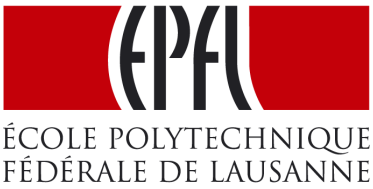


ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Laboratoire d'Algorithmique

Semester Project

8 ECTS

Implementing Some Feature Extracting Techniques to Model Human Visual System

Author:
Diego Marcos

Supervisor:
Amir Hesam Salavati
Professor:
Amin Shokrollahi

June 7, 2013

Proposed project description

In computer vision, there are various different techniques to extract important features from images. These features are then later used in patten recognition, image classification, etc. Some of these techniques are comparable to some models of specific parts in human visual system.

In this project, we are interested in implementing some of the widely used techniques in feature learning (extraction) and applying them to a dataset of natural images. This usually corresponds to solving some optimization problem to find the features that represent the data more accurately.

The implementation can be either done in C or MATLAB (MATLAB is preferred).

And here are some lines to give you an idea about why we are interested in this project: Once the feature extraction techniques are implemented, the learned features will then be used as inputs to a neural network which mimics some parts of human memory (neural associative memory). The ultimate goal would be to see if one will get better information storage capacities in artificial neural memories when the inputs are natural stimuli (such as images) and pre-processed before being stored. Here, pre-processing refers to the feature extraction procedure.

Chapter 1

Introduction

1.1 Feature extraction

In many real-life applications in general, and machine learning in particular, we have algorithms that have to deal with high dimensional data. As in other domains, this "curse of dimensionality" often slows down the algorithms and/or prevents them to produce meaningful results.

One way to overcome this problem is to reduce the dimension of the input data by means of *feature extraction*. In general, feature extraction refers to the process of eliminating redundant information in the input data and only keep those essential "features" that help our algorithms to achieve their objectives.

Feature extraction algorithms can be divided on two categories, depending on how they look for features. On one hand, we have the algorithms that look for "pre-defined" features in the data. Oriented edge-detection in images by means of Gabor-like filters is an example of such algorithms. On the other hand, there are algorithms in which the features are not pre-defined and they are required to learn what type of features to extract. We only fix the objective of the feature extraction process, i.e. reconstruction quality or classification success rate, and ask the algorithm to extract those features that achieve this objective.

In the case of natural images, the high redundancy usually allows to find a set of features with which we can express a good aproximation of the image using a much lower amount of information than than we would by simply writting down the values of all the pixels.

Chapter 2

Objectives of the project

In this work, our goal is to implement a few feature extraction techniques. A feature extraction approach comprises two stages: "learning what features to extract" and "extracting the features from the data". We refer to the first phase as "learning" and the later as "encoding" for brevity.

According to [1] the method chosen for the encoding phase seems to have more impact in the overall performance of feature extraction algorithms applied to image classification. For this reason, in this project we are interested in the second type of feature extraction algorithms. For the learning phase we will use random patches from the data set to fill in the dictionary of features.

We focus on implementing some well-known feature extraction algorithms for visual tasks. Some of these algorithms are known to have similarities to the algorithms used in mammalian visual pathway [4]. Thus, their outputs seem to be an excellent candidate to test the neural associative memory with exponential capacity proposed in [2].

As a result, we design a package that on one hand can be used as a separate feature extraction box (with the choice of different algorithms) that ensures the reconstruction quality of the images. On the other hand, it can be used as a middle processing stage in the model proposed in [2] which prepares the input data (images) for the associative memory that can achieve large retrieval capacities on the real image datasets.

For the purpose of this report though, we focus on the first task. We compare the different implemented feature extraction algorithms from different perspectives, i.e. reconstruction quality, running time, etc. We also propose some heuristics that achieve better performance compared to the implemented algorithms. We report the results for both these approaches in a later section.

Chapter 3

Methods

3.1 Overview

In this project, though the focus is on the encoding of small image patches, we have implemented all the steps required for image encoding and decoding. Overall this consists on the following steps:

1. **Image chopping:** The images are subdivided into smaller patches.
2. **Patch encoding:** Each patch is encoded as a vector of features.
3. **Patch decoding:** An approximation of the original patches is calculated from the feature vectors.
4. **Reconstruction:** The recovered patched are stiched together to reconstruct and approximation of the original image.

3.2 Learning Methods

In [1] the reaserchers arrive to the conclusion that the choice of the encoding method has much more impact in the performance of the algorithm than the choice of the training method. In particular, choosing patches from the dataset at random to fill the dictionary seemed to perform comparably to the much more computation hungry Sparse Coding. Therefore, for this project we will mostly consider Random Patches as the method for building the dictionary.

3.3 Encoding Methods

For this project, three different encoding algorithms will be considered: Orthogonal Matching Pursuit (OMP), Sparse Coding (SC) and Soft Threshold (T). Their performance has been compared in terms of benchmark image classification in Coates et al, 2011 [1]. We will compare the performance of each approach over standard image datasets in terms of image reconstruction quality.

3.3.1 Sparse coding (SC)

Each patch is encoded using the L1-penalized sparse coding formulation, which consist of finding. for an image x, the sparse code s that makes Ds as similar as possible to x, while maintaining s as sparse as possible, being D the chosen dictionary. This is to find s that minimizes the cost function:

$$C(s) = \|Ds - x\|_2^2 + \lambda\|s\|_1 \tag{3.1}$$

This minimization can be done with any off-the-shelf method for nonlinear minimization. In order to compute the value of s in MATLAB, we decided to use the Nonlinear Conjugate Gradient [5] method, which needs an explicit expression of the gradient of (3.1). Here we derive that expression with respect to the k^{th} component of s, s_k:

$$\frac{\partial C(s)}{\partial s_k} = \frac{\partial}{\partial s_k}\|Ds - x\|_2^2 + \frac{\partial}{\partial s_k}\lambda\|s\|_1 \tag{3.2}$$

The first term of 3.2 can be expressed as:

$$\frac{\partial}{\partial s_k}\|Ds - x\|_2^2 = \frac{\partial}{\partial s_k}\sum_{i=1}^{n}(D(i,:)s - x(i))^2 =$$

$$\sum_{i=1}^{n} 2(D(i,:)s - x(i))\frac{\partial}{\partial s_k}D(i,:)s = \sum_{i=1}^{n} 2(D(i,:)s - x(i))D(i,k) = \tag{3.3}$$

$$2\sum_{i=1}^{n}(D(i,k)D(i,:)s - D(i,k)x(i) = 2(D(:,k)^T Ds - D(:,k)x)$$

The second term of 3.3 can be approximated by:

$$\frac{\partial}{\partial s_k}\lambda\|s\|_1 = \lambda sign(s_k) \tag{3.4}$$

Thus, we get that the expression for the gradient of $C(s)$ is:

$$\vec{\nabla}C(s) = 2(D^T Ds - Dx) + \lambda\vec{sign}(s) \tag{3.5}$$

where D^T is the transpose of D.

As a sanity check, we have tested this method with $\lambda = 0$, where the method converges to the original image. Furthermore, increasing λ results in sparser solutions.

With this, we can use the Nonlinear Conjugate Gradient method as follows:

begin
  $s^0 = \vec{0}$
  $\Delta s^0 = -\vec{\nabla}C(s^0)$
  $\alpha^0 := \text{argmin}[C(s^0 + \alpha\Delta s^0)]$
  $\delta^0 := \overset{\alpha}{\Delta s^0}$
  $s^1 = s^0 + \alpha^0\delta^0$
  while not converged & maxiter not reached do
    $\Delta s^n = -\vec{\nabla}C(s^n)$

$\beta^n = \frac{\Delta s^{n\,T} \Delta s^n}{\Delta s^{n-1\,T} \Delta s^{n-1}}$
$\delta^n = \Delta s^n + \beta^n \delta^{n-1}$
$\alpha^n = \underset{\alpha}{\text{argmin}}[C(s^n + \alpha\delta^n)]$
$s^n = s^{n-1} + \alpha^n \delta^n$
od
end

Although a higher value of λ will certainly bring most values of the solution near to zero, almost none of them will become exactly zero. To ensure sparsity, we need to remove the values that are near enough to zero. This was implemented by adding an extra parameter, the dumping factor, which represents the maximum admissible loss of L_2-norm due to removing the smaller values of the solution.

3.3.2 Orthogonal Matching Pursuit (OMP)

Though sparse coding has the advantage of being able to find the optimal way of expressing an image x in terms of a given dictionary D, when D is big, the computational cost might become unacceptable. OMP, being a greedy algorithm, provides a sub-optimal solution at a much smaller computational cost. Matching Pursuit (MP) seeks at each iteration for the element in D, $D(k)$, that best matches the residual $x - Ds$ and adds $D(k)^T x$ to $s(k)$. OMP is a variant of MP in which, at each itaration, the elements of s already selected are updated by orthogonally projecting x onto them.

begin
$r^0 = x$
$z^0 = 0$
$\Gamma^0 = \emptyset$
for $n = 1$ to maxiter do
$\alpha = D^T r^{n-1}$
$i_{max} = \underset{i}{\text{argmax}}|\alpha_i|$
$\Gamma^n = \Gamma^{n-1} \cup i_{max}$
Factorize: $Q_{\Gamma^n} R_{\Gamma^n} = D_{\Gamma^n}$
$q = Q_{\Gamma^n}(:, n)$
$z^n = q^T x$
$z_{\Gamma^n}(n) = z^n$
$r^n = r^{n-1} - z^n q$
od
$s_{\Gamma^n} = R^{-1}_{\Gamma^n} z_{\Gamma^n}$
end

Where Γ^n is the set of indeces of the selected dictionary elements.

3.3.3 Soft Threshold (S)

This method simply selects the features in D whose dot product with the image x is greater than a certain threshold α:

$$s = max(0, D^T x - \alpha) \tag{3.6}$$

3.4 Chopping Methods

The above encoding algorithms are usually applied to smaller patches extracted from each image rather than the whole image itself. This approach speeds the feature extraction process and make parallelization of the whole process easier. Furthermore, it is closer to biological mechanisms where the receptive field for each part of the visual processing network is limited to a small part of the whole visible scene.

For this reason, we first "chop" each image into smaller patches, apply the encoding methods to each patch, and re-assemble the whole image from encoded patches for reconstructing the image.

However, there are different ways for chopping the image. The most obvious choice is to extract non-overlapping rectangular patches from the image by moving a small window over the whole image, which is one of the approaches we will use to chop images. This will be refered as Grid method.

Nevertheless, after visually analysing the image reconstruction results using OMP with very high sparsness settings, I found that the most noticeable artifact was the regular grid formed by the edges of the patches. As a possible way to reduce this effect came the idea of extracting the patches, not based on their position on a regular grid, but based on some measure that represent how "important" the content of the patch is.

To choose the proper measure, it is interesting to note that the human visual system relays heavily on local contrast, or the intensity diffrence between a point and adjacent points [3]. In terms of digital filtering, this behavior can be captured by using a Laplacian-of-Guassian filter:

$$LoG(x,y) = -\frac{1}{\pi\sigma^4}[1 - \frac{x^2+y^2}{2\sigma^2}e^{-\frac{x^2+y^2}{2\sigma^2}}] \tag{3.7}$$

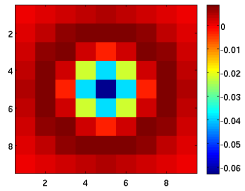


Figure 3.1: 9x9 Laplacian of Gaussian filter with $\sigma = 1.5$.

In order to capture features of different sizes, we apply three Laplacian-of-Gaussian filters to the image and use the result to choose the positions of the centers of the patches to be extracted.

Additionally, we will use the same measure to extract random patches for learning the dictionary. This way, the elements of the dictionary look more similar to the important elements in the patches we would like to encode and, hence, potentially reducing the number of non-zero elements in the encoded patch while maintaing the reconstruction quality.

3.5 Measuring reconstruction quality

After having said that human perception gives a lot of importance to local contrast, it would be incoherent to simply use some L_2-norm based quantity, which would penalize high and low contrast areas equally. Therefore I decided to measure both the normal SNR of the reconstruction as well as the SNR of the Laplacian of the reconstructed image against the Laplacian of the original image, since it represents the fidelity of the reconstruction of the variations in the image.

In order to measure the reconstruction of each method we use 2 different numbers: the Signal to Noise Ratio (SNR) between the reconstructed and the original images, and also the SNR between the Laplacian of the images (SNRL). We do this by simply applying a Laplacian filter to both the original and the reconstructed images and then computing the SNR between them.

On the other hand, we assume that there will be a trade-off between quality and sparseness. For measuring the latter, we use the compression ratio, defined as the ratio between the number of non-zeros in a feature vector, the compressed image, and the number of pixels in the original image.

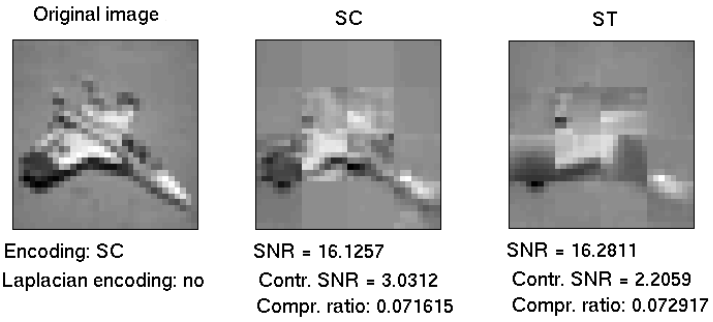


Figure 3.2: Example of an apparently better reconstruction, in the center, that get a worse SNR than an apparently worse one. The SNRL seems to represent better the subjective visual quality.

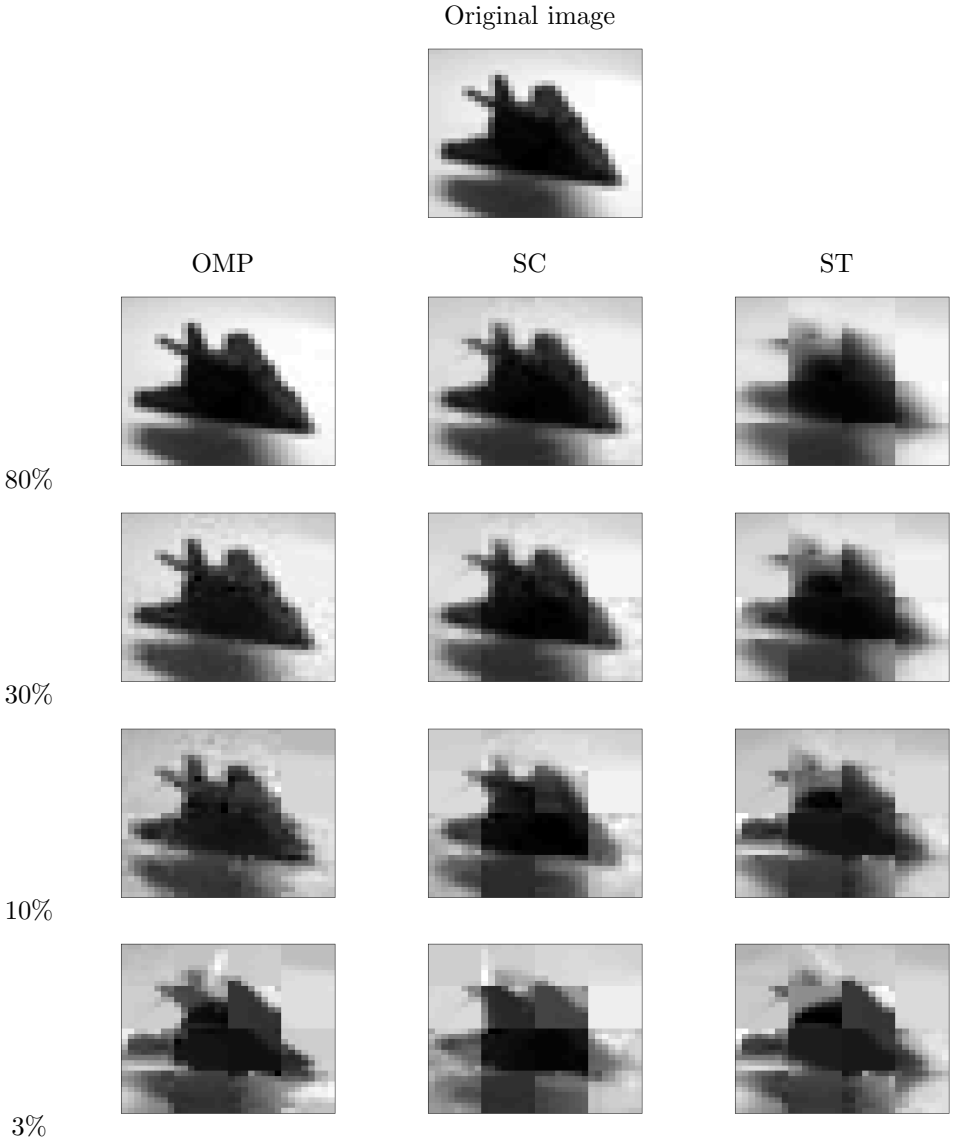
Chapter 4

Results

In this section, we evaluate the performance of the implemented encoding methods based on their reconstruction quality, compression ratio and running time. We also investigate the effect of proposed chopping method and modified learning algorithm on the performance of the encoding methods.

4.1 Performance of the Encoding Methods

Reconstructed images after having been encoded with the 3 methods, for different levels of sparseness:



Comparing the methods for a wide range of compression ratios in priciple requires to explore the whole parameter space to find the optimal values. The ST method has only one parameter to vary the sparseness of the solution, α. Varying α we can easily map the relationship between SNR and α to SNR versus compression ratio. OMP appears to have 2 parameters, but the maximum number of iterations should be set to the number of pixels per patch, leaving only the tolerance as parameter, in order to get the best possible solution. For SC, we keep a low enough value for the tolerance, i.e. when lowering it doesn't improve the reconstruction quality, and explore the 2D parametric space formed by λ and the dumping factor. Then we can take the best couple for each compression.

In Fig. 4.3 we see how OMP is a clear winner if we measure the quality in terms of SNR. But for high sparseness, SC does a better job in terms of SNRL. A smaller dictionary size increases the range in which SC performs better than OMP.

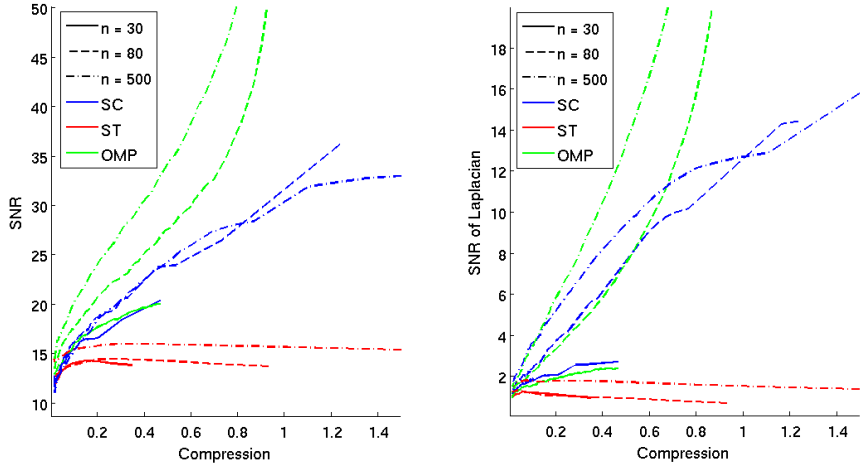


Figure 4.1: Achievable quality for any compression ratio for the 3 methods and 3 dictionary sizes. On the left, quality in terms of SNR, on the right in terms of SNR of the Laplacian. Test set of 800 8x8 test patches.

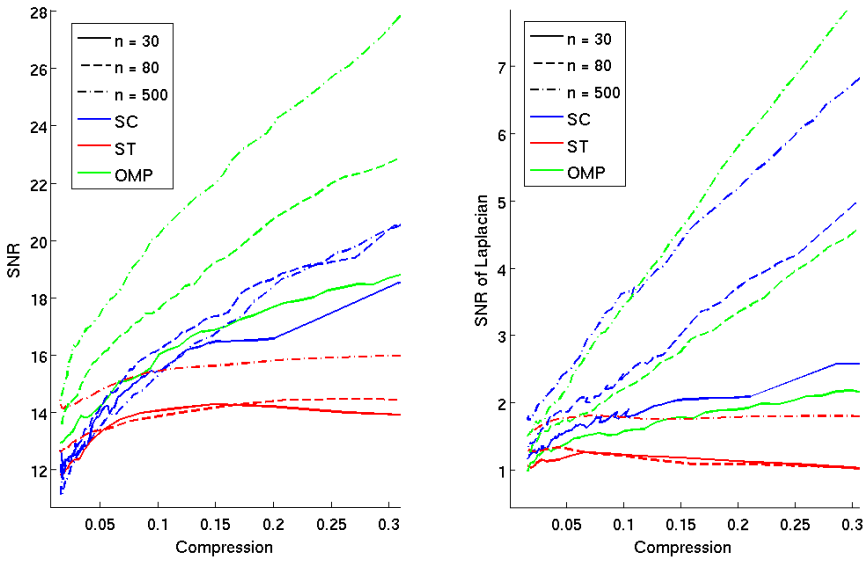


Figure 4.2: Detail of the previous figure.

It's quite surprising to notice that, using the SNR as measure, ST seems to do better than SC for compression under 10%, while measuring SNRL brings SC always above ST. Let us visually check what is happening in that area: in Figure (3.2) we can see an example that reproduces this trend. For a sparseness level of 7% SC seems to have captured more details than ST. This difference in subjective quality seems to be better captured by the SNRL than by the SNR,

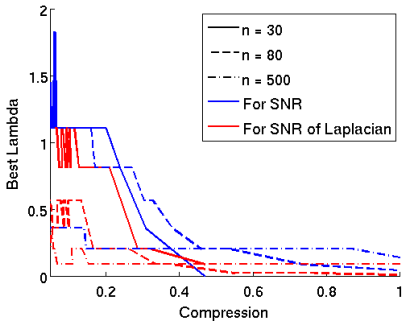


Figure 4.3: Optimal values of Lambda for every compression ratio.

which is actually lower for SC.

According to [1], SC is in general the best choice for an encoding algorithm (among those considered in the publication) in terms of image clasification performance.

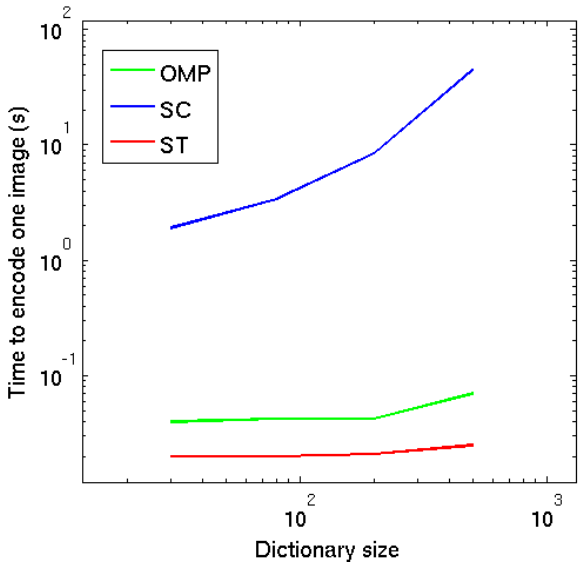


Figure 4.4: Computing time required by the 3 algorithms for different dictionary sizes N. ST and OMP seem to almost not depend on N, while SC grows exponentially. In seconds per image.

4.2 Boosting the SNR of the Laplacian

As they have been implemented, the 3 methods are oriented to either minimize the difference of the norm between the original and the encoded images (SC) or maximize the coincidence between the both (SC and ST). Since our results

suggest that SNRL could be a better indicator of visual quality of the reconstruction, we tried two different techniques aimed at improving this measure:

- **Feature based chopping:** Selecting the positions of the patches based on some interesting feature, like a Laplacian or a LoG filter.
- **Convolve before encoding:** Applying a Laplacian filter to the image prior to the encoding. Then deconvolving the image once reconstructed.

4.2.1 Feature based chopping

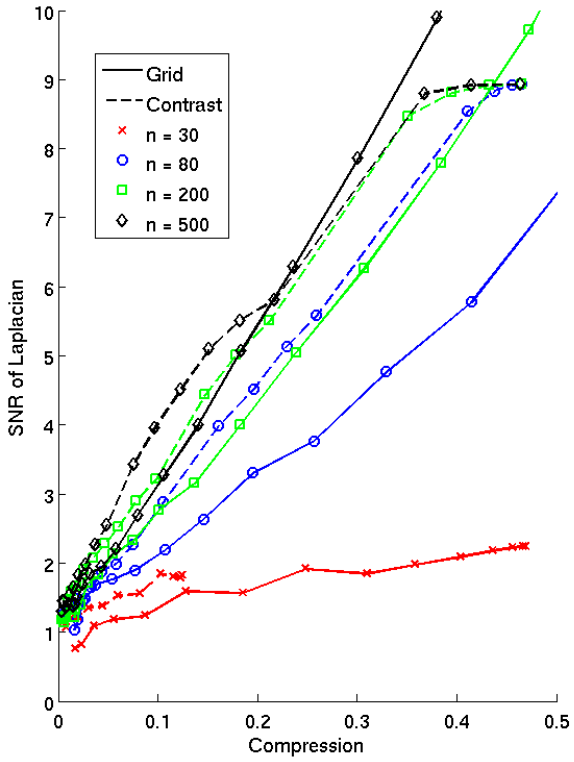


Figure 4.5: Difference in reconstruction quality using the two chopping methods: regular grid and contrast based for various dictionary sizes n.

A LoG filter is applied to the image to get a map of interesting areas. The patches are then greedily extracted, centered in the highest peaks of the map. In figure 4.5 we see how the SNRL does indeed improve substantially for high sparseness.

4.2.2 Convolve before encoding

A feature of interest, in this case a Laplacian filter, is convolved with the image, performing thus a first feature extraction step. The resulting map of features,

of the same size as the original image is then encoded as usual. After the reconstruction a deconvolution step has to be performed with a Laplacian filter before comparing the result to the original image.

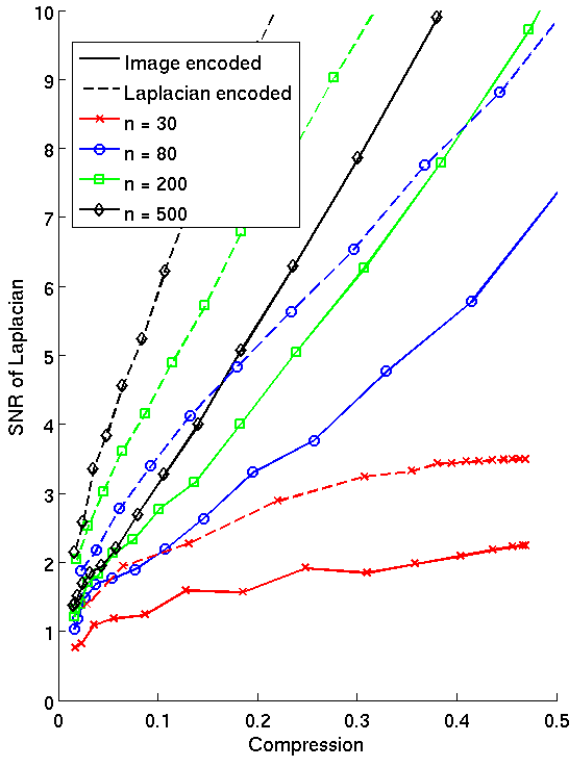


Figure 4.6: Difference in reconstruction quality by storing the image as is or the Laplacian of the image for various dictionary sizes n.

In figure 4.6 we see again that the SNRL has improved with respect to the standard method.

If we combine both techniques, figure 4.7, we get an extra improvement as compared to applying the two techniques separately.

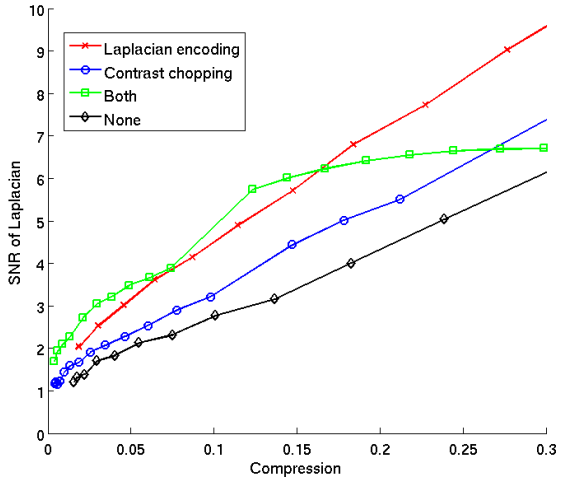


Figure 4.7: Difference in reconstruction quality by applying each one of the presented methods (i.e. storing patches of the Laplacian of the image and choosing the position of the patches based on features), both together or none (i.e. regular grid for chopping the patches and directly storing patches of the image, without performing a Laplacian operation first).

4.3 GUI

In order to make it easier to understand the effect of using the different algorithms and parameters on the sparsness and the reconstruction quality, as well the use of the implemented algorithms for feeding the NN algorithms mentioned in the projects objectives, it was decided to write a Graphic User Interface that allows to load a dataset, build a dictionary, encode a subset of the dataset, export the encoded images and show the reconstructed images along with information on the quality measures and sparsness.

- **Import:** The path to the .mat file containing the variable with the dataset has to be entered or browsed using the Browse button. The dataset has to be a matrix whose rows are the row-major vectorized images. Their lenght has to be a power of 2. The .mat file can contain several variables. The user will be asked to choose one after clicking the Create dictionary button.
- **Dictionary creation:**
  - *Method:* How the random patches are taken from the images: grid or contrast based.
  - *Samples:* Size of the random dictionary.
  - *Redundacy reduce to:* Keep only this number of elements in the random dictionary, by removing those that are very similar to others and those that are not similar to any other patch.

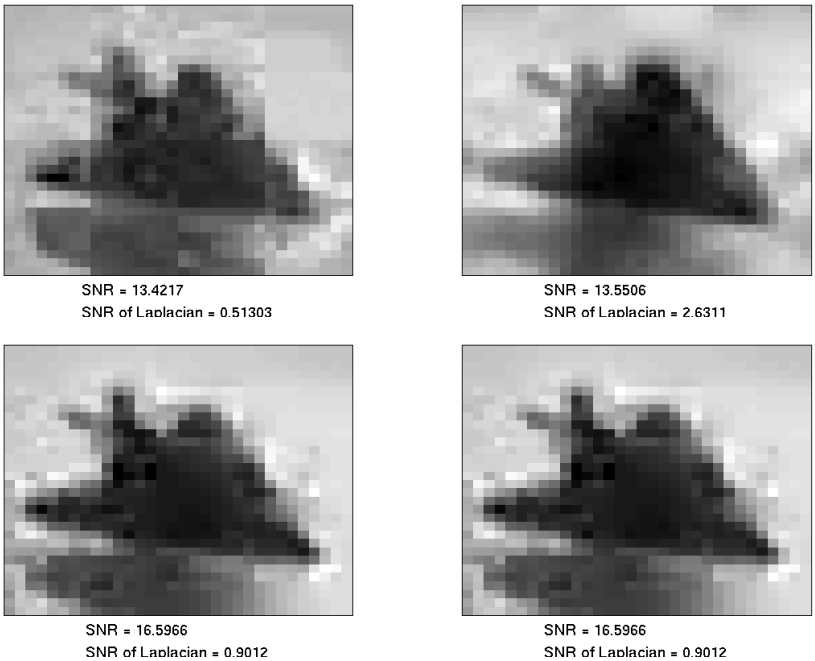


Figure 4.8: Standard method (up left), grid chopping after applying a Laplacian filter (up right), contrast based chopping (bottom left), contrast based chopping after applying a Laplacian filter (bottom right). All for sparseness of 7%.

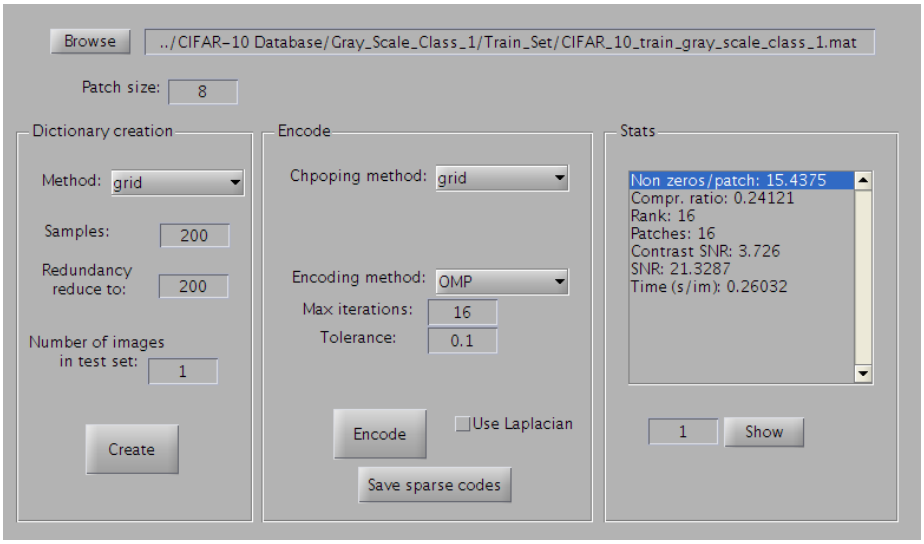


Figure 4.9: Main window of the GUI.

– *Number of images in test set:* The number of images that will be fed to the encoding algorithms. No patch from this images will be used for the dictionary.

- *Create:* Build the dictionary with the current settings. If several variables are found in the specified .mat file, a pop-up will appear to let the user choose one.

- **Encode:**
  - *Chopping method:* Method for choosing the positions of the patches in which the image is to be divided. Grid or contrast based. The following parameters are available if Contrast is selected:
    - *Decay size:* A large value means that the selected patches will be further apart. Must be greater than 0.
    - *Threshold:* A large value means that fewer patches will be selected. $0 \leq$ Values < 1.
  - *Encoding method:*
    - *OMP:* Orthogonal Matching pursuit:
      - *Max iteration:* Maximum number of non-zeros allowed in one feature vector. Value $\leq$ number of pixels per patch.
      - *Tolerance:* A large value means higher sparseness. $0 \leq$ Values < 1.
    - *ST:* Soft Threshold:
      - *Alpha:* A large value means higher sparseness.
    - *SC:* Sparse Coding:
      - *Lambda:* Penalty factor. A higher value increases sparsity. $0 <$ Value.
      - *Tolerance:* Controls convergence of the Conjugate Gradient method. Reduce it's value until it stops affecting the reconstruction quality. Try 1e-4 to 1e-3.
      - *Dumping factor:* A large value means higher sparsity. Usually it helps increasing Lambda is this is increased. $0 \leq$ Value < 1.
    - *Encode:* Performs the encoding over the test set with the selected parameters.
    - *Use Laplacian:* Choose whether to encode the Laplacian of the image instead of the image itself.
    - *Save Sparse codes:* Opens a file selector to save a .mat file with a variable containing the sparse features encoding the patches.
  - **Stats**
    - *Show:* Shows the selected test image and the reconstructed version.
    - *Compr. ratio:* Compression ratio.
    - *Rank:* Rank of the matrix of features.
    - *Patches:* Number of feature vectors.
    - *Contrast SNR:* SNR between the Laplacian of the reconstructed image and the Laplacian of the original one.
    - *SNR:* Signal to Noise Ratio between the reconstructed image and the original one.
    - *Time:* Average time required to encode one image, in seconds.

4.4 Conclusion

Of the implemented encoding methods OMP consistently provides the best SNR between the original and the reconstructed image with a computing time only twice the one required by ST. On the other hand, SC performs better when encoding with high sparseness when the SNR of the Laplacian of the image is used a a measure of the quality. Visual inspection suggests that SNRL could be better estimator of the visual quality of the reconstruction. The results attained by doing a first feature extraction step before the encoding, by means of choosing the patches' positions based on the feature or directly encoding the image after having been convolved with the feature, show that such techniques can substantially improve the SNRL.

References

[1] Andrew Y. Ng Adam Coates. The importance of encoding versus training with sparse coding and vector quantization. *ICML*, 2011.

[2] Amir Hesam; Shokrollahi Amin Karbasi, Amin; Salavati. Iterative learning and denoising in convolutional neural associative memories. *ICML*, 2013.

[3] Mike Swanston Nicholas Wade, Michael T. Swanston. *Visual Perception: An Introduction*. Psycology Press, 1991.

[4] Friedrich T. Rehn, Martin; Sommer. A network that uses few active neurones to code visual input predicts the diverse shapes of cortical receptive fields. *Journal of Computational Neuroscience*, 2011.

[5] Jonathan Richard Shewchuk. *An Introduction to the Conjugate Gradient Method Without the Agonizing Pain*. 1994.