# 1 Tasks

The goal of this exercise is to test a simple image retrieval system and a min-Hash. We will use Oxford database of about 5k images. The data (images, detected features with assigned visual words) and some functions that will be useful are provided. For more details see section *Data and Matlab Code* at the end of this document.

You will need (the second step might have been done for you already):

- Copy directory `s:/Wednesday/Retrieval` to your home directory. Do not attempt to copy the images or the data.
- Start Matlab
- Run `ir_load` to load supporting data structures.

## 1.1 My first retrieval

Implement a simple image retrieval based on bag of words representation, dot product similarity measure and tf-idf weighting scheme. The database will be represented as a sparse matrix DB. Each row of the matrix represents a bag of words representation of one document in the database. Each column of the DB matrix corresponds to a visual word. An element DB(i,j) is non zero if and only if image i contains visual word j. The value of the element DB(i,j) is proportional to the term-frequency of visual word j in image i (how many times this visual word appears in the image) and the idf weight of the visual word j. The bag of words vector is normalized to unit length: `sum(DB(i,:).^2) == 1`. To prepare the database matrix, use the following provided function:

```
    DB = ir_createdb(IMS, word_weights);
```

The `word_weights` is a one dimensional array (one row) containing the weight of each visual word (see `help ir_createdb`). The first task is to compute the *idf* weights for each visual word.

$$idf(X) = \log \frac{\text{\# documents}}{\text{\# documents containing X}}$$

Then, the query will be represented as a sparse column vector $q$ – a bag of words representation of features inside the query region. The retrieval score for each image in the database is then simply obtained by a dot product `DB * q`.

Implement function

```
        [img_ids, score] = ir_query(DB, lbls, word_weights)
```

that evaluates a query with visual words `lbls` weighted by `word_weights` (idf, for example) on a database matrix `DB`. The output will be a vector of document ids sorted by their relevance to the query, and their retrieval score. First, create a sparse query vector `q`, normalized to length 1. Use Matlab function `sparse` rather than a `for` loop. Use Matlab function `[value, index] = sort(score,'descend')` to obtain the ordered list of similar images in the array `index`. Use function `ir_show_results` to display the results.

**How to get a query.** You can either use whole images to query with in the beginning. Use prepared functions `ir_showim`, `ir_imagedata`,`ir_select_bbx`, and `ir_ptsinbb` to select a region of an image as a query. Use your function `ir_query_vector` to prepare the query vector.

## 1.2 My first min-Hash

Script `mh_test` implements a simple use of the min-Hash algorithm. For each document, it generates a single sketch. All images with a sketch collision are shown – one colliding bin per figure. Modify the script so that is generates more sketches. To merge the results, simply concatenate collision of each sketch and then use function `unionfind`. Observe, how the results are changed with changing the following factors: visual words weights (use idf instead of 1), the number of sketches, the size of sketches.

## 1.3 Spatial re-ranking

In this step, take result of the *tf-idf* ranking, select top few (say 100) results and estimate the affine transformation as well as the number of geometrically consistent features. To estimate the transformation between the query image and the result use function affinem2m that is provided. For details look at help `affinem2m`. To re-rank the documents, use the score returned by `affinem2m` (if larger than 2).
Hint: read properly the help of `affinem2m`, you will need to call `ir_invgeom`.

## 1.4 * Quantitative evaluation of the image retrieval

There are 55 queries with known ground truth results. The queries are specified in queries.mat. This file contains an array QUERIES with the following fields:

| | |
|---|---|
| image_id : | id of the query image in the database |
| bbx : | bounding box of the query region |
| Name : | name of the query image |

The result, list of image ids sorted by the relevance, can be evaluated using function `ir_evaluate`. Its parameters are the query number (1 – 55) and the list of image ids. Script `ir_dbeval` runs evaluation over all 55 queries. Just add your retrieval function where specified.

## 1.5 ** Image clustering

Combine the min-Hash and the retrieval. First, obtain 'seeds' of clusters by min-Hash (and spatial verification). Use the seed images as queries into the image retrieval to find more relevant images.

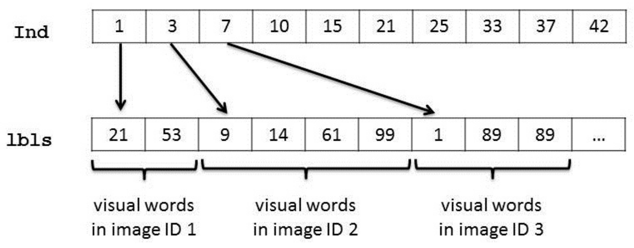
## 1.6 ** Query expansion

Implement the Query expansion. Take resulting images that are likely to be correct (have matching score more than 20, for example). Use the affine transformation returned by `affinem2m` function to back-project (`ir_geom_transA` function) the features from the result images to the query image (and clip the features by the query bounding box). Use the accumulated features as a new query. If you apply the geometry verification step, do not forged to sort the features according to the visual word id (for details see help `affinem2m`).

## 2 Data and Matlab Code

**Data structure.** All necessary data are stored in IMS data structure (`ir_load`). It has the following fields:

| | |
|---|---|
| lbls | int32 1xN array, visual words of all features in the database |
| geom | single 5xN array, geometric position of features in images |
| Ind | double 1xD+1, features of i-th image (in lbls and geom) start at index Ind(i), $D$ is the number of images |
| Names | strings cell array, names of the images in the database |
| imgs_size | double 2xD, actual size of images in pixels |
| nclus | double, the size of the vocabulary |
| dir | path to the image files |


Ind
1
3
7
10
15
21
25
33
37
42
lbls
21
53
9
14
61
99
1
89
89
...
visual words in image ID 1
visual words in image ID 2
visual words in image ID 3

**Examples.**
How many features are in image 501?

```
IMS.Ind(502)-IMS.Ind(501)
```

What visual words are in image 501?

```
IMS.lbls(IMS.Ind(501):IMS.Ind(502)-1)
```

Function `ir_imagedata` shows how to access visual words and geometry information of an image. You will need to directly access the lbls field for computational efficiency.

**Matlab code.** There is a number of functions to help you visualize the images and the results.

| | |
|---|---|
| `collect_duplicates` | collects sets of indices to identical rows of a matrix |
| `drawell` | draws elliptical regions in the image |
| `ir_createdb` | creates a sparse matrix bow representation of a database |
| `ir_geom_transA` | transforms features' geometry by an affine transformation |
| `ir_imagedata` | returns visual words and geometry of features in an image |
| `ir_imagetest` | script demonstrating how to use the displaying functions |
| `ir_ptsinbb` | returns indices of points that are inside a given rectangle |
| `ir_select_bbx` | allows to select a bounding box over an image with given id |
| `ir_showim` | displays image with given id |
| `ir_show_results` | allows to display result of your search (displays a list of images, and optionally transformed query bounding box) |
| `mh_minhashW` | generates a random weighted permutation for a min-Hash |
| `mh_sketch` | generates min-Hash signatures for documents |
| `mh_test` | test script for min-Hash |
| `unionfind` | Matlab implementation of the union-find algorithm |