FEATURE MATCHING

Bisogna capire come fare match fra due immagini A e B, ognuna con il proprio set di feature descriptor N_A e N_B.

È vero che $|N_A| = |N_B|$? Generalmente no!

Per fare matching si usa la forza bruta: $\forall \psi_a \in N_A$ cerchiamo un match in N_B.

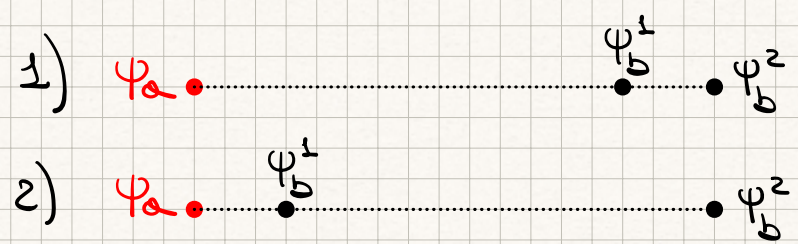
ψ_a è accoppiato con ψ_b come segue:

$$\psi_b = \underset{\psi \in N_B}{\arg\min} \|\psi - \psi_a\|^2$$

Quindi ψ_b è il più vicino descrittore in N_B a ψ_a per la distanza euclidea.

Problema: esiste sempre un descrittore "più vicino", anche se è molto lontano. Bisogna fare il confronto con tutti i descrittori, e non lo sappiamo a priori la distanza.

1) Possiamo inserire una threshold a indicare la massima distanza accettabile. Bene, ma qual è un valore ottimale?
Bisogna usare il concetto di distanza relativa:



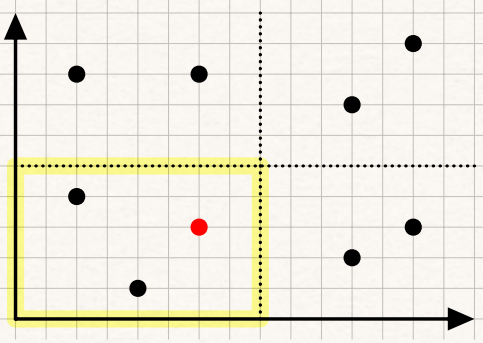
Ovviamente la seconda situazione è migliore perché possiamo distinguere meglio il match.

First-to-second nearest neighbor distance ratio (NNDR)
Dato $\psi_a \in N_A$ e cercando ψ_b^1 e $\psi_b^2 \in N_B$ il più vicino e il secondo più vicino feature descriptor a ψ_a, settiamo:

$$NNDR(\psi_a, \psi_b^1, \psi_b^2) = \frac{\|\psi_a - \psi_b^1\|}{\|\psi_a - \psi_b^2\|}$$

Se NNDR è basso, ψ_b^1 è un buon candidato per essere un buon match.
Empiricamente settiamo $NNDR = 0.8$ per avere buoni candidati (0.8 o meno).

Nota: Questo metodo funziona se abbiamo poche immagini nel DB. Bisogna trovare i ψ_b più vicini e confrontarli, complessità $O(n^2)$. Troppo.
Per questo viene usata una diversa struttura dati detta KD-tree.
Per costruire un KD-tree basta dividere lo spazio in griglie e limitare la ricerca nella griglia in cui cade l'ingresso:



Questo approccio soffre quando si hanno troppe dimensioni, e noi ne abbiamo tante...

Approximate Nearest Neighboor

La tecnologia chiave per i moderni ANN sono i vector quantisiation:

$$\Psi = \{\psi_1, \dots, \psi_m\} \quad \text{con} \quad \psi_i \in \mathbb{R}^d$$

Abbiamo il set $C = \{\mu_1, \dots, \mu_k\}$ con $\mu_i \in \mathbb{R}^d$.
C è detto codebook e i suoi elementi codeword.

VECTOR QUANTIZER

$$q : \Psi \to C$$

Un vector quantizer mappa ψ_i al suo più vicino μ_j secondo una qualche distanza

$$q(\psi) = \underset{\mu \in C}{\operatorname{argmin}} \| \psi - \mu \|$$

VORONOI CELL

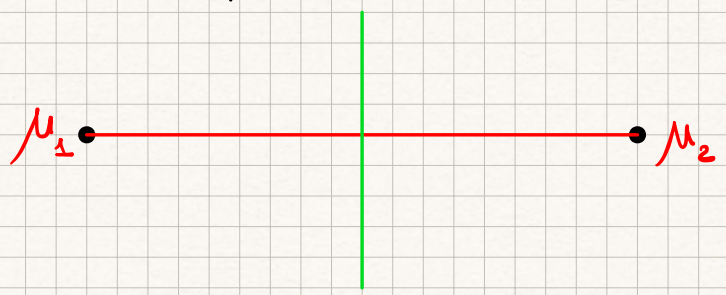
Dato una codword $\mu_i \in C$ definiamo il suo Voronoi cell V_i come:

$$V_i = \{x \in \mathbb{R}^d : q(x) = \mu_i\}$$

Ovvero è l'insieme di $x \in \mathbb{R}^d$ che hanno lo stesso vector quantizer μ_i (Il più vicino). Pertanto tutti gli elementi di una cella di voronoi sono codificati con lo stesso codeword

Il set di tutti gli V_i di un dato codebook è detto VORONOI TOSSELATION (Tassellazione di Voronoi):

Esempio
Se abbiamo due punti



Ha senso chiedersi se esista un ottimo codebook per un dato set Ψ di vettori. Un codebook C^* è ottimo per Ψ se minimizza la reconstruction mean square error, il che vuol dire

$$C_t^* = \underset{C}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} \| \psi_i - q(\psi_i) \|^2$$

Dato Ψ, il suo ottimo codebook C^* con K codeword fisse può essere calcolato con K-means algorithm.
I centroidi calcolati da K-means corrispondono ai $\mu_1^* \dots \mu_K^*$.

Dato un codebook C un inverted file index (IVF) costruito con Ψ e C, salva gli elementi di Ψ in K partizioni, chiamate anche posting list $L_1 \dots L_K$ dove

$$L_i = \{ \psi_j \in \Psi : q(\psi_j) = \mu_i \} \quad \forall i = 1 \dots K$$

A tempo di query processing, bisogna specificare il numero di partizioni p in cui si vuole cercare. Se $p<k$ la ricerca è approssimata o addirittura inesatta.

VISUAL BAG OF WORDS

Nel 2003 Google propone un approccio differente, per la distanza complesse da calcolare, basandosi sui modelli di tipo bag of word.
In questi modelli le feature locali sono considerate come parole e le immagini come documenti.

Vector quantization è usata per il train di un piccolo ($O(1000)$) codebook a partire dai descrittori di immagini. Queste Voronoi Cell sono chiamate VISUAL WORDS.
Feature locali vengono quantificate e assegnate a visual words, creando VISUAL DOCUMENT.

Quindi le query e le immagini sono accoppiate da uno score del tip TFIDF-like formula. I pesi di TFIDF $w_i(g_j)$ della visual word della immagine g_j:

$$w_i(g_j) = \underbrace{TF(\mu_i, g_j)}_{\text{numero di occorrenze della visual word } \mu_i \text{ nell'immagine } g_j} \cdot \underbrace{IDF(\mu_i)}_{\text{Questa è IDF di } \mu_i \text{ in tutta la collezione.}}$$

$$IDF(\mu_i) = \log \frac{n}{N_{\mu_i}}$$ → numero di immagini contenenti μ_i

Questo modello è detto visual bag of word.