Polygon Matching
Measuring similarity between polygons

Philip Mitchell

July 19, 2010
Outline

Introduction

Metrics

Comparing whole polygons

Comparing partial polygons

Summary

Questions
Introduction

Polygon matching or, more generally, shape matching measures how similar shapes are.

There are two common ways to do this:

- **Image intensity** compare two images based on colour and texture
- **Geometry** construct polygons and line segments from the image and compare those

We will focus the geometry comparisons.
Motivation

There are many fields where shape matching is vital:

- **Databases**  image retrieval based on content
- **Forensics**  comparison of faces, fingerprints, etc...
- **Archaeology**  assembling broken artifacts
- **Robotics**  object recognition
- **Photography**  post-processing of images
- **Agriculture**  identifying “bad apples”
- **Where’s Waldo?**  finding Waldo in all those books
Variants

There are several ways to look at the problem:

**Computation** compute the dissimilarity between two shapes

**Decision** decide whether the dissimilarity between two shapes is within a given threshold (with or without transformation)

**Optimization** find a transformation that minimizes the dissimilarity between two shapes (possibly within a specified factor)

Here, transformations can be any affine transformation or they can be limited to translation and/or rotation.
When measuring dissimilarity, we need to find a suitable metric based on our goals.

A metric $d : S \times S \rightarrow R$ must meet three criteria for all $x, y, z \in S$:

1. $d(x, x) = 0$
2. $d(x, y) = 0 \iff x = y$
3. $d(x, y) + d(x, z) \geq d(y, z)$ [triangle inequality]
Position functions

- Polygon boundaries are generally represented as a position function (or set of vertices).
- Linear distance (for example) can be used as a metric with position functions.
- Difficult to deal with transformations and noise.
- Noise can be removed by approximating the polygon (e.g. removing vertices that don’t change the shape much), but that is slow ($O(n^2 \lg n)$) and unreliable.
Turning functions are a cumulative measure of the angles through which a polygonal curve turns.

More formally, the turning function $\theta_A(s)$ of a polygon $A$ gives the angle between the counterclockwise tangent and the $x$-axis as a function of distance $s$ along the polygonal curve.

- Increases with left-hand turns, Decreases with right-hand turns
- Invariant under translation
- A rotation of $\theta$ causes a vertical shift of distance $\theta$
- Scaling can distort the function
Figure: The turning function of a polygon is its angle with the horizontal axis as a function of distance along the boundary edge.
An $L_p$ metric is defined as follows:

$$|f_{A,B}|^p = \int |\theta_A(x) - \theta_B(x)|^p dx$$

So, for example, if we wanted to compare two turning functions $\theta_A$ and $\theta_B$ using the $L_2$ metric, the dissimilarity $d_{A,B}$ would be:

$$d^2_{A,B} = \int (\theta_A(s) - \theta_B(s))^2 ds$$

This is a measure of the squared difference between the shapes.
Comparing polygons

Figure: Comparing two polygons under translation and rotation.
Comparing polygons using $L_2$ and turning functions

Considering all variables

We want to compare polygon $A$ with $m$ vertices and polygon $B$ with $n$ vertices.

Translation is ignored by the turning function, however, We need to consider rotation $\theta$:

$$d^2_{A,B}(\theta) = \int (\theta_A(s) - \theta_B(s) + \theta)^2 ds$$

Furthermore, because the polygonal curves on the boundaries of the polygons are closed, we must choose a starting point along one of the boundaries:

$$d^2_{A,B}(s_A, \theta) = \int (\theta_A(s + s_A) - \theta_B(s) + \theta)^2 ds$$

Scaling can be considered by rescaling both polygons so that each has a unit-length perimeter before computing the turning function.
Comparing polygons using $L_2$ and turning functions

Turning functions

Figure: The turning functions $\theta_A$ and $\theta_B$. The rectangular strips represent the difference between the functions.
Comparing polygons using $L_2$ and turning functions

Finding the rotation

If $s_A$ is fixed, then $d_{A,B}(s_A, \theta)$ is minimal for

$$\theta = \int \theta_B(s)ds - \int \theta_A(s)ds - 2\pi s_A$$

Assuming the polygon has straight edges, this is evaluated as the sum of $O(m + n)$ terms.
Comparing polygons using $L_2$ and turning functions

Finding $s_A$

- We must find, for each possible value of $s_A$, the optimal value of $\theta$ and thus the value of $d_{A,B}(s_A, \theta)$.
- Consider only locations where vertices meet, of which there are $O(mn)$ possibilities.
- Computing for all of these gives a naive algorithm that runs in $O(mn(m + n))$ time.
- This can be reduced to $O(mn \log(mn))$ time with incremental evaluation.
- Choose the minimum value over all $s_A$
Partial polygons

**Figure:** Comparing two polygons along a fixed-length edge segment under translation and rotation. As shown here, this can be useful for fitting together broken pieces of a larger polygon (along the darker line).
Comparing partial polygons using $L_2$ and turning functions

Added complexity

We still want to compare polygon $A$ with $m$ vertices and polygon $B$ with $n$ vertices. However, want to find the best match along a fixed-length portion of their boundaries.

Comparing two polygons based on a fixed-length $l$ portion of their boundaries reduces the distance around the shape that we must integrate, but it also adds an extra piece of complexity.

Specifically, we must also consider at what point $s_B$ along the polygon $B$ boundary to start.

\[
d^2_{A,B,l}(s_A, s_B, \theta) = \int_0^l (\theta_A(s + s_A) - \theta_B(s + s_B) + \theta)^2 ds
\]
Comparing partial polygons using $L_2$ and turning functions

Finding the rotation

If $s_A$ and $s_B$ are fixed, then $d_{A,B,l}(s_A, s_B, \theta)$ is minimal for

$$\theta = \frac{\int_0^l (\theta_A(s + s_A) - \theta_B(s + s_B)) ds}{l}$$

Substituting and simplifying, we get:

$$d_{A,B,l}^2(s_A, s_B) = \int_0^l (\theta_A(s + s_A) - \theta_B(s + s_B))^2 ds - \frac{\int_0^l (\theta_A(s+s_A)-\theta_B(s+s_B))ds^2}{l}$$

Similar to the full polygon case, this can be computed in $O(m + n)$ time.
Comparing partial polygons using $L_2$ and turning functions

Edge contribution length

Finding the optimal values for $s_A$ and $s_B$ is much harder than with the entire polygon because there are two degrees of freedom.

We define $X_{i,j}(s_A, s_B)$ to be the length of the contribution (overlap) of edge $a_i$ of polygon $A$ along edge $b_j$ of polygon $B$.

$$X_{i,j}(s_A, s_B) = |(\max\{0, a_i - s_A, b_j - s_B\}, \min\{l, a_{i+1} - s_A, b_{j+1} - s_B\})|$$

where $|(x, y)| = \max\{y - x, 0\}$. 
Comparing partial polygons using $L_2$ and turning functions

Edge contribution length (part 2)

$X_{i,j}$ has 10 possible values:

1. $l$
2. $a_{i+1} - s_A$
3. $b_{j+1} - s_B$
4. $l - (a_i - s_A)$
5. $a_{i+1} - s_A - (a_i - s_A)$
6. $b_{j+1} - s_B - (a_i - s_A)$
7. $l - (b_j - s_B)$
8. $a_{i+1} - s_A - (b_j - s_B)$
9. $b_{j+1} - s_B - (a_j - s_B)$
10. 0
Comparing partial polygons using $L_2$ and turning functions

Solution space

The plane defined by $s_A$ and $s_B$ represents the entire solution space. It can be divided into regions based on each $X_{i,j}$ function by adding lines where the entirety of either line $a_i$ or $b_j$ will be used:

1. $0 = a_i - s_A$ (horizontal)
2. $0 = b_i - s_B$ (vertical)
3. $a_i - s_A = b_i - s_B$ (diagonal)

as well as where no part of $a_i$ and $b_j$ will be used:

1. $l = a_i - s_A$ (horizontal)
2. $l = b_i - s_B$ (vertical)

Within each of the regions defined by these lines, $X_{i,j}$ is linear with respect to $s_A$ and $s_B$. 
Comparing partial polygons using $L_2$ and turning functions

Divided solution space

Figure: The solution space is split by horizontal, vertical, and diagonal lines into regions where the solution function is linear.
Comparing partial polygons using $L_2$ and turning functions

Finding the solution

To find the solution, we need only look at the crossing points on the plane defined by $s_A$ and $s_B$ after all $X_{i,j}$ have had their lines added.

1. Walk along each diagonal line. There are at most $nm$ of these.
   1.1 Find a crossing with a horizontal or vertical line. There are at most $2n + 2m$ of these
   1.2 Compute $d_{A,B,l}^2(s_A, s_B)$ in $O(n + m)$ time
   1.3 Find each subsequent crossing point, subtract the contribution from segments that no longer overlap and add the contribution from the segments that have been added in $O(1)$ time.

2. Check each of the $O(mn)$ crossings of horizontal and vertical lines, each in $O(m + n)$ time.
Summary

- The dissimilarity between two full polygons can be found in $O(mn \lg mn)$ time.
- The dissimilarity between two polygons along a fixed-length edge segment requires $O(nm(n + m))$ time, slower than that of the whole polygon due to the extra degree of freedom.
- In either case, only $O(m + n)$ space is used. It is not possible to do better because both polygons must be stored.
References

Questions?

Thank you