From approximate incidences to camera posing and navigation

Dror Aiger[1] Haim Kaplan[1,2] Efi Kokiopoulou[1] Micha Sharir[2]
Bernhard Zeisl[1]

[1]Google
[2]Tel-Aviv University

Summary

- What is approximate incidences?
- Duality based efficient algorithms
- Duality based algorithm for camera posing and augmented reality navigation
- Canonical surfaces for approximate incidences

Approximate incidences

- Given two finite sets S_1 and S_2 of geometric primitives (e.g., points, planes, lines, circles, or spheres), and some $\varepsilon > 0$, we define the set of *ε-incidences* to be

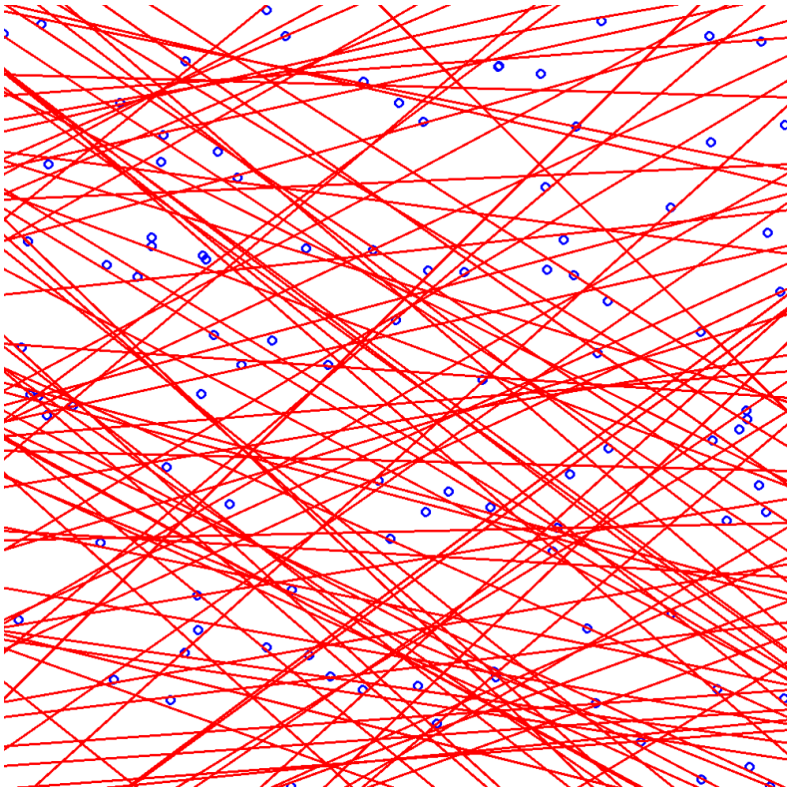
$$I_\varepsilon(S_1, S_2) = \{(s_1, s_2) \mid s_1 \in S_1, s_2 \in S_2, \mathbf{dist}(s_1, s_2) \leq \varepsilon\},$$

  where $\mathbf{dist}(s_1, s_2)$ is the Euclidean distance between s_1 and s_2, i.e.,

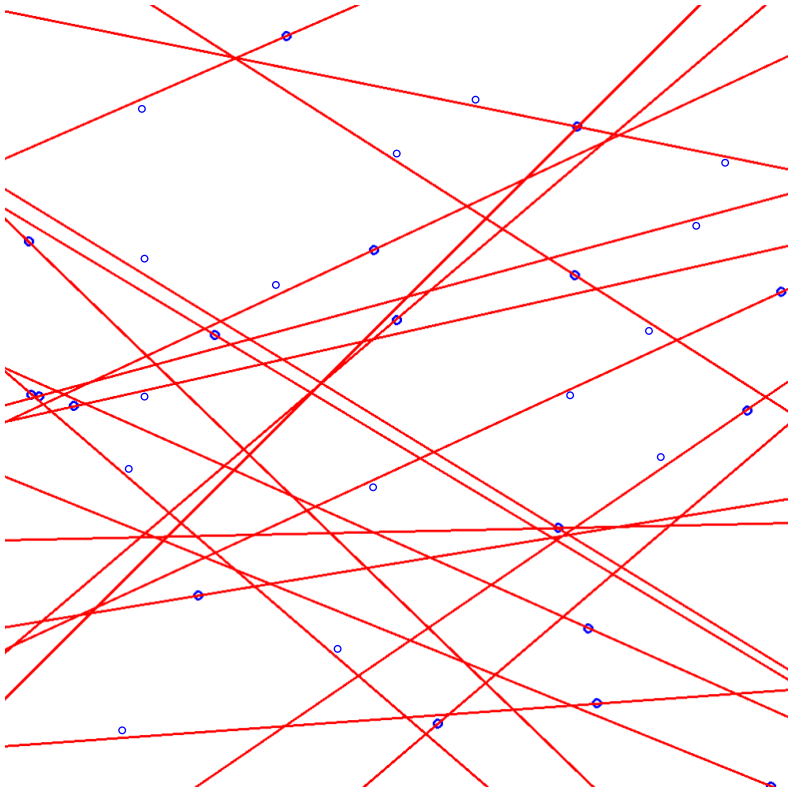
$$\mathbf{dist}(s_1, s_2) = \mathbf{inf}\{\mathbf{dist}(x, y) \mid x \in s_1, y \in s_2\}.$$

- We are interested in efficient algorithms for computing $I_\varepsilon(S_1, S_2)$, ideally in time linear in $|S_1| + |S_2| + |I_\varepsilon(S_1, S_2)|$.

Example: lines and points



Input: lines and points



Approximate incidences

Approximation

- Known algorithms for the exact case $(\varepsilon = 0)$ have large overhead. E.g., for points and lines in the plane, close to $O(m^{2/3}n^{2/3})$, and there are matching lower bounds
- Instead, our goal is algorithms that are really simple to implement, are linear in the input and output sizes, and have good performance in practice

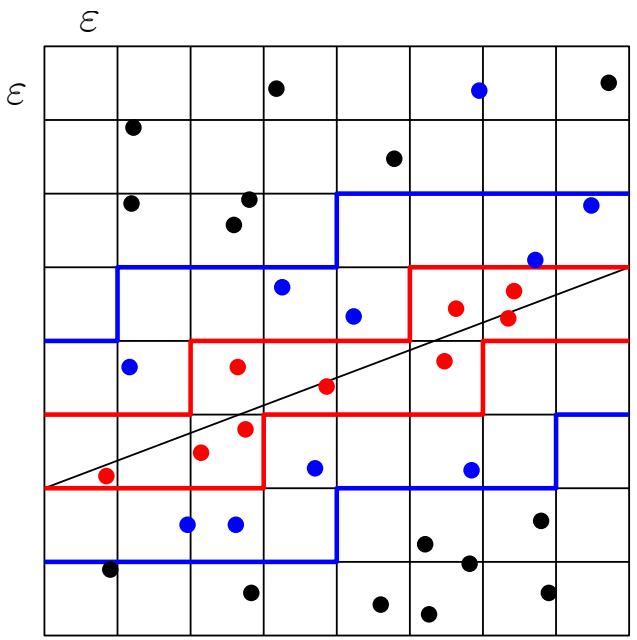
A naive algorithm

Simple example: lines and points in the plane

INPUT: A set L of n lines (in the unit square), A set P of m points, $0 < \varepsilon < 1/2$

OUTPUT: A set of pairs $(p, l) \in P \times L$ containing all pairs of $dist(p, l) \leq \alpha\varepsilon$ for small constant α

- Partition the unit square by a uniform grid, each of whose cells is a square of side length ε
- Store each point in P in a bucket corresponding to the grid cell that contains it
- For each line $l \in L$, report all the pairs involving l and the points in the grid cells that l crosses, and in their neighboring cells



A naive algorithm (Cont.)

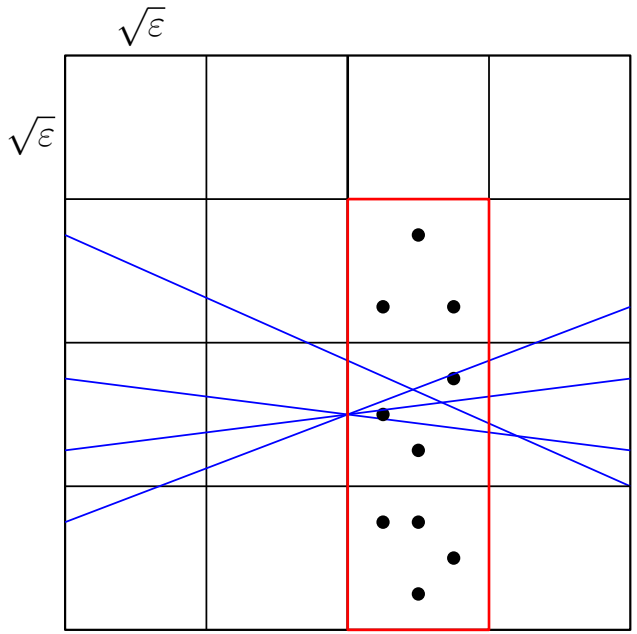
Simple example: lines and points in the plane

- The running time is $O(m + n/\varepsilon + k)$, where k is the number of reported approximate incidences
- All pairs $(p, l) \in P \times L$ with $dist(p, l) \leq \varepsilon$ will be reported, and each reported pair (p, l) satisfies $dist(p, l) \leq 2\sqrt{2}\varepsilon$
- If n is much larger than m, we can use duality and reduce the complexity to $O(n + m + \min(m, n)/\varepsilon + k)$
- This method can be applied in any dimension (increasing runtime)

Efficient algorithm using duality

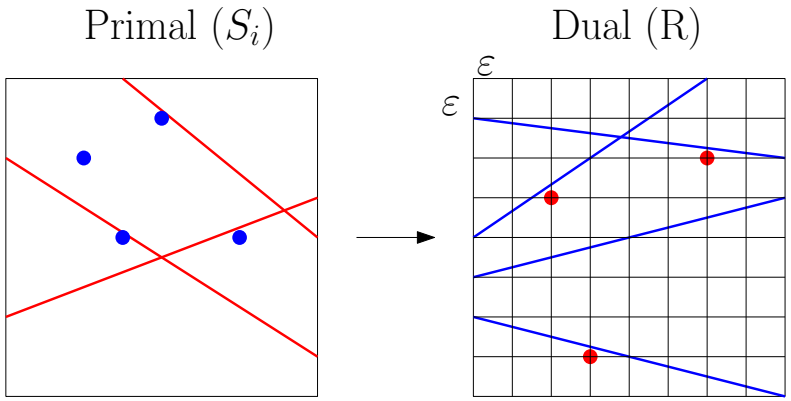
Simple example: n lines and n points in the plane

- Partition $S = [0,1]^2$ into $1/\varepsilon$ squares, $S_1, S_2, ..., S_{1/\varepsilon}$, each of side length $\sqrt{\varepsilon}$
- Compute for each point $p \in P$, the square S_i it belongs to
- Find, for each line $l \in L$ the squares that it crosses, in $O(1/\sqrt{\varepsilon})$ time
- Let L_i be the set of lines crossing S_i or neighboring squares. We then have all the sets P_i,L_i in overall $O(n/\sqrt{\varepsilon})$ time.



Efficient algorithm using duality (Cont.)

- Apply a duality transformation to each S_i separately, let R be the dual bounding rectangle. This duality preserves the vertical distance $dist_v(p, l)$
- Partition R into small rectangles of side length roughly ε
- Find, for each dual point l^* , for $l \in L_i$, the small rectangle that contains it, and, for each dual line p^* , for $p \in P_i$, the small rectangles that it crosses. This takes $O(n_i + m_i/\sqrt{\varepsilon})$ time



Efficient algorithm using duality (Cont.)

- Report, for each small rectangle R' , all the pairs $(p, l) \in P_i \times L_i$ for which l^* lies in R' and p^* crosses R'
- Repeat this over all small squares S_i and all respective neighboring rectangles
- The overall runtime is $O(n/\sqrt{\varepsilon})$
- For n lines and m points, a refined partition gives overall runtime of $O(n + m + \sqrt{nm/\varepsilon})$

Omitted details

- We need to partition L into two groups to bound $dist_v(p, l)$
- We actually need more accurate values for the smaller grid
- Neighboring grid cells should also be considered

Extensions to any dimension

- The machinery works for points and hyperplanes with runtime $O(m + n + \sqrt{nm}/\varepsilon^{(d-1)/2} + k)$
- For spheres, a more careful treatment with some variant gives runtime of $O((n + m)/\varepsilon^{(d-1)/2} + k)$
- The naive approach applied to points and hyperplanes/spheres would take $O(m + n/\varepsilon^{d-1} + k)$

Extensions

The duality framework can be extended to other kinds of surfaces in various dimensions with an appropriate duality transform.

Motivation: camera posing and augmented reality navigation

Index a huge collection of images with known pose (E.g. Google Street View), such that for any query image we can compute its camera pose in the world (6DOF) using the index (in real time).

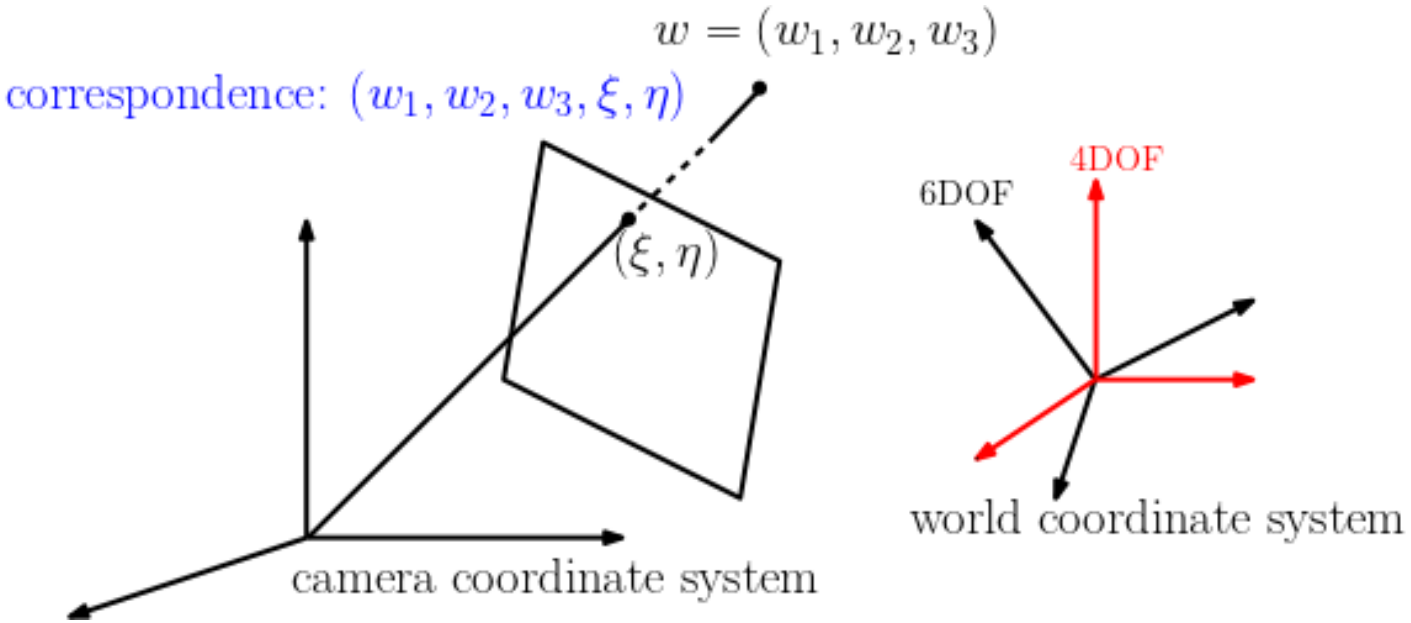
- ▸ Google AR Walking Navigation (Live View in Google maps)

Camera posing

Problem

Given n 2D-3D correspondences between points in the scene and points in the camera image (most of which are incorrect associations), determine the camera pose.

Using device IMUs (inertial measurement unit), we get the camera vertical direction, so we can work with 4DOF (translation and yaw, x, y, z, θ).



From posing to approximate incidences

- Posing: find a point in 4-space that maximizes the number of correspondences such that (w_1, w_2, w_3) is projected "close" to (ξ, η) in image frame.
- Each correspondence $\mathbf{w} = (w_1, w_2, w_3, \xi, \eta)$ defines a two-dimensional surface $\sigma_{\mathbf{w}}$ in parametric 4- space, which is the locus of all poses of the camera at which it sees (w_1, w_2, w_3) at coordinates (ξ, η) in its image (two equalities on four variables).
- Posing becomes: find a point in 4-space that maximizes the number of surfaces "close" to it.
- So we face: ε-Approximate incidences between points and 2-surfaces in 4-space.

The surfaces $\sigma_{\mathbf{w}}$

- Each correspondence is parameterized by the $\mathbf{5}$-tuple $(w_1, w_2, w_3, \xi, \eta)$.
- For a camera pose $v = (x, y, z, \kappa)$ ($\kappa = tan(\theta)$), and a point $w = (w_1, w_2, w_3)$, we write

$$\xi_v = \frac{(w_2 - y) - \kappa(w_1 - x)}{(w_1 - x) + \kappa(w_2 - y)}$$

$$\eta_v = \frac{w_3 - z}{\sqrt{(w_1 - x)^2 + (w_2 - y)^2}}$$

- This defines the projection of (w_1, w_2, w_3) on the image plane for the pose v.

The surfaces $\sigma_{\mathbf{w}}$

- The surface $\sigma_{\mathbf{w}}$ for a correspondence $\mathbf{w} = (w_1, w_2, w_3, \xi, \eta)$:

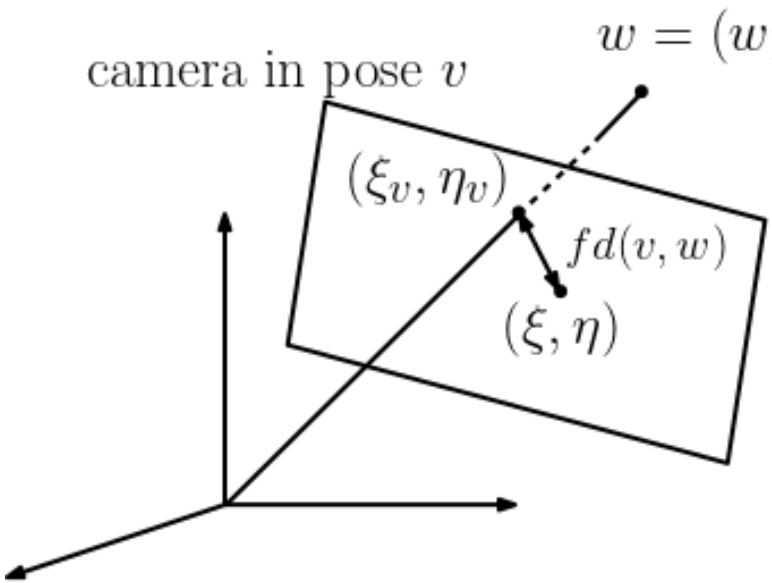
$$\xi = \frac{(w_2 - y) - \kappa(w_1 - x)}{(w_1 - x) + \kappa(w_2 - y)}$$

$$\eta = \frac{w_3 - z}{\sqrt{(w_1 - x)^2 + (w_2 - y)^2}}$$

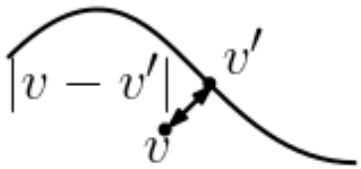
- Defines all poses (x, y, z, θ) ($\kappa = tan(\theta)$) where (w_1, w_2, w_3) is projected to (ξ, η).

Proximity

- For a pose $v = (x, y, z, \kappa)$ and a correspondence $\mathbf{w} = (w, \xi, \eta)$, $w = (w_1, w_2, w_3)$, we define the frame distance fd between v and w as the L_∞-distance between (ξ, η) and (ξ_v, η_v): $fd(v, w) = max\{|\xi_v - \xi|, |\eta_v - \eta|\}$.
- We prove (with minor assumptions): $|v - v'| \leq \varepsilon \Rightarrow fd(v, w) < \beta\varepsilon$ and $fd(v, w) \leq \varepsilon \Rightarrow |v - v'| \leq c\varepsilon$ for small constants c, β.



camera coordinate system



2-surface representing (w, ξ, η) in 4-space and a pose v

Primal-dual algorithm for geometric proximity

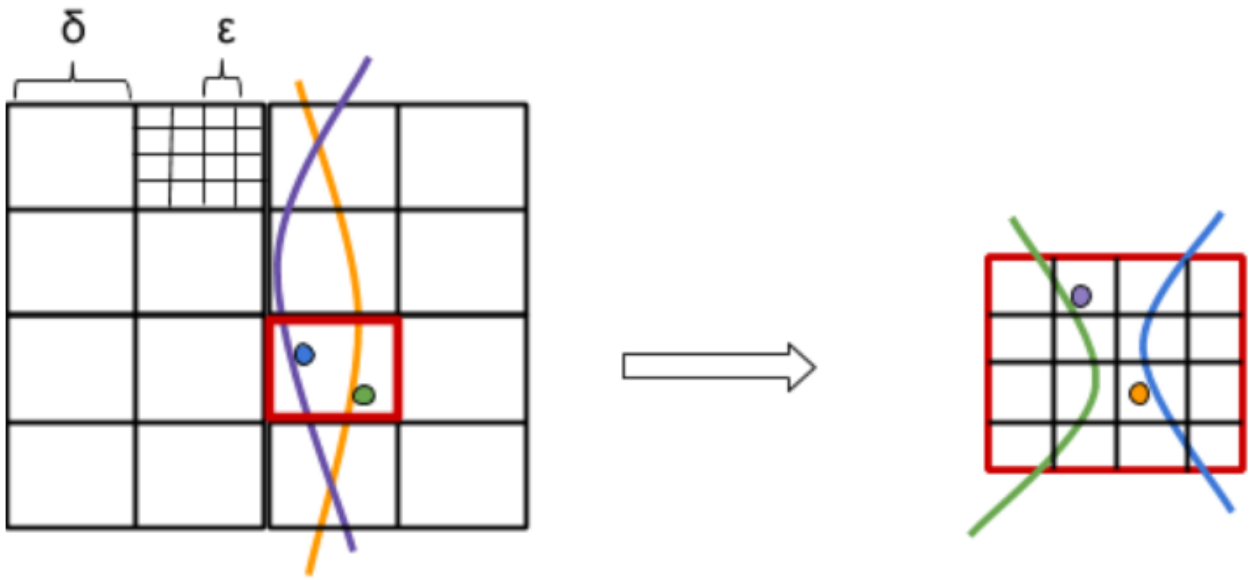
A (very rough) overview of of a two-stage primal-dual algorithm:

- Pick some δ and create a primal grid G_δ of cell size roughly δ.
- Apply the simple algorithm on G_δ.
- Discretize the set of all possible positions of the camera by the vertices of a finer grid G_ε.
- Dualize in each primal cell, τ, of G_δ, surfaces to points and vertices of G_ε to surfaces in appropriate space.
- Apply the simple algorithm in each τ of G_δ using G_ε.

Select δ that balances the runtime of the two stages.

Primal-dual algorithm for geometric proximity

- We use a dual five-dimensional space.
  surfaces $\equiv$ correspondences $\equiv$ dual points
  poses $\equiv$ $\mathbf{3}$-surfaces in dual $\mathbf{5}$-space



*We omit the technical details.

Primal-dual algorithm for geometric proximity

We prove (with minor assumptions and suitable values of grid cell sizes):

Theorem 1.

Given n data points that are seen (and identified) in a two-dimensional image taken by a vertically positioned camera, and an error parameter $\varepsilon > 0$, we can compute, in $O\left(\frac{n^{3/5}}{\varepsilon^{14/5}} + n + \frac{1}{\varepsilon^4}\right)$ time, a vertex v of G_ε that maximizes the approximate count of ε-incident correspondences, where "approximate" means that every correspondence $\mathbf{w}$ whose surface $\sigma_{\mathbf{w}}$ is at frame distance at most ε from v is counted and every correspondence that we count lies at frame distance at most $\alpha\varepsilon$ from v, for some fixed constant α.

This is asymptotically faster than the simple algorithm for $n > 1/\varepsilon^2$.

Geometric proximity via canonical surfaces

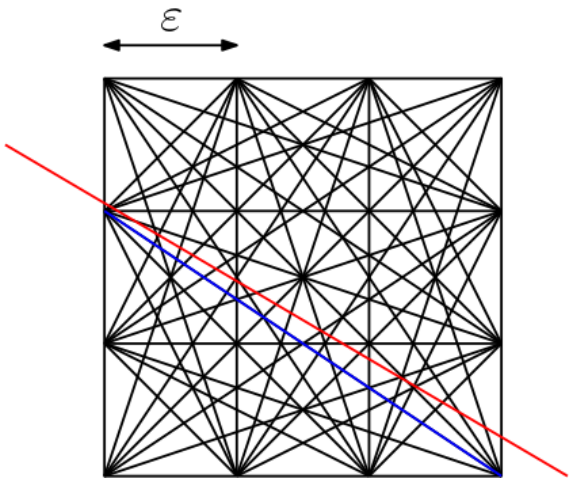
- We have a set S of n k-dimensional algebraic surfaces in $\mathbb{R}^d$ that cross the unit cube $[0,1]^d$, and a given error parameter ε.
- We assume that each $\sigma \in S$ is defined in terms of ℓ *essential parameters* $\mathbf{t} = (t_1, \ldots, t_\ell)$, and $d-k$ additional *free additive parameters* $\mathbf{f} = (f_{k+1}, \ldots, f_d)$:

$$x_j = F_j(x_1, \ldots, x_k; t_1, \ldots, t_\ell) + f_j, \text{for } j = k+1, \ldots, d$$

- We assume that t (resp., f) varies over $[0,1]^\ell$ (resp., $[0,1]^{d-k}$).

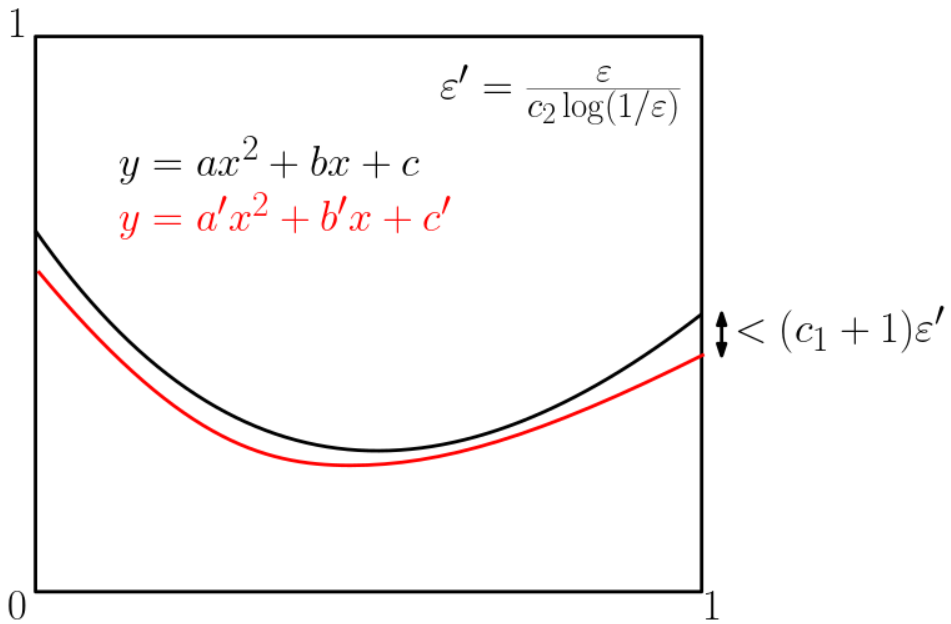
Geometric proximity via canonical surfaces

- We present a general technique to preprocess S into a data structure that can answer approximate incidences queries.
- We extend the case of halfspaces [Fonseca and Mount, 2010] to general surfaces, and apply it to the specific instance of 2-surfaces in 4-space that arise in the camera pose problem.
- Main trick is canonization: round the n original surfaces into a set of canonical surfaces, whose number depends only on ε, such that each original surface has a canonical surface that is "close" to it.



Canonizing the input surfaces

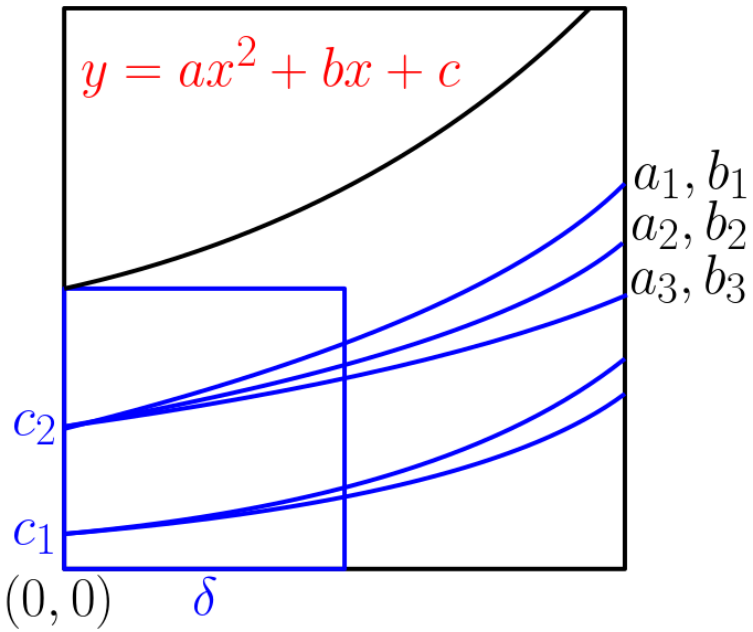
- All surfaces have bounded gradient with constant c_1 and bounded Lipschitz gradients with constant c_2.
- Any surface has ℓ essential parameters (gradient dependent) and free parameters (in the figure, a and b are essential and c is free).
- We round each coordinate to a multiple of $\varepsilon'/(\ell+1)$.



Quadtree-like method

- Surfaces are propagated down.
- We round all surfaces so that at each level the total number of surfaces is roughly the same.
- Each surface gets coarser and coarser as the level increases.
- For each smaller box of size $\delta = 1/2^j$, we round each coordinate in the essential parameters to a multiple of $\frac{\varepsilon'}{(\ell+1)\delta}$ and each free coordinate to a multiple of $\frac{\varepsilon'}{(\ell+1)}$.
- At the smallest box (of size ε), return all surfaces that got rounded to surfaces that cross this box.
- The process accumulates error (so we appropriately update ε to ε').

Quadtree-like method



- In the smaller square, all these can be rounded to have the same a, b if we anchor them all at the origin.
- The number of possible values of c is bounded since they all intersect the small square.
- The range of relevant values of $c \sim \delta$.

The number of surfaces

- We prove: the overall size of the structure, taking also into account the cost of the first phase, is

$$\begin{cases} O\left(\frac{1}{(\varepsilon')^{\ell+d-k}}\right) & \text{for } \ell > k \\ O\left(\frac{1}{(\varepsilon')^{d}} \log \frac{1}{\varepsilon}\right) & \text{for } \ell = k \\ O\left(\frac{1}{(\varepsilon')^{d}}\right) & \text{for } \ell < k. \end{cases} \tag{1}$$

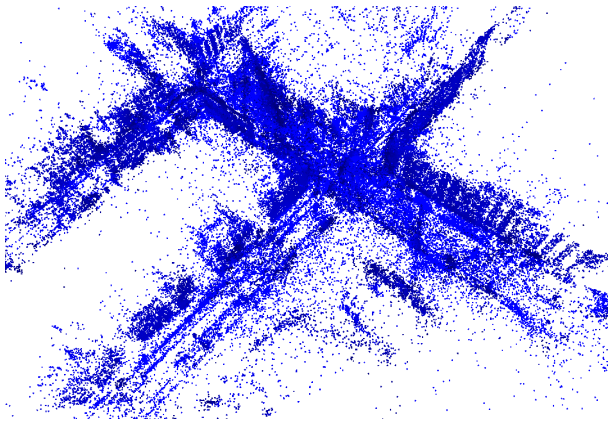
The overall size of the structure

Theorem 2.

Let S as above and let G be the (4ε)-grid within $[0,1]^d$. The algorithm reports for each vertex v of G an approximate ε-incidences count that includes all surfaces at distance at most ε from v and may include some surfaces at distance at most $(2\sqrt{d}+1)\varepsilon$ from v. The running time of this algorithm is proportional to the total number of rounded surfaces that it generates, which is given by Equation (1), plus an additive $O(n)$ term for the initial canonization of the surfaces.

For our camera posing problem, $\ell = 4, k = 2, d = 4$ which implies that the algorithm has running time of $O(\log(1/\varepsilon)/\varepsilon^6))$. It is asymptotically better than the primal-dual for $n \geq 1/\varepsilon^{16/3}$

Implementation and running time on real data



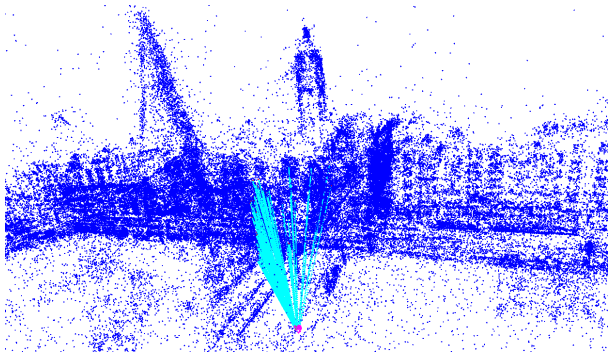
(a) *140000* $\mathbf{3D}$ *landmarks in correspondence to image features in 3c.*

(b) *Query image.*

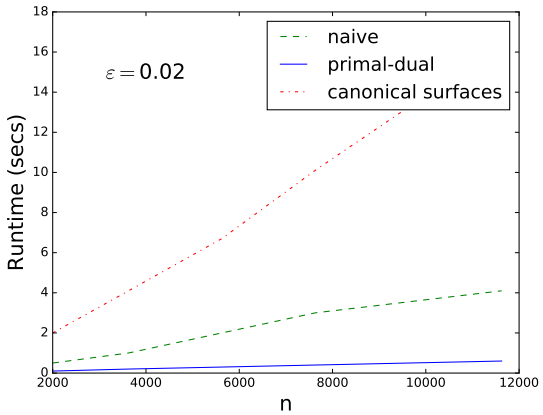
(c) *Corresponding* (ξ, η) *points, each corresponds to many landmarks.*

Implementation and running time on real data

(a) *Inliers ((ξ, η) with ε-close projection of landmarks) found by the algorithm (green) along with the rays from matched landmarks*



(b) *The pose found by the algorithm (within 3m and 10 degrees from the known true pose) with rays to matched landmarks.*



(c) *Runtime*

Conclusions

- General (implementable) techniques for approximate incidences between surfaces and points, all linear in the number of surfaces.
- Non-trivial dependency on ε.
- Application to camera posing that gives the first non-trivial efficient algorithm for this problem.