Overview of RGB-D SLAM Approaches

Seminar Computer Vision & Visual Tracking for Robotic Applications

Tobias Hollarek

Technische Universität München

June 5, 2012

Outline

Outline

Introduction

- Simultaneous Localization and Mapping
  - Localization: knowing your environment, calculate your position
  - Mapping: building a map of your environment
- SLAM using RGB-D data
  - traditional approaches:
    - SLAM with RGB data only
    - SLAM using laser scanners
  - new development: Kinect style cameras
    - => Cheap acquisition of RGB-D data

- => RGB-D SLAM

Introduction

- Simultaneous Localization and Mapping
  - Localization: knowing your environment, calculate your position
  - Mapping: building a map of your environment
- SLAM using RGB-D data
  - traditional approaches:
    - SLAM with RGB data only
    - SLAM using laser scanners
  - new development: Kinect style cameras
    - => Cheap acquisition of RGB-D data

- => RGB-D SLAM

Introduction

- Simultaneous Localization and Mapping
  - Localization: knowing your environment, calculate your position
  - Mapping: building a map of your environment
- SLAM using RGB-D data
  - traditional approaches:
    - SLAM with RGB data only
    - SLAM using laser scanners
  - new development: Kinect style cameras
    - => Cheap acquisition of RGB-D data

- => RGB-D SLAM

Introduction

- Simultaneous Localization and Mapping
  - Localization: knowing your environment, calculate your position
  - Mapping: building a map of your environment
- SLAM using RGB-D data
  - traditional approaches:
    1. SLAM with RGB data only
    2. SLAM using laser scanners
  - new development: Kinect style cameras

    => cheap acquisition of RGB-D data

=> RGB-D SLAM

Introduction

- Simultaneous Localization and Mapping
  - Localization: knowing your environment, calculate your position
  - Mapping: building a map of your environment
- SLAM using RGB-D data
  - traditional approaches:
    1. SLAM with RGB data only
    2. SLAM using laser scanners
  - new development: Kinect style cameras

    => cheap acquisition of RGB-D data

=> RGB-D SLAM

Introduction

- Simultaneous Localization and Mapping
  - Localization: knowing your environment, calculate your position
  - Mapping: building a map of your environment
- SLAM using RGB-D data
  - traditional approaches:
    1. SLAM with RGB data only
    2. SLAM using laser scanners
  - new development: Kinect style cameras

    => cheap acquisition of RGB-D data

=> RGB-D SLAM

Introduction

- Simultaneous Localization and Mapping
  - Localization: knowing your environment, calculate your position
  - Mapping: building a map of your environment
- SLAM using RGB-D data
  - traditional approaches:
    1. SLAM with RGB data only
    2. SLAM using laser scanners
  - new development: Kinect style cameras

    => cheap acquisition of RGB-D data

=> RGB-D SLAM

Introduction

- Simultaneous Localization and Mapping
  - Localization: knowing your environment, calculate your position
  - Mapping: building a map of your environment
- SLAM using RGB-D data
  - traditional approaches:
    1. SLAM with RGB data only
    2. SLAM using laser scanners
  - new development: Kinect style cameras

=> cheap acquisition of RGB-D data

=> RGB-D SLAM

Introduction

- Simultaneous Localization and Mapping
  - Localization: knowing your environment, calculate your position
  - Mapping: building a map of your environment
- SLAM using RGB-D data
  - traditional approaches:
    1. SLAM with RGB data only
    2. SLAM using laser scanners
  - new development: Kinect style cameras

    => cheap acquisition of RGB-D data

=> RGB-D SLAM

Introduction

- Simultaneous Localization and Mapping
  - Localization: knowing your environment, calculate your position
  - Mapping: building a map of your environment
- SLAM using RGB-D data
  - traditional approaches:
    1. SLAM with RGB data only
    2. SLAM using laser scanners
  - new development: Kinect style cameras

    => cheap acquisition of RGB-D data

=> RGB-D SLAM

Outline

The Approaches

- RGB-D Mapping using RGB-D ICP (Henry et al.)

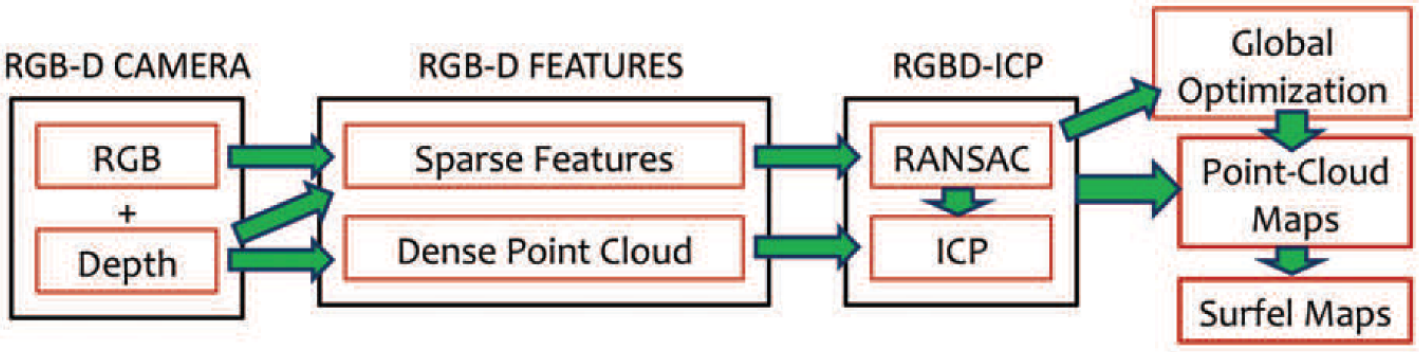

Figure: Henry et al.,*RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments* in: **International Journal of Robotics Research**, 2012

The Approaches

- RGB-D SLAM System (Endres et al.)

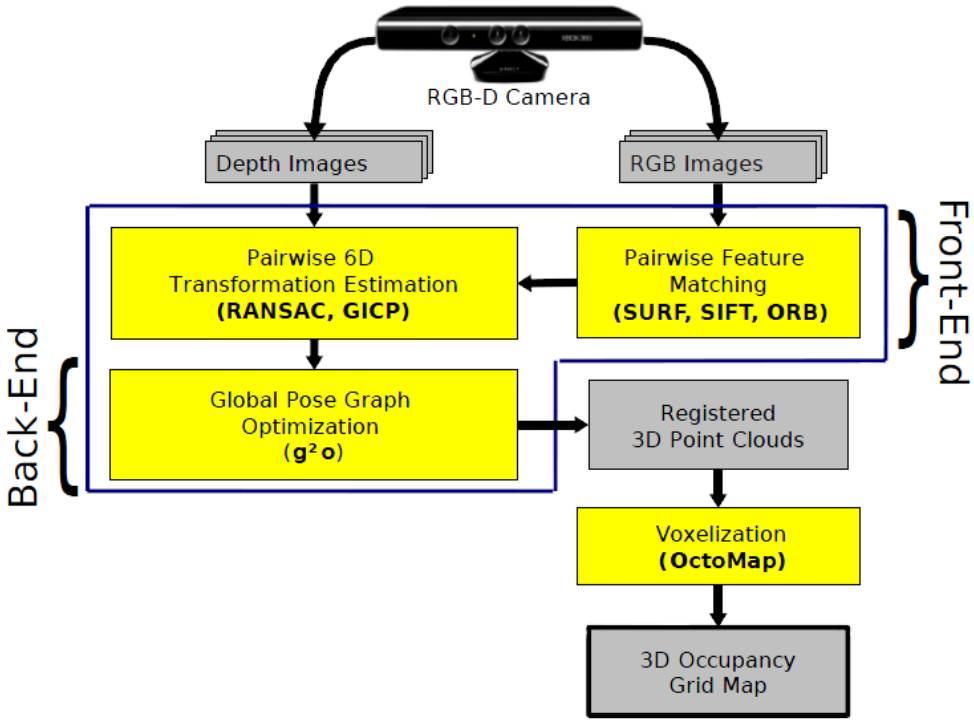

Figure: Endres et al.,*An Evaluation of the RGB-D SLAM System* in **Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)**, 2012

The Approaches

- Visual Odometry (Audras et al.)

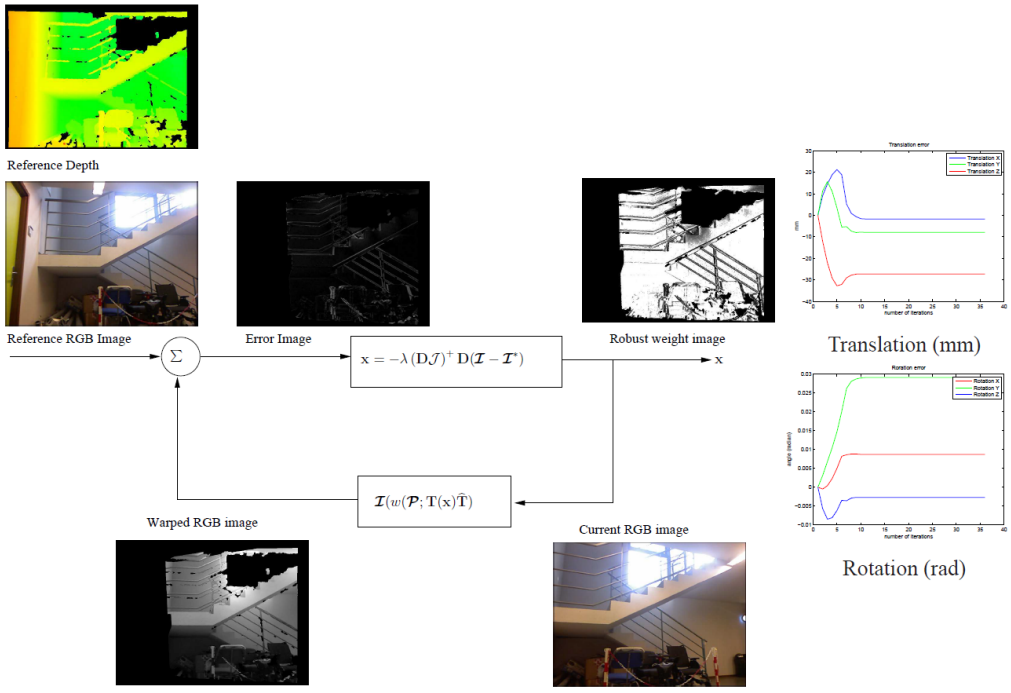

Figure: Audras et al.,*Real-time dense appearance-based SLAM for RGB-D sensors* in **Australian Conference on Robotics and Automation**, 2011

Outline

RGB-D ICP (Henry et al.)

Input: source RGB-D frame F_s, target RGB-D frame F_t
Output: optimized relative Transformation T

1. extract feature points from F_s and F_t

RGB-D ICP (Henry et al.)

- extract feature points from F_s and F_t

Figure: Henry et al.,*RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments* in: **International Journal of Robotics Research**, 2012

RGB-D ICP (Henry et al.)

Input: source RGB-D frame F_s, target RGB-D frame F_t
Output: optimized relative Transformation T

1. extract feature points from F_s and F_t
2. perform RANSAC alignment => first approximation T'
   - RANSAC = Random Sample Consensus
   - randomly choose three pairs of feature points
   - calculate a transformation from theses points
   - check errors from other feature points
   - repeat for other triplets
   - return Transformation with most inliers

RGB-D ICP (Henry et al.)

Input: source RGB-D frame F_s, target RGB-D frame F_t
Output: optimized relative Transformation T

1. extract feature points from F_s and F_t
2. perform RANSAC alignment => first approximation T'
   - RANSAC = Random Sample Consensus
   - randomly choose three pairs of feature points
   - calculate a transformation from theses points
   - check errors from other feature points
   - repeat for other triplets
   - return Transformation with most inliers

RGB-D ICP (Henry et al.)

Input: source RGB-D frame F_s, target RGB-D frame F_t
Output: optimized relative Transformation T

1. extract feature points from F_s and F_t
2. perform RANSAC alignment => first approximation T'
   - RANSAC = Random Sample Consensus
   - randomly choose three pairs of feature points
   - calculate a transformation from theses points
   - check errors from other feature points
   - repeat for other triplets
   - return Transformation with most inliers

RGB-D ICP (Henry et al.)

Input: source RGB-D frame F_s, target RGB-D frame F_t
Output: optimized relative Transformation T

1. extract feature points from F_s and F_t
2. perform RANSAC alignment => first approximation T'
   - RANSAC = Random Sample Consensus
   - randomly choose three pairs of feature points
   - calculate a transformation from theses points
   - check errors from other feature points
   - repeat for other triplets
   - return Transformation with most inliers

RGB-D ICP (Henry et al.)

Input: source RGB-D frame F_s, target RGB-D frame F_t
Output: optimized relative Transformation T

1. extract feature points from F_s and F_t
2. perform RANSAC alignment => first approximation T'
   - RANSAC = Random Sample Consensus
   - randomly choose three pairs of feature points
   - calculate a transformation from theses points
   - check errors from other feature points
   - repeat for other triplets
   - return Transformation with most inliers

RGB-D ICP (Henry et al.)

Input: source RGB-D frame F_s, target RGB-D frame F_t
Output: optimized relative Transformation T

1. extract feature points from F_s and F_t
2. perform RANSAC alignment => first approximation T'
   - RANSAC = Random Sample Consensus
   - randomly choose three pairs of feature points
   - calculate a transformation from theses points
   - check errors from other feature points
   - repeat for other triplets
   - return Transformation with most inliers

RGB-D ICP (Henry et al.)

Input: source RGB-D frame F_s, target RGB-D frame F_t
Output: optimized relative Transformation T

1. extract feature points from F_s and F_t
2. perform RANSAC alignment => first approximation T'
   - RANSAC = Random Sample Consensus
   - randomly choose three pairs of feature points
   - calculate a transformation from theses points
   - check errors from other feature points
   - repeat for other triplets
   - return Transformation with most inliers

RGB-D ICP (Henry et al.)

Input: source RGB-D frame F_s, target RGB-D frame F_t
Output: optimized relative Transformation T

1. extract feature points from F_s and F_t
2. perform RANSAC alignment => first approximation T'
   - RANSAC = Random Sample Consensus
   - randomly choose three pairs of feature points
   - calculate a transformation from theses points
   - check errors from other feature points
   - repeat for other triplets
   - return Transformation with most inliers

RGB-D ICP (Henry et al.)

Input: source RGB-D frame F_s, target RGB-D frame F_t
Output: optimized relative Transformation T

1. extract feature points from F_s and F_t
2. perform RANSAC alignment => first approximation T'
3. if $inliers < k_{low}$: discard T'
4. if $inliers > k_{high}$: return T' as final transformation
5. else: compute T' from ICP

RGB-D ICP (Henry et al.)

Input: source RGB-D frame F_s, target RGB-D frame F_t
Output: optimized relative Transformation T

1. extract feature points from F_s and F_t
2. perform RANSAC alignment => first approximation T'
3. if $inliers < k_{low}$: discard T'
4. if $inliers > k_{high}$: return T' as final transformation
5. else: compute T' from ICP

RGB-D ICP (Henry et al.)

Input: source RGB-D frame F_s, target RGB-D frame F_t
Output: optimized relative Transformation T

1. extract feature points from F_s and F_t
2. perform RANSAC alignment => first approximation T'
3. if $inliers < k_{low}$: discard T'
4. if $inliers > k_{high}$: return T' as final transformation
5. else: compute T' from ICP

RGB-D ICP (Henry et al.)

Input: source RGB-D frame F_s, target RGB-D frame F_t
Output: optimized relative Transformation T

1. extract feature points from F_s and F_t
2. perform RANSAC alignment => first approximation T'
3. if $inliers < k_{low}$: discard T'
4. if $inliers > k_{high}$: return T' as final transformation
5. else: compute T' from ICP

RGB-D ICP (Henry et al.)

Input: source RGB-D frame F_s, target RGB-D frame F_t
Output: optimized relative Transformation T

1. extract feature points from F_s and F_t
2. perform RANSAC alignment => first approximation T'
3. if $inliers < k_{low}$: discard T'
4. if $inliers > k_{high}$: return T' as final transformation
5. else: compute T' from ICP
   - ICP = Iterative Closest Points
   - match features of F_s and F_t
   - sparse features AND dense depth data used
   - improve T' by minimizing matching error
   - repeat matching and minimizing until $change(T') < \gamma$ or $iterations > n_{max}$

RGB-D ICP (Henry et al.)

Input: source RGB-D frame F_s, target RGB-D frame F_t
Output: optimized relative Transformation T

1. extract feature points from F_s and F_t
2. perform RANSAC alignment => first approximation T'
3. if $inliers < k_{low}$: discard T'
4. if $inliers > k_{high}$: return T' as final transformation
5. else: compute T' from ICP
   - ICP = Iterative Closest Points
   - match features of F_s and F_t
   - sparse features AND dense depth data used
   - improve T' by minimizing matching error
   - repeat matching and minimizing until $change(T') < \gamma$ or $iterations > n_{max}$

RGB-D ICP (Henry et al.)

Input: source RGB-D frame F_s, target RGB-D frame F_t
Output: optimized relative Transformation T

1. extract feature points from F_s and F_t
2. perform RANSAC alignment => first approximation T'
3. if $inliers < k_{low}$: discard T'
4. if $inliers > k_{high}$: return T' as final transformation
5. else: compute T' from ICP
   - ICP = Iterative Closest Points
   - match features of F_s and F_t
   - sparse features AND dense depth data used
   - improve T' by minimizing matching error
   - repeat matching and minimizing until $change(T') < \gamma$ or $iterations > n_{max}$

RGB-D ICP (Henry et al.)

Input: source RGB-D frame F_s, target RGB-D frame F_t
Output: optimized relative Transformation T

1. extract feature points from F_s and F_t
2. perform RANSAC alignment => first approximation T'
3. if $inliers < k_{low}$: discard T'
4. if $inliers > k_{high}$: return T' as final transformation
5. else: compute T' from ICP
   - ICP = Iterative Closest Points
   - match features of F_s and F_t
   - sparse features AND dense depth data used
   - improve T' by minimizing matching error
   - repeat matching and minimizing until $change(T') < \gamma$ or $iterations > n_{max}$

RGB-D ICP (Henry et al.)

Input: source RGB-D frame F_s, target RGB-D frame F_t
Output: optimized relative Transformation T

1. extract feature points from F_s and F_t
2. perform RANSAC alignment => first approximation T'
3. if $inliers < k_{low}$: discard T'
4. if $inliers > k_{high}$: return T' as final transformation
5. else: compute T' from ICP
   - ICP = Iterative Closest Points
   - match features of F_s and F_t
   - sparse features AND dense depth data used
   - improve T' by minimizing matching error
   - repeat matching and minimizing until $change(T') < \gamma$ or $iterations > n_{max}$

RGB-D ICP (Henry et al.)

Input: source RGB-D frame F_s, target RGB-D frame F_t
Output: optimized relative Transformation T

1. extract feature points from F_s and F_t
2. perform RANSAC alignment => first approximation T'
3. if $inliers < k_{low}$: discard T'
4. if $inliers > k_{high}$: return T' as final transformation
5. else: compute T' from ICP
   - ICP = Iterative Closest Points
   - match features of F_s and F_t
   - sparse features AND dense depth data used
   - improve T' by minimizing matching error
   - repeat matching and minimizing until $change(T') < \gamma$ or $iterations > n_{max}$

RGB-D ICP (Henry et al.)

Input: source RGB-D frame F_s, target RGB-D frame F_t
Output: optimized relative Transformation T

1. extract feature points from F_s and F_t
2. perform RANSAC alignment => first approximation T'
3. if $inliers < k_{low}$: discard T'
4. if $inliers > k_{high}$: return T' as final transformation
5. else: compute T' from ICP
   until $change(T') < \gamma$ or $iterations > n_{max}$

Front-End (Endres et al.)

- similar to RGB-D ICP
- RANSAC + ICP to perform alignment
- alignment with up to 20 frames

Front-End (Endres et al.)

- similar to RGB-D ICP
- RANSAC + ICP to perform alignment
- alignment with up to 20 frames

Front-End (Endres et al.)

- similar to RGB-D ICP
- RANSAC + ICP to perform alignment
- alignment with up to 20 frames

- only intensity data is used
- no feature extraction
- pixels with maximal gradient along one direction are chosen
- minimize jacobian of error function

- only intensity data is used
- no feature extraction
- pixels with maximal gradient along one direction are chosen
- minimize jacobian of error function

- only intensity data is used
- no feature extraction
- pixels with maximal gradient along one direction are chosen
- minimize jacobian of error function

- only intensity data is used
- no feature extraction
- pixels with maximal gradient along one direction are chosen
- minimize jacobian of error function

Outline

Global Optimization

- imperfect alignment => accumulated error (drift)
- goal: minimize drift
- different approaches possible
- here: loop closure detection
  1. detect if same location is visited for the second time
  2. use information to optimize map:

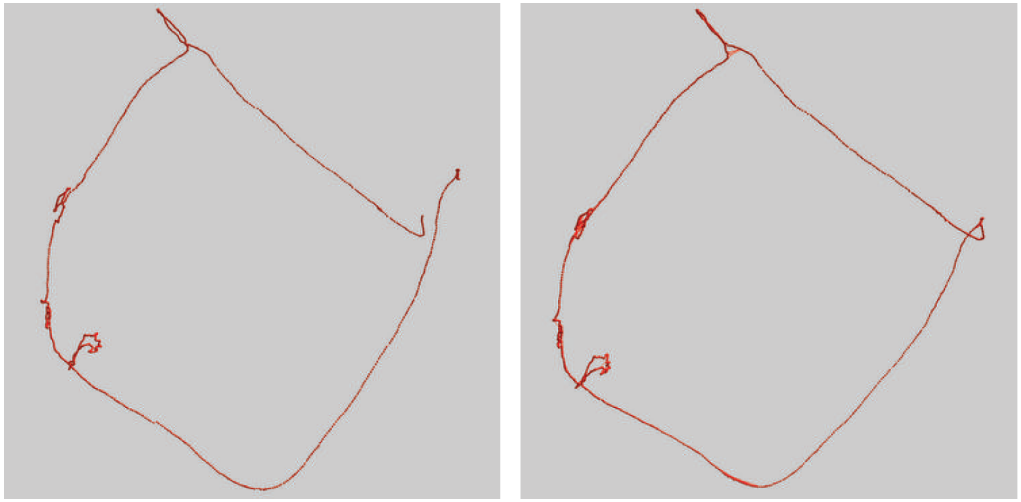

Figure: Henry et al.,*RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments* in: **International Journal of Robotics Research**, 2012

Global Optimization

- imperfect alignment => accumulated error (drift)
- goal: minimize drift
- different approaches possible
- here: loop closure detection
  1. detect if same location is visited for the second time
  2. use information to optimize map:

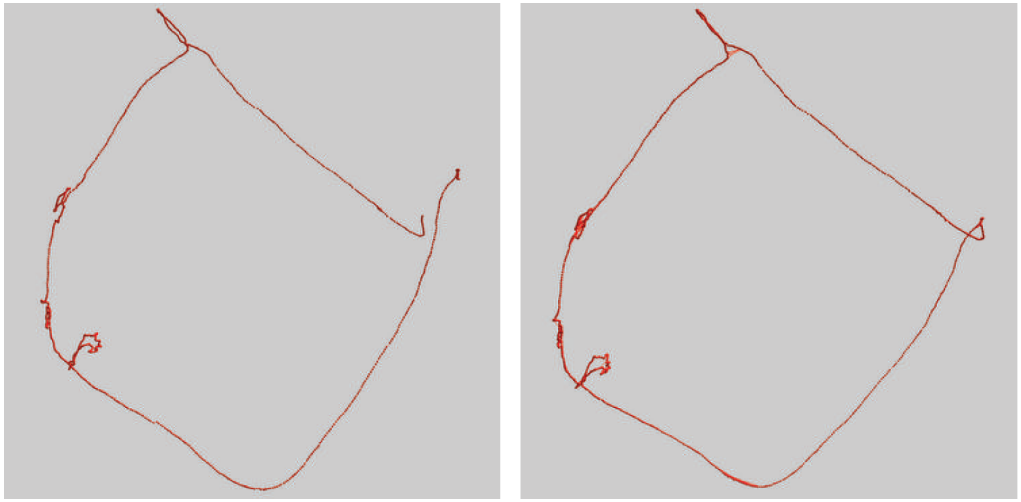

Figure: Henry et al.,*RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments* in: **International Journal of Robotics Research**, 2012

Global Optimization

- imperfect alignment => accumulated error (drift)
- goal: minimize drift
- different approaches possible
- here: loop closure detection
  1. detect if same location is visited for the second time
  2. use information to optimize map:

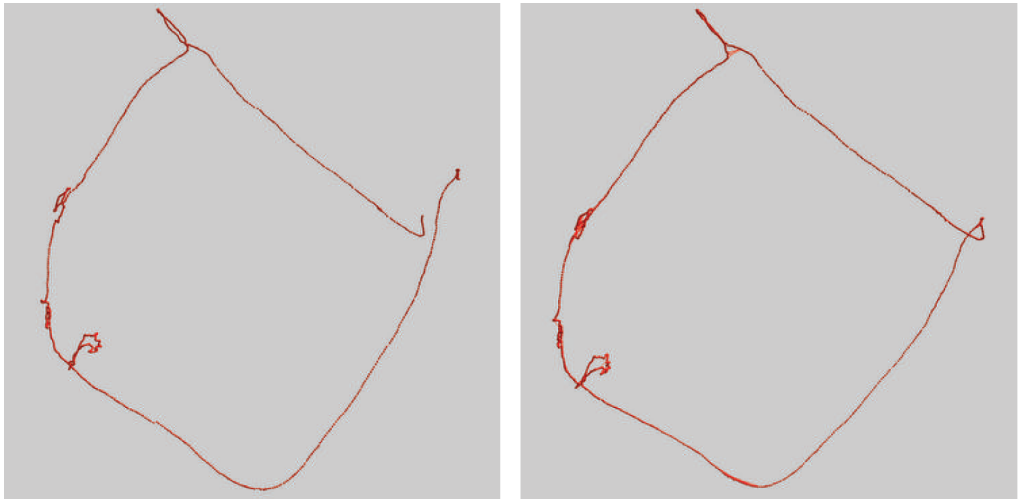

Figure: Henry et al.,*RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments* in: **International Journal of Robotics Research**, 2012

Global Optimization

- imperfect alignment => accumulated error (drift)
- goal: minimize drift
- different approaches possible
- here: loop closure detection
  1. detect if same location is visited for the second time
  2. use information to optimize map:

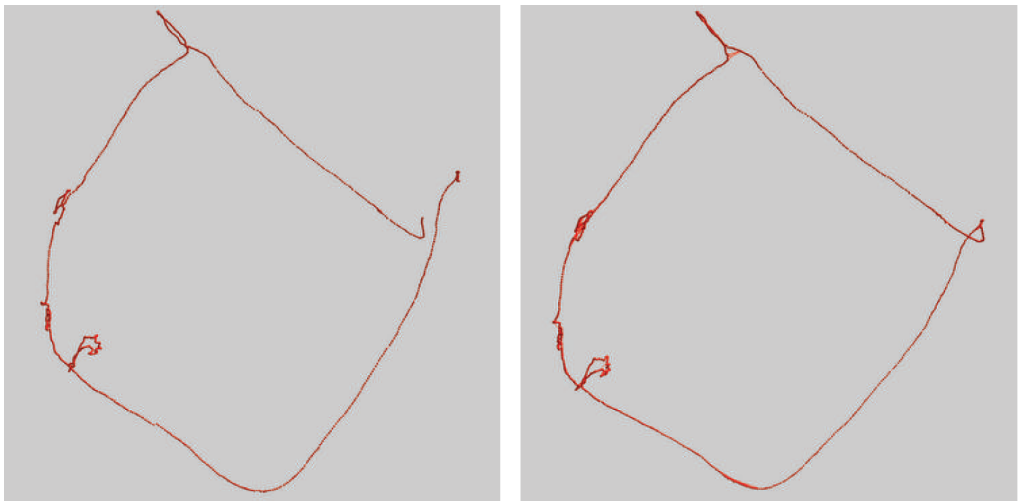

Figure: Henry et al.,*RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments* in: **International Journal of Robotics Research**, 2012

Global Optimization

- imperfect alignment => accumulated error (drift)
- goal: minimize drift
- different approaches possible
- here: loop closure detection
  1. detect if same location is visited for the second time
  2. use information to optimize map:

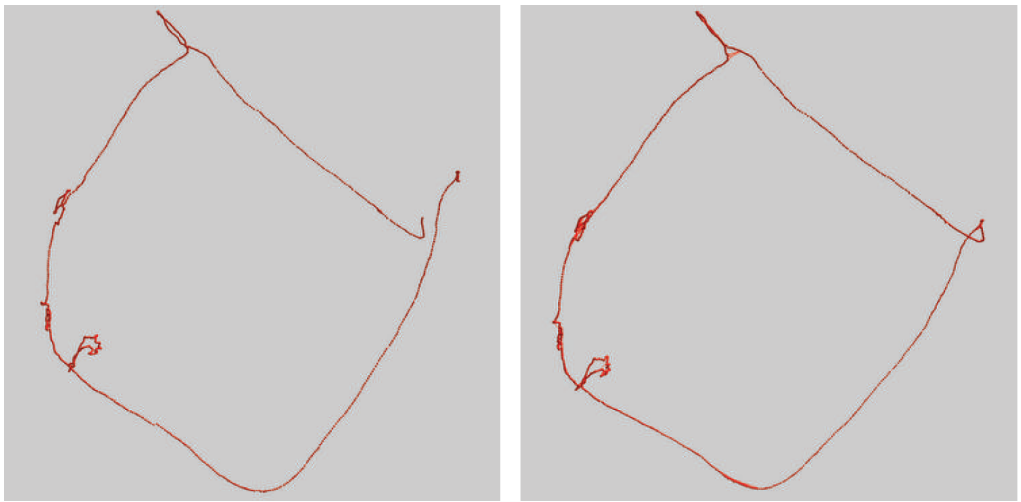

Figure: Henry et al.,*RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments* in: **International Journal of Robotics Research**, 2012

Global Optimization

- imperfect alignment => accumulated error (drift)
- goal: minimize drift
- different approaches possible
- here: loop closure detection
  1. detect if same location is visited for the second time
  2. use information to optimize map:

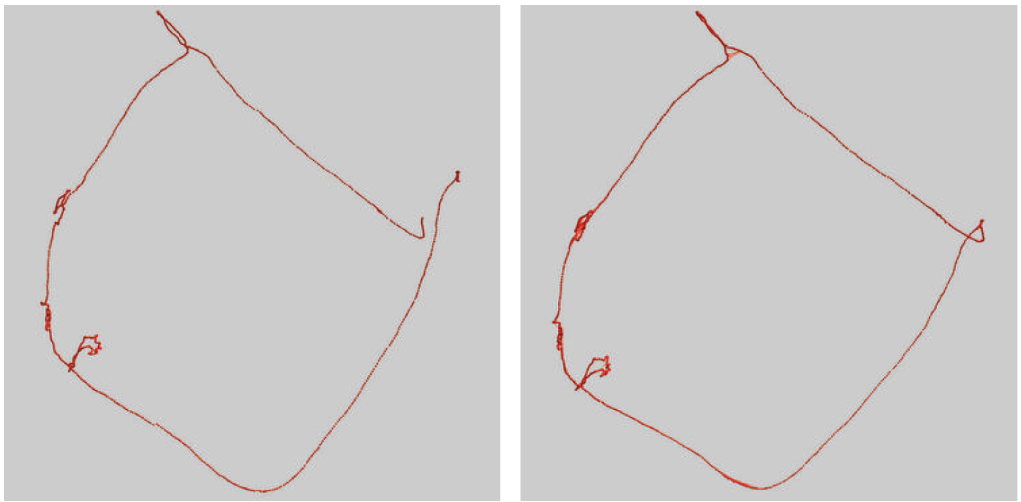

Figure: Henry et al.,*RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments* in: **International Journal of Robotics Research**, 2012

Outline

Internal Representation

- dense point cloud
  - expensive regarding space
  - no improvement possible
  - a lot of redundancy
- surfels
  - improvements possible
  - needs less space
- voxels
  - improvements possible
  - can store free space explicitly
  - multi resolution mapping

Internal Representation

- dense point cloud
  - expensive regarding space
  - no improvement possible
  - a lot of redundancy
- surfels
  - improvements possible
  - needs less space
- voxels
  - improvements possible
  - can store free space explicitly
  - multi resolution mapping

Internal Representation

- dense point cloud
  - expensive regarding space
  - no improvement possible
  - a lot of redundancy
- surfels
  - improvements possible
  - needs less space
- voxels
  - improvements possible
  - can store free space explicitly
  - multi resolution mapping

Internal Representation

- dense point cloud
  - expensive regarding space
  - no improvement possible
  - a lot of redundancy
- surfels
  - improvements possible
  - needs less space
- voxels
  - improvements possible
  - can store free space explicitly
  - multi resolution mapping

Internal Representation

- dense point cloud
  - expensive regarding space
  - no improvement possible
  - a lot of redundancy
- surfels
  - improvements possible
  - needs less space
- voxels
  - improvements possible
  - can store free space explicitly
  - multi resolution mapping

Internal Representation

- dense point cloud
  - expensive regarding space
  - no improvement possible
  - a lot of redundancy
- surfels
  - improvements possible
  - needs less space
- voxels
  - improvements possible
  - can store free space explicitly
  - multi resolution mapping

Internal Representation

- dense point cloud
  - expensive regarding space
  - no improvement possible
  - a lot of redundancy
- surfels
  - improvements possible
  - needs less space
- voxels
  - improvements possible
  - can store free space explicitly
  - multi resolution mapping

Internal Representation

- dense point cloud
  - expensive regarding space
  - no improvement possible
  - a lot of redundancy
- surfels
  - improvements possible
  - needs less space
- voxels
  - improvements possible
  - can store free space explicitly
  - multi resolution mapping

Internal Representation

- dense point cloud
  - expensive regarding space
  - no improvement possible
  - a lot of redundancy
- surfels
  - improvements possible
  - needs less space
- voxels
  - improvements possible
  - can store free space explicitly
  - multi resolution mapping

Internal Representation

- dense point cloud
  - expensive regarding space
  - no improvement possible
  - a lot of redundancy
- surfels
  - improvements possible
  - needs less space
- voxels
  - improvements possible
  - can store free space explicitly
  - multi resolution mapping

Internal Representation

- dense point cloud
  - expensive regarding space
  - no improvement possible
  - a lot of redundancy
- surfels
  - improvements possible
  - needs less space
- voxels
  - improvements possible
  - can store free space explicitly
  - multi resolution mapping

Outline

Practical Demonstration

- YouTube Video from Henry et al.:
- http://www.youtube.com/watch?v=58_xG8AkcaE&feature=player_

Outline

Thank you for your attention!
Questions?