Image alignment and tracking

Outline

- 2D transformations
- Direct methods
- Lucas Kanade

Geometric transformations

Think of an image as a function

Input (domain): pixel coordinates

Output (range): pixel intensities

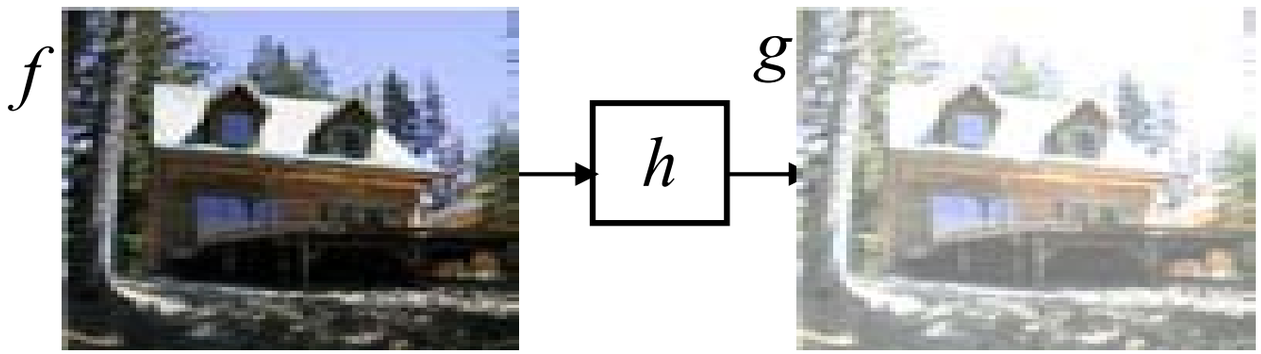
Filtering: transform the output range

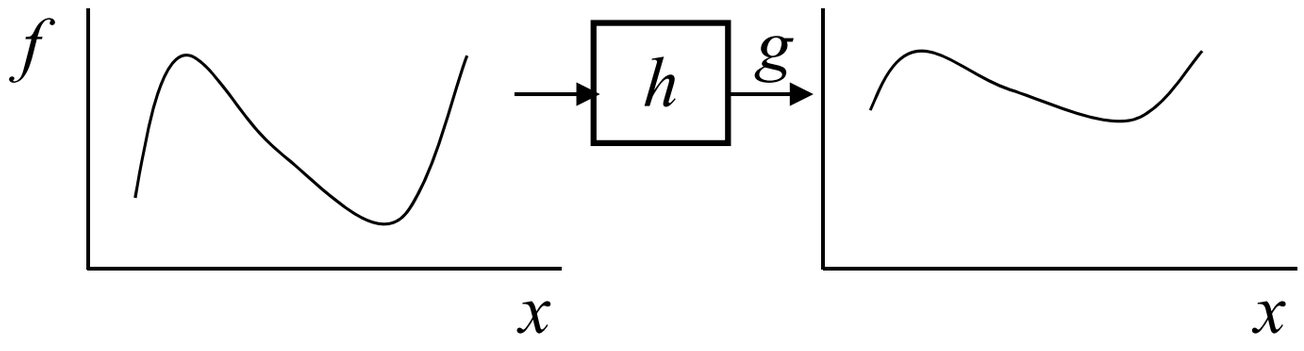
Goemetric warps : transform the input domain

Szeliski Sec. 3.6

Geometric transformations

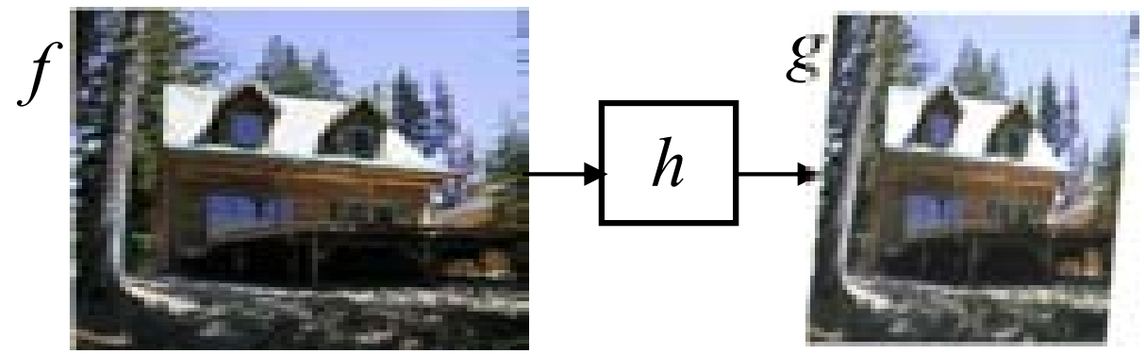
Filtering

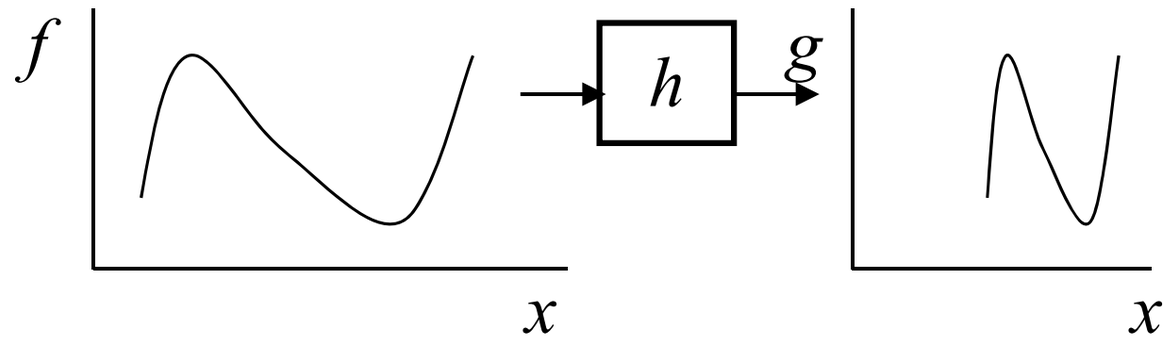




$$g(\mathbf{x}) = h(\mathbf{x}) * f(\mathbf{x})$$

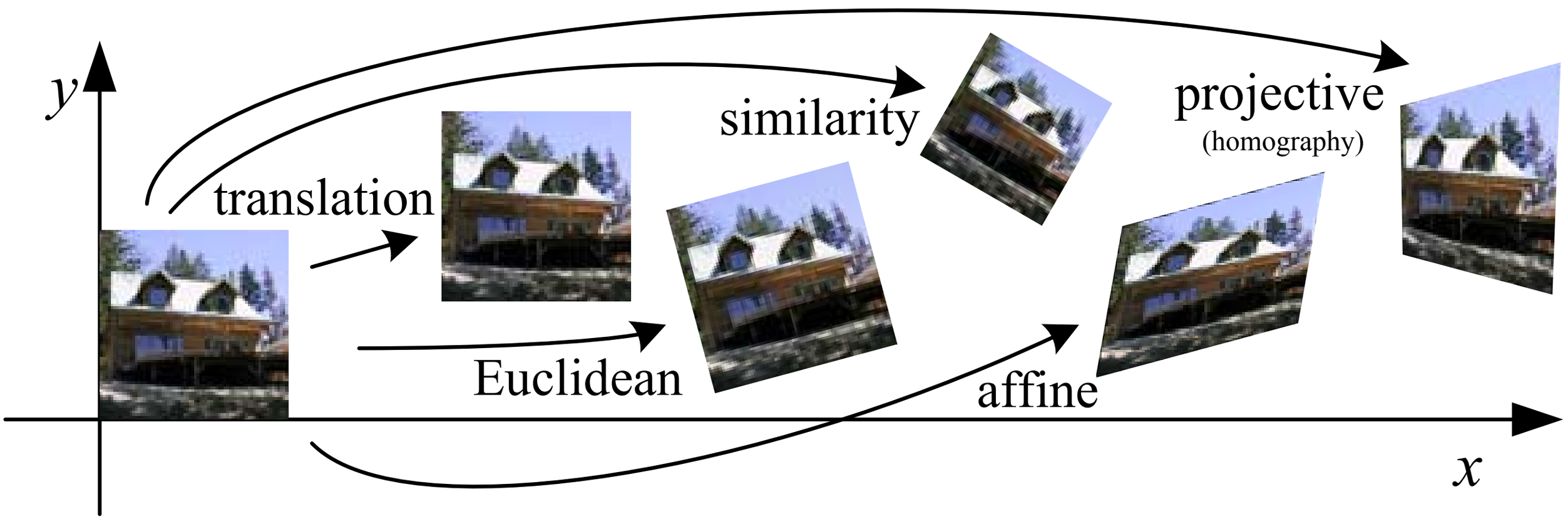
Warping



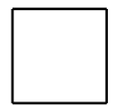


$$g(\mathbf{x}) = f(h(\mathbf{x}))$$

Family of 2D warps



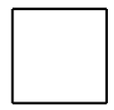
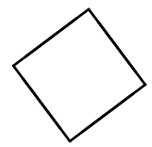
Family of 2D warps

Transformation	Matrix	# DoF	Preserves	Icon
translation	$\begin{bmatrix} \boldsymbol{I} \mid \boldsymbol{t} \end{bmatrix}_{2\times 3}$	2	orientation	

$$x' = x + t_x$$
$$y' = y + t_y$$

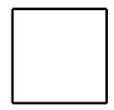
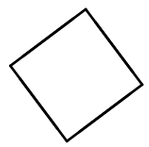
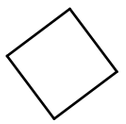
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Family of 2D warps

Transformation	Matrix	# DoF	Preserves	Icon
translation	$\left[\begin{array}{c\|c} \boldsymbol{I} & \boldsymbol{t} \end{array}\right]_{2\times 3}$	2	orientation	
rigid (Euclidean)	$\left[\begin{array}{c\|c} \boldsymbol{R} & \boldsymbol{t} \end{array}\right]_{2\times 3}$	3	lengths	

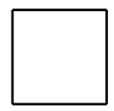
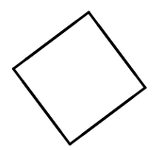
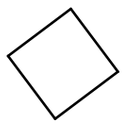
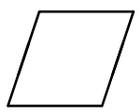
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Family of 2D warps

Transformation	Matrix	# DoF	Preserves	Icon
translation	$\left[\begin{array}{c\|c} \boldsymbol{I} & \boldsymbol{t} \end{array}\right]_{2\times3}$	2	orientation	
rigid (Euclidean)	$\left[\begin{array}{c\|c} \boldsymbol{R} & \boldsymbol{t} \end{array}\right]_{2\times3}$	3	lengths	
similarity	$\left[\begin{array}{c\|c} s\boldsymbol{R} & \boldsymbol{t} \end{array}\right]_{2\times3}$	4	angles	

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s\cos\theta & -\sin\theta & t_x \\ \sin\theta & s\cos\theta & t_y \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Family of 2D warps

Transformation	Matrix	# DoF	Preserves	Icon
translation	$\left[\begin{array}{c\|c} \boldsymbol{I} & \boldsymbol{t} \end{array}\right]_{2\times 3}$	2	orientation	
rigid (Euclidean)	$\left[\begin{array}{c\|c} \boldsymbol{R} & \boldsymbol{t} \end{array}\right]_{2\times 3}$	3	lengths	
similarity	$\left[\begin{array}{c\|c} s\boldsymbol{R} & \boldsymbol{t} \end{array}\right]_{2\times 3}$	4	angles	
affine	$\left[\begin{array}{c} \boldsymbol{A} \end{array}\right]_{2\times 3}$	6	parallelism	

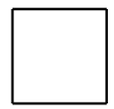
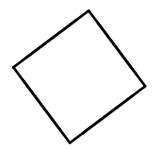
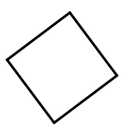
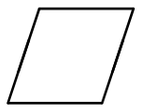
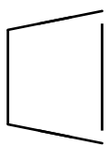
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Relates the image projections of the same 3D scene under 2 affine cameras

Think of as change of basis and 2D translation

Why not call this a “linear” transformation?

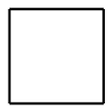
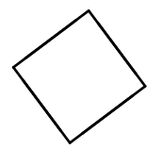
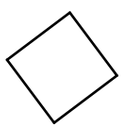
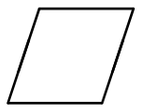
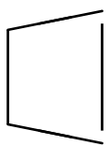
Family of 2D warps

Transformation	Matrix	# DoF	Preserves	Icon
translation	$\left[\begin{array}{c\|c} \boldsymbol{I} & \boldsymbol{t} \end{array}\right]_{2\times 3}$	2	orientation	
rigid (Euclidean)	$\left[\begin{array}{c\|c} \boldsymbol{R} & \boldsymbol{t} \end{array}\right]_{2\times 3}$	3	lengths	
similarity	$\left[\begin{array}{c\|c} s\boldsymbol{R} & \boldsymbol{t} \end{array}\right]_{2\times 3}$	4	angles	
affine	$\left[\begin{array}{c} \boldsymbol{A} \end{array}\right]_{2\times 3}$	6	parallelism	
projective	$\left[\begin{array}{c} \tilde{\boldsymbol{H}} \end{array}\right]_{3\times 3}$	8	straight lines	

$$\lambda \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

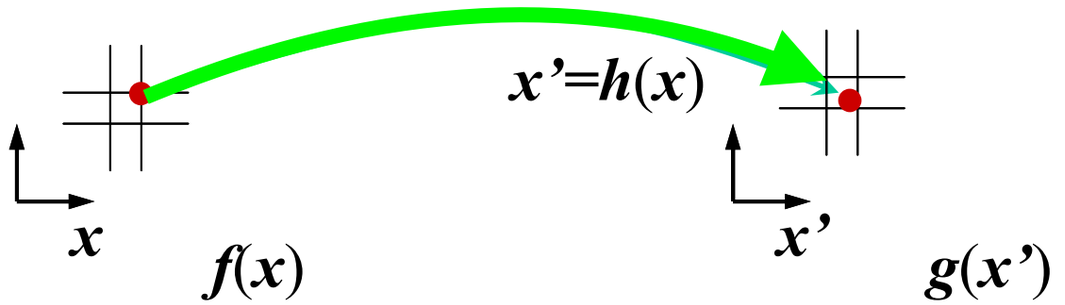
Relates the image projections of
(1) planar scene under pinhole cameras or (2) any scene under rotated cameras

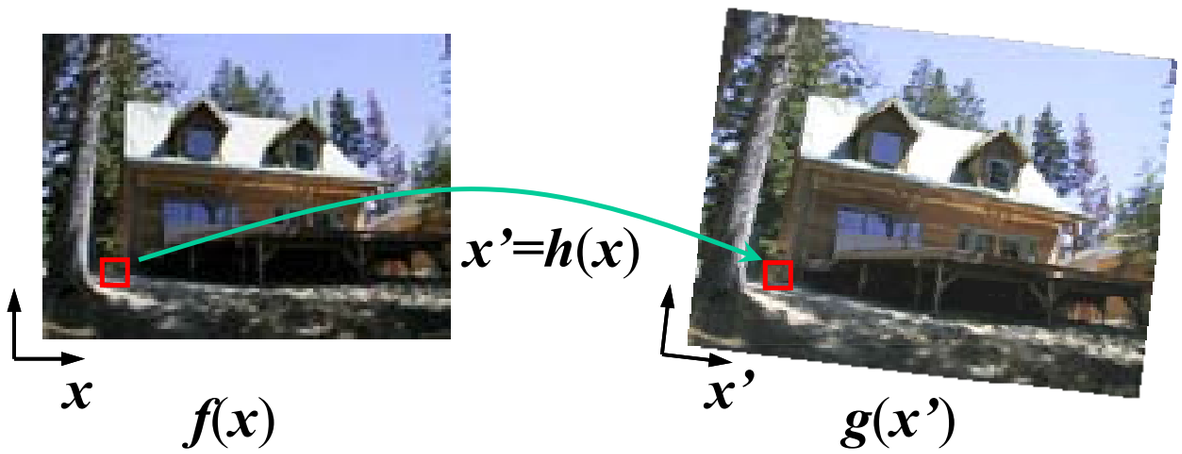
Family of 2D warps

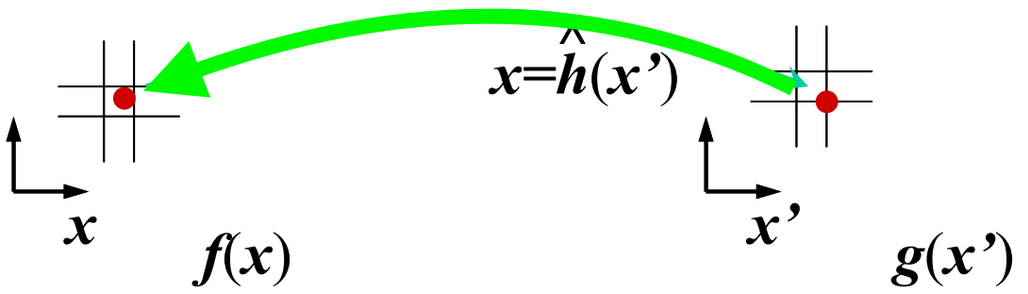
Transformation	Matrix	# DoF	Preserves	Icon
translation	$\left[\begin{array}{c\|c} \boldsymbol{I} & \boldsymbol{t} \end{array}\right]_{2\times 3}$	2	orientation	
rigid (Euclidean)	$\left[\begin{array}{c\|c} \boldsymbol{R} & \boldsymbol{t} \end{array}\right]_{2\times 3}$	3	lengths	
similarity	$\left[\begin{array}{c\|c} s\boldsymbol{R} & \boldsymbol{t} \end{array}\right]_{2\times 3}$	4	angles	
affine	$\left[\begin{array}{c} \boldsymbol{A} \end{array}\right]_{2\times 3}$	6	parallelism	
projective	$\left[\begin{array}{c} \tilde{\boldsymbol{H}} \end{array}\right]_{3\times 3}$	8	straight lines	

Do Euclidean transformations relate image projections of same scene under orthographic cameras?

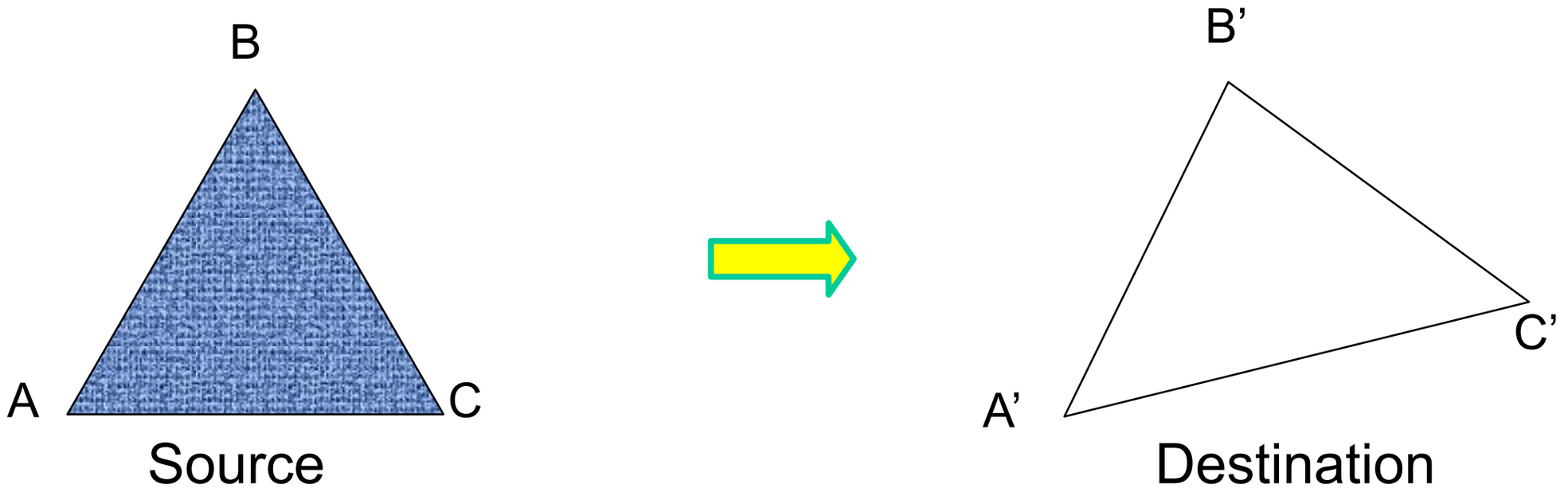
Forward vs inverse warping





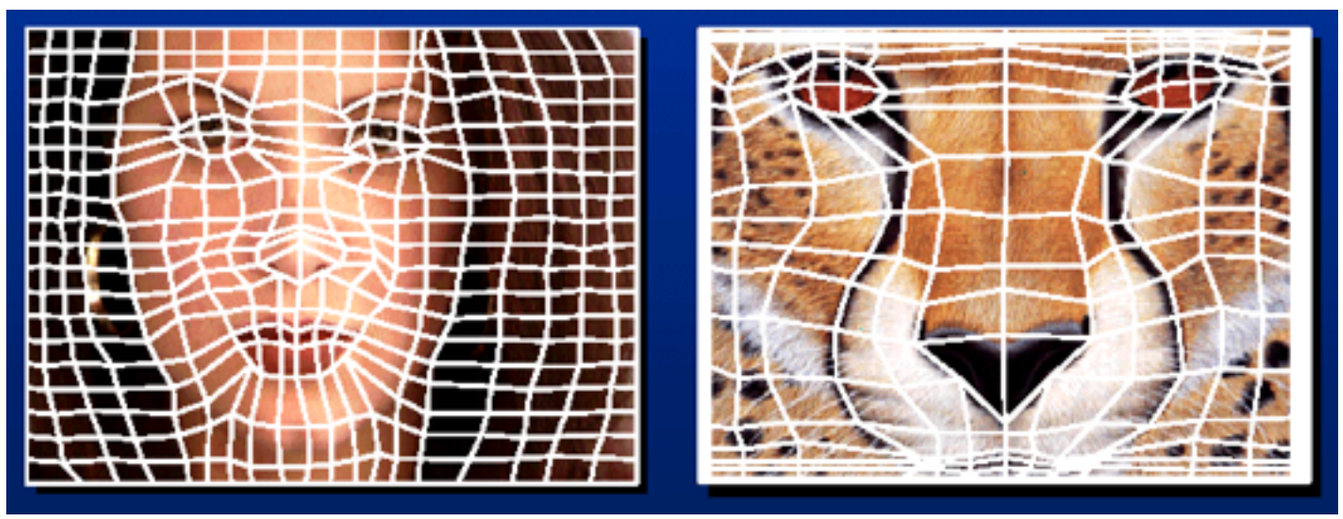


Example: warping triangles



What kind of transformation is this? How many DOFs?

Example application: image morphing



Piecewise affine warps (cut each quadrilateral into 2 triangles)

Example application: shape modeling

D'Arcy Thompson
"On Growth and Form" 1915

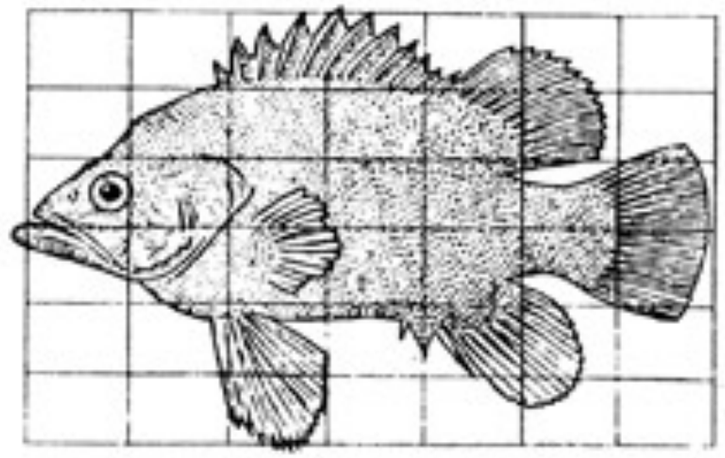


Fig. 150. *Polyprion.*

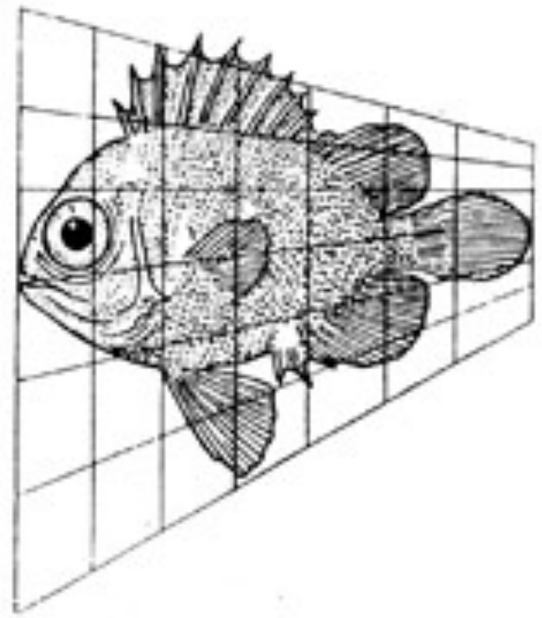


Fig. 151. *Pseudopriacanthus altus.*

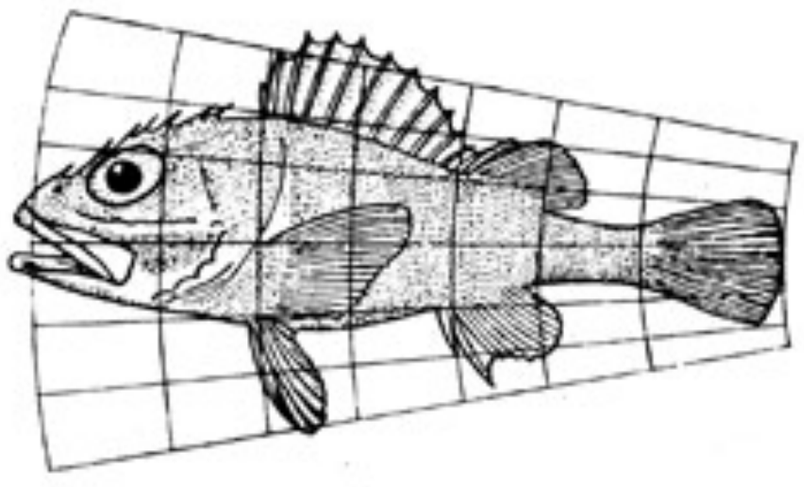


Fig. 152. *Scorpaena* sp.

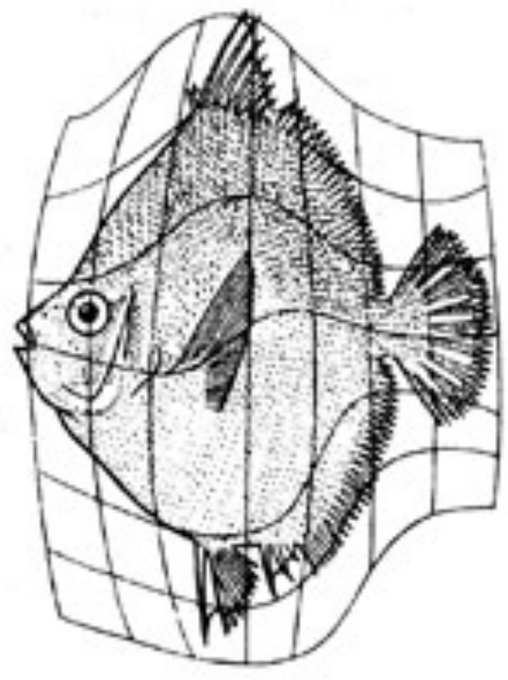
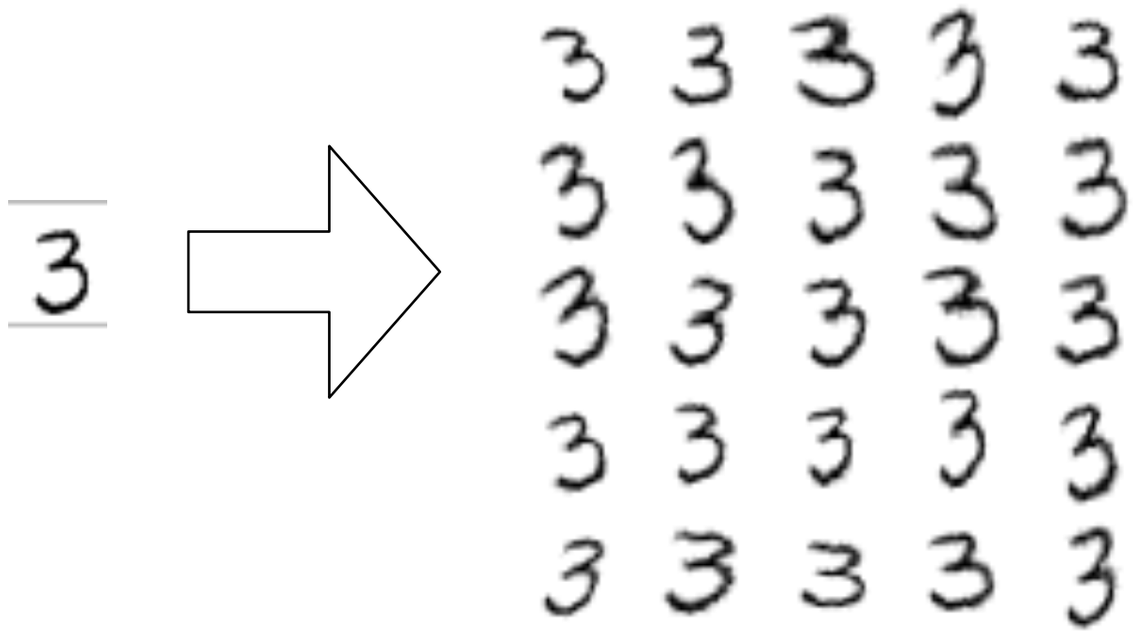


Fig. 153. *Antigonia capros.*

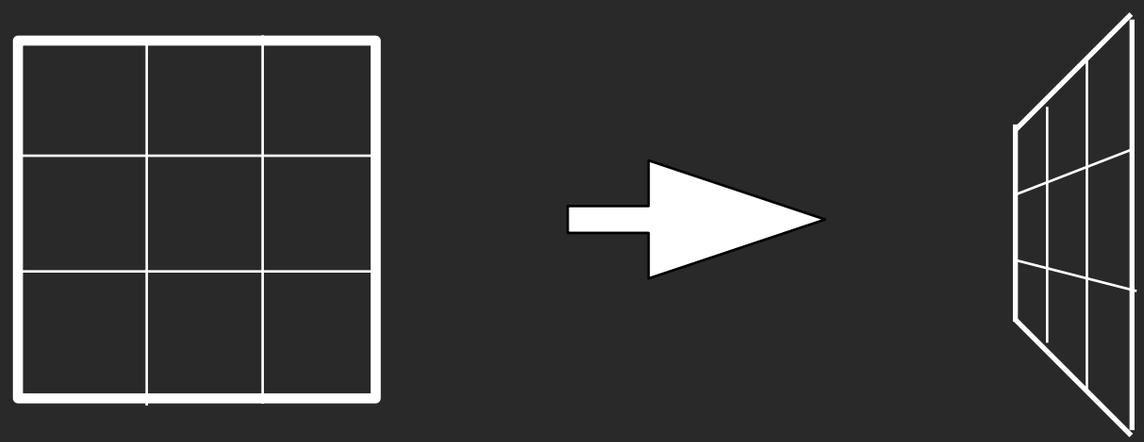
Example application: data augmentation



Hypothesis: we'll see more of this in the future!

Data augmentation by geometric warping

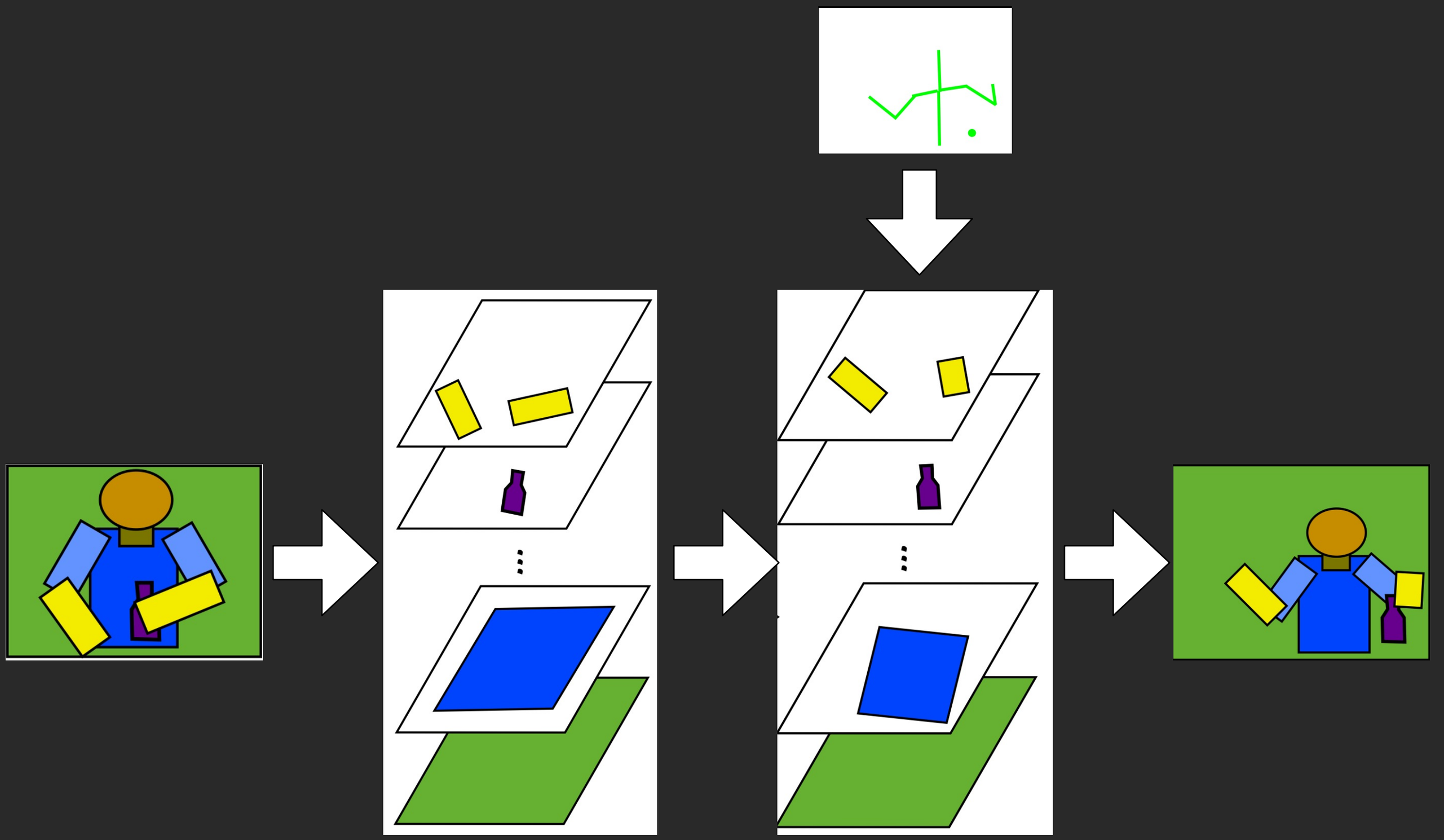
Cabinet Cargo Container Dryer Cardboard Box



Example application: interactive video tracking

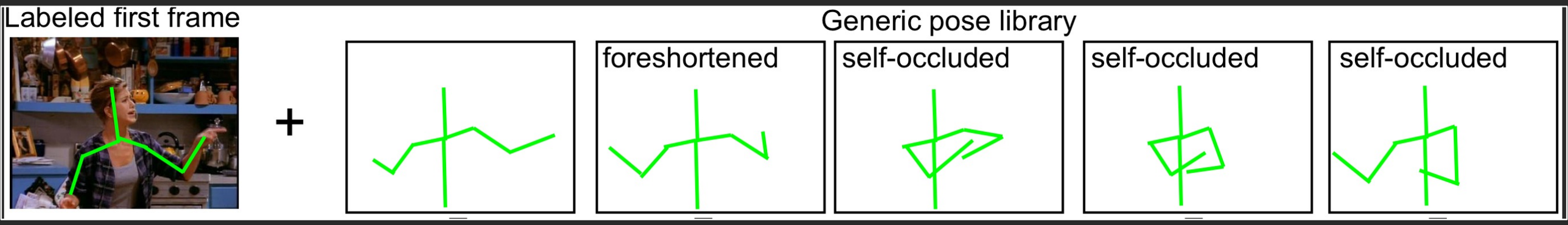
Labeled first frame

Layered (2.1D) warps



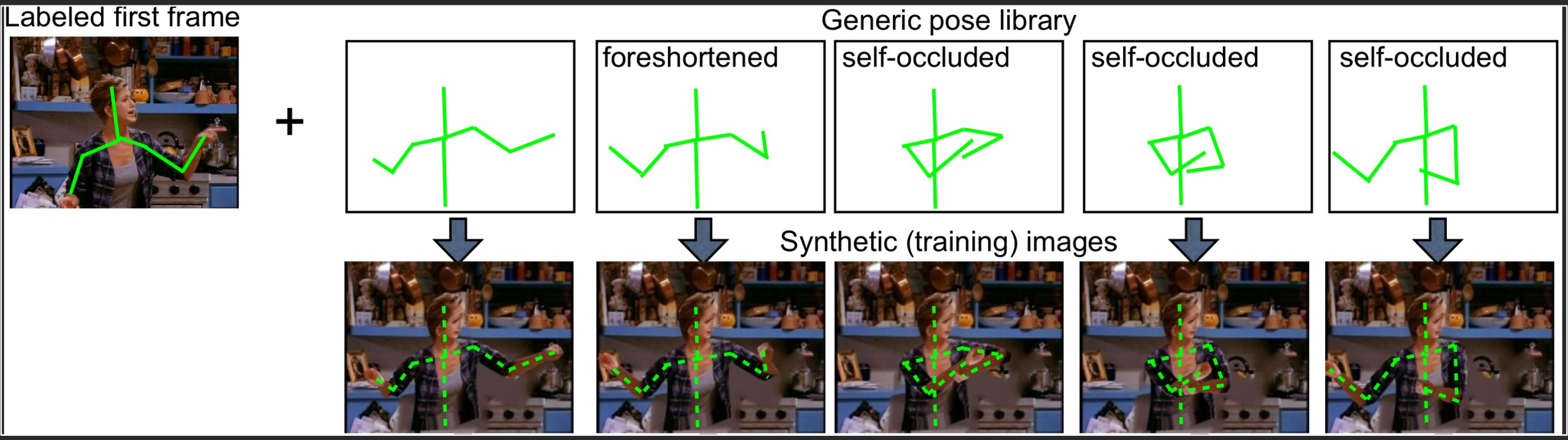
Similarity transformations

Overall pipeline

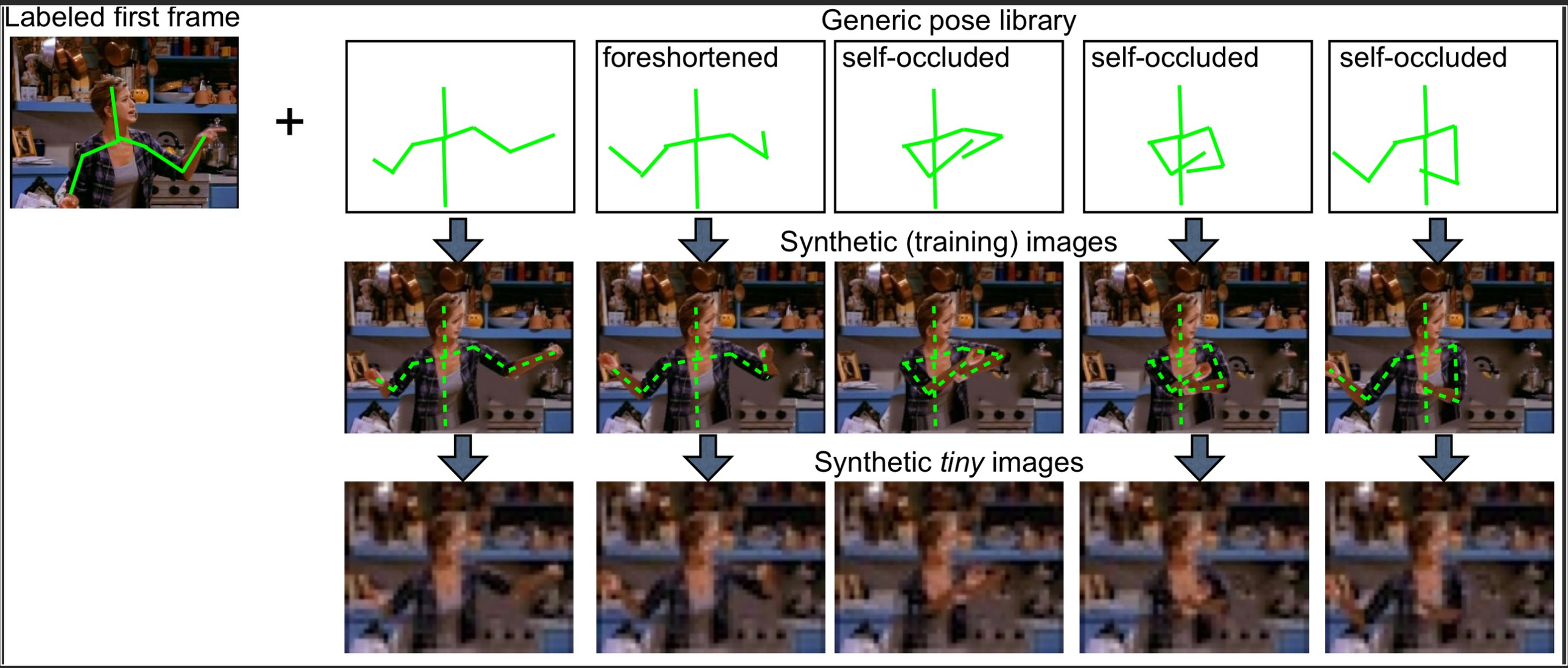


(lots of domain knowledge)

Overall pipeline



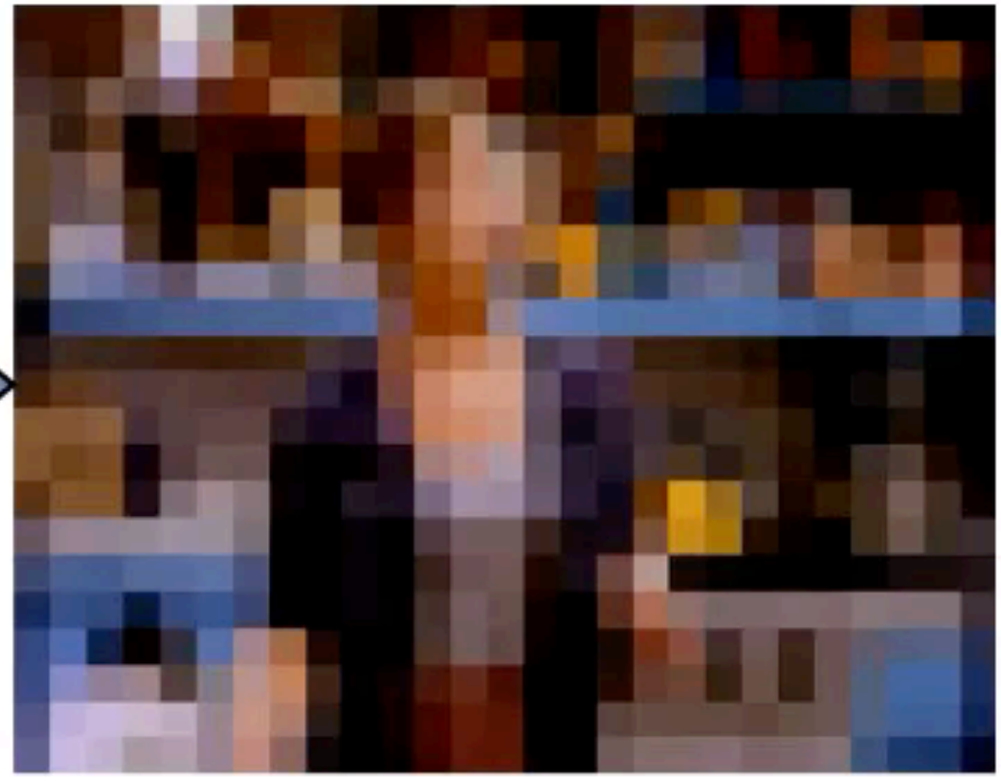
Overall pipeline



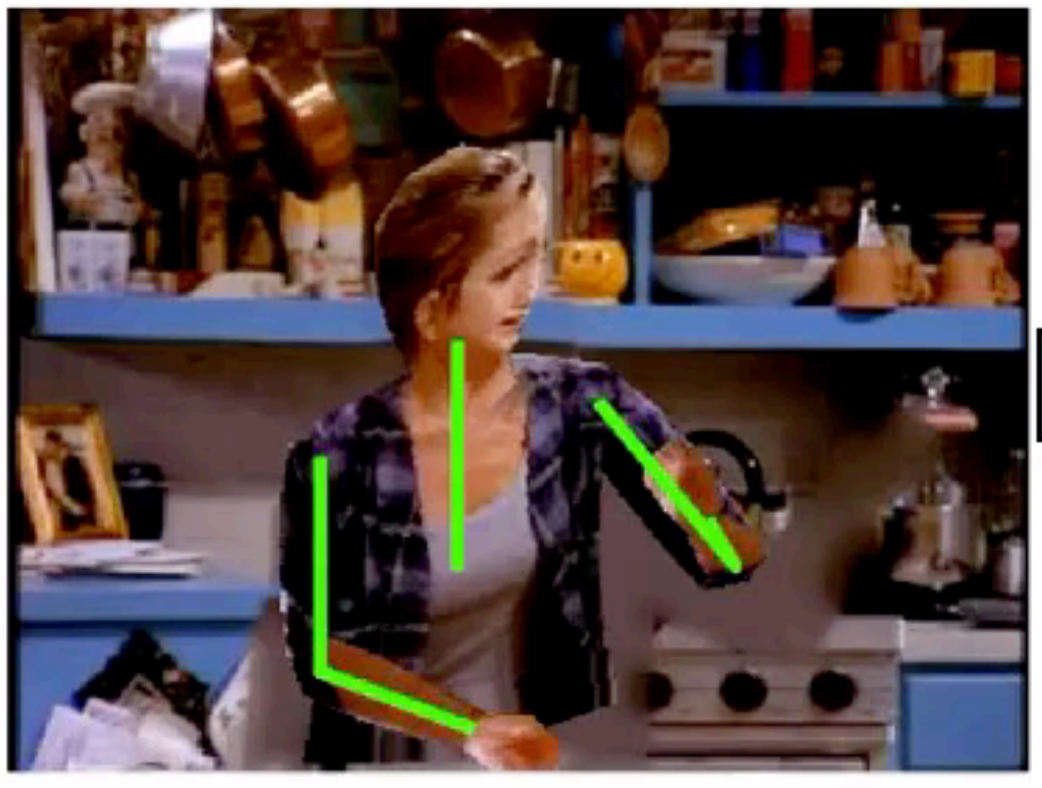
original

low resolution

test video

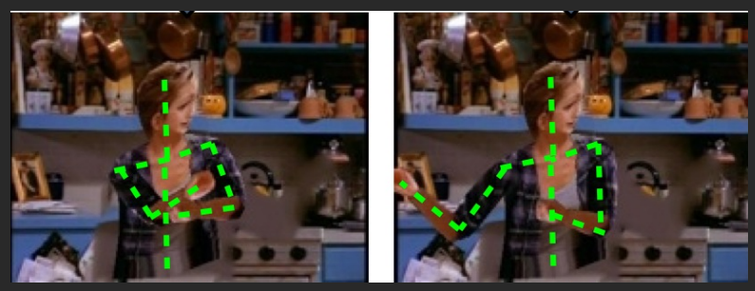


best match

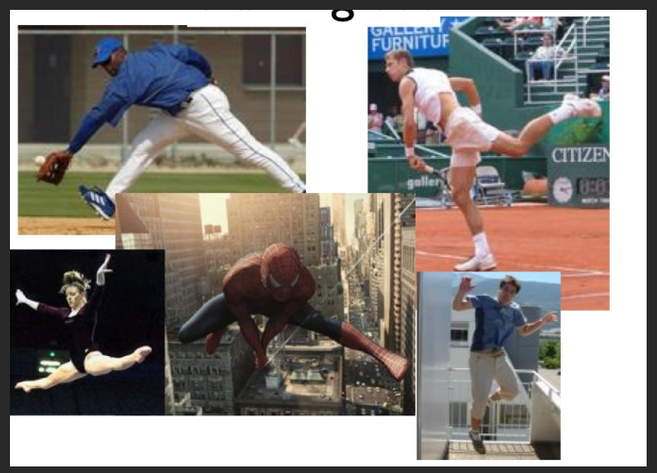


Some observations

1. Layered pixel models are particularly effective for occlusions



2. We do not need to synthesis appearance variations for a video

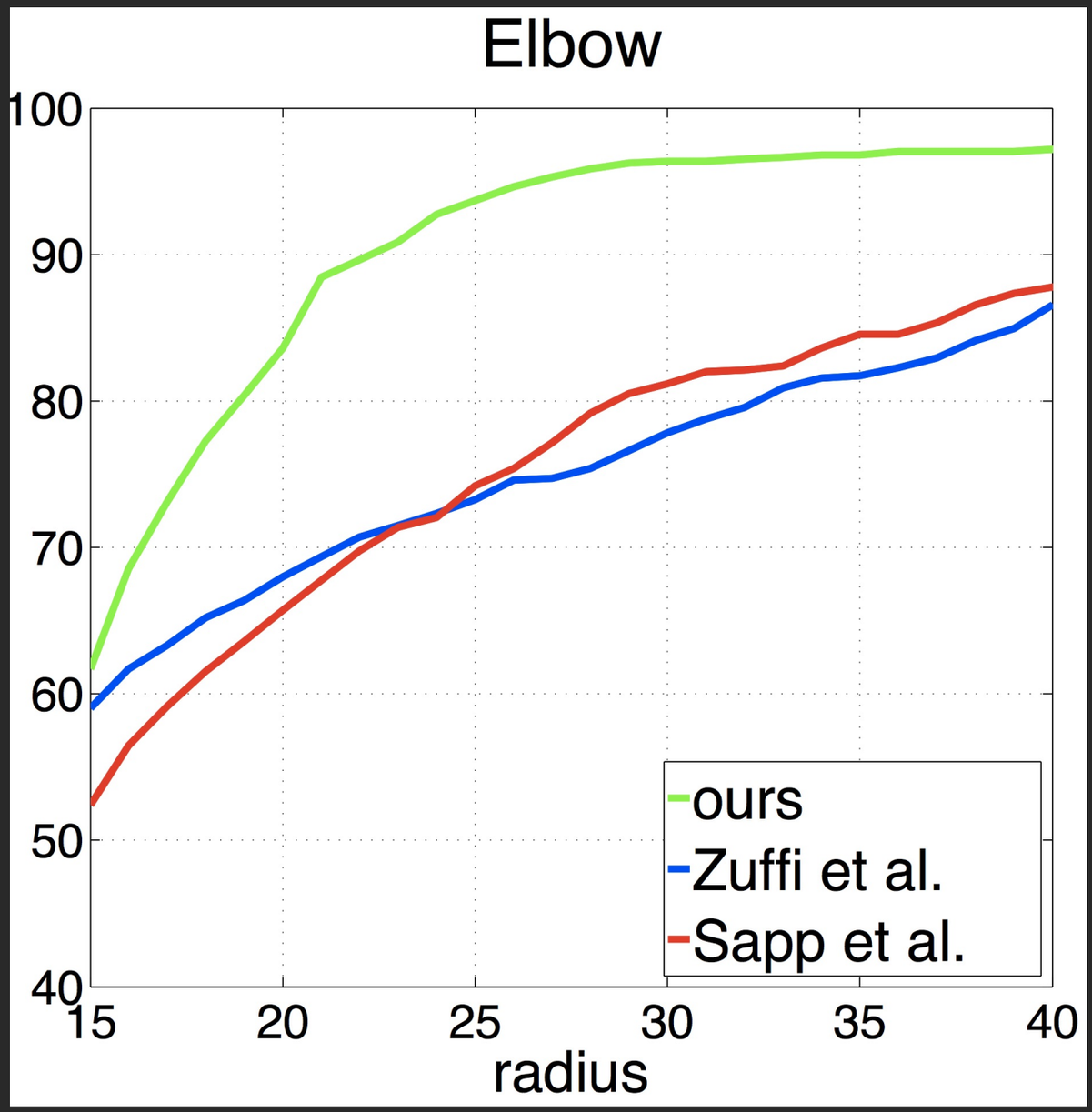


General training set

Video-specific training set

Surprisingly effective

% of correctly-predicted elbows



Outline

- 2D transformations
- Direct methods
- Lucas Kanade

Debate at ICCV 1999

In the present context, we define "Direct Methods" as methods for motion and/or shape estimation, which recover the unknown parameters directly from *measurable image quantities* at *each pixel* in the image. This is contrast to the "feature-based methods", which first extract a sparse set of distinct features from each image separately, and then recover and analyze their correspondences in order to determine the motion and shape. Feature-based methods minimize an

Feature Based Methods for Structure and Motion Estimation

P. H. S. Torr[1] and A. Zisserman[2]

[1] Microsoft Research Ltd, 1 Guildhall St
Cambridge CB2 3NH, UK
philtorr@microsoft.com
[2] Department of Engineering Science, University of Oxford
Oxford, OX1 3PJ, UK
az@robots.ox.ac.uk

All About Direct Methods

M. Irani[1] and P. Anandan[2]

[1] Dept. of Computer Science and Applied Mathematics,
The Weizmann Inst. of Science, Rehovot, Israel.
irani@wisdom.weizmann.ac.il
[2] Microsoft Research, One Microsoft Way,
Redmond, WA 98052, USA.
anandan@microsoft.com

Example: optimizing over motion models

$$E(u,v) = \sum [I(x+u,y+v) - T(x,y)]^2$$

template
(model)

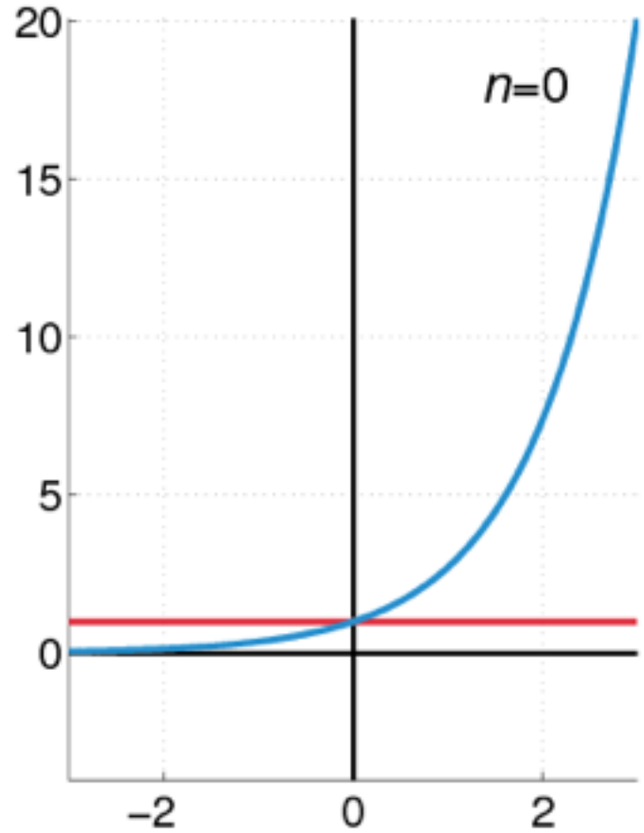
current frame

u,v = hypothesized location of
template in current frame

Nonlinear optimization

Apply standard linear optimization tricks after using Taylor series approximation

$$f(x+u) = f(x) + \frac{\partial f(x)}{\partial x}u + \frac{1}{2}\frac{\partial f(x)}{\partial xx}u^2 + \text{Higher Order Terms}$$



For multivariate functions, we make use of first-order gradient vector and second-order Hessian matrix

Nonlinear optimization

$$E(u) = \sum_x [I(x+u) - T(x)]^2$$

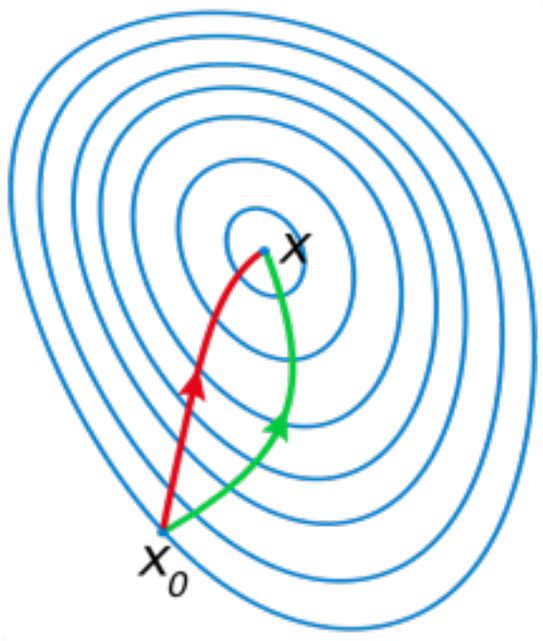
1. First order method (gradient descent)

$$u := u - \alpha g$$

2. Second order method (Newton's method)

https://en.wikipedia.org/wiki/Newton%27s_method_in_optimization

$$u := u - H^{-1} g$$



Nonlinear optimization

$$E(u) = \sum_x [I(x+u) - T(x)]^2$$

1. First order method (gradient descent)

$$u := u - \alpha g$$

2. Second order method (Newton's method)

https://en.wikipedia.org/wiki/Newton's_method_in_optimization

$$u := u - H^{-1} g$$

1.5 Gauss-Newton approximation (for nonlinear least-squares)

https://en.wikipedia.org/wiki/Gauss–Newton_algorithm

i. Perform first-order Taylor series expansion of terms inside squared error Δu

ii. Solve quadratic error for Δu

(a) Take derivative of error wrt Δu and set equal to 0

(b) It turns out that we'll compute an approximate hessian: $H \approx gg^T$

Outline

- 2D transformations
- Direct methods
- Lucas Kanade

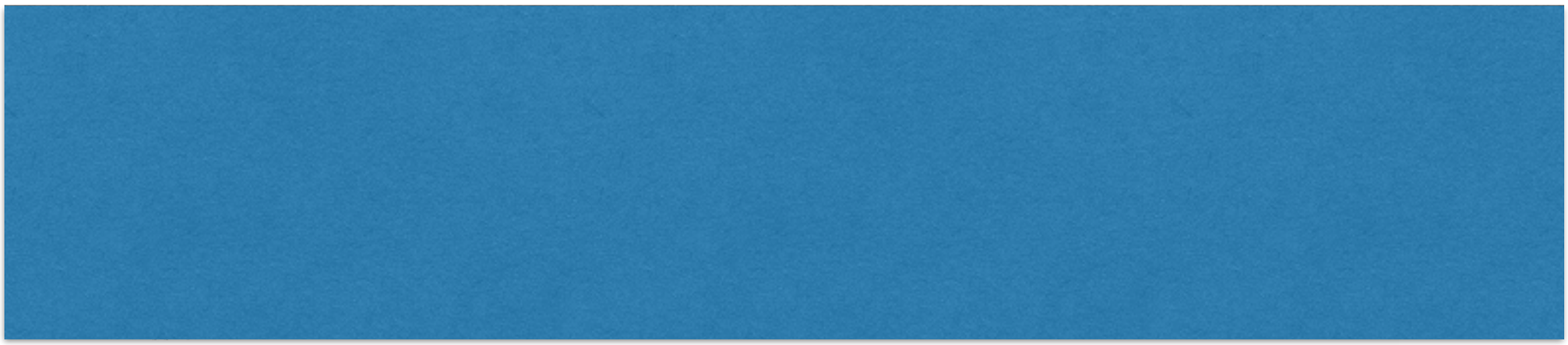
Lucas Kanade alignment

$$E(u,v) = \sum [I(x+u,y+v) - T(x,y)]^2$$

$$\approx \sum [I(x,y) + uI_x(x,y) + vI_y(x,y) - T(x,y)]^2 \quad \text{First order approx}$$

$$= \sum [uI_x(x,y) + vI_y(x,y) + D(x,y)]^2$$

Take partial derivs and set to zero



Form matrix equation

Lucas Kanade alignment

$$E(u,v) = \sum [I(x+u,y+v) - T(x,y)]^2$$

$$\sum \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = -\sum \begin{bmatrix} I_x D \\ I_y D \end{bmatrix}$$

Analogy with interest-point detection

Are corners easier to align?

HW3: Tracking by iterative template alignment

Start with template from first frame and repeat:

1. Align template to new frame with Lucas Kanade
2. Update template with new frame

Tracking by iterative template alignment

However, we can easily generalize Lucas-Kanade approach to other 2D parametric motion models (like affine or projective) by introducing a "warp" function W.

$$E(u,v) = \sum [I(x+u,y+v) - T(x,y)]^2 \xrightarrow{\text{generalize}} \sum [\, I(W([x,y];P)) - T([x,y])]^2$$

$$W([x,y];P]) = \begin{bmatrix} p_1 & p_2 & p_3 \\ p_4 & p_5 & p_6 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad \text{(for affine warps)}$$

Image Alignment

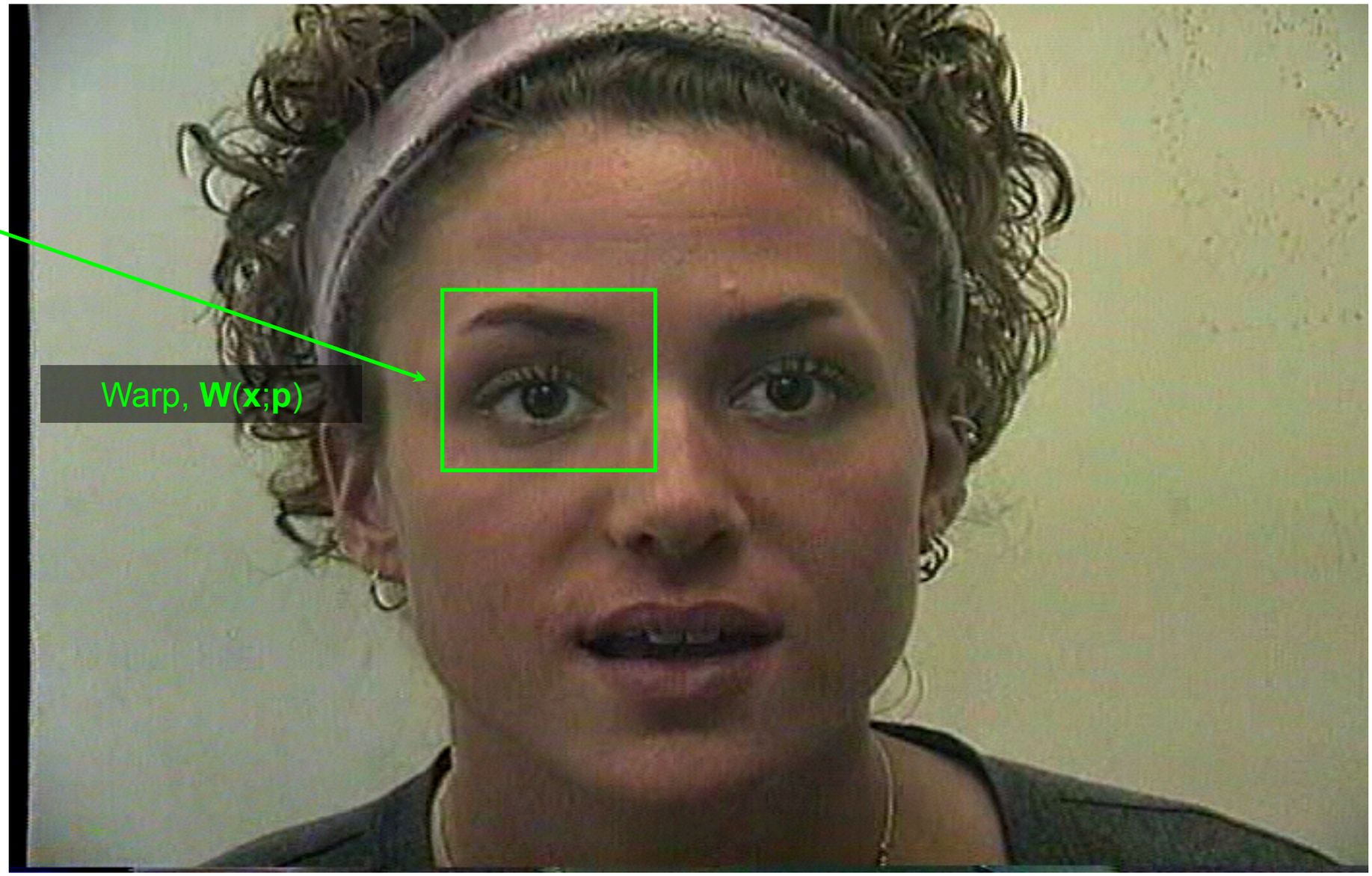
Template, $T(\mathbf{x})$

Image coordinates

$\mathbf{x} = (x, y)^T$

Warp parameters,

$\mathbf{p} = (p_1, p_2, \ldots, p_n)^T$



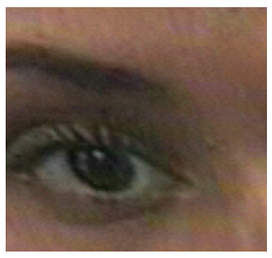
Image, $I(\mathbf{x})$

Want to: Minimize the Error

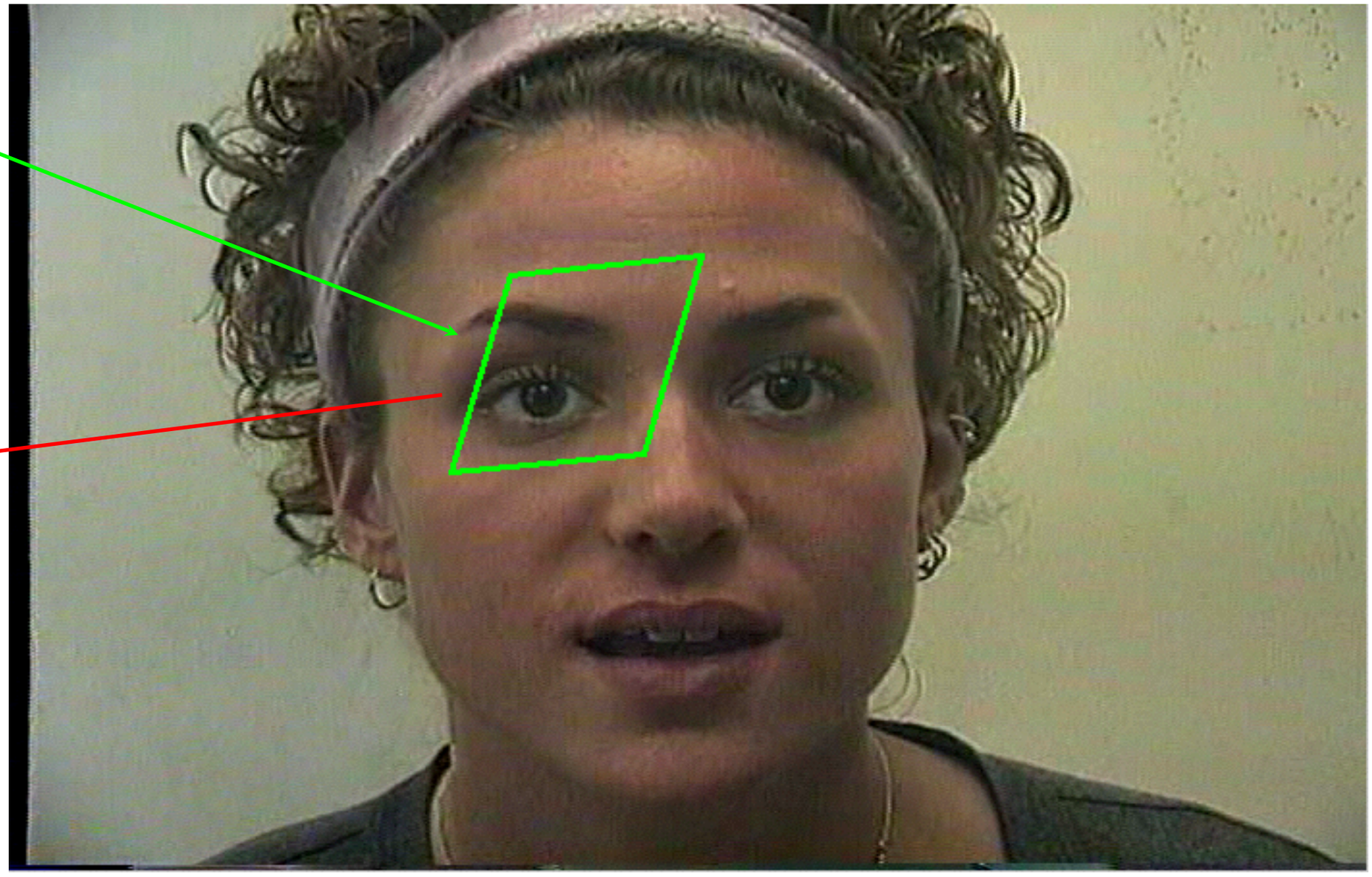
- Warp image to get $I(\mathbf{W}(\mathbf{x};\mathbf{p}))$ compute $[T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p}))]$

Template, $T(\mathbf{x})$

Warped, $I(\mathbf{W}(\mathbf{x};\mathbf{p}))$



$T(\mathbf{x})$ - $I(\mathbf{W}(\mathbf{x};\mathbf{p}))$



Nonlinear least squares

$$E(\mathbf{p}) = \sum_{\mathbf{x}} [I(\mathbf{W}(\mathbf{x};\mathbf{p})) - T(\mathbf{x})]^2$$

Apply taylor-series expansion to vector-valued function $\mathbf{W}$

$$\mathbf{W}(\mathbf{x};\mathbf{p}) = \begin{bmatrix} p_1 & p_2 & p_3 \\ p_4 & p_5 & p_6 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$W_x = p_1 x + p_2 y + p_3$$

$$W_y = p_4 x + p_5 y + p_6$$

In this case, the gradient vector generalizes to a Jacobian matrix

Derivation

Where we're headed:

$$I(\mathbf{W}(\mathbf{x};\mathbf{p}+\mathbf{\Delta p})) = I(\mathbf{W}(\mathbf{x};\mathbf{p})) + \Delta I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \Delta \mathbf{p}$$

We will derive this step-by-step. First, we need two background formula:

Chain rule

$z = f(x,y) \quad x = g(\tau) \quad y = h(\tau)$

$= f(g(\tau), h(\tau))$

Then

$$\frac{\partial z}{\partial \tau} = \frac{\partial z}{\partial x}\frac{\partial x}{\partial \tau} + \frac{\partial z}{\partial y}\frac{\partial y}{\partial \tau}$$

$$= \frac{\partial z}{\partial x}\frac{\partial g(\tau)}{\partial \tau} + \frac{\partial z}{\partial y}\frac{\partial h(\tau)}{\partial \tau}$$

Taylor series approximation

expand about an initial estimate $\tilde{\tau}$

$$z(\tilde{\tau}+\Delta\tau) = z(\tilde{\tau}) + \frac{\partial z}{\partial \tau}\Big|_{\tilde{\tau}} \Delta\tau + \text{higher order terms}$$

[from Robert Collins]

Single-parameter warp

(e.g., rotation by theta)

Shorthand notation: $I(\mathbf{x}, p) = I(\mathbf{W}(\mathbf{x}; p))$

$$I(x, y; \tilde{p} + \Delta p) \approx I(x, y; \tilde{p}) + \left.\frac{\partial I}{\partial p}\right|_{\tilde{p}} \Delta p \quad \text{Taylor approximation}$$

$$= I(x, y; \tilde{p}) + \underbrace{\left[\frac{\partial I}{\partial x}\frac{\partial W_x(p)}{\partial p} + \frac{\partial I}{\partial y}\frac{\partial W_y(p)}{\partial p}\right]_{\tilde{p}}}_{\text{chain rule}} \Delta p$$

$$I(\mathbf{x}; \tilde{p} + \Delta p) \approx I(\mathbf{x}; \tilde{p}) + \left[\frac{\partial I(\mathbf{x}, \tilde{p})}{\partial x} \quad \frac{\partial I(\mathbf{x}, \tilde{p})}{\partial y}\right] \left[\begin{array}{c}\frac{\partial W_x(\mathbf{x}, p)}{\partial p} \\ \frac{\partial W_y(\mathbf{x}, p)}{\partial p}\end{array}\right]_{\tilde{p}} \Delta p$$

current warped image

current warped image gradient

jacobian

parameter update

Multi-parameter warp

$$I(x,y;\tilde{p}_1,\ldots,\tilde{p}_n) + \begin{bmatrix} \frac{\partial I}{\partial x} & \frac{\partial I}{\partial y} \end{bmatrix} \begin{bmatrix} \partial w_x/\partial p_1 \\ \partial w_y/\partial p_1 \end{bmatrix}_{\tilde{p}_1} \Delta p_1 +$$

$$+ \begin{bmatrix} \frac{\partial I}{\partial x} & \frac{\partial I}{\partial y} \end{bmatrix} \begin{bmatrix} \partial w_x/\partial p_2 \\ \partial w_y/\partial p_2 \end{bmatrix}_{\tilde{p}_2} \Delta p_2 +$$

$$\vdots$$

$$I(\mathbf{x};\tilde{\mathbf{p}}+\mathbf{\Delta p}) \approx I(\mathbf{x};\tilde{\mathbf{p}}) + \left[\frac{\partial I(\mathbf{x},\tilde{\mathbf{p}})}{\partial x} \quad \frac{\partial I(\mathbf{x},\tilde{\mathbf{p}})}{\partial y}\right] \left[\begin{array}{c} \frac{\partial W_x(\mathbf{x},\mathbf{p})}{\partial \mathbf{p}} \\ \frac{\partial W_y(\mathbf{x},\mathbf{p})}{\partial \mathbf{p}} \end{array}\right]_{\tilde{\mathbf{p}}} \mathbf{\Delta p}$$

current warped image

current warped image gradient

jacobian matrix

parameter update vector

Example: jacobian of affine warp

affine warp function (6 parameters)

$$W([x,y];P) = \begin{pmatrix} 1+p_1 & p_3 & p_5 \\ p_2 & 1+p_4 & p_6 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

$$\frac{\partial W}{\partial P} = \frac{\partial \begin{bmatrix} x + xP_1 + yP_3 + P_5 \\ xP_2 + y + yP_4 + P_6 \end{bmatrix}}{\partial P}$$

$$= \begin{bmatrix} x & 0 & y & 0 & 1 & 0 \\ 0 & x & 0 & y & 0 & 1 \end{bmatrix}$$

Above affine parameterization is better conditioned for optimization because all-zero parameters default to the identity transformation

Back to the big-picture

$$\sum_{\mathbf{x}} [\, I(\mathbf{W}(\mathbf{x};\mathbf{p}+\Delta\mathbf{p})) - T(\mathbf{x})\,]^2$$

$$\approx \sum_{\mathbf{x}} \left[I(\mathbf{W}(\mathbf{x};\mathbf{p})) + \boldsymbol{\nabla} I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \Delta\mathbf{p} - T(\mathbf{x}) \right]^2$$

Gradient Descent Solution

Least squares problem: Minimize to solve for $\Delta\mathbf{p}$

$$\sum_{\mathbf{x}} \left[I(\mathbf{W}(\mathbf{x};\mathbf{p})) + \nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \Delta\mathbf{p} - T(\mathbf{x}) \right]^2$$

Solution,

$$\Delta\mathbf{p} = \sum_{\mathbf{x}} H^{-1} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]^{\mathrm{T}} [T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p}))]$$

Error Image

Gradient

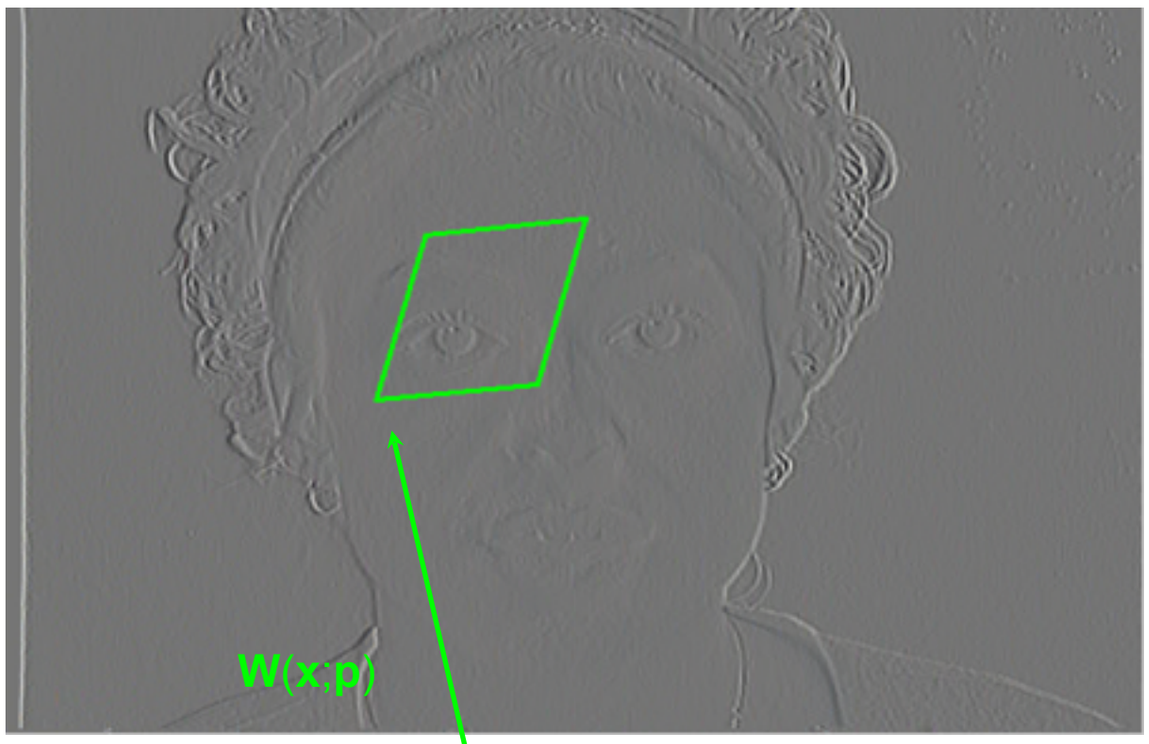
Hessian

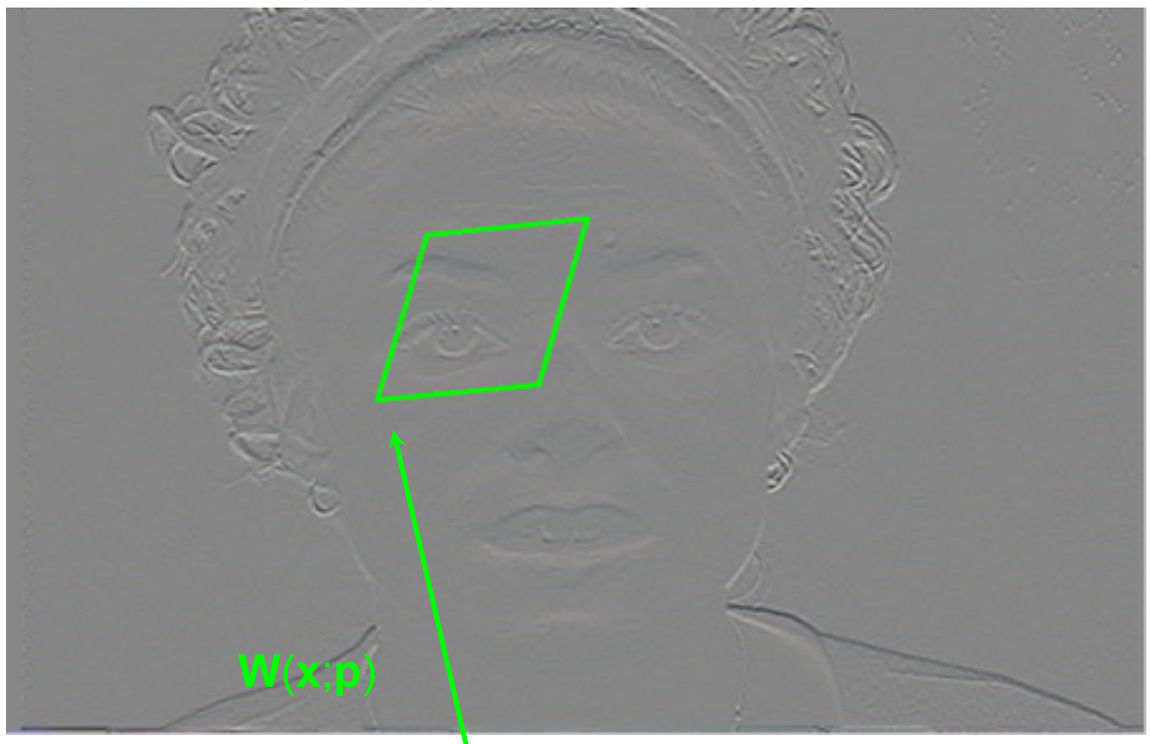
$$H = \sum_{\mathbf{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]^{\mathrm{T}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]$$

Jacobian

Gradient Images

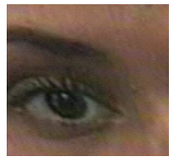
- Compute image gradient ∇I





∇I_x

$I(\mathbf{W}(\mathbf{x};\mathbf{p}))$



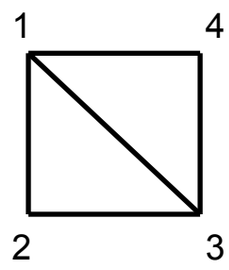
∇I_y

Jacobian

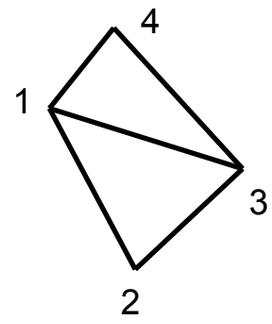
- Compute Jacobian $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$

Mesh parameterization

Template, $T(\mathbf{x})$



Warp, $\mathbf{W}(\mathbf{x};\mathbf{p})$



Image, $I(\mathbf{x})$

Image coordinates

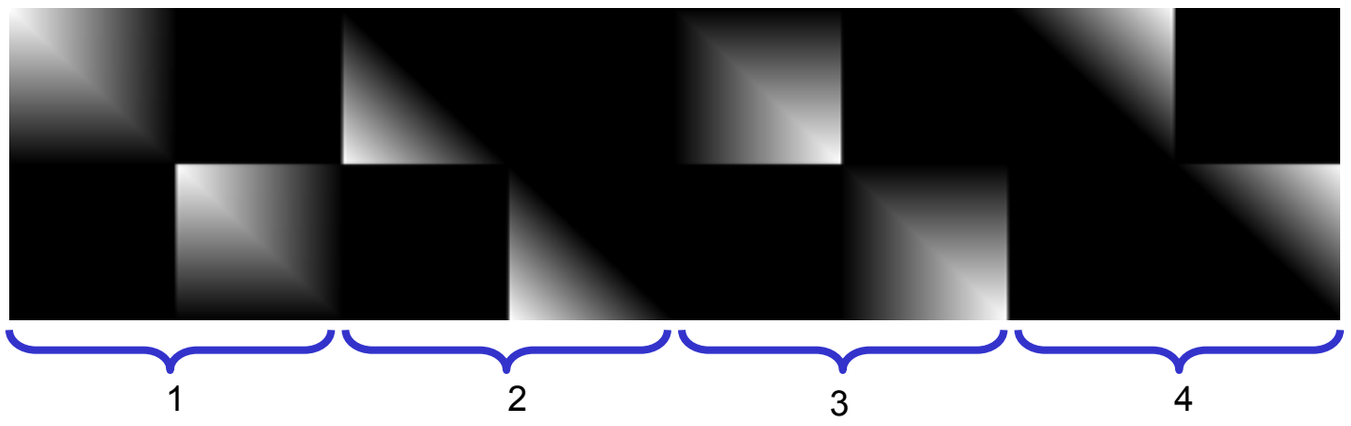
$\mathbf{x} = (x, y)^T$

Warp parameters,

$\mathbf{p} = (p_1, p_2, \ldots, p_n)^T = (dx_1, dy_1, \ldots, dx_n, dy_n)^T$

$$\mathbf{W} = (W_x(x,y), W_y(x,y))$$

$$\frac{\partial \mathbf{W}}{\partial \mathbf{p}} = \begin{pmatrix} \frac{\partial W_x}{\partial p_1} & \frac{\partial W_x}{\partial p_2} & \cdots & \frac{\partial W_x}{\partial p_n} \\ \frac{\partial W_y}{\partial p_1} & \frac{\partial W_y}{\partial p_2} & \cdots & \frac{\partial W_y}{\partial p_n} \end{pmatrix} =$$

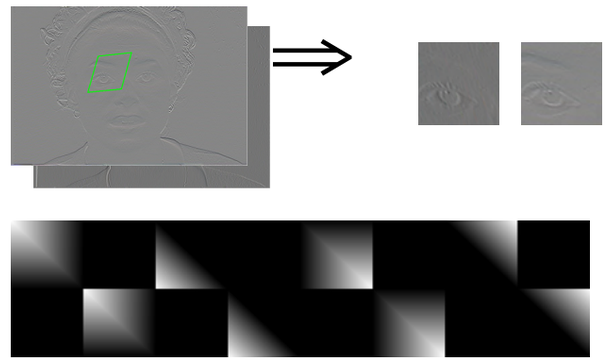


Lucas-Kanade Algorithm

1. Warp I with $\mathbf{W}(\mathbf{x};\mathbf{p}) \Rightarrow I(\mathbf{W}(\mathbf{x};\mathbf{p}))$

1. Compute error image $T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p}))$

1. Warp gradient of I to compute ∇I



1. Evaluate Jacobian $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$

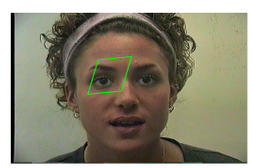
1. **Compute Hessian**

$$H = \sum_{\mathbf{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right]^{\mathrm{T}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right]$$

1. Compute $\Delta \mathbf{p}$

$$\Delta \mathbf{p} = \sum_{\mathbf{x}} H^{-1} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right]^{\mathrm{T}} [T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p}))]$$

1. Update parameters $\mathbf{p} \leftarrow \mathbf{p} + \Delta \mathbf{p}$



Fast Gradient Descent?

$$\Delta \mathbf{p} = \sum_{\mathbf{x}} H^{-1} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]^{\mathrm{T}} [T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x}; \mathbf{p}))]$$

$$H = \sum_{\mathbf{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]^{\mathrm{T}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]$$

- To reduce Hessian computation:
  1. Make Jacobian simple (*or constant*)
  2. Avoid computing gradients on *I*

Fantastic reference

Lucas-Kanade 20 Years On: A Unifying Framework

SIMON BAKER AND IAIN MATTHEWS
The Robotics Institute, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, USA
simonb@cs.cmu.edu
iainm@cs.cmu.edu

IJCV 2004

Overview

Additive warp: $\sum_{\mathbf{x}} [I(\mathbf{W}(\mathbf{x}; \mathbf{p} + \Delta\mathbf{p})) - T(\mathbf{x})]^2$ $\mathbf{p} \leftarrow \mathbf{p} + \Delta\mathbf{p}$.

Compositional warps: $\sum_{\mathbf{x}} [I(\mathbf{W}(\mathbf{W}(\mathbf{x}; \Delta\mathbf{p}); \mathbf{p})) - T(\mathbf{x})]^2$ $\mathbf{W}(\mathbf{x}; \mathbf{p}) \leftarrow \mathbf{W}(\mathbf{x}; \mathbf{p}) \circ \mathbf{W}(\mathbf{x}; \Delta\mathbf{p})$,

$$\mathbf{W}(\mathbf{x}; \mathbf{p}) = \begin{pmatrix} (1+p_1)\cdot x & + & p_3 \cdot y & + & p_5 \\ p_2 \cdot x & + & (1+p_4)\cdot y & + & p_6 \end{pmatrix}$$

$$\mathbf{W}(\mathbf{x}; \mathbf{p}) \circ \mathbf{W}(\mathbf{x}; \Delta\mathbf{p}) = \begin{pmatrix} (1+p_1)\cdot((1+\Delta p_1)\cdot x + \Delta p_3 \cdot y + \Delta p_5) \\ \quad + p_3 \cdot (\Delta p_2 \cdot x + (1+\Delta p_4)\cdot y + \Delta p_6) \\ \quad + p_5 \\ p_2 \cdot ((1+\Delta p_1)\cdot x + \Delta p_3 \cdot y + \Delta p_5) \\ \quad + (1+p_4)\cdot(\Delta p_2 \cdot x + (1+\Delta p_4)\cdot y \\ \quad + \Delta p_6) + p_6 \end{pmatrix}$$

Work out Taylor expansion; it turns out Jacobian is evaluated at $\mathbf{p} = 0$, which means it can be precomputed

Overview

Additive warp: $\sum_{\mathbf{x}} [I(\mathbf{W}(\mathbf{x};\mathbf{p}+\Delta\mathbf{p})) - T(\mathbf{x})]^2$ $\quad\quad$ $\mathbf{p} \leftarrow \mathbf{p} + \Delta\mathbf{p}$.

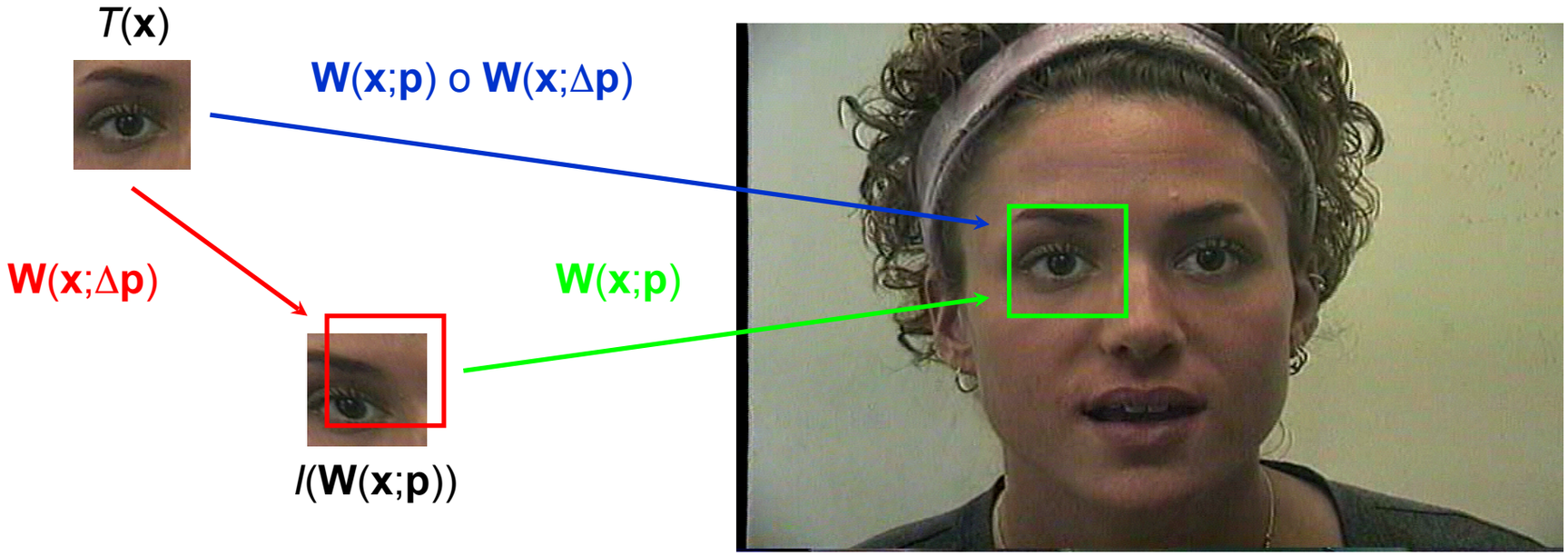
Compositional warp: $\sum_{\mathbf{x}} [I(\mathbf{W}(\mathbf{W}(\mathbf{x};\Delta\mathbf{p});\mathbf{p})) - T(\mathbf{x})]^2$ $\quad\quad$ $\mathbf{W}(\mathbf{x};\mathbf{p}) \leftarrow \mathbf{W}(\mathbf{x};\mathbf{p}) \circ \mathbf{W}(\mathbf{x};\Delta\mathbf{p})$,

Inverse compositional warp: $\sum_{\mathbf{x}} [\, T(\mathbf{W}(\mathbf{x};\Delta\mathbf{p})) - I(\mathbf{W}(\mathbf{x};\mathbf{p}))\,]^2$ $\quad\quad$ $\mathbf{W}(\mathbf{x};\mathbf{p}) \leftarrow \mathbf{W}(\mathbf{x};\mathbf{p}) \circ \mathbf{W}(\mathbf{x};\Delta\mathbf{p})^{-1}$.

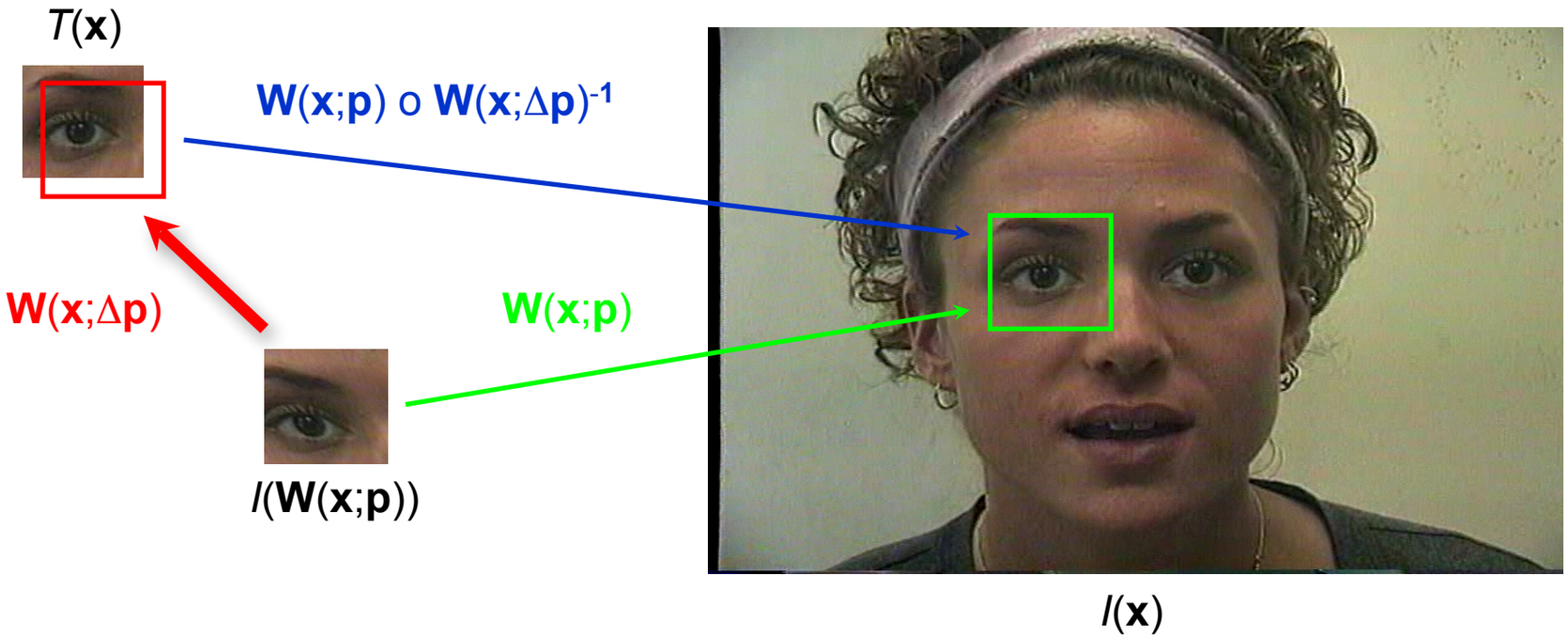
Work out Taylor expansion; both Jacobian and Hessian are not a function of current **p** and so can be precomputed

Forward and Inverse Compositional

- Forwards compositional



- Inverse compositional



Inverse Compositional

- Minimise,

$$\sum_{\mathbf{x}} [T(\mathbf{W}(\mathbf{x};\Delta\mathbf{p})) - I(\mathbf{W}(\mathbf{x};\mathbf{p}))]^2 \approx \sum_{\mathbf{x}} \left[T(\mathbf{W}(\mathbf{x};\mathbf{0})) + \nabla T \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \Delta\mathbf{p} - I(\mathbf{W}(\mathbf{x};\mathbf{p})) \right]^2$$

- Solution

$$H = \sum_{\mathbf{x}} \left[\nabla T \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]^{\mathrm{T}} \left[\nabla T \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]$$

$$\Delta\mathbf{p} = -\sum_{\mathbf{x}} H^{-1} \left[\nabla T \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]^{\mathrm{T}} [T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p}))]$$

- Update

$$\mathbf{W}(\mathbf{x};\mathbf{p}) \leftarrow \mathbf{W}(\mathbf{x};\mathbf{p}) \circ \mathbf{W}(\mathbf{x};\Delta\mathbf{p})^{-1}$$

Inverse Compositional

- Jacobian is **constant** - *evaluated at (**x**, **0**)*
- Gradient of template is **constant**
- Hessian is **constant**

$$H = \sum_{\mathbf{x}} \left[\nabla T \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]^{\mathrm{T}} \left[\nabla T \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]$$

$$\Delta \mathbf{p} = -\sum_{\mathbf{x}} H^{-1} \left[\nabla T \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]^{\mathrm{T}} [T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p}))]$$

- Can pre-compute everything but error image!

Peicewise affine-tracking