Treemap: Closing a Million-Landmarks Loop

Udo Frese
Simultaneous Localization and Mapping

- continuously estimate a map from sensor data
- input (yellow):
  - landmark observations
  - odometry
- output (blue):
  - landmark positions
  - robot pose
Simultaneous Localization and Mapping
Simultaneous Localization and Mapping

$n$ landmarks
$p$ robot poses
$k$ local
$=O(1)$ landmarks
Simultaneous Localization and Mapping
Simultaneous Localization and Mapping
Simultaneous Localization and Mapping
Simultaneous Localization and Mapping
Simultaneous Localization and Mapping
Simultaneous Localization and Mapping
Simultaneous Localization and Mapping
Simultaneous Localization and Mapping
Simultaneous Localization and Mapping
Simultaneous Localization and Mapping

Udo Frese (16)
Simultaneous Localization and Mapping
Simultaneous Localization and Mapping

• problem: accumulated error
Simultaneous Localization and Mapping

SLAM Uncertainty 1

• accumulated error affects position not shape

„Certainty of Relations despite Uncertainty of Positions“

Simultaneous Localization and Mapping

• closing a loop by re-identifying a landmark
• „bending“ the map
Simultaneous Localization and Mapping

- implicitly done by proper statistical evaluation
Simultaneous Localization and Mapping

- closing the loop: single measurement drastically reduces the overall error
Simultaneous Localization and Mapping

• optimal solution: (nonlinear) least square estimation following C.F. Gauss
• nonlinear maximum likelihood estimation
• linear equation system
• problem: computation time
### Simultaneous Localization and Mapping

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Quality</th>
<th>Storage</th>
<th>Computation time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max. Likel.</td>
<td>optimal</td>
<td>$n+kp$</td>
<td>$(n+p)^3$</td>
</tr>
<tr>
<td>EKF</td>
<td>linear</td>
<td>$n^2$</td>
<td>$n^2$</td>
</tr>
<tr>
<td>CEKF</td>
<td>linear</td>
<td>$n^{3/2}$</td>
<td>$k^2$</td>
</tr>
<tr>
<td>Treemap</td>
<td>nonlin.</td>
<td>$kn$</td>
<td>$k^2$</td>
</tr>
</tbody>
</table>

- $n$: landmarks (725)
- $p$: robot poses (3297)
- $k$: local landmarks (15)

**EKF**
- Linear
- $n^2$ storage
- $n^2$ computation time

**CEKF**
- Linear
- $n^{3/2}$ storage
- $k^2$ computation time

**Treemap**
- Nonlinear
- $kn$ storage
- $k^2$ computation time

- Same region
- New region
- Global
# Simultaneous Localization and Mapping

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Quality</th>
<th>Storage</th>
<th>Computation time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max. Likel.</td>
<td>optimal</td>
<td>(n+kp)</td>
<td>((n+p)^3)</td>
</tr>
<tr>
<td>EKF</td>
<td>linear</td>
<td>(n^2)</td>
<td>(n^2)</td>
</tr>
<tr>
<td>CEKF</td>
<td>linear</td>
<td>(n^{3/2})</td>
<td>(k^2)</td>
</tr>
<tr>
<td>Treemap</td>
<td>nonlin.</td>
<td>(kn)</td>
<td>(k^3 \log n)</td>
</tr>
</tbody>
</table>

- \(n\) landmarks (725)
- \(p\) robot poses (3297)
- \(k\) local landmarks (15)
### Simultaneous Localization and Mapping

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Quality</th>
<th>Storage</th>
<th>Computation time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max. Likel.</td>
<td>optimal</td>
<td>$n+kp$</td>
<td>$(n+p)^3$</td>
</tr>
<tr>
<td>EKF</td>
<td>linear</td>
<td>$n^2$</td>
<td>$n^2$</td>
</tr>
<tr>
<td>CEKF</td>
<td>linear</td>
<td>$n^{3/2}$</td>
<td>$k^2$</td>
</tr>
<tr>
<td>Treemap</td>
<td>nonlin.</td>
<td>$kn$</td>
<td>$k^2$</td>
</tr>
</tbody>
</table>

- **n**: landmarks (725)
- **p**: robot poses (3297)
- **k**: local landmarks (15)
The Hierarchical Treemap Algorithm

The Hierarchical Treemap Algorithm

- General idea
- Probabilistic propagation along the tree
- Linearization, integration, marginalization, sparsification
- Bookkeeping and hierarchical tree partitioning
- Closing a million-landmarks loop
Treemap Algorithm

- If the robot is in part A, what is the information needed about B?
- Only the marginal distribution of landmarks observable from A conditioned on observations in B.
Treemap Algorithm
Udo Frese (32)

Treemap Algorithm

landmarks
landmark-observations
odometry
robot poses
Treemap Algorithm

landmarks

landmark-observations

odometry

robot poses
Treemap Algorithm

landmarks

landmark-observations

odometry

robot poses
Treemap Algorithm

\[
p(X[n: \downarrow \uparrow] | z[n: \downarrow])
\]

\[
p(X[n: \uparrow \downarrow \uparrow] | z[n: \downarrow])
\]

\[
p(X[n: \uparrow \downarrow \uparrow] | z[n: \downarrow]) =
\]

\[
p(X[n: \downarrow \uparrow \downarrow] | z[n: \downarrow])
\]

\[
p(X[n: \downarrow \uparrow \downarrow] | z[n: \downarrow]) =
\]

Udo Frese (43)
\[ p(X[n_L: \downarrow \uparrow]|z[n_L: \downarrow]) = p(X[n_V: \downarrow \uparrow]|z[n_V: \downarrow]) = p(X[n: \uparrow \vee \downarrow \uparrow]|z[n: \uparrow]) = p(X[n: \downarrow \uparrow \vee \downarrow \uparrow]|z[n: \downarrow]) } \]
\[ p(X[n: \downarrow \uparrow \vee \downarrow \uparrow] \mid z[n \downarrow]) \]
\[ p(X[n: \downarrow \uparrow \vee \wedge \uparrow \downarrow] | z[n \downarrow]) \]
\[ p(X[n: \land \uparrow] | X[n: \downarrow \uparrow], z) \quad p(X[n: \downarrow \uparrow] | z[n: \downarrow]) \]
Treemap Algorithm

\[ p(X[n: \downarrow \uparrow] | z[n: \downarrow]) \]

\[ p(X[n: \downarrow \uparrow \lor \downarrow \uparrow] | z[n: \downarrow]) \]

\[ p(X[n: \downarrow \uparrow] | z[n: \downarrow], z) \]

\[ p(X[n: \downarrow \uparrow \lor \downarrow \uparrow] | z[n: \downarrow], z) \]

\[ p(X[n\downarrow: \downarrow \uparrow] | z[n\downarrow: \downarrow]) = \]
\[ p(X[n\downarrow: \uparrow \lor \downarrow \uparrow] | z[n\downarrow: \downarrow]) = \]
\[ p(X[n: \uparrow \lor \downarrow \uparrow] | z[n: \downarrow]) \]
Treemap Algorithm

\[ p(X[n: \downarrow \uparrow]\mid z) \]

\[ p(X[n: \downarrow \uparrow \downarrow \uparrow]\mid X[n: \downarrow \uparrow], z) \]

\[ p(X[n: \downarrow \uparrow \vee \downarrow \uparrow]\mid z) \]
Treemap Algorithm

Actual Implementation

- Gaussians defined by square-root information matrix.
- Upwards (●) by stacking.
- (M) by QR-decomposition
- Downwards (●) by back-substitution, i.e. solving a triangular equation system
\[ \chi^2(x) = x^T A x + x^T b + \gamma \]
\[ = (\begin{pmatrix} x \\ 1 \end{pmatrix})^T \begin{pmatrix} A & b/2 \\ b^T/2 & \gamma \end{pmatrix} \begin{pmatrix} x \\ 1 \end{pmatrix} \]
\[ = \|R x'\|_2^2, \quad A' = R^T R \]

\[ \chi^2(x') = \chi_1^2(x') + \chi_2^2(x') \]
\[ = \|R_1 x'\|_2^2 + \|R_2 x'\|_2^2 \]
\[ = \|(R_1 \begin{pmatrix} R_1 & R_2 \end{pmatrix} x')\|_2^2 \]
\[ = \|R x'\|_2^2, \quad \begin{pmatrix} R_1 \\ R_2 \end{pmatrix} = QR \]
\[ \chi^2 \left( \begin{pmatrix} y' \\ z' \end{pmatrix} \right) = \left\| \begin{pmatrix} C & D \\ 0 & E \end{pmatrix} \begin{pmatrix} y' \\ z' \end{pmatrix} \right\|_2^2 \]
\[ = \left\| \begin{pmatrix} C & D \\ 0 & E \end{pmatrix} \begin{pmatrix} y' \\ z' \end{pmatrix} \right\|_2^2 + \left\| \begin{pmatrix} 0 & E \end{pmatrix} \begin{pmatrix} y' \\ z' \end{pmatrix} \right\|_2^2 \]
\[ = \left\| \begin{pmatrix} C & D \\ 0 & E \end{pmatrix} \begin{pmatrix} y' \\ z' \end{pmatrix} \right\|_2^2 + \left\| Ez' \right\|_2^2 \]
\[ \chi^2 \left( \begin{pmatrix} y' \\ z' \end{pmatrix} \right) = \left\| C(y - C^{-1}Dz') \right\|_2^2 + \left\| Ez' \right\|_2^2 \]

\[ y_i = -\frac{1}{R_{ii}} \sum_{j=i+1}^{\dim y} R_{ij} y_j \]
Treemap Algorithm

Why is it fast?

- Many small matrices instead of one large matrix.
- Update only $O(\log n)$ nodes upwards.
- Downwards (●) operation is extremely fast.
- Requires topologically suitable building.
Treemap Algorithm

A „topologically suitable“ building
Experiments
Experiments

SLAM Video (uncut)

SLAM Video (abridged)
Experiments

Udo Frese (59)
Experiments

building: 60m × 45m
rooms: 29
distance traveled: 505m
large loops: 3
landmarks: $n = 725$
measurements: $m = 29142$
robot poses: $p = 3297$
local landmarks: $k \approx 16$
Experiments

Navigation Video
Linearization, Integration, Marginalization and Sparsification
Different Levels of Approximation

- keep all non-linear measurements
  - recompute Jacobians every time you need.
- linearize
  - integrate a whole region into one matrix
- marginalize
  - marginalize out old poses inside a region
- sparsify
  - duplicate some old poses and marginalize out
  - cutting odometry (like ESDS-Filter)
Different Levels of Approximation

- keep all non-linear measurements
  - recompute Jacobians every time you need.

- linearize
  - integrate a whole region into one matrix

- marginalize
  - marginalize out old poses

- sparsify
  - duplicate some old poses and marginalize out
  - cutting odometry (like ESDS-Filter)
Closing a Million-Landmarks Loop

landmarks

landmark-observations

odometry

robot poses
Closing a Million-Landmarks Loop

A: Nonlinear distributions
Closing a Million-Landmarks Loop

B: Linearize
Closing a Million-Landmarks Loop

C: Marginalize out inner poses

Udo Frese (69)
Closing a Million-Landmarks Loop

D:
Sparsify,
1: sacrifice
pose
equality
constraint
Closing a Million-Landmarks Loop

D:
Sparsify,
1: sacrifice pose equality constraint
2: marginalize out all poses
Bookkeeping and Hierarchical Tree Partitioning
Bookkeeping and HTP

• Which nodes to recompute?
• Rearrange the tree to improve computation time.
• NP-hard
• Multilevel Khernighan and Lin heuristics established in the field of graph partitioning
• Do some Khernighan and Lin runs after each update
• Optimize worst-case update time
Bookkeeping and HTP

- Choose a node $r$ from a queue
- Consider moving a single subtree $s$ from one side of $r$ to the other
Bookkeeping and HTP

- Choose a node \( r \) from a queue
- Consider moving a single subtree from one side of \( r \) to the other
Bookkeeping and HTP

- Try to move every subtree that shares a feature (KL) on the left of s to move to the right of s and vice versa (O(k log² n))
- Choose the best
- Try it for some steps even if it makes things worst (KL)
- Consider integration, marginalization when moving
- Consider sparsification as a last resort
Closing a Million-Landmarks Loop

Our homage our response
Application → Treemap Driver → Treemap Backend

- Observations
- Control policy
- Map estimate
- Gaussian mean
- Gaussians
The Experiments

Video: Closing a Million-Landmarks Loop
(http://www.informatik.uni-bremen.de/~ufrese/slammillionlandmarks/freemillionlandmarks.avi)

Video: Using Treemap for a Generic Least Square Backend for 6-DOF SLAM
(http://www.informatik.uni-bremen.de/~ufrese/slammillionlandmarks/avi)
Landmarks [1M]
time [ms]
global downward estimation
upward update
book-keeping
downward estimation
book-keeping
• 1000 Landmarks
upward update

[Graph showing time in milliseconds against a scale with values 0 to 250, with curves indicating upward update, book-keeping, and downward estimation.]
Treemap
- closes a loop over 1032271 features in 21ms (local) or 442ms (global)
- O($k^3 \log n + k^2 \log^2 n + kn$)
- generic backend & specific driver
- open source soon
- driver has to implement
  - measurement function, initial estimate, Jacobian
  - approximation policy
  - 2-D, 3-DOF: 690 lines of C++ code
  - 3-D, 6-DOF: 410 lines of C++ code