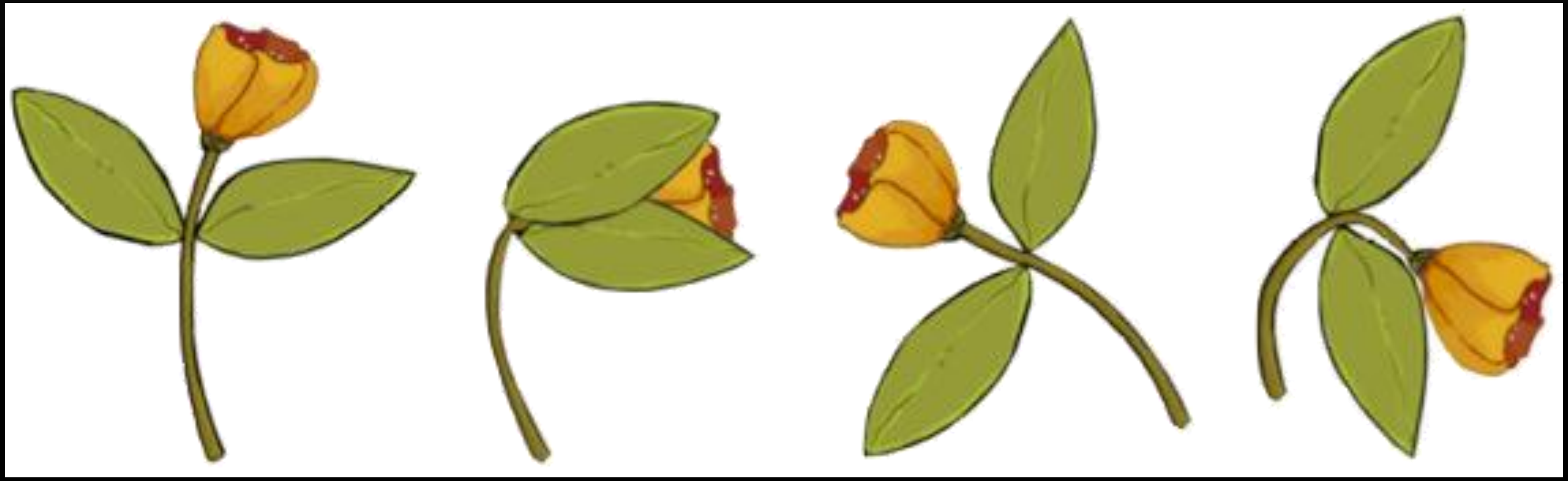
Planar Shape Deformation in X3D

Jung-Ju Choi
Ajou University

What to do for shape animation

- Objectives
  - Define an animation data for a 2D shape using X3D
  - By means of shape deformation
- Assumption
  - A 2D shape consists of vertices and triangles
  - No explicit underlying structure such as skeleton



How to define the shape animation

- Traditional vertex animation
  - Define the positions of vertices at selected frames, and
  - Interpolate the positions between two frames
    - MD2 file format by id Software in 1997
      - Consider only the animation of triangles
      - Vertex animation for approximately 10 fps
      - Limited to only predefined 20 animation sets
  - Advantages
    - Easy to understand and implement
    - Easy to define any animation
  - Disadvantages
    - Many selected frames due to the nonlinearity of shape animation
    - Data size due to the large amount of vertices

Suggested shape animation data

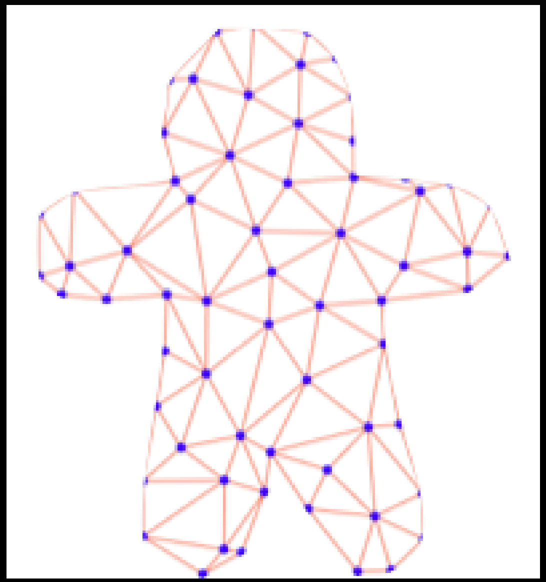
- Basic strategy: using the deformation
  - Split geometry and animation of the shape
  - Geometry represents the topology of the shape
    - Defined by a triangle mesh
  - Animation represents the shape change at all frames (or selected frames)
    - Defined by the motion of a small set of selected vertices
    - Compute the motion of other vertices at run-time
      - It requires computational cost at run-time

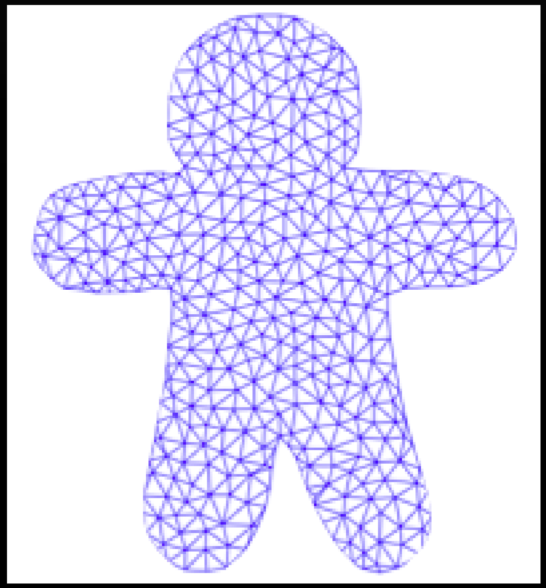
Suggested shape animation data

- Computing the positions of other vertices from those of selected vertices at each frame
  - Preserving some shape features of the rest pose as much as possible during the animation (as-rigid-as-possible shape animation)
    - Outline details
    - Angles of internal edges
    - Length of edges (area of triangles)
  - By nonlinear least squares optimization for
    - Laplacian coordinates of the boundary vertices
    - Mean value coordinates of the internal vertices
    - Edge length constraints of the edges
  - Slow, so requires quality control

Quality vs. Performance

- According to the number of selected vertices
  - The more vertices, the better quality
  - The less vertices, the more performance
    - Cubically proportional to the number of selected vertices
    - Pre-computation time is also affected by the number of selected vertices
    - Run time is affected by the number of other vertices





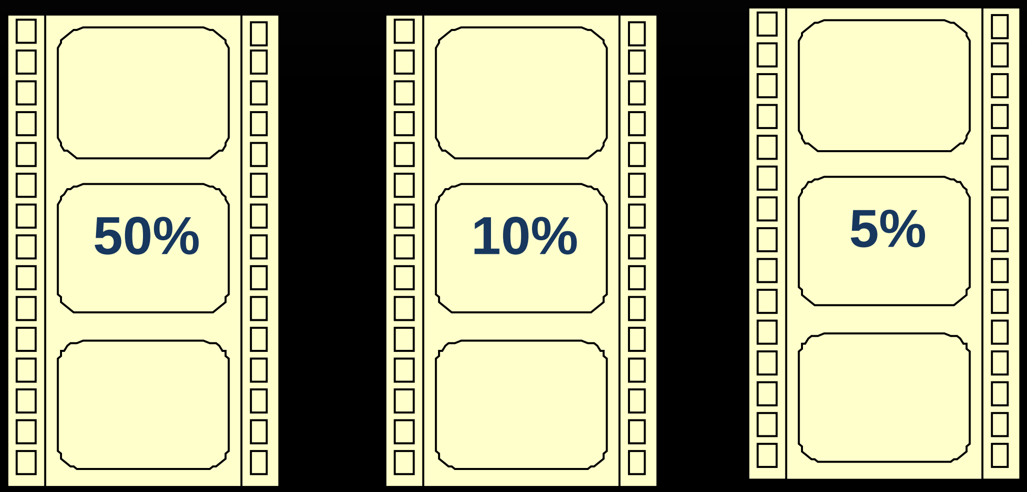
Quality vs. Performance

- When omitting pre-computation
  - Preserving the mean value coordinates shows the best performance
  - Preserving the edge length shows the best quality

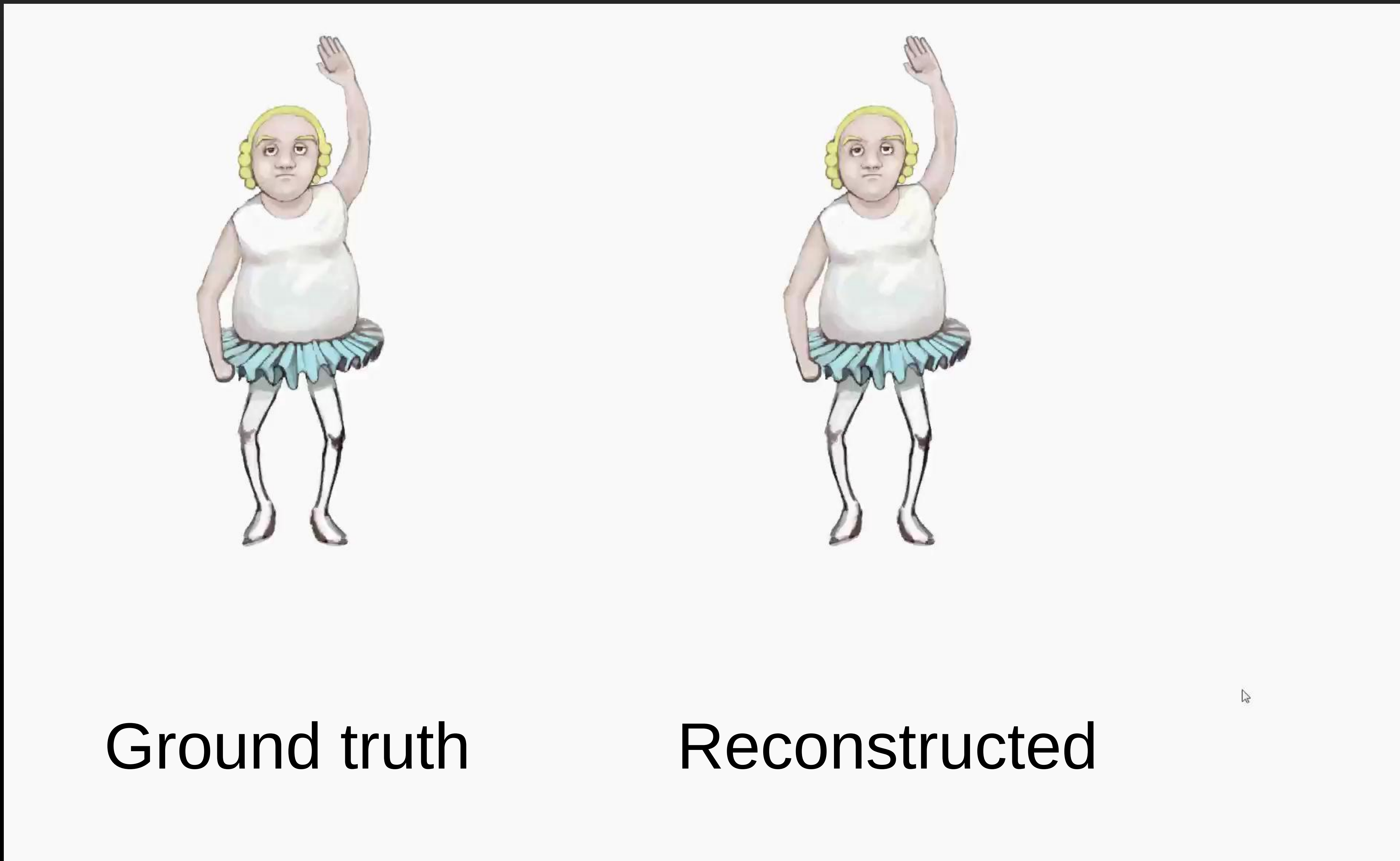
Quality vs. Performance

- Performance data for the dancer
  - Number of vertices: 355
  - Run-time: 3.162ms/frames for 10% samples

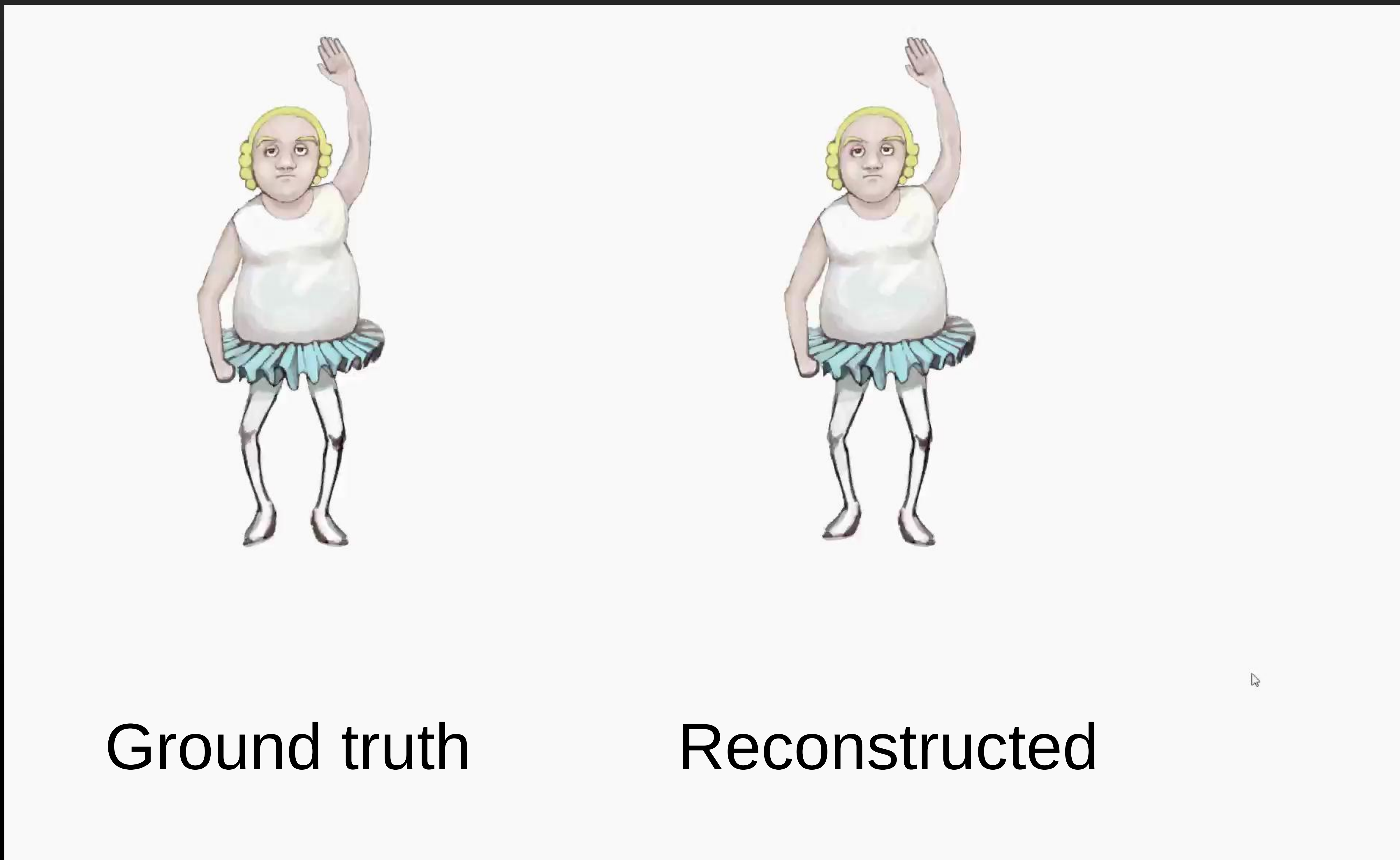
Dancer_JUMP	Sample(50%)	Sample(10%)	Sample(5%)
RMSE	0.72	4.17	4.90
Avg. difference	0.42	2.57	3.56
Max. difference	6.64	29.37	30.58



With 50% samples



With 10% samples



With 5% samples

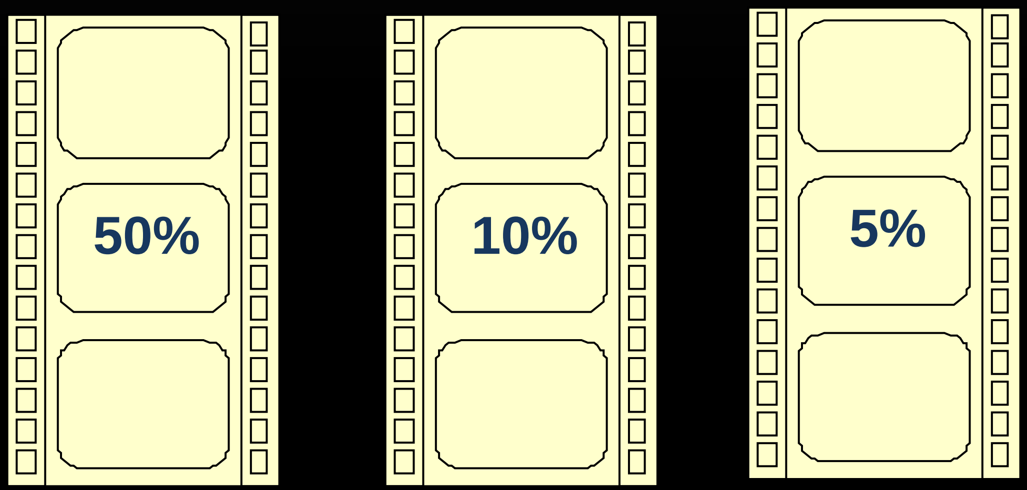
Ground truth

Reconstructed

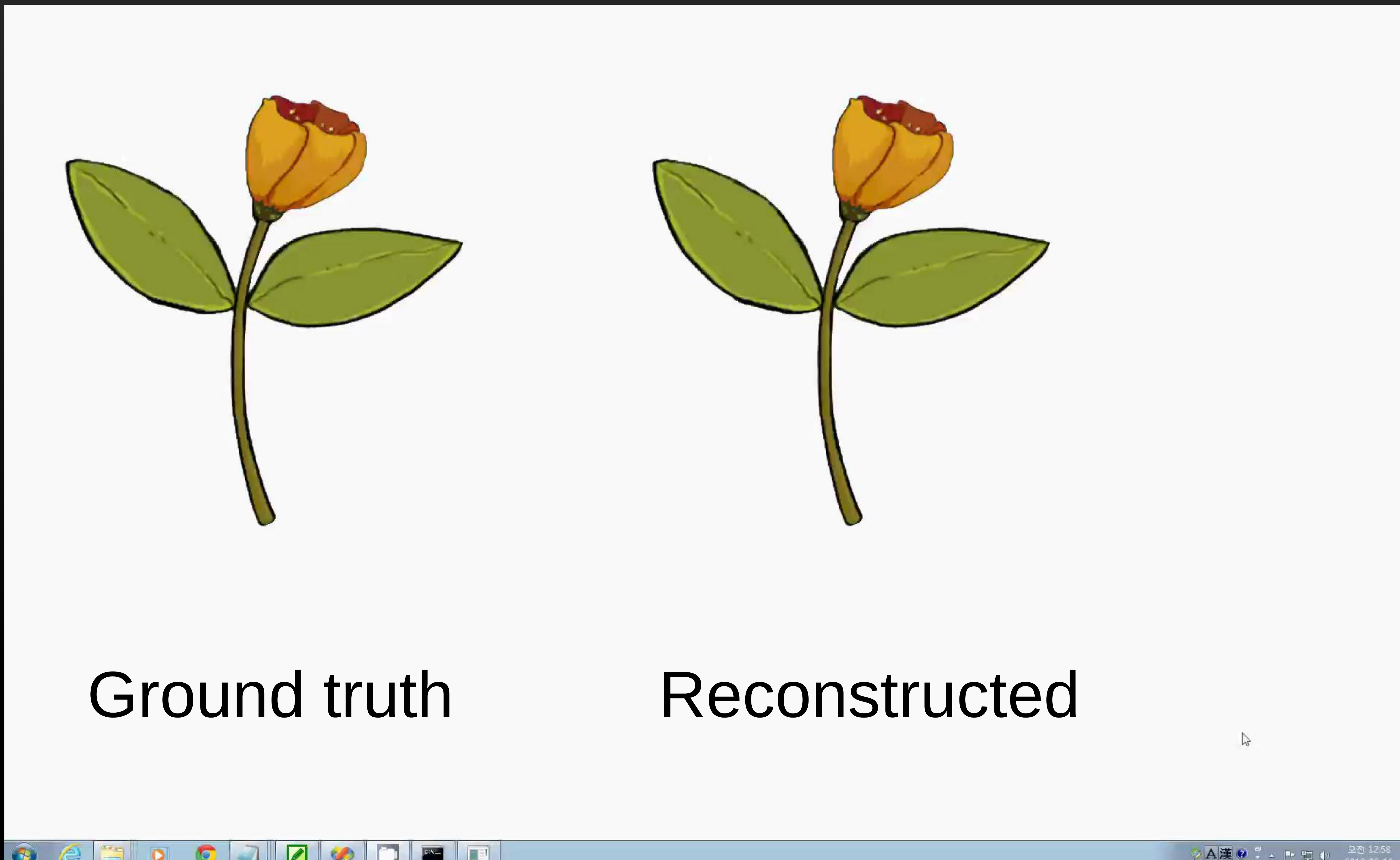
Quality vs. Performance

- Performance data for the flower
  - Number of vertices: 440
  - Run-time: 4.086ms/frames for 10% samples

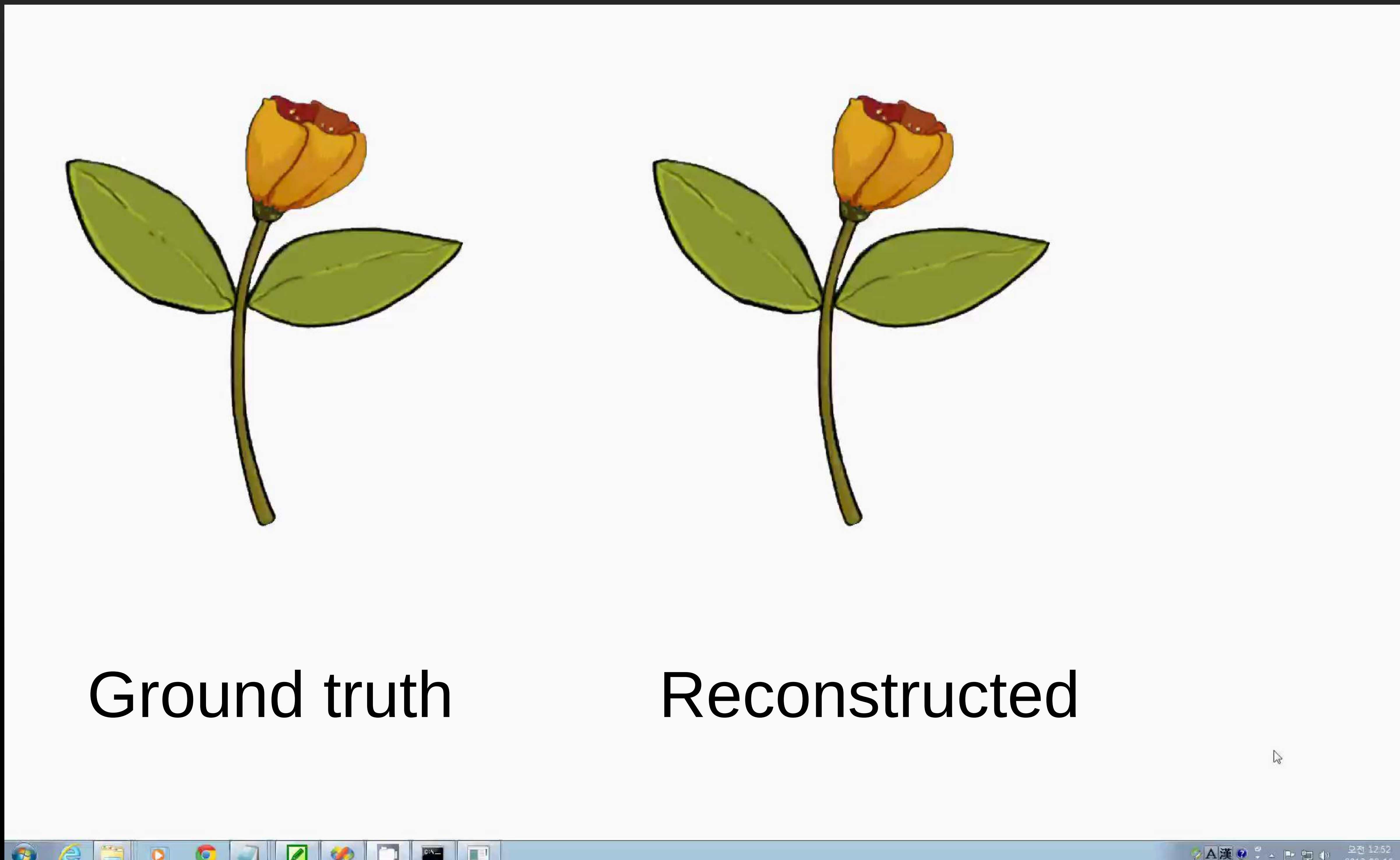
Flower_FLY	Sample(50%)	Sample(10%)	Sample(5%)
RMSE	0.32	1.34	2.67
Avg. difference	0.19	0.93	2.02
Max difference	3.36	9.54	11.09



With 50% samples



With 10% samples

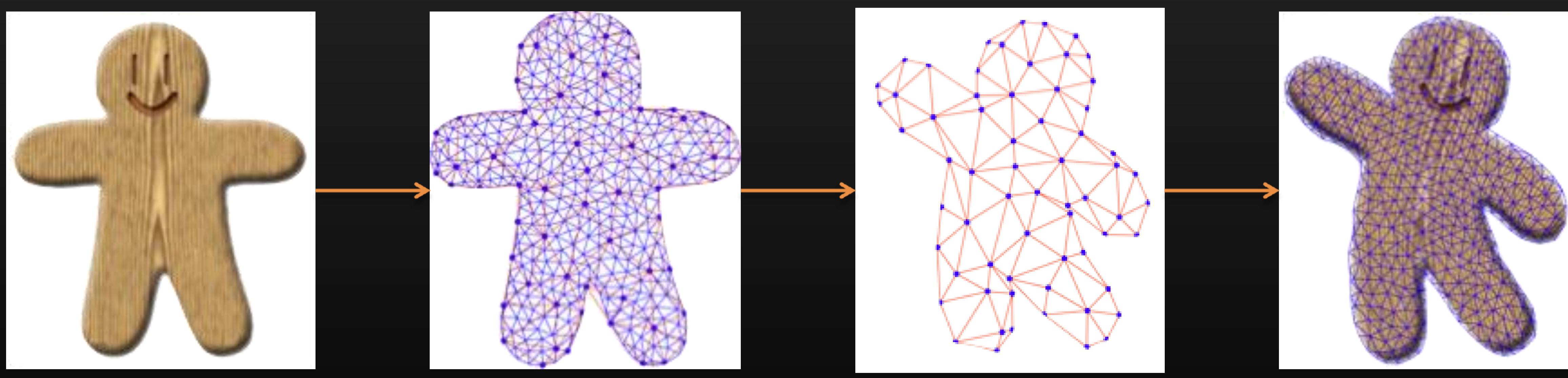


With 5% samples

Ground truth

Reconstructed

Shape animation data: summary



Shape | Geometry | Define the motion of selected vertices | Compute the motion of other vertices

- Shape Animation Data
  - Define the shape by a mesh
  - Define the motion of selected vertices at every frame
  - Compute the motion of other vertices at run time

Shape animation data using X3D

- Geometry data
  - Mesh information: to define shape
  - Quality information: to define selected vertices
- Animation data
  - Frame information: to define the motion of selected vertices
  - Computing information: to define the optimization method
- We can reduce the size of vertex animation by less than 1/10 with relatively small amount of errors in real time

Shape animation data using X3D

- Planar shape
  - Defines the geometry of a 2D shape
  - A set of triangles and their vertices

```
PlanarShape: X3DGeometryNode {
   SFNode    [in,out]   metadata    NULL   [X3DMetadataObject]
   MFVec2f   [in,out]   vertices    []     (-∞, ∞)
   MFInt32   [in,out]   faces       []     (0, ∞)
}
```

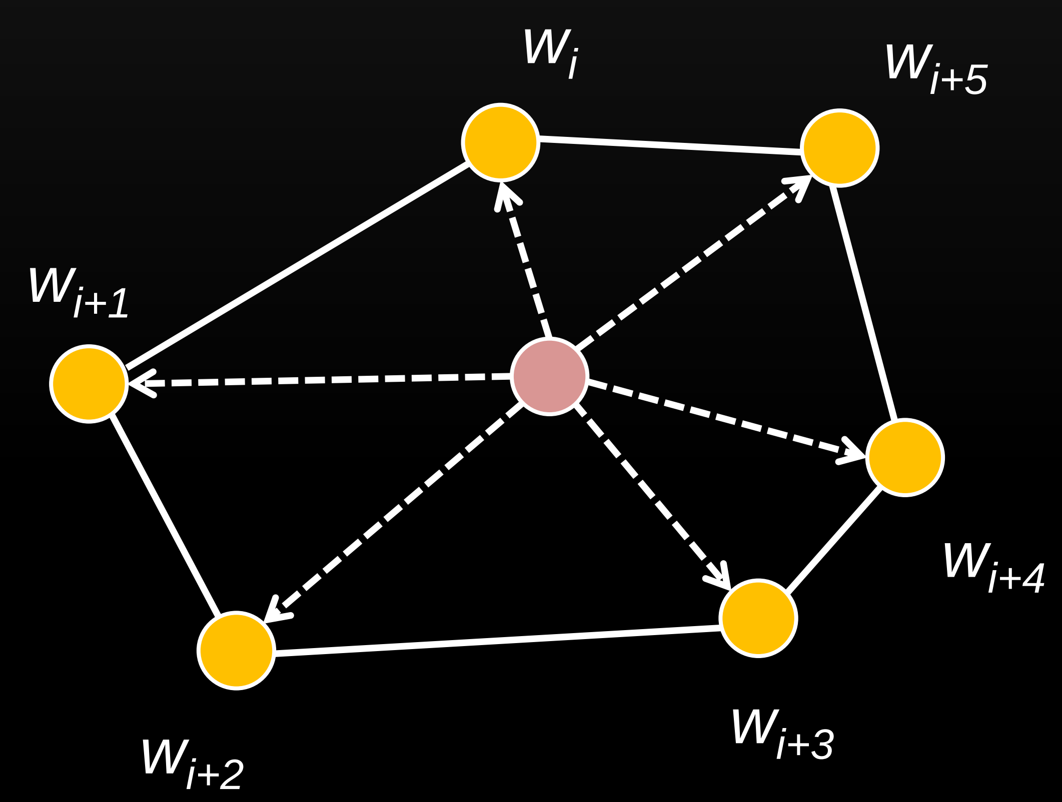
Shape animation data using X3D

- Deformable planar shape
  - Shape: vertices and faces of the shape mesh
  - Control: vertices and faces of the control mesh
  - Influences: relation between the shape and the control
    - Mean value coordinate can be used, then
    - It can be automatically computed from the shape and the control

```
DeformablePlanarShape: X3DGeometryNode {
    SFNode [in,out]  metadata    NULL  [X3DMetadataObject]
    SFNode [in,out]  shape       []    [PlanarShape]
    SFNode [in,out]  control     []    [PlanarShape]
    MFNode [in]      influences  []    [Influencer]
}
```

Shape animation data using X3D

```
Influencer:X3DGeometricPropertyNode {
     SFNode     [in,out]  metadata        NULL [X3DMetadataObject]
     MFInt32    [in,out]  coordIndex      []   (0, ∞)
     MFFloat    [in,out]  weights         []   [0,1]
}
```



- Vertex of shape mesh
- Vertex of control mesh

w_i : weights w.r.t. i

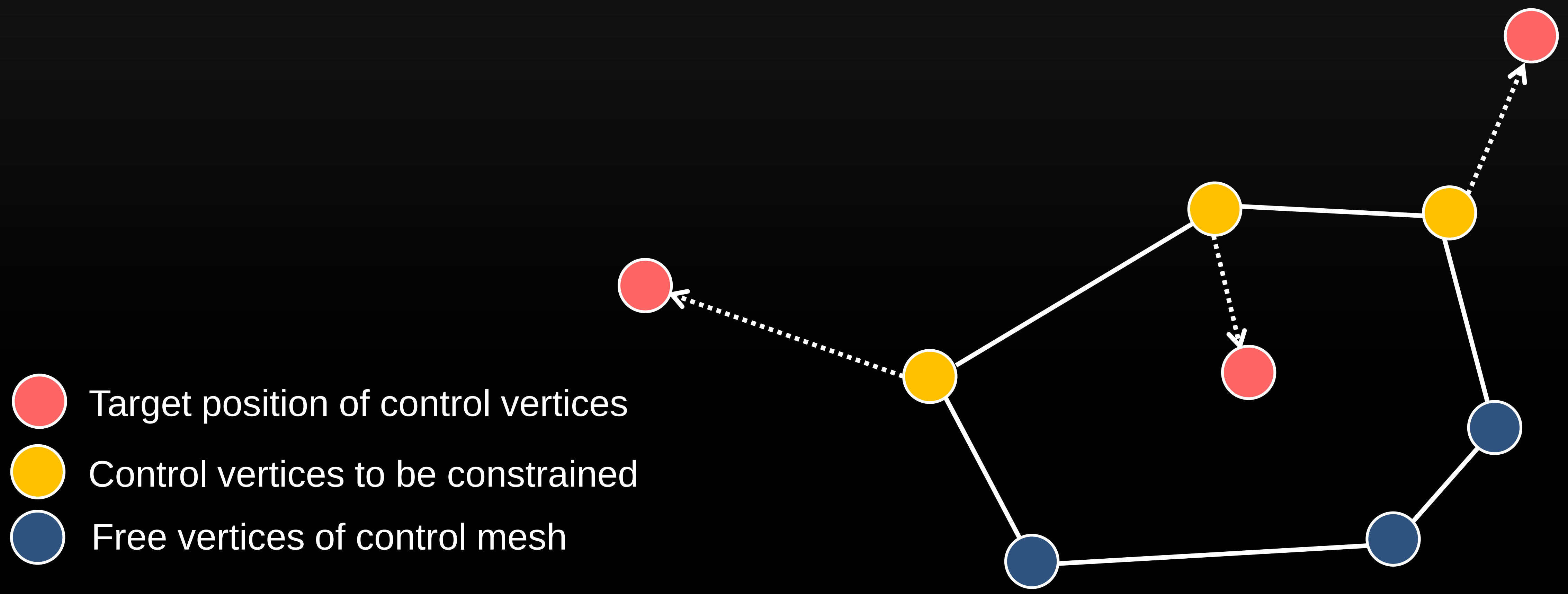
Shape animation data using X3D

- More controllable shape deformation
  - The control can also be computed using the small number of selected vertices

```
ConstrainedPlanarShape: X3DGeometryNode {
   SFNode [in,out]  metadata     NULL  [X3DMetadataObject]
   SFNode [in,out]  shape        []    [PlanarShape]
   SFNode [in,out]  control      []    [PlanarShape]
   MFNode [in]      weights      []    [Influencer]
   MFNode [in]      constraints  []    [PlanarShapeManipulator]
}
```

Shape animation data using X3D

```
PlanarShapeManipulator: X3DGeometricPropertyNode {
     SFNode     [in,out]  metadata        NULL [X3DMetadataObject]
     SFInt32    [in,out]  coordIndex      []    (0, ∞)
     SFFloat    [in,out]  targetPosition  []    (-∞, ∞)
}
```



Conclusion

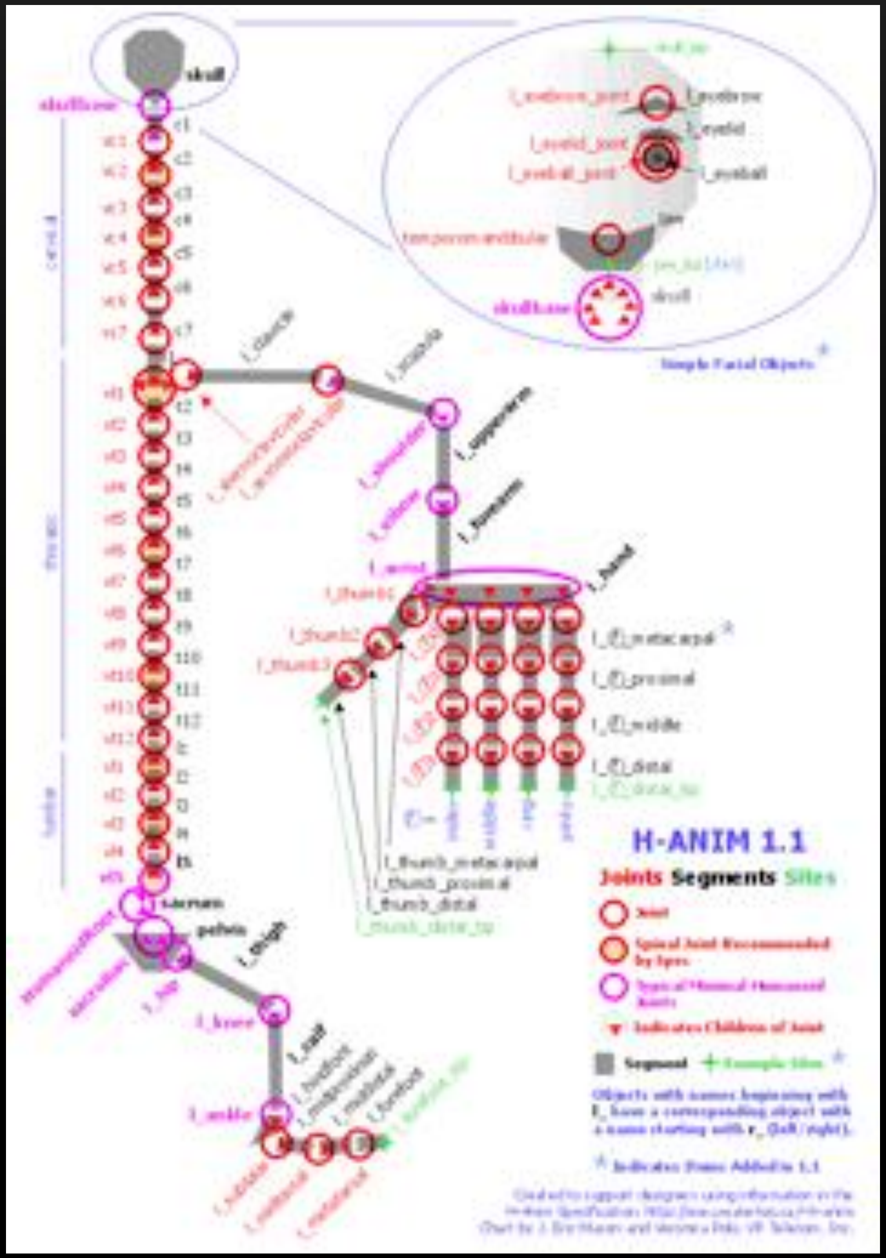
- Representing 2D shape animation
  - By means of spatial sampling
  - Shape deformation is an *available* technique
  - Related to geometry compression
- Future work items
  - Benefits
  - Use cases
  - Developing technical details
  - Defining the animation in X3D

Facial Animation in X3D

Jung-Ju Choi
Ajou University

H-Anim

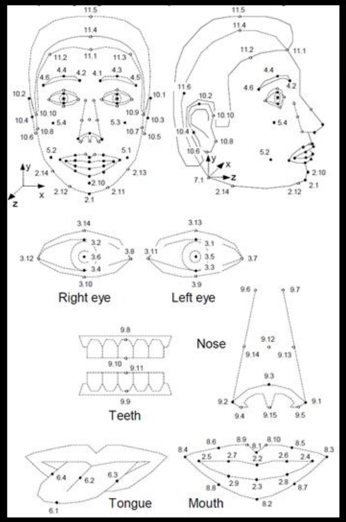
- ❖ ISO/IEC H-Anim
  - ◆ Define the animation of a humanoid character
  - ◆ Joints and their relation
  - ◆ Motion defined by either
    - ▪ Keyframing
    - ▪ Inverse Kinematics
    - ▪ Performance-based animation system
- ❖ H-Anim does not handle
  - ◆ Facial animation
- ❖ Objectives
  - ◆ Facial animation data of H-Anim



Marker-based facial animation

- Facial animation is defined by the marker positions at every frame
  - Compute the displacements between an expression and the neutral expression
  - Advantage
    - Reusable if the marker positions are restricted to the feature points (MPEG4)
  - Disadvantage
    - The more the markers, the larger the data
    - Large error for a big expression

Face capture



MPEG4 FAP

Synthesized face

Reduce the size of facial animation 1

- General compression technique
  - Principal Component Analysis
    - Clustering the marker positions into several groups
    - Each group is embedded to a 2D space (eigenvector space)
    - At least, 1/3 data size can be removed
  - Bit quantization
  - Entropy encoding

Reduce the size of facial animation 2

- Facial vertex animation
  - As similar as the 2D shape animation data
  - Define the motion of selected markers
  - Compute the motion of other markers

23,728 vertices
Width: 163.035
Height: 214.229
Depth: 128.017

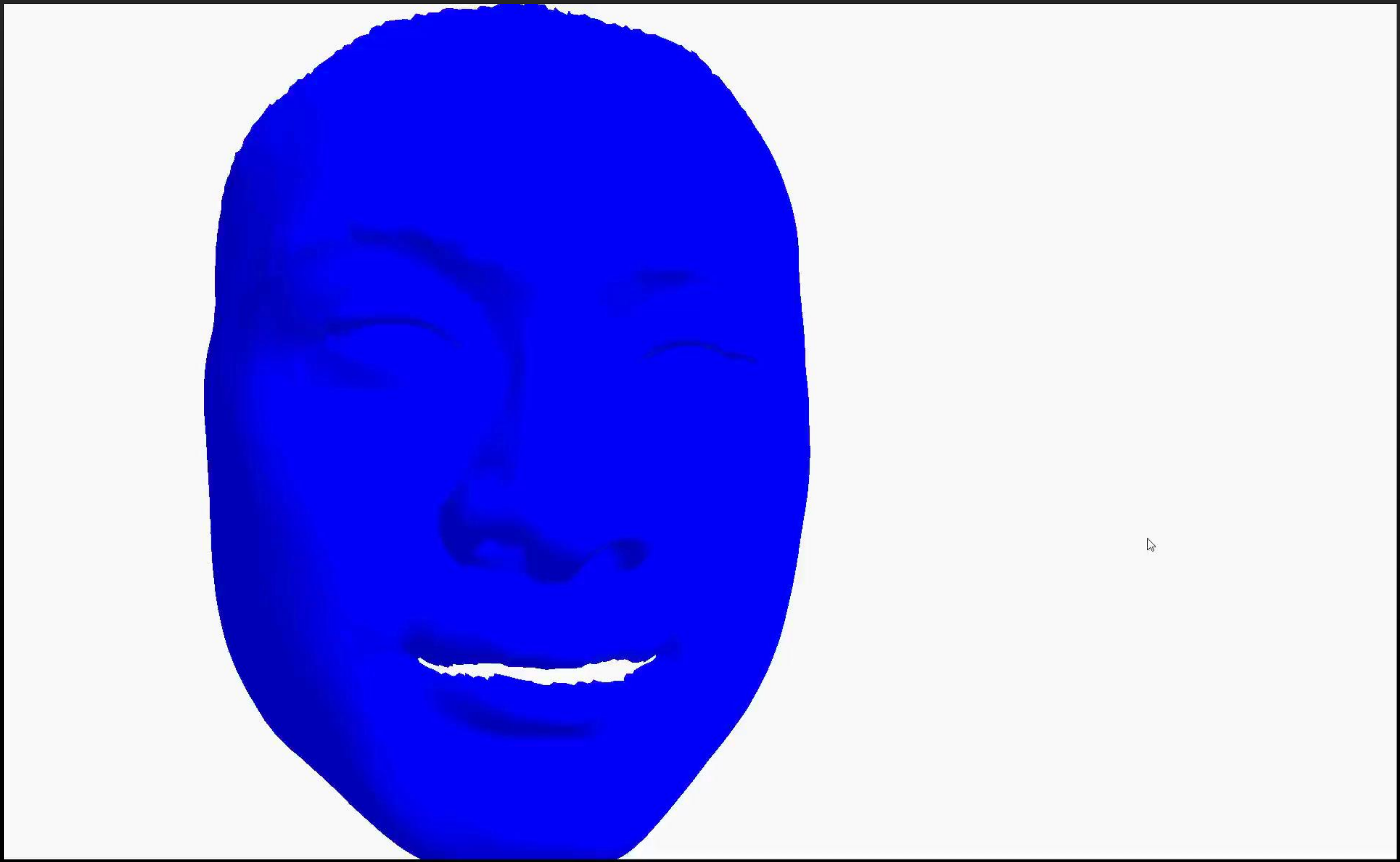
Reduce the size of facial animation 2

- Performance data
  - Using 3D MVC, but not using any facial components

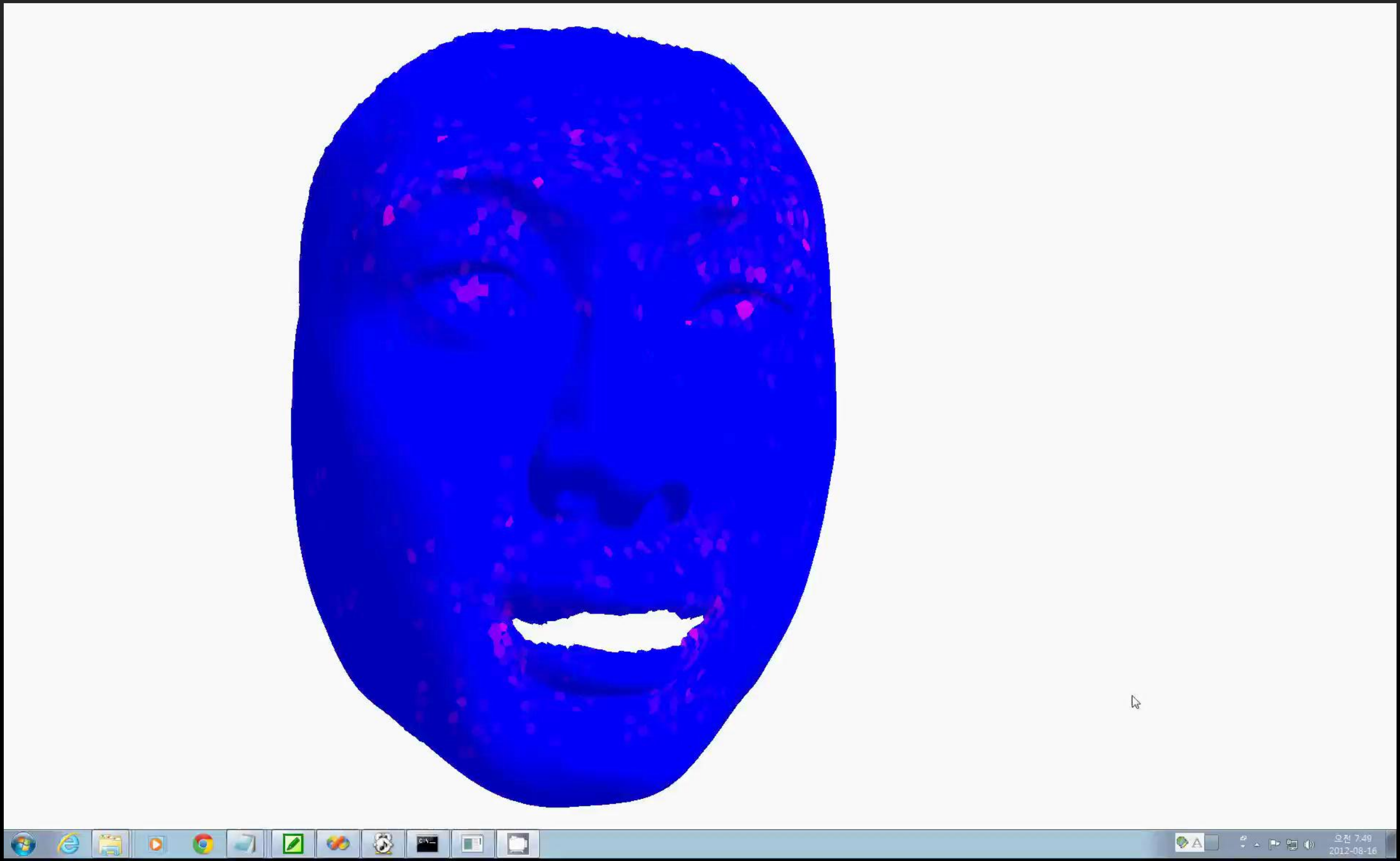
# Reconstructed	# Samples	RMSE	Max Diff.
6,294	17,434 (73%)	0.0216	0.3086
8,087	15,641 (66%)	0.0802	1.0558
12,763	10,965 (46%)	0.1339	2.6047
15,772	7,956 (33%)	0.2190	3.1428
19,482	4,246 (17%)	0.6252	11.4721

  - More experiments are yet required

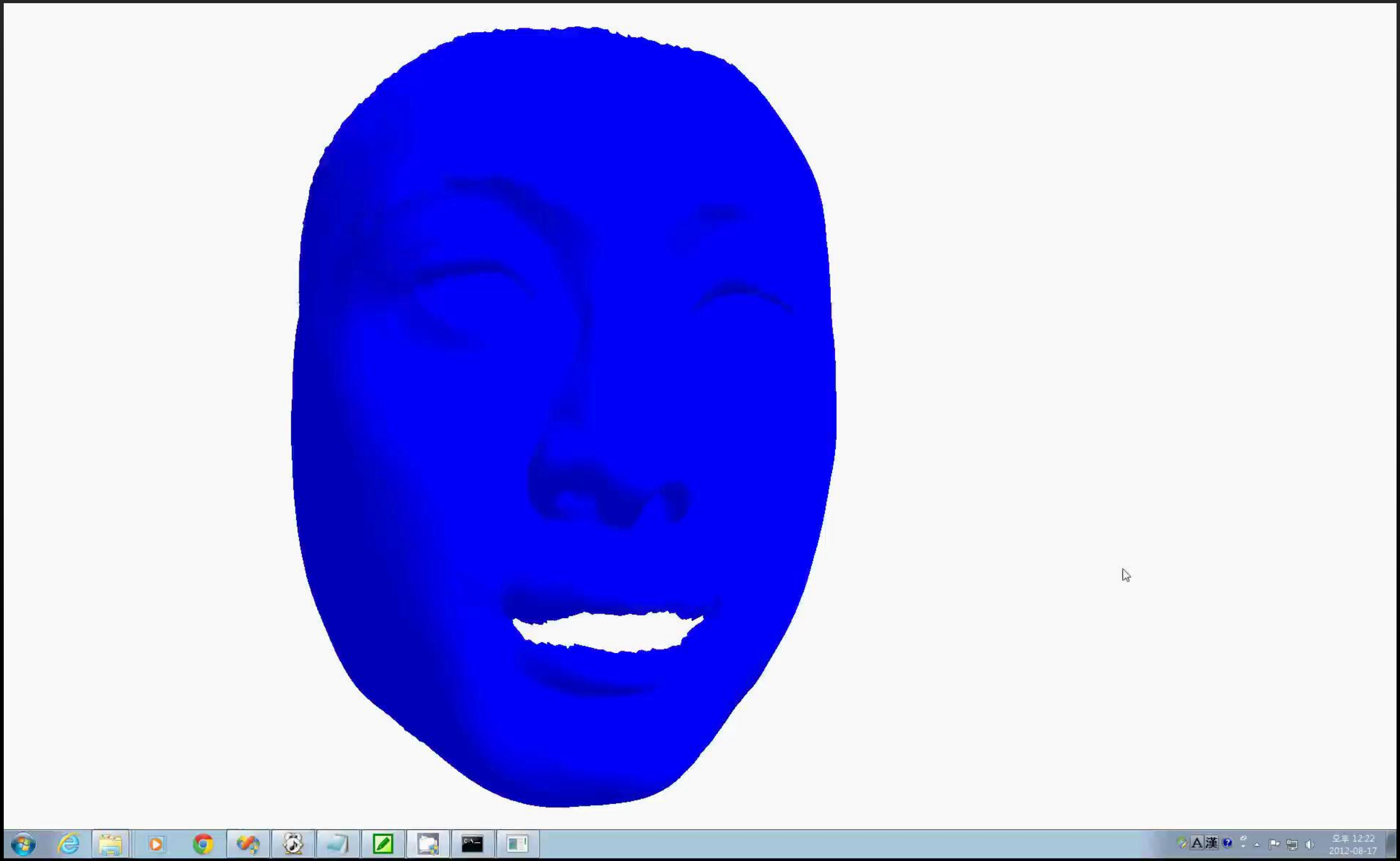
Ground truth facial animation



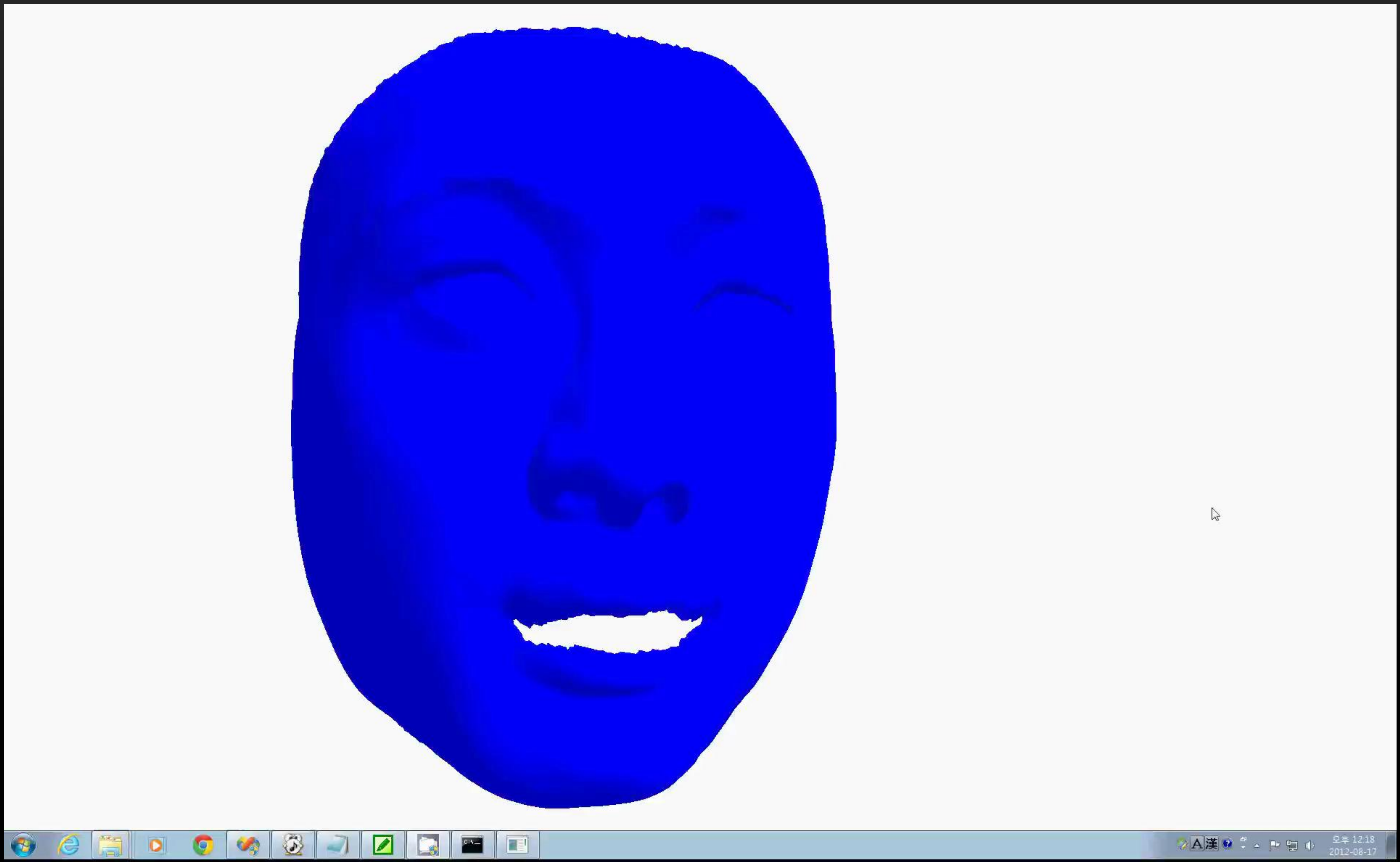
From 73% samples



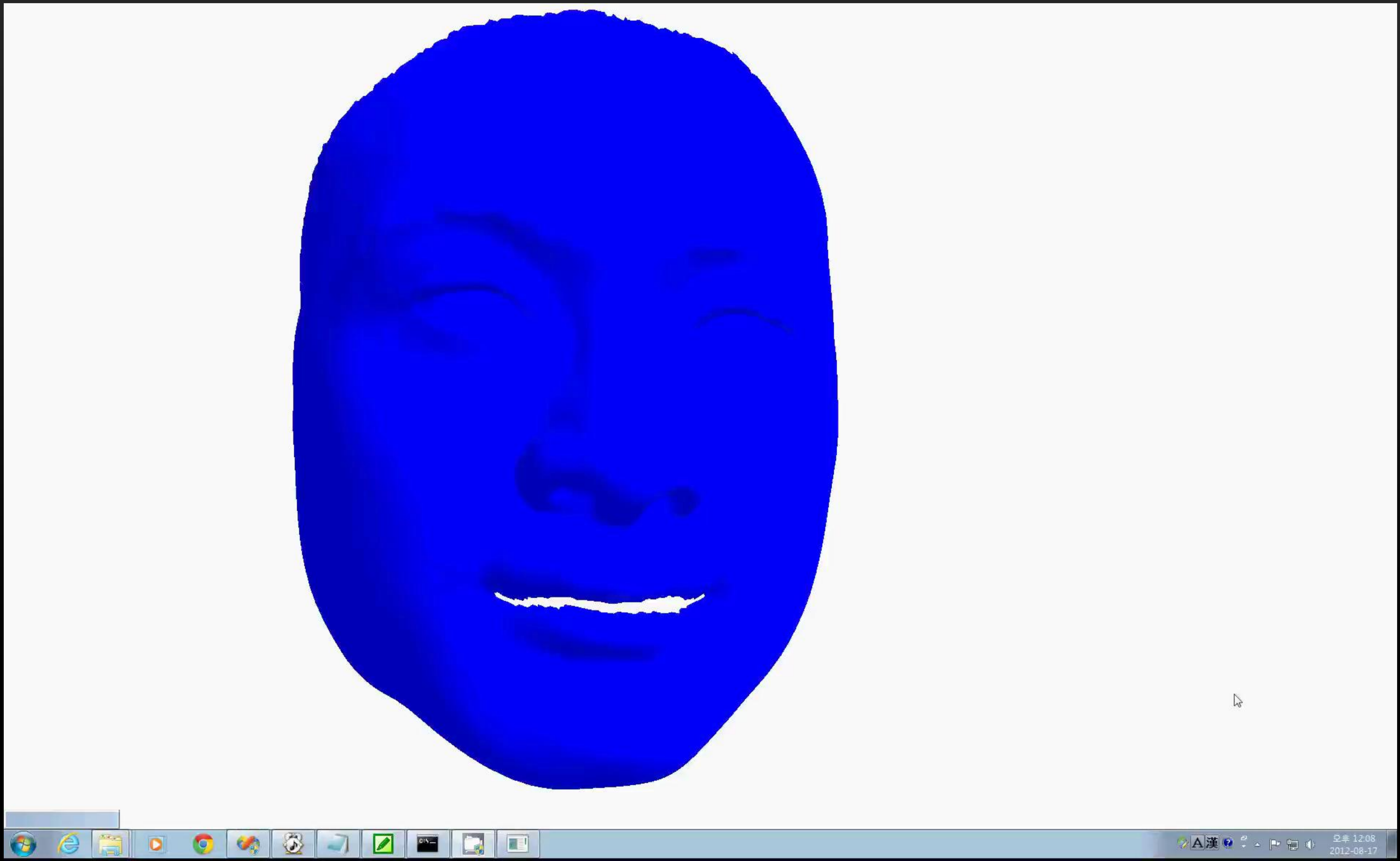
From 66% samples



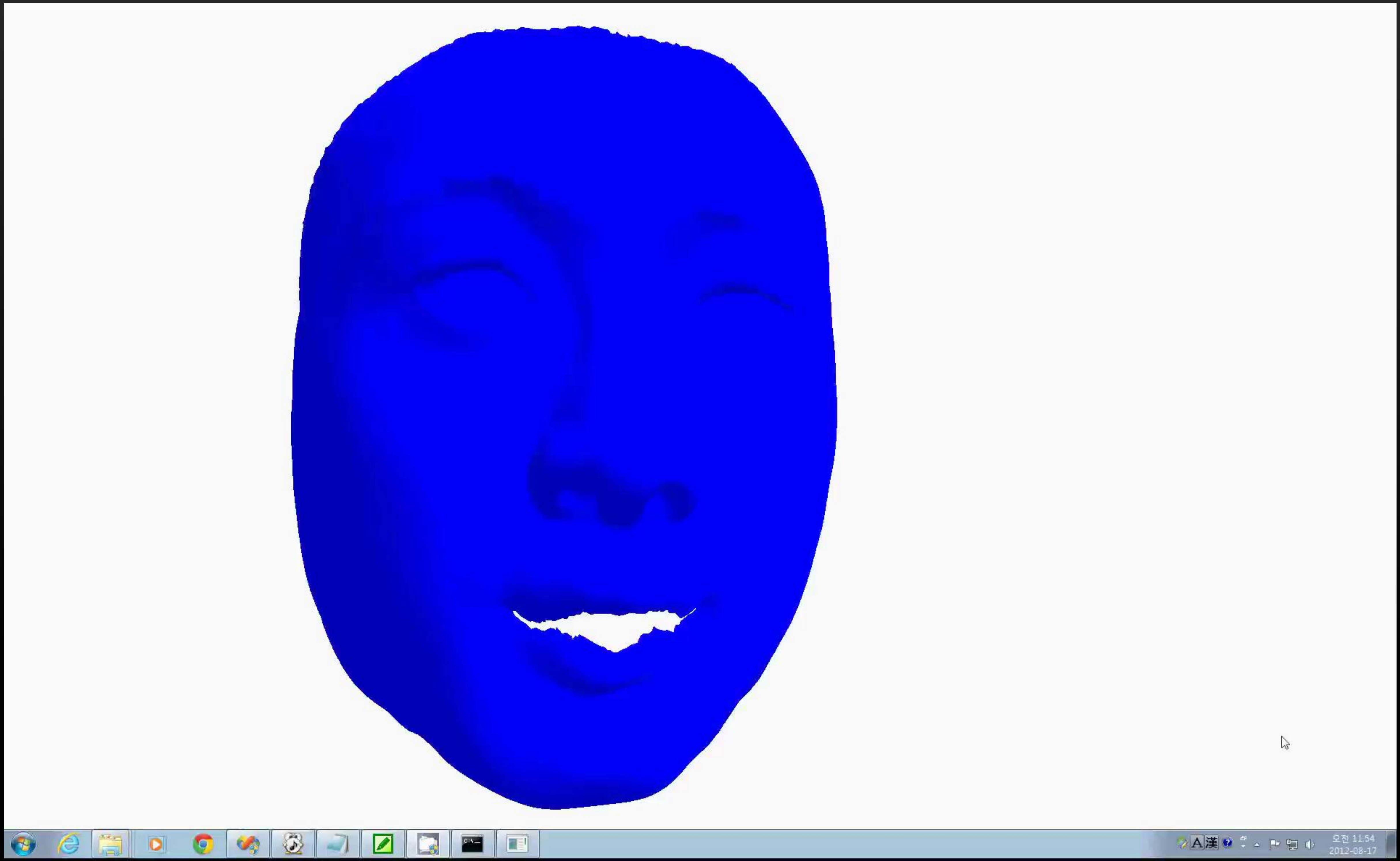
From 46% samples



From 33% samples



From 17% samples



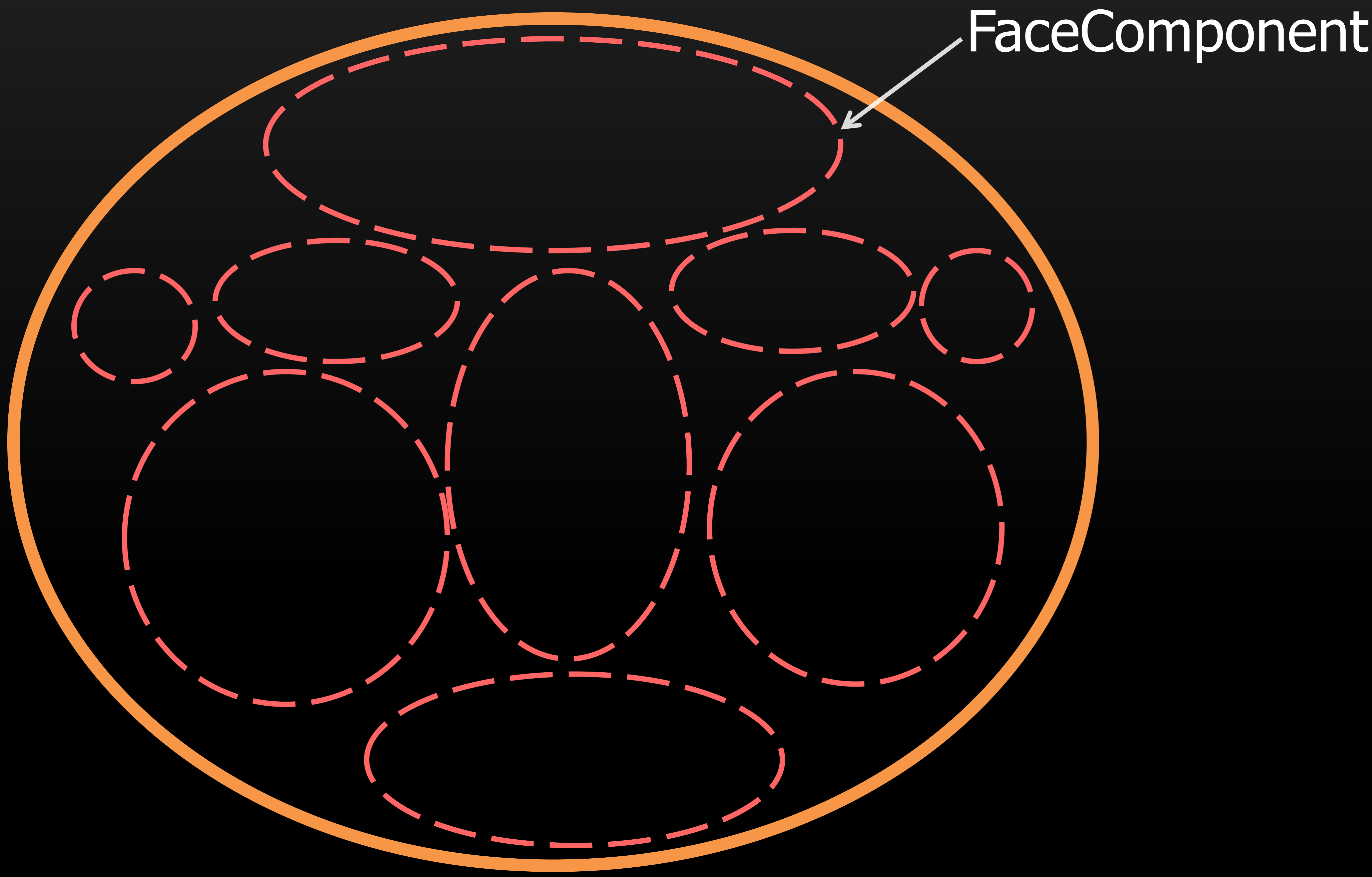
Facial Animation using X3D

- Face component
  - Defines a set of vertices
  - Typically, a meaningful subset of face vertices

```
FaceComponent: X3DChildNode {
  SFNode     [in,out]  metadata      NULL  [X3DMetadataObject]
  SFString   [in,out]  name          ""
  MFInt32    [in,out]  coordIndex    []    [0, ∞)
  MFNode     [in,out]  manipulators  []    [FaceManipulator]
}
```

Facial Animation using X3D

❖A FaceComponent is deformed by a FaceManipulator.



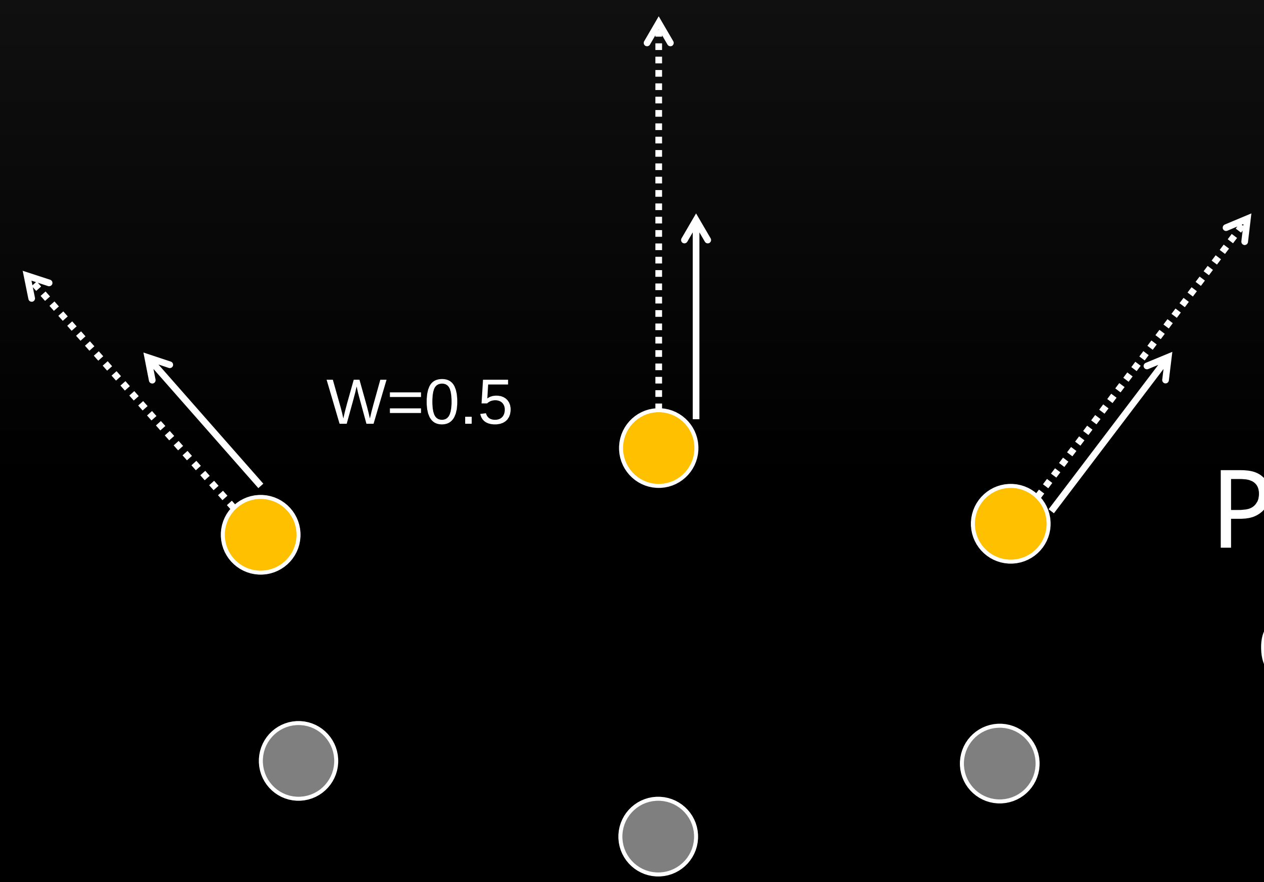
Facial Animation using X3D

- Face manipulator
  - Defines the position constraints of deformation

```
FaceManipulator: X3DGeometricPropertyNode {
   SFNode    [in,out] metadata        NULL [X3DMetadataObject]
   SFString  [in,out] name            ""
   MFInt32   [in,out] coordIndex      []   [0, ∞)
   MFVec3f   [in,out] targetDirection []   (-∞, ∞)
   SFFloat   [in,out] weight          0.0  (-∞, ∞)
}
```

Facial Animation using X3D

- Feature vertex to be moved
- Free vertex in facial part
- Target direction vector
- Actual motion by weight(w=0.5)



Position of free vertex is determined by feature vertex motion.

Facial Animation using X3D

- Action manipulator
  - Defines the control information for whole facial animation
  - Each manipulator can represent each component of facial expressions such as "open the left eye"

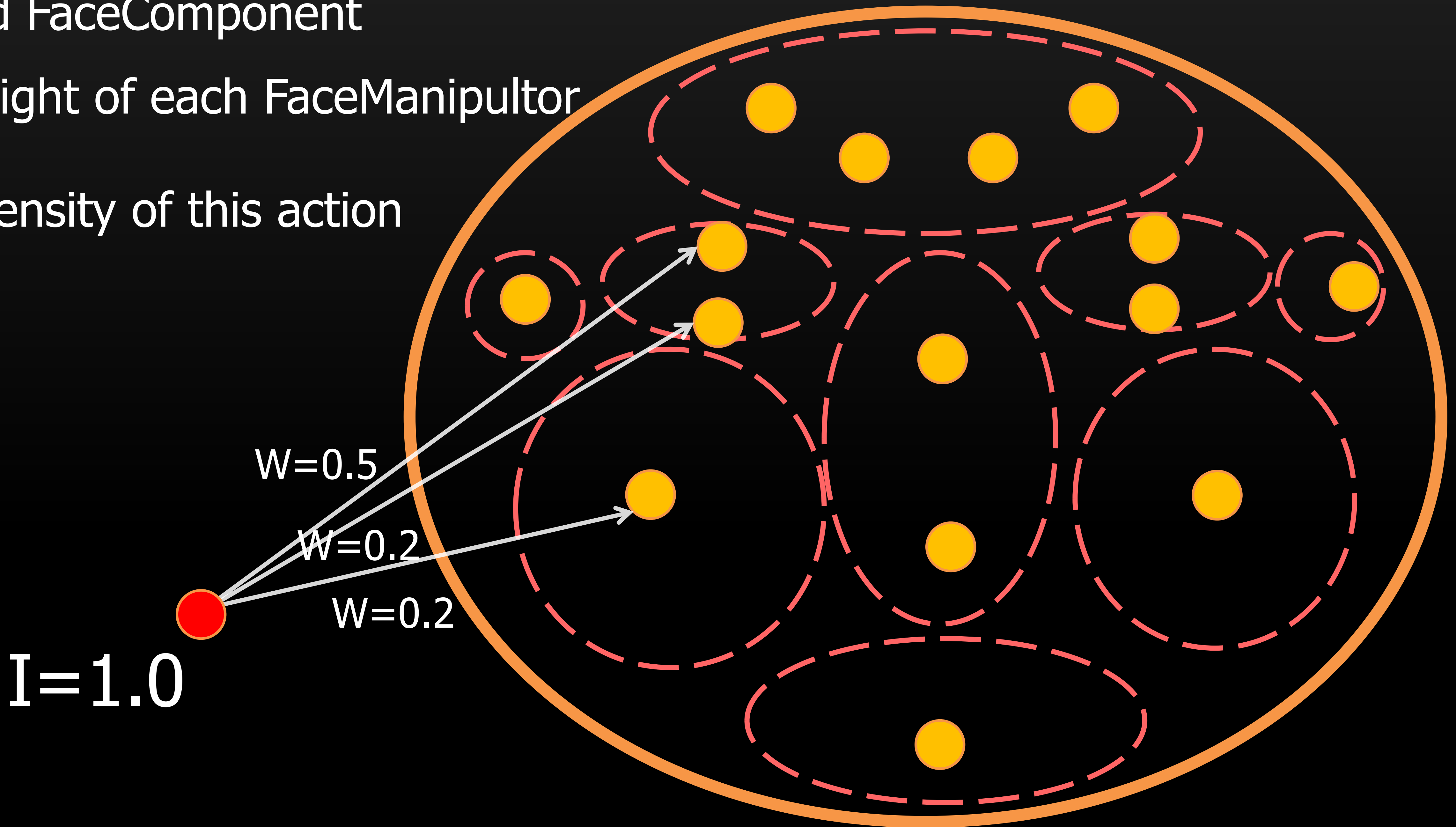
```
ActionManipulator: X3DGeometricPropertyNode {
  SFString [in,out]  name              ""
  MFInt32   [in,out]  componentIndex    []      [0, ∞)
  MFInt32   [in,out]  manipulatorIndex  []      [0, ∞)
  MFFloat   [in,out]  targetWeight      []      (-∞, ∞)
  SFFloat   [in,out]  intensity         0.0     [0, 1]
}
```

Facial Animation using X3D

- FaceManipulator node of each facial part
- ActionManipulator indexing of FaceManipulator and FaceComponent

W Weight of each FaceManipultor

I Intensity of this action



Conclusion

- Representing 3D facial animation
  - Select a small number of vertices,
  - Define a sequence of motion of the selected vertices,
  - Reconstruct the motion of other vertices using the positions of selected vertices at run time.
- Future work
  - How to increase benefits
  - How to select a small set of vertices
  - Reconcile with mpeg4 facial animation