

# LIDAR Feature Extraction

Noel Welsh

30 November 2010

- Information on the additional work required of Masters students (Level 4) is online
- Deadline for project report shifted back to 13 December 2010.
- Two best assignments and additional material on the web for Assignment 1
- New CMUCam GUI developed by David Rees available from the course web page (under Robot Kit Reference > CMUCam2)

- We were looking at localisation and LIDAR data. We're going to review this, then move on to LIDAR feature extraction.

- The localisation problem: “where am I?” Given a map.
- The simultaneous localisation and mapping problem (SLAM): “where am I?” *and* “what is the map of the world?”

# Types of Maps

- Metric maps
  - Precise location of all objects of interest
- Topological maps
  - Vaguely defined: any map that is not metrically precise
  - Usually a graph: nodes are places and edges are connections between places

# Metric Map Example

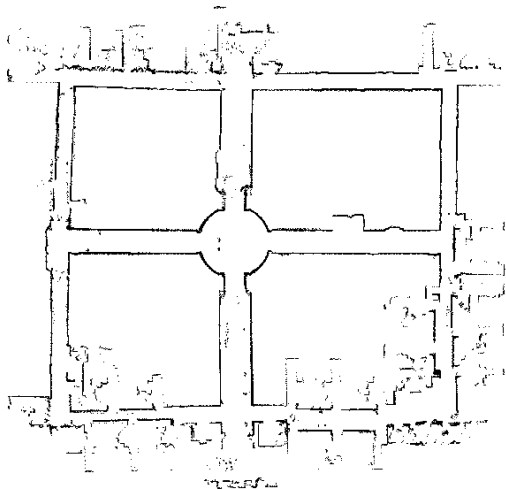


Figure: A metric map of the ACES building (Austin)

# Topological Map Example

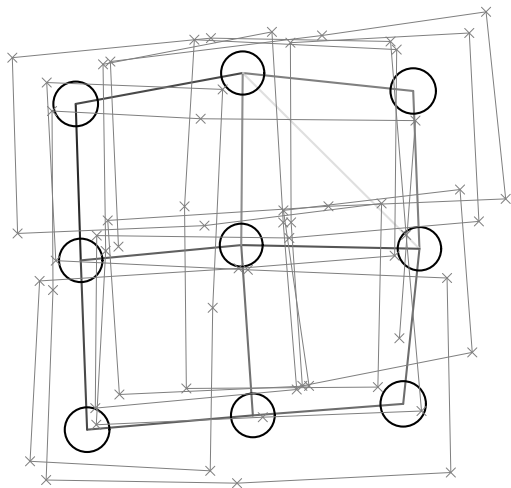


Figure: A topological map of the ACES building (Austin)

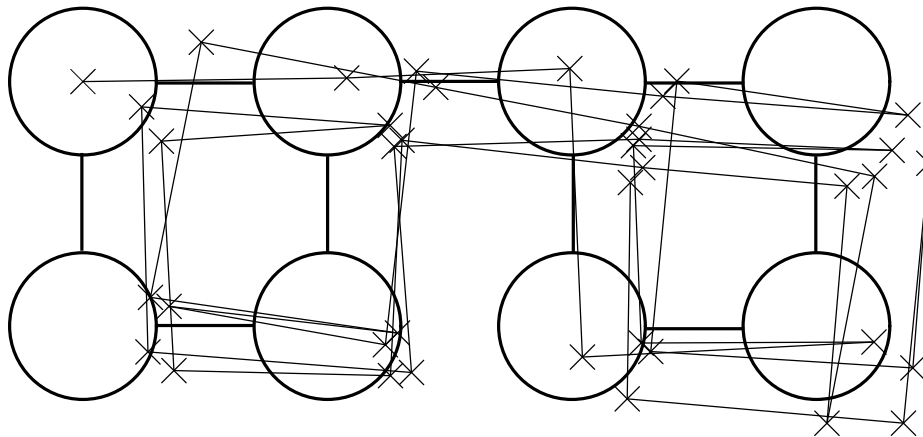
# The Particle Filter

- We looked at the particle filter as a method of solving the localisation problem. Uses two sources of information:
  - The observations the robot makes.
  - The controls we have sent to the robot.
- Basic idea:
  - Represent each hypothesised location by a *particle* with associated *weight*.
    - The particles are vectors containing  $x$ ,  $y$ , and bearing
    - The weights are proportional to the probability the particle's hypothesis is true

# Dealing with Control

- Build motion model for our robot. E.g. assume turns and forward motion are corrupted by Gaussian noise. Can (and should) measure this from the real robot.
- Given the previous location (particle), can *sample* from the motion model to approximate the distribution over the current location.
- Error grows without bound

# Odometry Error



**Figure:** Without reference to external data sources, odometry error grows without bound

# Dealing with Observations

- Need to answer  $P(\text{Observations}|\text{Particle})$ . That is, “what is the probability of observing the data we did observe, given the hypothesised location?”
- We have seen simple sensor models (e.g. Naive Bayes)
  - Not adequate for dealing with complex sensors.

# Range Sensors

- Range sensors are very common on mobile robotics
  - E.g. infra red, sonar, lidar
- Typically scan in an arc, sampling every degree. E.g. 270 degree scan, gives 270 range samples
- Can get 3D scanners – much more expensive
- Need a probability model to plug into our particle filter algorithm
  - (Not specific to particle filter – other localisation algorithms work the same way)

# LIDAR Example

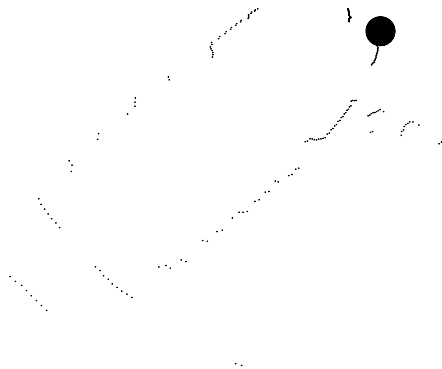


Figure: An example 2D LIDAR scan

- Two main problems
  - “Have I been here before”?
  - “What transformation best aligns this scan to a previous one”?
- The first is a classification problem.
- Techniques for the second can also be used to solve the first
- Two main classes of techniques:
  - Feature (also known as landmark or keypoint) extraction
  - Scan alignment

- Represent the data by a few distinctive features. E.g. sharp corners.
- Why?
  - Sparse representation of data
    - Faster search
    - Less storage
  - Features are cheap to compute

# What is a Feature?

- For classification purposes a feature is any representation that helps classify.
- This is a circular definition. The concept is ill-defined.
- The ideal feature is:
  - Easy to compute
  - Compact representation (low storage, fast to compare)
  - Robust against noise – we can find it every time we look
    - Particularly, rotationally invariant
  - Location doesn't change each time we find it

- Noise
- Occlusion
- Limited field of view
  - Often solve by using two sensors to get full circle scan

- Feature extraction is reasonably well developed in images
  - E.g. SIFT, SURF features are commonly used.
- Less well developed for range scanners. There is no standard method. Recent work:
  - [Bosse and Zlot(2009)]
  - [Li and Olson(2010)]
  - We'll focus on Bosse and Zlot. It is simpler to understand and implement, though less elegant.

# Feature Extraction Pipeline

- Two stage process:
  - Find features in scans
  - Process features to achieve compact representation

- Three methods to find feature *location* ( $x$  and  $y$  coordinates)
  - Segment centroids
  - Curvature clusters
  - Mean shift

# Segment centroids

- Scan points that are less than a certain distance apart are assigned to the same segment.
- Merge segments from different scans together if any point in the segments are less than a threshold apart.
- The *centroid* of each segment is its location.
- A *certain distance* is an environment dependent distance that you have to choose.
- Works well outdoors, where there are tight clusters of points for trees, people, etc. Probably fails indoors where walls won't segment easily.

- *Curvature* is essentially a measure of how much a curve differs from a straight line.
- It is the second derivative of a function.
- The first derivative is the rate of change or slope. The second derivative is the rate of change of the slope. The slope of a straight line doesn't change.
- Curvature clusters are formed by clustering together points with high *positive* curvature.
  - Positive curvature is like a ball
  - Negative curvature is like the inside of a ball. Tends to indicate occlusion.
- Second derivative is noisy to these features don't localise well.

- Mean shift iteratively recomputes a locally weighted mean till convergence.

$$\mu_{t+1} = \frac{\sum_i p_i W(p_i - \mu_t)}{\sum_i W(p_i - \mu_t)} \quad (1)$$

- $p_i$  is the  $i^{\text{th}}$  point.
- $W$  is a function that assigns decreasing weight to  $p_i$  the further it is from the mean.
  - E.g. a Gaussian!
- Ideally compute using every point as the starting mean and store the unique convergence points as the feature locations.

- The three methods each compute a number of feature locations. We then need to compute orientation. Their method is:
  - Compute a distance weighted histogram of all orientations of the points near the location
  - Select the peak of the histogram as the orientation
  - If there is more than one peak, duplicate the feature location as necessary.

# Feature Detection Summary

- To determine feature location run:
  - Segment centroids
  - Curvature clusters
  - Mean shift
- To determine feature orientation, use locally weighted histogram
- Fairly ad-hoc, with lots of parameters to specify



Michael Bosse and Robert Zlot.

Keypoint design and evaluation for place recognition in 2D lidar maps.

*Robotics and Autonomous Systems*, 57(12):1211–1224, 2009.  
ISSN 09218890.

doi: 10.1016/j.robot.2009.07.009.

URL <http://linkinghub.elsevier.com/retrieve/pii/S0921889009000992>.



Yangming Li and Edwin B. Olson.

A General Purpose Feature Extractor for Light Detection and Ranging Data.

*Sensors*, 10(11):10356–10375, November 2010.  
ISSN 1424-8220.

doi: 10.3390/s101110356.

URL <http://www.mdpi.com/1424-8220/10/11/10356/>.