3D Registration and Shape Fitting

3D Vision

University of Illinois

Derek Hoiem

Agenda

- Overview of shape fitting and registration
- Efficient RANSAC for Point-Cloud Shape Detection
- TEASER: Fast and Certifiable Point Cloud Registration

Fitting: find the parameters of a model that best fit the data

Registration: find the parameters of the transformation that best align matched points

Shape fitting and registration have many applications

Shape fitting

- Simplify mesh or remove noise from points
- Detect potential surfaces to fill in missing points
- Extract structure for matching to drawings or other representations

Registration

- Align point clouds of a building site captured on two dates for comparison
- Get a complete scan from several partial scans
- Determine the relative pose of two point clouds (e.g. for SLAM loop closure)
- Find an object with known shape in the scene

Fitting and Registration: Design Challenges

- Design a suitable **goodness of fit** measure
  - Similarity should reflect application goals
  - Encode robustness to outliers and noise

- Design an **optimization** method
  - Avoid local optima
  - Find best parameters quickly

Fitting and Registration: Methods

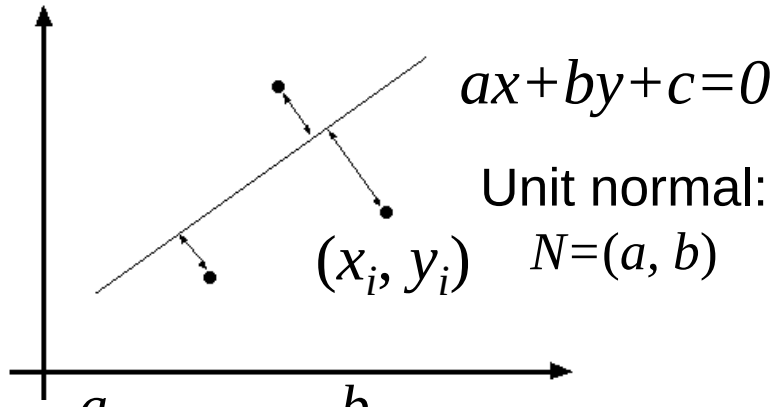
- Global optimization / Search for parameters
  - Least squares fit
  - Robust least squares
  - Iterative closest point (ICP)

- Hypothesize and test
  - Generalized Hough transform
  - RANSAC

Total least squares

Find (a, b, c) to minimize the sum of squared perpendicular distances

$$E = \sum_{i=1}^{n} (a x_i + b y_i + c)^2$$



$$\frac{\partial E}{\partial c} = \sum_{i=1}^{n} 2(a x_i + b y_i + c) = 0$$

$$c = -\frac{a}{n}\sum_{i=1}^{n} x_i - \frac{b}{n}\sum_{i=1}^{n} y_i = -a\bar{x} - b\bar{y}$$

$$E = \sum_{i=1}^{n} (a(x_i - \bar{x}) + b(y_i - \bar{y}))^2 = \left\| \begin{bmatrix} x_1 - \bar{x} & y_1 - \bar{y} \\ \vdots & \vdots \\ x_n - \bar{x} & y_n - \bar{y} \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} \right\|^2 = \mathbf{p}^T \mathbf{A}^T \mathbf{A} \mathbf{p}$$

$$\text{minimize } \mathbf{p}^T \mathbf{A}^T \mathbf{A} \mathbf{p} \quad \text{s.t. } \mathbf{p}^T \mathbf{p} = 1 \quad \Rightarrow \quad \text{minimize } \frac{\mathbf{p}^T \mathbf{A}^T \mathbf{A} \mathbf{p}}{\mathbf{p}^T \mathbf{p}}$$

Solution is eigenvector corresponding to smallest eigenvalue of A^TA

See details on Raleigh Quotient: http://en.wikipedia.org/wiki/Rayleigh_quotient

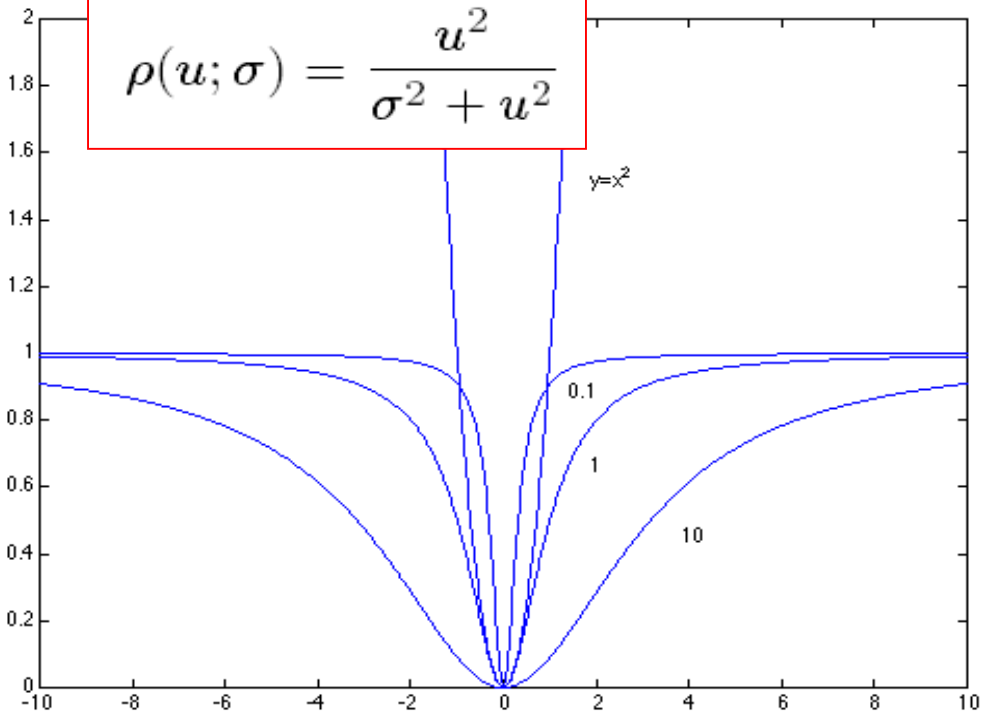
Robust least squares (to deal with outliers)

General approach:

minimize $\sum_{\mathrm{i}} \rho\left(\mathrm{u}_{\mathrm{i}}\left(\mathrm{x}_{\mathrm{i}}, \boldsymbol{\theta}\right); \boldsymbol{\sigma}\right)$ $\quad u^2 = \sum_{i=1}^{n} (y_i - mx_i - b)^2$

$u_i(x_i, \theta)$ – residual of ith point w.r.t. model parameters θ
ρ – robust function with scale parameter σ

$$\rho(u;\sigma) = \frac{u^2}{\sigma^2 + u^2}$$



The robust function ρ

- Favors a configuration with small residuals
- Constant penalty for large residuals

Slide from S. Savarese

Robust Estimator

1. Initialize: e.g., choose θ by least squares fit and
   $$\sigma = 1.5 \cdot \text{median}(error)$$
2. Choose params to minimize: $$\sum_i \frac{error(\theta, data_i)^2}{\sigma^2 + error(\theta, data_i)^2}$$
   – E.g., numerical optimization
3. Compute new $$\sigma = 1.5 \cdot \text{median}(error)$$
4. Repeat (2) and (3) until convergence

Hypothesize and test

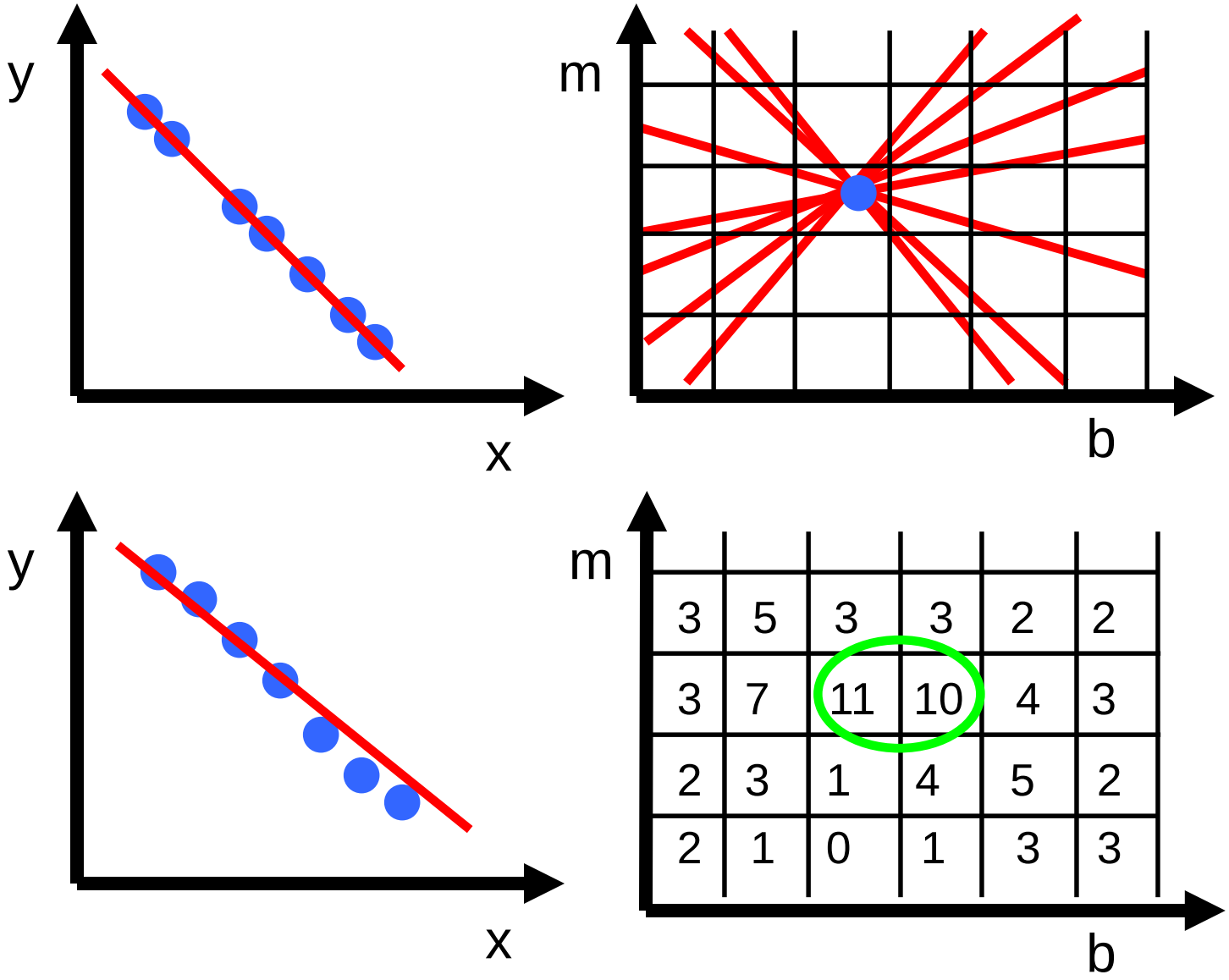
1. Propose parameters
   - Try all possible
   - Each point votes for all consistent parameters
   - Repeatedly sample enough points to solve for parameters

2. Score the given parameters
   - Number of consistent points, possibly weighted by distance

3. Choose from among the set of parameters
   - Global or local maximum of scores

4. Possibly refine parameters using inliers

Hough transform

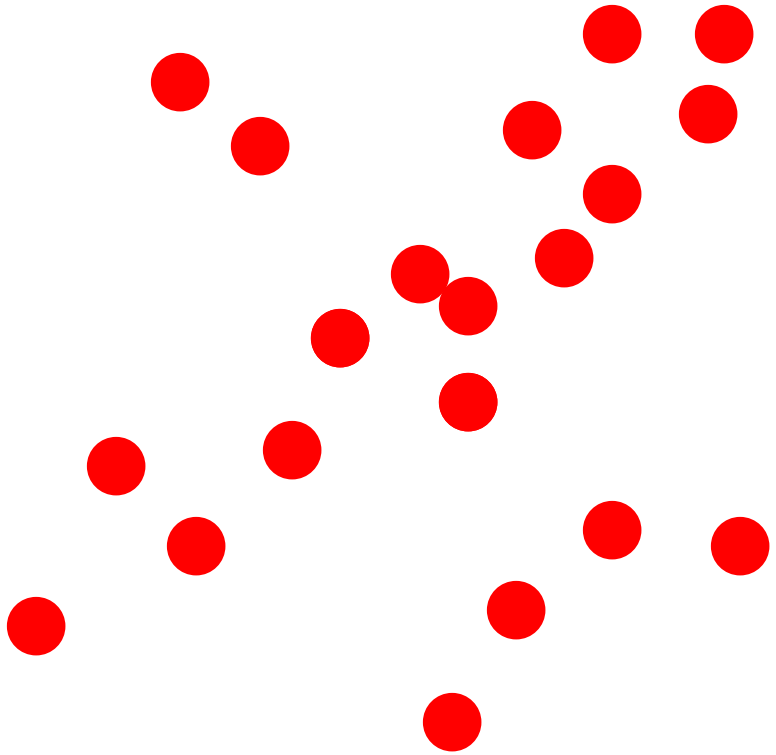


Slide from S. Savarese

RANSAC

(RANdom SAmple Consensus) :

Fischler & Bolles in ‘81.



Algorithm:

1. **Sample** (randomly) the number of points required to fit the model
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

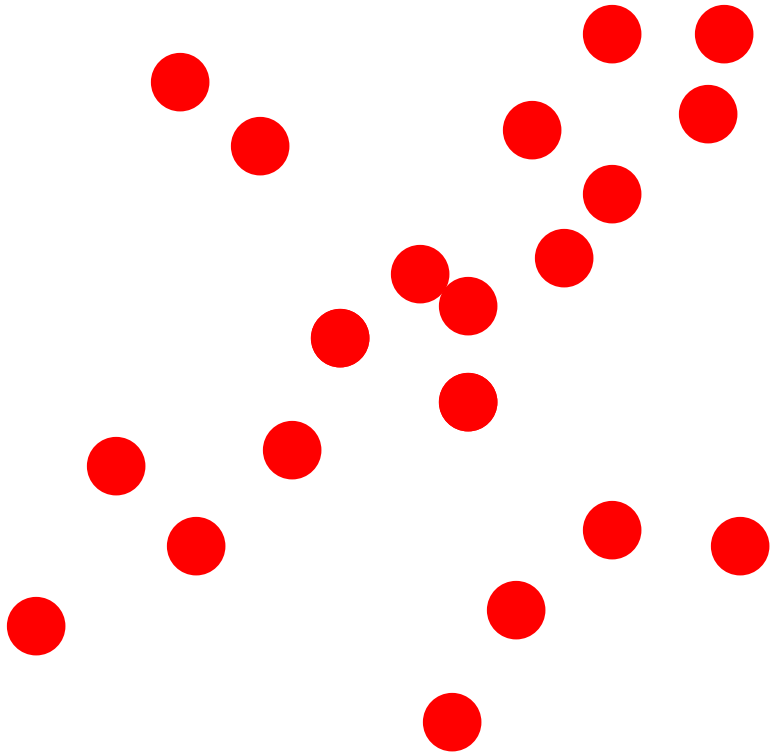
Repeat 1-3 until the best model is found with high confidence

Source: Savarese

RANSAC

(RANdom SAmple Consensus) :

Fischler & Bolles in ‘81.



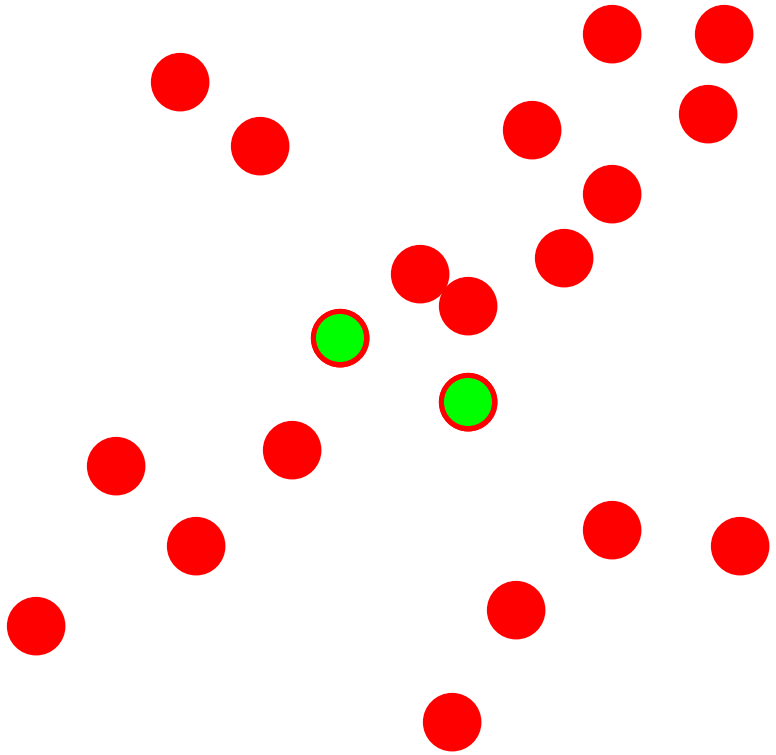
Algorithm:

1. **Sample** (randomly) the number of points required to fit the model
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

RANSAC

Line fitting example



Algorithm:

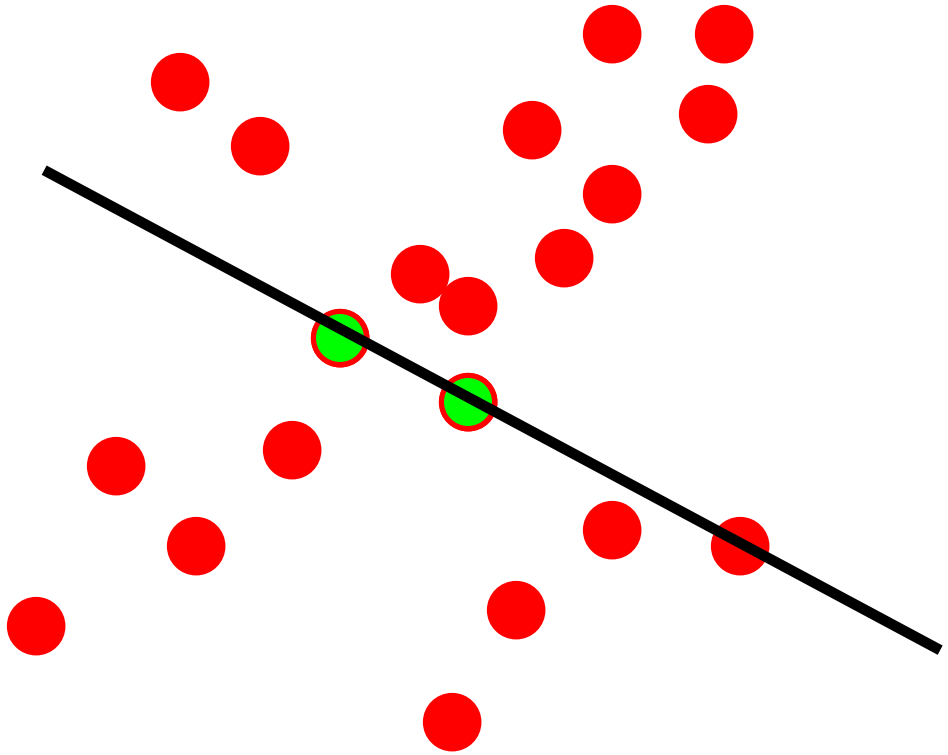
1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

Illustration by Savarese

RANSAC

Line fitting example



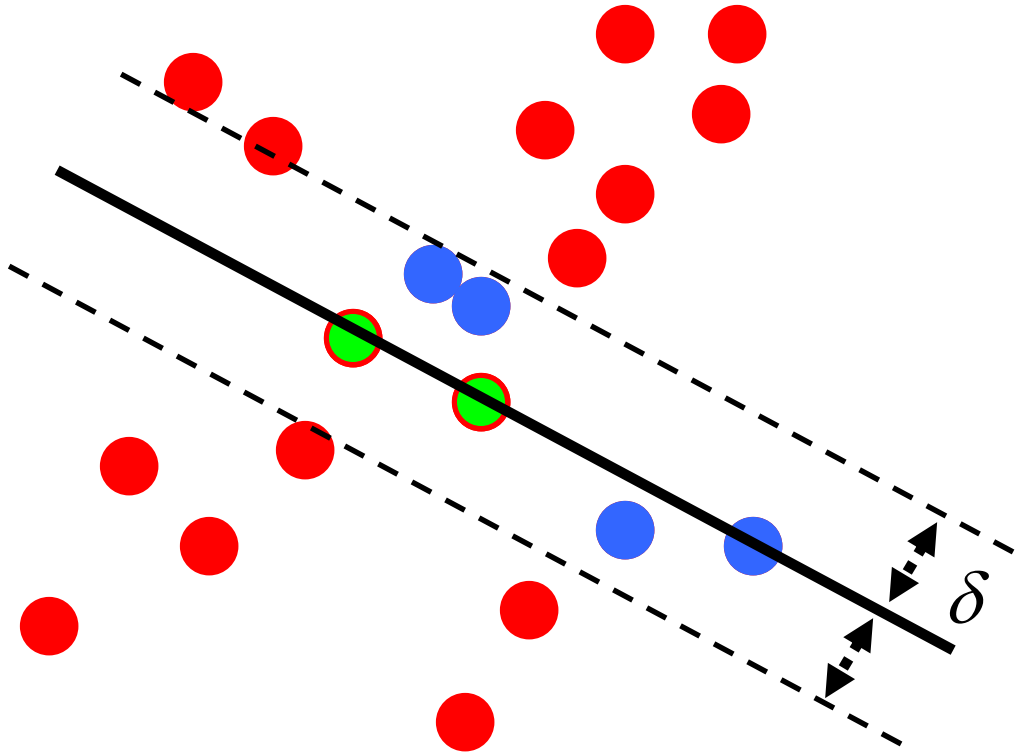
Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

RANSAC

Line fitting example



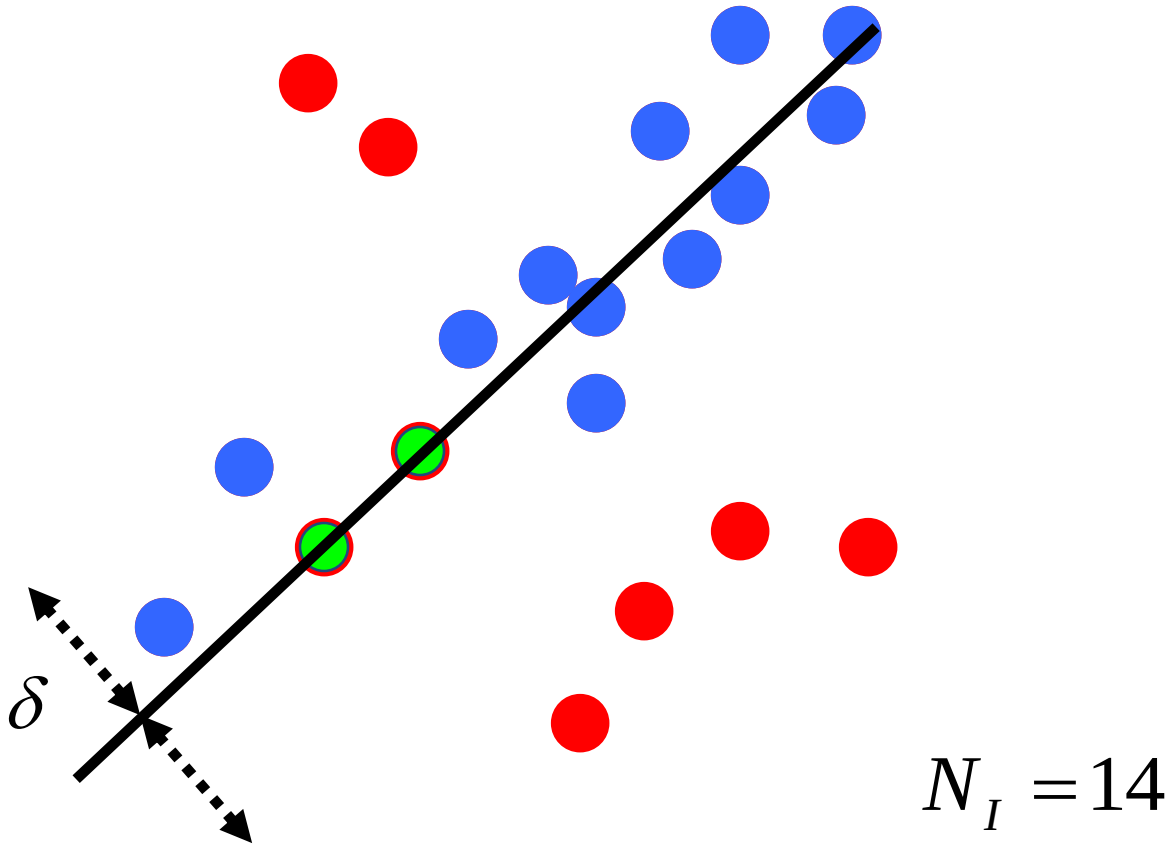
$$N_I = 6$$

Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

RANSAC



$N_I = 14$

Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

How to choose parameters?

- Number of samples *N*
  - Choose *N* so that, with probability *p*, at least one random sample is free from outliers (e.g. *p*=0.99) (outlier ratio: *e*)
- Number of sampled points *s*
  - Minimum number needed to fit the model
- Distance threshold δ
  - Choose δ so that a good point with noise is likely (e.g., prob=0.95) within threshold
  - Zero-mean Gaussian noise with std. dev. σ: $t^2=3.84\sigma^2$

$$N = \log(1-p)/\log\left(1-(1-e)^s\right)$$

	proportion of outliers *e*						
s	5%	10%	20%	25%	30%	40%	50%
2	2	3	5	6	7	11	17
3	3	4	7	9	11	19	35
4	3	5	9	13	17	34	72
5	4	6	12	17	26	57	146
6	4	7	16	24	37	97	293
7	4	8	20	33	54	163	588
8	5	9	26	44	78	272	1177

modified from M. Pollefeys

Iterative Closest Points (ICP) Algorithm

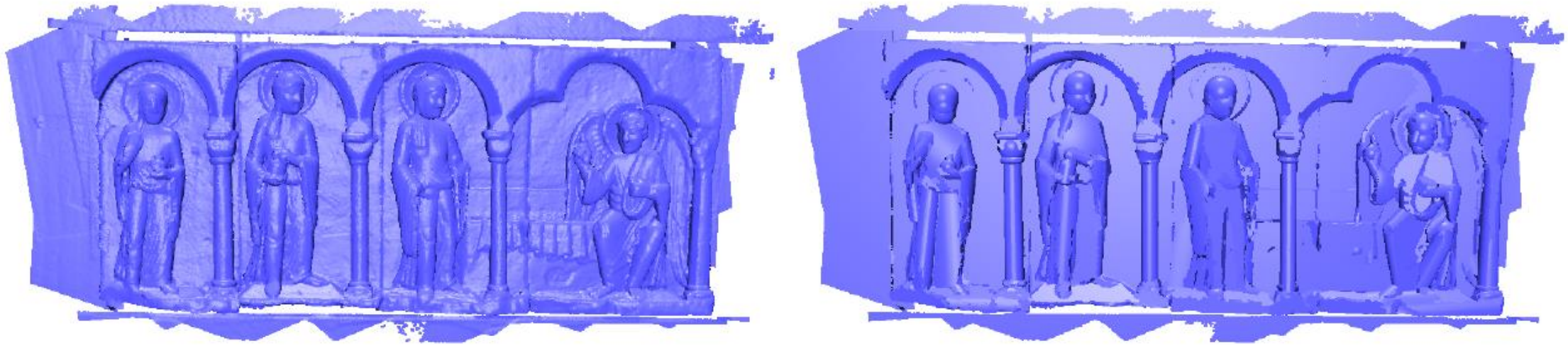
Goal: estimate transform between two dense sets of points without correspondences

1. **Initialize** transformation (e.g., compute difference in means and scale)
2. **Assign** each point in {Set 1} to its nearest neighbor in {Set 2}
3. **Estimate** transformation parameters
   - e.g., least squares or robust least squares
4. **Transform** the points in {Set 1} using estimated parameters
5. **Repeat** steps 2-4 until change is very small

Algorithm Summary

- Least Squares Fit
  - closed form solution
  - robust to noise
  - not robust to outliers
- Robust Least Squares
  - improves robustness to noise
  - requires iterative optimization
- Hough transform
  - robust to noise and outliers
  - can fit multiple models
  - only works for a few parameters (1-3 typically)
- RANSAC
  - robust to noise and outliers
  - works with a moderate number of parameters (e.g, 1-8)
- Iterative Closest Point (ICP)
  - Does not require initial correspondences
  - Needs good initial solution, gets stuck easily

Efficient RANSAC for Point Cloud Shape Detection (Schnabel et al. 2007)



(a) Original (b) Approximation

- Fit planes and shapes to 3D points, with partial extent
- Scale to millions of points

Efficient RANSAC Overview

Input: orientated 3D point set, empty sets of shapes and candidates

1. **Generate candidates** for all shape types by sampling minimum subsets
2. **Score each candidate** and add to candidate set
3. **Select best candidate** *m* if it is likely that no better candidate exists
   a. Add candidate *m* to shape set
   b. Remove points that fit *m* from point set
   c. Remove candidates that overlap with *m*
4. Return to 1 until there aren't any good candidates

Shape estimation from 3 sampled oriented points

- Plane
  - Fit to 3 unoriented points
  - Quick check: point/plane normal similarity
- Sphere
  - Fit to 2 oriented points
  - Find center pt closest to two rays; radius is average distance from center to points
  - Quick validate: point/sphere normal similarity, point/sphere surface similarity
- Cylinder
  - Fit to 2 oriented points
  - Axis is n1 x n2; find center similar to sphere in plane normal to axis; radius set by distance of p1 to center
  - Quick validate: normals and positions
- Cone
  - Fit to 3 oriented points
  - Apex is intersection of 3 planes defined by oriented points; axis is normal of plane formed by 3 unit-normalized directions from apex to points; opening angle is average angle from axis of apex to points

Runtime Complexity

Efficiency goals
- Maximize $P(n)$, prob of generating good candidate
- Minimize evaluation cost C

- Suppose
  - A point cloud has N points
  - n of the points fit a particular shape
  - The shape can be estimated from k points
- Probability of fitting the shape in one sampling

$$P(n) = \binom{n}{k} \Big/ \binom{N}{k} \approx \left(\frac{n}{N}\right)^k$$

- Probability of fitting the shape with s candidates

$$P(n,s) = 1 - (1 - P(n))^s$$

- T candidates needed to detect best shape of at least size n with probability $P(n,T) \geq p_t$

$$T \geq \frac{\ln(1-p_t)}{\ln(1-P(n))} \qquad T \approx \frac{-\ln(1-p_t)}{P(n)} \qquad O(TC) = O(\frac{C}{P(n)})$$

Evaluation cost

Sampling Strategy

- Local sampling with Octree
  - Nearby points are more likely to be on the same shape surface
  - Sample first point freely
  - Sample level along octree
  - Sample other two points within corresponding cell

$$P_{local}(n) = \frac{n}{Nd2^{k-1}}$$

Assumes at least one octree cell containing each point has at least 50% of points that fit shape; octree is depth *d*

- Example of shape with 1,066 points within set of 341,547 points
  - Uniform sampling: need 151,522,829 candidates to achieve 99% probability of good sample
  - Octree sampling: need 64,929 candidates

Figure 2: *A small cylinder that has been detected by our method. The shape consists of 1066 points and was detected among 341,587 points. That corresponds to a relative size of* 1/3000.

Improved octree level sampling

- Learn data-dependent prior for whether to sample close or distant points
- Initialize probability of sampling each level to $\frac{1}{d}$
- Keep track of sum of scores σ_l of each level l
- After testing a given number of candidates, assign probability of sampling level l

0.9

$$\hat{P}_l = x\frac{\sigma_l}{wP_l} + (1-x)\frac{1}{d} \qquad w = \sum_{i=1}^{d} \frac{\sigma}{P_i}$$

Normalizing sum

Uniform probability

Sampling: termination

- Stop sampling when there is less than 1% chance that a shape exists with more points that the best shape so far

Shape score

- Score = # inliers

- Points are inliers if distance to surface is less than ϵ and normal is less than surface normal by less than α
  - Use octree to find points near surface efficiently

- Keep only inliers that are in connected components
  - Create a low resolution bitmap over the surface of the shape
  - Project points onto that surface
  - Extract connected components

Scoring with random subsets

- Split entire point set into random subsets
- Compute score on subset and infer score confidence interval on larger set
- If score interval overlaps with score interval of best candidate, compute each with an extra subset, until there is no overlap

Refitting

- Perform least squares fit on inliers and check again for inliers

Summary of speed-ups

- Fast sample rejection
- Local sampling
- Improved level sampling
- Scoring with subsets

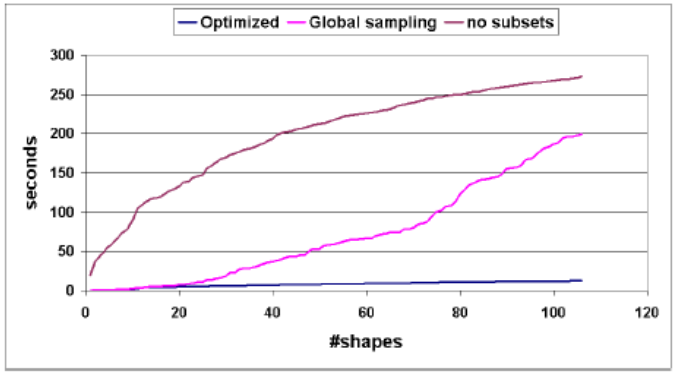
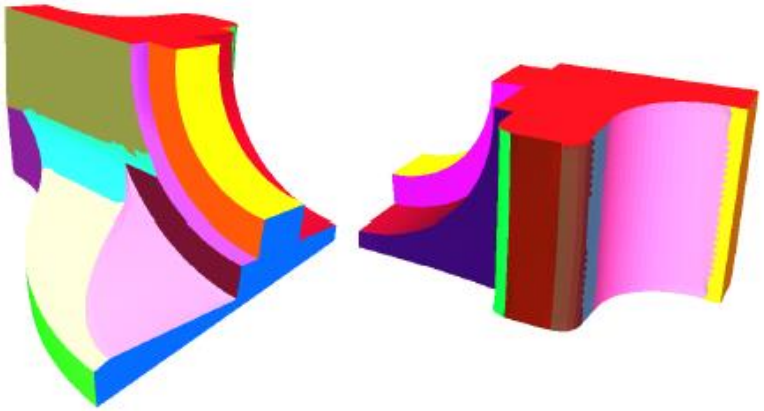
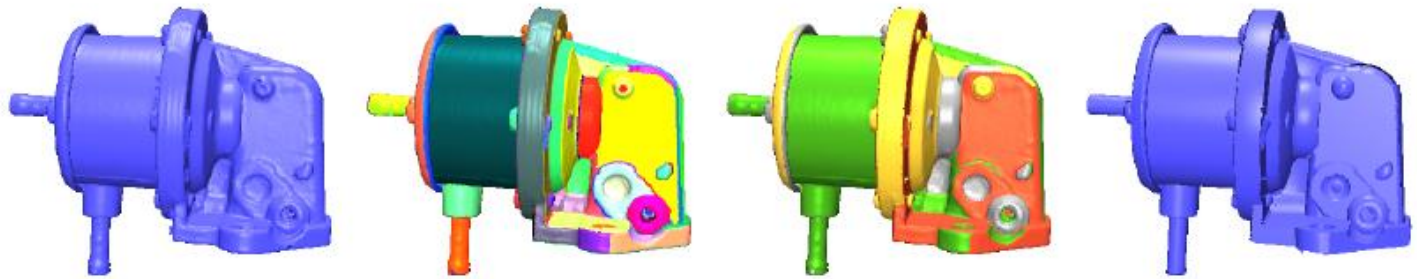


Figure 4: *The chart shows the times of detection of the shapes found in the oil pump model when either subset evaluation or the localized sampling is disabled. For comparison also the timings of the fully optimized version are plotted. Total runtime for the version without subsets was 272.5s, 199.1s without local sampling and 12.3s with both optimizations activated.*



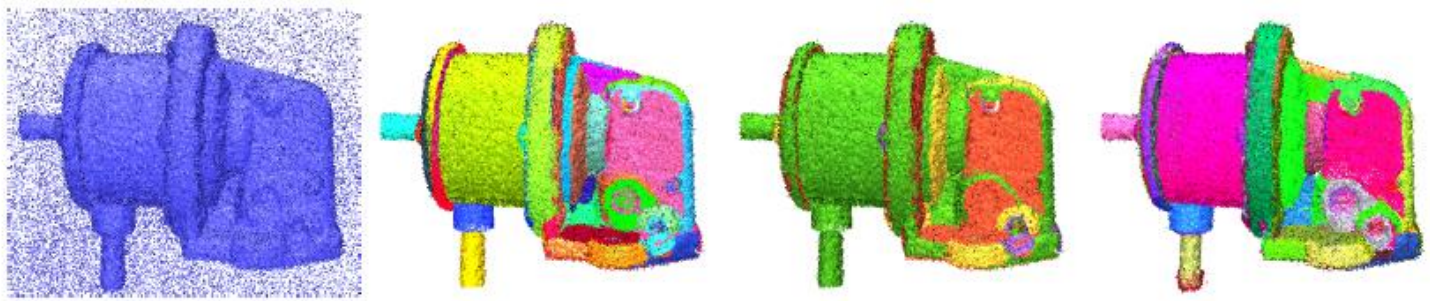
model	$\lvert\mathcal{P}\rvert$	ε	α	τ	$\lvert\Psi\rvert$	$\lvert\mathcal{R}\rvert$	sec
fandisk	12k	0.01	10	50	24	38	0.57
rocker arm	40k	0.003	20	50	73	1k	6.5
carter	546k	0.001	20	200	138	47k	29.1
rolling stage	606k	0.003	20	300	61	16k	15.1
oil pump	542k	0.0015	30	100	202	15k	30.9
master cyl.	418k	0.003	35	300	37	7k	12.1
house	379k	0.002	20	100	130	19k	10.7
church	1,802k	0.002	20	1000	160	690k	40.7
choir screen	1,922k	0.002	20	4,000	81	543k	20.8
				500	372	236k	61.5

P # total pts
ϵ Distance threshold
α Angle threshold (deg)
τ Min pts for shape
Ψ # shape found
R # remaining points



(a) Original (b) Random colors (c) Colored by type (d) Bitmaps

Figure 7: *a) The original scanned model with ca. 500k points. b) Points belonging to shapes in random colors. c) Points of the shapes colored according to the type of the shape they have been assigned to: planes are red, cylinders green, spheres yellow, cones purple and tori are grey. No remaining points are shown. d) The bitmaps constructed for connected component computations provide a rough reconstruction of the object.*



(a) Noisy original (b) Random colors (c) Colored by type (d) Random colors

Figure 8: *a) Distorted model with Gaussian noise and outliers b)-c) Results of the detection on the model with Gaussian noise but without added outliers. d) In addition to the Gaussian noise, 10% outliers were added (see a)).*

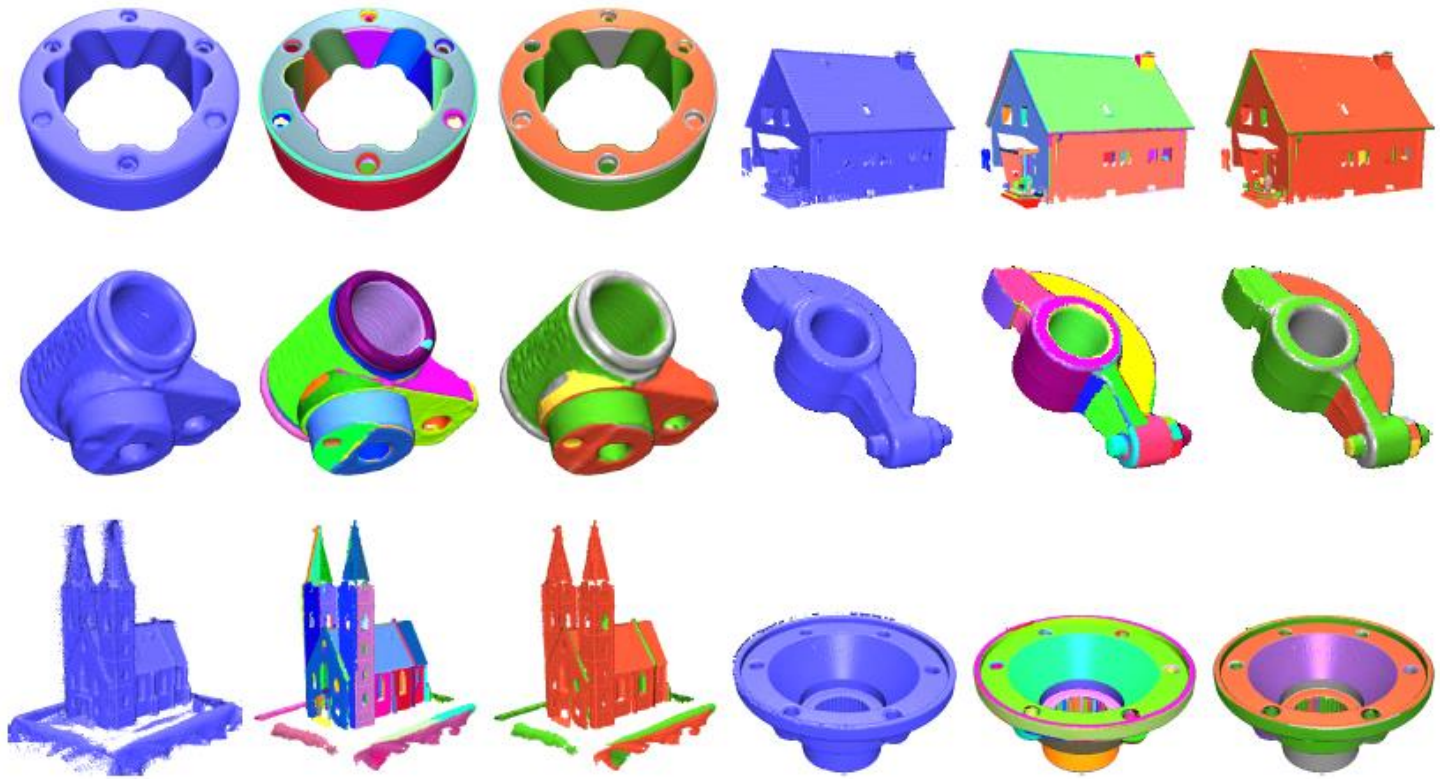


Figure 9: *First column: Original point-clouds. Second column: Shapes colored randomly. Last column: Shapes colored by type as in Fig. 7. Models are from top to bottom and left to right: rolling stage, house, master cylinder, rocker arm, church, and carter. For parameters and timings see table 1.*

Efficient RANSAC on CGAL

- https://doc.cgal.org/latest/Shape_detection/index.html

- Works well when points are uniformly distributed on the surface, like a laser scan

- Connected components part may need to be modified for sparse/MVS points due to gaps in reconstruction

TEASER: Fast and Certifiable Point Cloud Registration (Yang et al. 2020)

Algorithm 1: *Truncated least squares Estimation And SEmidefinite Relaxation* (TEASER).

Input: points $(\boldsymbol{a}_i, \boldsymbol{b}_i)$ and bounds β_i $(i = 1, \ldots, N)$, threshold $\bar{c}^2$ (default: $\bar{c}^2 = 1$), graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ (default: $\mathcal{G}$ describes the complete graph);

Output: $\hat{s}, \hat{\boldsymbol{R}}, \hat{\boldsymbol{t}}$;

% Compute TIM and TRIM

$\bar{\boldsymbol{b}}_{ij} = \boldsymbol{b}_j - \boldsymbol{b}_i$, $\bar{\boldsymbol{a}}_{ij} = \boldsymbol{a}_j - \boldsymbol{a}_i$, $\delta_{ij} = \beta_i + \beta_j$, $\forall (i,j) \in \mathcal{E}$

$s_{ij} = \frac{\|\bar{\boldsymbol{b}}_{ij}\|}{\|\bar{\boldsymbol{a}}_{ij}\|}$, $\alpha_{ij} = \frac{\delta_{ij}}{\|\bar{\boldsymbol{a}}_{ij}\|}$, $\forall (i,j) \in \mathcal{E}$

% Decoupled estimation of $s, \boldsymbol{R}, \boldsymbol{t}$

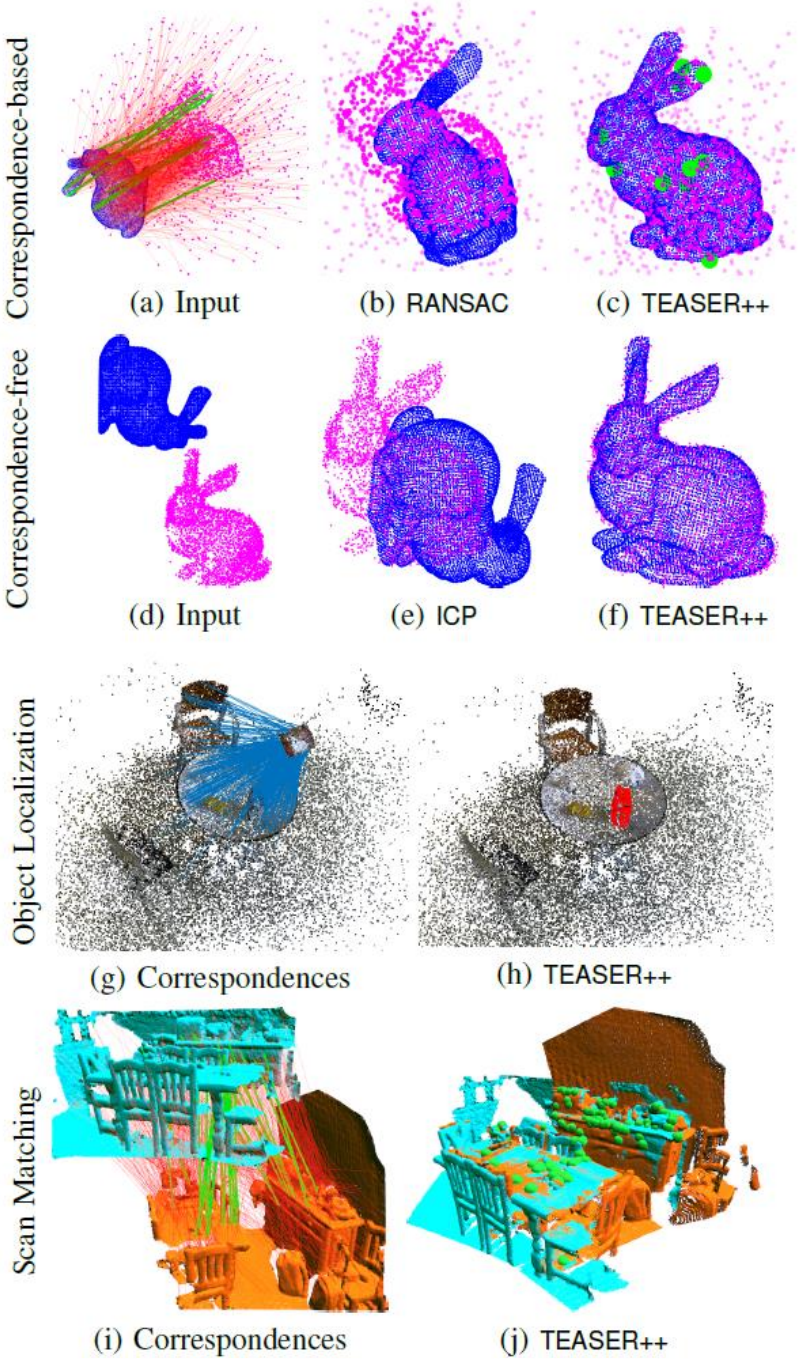
$\hat{s} = \texttt{estimate\_s}(\{s_{ij}, \alpha_{ij} : \forall (i,j) \in \mathcal{E}\}, \bar{c}^2)$

$\mathcal{G}'(\mathcal{V}', \mathcal{E}'') = \texttt{maxClique}(\mathcal{G}(\mathcal{V}, \mathcal{E}'))$ % prune outliers

$\hat{\boldsymbol{R}} = \texttt{estimate\_R}(\{\bar{\boldsymbol{a}}_{ij}, \bar{\boldsymbol{b}}_{ij}, \delta_{ij} : \forall (i,j) \in \mathcal{E}''\}, \bar{c}^2, \hat{s})$

$\hat{\boldsymbol{t}} = \texttt{estimate\_t}(\{\boldsymbol{a}_i, \boldsymbol{b}_i, \beta_i : i \in \mathcal{V}'\}, \bar{c}^2, \hat{s}, \hat{\boldsymbol{R}})$

return: $\hat{s}, \hat{\boldsymbol{R}}, \hat{\boldsymbol{t}}$



Problem formulation

- Given corresponding points (a_i, b_i), solve for translation t, rotation R, and scale s that minimize truncated least squares cost function
- Equivalent to maximizing inliers with a distance-based weighting
- Difficult optimization problem

$$\min_{s>0, \boldsymbol{R}\in \mathrm{SO}(3), \boldsymbol{t}\in\mathbb{R}^3} \sum_{i=1}^{N} \min\left(\frac{1}{\beta_i^2}\left\|\boldsymbol{b}_i - s\boldsymbol{R}\boldsymbol{a}_i - \boldsymbol{t}\right\|^2,\ \bar{c}^2\right)$$

Optimization approach

- Hypothesize pairs of correspondences (a_i, b_i) and (a_j, b_j)
- If both correspondences are inliers, they can vote for scale
- Given scale, it's much easier to solve for rotation
- Given scale and rotation, the pairs can vote for translation

TIM and TRIM

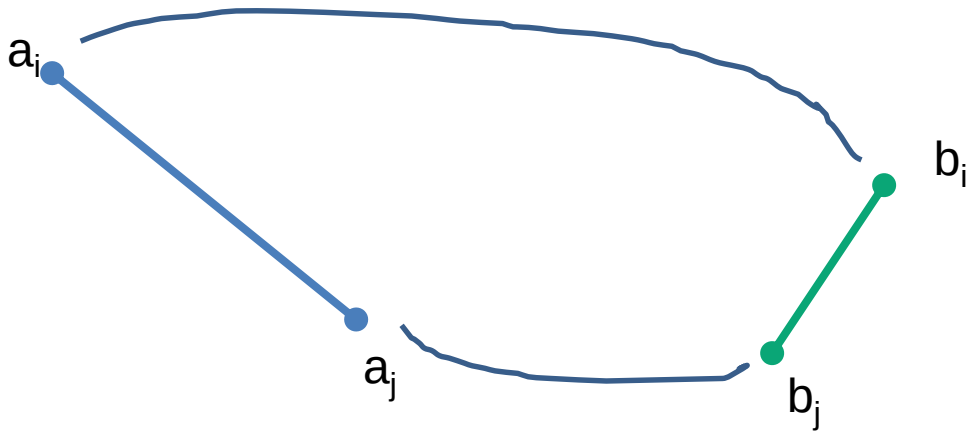
- Translation-rotation-invariant measures (TRIM)

$$\bar{\boldsymbol{a}}_{ij} \doteq \boldsymbol{a}_j - \boldsymbol{a}_i \qquad \bar{\boldsymbol{b}}_{ij} \doteq \boldsymbol{b}_j - \boldsymbol{b}_i \qquad s_{ij} \doteq \frac{\|\bar{\boldsymbol{b}}_{ij}\|}{\|\bar{\boldsymbol{a}}_{ij}\|}$$

- Translation-invariant measures (TIM)

$$\bar{\boldsymbol{b}}_{ij} = s\boldsymbol{R}\bar{\boldsymbol{a}}_{ij} + \boldsymbol{o}_{ij} + \boldsymbol{\epsilon}_{ij}$$

0 if i and j are inliers



Scale estimation

- Get consensus estimate for scale from TRIM

$$\hat{s} = \arg\min_{s} \sum_{k=1}^{K} \min\left(\frac{(s - s_k)^2}{\alpha_k^2} \, , \, \bar{c}^2 \right)$$

tolerance

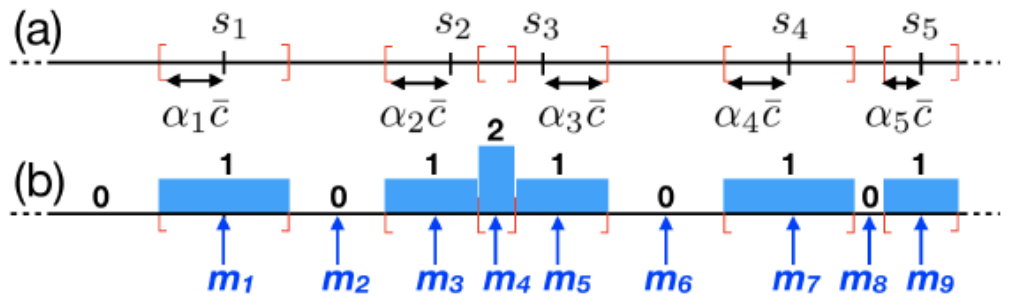


Fig. 3. (a) confidence interval for each measurement s_k (every s in the k-th interval satisfies $\frac{(s-s_k)^2}{\alpha_k^2} \leq \bar{c}^2$; (b) cardinality of the consensus set for every s and middle-points m_i for each interval with constant consensus set.

Algorithm 2: Adaptive Voting.

Input: $s_k, \alpha_k, \bar{c}$;
Output: $\hat{s}$, scale estimate solving (11);
% Define boundaries and sort
$\boldsymbol{v} = \text{sort}([s_1 - \alpha_1\bar{c}, s_1 + \alpha_1\bar{c}, \ldots, s_K - \alpha_K\bar{c}, s_K + \alpha_K\bar{c}])$
% Compute middle points
$m_i = \frac{\boldsymbol{v}_i + \boldsymbol{v}_{i+1}}{2}$ for $i = 1, \ldots, 2K-1$
% Voting
for $i = 1, \ldots, 2K-1$ **do**
  $\mathcal{I}_i = \emptyset$
  for $k = 1, \ldots, K$ **do**
    if $m_i \in [s_k - \alpha_k\bar{c}, s_k + \alpha_k\bar{c}]$ **then**
      $\mathcal{I}_i = \mathcal{I}_i \cup \{k\}$ % add to consensus set
    end
  end
end
% Enumerate consensus sets and return best
return: $\hat{s}$ from Eq. (14).

Outlier rejection

- After solving scale, discard edges that are outliers in scale

- Find maximal clique on graph and discard any edges not in that clique

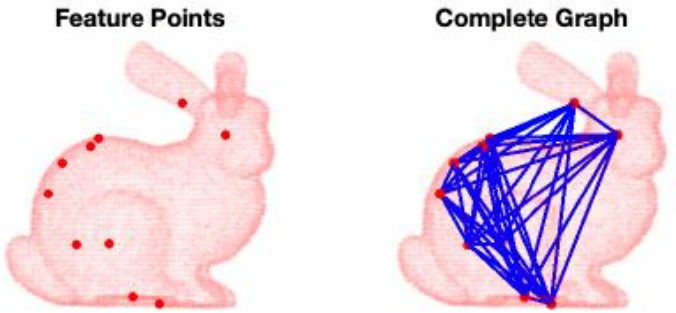


Fig. 2. TIMs generated from a complete graph in the Bunny dataset [97].

Theorem 6 (Maximal Clique Inlier Selection). *Edges corresponding to inlier* TIMs *form a clique in* $\mathcal{E}'$*, and there is at least one maximal clique in* $\mathcal{E}'$ *that contains all the inliers.*

Rotation estimation

- Get consensus estimate for rotation from TIM with scale now fixed

$$\hat{R} = \underset{\boldsymbol{R}\in \mathrm{SO}(3)}{\arg\min} \sum_{k=1}^{K} \min\left(\frac{\|\bar{b}_k - \hat{s}\boldsymbol{R}\bar{a}_k\|^2}{\delta_k^2} \,,\, \bar{c}^2 \right)$$

- Solve via semi-definite program relaxation (for certified result) or GNC (for speed)
  - GNC = graduated non-convex solver: https://arxiv.org/pdf/1909.08605.pdf

Translation estimation

- Solve for each translation component with scale and rotation fixed

$$\hat{t}_j = \arg\min_{t_j} \sum_{i=1}^{N} \min\left(\frac{(t_j - [\boldsymbol{b}_i - \hat{s}\hat{\boldsymbol{R}}\boldsymbol{a}_i]_j)^2}{\beta_i^2}, \bar{c}^2 \right)$$

- Same adaptive voting algorithm as for scale

TEASER++ implementation available

- C++ implementation
- Bindings for Python, Matlab, and ROS
- https://github.com/MIT-SPARK/TEASER-plusplus

Application: Object localization

- Use FPFH features to create correspondences

Fig. 8. Successful object pose estimation by TEASER on a real RGB-D dataset. Blue lines are the original FPFH [1] correspondences with outliers, green lines are the inlier correspondences computed by TEASER, and the final registered object is highlighted in red.

https://www.cvl.iis.u-tokyo.ac.jp/class2016/2016w/papers/6.3DdataProcessing/Rusu_FPFH_ICRA2009.pdf

Application: scan alignment

- 3DSmoothNet features w/ nearest neighbor search

	Mean	SD
Rotation error [rad]	0.066	0.043
Translation error [m]	0.069	0.053
# of FPFH correspondences	525	161
FPFH inlier ratio [%]	6.53	4.59

TABLE I
REGISTRATION RESULTS ON EIGHT SCENES OF THE RGB-D DATASET [36].

	Scenes								
	Kitchen (%)	Home 1 (%)	Home 2 (%)	Hotel 1 (%)	Hotel 2 (%)	Hotel 3 (%)	Study (%)	MIT Lab (%)	Avg. Runtime [s]
RANSAC-1K	91.3	89.1	74.5	94.2	84.6	90.7	86.3	81.8	0.008
RANSAC-10K	97.2	92.3	79.3	96.5	86.5	94.4	90.4	85.7	0.074
TEASER++	98.6	92.9	86.5	97.8	89.4	94.4	91.1	83.1	0.059
TEASER++ (CERT)	99.4	94.1	88.7	98.2	91.9	94.4	94.3	88.6	238.136

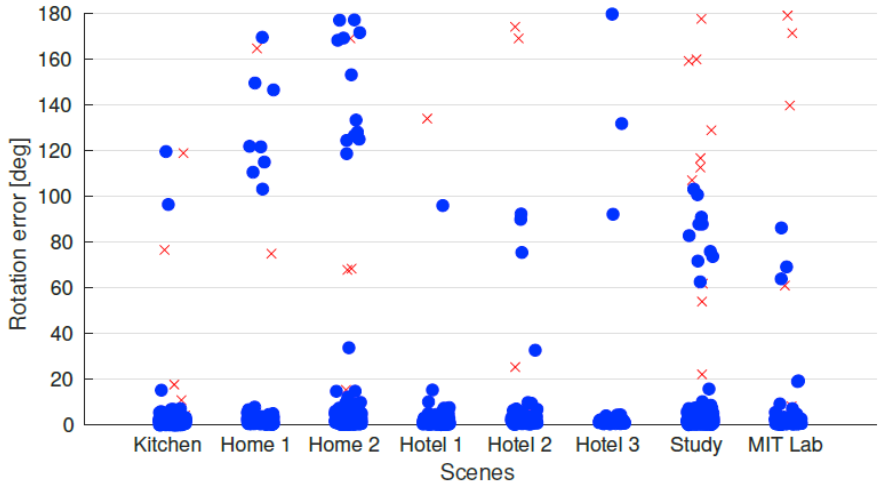
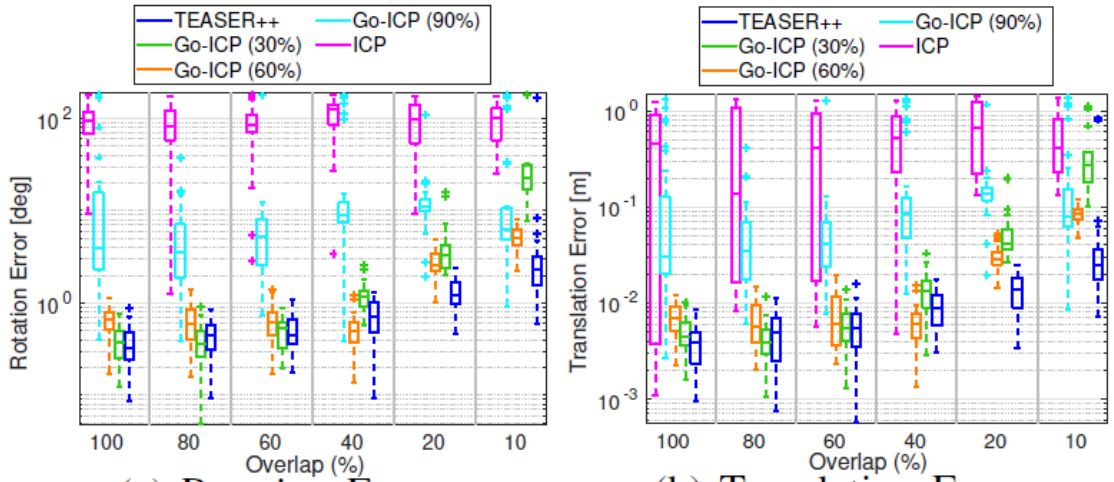


Fig. 10. Rotation errors for each scene (data points correspond to pairs of scans in the scene), with certified TEASER++ solutions (blue dots) vs. non-certified (red crosses).

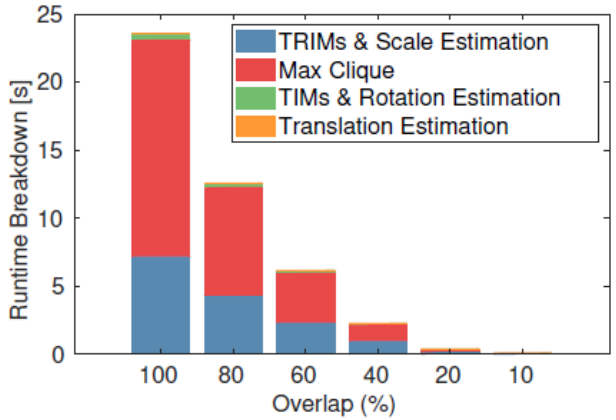
https://github.com/zgojcic/3DSmoothNet

Correspondence-free: start with all pairs of points as candidate correspondences



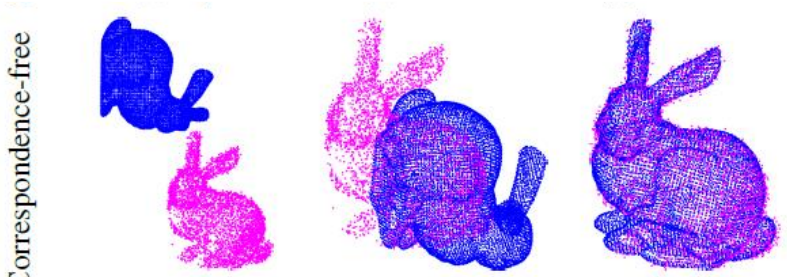
(a) Rotation Error

(b) Translation Error



(c) Timing Breakdown

Fig. 7. (a)-(b) Rotation and translation errors for TEASER++, ICP, and Go-ICP in a correspondence-free problem. (c) Timing breakdown for TEASER++.



(d) Input (e) ICP (f) TEASER++

Open problems / research ideas

- Most fitting/registration experiments either involve simple objects (bunny) or laser scans of scenes
  - Irregularity of image-based reconstructions creates major challenges
- Registration of “reality” to “model” is difficult – e.g. point cloud to BIM or drawing
- Likely potential to improve features for correspondence

Summary

- Fitting and registration are closely related problems with many potential solutions
  - Robust fit solver (relaxation, branch and bound, iterative)
  - RANSAC

- RANSAC is the easiest thing to try first

- Need for improved methods that work well for image-based reconstructions