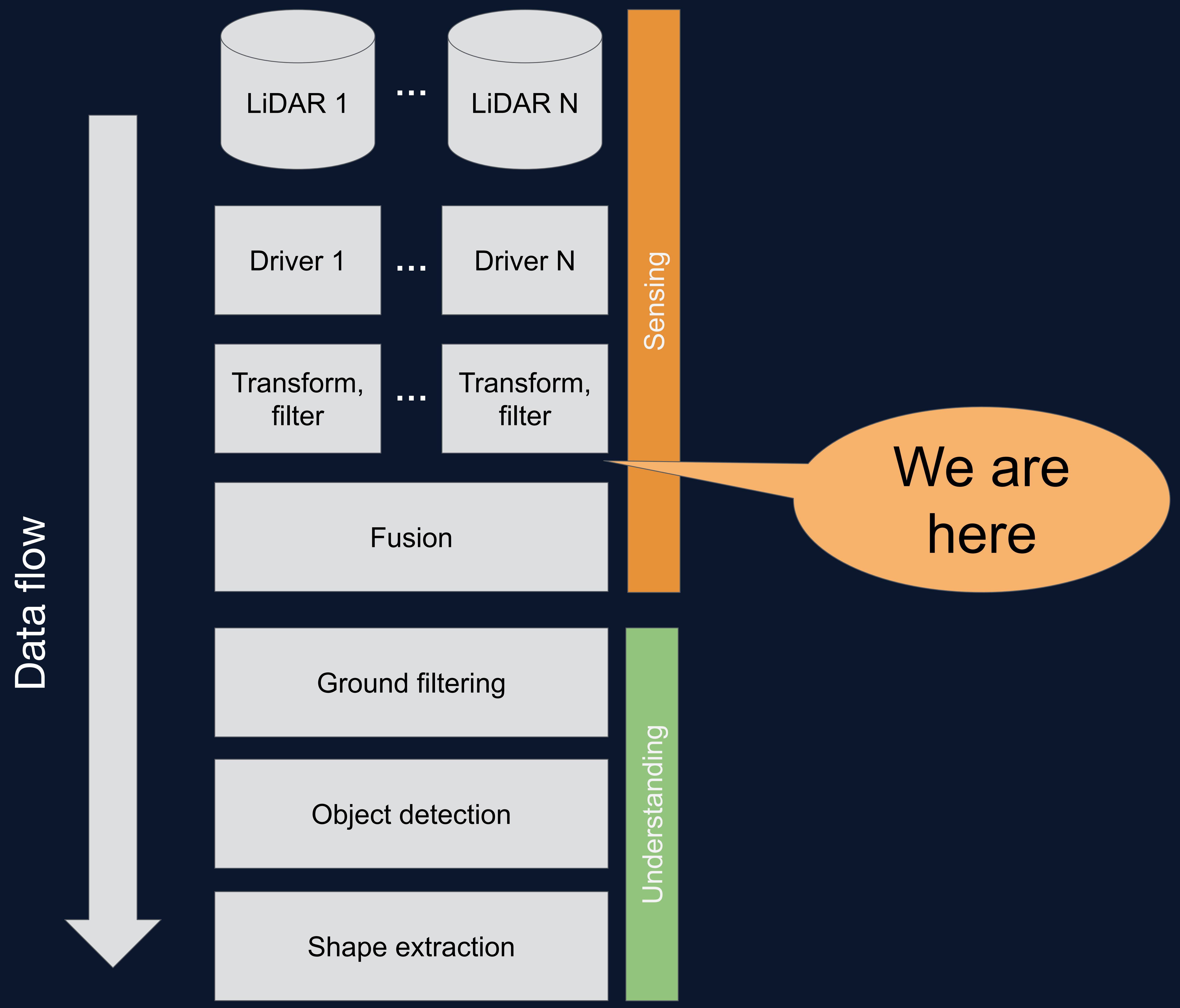
03 / LiDAR Preprocessing

Preprocessing in the Classical LiDAR Processing Stack

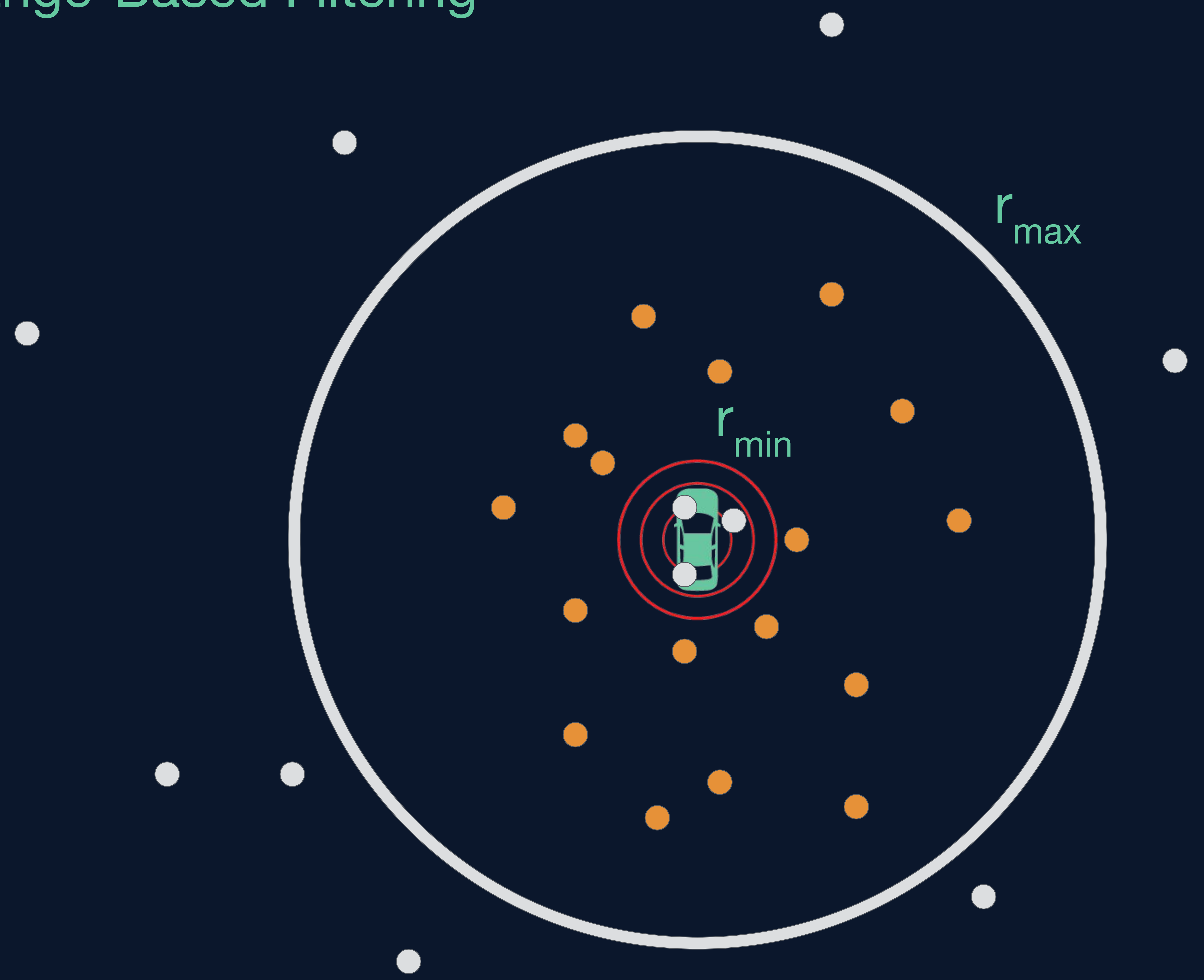


LiDAR Preprocessing, A Problem Statement

We want the minimum amount of information needed to produce the correct results:

- Remove useless data
- Remove problematic/bad data
- Remove redundant data
- Produce a single, consistent input

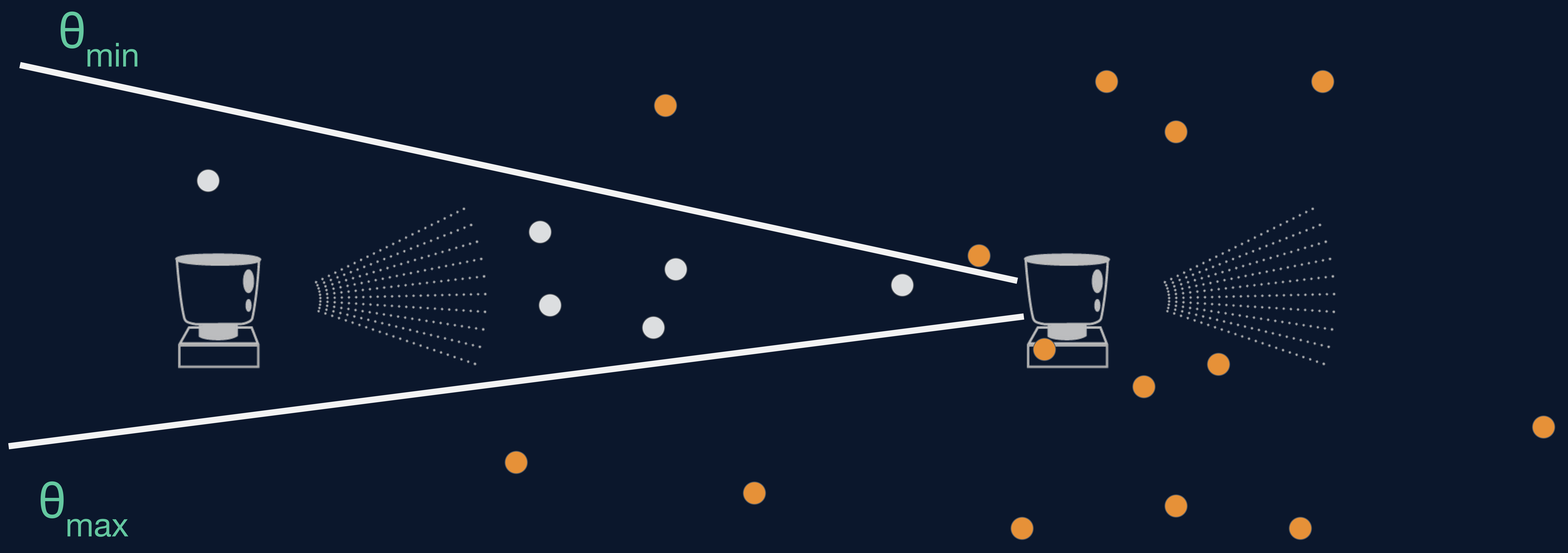
Range-Based Filtering



Remove points if $r < r_{min}$ or $r > r_{max}$:

- Distant points have no context
- Near points might fall onto ego vehicle

Angle-Based Filtering



Remove points if $\theta \notin (\theta_{min}, \theta_{max}]$

- Avoid problematic regions in sensor
  - Mitigate “flying birds” effect

Downsampling

Distill point cloud into a representative set of points

- Reduce computational complexity (make n smaller)
- Voxel grid approaches (Centroid, Approximate)
- Random sampling approaches

Fusing Point Clouds

Combine disparate point clouds into a single representation with respect to a common frame*

- Assume static vehicle -> appropriate for low-speed use cases
- Ideal handling requires ego-motion estimation (correct slewing)
- Can use message_filters or something else to obtain simple measurement alignment

LiDAR Preprocessing

Preprocessing gives you a single, lightweight representation needed by downstream algorithms

- Remove noisy data from problematic areas:
    - Range/angle filters
- Remove redundant data:
    - Voxel grids, other downsampling
- Create a single consistent representation:
    - Static transforms into common frame
    - Fuse into a single point cloud
    - Ego-motion is required for high speed use cases to correct for slewing

Preprocessing gives you the sufficient statistics for further algorithms