L3 – Preprocessing of Point Model

- Common nature of acquisition results
  - Unorganized scatter points
  - Present noises, outliers and non-uniformity
  - Some regions may be missed during acquisition
- Requirements by downstream algorithms
  - Consistently oriented normal vectors
  - Uniformly sampled
  - Noise and outlier free
  - Complete with missed region filled (or recovered)

Preprocessing Techniques

- Normal estimation
  - Principal Component Analysis (PCA)
  - Local surface fitting
  - Consistent orientation
- Denoising by projection
- Outlier removal and processing
  - Heuristic based removal methods
  - Robust statistic based processing

Search Data Structures

- Nearest-neighbor searches and range queries
  - Search and store in a *neighborhood table*
  - Or search on-site to reduce the memory usage
- K-d-tree based *approximate-nearest-neighbor* (ANN)
  - Efficient ($O(n \log n)$ in construction; $O(\log n)$ in query)
  - Static point set
  - Range query

  http://www.cs.umd.edu/~mount/ANN/
- Dynamic data may needs a hash data structure
  - Perform poorly in non-uniform data set

Principal Component Analysis (PCA)

- Computing the co-variant matrix of points

$$\sum_{\mathbf{p}_i \in N'(\mathbf{p})} (\mathbf{p}_i - \bar{\mathbf{p}})(\mathbf{p}_i - \bar{\mathbf{p}})^T$$

  Normal is chosen as the eigen-vector corresponding to the smallest eigen-value

- Why?
  - The minimization problem

$$\min \; n^T C n \quad \text{s.t.} \quad \| n \| = 1$$

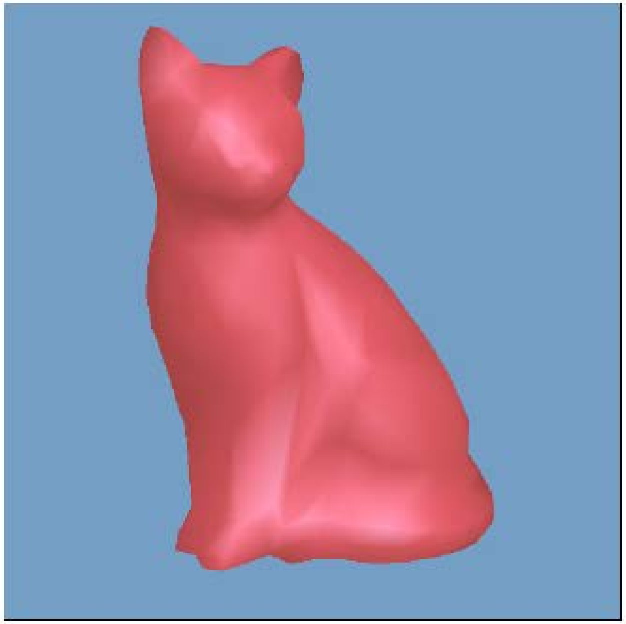
- How about the orientation?

Orientation Plays Important Role

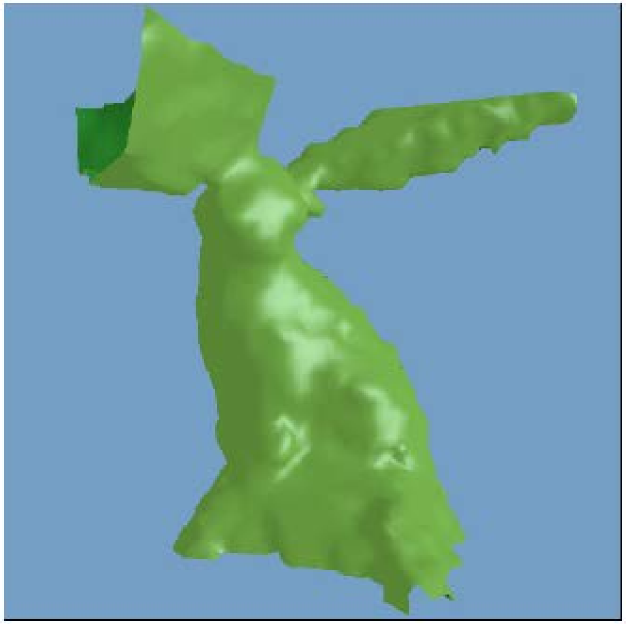
- Normal vectors give the definition of underlying surface to the first order
- Many implicit surface reconstruction methods rely on them to define the inside/outside fields
- However, eigen-vector analysis cannot provide a correct orientation
- Re-orienting the normal vectors is necessary

Orientation Propagation

- A relatively simple-minded algorithm to orient the points
  - Arbitrarily choose an orientation for some plane
  - Then "propagate" the orientation to neighboring planes
  - Where does the graph come from?
- However, the **order** of propagation is **important**



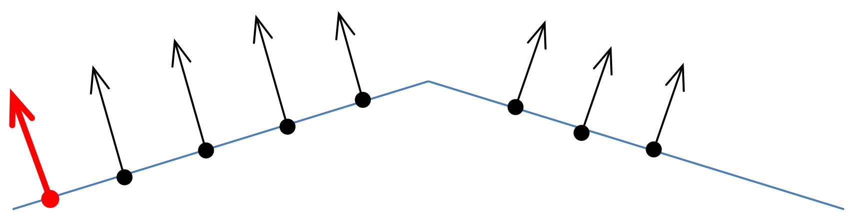
(a) Original mesh

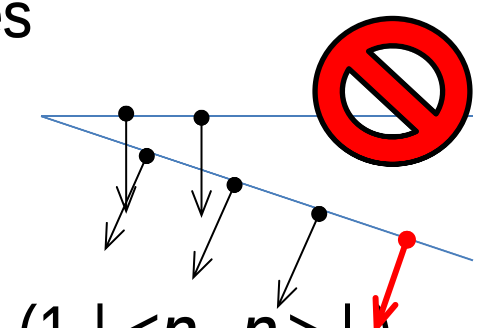


(b) Result of naive orientation propagation

Heuristic of "Best" Order

- Favor propagation from point x_i to x_j if the unoriented planes at them are nearly parallel
  - This order is advantageous because it tends to propagate orientation along directions of low curvature in the data, thereby largely avoiding ambiguous situations encountered when trying to propagate orientation across sharp edges

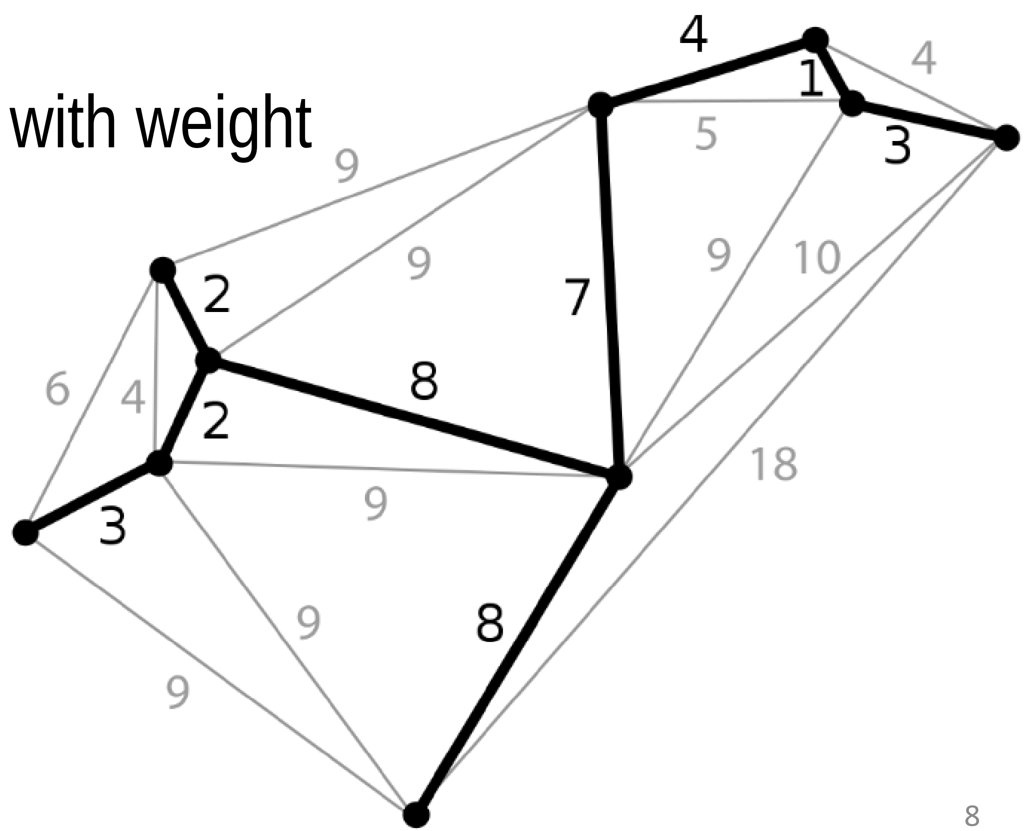




- Assign each edge in graph with the weight: $(1-|\langle n_i, n_j \rangle|)$
- Compute the order by traversing the *minimal spanning tree* (MST) of the graph

Minimum Spanning Tree

- Given a connected, undirected graph, a spanning tree of that graph is a subgraph which is a **tree** and connects **all** the vertices together.
- MST is a spanning tree with weight less than or equal to the weight of every other spanning tree



MST Construction – Prim's Algorithm

- Continuously increases the size of a tree, one edge at a time, starting from a single vertex until it spans all nodes

Input: A non-empty connected weighted graph with vertices V and edges E

Initialize: $V_{new} = \{x\}$, where x is an arbitrary node (starting point) from V, $E_{new} = \{\}$

Repeat until $V_{new} = V$:

1) Choose an edge (u, v) with minimal weight such that u is in V_{new} and v is not (if there are multiple edges with the same weight, any of them may be picked)

2) Add v to V_{new}, and (u, v) to E_{new}

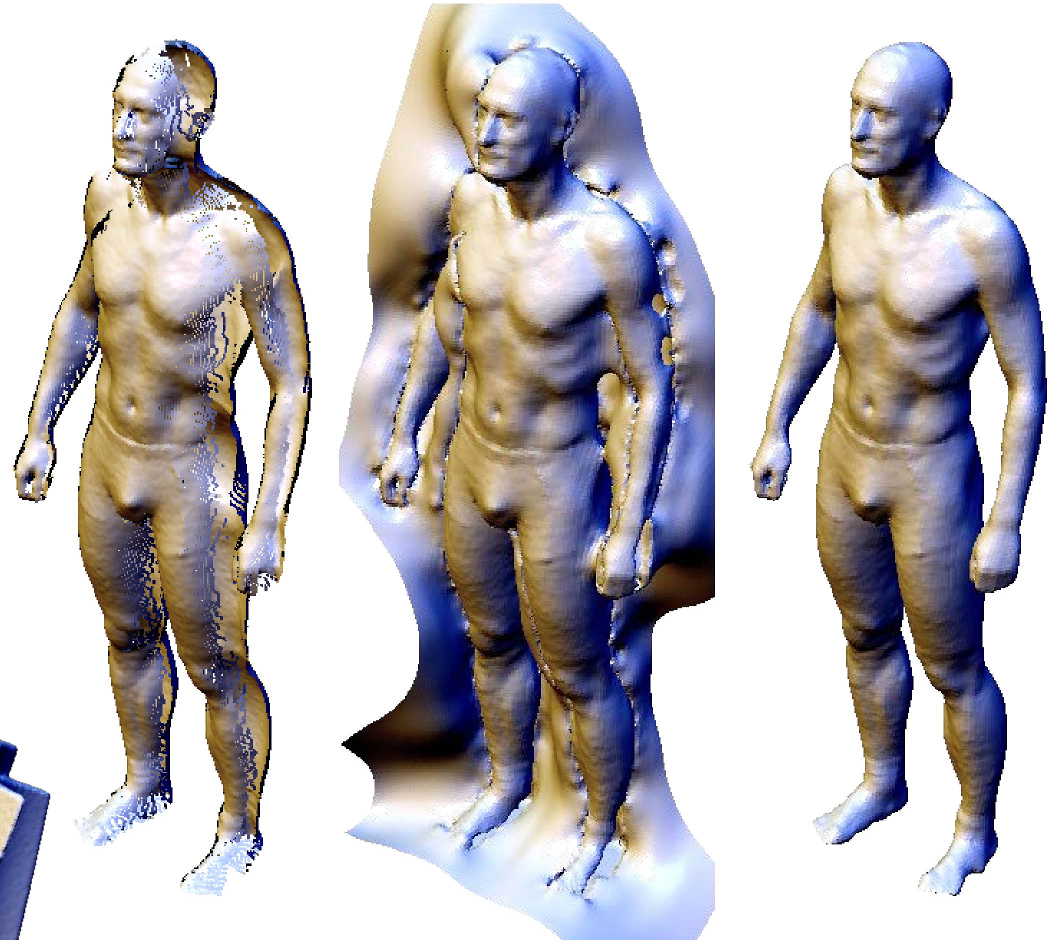
Output: V_{new} and E_{new} describe a minimal spanning tree.

*Implementation can use binary heap to achieve the complexity with $O((V + E) \log(V)) = O(E \log(V))$

Normal Orientation on MST

- To assign orientation to an initial plane, the unit normal of the plane whose center has the largest z coordinate is forced to point toward the $+z$ axis (as an *heuristic*).
- Then, rooting the tree at this initial node, we traverse the tree in *depth-first order*, assigning each plane an orientation that is consistent with that of its parent.
  - That is, if during traversal, the current plane at x_i has been assigned the orientation n_i and x_j is the next point to be visited, then n_j is replaced with $-n_j$ if ($\langle n_i, n_j \rangle < 0$)
- Such algorithm works successfully on well sampled models

Problems



Does not work very well on data set captured from real models by laser scanners

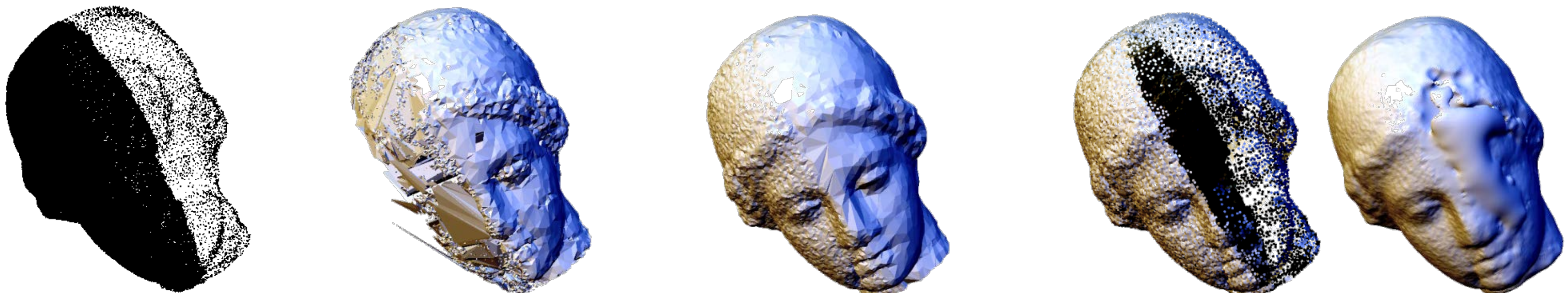
New Method (ORT)

- Using the integrating approach of meshing [Ohtake et al., 2005]
- A modified scheme of *Adaptive Spherical Cover* (ASC)
- An orientation-aware *Principle Component Analysis* (PCA)
- Different from Consolidation [Huang et al., 2009]
  - Do not remove or re-position points
  - Only re-assigning normal vectors to all the input sample points
- Although as a pre-processing step, plays an important role to the mesh surface reconstruction

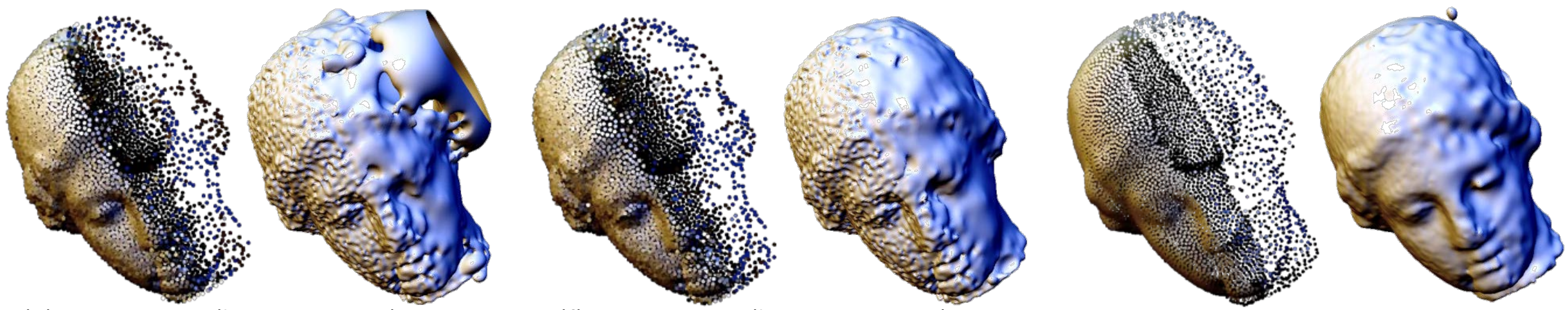
http://homepage.tudelft.nl/h05k3/projPOT.html

http://homepage.tudelft.nl/h05k3/pubs/SMI10PntOrienting.pdf

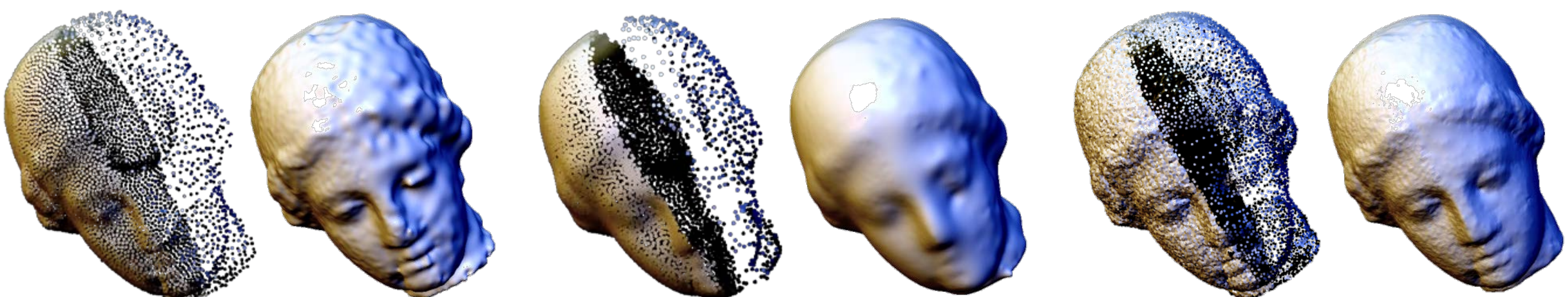
Comparison of ORT vs. Other Methods



(a) Scattered Points (b) Tight CoCone (c) ASC+Meshing (d) MST+RBF



(e) Cons+RBF (h=0.47; Iter=1) (f) Cons+RBF (h=0.47; Iter=10) (g) Cons+RBF (h=1.0; Iter=1)



(h) Cons+RBF (h=1.0; Iter=10) (i) Cons+RBF (h=3.0; Iter=10) (j) ORT+RBF

Denoising by Projection

- Moving Least Squares (MLS) surface - *semi-implicit*
  - Represents the surface by projecting all the points onto the estimated smooth surface – different from implicit surface reconstruction
- Two steps in one pass of projection:
  - 1) defining a local reference domain
  - 2) fitting a local bi-variate polynomial over the reference domain and projecting points onto the surface
- Theoretical analysis shows that *repeatedly applying such projection operators* **converges** to a *smooth* surface
- Details will be provided later in the MLS related lecture

Local Quadratic Surface Fitting

- Usually fit by quadratic surfaces

$$S(s,t) = as^2 + bt^2 + cst + ds + et$$

  - Construct local frame
  - Project the sample points onto the tangent plane
  - Determine the (*s*,*t*) parameters of points
  - Solving the coefficient (a, b, c, d, e) in a least-square manner

  * More polynomial choices by using more terms in the polynomial triangle
- Compatibility of locally constructed quadratic surfaces?
- Blending on least-square fitting is need
- The concept of *Moving Least Squares* (MLS) surface

Simpler Implementation of Projection

- Iteratively computes the locally weighted average position and projects the point along the normal direction yielding a new position until it converges
- Give a point *x*, the locally weighted average is defined as

$$a(x) = \frac{\sum_{i=1}^{N} \theta(||x - p_i||) p_i}{\sum_{i=1}^{N} \theta(||x - p_i||)}$$

with θ specifying the influence of the neighboring points

$$\theta(d) = e^{-d^2/h^2}$$

h is a factor that defines the Gaussian kernel width.

*Features would be smoothed out if their sizes are smaller than *h*

Simpler Implementation of Projection

- Choosing a suitable *h* is difficult for non-uniformly sampled point set
  - [Adamson and Alexa, 2004] computed *h* as the average Euclidean *k*-nearest neighborhood distance with *k* = 6
  - This gives an adaptive approximation of local sampling density
- The next updated position *x*' of *x* is computed by

$$x' = x - n(x)^T (x - a(x)) n(x)$$

with

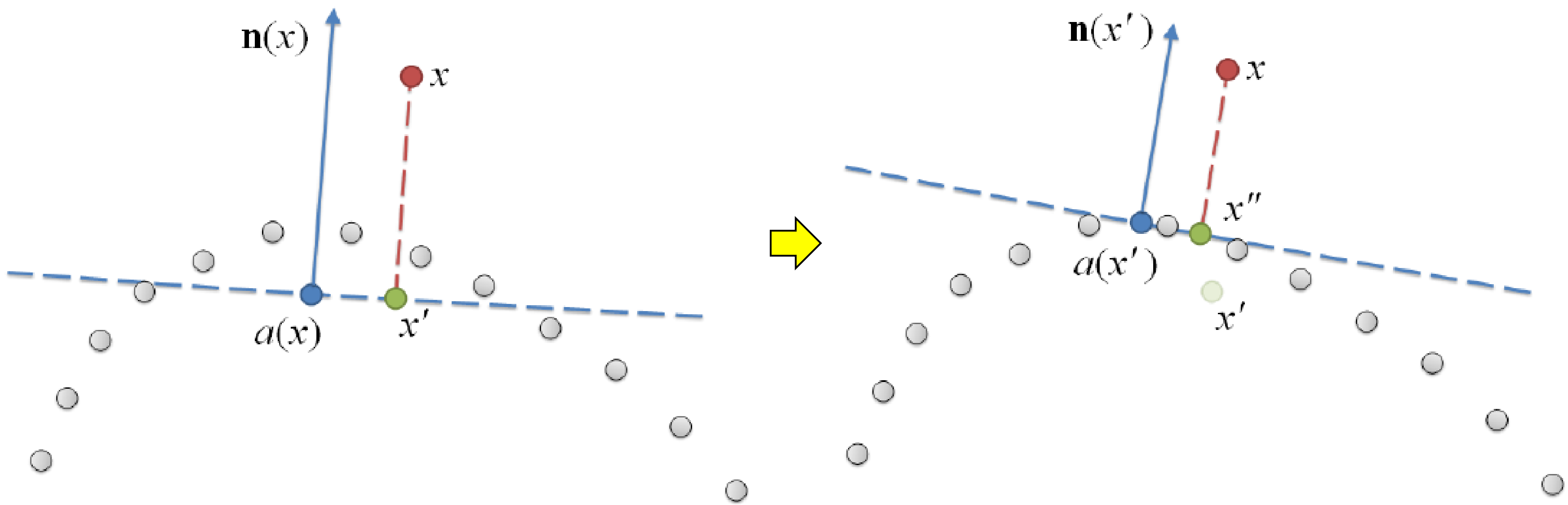
$$\mathbf{n}(x) = \arg\min \sum_{i=1}^{N} \|\mathbf{n}^T (x - p_i)\|^2 \theta(||x - p_i||)$$

or simply

$$\mathbf{n}(x) = \frac{\sum_i^N \theta(||x - p_i||)\mathbf{n}_i}{\| \sum_{i=1}^N \theta(||x - p_i||)\mathbf{n}_i \|}$$

Oriented Consistently?

Illustration of Simple Projection



- Computing

$$\mathbf{n}(x) = \arg\min \sum_{i=1}^{N} \|\mathbf{n}^T(x - p_i)\|^2 \theta(\|x - p_i\|)$$

need to solve Eigen value decomposition problem – heavier computation

Point Relaxation – Like Particles

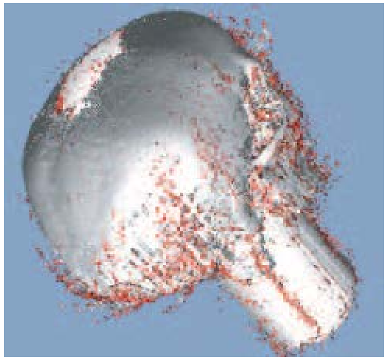
- To achieve a uniform distribution of the particles
  - Neighbored particles are let to repel each other [Pauly et al., 02]
  - Every particle **p** exerts a force $\mathbf{f}_i(\mathbf{p})$ on its neighbored particles $\mathbf{p}_i$

$$\mathbf{f}_i(\mathbf{p}) = k(r - \|\mathbf{p} - \mathbf{p}_i\|)\frac{\mathbf{p}_i - \mathbf{p}}{\|\mathbf{p}_i - \mathbf{p}\|}$$

  - The summation of all forces that act on a particle gives the resulting force
  - Finally, the new positions of the particles are computed by explicit Euler integration
- After each iteration, the particles are projected back onto the surface by applying the projection operator.

Heuristic Outlier Removal Methods

- Erroneous points outside the object surface are ***outliers*** that have to be removed
- Three outlier criteria
  - All deliver an estimator $\chi(\mathbf{p}) \in [0, 1]$ assigning the likelihood for a point sample $\mathbf{p}$ to be an outlier
  - All criteria are based only on $\mathbf{p}$'s k-nearest neighbors $N_k(\mathbf{p})$.
- Outliers are finally removed by applying a threshold to the resulting outlier classification

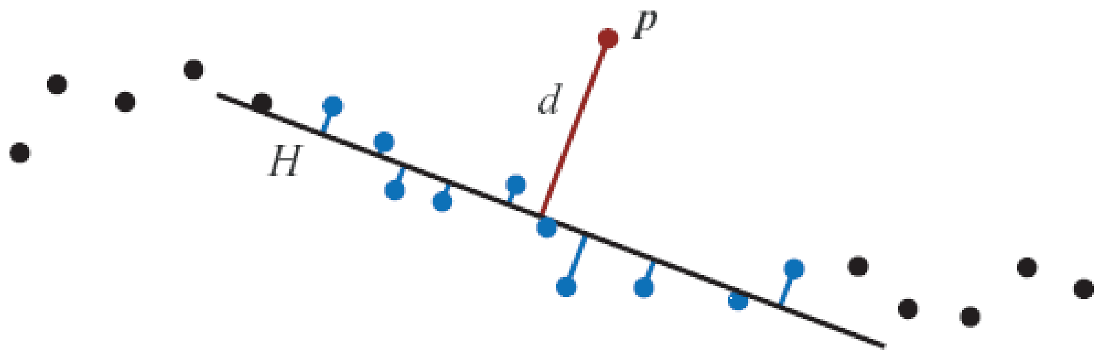


Plane Fit Criterion

- Plane H minimizing the squared distances to **p**'s neighbors

$$\min_H \sum_{\mathbf{q}\in\mathcal{N}_k(\mathbf{p})} \mathrm{dist}(\mathbf{p}, H)^2$$

- The plane fitting criterion is defined as: $\chi_{\mathrm{pl}}(\mathbf{p}) = \dfrac{d}{d+\bar{d}}$
  - d is the distance of **p** to H
  - $\bar{d}$ is the mean distance of points from N_k(**p**) to H

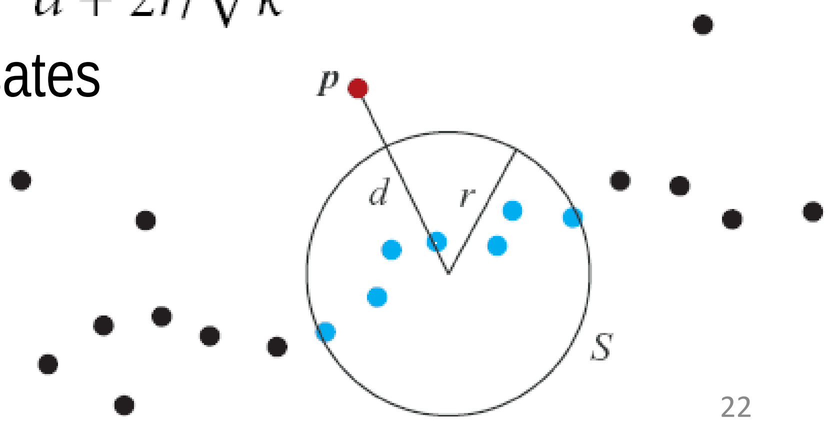


Minibal Criterion

- A point comparatively distant to the cluster built by its *k*-nearest neighbors is likely to be an outlier
  - The smallest enclosing sphere *S* around $N_k(\mathbf{p})$, can be considered as an approximation of the *k*-nearest-neighbor cluster
  - *d* is the distance from **p** to the center of *S*
- The *minimal criterion* is

$$\chi_{\mathrm{mb}}(\mathbf{p}) = \frac{d}{d + 2r/\sqrt{k}}$$

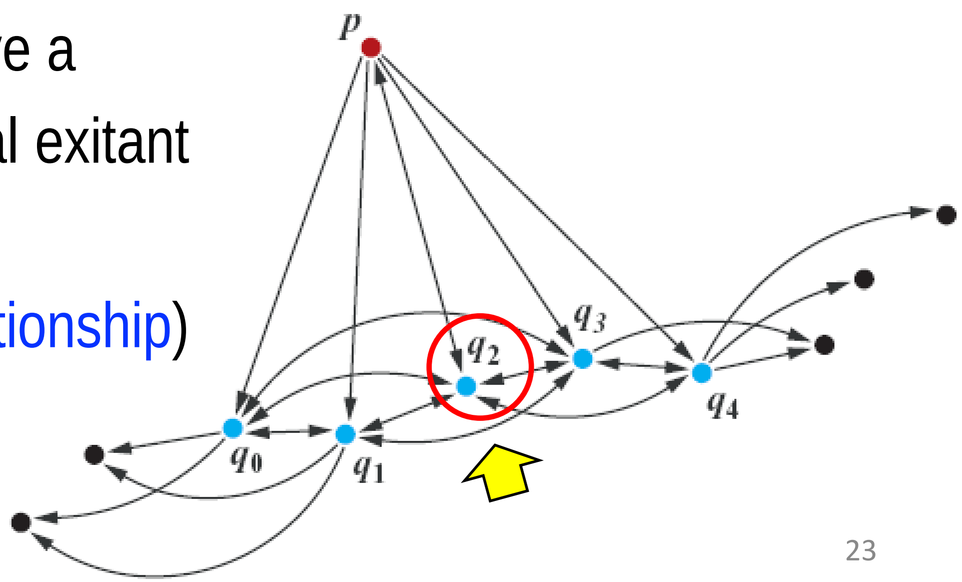
  - Normalization by sqrt(*k*) compensates for the diameter's increase with increasing number of k-neighbors at the object surface



Nearest-neighbor Reciprocity Criterion

- Based on the following observations:
  - A “valid” point sample **q** may be in the k-neighborhood of outlier
  - The outlier will most likely not be part of **q**’s k-neighborhood
- Such relationship can be expressed by means of a directed graph *G* of *k*-neighbor relationships
  - Outliers are assumed to have a high number of unidirectional exitant edges

(i.e., asymmetric neighbor relationship)



Nearest-neighbor Reciprocity Criterion

- Unidirectional neighbors of p are defined as

$$\mathcal{N}_{k,\mathrm{uni}}(\mathbf{p}) = \{\mathbf{q} \mid \mathbf{q} \in \mathcal{N}_k(\mathbf{p}), \mathbf{p} \notin \mathcal{N}_k(\mathbf{q})\}$$

- Bidirectional neighbors of p are

$$\mathcal{N}_{k,\mathrm{bi}}(\mathbf{p}) = \{\mathbf{q} \mid \mathbf{q} \in \mathcal{N}_k(\mathbf{p}), \mathbf{p} \in \mathcal{N}_k(\mathbf{q})\}$$

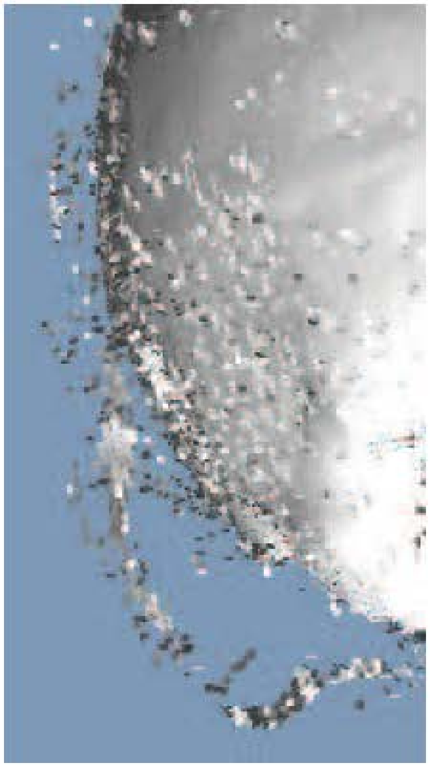
- The classifier is then expressed as:

$$\chi_{\mathrm{bi}}(\mathbf{p}) = \frac{\|\mathcal{N}_{k,\,\mathrm{uni}}(\mathbf{p})\|}{\|\mathcal{N}_{k,\,\mathrm{bi}}(\mathbf{p})\| + \|\mathcal{N}_{k,\,\mathrm{uni}}(\mathbf{p})\|} = \frac{\|\mathcal{N}_{k,\,\mathrm{uni}}(\mathbf{p})\|}{k}$$

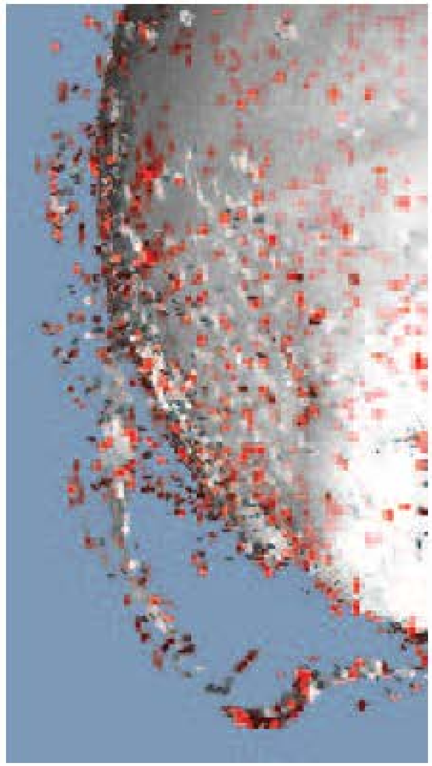
- **Integrated Classifier** by all three criteria

$$\chi(\mathbf{p}) = w_1 \chi_{\mathrm{pl}}(\mathbf{p}) + w_2 \chi_{\mathrm{mb}}(\mathbf{p}) + w_3 \chi_{\mathrm{bi}}(\mathbf{p}) \qquad \textstyle\sum_i w_i = 1$$

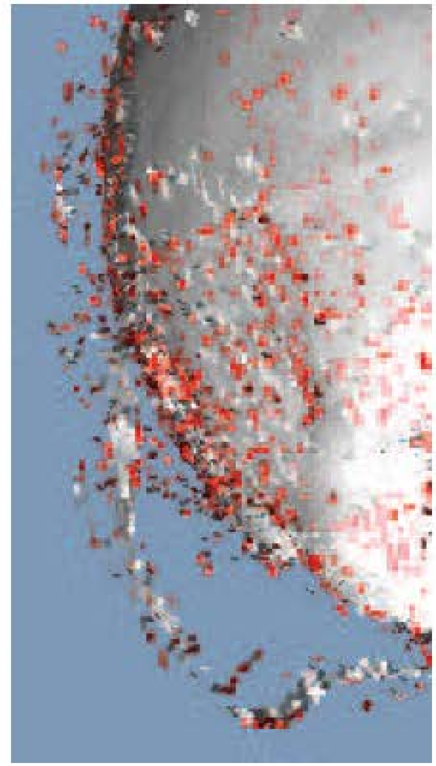
Heuristic Outlier Removal Results



Given



Miniball



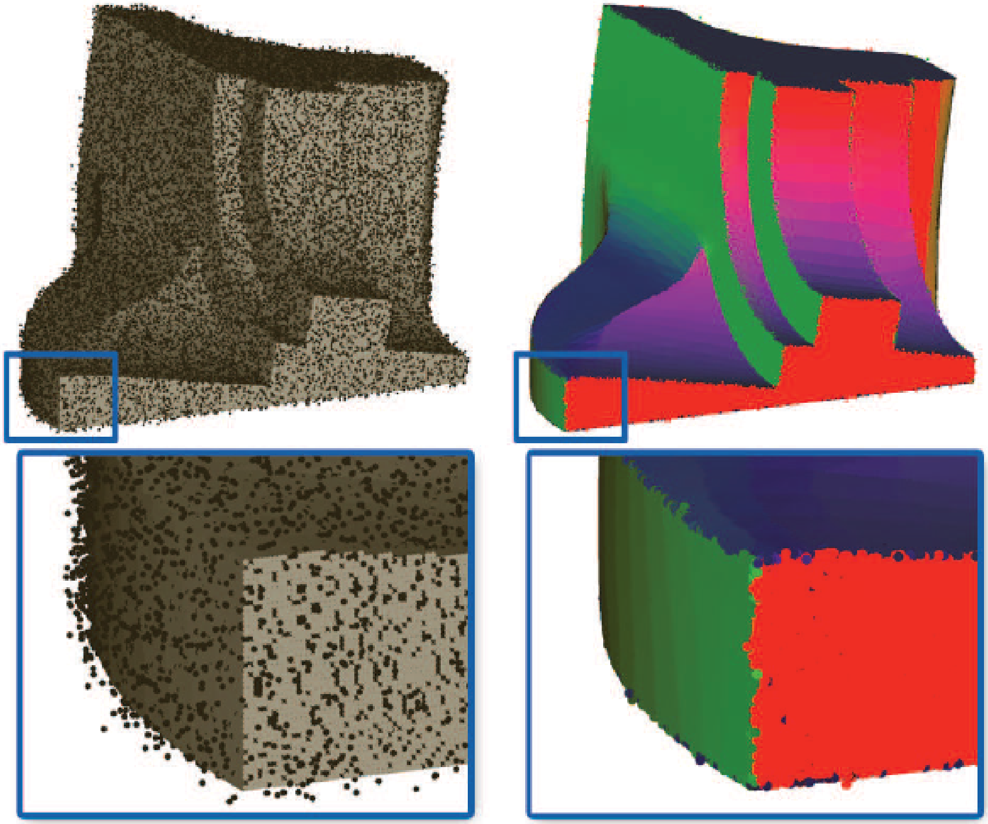
Plane Fit

k-nearest-neighbor graph

*All criteria were threshold to classify 7% of the surfels as outliers

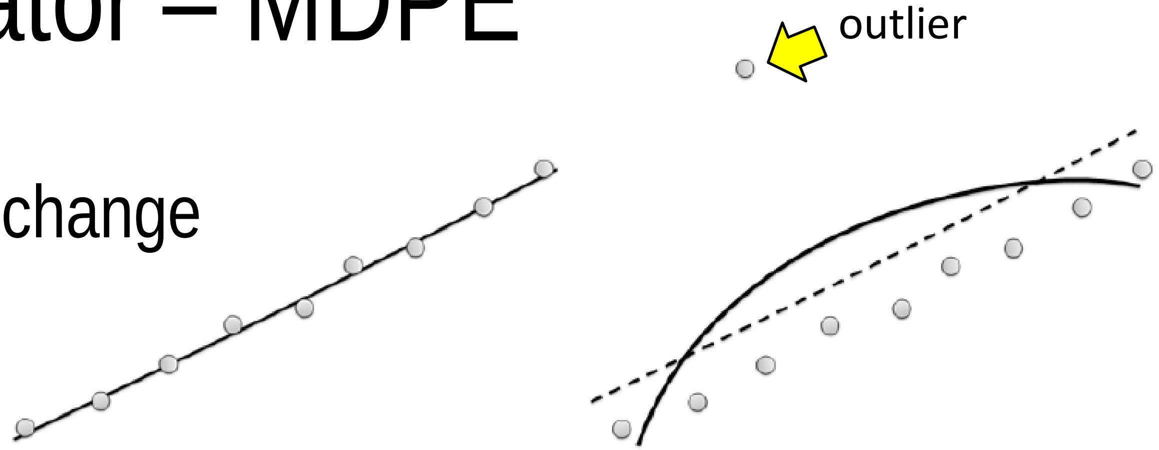
Robust Statistics Based Processing

- Robust local surface fitting and point projection
  - Fit a surface to the local shape around a sample p
  - p is projected onto the fitted surface
  - Normal vector at p is then estimated
- Problems to be solved
  - Noises
  - Outliers
  - Multiple structures



Robust Estimator – MDPE

- A single outlier can change the fitting arbitrarily



- When a model is correctly fitted, it should satisfy
  1) There are as many as possible data points on or near the model
  2) The residuals of inliers should be as small as possible

*_The least squares method only uses the second criterion as its objective function to minimize the residuals without distinguishing the inliers from outliers_

- A robust estimator is needed: MUSE, RANSAC, RESC, etc.

Surface Estimation by MDPE

- MDPE to find a quadratic surface best fitting a local shape
  - *p* points are randomly selected from *N*(x) of a sample point x
  - fit a quadratic surface *S* to these *p* points
  - the probability density power *DP* according to this fit is evaluated by the residuals of points in *N*(x) to *S*
  - **repeat** above steps for *m* times, and among the *m* fits, the surface with the **maximal score** in *DP* is the robust fitting result.
  - In [Sheung and Wang, 2009], they choose:
    - *p*=6 and fit a quadratic surface with 5 coefficients in a LS way with SVD.
    - The smaller *h* is used, the more sensitive to noises the estimator is.
    - However, some inliers may be ignored if *h* is too small.
    - By experience, *h* is selected as twice of the average point distance.

http://homepage.tudelft.nl/h05k3/pubs/SPM09PntCloud.pdf

Normal Estimation & Point Projection

- Theoretically, the value of repeated times, *m*, relates to the probability *P* that at least one clean p-subset is chosen

$$m = \frac{\log(1-P)}{\log[1-(1-\varepsilon)^p]}$$

  where ε is the fraction of outliers.
- In practice, *m* = 300 is used in twofold:
  - We do not know the value of the fraction of outliers, ε
  - Using a value of *m* computed by above formula still *cannot guarantee* to find a good fit among random selections
- After finding the best surface *S** (with maximum *DP*)
  - The projected position x' of x is the closest point x_c on *S** to x (which can be searched by Newton's method)
  - The normal of surface *S** at x_c is employed as the normal vector to equip x'.

Robust Moving Least Squares (RMLS)

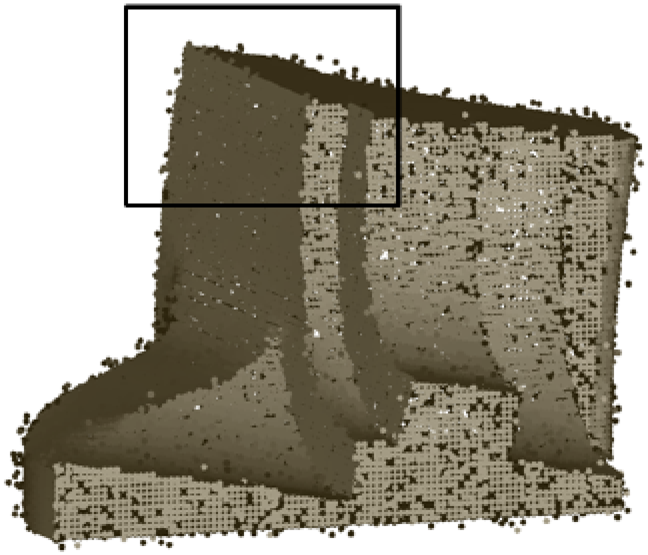
- Conventional MLS surface defines a surface that is smooth everywhere, thus it cannot preserve sharp features.
- [Fleishman et al., 05] introduced a robust method, *forward search algorithm*, to identify the outliers & multiple-structure
  - Starting from a small outlier-free region estimated by an initial robust estimator
  - One good sample is added iteratively to re-fit the polynomial
  - Until the largest residual is greater than a certain threshold
  - One surface is then classified and the whole process is repeated until the sample set is empty

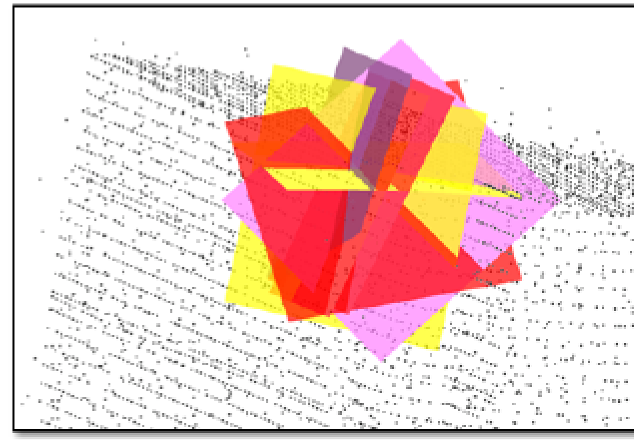
http://www.sci.utah.edu/~shachar/Publications/rmls.pdf

Problems of RMLS

- How to obtain an outlier-free initial region?
  - *Least Median of Squares* (LMS)
  - *k*th ordered statistics is employed in [Fleishman et al., 05] to improve the efficiency
- However, such technique still cannot guarantee to obtain an outlier-free initial region
- Considering about the MDPE based approach – it does not rely on the restrict condition of outlier-free

Comparison of MDPE and RMLS

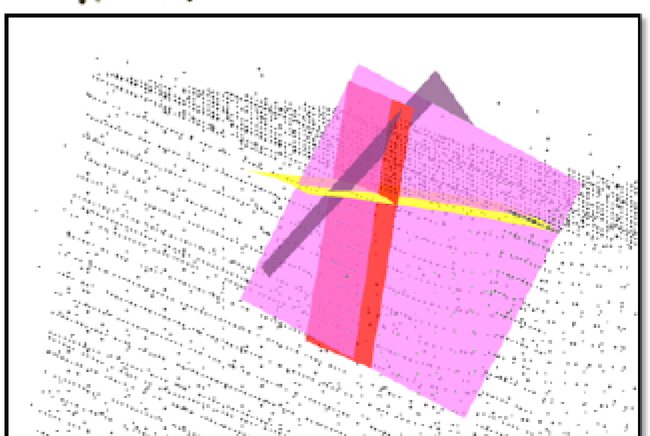




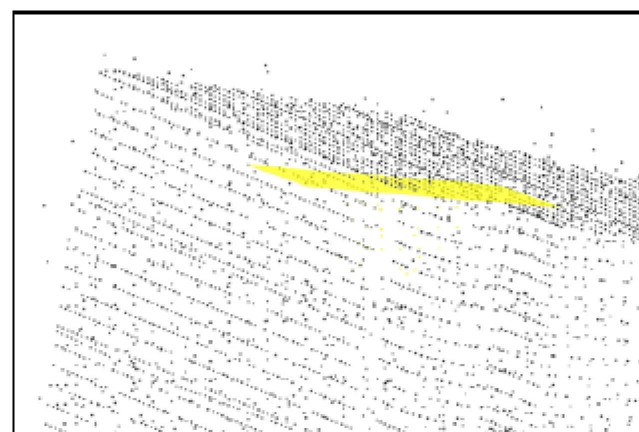
LMS

Fail to obtain outlier-free initial regions
- By Least Median of Squares

One surface mis-classified on outliers



kth order estimator



MDPE

Conclusion

- Normal estimation
  - Principal Component Analysis (PCA)
  - Local surface fitting
  - Consistent orientation
- Denoising by projection – simplified MLS projection
- Outlier removal and processing
  - Heuristic based removal methods
  - Robust statistic based processing
  - Any other suggestions?

Assignment 1 – Point Rendering

- Requirement:
  - To build the hash data structure of point set (e.g., 20 x 20 x 20 boxes)
  - To search *k* neighbors of each point with the help of hashing boxes
  - Using Principal Component Analysis (PCA) to compute the normal of every point by its neighbors
  - To display the point set with normals estimated from PCA

Assignment – Point Rendering (Cont.)

- Change point size in display (*Hint:* how about size of point? how to evaluate?)

```
glPointSize((float)(diameter))
```

- Point display with normal vectors

```
glNormal3f((float)nx,(float)ny,(float)nz);
glVertex3f((float)xx,(float)yy,(float)zz);
```

**Need to turn on the double-side display by*

```
glLightModelf(GL_LIGHT_MODEL_TWO_SIDE, 1.0);
```

- Change rectangular points into circular points

```
glEnable(GL_POINT_SMOOTH);
```

// without this, the rectangle will be displayed for point