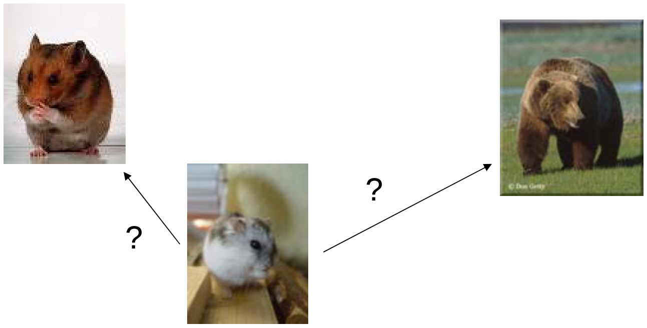
SVM-KNN : Discriminative Nearest Neighbor Classification for Visual Category Recognition

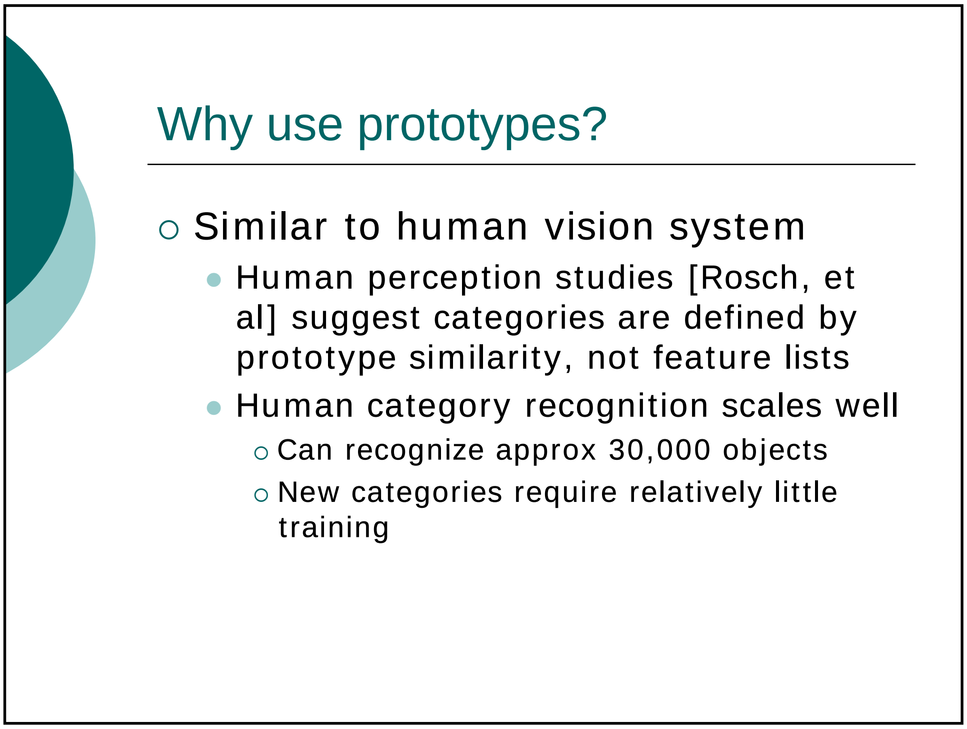
Hao Zhang, Alexander Berg, Michael Maire
Jitendra Malik
EECS, UC Berkeley

Presented by Adam Bickett

Objective

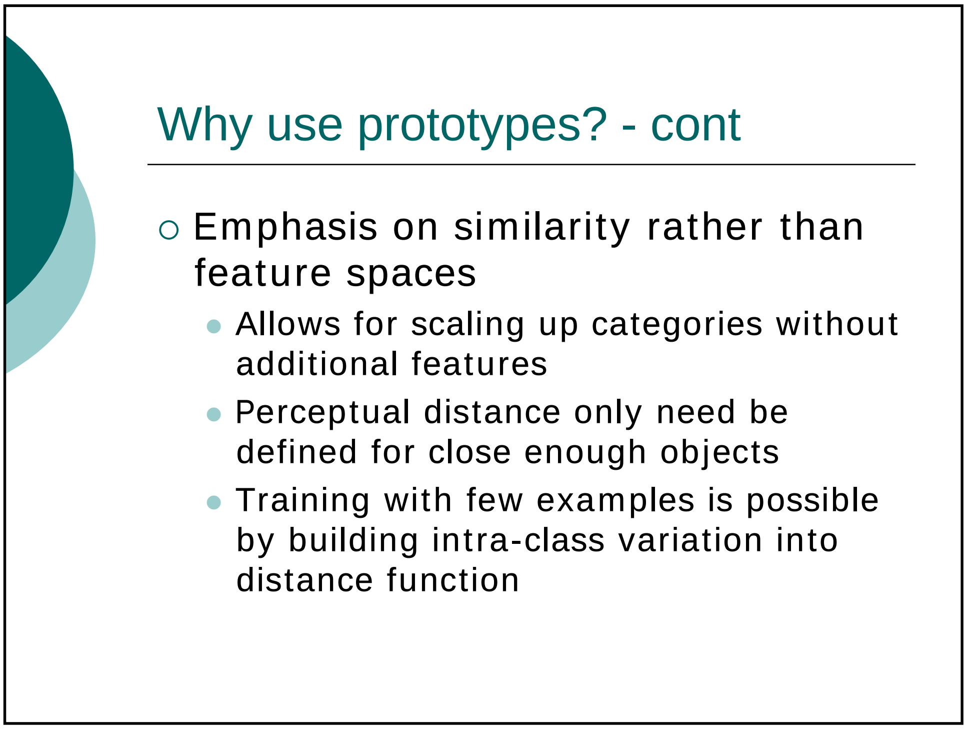
- Visual Category Recognition
  - Use prototype category examples, and measure similarity, or "perceptual distance" to these examples





Why use prototypes?

- Similar to human vision system
  - Human perception studies [Rosch, et al] suggest categories are defined by prototype similarity, not feature lists
  - Human category recognition scales well
    - Can recognize approx 30,000 objects
    - New categories require relatively little training



Why use prototypes? - cont

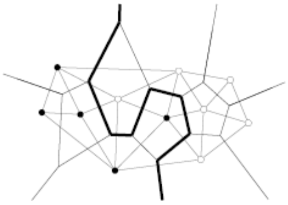
- Emphasis on similarity rather than feature spaces
  - Allows for scaling up categories without additional features
  - Perceptual distance only need be defined for close enough objects
  - Training with few examples is possible by building intra-class variation into distance function

Nearest Neighbor - Strengths

- Nearest neighbor often outperforms more sophisticated techniques
  - Many other techniques require explicit high-dimensional feature space, which may be intractable for some distance functions
  - NN naturally scales to multiple classes
  - Error rate of KNN asymptotically approaches the Bayes optimal

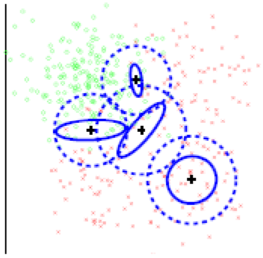
Nearest Neighbor - Weakness

- Dense sampling is required for optimal behavior
  - In practical applications, decision boundary can be jagged due to limited available sampling from the category distribution



Nearest Neighbor – Improvements

- Attempts have been made to remedy the limited sample issue by warping the distance metric:
  - DANN uses LDA to weight distance function based on neighborhood

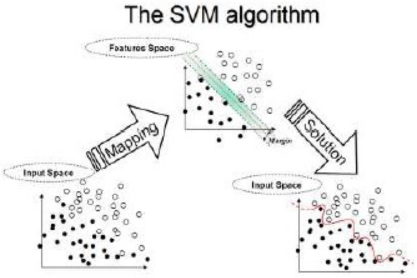


Nearest Neighbor - Improvements

- LFM-SVM trains a Support Vector Machine (SVM) on the whole data set to obtain a weighting
- HKNN forms linear subspaces from neighbors and measures distance to these subspaces

Using SVM locally

- What if we don’t want to distort our distance metric?
  - Use SVM on a small collection of nearest neighbors



Support Vector Machines

- SVM is a powerful method using hyperplanes to separate data with a maximum margin

$$\min \sum_{i=1}^{n} \alpha_i - \sum_{i,j} \alpha_i \alpha_j c_i c_j x_i^T x_j$$

  - Because the minimization involved in solving SVMs only involves inner products between data points, we can readily use the “Kernel trick” to transform it to work on our defined distance metric

$$K(x,y) = \langle x,y \rangle = \frac{1}{2}(\langle x,x \rangle + \langle y,y \rangle - \langle x-y, x-y \rangle) =$$

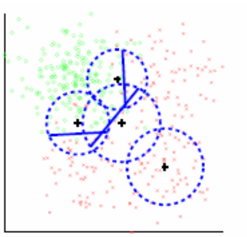
$$\frac{1}{2}(d(x,0) + d(y,0) - d(x,y))$$

Benefits of this approach

- SVMs operate directly on the kernel matrix K (basically the pairwise distance matrix)
  - Don’t have to deal with complex feature space directly
  - Allows us to use complex distance functions (not limited to L_2!)
  - Training an SVM is slow for large datasets, but performs very well with small neighbourhood, few classes

SVM – KNN

- Main Idea : Prune using NN, refine using SVM
  - Motivated from psychophysics results: humans can coarsely categorize fast, then take time to refine
  - Combines the efficiency of NN with a simple distance function and the better decision boundaries of SVMs

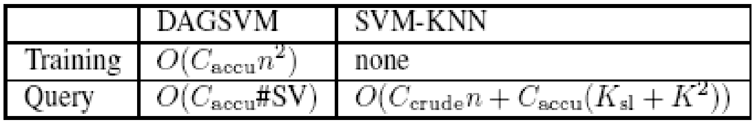


SVM-KNN – The Algorithm

1) Find a collection K_{sl} of neighbors using crude distance function (Like L_2) from query
2) Compute the “accurate” distance function on the K_{sl} samples, pick the K nearest neighbors
3) Compute (or read from cache) the pairwise “accurate” distance of K + {query}
4) Convert pairwise distance matrix into Kernel matrix using the kernel trick
5) Train a multiclass SVM (DAGSVM) on the kernel matrix, label the query with this classifier.

SMV-KNN - Analysis

- SVM-KNN can be viewed as a continuum between SVM and KNN
  - Small K behaves like KNN
  - K = n reduces to an SVM

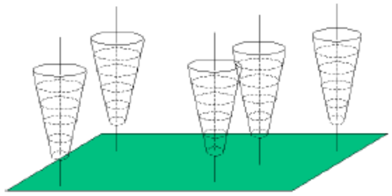


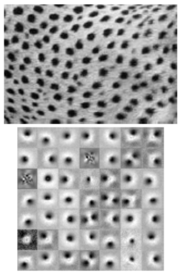
	DAGSVM	SVM-KNN
Training	$O(C_{\text{accu}} n^2)$	none
Query	$O(C_{\text{accu}} \#\text{SV})$	$O(C_{\text{crude}} n + C_{\text{accu}}(K_{\text{sl}} + K^2))$

- DAGSVM becomes intractable for large n or complex distance function
- SVM-KNN remains feasible given an efficient crude distance calculator, and reasonable local SVM calculation

Distance Functions - Texture

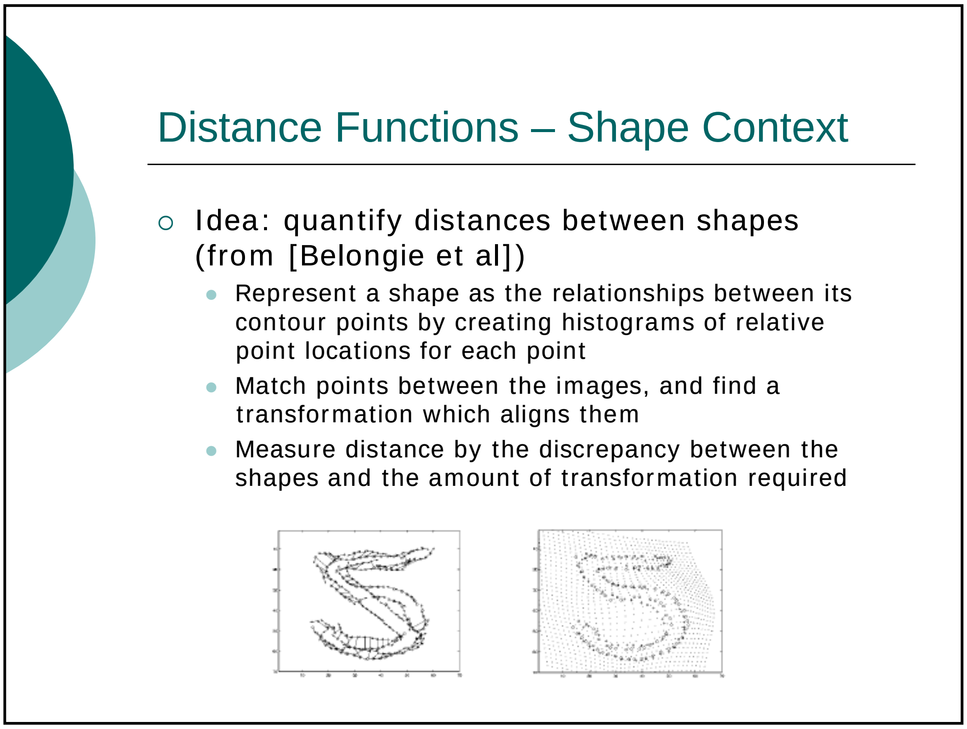
- Find the distance between textures of two images
  - Textons [Leung, Malik] – refers to image response to a bank of filters
  - Take the X^2 distance between histograms of textons





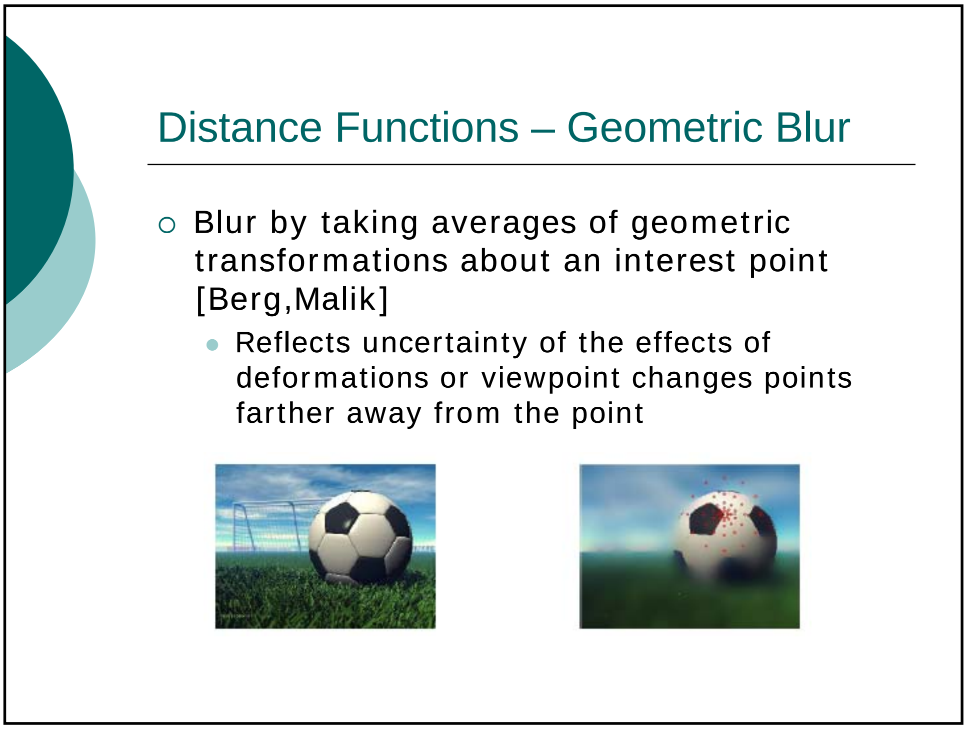
Distance Functions – Tangent Distance

- Idea from [Simard et al]: find distance between manifolds representing all transformations of the input images
  - Approximate these manifolds by applying small transformations to images, which forms the tangent to the manifold at the image point.
  - Find the distance between the tangents.



Distance Functions – Shape Context

- Idea: quantify distances between shapes (from [Belongie et al])
  - Represent a shape as the relationships between its contour points by creating histograms of relative point locations for each point
  - Match points between the images, and find a transformation which aligns them
  - Measure distance by the discrepancy between the shapes and the amount of transformation required

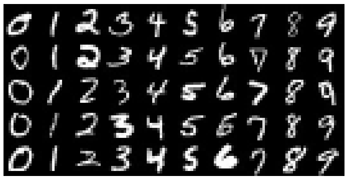


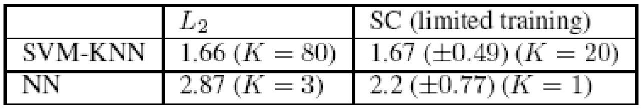
Distance Functions – Geometric Blur

- Blur by taking averages of geometric transformations about an interest point [Berg,Malik]
  - Reflects uncertainty of the effects of deformations or viewpoint changes points farther away from the point

Results – MNIST

- Handwritten digits
- 60k training, 10k test examples.
- SVM-KNN improves performance over NN



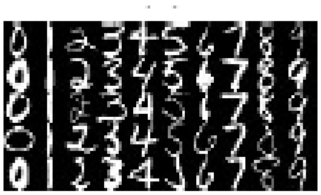


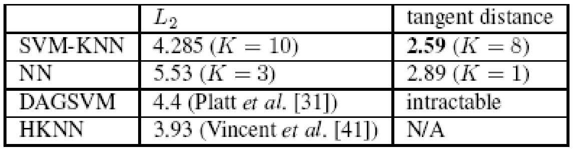
	L_2	SC (limited training)
SVM-KNN	1.66 ($K = 80$)	1.67 (± 0.49) ($K = 20$)
NN	2.87 ($K = 3$)	2.2 (± 0.77) ($K = 1$)

Table 2. error rate on MNIST (in percent): the parameter K for each algorithm is selected according to best performance(in range of [1,10] for NN and [5, 10, .., 100] for SVM-KNN). In SVM-KNN, the parameter $K_{sl} \approx 10K$, larger K_{sl} doesn't improve the empirical results.

Results – USPS

- Handwritten digits, 7291 training, 2007 test examples.
- Human error at 2.5%!
- SVM-KNN is only slightly slower than NN
- DAGSVM and HKNN can't be extended beyond L_2 distance.





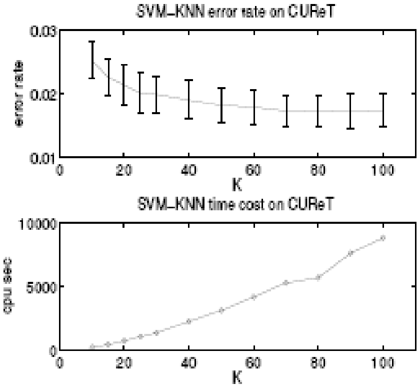
	L_2	tangent distance
SVM-KNN	4.285 ($K = 10$)	**2.59** ($K = 8$)
NN	5.53 ($K = 3$)	2.89 ($K = 1$)
DAGSVM	4.4 (Platt *et al.* [31])	intractable
HKNN	3.93 (Vincent *et al.* [41])	N/A

Results – CUReT

- Database of 61 real-world textures
- 46 images for training, 46 for testing for each texture
- SVM-KNN has a slight edge over DAGSVM, which must train 1803 pairwise SVMs!

	χ^2
SVM-KNN	**1.73** (± 0.24) ($K = 70$)
NN	2.53 (± 0.28) ($K = 3$) [40]
DAGSVM	1.75 (± 0.25)

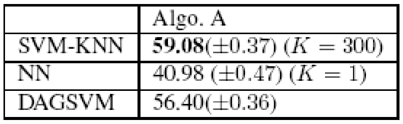
Results – Accuracy/Speed Tradeoff



By adjusting K, SVM-KNN can be tuned for speed or performance

Results – Caltech 101

- Images of 101 objects, and background
- Used texture distance and geometric blur features
- State of the art performance



	Algo. A
SVM-KNN	**59.08**($\pm$0.37) ($K = 300$)
NN	40.98 ($\pm$0.47) ($K = 1$)
DAGSVM	56.40($\pm$0.36)

Summary

- SVM-KNN gives a simple approach of achieving excellent category recognition results by refining local decision boundaries using SVM
- Much more efficient than SVM, and can be adjusted to fine tune performance and speed
- The distance function used can easily be tailored to the application
- Can be viewed as a model of the discrimination process in biological vision