# Simple tools for basic euclidean geometry

Guillaume Charpiat

September 17, 2008

## Introduction

The purpose of this is to document the basic algorithms I use to transform a binary image into a *shape*, which may have a non-trivial topology, how to compute normals that point towards the outside of the shape, and how to register any two shapes with simple, fast tools.

## Contents

**1 Segmentation of a binary image** **2**

**2 Pre-processing polygons** **2**
2.1 Smoothing . . . . . 2
2.2 Sub-sampling . . . . . 2
2.3 Over-sampling . . . . . 3
2.4 Computation of normals . . . . . 3
2.5 Smoothing of normals . . . . . 3

**3 Test whether a point is inside a polygon** **3**
3.1 Number of intersections between a half-line and a polygon . . . . . 3
3.2 Intersection of a half-line and a segment . . . . . 4

**4 Normals' orientation** **5**
4.1 Orientation of normals towards the outside of a polygon . . . . . 5
4.2 Orientation of normals towards the outside of a shape . . . . . 6

**5 Simple shape registration : translation/scale** **6**
5.1 Area of a polygon . . . . . 7
5.2 Center of mass . . . . . 7

# 1 Segmentation of a binary image

**Data :** a black and white image (grey levels = 0 and 255 only) : a black object, possibly not topologically trivial, on a white background
**Desired output :** the contour of the object (non-oriented yet)
**Difficulties :** pixelization, ordering segments

Finding open/closed connected components:

- Find edges : one side of a pixel (a "segment") belongs to the contour if and only if the two neighboring pixels have different colors
- Special case : one might decide that all black pixel sides on the image border belong to the contour (in that case all connected components of the contour will be closed curves)
- Find neighbors of all contour segments (2 in the middle of a straight line [check following special case], 1 on the image border if previous special case ignored)
- Special case : tangent components, i.e. a square window of $2 \times 2$ pixels with a black diagonal and a white diagonal. Arbitrary decision: the black pixels are connected, the white ones aren't. This solution minimizes the total number of (black) connected components.
- Pick up any segment and follow recursively its neighbors in the same direction (twice, one for each possible initial direction), until dead-end (if open connected component) or back to initial pixel. This is one component. Label segments accordingly.
- Do previous step as long as unlabeled segments exist.
- Store components into polygons (one polygon = ordered table of successive contour points, with a label "closed" or "open")

Complexity : linear in number of image pixels (mostly linear in number of contour length with a small factor).

# 2 Pre-processing polygons

## 2.1 Smoothing

Angles are only 0 or $\pi/2$, so there is a need for smoothing. Just apply Gaussian smoothing with a predefined standard deviation (e.g. 3 pixels) along the polygons.

## 2.2 Sub-sampling

- In case of previous regular discretization : pick regularly one vertex every $n$ initial vertices.

- Otherwise : pick up vertices so that the length between two successive vertices is never greater than $l$.

### 2.3 Over-sampling

- In case of previous regular discretization : just cut each segment into $n$ parts.
- Otherwise : add points along the contour, regularly spaced (does not necessarily include original points)

### 2.4 Computation of normals

- At any vertex : based on the two neighboring vertices. Consider $\overrightarrow{P_{i-1}P_{i+1}}$, turn it with the angle $\pi/2$, and normalize it.
- Special case (extremity of an open polygon) : orthogonal to the segment itself.
- On a segment : easy ! Turn $S_i = \overrightarrow{P_iP_{i+1}}$ with an angle $\pi/2$ and normalize it.

In all cases : orientation is arbitrary but coherent : all normals of a same component point towards the outside (or the inside) of the component (for example, always point to the right when following the contour). The sign of the curvature is not taken into account.

### 2.5 Smoothing of normals

The same : a Gaussian mask along the contour, a few pixel wide.

## 3 Test whether a point is inside a polygon

Point $A$, closed polygon $P$ with ordered successive vertices $(P_i)_{1 \leqslant i \leqslant n}$.
Notation: $P_{n+1} := P_1$.
$n$ segments : $S_i = [P_i, P_{i+1}]$.

### 3.1 Number of intersections between a half-line and a polygon

Consider any random (unit normed) direction $\overrightarrow{u}$, and the half-line $[A, \overrightarrow{u})$ starting from $A$ with direction $\overrightarrow{u}$. Just count the number $c$ of times this half-line intersects the polygon $P$. Then, if $c$ is odd, $A$ is inside, and otherwise $A$ is outside.

Pathological cases:

- if $A$ is on the boundary $P$ itself. Easy to detect ($A$ belongs to one segment). Boundary $P$ is considered as included in the "inside" of $P$.

- if the half-line includes one segment (very low probability). Easy to detect. Pick up another random direction.
- if the half-line goes through one vertex (then the line could cross the polygon at this point or be tangent). Easy detection (one vertex belongs to the half-line). Pick up another random direction. Note : this case includes the previous one.

To count the number of intersection with the polygon, just check successively how many segments of the polygon the half-line crosses.

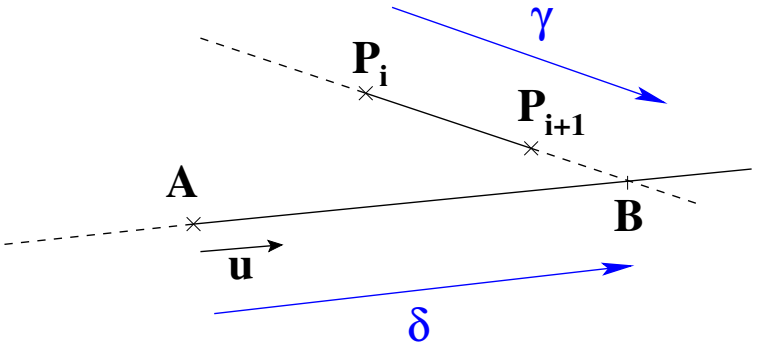
## 3.2 Intersection of a half-line and a segment

Point $A$, direction $\overrightarrow{u}$, segment $S_i = [P_i, P_{i+1}]$.
We know that the half-line $[A, \overrightarrow{u})$ does not go through $P_i$ or $P_{i+1}$.

Special case:

- if $S_i$ and $\overrightarrow{u}$ are parallel. Then because of hypotheses (the half-line does not go through the vertices), there is no intersection.

Normal case : let $B$ the intersection of the two lines, $\delta$ the possibly negative distance $\overline{AB}$ along $(A, \overrightarrow{u})$, and $\gamma$ the possibly negative relative distance $\overline{P_iB}/P_iP_{i+1}$ along $(P_i, P_{i+1})$.




Then:

- if $\delta < 0$ : no intersection
- else if $\gamma < 0$ or $\gamma > 1$ : no intersection
- else : intersection

Writing definitions gives :

$$\gamma \overrightarrow{P_iP_{i+1}} = \overrightarrow{P_iA} + \delta \overrightarrow{u}$$

With two projections:

$$\begin{cases} \gamma \, P_iP_{i+1}^2 = \overrightarrow{P_iA} \cdot \overrightarrow{P_iP_{i+1}} + \delta \overrightarrow{u} \cdot \overrightarrow{P_iP_{i+1}} \\ \gamma \, (\overrightarrow{P_iP_{i+1}} \cdot \overrightarrow{u}) = \overrightarrow{P_iA} \cdot \overrightarrow{u} + \delta \end{cases}$$

This implies

$$\gamma = \frac{\overrightarrow{P_iA} \cdot \overrightarrow{P_iP_{i+1}} - (\overrightarrow{P_iA} \cdot \overrightarrow{u})(\overrightarrow{P_iP_{i+1}} \cdot \overrightarrow{u})}{\overrightarrow{P_iP_{i+1}}^2 - (\overrightarrow{P_iP_{i+1}} \cdot \overrightarrow{u})^2}$$

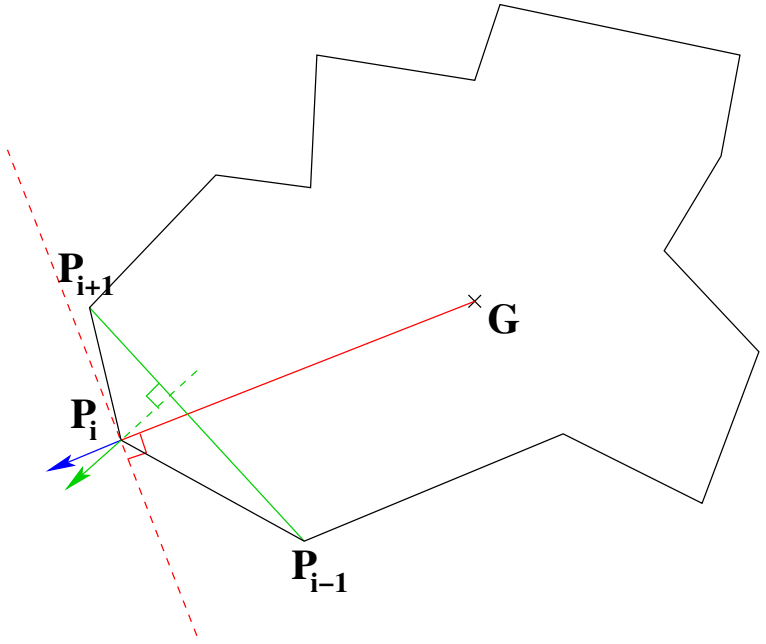
and

$$\delta = (\gamma \overrightarrow{P_iP_{i+1}} - \overrightarrow{P_iA}) \cdot \overrightarrow{u}$$

# 4 Normals' orientation

A shape is a set of non-intersecting polygons.
A polygon is here a closed contour (otherwise there is no inside/outside).
Hypothesis : normals have been coherently pre-computed along polygons (see 2.4).

## 4.1 Orientation of normals towards the outside of a polygon




Compute the gravity center $G$ (of the contour), consider the (or one of the) farthest point $P_i$ from $G$. A tangent line at this point cut the plane in two parts, one containing all other points $P_j$ and one without any. Then necessarily the inner product between the normal at point $P_i$ towards the outside of the polygon and the vector $\overrightarrow{GP_i}$ is positive.
Notes:

- Any point $P_i$ on the convex hull is ok, for example one can consider the vertex with smallest coordinate $x$...
- The normal at point $P_i$ should be computed carefully. Just considering $\overrightarrow{P_{i-1}P_{i+1}}$ rotated with $\pi/2$ may fail. Instead, just reduce the neighborhood to an isosceles triangle by replacing $P_{i-1}$ by $P_i + \overrightarrow{P_iP_{i-1}}/P_iP_{i-1}$, and $P_{i+1}$ by $P_i + \overrightarrow{P_iP_{i+1}}/P_iP_{i+1}$ to compute the normal at $P_i$.

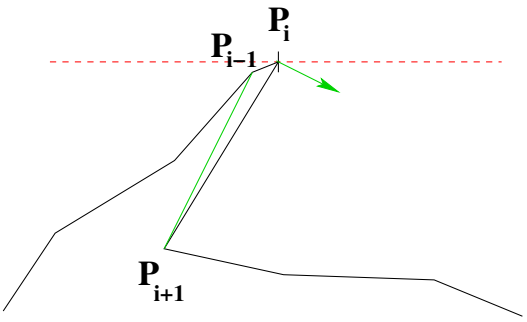



Figure 1: Bad case

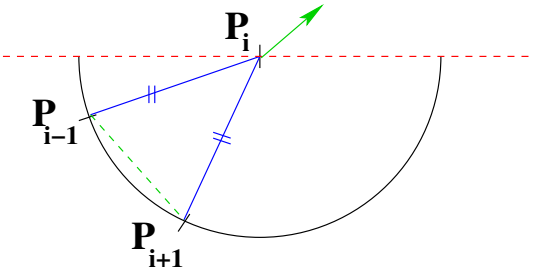



Figure 2: Isosceles is always good

### 4.2 Orientation of normals towards the outside of a shape

The orientation of normals on each polygon are coherent within the polygon and point towards the exterior of the polygon. The question is whether they point towards the interior of the *shape* or towards its exterior. Easy solution : for each polygon, count the number of polygons (of the shape) it is included into. To test if one polygon is (strictly) inside another one, just test whether one point of the former is inside or outside the latter... (section 3).

## 5 Simple shape registration : translation/scale

One has to choose a way to register one shape onto another one in order to compare them. One simple solution is to perform no registration, arguing that two shapes identical up to a rigid transformation are different. Another point of view is to claim that they are identical. The difficulty is when one tries to register (rigidly) two different shapes : what is the optimal registration ?

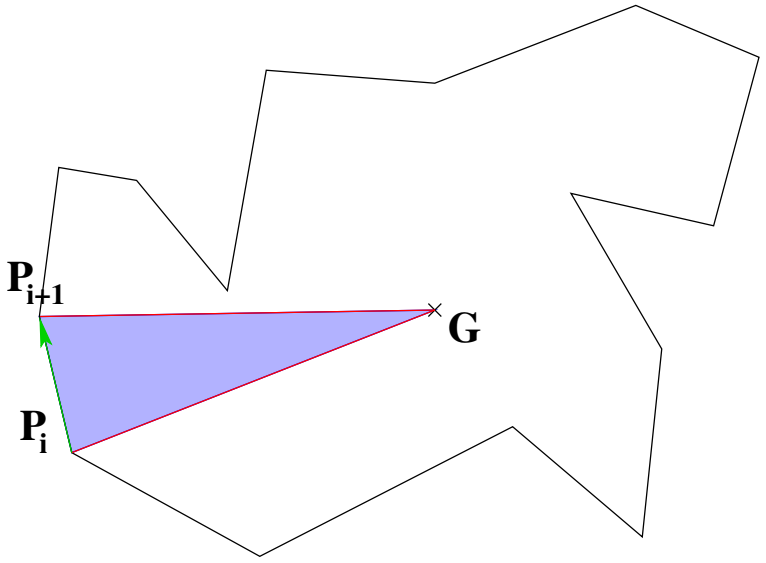
- Registration with respect to location (translations) : usual consensus and easy to perform.
- Registration with respect to angle (rotations) : in the simplest form, easy to perform (just align first moments) but this is ill-posed when there is no clear first direction (e.g. for a circle, a star, a square, anything not elongated). In the global form, very hard to perform (gradient descents with many initialization of angles ?) if the objects are not supposed to have about the same orientation.
- Registration with respect to size (scalings) : with respect to what ? Length (bad idea for noisy shapes or self-occluding shapes) ? Diameter ? Area ?

The good way to solve the problem is to first define a metric on the shape space and then minimize the distance between the two shapes with respect to all parameters (position, angle, scale). Also don't forget that the meaning of the matching computed between two shapes depends strongly on the kind of registration you performed earlier. Without registration : true movement. Registration with respect to position and angle : true deformation (but the first axis cannot change!). Registration with respect to size also : relative deformation, volume preserving.

Here I will just present the computation of the mass and of the center of mass (easy solution). For human silhouettes one can consider that an angle variation is an intrinsic deformation of the silhouette (i.e., that standing is different from lying).

## 5.1 Area of a polygon

Computing the area of a polygon is not more complex than computing its length.




One just needs to consider any point $G$ (not necessarily the center of mass) and sum the (possibly negative) areas of all triangles $GP_iP_{i+1}$.

$$Area = \left| \sum_i \overrightarrow{GP_i} \wedge \overrightarrow{P_iP_{i+1}} \right|$$

Note : this quantity does not depend on $G$.

## 5.2 Center of mass

Computing the center of mass $M$ of the whole polygon (the surface) is not more complex than computing the center of mass of its contour. The algorithm is essentially the same as the one for the area.

$$\overrightarrow{OM} = \frac{1}{3 \sum_j \overrightarrow{OP_j} \wedge \overrightarrow{P_jP_{j+1}}} \sum_i \left( \overrightarrow{OP_i} \wedge \overrightarrow{P_iP_{i+1}} \right) \left( \overrightarrow{OP_i} + \overrightarrow{OP_{i+1}} \right)$$

Note: the resulting point $M$ does not depend on the initial point $O$.

# Conclusion

Life is easy. No need for mesh triangulation.