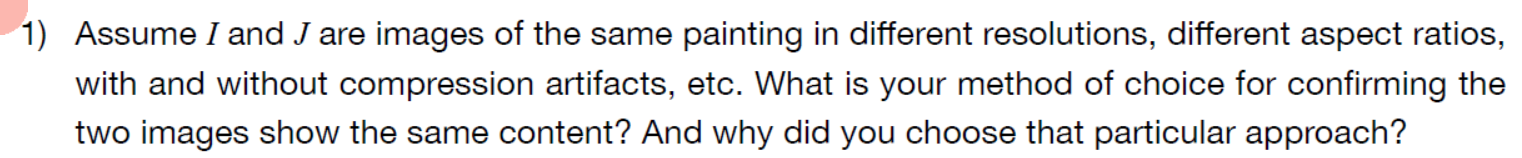
**Interview Challenge:**

**Computer Vision & Machine Learning for Image Retrieval Systems**

**Image Matching**

**The core functionality of Image Retrieval is based on the ability to identify matching images.**

****

*Three combinable approaches are given below.*

*Approach 1: Non-ideal case (e.g. photos of the painting)*

*If scale is different or perspective distortion is likely to be in at least one of the images, pixel comparison is not suitable. Instead, a Key Point Matching strategy based on Scale Invariant Feature Transform (SIFT) could be exploited, which is already implemented in opencv and its patent expired in the year 2020.*

*Since Key Points Matching (KPM) technique preserves structure information, a grey scale conversion previous to key point detection could be a good option. If color-structure information is relevant to the comparison, it is a good idea to apply the KPM technique to chrominance (Cb and Cr) and luminance (Y) channels rather than RGB channels. Where the latter is equivalent to the previous gray-scale conversion.*

*A very simple metric that can be used is the sum of distances between n best matches (those satisfying a ratio test, sorted and truncated). If good matches are less than n, assign an infinite metric. The larger the parameter n, the finer the resolution in the metric but it is more likely to have an infinite value.* ***See file image\_matching\_1.ipynb in repo*** [***https://github.com/carloshernangarrido/wc.git***](https://github.com/carloshernangarrido/wc.git) ***for a draft of the implementation.***

*A further step is to perform image registration using the matched key points by means the RANSAC technique, which removes outliers based on reprojection error. Now, the same metric can be applied to the reduced set of matching points. This further step is necessary when the perspective or the framing of photos is likely to be significantly different.*

*Once the metric is correctly working, a threshold should be defined using Signal Detection Theory on a labeled dataset, not necessarily large. The threshold could be selected by minimizing a cost function dependent on the cost of False Positives and of False Negatives. If these costs are not provided, the threshold value that maximizes informedness (Youden's J statistic) could be used.*

*Approach 2: Ideal case (e.g. versions of the same image available in different web pages, or post image registration)*

*Assuming the painting has the same scale in both images (the same part of the painting spans the same image proportion), and there is no relative aspect ratio distortion, translations, nor perspective distortion, i.e. images are centered equal projections; the following processing steps could be applied:*

*1. Crop both images in both dimensions to the smaller dimension. This solves the issue of aspect ratio difference.*

*2. Resample both images to a resolution coarser than that of the image with coarsest one. This reduces the computational cost of subsequent steps and allows comparison with classic metrics. Possibly, a resolution as low as 256x256 is good enough to distinguish classical painting, but finer resolution could be necessary to distinguish imitations. If one of the images has a resolution low enough, could be better to down-sample only the image with the finest resolution.*

*3. If one of the images could be in grey-scale (it can be checked with histograms), convert both images to grey-scale.*

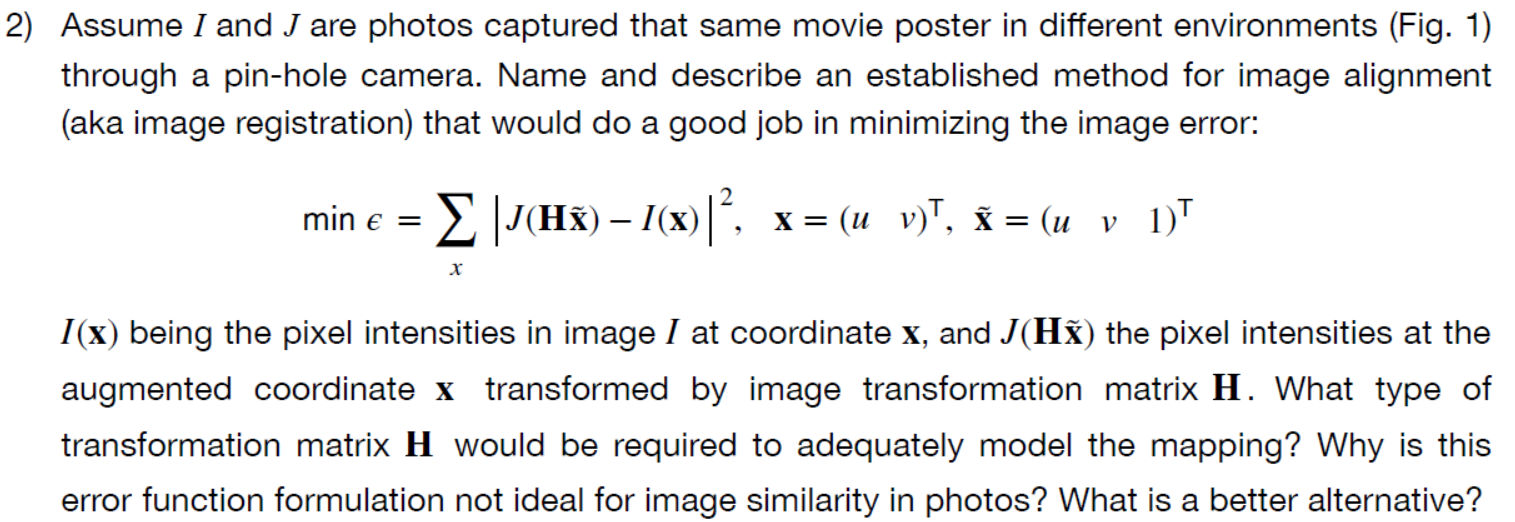
*4. If the images are to be compared with distance metrics, e.g. as mean square error, at least one of the images should be intensity rescaled in order to match the intensity scale of the other image. If the images are to be compared with correlation metrics, e.g. dot product, intensity re-scaling is not necessary.*

*If centering is not guaranteed, the central part of one image (e.g. 1/3 by 1/3) could be used as template for template matching in the other image.*

*A quicker approach could be measuring the correlation between histograms for each color channel. This does not require the 1-3 preprocessing steps. However, this approach is likely to confuse different paintings of the same artistic style, due to the loss of structure information. An obvious example is the case of the Chiaroscuro, which presents histograms with high frequencies for dark colors.*

*Combined Approach 3*

* *Use Approach 1 to register image J to image I*
* *Crop both images using the bounding box of key points (after outlier removal)*
* *Apply Approach 2 to the registered (and cropped) version of J and (cropped) I.*

****

*A stablished efficient method for minimizing the image error is based on FFT:*

*B. S. Reddy and B. N. Chatterji, "An FFT-based technique for translation, rotation, and scale-invariant image registration," in IEEE Transactions on Image Processing, vol. 5, no. 8, pp. 1266-1271, Aug. 1996, doi: 10.1109/83.506761.*

*where scaling and rotation properties of Fourier transform are exploited to find scale and rotation, and phase correlation technique is used to find translations.*

*or DFT, such as* [*https://github.com/matejak/imreg\_dft*](https://github.com/matejak/imreg_dft) *.*

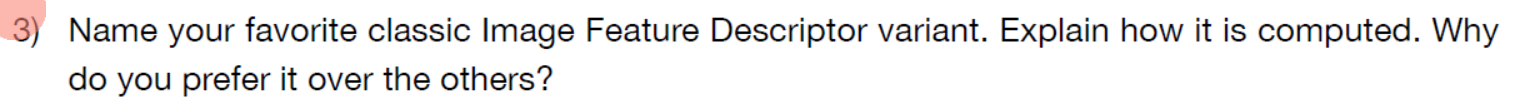
*In order to model the mapping, the transformation matrix should be a 2x3 matrix defining a (6 DOFs) 2D affine transform (scale, rotation, 2 translations, and 2 shears). This is the best option for the definitions given for the coordinates and augmented coordinates. However, even a full 2D affine transform cannot model perspective distortions as foreshortening or depth.*

*It is much better to use a homography (aka perspective transform), which has 8 DOFs.*

*The proposed error function is not good for image similarity in photos because of two reasons:*

* *It is computationally very expensive because it uses the pixels, which could lead to millions of terms in the objective function.*
* *The error function has low signal-to-noise ratio because a significant number of pixels do not actually match between the images. This is because of several reasons:*
  + *When images J and I are photos, they are different perspective projections of a hypothetical plane image. So, it is inevitable that one image has pixels (near the boundaries) that do not correspond to any of the pixels in the other image. Note that perspective projection is likely to produce shear and the domain of images is rectangular.*
  + *In a real-world case as the one shown in Fig. 1, it could happen that the images are not projections of the same plane image, but of similar images.*
  + *Compression artifacts affect the matching pixels differently in the images, even in the ideal case of using the same compression method, parameters and resolution.*

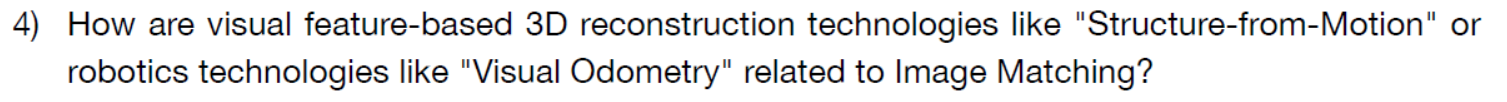
*Many of the problems of matching pixels could be mitigated with a pre-processing involving down-sampling of the images. However, it is a better alternative to* ***match key points.*** *That is, the perspective transform, i.e. a 3x3 matrix with 8 DOF, is found by minimizing an error function that compares locations of key points in image I with the projected locations of the matching key points of image J; besides outliers are iteratively removed. The outliers are those matching key points with large reprojection error (distance). This technique is named as RANdom Sample Consensus (RANSAC). A draft of the implementation is given in file* ***image\_matching\_2.ipynb of the repo*** [***https://github.com/carloshernangarrido/wc.git***](https://github.com/carloshernangarrido/wc.git)

**

*I start using SIFT because it has been largely tested, it works almost always (it is robust matching across affine distortion, change in 3D viewpoint, noise, and change in illumination). Besides, it is now patent-free.*

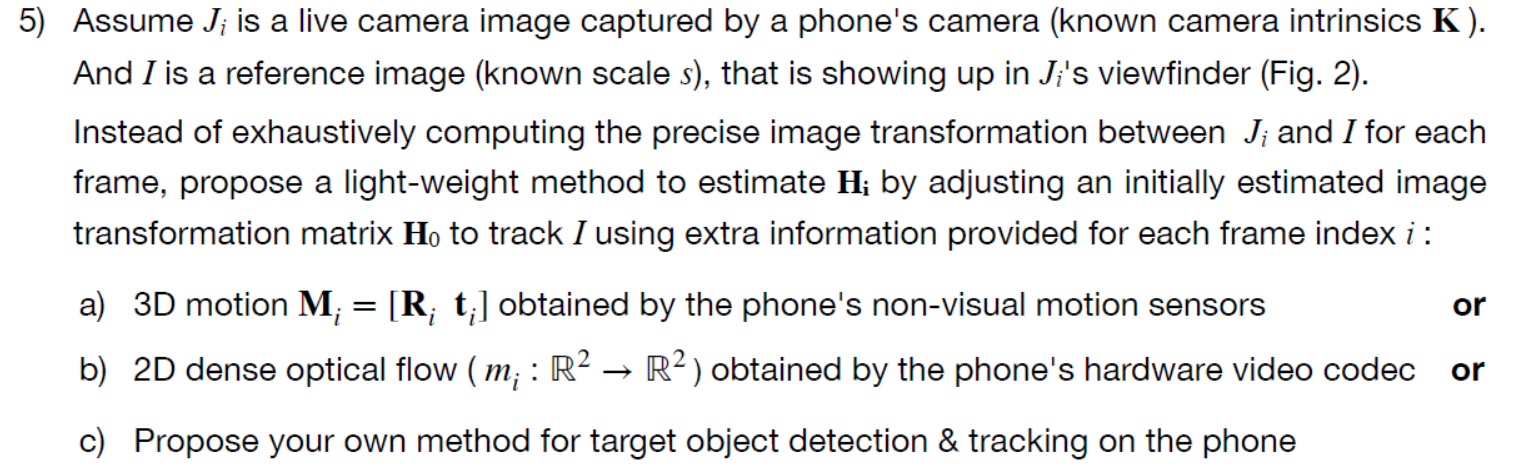
*The key steps of its computation are the following:*

* *Prior smoothing is performed to avoid too close key points.*
* *2D convolutions between the image and gaussian functions of different scales yield space-scale maps. The extrema in the difference of these maps for two nearby scales locate stable key points. This is because the difference-of-Gaussian function provides a close approximation to the scale-normalized Laplacian of Gaussian, and the extrema of this convoluted with the image are better key points locators than other well-known operations as gradient, hessian, or Harris corner. The difference-of-Gaussian function is not explicitly calculated, but distributive property of convolution is exploited to obtain difference-of-Gaussian function convoluted with the image for a set of different scales. This is done simply by subtracting adjacent image scales.*
  + *Several sets of multiple scales are grouped by octaves, which allows efficient resampling.*
* *Maxima and minima of the difference-of-Gaussian images are detected by comparing a pixel to its 26 neighbors in 3 × 3 regions at the current (8 neighbors) and adjacent scales (9+9 neighbors), and retaining only those lagger (lower) than all their neighbors.*
* *Extrema with low contrast are removed.*
* *Edges have high contrast and yield unstable extrema that are not removed by considering only contrast. Edge responses are removed using principal curvatures, but they are not directly calculated from the eigenvalues of the Hessian but invariance properties of trace and determinant are exploited to build a more efficient index.*
* *Scale and orientations are assigned to each key point location, and are sampled with a statistical procedure.*
* *Construction of the key point descriptor:*
  + *The gradient magnitude and orientation at each image sample point are computed in a region around the key point location, and weighted with a Gaussian window of proper scale.*
  + *Samples are accumulated into orientation histograms 4x4 regions.*
  + *A 128-element feature vector is constructed by taken 8 orientation beams in each of the 64 regions.*
  + *Finally, the feature vector is normalized to reduce the effects of illumination change.*

**

*Image matching, together with the camera model, allows calculating the relative pose of the camera for different photos of the same scene. Camera pose can be used for visual odometry, i.e. determination of the camera position and orientation from images.*

*Structure-from-motion consists in determining a 3D point cloud from matching key points across a set of 2D images and the camera model. The relation is that image matching is the first step of a Structure-from-motion workflow.*

**

*a) Assuming is an initially estimated homography transformation matrix such that:*

*where is the initial rotation matrix and is the initial translation vector corresponding to the initially estimated homography, they can be estimated as:*

*If 3D sensed motion has the form*

*where and are the current orientation and position, i.e.:*

*the rotations and translations relative to the first time-step can be estimated as:*

*Then, the subsequent homography matrices can be calculated as:*

*If the initial homography is found with cv2.findHomography(), it must be taken into account that such method returns a transformation matrix for planar homography, i.e. a 3x3 matrix such that:*

*If is arbitrarily set to 1, then:*

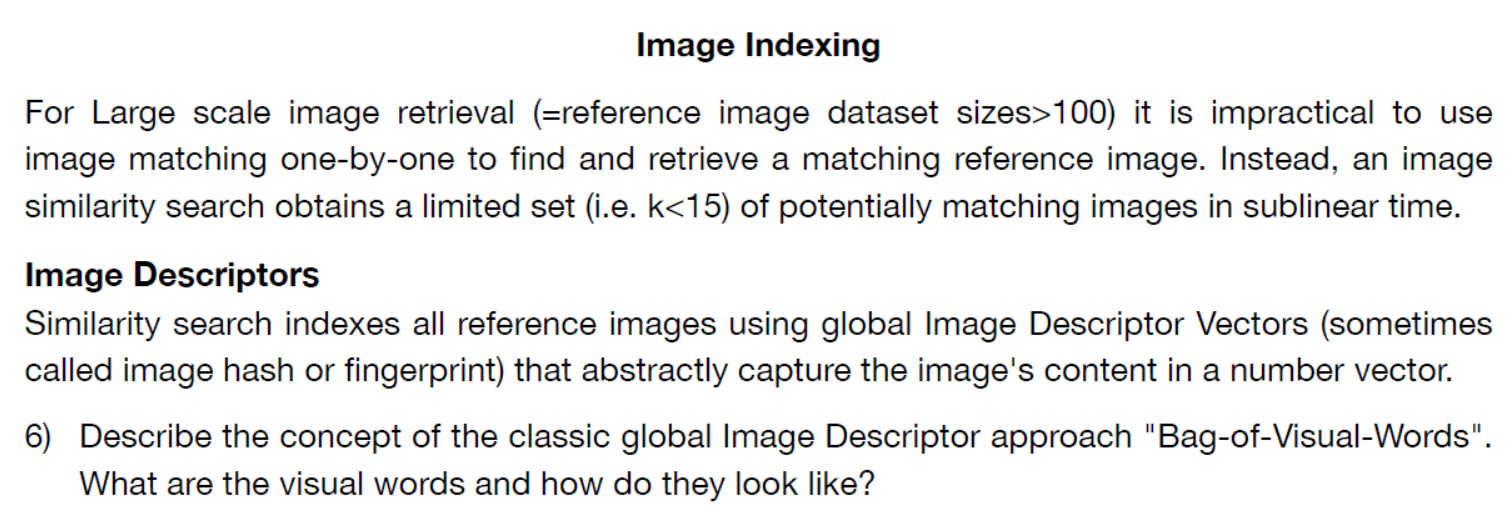
***Important****: Since can accumulate error with time, it is necessary to re—estimate each a certain amount of time, e.g. each 1 second.*

*b) 2D dense optical flow is quickly calculated by the video codec. However, its use requires extracting mean velocity from the set of all the 2x2 pixel regions in the image; which is computationally more expensive than using non-visual motion sensors. The mean velocity vector, together with the sample rate (or time step) could be used to translate the image back to so is still valid. This approach wont work if the phone rotates too close to the target object.*

*c) Target object detection and tracking*

*Here are some basic steps:*

* *Define an image of the target object.*
* *Match key points of the target object image to the camera image.*
* *Register the camera image to the target object image.*
* *Check with the Combined Approach given in point 1 if the content is likely to be the same.*
  + *If not, stop tracking.*
  + *If the content is the same, use the homography estimated during image registration to map a rectangular bounding box from the target object image to the camera image and draw the resulting quadrilateral over the camera image with a suitable label.*
  + *Update the homography 10-20 times per second using non-visual motion sensors*
  + *Update the homography each 1-2 seconds using again key point matching.*
  + *Repeat until user intervention.*

**

*Constructing a* ***global*** *image descriptor using the BoVW approach involves the following steps:*

*(i) automatically detect regions/points of interest,*

*(ii) compute* ***local*** *descriptors (e.g. SIFT) over those regions/points,*

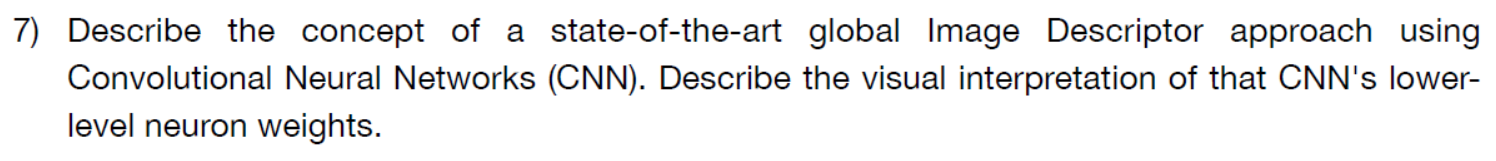
*(iii) quantize (e.g. using k-means) the descriptors into words to form the visual vocabulary, and*

*(iv) find the occurrences in the image of each specific word in the vocabulary for constructing the BoW feature (or a histogram of word frequencies)*

*Step (iii) is the most computationally demanding.*

*Step (iv) can implement strategies similar to those used in NLP; e.g., term frequency-inverse document frequency.*

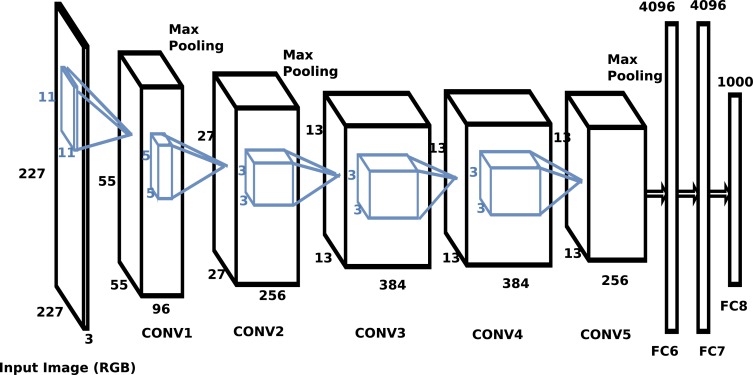
*A bag of visual words is a vector representing the histogram of visual word frequencies, where “visual word” means one of the possible values of a quantized local descriptor (abstract representation of a pattern).* ***Each element of the vector represents how many times appear in the image a certain pattern.***

**

*A state-of-the-art approach can be found in: Maria Tzelepi, Anastasios Tefas,*

*Deep convolutional learning for Content Based Image Retrieval, Neurocomputing, Volume 275, 2018, https://doi.org/10.1016/j.neucom.2017.11.022.*

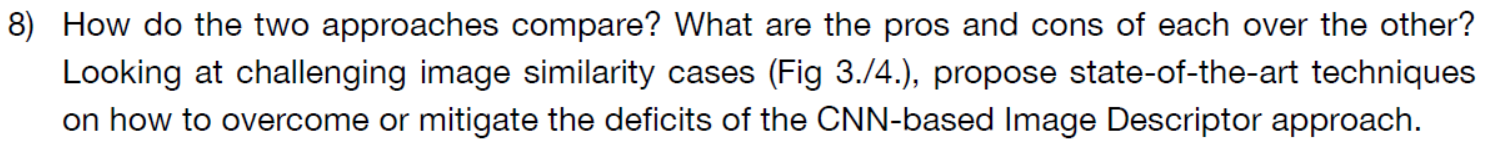
*The authors use the BVLC Reference CaffeNet model, an 8-layers CNN pre-trained on the on the ImageNet Large Scale Visual Recognition Challenge for classification into 1000 classes. The architecture is shown in the figure below:*



*The authors remove the last layers and use the activations of one of the convolutional layers as a set of feature maps (as many maps as filters in the last remaining convolutional layer). Interestingly, the weights learned for a* ***classification task*** *are used for a* ***feature extraction task****. In order to reduce dimensionality of the output, they add an extra pooling layer, the so-called Maximum Activations of Convolutions (MAC) layer, which implements the max-pooling operation over the width and height of the output volume, for each of the feature maps.*

*The MAC outputs are used then as target representations in a retraining for a* ***regression task****. The authors propose three retraining approaches, here I describe the first, name as Fully Unsupervised (FU). They retrain the pretrained CNN model on the given dataset, aiming at maximizing the cosine similarity between each image representation and its n nearest representations, in terms of cosine distance. The effect of this retraining is to amplify the original presumption that the relevant image representations are closer to the certain query representation in the feature space.*

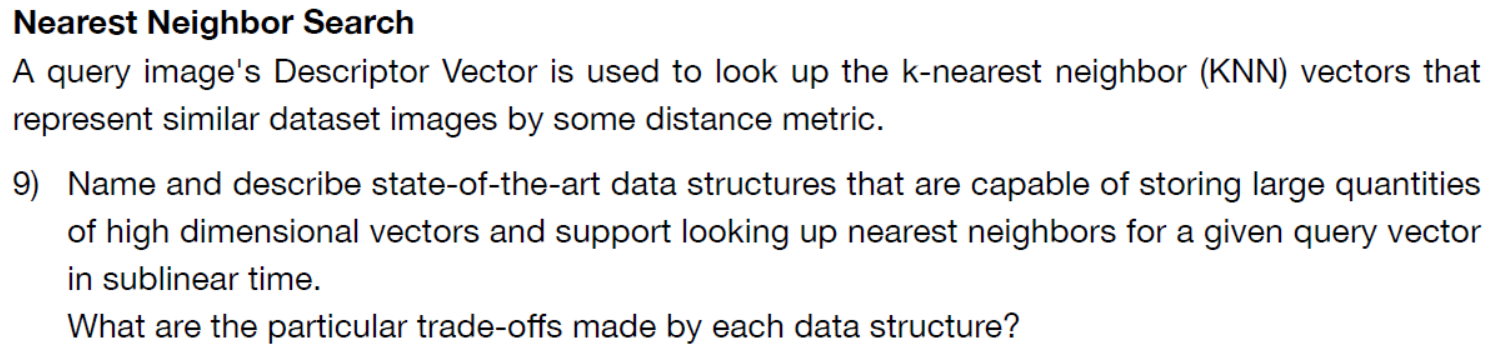
*The lower-level neuron weights in the convolutional layers are the spatial patterns that are useful to classify the images. In the first convolutional layer, these patterns are grouped simply by color (one input channel per color). In the subsequent convolutional layers, the patterns correspond to different space scales, being the last ones those with the coarser scales.*

**

*The MAC outputs of the retrained model can be used a fingerprint of the image, similar to a bag of visual words.*

*The main disadvantage of BoVW is that the spatial distribution of characteristic patterns is lost in the histogram step. CNN preserves information of spatial distribution, but is sensitive to the training datasets.*

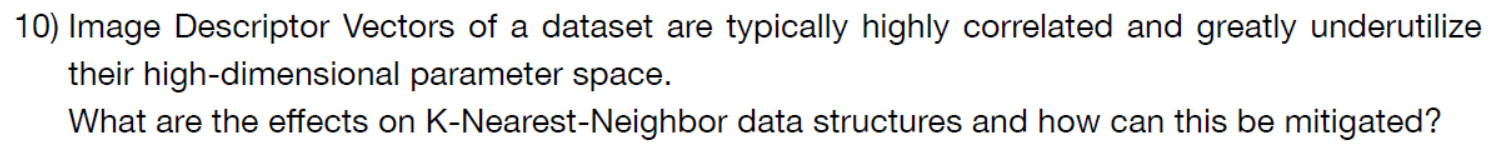
*Obviously, the ML approach is conditioned by the pre-training dataset and the number of classes. However, the general idea of pre-training on classification and retraining the convolutional part to amplify similarity recognition can be applied to more specific datasets.*

**

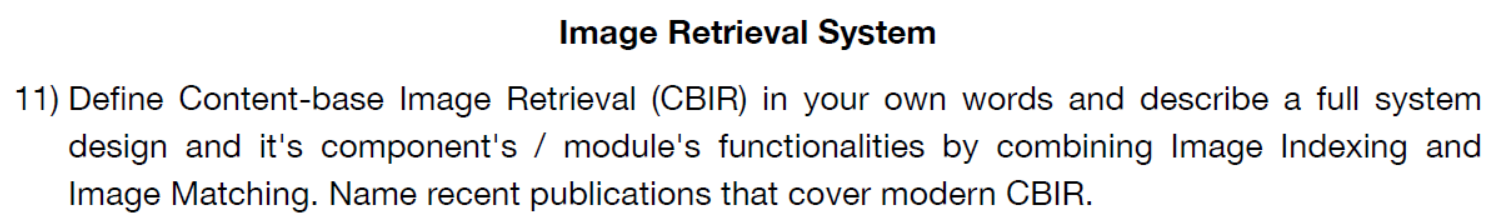
*FLANN is a library for performing fast approximate nearest neighbor searches in high dimensional spaces. It is written in C++ and contains bindings for the following languages: C, MATLAB, Python, and Ruby. This library implements algorithms that use one of two data structures:*

* *hierarchical k-means tree: it is constructed by splitting the data points at each level into K distinct regions using a k-means clustering, and then applying the same method recursively to the points in each region.*
* *multiple randomized kd-trees: they are built by choosing the split dimension randomly from the first D (typically 5) dimensions on which data has the greatest variance.*

*FLANN has the advantage of automatic algorithm configuration from a dataset.*

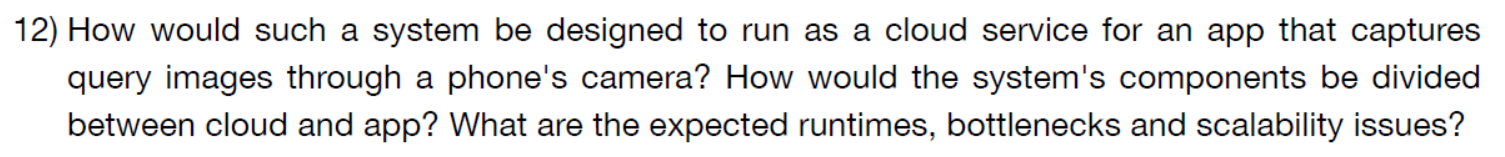
**

*There is redundant information across the levels of the trees. If the image descriptor vectors are highly correlated, linear dimensionality reduction techniques as PCA could be used to compress the vector prior to knn search.*

**

*CBIR is the task (or the system that performs it) of, from a query image supplied by the user, searching in a set of images for those that likely have the same content of the query image. For example, if the user supplies the picture of an apple, it is expected that the system retrieves a small set (e.g., 10) images containing apples. From the point of view of the user, it is a problem similar to face recognition and fingerprint lookup; but less specific.*

*Ibtihaal M. Hameed, Sadiq H. Abdulhussain & Basheera M. Mahmmod | D T Pham (Reviewing editor) (2021) Content-based image retrieval: A review of recent trends, Cogent Engineering, 8:1, DOI: 10.1080/23311916.2021.1927469*

**

*I would perform in the app the following tasks:*

* *Take the initial picture of the target object*
* *Give the user the possibility of defining a bounding box of interest*
* *Track and label with a perspective bounding box the target image by using the techniques proposed above (key point matching, image registration, homography, non-visual sensors).*
* *Select a set of images that give different cropped points-of-view of the same target object.*
* *Send query images to the cloud server*

*In the cloud, I would implement:*

* *Model training (development)*
* *Extraction of Global image descriptor for each image of the set*
* *Perform efficient k-nearest neighbors*
* *Return the result set of images matching*

*Probably the Extraction of Global image descriptor for each image of the set could be performed in the app after (say) 5 seconds. Then, only the descriptors must be sent. I do not have enough experience to confirm it, but I feel that latency is more important than transfer rate in this case. So, sending a set of images could be better than calculating the descriptors locally in the app.*

*On the other hand, calculating the descriptors in the cloud has the advantage of not needing updates in the app when the descriptor extraction module is changed.*