Learning on images with segmentation graph kernels

Zaïd Harchaoui
Telecom, Paris

TELECOM PARIS
école nationale supérieure des télécommunications

Francis Bach
Ecole des Mines de Paris

MINES PARIS

May 2007

Outline

- Learning on images
- Kernel methods
- Segmentation graph kernels
- Experiments
- Conclusion

Learning tasks on images

- Multiplication of digital media
- Many different tasks to be solved
  - Associated with different machine learning problems

Image retrieval

Classification, ranking, outlier detection

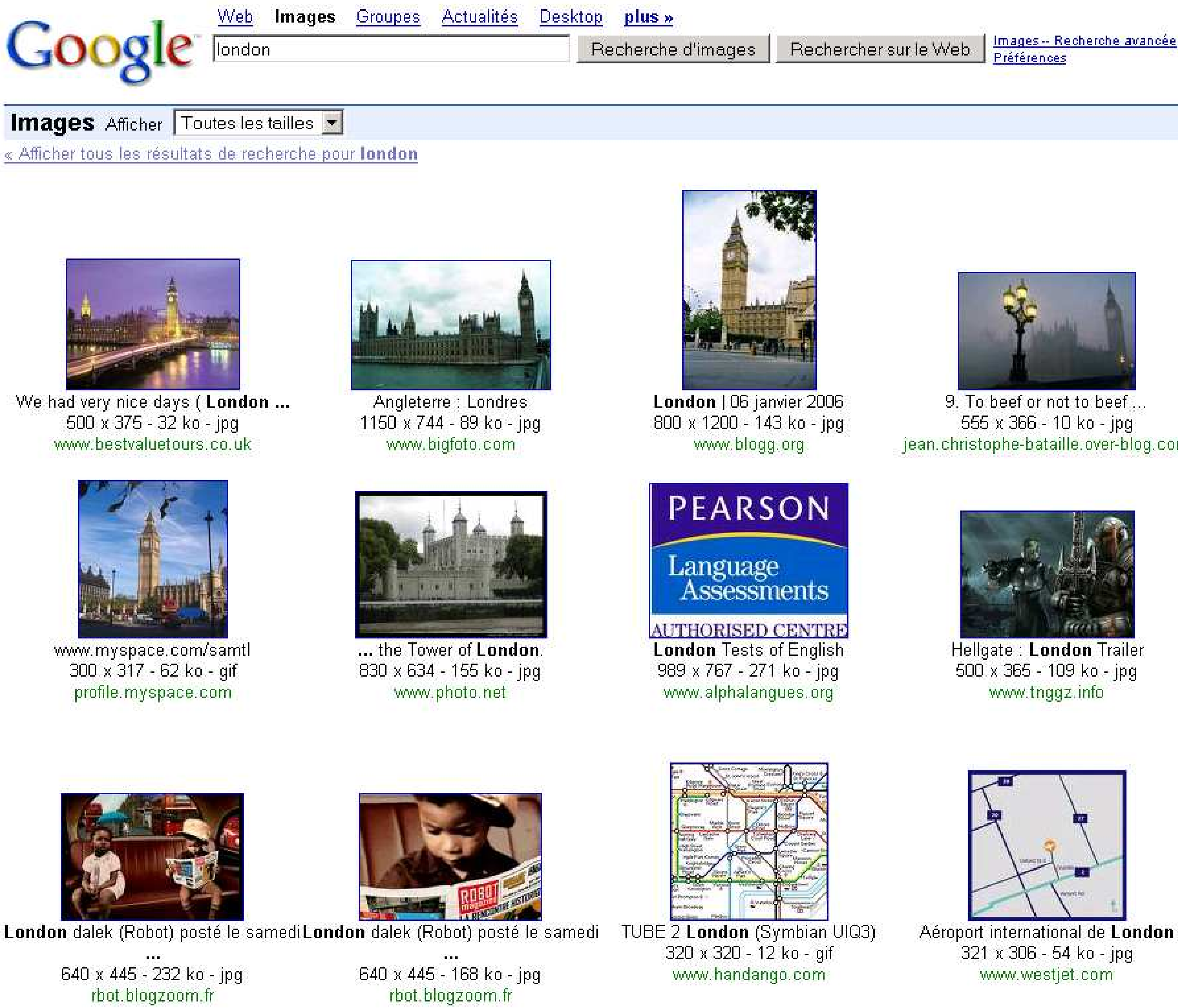


Image retrieval
Classification, ranking, outlier detection

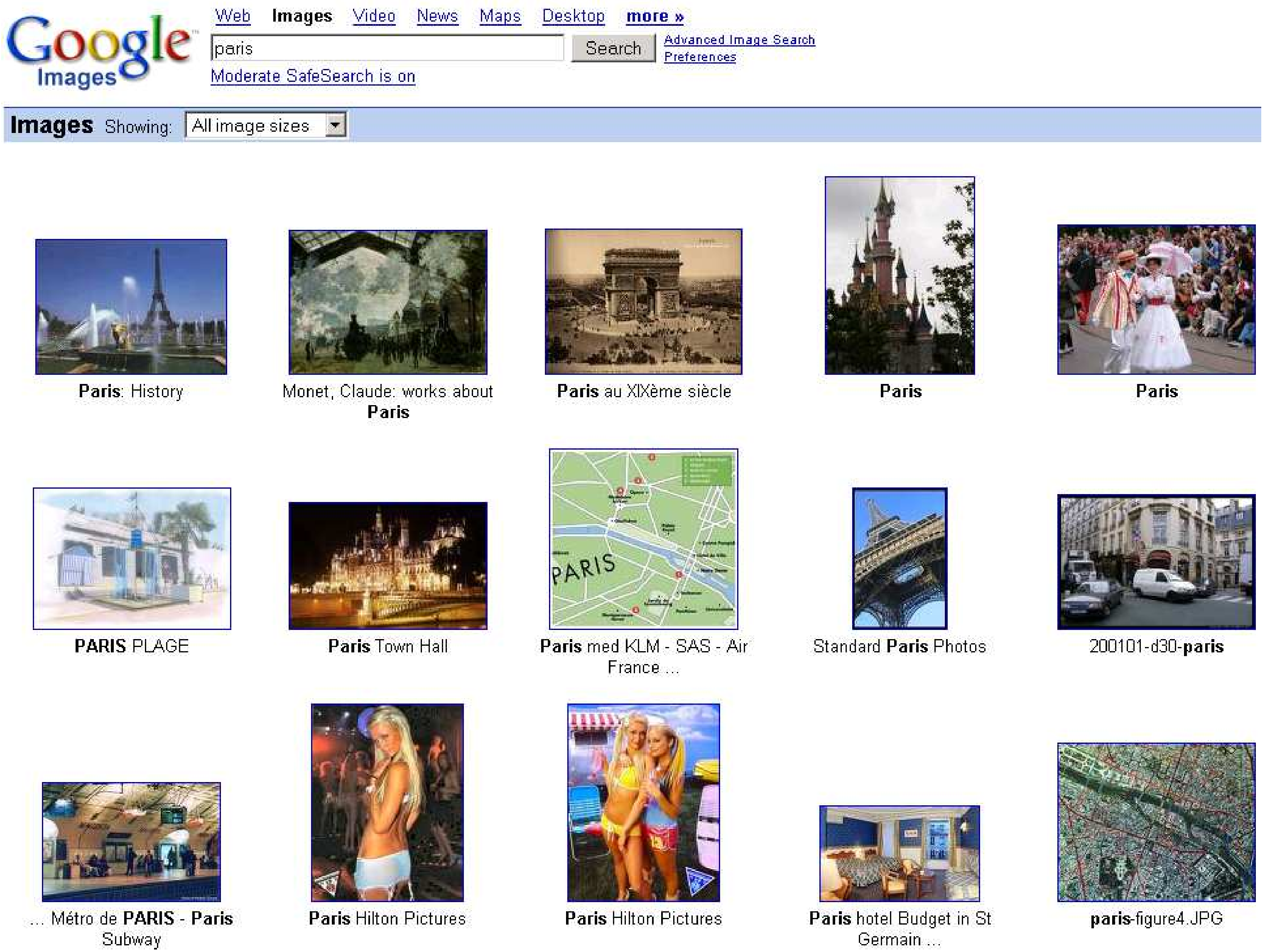
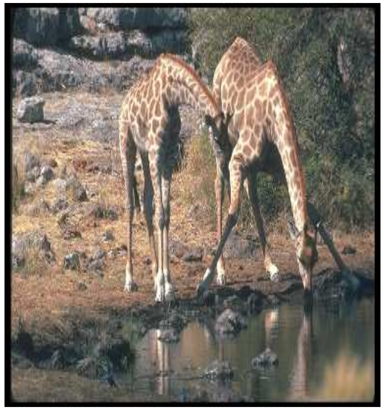


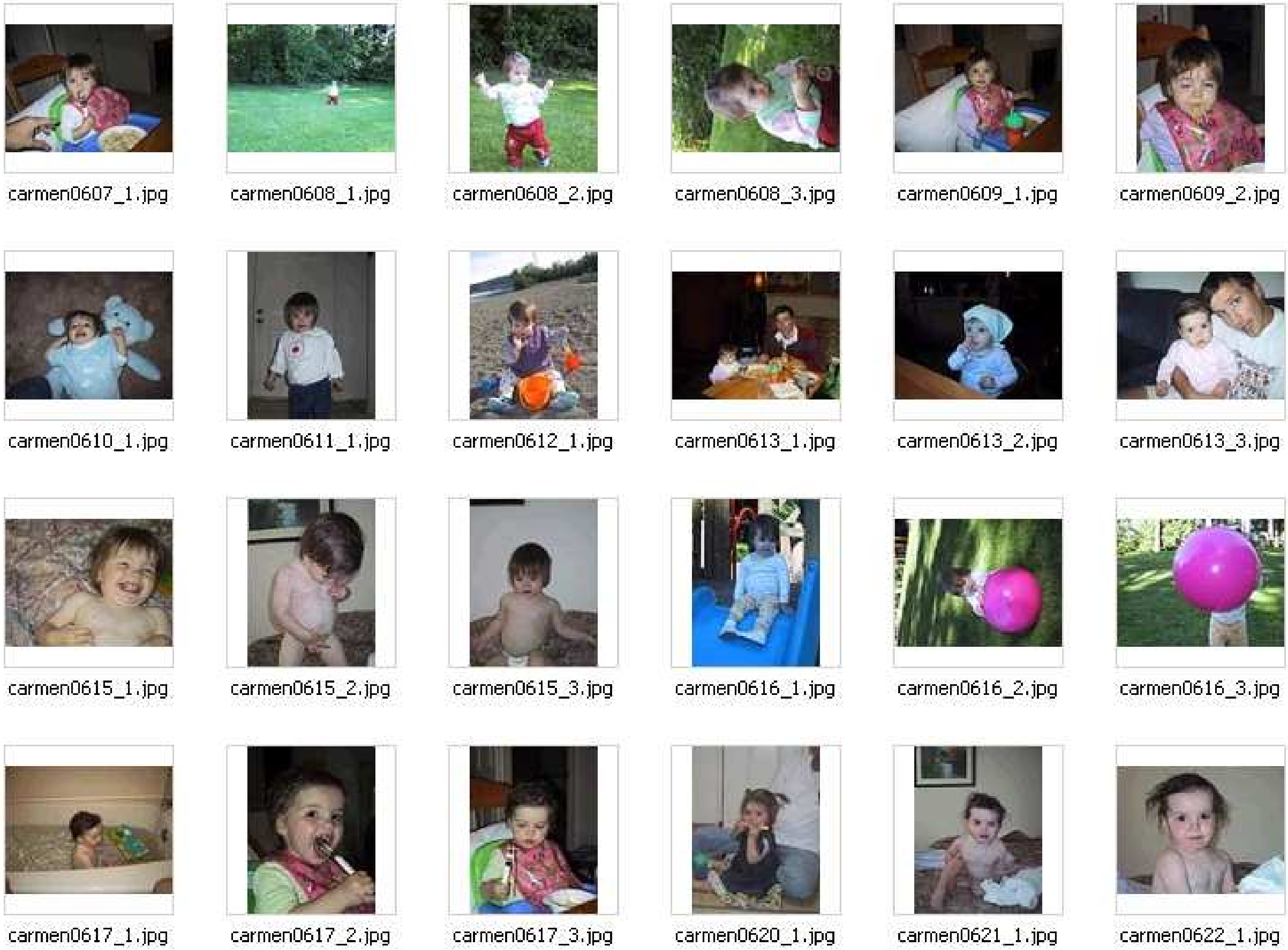
Image annotation
Classification, clustering






Personal photos

Classification, clustering, visualisation



Learning tasks on images

- Multiplication of digital media
- Many different tasks to be solved
  - Associated with different machine learning problems
- Application: retrieval/indexing of images
- Common issues:
  - Complex tasks
  - Heterogeneous data – links with other medias (text and sound)
  - Massive data

Learning tasks on images

- Multiplication of digital media
- Many different tasks to be solved
  - Associated with different machine learning problems
- Application: retrieval/indexing of images
- Common issues:
  - Complex tasks
  - Heterogeneous data – links with other medias (text and sound)
  - Massive data

$\Rightarrow$ **Kernel methods**

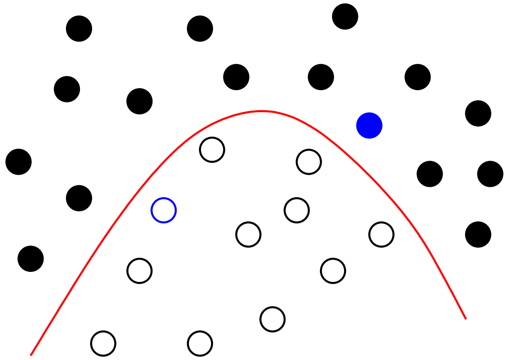
Kernel methods for machine learning

- **Motivation**:
  - Develop modular and versatile methods to learn from data
  - Minimal assumptions regarding the type of data (vectors, strings, graphs)
  - Theoretical guarantees

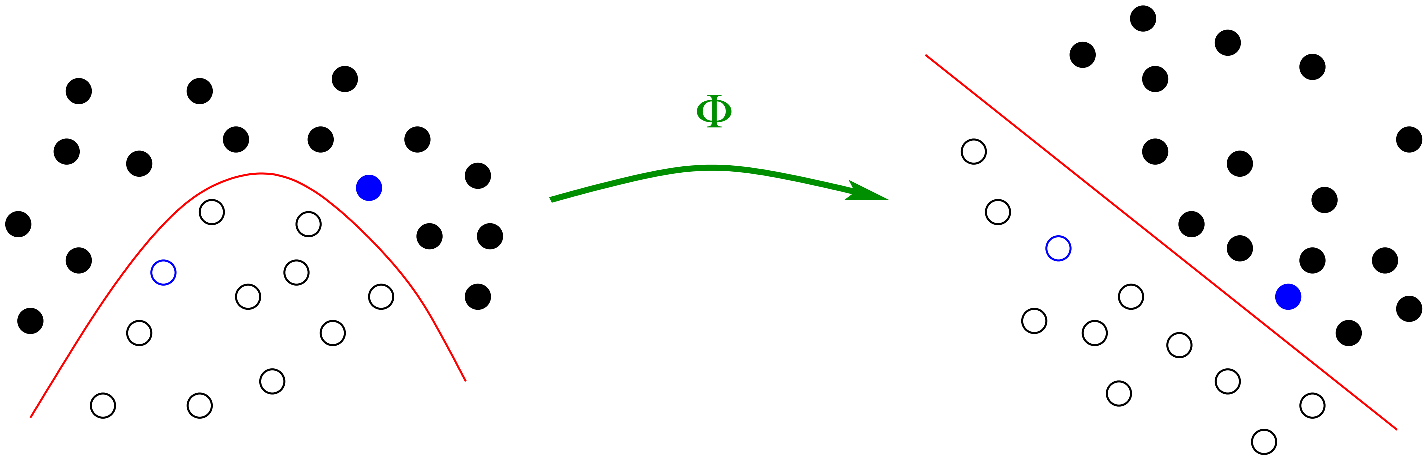
Kernel methods for machine learning

- **Motivation**:
  - Develop modular and versatile methods to learn from data
  - Minimal assumptions regarding the type of data (vectors, strings, graphs)
  - Theoretical guarantees
- **Main idea**:
  - use only pairwise comparison between objects through dot-products
  - use algorithms that depend only on those dot-products ("linear algorithms")

Kernel trick : linear $\Rightarrow$ non linear



Kernel trick : linear $\Rightarrow$ non linear



- Non linear map $\Phi : x \in \mathcal{X} \mapsto \Phi(x) \in \mathcal{F}$
- Linear estimation in "feature space" $\mathcal{F}$
- Assumption: results only depend on dot products $\langle \Phi(x_i), \Phi(x_j) \rangle$ for pairs of data points
- Kernel: $k(x, x') = \langle \Phi(x), \Phi(x') \rangle$
- Implicit embedding!

Kernel methods for machine learning

- **Definition**: given a set of objects $\mathcal{X}$, a positive definite kernel is a symmetric function $k(x, x')$ such that for all finite sequences of points $x_i \in \mathcal{X}$ and $\alpha_i \in \mathbb{R}$,

$$\sum_{i,j} \alpha_i \alpha_j k(x_i, x_j) \geqslant 0$$

  (i.e., the matrix $(k(x_i, x_j))$ is symmetric positive semi-definite)

- **Aronszajn theorem** (1950): k is a positive definite kernel if and only if there exists a Hilbert space $\mathcal{F}$ and a mapping $\Phi : \mathcal{X} \mapsto \mathcal{F}$ such that

$$\forall (x, x') \in \mathcal{X}^2, \; k(x, x') = \langle \Phi(x), \Phi(x') \rangle_{\mathcal{H}}$$

- $\mathcal{X} =$ "input space", $\mathcal{F} =$ "feature space", $\Phi =$ "feature map"
- Functional view: reproducing kernel Hilbert spaces

Kernel trick and modularity

- Kernel trick: any algorithm for finite-dimensional vectors that only uses pairwise dot-products can be applied in the feature space.
  - Replacing dot-products by kernel functions
  - Implicit use of (very) large feature spaces
  - Linear to non-linear learning methods

Kernel trick and modularity

- Kernel trick: any algorithm for finite-dimensional vectors that only uses pairwise dot-products can be applied in the feature space.
  - Replacing dot-products by kernel functions
  - Implicit use of (very) large feature spaces
  - Linear to non-linear learning methods

- Modularity of kernel methods
  1. Work on new algorithms and theoretical analysis
  2. Work on new kernels for specific data types

Kernel algorithms

- Classification and regression
  - Support vector machine, linear regression, etc...
- Clustering
- Outlier detection
- Ranking
- Integration of heterogeneous data

$\Rightarrow$ Developed independently of specific kernel instances

Kernels : kernels on vectors $x \in \mathbb{R}^d$

- Linear kernel $k(x,y) = x^\top y$
  - Linear functions
- Polynomial kernel $k(x,y) = (r + s x^\top y)^d$
  - Polynomial functions
- Gaussian-RBF kernels $k(x,y) = \exp(-\alpha \|x-y\|^2)$
  - Smooth functions
- Structured objects? Choice of parameters?

Kernels for images

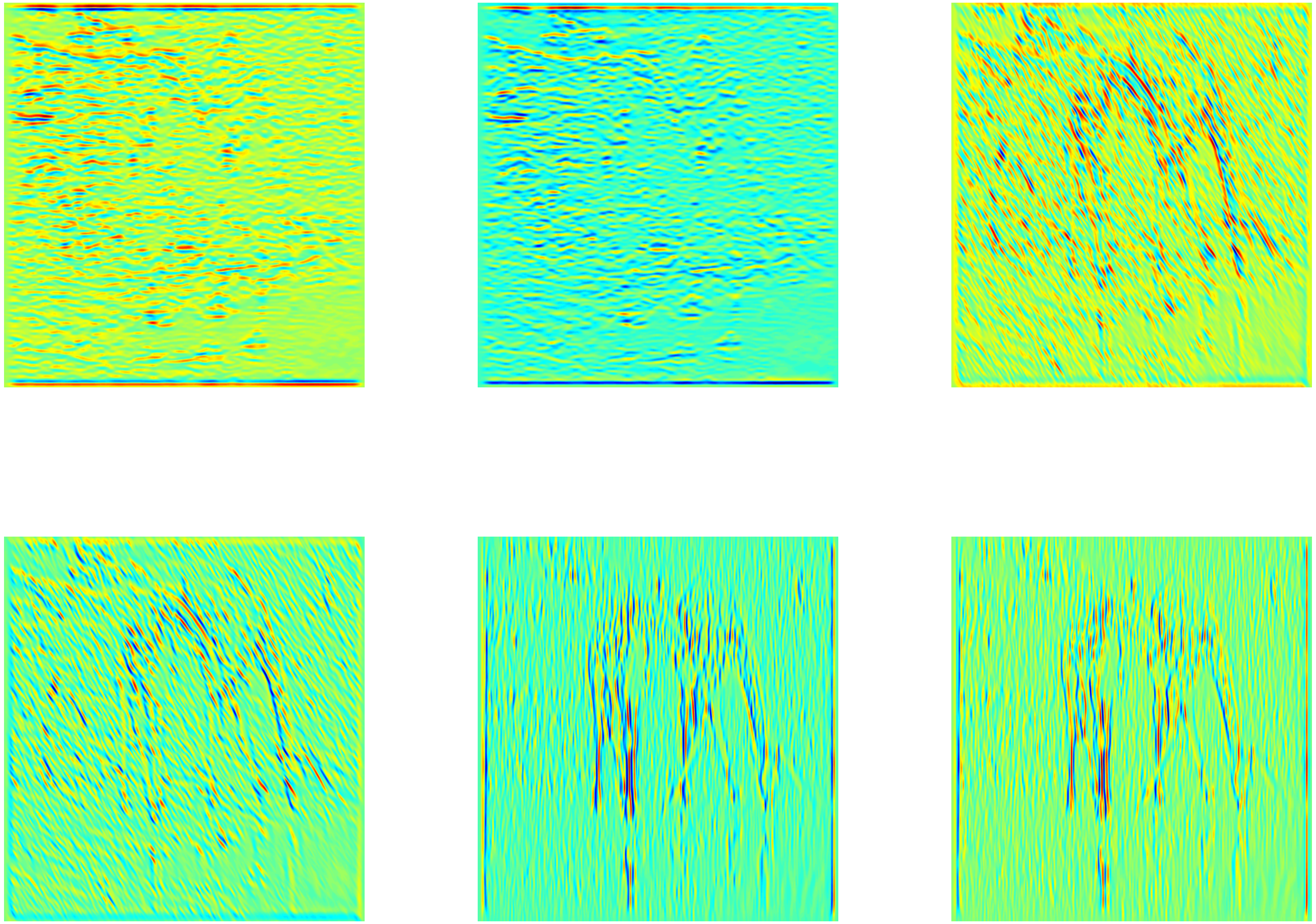
- Most applications of kernel methods to images
  - Compute a set of features (e.g., wavelets)
  - Run an SVM with many training examples
- Why not design specific kernels?
  - Using natural structure of images beyond flat wavelet representations
  - Using prior information to lower the number of training samples

kernel methods for images

- "Natural" representations
  - Vector of pixels + kernels between vectors (most of learning theory!)
  - Bags of pixels: leads to kernels between histograms
    (Chapelle & Haffner, 1999, Cuturi et al, 2006)
  - Large set of hand-crafted features (e.g., Osuna and Freund, 1998)

Input picture

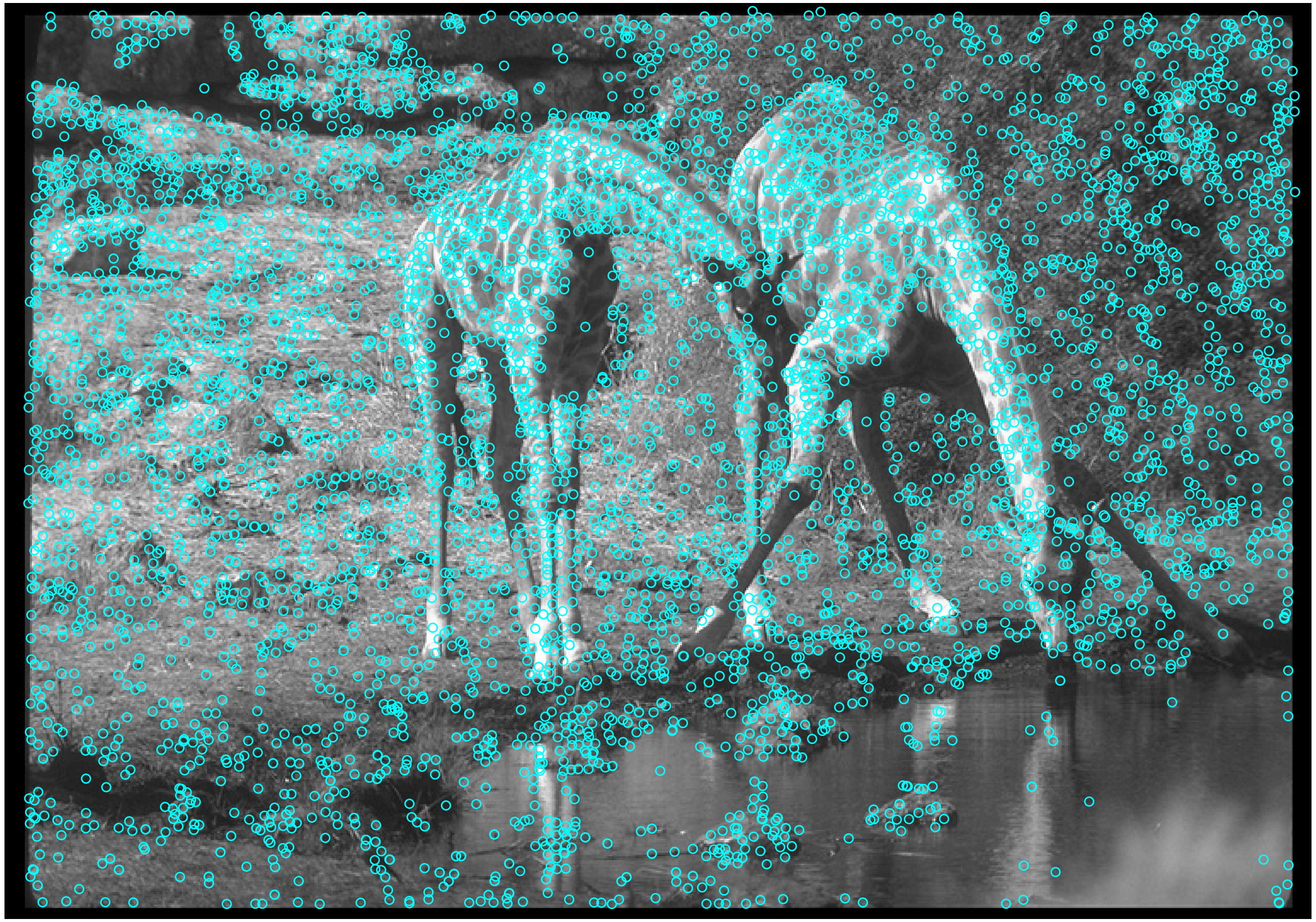
Wavelets



kernel methods for images

- "Natural" representations
  - Vector of pixels
  - Bags of pixels
  - Large set of hand-crafted features
- Loss of natural global geometry
  - Often requires a lot of training examples
- Natural representations
  - Salient points (SIFT features, Lowe, 2004)
  - Segmentation

SIFT features



Segmentation

- Goal: extract objects of interest
- Many methods available, ...
  - ... but, rarely find the object of interest entirely
- Segmentation graphs
  - Allows to work on “more reliable” over-segmentation
  - Going to a large square grid (millions of pixels) to a small graph (dozens or hundreds of regions)

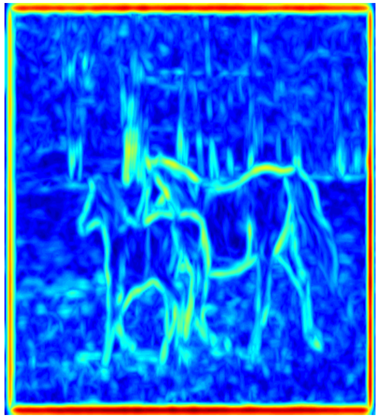
Image as a segmentation graph

- Segmentation method
  - LAB Gradient with oriented edge filters (Malik et al, 2001)
  - Watershed transform with post-processing (Meyer, 2001)
  - Very fast!

Watershed

image

gradient



watershed

287 segments

64 segments

10 segments

Watershed

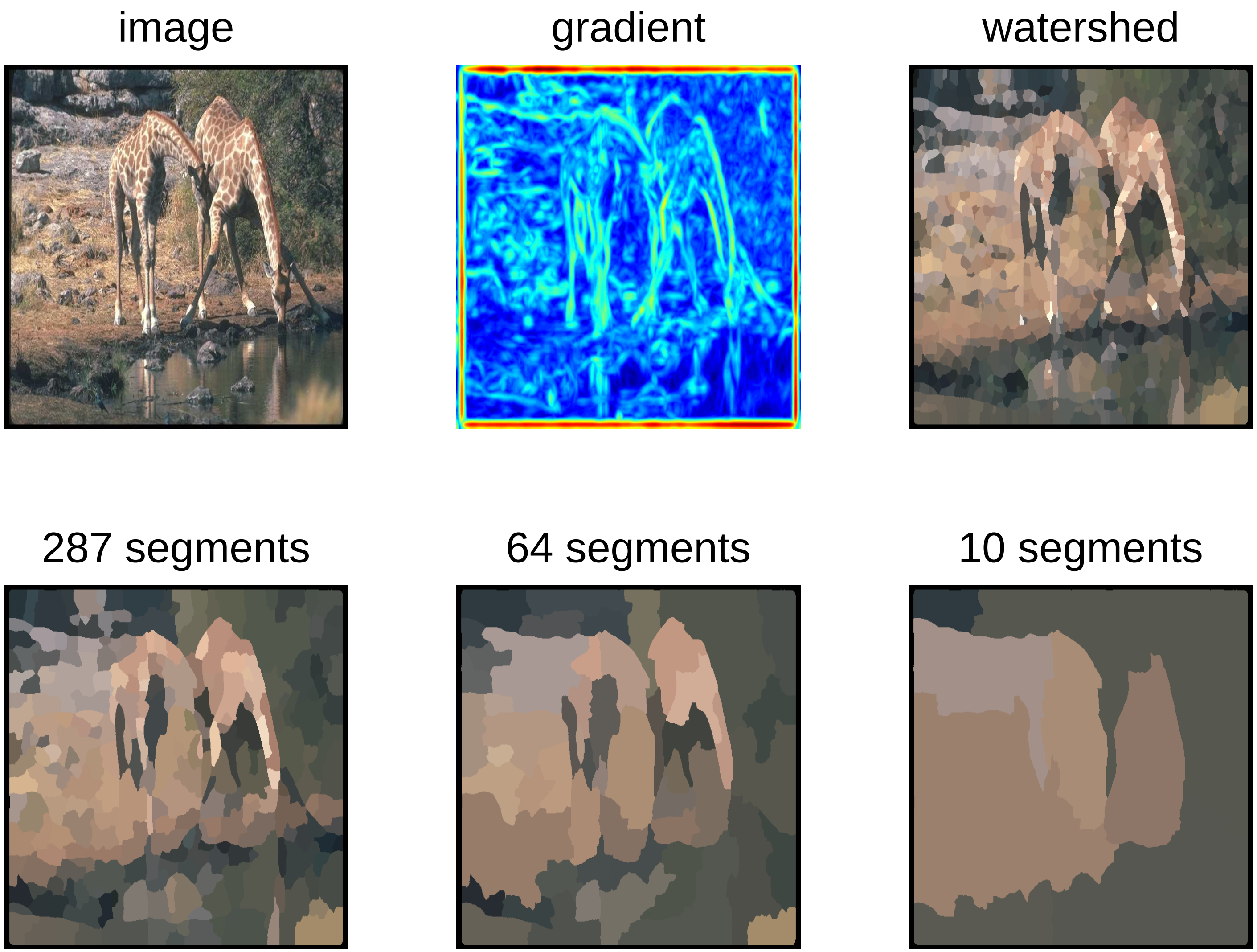


image
gradient
watershed
287 segments
64 segments
10 segments

Image as a segmentation graph

- Segmentation method
  - LAB Gradient with oriented edge filters (Malik et al, 2001)
  - Watershed transform with post-processing (Meyer, 2001)

- Labelled undirected Graph
  - Vertices: connected segmented regions
  - Edges: between spatially neighboring regions
  - Labels: region pixels

$\Rightarrow$

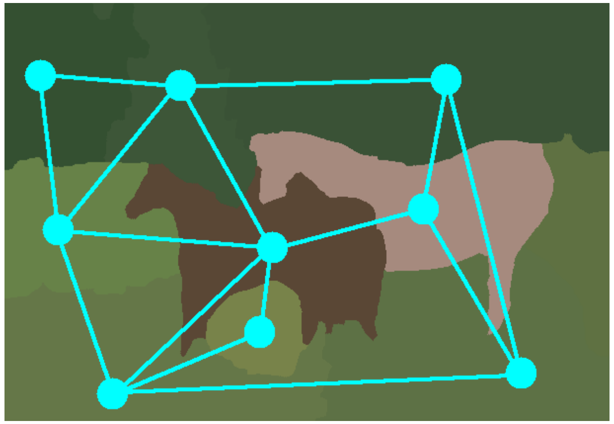


Image as a segmentation graph

- Segmentation method
  - LAB Gradient with oriented edge filters (Malik et al, 2001)
  - Watershed transform with post-processing (Meyer, 2001)

- Labelled undirected Graph
  - Vertices: connected segmented regions
  - Edges: between spatially neighboring regions
  - Labels: region pixels

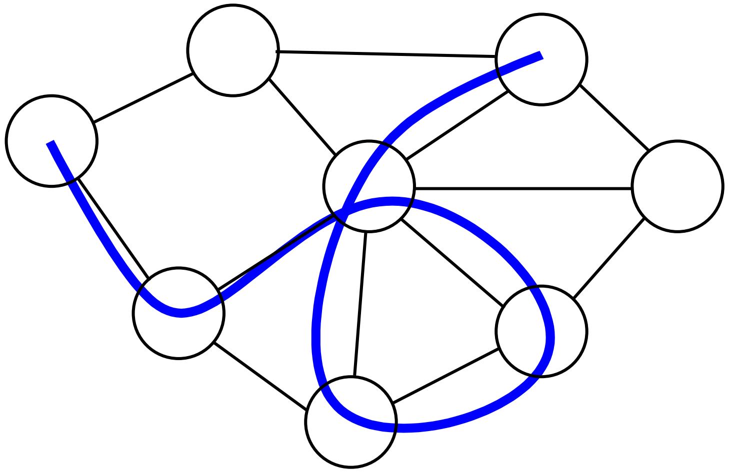
- Difficulties
  - Extremely high-dimensional labels
  - Planar undirected graph
  - Inexact matching

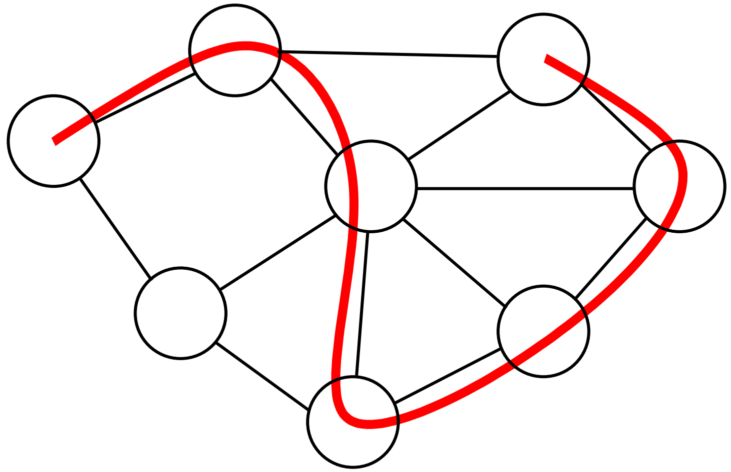
Kernels between structured objects
Strings, graphs, etc...

- Numerous applications (text, bio-informatics)
- From probabilistic models on objects (e.g., Saunders et al, 2003)
- Enumeration of subparts (Haussler, 1998, Watkins, 1998)
  - Efficient for strings
  - Possibility of gaps, partial matches, very efficient algorithms (Leslie et al, 2002, Lodhi et al, 2002, etc...)
- Most approaches fails for general graphs (even for undirected trees!)
  - NP-Hardness results (Gärtner et al, 2003)
  - Need alternative set of subparts

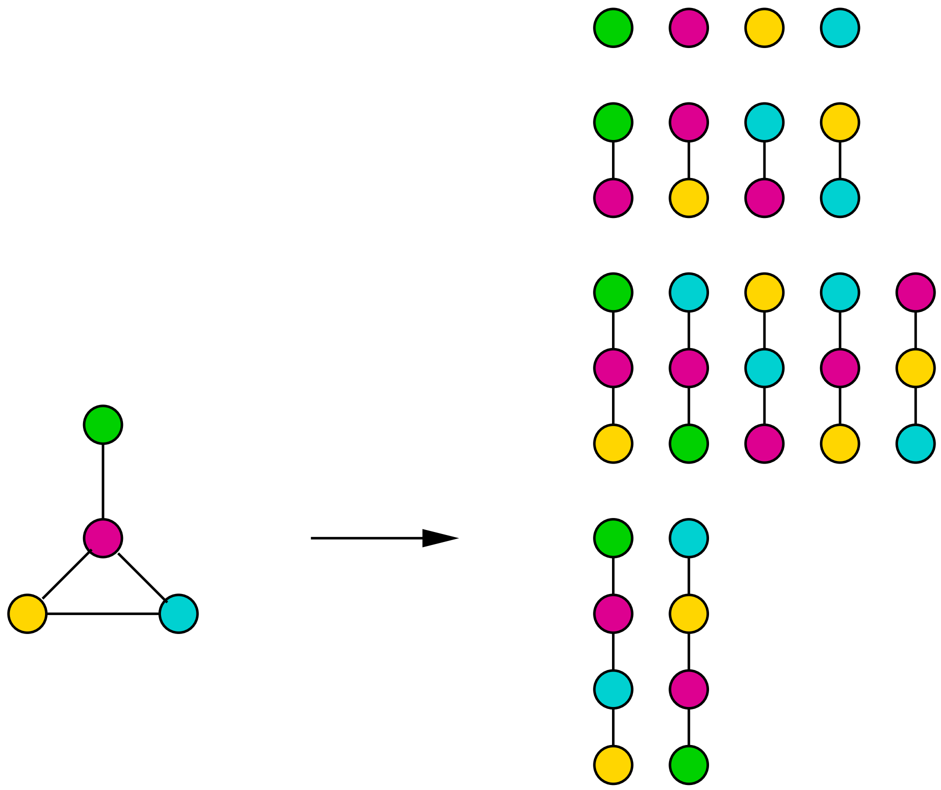
Paths and walks

- Given a graph G,
  - A path is a sequence of distinct neighboring vertices
  - A walk is a sequence of neighboring vertices
- Apparently similar notions

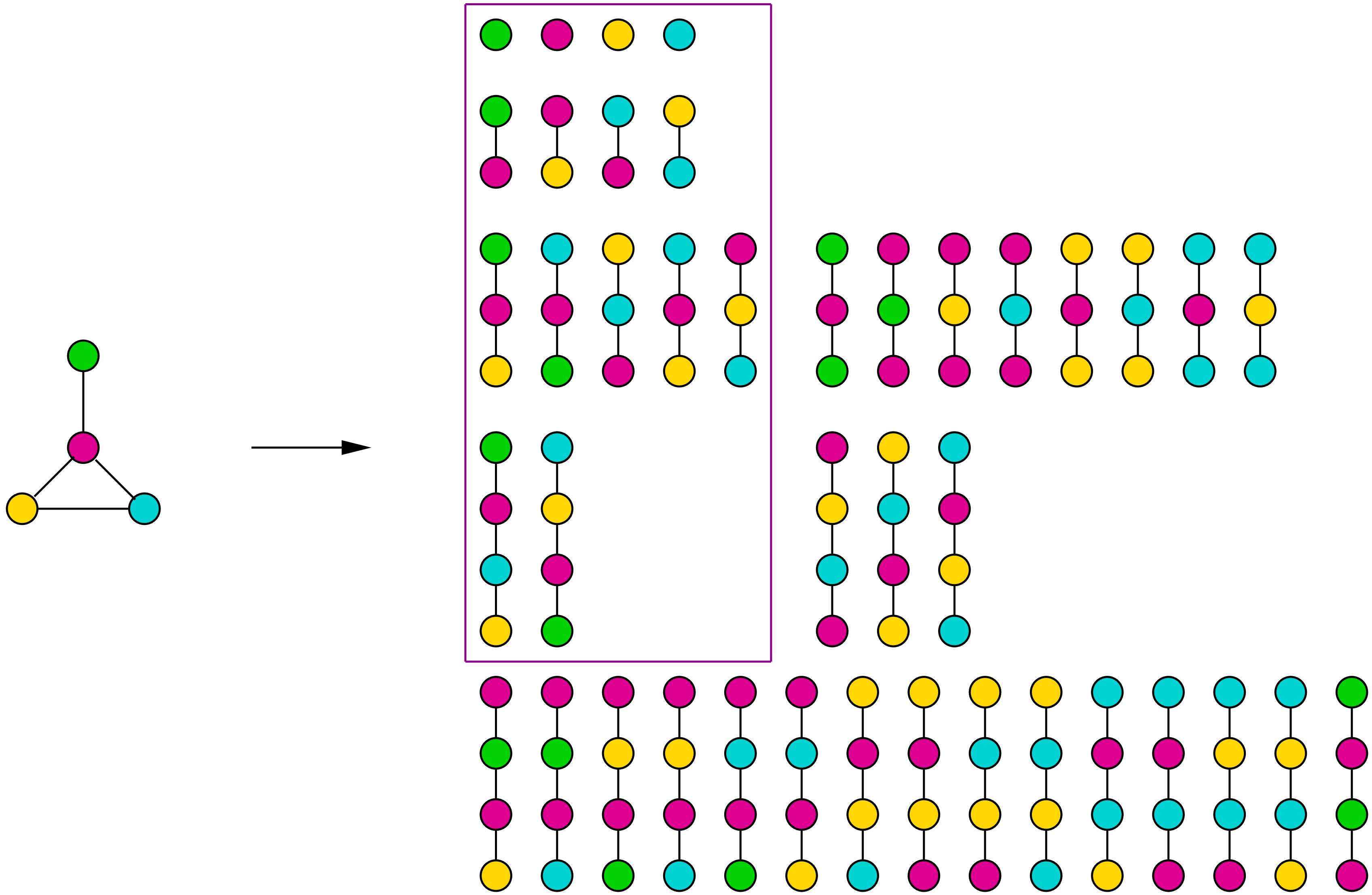




Paths



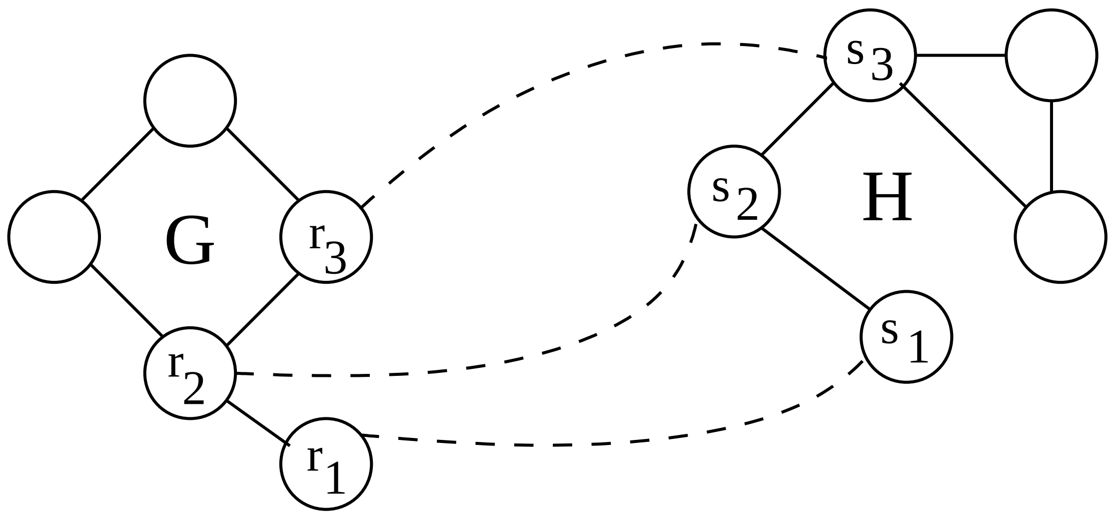
Walks



Walk kernel (Kashima, 2004, Borgwardt, 2005)

- $\mathcal{W}_{\mathbf{G}}^{p}$ (resp. $\mathcal{W}_{\mathbf{H}}^{p}$) denotes the set of walks of length p in $\mathbf{G}$ (resp. $\mathbf{H}$)
- Given *basis kernel* on labels $k(\ell, \ell')$
- p-th order walk kernel:

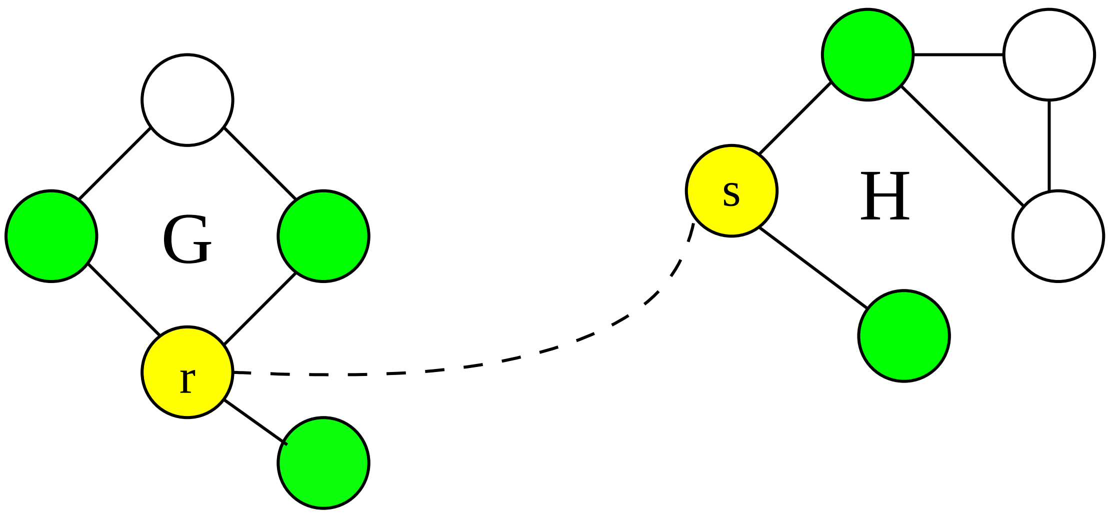
$$k_{\mathcal{W}}^{p}(\mathbf{G}, \mathbf{H}) = \sum_{\substack{(r_1, \ldots, r_p) \in \mathcal{W}_{\mathbf{G}}^{p} \\ (s_1, \ldots, s_p) \in \mathcal{W}_{\mathbf{H}}^{p}}} \prod_{i=1}^{p} k(\ell_{\mathbf{G}}(r_i), \ell_{\mathbf{H}}(s_i)).$$



Dynamic programming for the walk kernel

- Dynamic programming in $O(pd_{\mathbf{G}}d_{\mathbf{H}}n_{\mathbf{G}}n_{\mathbf{H}})$
- $k^p_{\mathcal{W}}(\mathbf{G}, \mathbf{H}, r, s) =$ sum restricted to walks starting at r and s
- recursion between $p-1$-th walk and p-th walk kernel

$$k^p_{\mathcal{W}}(\mathbf{G}, \mathbf{H}, r, s) = k(\ell_{\mathbf{G}}(r), \ell_{\mathbf{H}}(s)) \sum_{\substack{r' \in \mathcal{N}_{\mathbf{G}}(r) \\ s' \in \mathcal{N}_{\mathbf{H}}(s)}} k^{p-1}_{\mathcal{W}}(\mathbf{G}, \mathbf{H}, r', s').$$



Dynamic programming for the walk kernel

- Dynamic programming in $O(pd_{\mathbf{G}}d_{\mathbf{H}}n_{\mathbf{G}}n_{\mathbf{H}})$
- $k_{\mathcal{W}}^{p}(\mathbf{G},\mathbf{H},r,s)$ = sum restricted to walks starting at r and s
- recursion between $p-1$-th walk and p-th walk kernel

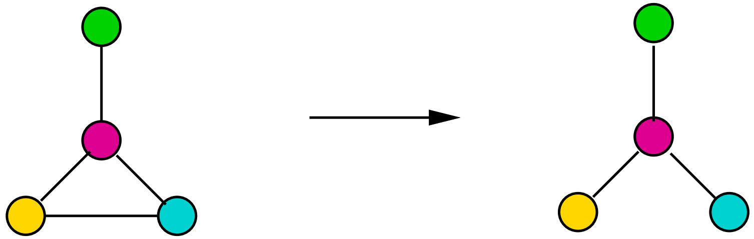
$$k_{\mathcal{W}}^{p}(\mathbf{G},\mathbf{H},r,s) = k(\ell_{\mathbf{G}}(r),\ell_{\mathbf{H}}(s)) \sum_{\substack{r' \in \mathcal{N}_{\mathbf{G}}(r) \\ s' \in \mathcal{N}_{\mathbf{H}}(s)}} k_{\mathcal{W}}^{p-1}(\mathbf{G},\mathbf{H},r',s')$$

- Kernel obtained as $k_{\mathcal{T}}^{p,\alpha}(\mathbf{G},\mathbf{H}) = \sum_{r\in\mathcal{V}_{\mathbf{G}},s\in\mathcal{V}_{\mathbf{H}}} k_{\mathcal{T}}^{p,\alpha}(\mathbf{G},\mathbf{H},r,s)$
- NB: more flexible than matrix inversion approaches

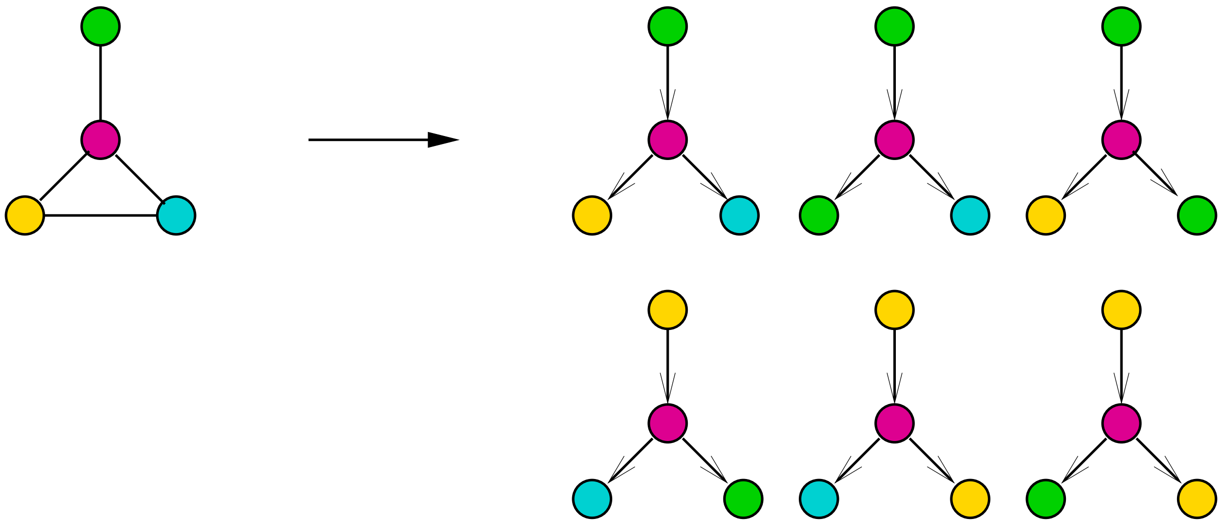
Subtrees and tree patterns

- subtree $=$ subgraph with no cycle
- tree-walks (or tree patterns)
  - natural extensions to subtrees to the "walk world"
  - α-ary tree-walk (a.k.a tree pattern) of $\mathbf{G}$: rooted directed α-ary tree whose vertices are vertices of $\mathbf{G}$, such that if they are neighbors in the tree pattern, they must be neighbors in $\mathbf{G}$ as well

Subtrees



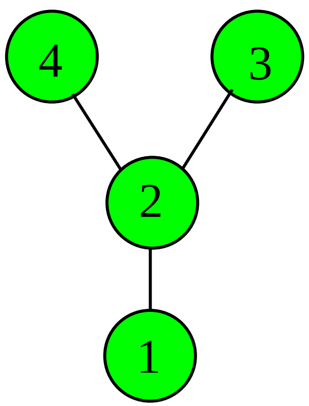
Tree patterns

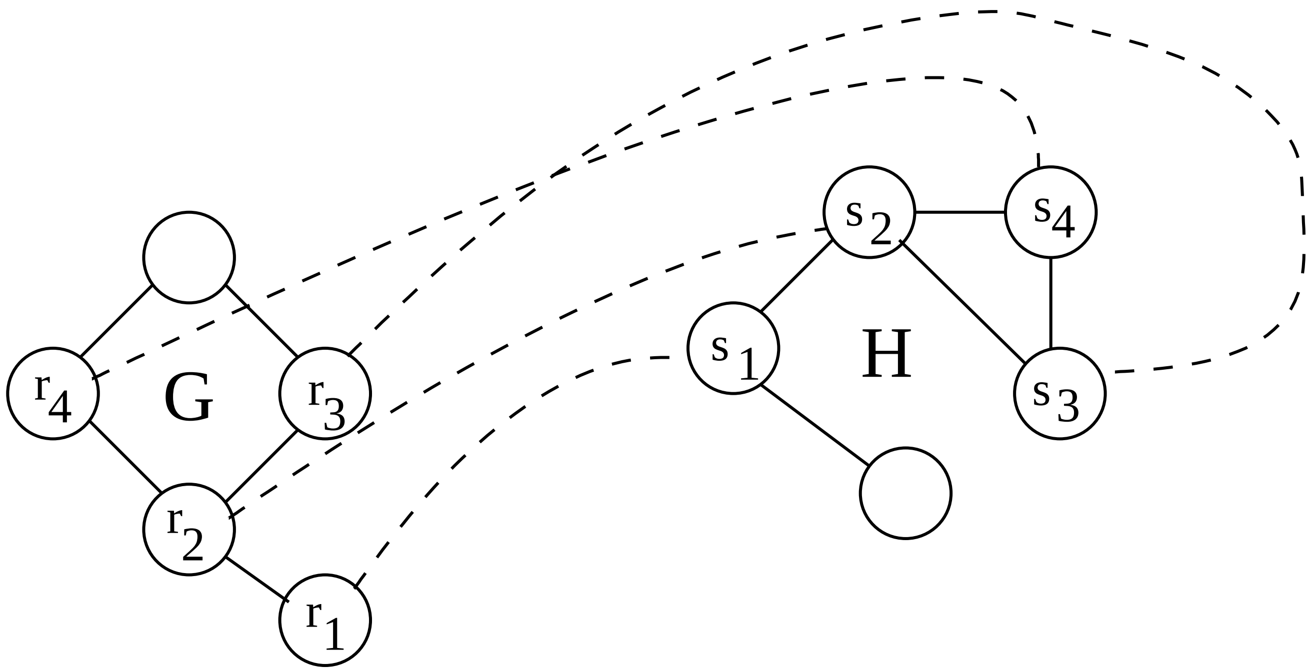


Treewalk kernel

- $\mathcal{T}_{\mathbf{G}}^{p,\alpha}$ (resp. $\mathcal{T}_{\mathbf{H}}^{p,\alpha}$) denotes the set of α-ary tree patterns of $\mathbf{G}$ (resp. $\mathbf{H}$) of depth p

- $k_{\mathcal{T}}^{p,\alpha}(\mathbf{G},\mathbf{H})$ is defined as the sum over all tree patterns in $\mathcal{T}_{p,\alpha}(\mathbf{G})$ and all tree patterns in $\mathcal{T}_{p,\alpha}(\mathbf{H})$ (that share the same tree structure)





Dynamic programming

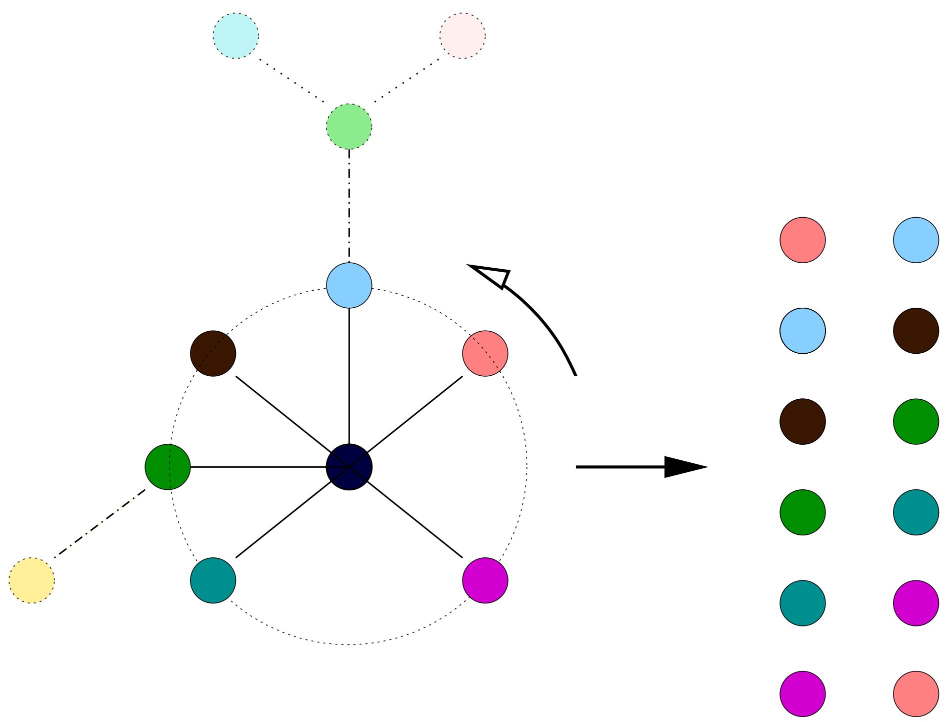
- Dynamic programming in $O(p\alpha^2 d_{\mathbf{G}} d_{\mathbf{H}} n_{\mathbf{G}} n_{\mathbf{H}})$
- NB: need planarity to avoid exponential complexity

$$k_{\mathcal{T}}^{p,\alpha}(\mathbf{G}, \mathbf{H}, r, s) \;=\; k(\ell_{\mathbf{G}}(r), \ell_{\mathbf{H}}(s)) \times \sum_{\substack{I \in \mathcal{I}_{\mathbf{G}}^{\alpha}(r) \\ J \in \mathcal{I}_{\mathbf{H}}^{\alpha}(s)}} \prod_{\substack{r' \in I \\ s' \in J}} k_{\mathcal{T}}^{p-1,\alpha}(\mathbf{G}, \mathbf{H}, r', s').$$

$$k_{\mathcal{T}}^{p,\alpha}(\mathbf{G}, \mathbf{H}) = \sum_{\substack{r \in \mathcal{V}_{\mathbf{G}} \\ s \in \mathcal{V}_{\mathbf{H}}}} k_{\mathcal{T}}^{p,\alpha}(\mathbf{G}, \mathbf{H}, r, s).$$

Planar graphs and neighborhoods

- Natural cyclic ordering of neighbors for planar graphs
- Example: intervals of length 2



Engineering segmentation kernels

- kernels between segments:
  - Chi-square metric: $d_\chi^2(P,Q) = \sum_{j=1}^{N} \frac{(p_i - q_i)^2}{p_i + q_i}$
  - P_ℓ = the histogram of colors of region labelled by ℓ

$$k(\ell,\ell') = k_\chi(P_\ell, P_{\ell'}) = e^{-\mu d_\chi^2(P_\ell, P_{\ell'})}$$

  - Segments weighting scheme $k(\ell,\ell') = \lambda A_\ell^\gamma A_{\ell'}^\gamma e^{-\mu d_\chi^2(P_\ell, P_{\ell'})}$ where A_ℓ is the area of the corresponding region

- Many (?) parameters:

Kernel	free param.	fixed param.
Histogram		μ
Walk	p	$\mu, \lambda, \alpha = 1$
Tree-walk	$p, \alpha > 1$	μ, λ
Weighted tree-walk	$p, \alpha > 1, \gamma$	μ, λ

Multiple kernel learning

- Given set of basis kernels K_j, learn a linear combination

$$K(\eta) = \sum_j \eta_j K_j$$

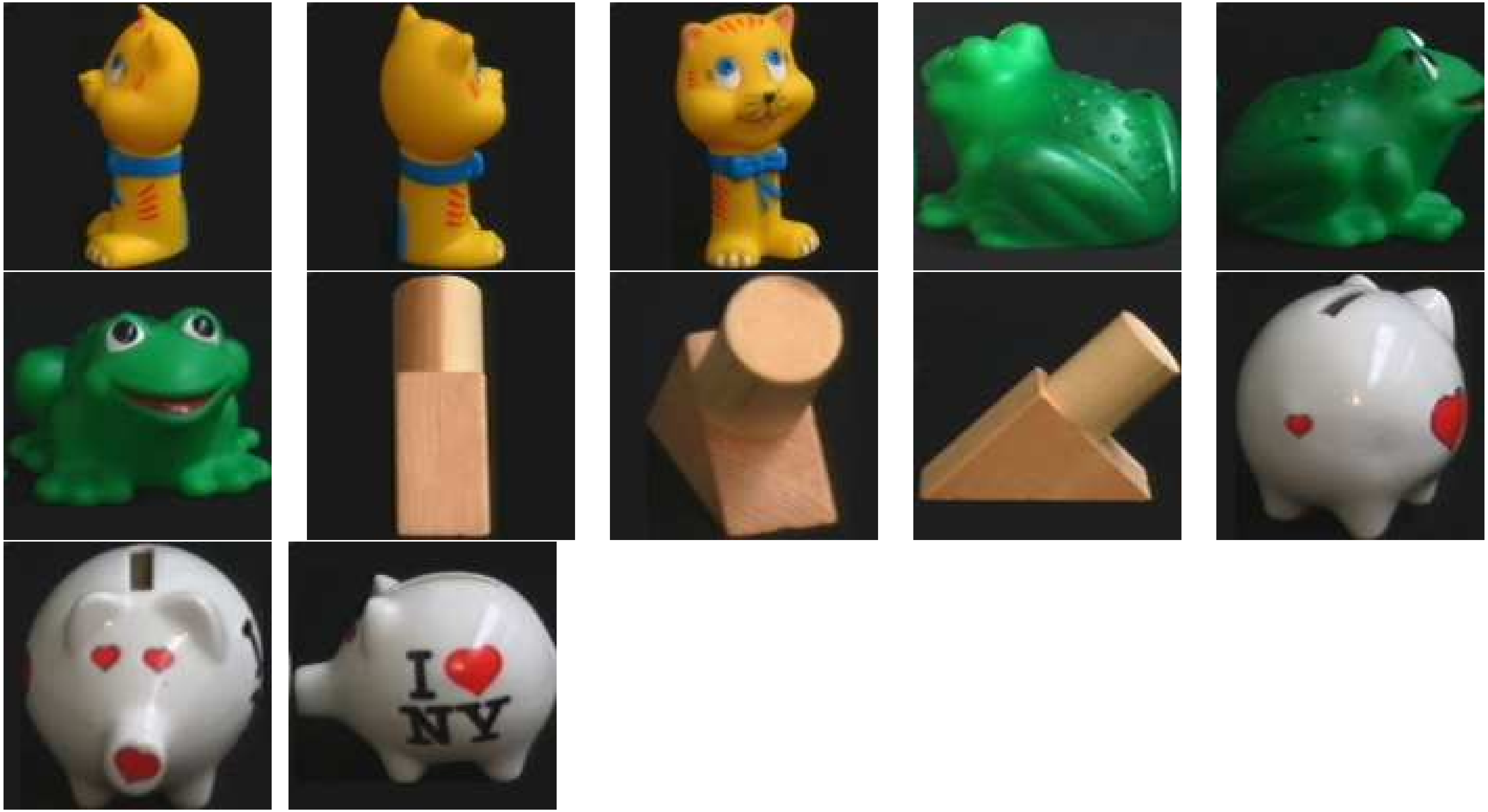
- Convex optimization problem which jointly learns η and the classifier obtained from $K(\eta)$
(Lanckriet et al, 2004, Bach et al, 2004, 2005)

- Kernel selection

- Fusion of heterogeneous kernels from different data sources

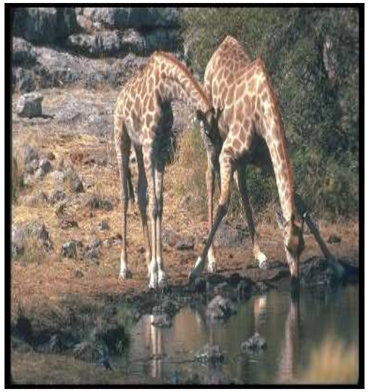
Classification experiments

- `Coil100`: database of 7200 images of 100 *objects in a uniform background*, with 72 images per object.



Classification experiments

- `Corel14` is a database of 1400 *natural images* of 14 different classes






Comparison of kernels

- kernels :
  - histogram kernel (**H**)
  - walk-based kernel (**W**)
  - tree-walk kernel (**TW**)
  - weighted-vertex tree-walk kernel (**wTW**)
  - combination of the above by multiple kernel learning (**M**)

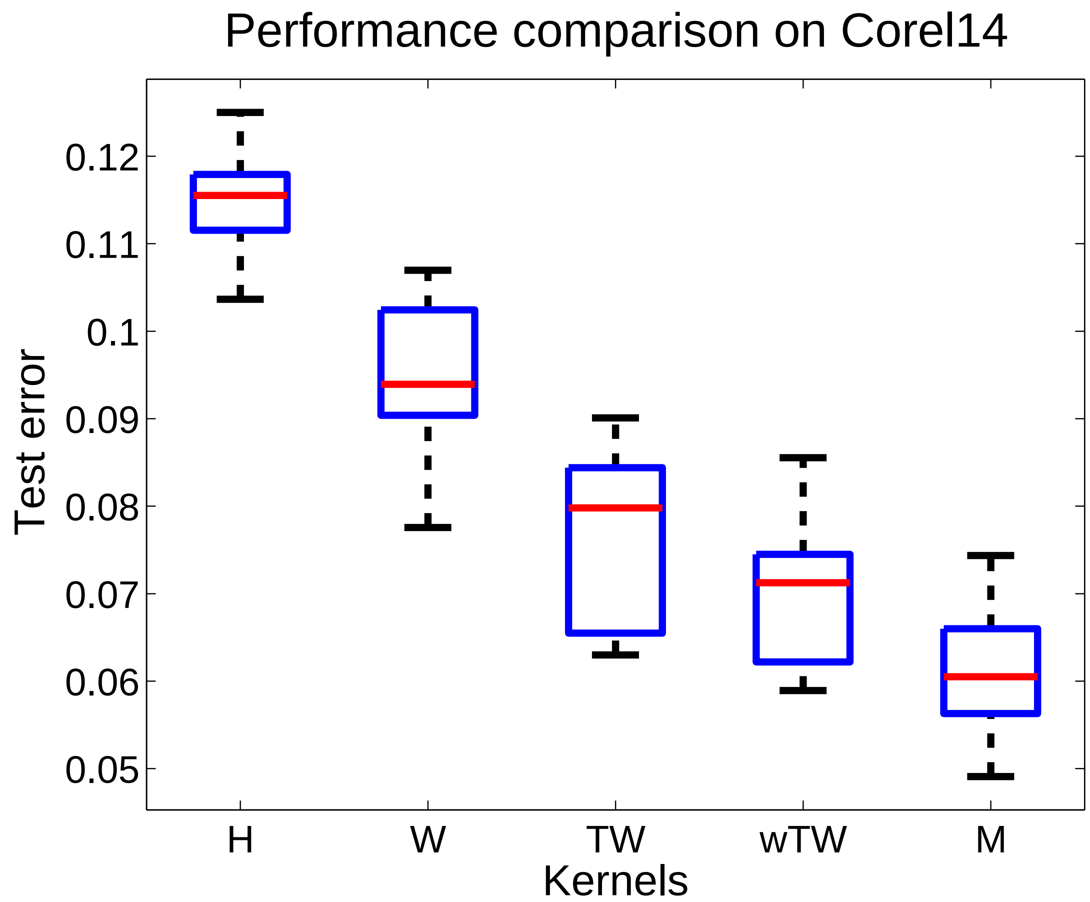
- Hyperparameters selected by cross-validation

- Error rates on ten replications:

	H	W	TW	wTW	M
`Coil100`	1.2%	0.8%	0.0%	0.0%	0.0%
`Corel14`	10.36%	8.52%	7.24%	6.12%	5.38%

Performance on Corel14 dataset

- histogram kernel (**H**)
- walk-based kernel (**W**)
- tree-walk kernel (**TW**)
- weighted-vertex tree-walk kernel (**wTW**)
- combination by MKL (**M**)



Multiple kernel learning

- 100 kernels corresponding to 100 settings of hyperparameters

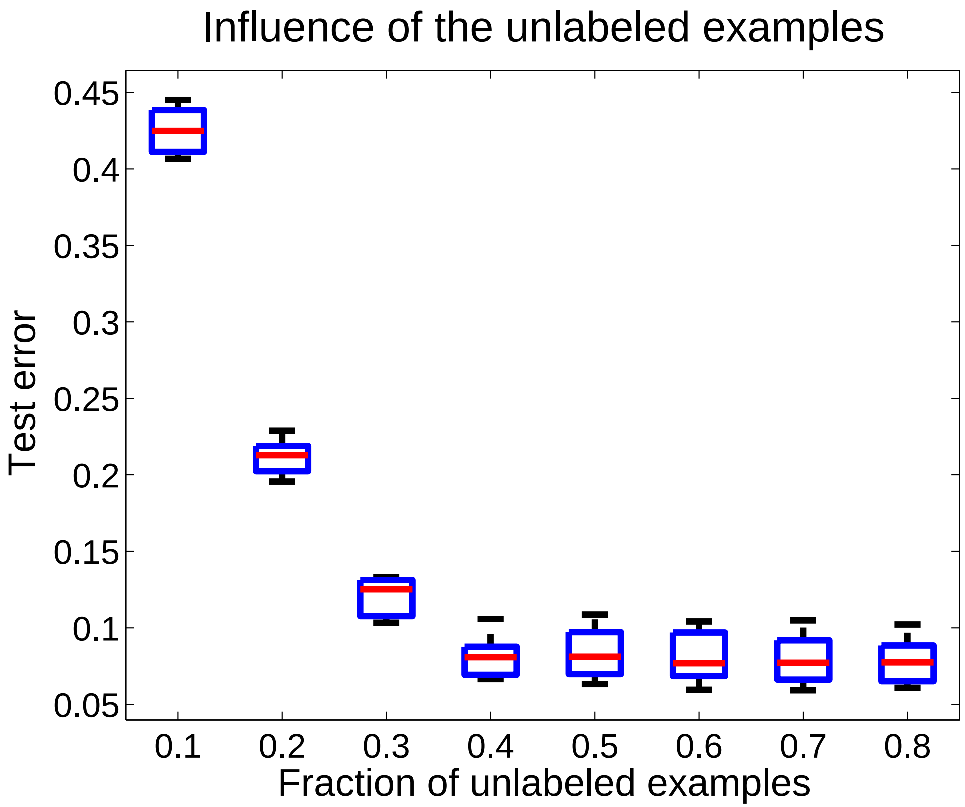
Kernel	free param.	fixed param.
Histogram		μ
Walk	p	$\mu, \lambda, \alpha = 1$
Tree-walk	$p, \alpha > 1$	μ, λ
Weighted tree-walk	$p, \alpha > 1, \gamma$	μ, λ

- Selected kernels

p, α, γ	$10, 3, 0.6$	$7, 1, 0.6$	$10, 3, 0.3$	$5, 3, 0.0$	$8, 1, 0.0$
η	0.12	0.17	0.10	0.07	0.04

Semi-supervised learning

- Kernels give task flexibility
- Example: semi-supervised algorithm of Chapelle and Zien (2004)
- 10% labelled examples, 10% test examples, 10% to 80% unlabelled examples



Conclusion

- Learning on images with kernels on segmentation graphs
  - Based on a natural and still noisy representation of images
  - Prior information allows better generalization performances
  - Modularity
- Current work and natural extensions:
  - Non-tottering trick (Mahé et al, 2005)
  - Allows gaps (Saunders et al, 2001)
  - Shock graphs (e.g., Suard et al., 2005)
  - SIFT features
- Application to image retrieval