Scene classification

Questo esercizio ci chiede di creare un classificatore di scene basato sull'approccio Bag Of Words e ci dà uno skeleton code da usare come base di partenza.

L'approccio Bag Of Words è simile al processo umano di identificazione e descrizione di un'immagine con un numero limitato di parole. Mediante le parole impiegate per descrivere l'immagine, si può intuire di cosa parliamo.

Abbiamo 4 classi di scene (che per comodità di programmazione abbiamo associato a dei numeri), un dataset di immagini , dei descrittori SIFT già precalcolati. Il dataset va diviso per funzionalità: abbiamo un numero di immagini per il training e il rimanente numero per il testing del nostro classificatore.

Lo script da richiamare per eseguire l'intero algoritmo è **esercizio2ofass3.m**.

Fase di training

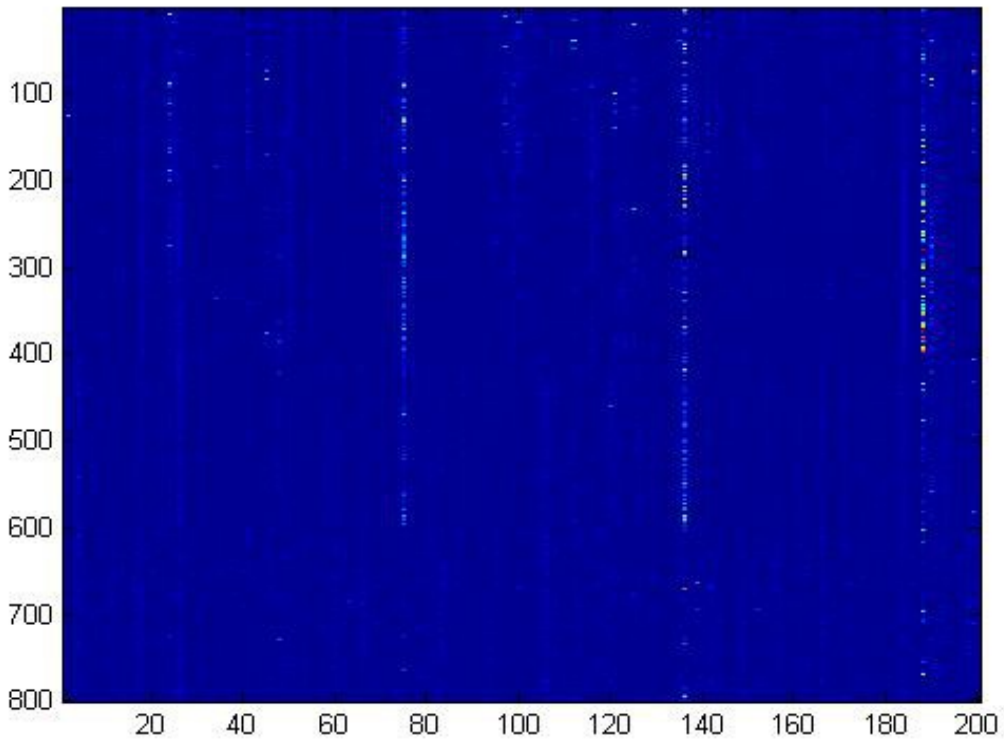
Dopo aver importato il dataset, dobbiamo ricavarci i vari istogrammi associati alle varie immagini.

Richiamiamo lo script **vq_sift_luke.m**.

Data un'immagine i, fornita di n descrittori (precalcolati mediante SIFT) vogliamo determinare per ciascun descrittore a quale parola del dizionario si avvicina. Per fare questo usiamo la funzione knnsearch che date due matrici restituisce un'array di indici ed (opzionalmente) la distanza, che è la minima trovata. Si potrebbe anche eseguire una ricerca con confronto "manuale", ma le performance calano drasticamente.

Ogni qualvolta, data la i-esima immagine e il j-esimo descrittore, viene scelta una parola, il relativo bin dell'istogramma viene incrementato di uno. Alla fine, avremo una matrice con tante righe quanto sono le immagini e con 200 colonne, tante quante sono le parole del nostro dizionario.

Ecco la matrice degli istogrammi, dopo aver scansito il dataset (immagina compressa con jpeg e ulteriormente compressa da libreoffice):



Fase di classificazione

Una volta creato una matrice (per l'uso che vediamo dopo) history, richiamo lo script **classify_nn_luke**.
Questo script accetta in input `TRAIN_IND,TEST_IND,TRAIN_LABEL,TEST_LABEL,HISTOGRAMS,K,` cioè rispettivamente l'elenco degli ID delle immagini di training, l'elenco degli ID delle immagini di test, le label delle immagini di training, le label delle immagini di test, la matrice degli istogrammi creata prima e il parametro K, con cui specifico quanti vicini voglio considerare per la predizione della mia classe.

Una volta scissa la matrice degli istogrami in due (training e test), cerco per ogni istogramma associato ad un'immagine di test, i K istogrammi di training più vicini (distanza euclidea) con annesse label. Ogni qual volta devo decidere quale delle K label considerare ai fini della mia predizione, scelgo quella con maggiore frequenza e creo il mio array di label predette.

Ora devo verificare quante delle label predette sono esatte e lo faccio semplicemente contando i riscontri esatti. Da questa somma mi ricavo la percentuale e la restituisco allo script chiamante.

Comportamento al variare di K

Vediamo come si comporta l'algoritmo al variare di K. Selezioniamo sempre numeri dispari in modo da non avere situazioni con un numero uguale di frequenze. Da K=1 fino a K=5 l'accuratezza aumenta perchè possiamo discernere tra tutte e 4 le classi in maniera corretta, però poi questa accuratezza cala drasticamente perchè introduciamo troppa ambiguità e consideriamo ai fini del voto anche campioni molto distanti.