10- bag of visual words

bag of words (bow)

① extract features from all the images in the dataset (SIFT, regular grid, interest point detector)

② cluster features for quantization (K-means) → number of clusters is important (underfit/overfit)

③ build the visual vocabulary (codebook) → compute weights for each word (inverse of frequency)

④ represent every image as a histogram of visual word or code-vector frequencies

⑤ given a new image, extract features and build a histogram

↳ map new image's features to the indexes of the closest words (codevectors) = feature quantization (cluster mean)

⑥ if the data has labels → train a classifier on the histograms (supervised)

⑦ if no labels, cluster the histograms too

textures = repeated elements, textons

inverse document frequency (IDF) = log (num_docs / num_docs_j_appears)

↳ term frequency-IDF scoring = give useless features low weights and the important ones high

↳ Bin_j_in_image = frequency_j_in_image * IDF_j

* if images has 1000 features, database has around 100.000 visual words, extremely sparse
* works well for image-level classification and recognizing object instances
* the performance degrades as the database grows

inverted file = mapping from words (codevectors) to documents (images) ★ → {1, 2, 3} ● → {1, 3}

↳ to compute the similarity quickly new img = {●, ★, ▲}

↳ it only considers database images whose bins overlap the query image (not whole database)

spatial pyramid matching = no location in considered in naive bow.

↳ compute histograms for each sub-regions, match them (always better match than single level)

naive bayes = to classify the histograms

↳ interested in the word if it is present so on