Geometric Tomography
with
Strip-based Model, Quad-trees and Greedy Algorithms

Yan Gerard
Groupe de Travail de Géométrie Discrète, June 13th 2013, Paris
Geometric Tomography with
Strip-based Model, Quad-trees and Greedy Algorithms
The different materials absorb differently the X-rays which cross it.

The intensity of the X-ray measured behind the object provides an information about the lengths of the different materials which have been crossed by the X-ray.
The intensity of the X-ray measured behind the object provides an information about the lengths of the different materials which have been crossed by the X-ray.

\[ I_{\text{measured}} = I_0 \int_{\text{path}} \exp(-f(x,y)) \, dl \]

\( f(x,y) \) is the absorbance coefficient.

It characterizes the material at point \((x,y)\).

The image of \( f(x,y) \) is interesting for many applications (medicine, non destructive control, geology…).
Geometric Tomography
with
Strip-based Model, Quad-trees and Greedy Algorithms
Computerized Tomography deals with the reconstruction of a continuous function on a continuous domain:
\[ f : [0, 1]^2 \rightarrow [0, 1] \]

Geometric Tomography deals with the reconstruction of a binary function on a continuous domain:
\[ f : [0, 1]^2 \rightarrow \{0, 1\} \]

Discrete Tomography deals with the reconstruction of a binary function on a discrete domain:
\[ f : Lattice \rightarrow \{0, 1\} \]

namely a lattice set.
Geometric Tomography
with
Strip-based Model, Quad-trees and Greedy Algorithms
a shape $S$

the projection of $S$ according to the length ray model

according to the strip-based model
**Input:** a finite set of directions for each direction, a projection

**Output:** A subset of $[0,1]^2$ with the prescribed projections
Geometric Tomography with
Strip-based Model, Quad-trees and Greedy Algorithms
A quad-tree

shapes are represented by unions of cells (quads) of different sizes
Geometric Tomography
with
Strip-based Model, Quad-trees and Greedy Algorithms
Geometric Tomography with strip-based model, Quadtrees and Greedy Algorithms
Initialization:
Take an empty current shape
## Initialization:

Take an empty current shape  
Compute error table

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

- **prescribed projections**: 2 3 2 1  
- **current projections**: 0 0 0 0  
- **error on projections**: 2 3 2 1
Initialization:
Take an empty current shape
Compute error table

Loop:
Find maximum error
Add point and update error table

End:
No more positive error in one direction

Geometric Tomography with strip-based model, Quadtrees and Greedy Algorithms
Plan

I Introduction

II Linear Programming

III Error Density & Greedy Algorithms

IV Heuristic

V Experiments
An instance
An instance

The strips define polytopes
The strips define polytopes

We can compute the area of each polytope: \(\text{Area}(P_i)\)

Let \(a_i\) be the area of the intersection of a solution with the polytope \(P_i\)

For each polytope \(P_i\), \(0 \leq a_i \leq \text{area}(P_i)\)
Let \( a_i \) be the area of the intersection of a solution with the polytope \( P_i \).

For each polytope \( P_i \), \[ 0 \leq a_i \leq \text{area}(P_i) \]

For each strip \( S_i \), we know \[ \sum_{i / P_i \text{ in strip } S_j} a_i = \text{projection}(\text{strip } S_j) \]
For each polytope $P_i$, $0 \leq a_i \leq \text{area}(P_i)$

For each strip $S_j$, we know

$$\sum_{i \in P \text{ in strip } S_j} a_i = \text{projection}(\text{strip } S_j)$$

**Linear Programming**

For each strip $S_j$,

$$\text{proj}(\text{strip } S_j) - h_j \leq \sum_{i \in P \text{ in strip } S_j} a_i \leq \text{proj}(\text{strip } S_j) + h_j$$

Minimize

$$\sum_j h_j$$
For each polytope $P_i$, \[ 0 \leq a_i \leq \text{area}(P_i) \]

For each strip $S_j$, \[ \text{proj}(\text{strip } S_j) - h_j \leq \sum_{i \in \text{strip } S_j} a_i \leq \text{proj}(\text{strip } S_j) + h_j \]

Minimize \[ \sum_j h_j \]

But how many variables? \[ 2 \times \text{number of directions} + \text{number of polytopes} \]

How many polytopes?
How many polytopes?

With 2 directions

\[ \text{number of captors}^2 \]
How many polytopes?

With 3 directions

With an even number of captors \((2n)\): \(6n^2\)

With an odd number of captors \((2n+1)\): \(6n^2+6n+1\)
How many polytopes?

With 16 directions and 4 captors

576 polytopes
With 16 directions and 8 captors

How many polytopes?
How many polytopes?

With 16 directions and 8 captors
## How many polytopes?

<table>
<thead>
<tr>
<th></th>
<th>4 captors</th>
<th>8 captors</th>
<th>(n) captors</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 directions</td>
<td>16</td>
<td>64</td>
<td>(n^2)</td>
</tr>
<tr>
<td>3 directions</td>
<td>24</td>
<td>96</td>
<td>(\frac{3n^2}{2}) or (\frac{3n^2-1}{2})</td>
</tr>
<tr>
<td>4 directions</td>
<td>48</td>
<td>232</td>
<td>?</td>
</tr>
<tr>
<td>8 directions</td>
<td>160</td>
<td>800</td>
<td>?</td>
</tr>
<tr>
<td>16 directions</td>
<td>576</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>(k) directions</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

It can probably make a lot of polytopes

A lot of variables for LP-solvers
At the end, the user has to choose where he puts the points in each polytope.
make something \textit{completely different} with quad-trees, greedy algorithms, and simulated annealing
Experiments

Heuristic

Back to Radon transform

Linear Programming

Error Density & Greedy Algorithms

Experiments

Heuristic

Error Density & Greedy Algorithms

Linear Programming

Back to Radon transform

Experiments

Heuristic

Linear Programming

Error Density & Greedy Algorithms

Back to Radon transform

Experiments

Heuristic

Linear Programming

Error Density & Greedy Algorithms

Back to Radon transform

Experiments

Heuristic

Linear Programming

Error Density & Greedy Algorithms

Back to Radon transform
Error Density

Principle of greedy algorithm

Initialization:
Take an empty current shape
Compute error table

Loop:
Find maximum error
Add point and update error table

It uses a local criterion of error

prescribed projections 2 3 2 1
current projections 2 3 2 1
error on projections 0 0 0 0
It uses a local criterion of error

An instance

A partition of the domain

A set of cells

Its projections

The error on each projection

What is the contribution of a cell to this error?
What is the contribution of a cell to this error?

<table>
<thead>
<tr>
<th>Projections of the cell</th>
<th>Errors on projections</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>+ 0.019820</td>
</tr>
<tr>
<td>0.0</td>
<td>- 0.052203</td>
</tr>
<tr>
<td>0.0</td>
<td>+ 0.045781</td>
</tr>
<tr>
<td>0.001231</td>
<td>- 0.010175</td>
</tr>
<tr>
<td>0.002307</td>
<td>- 0.001265</td>
</tr>
<tr>
<td>0.000843</td>
<td>+ 0.021479</td>
</tr>
<tr>
<td>0.000912</td>
<td>- 0.012728</td>
</tr>
<tr>
<td>0.000005</td>
<td>+ 0.022521</td>
</tr>
<tr>
<td>0.0</td>
<td>- 0.030404</td>
</tr>
<tr>
<td>0.0</td>
<td>- 0.025168</td>
</tr>
<tr>
<td>Sum for all directions</td>
<td>0.0</td>
</tr>
</tbody>
</table>

We compute for each cell: error sum
We compute for each cell its error density

But small cells are penalized: their error sum is small.

Then we introduce for each cell its error density

We use the error density to drive greedy algorithms.

If its negative, there is a deficit of points at the cell.

If its positive, there is an excess of points at the cell.
Greedy Algorithm

We use the error density to drive greedy algorithms

| If its negative, there is a deficit of points at the cell |
| If its positive, there is an excess of points at the cell |

Greedy Algorithm:

**Initialization:**
- Take a decomposition of the domain in cells
- Take an empty current set of cells
- Compute error density table

**Loop:**
- Find minimal (negative) error density
- Add cell and update error table

**End:**
- No more negative error density
**Initialization:**
Take a decomposition of the domain in cells
Take an empty current set of cells
Compute error density table

**Loop:**
Find minimal (negative) error density
Add cell and update error table

**End:**
No more negative error density

The cell structure is not modified!
May be, we can improve it?
Greedy Algorithm

The cell structure is not modified!
May be, we can improve it

**Initialization:**
Take a decomposition in cells

**Loop:**
Perform Greedy Algorithm
Refine decomposition
Greedy Algorithm

The cell structure is not modified!
May be, we can improve it

**Initialization:**
Take a decomposition in cells

**Loop:**
Perform Greedy Algorithm
Refine decomposition
Greedy Algorithm

The cell structure is not modified!
May be, we can improve it

Initialization:
Take a decomposition in cells

Loop:
Perform Greedy Algorithm
Refine decomposition
Greedy Algorithm

The cell structure is not modified!
May be, we can improve it

Initialization:
Take a decomposition in cells

Loop:
Perform Greedy Algorithm
Refine decomposition
Greedy Algorithm

The cell structure is not modified!
May be, we can improve it

Initialization:
Take a decomposition in cells

Loop:
Perform Greedy Algorithm
Refine decomposition
The cell structure is not modified!
May be, we can improve it

**Initialization:**
Take a decomposition in cells

**Loop:**
Perform Greedy Algorithm
Refine decomposition

Multi-resolution Greedy algorithm: we alternate routine greedy algorithm with cell decomposition
computation

cell structure

Result of greedy algorithm

Cell decomposition
Heuristic

Experiments

Energy Minimization

Linear Programming

Back to Radon transform
In this section, we investigate a heuristic (close to Simulated Annealing)

The algorithm is controlled by a temperature which decreases during the process

At each place, several things can happen:

- a cell can split

- cells with the same color can merge

- the color of a cell may change
- a cell can split
- cells with the same color can merge
- the color of a cell may change
- a cell can split

- cells with the same color can merge

- the color of a cell may change

The probability to change the color of a cell depends on
- the temperature
- the error density
We compute for each cell:

### Projections of the cell

<table>
<thead>
<tr>
<th>Projection</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>0.001231</td>
<td></td>
</tr>
<tr>
<td>0.002307</td>
<td></td>
</tr>
<tr>
<td>0.000843</td>
<td></td>
</tr>
<tr>
<td>0.000912</td>
<td></td>
</tr>
<tr>
<td>0.000005</td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td></td>
</tr>
</tbody>
</table>

### Errors on projections

<table>
<thead>
<tr>
<th>Error</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>0.019820</td>
</tr>
<tr>
<td>-</td>
<td>0.052203</td>
</tr>
<tr>
<td>+</td>
<td>0.045781</td>
</tr>
<tr>
<td>-</td>
<td>0.010175</td>
</tr>
<tr>
<td>-</td>
<td>0.001265</td>
</tr>
<tr>
<td>+</td>
<td>0.021479</td>
</tr>
<tr>
<td>-</td>
<td>0.012728</td>
</tr>
<tr>
<td>+</td>
<td>0.022521</td>
</tr>
<tr>
<td>-</td>
<td>0.030404</td>
</tr>
<tr>
<td>-</td>
<td>0.025168</td>
</tr>
</tbody>
</table>

### Sum for all directions

- Error sum: 0.0
- Absolute error sum: 0.0
We compute for each cell

\[ \text{error sum} \]

\[ \text{absolute error sum} \]

We compute their ratio

Its value is between \(-1\) and \(1\)

Ratio \(-1\) means **deficit** of points
Ratio \(+1\) means **excess** of points
Ratio \(~0\) means we don’t know

The probability to change the color of a cell depends on
- the temperature
- the **error density**
- the error ratio
- a cell can split
- cells with the same color can merge
- the color of a cell may change

The probability to split depends also on:
- the temperature
- the error ratio \( \text{error sum} \over \text{absolute error sum} \)
- a cell can split

- cells with the same color can merge

- the color of a cell may change

The probability to merge depends on
- *the temperature*
1. Back to Radon transform
2. Linear Programming
3. Energy Minimization
4. Heuristic
5. Experiments
Experiments

Phantom

Greedy Algorithm

Multiresolution GA

Simulated Annealing
Experiments

GA = Greedy Algorithm

GAME = Multiresolution GA

MPH = Multiresolution Probabilistic Heuristic
Experiments

Phantom

Greedy Algorithm

Multiresolution GA

Simulated Annealing
Experiments

GA = Greedy Algorithm
GAME = Multiresolution GA
MPH = Multiresolution Probalistic Heuristic
Thank you for your attention