Kernels for Computer Vision

- Adapted from Christoph H. Lampert : Kernel Methods in Computer Vision, in Foundations and Trends in Computer Graphics and Vision, 2009
- DOI: 10.1561/0600000027
- Partiellement sur Google Books

Problems to tackle

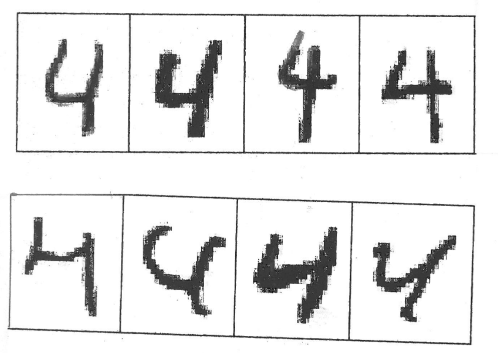
- Optical character recognition
- Object classification
- Action recognition
- Image segmentation (decomposition/partition into homogeneous regions)
- Content based image retrieval: queries

Universal Swiss Knife: Kernel methods!

Image kernels: how?

- Image: NxM pixels, values in [0..255] or {0,1}, hence vectors in $\mathbf{R}^{NM}$
- NB: very large dimension!
- Kernels:
  -- vectorial data: linear, gaussian
  -- binary images: polynomial useful
- Images should be of same sizes
- Caveat: Images are not vectors or sets of pixels! Additional 2D structure is essential!
- Example: figures, letters, etc.

Example



- A series of fours: insignificant variations in appearance and huge discrepancy for euclidean distances
- Invariants: translations, small rotations, small changes in size, blur, brightness, contrast, e.g.

Incorporating invariance

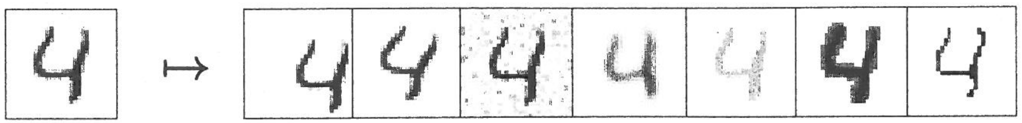
- Design SVM decision function: $f(x) = \Sigma\ \alpha_i\ k(x_i,x)$ (remember kernels' properties)
- k, kernel, large enough when x close to some sample x_i unless $f(x) \approx 0$
- Gaussian kernels of common use, although sensitive to slight geometric distorsions
- Main trouble comes from sharpness of Euclidean distances

4 remedies

- Extend training set
- Normalize images
- Define invariant kernels
- Work with invariant representations

Training set

- Apply small transformations to basic image: enrich samples with those “commonly” seen
- Automate the process
- Example: the figure 4 through translation, (small) rotation, blur, contrast changes, stroke width increase, thinning.



Normalization (1)

- Brightness, contrast: $x' = (x-x_{min})/(x_{max}-x_{min})$
- Position: moment normalization (binary pixels)

$$m = \sum_{i,j} x[i,j]$$

$$m_x = \frac{1}{m} \sum i\, x[i,j] \quad m_y = \frac{1}{m} \sum j\, x[i,j]$$

$$m_{xx} = \frac{1}{m} \sum (i-m_x)^2 x[i,j] \quad m_{yy} = \frac{1}{m} \sum (j-m_y)^2 x$$

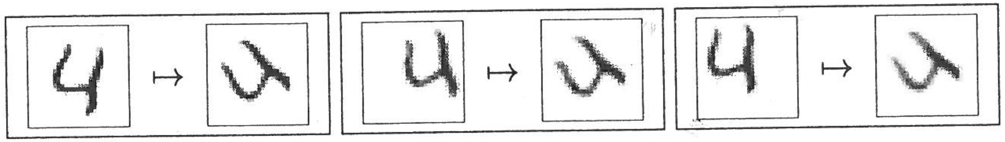
$$m_{xy} = \frac{1}{m} \sum (i-m_x)(j-m_y)\, x[i,j]$$

shift image by $-m_x, -m_y$

rotate image by $\alpha = \arctan(v_y/v_x)$ where $\begin{pmatrix} v_x \\ v_y \end{pmatrix} = \text{eig.V.} \begin{bmatrix} m_{xx} & m_{xy} \\ m_{yx} & m_{yy} \end{bmatrix}$

Normalization (2)

(a) *Illumination invariance* by brightness and contrast normalization. The right images are derived from the left by linearly transforming the gray values such that their minimum is 0 and their maximum is 1.



(b) *Geometric invariance* by moment normalization. The right images is derived from the left ones by translation and rotation such that $m_x = m_y = m_{xy} = 0$ and $m_{xx} = m_{yy} = 1$.

Fig. 3.3 Invariance through *image normalization*: transforming the image to a *standard representation* can remove the effect of image variability.

Normalizing on Gaussian kernel matrix

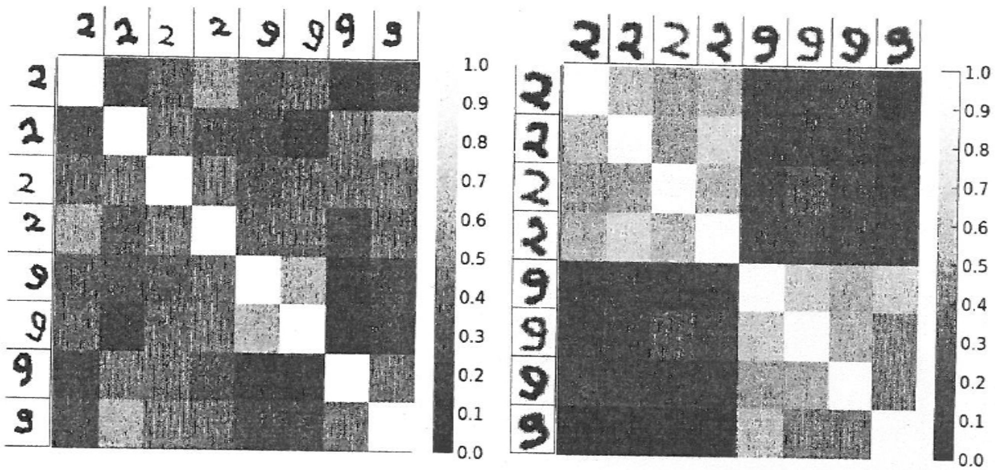


Fig. 3.4 Effect of normalization on Gaussian kernel matrix: raw input samples (left) can have large Euclidean distance even if they belong to the same class. This leads to a non-informative kernel. Normalizing each sample to uniform size, translation, and rotation (right) makes images of the same class more similar to each other, causing a more informative kernel matrix.

Invariant Kernel Functions

- Normalization as a part of the kernel itself
- No global reference frame
- Transformations to compute the distance between two samples are estimated
- Set of distorsion operators **J** and for J in **J**,
  $$d_{match}(x,x') = \min_J \| x - J(x') \|^2.$$
  Typically **J** is set of rotations, translations, small variations, small distorsions, etc.
- Drawback: time consuming!

Invariant kernels

- ==> Use tangent approximations of J's
  $$d_{tang}(x,x') = \min_J \| x - (x'+\delta t') \|^2$$
- t' is the small deplacement and δ the linear approximation of the distorsion J.
- From d_{match} and d_{tang}, one can compute (invariant)(gaussian) kernels:
  $$k_{inv}(x,x') = \exp(-\gamma(d(x,x')+d(x',x)))$$
- Symmetric and Positive!

Invariant representations

- Perform a geometric or analytic or bitmap (etc.) transformation on the original space, then use an appropriate kernel in the image space...
- Integration-based invariants:
  -- Fourier transform
  -- Haar integrals equivalent to the image left by 2π rotation of the original
- Edge-based invariants:
  gradients, thresholds, etc. (GIMP)
- Histograms (!)

Haar-integral transform

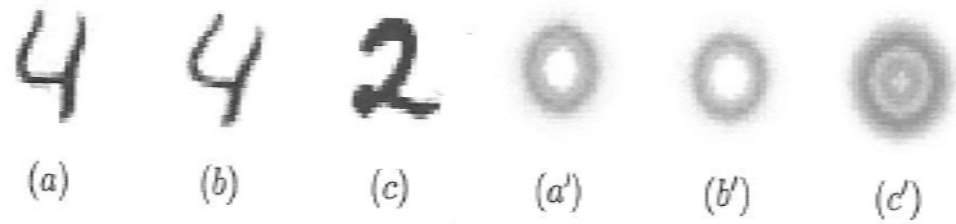


Fig. 3.5 Rotation invariant features by integration. (a),(b): The two digits 4 differ by a rotation, which makes them dissimilar in a Euclidean sense. (a′),(b′): Averaging over all their rotated versions yields very similar images. (c),(c′): The average over all rotated versions of a digit 2 image yields a feature image clearly different from (a′) and (b′).

Edge detection

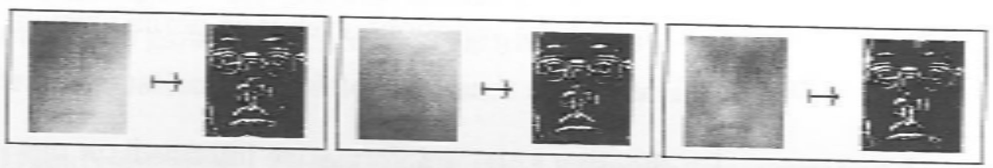


Fig. 3.6 Edge detection as invariant preprocessing step: image differentiation followed by thresholding removes global intensity variations while keeping the location of edges.

Histograms (1)

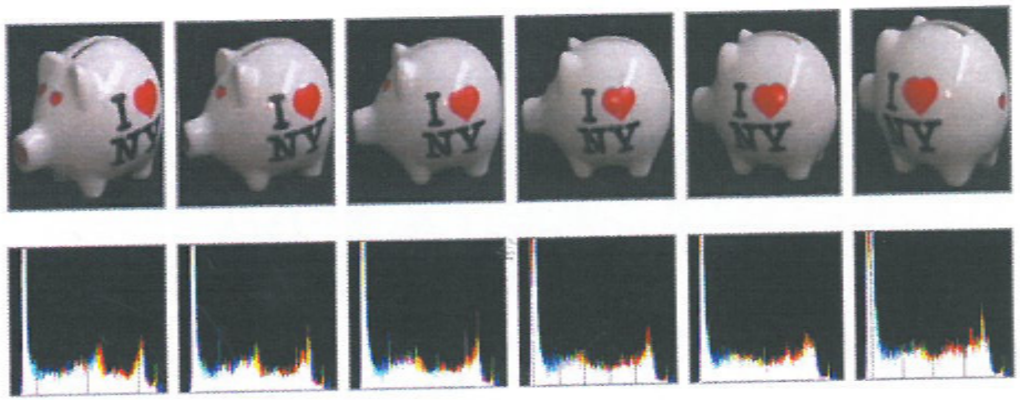


Fig. 3.7 Color histograms as perspective invariant features: the same object results in different images when viewed from different viewpoints. Nearly the same parts occur, but in different locations. The images differ when treated as vectors, but their color histograms are almost identical. Images: Colorado Object Image Library [73]

Histograms (2)

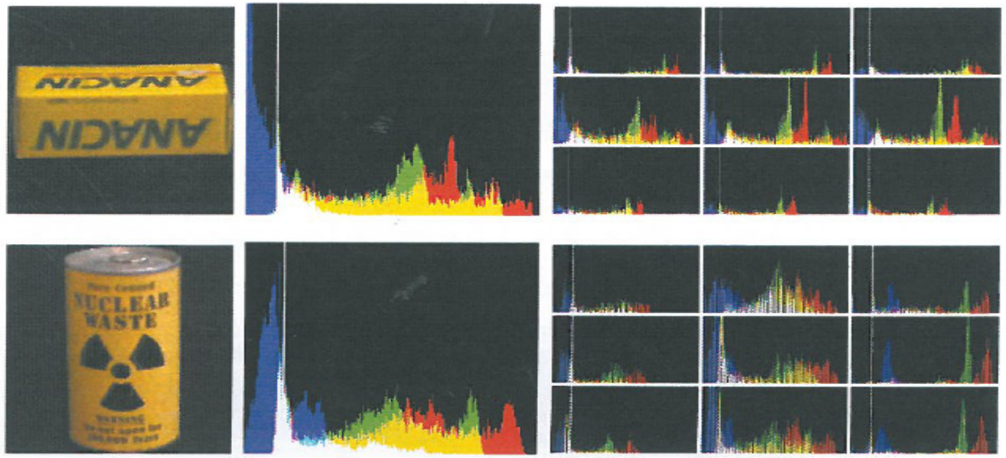


Fig. 3.8 Color histograms ignore all geometric relations, therefore very different objects (left) can have similar histogram representations (middle). By forming histograms over smaller image regions, e.g., the cells of a 3 × 3 grid (right), the feature map can be made more specific. On a global level it retains geometry but locally it is still invariant to changes, e.g., due to perspective. Images: Colorado Object Image Library [73].

Other criteria

- Interest points (low-level filters): differences of gaussian or wavelets coefficients
- Allow to characterize descriptors (by subsets of 50 to 200) or sub-areas of the pics
- Typical kernels:
  -- sum of distances of homologous points
  -- best fit of descriptors
- Drawback: quadratic computational cost
- Solution: pyramid match kernel; organize embedded structures

And also...

- Feature codebooks: *K*-means clustering and Voronoy cells

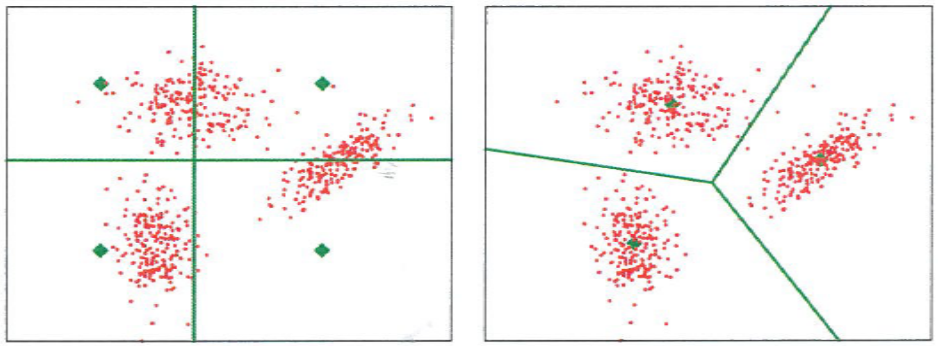


Fig. 3.10 Descriptor Codebooks: The descriptors that occur in natural images do not lie uniform in the space of all possible descriptors, but they form clusters. Axis-parallel subdivisions (left) do not respect the cluster structure and can cut through regions of high density. Clustering the descriptors followed by vector quantization (right) divides the descriptor space into Voronoy cells that respect the cluster structure.

Further topics

- Classification: multiclass SVMs
- Outlier detection in $\mathbf{R}^d$
- Regression
- Dimensionality reduction (Principal Component Analysis, Discriminant Analysis)
- Clustering
- Kernels for Regular Expressions
- Etc.