# Face Recognition

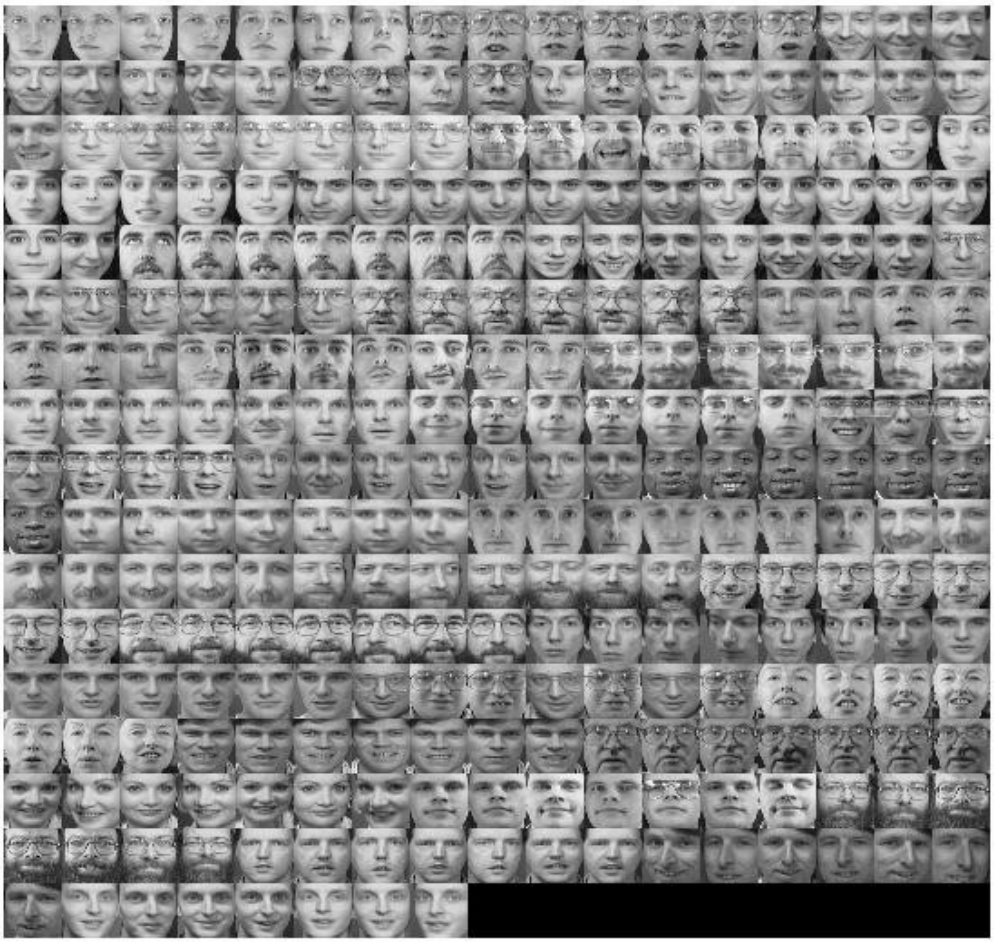
In questo esercizio, devo implementare un riconoscitore di volti usando una metrica basata sull'analisi delle caratteristiche tipiche di un volto.
L'analisi nello spazio delle facce prevede che data una faccia "media", le altre facce si trovino lungo le varie direzioni dello spazio (ogni direzione prevede un mutamento di una data caratteristica del volto).

Il problema consiste nel determinare la faccia media di riferimento, indicare al computer nella fase di training in che direzioni si è "mossa" l'ennesima faccia di training rispetto a quella media e data una faccia ignota, associarla a uno dei volti presenti in archivio con l'accuratezza migliore possibile.

L'idea è di "muovere" la faccia da esaminare lungo delle direzioni fino a quando si troverà nella regione dello spazio costituita dalle facce che le somigliano di più. Ogni faccia di questo spazio multidimensionale è già etichettata, così una volta trovati dei riscontri, possiamo identificare il volto.

La prima fase è preparare le strutture dati che ci serviranno, cioè il set di training e il set di testing e i rispettivi elenchi di label.

Training Set originale



Ecco qui il set di training:

A questo punto, ho ricavato la faccia media e l'ho sottratta a tutte le altre di training, in modo da ricavarmi una matrice di "differenze". Lo stesso farò anche per il dataset di test.

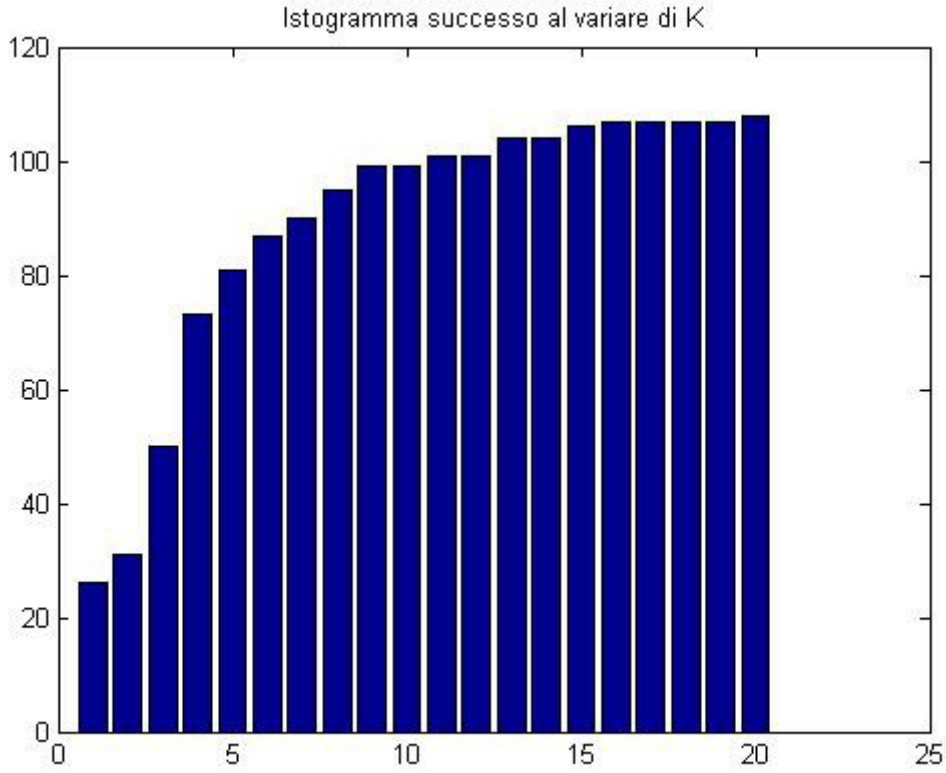
Mi ricavo da questa matrice quella di covarianza C e imposto il numero massimo di componenti che voglio estrarre k_max.

Inizializzo un vettore di successi che userò alla fine.

Per ogni K (che mi indica quanti componenti di C voglio prendere ad ogni iterazione) eseguo i seguenti passi:

1. mi ricavo i K autovettori tramite il comando **`eigs`**.
2. Se K è uguale al massimo valore impostato (nel nostro caso 20) mostro a schermo le eigen-faces e le stesse una volta riapplicata la faccia media.
3. Creo la matrice dei descrittori **p** associata al dataset di training.
4. Devo provare la correttezza del mio operato, pertanto ricostruisco il dataset di training usando **p** e **v** e lo confronto con quello iniziale. Se ho proceduto bene, i due dataset devono essere sempre più simili al crescere di K.
5. Creo la matrice dei descrittori **q** associato al dataset di test usando le componenti **v** calcolate in fase di training.
6. Per ogni elemento di q cerco il più vicino in p, in termini di distanza euclidea e usando `knnsearch` per maggiore efficienza.
7. Confronto ogni label predetta con quella vera e in caso di successo incremento il bin relativo a K nell'istogramma dei successi.

Mostro a schermo l'istogramma ottenuto:



Adesso voglio mostrare che al variare di K la classificazione è sempre più precisa, pertando riporterò per K = 1, per K=5, per K=12 e per K=20 le eigen-faces, sia naturali che ricomposte con la faccia media, e i training set ricostruiti.

**Eigenfaces**:

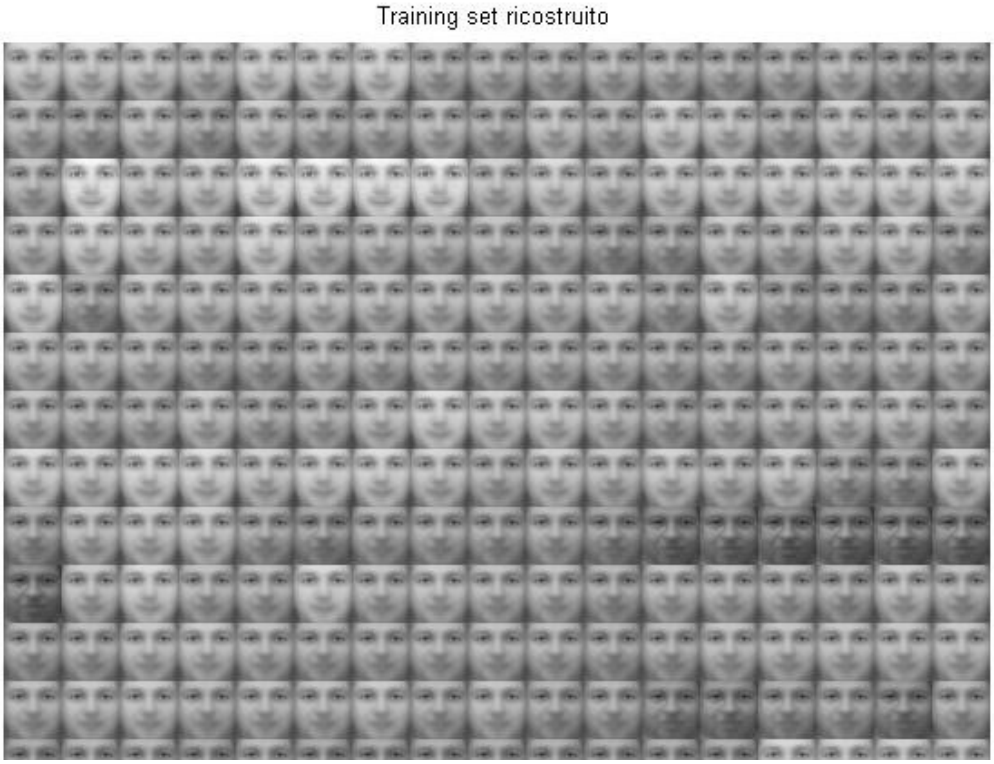


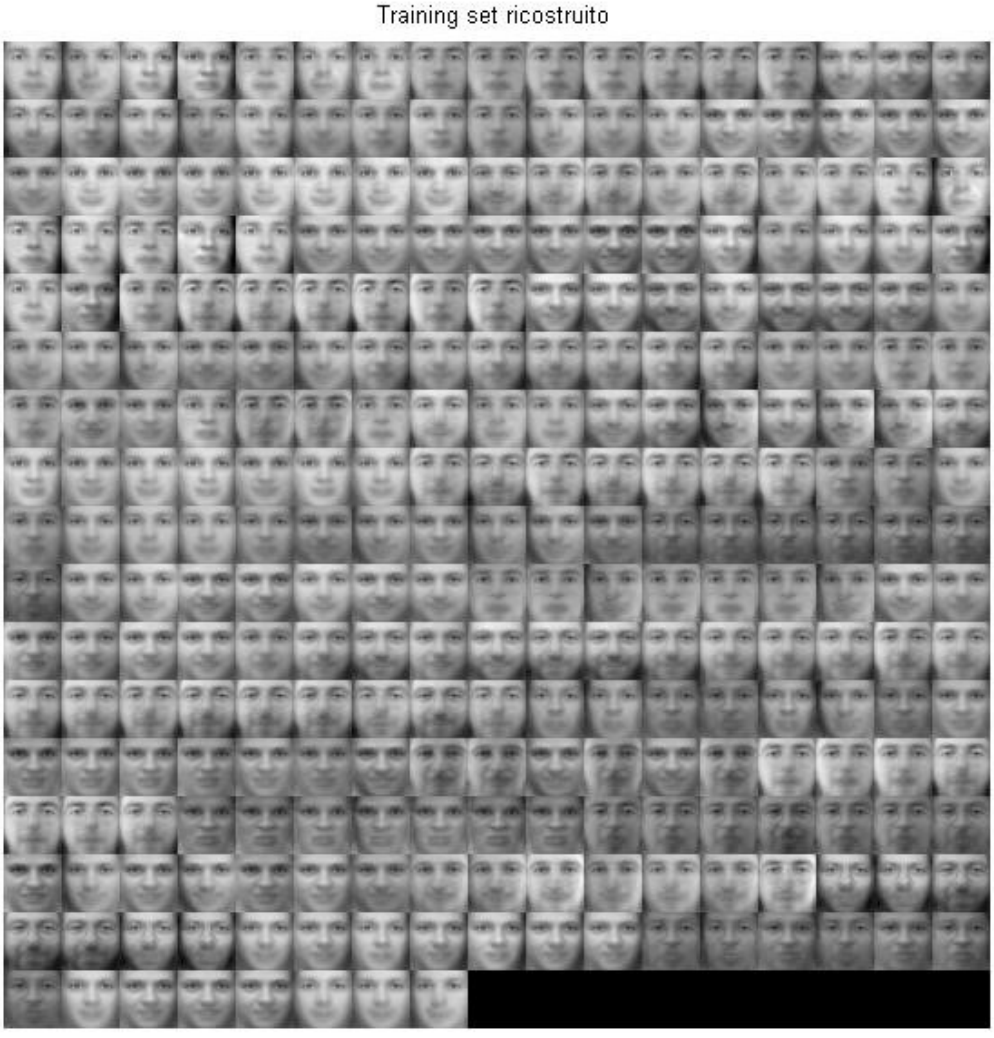


**Eigenfaces ricostruite con la faccia media riapplicata**:

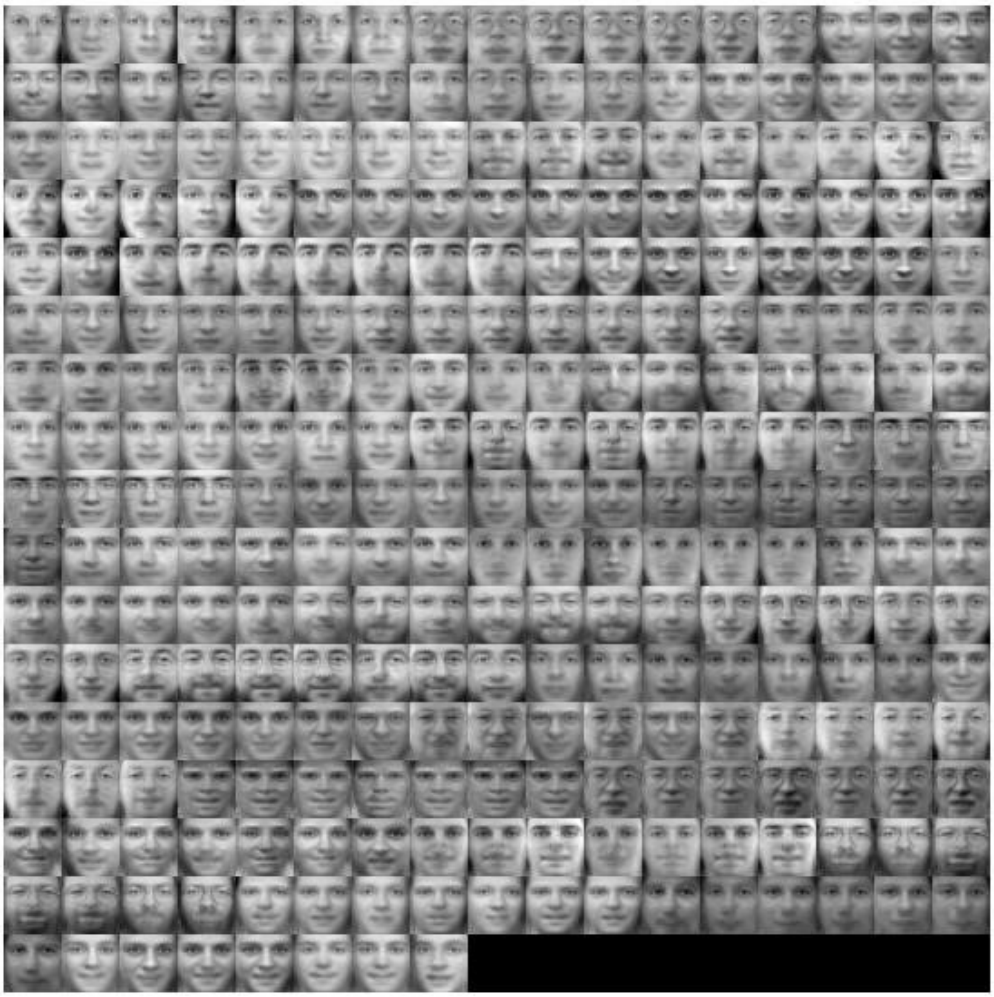



**Training set ricostruiti**:

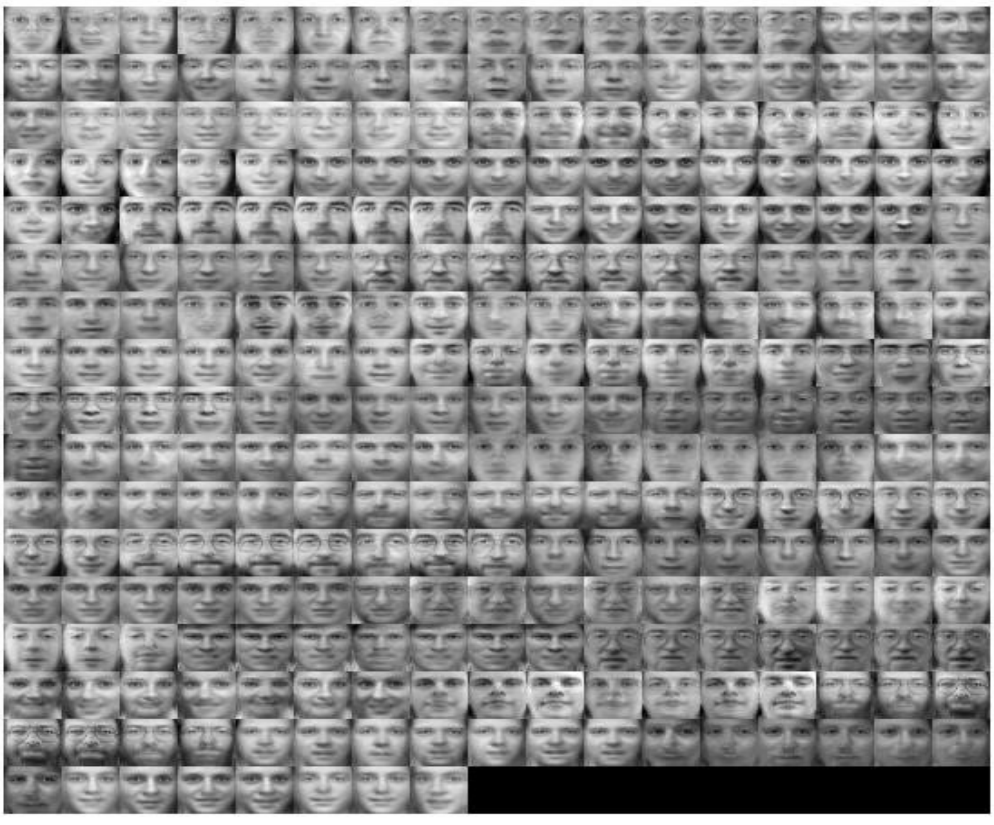


Training set ricostruito



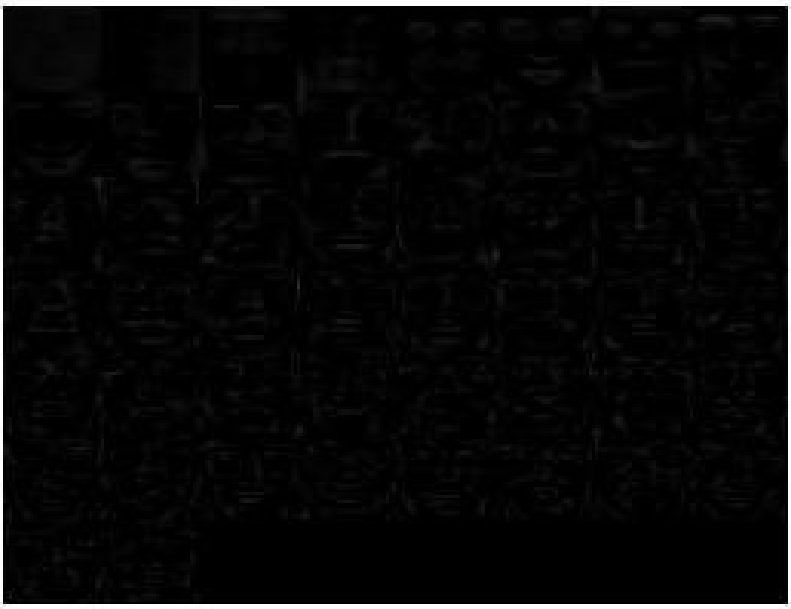
Training set ricostruito



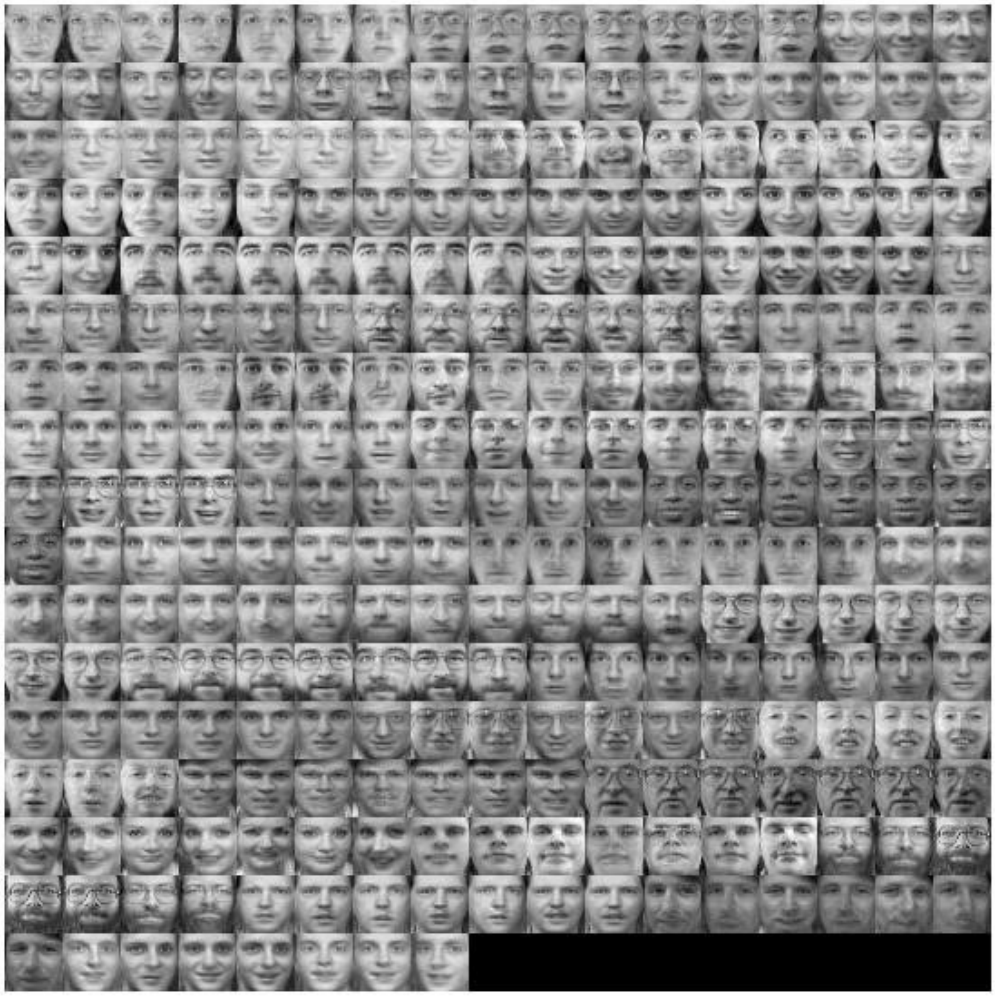
A questo punto mi sono chiesto fino a quanto potessi spingermi con la scelta di K per avere risultati migliori. Ho scelto K=50.

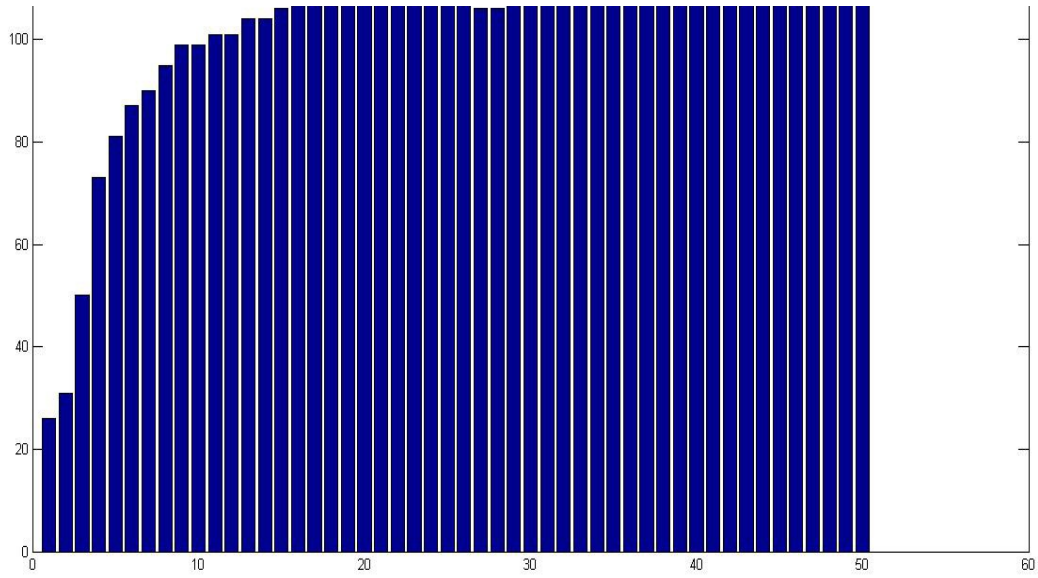
EigenFaces con la faccia media aggiunta di nuovo

EigenFaces



Training set ricostruito





Rispetto a K=20 la ricostruzione del dataset ha dato risultati più accurati,  ma il tasso delle facce riconosciute tende a crescere sempre più lentamente. Evidentemente, per risultati più accurati non basta solo prelevare più componenti possibili, ma bisogna anche estendere il dataset.