Energy-Based Visual SLAM for Outdoor Robotics

Matthew Grimes
New York University
LAGR (Learning Applied to Ground Robotics)

- 9 teams, 9 flash drives, 1 robot
- Navigate through a novel course in natural terrain.
- Three time trials, with memory retained between trials.
- Goal: beat baseline system's time by a factor of 3.
My part

- Replace this with SLAM

Diagram:

- Cameras
- Bumpers
- IR
- Long Range Vision
- Stereo-based obstacle detector
- Local Navigation
- Vehicle Map
- Drive Commands
- Global Position (error prone)
- Global Map
- Route to goal
- Goal
- Global planner
Localization (the L in SLAM)

- Given a map, determine the robot's location from observations.
- Example use: museum tour guide robot
- Picture: RHINO, from “Experiences with an Interactive Museum Tour–Guide Robot”, Wolfram Burgard et al
Mapping (the M in SLAM)

- Given the robot's location, create a map from observations.
- Example: Generating maps for localization, because blueprints are inaccurate.

• Simultaneous Localization and Mapping

• Given observations, simultaneously generate a map and localize the robot within it.

Picture from slam.ppt at www.probabilistic-robotics.org
Outline

- **SLAM:**
  - Classical Kalman Filter-based approach
  - Energy-based SLAM

- **Visual Odometry:**
  - Current methods
  - Cheaper approaches used in conjunction with SLAM
Classical SLAM (EKF)

- Estimate hidden online state and its covariance using an Extended Kalman Filter.

\[ \bar{x} = [r_x, r_y, r_\theta, m_1^x, m_1^y, \ldots, m_n^x, m_n^y] \]

- Observations = sensor input
  - For example, pixel coordinates of landmarks.

\[ \bar{z} = [z_1^x, z_1^y, \ldots, z_n^x, z_n^y] \]
Classical EKF SLAM

\[ \tilde{x}_t = x_{t-1} \]

\[ \text{Kalman update: } O(n^2) \]

\[ \| \tilde{z}_t - z_t \|_{C_z} \]

\[ \| \tilde{z}_t - z_t \|_{C_z} \]

\[ C_x \]

\[ x_t \]

\[ x_{t-1} \]

\[ \tilde{x}_t \]

\[ z_t \]

\[ \tilde{z}_t \]

\[ \text{state predictor} \]

\[ \text{observation predictor} \]

\[ \text{error function} \]

\[ \text{actual observation} \]

\[ \text{state covariance} \]

\[ \text{Kalman update: } O(n^2) \]
Classical EKF SLAM

From “slam.ppt” on www.probabilistic-robotics.org
Classical EKF SLAM

From “slam.ppt” on www.probabilistic-robotics.org
Classical EKF SLAM

From “slam.ppt” on www.probabilistic-robotics.org
Classical EKF SLAM

- There are more landmarks other sensor inputs than there are poses.
- EKF:
  - $O(n^2)$ in storage.
  - $O(n^2)$ in processing each time we move.
  - Linearizes process and observation models.
Graph SLAM

• An observation is a soft constraint between one robot pose and one landmark
Graph SLAM

- Odometry gives a soft constraint between successive poses.

Odometry covariance
Graph SLAM

• If you retain past robot poses, you can build up a graph.
If you retain past robot poses, you can build up a graph.
Graph SLAM

• If you retain past robot poses, you can build up a graph.
Graph SLAM

- EKF marginalized out past poses, and built a fully connected graph of landmarks.
Graph SLAM

- Formulates SLAM as optimizing an energy function of soft constraints.
- Full nonlinear optimization is slow, offline.
- Use stochastic updates instead.
Energy function

• Energy function: a sum of independent differentiable sub-energy modules.
• For example, the GPS energy:

\[ E_g(x_t, g_t) = \| g_t - x_t \|_{C_g} \]
Energy function

- The odometry energy:

\[ E_g(x_t, x_{t-1}, o_t) = \| \Delta x_t - o_t \|_{C_o} \]
Energy function

- The visual compassing energy:

\[ E_v(\theta_t, \theta_{t-1}, \phi_t) = \| \Delta \theta_t - \phi_t \|_{C_v} \]
Energy function

- The total energy:

\[ E = \sum_{t \in T} E_g + E_o + E_v + \cdots \]

- At each timestep,
  - Select a set \( T \) of poses to optimize.
  - Accumulate energy gradients into them and any landmarks they affect.
  - Gradient descent with diagonal Hessian.
Results: visual compassing

- Show movie
Results: Overcoming wheel slip
Future work

• Landmarks
  – Use Hessian uncertainty windows for efficient landmark search.

• Visual odometry
  – Use ground plane patches for translation information.

• Learning sensor uncertainty
  – Hold poses constant, optimize w.r.t. sensor covariance parameters.