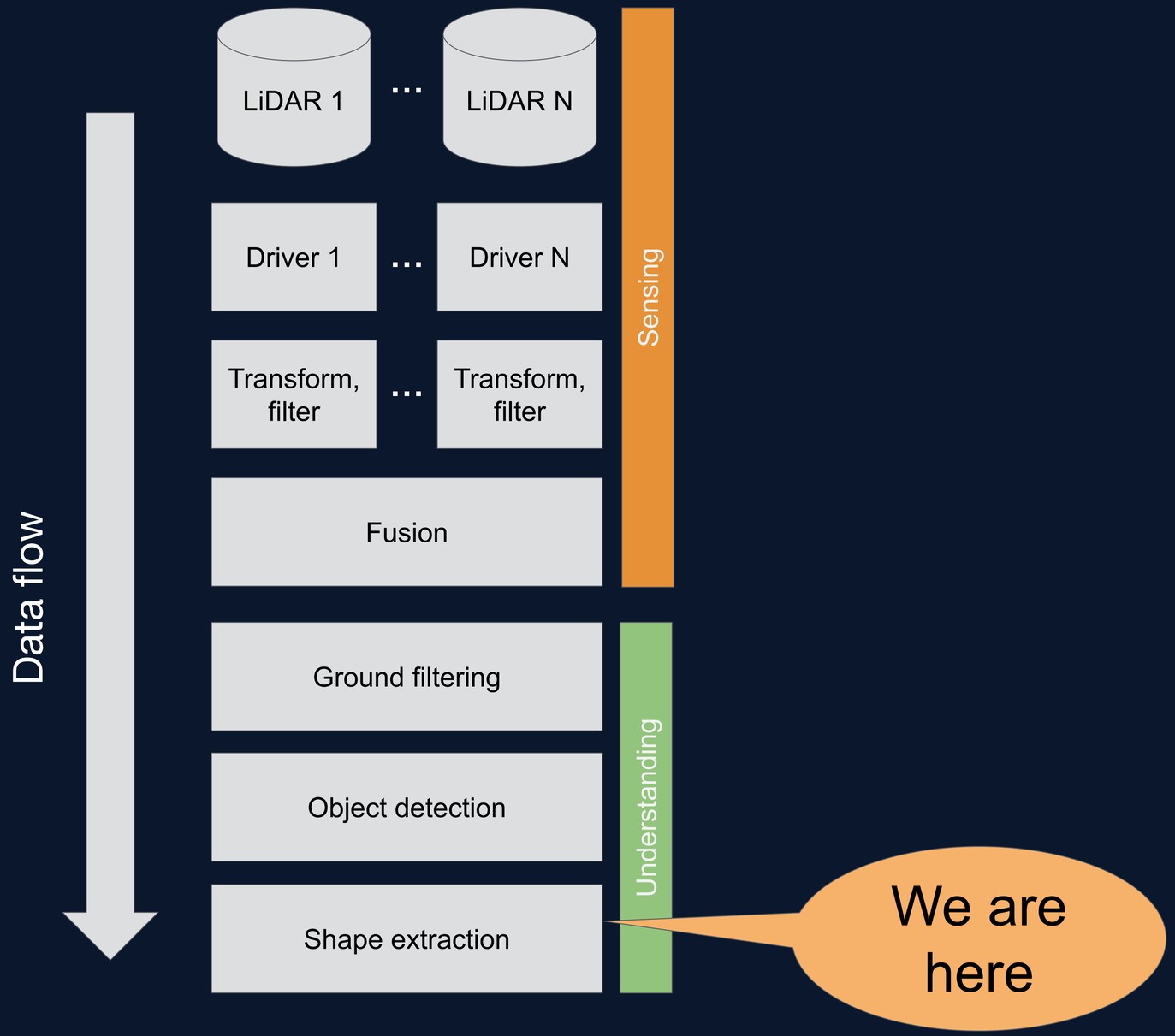
# 06 / Shape Extraction

# Shape Extraction in the Classical LiDAR Processing Stack



# An overview on shape extraction techniques

- Why?
- Different ways to compute bounding boxes:
  - Rotating calipers
  - Eigenvalues
  - Optimization
- Other shape extraction techniques

# Why not use point blobs directly?

Recall sufficient statistics:

- We only want the minimum of information to produce the right results
- Point blobs are big and unbounded
  - Additional communication and memory overhead
  - Additional computational overhead for collision checking, other use cases

# Shape extraction and the collision detection use case

## Object detection is generally to inform collision detection

The final object representation should:

- be easy to collision check
- be easy to compute
- reasonably represent the actual object

## Oriented bounding boxes are a good compromise

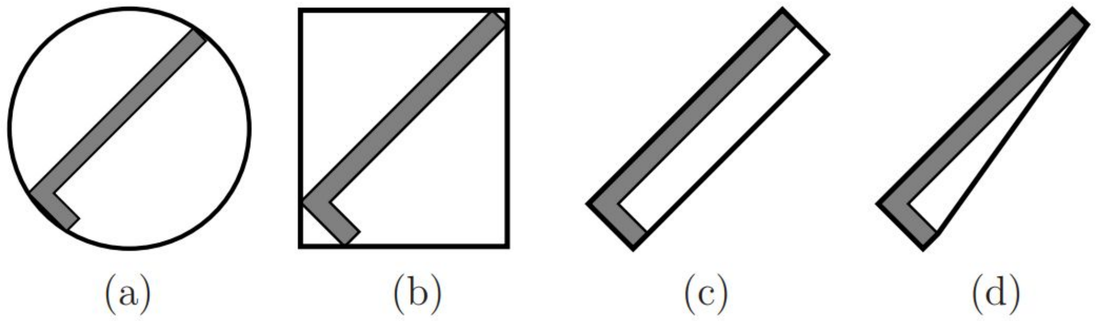
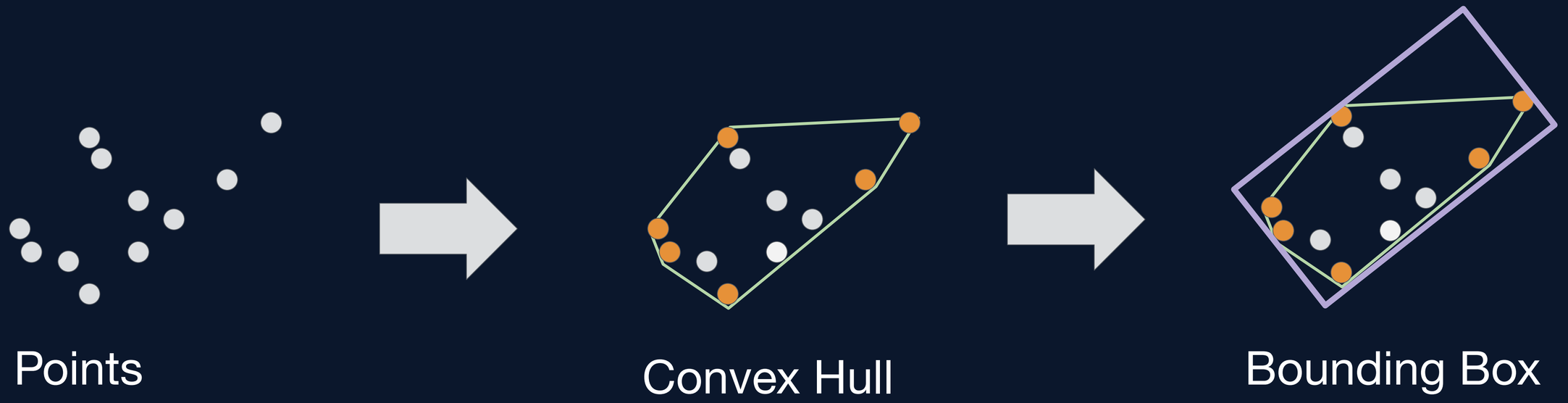



Figure 5.9: Four different kinds of bounding regions: (a) sphere, (b) axis-aligned bounding box (AABB), (c) oriented bounding box (OBB), and (d) convex hull. Each usually provides a tighter approximation than the previous one but is more expensive to test for overlapping pairs.

Image credit:
Planning Algorithms, Ch 5, Steven LaValle

# Minimum Bounding Boxes




- Rotating calipers: compute 2D bounding boxes with minimum perimeter or area in linear time
  - Requires a convex hull (computable in O(n log n) time)

# Minimum Bounding Boxes

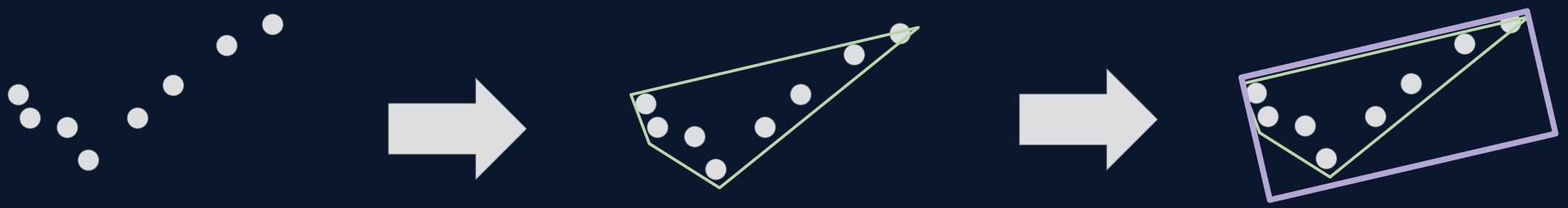
...but:

- only considers boundary points
  - approach is sensitive to noise
- it’s trivially easy to construct examples that won’t semantically fit data

# Minimum Bounding Boxes

...but:

- only considers boundary points
  - approach is sensitive to noise
- it’s trivially easy to construct examples that won’t semantically fit data, e.g.:



Points

Convex Hull

Bounding Box

# Eigenbox

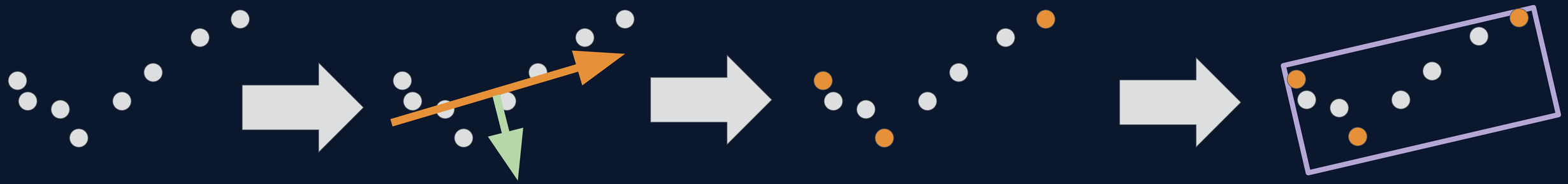
Idea: use PCA on all points to compute major and minor axes of bounding box

- Linear in time
- Uses information from entire object
- Bounded suboptimality guarantees

# Eigenbox

Idea: use PCA on all points to compute major and minor axes of bounding box

- Linear in time
- Uses information from entire object
- Bounded suboptimality guarantees
- ...but still doesn't produce semantically correct results



Points

Principal Components

Extent Points

Bounding Box

# Optimal Bounding Boxes

Can instead explicitly solve an optimization problem

- Shen et al
  - Box and partition that best fits L-shape
  - ...but relies on a single scanning LiDAR
- Zhang et al
  - Relaxes assumption of single scanning LiDAR
  - ...but introduces discretization error

$$
\begin{aligned}
\underset{P,\theta,c_1,c_2}{\text{minimize}} \quad & \sum_{i \in P} (x_i \cos\theta + y_i \sin\theta - c_1)^2 \\
& + \sum_{i \in Q} (-x_i \sin\theta + y_i \cos\theta - c_2)^2 \qquad (1) \\
\text{subject to} \quad & P \cup Q = \{1, 2, ..., m\} \\
& c_1, c_2 \in R \quad 0^\circ \leq \theta < 90^\circ
\end{aligned}
$$

# Bounding Boxes in Autoware.Auto

## Use approach from Shen et al, but:

- Compute principal component of blob
- Sort points along this axis
- Extra work, but:
  - No discretization error
  - Works on arbitrary point clouds

Unordered Points

Principal Component

Ordered Points

Partitioning Bounding Box Algorithm

# Further Approaches

## Many other ways to do shape extraction

- Convex hulls
  - Only loosely bounded memory complexity
- Minimum Volume Bounding Ellipsoids
  - e.g. Kumar and Yıldırım
  - Collision detection requires solving an optimization problem
- Super quadratics (i.e. Pascoal et al):
  - Handles concave shapes
  - Requires solving an optimization problem

## Some open problems

- Handling large, concave objects, e.g.
  - Inside of a small garage

# Shape Extraction - Summary

Shape extraction is necessary to reduce complexity:

- Communication
- Semantic
- Computational

There are many approaches:

- Bounding boxes
- Ellipsoids
- Convex hulls
- Others

And there are still some open problems:

- Large concave objects