Isothetic Covers for Digital Objects:
Algorithms and Applications

Partha Bhowmick

CSE, IIT Kharagpur

RESEARCH PROMOTION WORKSHOP
INTRODUCTION TO GRAPH AND GEOMETRIC ALGORITHMS
NOVEMBER 1–3, 2011 (PDPM IIITDM JABALPUR)
Object and Isothetic Cover

Isothetic Cover
P. Bhowmick

Introduction
Naive Combinatorial Applications
Hull Shape 3D

image
Object and Isothetic Cover

object = set of 1s
Object and Isothetic Cover

object = set of 1s
Object and Isothetic Cover

object = set of 1s
Object and Isothetic Cover

$g = 4$: Isothetic Cover
$g = 6$: Isothetic Cover
Object and Isothetic Cover

\[ g = 8: \text{Isothetic Cover} \]
Object and Isothetic Cover

$g = 10$: Isothetic Cover
Definitions

Digital plane, $\mathbb{Z}^2 = \text{set of all points having integer coordinates}$.
Definitions

Digital point (pixel) = a point in $\mathbb{Z}^2$. 
Digital object = a set $S$ of digital points.
Definitions

4-neighborhood of $p$:

$$N_4(p) = \{(x', y') : (x', y') \in \mathbb{Z}^2 \land |x - x'| + |y - y'| = 1\}$$
8-neighborhood of $p$:

$$N_8(p) = \{(x', y') : (x', y') \in \mathbb{Z}^2 \land \max(|x - x'|, |y - y'|) = 1\}$$
Two points \( p \) and \( q \) are \( k \)-connected in \( S \) if there exists a sequence \( \langle p := p_0, p_1, \ldots, p_n := q \rangle \subseteq S \) such that \( p_i \in N_k(p_{i-1}) \) for \( 1 \leq i \leq n \).
Definitions

For any point \( p \in S \), the maximum-cardinality set of points that are \( k \)-connected to \( p \) forms a \( k \)-connected component of \( S \).
Grid $\mathcal{G}$ with grid size $g = 1$ (red dashed lines)
Definitions

Grid $\mathcal{G}$ with grid size $g = 1$ (red dashed lines)
Definitions

Isothetic cover for $g = 1$
Definitions

Isothetic cover for $g = 2$
Definitions

 Isothetic cover for \( g = 3 \)
Naive algorithm
Naive algorithm
Naive algorithm
Naive algorithm
Naive algorithm
Naive algorithm
Naive algorithm
Naive algorithm
Naive algorithm
Naive algorithm
Naive algorithm
Naive algorithm
Naive algorithm

Disadvantages

- Scans the entire image
- Cell joining required to output the vertex sequence

Alternative solution: Combinatorial algorithm.
Naive algorithm

Disadvantages
- Scans the entire image
- Cell joining required to output the vertex sequence

Alternative solution: Combinatorial algorithm.
Vertex types

Fully black cells can be disregarded
Avoid also some partly black cells. Just consider the border cells.
Avoid also some partly black cells. Just consider the border cells.
Vertex types

Avoid the concept of cell joining
The isothetic polygon contains the object
Vertex angles are $90^0$ and $270^0$
Vertex types

Vertex angles are $90^0$ and $270^0$
Backtracking—A serious issue
Backtracking—A serious issue
Grid point classification
Grid point classification
Grid point classification
Grid point classification

[Image of a horse on a grid with labeled points and grid lines]
Grid point classification
Grid point classification
Grid point classification
Grid point classification
Grid point classification
Grid point classification
Grid point classification
Grid point classification
Grid point classification
Grid point classification
Grid point classification
Grid point classification
Grid point classification
Grid point classification
Grid point classification
Grid point classification
Grid point classification
Grid point classification
Grid point classification
Grid point classification
Grid point classification

Class 0

Class 1

Class 2A

Class 2B

Class 3

Class 4
Correctness & Runtime

The line of proof:

- The interior of a cell lies outside \( P_G(S) \) if and only if the cell has no object occupancy.
- All vertices are detected and correctly classified.
- If \( p \) is a point lying on \( P_G(S) \), then \( 0 < d_T(p, S) \leq g \).
- The construction of \( P_G(S) \) always concludes at the start vertex.

Runtime:\(^1\)

- Best case: \( O(|P|/g) \) ← found in practice
- Worst case: \( O(|P|) \)

\(^1\)\(|P| = \) perimeter of \( P_G(S)\)
Correctness & Runtime

The line of proof:

- The interior of a cell lies outside $P_G(S)$ if and only if the cell has no object occupancy.
- All vertices are detected and correctly classified.
- If $p$ is a point lying on $P_G(S)$, then $0 < d_T(p, S) \leq g$.
- The construction of $P_G(S)$ always concludes at the start vertex.

Runtime:\(^1\)

- Best case: $O(|P|/g) \leftarrow$ found in practice
- Worst case: $O(|P|)$

\(^1\)|P| = perimeter of $P_G(S)$
Correctness & Runtime

The line of proof:

- The interior of a cell lies outside $P_G(S)$ if and only if the cell has no object occupancy.
- All vertices are detected and correctly classified.
- If $p$ is a point lying on $P_G(S)$, then $0 < d_T(p, S) \leq g$.
- The construction of $P_G(S)$ always concludes at the start vertex.

Runtime:\(^1\)

- Best case: $O(|P|/g)$ ← found in practice
- Worst case: $O(|P|)$

\(^1|P| = \text{perimeter of } P_G(S)\)
Correctness & Runtime

The line of proof:

- The interior of a cell lies outside \( P_G(S) \) if and only if the cell has no object occupancy.
- All vertices are detected and correctly classified.
- If \( p \) is a point lying on \( P_G(S) \), then \( 0 < d_T(p, S) \leq g \).
- The construction of \( P_G(S) \) always concludes at the start vertex.

Runtime:\(^1\)

- Best case: \( O(|P|/g) \) ← found in practice
- Worst case: \( O(|P|) \)

\(^1\)|P| = perimeter of \( P_G(S) \)
Correctness & Runtime

The line of proof:

- The interior of a cell lies outside $P_G(S)$ if and only if the cell has no object occupancy.
- All vertices are detected and correctly classified.
- If $p$ is a point lying on $P_G(S)$, then $0 < d_T(p, S) \leq g$.
- The construction of $P_G(S)$ always concludes at the start vertex.

Runtime:\footnote{1}{\mbox{$|P|$ = perimeter of $P_G(S)$}}

- Best case: $O(|P|/g) \leftarrow$ found in practice
- Worst case: $O(|P|)$
Orthogonal convex hull

\( H_G(S) \) = smallest-area orthogonal polygon such that

- \( S \) lies inside \( H_G(S) \)
  \[ \Rightarrow P_G(S) \text{ lies inside } H_G(S) \]
- intersection of \( H_G(S) \) with any horizontal or vertical line is either empty or exactly one line segment.

Algorithm—Uses combinatorial rules over vertex subsequences.
Runtime—Linear on perimeter of \( P_G(S) \).
Orthogonal convex hull

\[ H_G(S) = \text{smallest-area orthogonal polygon such that} \]

- \( S \) lies inside \( H_G(S) \)
  \[ \Rightarrow P_G(S) \text{ lies inside } H_G(S) \]
- intersection of \( H_G(S) \) with any horizontal or vertical line is either empty or exactly one line segment.

Algorithm—Uses combinatorial rules over vertex subsequences.
Runtime—Linear on perimeter of \( P_G(S) \).
Orthogonal convex hull

\( H_G(S) = \) smallest-area orthogonal polygon such that

- \( S \) lies inside \( H_G(S) \)
  \[ \Rightarrow P_G(S) \text{ lies inside } H_G(S) \]
- intersection of \( H_G(S) \) with any horizontal or vertical line is either empty or exactly one line segment.

Algorithm—Uses combinatorial rules over vertex subsequences.

Runtime—Linear on perimeter of \( P_G(S) \).
Orthogonal convex hull

$H_G(S) =$ smallest-area orthogonal polygon such that

- $S$ lies inside $H_G(S)$
  $\Rightarrow P_G(S)$ lies inside $H_G(S)$

- intersection of $H_G(S)$ with any horizontal or vertical line is either empty or exactly one line segment.

**Algorithm**—Uses combinatorial rules over vertex subsequences.

**Runtime**—Linear on perimeter of $P_G(S)$. 
Orthogonal convex hull

\[ H_G(S) = \text{smallest-area orthogonal polygon such that} \]

- \( S \) lies inside \( H_G(S) \)
  \[ \Rightarrow P_G(S) \text{ lies inside } H_G(S) \]
- intersection of \( H_G(S) \) with any horizontal or vertical line is either empty or exactly one line segment.

**Algorithm**—Uses combinatorial rules over vertex subsequences.

**Runtime**—Linear on perimeter of \( P_G(S) \).
Orthogonal convex hull

\[ H_G(S) = \text{smallest-area orthogonal polygon such that} \]
- \( S \) lies inside \( H_G(S) \)
  \[ \Rightarrow P_G(S) \text{ lies inside } H_G(S) \]
- intersection of \( H_G(S) \) with any horizontal or vertical line is either empty or exactly one line segment.

**Algorithm**—Uses combinatorial rules over vertex subsequences.

**Runtime**—Linear on perimeter of \( P_G(S) \).
Orthogonal convex hull

\[ H_G(S) = \text{smallest-area orthogonal polygon such that} \]

- \( S \) lies inside \( H_G(S) \)
  \[ \Rightarrow P_G(S) \text{ lies inside } H_G(S) \]
- intersection of \( H_G(S) \) with any horizontal or vertical line is either empty or exactly one line segment.

**Algorithm**—Uses combinatorial rules over vertex subsequences.

**Runtime**—Linear on perimeter of \( P_G(S) \).
Orthogonal convex hull

\( H_G(S) = \) smallest-area orthogonal polygon such that

\( S \) lies inside \( H_G(S) \)

\( \Rightarrow P_G(S) \) lies inside \( H_G(S) \)

intersection of \( H_G(S) \) with any horizontal or vertical line is either empty or exactly one line segment.

**Algorithm**—Uses combinatorial rules over vertex subsequences.

**Runtime**—Linear on perimeter of \( P_G(S) \).
Orthogonal convex hull

$g = 14$
Orthogonal convex hull

\[ g = 8 \]
Orthogonal convex hull

$g = 4$
Convex partitioning
Convex partitioning
Shortest isothetic path
3D cover (outer)
3D cover (outer)

$g = 3$
3D cover (outer)

\[ g = 4 \]
3D cover (outer)

$g = 6$
3D cover (outer)

\[ g = 8 \]
3D cover (outer)

g = 10
3D cover (outer)

$g = 12$
3D cover (outer)

$g = 16$
3D cover (inner)

\[ g = 2 \]
3D cover (inner)

\[ g = 4 \]
$g = 6$
3D cover (inner)

\[ g = 8 \]
3D cover (inner)

\[ g = 12 \]
3D cover (inner)

\[ g = 16 \]
3D slicing

high resolution
3D slicing along $x$-axis
3D slicing

along $y$-axis
3D slicing along z-axis
3D slicing

low resolution
Isotthetic Cover
P. Bhowmick
Introduction
Naive
Combinatorial Applications
Hull Shape
3D

3D slicing

along $x$-axis
3D slicing

along y-axis
3D slicing

along z-axis
Further reading I


Further reading II


Thank You