Face model registration and facial animation retargeting

Jung-Ju Choi
Ajou University

Introduction

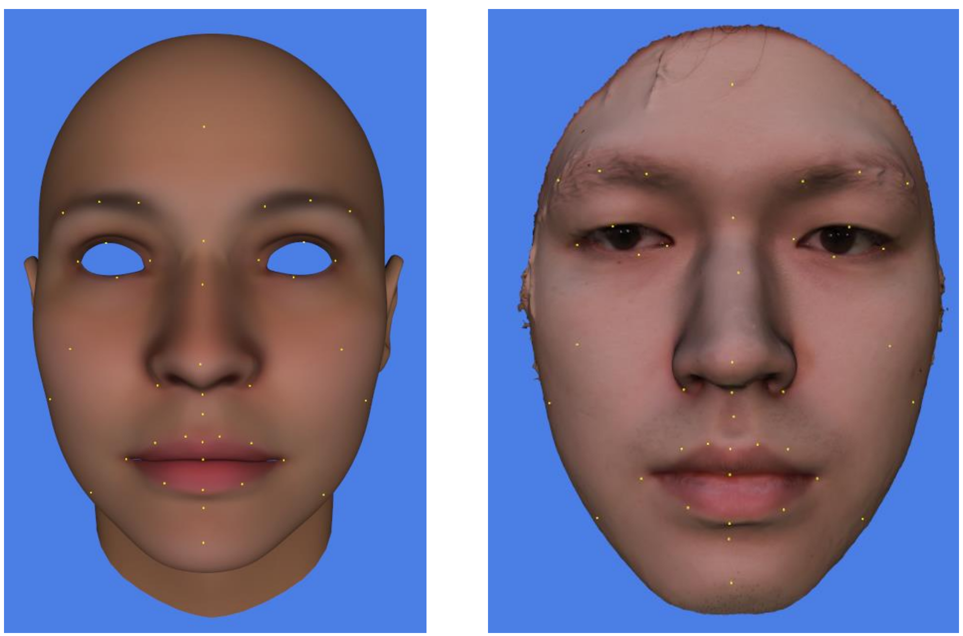
- For a given source face
  - Well-defined reference geometry and animation
- And for a given target face
  - Unorganized geometry and texture
  - Scanned data
- Generate reasonable target face
  - Deform the source model to the target
  - Transfer the source animation to the target

Our approach

- Registration
  - Find landmarks at the target model
  - Apply thin-plate spline interpolation between landmarks
  - Texture synthesis
- Animation
  - Training reference animation
  - Animate the thin-plate spline interpolated model

Registration

- Finding landmark correspondence
- Model fitting
  - Scattered data interpolation b.w.t. landmarks



Landmarks of source and target

Model fitting(TPS)

Landmark correspondence

- General Iterative Closest Point algorithm
  - For each landmark L at source face
    - While threshold
      - Grouping region for the landmark
        - » The region becomes smaller than the previous step
      - Find rigid transformation and apply transformation
    - Return the point in the target closest to L

Energy function to find rigid transformation b.w.t. two point clouds.

$$e(\theta, \mathbf{t}) = \sum_{i=1}^{N} \left\| rot(\theta)\mathbf{p}_i + \mathbf{t} - \mathbf{q}_i \right\|^2$$

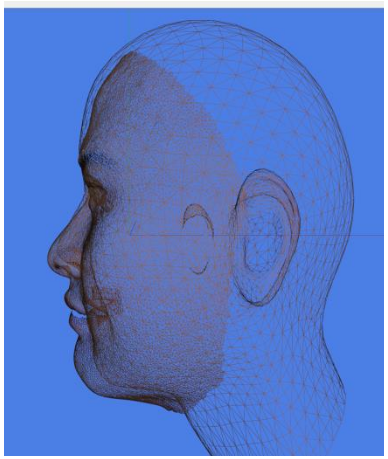
$\mathbf{p}_i$: points of source model
$\mathbf{q}_i$: points of target model

PCL(Point Cloud Library) - ICP algorithm

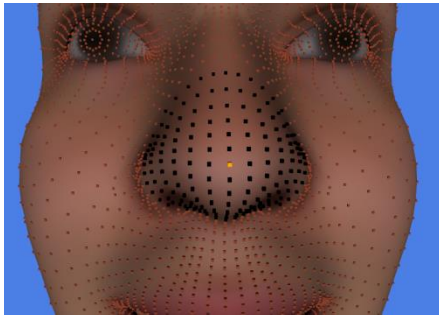
- Input
  - source point cloud, initial guess
  - target point cloud
- Iterate this process
  - Gathering correspondences(vertex distance)
    - $\mathbf{p}_i$ and $\mathbf{q}_i$ are assigned
  - Get rigid transform from correspondences
    - Minimize the energy function
  - Transform source cloud
  - Convergence check
    - # of iterations, transformation difference, fitness score

So, we do …

- Define source model landmarks
- Initial guess of rigid transformation
- For each landmark at source
  - – Grouping region
  - – Use PCL-ICP algorithm
    - Source cloud : grouped region
    - Target cloud : all vertices



Initial guess



Grouping region for ICP

Thin-plate spline interpolation

Displacement vector,

$$\mathsf{d}_i = \mathsf{q}_i - \mathsf{p}_i$$

$\mathbf{p}_i$: landmark of source model
$\mathbf{q}_i$: landmark of target model

$$S = \begin{bmatrix} 0 & u_{1,2} & \cdots & u_{1,k} & 1 & \mathbf{p}_1^T \\ u_{2,1} & 0 & & u_{2,k} & 1 & \mathbf{p}_1^T \\ \cdots & & \cdots & & & \cdots \\ u_{K,1} & u_{K,2} & & & 1 & \mathbf{p}_K^T \\ 1 & 1 & & 1 & 0 & 0 \\ \mathbf{p}_1 & \mathbf{p}_2 & & \mathbf{p}_K & 0 & 0 \end{bmatrix}.$$

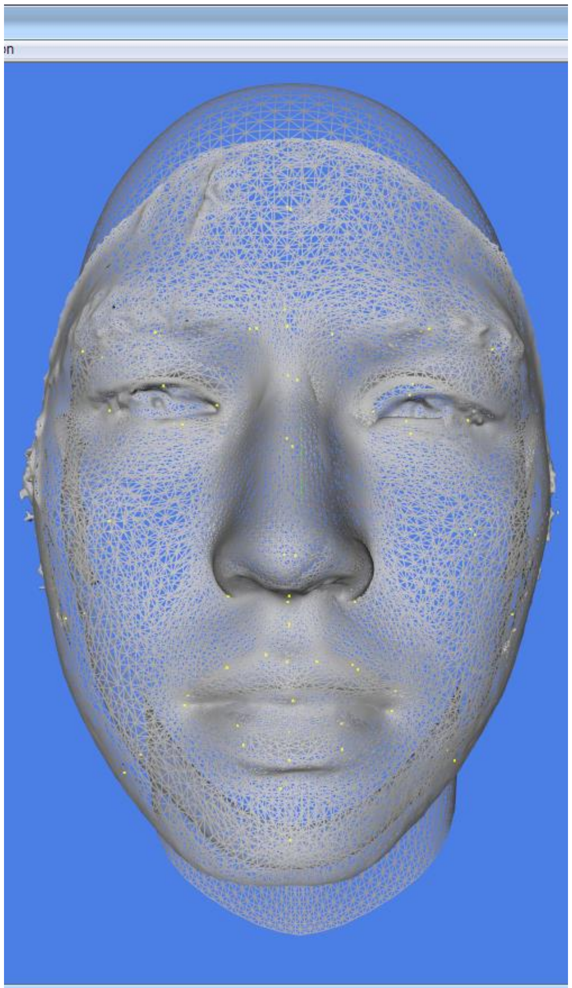
$$u_{i,j} = \|\mathsf{p}_i - \mathsf{p}_j\|.$$

$$W = S^{-1}\begin{bmatrix} d_1 & \dots & d_K & 0_{3\times3} \end{bmatrix}^T,$$

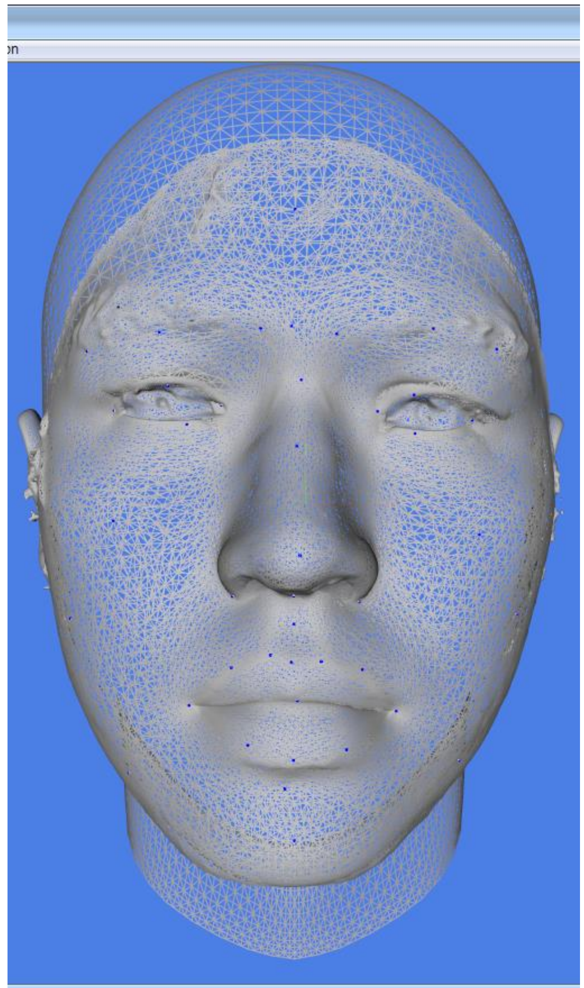
$$\mathbf{v}' = \mathbf{v} + \mathbf{W}^T\begin{bmatrix} u_1 & \dots & u_K & 1 & \mathbf{v}^T \end{bmatrix}^T. \qquad u_i = \|\mathbf{v} - \mathbf{p}_i\|.$$

$\mathbf{v}'$: new vertex position, $\mathbf{v}$: arbitrary point in the source mesh

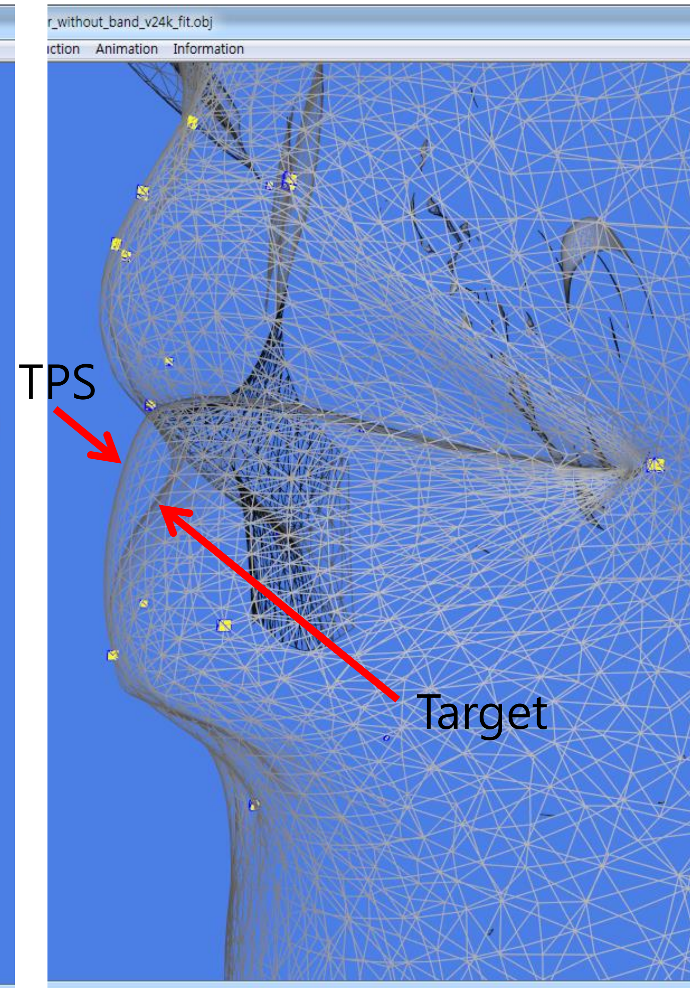
TPS result



Source and Target



TPS and Target

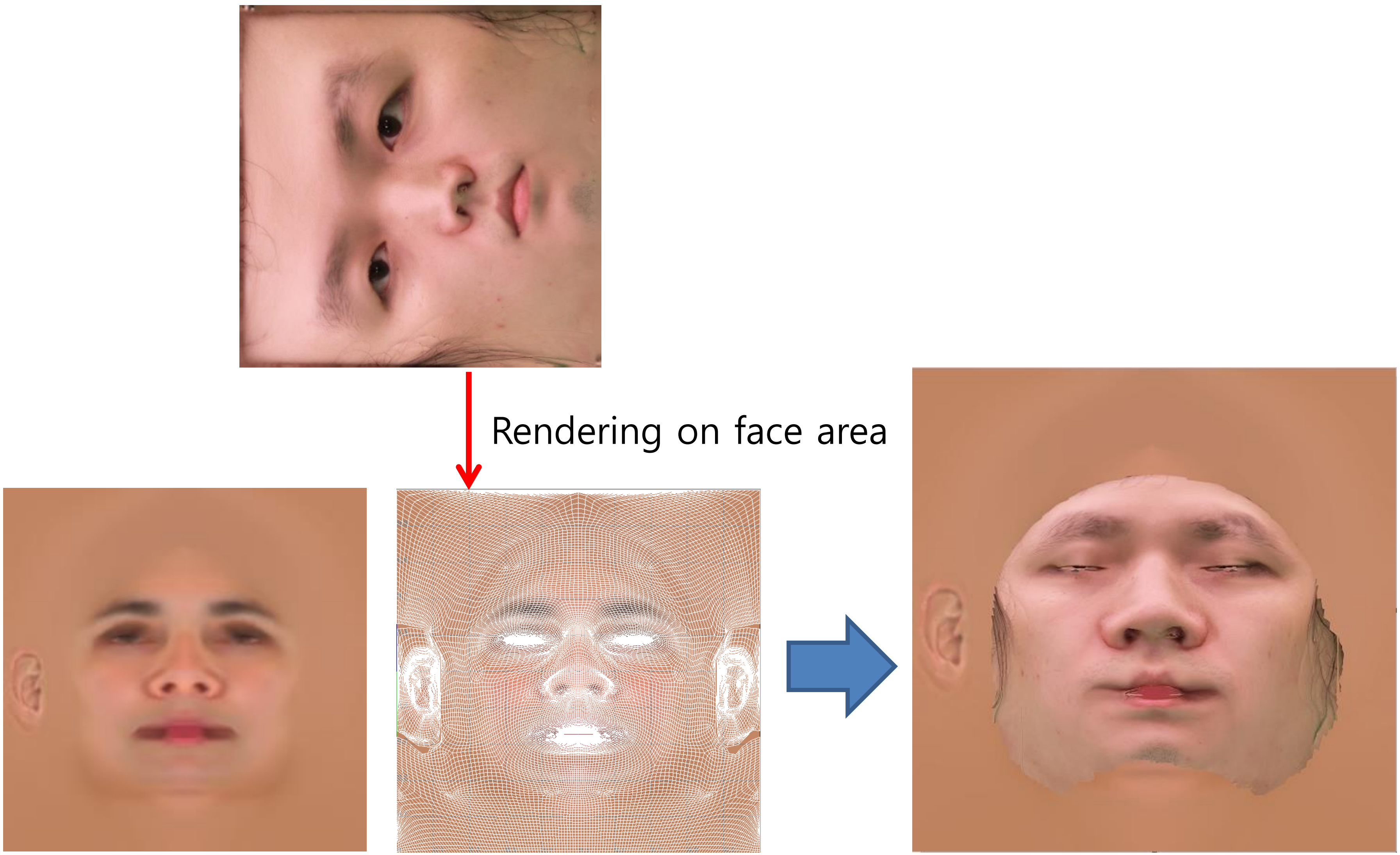


TPS and Target

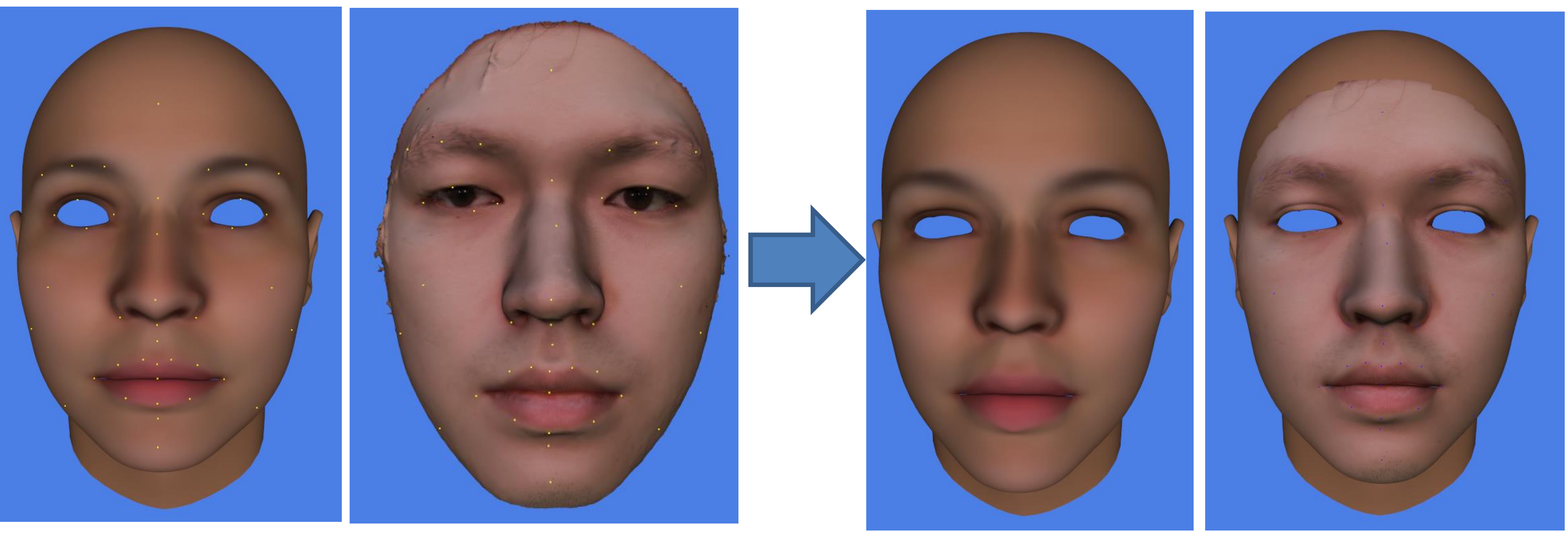
Texture synthesis

- Line-Plane intersection
  - Each vertex in TPS model
    - Position(vertex position)
    - Line direction(vertex normal)
  - Intersection test with target model
    - Find intersection triangles
      - Select a triangle with the minimum distance
      - Get barycentric coordinates on the triangle
      - Interpolate texture coordinates at the intersection
    - If not intersected, it is not on face area

Texture map synthesis



Texture transferred model



Landmarks of source and target

Model fitting

Texture transferred

Error of TPS model

- For vertices of face area

$$E = \frac{1}{n}\sum_{i=1}^{n}\left\| p_i - q_i \right\|.$$

p_i : vertex of TPS model in face area
q_i : projection of p_i on the nearest triangle of target model
n : number of vertices in face area

- Bounding size of models

	Width	Height	Depth
TPS model	6.378925	9.180843	6.630010
Target model	5.252716	5.831849	3.541810

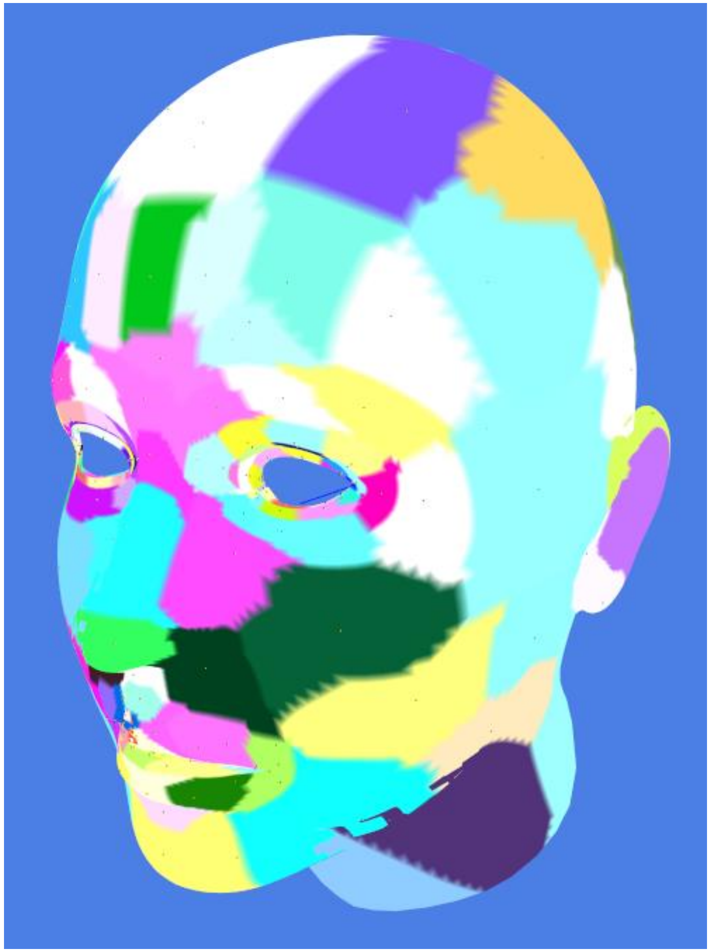
- Avg. error : 0.029703
- n: 11904

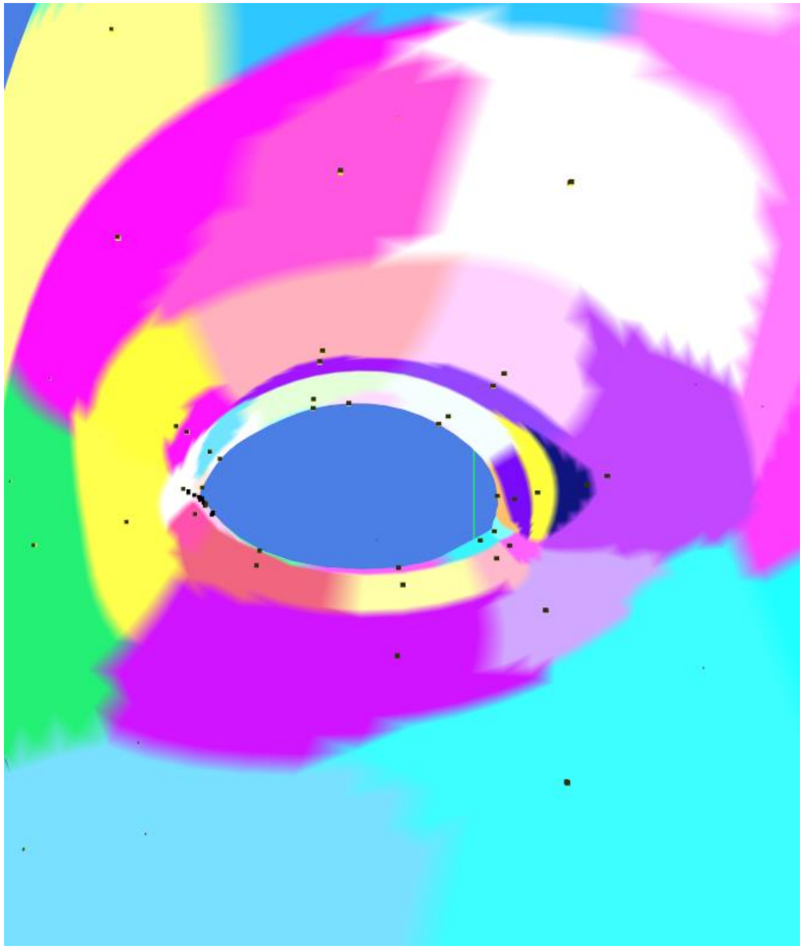
Facial animation

- Define regions on a face (manually)
- For each region, compute MVC of internal vertices
- Place some markers globally
  - Ex) one marker for each region
- Compute transformation between boundary and markers(needed animation data)
- Each frame,
  - Marker → Boundary → Internal vertices

    Transformation (Marker → Boundary), MVC (Boundary → Internal vertices)

Face regions

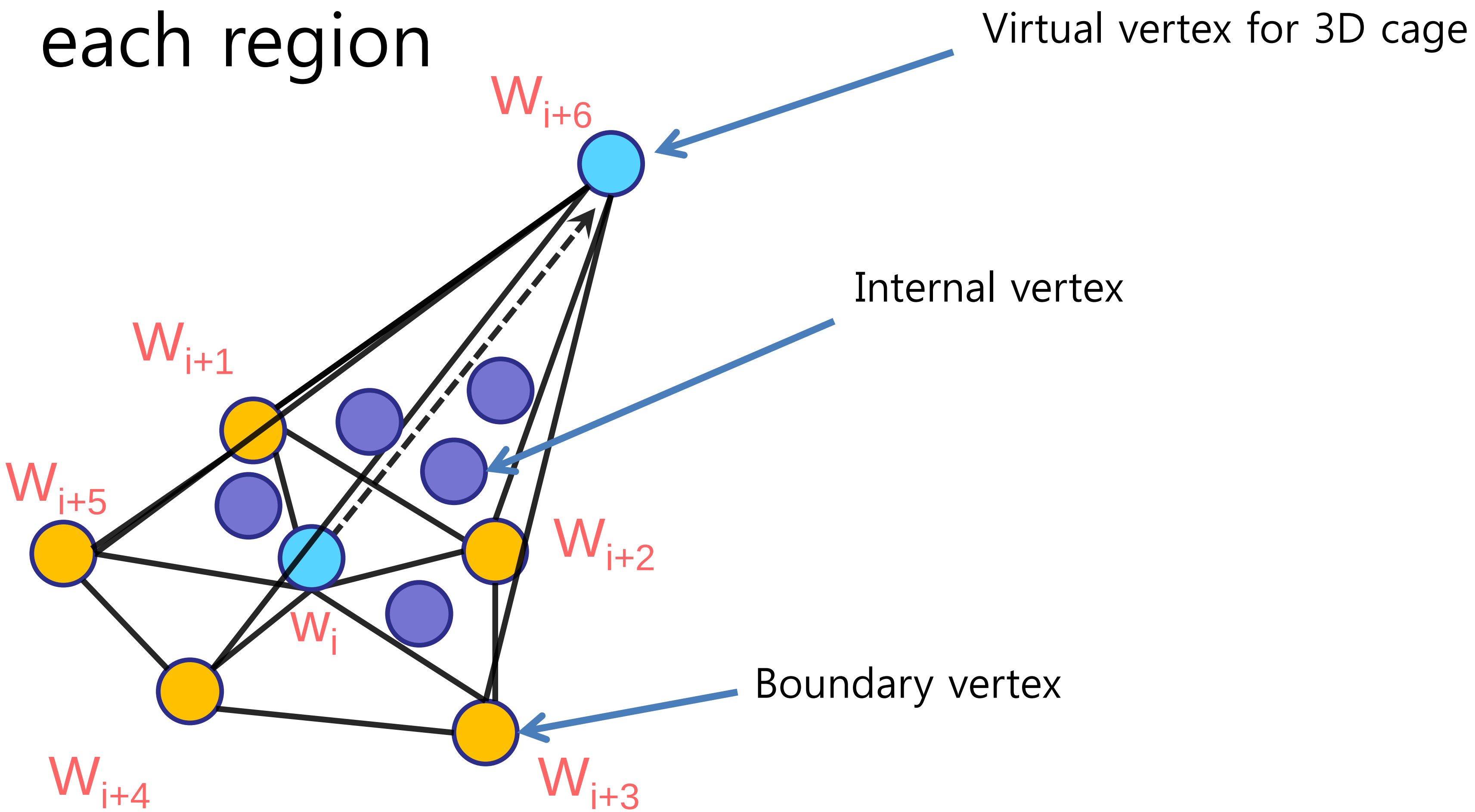




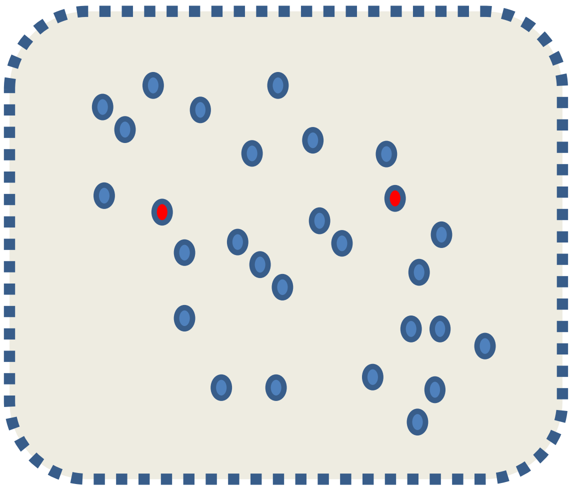
Regions for MVC reconstruction

MVC on a region

- Compute MVCs of internal vertices for each region

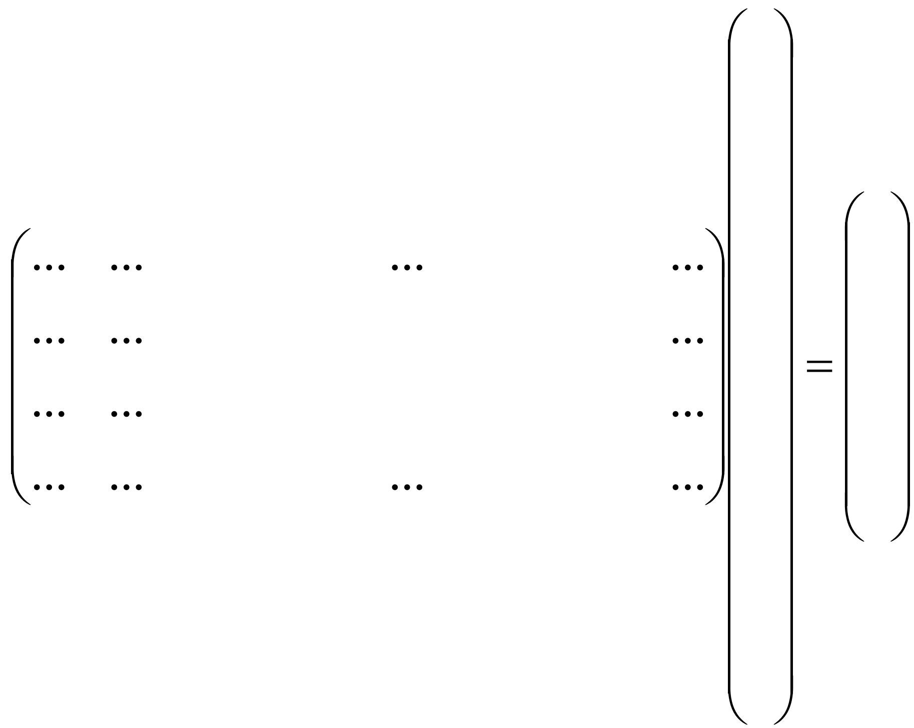


Boundary and marker positions



$$MV = U$$

Weight — Boundary — Marker



Marker training and computation of boundary from marker

- Example sequences

$$AV = U,$$

$$A = UV^{+}.$$

A : m x n unknown matrix

V : # of boundary vertices x # of sequences
n x s known matrix

U: # of markers x # of sequences
m x s known matrix

Compute A from examples

↓

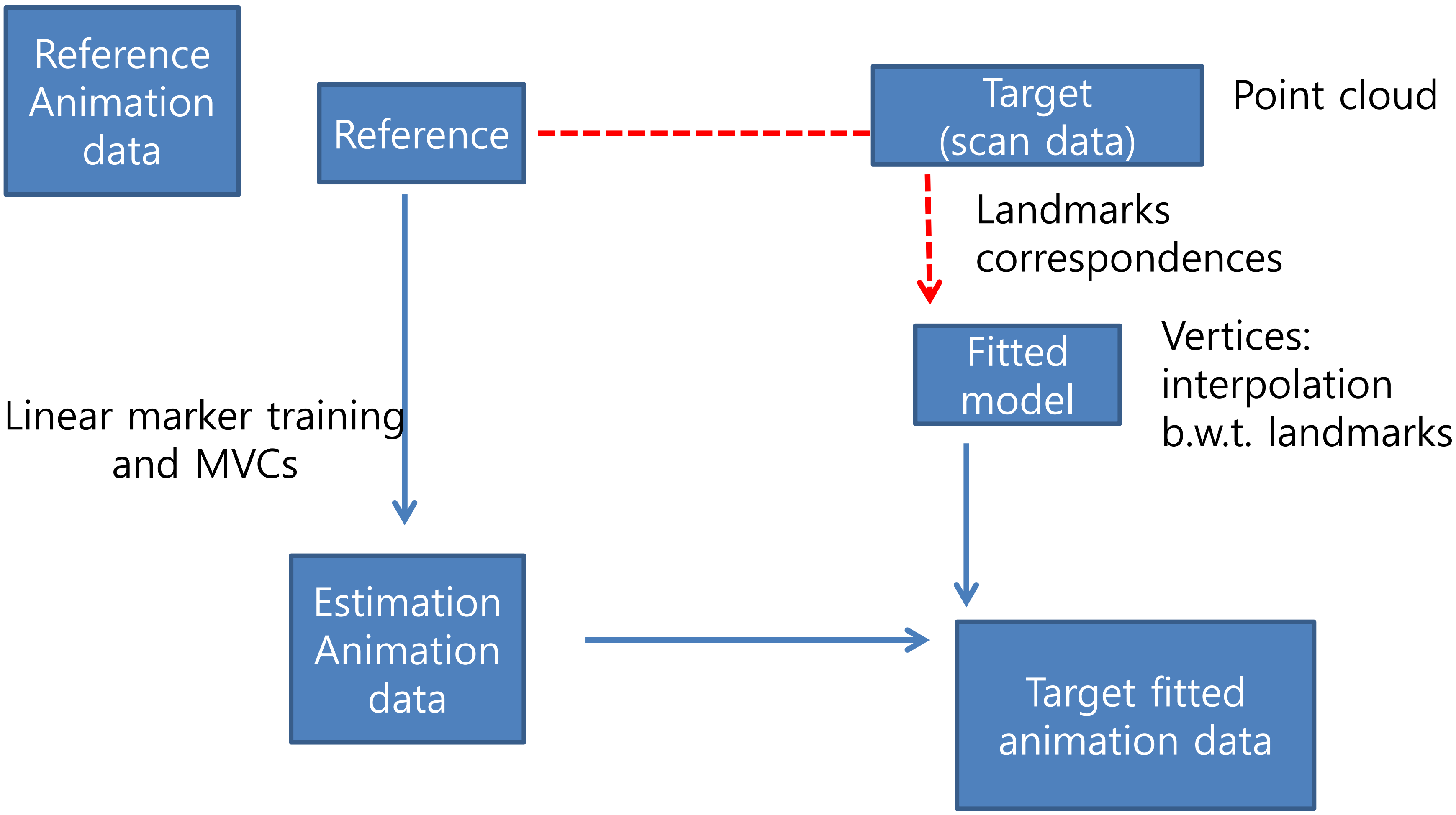
$$AV = U,$$

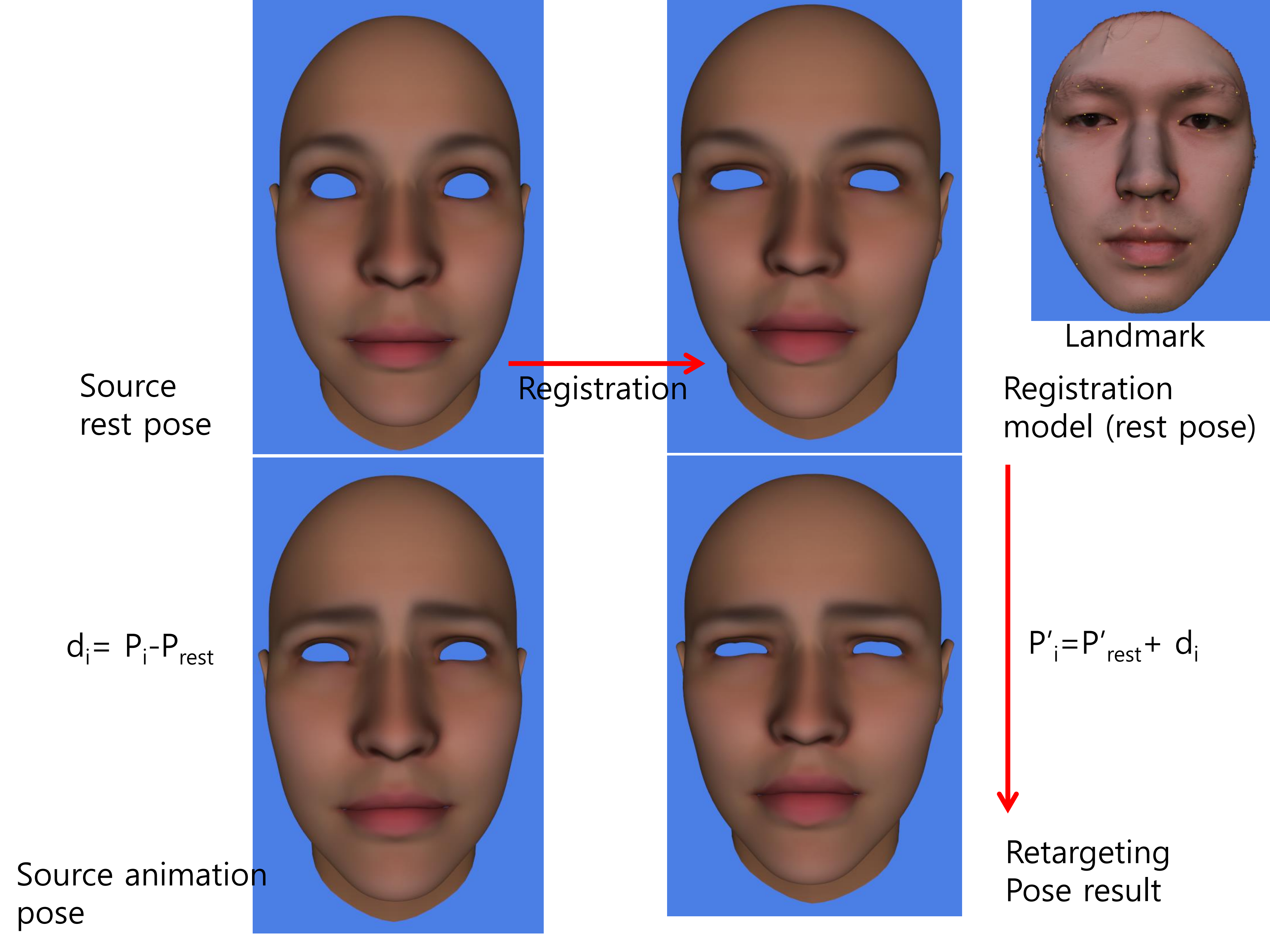
$$V' = A^{+}U.$$

We can get the pseudo inverse of A.

Then, boundary vertices can be computed from marker position.

Retargeting process




Landmark
Source rest pose
Registration
Registration model (rest pose)
$d_i = P_i - P_{rest}$
$P'_i = P'_{rest} + d_i$
Source animation pose
Retargeting Pose result

Result animation