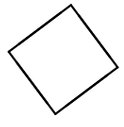
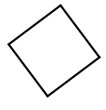
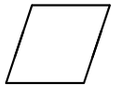
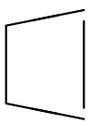
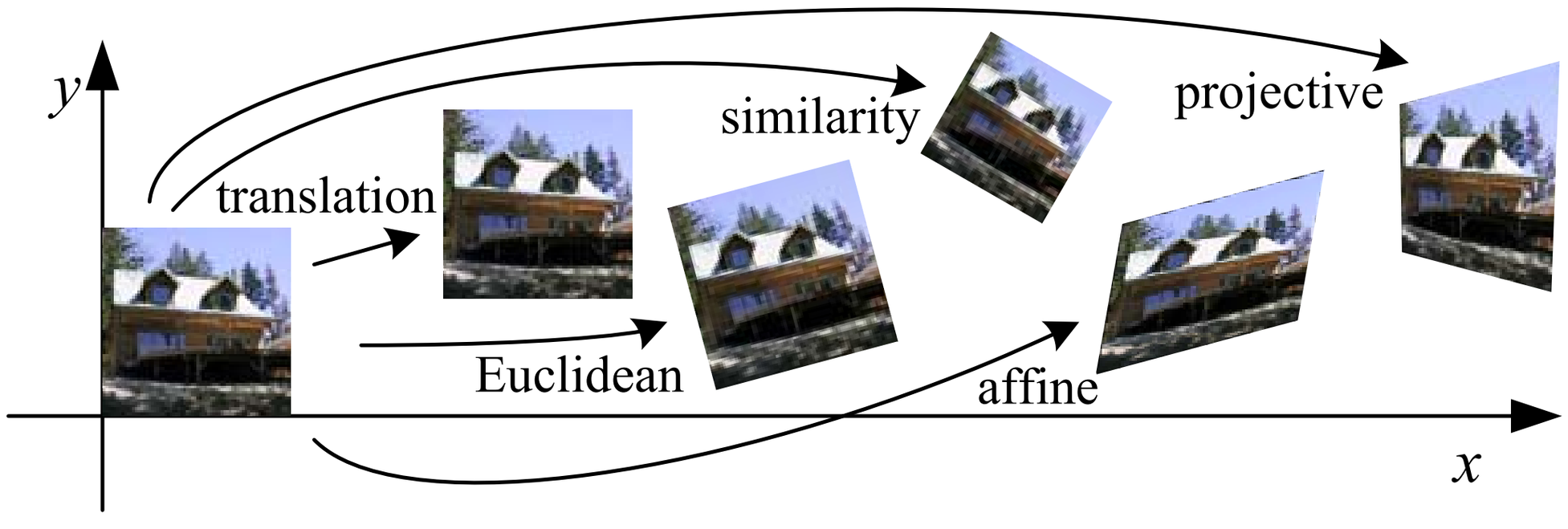
Image Alignment

(material for HW2)

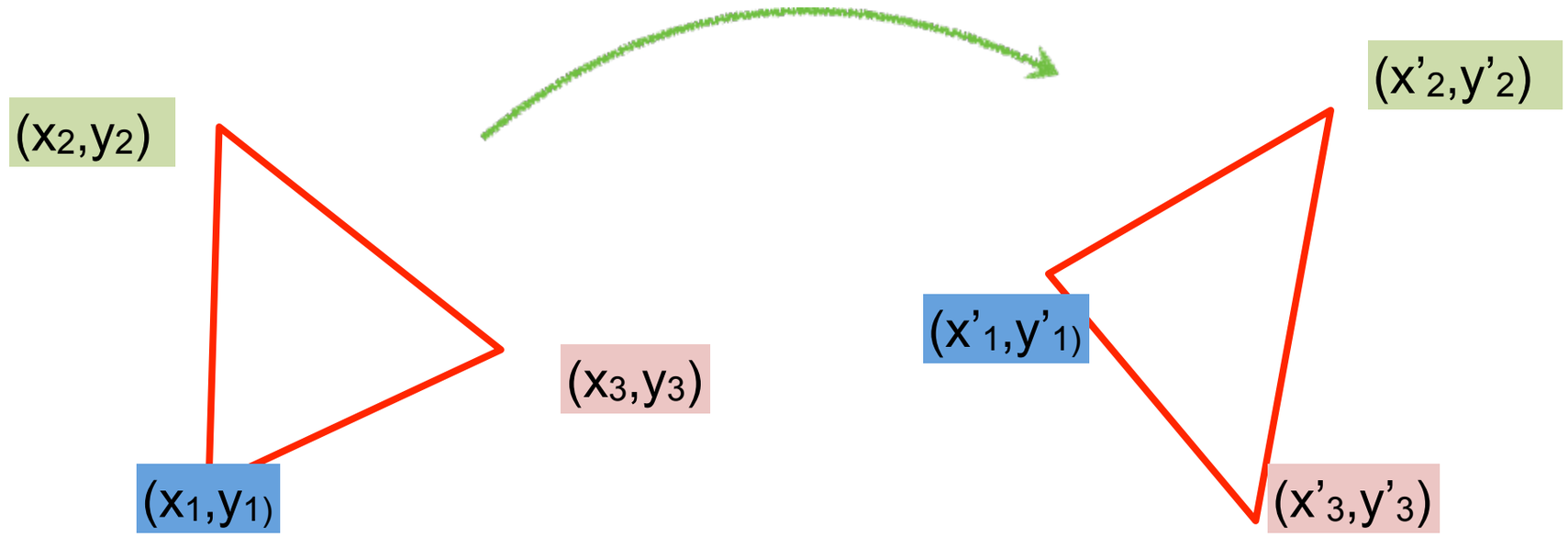
Recall:
family of 2D transformations representable by a matrix

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	



Estimating transformations from point correspondences

Suppose we have two triangles:

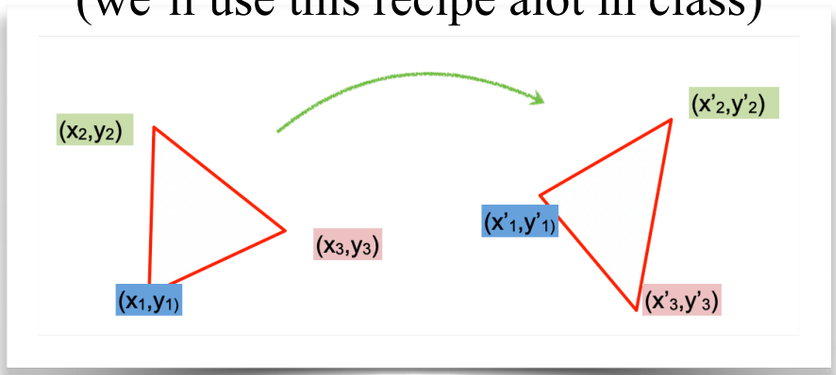


$$\begin{bmatrix} x'_1 \\ y'_1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

Given point correspondences, how do we estimate affine parameters (a,b,c,d,e,f)?

Estimating motion from correspondences

(we'll use this recipe alot in class)



Affine transformation:

$$ax + by + c = x'$$
$$sx + ey + f = y'$$

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

Write for all three points (1,2,3):

$$\begin{aligned} ax_1 + by_1 + c &= x'_1 \\ dx_1 + ey_1 + f &= y'_1 \\ ax_2 + by_2 + c &= x'_2 \\ dx_2 + ey_2 + f &= y'_2 \\ ax_3 + by_3 + c &= x'_3 \\ dx_3 + ey_3 + f &= y'_3 \end{aligned}$$

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1 & y_1 & 1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_2 & y_2 & 1 \\ x_3 & y_3 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_3 & y_3 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix} = \begin{bmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ x'_3 \\ y'_3 \end{bmatrix}$$

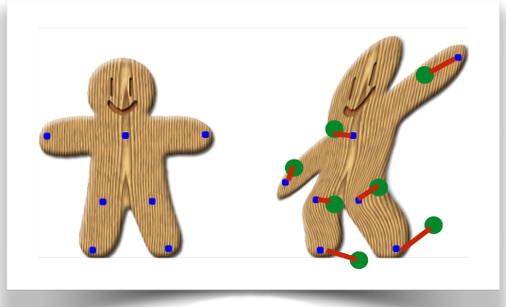
$$\mathbf{A} \quad \mathbf{x} = \mathbf{b}$$

$$\mathbf{x} = A^{-1}\mathbf{b}$$

```
x = numpy.linalg.solve(A,b)
```

When is A invertible? If 3 points are not *co-linear*

What if we have more than 3 points?

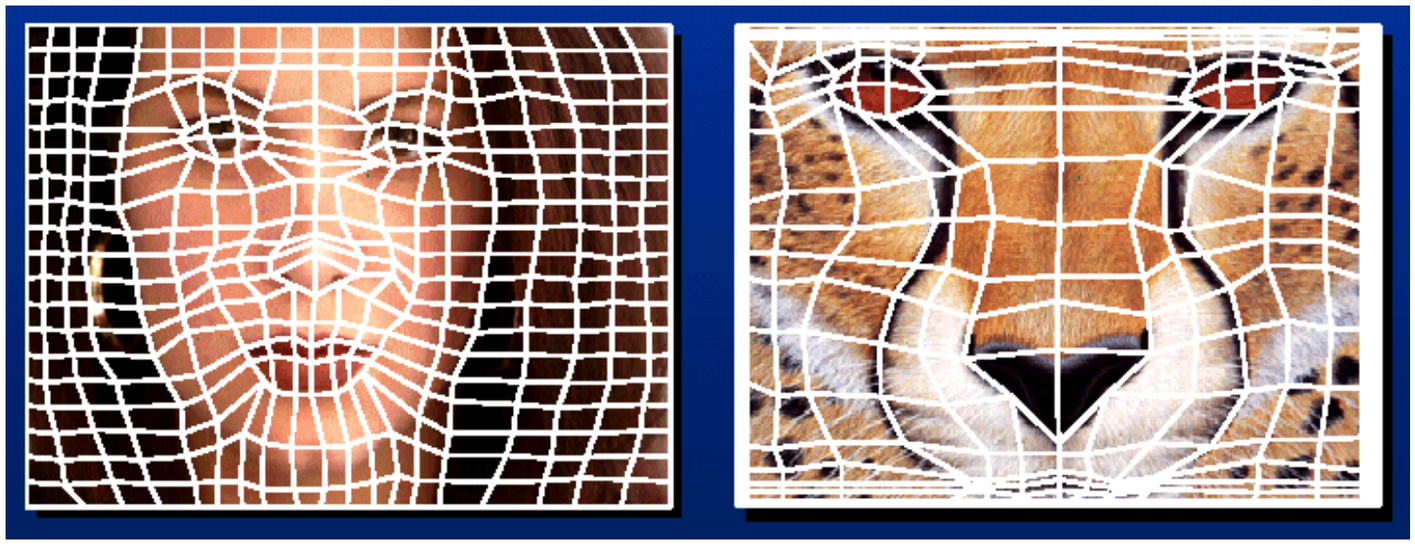


Estimate parameters that minimize (squared) euclidean error with least squares

$$\mathbf{x} = \underset{\mathbf{x}}{\operatorname{argmin}} \|\hat{\mathbf{b}} - \mathbf{b}\|^2 \text{ where } \hat{\mathbf{b}} = A\mathbf{x}$$

```
x = numpy.linalg.lstsq(A,b)
```

Image warping via *peicewise affine warps*



Piecewise affine warps (cut each quadrilateral into 2 triangles)

Compute intermediate vertex position by linear interpolation $\alpha \begin{bmatrix} x \\ y \end{bmatrix} + (1-\alpha) \begin{bmatrix} x' \\ y' \end{bmatrix}$

Compute intermediate color by linear interpolation $\alpha \begin{bmatrix} R \\ G \\ B \end{bmatrix} + (1-\alpha) \begin{bmatrix} R' \\ G' \\ B' \end{bmatrix}$

Example application: shape modeling

D'Arcy Thompson
"On Growth and Form" 1915

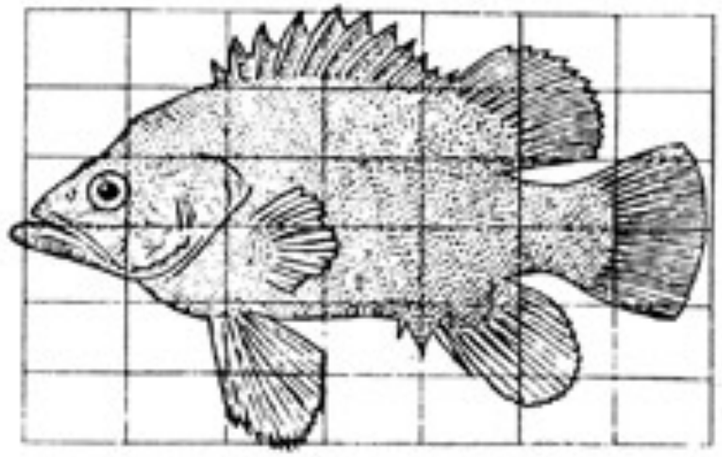


Fig. 150. *Polyprion.*

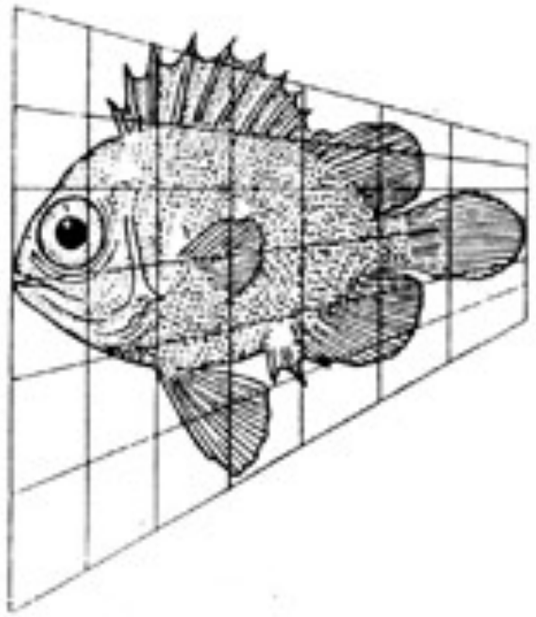


Fig. 151. *Pseudopriacanthus altus.*

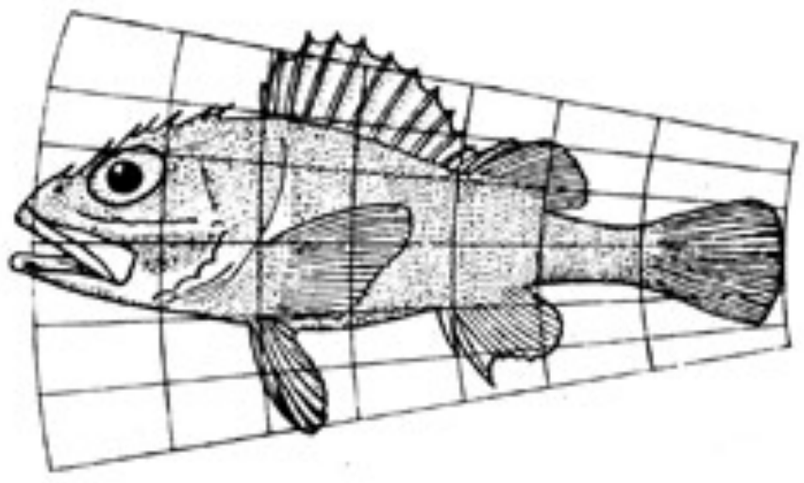


Fig. 152. *Scorpaena* sp.

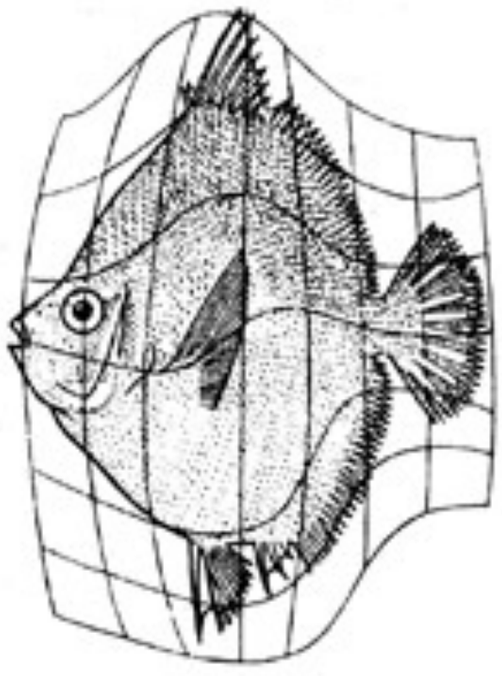
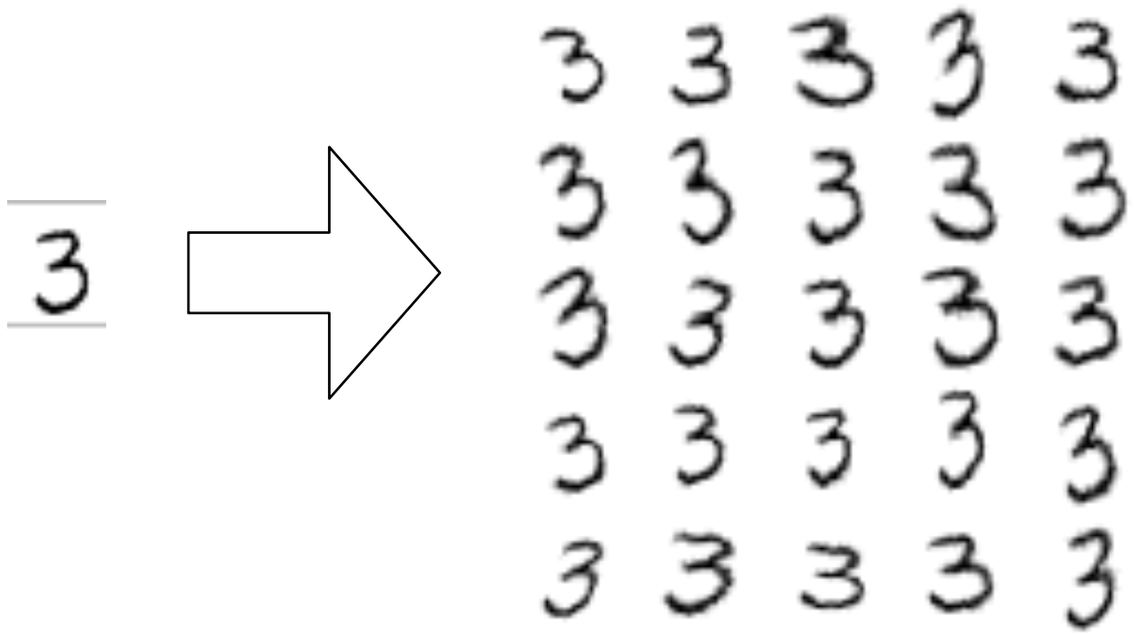


Fig. 153. *Antigonia capros.*

Application: data augmentation

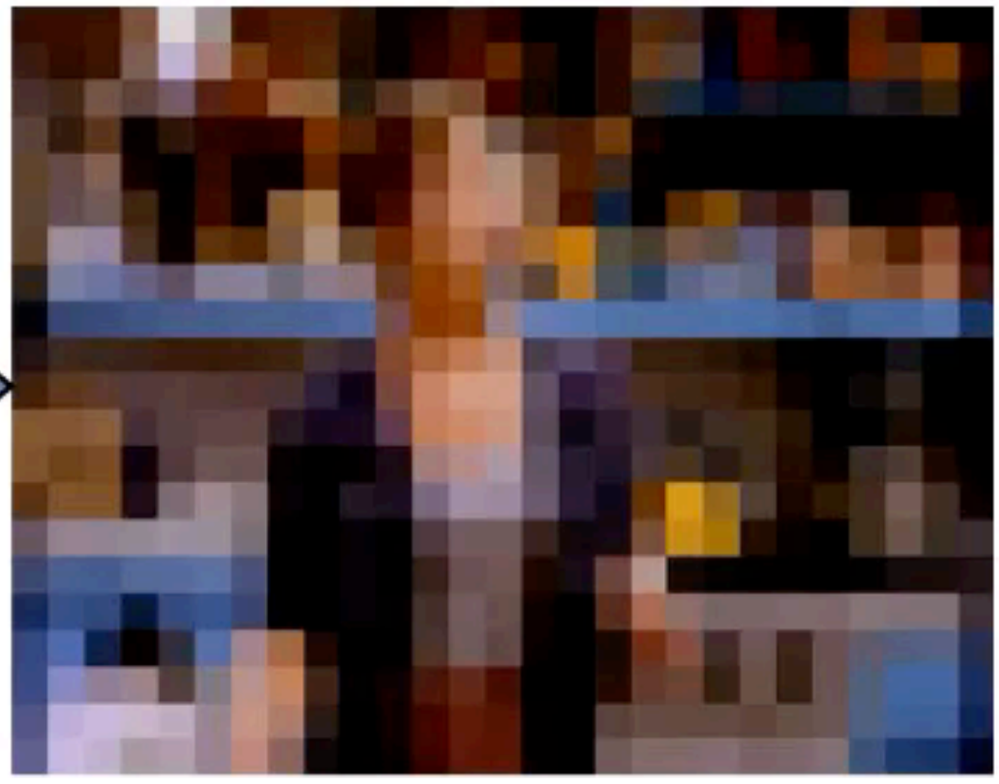


Hypothesis: we'll see even more of this in the future!

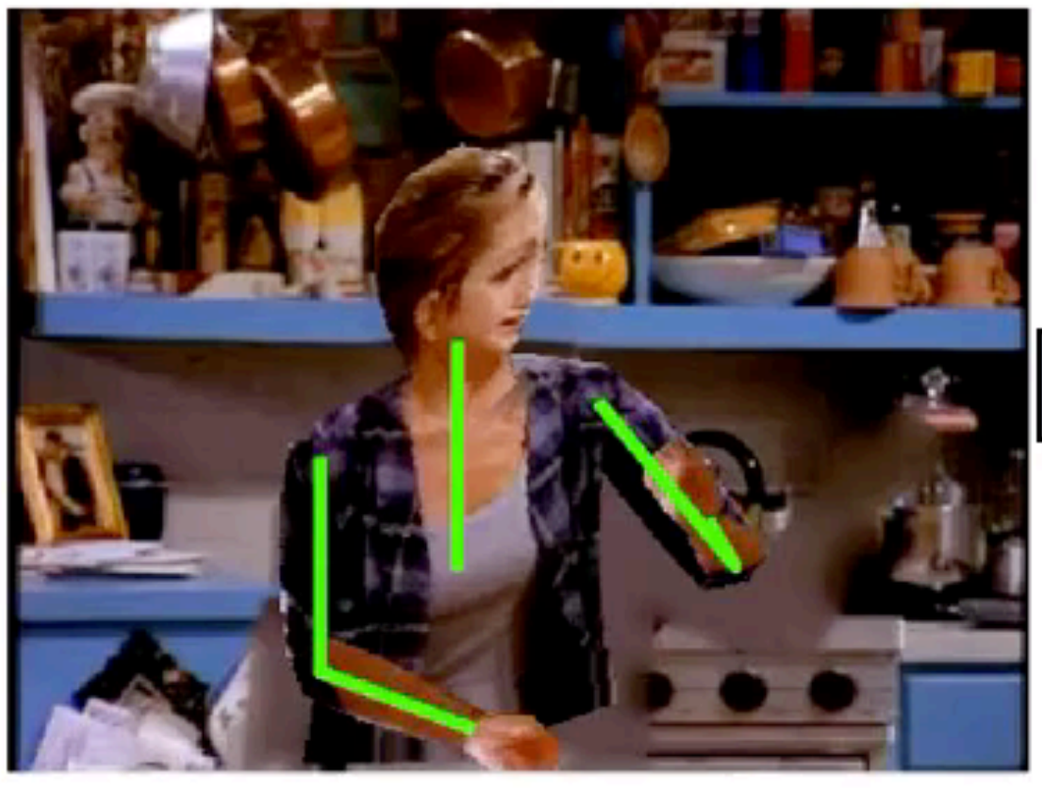
original

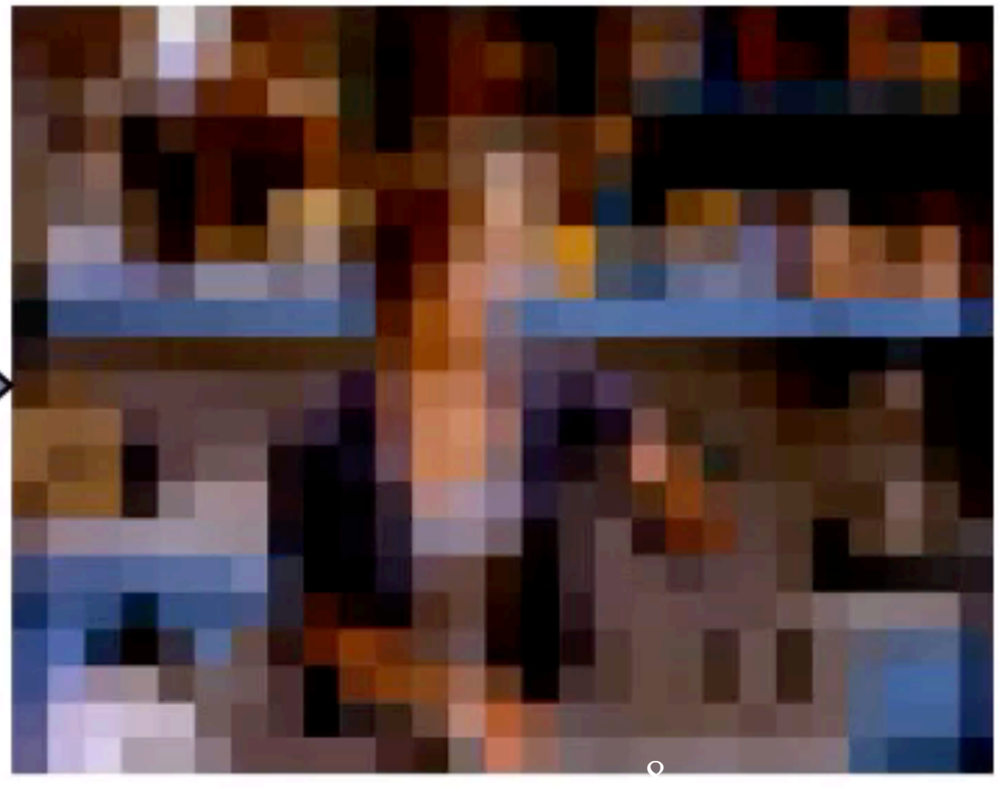
low resolution

test video

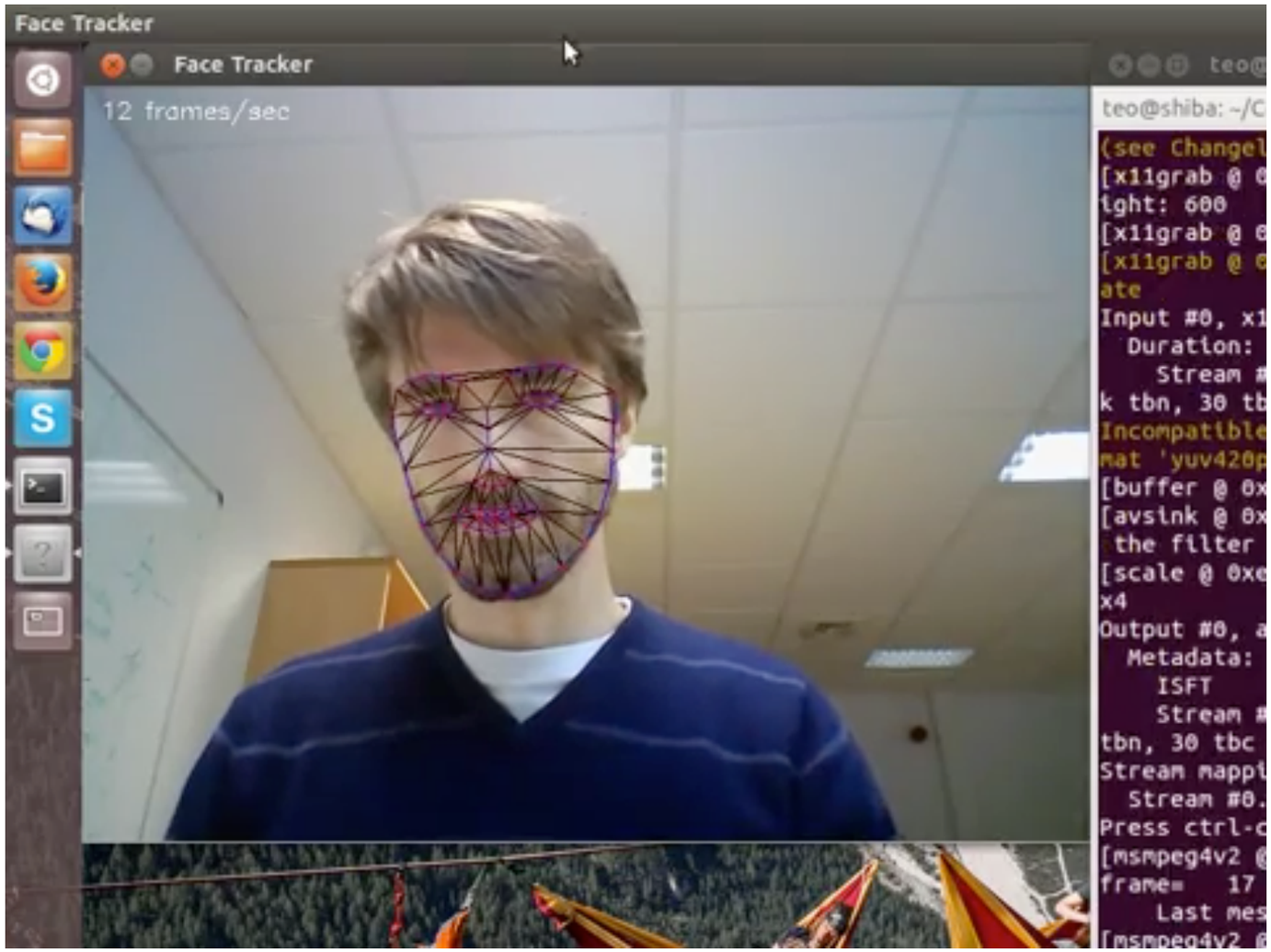


best match

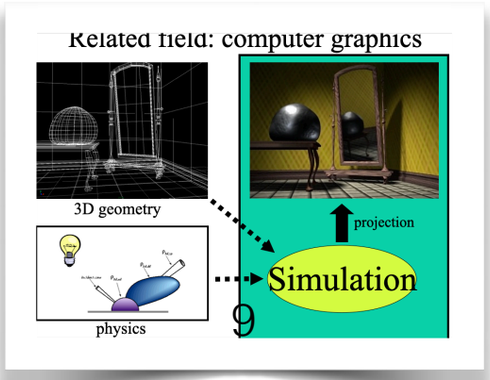


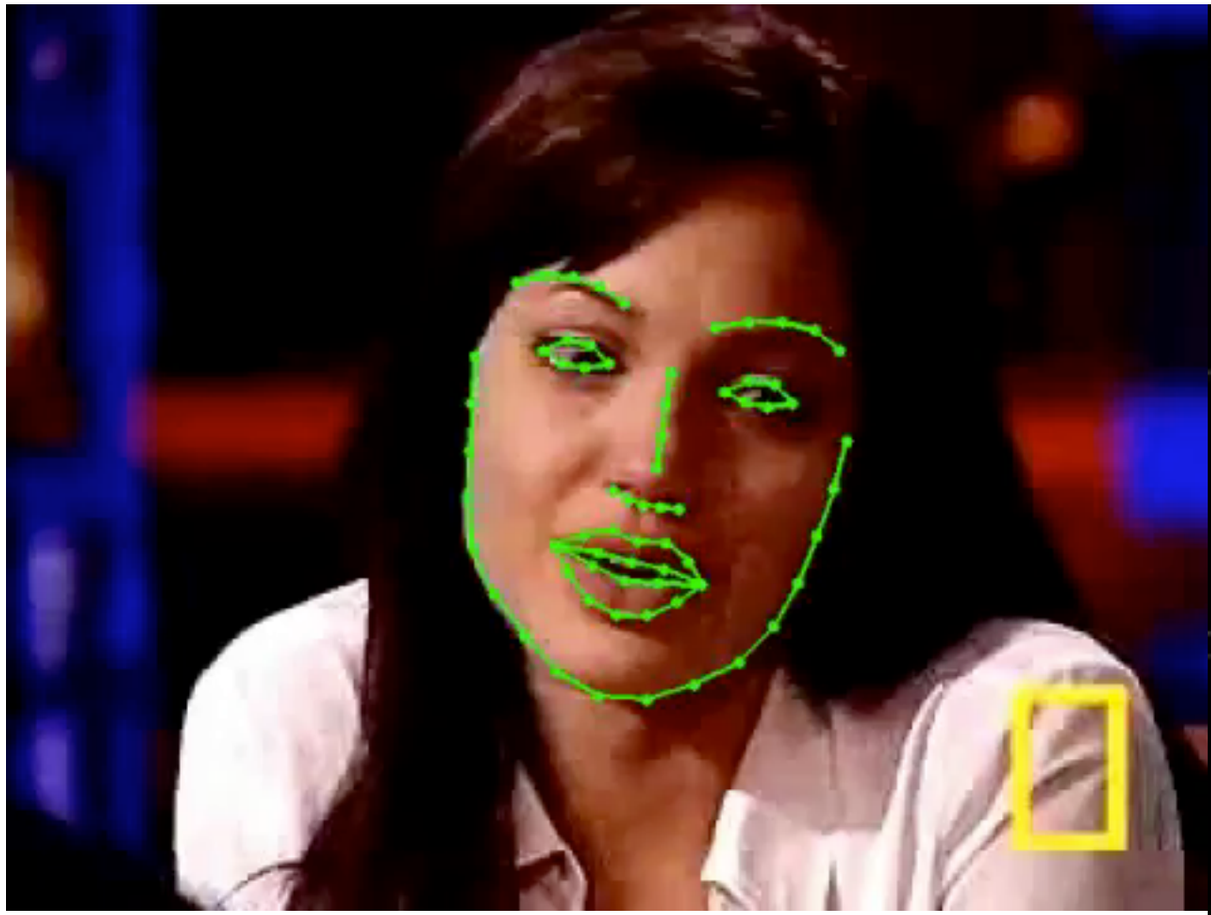


Application: Tracking (HW2)!



Really instance of vision as inverse graphics or “analysis by synthesis”; interpret the world by *optimizing* for best fit of model



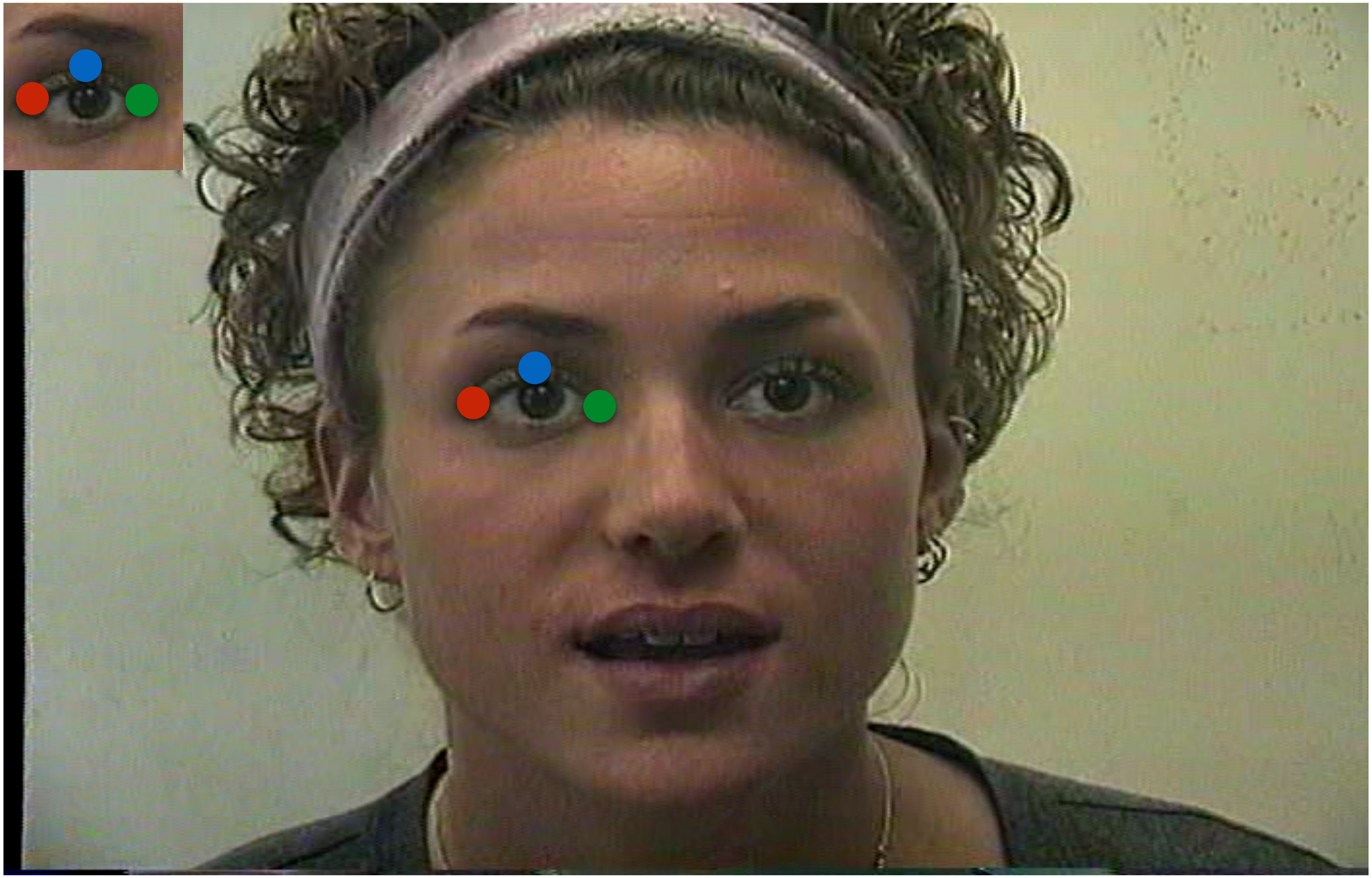


IntraFace


31 fps

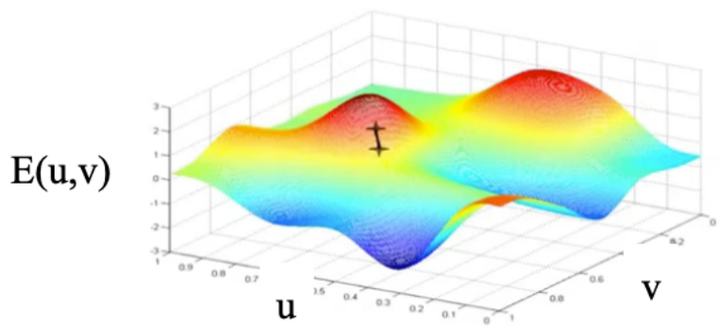
How can I find  in the image?

Idea #1: Least squares fit to corresponding points



Works great, but need to find corresponding points :(
(will explore automatic methods in HW3!)

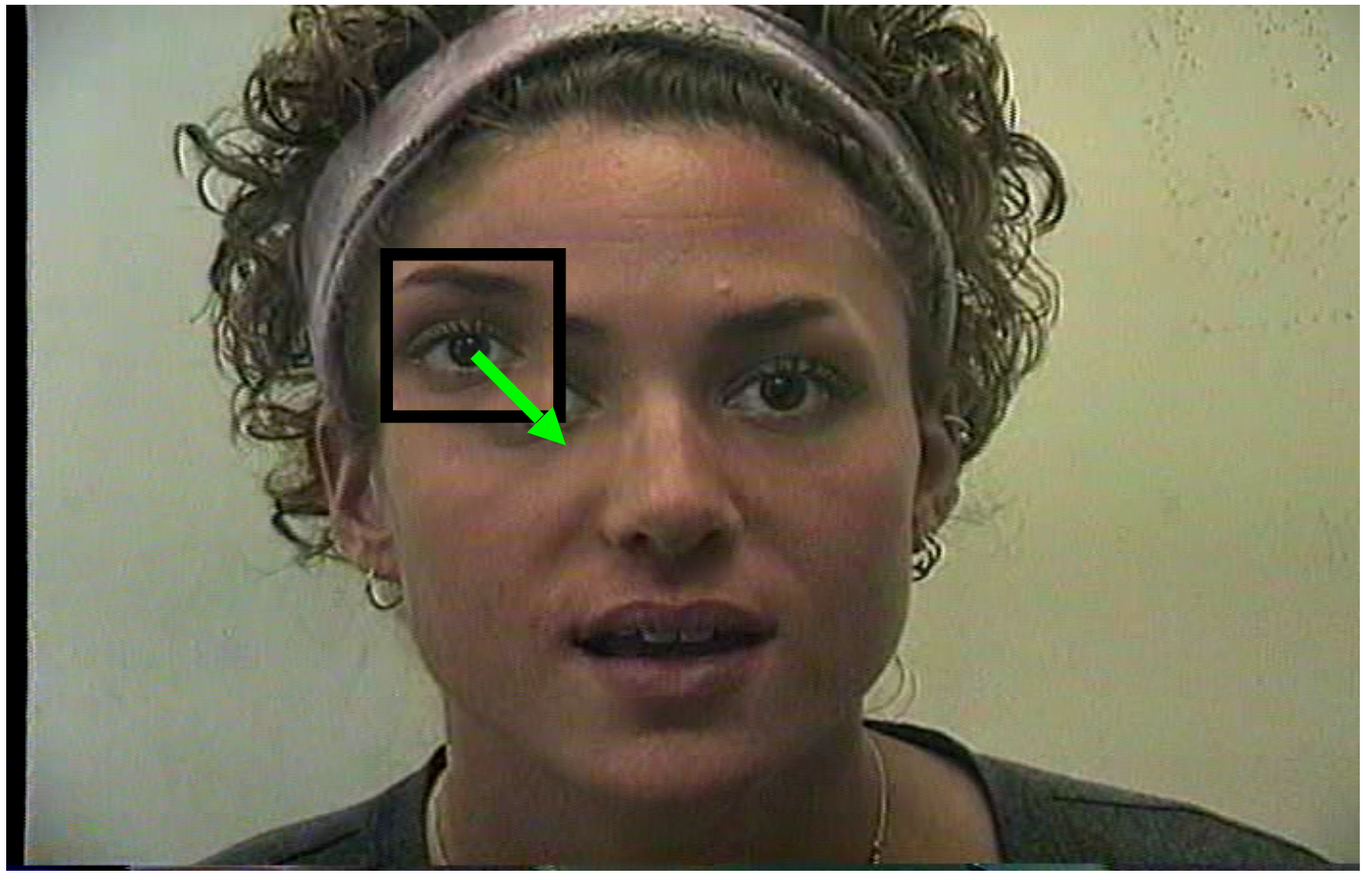
Idea #2: Template Matching by Filtering



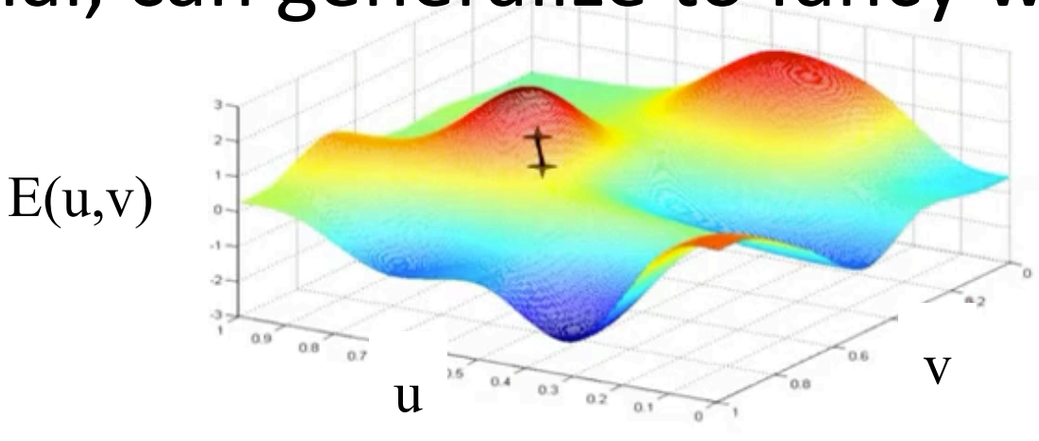
Global solution but slow

(hard to generalize beyond translation to affine warps)

Idea #3: Local search with gradient descent

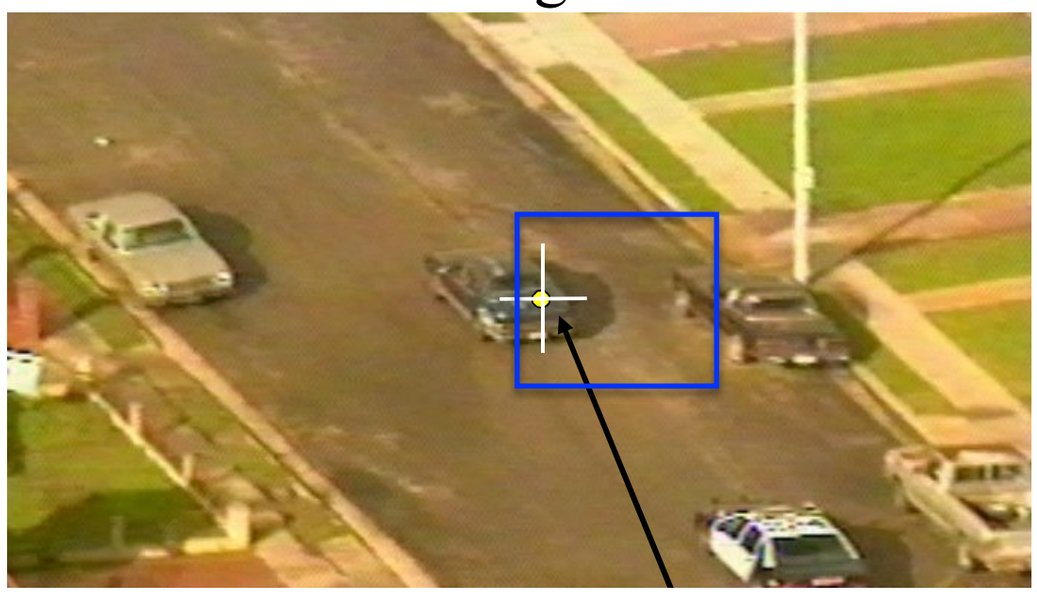


Locally optimal, can generalize to fancy warps, but need a good init



Where we are headed (HW2)

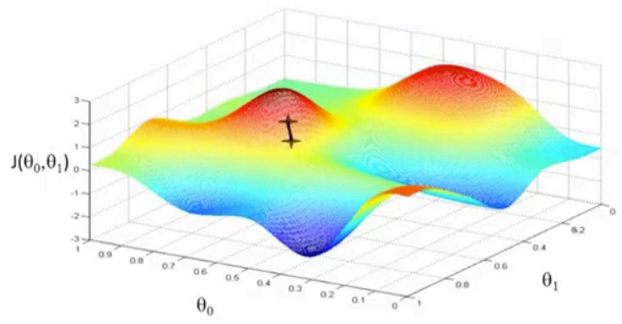
image



template

Tracking via template matching:
initialize bounding box in frame 1 and warp to match it to frame 2 (and repeat)

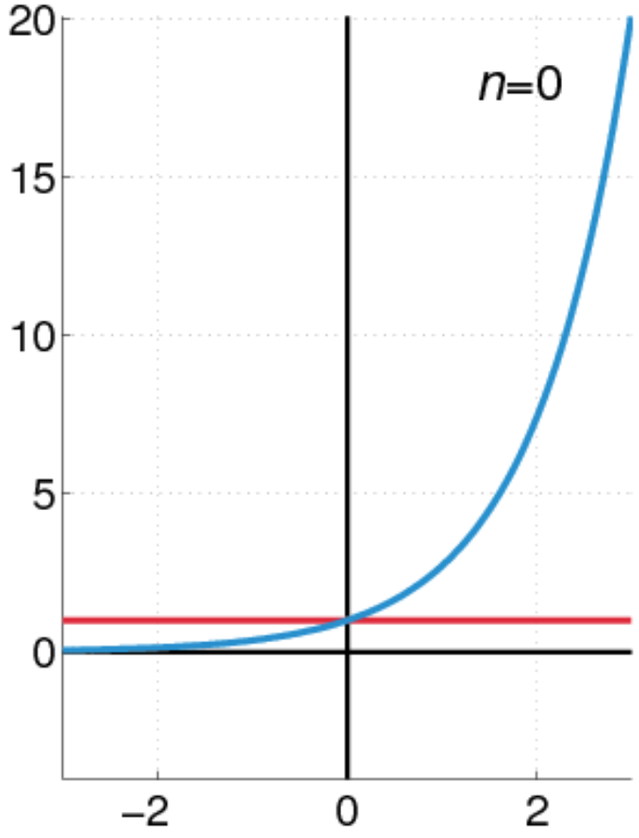
Turns out, we can do much better than gradient descent by opening up our “optimization toolbox”



Background: nonlinear optimization

Let's approximate a complicated function (or error surface) with a Taylor series

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



How do we generalize the above to a multivariate function f(u,v)?

symmetric matrix! Is it PSD?

only for convex functions

Generalize first derivative to $\begin{bmatrix} \frac{\partial f}{\partial u} \\ \frac{\partial f}{\partial v} \end{bmatrix}$ gradient and second derivative to $\begin{bmatrix} \frac{\partial^2 f}{\partial u \partial u} & \frac{\partial^2 f}{\partial u \partial v} \\ \frac{\partial^2 f}{\partial v \partial u} & \frac{\partial^2 f}{\partial v \partial v} \end{bmatrix}$ Hessian

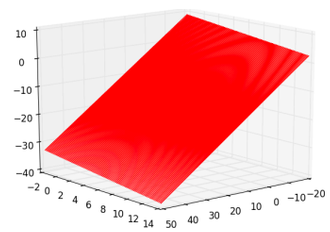
Background: nonlinear optimization

$$\min_{\mathbf{u}} f(\mathbf{u}) \quad \text{where } \mathbf{u} = (u,v)$$

1. First order method (gradient descent)

$$\begin{bmatrix} u \\ v \end{bmatrix} := \begin{bmatrix} u \\ v \end{bmatrix} - \alpha \begin{bmatrix} \frac{\partial f}{\partial u} \\ \frac{\partial f}{\partial v} \end{bmatrix}$$

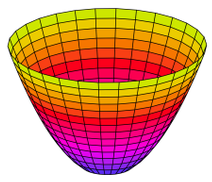
"fit a plane and take a step"



2. Second order method (Newton's method)

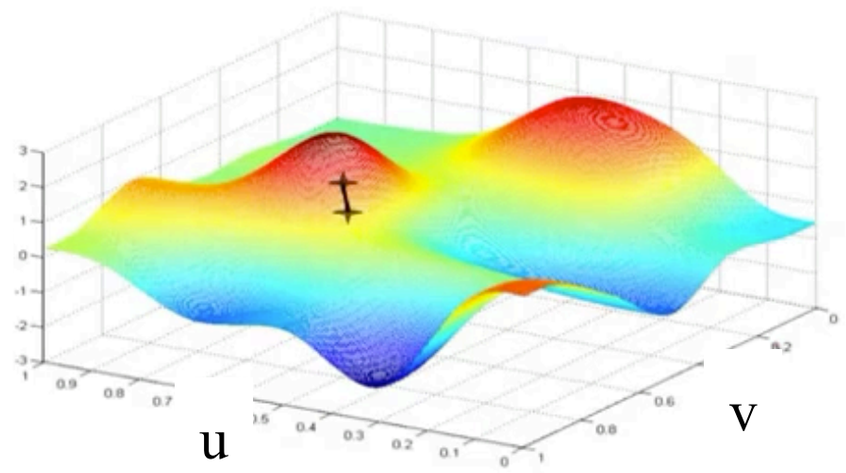
https://en.wikipedia.org/wiki/Newton's_method_in_optimization

"fit a paraboloid and jump to minimum"



$$\begin{bmatrix} u \\ v \end{bmatrix} := \begin{bmatrix} u \\ v \end{bmatrix} - \begin{bmatrix} \frac{\partial^2 f}{\partial u \partial u} & \frac{\partial^2 f}{\partial u \partial v} \\ \frac{\partial^2 f}{\partial v \partial u} & \frac{\partial^2 f}{\partial v \partial v} \end{bmatrix}^{-1} \begin{bmatrix} \frac{\partial f}{\partial u} \\ \frac{\partial f}{\partial v} \end{bmatrix}$$

E(u,v)



Some notation before we get into the math…

2D image transformation

$$\mathbf{W}(\mathbf{x};\mathbf{p}) = \begin{bmatrix} W_x(\mathbf{x};\mathbf{p}) \\ W_y(\mathbf{x};\mathbf{p}) \end{bmatrix}$$

2D image coordinate

$$\boldsymbol{x} = \begin{bmatrix} x \\ y \end{bmatrix}$$

Parameters of the transformation

$$\mathbf{p} = \begin{bmatrix} p_1 \\ p_2 \\ \vdots \end{bmatrix}$$

Warped image

$$I\Big(\mathbf{W}(\mathbf{x};\mathbf{p})\Big) = I\Big(W_x(\mathbf{x};\mathbf{p}), W_y(\mathbf{x};\mathbf{p})\Big)$$

Translation

$$\begin{bmatrix} W_x(\mathbf{x};\mathbf{p}) \\ W_y(\mathbf{x};\mathbf{p}) \end{bmatrix} = \begin{bmatrix} x + p_1 \\ y + p_2 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & p_1 \\ 0 & 1 & p_2 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$I\Big(\mathbf{W}(\mathbf{x};\mathbf{p})\Big) = I\Big(x + p_1, y + p_2\Big)$$

Affine

$$\begin{bmatrix} W_x(\mathbf{x};\mathbf{p}) \\ W_y(\mathbf{x};\mathbf{p}) \end{bmatrix} = \begin{bmatrix} p_1 x + p_2 y + p_3 \\ p_4 x + p_5 y + p_6 \end{bmatrix}$$

$$= \begin{bmatrix} p_1 & p_2 & p_3 \\ p_4 & p_5 & p_6 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$I\Big(\mathbf{W}(\mathbf{x};\mathbf{p})\Big) = I\Big(p_1 x + p_2 y + p_3, p_4 x + p_5 y + p_6\Big)$$

Find the warp parameters **p** such that minimize SSD

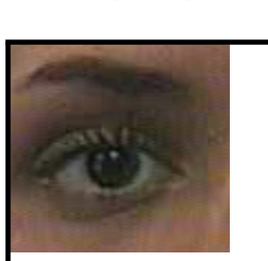
$$\min_{\boldsymbol{p}} \sum_{\boldsymbol{x}} [I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p})) - T(\boldsymbol{x})]^2$$

$$\min_{(p_1,p_2,\ldots)} \sum_{x,y} [I(p_1x + p_2y + p_3, p_4x + p_5y + p_6) - T(x,y)]^2$$

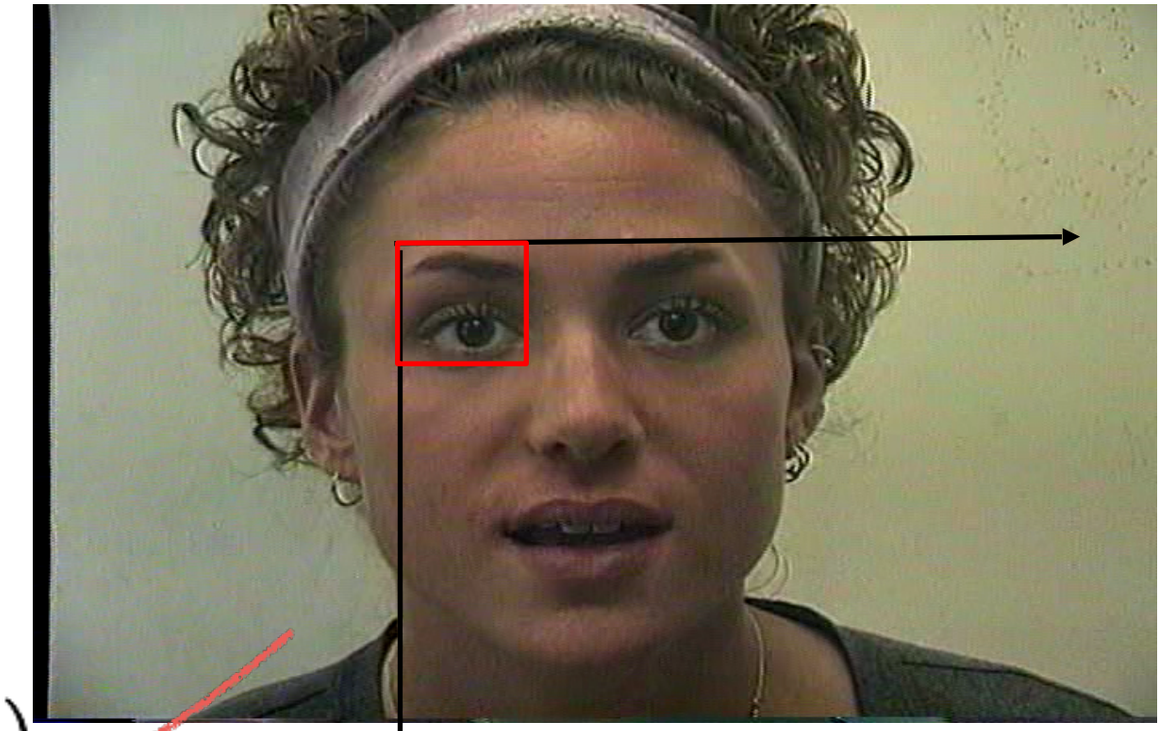
what pixels should be summed over?

(only those that are non-zero
in both template and warped image)

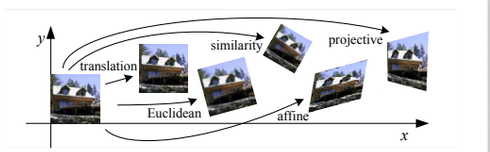
$T(\boldsymbol{x})$



$I(\boldsymbol{x})$



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



This is a non-linear least squares optimization!

What's the source of the nonlinearity? the image

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

$$\min_{(p_1,p_2,\ldots)} \sum_{x,y} [I(p_1 x + p_2 y + p_3, p_4 x + p_5 y + p_6) - T(x,y)]^2$$

Assuming a good initialization of **p**,
linearize objective and update incrementally

Lucas-Kanade alignment

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

$$\sum_{\boldsymbol{x}} \left[I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p})) - T(\boldsymbol{x}) \right]^2$$

$$\min_{(p_1,p_2,\ldots)} \sum_{x,y} [I(p_1 x + p_2 y + p_3, p_4 x + p_5 y + p_6) - T(x,y)]^2$$

Assuming an initial guess **p**, let's optimize over increment $\Delta\mathbf{p}$

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

This is **still** a nonlinear least squares optimization wrt $\Delta\mathbf{p}$

Key assumption: let's assume initial guess **p** is good and increment $\Delta\mathbf{p}$ will be small

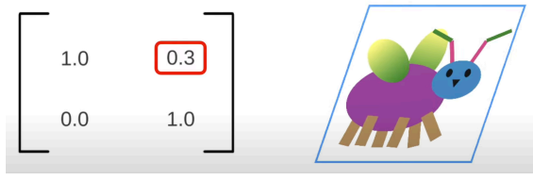
Then we can remove nonlinearity with a Taylor expansion

$$f(x,y) \approx f(a,b) + f_x(a,b)(x-a) - f_y(a,b)(y-b)$$

$$\begin{bmatrix} W_x(\mathbf{x};\mathbf{p}) \\ W_y(\mathbf{x};\mathbf{p}) \end{bmatrix} = \begin{bmatrix} p_1 x + p_2 y + p_3 \\ p_4 x + p_5 y + p_6 \end{bmatrix}$$

$$I\Big(\mathbf{W}(\mathbf{x};\mathbf{p})\Big) = I\Big(p_1 x + p_2 y + p_3, p_4 x + p_5 y + p_6\Big)$$

how does image change as I tweak the warp parameters?



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

chain rule

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

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

short-hand

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

Multivariable Taylor Series Expansion

$$f(x,y) \approx f(a,b) + f_x(a,b)(x-a) - f_y(a,b)(y-b)$$

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

What are the unknowns here?

Now, the squared error is linear in the unknowns (suggests we can use least squares to solve for them!)

The Jacobian $\dfrac{\partial \mathbf{W}}{\partial \boldsymbol{p}}$

(A matrix of partial derivatives)

Rate of change of the warp

$$\boldsymbol{x} = \begin{bmatrix} x \\ y \end{bmatrix}$$

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

$$\frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} = \begin{bmatrix} \frac{\partial W_x}{\partial p_1} & \frac{\partial W_x}{\partial p_2} & \cdots & \frac{\partial W_x}{\partial p_N} \\ \frac{\partial W_y}{\partial p_1} & \frac{\partial W_y}{\partial p_2} & \cdots & \frac{\partial W_y}{\partial p_N} \end{bmatrix}$$

$$\frac{\partial W_x}{\partial p_1} = \qquad \frac{\partial W_x}{\partial p_2} = \qquad \cdots$$

$$\frac{\partial W_y}{\partial p_1} = \qquad \cdots$$

$$\frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} =$$

The Jacobian $\dfrac{\partial \mathbf{W}}{\partial \boldsymbol{p}}$

(A matrix of partial derivatives)

Rate of change of the warp

$$\boldsymbol{x} = \begin{bmatrix} x \\ y \end{bmatrix}$$

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

$$\frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} = \begin{bmatrix} \frac{\partial W_x}{\partial p_1} & \frac{\partial W_x}{\partial p_2} & \cdots & \frac{\partial W_x}{\partial p_N} \\ \frac{\partial W_y}{\partial p_1} & \frac{\partial W_y}{\partial p_2} & \cdots & \frac{\partial W_y}{\partial p_N} \end{bmatrix}$$

$$\frac{\partial W_x}{\partial p_1} = x \qquad \frac{\partial W_x}{\partial p_2} = 0 \qquad \cdots$$

$$\frac{\partial W_y}{\partial p_1} = 0 \qquad \cdots$$

$$\frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} = \begin{bmatrix} x & 0 & y & 0 & 1 & 0 \\ 0 & x & 0 & y & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} W_x(\mathbf{x};\mathbf{p}) \\ W_y(\mathbf{x};\mathbf{p}) \end{bmatrix} = \begin{bmatrix} p_1 x + p_2 y + p_3 \\ p_4 x + p_5 y + p_6 \end{bmatrix}$$

$$I\Big(\mathbf{W}(\mathbf{x};\mathbf{p})\Big) = I\Big(p_1 x + p_2 y + p_3, p_4 x + p_5 y + p_6\Big)$$

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

pixel coordinate
(2 x 1)

$$\begin{bmatrix} W_x(\mathbf{x};\mathbf{p}) \\ W_y(\mathbf{x};\mathbf{p}) \end{bmatrix} = \begin{bmatrix} p_1 x + p_2 y + p_3 \\ p_4 x + p_5 y + p_6 \end{bmatrix}$$

$$I\Big(\mathbf{W}(\mathbf{x};\mathbf{p})\Big) = I\Big(p_1 x + p_2 y + p_3, p_4 x + p_5 y + p_6\Big)$$

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

pixel coordinate
(2 x 1)

image intensity
(scalar)

$$\begin{bmatrix} W_x(\mathbf{x};\mathbf{p}) \\ W_y(\mathbf{x};\mathbf{p}) \end{bmatrix} = \begin{bmatrix} p_1 x + p_2 y + p_3 \\ p_4 x + p_5 y + p_6 \end{bmatrix}$$

$$I\Big(\mathbf{W}(\mathbf{x};\mathbf{p})\Big) = I\Big(p_1 x + p_2 y + p_3, p_4 x + p_5 y + p_6\Big)$$

warp function
(2 x 1)

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

pixel coordinate
(2 x 1)

image intensity
(scalar)

$$\begin{bmatrix} W_x(\mathbf{x};\mathbf{p}) \\ W_y(\mathbf{x};\mathbf{p}) \end{bmatrix} = \begin{bmatrix} p_1 x + p_2 y + p_3 \\ p_4 x + p_5 y + p_6 \end{bmatrix}$$

$$I\Big(\mathbf{W}(\mathbf{x};\mathbf{p})\Big) = I\Big(p_1 x + p_2 y + p_3, p_4 x + p_5 y + p_6\Big)$$

warp function
(2 x 1)

warp parameters
(6 for affine)

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

pixel coordinate
(2 x 1)

image intensity
(scalar)

$$\begin{bmatrix} W_x(\mathbf{x};\mathbf{p}) \\ W_y(\mathbf{x};\mathbf{p}) \end{bmatrix} = \begin{bmatrix} p_1 x + p_2 y + p_3 \\ p_4 x + p_5 y + p_6 \end{bmatrix}$$

$$I\Big(\mathbf{W}(\mathbf{x};\mathbf{p})\Big) = I\Big(p_1 x + p_2 y + p_3, p_4 x + p_5 y + p_6\Big)$$

warp function
(2 x 1)

warp parameters
(6 for affine)

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

image gradient
(1 x 2)

pixel coordinate
(2 x 1)

image intensity
(scalar)

$$\begin{bmatrix} W_x(\mathbf{x};\mathbf{p}) \\ W_y(\mathbf{x};\mathbf{p}) \end{bmatrix} = \begin{bmatrix} p_1 x + p_2 y + p_3 \\ p_4 x + p_5 y + p_6 \end{bmatrix}$$

$$I\Big(\mathbf{W}(\mathbf{x};\mathbf{p})\Big) = I\Big(p_1 x + p_2 y + p_3, p_4 x + p_5 y + p_6\Big)$$

warp function
(2 x 1)

warp parameters
(6 for affine)

Partial derivatives of warp function
(2 x 6)

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

pixel coordinate
(2 x 1)

image intensity
(scalar)

image gradient
(1 x 2)

$$\begin{bmatrix} W_x(\mathbf{x};\mathbf{p}) \\ W_y(\mathbf{x};\mathbf{p}) \end{bmatrix} = \begin{bmatrix} p_1 x + p_2 y + p_3 \\ p_4 x + p_5 y + p_6 \end{bmatrix}$$

$$I\Big(\mathbf{W}(\mathbf{x};\mathbf{p})\Big) = I\Big(p_1 x + p_2 y + p_3, p_4 x + p_5 y + p_6\Big)$$

warp function
(2 x 1)

warp parameters
(6 for affine)

Partial derivatives of warp function
(2 x 6)

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

pixel coordinate
(2 x 1)

image intensity
(scalar)

image gradient
(1 x 2)

incremental warp
(6 x 1)

$$\begin{bmatrix} W_x(\mathbf{x};\mathbf{p}) \\ W_y(\mathbf{x};\mathbf{p}) \end{bmatrix} = \begin{bmatrix} p_1 x + p_2 y + p_3 \\ p_4 x + p_5 y + p_6 \end{bmatrix}$$

$$I\Big(\mathbf{W}(\mathbf{x};\mathbf{p})\Big) = I\Big(p_1 x + p_2 y + p_3, p_4 x + p_5 y + p_6\Big)$$

warp function
(2 x 1)

warp parameters
(6 for affine)

Partial derivatives of warp function
(2 x 6)

template image intensity
(scalar)

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

pixel coordinate
(2 x 1)

image intensity
(scalar)

image gradient
(1 x 2)

incremental warp
(6 x 1)

What variables depend on pixel coordinates **x**?

warp function
(2 x 1)

warp parameters
(6 for affine)

Partial derivatives of warp function
(2 x 6)

template image intensity
(scalar)

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

pixel coordinate
(2 x 1)

image intensity
(scalar)

image gradient
(1 x 2)

incremental warp
(6 x 1)

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

$$\frac{\partial W}{\partial \mathbf{p}} \Rightarrow \frac{\partial W(\mathbf{x};\mathbf{p})}{\partial \mathbf{p}}$$

Recall tricky shorthand notation!

Summary

(of Lucas-Kanade Image Alignment)

Problem:

$$\min_{\boldsymbol{p}} \sum_{\boldsymbol{x}} \left[I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p})) - T(\boldsymbol{x}) \right]^2$$

warped image — template image

Difficult non-linear optimization problem

Strategy:

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

Solve for increment assuming a good initial guess

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

Assume increment will be small and linearize with Taylor series

… and then solve for $\Delta\mathbf{p}$

OK, so how to solve this?

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

(move terms around)

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

vector of constants

vector of variables

constant

look familiar?

$$\min_{\mathbf{x}} \sum_{i} (A[i,:]\mathbf{x} - b[i])^2$$

$$\min_{\mathbf{x}} ||A\mathbf{x} - \mathbf{b}||^2$$

what are dimensions of A?
#pixels by 6 (for affine)

Hint: when implementing, explicitly compute matrix A and vector b in parallel and avoid looping over pixels

You may solve for $\Delta \mathbf{p}$ with np.linalg.lstsq(A,b) or np.linalg.svd(A) or … [next slides]

Recall least squares soln for overdetermined (tall,skinny) A is given by

$$x = (A^\top A)^{-1} A^\top b$$

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

is optimized when

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

after applying $x = (A^\top A)^{-1} A^\top b$

approximate Hessian matrix

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

$A^\top A$

what are the dimensions of H? 6x6 (for affine)

If this is confusing, read up on matrix-matrix products (Sec 2.3) in https://www.cs.cmu.edu/~zkolter/course/15-884/linalg-review.pdf

Solve:

$$\min_{\boldsymbol{p}} \sum_{\boldsymbol{x}} \left[I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p})) - T(\boldsymbol{x}) \right]^2$$

warped image — template image

Difficult non-linear optimization problem

Strategy:

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

Assume known approximate solution

Solve for increment

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

Taylor series approximation

Linearize

Solution:

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

Solution to least squares approximation

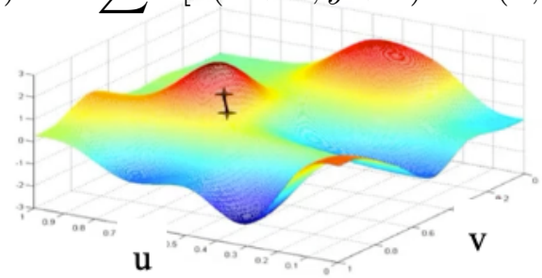
$$H = \sum_{\boldsymbol{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \right]^\top \left[\nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \right]$$

Approximate Hessian

What's really going on under the hood: nonlinear least squares optimization via Gauss-Newton descent

$$E(u,v) = \frac{1}{2}\left\| \begin{matrix} f(u,v) \\ g(u,v) \end{matrix} \right\|^2 = \frac{1}{2}||\mathbf{f}(\mathbf{u})||^2$$

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



1. First order method (gradient descent)

$$\begin{bmatrix} u \\ v \end{bmatrix} := \begin{bmatrix} u \\ v \end{bmatrix} - \alpha \begin{bmatrix} \frac{\partial f}{\partial u} & \frac{\partial f}{\partial v} \\ \frac{\partial g}{\partial u} & \frac{\partial g}{\partial v} \end{bmatrix}^T \begin{bmatrix} f(u,v) \\ g(u,v) \end{bmatrix}$$

jacobian $\mathbf{J}(\mathbf{u})$

1.5 Gauss-Newton descent

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

$$\mathbf{f}(\mathbf{u} + \boldsymbol{\Delta}\mathbf{u}) \approx \mathbf{f}(\mathbf{u}) + \mathbf{J}(\mathbf{u})\boldsymbol{\Delta}\mathbf{u}$$

$$\mathbf{u} := \mathbf{u} + \arg\min_{\boldsymbol{\Delta}\mathbf{u}} \frac{1}{2}||\mathbf{f}(\mathbf{u}) + \mathbf{J}(\mathbf{u})\boldsymbol{\Delta}\mathbf{u}||^2$$

Solve for update with least squares $\min_x ||-b + Ax||^2$
Get the performance of second-order optimizers with first-order compute
(without computing second derivatives)!

Lucas Kanade (Additive alignment)

1. Warp image $I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p}))$
2. Compute error image $[T(\boldsymbol{x}) - I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p}))]$
3. Compute gradient $\nabla I(\mathbf{W}(\mathbf{x};\mathbf{p}))$ dx,dy gradients of warped image
4. Evaluate Jacobian $\frac{\partial \mathbf{W}}{\partial \boldsymbol{p}}$
5. Compute Hessian H $H = \sum_{\boldsymbol{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}}\right]^{\top} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}}\right]$
6. Compute $\Delta \boldsymbol{p}$ $\Delta \boldsymbol{p} = H^{-1} \sum_{\boldsymbol{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}}\right]^{\top} [T(\boldsymbol{x}) - I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p}))]$
7. Update parameters $\boldsymbol{p} \leftarrow \boldsymbol{p} + \Delta \boldsymbol{p}$

Just 8 lines of code!

HW: 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 (or not?)

Important extensions

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

image

template

Key insight: exploit flexibility of warping the image *or* warping the template

Image Alignment

(start with an initial solution, match the image and template)

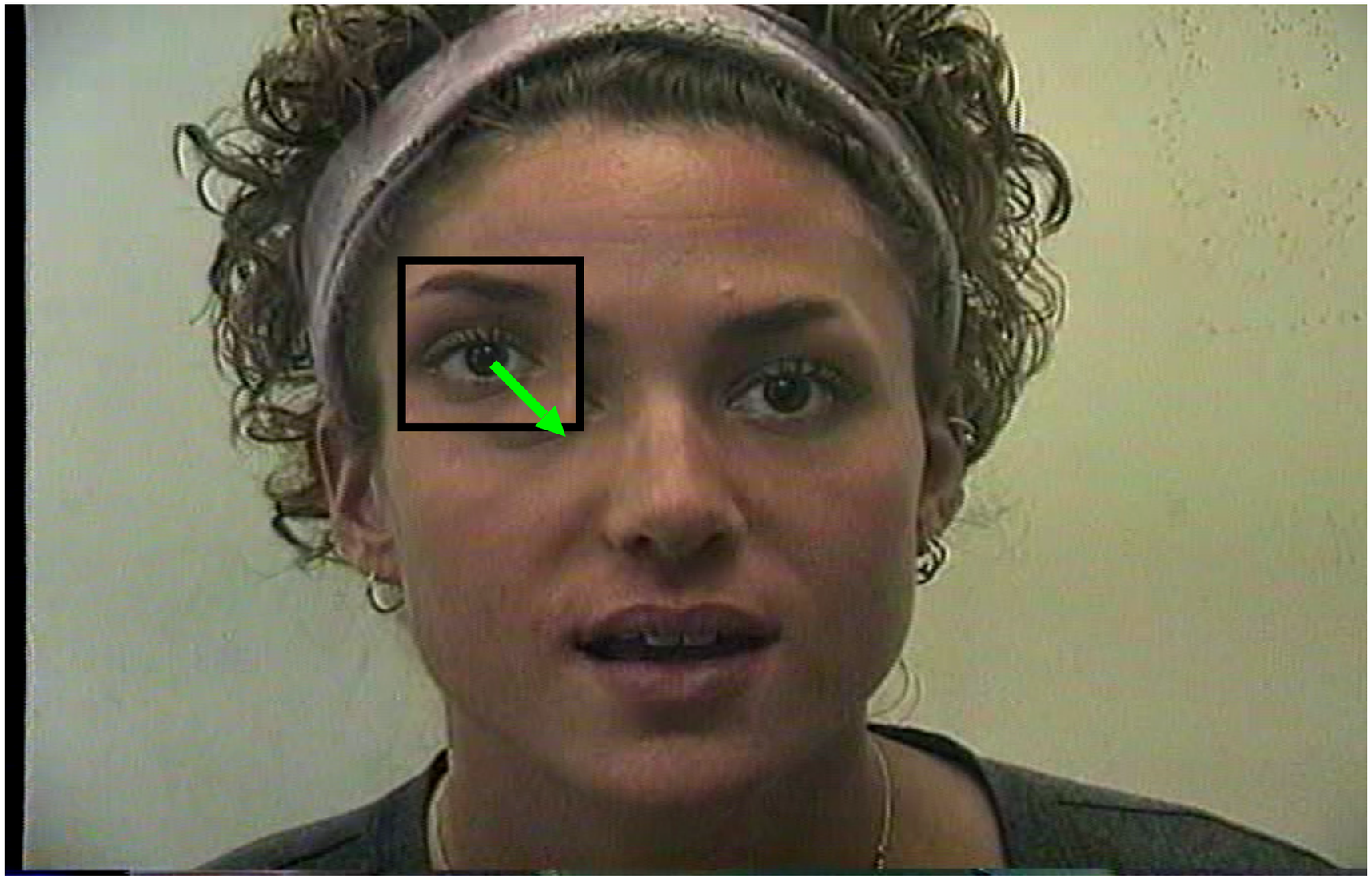


Image Alignment Objective Function

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

Given an initial solution…several possible formulations

Additive Alignment

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

incremental perturbation of parameters

Image Alignment Objective Function

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

Given an initial solution…several possible formulations

Additive Alignment

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

incremental perturbation of parameters

Compositional Alignment

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

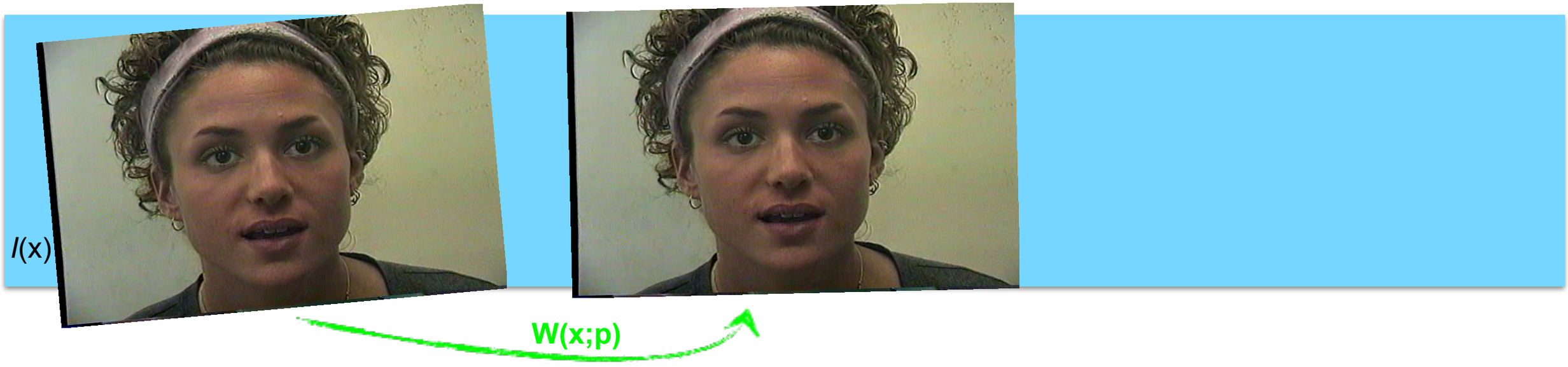
incremental warps of image

Additive strategy

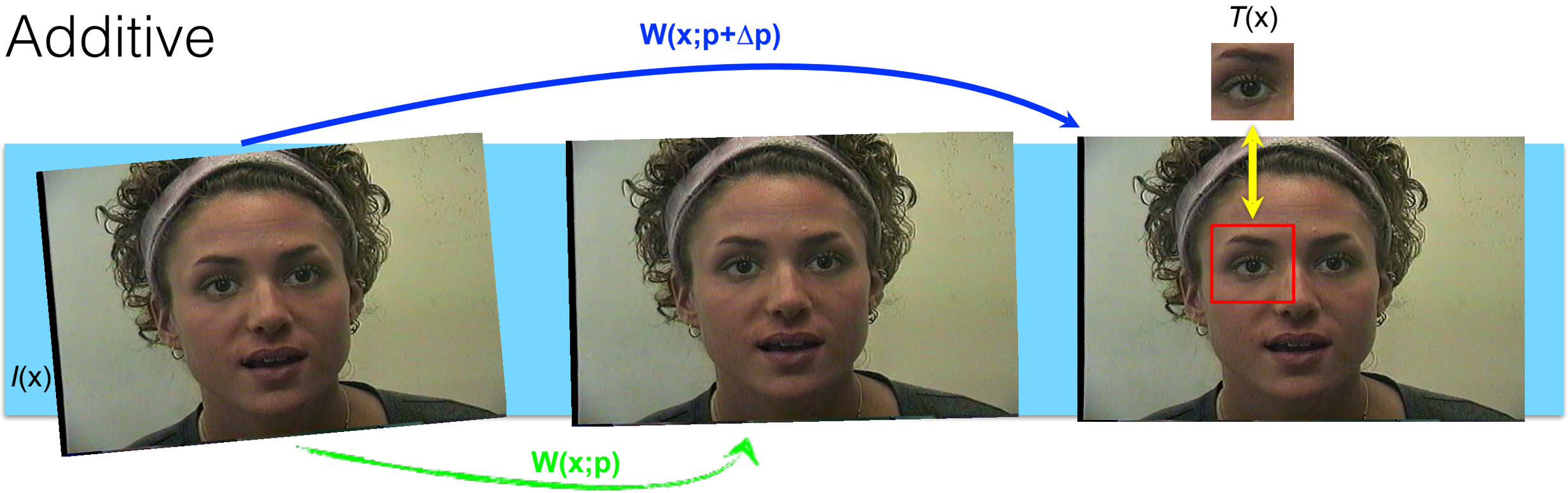
[Loose analogy]

Compositional strategy

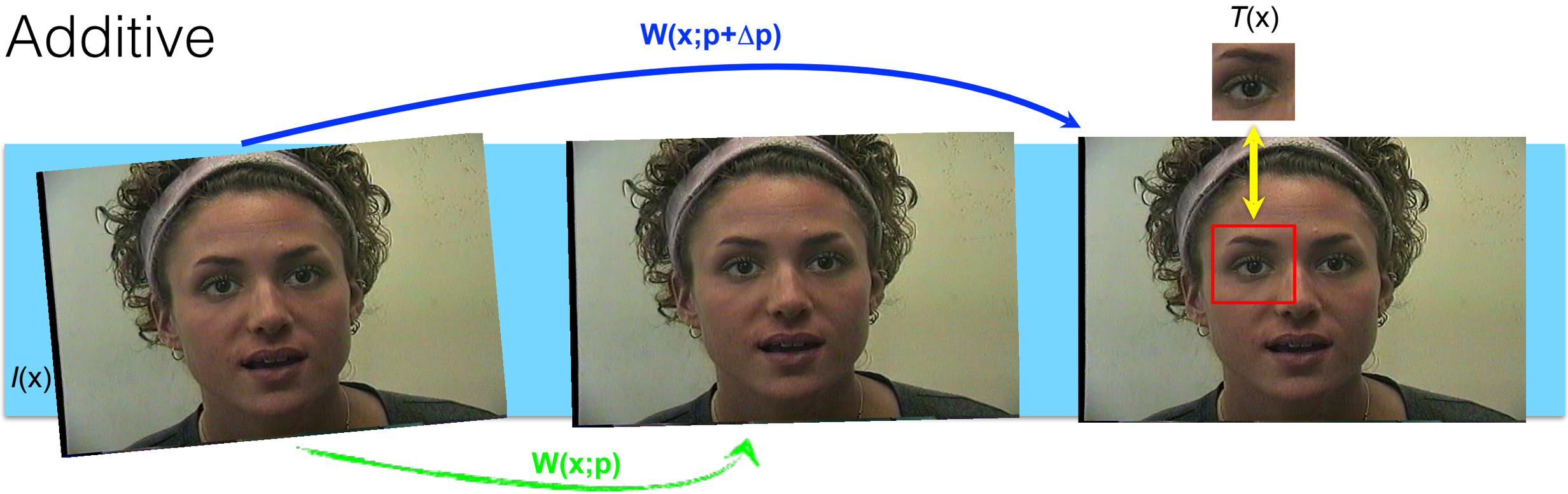
Additive



Additive

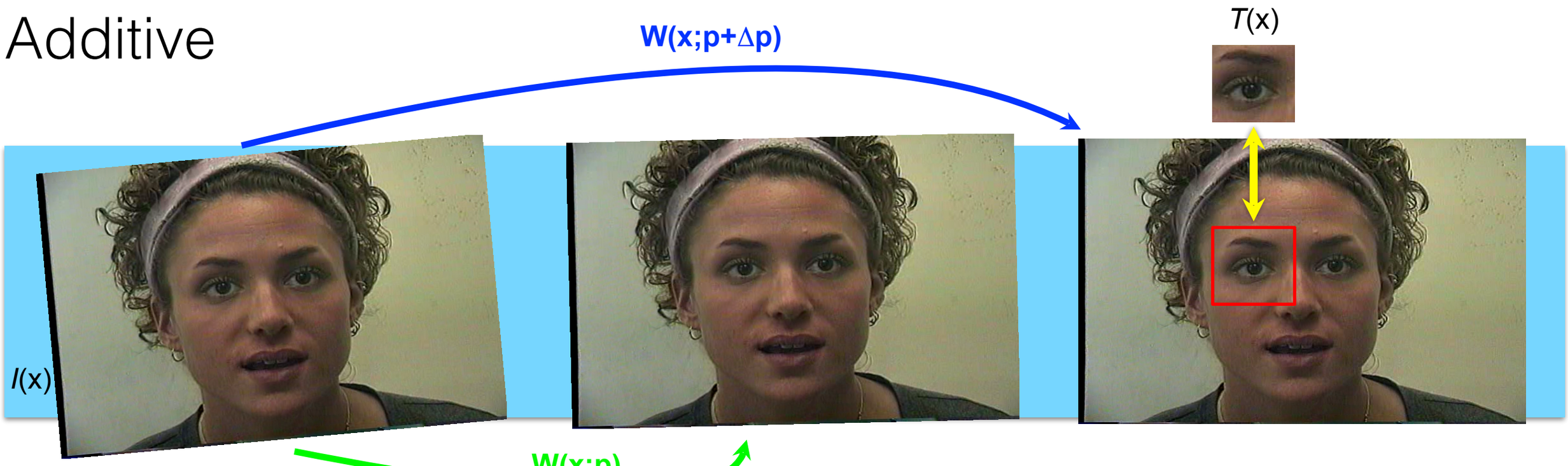


Additive

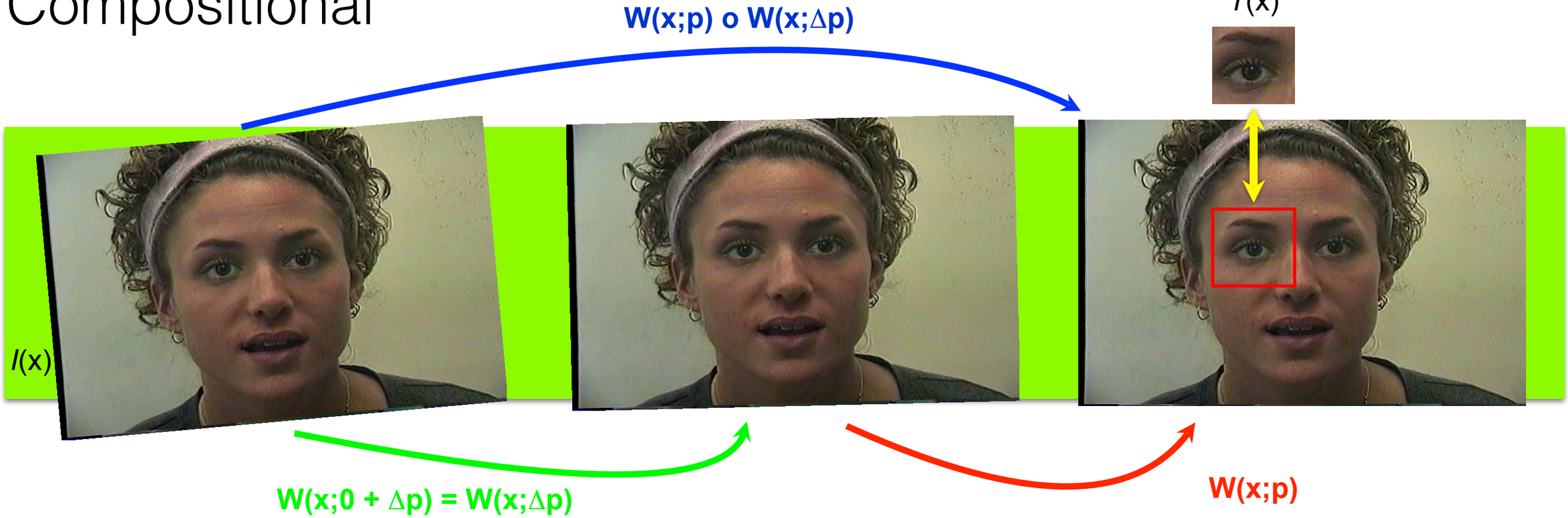


Compositional (with a twist; apply Δp warp *first* instead of *second*. We'll see why soon!)

Additive



Compositional



Compositional Alignment

Original objective function (SSD)

$$\min_{\boldsymbol{p}} \sum_{\boldsymbol{x}} \left[I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p})) - T(\boldsymbol{x}) \right]^2$$

Assuming an initial solution **p** and a compositional warp increment

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

Compositional Alignment

Original objective function (SSD)

$$\min_{\boldsymbol{p}} \sum_{\boldsymbol{x}} \left[I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p})) - T(\boldsymbol{x}) \right]^2$$

Assuming an initial solution **p** and a compositional warp increment

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

Another way to write the composition

$$\mathbf{W}(\boldsymbol{x};\boldsymbol{p}) \circ \mathbf{W}(\boldsymbol{x};\Delta\boldsymbol{p}) \equiv \mathbf{W}(\ \mathbf{W}(\boldsymbol{x};\Delta\boldsymbol{p});\boldsymbol{p}\ )$$

Identity warp

$$\mathbf{W}(\boldsymbol{x};\mathbf{0})$$

Compositional Alignment

Original objective function (SSD)

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

Assuming an initial solution **p** and a compositional warp increment

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

Another way to write the composition

$$\mathbf{W}(\boldsymbol{x};\boldsymbol{p}) \circ \mathbf{W}(\boldsymbol{x};\Delta\boldsymbol{p}) \equiv \mathbf{W}(\ \mathbf{W}(\boldsymbol{x};\Delta\boldsymbol{p});\boldsymbol{p}\ )$$

Identity warp

$$\mathbf{W}(\boldsymbol{x};\mathbf{0})$$

Skipping over the derivation…the new update rule is

$$\mathbf{W}(\boldsymbol{x};\boldsymbol{p}) \leftarrow \mathbf{W}(\boldsymbol{x};\boldsymbol{p}) \circ \mathbf{W}(\boldsymbol{x};\Delta\boldsymbol{p})$$

So what's so great about this compositional form?

Additive Alignment

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

linearized form

$$\sum_{\boldsymbol{x}} \left[I(\mathbf{W}(\boldsymbol{x}; \boldsymbol{p})) + \nabla I(\boldsymbol{x}') \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \Delta \boldsymbol{p} - T(\boldsymbol{x}) \right]^2$$

Compositional Alignment

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

linearized form

$$\sum_{\boldsymbol{x}} \left[I(\mathbf{W}(\boldsymbol{x}; \boldsymbol{p})) + \nabla I(\boldsymbol{x}') \frac{\partial \mathbf{W}(\boldsymbol{x}; \mathbf{0})}{\partial \boldsymbol{p}} \Delta \boldsymbol{p} - T(\boldsymbol{x}) \right]^2$$

Additive Alignment

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

linearized form

$$\sum_{\boldsymbol{x}} \left[I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p})) + \nabla I(\boldsymbol{x}') \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \Delta\boldsymbol{p} - T(\boldsymbol{x}) \right]^2$$

Jacobian of W(x;p)

Compositional Alignment

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

linearized form

$$\sum_{\boldsymbol{x}} \left[I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p})) + \nabla I(\boldsymbol{x}') \frac{\partial \mathbf{W}(\boldsymbol{x};\mathbf{0})}{\partial \boldsymbol{p}} \Delta\boldsymbol{p} - T(\boldsymbol{x}) \right]^2$$

Jacobian of W(x;0)

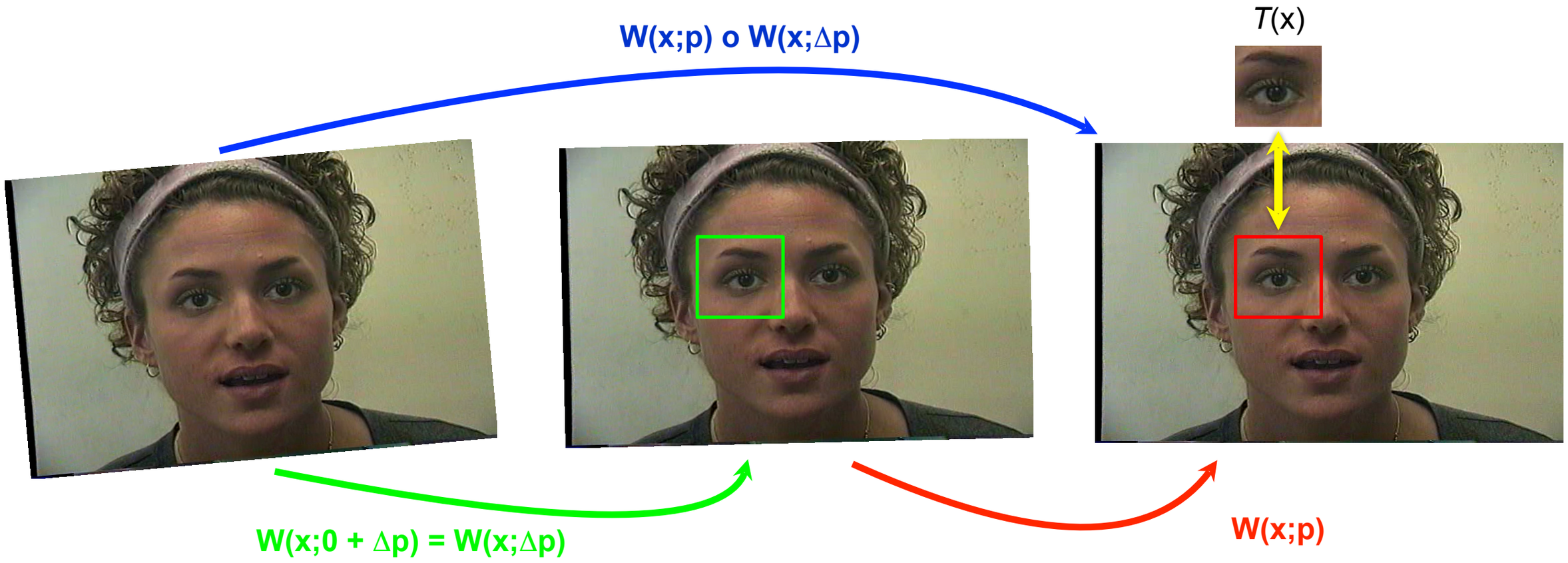
The Jacobian is constant and can be precomputed!

(benefit of applying incremental update *first*, since it requires linearizing about p = 0)

Compositional Image Alignment

Minimize

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



Jacobian is simple and can be precomputed

Lucas Kanade (Additive alignment)

1. Warp image $I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p}))$
2. Compute error image $[T(\boldsymbol{x}) - I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p}))]^2$
3. Compute gradient $\nabla I(\boldsymbol{x}')$
4. Evaluate Jacobian $\frac{\partial \mathbf{W}}{\partial \boldsymbol{p}}$
5. Compute Approx Hessian H
6. Compute $\Delta \boldsymbol{p}$
7. Update parameters $\boldsymbol{p} \leftarrow \boldsymbol{p} + \Delta \boldsymbol{p}$

Shum-Szeliski (Compositional alignment)

1. Warp image $I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p}))$

2. Compute error image $[T(\boldsymbol{x}) - I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p}))]^2$

3. Compute gradient $\nabla I(\boldsymbol{x}')$

4. Evaluate Jacobian $\frac{\partial \mathbf{W}(\boldsymbol{x};\mathbf{0})}{\partial \boldsymbol{p}}$

5. Compute Approx Hessian H

6. Compute $\Delta \boldsymbol{p}$

7. Update parameters $\mathbf{W}(\boldsymbol{x};\boldsymbol{p}) \leftarrow \mathbf{W}(\boldsymbol{x};\boldsymbol{p}) \circ \mathbf{W}(\boldsymbol{x};\Delta \boldsymbol{p})$

Any other speed up techniques?

Inverse alignment

Why not compute warp updates on the template?

Additive Alignment

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

Compositional Alignment

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

Why not compute warp updates on the template?

Additive Alignment

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

Compositional Alignment

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

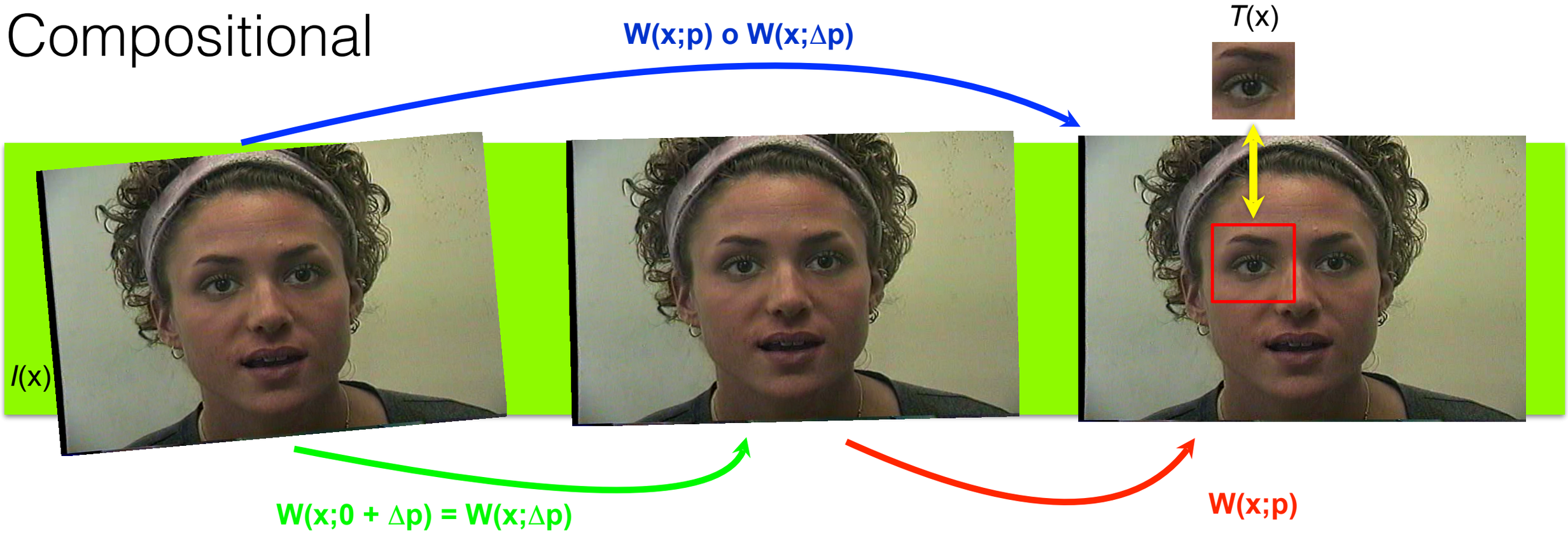
What happens if you let the template be warped too?

Inverse Compositional Alignment

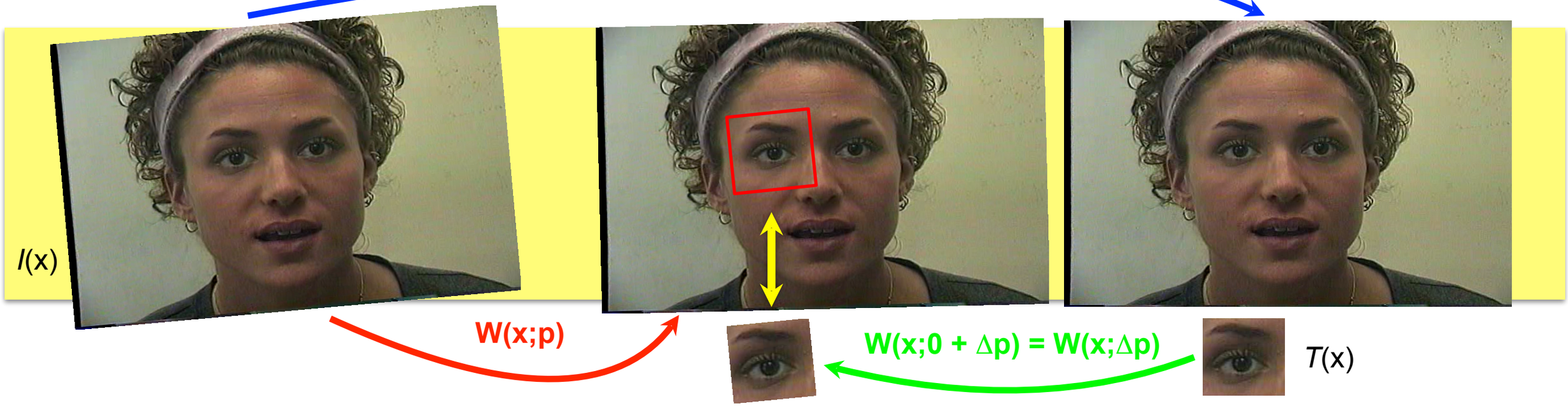
$$\sum_{\boldsymbol{x}} [T(\mathbf{W}(\boldsymbol{x};\Delta\boldsymbol{p})) - I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p}))]^2$$

1. Compute Δp warp on template.
2. But instead of applying it to the template, apply its *inverse* warp to the image. *This implies the linearized template (ie, jacobians, hessians) never need be recomputed since the template never changes!*

Compositional



Inverse compositional



Compositional strategy

Inverse Compositional strategy

So what's so great about this inverse compositional form?

Inverse Compositional Alignment

Minimize

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

Solution

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

can be precomputed from template!

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

Update

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

Properties of inverse compositional alignment

Jacobian can be precomputed

It is constant - evaluated at W(x;0)

Gradient of template can be precomputed

It is constant

Approx Hessian can be precomputed

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

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

(main term that needs to be computed)

Warp must be invertible

Lucas Kanade (Additive alignment)

1. Warp image $I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p}))$
2. Compute error image $[T(\boldsymbol{x}) - I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p}))]^2$
3. Compute gradient $\nabla I(\mathbf{W})$
4. Evaluate Jacobian $\frac{\partial \mathbf{W}}{\partial \boldsymbol{p}}$
5. Compute Hessian H
6. Compute $\Delta \boldsymbol{p}$
7. Update parameters $\boldsymbol{p} \leftarrow \boldsymbol{p} + \Delta \boldsymbol{p}$

Shum-Szeliski (Compositional alignment)

1. Warp image $I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p}))$
2. Compute error image $[T(\boldsymbol{x}) - I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p}))]$
3. Compute gradient $\nabla I(\boldsymbol{x}')$
4. Evaluate Jacobian $\frac{\partial \mathbf{W}(\boldsymbol{x};\mathbf{0})}{\partial \boldsymbol{p}}$
5. Compute Approx Hessian H
6. Compute $\Delta \boldsymbol{p}$
7. Update parameters $\mathbf{W}(\boldsymbol{x};\boldsymbol{p}) \leftarrow \mathbf{W}(\boldsymbol{x};\boldsymbol{p}) \circ \mathbf{W}(\boldsymbol{x};\Delta \boldsymbol{p})$

Baker-Matthews (Inverse Compositional alignment)

1. Warp image $I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p}))$

2. Compute error image $[T(\boldsymbol{x}) - I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p}))]$

3. Compute gradient $\nabla T(\mathbf{W})$

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

5. Compute Approx Hessian H $H = \sum_{\boldsymbol{x}} \left[\nabla T \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}}\right]^{\top} \left[\nabla T \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}}\right]$

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

7. Update parameters $\mathbf{W}(\boldsymbol{x};\boldsymbol{p}) \leftarrow \mathbf{W}(\boldsymbol{x};\boldsymbol{p}) \circ \mathbf{W}(\boldsymbol{x};\Delta \boldsymbol{p})^{-1}$

Algorithm	Efficient	Authors
Forwards Additive	No	Lucas, Kanade
Forwards compositional	No	Shum, Szeliski
Inverse Additive	Yes	Hager, Belhumeur
Inverse Compositional	Yes	Baker, Matthews

A look back

- Geometric warps
  - Foreshortening-vs-projective
  - Peicewise affine
  - Fwd-vs-inverse warps
- Lucas-Kanade Alignment
  - Gauss Newton optimiztion of nonlinear least squares
  - Inverse composition