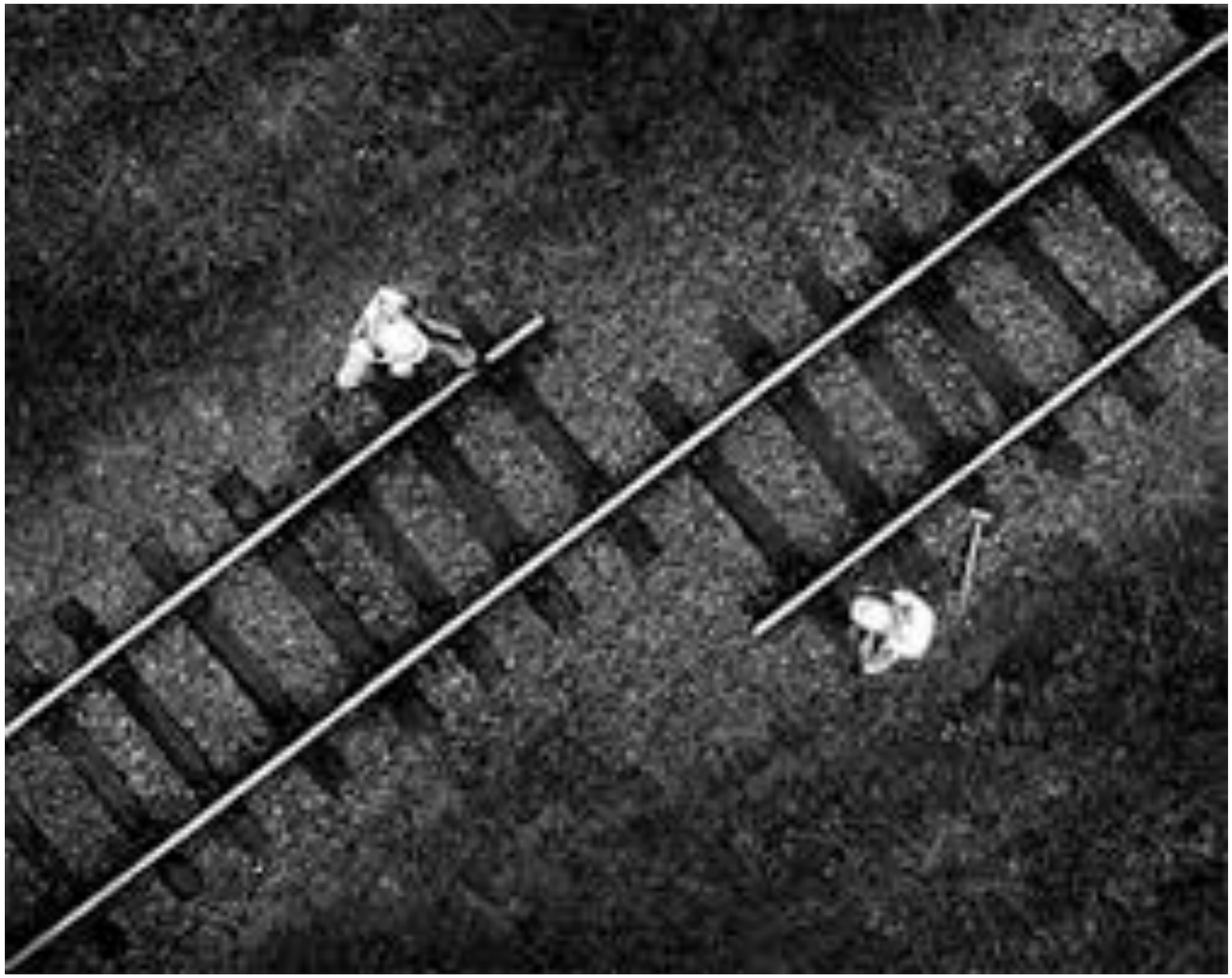
Transformations and alignment

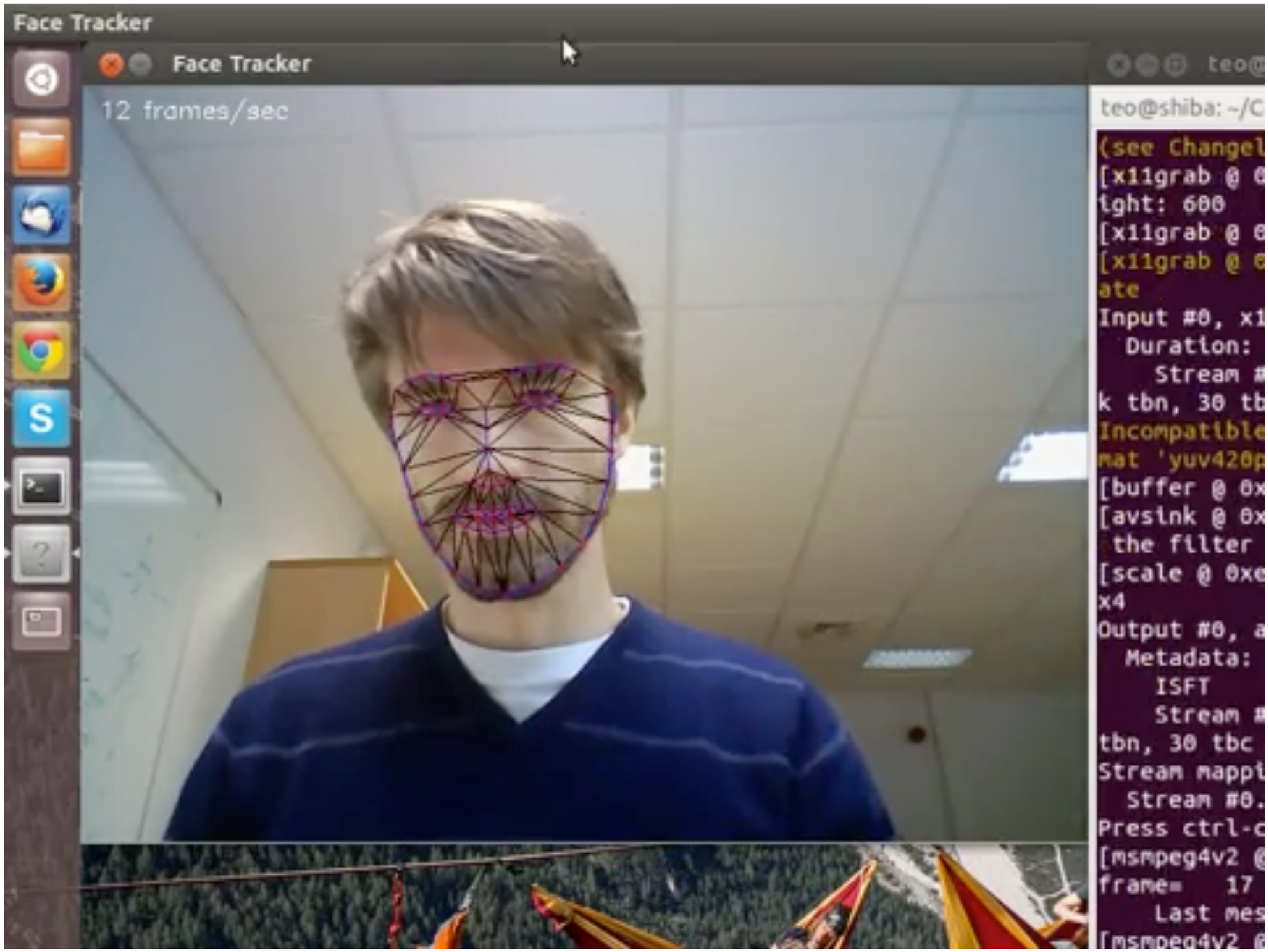


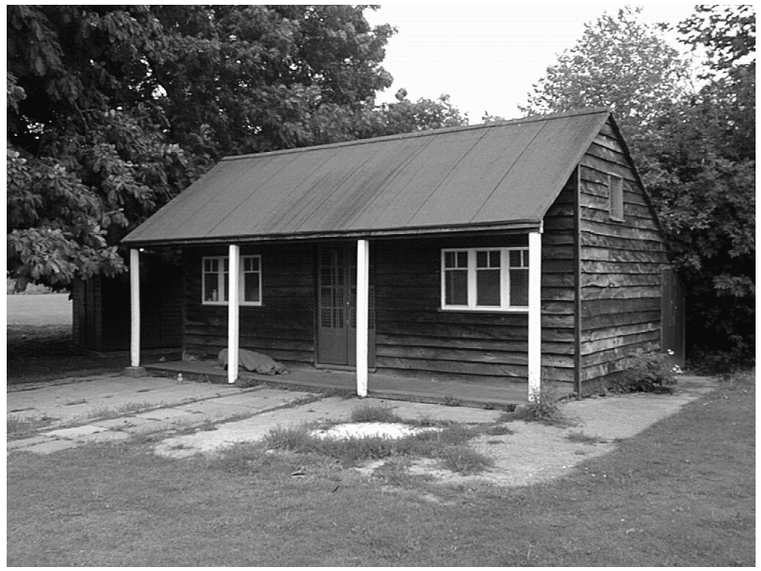
Slide credits: Deva Ramanan, Jitendra Malik

Outline

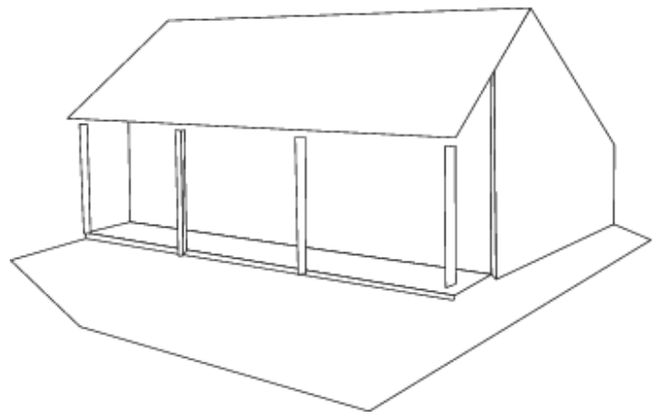
- Transformations (2D/3D)
- Direct methods
- Lucas Kanade

Where we are headed…





Given a 2D image



Can we infer the 3D world that created this image?

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

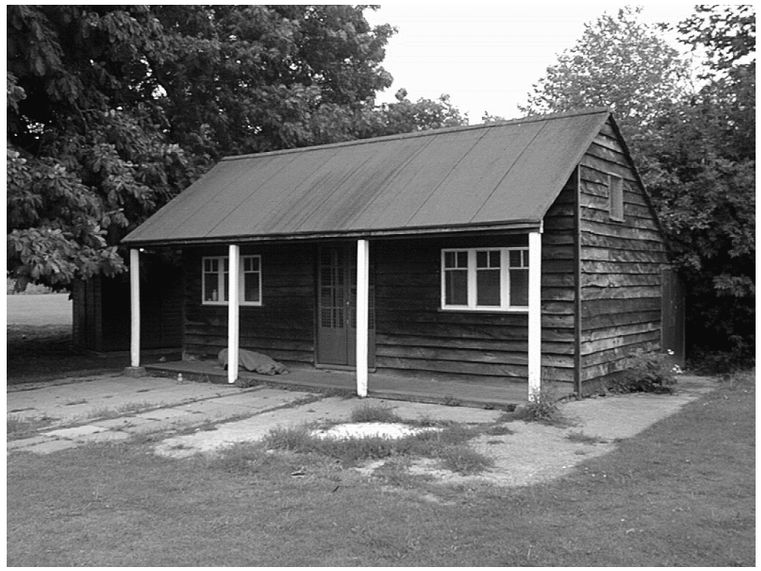
2D image points | **Camera matrix M** | **3D points X**

Problem: both M and X are unknown!

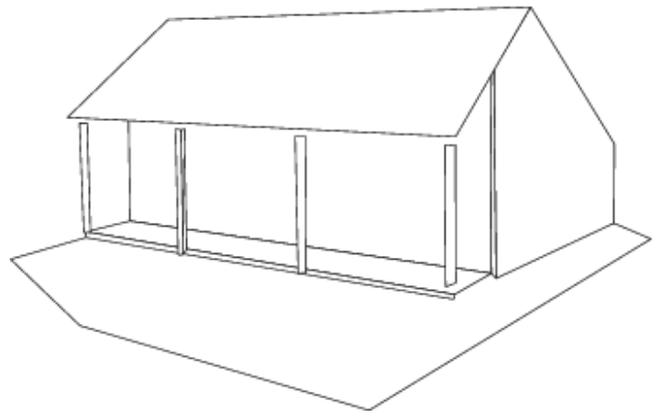
We can generate the same image with a modified camera M'=MQ and modified 3D geometry $Q^{-1}\mathbf{X}$ for any invertible matrix Q

$$M\mathbf{X} = \underbrace{MQ}\,\underbrace{Q^{-1}\mathbf{X}}$$

New Camera matrix | **New 3D points**

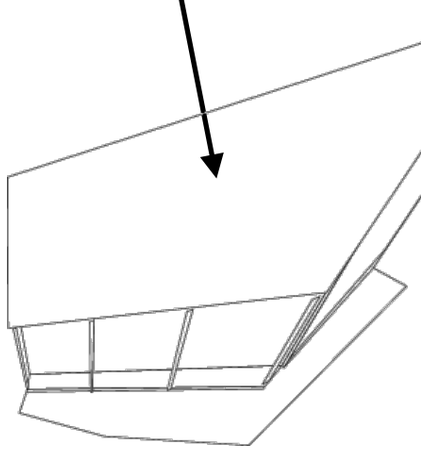


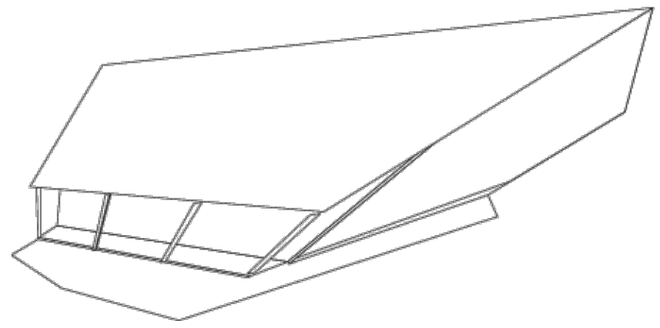
Given a 2D image



Can we infer the 3D world that created this image?

How do we know that this wasn't the 3D world that created this image?





$$M\mathbf{X} = \underbrace{MQ}_{\text{New Camera matrix}}\underbrace{Q^{-1}\mathbf{X}}_{\text{New 3D points}}$$

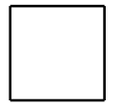
The different possible 3D reconstructions are all related by $Q^{-1}X$ (a 3D homography!)

Thus, what do we know about the 3D world?

straight lines, intersection

Not: angles, lengths, parallel lines

Family of 2D warps

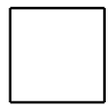
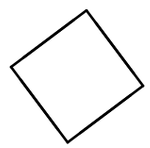
Transformation	Matrix	# DoF	Preserves	Icon
translation	$\begin{bmatrix} \boldsymbol{I} \mid \boldsymbol{t} \end{bmatrix}_{2\times 3}$	2	orientation	

$$x' = x + t_x$$

$$y' = y + t_y$$

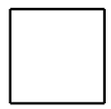
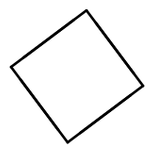
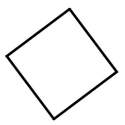
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

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	

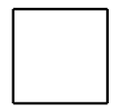
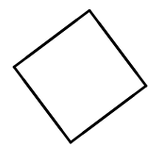
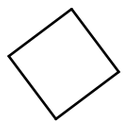
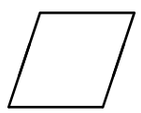
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Family of 2D warps

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

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

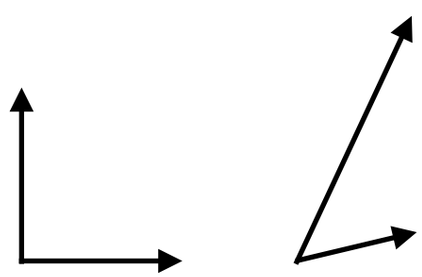
Family of 2D warps

Transformation	Matrix	# DoF	Preserves	Icon
translation	$\begin{bmatrix} \boldsymbol{I} \mid \boldsymbol{t} \end{bmatrix}_{2\times3}$	2	orientation	
rigid (Euclidean)	$\begin{bmatrix} \boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times3}$	3	lengths	
similarity	$\begin{bmatrix} s\boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times3}$	4	angles	
affine	$\begin{bmatrix} \boldsymbol{A} \end{bmatrix}_{2\times3}$	6	parallelism	

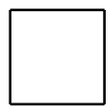
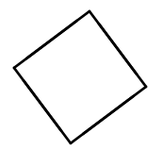
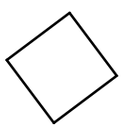
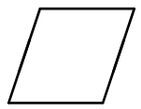
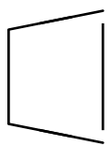
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

change of basis (rotate, scale x and y, rotate)

2D translation



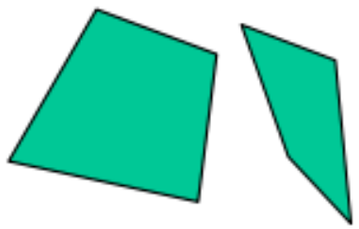
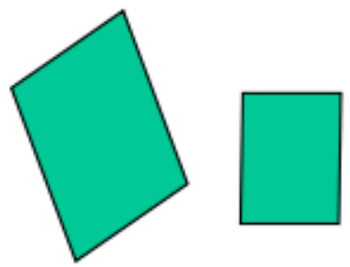
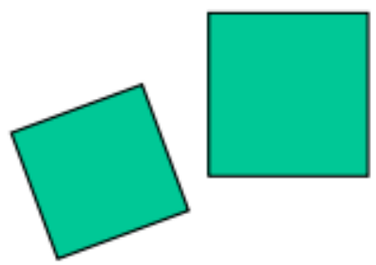
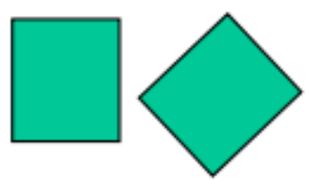
Family of 2D warps

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 any camera or (2) any scene under rotated cameras

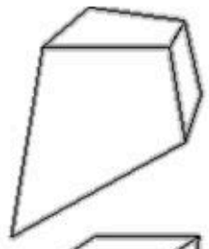
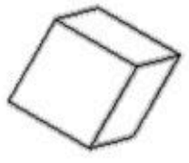
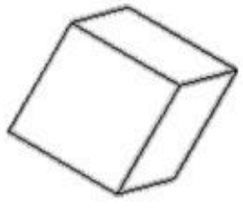
Hierarchy of 2D Transformations

		transformed squares	invariants
Projective 8dof	$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$		Concurrency, collinearity, order of contact (intersection, tangency, inflection, etc.), cross ratio
Affine 6dof	$\begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$		Parallelism, ratio of areas, ratio of lengths on parallel lines (e.g. midpoints), linear combinations of vectors (centroids), **The line at infinity l_∞**
Similarity 4dof	$\begin{bmatrix} sr_{11} & sr_{12} & t_x \\ sr_{21} & sr_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$		Ratios of lengths, angles, The circular points I,J
Euclidean 3dof	$\begin{bmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$		Absolute lengths, angles, areas

ETH

Slide credit: Marc Pollefeys Viktor Larsson

Transformations in 3D

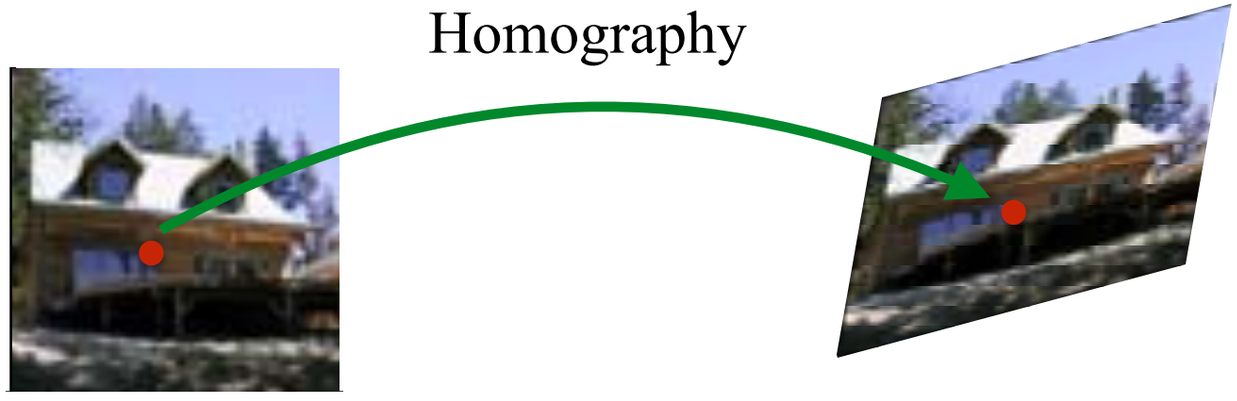
Projective 15dof	$\begin{bmatrix} A & t \\ v^T & v \end{bmatrix}$		Preserves intersection and tangency
Affine 12dof	$\begin{bmatrix} A & t \\ 0^T & 1 \end{bmatrix}$		Preserves parallellism, volume ratios
Similarity 7dof	$\begin{bmatrix} sR & t \\ 0^T & 1 \end{bmatrix}$		Preserves angles, ratios of length
Euclidean 6dof	$\begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix}$		Preserves angles, lengths

"rigid body"

Make use of 3D homogenous coordinates (Normalize by last coordinate to recover 3D points)

$$\lambda \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

How to warp an image



forward warp

“splat” color onto nearby grid points
(might still result in artifacts/holes)

How can we fix this?

inverse warp

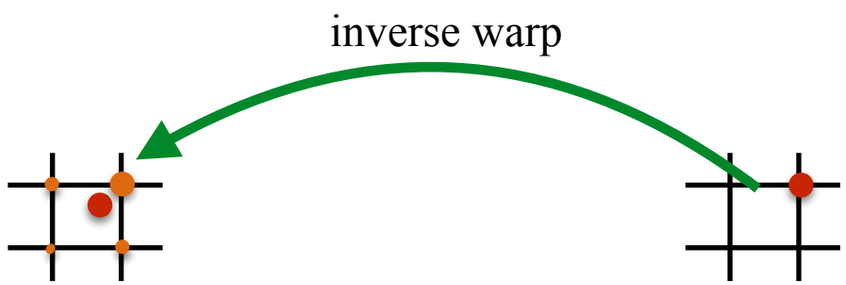
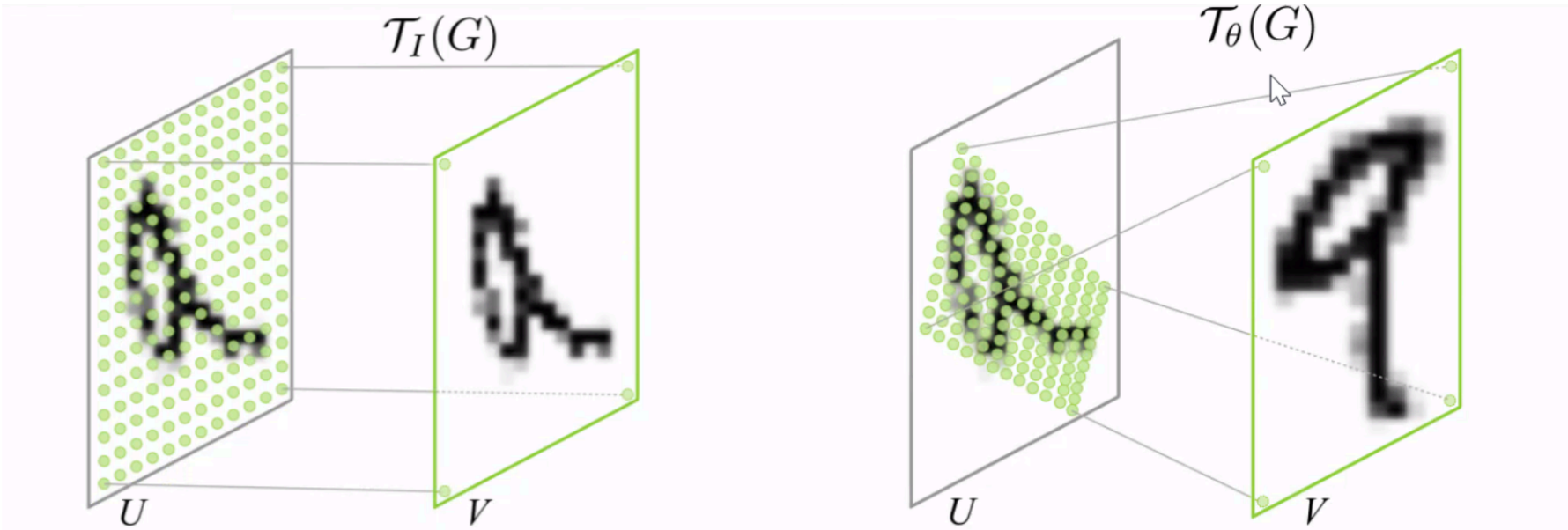


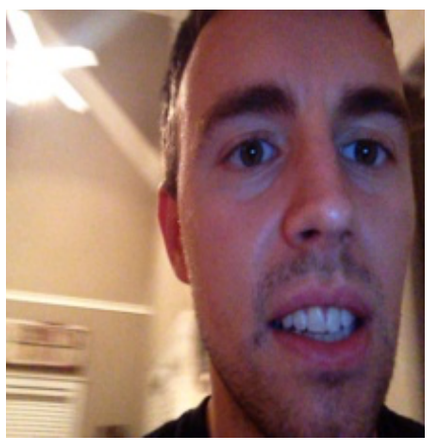
figure out from where each grid point should “steal” color

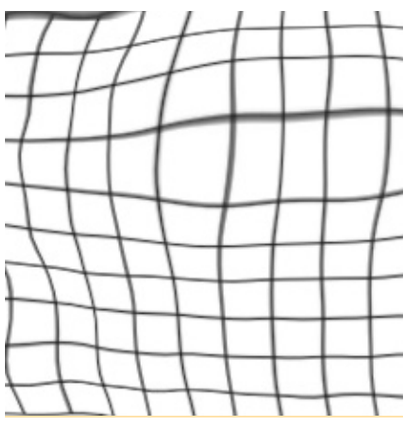
bilinearly interpolate

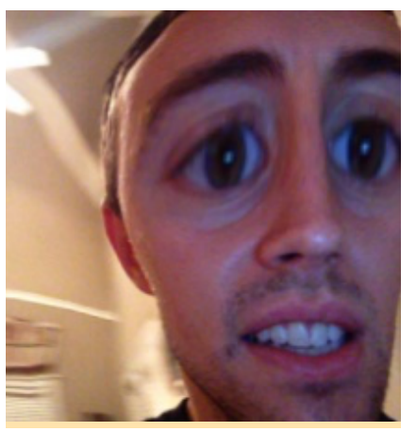
Digression: spatial transformer networks

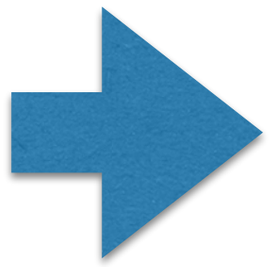


Neural networks that explicitly warp input images (or even feature maps)









classify eye_gaze direction

Spatial Transformer Networks

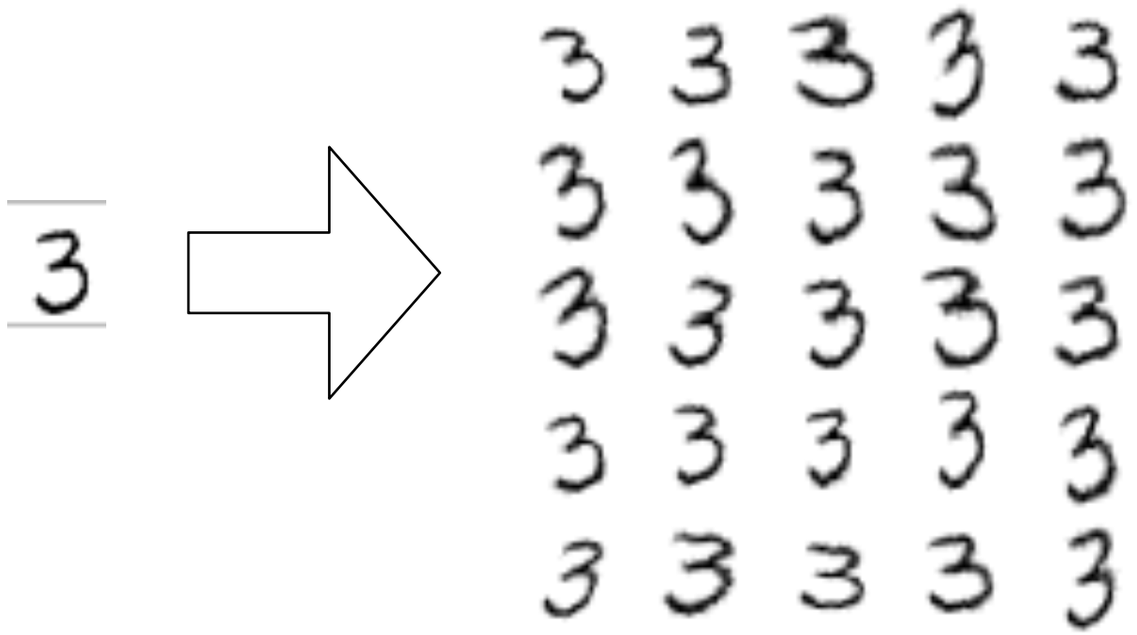
Max Jaderberg Karen Simonyan Andrew Zisserman Koray Kavukcuoglu

Google DeepMind, London, UK
{jaderberg,simonyan,zisserman,korayk}@google.com

Learning to Zoom: a Saliency-Based Sampling Layer for Neural Networks

Adrià Recasens*[1], Petr Kellnhofer*[1], Simon Stent[2], Wojciech Matusik[1], and Antonio Torralba[1]

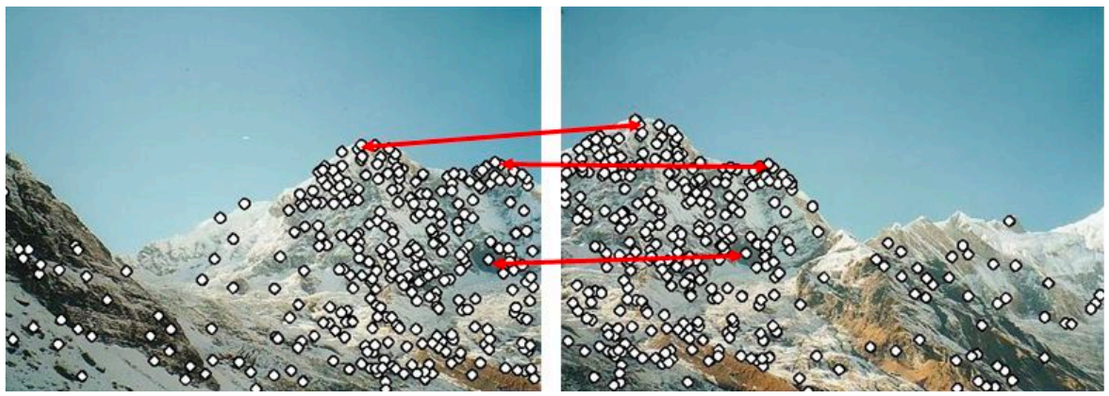
Example application: data augmentation



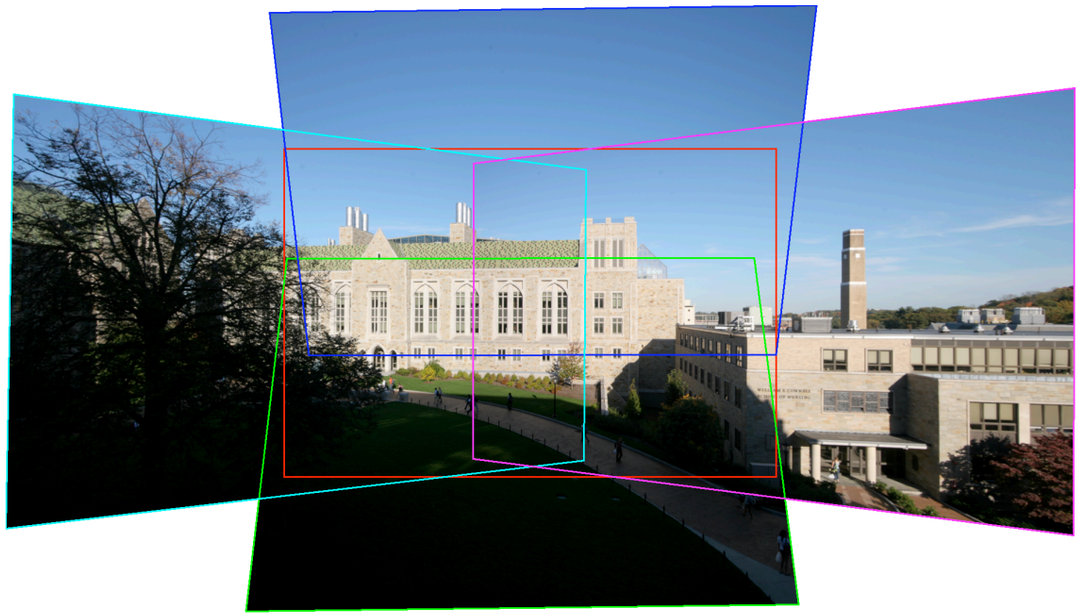
Outline

- Transformations (2D/3D)
- Direct methods
- Lucas Kanade

Approaches for image alignment



Alignment from sparse interest points



Dense, direct estimation of warp using **all** pixels

HW2: Tracking

Tracking

current frame

template
(model)

u,v = hypothesized location of
template in current frame

We can track an object by matching a template to an image in every frame

Tracking

current frame

template
(model)

u,v = hypothesized location of
template in current frame

$$\min_{u,v} E(u,v), \quad \text{where} \quad E(u,v) = \sum_{x,y} [I(x+u, y+v) - T(x,y)]^2$$

image patch at (u,v) **template**

What is the transformation that is being searched over? **Translation**

Think about: how would you modify this if you wanted to search over affine transformations

Tracking

current frame

template
(model)

u,v = hypothesized location of
template in current frame

$$\min_{u,v} E(u,v), \quad \text{where} \quad E(u,v) = \sum_{x,y} [I(x+u, y+v) - T(x,y)]^2$$

image patch at (u,v) **template**

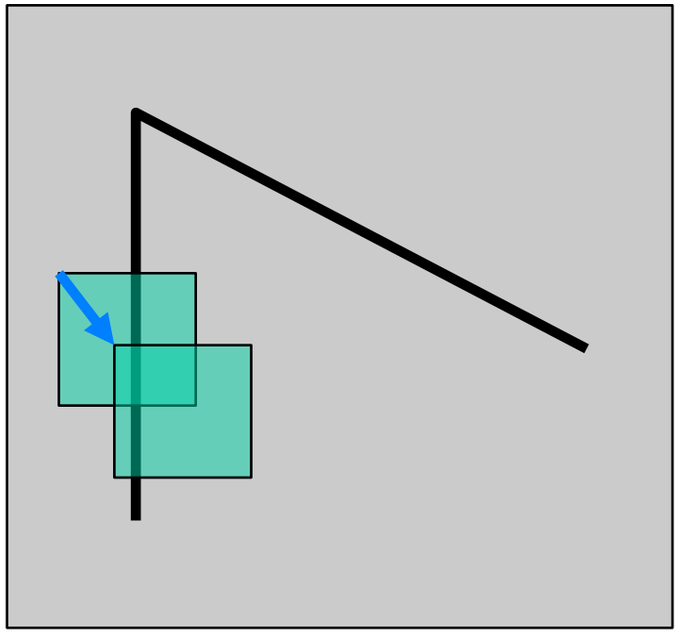
This is Non-linear Least squares

Why is it non-linear? **The image function I(…) is nonlinear**

Feature detection: The Math

Consider shifting an image patch by (u,v)

- How do the pixels in the patch change?
- Compare each pixel before and after by summing up the squared differences
- This defines an "error" of E(u,v):



$$\text{Corner}(x_0, y_0) = \min_{u^2+v^2=1} E(u,v)$$

image patch

$$E(u,v) = \sum_{(x,y)\in W} [I(x+u, y+v) - I(x,y)]^2$$

at (u,v) **original patch**

$$\approx \sum_{(x,y)\in W} [\mathbf{I} + \mathbf{I}_x u + \mathbf{I}_y v - \mathbf{I}]^2$$

1st order approx

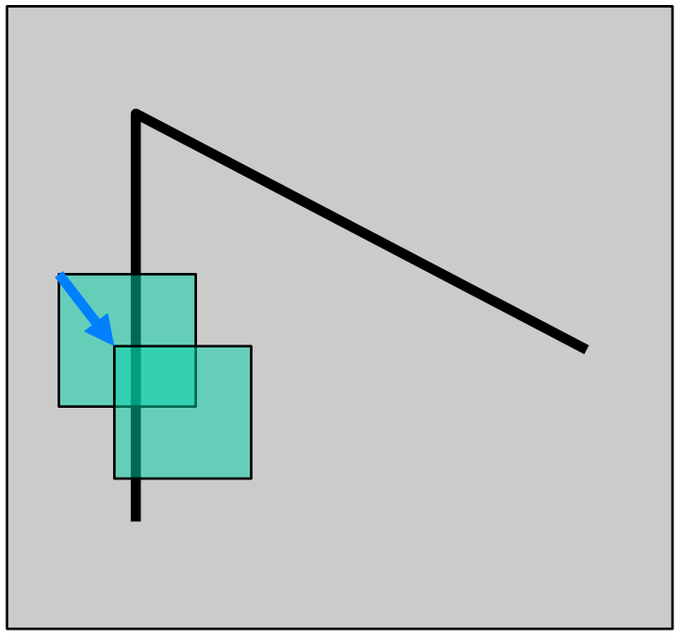
$$= \sum_{(x,y)\in W} [\mathbf{I}_x^2 u^2 + \mathbf{I}_y^2 v^2 + 2\mathbf{I}_x \mathbf{I}_y uv]$$

$$= \begin{bmatrix} u & v \end{bmatrix} A \begin{bmatrix} u \\ v \end{bmatrix}, \quad A = \sum_{(x,y)\in W} \begin{bmatrix} \mathbf{I}_x^2 & \mathbf{I}_x\mathbf{I}_y \\ \mathbf{I}_y\mathbf{I}_x & \mathbf{I}_y^2 \end{bmatrix}$$

Feature detection: The Math

Consider shifting an image patch by (u,v)

- How do the pixels in the patch change?
- Compare each pixel before and after by summing up the squared differences
- This defines an "error" of E(u,v):



$$\text{Corner}(x_0, y_0) = \min_{\text{(no constraint)}} E(u, v)$$

image patch

image patch at (u,v)

template

$$E(u,v) = \sum_{(x,y)\in W} [I(x+u, y+v) - \text{T(x,y)}]^2$$

1st order approx

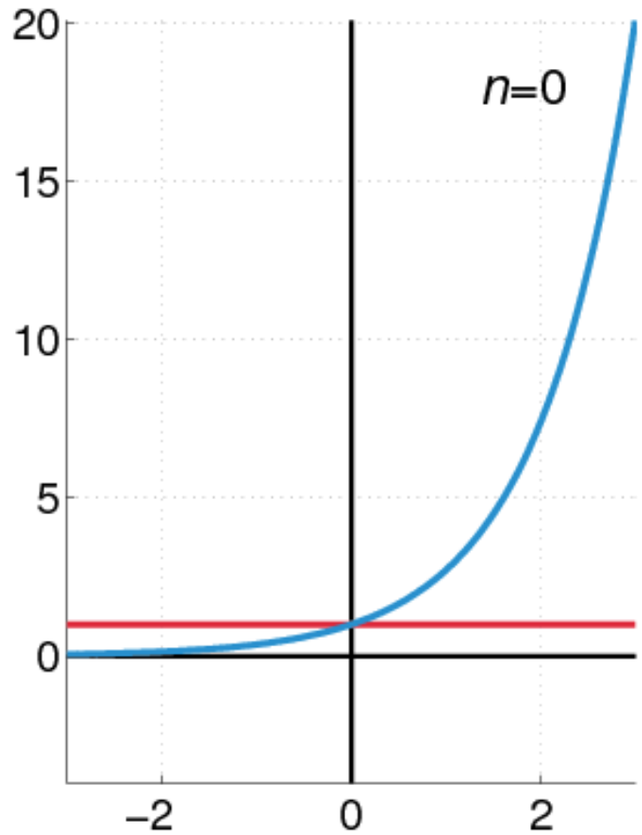
$$\approx \sum_{(x,y)\in W} [\mathbf{I} + \mathbf{I}_x u + \mathbf{I}_y v - \mathbf{I}]^2$$

$$= \sum_{(x,y)\in W} [\mathbf{I}_x^2 u^2 + \mathbf{I}_y^2 v^2 + 2\mathbf{I}_x \mathbf{I}_y uv]$$

$$= \begin{bmatrix} u & v \end{bmatrix} A \begin{bmatrix} u \\ v \end{bmatrix}, \quad A = \sum_{(x,y)\in W} \begin{bmatrix} \mathbf{I}_x^2 & \mathbf{I}_x \mathbf{I}_y \\ \mathbf{I}_y \mathbf{I}_x & \mathbf{I}_y^2 \end{bmatrix}$$

Recall: Taylor Series

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



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

Nonlinear least squares

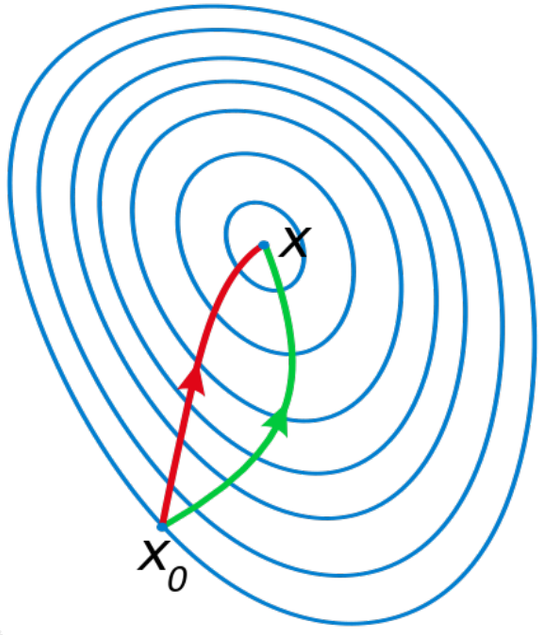
1. First order method (gradient descent)

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

2. Second order method (Newton's method)

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

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



We will do something a little different… (Gauss-Newton)

$$E(\mathbf{u}) = \frac{1}{2}||\mathbf{f}(\mathbf{u})||^2$$

function I want to minimize: outputs a vector (e.g. R, G, B errors)

u is a vector of my transformation parameters (e.g. translation, affine, etc)

How does the output of f change for small changes in u?

If f(u) outputs a single value

$$\nabla_u f(u) = \begin{bmatrix} \frac{\partial f(u)}{\partial u_1} \\ \frac{\partial f(u)}{\partial u_2} \end{bmatrix}$$

gradient vector

If f(u) outputs a vector:

$$[f_1(u) \quad f_2(u)]$$

$$J(u) = \begin{bmatrix} \frac{\partial f_1(u)}{\partial u_1} & \frac{\partial f_1(u)}{\partial u_2} \\ \frac{\partial f_2(u)}{\partial u_1} & \frac{\partial f_2(u)}{\partial u_2} \end{bmatrix}$$

Jacobian matrix

Nonlinear least squares

$$E(\mathbf{u}) = \frac{1}{2}||\mathbf{f}(\mathbf{u})||^2$$

function I want to minimize: outputs a vector (e.g. R, G, B errors)

u is a vector of my transformation parameters (e.g. translation, affine, etc)

If f(u) were a 1-D function: $E(u) = \frac{1}{2} f(u)^2$ $\quad \frac{dE(u)}{du} = f(u)\frac{df(u)}{du}$ $\quad u \leftarrow u + \alpha \frac{dE(u)}{du}$

If f(u) were a n-D function (but a 1D output):

$$E(u) = \frac{1}{2} f(u)^2 \qquad \nabla_u E(u) = f(u)\,\nabla_u f(u)$$

$$u \leftarrow u + \alpha \nabla_u E(u)$$

$$\nabla_u f(u) = \begin{bmatrix} \frac{\partial f(u)}{\partial u_1} \\ \frac{\partial f(u)}{\partial u_2} \end{bmatrix}$$

gradient vector

If f(u) were a n-D function (with an m-dimensional output):

$$E(u) = \frac{1}{2}\|f(u)\|^2 \qquad \nabla_u E(u) = J(u)^\top f(u)$$

$$u \leftarrow u + \alpha \nabla_u E(u)$$

$$J(u) = \begin{bmatrix} \frac{\partial f_1(u)}{\partial u_1} & \frac{\partial f_1(u)}{\partial u_2} \\ \frac{\partial f_2(u)}{\partial u_1} & \frac{\partial f_2(u)}{\partial u_2} \end{bmatrix}$$

Jacobian matrix

Nonlinear least squares

$$E(\mathbf{u}) = \frac{1}{2}||\mathbf{f}(\mathbf{u})||^2$$

function I want to minimize: outputs a vector (e.g. R, G, B errors)

u is a vector of my transformation parameters (e.g. translation, affine, etc)

Gauss-Newton method:

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

Instead of linearizing E(u) like on the previous slide, we will instead linearize f(u):

$$E(u + \Delta u) \approx \frac{1}{2}||\mathbf{f}(\mathbf{u}) + \mathbf{J}(\mathbf{u})\Delta\mathbf{u}||^2$$

1st order Taylor series approximation of f(u) around some initial guess *u*

Known: $f(u), J(u)$ Unknown: Δu

This approximation is now quadratic in Δu

We can now solve this for Δu with a simple least-squares!

Take gradient with respect to Δu and set equal to 0:

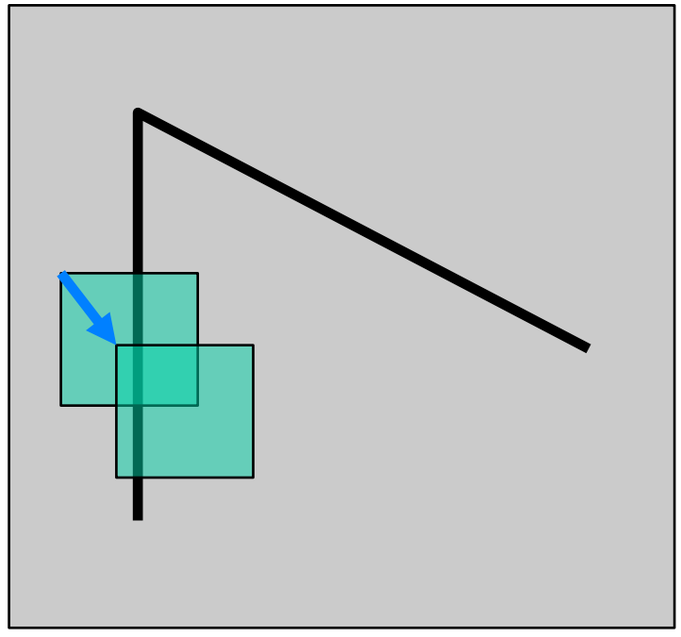
$$J(u)^\top (f(u) + J(u)\Delta u) = 0$$

$$\Delta u = -\left(J(u)^\top J(u)\right)^{-1} J(u)^\top f(u)$$

Feature detection: The Math

Consider shifting an image patch by (u,v)

- How do the pixels in the patch change?
- Compare each pixel before and after by summing up the squared differences
- This defines an "error" of E(u,v):



$$\text{Corner}(x_0, y_0) = \min_{u^2+v^2=1} E(u,v)$$

image patch

$$E(u,v) = \sum_{(x,y)\in W} [I(x+u, y+v) - I(x,y)]^2$$

at (u,v) **original patch**

$$\approx \sum_{(x,y)\in W} [\mathbf{I} + \mathbf{I}_x u + \mathbf{I}_y v - \mathbf{I}]^2$$

1st order approx

$$= \sum_{(x,y)\in W} [\mathbf{I}_x^2 u^2 + \mathbf{I}_y^2 v^2 + 2\mathbf{I}_x \mathbf{I}_y uv]$$

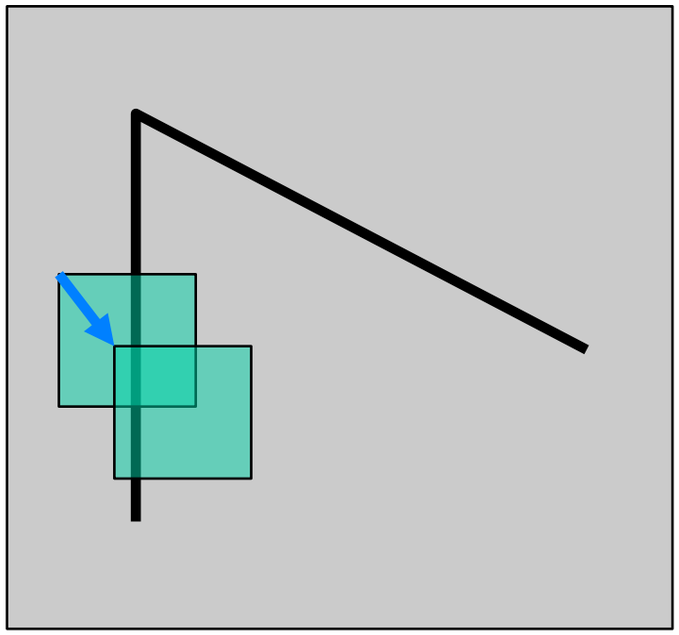
$$= \begin{bmatrix} u & v \end{bmatrix} A \begin{bmatrix} u \\ v \end{bmatrix}, \quad A = \sum_{(x,y)\in W} \begin{bmatrix} \mathbf{I}_x^2 & \mathbf{I}_x\mathbf{I}_y \\ \mathbf{I}_y\mathbf{I}_x & \mathbf{I}_y^2 \end{bmatrix}$$

Solve with SVD

Feature detection: The Math

Consider shifting an image patch by (u,v)

- How do the pixels in the patch change?
- Compare each pixel before and after by summing up the squared differences
- This defines an "error" of E(u,v):



$$\text{Corner}(x_0, y_0) = \min_{\cancel{u^2+v^2=1}} E(u,v)$$

image patch

$$E(u,v) = \sum_{(x,y)\in W} [I(x+u, y+v) - \text{T(x,y)}]^2$$

image patch at (u,v) — **template**

1st order approx

$$\approx \sum_{(x,y)\in W} [\mathbf{I} + \mathbf{I}_x u + \mathbf{I}_y v - \mathbf{I}]^2$$

$$= \sum_{(x,y)\in W} [\mathbf{I}_x^2 u^2 + \mathbf{I}_y^2 v^2 + 2\mathbf{I}_x \mathbf{I}_y uv]$$

$$= \begin{bmatrix} u & v \end{bmatrix} A \begin{bmatrix} u \\ v \end{bmatrix}, \quad A = \sum_{(x,y)\in W} \begin{bmatrix} \mathbf{I}_x^2 & \mathbf{I}_x\mathbf{I}_y \\ \mathbf{I}_y\mathbf{I}_x & \mathbf{I}_y^2 \end{bmatrix}$$

~~Solve with SVD~~

Matching a template to an image

current frame

template
(model)

u,v = hypothesized location of
template in current frame

$$\min_{u,v} E(u,v), \quad \text{where} \quad E(u,v) = \sum_{x,y} [I(x+u, y+v) - T(x,y)]^2$$

image patch at (u,v) **template**

Sum over pixels in template

Lucas Kanade alignment

image patch at (u,v) **template**

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

$$\approx \sum_{x,\,y} [I(x,y) + uI_x(x,y) + vI_y(x,y) - T(x,y)]^2$$

1st order approximation of I(x+u, y+v)

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

=I(x,y) - T(x,y)

Now Take partial derivs and set to zero (of the linearized energy function)

$$\frac{\delta E}{du} = \sum_{x,\,y} [uI_x(x,y) + vI_y(x,y) + D(x,y)]\, I_x(x,y) = 0$$

$$\frac{\delta E}{dv} = \sum_{x,\,y} [uI_x(x,y) + vI_y(x,y) + D(x,y)]\, I_y(x,y) = 0$$

Form matrix equation

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

Solve with: $x = A^{-1}b$

"Ax=b"

Lucas Kanade alignment

image patch at (u,v) **template**

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

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

"Ax=b" Solve with: $x = A^{-1}b$

But is A always invertible?

No, it is not invertible if one of the eigenvalues is 0

Thus, a template is "easy to track" of both eigenvalues are large

Does this sound familiar?

When we talked about corniness, we were looking for patches that were "easy to match"

We found a "good" solution (i.e. a corner) when both eigenvalues were large!

Lucas Kanade Alignment

Initialize the template to be aligned to some location in the image

Example: we can initialize based on the estimated position of the template in the previous frame

template

Lucas Kanade Alignment

Gauss-Newton step:

template

Solve for parameters to shift the template from the initialization [u, v]:

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

Lucas Kanade Alignment

This will result in a new guess as to the best location to align the template with the image

We can then take another Gauss-Newton step (same procedure as the previous slide)

We are now linearizing the image about a different point so we will get a different solution!

We can keep iterating

template

Is this guaranteed to converge to the right answer?

No - this is a non-linear optimization and we are just taking small update steps around the initialization

Is this guaranteed to converge at all?

Actually no - we are not performing gradient updates: $u \leftarrow u + \alpha \nabla_u f(u)$

The Gauss-Newton method is not guaranteed to converge

If we are in the *basin of attraction* (e.g. near the optimum), convergence tends to be fast (few iterations)

HW2: Tracking by iterative template alignment

Start with template from first frame and repeat:

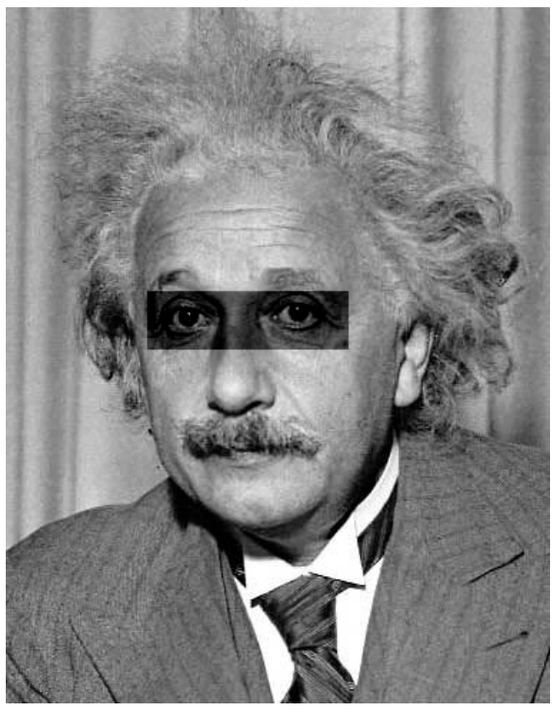
1. Align template to new frame with Lucas Kanade

2. (Optional) update template with new frame

What are some pros and cons of updating the template with the new frame?

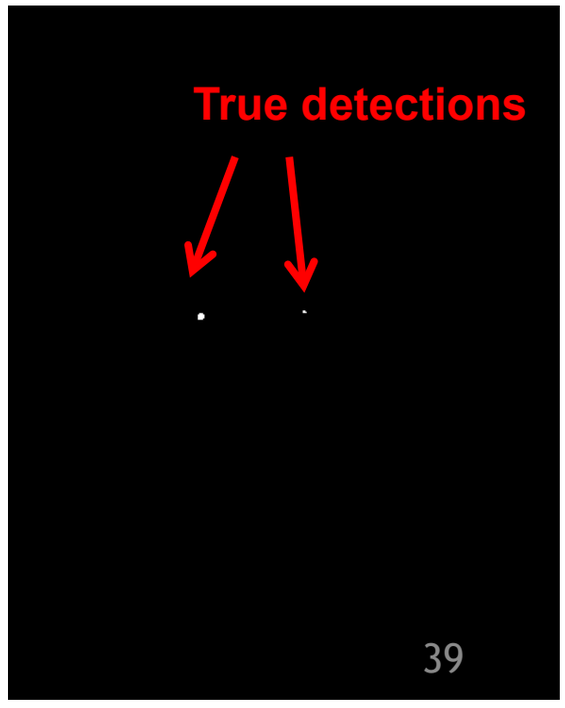
Recall: Normalized cross correlation (NCC)

$$NCC[i,j] = \frac{H^T F_{ij}}{||H||||F_{ij}||} = \cos\theta$$



NCC

NCC thresholded



Why do we need Gauss-Newton?

Why not just run normalized cross-correlation?

current frame

template
(model)

u,v = hypothesized location of
template in current frame

$$\min_{u,v} E(u,v), \quad \text{where} \quad E(u,v) = \sum_{x,y} [I(x+u, y+v) - T(x,y)]^2$$

image patch at (u,v) — template

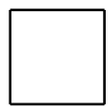
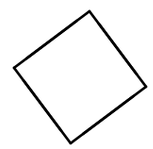
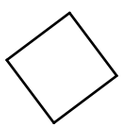
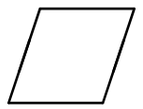
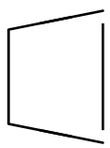
Sum over pixels in template

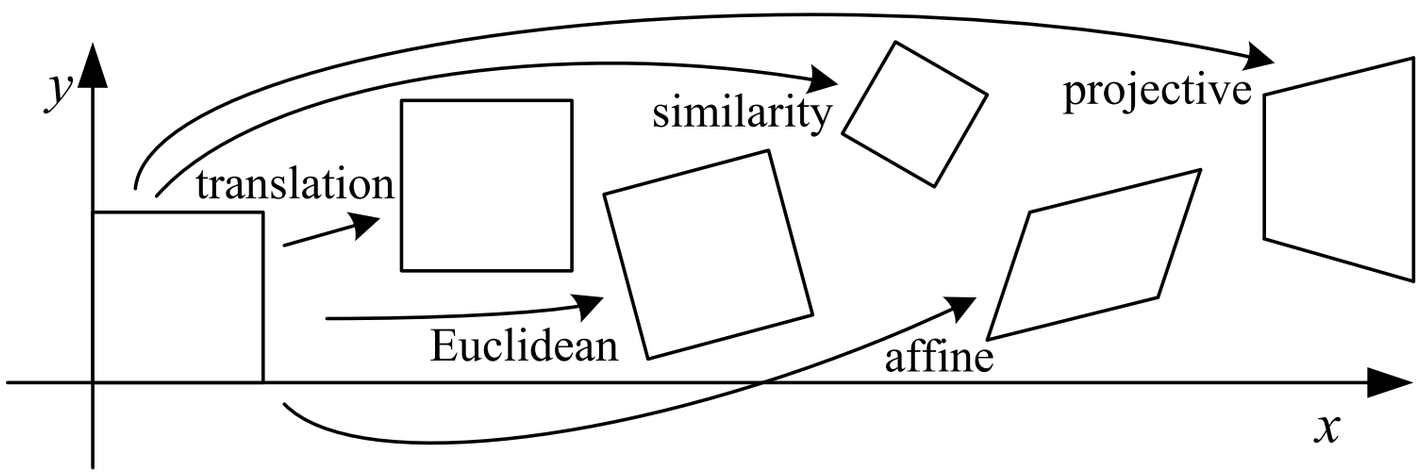
What is the transformation that is being searched over? **Translation**

For translation, NCC is sufficient

We need Gauss-Newton in case we need to search over other types of transformations (e.g. affine) - warping the template to better match the image

Family of 2D warps

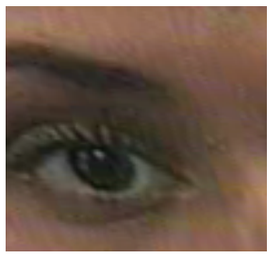
Transformation	Matrix	# DoF	Preserves	Icon
translation	$\left[\begin{array}{c\|c} \boldsymbol{I} & \boldsymbol{t} \end{array}\right]_{2\times 3}$	2	orientation	
rigid (Euclidean)	$\left[\begin{array}{c\|c} \boldsymbol{R} & \boldsymbol{t} \end{array}\right]_{2\times 3}$	3	lengths	
similarity	$\left[\begin{array}{c\|c}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

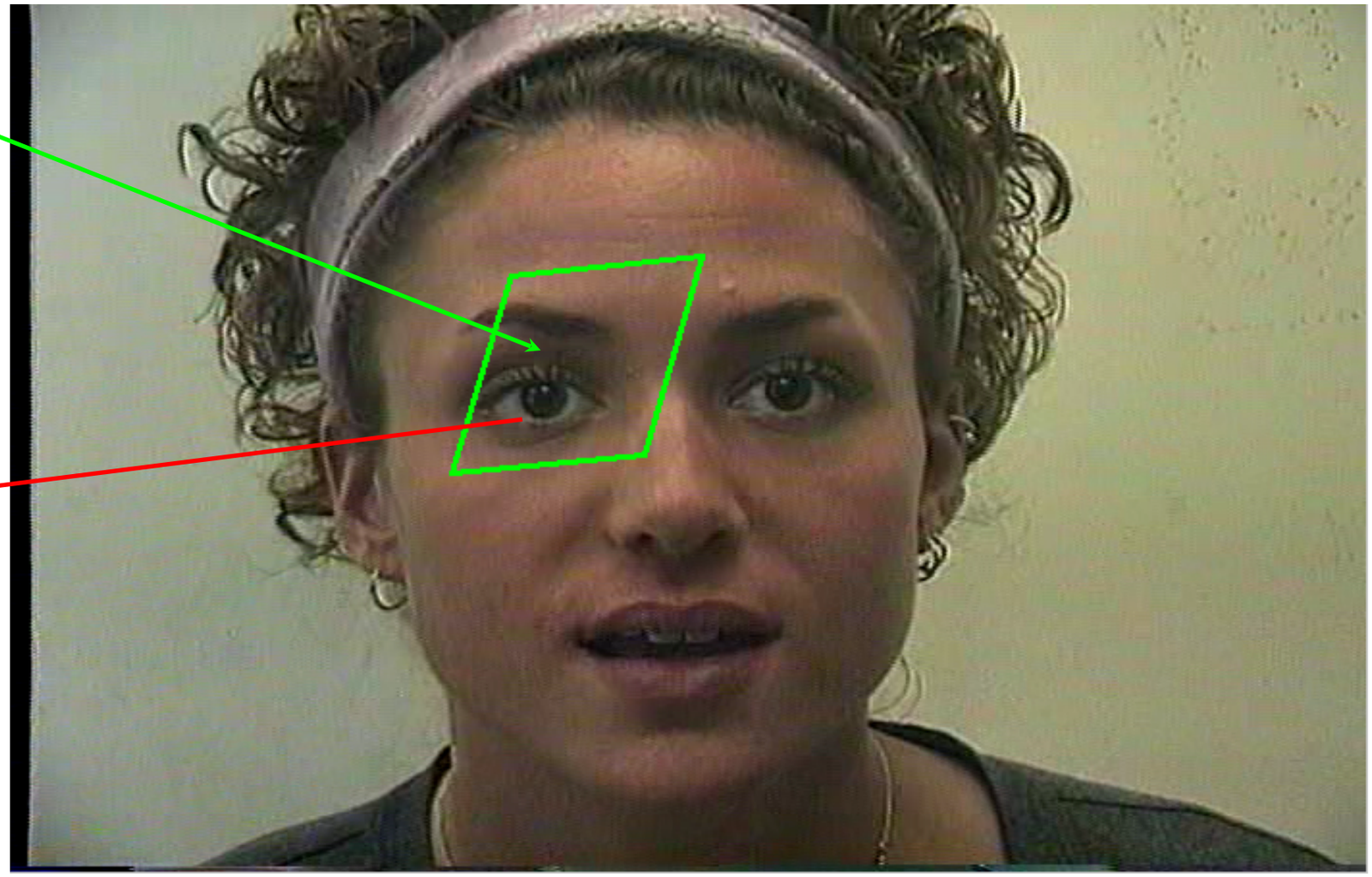


Example

Image patch

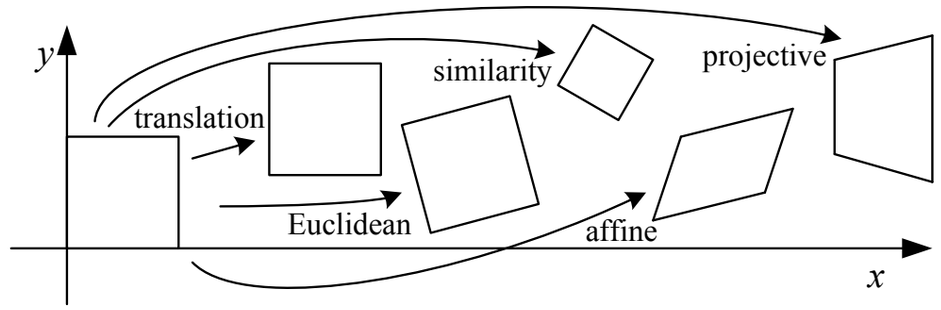
Warped patch





Replace [u,v] translation with a general warping function W(x,y; p)

image coordinates **warping parameters**



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

error function (how good of a match is this) **image coordinates** **warped image** **template**

e.g., for translation warps: $W(\mathbf{x};\mathbf{p}) = \begin{bmatrix} 1 & 0 & p_1 \\ 0 & 1 & p_2 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

e.g., for affine warps: $W(\mathbf{x};\mathbf{p}) = \begin{bmatrix} p_1 & p_2 & p_3 \\ p_4 & p_5 & p_6 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

Recall: Translation Warps

image patch at (u,v) **template**

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

$$\approx \sum_{x,\,y} [I(x,y) + uI_x(x,y) + vI_y(x,y) - T(x,y)]^2$$

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

=I(x,y) - T(x,y)

1st order approximation of the translated image I(x+u, y+v)

Now Take partial derivs and set to zero (of the approximate energy function)

$$\frac{\delta E}{du} = \sum_{x,\,y} [uI_x(x,y) + vI_y(x,y) + D(x,y)]\, I_x(x,y) = 0$$

$$\frac{\delta E}{dv} = \sum_{x,\,y} [uI_x(x,y) + vI_y(x,y) + D(x,y)]\, I_y(x,y) = 0$$

Form matrix equation

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

Solve with: $x = A^{-1}b$

"Ax=b"

Back to the big-picture

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

error function (how good of a match is this)

image coordinates

warped image

template

1st order approximation of the warped image (we will explain this step in more detail on the next slide)

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

template

current warped image

current warped image gradient

Jacobian of image coordinates w.r.t. warp parameters p

small parameter updates

Then we compute the gradient of this approximate energy function and set = 0:

$$\nabla_{\Delta p} E(\Delta p) = 0$$

Solve for Δp

1st order approximation of the warped image

1-parameter warp

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

shifted parameters

current warped image

current warped image gradient

Jacobian of image coordinates w.r.t. warp parameter p

small parameter update

multi-parameter warp (p_1, p_2)

$$I(W(\mathbf{x};\mathbf{p}+\mathbf{\Delta p})) \approx I(W(\mathbf{x};\mathbf{p})) + \left[\frac{\partial I(W(\mathbf{x};\mathbf{p}))}{\partial x} \quad \frac{\partial I(W(\mathbf{x};\mathbf{p}))}{\partial y}\right] \left[\begin{array}{cc}\frac{\partial W_x(\mathbf{x};\mathbf{p})}{\partial p_1} & \frac{\partial W_x(\mathbf{x};\mathbf{p})}{\partial p_2} \\ \frac{\partial W_y(\mathbf{x};\mathbf{p})}{\partial p_1} & \frac{\partial W_y(\mathbf{x};\mathbf{p})}{\partial p_2}\end{array}\right] \left[\begin{array}{c}\Delta p_1 \\ \Delta p_2\end{array}\right]$$

current warped image

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

current warped image gradient

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

Jacobian of image coordinates w.r.t. warp parameters p

$\Delta \mathbf{p}$

small parameter updates

Back to the big-picture

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

error function (how good of a match is this)

image coordinates

warped image

template

1st order approximation of the warped image

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

template

current warped image

current warped image gradient

Jacobian of image coordinates w.r.t. warp parameters p

small parameter updates

Then we compute the gradient of this approximate energy function and set = 0:

$$\nabla_{\Delta p} E(\Delta p) = 0$$

Solve for Δp

1st order approximation of the warped image

1-parameter warp

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

shifted parameters

current warped image

current warped image gradient

Jacobian of image coordinates w.r.t. warp parameter p

small parameter update

multi-parameter warp (p_1, p_2)

$$I(W(\mathbf{x};\mathbf{p}+\mathbf{\Delta p})) \approx I(W(\mathbf{x};\mathbf{p})) + \left[\frac{\partial I(W(\mathbf{x};\mathbf{p}))}{\partial x} \quad \frac{\partial I(W(\mathbf{x};\mathbf{p}))}{\partial y}\right] \left[\begin{array}{cc}\frac{\partial W_x(\mathbf{x};\mathbf{p})}{\partial p_1} & \frac{\partial W_x(\mathbf{x};\mathbf{p})}{\partial p_2} \\ \frac{\partial W_y(\mathbf{x};\mathbf{p})}{\partial p_1} & \frac{\partial W_y(\mathbf{x};\mathbf{p})}{\partial p_2}\end{array}\right] \left[\begin{array}{c}\Delta p_1 \\ \Delta p_2\end{array}\right]$$

current warped image

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

current warped image gradient

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

Jacobian of image coordinates w.r.t. warp parameters p

$\Delta \mathbf{p}$

small parameter updates

Let's look into how to compute this term

Example: Assume just a rotation

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

error function (how good of a match is this)

image coordinates

warped image

template

$$\mathbf{W}(\mathbf{x};p) = \begin{bmatrix} \cos(p) & -\sin(p) \\ \sin(p) & \cos(p) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$W_x = \cos(p)x - \sin(p)y$$

$$W_y = \sin(p)x + \cos(p)y$$

Now we need to re-derive the Gauss-Newton update!

Apply Taylor-series expansion to warped image **I(W(x; p)**

Make use of chain rule to compute derivate of image wrt p=θ:

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

Change of warped image w.r.t. p

Change of image w.r.t. x (x-derivative)

derivative of coordinates w.r.t. warp parameters

Change of image w.r.t. y (y-derivative)

derivative of coordinates w.r.t. warp parameters

Example: Affine warp

affine warp function (6 param

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

If I set all parameters to 0
then I get the identity transformatic

(this is better for optimization - I can optin
if I think that the identity is a good initi

Jacobian of warped image w.r.t. warp parameters p

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

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

Eigen-basis Warps

$$\mathbf{s} = \begin{bmatrix} x_1 & y_2 & x_2 & y_2 \ldots \end{bmatrix}$$

Face point coordinates

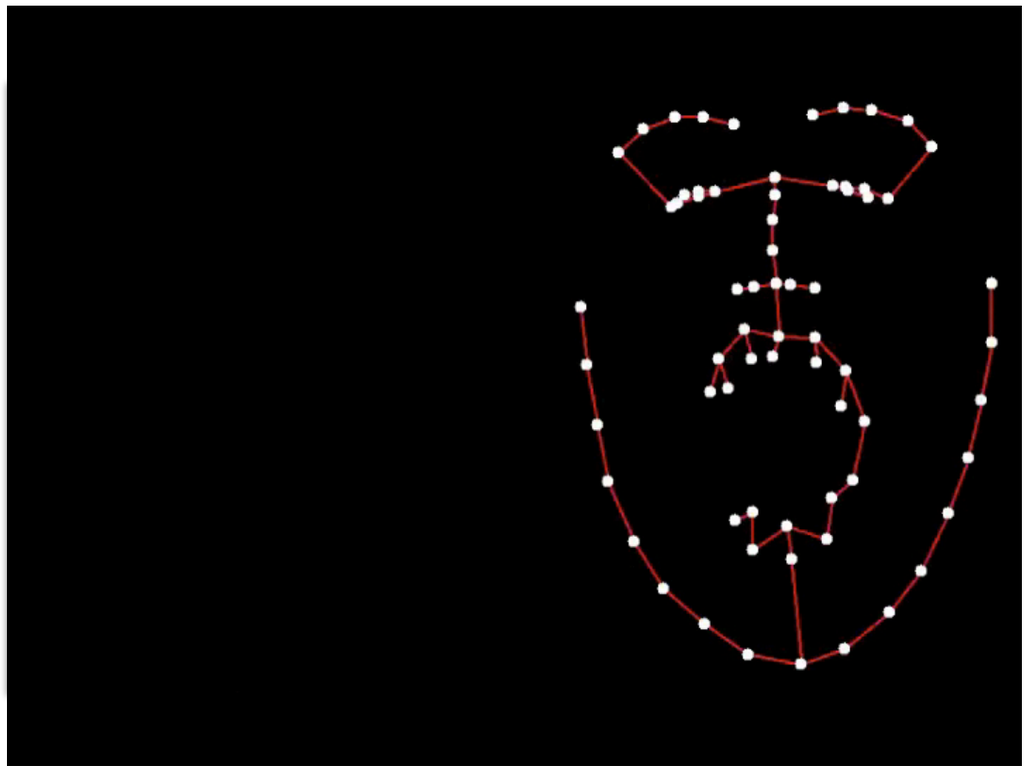
Define a new eigen-basis for image face coordinates

$$\mathbf{s} = \mathbf{s}_0 + p_1\mathbf{s}_1 + p_2\mathbf{s}_2 + \ldots$$

Pixel coordinates can be written as a combination of eigen-basis times scaling factors $p_1, p_2, \ldots$

These scaling factors are parameters of a warping function!

Visualizing a scaling of each eigenvector



We can use a Gauss-Newton update on these parameters

Back to the big-picture

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

error function (how good of a match is this)

image coordinates

warped image

template

1st order approximation of the warped image

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

template

current warped image

current warped image gradient

Jacobian of image coordinates w.r.t. warp parameters p

small parameter updates

Then we compute the gradient of this approximate energy function and set = 0:

$$\nabla_{\Delta p} E(\Delta p) = 0$$

Solve for Δp

1st order approximation of the warped image

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

template

current warped image

current warped image gradient

Jacobian of image coordinates w.r.t. warp parameters p

small parameter updates

Then we compute the gradient of this approximate energy function and set = 0:

$$\nabla_{\Delta p} E(\Delta p) = 0$$

Image Gradient

Error Image

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

Jacobian

template

currently warped image

Approx hessian

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

What happens when the error image is 0?

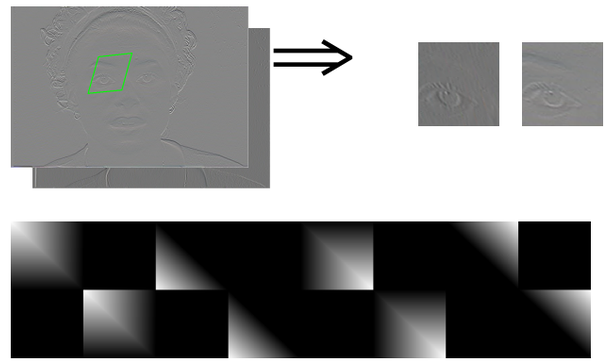
No update!

Lucas-Kanade Algorithm

1. Warp *I* with **W**(**x**;**p**) ⇒ *I*(**W**(**x**;**p**))
   curent warp warped image

2. Compute error image *T*(**x**) - *I*(**W**(**x**;**p**))
   template warped image

3. Warp gradient of *I* to compute ∇*I*



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

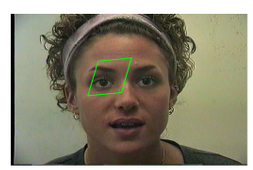
5. Compute approximate Hessian

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

6. Compute Δ**p**

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

7. Update warp parameters **p** ← **p** + Δ**p**



Extra Tricks to Speed up

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

Jacobian is always evaluated at $\mathbf{p} = 0$, which means it can be precomputed

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

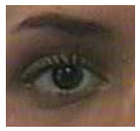
Solve for Δp to warp the template, but then instead warp the image with the inverse

Both Jacobian and Hessian are not a function of current $\mathbf{p}$ (since the warp changes the image, not the template) and so they can be precomputed

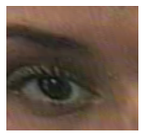
Forward and Inverse Compositional

- Forwards compositional

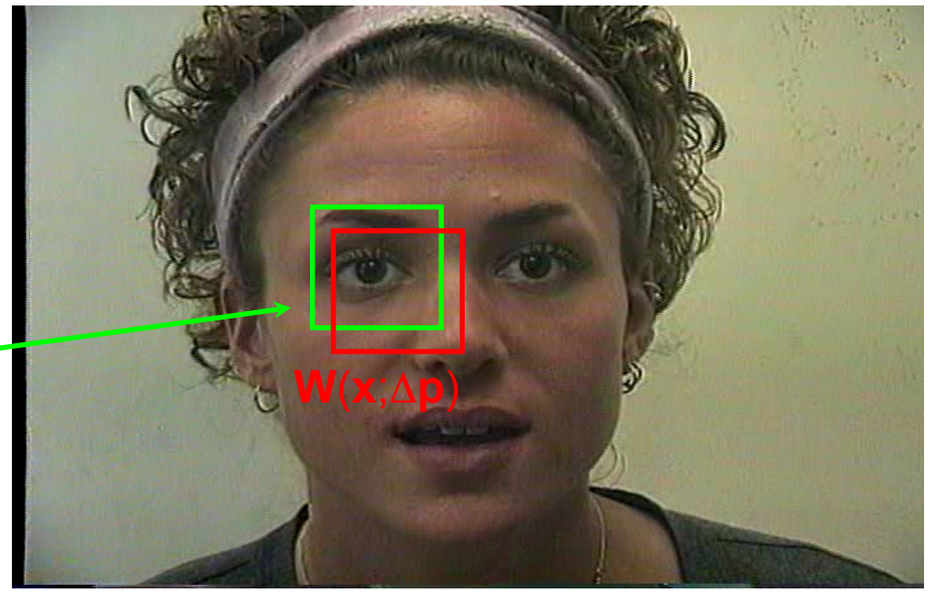
$T(\mathbf{x})$



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



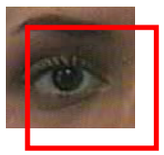
$I(\mathbf{W(x;p)})$



$I(\mathbf{x})$

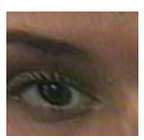
- Inverse compositional

$T(\mathbf{x})$

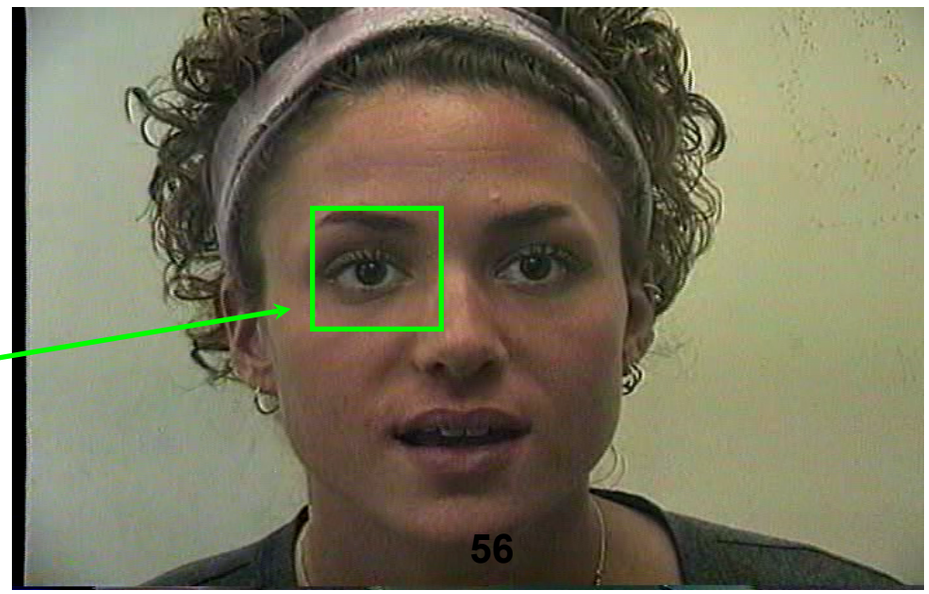


$\mathbf{W(x;\Delta p)}$

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

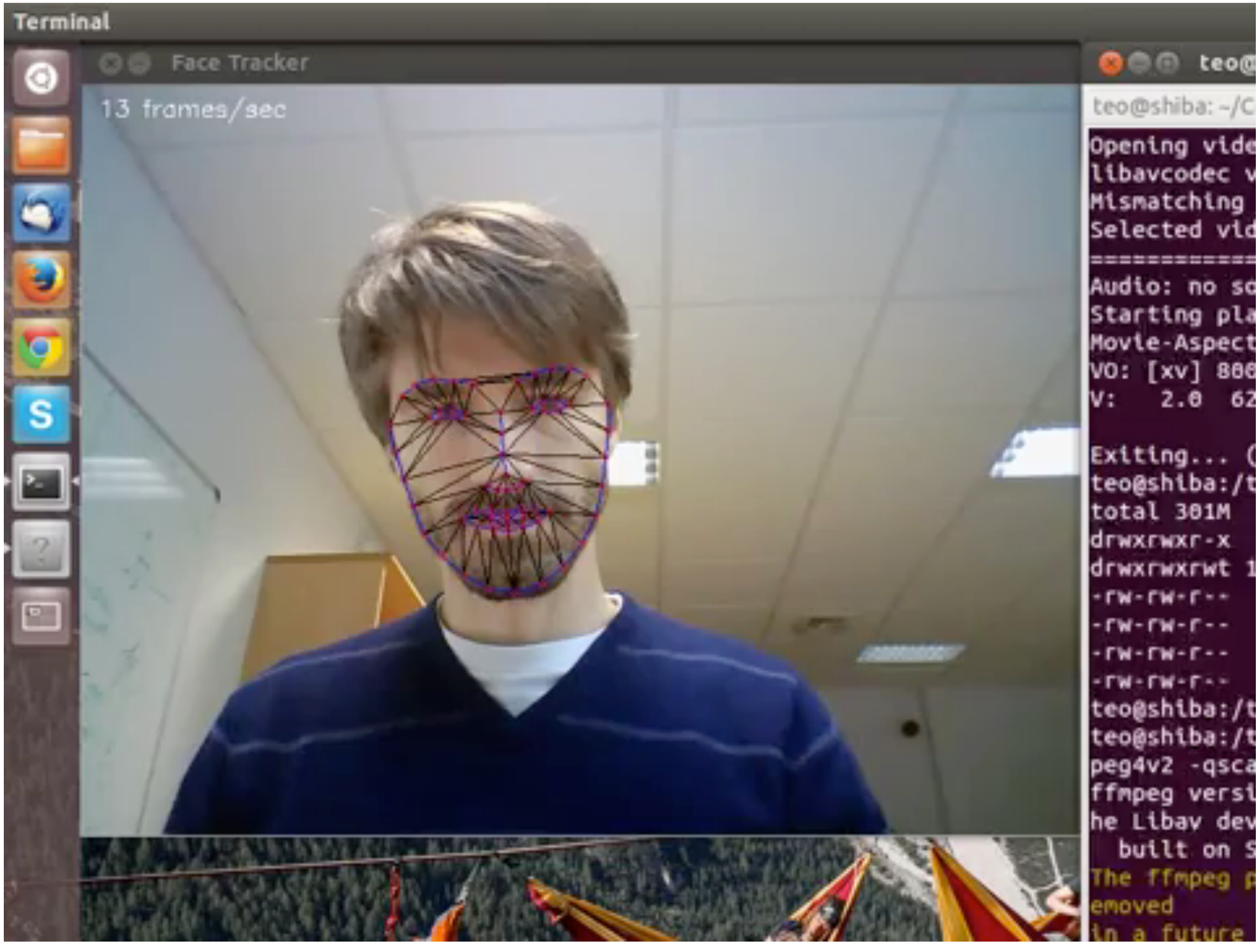


$I(\mathbf{W(x;p)})$



$I(\mathbf{x})$

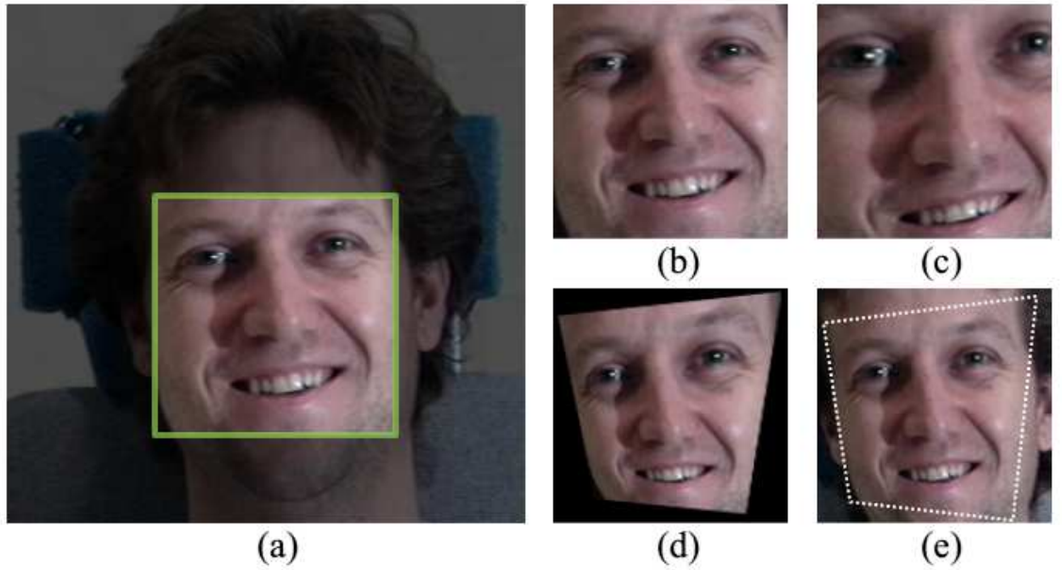
Real-time Piecewise affine-tracking

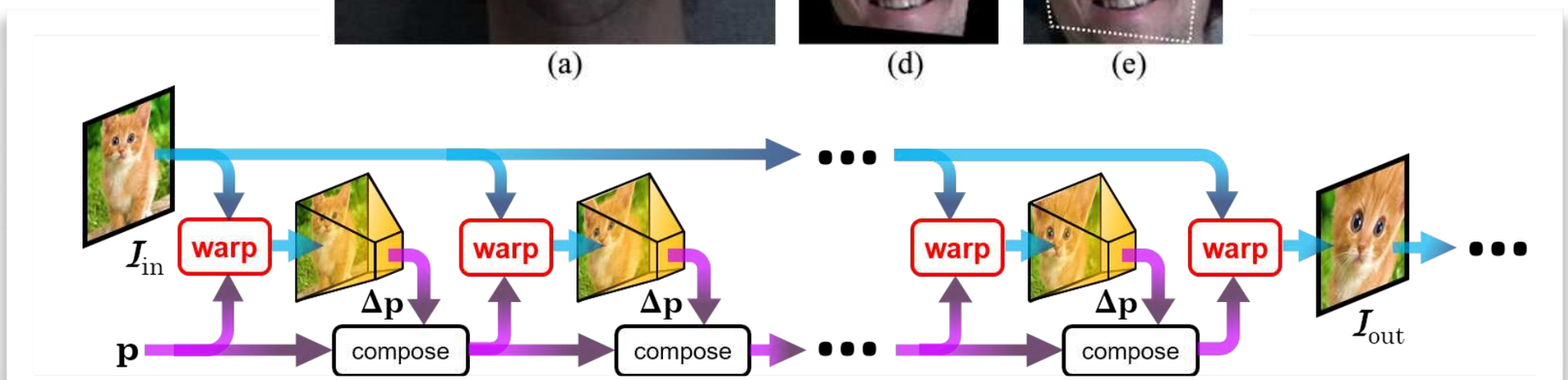


Contemporary extensions

Inverse Compositional Spatial Transformer Networks

Chen-Hsuan Lin Simon Lucey
The Robotics Institute
Carnegie Mellon University
chenhsul@andrew.cmu.edu slucey@cs.cmu.edu





Summary

- Gauss Newton optimization of nonlinear least squares

- Lucas Kanade for tracking a template