Facial Expression Transfer

Team P03

Ang Yihao, Liu Xiaofei, Zheng Yinjie, Zhou Xiaozhou

Outline

I. Problem Definition
II. Algorithm Design
III. Justifications
IV. Examples

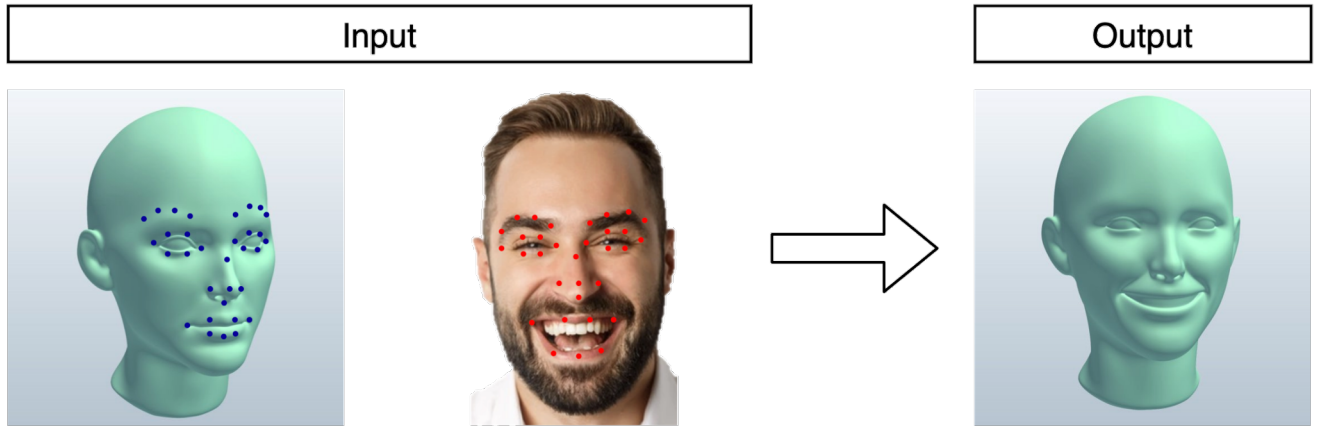
Motivation

- Observation of Facial Expression Transfer
  - Entertainment
  - Online Meeting

- https://voi.id/en/technology/27638/how-to-use-faceapp-on-tiktok-to-change-mens-faces-to-girls
- https://restream.io/blog/how-to-multistream-with-zoom/

Problem Objective

- To generate facial expressions from a target image to a 3D model.
  - A reference 3D model F with neutral expression (person A)
  - One depth image T with the desired expressions (person B)
  - Base points on both A and B with known correspondence
  - To generate similar expressions on the reference 3D model (person A)



Problem Objective

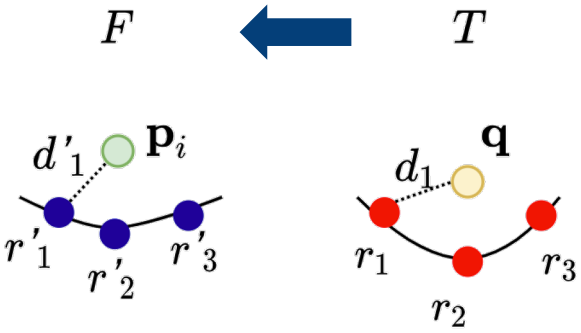
- To generate facial expressions from a target image to a 3D model.
  - Based points on both A and B with known correspondence
    - 32 points from the contour of eyes, nose, and mouth

Problem Definition

- Objective
  - Given a 3D face model with neutral expressions, one depth face with desired expressions, and some base points on both face with known correspondence, to map desired expressions from target on the reference to get a new 3D model
- Inputs
  - Reference model F with m mesh points p_i, where $i = 1, \dots, m$, and 32 base points r'_l, where $l = 1, \dots, 32$.
  - Target depth image T with n points q_j, where $\mathrm{j} = 1, \dots, n$, and 32 base points r_l, where $l = 1, \dots, 32$.
- Outputs
  - Resultant model R with m mesh points p'_i, where $i = 1, \dots, m$
  - Correspondence function $c: T \to F$
  - Shape change function $f: F \to R$, such that $\mathrm{p}'_i = f(\mathrm{p}_i)$, for $i = 1, \dots, m$

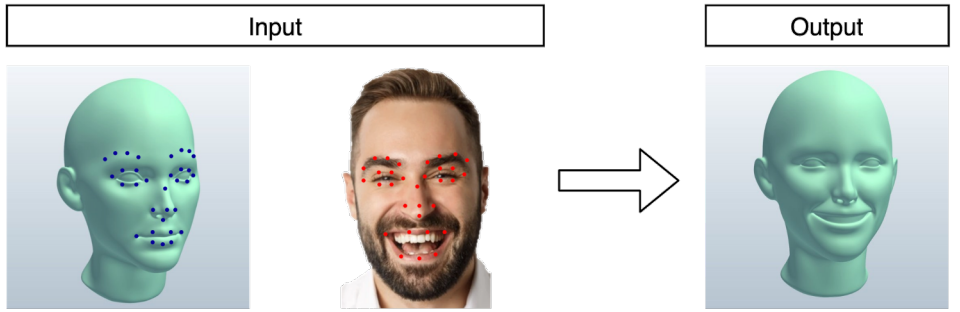
Problem Definition

- Why correspondence function?
  - $T \rightarrow F: c(\mathrm{q}) = \mathrm{p}_i$
  - F and T are from different people, we need to find the correspondence!



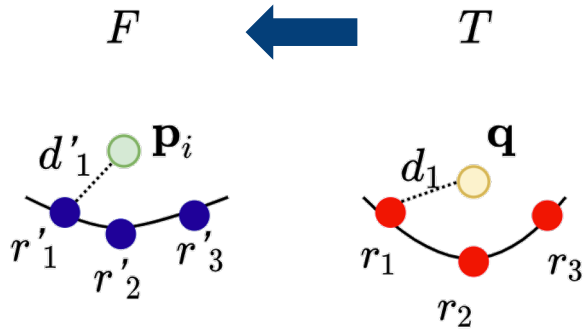
  - For each q_j on T, we find three nearest base points $\mathrm{r}_1, \mathrm{r}_2, \mathrm{r}_3$ that are not collinear
  - The distance between q_j and $\mathrm{r}_1, \mathrm{r}_2, \mathrm{r}_3$ are d_1, d_2, d_3

  - The corresponding base points of $\mathrm{r}_1, \mathrm{r}_2, \mathrm{r}_3$ on F would be $\mathrm{r}'_1, \mathrm{r}'_2, \mathrm{r}'_3$
  - The distance between the corresponding point of q_j on F and $\mathrm{r}'_1, \mathrm{r}'_2, \mathrm{r}'_3$ are d'_1, d'_2, d'_3



Problem Definition

- Why correspondence function?
  - $T \rightarrow F$: $c(\mathrm{q}) = \mathrm{p}_i$
  - F and T are from different people, we need to find the correspondence!



  - For each q_j on T, we find three nearest base points $\mathrm{r}_1, \mathrm{r}_2, \mathrm{r}_3$ that are not collinear
  - The distance between q_j and $\mathrm{r}_1, \mathrm{r}_2, \mathrm{r}_3$ are d_1, d_2, d_3
  - The corresponding base points of $\mathrm{r}_1, \mathrm{r}_2, \mathrm{r}_3$ on F would be $\mathrm{r}'_1, \mathrm{r}'_2, \mathrm{r}'_3$
  - The distance between the corresponding point of q_j on F and $\mathrm{r}'_1, \mathrm{r}'_2, \mathrm{r}'_3$ are d'_1, d'_2, d'_3
  - Ideal case: same faces, same expressions → $\frac{d_1}{d_2} = \frac{d_1}{d_2}, \frac{d_2}{d_3} = \frac{d_2}{d_3}, \frac{d_3}{d_1} = \frac{d_3}{d_1}$
  - Practical case: different faces, same expressions → $\frac{d_1}{d_2} = \frac{d'_1}{d'_2}, \frac{d_2}{d_3} = \frac{d'_2}{d'_3}, \frac{d_3}{d_1} = \frac{d'_3}{d'_1}$
  - In our case: different faces, different expressions → $\left|\frac{d_1}{d_2} - \frac{d'_1}{d'_2}\right| + \left|\frac{d_2}{d_3} - \frac{d'_2}{d'_3}\right| + \left|\frac{d_3}{d_1} - \frac{d'_3}{d'_1}\right|$ is small

Problem Definition

- Objective
  - Given a 3D face model with neutral expressions, one depth face with desired expressions, and some base points on both face with known correspondence, to map desired expressions from target on the reference to get a new 3D model
- Inputs
  - Reference model F with m mesh points p_i, where $i = 1, \dots, m$, and 32 base points r'_l, where $l = 1, \dots, 32$.
  - Target depth image T with n points q_j, where $\mathrm{j} = 1, \dots, n$, and 32 base points r_l, where $l = 1, \dots, 32$.
- Outputs
  - Resultant model R with m mesh points p'_i, where $i = 1, \dots, m$
  - Correspondence function $c: T \to F$
  - Shape change function $f: F \to R$, such that $\mathrm{p}'_i = f(\mathrm{p}_i)$, for $i = 1, \dots, m$
- Why shape change function?
  - This is the major goal!

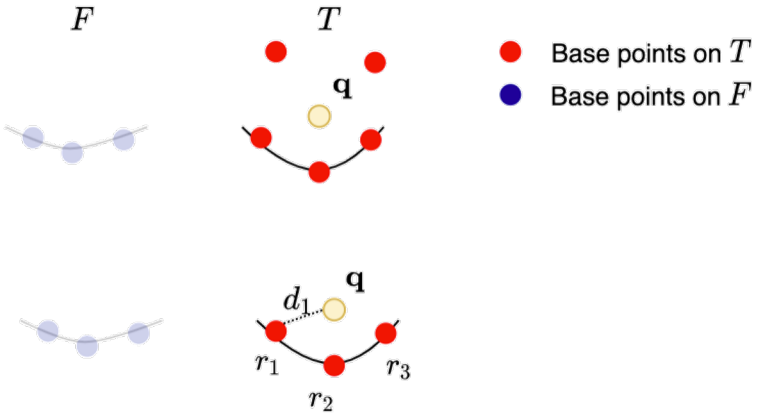
Problem Definition

- Inputs
  - Reference model F with m mesh points p_i, where $i = 1, \dots, m$, and 32 base points r'_l, where $l = 1, \dots, 32$.
  - Target depth image T with n points q_j, where $\mathrm{j} = 1, \dots, n$, and 32 base points r_l, where $l = 1, \dots, 32$.
- Outputs
  - Resultant model R with m mesh points p'_i, where $i = 1, \dots, m$
  - Correspondence function $c: T \to F$
  - Shape change function $f: F \to R$, such that $\mathrm{p}'_i = f(\mathrm{p}_i)$, for $i = 1, \dots, m$
- Optimization Objective
  - Close matching from R to T: Minimize $\sum_j \left\| \mathrm{p}'_j - \mathrm{q}_j \right\|^2$
- Constraints
  - Good correspondence from T to F: For $\mathrm{p}_i = c(\mathrm{q}_j)$,
    - Near enough: $\left\| \mathrm{p}_i - \mathrm{q}_j \right\|$ is small
    - p_i and q_j have three corresponding base points
    - The sum of error of the ratio of the distance between q_j and its three nearest base points and the distance between $c(\mathrm{q}_j)$ and its three nearest base points is small
  - R has normal face shape: $\| f(\mathrm{p}_i) - \mathrm{p}_i \|$ is small.
  - After shape change, no surface self-intersection

Algorithm Design

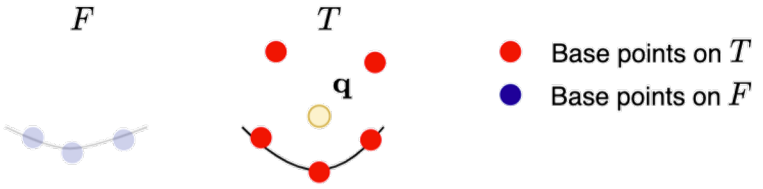
- How to search correspondence?
  - Most of points have unknown correspondence, but base points have known correspondence!

1. For a q on T, we find three nearest base points $\mathrm{r}_1, \mathrm{r}_2, \mathrm{r}_3$ that are not collinear. The distance between q_j and $\mathrm{r}_1, \mathrm{r}_2, \mathrm{r}_3$ are d_1, d_2, d_3.

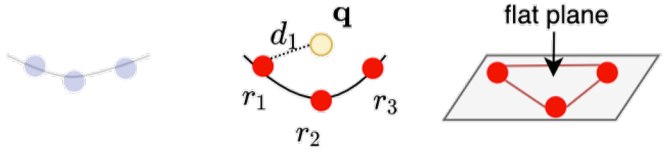


Algorithm Design

- How to search correspondence?
  - Most of points have unknown correspondence, but base points have known correspondence!



1. For a q on T, we find three nearest base points $\mathrm{r}_1, \mathrm{r}_2, \mathrm{r}_3$ that are not collinear. The distance between q_j and $\mathrm{r}_1, \mathrm{r}_2, \mathrm{r}_3$ are d_1, d_2, d_3

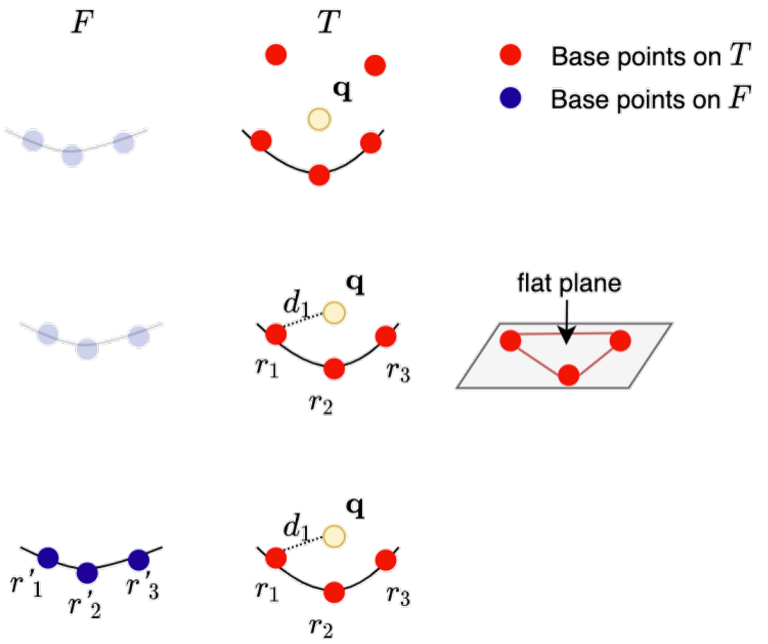


- Why three nearest base points?
  - Adopt $\left|\frac{d_1}{d_2} - \frac{d_1'}{d_2'}\right| + \left|\frac{d_2}{d_3} - \frac{d_2'}{d_3'}\right| + \left|\frac{d_3}{d_1} - \frac{d_3'}{d_1'}\right|$ to describe the "difference" between q and its corresponding point. So they should be on a flat surface.
  - Human face is not flat.
  - Three base points form small flat plane.
  - We assume this small flat plane is on the human face, because these three base points are close enough.
  - We get a small flat plane by three base points.

Algorithm Design

- How to search correspondence?
  - Most of points have unknown correspondence, but base points have known correspondence!

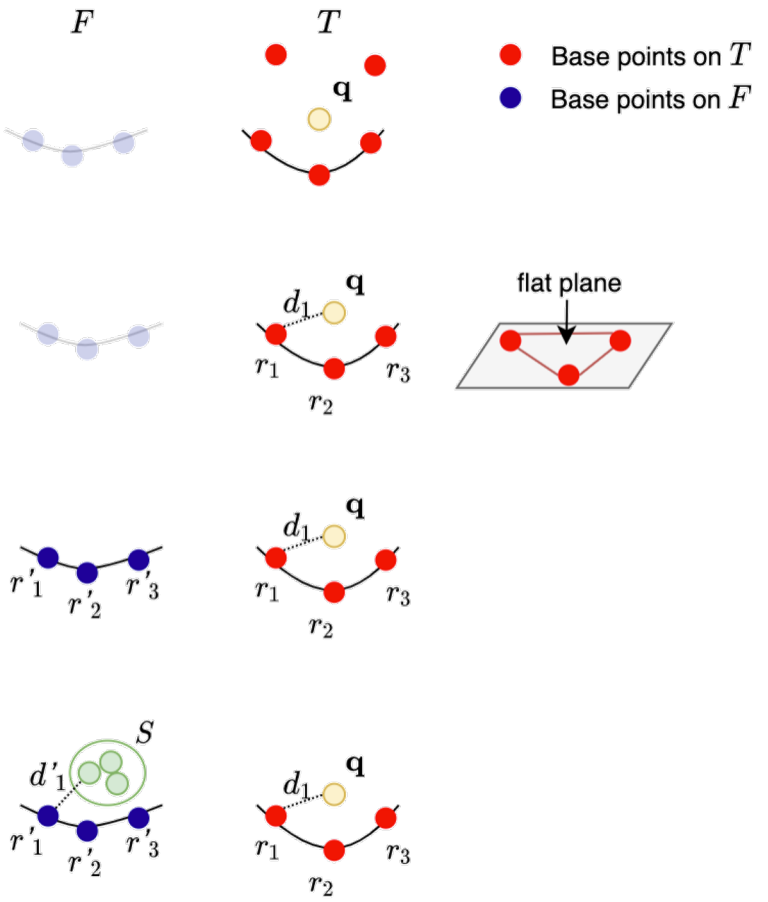
1. For a q on T, we find three nearest base points $\mathrm{r}_1, \mathrm{r}_2, \mathrm{r}_3$ that are not collinear. The distance between q_j and $\mathrm{r}_1, \mathrm{r}_2, \mathrm{r}_3$ are d_1, d_2, d_3
2. The corresponding base points of $\mathrm{r}_1, \mathrm{r}_2, \mathrm{r}_3$ on F would be $\mathrm{r}'_1, \mathrm{r}'_2, \mathrm{r}'_3$



Algorithm Design

- How to search correspondence?
  - Most of points have unknown correspondence, but base points have known correspondence!

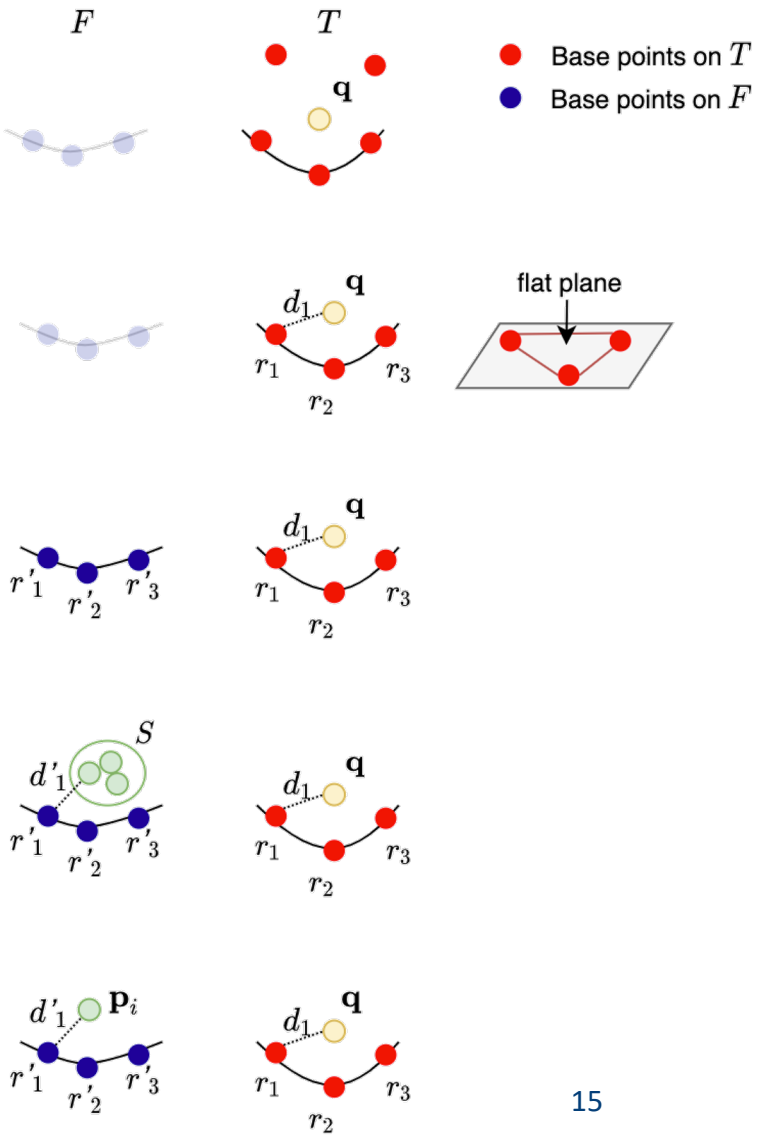
1. For a q on T, we find three nearest base points $\mathrm{r}_1, \mathrm{r}_2, \mathrm{r}_3$ that are not collinear. The distance between q_j and $\mathrm{r}_1, \mathrm{r}_2, \mathrm{r}_3$ are d_1, d_2, d_3
2. The corresponding base points of $\mathrm{r}_1, \mathrm{r}_2, \mathrm{r}_3$ on F would be $\mathrm{r}'_1, \mathrm{r}'_2, \mathrm{r}'_3$
3. Find the vertices that are near enough $S = \{\mathrm{p}_i \in F: \|\mathrm{p}_i - \mathrm{q}\| \leq D\}$



Algorithm Design

- How to search correspondence?
  - Most of points have unknown correspondence, but base points have known correspondence!

1. For a q on T, we find three nearest base points $\mathrm{r}_1, \mathrm{r}_2, \mathrm{r}_3$ that are not collinear. The distance between q_j and $\mathrm{r}_1, \mathrm{r}_2, \mathrm{r}_3$ are d_1, d_2, d_3
2. The corresponding base points of $\mathrm{r}_1, \mathrm{r}_2, \mathrm{r}_3$ on F would be $\mathrm{r}'_1, \mathrm{r}'_2, \mathrm{r}'_3$
3. Find the vertices that are near enough $S = \{\mathrm{p}_i \in F: \|\mathrm{p}_i - \mathrm{q}\| \leq D\}$
4. For each $\mathrm{p}_i \in S$ do
5. Calculate distance to $\mathrm{r}'_1, \mathrm{r}'_2, \mathrm{r}'_3$ as d'_1, d'_2, d'_3
6. Calculate $E'_i = \left|\frac{d_1}{d_2} - \frac{d'_1}{d'_2}\right| + \left|\frac{d_2}{d_3} - \frac{d'_2}{d'_3}\right| + \left|\frac{d_3}{d_1} - \frac{d'_3}{d'_1}\right|$
7. EndFor
8. Find one $\mathrm{p}_i \in S$ that have minimum E'_i

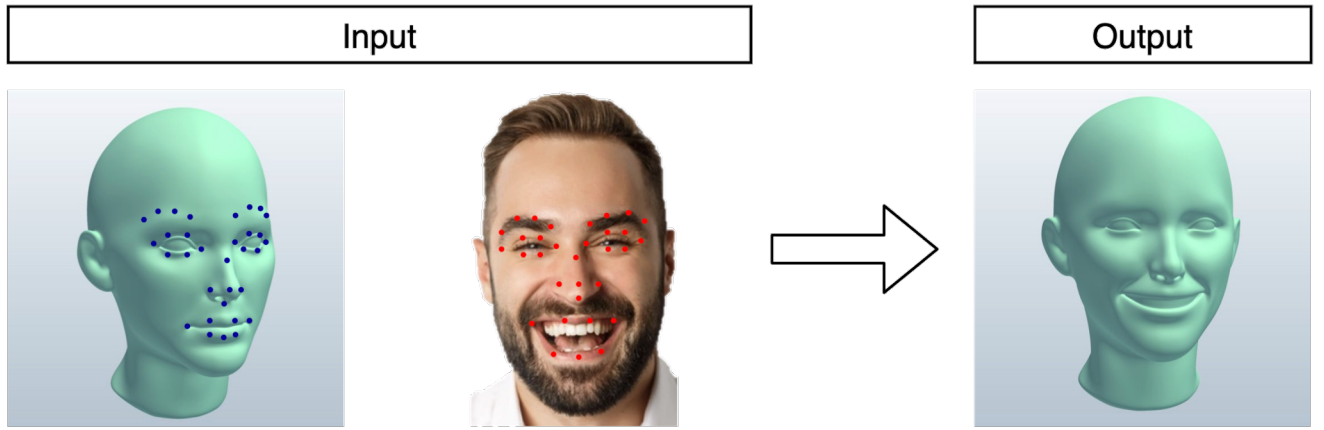


Algorithm Design

- A High-Level Algorithm

1. Initialization
2. Repeat
3. Based on points on Target, get the corresponding points on Reference. Generate the corresponding points set.
4. Perform shape change based on corresponding points set.
5. Until convergence.

- **Two key components**
  - Correspondence
  - Shape change: non-rigid transformation



Algorithm Design

- Overall Algorithm

1. Initialization: $R = F$, correspondence set $S = \emptyset$.
2. For $k = 1, \ldots, K$ do
3. If $k = K$ then
4. Relax correspondence search criteria.
5. For each mesh vertex q_j on T do
6. To search its corresponding point $c(\mathrm{q}_j)$ on R. → Correspondence Search
7. If $c(\mathrm{q}_j) \neq \emptyset$ and does not cause crossing then → No surface self-intersection
8. Compute location $\mathrm{u}_j = \mathrm{q}_j + (k/K)(c(\mathrm{q}_j) - \mathrm{q}_j)$.
9. Add $(\mathrm{u}_j, \mathrm{q}_j)$ into S.
10. Apply Laplacian Deformation on R given S. → Perform shape change

Algorithm Design

- How to avoid surface self-intersection?
  - This is caused by inappropriate correspondence
  - We need to check whether the points with its corresponding points violate the property of $\cos\theta(\mathrm{q}) + \cos\theta(\mathrm{p}) < \frac{\|\mathrm{q}-\mathrm{p}\|}{D}$

- How to ensure many correspondence?
  - Relax the search criteria
  - Change shape slowly and incrementally

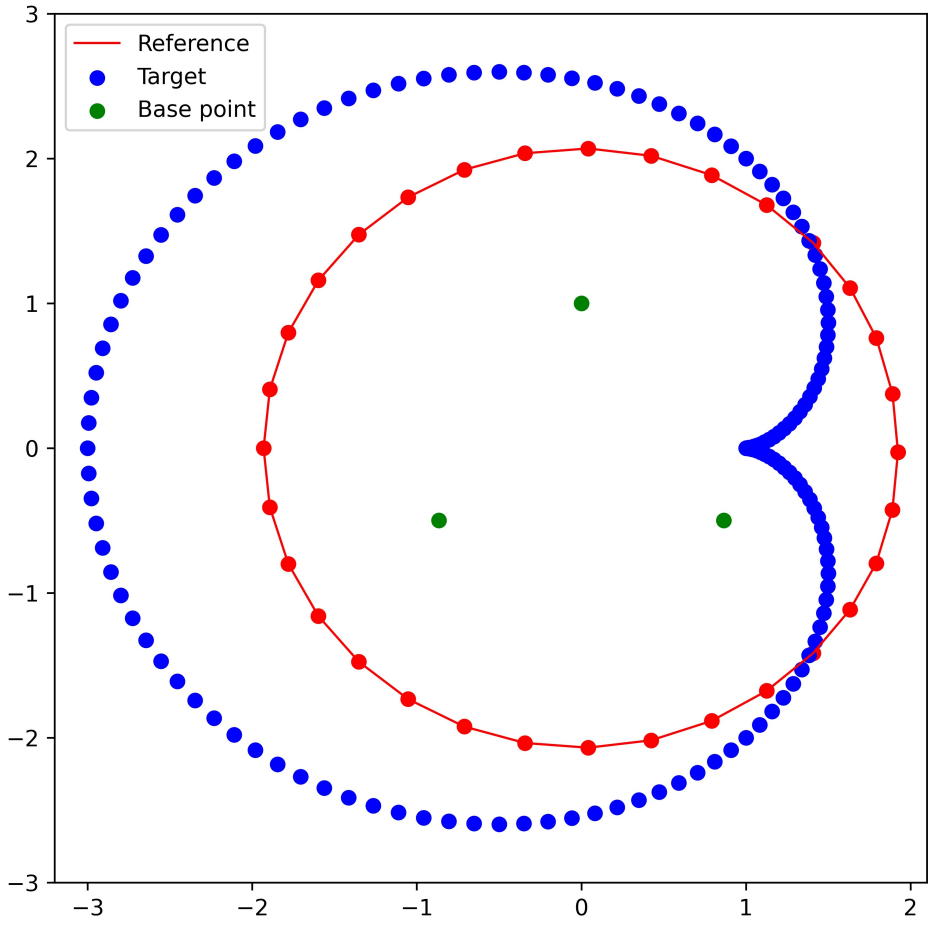
Justification

- Convergence
  - Initialization: $R = F$ is reasonable.
  - Correspondence: correspondence search algorithm will return the nearest (and most similar) points on R given a point on T.
  - Shape Change: Laplacian Deformation will move selected points on R towards their desired positions.

Justification

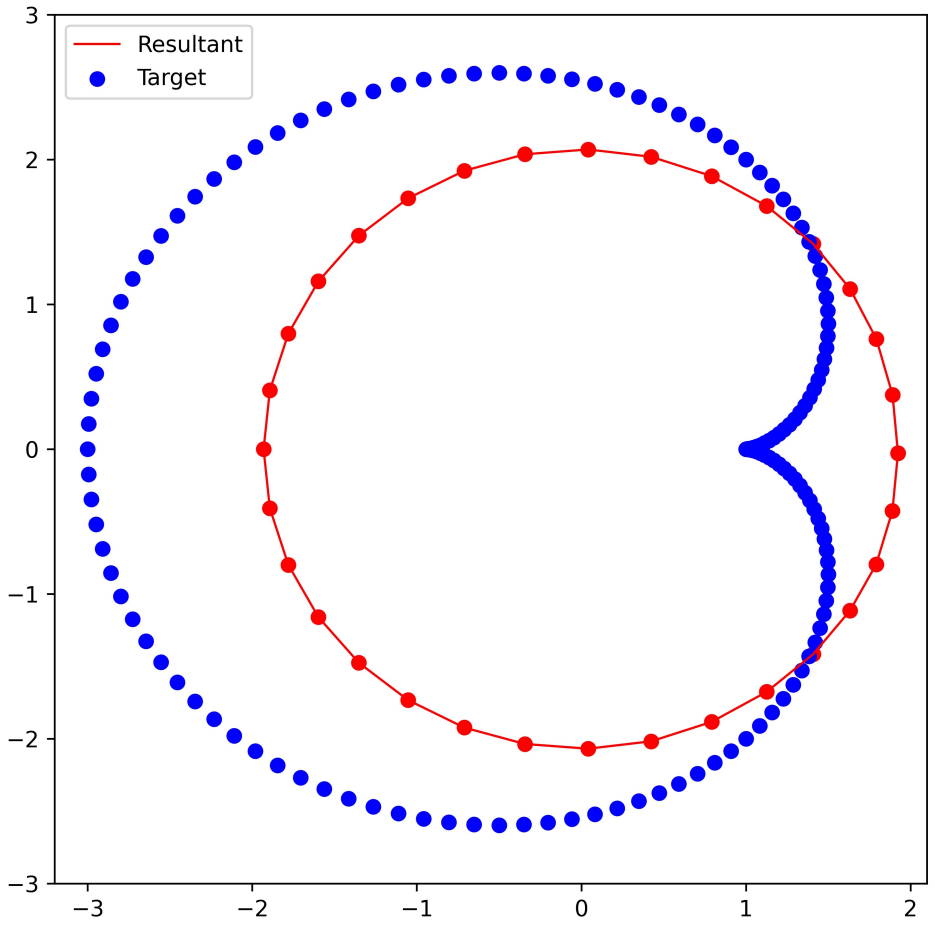
- Convergence
  - Initialization: $R = F$ is reasonable.
  - Correspondence: correspondence search algorithm will return the nearest (and most similar) points on R given a point on T.
  - Shape Change: Laplacian Deformation will move selected points on R towards their desired positions.
- Correctness
  - The optimization objective and the similar shape constraint are guaranteed by Laplacian Deformation.
  - Good correspondence is guaranteed by correspondence search algorithm.
  - In the case of bad correspondence (by changing shape fast), computation of u can make the shape change slow and incremental. This can also ensure that points on R and F are near enough.

Example: fit a circle to a heart line



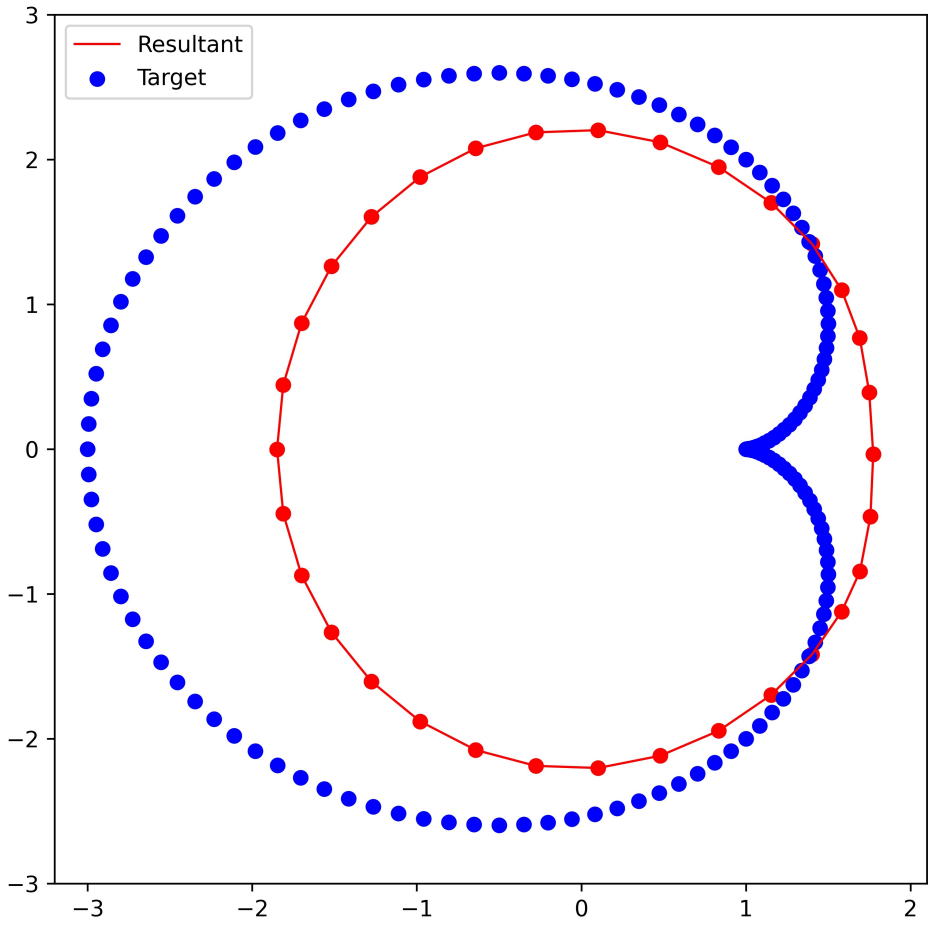
- Resultant model: initialized as F, which is a circle, to simulate a neutral expression.
- Target: a series of scatters representing desired exp-pression on a depth image.
- Base points: 3 vertices of an equilateral triangle within each graph.

Example: 1st iteration



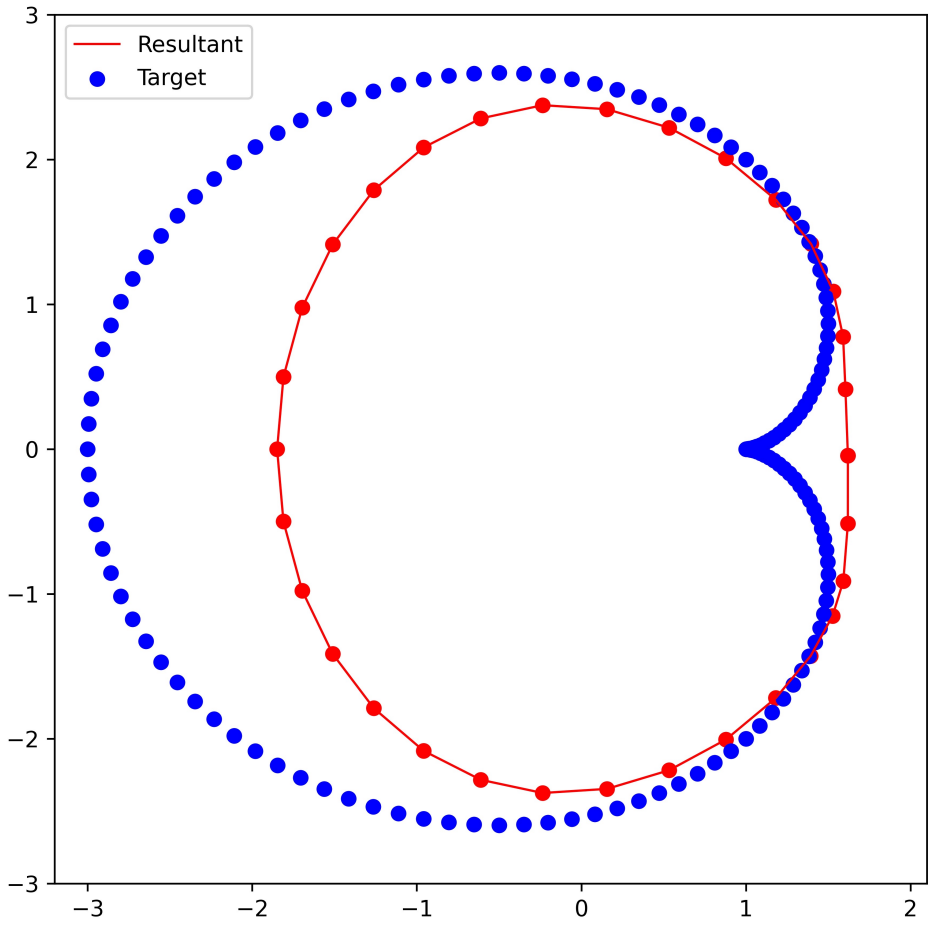

```
1th iteration
      Point 0          cannot find correspondance
      Point 1          cannot find correspondance
      Point 8          cannot find correspondance
      Point 9          cannot find correspondance
      Point 10         cannot find correspondance
      Point 11         cannot find correspondance
      Point 12         cannot find correspondance
      Point 13         cannot find correspondance
      Point 14         cannot find correspondance
      Point 15         cannot find correspondance
      Point 16         cannot find correspondance
      Point 17         cannot find correspondance
      Point 18         cannot find correspondance
      Point 19         cannot find correspondance
      Point 20         cannot find correspondance
      Point 21         cannot find correspondance
      Point 22         cannot find correspondance
      Point 23         cannot find correspondance
      Point 24         cannot find correspondance
      Point 31         cannot find correspondance
```

Example: 2nd iteration



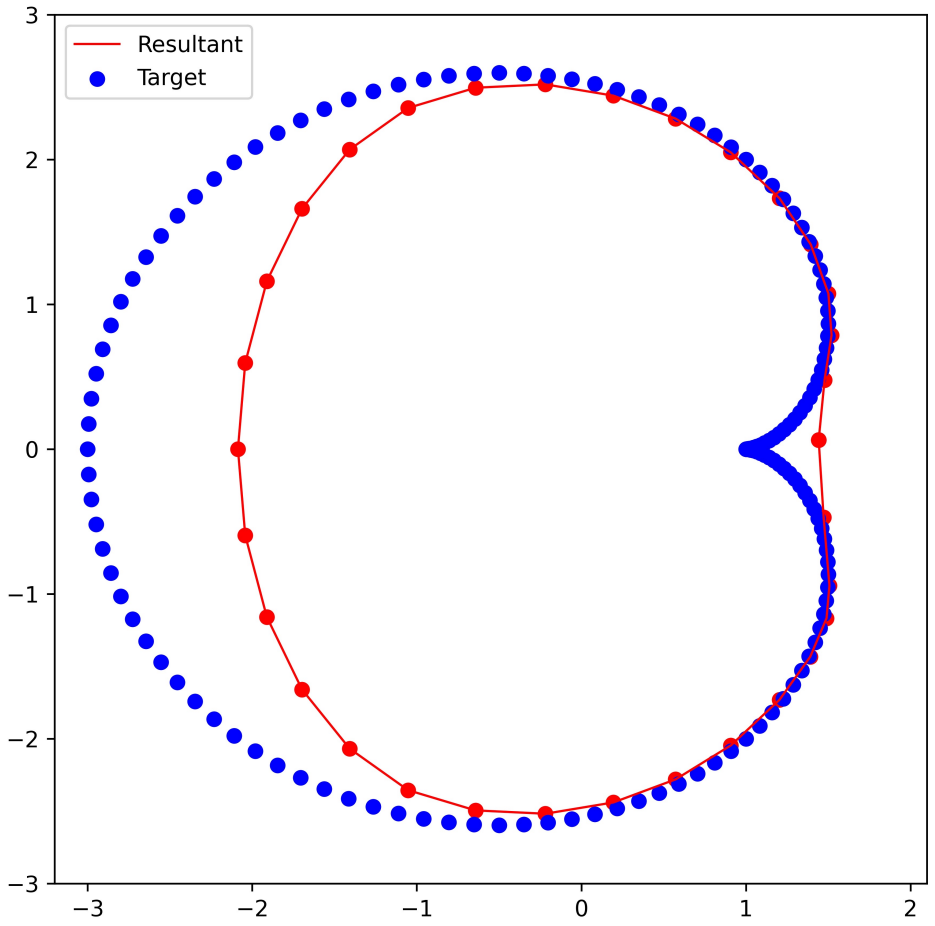

```
2th iteration
      Point 0          cannot find correspondance
      Point 9          cannot find correspondance
      Point 10         cannot find correspondance
      Point 11         cannot find correspondance
      Point 12         cannot find correspondance
      Point 13         cannot find correspondance
      Point 14         cannot find correspondance
      Point 15         cannot find correspondance
      Point 16         cannot find correspondance
      Point 17         cannot find correspondance
      Point 18         cannot find correspondance
      Point 19         cannot find correspondance
      Point 20         cannot find correspondance
      Point 21         cannot find correspondance
      Point 22         cannot find correspondance
      Point 23         cannot find correspondance
      Point 31         cannot find correspondance
```

Example: 3rd iteration



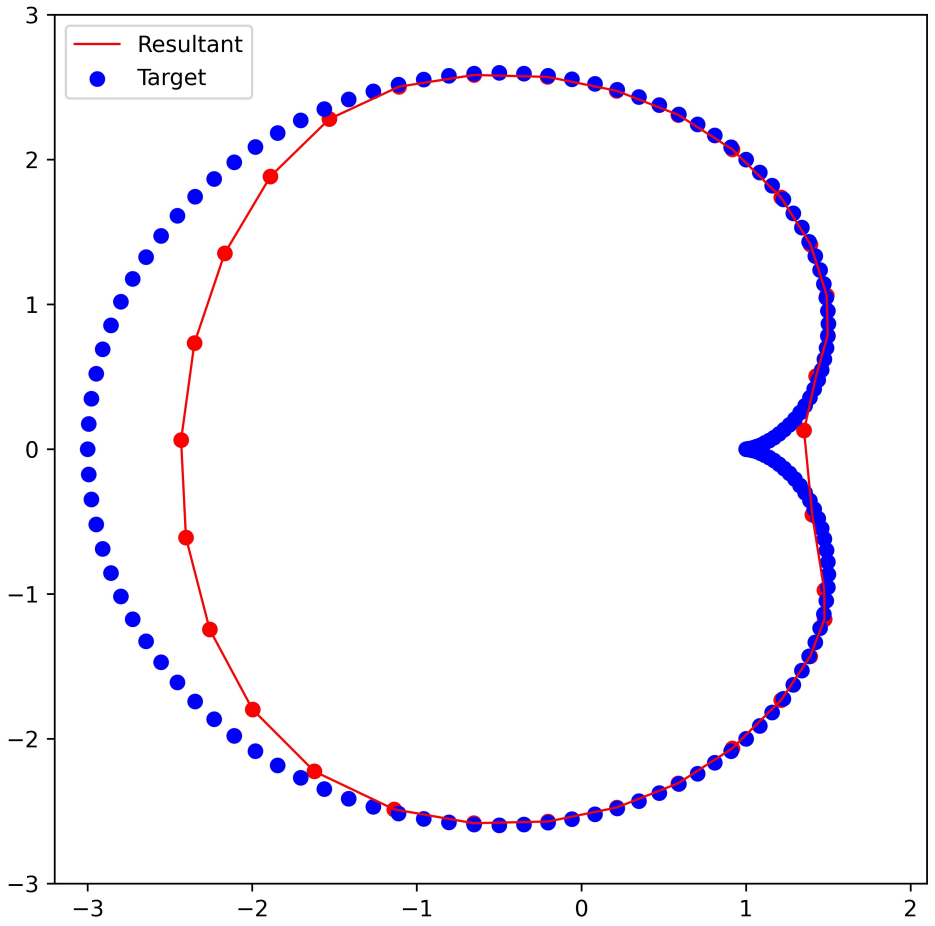

```
3th iteration
      Point 0          cannot find correspondance
      Point 10         cannot find correspondance
      Point 11         cannot find correspondance
      Point 12         cannot find correspondance
      Point 13         cannot find correspondance
      Point 14         cannot find correspondance
      Point 15         cannot find correspondance
      Point 16         cannot find correspondance
      Point 17         cannot find correspondance
      Point 18         cannot find correspondance
      Point 19         cannot find correspondance
      Point 20         cannot find correspondance
      Point 21         cannot find correspondance
      Point 22         cannot find correspondance
      Point 31         cannot find correspondance
```

Example: 4th iteration



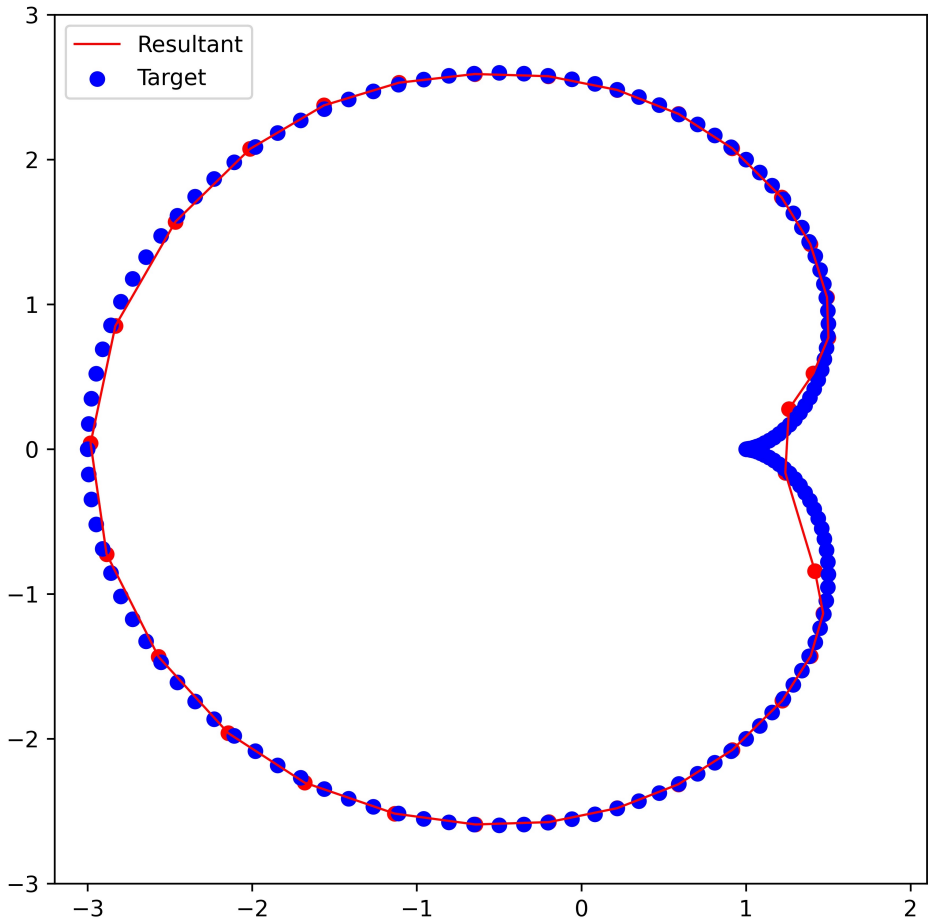

```
4th iteration
      Point 12          cannot find correspondance
      Point 13          cannot find correspondance
      Point 14          cannot find correspondance
      Point 15          cannot find correspondance
      Point 16          cannot find correspondance
      Point 17          cannot find correspondance
      Point 18          cannot find correspondance
      Point 19          cannot find correspondance
      Point 20          cannot find correspondance
      Point 31          cannot find correspondance
```

Example: 5th iteration




```
5th iteration
      Point 13          cannot find correspondance
      Point 14          cannot find correspondance
      Point 15          cannot find correspondance
      Point 16          cannot find correspondance
      Point 17          cannot find correspondance
      Point 18          cannot find correspondance
      Point 19          cannot find correspondance
      Point 31          cannot find correspondance
```

Example: 6th iteration



6th iteration

- Along our matching process, each vertex will be moved to its corresponding location gradually.
- The Good and Many correspondence constraint can be satisfies in the process.

THANK YOU