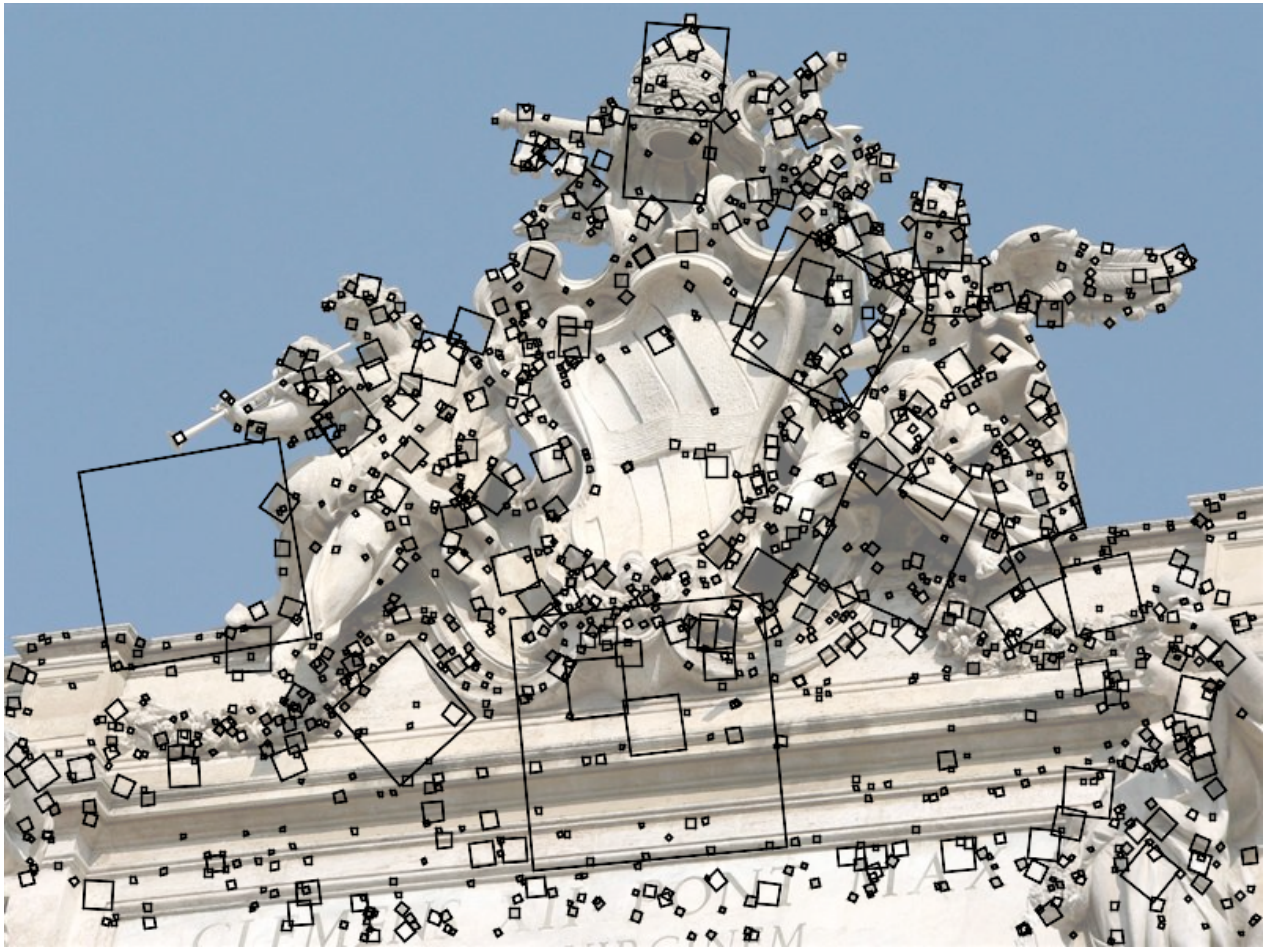
Correspondence



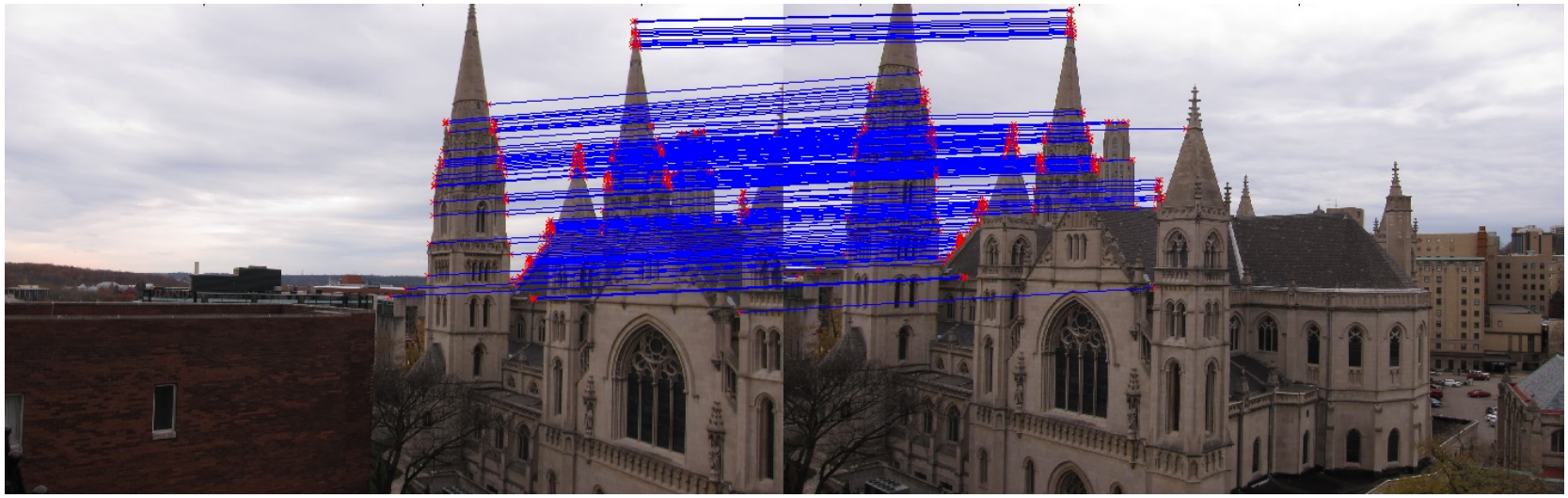
Logistics

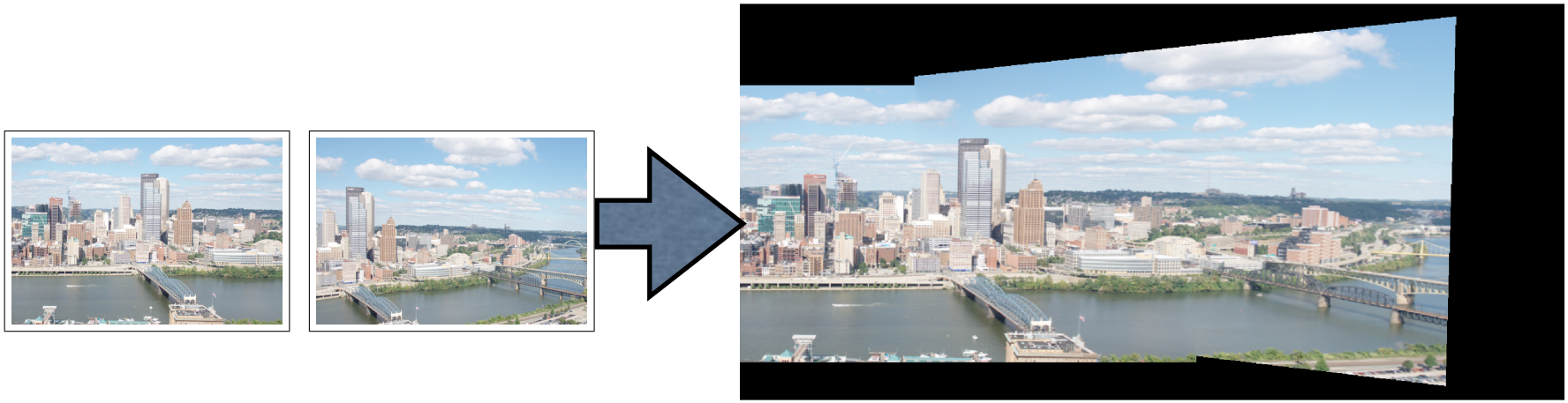
Date	Topic	References	Misc
1/14	Introduction		Linear Algebra Review
1/16	Filters		Special Recitation on HW0; NSH 1305 at 5pm
1/21	Edges		HW0 - Python Intro (not graded)
1/23	Texture		
1/28	Linear algebra review		
1/30	Warping		HW1 - HOG Image Classification
2/4	Alignment		
2/6	Frequency	Guest Lecture (tentative)	
2/11	Correspondence		
2/13	Descriptors		HW2 - LK Tracking

Outline

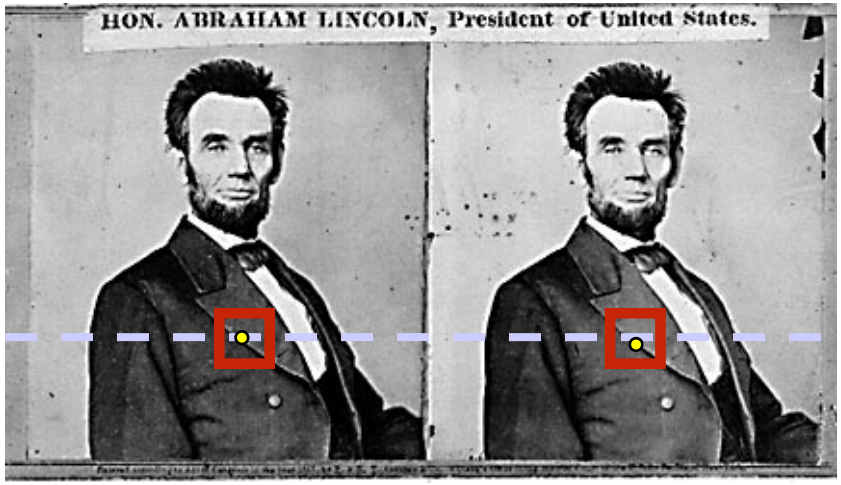
- **Motivation**
- Interest point detection
  - Cornerness
  - PSD matrices
  - Optimization (Taylor series, coarse-to-fine)
  - Scale-space
- Descriptors
  - SIFT
  - Other fast descriptors (SURF, BRIEF)
- RANSAC matching

Core visual understanding task: finding correspondences between images

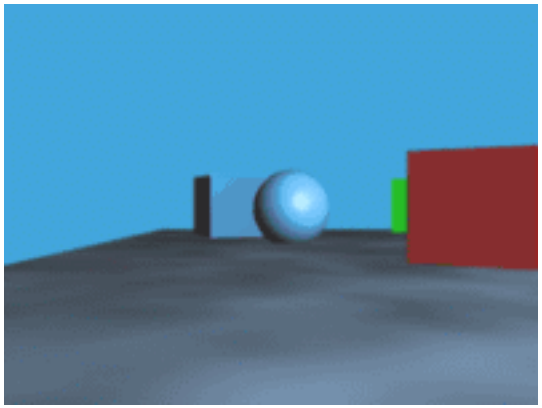




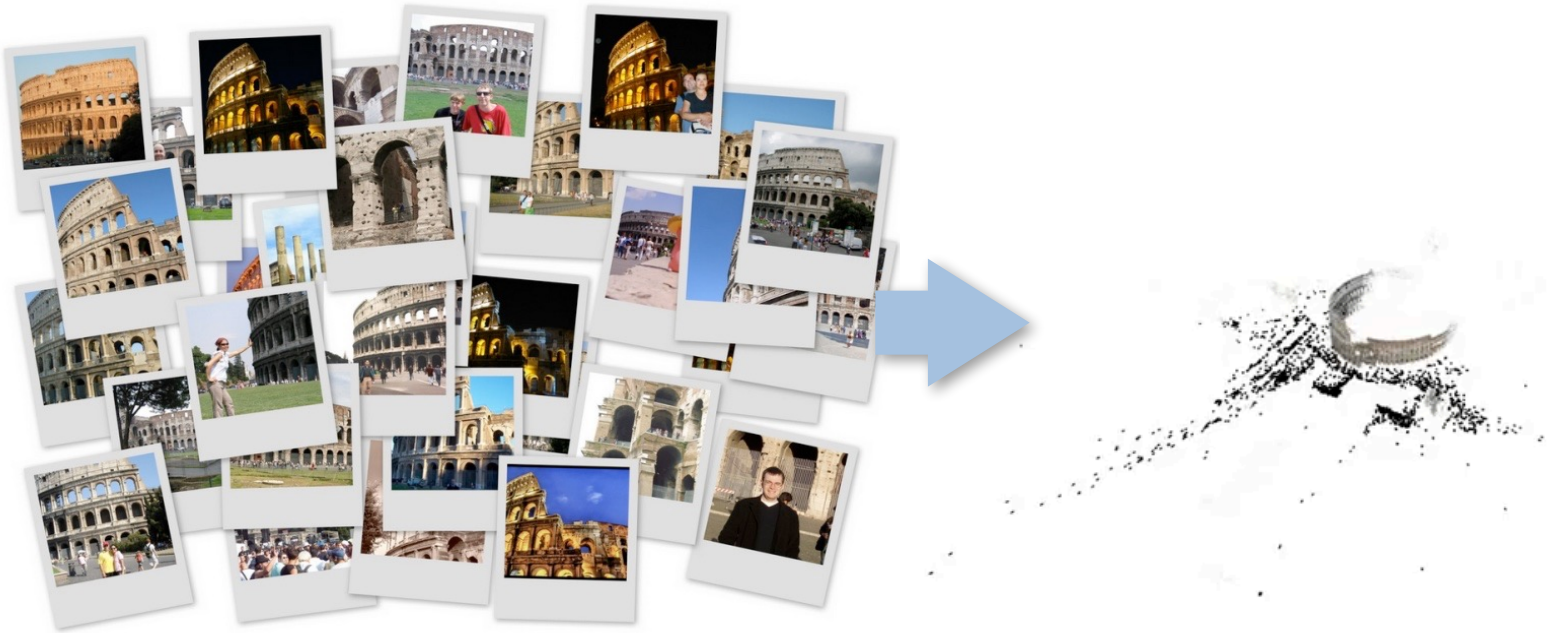
Second section of course; multiview geometry!



Finding corresponding points between pairs of views of allows us to *infer* depth in 3D; points that don't move between images are far away

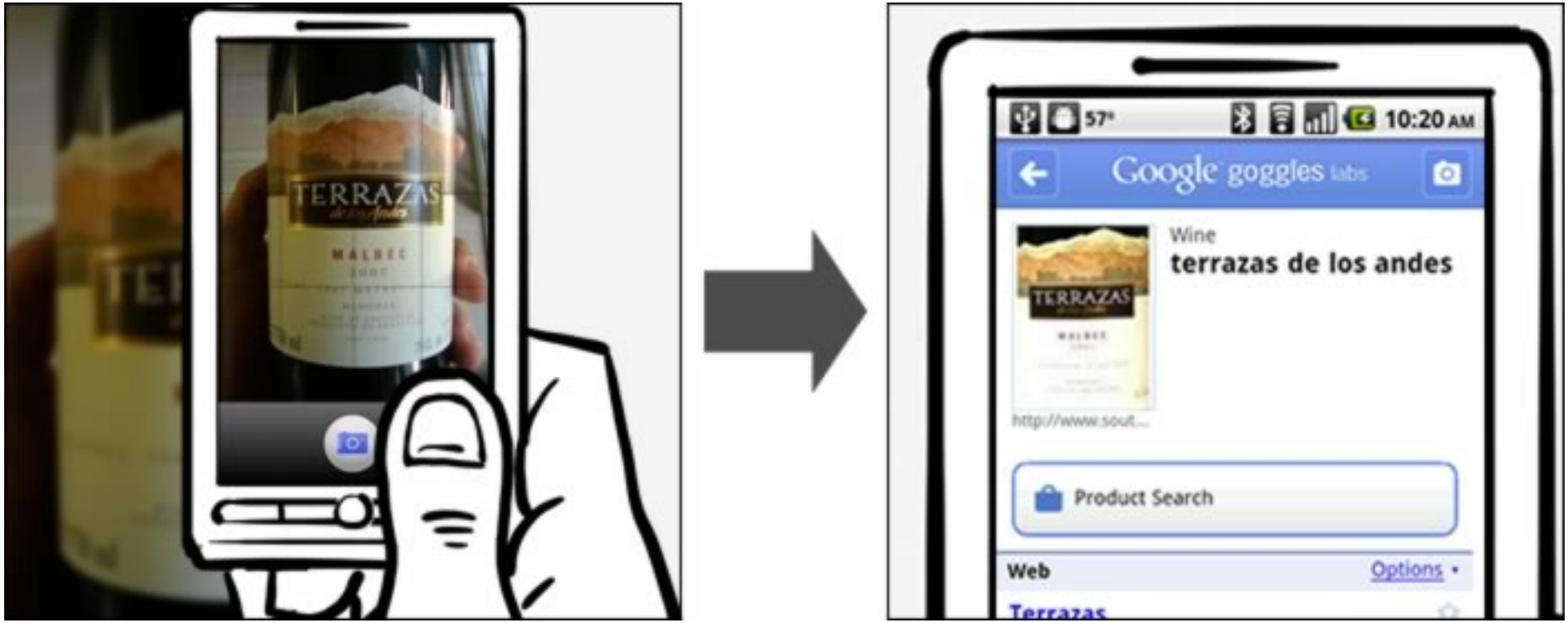


Example: image matching of landmarks



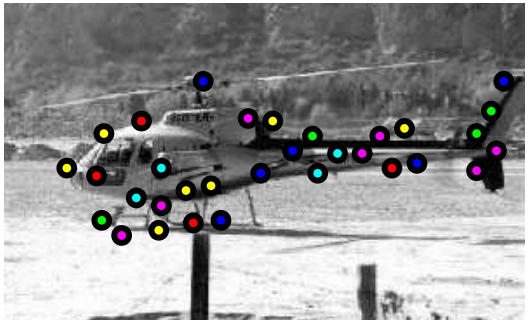
Correspondence + geometry estimation

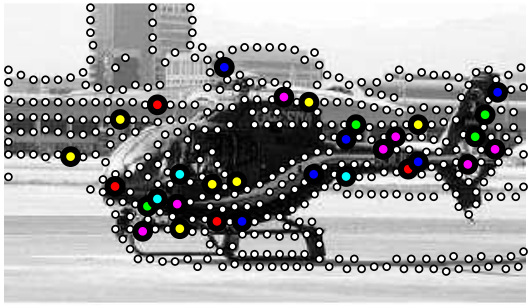
Recognition via matching to large library of training images

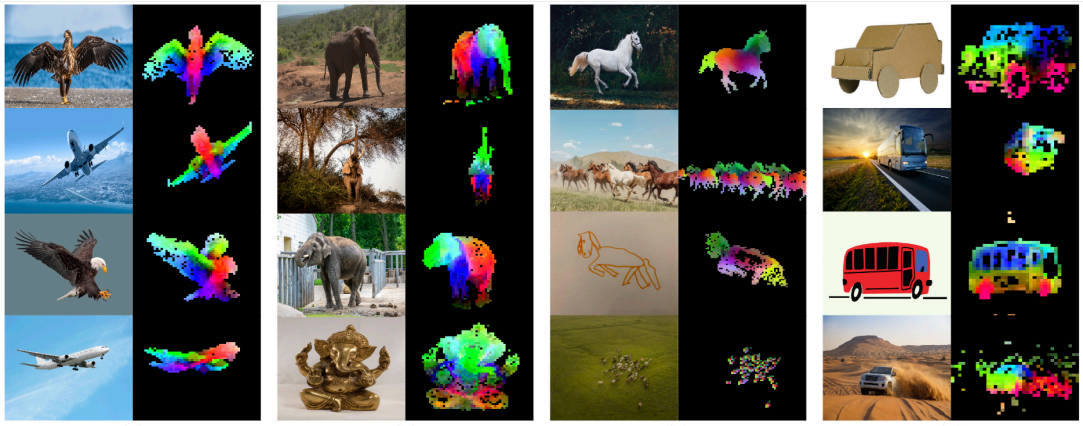


Categorical recognition by *semantic* correspondence matching

Instead of matching the same physical landmark across two images, match the same semantic keypoint (e.g., nose, tailfin-tip, …)







(a) (b) (c) (d)

Figure 1: **Visualization of the first PCA components.** We compute a PCA between the patches of the images from the same column (a, b, c and d) and show their first 3 components. Each component is matched to a different color channel. Same parts are matched between related images despite changes of pose, style or even objects. Background is removed by thresholding the first PCA component.

DinoV2
(768-dim descriptor)

Seminal references

Distinctive Image Features from Scale-Invariant Keypoints

IJCV 04

David G. Lowe
Computer Science Department
University of British Columbia
Vancouver, B.C., Canada
lowe@cs.ubc.ca

Structure-from-Motion Revisited

CVPR2016

Johannes L. Schönberger[1,2*], Jan-Michael Frahm[1]
[1]University of North Carolina at Chapel Hill
[2]Eidgenössische Technische Hochschule Zürich
jsch@inf.ethz.ch, jmf@cs.unc.edu

Contemporary codebase: https://colmap.github.io/

Still used in state-of-the-art neural methods (e.g., NERF)

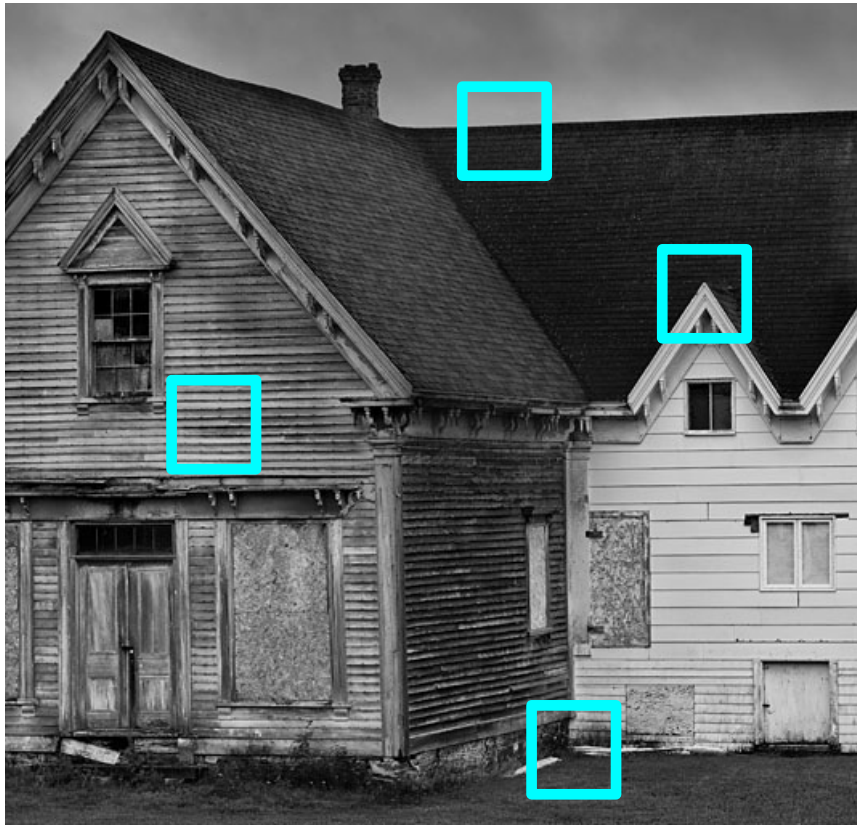
Overall pipeline for finding correspondence in 2 images

- Find N “interesting” patches in left image and M “interesting” patches in right image

- Compute O(NM) patch similarities and candidate correspondences (e.g., each patch in left image, find best match in right)

- Extract subset of correspondences that are consistent with a motion model

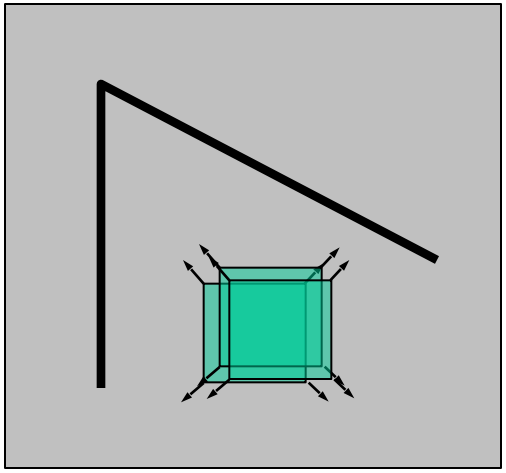
Motivation



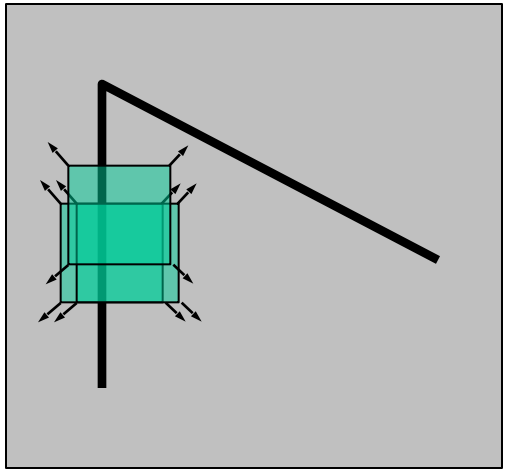
Which of these patches are easier to match?

Why? How can we mathematically operationalize this?

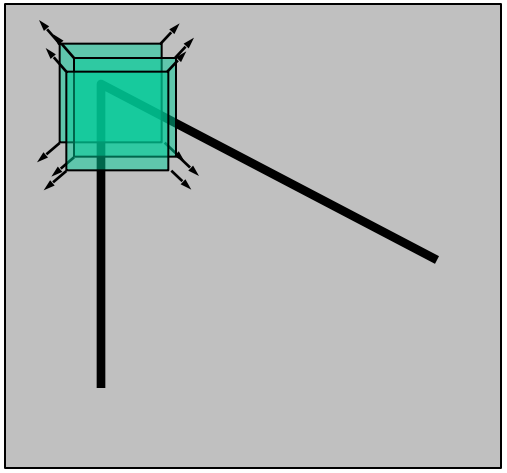
Corner Detector: Basic Idea



"flat" region:
no change in any direction



"edge":
no change along the edge direction

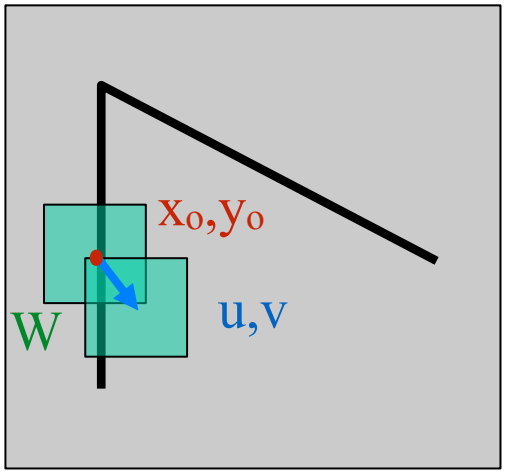


"corner":
significant change in all directions

Defn: points are "matchable" if small shifts always produce a large SSD error

The math

Defn: points are “matchable” if small shifts always produce a large SSD error



$$\text{cornerness}(x_0, y_0) = \min_{u,v} E_{x_0,y_0}(u, v)$$

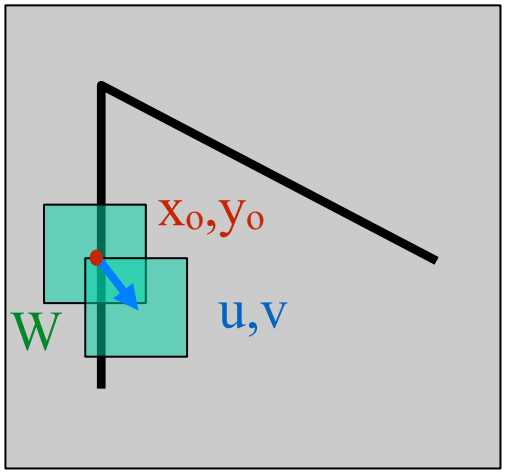
where

$$E_{x_0,y_0}(u, v) = \sum_{(x,y)\in W(x_0,y_0)} [I(x + u, y + v) - I(x, y)]^2$$

Why can’t this be right?

The math

Defn: points are “matchable” if small shifts always produce a large SSD error



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

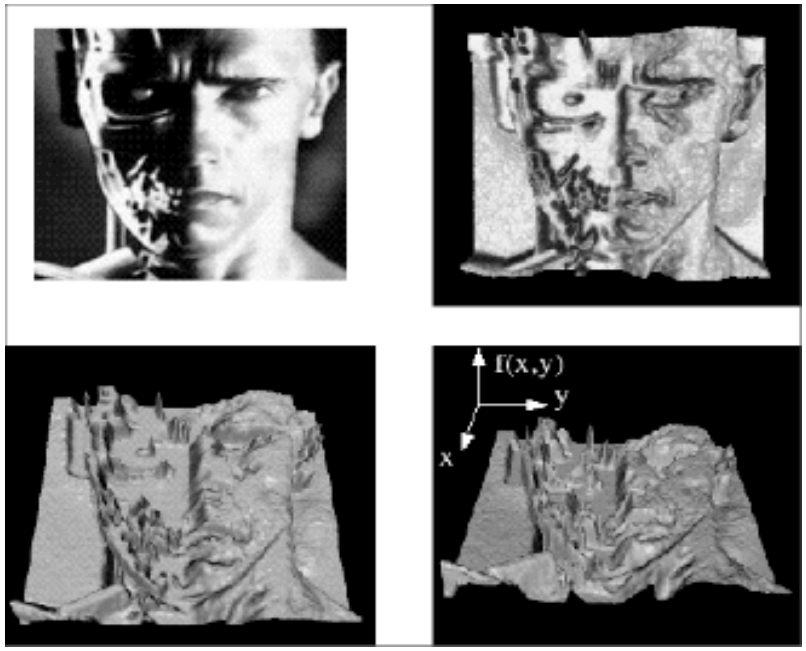
where

$$E_{x_0,y_0}(u, v) = \sum_{(x,y) \in W(x_0,y_0)} [I(x + u, y + v) - I(x, y)]^2$$

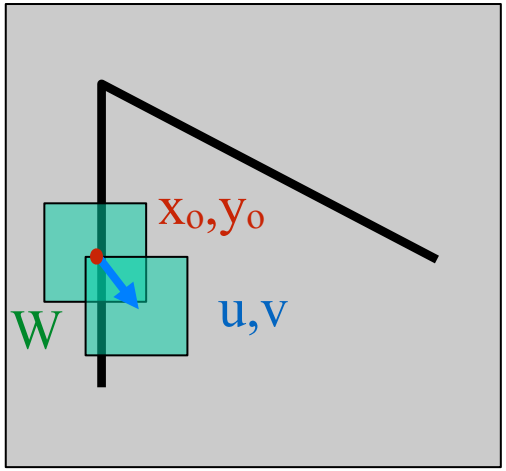
Brute-forcing this optimization for all pixel locations far too slow. How to speed up?

Approximate image locally with taylor series

$$I(x+u, y+v) = I(x,y) + \begin{bmatrix} \frac{\partial I(x,y)}{\partial x} & \frac{\partial I(x,y)}{\partial y} \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} + \text{Higher order terms}$$



Feature detection: the math



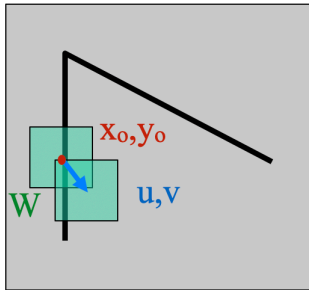
$$E(u,v) = \sum_{(x,y)\in W} [I(x+u, y+v) - I(x,y)]^2$$

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

$$= \sum_i (A[i,:]\mathbf{u})^2 = ||A\mathbf{u}||^2 = \mathbf{u}^T A^T A\mathbf{u}$$

for small u,v

Aside: comparison to old math



$$E(u,v) = \sum_{(x,y)\in W} [I(x+u, y+v) - I(x,y)]^2$$
$$= \sum_{(x,y)\in W} [I_x(x,y)u + I_y(x,y)v]^2$$
$$= \sum (A[i,:]\mathbf{u})^2 = ||A\mathbf{u}||^2 = \mathbf{u}^T A^T A\mathbf{u}$$

Corner detection

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

(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?

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

Lucas Kanade Alignment

Equivalent to the A matrix that we'd compute for a *translation* warp in LK alignment

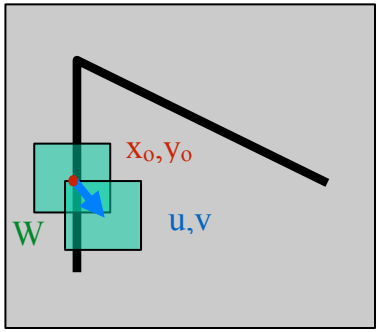
Translation

$$\begin{bmatrix} W_x(\mathbf{x};\mathbf{p}) \\ W_y(\mathbf{x};\mathbf{p}) \end{bmatrix} = \begin{bmatrix} x + p_1 \\ y + p_2 \end{bmatrix}$$
$$= \begin{bmatrix} 1 & 0 & p_1 \\ 0 & 1 & p_2 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$
$$I(\mathbf{W}(\mathbf{x};\mathbf{p})) = I(x + p_1, y + p_2)$$

Intuitively, patch at (x0,y0) is a corner if there's a *unique* solution to one iteration of LK-alignment

When is there a unique soln?

If small patch shifts always produce a large SSD error



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

where

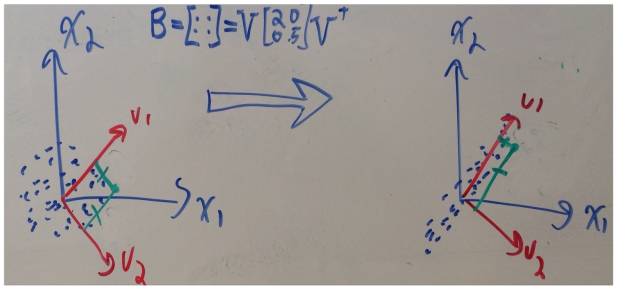
$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} B \begin{bmatrix} u \\ v \end{bmatrix}, \quad B = \sum_{(x,y)\in W(x_0,y_0)} \begin{bmatrix} I_x(x,y)I_x(x,y) & I_x(x,y)I_y(x,y) \\ I_y(x,y)I_x(x,y) & I_y(x,y)I_y(x,y) \end{bmatrix}$$

Claim 1: 'B' is symmetric ($B^T = B$) and PSD

Proof. $B = A^TA$ for A from previous slide

Claim 2: Corner-ness is given by min eigenvalue of 'B'

Review of positive semi-definite (PSD) matrices



Consider the *norm* of an output point after a linear transformation A

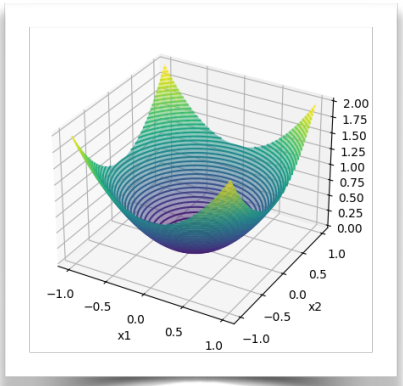
$$||\mathbf{y}||^2 = \mathbf{y}^T\mathbf{y} = \mathbf{x}^T A^T A\mathbf{x} = \mathbf{x}^T B\mathbf{x}$$

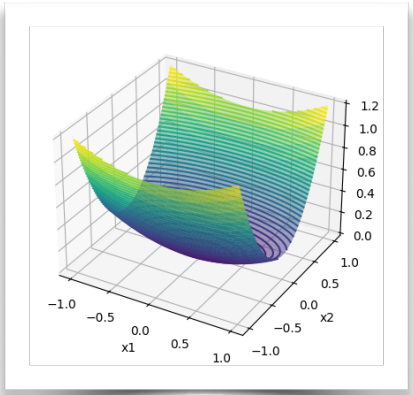
$$f(x_1, x_2) = [x_1 \quad x_2] \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = B_{11}x_1^2 + B_{12}x_1x_2 + B_{21}x_2x_1 + B_{22}x_2^2$$

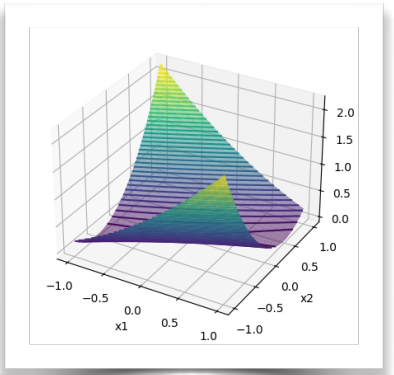
$$= B_{11}x_1^2 + (B_{12} + B_{21})x_2x_1 + B_{22}x_2^2$$

Aside: we need only care about quadratic forms of *symmetrized* matrix B' = (B+B^T)/2

Such *quadratic forms* generalize linear functions of (x_1,x_2) to quadratic functions of (x_1,x_2)







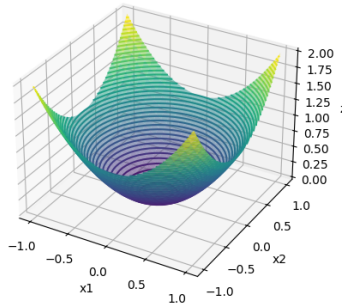
Visualizing PSD matrices as quadratic forms

$$f(x_1, x_2) = [x_1 \quad x_2]\, B \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

where B is

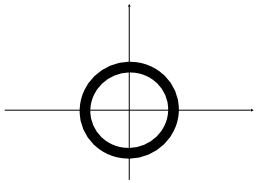
$$B = A^T A = V \Sigma^2 V^T = \begin{vmatrix} | & | \\ \mathbf{v}_1 & \mathbf{v}_2 \\ | & | \end{vmatrix} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} -\mathbf{v}_1^T - \\ -\mathbf{v}_2^T - \end{bmatrix}$$

$$f(x_1, x_2) = [x_1 \quad x_2] \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = x_1^2 + x_2^2$$

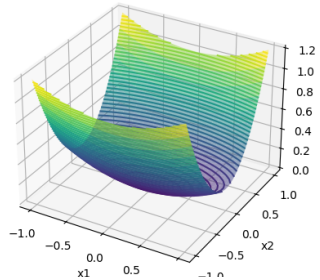


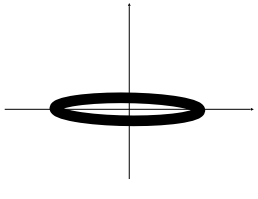
2D visual:

$f(x_1, x_2) = 1$

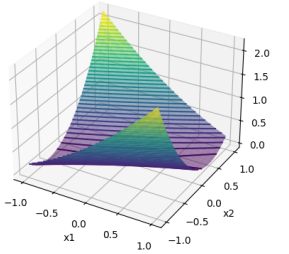


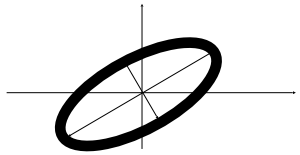
$$f(x_1, x_2) = [x_1 \quad x_2] \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \lambda_1 x_1^2 + \lambda_2 x_2^2$$





$$f(x_1, x_2) = [x_1 \quad x_2] \begin{vmatrix} | & | \\ \mathbf{v}_1 & \mathbf{v}_2 \\ | & | \end{vmatrix} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} -\mathbf{v}_1^T - \\ -\mathbf{v}_2^T - \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

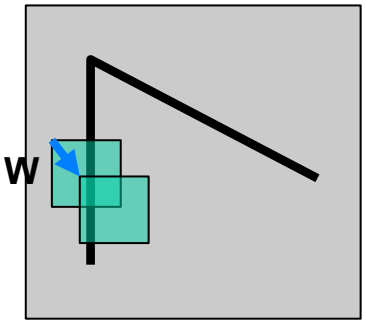




$$= \lambda_1 \hat{x}_1^2 + \lambda_2 \hat{x}_2^2, \quad \text{where} \quad \begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \end{bmatrix} = \begin{bmatrix} -\mathbf{v}_1^T - \\ -\mathbf{v}_2^T - \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Back to corner(ness)

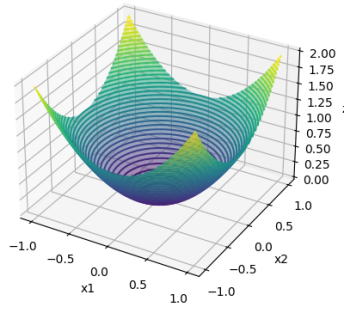
Defn: points are “matchable” if small shifts always produce a large SSD error



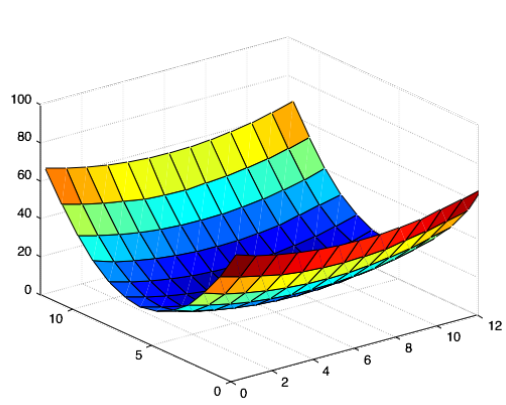
$$\text{Corner}(x_0, y_0) = \min_{u^2+v^2=1} E(u, v)$$

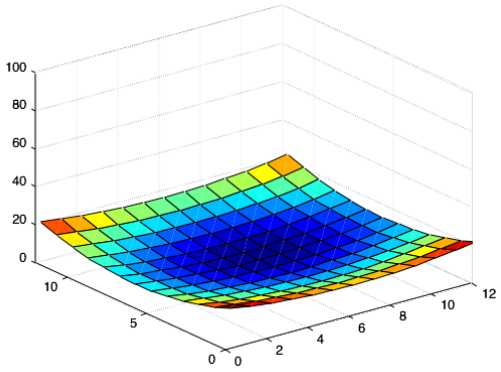
where

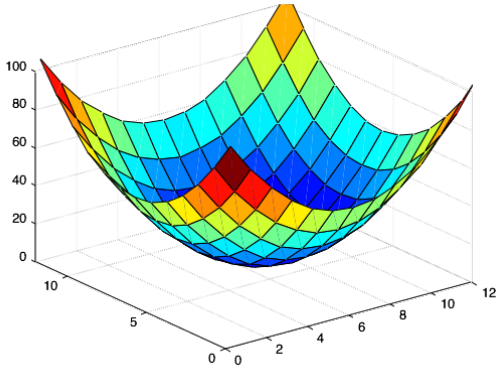
$$E(u, v) \approx \begin{bmatrix} u & v \end{bmatrix} B \begin{bmatrix} u \\ v \end{bmatrix}, \quad B = \sum_{(x,y)\in W(x_0,y_0)} \begin{bmatrix} I_x(x, y)I_x(x, y) & I_x(x, y)I_y(x, y) \\ I_y(x, y)I_x(x, y) & I_y(x, y)I_y(x, y) \end{bmatrix}$$



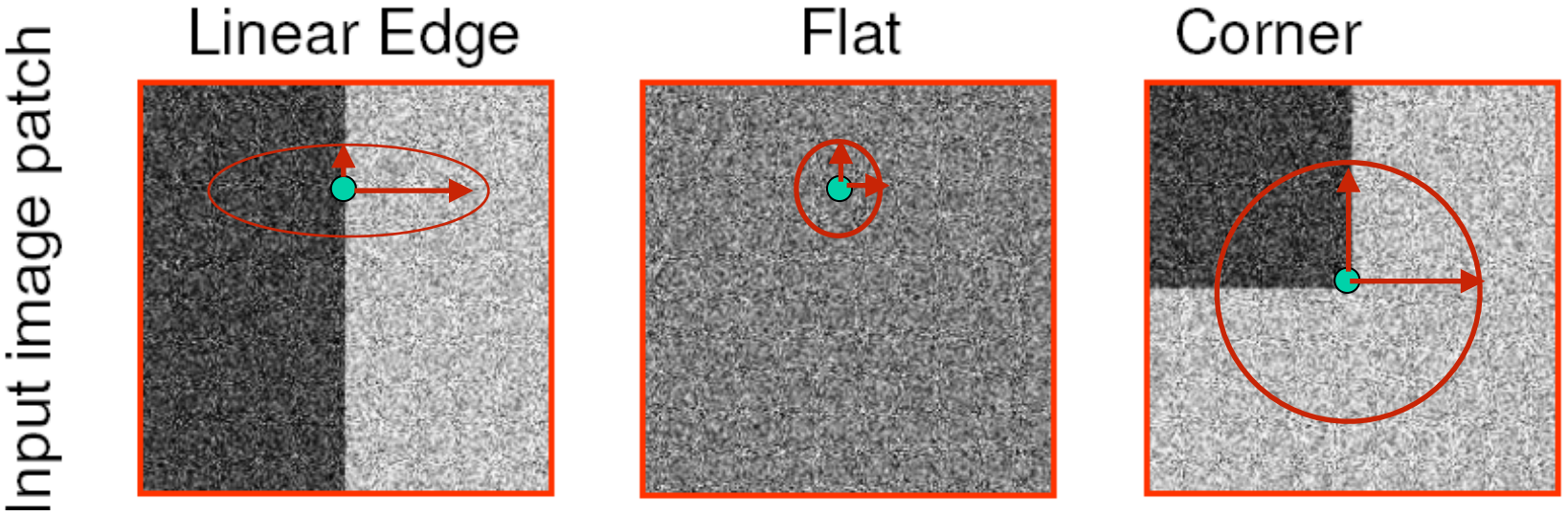
Which error surface indicates a good image feature?







What kind of image patch do these surfaces represent?



Efficient computation

Computing eigenvalues (and eigenvectors) is expensive

Recall that it's easy to compute their sum (trace) and product (determinant)

Notation switch: subsequent slides will refer to 2x2 PSD matrix as A (not B)

- $\text{Det}(A) = \lambda_1\lambda_2$ ($\det = A_{11}A_{22} - A_{12}A_{21}$)
- $\text{Trace}(A) = \lambda_1+\lambda_2$ ($\text{trace} = A_{11} + A_{22}$)

Define as easy-to-compute value that is large when both λ_1 and λ_2 are large

$$R = \lambda_1\lambda_2 - .5(\lambda_1 + \lambda_2) = \text{Det}(A) - \alpha\text{Trace}(A)$$

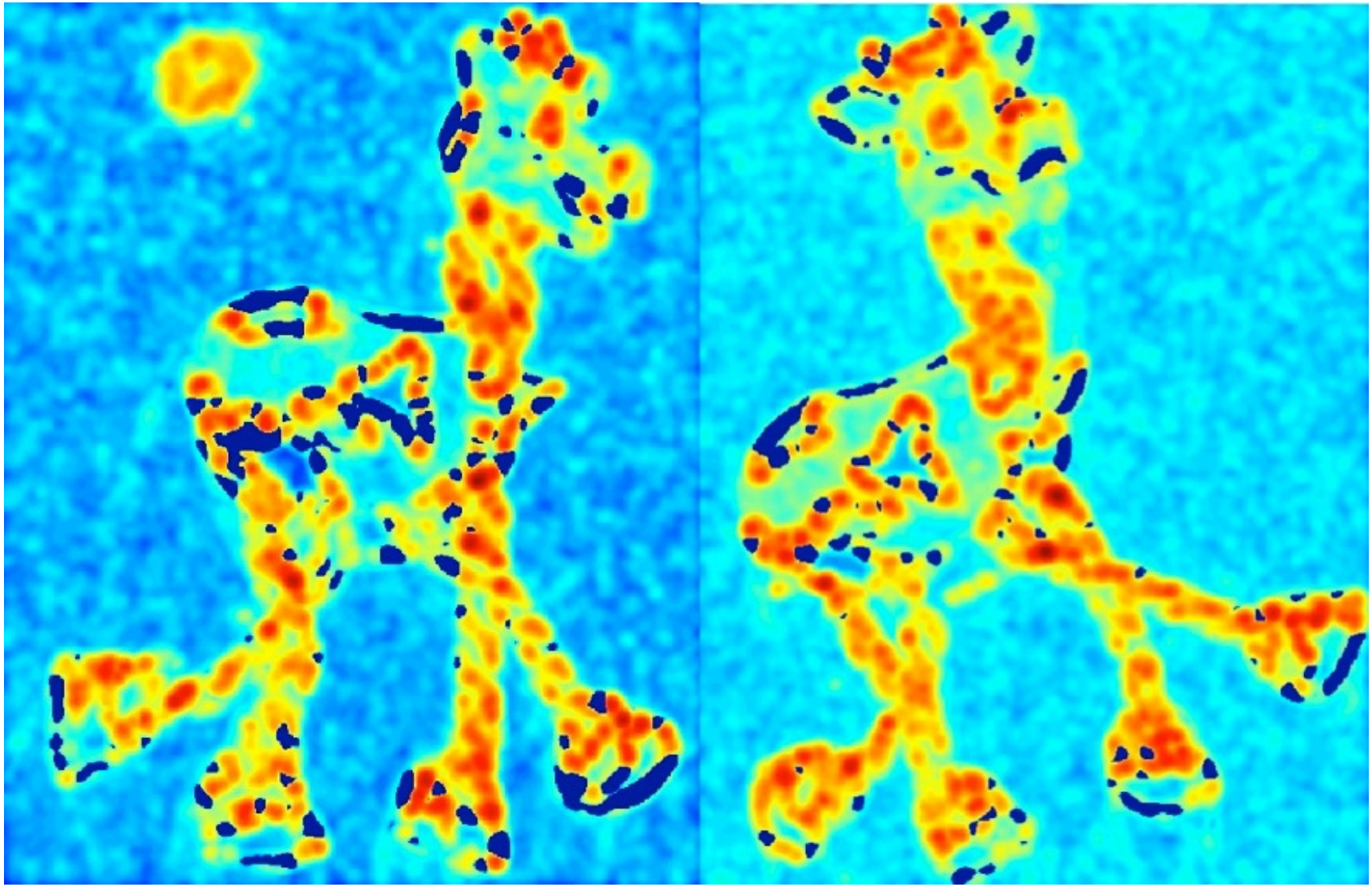
https://en.wikipedia.org/wiki/Harris_corner_detector

Harris detector example

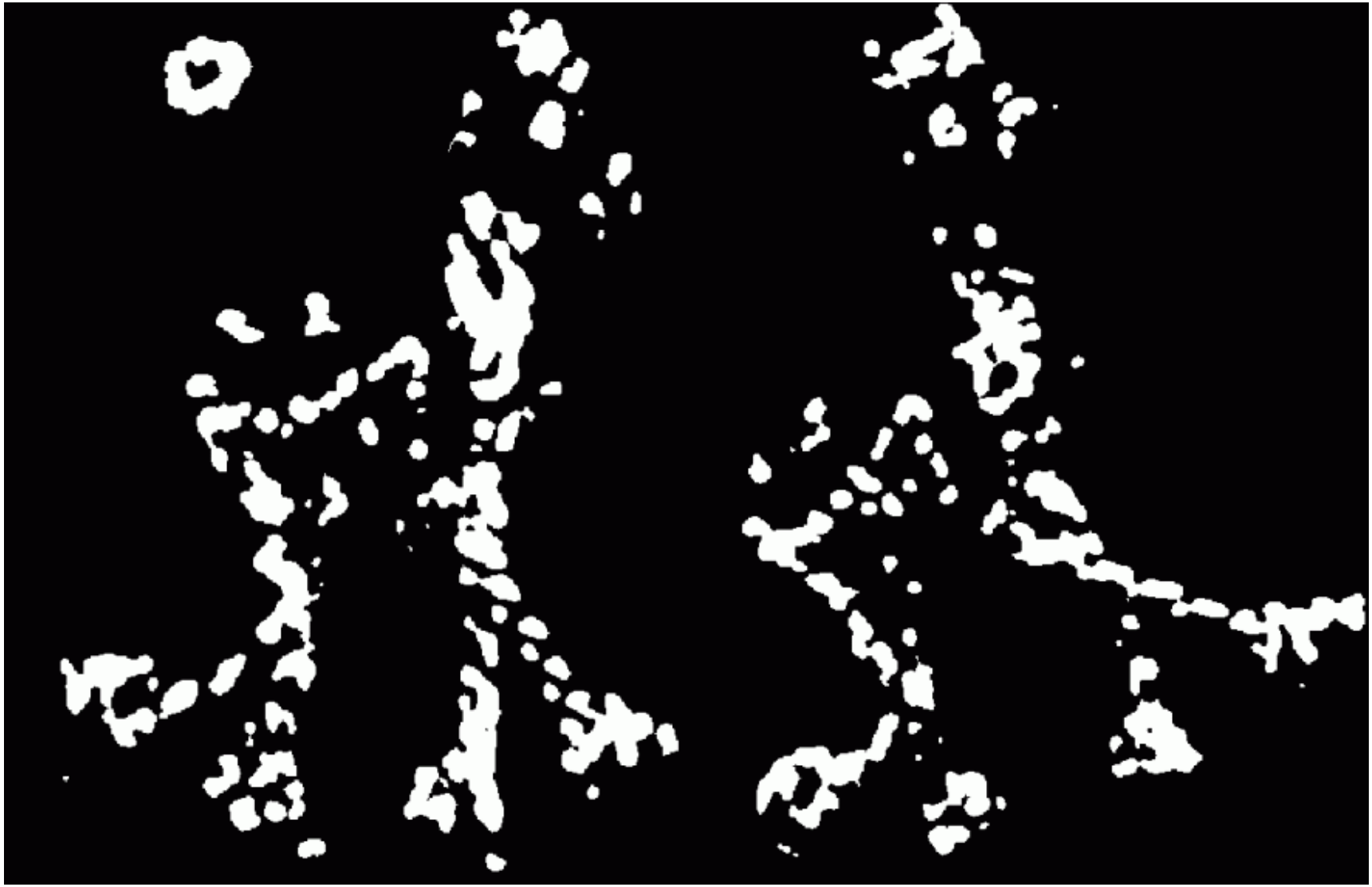
corner value (red high, blue low)

Question: can we compute these heat maps with convolutions?

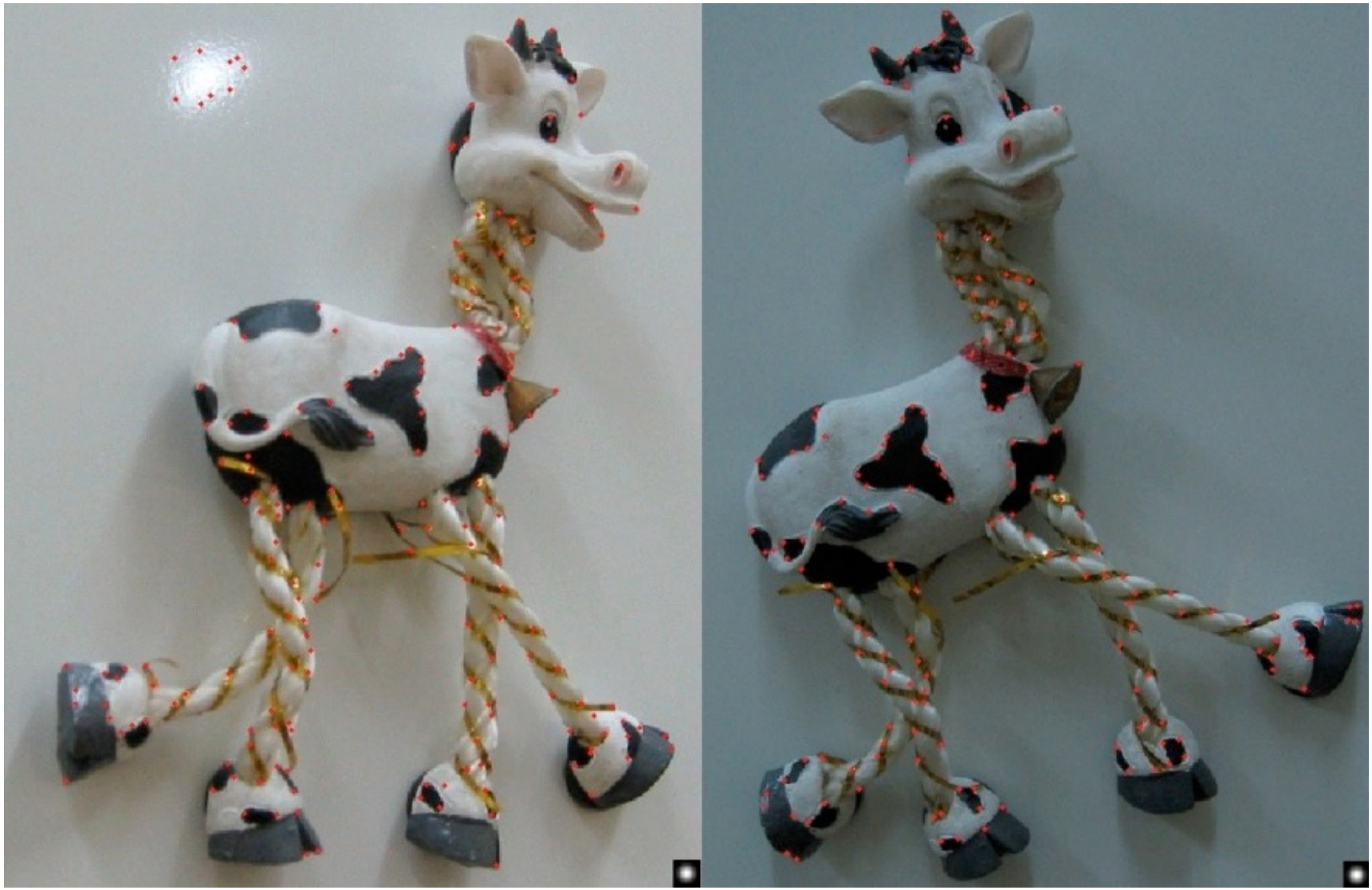
yes; apply box filter to gradient images



Threshold (f > value)



Harris features (in red)

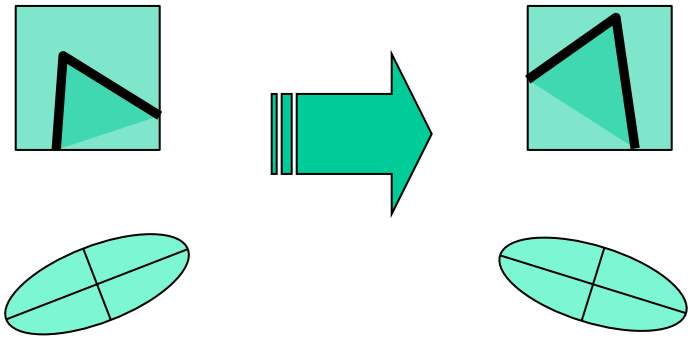


The tops of the horns are detected in both images

Scale and rotation invariance

Will interest point detector still fire on rotated & scaled images?

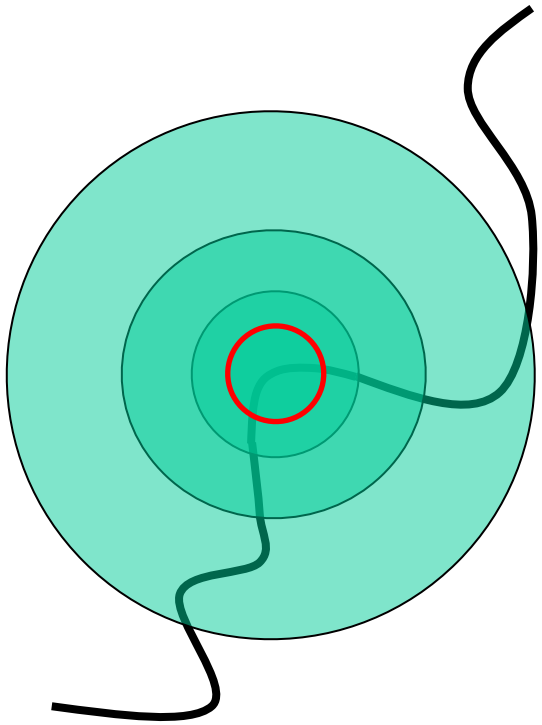
Rotation invariance (?)



Are eigenvalues stable under rotations? Yes

Implies Harris detector will fire on same regions on rotated image

Scale invariance?

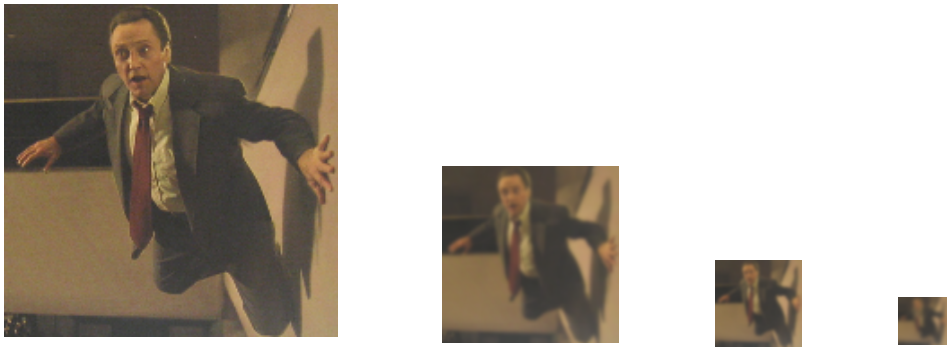


Are eigenvalues stable under scalings? No

Implies Harris detector won't fire on same regions on scaled image

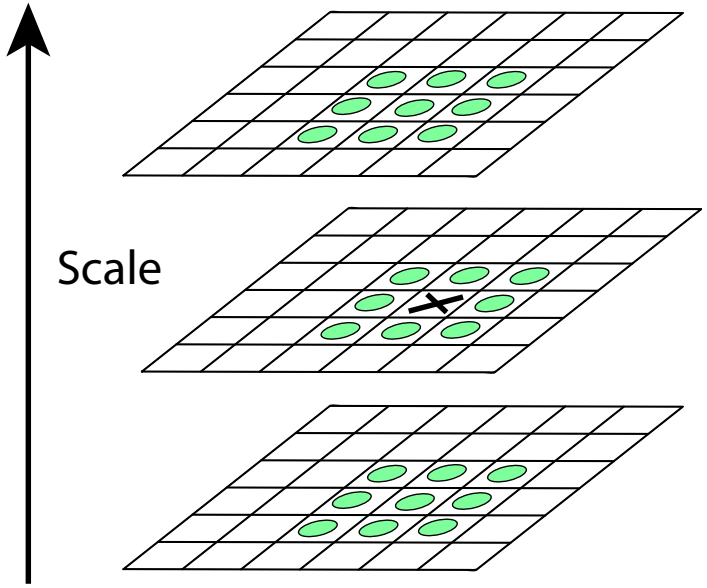
A solution to scale

search over image pyramid scales



$$A(x_0, y_0, \sigma) = \sum_{(x,y) \in W(x_0,y_0)} \begin{bmatrix} I_x(x,y,\sigma) I_x(x,y,\sigma) & I_x(x,y,\sigma) I_y(x,y,\sigma) \\ I_y(x,y,\sigma) I_x(x,y,\sigma) & I_y(x,y,\sigma) I_y(x,y,\sigma) \end{bmatrix}$$

A solution to scale

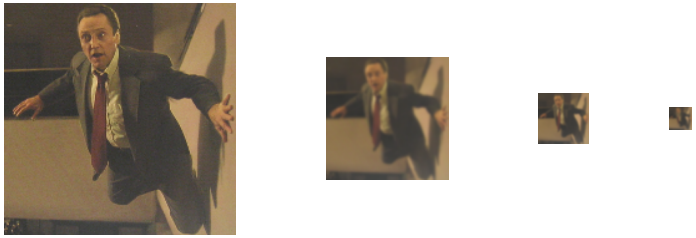


$$\text{cornerness}(x_o, y_o, \sigma) = \det(A(x_0, y_0, \sigma)) - \alpha \text{Trace}^2(A(x_0, y_0, \sigma))$$

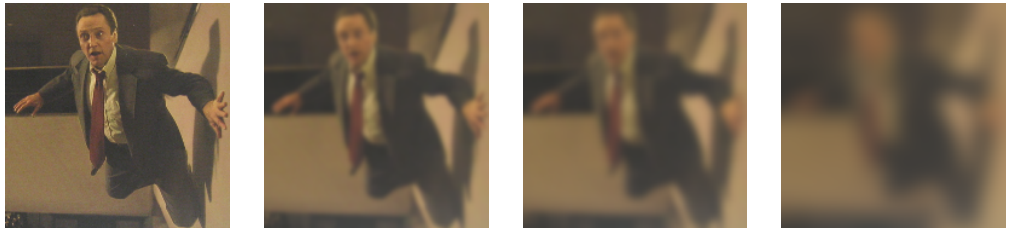
Look for local maxima in (x,y,sigma) … with say, non-maxima suppression

Annoying "details"

1. Positions across scales don't align



Soln: construct blurred (but not subsampled) versions of image



2. Gradients across scales aren't comparable (gradients *always* smaller on blurred images)

Soln: multiply gradients by scale factor

Scale-space theory: A basic tool for analysing structures at different scales

Tony Lindeberg

Computational Vision and Active Perception Laboratory (CVAP)
Department of Numerical Analysis and Computing Science
Royal Institute of Technology, S-100 44 Stockholm, Sweden

Putting it all together: Harris-Laplacian detector

$$A(x_0, y_0, \sigma_D) = \sum_{(x,y)\in W(x_0,y_0)} G(x, y; x_0, y_0, \sigma_I) \begin{bmatrix} I_x(x, y, \sigma_D) I_x(x, y, \sigma_D) & I_x(x, y, \sigma_D) I_y(x, y, \sigma_D) \\ I_y(x, y, \sigma_D) I_x(x, y, \sigma_D) & I_y(x, y, \sigma_D) I_y(x, y, \sigma_D) \end{bmatrix}$$

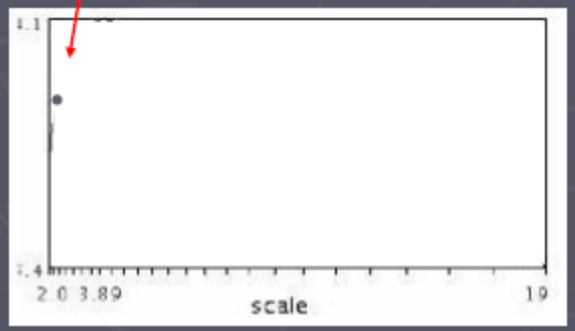
Relate Gaussian for *integration* (patch size) with Gaussian for computing *derivatives* (edge scale)

Heuristic: $\sigma_D = .7\sigma_I$ (roughly corresponds to 3sigma rule)

https://en.wikipedia.org/wiki/Harris_affine_region_detector

Scale selection in 2D

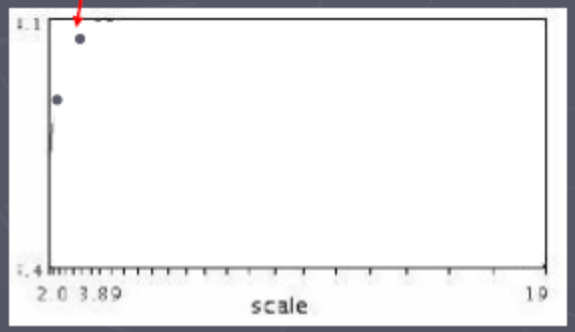
Lindeberg et al., 1996



$$f(I_{i_1 \ldots i_m}(x,\sigma))$$

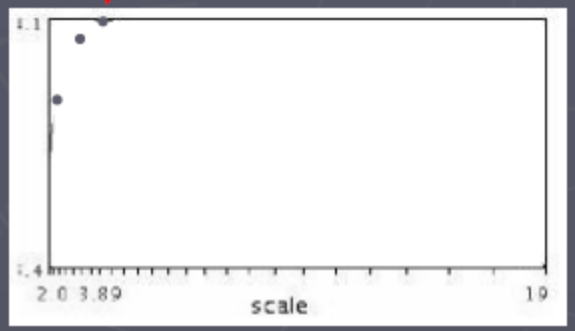
Slide from Tinne Tuytelaars

Scale selection in 2D



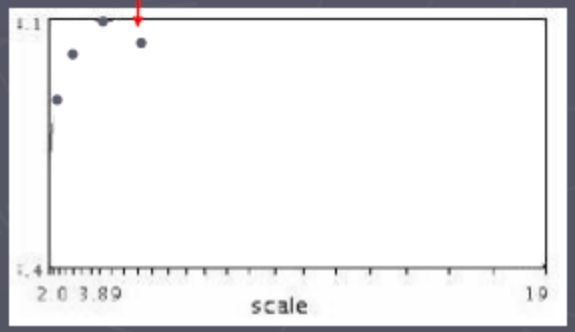
$$f(I_{i_1 \ldots i_m}(x,\sigma))$$

Scale selection in 2D



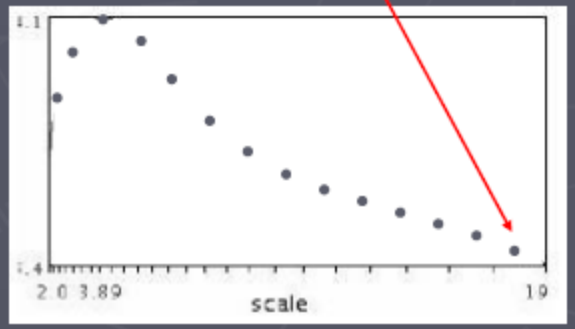
$$f(I_{i_1 \ldots i_m}(x,\sigma))$$

Scale selection in 2D



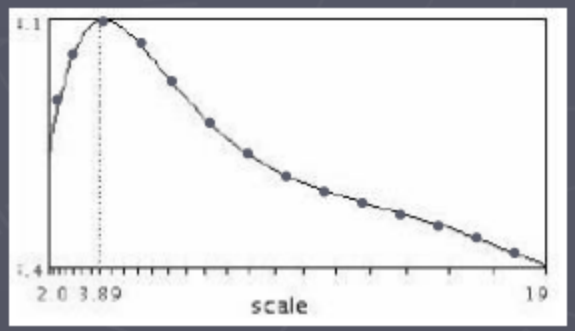
$$f(I_{i_1 \ldots i_m}(x,\sigma))$$

Scale selection in 2D



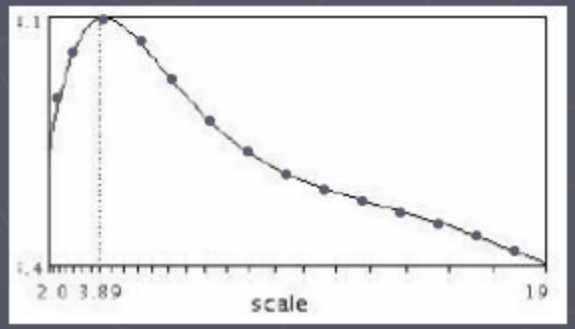
$$f(I_{i_1 \ldots i_m}(x,\sigma))$$

Scale selection in 2D

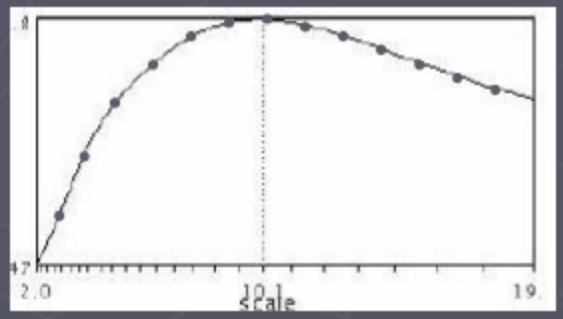


$$f(I_{i_1 \ldots i_m}(x,\sigma))$$

Scale selection in 2D



$f(I_{i_1 \ldots i_m}(x,\sigma))$



$f(I_{i_1 \ldots i_m}(x',\sigma'))$

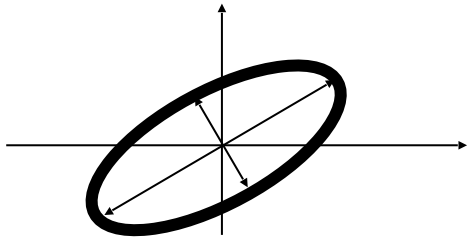
Challenge: correspondence across viewpoint changes

Extension: anisotropic scale

Need richer description of "spatial neighborhooh of integration"

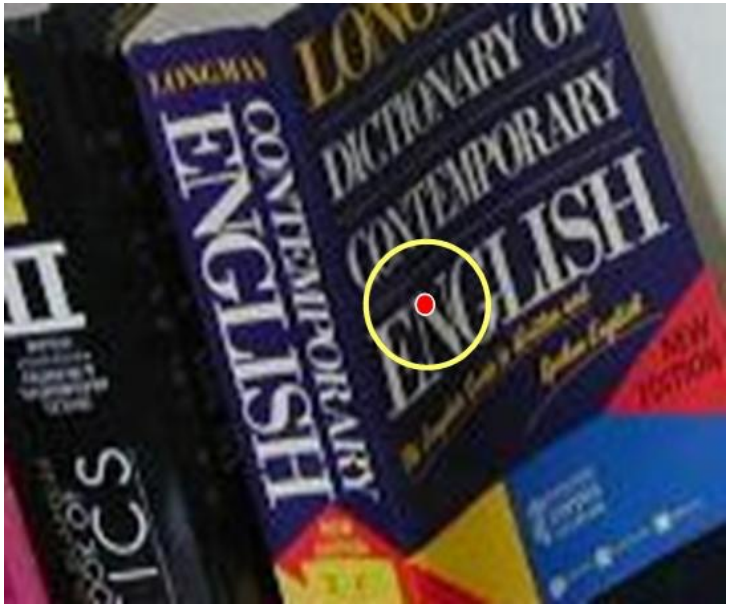
Replace scalar σ with $\sum$

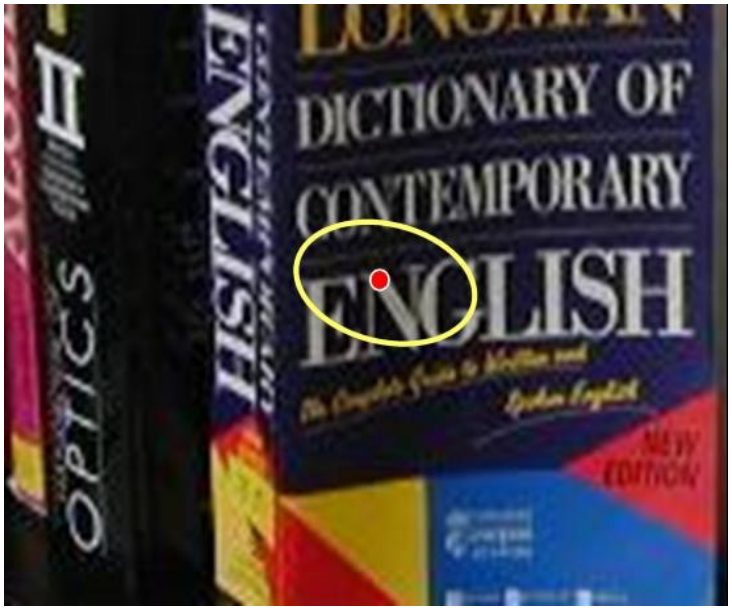
(e.g., scale differently long x and y, or even a diagonal axis)

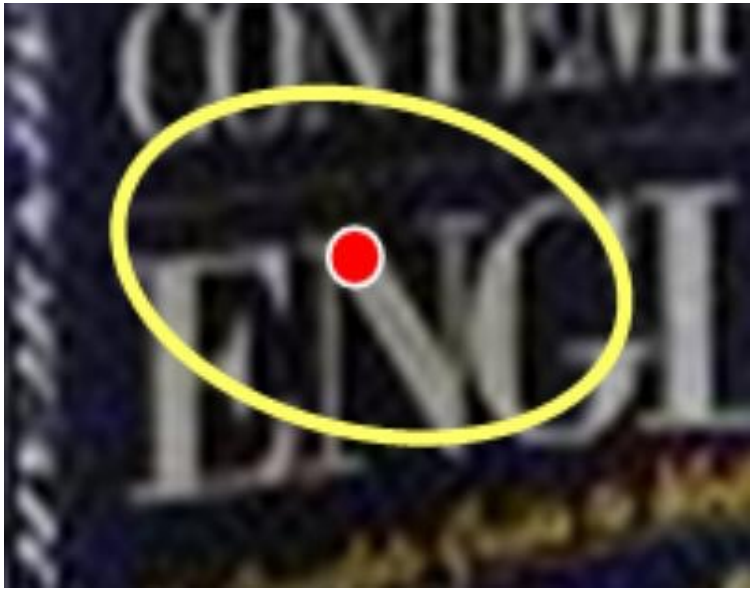


1. Optimize cornerness(x,y,Σ) over discrete set of locations and scales

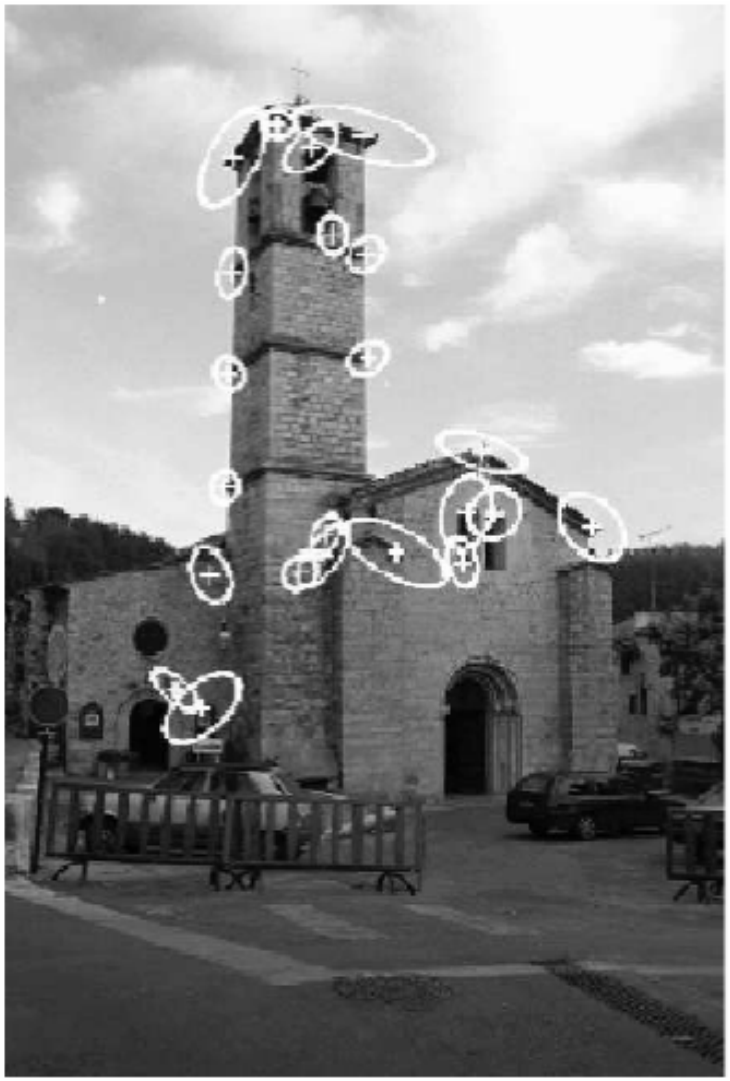
Affine Invariance

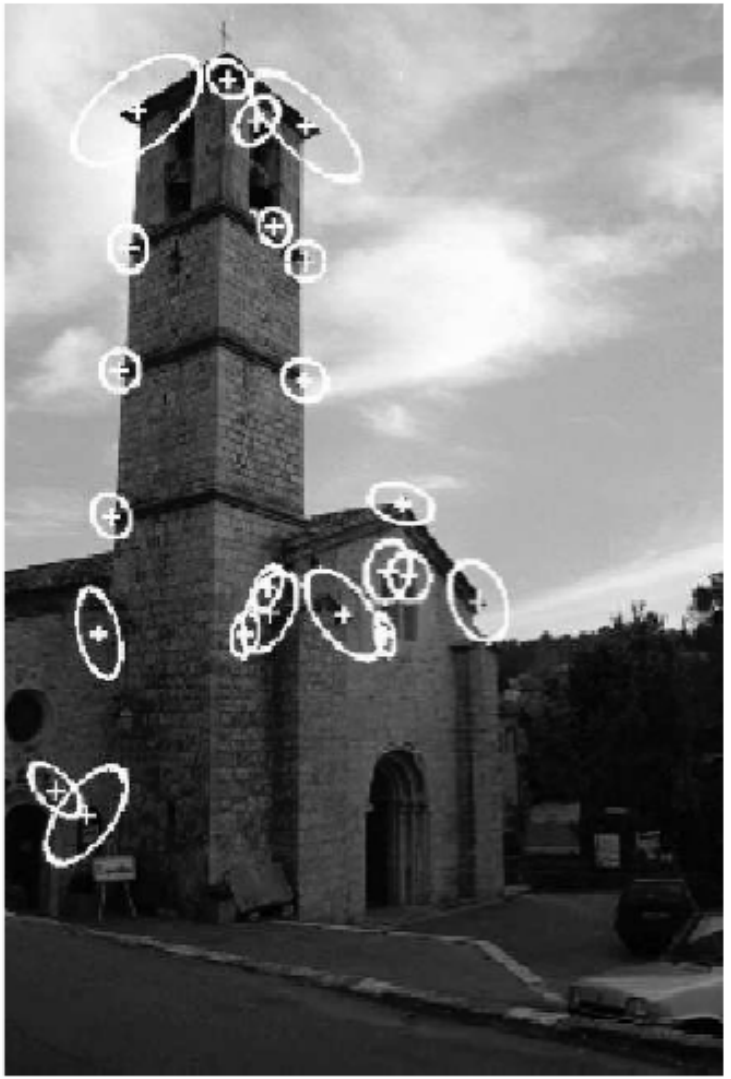






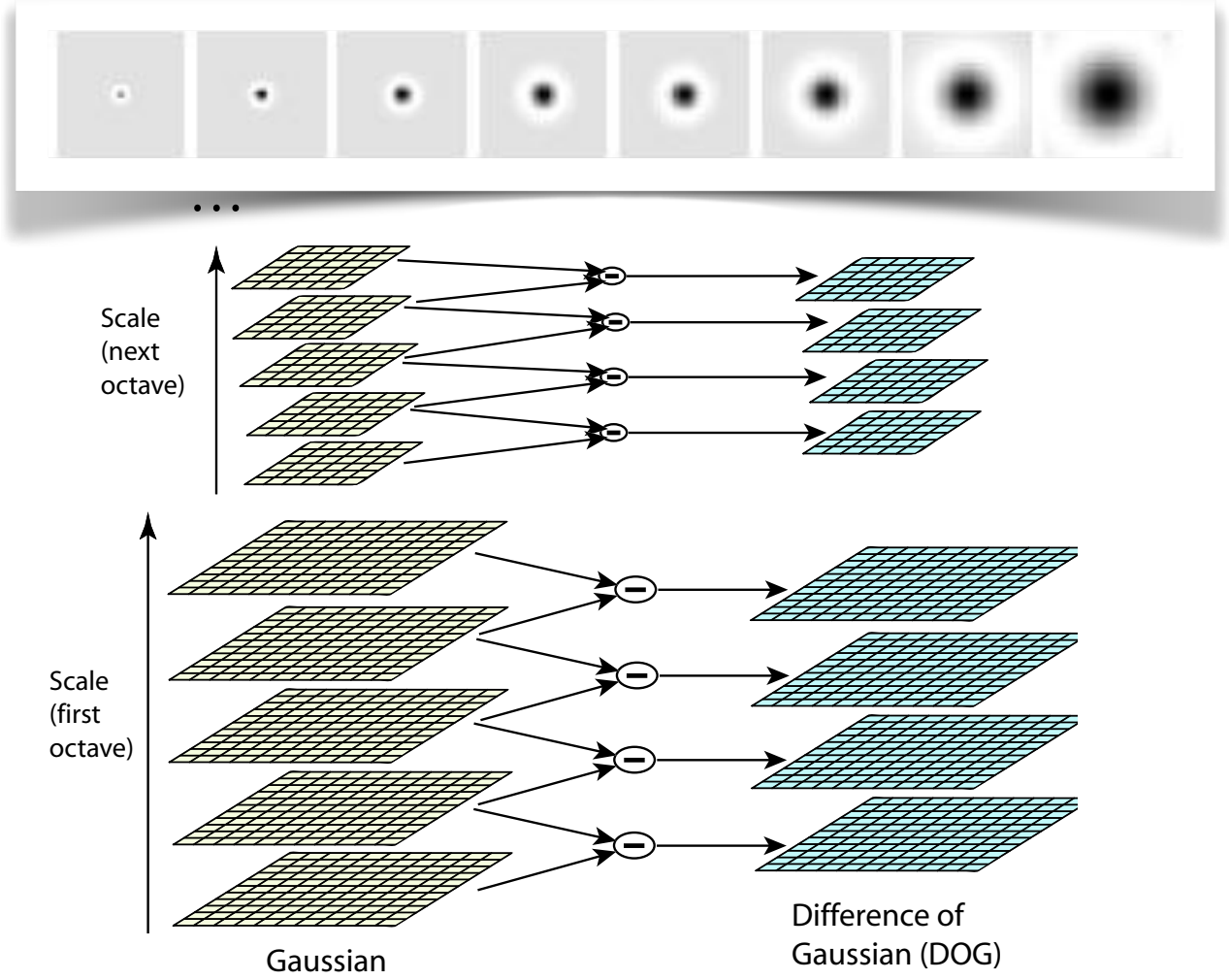
Application: Finding correspondences





Much simpler alternative: throw away Taylor-series machinery and look for local maxima of blob detector in [x,y,scale] space!

https://en.wikipedia.org/wiki/Scale-invariant_feature_transform



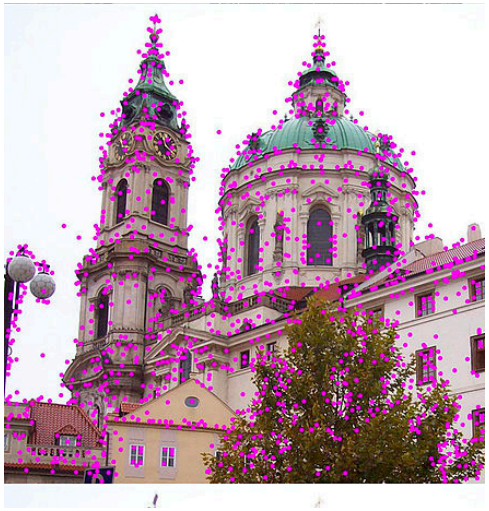
$$D(x, y, \sigma) \quad = \quad (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y)$$

$k = 2^{\frac{1}{s}}$ where s = # levels in an "octave" or 2X subsampling

Look for “blob detections” that are

locally maximal, high confidence, and localizeable

Local maxima of D(x,y,sigma)



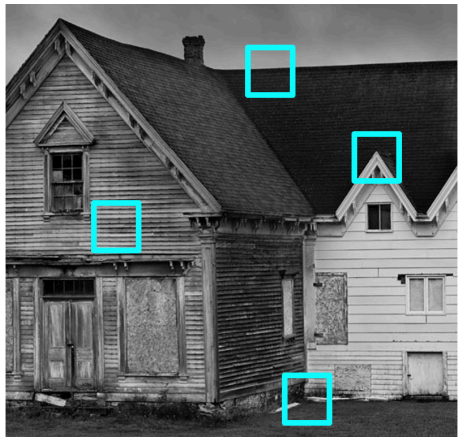
D(x,y,sigma) > thresh

$$\begin{bmatrix} D_{xx} & D_{xy} \\ D_{yx} & D_{yy} \end{bmatrix}$$

Use *second*-order taylor expansion (Hessian) to get “subpixel” accuracy

https://en.wikipedia.org/wiki/Scale-invariant_feature_transform

A look back

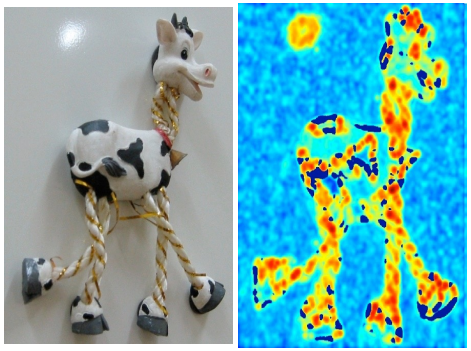


$$\text{cornerness}(x_0, y_0) = \min_{u^2 + v^2 = 1} E_{x_0,y_0}(u, v)$$

where

$$E_{x_0,y_0}(u, v) = \sum_{(x,y) \in W(x_0,y_0)} [I(x + u, y + v) - I(x, y)]^2$$

$$E(u, v) \approx \begin{bmatrix} u & v \end{bmatrix} B \begin{bmatrix} u \\ v \end{bmatrix}, \quad B = \sum_{(x,y) \in W(x_0,y_0)} \begin{bmatrix} I_x(x, y) I_x(x, y) & I_x(x, y) I_y(x, y) \\ I_y(x, y) I_x(x, y) & I_y(x, y) I_y(x, y) \end{bmatrix}$$



Overall pipeline for finding correspondence in 2 images

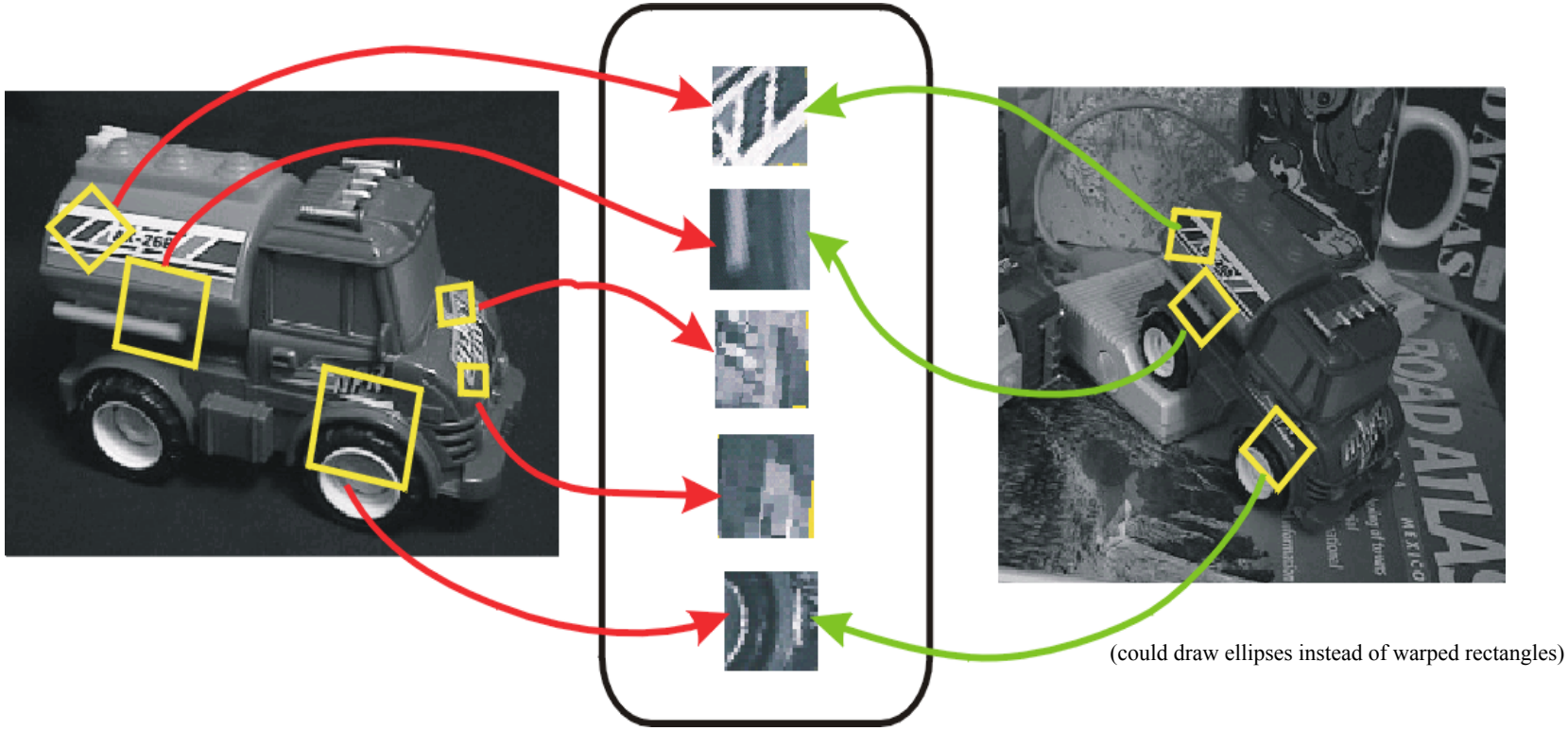
- *Interest point detection*: Find N "interesting" patches in left image and M "interesting" patches in right image


- ***Descriptors*: Compute O(NM) patch similarities and candidate correspondences (e.g., each patch in left image, find best match in right)**


- *RANSAC*: Extract subset of correspondences that are consistent with a model

Coordinate frames

Because surfaces will appear warped from different views, we want to ideally extract features in a *normalized* coordinate frame



(could draw ellipses instead of warped rectangles)

Represent each patch in a local coordinate frame

$$d(p_1, p_2) = \left\| \begin{bmatrix} \cdot \\ \cdot \end{bmatrix} - \begin{bmatrix} \cdot \\ \cdot \end{bmatrix} \right\|$$

Coordinate frame search vs descriptor invariance

Interest-point / coordinate-frame search:

$$cornerness(x, y)$$

Ideal descriptor invariance:

scale (and rotation, viewpoint, illumination,…)

Coordinate frame search vs descriptor invariance

Interest-point / coordinate-frame search:

$cornerness(x, y)$

$cornerness(x, y, \sigma)$

Ideal descriptor invariance:

scale (and rotation, viewpoint, illumination,…)

~~scale~~ (and rotation, viewpoint, illumination,…)
assuming we get scale right….

Coordinate frame search vs descriptor invariance

Interest-point / coordinate-frame search:

$cornerness(x, y)$

Ideal descriptor invariance:

scale (and rotation, viewpoint, illumination,…)

$cornerness(x, y, \sigma)$

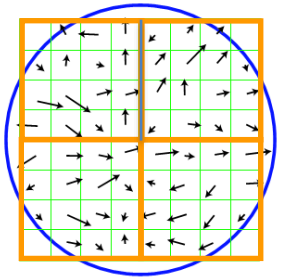
~~scale~~ (and rotation, viewpoint, illumination,…)
assuming we get scale right….

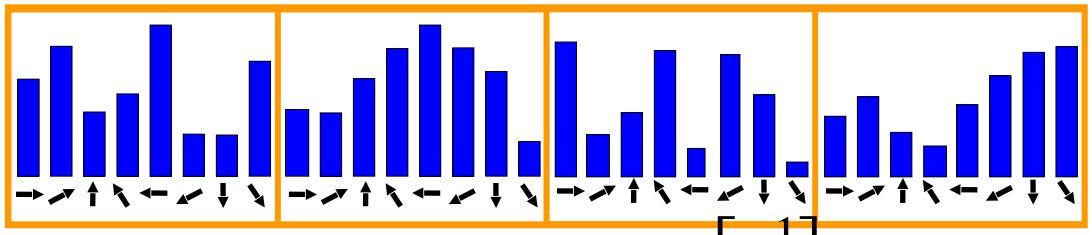
$cornerness(x, y, \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix})$

~~scale~~ (and ~~rotation~~, ~~viewpoint~~, illumination,…)
assuming we get affine coordinate right….

Recall the SIFT descriptor

Histograms of gradient directions over spatial cells





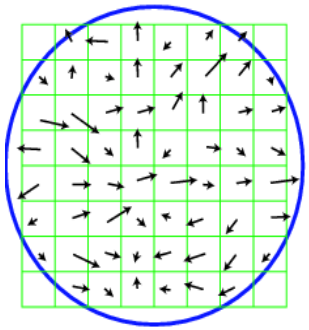
1. Compute gradient vector at each pixel (with [-1 0 1] and $\begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$ filters)
2. Snap gradient vector to one of 8 orientations (North, West, Northwest,…)
3. Split up patch into 2x2 quadrants and count the number of (North, West, Northwest,…) gradients
4. Normalize (and clip and renormalize)

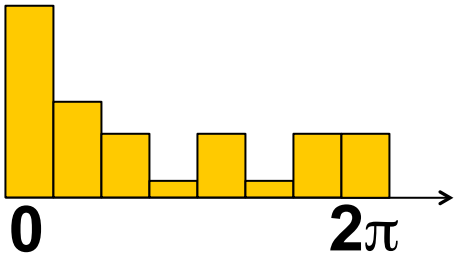
Problem: descriptor is not rotation invariant.

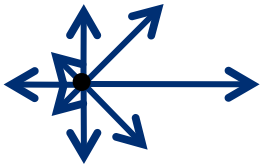
How can one build rotation invariance (to descriptor or search procedure)?

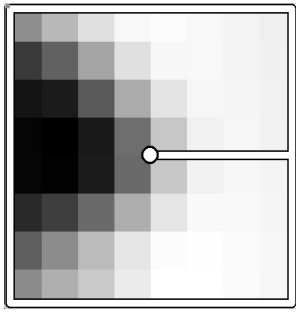
SIFT's approach: use descriptor to rotate coordinate frame

1. Compute gradients for all pixels in patch.
2. Histogram (bin) gradients by orientation.
3. Rotate patch coordinate frame (by *shifting* bin indices) so that dominant orientation is bin "0"









Question: what happens when there are multiple peaks?

Instantiate multiple descriptors!

The enormous impact of SIFT

Distinctive Image Features from Scale-Invariant Keypoints

IJCV 04

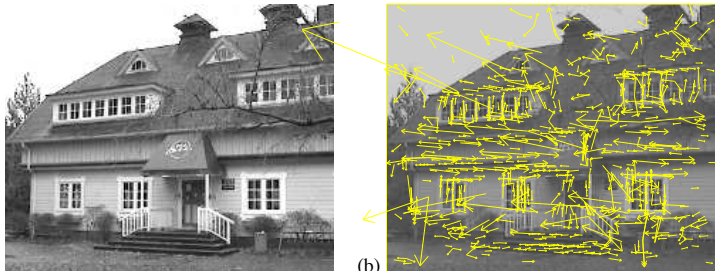
David G. Lowe
Computer Science Department
University of British Columbia
Vancouver, B.C., Canada
lowe@cs.ubc.ca

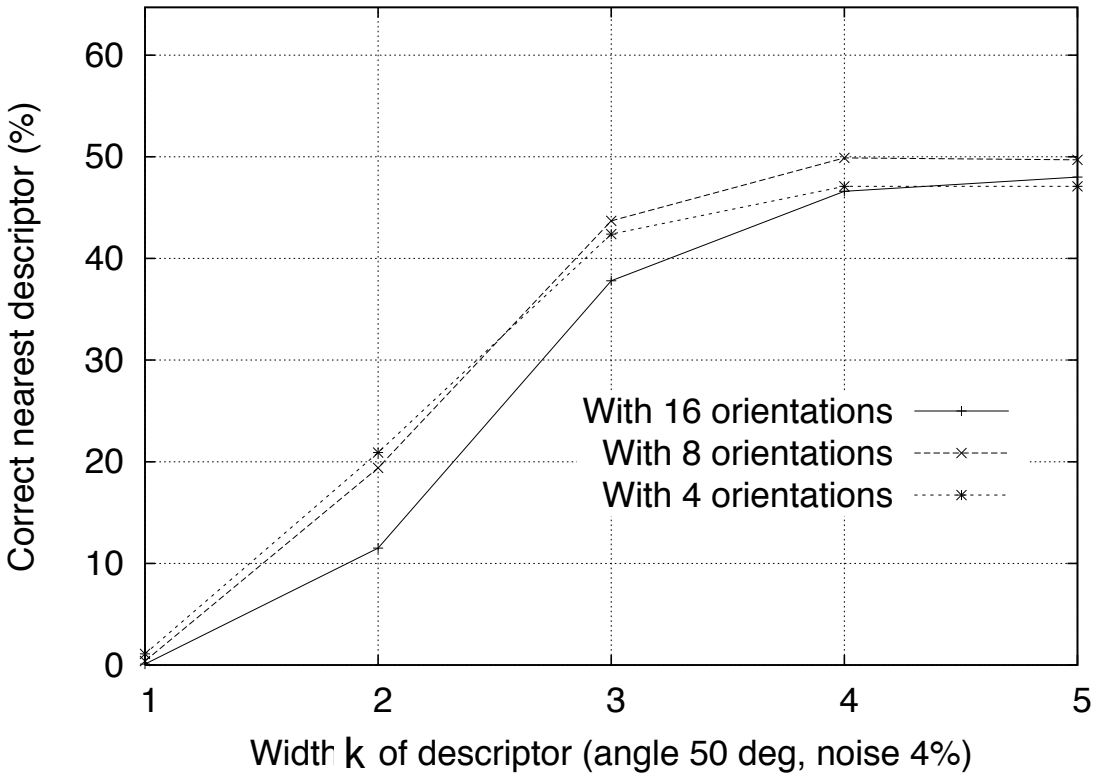
Single handedly improved the impact factor of IJCV to be one of the most cited journals in all of computer science. Why?

SIFT

What made this work? Exhaustive evaluation of hyper-parameters on annotated dataset of good vs bad matches

“Human learning” vs “machine learning”

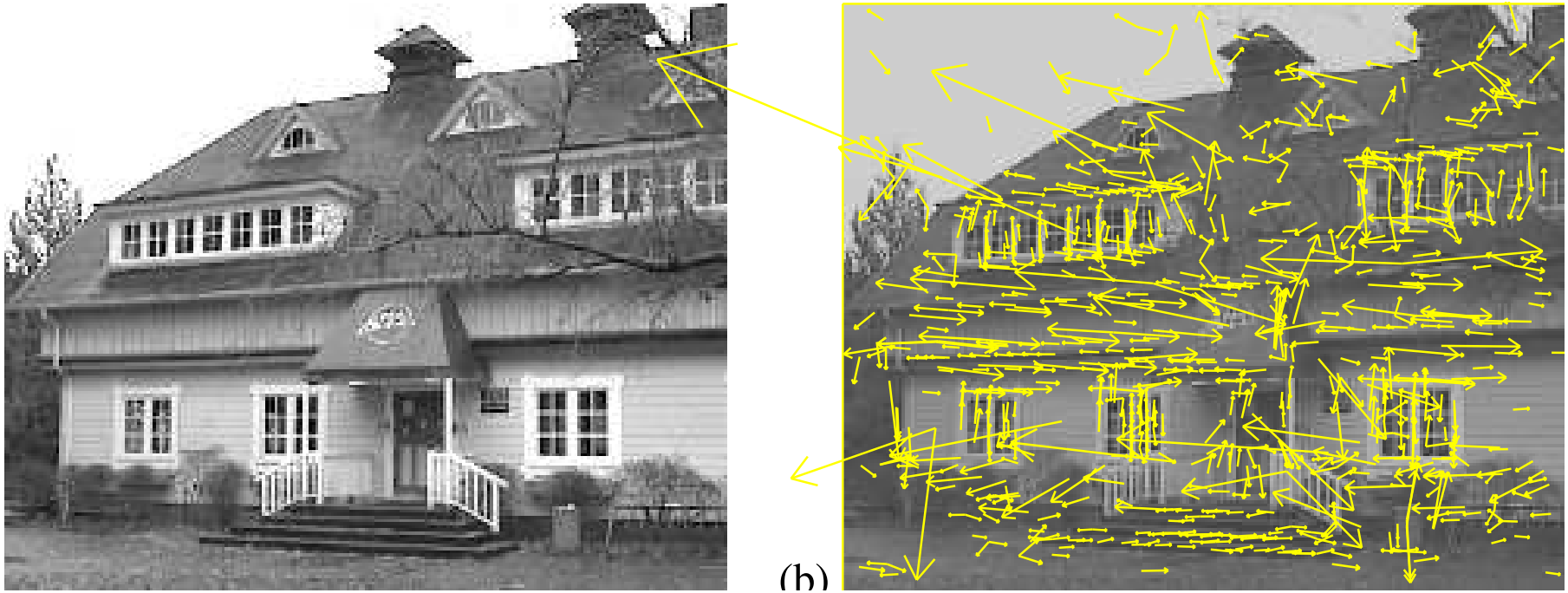




SIFT

Where does scale invariance for the “Scale-Invariant Feature Transform” come from?

Interest point detector; the descriptor itself is not invariant to scale!



Visualization of oriented+scaled patches given by oriented+scaled arrows

Outline

- Motivation
- Interest point detection
  - Cornerness
  - Optimization (first-order,second-order)
  - Scale-space
- Descriptors
  - SIFT
  - Other fast descriptors (SURF, BRIEF)
  - Dense descriptors
- **Ransac [interlude]**