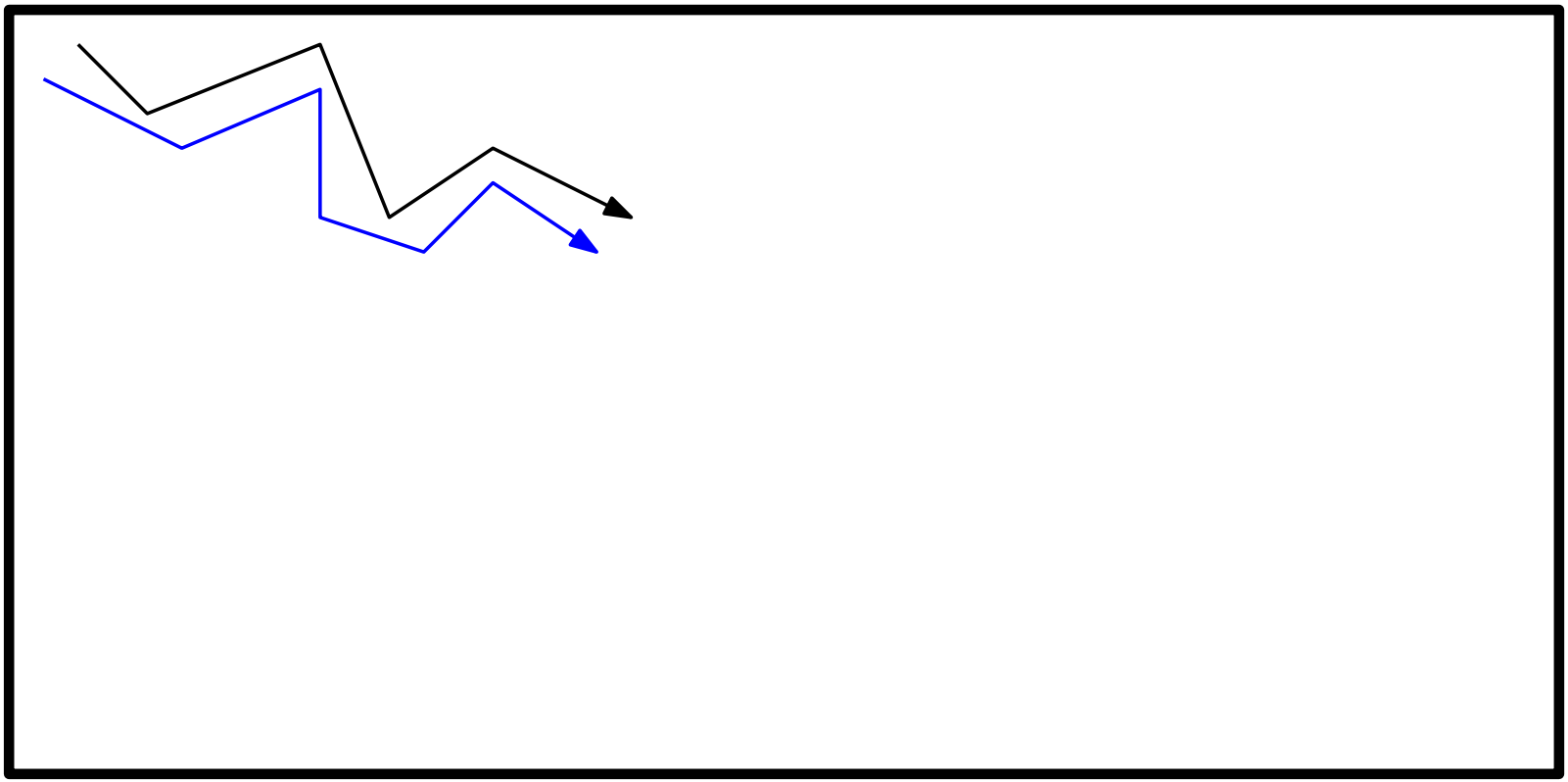
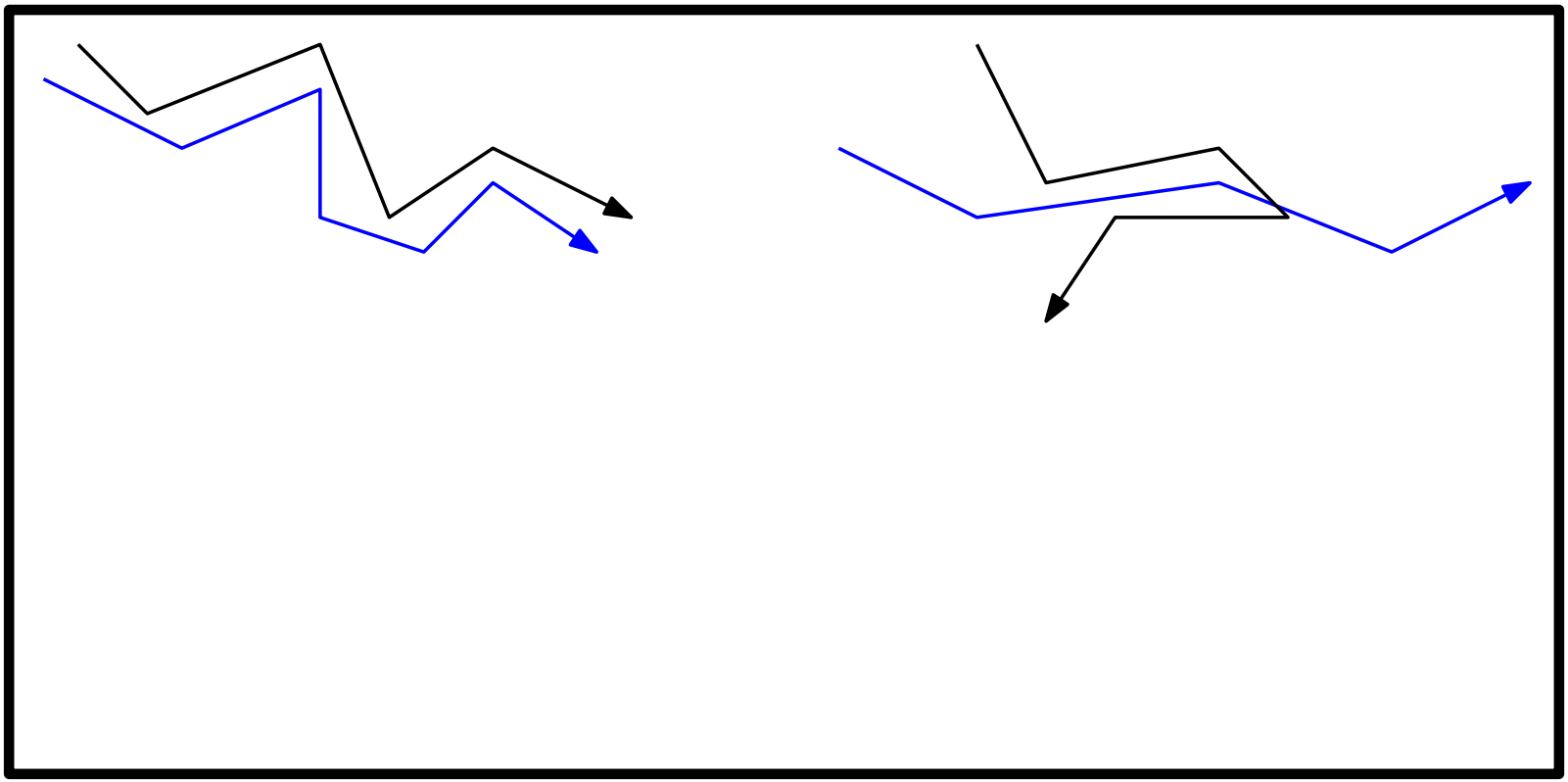
How similar are these curves?

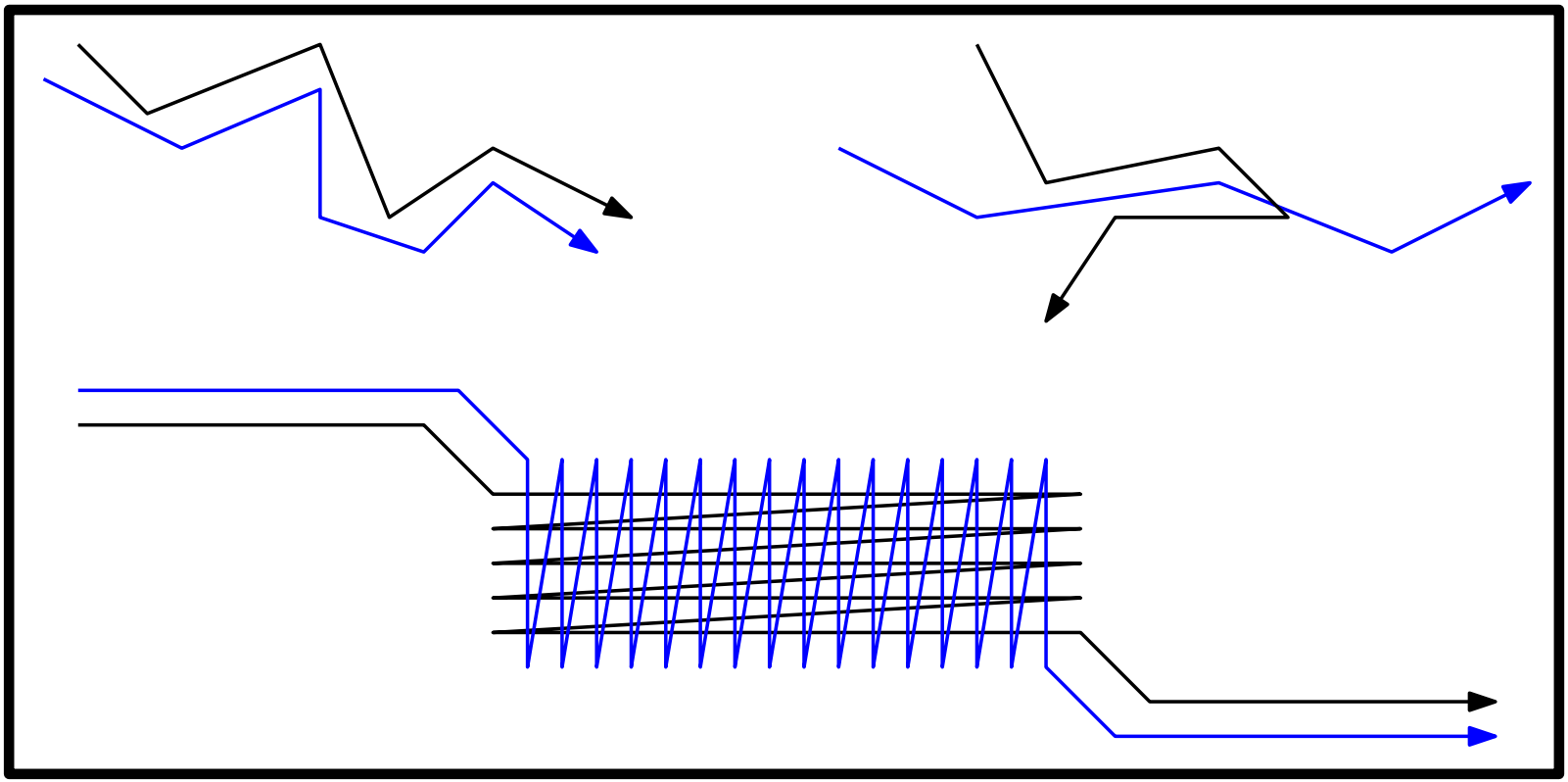
How similar are these curves?



How similar are these curves?



How similar are these curves?



Measuring Similarity of Shapes

- There are many different measures of similarity.

Measuring Similarity of Shapes

- There are many different measures of similarity.
- The Fréchet distance is a natural measure for continuous shapes such as curves and surfaces.

Measuring Similarity of Shapes

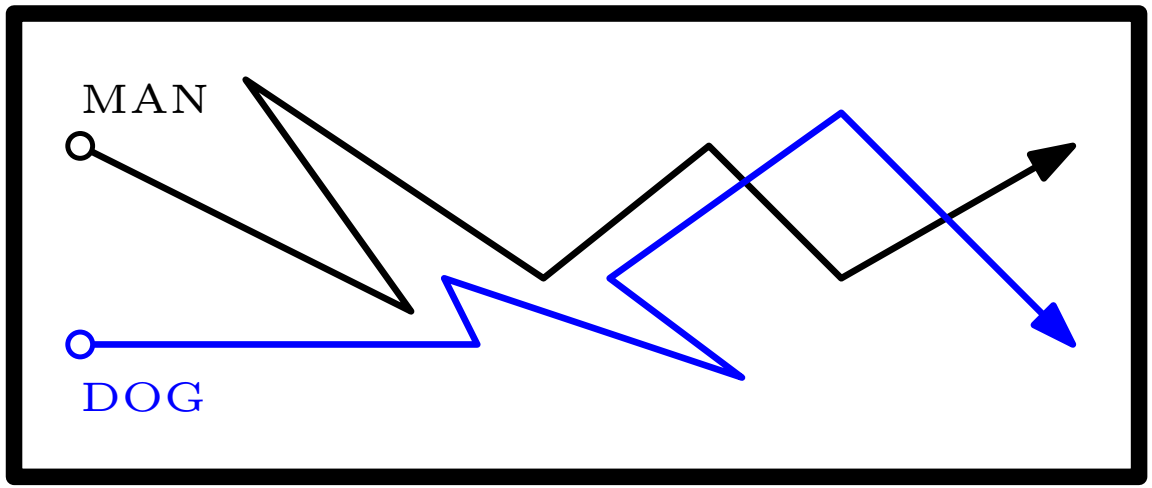
- There are many different measures of similarity.
- The Fréchet distance is a natural measure for continuous shapes such as curves and surfaces.
- Our prior work has focused on computing the Fréchet distance between surfaces.

Measuring Similarity of Shapes

- There are many different measures of similarity.
- The Fréchet distance is a natural measure for continuous shapes such as curves and surfaces.
- Our prior work has focused on computing the Fréchet distance between surfaces.
- So what is the Fréchet distance?

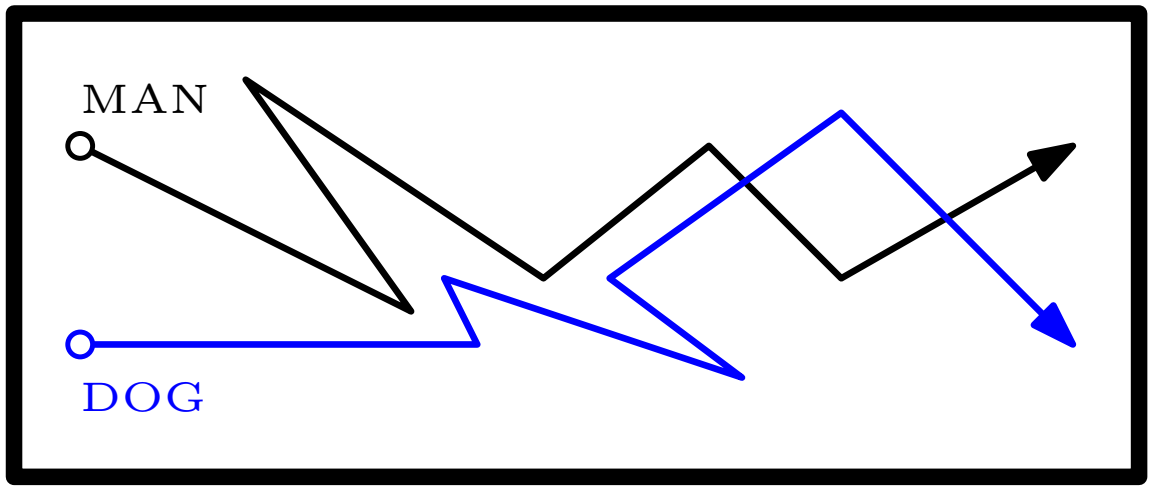
Fréchet Distance for Curves Example

- A man and a dog walk along their assigned curves.



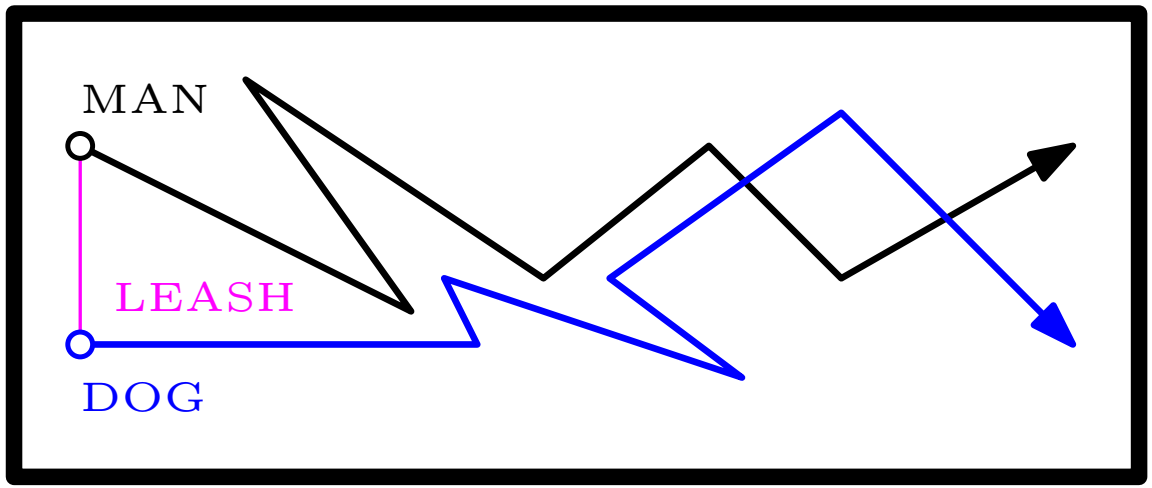
Fréchet Distance for Curves Example

- A man and a dog walk along their assigned curves.
- They are connected by a leash.



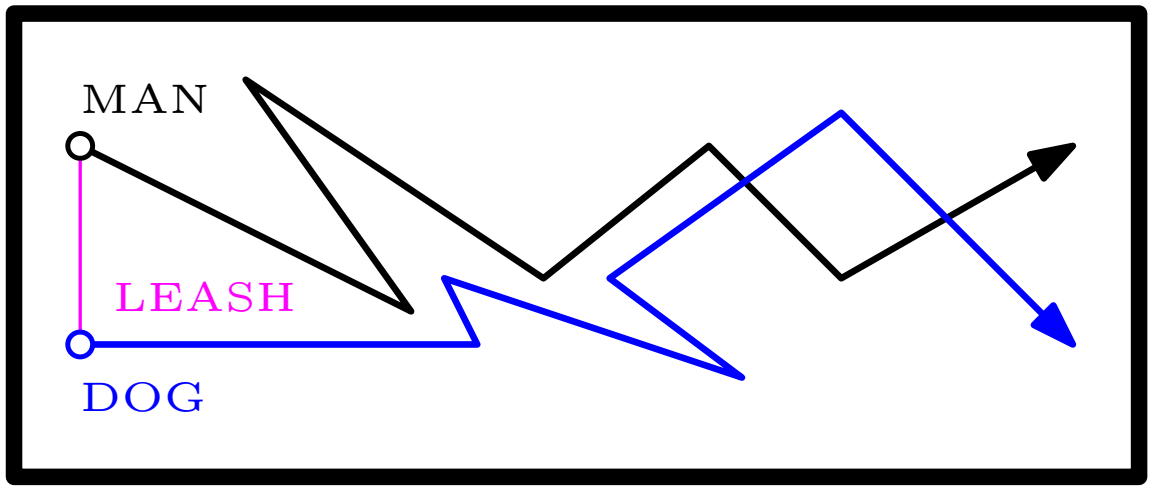
Fréchet Distance for Curves Example

- A man and a dog walk along their assigned curves.
- They are connected by a leash.
- They can control their speeds but cannot go backwards.



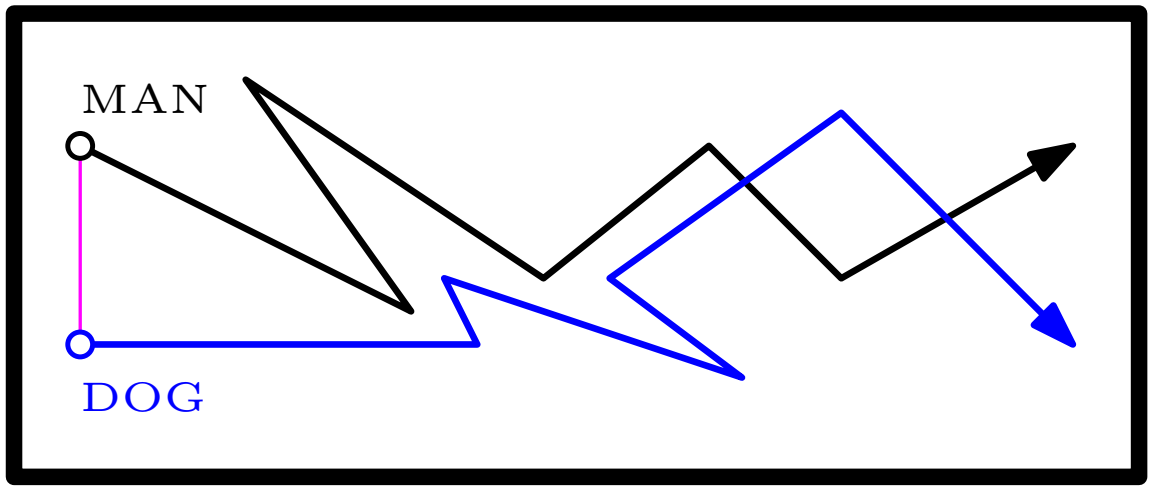
Fréchet Distance for Curves Example

- A man and a dog walk along their assigned curves.
- They are connected by a leash.
- They can control their speeds but cannot go backwards.
- The minimum leash required for any possible walk is the Fréchet distance of the curves.



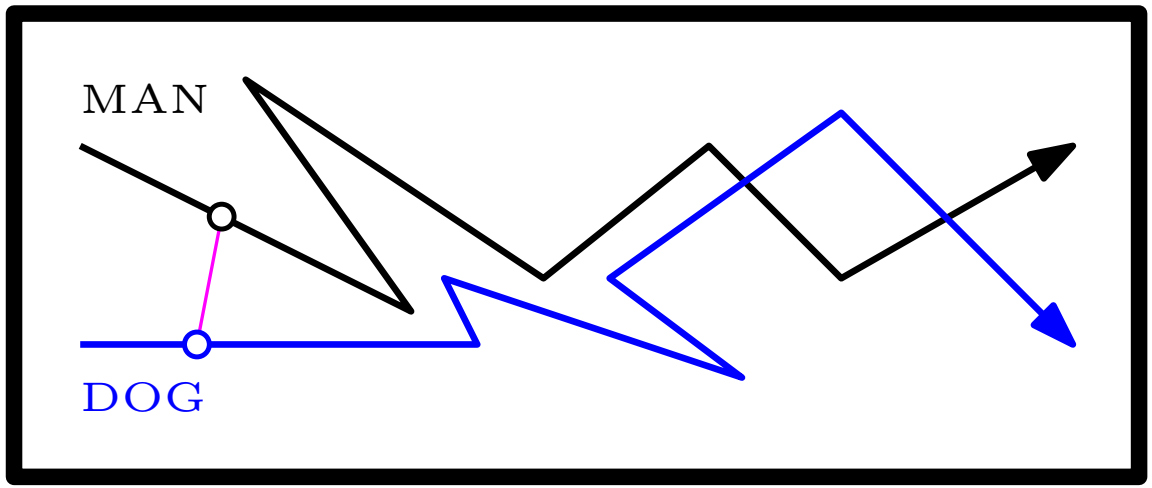
Fréchet Distance for Curves Example

- A man and a dog walk along their assigned curves.
- They are connected by a leash.
- They can control their speeds but cannot go backwards.
- The minimum leash required for any possible walk is the Fréchet distance of the curves.



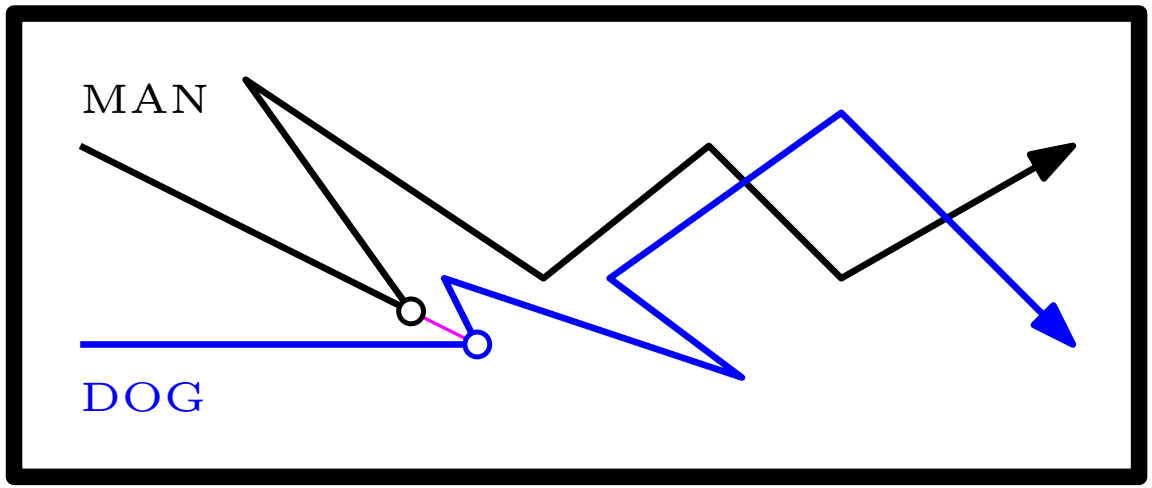
Fréchet Distance for Curves Example

- A man and a dog walk along their assigned curves.
- They are connected by a leash.
- They can control their speeds but cannot go backwards.
- The minimum leash required for any possible walk is the Fréchet distance of the curves.



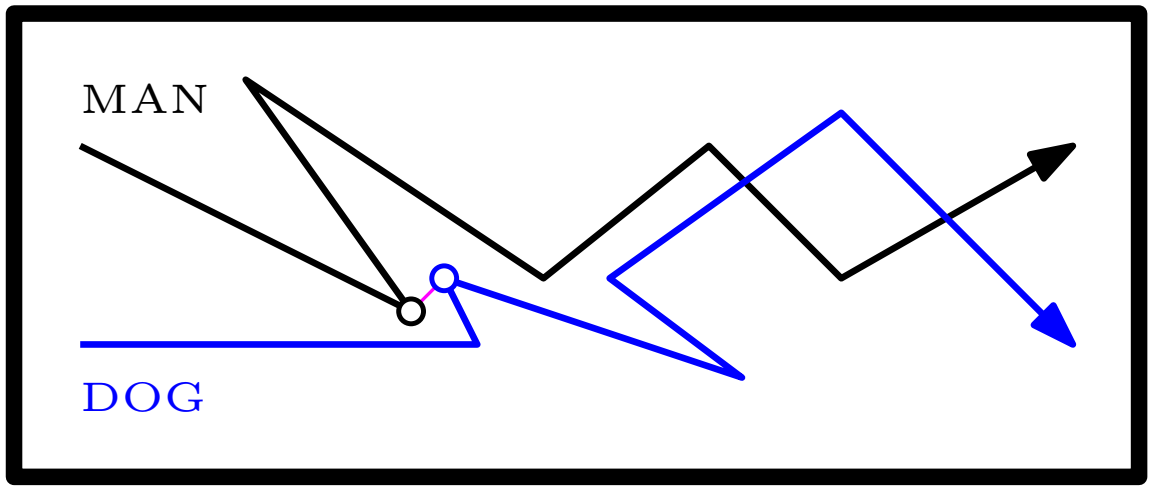
Fréchet Distance for Curves Example

- A man and a dog walk along their assigned curves.
- They are connected by a leash.
- They can control their speeds but cannot go backwards.
- The minimum leash required for any possible walk is the Fréchet distance of the curves.



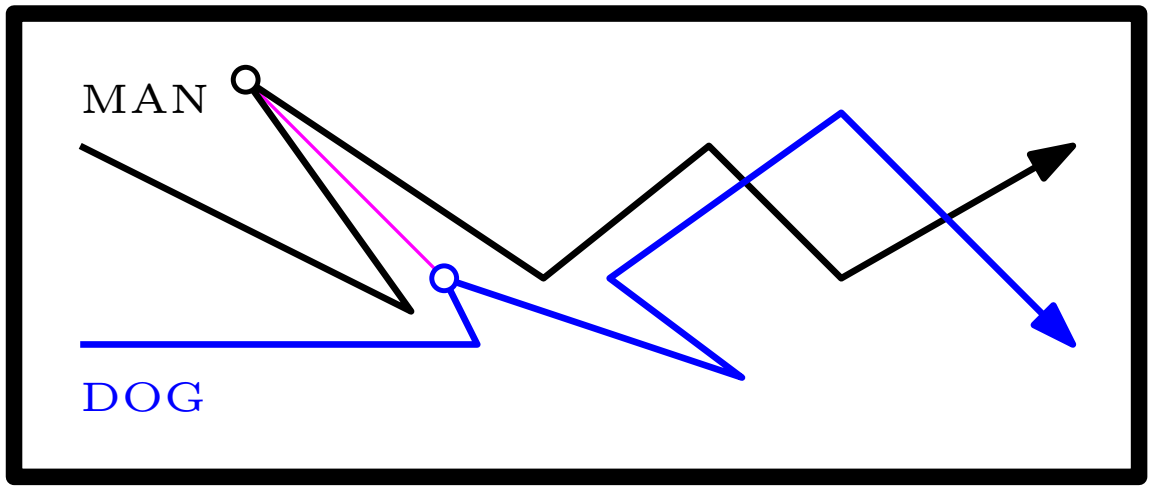
Fréchet Distance for Curves Example

- A man and a dog walk along their assigned curves.
- They are connected by a leash.
- They can control their speeds but cannot go backwards.
- The minimum leash required for any possible walk is the Fréchet distance of the curves.



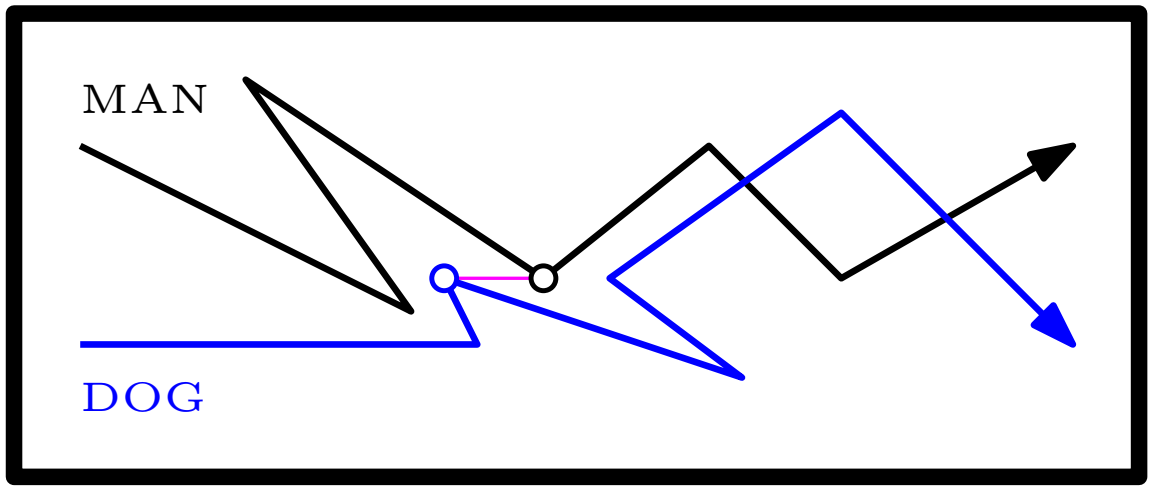
Fréchet Distance for Curves Example

- A man and a dog walk along their assigned curves.
- They are connected by a leash.
- They can control their speeds but cannot go backwards.
- The minimum leash required for any possible walk is the Fréchet distance of the curves.



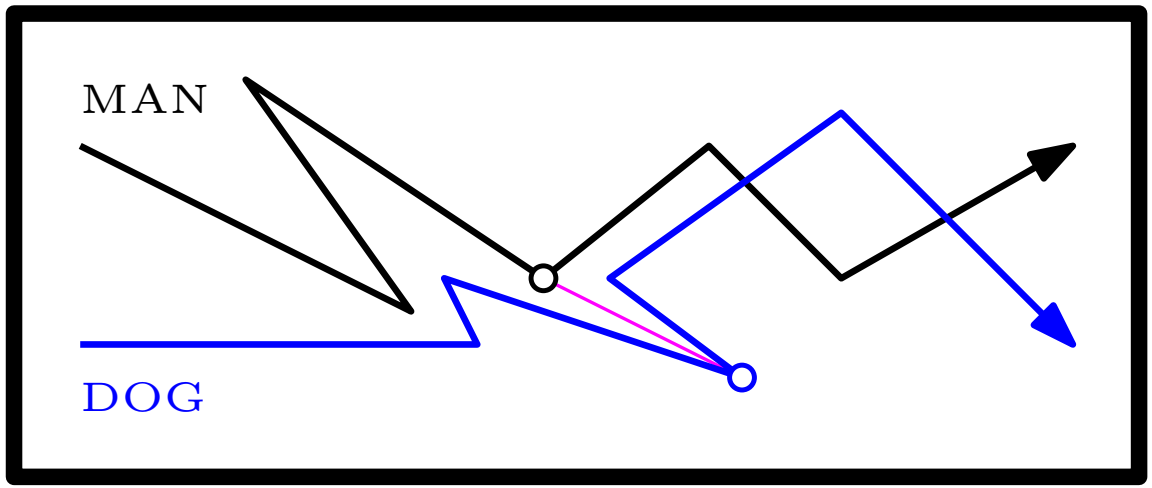
Fréchet Distance for Curves Example

- A man and a dog walk along their assigned curves.
- They are connected by a leash.
- They can control their speeds but cannot go backwards.
- The minimum leash required for any possible walk is the Fréchet distance of the curves.



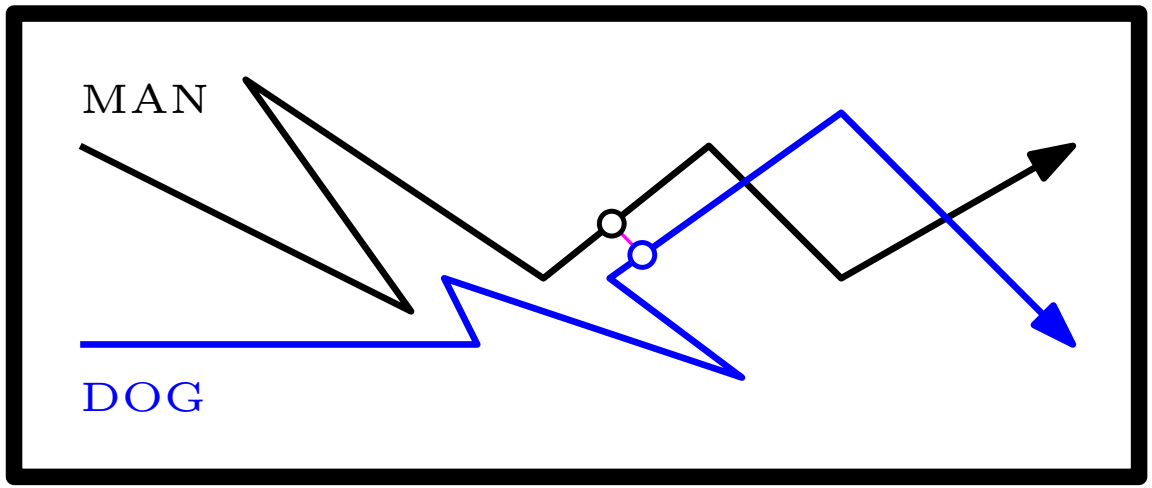
Fréchet Distance for Curves Example

- A man and a dog walk along their assigned curves.
- They are connected by a leash.
- They can control their speeds but cannot go backwards.
- The minimum leash required for any possible walk is the Fréchet distance of the curves.



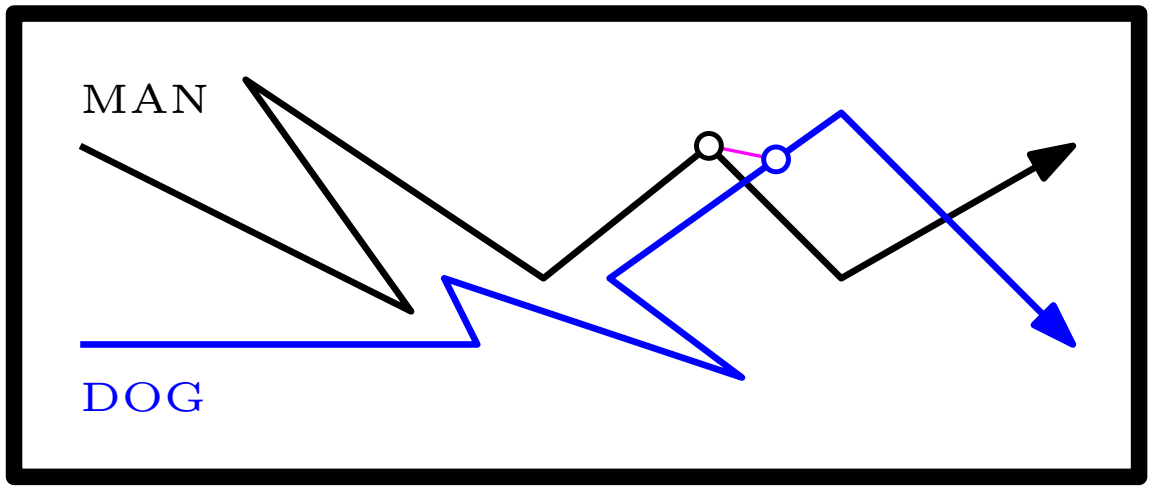
Fréchet Distance for Curves Example

- A man and a dog walk along their assigned curves.
- They are connected by a leash.
- They can control their speeds but cannot go backwards.
- The minimum leash required for any possible walk is the Fréchet distance of the curves.



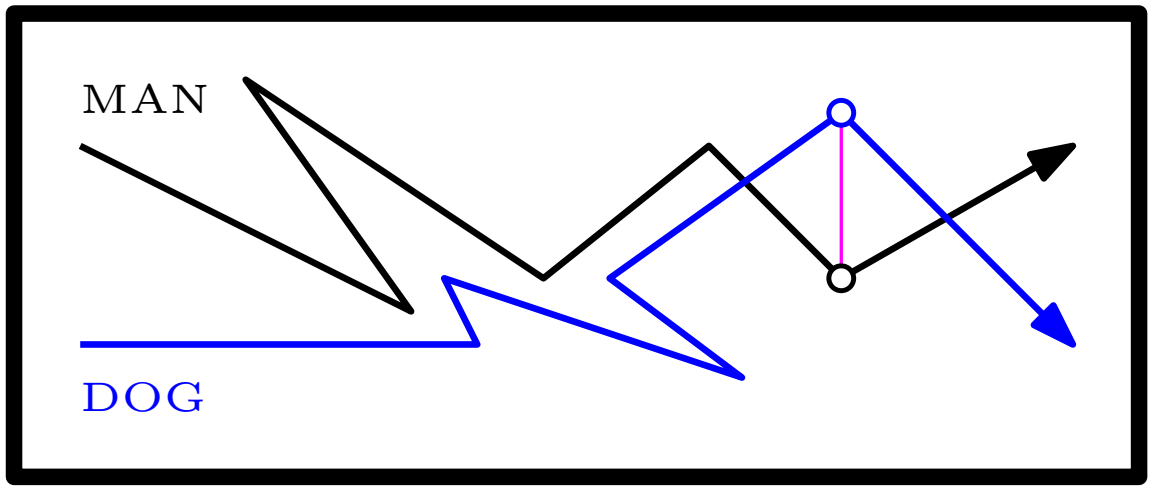
Fréchet Distance for Curves Example

- A man and a dog walk along their assigned curves.
- They are connected by a leash.
- They can control their speeds but cannot go backwards.
- The minimum leash required for any possible walk is the Fréchet distance of the curves.



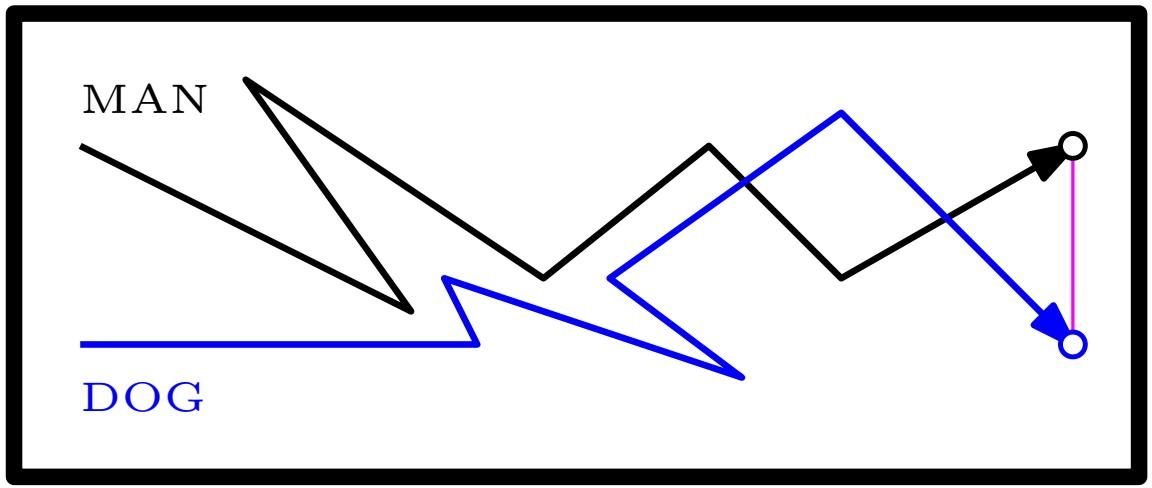
Fréchet Distance for Curves Example

- A man and a dog walk along their assigned curves.
- They are connected by a leash.
- They can control their speeds but cannot go backwards.
- The minimum leash required for any possible walk is the Fréchet distance of the curves.



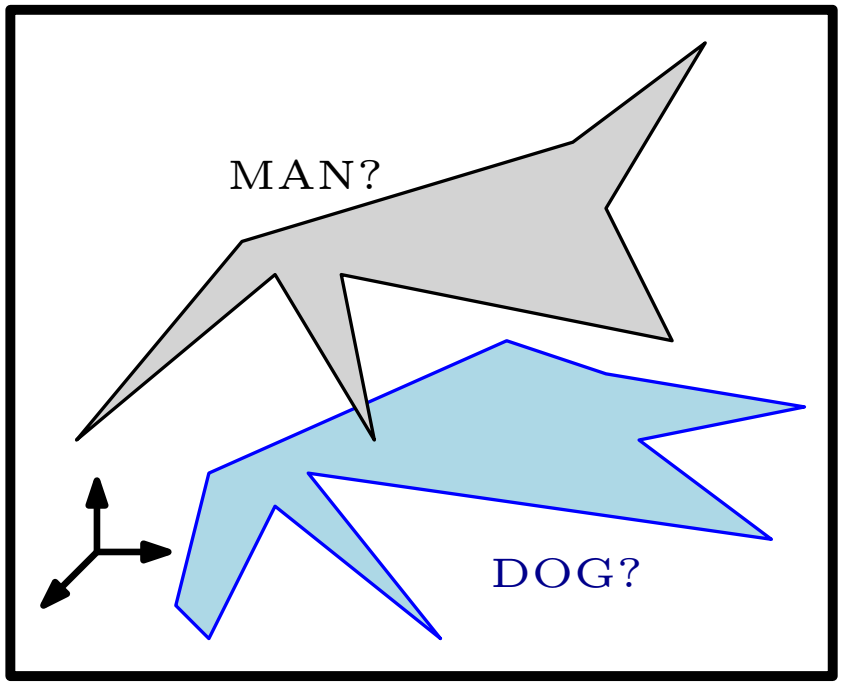
Fréchet Distance for Curves Example

- A man and a dog walk along their assigned curves.
- They are connected by a leash.
- They can control their speeds but cannot go backwards.
- The minimum leash required for any possible walk is the Fréchet distance of the curves.



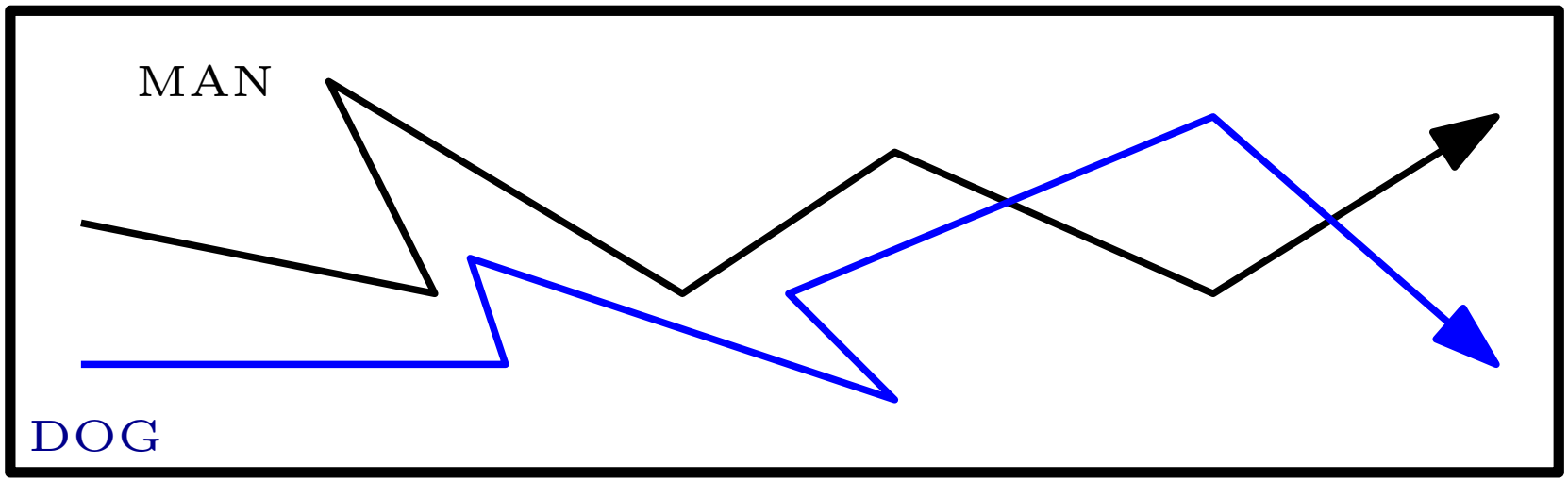
Fréchet Distance between Surfaces

- This analogy breaks down for surfaces.



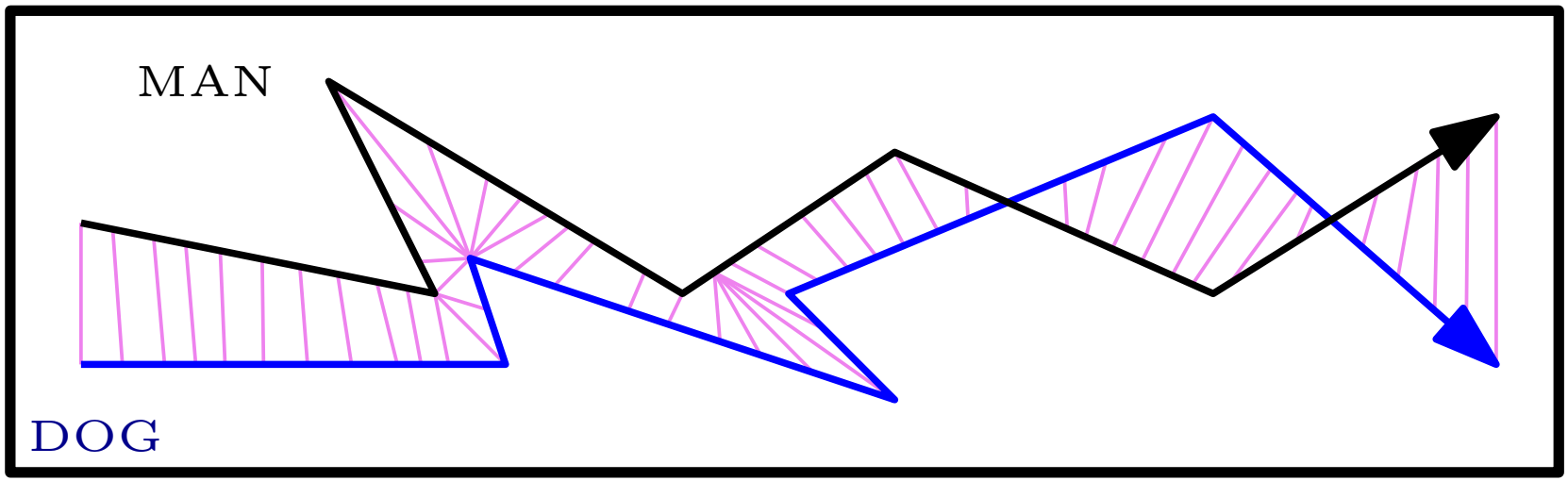
Fréchet Distance Idea

- Consider the curves case again.



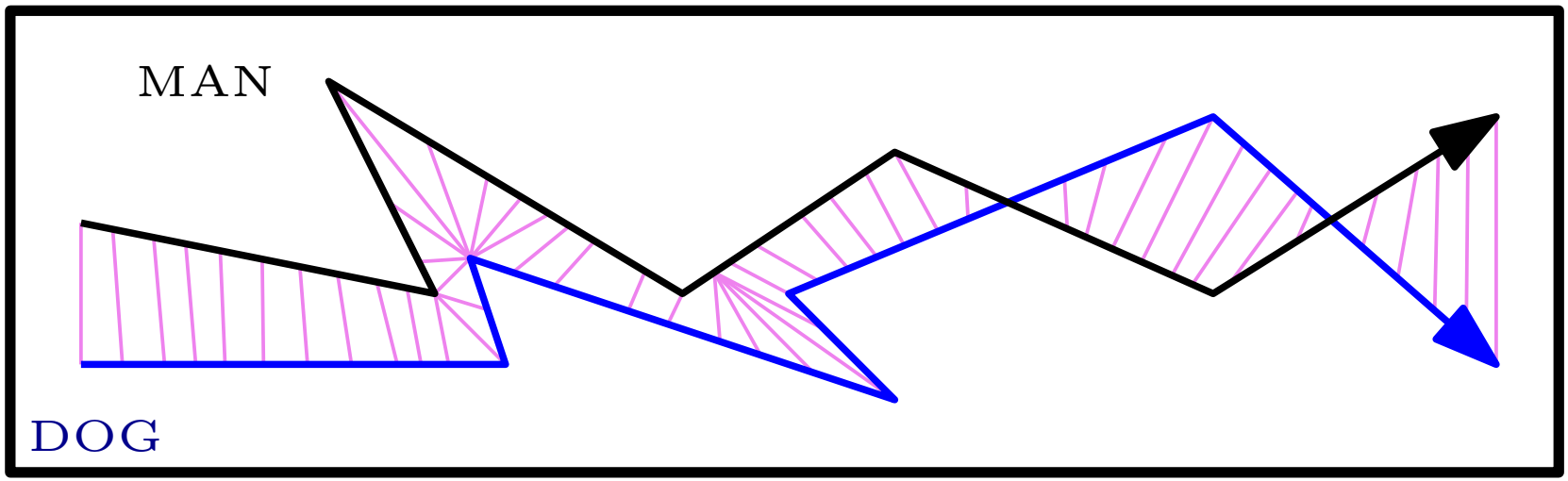
Fréchet Distance Idea

- Consider the curves case again.
- The points connected by leashes for a given walk correspond to a 'morphing' between the curves.



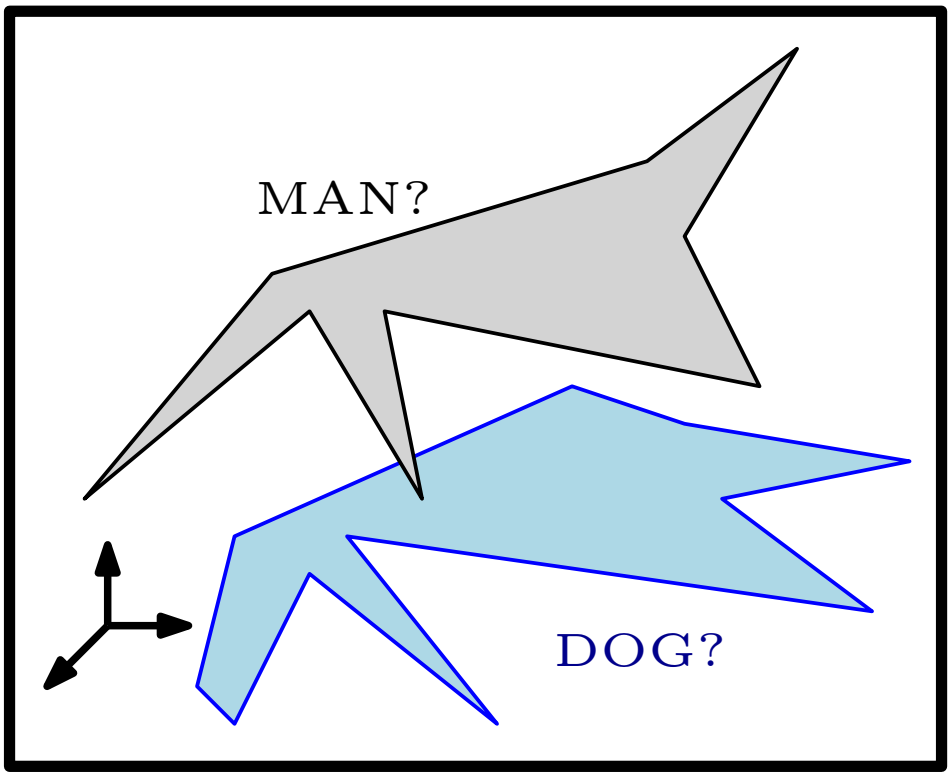
Fréchet Distance Idea

- Consider the curves case again.
- The points connected by leashes for a given walk correspond to a 'morphing' between the curves.
- Intuitively we're trying to find a morphing between the curves which minimizes the maximum distance any point is morphed. This distance is the Fréchet distance of the curves.



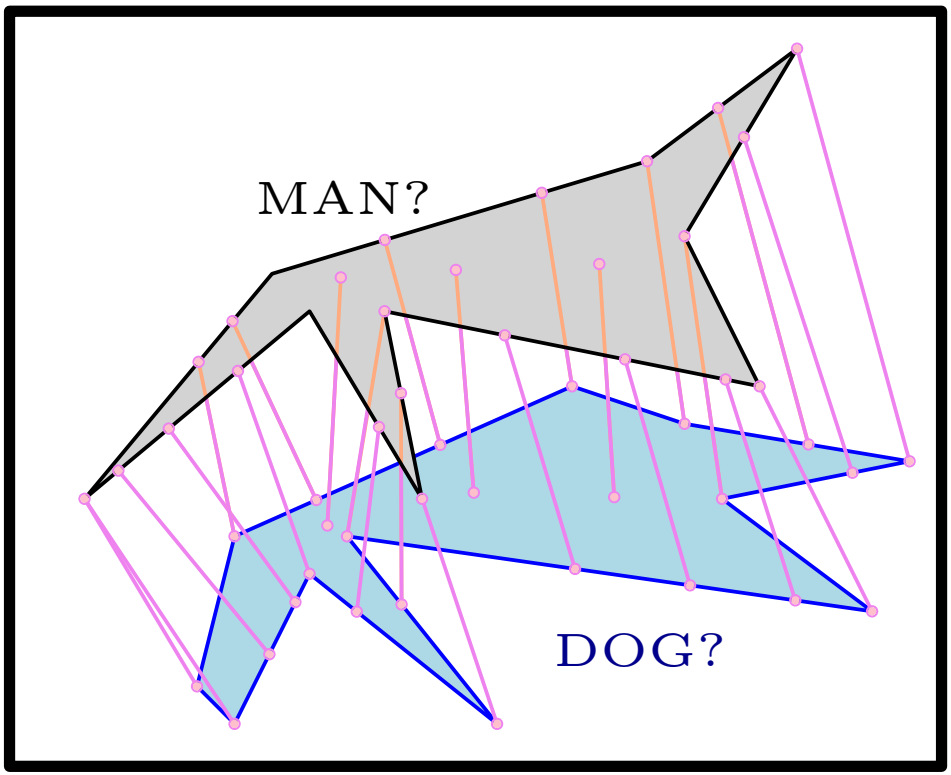
Fréchet Distance between Surfaces Idea

- Likewise, in the case of surfaces one can consider a morphing between them.



Fréchet Distance between Surfaces Idea

- Likewise, in the case of surfaces one can consider a morphing between them.



Fréchet Distance Definition

Fréchet Distance Definition

- $\delta_F(P, Q) = \inf_{\sigma:\, P \to Q} \sup_{p \in P} \|p - \sigma(p)\|$

Fréchet Distance Definition

Fréchet Distance Definition

- $\delta_F(P, Q) = \inf_{\sigma:\, P \to Q} \sup_{p \in P} \|p - \sigma(p)\|$

- $\|\cdot\|$ is the Euclidean norm

Fréchet Distance Definition

Fréchet Distance Definition

- $\delta_F(P, Q) = \inf_{\sigma:\, P \to Q} \sup_{p \in P} \|p - \sigma(p)\|$

- $\|\cdot\|$ is the Euclidean norm
- σ (sigma) ranges over orientation-preserving homeomorphisms (our 'mapping') that map each point $p \in P$ to an image point $q = \sigma(p) \in Q$

Fréchet Distance Definition

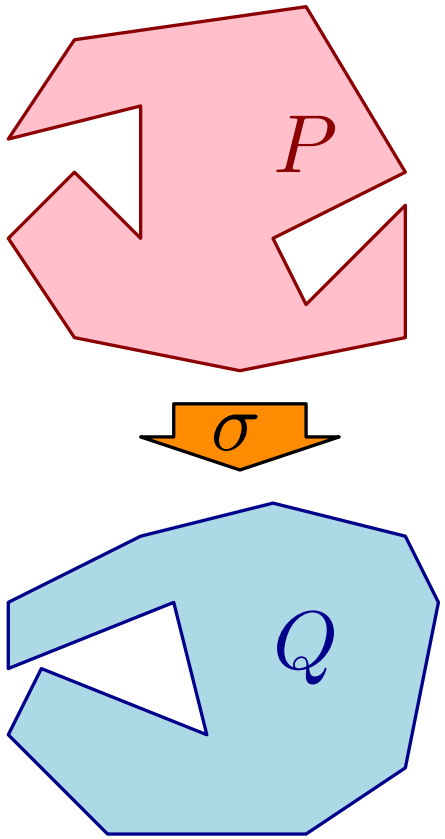
Fréchet Distance Definition

- $\delta_F(P, Q) = \inf_{\sigma:\, P \to Q} \sup_{p \in P} \|p - \sigma(p)\|$

- $\|\cdot\|$ is the Euclidean norm
- σ (sigma) ranges over orientation-preserving homeomorphisms (our 'mapping') that map each point $p \in P$ to an image point $q = \sigma(p) \in Q$
- Each σ corresponds to some walk. For each σ, $\sup_{p \in P} \|p - \sigma(p)\|$ corresponds to the leash length. The Fréchet distance is the minimum (infimum) leash length across all possible σ.

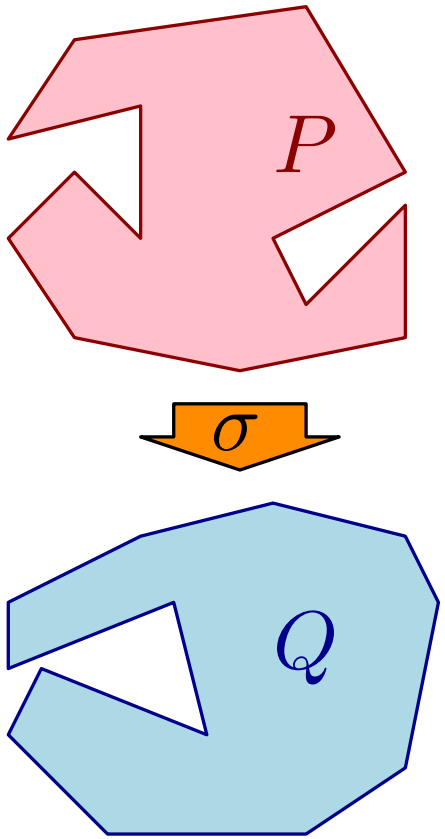
Many Homeomorphisms

- So, now that we know what it is, how do you compute the Fréchet distance of a pair of surfaces?



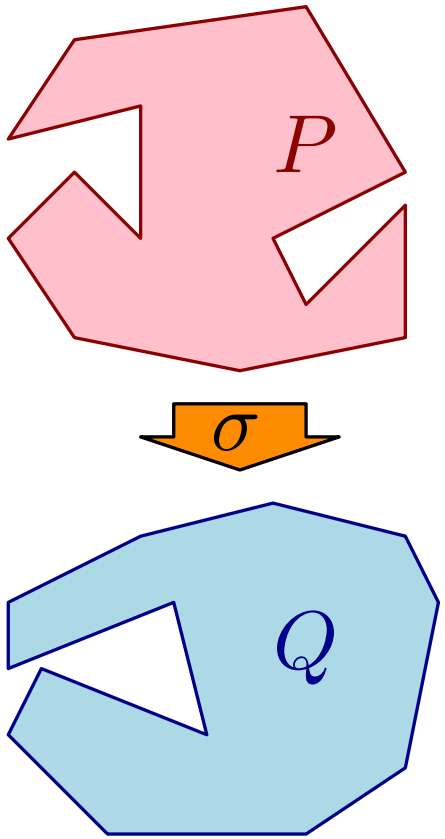
Many Homeomorphisms

- So, now that we know what it is, how do you compute the Fréchet distance of a pair of surfaces?
- Let us first consider the case where the surfaces are simple polygons (flat).



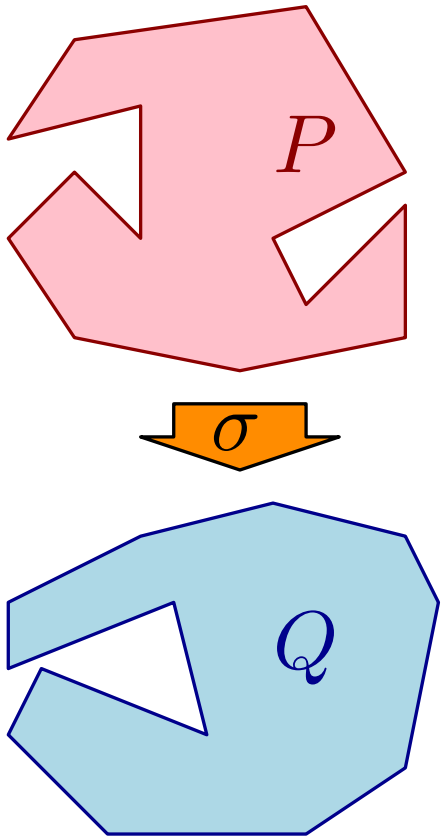
Many Homeomorphisms

- So, now that we know what it is, how do you compute the Fréchet distance of a pair of surfaces?
- Let us first consider the case where the surfaces are simple polygons (flat).
- There are an infinite number of homeomorphisms between a pair of simple polygons.



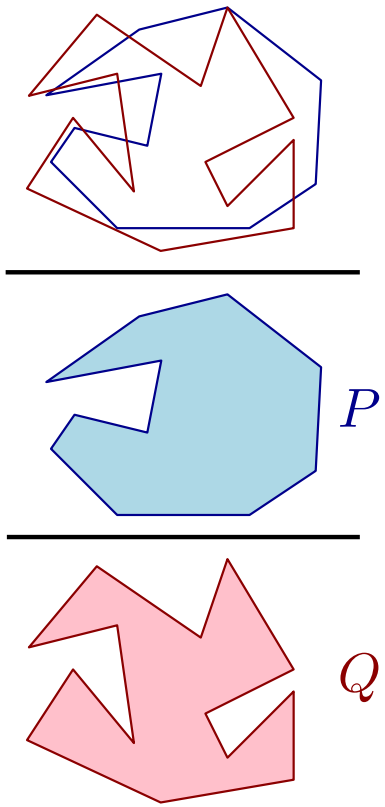
Many Homeomorphisms

- So, now that we know what it is, how do you compute the Fréchet distance of a pair of surfaces?
- Let us first consider the case where the surfaces are simple polygons (flat).
- There are an infinite number of homeomorphisms between a pair of simple polygons.
- To efficiently compute their Fréchet distance this search space must be reduced.



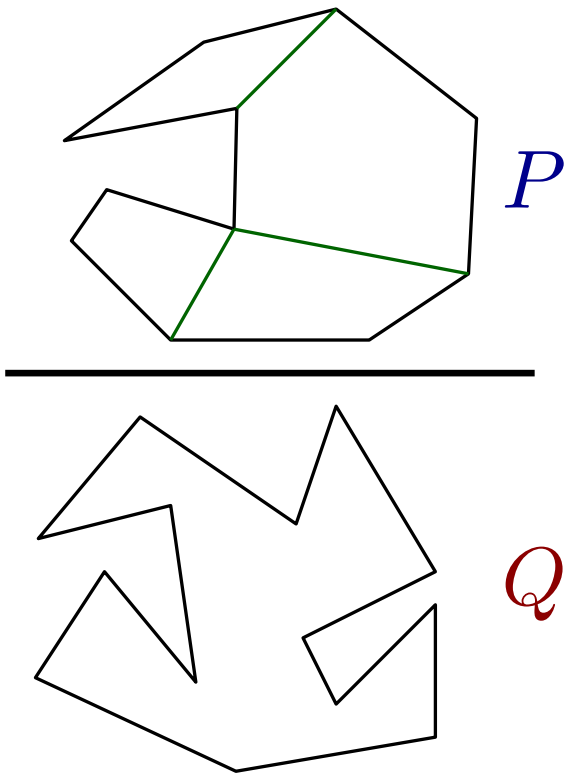
Simple Polygons Example

- Consider this pair of polygons.



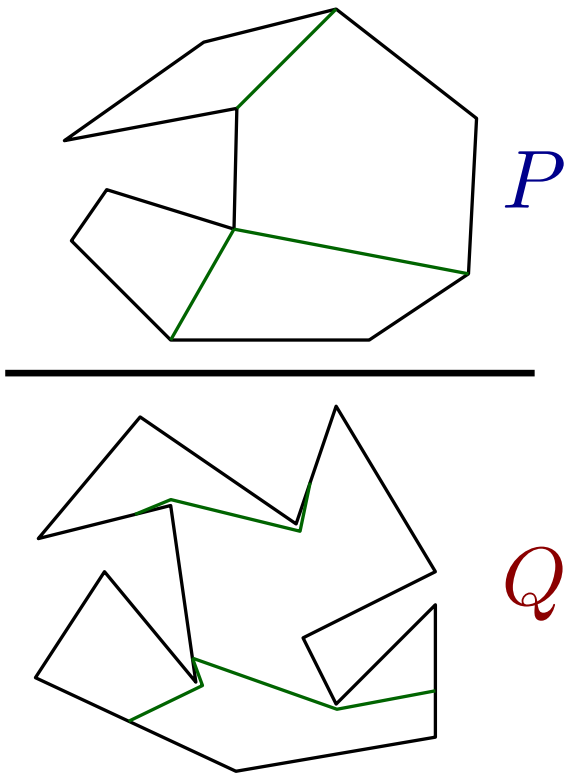
Simple Polygons Example

- Consider this pair of polygons.
- The idea is that P is subdivided into convex portions and each of these are mapped over to Q.



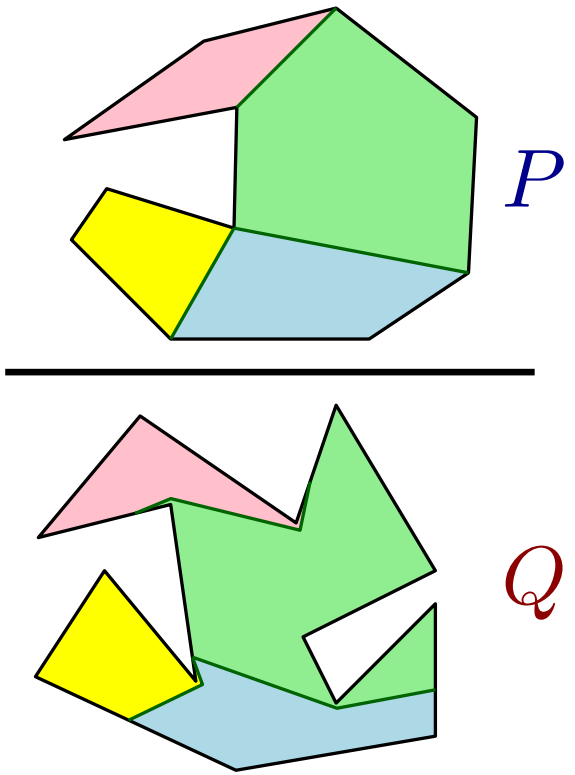
Simple Polygons Example

- Consider this pair of polygons.
- The idea is that P is subdivided into convex portions and each of these are mapped over to Q.
- For flat surfaces it suffices to map the edges used to subdivide P over to paths in Q.



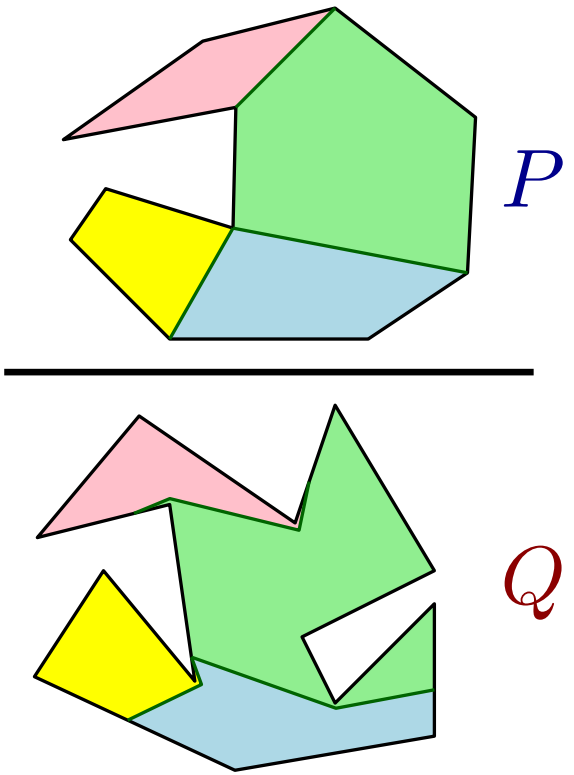
Simple Polygons Example

- Consider this pair of polygons.
- The idea is that P is subdivided into convex portions and each of these are mapped over to Q.
- For flat surfaces it suffices to map the edges used to subdivide P over to paths in Q.
- These mapped paths subdivide Q into portions which match with those in P.



Simple Polygons Example

- Consider this pair of polygons.
- The idea is that P is subdivided into convex portions and each of these are mapped over to Q.
- For flat surfaces it suffices to map the edges used to subdivide P over to paths in Q.
- These mapped paths subdivide Q into portions which match with those in P.
- This is the rough idea of how the Fréchet distance is computed for simple polygons.

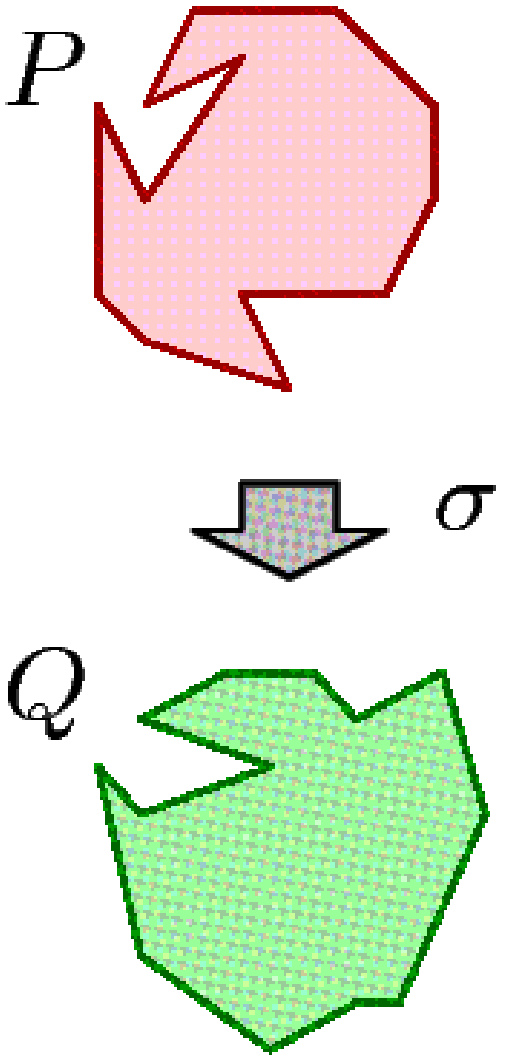


Problem Review

- For simple polygons the FD is polynomial time computable.
- For terrains and polygons with holes the FD is NP-hard to compute.
- There is a gap between these known results.
- Are there more general classes of surfaces for which the Fréchet distance can be computed in polynomial time?
- We first review the simple polygons algorithm and then consider extending it to a more general class of surfaces which we call folded polygons.

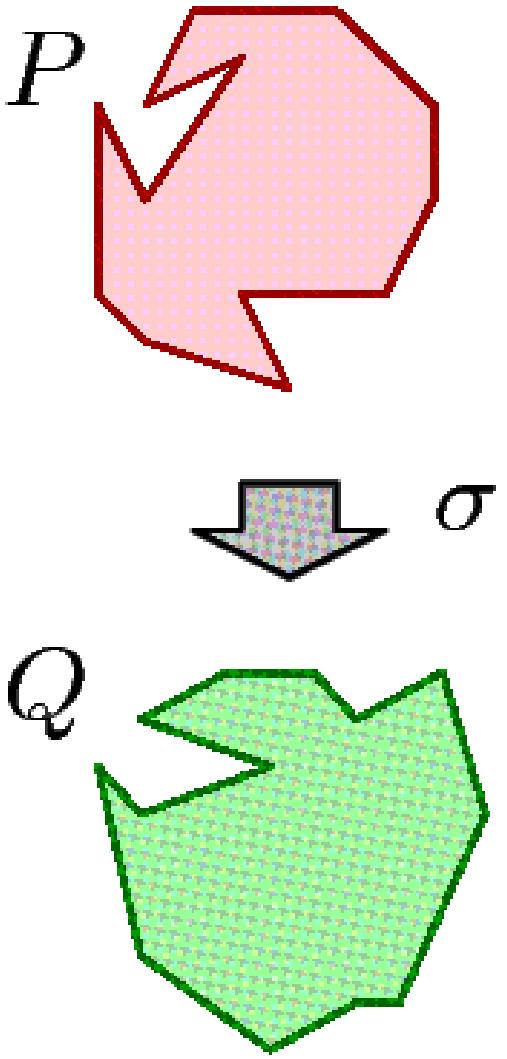
Simple Polygons Algorithm [BBW06]

- Next, we give a short summary of the simple polygons algorithm.



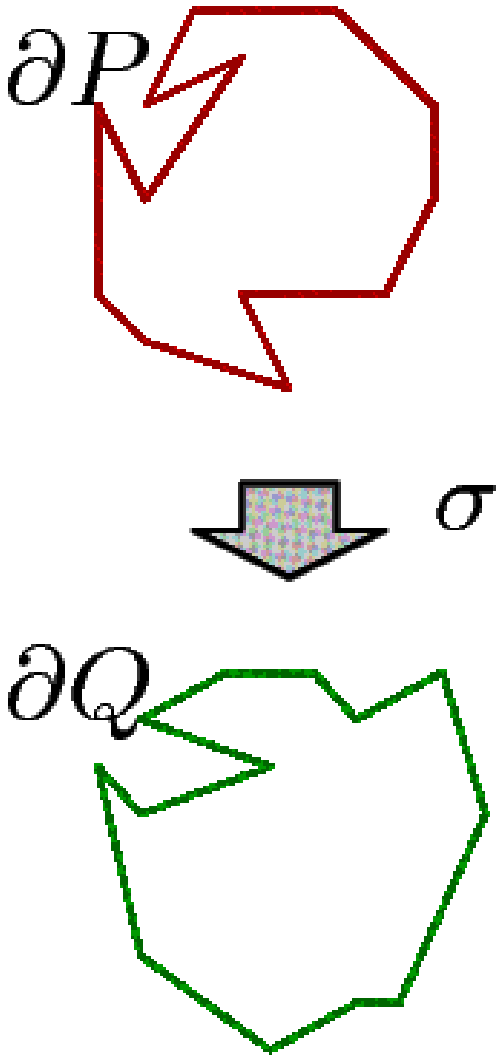
Simple Polygons Algorithm [BBW06]

- There are an infinite number of homeomorphisms between a pair of simple polygons.
- To efficiently compute their Fréchet distance the this search space must be reduced.



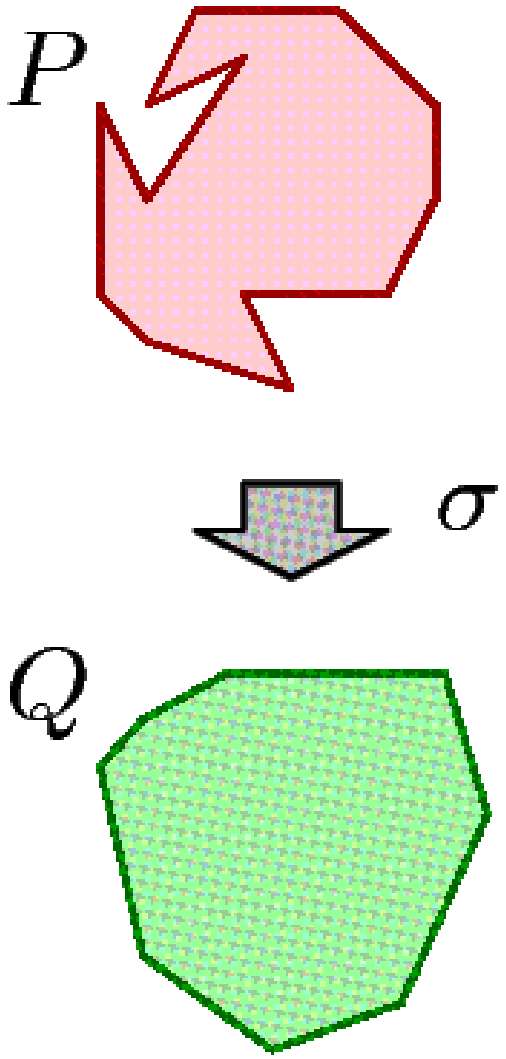
Simple Polygons Algorithm [BBW06]

- One could first consider the boundaries of the polygons.
- The Fréchet distance between closed polygonal curves can be computed in polynomial time.
- The boundaries of two simple polygons can be compared with this.

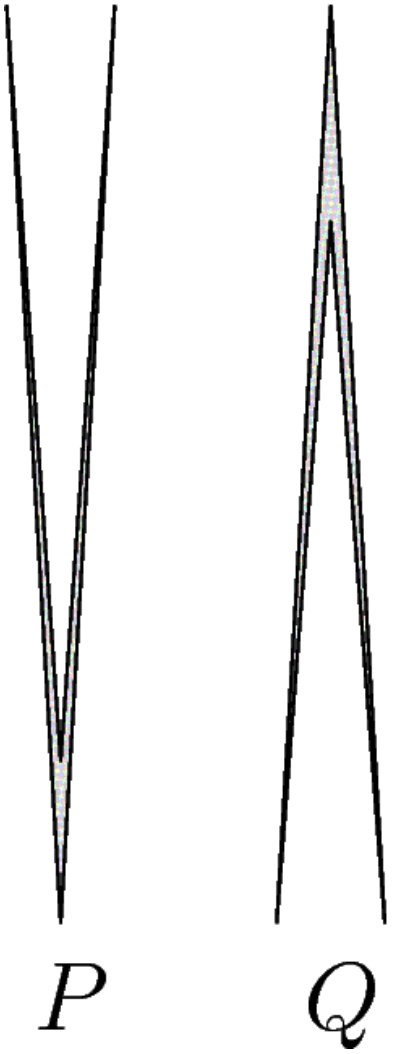


Simple Polygons Algorithm [BBW06]

- The authors prove that the Fréchet distance between a convex polygon and a simple polygon is the same as that between their boundary curves.



Simple Polygons Algorithm [BBW06]



- Unfortunately, this is not always the case between two simple polygons.

Simple Polygons Algorithm [BBW06]

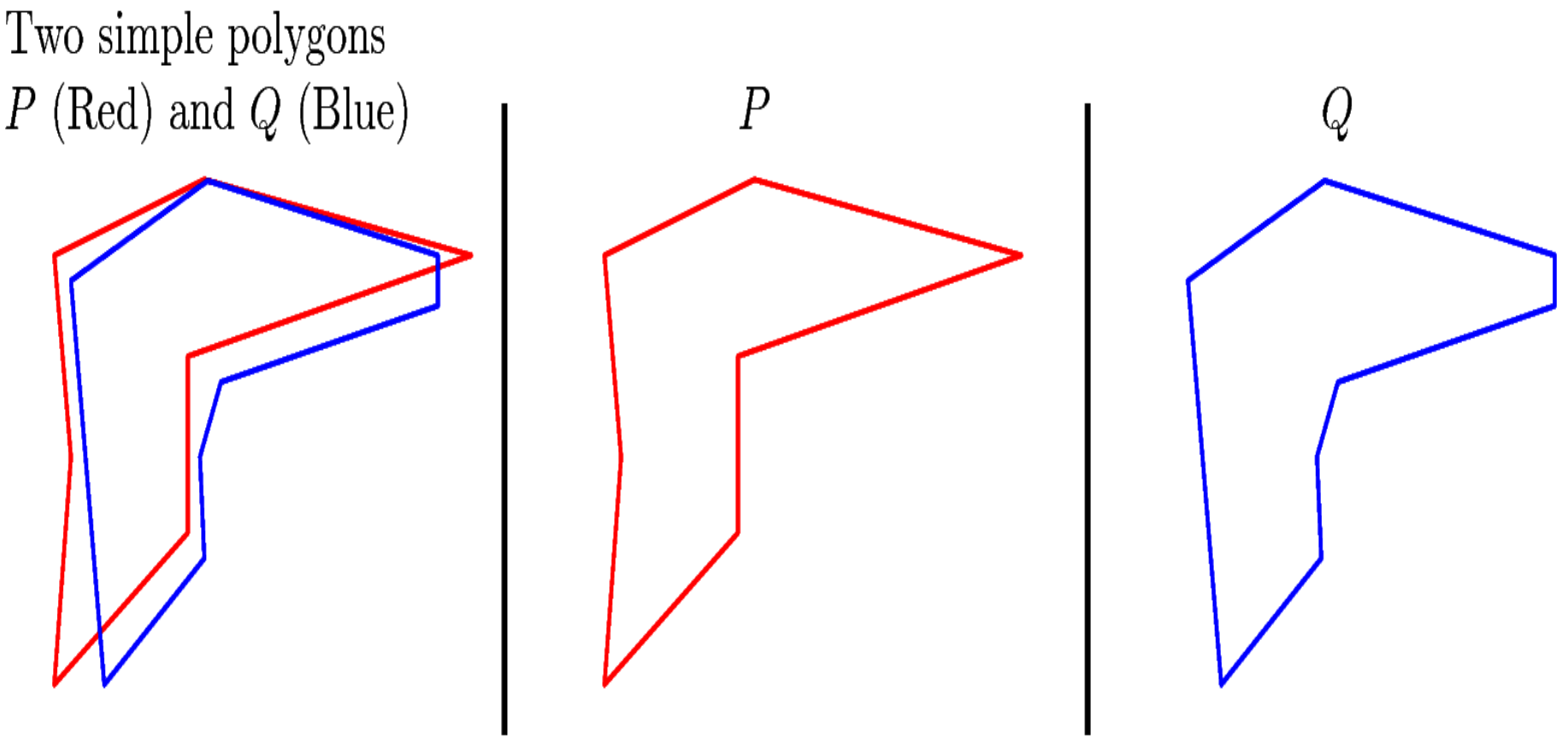
- **Idea: Restrict the class of mappings to consider**
  - Given two simple polygons P and Q
  - Divide them up into matched pairs of convex polygons and simple polygons.
  - Then use the closed polygonal curves algorithm mentioned before to check whether the distance is within some ε .

Simple Polygons Algorithm [BBW06]

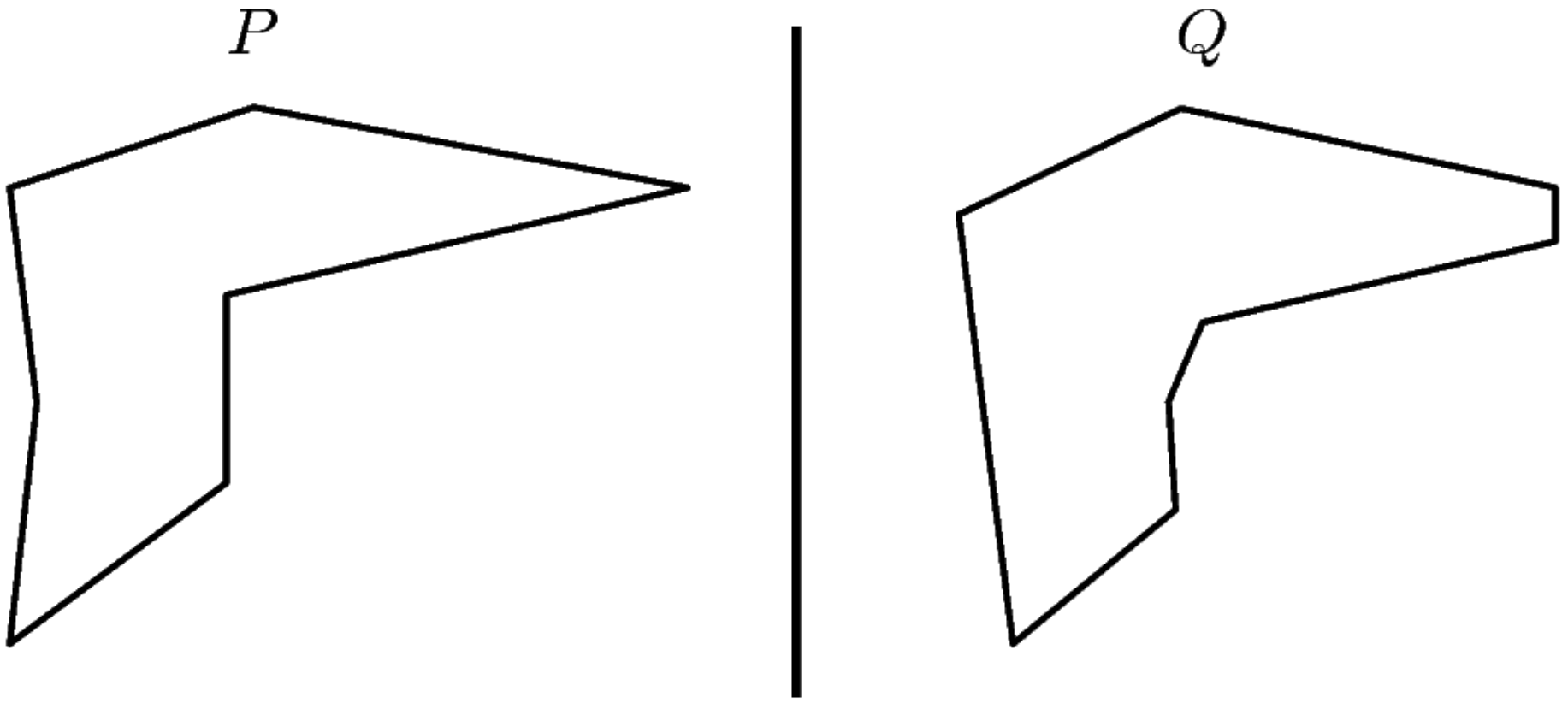
- “diagonals” in P are the line segments in a convex subdivision of P
- “edges” in Q are the line segments in a convex subdivision of Q
- Map the diagonals in a convex subdivision of P to image curves in Q.
  - [BBW06] demonstrate that it suffices to map diagonals to shortest paths in Q.

  $\rightarrow$ Only consider a restricted class of mappings

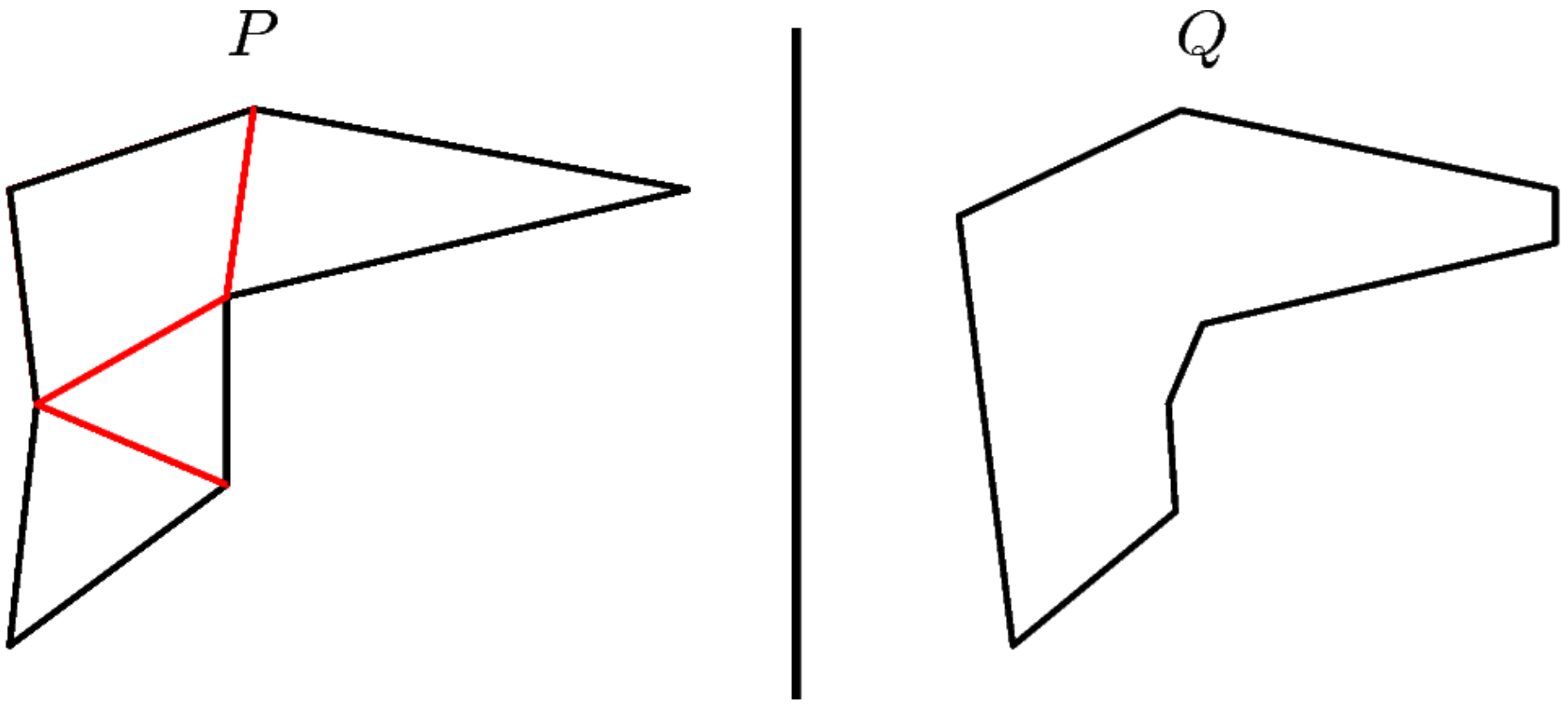
Simple Polygons Algorithm [BBW06]: Example



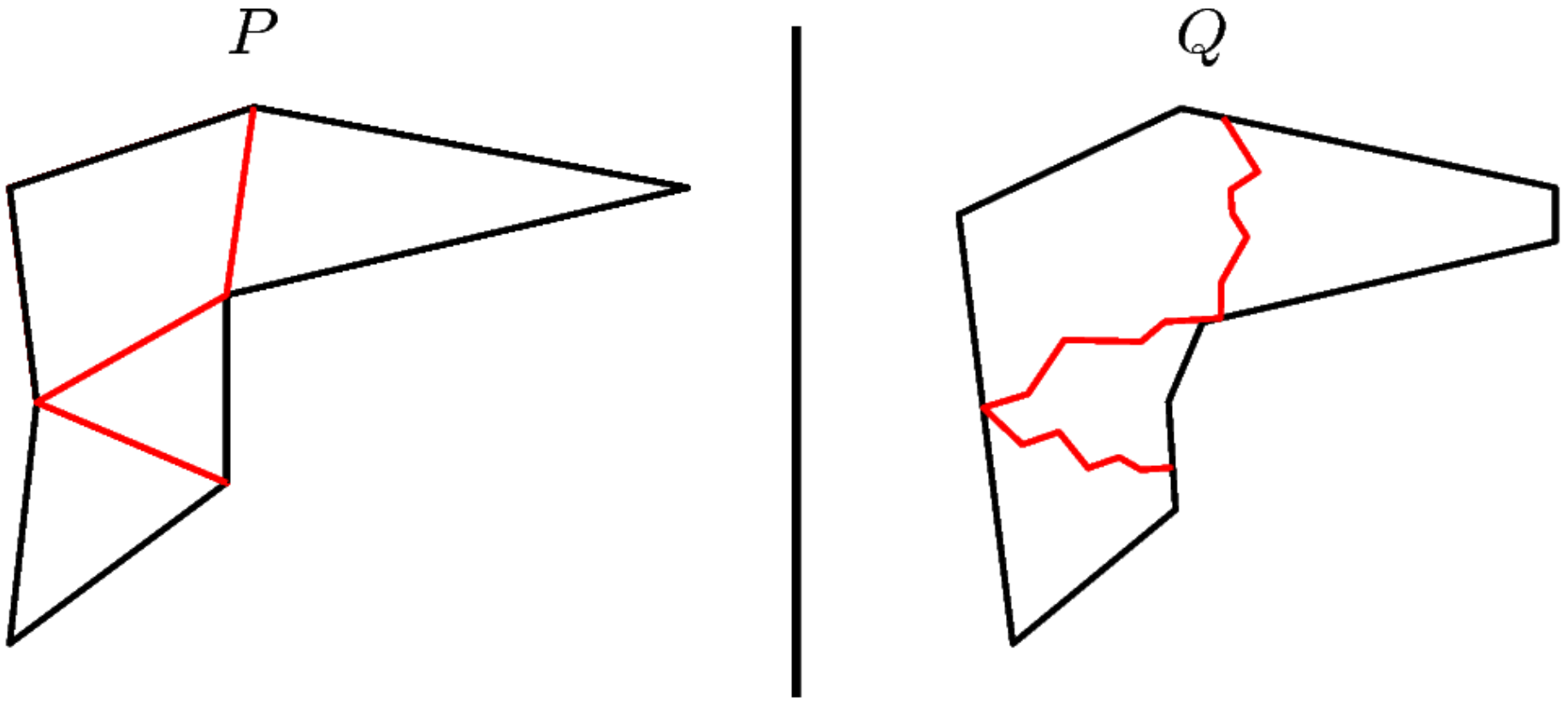
Simple Polygons Algorithm [BBW06]: Example



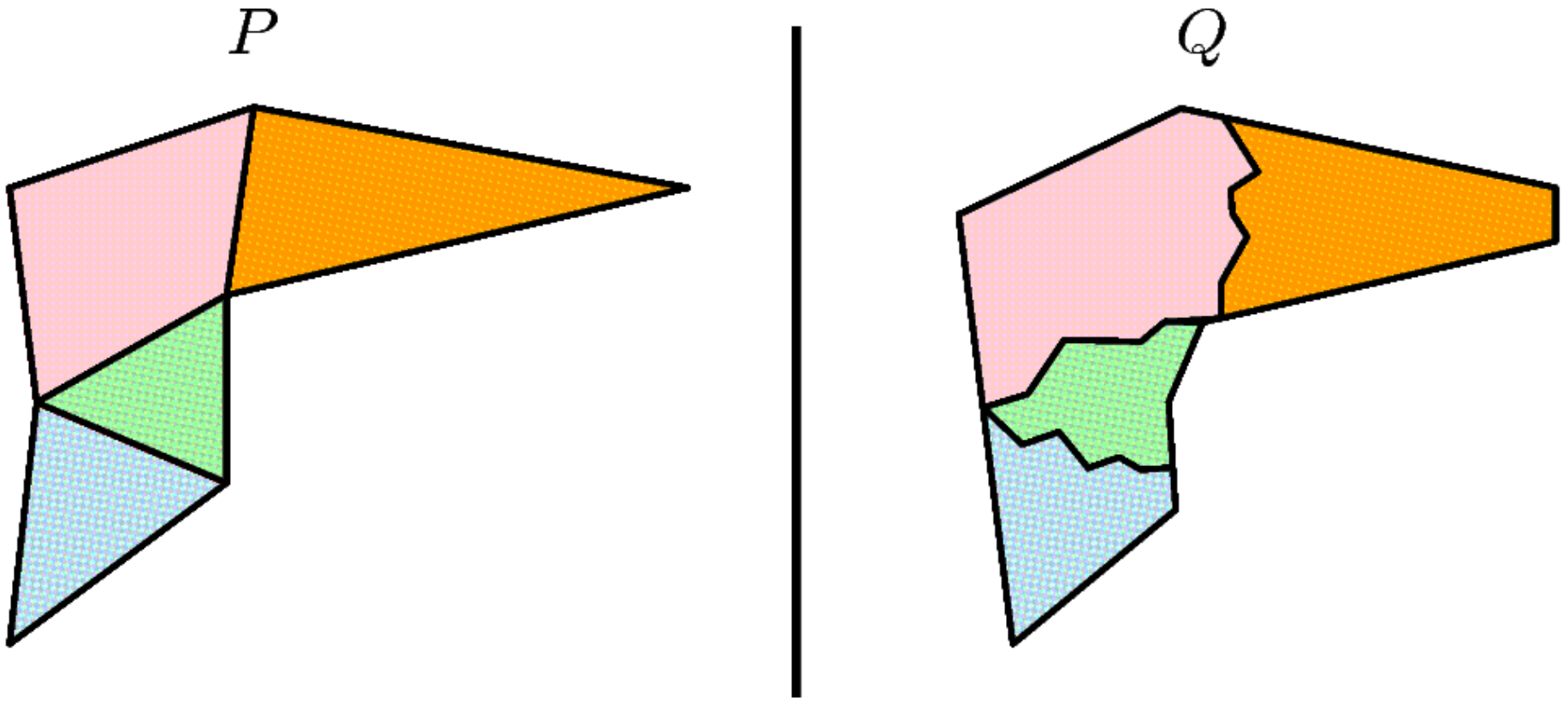
Simple Polygons Algorithm [BBW06]: Example



Simple Polygons Algorithm [BBW06]: Example



Simple Polygons Algorithm [BBW06]: Example



Simple Polygons Algorithm [BBW06]

- The authors show that it is sufficient to test the following two things to find if a homeomorphism exists between the surfaces for distance ε .

  Simple Polygons

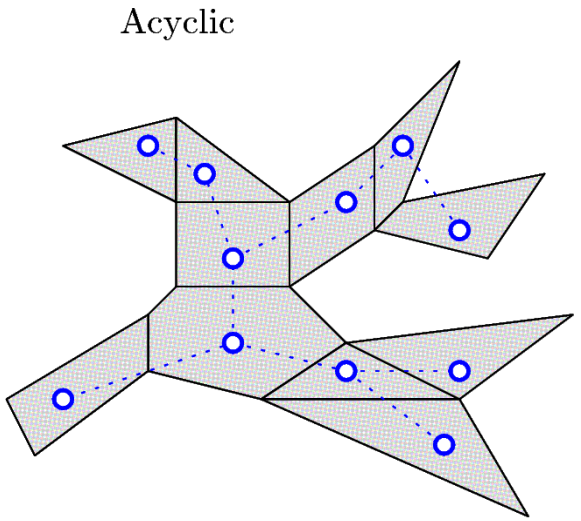
  - $\delta_F(\partial P, \partial Q) \leq \varepsilon$ (this specifies a mapping of the diagonal endpoints)
  - For every diagonal in P, the corresponding shortest path in Q has Fréchet distance distance at most ε to it.

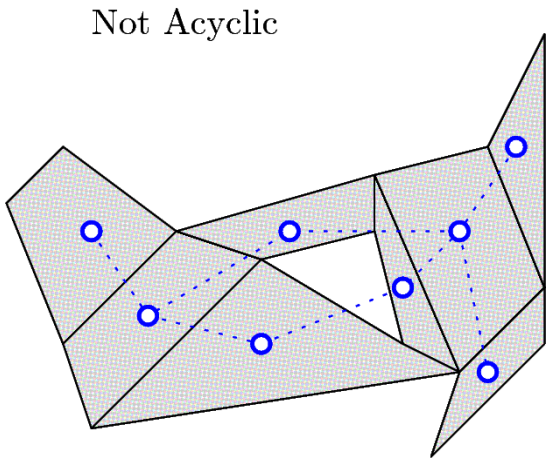
Simple Polygons Algorithm [BBW06]: Run Time

- n = the complexity of Q
- m = the complexity of P
- k = the number of diagonals in P
- T(N) = the time to multiply two NxN matrices

The Fréchet distance of two simple polygons can be computed in time $O(kT_{matrixmult}(mn)\log(mn))$.

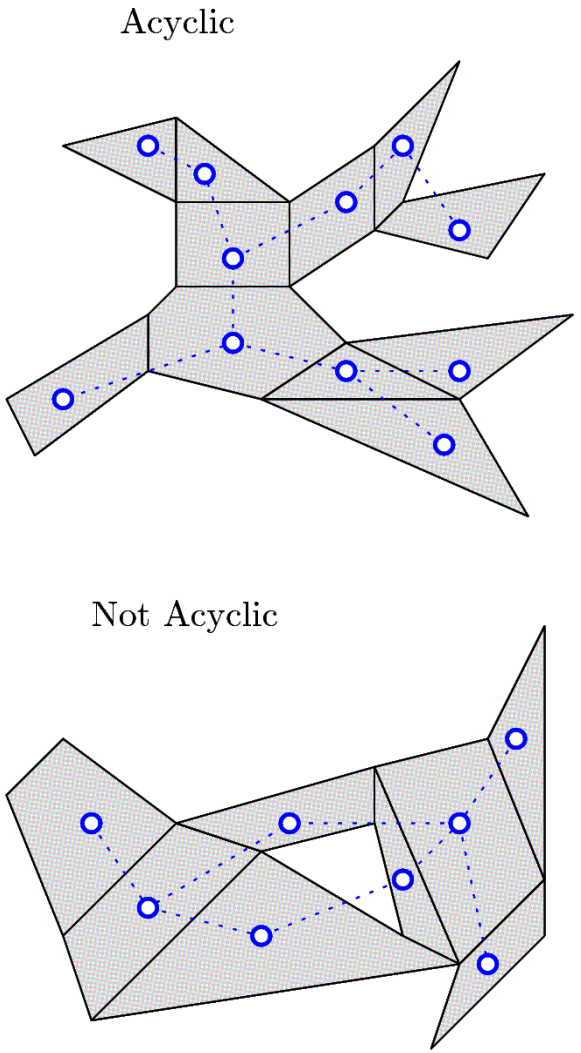
Folded Polygons





- We extend the simple polygons algorithm to non-flat surfaces.
- Specifically, we consider piecewise linear surfaces with a convex subdivision which has an acyclic dual graph (“folded polygons”)

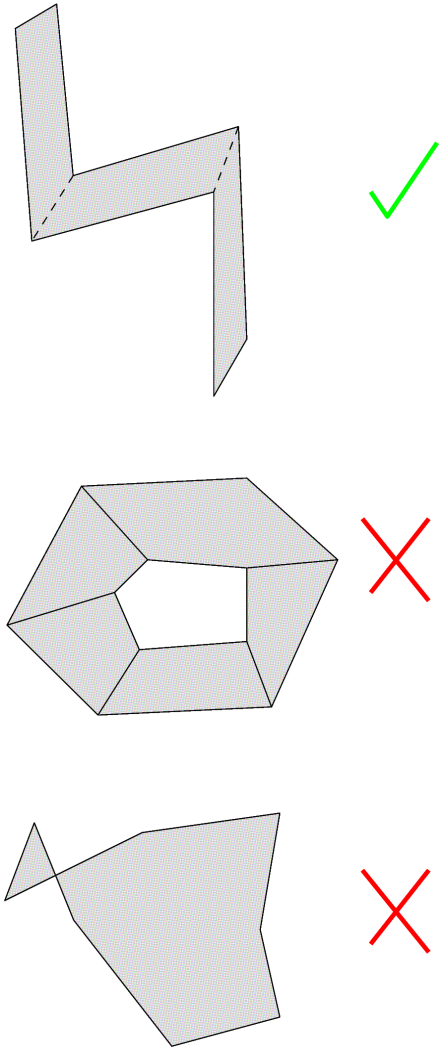
Folded Polygons



- We extend this algorithm to non-flat surfaces.
- Specifically, we consider piecewise linear surfaces with a convex subdivision which has an acyclic dual graph (“folded polygons”)
- No interior vertices.

Folded Polygons

- Features:
  - Folds only along line segments.
  - No holes.
  - No self-intersection.



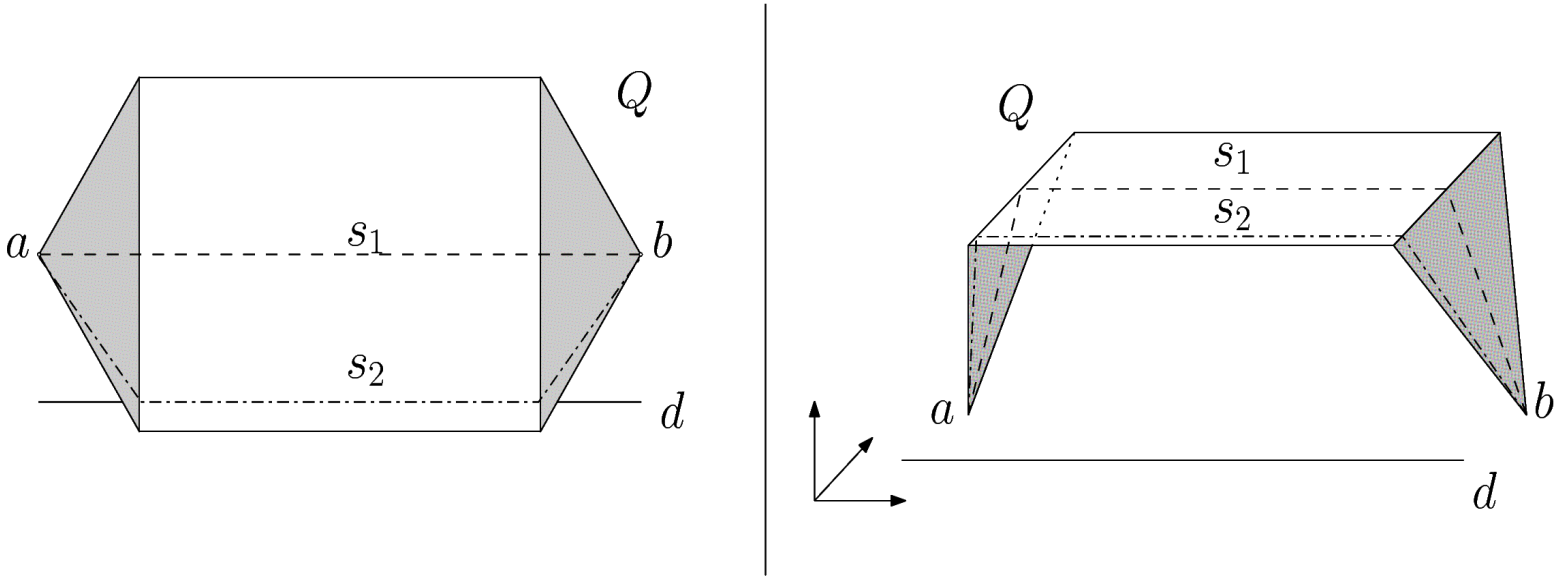
Simple Polygons Algorithm?

- Can we just use the simple polygons algorithm?

Simple Polygons Algorithm?

- Can we just use the simple polygons algorithm?
- Unfortunately, no:
  - The simple polygons algorithm maps diagonals in P to image curves which are shortest paths in the Q.
  - When P is a folded polygon instead of a simple polygon, we can find examples where the Fréchet distance between the shortest path and the diagonal is not optimal (i.e., some other path with the same end points has smaller FD with the diagonal).

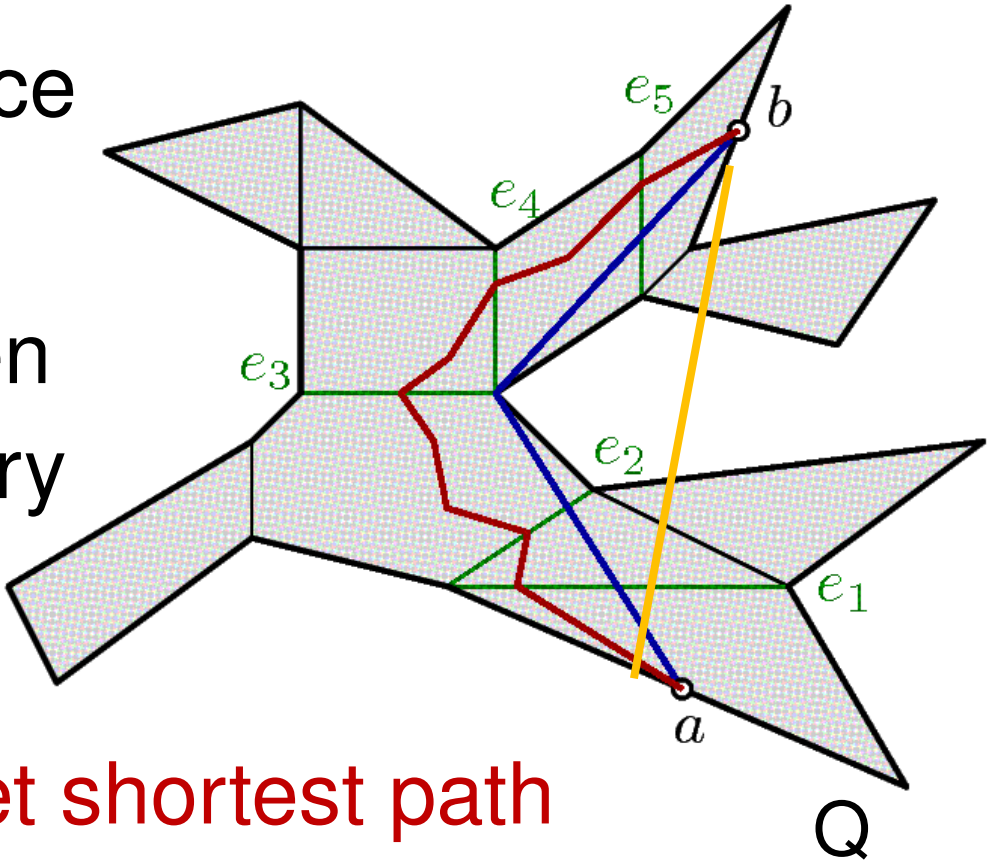
Shortest Path Counter Example



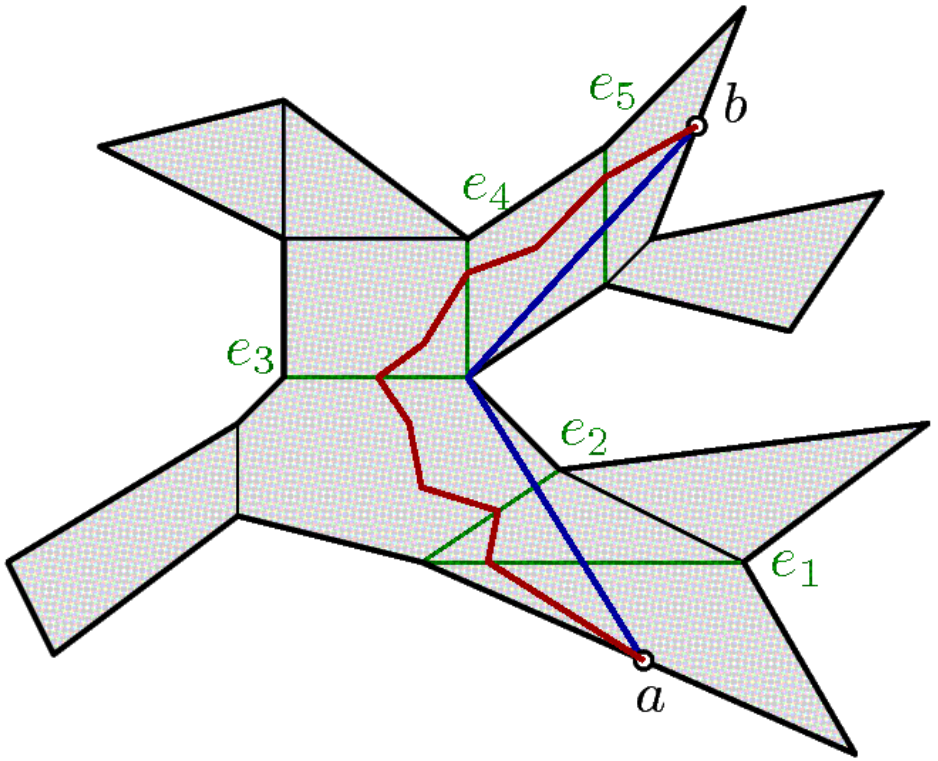
- s_1 is the shortest path between a and b, but the diagonal d has smaller Fréchet distance to s_2 than to s_1.

Fréchet Shortest Paths

- Fréchet shortest paths = paths with Fréchet distance ε to a given diagonal
- The shortest path between two points on the boundary of Q crosses some sequence of edges.
- We prove that any Fréchet shortest path between those points crosses the exact same edge sequence.



Fréchet Shortest Paths



- We can find a Fréchet shortest path between two points on the boundary of Q within Fréchet distance ε of a diagonal in time O(n) if one exists. (n = # edges in Q)
- Same run time as testing for a shortest path in the simple polygons algorithm.

Diagonal Monotonicity Test

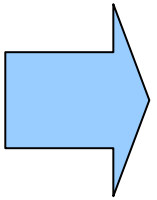
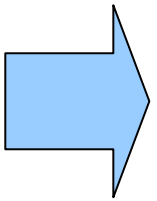
- We modify the original simple polygons algorithm to use this new class of paths.
- P and Q "pass the **diagonal monotonicity test** for ε " iff:

  Simple Polygons

  - $\delta_F(\partial P, \partial Q) \leq \varepsilon$ (this specifies a mapping of the diagonal endpoints)
  - For every diagonal in P, the corresponding shortest path in Q has Fréchet distance distance at most ε to it.

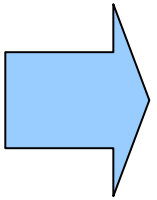
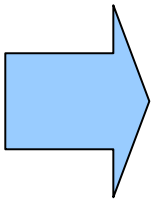
Diagonal Monotonicity Test

- We modify the original simple polygons algorithm to use this new class of paths.
- P and Q "pass the **diagonal monotonicity test** for ε " iff:

Simple Polygons		Folded Polygons
– $\delta_F(\partial P, \partial Q) \leq \varepsilon$ (this specifies a mapping of the diagonal endpoints)		– $\delta_F(\partial P, \partial Q) \leq \varepsilon$ (this specifies a mapping of the diagonal endpoints)
– For every diagonal in P, the corresponding shortest path in Q has Fréchet distance distance at most ε to it.		– For every diagonal in P, a corresponding Fréchet shortest path in Q has Fréchet distance at most ε to it.

Diagonal Monotonicity Test

- We modify the original simple polygons algorithm to use this new class of paths.
- P and Q "pass the **diagonal monotonicity test** for ε " iff:

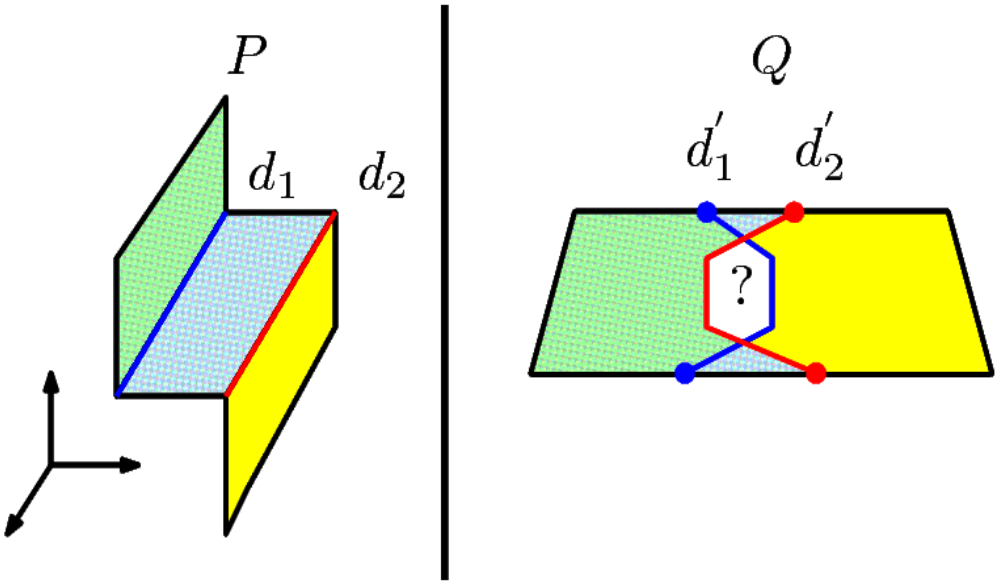
Simple Polygons		Folded Polygons
– $\delta_F(\partial P, \partial Q) \leq \varepsilon$ (this specifies a mapping of the diagonal endpoints)		– $\delta_F(\partial P, \partial Q) \leq \varepsilon$ (this specifies a mapping of the diagonal endpoints)
– For every diagonal in P, the corresponding *shortest path* in Q has Fréchet distance distance at most ε to it.		– For every diagonal in P, a corresponding *Fréchet shortest path* in Q has Fréchet distance at most ε to it.

Problem of Tangled Image Curves

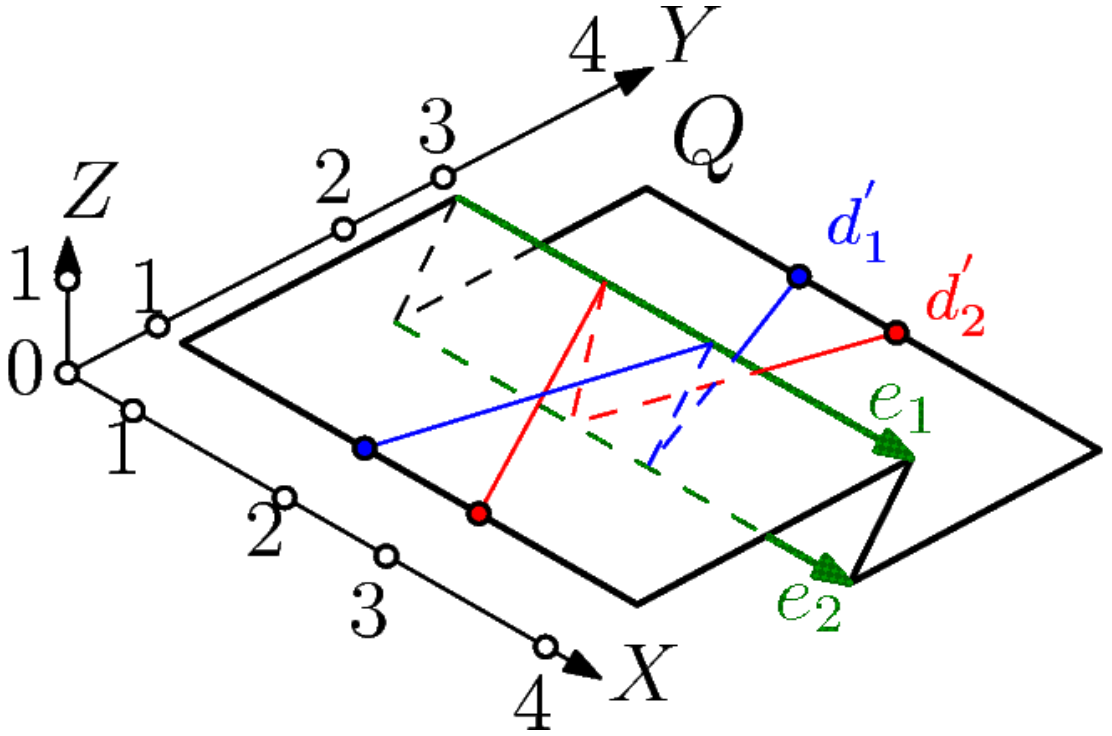
- Because we use Fréchet shortest paths instead of shortest paths we have an additional problem:
  - The image curves may intersect an edge in the convex decomposition in the wrong order.
- We refer to such image curves as being tangled.

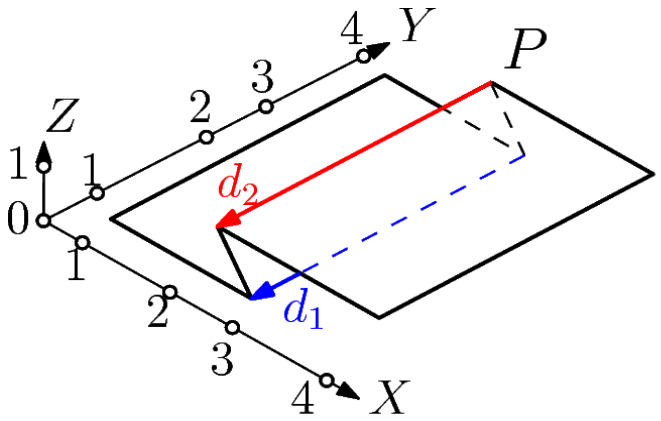
Problem of Tangled Image Curves

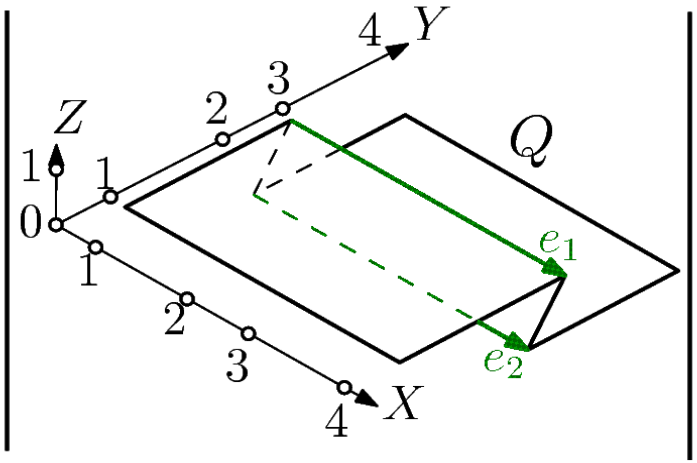
- The image curves cross, thus the subdivision of Q is no longer valid.
- Only a toy example. Can this really happen?

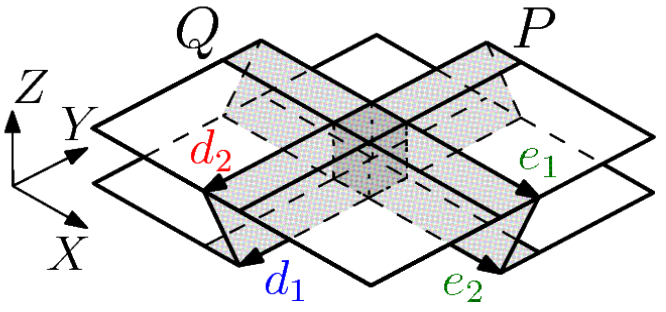


Problem of Tangled Image Curves

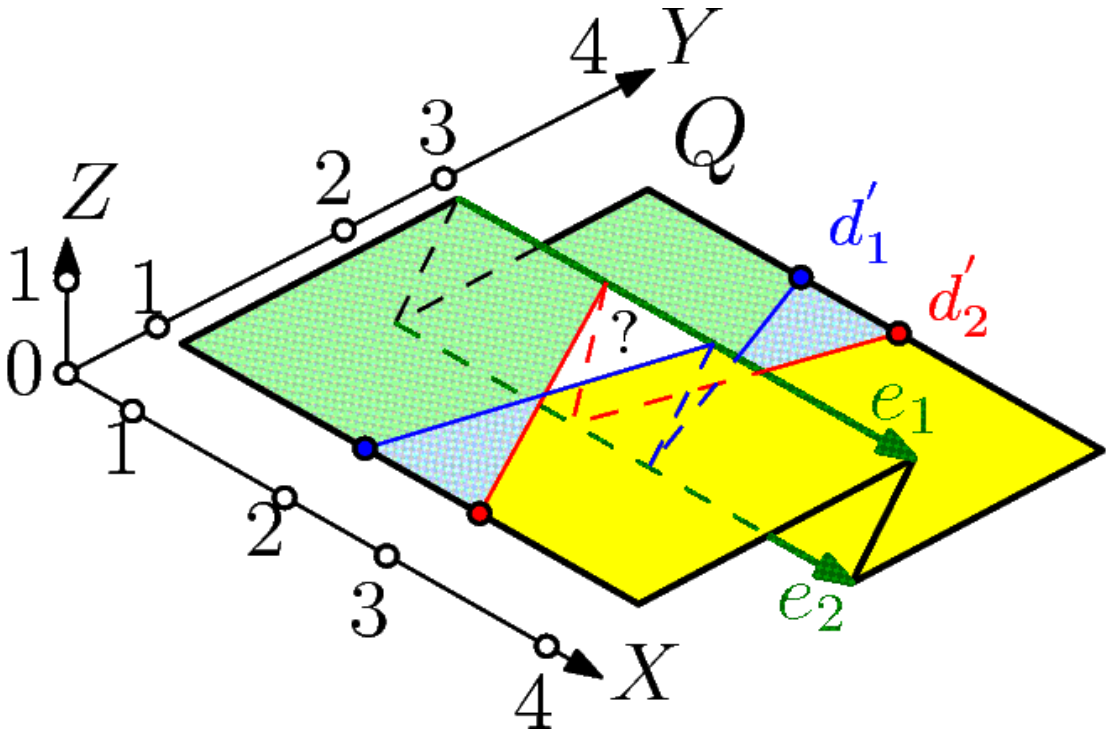


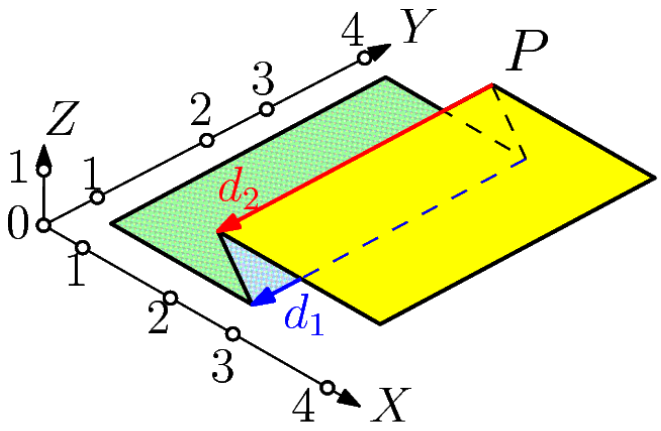


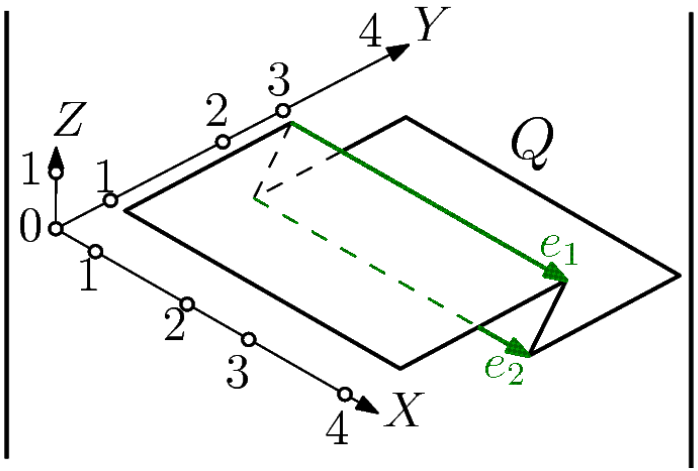


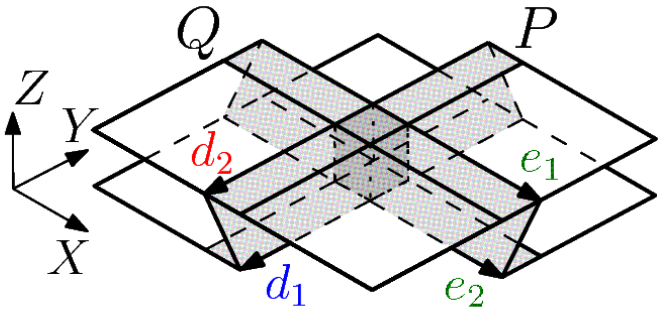


Problem of Tangled Image Curves









Results

- To ensure a homeomorphism exists between the surfaces we must address such tangles. We consider three approaches:
  - Use an approximation algorithm which avoids the tangles altogether. $\rightarrow$poly-time approx. algorithm
  - Compute the constraints posed by such tangles directly.$\rightarrow$fixed parameter tractable algorithm
  - Consider a special non-trivial class of folded polygons for which we can use shortest paths instead of Fréchet shortest paths. $\rightarrow$ poly-time algorithm
    - Axis-aligned surfaces using L_∞ distance metric.

Results

- To ensure a homeomorphism exists between the surfaces we must address such tangles. We consider three approaches:
  - Use an approximation algorithm which avoids the tangles altogether. $\rightarrow$poly-time approx. algorithm
  - Compute the constraints posed by such tangles directly.$\rightarrow$fixed parameter tractable algorithm
  - Consider a special non-trivial class of folded polygons for which we can use shortest paths instead of Fréchet shortest paths. $\rightarrow$ poly-time algorithm
    - Axis-aligned surfaces using L_∞ distance metric.

The run time for all three approaches is the same as that for the simple polygons algorithm (plus an additional exponential factor for the FPT algorithm).

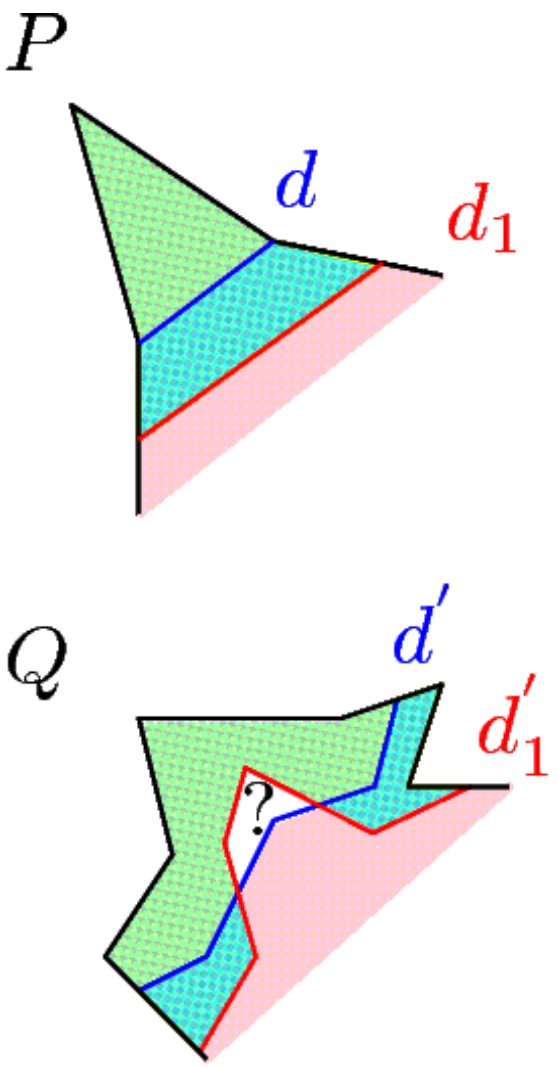
1) Approximation Algorithm

- Key Idea: Approximate away the tangles.

1) Approximation Algorithm

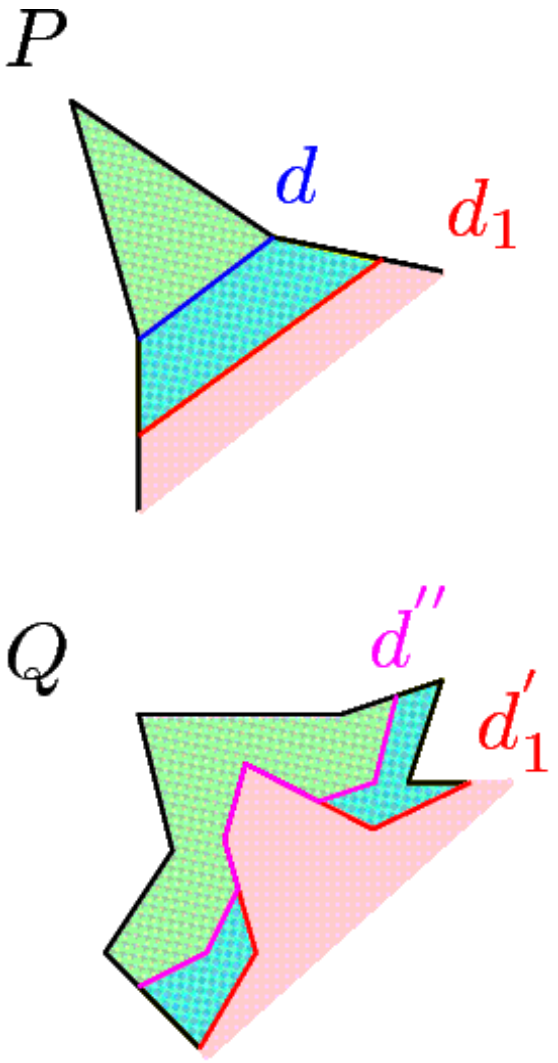
- Key Idea: Approximate away the tangles.
- Suppose P and Q pass the diagonal monotonicity test for ε . We prove that $\delta_F(P,Q) \leq 9\varepsilon$.
- We can then optimize this ε in polynomial time using binary search and the diagonal monotonicity test. Thus, we have a 9 approximation algorithm.
- So, how do we prove $\delta_F(P,Q) \leq 9\varepsilon$?

9-Approximation: Proof Sketch



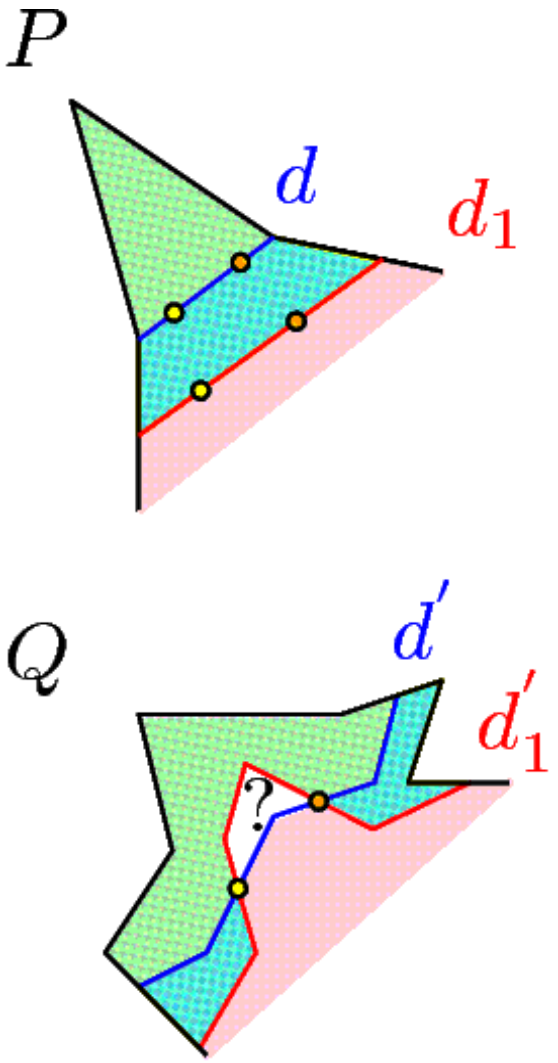
- Choose a diagonal d in P which cuts off an ear.
- To have a homeomorphism between P and Q the image curve of d in Q, call it d', must also cut off an ear.
- If another image curve d_1' crosses d‘ then we no longer have a homeomorphism.

9-Approximation: Proof Sketch



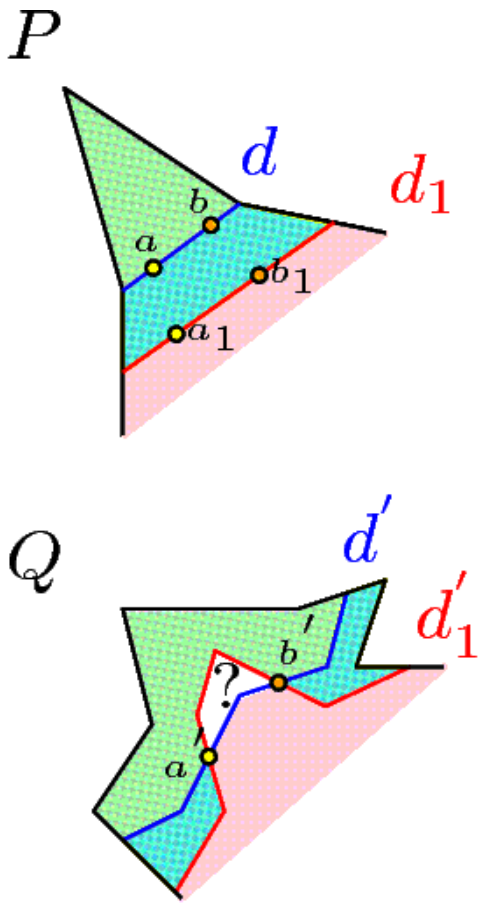
- Idea: Let's map d to the “upper envelope” of the image curves, call it d’’ .
- How much do we need to increase ε to do this?

9-Approximation: Proof Sketch

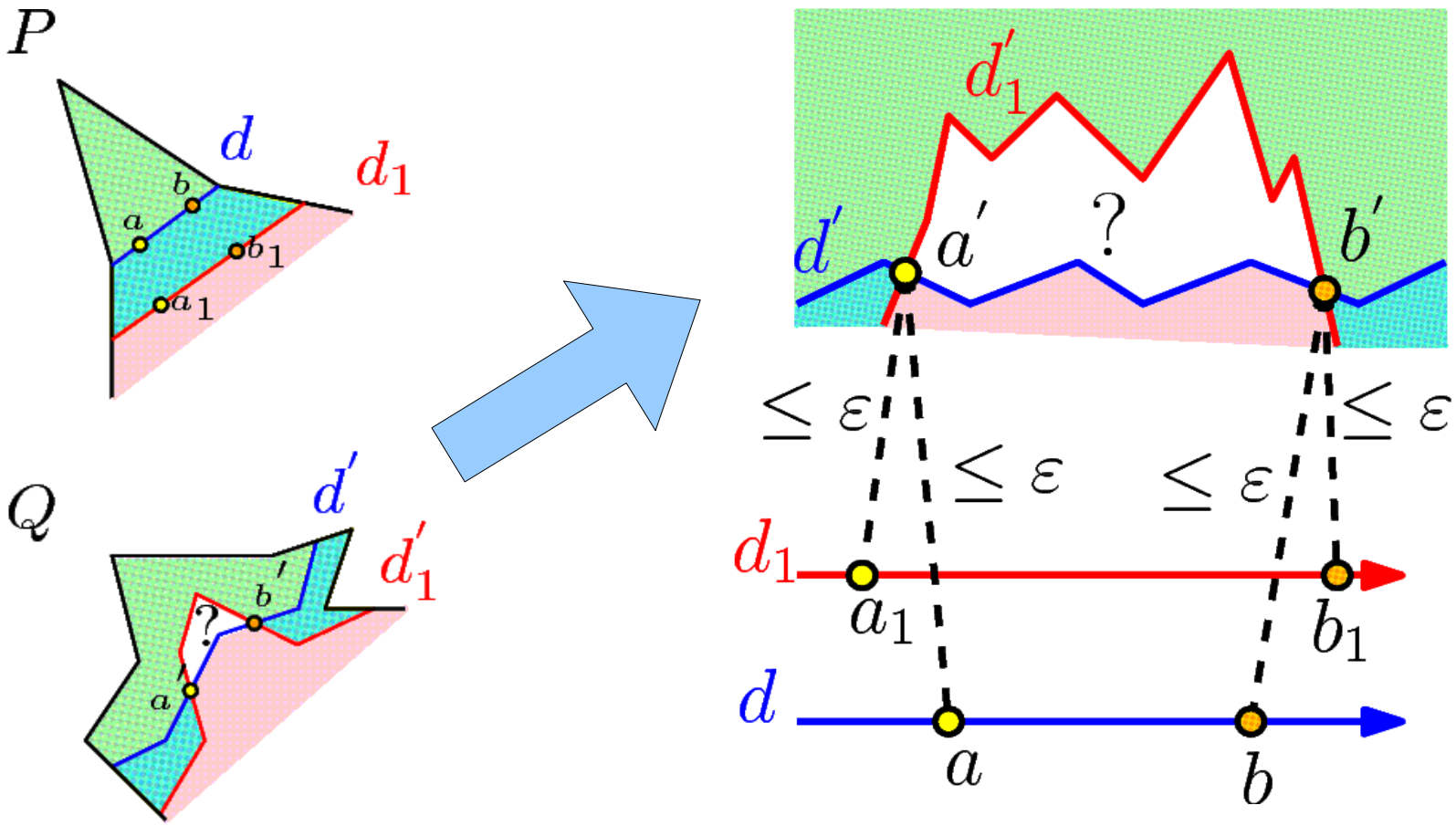


- Consider the pre-images of the points where d' and d_1' cross.
- We can use these to bound how far d is from the part of d_1' that crosses above it.

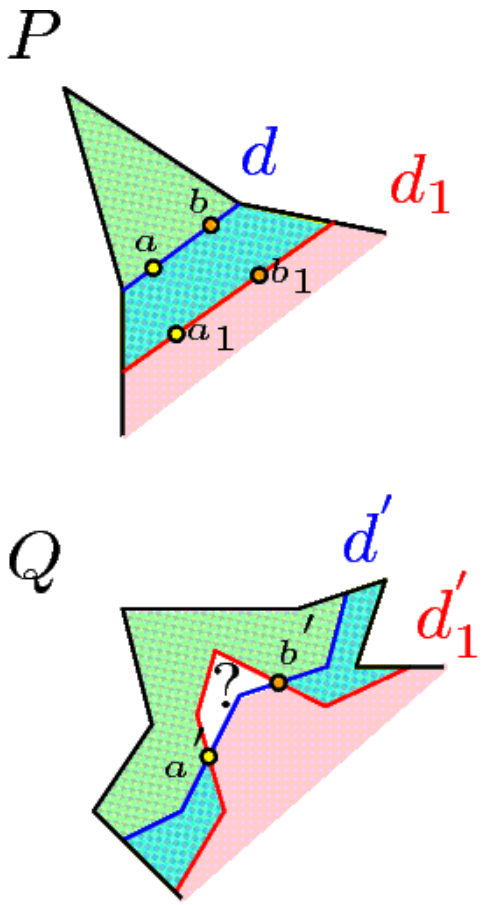
9-Approximation: Proof Sketch

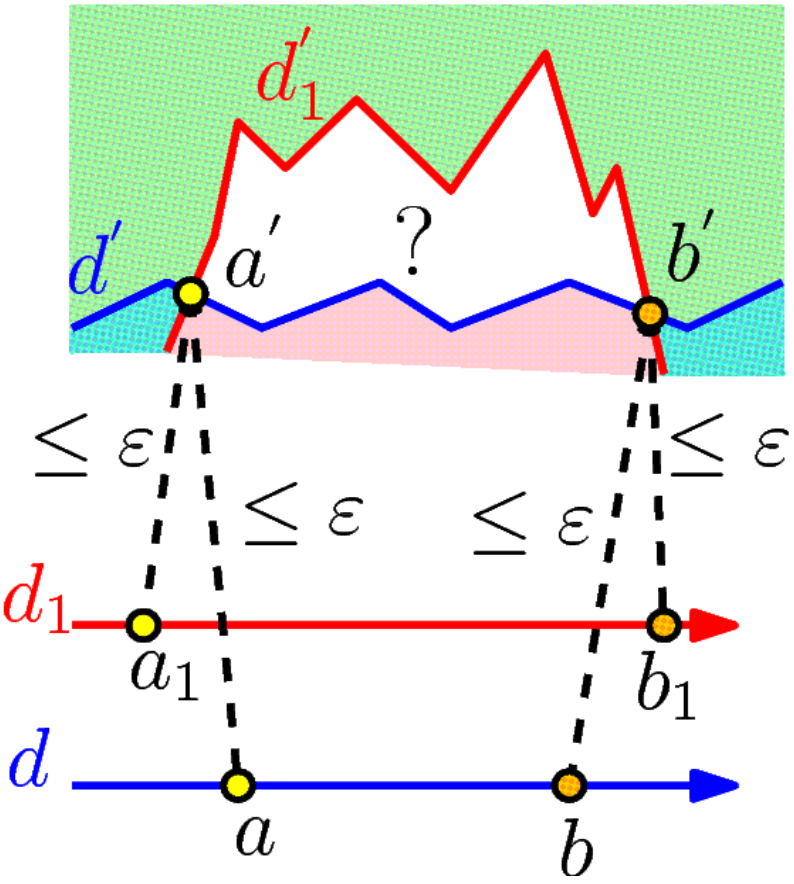


9-Approximation: Proof Sketch

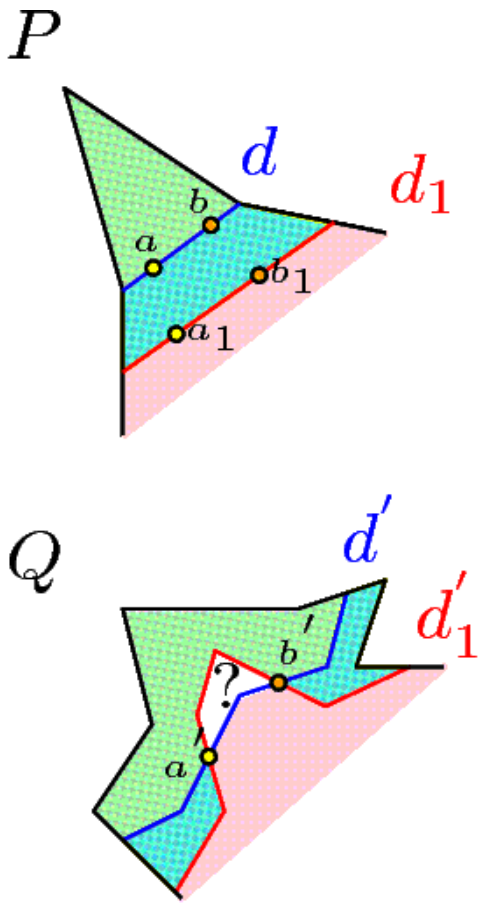


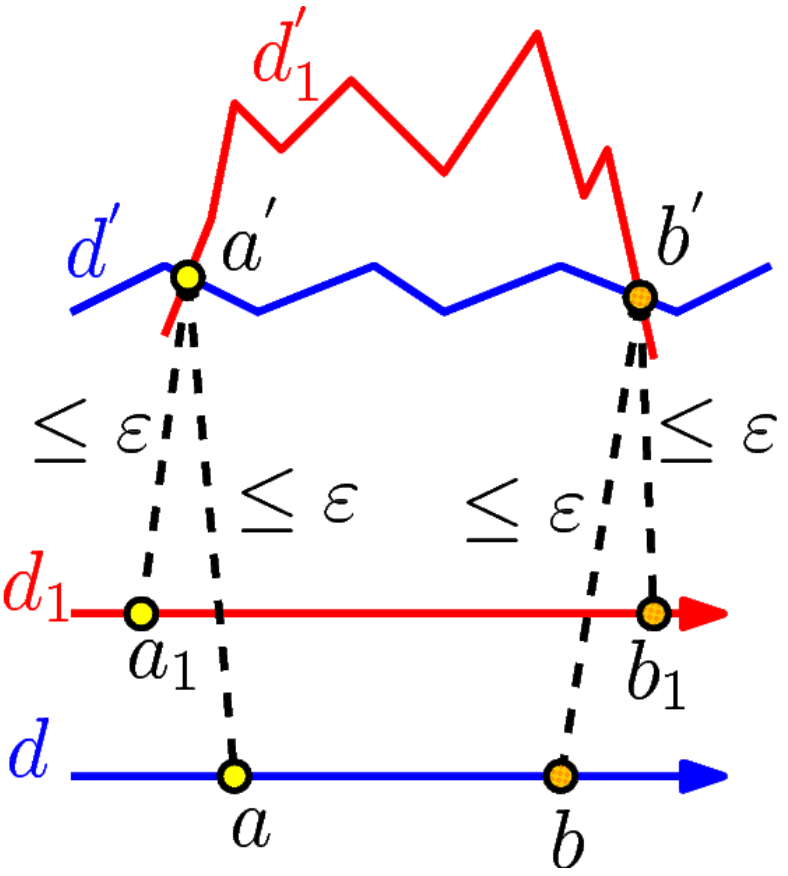
9-Approximation: Proof Sketch



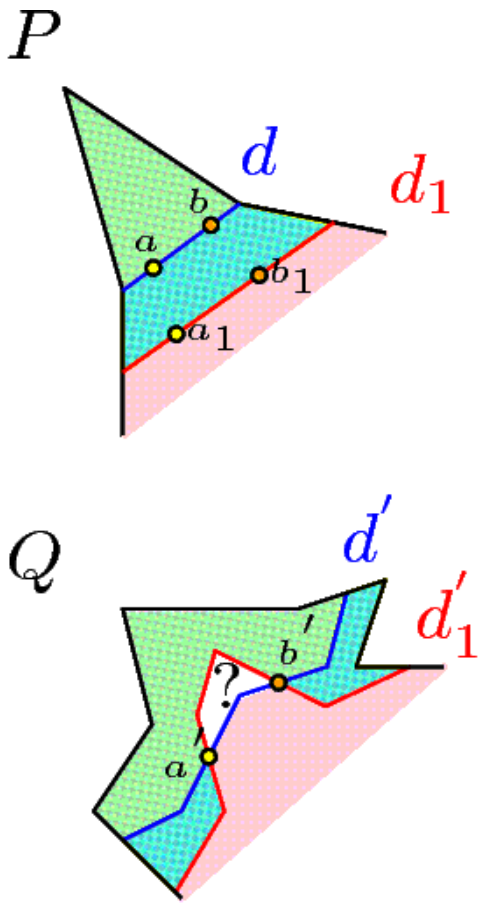


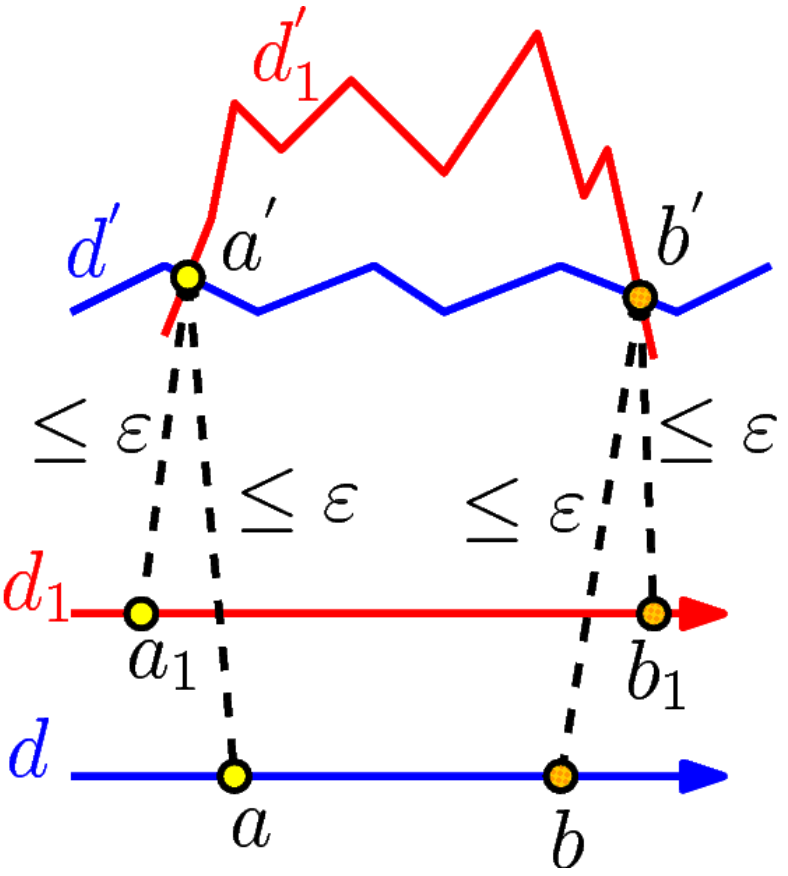
9-Approximation: Proof Sketch



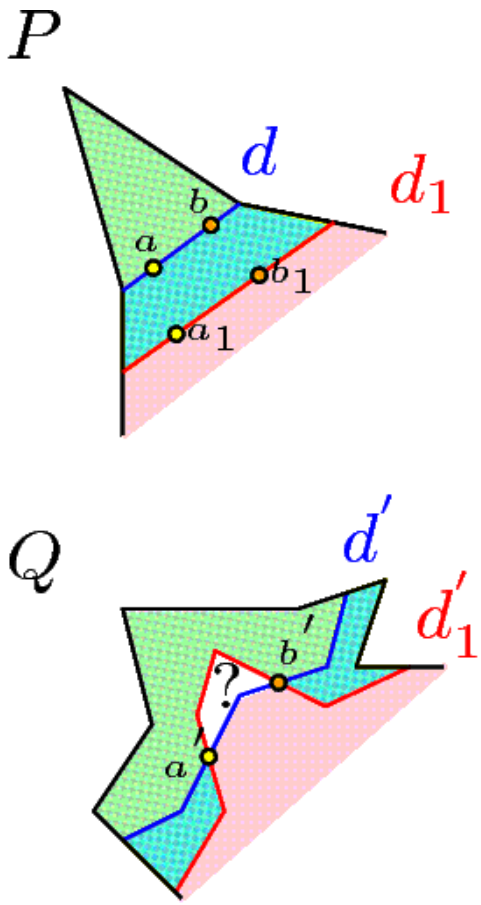


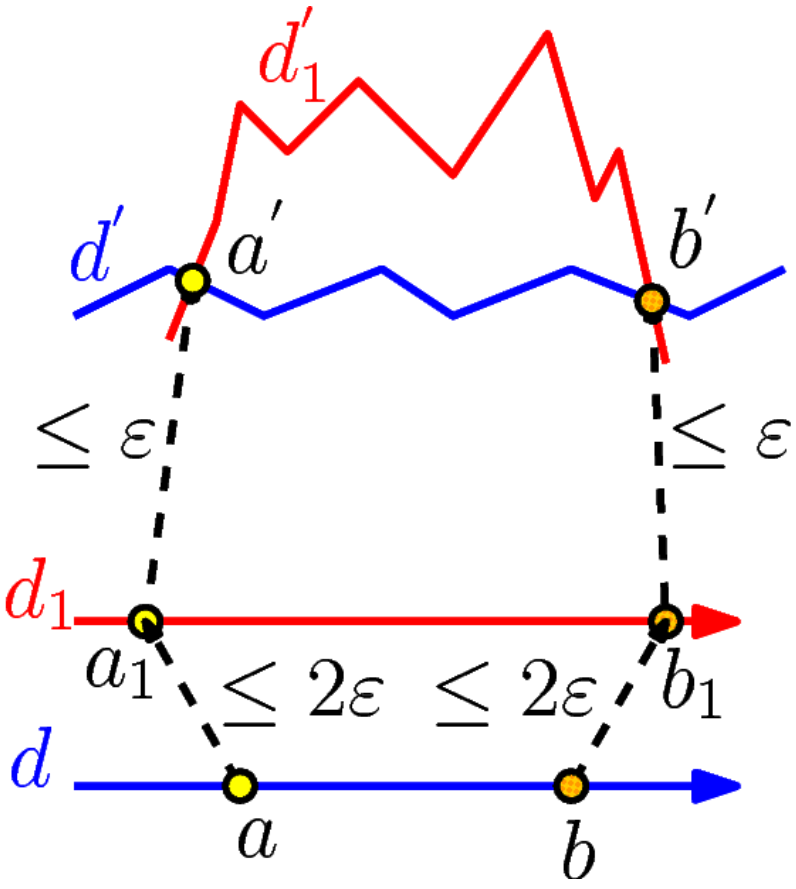
9-Approximation: Proof Sketch





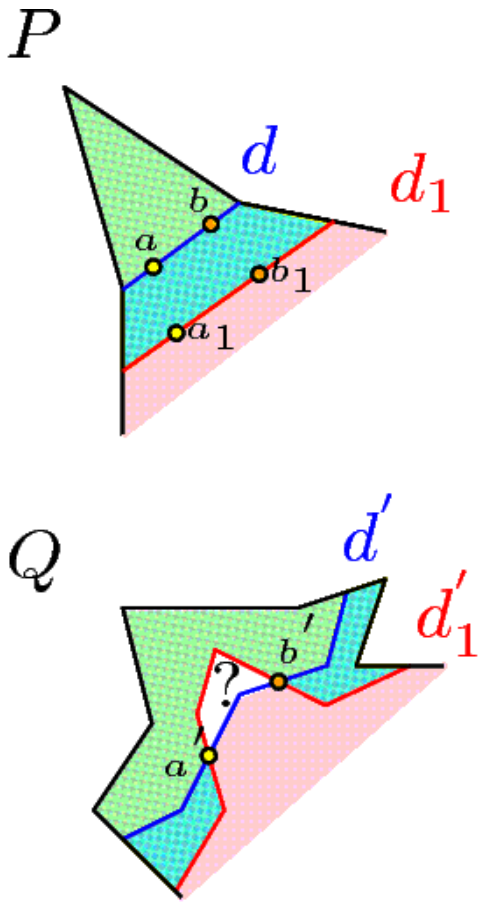
9-Approximation: Proof Sketch

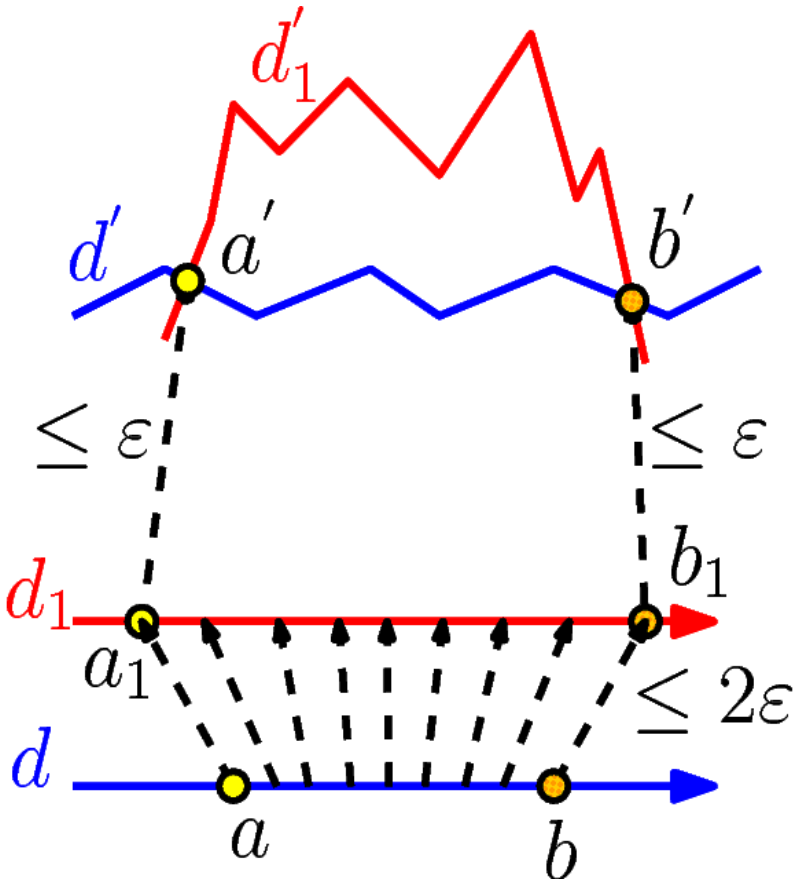




- From triangle inequality: $\delta_F(ab, a_1b_1) \leq 2\varepsilon$.

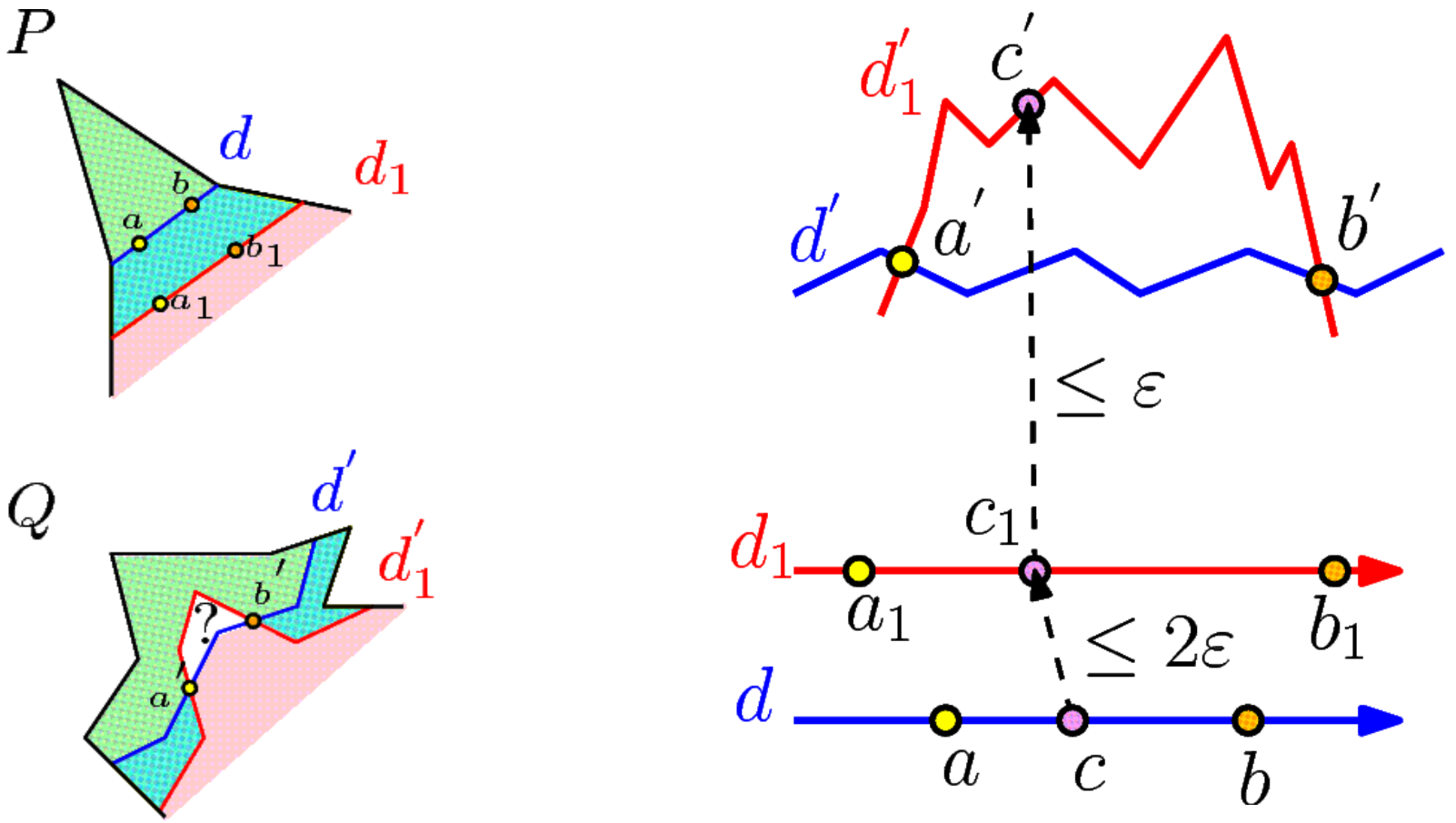
9-Approximation: Proof Sketch





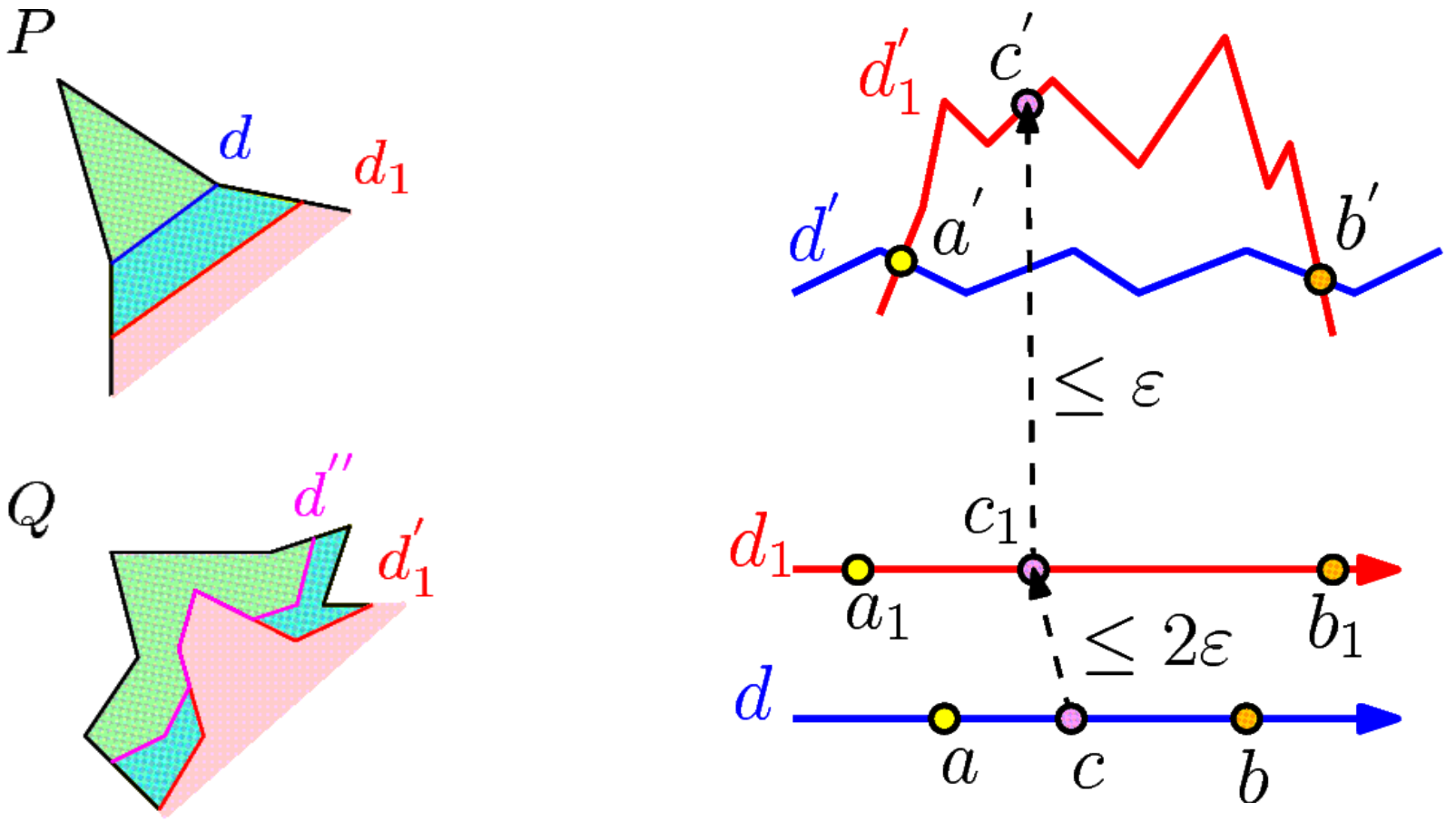
- From triangle inequality: $\delta_F(ab, a_1b_1) \leq 2\varepsilon$.

9-Approximation: Proof Sketch



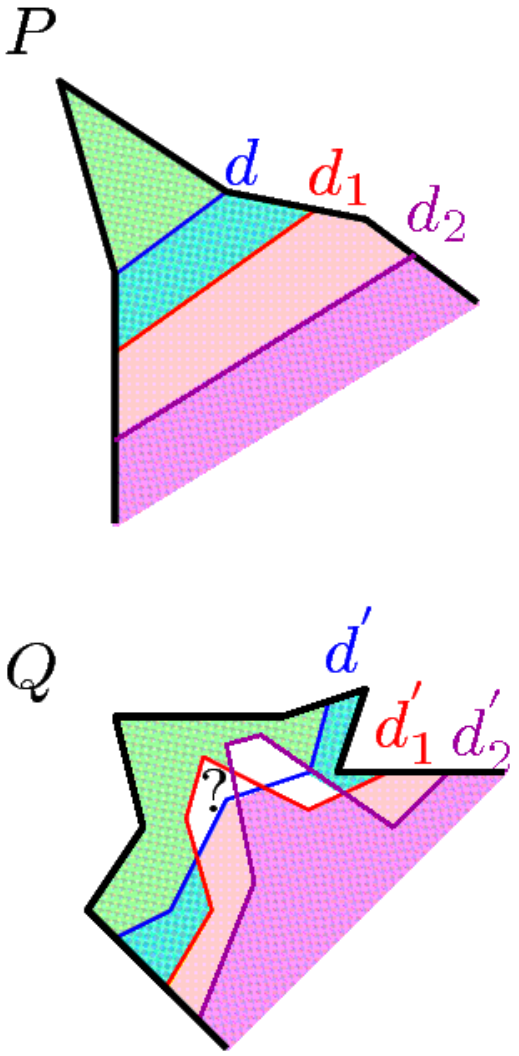
- Thus, $\delta_F(ab,a'b') \leq 3\varepsilon$.

9-Approximation: Proof Sketch



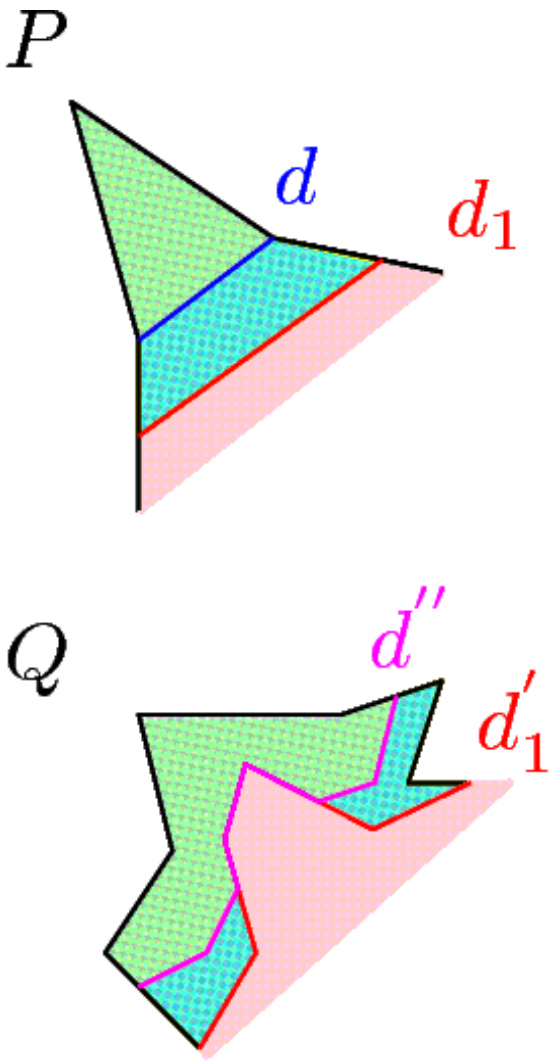
- Thus, $\delta_F(ab,a'b') \leq 3\varepsilon$.

9-Approximation: Proof Sketch



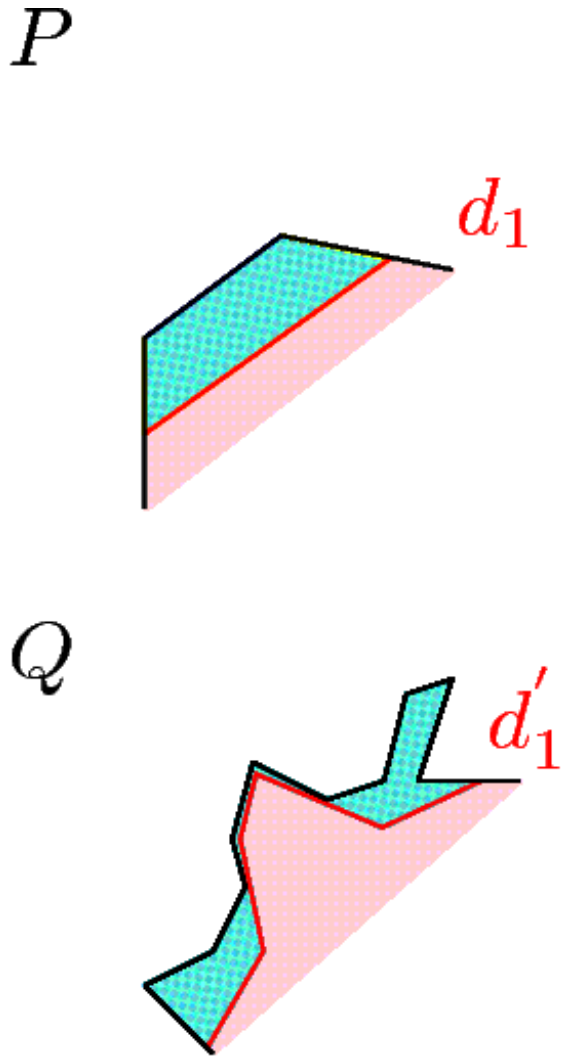
- More complicated cases can occur with additional image curves.
- We show that these cases can be approximated with an additional 6ε factor for a total of 9ε .

9-Approximation: Proof Sketch



- Using the above approach we can incrementally cut off ears from P and map them to Q, in order to obtain an overall mapping witnessing $\delta_F(P,Q) \leq 9\varepsilon$.

9-Approximation: Proof Sketch



- Using the above approach we can incrementally cut off ears from P and map them to Q, in order to obtain an overall mapping witnessing $\delta_F(P,Q) \leq 9\varepsilon$.

Folded Polygons Conclusion

- We gave the first results to compute, or approximate, the Fréchet distance for a class of non-flat surfaces (“folded polygons”)
- Can the approximation factor be improved?
- Is there a poly-time algorithm for folded polygons?

Outline

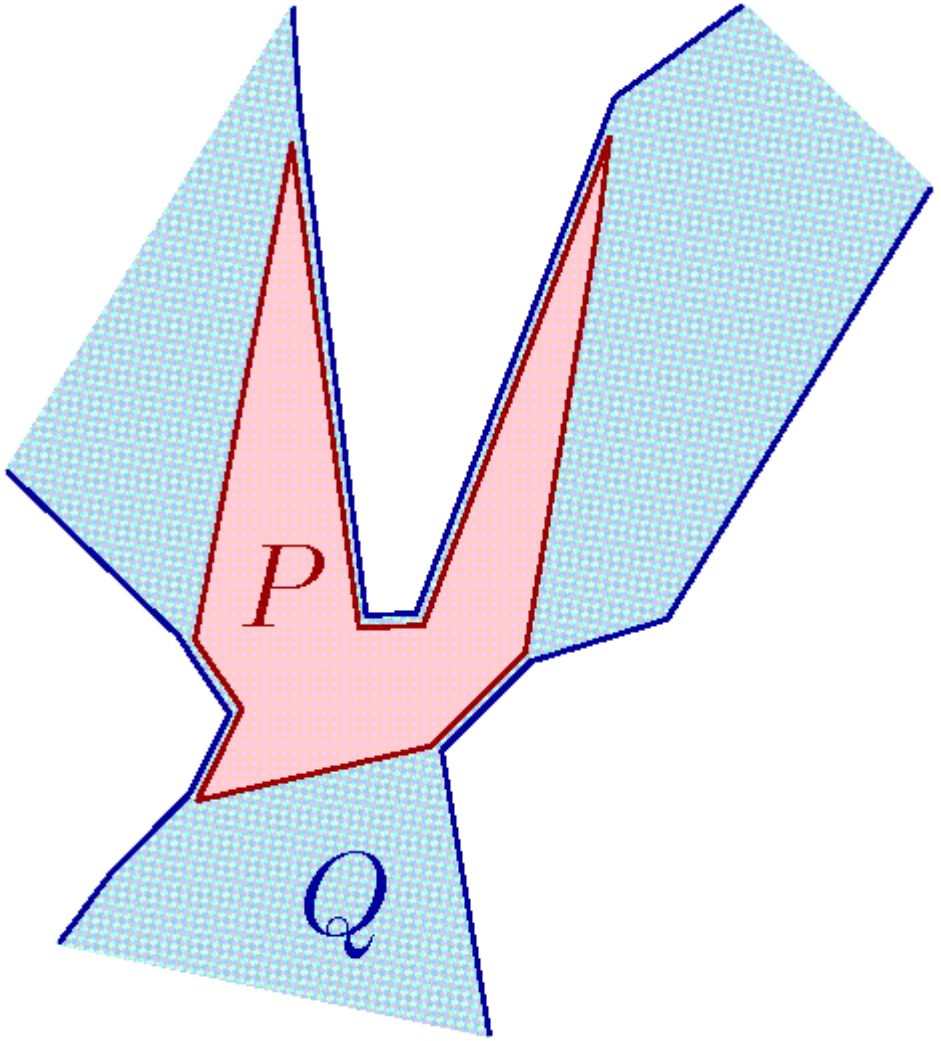
- Degree Plan
- Problem and Motivation
- Fréchet Distance
- **Prior Contributions**
  - Folded Polygons Paper
  - **Partial Matching Paper**
- Future Work
  - Constrained Embeddings
  - Flippy Distance

Partial Matching between Surfaces Using Fréchet Distance

Submitted to the Symposium on Computational Geometry (SoCG) 2012

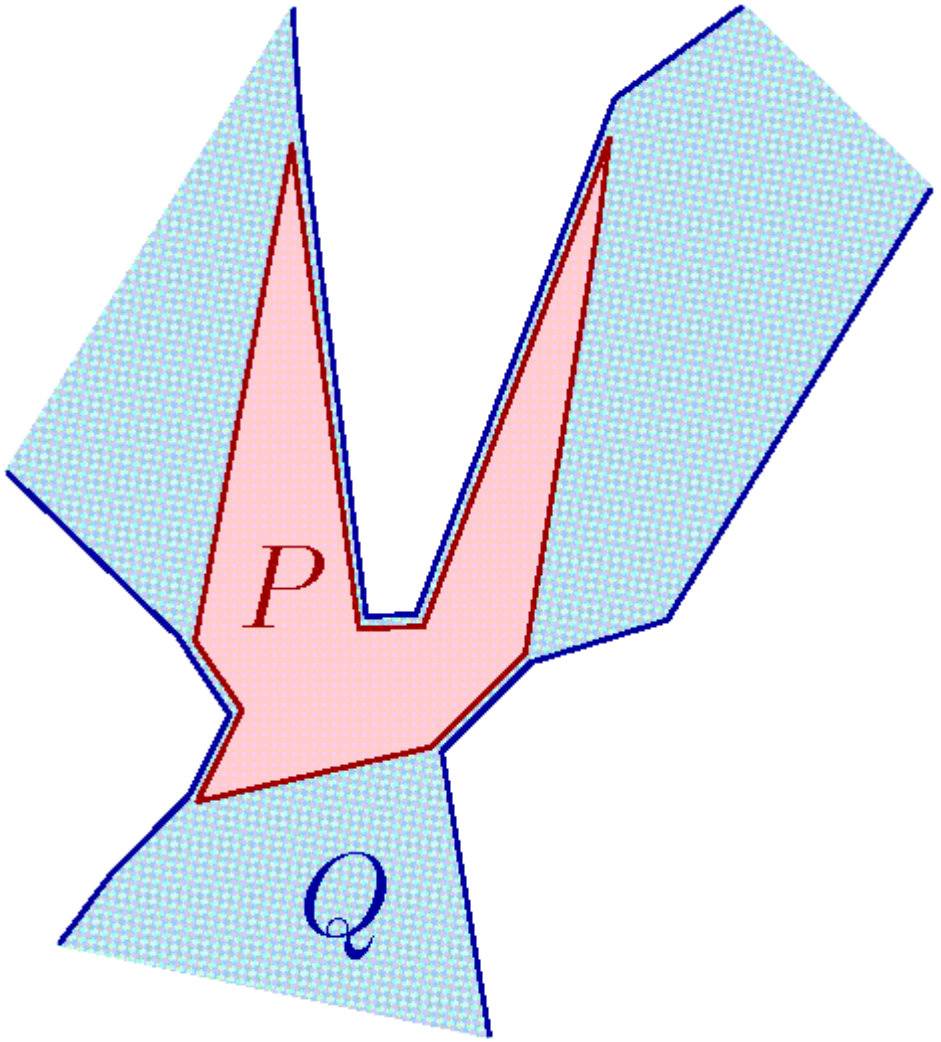
Authors: Jessica Sherette, Carola Wenk

Problem and Motivation



- This previous algorithm matches the entirety of the surfaces.
- An interesting variant to consider is partial matching.
- For certain applications we would like to consider this form of similarity.

Problem and Motivation



- Many possible definitions.
  - We examine one possible definition.

Partial Matching Problem

- We consider the following problem:

Given two coplanar triangulated simple polygons P and Q and some $\varepsilon > 0$, decide whether there exists a simple polygon $R \subseteq Q$ such that $\delta_F(P, R) \leq \varepsilon$.

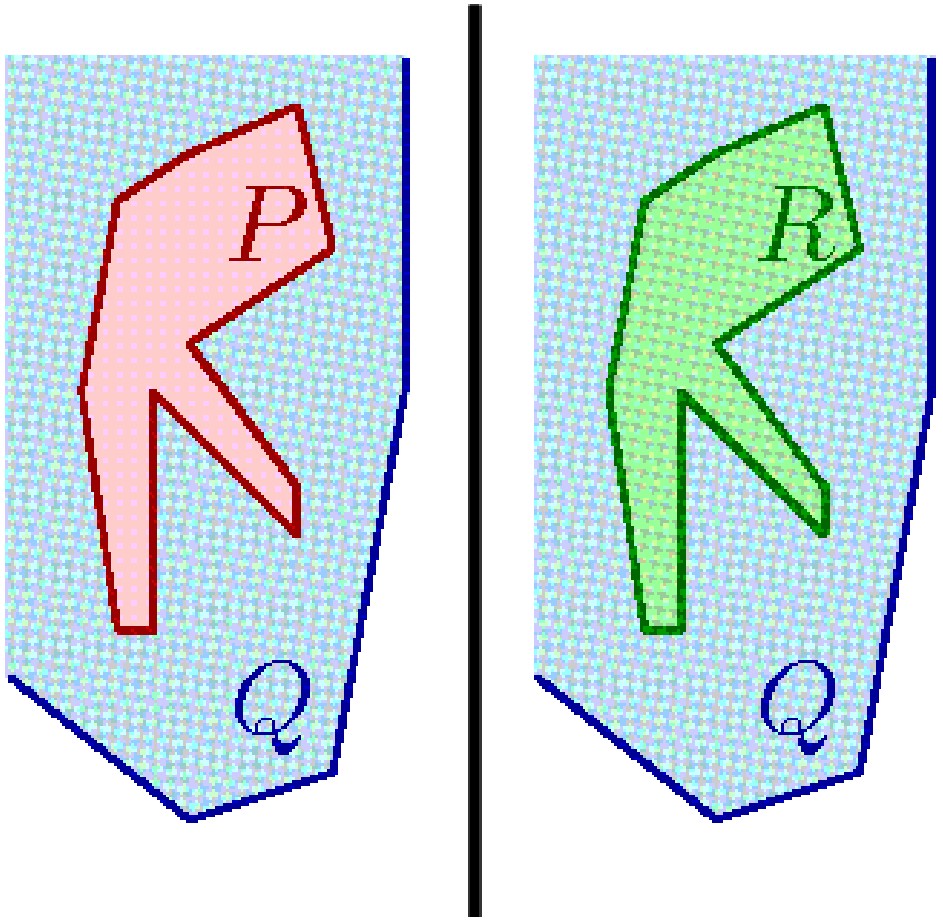
Partial Matching Problem

- We consider the following problem:

> Given two coplanar triangulated simple polygons P and Q and some $\varepsilon > 0$, decide whether there exists a simple polygon $R \subseteq Q$ such that $\delta_F(P, R) \leq \varepsilon$.

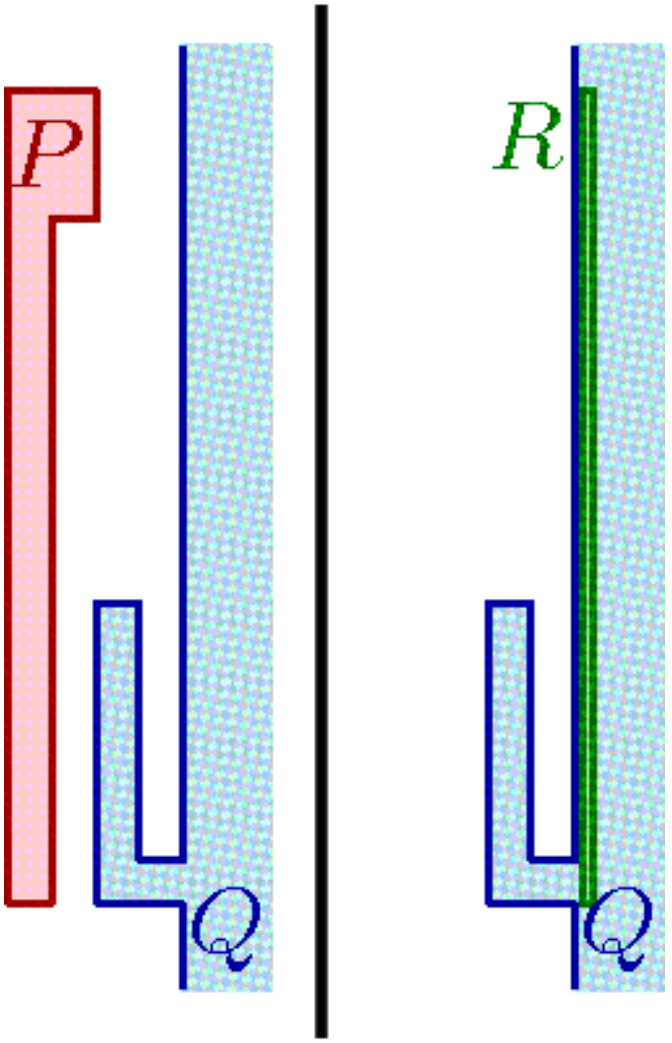
- Notice that this definition is directed. P is matched to some part of Q.
- Next we consider several simple cases of this problem.

Partial Matching Problem



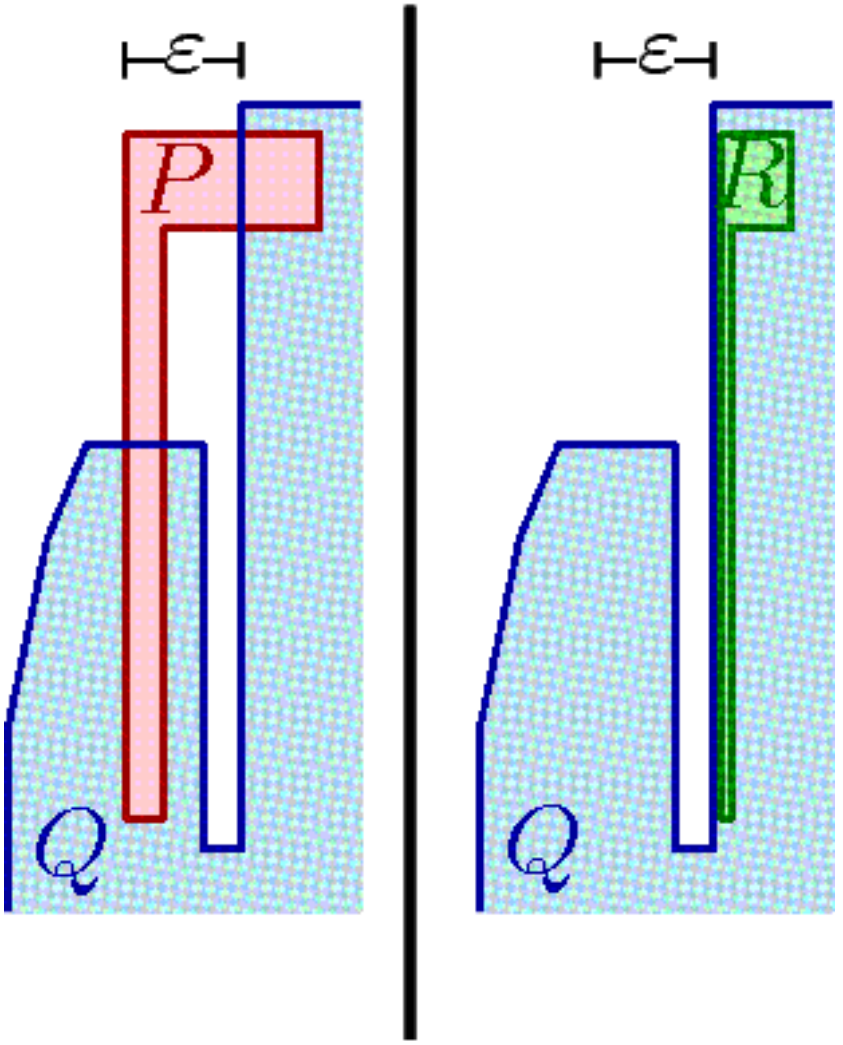
- If P overlaps completely with Q then we can use R = P and epsilon = 0.

Partial Matching Problem



- If P does not intersect Q then it is similar to projection to the boundary of Q.

Partial Matching Problem



- This case is a bit harder. The points which overlap with Q are not always mapped straight down.
- How we map a point in P to a point in Q depends on how other points can be mapped.

Partial Matching Problem

- Finally note the following:
  - It is NP-hard to decide the partial Fréchet distance (for this def.) between two polygons with holes or two terrains.
- This can be shown using the same reductions as outlined in [BBS10].

Partial Matching Problem

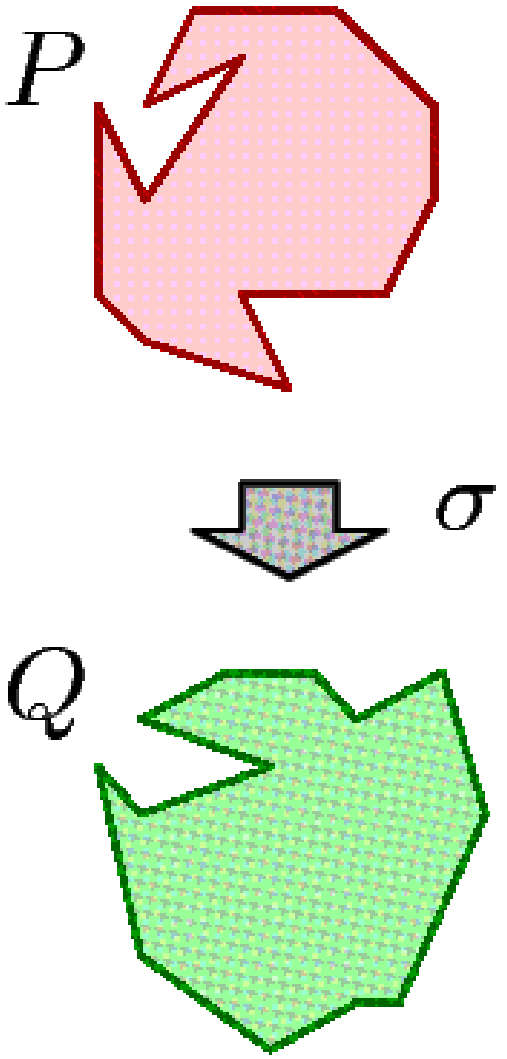
- We adapt the algorithm for simple polygons to one for our problem.
- This yields a polynomial time algorithm.
- Ours is the first algorithm for computing the partial FD between surfaces.

Partial Matching Problem

- We adapt the algorithm for simple polygons to one for our problem.
- This yields a polynomial time algorithm.
- Ours is the first algorithm for computing the partial FD between surfaces.
- Next we give some additional details about the simple polygons algorithm.

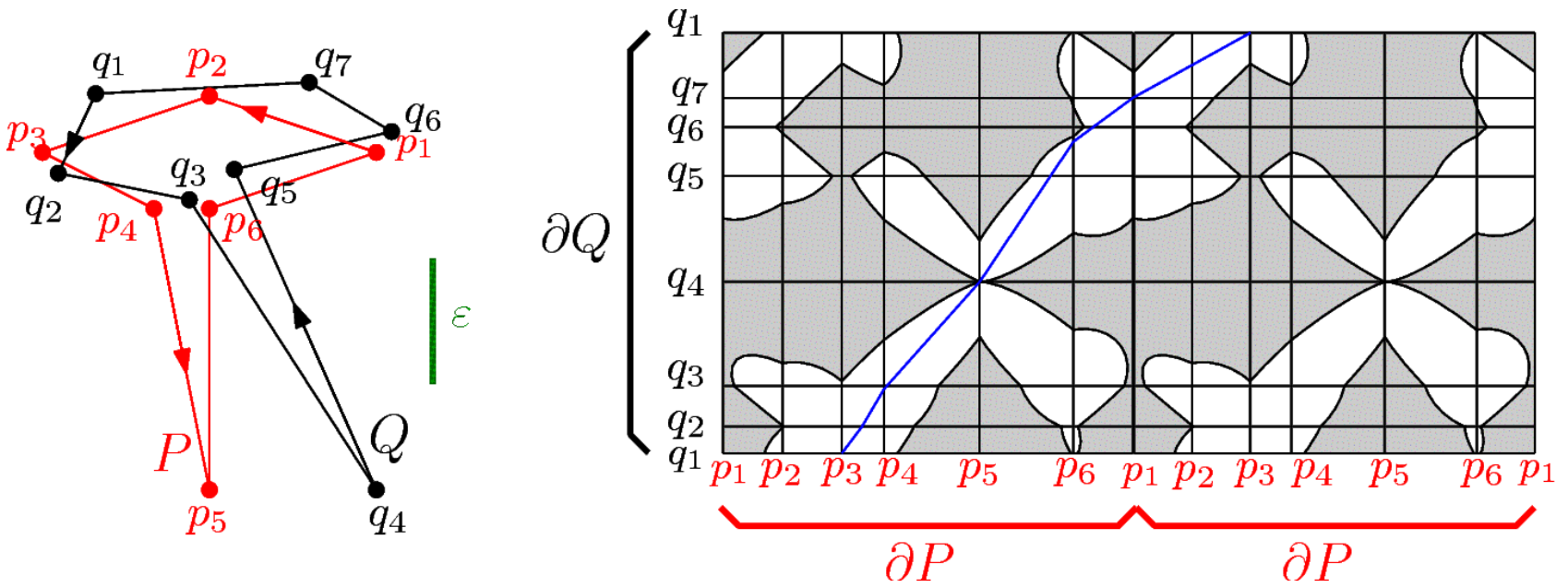
Simple Polygons Algorithm [BBW06]

- As mentioned before we can compute the FD between the boundaries of two simple polygons in polynomial time.
- How do you compute this?



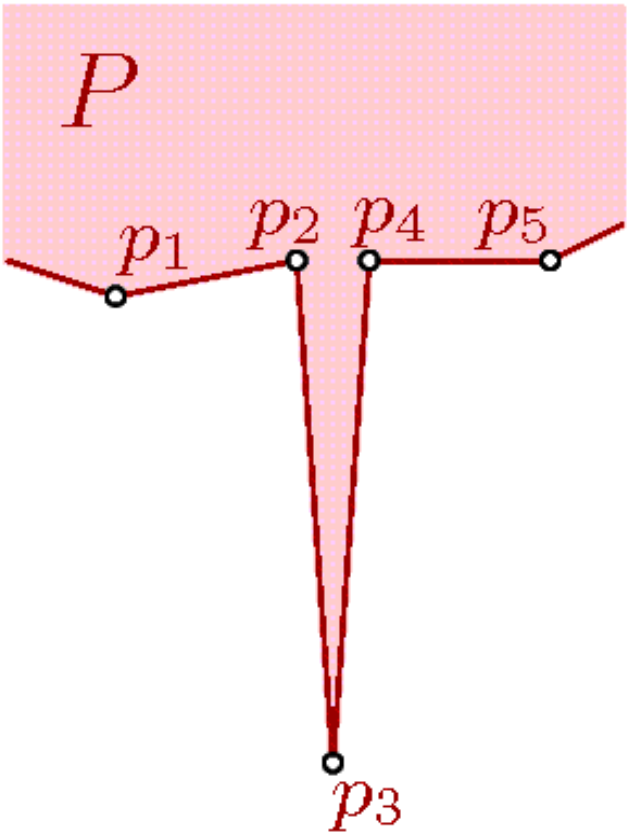
Simple Polygons Algorithm [BBW06]

- The free space diagram (FSD) encodes which points on the boundary on the polygons may be mapped together with distance epsilon.



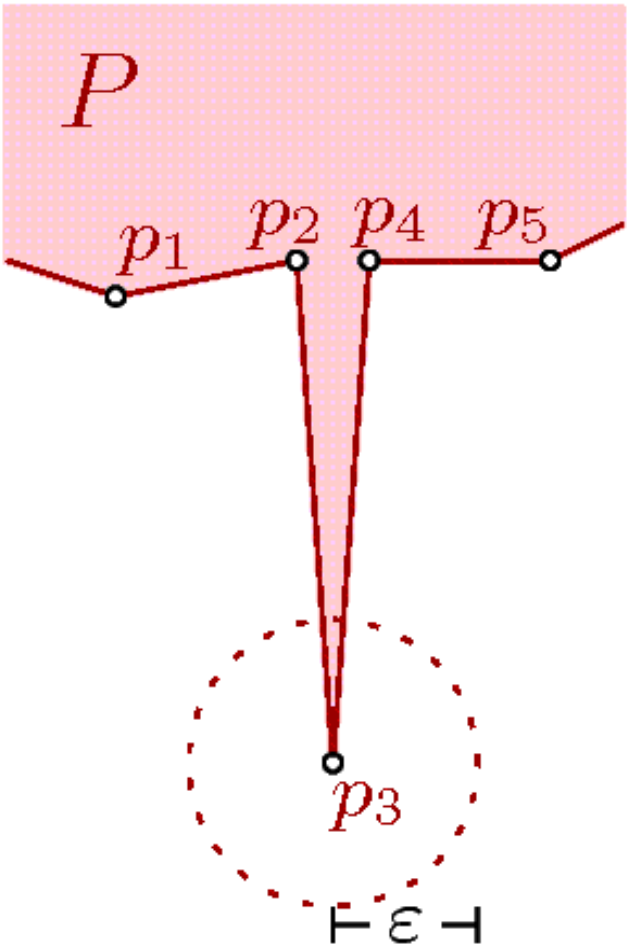
Simple Polygons Algorithm [BBW06]

- Here is part of a simple polygon P.
- First consider the point p_3 in P.



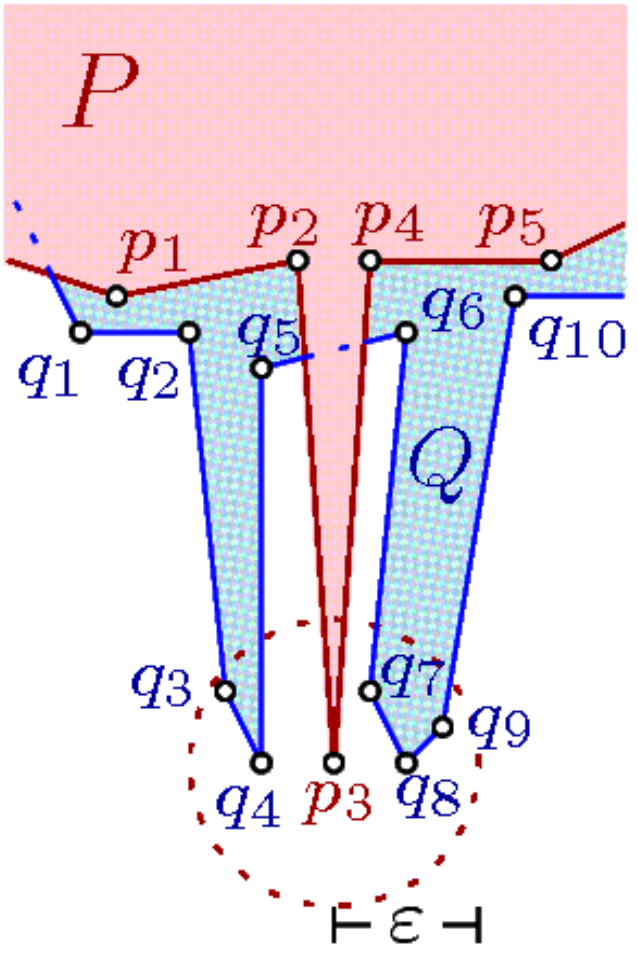
Simple Polygons Algorithm [BBW06]

- Consider the points within epsilon distance of p_3.
  - (Euclidean distance)



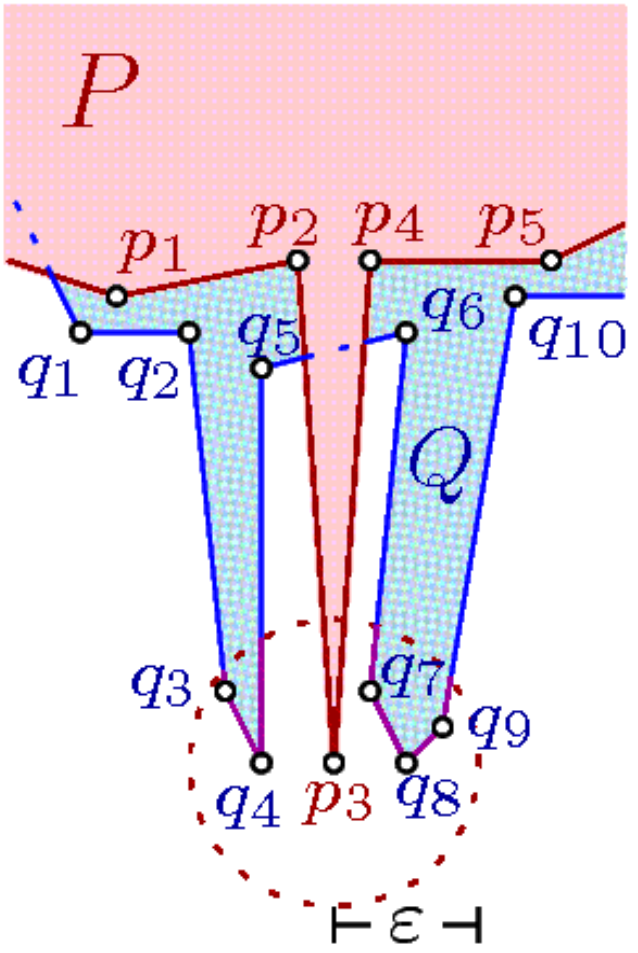
Simple Polygons Algorithm [BBW06]

- The points on the boundary Q in this disc are those that p_3 can be mapped to.



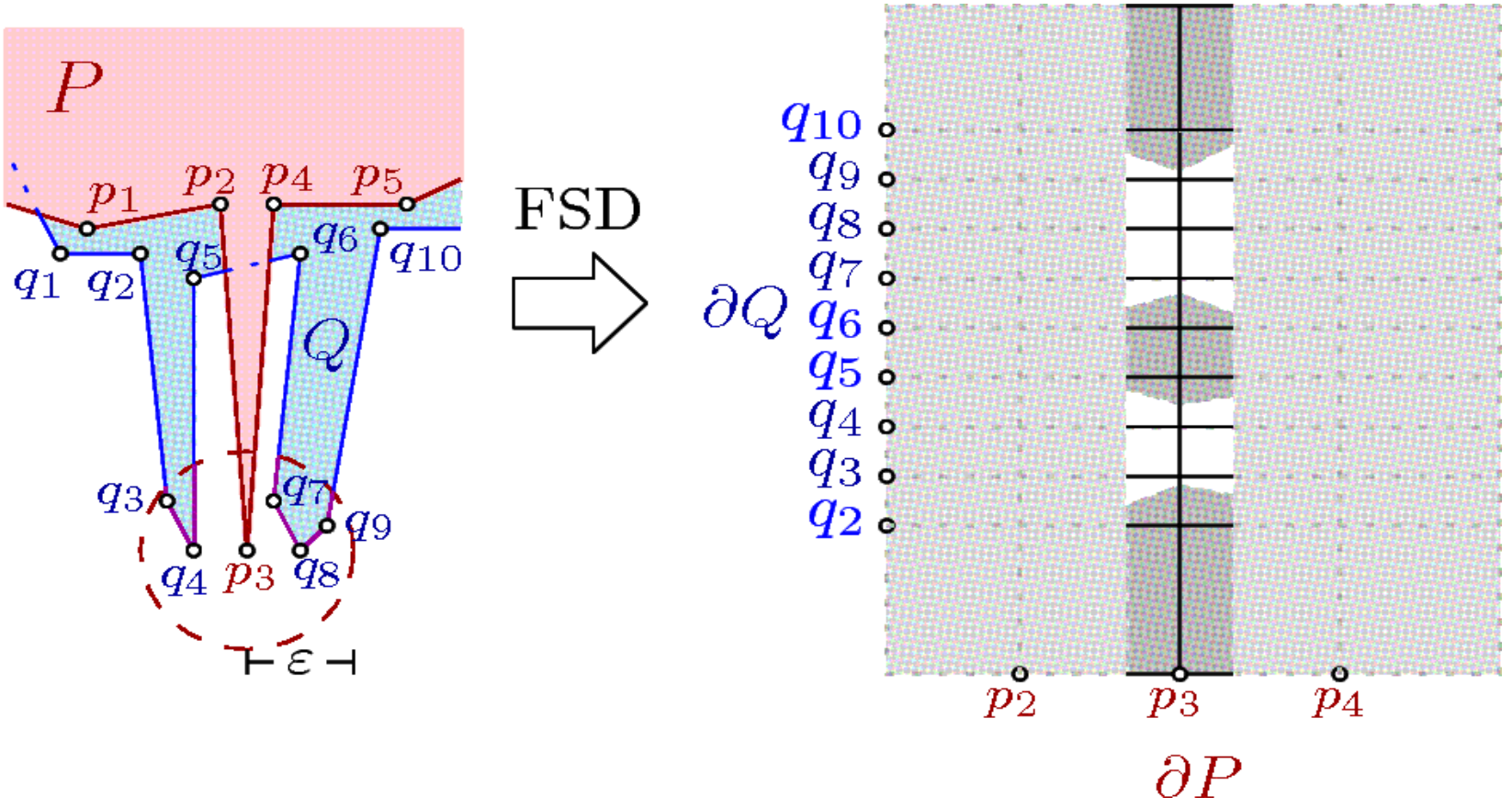
Simple Polygons Algorithm [BBW06]

- Note the highlighted parts.



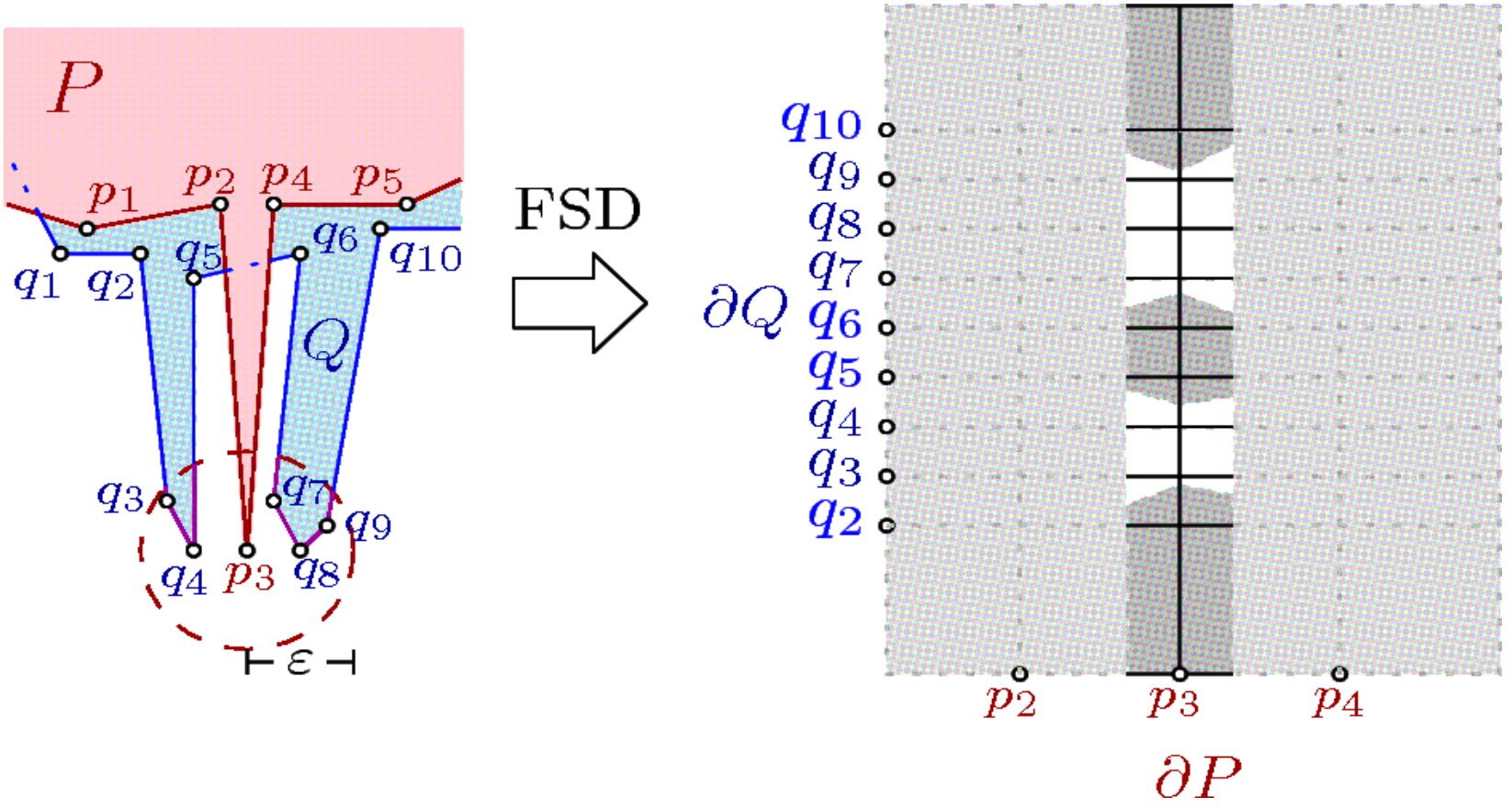
Simple Polygons Algorithm [BBW06]

- Below is a small part of the free space diagram (FSD).



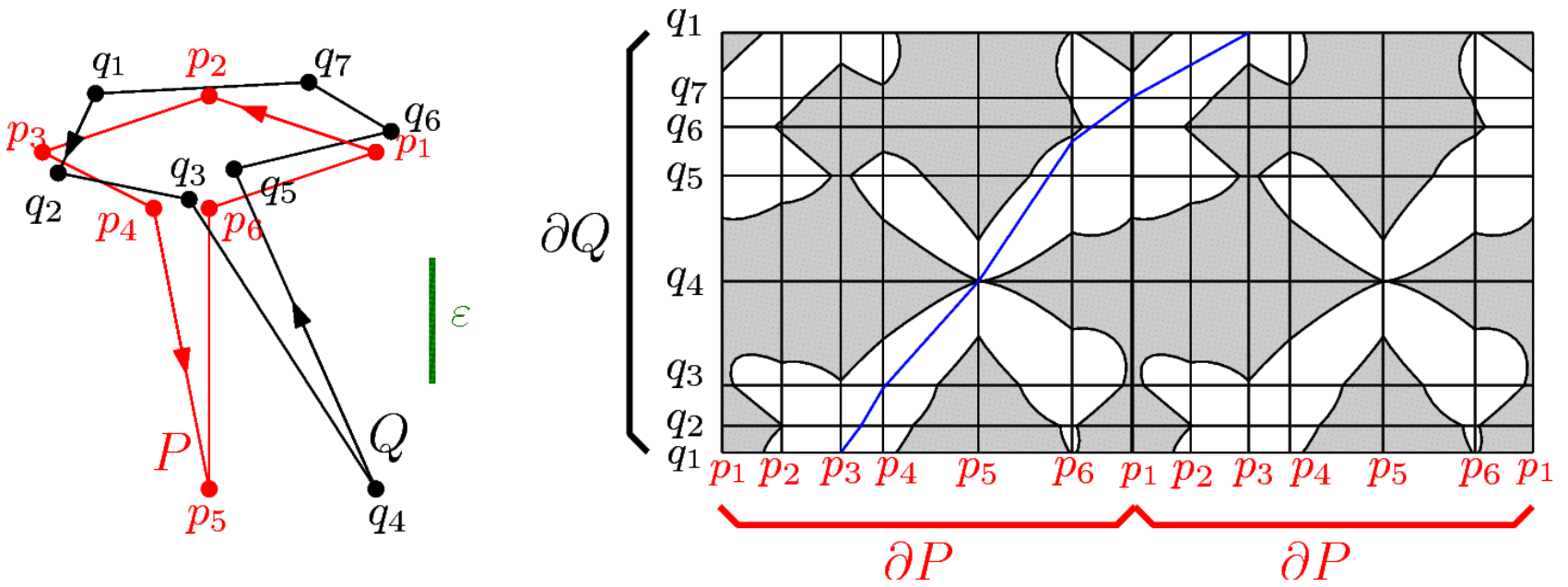
Simple Polygons Algorithm [BBW06]

- White points along the line segment are ones that p_3 can be mapped to.



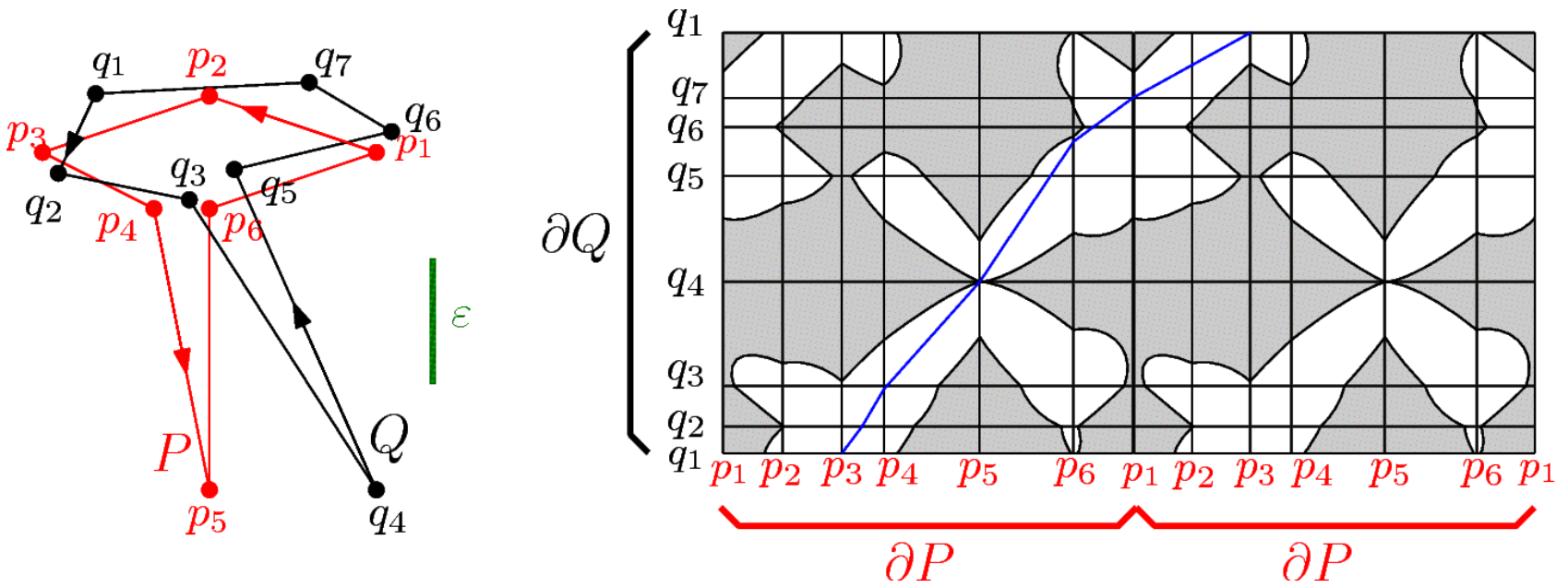
Simple Polygons Algorithm [BBW06]

- Below is an example of an actual FSD between two simple polygons for some epsilon.



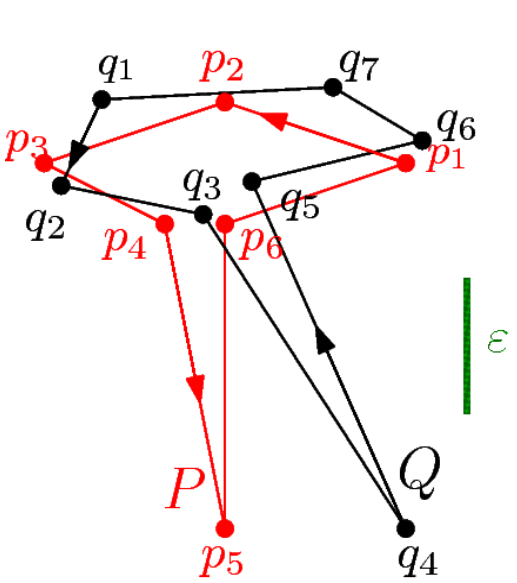
Simple Polygons Algorithm [BBW06]

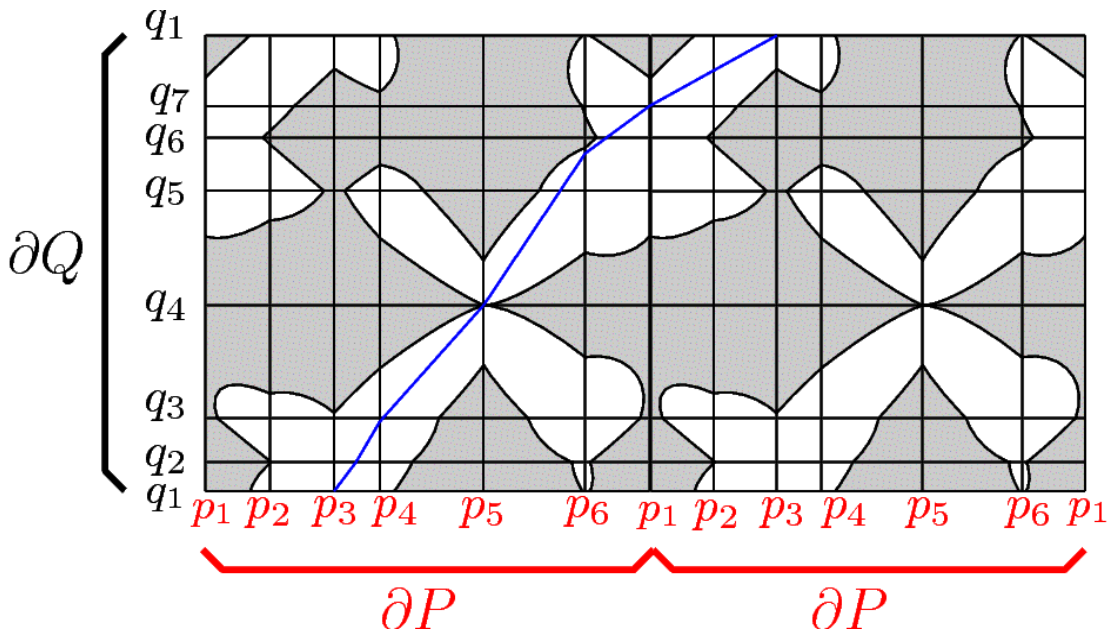
- Decide if a monotone path from the bottom of the FSD to the top which maps every point in P once and only once exists. (see blue path)



Simple Polygons Algorithm [BBW06]

- If such a path exists (plus the diagonal details from before) then P and Q are within Fréchet distance epsilon (see green line segment).





Partial Matching Algorithm: FSD

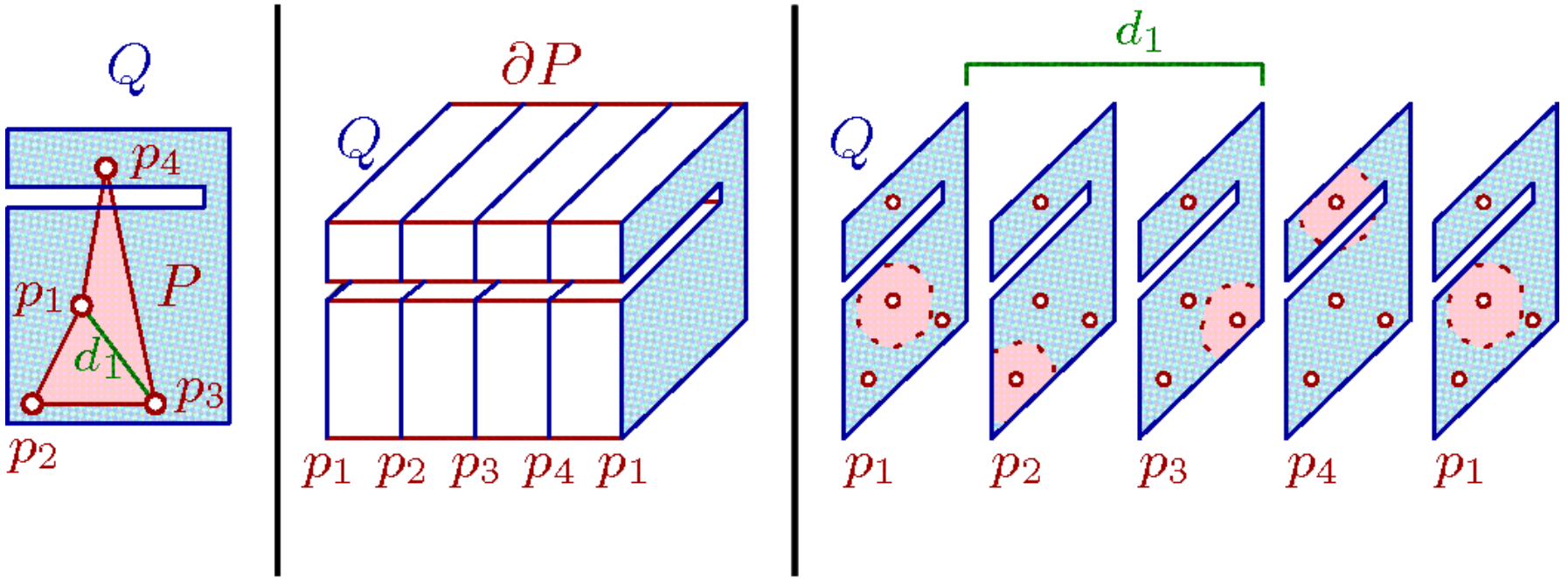
- We want to adapt this to our problem.

Partial Matching Algorithm: FSD

- We want to adapt this to our problem.
- The FSD in the simple polygons algorithm pairs points on the **boundary of P** to points on the **boundary of Q**.
- We want to map points on the **boundary of P** to any points in **Q**.
- We use a 3d FSD diagram between the **boundary of P** and **Q**.

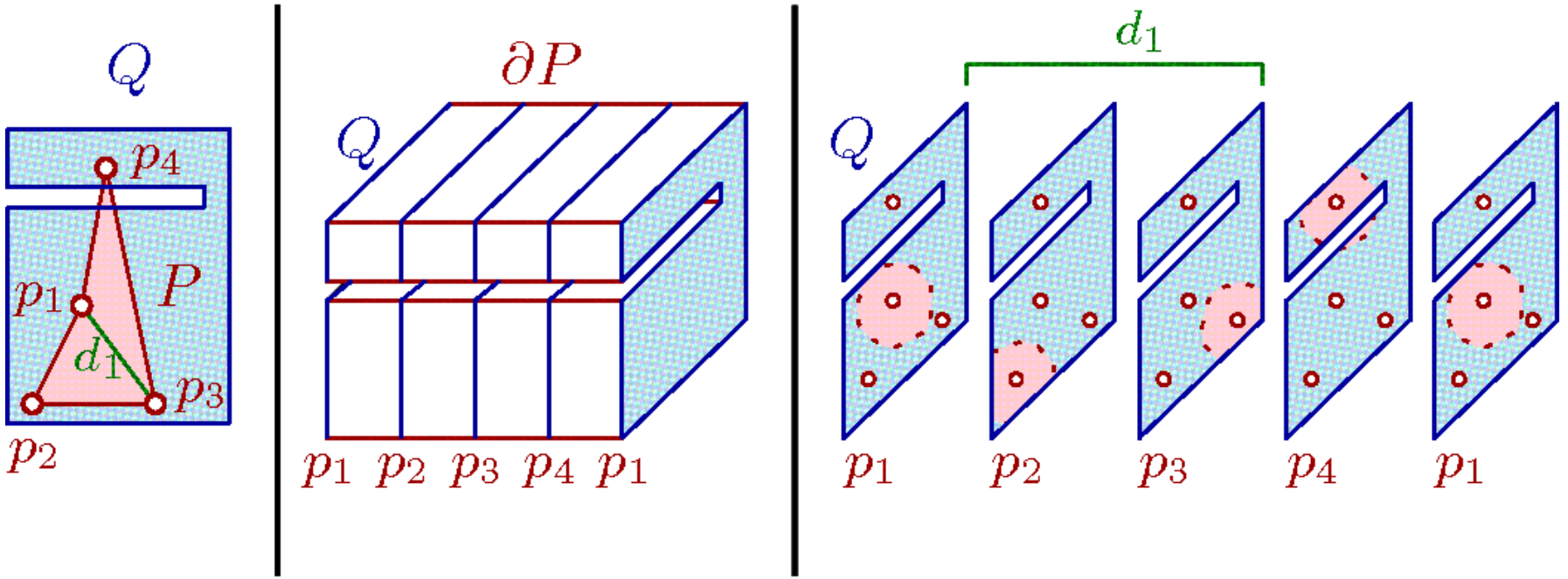
Partial Matching Algorithm

- Below is the 3D FSD diagram associated with the simple polygons P and Q for some epsilon. We refer to the portion of a FSD associated with a point as a slice of the FSD.



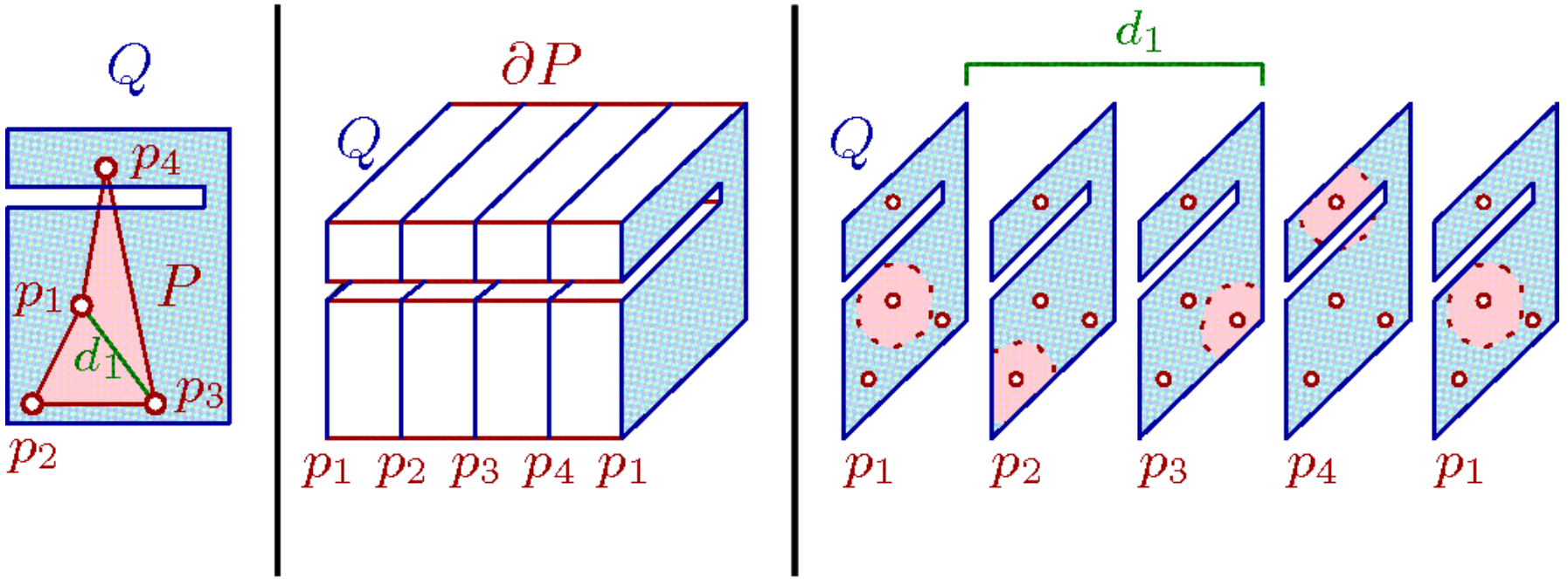
Partial Matching Algorithm

- We cut the boundary of P open at the point p_1 so it appears twice in the slices. Naturally, our mapped boundary must start and end at the same point.



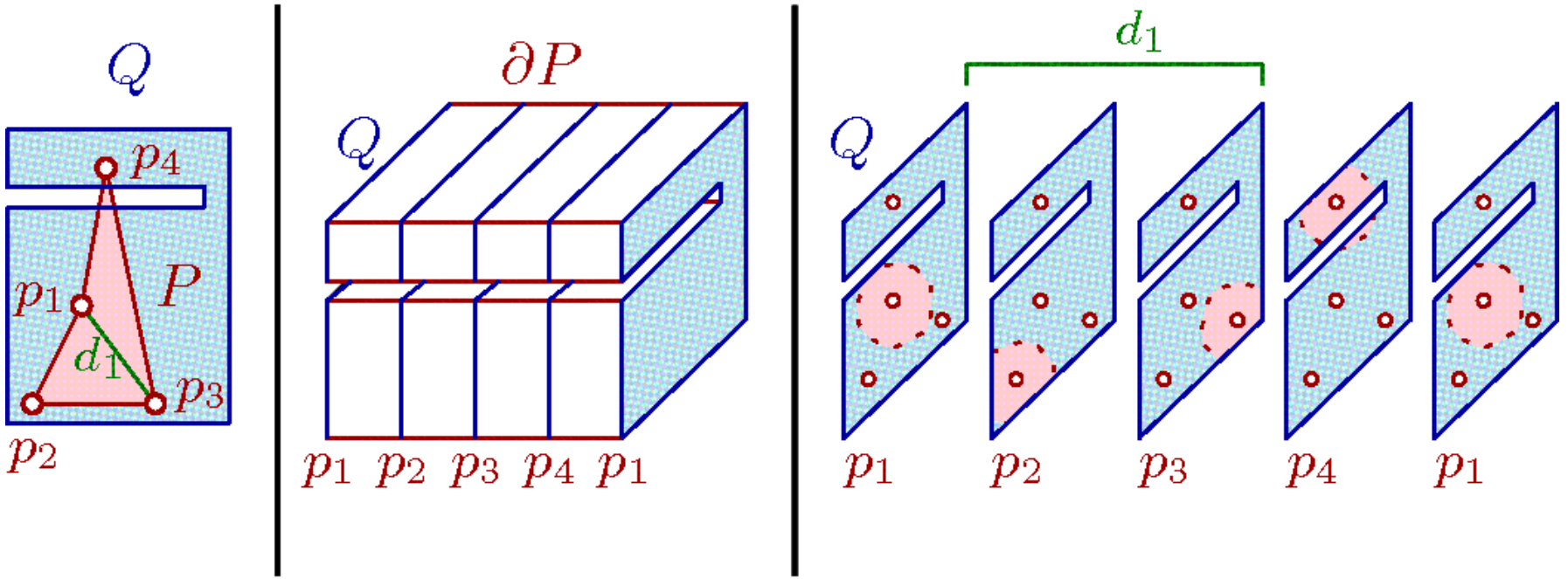
Partial Matching Algorithm

- We now want to find a monotone path through the FSD from the first slice to the last slice. This yields a mapping of the boundary of P into Q.



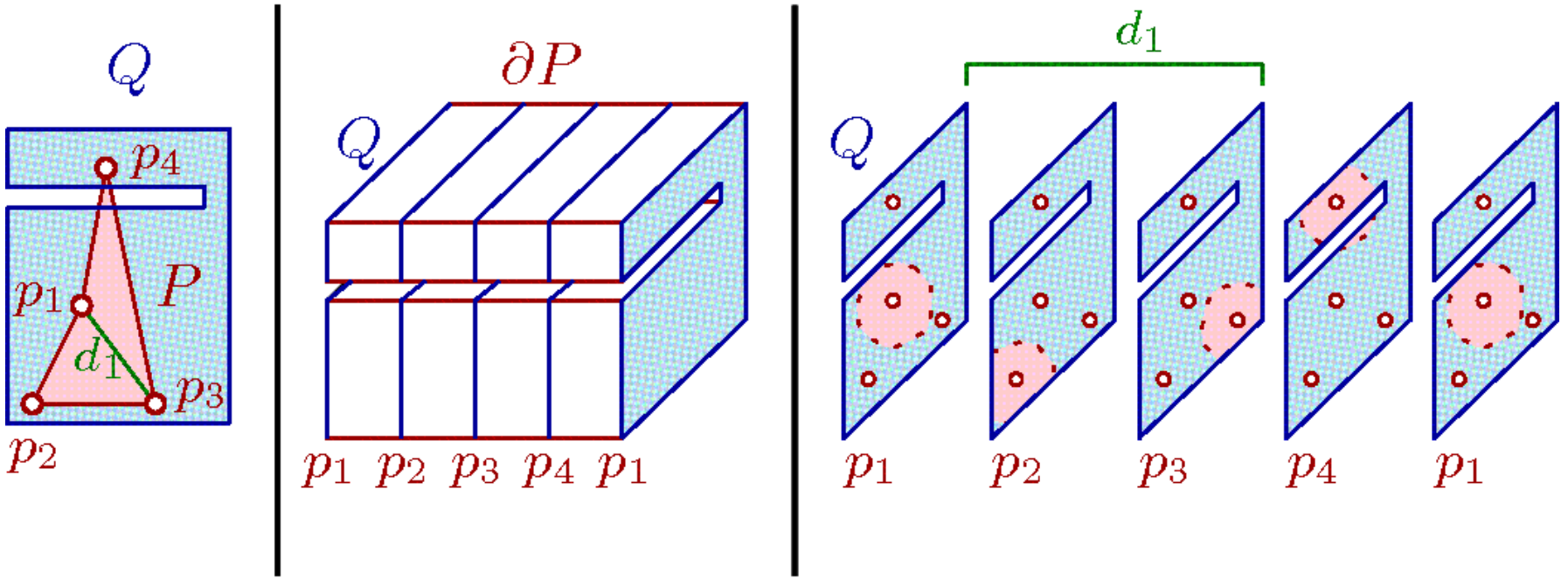
Partial Matching Algorithm

- Note also that the path is only required to be monotone along the boundary of P now. (there are issues with Q as well but we deal with them later)

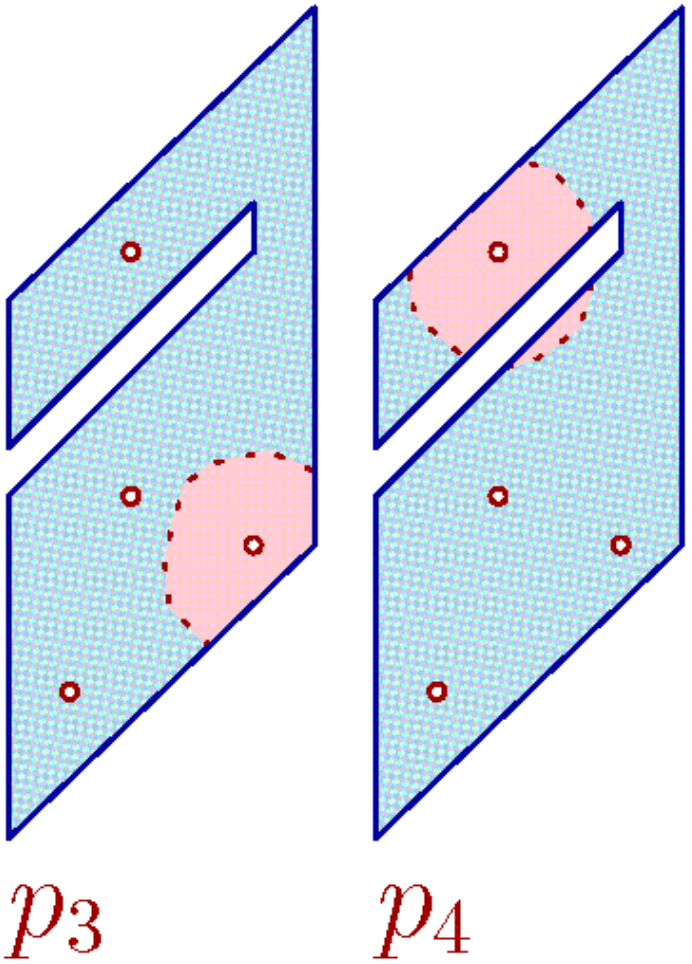


Partial Matching Algorithm

- To find such a path in the FSD we propagate reachability information in the FSD through the slices.

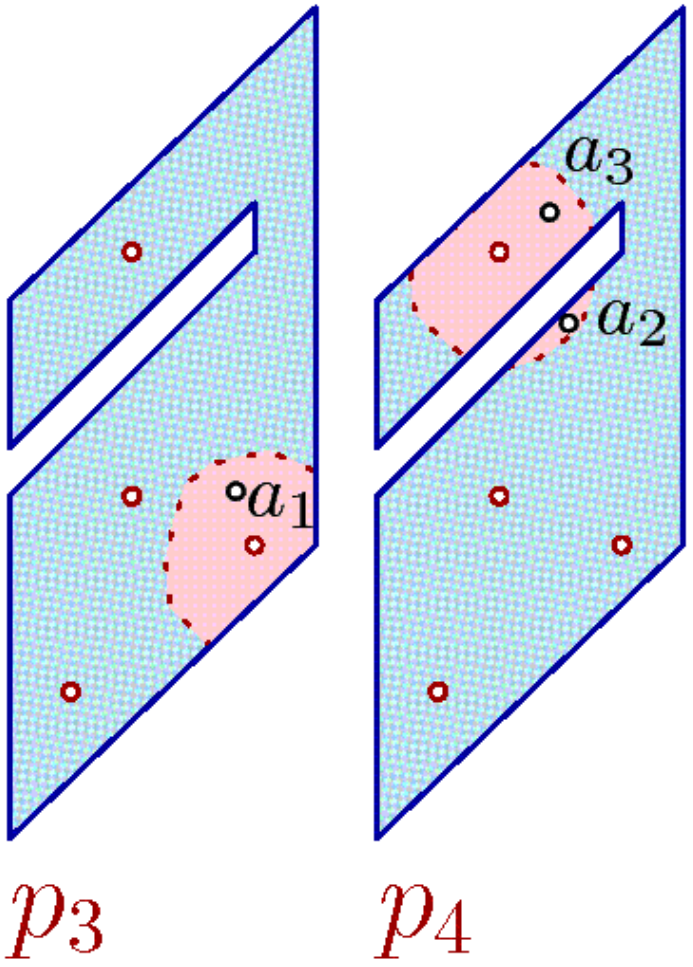


Partial Matching Algorithm



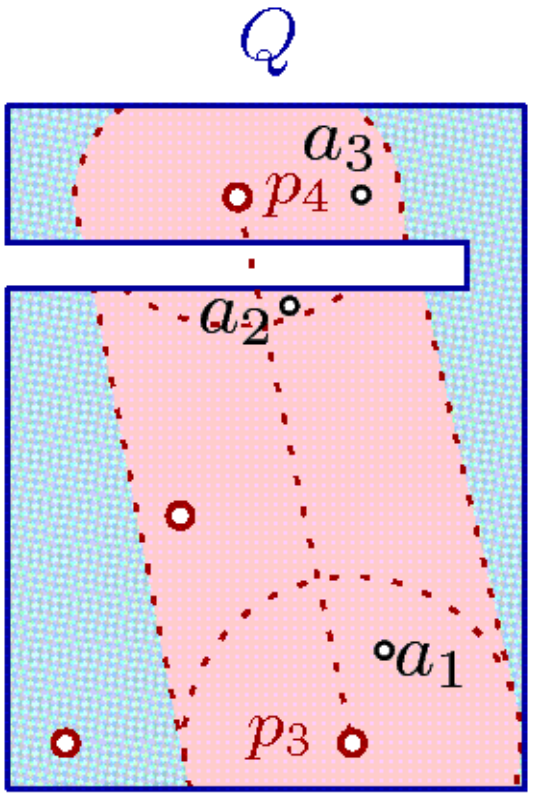
- Consider a pair of adjacent slices in the FSD.

Partial Matching Algorithm



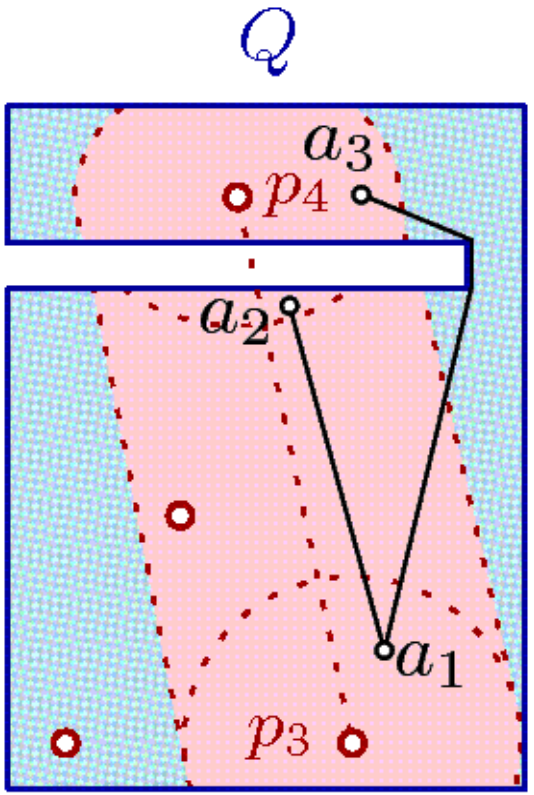
- Consider a pair of adjacent slices in the FSD.
- A point a_2 is reachable from a_1 in the FSD iff the shortest path in Q between a_2 and a_3 is within FD epsilon of the line segment p_4p_3.
- (it follows from a simple shortcutting argument that one can consider only shortest paths)

Partial Matching Algorithm



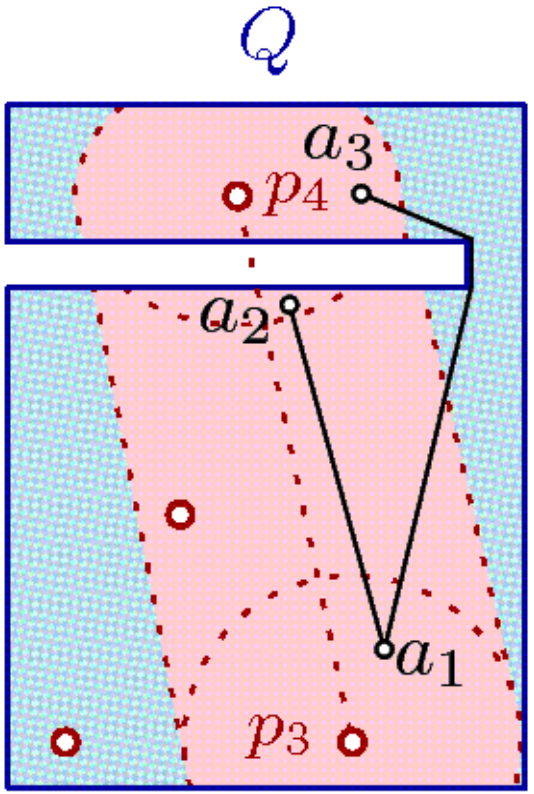
- Consider a pair of adjacent slices in the FSD.
- A point a_2 is reachable from a_1 in the FSD iff the shortest path in Q between a_2 and a_3 is within FD epsilon of the line segment p_4p_3.
- (it follows from a simple shortcutting argument that one can consider only shortest paths)

Partial Matching Algorithm



- Consider a pair of adjacent slices in the FSD.
- A point a_2 is reachable from a_1 in the FSD iff the shortest path in Q between a_2 and a_3 is within FD epsilon of the line segment p_4p_3.
- (it follows from a simple shortcutting argument that one can consider only shortest paths)

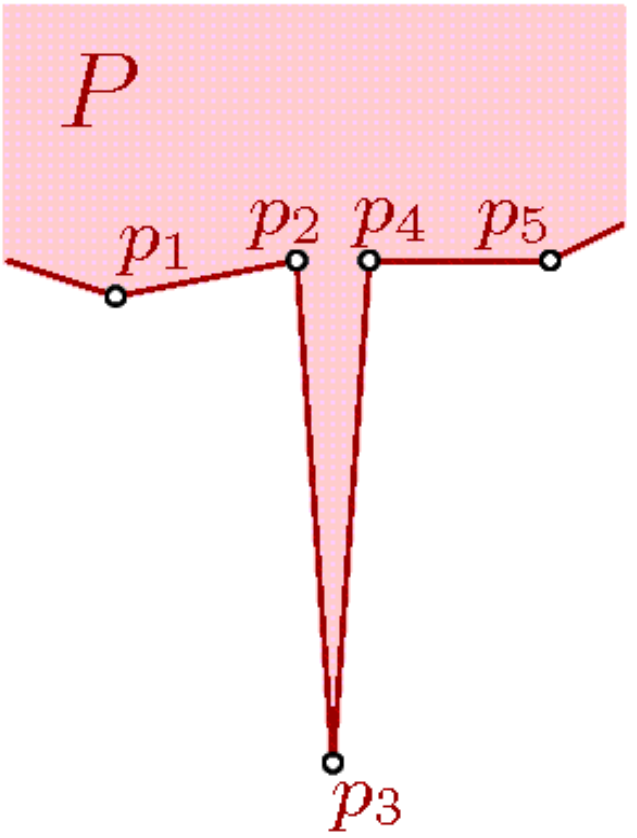
Partial Matching Algorithm



- So a_2 is reachable from a_1 but a_3 is not.
- There are many possible points to map to in the disc. How can this avoided?

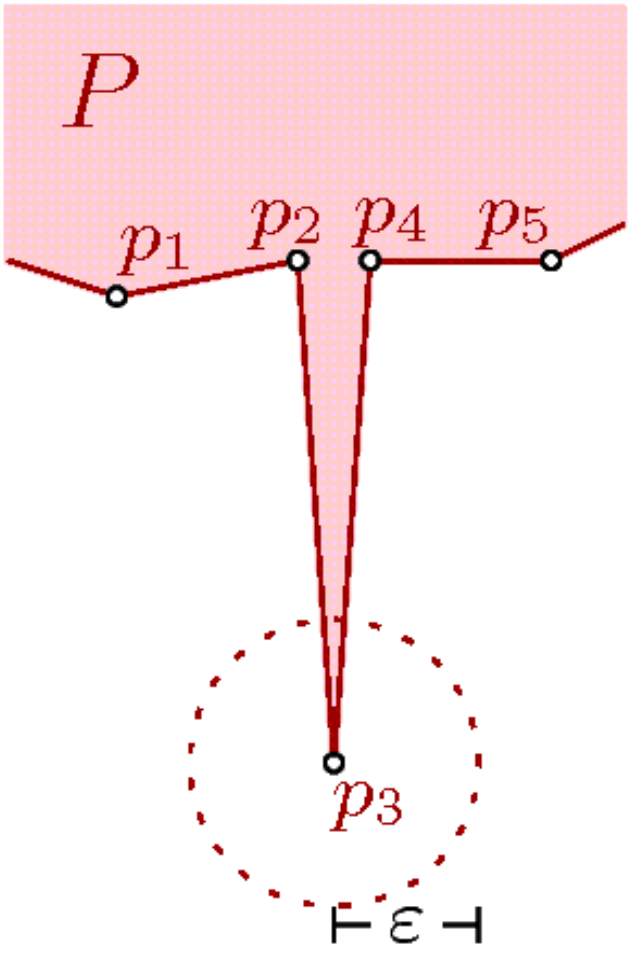
Partial Matching Algorithm: FSD

- Lets go through this simple example again but for the 3D FSD.
- Again consider the point p_3 in P.



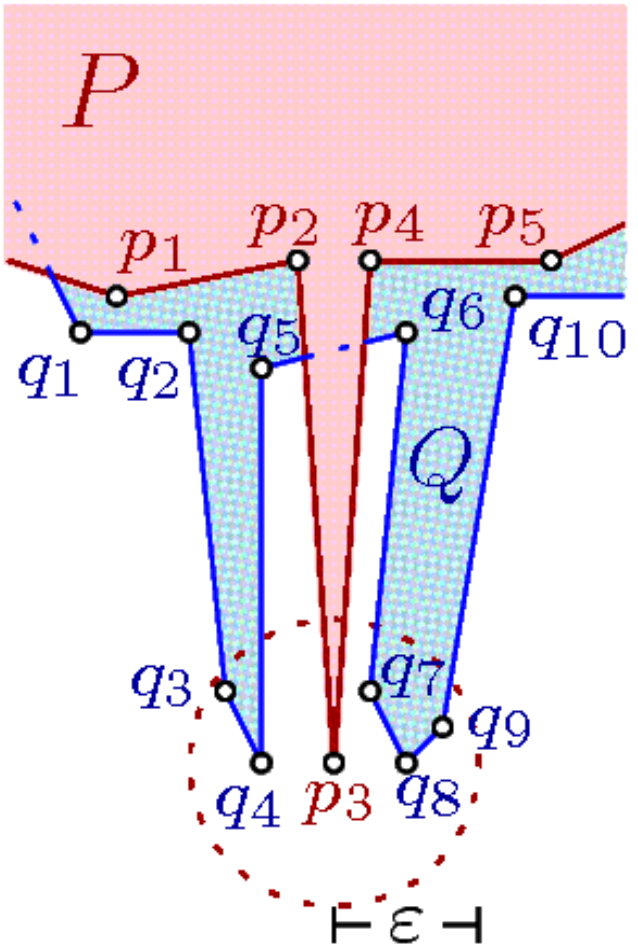
Partial Matching Algorithm: FSD

- Again the points within epsilon distance of p_3.



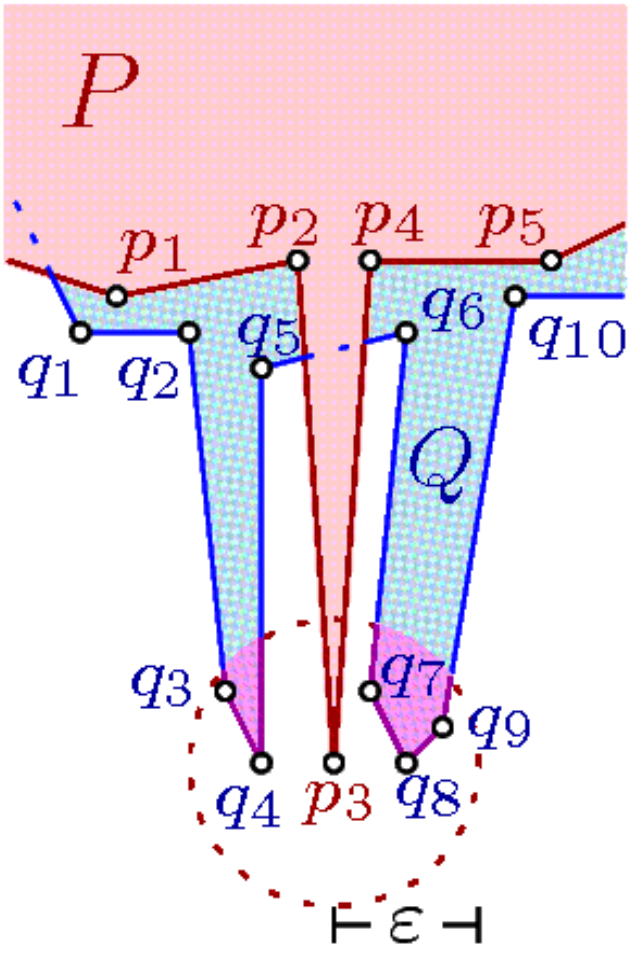
Partial Matching Algorithm: FSD

- Now we are interested in the points in Q which are in this disc. Those are the points that p_3 can be mapped to.



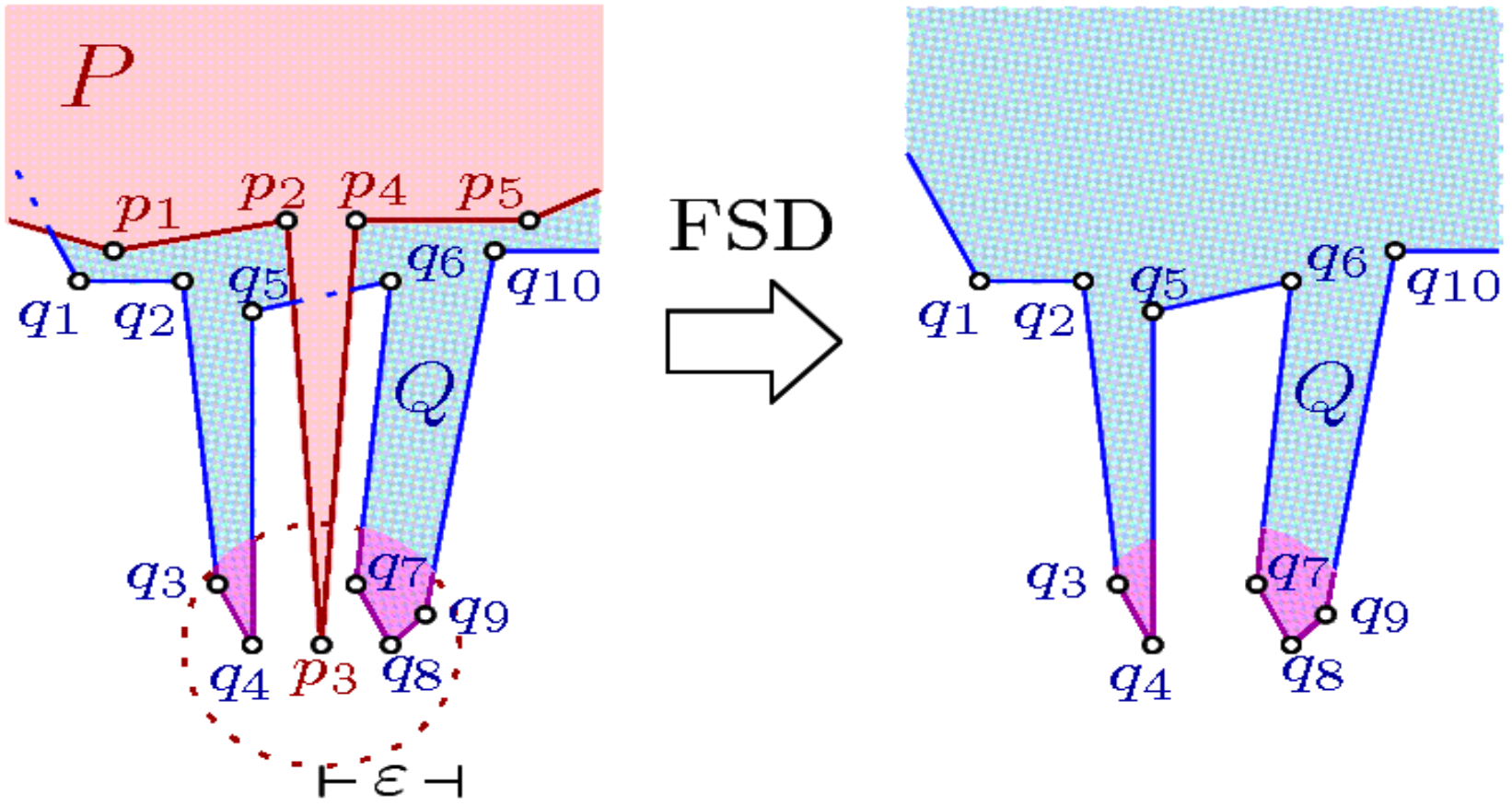
Partial Matching Algorithm: FSD

- Note the highlighted parts.



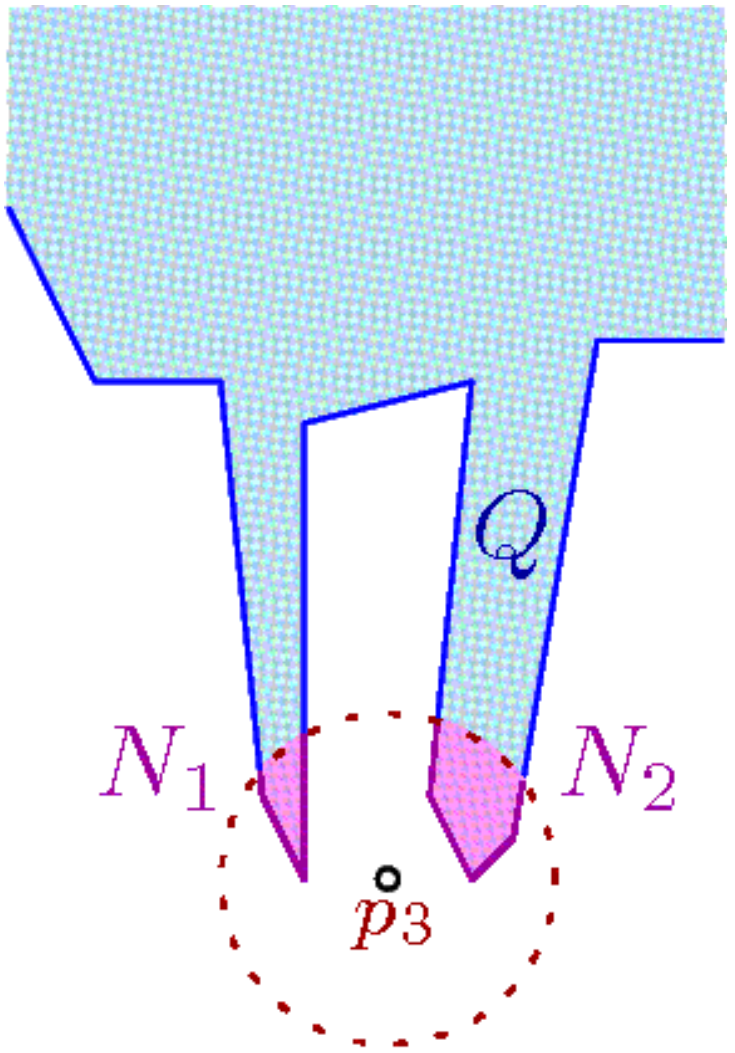
Partial Matching Algorithm: FSD

- Here is the part of the 3D FSD associated with the point p_3.

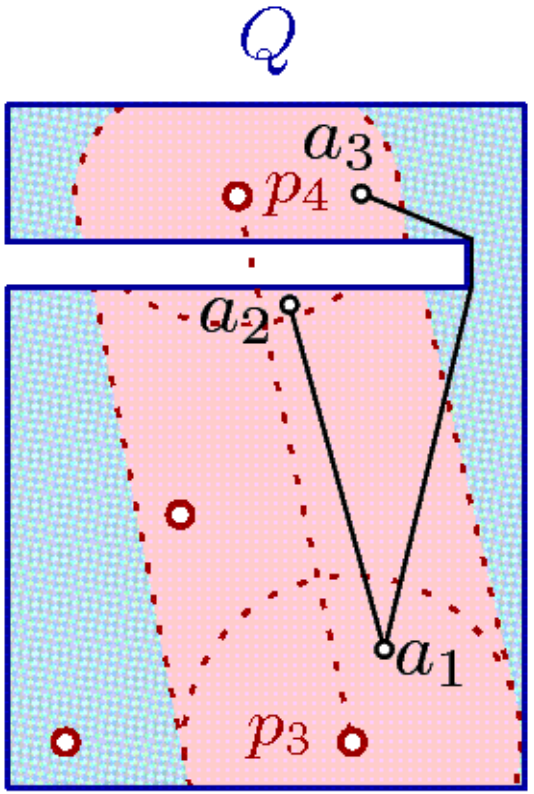


Partial Matching Algorithm: Neighborhoods

- We define a neighborhood of a point is a maximal connected subset (of the intersection of the epsilon disc around p and Q.)
- In our example there are two neighborhoods associated with the point p_3.

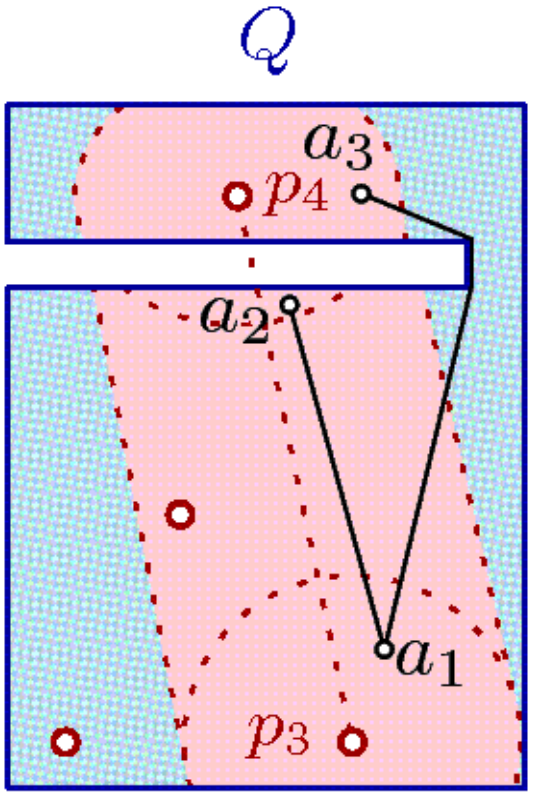


Partial Matching Algorithm



- These neighborhoods simplify the approach.
- Let N1 and N2 be a pair of neighborhoods.
- We prove that if a point in N1 is reachable from a point in N2 then every point in N1 is reachable from every point in N2

Partial Matching Algorithm



- From this we generate a polynomial time algorithm to find such a path.
- We actually get a set of neighborhoods associated with such a path.
- We refer to this as a (Q,eps)-valid set of neighborhoods.

Partial Matching Algorithm

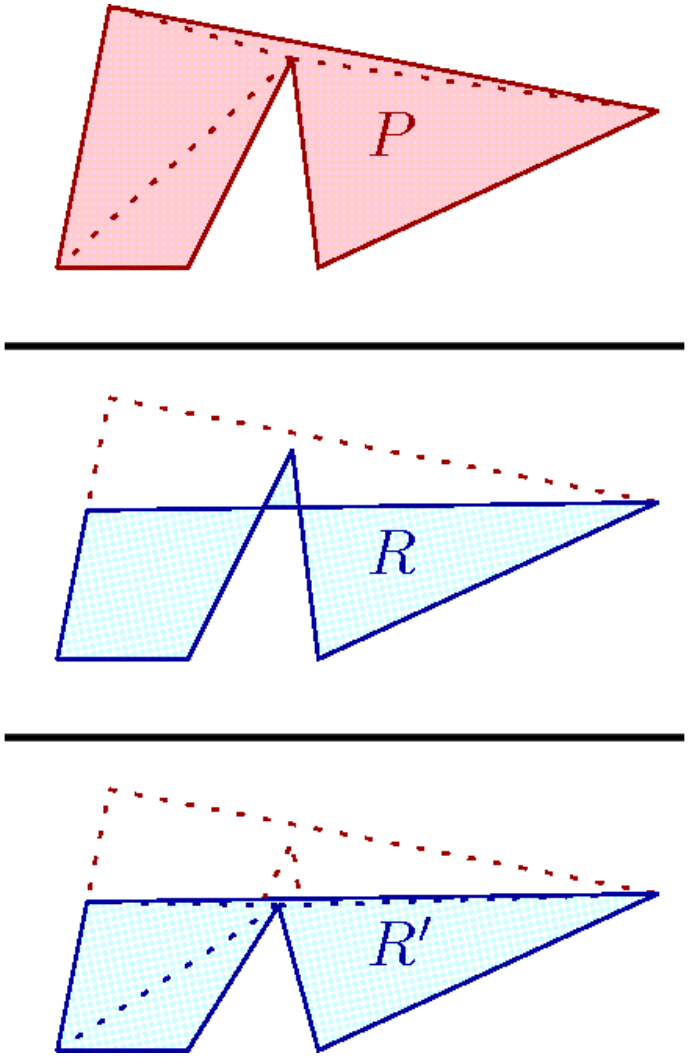
- So does this give us a simple polygon R in Q with FD epsilon to P?

Partial Matching Algorithm

- So does this give us a simple polygon R in Q with FD epsilon to P?
  - Not quite. The polygon we get in Q may not be simple.
  - Fortunately we can prove that an R which is a simple polygon can always be found.

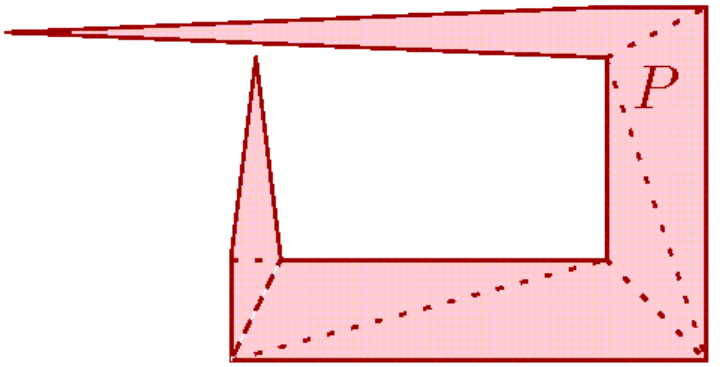
Partial Matching Algorithm: Proof

- This proof is somewhat long.
- We give a short summary of the algorithm to compute such a polygon R.



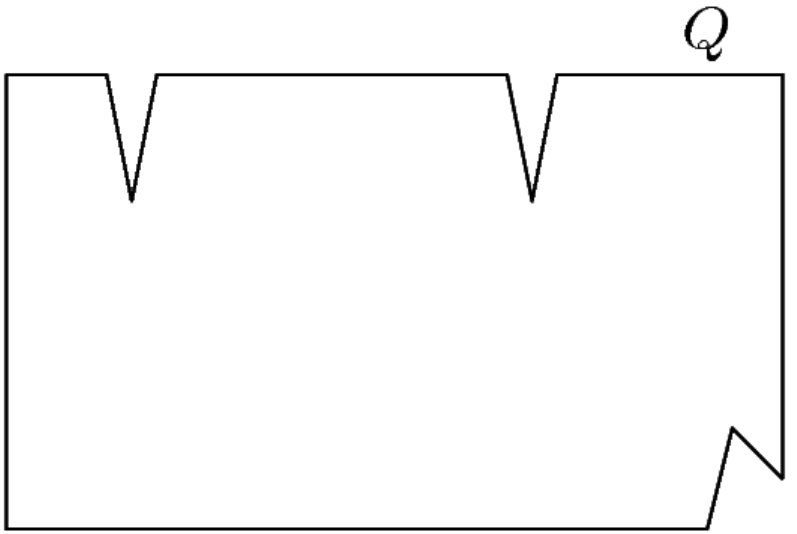
Partial Matching Algorithm: Proof

- We have a simple polygon P.



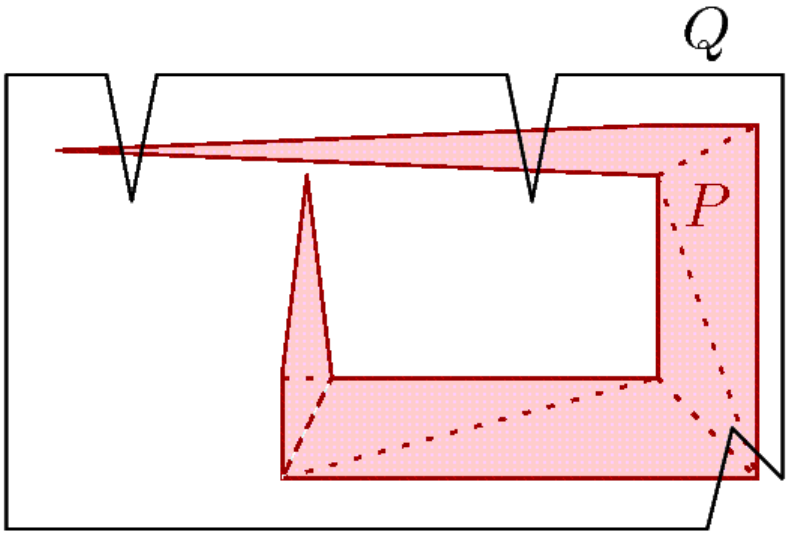
Partial Matching Algorithm: Proof

- We have a simple polygon P.
- We have a simple polygon Q.



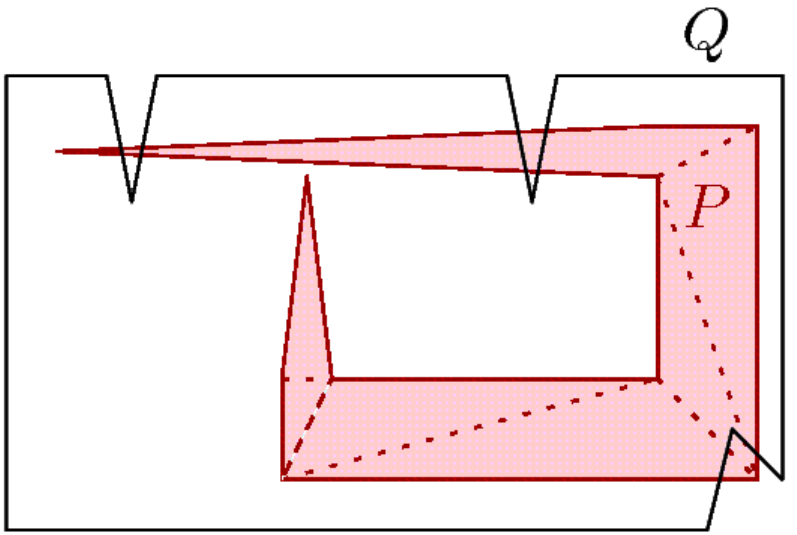
Partial Matching Algorithm: Proof

- We have a simple polygon P.
- We have a simple polygon Q.
- We have a (Q,eps)-valid set of neighborhoods for the points in P.



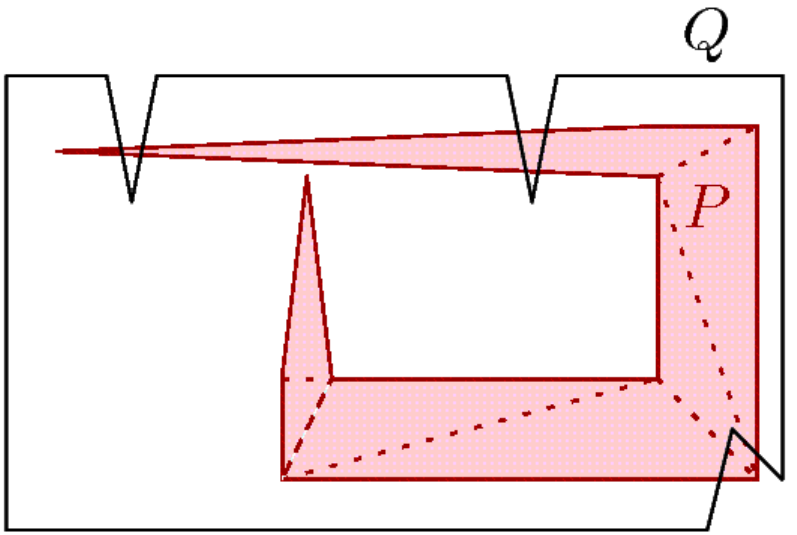
Partial Matching Algorithm: Proof

- To find such a R we iteratively map points of P to Q.
- At each iteration we show that the points can be mapped to form a simple polygon in Q.



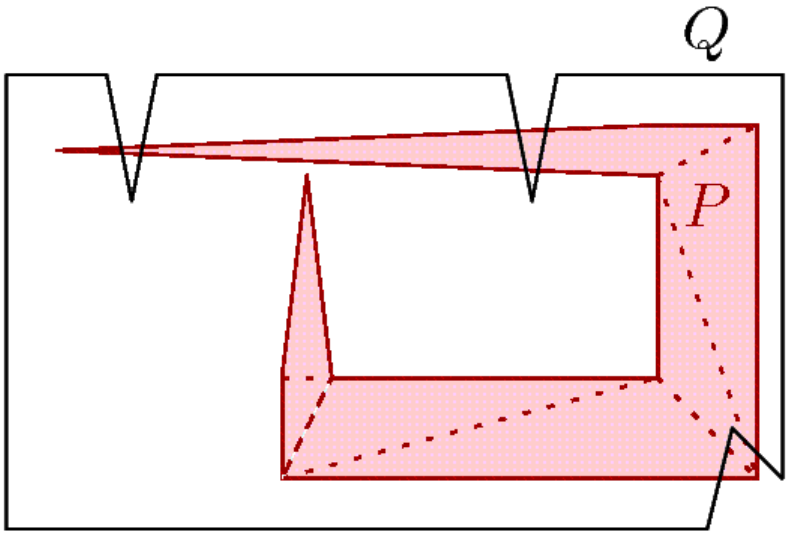
Partial Matching Algorithm: Proof

- To find such a R we iteratively map points of P to Q.
- At each iteration we show that the points can be mapped to form a simple polygon in Q.



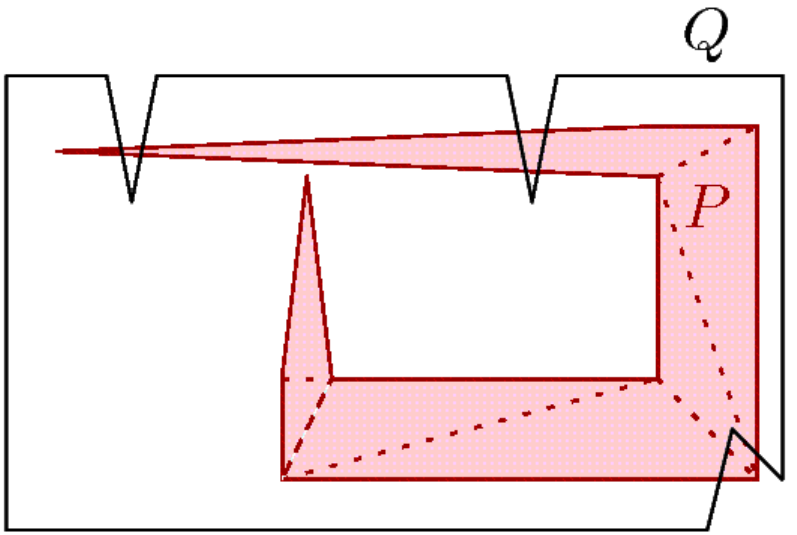
Partial Matching Algorithm: Proof

- Specifically, we allow remapping the points within their associated neighborhood in the valid set.
- By properties of neighborhoods the FD remains less than epsilon.



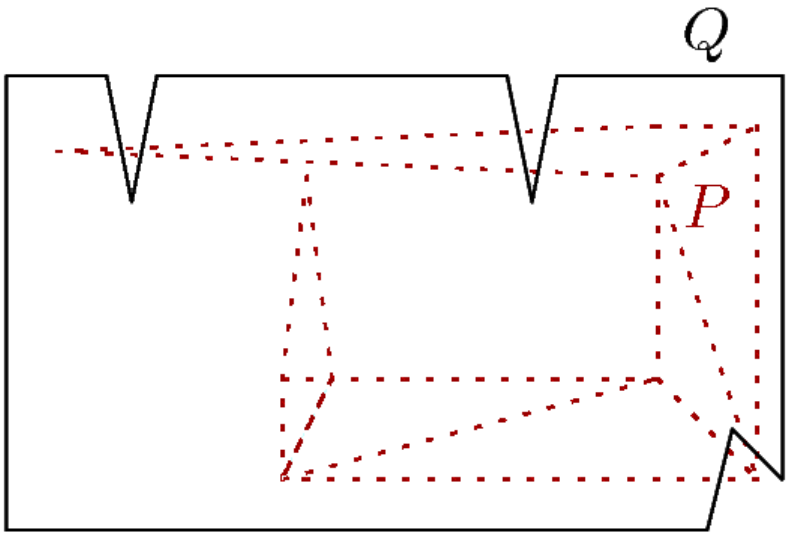
Partial Matching Algorithm: Proof

- Next we go through a simple example demonstrating the algorithm.



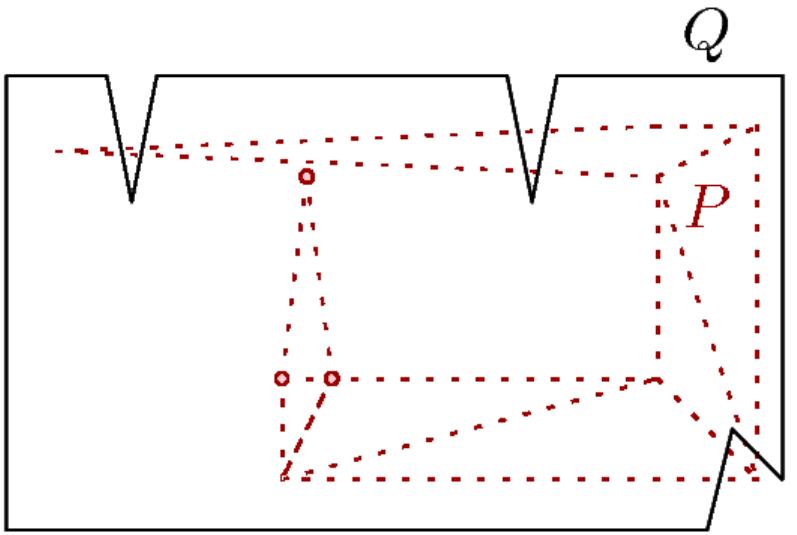
Partial Matching Algorithm: Proof

- So we want to iteratively map points of P to Q.



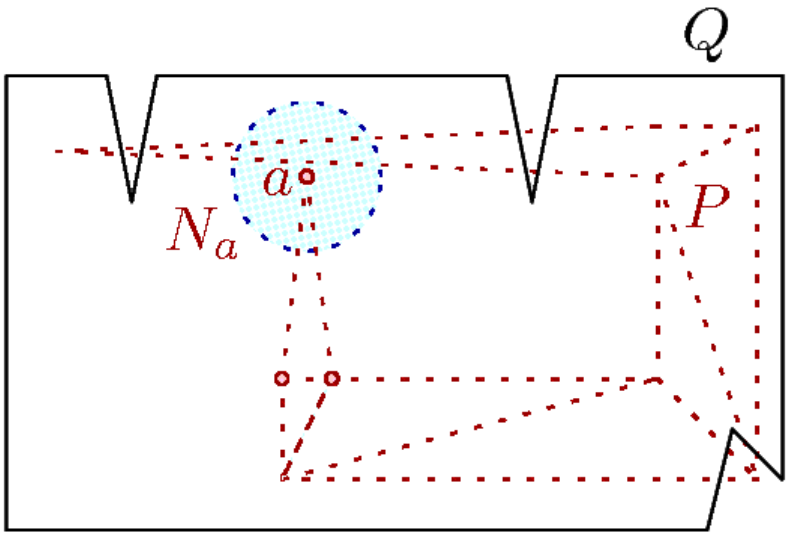
Partial Matching Algorithm: Proof

- So we want to iteratively map points of P to Q.
- In our initial step we choose three points which form a triangle in P.



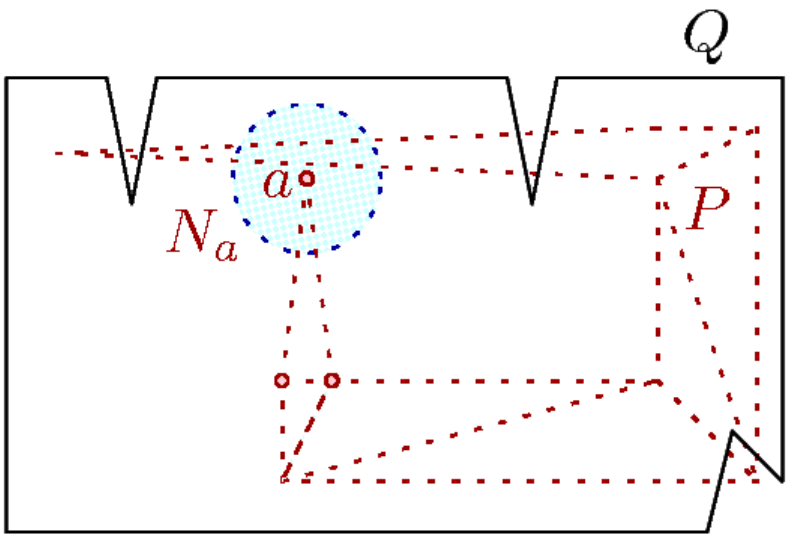
Partial Matching Algorithm: Proof

- So we want to iteratively map points of P to Q.
- In our initial step we choose three points which form a triangle in P.
- We want to map each point a in its associated neighborhood N_a in Q.



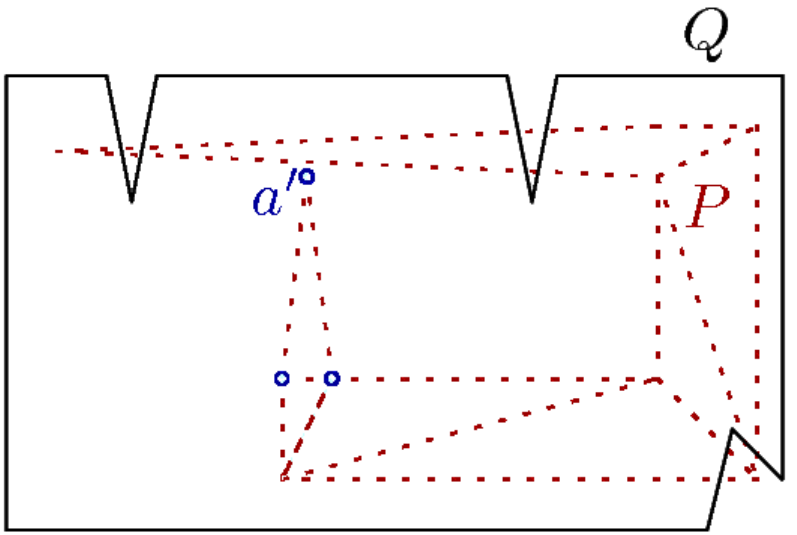
Partial Matching Algorithm: Proof

- Note that in this case a is actually in the neighborhood N_a.



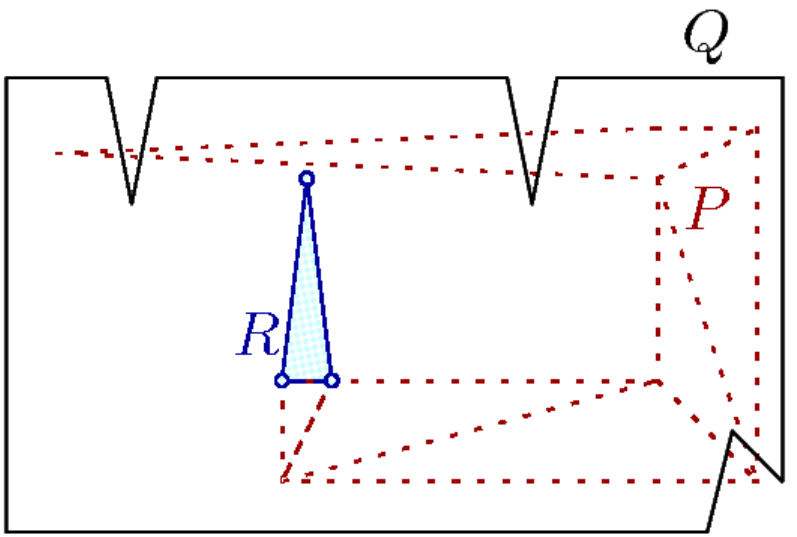
Partial Matching Algorithm: Proof

- Note that in this case a is actually in the neighborhood N_a.
- We can just choose its mapping a' to be a.



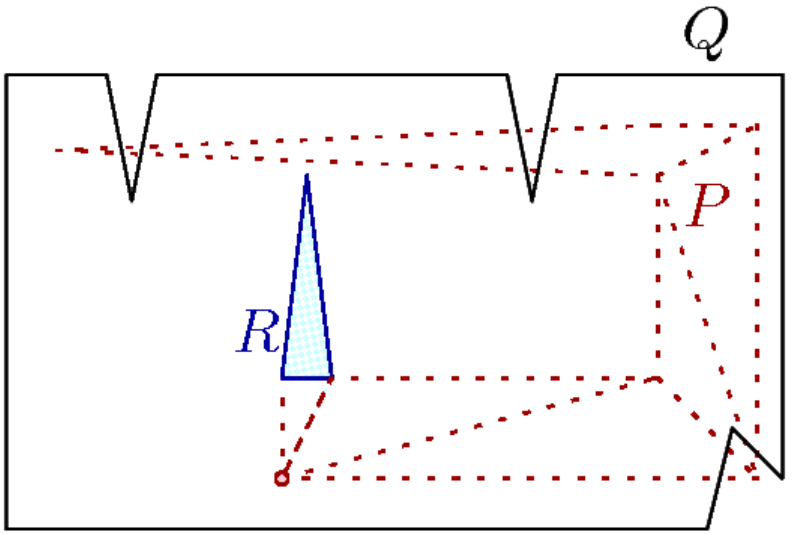
Partial Matching Algorithm: Proof

- Note that in this case a is actually in the neighborhood N_a.
- We can just choose its mapping a' to be a.
- (Similar for the other points of the triangle)



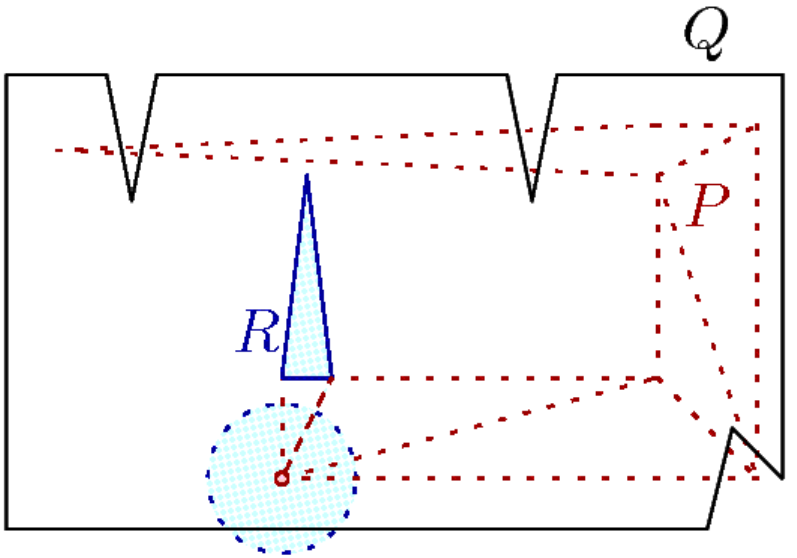
Partial Matching Algorithm: Proof

- In the iterative step, we add points in P which are connected to the points already mapped.
- In this case there is only one possible point to add.



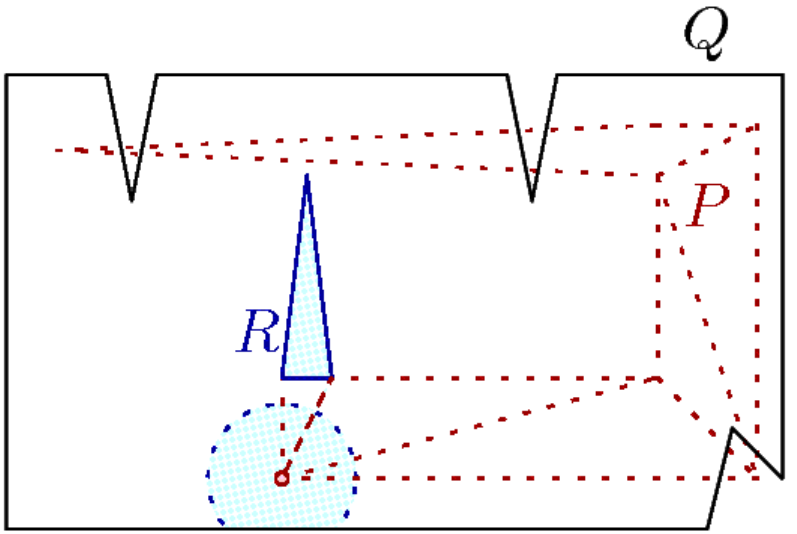
Partial Matching Algorithm: Proof

- In the iterative step, we add points in P which are connected to the points already mapped.
- In this case there is only one possible point to add.



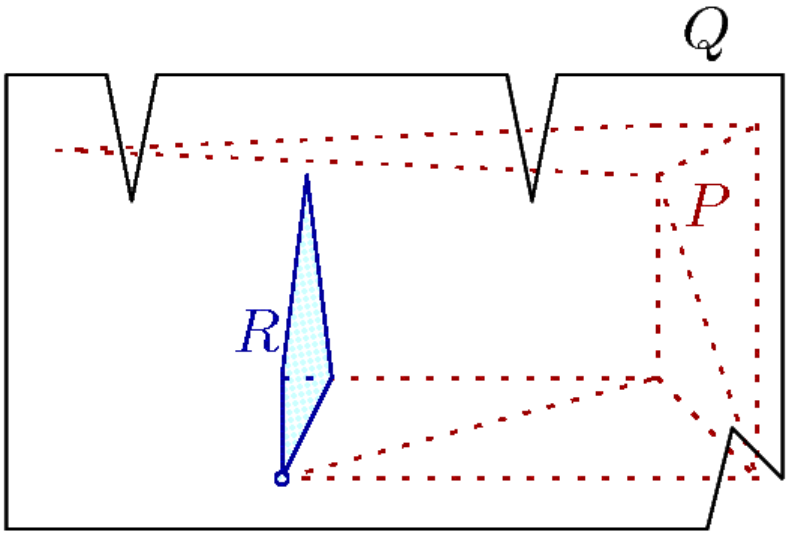
Partial Matching Algorithm: Proof

- In the iterative step, we add points in P which are connected to the points already mapped.
- In this case there is only one possible point to add.
- As mentioned earlier we want to map it inside the neighborhood associated with it in a valid set.



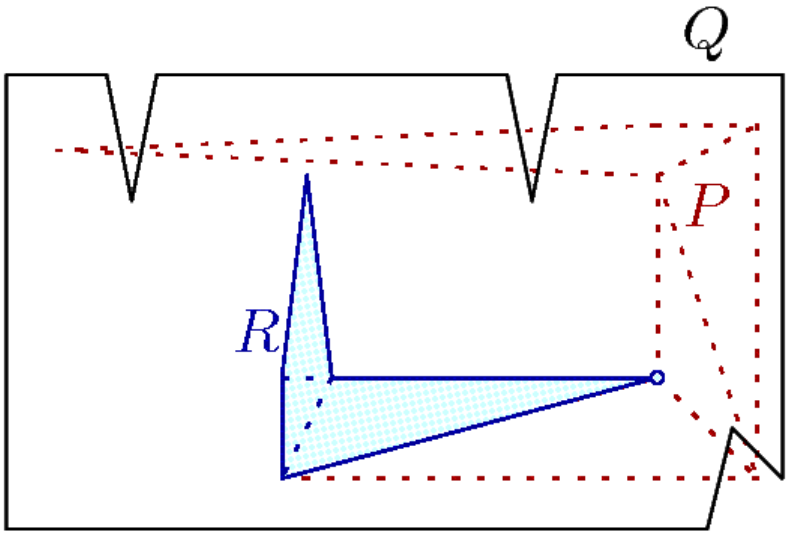
Partial Matching Algorithm: Proof

- In the iterative step, we add points in P which are connected to the points already mapped.
- In this case there is only one possible point to add.
- As mentioned earlier we want to map it inside the neighborhood associated with it in a valid set.
- (again we can just map to original point)



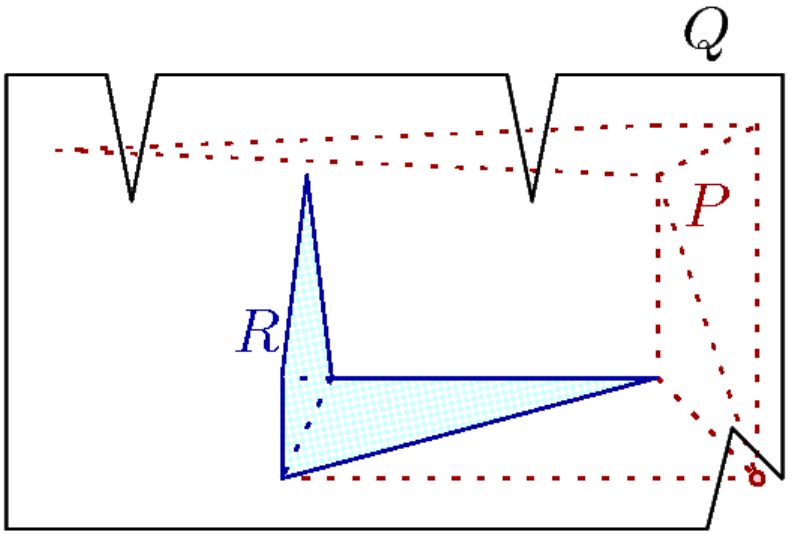
Partial Matching Algorithm: Proof

- The next point is easy as well.



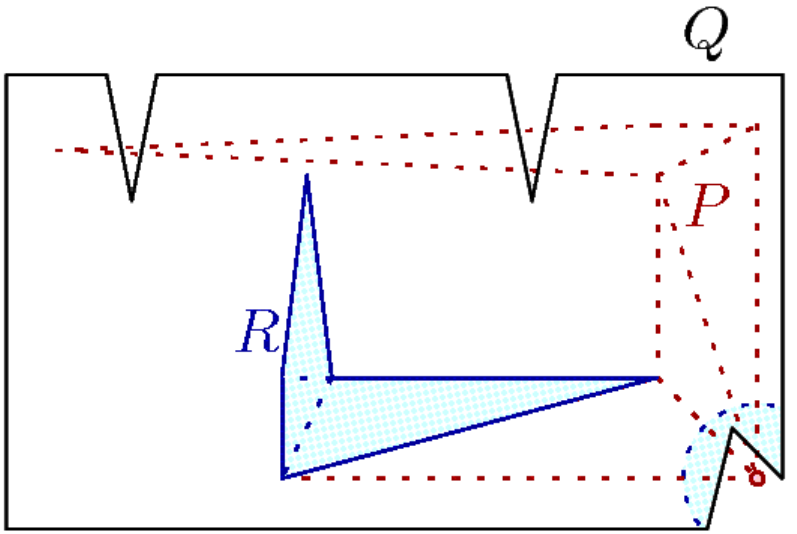
Partial Matching Algorithm: Proof

- This point is a bit different.



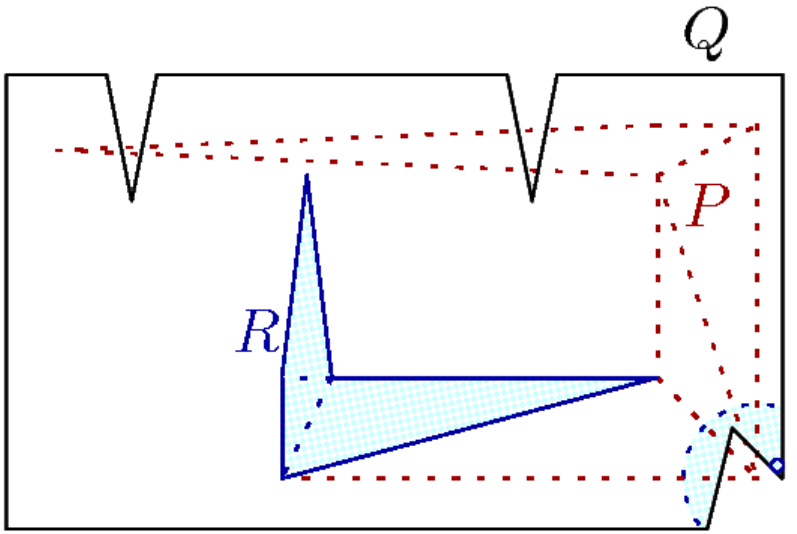
Partial Matching Algorithm: Proof

- This point is a bit different.
- The neighborhood does not contain the original point.



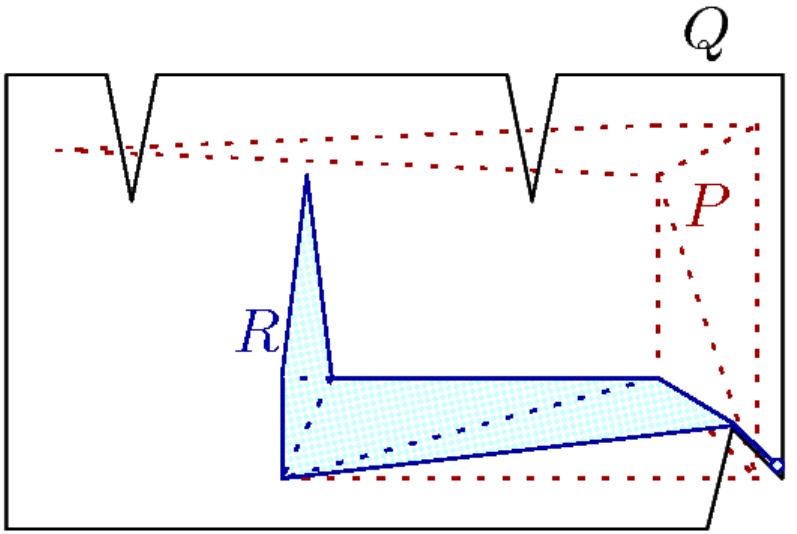
Partial Matching Algorithm: Proof

- This point is a bit different.
- The neighborhood does not contain the original point.
- It has to be mapped inside Q.



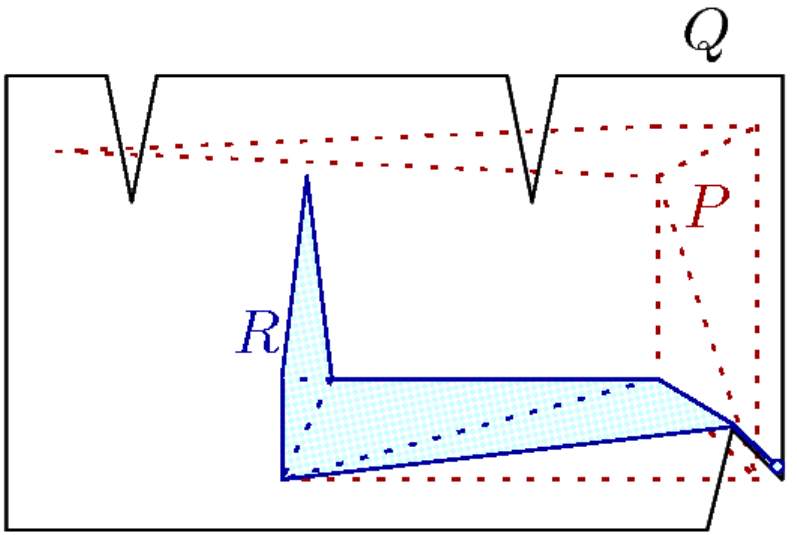
Partial Matching Algorithm: Proof

- This point is a bit different.
- The neighborhood does not contain the original point.
- It has to be mapped inside Q.
- The previous points are connected to it via shortest paths in Q.



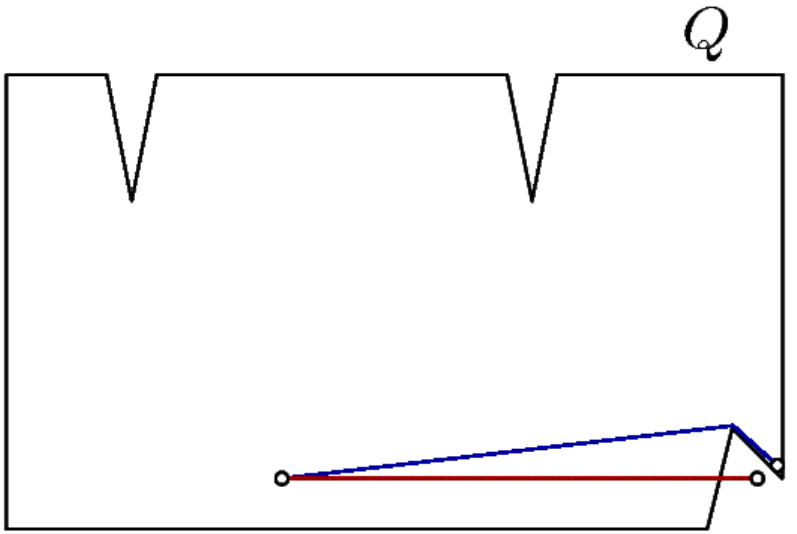
Partial Matching Algorithm: Proof

- Using shortest paths is okay by properties of a valid set of neighborhoods.



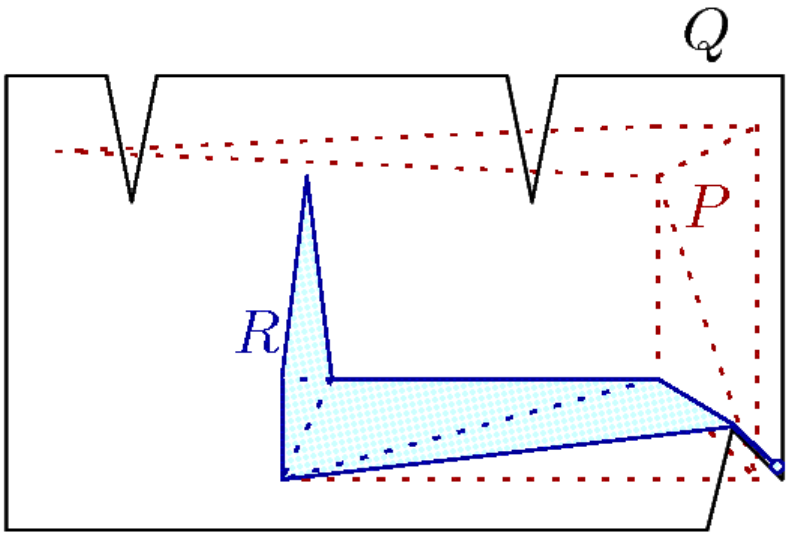
Partial Matching Algorithm: Proof

- Using shortest paths is okay by properties of a valid set of neighborhoods.
- Line segment on the boundary of P and a shortest path in Q.



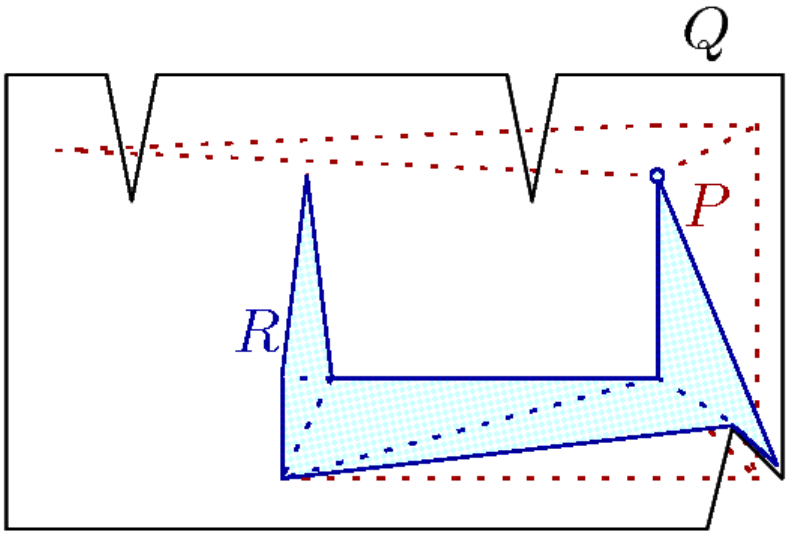
Partial Matching Algorithm: Proof

- Using shortest paths is okay by properties of a valid set of neighborhoods.
- Line segment on the boundary of P and a shortest path in Q.



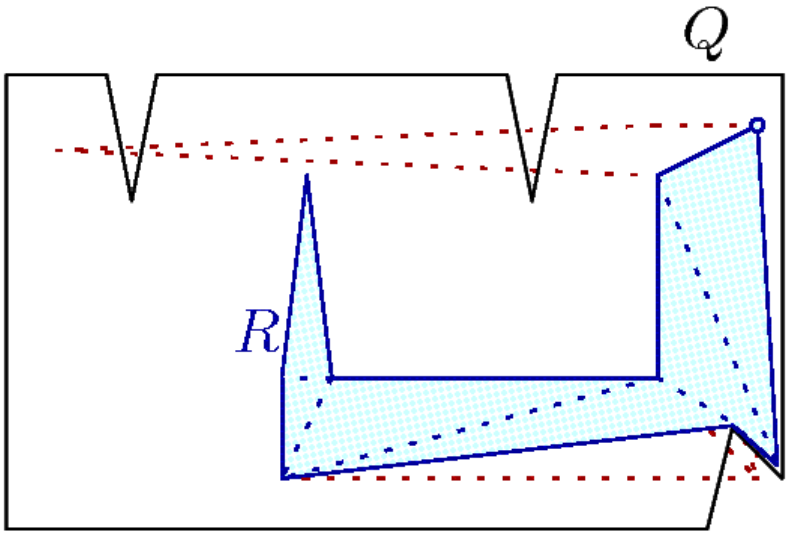
Partial Matching Algorithm: Proof

- The next point is easy.



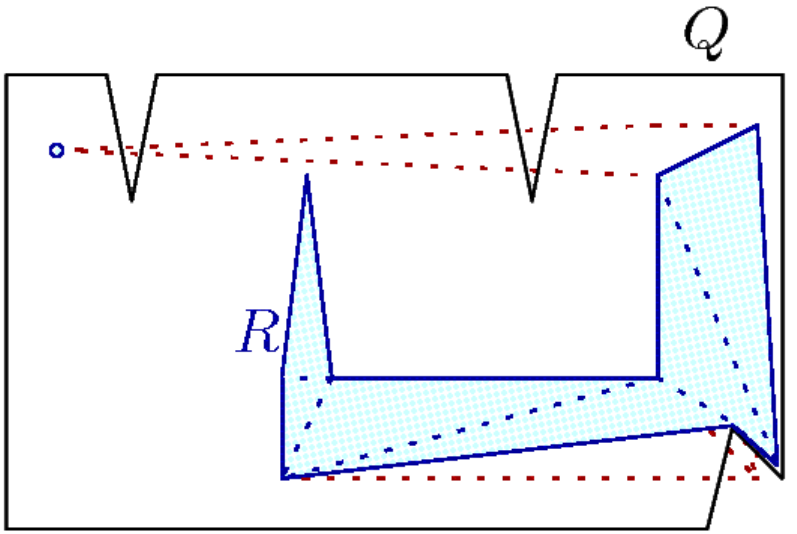
Partial Matching Algorithm: Proof

- The next point is again easy.



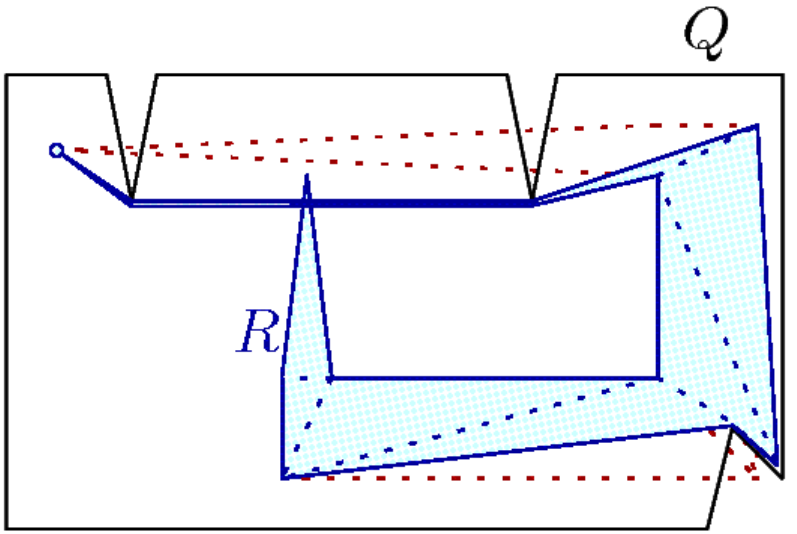
Partial Matching Algorithm: Proof

- This point is also a bit different.



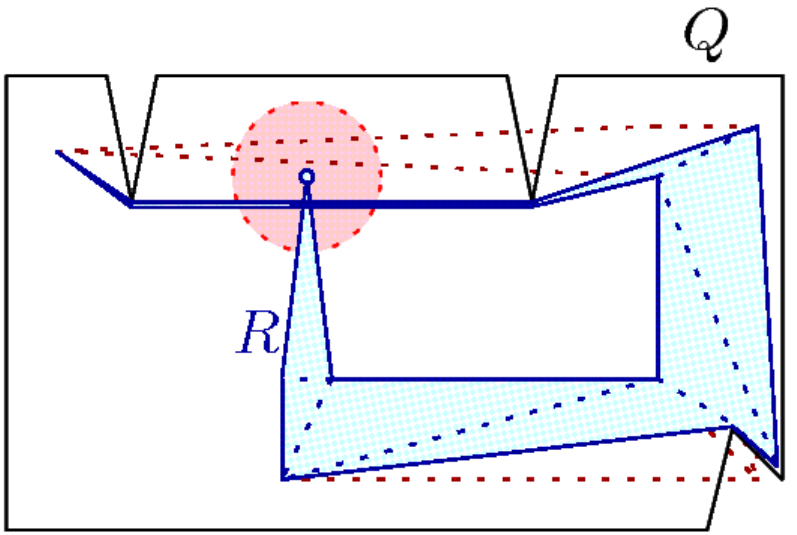
Partial Matching Algorithm: Proof

- This point is also a bit different.
- Adding the point and its associated shortest paths yields a self-intersecting polygon R.



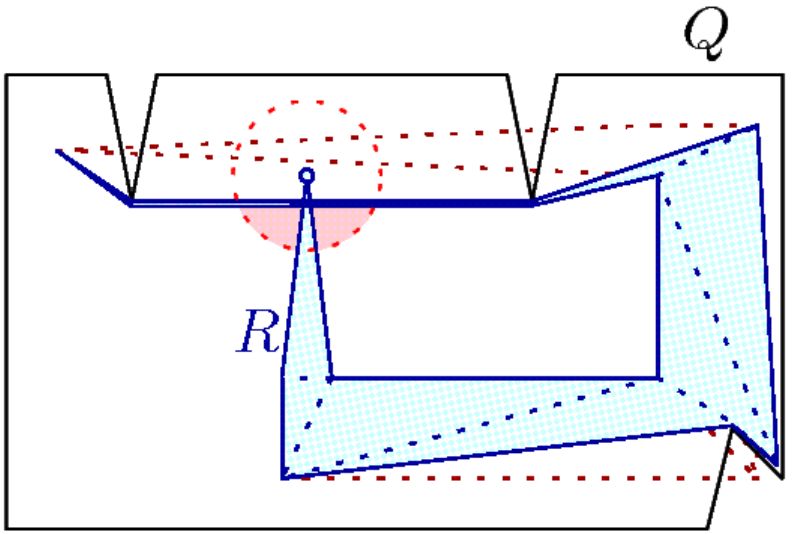
Partial Matching Algorithm: Proof

- This point is also a bit different.
- Adding the point and its associated shortest paths yields a self-intersecting polygon R.
- The neighborhood around the point a is crossed by the added image curves.



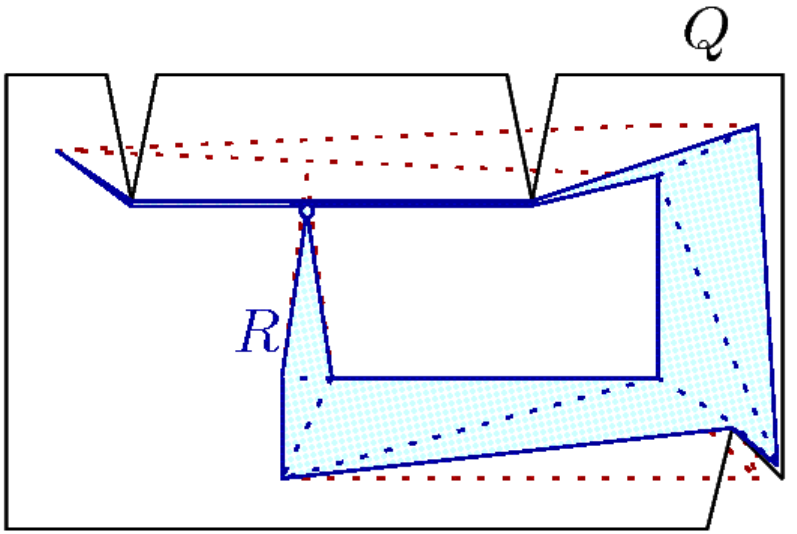
Partial Matching Algorithm: Proof

- We now need to remap a to a point below the shortest paths.



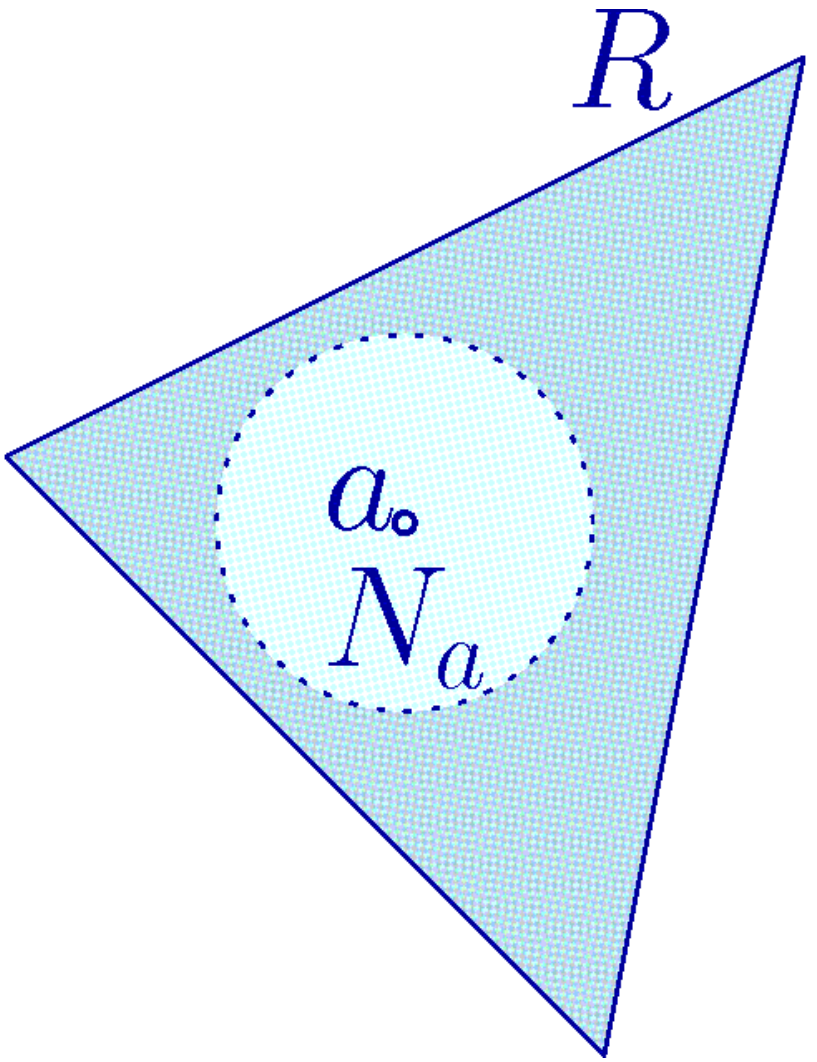
Partial Matching Algorithm: Proof

- We now need to remap a to a point below the shortest paths.



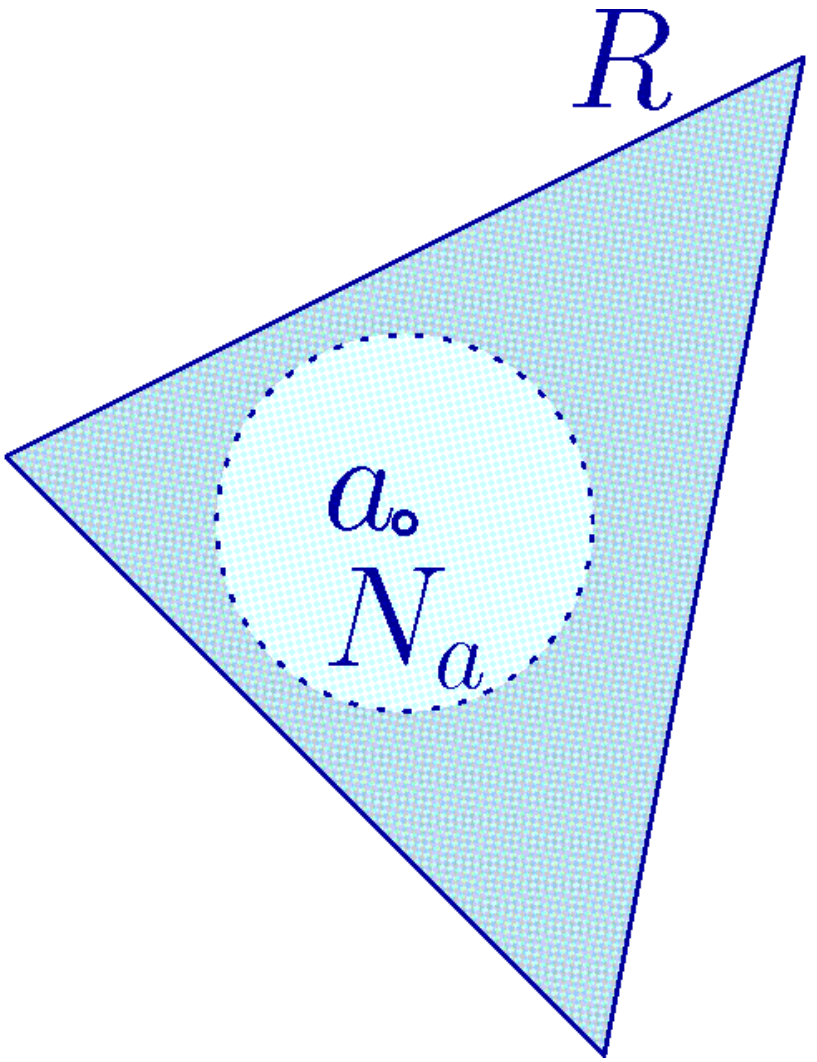
Partial Matching Algorithm: Proof

- We now need to remap a to a point below the shortest paths.
- We prove that the neighborhood of a point will never be completely covered by R.



Partial Matching Algorithm: Proof

- We now need to remap a to a point below the shortest paths.
- We prove that the neighborhood of a point will never be completely covered by R.
- We also update the point in a way to avoid cascading chains of updates.



Partial Matching Algorithm: Constrained Embedding Problem

- This proof requires solving a variant of the constrained embedding problem.
- We give more details about this in the future work section.

Partial Matching Algorithm

- The described algorithm solves the decision variant.
- We can optimize epsilon such that there exists a simple polygon R in Q with FD epsilon to P.
  - Similar to the simple polygons algorithm: We compute the constraints between P and Q. Sort them. And perform a binary search on these using the decision variant outlined above.

Partial Matching Algorithm: Conclusion

- We presented the first algorithm for computing partial FD between surfaces. This algorithm runs in polynomial time.
- In the future it would be interesting to consider other variants of partial FD.
- It may also be interesting to consider extending this algorithm other classes of surfaces.