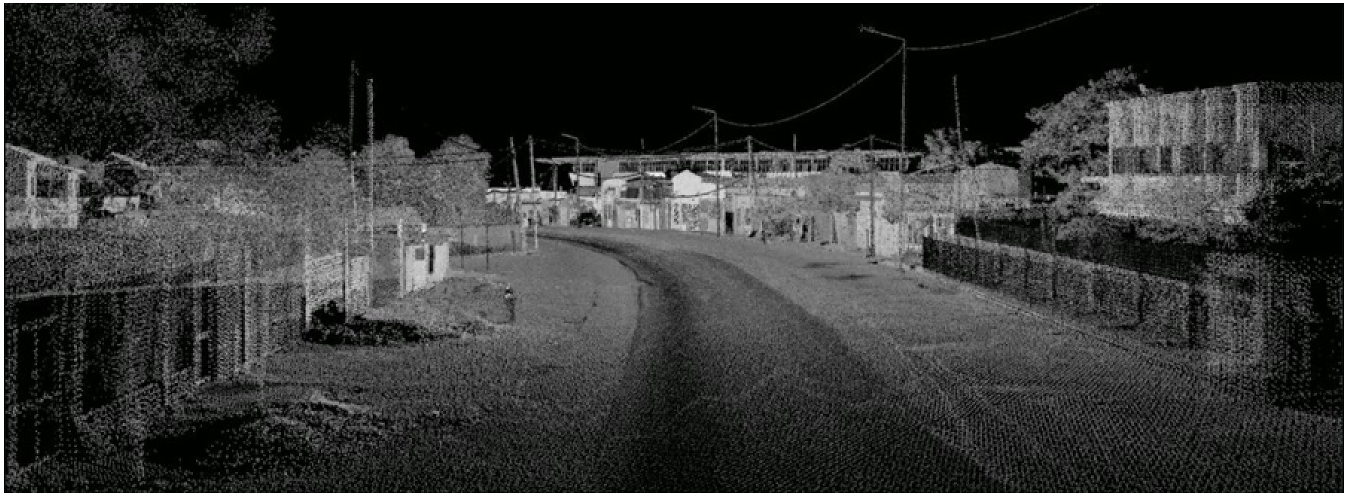
More Point Clouds

Siddhartha Chaudhuri http://www.cse.iitb.ac.in/~cs749

Point Clouds

- Sample points from a shape
- Simple and “raw” representation
- Can be acquired from real or virtual data
- We discussed how to sample efficiently (both time and space), for accurate coverage



Today

Extracting structure from point clouds

The geometry of a point cloud

- How large is it?
- What region does it cover?
- How many components does it have?
- What is the local shape?
- What is the global shape?
- ...

How large is it?

- A simple measure: **Axis-Aligned Bounding Box (AABB)**

```
class Vec3 {
  public:
    double x, y, z;

    // constructors...
    // operators...
};
```

```
class AABB {
  private:
    bool empty;
    Vec3 lo, hi;

  public:
    AABB() : empty(true) {}
    AABB(Vec3 l, Vec3 h)
      : empty(false), lo(l), hi(h) {}

    bool isEmpty() { return empty; }
    Vec3 const & low() const { return lo; }
    Vec3 const & high() const { return hi; }

    void merge(Vec3 const & v) {
      if (empty) {
        lo = hi = v;
        empty = false;
      } else {
        lo = lo.min(v);
        hi = hi.max(v);
      }
    }
};
```

How large is it?

- A simple measure:
  Axis-Aligned Bounding Box (AABB)

```
AABB box;
for (size_t i = 0;
     i < points.size();
     ++i)
{
  box.merge(points[i]);
}
```

How large is it?

- A simple measure:
  Axis-Aligned Bounding Box (AABB)

```
AABB box;
for (size_t i = 0;
     i < points.size();
     ++i)
{
  box.merge(points[i]);
}
```

How large is it?

- A better measure: **Oriented Bounding Box (OBB)**

```
class OBB {
  private:
    AABB aabb;
    CoordinateFrame aabb;

  public:
    AABB const &
    getLocalAABB() const
    { return aabb; }

    CoordinateFrame const &
    getLocalFrame() const
    { return frame; }
};
```

How large is it?

- A better measure: **Oriented Bounding Box (OBB)**

```
class OBB {
  private:
    AABB aabb;
    CoordinateFrame aabb;

  public:
    AABB const &
    getLocalAABB() const
    { return aabb; }

    CoordinateFrame const &
    getLocalFrame() const
    { return frame; }
};
```

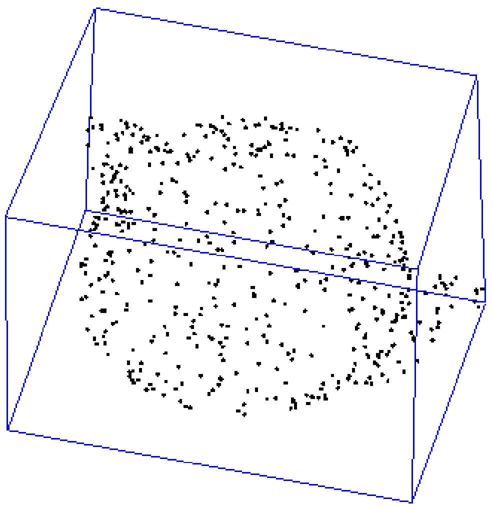
How large is it?

- A better measure: **Oriented Bounding Box (OBB)**

```
class OBB {
  private:
    AABB aabb;
    CoordinateFrame aabb;

  public:
    AABB const &
    getLocalAABB() const
    { return aabb; }

    CoordinateFrame const &
    getLocalFrame() const
    { return frame; }
};
```



Box with minimum volume
(63.48 units)

How large is it?

- Instead of bounding dimensions (sensitive to outliers), find the degrees of largest variance in the data
  - **Principal Component Analysis (PCA)**
    - Data is a set $X = \{\mathrm{x}_1, \mathrm{x}_2, \ldots \mathrm{x}_n\}$ of d-dimensional points (here $d = 3$)
    - Assume the points have mean zero (if not, subtract the centroid $\bar{\mathrm{x}}$ of the points first)
    - Find unit vector w_1 such that X has the largest variance when projected onto w_1, then unit vector w_2 such that the remaining dimensions of X have the largest variance when projected onto w_2, and so on until w_d

Principal Component Analysis

- First principal component: unit vector w_1 such that X has the largest variance when projected onto w_1

$$\underset{\|\mathbf{w}\|=1}{\arg\max} \left\{ \sum_i (\mathbf{x}_i \cdot \mathbf{w})^2 \right\} = \underset{\|\mathbf{w}\|=1}{\arg\max} \left\{ \mathbf{w}^T \left(\sum_i \mathbf{x}_i \mathbf{x}_i^T \right) \mathbf{w} \right\}$$

Note: The actual variance is $\Sigma_i(\mathbf{x}_i.\mathbf{w})^2 / (n - 1)$, but for the maximization we can drop the $n - 1$ for convenience

- Maximized when w_1 is the eigenvector for the largest eigenvalue of $\sum_i \mathbf{x}_i \mathbf{x}_i^T$

Dividing this by $n - 1$ gives the covariance matrix of the (zero-mean) data

  - This eigenvalue (divided by n) gives the variance

- Subsequent eigenvalues yield remaining principal components

Maximizing quadratic form

- Claim: $\mathbf{w}^T \left(\sum_i \mathbf{x}_i \mathbf{x}_i^T \right) \mathbf{w}$ is maximized, for $\|\mathbf{w}\| = 1$, when $\mathbf{w}$ is the eigenvector for the largest eigenvalue of $A = \sum_i \mathbf{x}_i \mathbf{x}_i^T$
- Proof:
  - A is a real symmetric matrix, so can be diagonalized as $A = UDU^T$, where U is an orthonormal matrix of eigenvectors, and D is a diagonal matrix of real eigenvalues
  - $\mathbf{w}^T A \mathbf{w} = \mathbf{w}^T UDU^T \mathbf{w} = \mathbf{u}^T D \mathbf{u}$, where $\mathbf{u} = U^T \mathbf{w}$
  - Also, $\|\mathbf{u}\| = \mathbf{u}^T \mathbf{u} = \mathbf{w}^T UU^T \mathbf{w} = \mathbf{w}^T \mathbf{w} = \|\mathbf{w}\|$
    - So $\|\mathbf{w}\| = 1$ iff $\|\mathbf{u}\| = 1$

Maximizing quadratic form

- Claim: $\mathbf{w}^T\left(\sum_i \mathbf{x}_i\mathbf{x}_i^T\right)\mathbf{w}$ is maximized, for $\|\mathbf{w}\| = 1$, when $\mathbf{w}$ is the eigenvector for the largest eigenvalue of $A = \sum_i \mathbf{x}_i\mathbf{x}_i^T$
- Proof (contd):

$$
\begin{aligned}
\max_{\|\mathbf{w}\|=1}\left\{\mathbf{w}^T A\mathbf{w}\right\} &= \max_{\|\mathbf{u}\|=1}\left\{\mathbf{u}^T D\mathbf{u}\right\} \\
&= \max_{\mathbf{z}} \sum_{i=1}^{d} z_i D_{ii}, \qquad \text{for } z_i \geq 0,\ \sum_{i=1}^{d} z_i = 1
\end{aligned}
$$

(writing $z_i = u_i^2$)

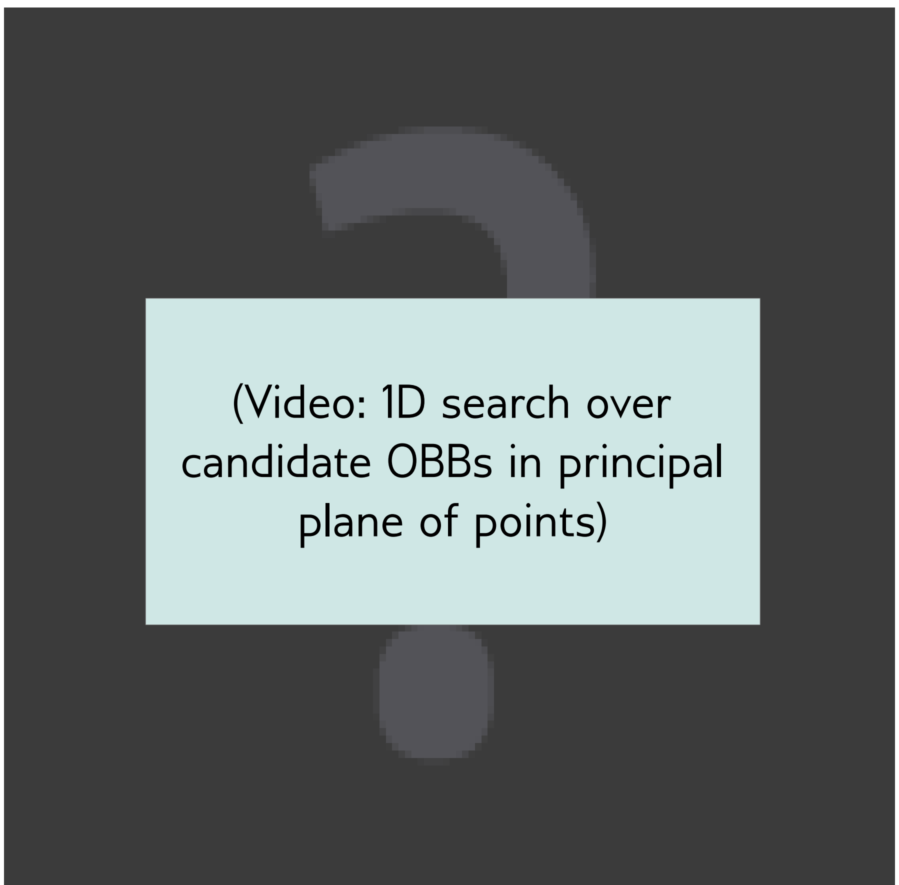
- Convex combination of real numbers, whose maximum is largest eigenvalue D_{kk}, and minimum is smallest eigenvalue D_{hh}
- At the maximum, $z_k = 1$, all other z_i's $= 0$. Plugging this into $U\mathbf{u} = \mathbf{w}$ (remember $U^{-1} = U^T$) directly gives the eigenvector ■

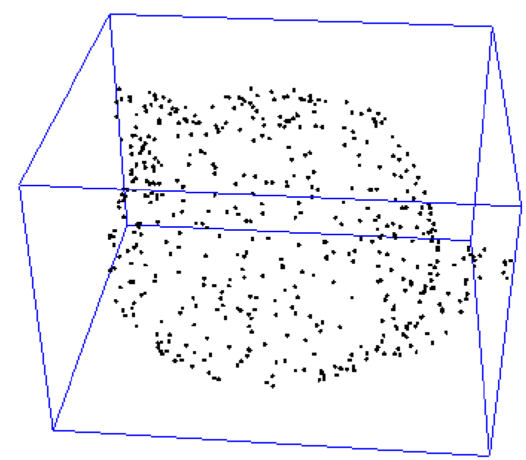
How large is it?

- Faster OBB approximation: Consider only boxes parallel to the plane of the largest two principal components

How large is it?

- Faster OBB approximation: Consider only boxes parallel to the plane of the largest two principal components





Box with minimum volume
(65.31 units)

Thought for the Day #1

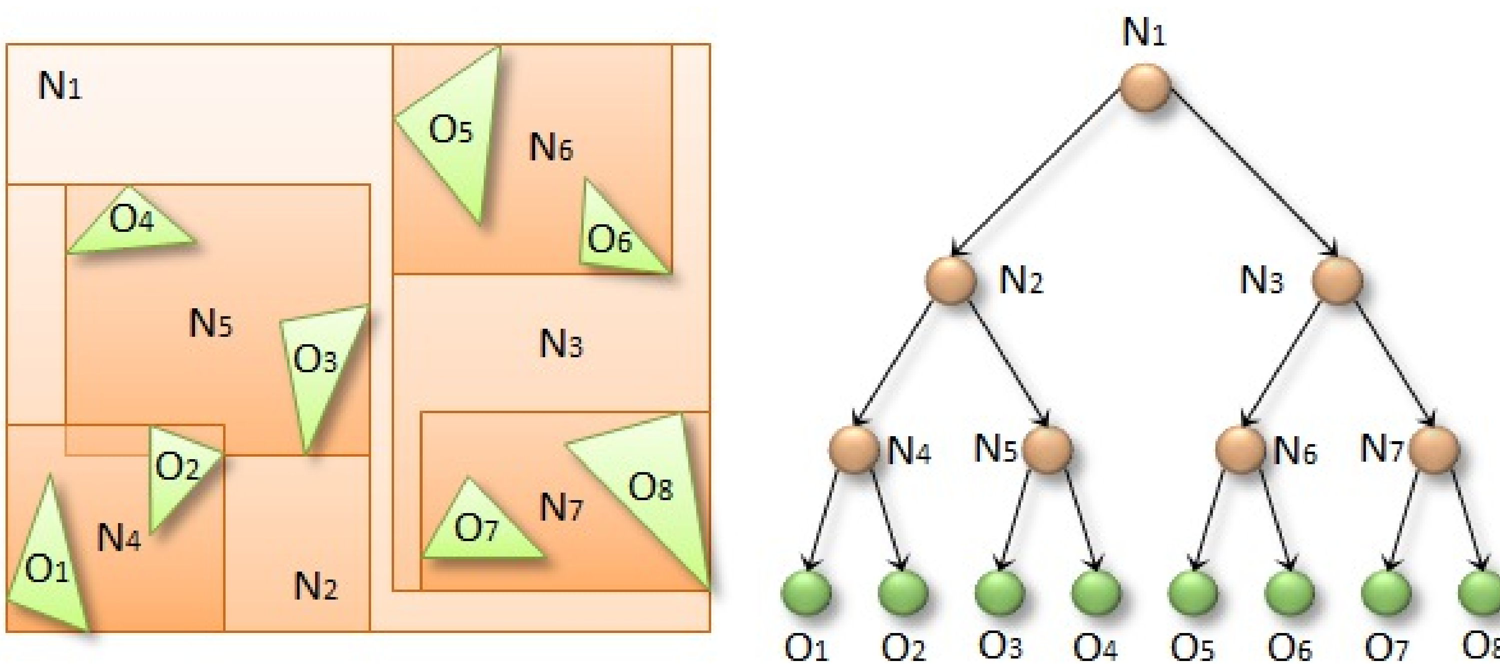
Find a distribution of points whose OBB is not aligned with all the PCA directions

What region does it cover?

- Typical spatial queries:
  - Nearest Neighbor
    - Find the closest point to $\mathbf{x}$
    - k-Nearest Neighbors: Find the k points closest to $\mathbf{x}$
  - Range Query
    - Find all points lying within a box, or sphere, or other range
- Brute force is very slow (linear in #points)
- Both queries can be answered efficiently with a **bounding volume hierarchy**, or other acceleration structure

Bounding Volume Hierarchy (BVH)

- Recursive grouping of objects
- Each group is bounded by a simple shape, typically a box or a sphere

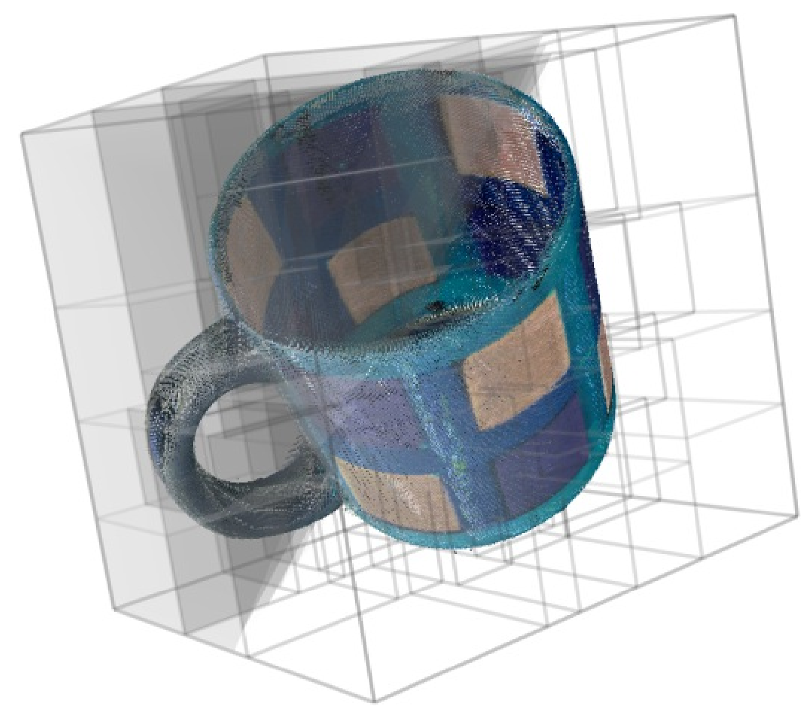


Bounding Volume Hierarchy (BVH)

- Nearest Neighbor:
  - Recurse into subtree only if distance to BV is less than distance to current nearest neighbor
  - When processing the children of a node, start with the one whose BV is closest to the query
- Range Query:
  - Recurse into subtree only if range intersects its BV
- With careful recursion, the query can itself be a BVH → efficient distance between two shapes

k-d Tree

- Recursively split points along coordinate axes
  - Classical strategy: Go through the coordinate axes in sequence and repeat
  - A good practical strategy: Split the longest/most variant dimension each time
  - Where to split?
    - The mean coordinate is quick to compute
    - The median coordinate gives a perfectly balanced partition
- A k-d tree is essentially just a bounding box hierarchy

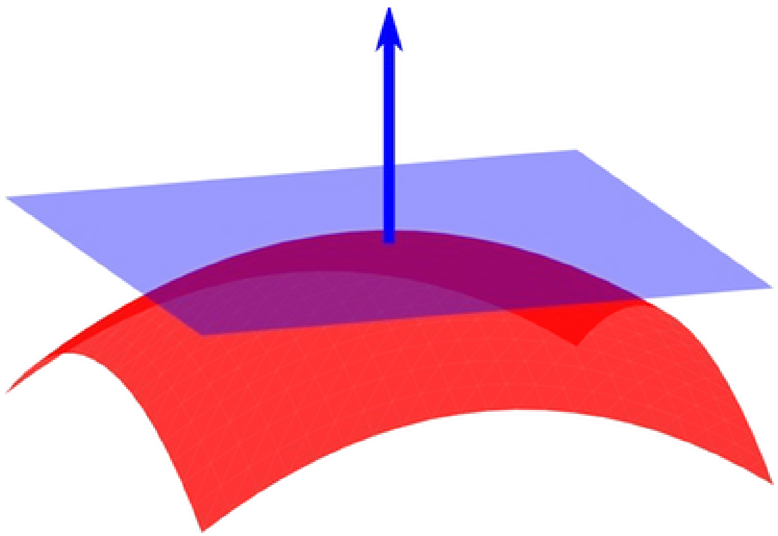


Thought for the Day #2

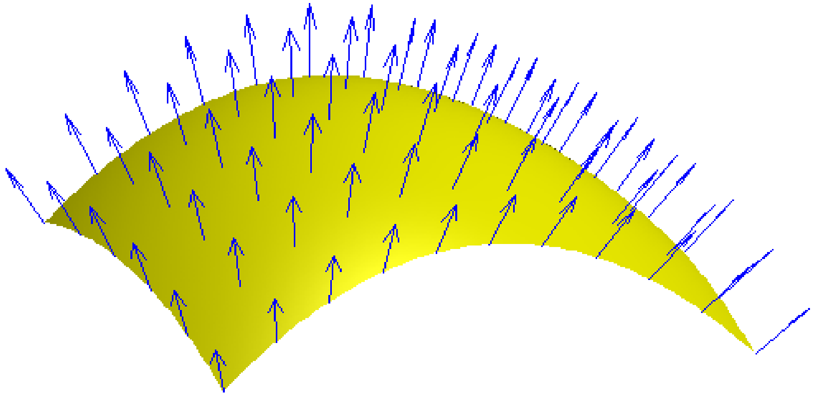
Why doesn't a kd-tree work well for nearest neighbor queries in very high dimensions?

What is the local geometry?

- Estimating the **normals** of the shape



The normal is orthogonal to the local tangent plane

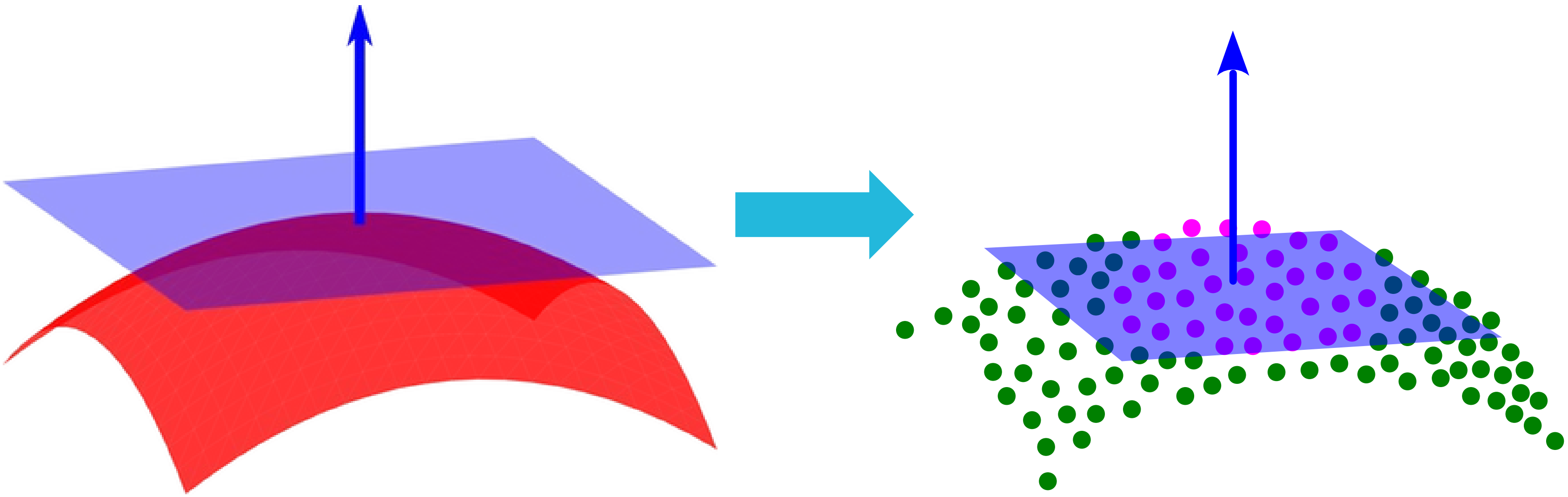


The normals define a vector field over the surface

(positions, colors etc define other vector fields)

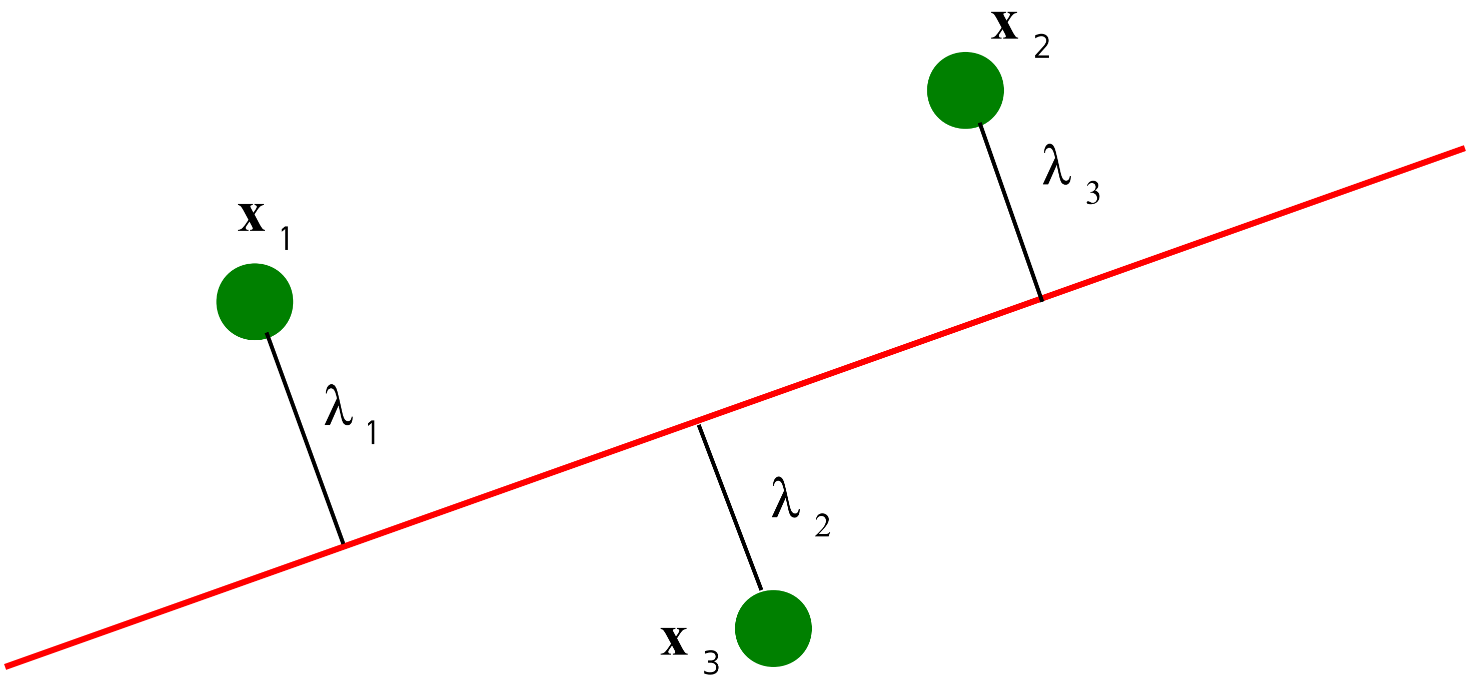
Estimating normals

- **Plane-fitting:** Approximate the tangent plane at a point by the plane best fitting the local neighborhood
  - Assumes surface is ***locally linear***, if you look closely enough



Orthogonal Regression

- Minimize sum of perpendicular distances to plane



minimize $\sum \lambda_i^2$

Orthogonal Regression

- $\lambda_i = |\mathbf{n}.(\mathbf{x}_i - \mathbf{a})|$, where $\mathbf{n}$ is the unit normal to the plane, and $\mathbf{a}$ is some fixed point on the plane
- *Minimize* $\sum \lambda_i^2 = \sum_i ||\mathbf{n}.(\mathbf{x}_i - \mathbf{a})||^2 = \sum_i \mathbf{y}_i^{\mathrm{T}}\, \mathbf{n}\, \mathbf{n}^{\mathrm{T}}\, \mathbf{y}_i$

  $$= \sum_i \mathbf{n}^{\mathrm{T}}\, \mathbf{y}_i\, \mathbf{y}_i^{\mathrm{T}}\, \mathbf{n}$$

  $$= \mathbf{n}^{\mathrm{T}} \sum_i \left(\mathbf{y}_i\, \mathbf{y}_i^{\mathrm{T}}\right)\, \mathbf{n}$$

  for $||\mathbf{n}|| = 1$
- Looks like the PCA problem! Given $\mathbf{a}$, the $\mathbf{y}_i$'s are fully defined and hence $\mathbf{n}$ is the eigenvector for the *smallest* eigenvalue of $\sum_i \left(\mathbf{y}_i\, \mathbf{y}_i^{\mathrm{T}}\right)$

Orthogonal Regression

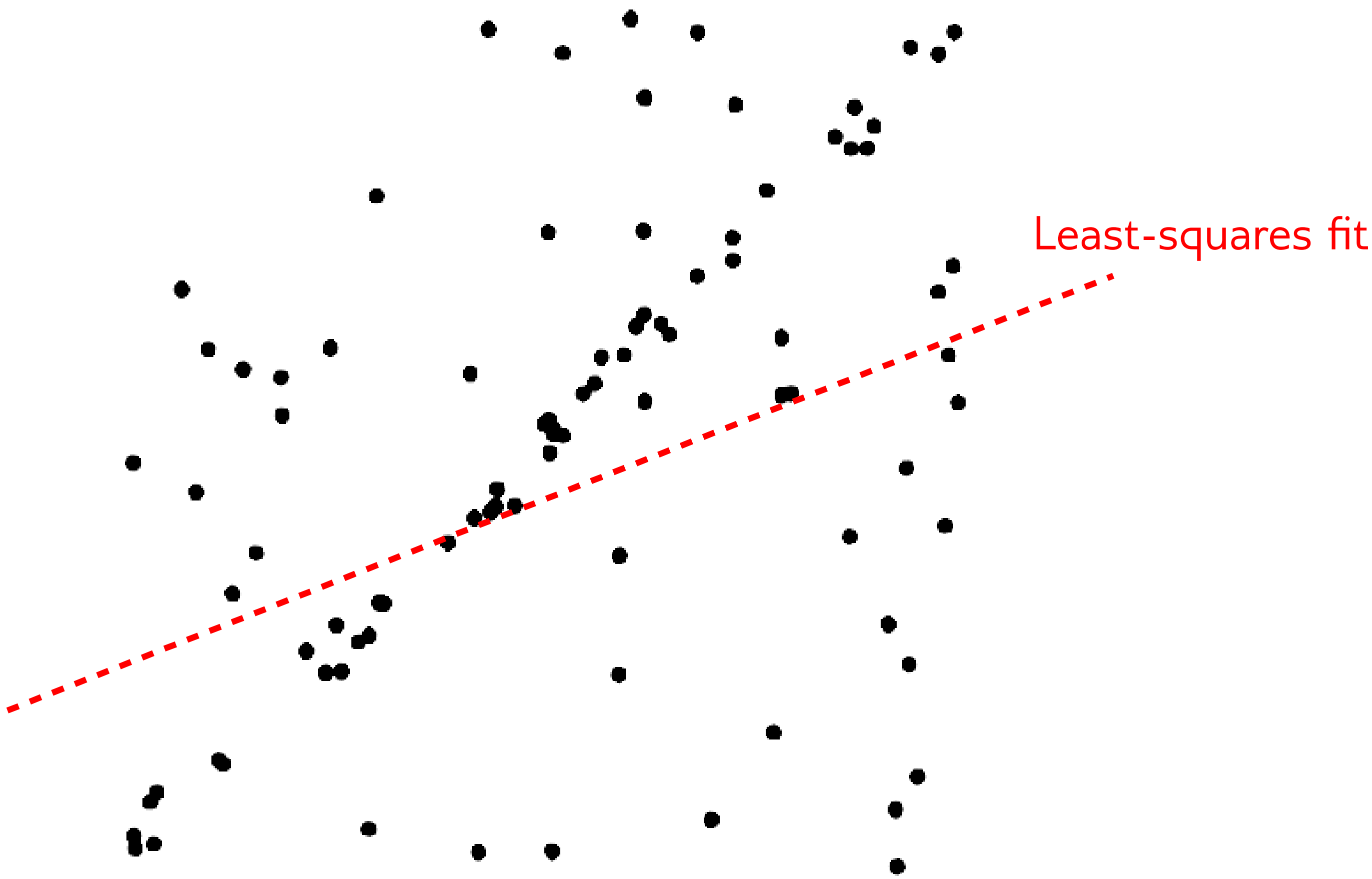
- How to find a fixed point $\mathbf{a}$ on the optimal plane?
- *Recall:* we're minimizing $E = \sum \lambda_i^2 = \sum_i \mathbf{y}_i^{\mathrm{T}} (\mathbf{n}\,\mathbf{n}^{\mathrm{T}})\, \mathbf{y}_i$
- Take the gradient w.r.t. $\mathbf{a}$

$$\frac{\partial E}{\partial \mathbf{a}} = -2\mathbf{n}\mathbf{n}^T \sum_i \mathbf{y}_i$$

- The derivative is zero when $\mathbf{a} = \frac{1}{m} \sum_{i=1}^{m} \mathbf{x}_i$, i.e. $\mathbf{a}$ is the centroid of the points. This completes the definition of the plane.

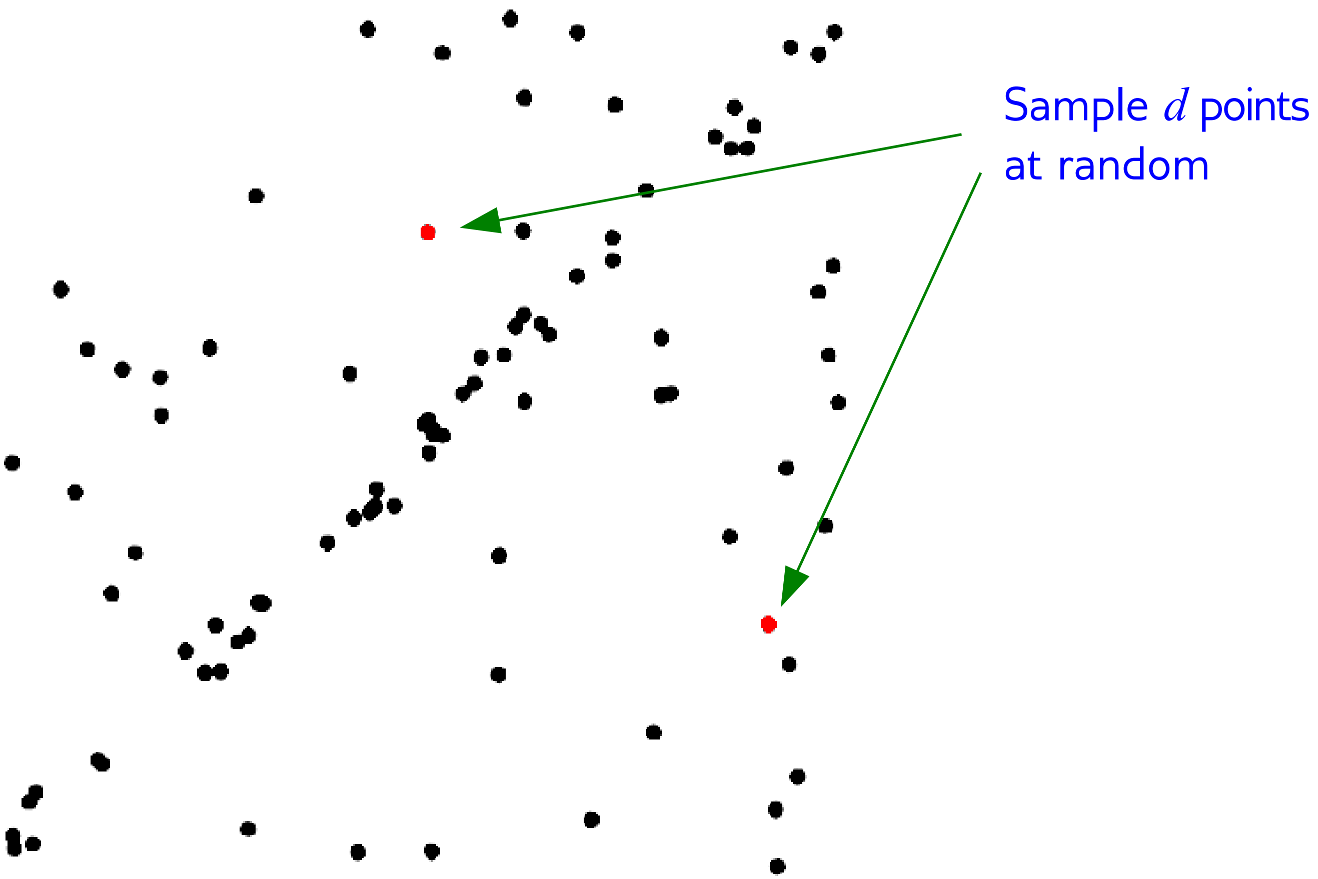
RANSAC

- **RAN**domized **SA**mple **C**onsensus
- **Advantage:** Robust to outliers



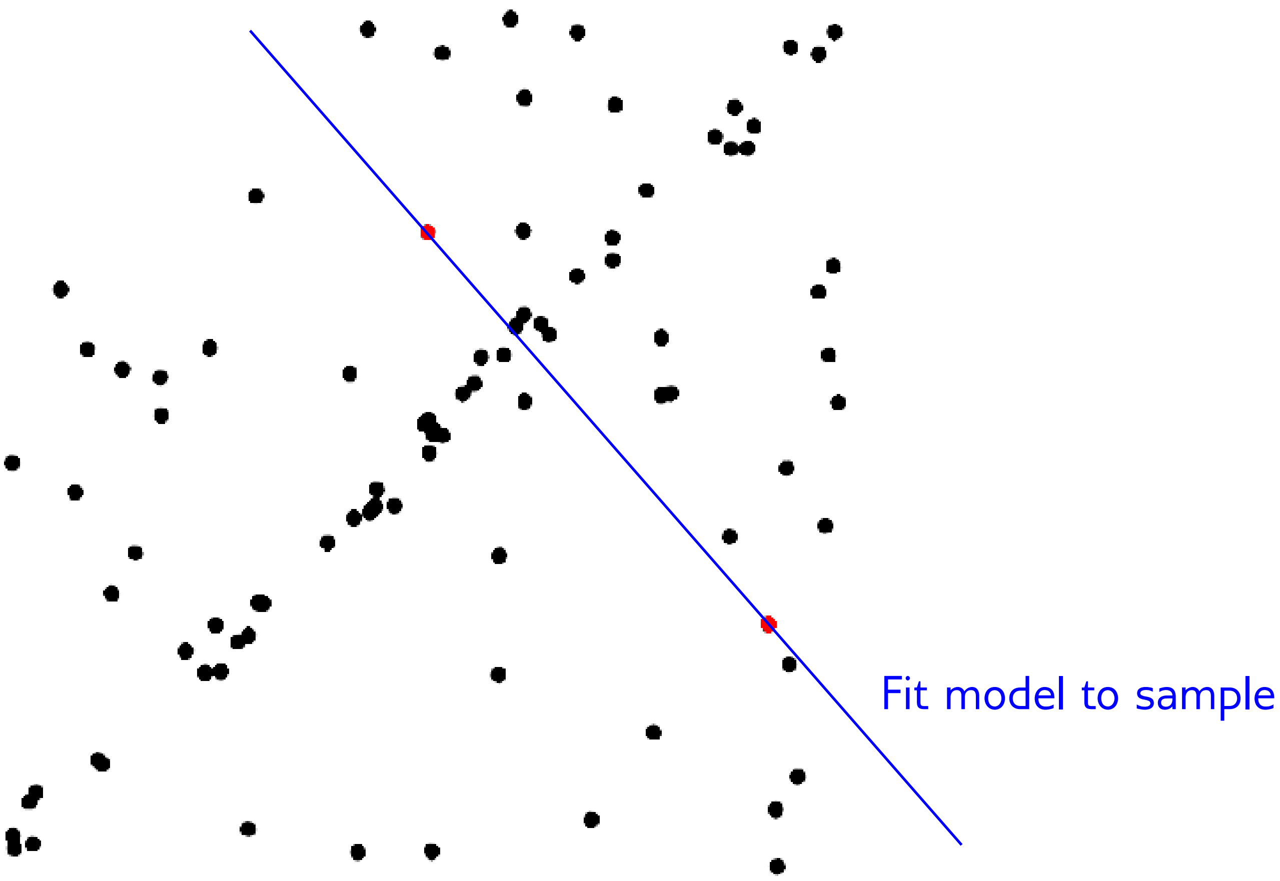
RANSAC

- **RAN**domized **SA**mple **C**onsensus
- **Advantage:** Robust to outliers



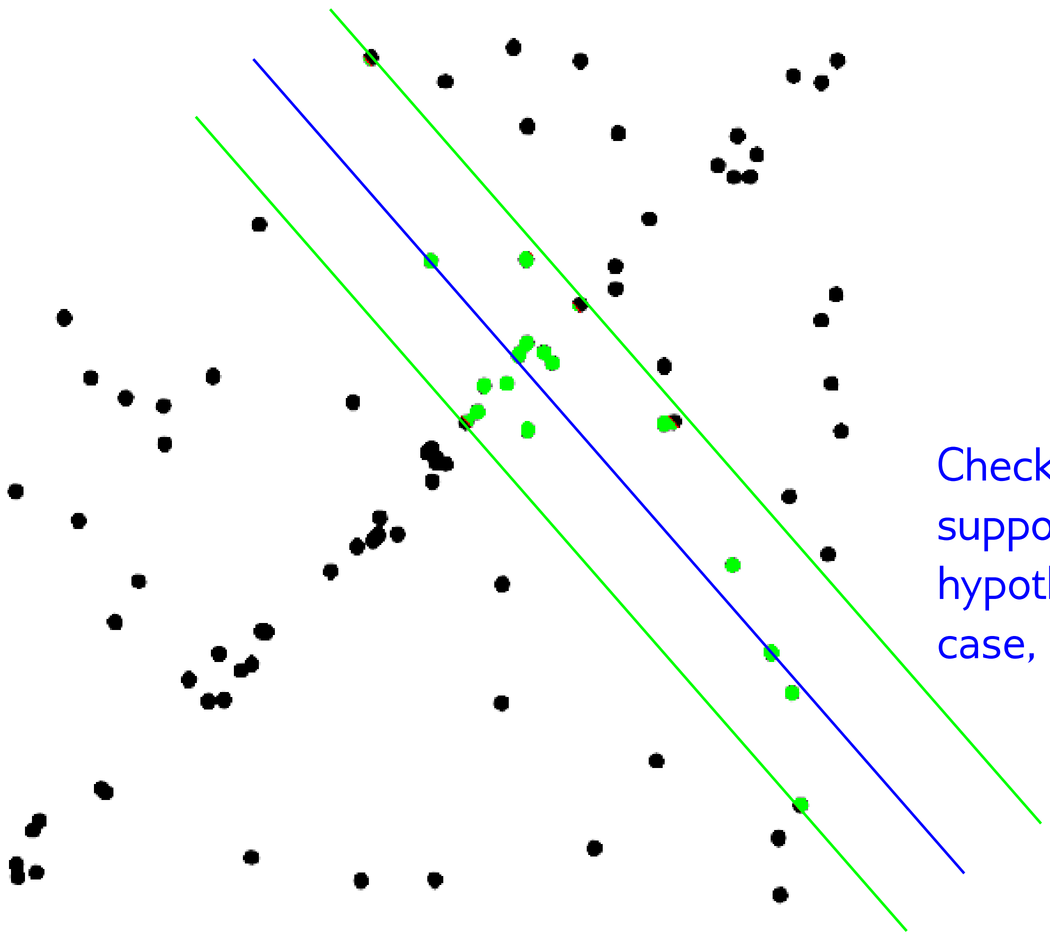
RANSAC

- **RAN**domized **SA**mple **C**onsensus
- **Advantage:** Robust to outliers



RANSAC

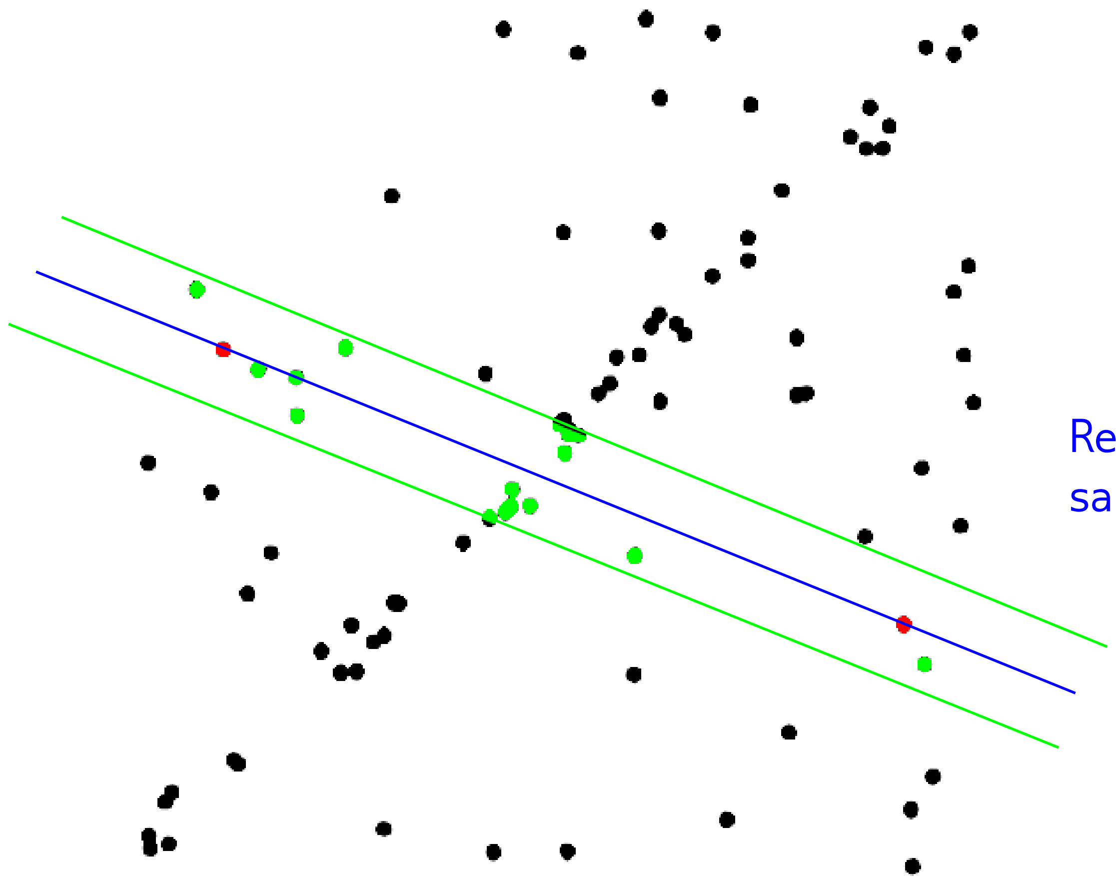
- **RAN**domized **SA**mple **C**onsensus
- **Advantage:** Robust to outliers



Check how much data supports current hypothesis (in this case, very little)

RANSAC

- **RAN**domized **SA**mple **C**onsensus
- **Advantage:** Robust to outliers

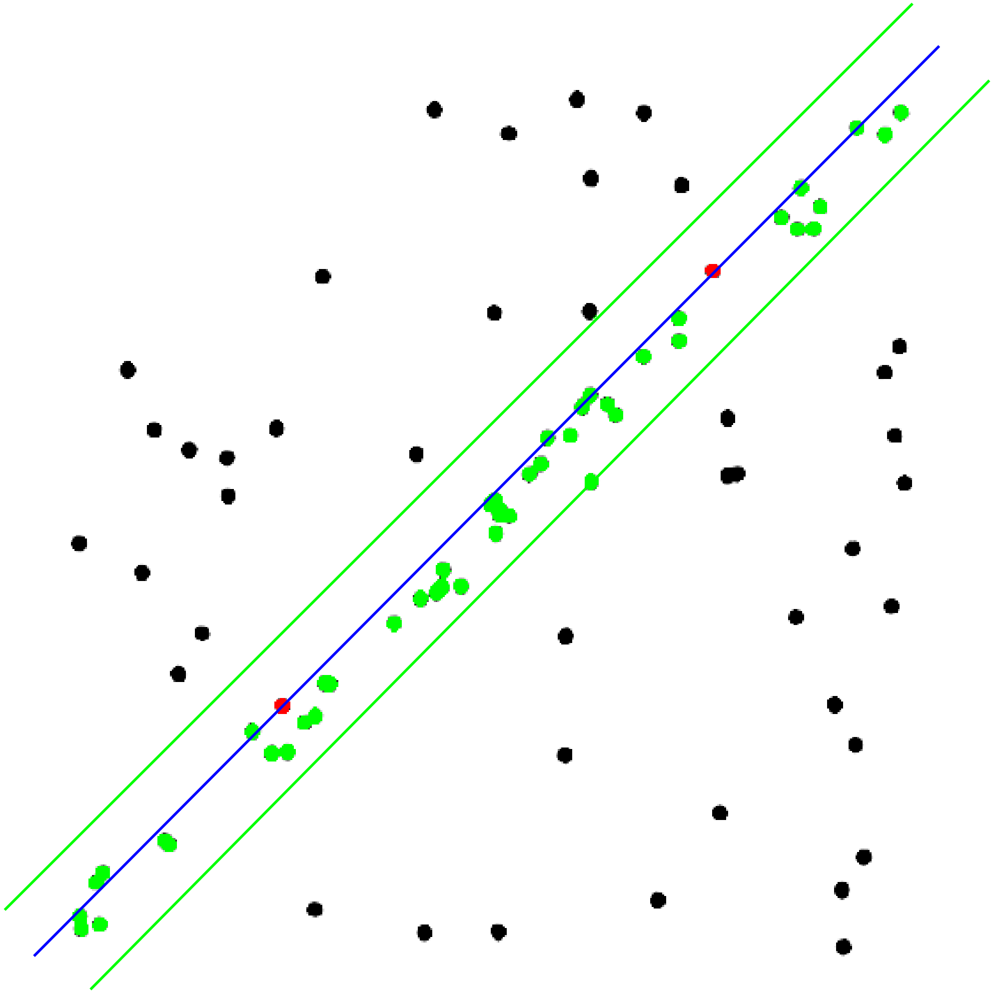


Repeat for a new sample (again no luck)

RANSAC

- **RAN**domized **SA**mple **C**onsensus
- **Advantage:** Robust to outliers

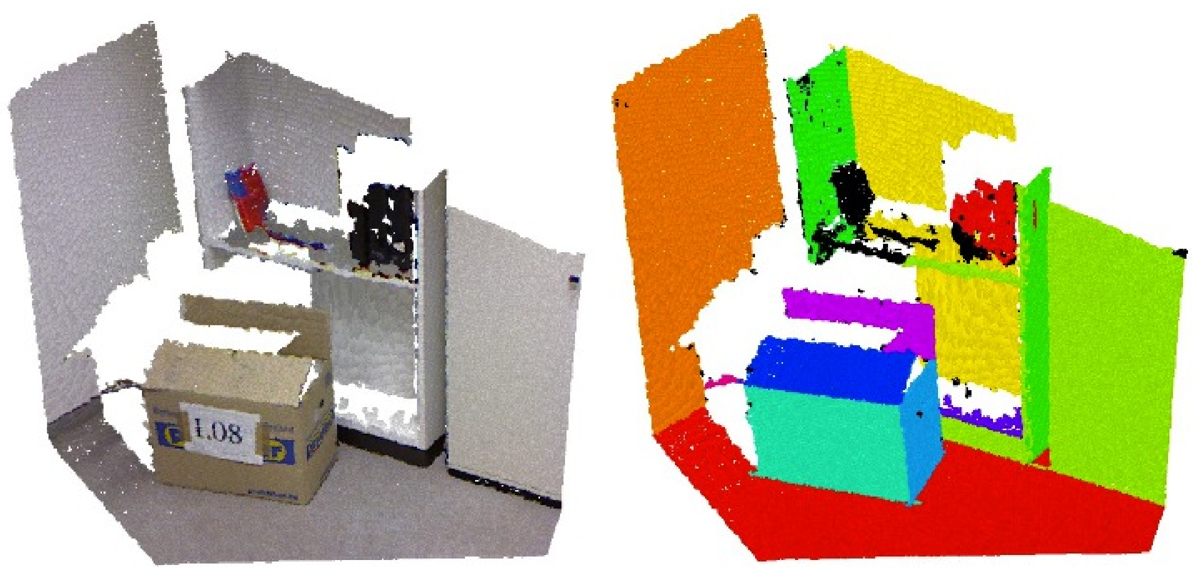
Challenge: Setting the acceptance threshold correctly



Third time lucky!

RANSAC

- **RAN**domized **SA**mple **C**onsensus
- **Advantage:** Robust to outliers
- Popular for detecting planar regions in 3D scans
- Can be used for non-planar fitting as well (any low-dimensional parametric model)

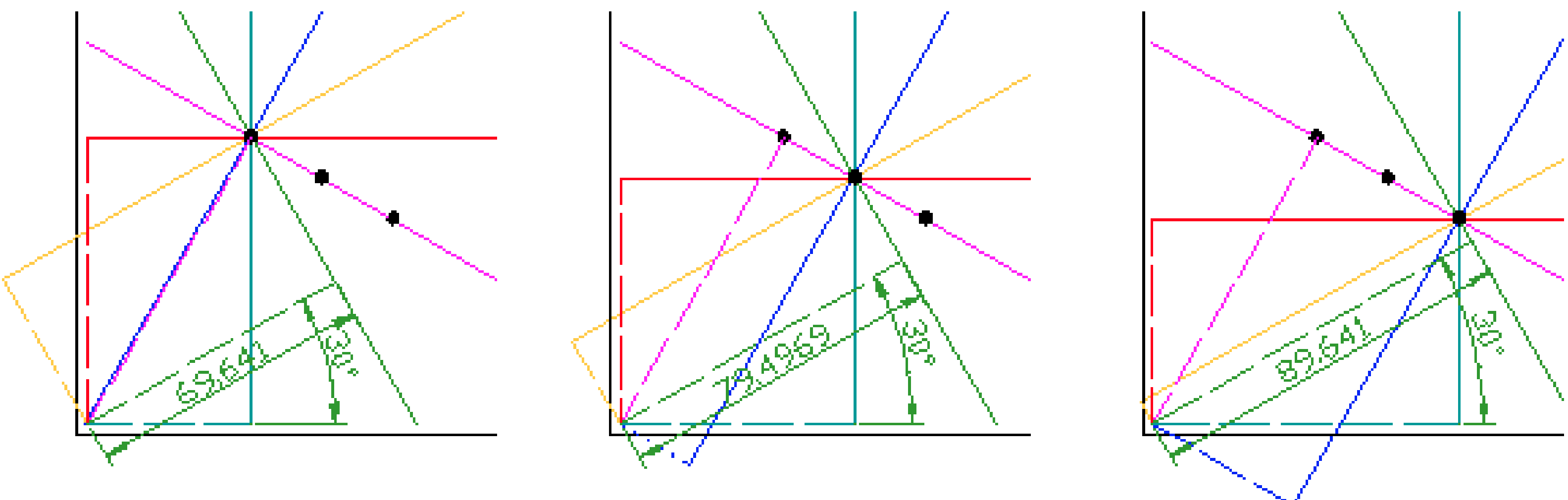


The Hough Transform

- **“Fitting by voting”**
  - Each data point (or small sample of points) votes for all models (here, planes) that it supports
  - The vote is cast in the space of model parameters
  - At the end, look for the (discretized) sets of model parameters with large numbers of votes

The Hough Transform

- Example: Vote for all lines supported by a simple dataset of 3 points



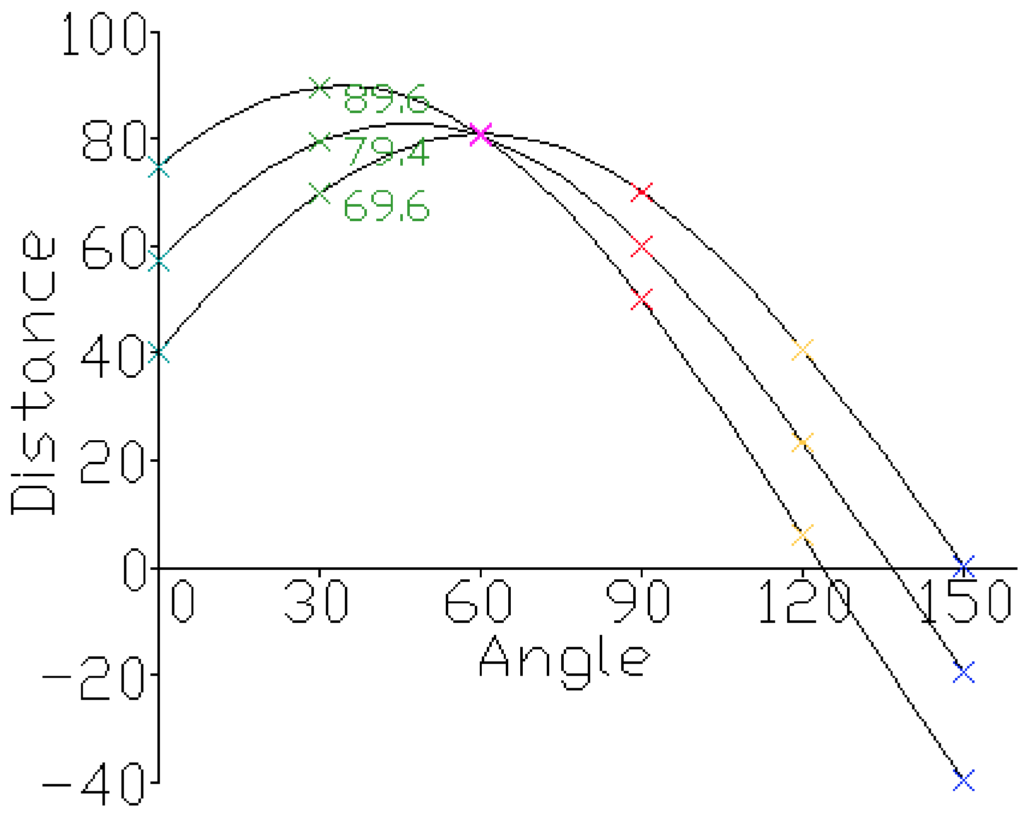
Angle	Dist.
0	40
30	69.6
60	81.2
90	70
120	40.6
150	0.4

Angle	Dist.
0	57.1
30	79.5
60	80.5
90	60
120	23.4
150	-19.5

Angle	Dist.
0	74.6
30	89.6
60	80.6
90	50
120	6.0
150	-39.6

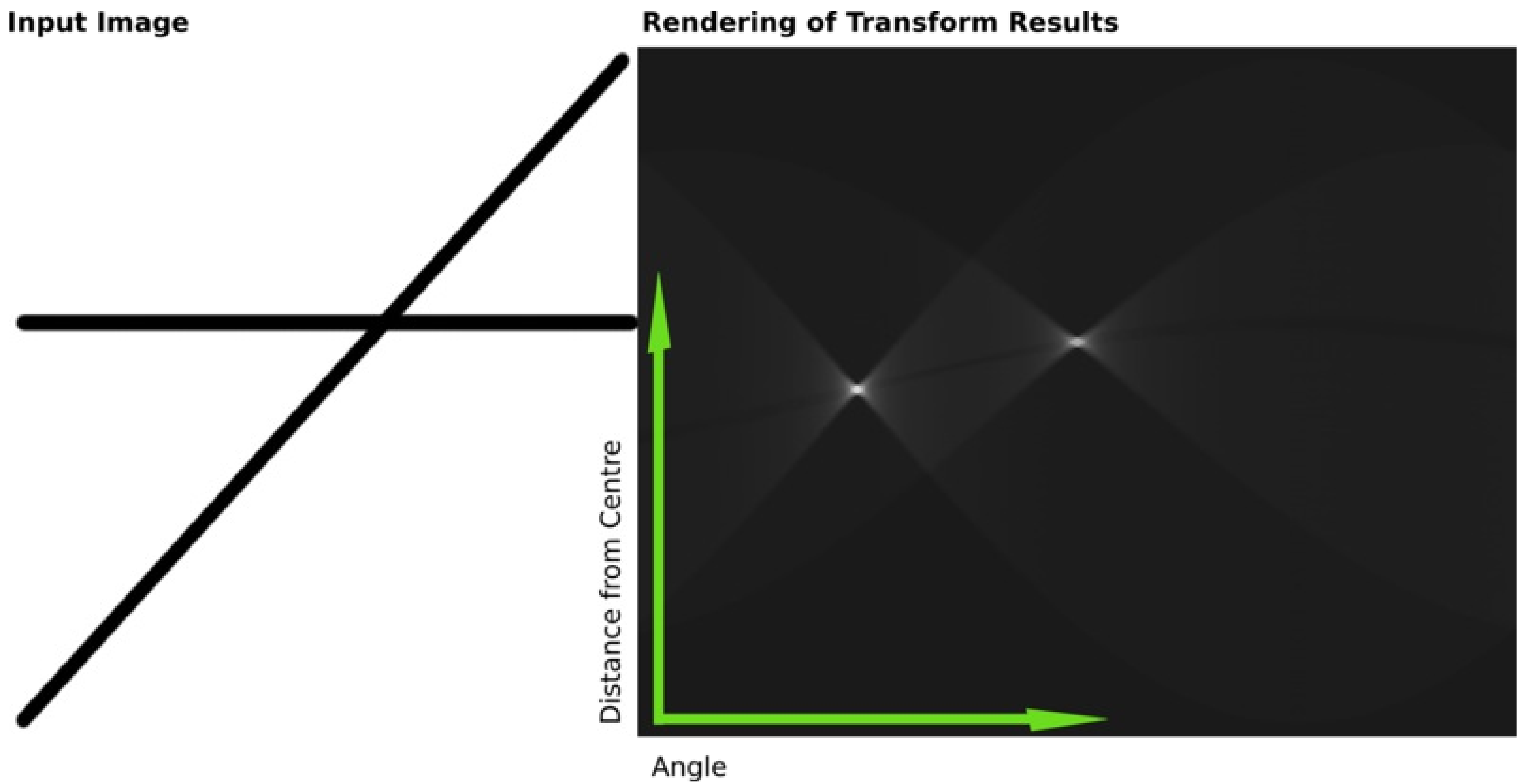
The Hough Transform

- The plot of all the votes in the space of lines (parametrized by angle and distance from origin)



The Hough Transform

- A more complex example



The Hough Transform

- **"Fitting by voting"**
  - Each data point (or small sample of points) votes for all models (here, planes) that it supports
  - The vote is cast in the space of model parameters
  - At the end, look for the (discretized) sets of model parameters with large numbers of votes
- Possible optimization for point clouds: at each point, vote only for planes that are roughly aligned with the estimated local normal

Thought for the Day #3

Why do techniques like RANSAC or the Hough Transform not work well for models with a large number of parameters?