Image Processing and Computer Graphics

Image Processing

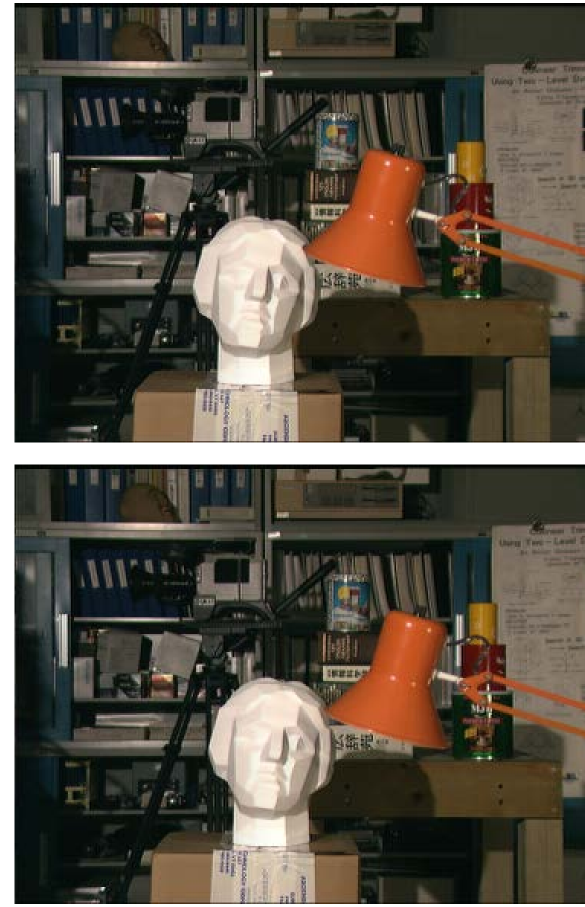
Class 8

Interest points and local descriptors

Matching of local structures

- Key problem in computer vision appearing in:
  - Motion estimation
  - Camera calibration
  - Stereo
  - Image retrieval
  - Object recognition

Object recognition: training image on the left, test image on the right. Matching here is quite hard.



Stereo pair: point matching needed to compute depth

Outline

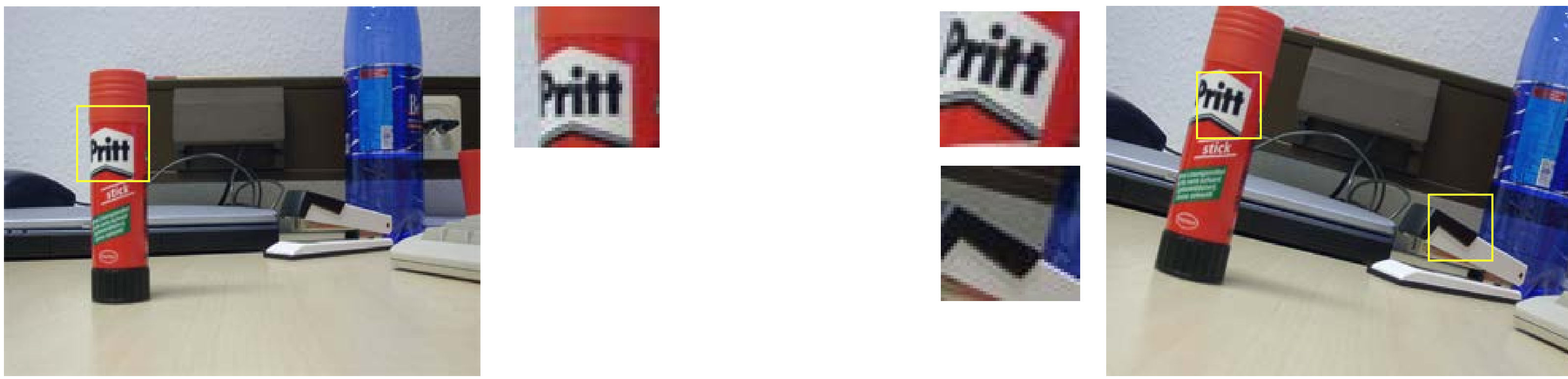
1. Interest points
   – What are good regions to match?

2. Manual descriptor design
   – What is required for a good descriptor?

3. Feature learning
   – How can we optimize descriptors for a particular task?

Block matching

- Straightforward way to match points in images:
  - Regard the image patch around each point in image 1
  - Compare it to the image patches around all points in image 2



- Computationally expensive
  $O(kN^2)$, k: size of patch, N: size of image (in pixels)
- Not **invariant** to typical appearance changes

Interest points

- Often we need only a limited number of matches
- Idea: Do not match all points in the images, but only promising subsets
  → significantly reduced complexity
- Requirements for good interest points:

1. Points must come with enough information for unique matching

2. Subset in image 2 must contain matches from subset in image 1

Corner detection

- Choose points with high information content and clear localization
  → typically corner points

- Corner detection with the structure tensor:
  (Förstner-Gülch 1987, Harris-Stevens 1988)

$$J_\rho = K_\rho * (\nabla I \nabla I^\top) = \begin{pmatrix} K_\rho * I_x^2 & K_\rho * I_x I_y \\ K_\rho * I_x I_y & K_\rho * I_y^2 \end{pmatrix}$$

- Measure of cornerness (fast to compute):

$$c = \det J_\rho - \alpha \operatorname{tr} J_\rho$$

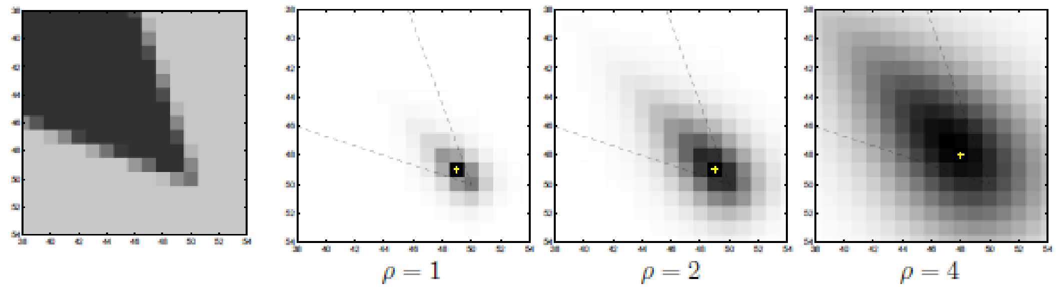
= gradient magnitude

Corner detection

- Eigenvalue decomposition of the structure tensor:

$$J_\rho = T \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} T^\top$$

$$c = \lambda_2$$



Input and second eigenvalue for different ρ

- Interpretation:
  - Smoothing of J integrates gradients from the neighborhood
  - Eigenvectors in T yield the dominant orientation in this neighborhood and the perpendicular orientation
  - Eigenvalues yield the structure magnitude in these directions
  - A large second eigenvalue indicates strong structures in multiple directions → corners

Corner detection

- Corners: local maxima of the second eigenvalue

Scale invariance

- Problem: Detected corners depend on the image scale

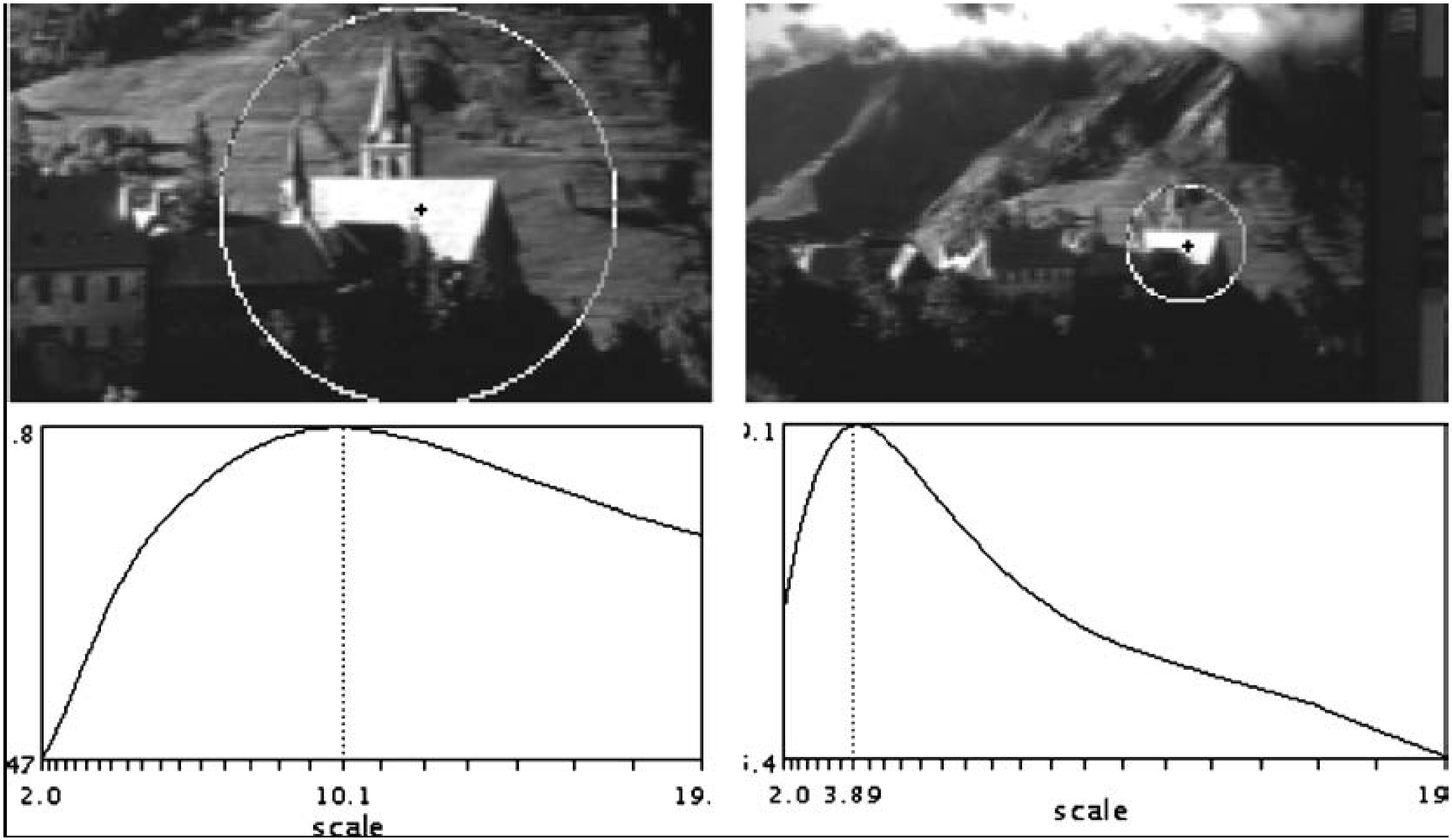
Characteristic scale

- Consider a **Gaussian pyramid** of smoothed images
- The characteristic scale can be computed based on the **Laplacian**: (Lindeberg 1998)

$$\sigma_c = \operatorname{argmax}_\sigma \left(\sigma^2 \cdot |\partial_{xx}(K_\sigma * I) + \partial_{yy}(K_\sigma * I)|\right)$$

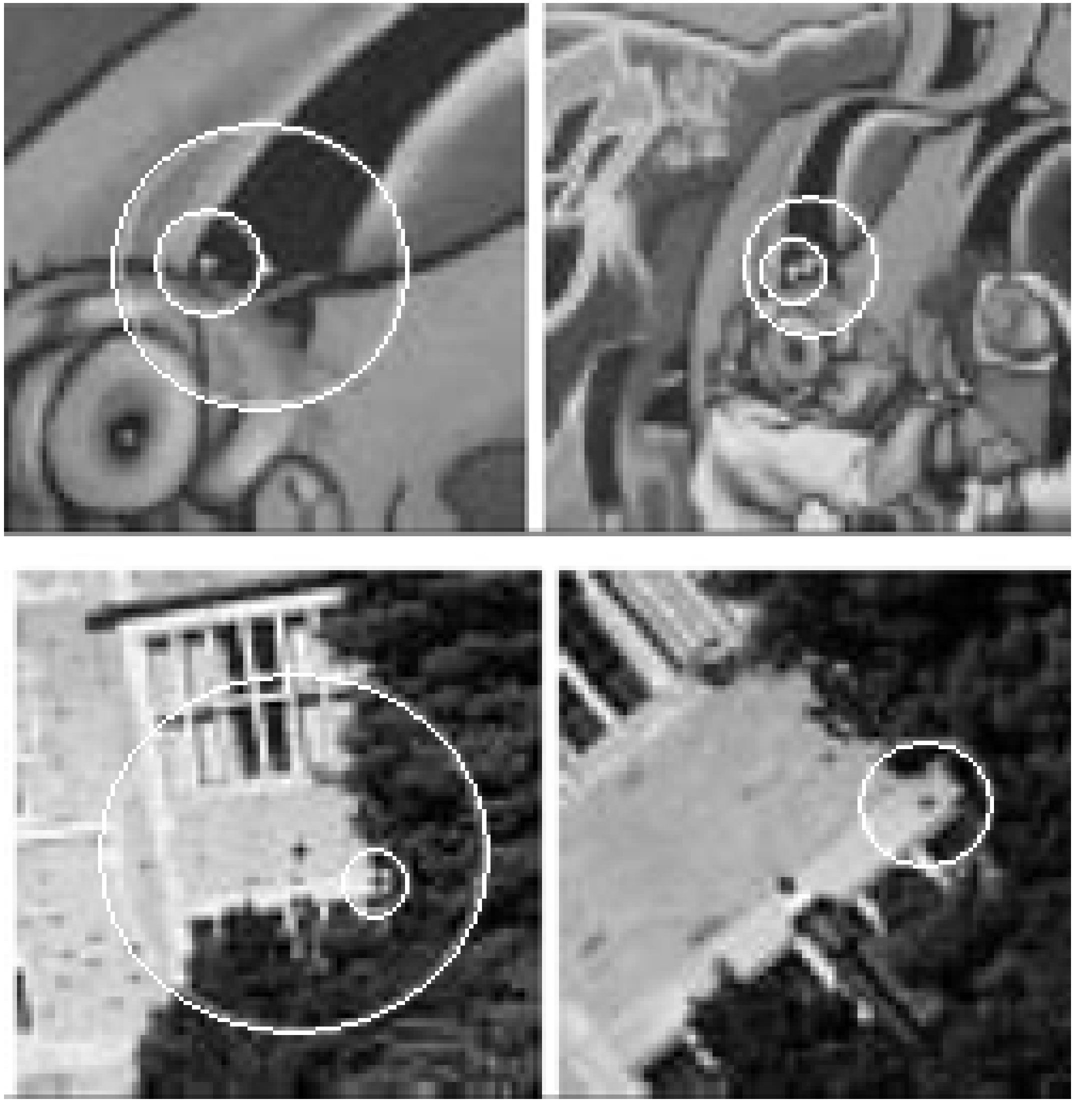
- Yields an estimate of the scale shift between two images
- Uniqueness is not ensured
  - There may be multiple maxima

Characteristic scale



Authors: Krystian Mikolajczyk and Cordelia Schmid

Harris-Laplace detector



Authors: Krystian Mikolajczyk and Cordelia Schmid

Scale invariant feature transform (SIFT)

- Alternative to Harris-Laplace detector

- Considers local maxima of the Laplacian in scale space:

$$x^*, y^*, \sigma^* = \mathrm{argmax}_{x,y,\sigma} \left(\sigma^2 \cdot |\partial_{xx}(K_\sigma * I(x,y)) + \partial_{yy}(K_\sigma * I(x,y))|\right)$$

- Advantage: Does not mix apples and oranges (corner detector and Laplacian)

- Laplacian focuses on blobs rather than corners
  → complementary information
  → one might be interested in using both

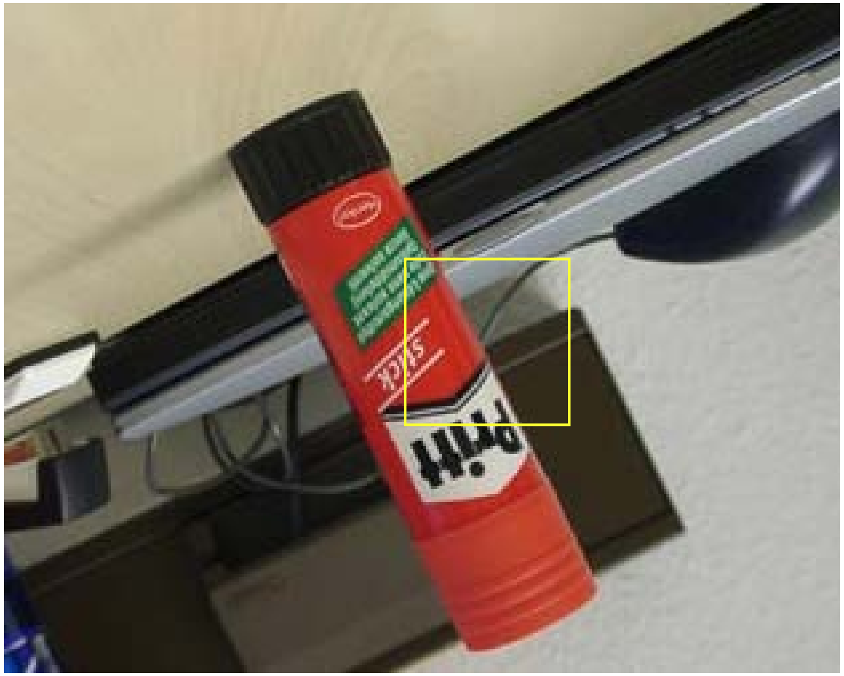
Block matching at interest points

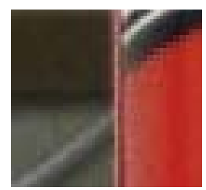
- Positive issue of interest points:
  - Significantly reduced complexity
  - With 100 detected points in both images, one has to compare only 10000 patches instead of 96 billion(!) in 640x480 images

- Negative issues:
  - Non-dense displacement fields (important matches might be missed)
  - Corresponding patches can be slightly shifted

- Further problems (independent of interest points)
  - Patches in both images may look very different due to:
    - Rotations
    - Projective transformations (different viewing angles)
    - Lighting changes (shadows, flickering)
    - Blurring
  - Subpixel accuracy not available

Example: rotation and scaling

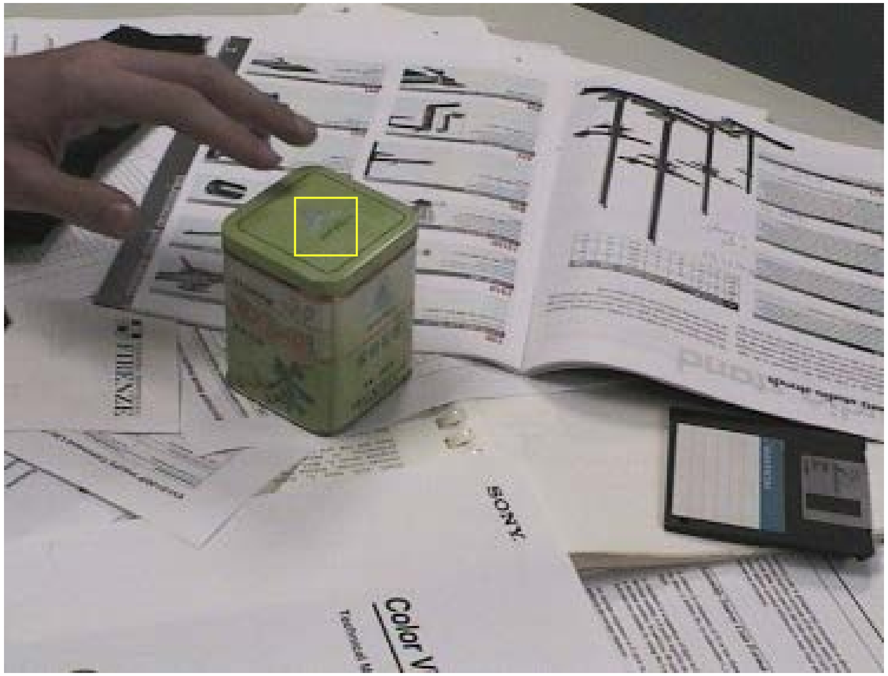


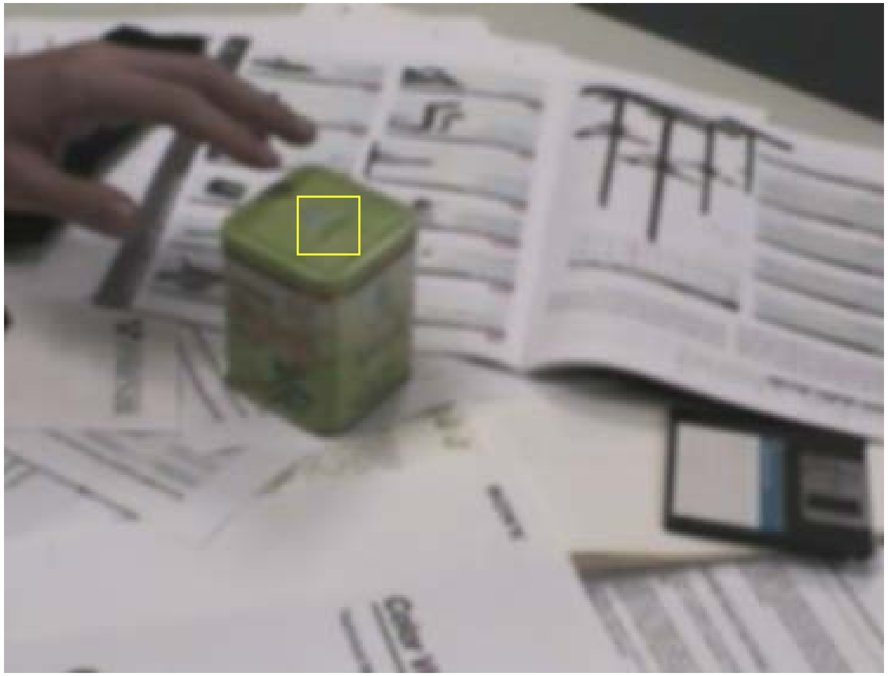


Example: projective transformation

Example: lighting changes

Example: blurring



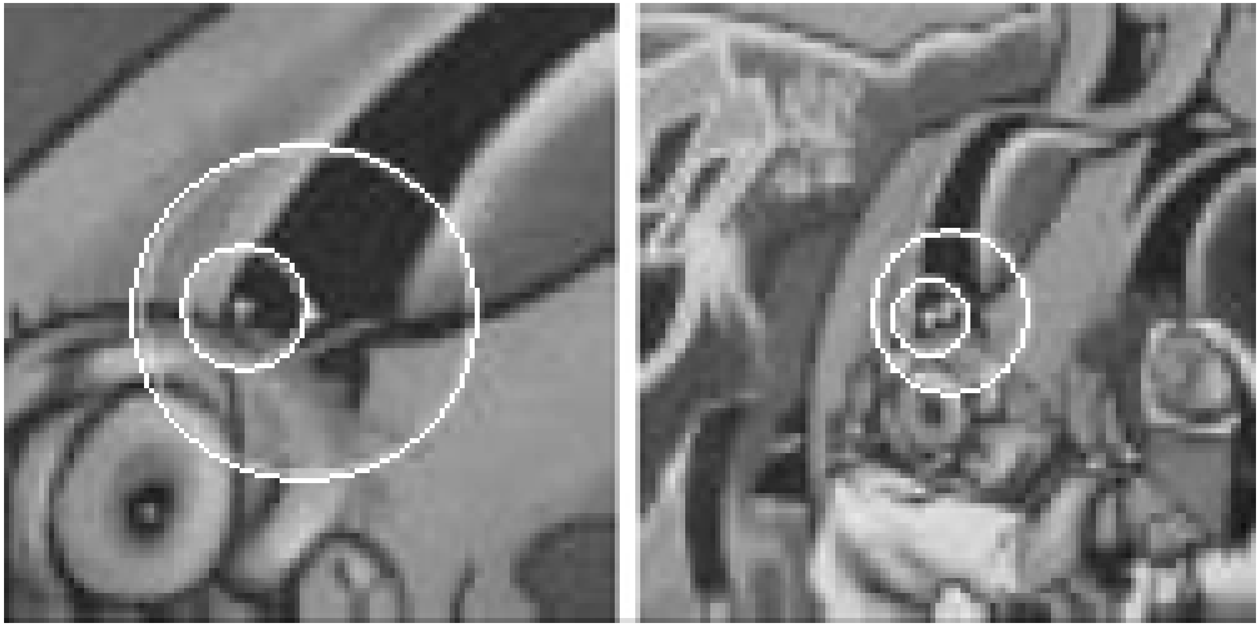


Invariant local descriptors

- **Local descriptors**: vectors that contain information about the local neighborhood of an interest point
- Simplest local descriptor: block of a certain size centered at the interest point
- Goal: design local descriptors that are **invariant** under the mentioned selected transformations
- Careful: If a descriptor is invariant to all sorts of transformations, it may not be descriptive anymore

Providing scale invariance

- Use characteristic scale from interest point detection
- Choose and normalize the size of the blocks, such that structures have the same scale in both images

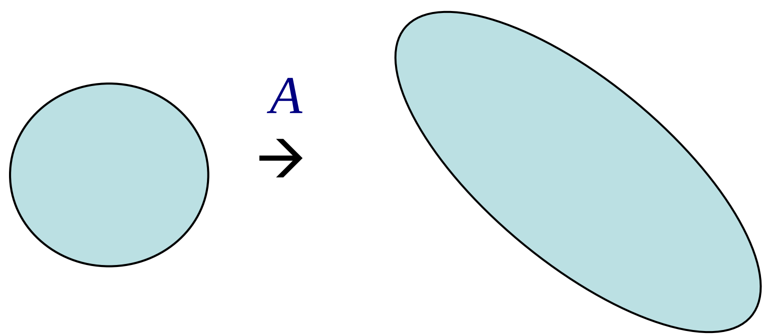


Affine invariance

- Affine transformation:

$$f(x) = Ax + t$$

- Maps a circle to an ellipse (or vice-versa):



- Approximation of a projective transformation

- Parameters can be estimated, e.g., from a region detector (maximally stable extremal regions)

Affine region detector

- Maximally stable extremal regions
(Matas et al. 2002)
  – Regions encircled by large gradients
  – Obtained by watershed-like algorithm

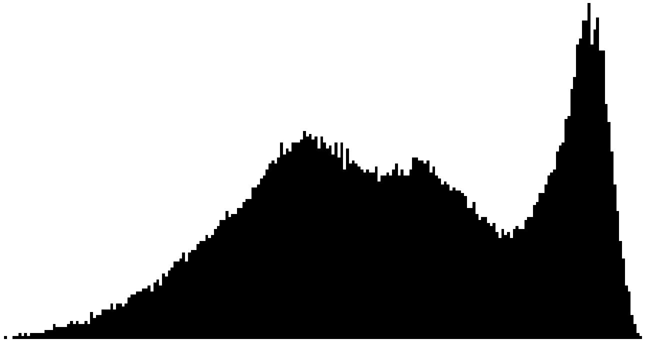
Maximally stable extremal regions and fitted ellipses. Author: Andrea Vedaldi

- Apart from scale also yields elongation of fitted ellipses
→ allows for affine invariance

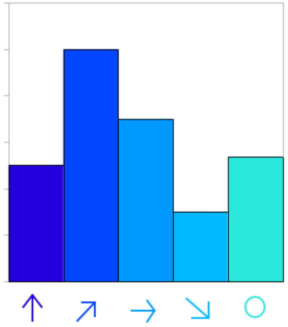
Histograms

- Alternative to a normalized neighborhood:
  derive invariant features within the fixed block

- Gray value histogram:
  – Rotational invariance
  – Invariant to blurring
  – Sensitive to lighting changes (bad)
  – Significant loss of information (very bad)

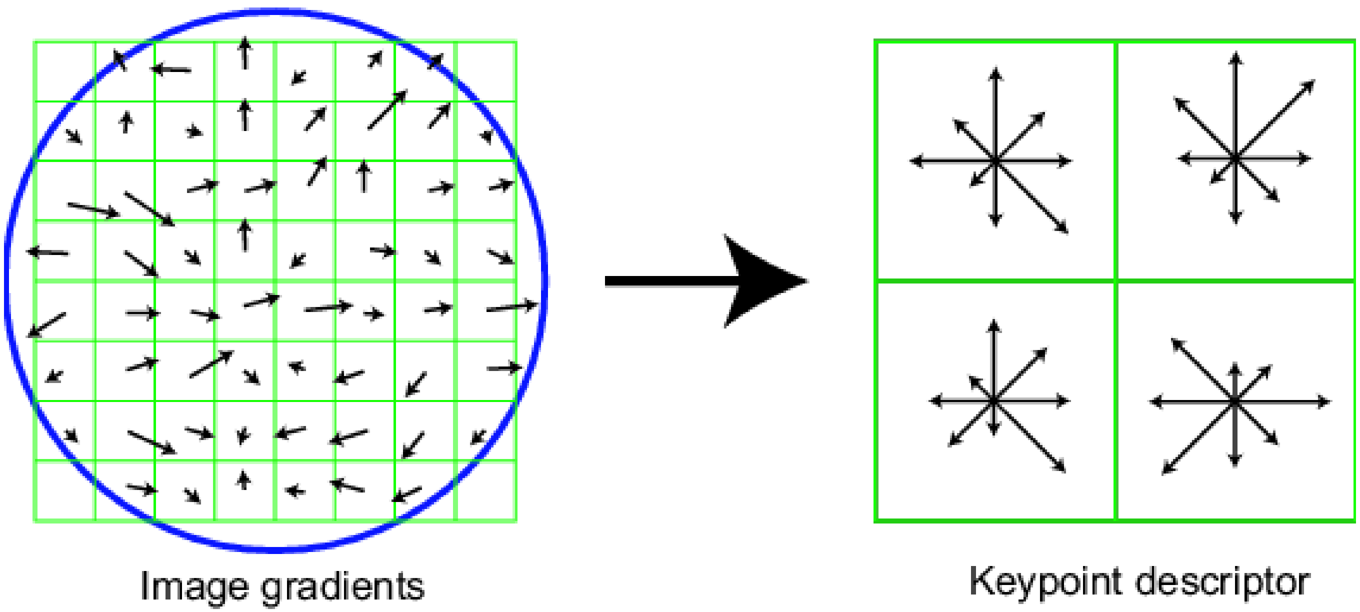


- Histogram of the gradient direction (**orientation histograms**)
  – Invariant to (additive) lighting changes
  – Building block of many successful descriptors



SIFT/HOG descriptor

- Very popular local descriptor (several variants exist)
- Based on local assembly of orientation histograms and adaptive local neighborhoods



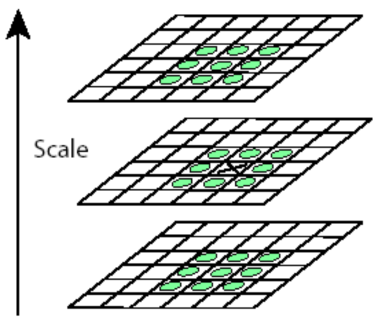
Author: David Lowe

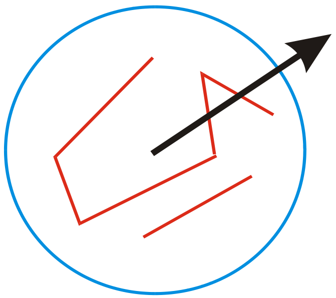
SIFT detector

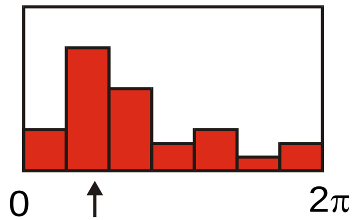
- Extract SIFT feature points
  - Strongest responses of Laplacian in scale space → position and scale
  - Fit quadratic function to obtain subpixel accuracy
- Create orientation histogram at selected scale
  - Peak of smoothed histogram estimates orientation
  - In case of two peaks, create two feature points

→ Estimation of position, scale, and orientation

- Affine invariance can be provided with MSER
- In object recognition: dense sampling of such points at all positions and all scales, no rotation invariance



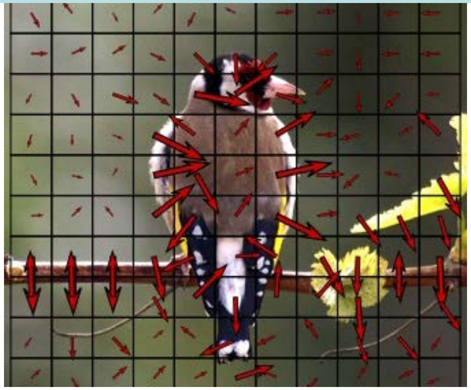




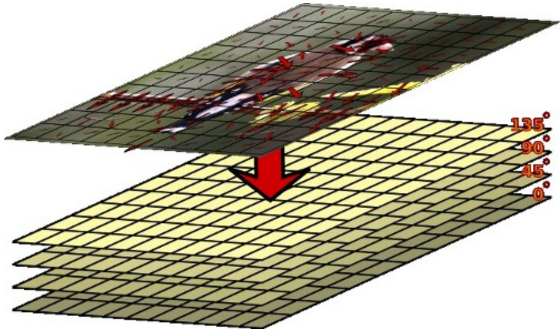
Author: David Lowe

Dense computation of SIFT/HOG descriptors

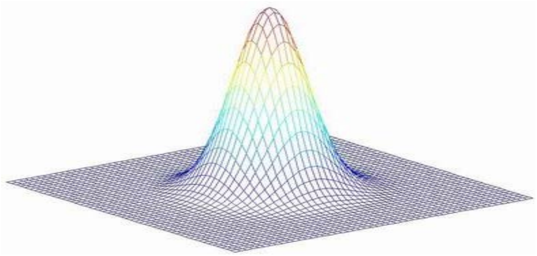
1. Compute gradient orientation and magnitude <u>at each pixel</u>



2. Compute orientation indicator at each pixel
   - Create NxMx8 array and initialize with zero
   - Quantize the orientation at each pixel (here 8 bins) and add the respective magnitude to the respective entry in the array

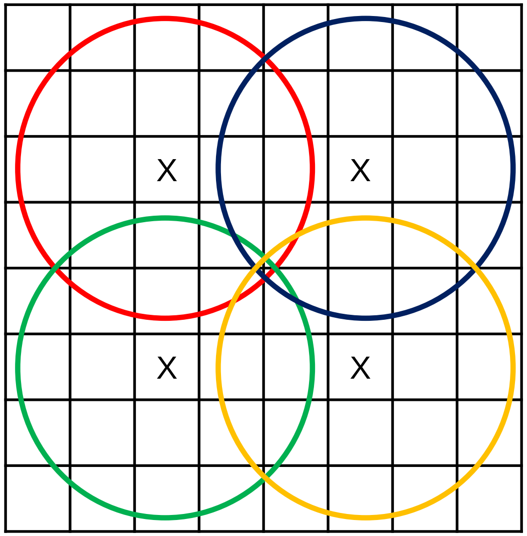


3. Local integration → orientation histogram
   Smooth array with a Gaussian kernel
4. Smooth in orientation direction
   (among neighboring channels)



Dense computation of SIFT/HOG descriptors

5. Sample feature vectors from the histogram image
   – Original SIFT:
     4 pixel spacing, 4x4 histogram array
     → 128-D vector
   – HOG for person detection (Dalal-Triggs 2005):
     6 pixel spacing, 16x8 histogram array,
     9 orientations
     → 1152-D vector



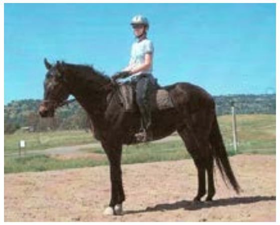
Block consisting of
4 cells

6. Normalize the feature vector
   – SIFT: normalize to unit length
   – HOG: normalize all **cells** relative to the neighbors of a **block**

Feature learning

- Instead of manual descriptor design, let the computer find the optimal descriptor for a task defined by a training set

- Example task: object classification
  → training set consists of images and their class labels



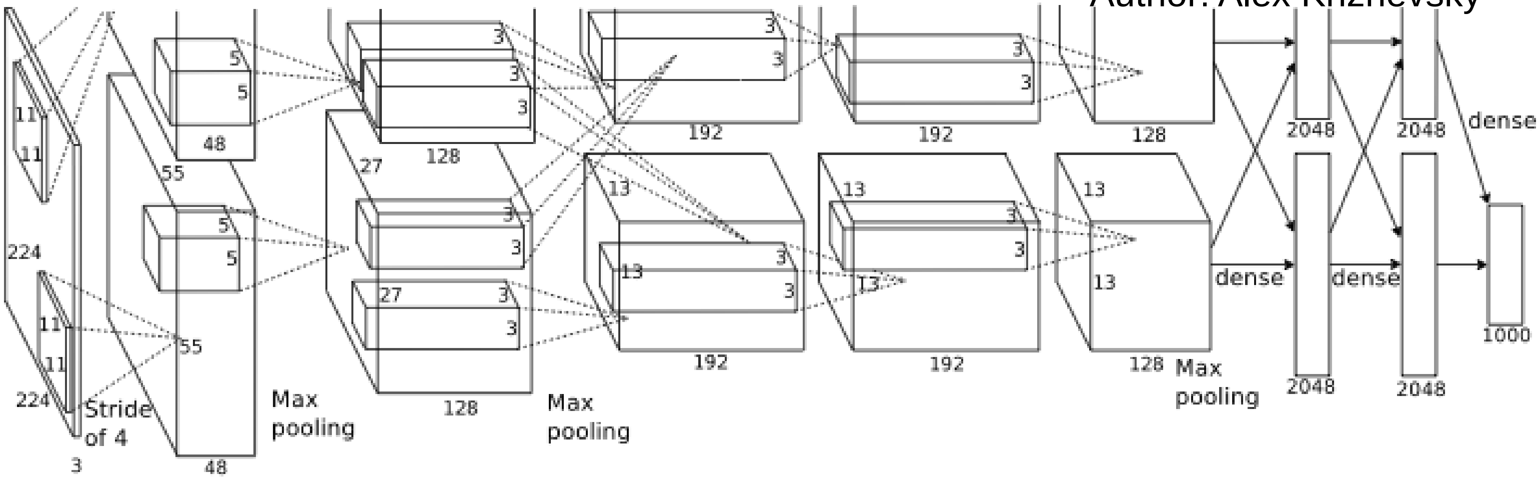
$$\text{image} \xrightarrow{f(I)} \text{features} \xrightarrow[\text{classifier}]{c(f(I))} \boxed{\text{Class}}$$

- Shallow modeling of the function $f(I)$ is not efficient to cover all the variation that appears in an object class
  → hierarchy of functions, “deep” representation

Descriptors learned with convolutional networks

Author: Alex Krizhevsky

Input image



Class output

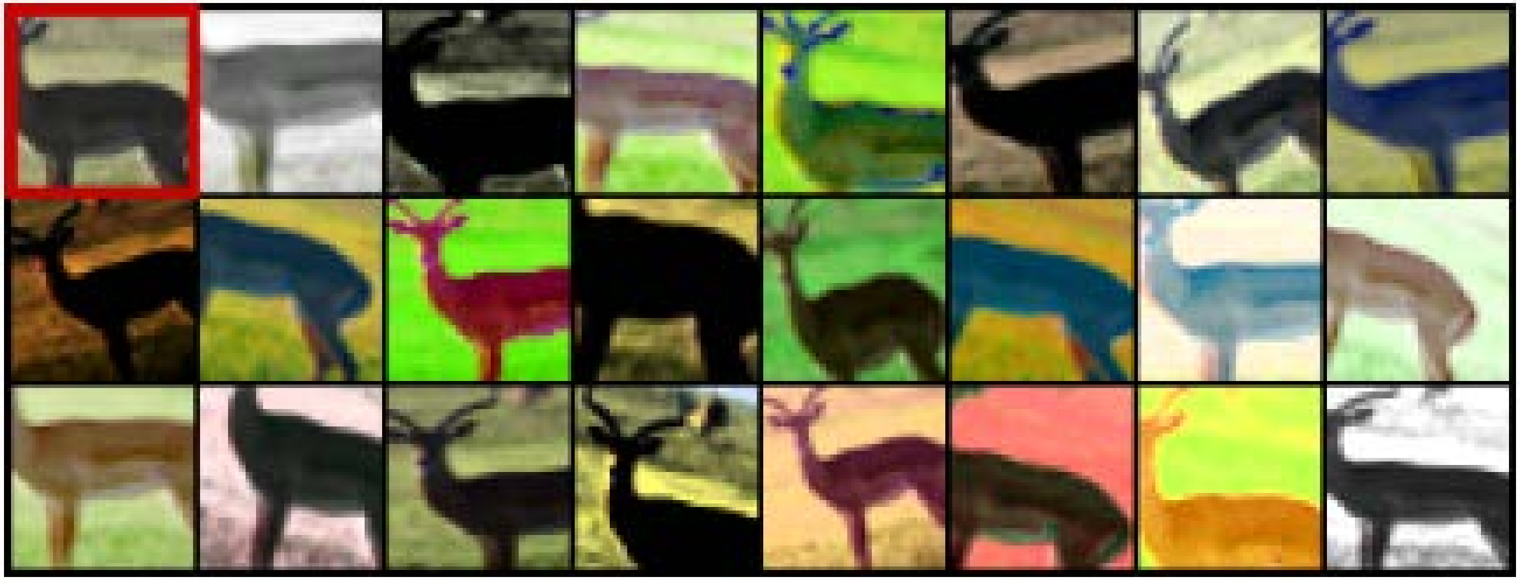
Intermediate representations

- Classification networks are trained on large datasets with class labels (e.g. ImageNet with 1M images)
  → network learns a representation that is good for object classification

- Intermediate layer outputs turn out to be good generic descriptors

Unsupervised training to trigger invariant features

- Train CNN to discriminate **surrogate classes** (Dosovitskiy et al. 2015)

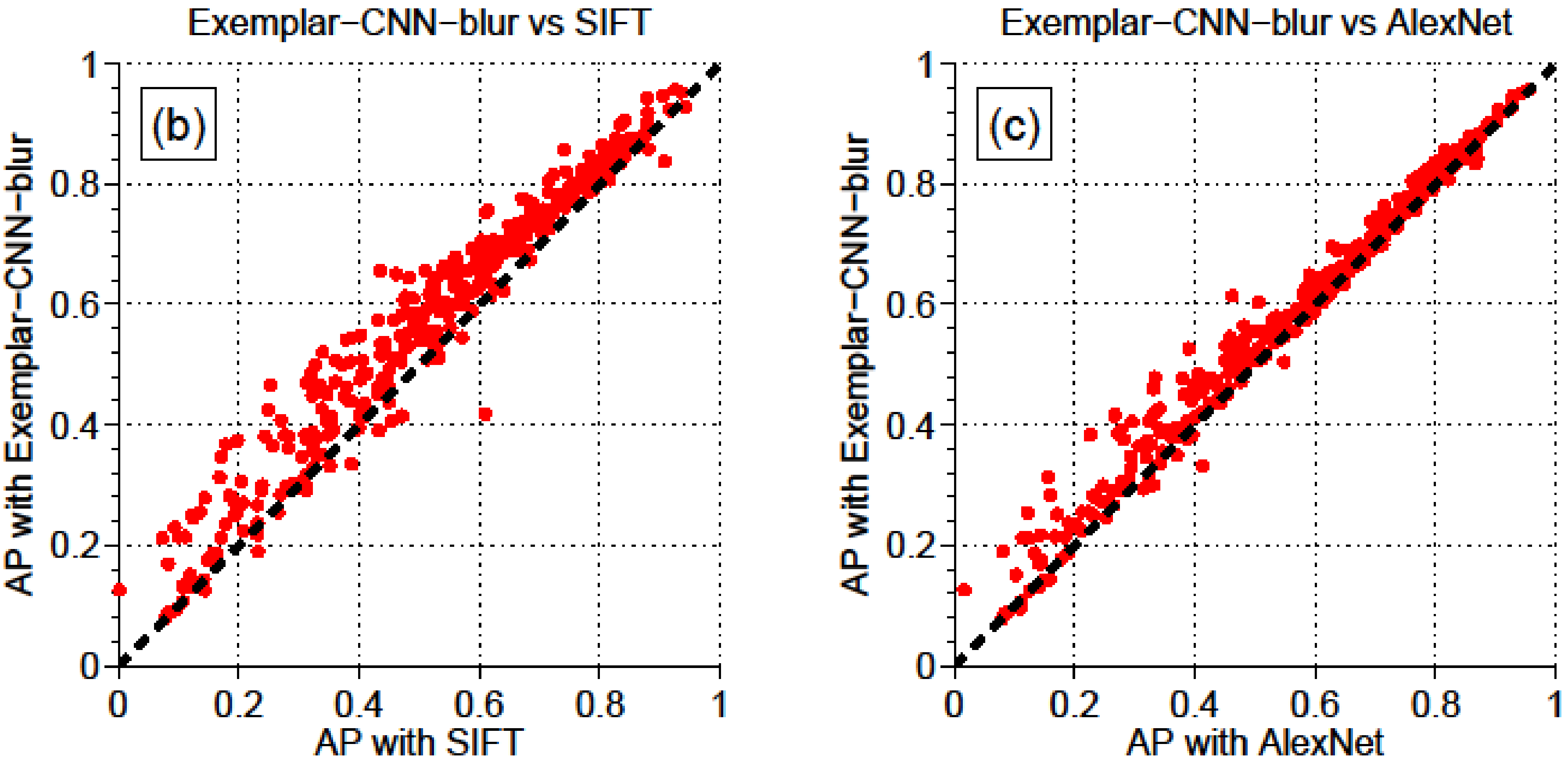


Seed patch and transformed versions of it make up a surrogate class

- Applied transformations:
  translation, rotation, scaling, color, contrast, brightness, blur

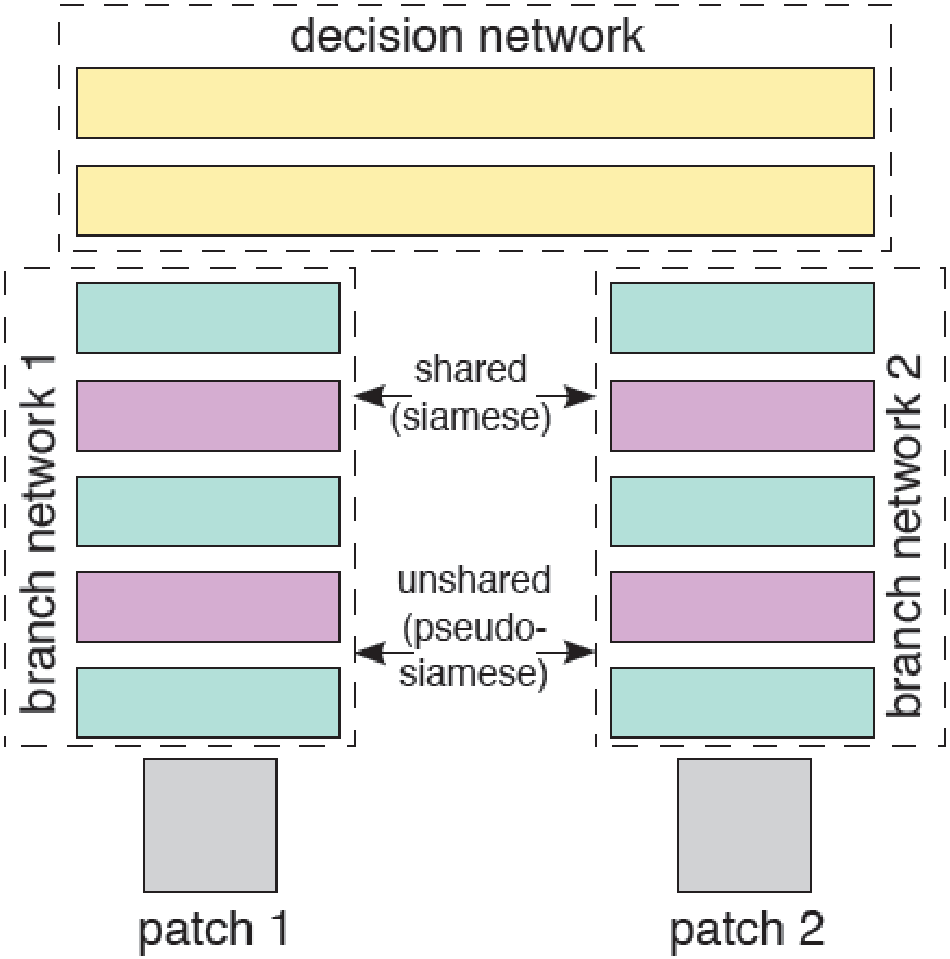
- Transformations define invariance properties of the features to be learned by the network

Descriptor matching performance



Siamese networks

- Trained directly on matching and non-matching patches



Learns the metric

Learns the descriptor

Example from Zagoruyko&Komodakis 2015

Summary

- Interest points are distinctive points in an image with a significant information content in their neighborhood
- Interest point detection can help establish invariance to certain image transformations.
- Local descriptors describe a local area in the image for the purpose of matching.
- The SIFT descriptor is based on a grid of orientation
- Intermediate layers of convolutional networks yield good descriptors

References

- T. Lindeberg: Feature detection with automatic scale selection, International Journal of Computer Vision, 30(2): 79-116, 1998.
- D. G. Lowe: Distinctive image features from scale-invariant keypoints, International Journal of Computer Vision 60(2):91-110, 2004.
- J. Matas, O. Chum, M. Urban, T. Pajdla: Robust wide baseline stereo from maximally stable extremal regions, Proc. British Machine Vision Conference, 2002.
- N. Dalal, B. Triggs: Histograms of oriented gradients for human detection, CVPR, 2005.
- S. Belongie, J. Malik, J. Puzicha: Shape matching and recognition using shape contexts, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):509-522, 2002.
- A. Dosovitskiy, P. Fischer, T. Springenberg, M. Riedmiller, T. Brox: Discriminative unsupervised feature learning with convolutional neural networks, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2016.
- S. Zagoruyko, N. Komodakis: Learning to Compare Image Patches via Convolutional Neural Networks, *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

Programming assignment

- Implement the corner detector based on the second eigenvalue of the structure tensor
  - For computing derivatives and for smoothing images you can make use of the predefined filter masks as well as the convolution operations in `CFilter.h`.
  - The structure tensor of a color image is the sum of tensors over all channels
  - See an online math lecture if you do not remember how to compute the eigenvalues of a matrix http://www.khanacademy.org/
- Apply the corner detector to the images in `ImageProcessing08Ex03.zip` and play with the parameters
- Implement the dense SIFT/HOG descriptor (without the detector). Use a 4 pixel spacing and a 3x3 grid of histograms. You can ignore scale and rotation invariance and even skip normalization for this exercise.
- Run your corner detector on `tennis500.ppm` and manually select among the interest points the 10 visually most interesting ones. Compute SIFT descriptors for these points.
- Compute SIFT descriptors for all points in `tennis505.ppm`. For each descriptor in `tennis500.ppm` find the best match in `tennis505.ppm` and visualize them in your result image.
- Play with the amount of smoothing, the spacing, and the number of histograms when computing the descriptors.