We used the ORL face database composed of 400 images of size 112 x 92. There are 40 persons, 10 images per each person. The images were taken at different times, lighting and facial expressions. The faces are in an upright position in frontal view, with a slight left-right rotation. In example we performed factorization on original face images by constructing a matrix of shape 10304 (pixels) x 400 (faces). To avoid too large values, the data matrix is divided by max value present in all the images. Indeed, this division does not have any major impact on performance of the Matrix Factorization methods. The data has been split in two-parts train and test. For each catalog we use the first 9 photograph s for training and the last photograph for test. However, we only use the training part for this lab. The final data matrix shape is given 10304 (pixels) x 360 (faces)

All the initial image processing is given in main_lab_function.m . in this function you will need to call the NMF function.

We want to decompose the matrix V=BW.

1) Create the NMF function

```
function [B,W,obj,k] = nmf(V,rank,max_iter,lambda)

% NMF - Non-negative matrix factorization
% [B,W,OBJ,NUM_ITER] = NMF(V,RANK,MAX_ITER,LAMBDA)
% V - Input data.
% RANK - Rank size.
% MAX_ITER - Maximum number of iterations (default 50).
% LAMBDA - Convergence step size (default 0.0001).
% B - Set of basis images.
% W - Set of basis coefficients.
% OBJ - Objective function output.
% NUM_ITER - Number of iterations run.


% Create initial matrices random initialization
% Make sure W has unit sum columns! (each column should sum
to one.


% Calculate initial objective
```

As seen in the lecture for the KL divergence

```
% Start iteration
```

$$B = B \otimes \frac{\left(\frac{V}{BW}\right)W^T}{1W^T} \qquad W = W \otimes \frac{B^T\left(\frac{V}{BW}\right)}{B^T 1}$$

$\frac{\cdot\cdot}{\cdot\cdot}$ The division is elements by element.
$\otimes$ The multiplication is elements by elements

```
% Calculate the new objective function value.

Stop when the absolute value of the new objective function
value - old objective value is smaller or equal to Lambda or
the max number of iterations is yield.

The objective function is given as fellow
function [obj] = compute_objective(V,W,H)
  obj = sum(sum(V.*log(W*H) - (W*H)));
```

2) Output the new bases and weights and plot all the 40 new NMF bases as given in the main script.

If you call your NMF function as fellow,

```
[B,W,obj,k] = nmf(X,40,500,0.001);
```

you will obtain something like fellow

3) Compare your results to the NMF predefined function in Matlab

```
opt = statset('MaxIter',500,'Display','final');
[B,W] = nnmf(X,40,'options',opt,'algorithm','mult');
```

4) Sparsity: (extra)

Imposing sparsity to both B and W will generate the following updating rules.

$$B = B \otimes \frac{\left(\frac{V}{BW}\right)W^T}{1W^T + \beta}$$

$$W = W \otimes \frac{B^T\left(\frac{V}{BW}\right)}{B^T 1 + \alpha}$$

Create a new function

```
function [B,W,obj,k] = ssnmf(V,rank,max_iter,lambda,alpha,beta)
% NMF - Non-negative matrix factorization
% [W,H,OBJ,NUM_ITER] = SSNMF(V,RANK,MAX_ITER,LAMBDA)
% V - Input data.
% RANK - Rank size.
% MAX_ITER - Maximum number of iterations (default 50).
% LAMBDA - Convergence step size (default 0.0001).
% ALPHA - Sparse coefficient for W.
% BETA - Sparse coefficient for B.
% W - Set of basis images.
% H - Set of basis coefficients.
% OBJ - Objective function output.
% NUM_ITER - Number of iterations run.
```

The new objective function is

```
function [obj] = compute_objective(V,W,H,alpha,beta)
  obj = sum(sum(V.*log(W*H) - (W*H))) + alpha*sum(sum(H))+ beta*sum(sum(W));
```

if you call the ssnmf function as fellow

```
[B,W,obj,k] = ssnmf(X,40,500,0.001,100,1);
```

you will obtain the following outputs.