priors (e.g. from odometry).
- Covariance of the solution could also be principally calculated (this'll be important)

[https://www.cs.cmu.edu/~motionplanning/papers/sbp\\_papers/integrated3/konolidge\\_markov\\_correl.pdf](https://www.cs.cmu.edu/~motionplanning/papers/sbp_papers/integrated3/konolidge_markov_correl.pdf)

<https://april.eecs.umich.edu/pdfs/olson2009icra.pdf>

<https://april.eecs.umich.edu/pdfs/olson2015scanmatch.pdf>

# Correlative Scan Matching

Optimal transformation

All plausible transformation

$$T^* = \operatorname{argmax}_T \sum_{i \in N} L(T p_i)$$

Sum over all scan points

Lookup function

Points transformed into common frame of reference

# Final Notes

- The task of the pose-graph optimization is to seek for a configuration of the nodes that maximizes the likelihood of the measurements encoded in the constraints.
- We use Sparse Pose Adjustment to find the graph configuration.

# Sparse Pose Adjustment

Robot Poses

Rotation Matrix of  $\theta_i$

Translation

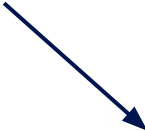
Measurement equation,  
or constraint, or offset


Difference in Angle

$$h(c_i, c_j) \equiv \begin{cases} R_i^\top (t_j - t_i) \\ \theta_j - \theta_i \end{cases}$$

# Sparse Pose Adjustment

Error function


$$e_{ij} \equiv \bar{z}_{ij} - h(c_i, c_j)$$



Measured offset between  $c_i$  and  $c_j$   
From scan matching, with a precision matrix (inverse of covariance)



# Sparse Pose Adjustment

Total Error


$$\chi^2(\mathbf{c}, \mathbf{p}) \equiv \sum_{ij} e_{ij}^\top \Lambda_{ij} e_{ij}$$

The optimal placement of each  $\mathbf{c}$  is found by minimizing the total error. This is a Least Square problem. We use the Levenberg-Marquardt (LM) framework to solve it.



Precision Matrix

# What's Next?

- A natural step after creating a map is to localize the robot within a known map.
  - We've done that already - Particle Filter
- With the ability to make a map and localize, we'll discuss motion planning in the next few lectures.
- We'll also go over how to run `slam_toolbox` on your cars today.