Chapter 5

Classification and synthesis of textures

5.1 Introduction

In this chapter we try to arrive to a description of textures in such a way that it should be able both to classify and synthesize textures. Why precisely these two purposes? The idea is to derive a texture representation that can be directly used to recognize or classify textures by defining a suitable distance or similarity measure between two such descriptions, and, *at the same time*, be able to apprehend what's the texture(s) ideally recognized or belonging to each class in a supervised framework. On way to do that is being able to synthesize any number of texture samples from a given built class. Thus, we can sort of 'explain' the decisions of the classifier. Although we limit to classification and synthesis this approach can be extended later to other areas as compression, filtering, segmentation, etc.

Among different proposals studied, we have chosen a probabilistic model to characterize and describe texture, because it is one of the most versatile. Besides, most of the studied models have some probabilistic background. When we see texture images we perceive that some relations and patterns repeat over the entire image. This behavior, to a higher or lower degree, allows us to guess some hidden probability model that we want to exploit.

Hence, to model texture we use the probability density function (pdf) based on the occurrence of pixel gray level values given a certain neighborhood, in a similar way as in the work of Popat [73]. To clarify the idea of this model let us consider this simple example: if we restrict this neighborhood to a single pixel we obtain as pdf the histogram of the studied image, which can be readily used. For instance, a L_1 or L_2 histogram distance functions could perhaps classify a set of textures. Similarity, the histogram could be used to predict the value of the next pixel based on that of the present one in certain causal scanning order.

But this is a too naïf situation. In order to obtain acceptable classification and (definitely) synthesis results, we must increase the size of neighborhood. We also explore other situations; changing the distribution of neighbors of a pixel we obtain different

density functions associated to such neighborhoods. These pdfs are n-dimensional functions, where n is the number of neighbors considered. There are as many pdfs as neighborhoods can be defined. Large neighborhoods allow representing better the complex interrelations present in real-life textures but this quickly sentences us to the curse of dimensionality. Therefore it is increasingly difficult or even impossible in practice to have enough data to obtain an accurate representation. Small or non-existent neighborhoods are easy to manipulate but they do not summarize long range the behaviors and relations between pixels. First solution, with large neighborhoods, is mandatory for several applications as synthesis of texture where relations among pixels must be taken into account. And there are some other solutions where so much demanding requirements are not needed as in the classification case. We should arrive to an average solution to obtain a good representation in all the cases, though it could be adapted to each situation.

5.1.1 Goals

This work represents an important change in relation to the preceding chapters. It is planned as an initial contact to probabilistic texture modelization and synthesis. Here, we present basically results obtained in classification and synthesis, but the most important part of this work is the different proposed alternatives and experiments done to arrive to these results.

A principal objective of this partial work is to arrive to a description useful to identify a texture. We searched in the literature which were the principal ideas in this field of texture modelization and realized that synthesis needs strongly a good model in which it can be based because procedural synthesis methods usually fail to generalize to different classes of textures. Other problems related to texture analysis as classification do not have this major requirement of a good and complete model although there is always an implicit model. Hence, we want a trade-off between the two approaches analyzed.

Also, and based on the model of texture we also want to develop new strategies to classify and synthesize texture. To perform such tasks we propose for classification a direct comparison of texture models with a metric adapted to those models. For synthesis we explore previous works [72, 73] and propose alternatives to simplify calculation and to extend the model to a multiresolution decomposition framework.

5.1.2 Background

As we mentioned before, synthesis is a good starting point to search a plausible model because it is a very demanding process over it. It is necessary to well characterize a texture in order to produce similar but not simple copies of the original image. Once we have a model, we can of course validate it through synthesis. Having such a representation allow us to address other problems related to texture analysis with a better understanding of their internal relations and their construction. Besides, if we obtain a compact representation, then chances are that we can employ it in new fields like compression, restoration, and generation of computer graphic images. Specifically, computer graphics field is one of the most actives areas with regard to

synthesis, because it feeds on texture to achieve good and realistic visualizations.

Synthesis requires several model parameters to take place. Therefore, we need a previous stage of analysis where an image or a set of images is presented to an algorithm which extracts a parametric model. This process is known as 'synthesis by analysis'.

A review of the literature in this field points out the seminal work of Heeger and Berger [39], a clear precursor of other more recent works. They start with an analysis stage by applying a decomposition based on a steerable pyramid [88]. Then, to achieve a synthetic image, they apply the same process to a random noise and try to match the histograms of both trees at each level of the decomposition. Applying this matching of histograms a certain number of times they obtain a synthetic image remarkably similar to the first one. Besides, this technique was extended to color images using the K-L transform an applying to each output channel the same process. This algorithm presents some drawbacks: the number of iterations needed to achieve good results must be set by trial and error, and the impossibility of synthesizing textures presenting big and well-defined patterns (textural elements). But, in spite of these problems, results are great.

Portilla et al. [74] use also a decomposition scheme but in this case based on a Gabor transform. This decomposition is part of the analysis devoted to extract simple parameters that allow the texture description. The synthesis is also done from an initial noise image, which is transformed and adapted taking the parameters from the analysis process. Results are good for statistical textures but similarity decreases when some structured patterns appear.

De Bonet and Viola [22] are able to synthesize almost any texture, no matter whether statistical or structured. They propose a strategy based on a decomposition of one example image in a Laplacian pyramid. Then, the synthetic image is built by substituting each path of the original decomposition tree by similar paths of the same decomposition. In this way we obtain a new image with a similar aspect to the initial texture image. This procedure yields a good visual impression because it plays with the same information for the synthesis than the source image, with non-radical changes in the representation.

Popat and Picard [73] exploit the fact that texture is a property that somehow repeats over the space, which can thus be explained as a probabilistic process. They propose the construction of a probability density function of an image by a sliding neighborhood. In [72] Popat explains a procedure to synthesize texture images from this pdf in a sequentially and hierarchical way. This approach is the one that we have adopted due to its versatility for different purposes.

Portilla and Simoncelli [75] try to merge the most important aspects of each approach, using joint statistical constraints of a decomposition based on a steerable pyramid. They analyze several kinds of textures finding possible failures and including such statistic constraints to avoid them. Synthesis results are impressive for many kind of textures and, compared to the former methods, their method is definitely able to generalize the learnt texture; it means that generated results, although perceived as the same texture, are completely different to those used in the analysis.

Bar-Joseph et al. [7] extend previous techniques to 1D, 2D, and 3D (temporal) textures. They even provide a method to fuse pairs of textures. In the 2D case

they decompose images with again a steerable pyramid and obtain several trees. To synthesize or mix textures they generate a new tree in a similar way than [22], but with several trees as input from the same texture (synthesis) or from different textures (mixing).

In sum, results for most methods presented here are good some of them have not a real parametric model behind, some others are visually a simple repetition of patches of an original texture. We want to build a model in order to recognize textures trying to discern the decisions taken by the classifier. This works is still in progress so some of our results and tests have not been explored in-depth.

5.2 Estimation and comparison of probability density functions

Once we have decided that texture will be characterized by a model based on a pdf, it is necessary to build this function from samples of the original texture(s) in the analysis step. Here, we are going to explain some basic aspects of how to perform this task. Afterwards, we have to select a criterion to compare them for classification purposes. In this section we will review some alternatives proposed in the literature and in the next section we will explain our specific choice.

5.2.1 Estimation of pdfs

To modelize a texture we will use a probability density function extracted from data of an image. We define a neighborhood and we model the image as the independent realization, for each pixel and its neighbors, of a probability density function[1] ($p(\boldsymbol{x})$).

Data used to envisage the model are extracted in an analysis stage from a texture image. The process used in this data extraction is a simply sliding neighborhood that scans the area of interest (in some causal order) accounting the frequency of these values. If we limit the neighborhood to a single pixel, we obtain the histogram normalized to area 1 as the simplest 1D pdf approximation. More interestingly, for larger neighborhoods we obtain probability laws of higher dimensionality:

$$p(\mathbf{X}) = \text{Prob}\{\mathbf{x} = \mathbf{X}\}, \quad \mathbf{X} \in X^d \ ,$$

where d is the dimension of the probability space, $\mathbf{x} = (x_1, x_2, ..., x_d)$ is a random vector, $\mathbf{x}$ is a particular value of $\mathbf{X}$, and $x_i \in X_i$. The proposed model implies independence among the different realizations, which actually is not true in a real image. Nevertheless, if internal dependences of the vector components are stronger than external dependence, then this requirement can be relaxed.

These probability laws are not the data itself. Data represent a series of realization of some random variables following those probability functions. Therefore, we need a procedure to generalize the observed data, and we follow the steps presented in [73].

[1] Strictly speaking, in the case of conventional gray level images we do not have continuous variables that can give rise to a true pdf, but a probability mass function (pmf). Nevertheless, for the sake of generalization, we will stick to the continuous notation, in case $\boldsymbol{x}$ is real valued (a floating point quantity.

One of these techniques is the histogram, which tends to the real function when data goes to infinity. But in practice and for high dimensionality spaces, this condition is never reached. This is a serious problem because most points in the space have zero probability even in high probability areas. Histograms summarize data, but cannot generalize. Another technique that tries to generalize is kernel regression [84]. In this case each sample data propagates its probability to the surrounding space with the help of a kernel function. In this case, we can generalize but we need to store all the data. This means that though we can generalize, we cannot summarize and it is an important drawback for a huge amount of data. As a final solution derived from the kernel regression is a technique based on cluster analysis. First, our pdf is expressed as a weighted sum of Gaussian functions [9]. This estimation tries to model our data as a set of clusters, the more complicate the distribution of data over the space, the more clusters we need for a reliable representation. An isolated cluster, in our case, is modeled as a Gaussian function. Second, in order to summarize the representation, a clustering process by k-means is performed, followed by a refinement step with the EM algorithm, which tries to slightly move the k centers to the best places.

k-means is a simple process that splits data into several clusters. The algorithm starts throwing some centers over the space[2] and associating to each center the nearest data. Then, we recalculate the new centers of each cluster as its center of mass and iterate the process until stationarity. This is a fast technique to achieve a preliminary result of the model. At this point we can try to represent each cluster with a unique Gaussian function.

A refinement step proceeds with the Expectation-Maximization algorithm. This is in essence very similar to the previous one, but in this case we do not group data to clusters by an Euclidean distance criterion. Instead, data are softly grouped based on a 'probability distance', data is assigned to each center of a group weighted by the expectation of belonging to that group. The mixture of Gaussian functions evolves in time to reach a good match between the data and the model.

This is the pdf estimation procedure we have selected.

5.2.2 Comparison of pdfs

Now, the idea is to use these pdfs in order to decide if an image belongs to a certain class of textures, represented by another pdf. The main reason to do this comparison is that, ideally, textures of the same class should give rise to the same probability function. Therefore, we want to measure how much similar are two pdfs. To compare pdfs we use a criterion based on the relative entropy or Kullback Leibler measure (Eq. (5.1)) that is a natural way to compare probability distributions [19]:

$$D(p||q) = \sum_{x \in X} p(x) \log \frac{p(x)}{q(x)} \quad . \tag{5.1}$$

The relative entropy is a measure of the 'distance' between two distributions. Although it is not a real distance (does not satisfy all the requirements of distance), it is often useful to think of relative entropy as a distance between distributions.

[2] We have chosen as centers samples of the initial data randomly.

Other works that rely on similar probability models [73] use for classification purposes a measure related to relative entropy. They assign to an image the class where it is coded with the fewest bits. Due to the fact that relative entropy $D(p||q)$ is a measure of the inefficiency of assuming that the distribution is q when the true distribution is p; hence, if we hit on the correct distribution of an image the coder give us a reduced relative entropy and compact output.

Also, we use this measure to know if a certain model is better than another one over a set of data. Thus, we can decide which are the best parameters of a certain process.

5.2.3 Generation of new values

To perform synthesis we need to obtain new values from data of a given neighborhood already calculated (we assume a causal order in the image generation). For a certain pixel location, let $X_1, X_2, ..., X_{d-1}$ the values of the preceding neighborhood pixels, and x_d that of the pixel to be calculated. We choose x_d as the value that better explains the occurrence of this neighborhood $X_1, X_2, ..., X_{d-1}$. This calculation can be done in different ways: (i) maximum a posteriori probability (MAP), (ii) least expected squared error (LSE), and we also choose (iii) pseudorandom values according to its conditional pdf. To get a random number with a particular distribution we obtain the function of distribution of this probability and apply it as an inverse map of a uniform random number.

$$\text{MAP} \qquad X_d = \arg\max_{x_d} p(x_d|X_1, X_2, ..., X_{d-1}) \ , \tag{5.2}$$

$$\text{LSE} \qquad X_d = E_p(x_d|X_1, X_2, ..., X_{d-1}) \ . \tag{5.3}$$

where E_p is the expected value according to pdf p.

All three of them have been assessed in the results section.

5.3 Our proposal

Here we explain which are the solutions adopted in our test in the three domains pointed out before: pdf estimation, pdf comparison, and texture synthesis.

5.3.1 Estimation of the model

We have compared several strategies such as histograms, mixture of Gaussians, and also we have tried to model the conjoint pdf as a product of independent single probabilities in order to estimate it as a product of marginals. Nevertheless, the mixture of Gaussian functions has gained most of the efforts because it has proven the best in synthesis (though not in time or memory performance). We started with simple models as 1D pdf rising to 2D pdfs and higher dimensions just as they are or with a reduction of dimensionality. To limit the amount of possibilities we have studied the estimation based on a 2D mixture pdf changing parameters like: number of centers, shape of the function that approximate cluster, number of iterations, and addition or not of a refinement step.

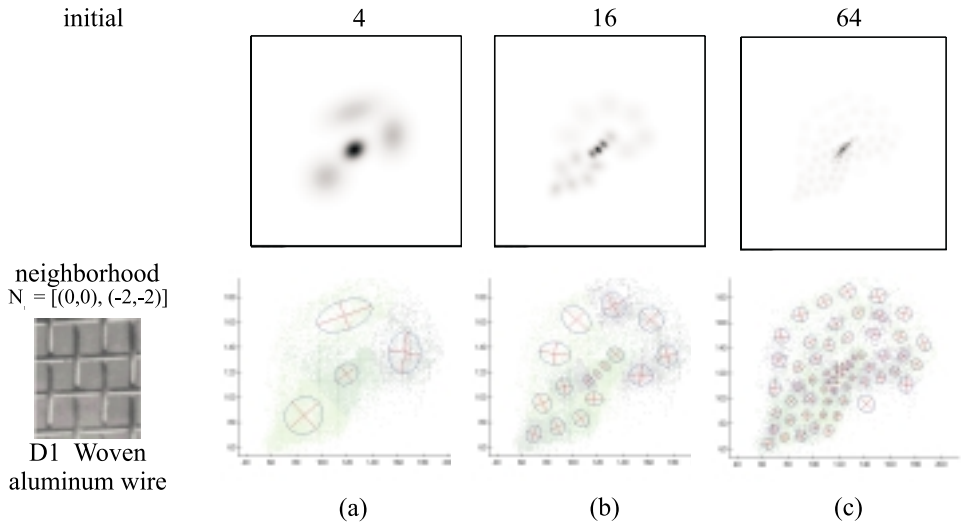

Figure 5.1: Estimation of an initial pdf as a mixture of Gaussian functions varying the number of centers. Up: initial pdf and approximations by a mixture. Bellow: texture used in the pdf computation and plots that shows centers and elongations of the Gaussians.

Figure 5.1 shows in the bottom left part the texture used in these preliminary trials of estimation. We use the neighborhood $N_1 = \{(0,0),(-2,-2)\}$; where $(0,0)$ means the analyzed pixel and $(-2,-2)$ the neighbor at this relative position. The accumulation of data is showed at top left and we see that its distribution cannot be easily explained with a single function. A first test, reflected in this figure, shows how the shape is better approximated when the number of centers increases. This approximation has been obtained with the k-means algorithm without the refinement step.

A second test tries to determine which is the best shape of the Gaussian functions given a certain number of centers. To do this experiments we force the covariance of each cluster to be: (a) a single value (spherical case), (b) a diagonal matrix (diag. case) or (c) without restriction (full case). To assess which is the best approximation we compute the relative entropy. We see in Fig. 5.2 how full case gives better results. Due to the fact that the initialization is a random step we can obtain different results; therefore we choose the best results among a set of trials.

After we have analyzed those results of the k-means step, we apply the refinement based on the EM algorithm to better adapt our model. We conclude that the shape of the function is important, but the introduction of this new stage allows a strong reduction of the distance between the estimated and the true model (Fig. 5.3).

A further improvement can be reached in the light of the previous experiments: to increase the number of centers, do not limit the shape of the Gaussian functions, and to apply the EM refinement. But all these things imply also a rise in computation time and storage requirements. Figure 5.4 illustrates how are affected this improvement in relation to the number of centers for a full case. We can see how the EM step gives a

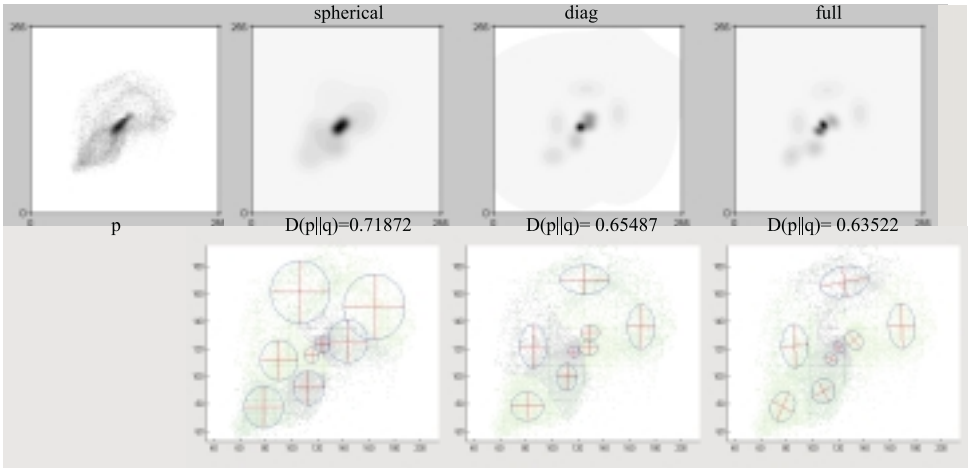

Figure 5.2: The best estimation of ten tests with several shapes of Gaussian functions and their relative entropy.

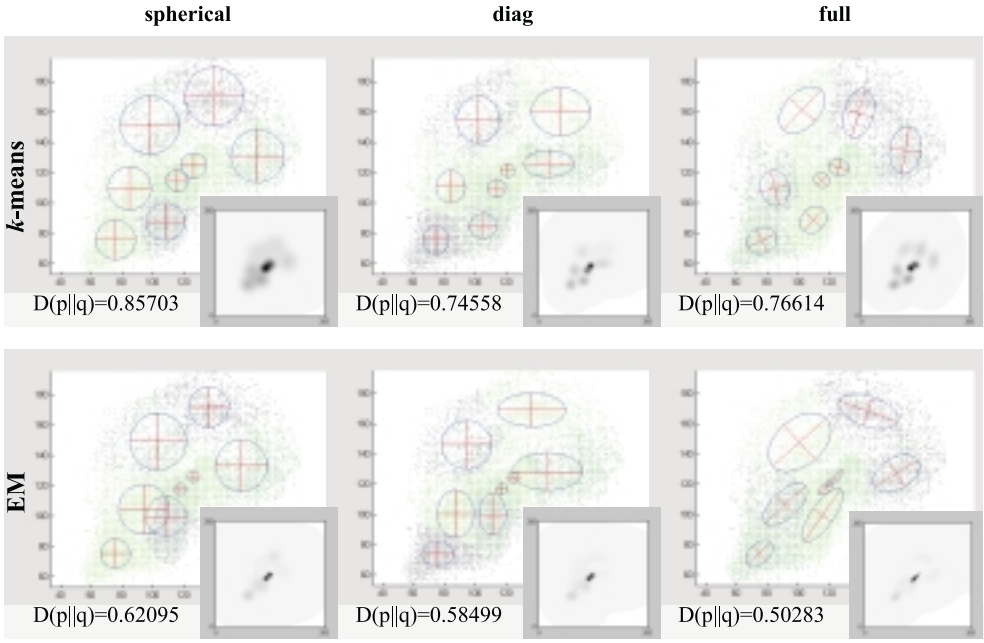

Figure 5.3: Improvement of estimation using EM algorithm.

better approximation of the model, but also shows how the number of centers is not a critical parameter because we soon arrive to a stationarity case.

In this search of a good estimation, we only show and specific configuration of neighbors but it is clear that global shape on the accumulation space depends on the

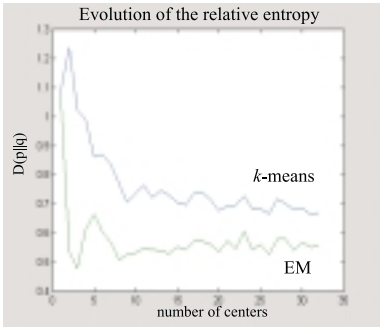

Figure 5.4: Stabilization of the goodness of the estimation increasing the number of centers.

neighborhood. We have done several tests changing those neighborhoods, but we have not thoroughly studied how those distributions affect the classification and synthesis results. One special case must be point out for synthesis purposes, which use causal neighborhoods where the new values need as neighbors already calculated positions.

We try to introduce a multiresolution approach at this point doing pdf estimation by means of the solution presented in [98]. In this case, we obtain an estimation reconstructing a few levels of the decomposition. Figure 5.5 shows a histogram generated from a texture image and three estimations of this distribution where the number of levels in the reconstruction increases and therefore also the similarity to the original histogram. The extension of this procedure to 2D pdfs is easy to implement, but if we grow further in dimensionality the computational requirements of size and time increase exponentially. Therefore, we have stopped these experiments at this point, but this can be considered as a future theme to be explored in-depth.

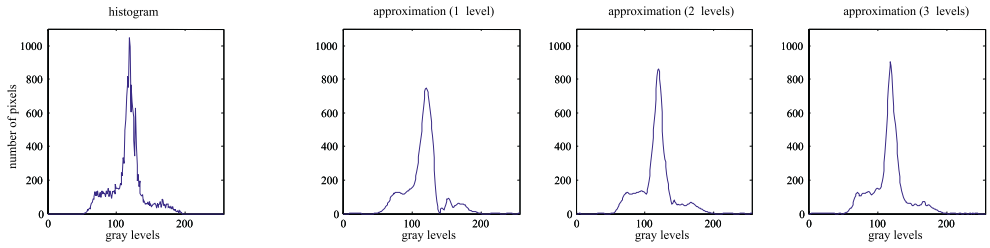

Figure 5.5: Estimation of a density function by partial reconstruction of a wavelet decomposition.

5.3.2 Classification

To estimate the degree to which a sample belongs to a certain class we compare the estimated pdfs of both. The classification rule is to assign the image to the class which reaches minimum distance. Tests have been done using simple distances like the L_1, i.e. the sum of absolute values of the difference of components; or L_2, i.e. the

Euclidean distance. These two measures are generic distances and not well adapted to our case. We need to compare pdfs that are a special case of functions, and which have specific similarity measures to perform that comparison. Therefore, we also adopt as potential measure some 'distances' based on the relative entropy explained in Sec. 5.2.2.

$D(p||q)$ is always non-negative and is zero if and only if $p = q$. However, it is not a true distance between distributions because does not satisfy symmetry and the triangle inequality. To build a real distance we need to obtain both properties without losing the other two. We try to remove the non-symmetry drawback calculating the maximum between the relative entropy in both senses:

$$\mathcal{D}(p,q) = \max(D(p||q), D(q||p)) \ . \tag{5.4}$$

It means that with this similarity measure comparison between p and q gives the same result than between q and p. This new measure $\mathcal{D}$ is not a real distance; it still lacks the triangle inequality, but allows a fair comparison of a pair of pdfs. Proving that this measure is symmetric is trivial and that triangular inequality is not satisfied can be done easily by a counterexample. We have also tested other symmetric distances such as the sum of both relative entropies, but similar results were obtained. It remains as an open issue to find a real distance able to establish a metric in the space of those models.

5.3.3 Synthesis

The idea of synthesis based in a probability model is to obtain pixels of the new image following the probability law of the model. We have done a lot of trials varying several aspects of this idea. First, in most trials, density estimation was performed using a mixture of Gaussian functions, estimating the pdf with a k-means as initialization stage and after a stepwise refinement with EM algorithm. In some experiments we have expressed the probability model as an independent probability process. Second, and related to how to generate new values, we use the three proposal of MAP, LSE and random mentioned before.

If we try to build a new image as a set of independent trials following a correct probability distribution, we fail if we do not include neighborhood information. See in Fig. 5.6 how result looks like a noise image without defined texture patterns although it obeys the correct density function. To do this test we build a model of ten dimensions and generate random samples obeying this law but with random neighbors. This neighborhood, also used in other tests, is: $N_{10} = \{(0,0), (-1,0), (-2,0), (2,-1), (1,-1), (0,-1), (-1,-1), (-2,-1), (1,-2), (0,-2), (-1,-2)\}$.

First attempts of synthesis follow the guidelines of [72]. With a causal neighborhood (N_{10}) we try to perform a sequential synthesis (see Fig. 5.15) and a hierarchical synthesis (see Fig. 5.16). An aspect that strongly affects the final result is the previous initialization. We use a causal neighborhood, but if first values do not correspond to typical values of the image the synthesis does not 'start' generating similar values and soon gets completely lost. Figure 5.7 shows on the first group of synthesis how it fails completely for MAP and LSE cases if we choose a uniform initialization, only