CS5440 Pattern Recognition

Using Visual Words for Image Classification

u9562171 雷禹恆 (Yu-Heng Lei)
June 17, 2010

MiRA research group @ CMLab, NTU

Original Proposal

Fast Concurrent Object Localization and Recognition

Yeh, Lee, Darrell @ CVPR 2009

OTL …

Outline

- Motivation
- Construction of visual words
- Dataset
- Experiment results
- **Demo**
- Discussions

Motivation

- Visual words are largely used in image retrieval
- Motivation from text retrieval:

  An article consists of words

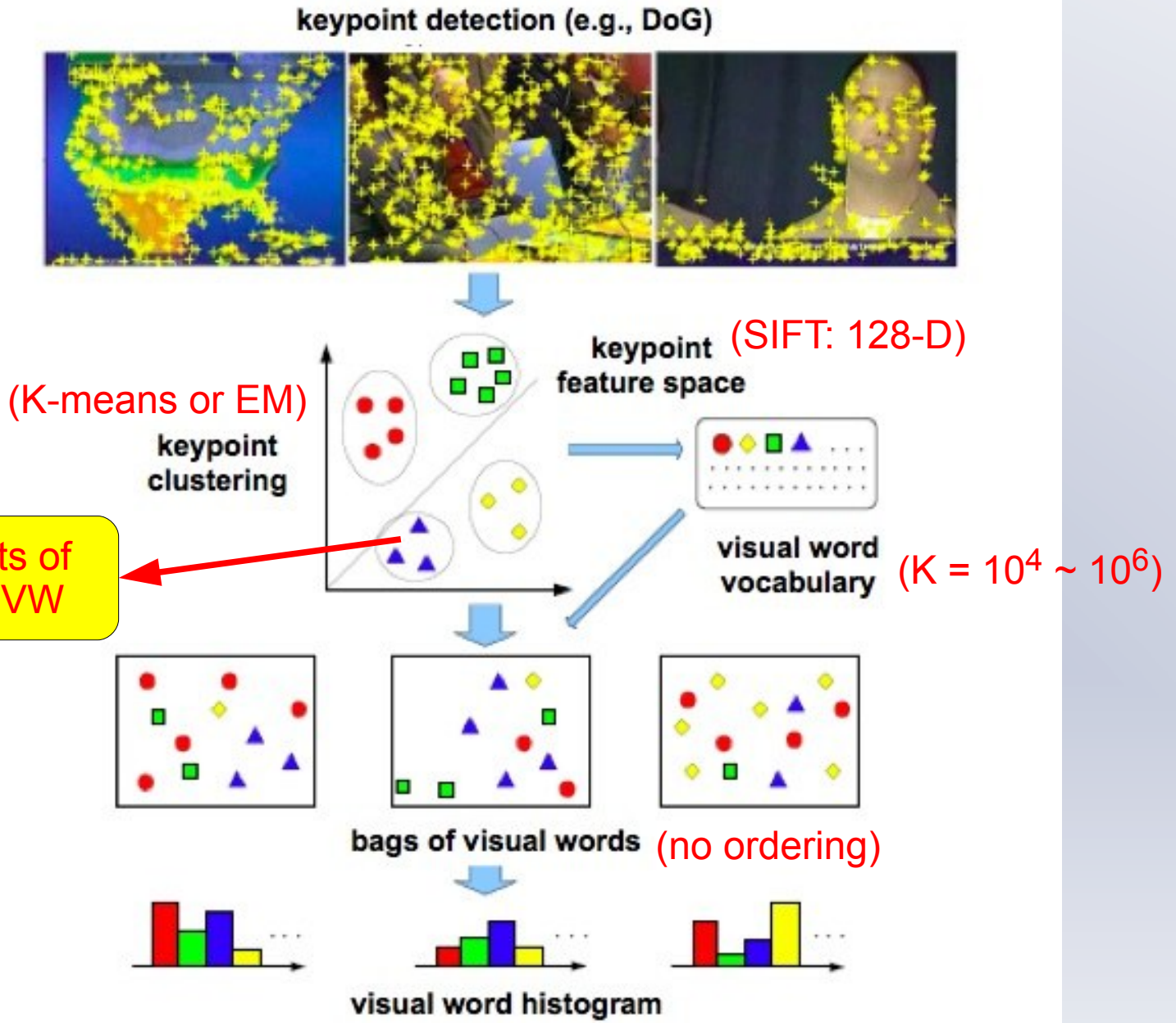
  ↔ An image consists of visual words (VWs)
- Apply text-based search techniques in image search
- Reference: "*Video Google: A Text Retrieval Approach to Object Matching in Videos*", Sivic @ ICCV 2003.

Construction of Visual Words

- 1. Detect SIFT keypoints and compute their SIFT descriptors (128-dimensional)
- 2. In keypoint space → K-means clustering
- 3. Each Image represented by a "**visual word histogram**"
- (Local) keypoint descriptors + clustering

  → (Global) visual words histogram

Construction of Visual Words



* Jun Yang, *et. al.* , MIR, 2007.

Dataset

- Subset of **Caltech 101**: 4 classes
- Number of images (train : test = 9 : 1):

Label	Class	Total	Training	Testing
1	Airplane	800	720	80
2	Face	435	391	44
3	Motorbike	797	717	80
4	None	368	331	37
	Σ	2400	2159	241

Dataset

Experiment – 1st Attempt

- K = 10,000 + MATLAB's built-in SVMs + one-against-one majority vote
- Confusion matrices:

training

$$\begin{bmatrix} 720 & 0 & 0 & 0 \\ 0 & 391 & 0 & 0 \\ 0 & 0 & 717 & 0 \\ 0 & 0 & 0 & 331 \end{bmatrix}$$

testing

$$\begin{bmatrix} 80 & 0 & 0 & 0 \\ 44 & 0 & 0 & 0 \\ 80 & 0 & 0 & 0 \\ 37 & 0 & 0 & 0 \end{bmatrix}$$

- All points are support vectors
- Extremely over-fit !!

Experiment – 2nd Attempt

- K = 10,000 + libSVM for multi-class + libSVM parameter selection
- Result:

  Accuracy = 95.4357% (230/241)

$$confusion = \begin{bmatrix} 76 & 0 & 1 & 3 \\ 0 & 43 & 0 & 1 \\ 0 & 0 & 79 & 1 \\ 4 & 1 & 0 & 32 \end{bmatrix}, \quad recall = \begin{bmatrix} 0.95 \\ 0.98 \\ 0.99 \\ 0.86 \end{bmatrix}, \quad precision = \begin{bmatrix} 0.95 \\ 0.98 \\ 0.99 \\ 0.87 \end{bmatrix}$$

- Much, much better !!

Experiment – 3rd Attempt

- K = 1,000 + libSVM for multi-class + libSVM parameter selection + PCA (total variance = 98%, → 195-dimensional)
- Result:

  Accuracy = 95.8506% (231/241)

$$confusion = \begin{bmatrix} 72 & 0 & 1 & 2 \\ 0 & 44 & 0 & 0 \\ 0 & 0 & 80 & 0 \\ 4 & 1 & 2 & 30 \end{bmatrix}, recall = \begin{bmatrix} 0.96 \\ 1.00 \\ 1.00 \\ 0.81 \end{bmatrix}, precision = \begin{bmatrix} 0.95 \\ 0.98 \\ 0.96 \\ 0.94 \end{bmatrix}$$

- Nothing better, but MUCH faster !

Experiment – 4th Attempt

- Guess from the PCA experience:
- K = 125 + libSVM for multi-class + libSVM parameter selection (no PCA)
- Result:

  Accuracy = 94.6058% (228/241)

$$confusion = \begin{bmatrix} 75 & 0 & 1 & 4 \\ 0 & 44 & 0 & 0 \\ 0 & 0 & 80 & 0 \\ 4 & 2 & 2 & 29 \end{bmatrix}, recall = \begin{bmatrix} 0.94 \\ 1.00 \\ 1.00 \\ 0.78 \end{bmatrix}, precision = \begin{bmatrix} 0.95 \\ 0.96 \\ 0.96 \\ 0.88 \end{bmatrix}$$

- Just as fast, just as accurate.

Demo

Discussions

- Choices of parameters
  1. SVM parameters (C, γ)

  → Use the grid.py provided by libSVM

  2. PCA target dimension (% of total variance)

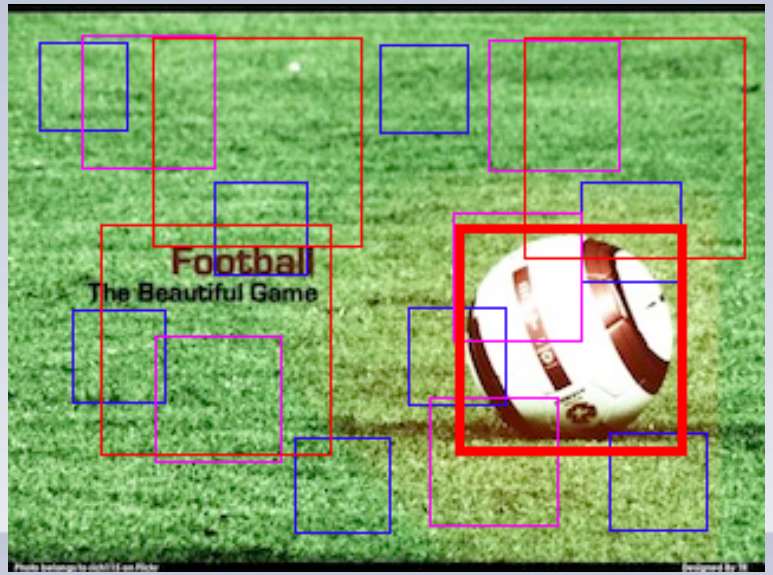
  → Choose the lowest one with good performance

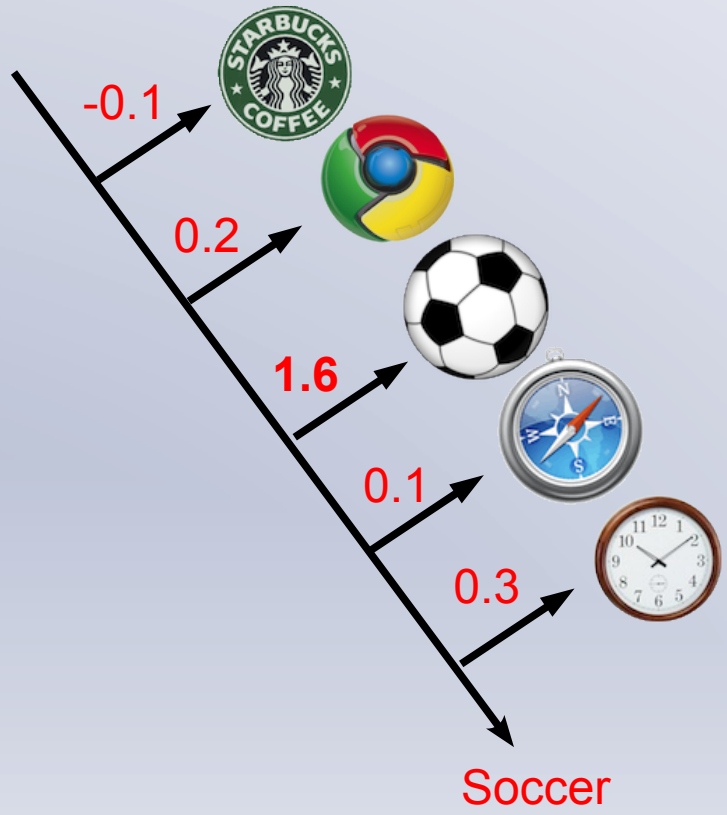
  3. Vocabulary size (K in K-means)

  → Only try the available K = 10^4, 10^3, 5^3
- Did not filter out “very common words”

Discussions

- Challenges
  1. More difficult datasets
  2. Speed for Localization and Recognition





Discussions

- Project hosting at Google Code:

  http://code.google.com/p/search

Thank You