Lecture 12: Recognition

1 Recognition

Recognition applications are: large scale image retrieval, recognition for smartphones, face recognition (four basic types of feature detectors, white areas are subtracted from black ones, deep learning), technology to convert scanned documents to text, pedestrian recognition.
Challenges are: intra class variations (how to detect ANY car), context and human experience.
There are essentially two schools of approaches:

- **Model based:** tries to fit a model (2D or 3D) using a set of corresponding features (SIFT + RANSAC).
  Example: Is this book present in the scene? Look for corresponding matches: if most of the book's keypoints are present in the scene, the book is present in the scene.
- **Appearance based:** the model is defined by a set of images representing the object (template matching for example).
  Example: The model of the object is simply an image (template). Shift the template over the image and compare (NCC, SSD). This works only if the object and the template are identical.

The main goal of object recognition is to **classify**. Say yes or no to presence of an oject or categorize it. Either with bounding box or full segmentation.

1.1 Detection via Classification

1.1.1 Main Idea

We need to

- Obtain training data.
- Define features.
- Define classifiers.

Consider all subwindows in an image and sample at multiple scales and positions. Make a decision per window: does this contain the object we are looking for or not?

1.2 Generalization: Machine Learning approach

The main concept is to apply a prediction function to a feature representation of the image to get the desired output.

$$\underbrace{y}_{\text{output}} = \underbrace{f}_{\text{prediction function}} (\underbrace{x}_{\text{input: image features}}). \tag{1.1}$$

In general, one deals then with two different procedures:
Training: Given a training set of labeled (with correct answers!) examples, estimate the prediction function by minimizing the prediction error on the set.
Testing: Apply f to a never-before-seen test example x and output the predicted value $y = f(x)$.
Examples of features could be blob features, image histograms or histograms of oriented gradients.

1.2.1 Classifiers: Nearest neighbor

Features are represented in the descriptor space. Brute force matching, compute distances. Important facts are

- **No training** is required.
- All we need is a distance function for our inputs.
- Problem: need to compute distances to all training examples.

Features could be blob features, image histograms or histograms of oriented gradients.

1.2.2 Classifiers: Linear

Find a linear function to separate the classes

$$f(x) = \mathrm{sgn}(wx + b). \tag{1.2}$$

1.2.3 Classifiers: Nonlinear

Find a nonlinear function to separate the classes.
How can one define a classifier? We need to cluster the training data, then we need a distance function to determine to which cluster the query image belongs to.

1.2.4 K-Means Clustering

This is an algorithm to partition n observations into k clusters in which each observation x belongs to the cluster S_i with centroid m_i. It minimizes the sum of squared Euclidean distances between points x and their nearest cluster centers m_i:

$$D(X, M) = \sum_{i=1}^{k} \sum_{x \in S_i} (x - m_i)^2. \tag{1.3}$$

The algorithm reads:

1. Randomly initialize k cluster centers
2. Iterate untl convergence:
   - Assign each data point x_i to the nearest center m_i.
   - Recompute each cluster center as the mean of all points assigned to it.

1.3 Bag of Words

This method is used e.g. for large-scale image retrieval.

1.3.1 Visual Place Recognition

We want to find the most similar images of a query image in a database of N images. The complexity is $\frac{N^2M^2}{2}$ feature comparisons (assuming each image has M features. Each image must be compared with all other images. Only with $M = N = 100$, you get 50 million images. The solution is to use an **inverted file index**, which reduces the complexity to $N \times M$.

1.3.2 Inverted File Text

For text documents, an index is important. We want to find every image in which a feature occurs. How many SIFT or BRISK features exist ?

- SIFT: infinite.
- BRISK-128: $2^{128} = 2.4 \cdot 10^{38}$.

For this reason, we need to create visual words, then put them into a vocabulary. Basically, we collect images and we extract features. A visual word is the centroid of a cluster. We then cluster the descriptors with the different words.
An inverted file index lists all visual words in the vocabulary (extracted at training time). Each word points to a list of images from the all image Data Base, in which which that word appears. The DB grows as the robot navigates and collects new images. **Voting Array**: has as many cells as the images in the DB. Each word in the query image votes for an image.

1.3.3 Robust object/scene recognition

Visual Vocabulary discards the spatial relationships between features: two images with the same features shuffled around will return a 100% match when using only appearance information. This can be overcome with **geometric verification**: test the h most similar images to the query image for geometric consistency (5,8 point RANSAC) and retain the image with the smallest reprojection error and largest number of inliers. More words is better!

1.4 Understanding Check

Are you able to answer the following questions?

- *What is an inverted file index?*
- *What is a visual word?*
- *Why do we need hierarchical clustering?*
- *How does K-means clustering work?*
- *Explain and illustrate image retrieval using Bag of Words.*
- *Discussion on place recognition: what are the open challenges and what solution have been proposed?*