Compression of Facial Animation Data in the Novel Data Acquisition Process

Sangwook Yoo

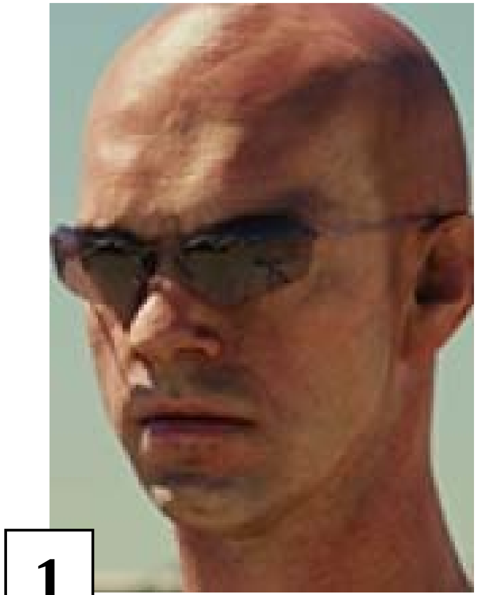
KAIST (Korea Advanced Institute of Science and Technology)

Outline

- Introduction
- Background
  - Stereo vision
  - Mesh compression
  - Optical flow
- My Idea
- Conclusion

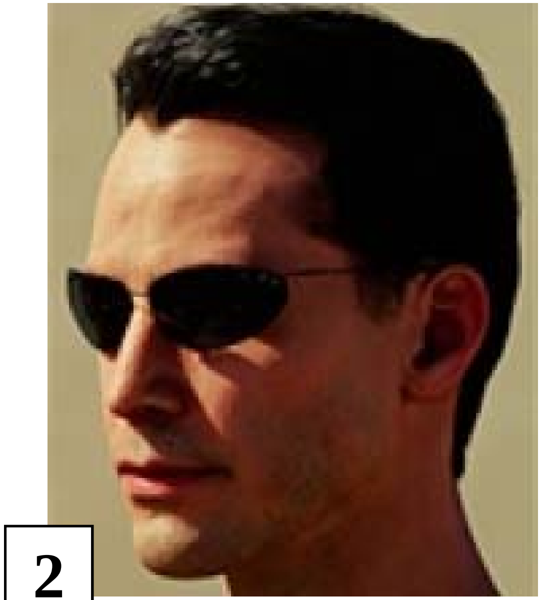
Introduction

- **3D Facial animation**
  - **Reality is important**
    1) **Use capture images**
    2) **Computer vision technique for convincing result**



1

Synthesized



2

Synthesized

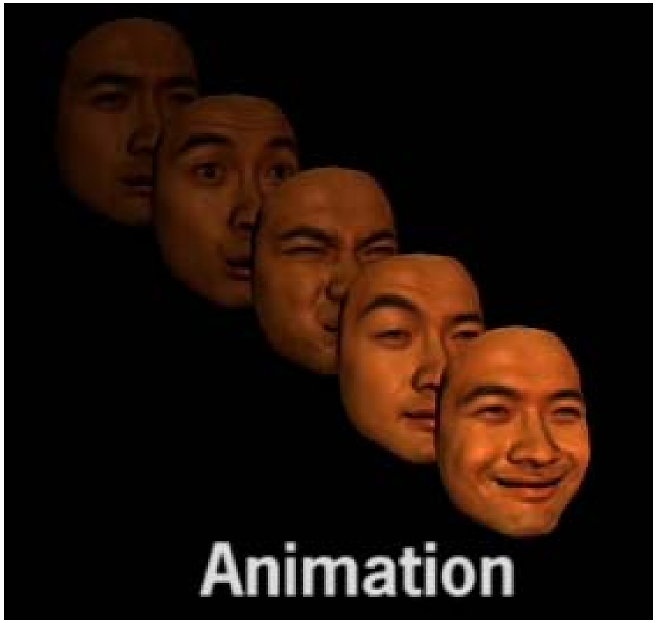
3

Real Face

Introduction

- **Compression of 3D Meshes is needed**
  - **Raw data cannot even be loaded as it is ...**
    1) **Spatially large**
    2) **Temporally large**

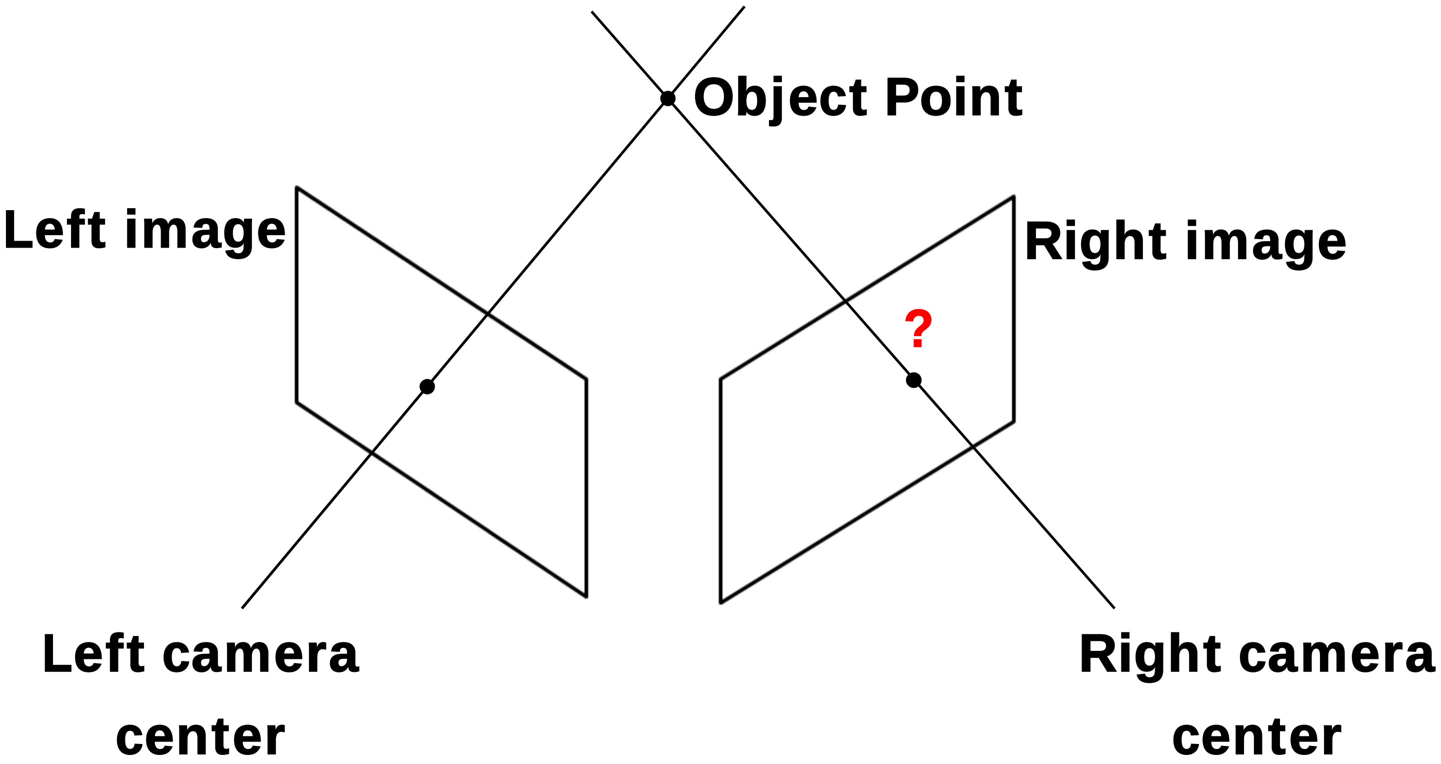
Spatially large



Temporally large

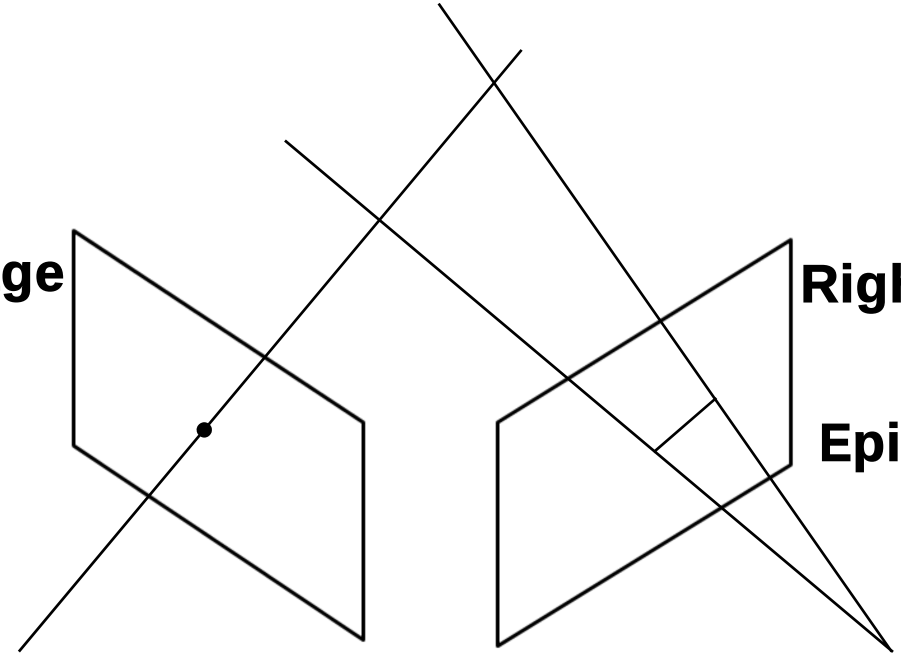
Background

- Stereo Vision
  - Depth estimation from stereo image
  - Correspondences need to be estimated



Background

- Epipolar Line
  - Line on which the corresponding point must lie



Left image

Right image

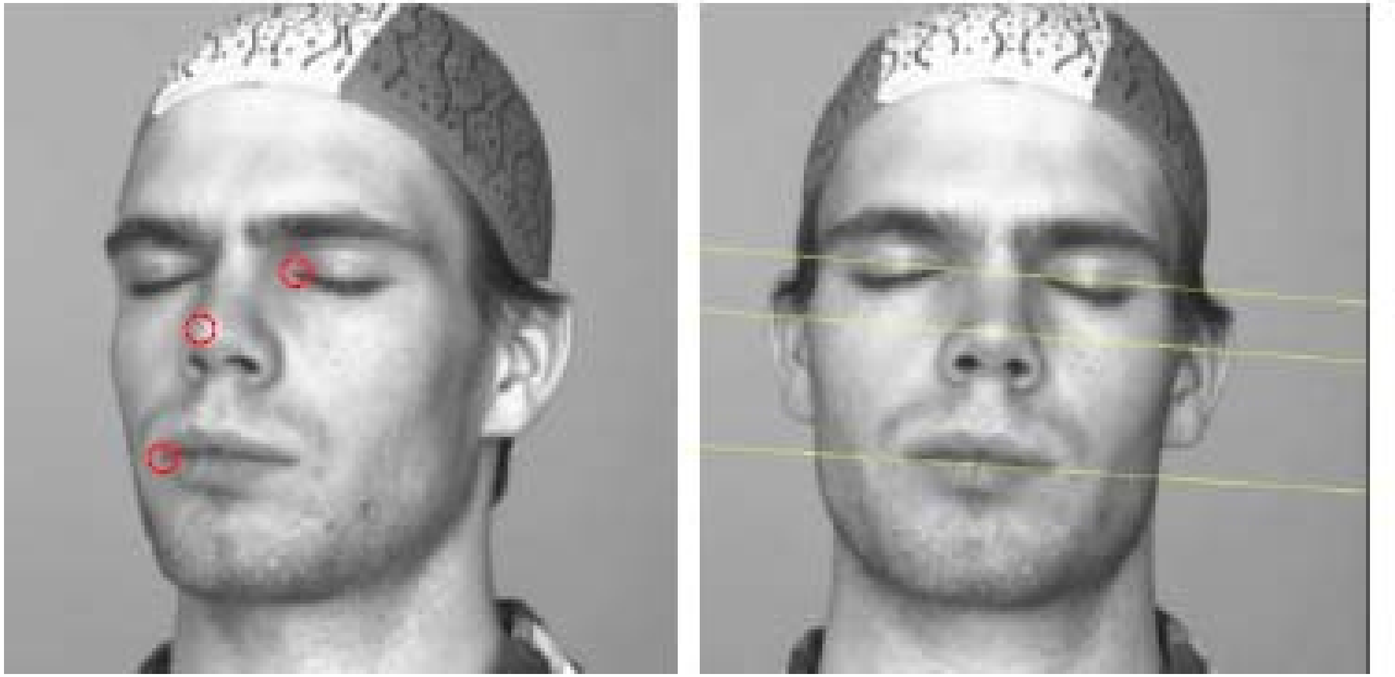
Epipolar line

Left camera center

Right camera center

Background

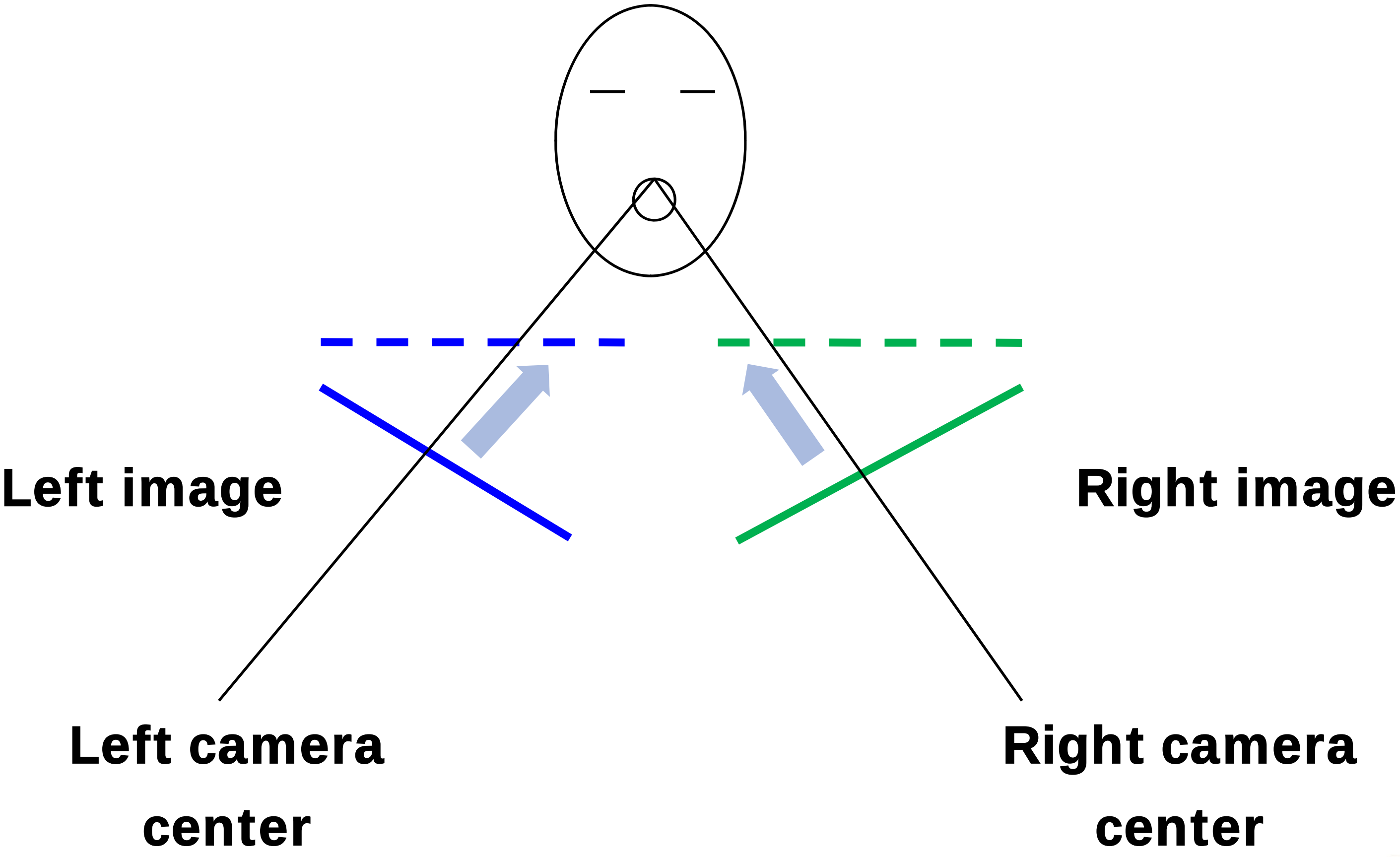
- To find corresponding point,
  - We just need to search the corresponding epipolar line



- Pixels in the epipolar line ?

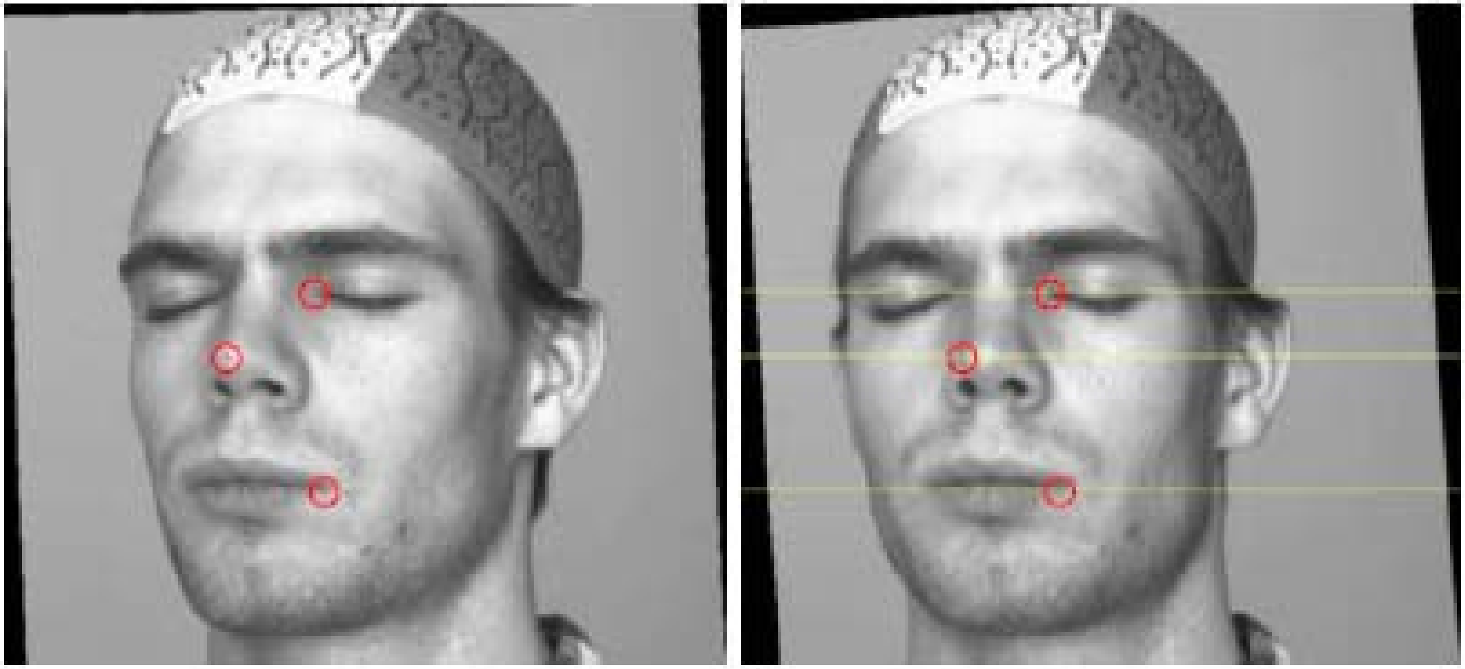
Background

- Retification
  - Reproject into parallel virtual image planes



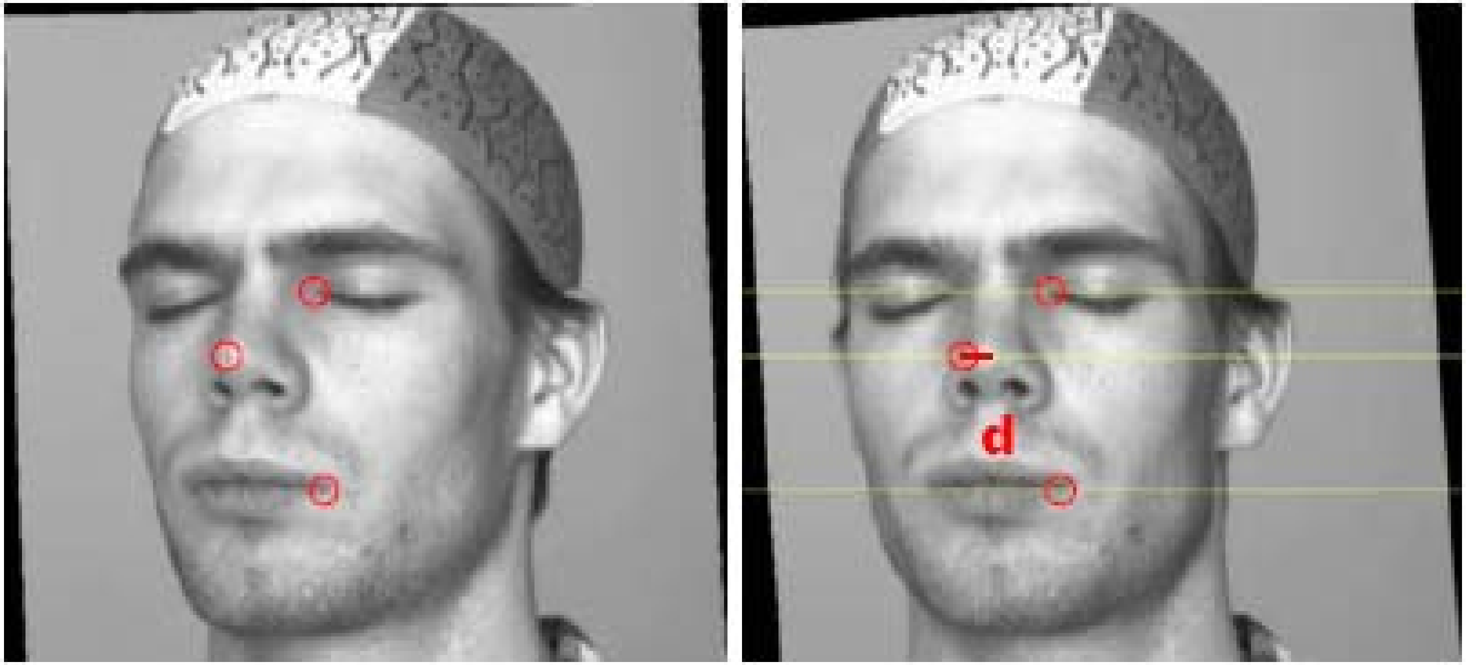
Background

- Retification
  - Result



Background

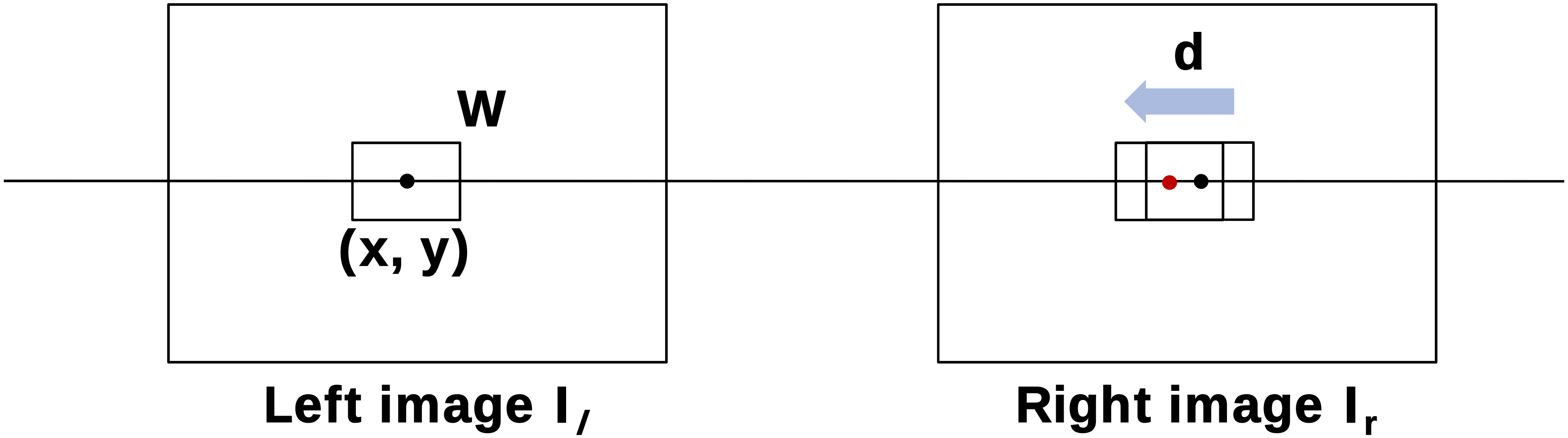
- **Disparity**
  - **Find horizontal shift d along epipolar line**



- **Pixel by Pixel ?**

Background

- Disparity
  - To resolve ambiguity, match small windows W around (x, y)



$$ssd(d) = \sum_{(x,y)\in W} e(I_l(x,y), I_r(x-d,y)), \quad e(a,b) = (a-b)^2$$

Background

- **Efforts for better result**
  - **Project Stripe patterns regularly**
    - **To give features on the featureless area**

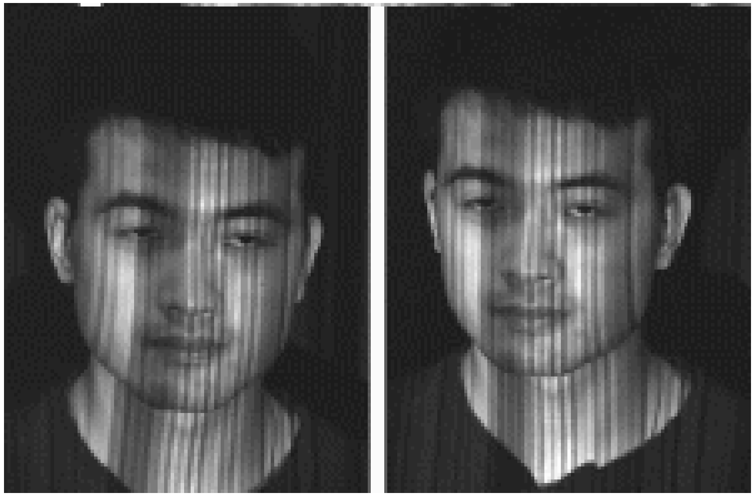
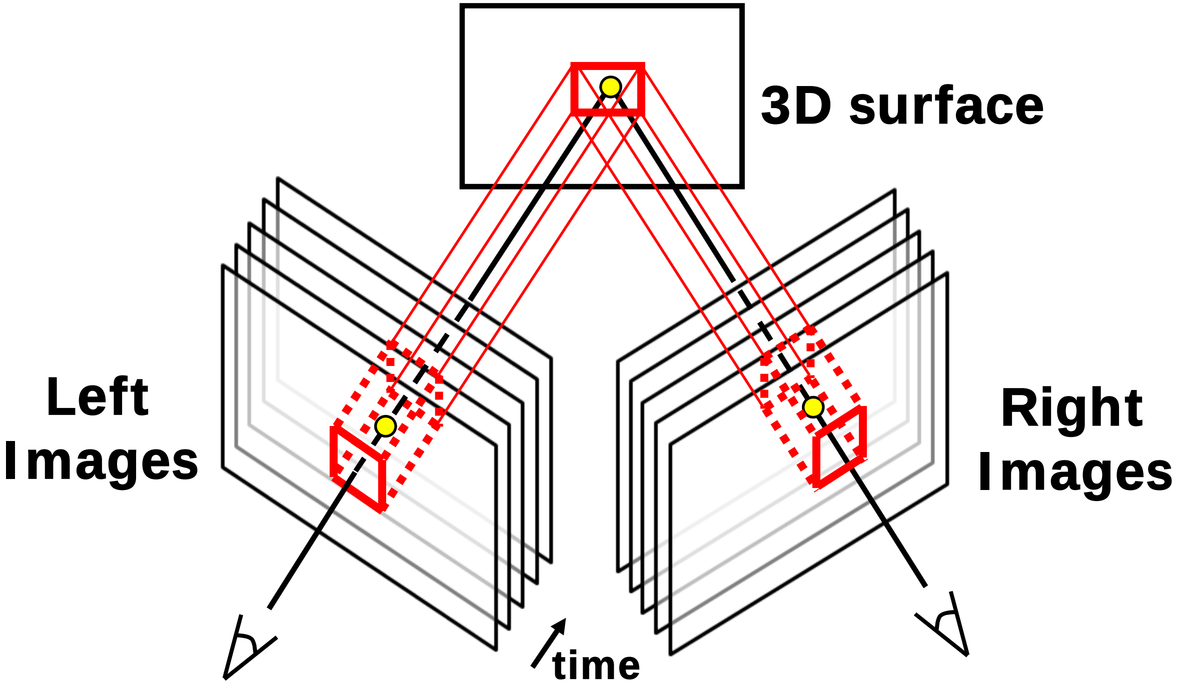


Image courtesy of L. Zhang

- **However, natural performance is difficult due to the flickering**

Background

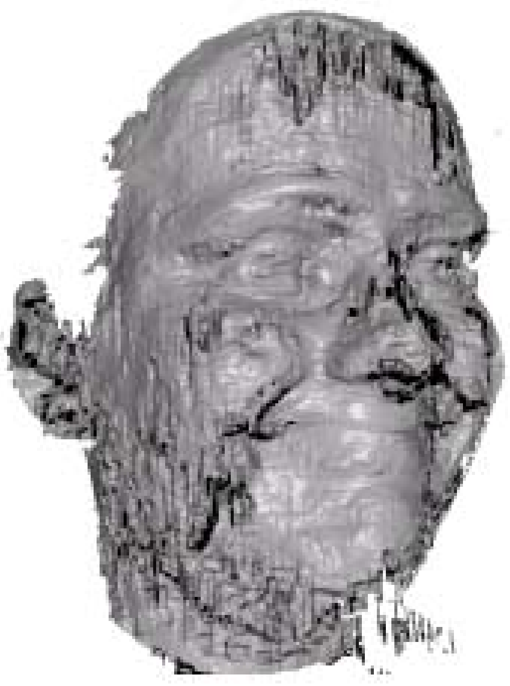
- **Efforts for better result**
  - **Temporal window matching [Zhang et al. 03]**



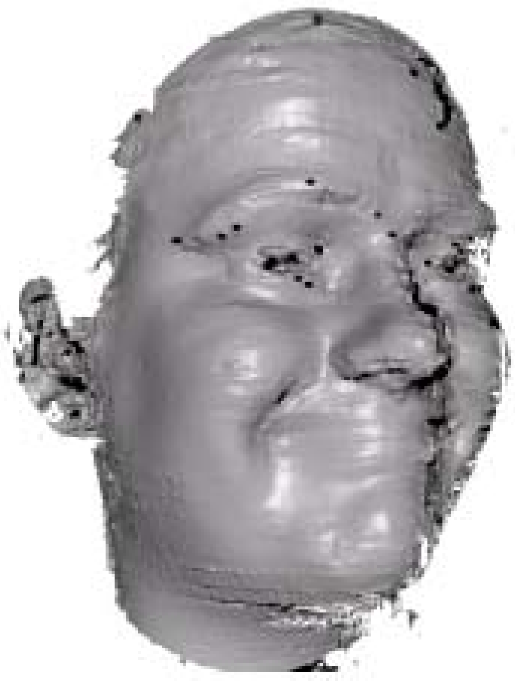
$$ssd(d) = \sum_{(x,y,t)\in W} e(I_l(x,y,t), I_r(x-d,y,t)), \quad e(a,b) = (a-b)^2$$

Background

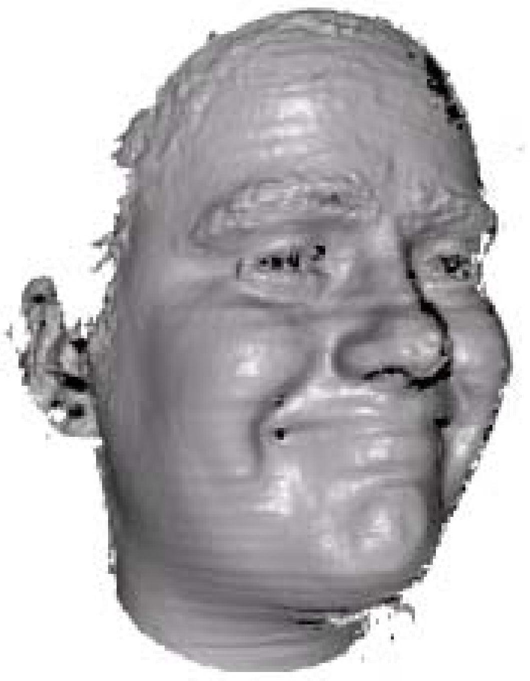
- Efforts for better result
  - Result



Before



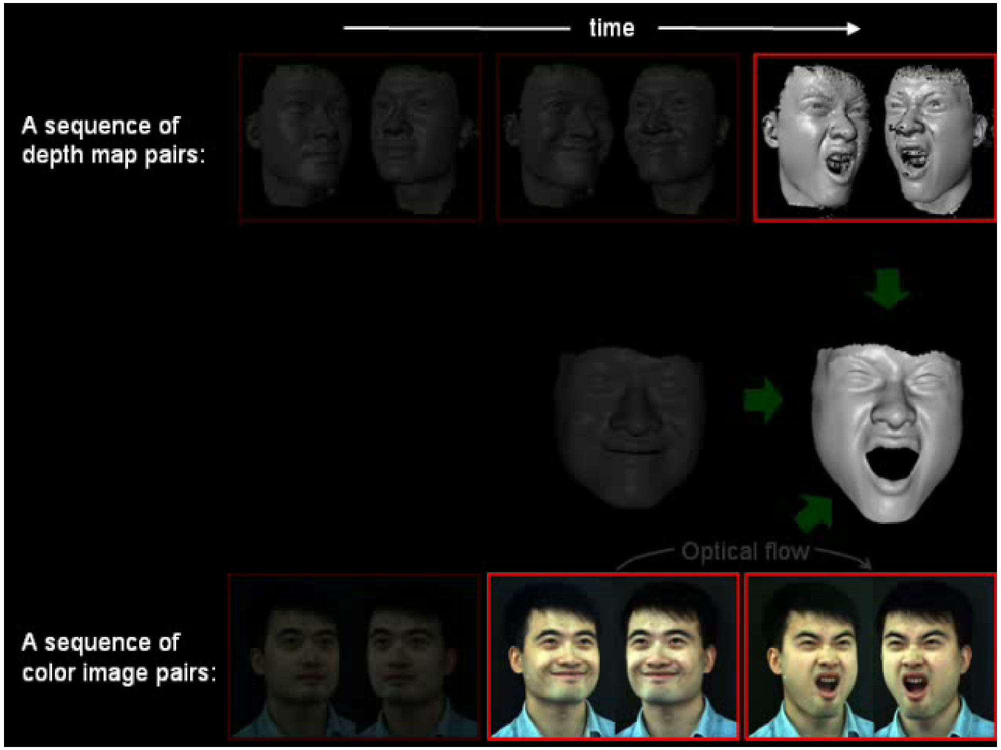
Stripe pattern



Stripe pattern
+
Cubic matching

Background

- Template mesh tracking [Zhang et al. 04]
  - Compute the template mesh for each frame such that shape matches the depth information


time
A sequence of depth map pairs:
Optical flow
A sequence of color image pairs:

Video courtesy of L. Zhang

Background

- **3D Mesh Compression**
  - **To accelerate the rendering**
  - **For visualizing and simulating in networked environment**
    1. **Single-rate compression**
       1) **Lossless: Remove the redundancy**
       2) **Lossy**
    2. **Progressive compression**

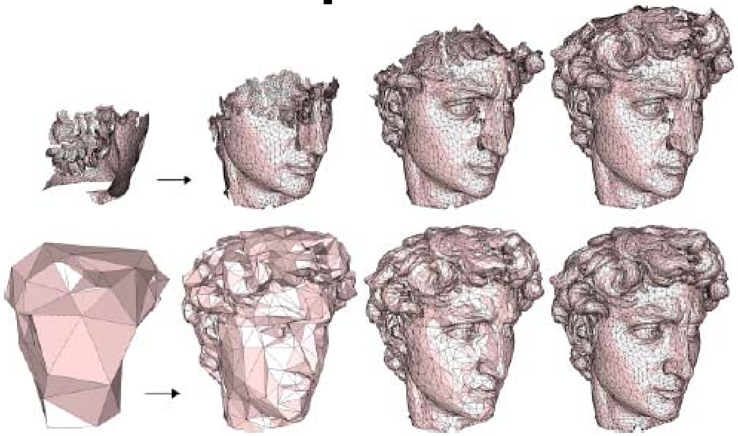
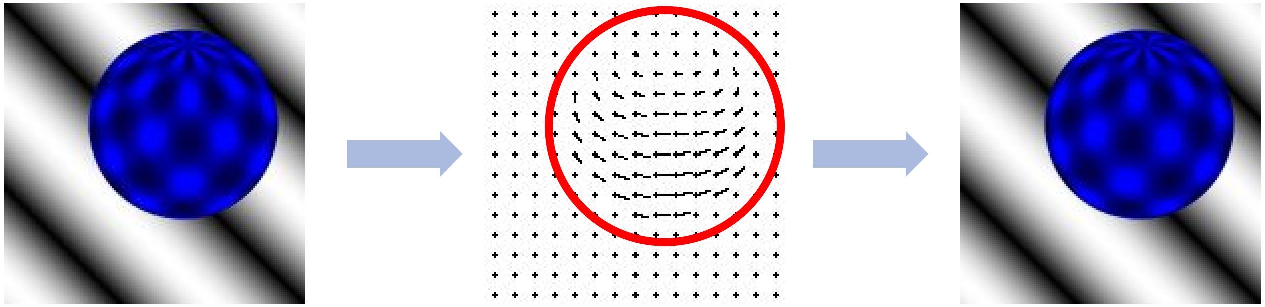


Image courtesy of P. Alliez and C. Gotsman

Background

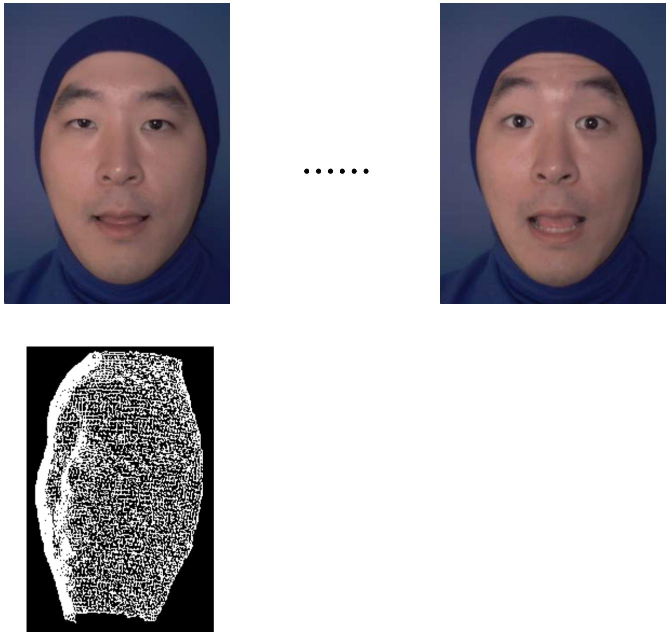
- **Optical Flow**
  - **Estimates the motion of object in the consecutive images**



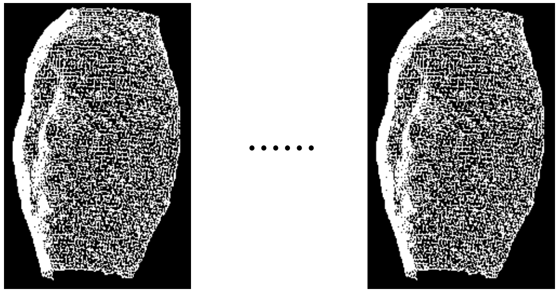
  - **Three assumptions**
    - **Gray value constancy assumption**
    - **Gradient constancy assumption**
    - **Smoothness assumption**

My idea

- **Data Acquisition + Compression**

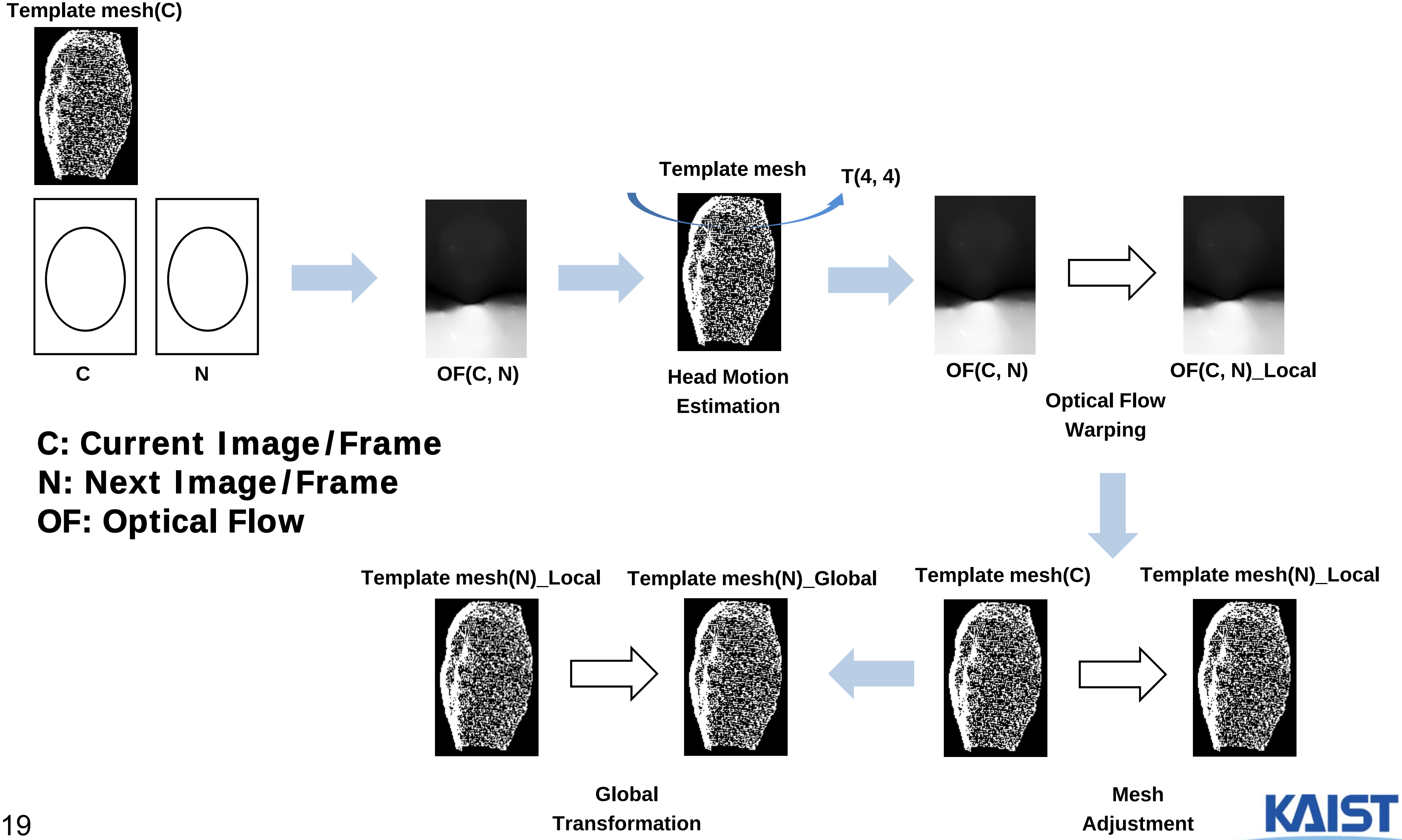


Given



Compressed 3D Meshes

My idea - Overview



Template mesh(C)

C

N

OF(C, N)

Template mesh

T(4, 4)

Head Motion Estimation

OF(C, N)

OF(C, N)_Local

Optical Flow Warping

Template mesh(N)_Local

Template mesh(N)_Global

Template mesh(C)

Template mesh(N)_Local

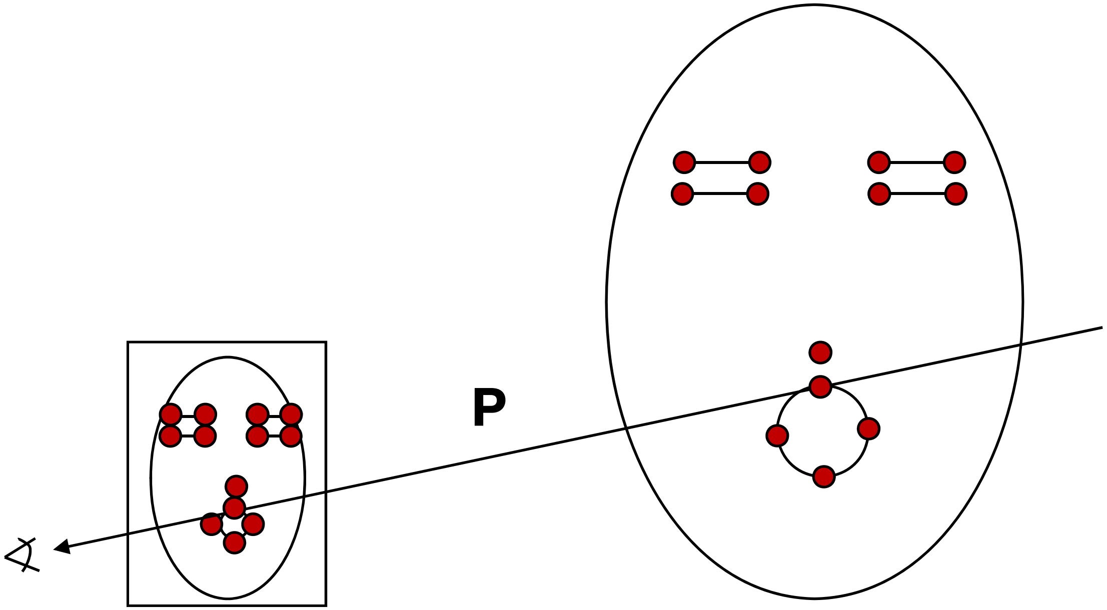
Global Transformation

Mesh Adjustment

C: Current Image/Frame
N: Next Image/Frame
OF: Optical Flow

My idea

- Projection matrix P estimation
  - Use interactively specified feature points
  - Mapping function from 3D to 2D

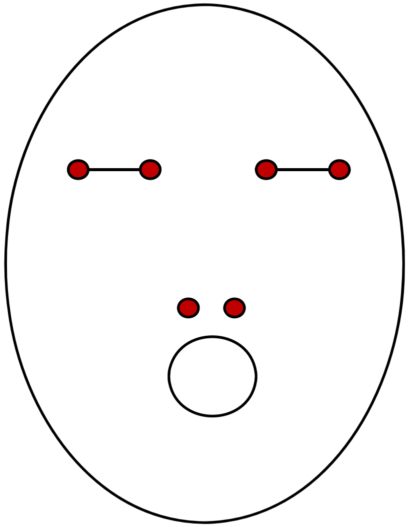


Image

Template mesh

My idea

- Global Head Motion Estimation
  - Head tends to move while doing the performance
  - To estimate the head motion, specify vertices to track
    - which are difficult to move locally

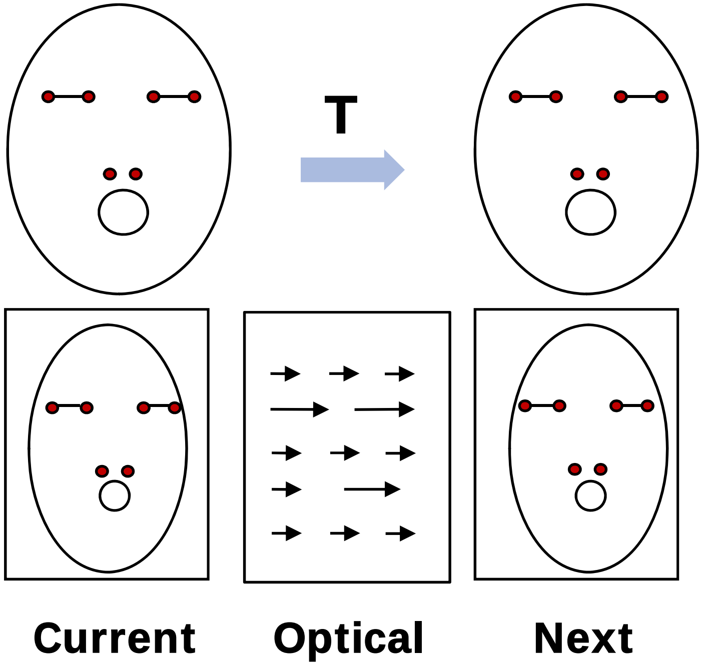


Template mesh

My idea

- Global Head Motion Estimation
  - From optical flow, next 2D position of predefined feature vertices can be known
    - Estimate the rigid transformation matrix T

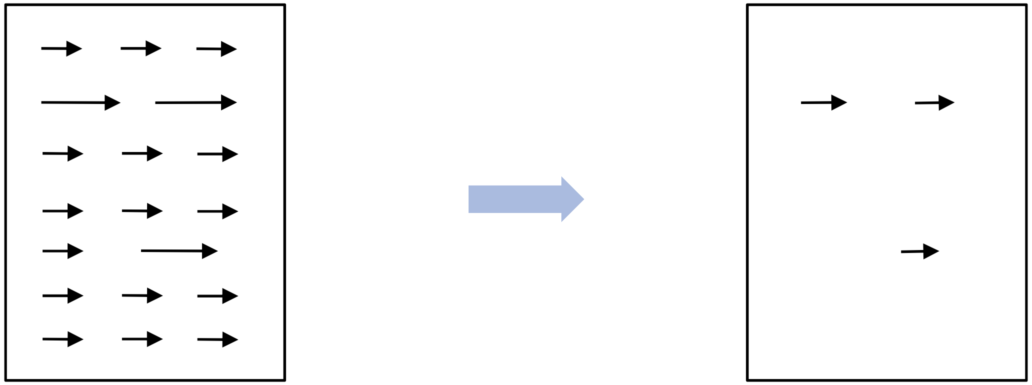
Keep track some Predefined feature vertices



Flow

My idea

- Optical Flow Warping
  - Remove Global Motion Effects using T
    - e.g. smile, translation → smile



Local motion

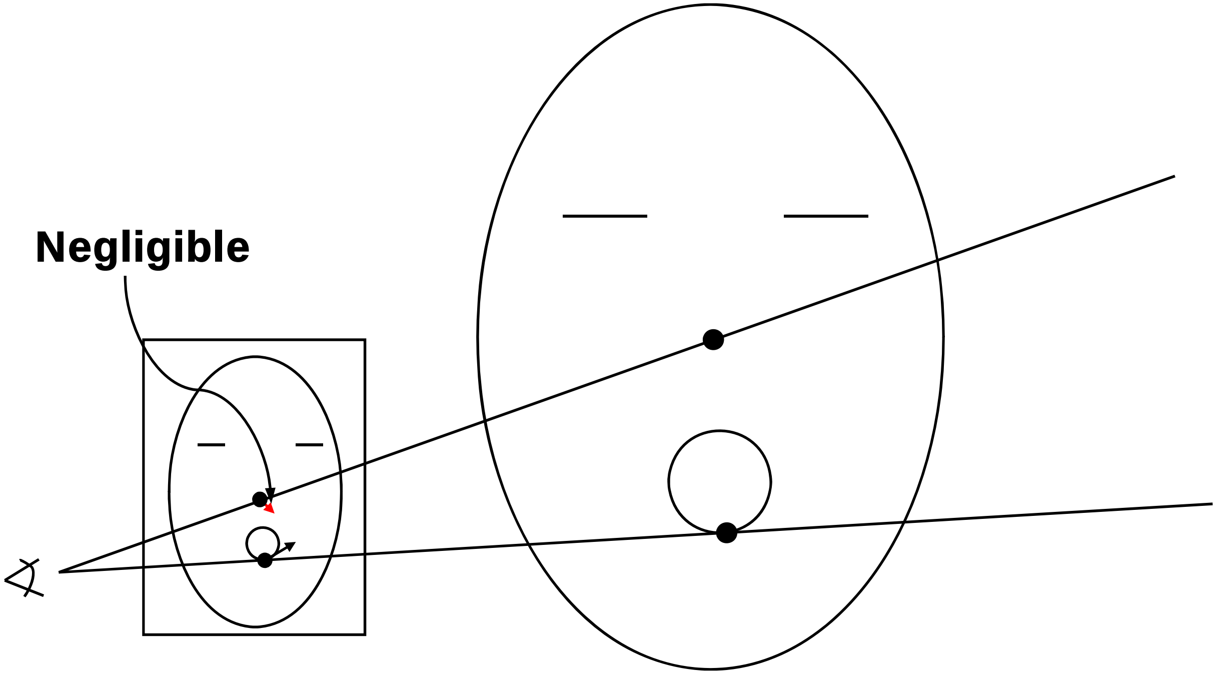
+

Global motion

Local motion only

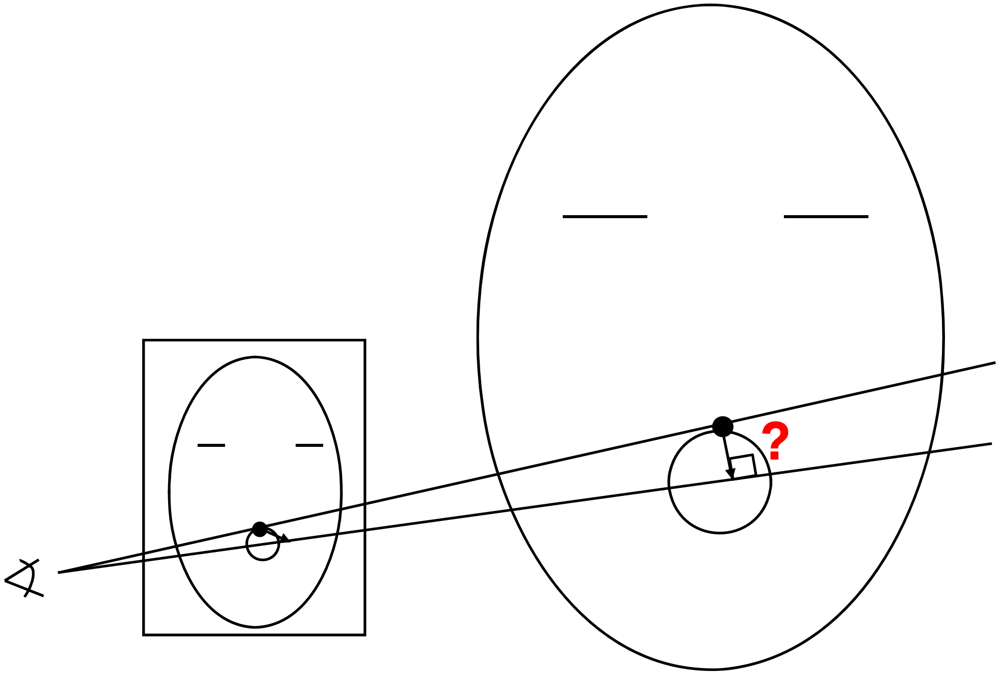
My idea

- Estimate next position of each vertex using optical flow
  - If the corresponding optical flow is negligible,
    - Do not compute & store next position for that vertex
    - Compression effect: Redundancy removed



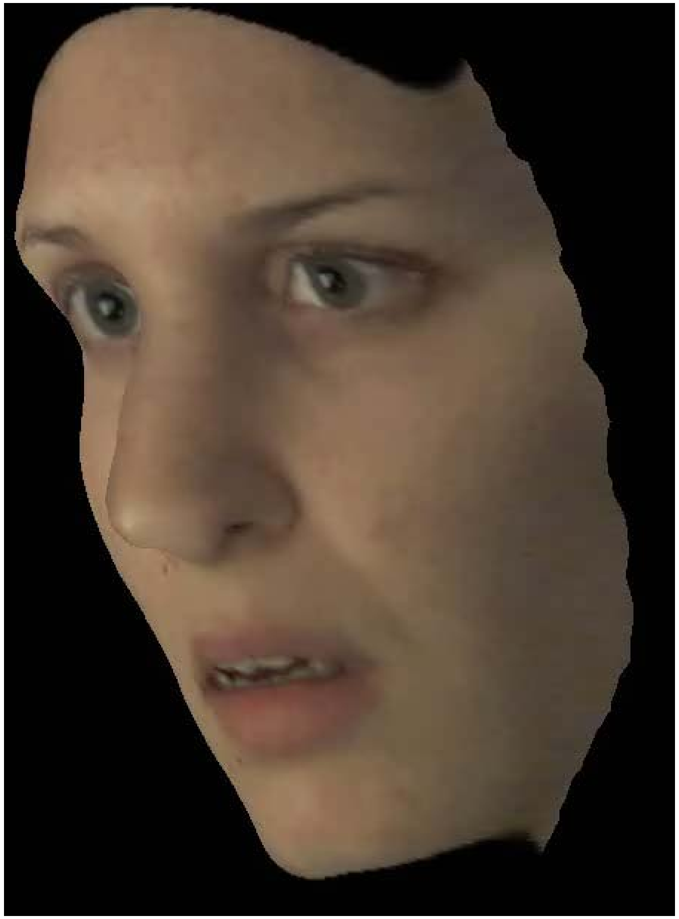
My idea

- Initial estimation of next position
  - We know only the line on which the next 3D position must lie
  - Move to the shortest position



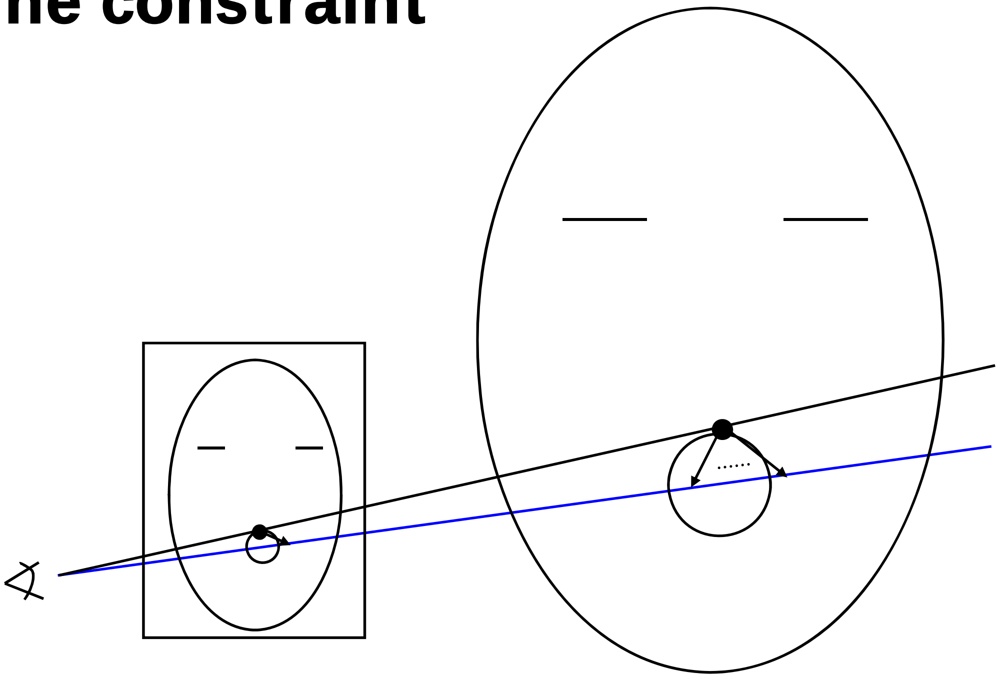
My idea

- **Initial estimation of next position**
  - **Result**



My idea

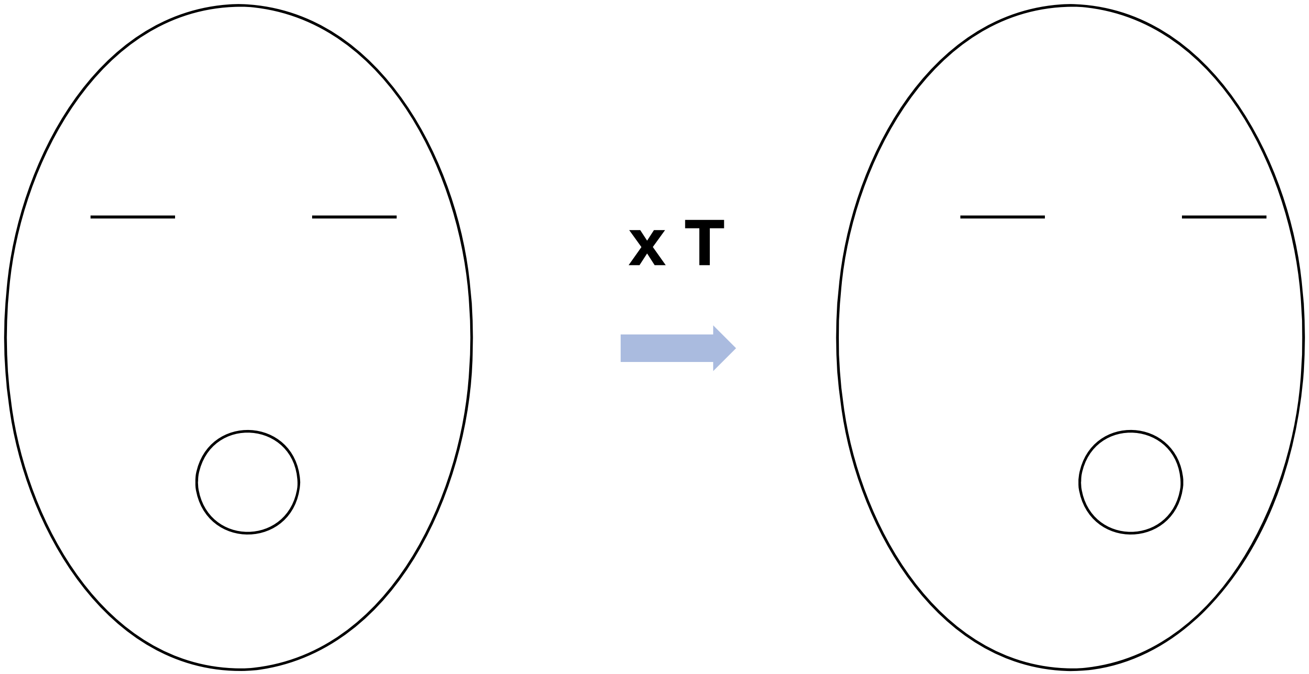
- Compute plausible mesh for the next frame from initial estimation using,
  - Physical modeling of Facial skin as a thin plate
  - Line constraint



Must move onto this line !

My idea

- Global Transformation
  - Recover the Head Motion

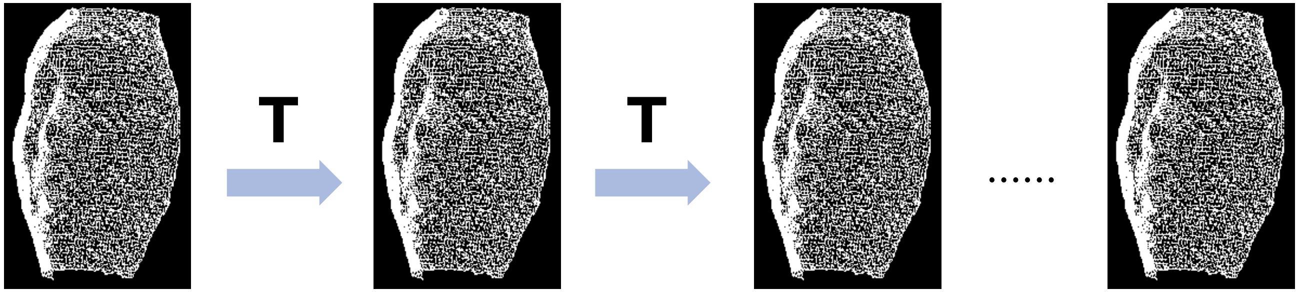


Template mesh with only local motion

Template mesh with global motion

My idea

- Output
  - Head Motion for each consecutive frames
  - Local Motion



T

T

……

Full Geometry

Local Geometry

Local Geometry

Local Geometry

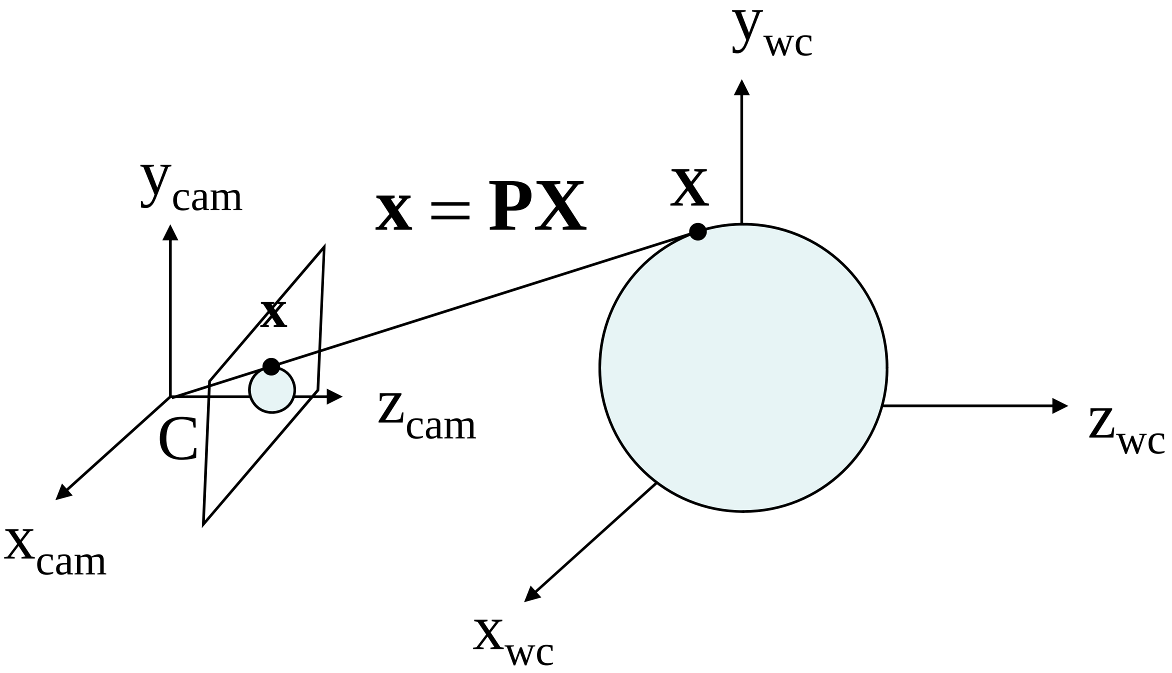
Conclusion

- New capture method
  - Use only one camera
  - Do not use stripe patterns
- Compression method
  - Only store the local movements and global transformation

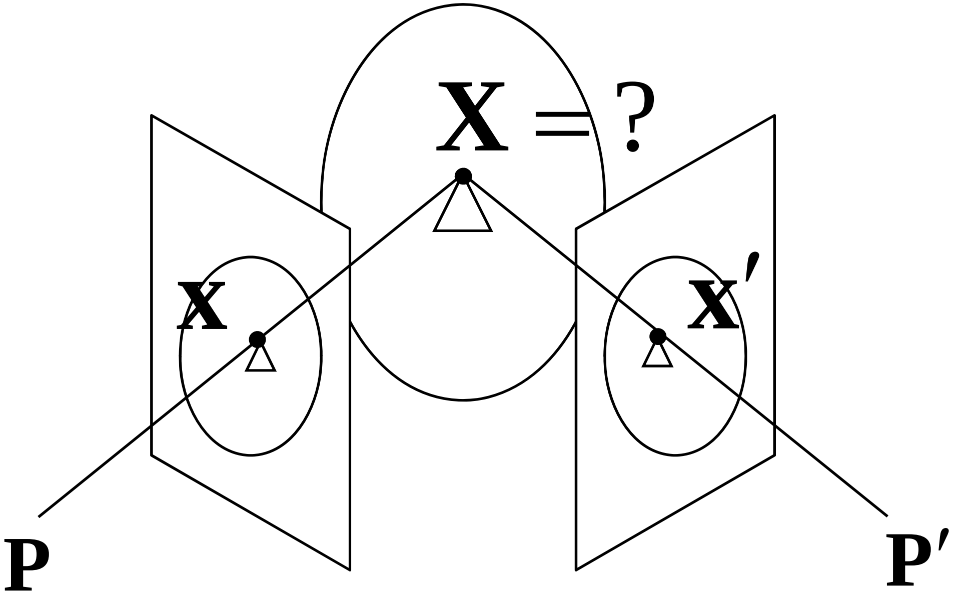
Supplement

Camera Model

- **Camera Model**
  - **Mapping between the 3D world and a 2D image**
  - **Represented by 3 x 4 matrix like,** $\mathbf{P}_{3\times 4}, \mathbf{P}'_{3\times 4}$



3D Position Recovery



$$\mathbf{x} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}, \quad \mathbf{x}' = \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix}, \quad \mathbf{X} = \begin{pmatrix} X \\ Y \\ Z \\ W \end{pmatrix}$$

$$\mathbf{P} = \begin{pmatrix} \mathbf{P}^{1\mathrm{T}} \\ \mathbf{P}^{2\mathrm{T}} \\ \mathbf{P}^{3\mathrm{T}} \end{pmatrix}, \quad \mathbf{P}' = \begin{pmatrix} \mathbf{P}'^{1\mathrm{T}} \\ \mathbf{P}'^{2\mathrm{T}} \\ \mathbf{P}'^{3\mathrm{T}} \end{pmatrix}$$

3D Position Recovery

$$\mathbf{x} = \mathbf{P}\mathbf{X},\quad \mathbf{x}' = \mathbf{P}'\mathbf{X} \quad \Rightarrow \quad \mathbf{x} \times \mathbf{P}\mathbf{X} = \mathbf{0},\quad \mathbf{x}' \times \mathbf{P}'\mathbf{X} = \mathbf{0}$$

$$\begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ x & y & 1 \\ \mathbf{P}^{1T}\mathbf{X} & \mathbf{P}^{2T}\mathbf{X} & \mathbf{P}^{3T}\mathbf{X} \end{vmatrix} = (y\mathbf{P}^{3T}\mathbf{X} - \mathbf{P}^{2T}\mathbf{X})\mathbf{i} - (x\mathbf{P}^{3T}\mathbf{X} - \mathbf{P}^{1T}\mathbf{X})\mathbf{j} + (x\mathbf{P}^{2T}\mathbf{X} - y\mathbf{P}^{1T}\mathbf{X})\mathbf{k} = \mathbf{0}$$

$$\therefore \quad y\mathbf{P}^{3T}\mathbf{X} - \mathbf{P}^{2T}\mathbf{X} = \mathbf{0}$$
$$x\mathbf{P}^{3T}\mathbf{X} - \mathbf{P}^{1T}\mathbf{X} = \mathbf{0}$$

likewise, $y\mathbf{P}'^{3T}\mathbf{X} - \mathbf{P}'^{2T}\mathbf{X} = \mathbf{0}$

$$x\mathbf{P}'^{3T}\mathbf{X} - \mathbf{P}'^{1T}\mathbf{X} = \mathbf{0}$$

$$\mathbf{A}\mathbf{X} = \mathbf{0} \qquad \textit{where} \quad \mathbf{A} = \begin{pmatrix} y\mathbf{P}^{3T} - \mathbf{P}^{2T} \\ x\mathbf{P}^{3T} - \mathbf{P}^{1T} \\ y\mathbf{P}'^{3T} - \mathbf{P}'^{2T} \\ x\mathbf{P}'^{3T} - \mathbf{P}'^{1T} \end{pmatrix}$$