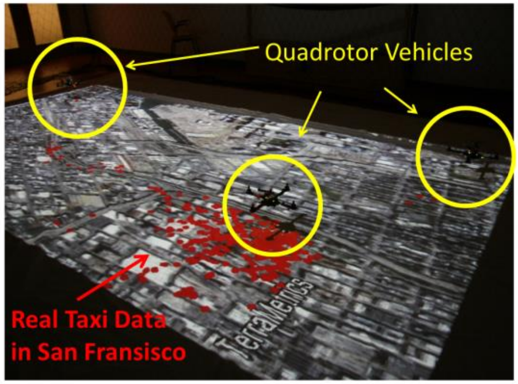
Learning streaming and distributed big data using core-sets



Dan Feldman

Robotics & BigData Lab
University of Haifa

אוניברסיטת חיפה
University of Haifa

אוניברסיטת חיפה
University of Haifa

Challenges of this talk

Forge links between:

- Computational Geometry
- Core-sets
- Machine Learning of Big Data
- Robotics

Challenges of this talk

Forge links between:

- Approximated Caratheodory Theorem
- Core-sets for mean queries
- Google's PageRank
- Real time pose estimation

Big Data

- Volume: huge amount of data points
- Variety: huge number of sensors
- Velocity: data arrive in real-time streaming

Need:

- Streaming algorithms (use logarithmic memory)
- Parallel algorithms (use networks, clouds)
- Simple computations (use GPUs)
- No assumption on order of points

Big Data Computation model

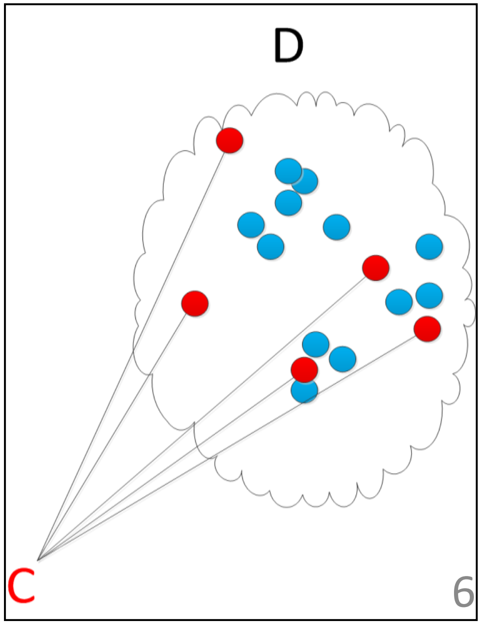
- = Streaming + Parallel computation
- Input: infinite stream of vectors
- $n =$ vectors seen so far
- $\sim\log n$ memory
- *M* processors
- *~log (n)/M* insertion time per point

(Embarrassingly parallel)

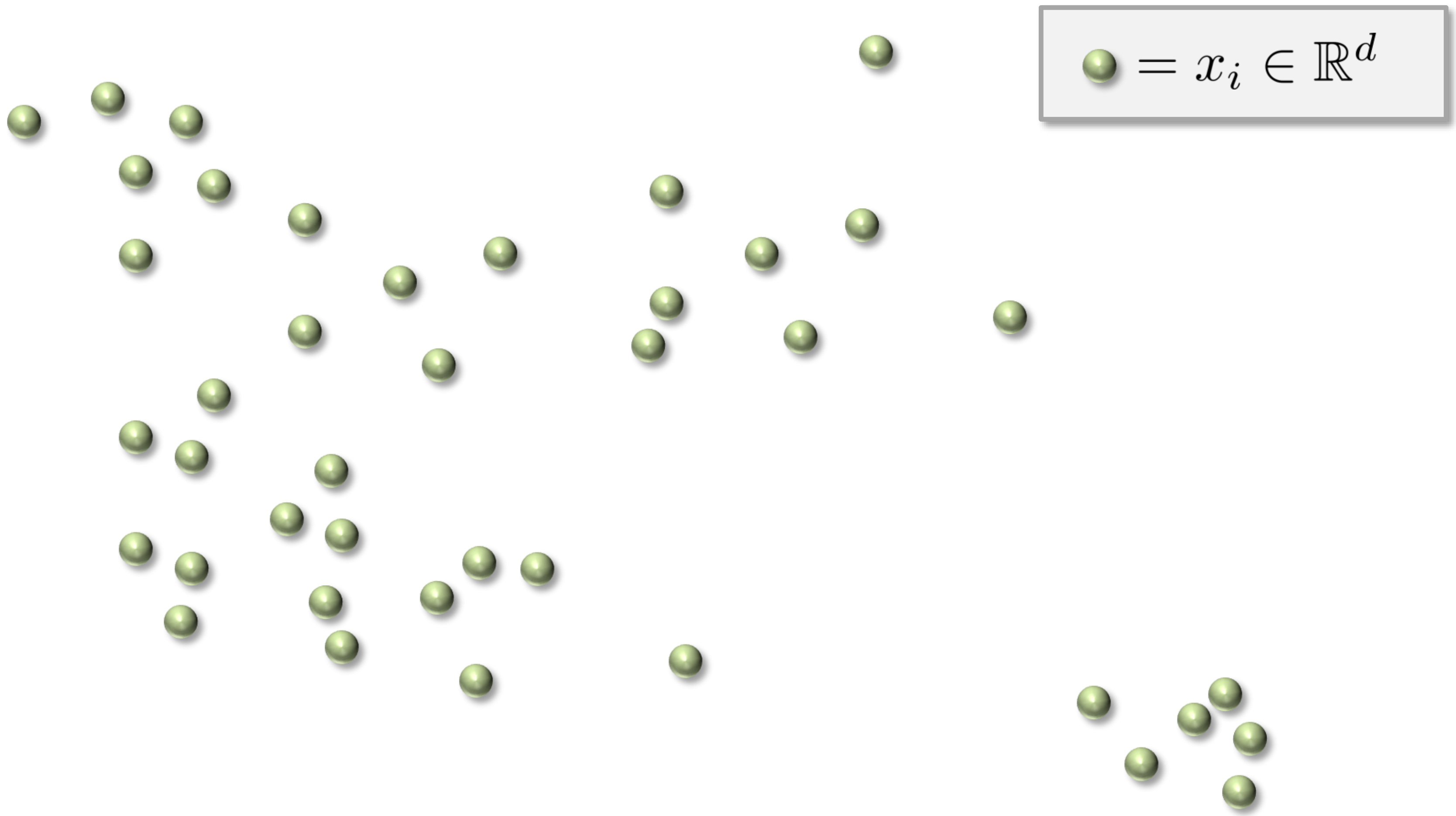
Challenge:
Find RIGHT data from Big Data

Given data *D* and Algorithm *A* with *A(D)* intractable, can we efficiently reduce *D* to *C* so that *A(C)* fast and *A(C)~A(D)*?

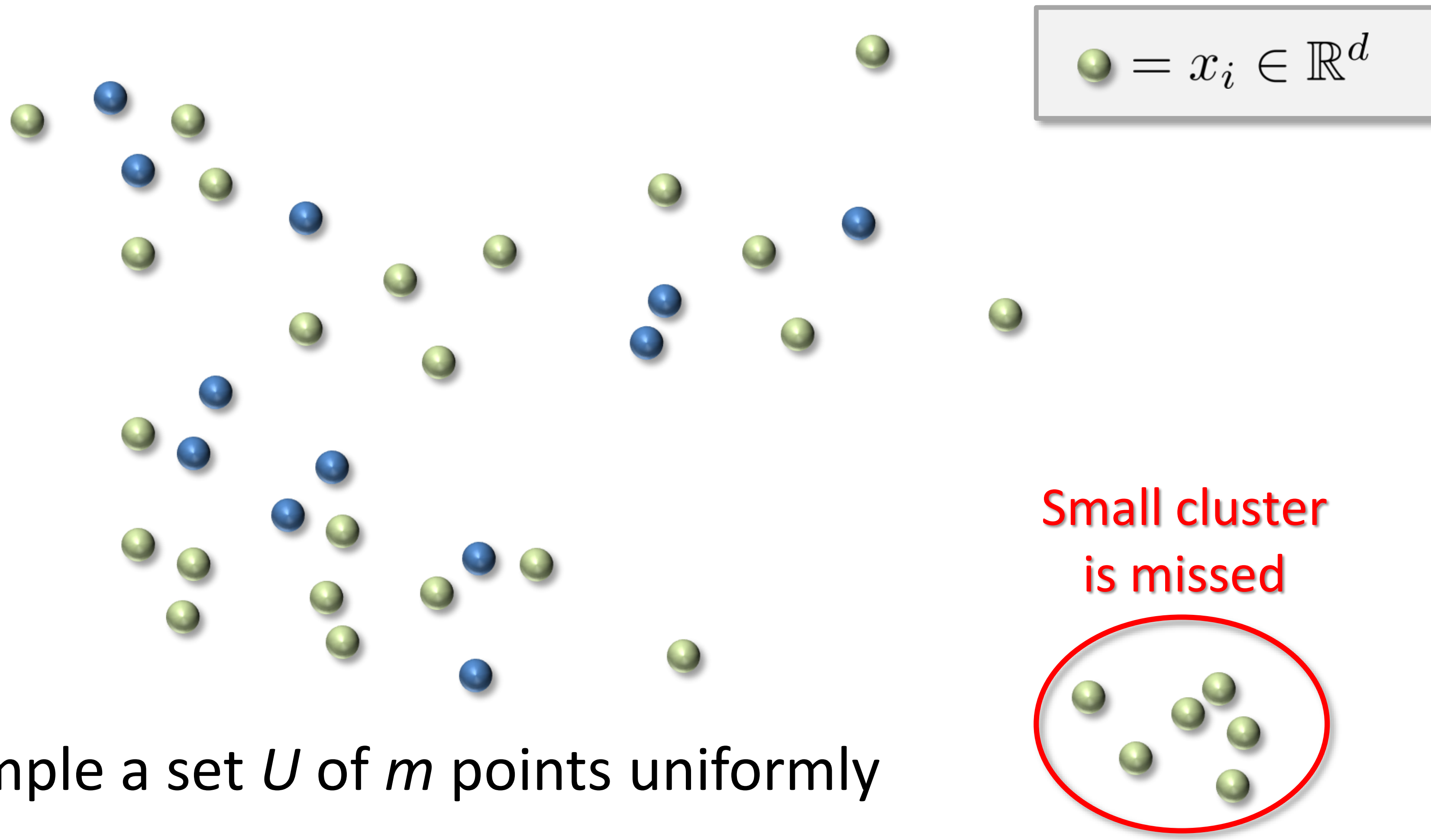
Provable guarantees on approximation with respect to the size of C



Naïve Uniform Sampling (RANSAC)



Naïve Uniform Sampling



Sample a set U of m points uniformly

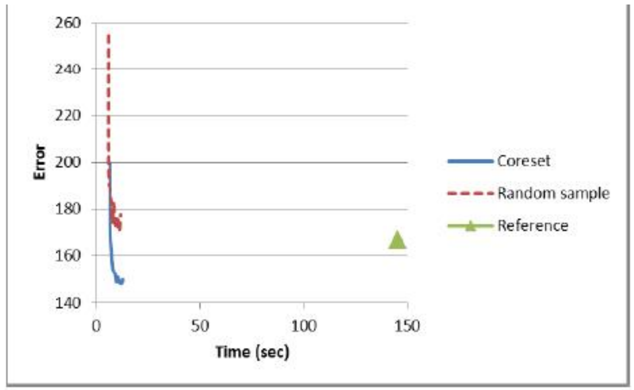
Coreset for Image Denoising

[F, Feigin , Sochen [SSVM'13]

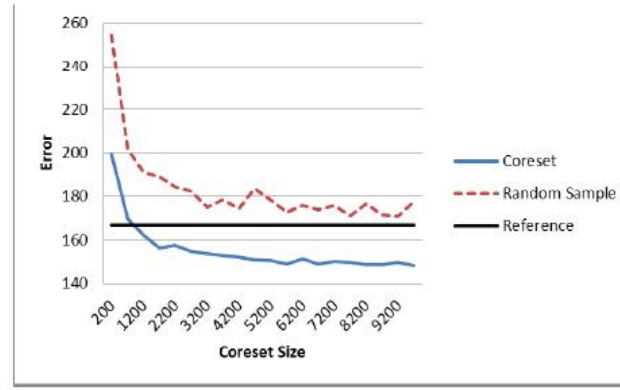
- Existing de-noising algorithms works only on small (low-definition) images off-line
- For HD or real-time streaming:
  Use random sampling (RANSAC)

RANSAC will not find rare but important parts

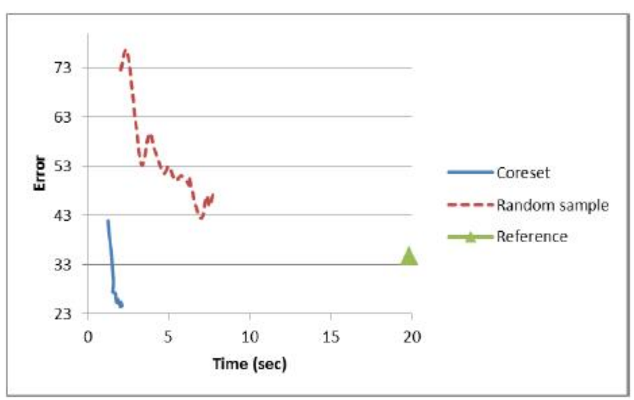
(g) Image

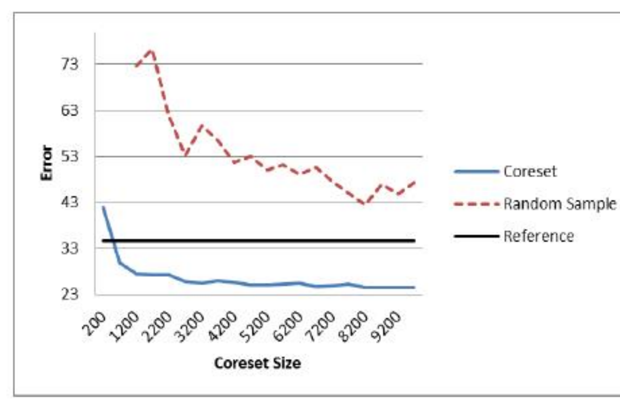


(h) Runtime vs. Quality



(i) Size vs. Quality

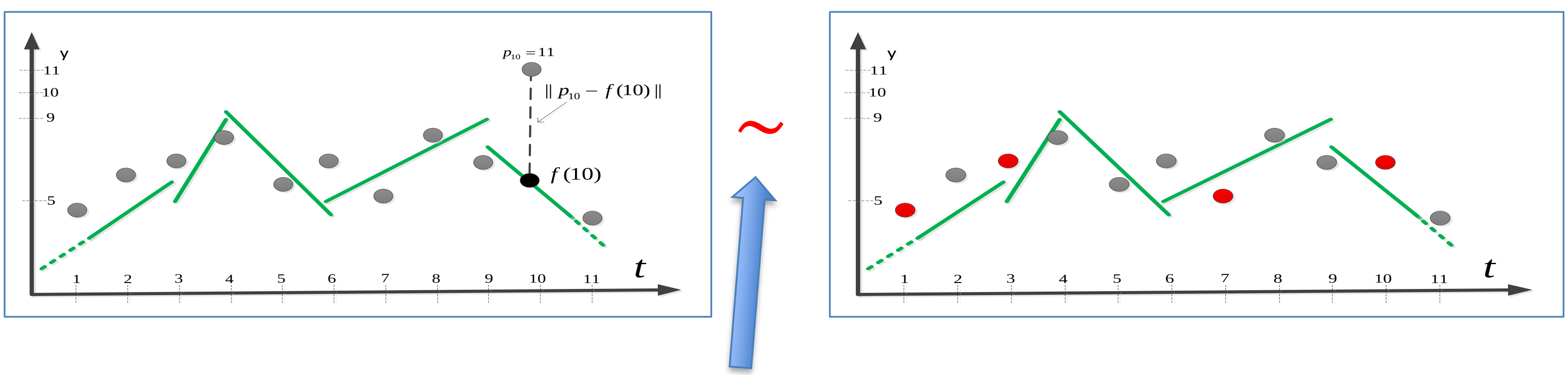




From Big Data to Small Data

Suppose that we can compute such a corset C of size $\frac{1}{\epsilon}$ for every set P of n points

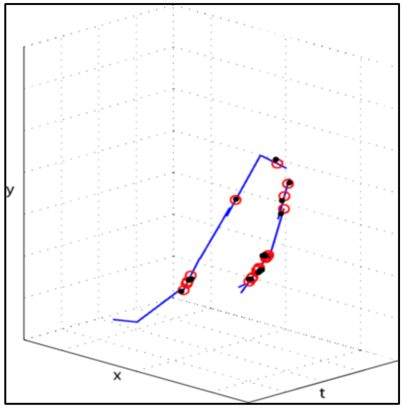
- in time n^3,
- off-line, non-parallel, non-streaming algorithm



$(1 \pm \epsilon)$

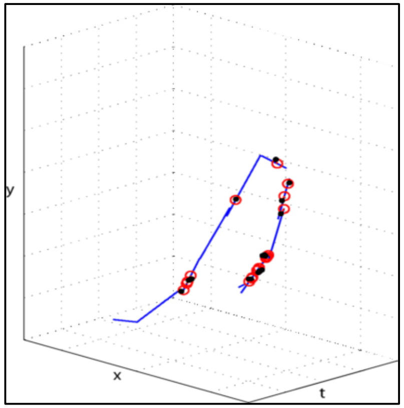
Read the first $\frac{2}{\epsilon}$ streaming points and reduce them into $\frac{1}{\epsilon}$ weighted points in time $\left(\frac{2}{\epsilon}\right)^5$

$1+\epsilon$ corset for P_1

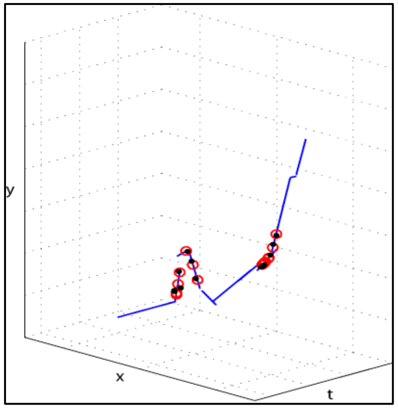


Read the next $\frac{2}{\epsilon}$ streaming point and reduce them into $\frac{1}{\epsilon}$ weighted points in time $\left(\frac{2}{\epsilon}\right)^5$

$1+\epsilon$ corset for P_1

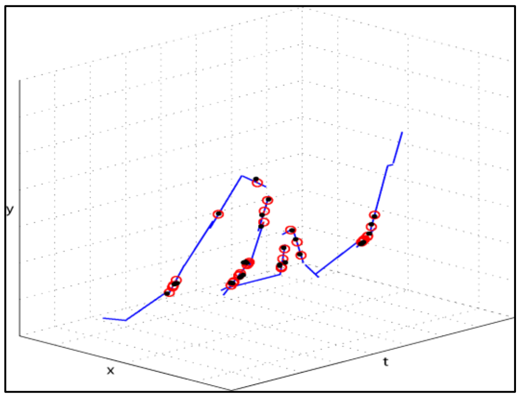


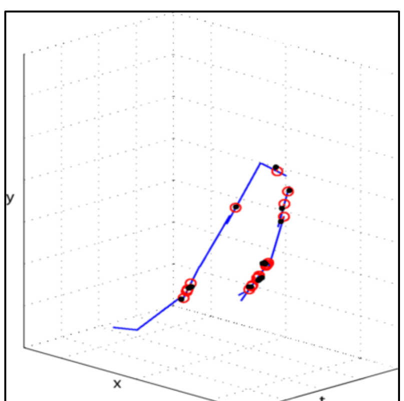
$1+\epsilon$ corset for P_2

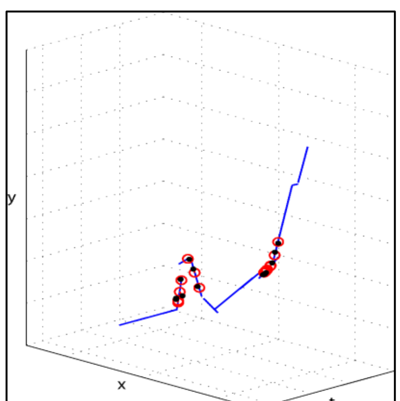


Merge the pair of ϵ-coresets into an ϵ-corset of $\frac{2}{\epsilon}$ weighted points

$1 + \epsilon$-corset for $P_1 \cup P_2$

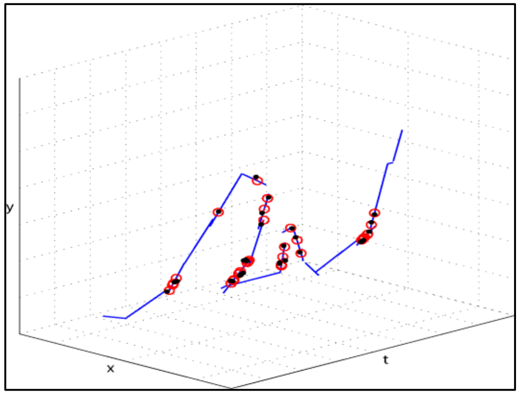


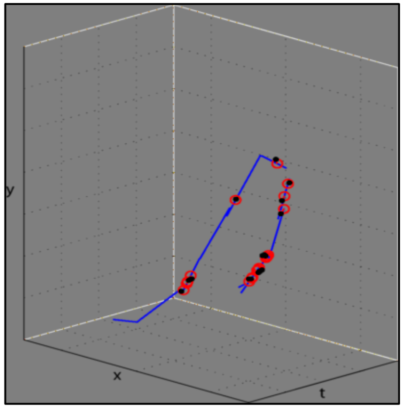


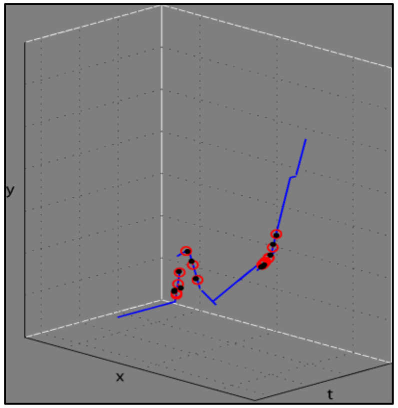


Delete the pair of original coresets from memory

$1 + \epsilon$-corset for $P_1 \cup P_2$



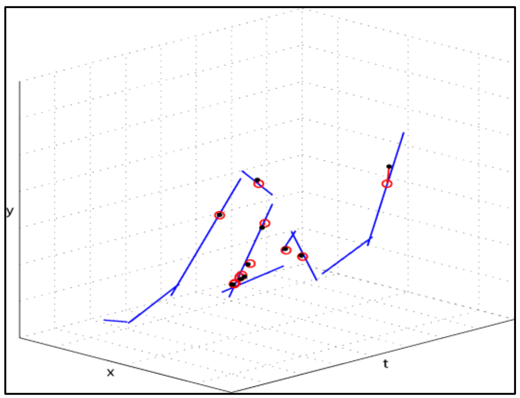


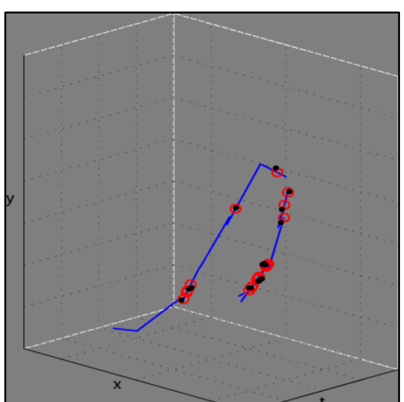


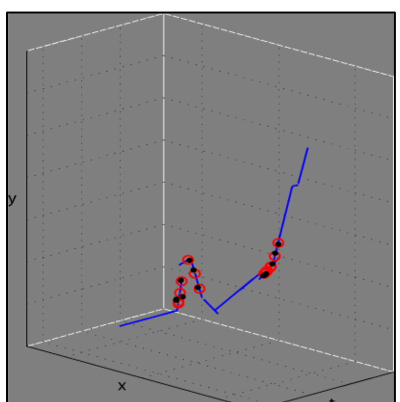
Reduce the $\frac{2}{\epsilon}$ weighted points into $\frac{1}{\epsilon}$ weighted points by constructing their coreset

$1+\epsilon$-corset for

$1+\epsilon$-corset for $P_1 \cup P_2$



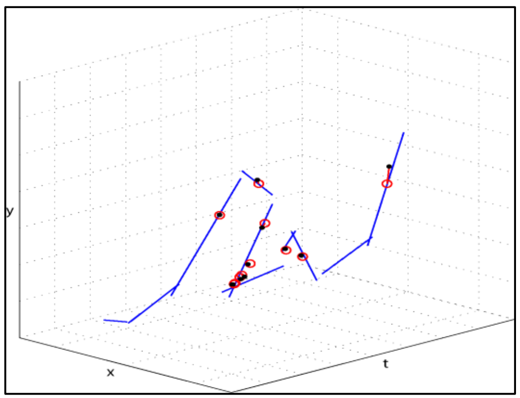




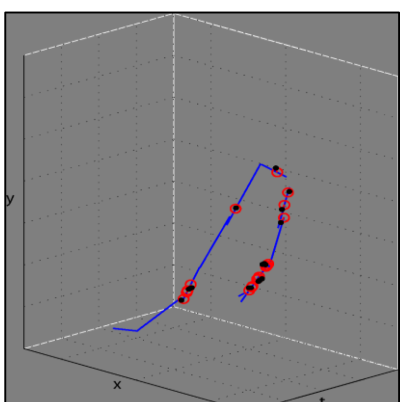
Reduce the $\frac{2}{\epsilon}$ weighted points into $\frac{1}{\epsilon}$ weighted points by constructing their coreset

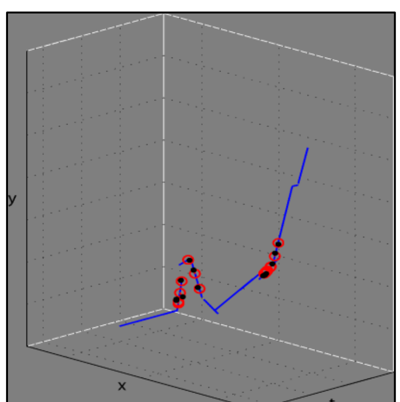
$1+\epsilon$-corset for

$1+\epsilon$-corset for $P_1 \cup P_2$

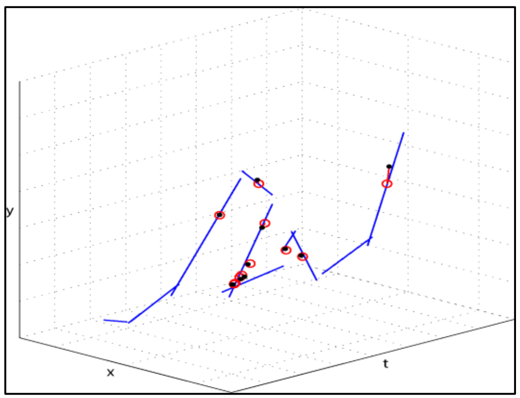


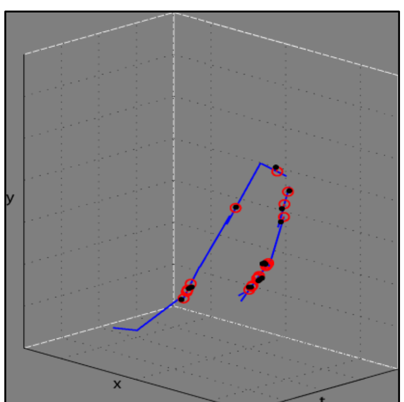
$=(1+\epsilon)^2$-corset for $P_1 \cup P_2$

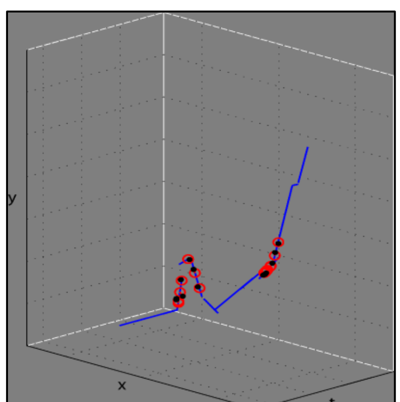




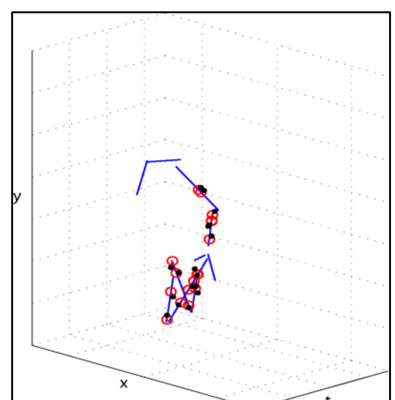
$(1+\epsilon)^2$-corset for $P_1 \cup P_2$



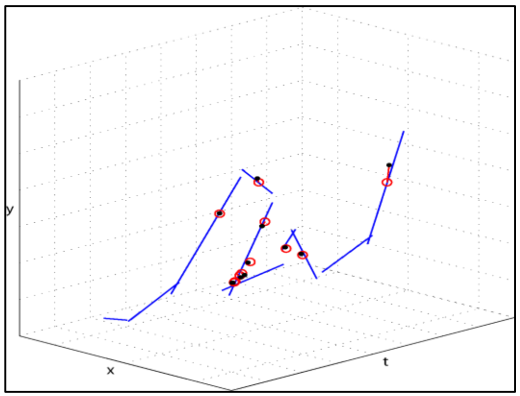


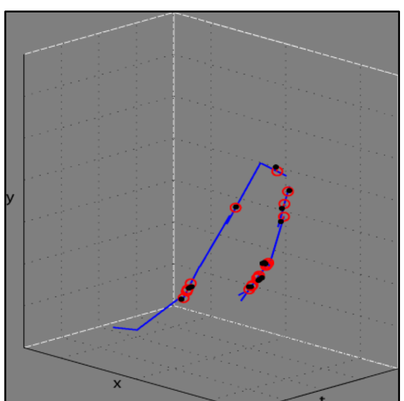


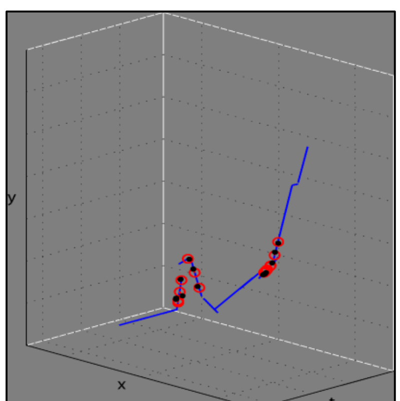
$(1+\epsilon)$-corset for P_3



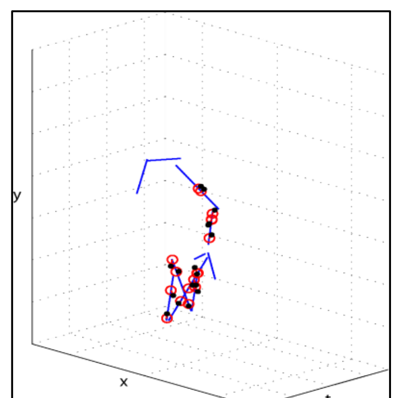
$(1+\epsilon)^2$-corset for $P_1 \cup P_2$



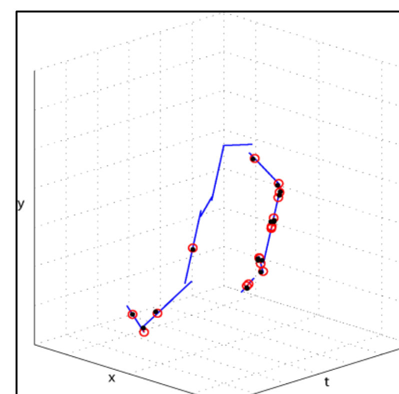




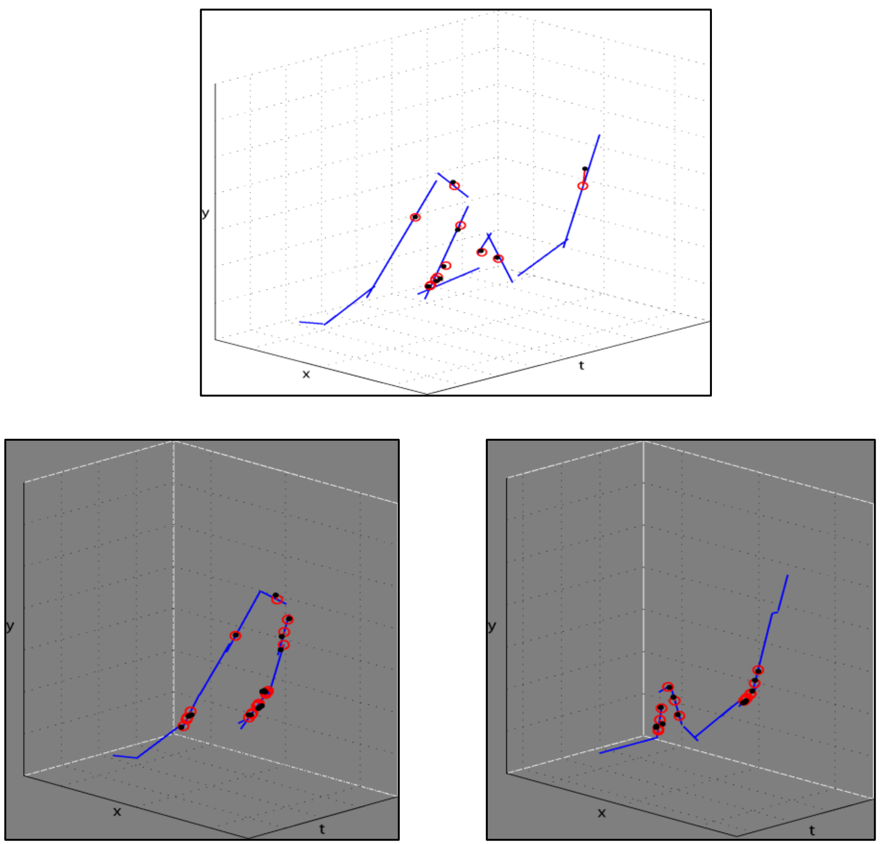
$(1+\epsilon)$-corset for P_3



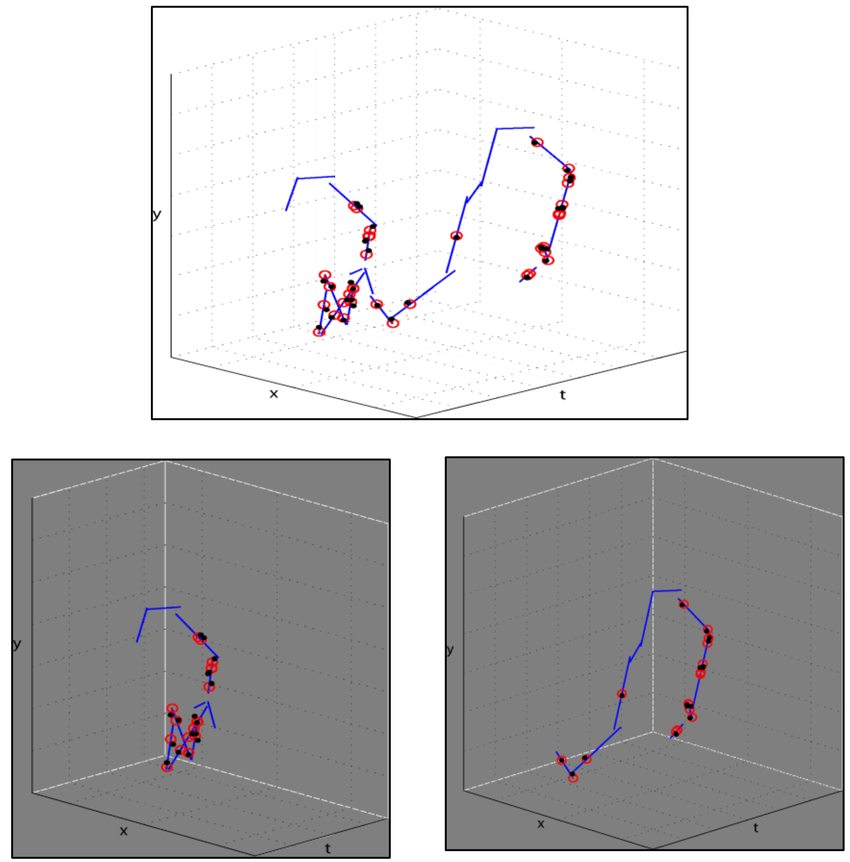
$(1+\epsilon)$-corset for P_4



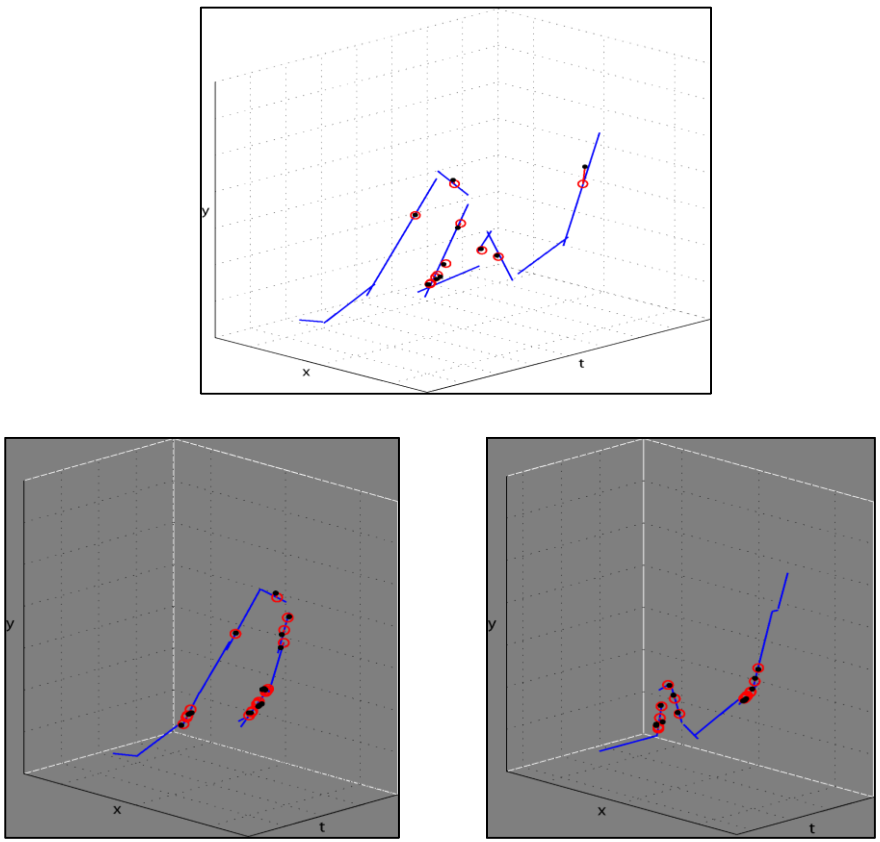
$(1+\epsilon)^2$-corset for $P_1 \cup P_2$



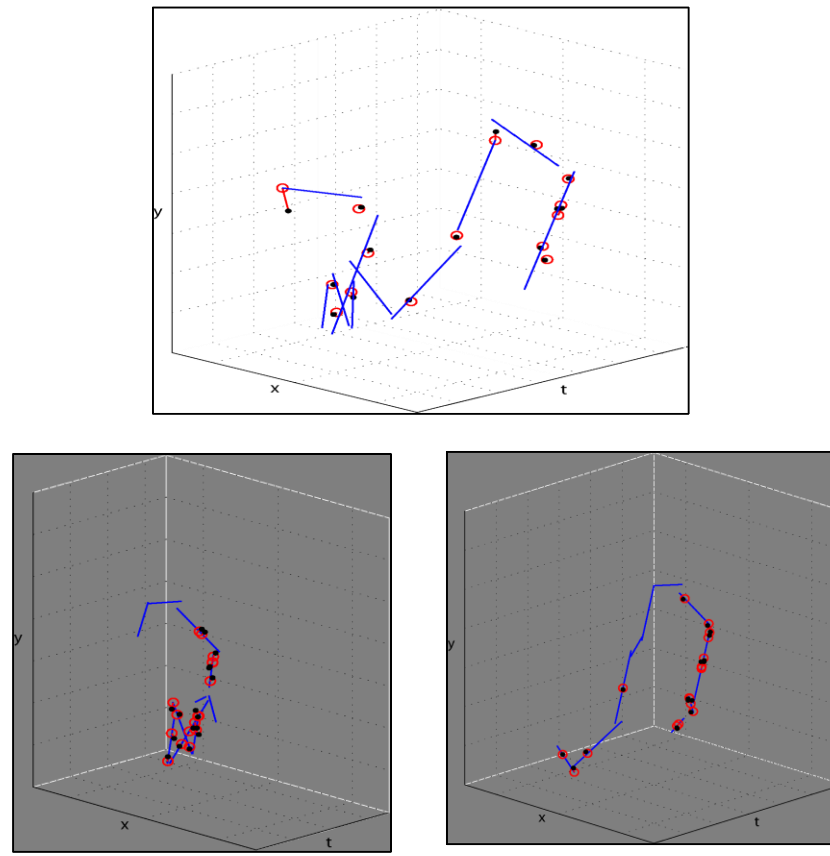
$(1+\epsilon)$-corset for $P_3 \cup P_4$

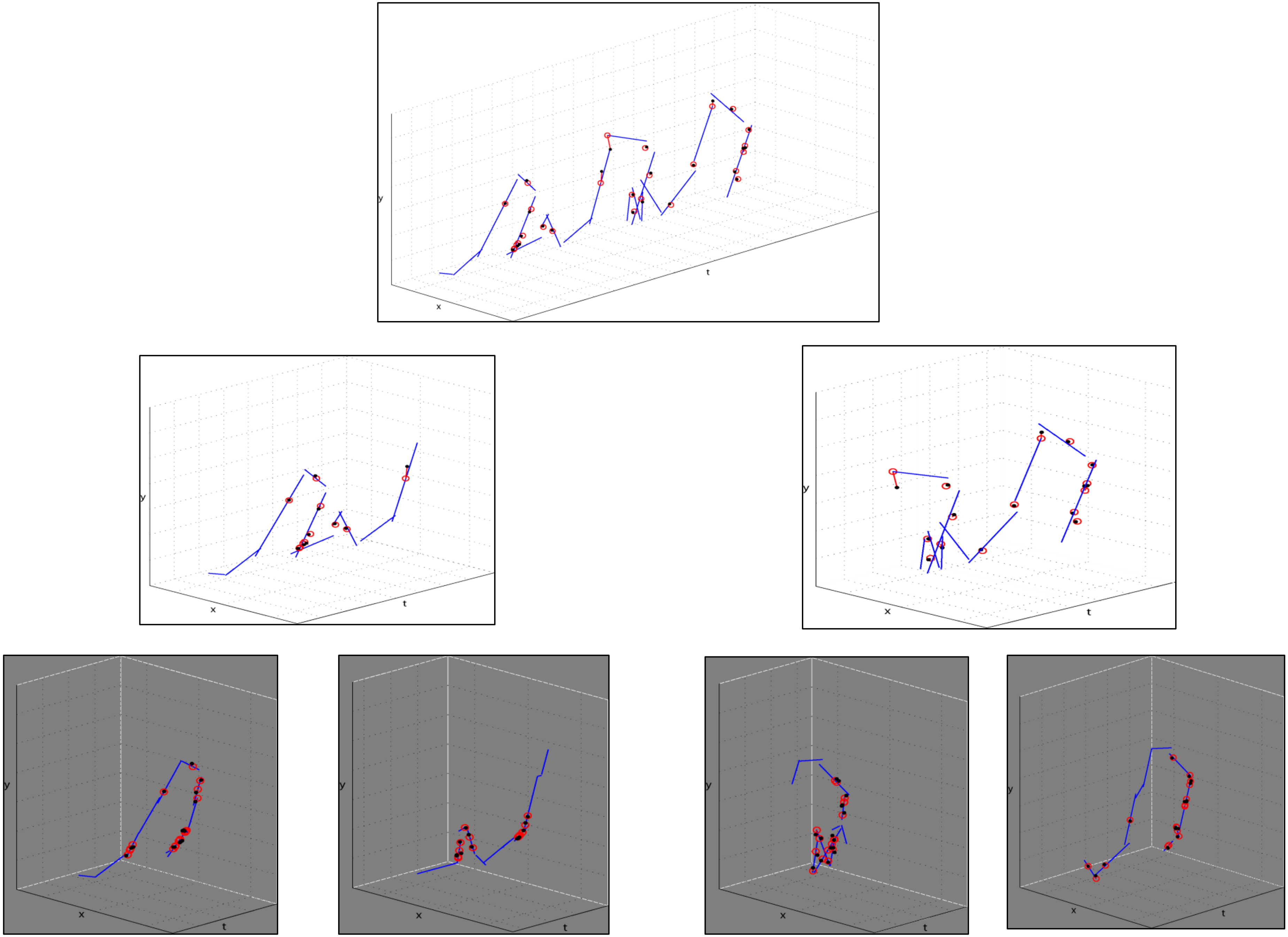


$(1+\epsilon)^2$-corset for $P_1 \cup P_2$



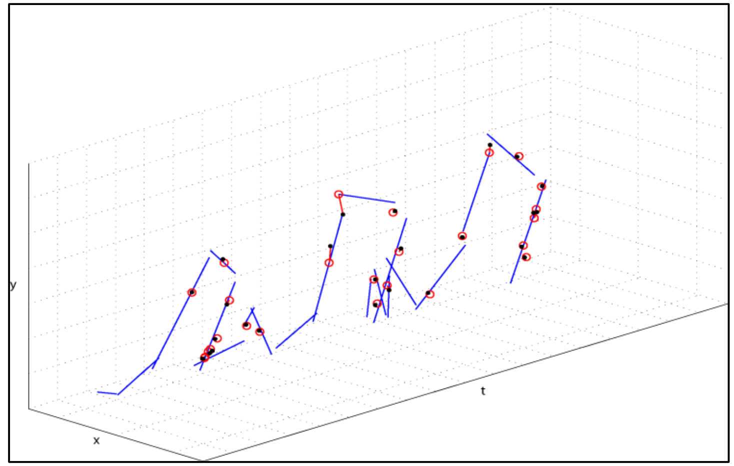
$(1+\epsilon)^2$-corset for $P_3 \cup P_4$

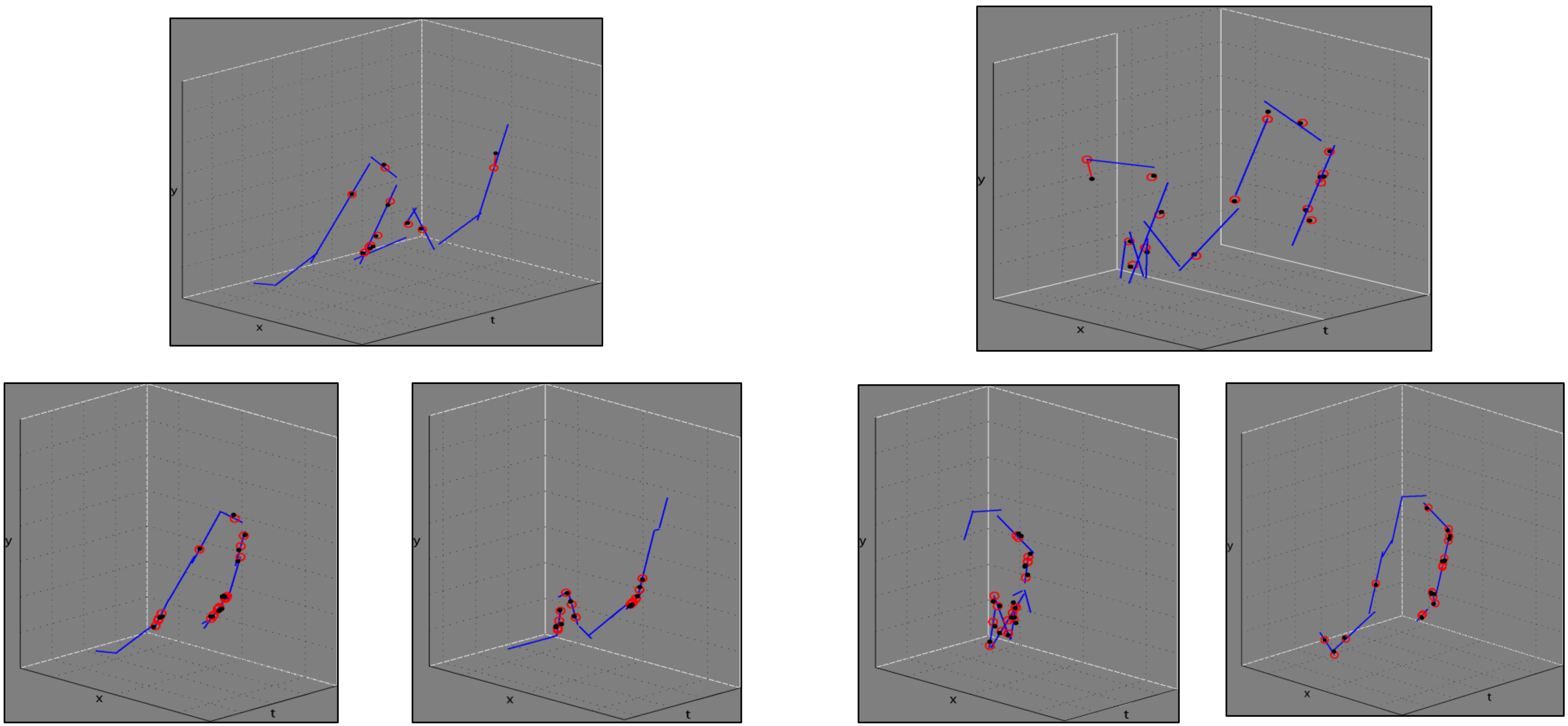



y
x
t

$(1+\epsilon)^2$-coreset for

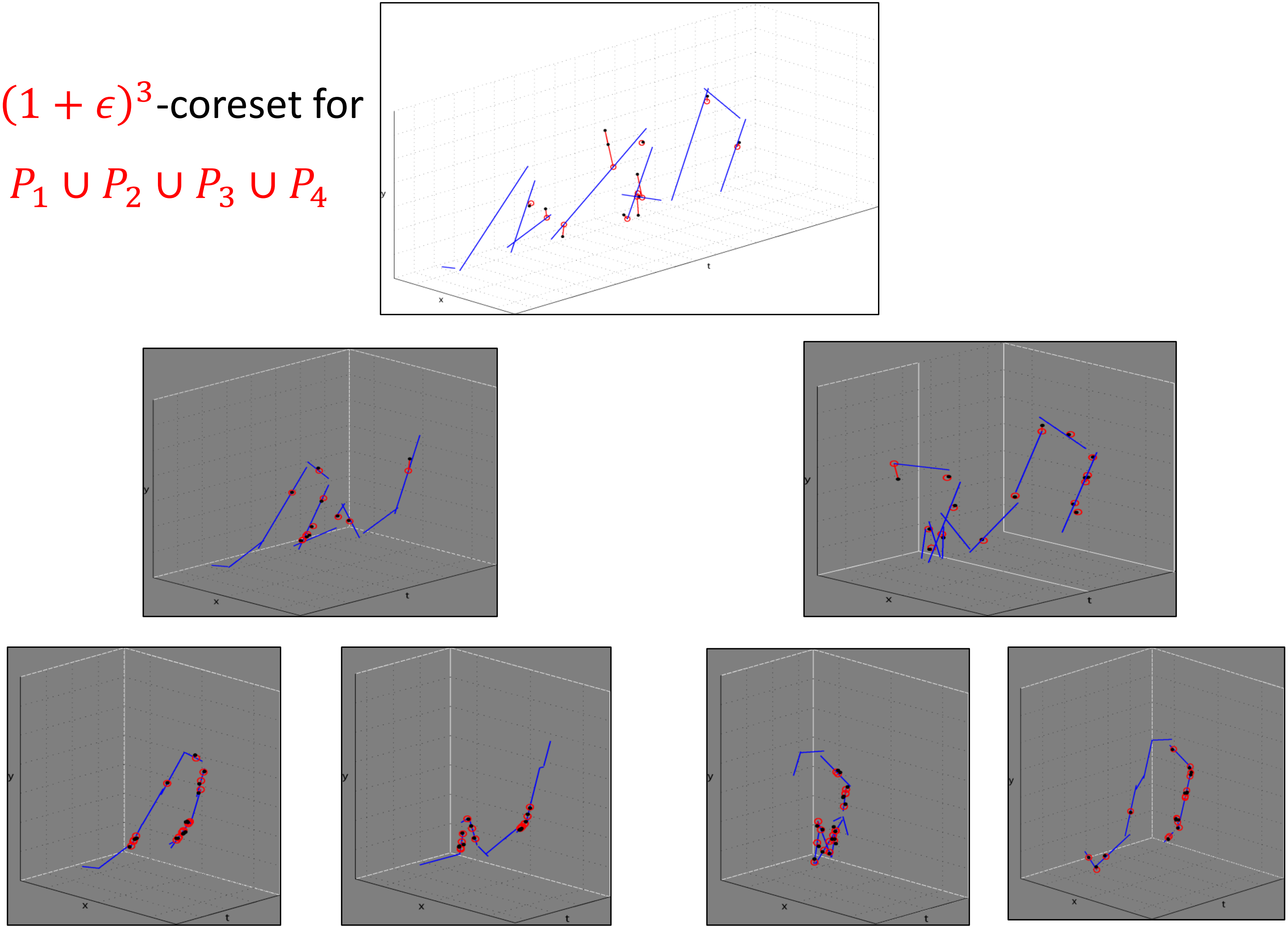
$P_1 \cup P_2 \cup P_3 \cup P_4$

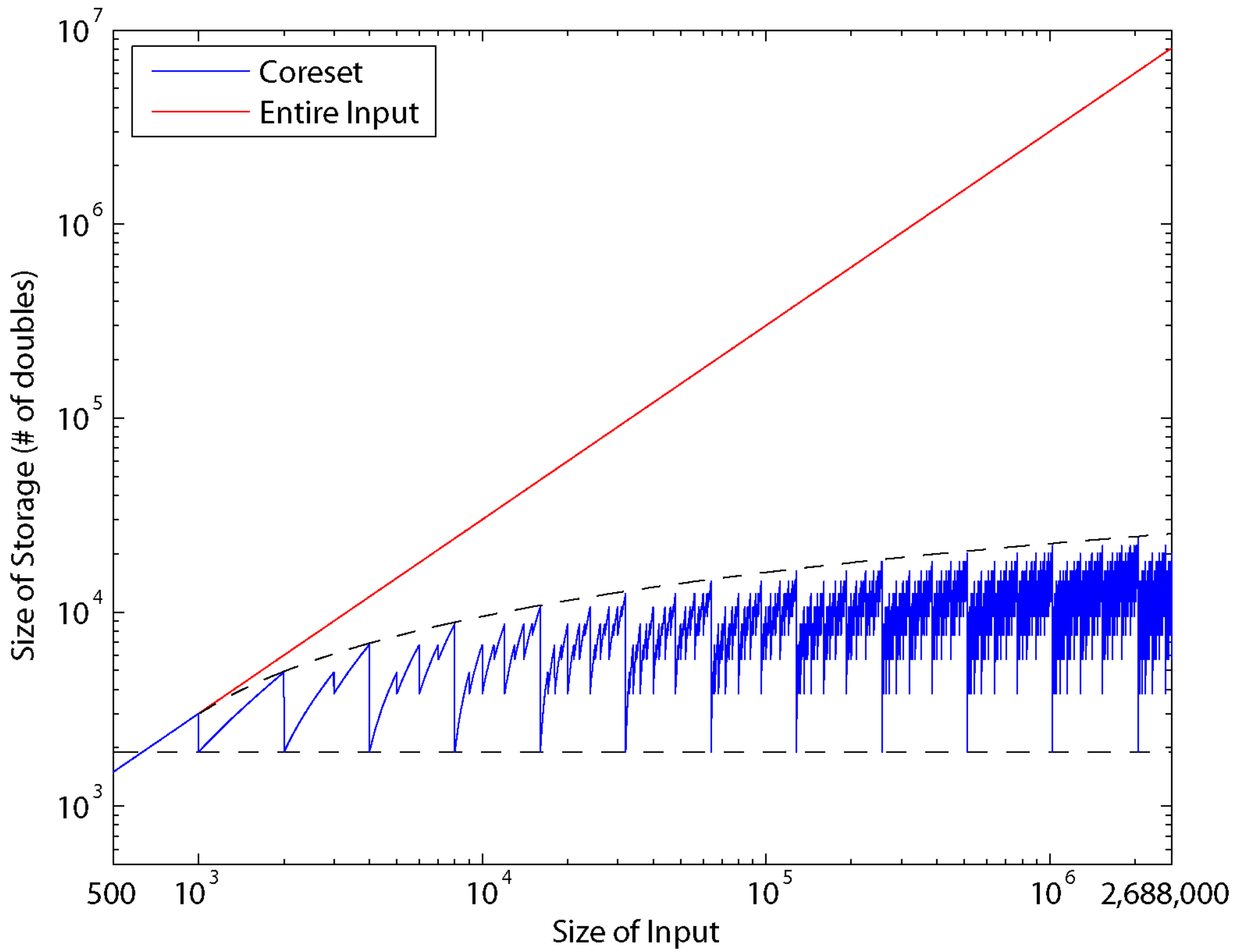




$(1+\epsilon)^3$-coreset for

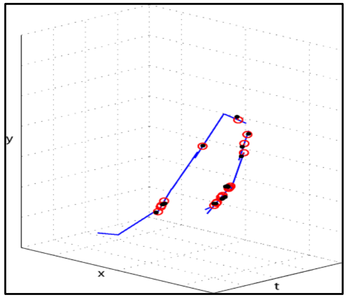
$P_1 \cup P_2 \cup P_3 \cup P_4$

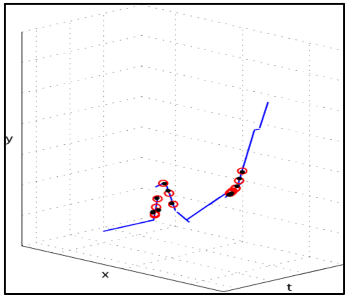


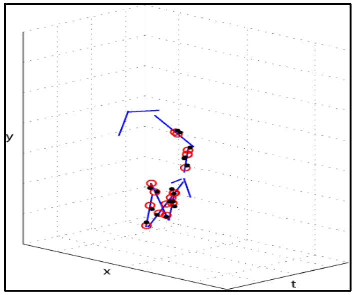


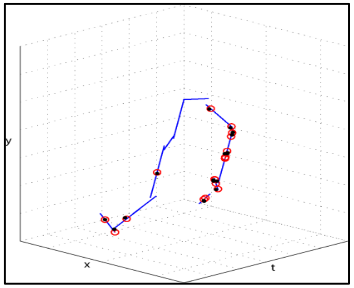
Coreset
Entire Input
Size of Storage (# of doubles)
Size of Input
500
2,688,000

Parallel Computation

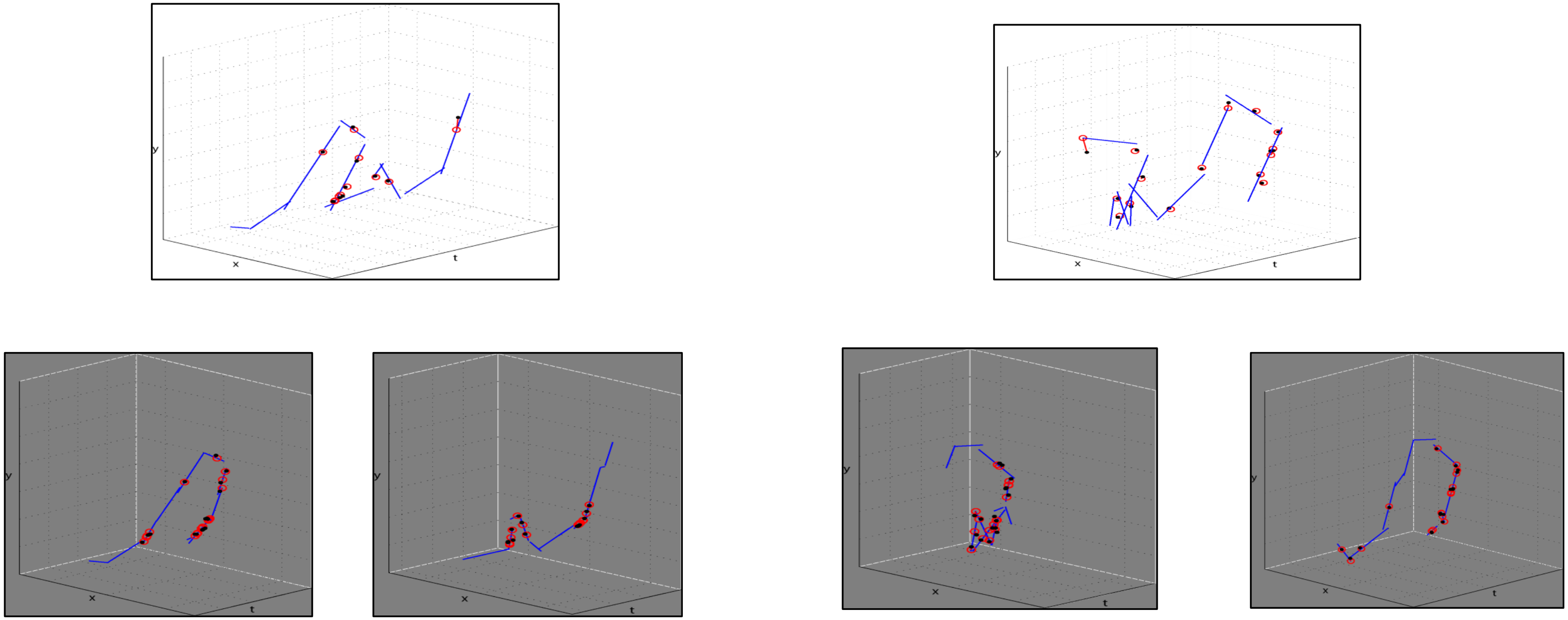






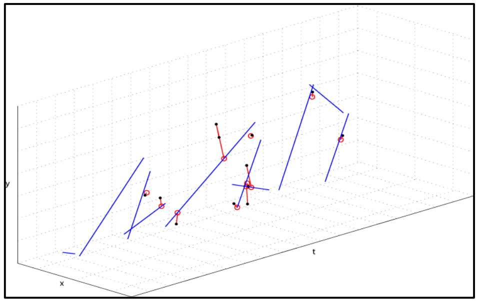


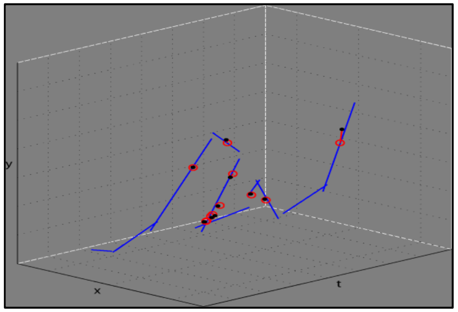
Parallel Computation

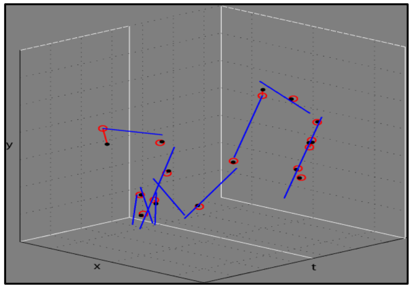


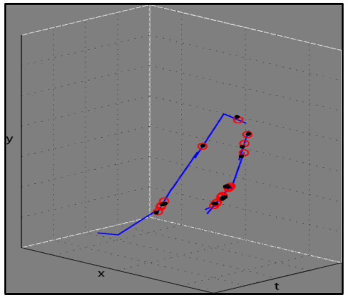
Parallel Computation

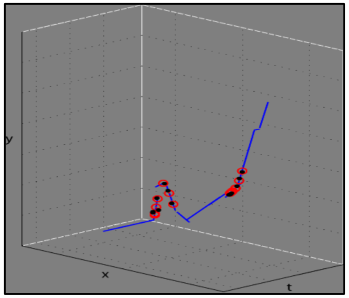
Run off-line algorithm on corset using single computer

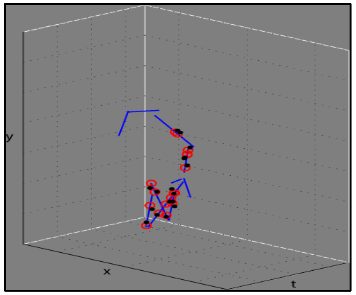


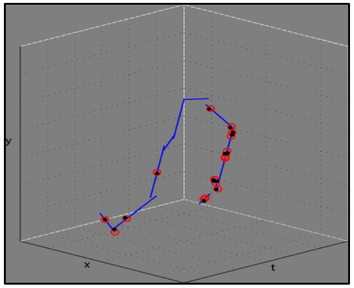




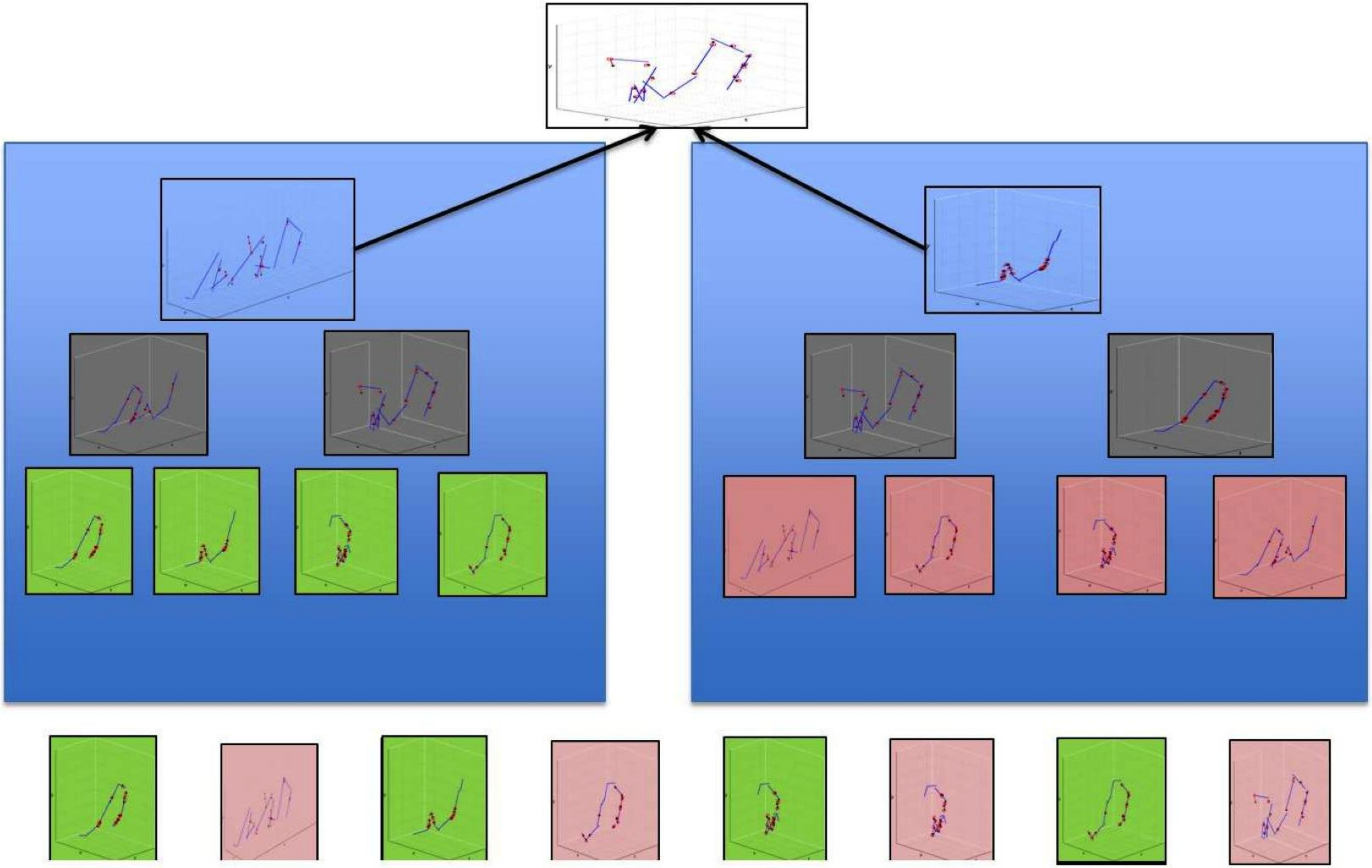






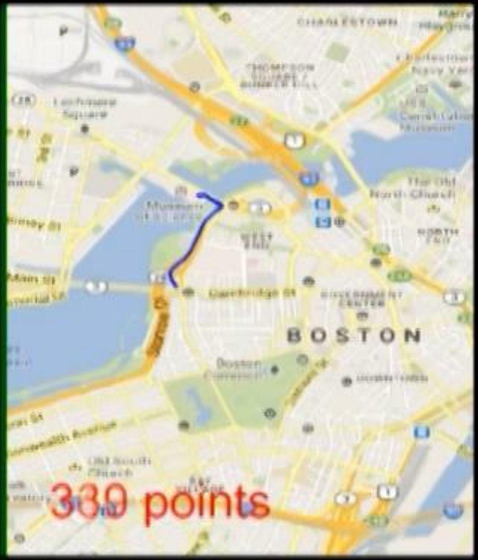


Parallel+ Streaming Computation

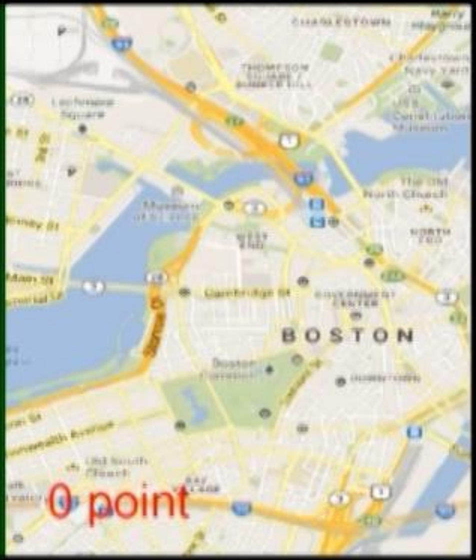


ICRA’14 (With Rus, Paul and Newman)

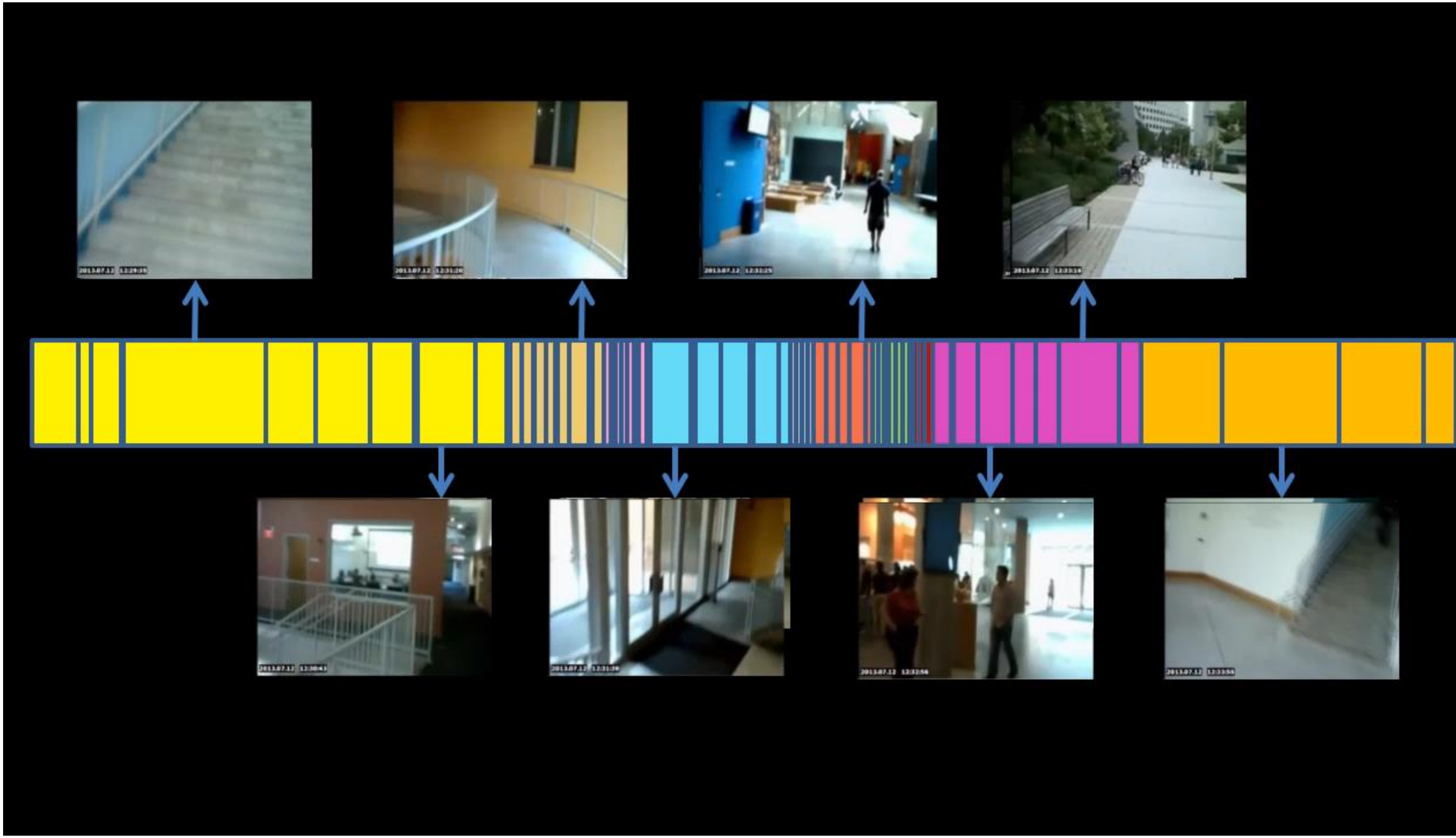
Video (32X)



Raw GPS Points



Coreset Segment



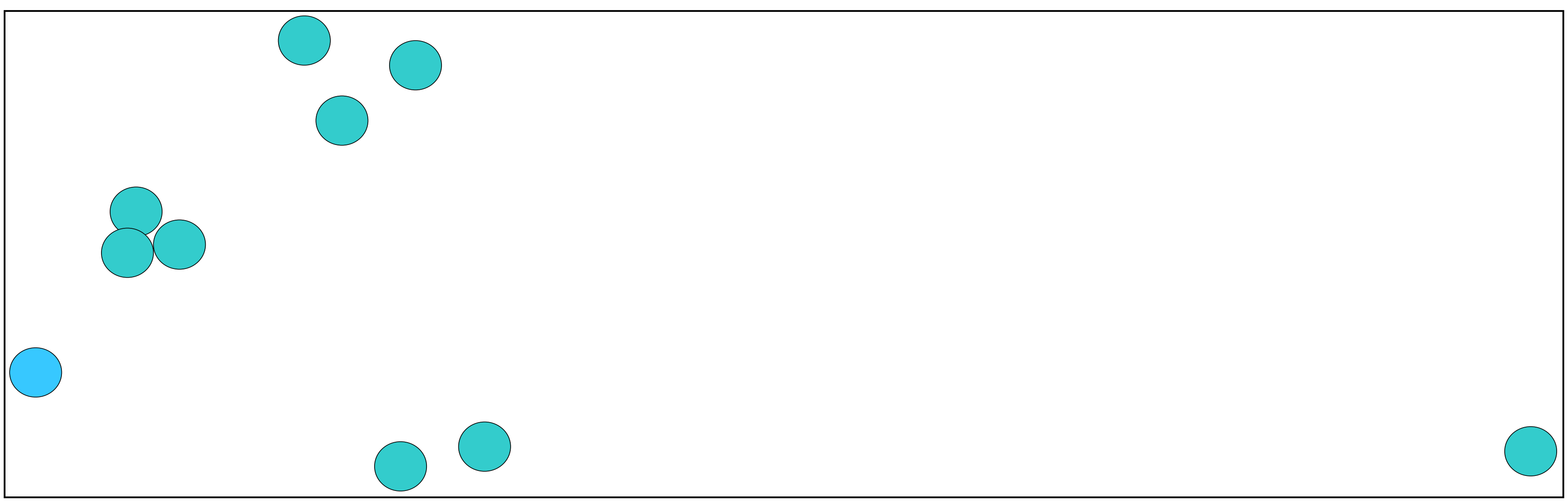
Example Coresets

- Graph/Vector Summarization [F, Rus, Ozer]
- LSA/PCA/SVD [F, Rus, and Volkob, NIPS'16]
- k-Means [F, Barger, SDM'16]
- Non-Negative Matrix Factorization [F, Tassa, KDD15]
- Robots Localization [F, Cindy, Rus, ICRA'15]
- Robots Coverage [F, Gil, Rus, ICRA'13]
- Segmentation [F, Rosman, Rus, Volkob, NIPS'14]
- Dictionary Learning and Image Denoising
  [F, Sochen, J. of Math. Image & Vision, 12]
- Mixture of Gaussians [F Krause, NIPS'11]
- k-Line Means [F, Fiat, Sharir, FOCS'06]
- ...

Coreset for robotics (video)

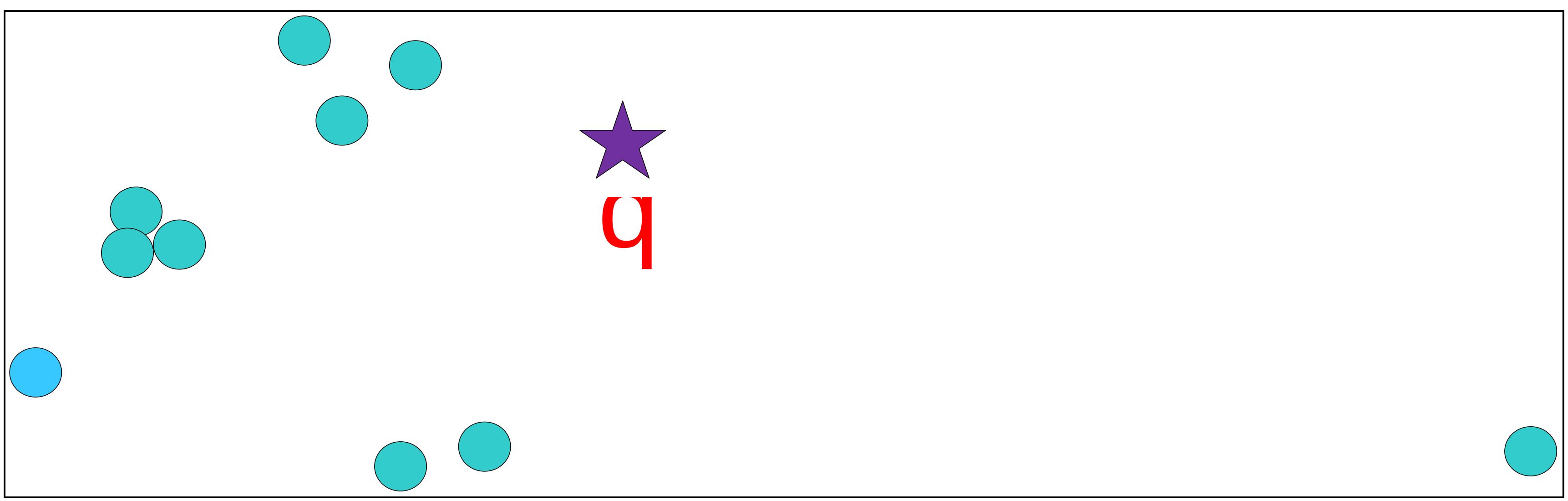
Mean Queries

- Input: P in R^d



Mean Queries

- Input: P in R^d
- Query: a point q 2 R^d

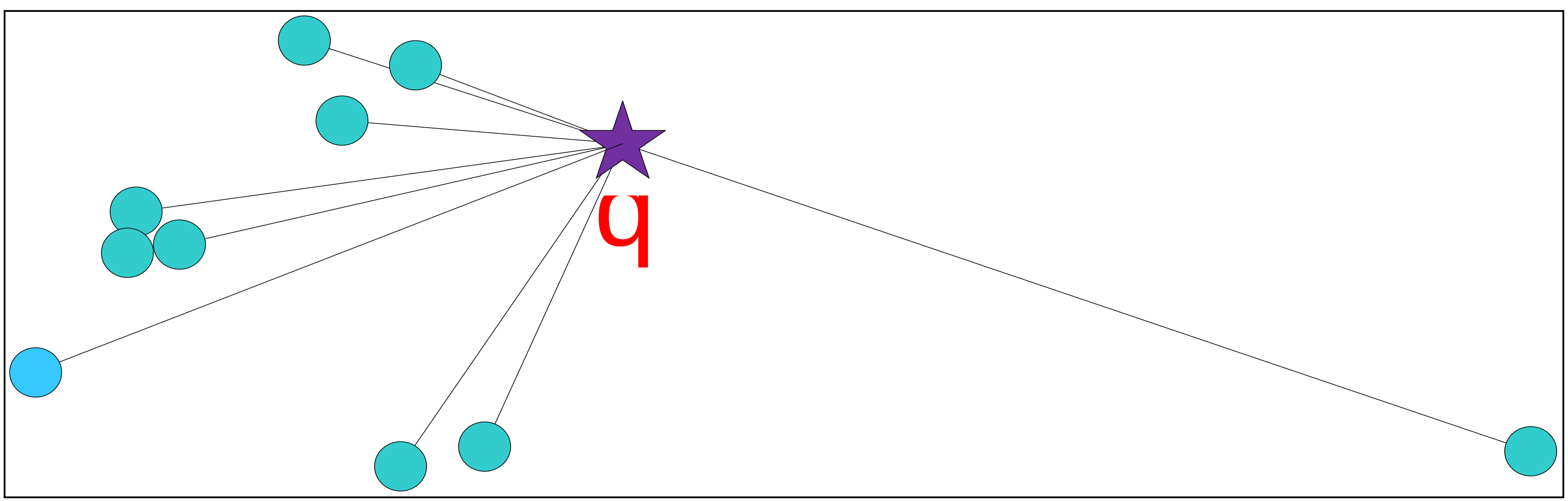


Mean Queries

² Input: P in R^d

² Query: a point q 2 R^d

- Output: $f(P,q) = \sum_{p \in P} (\mathrm{dist}(p,q))^2$



Coreset For Mean Queries

$$\forall \quad \left(\mathrm{dist}(p,q)\right)^2 = \|p - q\|^2$$

$$= \|p\|^2 + \|q\|^2 - 2p \cdot q$$

$$\forall \quad \sum_{p \in P} \left(\mathrm{dist}(p,q)\right)^2 = \sum_{p \in P} \|p\|^2 + n\|q\|^2 - 2q \cdot \sum_{p \in P} p$$

Coreset For Mean Queries

$$\forall q \quad \left(\operatorname{dist}(p,q)\right)^2 = \|p - q\|^2$$

$$= \|p\|^2 + \|q\|^2 - 2p \cdot q$$

$$\forall q \quad \sum_{p \in P} \left(\operatorname{dist}(p,q)\right)^2 = \sum_{p \in P} \|p\|^2 + n\|q\|^2 - 2q \cdot \sum_{p \in P} p$$

Problem: compute a small weighted **subset** deterministically.

[ICML'17, with Rus and Ozer]

Relation to Google's PageRank

- Input: Binary adjacency matrix G of a graph.
- Scale every column to have sum of 1
  - (G is now a stochastic matrix)
- Let $d = 0.85$ to get a positive stochastic matrix:

$$A = d * G + (1 - d) \cdot \mathbf{1}$$

- There is a distribution x such that $Ax = x$

  (Perron–Frobenius theorem)
- $Bx = 0$ for $B = A - I$
- Output: x (PageRank vector)

Relation to Google's PageRank

- Input: Binary adjacency matrix G of a graph.
- Scale every column to have sum of 1
  - (G is now a stochastic matrix)
- Let $d = 0.85$ to get a positive stochastic matrix:

$$A = d * G + (1 - d) \cdot \mathbf{1}$$

- There is a distribution x such that $Ax = x$

  (Perron–Frobenius theorem)
- $Bx = 0$ for $B = A - I$
- Output: x (PageRank vector)

- Core-set: a sparse x' such that $||Bx'|| < \epsilon$

Common Localization of quadcopter

- Many sensors:
  GPS, Kinect, GoPro, LiDAR, IMU, Sonar
- Good:
  Easy to hover and navigate
- Bad:
  - Dangerous, expensive, heavy
  - Hard to compare & analyze

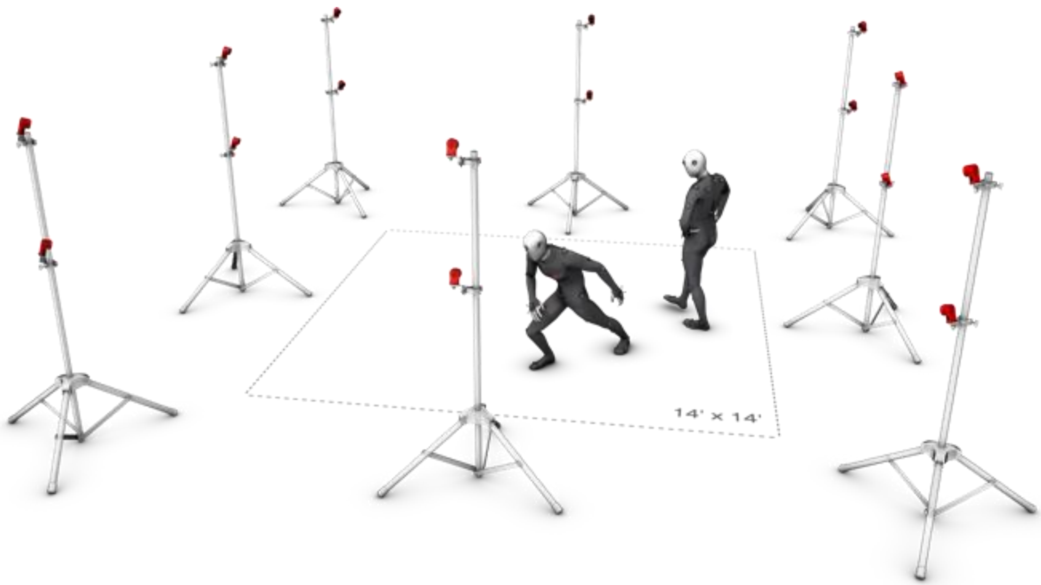
Our Robotics & Big Data lab

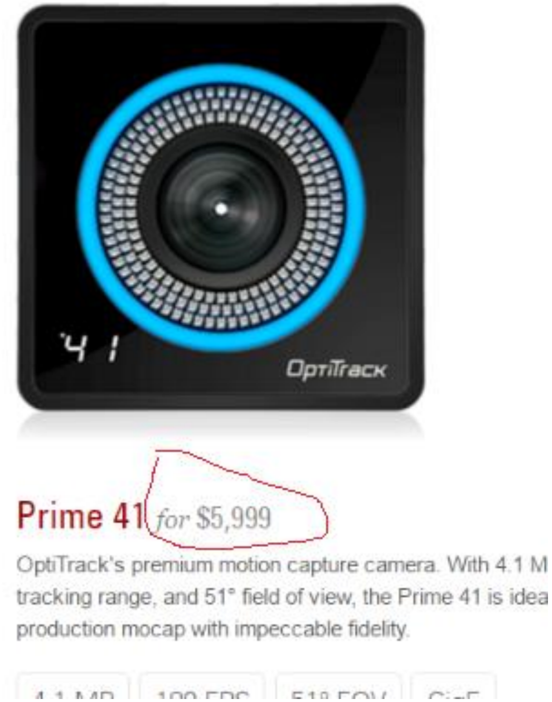
- Toy-drones, no sensors or tiny analog camera

- Good:
  - Safe for indoor navigation, and low-cost
  - Easy to model

- Bad:
  - Unstable
  - Need ~ 30 location updates per second

Expensive Tracking System

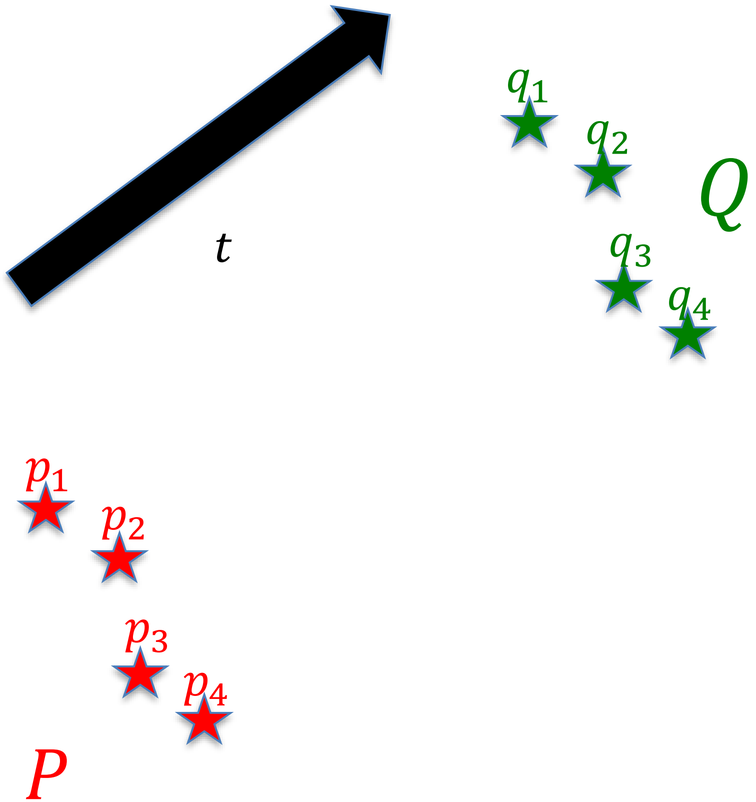




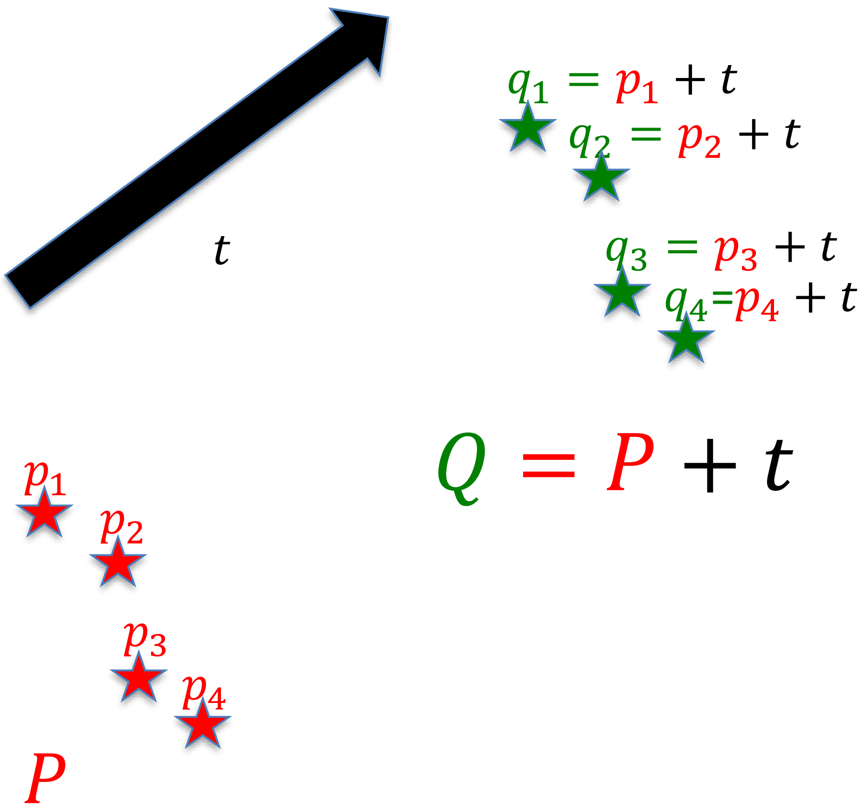
Challenge: use weak hardware

Using stronger algorithms

Exact Translation Recovery



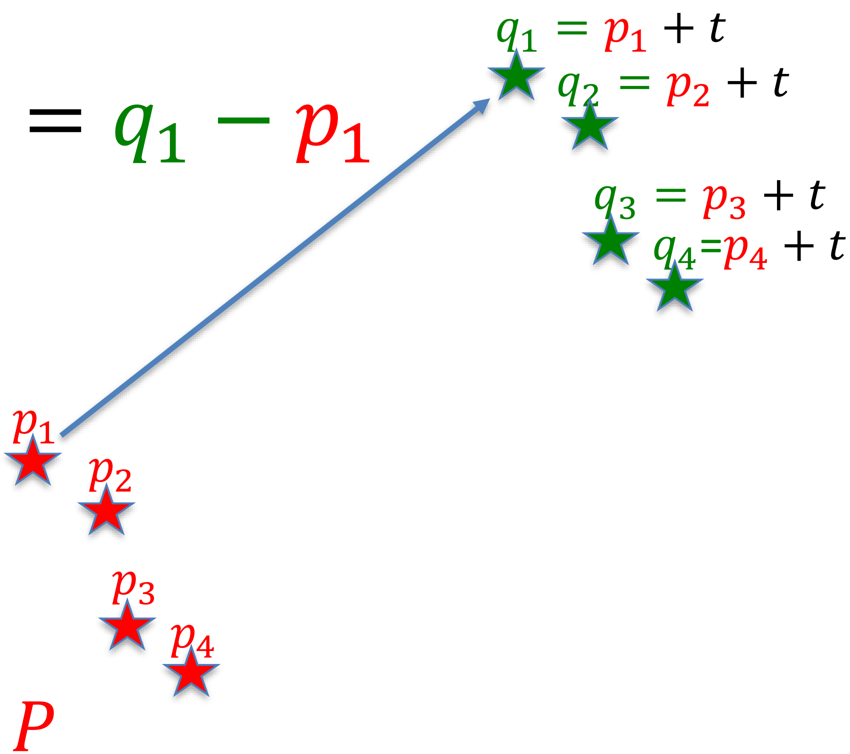
Exact Translation Problem



$$Q = P + t$$

Exact Translation Recovery

$t = q_1 - p_1$

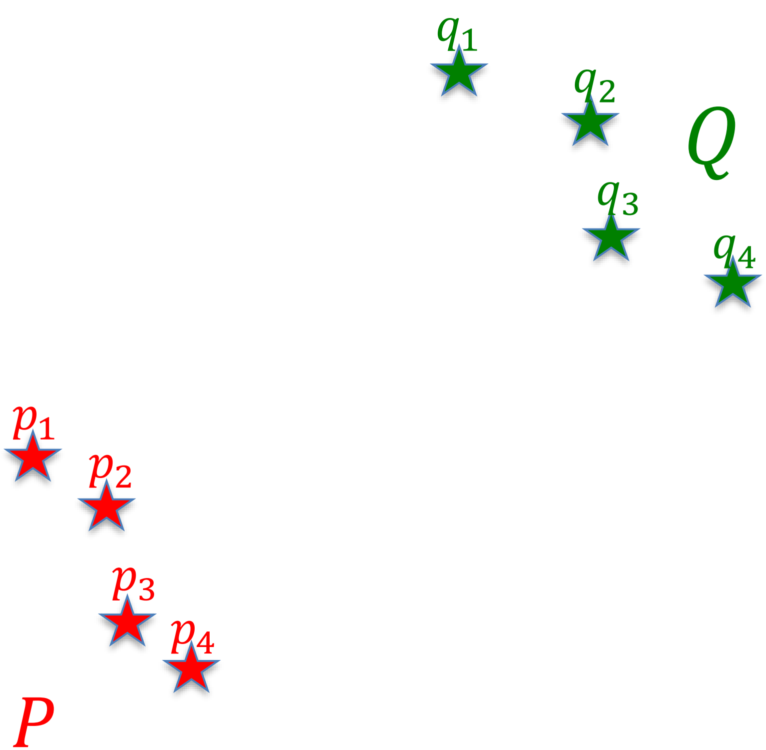


Solution:

$t = q_1 - p_1$

$Q = P + t$

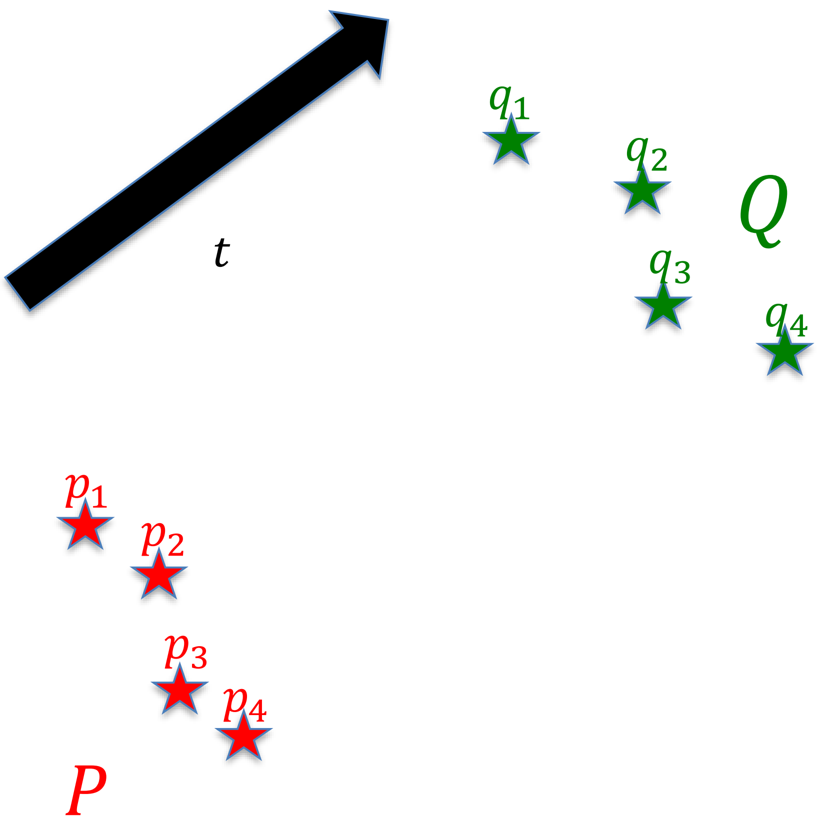
Noisy Observations



Added Gaussian noise due to:

- Low resolution
- Few Frames Per Second (FPS)
- Latency (delay)
- Communication errors
- Camera Tilting

Translation Estimation

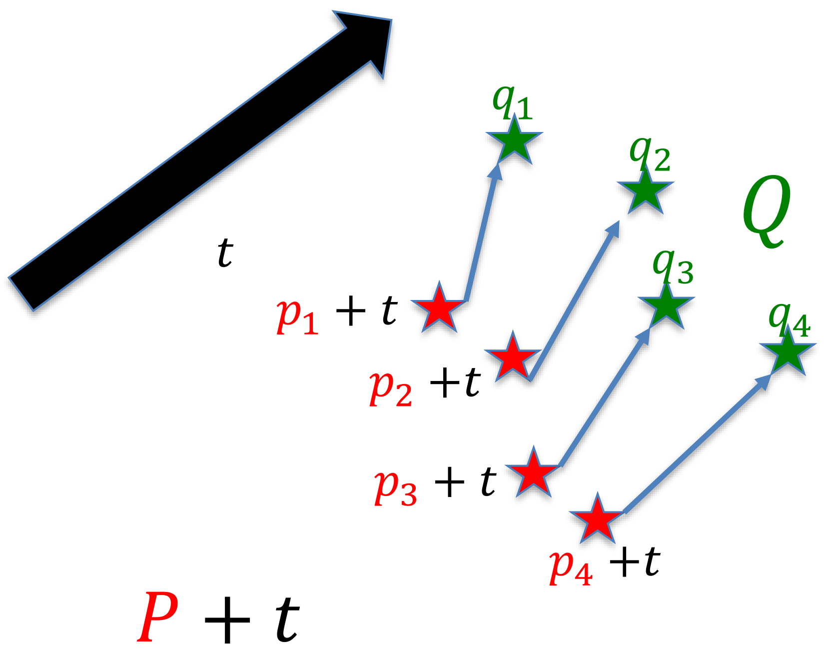


Added Gaussian noise due to:

- Low resolution
- Few Frames Per Second (FPS)
- Latency (delay)
- Communication errors
- Camera Tilting

$$P + t \sim Q$$

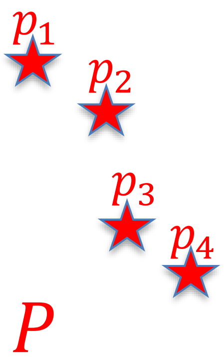
Translation Estimation



Compute a translation t of P that minimizes the sum of squared distances to Q

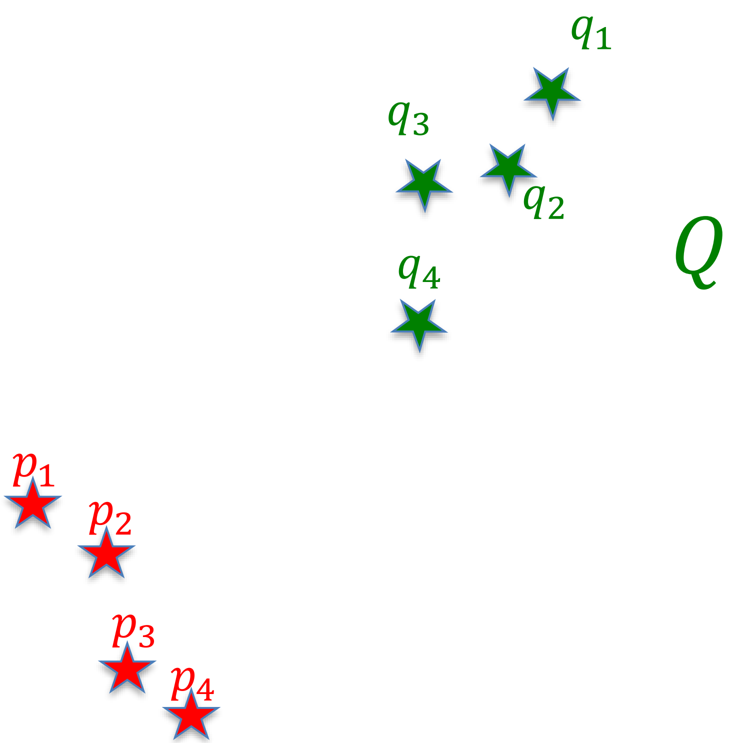
$$\min_t \sum_{i=1}^{n} \mathrm{dist}^2(p_i + t, q_i)$$

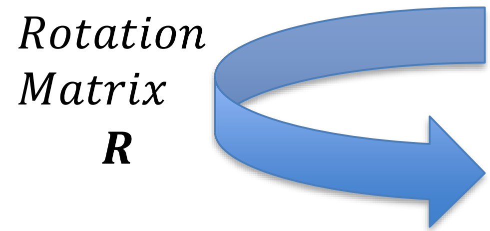
Q = Translation & Rotation of P



The object not only moves, but also rotates in space

The Pose-Estimation Problem



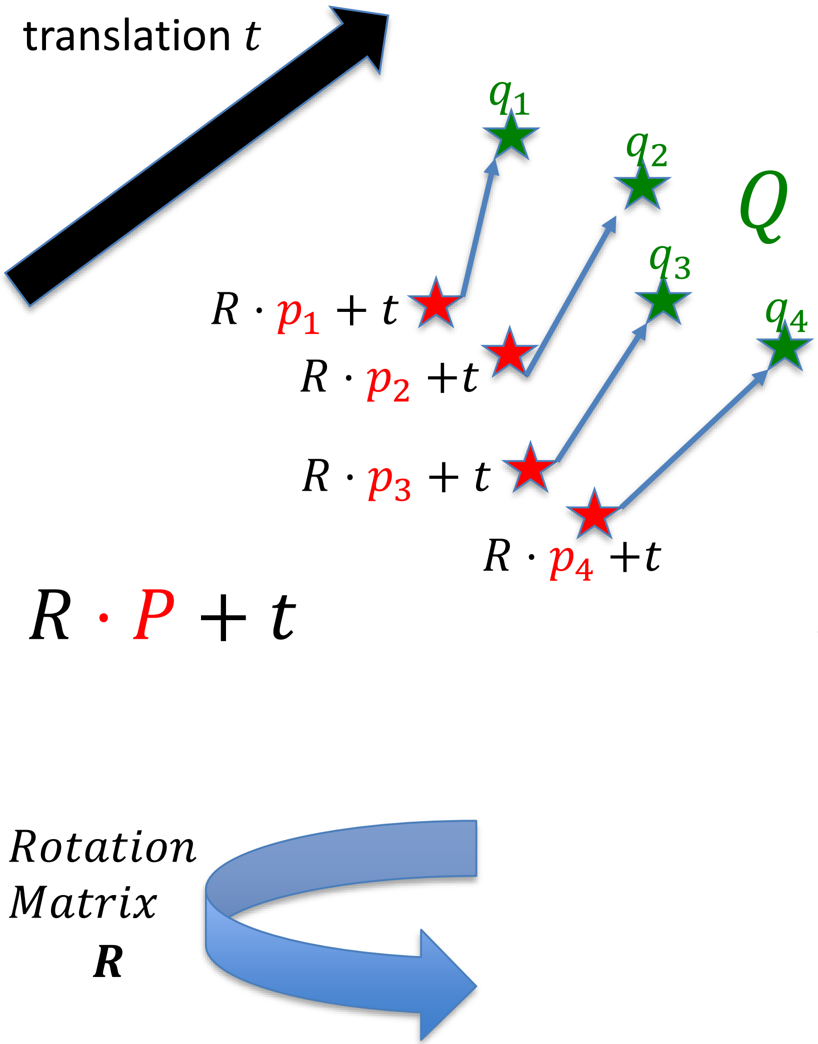


$$P \rightarrow R \cdot P \rightarrow R \cdot P + t$$

A rotation corresponds to a rotation matrix R in $\mathbb{R}^{d \times d}$:

$$q_i = R p_i + t$$

The Pose-Estimation Problem

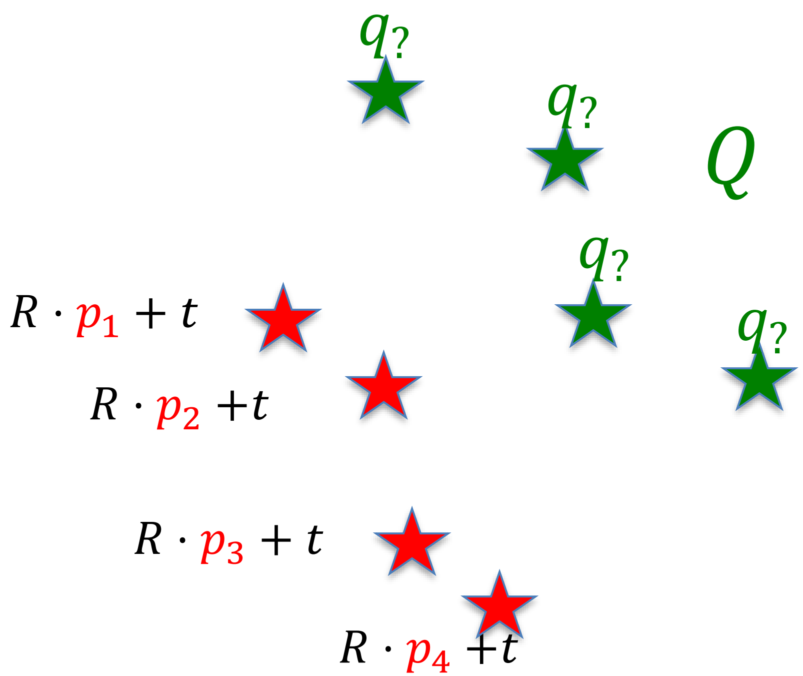


Compute Rotation & Translation of P that minimizes its sum of squared distances to Q :

$$\min_{t,R} \sum_{i=1}^{n} \mathrm{dist}^2(R \cdot p_i + t, q_i)$$

Matching & Pose-Estimation

- Matching of each p_i to its q_i is also unknown.

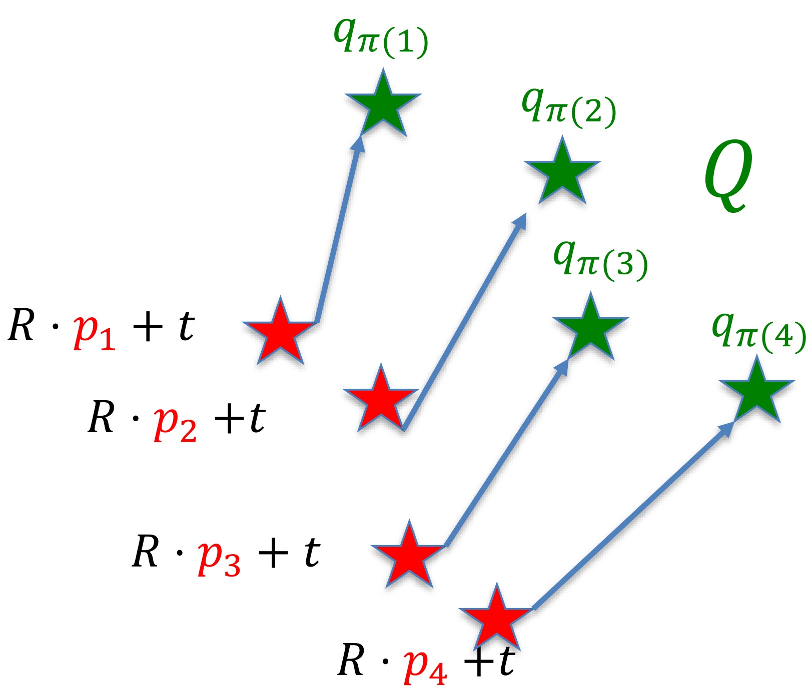


- Needs to compute a permutation
  $\pi: \{1, \cdots, n\} \to \{1, \cdots, n\}$
  where p_i is assigned to $q_{\pi(i)}$

Matching & Pose-Estimation

Compute **Permutation,** Rotation & Translation of P that minimizes its sum of squared distances to Q :

$$\min_{\pi, t, R} \sum_{i=1}^{n} \text{dist}^2\left(R \cdot p_i + t, q_{\pi(i)}\right)$$



Existing Solutions

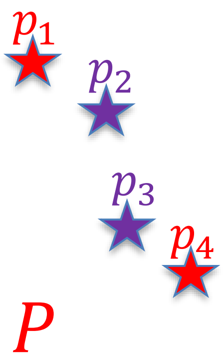
- Optimal Translation is simply the mean
- Let UDV^T be a Singular Value Decomposition (SVD) of the matrix P^TQ. That is:

$$UDV^T = \mathrm{P}^{\mathrm{T}}\mathrm{Q}$$

- **Theorem 1 ($\boldsymbol{Kabsch\ algorithm}$).**
- The matrix $R^* = VU^T$ is the optimal rotation and can be computed in $O(nd^2)$ time.

Core-set For Pose Estimation

Observed ordered set Q (now) of n markers

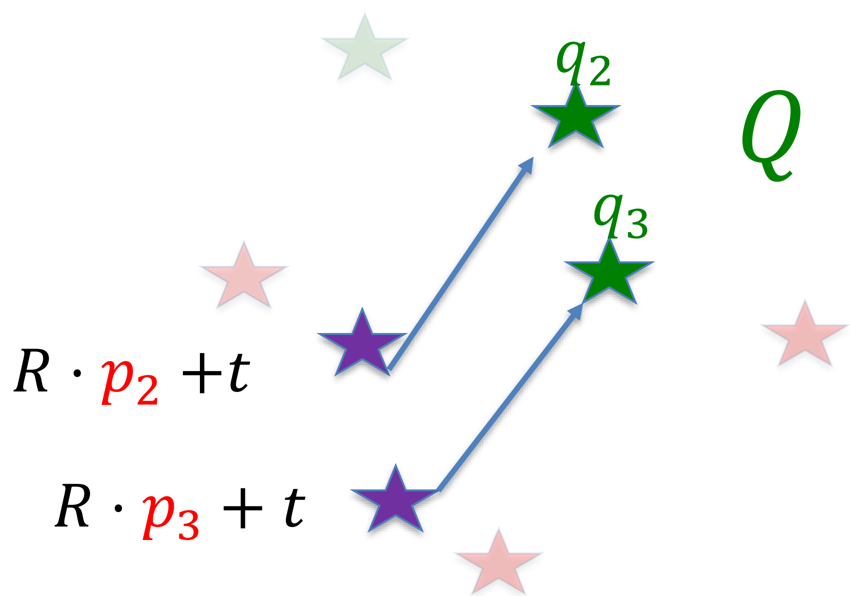


Ordered set $|P|$ of n markers.
Initial position of object.

Core-set For Pose Estimation

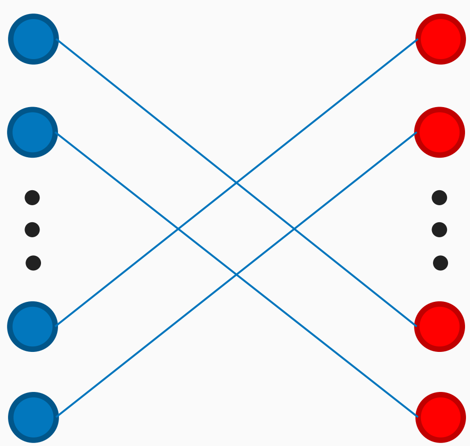
A weight vector $w_1, \cdots, w_n \geq 0$ whose most entries are zeroes and for every R and t:

$$\sum_{i=1}^{n} \text{dist}^2(R \cdot p_i + t, q_i) = \sum_{i=1}^{n} \mathrm{w_i} \text{dist}^2(R \cdot p_i + t, q_i)$$

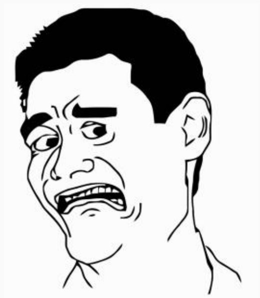


The Pose-Estimation Problem
“Full version”

Matching. Assuming P is an initial set of n markers (points in R^d), and Q is the observed set of markers, we need to match each point in P to it’s corresponding point in Q.



$O(n!)\ Permutations$

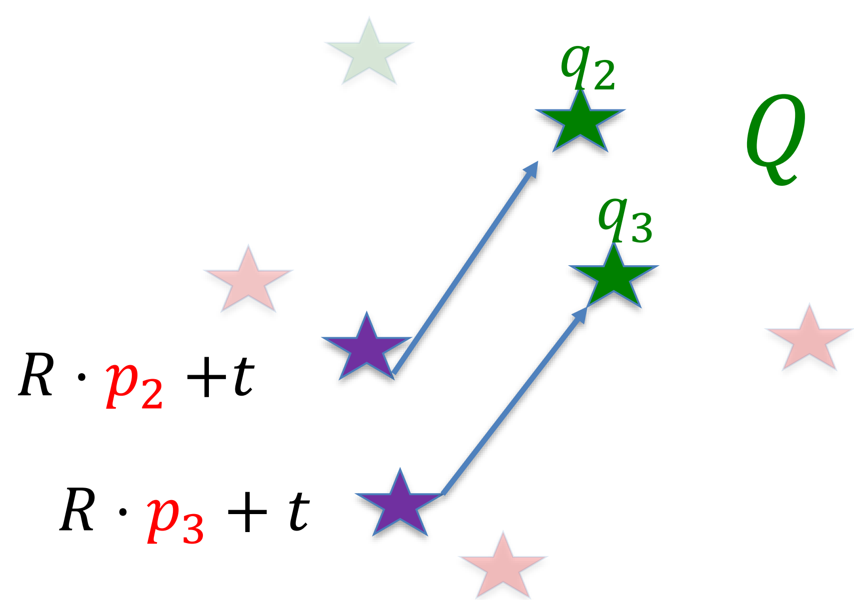


Main Theorem [S. Nasser, I. Jubran, F]

Every set of n points has a core-set of size $O(d^2)$ that can be computed in $O(nd)$ time.

$$\sum_{i=1}^{n} \mathrm{dist}^2(R \cdot p_i + t, q_i) =$$

$$\sum_{i=1}^{n} \mathrm{w_i}\mathrm{dist}^2(R \cdot p_i + t, q_i)$$



Off-line solution

- Optimal rotation:

$$\hat{R} = VU^T \; where \; SVD\left(\sum_{i=1}^{n} p_i^T q_i\right) = \; UDV^T$$

$$A_{3x3} = \sum_{i=1}^{n} p_i^T q_i = \sum_{i=1}^{n} A_{i_{3x3}} \cong \sum_{i=1}^{n} a_{i_{1x9}}$$

Solving the Problem cont.

- $A_{3x3} \cong A_{1x9} = \sum_{i=1}^{n} a_{i_{1x9}} = \sum_{i=1}^{k} \omega_i a_{i_{1x9}}$

$a_{i_{3x3}} \cong a_{i_{1x9}}$

Coreset

Matrix Approximation by rows subset

For every matrix A there is a diagonal matrix W of only d^2 non-zeros entries such that for every $\boldsymbol{x} \in R^{\boldsymbol{d}}$

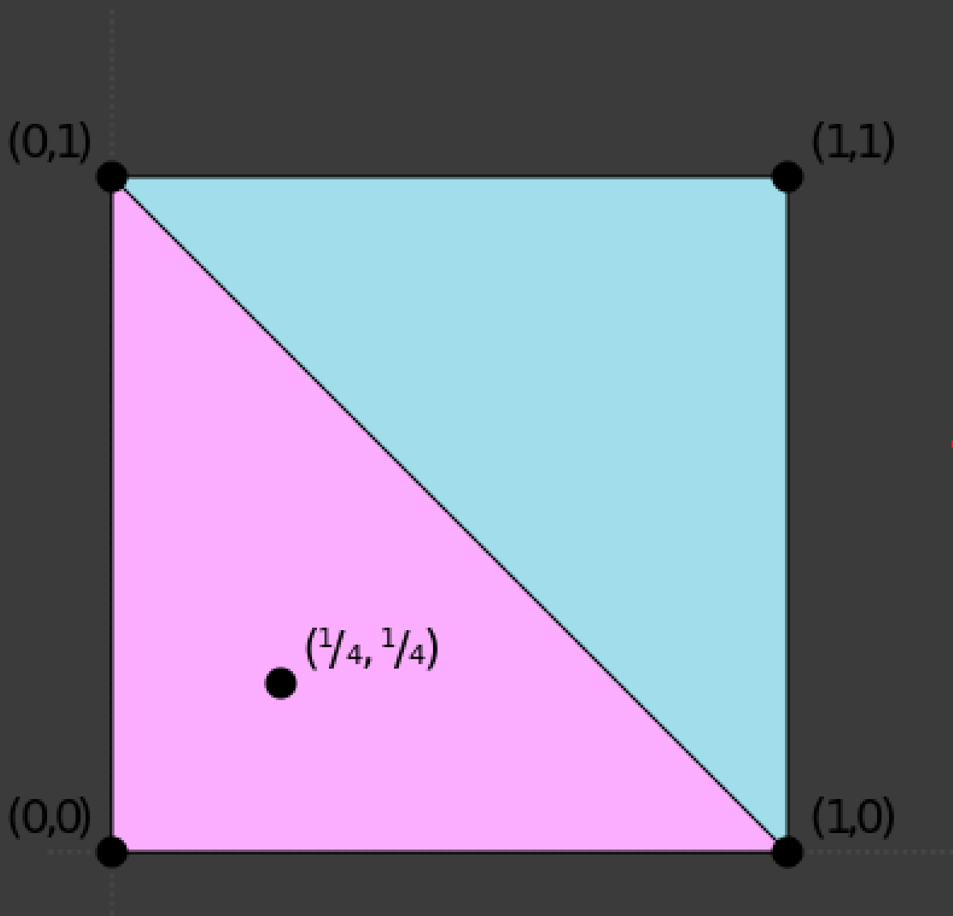
$$\left|\left|\boldsymbol{Ax}\right|\right| = ||\boldsymbol{WAx}||$$

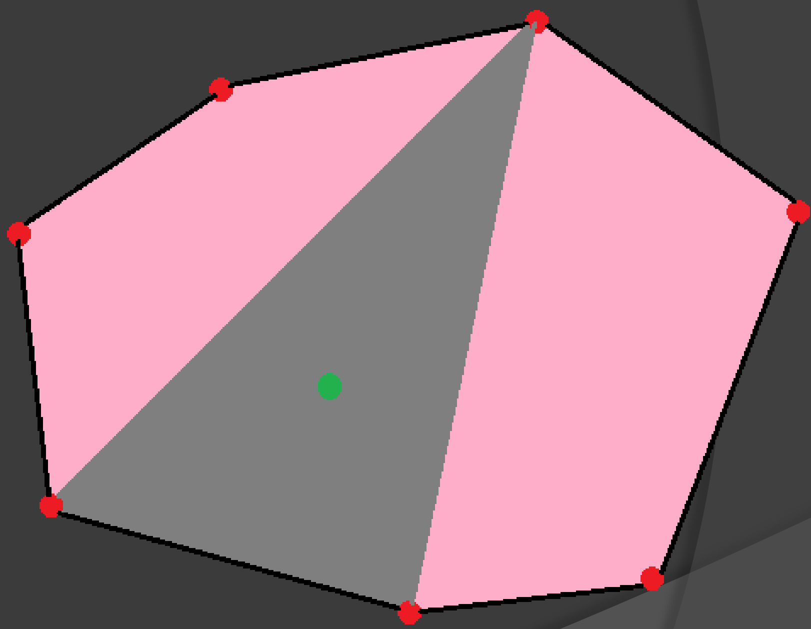
Proof: $\left|\left|Ax\right|\right|^2 = x^T(A^TA)x = x^T\left(\sum_i a_i a_i^T\right)x$

$$= x^T\left(\sum_i w_i a_i a_i^T\right)x$$

$$=x^T(A^TW^TWA)x = \left|\left|WAx\right|\right|^2$$

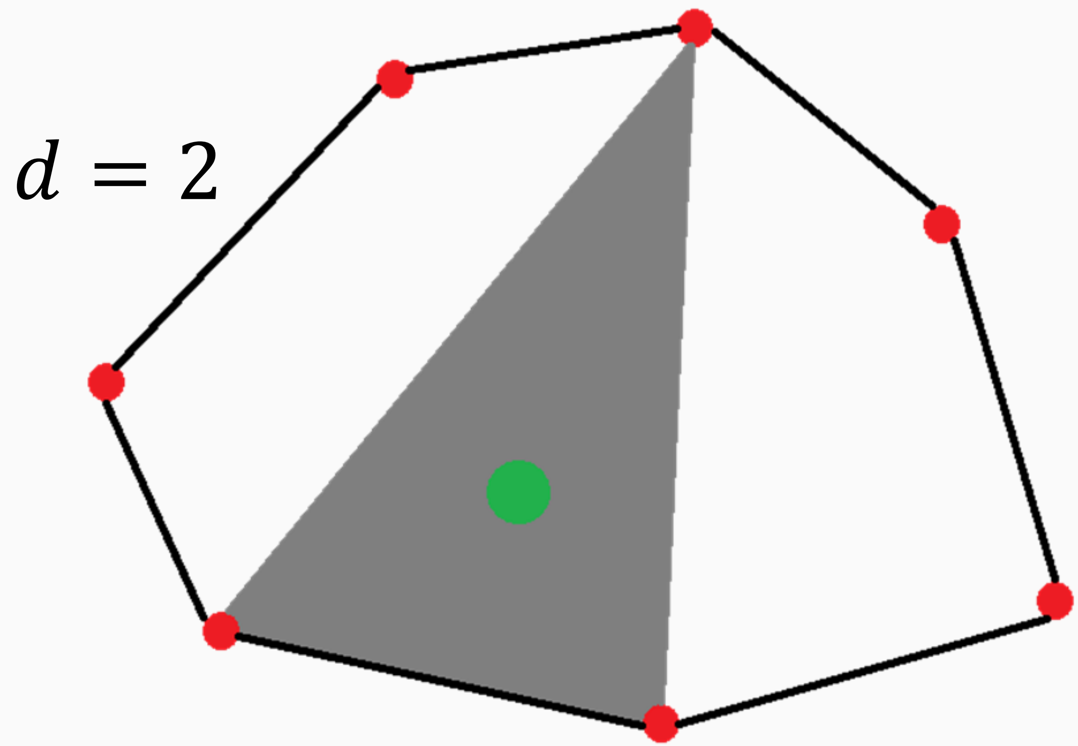
Intuition ($d = 2$)



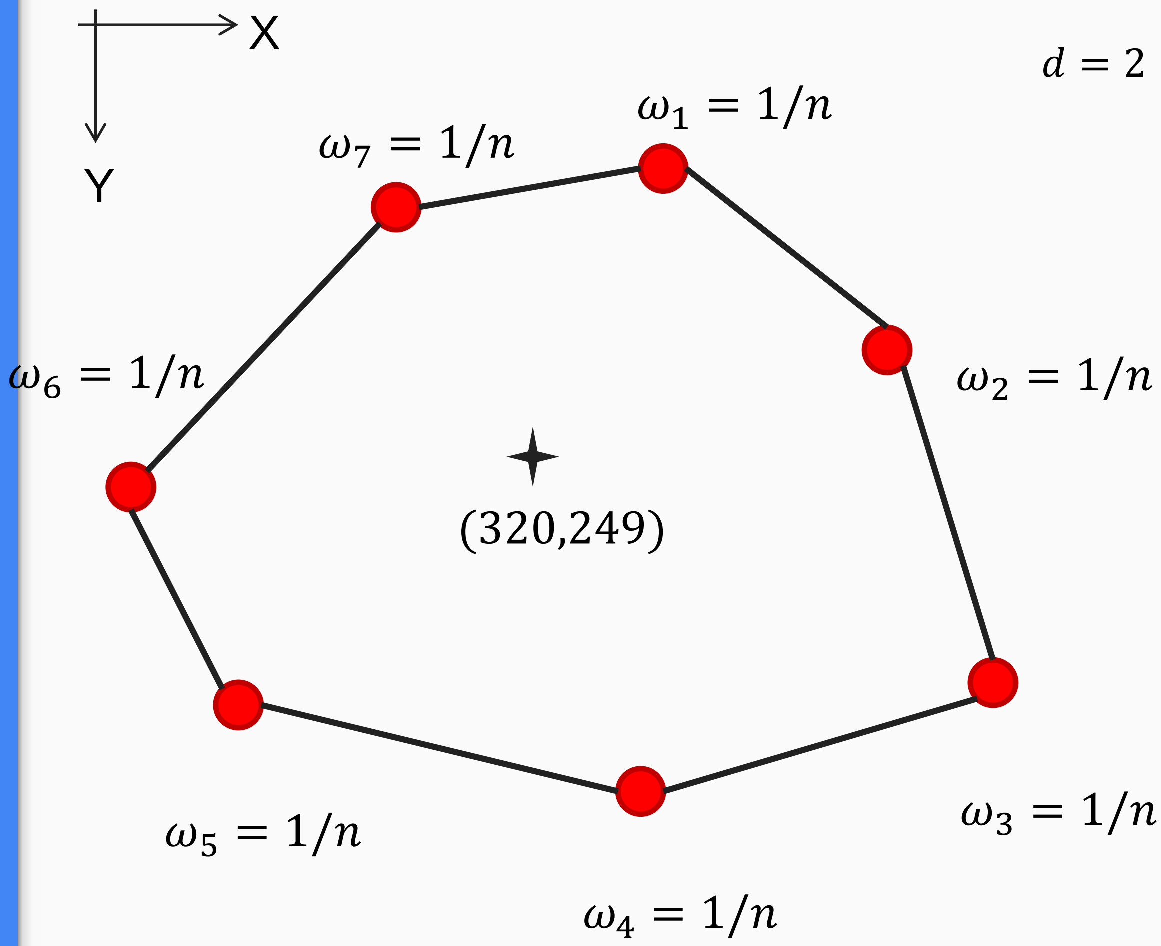


Caratheodory's Theorem

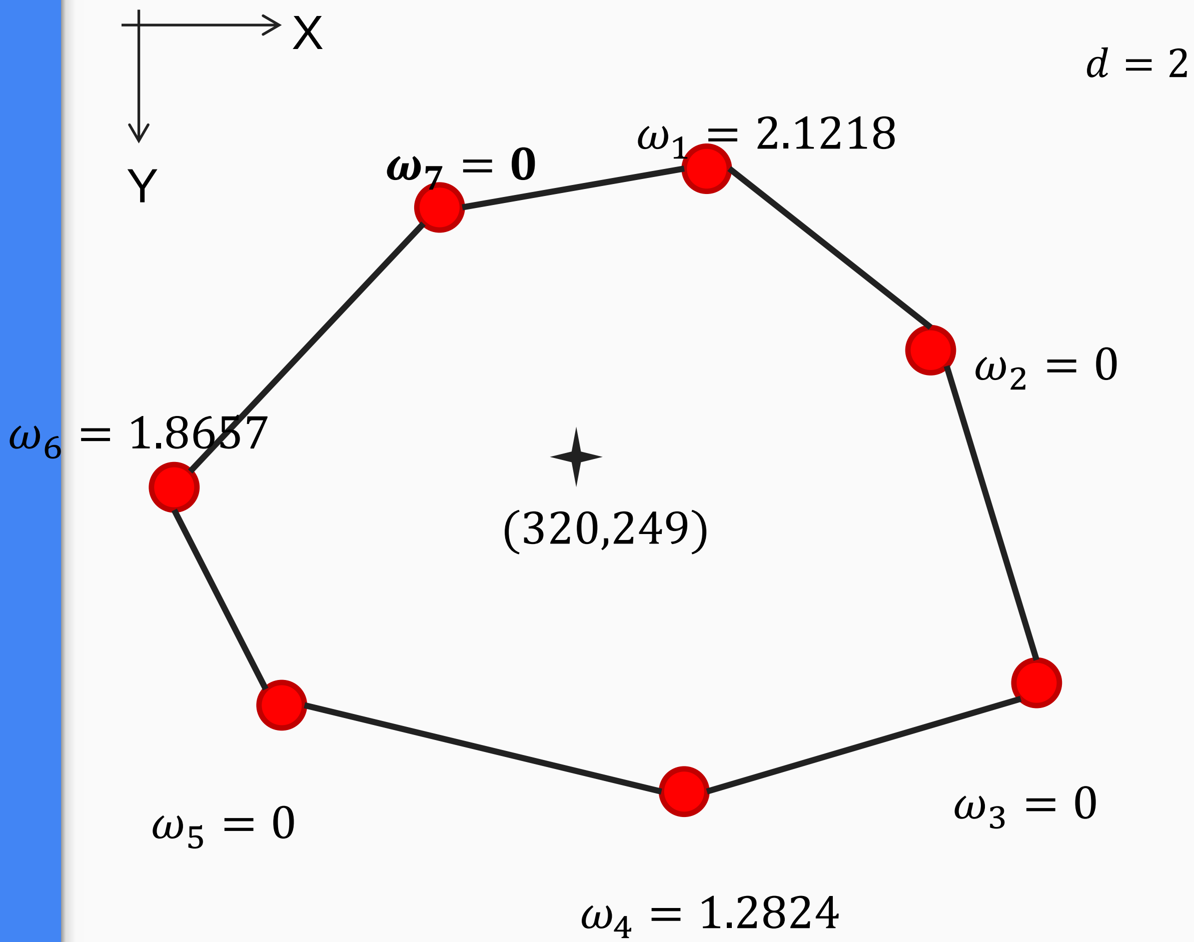
If a point x lies in the convex hull of a set, there is a subset consisting of at most $d + 1$ points such that x lies in the convex hull of P'.

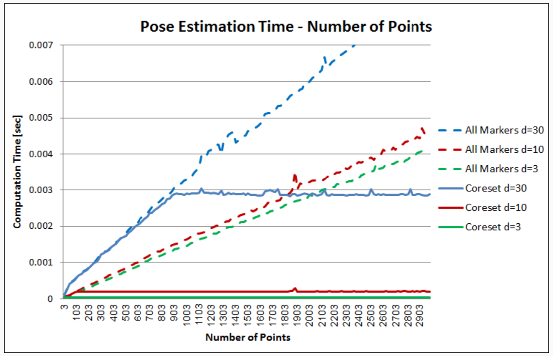


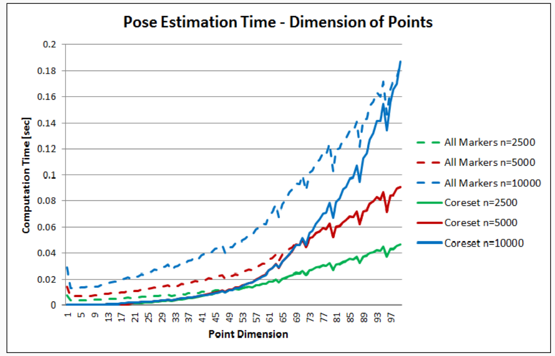
Caratheodory's Theorem (Illustration)

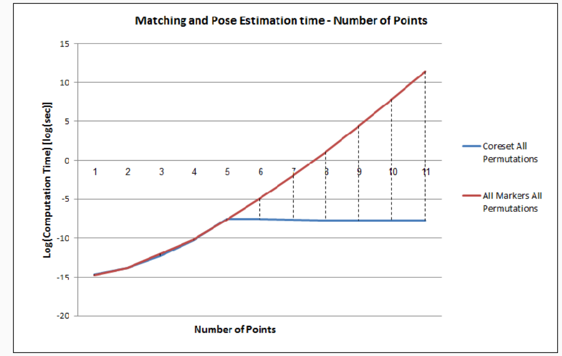


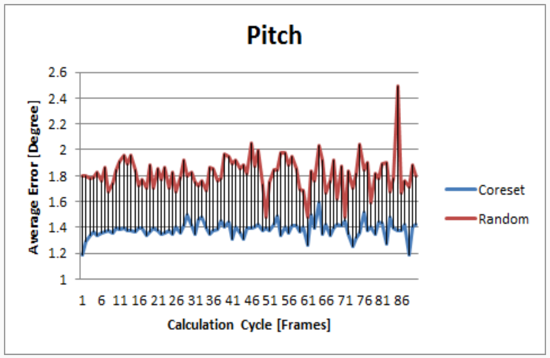
Caratheodory's Theorem (Illustration)

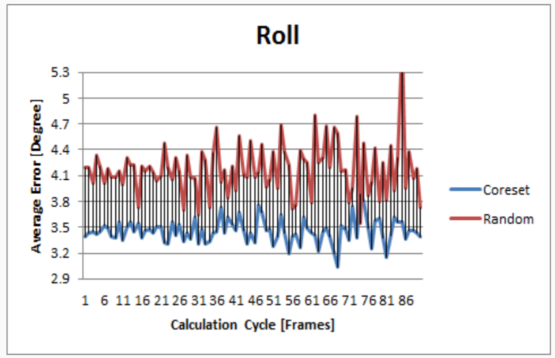


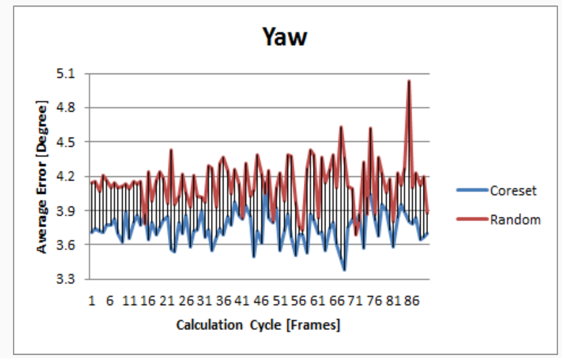

Pose Estimation Time - Number of Points
Computation Time [sec]
Number of Points
All Markers d=30
All Markers d=10
All Markers d=3
Coreset d=30
Coreset d=10
Coreset d=3

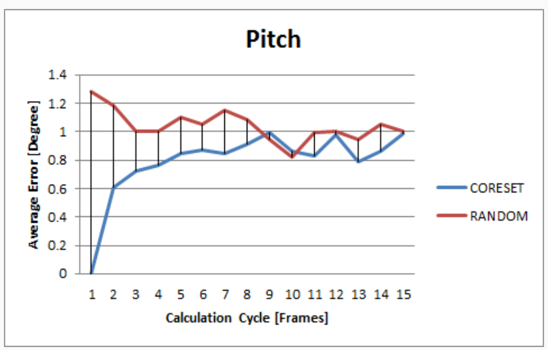

Pose Estimation Time - Dimension of Points
Computation Time [sec]
Point Dimension
All Markers n=2500
All Markers n=5000
All Markers n=10000
Coreset n=2500
Coreset n=5000
Coreset n=10000

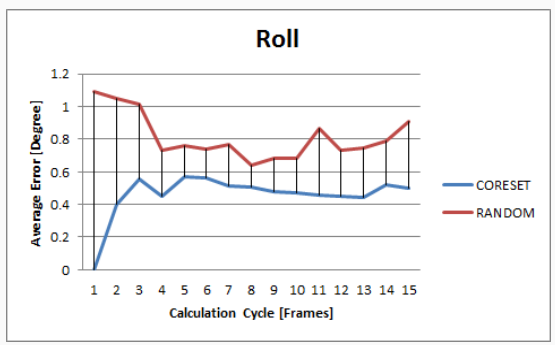

Matching and Pose Estimation time - Number of Points
Log(Computation Time) [log(sec)]
Number of Points
Coreset All Permutations
All Markers All Permutations

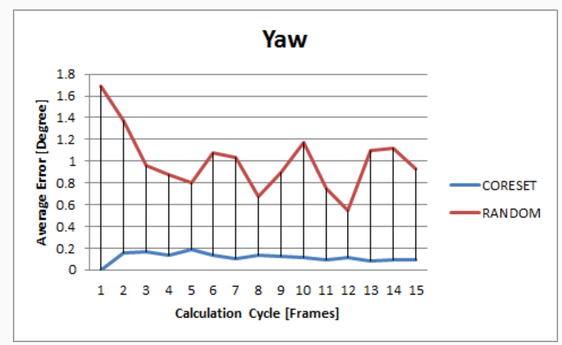

Pitch
Average Error [Degree]
Calculation Cycle [Frames]
Coreset
Random

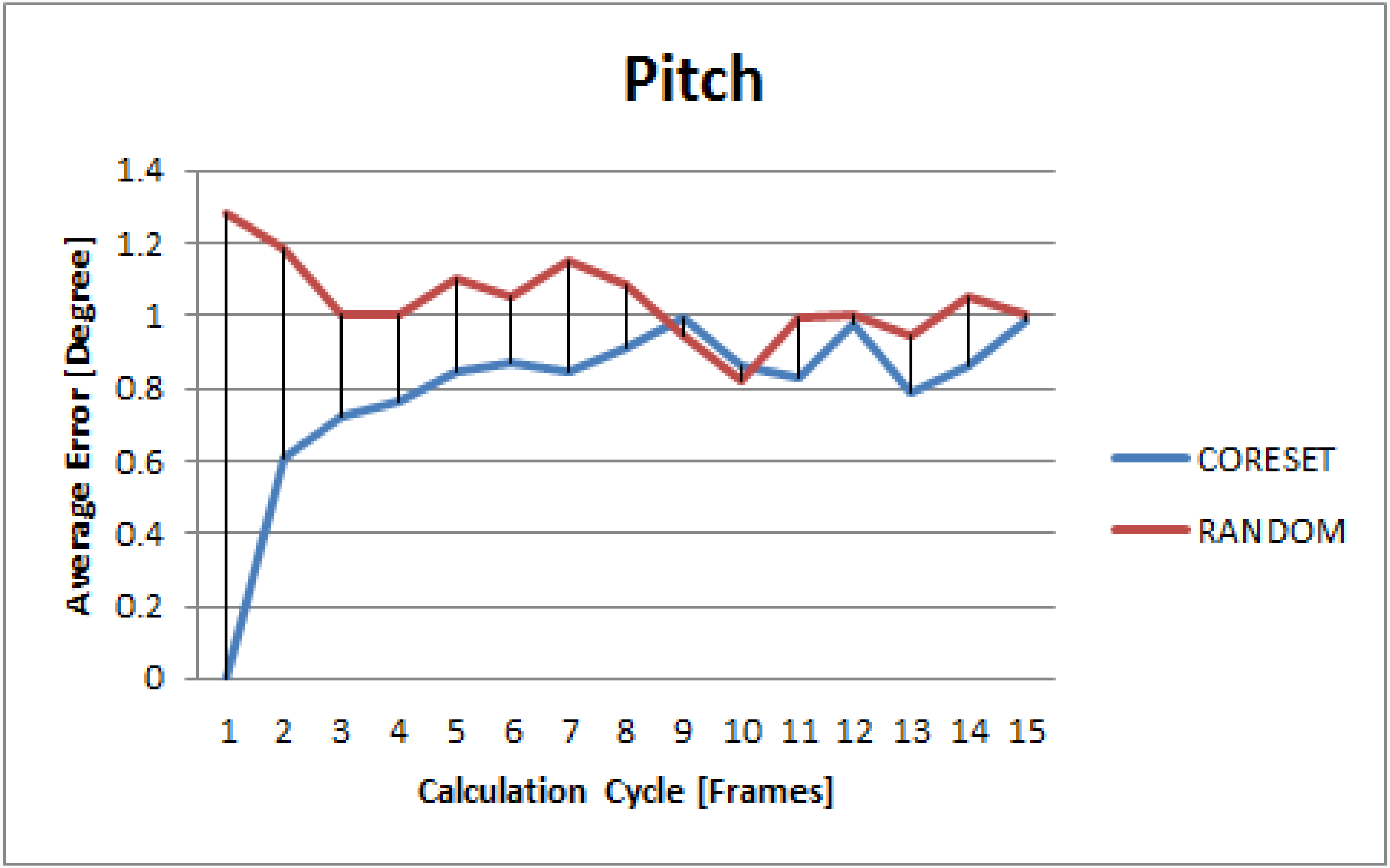

Roll
Average Error [Degree]
Calculation Cycle [Frames]
Coreset
Random


Yaw
Average Error [Degree]
Calculation Cycle [Frames]
Coreset
Random

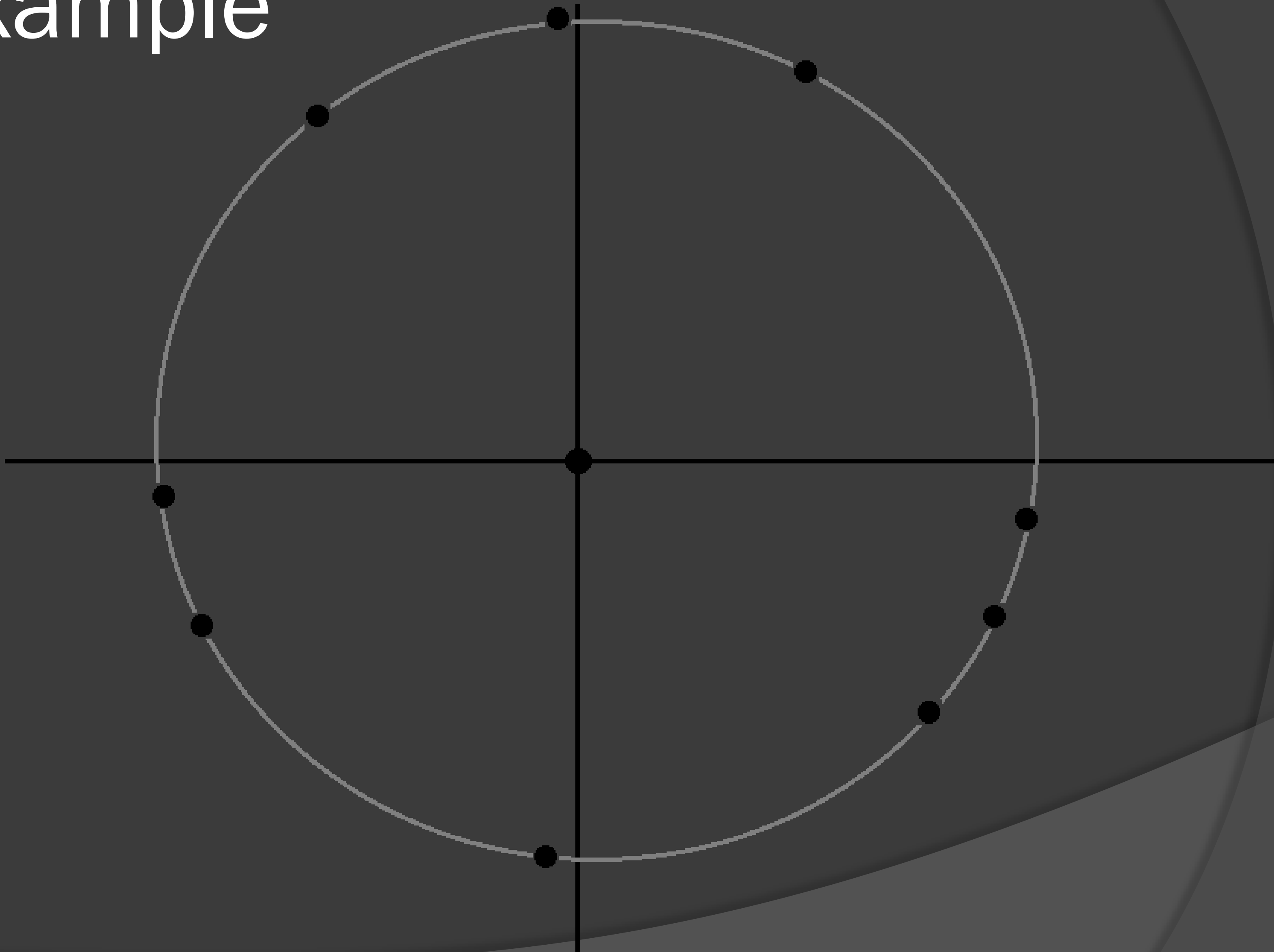

Pitch
Average Error [Degree]
Calculation Cycle [Frames]
CORESET
RANDOM


Roll
Average Error [Degree]
Calculation Cycle [Frames]
CORESET
RANDOM

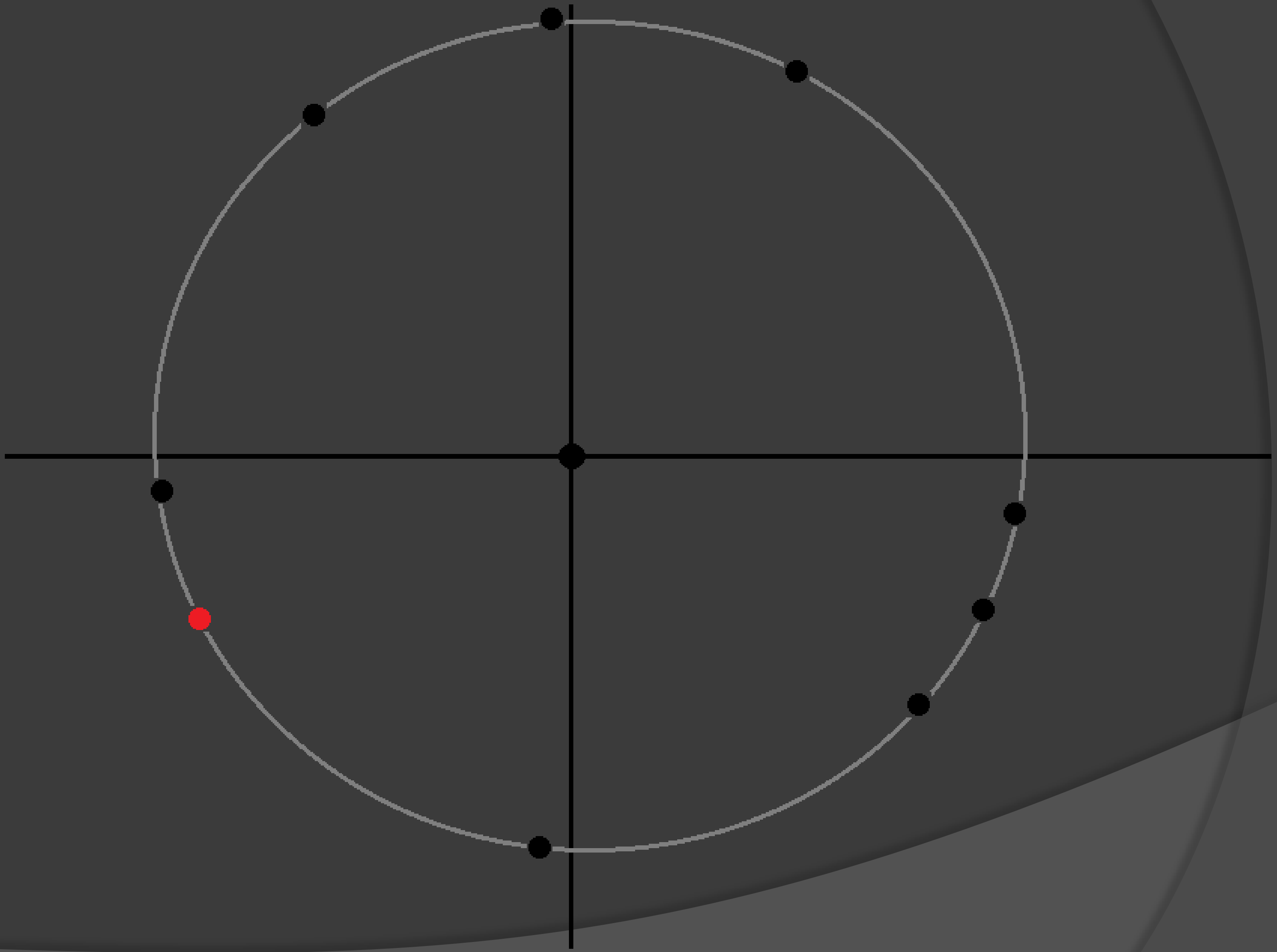

Yaw
Average Error [Degree]
Calculation Cycle [Frames]
CORESET
RANDOM


Pitch
Average Error [Degree]
1.4
1.2
1
0.8
0.6
0.4
0.2
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
Calculation Cycle [Frames]
CORESET
RANDOM

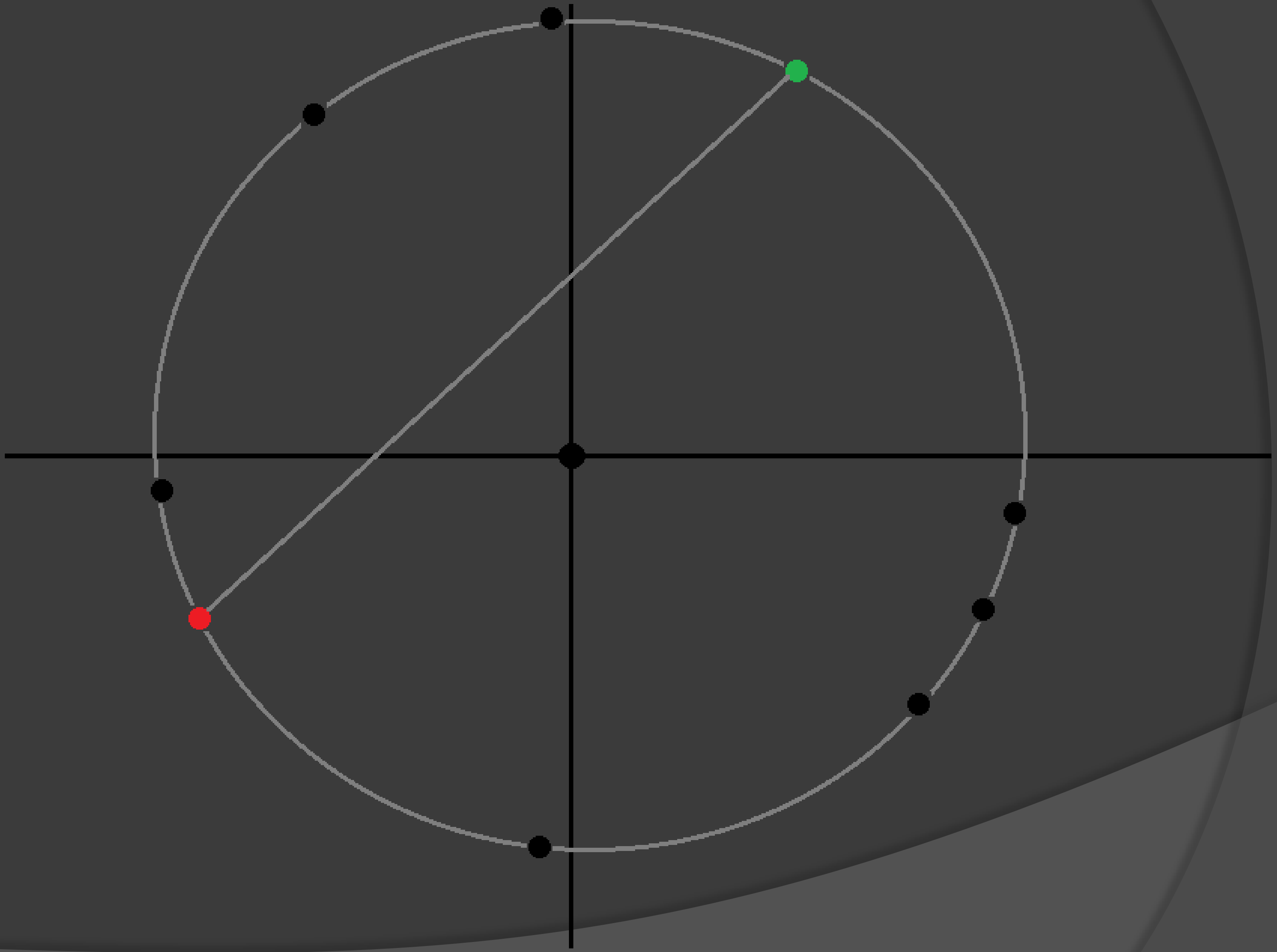
Example



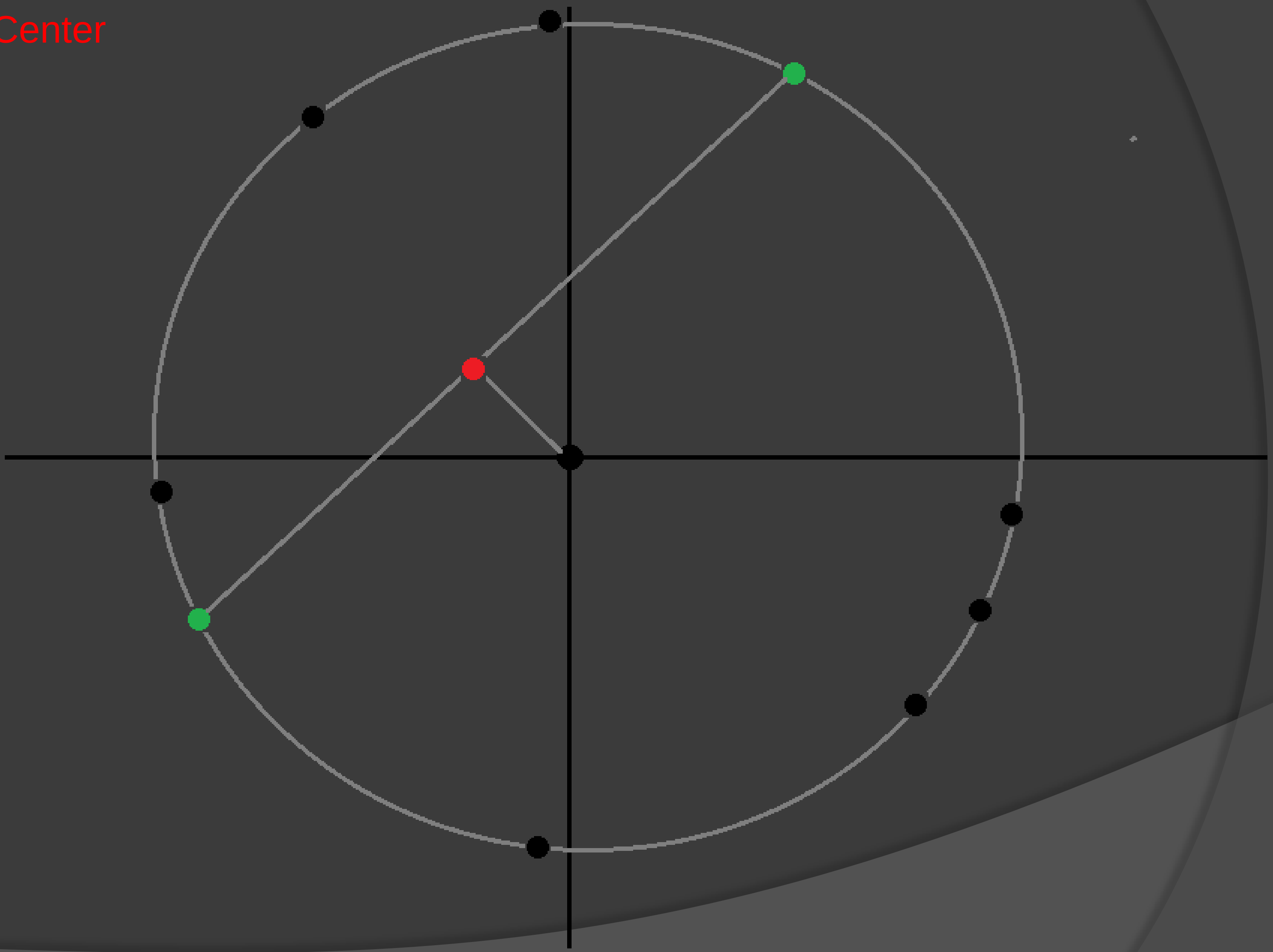
1) Initialize



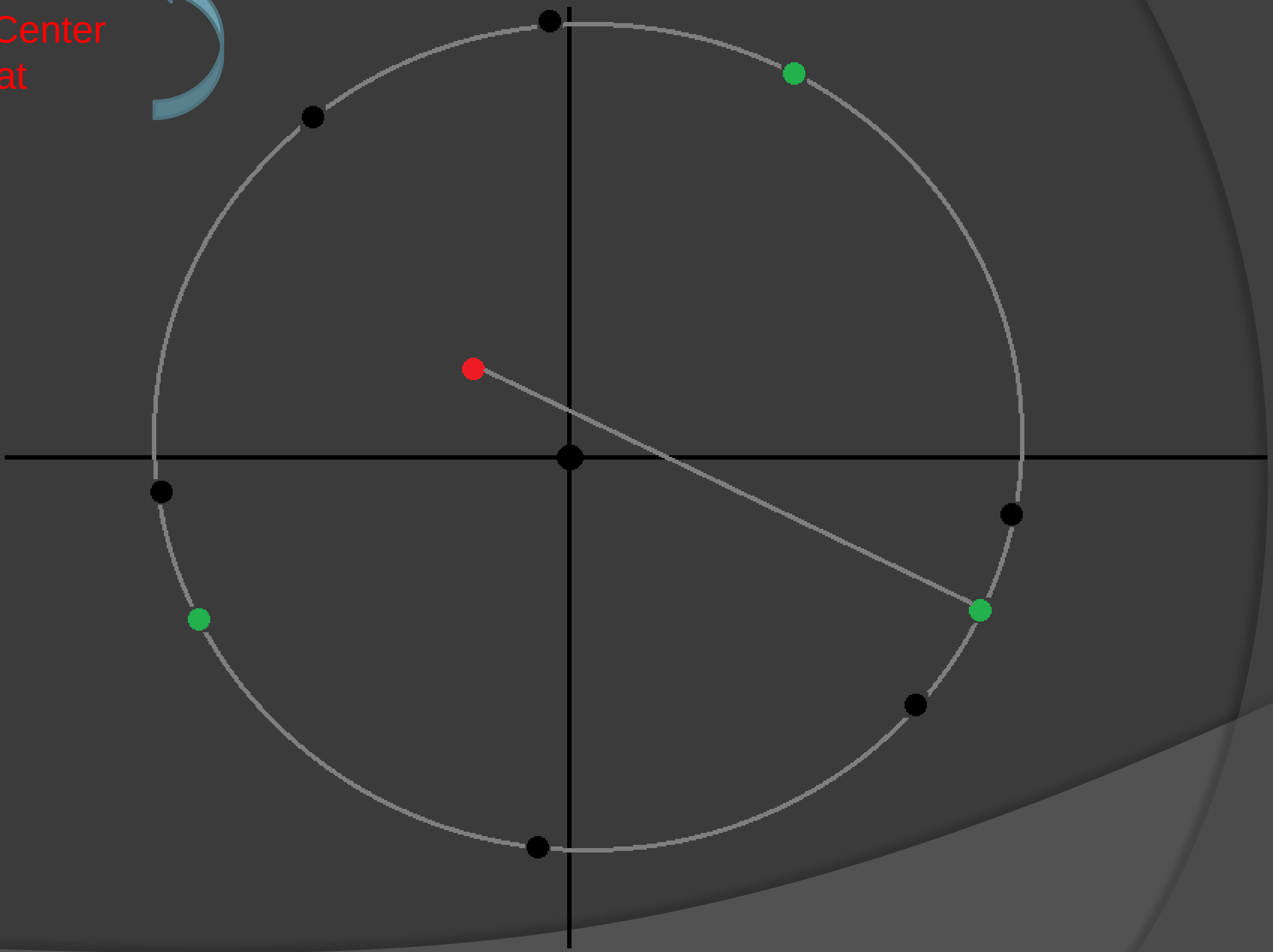
1) Initialize
2) Farthest Point



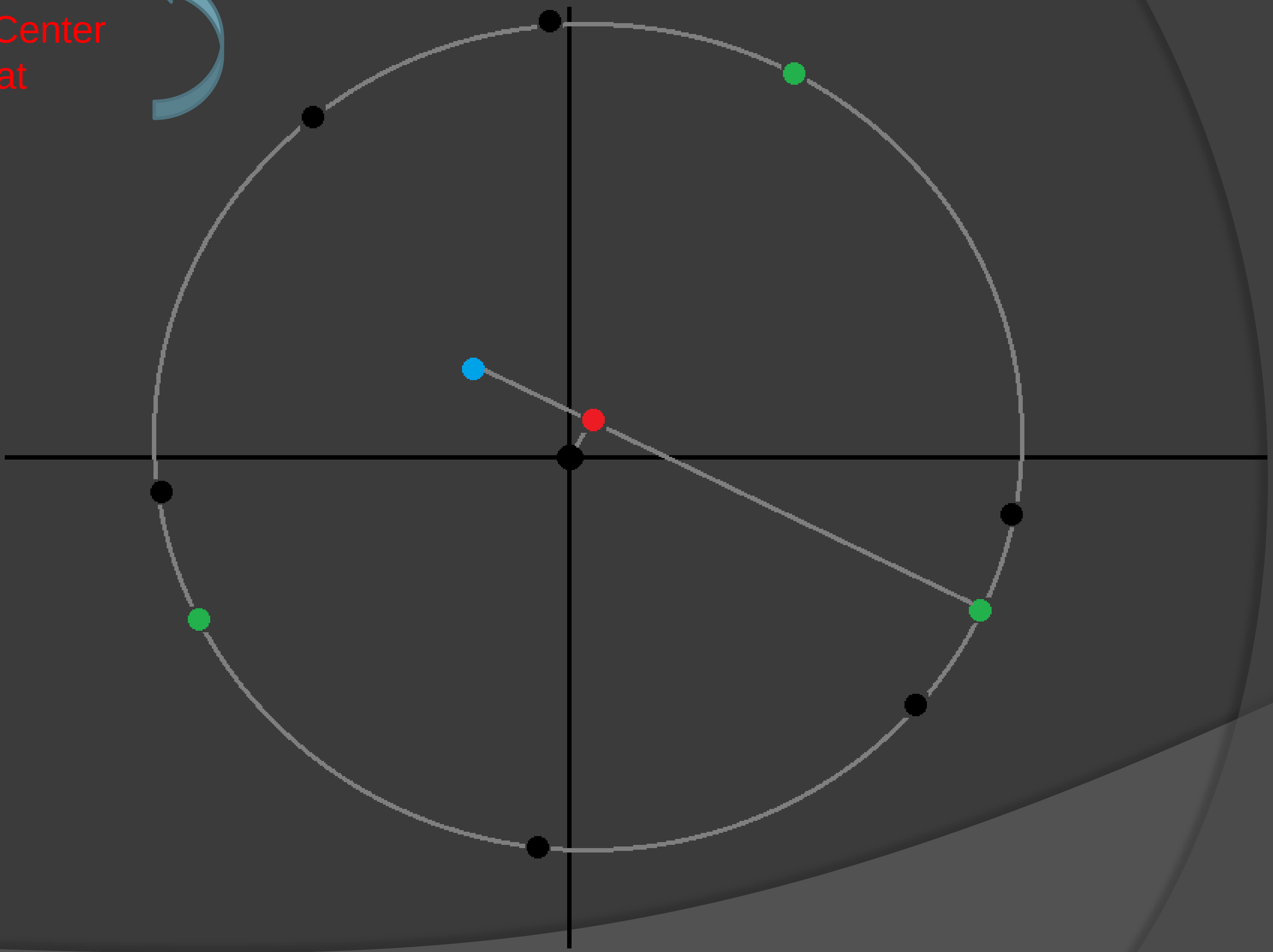
1) Initialize
2) Farthest Point
3) New Center



1) Initialize
2) Farthest Point
3) New Center
4) Repeat



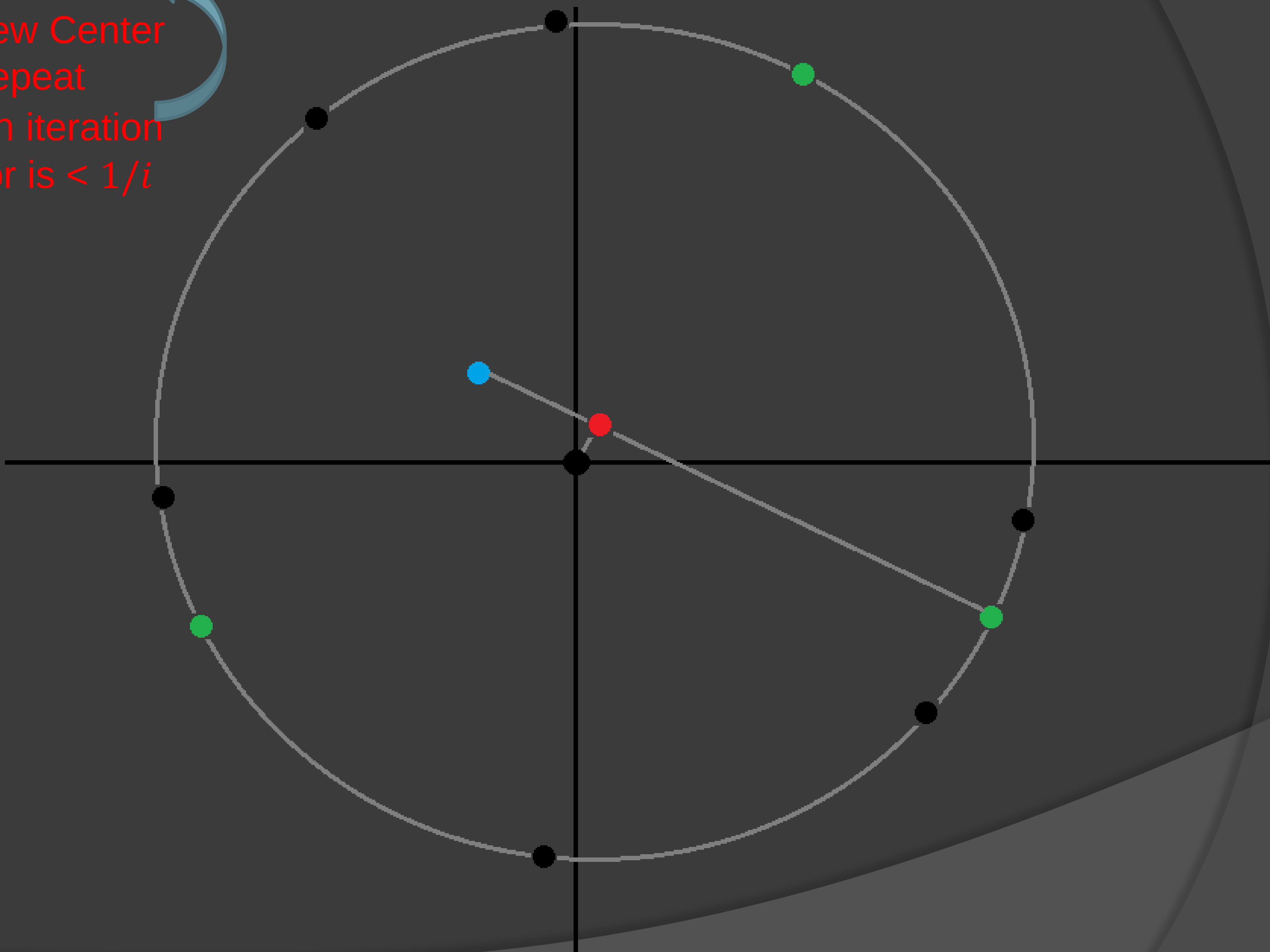
1) Initialize
2) Farthest Point
3) New Center
4) Repeat



1) Initialize
   2) Farthest Point
   3) New Center
   4) Repeat

In the ith iteration

The error is < $1/i$



Open Problems

- More Coresets
  - Deep learning, Topological Data, Sparse data
  - 3D Navigation and Mapping, Robotics
- Sensor Fusion (GPS+Video+Audio+Text+..)
- Private Coresets, [STOC'11, with Fiat et al.]
  - For biometric face database (with R. Osadchy)
- Coresets for Cybersecurity (with S. Goldwasser)
- Generic software library
  - Coresets on Demand on the cloud

Thank you !

Theorem [Feldman, Langberg, STOC'11]

Suppose that

$$\mathrm{cost}(P, q) := \sum_{p \in P} w(p)\mathrm{dist}(p, q)$$

where $\mathrm{dist}: P \times Q \to [0, \infty)$.

A sample $C \subseteq P$ from the distribution

$$\mathrm{sensitivity(p)} = \max_{q \in Q} \frac{dist(p, q)}{\sum_{p'} dist(p', q)}$$

is a coreset if $|C| \geq \frac{\text{dimension of } Q}{\epsilon^2} \cdot \sum_p \mathrm{sensitibity}(p)$

Theorem [Feldman, Langberg, STOC'11]

Suppose that

$$\mathrm{cost}(P, q) := \sum_{p \in P} w(p) \mathrm{dist}(p, q)$$

where $\mathrm{dist}: P \times Q \to [0, \infty)$.

A sample $C \subseteq P$ from the distribution

$$\mathrm{sensitivity(p)} = \max_{q \in Q} \frac{dist(p, q)}{\sum_{p'} dist(p', q)}$$

is a coreset if $|C| \geq \frac{\text{dimension of } Q}{\epsilon^2} \cdot \sum_p \mathrm{sensitibity}(p)$

Surprising Applications

1. (1-epsilon) approximations:
   Heuristics work better on coresets

2. Running constant factor on epsilon-coresets helps

3. Coreset for one problem is good for a lot of unrelated problems

4. Coreset for O(1) points

Dan Feldman dannyf@csail.mit.edu

Implementation

- The worst case and sloppy (constant) analysis is not so relevant

- In Thoery:
  a random sample of size $1/\epsilon$ yields $(1+\epsilon)$ approximation with probability at least $1-\delta$.
  In Practice:
  Sample s points, output the approximation ϵ and its distribution

- Never implement the algorithm as explained in the paper.

Coreset for k-means

[Feldman, Sohler, Monemizadeh, SoCG'07]

Coreset for k-means can be computed by choosing points from the distribution:

$$\text{sensitivity}(p) = \frac{dist(p,q^*)}{\sum_{p'} dist(p',q^*)} + \frac{1}{n_p}$$

q^* = k-means of P

n_p = number of points in the cluster of p

$|C| = \frac{k \cdot d}{\epsilon^2}$

Coreset for k-means

[Feldman, Sohler, Monemizadeh, SoCG'07]

Coreset for k-means can be computed by choosing points from the distribution:

$$\text{sensitivity}(p) = \frac{dist(p,q^*)}{\sum_{p'} dist(p',q^*)} + \frac{1}{n_p}$$

q^* = k-means of P Or approximation [SoCg07, Feldma, Sharir, Fiat]

n_p = number of points in the cluster of p

$|C| = \frac{k \cdot d}{\epsilon^2}$

Coreset for k-means

[Feldman, Sohler, Monemizadeh, SoCG'07]

Coreset for k-means can be computed by choosing points from the distribution:

$$\text{sensitivity}(p) = \frac{dist(p,q^*)}{\sum_{p'} dist(p',q^*)} + \frac{1}{n_p}$$

$q^* = k$-means of P Or approximation [SoCg07, Feldma, Sharir, Fiat]

n_p = number of points in the cluster of p

$$|C| = \cancel{\frac{k\cdot d}{\epsilon^2}}\ \frac{k\cdot\left(\frac{k}{\varepsilon}\right)}{\epsilon^2}$$

[SODA'13, Feldman, Schmidt, ..]

Coreset for k-means

[Feldman, Sohler, Monemizadeh, SoCG'07]

Coreset for k-means can be computed by choosing points from the distribution:

$$\text{sensitivity}(p) = \frac{dist(p,q^*)}{\sum_{p'} dist(p',q^*)} + \frac{1}{n_p}$$

q^* = k-means of P

n_p = number of points in the cluster of p

$|C| = \frac{k \cdot d}{\epsilon^2}$

Coreset for k-means

[Feldman, Sohler, Monemizadeh, SoCG'07]

Coreset for k-means can be computed by choosing points from the distribution:

$$\text{sensitivity}(p) = \frac{dist(p,q^*)}{\sum_{p'} dist(p',q^*)} + \frac{1}{n_p}$$

q^* = k-means of P Or approximation [SoCg07, Feldma, Sharir, Fiat]

n_p = number of points in the cluster of p

$|C| = \frac{k \cdot d}{\epsilon^2}$

Coreset for k-means

[Feldman, Sohler, Monemizadeh, SoCG'07]

Coreset for k-means can be computed by choosing points from the distribution:

$$\text{sensitivity}(p) = \frac{dist(p,q^*)}{\sum_{p'} dist(p',q^*)} + \frac{1}{n_p}$$

q^* = k-means of P Or approximation [SoCg07, Feldma, Sharir, Fiat]

n_p = number of points in the cluster of p

$|C| = \frac{k \cdot d}{\epsilon^2}$

Coreset for k-means

[Feldman, Sohler, Monemizadeh, SoCG'07]

Coreset for k-means can be computed by choosing points from the distribution:

$$\text{sensitivity}(p) = \frac{dist(p,q^*)}{\sum_{p'} dist(p',q^*)} + \frac{1}{n_p}$$

q^* = k-means of P Or approximation [SoCg07, Feldma, Sharir, Fiat]

n_p = number of points in the cluster of p

$|C| = \cancel{\frac{k\cdot d}{\epsilon^2}}\ \frac{k\cdot\left(\frac{k}{\varepsilon}\right)}{\epsilon^2}$ [SODA'13, Feldman, Schmidt, ..]

The chicken-and-egg problem

1. We need approximation to compute the coreset
2. We compute coreset to get a fast approximation to a problem

Lee-ways:

I. Bi-criteria approximation
II. Heuristics
III. polynomial time reduced to linear time by the merge-reduce tree