Computing the Nearest Neighbor Transform Exactly with only Double Precision

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

20 April 2013
Given $n$ sites on a pixel grid, what is the closest site to each pixel?
Given $n$ sites on a pixel grid, what is the closest site to each pixel?
Given $n$ sites on a pixel grid, what is the closest site to each pixel?

How much precision is needed to determine this?
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]
Is $q$ closer to $s_1$?

Let $\mathbb{U} = \{1, 2, \ldots, U\}$, $s_1, s_2, q \in \mathbb{U}^2$.

Then:

- $s_1 = (x_1, y_1)$
- $s_2 = (x_2, y_2)$
- $q = (x_q, y_q)$

We have:

$$\|q - s_1\|^2 \geq \|q - s_2\|^2$$
Analyzing Precision [LPT99]

Is $q$ closer to $s_1$?

\[ \mathbb{U} = \{1, 2, \ldots, U\} \]

$s_1, s_2, q \in \mathbb{U}^2$

$s_1 = (x_1, y_1)$

$s_2 = (x_2, y_2)$

$q = (x_q, y_q)$

\[ \|q - s_1\|^2 \geq \|q - s_2\|^2 \]

\[ f(s_1, s_2, q) = \text{sign}((x_q - x_1)^2 + (y_q - y_1)^2 - (x_q - x_2)^2 - (y_q - y_2)^2) \]
Is \( q \) closer to \( s_1 \)?

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

\[
\| q - s_1 \|^2 \geq \| q - s_2 \|^2
\]

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

\[
= \text{sign}(x_1^2 - 2x_1 x_q - 2y_1 y_q + y_1^2 - x_2^2 + 2x_2 x_q + 2y_2 y_q - y_2^2)
\]
Analyzing Precision [LPT99]

Is \( q \) closer to \( s_1 \)?

\[
\mathbb{U} = \{1, 2, \ldots, U\}
\]

\( s_1, s_2, q \in \mathbb{U}^2 \)

\( s_1 = (x_1, y_1) \)

\( s_2 = (x_2, y_2) \)

\( q = (x_q, y_q) \)

\[\|q - s_1\|^2 \geq \|q - s_2\|^2\]

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

\[= \text{sign}(x_1^2 - 2x_1 x_q - 2y_1 y_q + y_1^2 - x_2^2 + 2x_2 x_q + 2y_2 y_q - y_2^2)\]

\[= \text{sign}(\circled{2})\]
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).
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 \( \ell \)-bit integer \( \Rightarrow x_i \in \{-2^\ell, \ldots, 2^\ell\} \).
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 $\ell$-bit integer $\implies x_i \in \{-2^\ell, \ldots, 2^\ell\}$. Each monomial is in $\{-2^{\ell k}, \ldots, 2^{\ell k}\}$. 

\[
\|q - s_1\|_2 \succ \|q - s_2\|_2
\]

\[
f(s_1, s_2, q) = \text{sign}
\]

\[
= \text{sign} \left( (2^{\ell k}) \right)
\]
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 $\ell$-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 $\{-s2^{\ell k}, \ldots, s2^{\ell k}\}$. 
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 $\ell$-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 $\ell$-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 $\ell k$ bits is enough to evaluate the sign.
Given
A grid of size $U$ and Sites $S = \{s_1, \ldots, s_n\} \subset U^2$

Label
Each grid point of $U^2$ with the closest site of $S$
Given
A grid of size $U$ and Sites $S = \{s_1, \ldots, s_n\} \subset U^2$

Label
Each grid point of $U^2$ with the closest site of $S$

<table>
<thead>
<tr>
<th>Alg</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brute Force</td>
<td>$O(nU^2)$</td>
</tr>
<tr>
<td>Nearest Neighbor Trans.</td>
<td>$O(U^2)$</td>
</tr>
<tr>
<td>[B90]</td>
<td>$O(U^2)$</td>
</tr>
<tr>
<td>Discrete Voronoi diagram</td>
<td>$O(U^2)$</td>
</tr>
<tr>
<td>[C06, MQR03]</td>
<td>$O(U^2)$</td>
</tr>
<tr>
<td>GPU Hardware [H99]</td>
<td>$\Theta(nU^2)$</td>
</tr>
</tbody>
</table>
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 (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-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 (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, \( y_q = Y \)

\[
2x_i x_q + 2y_i y_q - x_i^2 - y_i^2 > 2x_j x_q + 2y_j y_q - x_j^2 - y_j^2
\]
\[
2x_i x_q + (2y_i Y - x_i^2 - y_i^2) > 2x_j x_q + (2y_j Y - x_j^2 - y_j^2)
\]
\[
\mathbf{1} x_q + \mathbf{2} > \mathbf{1} x_q + \mathbf{2}
\]
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, \( y_q = Y \)
\[
2x_i x_q + 2y_i y_q - x_i^2 - y_i^2 > 2x_j x_q + 2y_j y_q - x_j^2 - y_j^2
\]
\[
2x_i x_q + (2y_i Y - x_i^2 - y_i^2) > 2x_j x_q + (2y_j Y - x_j^2 - y_j^2)
\]
\[
(1) x_q + (2) > (1) x_q + (2)
\]

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 \( \ell^Y_i \in L_Y \) with maximum y-coordinate.
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.
Sketch of NNTransform Algorithm

\[ Y = 7 \]

\[ \ell_7 \parallel \ell_6 \]

\[ \ell_3 \parallel \ell_4 \]

\[ \ell_7 \parallel \ell_6 \]

\[ \ell_3 \parallel \ell_4 \]
Three Algorithms for Computing the DUE

**Given** $m$ lines of the form $y = \circled1 x + \circled2$

### 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 $y = \text{①} x + \text{②}$

<table>
<thead>
<tr>
<th>Discrete Upper Envelope construction</th>
<th>$\text{DUE-DEG3: } O(m + U)$ time and degree 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{DUE-ULgU: } O(m + U \log U)$ time and degree 2</td>
<td></td>
</tr>
<tr>
<td>$\text{DUE-U: } O(m + U)$ expected time and degree 2</td>
<td></td>
</tr>
</tbody>
</table>

For each algorithm:

1. Reduce to at most $O(U)$ lines.
2. Compute DUE of lines.
Given $m$ lines of the form $y = \cdot x + \cdot$

**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.  

\[
\ell_1^7, \ell_2^7, \ell_3^7, \ell_4^7
\]

\[
\ell_1^*, \ell_2^*, \ell_3^*, \ell_4^*
\]
Discrete Upper Envelope Lemma \((\text{DUE-DEG3})\)

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 \(y = \underbrace{1}_{1} x + \underbrace{2}_{2}\) \(\implies\) dual point form is \((\underbrace{1}_{1}, \underbrace{2}_{2})\)
Precision of \textsc{Due-DEG3}

Orientation test

\[ \ell^*_1, \ell^*_2 \text{ and } \ell^*_3 \text{ have form } (1, 2) \]

\[
\text{orient}(\ell^*_1, \ell^*_2, \ell^*_3) = \text{sign} \left( \begin{array}{ccc}
0 & 1 & 2 \\
0 & 1 & 2 \\
0 & 1 & 2 \\
\end{array} \right) = \text{sign}(3)
\]
Three Algorithms for Computing the DUE

**Given** \( m \) lines of the form \( y = \begin{pmatrix} 1 \end{pmatrix} x + \begin{pmatrix} 2 \end{pmatrix} \)

**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
Given $m$ lines of the form $y = \mathcal{A}x + \mathcal{B}$

<table>
<thead>
<tr>
<th>Discrete Upper Envelope construction</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DUE-DEG3</strong>: $O(m + U)$ time and degree 3</td>
</tr>
<tr>
<td><strong>DUE-ULgU</strong>: $O(m + U \log U)$ time and degree 2</td>
</tr>
<tr>
<td><strong>DUE-U</strong>: $O(m + U)$ expected time and degree 2</td>
</tr>
</tbody>
</table>
**Main predicate** \textbf{OrderOnALine}

Order on a Line

\[ x = X \]

\[ \ell_1, \ell_2 \text{ have form } y = 1x + 2 \]

\[ X = 1 \]

orderOnLine(\(\ell_1, \ell_2, X\)) = \text{sign}(1X + 2 - 1X + 2) = \text{sign}(2) \]
Main predicate **OrderOnALine**

Order on a Line

\[ x = X \]

\[ \ell_1 \]

\[ \ell_2 \]

\[ X = \begin{pmatrix} 1 \\ 2 \end{pmatrix} \]

\[ \ell_1, \ell_2 \text{ have form } y = \begin{pmatrix} 1 \\ 2 \end{pmatrix} x + \begin{pmatrix} 2 \end{pmatrix} \]

\[ \text{orderOnLine}(\ell_1, \ell_2, X) = \text{sign}(\begin{pmatrix} 1 \\ 2 \end{pmatrix} X + \begin{pmatrix} 2 \end{pmatrix} - \begin{pmatrix} 1 \\ 2 \end{pmatrix} X + \begin{pmatrix} 2 \end{pmatrix}) = \text{sign}(\begin{pmatrix} 2 \end{pmatrix}) \]

**Lemma** **IntersectCol**

Given two lines \( \ell_1 \) and \( \ell_2 \) of the form \( y = \begin{pmatrix} 1 \\ 2 \end{pmatrix} x + \begin{pmatrix} 2 \end{pmatrix} \), construction \( \text{IntersectCol}(\ell_1, \ell_2) \) returns the column containing \( \ell_1 \cap \ell_2 \) in \( O(\log U) \) time and degree 2.
Three Algorithms for Computing the DUE

Given $m$ lines of the form $y = a + 2$

<table>
<thead>
<tr>
<th>Discrete Upper Envelope construction</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DUE-DEG3</strong>: $O(m + U)$ time and degree 3</td>
</tr>
<tr>
<td><strong>DUE-ULgU</strong>: $O(m + U \log U)$ time and degree 2</td>
</tr>
<tr>
<td><strong>DUE-U</strong>: $O(m + U)$ expected time and degree 2</td>
</tr>
</tbody>
</table>
Discrete Upper Envelope Lemma (DUE-ULgU)
Discrete Upper Envelope Lemma (DUE–ULgU)
Three Algorithms for Computing the DUE

Given $m$ lines of the form $y = \theta x + \omega$

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 $y = 1 \cdot x + 2$

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Time Complexity</th>
<th>Degree</th>
</tr>
</thead>
<tbody>
<tr>
<td>DUE–DEG3</td>
<td>$O(m + U)$</td>
<td>3</td>
</tr>
<tr>
<td>DUE–ULgU</td>
<td>$O(m + U \log U)$</td>
<td>2</td>
</tr>
<tr>
<td>DUE–U</td>
<td>$O(m + U)$</td>
<td>2</td>
</tr>
</tbody>
</table>

For each algorithm:
1. Reduce to at most $O(U)$ lines.
2. Compute DUE of lines.
Discrete Upper Envelope Lemma (DUE-U)
Three Algorithms for Computing the DUE

Given $m$ lines of the form $y = \theta_1 x + \theta_2$

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 \( \mathbb{U} \)

<table>
<thead>
<tr>
<th>Nearest Neighbor Transform construction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deg3: ( O(U^2) ) time and degree 3</td>
</tr>
<tr>
<td>USqLgU: ( O(U^2 \log U) ) time and degree 2</td>
</tr>
<tr>
<td>USq: ( O(U^2) ) expected time and degree 2</td>
</tr>
</tbody>
</table>
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

\[ s_3 s_4 s_7 s_6 \]
\[ s_5 \]
\[ s_1 \]

\[ Y = 7 \]

\[ s_3 \]
\[ s_4 \]
\[ s_7 \]
\[ s_6 \]

\[ Y = 7 \]

\[ s_3 \]
\[ s_4 \]
\[ s_7 \]
\[ s_6 \]

\[ Y = 7 \]
Experiments Part 1

Time per pixel

- **Maurer**
- **Usq**
- **UsqLgU**
- **Deg3**

Time (micro s)

Density

- 512
- 2048
- 8192

MILLMAN, Love, Chan, Snoeyink

Double Precision Computation of the NNTransform
Described and implemented three algorithms for computing the DUE of lines of the form $y = \frac{1}{2}x + \frac{2}{3}$:

- **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.
Can we compute the NNTraform with degree 2 without randomization?

What about $L_1$ or $L_\infty$?

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

Contact

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