COMP3005/Computer Vision

9

Image search and Bag of Visual Words

Summary

Content-based image retrieval (CBIR) is the name given to systems that can search for images with an *image as the query*. Research on CBIR systems started in the early 90's, but it is only more recently with ubiquitous mobile computing and applications like Google Goggles that the technology has matured. In previous lectures, we've seen how (local) descriptors can be used to find matching objects within images, but we've also seen that the matching process is rather computationally expensive. For CBIR applications we need to be able to search datasets of millions of images almost instantaneously. In the field of textual document search, techniques to efficiently index and efficiently search massive datasets of text documents are well understood. One of the biggest advances in CBIR has been to apply these textual indexing techniques to the image domain by extracting *bags of visual words* from images and indexing these.

Key points

Text-search basics

- Most text-search systems (and textual document classification systems) represent the text in a form called a **bag of words**.
    - A **bag** is an unordered data structure like a **set**, but which unlike a set allows elements to be inserted multiple times.
    - In a bag of words, the order of the words in the document is irrelevant.
    - To create a bag of words from a text document, there are a two key processes:
        - Breaking the document into its constituent words (*tokenisation*);
        - Processing the words to reduce variability in the vocabulary
            - Often the words are processed using techniques like *stemming* (which removes variations in words like the letters *s* and *ing* at the end of some words).
            - Certain words are also removed (*stop word removal*) – words like "a", "the", "at", "which", etc. which don't have semantic meaning.
- There are a number of computational models for text search systems, but we're interested in one called the **vector-space model.**
    - In the vector-space model, text documents are represented by vectors
        - Obviously, this has analogies to the feature vectors we've been extracting from images.
    - The vectors from text documents contain counts of the number of times each word in the **lexicon** (the set of all possible words) occurs in the document.
        - Essentially the vectors are just **histograms of word counts**.
        - The vector for any given document is **highly sparse** – a document is only likely to contain a small proportion of all possible words!
    - Searching using the vector space model is simple:
        - A query can be turned into a vector form, and all the documents in the system can be ranked by their similarity to the query.
        - Cosine similarity (i.e. the angle between the vectors) is often used, as it is less affected by the vector magnitude (the query vector probably only contains a few words, so has a much lower magnitude (e.g. L1 or L2 norm) compared to the document vectors).

                - Many of the documents will have a similarity of 0 as they don't share any terms with the query.
            - Often, the cosine similarity function is modified to weight the elements of vectors being compared.
                - The intuition is that words that appear a lot in all documents should have less weight.
                - A commonly used weighting scheme is term **frequency-inverse document frequency** (*tf-idf)*
    - In practice, actual vectors are never created (it would just be too inefficient), and the bag of words is indexed directly in a structure called an **inverted index**.
        - An inverted index is a **map of words to postings lists**.
            - A **postings list** contains **postings**.
                - A **posting** is a *pair* containing a document identifier and word count.
                - Postings are only created if the word count is bigger than 1.
        - Using an inverted index, you can quickly find out which documents a word occurs in, and how many times that word occurs in each of those documents.
            - This allows for really efficient computation of the cosine similarity, as you only need to perform calculations for the words that actually appear in the query, and the documents containing those words.

Vector-quantisation

- Vector quantisation is a lossy data compression technique.
- Given a set of vectors, a technique like K-Means clustering can be used to learn a fixed size set of representative vectors.
- Vector quantisation is achieved by representing a vector by another approximate vector, which is drawn from a pool of representative vectors. Each input vector is assigned to the "closest" vector from the pool.

The bag-of-visual-words (BoVW)

- The **bag of *visual words*** methodology is an attempt to apply the techniques used for representing textual documents to the computer vision domain.
- A **visual word** is a local descriptor vector (e.g. a SIFT vector) that has been appropriately vector quantised to a representative vector.
    - The set of representative vectors is the *visual equivalent of the lexicon* – it's often referred to as a **codebook**.
    - In the case of SIFT, each visual word represents a prototypical pattern of local image gradients (and the underlying arrangement of pixels that created them).
- The (potentially variable sized) set of local descriptors representing an image can be transformed to a fixed dimensionality histogram formed by counting the number of occurrences of each representative vector.

BoVW Retrieval

- From the BoVW, retrieval follows naturally using the techniques developed for text retrieval.
    - Visual words can be indexed directly in an inverted index, and search can be performed using cosine similarity, etc.
    - There is one very important parameter: the size of the codebook (i.e. the number of possible visual words)
        - Inverted indexing will only work efficiently if the occurrence vectors are sparse!
        - Also want to ensure that the visual words a sufficiently distinctive to minimise mismatching.
            - Implies you need a very large codebook.
                - Typically around 1 million visual words in a modern retrieval system.

    - Performing k-means with millions of samples to learn 1 million centroids in a 128-dimensional space, followed by vector quantisation of many millions of features is non-trivial!
        - Nearest-neighbour search is massively expensive – have to use approximate techniques, like approximate k-d tree (mentioned last lecture) search to make it computationally tractable in a reasonable amount of time.
- Overall process for building a retrieval system:
    - Find interest points and extract local feature from all the images
    - Learn a codebook from (a sample of) the features
    - Perform vector quantisation to assign each feature to a representative visual work
    - Construct an inverted index

Further reading

- Wikipedia has good articles on:
    - The Vector-space model http://en.wikipedia.org/wiki/Vector_space_model
    - TF-IDF: http://en.wikipedia.org/wiki/Tf–idf
    - Inverted indexes: http://en.wikipedia.org/wiki/Inverted_index
    - Vector quantisation: http://en.wikipedia.org/wiki/Vector_quantization
    - Bag of Visual Words (and applications): http://en.wikipedia.org/wiki/Bag-of-words_model_in_computer_vision
- The seminal paper on using visual words for retrieval is the "Video Google" paper by Josef Sivic and Andrew Zisserman from Oxford: http://web.cs.swarthmore.edu/~turnbull/cs97/f08/paper/sivic03.pdf

Practical exercises

- Can you build your own bag-of-words representation for a set of images?
    - Recap:
        - Chapter 3 of the OpenIMAJ tutorial covers k-means clustering (you might want to use an approximate variant though)
        - Chapter 5 of the OpenIMAJ tutorial covers DoG-SIFT features which can be used as a basis for your visual words.
- With your bag of words histogram representations can you find some images that are similar to each other and some that are dissimilar?
    - What do you observe?