The study of Google’s Cartographer

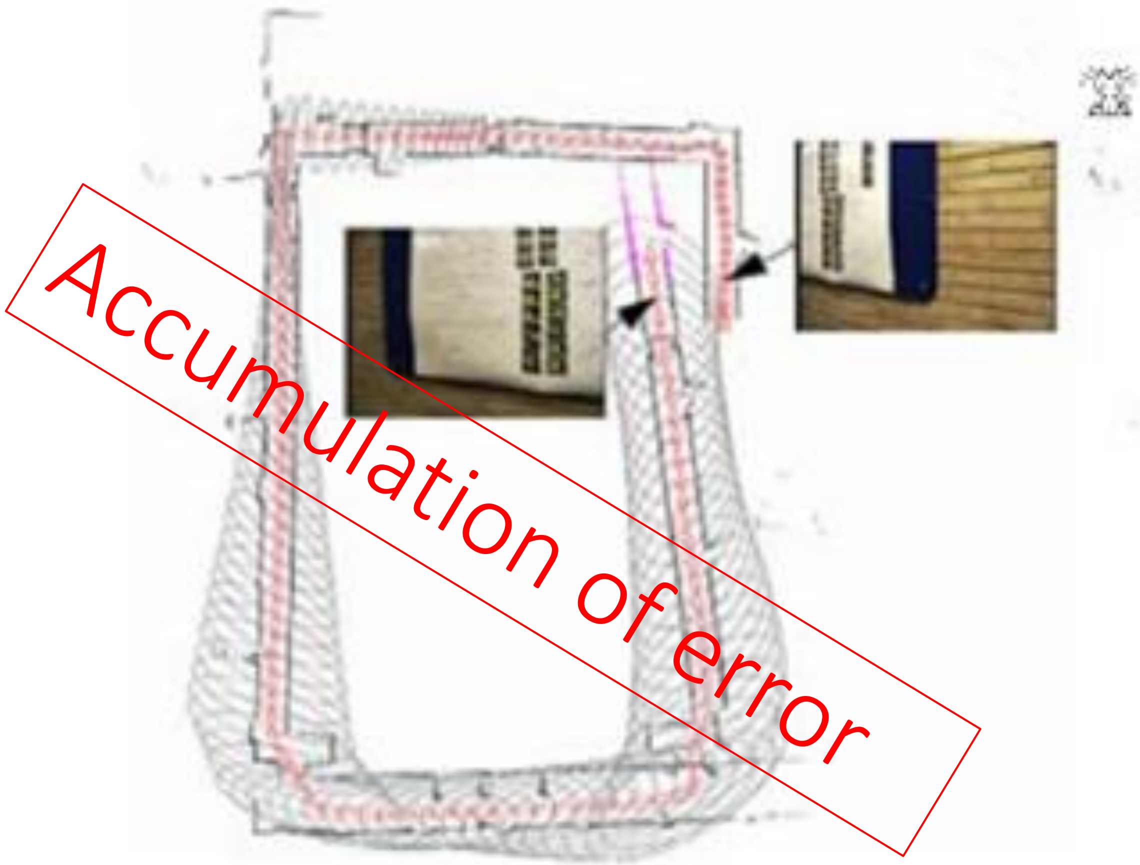
From Real-Time Loop Closure in 2D LIDAR SLAM

To IMPROVING GOOGLE’S CARTOGRAPHER 3D MAPPING BY CONTINUOUS-TIME SLAM

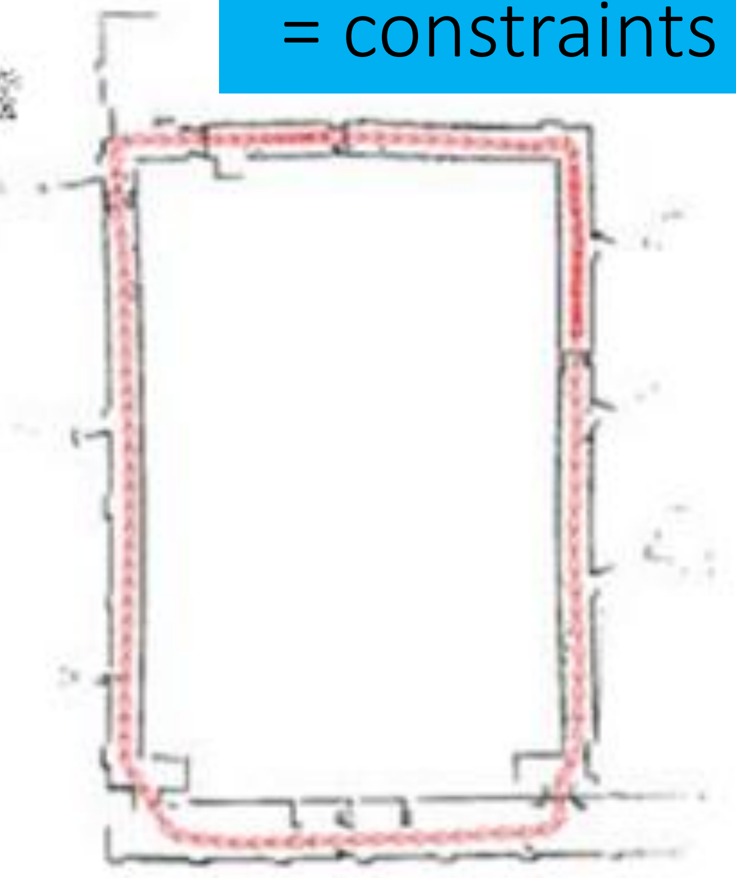
Zhang Xudong

What is loop closure detection of SLAM?

= constraints

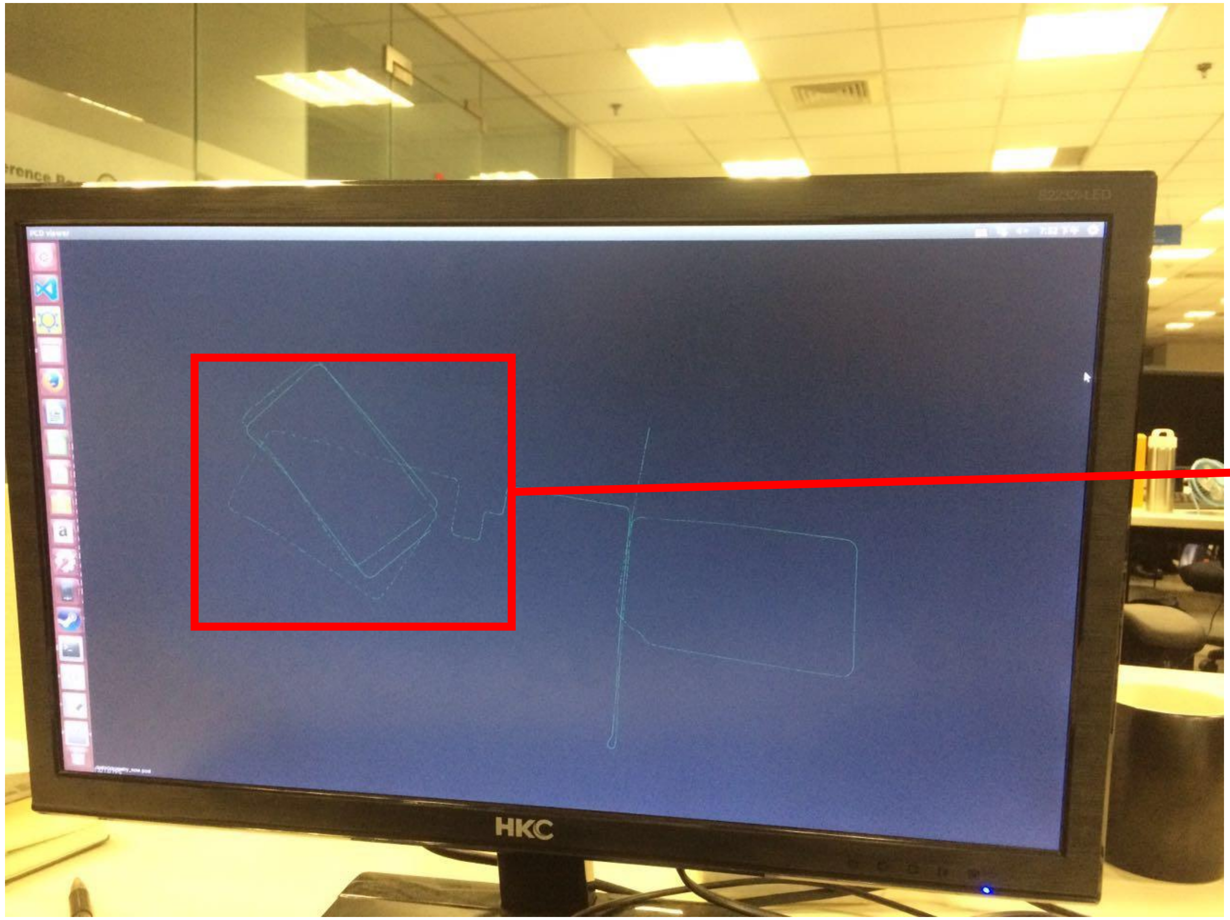


Without loop closure



With loop closure

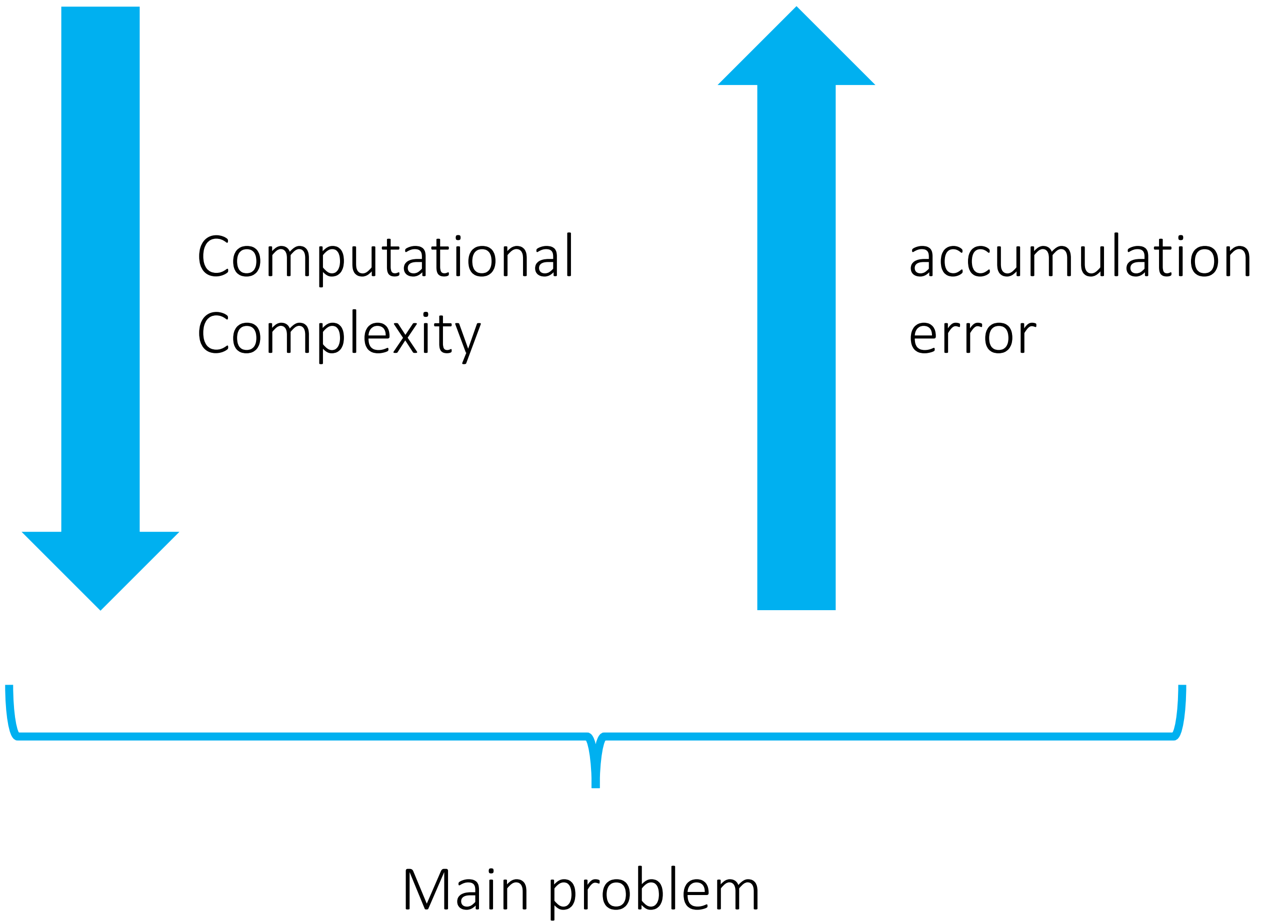
What trouble we have



Classification of SLAM (lidar)

- Scan-to-scan LOAM
- Scan-to-map
- Pixel-accurate scan

Loop closure detection



What does Cartographer do?

1. Loop closure detection using submaps

2. Reduce computational complexity using branch-and-bound scan matching

What is the submap?

- Discussion with the 2d lidar point cloud

$$\xi = (\xi_x, \xi_y, \xi_\theta)$$

translation rotation

$$\boxed{T_\xi p = \underbrace{\begin{pmatrix} \cos\xi_\theta & -\sin\xi_\theta \\ \sin\xi_\theta & \cos\xi_\theta \end{pmatrix}}_{R_\xi} p + \underbrace{\begin{pmatrix} \xi_x \\ \xi_y \end{pmatrix}}_{t_\xi}.}$$

What is the submap?

$$\text{odds}(p) = \frac{p}{1-p}, \tag{2}$$

$$M_{\text{new}}(x) = \text{clamp}(\text{odds}^{-1}(\text{odds}(M_{\text{old}}(x)) \cdot \text{odds}(p_{\text{hit}}))) \tag{3}$$

& equivalently for misses.

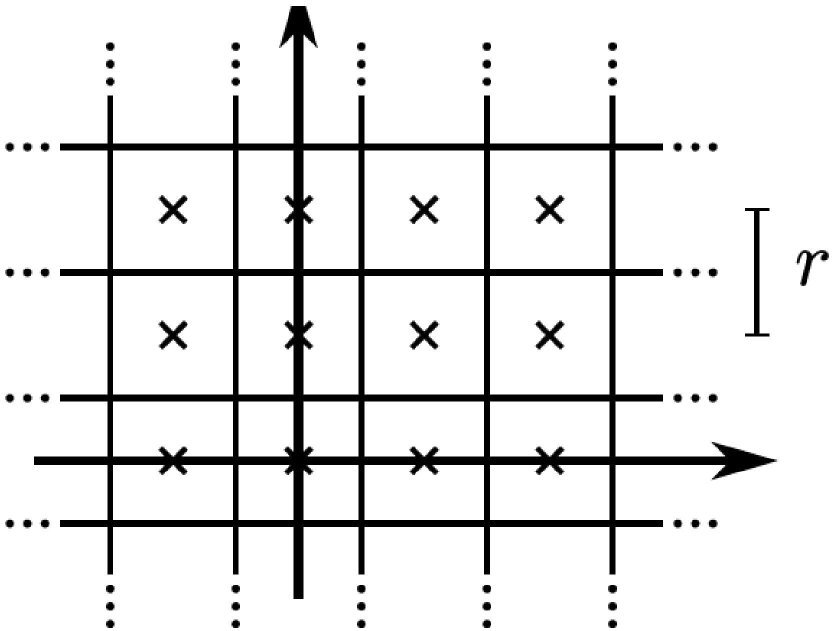


Fig. 1. Grid points and associated pixels.

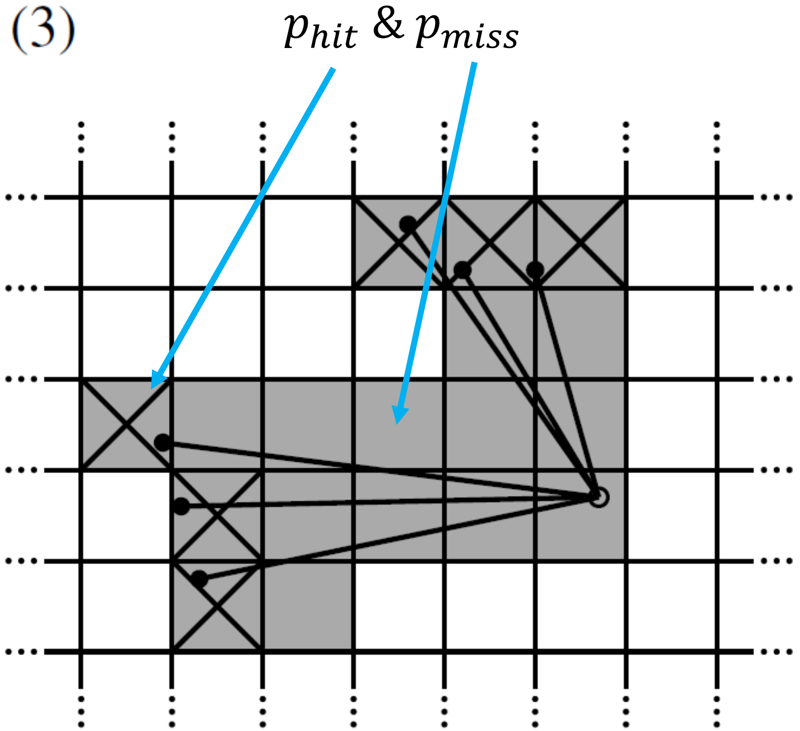


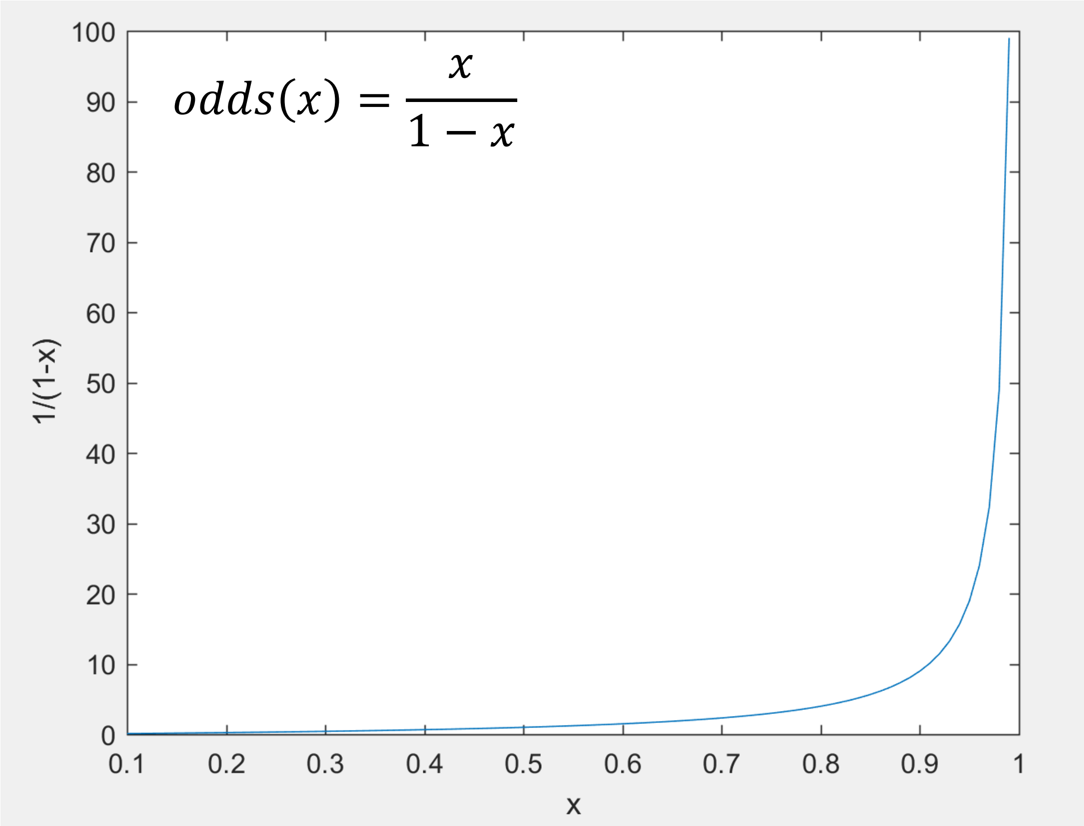
Fig. 2. A scan and pixels associated with *hits* (shaded and crossed out) and *misses* (shaded only).

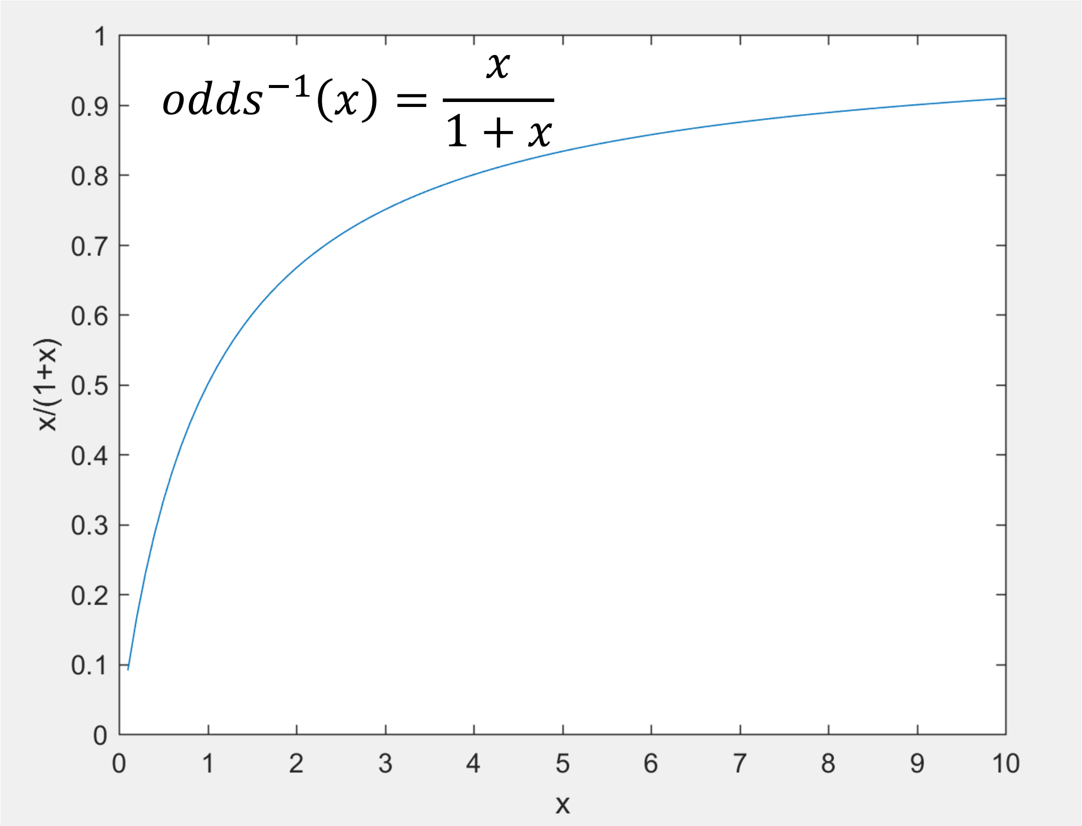
What is the submap?

$$\text{odds}(p) = \frac{p}{1-p}, \tag{2}$$

$$M_{\text{new}}(x) = \text{clamp}(\text{odds}^{-1}(\text{odds}(M_{\text{old}}(x)) \cdot \text{odds}(p_{\text{hit}}))) \tag{3}$$

& equivalently for misses.





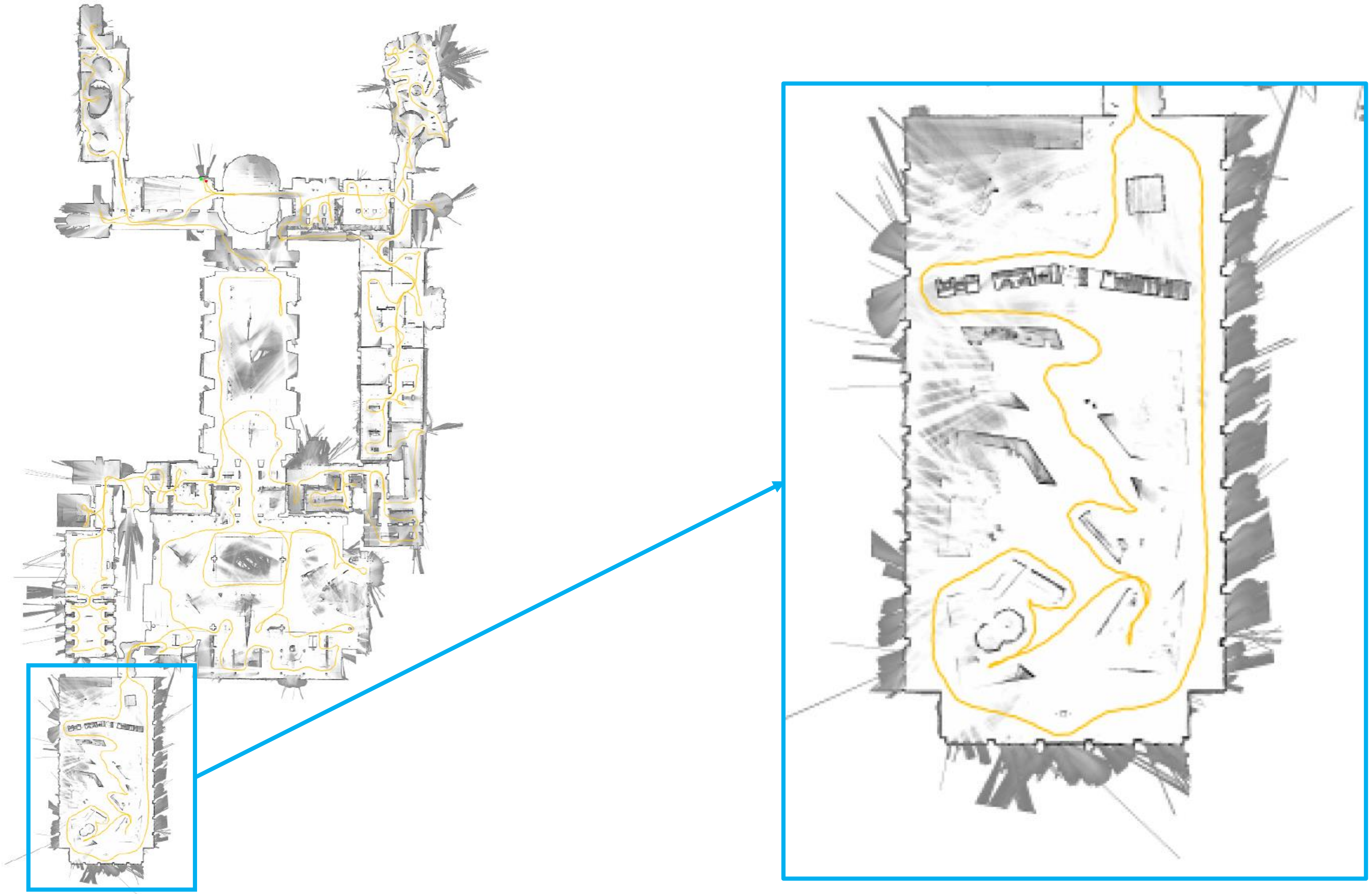


Fig. 4. Cartographer map of the 2nd floor of the Deutsches Museum.

Classification of SLAM (lidar)

- Scan-to-scan ← LOAM
- Scan-to-map
- Pixel-accurate scan

Loop closure detection

Computational Complexity

accumulation error

Main problem

How to produce submaps?

- Ceres scan matching

$$\underset{\xi}{\operatorname{argmin}} \quad \sum_{k=1}^{K} \left(1 - M_{\text{smooth}}(T_{\xi} h_k)\right)^2 \qquad \text{(CS)} \quad \boxed{\approx ICP}$$

Ceres Solver¶

Open source

Ceres Solver [1] is an open source C++ library for modeling and solving large, complicated optimization problems. It can be used to solve Non-linear Least Squares problems with bounds constraints and general unconstrained optimization problems. It is a mature, feature rich, and performant library that has been used in production at Google since 2010. For more, see Why?.

What is the loop closure of Cartographer?

- SPA

$$\operatorname*{argmin}_{\Xi^{\mathrm{m}},\Xi^{\mathrm{s}}} \quad \frac{1}{2}\sum_{ij} \rho\big(E^2(\xi_i^{\mathrm{m}}, \xi_j^{\mathrm{s}}; \Sigma_{ij}, \xi_{ij})\big) \qquad \text{(SPA)}$$

where the submap poses $\Xi^{\mathrm{m}} = \{\xi_i^{\mathrm{m}}\}_{i=1,\ldots,m}$ and the scan poses $\Xi^{\mathrm{s}} = \{\xi_j^{\mathrm{s}}\}_{j=1,\ldots,n}$ in the world are optimized given some *constraints*.

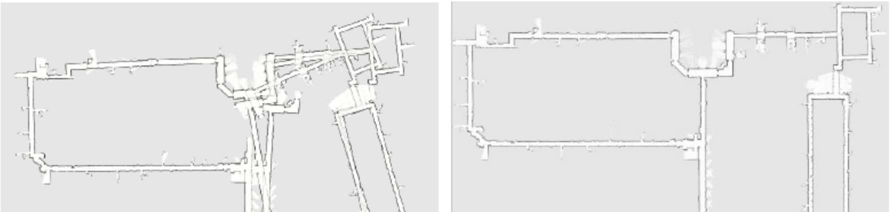
Efficient Sparse Pose Adjustment for 2D Mapping

Kurt Konolige
Willow Garage
Menlo Park, CA 94025
Email: konolige@willowgarage.com

Giorgio Grisetti, Rainer Kümmerle, Wolfram Burgard
University of Freiburg
Freiburg, Germany
Email: grisetti@informatik.uni-freiburg.de

Benson Limketkai, Regis Vincent
SRI International
Menlo Park, CA 94025
Email: regis.vincent@sri.com

***Abstract*— Pose graphs have become a popular representation for solving the simultaneous localization and mapping (SLAM) problem. A pose graph is a set of robot poses connected by nonlinear constraints obtained from observations of features common to nearby poses. Optimizing large pose graphs has**



How to make it fast?

- Branch-and-bound scan matching → efficiency

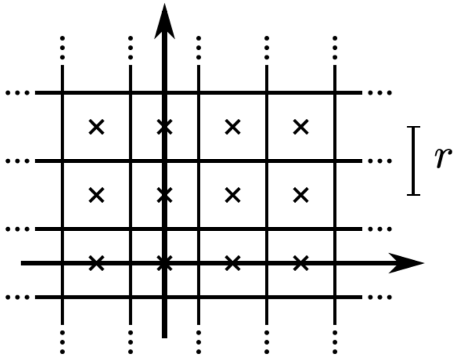
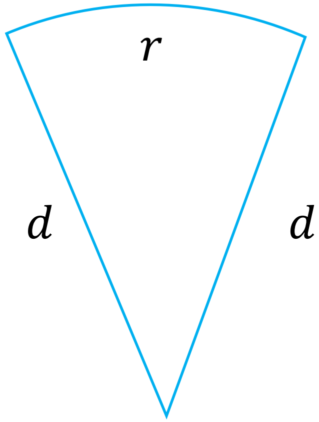


Fig. 1. Grid points and associated pixe

$$\xi^\star = \operatorname*{argmax}_{\xi \in \mathcal{W}} \sum_{k=1}^{K} M_{\text{nearest}}(T_\xi h_k), \qquad \text{(BBS)}$$

$$d_{\max} = \max_{k=1,\ldots,K} \|h_k\|, \qquad (6)$$

$$\delta_\theta = \arccos\left(1 - \frac{r^2}{2d_{\max}^2}\right). \qquad (7)$$



$$w_x = \left\lceil \frac{W_x}{r} \right\rceil, \quad w_y = \left\lceil \frac{W_y}{r} \right\rceil, \quad w_\theta = \left\lceil \frac{W_\theta}{\delta_\theta} \right\rceil. \qquad (8)$$

$$\boxed{\begin{array}{l} W_x = W_y = 7m \\ W_\theta = 30° \end{array}}$$

$$\overline{\mathcal{W}} = \{-w_x, \ldots, w_x\} \times \{-w_y, \ldots, w_y\} \times \{-w_\theta, \ldots, w_\theta\}, \qquad (9)$$

$$\mathcal{W} = \{\xi_0 + (rj_x, rj_y, \delta_\theta j_\theta) : (j_x, j_y, j_\theta) \in \overline{\mathcal{W}}\}. \qquad (10)$$

Algorithm 1 Naive algorithm for (BBS)

$best\_score \leftarrow -\infty$
for $j_x = -w_x$ **to** w_x **do**
  for $j_y = -w_y$ **to** w_y **do**
    for $j_\theta = -w_\theta$ **to** w_θ **do**
      $score \leftarrow \sum_{k=1}^{K} M_{\text{nearest}}(T_{\xi_0+(rj_x,rj_y,\delta_\theta j_\theta)}h_k)$
      if $score > best\_score$ **then**
        $match \leftarrow \xi_0 + (rj_x, rj_y, \delta_\theta j_\theta)$
        $best\_score \leftarrow score$
      end if
    end for
  end for
end for
return $best\_score$ and $match$ when set.

Is it fast enough?

- Depth-first search (DFS)

  ≈scale

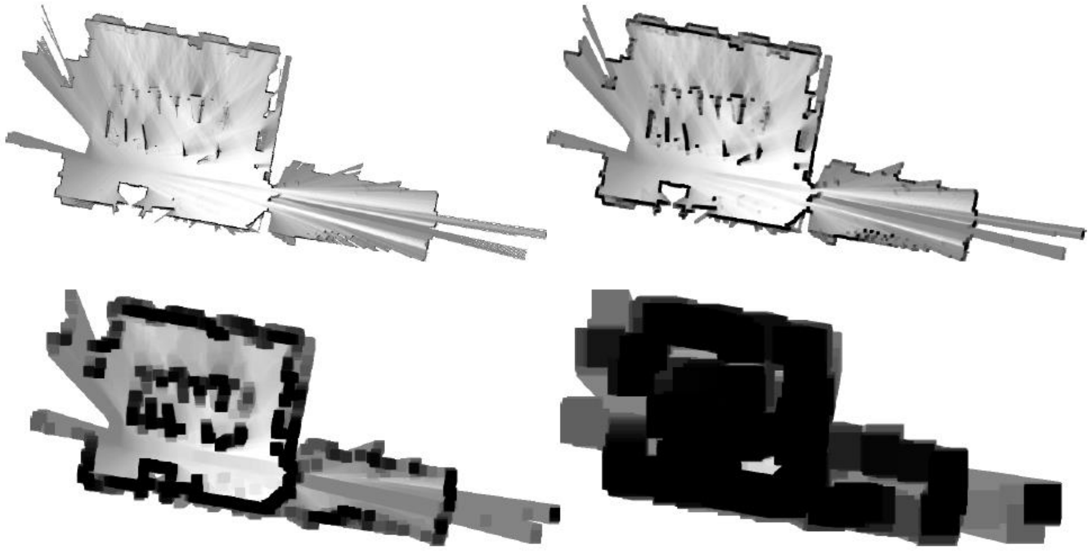
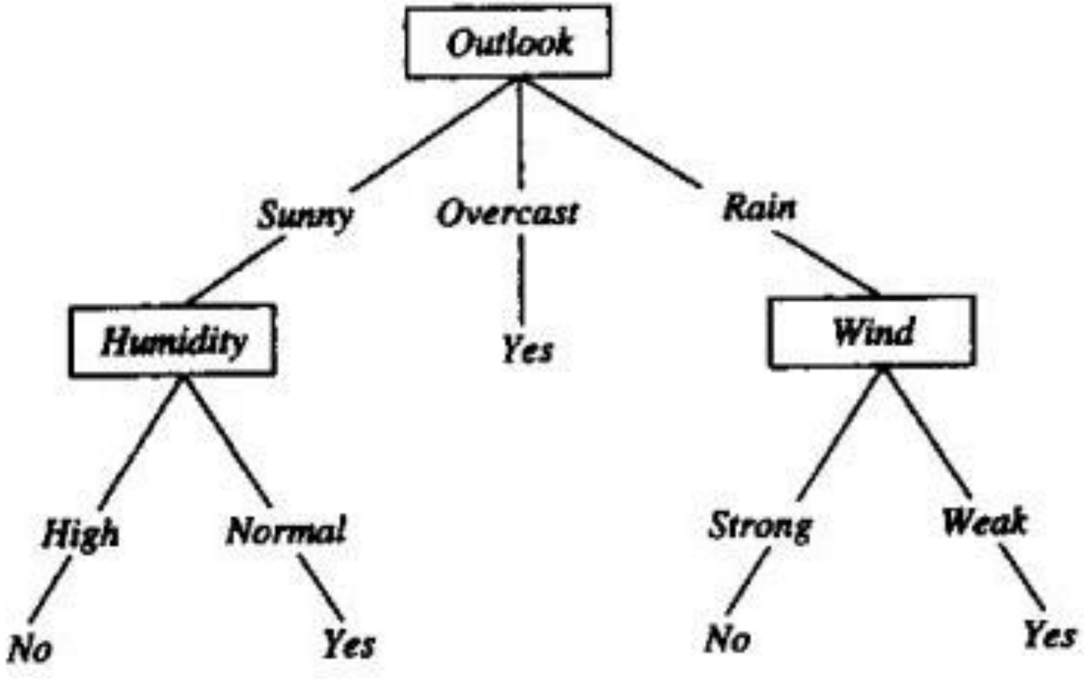


Fig. 3. Precomputed grids of size 1, 4, 16 and 64.



Details?

$$c = (c_x, c_y, c_\theta, c_h) \in \mathbb{Z}^4.$$

$$\overline{\overline{\mathcal{W}}}_c = \left(\left\{ (j_x, j_y) \in \mathbb{Z}^2 : \begin{array}{l} c_x \le j_x < c_x + 2^{c_h} \\ c_y \le j_y < c_y + 2^{c_h} \end{array} \right\} \times \{c_\theta\} \right), \tag{11}$$

$$\overline{\mathcal{W}}_c = \overline{\overline{\mathcal{W}}}_c \cap \overline{\mathcal{W}}. \tag{12}$$

Algorithm 2 Generic branch and bound

$best\_score \leftarrow -\infty$
$\mathcal{C} \leftarrow \mathcal{C}_0$
while $\mathcal{C} \neq \emptyset$ **do**
  Select a node $c \in \mathcal{C}$ and remove it from the set.
  if c is a leaf node **then**
    if $score(c) > best\_score$ **then**
      $solution \leftarrow n$
      $best\_score \leftarrow score(c)$
    end if
  else
    if $score(c) > best\_score$ **then**
      Branch: Split c into nodes $\mathcal{C}_c$.
      $\mathcal{C} \leftarrow \mathcal{C} \cup \mathcal{C}_c$
    else
      Bound.
    end if
  end if
end while
return $best\_score$ and $solution$ when set.

Algorithm 3 DFS branch and bound scan matcher for (BBS)

$best\_score \leftarrow score\_threshold$
Compute and memorize a score for each element in $\mathcal{C}_0$.
Initialize a stack $\mathcal{C}$ with $\mathcal{C}_0$ sorted by score, the maximum score at the top.
while $\mathcal{C}$ is not empty **do**
  Pop c from the stack $\mathcal{C}$.
  if $score(c) > best\_score$ **then**
    if c is a leaf node **then**
      $match \leftarrow \xi_c$
      $best\_score \leftarrow score(c)$
    else
      Branch: Split c into nodes $\mathcal{C}_c$.
      Compute and memorize a score for each element in $\mathcal{C}_c$.
      Push $\mathcal{C}_c$ onto the stack $\mathcal{C}$, sorted by score, the maximum score last.
    end if
  end if
end while
return $best\_score$ and $match$ when set.

Is it effective?

TABLE I
QUANTITATIVE ERRORS WITH REVO LDS

Laser Tape	Cartographer	Error (absolute)	Error (relative)
4.09	4.08	−0.01	−0.2 %
5.40	5.43	+0.03	+0.6 %
8.67	8.74	+0.07	+0.8 %
15.09	15.20	+0.11	+0.7 %
15.12	15.23	+0.11	+0.7 %

Fig. 5. Cartographer map generated using Revo LDS sensor data.

TABLE II

QUANTITATIVE COMPARISON OF ERROR WITH [21]

	Cartographer	GM
Aces		
Absolute translational	0.0375 ± 0.0426	0.044 ± 0.044
Squared translational	0.0032 ± 0.0285	0.004 ± 0.009
Absolute rotational	0.373 ± 0.469	0.4 ± 0.4
Squared rotational	0.359 ± 3.696	0.3 ± 0.8
Intel		
Absolute translational	0.0229 ± 0.0239	0.031 ± 0.026
Squared translational	0.0011 ± 0.0040	0.002 ± 0.004
Absolute rotational	0.453 ± 1.335	1.3 ± 4.7
Squared rotational	1.986 ± 23.988	24.0 ± 166.1
MIT Killian Court		
Absolute translational	0.0395 ± 0.0488	0.050 ± 0.056
Squared translational	0.0039 ± 0.0144	0.006 ± 0.029
Absolute rotational	0.352 ± 0.353	0.5 ± 0.5
Squared rotational	0.248 ± 0.610	0.9 ± 0.9
MIT CSAIL		
Absolute translational	0.0319 ± 0.0363	0.004 ± 0.009
Squared translational	0.0023 ± 0.0099	0.0001 ± 0.0005
Absolute rotational	0.369 ± 0.365	0.05 ± 0.08
Squared rotational	0.270 ± 0.637	0.01 ± 0.04

Graph Mapping

TABLE IV

LOOP CLOSURE PRECISION

Test case	No. of constraints	Precision
Aces	971	98.1 %
Intel	5786	97.2 %
MIT Killian Court	916	93.4 %
MIT CSAIL	1857	94.1 %
Freiburg bldg 79	412	99.8 %
Freiburg hospital	554	77.3 %

TABLE V

PERFORMANCE

Test case	Data duration (s)	Wall clock (s)
Aces	1366	41
Intel	2691	179
MIT Killian Court	7678	190
MIT CSAIL	424	35
Freiburg bldg 79	1061	62
Freiburg hospital	4820	10

Remote Sens. **2013**, *5*, 5871-5906; doi:10.3390/rs5115871

OPEN ACCESS

Remote Sensing

ISSN 2072-4292

www.mdpi.com/journal/remotesensing

Article

Algorithmic Solutions for Computing Precise Maximum Likelihood 3D Point Clouds from Mobile Laser Scanning Platforms

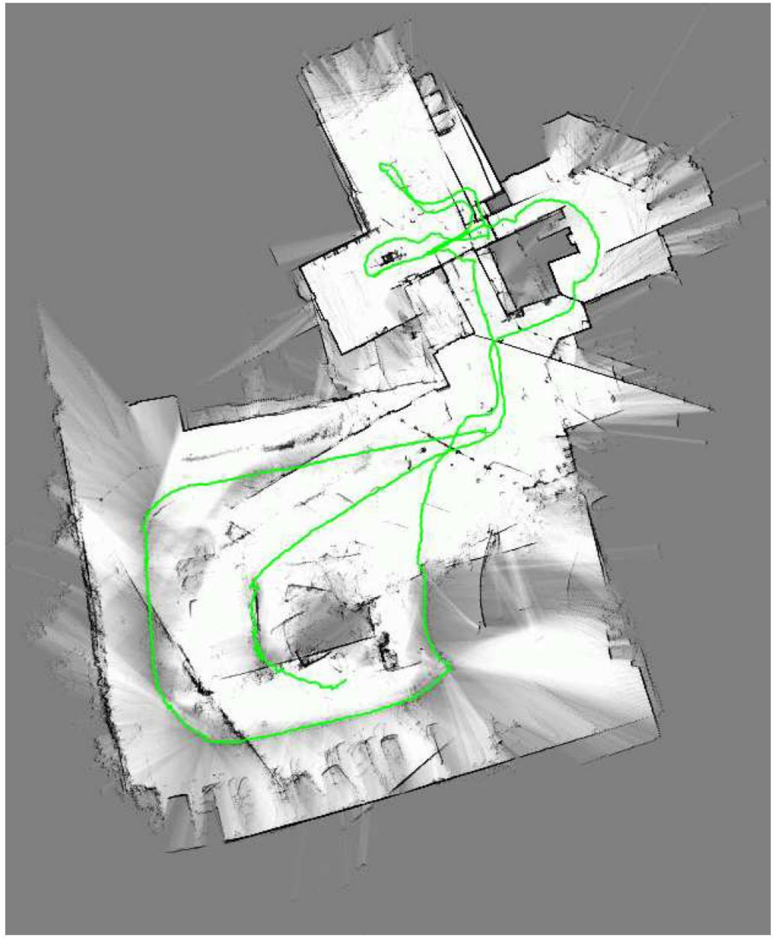
Jan Elseberg [1], Dorit Borrmann [2] and Andreas Nüchter [2,*]

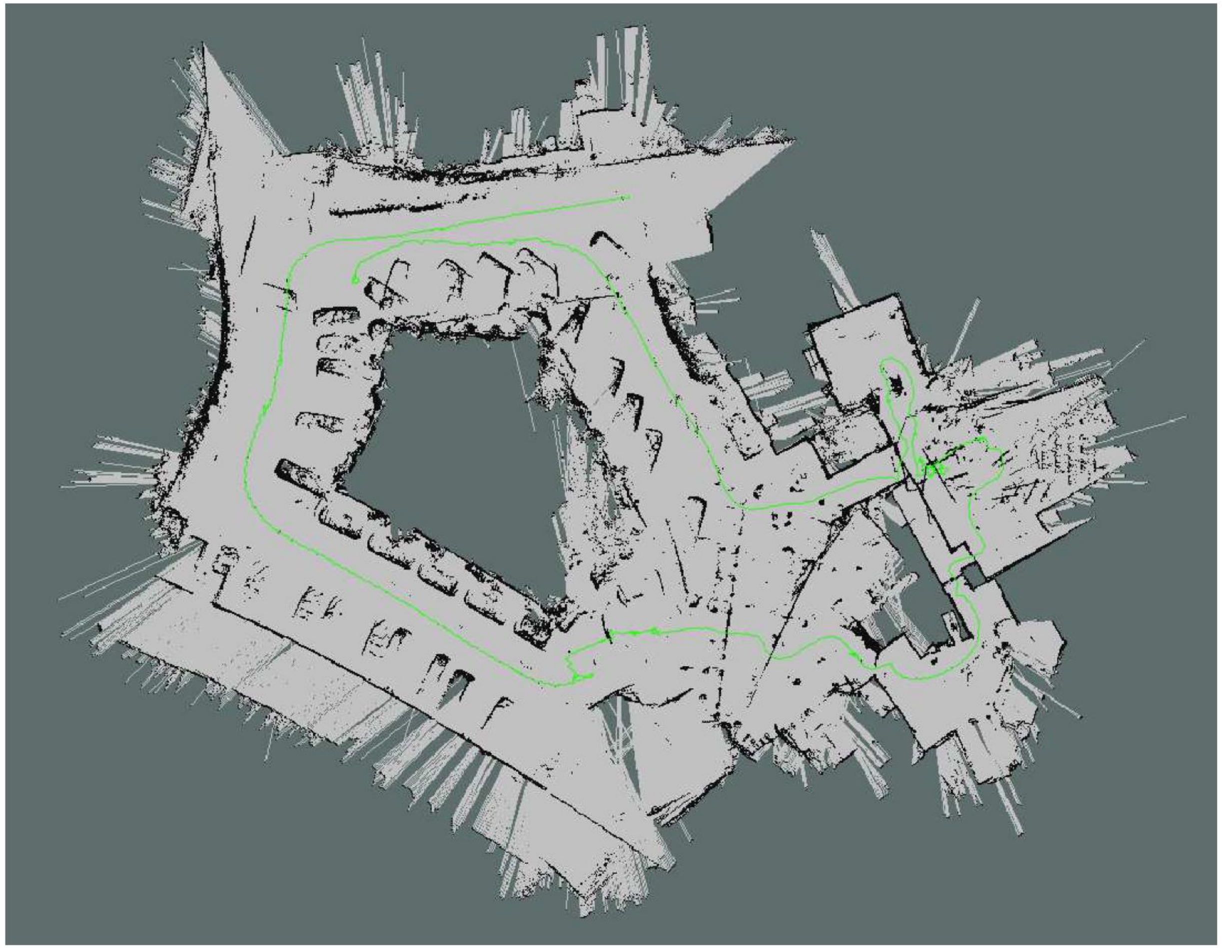
The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume XLII-2/W3, 2017
3D Virtual Reconstruction and Visualization of Complex Architectures, 1–3 March 2017, Nafplio, Greece

IMPROVING GOOGLE'S CARTOGRAPHER 3D MAPPING BY CONTINUOUS-TIME SLAM

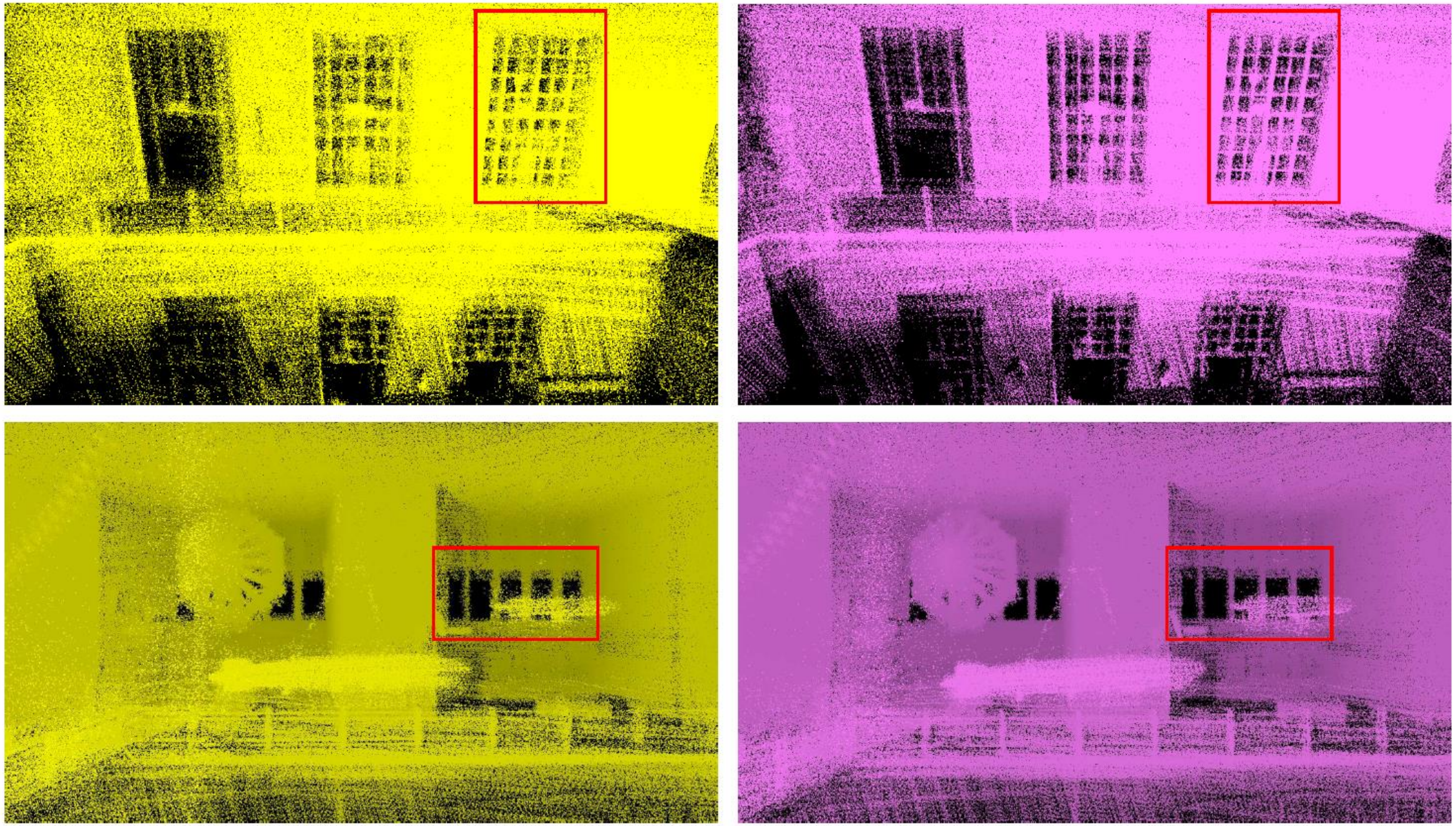
Andreas Nüchter [a,b], Michael Bleier [b], Johannes Schauer [a,b], Peter Janotta [c]

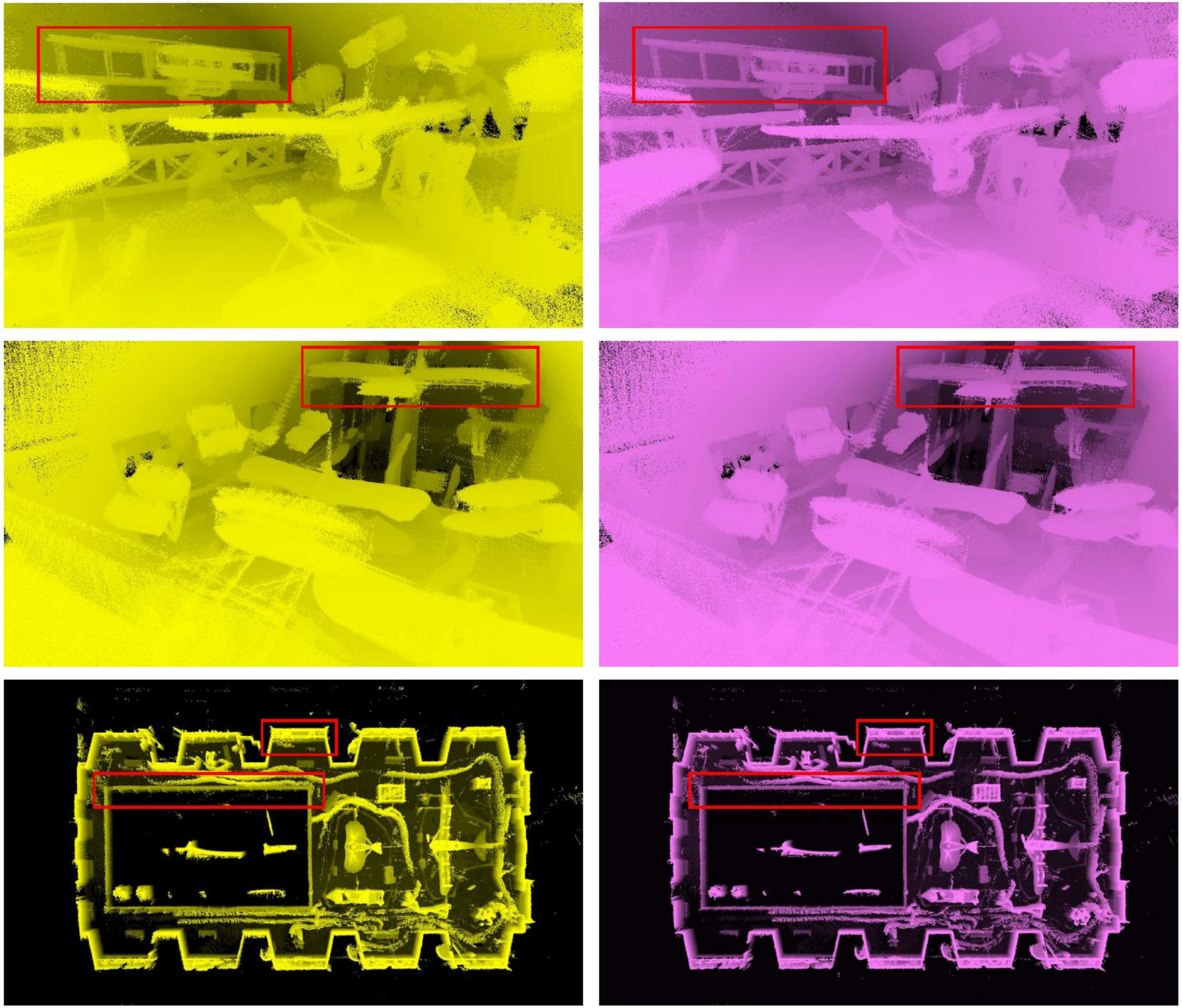
How does it work with 3D point cloud?





How does it work with 3D point cloud?





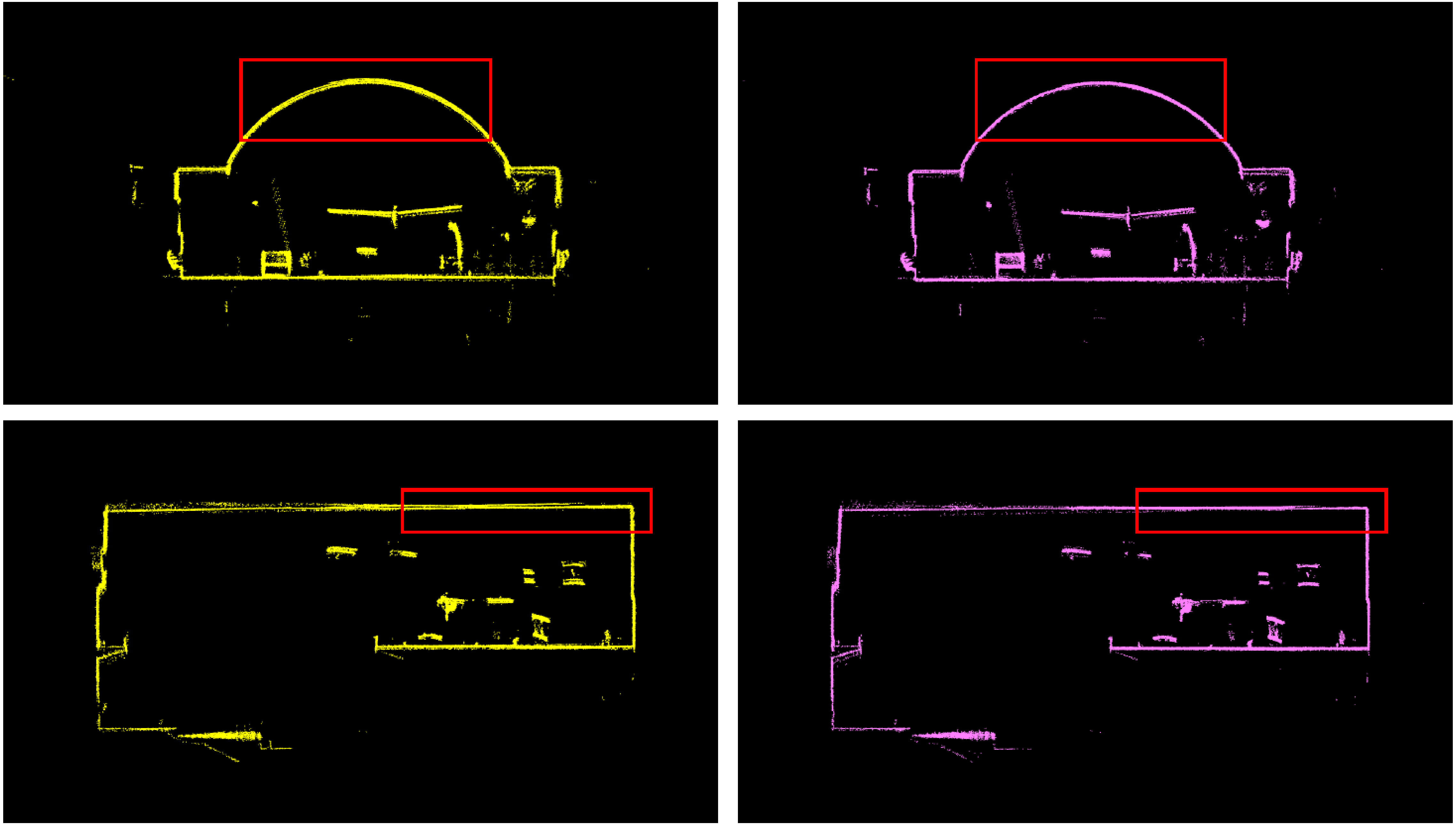


Figure 4: Results of continuous-time SLAM on Google's Cartographer sample data set Deutsches Museum in München. Left: input. Right: output of our solution. Shown are sectional views of the museum hall. Major changes in the point cloud are highlighted in red.

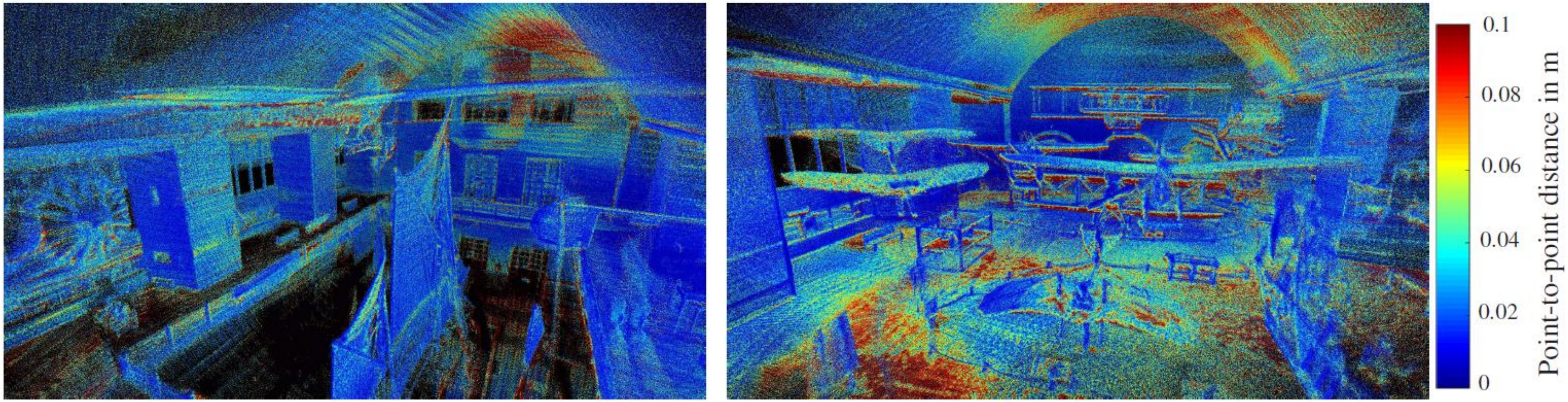


Figure 5: Visualization of the changes in the point cloud. Shown are two views of the optimized 3D point cloud, colored with the difference to the result from Google's Cartographer.

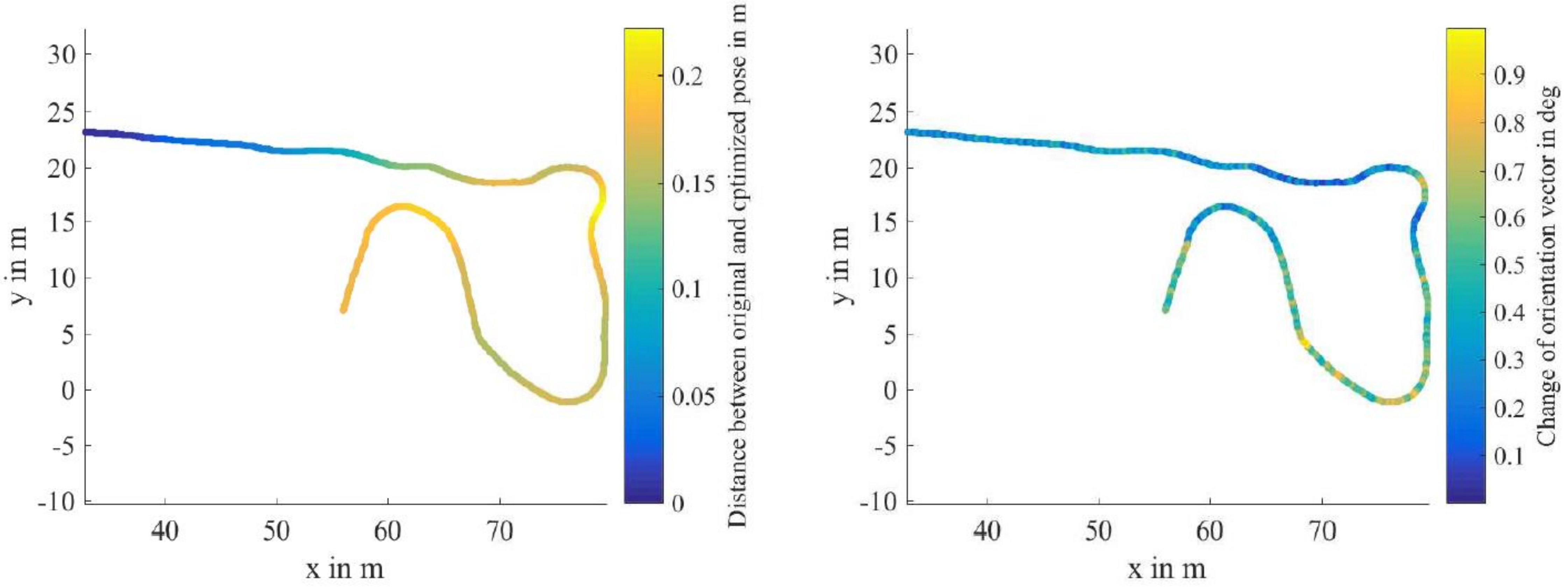


Figure 6: Visualization of the changes in the trajectory computed by our method to bootstraped trajectory. Left: distance. Right: orientation