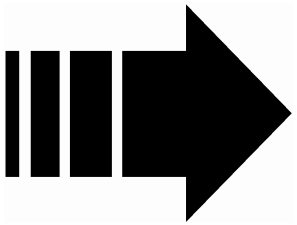
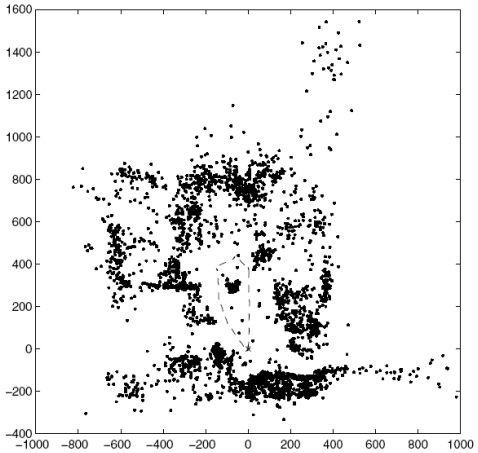
SLAM with SIFT

(aka Mobile Robot Localization and Mapping with Uncertainty using Scale-Invariant Visual Landmarks)

Se, Lowe, and Little

+





Presented by Matt Loper

CS296-3: Robot Learning and Autonomy

Brown University

BROWN

April 4th, 2007

BROWN

Objective

- SLAM: Simultaneous Localization and Mapping
  - Inputs: odometry and sensor readings, over time
  - Outputs: a map, and the set of locations traversed on it
- Uncertainty modeling...
  - on the inputs is essential, since sensors and odometry are noisy
  - in the output is great if you can do it (this paper does).

BROWN

Overview

- SIFT Stereo
- Ego-motion estimation: improving on odometry
- Landmark Tracking: remembering what we've seen
- First Results
- Heuristic improvements: tricks to improve success
- Error Modeling
- More Results

SIFT Stereo

- Sensors: typical sensors include...
  - Laser
  - Sonar
  - Vision, with an instrumented environment
- Instead, this paper uses the Digiclops, a trinocular camera rig

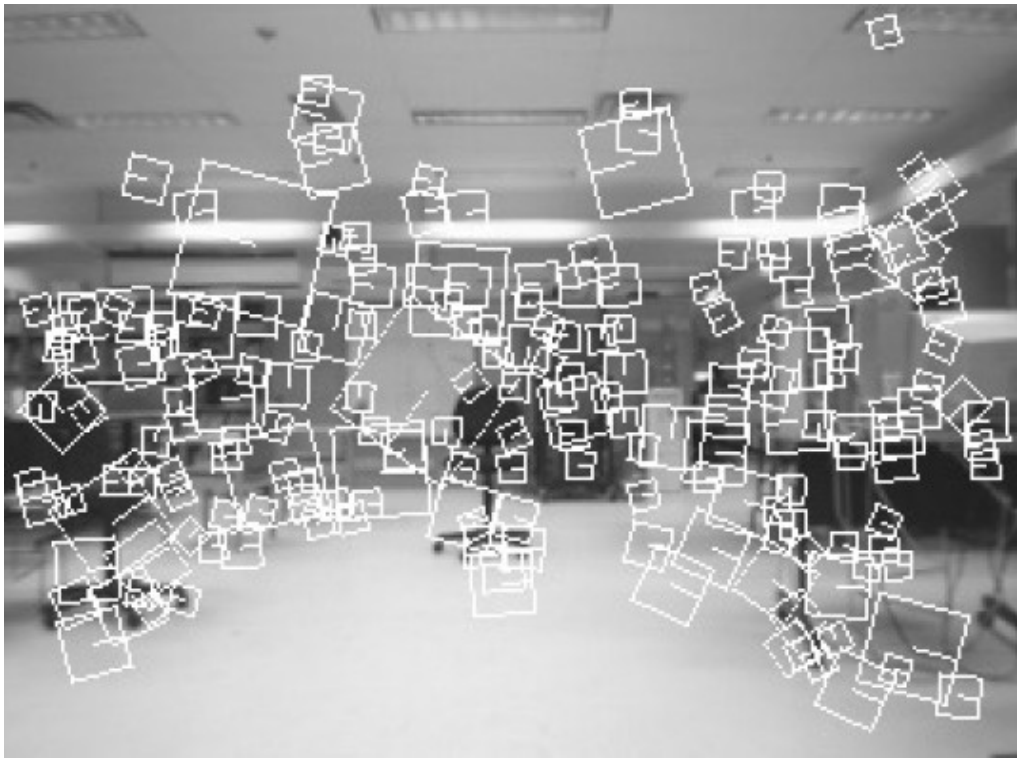
BROWN

SIFT Stereo

- Goal: find features in images, and compute their 3D relative position
- Three Steps:
  - Detect unique features in each of 3 cameras
  - Match across cameras
  - Recover feature positions in 3D, relative to camera

SIFT Stereo

- **Step 1 of 3**: Detect unique features in each of 3 cameras
- SIFT allows detection and characterization of image features



SIFT Stereo

- **Step 2 of 3**: Match features across cameras
- Given our three input images, we could just match all features on each against all others
- But we can constrain matches further:
  - Epipolar constraint
  - Disparity constraint: for a rightmost camera, objects should appear more to the left
  - Orientation, scale should be about the same in the two images
  - Should be only one good match

SIFT Stereo

- Constraint demonstration

Left Camera

Right Camera

SIFT Stereo

- **Step 3 of 3**: Recover feature positions in 3D, relative to camera
- Approach #1: what they did
  - The size of the disparity gives the distance
  - We have two disparities, so we could take the average
- Approach #2:
  - We could project rays through the pixels, and find the closest intersection

SIFT Stereo

- Note vertical and horizontal disparities

Left Camera

Right Camera

BROWN

Ego-motion estimation

- Odometry gives us change in position/orientation
- Let's refine these moment-to-moment estimations of change in position/orientation
- Note: We will still have the problem of slowly-degrading sums of differences! It will just be less extreme

BROWN

Ego-motion estimation

- Find matches across two frames
- Just as we had constraints between two cameras, we can have constraints between two frames
  - position, scale, orientation, disparity
- Given >= 3 matches, we can solve for change in camera position/orientation
- Given odometry as initializer, minimize reprojection error (could use, for example, Matlab's `lsqnonlin`)

BROWN

Landmark Tracking

- Let's turn SIFT features into landmarks
- For each 3D SIFT feature found, let's associate some data with it....
  - X,Y,Z: 3D position in the real world
  - s: scale of landmark in real world
  - o: orientation of landmark in real world (from top down)
  - l: count of number of times this landmark has been unseen, when it should have been seen. This is like a measure of error for this landmark.

BROWN

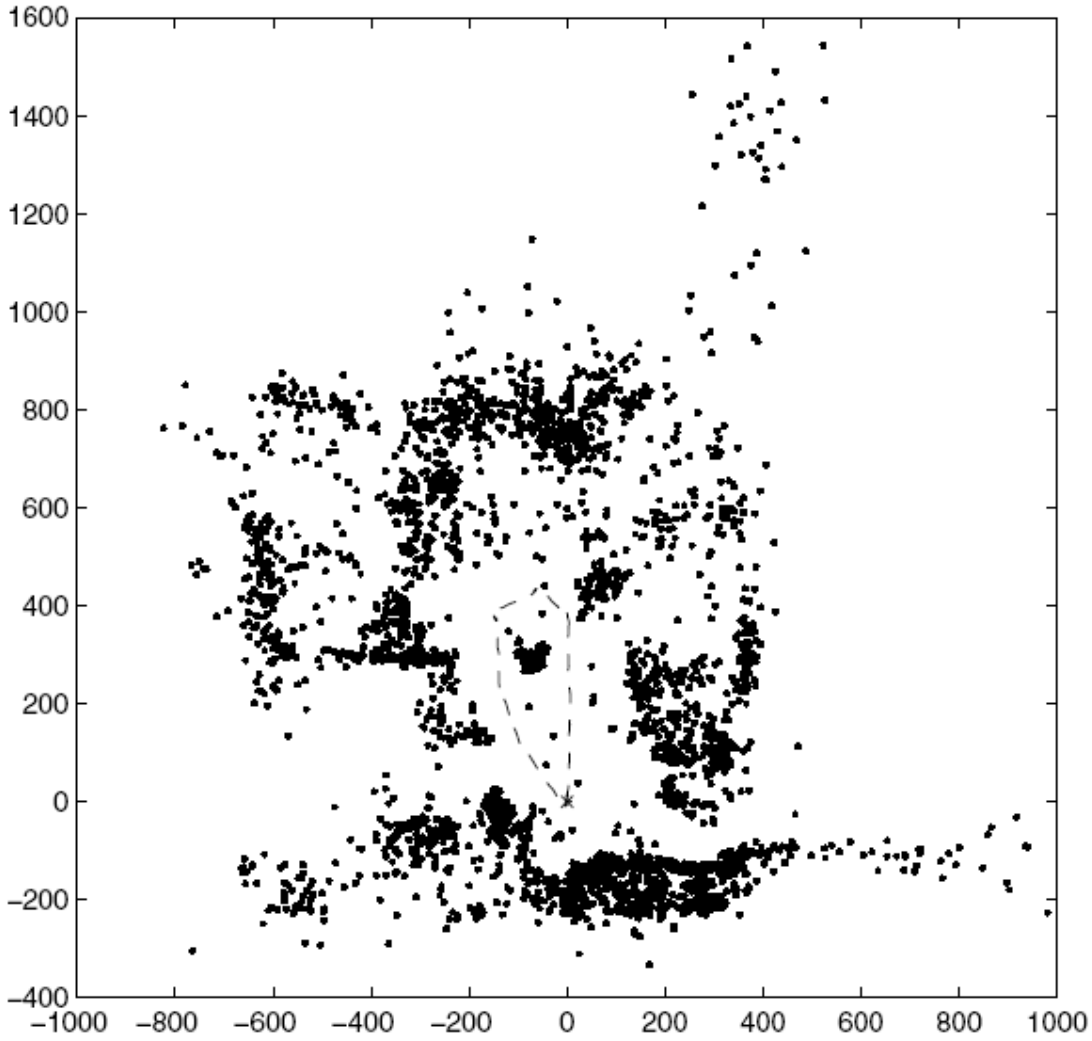
Landmark Tracking

- Rules for landmarks:
  - If a landmark is expected to be seen...
    - ...but is not seen, that landmark's “missed” count goes up by one
    - ...and is seen, reset its miss count to zero
  - If a new landmark is found, add it to the database and set its missed count to zero
  - If “missed” count goes above 20, remove landmark

BROWN

First results

- Generated map
- Starting point at cross
- Path on dotted line
- Come back to starting point, and measure error
- That error is very small



BROWN

Heuristic improvements

- Only insert new landmarks if they've been seen a few frames
- Build “permanent” landmarks when the environment is clear
- Associate view vector with each landmark
  - If we're “behind” the landmark, or more than 20 degrees to the side of it, don't expect to see it
- Allow multiple features per landmark (ie multiple scales, orientations, and view vectors)

BROWN

Error Modeling: Robot Position

- Uses a Kalman filter
- During each frame, we should maintain...
  - **X: State**
  - **P: Uncertainty** over that state
- **State**: position and angle, ie...
  - position: x,y
  - angle: yaw, pitch, roll
- **Uncertainty** over that state:
  - a covariance matrix P (ie ellipsoidal Gaussian)

Error Modeling: Robot Position

- x_{LS} is pose predicted by SIFT stereo
- P_{LS} is covariance over this prediction, computed as inverse of $J^T J$
- The state update is

$$\mathbf{x}(k+1|k+1) = \mathbf{x}(k+1|k) + \mathbf{W}(k+1)[\mathbf{x}_{LS} - \mathbf{x}(k+1|k)].$$

The covariance update is

$$\begin{aligned}\mathbf{P}(k+1|k+1) &= \mathbf{P}(k+1|k) - \mathbf{P}(k+1|k) \\ &\quad [\mathbf{P}(k+1|k) + \mathbf{P}_{LS}]^{-\mathrm{T}} \mathbf{P}(k+1|k)^{\mathrm{T}}.\end{aligned}$$

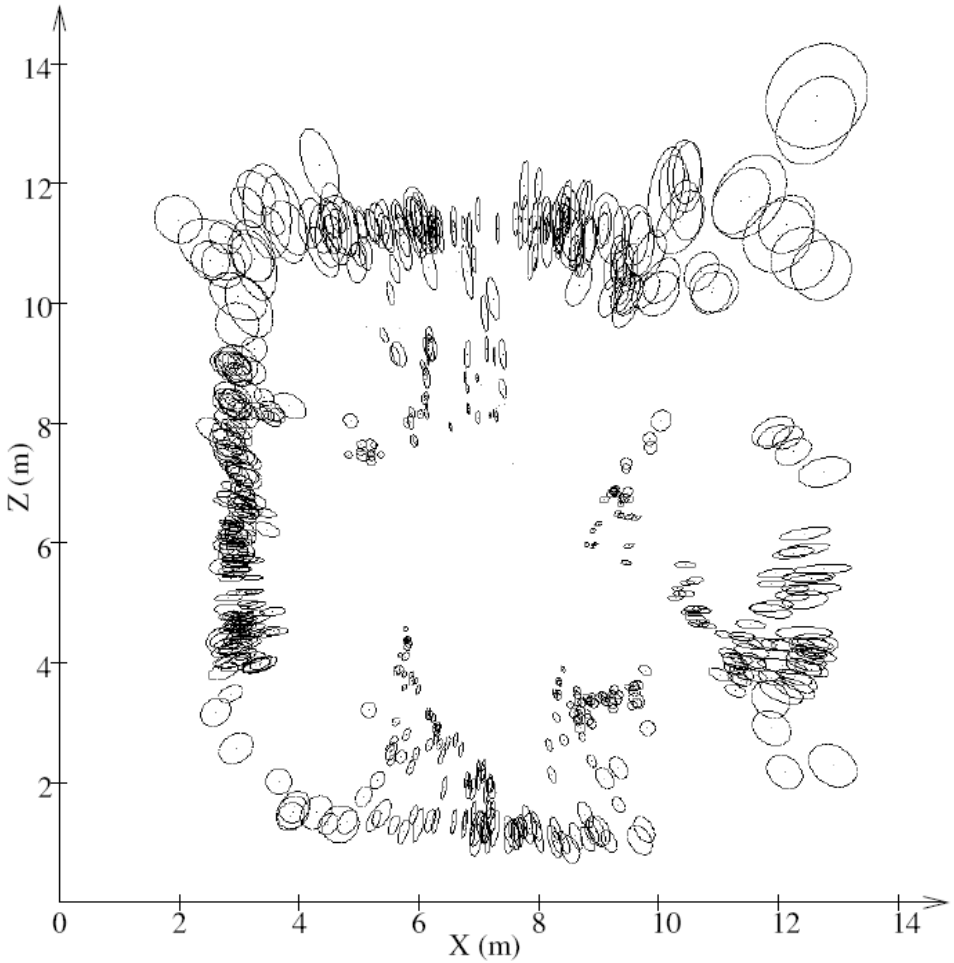
BROWN

Error Modeling: Landmarks

- Each landmark has a covariance matrix
- Gets updated on a frame-by-frame-basis
- Math is in the paper

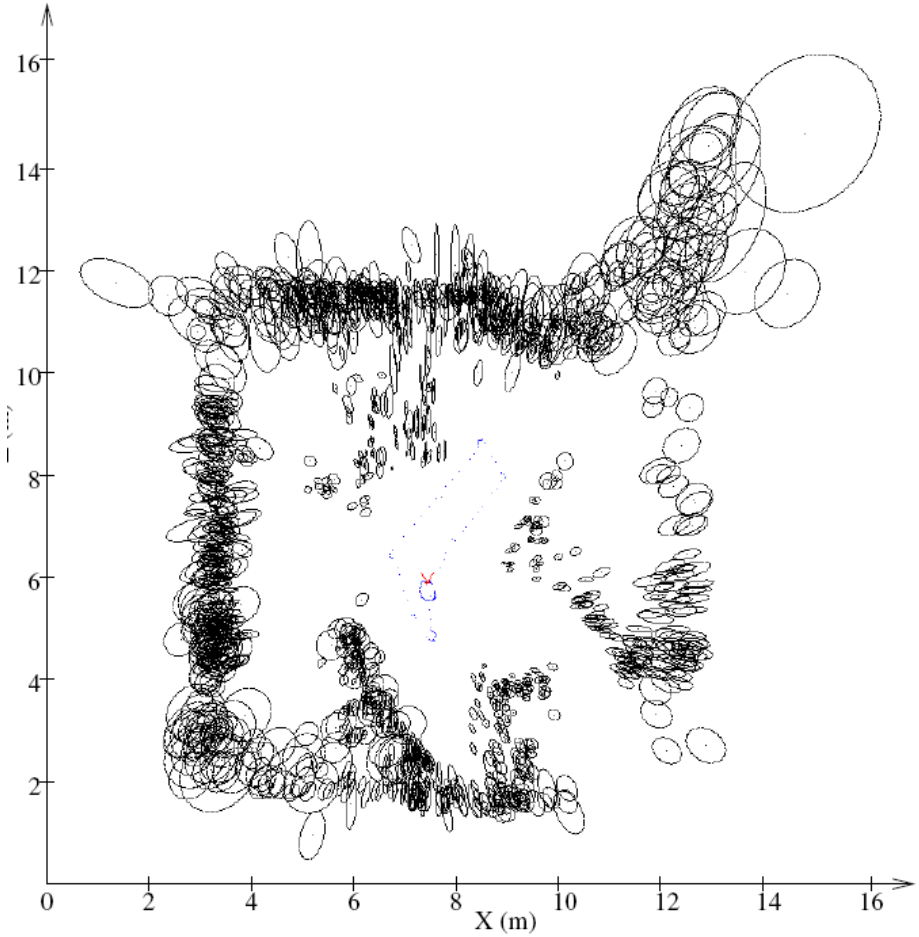
More results

- Just spinning around



More results

- Going in a path around the room



More results

- Uncertainty over robot position