Fast and Robust Normal Estimation for Point Clouds with Sharp Features

Alexandre Boulch & Renaud Marlet

University Paris-Est, LIGM (UMR CNRS), Ecole des Ponts ParisTech

École des Ponts
ParisTech

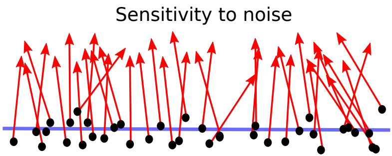
Symposium on Geometry Processing 2012

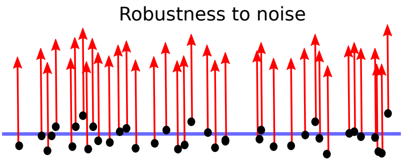
Normal estimation

Data

Point clouds from photogrammetry or laser acquisition:

- may be noisy

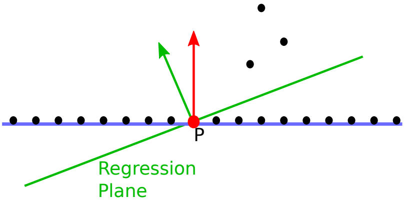




Data

Point clouds from photogrammetry or laser acquisition:

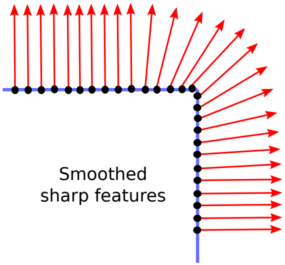
- may be noisy
- may have outliers

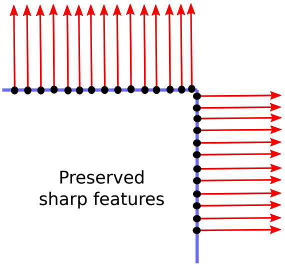


Data

Point clouds from photogrammetry or laser acquisition:

- may be noisy
- may have outliers
- most often have sharp features

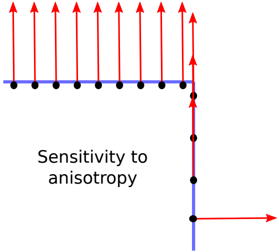


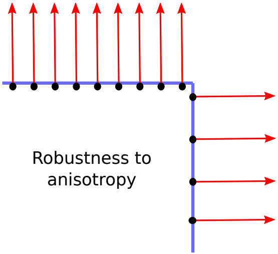


Data

Point clouds from photogrammetry or laser acquisition:

- may be noisy
- may have outliers
- most often have sharp features
- may be anisotropic





Data

Point clouds from photogrammetry or laser acquisition:

- may be noisy
- may have outliers
- most often have sharp features
- may be anisotropic
- may be huge (more than 20 million points)

Normal Estimation

Normal estimation for point clouds

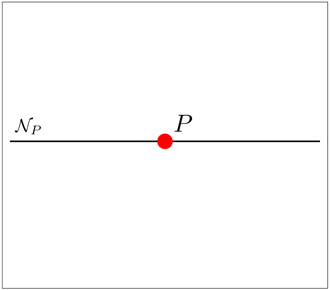
Our method

Experiments

Conclusion

Basics of the method (2D case here for readability)

Let P be a point and $\mathcal{N}_P$ be its neighborhood.

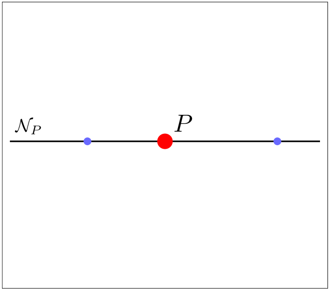


Basics of the method (2D case here for readability)

Let P be a point and $\mathcal{N}_P$ be its neighborhood.

We consider two cases:

- P lies on a planar surface

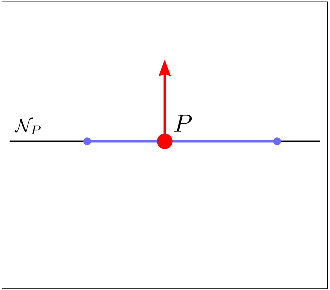


Basics of the method (2D case here for readability)

Let P be a point and $\mathcal{N}_P$ be its neighborhood.

We consider two cases:

- P lies on a planar surface

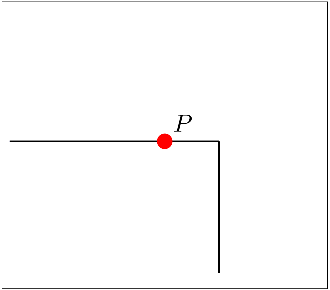


Basics of the method (2D case here for readability)

Let P be a point and $\mathcal{N}_P$ be its neighborhood.

We consider two cases:

- P lies on a planar surface
- P lies next to a sharp feature

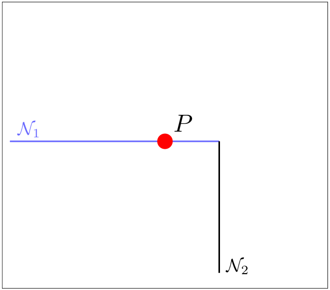


Basics of the method (2D case here for readability)

Let P be a point and $\mathcal{N}_P$ be its neighborhood.

We consider two cases:

- P lies on a planar surface
- P lies next to a sharp feature

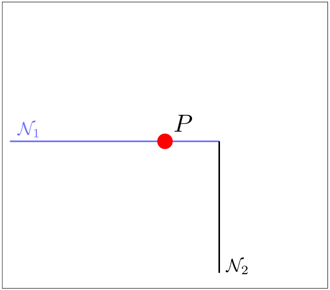


Basics of the method (2D case here for readability)

Let P be a point and $\mathcal{N}_P$ be its neighborhood.

We consider two cases:

- P lies on a planar surface
- P lies next to a sharp feature



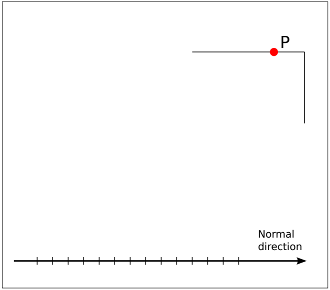
If $Area(\mathcal{N}_1) > Area(\mathcal{N}_2)$, picking points in $\mathcal{N}_1 \times \mathcal{N}_1$ is more probable than $\mathcal{N}_2 \times \mathcal{N}_2$, and $\mathcal{N}_1 \times \mathcal{N}_2$ leads to “random” normals.

Basics of the method (2D case here for readability)

Main Idea

Draw as many primitives as necessary to estimate the normal distribution, and then the most probable normal.

- Discretize the problem



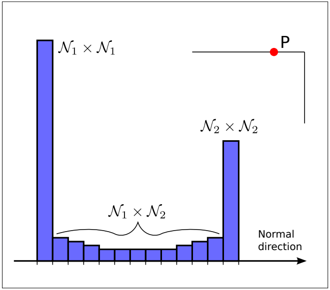
N.B. We compute the normal direction, not orientation.

Basics of the method (2D case here for readability)

Main Idea

Draw as many primitives as necessary to estimate the normal distribution, and then the most probable normal.

- Discretize the problem
- Fill a Hough accumulator

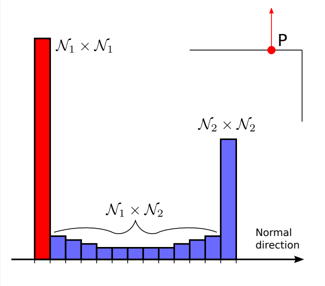


N.B. We compute the normal direction, not orientation.

Main Idea

Draw as many primitives as necessary to estimate the normal distribution, and then the most probable normal.

- Discretize the problem
- Fill a Hough accumulator
- Select the good normal



N.B. We compute the normal direction, not orientation.

Robust Randomized Hough Transform

- T, number of primitives picked after T iteration.
- T_{min}, number of primitives to pick
- M, number of bins of the accumulator
- $\hat{p}_m$, empirical mean of the bin m
- p_m, theoretical mean of the bin m

T_{min} such that:

$$\mathbb{P}(\max_{m \in \{1,\ldots,M\}} |\hat{p}_m - p_m| \leq \delta) \geq \alpha$$

From Hoeffding's inequality, for a given bin:

$$\mathbb{P}(|\hat{p}_m - p_m| \geq \delta) \leq 2\exp(-2\delta^2 T_{\mathrm{min}})$$

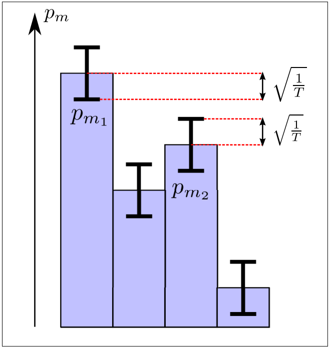
Considering the whole accumulator:

$$T_{\mathrm{min}} \geq \frac{1}{2\delta^2} \ln(\frac{2M}{1-\alpha})$$

Idea: if we pick often enough the same bin, we want to stop drawing primitives.
From the Central Limit Theorem, we can stop if:

$$\hat{p}_{m_1} - \hat{p}_{m_2} \geq 2\sqrt{\frac{1}{T}}$$

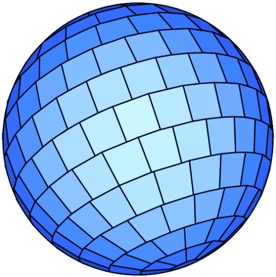
i.e. the confidence intervals of the most voted bins do not intersect (confidence level 95%)



Accumulator

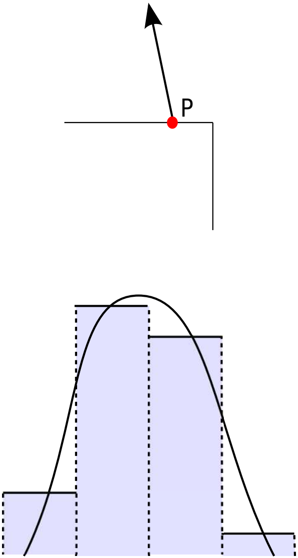
Our primitives are planes directions (defined by two angles).
We use the accumulator of Borrmann & al (*3D Research*, 2011).

- Fast computing
- Bins of similar area



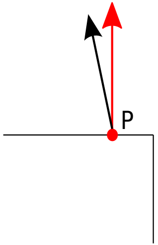
Discretization issues

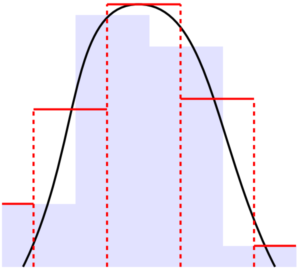
The use of a discrete accumulator may be a cause of error.



Discretization issues

The use of a discrete accumulator may be a cause of error.



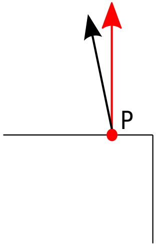


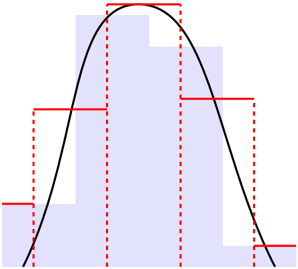
Discretization issues

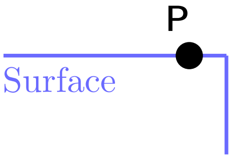
The use of a discrete accumulator may be a cause of error.

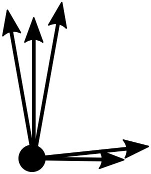
Solution

Iterate the algorithm using randomly rotated accumulators.

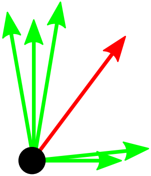




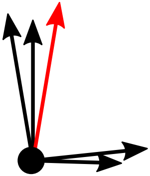




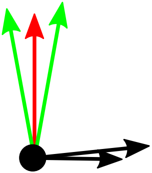
Normal directions
obtained by rotation
of the accumulator



Mean over all
the normals



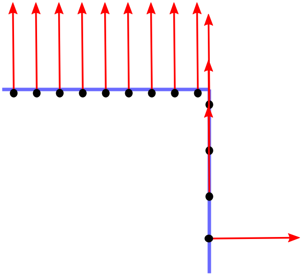
Best confidence



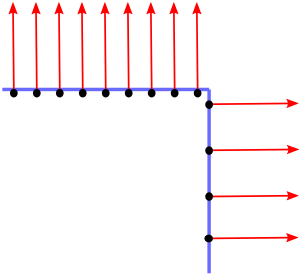
Mean over
best cluster

Dealing with anisotropy

The robustness to anisotropy depends of the way we select the planes (triplets of points)



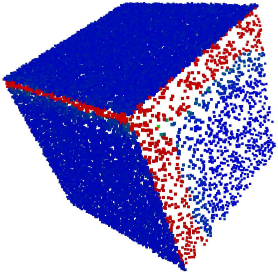
Sensitivity to anisotropy



Robustness to anisotropy

Random point selection among nearest neighbors

Dealing with anisotropy



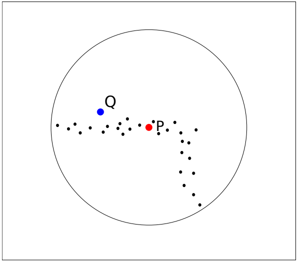
The triplets are randomly selected among the K nearest neighbors.

Fast but cannot deal with anisotropy.

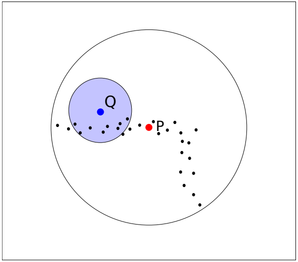
Uniform point selection on the neighborhood ball

Dealing with anisotropy

- Pick a point Q in the neighborhood ball



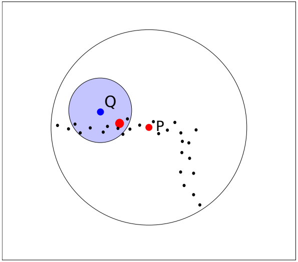
- Pick a point Q in the neighborhood ball
- Consider a small ball around Q



Uniform point selection on the neighborhood ball

Dealing with anisotropy

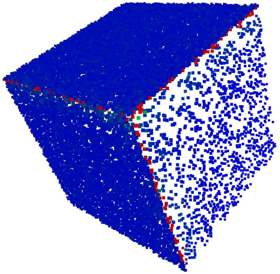
- Pick a point Q in the neighborhood ball
- Consider a small ball around Q
- Pick a point randomly in the small ball



Uniform point selection on the neighborhood ball

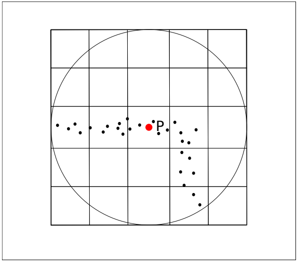
Dealing with anisotropy

- Pick a point Q in the neighborhood ball
- Consider a small ball around Q
- Pick a point randomly in the small ball
- Iterate to get a triplet



Deals with anisotropy, but for a high computation cost.

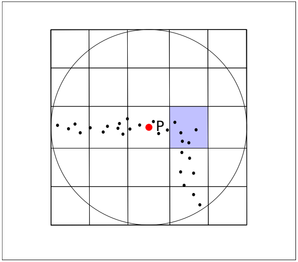
- Discretize the neighborhood ball



Cube discretization of the neighborhood ball

Dealing with anisotropy

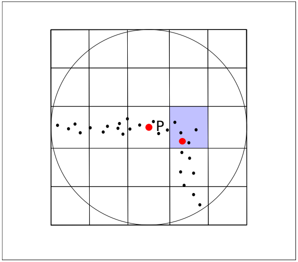
- Discretize the neighborhood ball
- Pick a cube



Cube discretization of the neighborhood ball

Dealing with anisotropy

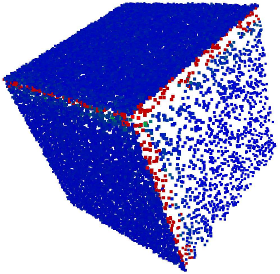
- Discretize the neighborhood ball
- Pick a cube
- Pick a point randomly in this cube



Cube discretization of the neighborhood ball

Dealing with anisotropy

- Discretize the neighborhood ball
- Pick a cube
- Pick a point randomly in this cube
- Iterate to get a triplet



Good compromise between speed and robustness to anisotropy.

Normal Estimation

Normal estimation for point clouds

Our method

Experiments

Conclusion

Methods used for comparison

- Regression
  - Hoppe & al (*SIGGRAPH*,1992): plane fitting
  - Cazals & Pouget (*SGP*, 2003): jet fitting

	Plane fitting	Jet fitting		
Noise	✓	✓		
Outliers				
Sharp fts				
Anisotropy				
Fast	✓	✓		

Methods used for comparison

- Regression
  - Hoppe & al (*SIGGRAPH*,1992): plane fitting
  - Cazals & Pouget (*SGP*, 2003): jet fitting
- Voronoï diagram
  - Dey & Goswami (*SCG*, 2004): NormFet

	Plane fitting	Jet fitting	NormFet	
Noise	✓	✓		
Outliers				
Sharp fts			✓	
Anisotropy			✓	
Fast	✓	✓	✓	

Methods used for comparison

- Regression
  - Hoppe & al (*SIGGRAPH*,1992): plane fitting
  - Cazals & Pouget (*SGP*, 2003): jet fitting
- Voronoï diagram
  - Dey & Goswami (*SCG*, 2004): NormFet
- Sample Consensus Models
  - Li & al (*Computer & Graphics*, 2010)

	Plane fitting	Jet fitting	NormFet	Sample Consensus
Noise	✓	✓		✓
Outliers				✓
Sharp fts			✓	✓
Anisotropy			✓	
Fast	✓	✓	✓	

Precision

Two error measures:

- Root Mean Square (RMS):

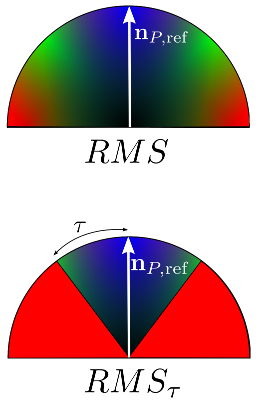
$$RMS = \sqrt{\frac{1}{|\mathcal{C}|} \sum_{P \in \mathcal{C}} \widehat{\mathbf{n}_{P,\mathrm{ref}} \mathbf{n}_{P,\mathrm{est}}}^{2}}$$

- Root Mean Square with threshold (RMS_τ):

$$RMS\_\tau = \sqrt{\frac{1}{|\mathcal{C}|} \sum_{P \in \mathcal{C}} v_P^2}$$

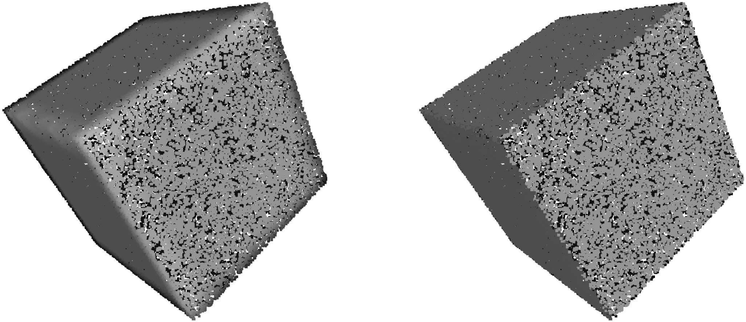
where

$$v_P = \begin{cases} \widehat{\mathbf{n}_{P,\mathrm{ref}} \mathbf{n}_{P,\mathrm{est}}} & \text{if } \widehat{\mathbf{n}_{P,\mathrm{ref}} \mathbf{n}_{P,\mathrm{est}}} < \tau \\ \frac{\pi}{2} & \text{otherwise} \end{cases}$$



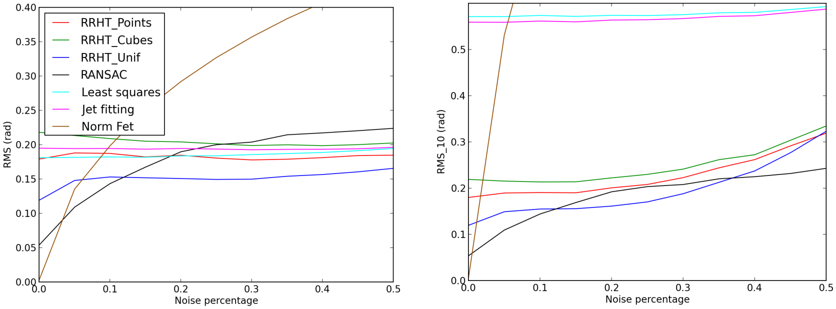
More suited for sharp features

Visual on error distances



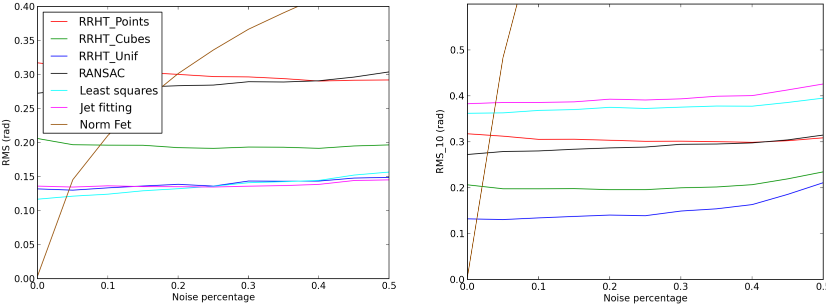
Same RMS, different RMS_{τ}

Precision (with noise)



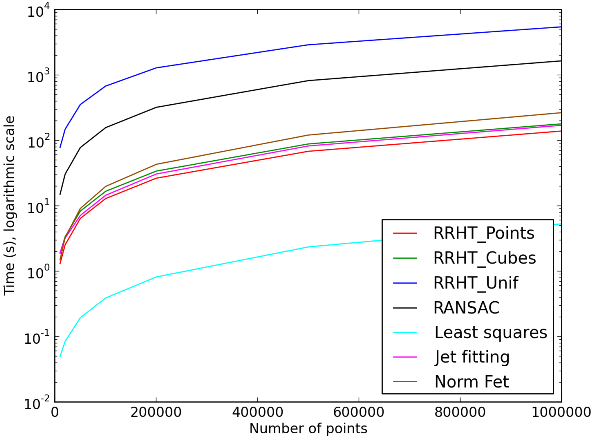
Precision for cube uniformly sampled, depending on noise.

Precision (with noise and anisotropy)



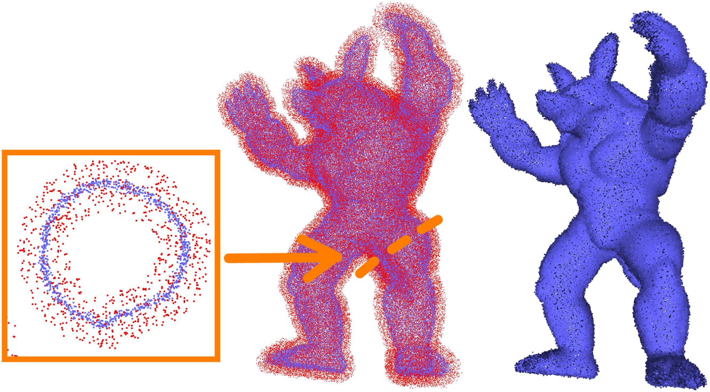
Precision for a corner with anisotropy, depending on noise.

Computation time



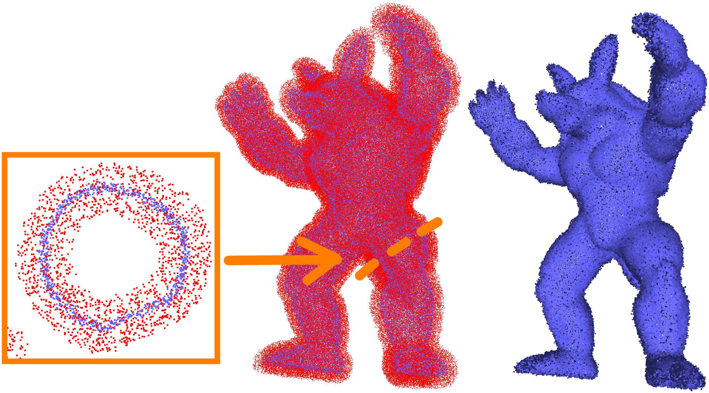
Computation time for sphere, function of the number of points.

Robustness to outliers



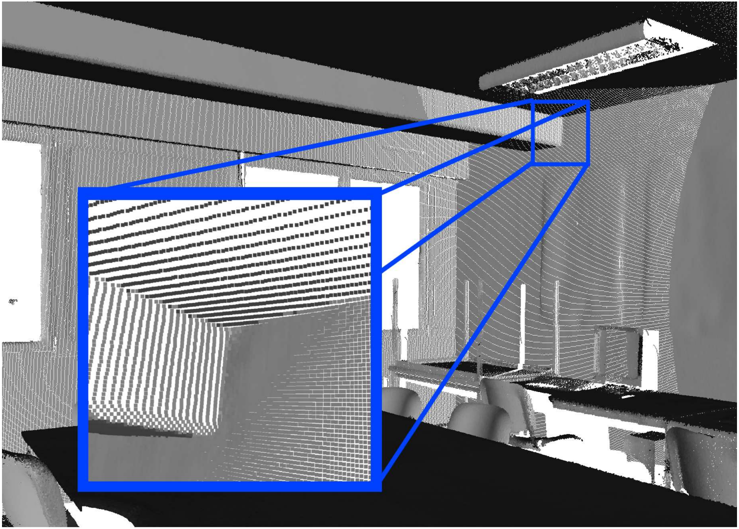
Noisy model (0.2%) + 100% of outliers.

Robustness to outliers

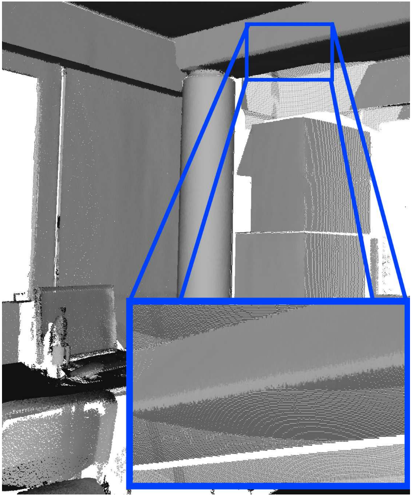


Noisy model (0.2%) + 200% of outliers.

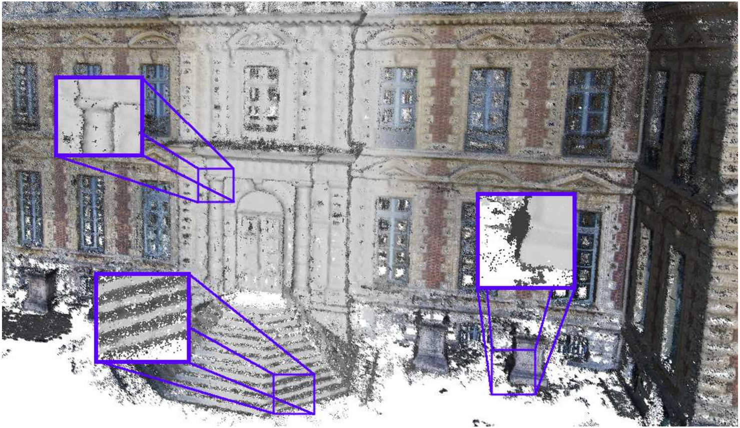
Robustness to anisotropy



Preservation of sharp features



Robustness to “natural” noise, outliers and anisotropy



Point cloud created by photogrammetry.

Normal Estimation

Normal estimation for point clouds

Our method

Experiments

Conclusion

Conclusion

	Plane fitting	Jet fitting	NormFet	Sample Consensus	Our method
Noise	✓	✓		✓	✓
Outliers				✓	✓
Sharp fts			✓	✓	✓
Anisotropy			✓		✓
Fast	✓	✓	✓		✓

Compared to state-of-the-art methods that preserve sharp features, our normal estimator is:

- at least as precise
- at least as robust to noise and outliers
- almost 10x faster
- robust to anisotropy

Code available

Web site

https://sites.google.com/site/boulchalexandre

Two versions under GPL license:

- for Point Cloud Library (http://pointclouds.org)
- for CGAL (http://www.cgal.org)

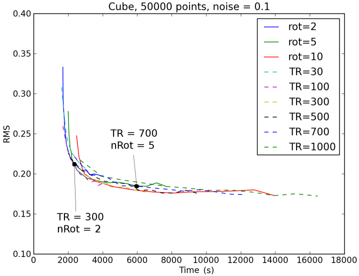
Computation time

Model (# vertices)	T_{min}=700 n_{rot}=5 w/o interv.	with interv.	T_{min}=300 n_{rot}=2 w/o interv.	with interv.
Armadillo (173k)	*21 s*	*20 s*	*3 s*	*3 s*
Dragon (438k)	*55 s*	*51 s*	*8 s*	*7 s*
Buddha (543k)	1.1	1	*10 s*	*10 s*
Circ. Box (701k)	1.5	1.3	*13 s*	*12 s*
Omotondo (998k)	2	1.2	*18 s*	*10 s*
Statuette (5M)	11	10	1.5	1.4
Room (6.6M)	14	8	2.3	1.6
Lucy (14M)	28	17	4	2.5

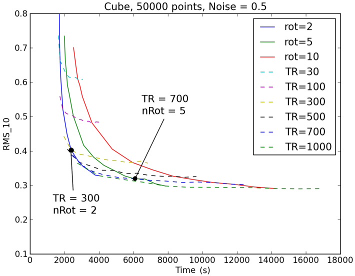
Parameters

- K or r: number of neighbors or neighborhood radius,
- T_{min}: number of primitives to explore,
- n_ϕ: parameter defining the number of bins,
- n_{rot}: number of accumulator rotations,
- c: presampling or discretization factor (anisotropy only),
- $a_{cluster}$: tolerance angle (mean over best cluster only).

Efficiency



Cube, 50000 points, noise = 0.1
RMS
Time (s)
rot=2
rot=5
rot=10
TR=30
TR=100
TR=300
TR=500
TR=700
TR=1000
TR = 700
nRot = 5
TR = 300
nRot = 2



Cube, 50000 points, Noise = 0.5
RMS_10
Time (s)
rot=2
rot=5
rot=10
TR=30
TR=100
TR=300
TR=500
TR=700
TR=1000
TR = 700
nRot = 5
TR = 300
nRot = 2

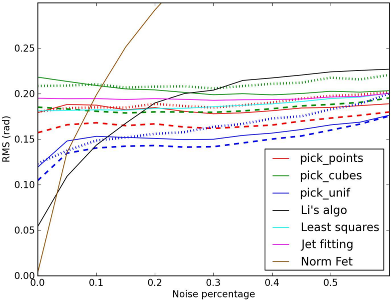
Influence of the neighborhood size

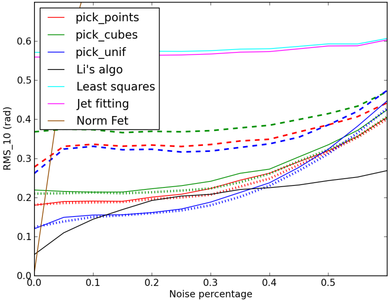
$K = 20$

$K = 200$

$K = 400$

Precision (with noise)





Precision for cube uniformly sampled, depending on noise.