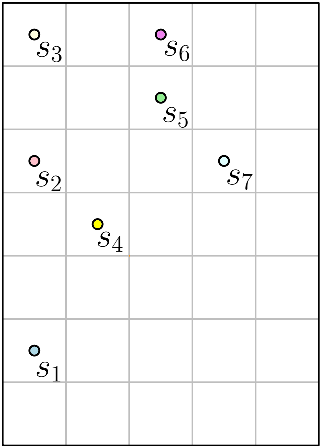
Computing the Nearest Neighbor Transform Exactly with only Double Precision

DAVID L. MILLMAN Steven Love
Timothy M. Chan Jack Snoeyink

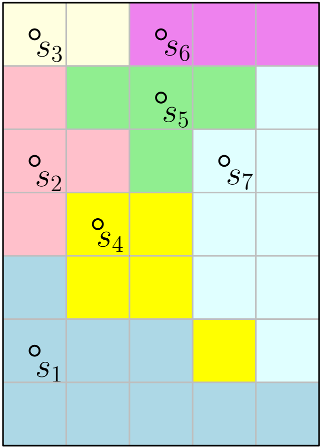
20 April 2013

The Problem



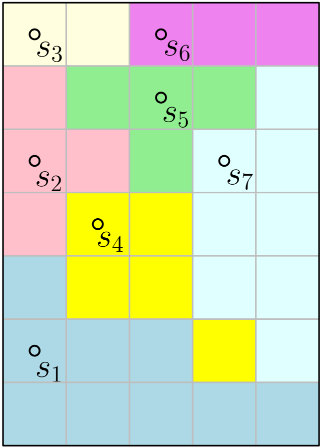
Given *n* sites on a pixel grid, what is the closest site to each pixel?

The Problem



Given *n* sites on a pixel grid, what is the closest site to each pixel?

The Problem



Given *n* sites on a pixel grid, what is the closest site to each pixel?

How much precision is need to determine this?

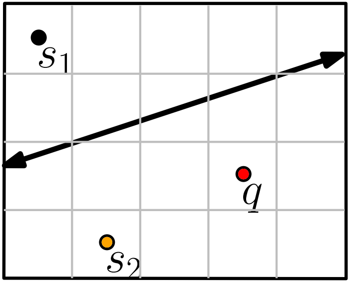
Other precision/robust approaches

Techniques for implementing geometric algorithms with finite precision computer arithmetic:

- Rely on machine precision (+epsilon)
- Topological Consistency [S99, S01, SI90, SI92, SII*00]
- Exact Geometric Computation [Y97]
  - Software based arithmetic [CORE, LEDA, MPFR]
  - Predicate eval. schemes [C92, FW93, ABO*97, S97]
  - Degree-driven algorithm design [LPT99]

Analyzing Precision[LPT99]

Is q closer to s_1?

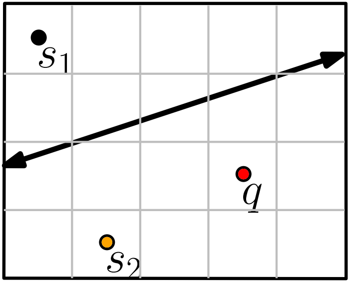


$$
\begin{aligned}
\mathbb{U} &= \{1, 2, \ldots, U\} \\
s_1,\ s_2,\ &q \in \mathbb{U}^2 \\
s_1 &= (x_1, y_1) \\
s_2 &= (x_2, y_2) \\
q &= (x_q, y_q)
\end{aligned}
$$

$$\|q - s_1\|^2 \gtrless \|q - s_2\|^2$$

Analyzing Precision[LPT99]

Is q closer to s_1?

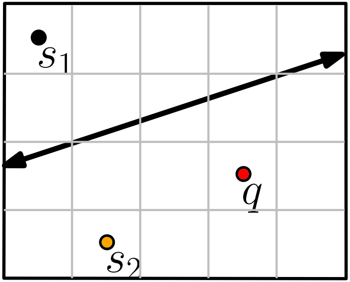


$$
\begin{aligned}
\mathbb{U} &= \{1, 2, \ldots, U\} \\
s_1, \quad s_2, \quad q &\in \mathbb{U}^2 \\
s_1 &= (x_1, y_1) \\
s_2 &= (x_2, y_2) \\
q &= (x_q, y_q)
\end{aligned}
$$

$$\|q - s_1\|^2 \quad \gtrless \quad \|q - s_2\|^2$$

$$f(s_1, s_2, q) = \mathrm{sign}((x_q - x_1)^2 + (y_q - y_1)^2 - (x_q - x_2)^2 - (y_q - y_2)^2)$$

Is q closer to s_1?



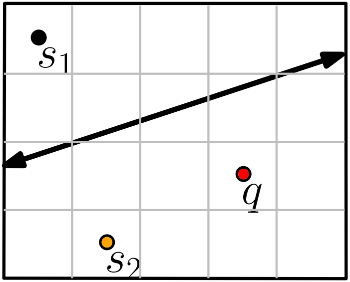
$$\mathbb{U} = \{1, 2, \ldots, U\}$$
$$s_1, \quad s_2, \quad q \in \mathbb{U}^2$$
$$s_1 = (x_1, y_1)$$
$$s_2 = (x_2, y_2)$$
$$q = (x_q, y_q)$$

$$\|q - s_1\|^2 \quad \gtrless \quad \|q - s_2\|^2$$

$$f(s_1, s_2, q) = \mathrm{sign}((x_q - x_1)^2 + (y_q - y_1)^2 - (x_q - x_2)^2 - (y_q - y_2)^2)$$
$$= \mathrm{sign}(x_1^2 - 2x_1 x_q - 2y_1 y_q + y_1^2 - x_2^2 + 2x_2 x_q + 2y_2 y_q - y_2^2$$

Analyzing Precision[LPT99]

Is q closer to s_1?



$$\mathbb{U} = \{1, 2, \ldots, U\}$$
$$s_1,\quad s_2,\quad q \in \mathbb{U}^2$$
$$s_1 = (x_1, y_1)$$
$$s_2 = (x_2, y_2)$$
$$q = (x_q, y_q)$$

$$\|q - s_1\|^2 \gtrless \|q - s_2\|^2$$

$$\begin{aligned} f(s_1, s_2, q) &= \mathrm{sign}((x_q - x_1)^2 + (y_q - y_1)^2 - (x_q - x_2)^2 - (y_q - y_2)^2) \\ &= \mathrm{sign}(x_1^2 - 2x_1x_q - 2y_1y_q + y_1^2 - x_2^2 + 2x_2x_q + 2y_2y_q - y_2^2 \\ &= \mathrm{sign}(②) \end{aligned}$$

Analyzing Precision[LPT99]

How the degree relates to precision:

Consider multivariate poly $Q(x_1, \ldots, x_n)$ of deg k and s monomials (for simplicity, assume that coefficient of each monomial is 1).

Analyzing Precision[LPT99]

How the degree relates to precision:

Consider multivariate poly $Q(x_1, \ldots, x_n)$ of deg k and s monomials (for simplicity, assume that coefficient of each monomial is 1).

Let each x_i be an ℓ-bit integer $\implies x_i \in \{-2^\ell, \ldots, 2^\ell\}$.

Analyzing Precision[LPT99]

How the degree relates to precision:

Consider multivariate poly $Q(x_1, \ldots, x_n)$ of deg k and s monomials (for simplicity, assume that coefficient of each monomial is 1).
Let each x_i be an ℓ-bit integer $\implies x_i \in \{-2^\ell, \ldots, 2^\ell\}$.
Each monomial is in $\{-2^{\ell k}, \ldots, 2^{\ell k}\}$.

How the degree relates to precision:

Consider multivariate poly $Q(x_1, \ldots, x_n)$ of deg k and s monomials (for simplicity, assume that coefficient of each monomial is 1).
Let each x_i be an ℓ-bit integer $\implies x_i \in \{-2^{\ell}, \ldots, 2^{\ell}\}$.
Each monomial is in $\{-2^{\ell k}, \ldots, 2^{\ell k}\}$.
The value of $Q(x_1, \ldots, x_n)$ is in $\{-s2^{\ell k}, \ldots, s2^{\ell k}\}$.

Analyzing Precision[LPT99]

How the degree relates to precision:

Consider multivariate poly $Q(x_1, \ldots, x_n)$ of deg k and s monomials (for simplicity, assume that coefficient of each monomial is 1).

Let each x_i be an ℓ-bit integer $\implies x_i \in \{-2^\ell, \ldots, 2^\ell\}$.

Each monomial is in $\{-2^{\ell k}, \ldots, 2^{\ell k}\}$.

The value of $Q(x_1, \ldots, x_n)$ is in $\{-s2^{\ell k}, \ldots, s2^{\ell k}\}$.

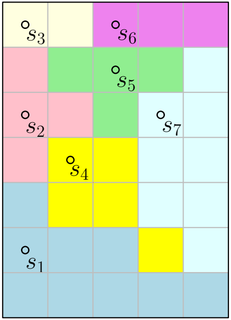
$\implies \ell k + \log(s) + O(1)$ bits are enough to evaluate Q.

Analyzing Precision[LPT99]

How the degree relates to precision:

Consider multivariate poly $Q(x_1, \ldots, x_n)$ of deg k and s monomials (for simplicity, assume that coefficient of each monomial is 1).
Let each x_i be an ℓ-bit integer $\implies x_i \in \{-2^\ell, \ldots, 2^\ell\}$.
Each monomial is in $\{-2^{\ell k}, \ldots, 2^{\ell k}\}$.
The value of $Q(x_1, \ldots, x_n)$ is in $\{-s2^{\ell k}, \ldots, s2^{\ell k}\}$.
$\implies \ell k + \log(s) + O(1)$ bits are enough to evaluate Q.

Note that ℓk bits is enough to evaluate the sign.



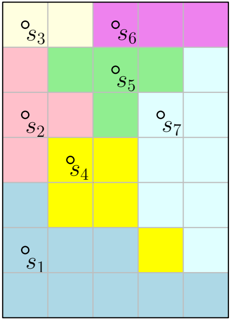
Given

A grid of size U and
Sites $S = \{s_1, \ldots, s_n\} \subset \mathbb{U}^2$

Label

Each grid point of $\mathbb{U}^2$ with the closest site of S

Nearest Neighbor Transform



Given
A grid of size U and
Sites $S = \{s_1, \ldots, s_n\} \subset \mathbb{U}^2$

Label
Each grid point of $\mathbb{U}^2$ with the closest site of S

	Alg	Time
Brute Force	deg 2	$O(nU^2)$
Nearest Neighbor Trans. [B90]	deg 5	$O(U^2)$
Discrete Voronoi diagram [C06, MQR03]	deg 3	$O(U^2)$
GPU Hardware [H99]	-	$\Theta(nU^2)$

Problem Transformations–Part 1

Problem (NNTrans-min)

For each pixel q, find the site with lowest index $s_i \in S$ minimizing $\|q - s_i\| < \|q - s_j\|$.

Problem Transformations–Part 1

Problem (NNTrans-min)

For each pixel q, find the site with lowest index $s_i \in S$ minimizing $\|q - s_i\| < \|q - s_j\|$.

$$\|q - s_i\|^2 < \|q - s_j\|^2$$
$$q \cdot q - 2q \cdot s_i + s_i \cdot s_i < q \cdot q - 2q \cdot s_j + s_j \cdot s_j$$
$$2x_i x_q + 2y_i y_q - x_i^2 - y_i^2 > 2x_j x_q + 2y_j y_q - x_j^2 - y_j^2.$$

Problem Transformations–Part 1

Problem (NNTrans-min)

For each pixel q, find the site with lowest index $s_i \in S$ minimizing $\|q - s_i\| < \|q - s_j\|$.

$$\|q - s_i\|^2 < \|q - s_j\|^2$$
$$q \cdot q - 2q \cdot s_i + s_i \cdot s_i < q \cdot q - 2q \cdot s_j + s_j \cdot s_j$$
$$2x_i x_q + 2y_i y_q - x_i^2 - y_i^2 > 2x_j x_q + 2y_j y_q - x_j^2 - y_j^2.$$

Problem (NNTrans-max)

For each pixel q, find the site with lowest index $s_i \in S$ maximizing $2x_i x_q + 2y_i y_q - x_i^2 - y_i^2$.

Problem Transformations–Part 2

Problem (NNTrans-max)

For each pixel q, find the site with lowest index $s_i \in S$ *maximizing* $2x_i x_q + 2y_i y_q - x_i^2 - y_i^2$.

Problem Transformations–Part 2

Problem (NNTrans-max)

For each pixel q, find the site with lowest index $s_i \in S$ maximizing $2x_i x_q + 2y_i y_q - x_i^2 - y_i^2$.

For a fixed row, $y_q = Y$

$$2x_i x_q + 2y_i y_q - x_i^2 - y_i^2 > 2x_j x_q + 2y_j y_q - x_j^2 - y_j^2$$

$$2x_i x_q + (2y_i Y - x_i^2 - y_i^2) > 2x_j x_q + (2y_j Y - x_j^2 - y_j^2)$$

$$①x_q + ② > ①x_q + ②$$

Problem Transformations–Part 2

Problem (NNTrans-max)

For each pixel q, find the site with lowest index $s_i \in S$ maximizing $2x_ix_q + 2y_iy_q - x_i^2 - y_i^2$.

For a fixed row, $y_q = Y$

$$2x_ix_q + 2y_iy_q - x_i^2 - y_i^2 > 2x_jx_q + 2y_jy_q - x_j^2 - y_j^2$$

$$2x_ix_q + (2y_iY - x_i^2 - y_i^2) > 2x_jx_q + (2y_jY - x_j^2 - y_j^2)$$

$$①x_q + ② > ①x_q + ②$$

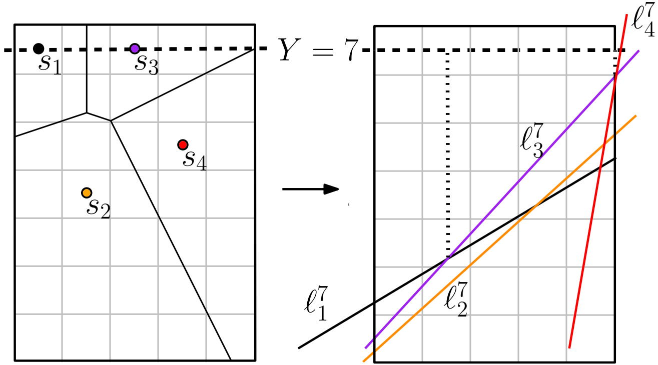
Problem (DUE-Y)

For a fixed $1 \leq Y \leq U$, and for each $1 \leq X \leq U$, find the line with lowest index $\ell_i^Y \in L_Y$ with maximum y-coordinate.

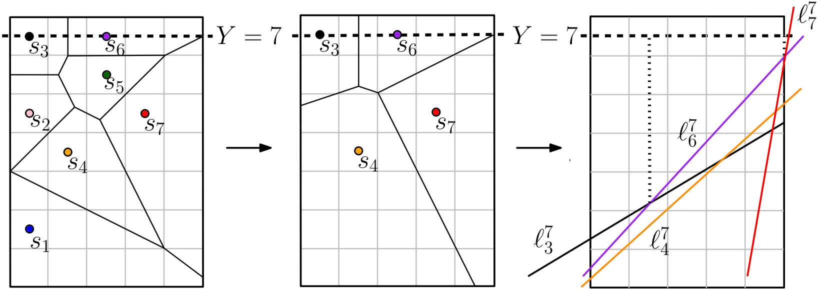
Problem Transformations–Part 2

Problem (DUE-Y)

For a fixed $1 \le Y \le U$, and for each $1 \le X \le U$, find the line with lowest index $\ell_i^Y \in L_Y$ with maximum y-coordinate.



Sketch of NNTransform Algorithm



Three Algorithms for Computing the DUE

Given m lines of the form $y = ①x + ②$

Discrete Upper Envelope construction

- `DUE-DEG3`: $O(m + U)$ time and degree 3
- `DUE-ULgU`: $O(m + U \log U)$ time and degree 2
- `DUE-U`:$O(m + U)$ expected time and degree 2

Three Algorithms for Computing the DUE

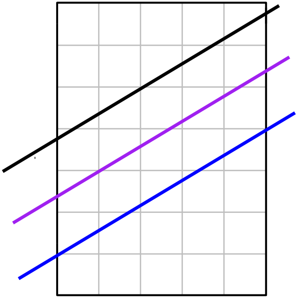
Given m lines of the form $y = \textcircled{1}x + \textcircled{2}$

Discrete Upper Envelope construction

- `DUE-DEG3`: $O(m + U)$ time and degree 3
- `DUE-ULgU`: $O(m + U \log U)$ time and degree 2
- `DUE-U`:$O(m + U)$ expected time and degree 2

For each algorithm:

1. Reduce to at most $O(U)$ lines.
2. Compute DUE of lines.



Three Algorithms for Computing the DUE

Given m lines of the form $y = ① x + ②$

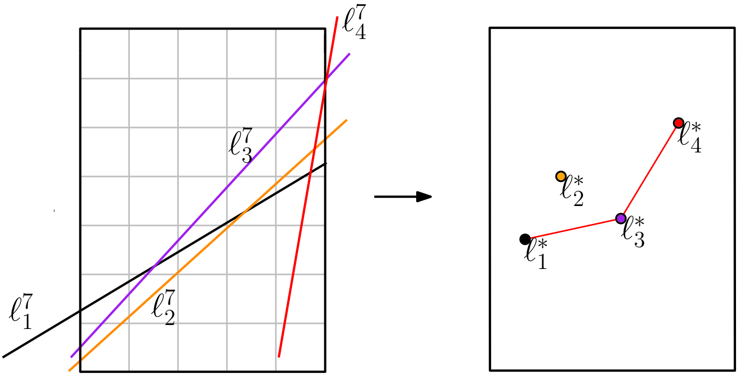
Discrete Upper Envelope construction

- `DUE-DEG3`: $O(m + U)$ time and degree 3
- `DUE-ULgU`: $O(m + U \log U)$ time and degree 2
- `DUE-U`: $O(m + U)$ expected time and degree 2

Discrete Upper Envelope Lemma (DUE−DEG3)

Two steps:

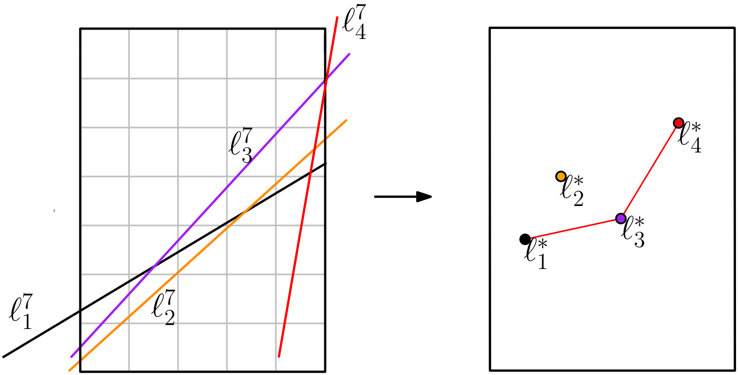
1. Compute upper env. via the lower hull of dual points
   line $y = mx + b$ maps to dual point $(m, -b)$.
2. Discretize upper env to DUE.



Discrete Upper Envelope Lemma (DUE−DEG3)

Two steps:

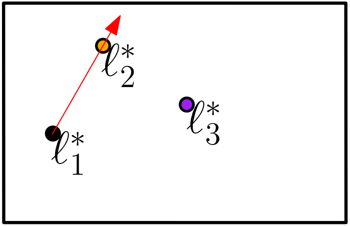
1. Compute upper env. via the lower hull of dual points
   line $y = mx + b$ maps to dual point $(m, -b)$.
2. Discretize upper env to DUE.



line form is $y = ①x + ②$ $\implies$ dual point form is $(①, ②)$

Precision of DUE-DEG3

Orientation test



ℓ_1^*, ℓ_2^* and ℓ_3^* have form (①, ②)

$$\texttt{orient}(\ell_1^*, \ell_2^*, \ell_3^*) = \operatorname{sign}\left(\begin{vmatrix} ⓪ & ① & ② \\ ⓪ & ① & ② \\ ⓪ & ① & ② \end{vmatrix}\right) = \operatorname{sign}(③)$$

Three Algorithms for Computing the DUE

Given m lines of the form $y = ①x + ②$

Discrete Upper Envelope construction

- `DUE-DEG3`: $O(m + U)$ time and degree 3
- `DUE-ULgU`: $O(m + U \log U)$ time and degree 2
- `DUE-U`: $O(m + U)$ expected time and degree 2

Three Algorithms for Computing the DUE

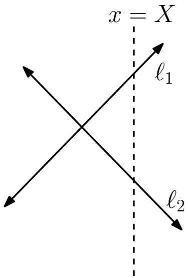
Given *m* lines of the form $y = ①x + ②$

Discrete Upper Envelope construction

- `DUE-DEG3`: $O(m + U)$ time and degree 3
- `DUE-ULgU`: $O(m + U \log U)$ time and degree 2
- `DUE-U`:$O(m + U)$ expected time and degree 2

Main predicate `OrderOnALine`

Order on a Line

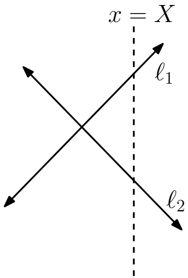


$X = ①$
ℓ_1, ℓ_2 have form $y = ①x + ②$

`orderOnLine`$(\ell_1, \ell_2, X) = \mathrm{sign}(①X + ② - ①X + ②) = \mathrm{sign}(②)$

Main predicate `OrderOnALine`

Order on a Line



$X = ①$

ℓ_1, ℓ_2 have form $y = ①x + ②$

$$\texttt{orderOnLine}(\ell_1, \ell_2, X) = \text{sign}(①X + ② - ①X + ②) = \text{sign}(②)$$

Lemma `IntersectCol`

Given two lines ℓ_1 and ℓ_2 of the form $y = ①x + ②$, construction `IntersectCol`(ℓ_1, ℓ_2) returns the column containing $\ell_1 \cap \ell_2$ in $O(logU)$ time and degree 2.

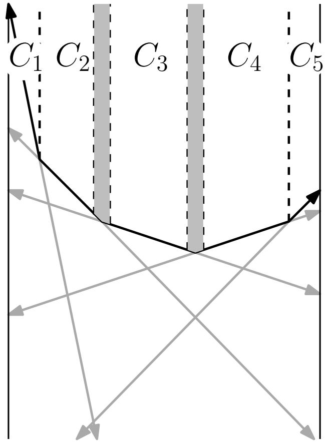
Three Algorithms for Computing the DUE

Given m lines of the form $y = ①x + ②$

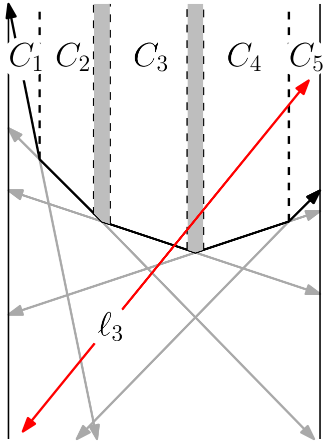
Discrete Upper Envelope construction

- `DUE-DEG3`: $O(m + U)$ time and degree 3
- `DUE-ULgU`: $O(m + U \log U)$ time and degree 2
- `DUE-U`: $O(m + U)$ expected time and degree 2

Discrete Upper Envelope Lemma (DUE−ULgU)



Discrete Upper Envelope Lemma (DUE−ULgU)



Three Algorithms for Computing the DUE

Given m lines of the form $y = ①x + ②$

Discrete Upper Envelope construction

- `DUE-DEG3`: $O(m + U)$ time and degree 3
- `DUE-ULgU`: $O(m + U \log U)$ time and degree 2
- `DUE-U`: $O(m + U)$ expected time and degree 2

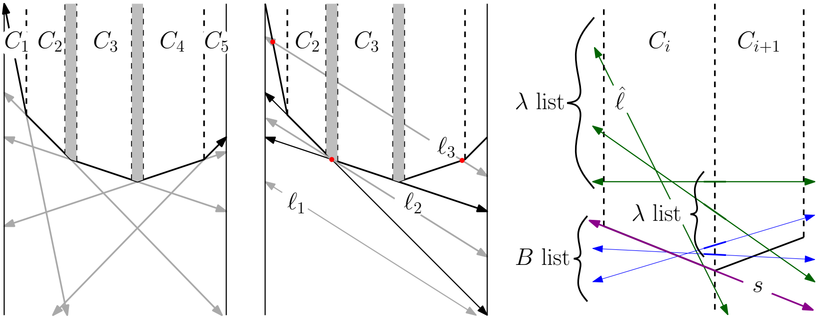
Three Algorithms for Computing the DUE

Given m lines of the form $y = ①x + ②$

Discrete Upper Envelope construction

- `DUE-DEG3`: $O(m + U)$ time and degree 3
- `DUE-ULgU`: $O(m + U \log U)$ time and degree 2
- `DUE-U`: $O(m + U)$ expected time and degree 2

Discrete Upper Envelope Lemma (DUE−U)



Three Algorithms for Computing the DUE

Given m lines of the form $y = ①x + ②$

Discrete Upper Envelope construction

- `DUE-DEG3`: $O(m + U)$ time and degree 3
- `DUE-ULgU`: $O(m + U \log U)$ time and degree 2
- `DUE-U`: $O(m + U)$ expected time and degree 2

Three Algs for Computing the NNTransform

Given n sites from $\mathbb{U}$

Nearest Neighbor Transform construction

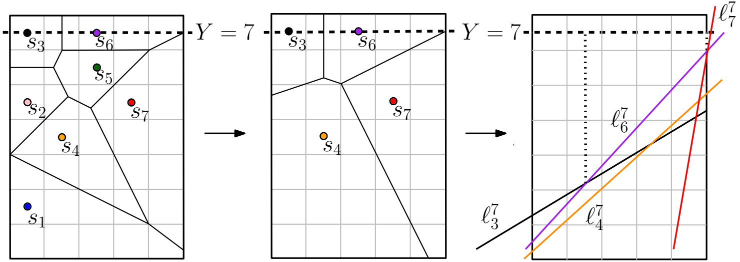
- `Deg3`: $O(U^2)$ time and degree 3
- `UsqLgU`: $O(U^2 \log U)$ time and degree 2
- `Usq`: $O(U^2)$ expected time and degree 2

Three Algs for Computing the NNTransform

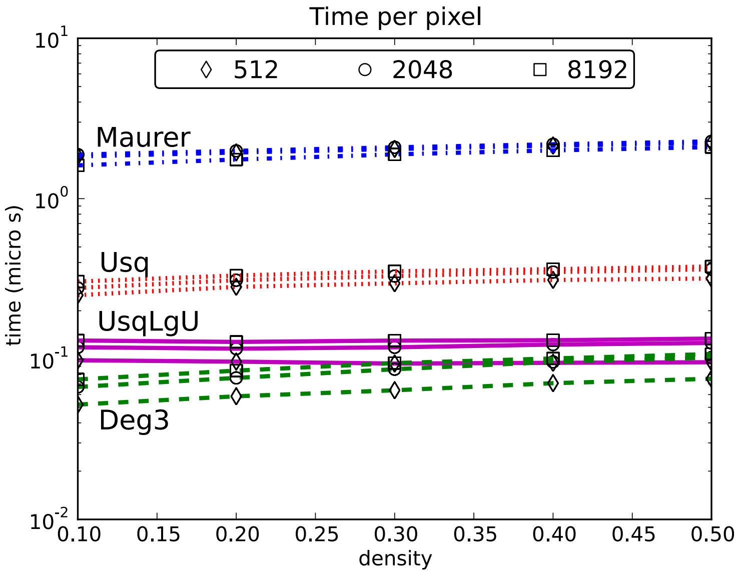
Given n sites from $\mathbb{U}$

Nearest Neighbor Transform construction

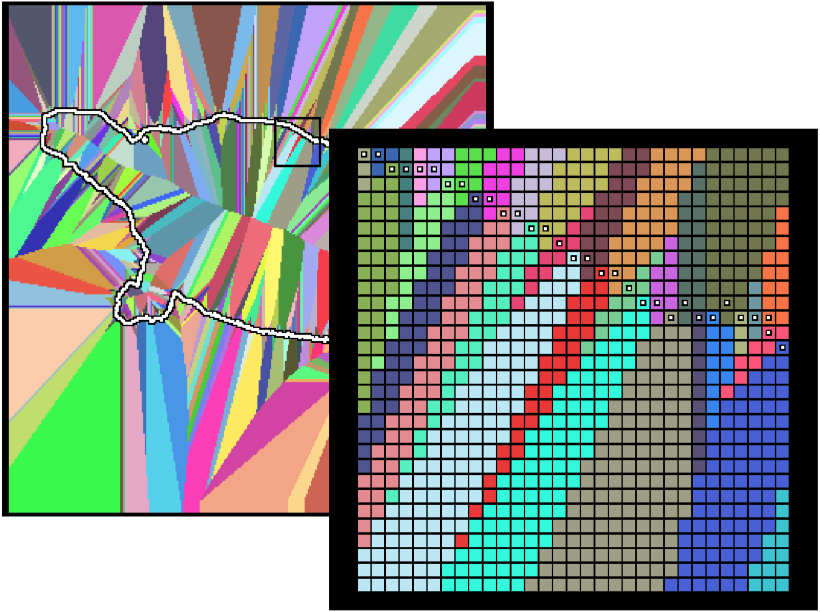
- `Deg3`: $O(U^2)$ time and degree 3
- `UsqLgU`: $O(U^2 \log U)$ time and degree 2
- `Usq`: $O(U^2)$ expected time and degree 2



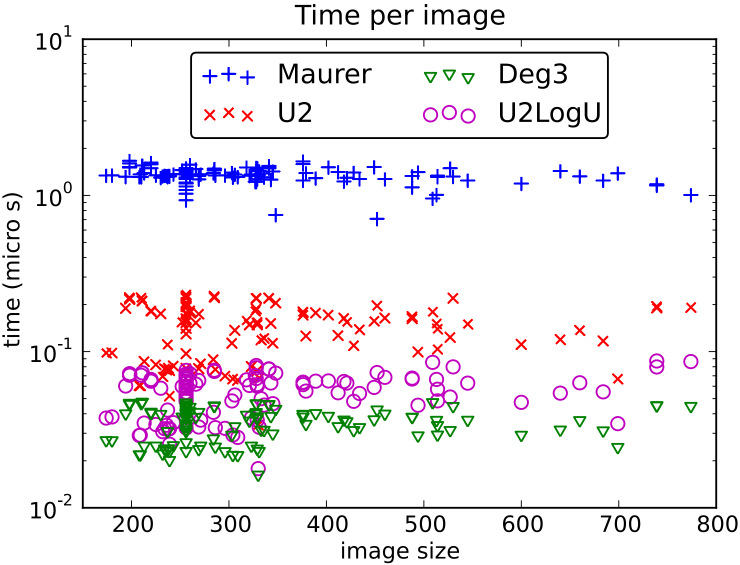
Experiments Part 1



Experiments Part 2



Experiments Part 2



Summary

Described and implemented three algorithms
for computing the DUE of lines of the form $y = ①x + ②$:

- `DUE-DEG3`: $O(n + U)$ and degree 3
- `DUE-ULgU`: $O(n + U \log U)$ and degree 2
- `DUE-U`: $O(n + U)$ expected time and degree 2

Which gave us three algorithms for computing the NNTransform:

- `Deg3`: $O(U^2)$ and degree 3
- `UsqLgU`: $O(U^2 \log U)$ and degree 2
- `Usq`: $O(U^2)$ expected time and degree 2.

Conclusions

Can we compute the NNTraform with degree 2 without randomization?

What about L_1 or L_∞?

What other geometric problems can be considered using degree-driven algorithm design?

Summary

Described and implemented three algorithms
for computing the DUE of lines of the form $y = ① x + ②$:

- `DUE-DEG3`: $O(n + U)$ and degree 3
- `DUE-ULgU`: $O(n + U \log U)$ and degree 2
- `DUE-U`: $O(n + U)$ expected time and degree 2

Which gave us three algorithms for computing the NNTransform:

- `Deg3`: $O(U^2)$ and degree 3
- `UsqLgU`: $O(U^2 \log U)$ and degree 2
- `Usq`: $O(U^2)$ expected time and degree 2.

Contact

Presenter: David L. Millman email: dave@cs.unc.edu
web: `http://cs.unc.edu/~dave`