SIGGRAPH2006

Performance-Driven Facial Animation

Chris Bregler
George Borshukov
Parag Havaldar
J.P.Lewis
Fred Pighin
Jim Radford
Mark Sagar
Steve Sullivan
Li Zhang

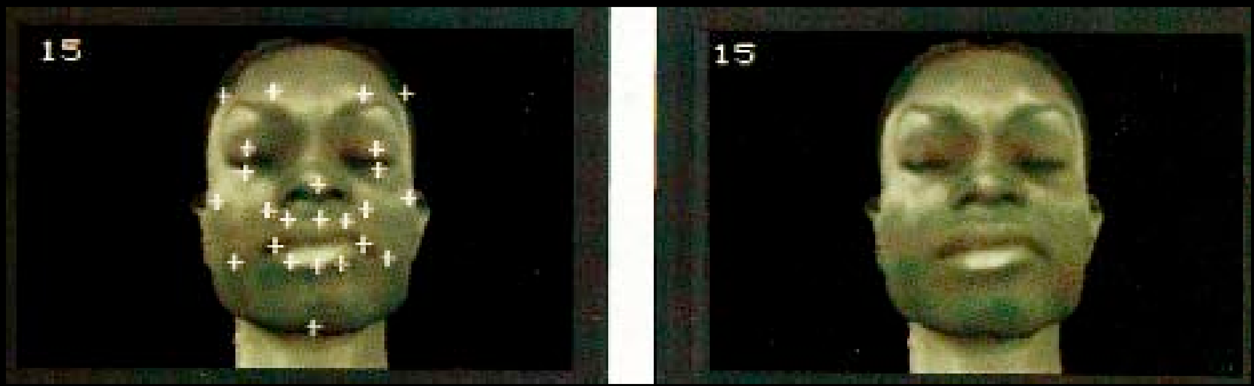
…and Thomas Kang!

Performance-Driven Facial Animation

- Use the human face itself as the “input device”
- Motivations:
  - Much easier to produce a facial expression than to adjust sliders
  - Some people believe that keyframe animation cannot consistently capture the subtleties of human motion
  - More people believe that some animators can achieve realistic motion, but that getting consistent results is time consuming and expensive.

Early History

- Lance Williams, Performance-Driven Facial Animation, SIGGRAPH 1990



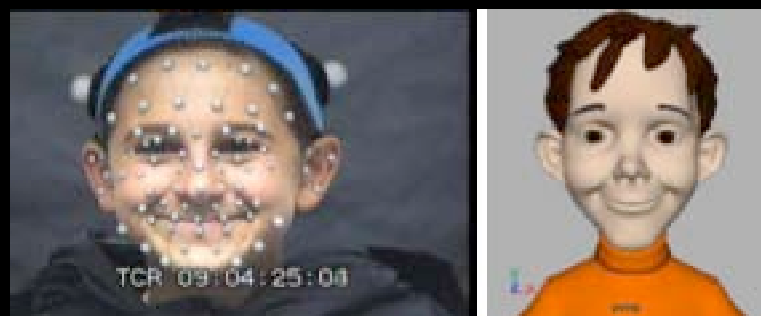
- SimGraphics “face waldo” demonstration at SIGGRAPH 1992

Face Tracking Approaches

- Photogrammetry, stereo

Face Tracking Approaches

- Marker-based hardware motion capture systems



- Tom Tolles (House of Moves) presentation 9:00 (next)
- Parag Havaldar (Sony Pictures Imageworks) presentation at 2:15 pm

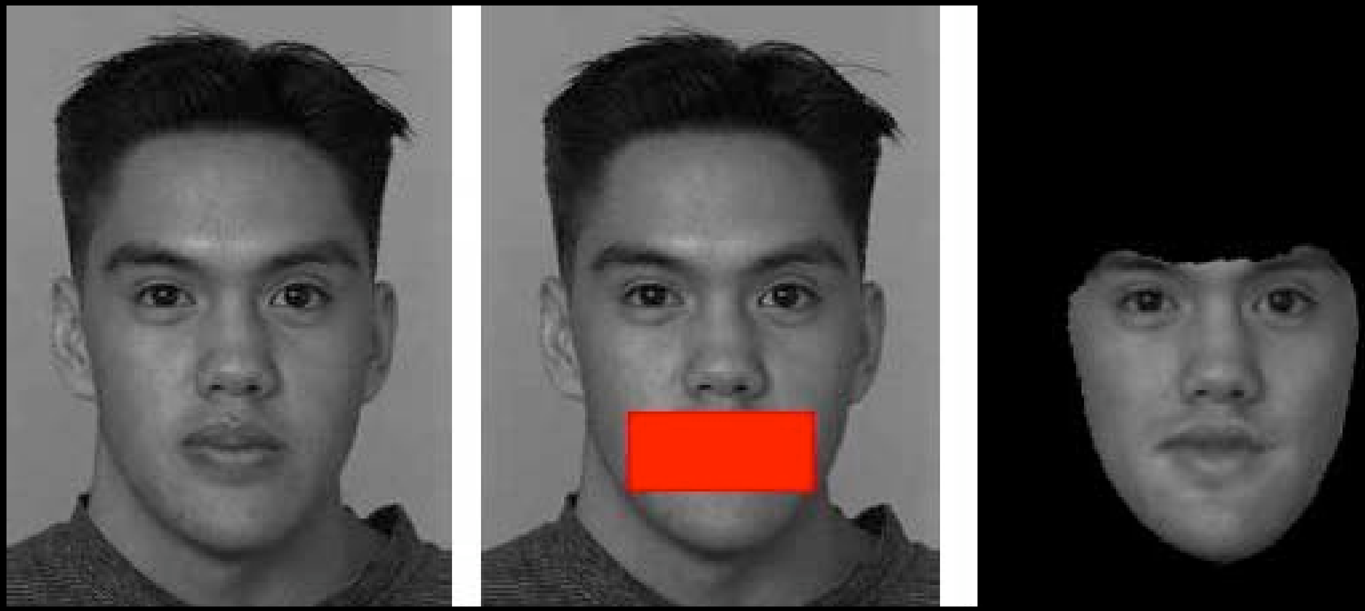
Structured Light

- Eyetronics - used in Discovery Channel's *Virtual History* (Jim Radford presentation at 4:00 pm)

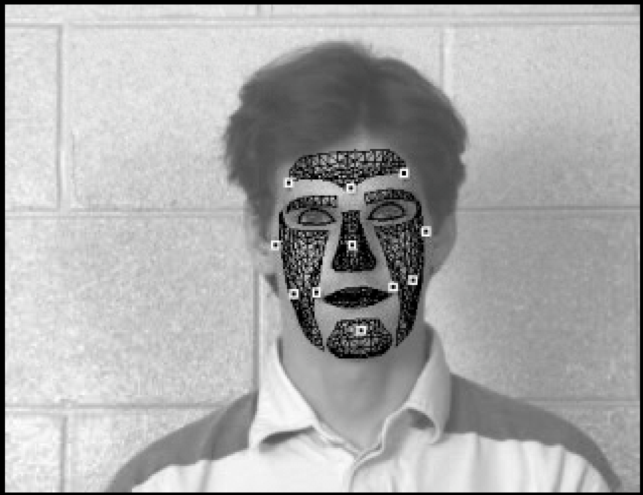
- Li Zhang - Space-time stereo
  (presentation at 11:15)

Appearance Models

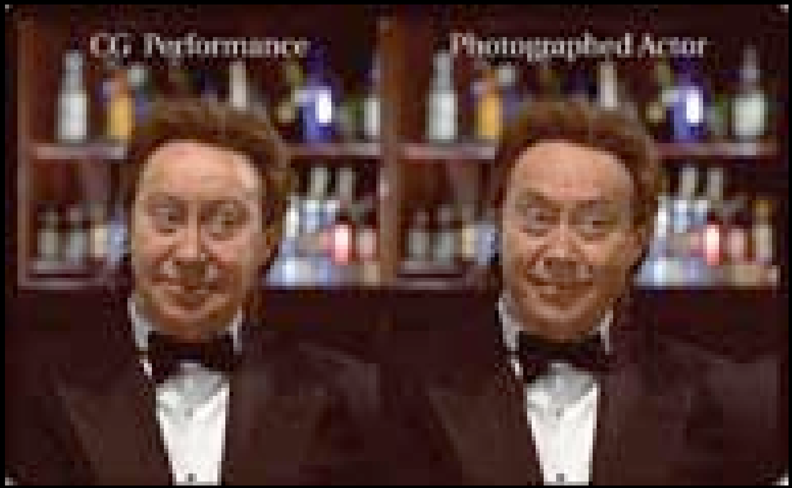
- Discussed in Chris Bregler's presentation, 11:15 am



Model-Based Tracking



DeCarlo, Metaxas, 1999

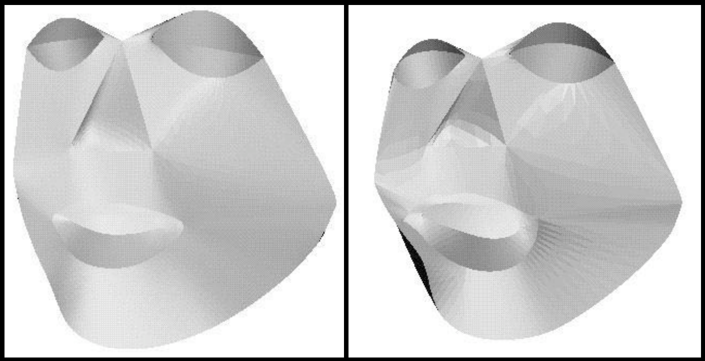


Disney Facial Test, 2002

Automatic derivation of models from video

SIGGRAPH2006

- Chris Bregler’s presentation, 11:15 am



PFDA in Entertainment

SIGGRAPH2006

PFDA in Entertainment

SIGGRAPH2006

- Movie industry tests: LifeFX, ILM's Hugo,

Did <u>not</u> use PDFA…

SIGGRAPH2006

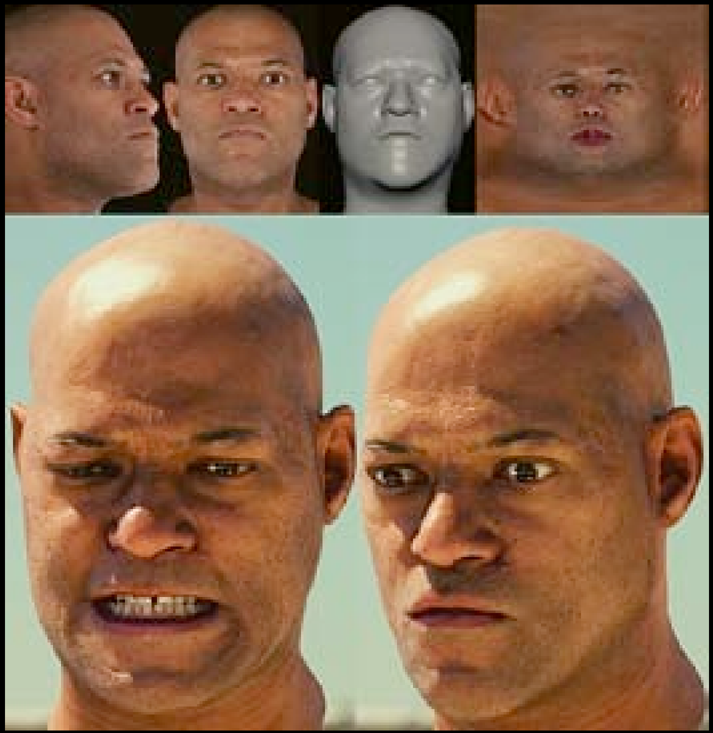
- *Final Fantasy* (2000): bodys were mocap, faces were manually animated
- *Lord of the Rings*: Gollum was manually animated, but based heavily on video reference -- “roto PFDA”?

… see Mark Sagar’s presentation on PFDA in King Kong at 3:00pm

The Matrix Reloaded (2003)

SIGGRAPH2006

- Further development this “Ucap” approach at Electronic Arts: see George Borshukov’s presentation at 4:45



The Polar Express (2004)

- First project where PFDA used exclusively for main characters

- See Parag Havaldar's presentation on PFDA at Sony (*Monster House*, …) at 2:15 pm

Schedule

SIGGRAPH2006

8:30	Introduction and Overview Fred Pighin, Industrial Light + Magic
9:00	Facial Motion Capture in Production Parag Havaldar, Sony, and Tom Tolles, House of Moves
10:00	Break
10:15	Facial Retargeting J.P. Lewis, Stanford, and Fred Pighin, ILM
11:15	Markerless Face Capture and Automatic Model Construction Chris Bregler, NYU, and Li Zhang, Columbia
12:15	Lunch
1:30	Performance Driven Facial Animation at ILM Steve Sullivan and Christophe Hery, ILM
2:15	Monster House Parag Havaldar, Sony
3:00	King Kong Mark Sagar, Weta
4:00	Virtual History - Jim Radford, Moving Picture Company, Face Robot - Thomas Kang, Softimage
4:45	Playable Universal Capture at Electronic Arts George Borshukov, EA
5:15	Panel on the future of performance-driven animation all speakers

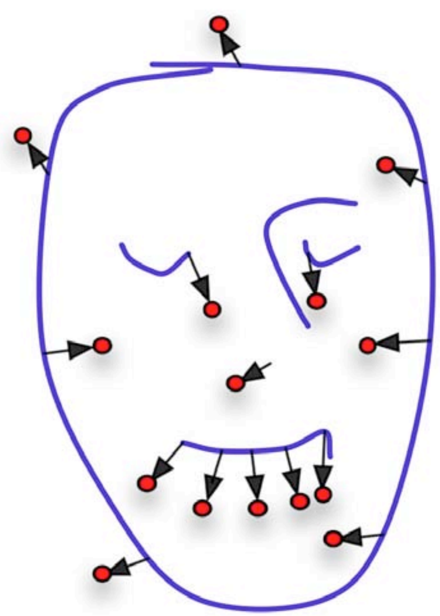
Performance-driven facial animation: background mathematics

J.P. Lewis and Fred Pighin

Background

Scattered Data Interpolation

- Interpolate data at arbitrary (irregularly spaced) locations
- Standard application: warp generic face mesh to fit mocap markers

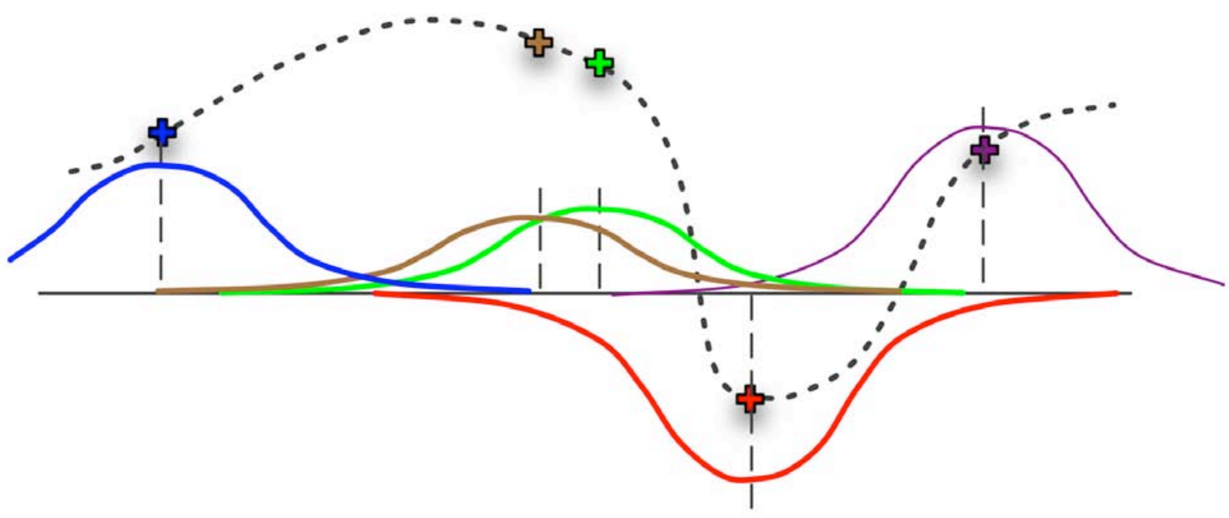


Background

Radial Basis Functions

- Data at arbitrary (irregularly spaced) locations can be interpolated with a weighted sum of radial functions situated at each data point.

$$\hat{d}(\mathbf{x}) = \sum_{k}^{N} w_k R(\|\mathbf{x} - \mathbf{x}_k\|)$$



Background

Radial Basis Functions

- *Any* function other than constant can be used!
- Common choices:
  - Gaussian: $R(r) = \exp(-r^2/\sigma^2)$
  - Thin plate spline: $R(r) = r^2 \log r$ (in 2D)
  - Hardy Multiquadratic: $R(r) = \sqrt{(r^2 + c^2)}, c > 0$
- Notice: the last two *increase* as a function of radius

Warping Facial Geometry

$$d(\mathbf{x}) = \sum_{k}^{N} w_k R(\|\mathbf{x} - \mathbf{x}_k\|)$$

For warping facial geometry:

d(x) is the x,y, or z displacement between a point and where it should go, e.g. between a model point and the corresponding mocap marker location

Background

Radial Basis Functions

- Given data points $d_k(x_k)$ to interpolate, solving for the weights is just a matrix inverse:

$$d(\mathbf{x}) = \sum_{k}^{N} w_k R(\|\mathbf{x} - \mathbf{x}_k\|)$$

$$e = ||(\mathbf{d} - \mathbf{R}\mathbf{w})||^2$$

$$e = (\mathbf{d} - \mathbf{R}\mathbf{w})^T(\mathbf{d} - \mathbf{R}\mathbf{w})$$

$$\frac{de}{d\mathbf{w}} = 0 = -\mathbf{R}^T(\mathbf{d} - \mathbf{R}\mathbf{w})$$

$$\mathbf{R}^T\mathbf{d} = \mathbf{R}^T\mathbf{R}\mathbf{w}$$

$$\mathbf{w} = (\mathbf{R}^T\mathbf{R})^{-1}\mathbf{R}^T\mathbf{d} = \mathbf{R}^{-1}\mathbf{d}$$

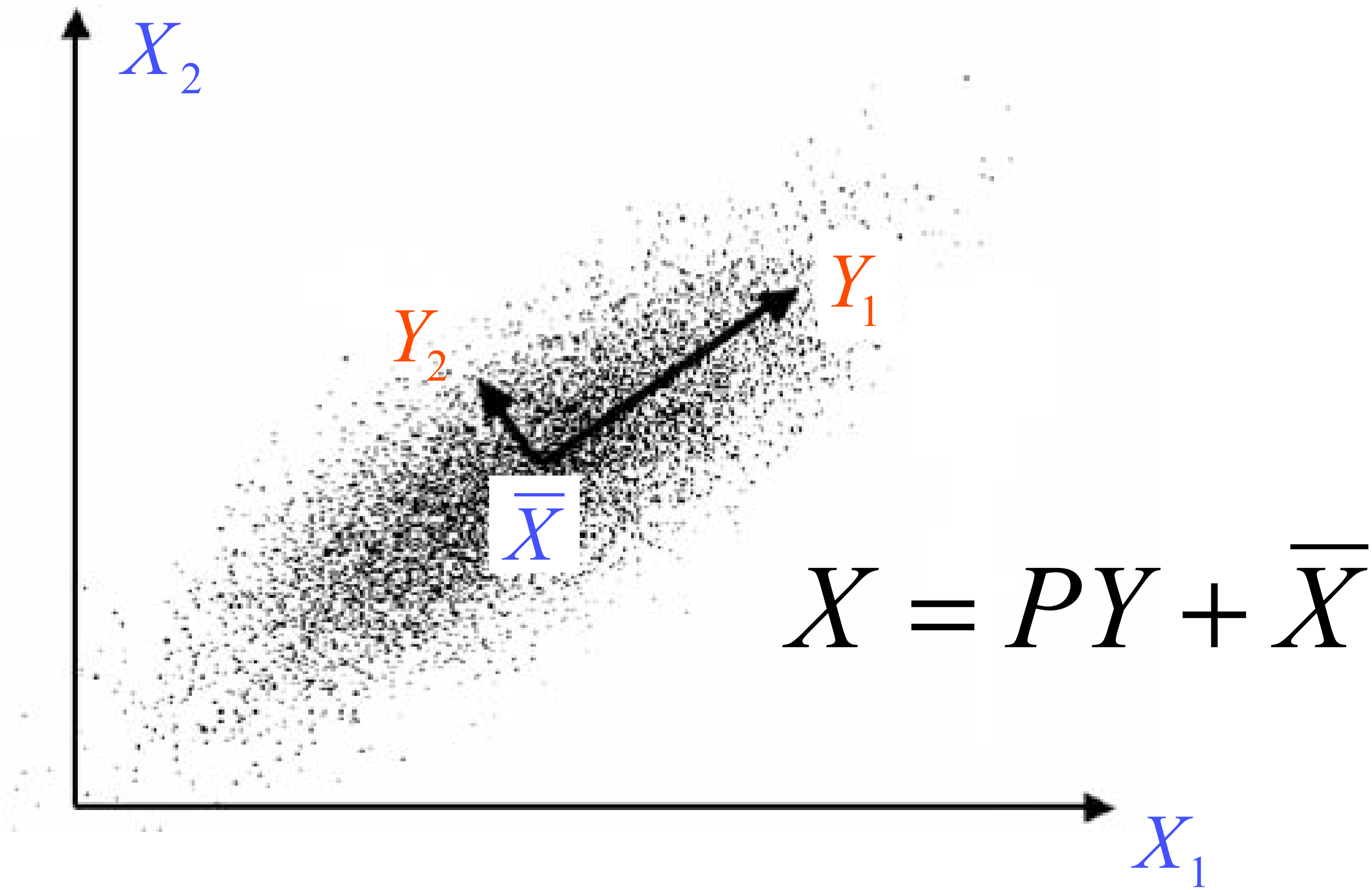
Background

Principal Component Analysis

Principal Component Analyis (PCA):

- Gives a best cartesian coordinate system for your data
- Is a method for dimensionality reduction
- Automatically makes a linear (blendshape-like) model of data

Example



Mathematical Formulation

1. Data centering $X' = X - \overline{X}$
2. Computing new basis

$$\text{Covariance matrix: } Cov = [c_{ij}]$$

$$\text{where } c_{ij} = \sum_{\text{all samples}} x'_i x'_j$$

The eigenvectors of *Cov* are the vector of the PCA basis $\{Y\}$

Limitations of PCA

- Linear model (linear meaning the subspace consists of a line, or a plane, or …)
- Not ideal if data is not Gaussian distributed

Manifold Learning

- Generalize away from the linear (line, plane, …subspace) restriction of PCA
- “Manifold”: a subspace with continuity and smoothness properties. Essentially, *a surface in a higher dimensional space.*

Manifold Learning

Dimensional Thinking:

- A "dimension" is anything that you can independently change
- 3D space: every point describe by 3 numbers, (x,y,z).
- Image space: 1000x1000 "megapixel" image has one million pixels. A particular image is described by one million numbers   (ignore RGB, say a greyscale image)

→ An image is a point in a million-dimensional space.

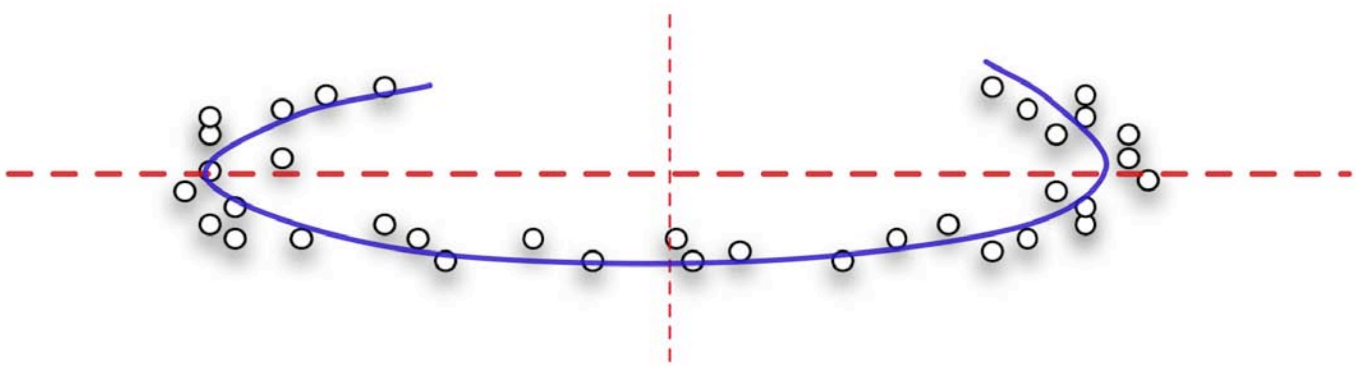
Manifold Learning

- (recall) An image is a point in a (e.g.) million-dimensional space
- Animate an image slightly and the “point” moves.
- Image sequence of a moving head (for example): corresponds to movement along a one-dimensional curve in the million dimensional space.
- Manifold learning: discover this curve
- See Expression/Style mapping section of Cross-Mapping session this morning

Manifold Learning

Red dashed: principal component analysis,

Blue solid: manifold learning



Manifold Learning

Multidimensional Scaling (MDS): a linear predecessor

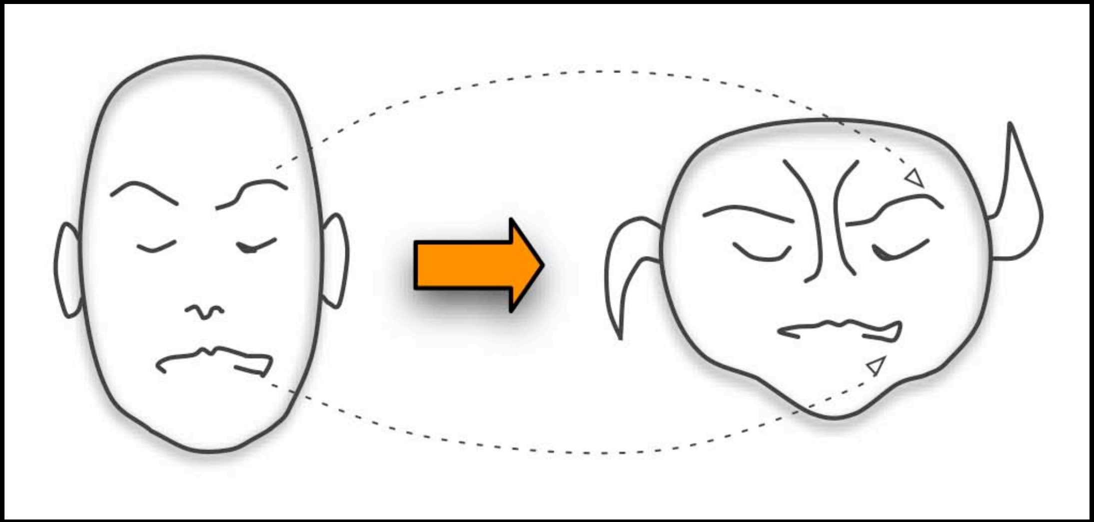
- Algorithms:
  - Local Linear Embedding (LLE),
  - Isomap (non-linear version of MDS)

  } The two original algorithms

  - Laplacian Eigenmaps: related to LLE, clarified and simpler
  - Hessian Eigenmaps
  - Kernel PCA
  - Gaussian Process latent variable model …

- General approach: find a placement of points in a low-dimensional space (the manifold) such that the distance between points is proportional to the distance between the original points in the high dimensional space.

Retargeting



Algorithms for Performance-Driven Animation

J.P. Lewis Fred Pighin

"Don't cross the streams." (Ghostbusters)

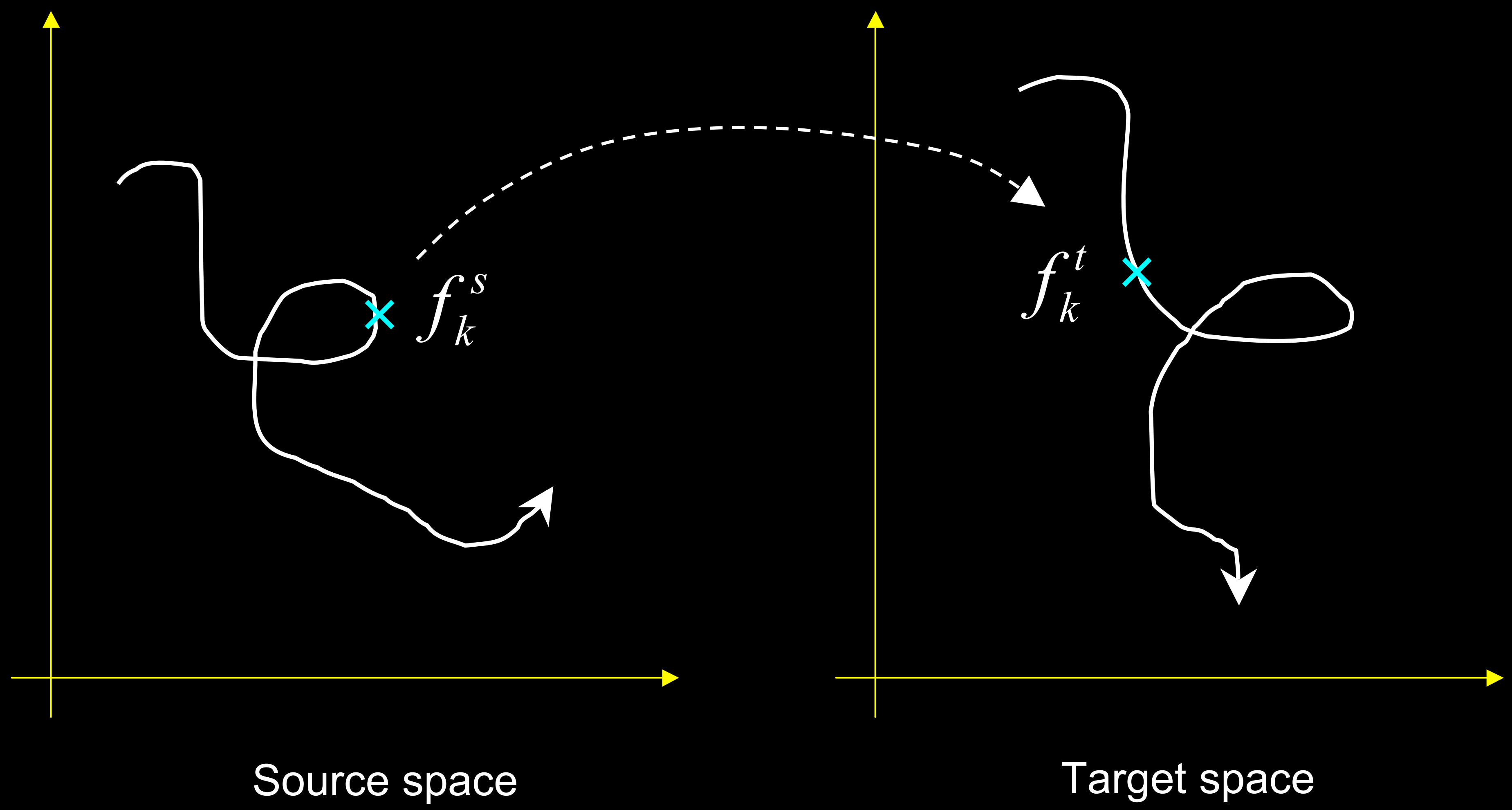
- Why cross-mapping?
  - Different character
  - Imperfect source model
- Also known as:
  - Performance-driven animation
  - Motion retargeting

Performance Cloning History

SIGGRAPH2006

- L. Williams, Performance-driven Facial Animation, *SIGGRAPH* 1990
- SimGraphics systems, 1992-present
- LifeFX “Young at Heart” in Siggraph 2000 theater
- J.-Y. Noh and U. Neumann, Expression Cloning, *SIGGRAPH* 2001
- B. Choe and H. Ko, “Muscle Actuation Basis”, *Computer Animation* 2001 (used in Korean TV series)
- Wang et. al., *EUROGRAPHICS* 2003
- *Polar Express* movie, 2004

Face space mapping

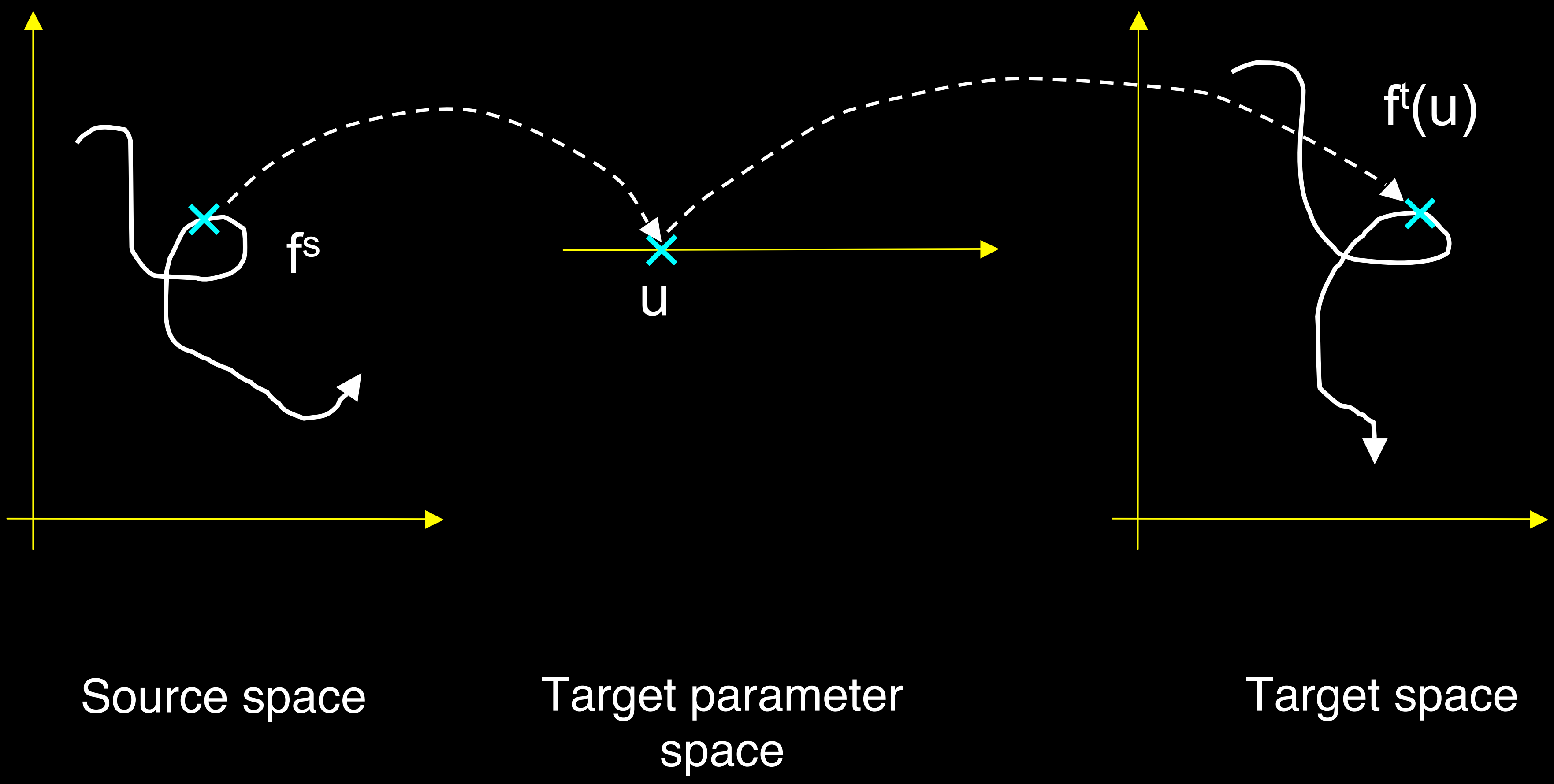


Parameterizing the target space: a rig

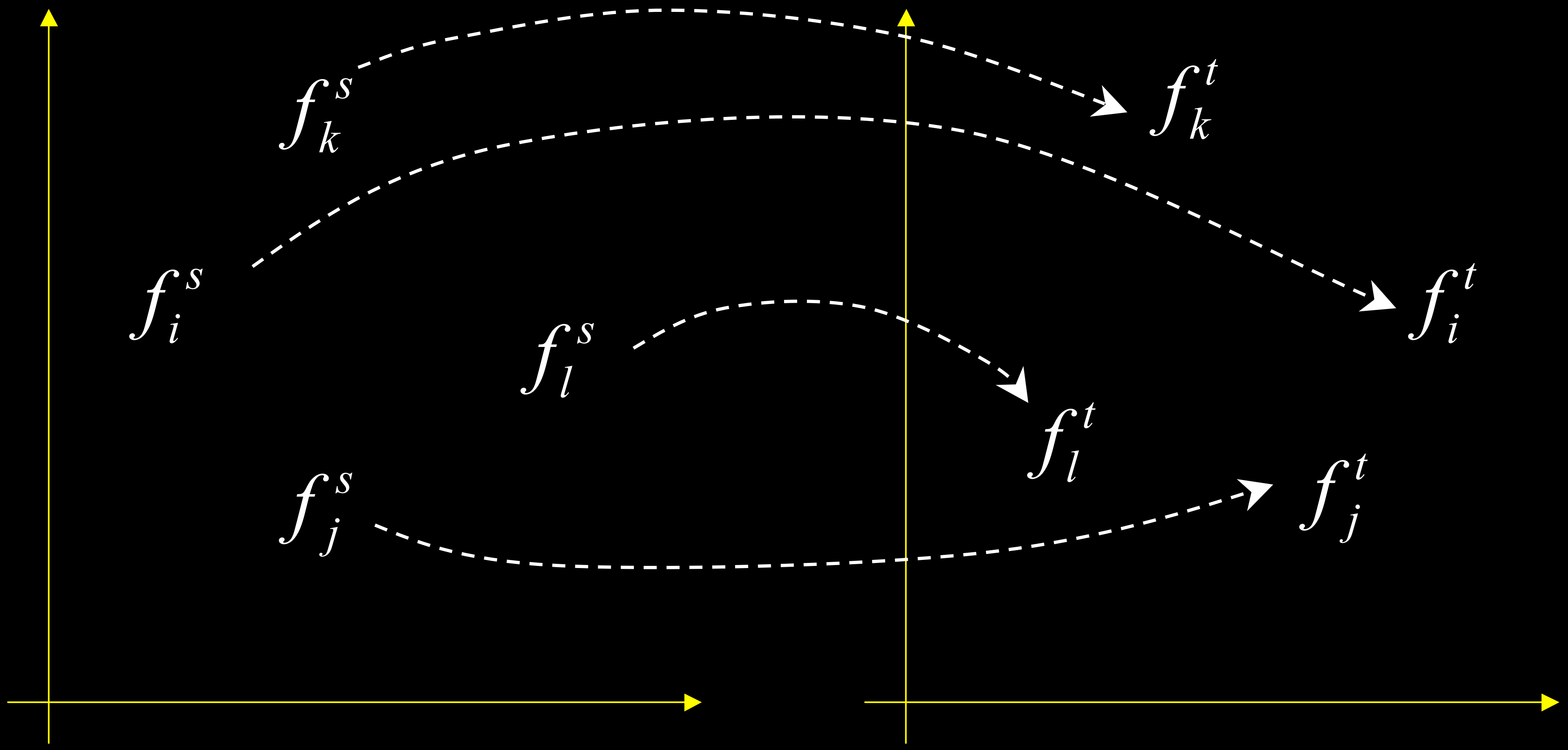
SIGGRAPH2006

- A facial rig defines a set of parameters/controllers for the face
- Interpolation in parameter space generates “valid” expressions

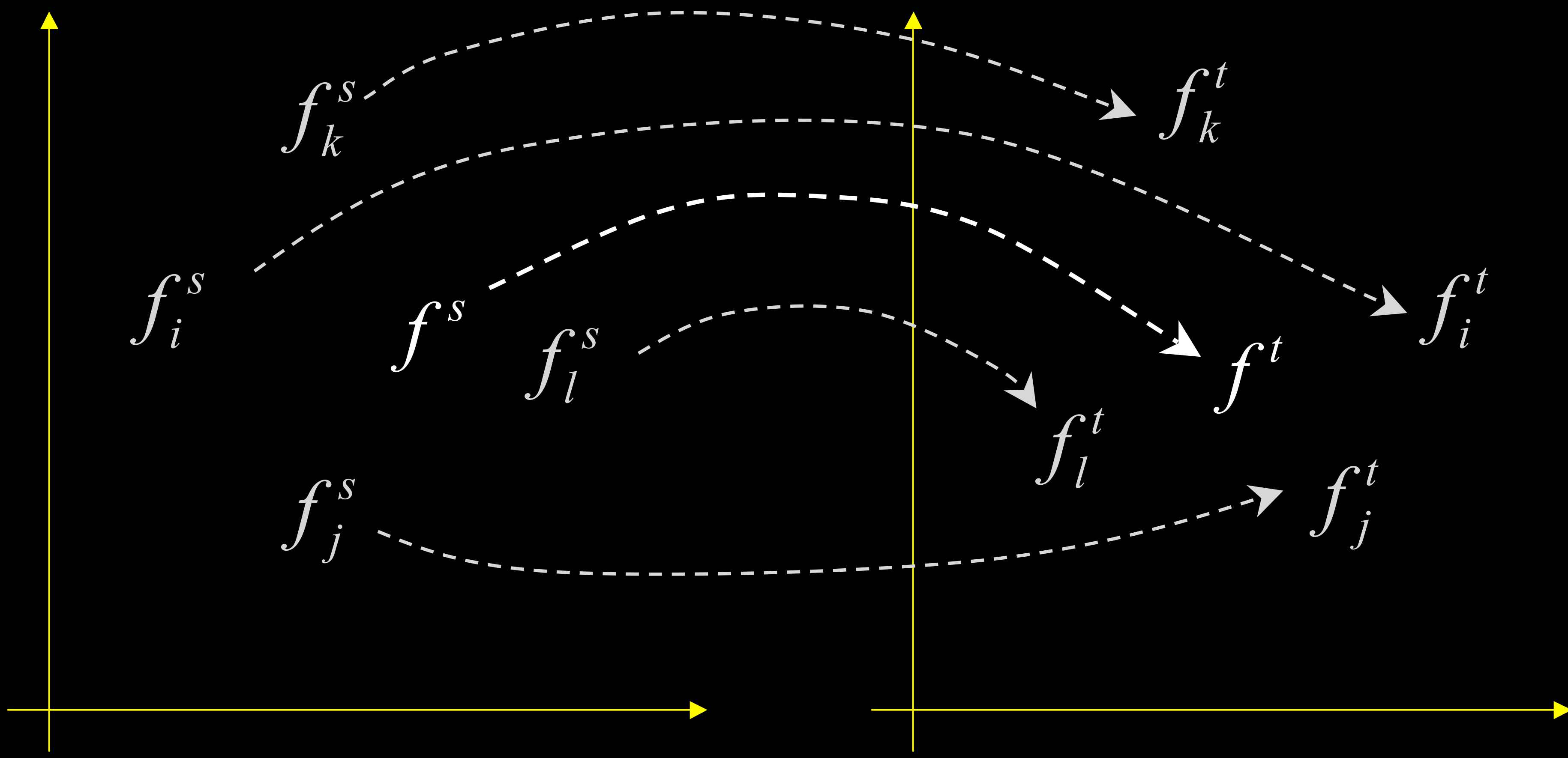
Face space mapping with rig



Building a map from correspondences



Building a map from correspondences



$$f^t = M\left(f^s \mid (f_i^s, f_i^t), i \in \{1..N\}\right)$$

Main issues

- How are the corresponding faces created?
- How to build mapping from correspondences?

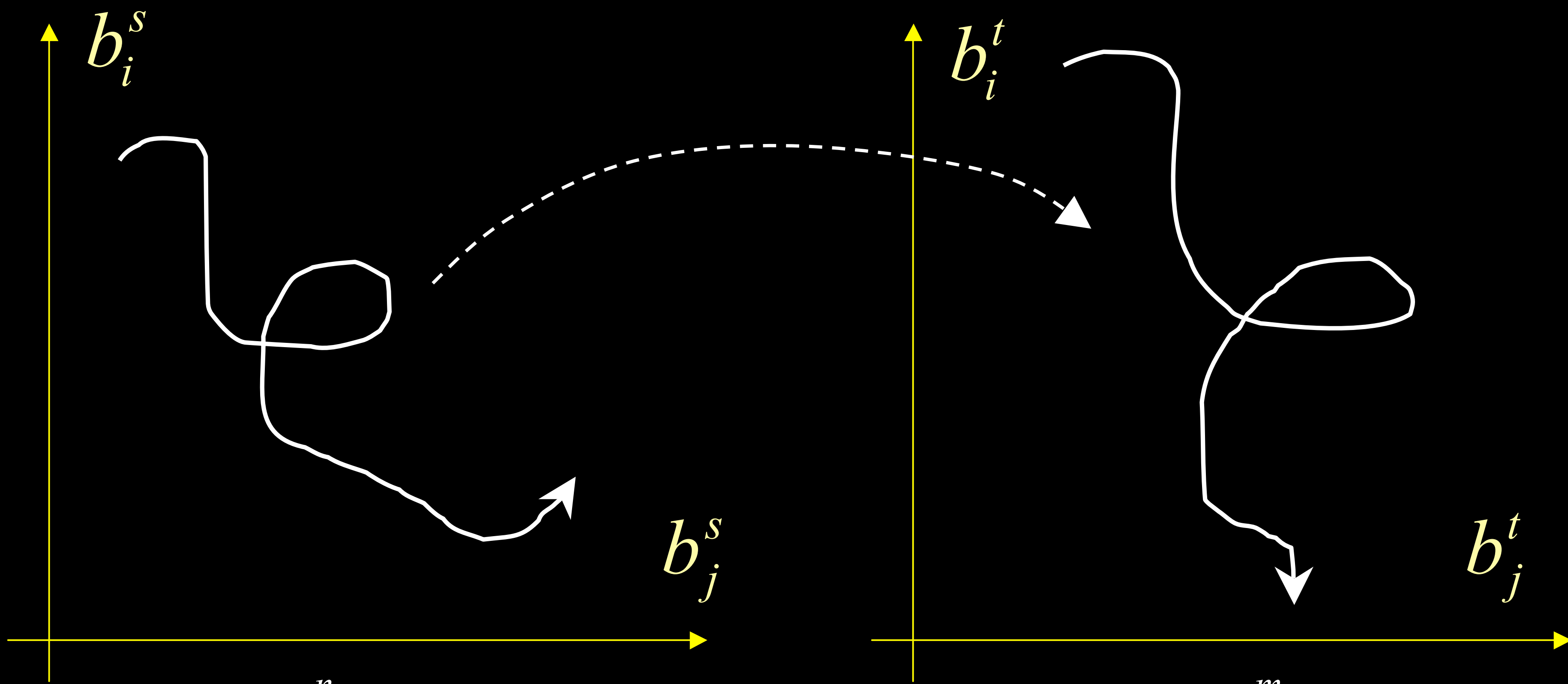
Linear vs. non-linear mapping

SIGGRAPH2006

- Linear
  - **Global blend-shape mapping**
    - B. Choe and H. Ko, **Analysis and Synthesis of Facial Expressions with Hand-Generated Muscle Actuation Basis**, *Computer Animation* 2001.
    - E. Chuang and C. Bregler, **Performance Driven Facial Animation using Blendshape Interpolation**, CS-TR-2002-02, Department of Computer Science, Stanford University
- Non-linear
  - **Piece-wise blend-shape mapping**
    - I. Buck, A. Finkelstein, C. Jacobs, A. Klein, D. H. Salesin, J. Seims, R. Szeliski, and K. Toyama, **Performance-driven hand-drawn animation**, *NPAR 2000.*
    - J.-Y. Noh and U. Neumann, **Expression Cloning**, *SIGGRAPH* 2001.
  - **Manifold learning**
    - Y. Wang, X. Huang, C.-S. Lee, S. Zhang, D. Samaras, D. Metaxas, A. Elgammal, and P. Huang, **High Resolution Acquisition, Learning, and Transfer of Dynamic 3-D Facial Expressions**, *Eurographics* 2004.
  - **Single-correspondence mapping**
    - J.-Y. Noh and U. Neumann, **Expression Cloning**, *SIGGRAPH* 2001.

Linear Mapping

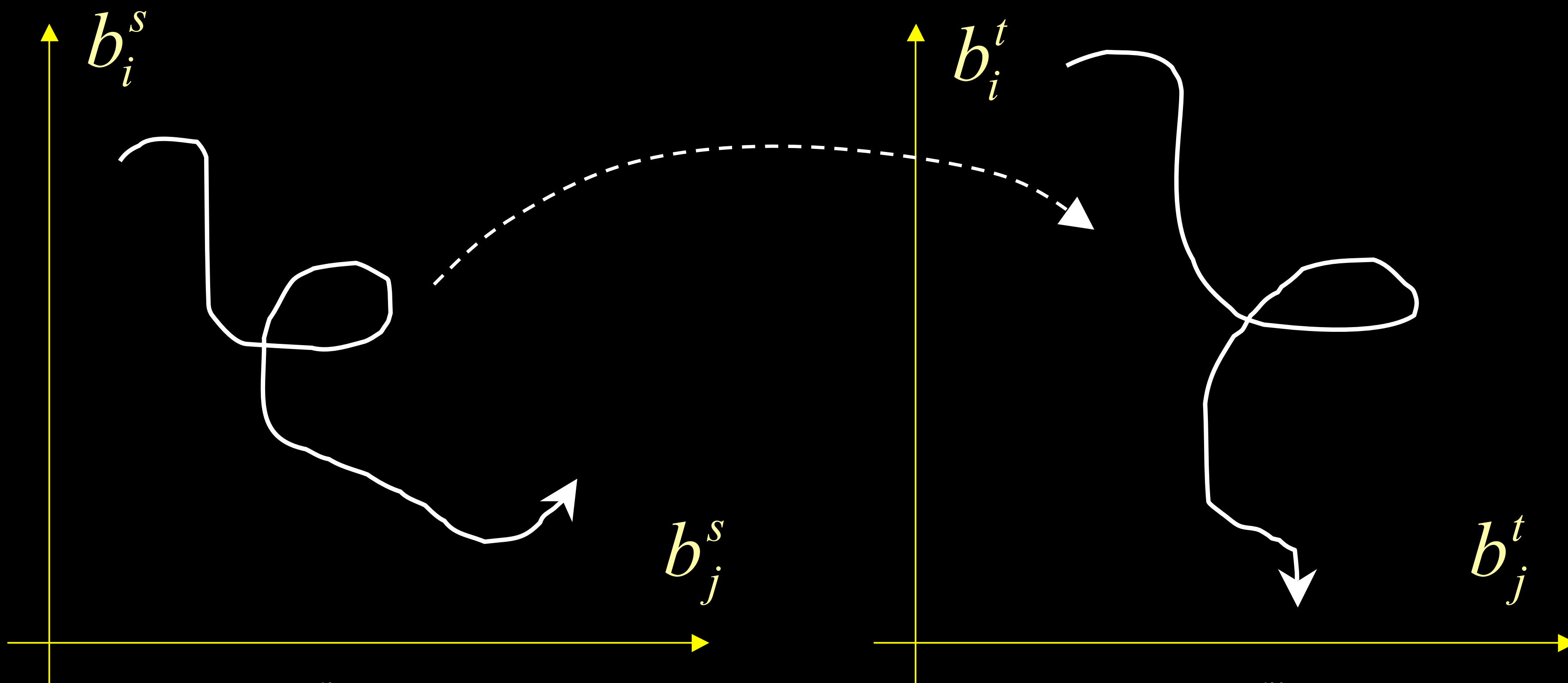
Blendshape mapping



$$f_k^s = \sum_{i=1}^{n} w_{k,i}^s b_i^s \longrightarrow f_k^t = \sum_{i=1}^{m} w_{k,i}^t b_i^t$$

(Global) Change of coordinate system

Blendshape mapping



$$f_k^s = \sum_{i=1}^{n} w_{k,i}^s b_i^s \longrightarrow f_k^t = \sum_{i=1}^{m} w_{k,i}^t b_i^t$$

(Global) Change of coordinate system

Blendshapes: definition

SIGGRAPH2006

- per-vertex view: $\vec{\mathbf{f}}_k = \sum w_k \vec{\mathbf{b}}_k$

	Industry term	Math usage
$\mathbf{w}_k$	Slider values	weights
b_k	Blendshape target	Basis vector

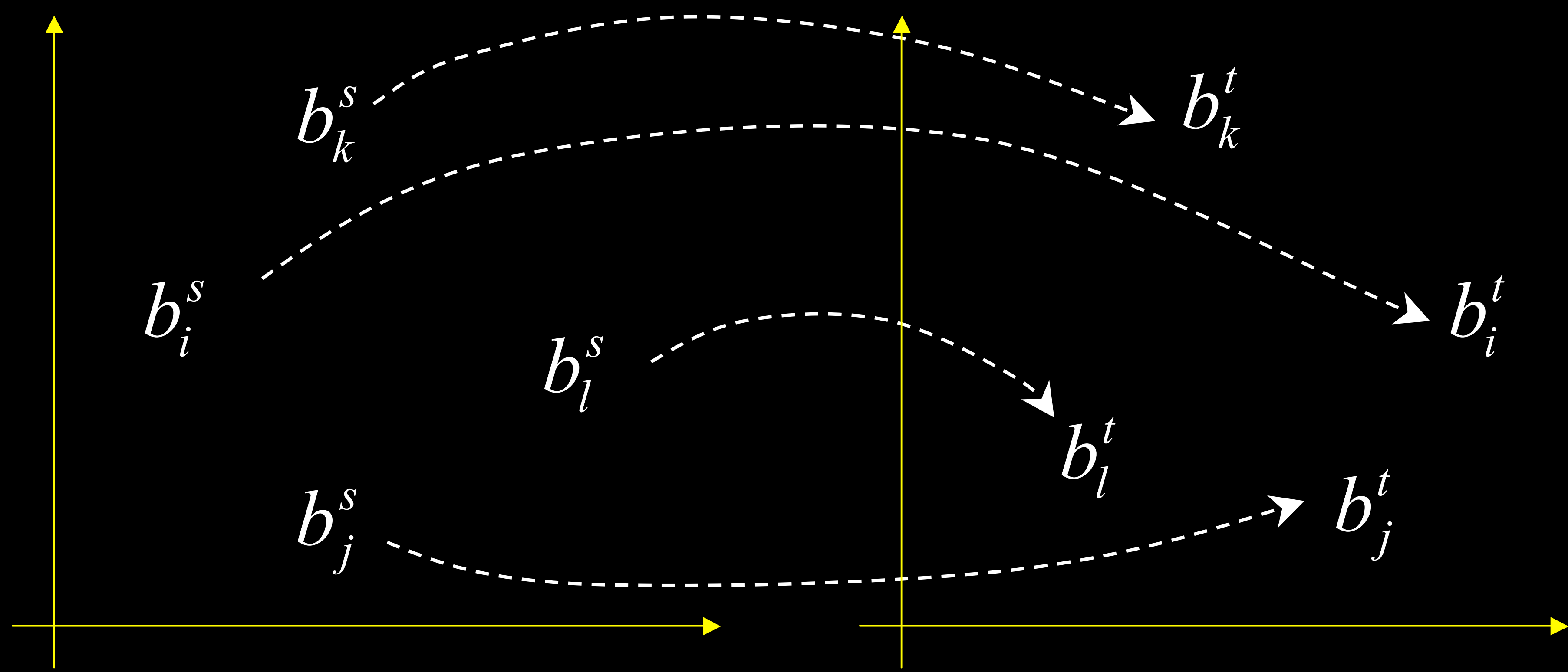
Blendshapes: definition

SIGGRAPH2006

- Linear algebra of blendshapes:
- per-vertex view: $\vec{\mathbf{f}}_k = \sum w_k \vec{\mathbf{b}}_k$
- global view: $\mathbf{f} = \mathbf{B}\mathbf{w}$
- **f** : 3*n* x 1 vector containing all *n* vertices of the face, in some packing order e.g. *xyzxyzxyz....*
- **B**: 3*n* x *m* matrix; each column is one of the *m* blendshapes, using the same packing order.
- **w**: vector of *m* weights, animated over time

Parallel blendshapes

SIGGRAPH2006



Use same blending weights: $w_{k,i}^t = w_{k,i}^s$

Parallel Model Construction

SIGGRAPH2006

- Have similar blendshape controls in source, target models
- Advantage: conceptually simple
- Disadvantage: twice the work (or more!) -- unnecessary!
- Disadvantage: cannot use PCA

Solutions

SIGGRAPH2006

- Adapt generic model to source (Choe et. al.)
- Derive source basis from data (Chuang and Bregler)
- Allow different source, target basis

Source model adaptation

SIGGRAPH2006

- B. Choe and H. Ko, **Analysis and Synthesis of Facial Expressions with Hand-Generated Muscle Actuation Basis**, *Computer Animation* 2001
- Cross-mapping obtained simply by constructing two models with identical controls.
- Localized (delta) blendshape basis inspired by human muscles
- Face performance obtained from motion captured markers

Choe and Ko

Muscle actuation basis

SIGGRAPH2006

- Model points corresponding to markers are identified
- Blendshape weights determined by least-squares fit of model points to markers
- Fit of model face to captured motion is improved with an alternating least squares procedure

Choe and Ko

Muscle actuation basis

SIGGRAPH2006

- Fitting the model to the markers:
- alternate 1), 2)
  – 1) solve for weights given markers and corresponding target points
  – 2) solve for target points location
- warp the model geometry to fit the final model points using radial basis interpolation.

Choe and Ko

Muscle actuation basis

- Fitting the model to the markers:

$$f_k = Bw_k \quad \text{for all frames k}$$

$$F = BW \quad \text{stack all f,w in matrices}$$

- Alternate: solve for B, solve for W
- warp the model geometry to fit the final model points using radial basis interpolation.

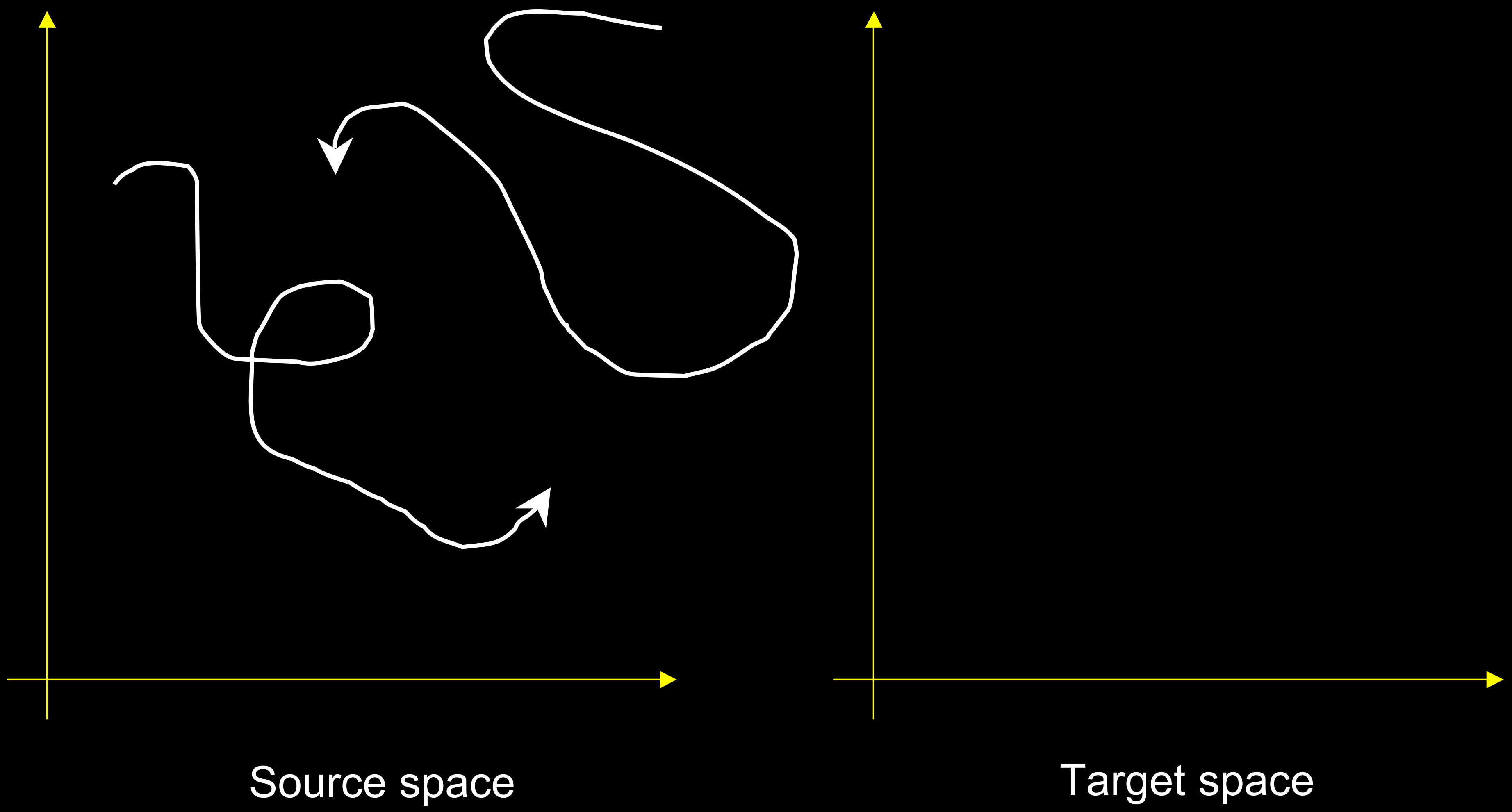
Choe & Ko

SIGGRAPH2006

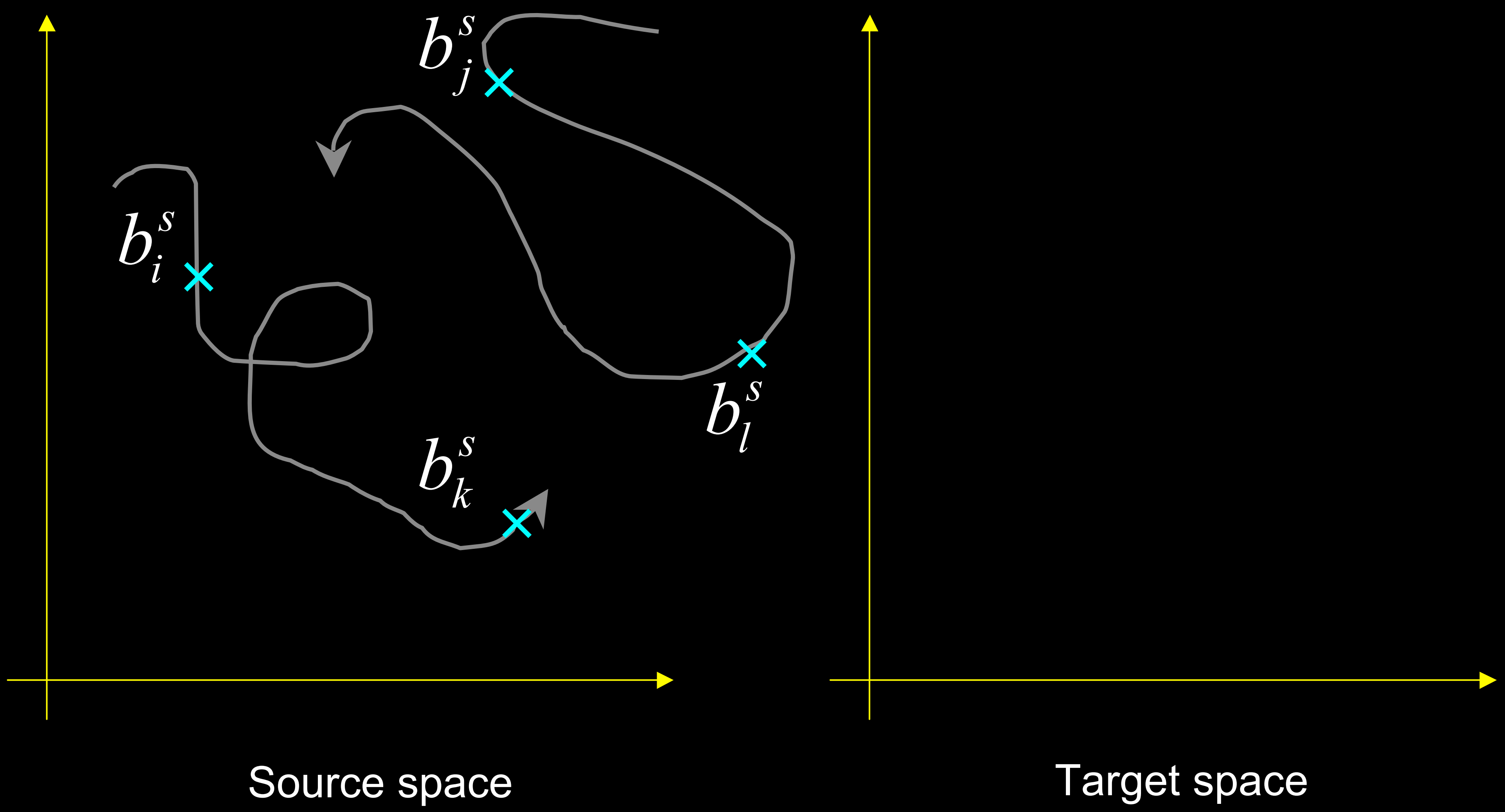
Derive source blendshapes from data

- Principal Component Analysis
- [Chuang and Bregler, 2002]

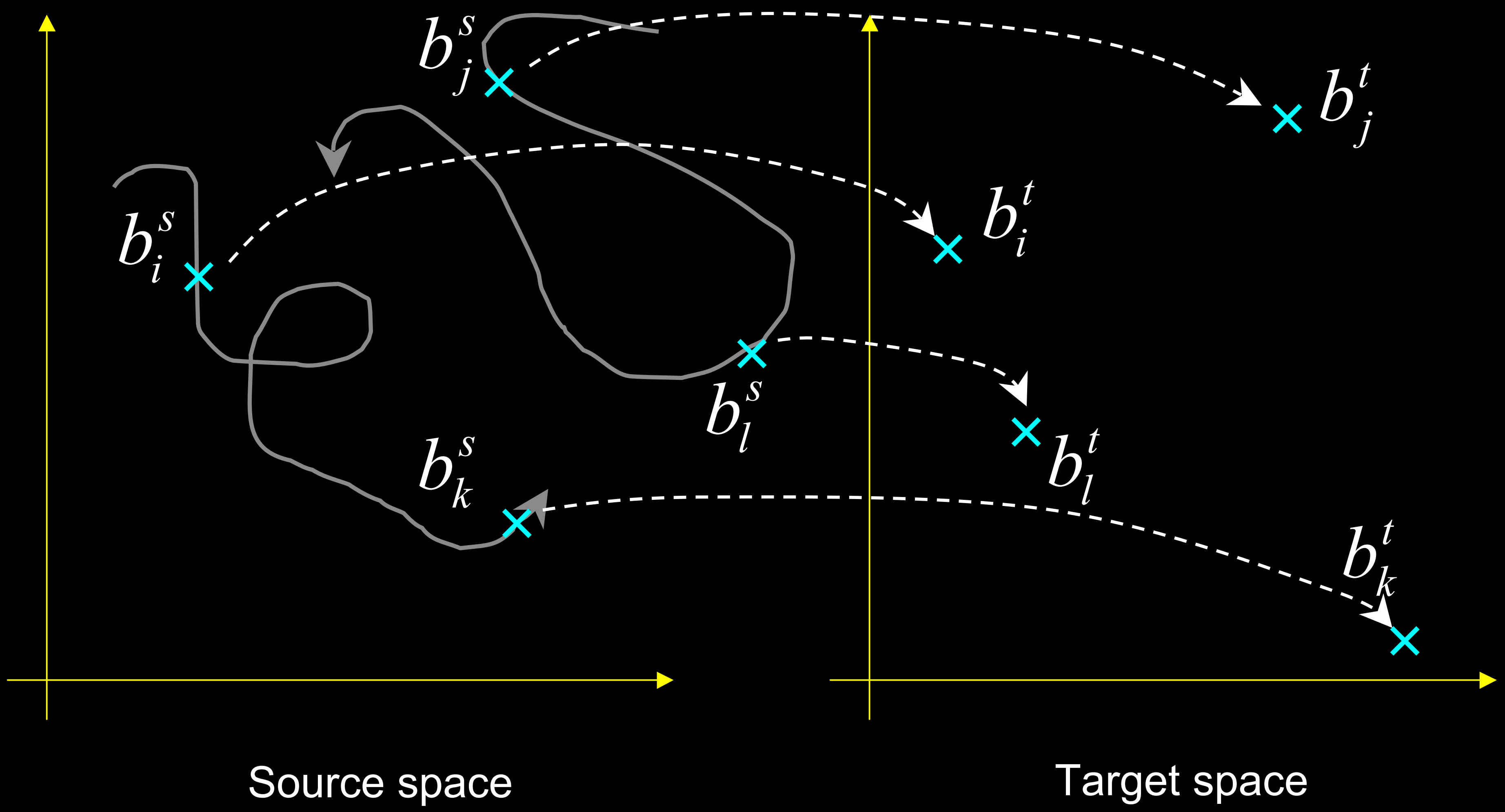
Recorded source motions



Source basis estimation



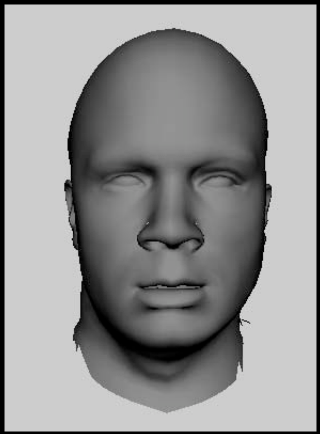
Target basis construction

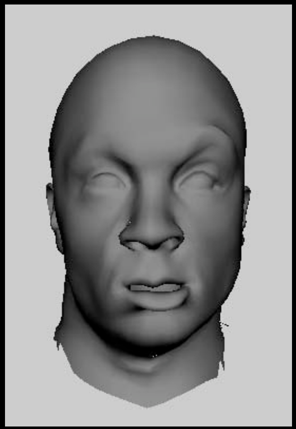


Blendshapes by Principal Component Analysis (PCA)

SIGGRAPH2006

- Automatic construction of blendshape model (given movement data)
- Advantage: automatic; the most accurate model for a given number of sliders (L2 sense), easy
- Disadvantage: the resulting model is not intuitive





Derive source basis from data

SIGGRAPH2006

- E. Chuang and C. Bregler, **Performance Driven Facial Animation using Blendshape Interpolation**, CS-TR-2002-02, Department of Computer Science, Stanford University

Chuang and Bregler,

Derive source basis from data

- Parallel Model Construction approach:
  - Source model automatically derived,
  - Target manually sculpted
- Using PCA would be unpleasant

Chuang and Bregler,

Derive source basis from data

- Using PCA would be unpleasant
- Solution: use a subset of the motion capture frames as the blendshape model.
  - Subsets of the original motion capture start to “span the space” of that motion capture.
- Two new problems:
  1) Which motion capture frames to use?
  2) Source blendshape basis is not exact.

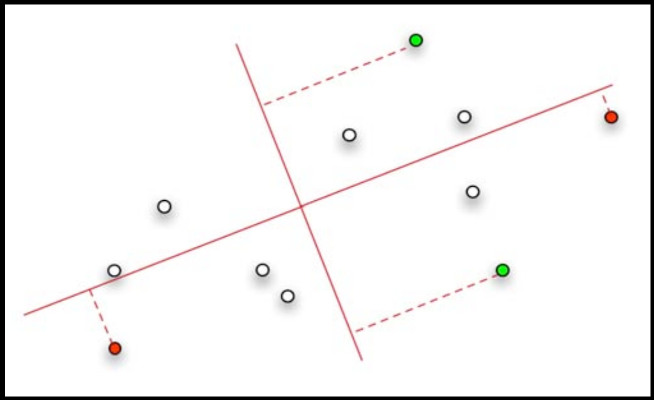
Chuang and Bregler,

Derive source blendshapes from data

Which motion capture frames to use?

Heuristic: for each of the leading PCA vectors,

Pick the mocap frame that have the largest (min,max) projections on that eigenvector.



Chuang and Bregler,

Derive source blendshapes from data

SIGGRAPH2006

Two new problems:

1) Which motion capture frames to use?
2) Source blendshape basis is only approximate

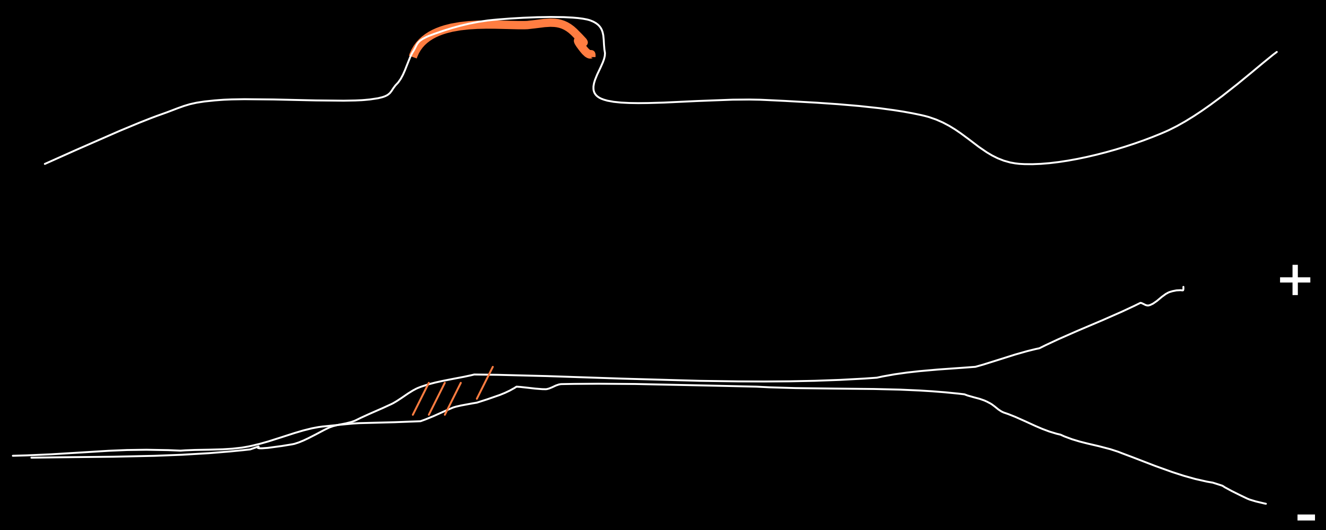
Observation:

- Directly reusing weights works poorly when the source model is not exact
  - Errors in representing the source can result in large cancelling basis combinations (nearly cancelling positive, negative weights)
  - Transferring these cancelling weights to target results in poor shapes

Chuang and Bregler,

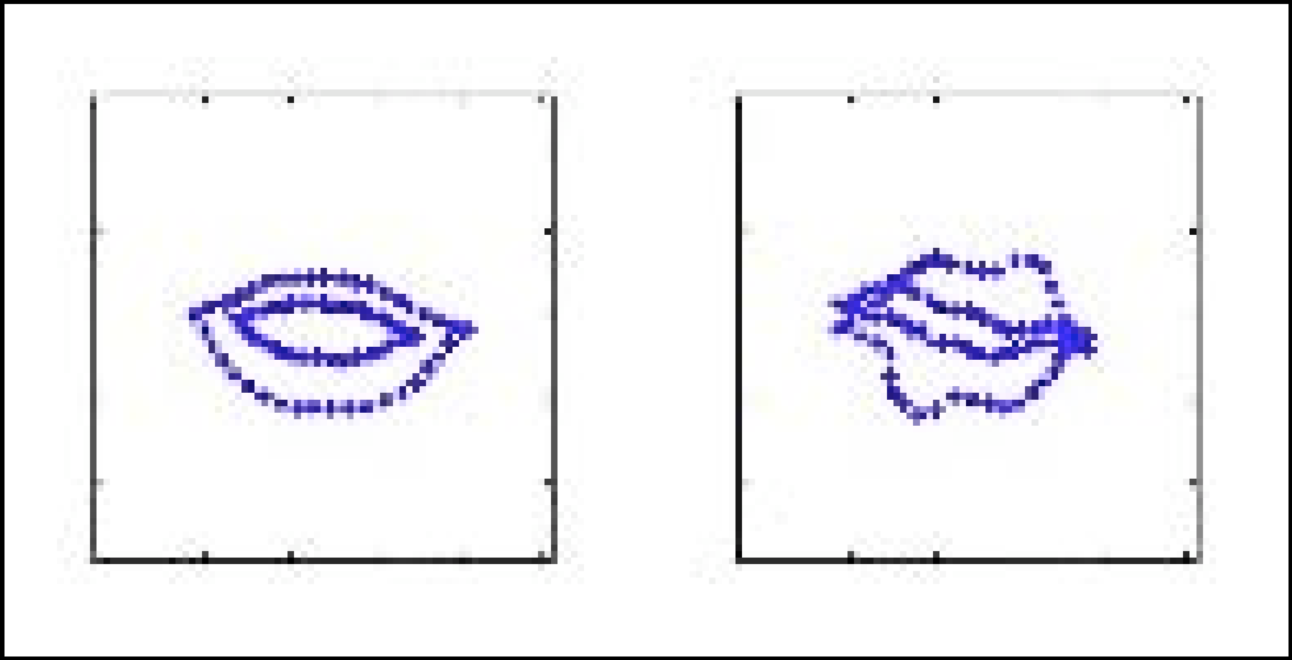
Derive source blendshapes from data

Errors in representing the source can result in large cancelling basis combinations (nearly cancelling positive, negative weights)



Chuang and Bregler,

Derive source blendshapes from data



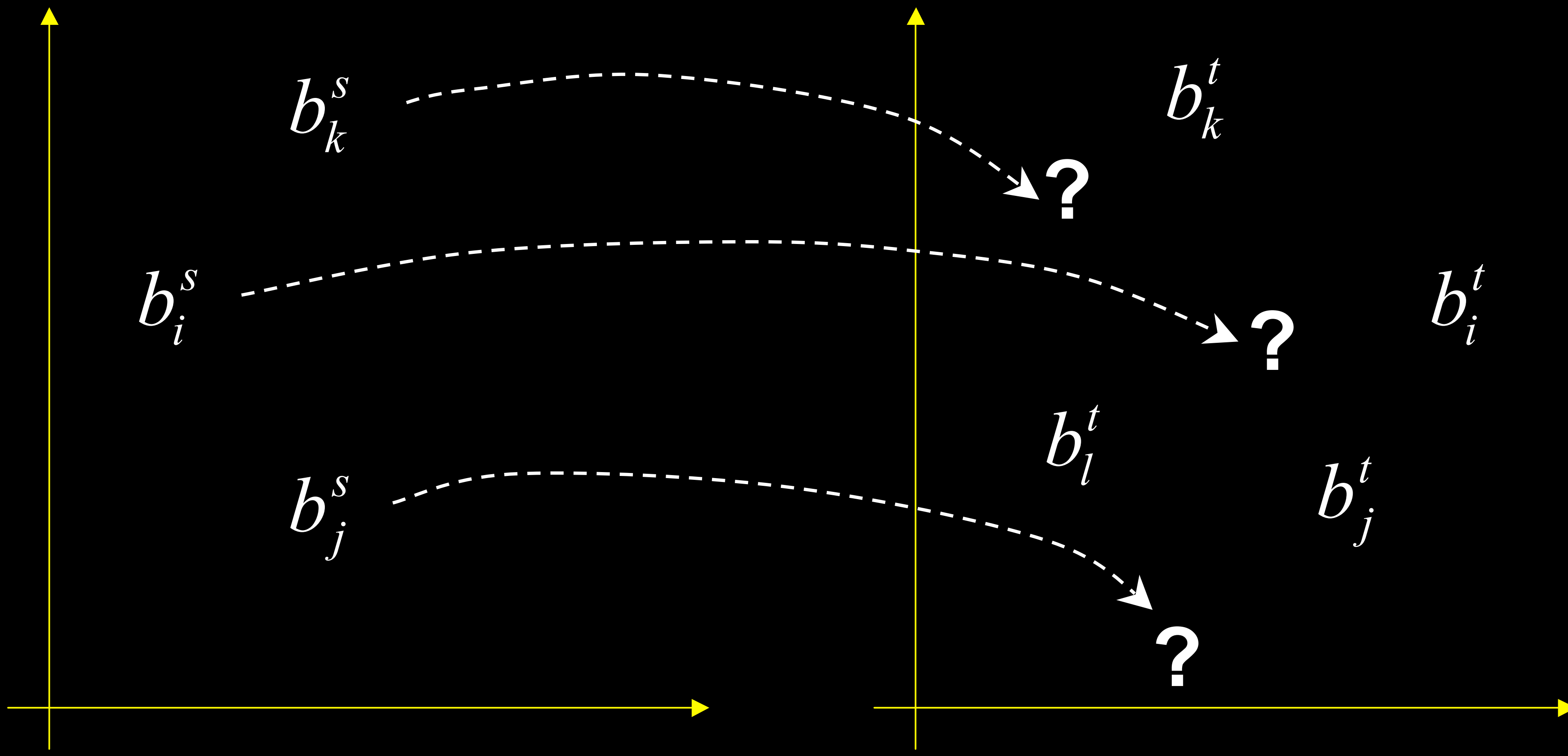
Chuang and Bregler,

Derive source basis from data

- Solution
  - Solve for the representation of the source with non-negative least squares. Prohibiting negative weights prevents the cancelling combinations.
  - Robust cross mapping.

Performance Driven Facial Animation using Blendshape Interpolation

Non-corresponding blendshapes



Is there a "best" blendshape basis?

SIGGRAPH2006

- There are an infinite number of different blendshape models that have exactly the same range of movement. Proof:

$$\begin{aligned} f &= Bw \\ &= B(KK^{-1})w \\ &= (BK)(K^{-1}w) \\ &= Dx \text{ with } D \equiv BK, x \equiv K^{-1}w \end{aligned}$$

- And it's easy to interconvert between different blendshapes

analogy: what is the best view of a 3D model? Why restrict yourself to only one view??

Global blendshape mapping

SIGGRAPH2006

Motivating scenarios:

1) Use PCA for source!
2) Source or target model is pre-existing (e.g. from a library)

Global blendshape mapping

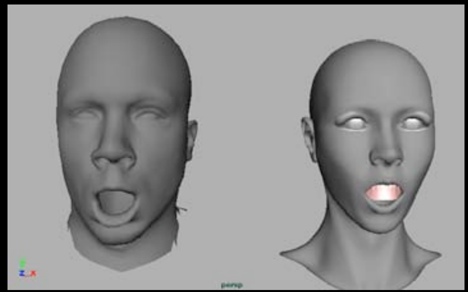
SIGGRAPH2006

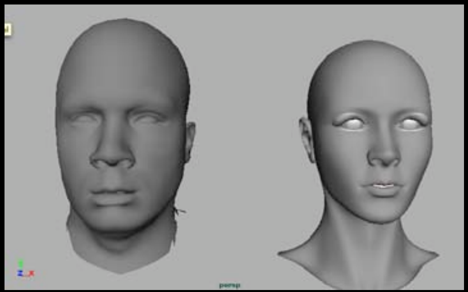
$$\mathbf{s} = \mathbf{B}\mathbf{w} \quad \text{weights } \mathbf{w} \text{ pose the source}$$

$$\mathbf{t} = \mathbf{C}\mathbf{v} \quad \text{weights } \mathbf{v} \text{ pose the target}$$

$\mathbf{B},\mathbf{C} \in \Re^{3n \times m}$ n vertices, m blendshape targets

Manually create $p \geq m$ corresponding poses of each model, with weights $\mathbf{v}_k$, $\mathbf{w}_k$





Global blendshape mapping

SIGGRAPH2006

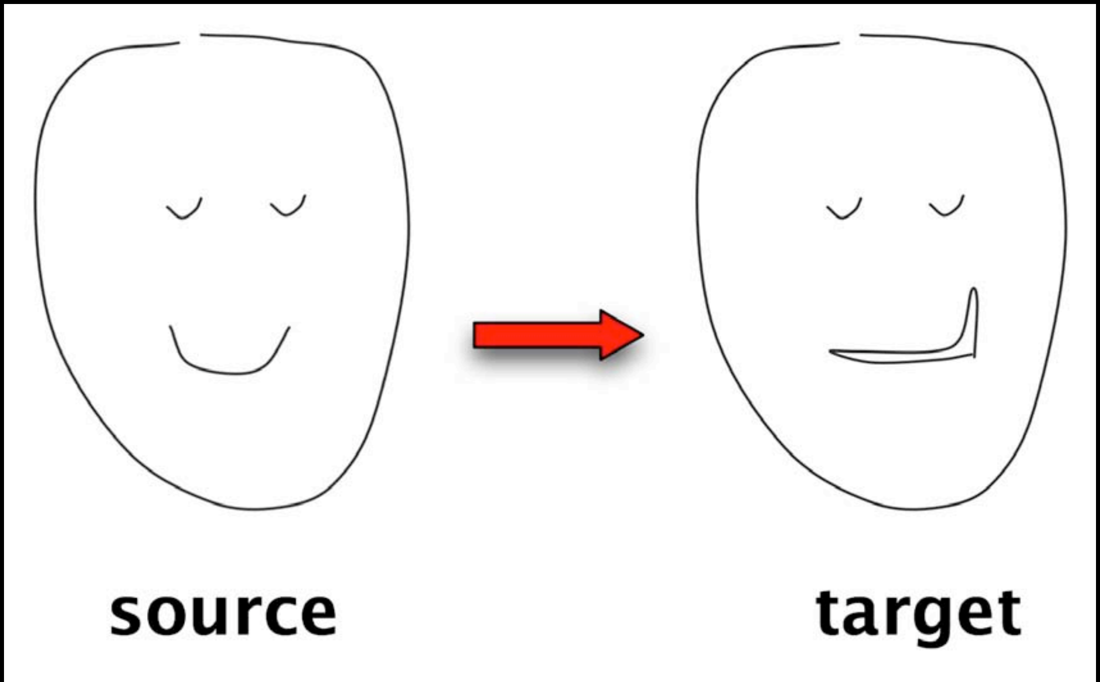
- Gather pose weight vectors $\mathbf{v}_k$, $\mathbf{w}_k$ in columns of $\mathbf{V},\mathbf{W}$
- Solve for the "cloning matrix" $\mathbf{E}$:

$$V = EW \;\Rightarrow\; E = VW^T(WW^T)^{-1}$$

this matrix converts weights for one model to produce the equivalent expression in the other model.

Global blendshape mapping

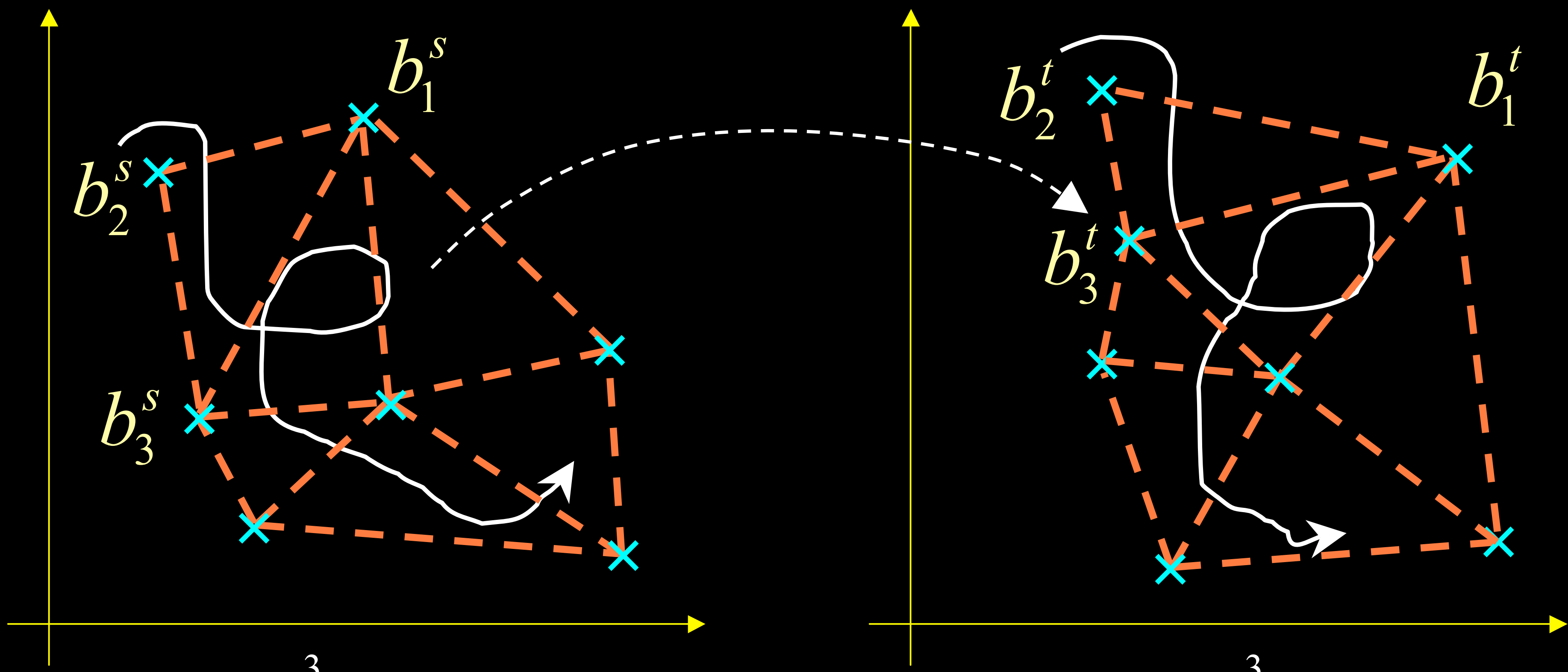
- *Intentional* source-target mismatch: style transfer (person on the right has asymmetric smile)



Non-linear Mapping

- Piecewise linear
- Manifold learning
- Single-correspondence

Piecewise linear mapping



$$f_k^s = \sum_{i=1}^{3} w_{k,i} b_{k,i}^s \longrightarrow f_k^t = \sum_{i=1}^{3} w_{k,i} b_{k,i}^t$$

(Local) Change of coordinate system

Piecewise linear mapping

SIGGRAPH2006

- I. Buck, A. Finkelstein, C. Jacobs, A. Klein, D. H. Salesin, J. Seims, R. Szeliski, and K. Toyama, **Performance-driven hand-drawn animation**, *NPAR 2000*

Buck et. al.

Piecewise linear mapping

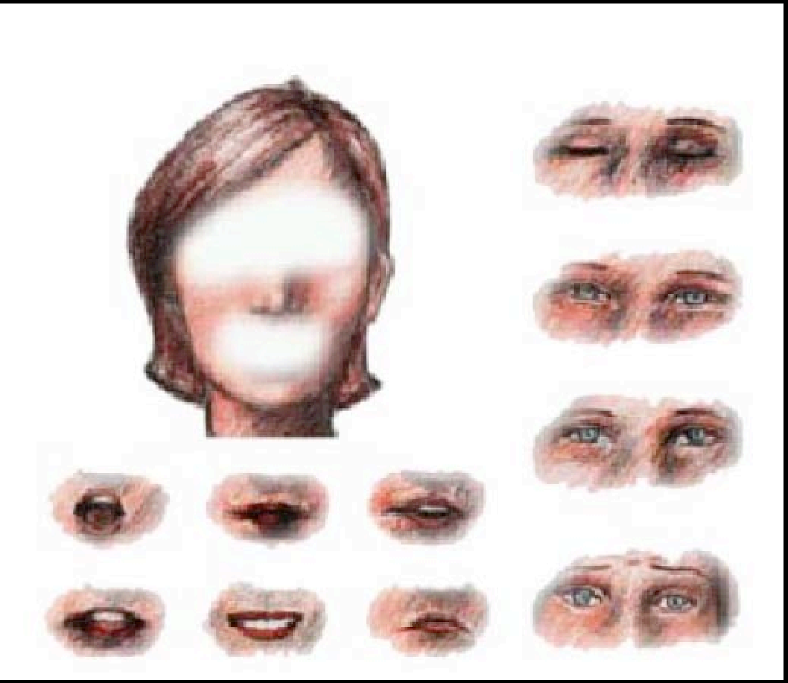
SIGGRAPH2006

- Project motion onto a 2D space (PCA)
- Construct Delaunay triangulation based on source blendshapes
- Within a triangle use barycentric coordinates as blending weights

Buck et. al.

Piecewise linear mapping

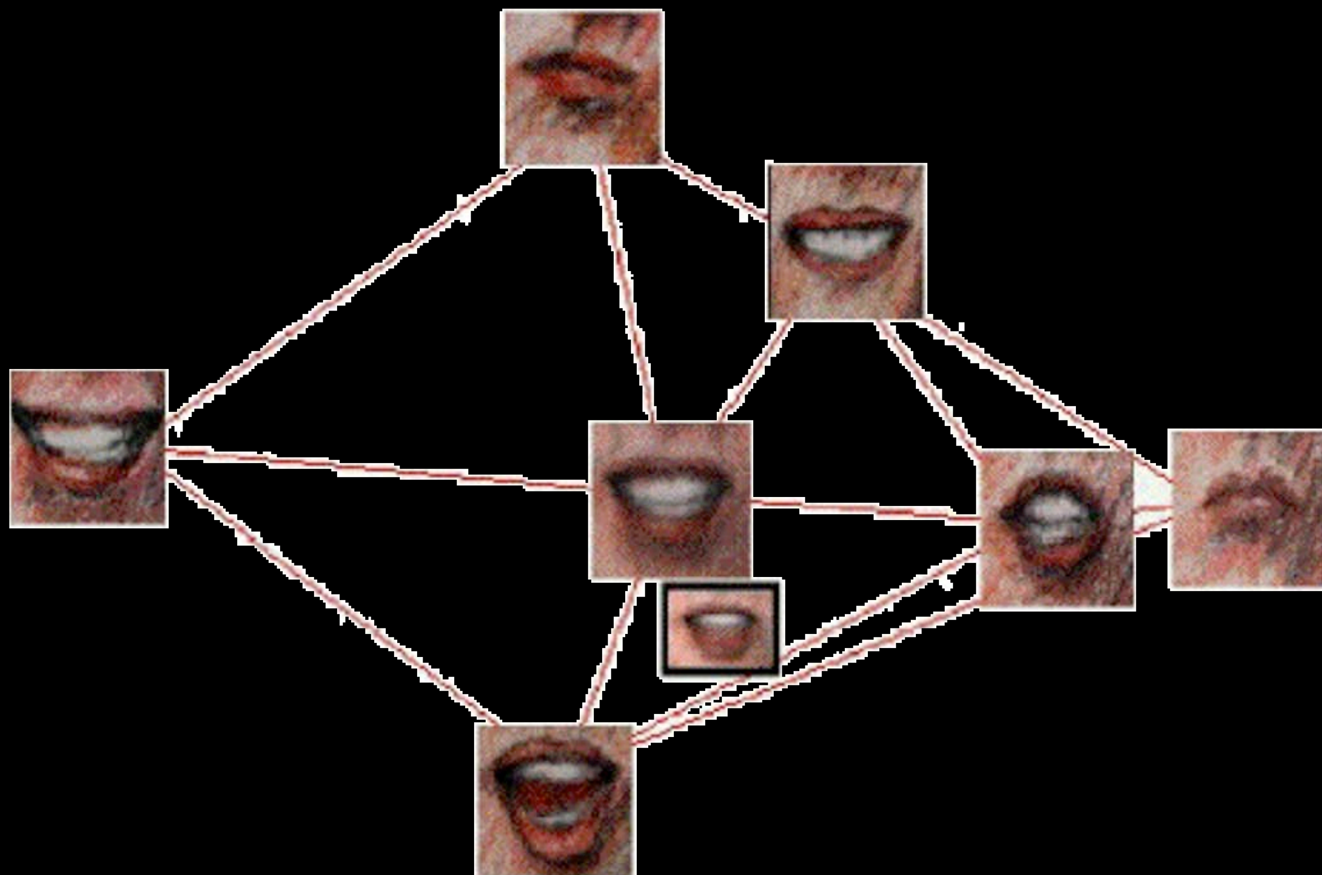
- Split face into 3 regions
  (eyes, mouth,
  rest of the face)



- If frame outside of triangulated area, project onto convex hull

Buck et. al.

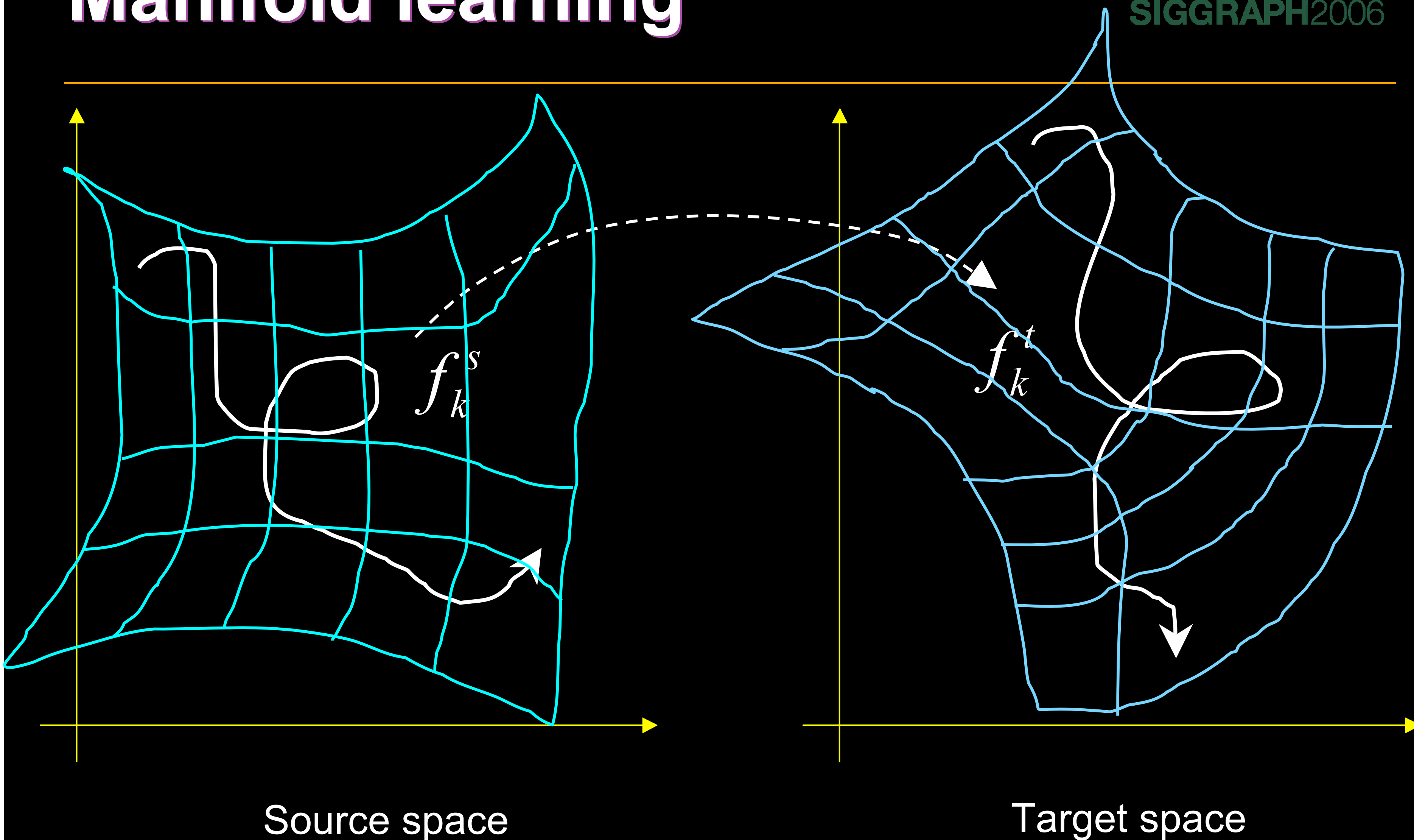
Triangulation



Video

- hand_drawn.avi

Manifold learning

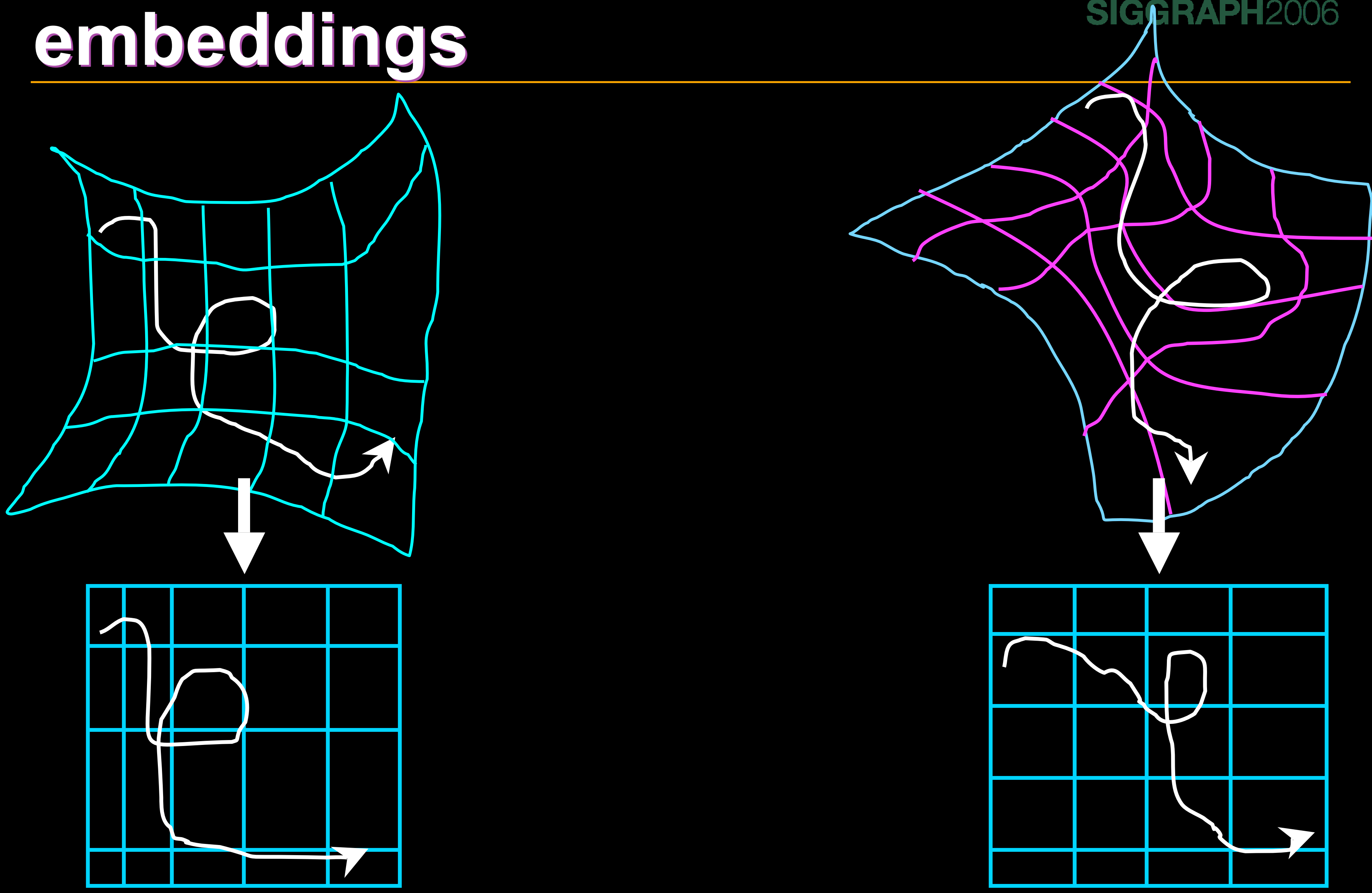


Manifold learning

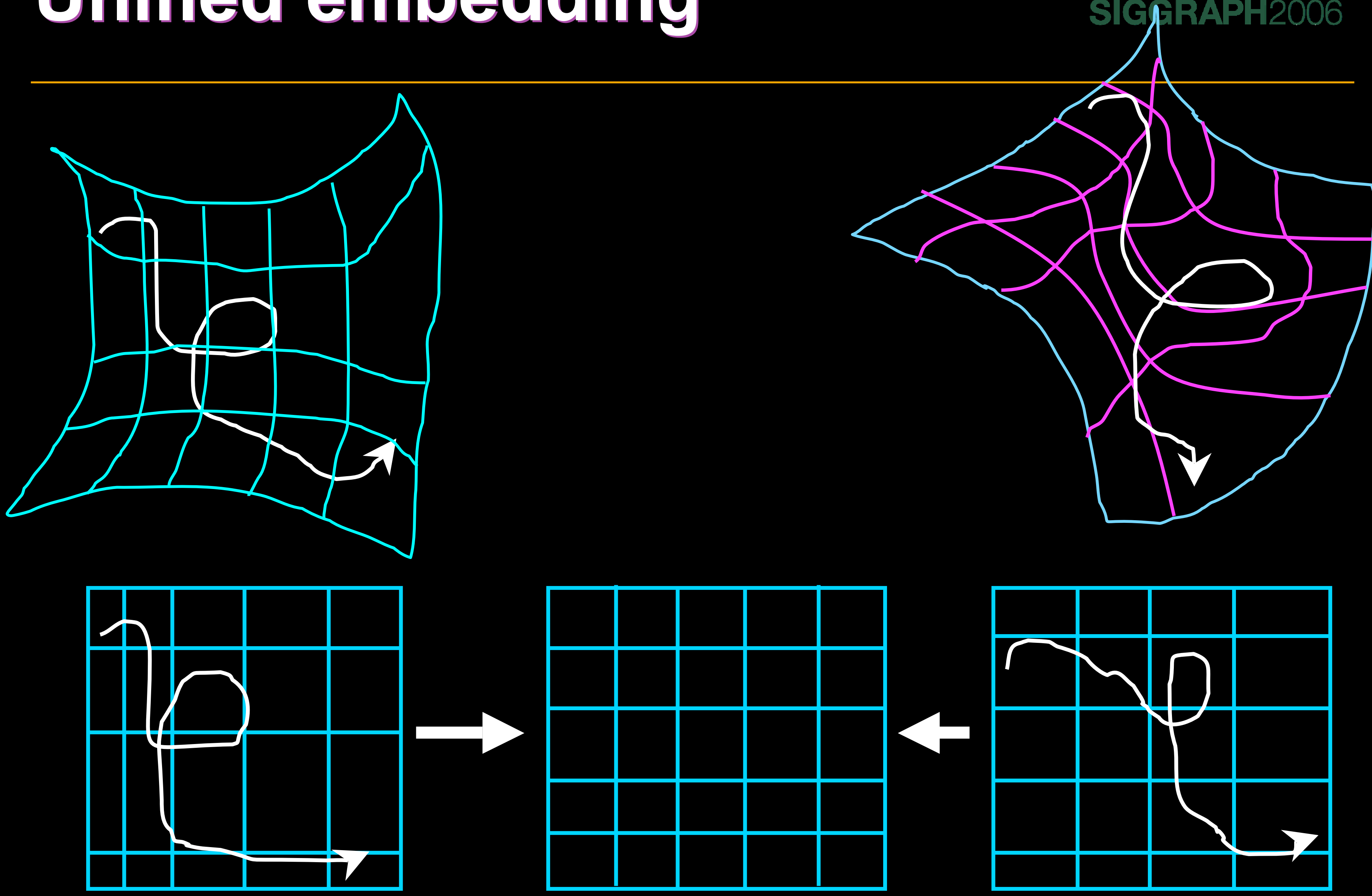
SIGGRAPH2006

- Y. Wang, X. Huang, C.-S. Lee, S. Zhang, D. Samaras, D. Metaxas, A. Elgammal, and P. Huang, **High Resolution Acquisition, Learning, and Transfer of Dynamic 3-D Facial Expressions**, *Eurographics* 2004.

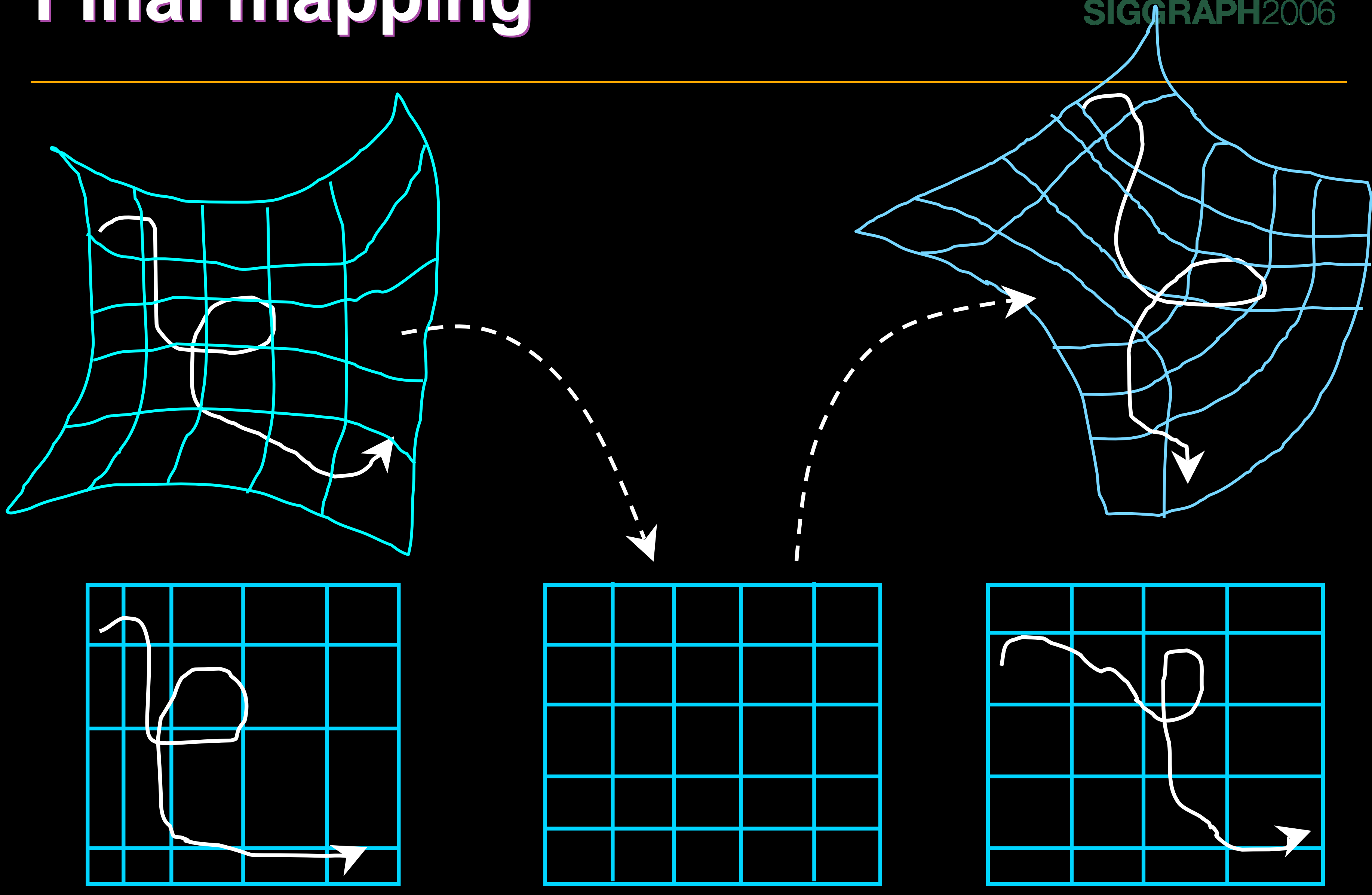
Source and target embeddings



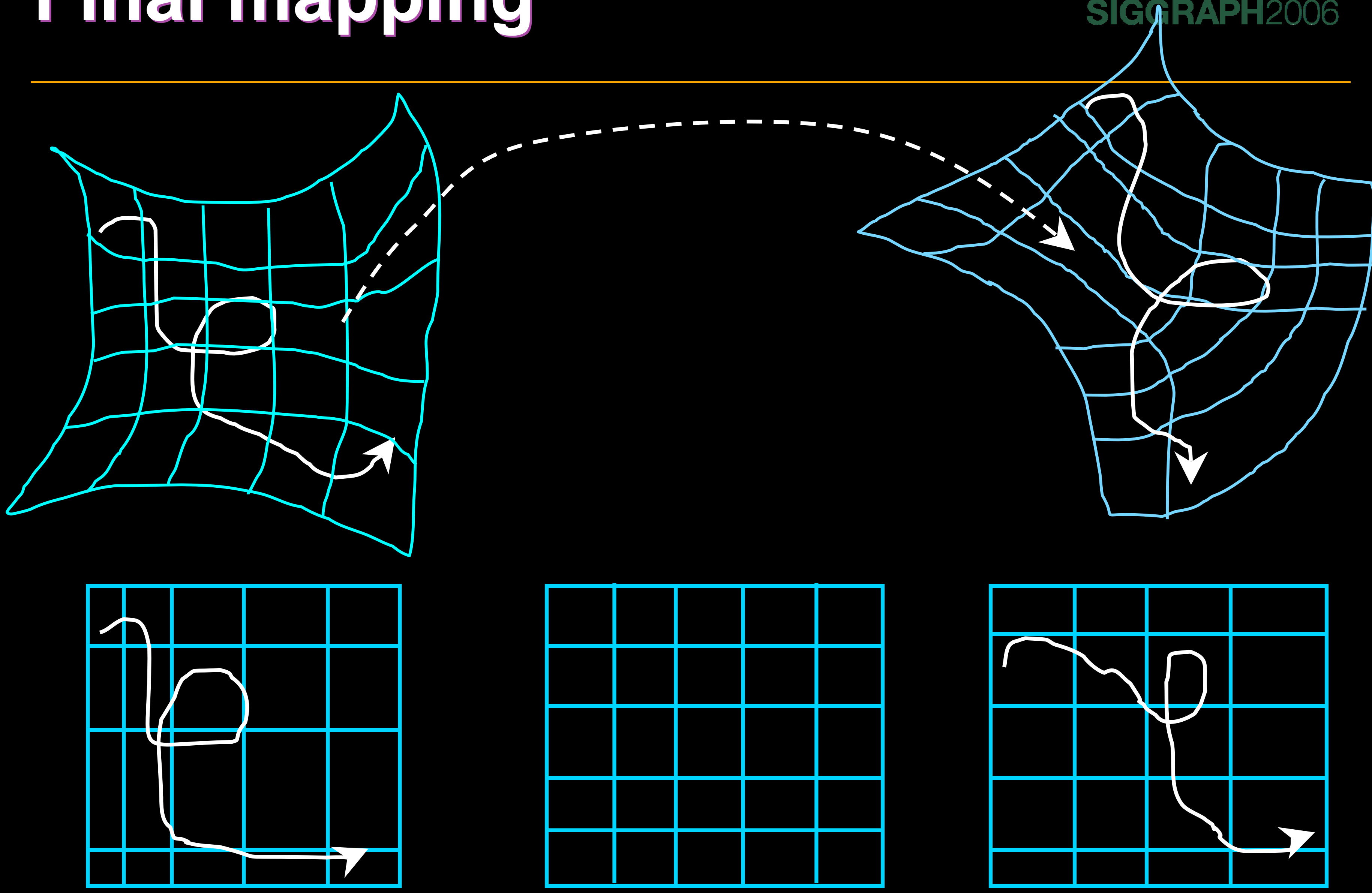
Unified embedding



Final mapping

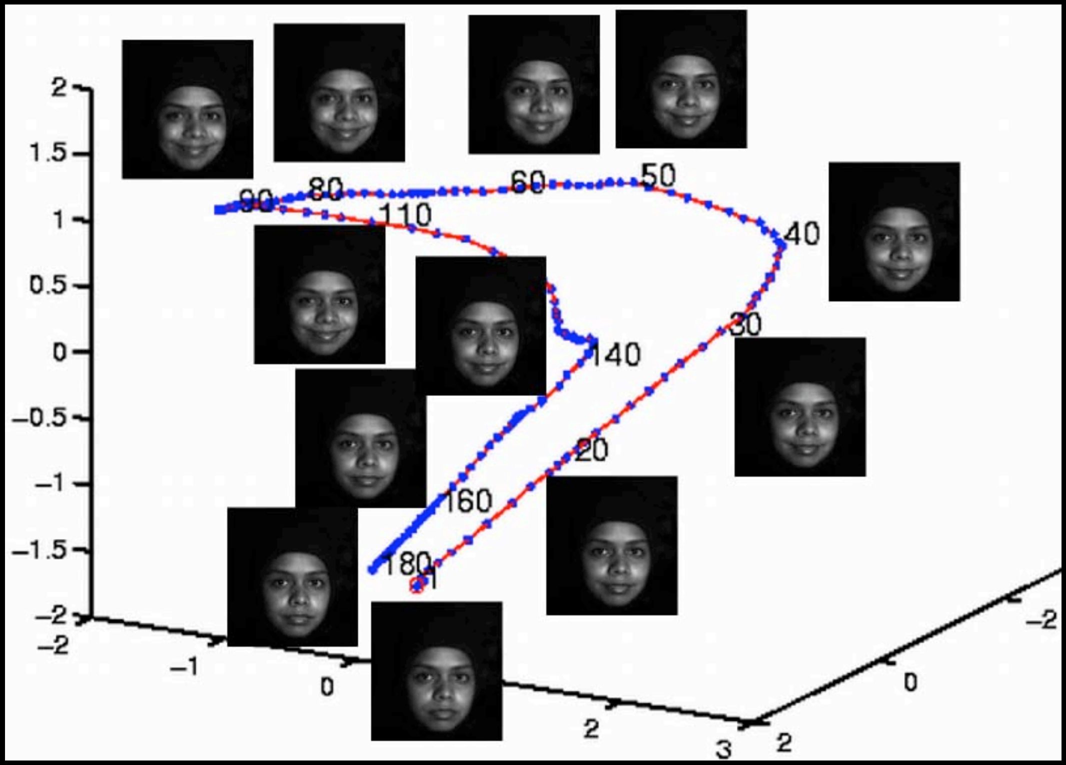


Final mapping



Wang et. al.

Manifold learning

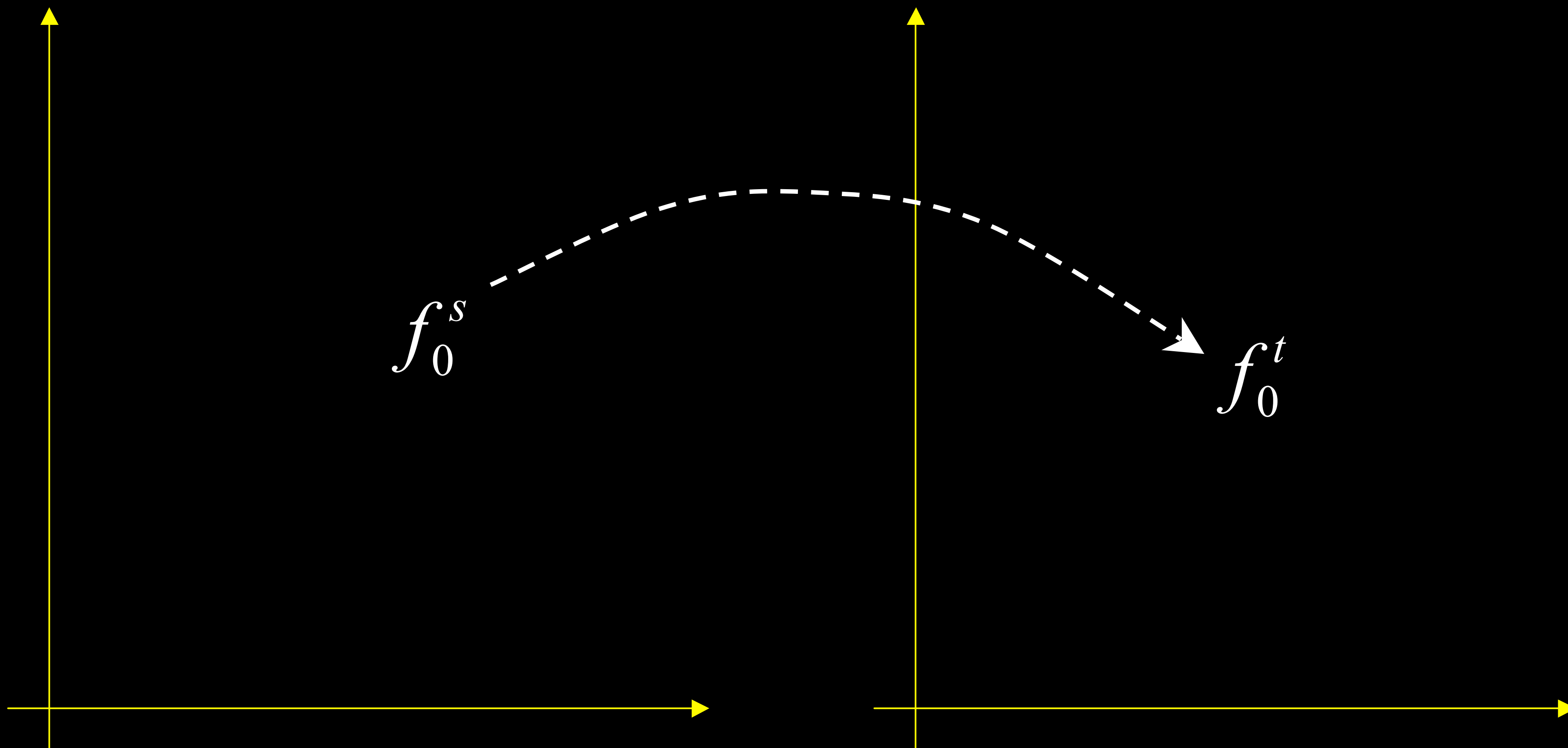


Manifold (curve) of smile motion obtained by Local Linear Embedding (from Wang et. al.)

Video

- final-video2-edit.mov

Mapping from a single correspondence



$$f^t = M\left(f^s \mid (f_0^s, f_0^t)\right)$$

Mapping from a single correspondence

- J.-Y. Noh and U. Neumann, **Expression Cloning**, *SIGGRAPH* 2001.

Noh et. al.
Two issues

- Find dense geometric correspondences between the two face models
- Map motion using local geometric deformations from source to target face

Noh et. al.

Estimating geometric correspondences

- Sparse correspondences through feature detection
- Dense correspondences by interpolating matching features (RBF)

Noh et. al.

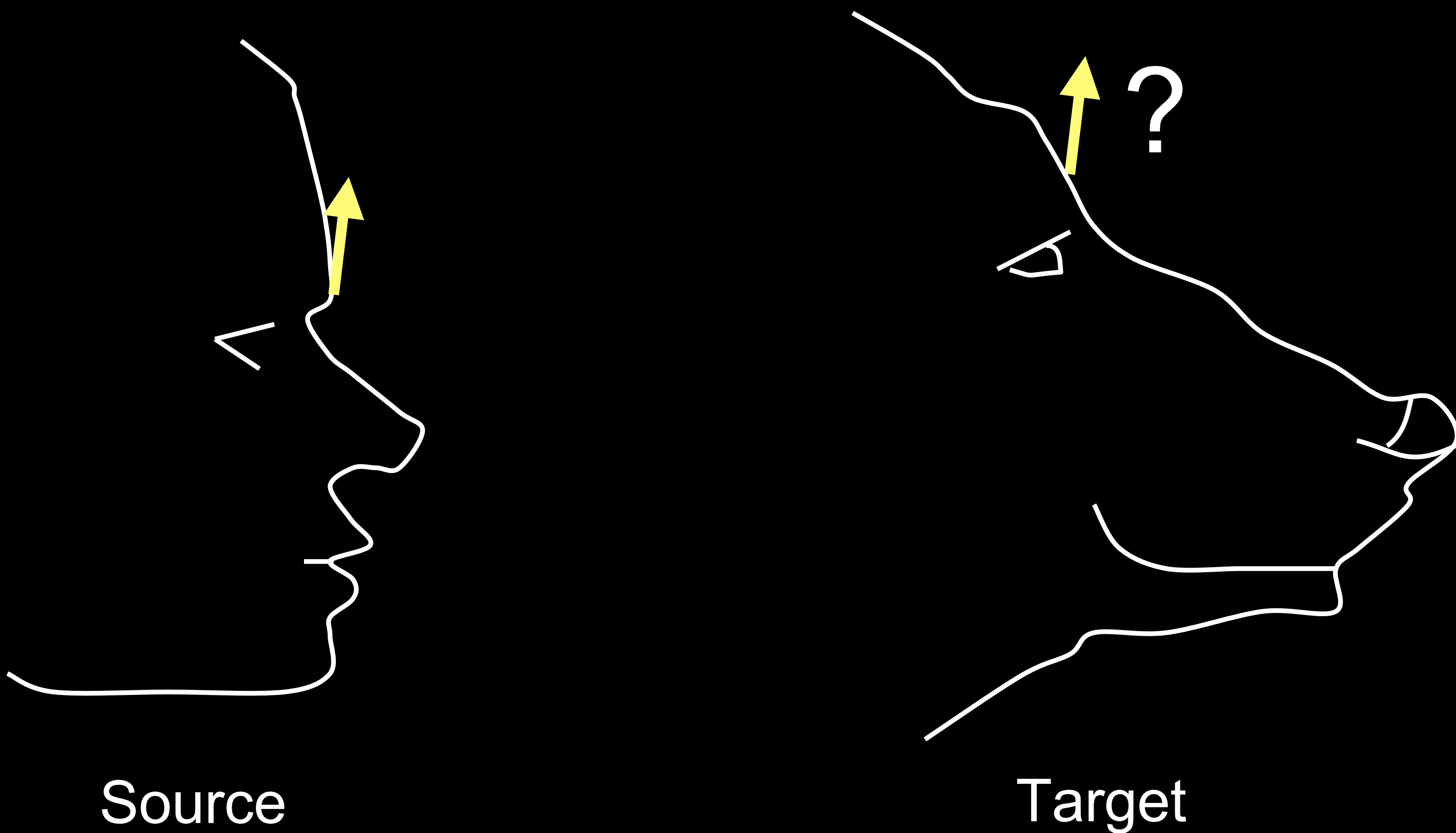
Local geometric motion transformation

Noh et. al.

Animation as displacement from the neutral/rest face

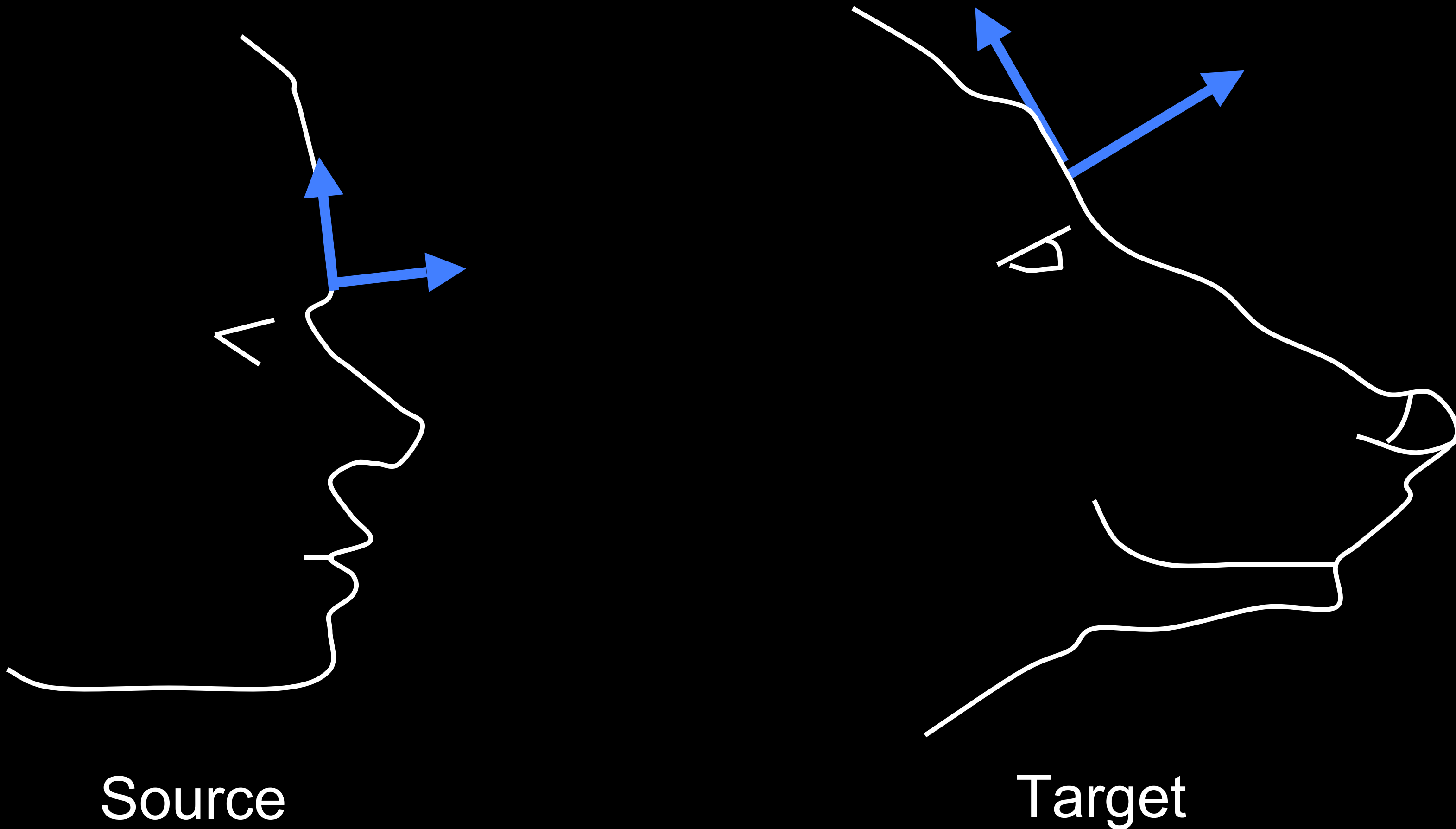
Noh et. al.

Local geometric motion transformation



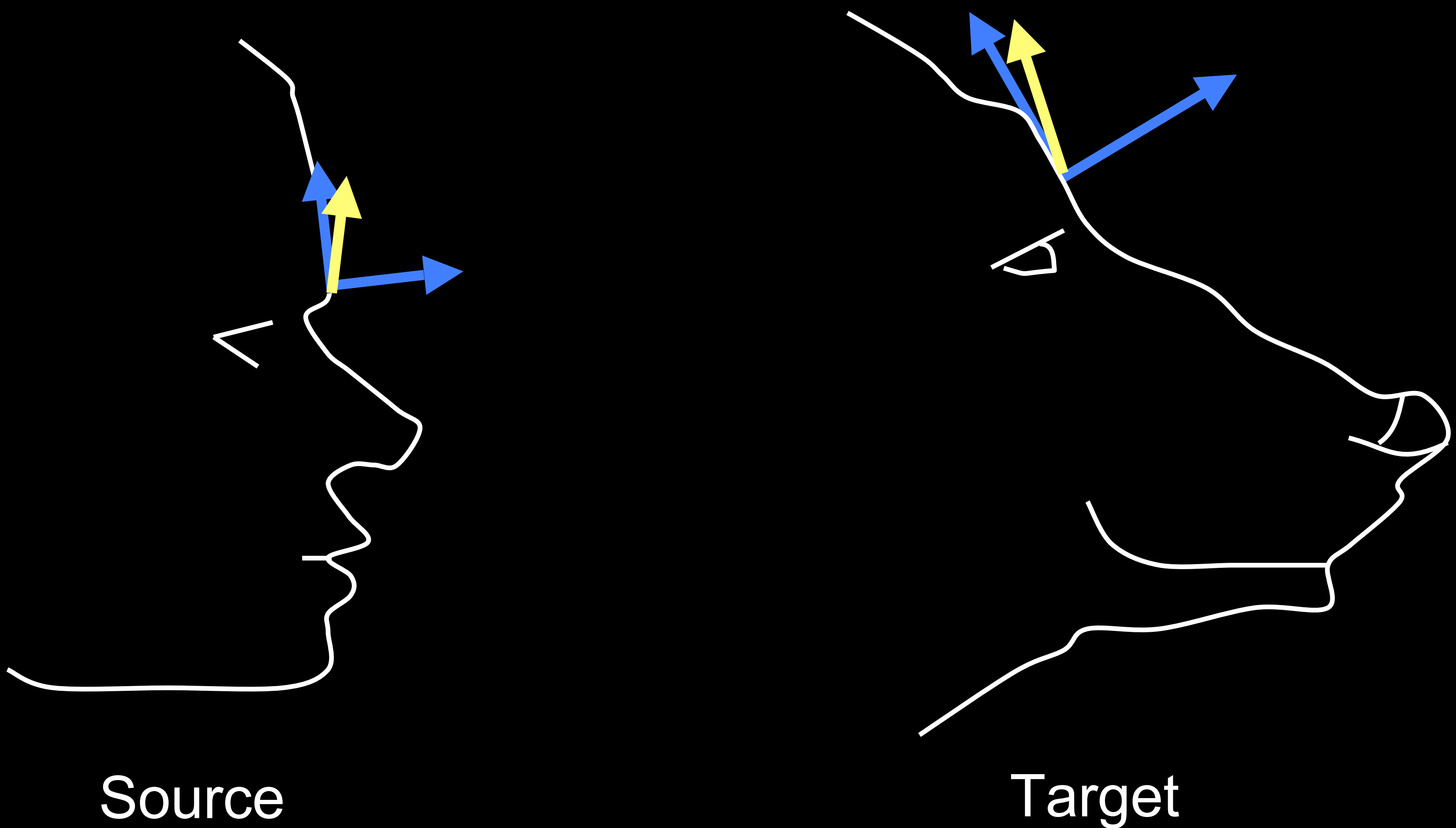
Noh et. al.

Local change of coordinates system



Noh et. al.

Local change of coordinates system



Noh et. al.

Local coordinates system

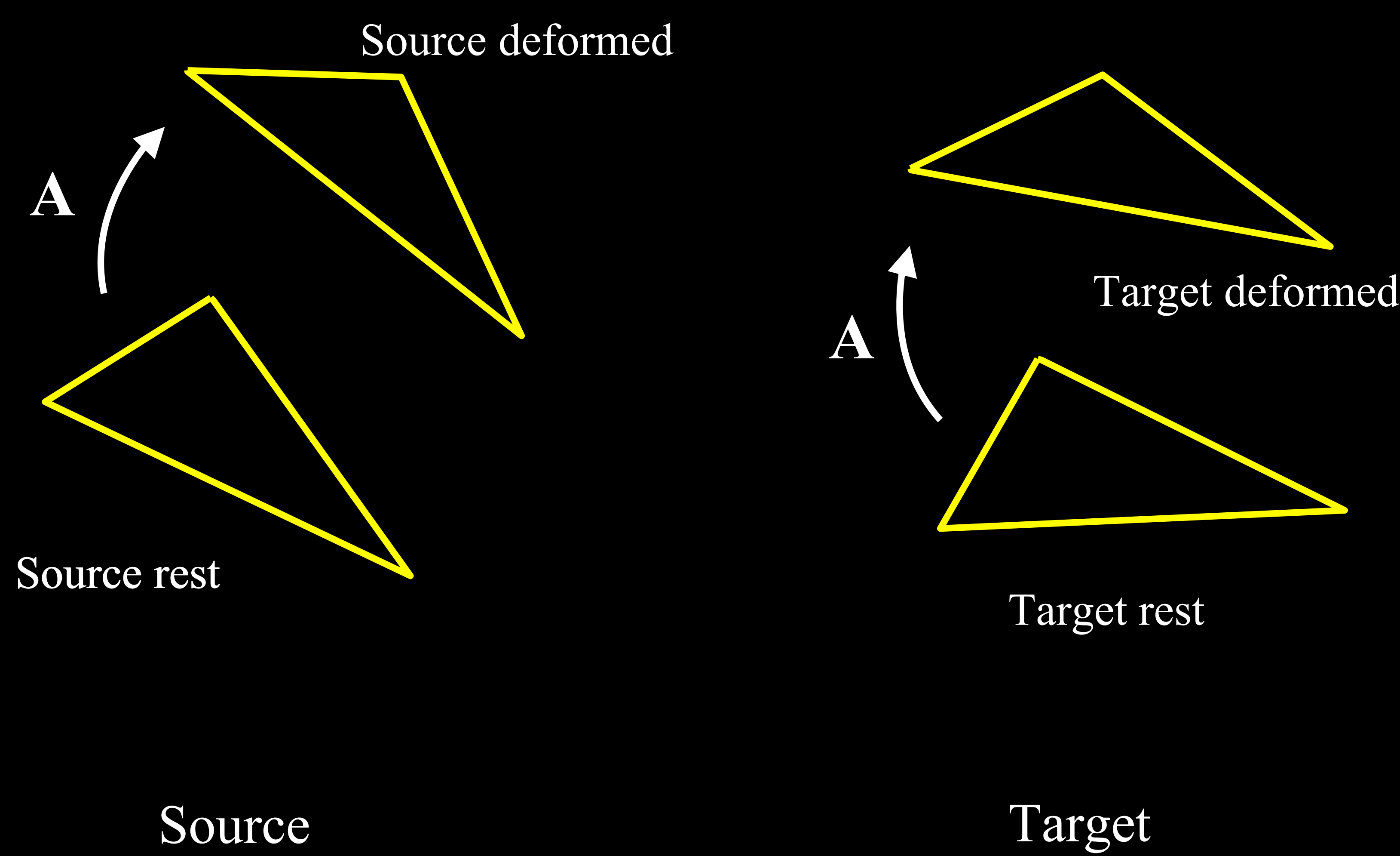
- Defined by
  - Tangent plane and surface normal
  - Scale factor: ratio of bounding boxes containing all triangles sharing vertex

Video

- monkeyExp.mov

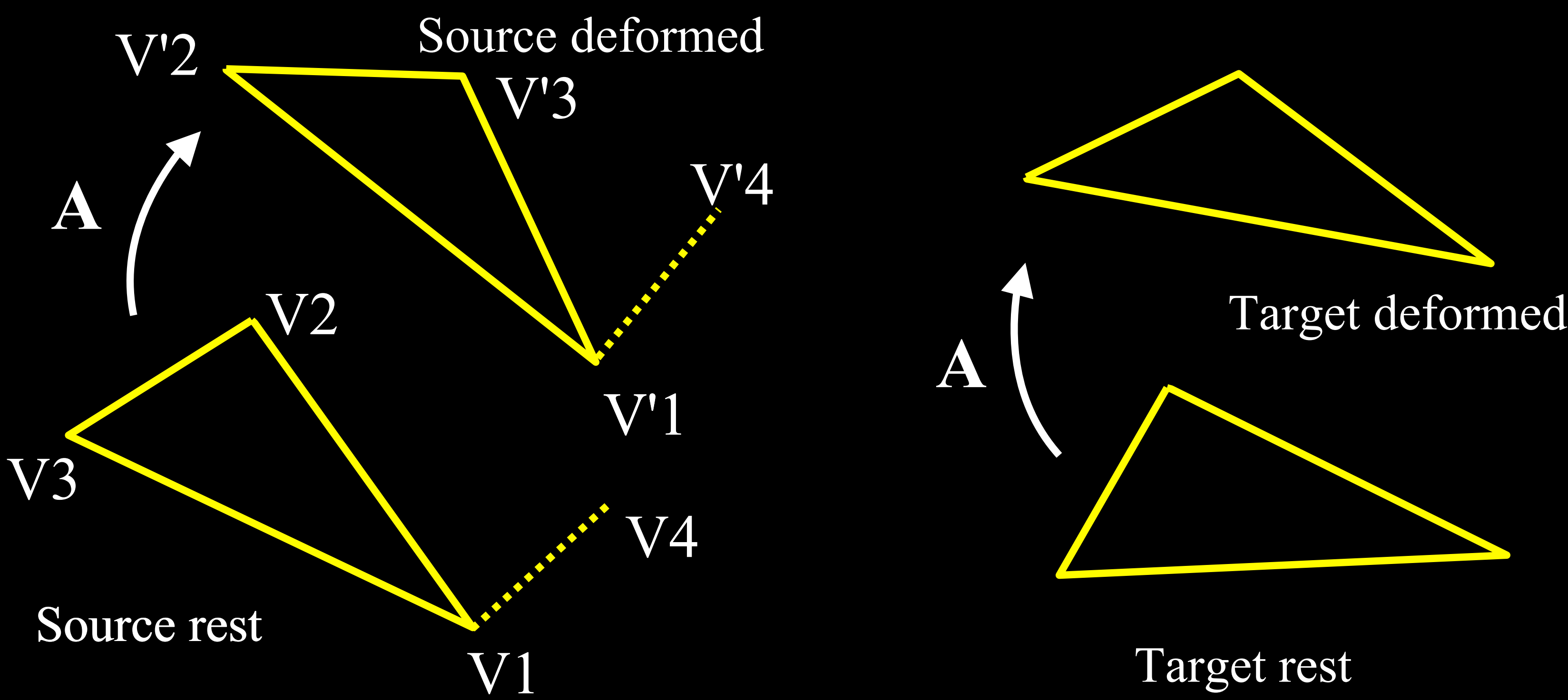
Summer and Popovic

Global geometric motion transformation



Summer and Popovic

Global geometric motion transformation



$$(V1, V2, V3, V4) \xrightarrow{\mathbf{A}} (V'1, V'2, V'3, V'4)$$

with V4 = V1 + (V2 – V1) x (v3 – v1)/ sqrt (||(V2 – V1) x (v3 – v1)||)

Summer and Popovic

Global geometric motion transformation

- Constraint vertices to move consistently with respect to each triangle it belongs to
- Solve system of linear equations

References

- B. Choe and H. Ko, **Analysis and Synthesis of Facial Expressions with Hand-Generated Muscle Actuation Basis**, *Computer Animation* 2001.
- I. Buck, A. Finkelstein, C. Jacobs, A. Klein, D. H. Salesin, J. Seims, R. Szeliski, and K. Toyama, **Performance-driven hand-drawn animation**, *NPAR 2000.*
- E. Chuang and C. Bregler, **Performance Driven Facial Animation using Blendshape Interpolation**, CS-TR-2002-02, Department of Computer Science, Stanford University
- J.-Y. Noh and U. Neumann, **Expression Cloning**, *SIGGRAPH* 2001.
- R. W. Summer and J. Popovic, **Deformation Transfer for Triangle Meshes**,*SIGGRAPH* 2004
- Y. Wang, X. Huang, C.-S. Lee, S. Zhang, D. Samaras, D. Metaxas, A. Elgammal, and P. Huang, **High Resolution Acquisition, Learning, and Transfer of Dynamic 3-D Facial Expressions**, *Eurographics* 2004.

Future work

- Artists-driven retargeting
- Physically-based retargeting

SIGGRAPH Course 30: Performance-Driven Facial Animation

For Latest Version of Bregler's Slides and Notes please go to:

http://cs.nyu.edu/~bregler/sig-course-06-face/

SIGGRAPH Course 30: Performance-Driven Facial Animation

Section:

Markerless Face Capture and Automatic Model Construction

Part 1: Chris Bregler, NYU

Markerless Face Capture

SIGGRAPH2006

Markerless Face Capture - Overview -

SIGGRAPH2006

- Single / Multi Camera Input
- 2D / 3D Output
- Real-time / Off-line
- Interactive-Refinement / Face Dependent / Independent
- Make-up / Natural
- Flow / Contour / Texture / Local / Global Features
- Hand Crafted / Data Driven
- Linear / Nonlinear Models / Tracking

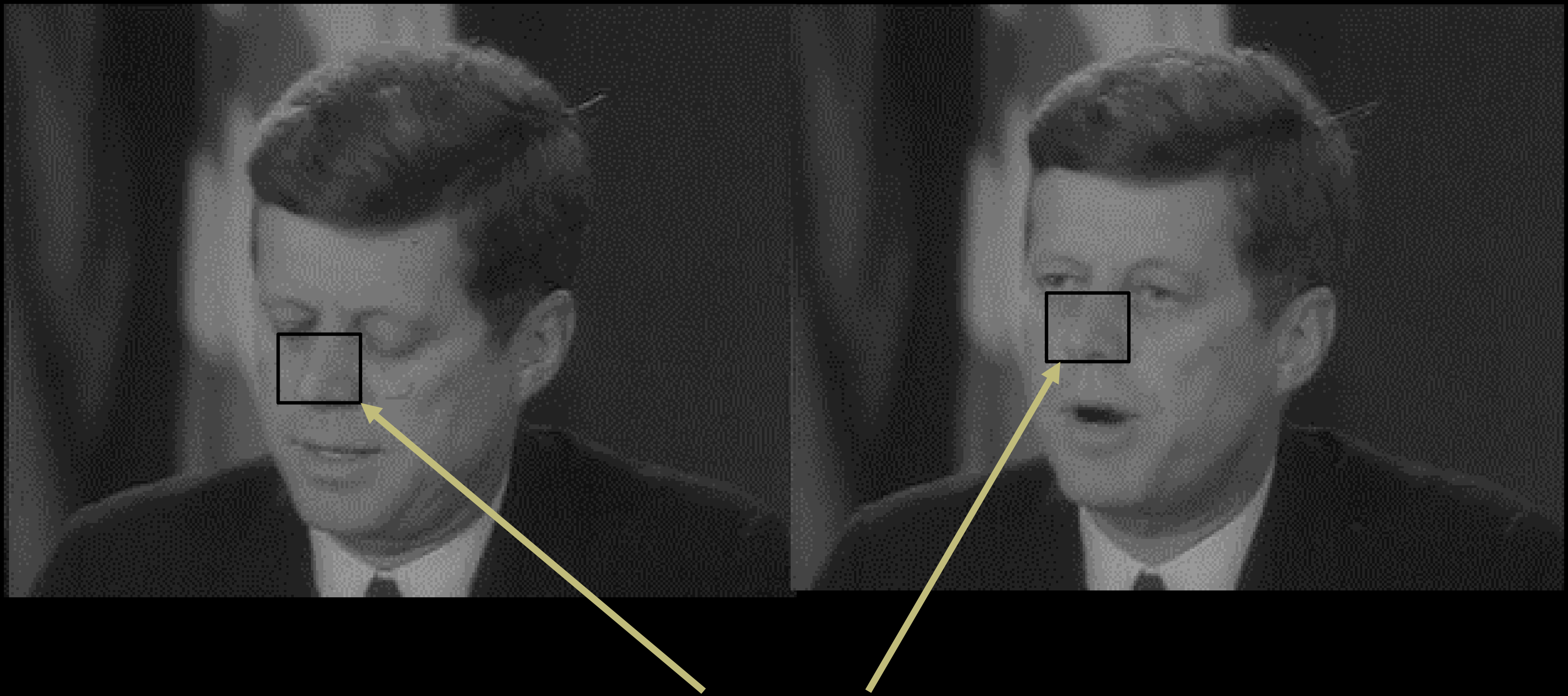
Markerless Face Capture – History –

SIGGRAPH2006

- Single Camera Input
- 2D Output
- Off-line
- Interactive-Refinement
- Make-up
- Contour / Local Features
- Hand Crafted
- Linear Models / Tracking

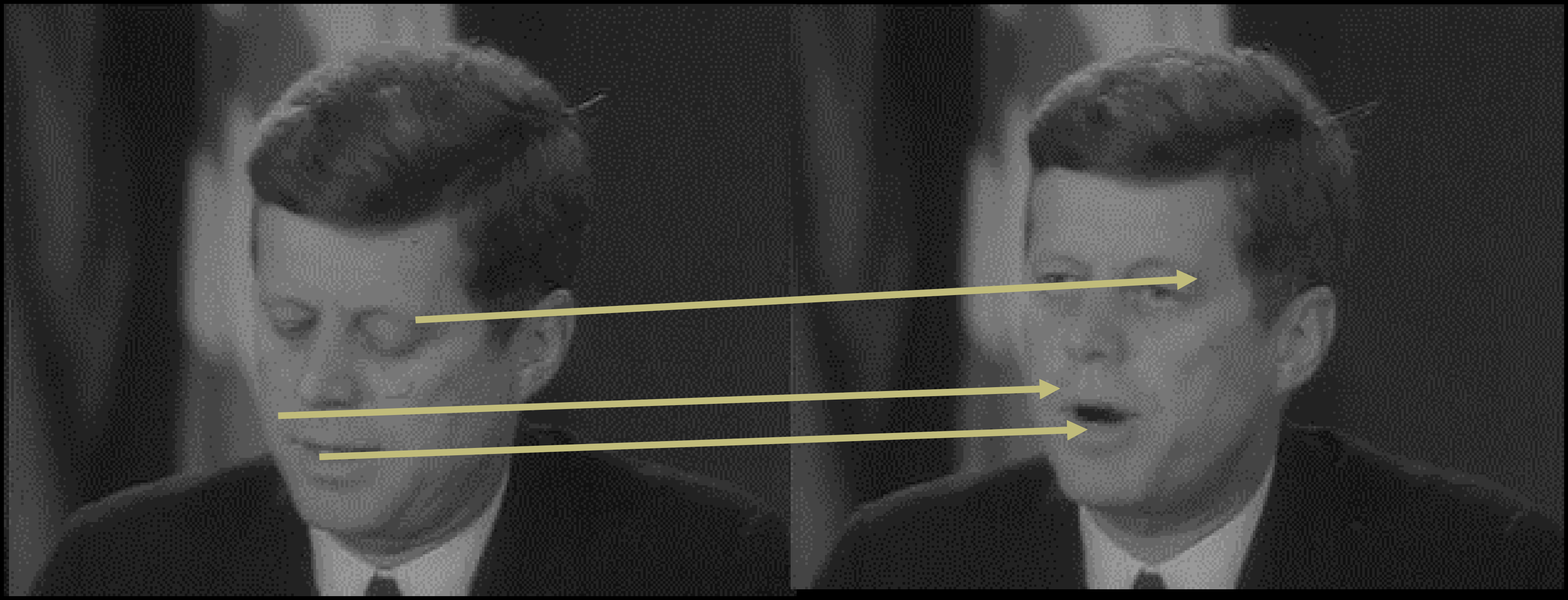
Kass, M., Witkin, A., & Terzopoulos, D. (1987) Snakes: Active contour models.

Tracking = Error Minimization



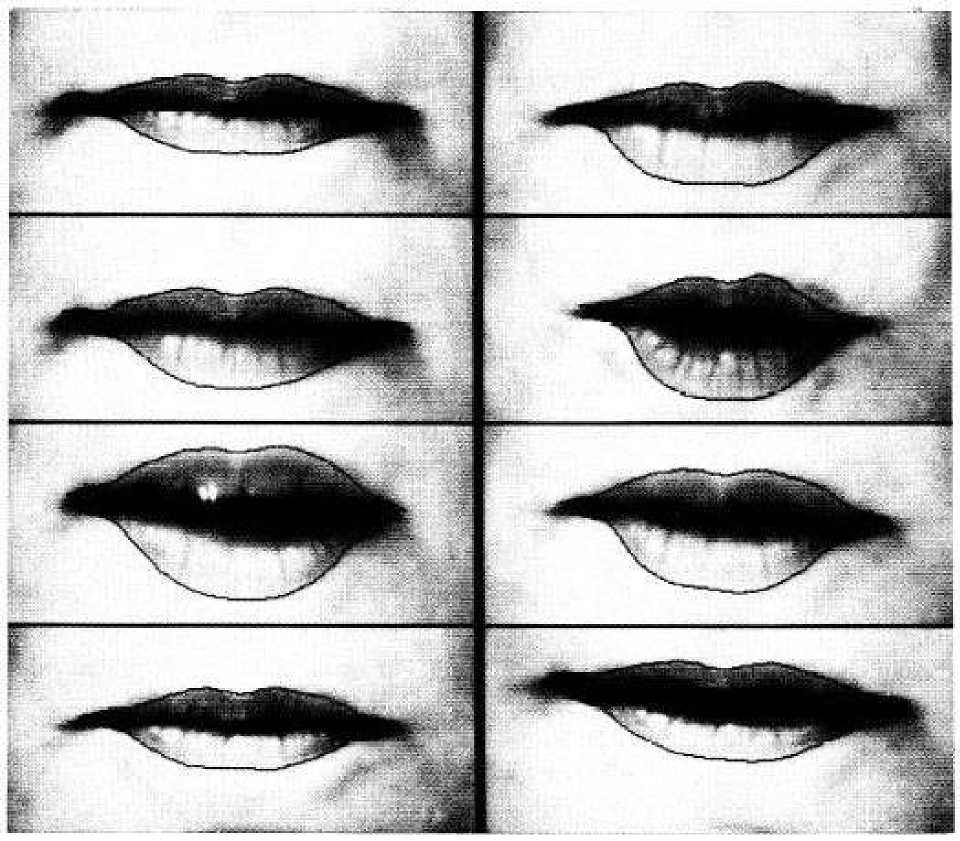
$$Err(u,v) = \sum || I(x,y) - J(x+u, y+v) ||$$

Tracking = Error Minimization



In general: ambiguous using local features

Tracking = Error Minimization



$$E^{*}_{\text{snake}} = \int_0^1 E_{\text{snake}}(\mathbf{v}(s))\,ds$$
$$= \int_0^1 E_{\text{int}}(\mathbf{v}(s)) + E_{\text{image}}(\mathbf{v}(s)) + E_{\text{con}}(\mathbf{v}(s))\,ds$$

Kass, M., Witkin, A., & Terzopoulos, D. (1987) Snakes: Active contour models.

Tracking = Error Minimization

SIGGRAPH2006

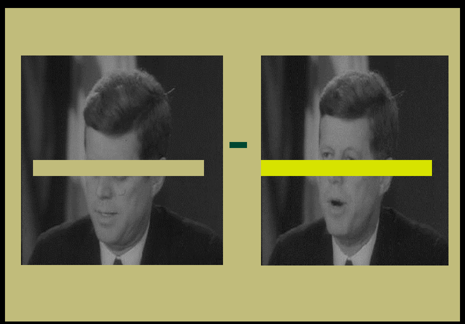
Error = Feature Error + Model Error

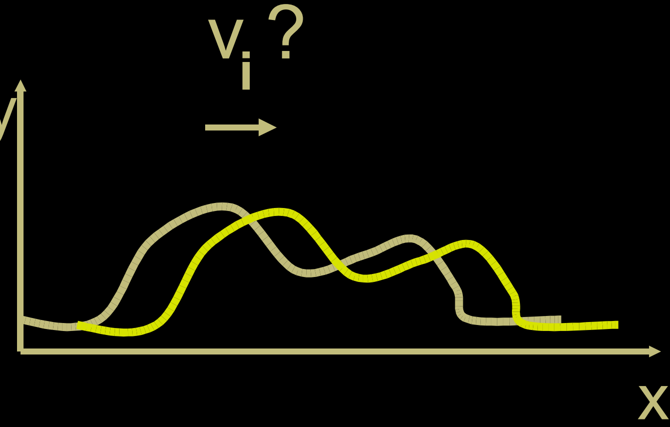
Tracking = Error Minimization

Error = **Optical Flow** + Model Error

Optical Flow (Lucas-Kanade)

I J

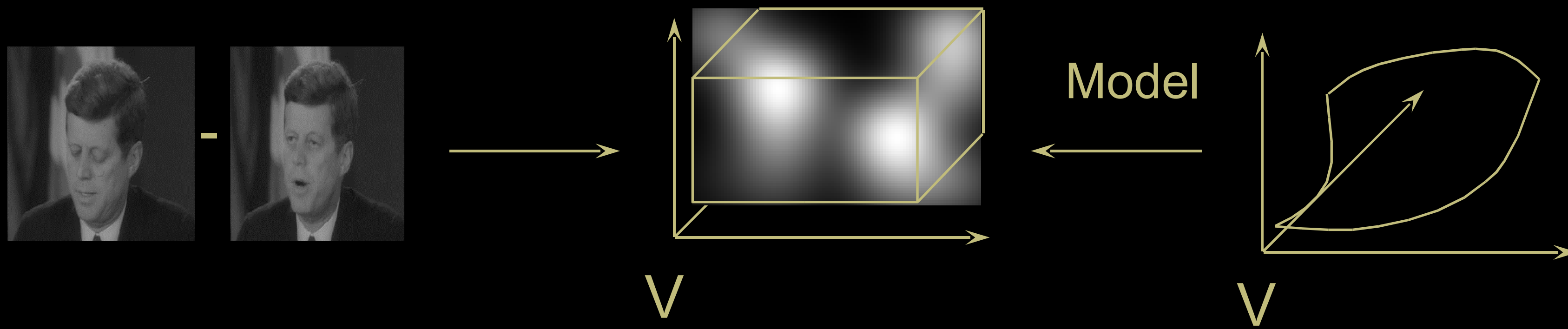




$$E(V) \quad \left\| \mathbf{I}(x_i) - \mathbf{J}(x_i + v_i) \right\|^2$$

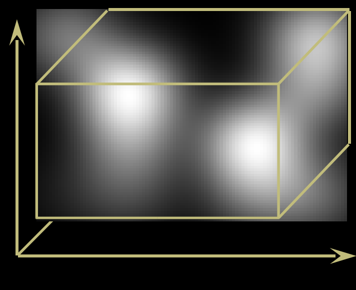
linearize $$\left\| \mathbf{I}_t(x_i) - \nabla \mathbf{I}(x_i) v_i \right\|^2$$

Optical Flow + Model



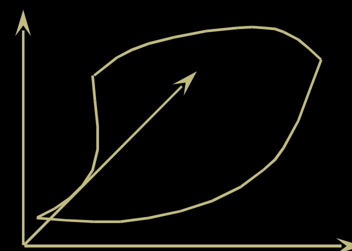
$$\left\| \begin{matrix} I_t(1) - \nabla I(1)\,\vec{v}_1 \\ I_t(2) - \nabla I(2)\,\vec{v}_2 \\ \cdots \\ I_t(n) - \nabla I(n)\,\vec{v}_n \end{matrix} \right\|^2 = E(V)$$

Optical Flow + Model



V

Model



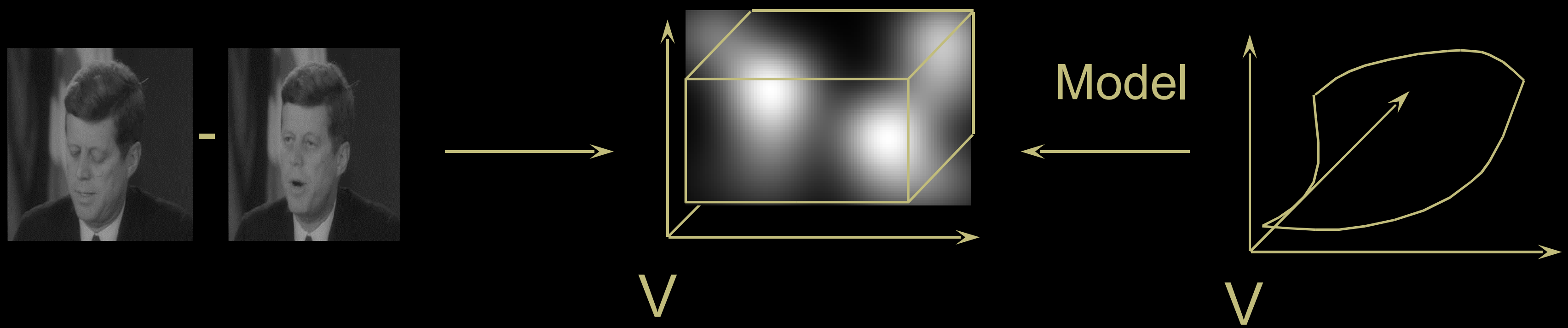
V

$$\left\| \begin{matrix} I_t(1) - \nabla I(1)\, \vec{v}_1 \\ I_t(2) - \nabla I(2)\, \vec{v}_2 \\ \dots \\ I_t(n) - \nabla I(n)\, \vec{v}_n \end{matrix} \right\|^2 = E(V)$$

$$V = M(\theta)$$

Optical Flow + linearized Model

SIGGRAPH2006



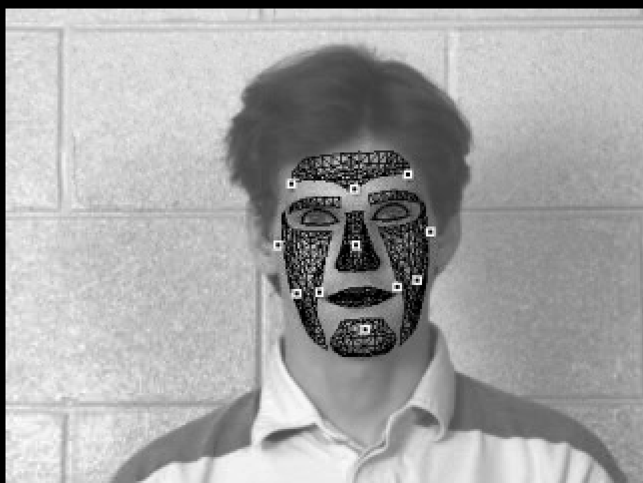
$$\left\| Z + H V \right\|^2$$

$$V = M \theta$$

$$\left\| Z + C \theta \right\|^2$$

Optical Flow + Hand-Crafted Model

SIGGRAPH2006

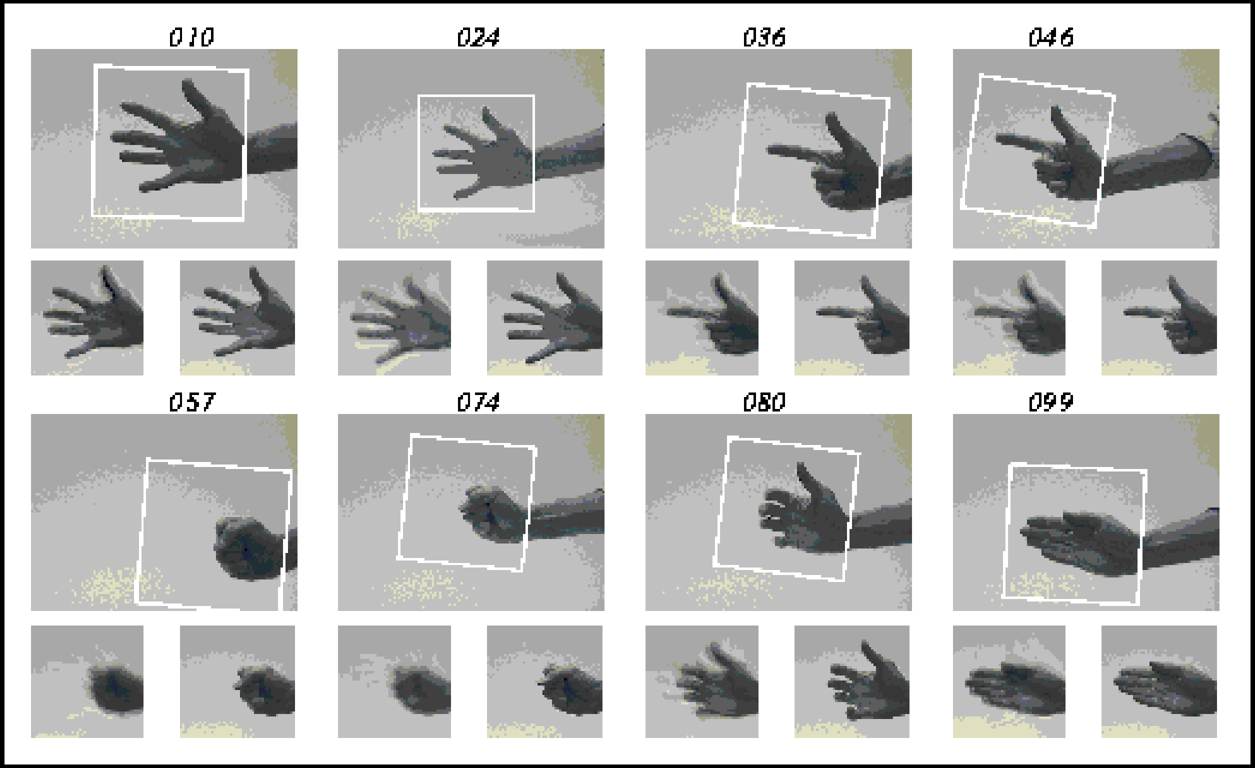


DeCarlo, Metaxas, 1999

Williams et a,l 2002

Optical Flow and PCA

Eigen Tracking
(Black and Jepson)



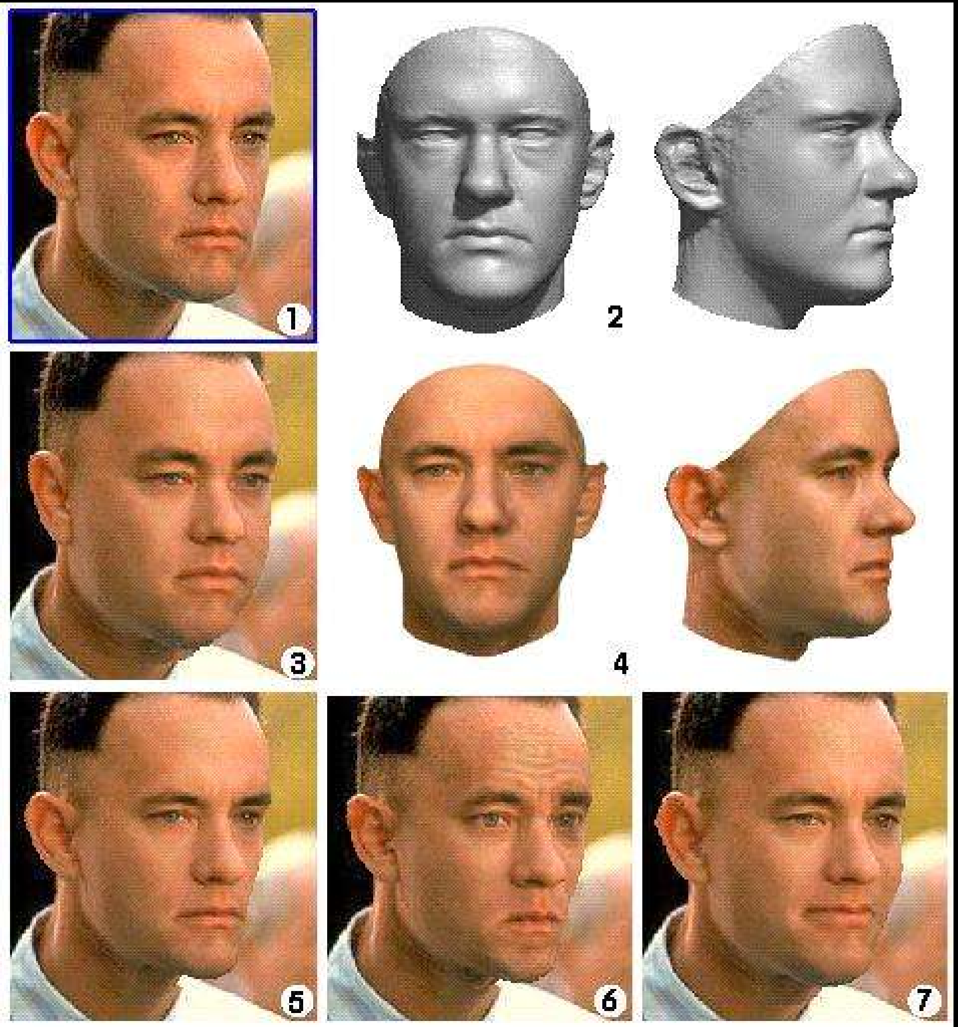
PCA over 2D texture and contours

Active Appearance Models (AAM): (Cootes et al)

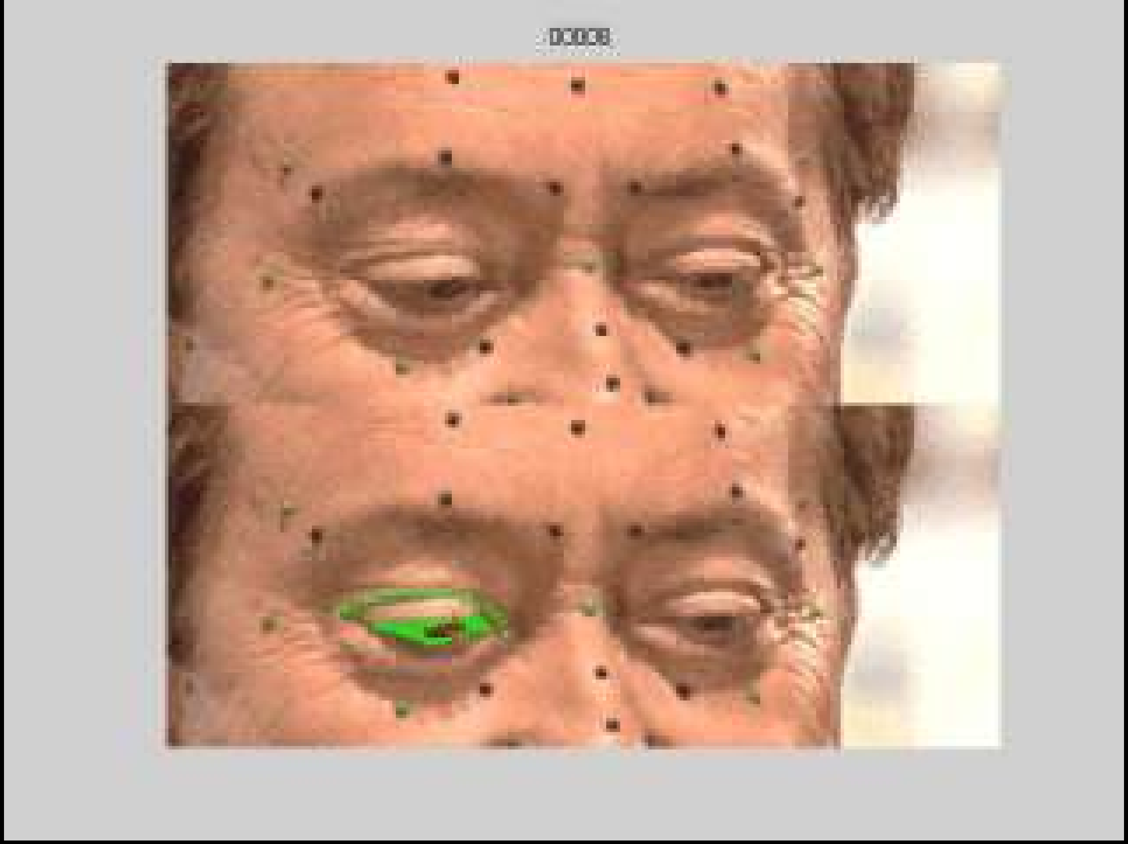
PCA over 2D texture and contours

PCA over texture and 3D shape

3D Morphable Models
(Blanz+Vetter 99)



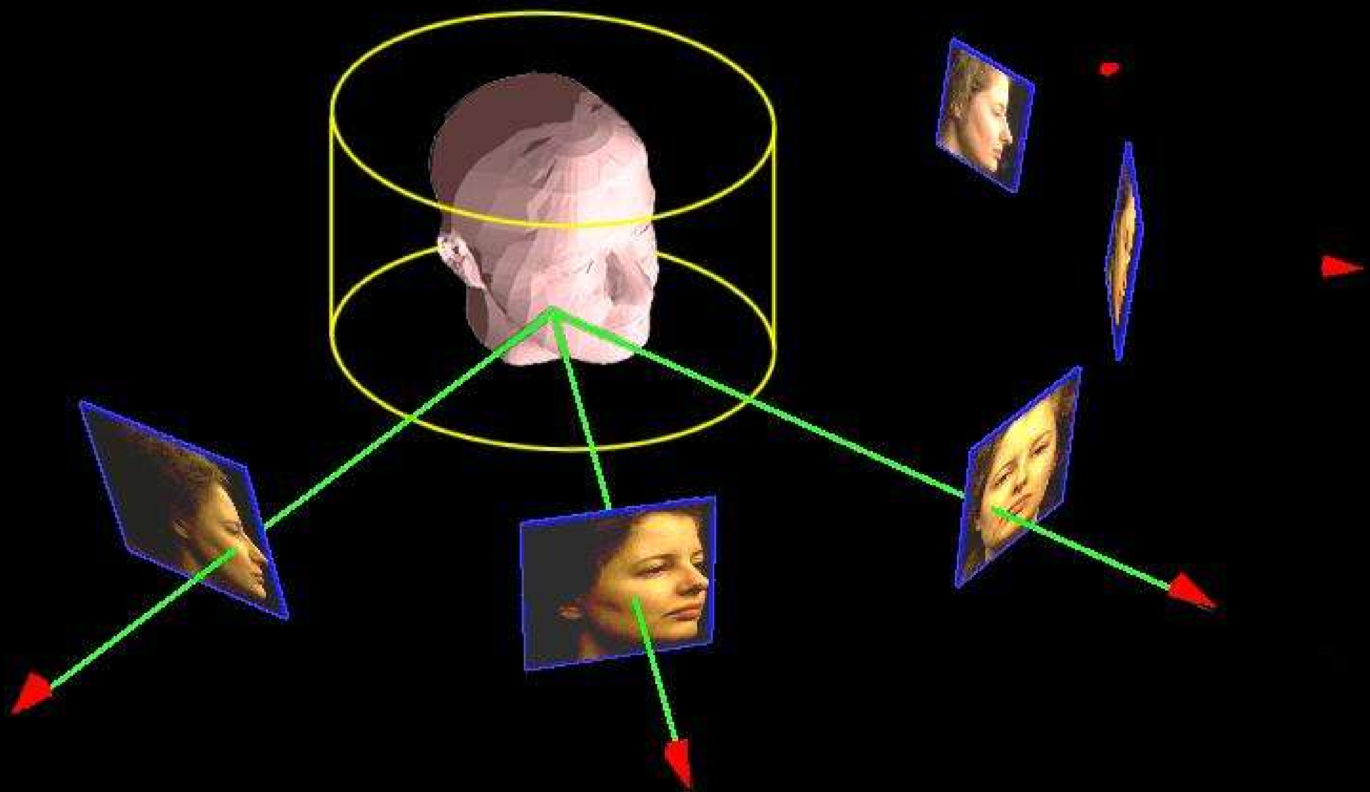
Affine Flow and PCA



3D Model Acquisition

SIGGRAPH2006

- Multi-view input: Pighin et al 98

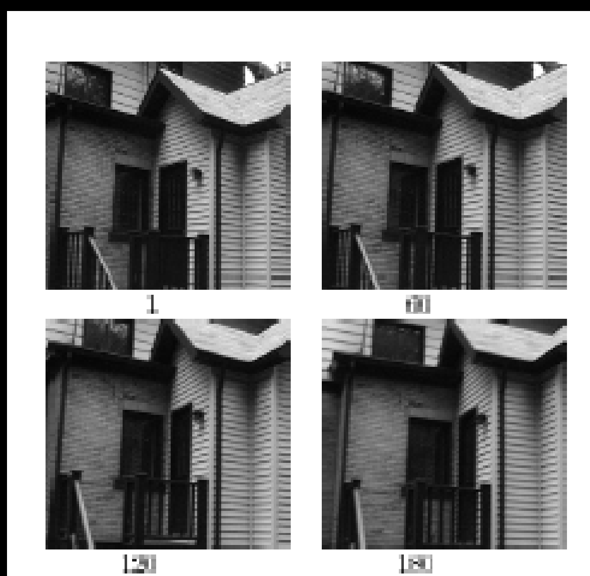


Solution for Rigid 3D Acquisition

SIGGRAPH2006

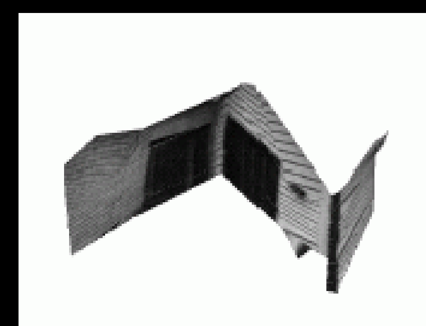
Structure from Motion:

- Tomasi-Kanade-92

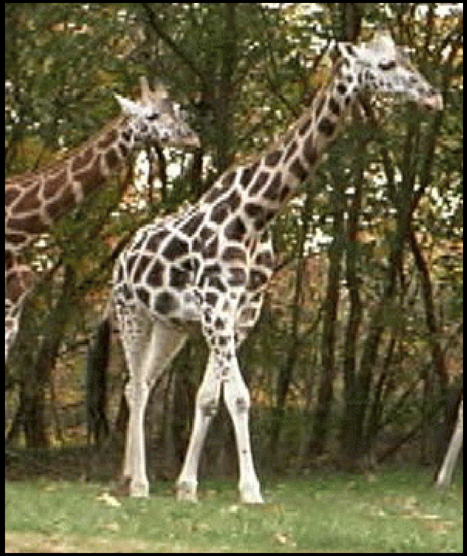


Factorization

3D Pose
3D rigid Object



Acquisition without prior model ?



- No Model available ?
- Model too generic/specific ?
- Stock-Footage only in 2D ?

Solution based on Factorization

- We want 3 things:
  - 3D non-rigid shape model
  - for each frame:
    - 3D Pose
    - non-rigid configuration (deformation)

-> **Tomasi-Kanade-92:**

Rank 3

$$W = P\,S$$

Solution based on Factorization

SIGGRAPH2006

- We want 3 things:
  - 3D **non-rigid shape model**
  - for each frame:
    - 3D Pose
    - **non-rigid configuration (deformation)**

-> **PCA-based representations:**

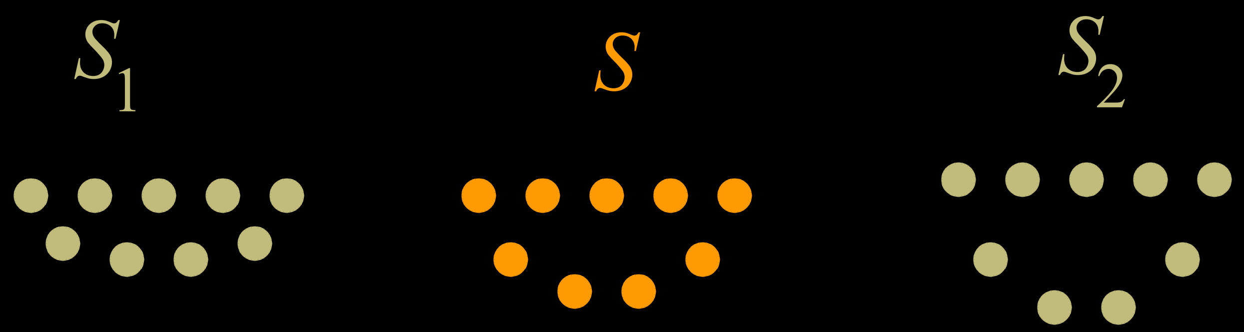
Rank K

$$W = P_{\textit{non-rigid}}\, S$$

3D Shape Model

Linear Interpolation between 3D Key-Shapes:

$$S = \sum_{i=1}^{K} l_i \cdot S_i \qquad S, S_i \in \mathbb{R}^{3 \times P}, l_i \in \mathbb{R} \qquad (1)$$



Basis Shape Factorization

SIGGRAPH2006

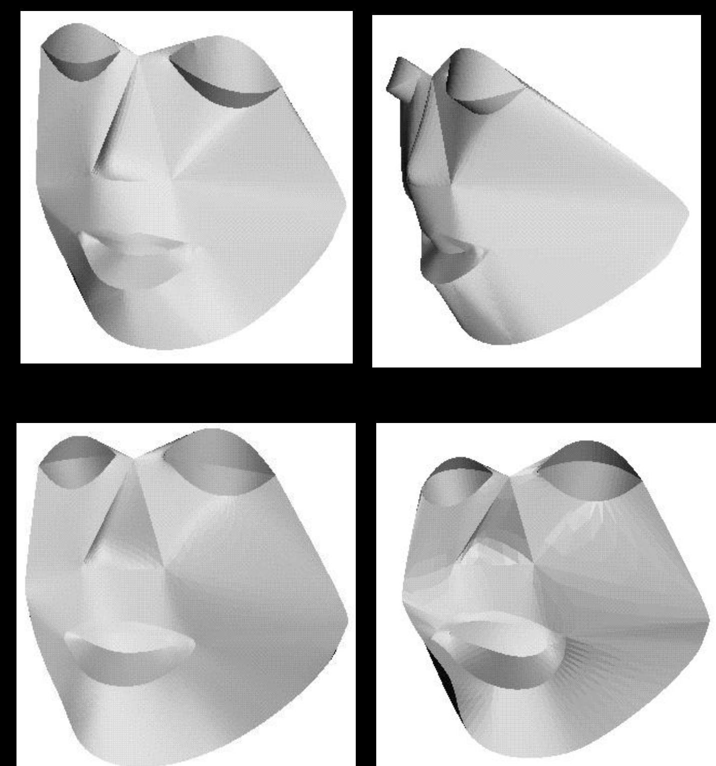
$$W = \begin{bmatrix} u_1^{(1)} & \dots & u_P^{(1)} \\ v_1^{(1)} & \dots & v_P^{(1)} \\ u_1^{(2)} & \dots & u_P^{(2)} \\ v_1^{(2)} & \dots & v_P^{(2)} \\ & \dots & \\ u_1^{(N)} & \dots & u_P^{(N)} \\ v_1^{(N)} & \dots & v_P^{(N)} \end{bmatrix} = \underbrace{\begin{bmatrix} l_1^{(1)} R^{(1)} & \dots & l_K^{(1)} R^{(1)} \\ l_1^{(2)} R^{(2)} & \dots & l_K^{(2)} R^{(2)} \\ & \dots & \\ l_1^{(N)} R^{(N)} & \dots & l_K^{(N)} R^{(N)} \end{bmatrix}}_{Q} \cdot \underbrace{\begin{bmatrix} S_1 \\ S_2 \\ \dots \\ S_K \end{bmatrix}}_{B} \qquad (5)$$

Complete 2D Tracks or Flow

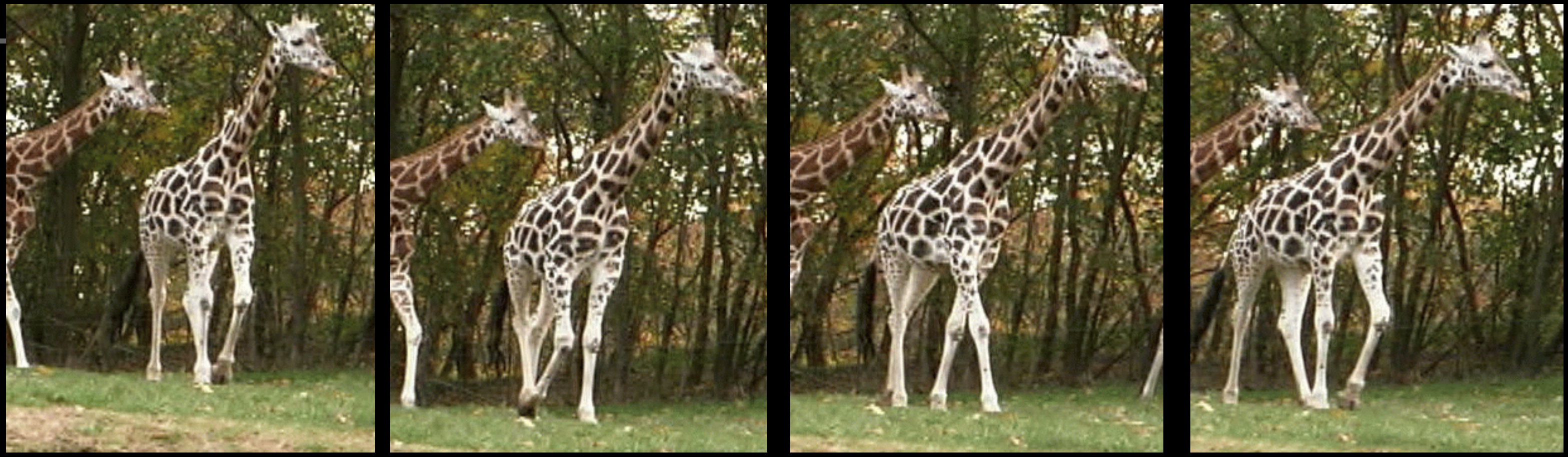
Matrix-Rank <= 3*K

Nonrigid 3D Kinematics from point tracks

$$W = \underbrace{\begin{bmatrix} l_1^{(1)} R^{(1)} & \dots & l_K^{(1)} R^{(1)} \\ l_1^{(2)} R^{(2)} & \dots & l_K^{(2)} R^{(2)} \\ & \dots & \\ l_1^{(N)} R^{(N)} & \dots & l_K^{(N)} R^{(N)} \end{bmatrix}}_{Q} \cdot \underbrace{\begin{bmatrix} S_1 \\ S_2 \\ \dots \\ S_K \end{bmatrix}}_{B} \quad (5)$$



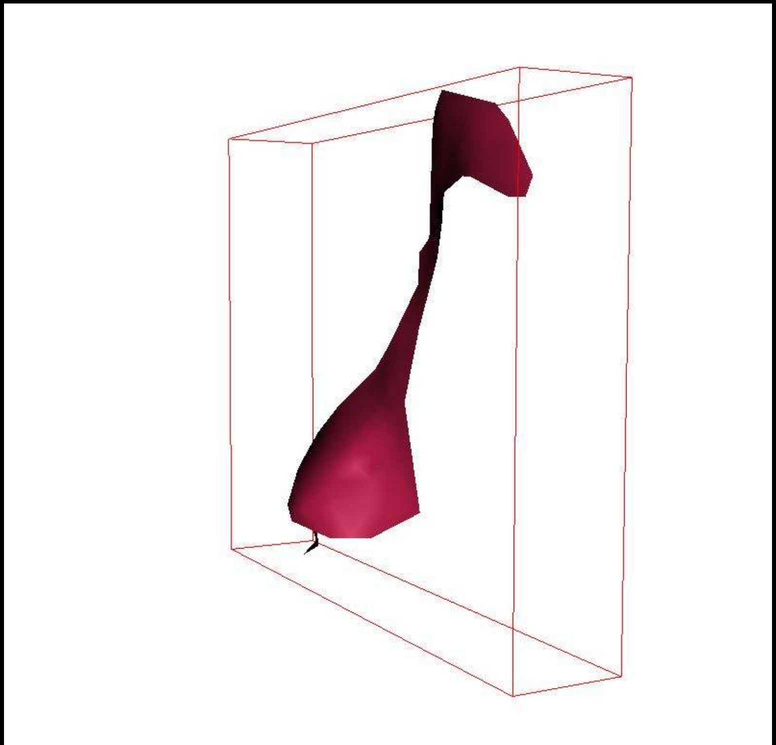
Nonrigid 3D Kinematics from dense flow

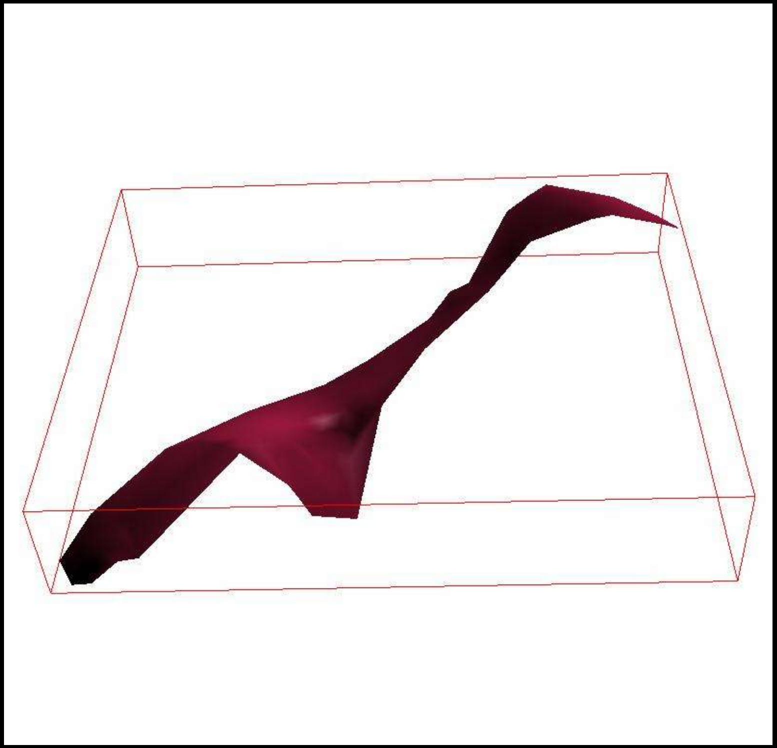


Nonrigid 3D Kinematics from dense flow

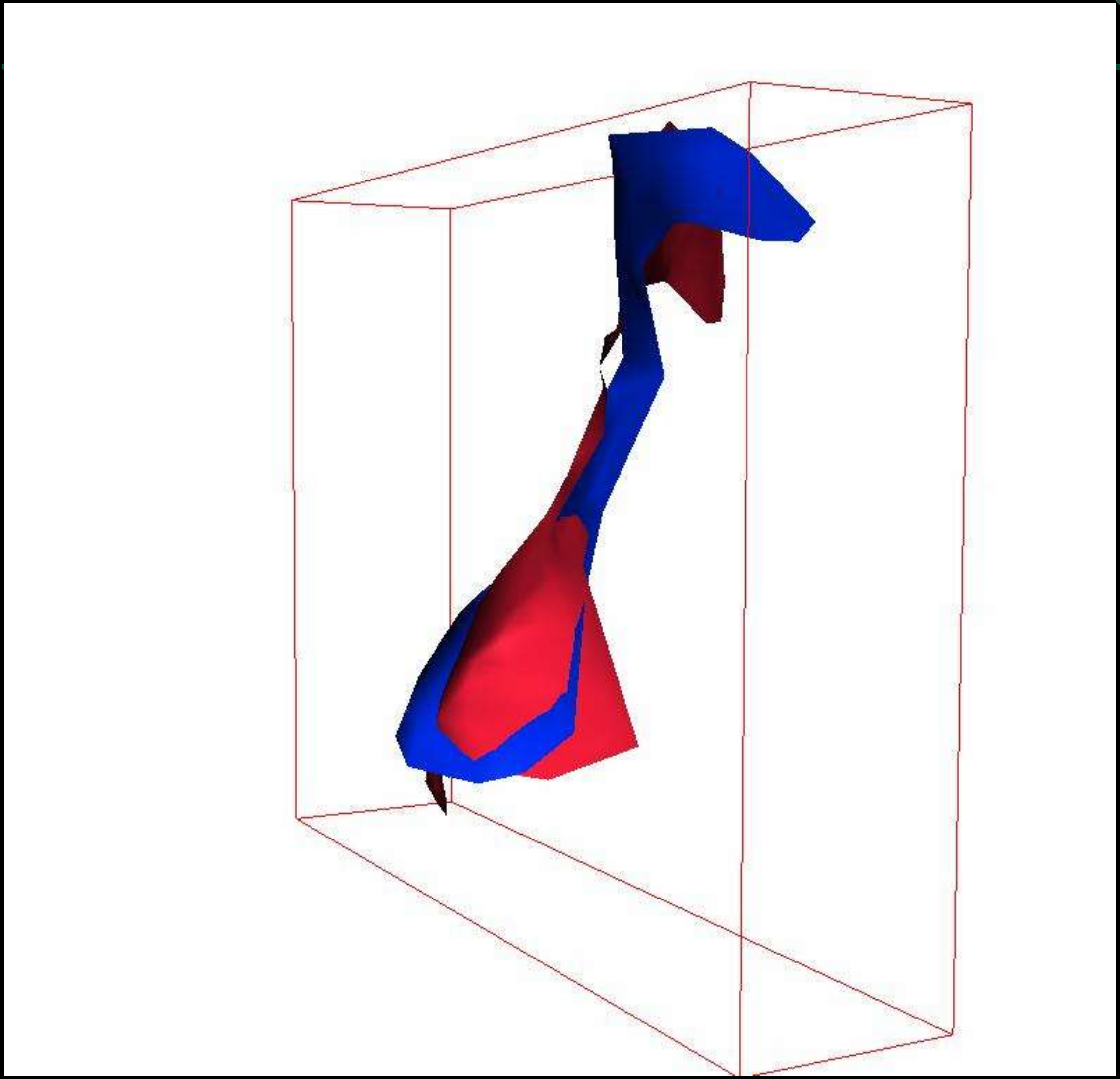
SIGGRAPH2006

–



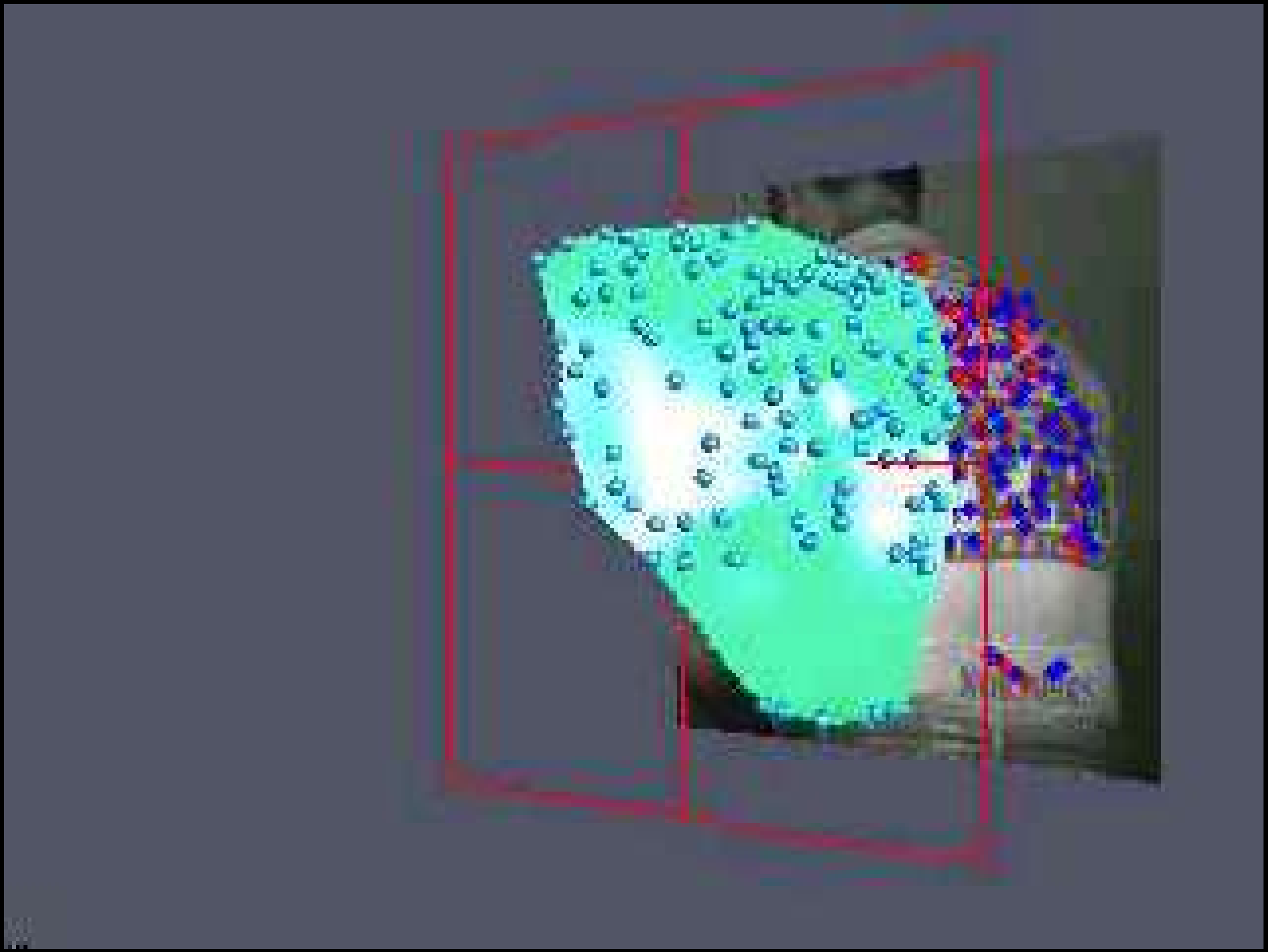


Nonrigid 3D Kinematics from dense flow



-

Nonrigid 3D Kinematics from dense flow



Markerless Face Capture - Summary -

SIGGRAPH2006

- Single / Multi Camera Input
- 2D / 3D Output
- Real-time / Off-line
- Interactive-Refinement / Face Dependent / Independent
- Make-up / Natural
- Flow / Contour / Texture / Local / Global Features
- Hand Crafted / Data Driven
- Linear / Nonlinear Models / Tracking

SIGGRAPH Course 30:
Performance-Driven Facial Animation

Section:

Markerless Face Capture and Automatic Model Construction

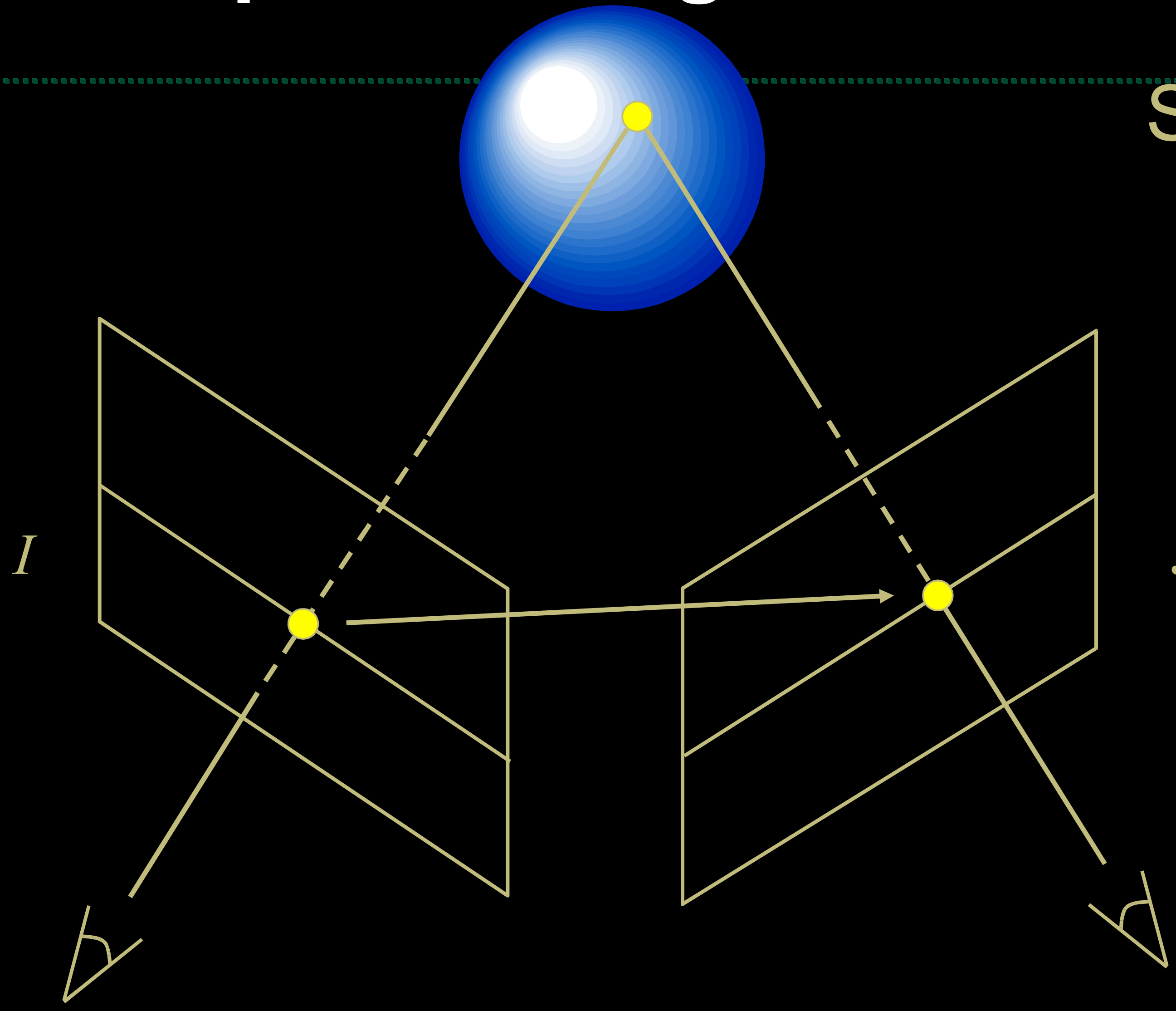
Part 2: Li Zhang, Columbia University

Outline

1. Scanning face models
   - Triangulation methods
   - Non triangulation methods
2. Dense facial motion capture
   - Marker based capture
   - Template fitting for face scans

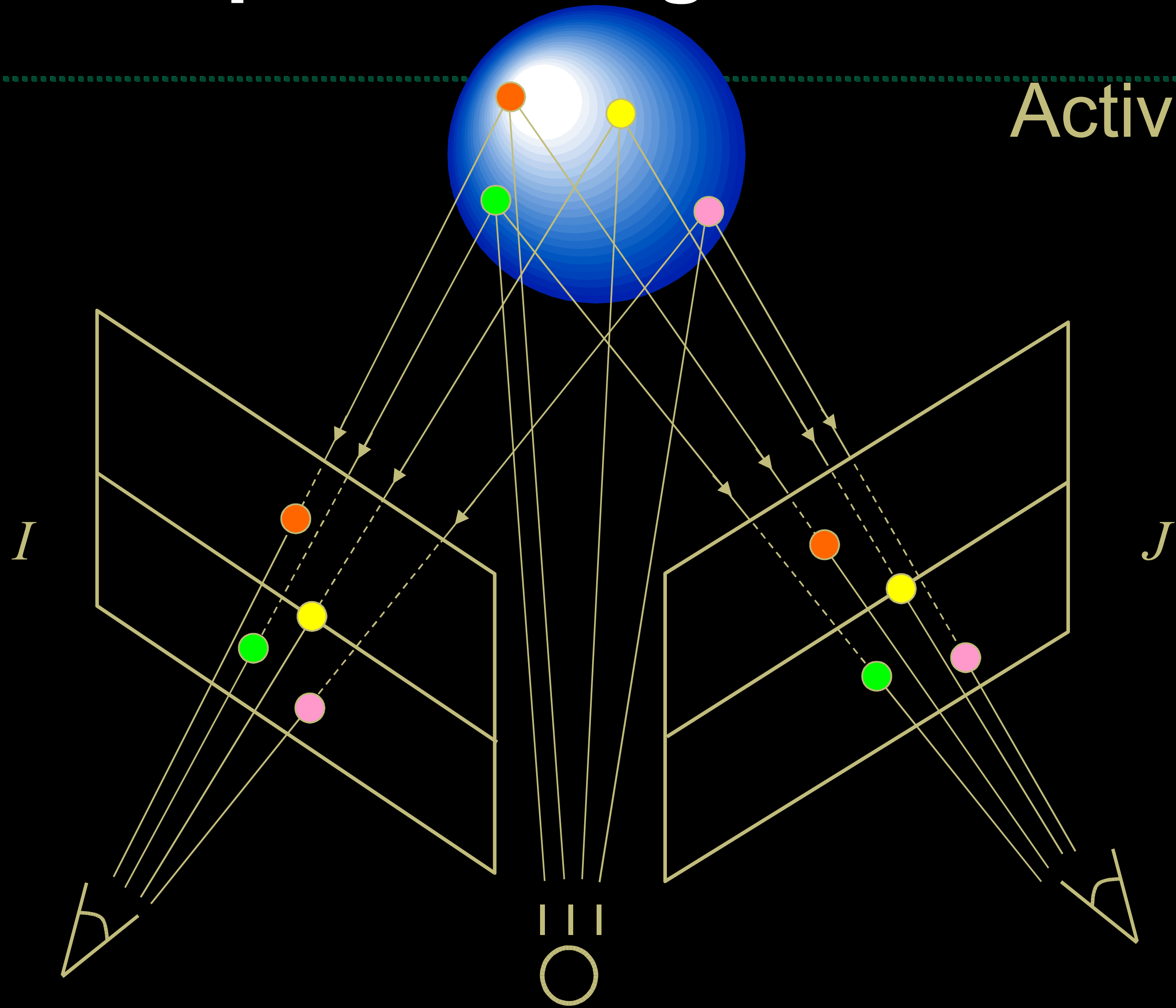
Principle 1: triangulation

Stereo



Principle 1: triangulation

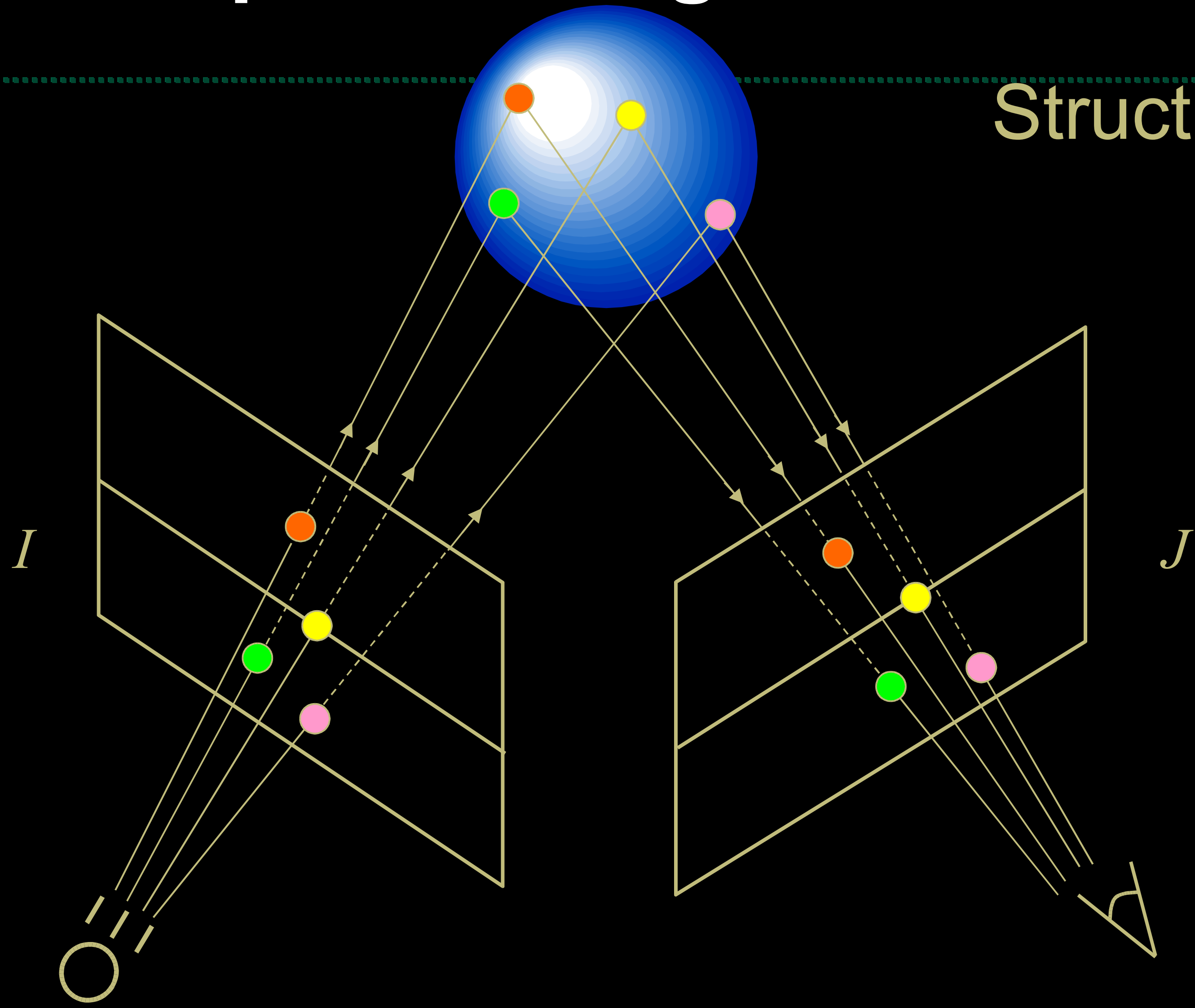
SIGGRAPH2006



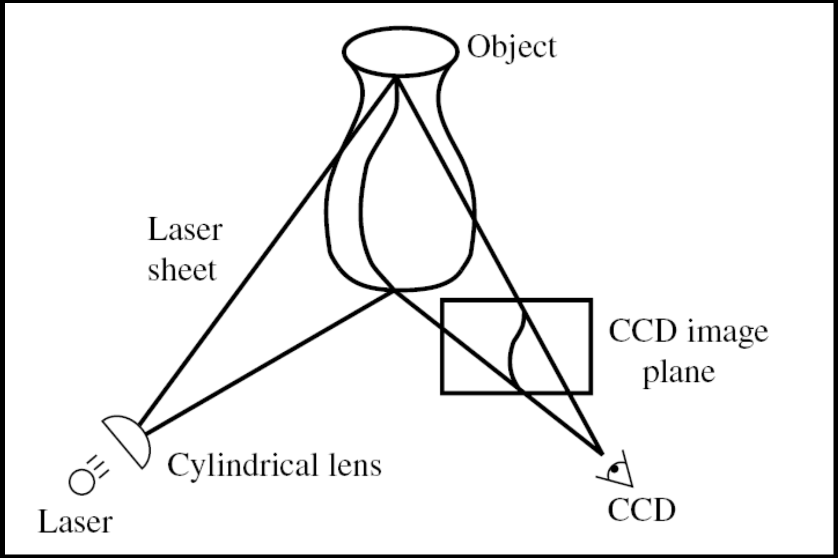
Principle 1: triangulation

SIGGRAPH2006

Structured light



Laser scanner

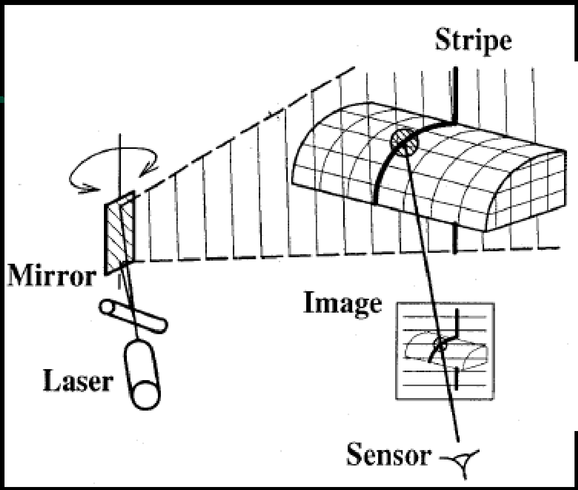


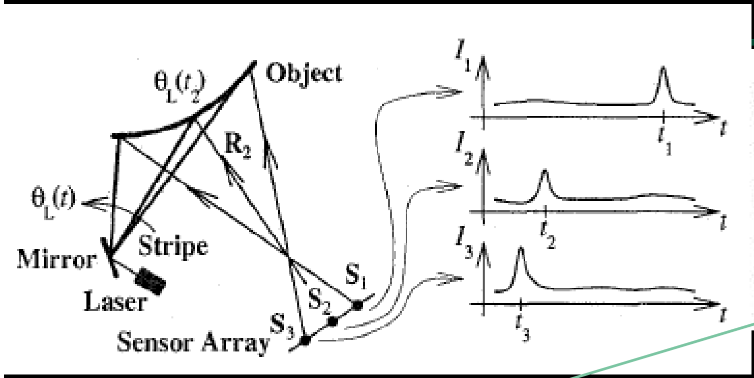
+ very accurate <0.01mm
− >10sec per scan

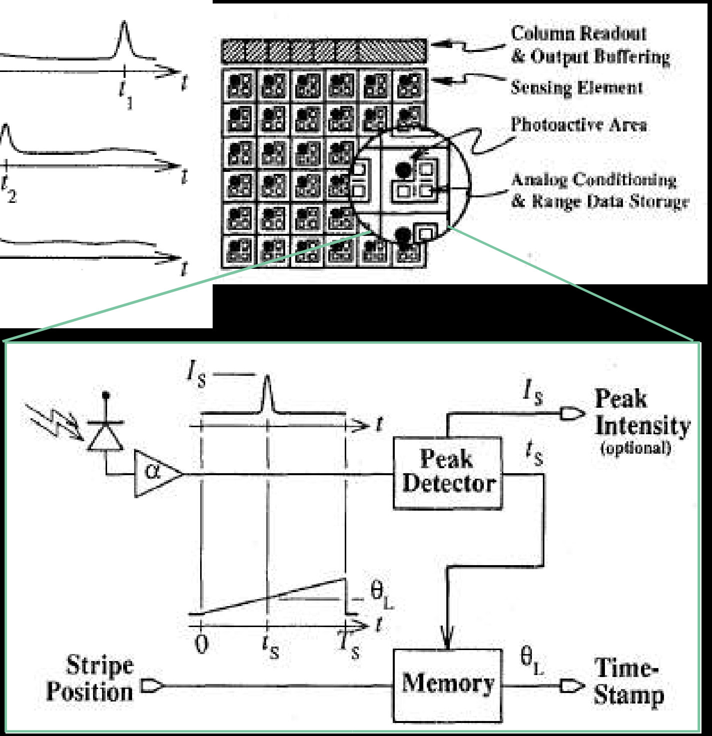
Cyberware® face and head scanner

Fast laser scanner (temporal)

SIGGRAPH2006







+ Fast – up to 1000Hz

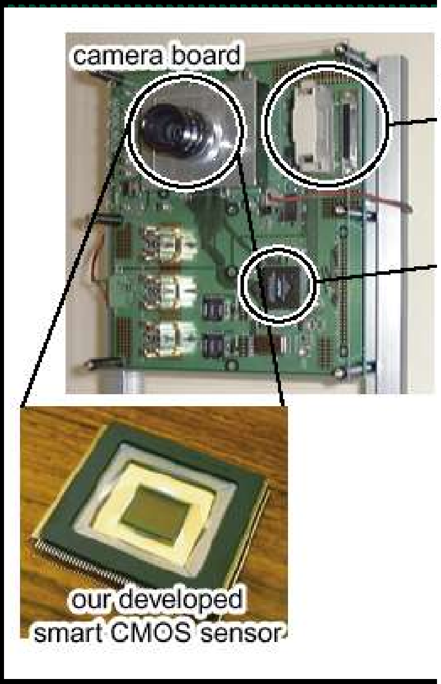
− Customized device

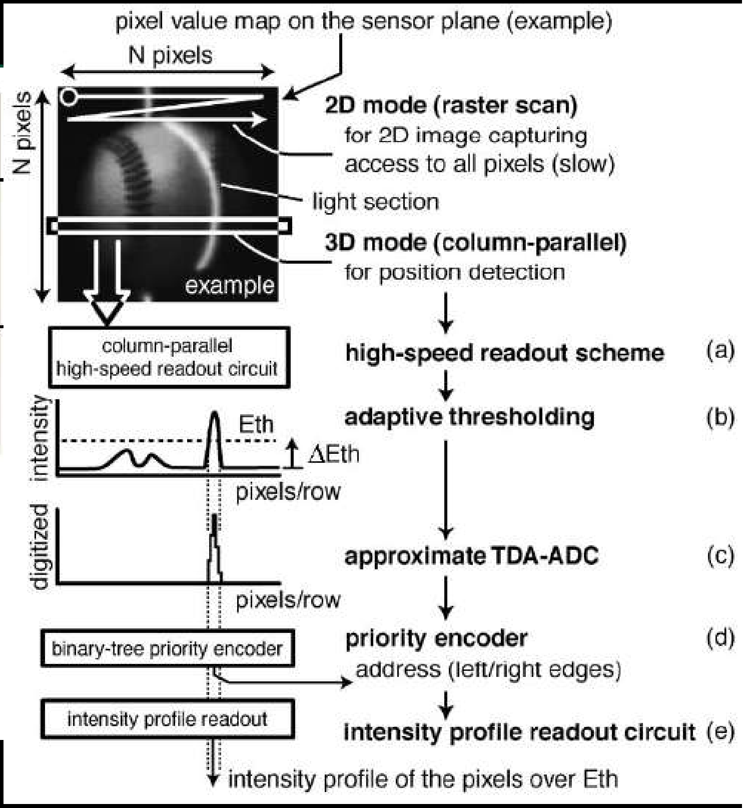
A. Gruss, S. Tada, and T. Kanade "A VLSI Smart Sensor for Fast Range Imaging," ICIRS 1992

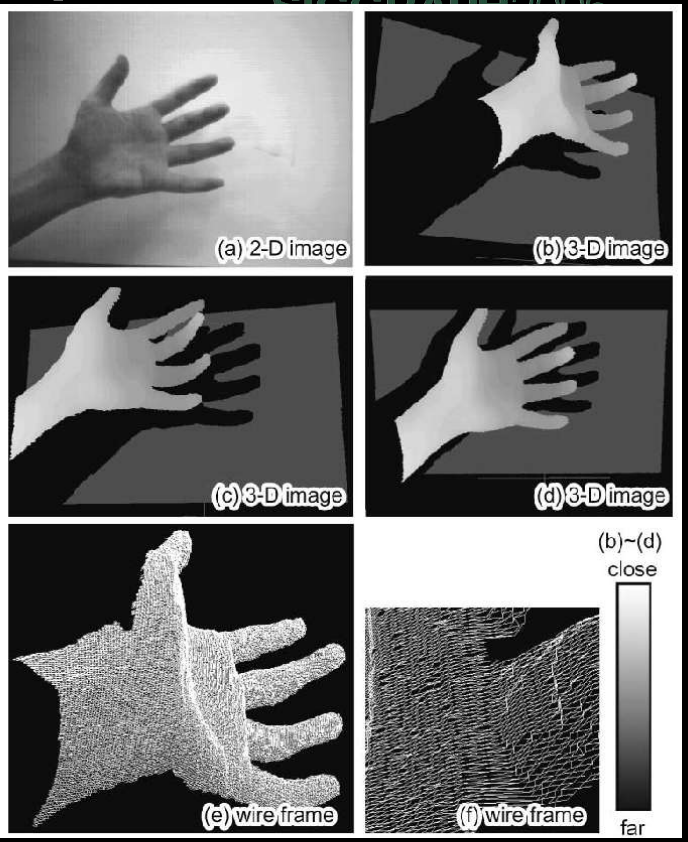
Working Volume: 350-500mm - Accuracy: 0.1%

Spatial Resolution: 28x32 - Speed: 1000Hz

Fast laser scanner (spatial)







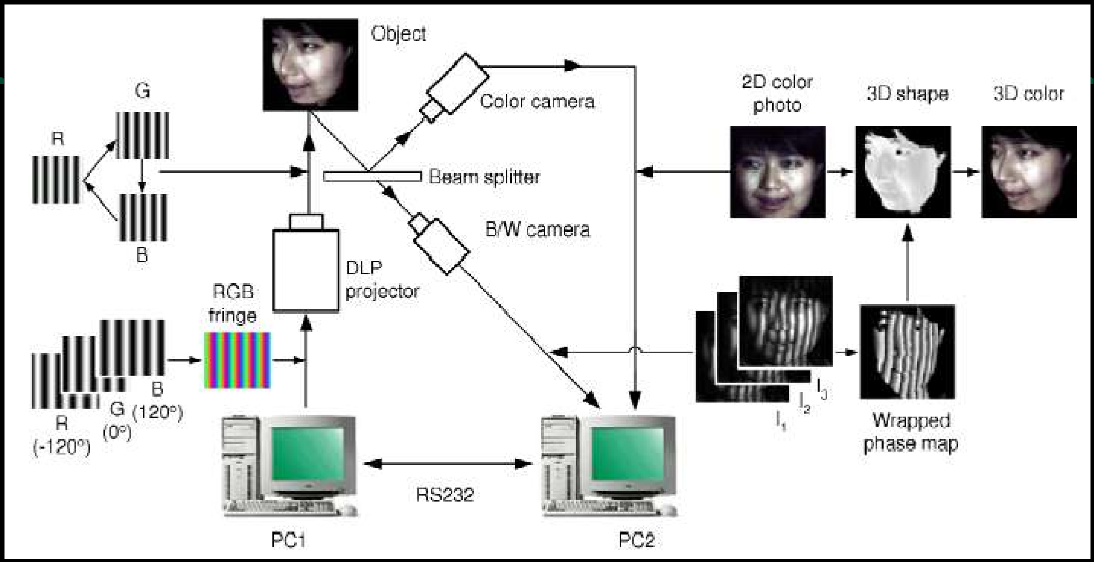
Possible issue: Stripes within a range map are not simultaneously measured.

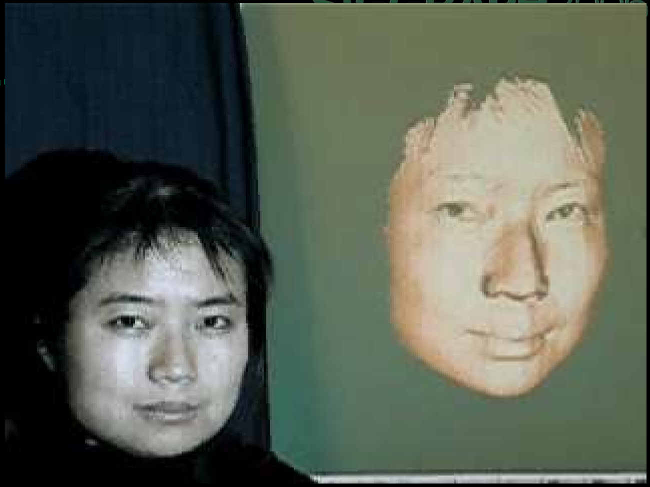
Oike, Y. Ikeda, M. Asada, K., “Design and implementation of real-time 3-D image sensor with 640x480 pixel resolution”, IEEE Journal of Solid-State Circuits, 2004.

Working Volume: 1200mm - Accuracy: 0.07%

Spatial Resolution: 640x480 - Speed: 65Hz

Digital fringe range sensor





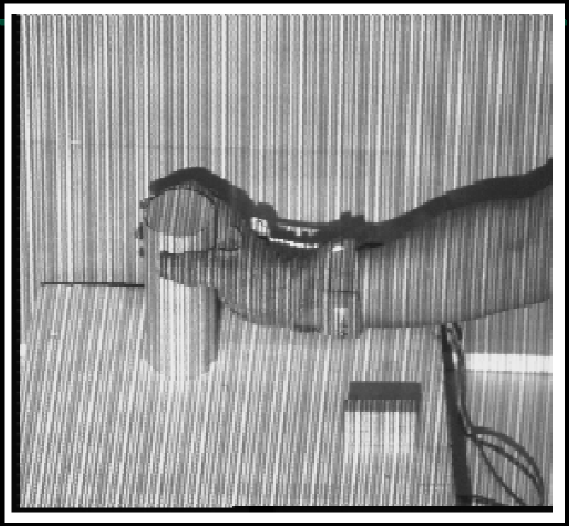
+ Real time performance
− Phase ambiguity near discontinuities
− Customized device
− Capture from one viewpoint at a time

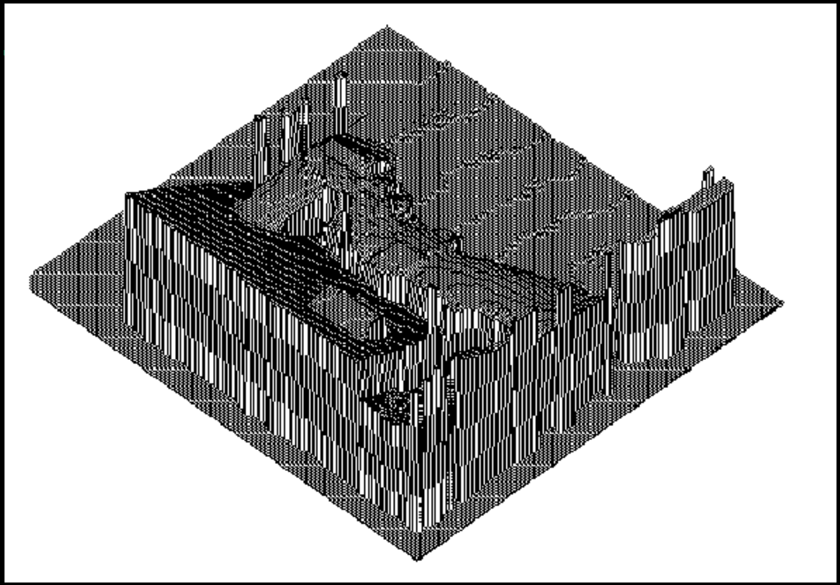
P. Huang, C. Zhang, F. Chiang, "High-speed 3-D shape measurement based on digital fringe projection", Journal of Optical Engineering, 2003

Working Volume: 10-2000mm - Accuracy: 0.025%

Spatial Resolution: 532x500 - Speed: 120Hz

Active multi-baseline stereo





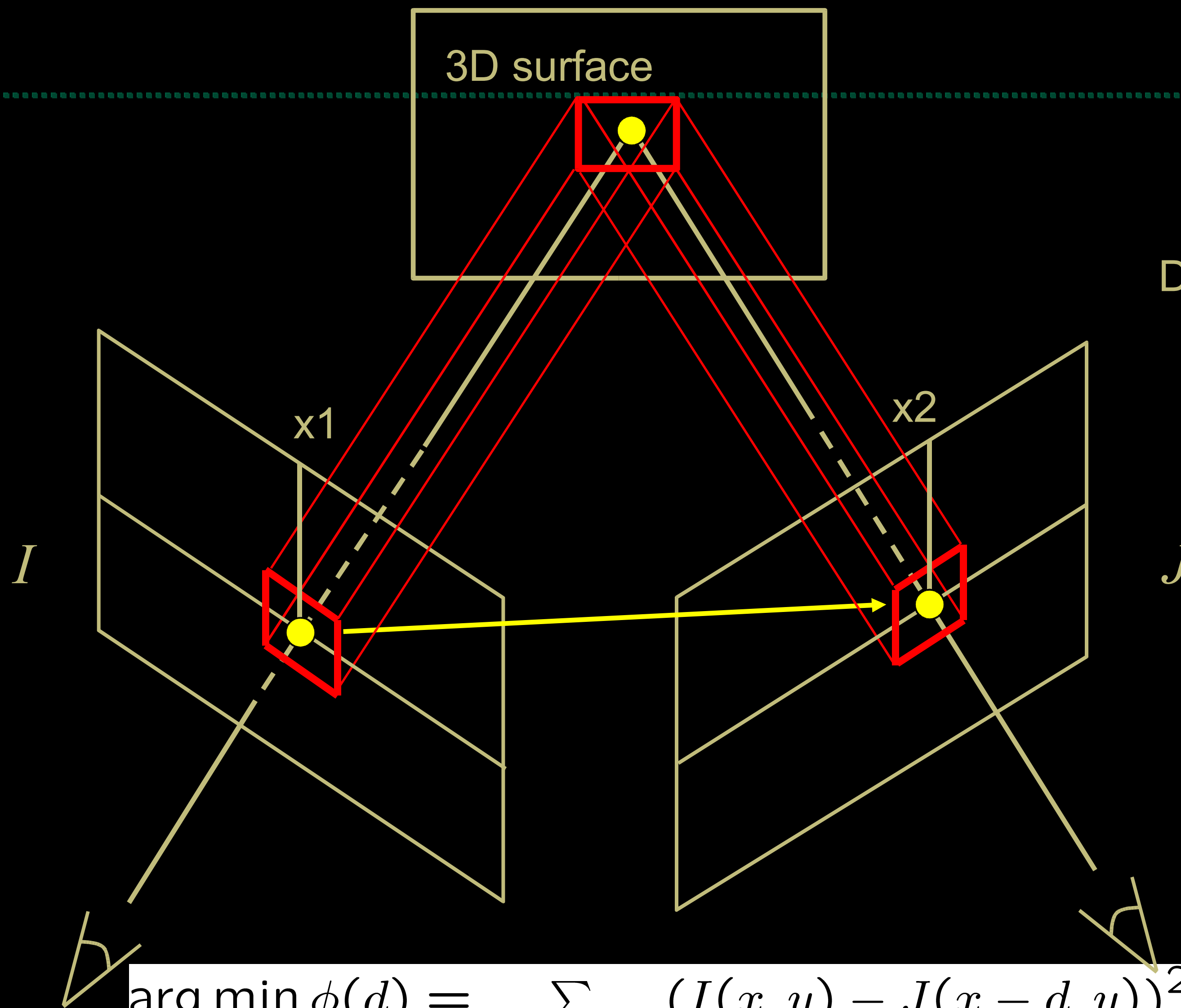
+ Only require one image per camera
+ Simultaneous multi-view capture
− Less accurate than laser scanners or fringe scanners

S. Kang, J.A. Webb, C. Zitnick, and T. Kanade, "A Multibaseline Stereo System with Active Illumination and Real-time Image Acquisition," ICCV 1995.

Working Volume: 2000mm - Accuracy: 0.1%

Spatial Resolution: 100x100? - Speed: 30Hz

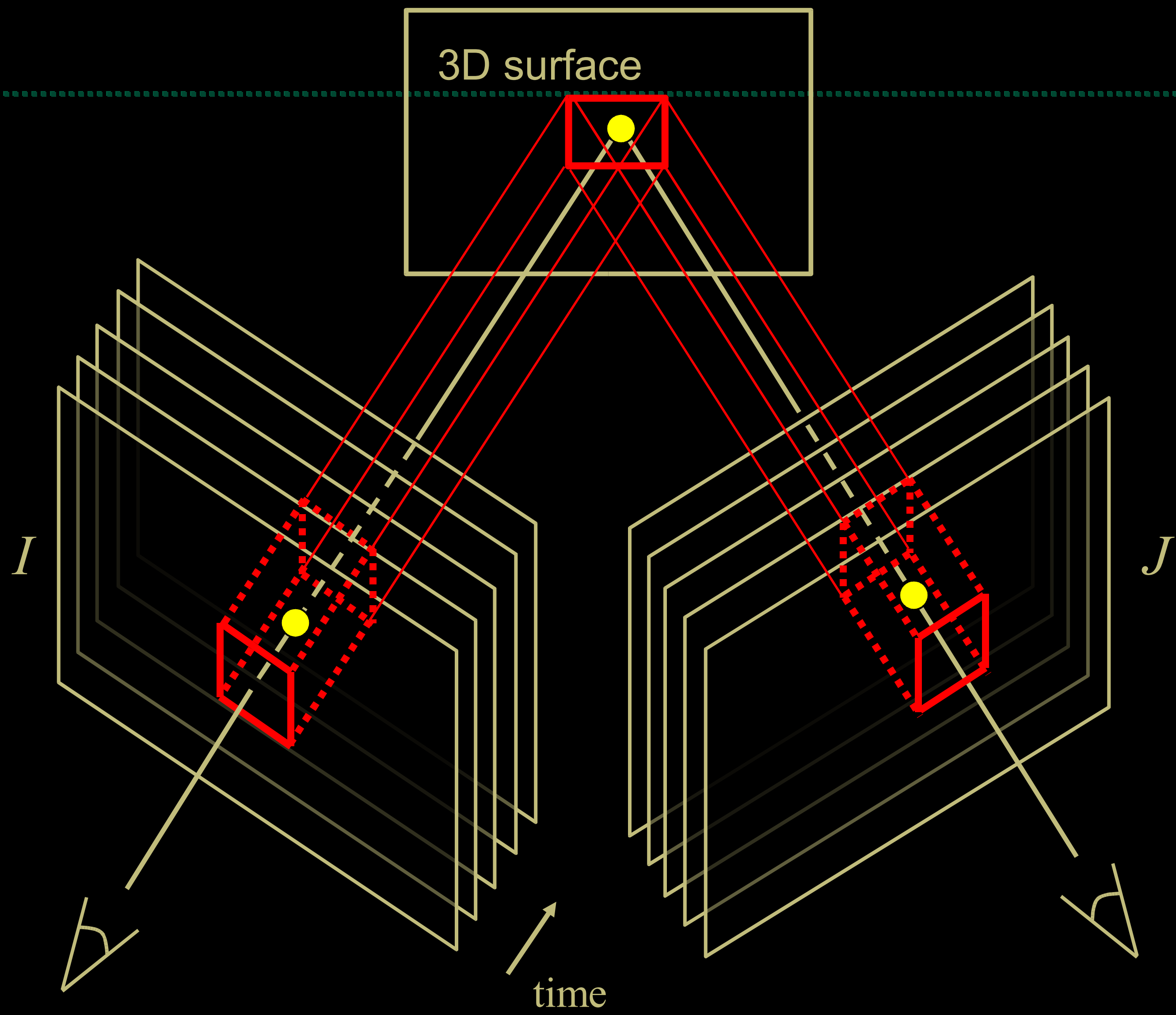
Spacetime stereo



Disparity: d = x1 – x2

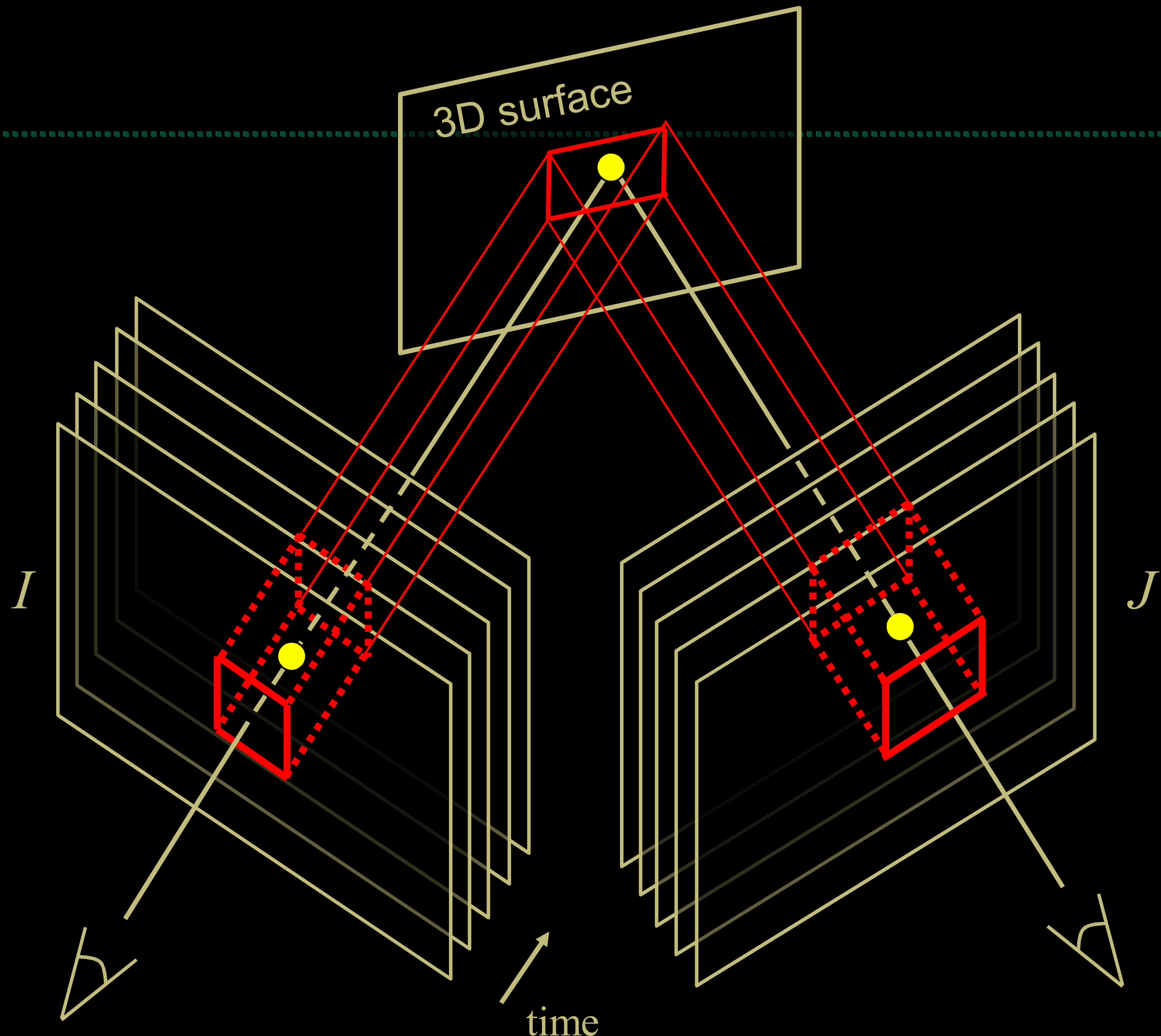
$$\arg\min_{d} \phi(d) = \sum_{(x,y)\in W} \left(I(x,y) - J(x-d,y)\right)^2$$

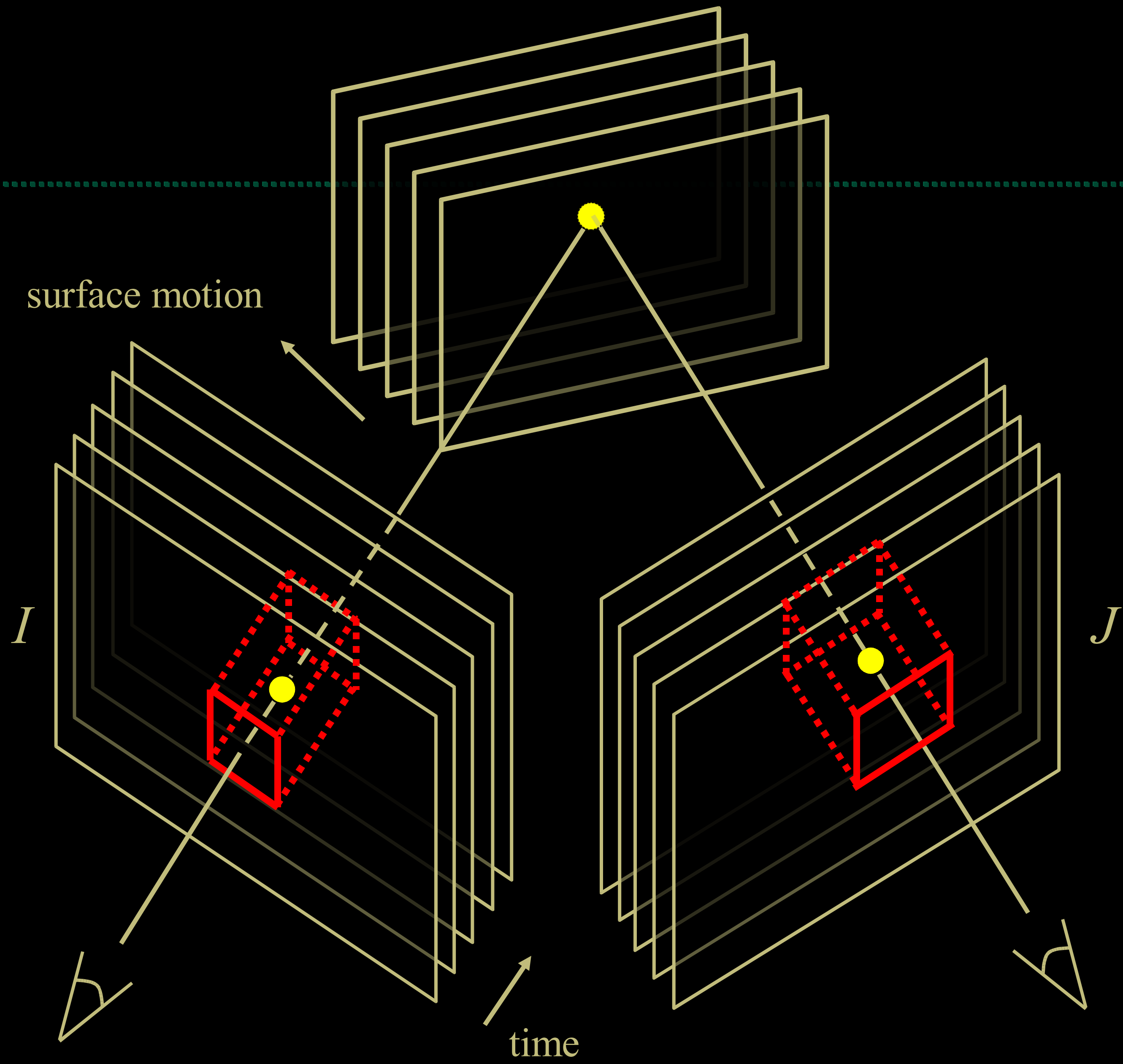
Spacetime stereo



Spacetime stereo

SIGGRAPH2006

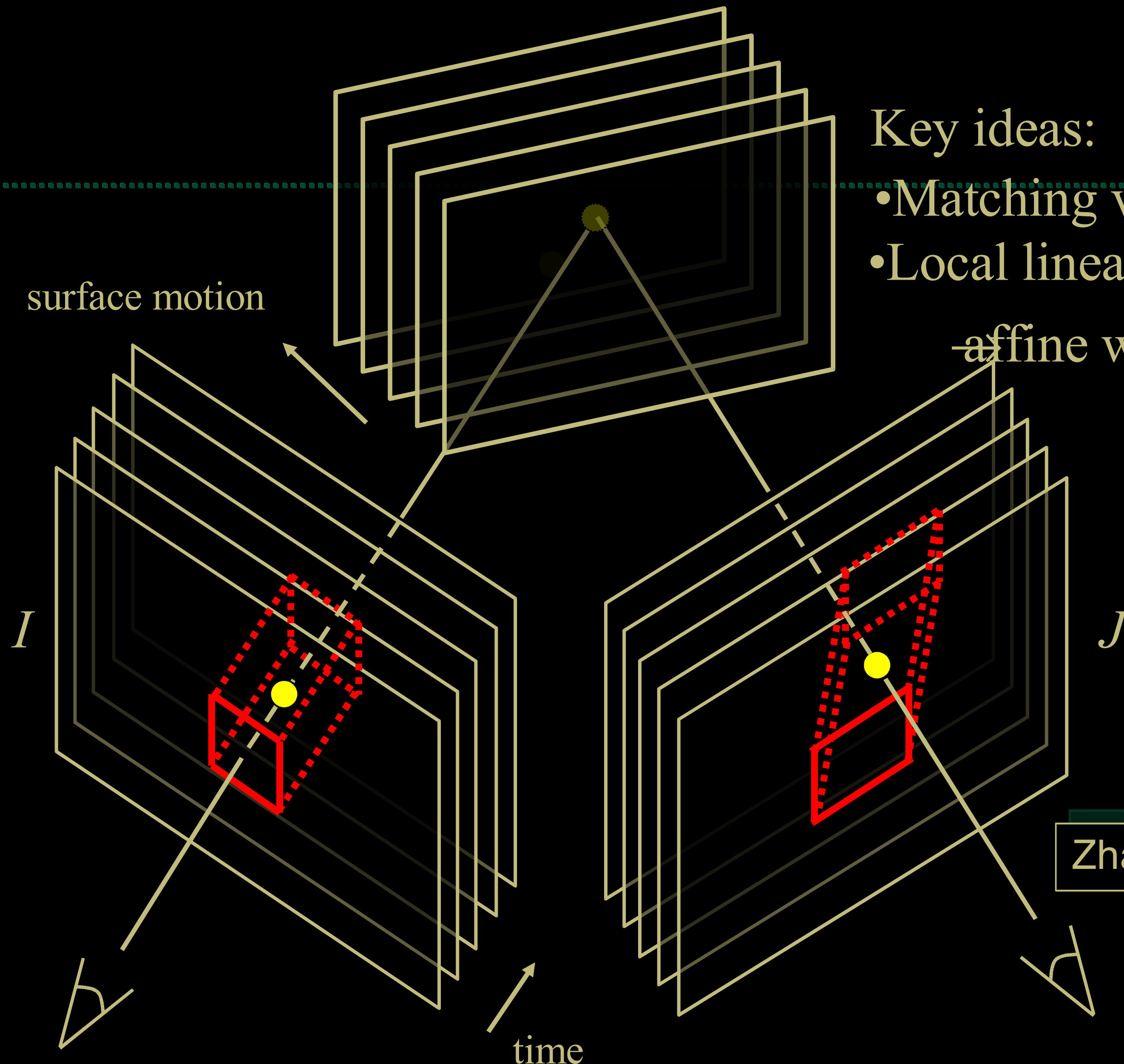




SIGGRAPH2006

Key ideas:

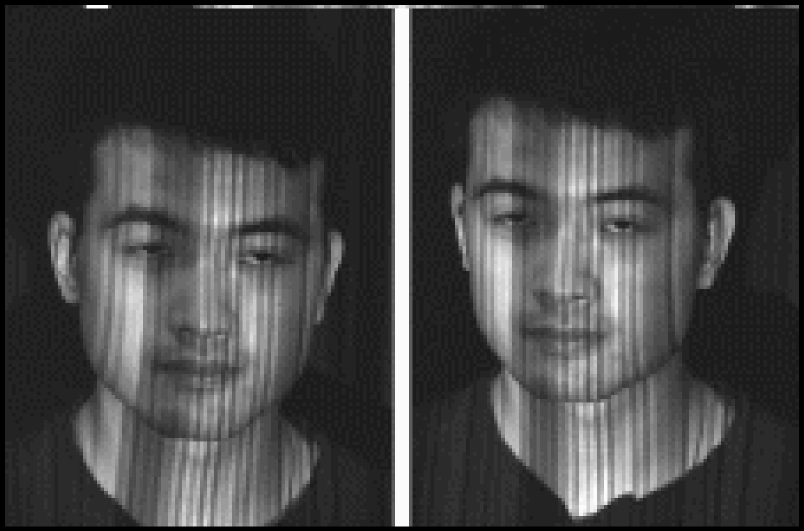
- Matching volumetric window
- Local linear disparity change
  - → affine window warp



Zhang et al. CVPR 2003

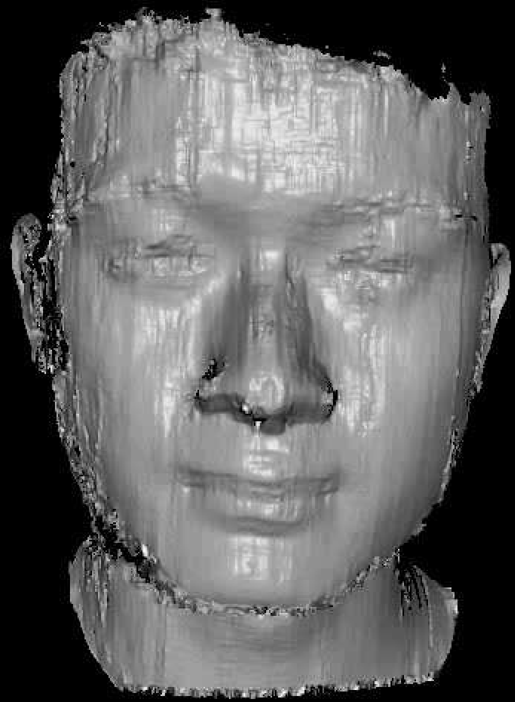
Spacetime stereo

Input stereo video:



656x494x60fps videos captured by firewire cameras

Face Example: Result Comparison



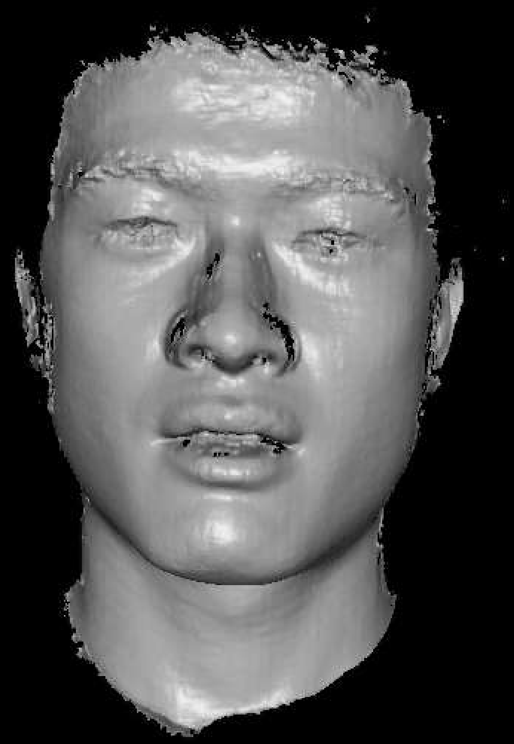
Frame-by-frame stereo

WxH=15x15 window

Spacetime stereo

WxHxT=9x5x5 window

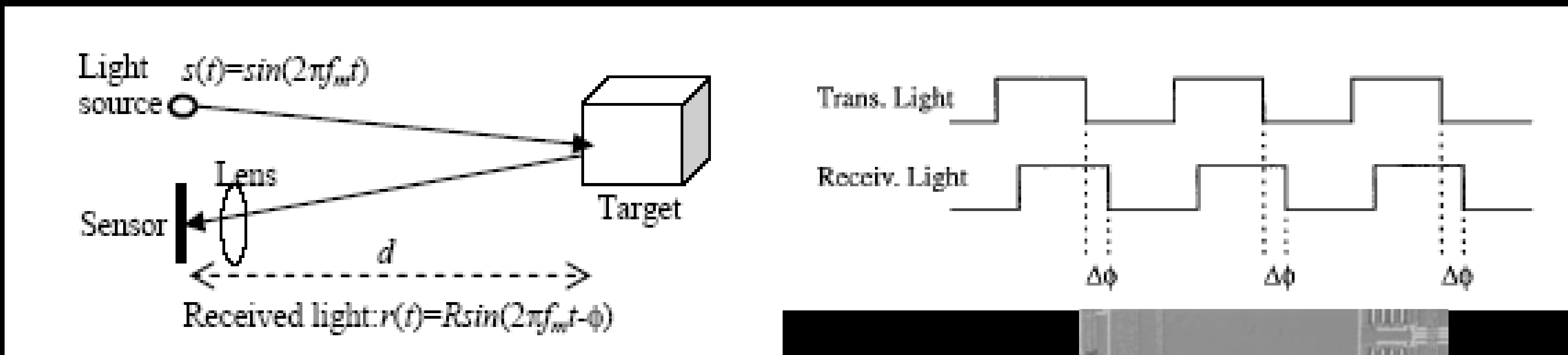
Face Example: Mouth motion

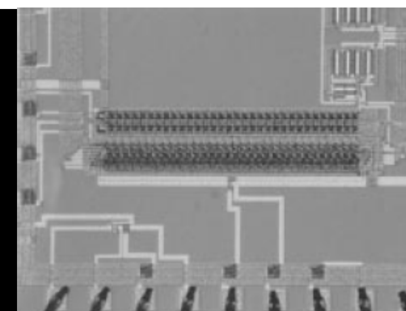


+ More accurate and stable than frame by frame stereo
+ Simultaneous multi-view capture
− Offline computation (3min per frame)

Zhang, L., Curless, B., Seitz, S., “Spacetime stereo”, CVPR 2003,
Working Volume: 300mm - Accuracy: 0.1%
Spatial Resolution: 640x480- Speed: 60Hz

Principle 2: Time-of-flight





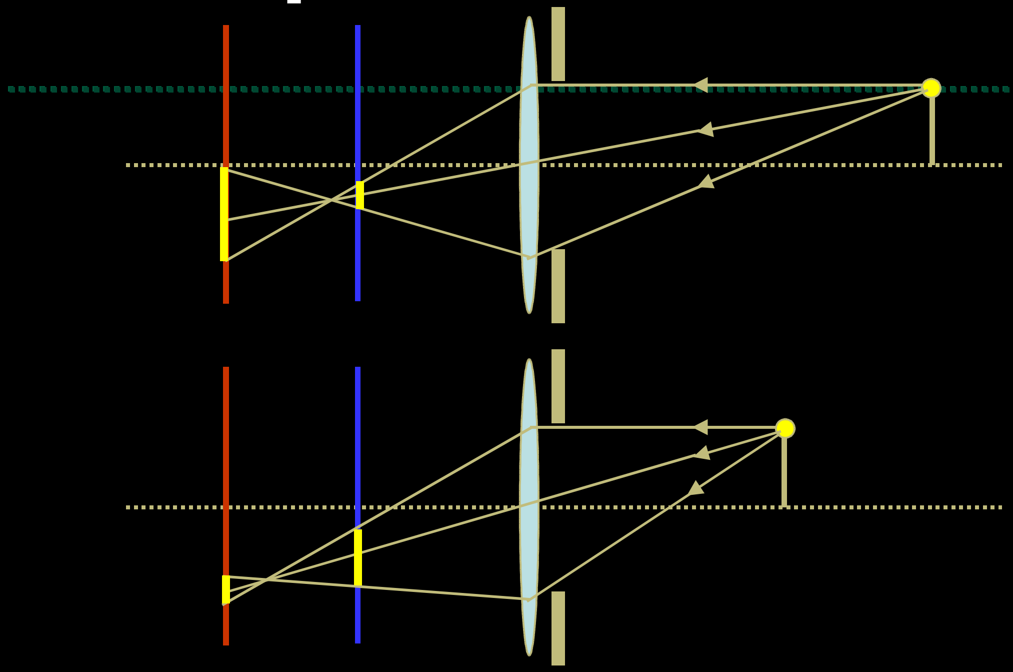
+ No baseline, no parallax shadows
+ Mechanical alignment is not as critical
− Low depth accuracy
− Single viewpoint capture

Miyagawa, R., Kanade, T., “CCD-Based Range Finding Sensor”, IEEE Transactions on Electron Devices, 1997

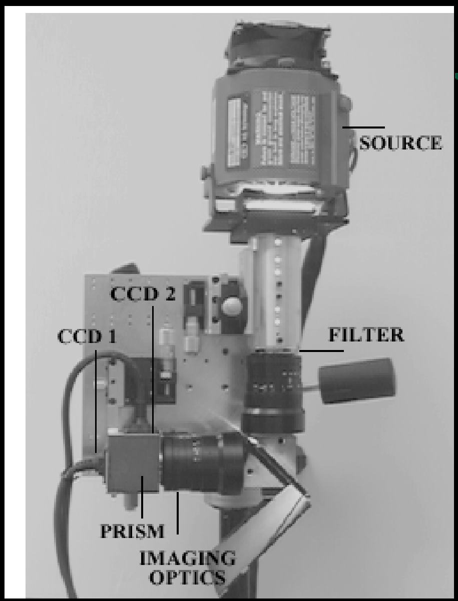
Working Volume: 1500mm - Accuracy: 7%

Spatial Resolution: 1x32- Speed: ??

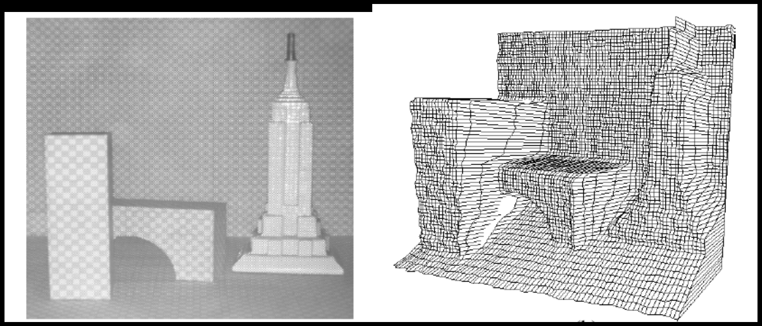
Principle 3: Defocus



Principle 3: Defocus



+ Hi resolution and accuracy, real-time
− Customized hardware
− Single view capture?



Nayar, S.K., Watanabe, M., Noguchi, M., "Real-Time Focus Range Sensor", ICCV 1995

Working Volume: 300mm - Accuracy: 0.2%

Spatial Resolution: 512x480 - Speed: 30Hz

Commercial products

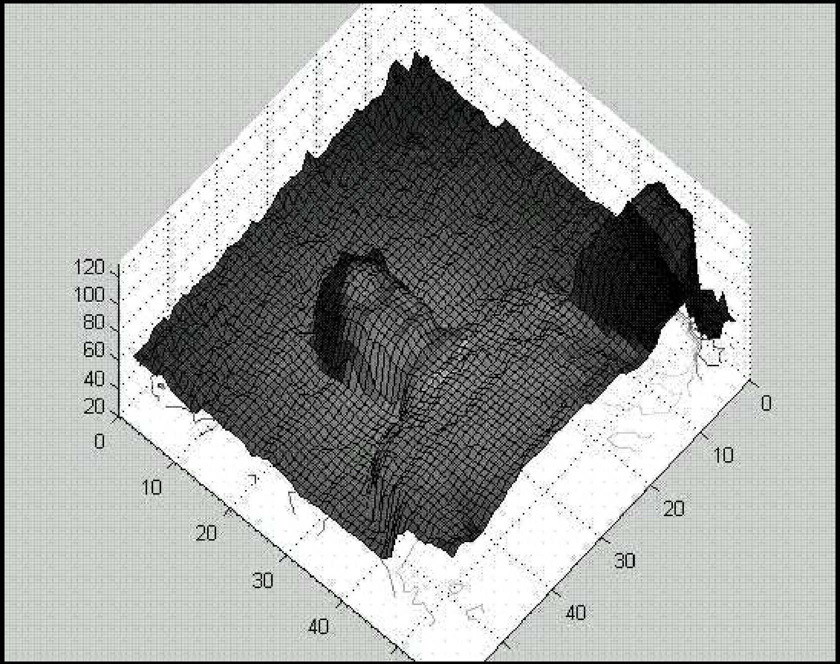
Company	Working principle	XY resolution	Depth accuracy	Speed
Cyberware	Laser	>500x500	0.01mm	>10sec per scan
XYZRGB	Laser	Very high	0.01mm	>10sec per scan
Eyetronics	Structrued light	High	<2mm	<0.1sec
3Q	Active stereo	High	?	<0.1sec
3DV	Time of flight	720x486	1-2cm	30Hz
Canesta	Time of flight	64x64	1cm	30Hz

Comercial products

Canesta

64x64@30hz
Accuracy 1-2cm

Not accurate enough for face modeling, but good enough for layer extraction.

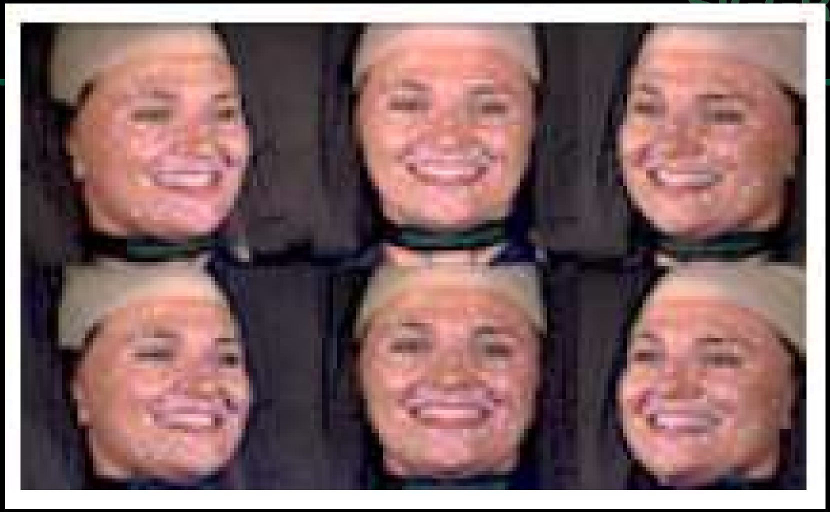


Outline

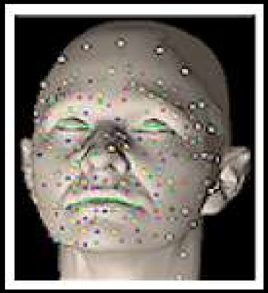
1. Scanning face models
   - Triangulation methods (created most accurate face models)
   - Non triangulation methods
2. Dense facial motion capture
   - Marker based capture
   - Template fitting for face scans

Marker based approach

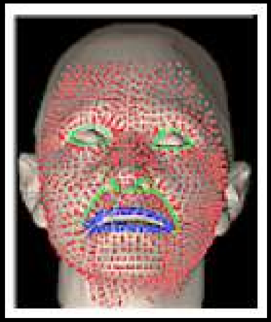
182 colored dots on a face



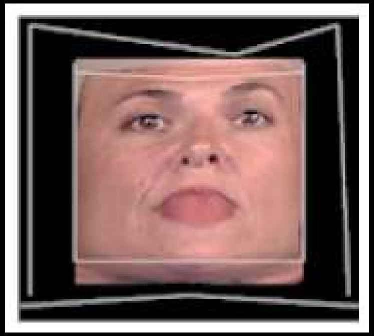
6 cameras videotaping performance



3D dot motion



deforming face model



Dot removal for texturemap

Guenter et al SIGGRAPH 1998

Making faces

SIGGRAPH2006

+ Realistic appearance
− Limited geometry details
− The overhead of painting faces

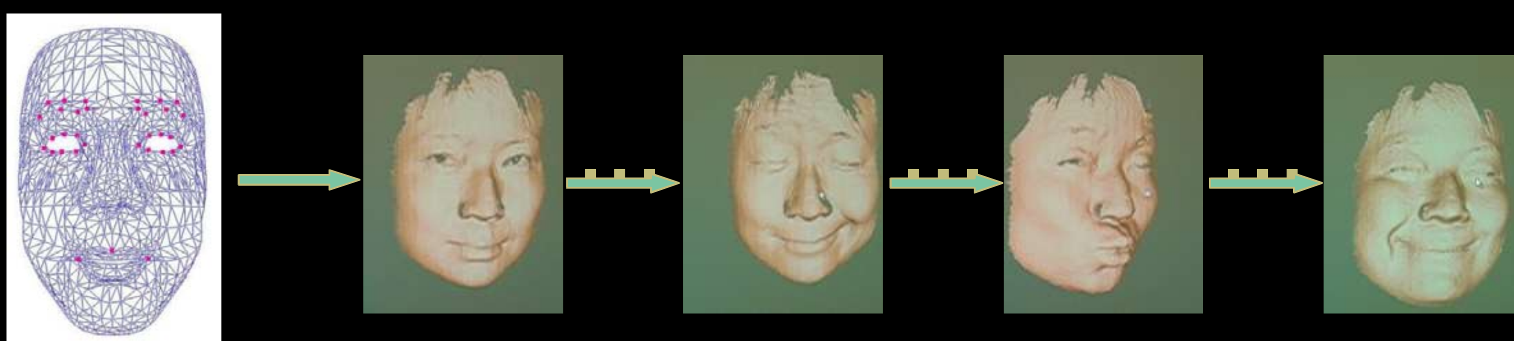
Guenter et al SIGGRAPH 1998

High Resolution Acquisition of Dynamic 3-D expression

SIGGRAPH2006

Problem: estimating 3D motion between shape measurement

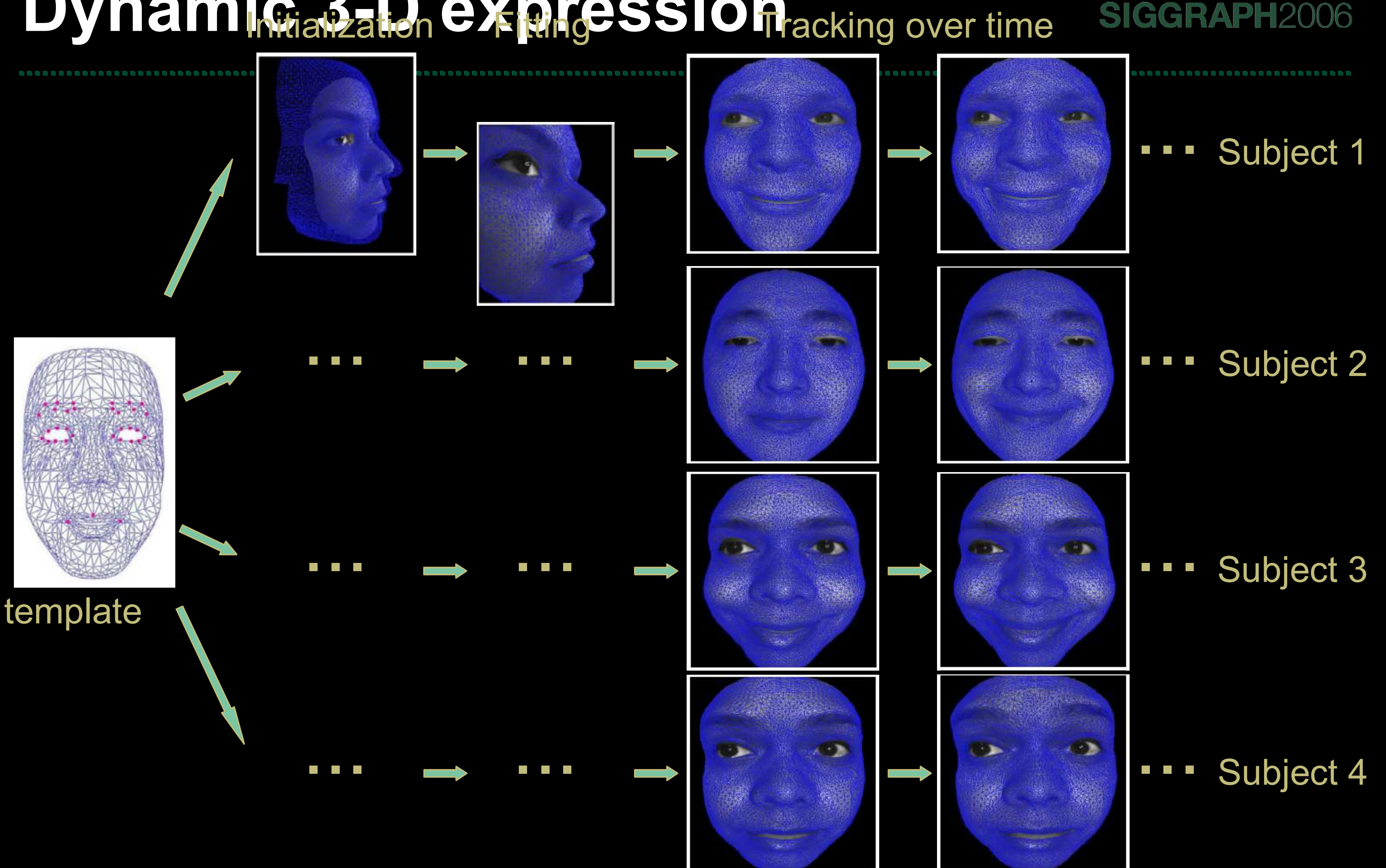
Approach: template fitting



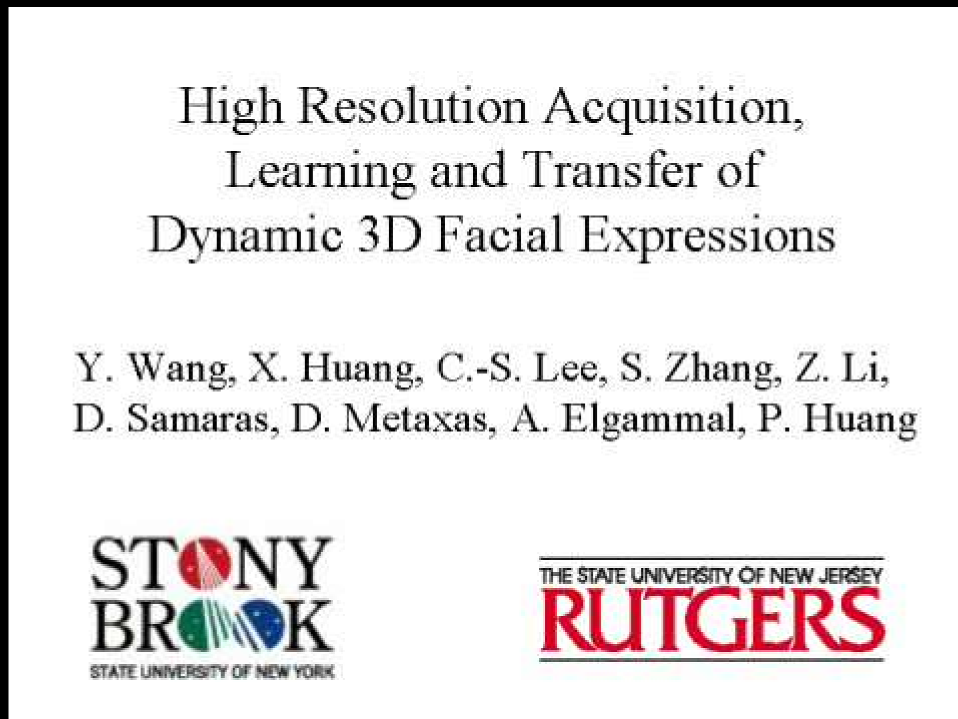
template

Wang et al Eurographics 2004

High Resolution Acquisition of Dynamic 3-D expression



High Resolution Acquisition of Dynamic 3-D expression


High Resolution Acquisition, Learning and Transfer of Dynamic 3D Facial Expressions

Y. Wang, X. Huang, C.-S. Lee, S. Zhang, Z. Li, D. Samaras, D. Metaxas, A. Elgammal, P. Huang

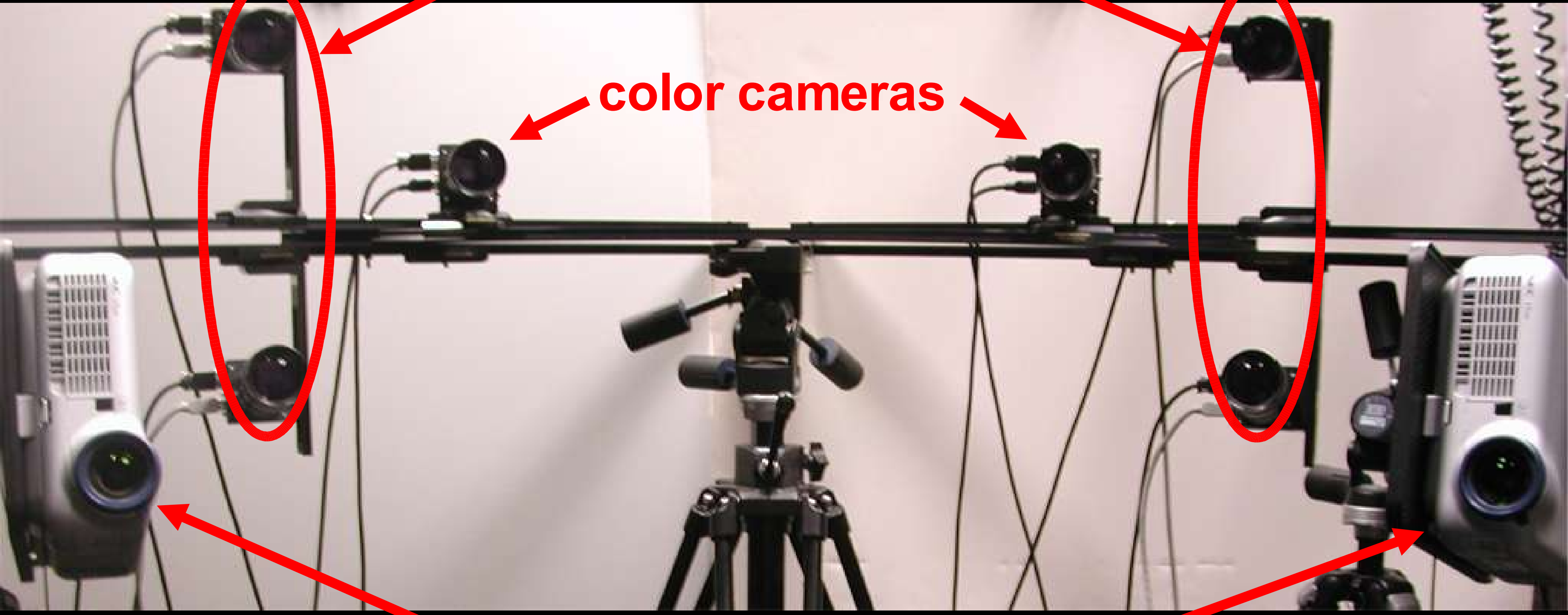
STONY BROOK STATE UNIVERSITY OF NEW YORK

THE STATE UNIVERSITY OF NEW JERSEY RUTGERS

+ High resolution motion
− less robust for larger inter-frame deformation

Wang et al Eurographics 2004

Spacetime faces

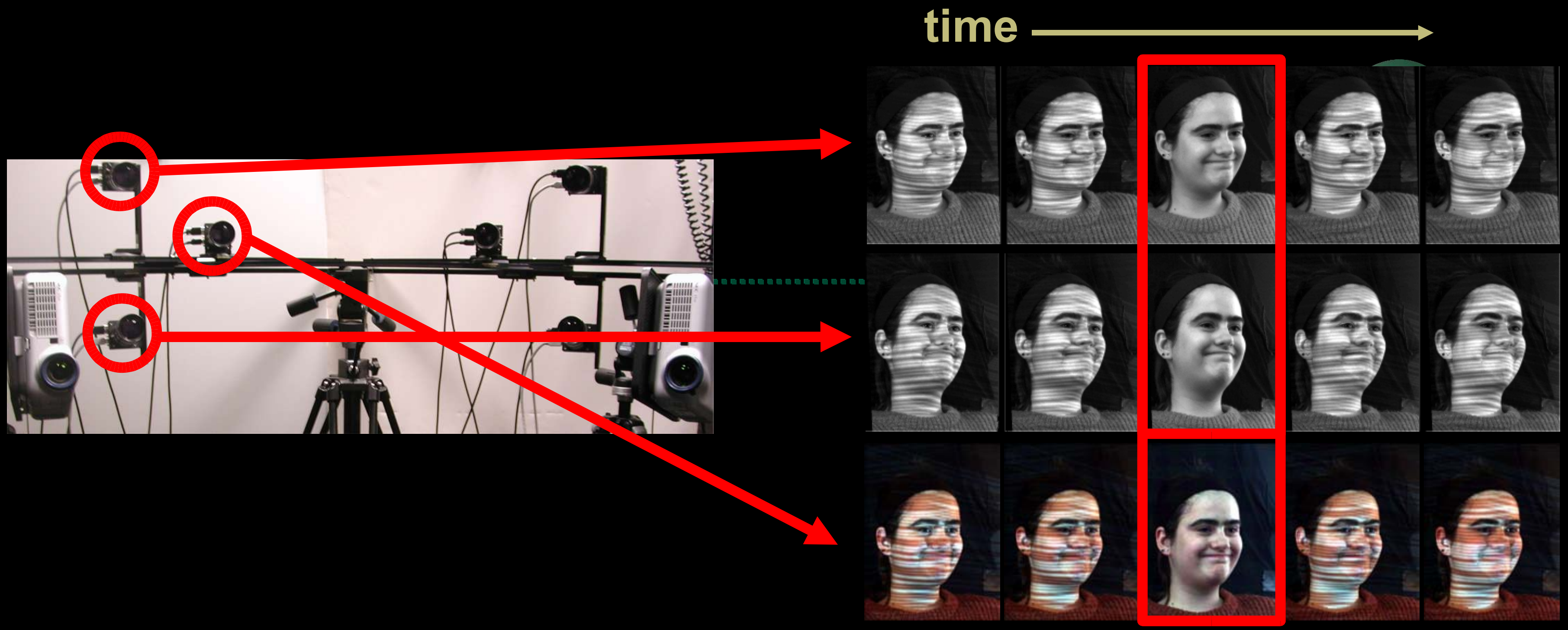


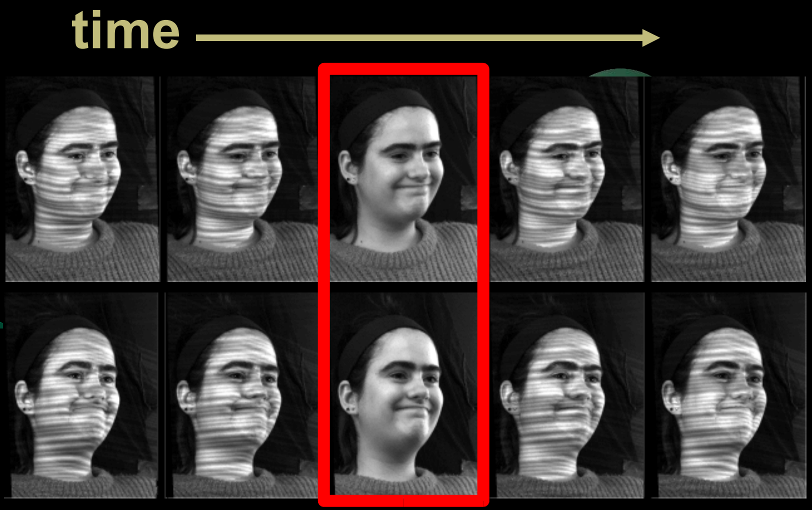
Face capture rig

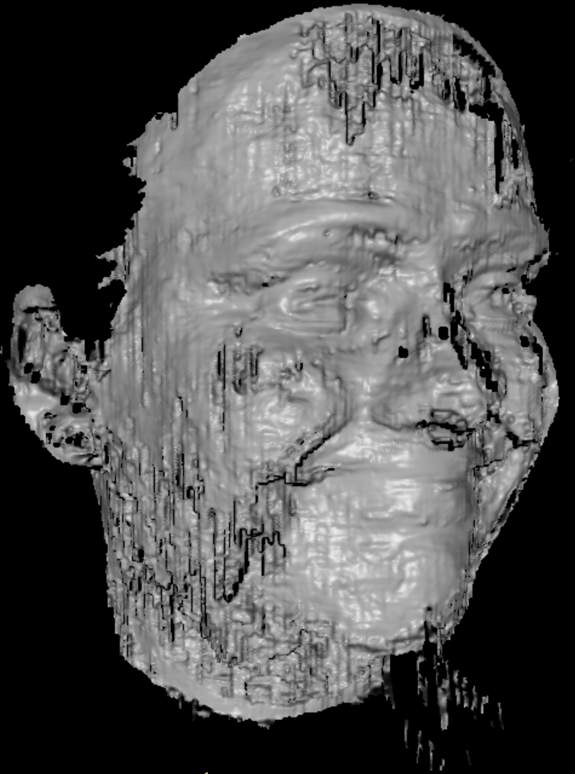
Zhang et al SIGGRAPH 2004

Capture process

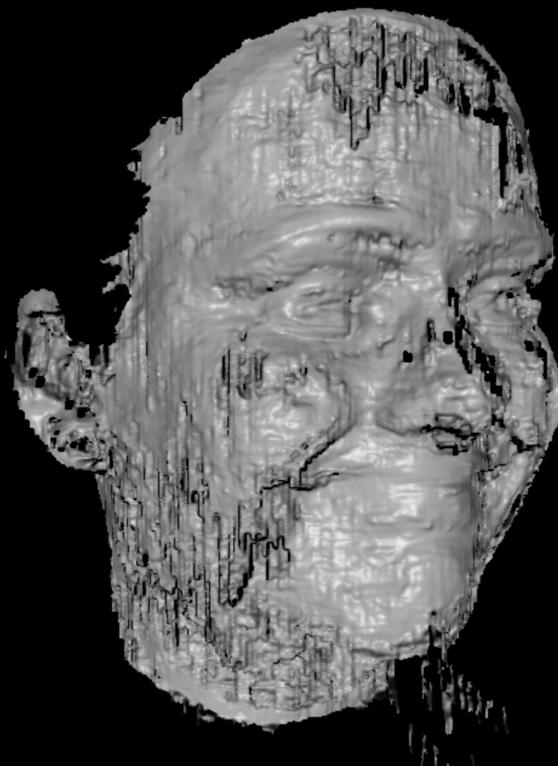
Input videos (640x480, 60fps)


time

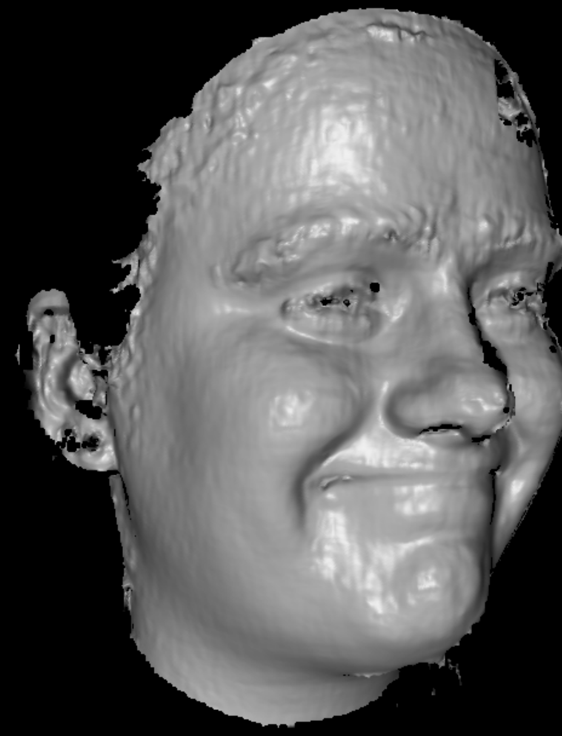

time


stereo

time

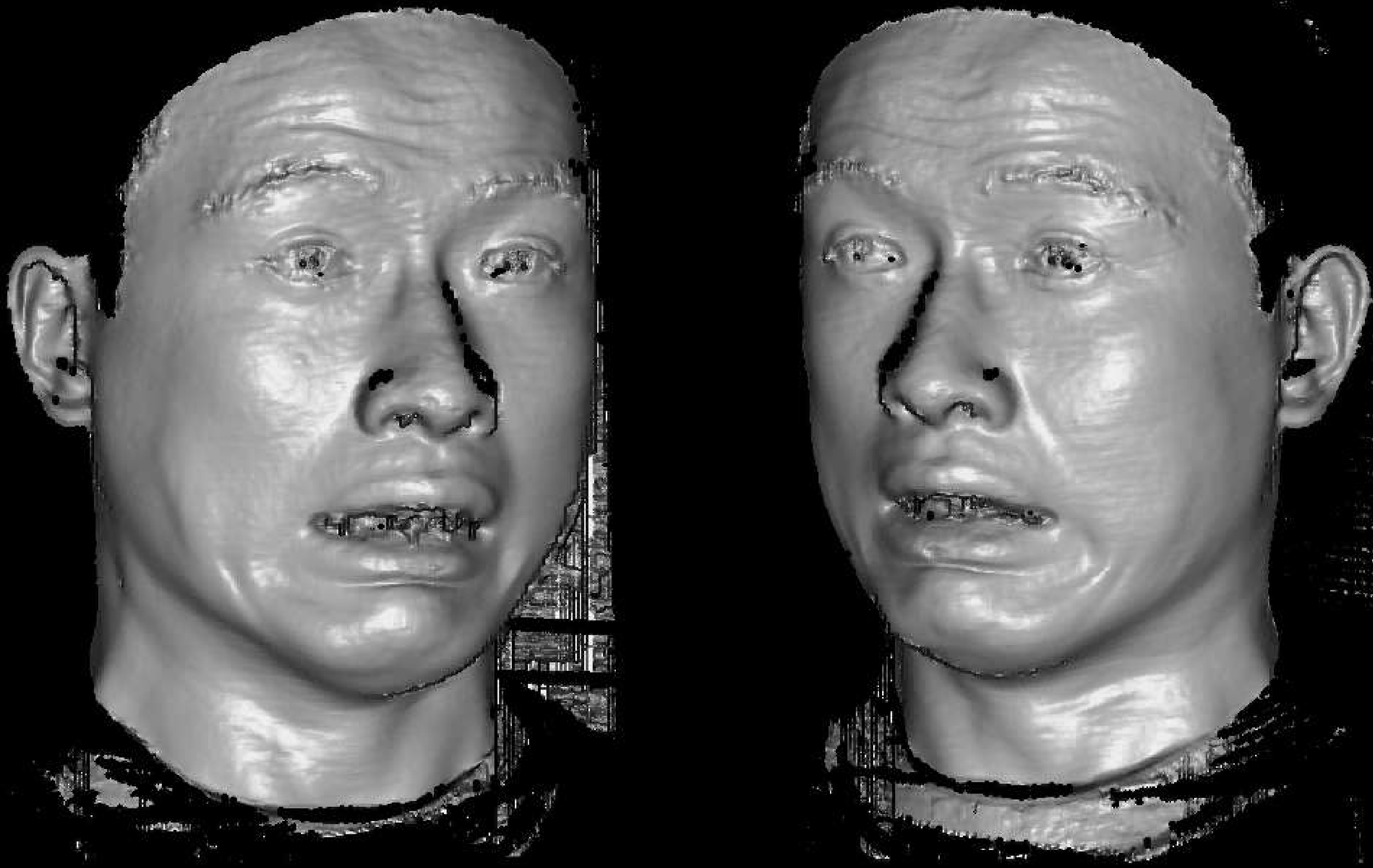


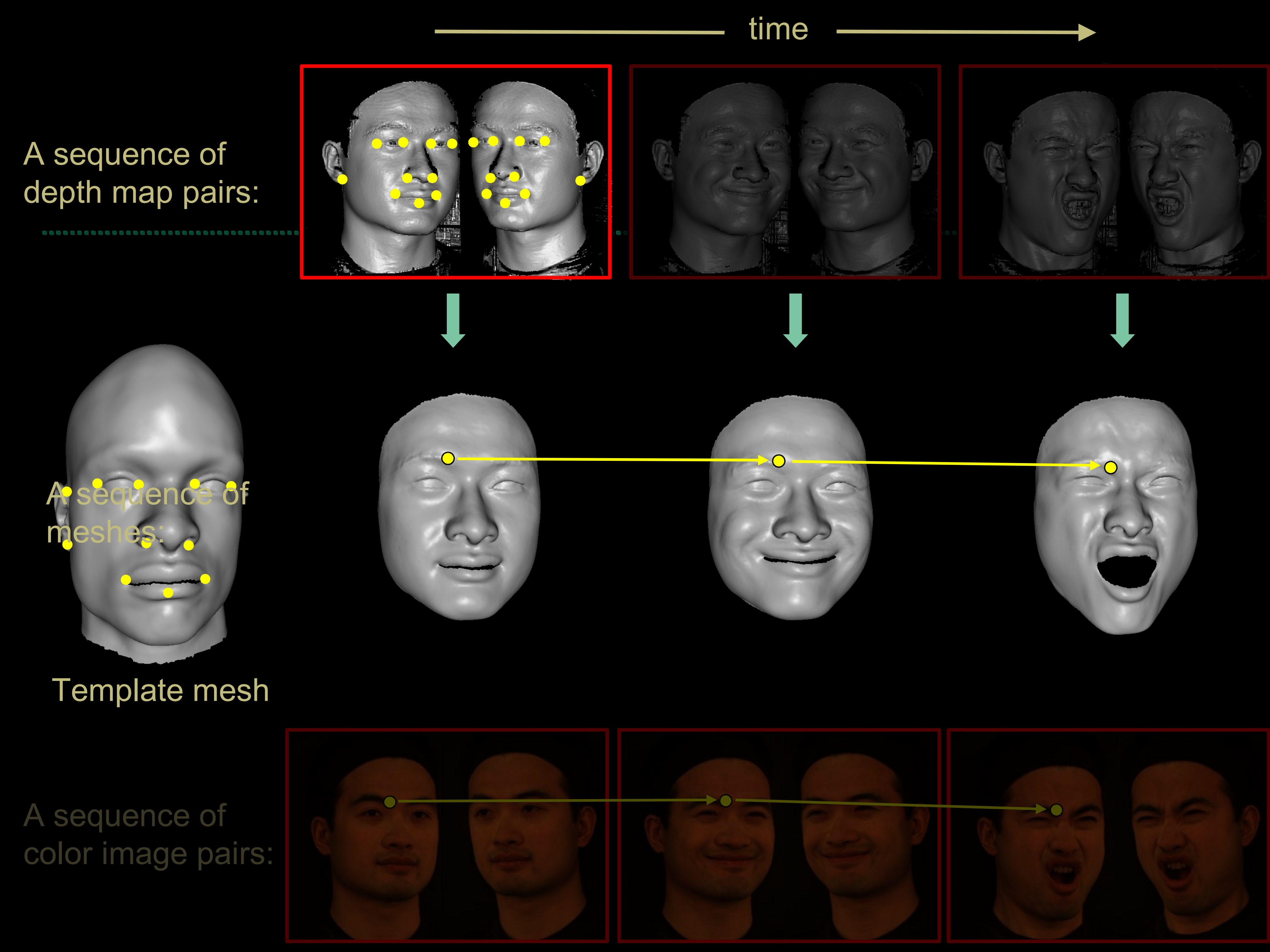
stereo



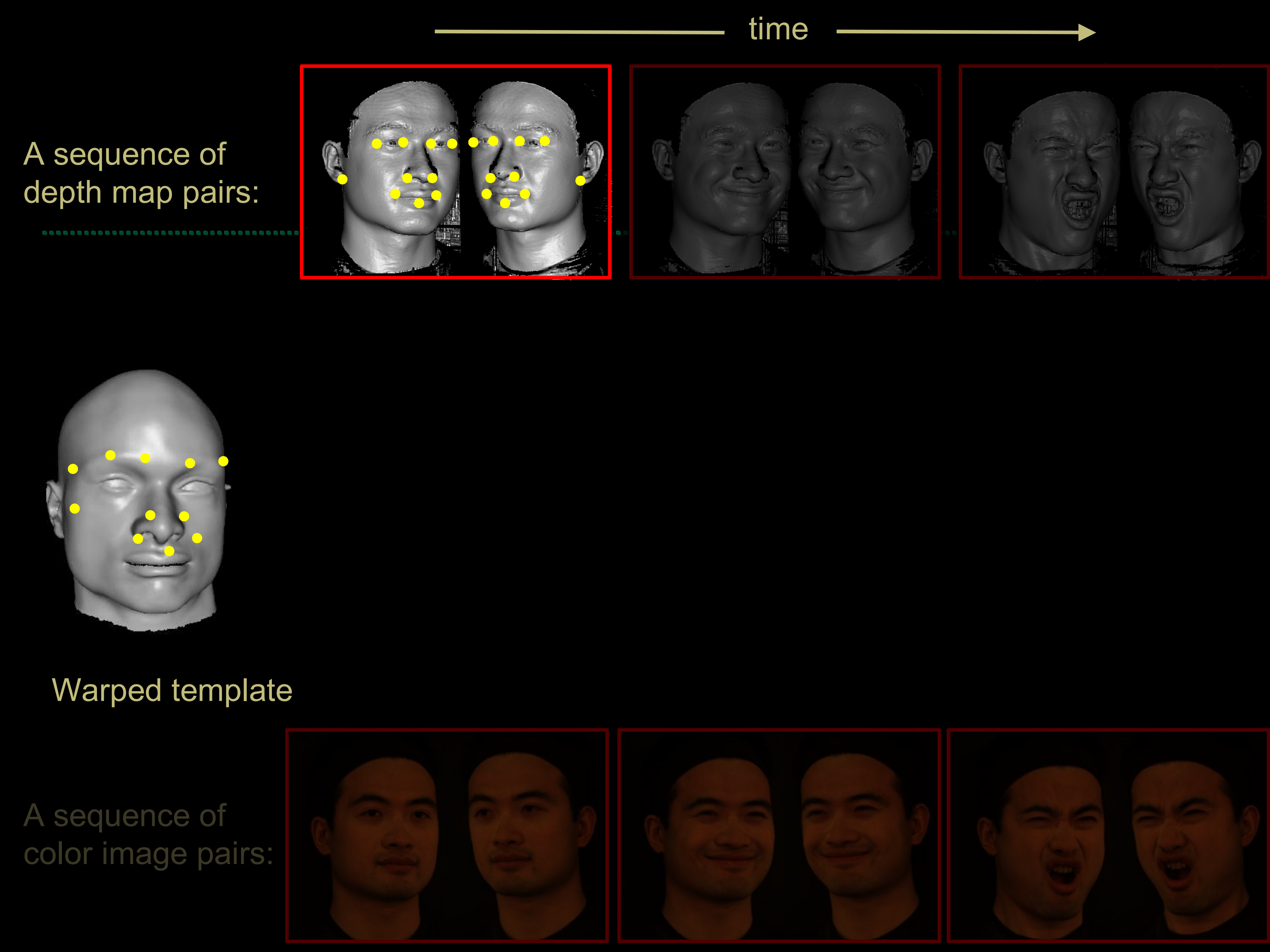
spacetime

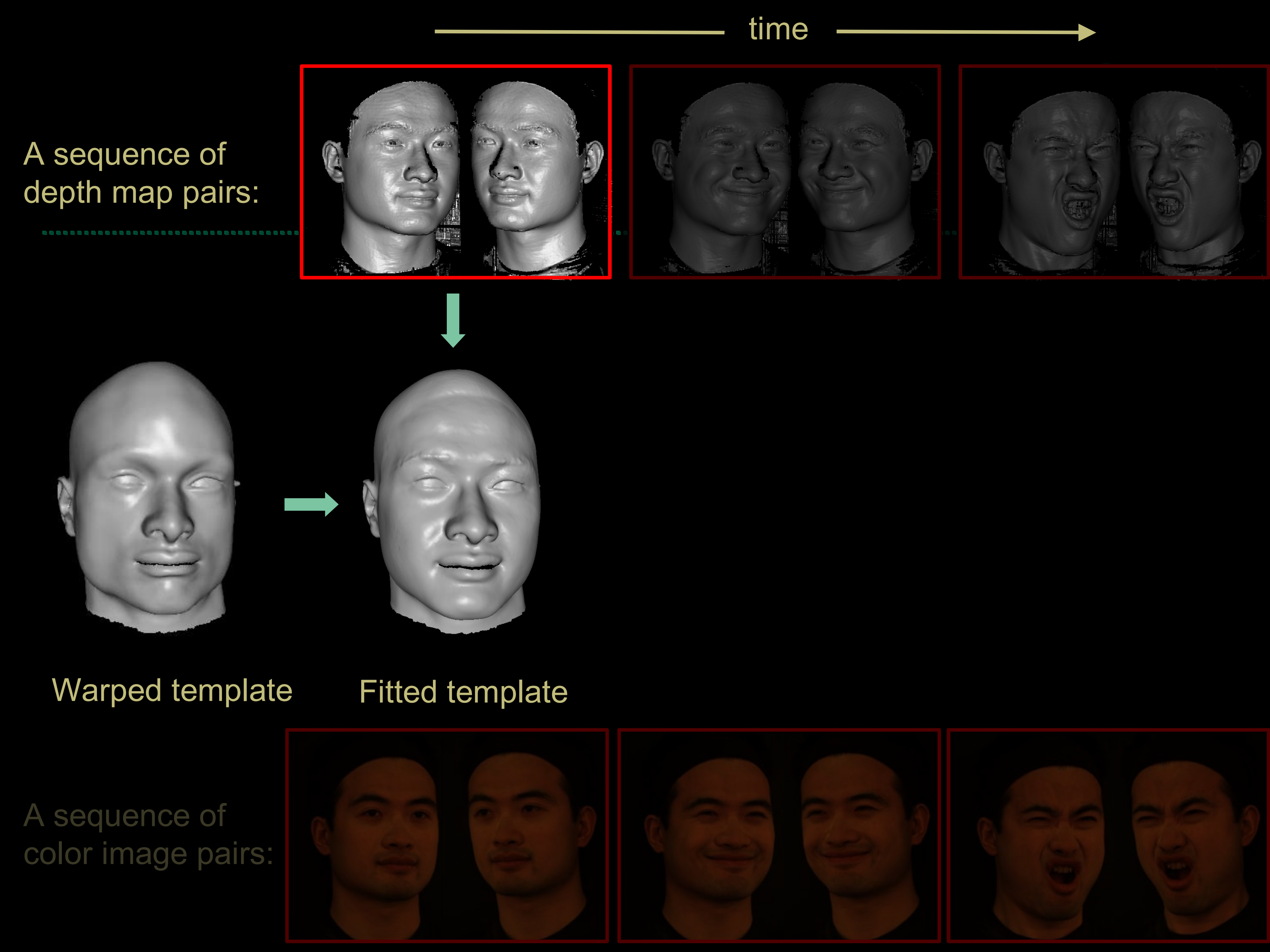
Global spacetime stereo

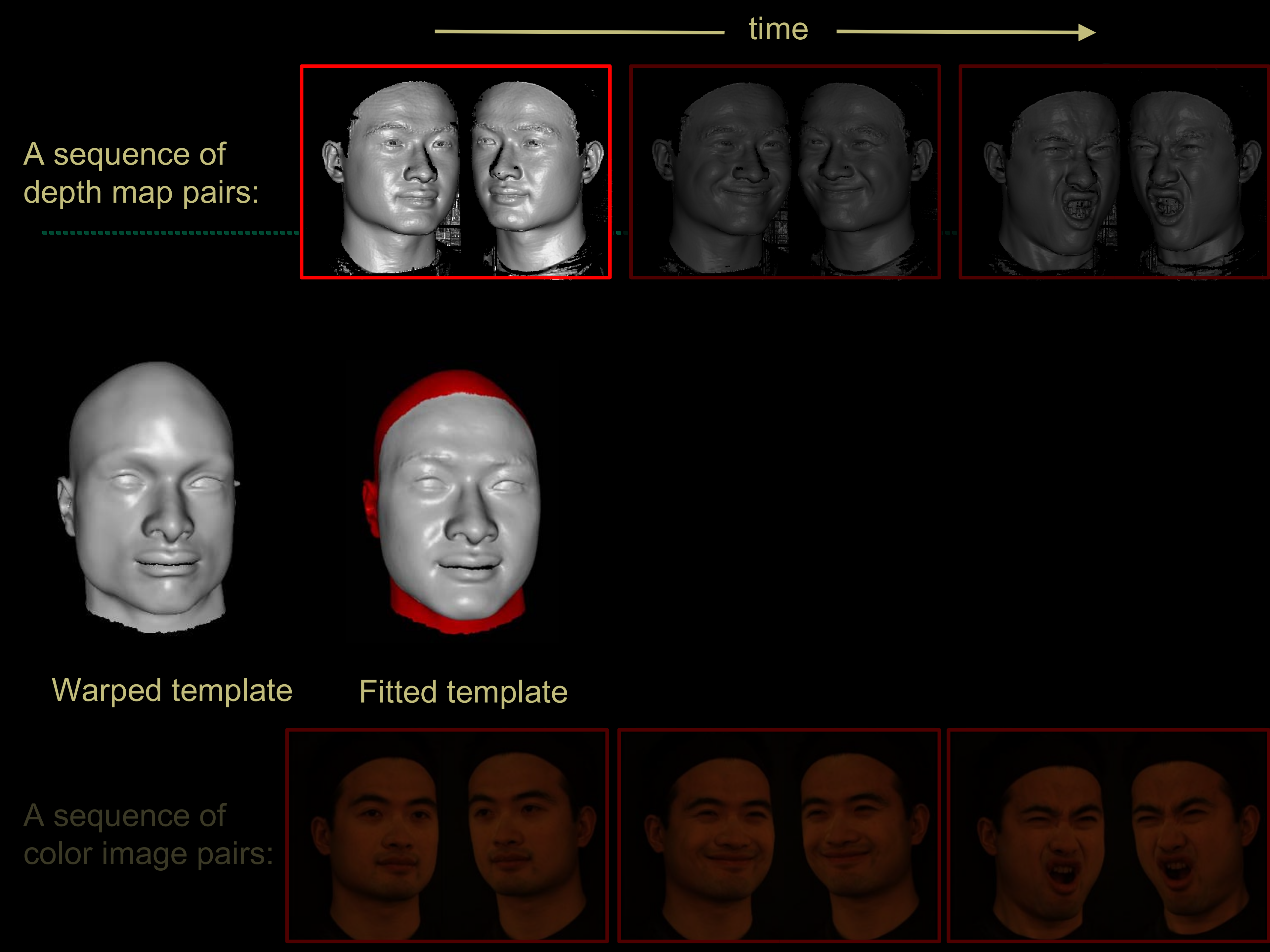


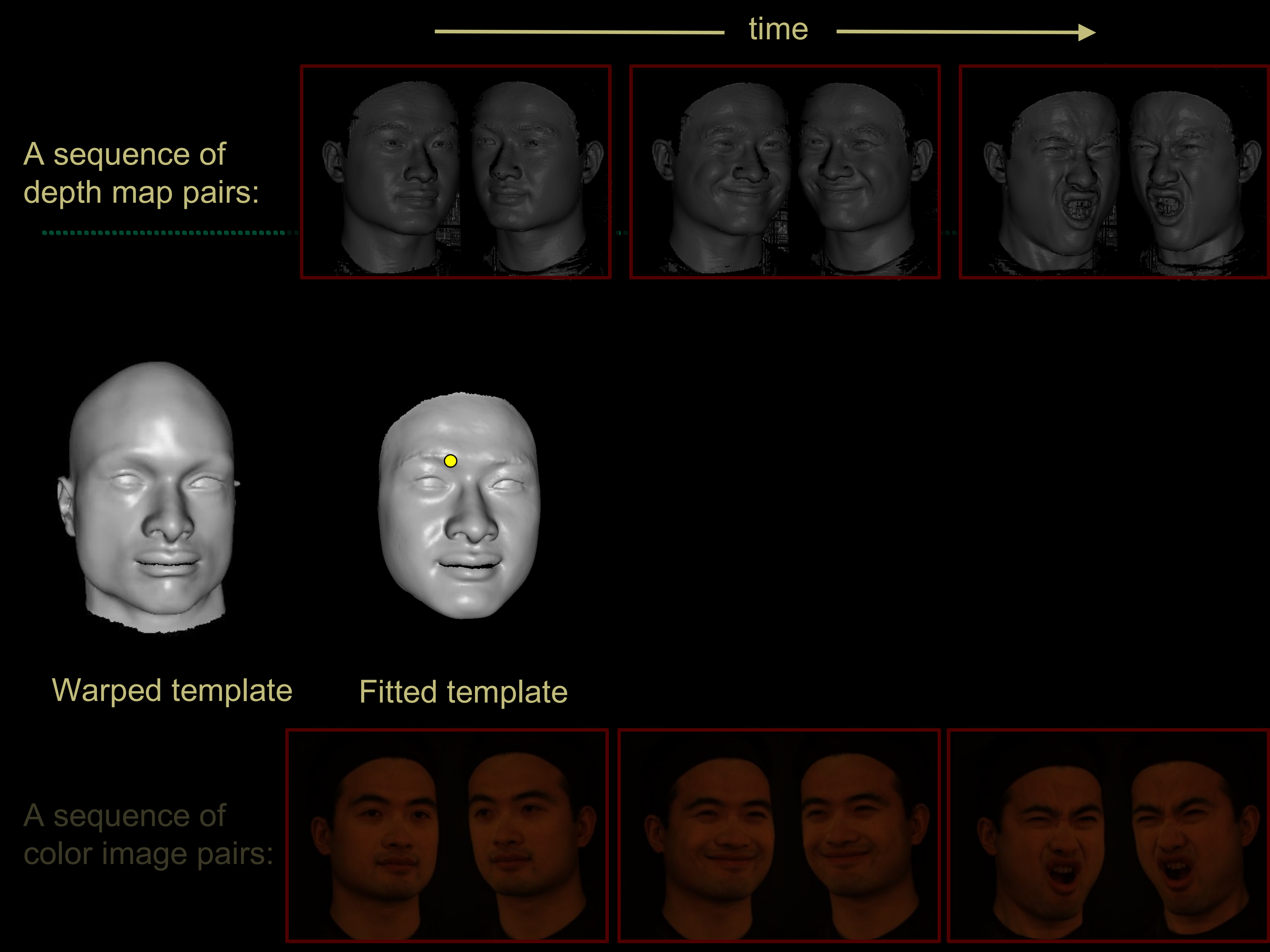


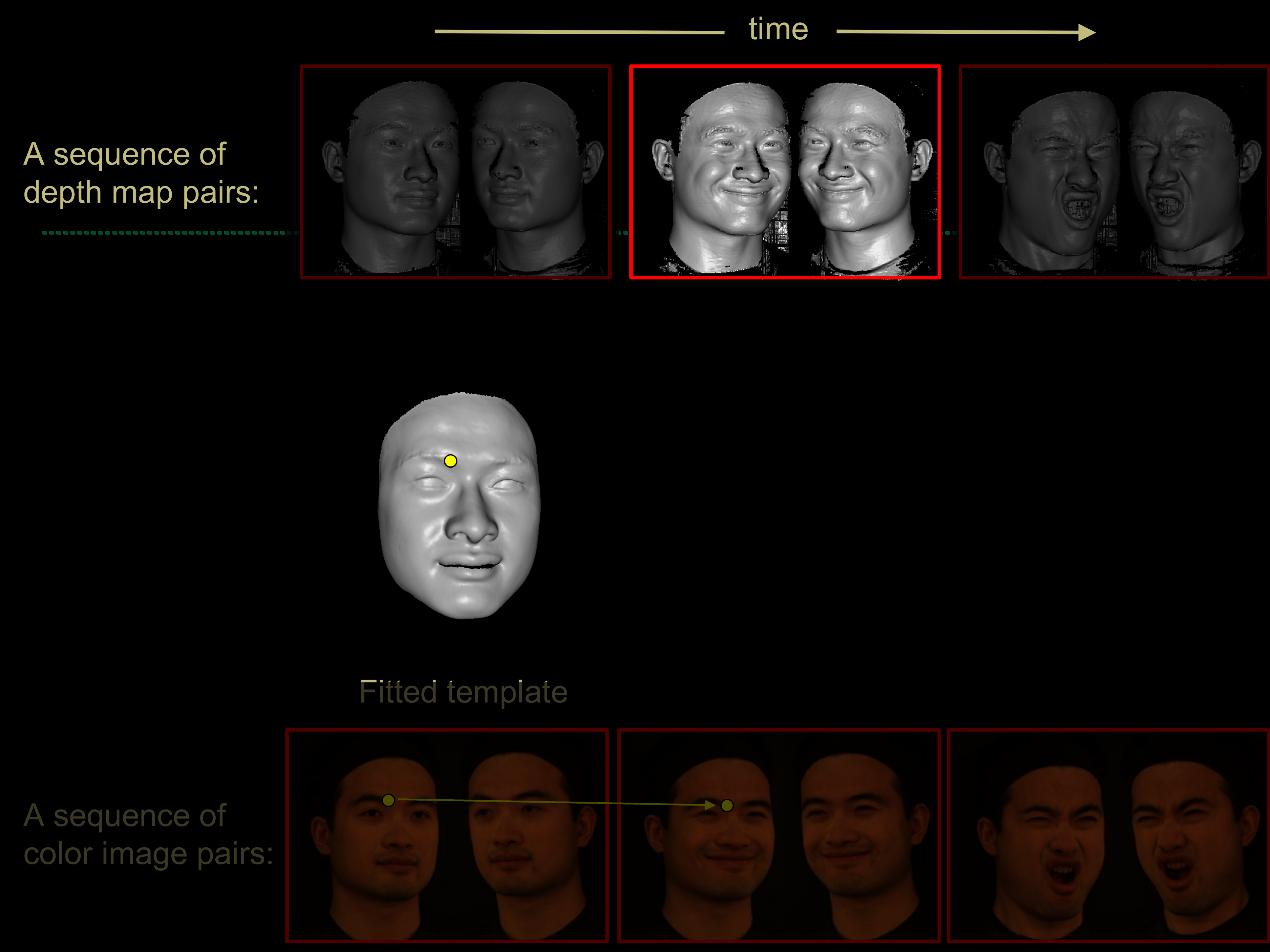
time
A sequence of depth map pairs:
A sequence of meshes:
Template mesh
A sequence of color image pairs:

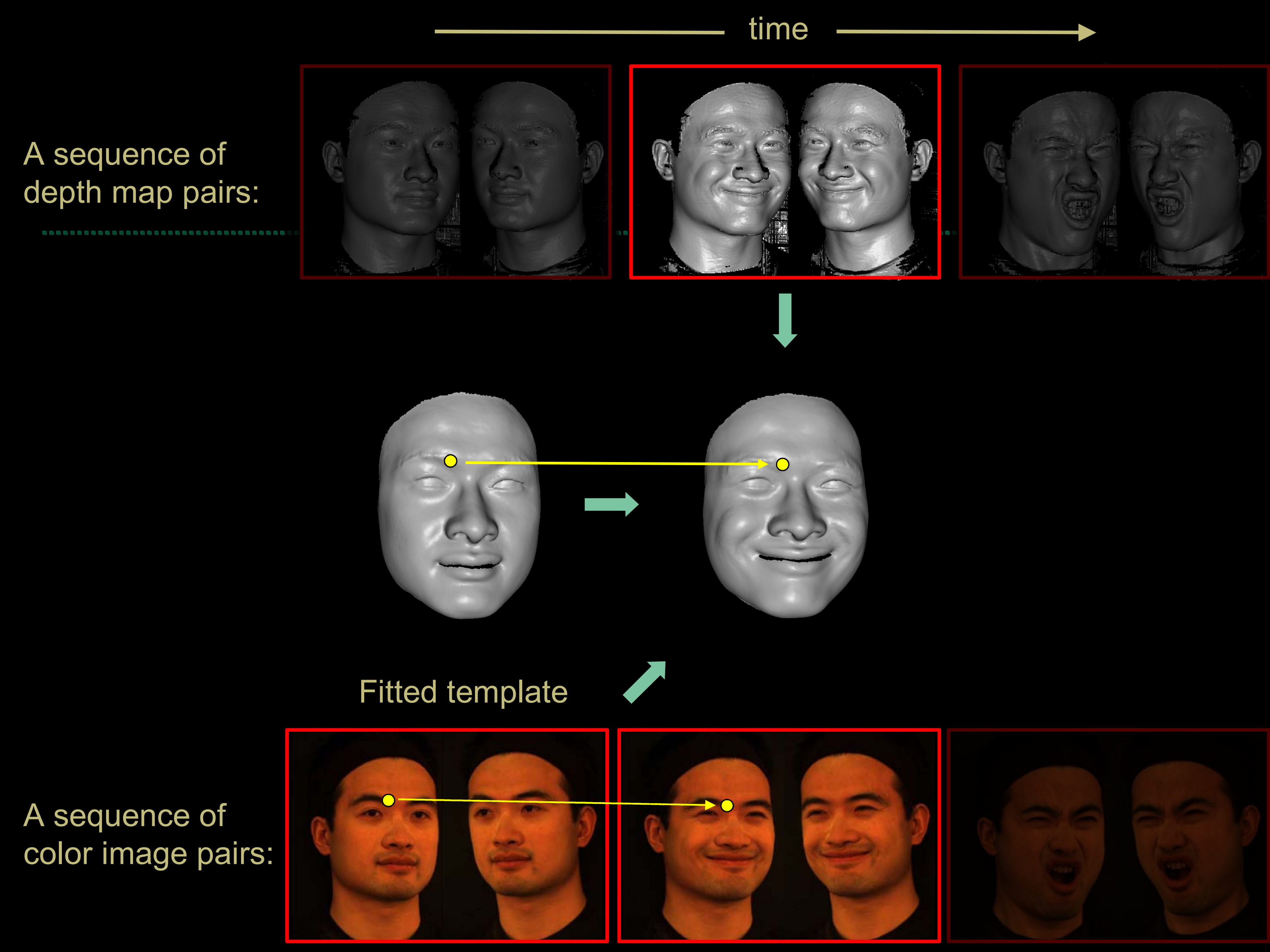

time
A sequence of
depth map pairs:
Warped template
A sequence of
color image pairs:

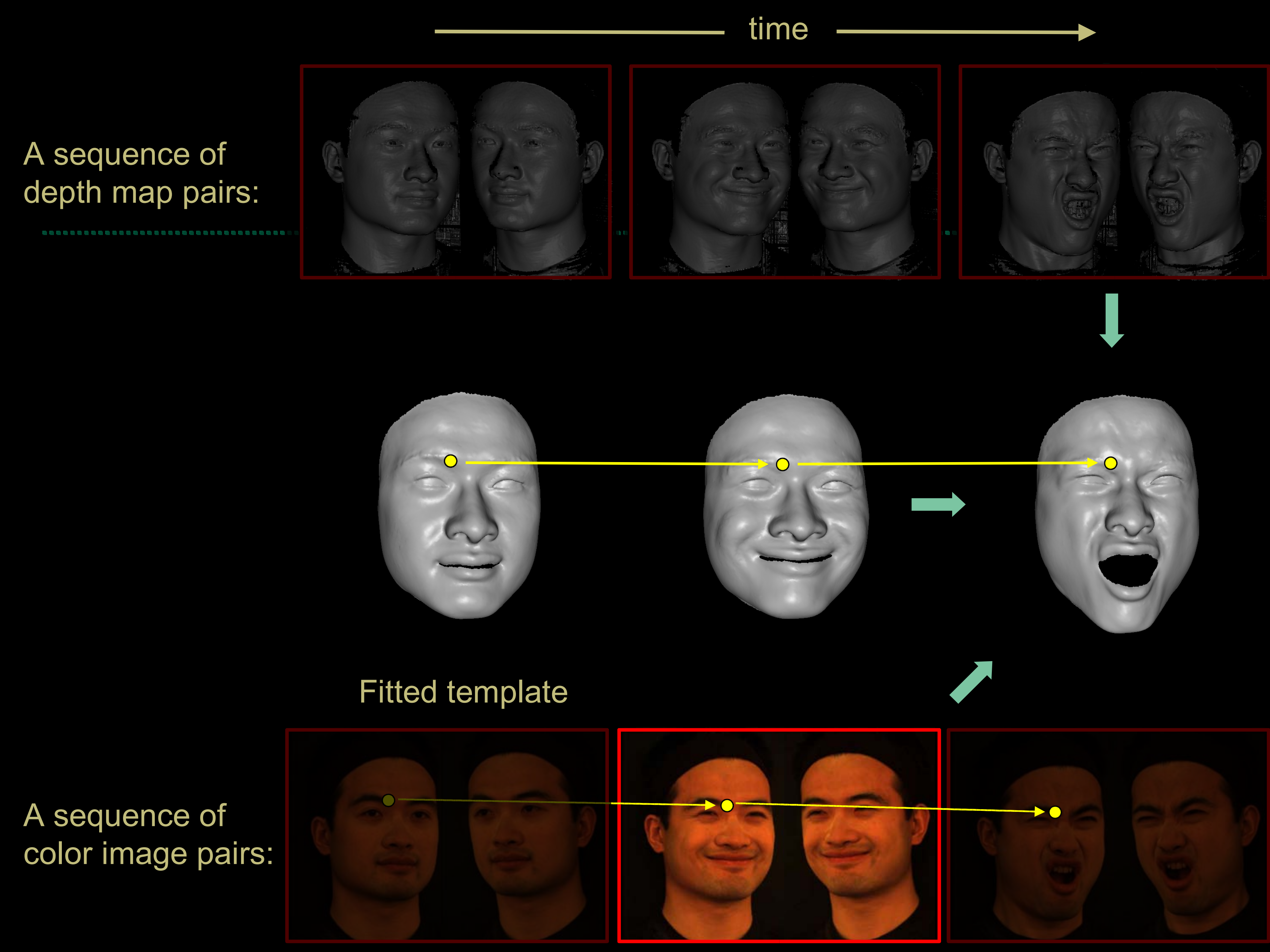

time
A sequence of depth map pairs:
Warped template
Fitted template
A sequence of color image pairs:


time
A sequence of depth map pairs:
Warped template
Fitted template
A sequence of color image pairs:


time
A sequence of
depth map pairs:
Warped template
Fitted template
A sequence of
color image pairs:


time
A sequence of depth map pairs:
Fitted template
A sequence of color image pairs:

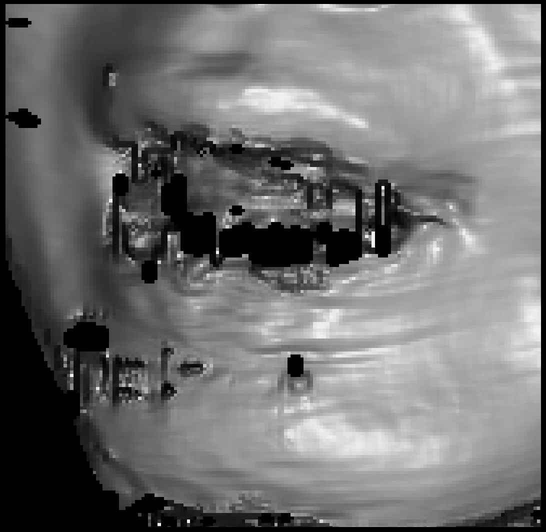

time
A sequence of depth map pairs:
Fitted template
A sequence of color image pairs:


time
A sequence of
depth map pairs:
Fitted template
A sequence of
color image pairs:

Spacetime faces

SIGGRAPH2006

+ High resolution motion (~20K vertices)
− not robust for very fast motion



⇒ Fast cameras

⇒ Better skin models for template fitting

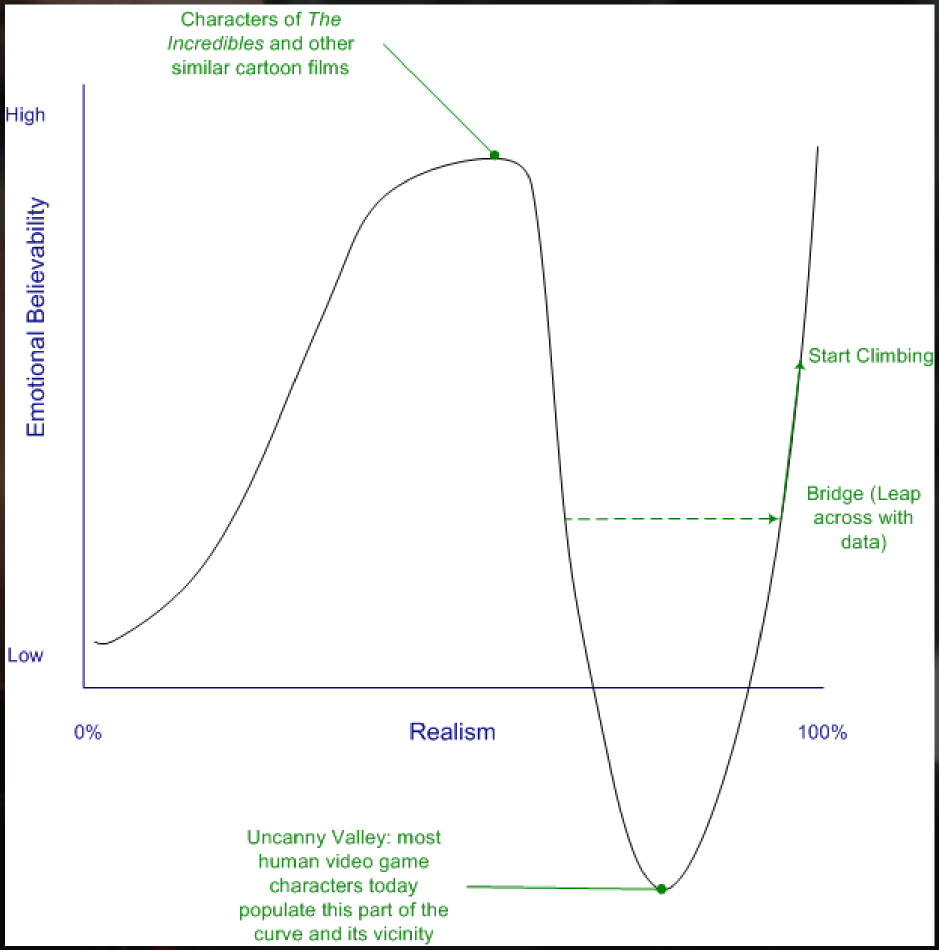
Zhang et al, SIGGRAPH 2004

Playable Universal Capture

George Borshukov Jefferson Montgomery
Witek Werner
James Lau Patrick Mooney Barry Ruff Dave Raposo
Electronic Arts, Inc.

Introduction

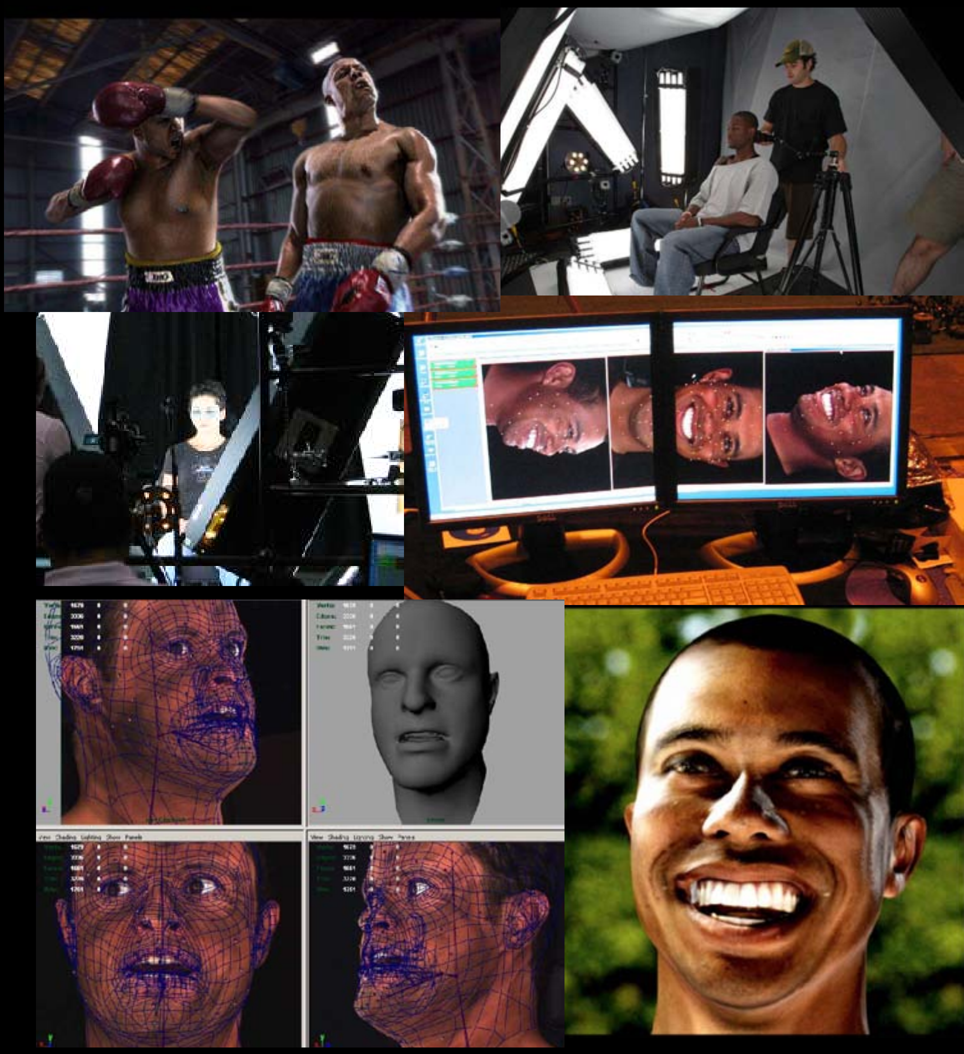
- UCap: High fidelity digital face cloning through accurate capture and reconstruction of both facial motion and texture
- What it gives you today
  - Emotionally Believable Characters??
  - Climb up higher on the right side of the Uncanny Valley?



Universal Capture (UCap)

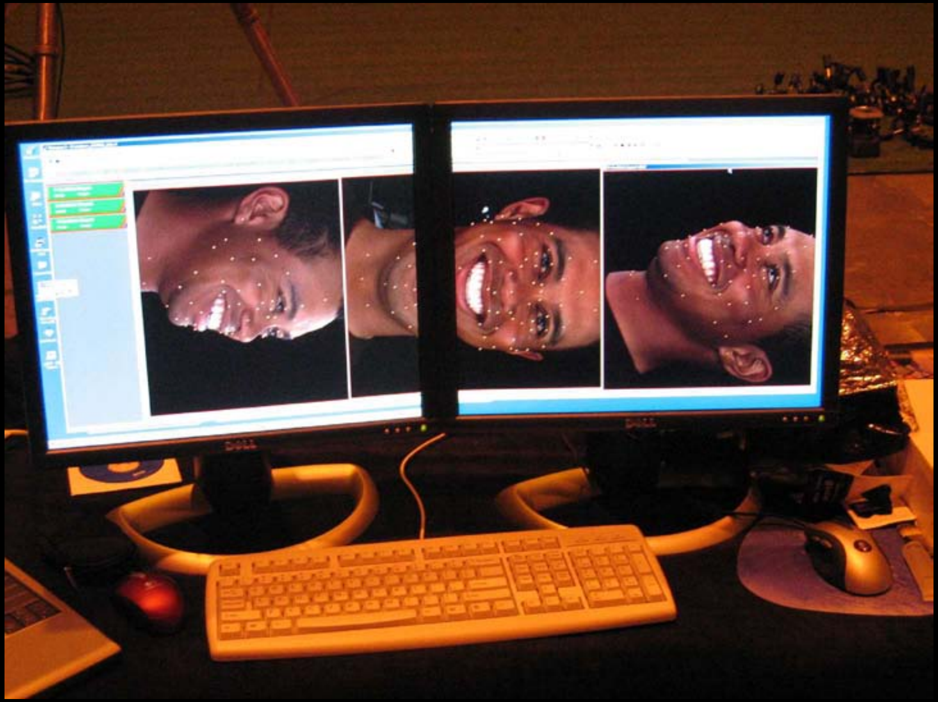
Collaborations

- Fight Night – Nov'04
- Tiger Woods – Jan/Dec'05
- EAJ Fighting Test – June'05
- NBA Live – Dwyane Wade July'05
- MOH:A – Tokyo Game Show – Aug'05
- C&C – Cane, principal character – Aug'05
- EA Mocap - ongoing tech transfer almost complete



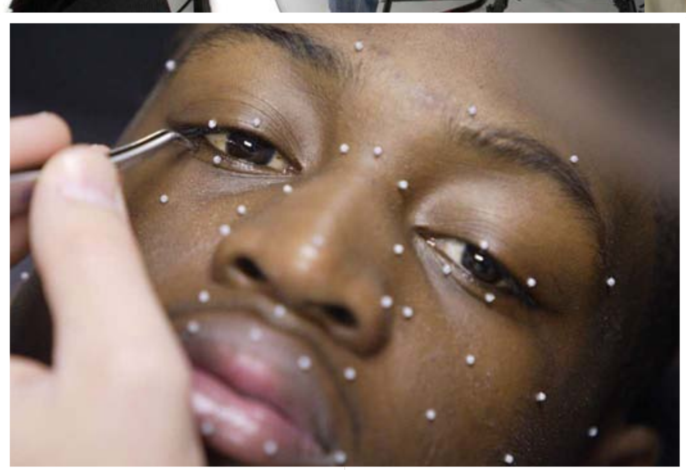
Tiger Woods – Jan/Dec 2005

- Session in Orlando
- Session in LA with 4 other pro golfers



Dwyane Wade – July 2005

- Session in Burnaby

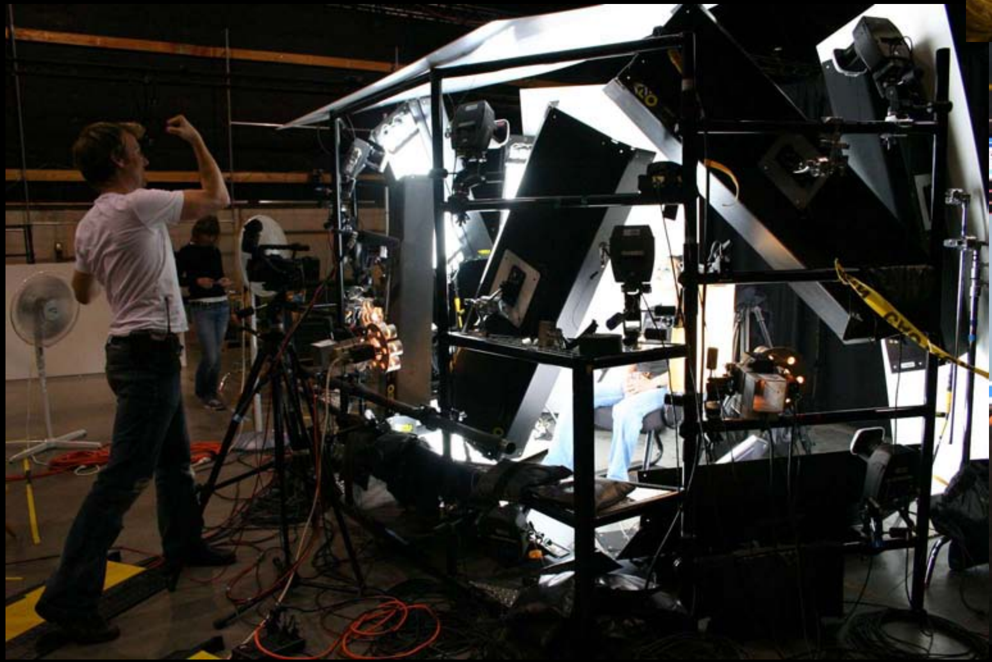


Evolution: UCap -> Playable UCap

- Universal Capture – linear
- Playable Universal Capture – non-linear
  - Capture an Emotion Tree (move tree/motion graph)
  - Apply "move tree" idea used almost universally for body animation in games to the face
  - Robust processing pipeline and tech transfer to EA Mocap
  - Variable basis PCA encoding of geometry & texture for memory efficient real-time playback
    - No facial rig for the runtime: use compressed PCA vertex streams for all facial deformations, which are decompressed at runtime on the SPU (PS3) or GPU (Xbox 360)
  - Identify create smooth transitions, loops
  - Interactive sequencing

EAJ Playable UCap Prototype

- Session in June
- Results presented in November



WWVG_TB

Real-time Demo Team

- George Borshukov
- Witek Werner
- Jefferson Montgomery
- James Lau
- Barry Ruff
- Dave Raposo
- Patrick Mooney

Real-time Demo Team

- George Borshukov
- Witek Werner
- Jefferson Montgomery
- James Lau
- Barry Ruff
- Dave Raposo
- Patrick Mooney

Can Do

- Free camera
- Lighting - always best to mix at least 30% of the original which contains
  - subsurface scattering
  - ambient occlusion
  - microsopic wrinkling and self shadowing effects
- Stylize
  - Shoot actor in make up
  - Through the shaders

Can Do

- Apply deformations to mesh after data has been processed (example superpunch)
- Layer damage/sweat effects throught textures shader parameter adjustment
- Display results from a very complex facial rig
- Remove overall head movement and apply new overall head movement which can come from body mocap, hand animation, procedural techniques
- Interactively switch from clip to clip at pretty much any point triggering appropriate moves

Cannot Do

- Shoot one person and apply results to another
- Create a new performance or "meaningfully" edit that was not captured
- Change eye gaze direction
  - first problem we want to tackle in the next stage

Main Benefits

- Accurate facial shape representation
- Accurate lip shape and tongue placement
- Animated color texture maps

Playable UCap Assets

Head Geo at runtime:

- facial lifecast XYZ scanned at 250 microns (.25 mm)
- 3500 Quads - > 10,000 Quads after 1 level of subdivision
- 20,000 triangles

Facial Rig (used only during processing):

- translation bone-based (slightly modified Mocap facial rig)
- approx 70 bones, 1 per mocap marker
- 8 lip/mouth bones for hand-tracking the lips
- weighting is key

Textures:

- static texture for ears and back of head
- animating textures for face and neck – 1 texture per frame

Maps:

- normal map (static)
- specular map (static)
- eye/lip material mask to isolate for tweaking

WWVG_TB

Shaders

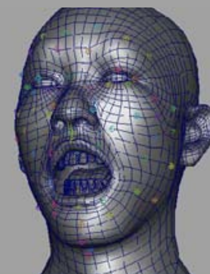
- Take advantage of image-based lighting techniques

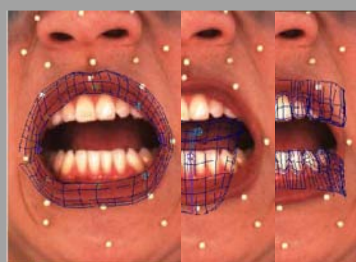
  -diffuse env map lookup with normal vector

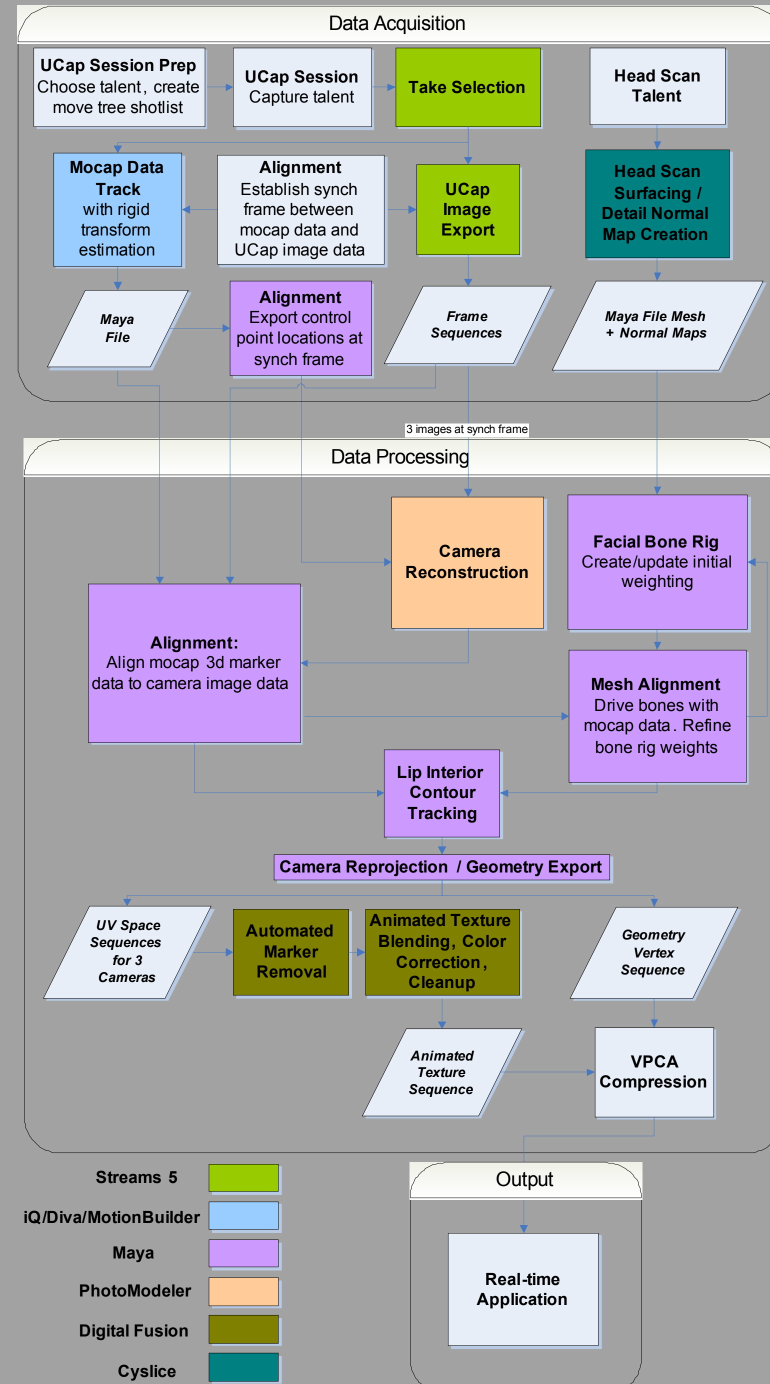
  -preblurred spec environment map look up with reflection vector
- Do not separate eyes. Use masks to relight
- Apply gamma correction at the end off calculations (correct your color texture on read)

Advanced Shading Examples

UCap Pipeline







Designing the Move Tree Shot List

Goal

- interactive ucap demo
- with full dynamic range of motions
- achieving believability and responsiveness

Tools

- DVD reference
- contrast (typical fighting facial expressions vs. openings & inserts)
- layered approach /intensities/
- classification

  ATTACK
  REACT
  DEFENSE
  INSERT
  OPENING
  BREATHING IDLES

A C T I O N S

NAME / INTENSITY LEVEL	SH#	FR#	FR#	DURATION	PRIORITY	NOTES	COMPLETION	COMPLETION
ATTACK_ANGRY_intenL1	0037_1	152	193	41	M		G	G
ATTACK_ANGRY_intenL2	0037_1	750	798	48	H		G	G
ATTACK_ANGRY_intenL3	0037_1	257	300	43	M		G	G
ATTACK_ANGRY_intenL4	0037_1	661	709	48	M		G	G
ATTACK_COCKY_intenL1	0038_1	203	257	54	M		W	W/G
ATTACK_COCKY_intenL2	0038_1	82	149	67	H		W	W/G
ATTACK_AGGRESIVE_intenL1	0039_1	78	110	32	L		W	W/G
ATTACK_AGGRESIVE_intenL2	0039_1	167	199	32	M		W	W/G
ATTACK_AGGRESIVE_intenL3	0039_1	527	560	33	M		W	W/G
ATTACK_AGGRESIVE_intenL4	0039_1	618	653	35	H		W	W/G
ATTACK_TIRED_intenL1	0041_1	722	744	22	L		W	W/G
ATTACK_TIRED_intenL2	0041_1	92	135	43	M		W	W/G
ATTACK_TIRED_intenL3	0041_1	197	236	39	M		W	W/G
ATTACK_TIRED_intenL4	0041_1	640	690	50	H		W	W/G
ATTACK_TIRED_intenL5	0041_1	816	868	52	M		W	W/G
ATTACK_STRENUOUS_intenL1	0042_1	66	131	65	H		P	W/G
ATTACK_STRENUOUS_intenL2	0042_1	171	244	73	M		P	W/G
ATTACK_STRENUOUS_intenL3	0042_1	522	580	58	L		P	W/G
ATTACK_STRENUOUS_intenL4	0042_1	616	718	102	M		P	W/G
ATTACK_KILLER_intenL1	0043_1	446	485	39	L		G	G
ATTACK_KILLER_intenL2	0043_1	188	234	46	H		G	G
ATTACK_KILLER_intenL3	0043_1	90	124	34	M		G	G
ATTACK_KILLER_intenL4	0043_1	344	430	86	L		G	G
REACT_FACE_R	0044_2	580	627	47	H		D	G
REACT_FACE_L	0044_2	657	704	47	H		D	G
REACT_TOES	0044_2	740	776	36	H		D	G
REACT_STOMACH_intenL1	0044_2	809	848	39	L		D	W/G
REACT_STOMACH_intenL2	0044_2	961	1025	64	M		D	G
REACT_STOMACH_intenL3	0044_2	897	942	45	H		D	G
REACT_AVOIDANCE_L	0048_3	37	64	27	H		P	G
REACT_AVOIDANCE_R	0048_3	98	127	29	H		P	G
REACT_AVOIDANCE_UP	0048_3	151	180	29	H		P	G
REACT_MORTAL	0049_1	91	216	125	H		G	G
DEFENSE_COCKY	0050_3	260	302	42	H		G	G
DEFENSE_DESPERATE_intenL1	0051_1	541	586	45	H		D	G
DEFENSE_DESPERATE_intenL2	0051_1	626	673	47	L		D	G
DEFENSE_HORRIFIED_intenL1	0052_2	64	97	33	L		P	W/G
DEFENSE_HORRIFIED_intenL2	0052_2	137	172	35	M		P	W/G
DEFENSE_HORRIFIED_intenL3	0052_2	217	251	34	H		P	W/G
INSERT_FRUSTRATION	0057_1	217	269	52	H		P	W/G
INSERT_PANIC	0058_2	100	174	74	H		W	W/G
INSERT_INTIMIDATION	0059_1	542	666	124	H		W	W/G
OPENING_PEACEFUL_SPIRITUAL	0060_1	295	484	189	H		P	W/G
OPENING_PHYSICAL_STRENGH	0061_2	355	448	93	H		P	W/G
OPENING_DISRESPECTFUL_AROGANT	0063_1	497	601	104	H		W	W/G
EYE_BLINK	0060_1	272	294	22	H		P	W/G

I D L E L O O P S

NAME / INTENSITY LEVEL	SH#	FR#	FR#	DURATION	PRIORITY	NOTES	COMPLITION

Captured Facial Move Tree

Post processing

Reference creation

- video analysis (roamer camera)
- comparison against wanted moves
- animation reference table creation (group moves)
- sub range identification (Adobe Premiere editing)
- final take list (clip ranges)
- request to process takes (motion, textures)

Leanne Processing Stats

- Length of capture session - 3 hours
- Total data captured - 197 GB
- Total # of moves captured - 33
- # of moves selected for use – 21

(ultimately split into 55 separate clips and grouped for state flow)

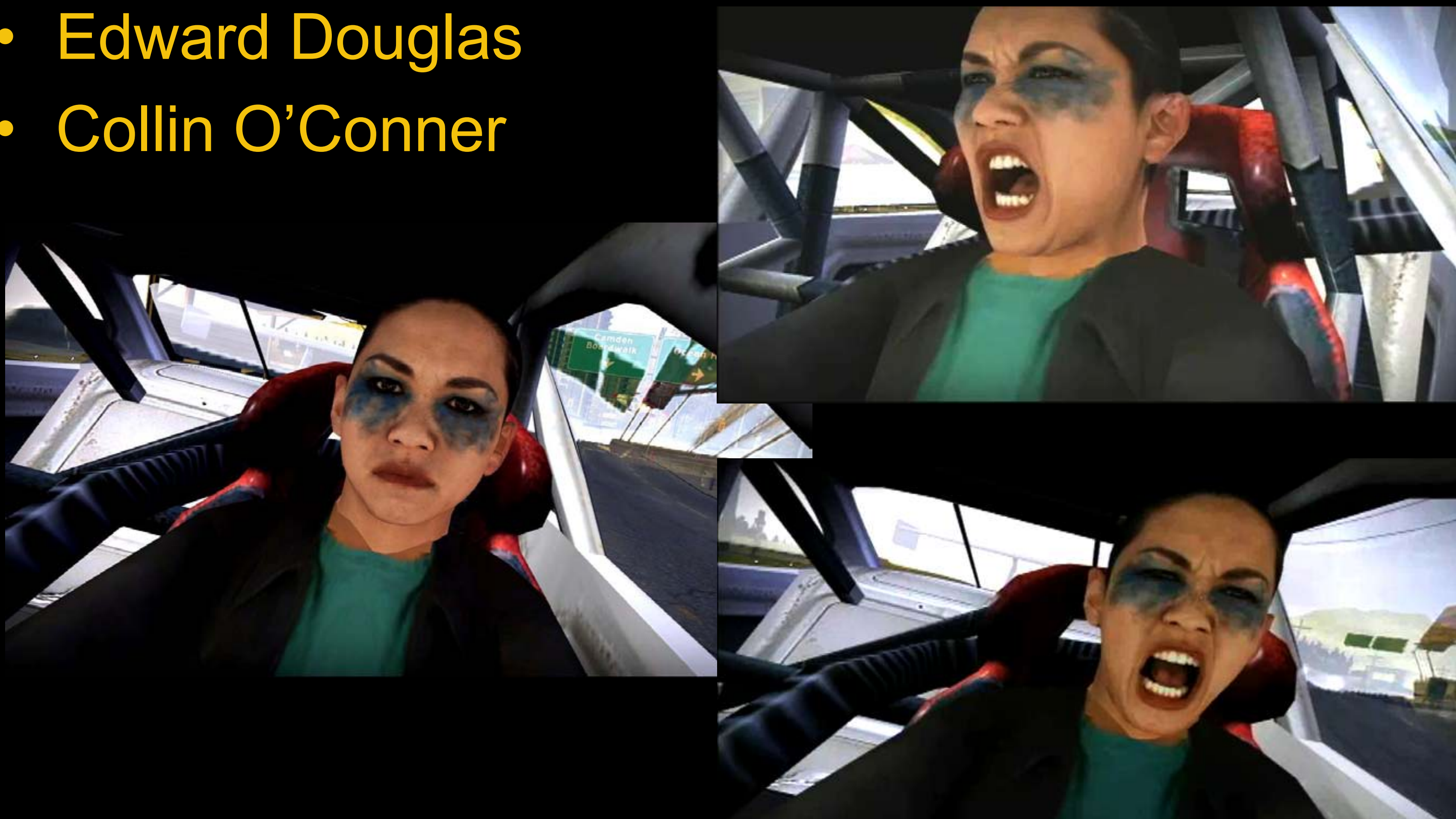
- Total minutes of footage captured - ~17 min
- Minutes of footage selected for use - ~ 2 min
- Average # of frames/move - 65 frames
- Median # of frames/move - 47 frames
- Shortest move - 27 frames, Longest move - 220 frames
- # of trackers – 3, approximate processing time - 1.5 months

Video (includes speech processing)

- UCAP_siggraph2006_852x480_H264_stereo_EA_Watermarked.mov

NFS In-Game Prototype

- Edward Douglas
- Collin O'Conner

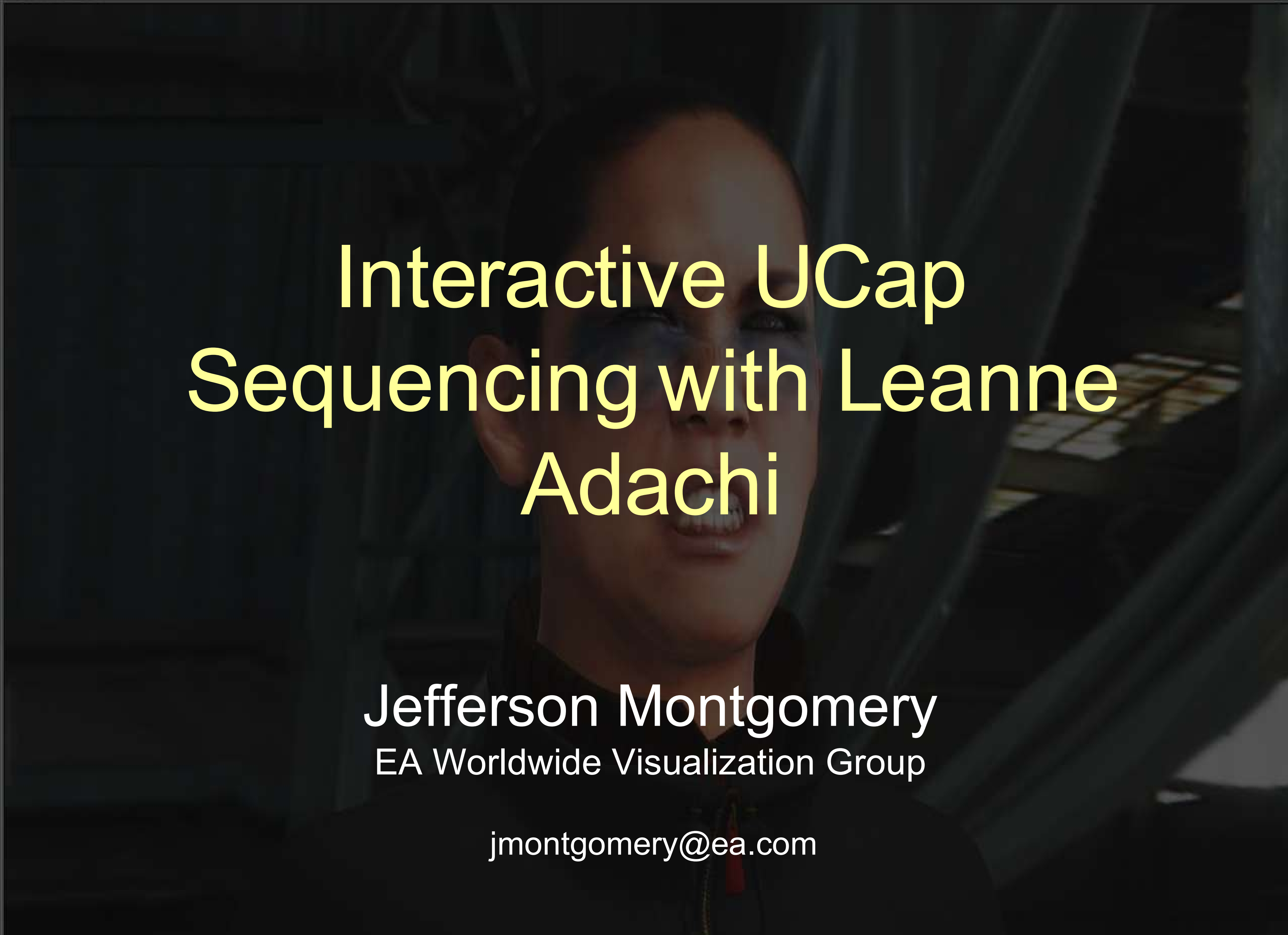


Engineering Overview

- Jefferson Montgomery

Special Thanks

- John Hable, Hakan Kihlstrom, Jean-Luc Dupra, Paul Thuriot, Kevin Noone, James Grieve, Paul Lalonde
- Stefan Van Niekerk, Ben Guthrie, Doug Griffin and the rest of the EA Mocap crew
- Neil Eskuri, Sean Smillie, Collin O'Conner, Edward Douglas and the rest of the innovative NFS team
- Jeff O'Connell, Brian Wideen, Glenn Entis



Interactive UCap Sequencing with Leanne Adachi

Jefferson Montgomery

EA Worldwide Visualization Group

jmontgomery@ea.com

What is UCap?

- Video- and Motion-captured performances
- Facial Animation
- Streaming Textures
- Normal Maps

- All high definition and highly accurate data that captures subtlety of performance

Runtime Challenges

- Enormous amount of source data (GBytes per performance)

- Storage?
- GPU delivery?

⇒ Data Compression!

Data Compression

- VP6, MPG, etc.
  - Difficult random access
  - Artifacts can be abrasive and difficult to control
  - Complicated decode algorithms ★★★

Principle Component Analysis

Advantages

- Forgiving defects (blurring)
- Potential for very high compression
- Automatic pipeline
- Simple reconstruction; well suited to vector processors

Forgiving Defects

Captured PCA

High Compression Ratio

3,652 frames of performance (Leanne demo)

=

6,159 Mbytes of animated texture & geometry

↓

7,147 Kbytes PCA-compressed

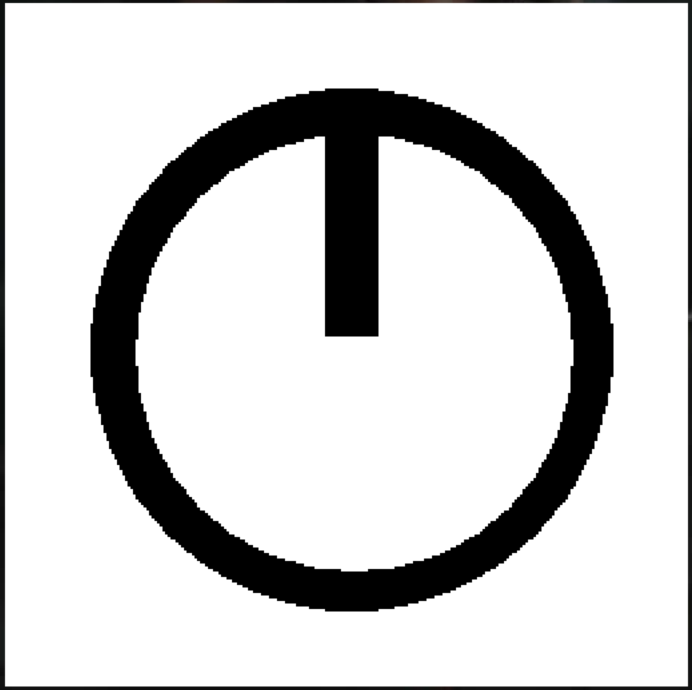
Simple Reconstruction

```
float PcaDecompress(
        float4 weights[4],
        float4 components[4])
{
   return     dot(weights[0], components[0]) +
              dot(weights[1], components[1]) +
              dot(weights[2], components[2]) +
              dot(weights[3], components[3]);
}
```

Per colour or vertex attribute

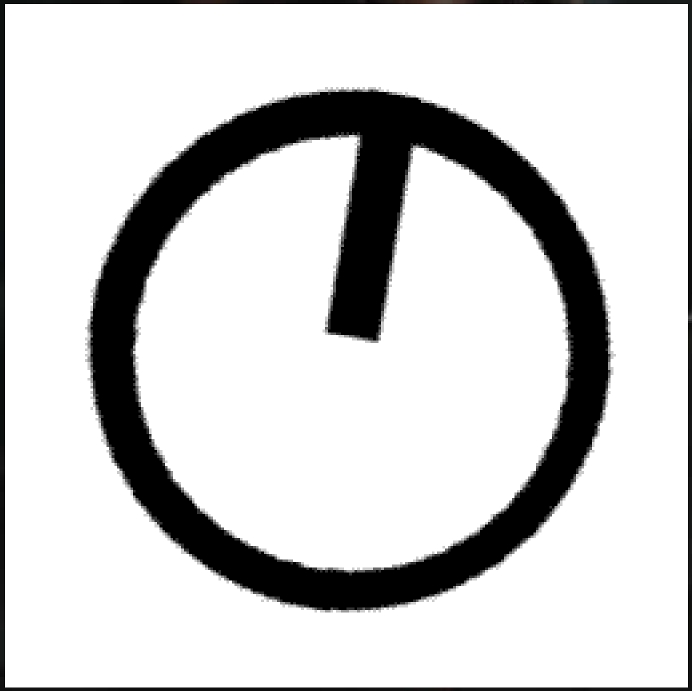
How it works…

- Toy example:



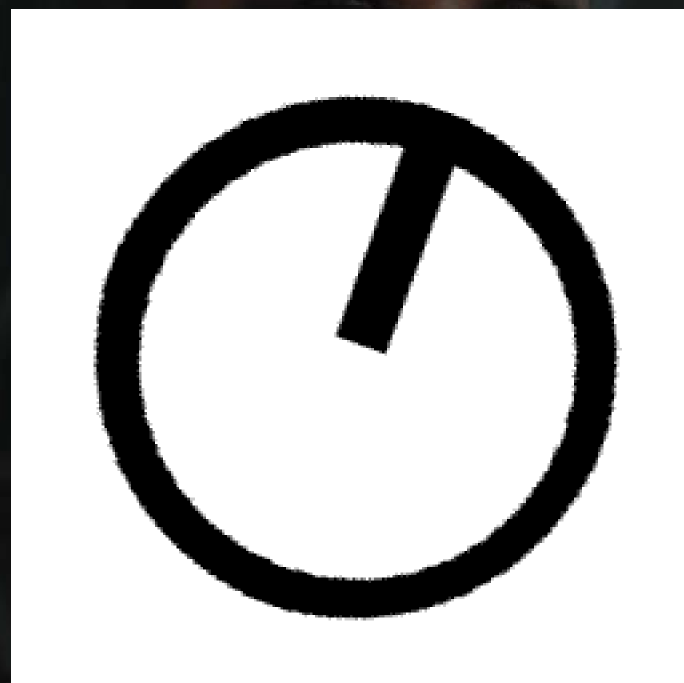
How it works…

- Toy example:



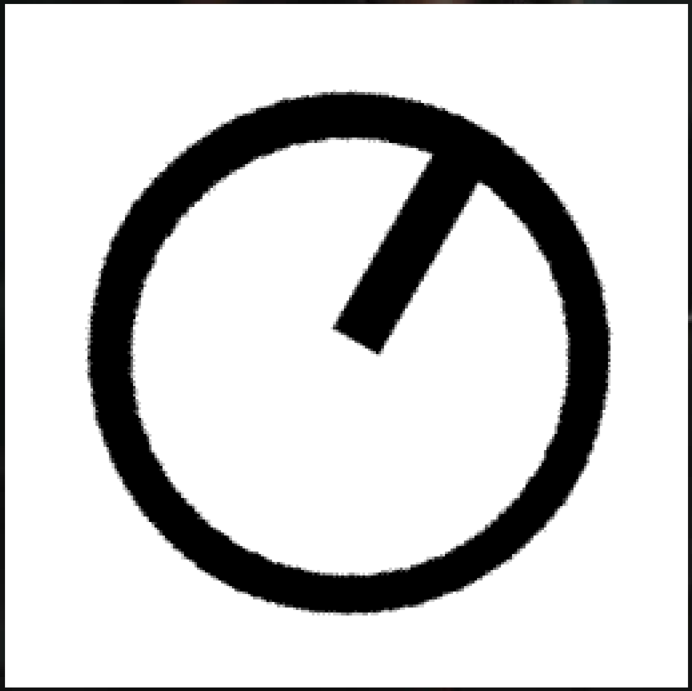
How it works…

- Toy example:



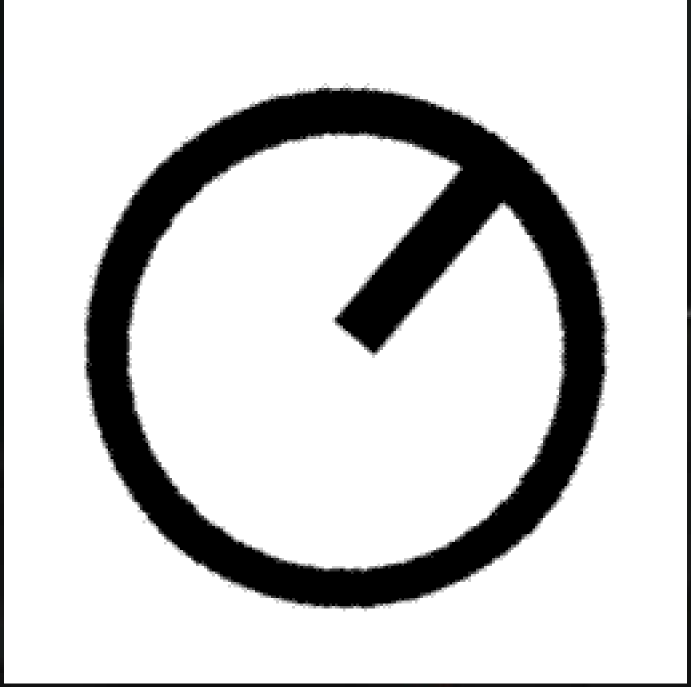
How it works…

- Toy example:



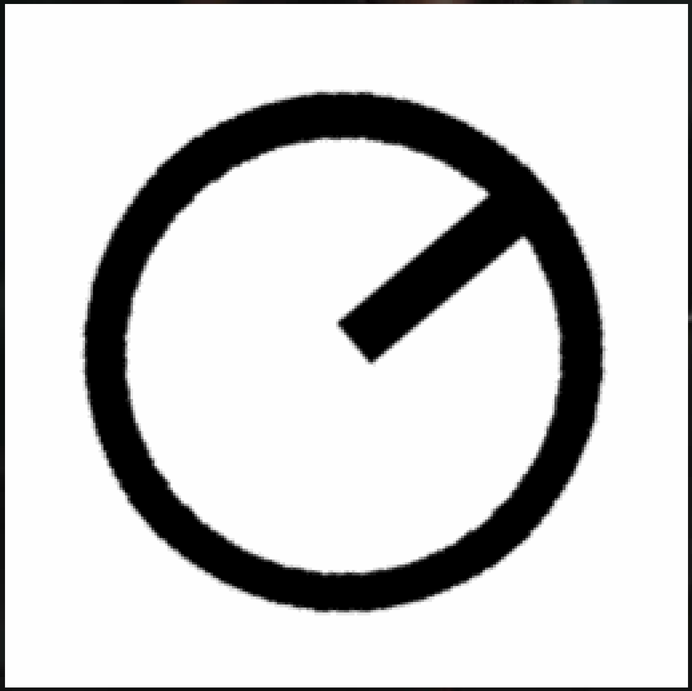
How it works…

- Toy example:



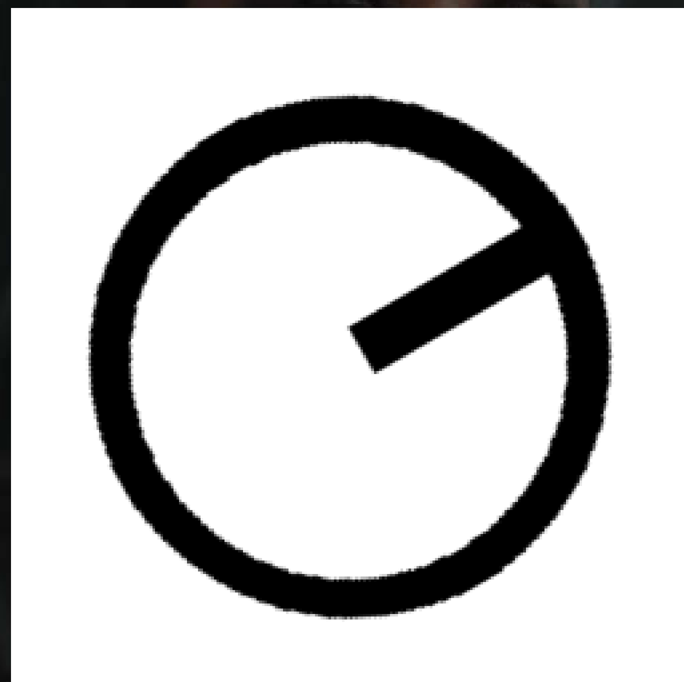
How it works…

- Toy example:



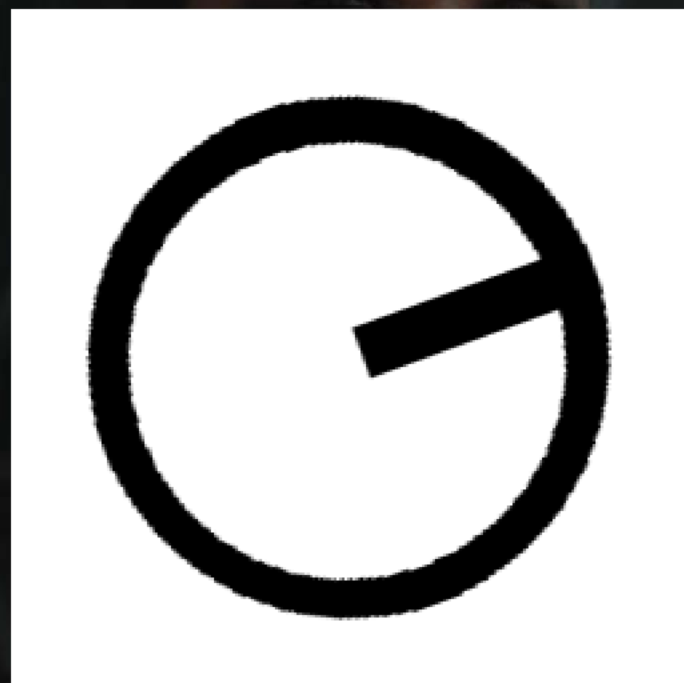
How it works…

- Toy example:



How it works…

- Toy example:

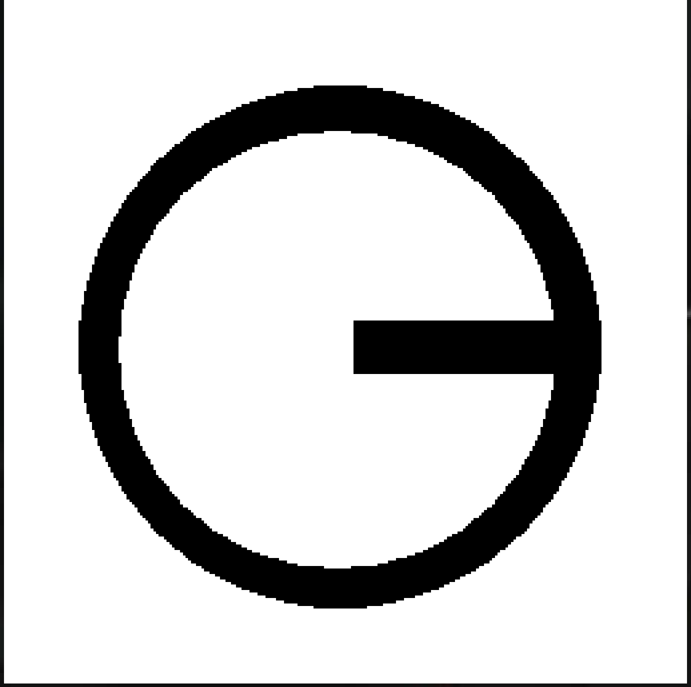


How it works…

- Toy example:

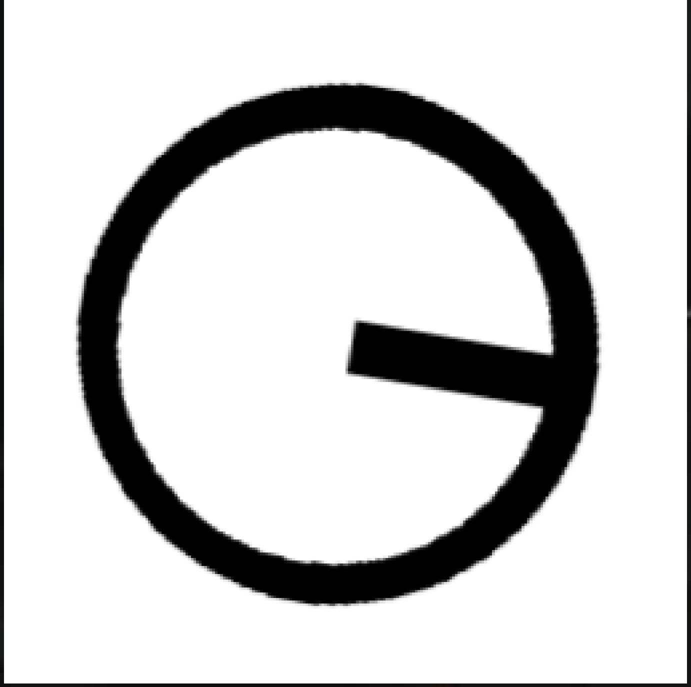
How it works…

- Toy example:



How it works…

- Toy example:

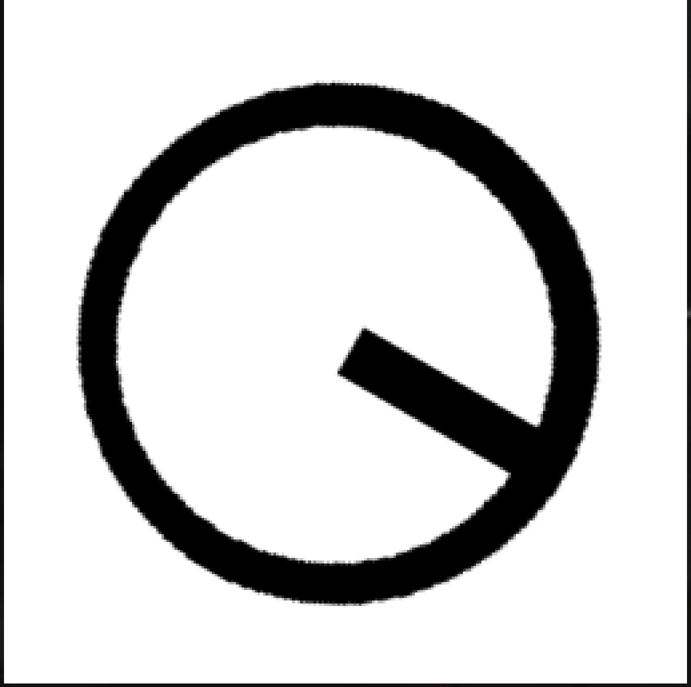


How it works…

- Toy example:

How it works…

- Toy example:



How it works…

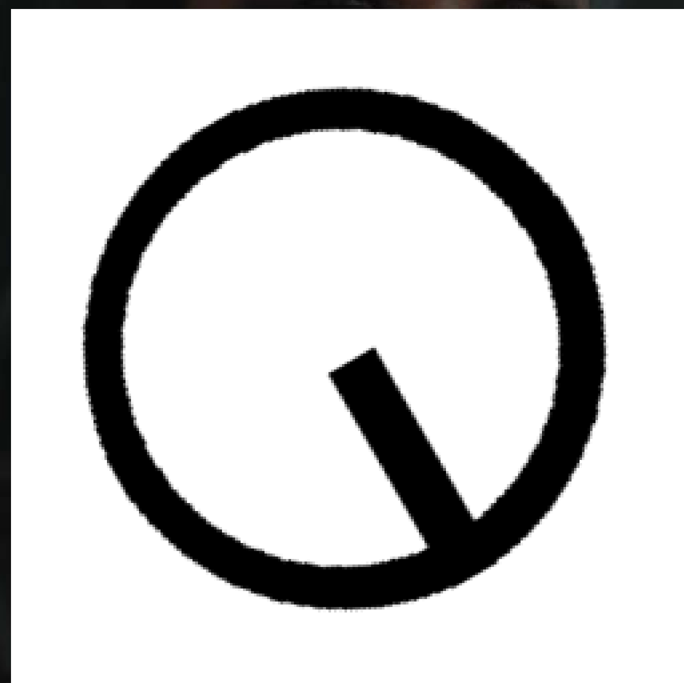
- Toy example:

How it works…

- Toy example:

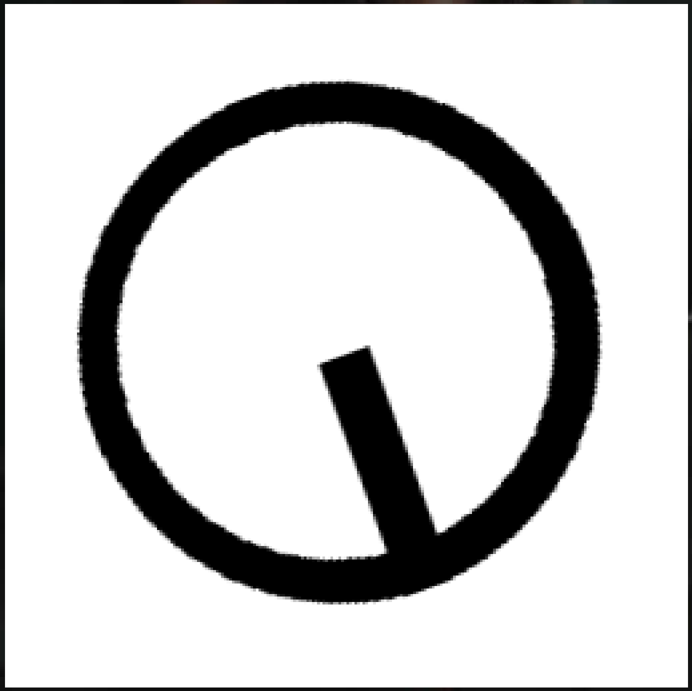
How it works…

- Toy example:



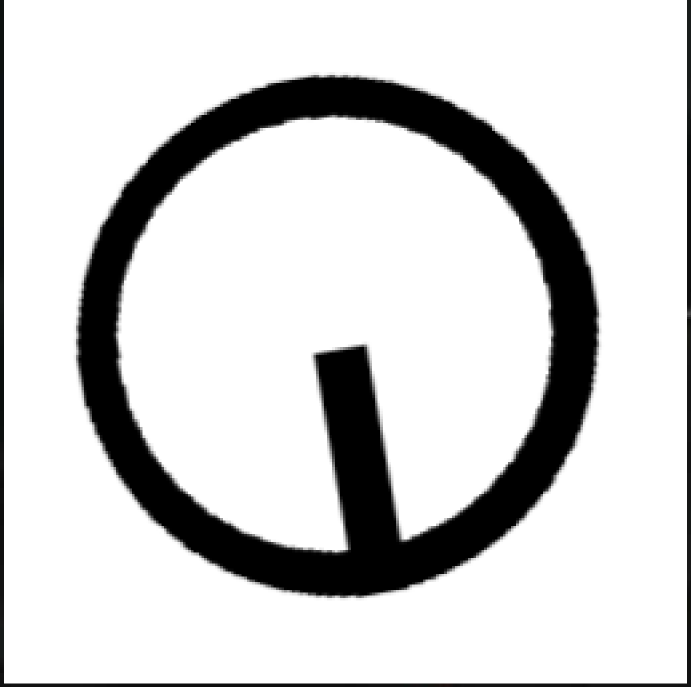
How it works…

- Toy example:



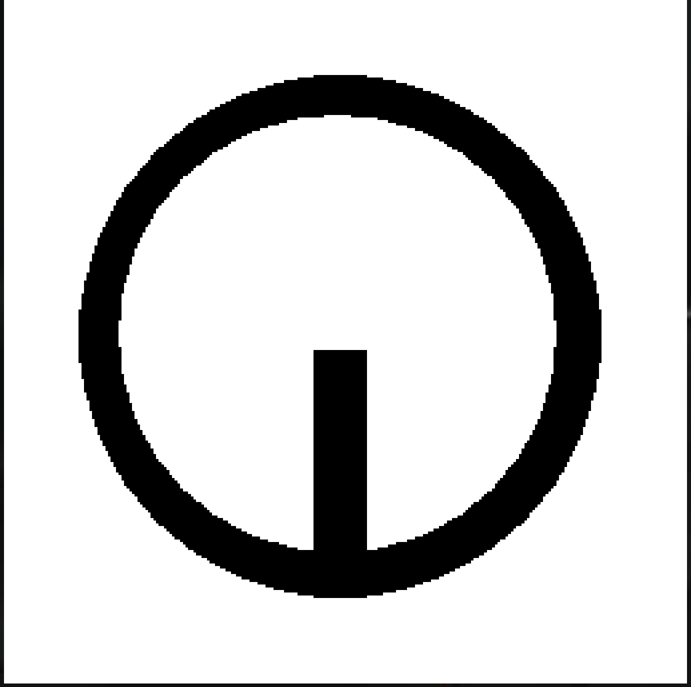
How it works…

- Toy example:



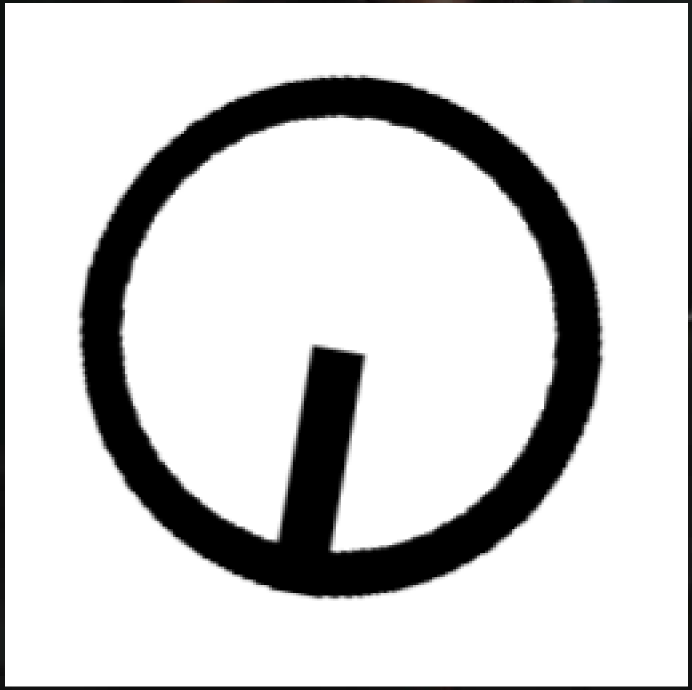
How it works…

- Toy example:



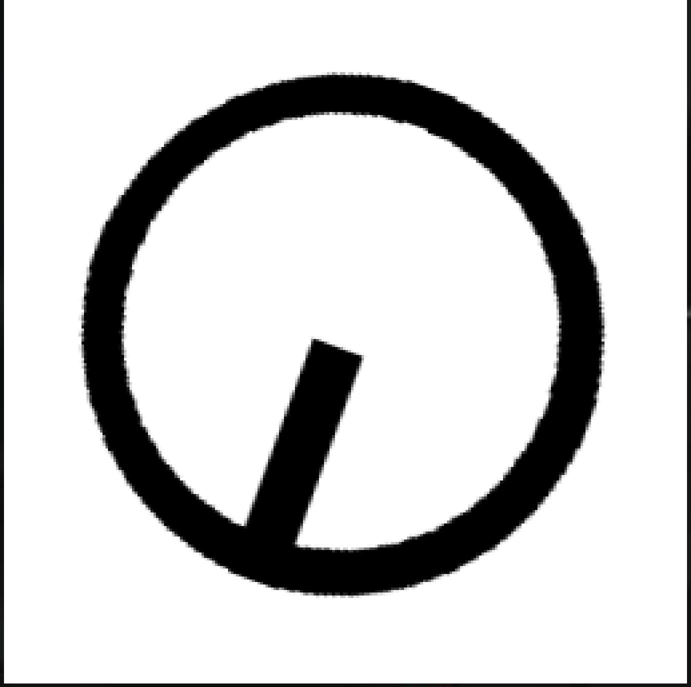
How it works…

- Toy example:



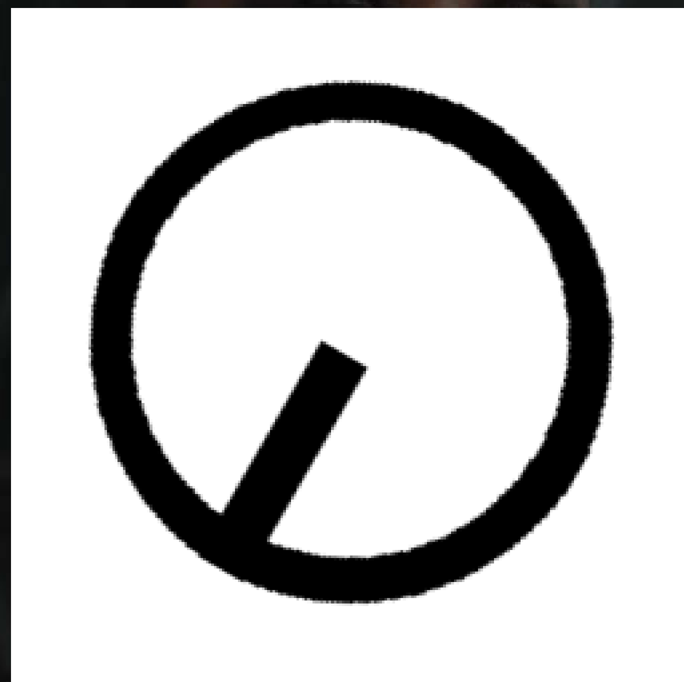
How it works…

- Toy example:



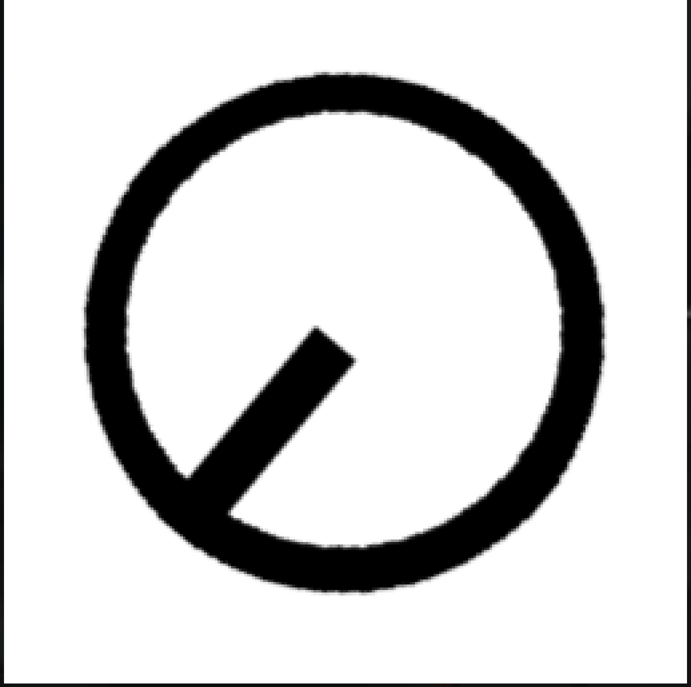
How it works…

- Toy example:



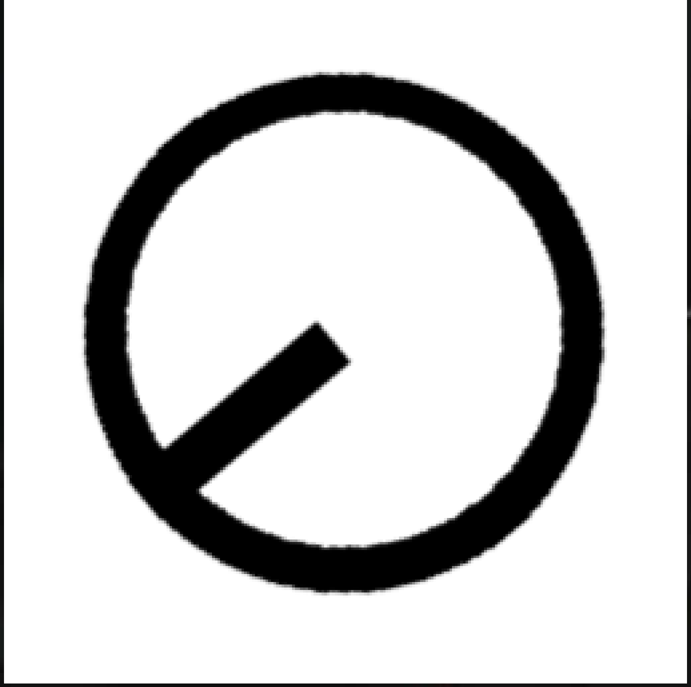
How it works…

- Toy example:



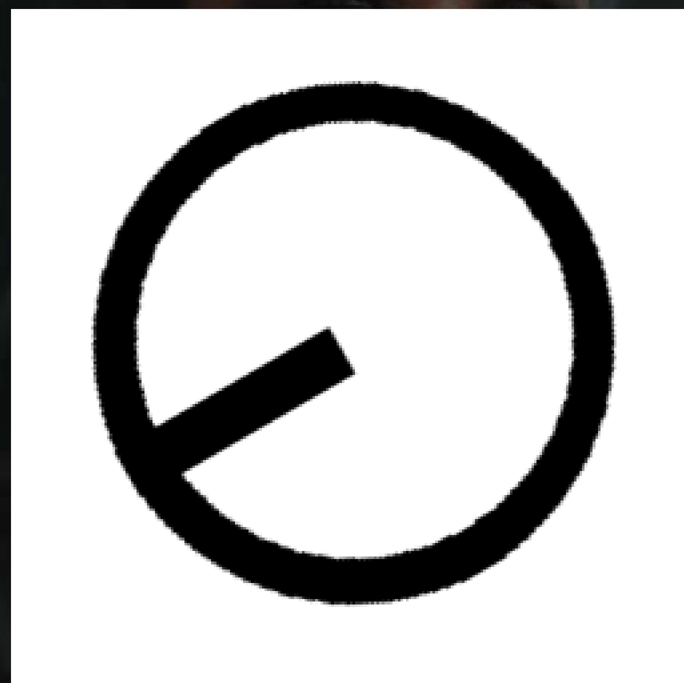
How it works…

- Toy example:



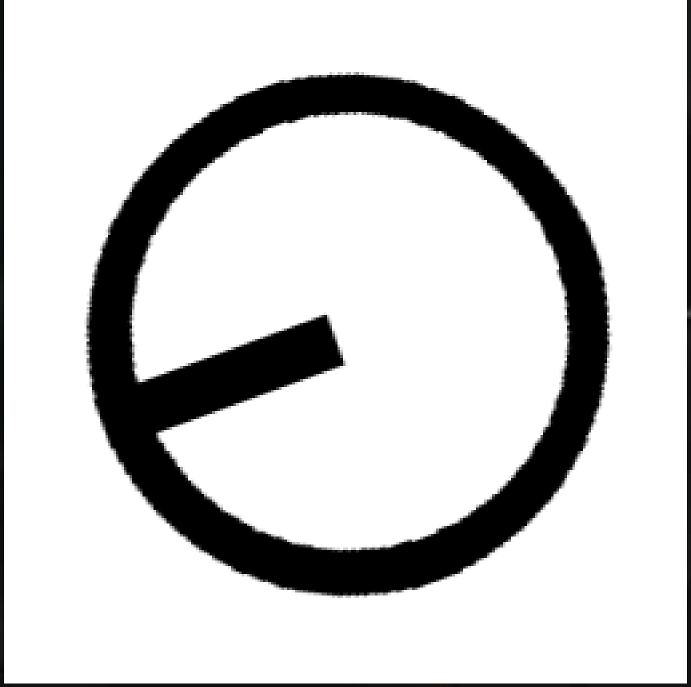
How it works…

- Toy example:



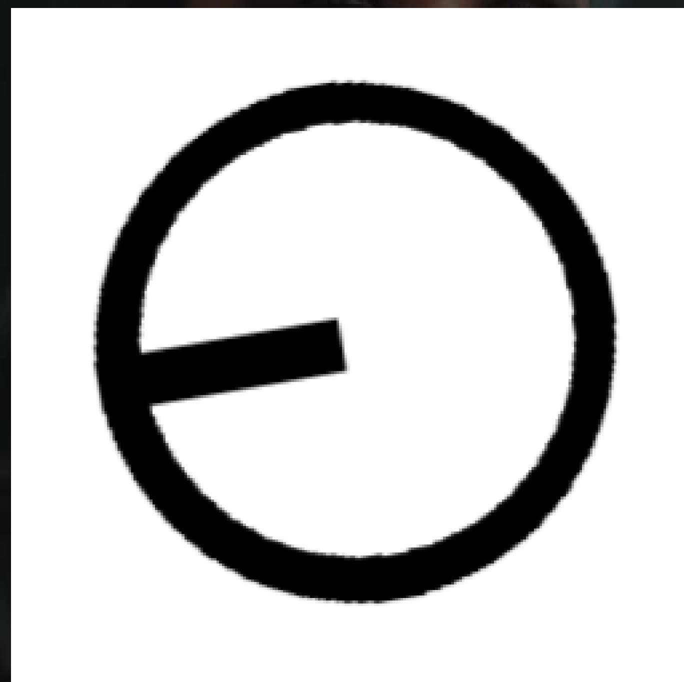
How it works…

- Toy example:



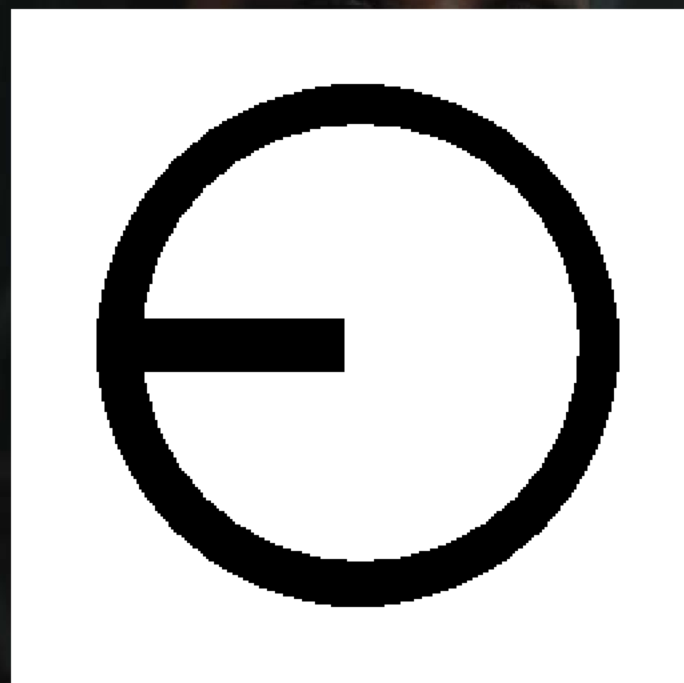
How it works…

- Toy example:



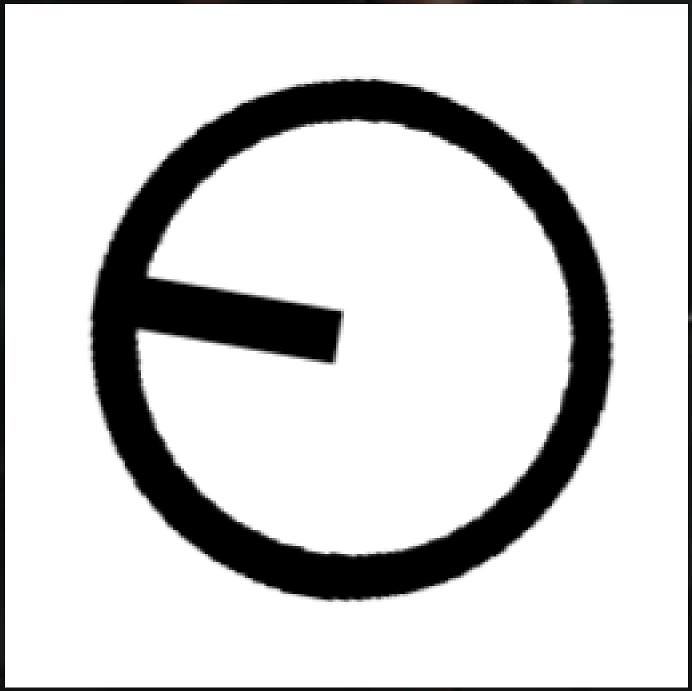
How it works…

- Toy example:



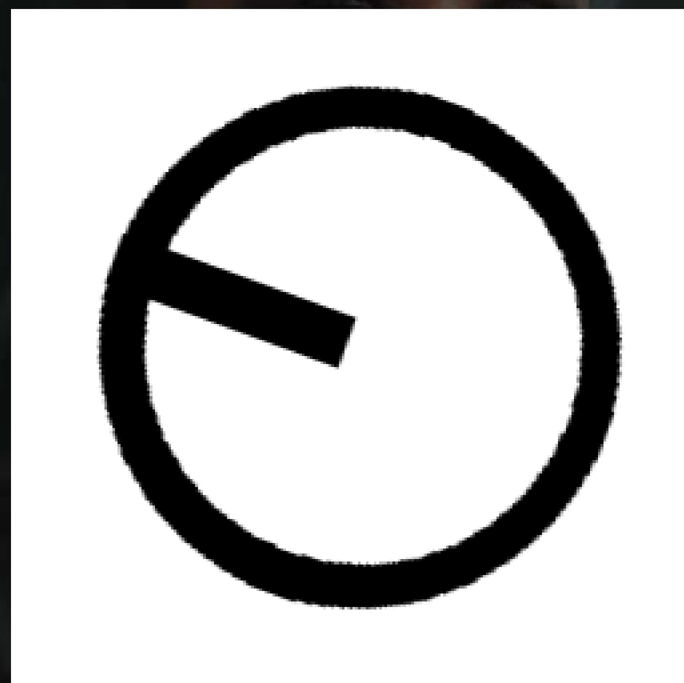
How it works…

- Toy example:



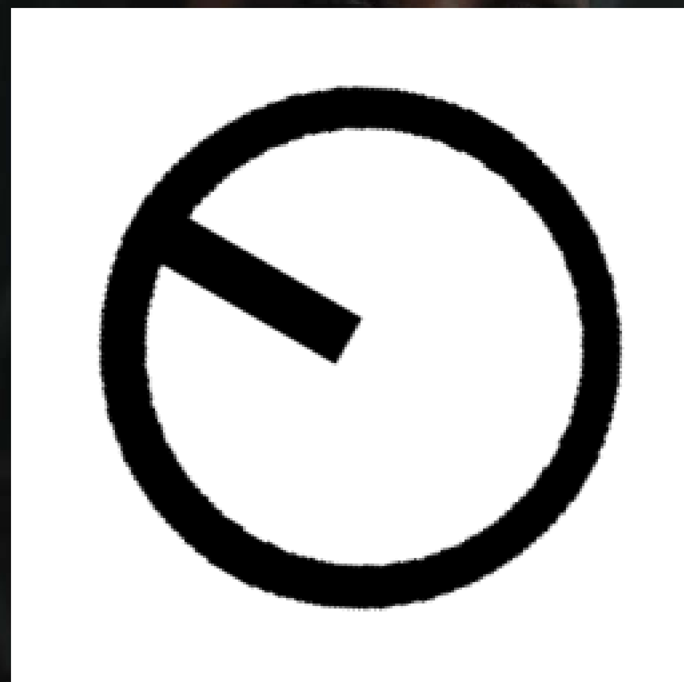
How it works…

- Toy example:



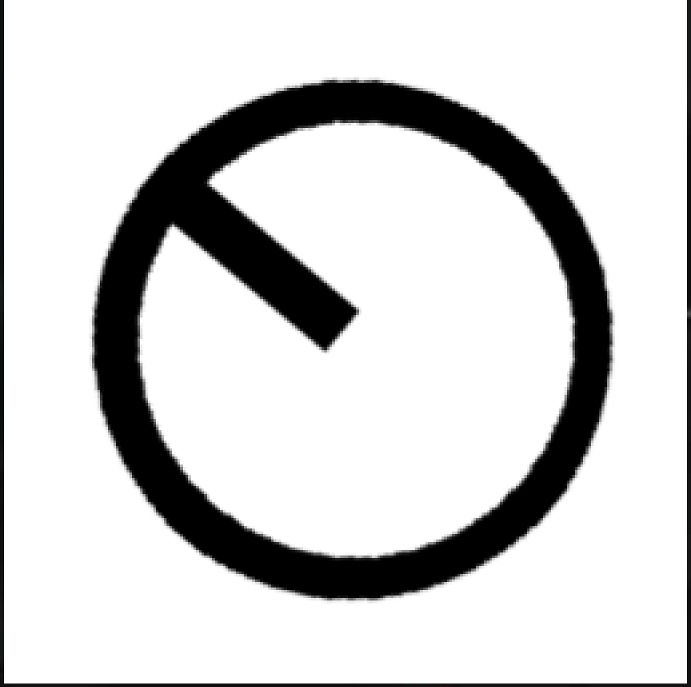
How it works…

- Toy example:



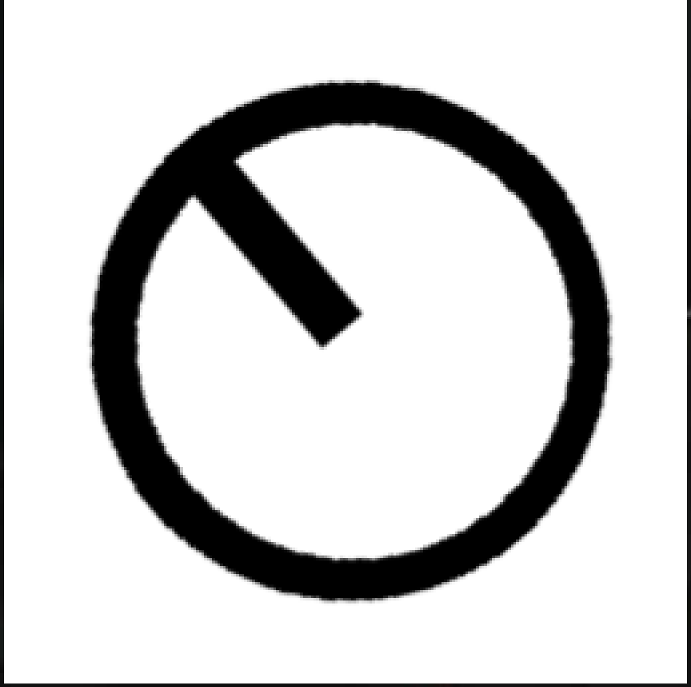
How it works…

- Toy example:



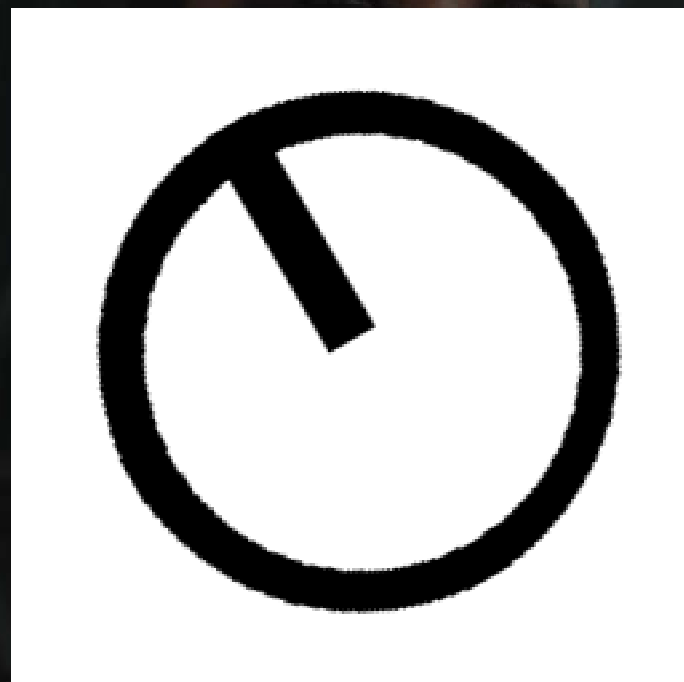
How it works…

- Toy example:



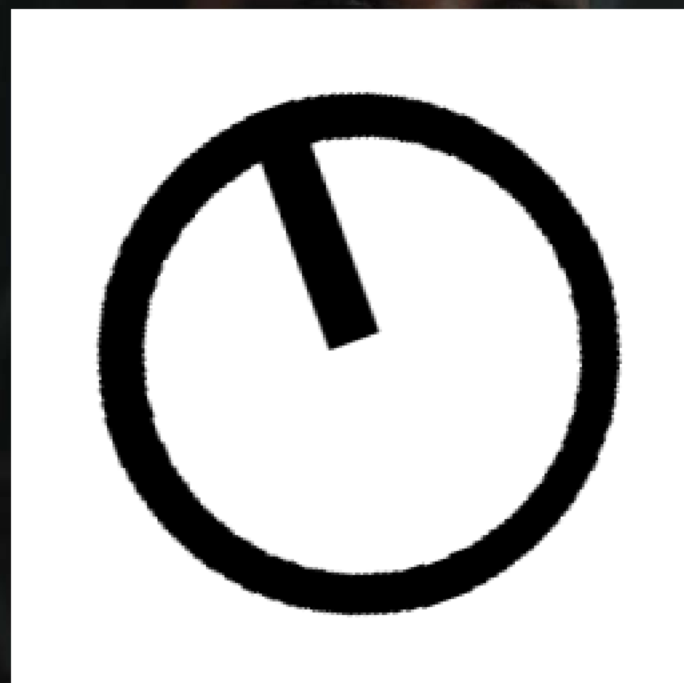
How it works…

- Toy example:



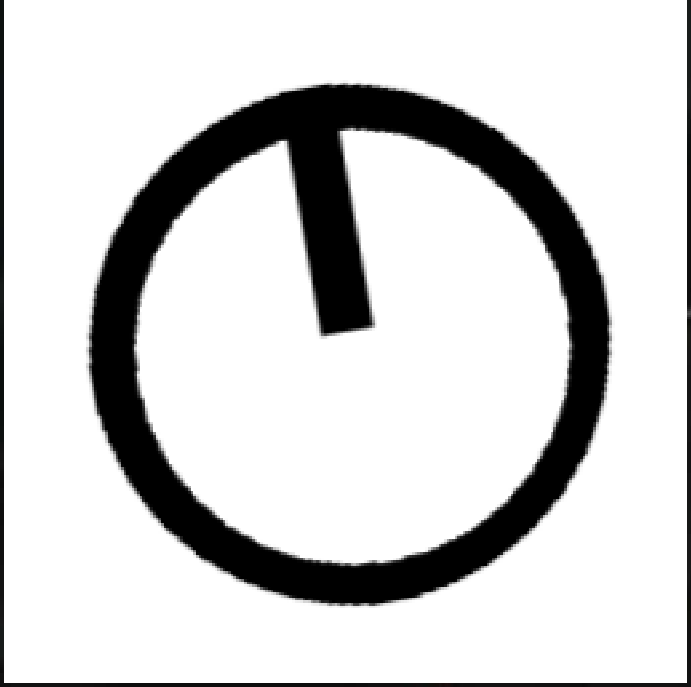
How it works…

- Toy example:



How it works…

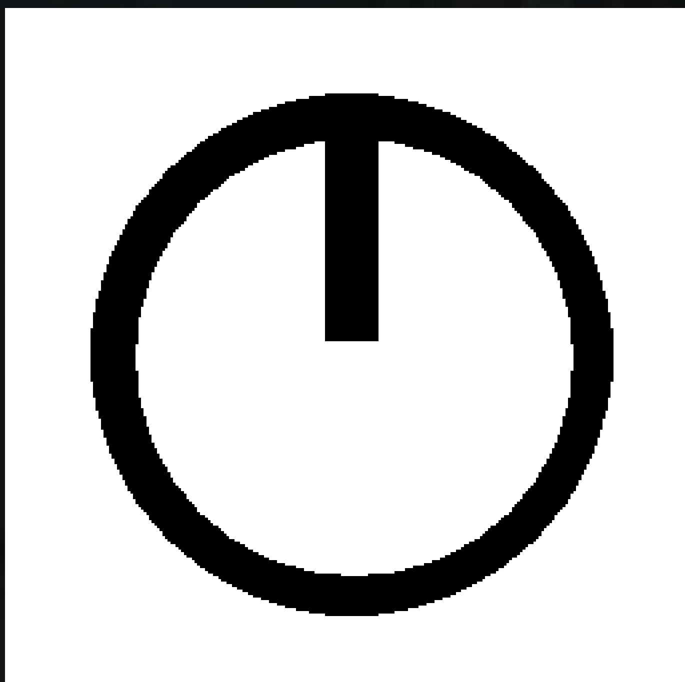
- Toy example:



Toy example

- 36, 256x256 images ~ 9 Mbytes

Alternate representation



&

Θ = { 0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150, 160, 170, 180, 190, 200, 210, 220, 230, 240, 250, 260, 270, 280, 290, 300, 310, 320, 330, 340, 350 }

Alternate representation

1, 256x256 image + 36 angles

=

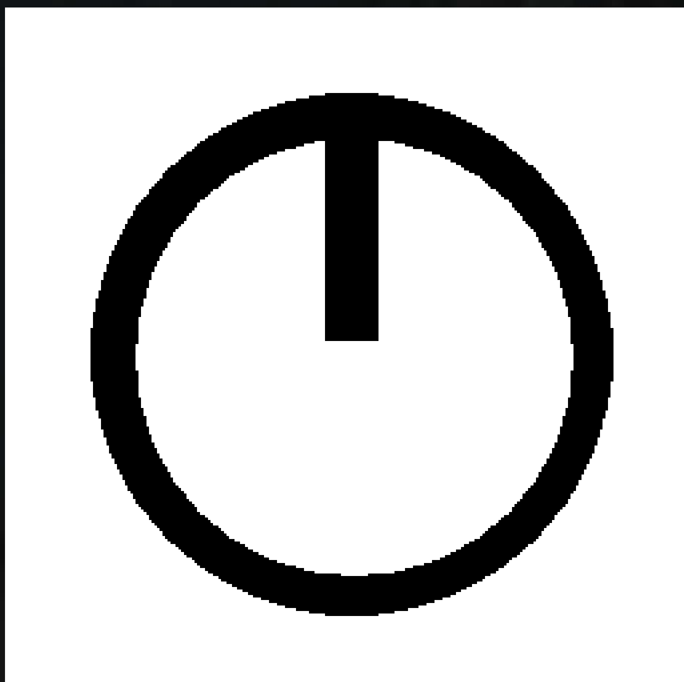
64 Kbytes

=

144:1 compression

This is PCA?

Component(s)

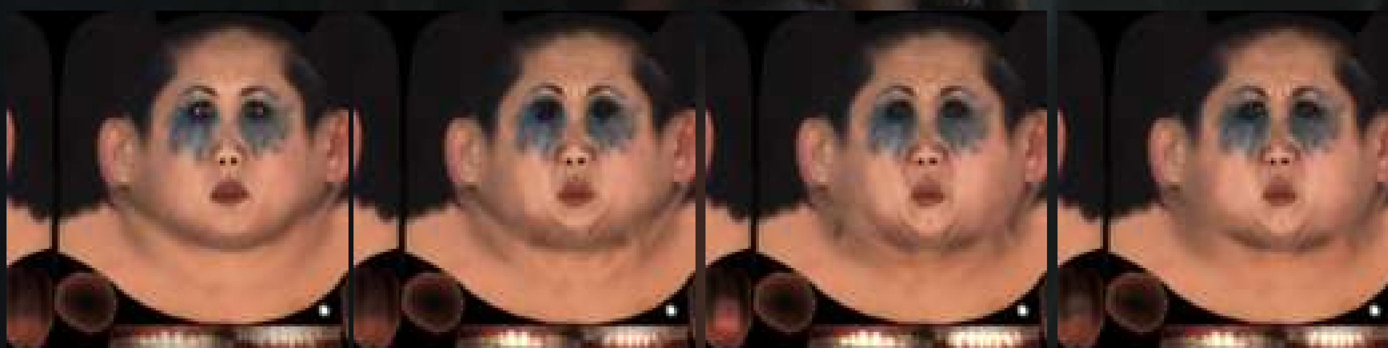


&

Weights

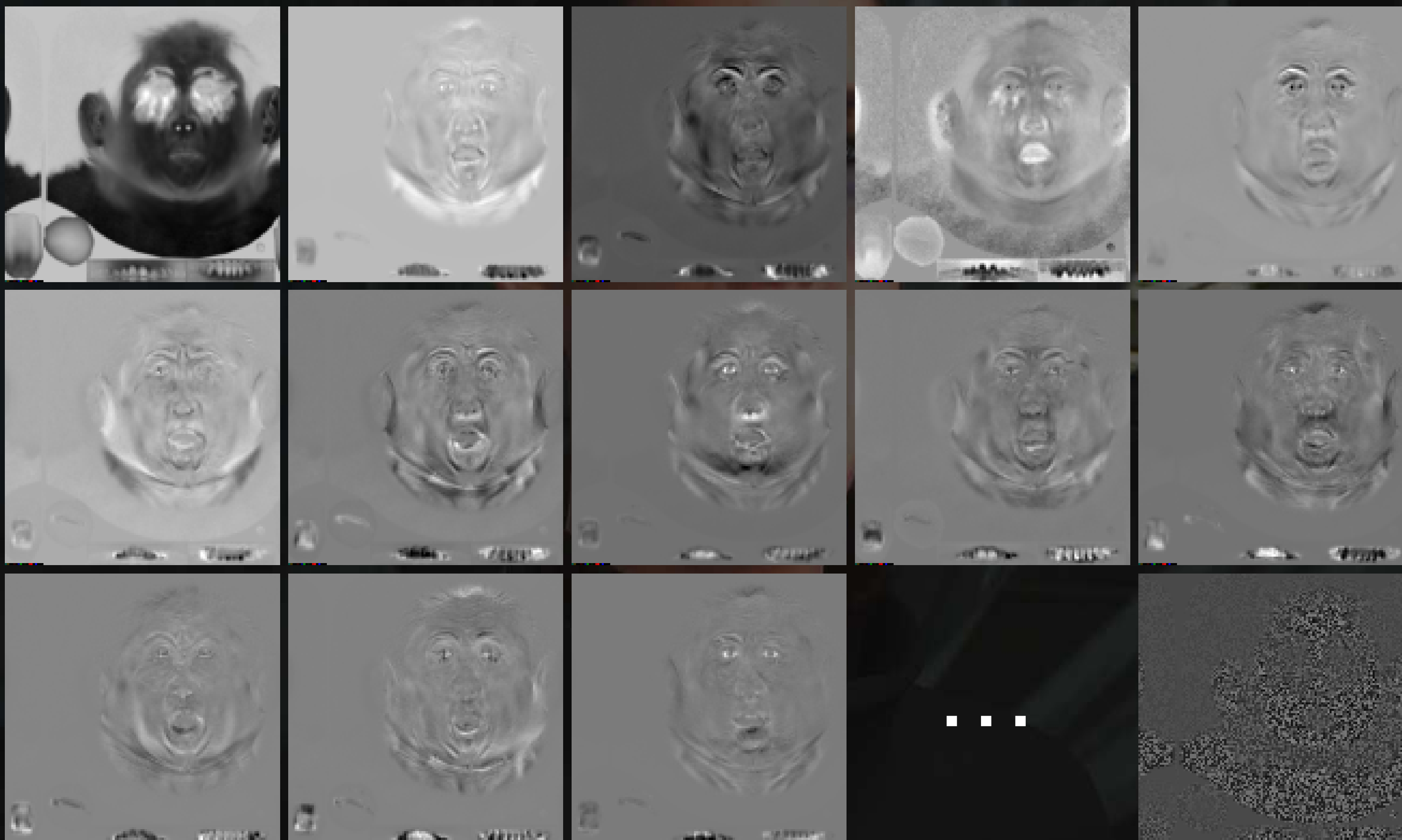
Θ = { 0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150, 160, 170, 180, 190, 200, 210, 220, 230, 240, 250, 260, 270, 280, 290, 300, 310, 320, 330, 340, 350 }

Real data: not so nice



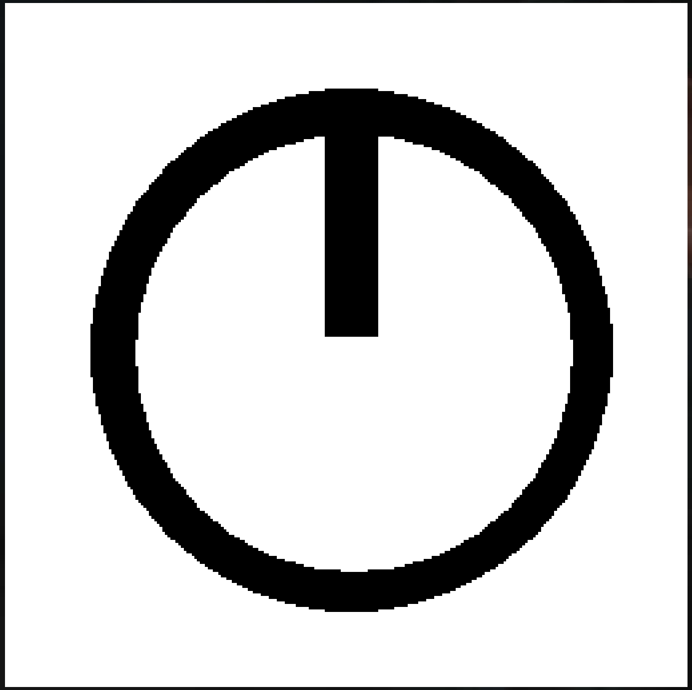
...

Data Components



Alternate representation

Component(s)



&

Weights

$\Theta = \{ 0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150, 160, 170, 180, 190, 200, 210, 220, 230, 240, 250, 260, 270, 280, 290, 300, 310, 320, 330, 340, 350 \}$

Alternate representation

Component(s)

Weights

$\Theta = \{ w_0, w_1, w_2, w_3, \ldots \}$

&

C-dimensional weight vectors

C components

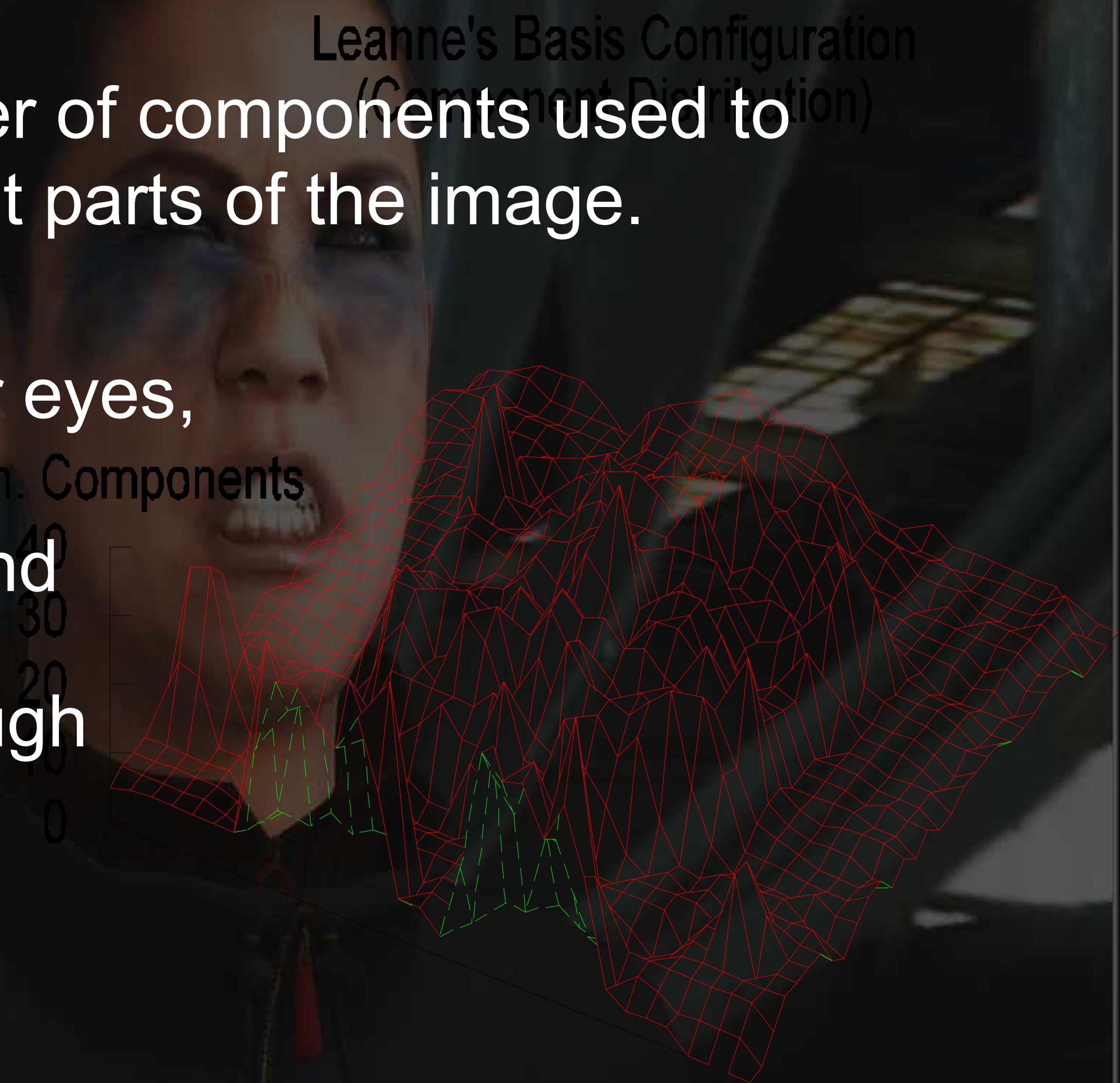
(For UCap, C ~ 16)

GPU Implementation

- Components <u>are static</u>
  - Live in GPU as textures, vertex attributes
- Weight vectors <u>are small</u>
  - Uploaded per frame
- Dot product is cheap

Variable Representation

- Modify the number of components used to represent different parts of the image.
- E.g., more representation for eyes, mouth, forehead
- Both automatic and artist-controlled optimization through weighting maps



Leanne Example

Component distribution

Source

PCA
(equivalent size)

Decompressed
result

Explosion Example

Component Usage

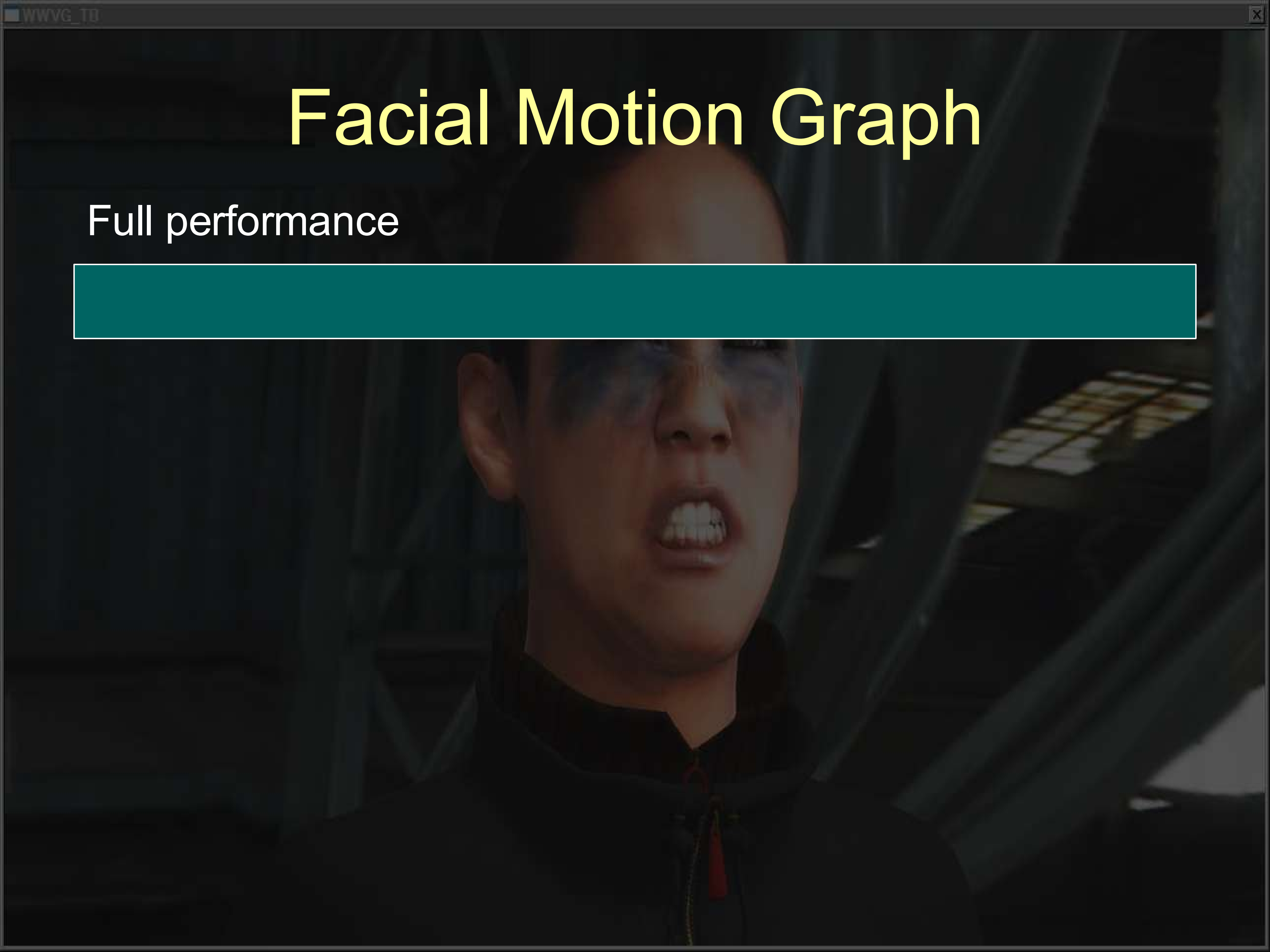
1) Original Data
2) Compressed to 1/3 size using PCA
3) Compressed to 1/12 using PCA
4) Compressed to 1/12 using VPCA

Compression Conclusion

- PCA/VPCA compression technique
  - High compression ratio (UCap performance in ~8Mbytes)
  - Low bandwidth requirements (16 float upload per frame)
  - Low decompression complexity (1 dot product of a 16D vector)

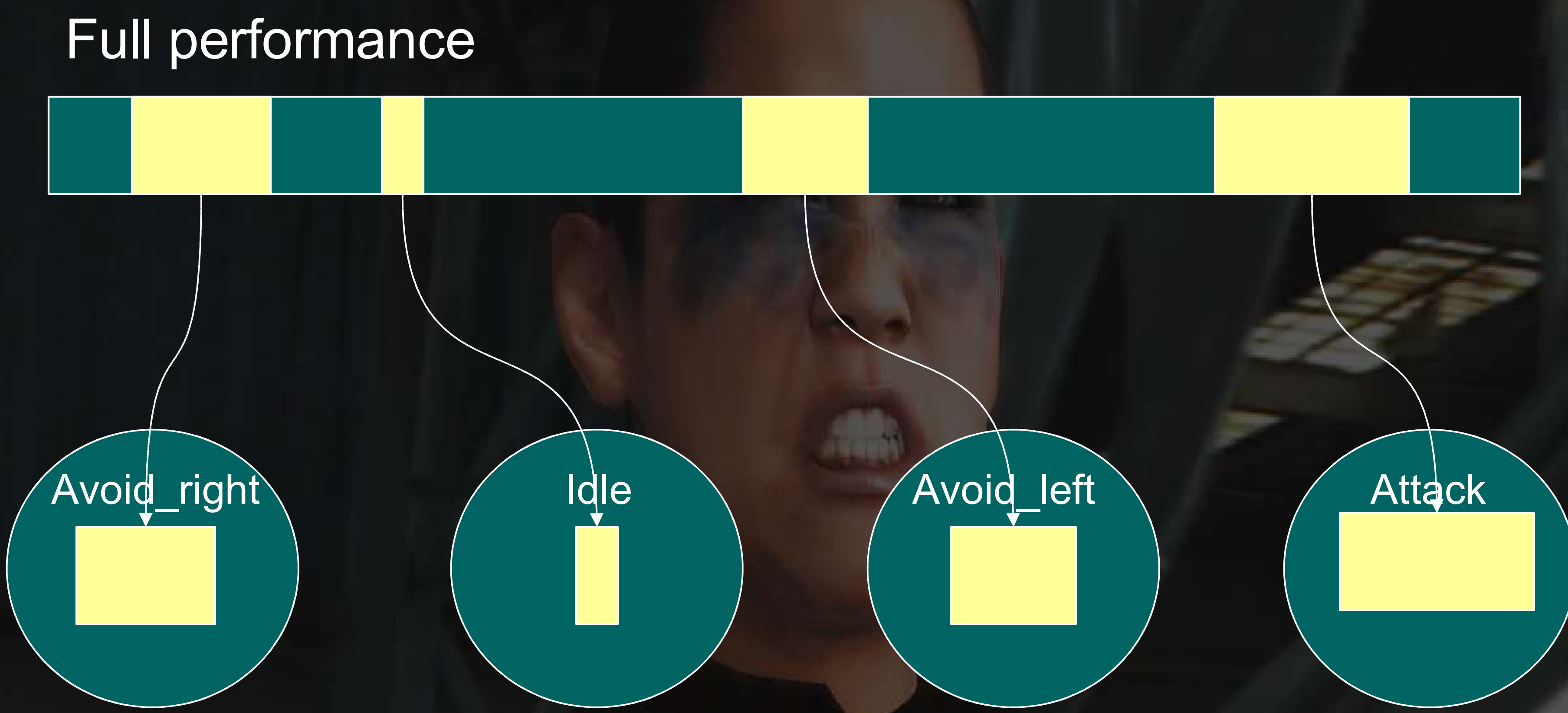
UCap Sequencing

- Segment captured sequences and form triggerable state machine
- ANT authoring
- Geometry and texture blending over transitions
  - Pre-decompression blending (component weights)


WWVG_TD
Facial Motion Graph
Full performance

Facial Motion Graph

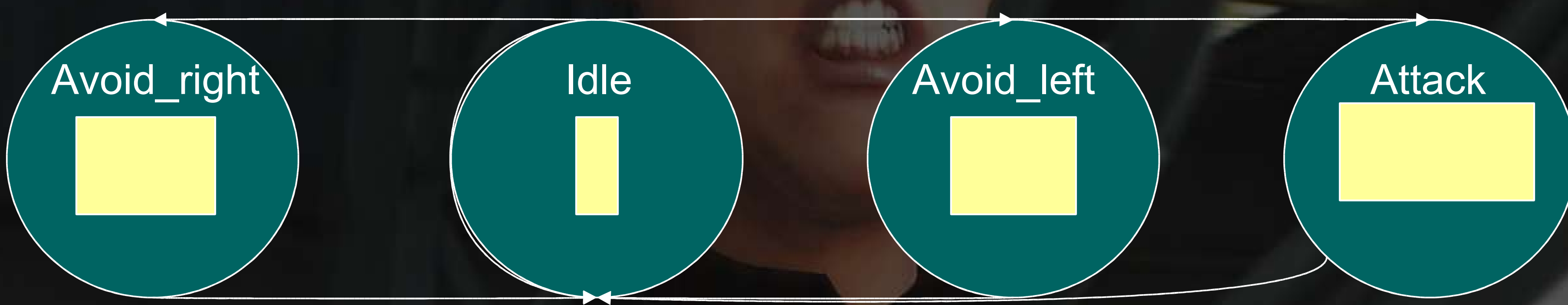
Facial Motion Graph



Expressions become states in state machine

Facial Motion Graph

State machine triggered (AI/game pad/etc.) to sequence facial expressions



Geometry & Texture blended across transitions