Laboratório VISGRAF
Instituto de Matemática Pura e Aplicada

Super-Resolution

Regis Destobbeleire
Luiz Velho (Supervisor)

Technical Report TR-02-08 Relatório Técnico

June - 2002 - Junho

The contents of this report are the sole responsibility of the authors.
O conteúdo do presente relatório é de única responsabilidade dos autores.

Super-Resolution

Régis DESTOBBELEIRE
Regis.Destobbeleire@polytechnique.org

Training period at the IMPA
supervised by L. Velho (IMPA, Brazil) and S. Mallat (École polytechnique, France)

May - June 2002

Résumé

La netteté d'une image est un paramètre important du traitement du signal. En effet, l'interprétation d'une image est conditionnée par la possibilité d'en extraire l'information qu'elle contient. Plus une image est nette, et plus on pourra y déceler de détails. Si elle est floue, son interprétation sera difficile. Nous sommes donc limités par la résolution de l'image : si nous voulons zoomer au-delà de cette résolution, l'interpolation nécessaire produira une image floue. Il est en fait impossible de créer l'information manquante correspondant aux détails absents de l'image de départ, mais que l'on voudrait voir apparaître après un zoom.

Cependant il est envisageable de deviner cette information manquante. Cela permettrait d'augmenter la résolution d'origine d'une image : c'est la super-résolution. Je me suis intéressé à deux approches différentes :

La première, développée par William T. Freeman, est d'utiliser un ensemble de référence, composé d'images nettes. Celui-ci va permettre d'apprendre la relation entre les hautes et les basses fréquences d'une image, et donc de deviner les hautes fréquences manquantes d'une image floue.

L'autre, développée par Aaron Hertzmann, utilise comme référence une paire d'images (les versions nette et floue de la même scène) pour deviner, par analogie, la version nette d'une autre scène.

J'ai implémenté un algorithme décrit par W.T. Freeman en C++, ainsi qu'une interface graphique, qui permet d'afficher les images et de contrôler certains paramètres.

Abstract

Sharpness is an important parameter in signal processing. Indeed, interpreting an image is determined by the possibility of extracting the information it contains. The sharper an image is, the more details it shows. If it is blurred, it will be hard to interpret it. Thus we are limited by the resolution of the image : if we want to zoom beyond this resolution, the necessary interpolation will generate a blurred image. In fact, it is impossible to create the missing information, which corresponds to the details we cannot see in the original image, and we would like to see in the zoomed one.

However, it is conceivable to guess this missing data. This would allow to increase the original resolution of an image : this is called super-resolution. I have chosen to study two different approches :

The first one, which was developed by William T. Freeman, consists in using as a reference a set of sharp images. This set allows us to learn the relation between high- and low-frequency bands in an image, and then to guess the missing high-frequency band of a blurred image.

The other one, developed by Aaron Hertzmann, uses as a reference a couple of images (the sharp and blurred versions of the same scene) to create, by analogy, the sharp version of another scene.

I implemented an algorithm described by W. T. Freeman in C++, as an interface, which permits to display the images and to control some parameters.

(a) (b) (c)

Figure 1 : Same original image, zoomed in by a factor of 2 and 4 with bilinear interpolation (a), with cubic B-spline interpolation (b) and with super-resolution (c)

I – Introduction

Dealing with sharp images is a necessity for many image-processing applications. The first problem to face is to define what sharpness is. A sharp image could be defined as an image that *looks* like a real scene, a natural one. But the human feeling is not easy to compute, and we will admit that a sharp image has all its frequency bands. On the contrary, a blurred image is defined as an image that has lost the high-frequency information, which often prevents from its interpretation. But the acquisition of images - such as scanning a photography - tends to produce blurred images. If one takes a picture with a digital camera while moving, the picture will be blurred. When you zoom into an image, the necessary interpolation causes blur. Compression creates artefacts. Therefore sharpening images is an important element of image rendering.

One solution would be to use polygon-based representations of a scene. An image generated from a polygon-object has always sharp edges. Furthermore, it is resolution-independent : you can zoom at any range, the edges will remain sharp (you are only limited by the size of the polygon edges). But real scenes are often too complex to be modelled by polygon-objects.

Another solution would be to know how to sharpen an image. That would allow photo enhancement, and thus zooming without losing sharpness. But going beyond the original resolution of an image is creating information, which seems impossible. We can try to guess it. This is called *super-resolution*.

Super-resolution consists in creating a sharpened version from a given blurred image called *input*, using a database created beforehand, called *training set*. The result is called he *scene*.

Our images could then be resolution-independent even if they represent complex scenes. The resolution independence would not be complete, but it would allow an increase of their resolution, which is an important problem in image processing.

I.1 History

Analytical approaches have reached limited success. Interpolation, bilinear or cubic B-Spline, blurs the edges and deletes image details. High-frequency enhancement is not really robust, and needs a human operator to decide the level of enhancement. But some statistical approaches have had good results.

The robust and fast algorithm that would sharpen an image does not exist, though many people have tried to create it. In [8] and [9], Kersten and later, Hurlbert and Poggio used linear approaches, but they were weak in most cases. In [1] and later in [2], Freeman used a Bayesian propagation algorithm, which turned out to be more efficient. This algorithm uses a training data to derive the propagation parameters. In [2] he tries to use a one-pass algorithm, without using the slow Markov Network model he had previously used.

Similarly, Hertzmann, in [3], uses only a pair of images (the source image and the filtered image) as a training data. By analogy, he manages to transform an input image into a filtered image, which seems to be filtered in the same way as the training pair. By choosing a pair of blurred and sharp images as the training data, the Image Analogy program will sharpen the input image.

I.2 Overview

My study is based on Freeman's work in [2]. In this paper he explains how to guess the missing high frequency band in an image. He describes two algorithms. The first one uses the Markov Network model. It is efficient, but slow. Then he describes another algorithm, which is faster and gives good results too. It combines the advantages of efficiency and speed.

I have implemented the second one. The first release used a brute force matching process, and is not really fast. Thus I began to implement the TSVQ (Tree Structured Vector Quantification) to increase its speed. My goal is to compare the images my algorithm creates to the ones that the Image Analogies software can create.

I have applied super-resolution to the same images that Hertzmann sharpens by Image Analogy in [3], in order to compare my results to his.

II - Super-resolution method

The main idea of super-resolution is to use some images as references with the goal of learning how to sharpen an image.

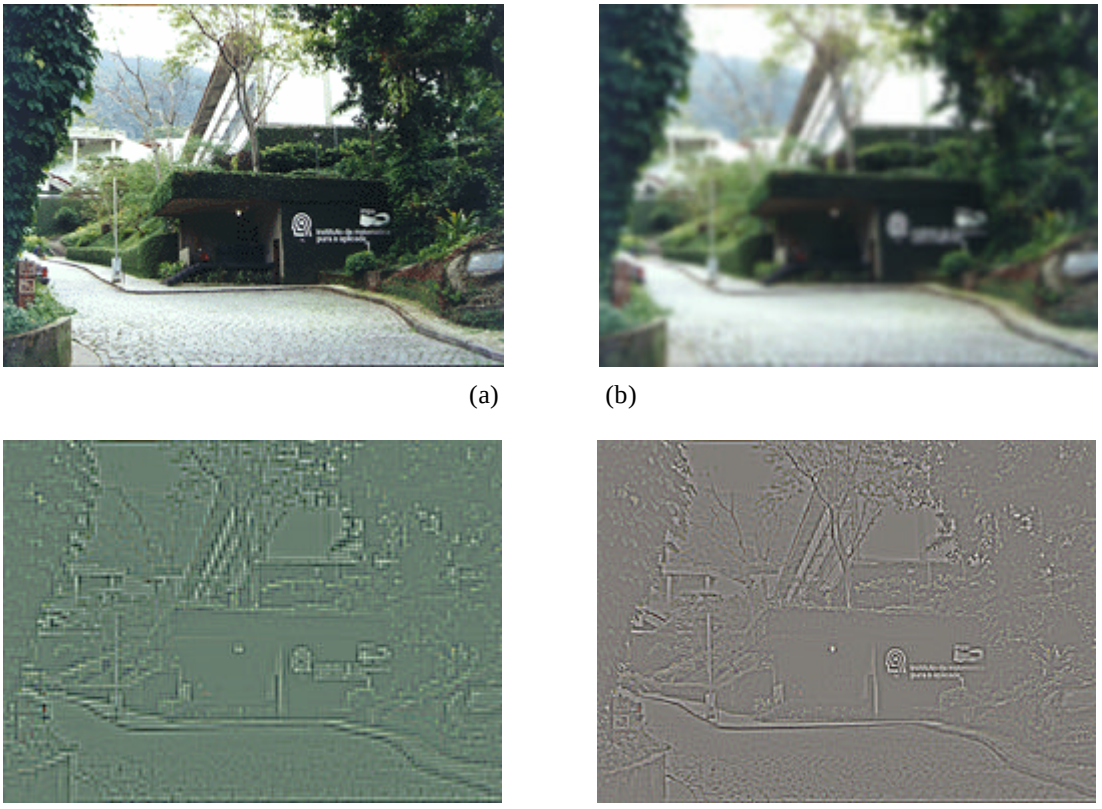
These references are called *Training Set*. They are sharp images, containing low-, mid- and high-frequency data. The *Input* image is the image you want to zoom in. First you have to scale it up (with a factor of 2), interpolating the missing pixels. You then obtain a bigger image, but the high frequency data is missing.

One could fear a possible need to use a huge Training Set to be able to reach the diversity, the richness of natural images. Computing it, if possible, would be dreadfully long. Thus we will assume some properties of natural images, to increase the efficiency of a reasonable-sized Training Set.

First, assuming we are working with three frequency bands (using a pyramid image decomposition), we say that the highest frequencies are conditionally independent from the lowest frequency band. We can write : $P(H \mid M, L) = P(H \mid M)$, where L, M and H are the low, mid and high frequency bands.

That allows to work only with two bands, and thus not to consider the diversity of the lowest band.

Then, we assume that the relationship between the high and mid bands are independent of the local image-contrast. Thus, by normalising the contrast in each image, we reduce their variability, and increase the efficiency of the Training Set.



(a) (b)

(c) (d)

Figure 2 : Original image (a), and his low-frequency band (b), mid-frequency band (c) and high-frequency band (d). Note that (c) and (d) do not have values between 0 and 255, and they have to be normalised to be displayed.

Super-resolution can be seen as two main independent steps : the first one consists in preparing the *Training Set* in a way that will permit the second one to construct the high frequency band that is missing in the scaled up *input* image.

II.1 The Training Set generation

This step is the deciding factor in the quality of the results, because it creates the relationship between mid- and high-frequency bands.

We consider local neighbourhoods, or patches, in each band. Each low-resolution patch is associated to a high-resolution one, centered on the same pixel, but not necessarily with the same size. But the local patch alone does not contain enough data to estimate plausible high-resolution detail : for one input patch, we can select the handful of the closest low-resolution patches from the Training Set. Freeman has shown that the corresponding high-resolution patches are very different. And then, choosing the nearest neighboor low-resolution patch to build the high-frequency band of an input image would lead to a very bad estimation of the real high-frequency band.

Spatial neighbouring effects have to be taken into account. For that purpose, in [2], Freeman describes two different algorithms : the Markov Network algorithm, and the single-pass one.

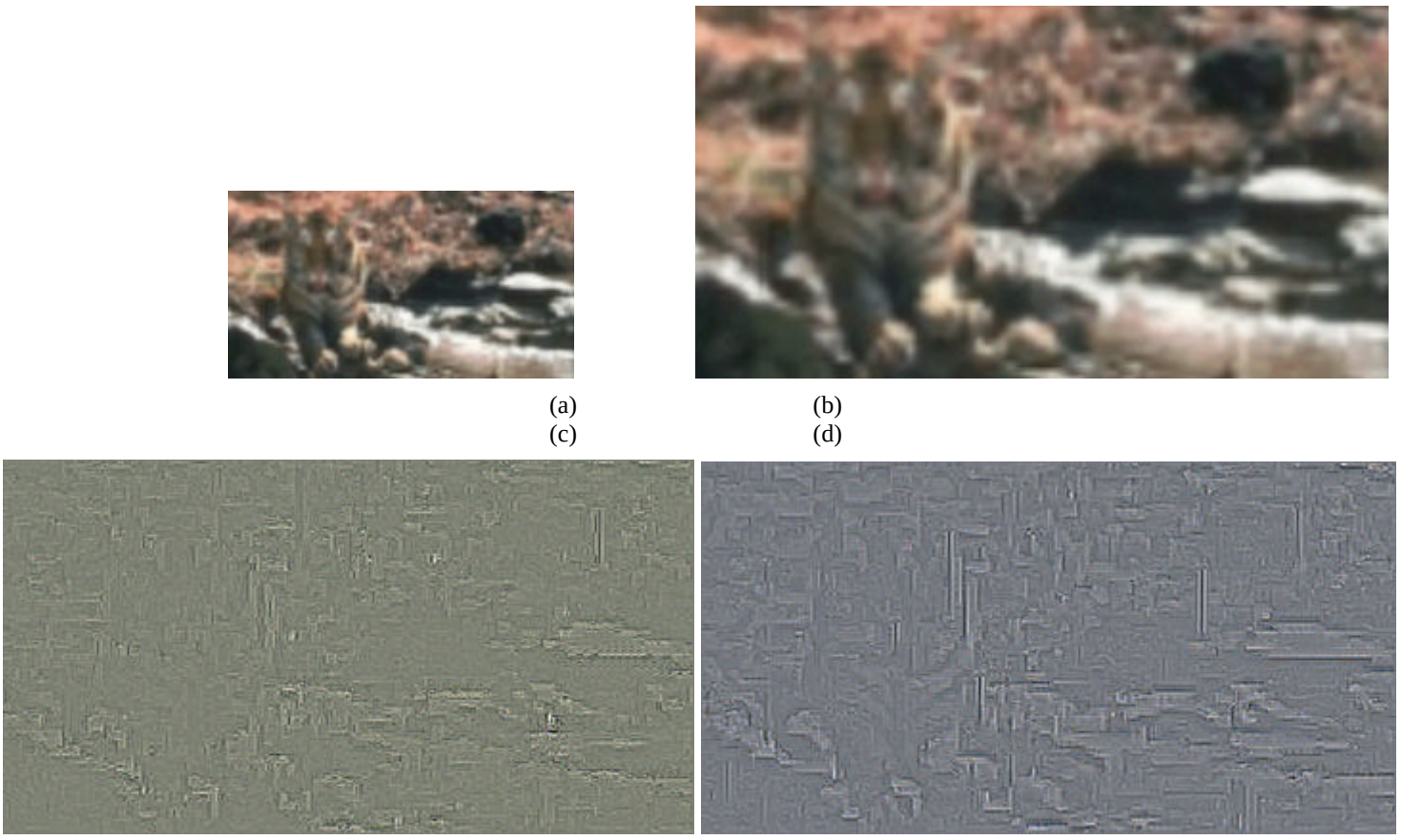


Figure 3 : The original image (a), up sampled (b), and the high-frequency bands of results of super-resolution, using a nearest-neighbour search (a), taking the spatial effects into account (b). Even if the result is not perfect, the overlap helps creating a better high-frequency band.

II.1.a The Markov Network algorithm

Freeman first modelled the spatial relationship using a Markov network. For given input image *y* (after been pre-processed), we seek to estimate the underlying scene *x*. The image *y* is made of observation nodes (the low-resolution patches), which have an underlying explanation, the high-resolution patches. In figure 4, the lines indicate statistical dependencies between the nodes. The Training Set is used to compute the probability matrices Ψ (representing the horizontal relation between high-resolution nodes), and Φ (representing the vertical relation between high-resolution and low-resolution nodes).

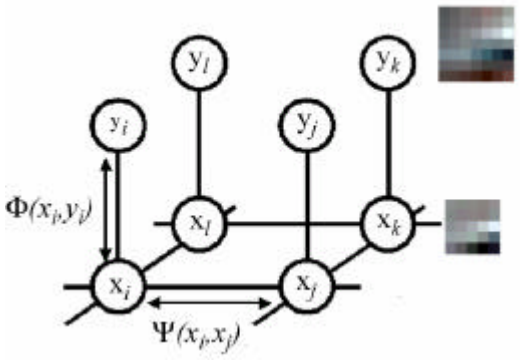


Figure 4 : Markov network model applied to super-resolution.

The optimal high-resolution patch is the one that maximizes the probability of the Markov network. An interactive algorithm called Belief Propagation has been used to compute the high-resolution image.

However Freeman decided to try a one-pass algorithm since only 3 or 4 iterations were enough to get a plausible high-frequency band.

II.1.b The single-pass algorithm

Its general purpose is easy to understand : the single-pass algorithm generates the missing high-band of a zoomed image, sequentially. The image, once pre-processed, is broken into patches. Then it is scanned in a raster-scan order, predicting at every step the high-resolution patch. The high-resolution patches are then added to the previous image, to get a sharp image, containing all the frequency bands.

The prediction of the high-resolution patch is the important part. We have two constraints.

- First, we have a frequency constraint : the high-resolution patch must be linked in the Training Set to a low-resolution one which has to be close (using the Euclidian distance) to the input low-resolution patch.
- The second constraint is spatial : we want to have a continuity in the built high frequencies, which means that the new high-resolution patch must match to the previously chosen ones.

The first requirement is easy to fulfil, by matching the low-resolution patch of the image we are zooming in and finding a match in the Training Set, made of pairs of high- and low-resolution patches. For the second criterion, the trick is to overlap the high-resolution patches

in the created image (see Figure 5). We add this information, and use it to select the good pair while searching in the Training Set.. To control the relative importance of these constraints, a parameter a is used. It is set to $a = 0{,}5$ in my computation.

The Training Set is composed of a searching vector and a high resolution patch, as described in Figure 6. The searching vector contains the low-resolution patch and the overlapped part of the high-resolution patch.

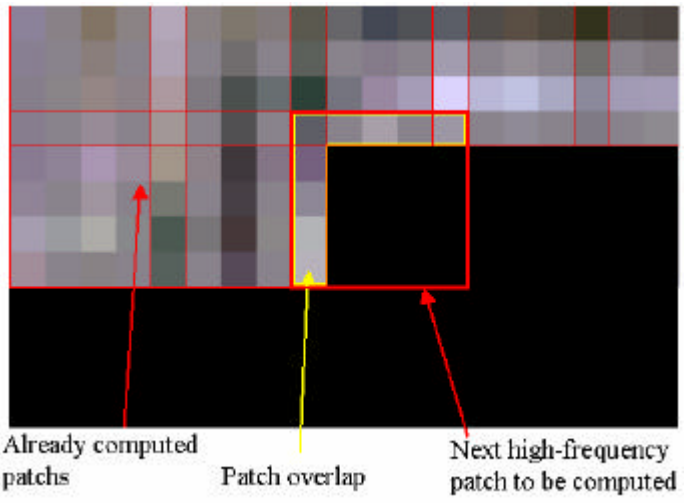


Figure 5 : Patch overlap in the high-frequency band.

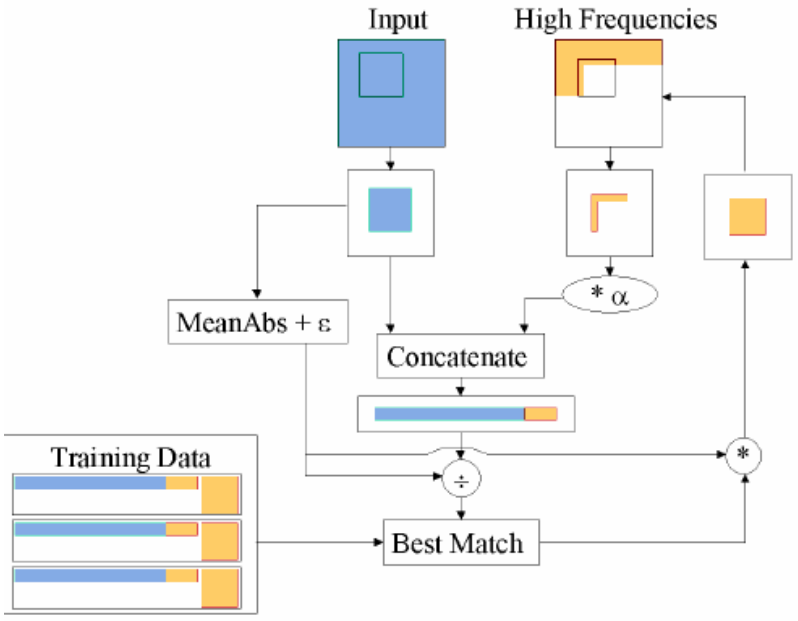


Figure 6 : Block-diagram showing raster-order per-patch processing. Low-frequency details are in blue, high-frequency ones in orange.

II.1.c Pre-processing functionsfor the single-pass algorithm

The images that compose the Training Set have to be processed before all to take the two assumptions described above into account.

By blurring them, scaling them down, and finally scaling them back up, using an interpolation, we create a low-passed image of the original version. The difference constitutes the high frequency band. Then we remove the lowest frequencies of the interpolated image, using another low pass filter, in in order to get the mid-band. The interpolated image is a degraded version of a high resolution one that has been degraded in the way the interpolated *input* image is assumed to be degraded.

Once this done, we break the built images into patches. We get high-resolution and low-resolution patches. Their size has been set to 5x5 for the former ones, and 7x7 for the later ones. The Training Set is constituted by these pairs of patches.

Then we have to normalise the contrast. I have carried out several experiments.
The first one is to linearly normalise the values of each band to get values between 0 and 255.
Another possibility is to divide the bands by an *energy image*, which describes the average absolute value of the image in each point.

But finally I used a local contrast normalisation : instead of normalising the whole image, I normalised each patch, dividing the high- and low-resolution patches by a local energy. I undid the normalisation after the matching process, to get the right (un-normalised) high-frequency patch.

The parameters of these pre-processing functions are described in the appendix.

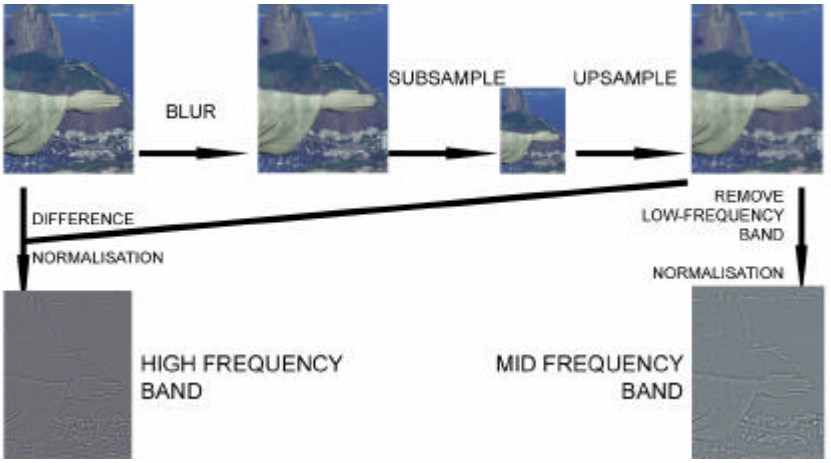


Figure 7 : Pre-processing steps.

II.2 Matching the patches

The matching process uses the Euclidian norm, modified by the a parameter. We can write a search vector form the Training Set $V = (V_l, V_h)$, where V_l is the vector containing the low-resolution data and V_h contains the high-resolution data (the overlap). Then we have :

$$d(V,W)^2 = |V_l - W_l|^2 + a|V_h - W_h|^2$$

II.2.a The brute force

I first have computed a brute force matching function. Given the input search vector, it covers the whole Training Set vectors to find the best match, and return the high-resolution patch.

But the number of patches in the Training Set and the dimension of the search vector make this matching process too slow. We usually work with 100000 to 500000 Training Set patches, and the dimension of the search vector is 174 (with RGB images). Thus I tried to compute a faster matching algorithm.

II.2.b The TSVQ acceleration

We consider patches as vectors ; actually my searching vector is constituted by the low-resolution patch and the overlap part of the high-resolution patch, and their dimension is 174. Quantification refers to the fact of using, a smaller set of searching vectors instead of the Training Set vectors. Finding the nearest neighbour of an input searching vector is easier and faster. This smaller set is called *codebook*, and is composed of *code words*.

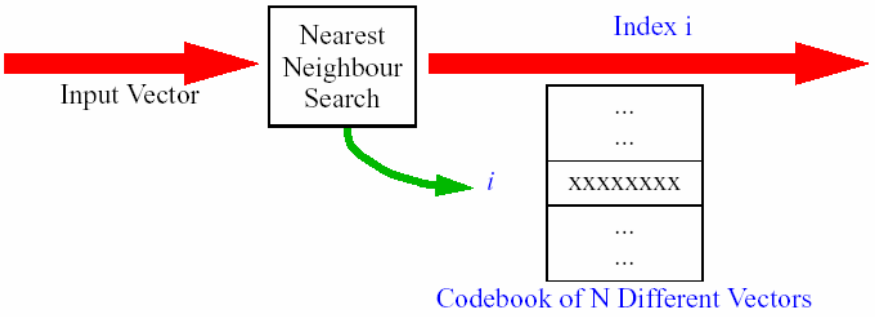


Figure 8 : principle of Vector Quantification.

The codebook is computed to have vectors as close as the Training Set vectors. It has a balanced tree-structure, so that the decoding (i.e. the matching process) only consists in "reading" the tree in depth, just deciding at each level to choose th e left or the right leaf of your node. The code words leaves have to minimize the distortion measured with the distance already defined with the training vectors. The first step is to compute the centroid of the training set and use it as the root of the tree. To find the two children of a codeword, two perturbed centroids are chosen as initial values. Then the Lloyd algorithm is used to find the optimal code words for the two children. By iterating this process, we build the tree until we reach the desired depth or the desired distortion.

The encoding is slow, but it has to be calculated only once. When the codebook is created, the creation on the high-frequency band is really fast, because it is proportional to the depth of the tree. And we know that $depth \leq \log_2(cardinal(TrainingSet))$. The larger the book, the smaller the distortion is, but the more time the process takes.

We could even expect to be able to do real-time super-resolution, and thus being able to watch a movie in a bigger resolution.

But I did not have time to finish to implement the TSVQ, and the last release still uses the brute force match.

III - Results

In this section I present the most interesting results I got. The two first sections are devoted to understand to influence of the Training Set images and of the α parameter. Then I present the results on zooming in textures.

And the end, in the last two sections, I show some examples, some providing good results, and others that are not really satisfactory.

III.1 Influence of the training set

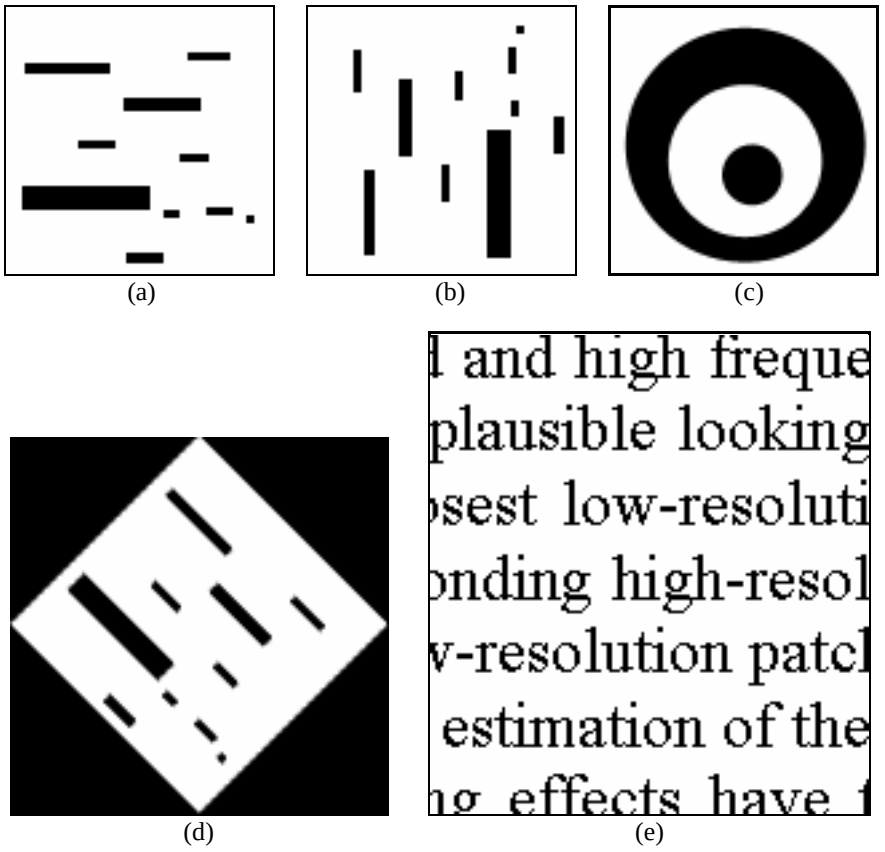


Figure 9 : Different Training Set images used in this section.

Figure 10 : Input image, at its original size, and up sampled with bilinear interpolation. The last image is the mid-band of the interpolated image.

Here the training set images were chosen to verify their influence on the result. The horizontal, vertical and diagonal examples show how the style of the training set can be seen in the result. In figure 14 (using the circle image), the leaves are all rounded, but the edges are sharp. When these four images are used together, the result is much better, much more realistic. This shows that instead of using one big image as training data, it is better to use a diversified set of smaller images.

The test with the “text image” shows how different parts of the training set (here different characters) are combined to create edges.

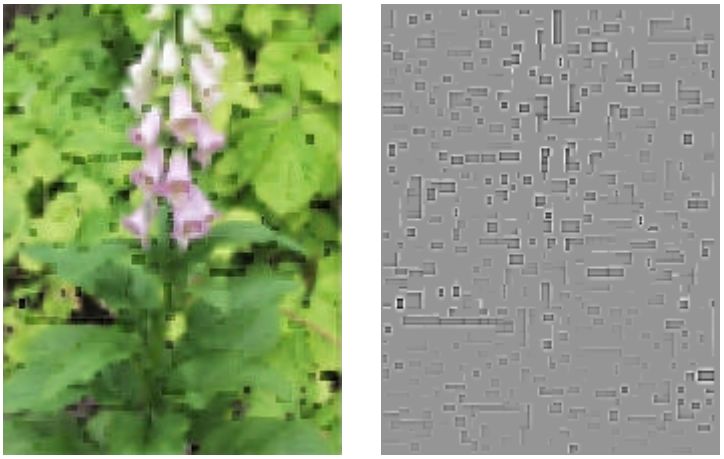


Figure 11 : Super-resolution using the “vertical” image (a). On the right is shown the high-frequency band generated by the program.

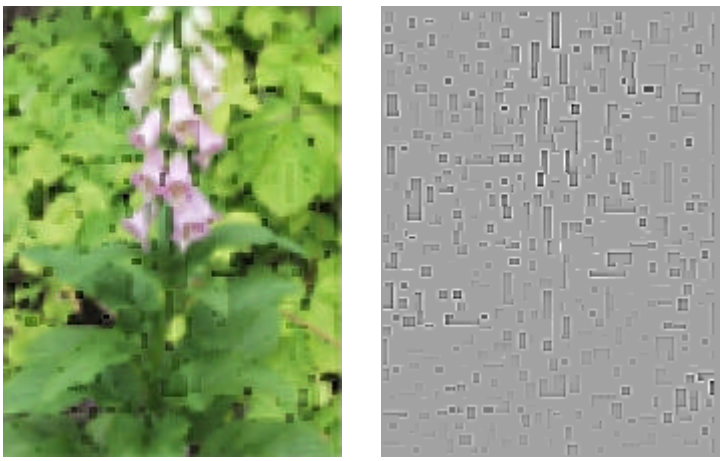


Figure 12 : Super-resolution using the “horizontal” image (b).

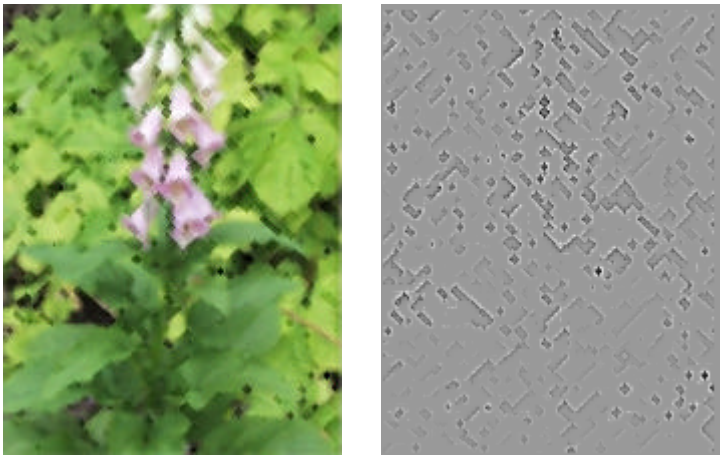
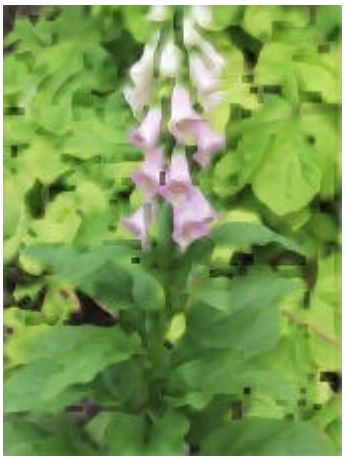


Figure 13 : Super-resolution using the “diagonal” image (d).

Figure 14 : Super-resolution using the “circle” image (c).



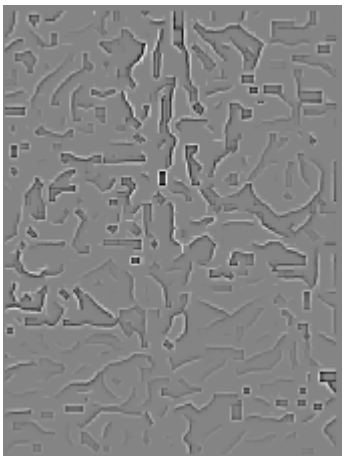
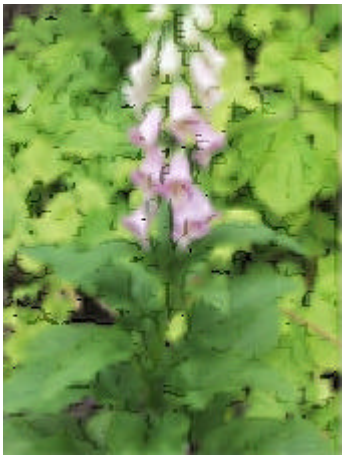


Figure 15 : Super-resolution using the four images (a), (b), (c) and (d).



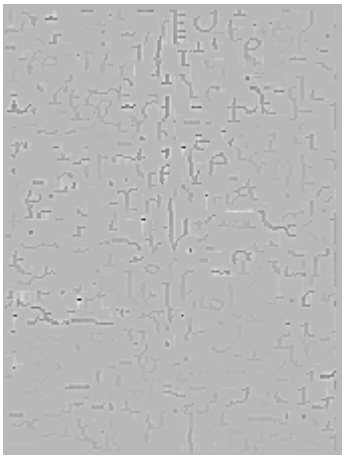


Figure 16 : Super-resolution using the “text” image (e). Note that the algorithm tries to “create” edges with the characters.

III.2 Influence of the a parameter

I have done the same tests with different values for a ; the Training Set was composed of one image, also representing lily of the valley. In each case, only an interesting part of the result is displayed. All images are displayed with the double of their real size for a convenience purpose (except for $a = 0$).

- $a = 0$: the overlap does not affect the matching process, only frequential effects are taken into account. (here the images are displayed four times bigger than their real size).

Figure 17 : $\alpha = 0$

On the left is the result of super-resolution with $a = 0{,}5$ and on the right with $a = 0$. Here the result is not as bad as Freeman said, but we can see some artefacts when $a = 0$, due to the discontinuity of the high-frequency band. The artefacts are 4 pixel-large (the size of the high-frequency patches (5 px) minus the overlap (1 px)).

- $a = 0{,}5$: this is the value recommended by Freeman. The quality is good.

Figure 18 : $\alpha = 0{,}5$

On the left is the bilinearly interpolated image, and on the right the result of super-resolution. Note that with super-resolution we get a foreground (the two big flowers) and a background (on the top-right and on the bottom-left), which we cannot really see with bilinear interpolation.

- $a = 5$: the overlap has a more important influence in the matching process.

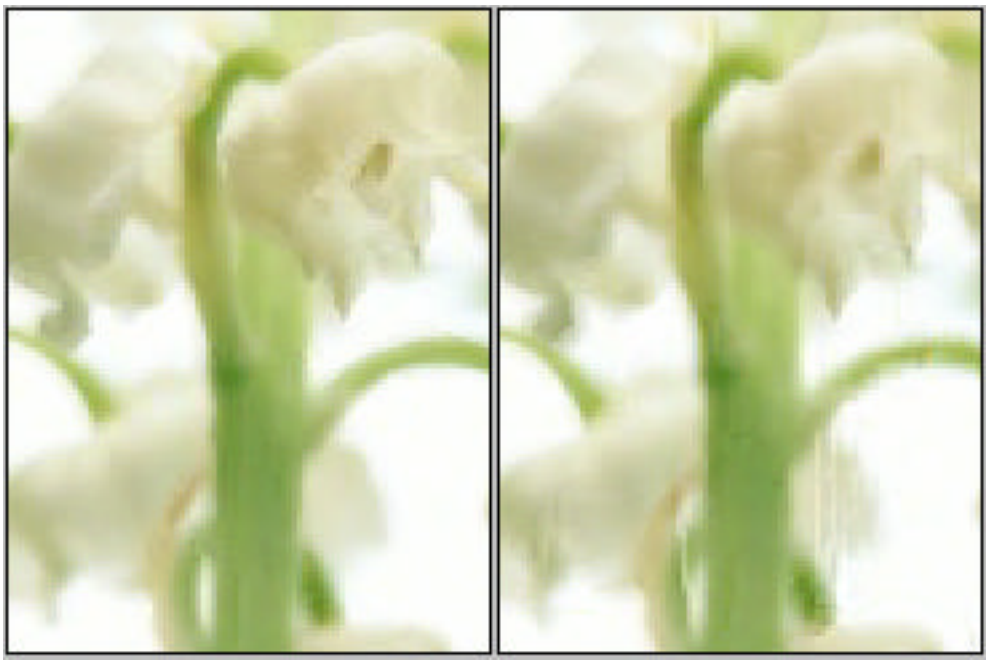


Figure 19 : $\alpha = 5$

On the left is shown the generated image with $\alpha = 0{,}5$ and on the right with $\alpha = 5$. We can see that some artefacts appear, even in smooth parts (on the bottom-right).

- $a = 500$: the frequential effects are hardly taken into account.

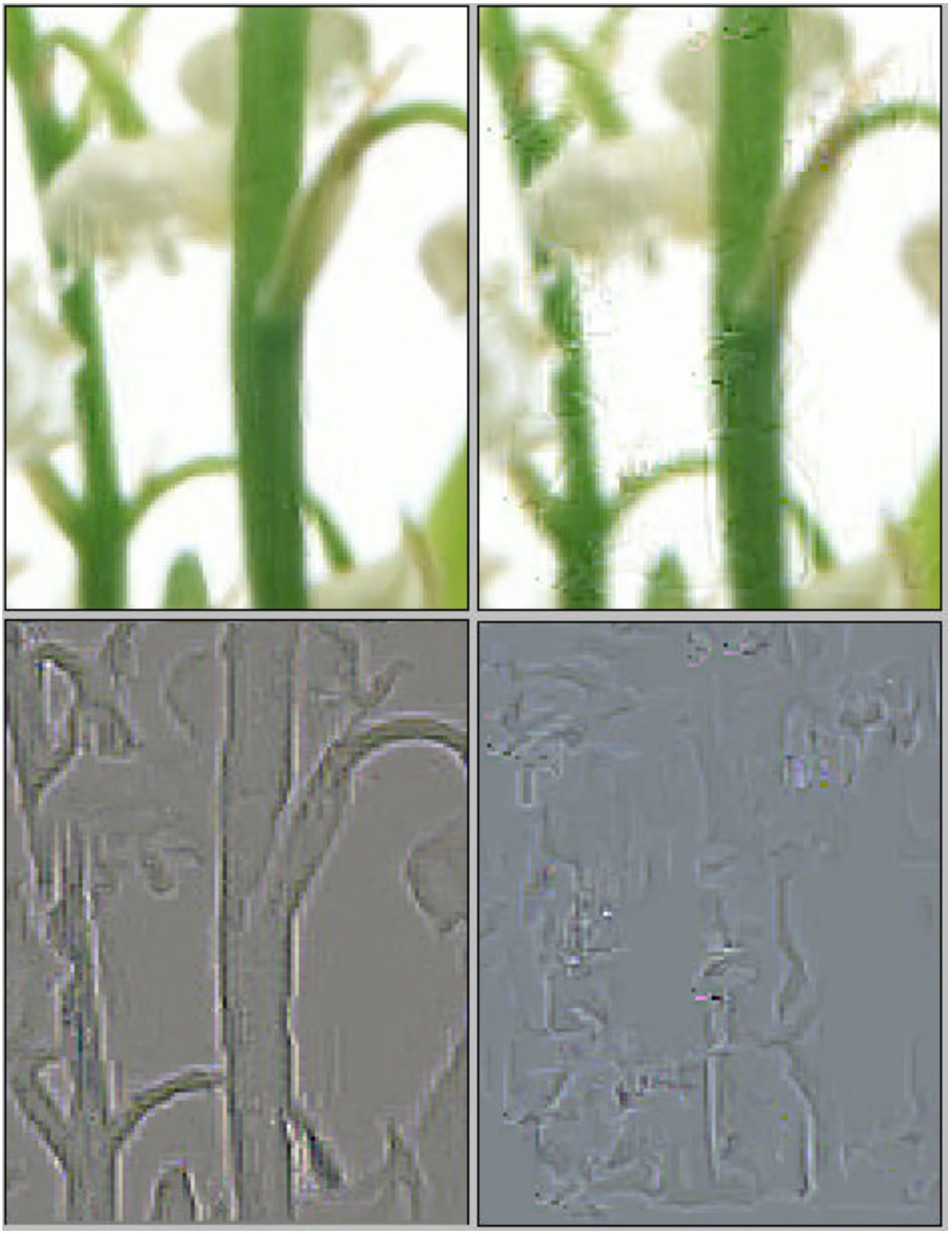


Figure 20 : α = 500

Here are displayed on the top, the results with $a = 0{,}5$ (left) and $a = 500$ (right), and on the bottom the corresponding high-frequency bands generated by super-resolution. The high-frequency band does not look like the image itself, and thus, once added to the interpolated image, it creates many artefacts.

III.3 Textures

I have tried to zoom in textures. Usually when you zoom in a textured polygon-based object, you have the choice between :

- interpolate the images on their surface, and then create a blurred texture.
- keep the same texture : the object is bigger, but you see the same texture. But that does not look natural.

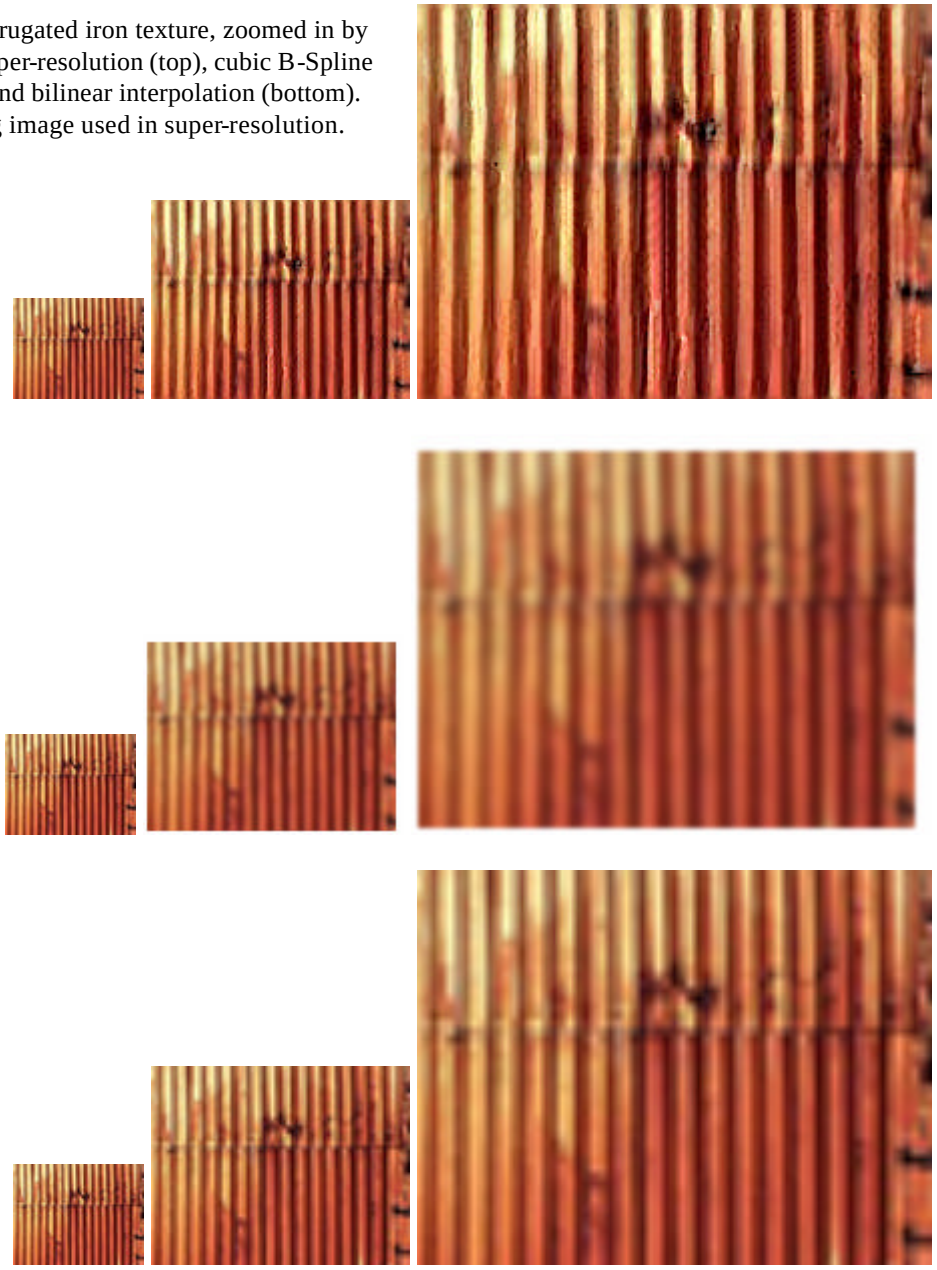
Super-resolution can help creating plausible details on the texture. By using the texture example itself – or any image related to it – as a training image, it is possible to generate a zoomed version of it : you get a sharp and bigger version of your former texture.

Wall.

Figure 21 : a 52x52 wall texture, zoomed in by a factor 2 and 4 with super-resolution (top), cubic B-Spline interpolation (center) and bilinear interpolation (bottom). On the bottom-right is the training set image used in super-resolution.

Corrugated iron.

Figure 22 : a 65x50 corrugated iron texture, zoomed in by a factor 2 and 4 with super-resolution (top), cubic B-Spline interpolation (center) and bilinear interpolation (bottom). Up-left is the training image used in super-resolution.



Palm tree trunk.

Figure 23 : palm tree trunk (left), zoomed in by a factor 2 with super-resolution (center-left), cubic B-Spline interpolation (center-right) and bilinear interpolation (right).
Down-left is the training image used in super-resolution.

Coloured circles.

Figure 24 : Training set image.

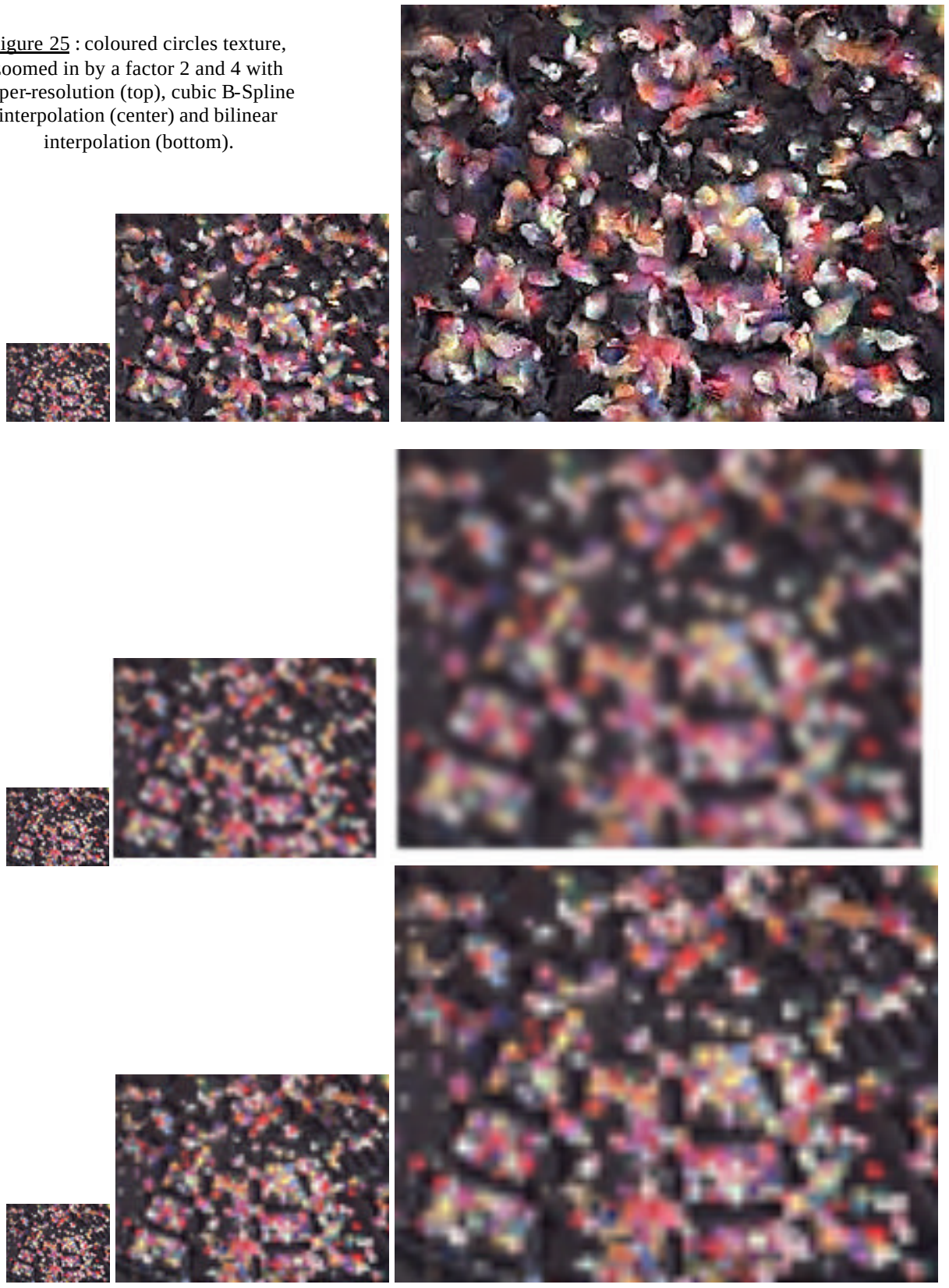


Figure 25 : coloured circles texture, zoomed in by a factor 2 and 4 with super-resolution (top), cubic B-Spline interpolation (center) and bilinear interpolation (bottom).

III.4 Others results.

All the results of this section have been calculated with this Training Set :

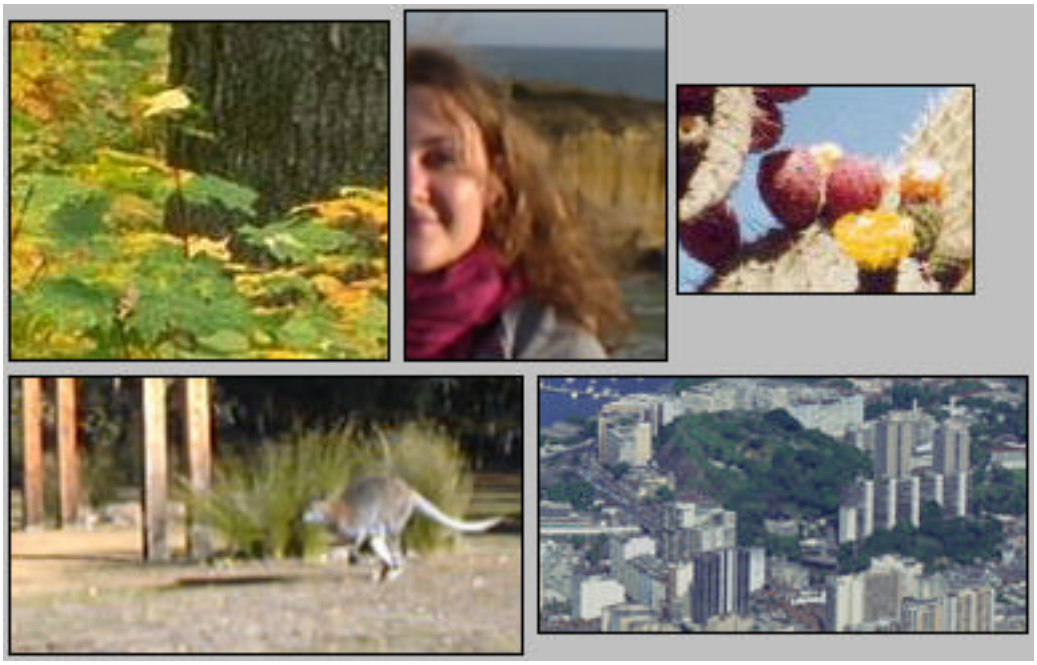


Figure 26 : Training Set images.

In my examples some images are too big to be shown in this paper, I only display parts of them.

Corcovado.

Figure 27 : detail of the Corcovado, zoomed in by a factor 2 and 4 with super-resolution.
Note that the jpeg artefacts from the original image are amplified by super-resolution.

Stone wall.

Figure 28 : detail of a stone wall, zoomed in by a factor 2 with bilinear interpolation (left) and with super-resolution (right).

Branches.

Figure 29 : Branches, zoomed in by a factor 2 with bilinear interpolation (left) and with super-resolution (right).

Desert plant.

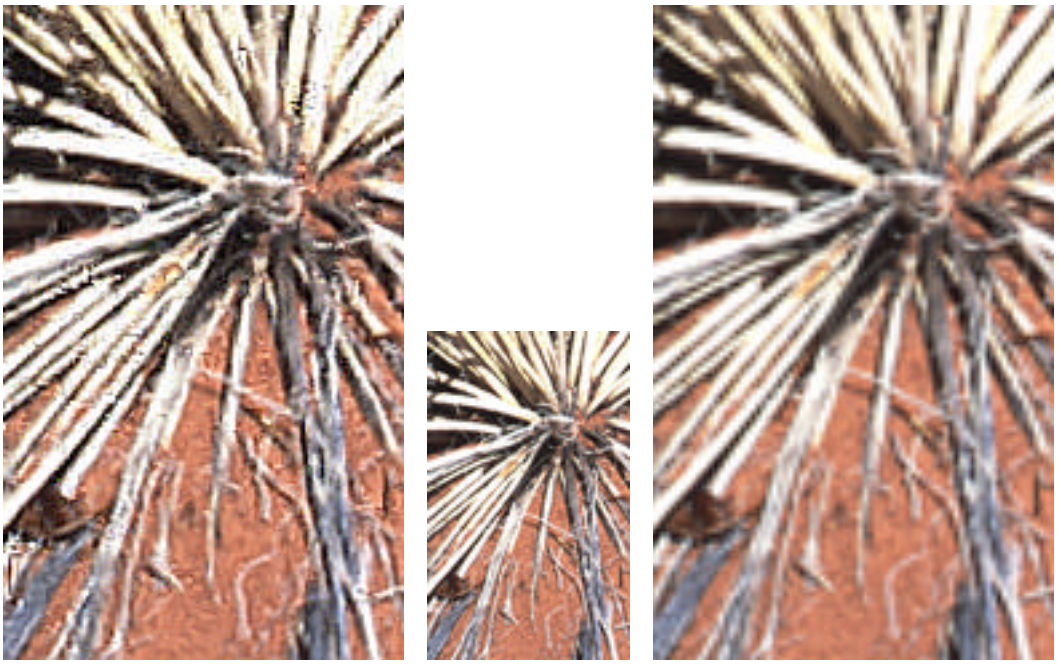


Figure 30 : Desert plant : original image (center) sharpened with super-resolution (left), and up sampled with bilinear interpolation (right).

Jaguar.

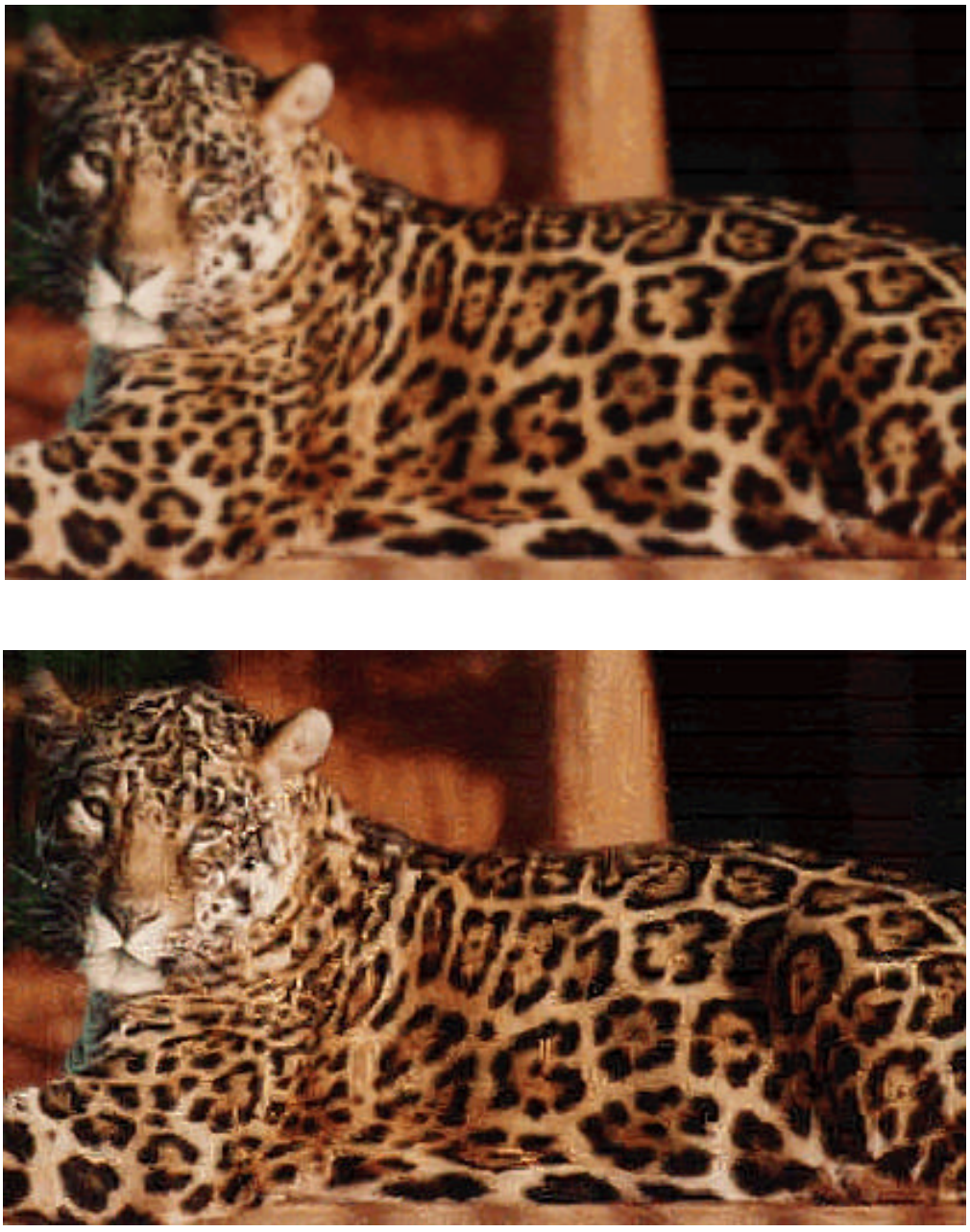


Figure 31 : Jaguar, zoomed in by a factor 2 with bilinear interpolation (top) and with super-resolution (bottom).

III.5 Bad results

With the example of the Corcovado, we have seen that zooming with a factor 4 using super-resolution amplifies the jpeg artefacts. Actually, the little errors created while zooming of a factor 2 are also amplified during the second pass. Here in the aerial view (figure 32), the factor 2 zoom is satisfactory, but the factor 4 is not, just like with the branches (figure 33).

Aerial view.

Figure 32 : Aerial view of a city, zoomed in by a factor 2 and 4 with super-resolution.

Branches.

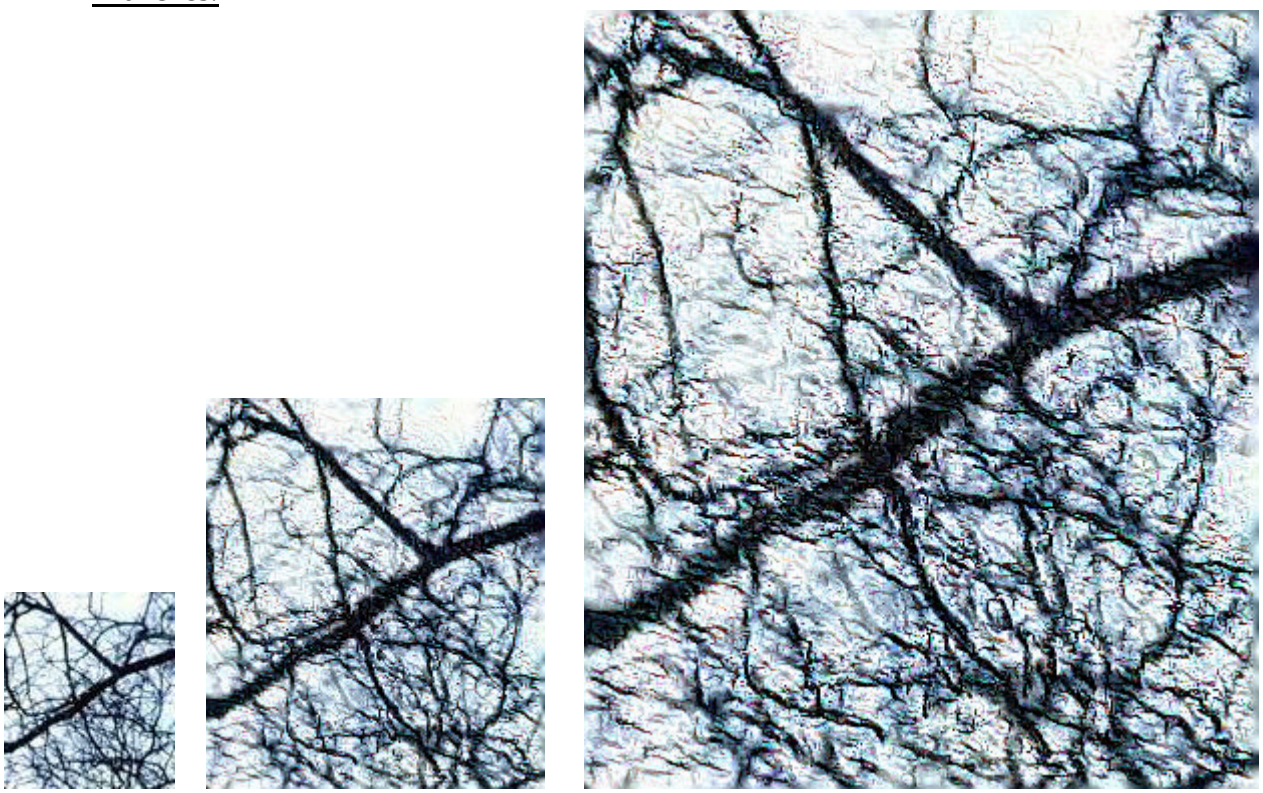


Figure 32 : Branches, zoomed in by a factor 2 and 4 with super-resolution.

IV - Image analogies

In his paper, Hertzmann gives his results for two super-resolution examples : first a set of maple trees, and then a Dobag rug. I have applied super-resolution to the same images. In this section my program does not zoom in the input image, and assumes the input image is a blurred one that only needs to be sharpened.

IV.1 The maple trees

Hertzmann has used three training pairs of images, but I just used the sharp image of the first pair, thinking that many look-like images do not lead to better results but increase the processing time.

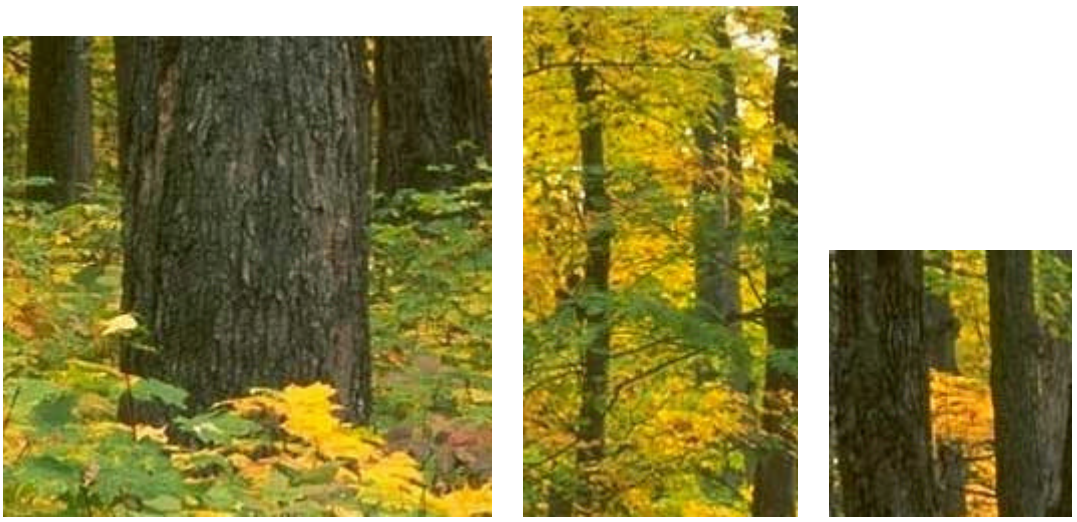


Figure 33 : Training images used for Image Analogies.
applied to the maple trees. Only the first one
is used in Super-resolution.

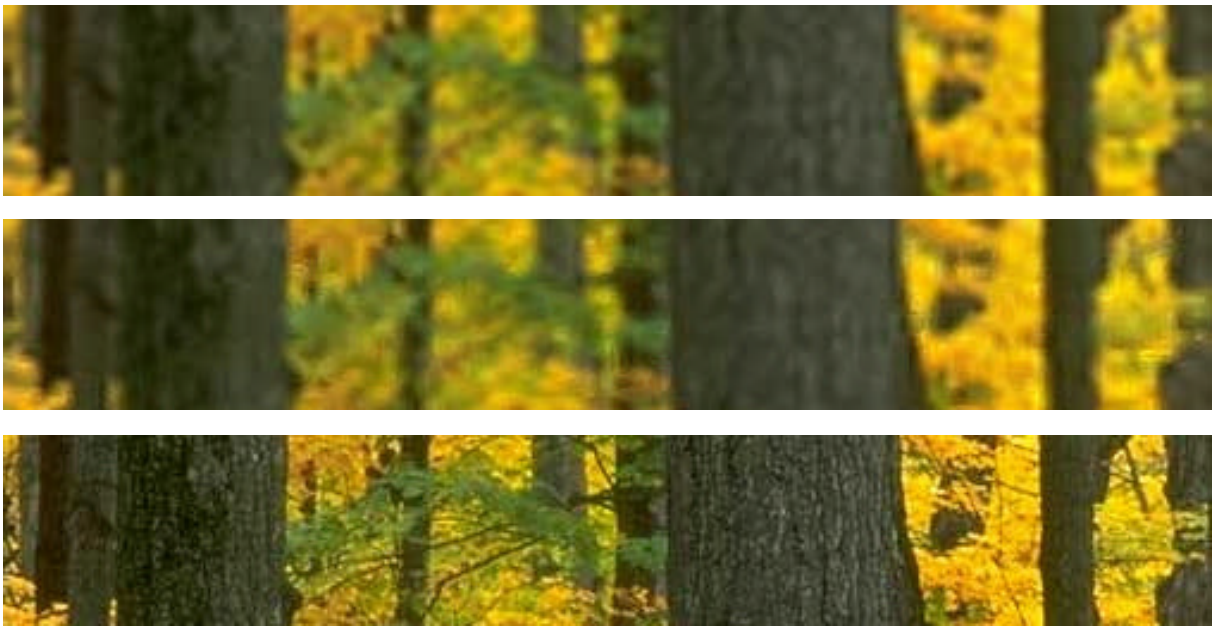


Figure 34 : The original image (top), sharpened
by super-resolution (center) and by Image Analogies (bottom)

Here we can see that super-resolution does not work at all, and the difference between the input image and the calculated scene is hard to detect. On the contrary, Image Analogies creates a very good sharp image.

Then I tried to use super-resolution the way it was originally built for : zooming in a sharp image. Thus I have subsampled the sharp version of the input image, and applied super-resolution to it. The results are better, as you can see in figure 35.

Figure 35 : The subsampled image (top), interpolated (center-top),
Sharpened by super-resolution (center-bottom) and by Image Analogies (bottom)

IV.2 The rug

I used the same training images as Hertzmann :

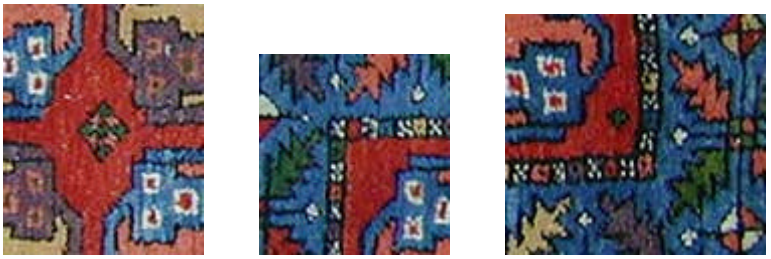


Figure 36 : Training images used for Image Analogies
and Super-resolution, applied to the rug. Image Analogies
also uses the blurred version of these images.

I did some different tests.

First I just sharpened the input image : I created a new image of the same size, assuming that the blurred input image had just lost its highest-frequency band. But the result was not good at all, since the input image was, to my mind, too blurred, and too much information was missing. The results are in figure 37.

Figure 37 : First attemp : the original blurred image (left),
sharpenend by super-resolution (center) and by Image Analogies (right).

Then I tried to cheat : instead of using only the sharp training images, I decided to use the blurred ones too. During the generation of the Training Set, instead of selecting the high- and mid-frequency bands of the sharp training image, I used the mid-band of the blurred image and the high-band of the sharp one. The result was not really good, as you can see in figure 38.

Figure 38 : Second attemp : sharpening with super-resolution (left),
original sharp image (center), and sharpening with Image Analogies (right).

Finally I decided not to use the blurred image as an input. I simulated a real zoom, by first blurring and subsampling the *sharp* input image. Then, I resampled it to its original size, and tried to sharpen *this* blurred image. The result was much better, as you can see in figure 39.

Figure 39 : Third attemp : the original sharp image (left),
blurred by sub- and up-sampling (center),
and finally sharpenend by super resolution (right).

IV.3 Discussion

In both cases, Image Analogies give better results than super-resolution, because the input images are too blurred.

Super-resolution differs from Image Analogies in that we use patches to create the new image, instead of operating per-pixel. That difference provides a performance benefit to super-resolution.

Furthermore, we normalise the contrast of our Training Set, and assume that the high-frequency only depends on the mid-frequency band, without considering the lowest octaves. These two generalizations increase the efficiency of our Training Set, freeing us from the constraint of operating on input images similar to the Training Set images.

V - Conclusion

I have implemented the one-pass algorithm that Freeman describes in [2], and I got many good results, especially with textures. Many images zoomed by super-resolution look better than with bilinear or cubic B-Spline interpolation. However, sometimes some artefacts are created, and amplified when zooming with a factor of 4. My program is really slow (more than hours to generate the Training Set shown in figure 26, and many hours more for zooming in the input images) and this was a brake for setting the paramaters (choice of the Training Set images, value of α).

The bad results when compared to Image Analogies may be explained dy the fact that super-resolution is better at zooming in a sharp image, when you are sure that only the high-frequency band is missing. It cannot sharpen an image that is too blurred.

One main thing remains to be done : implementing the TSVQ acceleration, allowing us to set the parameters, and, once set, to be able to do real-time processing. It may be done soon, in order to be applied to the Visiorama project developed in the VISGRAF at the IMPA [10]

As we saw that zooming with a factor of 4 or more does not provide good results (except for texture rendering), one could try to generalize super-resolution using a larger pyramid decomposition of the images. Super-resolution only decomposes images using three levels : low-, mid-, and high-frequency bands, and get satisfactory results when zooming with a factor of 2. Jumping directly from one original resolution to a better one may be possible by decomposing images in more bands.

An interesting application would be when working with satellite images : you only need a few very-high-resolution images for your training set, and later you can increase the resolution of images from another satellite which has weaker cameras.

Other points could be developed. The interface, particularly, should evolve in many ways, as described in the Appendix 2. Others techniques of interpolation, and other frequency filters could be tested to find the best ones.

Acknowledgments

Special thanks to : Stephane Mallat for having proposed this intership in Rio to me ; Luiz Velho for his warm welcoming and his support ; Adelailson Peixoto and Sergio Pinheiro for their accelerated C++ lessons ; Loïc, Edwige, Simon, Marie-Claire, Frédéric and Mathieu from the Ecole polytechnique for calculating many results on their own computer for me ; Gérard Rhinn for his high-resolution photos ; all the VISGRAF students for taking me around Rio at night.

Bibliography

[1] W. T. Freeman, E. C. Pasztor, and O. T. Carmichael. Learning low-level vision. *Intl. J. Computer Vision*, 40(1):24-47, 2000.

[2] W. T. Freeman, T. R. Jones, and E. C. Pasztor. Example-based super-resolution. *MERL*, TR-2001-30.

[3] A. Hertzmann, C. E. Jacobs, N. Olivier, B. Curless, and D. H. Salesin. Image analogies. In *ACM SISGGRAPH*, 2001. In *Computer Graphics Proceedings*, Annual Conference Series.

[4] W. T. Freeman and E. C. Pasztor. Learning to estimate scenes from images. In M. S. Keaarns, S.A. Solla, and D. A. Cohn, editors *Adv Neural Information Processing Systems*, volume 11, Cambridge, MA, 1999, MIT Press.

[5] E. P. Simoncelli. Statistical models for images : Compression restoration and synthesis. In *31st Asilomar Conf. On. Sig. Sys. And Computers*, Pacific Grove, CA, 1997.

[6] Wei, L.-Y., and Levoy, M. Fast texture synthesis using tree-structured vector quantization. *Proceedings of SIGGRAPH 2000*, July 2000, 479-488.

[7] Jill R. Goldschneider. Source code of TSVQ. http://rcs.ee.washington.edu/COMPRESSION/code/tsvq/

[8] D. Kerstem, A.J. O'Toole, M.E. Sereno, D.C. Knill, and J.A Anderson. *Associative learning of scene parametres from images*. In Applied Optics, 26(23):4999-5006.

[9] A.C. Hulbert and T.A. Poggio. *Synthesizing a color algorithm from examples*. Science, 239:482-485.

[10] Visiorama Project. http://www.visgraf.impa.br/visorama/

[11] Paul Bourke. Bicubic Interpolation for image scaling. http://astronomy.swin.edu.au/~pbourke/colour/bicubic/

Appendix 1 : Pre-processing functions

A. Blur-filter

The image is blurred by applying $[0{,}25 \quad 0{,}5 \quad 0{,}25]$ in each direction.

B. Interpolation

A bilinear interpolation is used. It is not the best interpolation, and thus our algorithm has more work than if we used another one. However, working with bilinearly interpolated images in the Training Set enables the program to sharpen blurred images more easily.

C. Contrast normalisation

To normalise the pair of high- and low-resolution patches, I divide it by the *energy* of the low-resolution patch. This energy is the average absolute value of the low-resolution patch:

$$energy = 0{,}01 + \sqrt{\sum |m_i|^2}$$

where m_i is the value of the pixel number i in the low-resolution patch. The $0{,}01$ constant is added to prevent from dividing by zero.

D. Mid-band filter

As we have to get the mid-band from an image that has no high-frequency band, we remove the lowest frequency band, by applying the following filter :

$$\begin{bmatrix} -1/9 & -1/9 & -1/9 \\ -1/9 & +8/9 & -1/9 \\ -1/9 & -1/9 & -1/9 \end{bmatrix}$$

Appendix 2 : Software

While computing Freeman's algorithm, I manipulated various images, creating others, and changing parameters to find the best ones. The need of an interface for loading and displaying these images appeared, and thus I have implemented a first release of a super-resolution software.

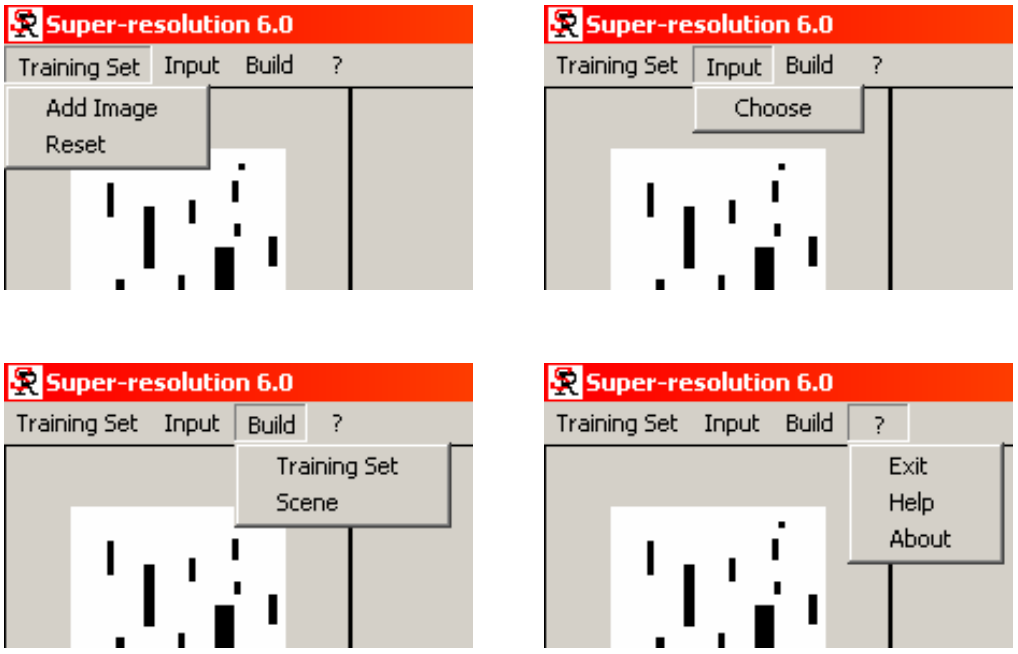
Display

The displaying is composed of four parts :

- Up-left, we can see the Training Set images ;
- Down-left are the properties of these images (name, size, number of patches) and of the Training Set itself, particularly its status (generated or not)..
- Down-right are displayed the input image and the output scene when it is calculated.
- Up-right are the input image properties (name, size, number of patches) and the scene properties. The complexity is the product of the number of patches in the Training Set and the number of patches that need to be calculated.

Functions

This software only has basic functions, many others will be added in a later time. We can load the Training Set and the input images, empty the Training Set, build it, and build the scene. The “Help” function is not computed yet, as the use is really easy.



Future extensions of the software:

1. **Multithread** : while calculating the scene or the Training Set, the program cannot do anything else, like re-drawing the dialog window. During this time we can do nothing else but wait for the end of the process, without knowing how long we have to wait for the result. A multithread release would permit to display the progression of the process.
2. **Zooming factor** : this release zooms with a factor 2. It would be interesting to have the possibility to choose it : factor 0 (just sharpening), or factor 2^n, for any integer n.
3. **Alpha parameter** : the α parameter permits to adjust the relative importance of the low-frequency patch against the high-frequency overlap. It would be interesting to be able to control it, particularly using the TSVQ acceleration, to see in a dynamic way what is his best value.
4. **Saving the Training Set into a file** : generating the Training Set is quite a long process, especially using the TSVQ. Instead of generating it every time you launch the program, it is better to load an already-generated one and re-use it directly.
5. **Twin display :** once it is created, the scene has to be compared to an interpolated image (or the input image if the zooming factor is 0), to be sure that super-resolution is better than interpolation. Therefore, the idea would be to have a window showing the same part of these two images, and to have the possibility to zoom and to move them at the same time.