Local features detection and description

MVA – Vision 3D artificielle

Loic Landrieu

loiclandrieu.com

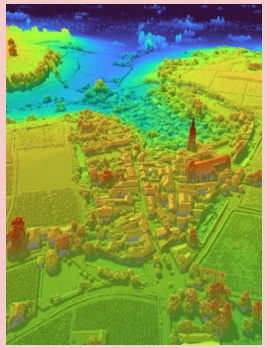
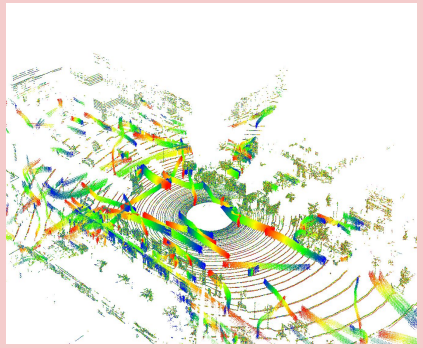
With slides from Mathieu Aubry, Renaud Marlet and François Darmon

What do I work on?

-> let me know if you are interested in internships/PhDs on these topics

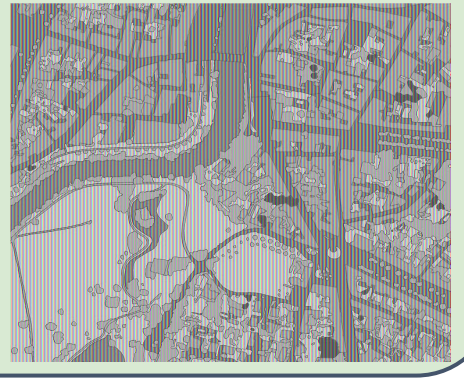
Main Focus: **Efficient machine learning + large-scale geospatial data**

3D data: large-scale + real time + forestry

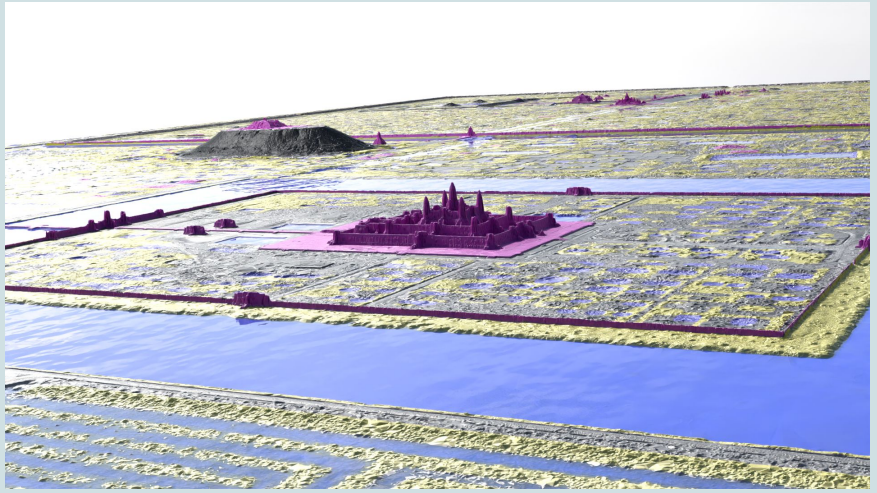




Earth observation: crop mapping /food security

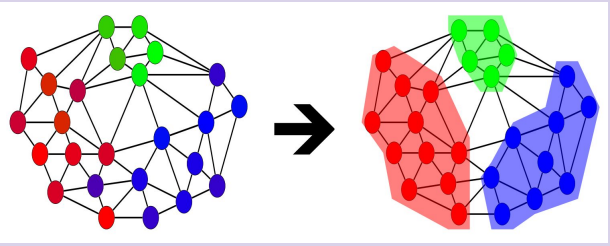




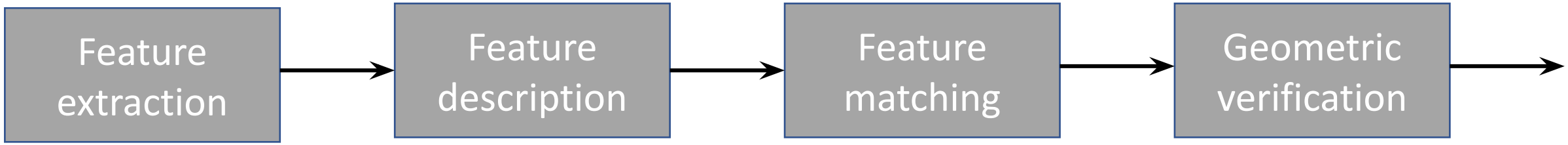
historical geodata:



optimization on graphs:



Local features and correspondences pipeline for 3D reconstruction

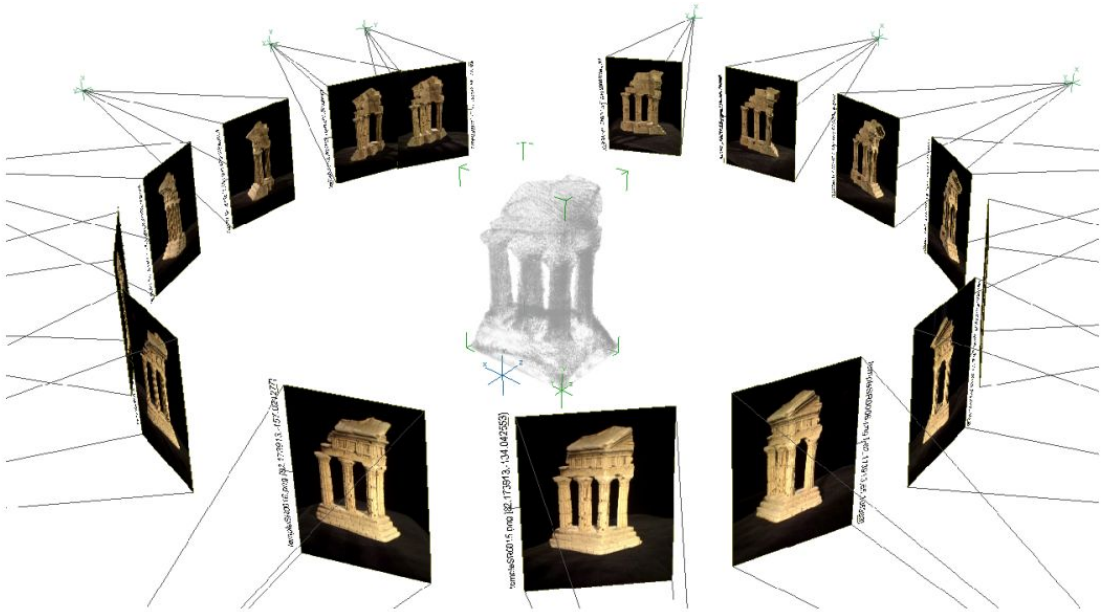


Extract location of ‘local features’ (interest points, keypoints)

Summarize their appearance in a vector

Find matching features

Filter the matches based on geometry



Credit: Shilin Zhu

Motivation: panorama

3D reconstruction

- External camera calibration

  = determination of pose (i.e., location and orientation)
  of each camera in a common coordinate system

  - requires enough corresponding points in several images

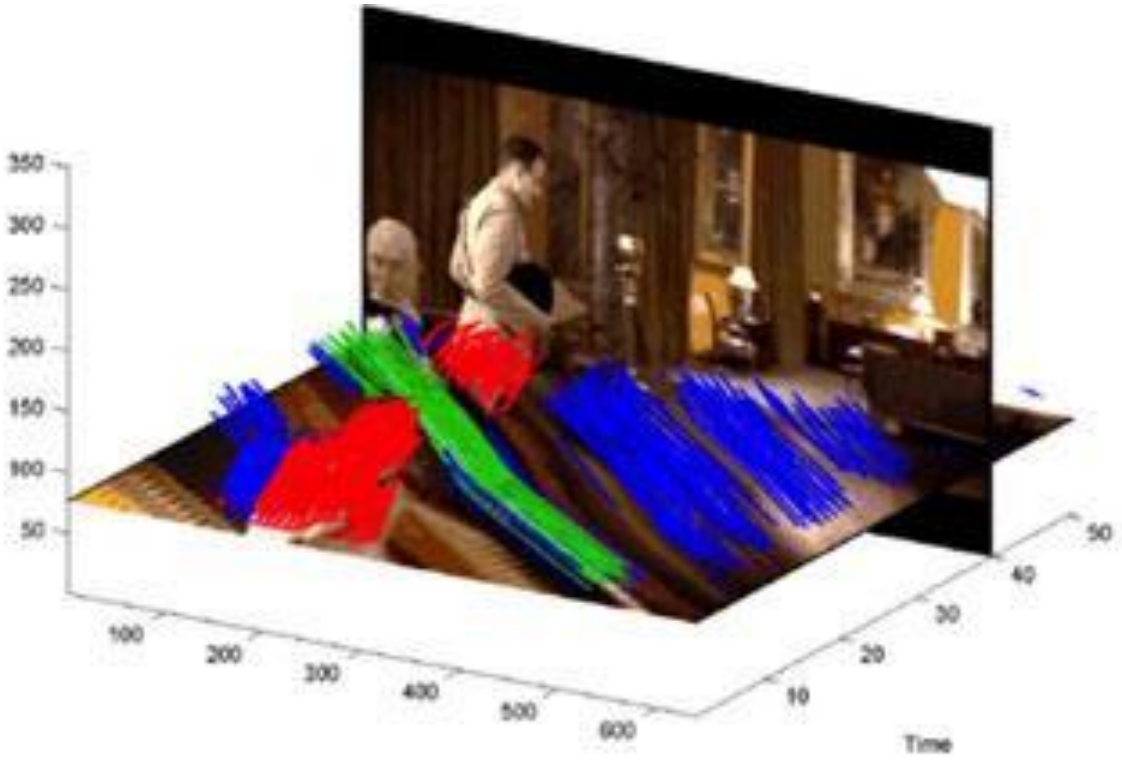
  → **detection** and **matching** of salient points

- Dense 3D reconstruction

  = by triangulation, given camera pose
  (!) not restricted to salient points only

  - requires **matching** image patches in several images

Motivation: tracking



J. Lezama, K. Alahari, J. Sivic, I. Laptev

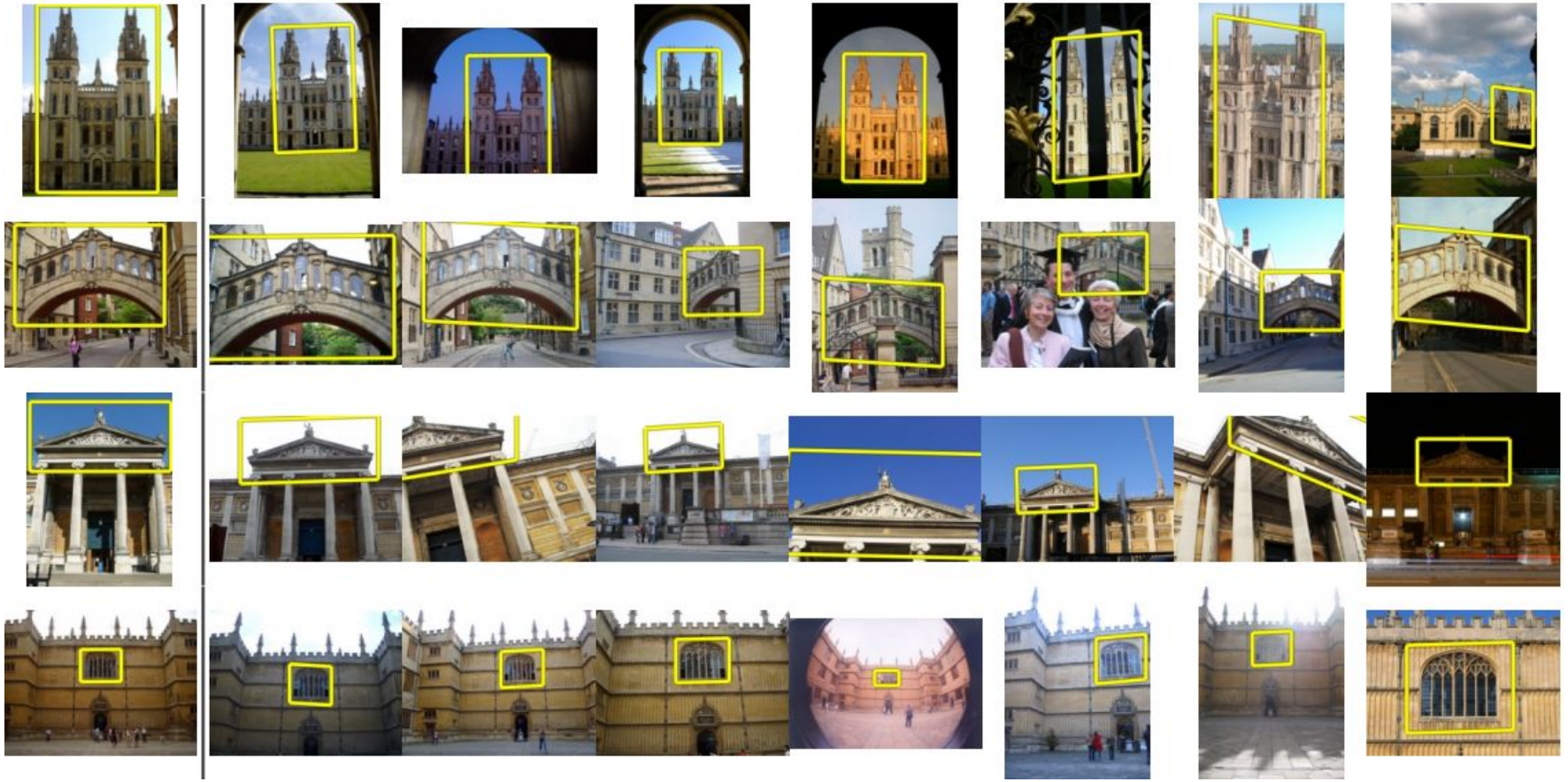
Track to the Future: Spatio-temporal Video Segmentation with Long-range Motion Cues

CVPR 2011

Motivation: instance retrieval

1. Identify salient points
2. Look for similar salient points in other image
3. Check geometrical consistency (rigid or deformable)

Motivation: content-based image retrieval

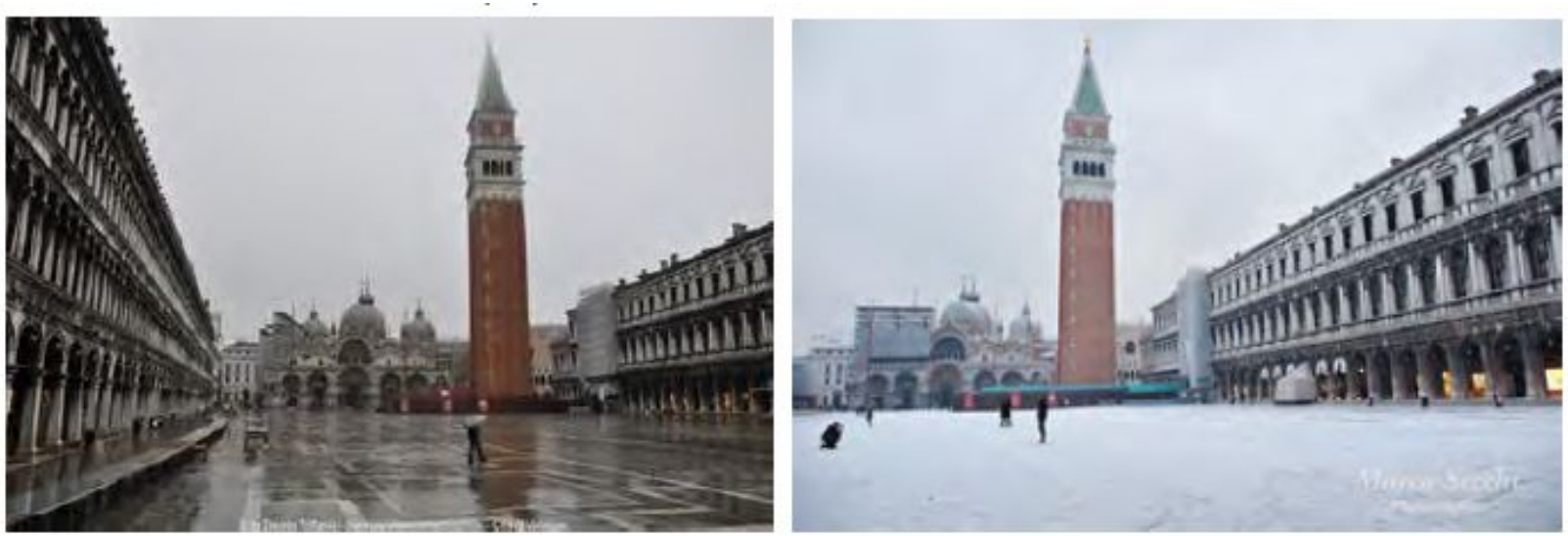


Philbin, J. , Chum, O. , Isard, M. , Sivic, J. and Zisserman, A.
Object retrieval with large vocabularies and fast spatial matching
CVPR 2007

Difficulty

Illumination changes

Difficulty



Season changes

Difficulty

Viewpoint changes

Difficulty

Clutter and occlusion

Difficulty

- change of scale (camera motion or change of focal length)
- change of orientation (rotation)
- change of viewpoint (affine, projective transformations)
- change of illumination (time of day, weather, flash...)

And also

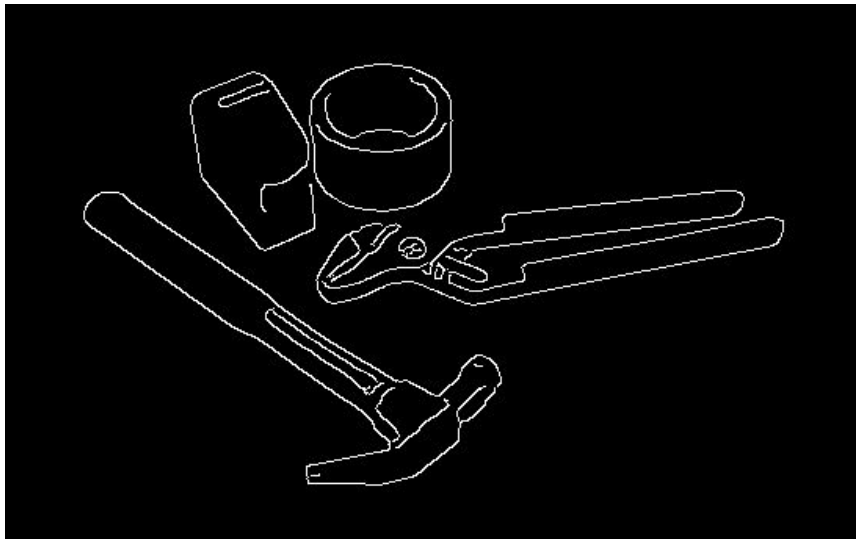
- change of camera parameters (speed/aperture ...)
- non-rigid scene (objects in motion, deformable surface)
- surface reflectance (Lambertian or not, reflection,transpar.)
- repetitive patterns (windows, road marks...)

What is a local feature?

- Distinctive and identifiable region of an image
- Invariant to various transformations and noise
- Can be retrieved robustly
  - Points
  - Edges
  - Regions

Which features?

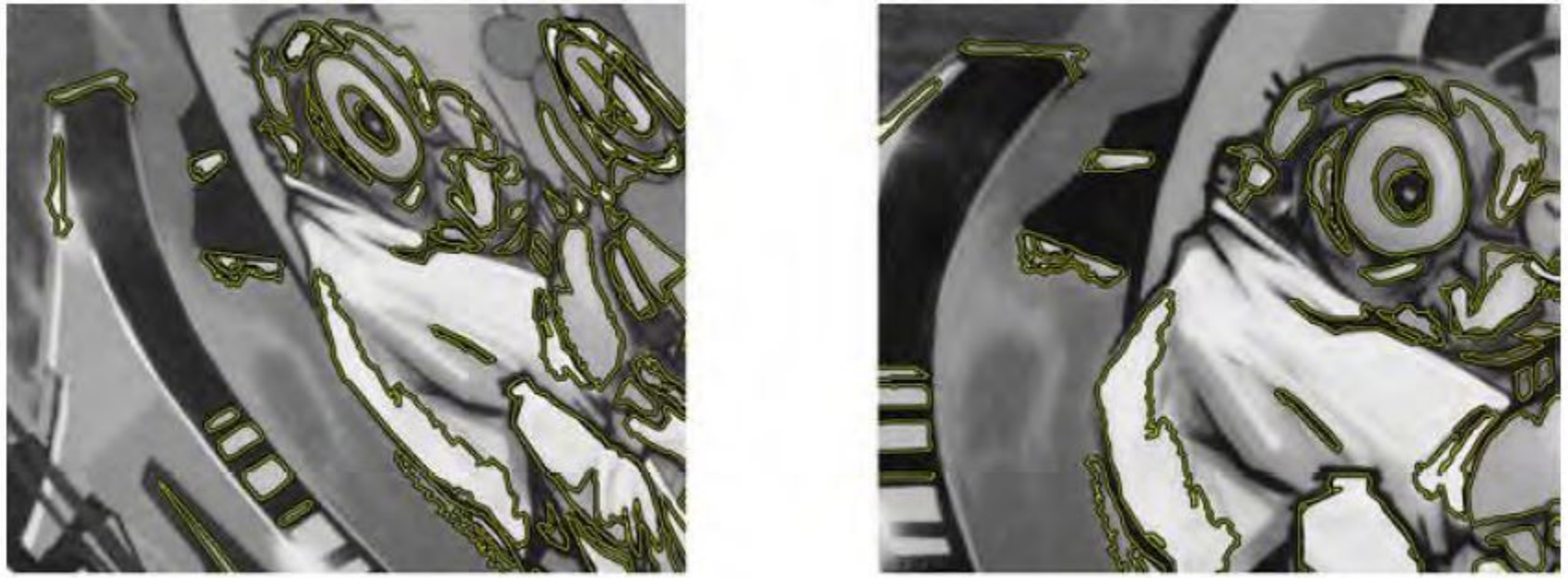
- **Edges**: eg. Canny edges



Canny, J. (1986). A computational approach to edge detection.
IEEE Transactions on pattern analysis and machine intelligence

Which features?

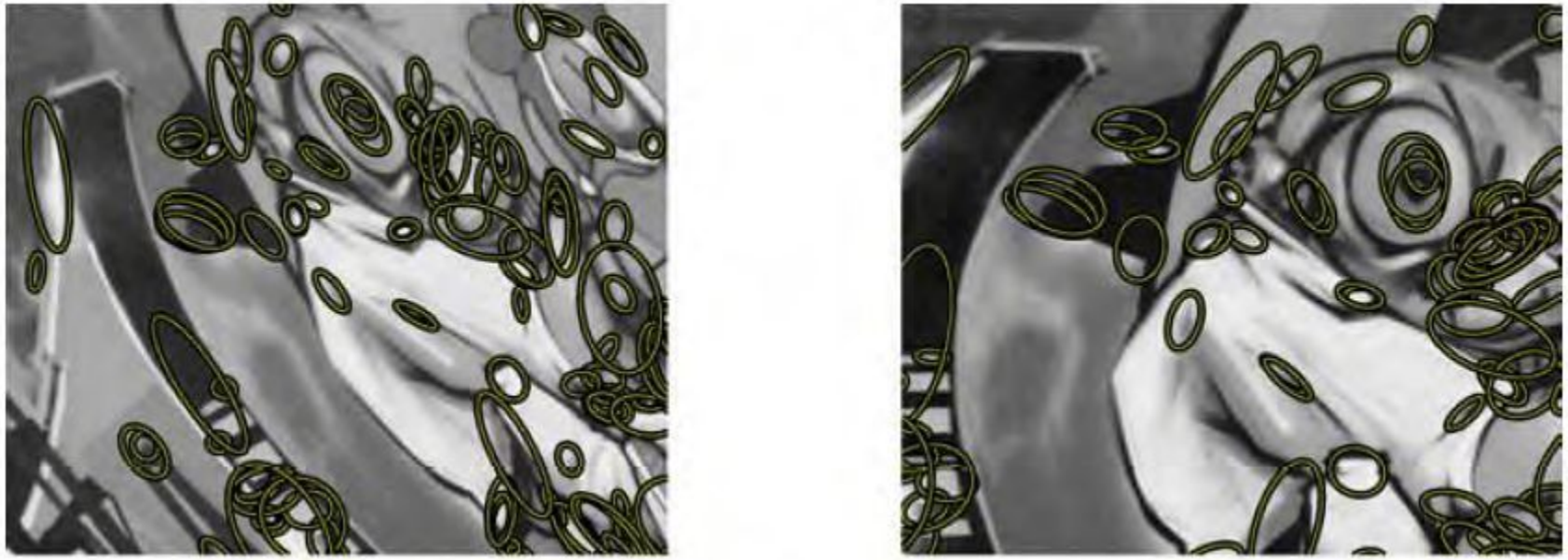
- Regions: eg. MSER (maximally stable extremal regions)



J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. Image and Vision Computing, 2004

Which features?

- Simple regions, blobs: eg. Harris-affine

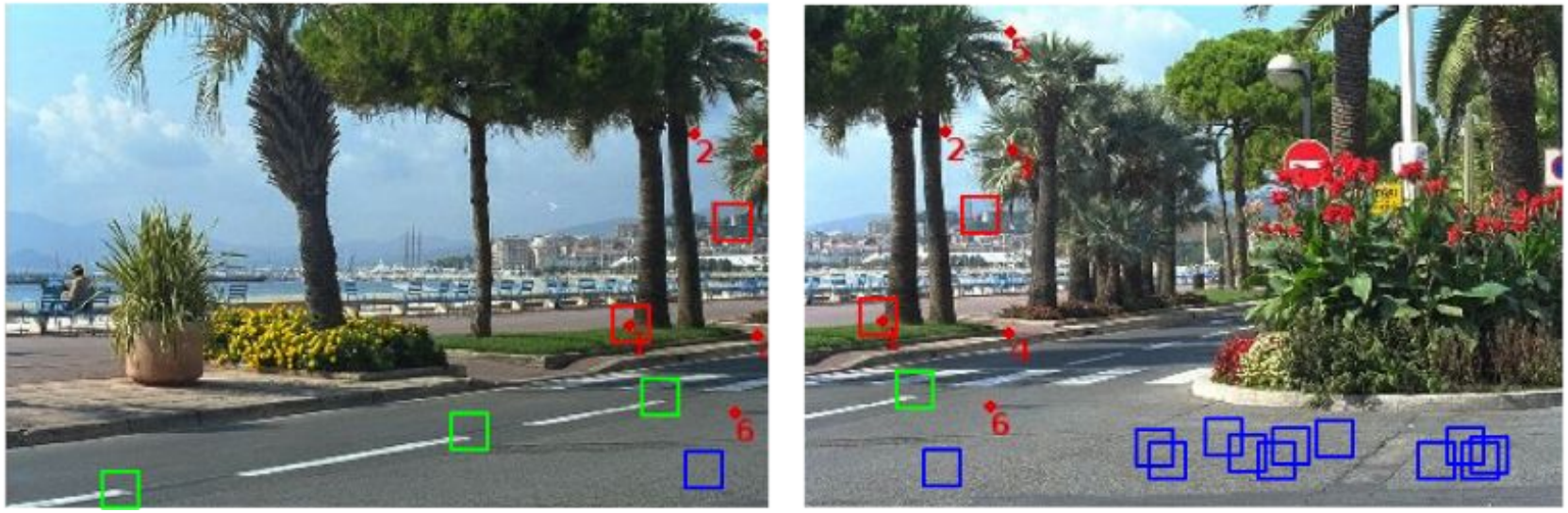


Which features?

- Points

Which features?

- distinctive/repeatable



Outline

1. **<u>Classical local features</u>**

   - Reminder on **convolutions/derivatives**
   - **Feature detection**: How to extract informative features consistently?
   - **Feature description**: how to compare features?
   - Some more discussion of **SIFT and SURF**

2. Deep local features
3. Some deep 3D reconstruction

Convolutions

- $f, g : \mathbb{R}^d$ or $\mathbb{Z}^d \to \mathbb{R}$ (or with values in $\mathbb{C}$)

- Ex. 2D continuous convolution

$$(f*g)(x,y) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} f(u,v)\,g(x-u,y-v)\,du\,dv = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} f(x-u,y-v)\,g(u,v)\,du\,dv$$

- Ex. 2D discrete convolution

$$(f*g)(i,j) = \sum_{k=-\infty}^{+\infty}\sum_{l=-\infty}^{+\infty} f(i,j)\,g(i-k,j-l) = \sum_{k=-\infty}^{+\infty}\sum_{l=-\infty}^{+\infty} f(i-k,j-l)\,g(k,l)$$

... if integral/sum exists

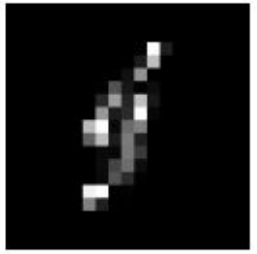
- sufficient: f compactly supported, f integrable and g bounded...

- Efficient convolution in Fourier

Blur-Convolution

- Blurred image: $O = K * I$

e.g. uniform motion blur



K

O

I

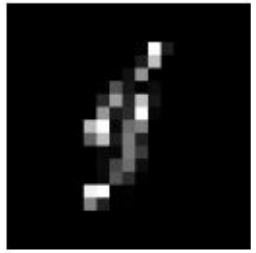
Levin, A., Weiss, Y., Durand, F., & Freeman, W. T. (2009, June). Understanding and evaluating blind deconvolution algorithms. *CVPR 2009*

Convolution

$$(K * I)(i,j) = \sum_{k,l} K(-k,-l)I(i+k,j+l)$$

I

*



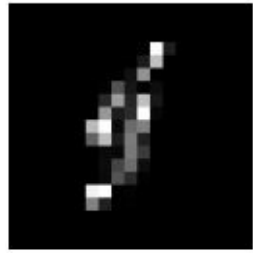
K

Convolution

$$(K * I)(i,j) = \sum_{k,l} K(-k,-l)I(i+k,j+l)$$

I

$*$



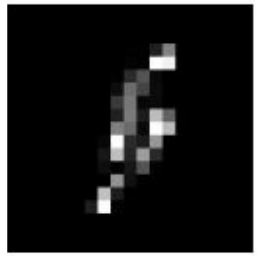
K

Convolution

$$(K * I)(i,j) = \sum_{k,l} K(-k,-l)I(i+k,j+l)$$

I

$*$



$K(-.,-.)$

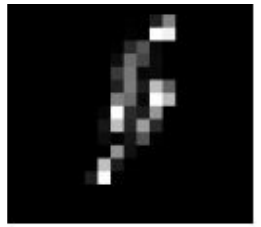
Convolution

$$(K * I)(i,j) = \sum_{k,l} K(-k,-l)I(i+k,j+l)$$

I

$*$

$(K * I)(i,j)$



Weighted average

$(K * I)(i,j)$

Convolution

$$(K * I)(i,j) = \sum_{k,l} K(-k,-l)I(i+k,j+l)$$

I

$(K * I)(i,j)$

Practice with linear filters

Original

0	0	0
0	1	0
0	0	0

?

Source: D. Lowe

Practice with linear filters

Original

0	0	0
0	1	0
0	0	0

Filtered
(no change)

Source: D. Lowe

Practice with linear filters

Original

0	0	0
0	0	1
0	0	0

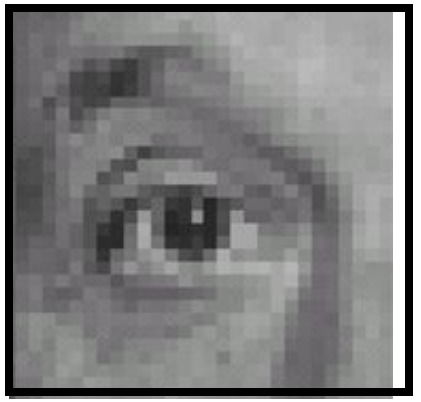
?

Source: D. Lowe

Practice with linear filters

Original

0	0	0
0	0	1
0	0	0



Shifted *left*
By 1 pixel

Source: D. Lowe

Practice with linear filters

Original

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

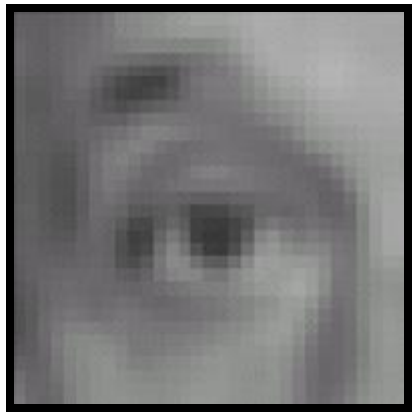
?

Source: D. Lowe

Practice with linear filters

Original

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$



Blur (with a box filter)

Source: D. Lowe

Practice with linear filters

Original

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad ?$$

(Note that filter sums to 1)

Source: D. Lowe

Practice with linear filters

Original

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Sharpening filter

- Accentuates differences with local average

Source: D. Lowe

Derivatives on images

For 2D function f(x,y), the partial derivative is:

$$\frac{\partial f(x,y)}{\partial x} = \lim_{\varepsilon \to 0} \frac{f(x+\varepsilon, y) - f(x,y)}{\varepsilon}$$

For discrete data, we can approximate using finite differences:

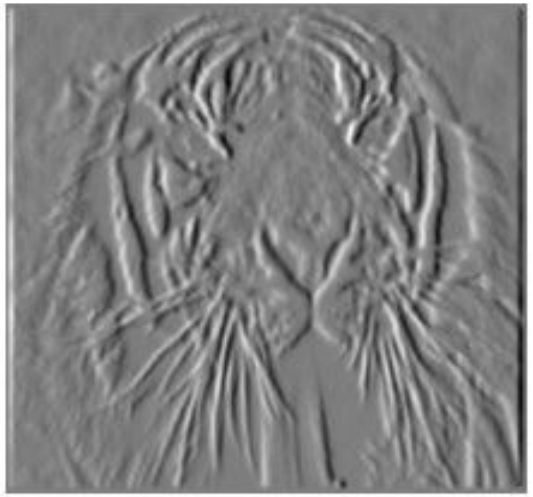
$$\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x+1, y) - f(x,y)}{1}$$

To implement the above as convolution, what would be the associated filter?

Source: K. Grauman

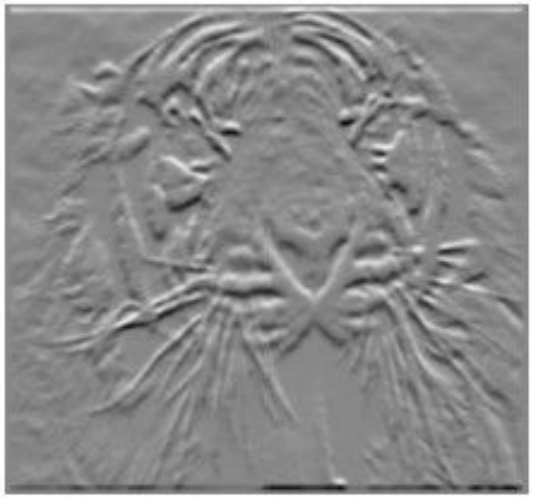
Partial derivatives of an image

$$\frac{\partial f(x,y)}{\partial x}$$



-1	1

$$\frac{\partial f(x,y)}{\partial y}$$



-1
1

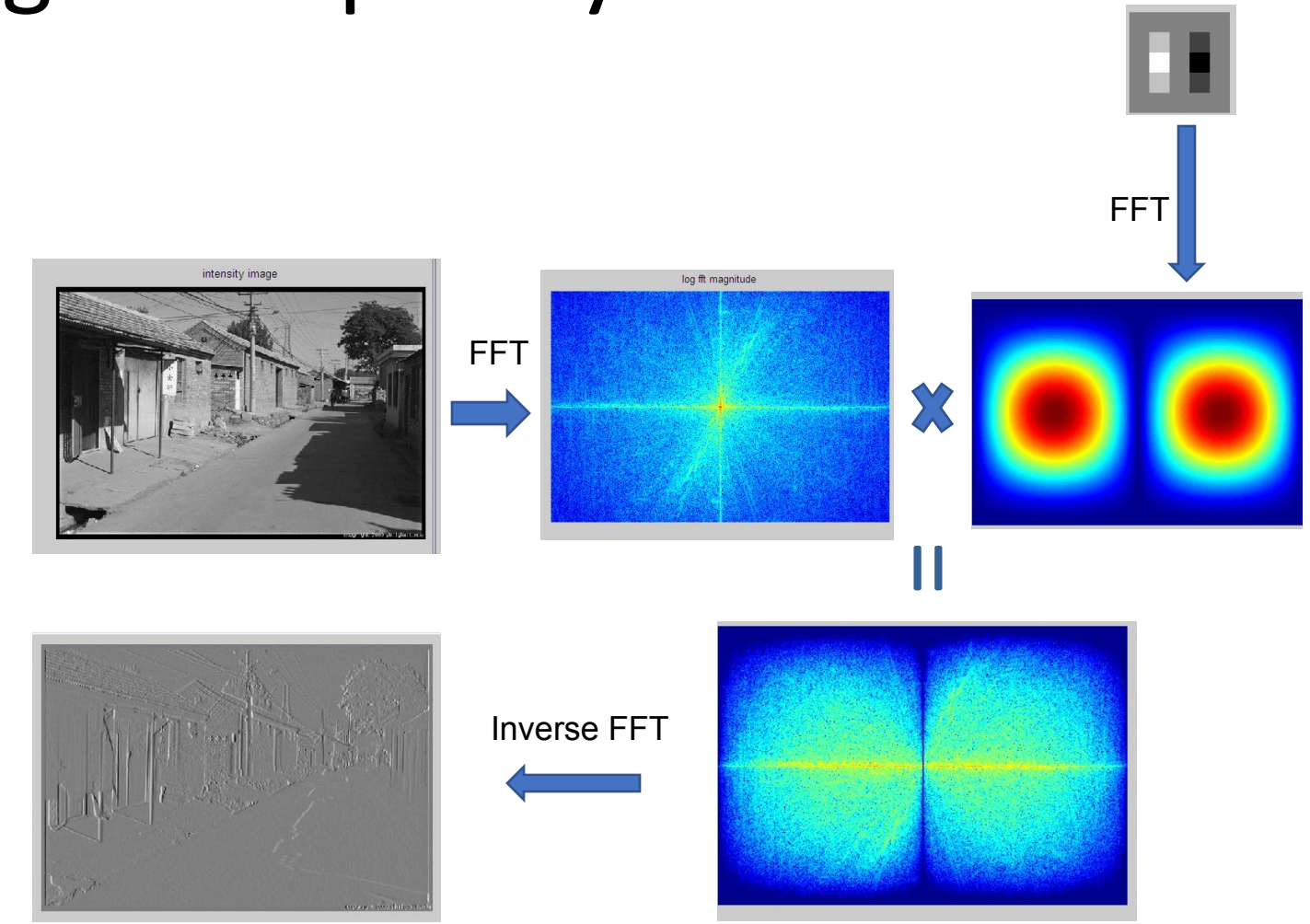
or

1
-1

Which shows changes with respect to x?

Source: L. Lazebnik

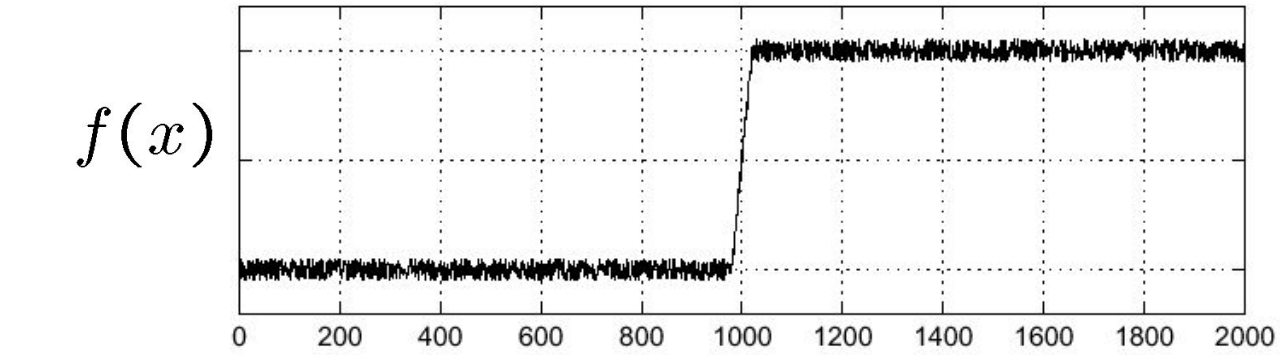
Filtering in frequency domain

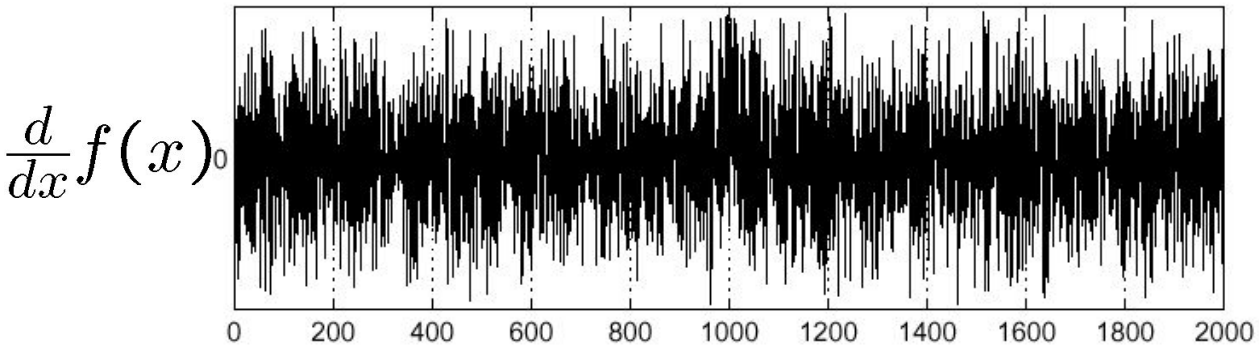


Slide: Hoiem

Effects of noise

- Consider a single row or column of the image

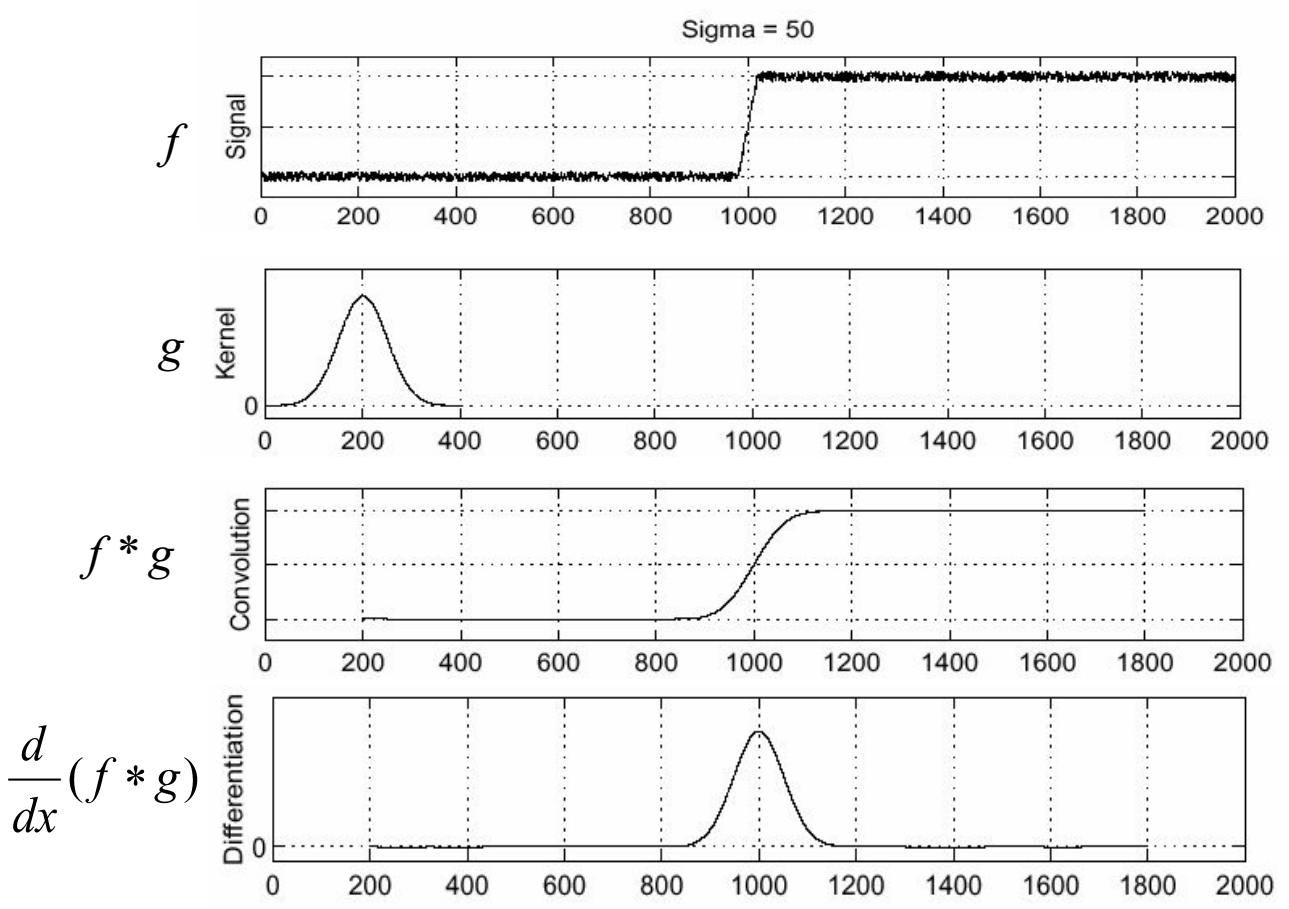




Where is the edge?

Source: S. Seitz

Solution: smooth first



- To find edges, look for peaks in $\frac{d}{dx}(f * g)$

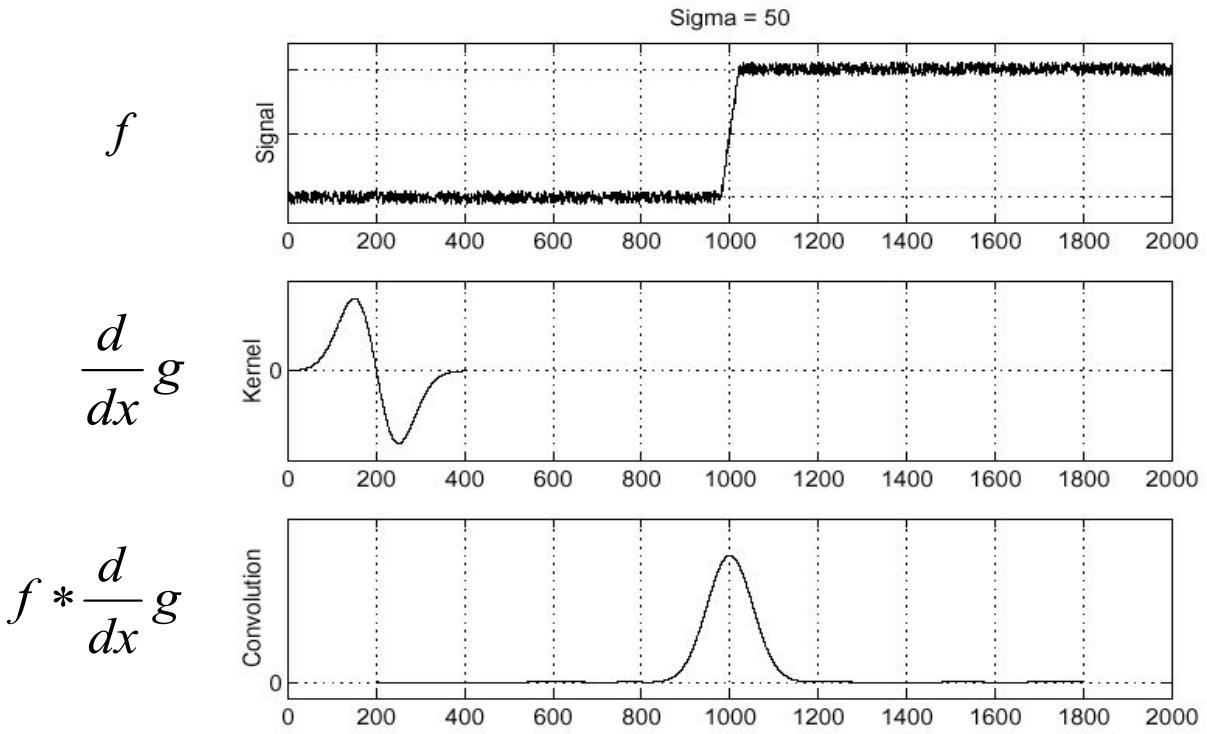
Source: S. Seitz

Derivative theorem of convolution

- Differentiation is convolution, and convolution is associative:

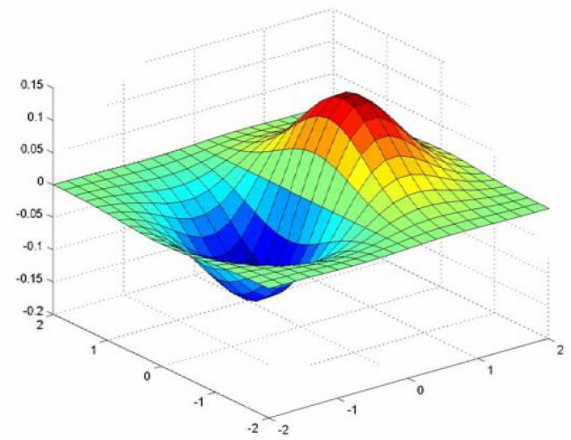
$$\frac{d}{dx}(f * g) = f * \frac{d}{dx} g$$

- This saves us one operation:

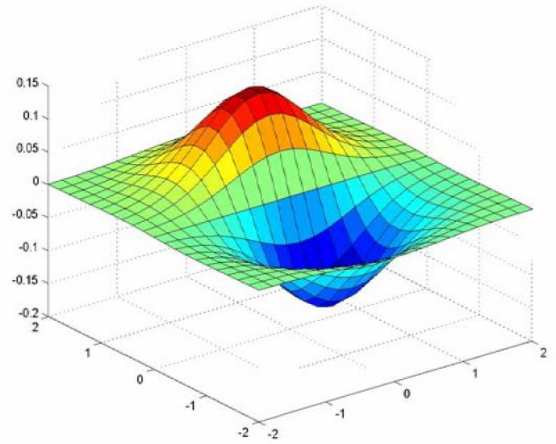


Source: S. Seitz

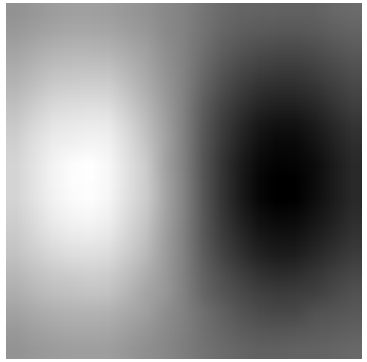
Derivative of Gaussian filters

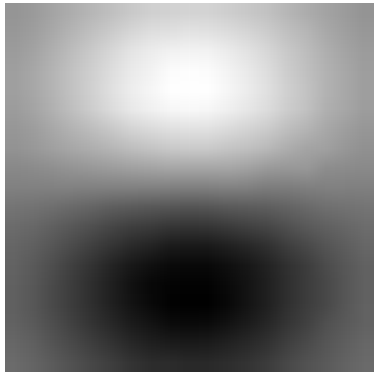


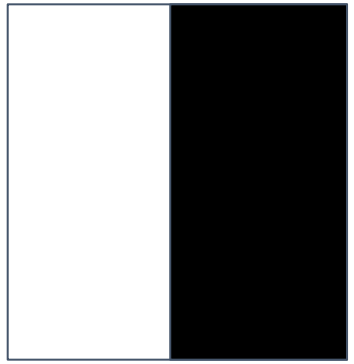
x-direction

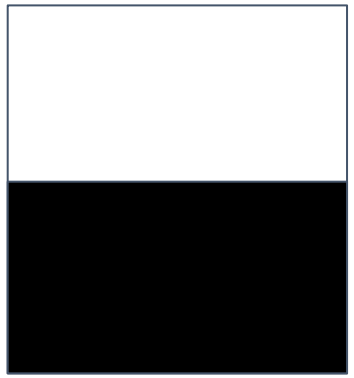


y-direction









- Which one finds horizontal/vertical edges?

Source: L. Lazebnik

Differential operators

More generally, rather than trying to manually design a filter, one can apply an operator to the Gaussian kernel, discretize the result and convolve with the image.

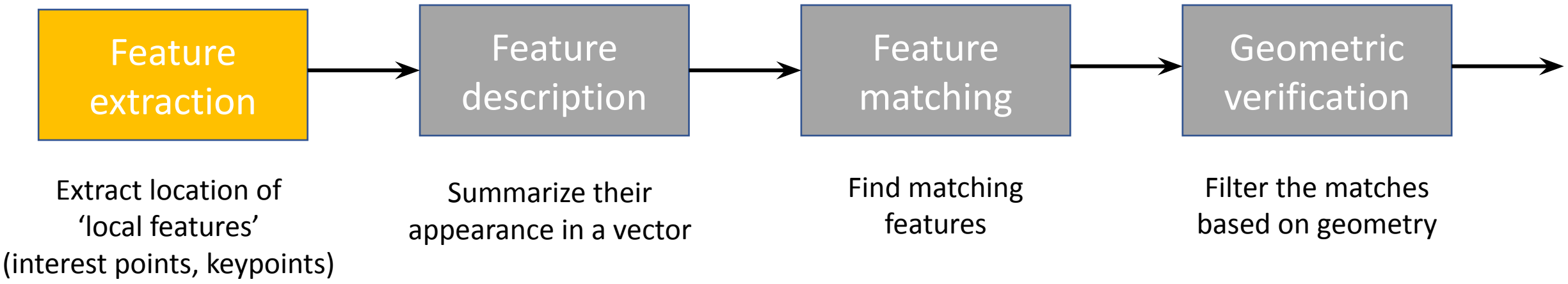
- Noise reduction
- Scale election
- Isotropic

Outline

1. Classical local features
   - Reminder on convolutions
   - Feature detection: How to extract informative features consistently? **Harris corners, Laplacian/Hessian blobs, scale and orientation**
   - Feature description: how to compare features?
   - Some more discussion of SIFT and SURF

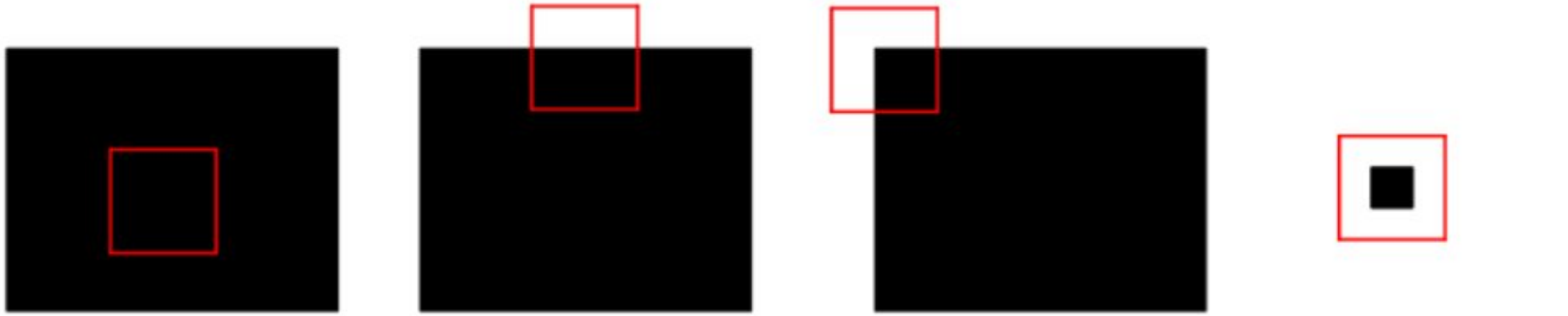
2. Deep local features
3. Some deep 3D reconstruction

Local features and correspondences pipeline for 3D reconstruction



Auto-correlation for corner detection (Moravec 1980)

- Corner?



A. Interior Region
Little intensity variation
in any direction

B. Edge
Little intensity variation
along edge, large
variation perpendicular
to edge

C. Edge
Large intensity variation
in all directions

D. Edge
Large intensity variation
in all directions

Harris Corner

Idea: compare a patch P and a patch shifted by (Δx,Δy)

$$S(\Delta_x, \Delta_y) = \sum_{x,y \in P} (I(x + \Delta_x, y + \Delta_y) - I(x, y))^2$$

-> if the difference is large for all (Δx,Δy) the patch is distinctive.

Harris Corner

Idea: compare a patch P and a patch shifted by (Δx,Δy)

$$S(\Delta_x, \Delta_y) = \sum_{x,y \in P} \left(I(x + \Delta_x, y + \Delta_y) - I(x,y)\right)^2$$

Use Taylor extension of *I* (Harris and Stephen 1988):

$$I(x + \Delta_x, y + \Delta_y) \simeq I(x,y) + I_x(x,y)\Delta_x + I_y(x,y)\Delta_y$$

$$S(\Delta_x, \Delta_y) \simeq \sum_{x,y \in P} \begin{bmatrix} \Delta_x & \Delta_y \end{bmatrix} \begin{bmatrix} I_x^2 & I_y I_x \\ I_x I_y & I_y^2 \end{bmatrix} \begin{bmatrix} \Delta_x \\ \Delta_y \end{bmatrix}$$

Harris Corner

- *S* large in all direction <-> condition on $A = \sum_{x,y \in P} \begin{bmatrix} I_x^2 & I_y I_x \\ I_x I_y & I_y^2 \end{bmatrix}$

Spectral theorem

We can diagonalize the any symmetric positively defined matrix M in an orthonormal basis $(e_1, \ ..., \ e_m)$ i.e. write

$$M = \sum_{i=1}^{m} \lambda_i e_i e_i^T \qquad\qquad Me_i = \lambda_i e_i$$

$\lambda_1 \geq ... \geq \lambda_m \geq 0$ are the eigenvalues

- If $u = \sum_{i=1}^{m} u_i e_i$

$$\min_i \lambda_i \|u\|^2 \leq u^T M u = \sum_{i=1}^{m} \lambda_i u_i^2 \leq \max_i \lambda_i \|u\|^2$$

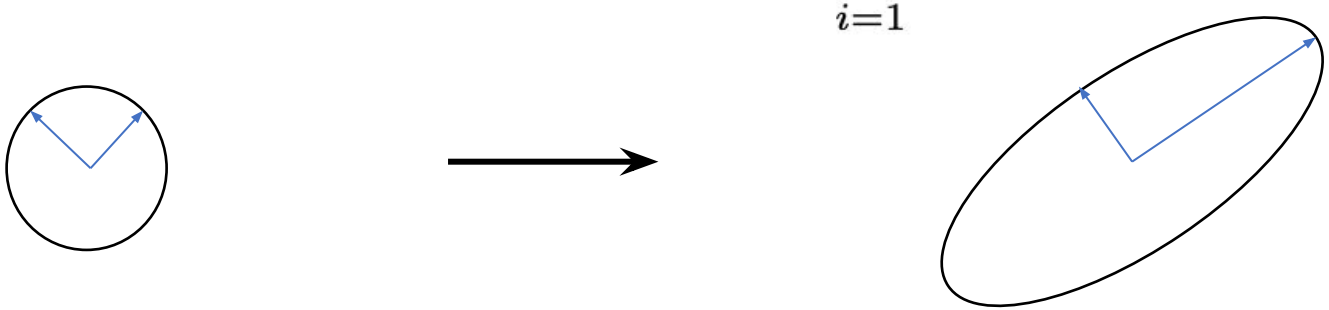
Spectral theorem

We can diagonalize the any symmetric positively defined matrix M in an orthonormal basis $(e_1, \ ..., \ e_m)$ i.e. write

$$M = \sum_{i=1}^{m} \lambda_i e_i e_i^T \qquad\qquad Me_i = \lambda_i e_i$$

$\lambda_1 \geq ... \geq \lambda_m \geq 0$ are the eigenvalues

Interpretation: level-sets of $u^T M u = \sum_{i=1}^{m} \lambda_i u_i^2$



Harris Corner

- *S* large in all direction <-> the two eigenvalues of are large

$$A = \sum_{x,y \in P} \begin{bmatrix} I_x^2 & I_y I_x \\ I_x I_y & I_y^2 \end{bmatrix}$$

Harris Corner

- *S* large in all direction <-> the two eigenvalues of are large

$$A = \sum_{x,y \in P} \begin{bmatrix} I_x^2 & I_y I_x \\ I_x I_y & I_y^2 \end{bmatrix}$$

- Harris and Stephens suggest to use

$$M_c = \lambda_1 \lambda_2 - \kappa \left(\lambda_1 + \lambda_2\right)^2 = \det(A) - \kappa \ \mathrm{trace}^2(A)$$

the *auto-correlation* or *second moment* matrix

w Gaussian -> invariant to in plane rotation

Harris Corner

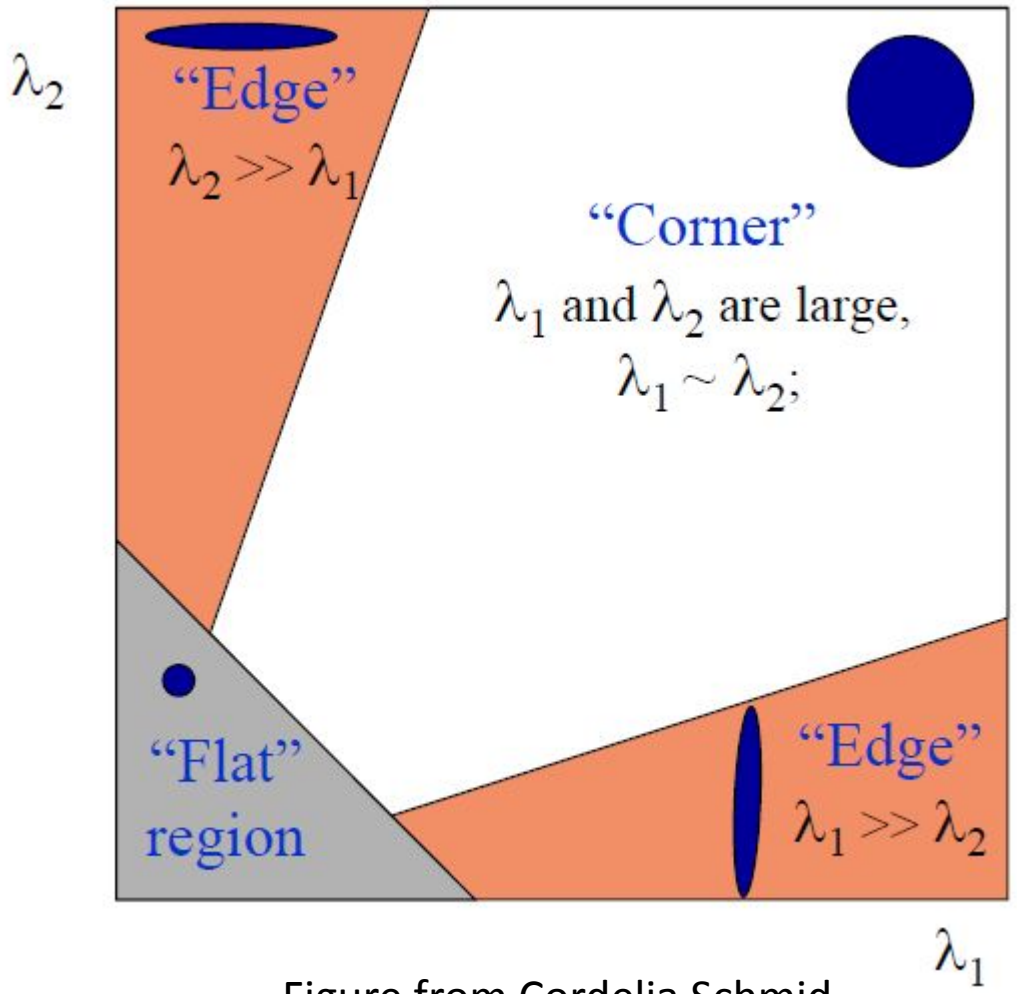


Figure from Cordelia Schmid

Harris Corner: algo

- Compute images derivatives $I_x(\boldsymbol{x}_q)$ and $I_y(\boldsymbol{x}_q)$ for each pixel q of I
  - compute smooth derivation operators G_x and G_y
  - compute "image derivatives" I_x and I_y: convolve I with masks G_x and G_y
- Compute "product images" $I_x^2, I_x I_y, I_y^2$ (not matrix products!)
  - then add extra smoothing using an "integration" Gaussian (e.g., $\sigma_i = 2$) (again using two 1D-convolutions rather than one 2D-convolution)
- Consider auto-correlation matrix $\boldsymbol{A} = [I_x^2 \; I_x I_y; \; I_x I_y \; \; I_y^2]$
  - compute corner response (or strength) for each q
  - response above threshold and local maximum (8 neighbors) → detection
  - possibly: only keep locally significant responses (see ANMS below)

Blob detection

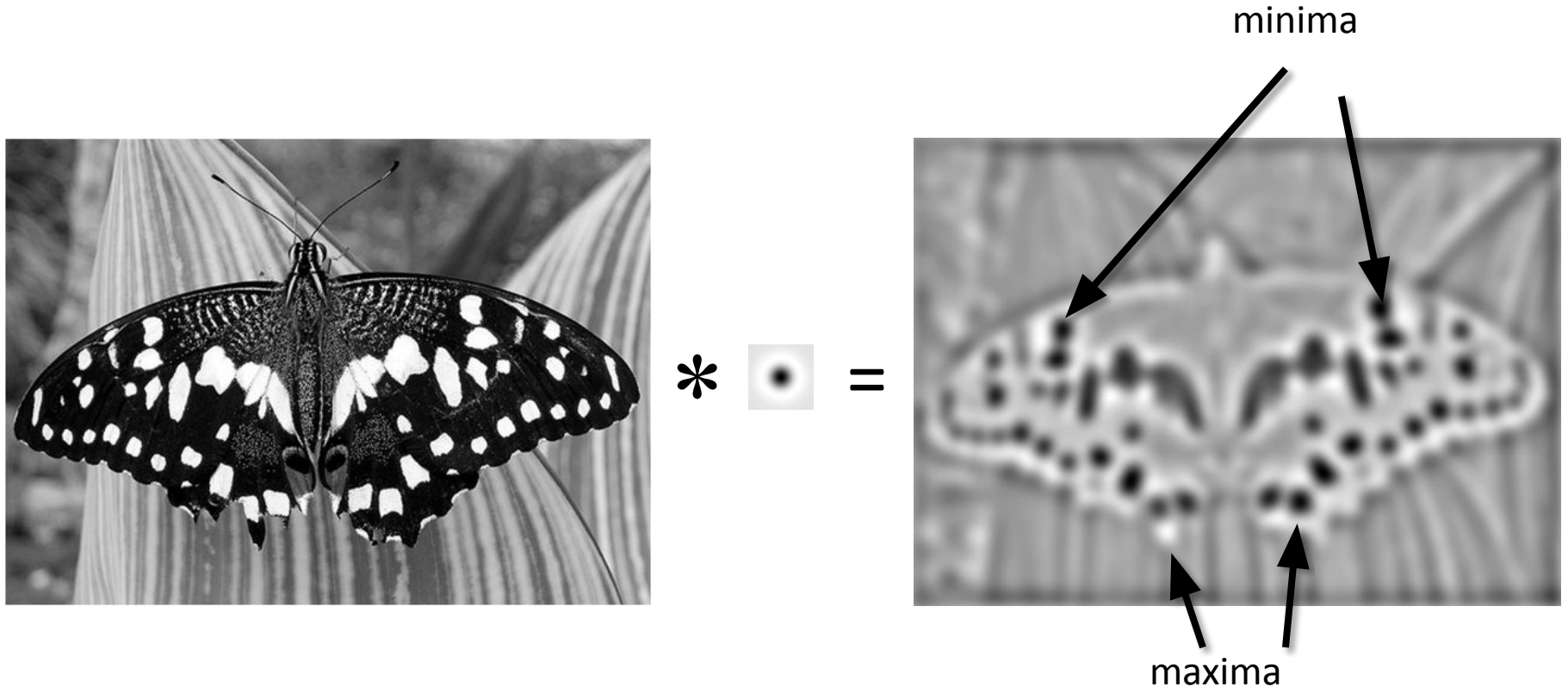
Blob detection: Hessian

Idea: do a quadratic approximation of the image
Use the Hessian: its eigen-values/vector give principal curvatures of the image

$$H = \sum_{x,y \in P} \begin{bmatrix} I_{x,x} & I_{x,y} \\ I_{x,y} & I_{y,y} \end{bmatrix}$$

$$I(x + \Delta_x, y + \Delta_y) \simeq L(x, y) + \nabla L(x, y)^\intercal \begin{bmatrix} \Delta_x \\ \Delta_y \end{bmatrix} + \begin{bmatrix} \Delta_x & \Delta_y \end{bmatrix} H(x, y) \begin{bmatrix} \Delta_x \\ \Delta_y \end{bmatrix}$$

Blob detection: LoG idea



- Find maxima *and minima* of blob filter response in space *and scale*

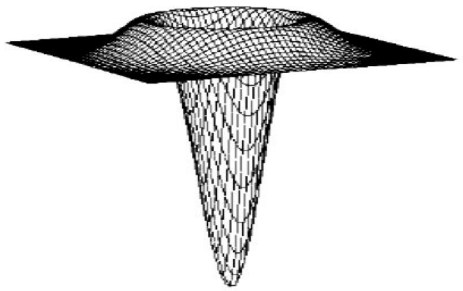
Source: N. Snavely

Blob detection: LoG

- Idea: convolve image with Laplacian of Gaussian and look for extrema

$$\Delta f = \nabla^2 f = \nabla \cdot \nabla f = \sum_{i=1}^{n} \frac{\partial^2 f}{\partial x_i^2} = \operatorname{tr}(H(f))$$

$$\Delta G(x, y; \sigma) = -\frac{1}{\pi\sigma^4}\left[1 - \frac{x^2 + y^2}{2\sigma^2}\right] e^{-\frac{x^2+y^2}{2\sigma^2}}$$



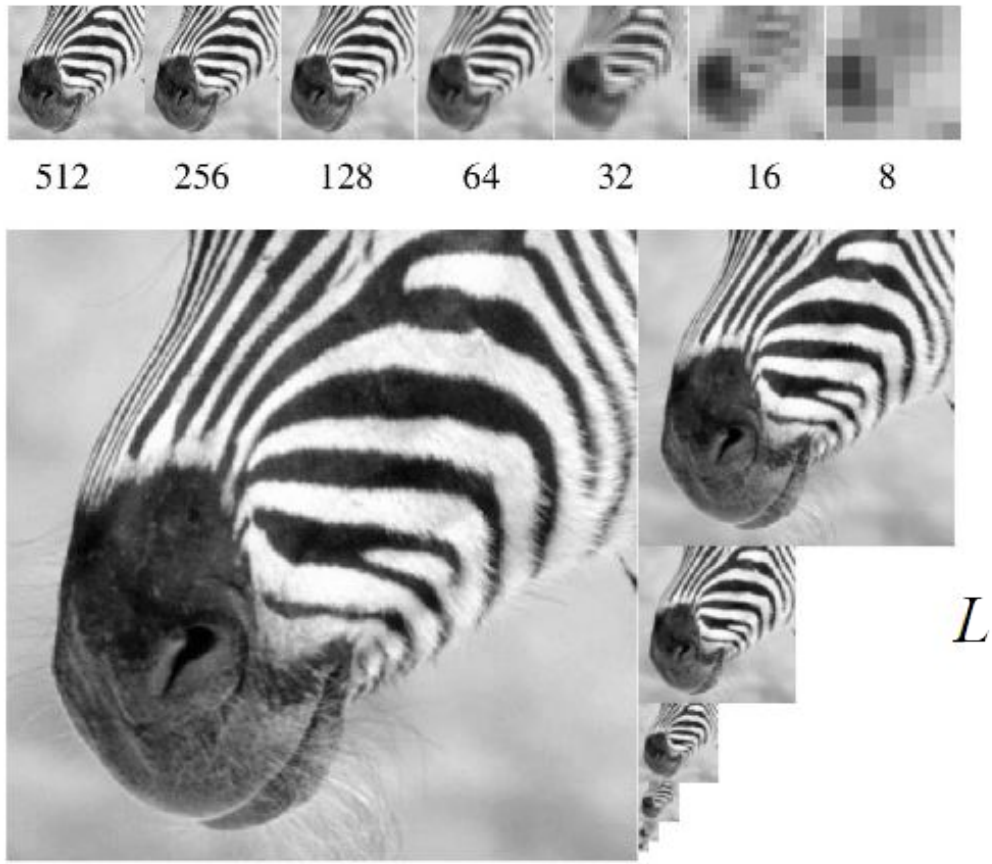
- Laplacian of Gaussian (LoG) detector score:

$$(\Delta G) * I = \Delta(G * I) = \Delta L$$

with $L(x, y; \sigma) = G(x, y; \sigma) * I(x, y)$

Multi-scale

Gaussian pyramid



- Convolution with Gaussian of varying σ

$$G(x, y; \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

- Scale-space representation

$$L(x, y; \sigma) = G(x, y; \sigma) * I(x, y)$$

- Scale pyramid

Space: x, y dimensions (location)
Scale-space: σ dimension

Scale-space

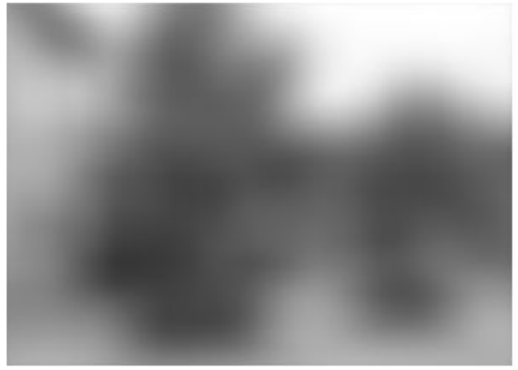
σ = 0 (original image)

σ = 1

σ = 4

σ = 16

σ = 64

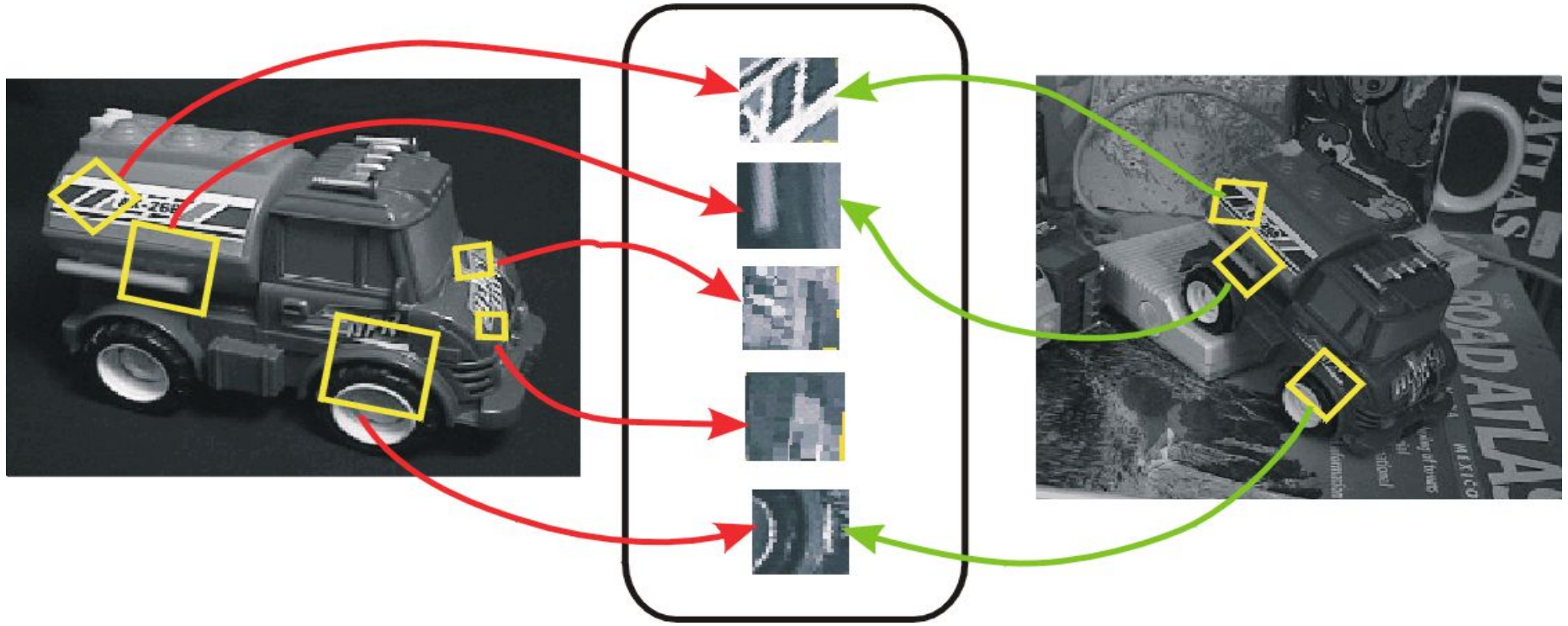


σ = 256

Be carefull when comparing detector responses at different scales, you might need a scaling factor!

e.g. scale normalized LoG $\nabla^2_{\text{norm}} G = \sigma^2 \nabla^2 G$

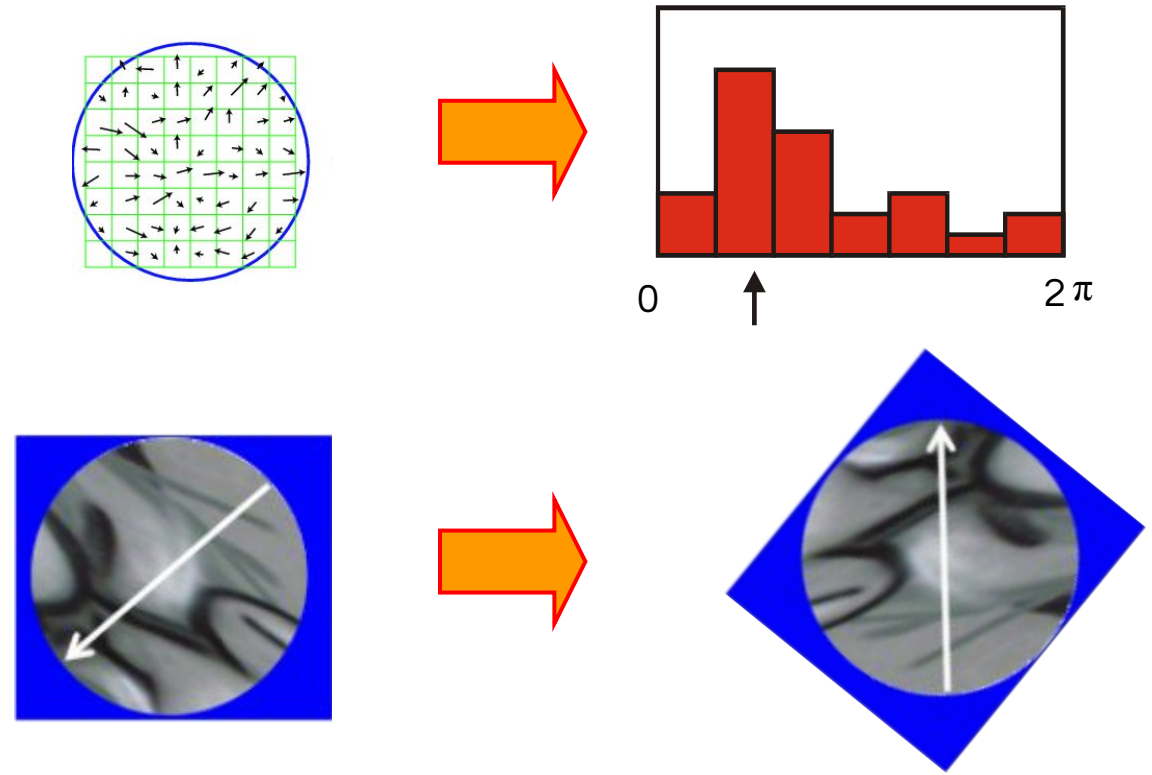
Covariance / Invariance



- We want the description to be *invariant*:
  features(transform(image)) = features(image)
- A solution is to have a *covariant* detection:
  det(transform(image)) = transform(det(image))

Rotation invariance/covariance

- example: use the dominant gradient direction to rotate the image



Affine invariance/covariance

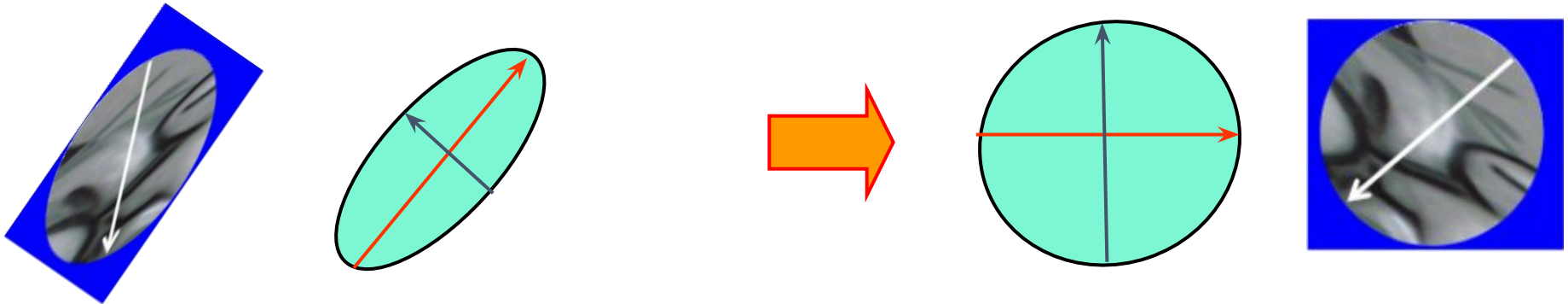
- example: use the eigen-decomposition of the second moment matrix

$$M = \begin{pmatrix} L_x^2 & L_x L_y \\ L_x L_y & L_y^2 \end{pmatrix}$$

  - Give direction of maximum and minimum variation of the image and a characteristic scale

-> Normalize the image

$$x' \rightarrow M^{\frac{1}{2}} x$$



Evaluation

- The main criteria for a detector is repeatability, i.e. to detect the same features in two different views of the same scene.
- Another criteria is the number of features detected/image
- ”Same” can mean different things depending on the feature type (location, scale, orientation…)

Non-Max suppression

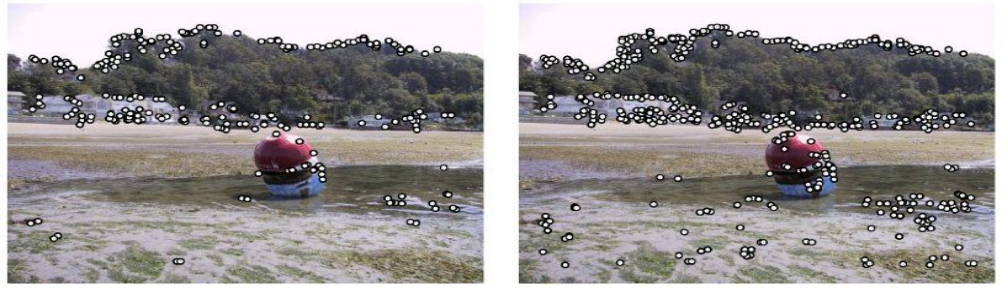
Non-Maximum Suppression

- Problem:

  maximality in 3x3 neighborhood

  → uneven distribution
  (dense where high contrast)

  → poor robustness
  (sensitive to noise)



(a) Strongest 250 (b) Strongest 500

- Solution 1 (NMS):
  - check in larger region around p (e.g., disk of given radius r):

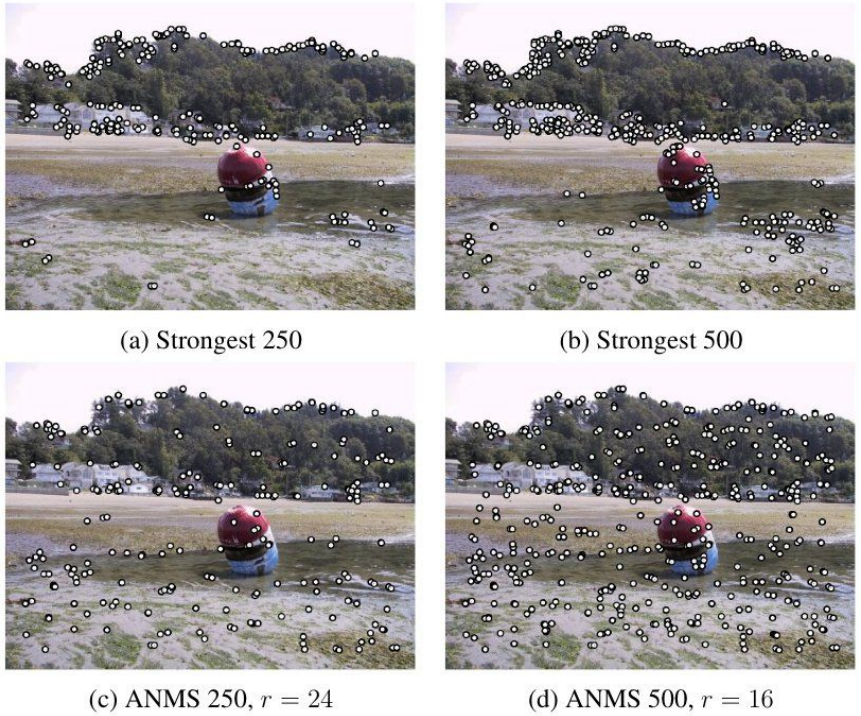
    check maximality w.r.t. all points q such that $|| \boldsymbol{x}_p - \boldsymbol{x}_q || \leq r$

  - check almost largest response (e.g., within 10%):

    $\forall q \;\; || \boldsymbol{x}_p - \boldsymbol{x}_q || \leq r \;\; \Rightarrow \;\; c\, f(\boldsymbol{x}_q) \leq f(\boldsymbol{x}_p)$ [e.g., $c = 0.9$]

Adaptive non-maximal suppression (ANMS)

- Problem with NMS
  - Distribution still uneven
  - Need to tune r
- Solution 2: ANMS
  - Compute a radius for each point
  - Sort by radius



(a) Strongest 250 (b) Strongest 500

(c) ANMS 250, $r = 24$ (d) ANMS 500, $r = 16$

ANMS : algo

Sort $DetectedPoints$ by decreasing response

$p_1 \leftarrow$ point with highest response

$r_{p_1} \leftarrow +\infty$

$ProcessedPoints \leftarrow \{p_1\}$, and remove p_1 from $DetectedPoints$

For each detection $p \in DetectedPoints$, in decreasing strength order

$r_p \leftarrow \min_{q \in ProcessedPoints} \| \boldsymbol{x}_p - \boldsymbol{x}_q \|$ such that $f(\boldsymbol{x}_p) < c\, f(\boldsymbol{x}_q)$

[as $f(\boldsymbol{x}_p) > c\, f(\boldsymbol{x}_q)$ guaranteed for $q \notin ProcessedPoints$]

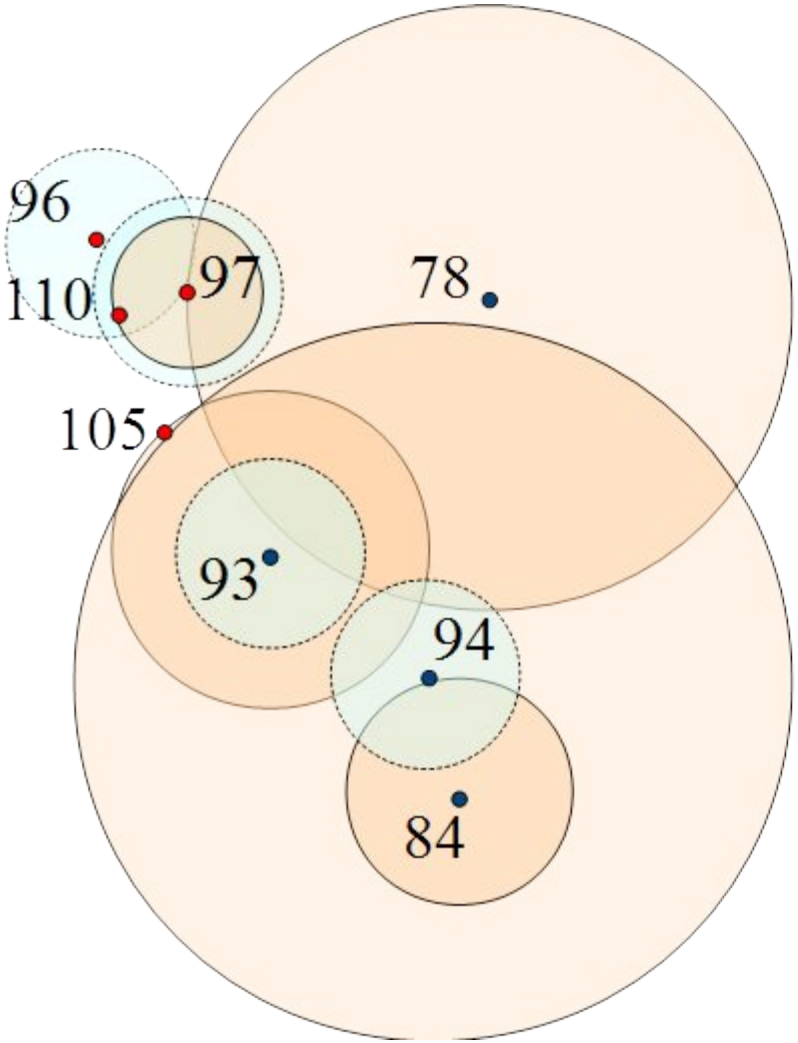
add p to $ProcessedPoints$

Return n first points p with the highest suppression radius r_p

// Still quadratic in number of points. (But there are subquadratic algorithms.)

// Compute, store and compare r^2 rather than r to avoid computing a square root for $r = \| \boldsymbol{x}_p - \boldsymbol{x}_q \|$

ANMS : example

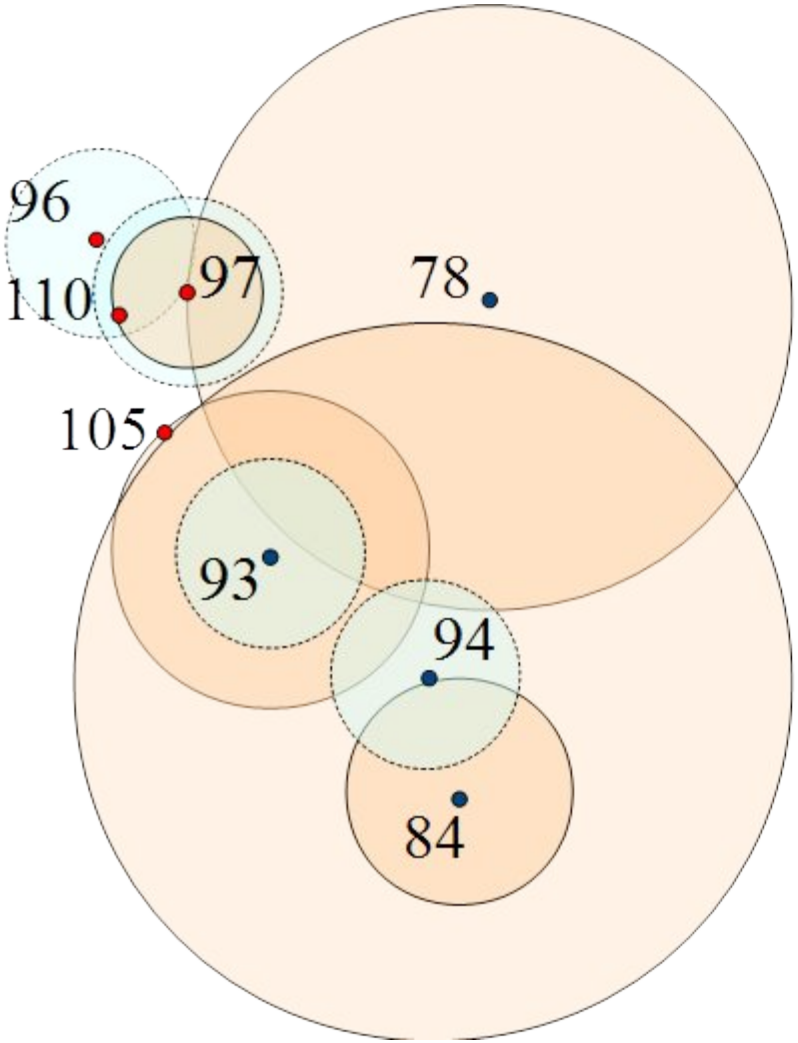


4 strongest points:

4 strongest points with NMS:

4 strongest points with ANMS:

ANMS : example



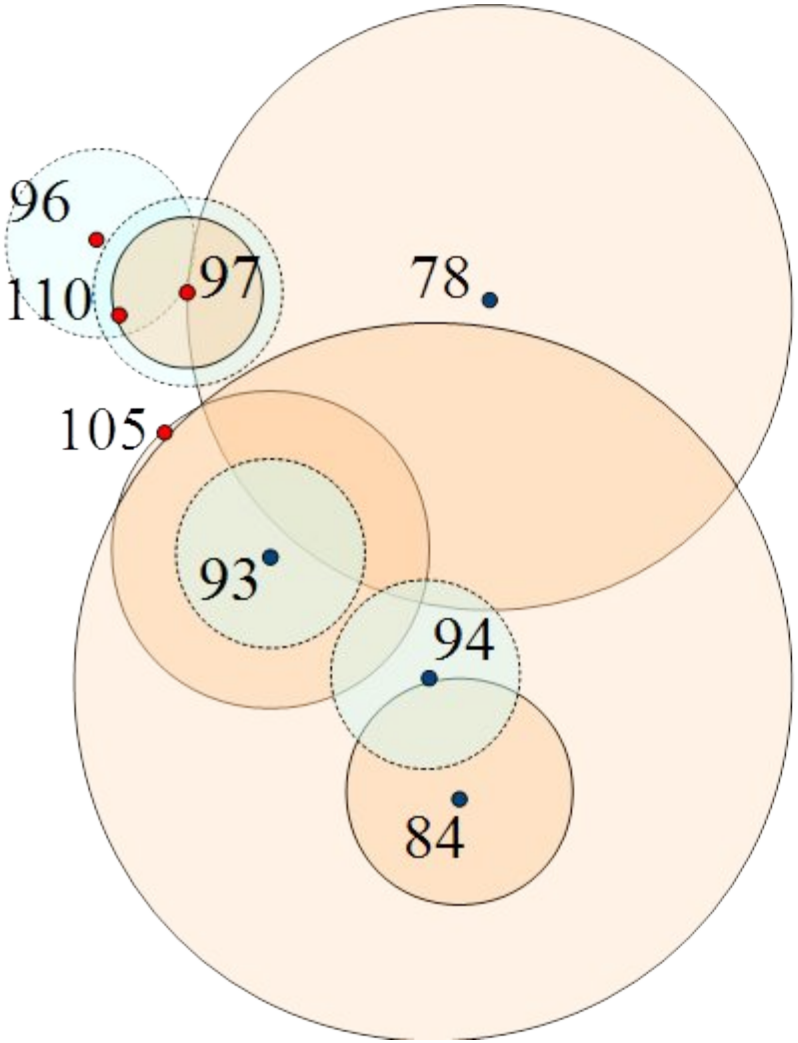
4 strongest points:

110, 105, 97, 96

4 strongest points with NMS:

4 strongest points with ANMS:

ANMS : example



4 strongest points:

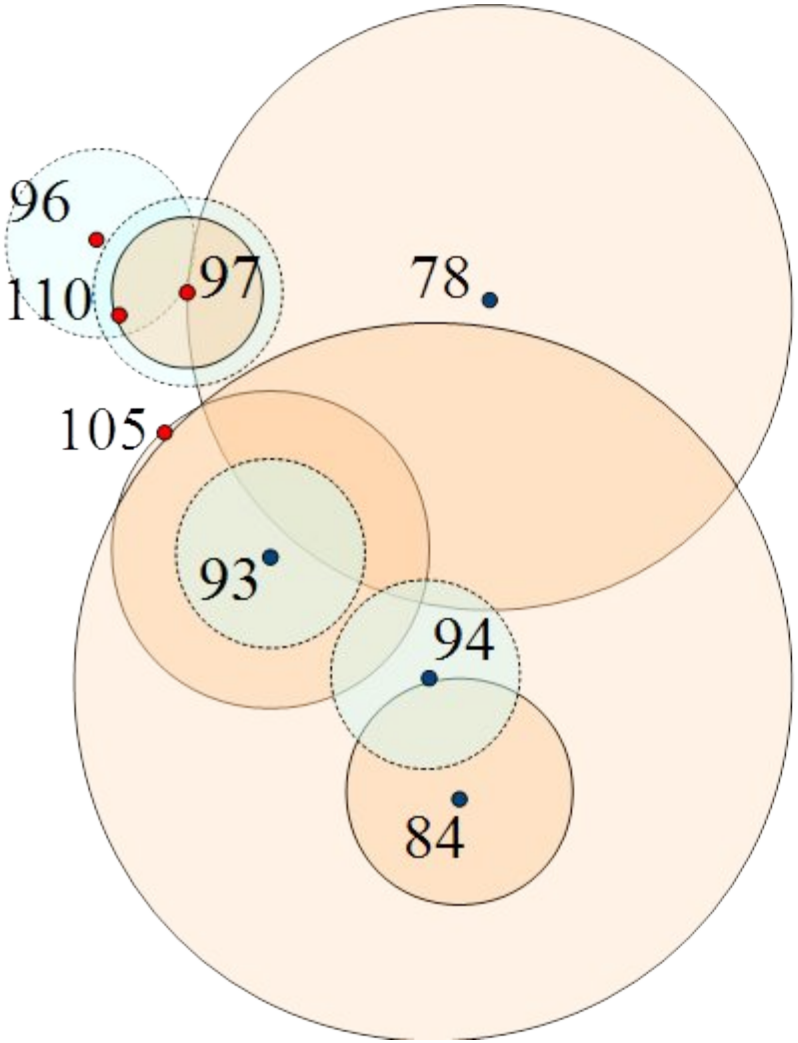
110, 105, 97, 96

4 strongest points with NMS:

110, 105, 94, 93

4 strongest points with ANMS:

ANMS : example



4 strongest points:

110, 105, 97, 96

4 strongest points with NMS:

110, 105, 94, 93

4 strongest points with ANMS:

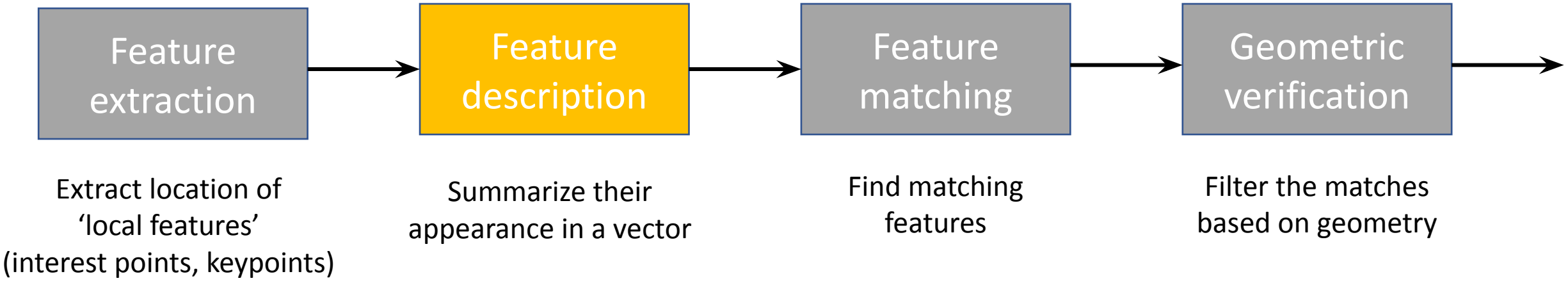
110, 105, 94, 78

Outline

1. Classical local features
   - Reminder on convolutions
   - Feature detection: How to extract informative features consistently?
   - Feature description: how to compare features? **ShapeContext/HOG/BRIEF**
   - Some more discussion of SIFT and SURF

2. Deep local features
3. Some deep 3D reconstruction

Local features and correspondences pipeline for 3D reconstruction



Feature descriptors

- How to compare patches?
  - Directly compare pixels
  - Look at a more meaningful embedding (descriptor)

- Which distance/similarity?

Feature descriptors

Many type of descriptors

- Pixel values
- Based on local image statistics (local derivatives, answer to filters…)
- Based on local histograms
- Binary comparisons
- CNN-based
- …

Comparing patches using pixel values

Two square patches P_0 and P_1 of size w

- L2 distance:

$$\sum_{i,j}(P_0(i,j) - P_1(i,j))^2 = \sum_{i,j}(P_0(i,j)^2 + P_1(i,j)^2) - 2\sum_{i,j} P_0(i,j).P_1(i,j)$$

Sensitive to illumination changes – average luminosity of the patch

Comparing patches using pixel values

Two square patches P_0 and P_1 of size w

- Zero-mean Normalized Cross-correlation (ZNCC)

$$\frac{1}{w}\sum_{i,j}\frac{P_0(i,j)-\mu_0}{\sigma_0}\cdot\frac{P_1(i,j)-\mu_1}{\sigma_1}$$

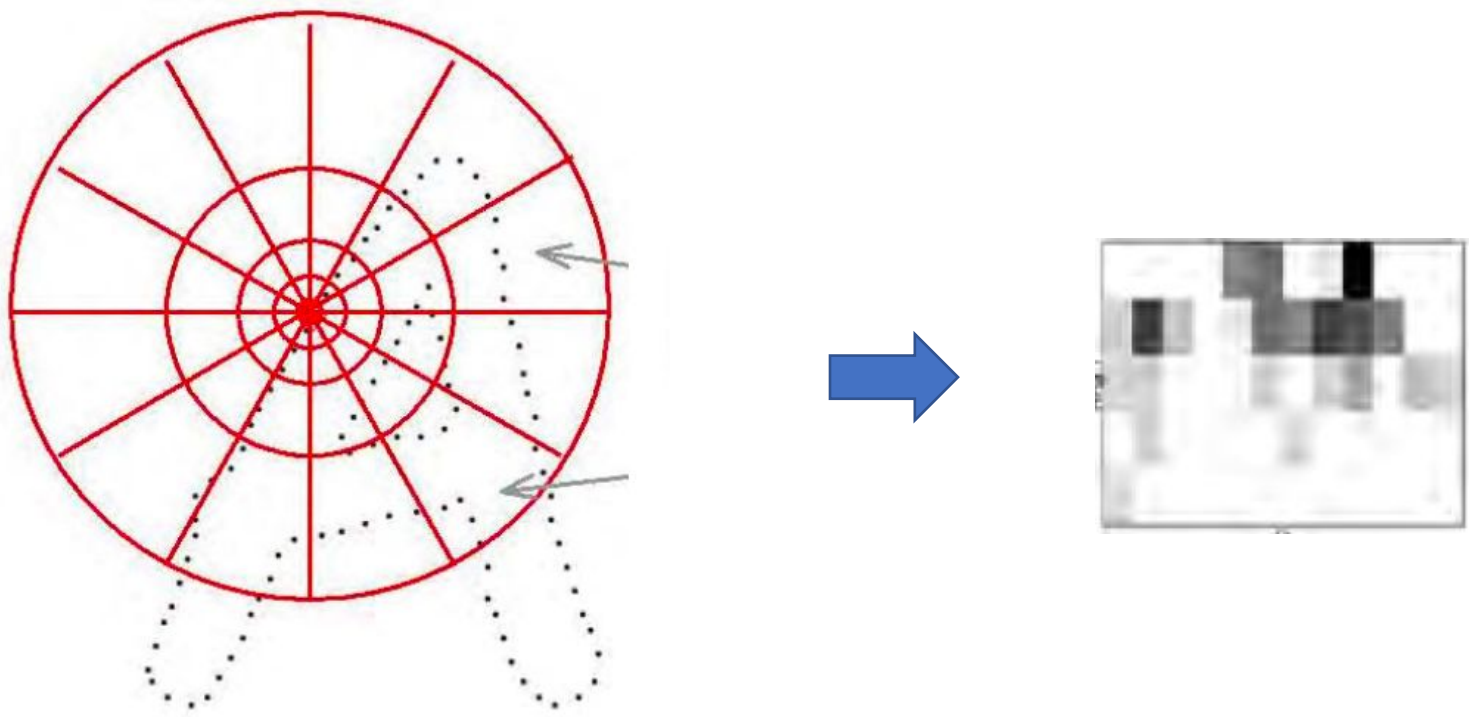
$$\mu_k=\frac{1}{w}\sum_{i,j}P_k(i,j) \qquad \sigma_k^2=\frac{1}{w}\sum_{i,j}(P_k(i,j)-\mu_k)^2$$

Invariant to affine illumination changes, robust to noise.

-> Problem: still limited robustness

Shape context

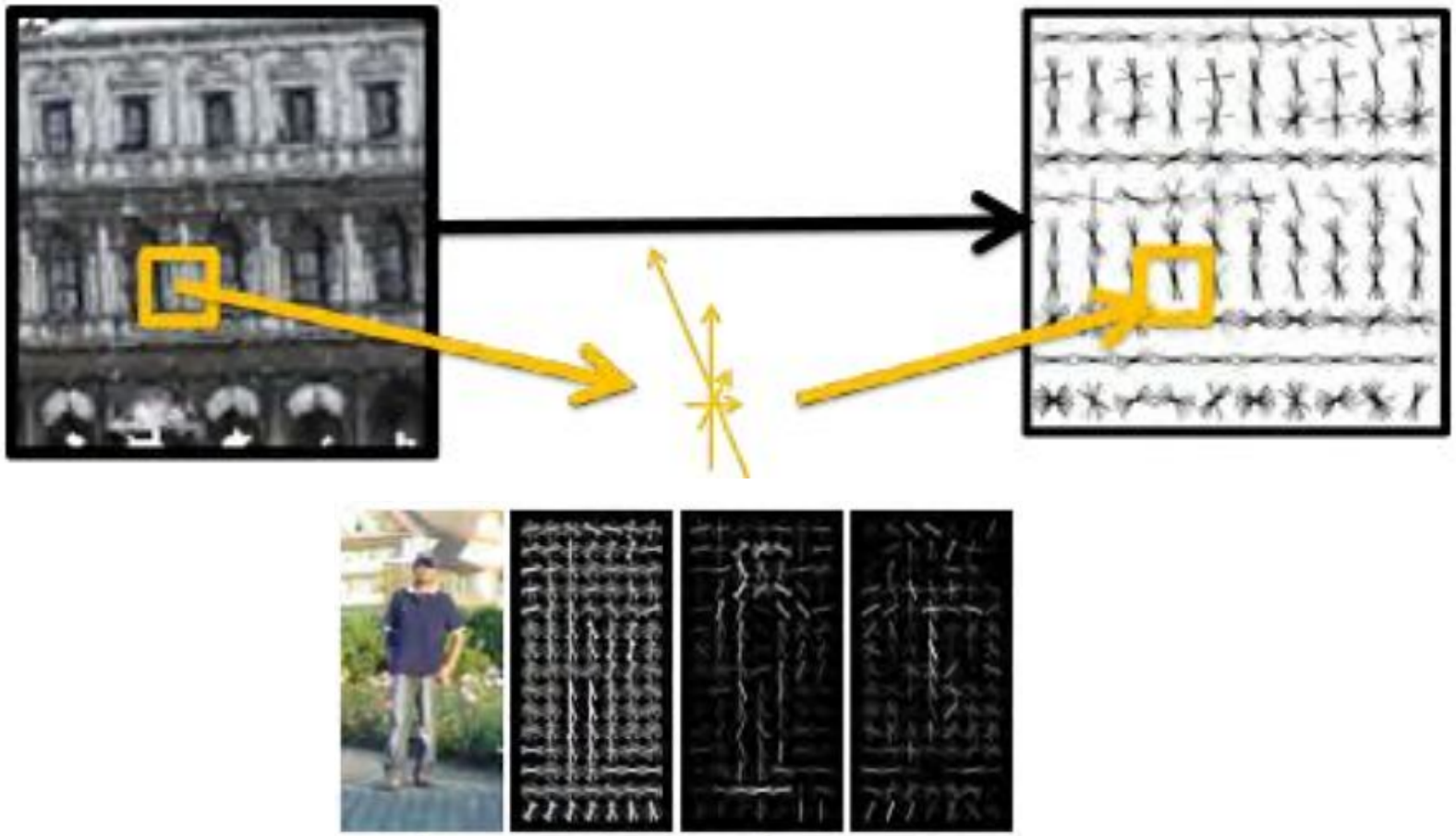
1. Detect edges ; 2. Sample points ; 3. Build histogram



Belongie, S., Malik, J., & Puzicha, J.
Shape matching and object recognition using shape contexts. *PAMI 2002*

Histograms of Oriented Gradients

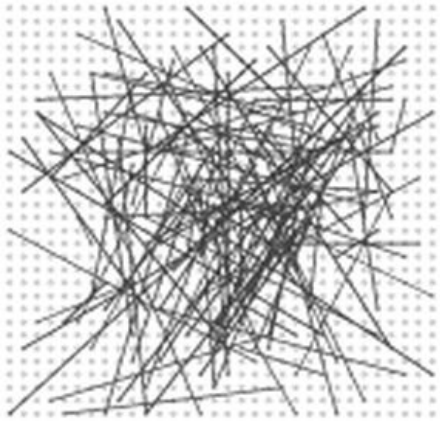
Same idea as SIFT (coming in a few slides): histogram of gradients orientations



Dalal, N., & Triggs, B. Histograms of oriented gradients for human detection. *CVPR 2005*.

BRIEF

Using binary comparisons between random locations



$$\tau(\mathbf{p};\mathbf{x},\mathbf{y}) := \begin{cases} 1 & \text{if } \mathbf{p}(\mathbf{x}) < \mathbf{p}(\mathbf{y}) \\ 0 & \text{otherwise} \end{cases}$$

$$f_{n_d}(\mathbf{p}) := \sum_{1 \le i \le n_d} 2^{i-1}\, \tau(\mathbf{p};\mathbf{x}_i,\mathbf{y}_i)$$

Calonder, M., Lepetit, V., Strecha, C., & Fua, P. Brief: Binary robust independent elementary features. *ECCV 2010*

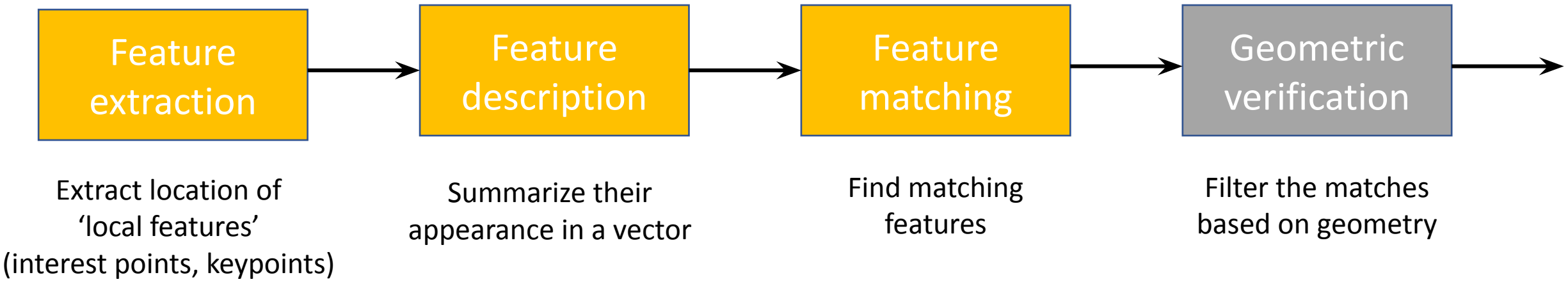
See also Local Binary Patterns (LBP)

Outline

1. Classical local features
   - Reminder on convolutions
   - Feature detection: How to extract informative features consistently?
   - Feature description: how to compare features?
   - **Some more discussion of SIFT and SURF**

2. Deep local features
3. Some deep 3D reconstruction

Local features and correspondences pipeline for 3D reconstruction



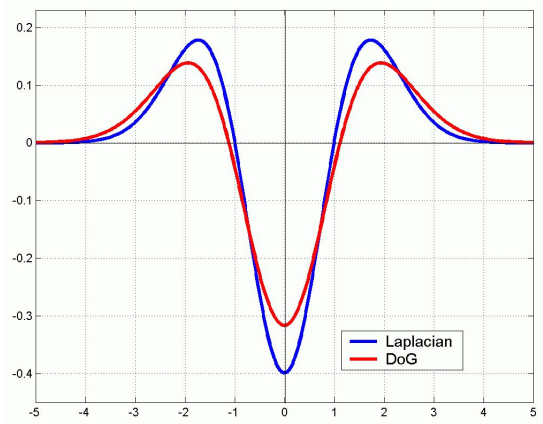
SIFT

- Scale Invariant Feature Transform,
  David Lowe, ICCV 1999, IJCV2004
- Detector + descriptor
- Optimized for speed and precision, designed using performance over synthetic transformations (rotation, scaling, affine stretch, change in brightness and contrast, and addition of image noise)
- Still the main baseline for sparse features, even if deep methods now lead to better detectors/descriptors

SIFT Detector

- Approximating LoG with DoG

$$\partial G / \partial \sigma = \sigma \nabla^2 G$$



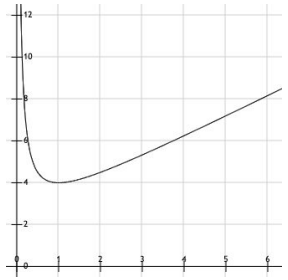
- Localization in the 3D scale space beyond discretization

$$\tilde{D}(\boldsymbol{x}) = D + \frac{\partial D^T}{\partial \boldsymbol{x}} \boldsymbol{x} + \frac{1}{2} \boldsymbol{x}^T \frac{\partial^2 D}{\partial \boldsymbol{x}^2} \boldsymbol{x}$$

$$\frac{\partial \tilde{D}}{\partial \boldsymbol{x}}(\hat{\boldsymbol{x}}) = 0 \Leftrightarrow \hat{x} = -\frac{\partial^2 D^{-1}}{\partial x^2} \frac{\partial D}{\partial x}$$

- Rejection of unstable keypoints with low contrast / edges

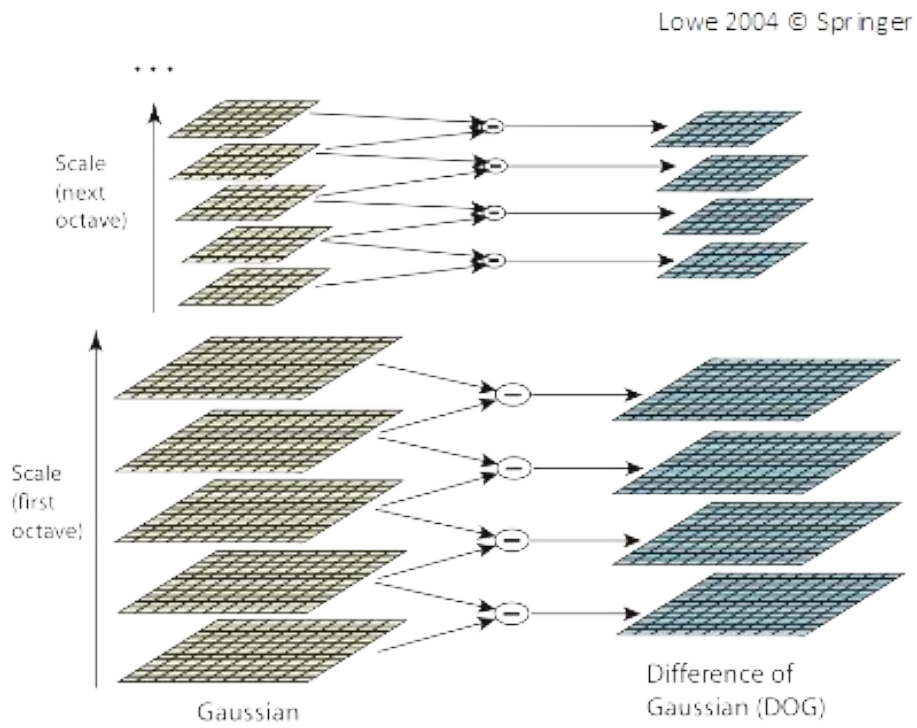
$$\frac{\mathrm{trace}(\boldsymbol{H})^2}{\det(\boldsymbol{H})} = \frac{(\lambda_0 + \lambda_1)^2}{\lambda_0 \lambda_1} = \frac{(r+1)^2}{r} \qquad r = \lambda_1 / \lambda_0$$



- Orientation assignment from local gradient

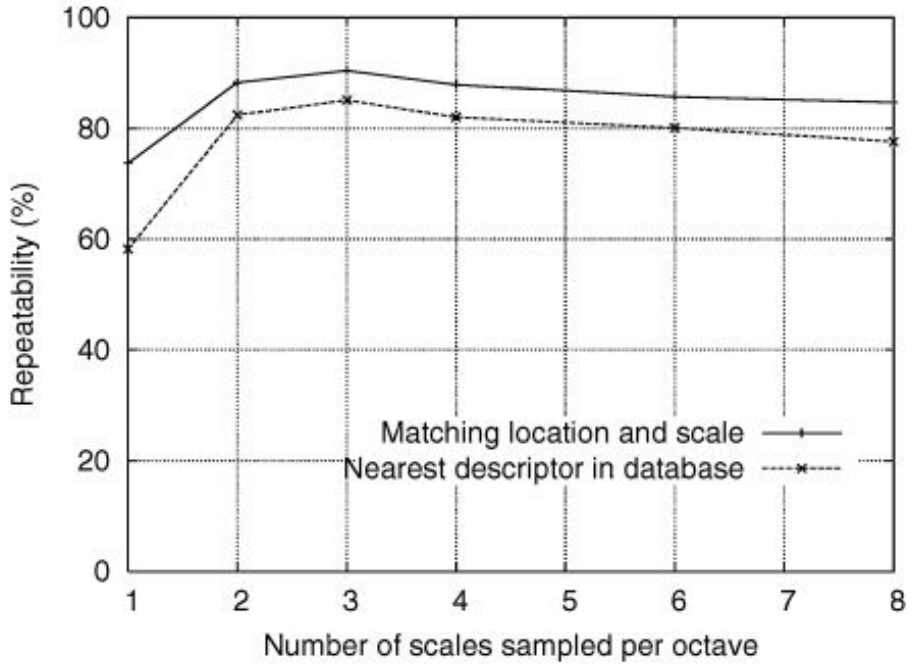
Efficient computation of scale-space: DoG

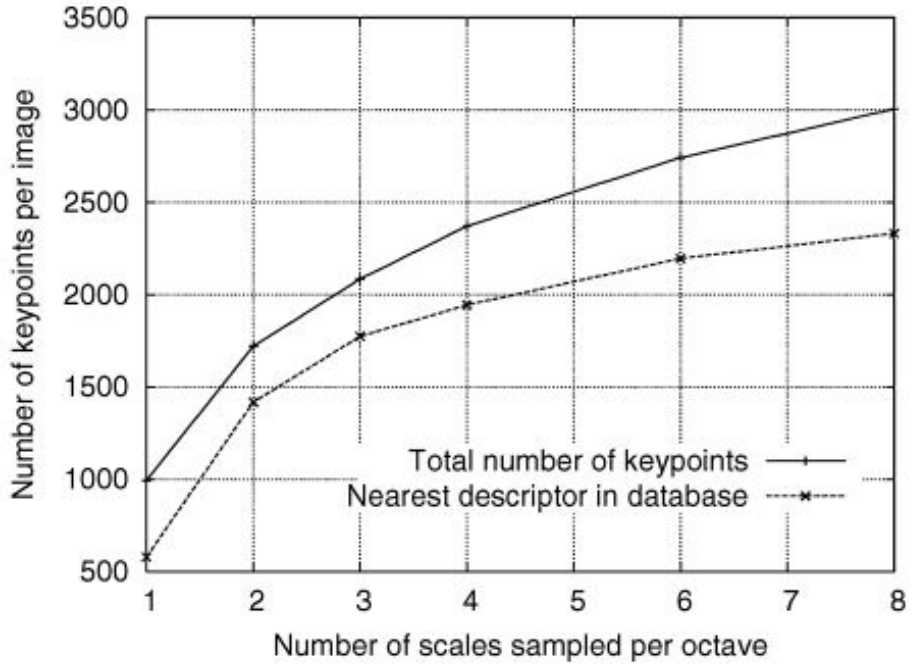
- Geometric progression of scales with ratio $k = 2^{1/s}$
- Successive convolutions
- At each octave (i.e., every sample $s \rightarrow$ scale factor of 2), resample image
  - every second pixel in each row and column
  - no accuracy loss
  - space & time efficient



Gaussian and DoG pyramids

Scale discretization parameter





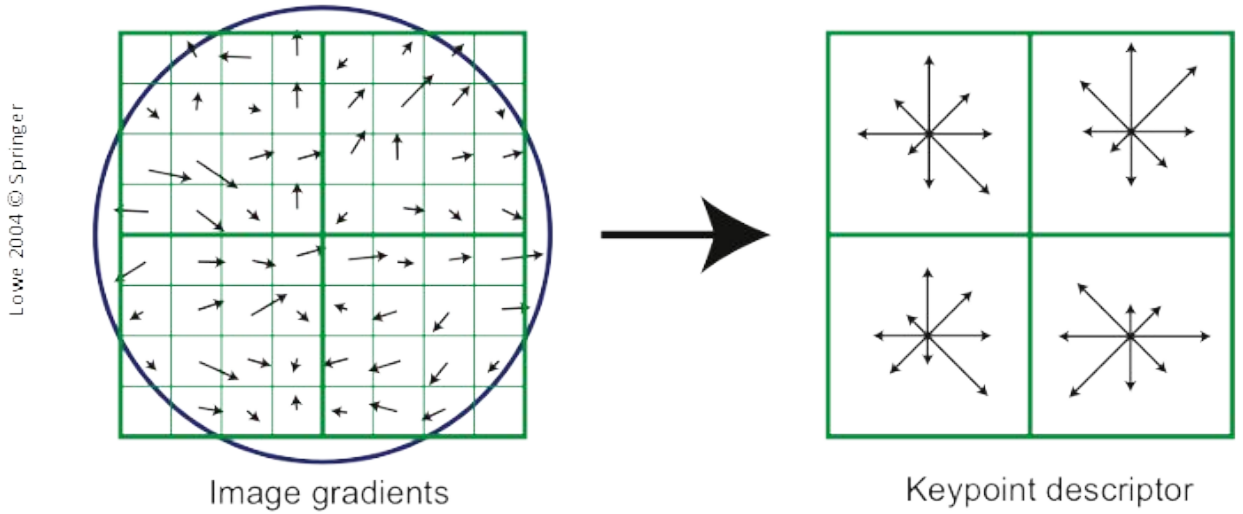
Similar experiments for every SIFT parameter -> a huge engineering paper

SIFT descriptor

For a given keypoint at a given scale

- resize the region to NK x NK
- split it in a KxK grid of cells of NxN pixels
- Build a weighted histogram of gradient orientations in M directions

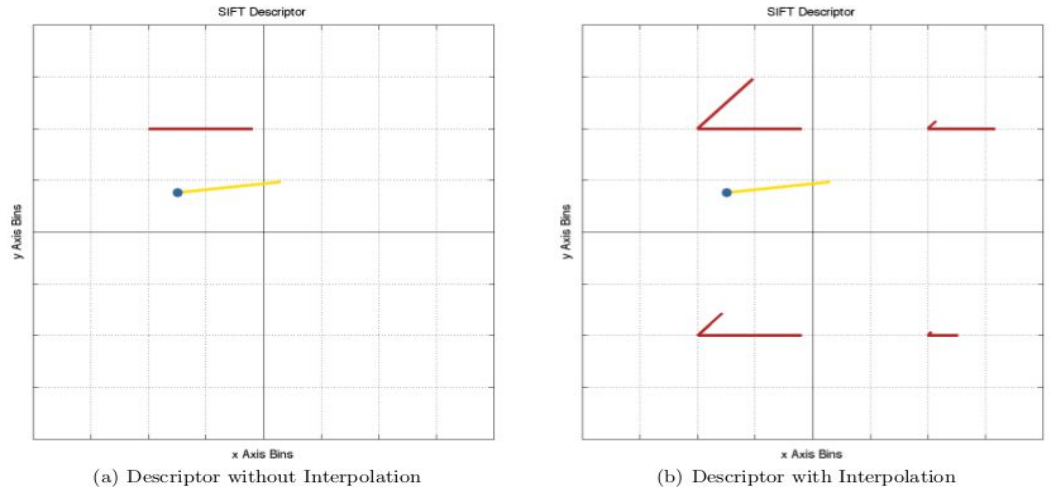
Typically, N=K=4, M=8
[shown: 2x2 grid of 4x4-pixel cells]



SIFT descriptor

Lots of tricks again:

- Gaussian weight on gradient magnitude
- Histogram smoothing
- Normalization, thresholding of gradient, renormalization



(a) Descriptor without Interpolation (b) Descriptor with Interpolation

Figure 5: The effect of interpolation when generating the SIFT descriptor. *A single pixel's gradient*

SIFT matching

- Measure of similarity between descriptors
  - Euclidean distance

- First to second nearest neighbor ratio test (to reduce nb of outliers), keep match only if ratio < 0.8

Does:
- orange match blue?
- Red match green?

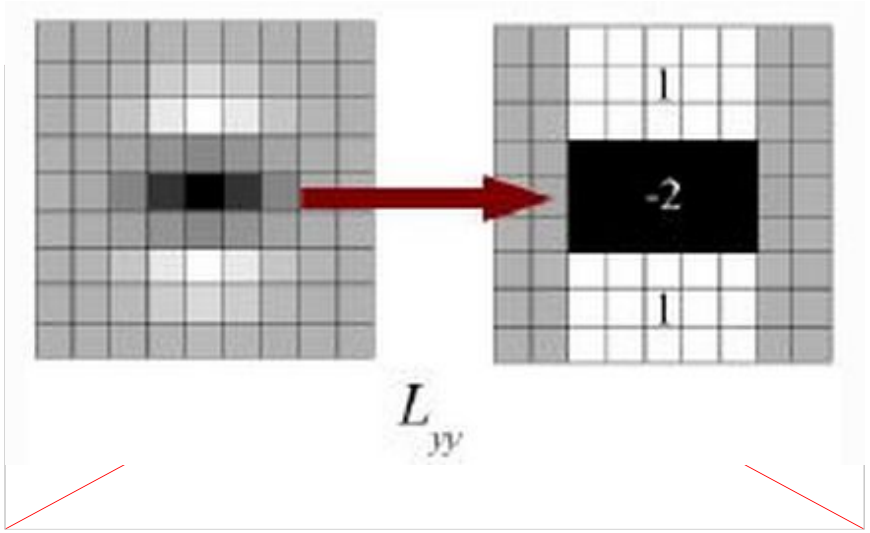
SIFT

- Tons of parameters (sizes, thresholds, etc.)
  - "good" parameters found by experimentation
    - ☛ possible bias towards used image database
- Many tiny details, some unsaid at all

☛ Most likely no 2 implementations give the same result

SURF

- Inspired by SIFT
- Faster, using approximations and integral images



Bay, H., Tuytelaars, T., & Van Gool, Surf: Speeded up robust features, *ECCV 2006*

Classical local features: summary

- Detectors: Harris, blob
- Description: ZNCC, shape context, HoG
- SIFT: detector + descriptor, 1st/2nd NN ratio test
- Evaluation/parameter tuning
- Lots of small but important and very general idea:
  - 1st to 2nd NN ratio, soft assignment, Taylor expension / spectral decomposition, scale space, invariance/covariance…

Outline

1. Classical local features
2. Deep local features
   - learning patch descriptors
   - learning dense detectors and descriptors
   - learning feature matching
3. Some deep 3D reconstruction

Disclaimer: not a full litterature review!

There are tons of paper over the last 2-5 years,
this is a biaised selection to illustrate idea I think I worth knowing

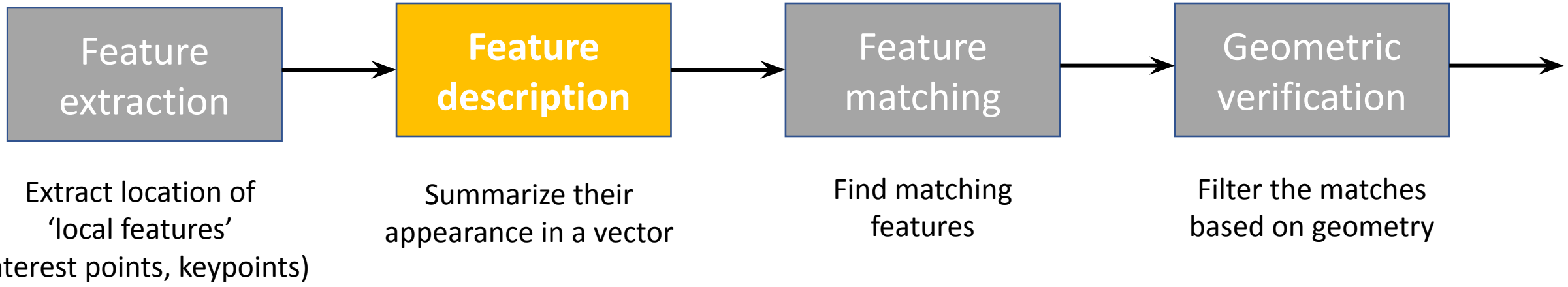
Learning features and correspondences

1. Learning feature detectors and descriptors
    1. Patch-based
    2. Dense
2. Learning image matching
    1. Coarse Flow
    2. Fine flow
    3. MVS
3. Learning to filter correspondences

Disclaimer: not a full litterature review!

There are tons of paper over the last 2-5 years,
this is a biaised selection to illustrate idea I think I worth knowing

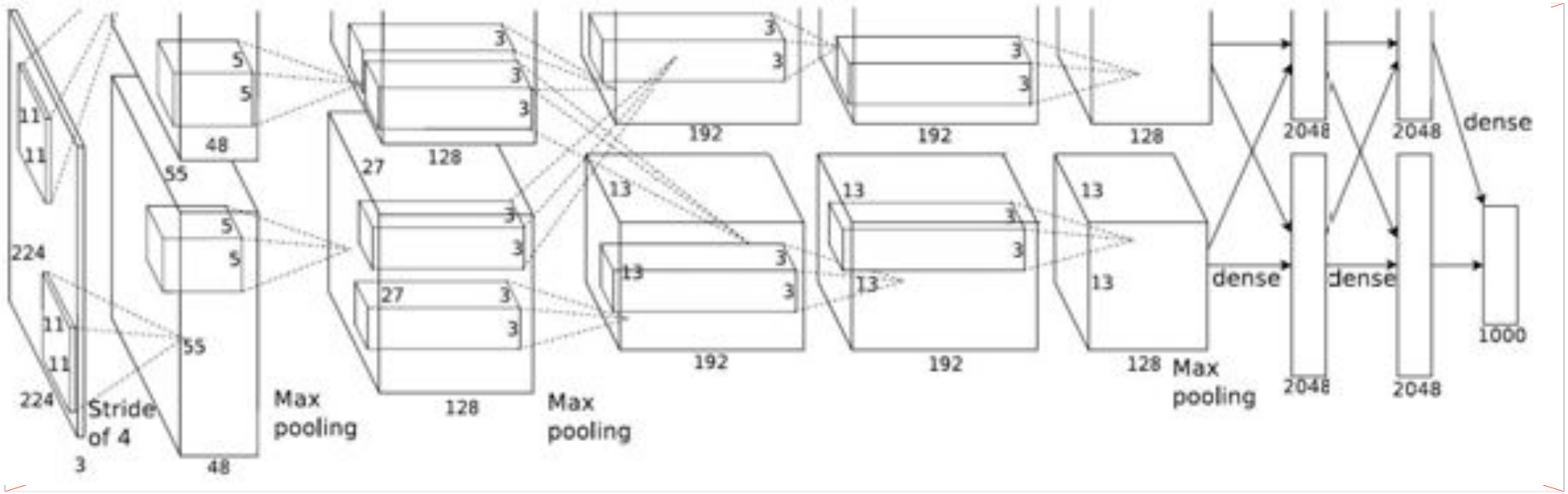
Local features and correspondences pipeline for 3D reconstruction



CNNs/Deep features

Standard CNNs (eg. AlexNet):

- Succession of convolutions, non linearities (ReLu) and max-poolings
- Trained for image classification (1 million images from ImageNet)



Fischer, P., Dosovitskiy, A., & Brox, T. , Descriptor matching with convolutional neural networks: a comparison to sift. *arXiv preprint 2014*

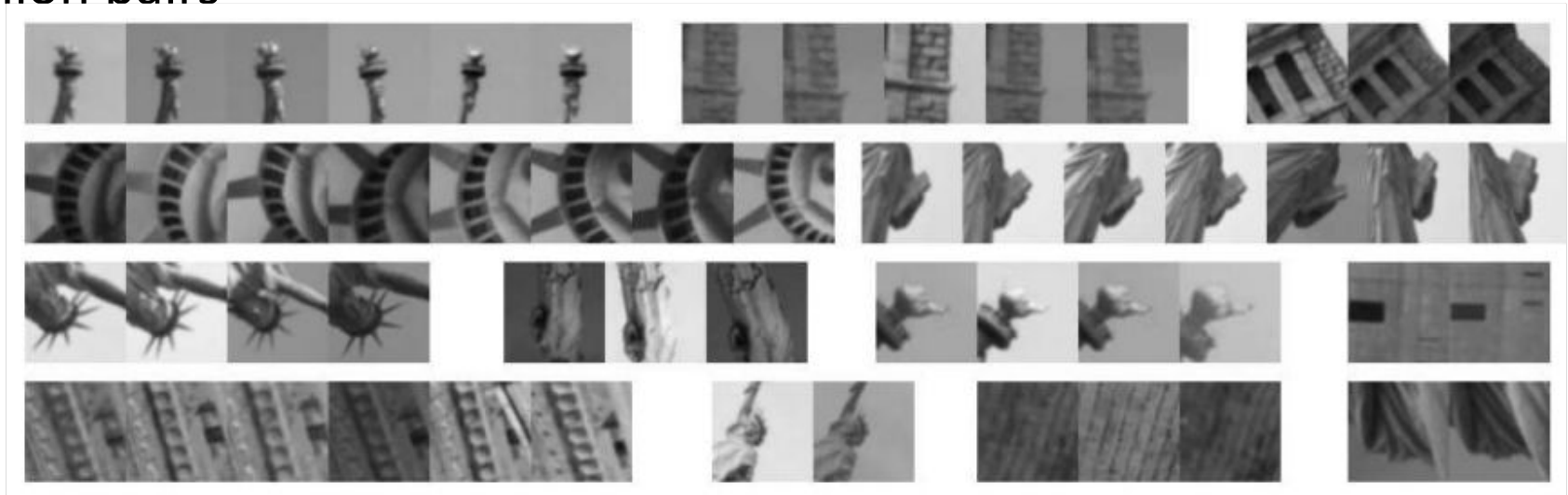
Conv 4 features seem generic and outperform SIFTs

Patch descriptor learning

Idea: create a large database of ground truth local feature matches using the 3D of reconstructed scenes and use it to learn the parameter of a descriptor.

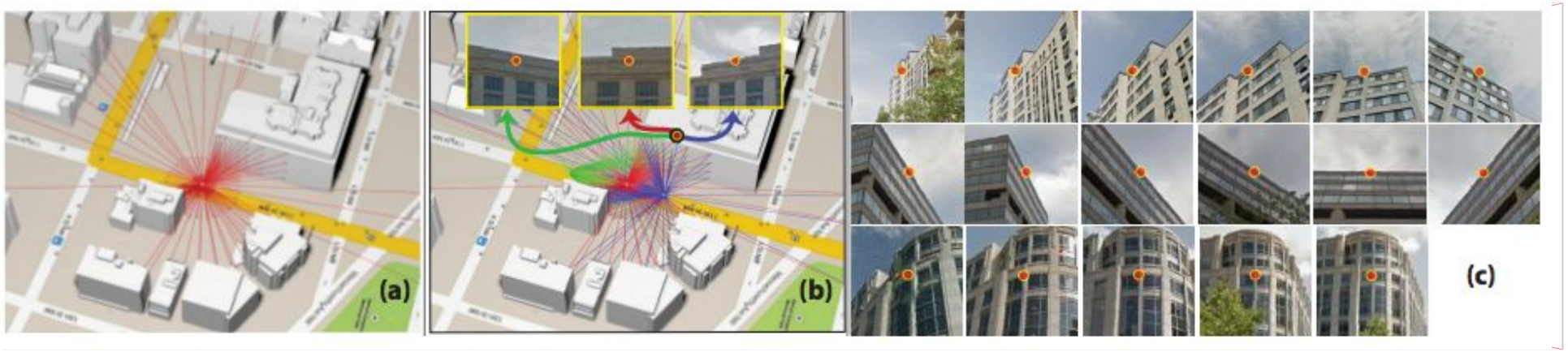
One of the first: M. Brown, G. Hua, and S. Winder. Discriminative learning of local image descriptors. PAMI 2011

-> 0.5 million pairs



Going larger scale

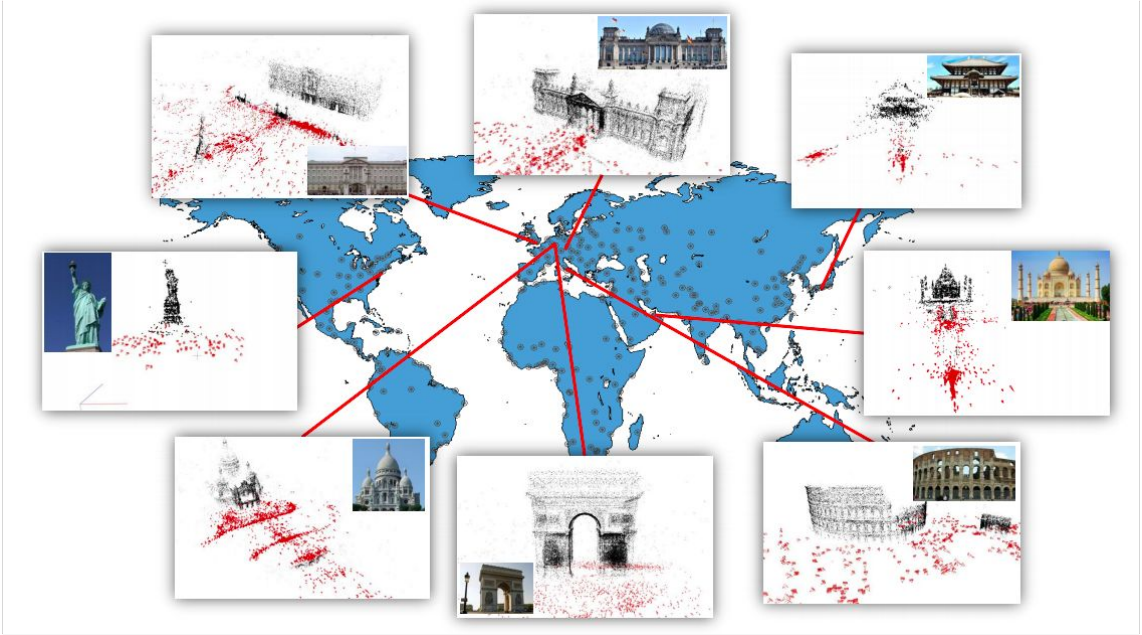
- 0.5 billion correspondences from Google Street View



Zamir, A. R., Wekel, T., Agrawal, P., Wei, C., Malik, J., & Savarese, S.
Generic 3D Representation via Pose Estimation and Matching, ECCV 2016

Going larger scale

- Large datasets have been curated for SFM

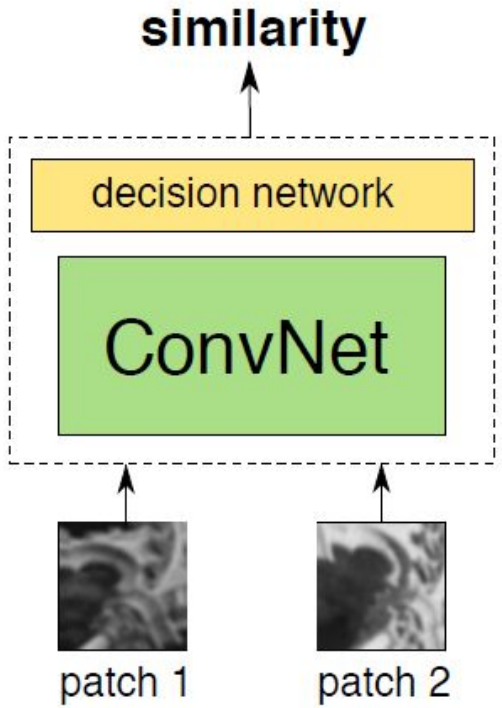


73 models
J. Heinly, J. Schoenberger, E. Dunn, and J.-M. Frahm.
Reconstructing the World in Six Days. CVPR, 2015
200 models
Li, Z., & Snavely, N.
Megadepth: Learning single-view depth prediction from internet photos. CVPR 2018

-> can be used for training, but not/far from perfect

Deep patch feature descriptors

Idea: learning to compare features using a large database of ground truth correspondences

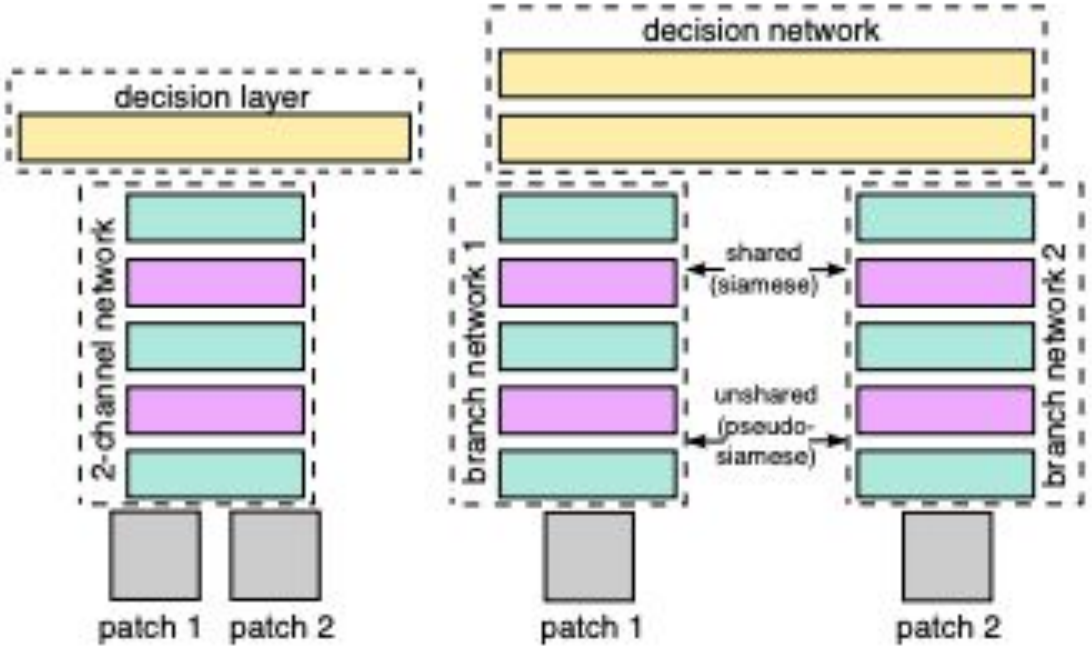


Zagoruyko, S., & Komodakis, N. Learning to Compare Image Patches via Convolutional Neural Networks. *CVPR 2015*

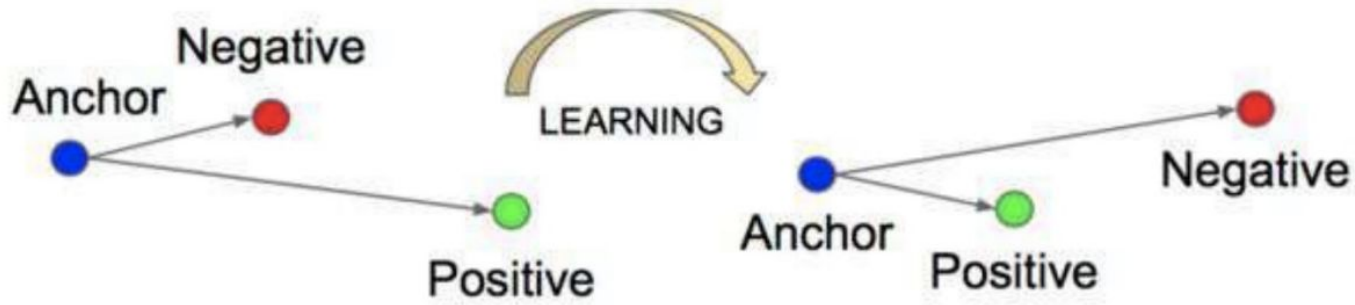
Many options

- Architecture
  e.g. decision network with early vs. late fusion

- In recent works, simply feature comparison with cosine/L2 distance



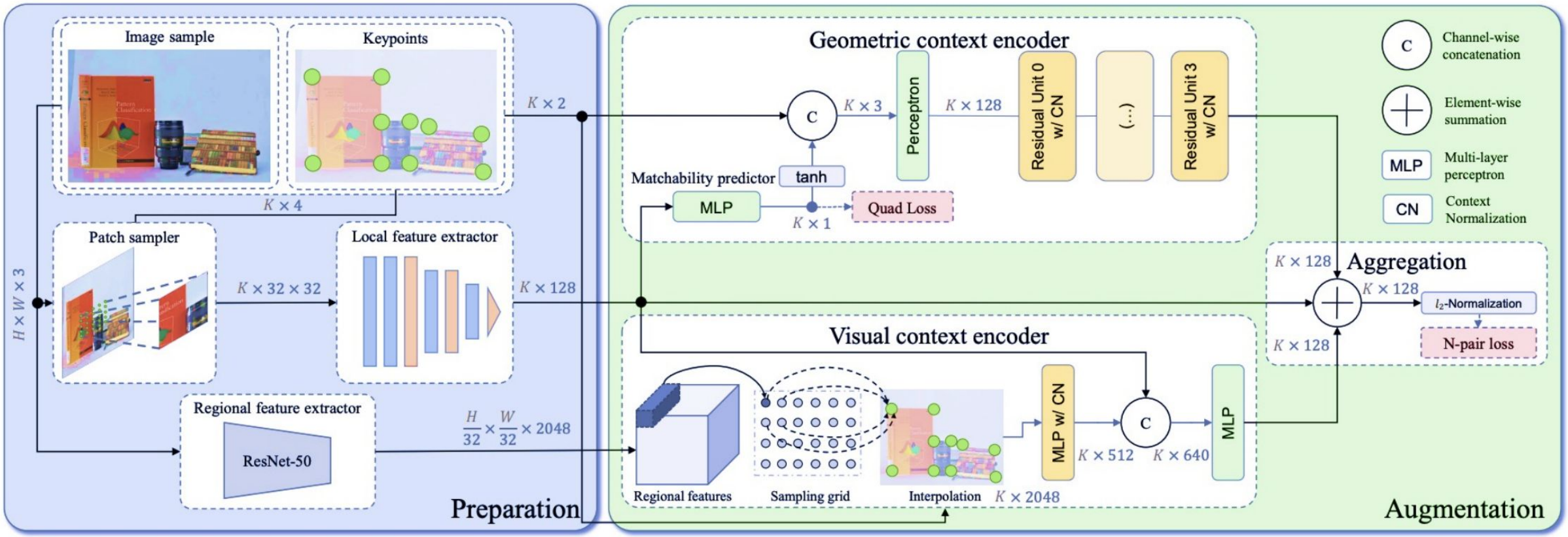
Triplet loss / Contrastive loss



$$Loss = \sum_{i=1}^{N} \left[\| f_i^a - f_i^p \|_2^2 - \| f_i^a - f_i^n \|_2^2 + \alpha \right]$$

→ Choice of positive and negative is important (hard negative/positive mining)

Contextdesc



ContextDesc: Local Descriptor Augmentation with Cross-Modality Context in *CVPR 2019*
Zixin Luo Tianwei Shen Lei Zhou Jiahui Zhang Yao Yao Shiwei Li Tian Fang and Long Quan

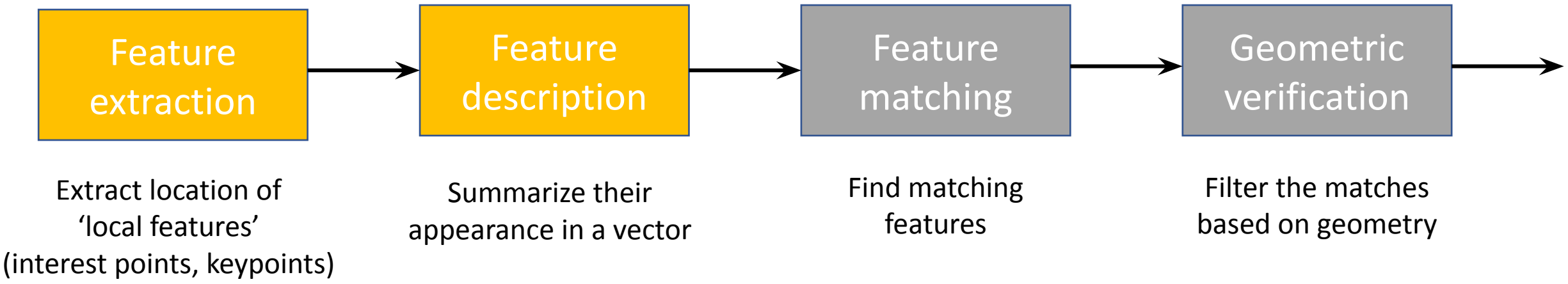
Outline

1. Classical local features
2. Deep local features
   - learning patch descriptors
   - **learning dense detectors and descriptors**
   - learning feature matching
3. Some deep 3D reconstruction

Disclaimer: not a full literature review!

There are tons of paper over the last 2-5 years,
this is a biased selection to illustrate idea I think I worth knowing

Local features and correspondences pipeline for 3D reconstruction



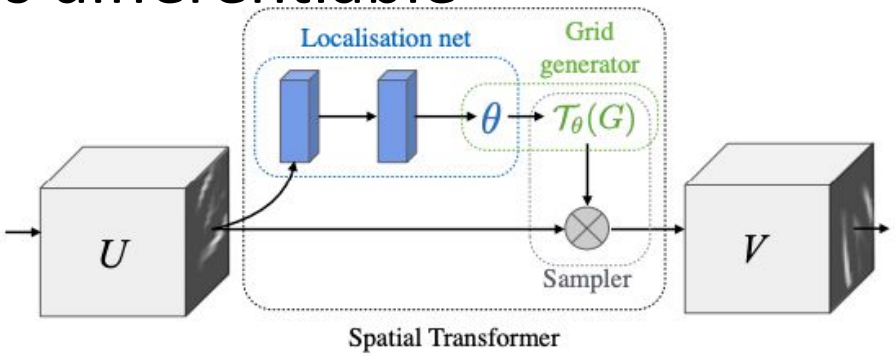
Deep detectors/dense local descriptors

- Use the full image
- Use a fully convolutional architecture

2 key elements:

- **Spatial transformer network**: cropping is differentiable

Jaderberg, M., Simonyan, K., & Zisserman, A.
Spatial transformer networks. NIPS 2015

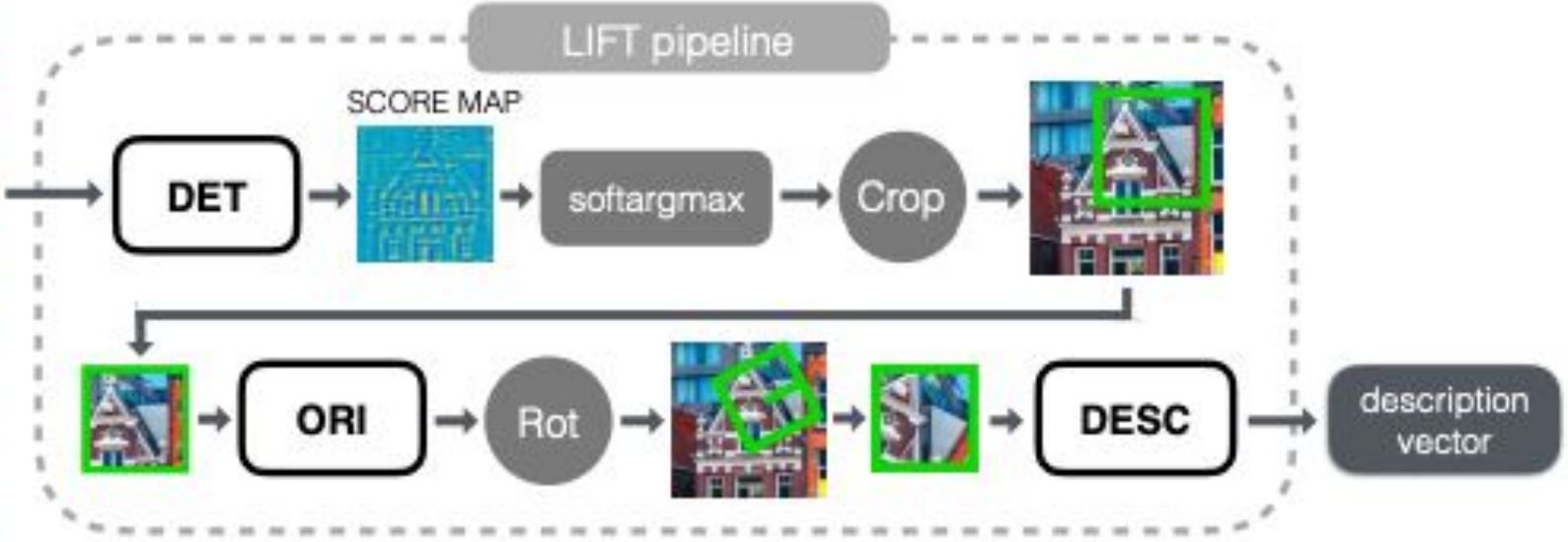


- **Soft-argmax**: similar to softmax

$$\text{softargmax}\,(\mathbf{S}) = \frac{\sum_{\mathbf{y}} \exp(\beta \mathbf{S}(\mathbf{y}))\mathbf{y}}{\sum_{\mathbf{y}} \exp(\beta \mathbf{S}(\mathbf{y}))}$$

LIFT

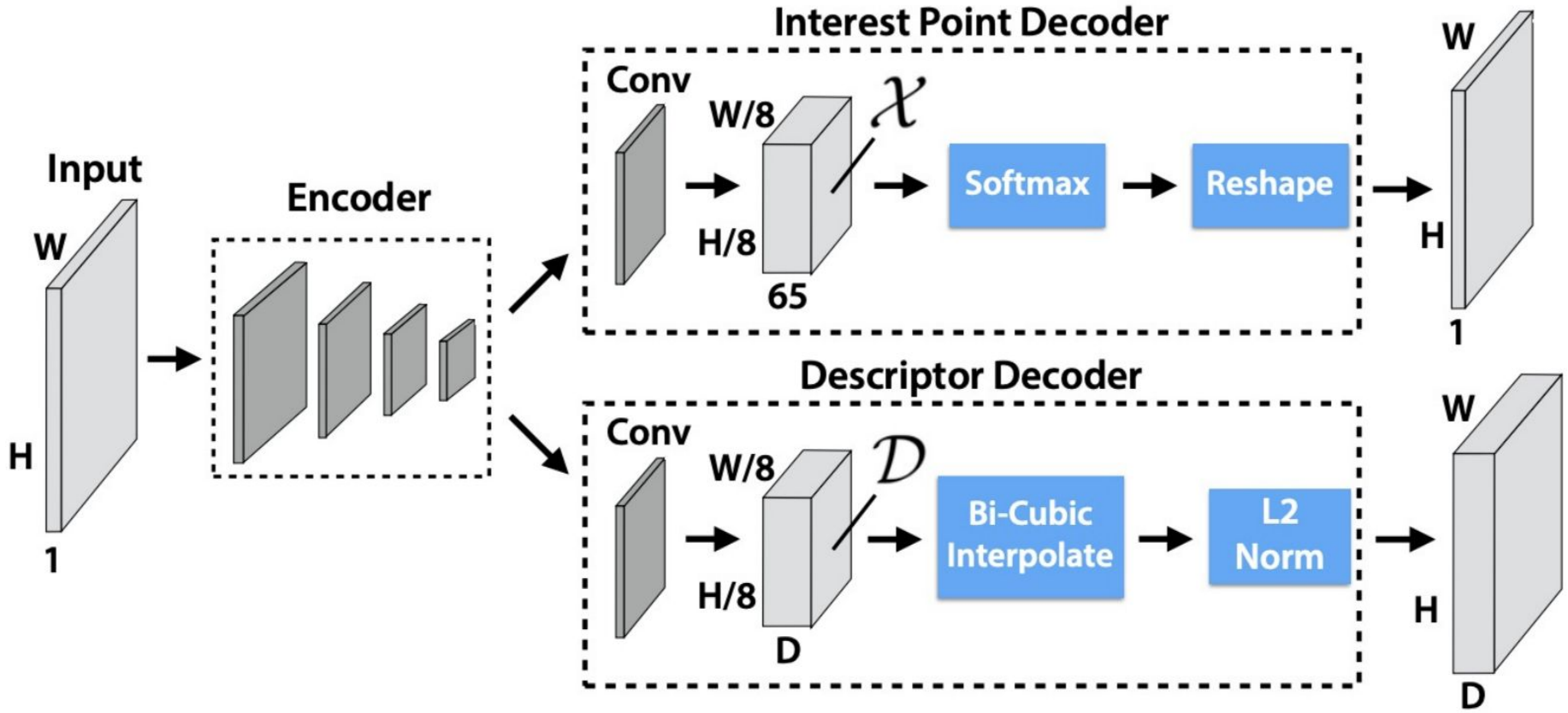
- Stay close to SIFT pipeline



- every operation can be made differentiable (softargmax, spatial transformer networks)
- Training data: SIFT-based SFM points
- Loss descriptors: hinge embedding $\mathcal{L}_{\text{desc}}(\mathbf{p}_\theta^k, \mathbf{p}_\theta^l) = \begin{cases} \left\| h_\rho(\mathbf{p}_\theta^k) - h_\rho(\mathbf{p}_\theta^l) \right\|_2 & \text{for positive pairs, and} \\ \max\left(0, C - \left\| h_\rho(\mathbf{p}_\theta^k) - h_\rho(\mathbf{p}_\theta^l) \right\|_2\right) & \text{for negative pairs ,} \end{cases}$
- Loss orientation: minimize descriptors distance
- Loss detector: peaked distribution in regions with SfM points + flat distribution in regions with no SfM points + leads to similar descriptors for positives (pre-training with IoU of reconstructed points)
- Descriptor, orientation and detector learned one after the other

Yi, K. M., Trulls, E., Lepetit, V., & Fua, P.
Lift: Learned invariant feature transform, ECCV 2016

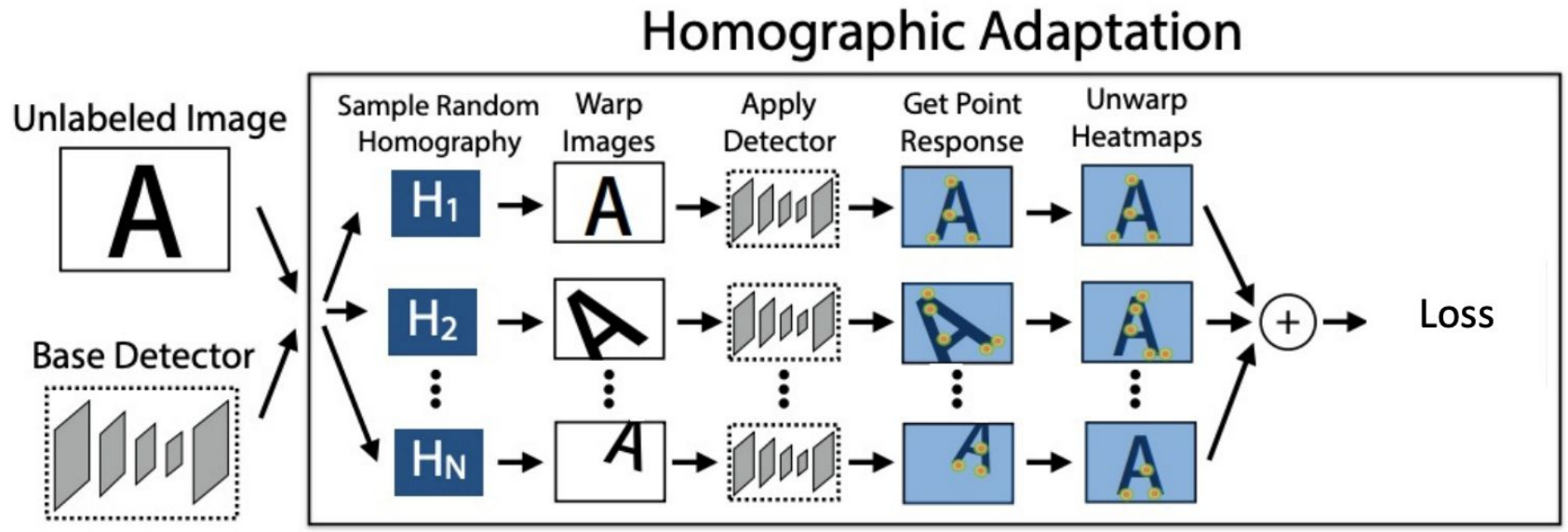
Superpoint



DeTone, D., Malisiewicz, T., & Rabinovich, A.
Superpoint: Self-supervised interest point detection and description.
CVPR 2018

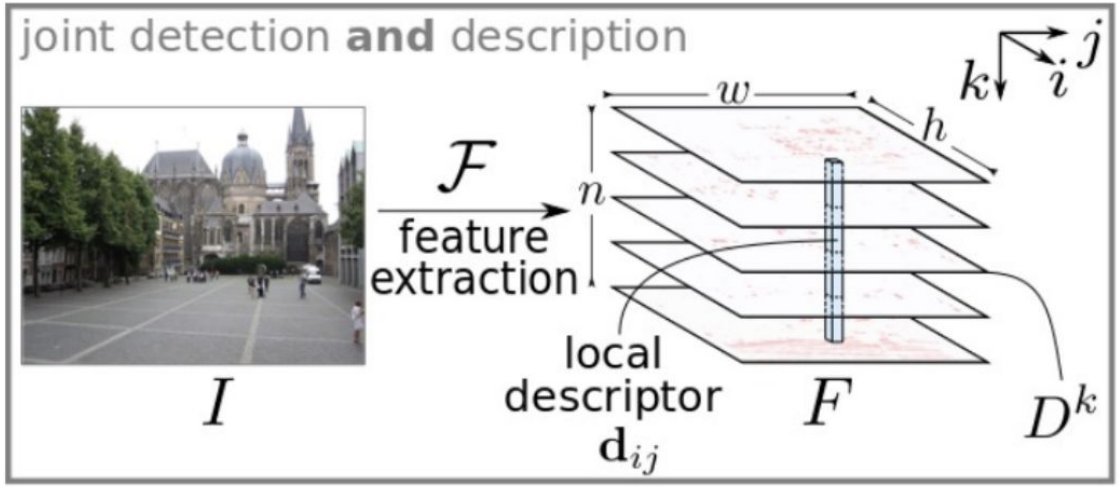
Superpoint

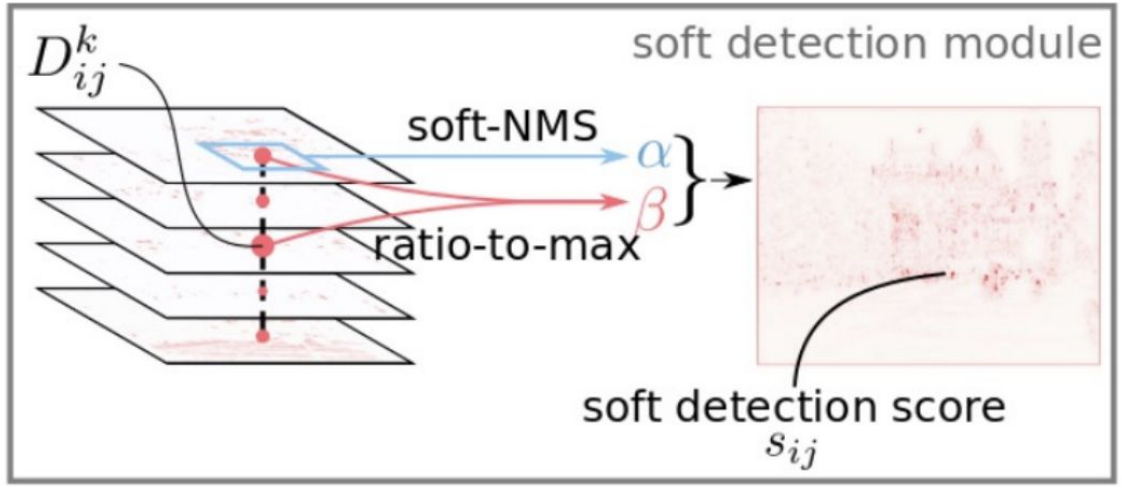
- 1st training on synthetic shape (including interest points labels)
- 2nd training on synthetic image transformations



DeTone, D., Malisiewicz, T., & Rabinovich, A.
Superpoint: Self-supervised interest point detection and description.
CVPR 2018

D2-net

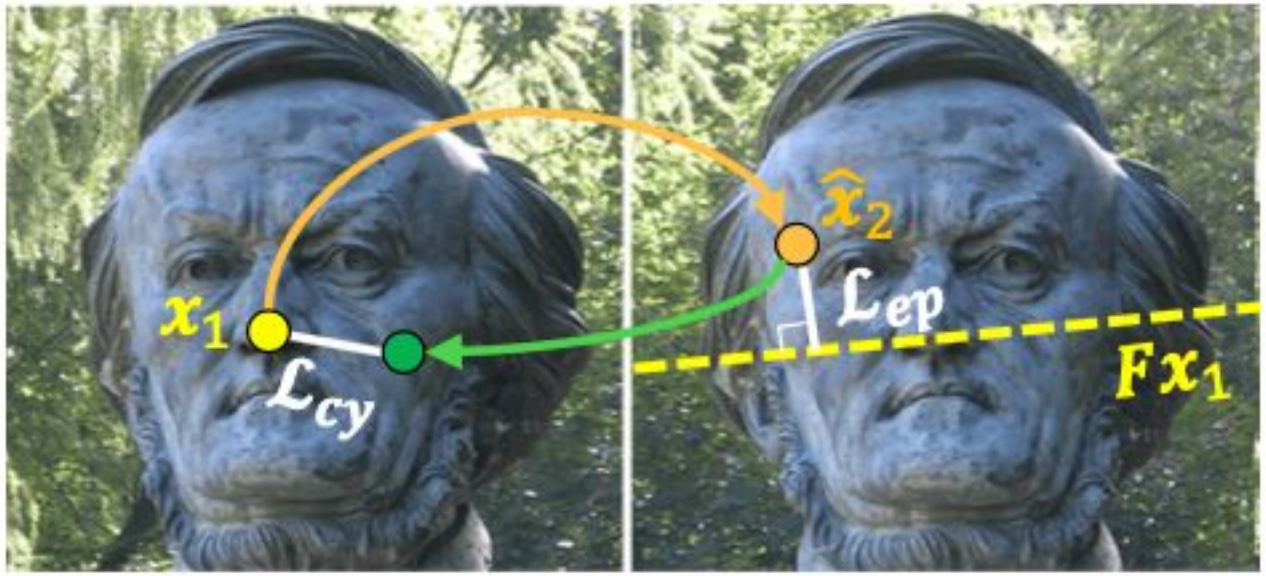




- Descriptors are simply the CNN features
- Detection is max norm features + NMS made differentiable

Dusmanu, M., Rocco, I., Pajdla, T., Pollefeys, M., Sivic, J., Torii, A., & Sattler, T.
D2-net: A trainable cnn for joint description and detection of local features.
CVPR 2019

Weak supervision from epipolar geometry



Wang, Q., Zhou, X., Hariharan, B., & Snavely, N.
Learning feature descriptors using camera pose supervision.
ECCV 2020

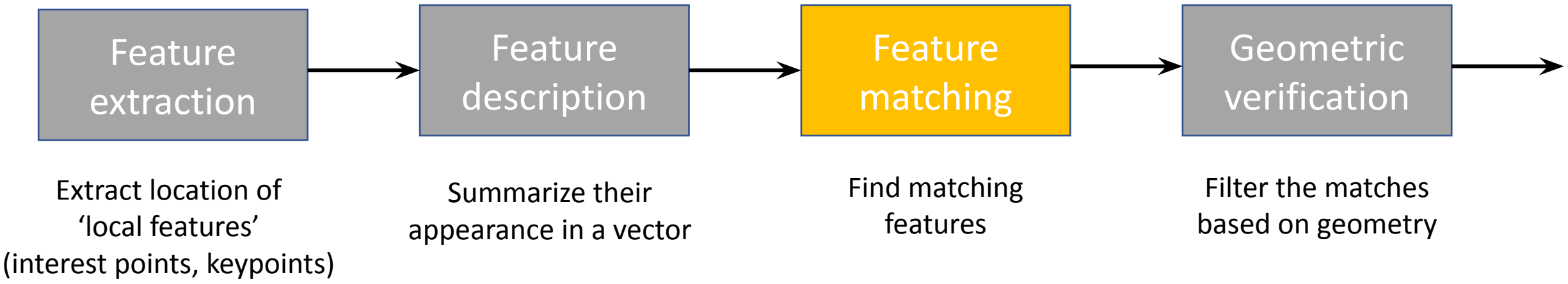
Outline

1. Classical local features
2. Deep local features
   - learning patch descriptors
   - learning dense detectors and descriptors
   - **learning feature matching**
3. Some deep 3D reconstruction

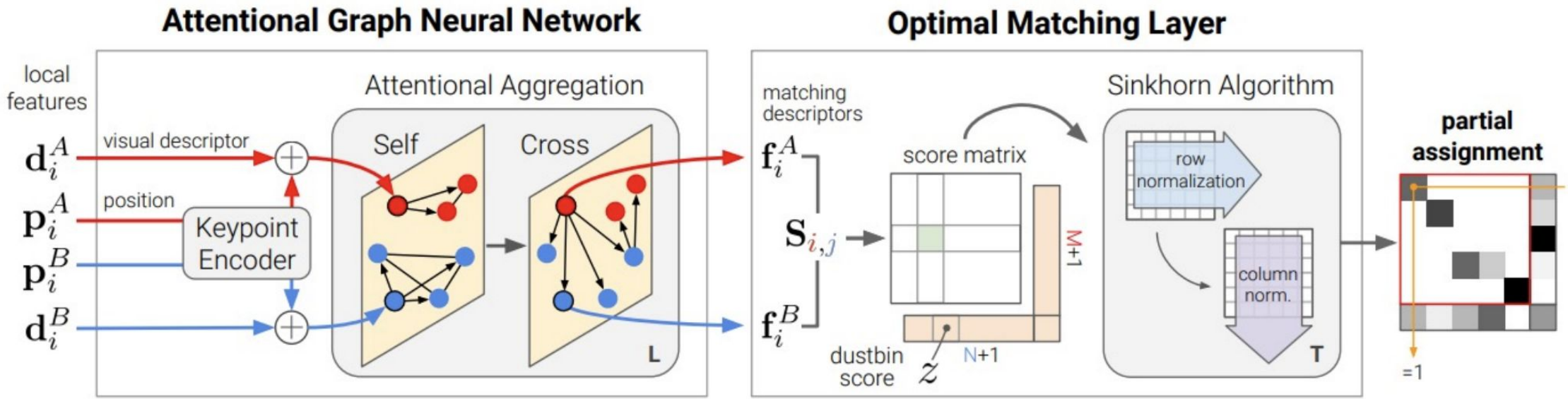
Disclaimer: not a full literature review!

There are tons of paper over the last 2-5 years,
this is a biased selection to illustrate idea I think I worth knowing

Local features and correspondences pipeline for 3D reconstruction



Superglue



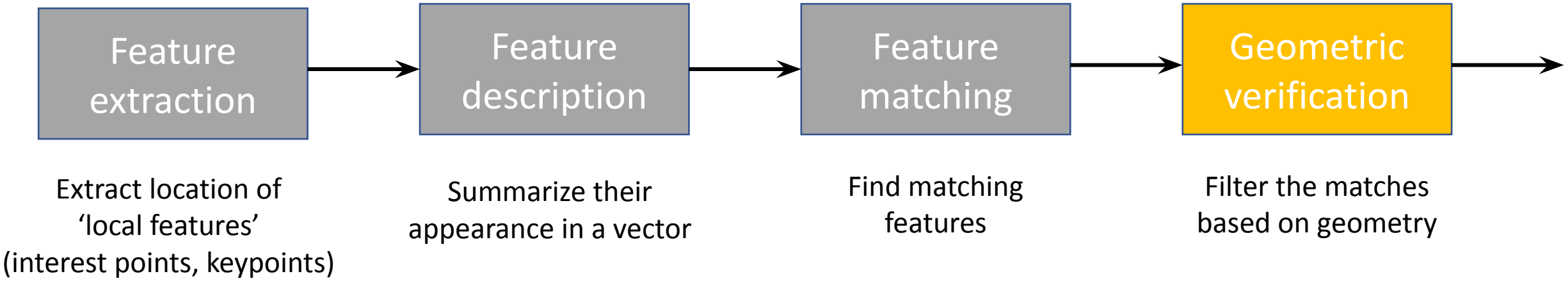
- Consider keypoint extraction and description done
- Feature matching problem = Graph matching
- Use GNN (here transformers)

Sarlin et al. SuperGLue Learning Feature Matching With Graph Neural Network *CVPR 2020*

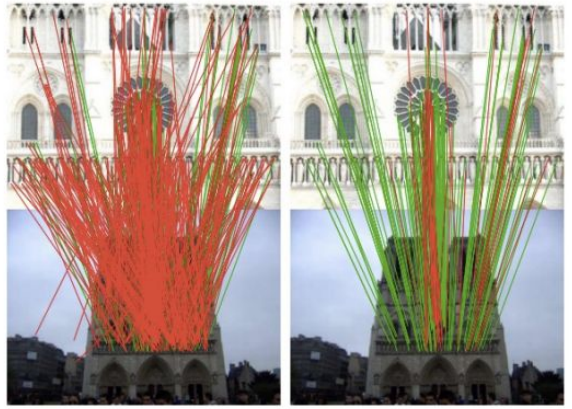
Superglue

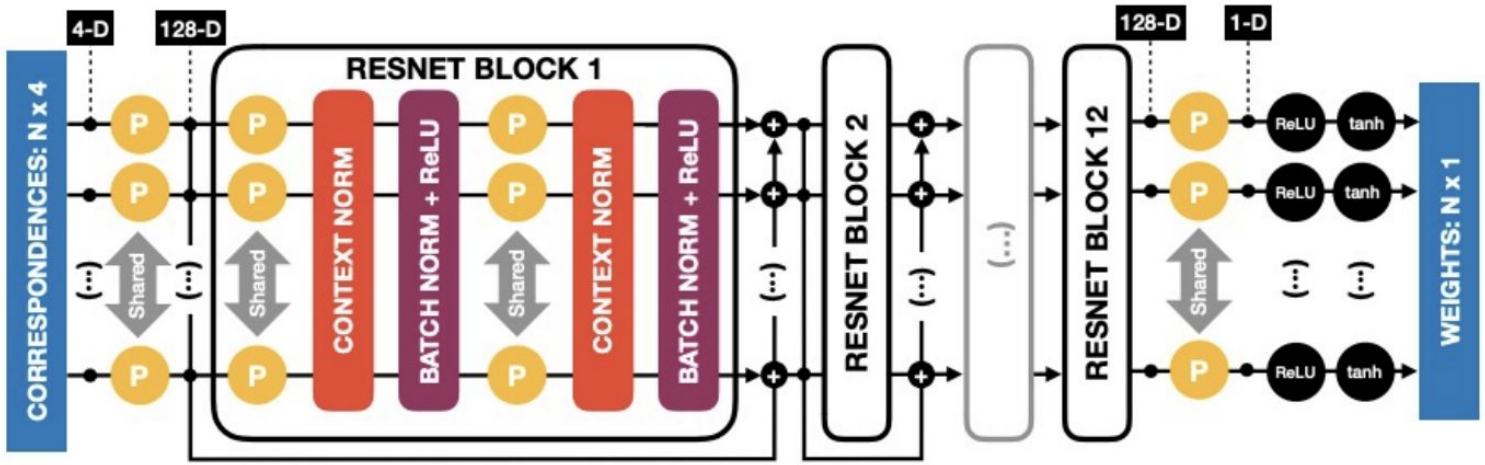
- Training on mix of synthetic transformations and real SfM
- Trained detector and descriptors + superglue à best method today

Local features and correspondences pipeline for 3D reconstruction



Learning to filter correspondences





Input = raw matches (m*4)
Output = weights « how good the match is »
Supervision = GT geometry

Moo Yi, K., Trulls, E., Ono, Y., Lepetit, V., Salzmann, M., & Fua, P.
Learning to find good correspondences. CVPR 2018

Learning to filter correspondences

- Computing the essential matrix is a SVD which is a differentiable operation
- Compute the SVD with weighted correspondences and compare it to the GT essential matrix -> optimize weights
- Also uses (and start with only) cross entropy with “GT” labels

Moo Yi, K., Trulls, E., Ono, Y., Lepetit, V., Salzmann, M., & Fua, P.
Learning to find good correspondences. CVPR 2018

Evaluation

- Remains an important challenge, as datasest with good Ground Truth are rare, small, biased…

- A solution can be to evaluate another task than 3D reconstruction, for which GT is easier to get, e.g. localization

Outline

1. Classical local features
2. Deep local features
3. Some deep 3D reconstruction

Disclaimer: not a full litterature review!

There are tons of paper over the last 2-5 years,
this is a biaised selection to illustrate idea I think I worth knowing

Outline

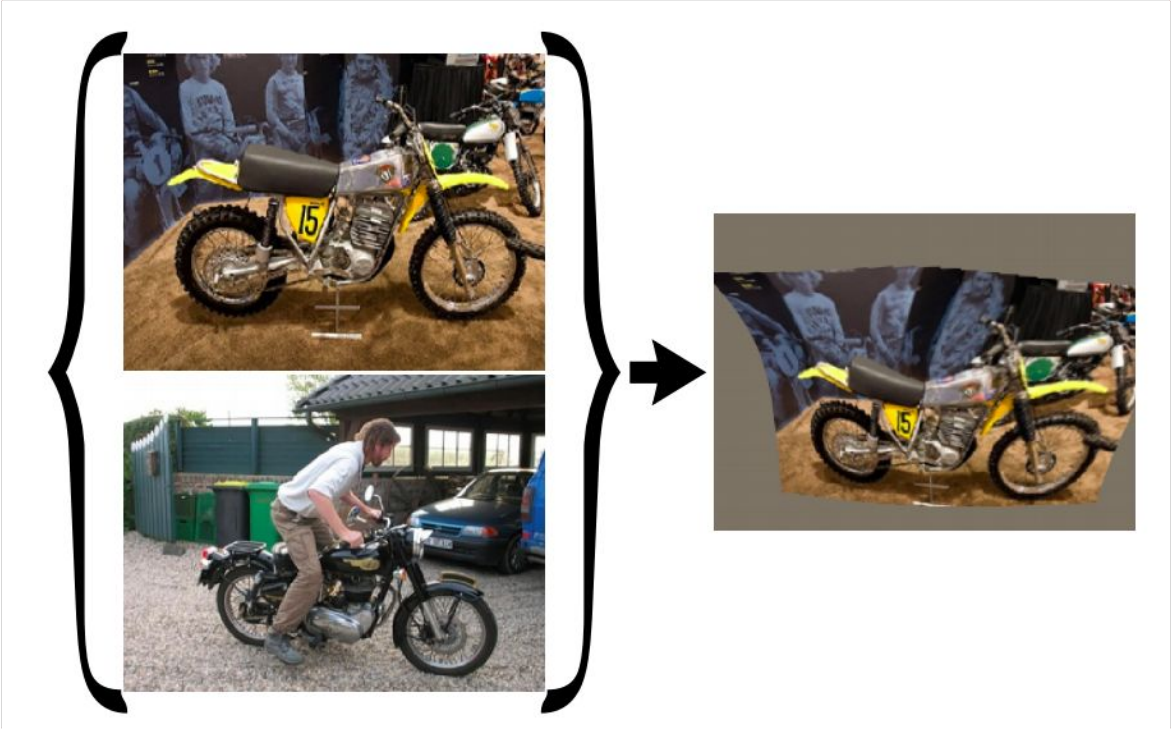
1. Classical local features
2. Deep local features
3. Some deep 3D reconstruction
   - Correspondences without keypoints
     - Coarse Flow
     - Fine flow
     - Deep MVS
   - NERFs

Disclaimer: not a full litterature review!

There are tons of paper over the last 2-5 years,
this is a biaised selection to illustrate idea I think I worth knowing

Learning transformation

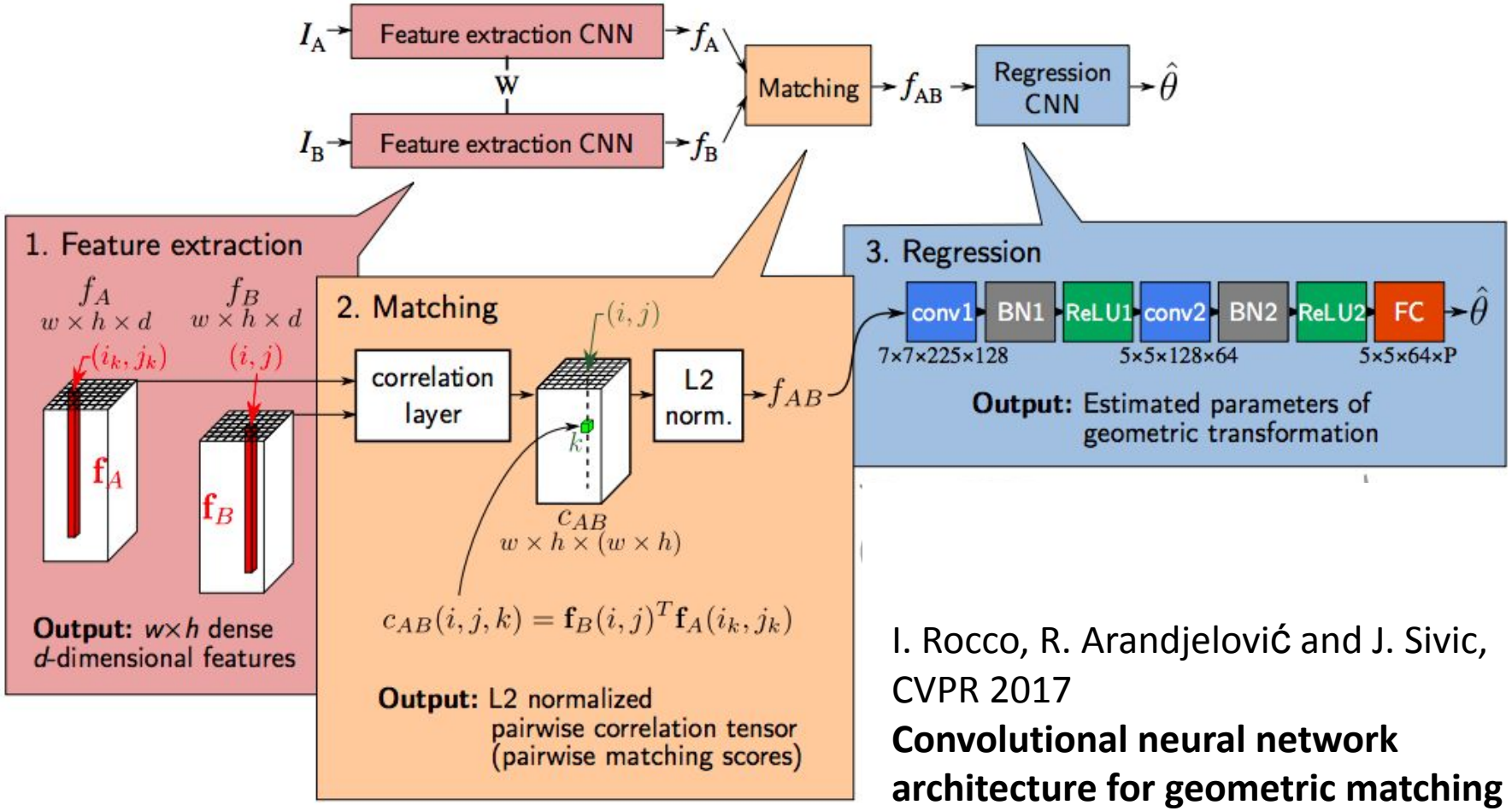
- By learning to predict transformation



I. Rocco, R. Arandjelović and J. Sivic, CVPR 2017
Convolutional neural network architecture for geometric matching

Learning transformation

- By learning to predict transformation

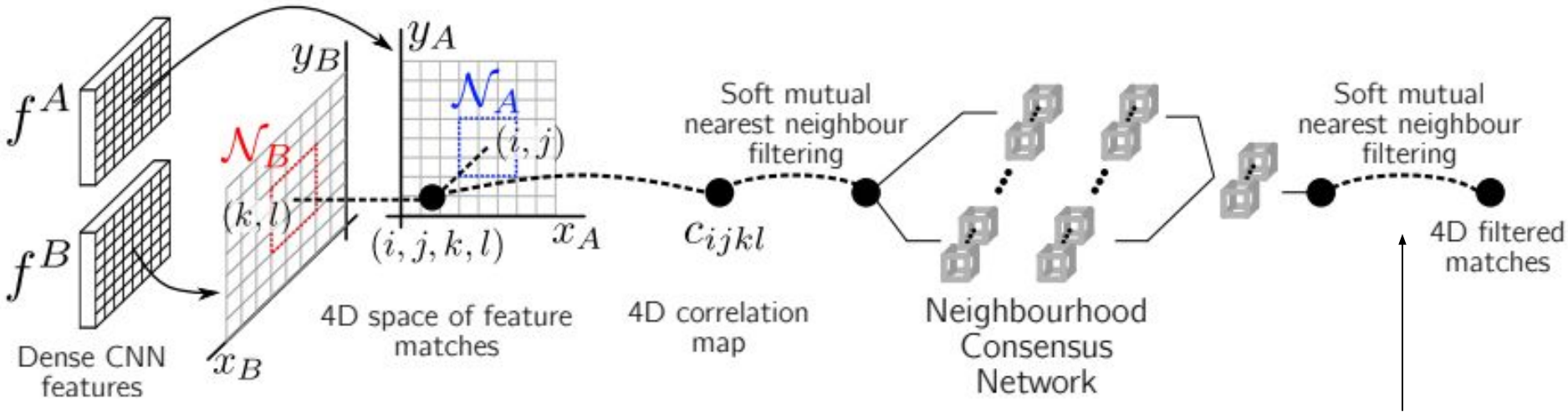


I. Rocco, R. Arandjelović and J. Sivic, CVPR 2017
Convolutional neural network architecture for geometric matching

Learning to match features

- Using consistency

$$r^A_{ijkl} = \frac{c_{ijkl}}{\max_{ab} c_{abkl}}, \quad \text{and} \quad r^B_{ijkl} = \frac{c_{ijkl}}{\max_{cd} c_{ijcd}}$$



$$s^A_{ijkl} = \frac{\exp(c_{ijkl})}{\sum_{ab} \exp(c_{abkl})} \quad \text{and} \quad s^B_{ijkl} = \frac{\exp(c_{ijkl})}{\sum_{cd} \exp(c_{ijcd})}$$

Rocco, I., Cimpoi, M., Arandjelović, R., Torii, A., Pajdla, T., & Sivic, J
Neighbourhood consensus networks. NIPS 2018

Learning to match features

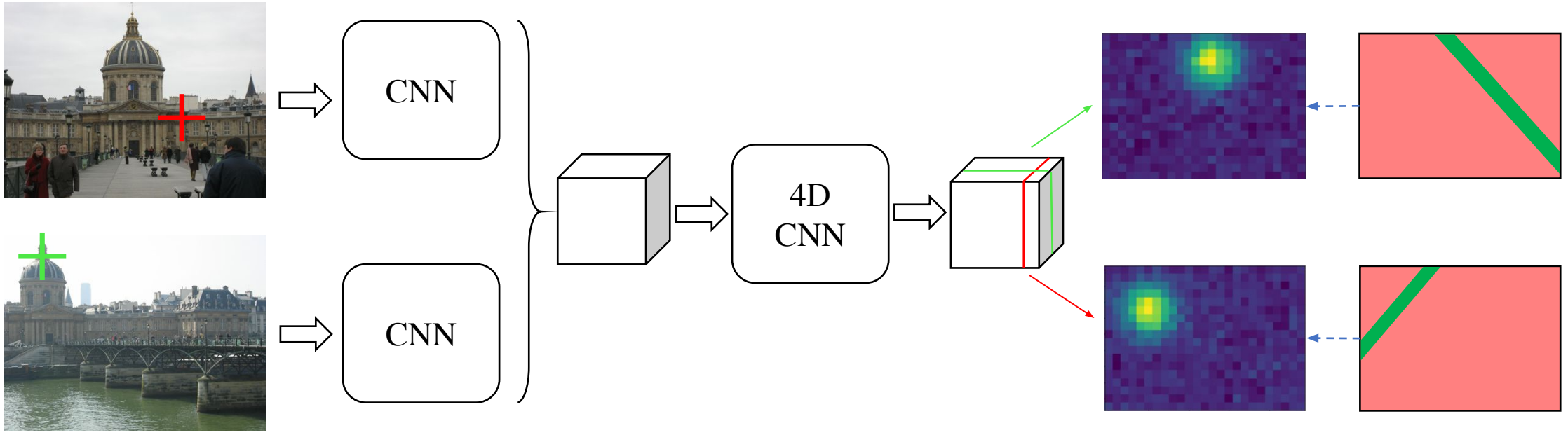
- Using weak supervision only
  (pairs of matching and non matching images)

$$\mathcal{L}(I^A, I^B) = -y\left(\bar{s}^A + \bar{s}^B\right)$$

y=1 for positive pairs, -1 for negative pairs

Rocco, I., Cimpoi, M., Arandjelović, R., Torii, A., Pajdla, T., & Sivic, J
Neighbourhood consensus networks. NIPS 2018

Epipolar supervision



ResNet Feature extraction | Correlation volume | Output volume | Heatmaps | Supervision signal

Darmon, F., Aubry, M., & Monasse, P. Learning to Guide Local Feature Matches, 3DV 2020

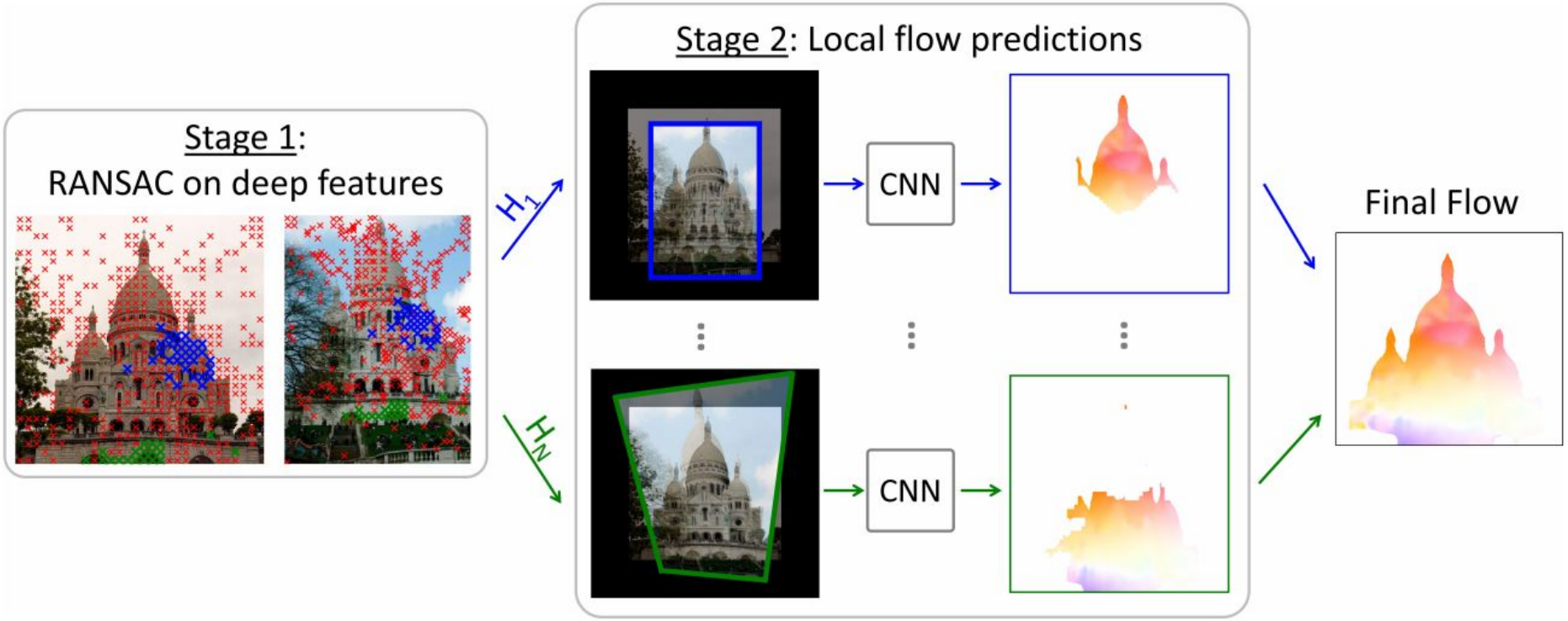
Outline

1. Classical local features
2. Deep local features
3. Some deep 3D reconstruction
   - Correspondences without keypoints
     - Coarse Flow
     - **Fine flow**
     - Deep MVS
   - NERFs

Disclaimer: not a full litterature review!

There are tons of paper over the last 2-5 years,
this is a biaised selection to illustrate idea I think I worth knowing

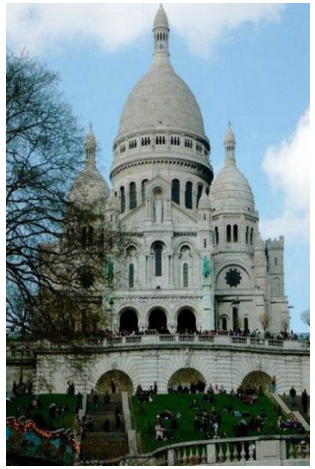
RANSAC-Flow



Shen, X., Darmon, F., Efros, A. A., & Aubry, M.
RANSAC-Flow: generic two-stage image alignment.
ECCV 2020

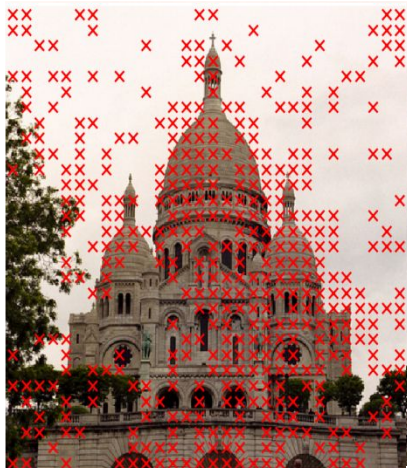
RANSAC-Flow

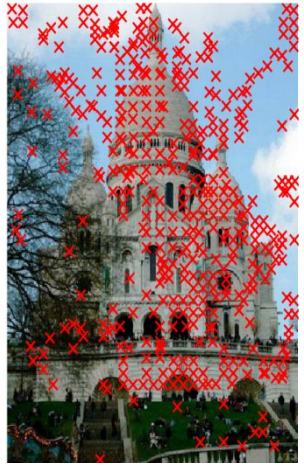
Stage 1:
RANSAC on deep features



RANSAC-Flow

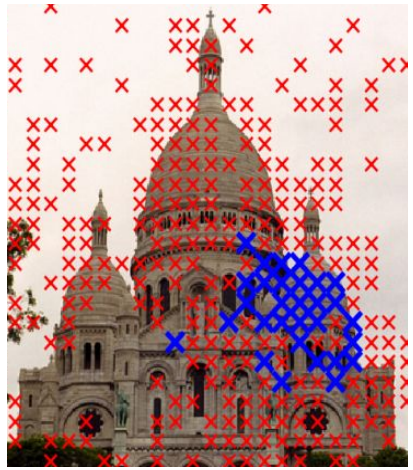
Stage 1:
RANSAC on deep features

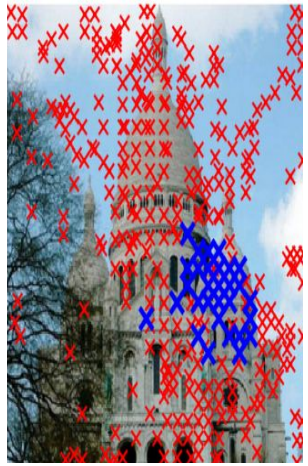




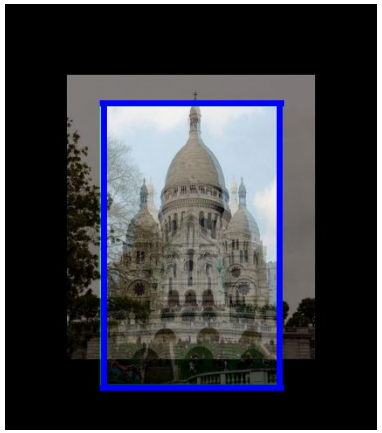
RANSAC-Flow

<u>Stage 1</u>:
RANSAC on deep features



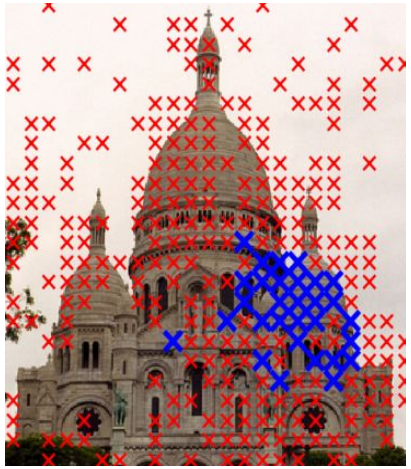


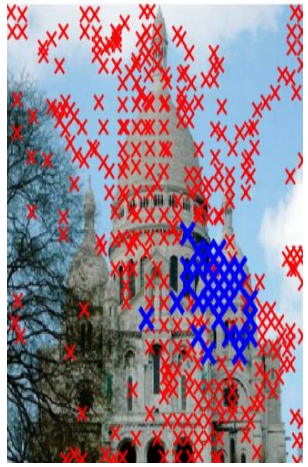
H



An unsupervised two-stage method

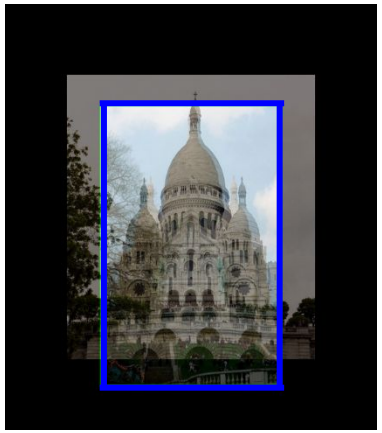
Stage 1:
RANSAC on deep features



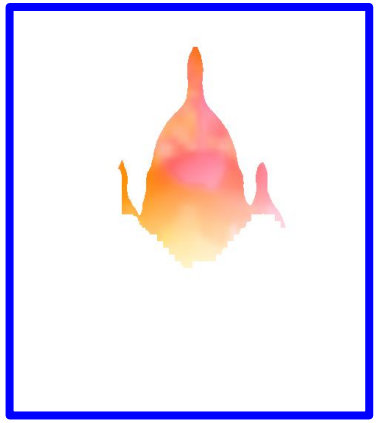


H

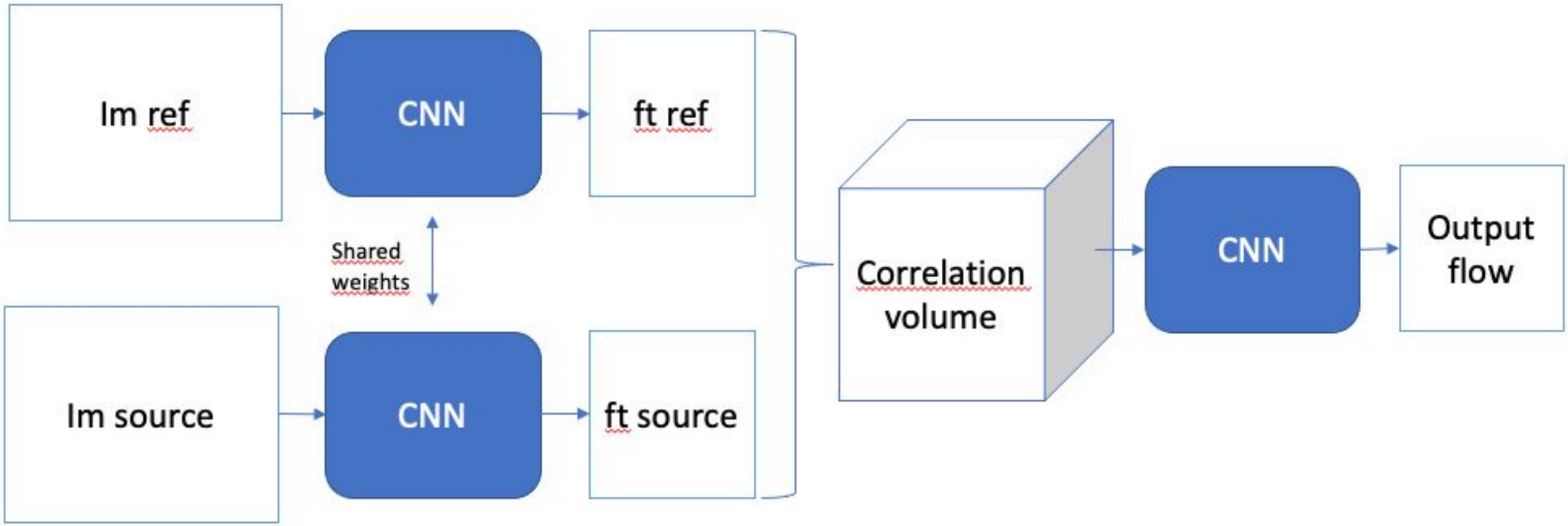
Stage 2: Local flow predictions



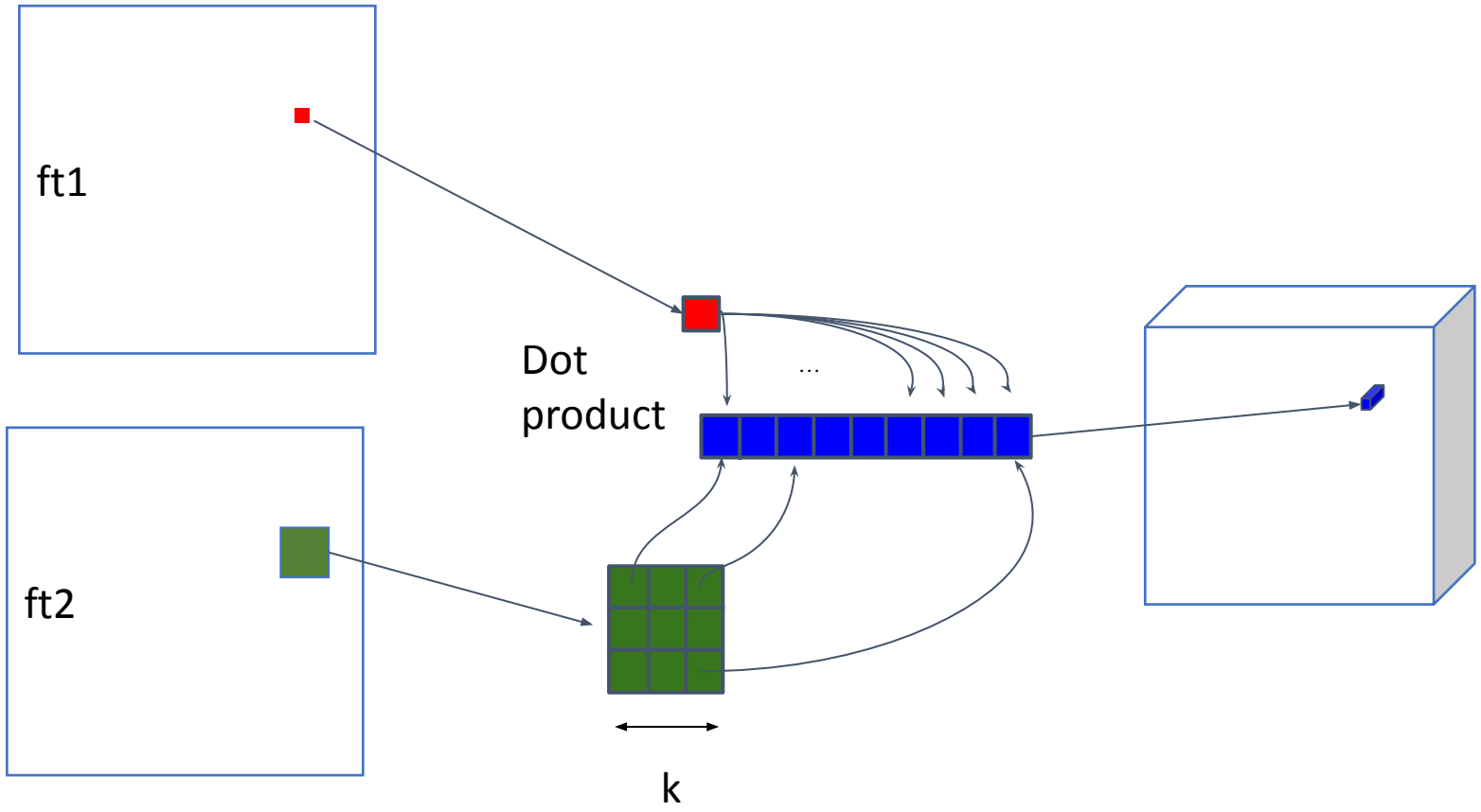
CNN



Architecture (RANSAC-Flow)



Correlation volume (RANSAC-flow, standard in OF)

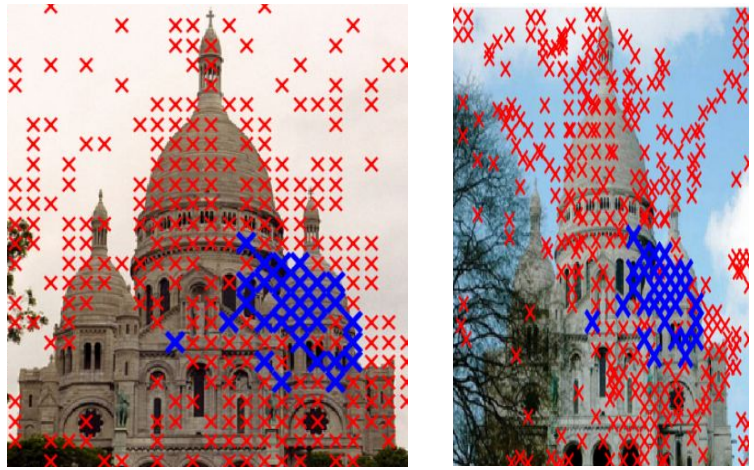


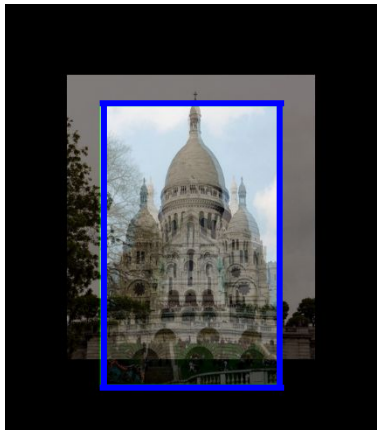
RANSAC-Flow

SSIM + mask + cycle-consistency loss

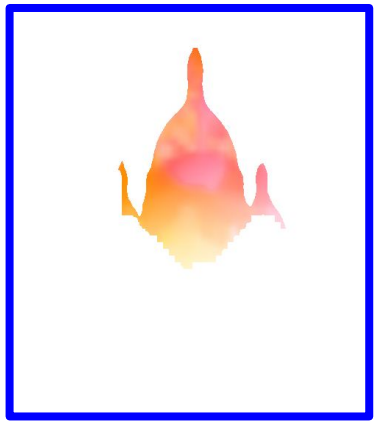
Stage 2: Local flow predictions

Stage 1:
RANSAC on deep features





CNN



$$\mathcal{L}_m(I_s, I_t) = \sum_{(x,y)\in I_t} |M_t^{cycle}(x,y) - 1|$$

Mask loss

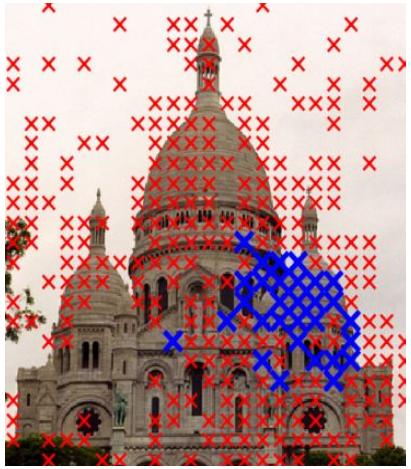
Confidence at (x,y)

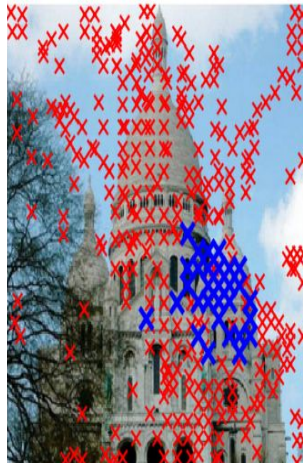
RANSAC-Flow

SSIM + mask + cycle-consistency loss

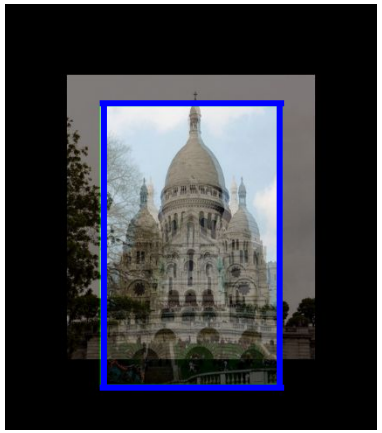
Stage 2: Local flow predictions

Stage 1:
RANSAC on deep features

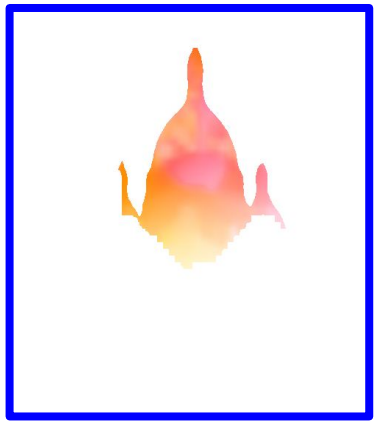




H



CNN



$$\mathcal{L}_{rec}^{SSIM}(I_s, I_t) = \sum_{(x,y)\in I_t} M_t^{cycle}(x,y)\,(1 - SSIM\,(I_s(x',y'), I_t(x,y)))$$

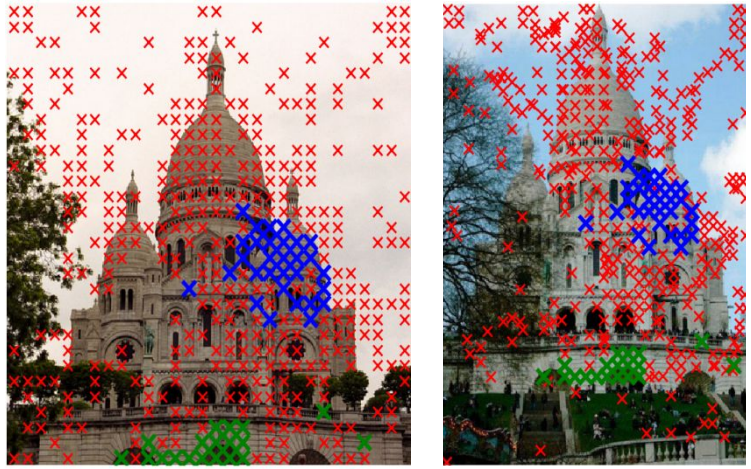
Confident regions

$$\mathcal{L}_c(I_s, I_t) = \sum_{(x,y)\in I_t} M_t^{cycle}(x,y)\|(x',y'), \mathbf{F}_{t\to s}(x,y)\|_2$$

RANSAC-Flow

SSIM + mask + cycle-consistency loss

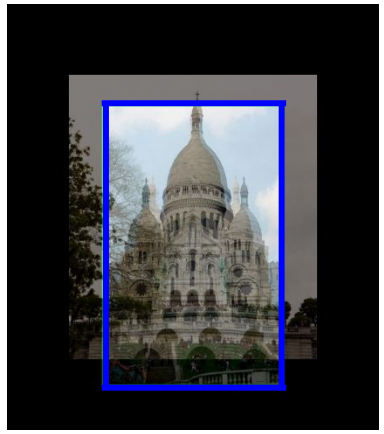
Stage 1:
RANSAC on deep features



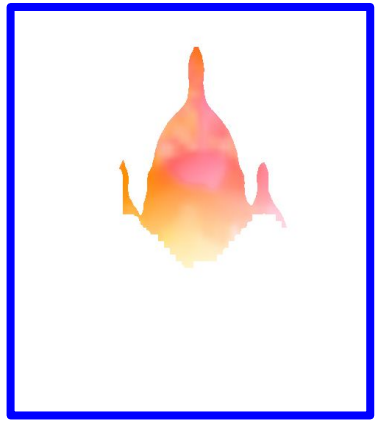
H_1

H_K

Stage 2: Local flow predictions



CNN



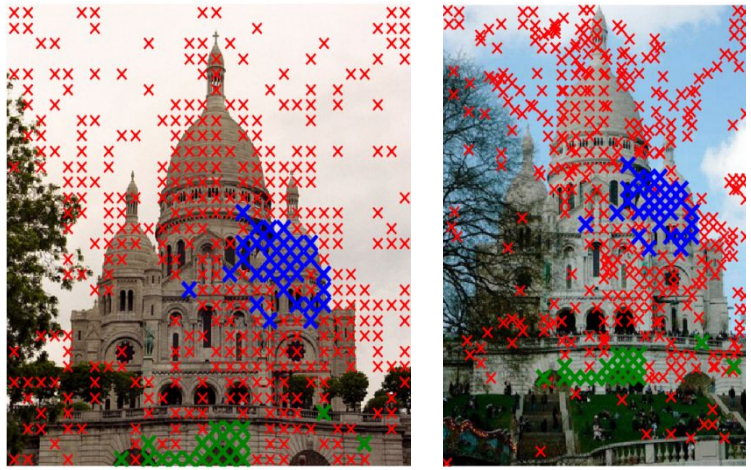
CNN

RANSAC-Flow

SSIM + mask + cycle-consistency loss

E.g. : MOCO features

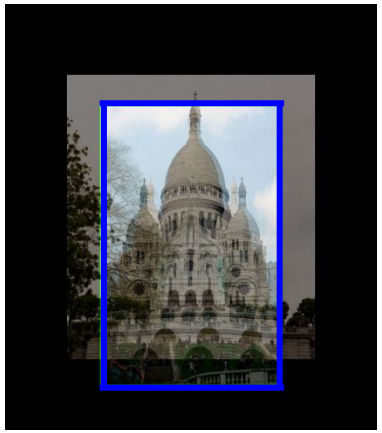
Stage 1:
RANSAC on deep features



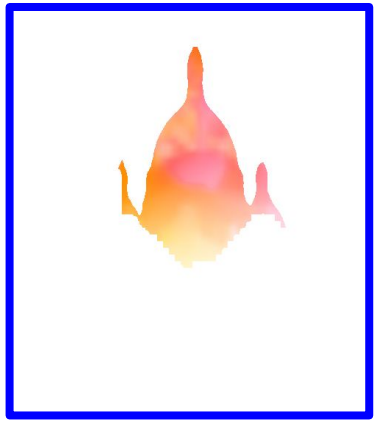
H_1

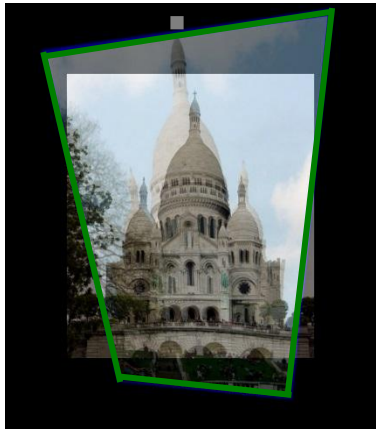
H_K

Stage 2: Local flow predictions



CNN





CNN

Final Flow

No 3D, unsupervised generic image alignment

Optical flow

Localization

Dense image alignment

2-view geometry estimation

Application: Aligning artworks from Brueghel [6,7]

[6]: Brueghel family, http://www.janbrueghel.net/

[7]: Shen Xi et al., Discovering visual patterns in art collections with spatially-consistent feature learning, CVPR 2019

Average images of the inputs

Average image after our coarse alignment

Average image after our fine alignment

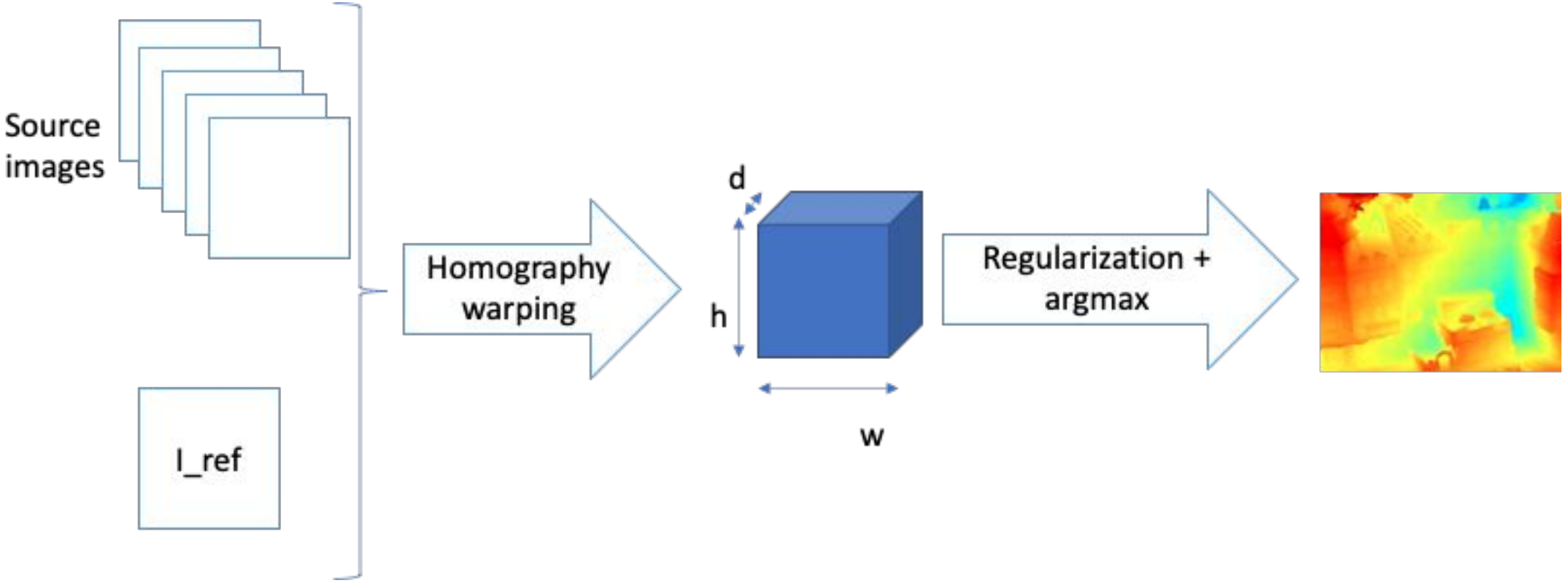
Outline

1. Classical local features
2. Deep local features
3. Some deep 3D reconstruction
   - Correspondences without keypoints
     - Coarse Flow
     - Fine flow
     - **Deep MVS**
   - NERFs

Disclaimer: not a full litterature review!

There are tons of paper over the last 2-5 years,
this is a biaised selection to illustrate idea I think I worth knowing

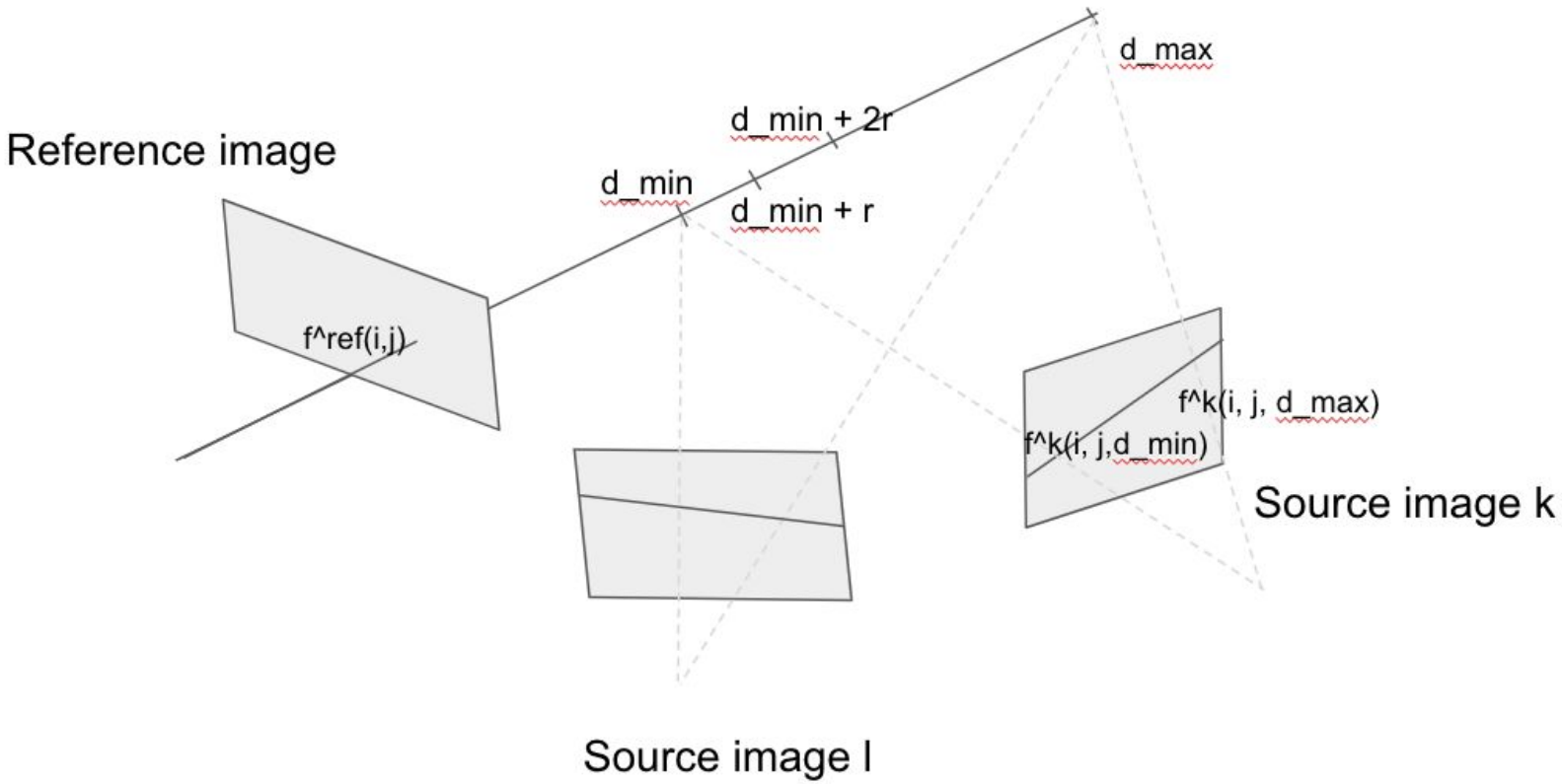
Deep MVS



Huang, P. H., Matzen, K., Kopf, J., Ahuja, N., & Huang, J. B.
DeepMVS: Learning multi-view stereopsis.
CVPR2018

Deep MVS

$$c_{i,j,d} = var\left(f^{ref}(i,j), f^{1}(i,j,d), \cdots, f^{N}(i,j,d)\right)$$



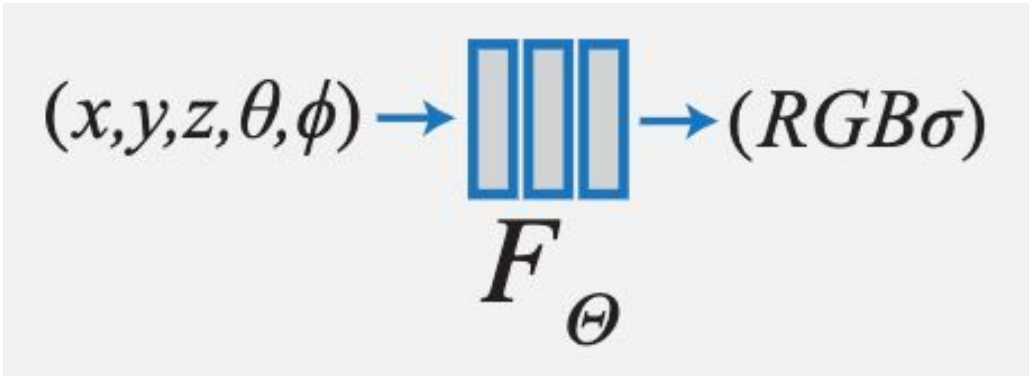
Outline

1. Classical local features
2. Deep local features
3. Some deep 3D reconstruction
   - Correspondences without keypoints
     - Coarse Flow
     - Fine flow
     - Deep MVS
   - **NERFs**

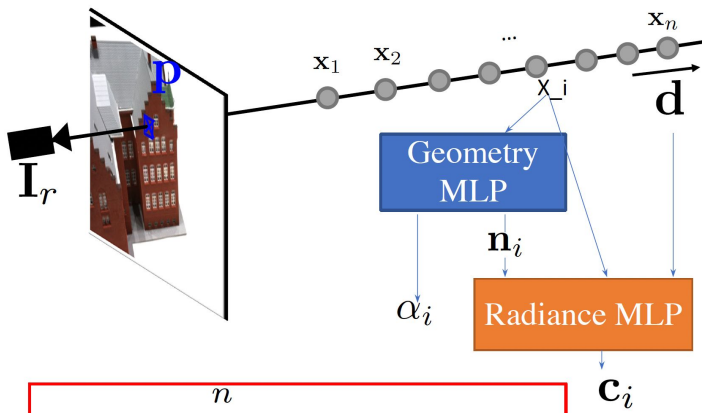
Disclaimer: not a full litterature review!

There are tons of paper over the last 2-5 years,
this is a biaised selection to illustrate idea I think I worth knowing

Parametric scene / NeRF [Mildenhall20]

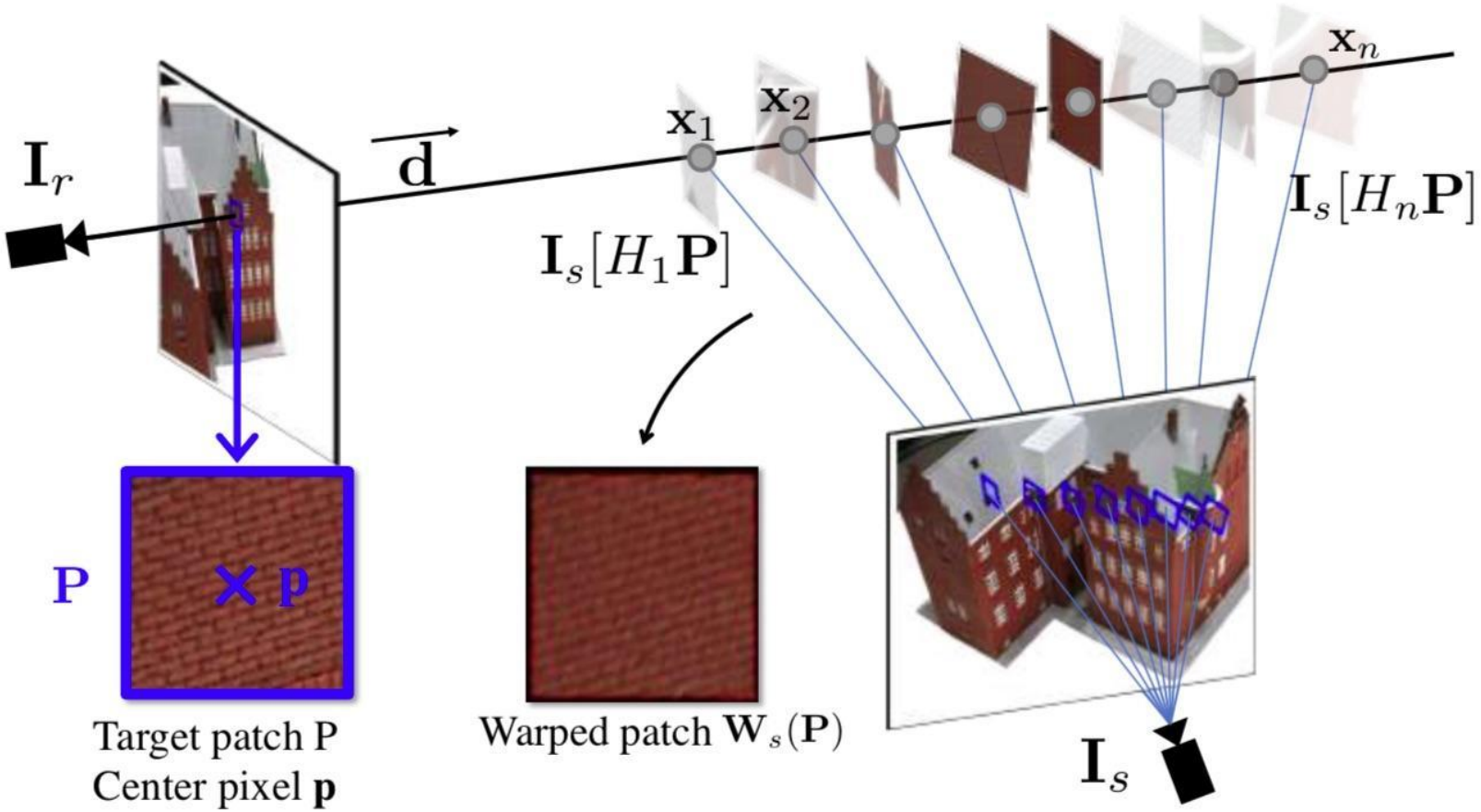


More in details



$$V(\mathbf{p}) = \sum_{i=1}^{n} \alpha_i \prod_{j<i} (1-\alpha_j)\mathbf{c}_i$$

Can use correspondences



NeuralWarp: Improving neural implicit surfaces geometry with patch warping
F. Darmon, B. Bascle, J.-C. Devaux, P. Monasse, M. Aubry
CVPR 2022

nerfstudio

Github 2,312

colab

Search

GETTING STARTED

Installation

Training your first model

Using custom data

Using the viewer

Data conventions

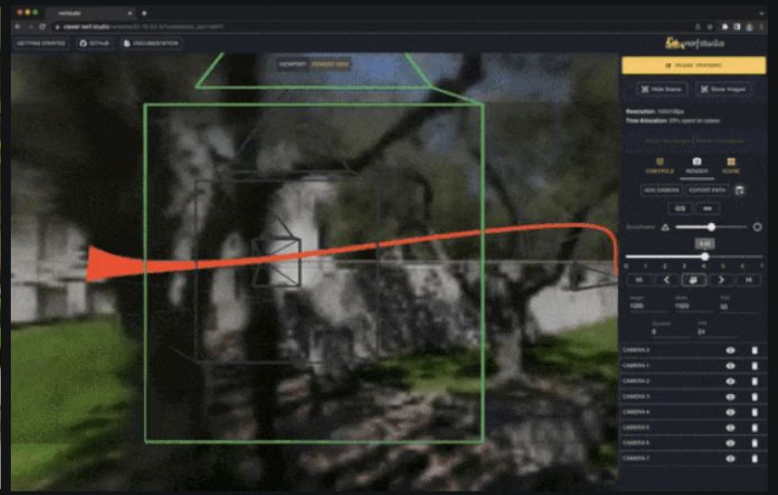
Contributing

NERFOLOGY

Methods

Model components

nerfstudio



Nerfstudio provides a simple API that allows for a simplified end-to-end process of creating, training, and visualizing NeRFs. The library supports an **interpretable implementation of NeRFs by modularizing each component.** With modular NeRF components, we hope to create a user-friendly experience in exploring the technology. Nerfstudio is a contributor-friendly repo with the goal of building a community where users can easily build upon each other's contributions.

It's as simple as plug and play with nerfstudio!

Conclusion

- Detection-description powerful idea, part of SFM success with classical detector-descriptors as SIFT
- Lots of classical approaches are being “deepified”, i.e. formulated as modular and end-to-end learnable framework, with important performance gains
- Tricks from classical approach often remain important in NN-architectures
- Are NeRFs the future of MVS?