Elastion Fusion

Dense SLAM without a Pose Graph

L. Freda

ALCOR Lab
DIAG
University of Rome "La Sapienza"

September 27, 2016

Outline

1. Introduction
   - V-SLAM Main Components
   - V-SLAM Challenges and Approaches
2. Optimization Graphs
   - Pose Graph vs Deformation Graph
3. Elastic Fusion
   - Main Characteristics
   - Elastic Fusion Pipeline
   - Depth Map Pre-processing
   - Surfel-based Map
   - Dense Camera Tracking
   - Local Loop Closure
   - Global Loop Closure
   - Deformation Graph

Outline

1. Introduction
   - V-SLAM Main Components
   - V-SLAM Challenges and Approaches
2. Optimization Graphs
   - Pose Graph vs Deformation Graph
3. Elastic Fusion
   - Main Characteristics
   - Elastic Fusion Pipeline
   - Depth Map Pre-processing
   - Surfel-based Map
   - Dense Camera Tracking
   - Local Loop Closure
   - Global Loop Closure
   - Deformation Graph

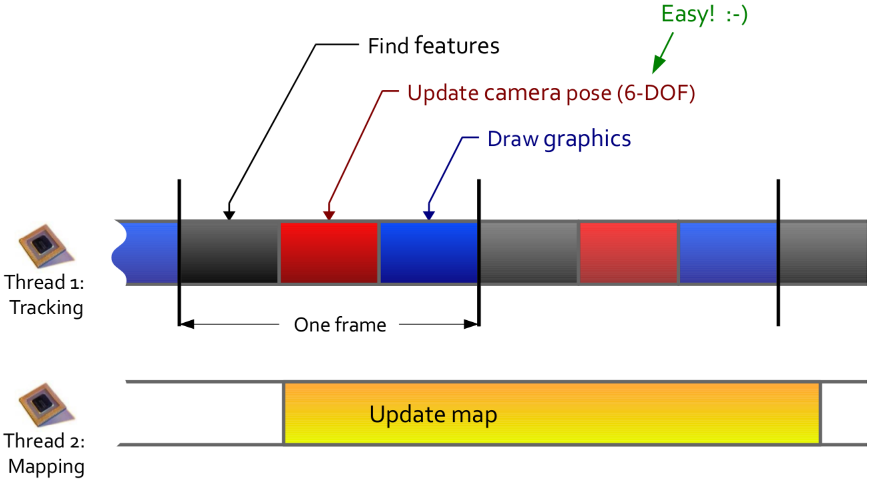
Typical V-SLAM **main modules**:

- **Front-end**
  1. Initialization module
  2. Real-time camera tracking
  3. Key-frames selection module
  4. Loop closure detection: local and/or global
  5. Map management (fusion/insertion of new data)
- **Back-end**
  6. Bundle adjustment: optimization on both key-points and poses; this can be local(windowed) and/or global
     or/and
  7. Pose graph optimization: refinement only on poses

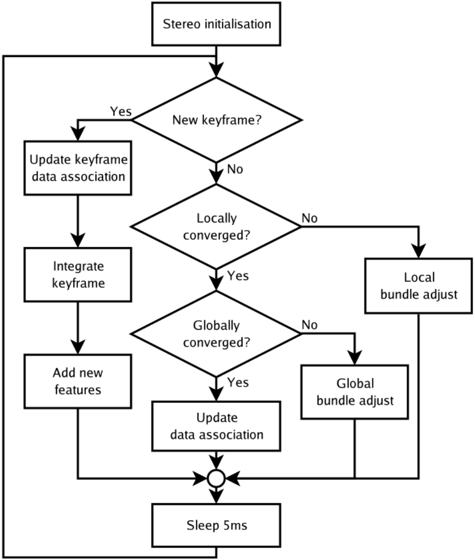
NOTES

- 2,3 and 6(local) can be considered part of a standard **visual odometry** module
- If robot gets lost, loop closure detection module is generally used as a **relocalizer**

PTAM Threads



PTAM steps



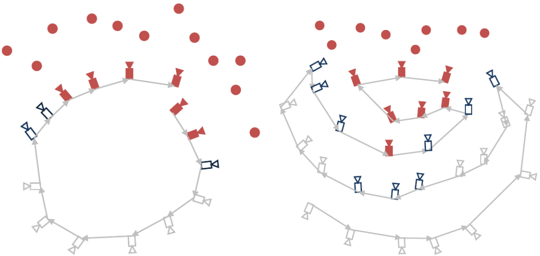
Stereo initialisation
New keyframe?
Yes
No
Update keyframe data association
Integrate keyframe
Add new features
Locally converged?
No
Yes
Local bundle adjust
Globally converged?
No
Yes
Global bundle adjust
Update data association
Sleep 5ms

Outline

1. Introduction
   - V-SLAM Main Components
   - V-SLAM Challenges and Approaches
2. Optimization Graphs
   - Pose Graph vs Deformation Graph
3. Elastic Fusion
   - Main Characteristics
   - Elastic Fusion Pipeline
   - Depth Map Pre-processing
   - Surfel-based Map
   - Dense Camera Tracking
   - Local Loop Closure
   - Global Loop Closure
   - Deformation Graph

Visual SLAM has to cope with the following **challenge**: sensors typically make movements which are both

- long exploration motions (towards unknown regions)
- loopy "painting" motions in the close vicinity (criss-cross loop back on themselves)

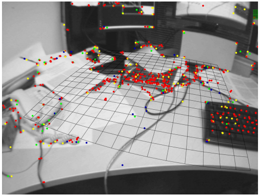


(a) Exploration & loop closure (b) Loopy browsing

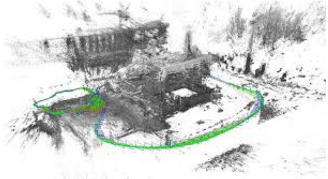
Visual SLAM Challenges

Visual SLAM methods typically target one of the two following **scenarios**

1. small areas with loopy motions; goal: accurate localization in the immediate space

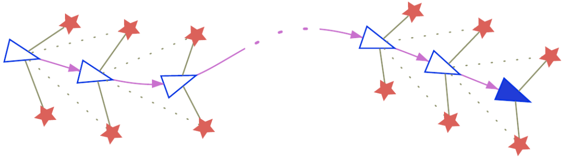


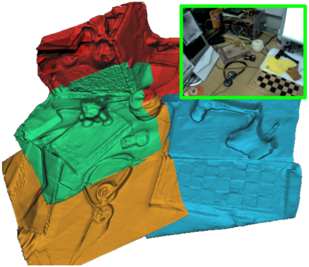
2. large areas with "corridor-like" motions and infrequent loops; goal: long range navigation, exploration and planning



Sparse feature-based V-SLAM deals with

1. **loopy local motions** by estimating at the same time poses and features with:
    - joint probabilistic filtering: e.g. EKF, particle filters
      OR
    - in-the-loop joint optimization: bundle adjustment
2. **large scale loop** by
    - partioning the map into local maps or keyframes
    - applying pose graph optimization





In **dense** V-SLAM systems

1. the **number of points** matched and measured at each sensor frame is much higher than in feature-based systems (typically hundreds of thousands)
2. **joint filtering or bundle adjustment**, on both features and poses, are computationally **unfeasible**
3. per-surface element independent filtering is a widely used technique (**volumetric fusion** based mapping, parallel implementation)

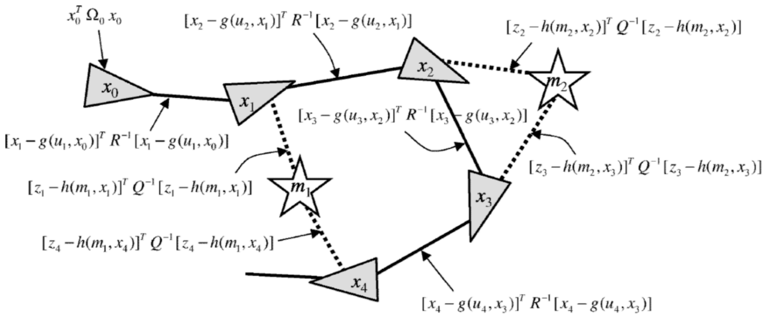
Outline

1. Introduction
   - V-SLAM Main Components
   - V-SLAM Challenges and Approaches
2. Optimization Graphs
   - Pose Graph vs Deformation Graph
3. Elastic Fusion
   - Main Characteristics
   - Elastic Fusion Pipeline
   - Depth Map Pre-processing
   - Surfel-based Map
   - Dense Camera Tracking
   - Local Loop Closure
   - Global Loop Closure
   - Deformation Graph

Pose Graph vs Deformation Graph

- What do these graphs perform? Answer: Optimization/Refinement
- Where do these graphs focus on?
  - a **pose graphs** primarily focus on optimising the camera **trajectory** (trajectory-centric approach)
  - a **deformation graph** instead focuses on optimising the **map** (map-centric approach)
- Where do these graphs are located?
  - a pose graph is embedded in the trajectory and rigidly transform its independent **keyframes**
  - a deformation graph is directly embedded in the **surface** model of the environment (map)

Pose Graph



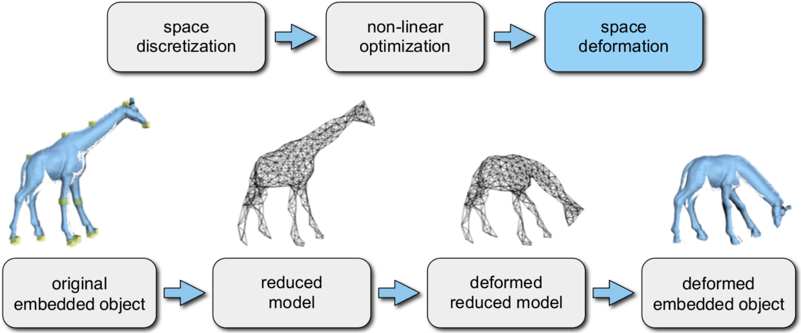
<u>Sum of all constraints:</u>

$$J_{\text{GraphSLAM}} = x_0^T \Omega_0 x_0 + \sum_t [x_t - g(u_t, x_{t-1})]^T R^{-1} [x_t - g(u_t, x_{t-1})] + \sum_t [z_t - h(m_{c_t}, x_t)]^T Q^{-1} [z_t - h(m_{c_t}, x_t)]$$

- initial location constraint (on x_0)
- relative motion constraints (between x_i and x_j directly)
- relative measurement constraints (between x_k and x_h through m_i)

Deformation Graph

See slides *"Deformation Graphs"* by Mark Pauly



NOTE: think about the deformation graph as a net of small spheres inter-connected by springs

Outline

1. Introduction
   - V-SLAM Main Components
   - V-SLAM Challenges and Approaches
2. Optimization Graphs
   - Pose Graph vs Deformation Graph
3. Elastic Fusion
   - Main Characteristics
   - Elastic Fusion Pipeline
   - Depth Map Pre-processing
   - Surfel-based Map
   - Dense Camera Tracking
   - Local Loop Closure
   - Global Loop Closure
   - Deformation Graph

Elastic Fusion is a dense V-SLAM which uses **RGB-D cameras**
Its main characteristics

- Real-time **dense** frame-to-model camera **tracking**[1] by using both photometric and geometric errors
- **Surfel-based map** (room scale) and windowed surfel-based fusion
- Model optimization through **non-rigid surface deformations** (surface deformation graph, no pose-graph optimization)
- **Local** model-to-model surface **loop closures** with non-rigid space deformation
- **Global loop-closures** to recover from drift (appearance-based place recognition)

[1] Full depth maps are fused into a surfel-based map, which is then rendered to produce a predicted surface that the subsequently captured depth map is matched against using ICP

GPU-based Pipeline

- CUDA are used to implement tracking (fast parallel processing)
- OpenGL Shading Language is used for view prediction and map management

Outline

1. Introduction
   - V-SLAM Main Components
   - V-SLAM Challenges and Approaches
2. Optimization Graphs
   - Pose Graph vs Deformation Graph
3. Elastic Fusion
   - Main Characteristics
   - Elastic Fusion Pipeline
   - Depth Map Pre-processing
   - Surfel-based Map
   - Dense Camera Tracking
   - Local Loop Closure
   - Global Loop Closure
   - Deformation Graph

Main pipeline

1. **grab** current RGB-D image: color data $\mathcal{C}_i$ and depth data $\mathcal{D}_i$
2. **pre-process** the depth data (bilateral filtering)
3. **estimate** the current six 6DoF **camera pose** relative to the scene model (frame-to-model camera tracking)
4. use the estimated pose to convert depth samples into a unified coordinate space and **fuse** them into an accumulated global model
5. check for **local** surfaces **loop** closures
6. check for **global loop** closures
7. **refine** in a separate thread the surfel-based map by using the **deformation graph**

Outline

1. Introduction
   - V-SLAM Main Components
   - V-SLAM Challenges and Approaches
2. Optimization Graphs
   - Pose Graph vs Deformation Graph
3. Elastic Fusion
   - Main Characteristics
   - Elastic Fusion Pipeline
   - Depth Map Pre-processing
   - Surfel-based Map
   - Dense Camera Tracking
   - Local Loop Closure
   - Global Loop Closure
   - Deformation Graph

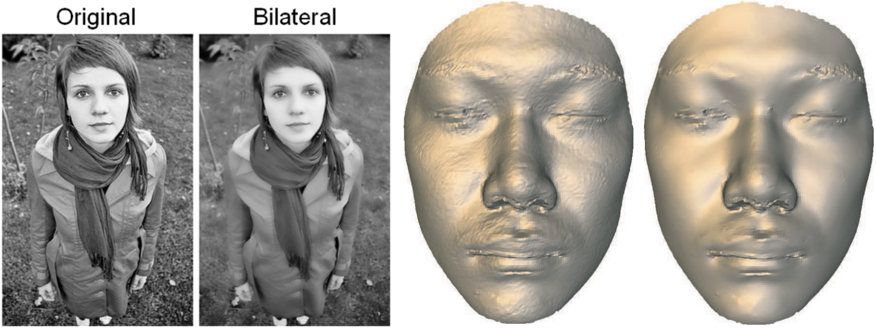
Depth Map Processing

- at time frame i sensor returns a **raw depth map** $\mathcal{D}_i$ and a **raw color map** $\mathcal{C}_i$
- pixel $\mathbf{u} = [x, y]^T \in \Omega \subset \mathbb{N}^2$
- $d_i(\mathbf{u}) \in \mathbb{R}$ is the **depth** along the direction at pixel $\mathbf{u}$
- $\mathbf{c}_i(\mathbf{u}) \in \mathbb{N}^3$ is the **color** at pixel $\mathbf{u}$ and $I(\mathbf{u}, \mathcal{C}_i) \triangleq (c_1 + c_2 + c_3)/3$
- given the intrinsic camera calibration matrix $\mathbf{K}$, $\mathcal{D}_i$ is transformed into a corresponding **vertex map** $\mathcal{V}_i$, by converting each depth sample $d_i(u)$ into a vertex position $\mathbf{p}_i(\mathbf{u}, \mathcal{D}_i) = d_i(\mathbf{u})\mathbf{K}^{-1}[\mathbf{u}, 1]^T \in \mathbb{R}^3$ in camera space
- a **normal map** $\mathcal{N}_i$ is computed from the vertex map by using the central difference $\mathbf{n}_i(\mathbf{u}) = \mathrm{normalize}(\mathbf{p}_k(x+1, y) - \mathbf{p}_k(x, y)) \times (\mathbf{p}_k(x, y+1) - \mathbf{p}_k(x, y)) \in \mathbb{R}^3$
- the camera projection of a 3D point $\mathbf{p} = [x, y, z] \in \mathbb{R}^3$ (represented in camera frame) is $\mathbf{u} = \pi(\mathbf{K}\mathbf{p})$, where $\pi(\mathbf{p}) = [x/z, y/z]$ denotes the dehomogenisation operation
- a camera pose is represented by the matrix $\mathbf{T}_i = \begin{bmatrix} \mathbf{R}_i & \mathbf{t}_i \\ \mathbf{0} & 1 \end{bmatrix} \in SE(3)$

  one has $\mathbf{p}^w = \mathbf{T}_i \mathbf{p}^c$

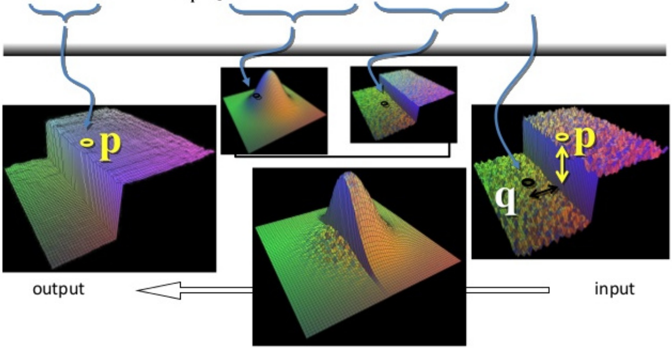
Depth Map Pre-processing

- before computing the vertex map $\mathcal{V}_i$, the raw depth map $\mathcal{D}_i$ is processed by using a bilateral filter (edge-preserving filter)



Bilateral Filter

$$BF[\dot{Q}]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q}\in S} G_{\sigma_s}(\|\mathbf{p}-\mathbf{q}\|)\; G_{\sigma_r}(|I_{\mathbf{p}}-I_{\mathbf{q}}|)\; \dot{Q}_{\mathbf{q}}$$

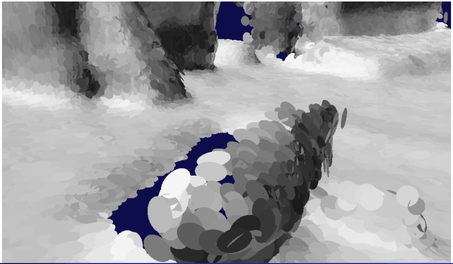


Outline

1. Introduction
   - V-SLAM Main Components
   - V-SLAM Challenges and Approaches
2. Optimization Graphs
   - Pose Graph vs Deformation Graph
3. Elastic Fusion
   - Main Characteristics
   - Elastic Fusion Pipeline
   - Depth Map Pre-processing
   - Surfel-based Map
   - Dense Camera Tracking
   - Local Loop Closure
   - Global Loop Closure
   - Deformation Graph

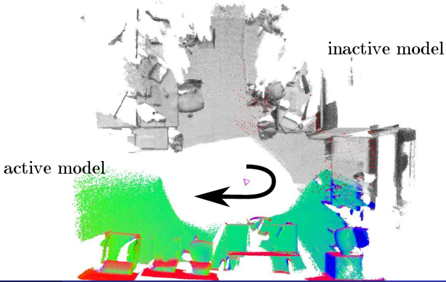
Map representation

- the scene representation $\mathcal{M}$ is an unordered list of surfels
- each surfel $\mathbf{s}_k$ has the following **attributes**:
  - position $\mathbf{p}_k \in \mathbb{R}^3$
  - normal $\mathbf{n}_k \in \mathbb{R}^3$
  - colour $\mathbf{c}_k \in \mathbb{N}^3$
  - weight $w_k \in R$ (*confidence* counter)
  - radius $r_k \in R$ (represents the local surface around point $\mathbf{p}_k$)
  - initialisation timestamp t_k^0 and last updated timestamp t_k



Active model vs Inactive model

- a time window threshold δt divides the map $\mathcal{M}$ into surfels which are active and inactive
- a surfel $\mathbf{s}_k$ in $\mathcal{M}$ is declared as inactive when the time since that surfel was last updated (i.e. had a raw depth measurement associated with it for fusion) is greater than δt (i.e. one has $t - t_k > \delta t$)
- only surfels which are marked as active model surfels are used for camera pose estimation and depth map fusion



Surfels-based map fusion

Once the new camera pose is estimated, input points are fused into the global model

- a point confidence w_k evolves from unstable to stable status based on the confidence it gathered (essentially on how often it was observed by the sensor)
- data fusion first projectively associates each point in the current depth map $\{\mathcal{C}_i, \mathcal{D}_i\}$ with the set of points in the global model $\mathcal{M}$, by rendering the model points as an index map
- if corresponding points are found, the most reliable point is merged with the new point estimate using a weighted average
- if no reliable corresponding points are found, the new point estimate is added to the global model as an unstable point
- the global model is cleaned up over time to remove outliers due to visibility and temporal constraints (old unstable points)

Outline

1. Introduction
   - V-SLAM Main Components
   - V-SLAM Challenges and Approaches
2. Optimization Graphs
   - Pose Graph vs Deformation Graph
3. Elastic Fusion
   - Main Characteristics
   - Elastic Fusion Pipeline
   - Depth Map Pre-processing
   - Surfel-based Map
   - Dense Camera Tracking
   - Local Loop Closure
   - Global Loop Closure
   - Deformation Graph

Dense Camera Tracking

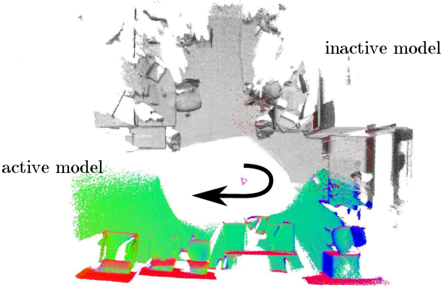
- the current camera view $\{\mathcal{C}_i, \mathcal{D}_i\}$ needs to be registered w.r.t. the active map model
- the registration provides the relative change from $\mathbf{T}_{i-1}$ to $\mathbf{T}_i$

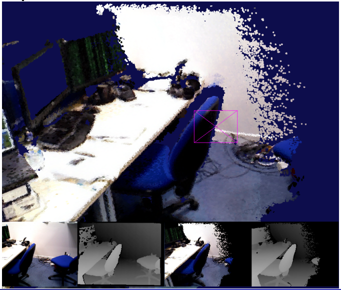
Basic steps

1. **render** the active map model from previous pose
2. **projective data association** between map vertices and current camera vertices
3. **minimize geometric** (ICP) and **photometric errors** for pose estimation

Dense Camera Tracking: Render Active Model

- the active map model is rendered into a synthetic colored depth map $\{\hat{\mathcal{C}}_{i-1}, \hat{\mathcal{D}}_{i-1}\}$, as seen from the previous frame's camera pose $\mathbf{T}_{i-1}$
- the renderer (OpenGL) draws the point-based representation using a simple surface-splatting technique: this renders into the viewport all the overlapping, disk-shaped surface splats that are spanned by the model point's position $\mathbf{p}_k$, radius r_k and normal $\mathbf{n}_k$
- only **active points** are rendered during this process



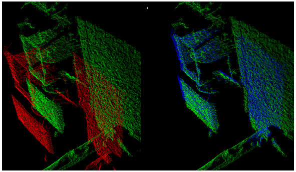


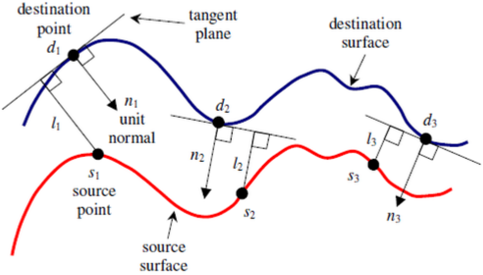
Dense Camera Tracking: **Projective Data Association**

- the first step of ICP finds correspondences between the current oriented points computed from $\{\mathcal{C}_i, \mathcal{D}_i\}$ and the set of active points in $\{\hat{\mathcal{C}}_{i-1}, \hat{\mathcal{D}}_{i-1}\}$
- given the inverse global camera pose $\mathbf{T}_{i-1}^{-1}$ and the camera calibration matrix $\mathbf{K}$, each point $\mathbf{p}_k \in \mathcal{M}$ is projected onto the image plane of the current camera view as $\mathbf{u}_k = \pi(\mathbf{K}\mathbf{T}_i^{-1}\mathbf{p}_k)$
- the index k is stored in pixel $\mathbf{u}_k$, building a sparse **index map** $\mathcal{I}_i$

Dense Camera Tracking: **ICP**

- a dense hierarchical ICP is used to align the bilateral filtered input depth map $\mathcal{D}_i$ (of the current frame i) with the reconstructed model
- 3 hierarchy levels, with the finest level at the cameras resolution; unstable model points are ignored



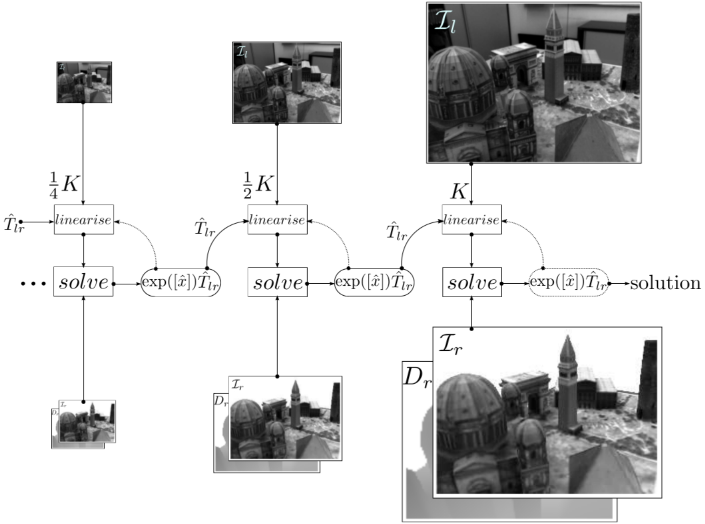


Point-to-plane error between two surfaces.

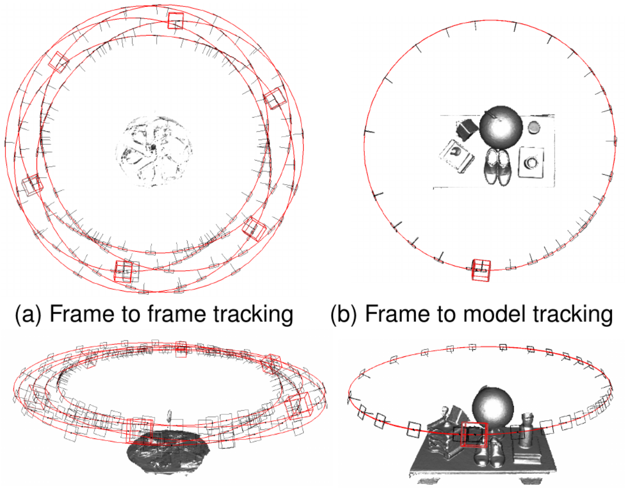
Dense Camera Tracking: **Errors Minimization**

- compute the relative change from $\mathbf{T}_{i-1}$ to $\mathbf{T}_i$, that is $\mathbf{T}_{i-1,i}$
- **geometric error**
  $E_{icp} = \sum_k \left(\left(\mathbf{p}_k - \exp(\xi)\mathbf{T}_{i-1,i}\mathbf{p}_k^i \right) \cdot \mathbf{n}^k \right)^2$
- **photometric error**
  $E_{rgb} = \sum_{\mathbf{u}\in\Omega} \left(I(\mathbf{u}, \mathcal{C}_i) - I\left(\pi(\mathbf{K}\exp(\xi)\mathbf{T}_{i-1,i}\mathbf{p}(\mathbf{u}, \mathcal{D}_i), \hat{\mathcal{C}}_{i-1} \right) \right)^2$
- **total error**
  $E_{track} = E_{icp} + w_{rgb}E_{rgb} \qquad \text{with } w_{rgb} = 0.1$
- the solution is found by using **Gauss-Newton non-linear least-squares method**. At each optimization iteration $\mathbf{T}_{i-1,i}^{h+1} = \exp(\xi)\mathbf{T}_{i-1,i}^{h}$

Dense Camera Tracking: Hierarchical ICP

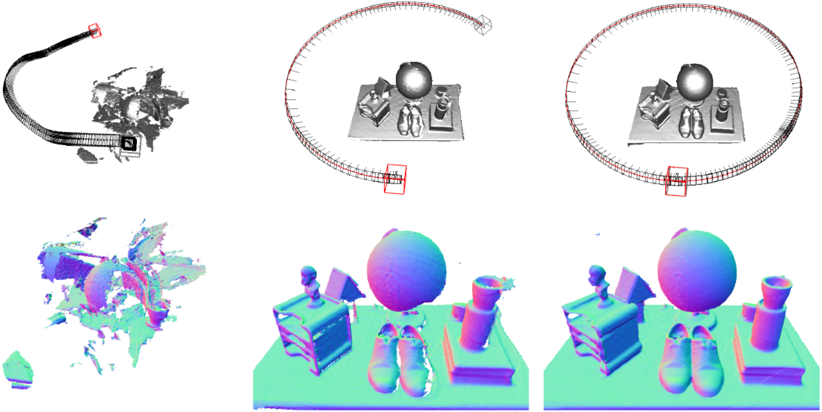


Dense Camera Tracking: Frame-to-Frame vs Frame-To-Model



(a) Frame to frame tracking (b) Frame to model tracking

Dense Camera Tracking: Frame-to-Frame vs Frame-To-Model



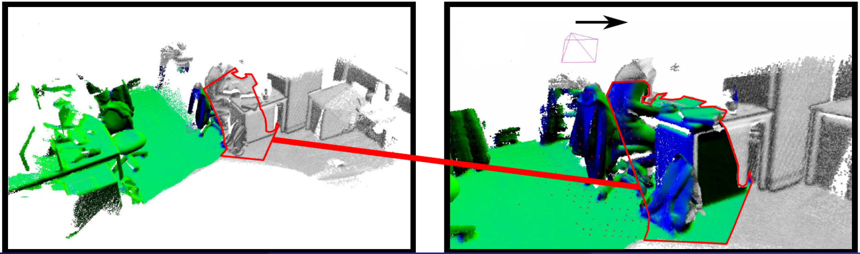
(a) Frame to frame tracking (b) Partial loop (c) Full loop

Outline

1. Introduction
   - V-SLAM Main Components
   - V-SLAM Challenges and Approaches
2. Optimization Graphs
   - Pose Graph vs Deformation Graph
3. Elastic Fusion
   - Main Characteristics
   - Elastic Fusion Pipeline
   - Depth Map Pre-processing
   - Surfel-based Map
   - Dense Camera Tracking
   - Local Loop Closure
   - Global Loop Closure
   - Deformation Graph

Local Loop Closure

- the **inactive area** of the map is not used for live frame tracking
- local loop closures detects **loops between the active model and inactive model**, at which point the matched inactive area becomes active again
- a **model-to-model** tracking is performed trying to register the current active model with the inactive model
- **register the predicted surface renderings** of active model, $\{\hat{\mathcal{C}}_i^a, \hat{\mathcal{D}}_i^a\}$, and inactive model, $\{\hat{\mathcal{C}}_i^{in}, \hat{\mathcal{D}}_i^{in}\}$, as seen from the latest pose estimate
- the found set of surface associations is used to define **local constraints** in the **deformation graph optimization**



Outline

1. Introduction
   - V-SLAM Main Components
   - V-SLAM Challenges and Approaches
2. Optimization Graphs
   - Pose Graph vs Deformation Graph
3. Elastic Fusion
   - Main Characteristics
   - Elastic Fusion Pipeline
   - Depth Map Pre-processing
   - Surfel-based Map
   - Dense Camera Tracking
   - Local Loop Closure
   - Global Loop Closure
   - Deformation Graph

Global Loop Closure

- **appearance-based** place recognition
- **ferns** encode an RGB-D image as a **string of codes**
- a **fern encoded frame database** is built
- if a matching frame $\{\mathcal{C}^f, \mathcal{D}^f\}$ is found in the database a number of steps to potentially globally align the surfel map with it is performed
- first, the method tries to **align** $\{\mathcal{C}_i, \mathcal{D}_i\}$ with $\{\mathcal{C}^f, \mathcal{D}^f\}$ and computes a relative transformation
- then, if the transformation is computed successfully, it is used to globally optmize the deformation graph and check if the computed deformation is **consistent** with **map geometry**
- if the quality of the optimization is good the map is updated

Outline

1. Introduction
   - V-SLAM Main Components
   - V-SLAM Challenges and Approaches
2. Optimization Graphs
   - Pose Graph vs Deformation Graph
3. Elastic Fusion
   - Main Characteristics
   - Elastic Fusion Pipeline
   - Depth Map Pre-processing
   - Surfel-based Map
   - Dense Camera Tracking
   - Local Loop Closure
   - Global Loop Closure
   - Deformation Graph

Deformation Graph Optimization

- ensures local and global surface **consistency**
- **non-rigidly deforms** all surfels (both active and inactive) according to **surface constraints** provided by both **local** and **global** loop closure methods
- **total error**
  $E_{def} = w_{rot} E_{rot} + w_{reg} E_{reg} + w_{con} E_{con} + w_{pin} E_{pin}$

Deformation Graph

- in this model, a space deformation is defined by a **collection of affine transformations**
- **one transformation** is associated with **each node** of the deformation graph embedded in the map
- each affine transformation induces a **localized deformation** on the nearby space
- undirected edges connect nodes of overlapping influence
- the node positions are given by $\mathbf{g}_j \in \mathbb{R}^3$
- the affine transformation for node j is specified by a 3x3 matrix $\mathbf{R}_{\ j}$ and a 3x1 translation vector t_j
- the influence of the transformation is centered at the node's position $\mathbf{g}_j$ so that it maps any point $\mathbf{p} \in \mathbb{R}^3$ to the position $\mathbf{p}_{new}$ according to $\mathbf{p}_{new} = \mathbf{R}_j(\mathbf{p} - \mathbf{g}_j) + \mathbf{g}_j + \mathbf{t}_j$
- the influence of individual graph nodes is smoothly blended so that the deformed position $\mathbf{p}_{new}$ of each shape point $\mathbf{p}$ is $\mathbf{p}_{new} = \sum_j w_j(\mathbf{p}_j)\mathbf{R}_j(\mathbf{p}_j - \mathbf{g}_j) + \mathbf{g}_j + \mathbf{t}_j$

Deformation Graph: Update

Algorithm 1: Deformation Graph Application

Input: $\mathcal{M}^s$ surfel to be deformed
$\mathcal{G}$ set of deformation nodes
α number of nodes to explore
Output: $\hat{\mathcal{M}}^s$ deformed surfel
do

```
// Find closest node in time
```
$c \leftarrow \arg\min_i \left\| \mathcal{M}^s_{t_0} - \mathcal{G}^i_{t_0} \right\|_1$
```
// Gather set of temporally nearby nodes
```
$\mathcal{I} \leftarrow \emptyset$
for $i \leftarrow -\alpha/2$ **to** $\alpha/2$ **do**
$\quad \mathcal{I}^{i+\alpha/2} \leftarrow c + i$
sort_by_euclidean_distance$(\mathcal{I}, \mathcal{G}, \mathcal{M}^s_{\mathbf{p}})$
```
// Take closest k as influencing nodes
```
$\mathcal{I}(\mathcal{M}^s, \mathcal{G}) \leftarrow \mathcal{I}^{0 \to k-1}$
```
// Compute weights
```
$h \leftarrow 0$
$d_{max} \leftarrow \left\| \mathcal{M}^s_{\mathbf{p}} - \mathcal{G}^{\mathcal{I}^k}_{\mathbf{g}} \right\|_2$
for $n \in \mathcal{I}(\mathcal{M}^s, \mathcal{G})$ **do**
$\quad w^n(\mathcal{M}^s) \leftarrow (1 - \left\| \mathcal{M}^s_{\mathbf{p}} - \mathcal{G}^n_{\mathbf{g}} \right\|_2 / d_{max})^2$
$\quad h \leftarrow h + w^n(\mathcal{M}^s)$
```
// Apply transformations
```
$\hat{\mathcal{M}}^s_{\mathbf{p}} = \sum_{n \in \mathcal{I}(\mathcal{M}^s, \mathcal{G})} \frac{w^n(\mathcal{M}^s)}{h} \left[\mathcal{G}^n_{\mathbf{R}} (\mathcal{M}^s_{\mathbf{p}} - \mathcal{G}^n_{\mathbf{g}}) + \mathcal{G}^n_{\mathbf{g}} + \mathcal{G}^n_{\mathbf{t}} \right]$
$\hat{\mathcal{M}}^s_{\mathbf{n}} = \sum_{n \in \mathcal{I}(\mathcal{M}^s, \mathcal{G})} \frac{w^n(\mathcal{M}^s)}{h} \mathcal{G}^{n\ -1\top}_{\mathbf{R}} \mathcal{M}^s_{\mathbf{n}}$