Local features
detection and description

MVA/IMA – Vision 3D artificielle

Mathieu Aubry
http://imagine.enpc.fr/~aubrym/

11/3/2020

With slides from Renaud Marlet and François Darmon
What do I work on?

Historical and non photo-realistic data

Deep 3D model generation/analysis.

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

Goal: “Draw” correspondences between images

1. Intro

2. Classical approaches

3. Deep approaches
Local features and correspondences pipeline for 3D reconstruction

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

Feature description
- Summarize their appearance in a vector

Feature matching
- Find matching features

Geometric verification
- Filter the matches based on geometry
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?

Some local characteristic part of the image that we can retrieve robustly:

• Edges
• Points
• 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

Which features?

- Simple region, blobs: eg. Harris-affine
Which features?

• Points
Which features?

- distinctive/repeatable
Outline

1. Classical local features
   • 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
Convolution

- \( f, g : \mathbb{R}^d \) 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:

\[ O = K \ast I \]

e.g. uniform motion blur

Convolution

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

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

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

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

\[(K \ast I)(i, j) = \sum_{k,l} K(-k, -l)I(i + k, j + l)\]
Practice with linear filters

Original

Source: D. Lowe
Practice with linear filters

Original

Filtered
(no change)

Source: D. Lowe
Practice with linear filters

Original

Source: D. Lowe
Practice with linear filters

Original

Shifted left
By 1 pixel

Source: D. Lowe
Practice with linear filters

Original

Source: D. Lowe
Practice with linear filters

Original

Blur (with a box filter)

Source: D. Lowe
Practice with linear filters

Original

(Note that filter sums to 1)
Practice with linear filters

Original

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

Which shows changes with respect to x?

\[
\frac{\partial f(x, y)}{\partial x}
\]

\[
\frac{\partial f(x, y)}{\partial y}
\]

-1   1

or

-1    1

Source: L. Lazebnik
Filtering in frequency domain

- Fourier Transform (FFT)
- Frequency domain filtering
- Inverse Fourier Transform (Inverse FFT)

Slide: Hoiem
Effects of noise

- Consider a single row or column of the image

\[ f(x) \]

\[ \frac{d}{dx} f(x) \]

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

• 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.

The Gaussian also gives a scale to the operator/image.
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

- **Feature extraction**: Extract location of ‘local features’ (interest points, keypoints)
- **Feature description**: Summarize their appearance in a vector
- **Feature matching**: Find matching features
- **Geometric verification**: Filter the matches based on geometry
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 centered in \((0,0)\) defined by the weights \(w\) to a patch centered in \((x,y)\) using pixel intensity

\[
S(x, y) = \sum_u \sum_v w(u, v) \left( I(u + x, v + y) - I(u, v) \right)^2
\]

\(-\) if the difference is large for all \((x,y)\) the patch is distinctive.
Harris Corner

Idea: compare a patch centered in \((0,0)\) defined by the weights \(w\) to a patch centered in \((x,y)\) using pixel intensity

\[
S(x, y) = \sum_u \sum_v w(u, v) \left( I(u + x, v + y) - I(u, v) \right)^2
\]

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

\[
I(u + x, v + y) \approx I(u, v) + I_x(u, v)x + I_y(u, v)y
\]

\[
S(x, y) \approx \sum_u \sum_v w(u, v) \left( I_x(u, v)x + I_y(u, v)y \right)^2
\]
Harris Corner

Idea: compare a patch centered in \((0,0)\) defined by the weights \(w\) to a patch centered in \((x,y)\) using pixel intensity

\[
S(x, y) = \sum_u \sum_v w(u, v) \left( I(u + x, v + y) - I(u, v) \right)^2
\]

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

\[
I(u + x, v + y) \approx I(u, v) + I_x(u, v)x + I_y(u, v)y
\]

\[
S(x, y) \approx (x \ y) \sum_u \sum_v w(u, v) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}
\]
Harris Corner

- $S$ large in all direction $\iff$ condition on $\sum_u \sum_v w(u, v) \begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix}$
Spectral theorem

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

$$M = \sum_{i=1}^{m} \lambda_i e_i e_i^T \quad \text{and} \quad Me_i = \lambda_i e_i$$

$\lambda_1 \geq \ldots \geq \lambda_m \geq 0$ 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 $(e_1, \ldots, e_m)$ i.e. write

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

$$Me_i = \lambda_i e_i$$

$\lambda_1 \geq \ldots \geq \lambda_m \geq 0$ 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 $\iff$ the two eigenvalues of
  $$\sum_u \sum_v w(u,v) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$
  are large
Harris Corner

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

\[ \sum_u \sum_v w(u, v) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \]

• Harris and Stephens suggest to use

\[ M_c = \lambda_1 \lambda_2 - \kappa (\lambda_1 + \lambda_2)^2 = \det(A) - \kappa \text{trace}^2(A) \]

with \[ A = \sum_u \sum_v w(u, v) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = w \ast \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \]

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 $I_x(x_q)$ and $I_y(x_q)$ 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 $A = \begin{bmatrix} I_x^2 & I_x I_y & I_x I_y & I_y^2 \end{bmatrix}$
  - compute corner response (or strength) for each $q$
  - response above threshold and local maximum (8 neighbors) $\Rightarrow$ 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

\[
H = \begin{pmatrix}
L_{xx} & L_{xy} \\
L_{xy} & L_{yy}
\end{pmatrix}
\]

\[
L(x + dx, y + dy) \simeq L(x, y) + \nabla(L)^T(dx, dy)^T + (dx, dy)H(x, y)(dx, dy)^T
\]

Use \[\text{Tr}(H) = L_{xx} + L_{yy} = \lambda_0 + \lambda_1\] and \[\text{Det}(H) = \lambda_0 \lambda_1\]
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

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

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

• Laplacian of Gaussian (LoG) detector score:

\[ (\Delta G) \ast I = \Delta (G \ast I) = \Delta L \]

with \( L(x, y; \sigma) = G(x, y; \sigma) \ast I(x, y) \)
Multi-scale

Gaussian pyramid

- Convolution with Gaussian of varying $\sigma$

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

- Scale-space representation

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

- Scale pyramid

Space: $x, y$ dimensions (location)

Scale-space: $\sigma$ dimension
Be careful when comparing detector responses at different scales, you might need a scaling factor!

E.g. scale normalized LoG

$$\nabla_{\text{norm}}^2 G = \sigma^2 \nabla^2 G$$
Covariance / Invariance

- We want the description to be invariant: 
  \[ \text{features(transform(image)))} = \text{features(image)} \]
- A solution is to have a covariant detection: 
  \[ \text{det(transform(image)))} = \text{transform(det(image))} \]
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)

• 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 $\| x_p - x_q \| \leq r$
  - check almost largest response (e.g., within 10%):
    $\forall q \quad \| x_p - x_q \| \leq r \Rightarrow c f(x_q) \leq f(x_p)$  [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
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

$r_p \leftarrow \min_{q \in ProcessedPoints} \| x_p - x_q \| \text{ such that } f(x_p) < c f(x_q)$

[as $f(x_q) > c f(x_q)$ guaranteed for $q \in ProcessedPoints$]

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 $r = \| x_p - x_q \|$
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 ANMS:
97, 93, 94, 96

4 strongest points with NMS:
110, 105, 97, 96
ANMS : example

4 strongest points:
110, 105, 97, 96

4 strongest points with:
110, 105, 94, 93

4 strongest points with:
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 extraction
- Extract location of ‘local features’ (interest points, keypoints)

Feature description
- Summarize their appearance in a vector

Feature matching
- Find matching features

Geometric verification
- Filter the matches based on geometry
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)

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

\[
\mu_k = \frac{1}{w} \sum_{i,j} P_k(i,j) \quad \sigma_k^2 = \frac{1}{w} \sum_{i,j} (P_k(i,j) - \mu_k)^2
\]

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

BRIEF

Using binary comparisons between random locations

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

\[ f_{n_d}(p) := \sum_{1 \leq i \leq n_d} 2^{i-1} \tau(p; x_i, y_i) \]

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

- **Feature extraction**: Extract location of ‘local features’ (interest points, keypoints)
- **Feature description**: Summarize their appearance in a vector
- **Feature matching**: Find matching features
- **Geometric verification**: Filter the matches based on geometry
SIFT

• Scale Invariant Feature Transform, David Lowe, ICCV 1999, IJCV 2004
• 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
  \[ \frac{\partial G}{\partial \sigma} = \sigma \nabla^2 G \]

- Accurate localization in the 3D scale space
  \[ \tilde{D}(x) = D + \frac{\partial D^T}{\partial x} x + \frac{1}{2} x^T \frac{\partial^2 D}{\partial x^2} x \]
  \[ \frac{\partial \tilde{D}}{\partial x}(\hat{x}) = 0 \iff \hat{x} = -\frac{\partial^2 D^{-1}}{\partial x^2} \frac{\partial D}{\partial x} \]

- Rejection of unstable keypoints with low contrast / edges
  \[ \frac{\text{trace}(H)^2}{\det(H)} = \frac{(\lambda_0 + \lambda_1)^2}{\lambda_0 \lambda_1} = \frac{(r + 1)^2}{r} \]
  \[ r = \lambda_1 / \lambda_0 \]

- 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 $s \to$ scale factor of 2), resample image
  - every second pixel in each row and column
  - no accuracy loss
  - space & time efficient
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
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 expansion / 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 literature review!
There are tons of paper over the last 2-5 years,
this is a biased selection to illustrate the ideas I think are 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 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

- **Feature extraction**: Extract location of ‘local features’ (interest points, keypoints)
- **Feature description**: Summarize their appearance in a vector
- **Feature matching**: Find matching features
- **Geometric verification**: Filter the matches based on geometry
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)


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

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**

![Diagram showing the concept of triplet loss](image)

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

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

Feature extraction

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

Feature description

- Summarize their appearance in a vector

Feature matching

- Find matching features

Geometric verification

- Filter the matches based on geometry
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

\[
\text{softargmax}(S) = \frac{\sum_y \exp(\beta S(y)) y}{\sum_y \exp(\beta S(y))}
\]
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
  \[ L_{\text{desc}}(p^k_{\theta}, p^l_{\theta}) = \begin{cases} 
  \|h_\rho(p^k_{\theta}) - h_\rho(p^l_{\theta})\|_2 & \text{for positive pairs,} \\
  \max(0, C - \|h_\rho(p^k_{\theta}) - h_\rho(p^l_{\theta})\|_2) & \text{for negative pairs,} 
\end{cases} \]
- 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

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 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

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

Feature description:
Summarize their appearance in a vector

Feature matching:
Find matching features

Geometric verification:
Filter the matches based on geometry
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

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

Feature description
- Summarize their appearance in a vector

Feature matching
- Find matching features

Geometric verification
- Filter the matches based on geometry
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 dataset 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 literature 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

\[ r_{ijkl}^A = \frac{c_{ijkl}}{\max_{ab} c_{abkl}}, \quad \text{and} \quad r_{ijkl}^B = \frac{c_{ijkl}}{\max_{cd} c_{ijcd}} \]

\[ s_{ijkl}^A = \frac{\exp(c_{ijkl})}{\sum_{ab} \exp(c_{abkl})} \quad \text{and} \quad s_{ijkl}^B = \frac{\exp(c_{ijkl})}{\sum_{cd} \exp(c_{ijcd})} \]

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)

\[ \mathcal{L}(I^A, I^B) = -y (\bar{s}^A + \bar{s}^B) \]

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 literature review!
There are tons of paper over the last 2-5 years, this is a biased selection to illustrate ideas 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
Stage 1:
RANSAC on deep features
RANSAC-Flow

Stage 1:
RANSAC on deep features
Stage 1:
RANSAC on deep features
An unsupervised two-stage method

**Stage 1:**
RANSAC on deep features

**Stage 2:**
Local flow predictions
Architecture (RANSAC-Flow)
Correlation volume (RANSAC-flow, standard in OF)
RANSAC-Flow

Stage 1:
RANSAC on deep features

Stage 2:
Local flow predictions

SSIM + mask + cycle-consistency loss

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

Mask loss

Confidence at \((x,y)\)
RANSAC-Flow

Stage 1:
RANSAC on deep features

Stage 2:
Local flow predictions

\[ L_{SSIM}^{rec}(I_s, I_t) = \sum_{(x,y) \in I_t} M_{t}^{cycle}(x,y) \left( 1 - \text{SSIM}(I_s(x',y'), I_t(x,y)) \right) \]

Confident regions

\[ L_c(I_s, I_t) = \sum_{(x,y) \in I_t} M_{t}^{cycle}(x,y) \left\| (x',y'), F_{t \rightarrow s}(x,y) \right\|_2 \]
RANSAC-Flow

Stage 1:
RANSAC on deep features

Stage 2:
Local flow predictions

SSIM + mask + cycle-consistency loss
RANSAC-Flow

Stage 1: RANSAC on deep features

Stage 2: Local flow predictions

E.g.: MOCO features

SSIM + mask + cycle-consistency loss

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.  
CVPR2018
Deep MVS

\[ c_{i,j,d} = \text{var} \left( f^{ref}(i, j), f^1(i, j, d), \ldots, f^N(i, j, d) \right) \]
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 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
Parametric scene / NeRF [Mildenhall20]

\[(x, y, z, \theta, \phi) \rightarrow F_{\Theta} \rightarrow (RGB\sigma)\]

More in details

Geometry network

\[x_i \rightarrow n_i, \alpha_i\]

Radiance network

\[x_i, n_i \rightarrow c_i\]

\[d \rightarrow \mathbf{I}_r\]

\[V(p) = \sum_{i=1}^{n} \alpha_i \prod_{j<i} (1 - \alpha_j)c_i\]
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 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?