Range-Only SLAM for Robots Operating Cooperatively with Sensor Networks

Authors: Joseph Djugash, Sanjiv Singh, George Kantor and Wei Zhang

Reading Assignment
16-735: Robot Motion Planning
Fall 2007

Presented by
Michael Dille
Umashankar Nagarajan

Introduction to Paper

- A mobile robot operating among beacons (ad hoc sensor network)
  - Robot has beacon too

Pioneer I

Introduction to Paper

- A mobile robot operating among beacons (ad hoc sensor network)
  - Robot has beacon too
- Everything communicates using radio
- Measure range to other beacons using sonar
  - No practical implementations of radio-based ranging available
- The beacons can also move

Extended Kalman Filter (Review)

- Noise-corrupted process: $q_k = f(q_{k-1}, u_k, w_{k-1})$
- Noise-corrupted measurements: $y_k = h(q_k, v_k)$

- Predict (project state forward)
  - $\hat{q}_k^- = f(\hat{q}_{k-1}, u_k, 0)$
  - $P_k^- = A_k P_{k-1} A_k^T + W_k Q_{k-1} W_k^T$
- Correct (incorporate measurements)
  - $K_k = P_k H_k^T (H_k P_k^- H_k^T + V_k R_k^- V_k^T)^{-1}$
  - $\hat{q}_k = \hat{q}_k^{-1} + K_k (y_k - h(\hat{q}_k^-, 0))$
  - $P_k = (I - K_k H_k) P_k^-$

$$A_{[i,j]} = \frac{\partial f_{[i]}}{\partial q_{[j]}}(\hat{q}_{k-1}, u_k, 0)$$

$$W_{[i,j]} = \frac{\partial f_{[i]}}{\partial w_{[j]}}(\hat{q}_{k-1}, u_k, 0)$$

$$H_{[i,j]} = \frac{\partial h_{[i]}}{\partial q_{[j]}}(\hat{q}_k, u_k, 0)$$

$$V_{[i,j]} = \frac{\partial h_{[i]}}{\partial v_{[j]}}(\hat{q}_k, u_k, 0)$$

Introduction to SLAM

- Scenario
  - Landmarks & sensor to detect them
- Localization
  - Landmark locations known
  - Use successive measurements to derive position
- Mapping
  - Robot position known
  - Use successive measurements to determine landmark locations

Introduction to SLAM

- Simultaneous Localization & Mapping
  - Know neither robot nor landmark locations
  - Use successive measurements to derive position of robot *and* landmarks

Range-only SLAM

- In “typical SLAM,” get range & bearing (displacement vector)
- May only get range readings
  - Often, no line of sight
    - Underwater
    - Inside buildings
    - Emergency response: smoky, flame-filled areas
  - Sensors: radio, sonar, …
    - Range from signal strength, time of flight, etc.

Related: Bearing-only SLAM

- Monocular structure from motion in computer vision
  - Each point in camera image corresponds to a ray in space

EKF Localization
(assuming landmark locations known)

- Robot state: q_k
  - Position & orientation
- Motion model: $q_{k+1} = F(q_k) + G(u_k) + v(q_k)$
  - System dynamics, control input, noise
- Measurement model: $y_k = H(q_k) + w(q_k)$
  - Sensing of relative landmark locations, noise
  - In 2-D:

$$H = \begin{bmatrix} \vdots \\ \sqrt{(x_i - x_r)^2 + (y_i - y_r)^2} \\ \operatorname{atan2}(y_i - y_r, x_i - x_r) \\ \vdots \end{bmatrix}$$

Extending EKF for SLAM

- Localize with Kalman filter as before
- Discover landmarks while moving
- Add landmark locations to state estimate
  - In 2-D: $q_k = \begin{bmatrix} x_k & y_k & \theta_k & x_{l1} & y_{l1} & \dots & x_{ln} & y_{ln} \end{bmatrix}^T$
- Need some initial estimate of landmark locations
- Dreaded data association problem
  - Landmark ⇔ measurement?

Paper Contribution

Comparative study of five different algorithms

- Just range-only localization
  - EKF on robot measuring distances to known beacon locations
- Basic range-only SLAM
  - Same EKF, beacon locations unknown
- Extensions
  - Add beacon-to-beacon measurements to EKF
  - Offline optimization of robot path & beacon locations
  - Robot as virtual node, solve submaps with MDS, patch together

Range-only Localization

EKF model used in paper:

Robot state (Position and Orientation): $q_k = \begin{bmatrix} x_k & y_k & \theta_k \end{bmatrix}^T$

Dynamics of the wheeled robot:

$$q_{k+1} = \begin{bmatrix} x_k + \Delta D_k \cos(\theta_k) \\ y_k + \Delta D_k \sin(\theta_k) \\ \theta_k + \Delta\theta_k \end{bmatrix} + \nu_k$$

where ν_k is a noise vector, ΔD_k is the odometric distance traveled $\Delta\theta_k$ is the orientation change

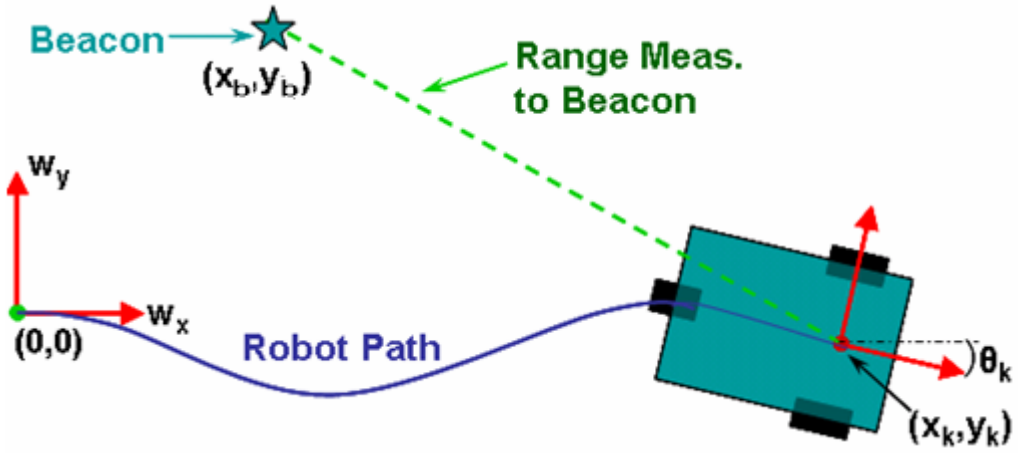
Control input vector: $u_k = \begin{bmatrix} \Delta D_k & \Delta\theta_k \end{bmatrix}^T$

Range-only Localization

EKF measurement model in paper:

Range Measurement: $\hat{r}_k = \sqrt{(x_k - x_b)^2 + (y_k - y_b)^2} + \eta_k$

where $\hat{r}_k$ is the estimate of the range from the beacon to the current state, (x_b, y_b) is the location of the beacon from which the measurement was received, η_k is the zero mean Gaussian noise.



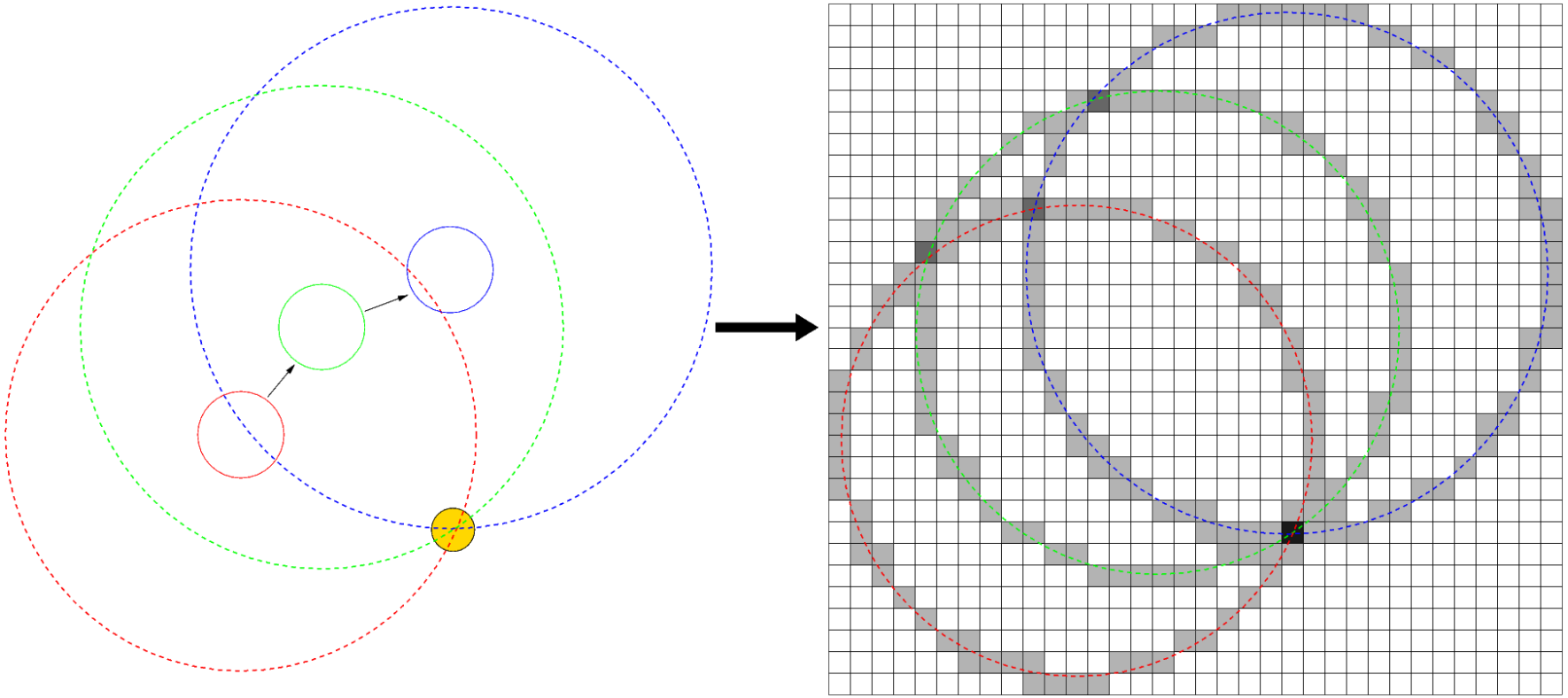
Range-only SLAM

- Same as general SLAM formalism, just no bearings
  - Use same EKF as localization
  - Add beacon locations to state
    - State vector now: $q_k = \begin{bmatrix} x_k & y_k & \theta_k & x_{b1} & y_{b1} & \dots & x_{bn} & y_{bn} \end{bmatrix}^T$
- Still need decent initial estimates!

Range-only SLAM

To find initial beacon location estimates…

Drive around a bit and build 2-D probability grid:



Adding Beacon-to-Beacon Measurements

- Extending SLAM

  Beacon-to-Beacon measurements are used to update the states of the beacons that create the measurement

  The measurement model is modified to include the beacon-to-beacon as well as robot-to-beacon measurements

  The estimation process then proceeds via a standard application of the EKF

Adding Beacon-to-Beacon Measurements

- Improving SLAM Map

  Uses a post-processing step to improve the map that results from a SLAM run

  Finds the state that minimizes the cost function, which is Mahalanobis distance from SLAM map added to the sum of squares of beacon-to-beacon range innovations

$$J(q^M) = (q^M - \hat{q}^M)^T P_M^{-1} (q^M - \hat{q}^M) + \sum_{r_{ij} \in \mathcal{R}} \frac{(r_{ij} - \|q_i^M - q_j^M\|)^2}{\sigma_b^2}$$

Adding Beacon-to-Beacon Measurements

- Improving SLAM Map

Cost Function

$$J(q^M) = (q^M - \hat{q}^M)^T P_M^{-1} (q^M - \hat{q}^M) + \sum_{r_{ij} \in \mathcal{R}} \frac{(r_{ij} - \|q_i^M - q_j^M\|)^2}{\sigma_b^2}$$

Mahalanobis Distance

where,

($\hat{q}$, P) – Kalman Filter SLAM estimate at the end of a run; P is the Kalman covariance matrix for the state $\hat{q}$

($\hat{q}^M$, P_M) – portions of $\hat{q}$ and P that correspond to the beacon locations

q_i^M – part of the state that corresponds to (x,y) location of the ith beacon

r_{ij} – range measurement between beacons i and j

σ_b^2 – covariance of the error in range measurements

Adding Beacon-to-Beacon Measurements

- Self-Localization with Virtual Nodes
  - Adds virtual nodes (different robot locations) as a part of the beacon network
  - Distances between virtual nodes are calculated using robot odometry and they form edges in the network

Self-Localization

- If each beacon can obtain measurements to enough of its neighbors, then it is theoretically possible to build a map using only beacon-to-beacon measurements
- Self-Localization techniques available in literature require cliques (fully interconnected subgraphs) of degree four or higher
- Multidimensional scaling (MDS) is used to determine the relative locations of the clique nodes

Multidimensional Scaling (MDS)

- Set of related statistical techniques often used in data visualization for exploring similarities or dissimilarities in data
- An MDS algorithm starts with a matrix of item-item similarities, then assigns a location of each item in a low-dimensional space, suitable for graphing or 3D visualization
- Applications include scientific visualization and data mining in fields such as cognitive science, information science, psychophysics, psychometrics, marketing and ecology

MDS (of our interest)

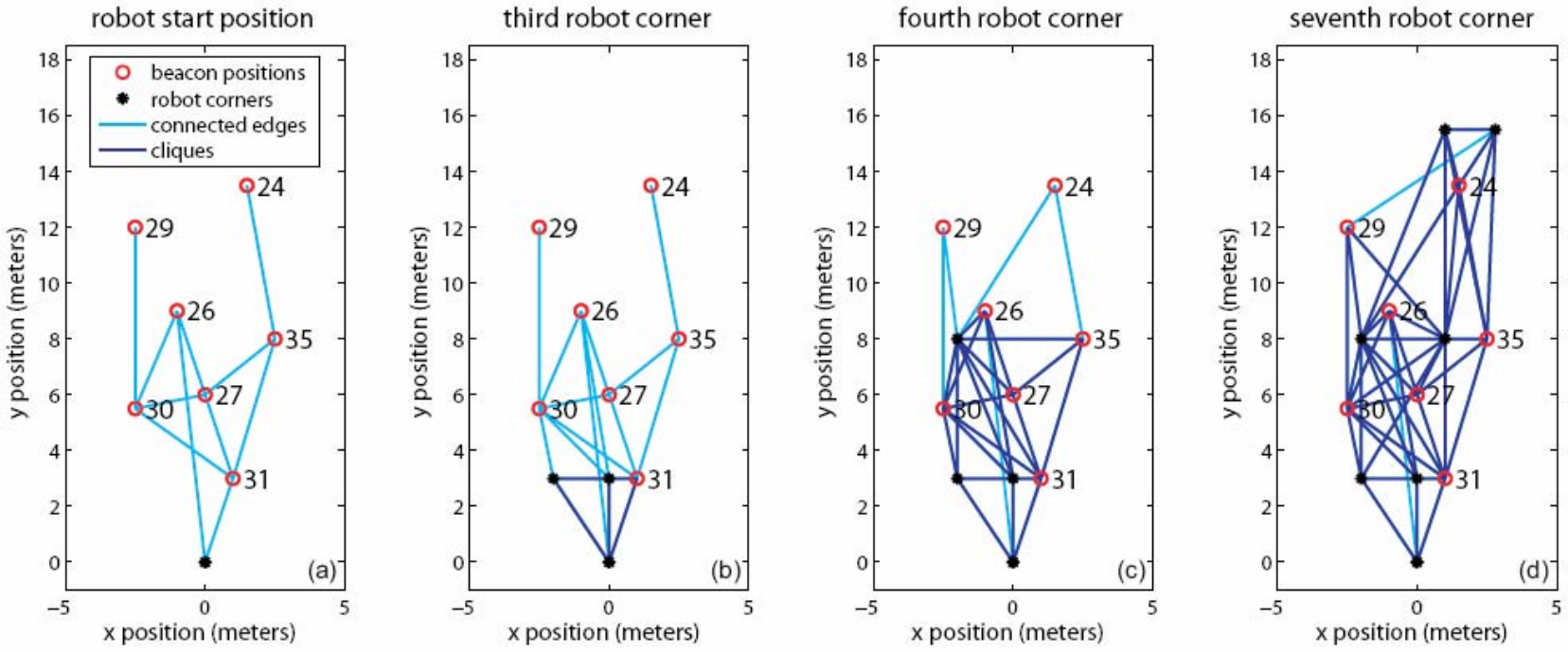
- Given
  - n points in m-D space (usually $n > 3$ for $m = 2$)
  - distances between these
- Returns “best” estimate of locations of points
  - (up to a rigid transformation)
- Even if some distances are unknown, it can still be applied by optimizing over these

Adding Beacon-to-Beacon Measurements

- Self-Localization with Virtual Nodes
  - The robot follows a path and the corners (places where it stops and turns) of the path are taken as virtual nodes
  - First clique appears at the third robot corner
  - At the seventh robot corner, there is enough information in the graph to uniquely localize every beacon

Adding Beacon-to-Beacon Measurements

- Self-Localization with Virtual Nodes



Results and Summary

	LOC.	$SLAM_1$	$SLAM_2$	IMP. MAP	VIR. N.
XTE $\frac{\mu}{\sigma}$	$\frac{0.143}{0.110}$	$\frac{0.189}{0.160}$	$\frac{0.184}{0.148}$	$\frac{0.160}{0.139}$	$\frac{0.189}{0.161}$
RMS	–	0.177	0.171	0.124	0.224

where, (All in meters)

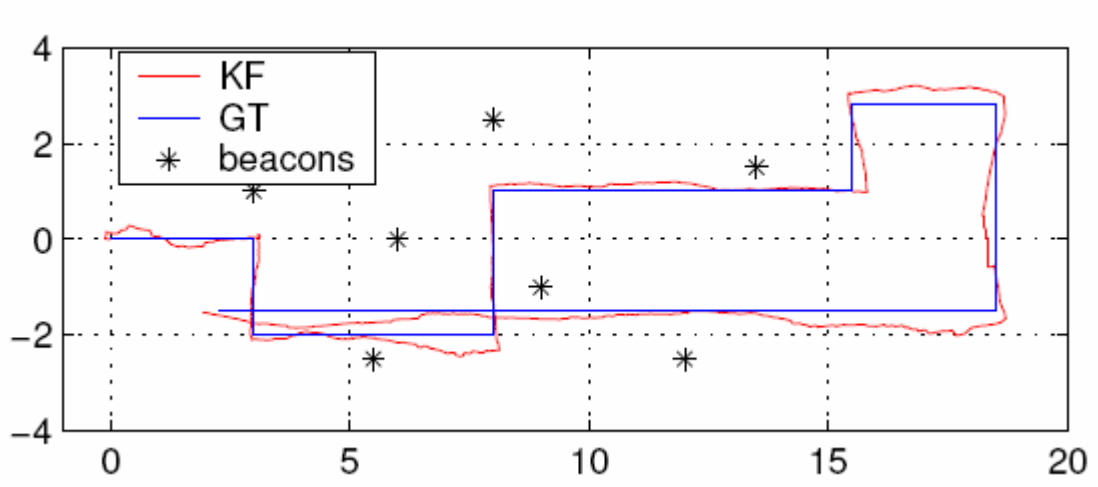
μ : Mean absolute Cross Track Error (XTE)

σ : Standard Deviation

RMS : Root Mean Square error

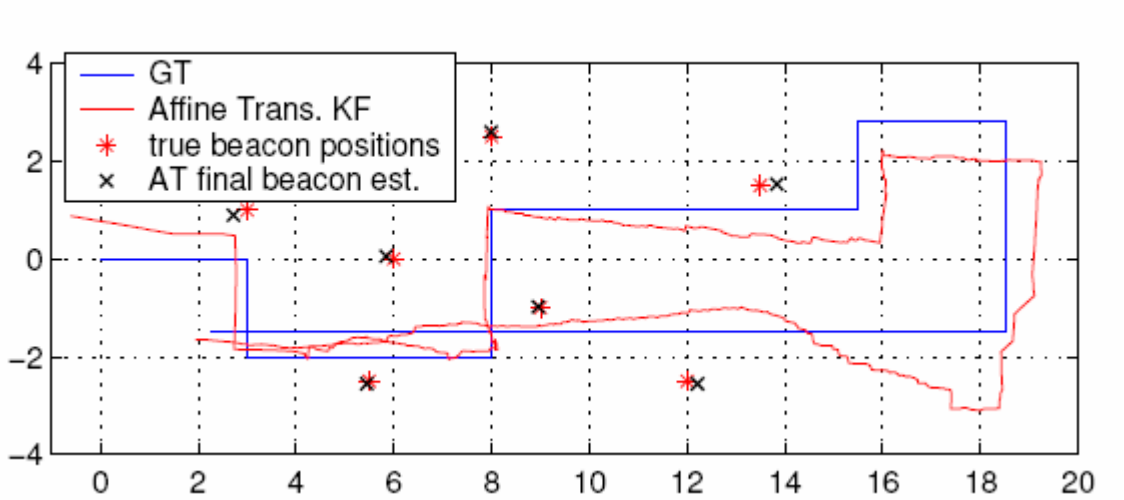
Cross Track Error : Distance off track

Results and Summary



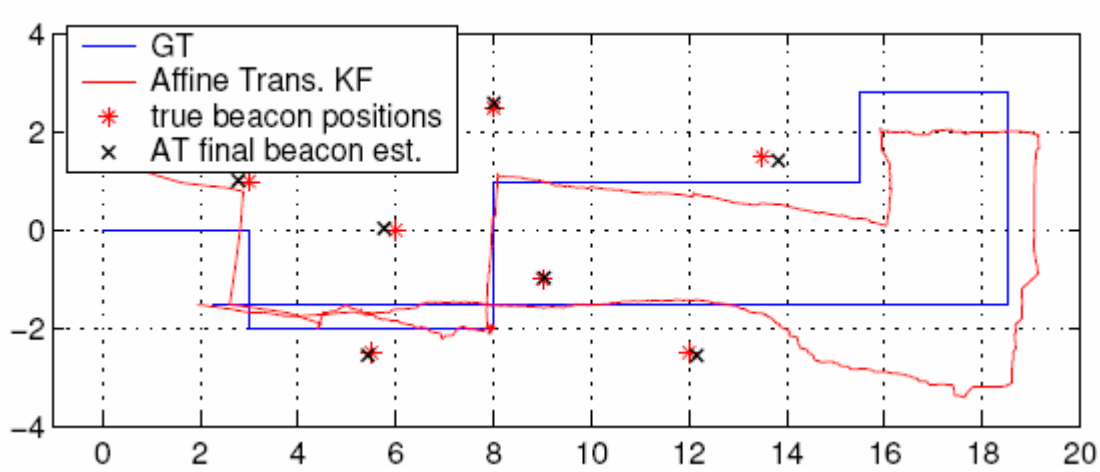
- Localization using Kalman Filter (KF) and the reference ground truth (GT)
- Location of the beacons is known apriori

Results and Summary



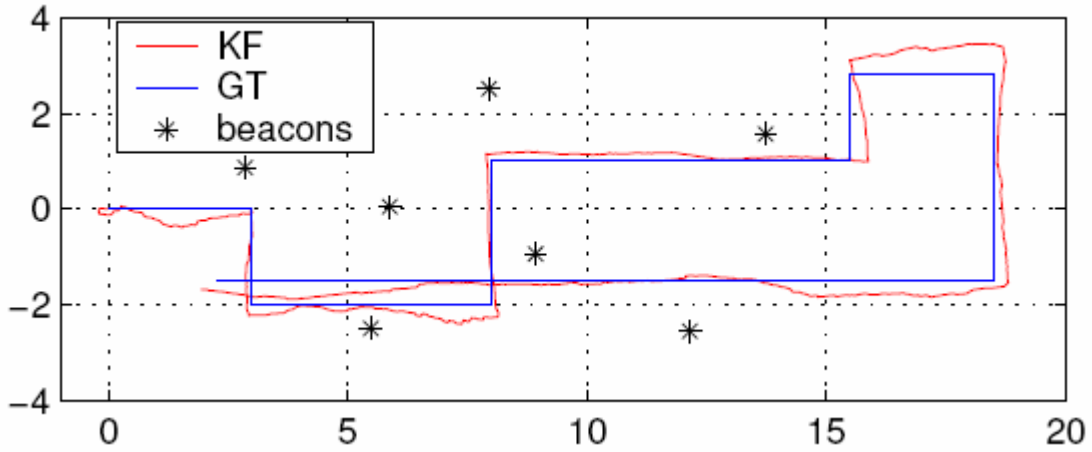
- Kalman Filter (KF) SLAM and the reference ground truth (GT)
- Location of the beacons is unknown at the start of SLAM
- The path and beacon estimates shown include an affine transform that re-aligned the local solution into a global coordinate frame

Results and Summary



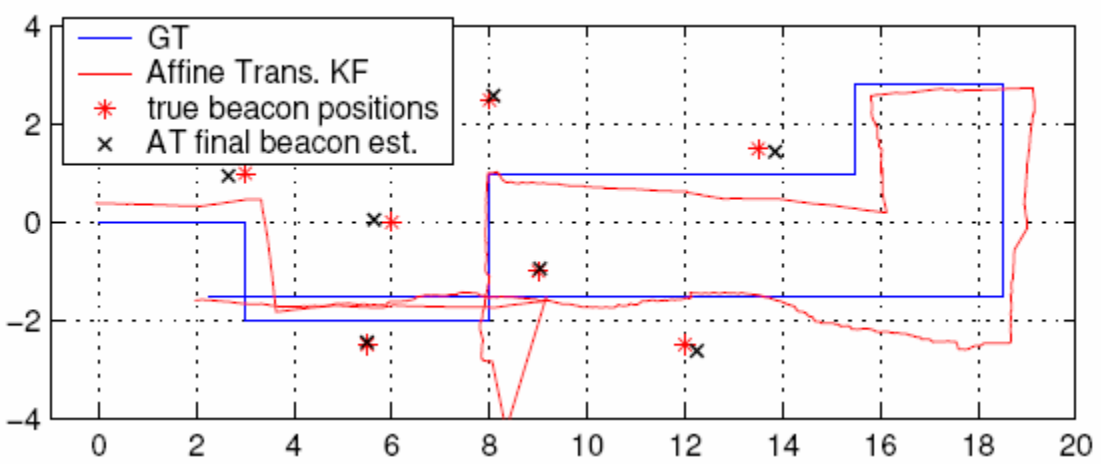
- Kalman Filter (KF) SLAM with inter-beacon measurements and the reference ground truth (GT)
- Location of the beacons is unknown at the start of SLAM
- The path and beacon estimates shown include an affine transform that re-aligned the local solution into a global coordinate frame

Results and Summary



- Localization on a map from Kalman Filter (KF) SLAM after offline optimization step
- The map from Kalman Filter SLAM is used as initial conditions for the optimization process

Results and Summary



- Kalman Filter (KF) SLAM with beacons initialized through self-localization method with virtual nodes
- The path and beacon estimates shown include an affine transform that re-aligned the local solution into a global coordinate frame
- The sudden jumps seen in the Kalman Filter result is an artifact that can be observed when the filter switches its behavior to not only use odometric information but also any range measurements it receives from any initialized beacon

Questions ?