An Attention Model and Steerable filters

A course project for

CSC 2523S Computational Vision II

TEH Yee Whye
009753429

December 13, 1999

Contents

1 Introduction

In this project we implemented the selective tuning algorithm of Tsotsos et al [7, 2] to the case when our features are edges. We used steerable filters by developed by Adelson, Freeman and Simoncelli [3, 6] to get edge filters with any desired orientation.

In section 2 we described steerable filters. In section 3 we give a general overview of the selective tuning algorithm. In section 4 we described some ways of speeding up the algorithm. In section 5 we described our test bed and in section 6 we presented some results of our simulations.

2 Steerable filters

Steerable filters were introduced by Adelson, Freeman and Simoncelli [3, 6] as a way of efficiently synthesizing oriented filters of arbitrary orientations from linear combinations of a fixed set of basis filters. The steerable filters they developed are jointly-localizaed in both the space and frequency domain, produces no aliasing, and are self-inverting, although they form an overcomplete basis. We give a simple overview of steerable filters here. For further information please refer to [3, 6].

In this section, we express functions on $\mathbb{R}^2$ in Cartesian coordinates $f(x, y)$ and in polar coordinates : $f(r, \phi)$ where $r = \sqrt{x^2 + y^2}$, $\phi = \tan^{-1}\frac{y}{x}$. Given a function $f(r, \phi)$ we rotate f about the origin by $f^\theta(r, \phi) = f(r, \phi + \theta)$ for all r, ϕ, θ.

We say f can be steered if there are $M \in \mathbb{N}$ and $\theta_j \in \mathbb{R}$ for $1 \leq j \leq M$ such that

$$f^\theta(r, \phi) = \sum_{j=1}^{M} k_j(\theta) f^{\theta_j}(r, \phi) \tag{2.1}$$

The $k_j(\theta)$'s are called the interpolation functions for the basis functions f^{θ_j}'s. Note that the steerability property commutes with linear operations. For example, if I is an image and we convolve f^θ with I, then

$$I * f^\theta = I * \left(\sum_{i=1}^{M} k_j(\theta) f^{\theta_j} \right) = \sum_{i=1}^{M} k_j(\theta) \left(I * f^\theta \right) \tag{2.2}$$

In actual implementation we deal with pixelated versions of images and filters, and not functions on $\mathbb{R}^2$. Luckily, if we view the space of real-valued functions on $\mathbb{R}^2$ as a vector space, then pixelating a function on $\mathbb{R}^2$ can be viewed as a projection onto the subspace of those functions which are zeros everywhere except on the discretization points. This means all the theory developed for steerable filters on $\mathbb{R}^2$ extends to the pixelated case too.

2.1 A simple example

Consider a simple Gaussian function $g(x, y) = e^{-(x^2+y^2)}$. The derivative along the x-direction is

$$g_x(x, y) = -2xg(x, y) \tag{2.3}$$

and the derivative along the y-direction is

$$g_y(x,y) = -2yg(x,y) \tag{2.4}$$

Let $u = (\cos\theta, \sin\theta) \in \mathbb{R}^2$. Then the derivative of f along u is

$$g_u(x,y) = \cos\theta g_x(x,y) + \sin\theta g_y(x,y) \tag{2.5}$$

Noting that $g_x = g_x^0$, $g_y = g_x^{\frac{\pi}{2}}$ and $g_u = g_x^\theta$, we have

$$g_x^\theta(x,y) = \cos\theta g_x^0(x,y) + \sin\theta g_x^{\frac{\pi}{2}}(x,y) \tag{2.6}$$

so $g_x(x,y)$ is steerable.

We can easily generalize this to show that higher order directional derivatives are also steerable.

2.2 Steerability theorems

The first theorem shows that a large class of functions are steerable, and describes the form of the interpolation functions given the basis functions. The proofs of the following theorems can be found in [3].

Theorem 2.1 *f can be steered (i.e. (2.1) holds) if and only if the fourier transform of f wrt ϕ has finite support, i.e.*

$$f(r,\phi) = \sum_{n=-N}^{N} a_n(r)e^{in\phi} \tag{2.7}$$

for some finite $N \in \mathbb{N}$ and $a_n : \mathbb{R}^+ \to \mathbb{R}$ for $-N \le n \le N$. If (2.7) holds, then (2.1) holds with

$$\begin{pmatrix} 1 \\ e^{i\theta} \\ \vdots \\ e^{iN\theta} \end{pmatrix} = \begin{pmatrix} 1 & \dots & 1 \\ e^{i\theta_1} & \dots & e^{i\theta_M} \\ \vdots & & \vdots \\ e^{iN\theta_1} & \dots & e^{iN\theta_M} \end{pmatrix} \begin{pmatrix} k_1(\theta) \\ k_2(\theta) \\ \vdots \\ k_M(\theta) \end{pmatrix} \tag{2.8}$$

If for any n, $a_n(r) = a_{-n}(r) = 0$ for all r then the n^{th} row of the LHS and the matrix of (2.8) should be removed.

Note that the k_j's are independent of $a_n(r)$ for all n, and are only dependent on θ and on the support of f, i.e. n's for which $a_n(r) = 0$ for all r.

The next theorem shows that the second part of the first theorem is optimal.

Theorem 2.2 *If T is the number of non-zero coefficients $a_n(r)$ for $n \ge 0$ in (2.7) and if $f^\theta(r,\phi) = \sum_{j=1}^{M} k_j(\theta)g_j(r,\phi)$ then $M \ge T$.*

The following theorem gives us steerable functions of a more familiar and useful form.

Theorem 2.3 *Let $f(x,y) = W(r)P_N(x,y)$ where $W(r)$ is an arbitrary windowing function where $r = \sqrt{x^2+y^2}$, and $P_N(x,y)$ is an N^{th} order polynomial in x and y whose coefficients may depend on r. Then linear combinations of $2N+1$ basis functions are sufficient to synthesize $f(x,y)$ rotated to any angle θ. The interpolation functions $k_j(\theta)$ are given by (2.8).*

The simple example in section 2.1 satisfies the above theorem. Note that the above theorem is not optimal in the sense that much less basis functions than $2N+1$ is sometimes sufficient.

3 The attention algorithm

Tsotsos et al [7, 2] proposed a model of visual attention where the processing is performed in a distributed manner by neurons forming an image pyramid.

3.1 The algorithm on paper

The pyramid consists of a number of layers of processing elements or units. The first, bottom layer is the input layer and the topmost layer is the output layer.

The algorithm can be separated into three processes. In the interpretation process information regarding the image is propagated up the pyramid. In the attention process, a sequence of units is selected and attended to, starting from the top layer and ending at the bottom layer. In the inhibition process, the attended units' activations are inhibited to allow for the attention process to select another locale to attend to.

3.1.1 Interpretation process

A detailed feature map is first extracted by the first layer from the image. If the feature is intensity the units in the first layer would compute the intensity level at each pixel from the RGB values of the pixel in the image. If edges are the features the first layer interpretive units would convolve the image with an appropriate edge detection kernel. the activities of the units are assumed to be non-negative, and we assume that higher values mean more salient features. Rectification might sometimes be required to assure non-negativity.

Given the activities of the first layer, each unit in the second layer then computes a weighted sum of its inputs. The inputs of each unit is assumed to be a localized patch of units in the layer below. In this project the input regions are rectangular.

Given the activities of the second layer, the third layer units similarly compute weighted sums of their inputs in the second layer. The activities are propagated up to the topmost layer in the same fashion.

3.1.2 Attention process

Starting from the top layer, the attention process selects a "beam" of units from the top layer to the bottom layer to be attended to.

At the top layer, the most salient units are chosen by a winner-take-all (WTA) competition. Note that more than one unit can be selected, and that the selected units need not be contiguous. After selecting the units at the top layer, another WTA competition is held among the inputs of these units. The inputs are those units in the layer below that feed into the selected units. This selects some units at the second to top layer, and the process is repeated for the inputs to these units and so on until the bottom layer. The end result is an inner "pass" zone of attended units, and an outer "inhibit" zone of unattended units that participated in the WTA competitions but did not get selected.

The WTA competition is held by application of an exponential decaying rule. Each unit in the competition inhibits all other units with lower activity than it does. Let the activity of unit i at time step t be a_i^t. The influence of unit j on unit i where $j \neq i$ is defined to be

$$\Delta_{i,j}^t = \begin{cases} w_i a_i^{t-1} - w_j a_j^{t-1} & \text{if } \theta < w_i a_i^{t-1} - w_j a_j^{t-1}, \\ 0 & \text{otherwise.} \end{cases} \tag{3.1}$$

where w_i is a weight reflecting the a priori importance of unit i and $\theta > 0$ is a threshold value. At each time step the activities are updated

$$a_i^t = \left[a_i^{t-1} - \sum_{j \neq i} \Delta_{i,j}^t \right]^+ \tag{3.2}$$

where $[\cdot]^+$ is a rectification function :

$$[x]^+ = \begin{cases} x & \text{if } x \geq 0, \\ 0 & \text{otherwise.} \end{cases} \tag{3.3}$$

The activities of all units with lower saliency than the most salient ones decay exponentially with time. It can be shown that the WTA decay converges within a finite number of steps. Units with positive activities at convergence are the winning units. It can also be shown that these are the units i whose weighted saliencies $(w_i a_i)$ are within θ of the maximum.

3.1.3 Inhibition process

After attending to the units in the pass zone, the winning units in the bottom layer are inhibited (i.e. have their activities set to zero), and the whole pyramid is recomputed and the whole process is repeated to select another set of units to attend to.

3.2 Implemented algorithm

There are a number of differences between the algorithm on paper and the actual implemented algorithm.

3.2.1 The competing elements

Each unit competing in the algorithm on paper is a unit in the interpretive pyramid. In the implemented algorithm, each competing unit is actually a receptive field (RF). A RF is

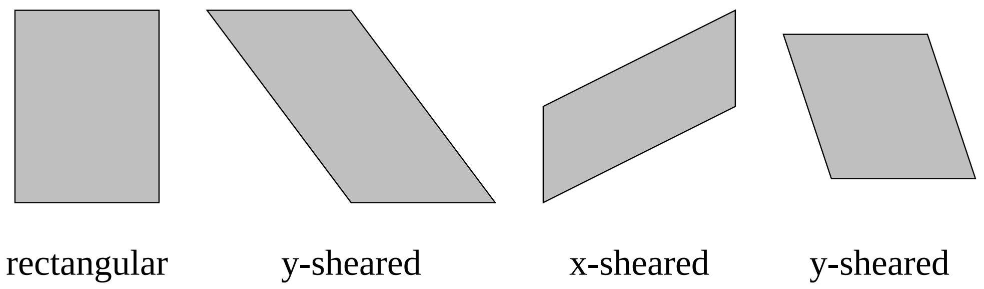

Figure 3.1: The various types of RFs the implemented algorithm deals with.

defined to be a contiguous subset of units in a layer of the pyramid. In particular the implemented algorithm only deal with sheered rectangular RFs. These RFs are either rectangular, or are rectangles sheered either about the x-axis or y-axis, as shown in figure 3.1.

When a RF wins in a layer, a WTA competition is held among the RFs of the layer below. The competing RFs are those which consist of inputs to the units in the winning RF above.

In a layer of size $n \times n$ there are only n^2 units but approximately $\frac{n^4}{4}$ rectagular RFs. So much more computation is required for simulation on a serial computer. Speeding up the computations required is the topic of section 5. In our implementation, all the RFs we deal with in a pyramid are of the same type and are sheered by the same angle. This is because each pyramid deals with only edges of a particular orientation and the only meaningful RFs are those aligned with the edges (see section 4.3).

3.2.2 The saliency of RFs of different sizes

To determine the most salient RF, we first have to define the saliency of the RF given the activities of its units and its size.

One possibility is to let the overall saliency be the weighted average of the activities of the units, with the weights summing to one. Since a weighted average is always less than the maximum, this will not be satisfactory – the competition will always be won by the RF consisting of only the most active unit.

Another possibility is to simply sum up the activities of the units. This is not satisfactory either – the largest RF will always win, with our assumption that the activities are always positive.

Instead we employed the third possibility – letting the saliency be a function $f(n, m, S)$ of the size $n \times m$ of the RF and the average of the activities S. The function has to satisfy some requirements. Firstly it has to prefer larger RFs to smaller ones if the averages are approximately the same, so it should be monotonically increasing in n and m. Secondly it should be sub-linear in nm for n and m large. If it were linear or super-linear, then for sufficiently large n and m, $f(n, m, S)$ would be maximum for the largest RF, namely the whole image. Finally, for smaller values of n and m we prefer larger RFs, so that the gradient of $f(n, m, S)$ with respect to n and m should be large.

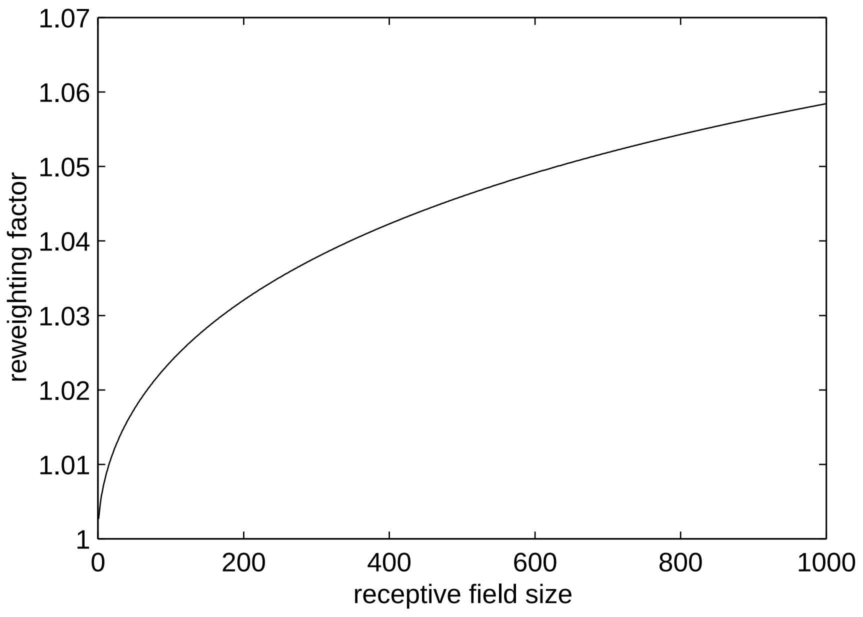

Figure 3.2: Plot of the reweighting factor versus RF size.

Tsotsos et al settled on the function

$$f(n, m, S) = \frac{a+1}{a + b^{-\sqrt{nm}}} S$$

which simply multiplies the average activity by a reweighting factor which depends on the size of the RF. Values of $a \approx 10$ and $b \approx 1.03$ were used by Culhane [2].

However we found that the above reweighting factor does not do very well in our experiments. The plot of the reweighting factor versus the RF size is given in figure 3.2. The RFs that we deal with normally have sizes less than 1000 – since we are dealing with edges, our RFs are normally long (at most 100 pixels in an image with side 256) and thin (say 10 pixels). In this range, the reweighting factor is essentially constant (ranging from 1 to 1.07) and does not affect the choice of RFs by much. In fact simulations show that we normally get small squarish RFs with high average activities.

Another problem is that the above formula does not take into account the different sizes of the layers of the pyramid. In the highest layer, a RF of size 100 might be considered very large, but in the lowest layer it might be considered very small compared to the whole image. Also, the $a+1$ numerator in the factor actually does not affect the WTA competition at all, since it multiplies all RFs by the same amount.

We settled on another reweighting factor instead :

$$f(n, m, N, M, S) = \frac{1}{1 + \exp(-b \frac{\max(n,m)}{\sqrt{NM}})} S \tag{3.4}$$

where the layer is of size $N \times M$. Instead of using the RF size directly, we use the RF size relative to the size of the layer. The RF size is given by $\max(n, m)$ because we are dealing

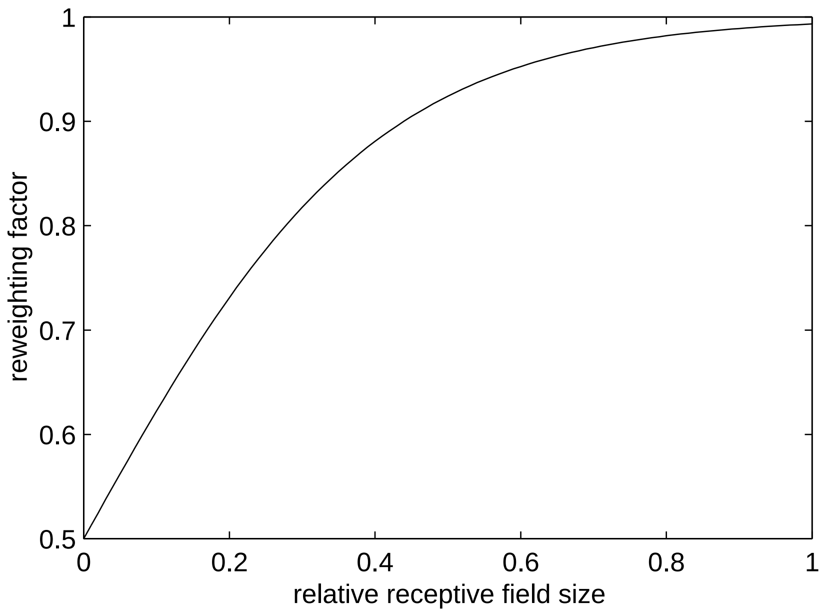

Figure 3.3: Plot of the reweighting factor versus relative RF size.

with edge features and want our RFs to be as long as possible regardless of how wide it is. The layer size is given by $\sqrt{NM}$ with the square root to bring it to the same dimension as $\max(n, m)$. We used a value of 5 for b. The plot of the reweighting factor is given in figure 3.3. Note that it penalizes small RFs by halving the saliency. The relative RF size is $\frac{\max n,m}{\sqrt{NM}}$.

3.2.3 The WTA competition

For efficiency reasons, instead of using a decaying update rule and applying it until the units' activities stabilize, we simply determine the RF with maximum saliency by comparing them directly. We lose the dynamic aspect of the algorithm, but we are only interested in the final result of what RF is chosen.

3.2.4 Multiple pyramids

Edges of different orientations have different pyramids computing them. Each pyramid has a different type and angle of RF. The orientation of the RFs equals the orientation of the edges found (section 4.3). It is assumed that the structure of the pyramids are the same, so that we can identify units in the same location in each pyramid.

To determine the winning RF from the different pyramids, a winning RF at the bottom layer is first picked using the attention process in section 3.1.2 from each pyramid. Then another WTA competition is used to determine which RF is most salient. Although we did not implement it, it is trivial to reweight the RFs' saliencies different depending on our preference on edge orientations.

In simulations, it is found that this algorithm can sometimes get stuck in local minimas. Note that the winning RF of a pyramind need not be the one with the highest saliency.

If the winning RF of a pyramid has low saliency, then it will stay being the winning RF of that pyramid until it has been chosen to be the overall winning RF. Tsotsos et al deals with this as follows. After WTA competitions are held at the same layer of each pyramid, another WTA competition is held to determine the winning RF from among the winners of that layer in each pyramid. This winning RF then determines the corresponding area in the layer below in every pyramid over which the next WTA competitions are to be held. This avoids the local minima problem because the WTA competitions are held over a different area of the layer everytime. However this solution is not intuitively correct. Given that an RF is some pyramid wins, then the WTA competition at the layer below can be viewed as determining the cause of the high saliency of the winning RF. Since the other pyramids did not contribute to the saliency of the RF, it should not be the case that the winning RF determines the area in other pyramids over which the next round of competitions are to be held.

3.2.5 Inhibiting units

Inhibiting a RF involves inhibiting each unit in the RF. With multiple pyramids, the corresponding units in the other pyramids are inhibited as well. Although we did not implement it, it is also trivial to instead inhibit the other pyramids by an amount proportional to how close the orientations of the edges are to the orientation of the edges in the winning pyramid.

4 Speeding up the computation

4.1 Why naive algorithm is slow

Consider finding the most salient RF from among rectangular RFs in an image of size $n \times n$. If we calculate the saliency of each RF separately and compare them, $O(n^6)$ flops will be required. Suppose we have a RF with top left unit (i, j) and bottom right unit (k, l). To determine the saliency of the RF requires $O((k-i)(l-j))$ additions (and multiplications if the saliency is a weighted sum of the unit activities). So the number of flops we require is

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=i+1}^{n}\sum_{l=j+1}^{n} O((k-i)(l-j)) = O(n^6)$$

With $n = 32$, about a billion flops are required. This requires on the order of tens of minutes of computation time on a DEC Alpha, which is unacceptable.

4.2 Computing activity-sums of RFs

Suppose the saliency of a RF is a function of the total activity of the units in the RF. The computational cost is dominated by the cost of summing the actitivies of the units in the RF.

Suppose the activity of unit (i, j) is $a_{i,j}$ where $1 \leq i, j \leq n$. Let $r_{i,j} = \sum_{i'=1}^{i} a_{i',j}$ be the total activity of units to the top of unit (i, j) and $s_{i,j} = \sum_{i'=1}^{i} \sum_{j'=1}^{j} a_{i',j'}$ be the total activity

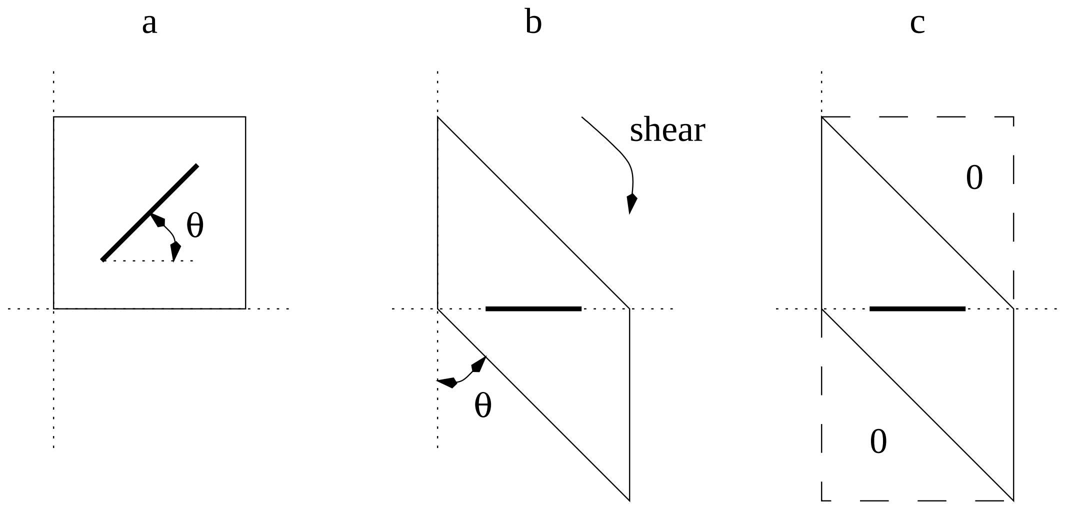

Figure 4.1: Shear and pad the image.

of units to the top and left of unit (i, j). Note that $r_{i,j} = r_{i-1,j} + a_{i,j}$ and $s_{i,j} = s_{i,j-1} + r_{i,j}$ so that the $s_{i,j}$'s can be computed with $2n^2$ flops. Note that the total activity of units in a RF whose top left unit is (i, j) and bottom right unit is (k, l) is

$$s_{k,l} - s_{i-1,l} - s_{k,j-1} + s_{i-1,j-1} \tag{4.1}$$

So only 3 flops is required to determine the total activity of the units in the RF given the $s_{i,j}$'s.

There are $O(n^4)$ RFs in an image of size $n \times n$. This means that $O(n^4)$ flops is sufficient to determine the saliency of all RFs in the image. This is a significant improvement over the naive algorithm. Our implementation based on this algorithm (see section 6) runs in about a tenth of a second on images of size 32×32.

4.3 Handling multiple angled RFs

When dealing with edge features with edges of varying angles, it is important to have RFs with the same angle as the edges. This is because we want the RFs to match the edges we are interested in as close as possible. In this case since we are interested in long edges at some angle, the RFs will have to be at that angle too to contain as long an edge as possible and as little of the areas surrounding the edge as possible.

However the algorithm for computing the activity sums of the RFs in the previous subsection only works for rectangular RFs.

Suppose we are dealing with edges at an angle of θ from the x-axis.

Suppose $-\frac{\pi}{4} \leq \theta \leq \frac{\pi}{4}$ (figure 4.1a). A solution to this problem is to shear the image by an angle of $-\theta$ from the x-axis (figure 4.1b). Note that after transforming the image the edges of interest simply become horizontal, so that rectangular RFs suffice and the activity sum algorithm can be applied. If we now transform back the image, the RFs become sheered rectangular (figure 3.1).

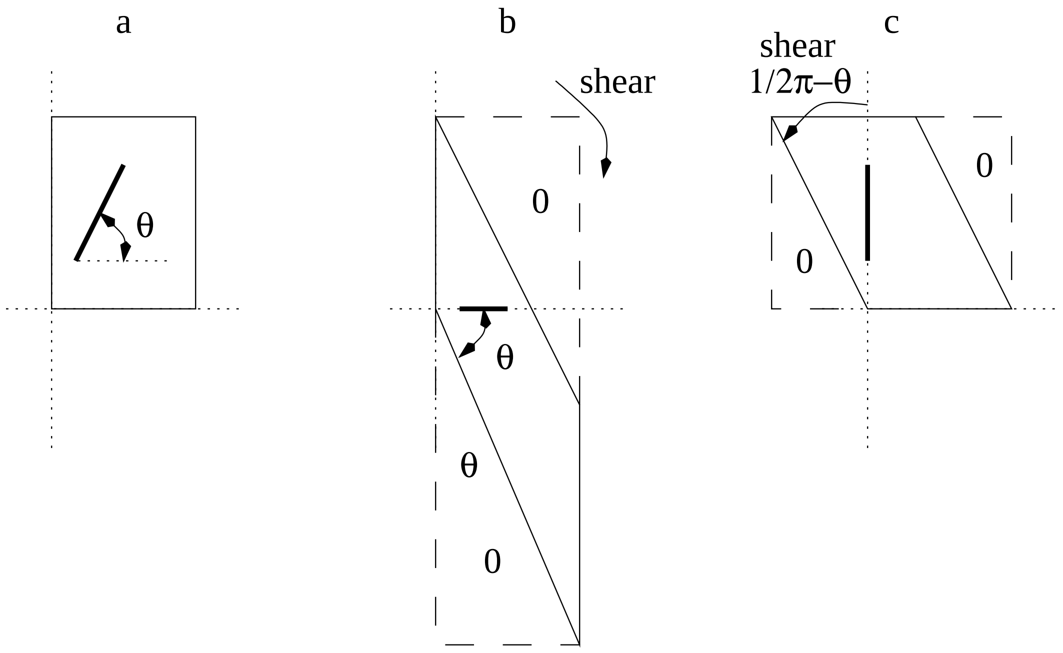

Figure 4.2: Shear and pad the image. In panel b, the resulting image is large. So we shear about the y-axis instead, as in panel c.

Note that shearing a pixelated image is easy – we just need to move up or down each column of pixels by some amount. If instead we had rotated the image clockwise by θ, the pixels of the resulting image would be out of place and some averaging of the pixels would be required. This is complicated and causes aliasing as well.

To simplify our algorithm, we pad the sheered image above and below with 0's so that the image is rectangular again (figure 4.1c). Padding the image with 0's will at most double the size of the image (as $|\theta| < \frac{\pi}{4}$). This is worthwhile given the large amounts of memory at the disposal of modern computers, as it saves a lot of cumbersome programming. If memory is tight the shearing can be done in a logical manner, i.e. the image is not modified, but rather clever indexing is required to achieve the same effect.

Suppose $\frac{\pi}{4} \leq \theta \leq \frac{3\pi}{4}$ (figure 4.2a). Then shearing the image about the x-axis would more than double the size of the resulting image (figure 4.2b). Instead we can shear the image by θ about the y-axis and pad the sheered image on the left and right with 0's (figure 4.2c).

5 Test bed

We decided that it would be more effective implementing the algorithm in MATLAB than in C.

Culhane [1] implemented the attention mechanism in C. It is very efficient, and uses parallel or concurrent processing depending on the computer architecture to make use of computational resources more effectively. However this is achieved at the cost of having a large amount of complex code, making it harder for others to understand and extend his implementation. Also, because a low level language like C was used, a large proportion of the code is actually devoted to implementing low level services like the graphical interface that is used by but not directly related to the attention algorithm.

On the other hand, MATLAB [4] is a high level programming language and environment for scientific and technical computing. Many high level routines required by the algorithm have actually been implemented in MATLAB or related toolboxes, so we only need to concentrate on programming the actual algorithm. MATLAB is an easy language to master, and is also interpreted and easy to interact with online. This means that development using MATLAB is usually much faster than using C. Because of the high level nature of MATLAB, our MATLAB program is also much simpler and easier to understand and extend. Further, when speed is required, C routines (called MEX files) can be written and interfaced with MATLAB, so that we get the best of both worlds.

Our implementation requires MATLAB version 5, as well as the MEX compiler and the image processing toolbox. We used an already available implementation of steerable filters from Simoncelli [5]. It was implemented in MATLAB with certain compute intensive routines which cannot be done efficiently in MATLAB implemented in MEX files. the implementation can be obtained at `http://www.gatsby.ucl.ac.uk/~ywteh/courses/csc2523s.tgz`.

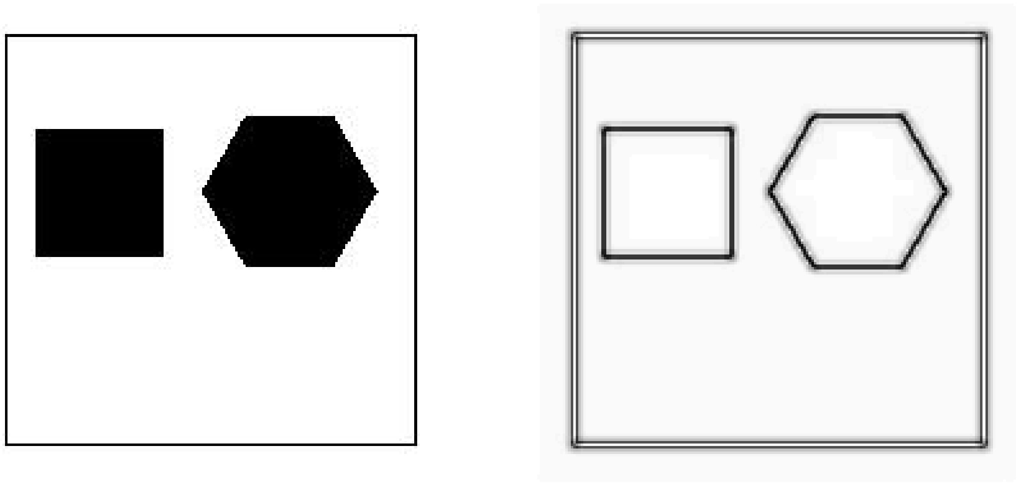

Figure 6.1: Left : the image of a square and a hexagon. Right : the responses of the edge filters. The image is obtained by summing the responses from each edge filter. Black means higher response, and white means no response.

6 Example runs

6.1 A square and a hexagon

For the first test we looked for salient edges in an image consisting of a square and a hexagon both bounded by a square box (figure 6.1, left panel). The image size is 187×187.

The responses of the edge filters are shown at the right of figure 6.1. The edge orientations are $0°, 30°, 45°, 60°, 90°, 120°, 135°$ and $150°$. The pyramid has 3 layers, with sizes 187×187, 80×80 and 30×30 respectively. The filter used to propagate the responses up the pyramid is a simple averaging filter over a 5×5 area.

The sequence of foveations is shown in figure 6.2. Note that longer RFs are perferred and are chosen first. Also, RFs with stronger responses are preferred over weaker ones – the responses from the diagonal edges of the hexagon are stronger than the edge responses of the square, and are chosen before the edges of the square, even though the square has longer edges.

Computing the edge responses and setting up the pyramids required 15.7 seconds on a DEC Alpha. Most of the computations are spent on computing the edge responses. I believe this can be improved, although it is not the concern of this project. Computing the 15 seccards took 16.5 seconds, which is a little over 1 second per seccard.

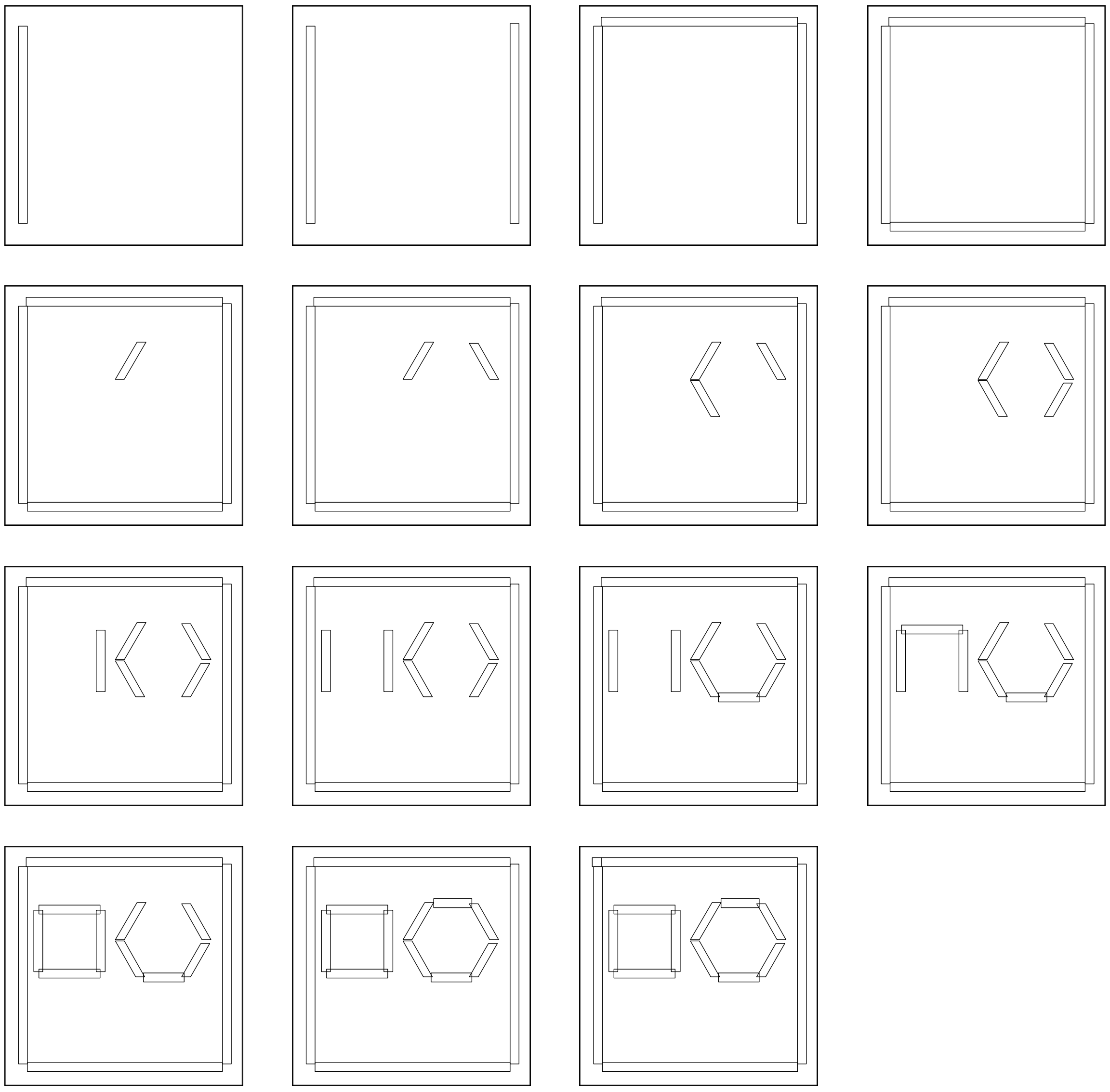

Figure 6.2: The sequence of RFs chosen in order of saliency. Image sequence is from left to right, and then top to bottom.

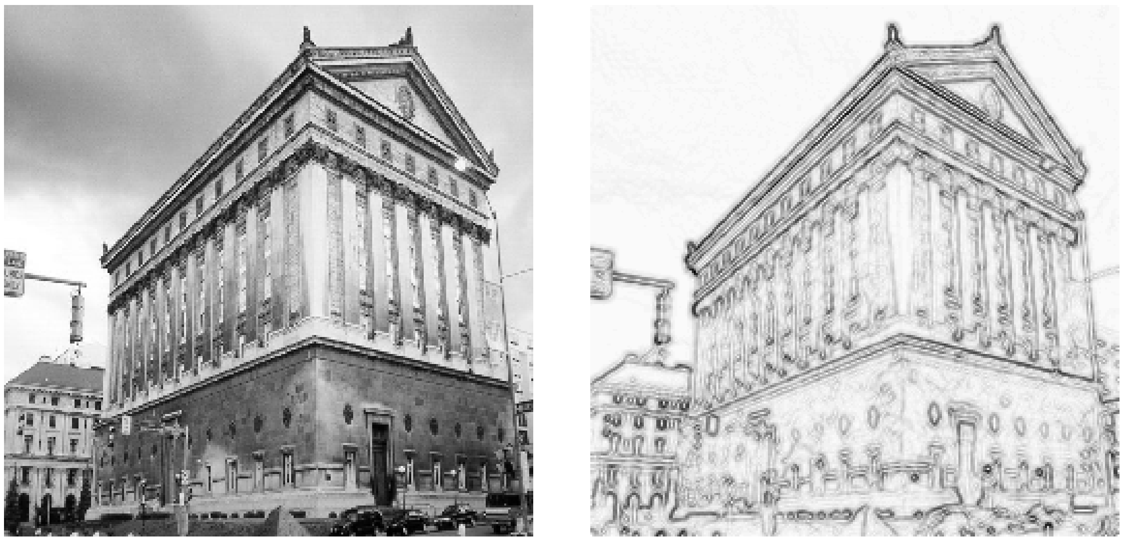

Figure 6.3: Left : the image of the building. Right : the edge responses of the image.

6.2 Building

For the second example we use a photograph of a building (left side of figure 6.3). The image size is 256x256. Again the edge orientations we used are $0°, 30°, 45°, 60°, 90°, 120°, 135°$ and $150°$. The pyramid has 3 layers, with sizes 256×256, 100×100 and 30×30 respectively. the filter used for up propagating the responses is an averaging filter over a 5×5 area. The edge responses are shown at the right of figure 6.3. Computing the edge responses and setting up the pyramids took 16.4 seconds. The 15 foveations took 22.7 seconds, which is around 1.5 seconds per secard.

The sequence of foveations are given in figure 6.4. Note that the last four RFs are very small compared to the first eleven, even though there still are long edges in the image not attended to yet. This is because the orientations of the RFs we chose do not correspond well with the orientations of the edges left in the image. Because of this, the RFs cannot capture the whole edge, but only small parts of the edges, resulting in small RF sizes.

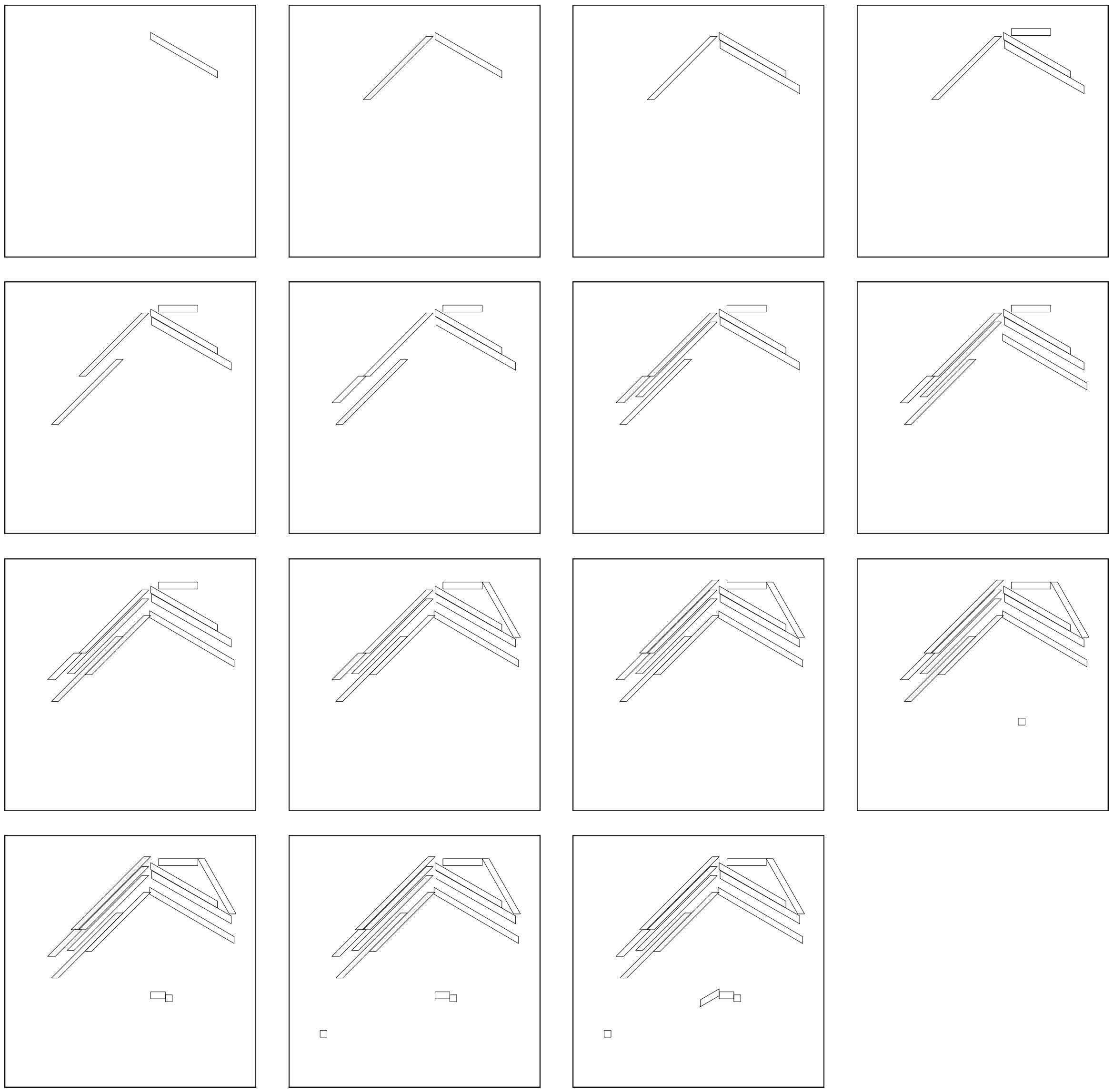

Figure 6.4: The sequence of RFs chosen in order of saliency. Image sequence is from left to right, and then top to bottom.

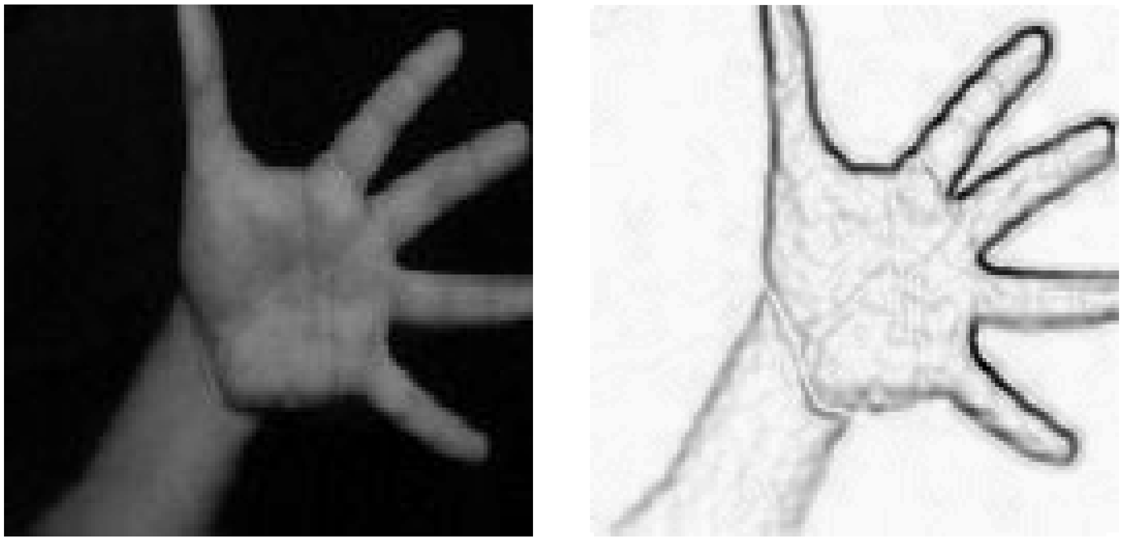

Figure 6.5: Left : the image of the hand. Right : the edge responses of the image.

6.3 Hand

For the last example we used the hand image (`images/hand.pgm`) in Culhane's Attnware distribution [1]. The details are the same as for the previous two examples, except that the images are of size 128×128 and the pyramid layer sizes are 128×128, 64×64 and 32×32. The initialization time is 17.1 seconds, and the time required for 22 seccards is 29.3, which is around 1.3 seconds per seccard. The image and edge response are shown in figure 6.5, and the sequence of winning RFs is shown in figure 6.6.

7 Conclusions

We have implemented the selective tuning algorithm with special consideration for edge features. Our implementation can deal with edges with any orientation by using steerable filters. The implementation can also handle RFs of any orientation, so that long edges can be captured in long and thin RFs. We have found an efficient way of determining the saliency of all RFs given the activities of the units in the layer. We have also found a reweighting factor for RFs of different sizes that gives us better results.

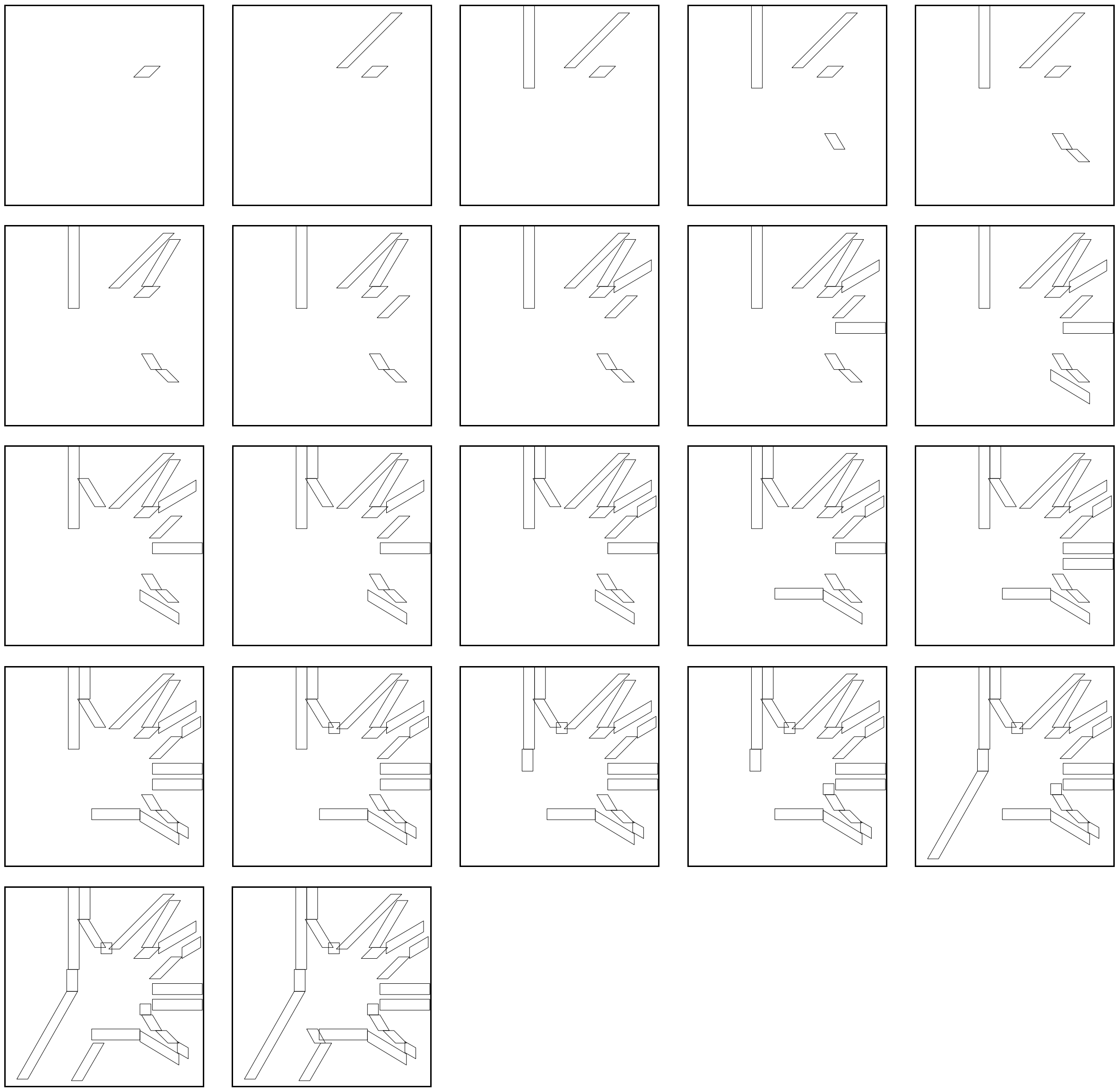

Figure 6.6: The sequence of RFs chosen in order of saliency. Image sequence is from left to right, and then top to bottom.

References

[1] S. M. Culhane. Attention ware.
`http://www.cs.toronto.edu/~culhane/AttentionWare/`.

[2] S. M. Culhane. Implementation of an attentional prototype for early vision. Master's thesis, Department of Computer Science, University of Toronto, 1992.

[3] W. T. Freeman and E. H. Adelson. The design and use of steerable filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(9):891–906, 1991.

[4] MATLAB.
`http://www.mathworks.com/products/matlab/`.

[5] E. P. Simoncelli. The steerable pyramid.
`http://www.cis.upenn.edu/~eero/steerpyr.html`.

[6] E. P. Simoncelli, W. T. Freeman, E. H. Adelson, and D. J. Heeger. Shiftable multi-scale transforms. *IEEE Transactions on Information Theory*, 38(2):587–607, 1992.

[7] J. K. Tsotsos, S. M. Culhane, W. Y. K. Wai, Y. Lai, N. Davis, and F. Nuflo. Modeling visual attention via selective tuning. *Artificial Intelligence*, 78:507–545, 1995.