Texture Analysis in Aerial Photography

Jörn Hoffmann and Yossi Rubner

December 4, 1998

1 Abstract

Numerous practical applications of aerial photography require images to be interpreted in terms of their local texture. In order to facilitate the task of finding a texture of interest in an image or a set of images, a system is developed which employs a spatial frequency characterization of textures to detect a given texture in an image. The frequency content in different parts of the image is analyzed by an expansion into a set of *Gabor functions*. The coefficients in this expansion are used to characterize the texture in terms of different spatial scales and spatial frequencies. A similarity measure that compares local distributions of these parameters between a sample texture and the image can be used to highlight similar textures in an image.

The details of this approach are the topic of this project paper. In particular we discuss the idea of Gabor filtering in the fourier domain. An argument is made for the chosen set of Gabor functions (Gabor filters). Experiments are perfor1med to assess the ability of the system to detect textures at different scales in real images. Furthermore, we evaluate the possibility of detecting similar textures independently of their orientation in the image, which will be desirable in a number of possible applications.

2 Introduction

Aerial photography and other remote sensing image products have increasingly become an important means of acquiring information on a wide range of structures and parameters on the earths surface. The vast amount of data acquired by different imaging systems is leading to difficulties in the time-intensive task of analyzing the resulting images. The development of tools which render the analysis of images more efficiently and that are making classifications within images possible in a more objective way is therefore desirable.

A lot of the information in an image is contained in its local texture and a number of applications are concerned with the detection of areas within a set of images that match a given texture pattern. Examples of this are land-use analysis or studies of urban development. Suburban development or agricultural monocultures as well as certain agricultural techniques appear as a characteristic texture in an aerial photo.

The perception of texture is a fundamentally neurophysiological problem. Recent evidence from studies of the visual cortex of mammal brains show, that the filter response profiles of the main class of linearly-responding cortical neurons (called *simple cells*) can be modeled as a family of self-similar 2D-Gabor wavelets [4]. We therefor chose to employ such a set of Gabor wavelets as a means to characterize the texture.

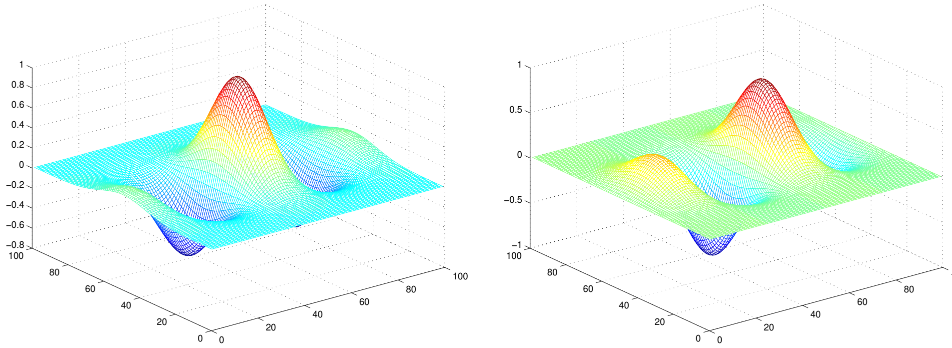
The system allows the user to specify a sample area in the image and compares the characteristics of the filter responses for this area to the filter responses in the whole image. The distributions are compared in a chi-squared-sense. Similar distributions are used to identify areas of similar texture and are highlit in the image.

Filtering any given image with a two-dimensional filter in the space domain would be far too expensive. The proposed texture analysis algorithm by examining the response of a dictionary of Gabor filters requires the image to be filtered with every filter in the dictionary. Our system employs 24 of these filters. The cost of 24 two-dimensional convolutions for any image of larger size would not be feasible, even on a fast computer.

The reason this approach can be pursued is the existence of the FFT algorithm and the property of the Fourier transforms, that a convolution in one domain is performed by a simple multiplication in the other domain. As the filter wavelets can be stored or constructed in their frequency representation already, we simply take the two-dimensional Fourier transform of the image and multiply it with every filter in the dictionary. The results are then inverse transformed.

3 Gabor filters

A Gabor function is a two-dimensional Gaussian, modulated with a complex exponential. Its frequency domain representation is also Gaussian, convolved with a



(a) Real part of g(x,y) (b) Imaginary part of g(x,y)

Figure 1: Real and imaginary parts of the Gabor function g(x,y).

δ-function, i.e. a two-dimensional Gaussian with some displacement in the (u,v)-plane.

$$g(x,y) = \left(\frac{1}{2\pi\sigma_x\sigma_y}\right) e^{-\frac{1}{2}(\frac{x^2}{\sigma_x^2}+\frac{y^2}{\sigma_y^2})} e^{i2\pi Wx} \tag{1}$$

$$\begin{aligned} G(u,v) &= e^{-2\pi^2(\sigma_x^2u^2+\sigma_y^2v^2)} **\delta(u-W) \\ &= e^{-2\pi^2(\sigma_x^2(u-W)^2+\sigma_y^2v^2)} \\ &= e^{-\frac{1}{2}\left(\frac{(u-W)^2}{\sigma_u^2}+\frac{v^2}{\sigma_v^2}\right)} \end{aligned} \tag{2}$$

where $\sigma_u = (2\pi\sigma_x)^{-1}$ and $\sigma_v = (2\pi\sigma_y)^{-1}$.

The shape of the real and imaginary parts of $g(x,y)$ is shown in figure (1).

In equation (2) the fourier transform convolution theorem was employed. Functions of the type (1) are called *Gabor functions*. Gabor discovered that any one-dimensional signal or filter must be quantized in a way that it cannot occupy less than some minimal area in the conjoint time-frequency domain [3].*

The parameters σ_x and σ_y have the effect of generalizing the shape of the Gaussian hump in two dimensions, so that the contour lines can now be chosen to be elliptical rather than circular.

*This is analog to the uncertainty principle of quantum physics, were the product of uncertainties in space and momentum cannot be less than a specified constant (the Planck constant).

Applying the function $g(x, y)$ as a filter to an image amounts to a convolution of the image with this function. In the frequency domain, the fourier transform of the input image is multiplied with the transfer function, i.e. G(u,v). Again, the choice of σ_u and σ_v controls the shape of the contour lines of the transfer function while W can be chosen to place the Gaussian hump in the (u,v)-plane. These properties make Gabor functions suitable for the analysis of images with respect to their local frequency content.

We now want to construct a set of Gabor functions which are chosen to be sensitive to various scales of spatial frequency at different orientations in an image. Placing the Gaussians at a larger radial distance from the origin in the (u,v)-plane increases the frequencies the filter is sensitive, the angle at which the Gaussian is located corresponds to the orientation. The rotation theorem ensures that the angle in the frequency domain corresponds to the same angle in the space domain. To understand this idea, a pictorial viewpoint in terms of corrugations and the respective filter responses is given in figures (3,4). For now, we define a filter bank as

$$\begin{aligned} g_{mn}(x, y) &= a^{-m} g(x', y') \\ x' &= a^{-m}(x \cos\theta + y \sin\theta) \\ y' &= a^{-m}(-x \sin\theta + y \cos\theta) \end{aligned} \tag{3}$$

where $a > 1$; m,n integer.

The angle $\theta = n\pi/K$ determines the orientation of the function. Here, the parameter K is the number of orientations. a is the scale factor which can be used as a parameter to change the radial spacing of the centers of consecutive filters in the (u,v)-plane. The factor a^{-m} in equation (3) ensures that all filters in the filter bank have the same energy:

$$
\begin{aligned}
E_{mn} &= \iint_{-\infty}^{\infty} |g_{mn}(x,y)|^2 dxdy \\
&= \iint_{-\infty}^{\infty} |a^{-m} g(x',y')|^2 dxdy \\
&= \iint_{-\infty}^{\infty} |g(x',y')|^2 dx'dy' \\
&= \iint_{-\infty}^{\infty} |G(u,v)|^2 dudv \\
&= (2\pi\sigma_x\sigma_y)^{-1} = \text{const}
\end{aligned}
\tag{4}
$$

which is obviously independent of the choice of m and n. In equation (4) we made use of Parsevals theorem.

The rotation theorem of two dimensional fourier transforms states that a rotation in either domain corresponds to a rotation by the same amount in the other domain. Making use of this property, the transformation (3) can be used to place a number of these filter wavelets in the (u,v)-plane at different radial distances and different angles, eventually covering most of it.

This is shown in figure (2), where the half-width contours of a set of wavelets are plotted. As can be seen from the plot, they are arranged such that the half-width contours of two neighboring filters touch. Note that adding all the contribution of these filters together will *not* give a constant response for all frequencies. But as the transfer function is known, the original image could be reconstructed from the filtered results. Of course, to make such a reconstruction possible the filter bank must be chosen to capture most of the frequencies, which is generally true in our design. The reconstruction of the original image from the coefficients of this expansion into Gabor wavelets has been the topic of a variety of computer visualization research, especially since it has been suggested that this approach parallels mammal vision [1].

4 The filter parameters

There are some requirements and system design considerations that have to be used in the optimal choice of the parameters (σ_x, σ_y), or equivalently (σ_u, σ_v). Firstly, the filter Gaussians are to be spaced logarithmically in the frequency plane,

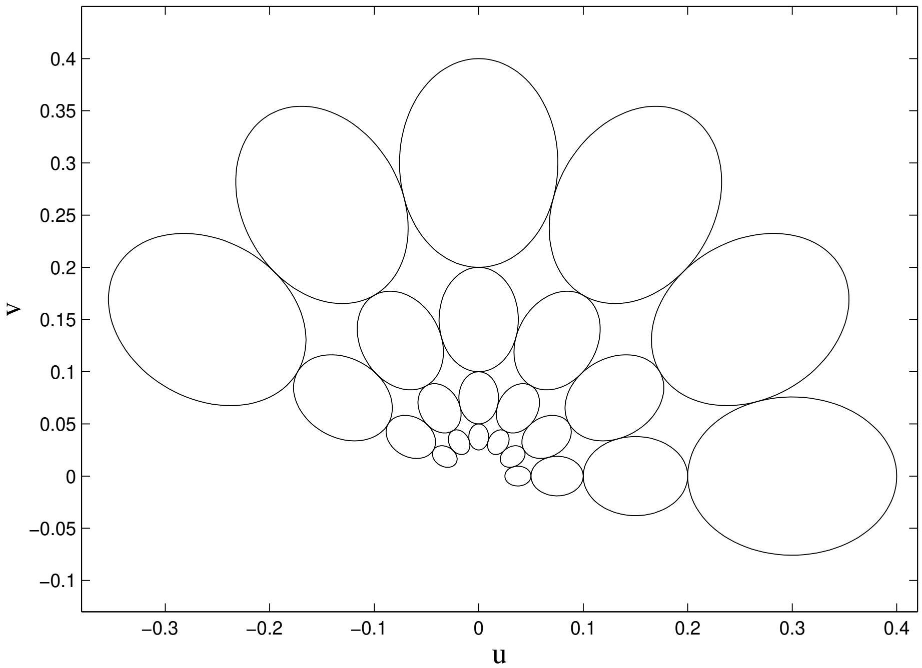


Figure 2: Halfwidth of filters in the spatial frequency domain. The different filters are spaced in a way that they intersect at their half-amplitude level. The parameters for this plot are $U_h = 0.3$ and $a = 2$.

i.e. their width increases exponentially with distance from the center of the (u,v)-plane. This requirement is justified here only by reference to earlier work on Gabor filter expansion (i.e. [5]). A somewhat hand-waiving way to justify this is to say, that natural images have been shown to exhibit an exponential decrease of energy with frequency ([2]). Using a logarithmic spacing of the filter wavelet in the frequency domain therefore 'bins' a similar amount of energy into every one of the filters.

Denoting the lowest and highest frequencies of interest U_l and U_h respectively, the first filter is centered at distance U_l from the center. If S denotes the number of scales used in the expansion, the centers of the Gaussians can be located in exponentially increasing distances:

$$u_s = U_l * a^s$$

where

$$\begin{aligned} a &= (\tfrac{U_h}{U_l})^{\frac{1}{S-1}} \\ s &= 0, 1, 2, \ldots, S-1. \end{aligned}$$

The standard deviations of the Gaussian σ_u and σ_v (or equivalently σ_x and σ_y) are chosen so they touch at their half-amplitude level in the frequency domain. A detailed derivation of the following expressions can be found in Appendix A.

$$\sigma_u = \frac{a-1}{(a+1)2\ln 2} U_h \tag{5}$$

$$\sigma_v = \tan(\frac{\pi}{2K}) \sqrt{\frac{U_h^2}{2\ln 2} - \sigma_u^2} \tag{6}$$

For our system we chose U_h to be 0.3 and fixed the parameter a to be 2, placing one filter every octave. This seems to be a natural choice, although experiments with closer and wider spacing could be performed to optimize the system. The optimal choice of these parameters depends on the image to be analyzed and the texture of interest. The upper bound for the spatial frequency is 0.5, which is the Nyquist frequency for an image that is 'sampled' at every pixel. U_l cannot be chosen arbitrarily small, because as the filter becomes narrower in the frequency domain, it becomes wider in the spatial domain. If the filter becomes wider than the patches of the textures of interest in the image, the performance of the system decreases rapidly, as the filter is now only sensitive to some average property rather than the *local* texture.

Figure (2) shows the location and width of the implemented filter wavelets.

5 Sample responses

Figures (3,4) are supposed to give a pictorial impression on how the filter works. On the left hand side a corrugation at a certain frequency and certain direction is given (except for the last case). On the right hand side, an array of 4 by 6 images of the same size as the input image (scaled for display) is shown. Every field is the filter response of the filter with the corresponding scale and orientation. The brighter the respective area, the greater the response. The responses of all 24 filters can be regarded as a vector of 24 components. There is one of these vectors for *every point in the image*. The distribution of these vectors in a certain area can be interpreted in terms of texture. The details of the classification are explained in the next section. It can be seen from figures (3,4) how every corrugation leads to a unique and characteristic filter response, if all responses are taken into account simultaneously. The last row with the circular symmetry shows how only those parts of the filter show a response, where the corresponding orientation is met in the image!

Similarly, figure (5) shows a synthetic image we used to examine the performance of the filtering. The top left area is a cosine-corrugation, the top right area is a corrugation rotated by 45 degrees, the bottom left area is the sum of the two top corrugations and the fourth area is random noise.

We specified a test-area in all four areas respectively. The results are illustrated in figure (6). The left hand side of the figure shows the sample textures used and the right hand side shows the chi-square distance. Brighter areas indicate smaller distances. The system clearly proves to be capable of finding and highlighting the correct textures in this simple image for these unambiguous choices of sample areas.

Interestingly it is found, that random noise can be treated as a texture pattern.

For ambiguous choices of a sample area (figure (7)) the results are as would be expected: The boundary regions between the two chosen areas give the brightest return. Areas with textures that are only partly found in the sample, have a darker color while the unmatched areas remain black. It is noticeable, that in the case where random noise and the top right corrugation are sampled, the return from the noise area is considerably brighter.

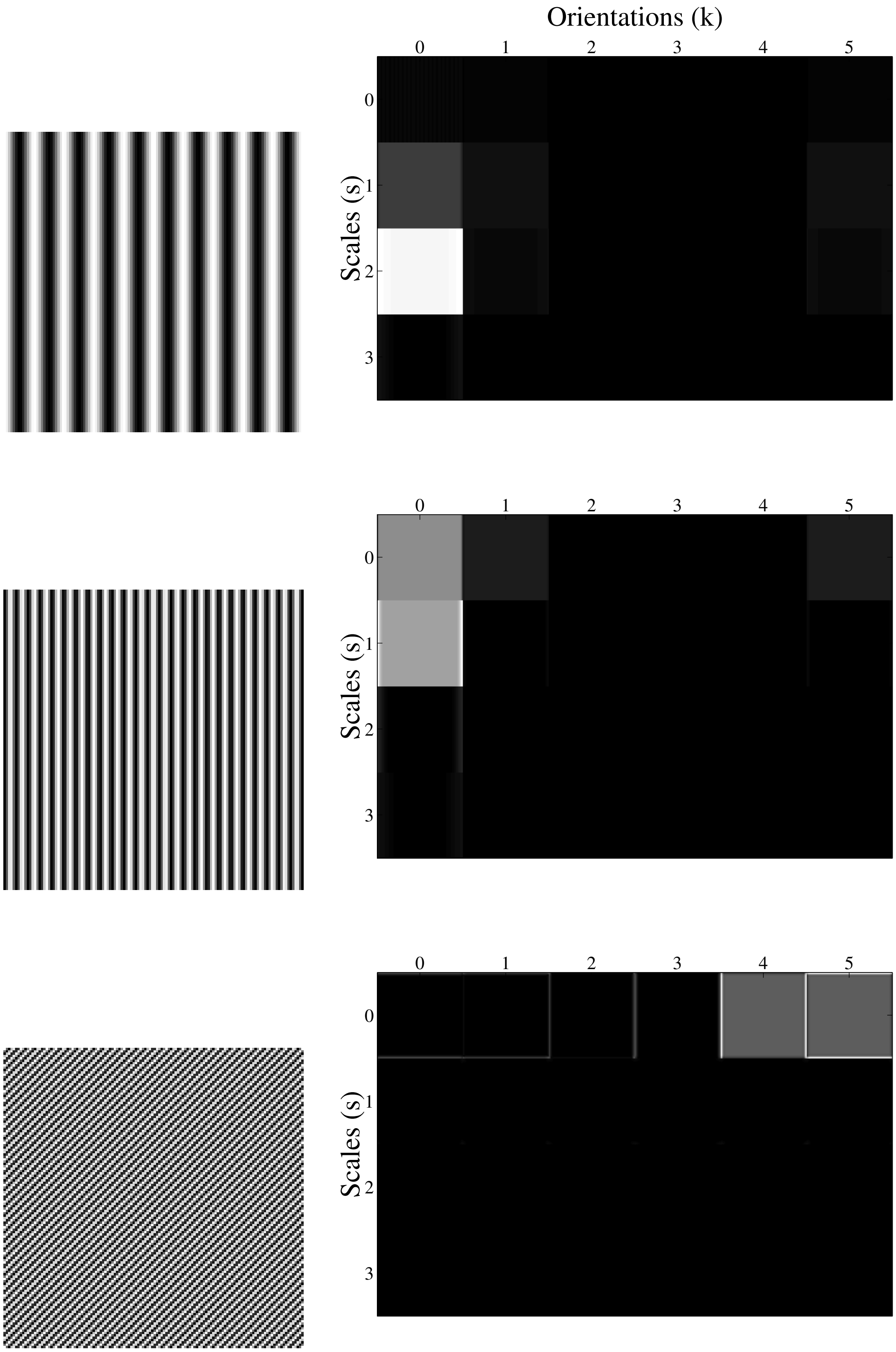


Figure 3: Filter responses for all 24 filters for some corrugation images.

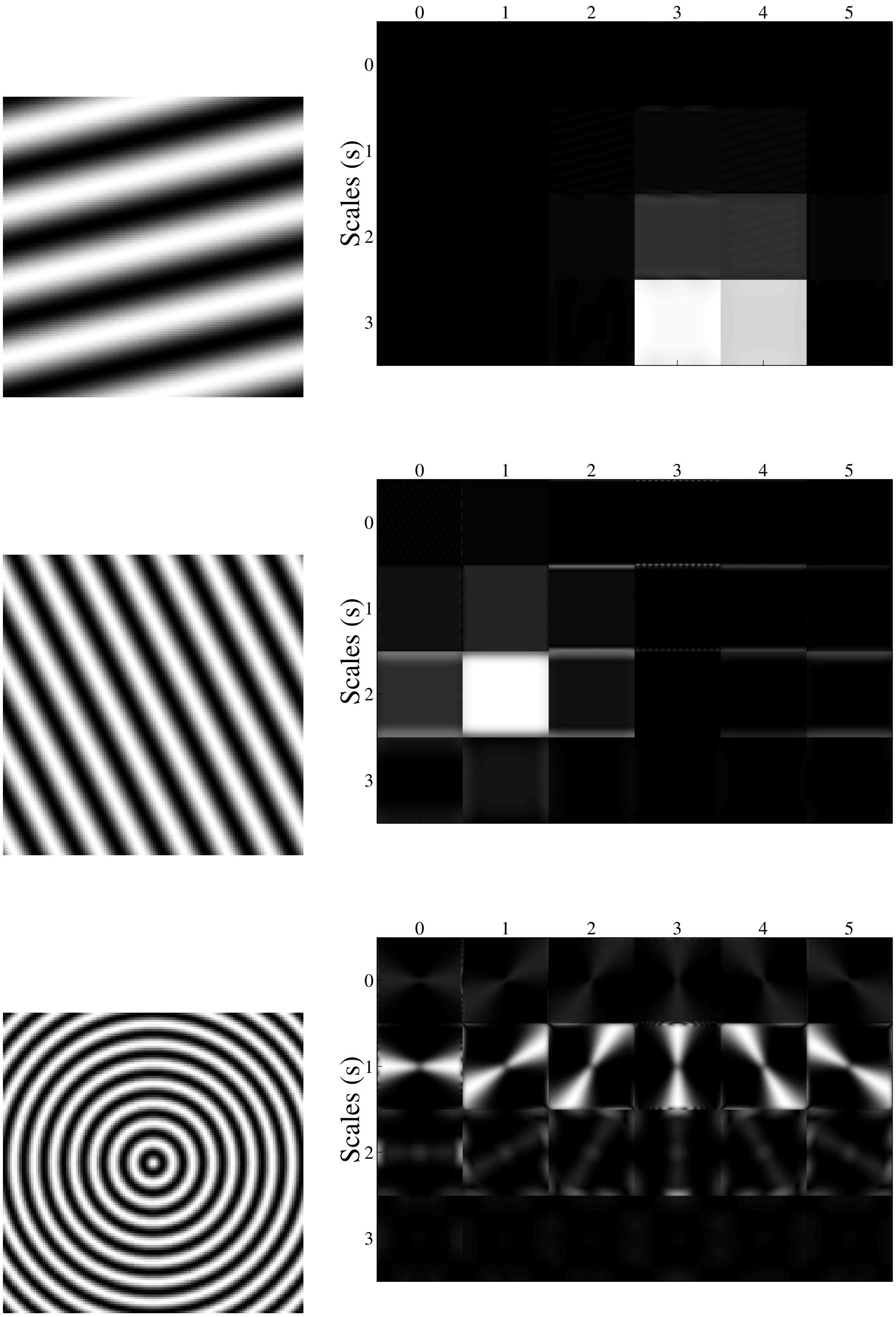


Figure 4: Filter responses for all 24 filters for some corrugation images.

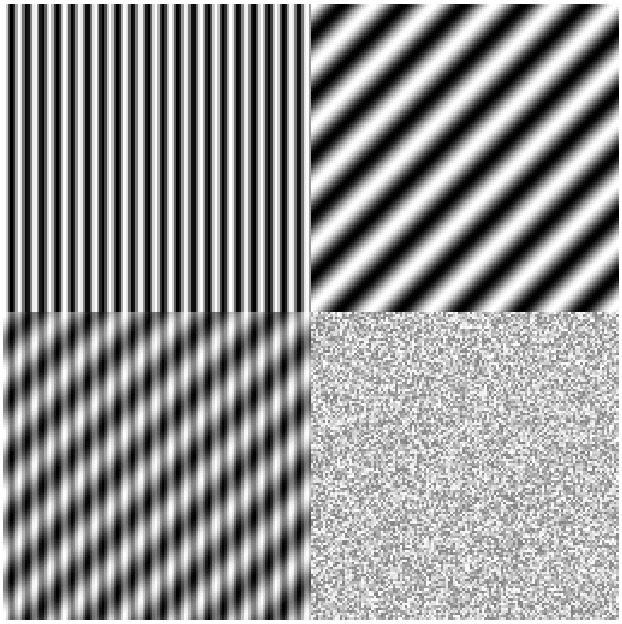


Figure 5: Synthetic image used for test of texture analysis system. The bottom left texture is the mean of the two top images, the bottom right image is random noise.

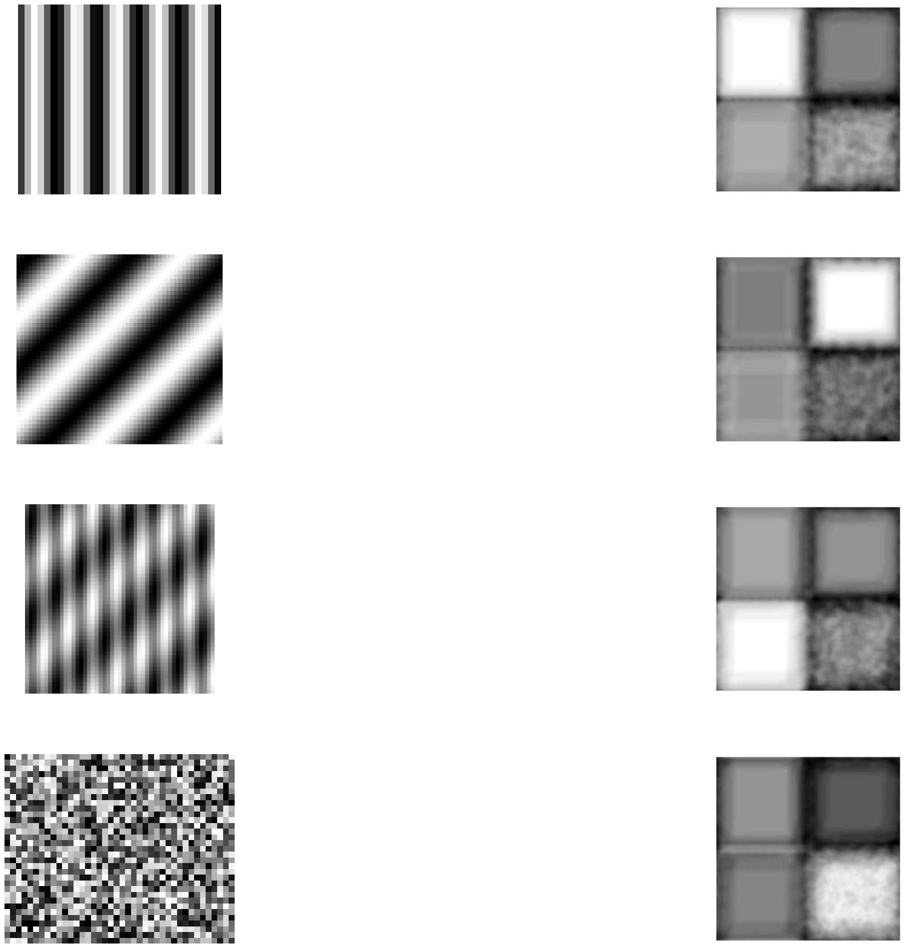


Figure 6: System output for some unambiguous choices of a sample area. The left hand side samples were specified. The right hand side shows the results of the texture analysis. Notice that the random noise can also be treated as a texture!

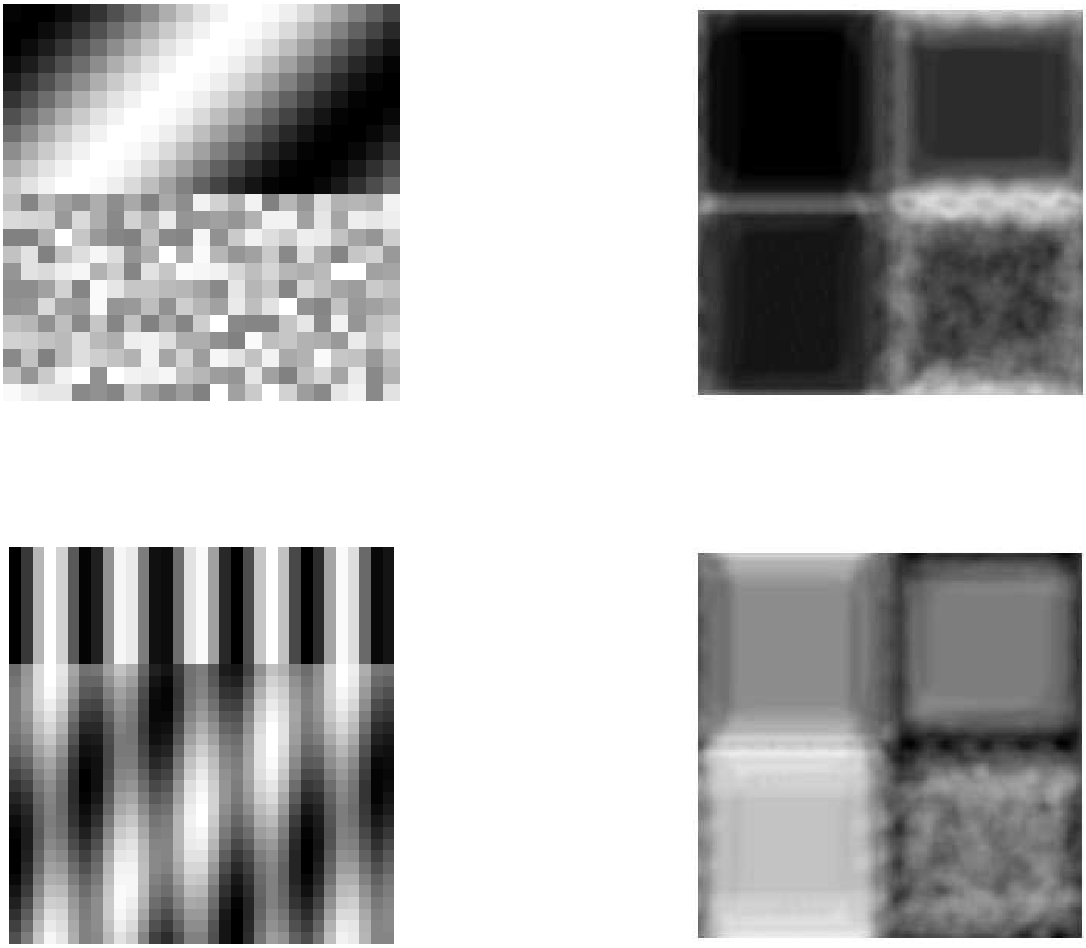


Figure 7: System output for two ambiguous choices of a sample areas. The left hand side samples were specified. The right hand side shows the results of the texture analysis.

6 Texture annotation

In this project we concentrate on finding similar textures in aerial photographs where the user selects a region of interest from the image and the system displays other areas on the image with similar texture contents. Regions of interest can be for example, sea, buildings, trees, etc. We call this application *texture annotation*.

First we apply our Gabor filter bank to the image. Using 4 scales and 6 orientations results in a 24 dimensional vector for every pixel. In order to get a measure of similarity between the texture content at two different locations in the image, we could compare the two representative texture vectors at the corresponding pixels. In practice, this approach does not work well. We found that even within a uniform texture, the texture vectors can be quite different. The reasons for the difference are that some of the scales that we use might be too fine or too coarse for the texture in hand. In addition, noise, changes in the illumination, and texture distortions can have large effects on the vectors. To overcome this problems we take the texture vectors within a neighborhood of the image locations we are comparing, and rather then comparing single vectors, we compare the distributions of texture vectors.

In order to compare the distributions, we first need to quantize the texture vectors so that we can represent the distributions by histograms. The proper quantization should be adaptive so that it will adapt the size of the bins to the actual content of the distributions in the different bands. In this project, we used a uniform quantization with a bin size of $\sigma/2$, where σ is the standard deviation of the filter responses at the different bands. The following considerations should be taken into account when choosing the bin size: Bins that are too small will result is slow computation and might be too sensitive to variations of the filter responses: responses that are similar to each other might end in different bins. On the other end, bins that are too large will fail to express well the distribution. In section 7 we show examples of using different bin sizes.

To measure similarities between the binned distributions we use the *chi-square test*. Let $H = \{h_{\mathbf{i}}\}$ and $K = \{k_{\mathbf{i}}\}$ be two binned histograms where $\mathbf{i}$ represents the bin index. The chi-square statistics is

$$\chi^2 = \sum_{\mathbf{i}} \frac{(h_{\mathbf{i}} - m_{\mathbf{i}})^2}{m_{\mathbf{i}}} ,$$

where $m_{\mathbf{i}} = \frac{h_{\mathbf{i}} + k_{\mathbf{i}}}{2}$. A large value for χ^2 indicates that it is rather unlikely that the two distributions were drawn from the same population. A nice property of the chi-square test is that thresholds on the test results can be derived in a principle way and will be the same for all images.

In our experiments, we compute the histogram of the distribution of texture vectors in the region the user selected. This histogram is normalized to be a proper probability distribution. The resulting histogram is locally compared with small nighbourhoods over the image. We loop on the image pixels and compute a local histogram for the texture vectors in a 9 by 9 block centered at the pixel. After normalizing the histogram, we use the chi-square test to compare it to the histogram of the region that was selected by the user.

The block that is being used to compute the local histograms should not be too small since they will not represent properly the distribution of the texture. On the other hand if the blocks are too large, the affective boundaries between textures will get bigger, and we will end with smaller number of points with distributions that come from a single texture. In section 7 we show examples of using different block sizes.

7 Application to an aerial photograph

In this section we describe experiments that use the methods we presented for aerial photography. Figure 8 shows a typical 512 by 512 aerial photograph. All the experiments in this section where done on this image.

Once we compute the Gabor responses, we can reconstruct the original image reasonably well by simply adding up the responses. Since our Gabor functions only cover one half of the frequency domain, we need also to rotate each function by 180 degrees and add also the responses for these filter. The reconstructed image is shown in figure 9. The major difference between the original and reconstructed images in that the reconstructed image lost the DC component together with the very low frequencies. This results in slow variations of local average intensities.

In the first experiment, we selected a region of coarse textured trees. In figure 10, part (a) shows the selected region. Part (b) shows the result of the Chi-square test, and part (c) is the result after threshold. Notice that textures that represent finer scale tress (smaller and closer) where not found. Similarly, textures that are rotated versions of the selected one where not found too. In the next section we discuss ways to deal with rotation and scale invariance.

In the next experiment, we selected a small region out of a texture of buildings. Figure 11 shows the results. It is interesting to notice that few isolated houses where discovered too.

Figure 8: An aerial photograph.

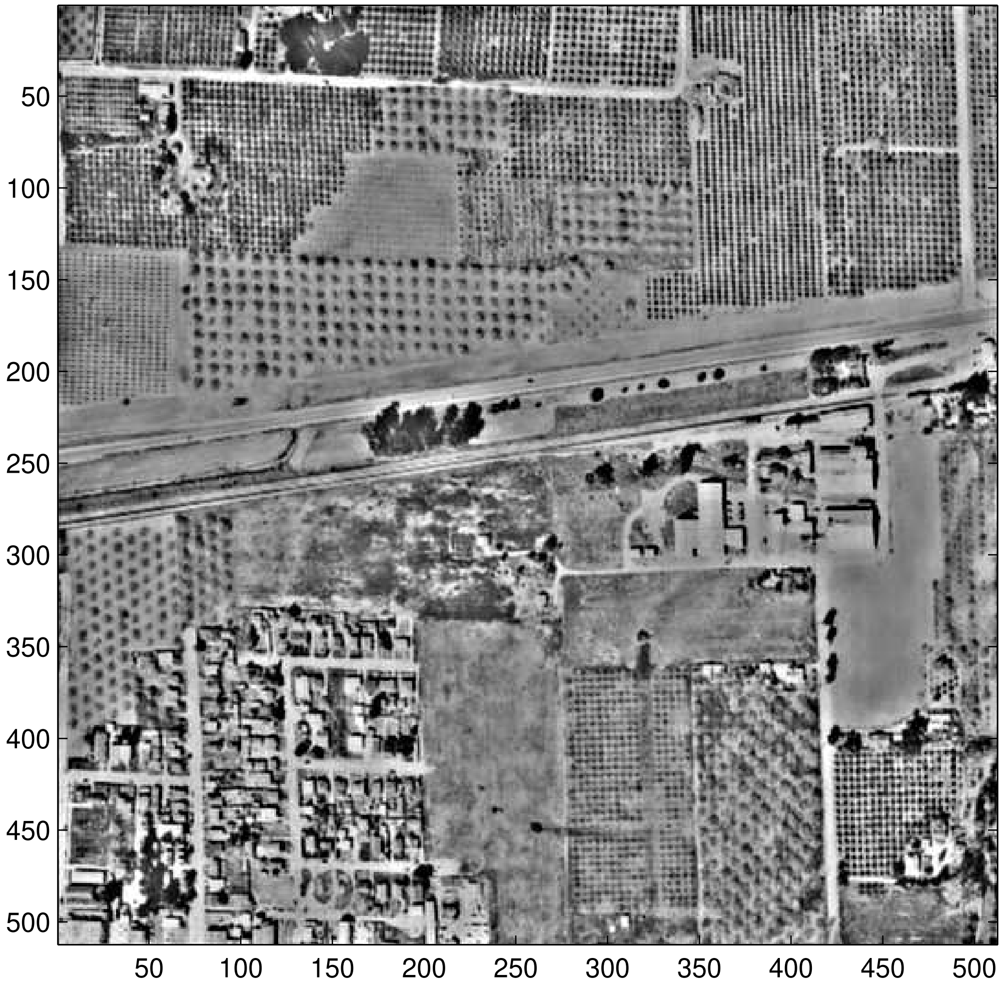
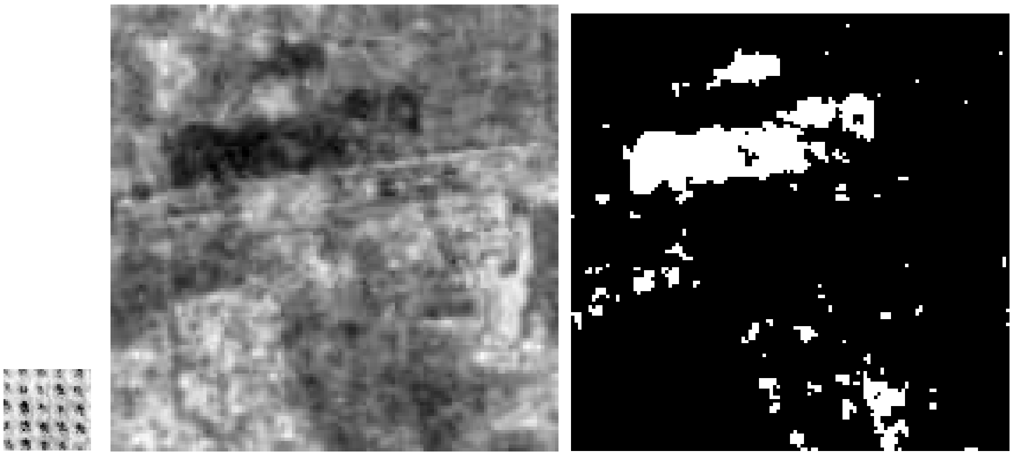
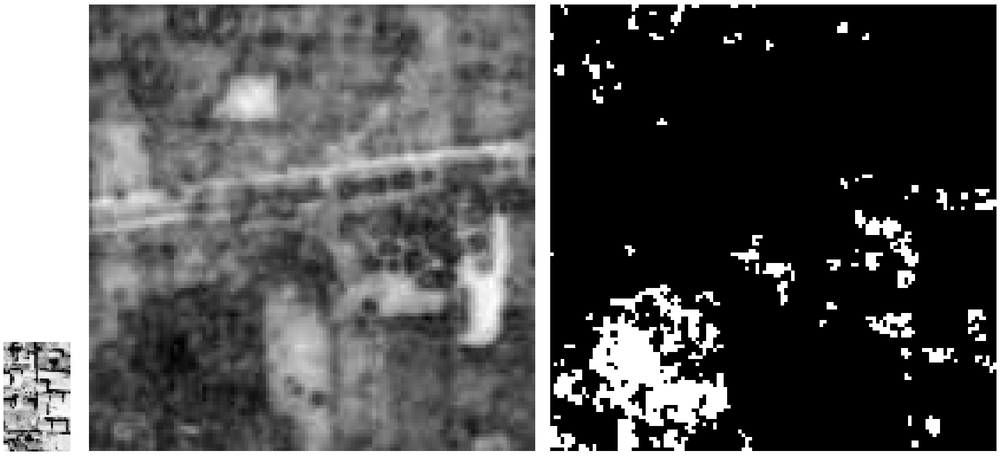


Figure 9: The reconstructed image.



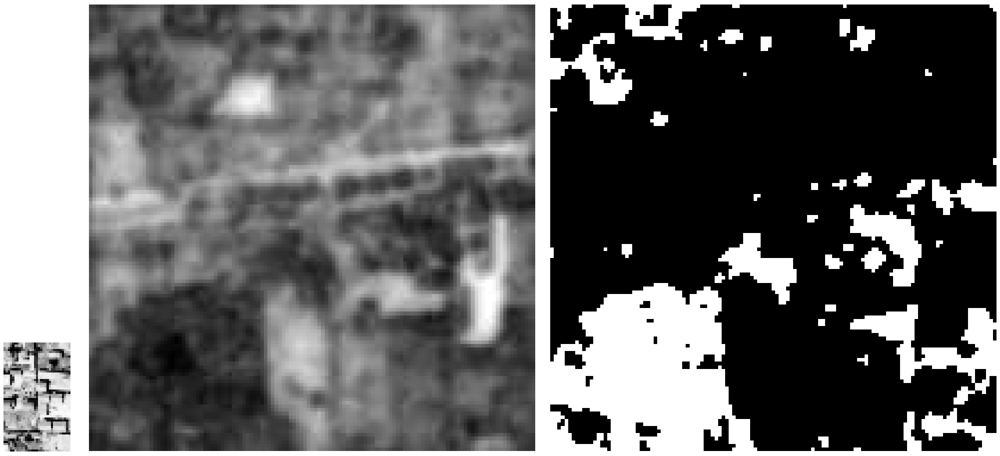
(a) (b) (c)

Figure 10: Looking for trees. (a) Region of interest. (b) Result of the Chi-square test. Darker points correspond to more similar textures. (c) After thresholding.



(a) (b) (c)

Figure 11: Looking for buildings. (a) Region of interest. (b) Result of the Chi-square test. Darker points correspond to more similar textures. (c) After thresholding.



(a) (b) (c)

Figure 12: Using blocksize=15. (a) Region of interest. (b) Result of the Chi-square test. Darker points correspond to more similar textures. (c) After thresholding.

In the previous experiments, we used a binsize of $\sigma/2$ and blocksize of 9 pixels. To see how this parameters change the result, we used the same selection of the buildings texture with 15 by 15 blocks. The results are shown in figure 12

similarly, figure **??** shows the results for using a binsize of $\sigma/8$

8 Further improvement and Conclusion

We saw in section 7 that there are cases where there is a need for a texture similarity measure that is rotation or/and scale invariant. If we rotate a texture or slightly change its scale, it should still be similar to the original texture. One way to achieve this is by a brute force approach: rotate the local histograms by one orientation at a time, and perform the Chi-square test for all orientations (6 in our case). The final result will be the minimum over all tests. Rotating the histogram is done by cyclic shifting it so that values that correspond to orientation i (i.e. were computed by a Gabor function that was tuned to that particular orientation) will be reassigned to orientation $i + 1$, and so forth.

A similar "trick" can be dome for rotation invariance by shifting the filter

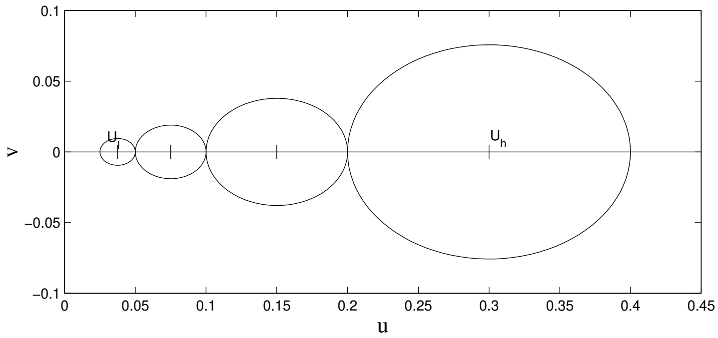


Figure 13: Half-amplitude contours of different scales of filters.

responses up and down in the scale space and taking the minimum over all Chi-square tests.

9 Appendix A

To derive the standard deviation of the Gaussians in the radial direction, consider the elliptical half-amplitudes in figure (13).

Denote the half-width of the narrowest Gaussian along the longer main axis as x. Then the whole frequency range between U_l and U_h can be expressed as

$$
\begin{aligned}
U_h - U_l &= x + 2ax + 2a^2x + \ldots + 2a^{S-2}x + a^{S-1}x \\
&= 2x(\sum_{s=0}^{S-1} a^s) - (1 + a^{S-1})x \\
&= 2x\frac{1-a^S}{1-a} - (1 + a^{S-1})x \\
&= x\frac{1 - a^S - a^{S-1} + a}{1-a}
\end{aligned}
$$

where the geometrical series expansion was used.

Using $U_h = a^{S-1}U_l$ and the half-amplitude of a Gaussian $(a^s x) = \sqrt{2\ln 2}\sigma_u$

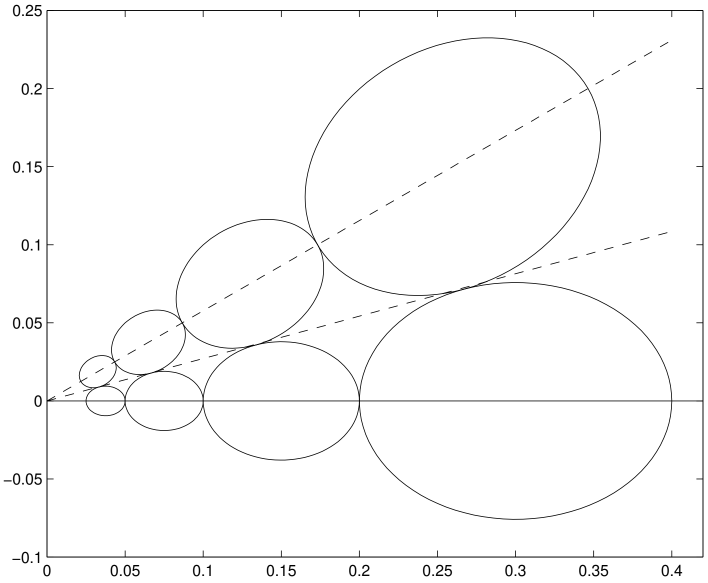


Figure 14: Half-amplitude contours of two neighboring Gaussians for different orientations.

$$
\begin{aligned}
U_l &= x\frac{1-a^S-a^{S-1}+a}{(1-a)(a^{S-1}-1)} \\
&= \frac{a+1}{a-1}x \\
\sigma_u &= \frac{a-1}{(a+1)\sqrt{2\ln 2}}a^s U_l
\end{aligned}
$$

Rather than changing the standard deviations of the different filter wavelets, the filter wavelets are constructed by scaling the variables (equation (3)). This leads to the result for σ_u:

$$\sigma_u = \frac{a-1}{(a+1)\sqrt{2\ln 2}}a^{S-1}U_l \quad = \frac{a-1}{(a+1)\sqrt{2\ln 2}}U_h$$

To derive the half-width σ_v in the other dimension, note that two neighboring ellipses meet on the line halving the angle ϕ (compare figure (14)).

Finding σ_v is thus accomplished by finding the value of σ_v for which the line with gradient $\tan(\frac{\phi}{2})$ is tangent to the ellipse

$$\frac{(U_h+u)^2}{2\ln 2\sigma_u^2}+\frac{v^2}{2\ln 2\sigma_v^2}=1 \tag{7}$$

Inserting $v=\tan(\frac{\phi}{2})u$ into equation (7):

$$\begin{gathered}
\frac{(U_h+u)^2}{2\ln 2\sigma_u^2}+\frac{(\tan(\frac{\phi}{2})u)^2}{2\ln 2\sigma_v^2}=1 \\
u^2(\frac{\tan(\frac{\phi}{2})^2}{\sigma_v^2}+\frac{1}{\sigma_u^2})+\frac{U_h^2+2U_h u}{\sigma_u^2}-2\ln 2=0 \\
u^2+2\frac{\sigma_v^2 U_h}{\sigma_u^2\tan(\phi/2)+\sigma_v^2}u+\frac{(U_h^2-2\ln 2\sigma_u^2)\sigma_v^2}{\sigma_u^2\tan(\phi/2)+\sigma_v^2}=0
\end{gathered}$$

This equation has only one solution (tangent case), if

$$\begin{gathered}
(\frac{\sigma_v^2 U_h}{\sigma_u^2\tan(\phi/2)+\sigma_v^2})^2-\frac{(U_h^2-2\ln 2\sigma_u^2)\sigma_v^2}{\sigma_u^2\tan(\phi/2)+\sigma_v^2}=0 \\
\sigma_v^2 U_h^2=(U_h^2-2\ln 2\sigma_u^2)(\sigma_u^2\tan(\phi/2)+\sigma_v^2) \\
\sigma_v^2(U_h^2-U_h^2+2\ln 2\sigma_u^2)=(U_h^2-2\ln 2\sigma_u^2)\sigma_u^2\tan(\phi/2) \\
\sigma_v=\tan(\phi/2)\sqrt{\frac{U_h^2}{2\ln 2}-\sigma_u^2}
\end{gathered}$$

where $\phi=\frac{\pi}{2K}$ and K is the number of orientations. We used $K=6$ in our system.

References

[1] J. D. Daugman. Complete discrete 2-d Gabor transforms by neural networks for image analysis and compression. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 36:1169–1179, 1988.

[2] David J. Field. Relations between the statistics of natural images and the response properties of cortical cells. *Journal of the Optical Society of America A*, 4(12):2379–2394, December 1987.

[3] D. Gabor. Theory of communication. *The Journal of the Institute of Electrical Engineers, Part III*, 93(21):429–457, January 1946.

[4] T. S. Lee. Image representation using 2d Gabor wavelets. *IEEE Transactions on Pattern Analysis and Machine Intellige*, 18(10):959–971, 1996.

[5] W. Y. Ma and B. S. Manjunath. Texture features and learning similarity. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 425–430, San Francisco, CA, 1996.