A Neural Model for Straight Line Detection in the Human Visual Cortex

Theju Jacob[a,1,*], Wesley Snyder[a], Jing Feng[b], HeeSun Choi[b]

[a] *Department of Electrical and Computer Engineering, NC State University, Raleigh, USA*
[b] *Department of Psychology, NC State University, Raleigh, USA*

Abstract

Building a computational model for how the visual cortex identifies objects is a problem that has attracted much attention over the years. Generally, the interest has been in creating models that are translation, rotation, and luminance invariant. In this paper, we utilize the philosophy of Hough Transform to create a model for detecting straight lines under conditions of discontinuity and noise. A neural network that can learn to perform a Hough Transform-like computation in an unsupervised manner is the main takeaway from this work. Performance of the network when presented with straight lines is compared with that of human subjects. Optical illusions like the Poggendorff illusion could potentially find an explanation in the framework of our model.

Keywords: Hough Transform, Straight Line Detection, Primary Visual Cortex, Oja's Rule, Orientation Columns

1. Introduction

The human visual system is vastly superior to man made vision systems in its ability to identify objects. Building a model for how the visual cortex identifies different objects is a problem that has attracted much attention over the years.

*Corresponding author
Email address: theju.jacob@mgh.harvard.edu (Theju Jacob)
URL: www.tjacob.org (Theju Jacob)
[1]Current Affiliation: Massachusetts General Hospital/Harvard Medical School, Boston, USA

Both top-down and bottom-up [1, 42, 4] approaches have been proposed towards this end. Top-down models start at a high-level representation of the incoming visual input, while bottom-up models start with simple features within the input and then move to more complex features.

Oriented lines are one of the first features detected by the visual cortex in a bottom-up approach, see [18, 19]. In this paper, we utilize the philosophy of the Hough Transform to explain how the primary visual cortex could possibly detect straight lines. We describe in detail a neural model that can learn a Hough Transform-like structure in an unsupervised manner. Our focus is on the learning principles and patterns of connectivity of the network of artificial neurons, without getting into creating a detailed biologically realistic network like in [33]. Experiments involving human subjects that add support to our hypothesis are also outlined.

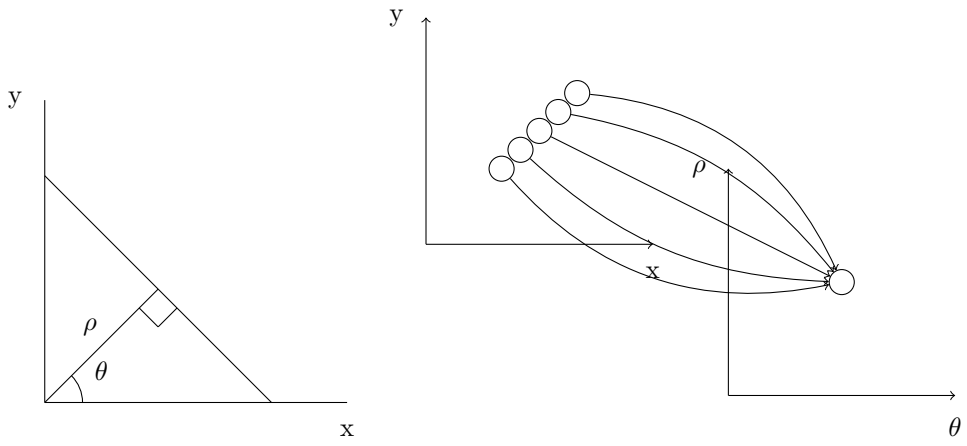
The Hough Transform is a popular feature detection technique in computer vision applicatons, see [32, 7, 39]. The Hough domain, when used to detect straight lines, is characterized by two parameters : 1) the orientation of the line, and 2) the shortest distance from the origin to the line, Fig.1a. The underlying parametric transformation from Cartesian coordinates is represented by the following equation:

$$\rho = x\cos(\theta) + y\sin(\theta) \tag{1}$$

A straight line in the Cartesian domain is thus represented by a single ρ, θ point in the Hough domain.

Practical implementation of the Hough Transform algorithm relies on an accumulator array $\mathcal{A}(\rho,\theta)$ with all possible orientations and distances. The parameters of the line are obtained by looking at the ρ and θ that reflect the maximum increments. The transform is insensitive to clutter and partial occlusion, as we rely on a voting process to determine the parameters of the line.

Neural Networks that can detect straight lines have appeared in literature. [34] discusses a neural network structure for detecting straight lines of various



(a) A line in Cartesian Space (b) Mapping from Cartesian Space to Hough Space

Figure 1: Parameters of a straight line in the Hough space.

orientations. However, no learning happens in the network. The network replicates input lines during detection, and no accumulation is involved. A spiking neural network model that implements the Hough Transform is discussed in [43]. Once again, no learning is involved as the weights for the network connections are derived directly from the Hough Transform formulation.

A neural network capable of achieving a Hough Transform parameterization is discussed in [2]; however, the accumulator array concept is not utilized and the network needs to search for the weight vectors representing straight lines over multiple iterations, every time. By comparison, after training, our network detects the lines in a two-step computation. [28] also discusses a neural network for learning the Hough Transform parameter space. Their training data includes parameter values and input image values, and the network is trained using backpropagation. Our network is unsupervised, and uses only the incoming input image as training data. The problem of finding the maxima in a Hough Transform parameter space using biologically inspired ideas as opposed to computing the Hough Transform itself is addressed in [5].

Neural activity models that lead to orientation-filter like characteristics have

also appeared literature previously. For example, learning a sparse code for images as a model for neural activity is discussed in [36, 9]. Predictive coding as an explanation for visual processing in the cortex is discussed in [38]. Information maximization as a goal of sensory coding is discussed in [3, 31, 30]. A self-organizing map architecture capable of extracting features similar to that extracted in the early stages of visual processing is discussed in [25].

Our network differs from previous networks in that we explain not only how neurons can learn to be orientation sensitive, but also how the neurons in subsequent layers can pool information from orientation sensitive cells and learn to detect entire lines even in presence of discontinuity and noise. Orientation sensitivity becomes evident in two aspects of our network after training: 1) In the connections developed by neurons in the same layer 2) In connections between neurons in different layers. The former contributes to an associative memory like behavior [26, 27], and a derivation of our learning rule that enables it is discussed in our work [24].

The latter demonstrates itself in the receptive fields of neurons in the receiving layer. This in turn causes a single neuron or a small group of neurons in the later layer to represent an entire line in the previous layer, much like a mapping from Cartesian space to the Hough Space. The idea of pooling of neuronal outputs in subsequent layers has been explored previously [29, 12]. Further, the local nature of connections employed by our network leads to a topological mapping of data between subsequent layers - that is, neighboring lines in the input space would be represented by neighboring neurons in our network. Local connections between neurons and topological mapping are features that are present in the cortex as well.

After training, the computation done by our network is comparable to that of the standard model of object recognition[40]. In the standard model, a battery of filters is fixed as S1 cells, and their outputs are combined in C1 cells. After training, the receptive fields of our network neurons become Gabor-like, like those of S1 cells, while the winner-neuron computation performed by our network becomes a computation comparable to that of a C1 cell operation.

Parallels may be drawn between our network architecture and that of Boltzmann/ Restricted-Boltzmann machines [16, 17]. However, the learning algorithms are very different — for Boltzmann machines, the parameter of interest is the global energy of the network. In our network, the interest of the learning rule is in the normalization of incoming weights to individual neurons. Boltzmann machines aim to model the input distribution, while our network learns input features and pool them, retaining spatial relationships.

It is well established from experiments by Hubel and Wiesel, see [18, 19], that the primary visual cortex contains cells that are sensitive to the orientation of the visual stimuli. It is further known that the orientation sensitive cells aggregrate into columns called orientation columns, and that the sensitivity of the columns themselves vary in a sequential manner. We also know that the cells of the visual cortex are organized into retinotopic maps. That is, neighboring areas in the visual fields map to neighboring areas in the cortex itself, [20, 21, 22, 23]. Parallels may thus be drawn between architectural features of Hubel and Wiesel's 'ice-cube model' of the cortex and that of the accumulator array.

An accumulator like structure could potentially explain the phenomenon of log polar transformation observed in the cortex [41, 6]. For this discussion, assume that the center of an input image acts as the origin, and that the origin corresponds to the fovea. Suppose there is a straight line in the input image. Every point along that line will increment a set of cells in the accumulator array, and a maximum is obtained in a single cell or set of cells. The cell or group of cells with the maximum value would indicate the position, or the distance of the line from the origin, ρ.

Now consider a particular point p in the input image. Let the point be at a distance d from the origin. Any line which passes through p will have a ρ which is less than or equal to d, i.e., $\rho \leq d$. Since p will contribute to the accumulator array every time a line passes through p, the cells excited by point p will indicate lines with ρ from 0 to d. That is, p should excite a total of d number of cells in the accumulator array. Since a point closer to the origin will have a d that is smaller than that of a point which is farther away from the origin, this in turn

implies that a point closer to the origin will excite fewer cells when compared to a point farther away from the origin. Hence all points which lie at the same distance from the origin will excite the same *number* of cells in the accumulator. This is in agreement with the log polar transform idea.

In this paper, we examine how artificial neurons can self organize into accumulator arrays in an unsupervised manner. Starting with neurons that learn to become orientation sensitive, we show how subsequent layer of neurons can learn to accumulate information from the previous layer and detect the position of a line. We adopt from Hebbian Learning, see [15], to do this. The performance of our network is then compared with that standard Hough Transform implemenations and that of human subjects.

2. Architecture and Methods

The key ideas addressed in this paper can be stated as follows, of which Hypothesis 1 is addressed in this section:

- **Hypothesis 1:** Artificial neurons can learn to do the accumulator-array like computation in an unsupervised manner.
- **Hypothesis 2:** The trained accumulator-array network performance is comparable to that of the human visual system.

Suppose we are interested in 4 of all the possible orientations - θ_1, θ_2, θ_3, θ_4. When a point in the visual field is illuminated, all 4 of the orientation sensitive cells at the point receive stimulus. They feed into all of the corresponding cells in the accumulator array. When a line of a given orientation falls in the visual field, a collinear set of points get excited. The cells in the corresponding orientation column respond strongly. As points along a line of given orientation would have the same ρ, they all excite a single accumulator cell very strongly. This can be represented by the following equation where $\mathcal{A}(\rho,\theta)$ is the accumulator cell, and

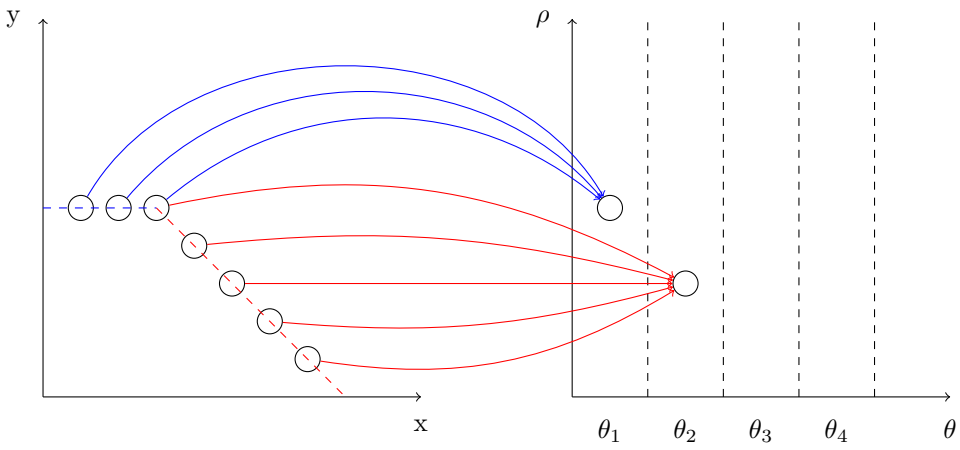


Figure 2: Connections between pixels in the image plane and the cells in the accumulator. For clarity, only the connections which give rise to the two strongest responses in the accumulator array are shown.

N is the number of points along the line.

$$\mathcal{A}(\rho, \theta) = \sum_{i=1}^{N} \delta(\rho - (x_i cos(\theta) + y_i sin(\theta))) \tag{2}$$

The case with multiple lines is demonstrated in Fig.2. The θ_2 receptors of the points lying along the red line feed into a single cell in the θ_2 column. The θ_1 receptors of the points lying along the blue line feed into a single cell in the θ_1 column. The point at the junction of the two lines feed into both columns.

The network described clearly leads to an accumulator array. The important question is: can such a network be learned? We next show how the neurons that receive their inputs from the orientation sensitive cells could learn the position of the given line. That is, we are interested in learning the connections from the θ sensitive cells to the correponding ρ sensitive cells. In the case of stimuli consisting of straight lines, this combination of orientation and position amounts to an accumulator array.

Network Architecture:

The network of interest consists of two groups of neurons, both arranged in 2-dimensional lattices - let us call them the input array and the output array.

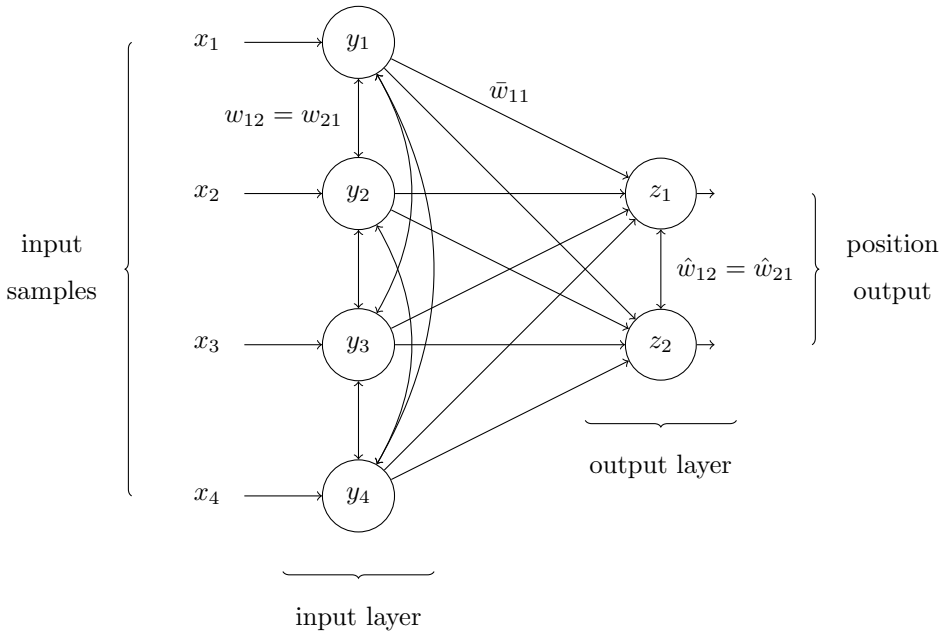


Figure 3: Example Architecture

An example is shown in Fig.3. The input array neurons receive signals from the input image, and also from neighboring neurons within the same array. These cells can be thought of as the orientation-sensitive simple cells in the cortex.

The output array neurons receive signals from the input array as well as from other neurons within the same array. The weights from the input array to the output array correspond to the mapping from Cartesian space to Hough space as shown in Fig. 1b. Every output neuron is fully connected to a neighborhood of underlying neurons from the input array as well as to neighboring neurons in the same array. They represent what will be the position sensitive cells, after training. The neuron with the maximum output will indicate the location of the input in the given image. For example, after training, when a test image with a horizontal line at the second row is given as input to the network, the maximum in the output array will occur at the neuron at the center of the second row. Assuming the origin to lie in the center of the output array, the neuron with the maximum indicates the shortest distance from the origin, which is also ρ.

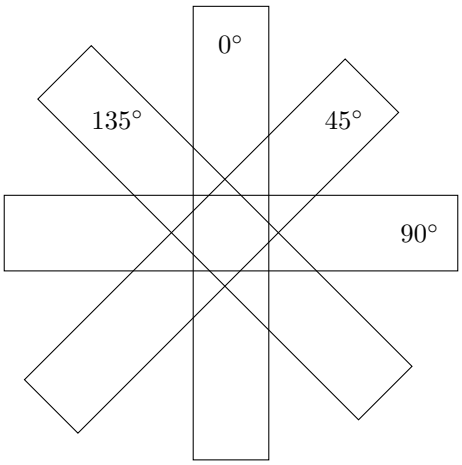


Figure 4: Output arrays of the 4 networks, arranged as a pinwheel.

The interarray connection strengths are assumed to be symmetric - that is, the strength of the connection going from neuron a to b within the same array is assumed to be the same as that going from neuron b to a .

We built a network as shown in Fig.3 for every orientation under consideration. After training, the output array of any given network will indicate the position of a line of corresponding orientation. This leads to an accumulator array like structure.

The connections in our network, both within the layer and between the layers, are restricted to neighborhoods - that is, all of the connections are local. There are no long range connections in the network. Every input layer neuron receives input from a single pixel in the input, while every output layer neuron receives neurons from a neighborhood of input neurons. The output layer therefore has a larger receptive field than the input layer neurons. If additional layers were to be built, the neurons in the later layers can be thought of as having a larger receptive field than the neurons in the preceeding layers.

The ice-cube model of the cortex and how that fits the accumulator idea was described previously. Another prevalent model of the cortex is the pin-wheel model [8]. In the pin-wheel model, orientation sensitive cells are arranged in a pin-wheel like fashion around a center. While traversing the pin-wheel structure,

either in clockwise or in anti-clockwise direction, the orientations to which the encountered cells are sensitive to, vary smoothly.

The network presented in this paper fits this construct as well. Consider the second array of 4 trained networks. When overlapped with centers coinciding, they present a pin-wheel like structure, see Fig. 4. The orientations to which the cells are sensitive vary smoothly in both clockwise and counter-clockwise direction in the structure.

Network Equations:

Oja's rule from [35] is used to modify the weights between the input and output array. Ideas borrowed from [26, 27] and [11, 9, 10] are used to modify weights between neurons in the same array.

If x denotes the external input, y denotes the input array and z denotes the output array, the output at each array is computed as follows:

$$\begin{aligned} y_i(t) &= f(\sum_j w_{ij} y_j(t - \Delta t) + x_i) \\ z_i(t) &= f(\sum_j \hat{w}_{ij} z_j(t - \Delta t) + \sum_j \bar{w}_{ij} y_i) \end{aligned} \tag{3}$$

In (3), f refers to a smooth thresholding function like the sigmoid or the hyperbolic tangent. Δt indicates increments in time. The neuron outputs decay as follows, where $decay < 1$:

$$\begin{aligned} y_i(t + \Delta t) &= decay * y_i(t) \\ z_i(t + \Delta t) &= decay * z_i(t) \end{aligned} \tag{4}$$

The decay parameter was chosen by trial and error.

While updating weights, the aim is to strengthen the connections between neurons that fire together, while resulting in a stable network. For weights between the input and output arrays, Oja's rule is used. For neurons in the same array, the weights increase if the outputs of the neurons correlate, but with a restraining factor, β.

The weights between the input and output array are updated as follows, where α is the learning parameter:

$$\bar{w}_{ij} = \bar{w}_{ij} + \alpha z_i(t)(y_j(t) - z_i(t)\bar{w}_{ij}) \tag{5}$$

The weights between the neurons in the output array are updated as follows, where β is a constant:

$$\hat{w}_{ij} = \hat{w}_{ij} + \alpha(z_i(t)z_j(t) - \beta\hat{w}_{ji}) \tag{6}$$

A derivation of the above learning rule is given in our work [24].

The weights between neurons in the input array are similarly updated. The learning parameter decreases with time in an exponential manner. That is, if α_o denotes the initial value of the learning parameter:

$$\alpha(t) = \alpha_o \exp(-\frac{t}{\tau}) \tag{7}$$

The input array is assumed to have the same size as that of the incoming input image. The output array has a reduced size along one dimension - if the array is of size $n \times m$, $m << n$. This is noticeable in the supplementary results. The reduced size for the output array is justified as follows: Suppose an output array of the same size as that of the input array is used. After training with horizontal lines, for example, it can be found that only a few neurons in each row of the output array are needed to represent the input samples. We can hence do away with some of the columns in the output array, thereby reducing the dimension of the output array.

Such a two-array network was built for every orientation of interest. Each network was trained separately. The training samples consist of lines of the chosen orientation, located at all possible locations in the input image.

This network differs from previous works [26, 9, 13] in that neurons within the same array are not assumed to inhibit each other. Our premise is that of Hebb: if two neurons within the same array fire together repeatedly, the firing of one of them alone should cause some excitation in the other.

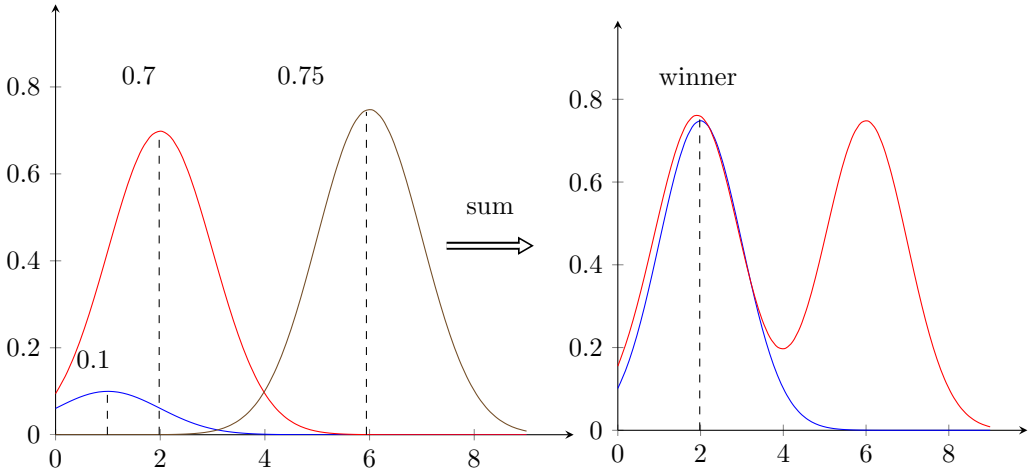


Figure 5: Winner computation example. Suppose we have an array of neurons. Let the outputs for neurons 1,2, and 6 be 0.1,0.7, and 0.75 respectively. All other neuron outputs are presumed 0. Conventional wisdom would declare neuron 6 as the winner. In our computation, the neuron outputs are multiplied by a Gaussian centered at the neuron locations (represented by red, blue, and grey curves on the left) added together (result represented by red curve on the right). The resulting maximum is now located at neuron 2. Neuron 2 is declared the winner (represented by blue curve on the right).

After training, the networks were tested using images of full lines as well as line segments under various conditions. The peak point or the winning neuron in the output array was determined as follows: every neuron output is multiplied by a two dimensional Gaussian function, and are then summed together. The neuron located at the location of the maximum value is declared the winner, see Fig.5. If, for example, i^* is the winning index in the output array,

$$\begin{aligned} g_i(t) &= \sum_j z_j(t)\tilde{r}_{ij}(t) \\ i^*(t) &= \underset{i}{\operatorname{argmax}}(g_i(t)) \end{aligned} \tag{8}$$

The $\tilde{r}_{ij}$ term represents a 2 dimensional Gaussian neighborhood. This winning neighborhood approach leads to more robust results when compared to the conventional approach of declaring the neuron with maximum output as winner.

Results:

Network Training

The incoming weights to the output array neurons were initialized to small random numbers normalized to 1. The weights between neurons in the same array were initialized to 0. The starting value of the learning parameter α was 0.1 and the decay parameter was 0.5. Every output array neuron received inputs from a 4x4 array of neighbouring input array neurons.

The training set consisted of images with lines of a given orientation located at all posible positions, one line in any given input. The network was trained with the images chosen at random, one array at a time - that is, the input array training was completed before the output array was started. The number of iterations were to the order of about 5000 for every array. The choice of the number of iterations was driven by the rate of decay of the learning parameter - once the learning parameter reached a point where the neurons had little effect on each other, training was stopped.

A set of result images showing line completion/detection can be found in the supplementary material.

The network can learn from noisy data as well. In practice the rate of decay of the learning parameter may need to be reduced for noisy data. The layer by layer training of the network becomes important when the test images contain orientations a particular network was not trained on. For example, a network for horizontal lines, when not trained layer by layer, might give a strong maxima in the second layer for a 15° line. This is because the second layer connections are not as clear cut as they would be if they were exposed to the inputs from the first layer after the latter was trained. Formulation of the network operation in the context of achieving a global minima as opposed to a local minima will be examined in a future paper.

Performance Comparison

Next compared are the results from 2 different implementations of the Hough Transform, and the neural network trained on straight lines. By this comparison, we seek to establish that the trained network exhibits the same behavior as the common versions of the Hough Transform. These results are later compared

with results obtained from human subjects. The 2 implementations of the Hough Transform considered are -

1. The Conventional Hough Transform (CT)
2. A Hough Transform implementation using Gabor Filters (GT)

Pseudo code representations for the two implementations can be found in the supplementary material.

CT has been discussed already. In GT, a Gabor filter for every θ is applied at every pixel. The following equations represent the operation of a Gabor filter, where x and y refers to the coordinates in the x-y plane and θ is the orientation of interest.

$$\begin{aligned} x' &= xcos(\theta) + ysin(\theta) \\ y' &= -xsin(\theta) + ycos(\theta) \\ gf(x, y, \theta) &= exp(-\frac{(x')^2 + \gamma^2(y')^2}{2\sigma^2}) \cos(\frac{2\pi x'}{\lambda}) \end{aligned} \tag{9}$$

The value from the Gabor filter is then used to increment the corresponding cell in the accumulator array. The following values were used in (9): σ = 1,γ =1, λ =4.

In all the implementations of the Hough Transform, sampling interval for θ is 15°, while for the neural network, it is 45°. All possible ρ values are covered.

Three conditions are tested:

1) Difference in θ 2) Difference in ρ and 3) Presence of Noise.

The parameter being plotted is the difference between the highest peak and the second highest peak in the accumulator, as explained in Fig.6. Sample test images are shown in Fig.7,8,9.

For testing the difference in θ, we presented images with two line segments, which were aligned at first. The angle between the two line segments were gradually increased. The difference between the highest and the second highest value in the accumulator was plotted. The difference gradually decreased as the lines became more distinct. Fig.10 show the results. The curves follow a similar trend in all cases.

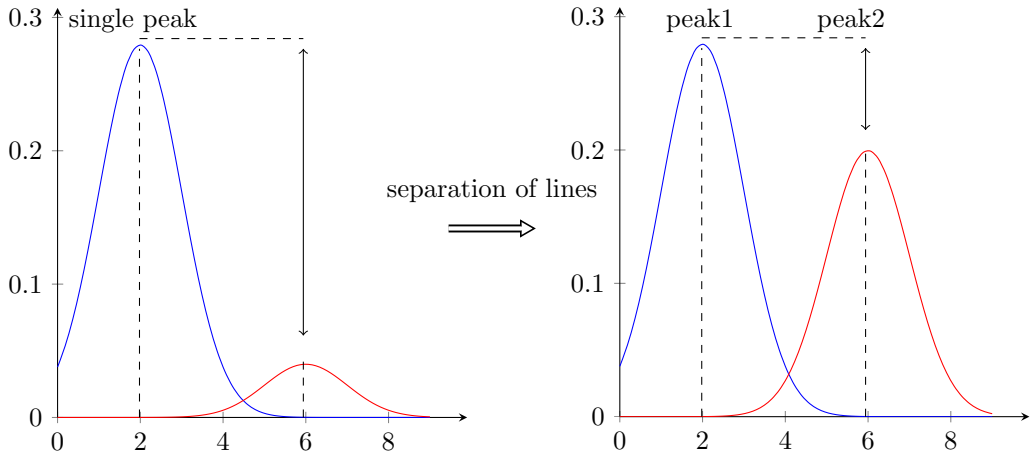


Figure 6: Separation of peaks example. When two line segments are aligned, only one peak is observed in the accumulator. As the line segments separate, the peaks become more distinct. In the case of addition of noise, more peaks of similar amplitude appears in the accumulator as the standard deviation of the added noise increases. As the peaks become more distinct, the difference between the highest peak in the accumulator and the second highest peak in the accumulator diminishes.

For testing the difference in ρ, we presented images with two line segments, which were aligned at first. One of the line segments was then displaced. The difference between the highest and the second highest value in the accumulator was plotted. The difference gradually decreased as the lines became more distinct. Fig.11 show the results.

For testing the performance in the presence of noise, additive white gaussian noise was added to the test images with two aligned line segments. The standard deviation of the noise was increased. As standard deviation was increased from zero, multiple peaks appeared in the accumulator array for all 4 implementations. The results are shown in Fig.12.

The trained neural networks and all of the Hough transform implementations follow a similar trend in all tests. Details of extension of the network to all orientations can be found in the supplementary material.

(a) 0° (b) 15°

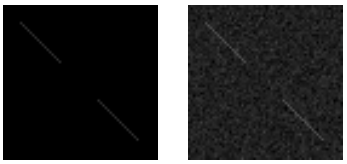
(c) 30° (d) 45°

Figure 7: Variation in angle test images. The angle between the two line segments increase as we move from (a) to (d).

(a) 0 pixels (b) 4 pixels

(c) 6 pixels (d) 10 pixels

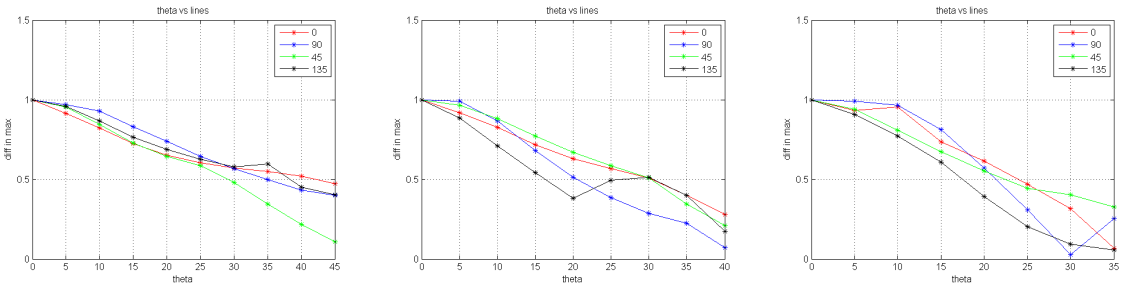
Figure 8: Variation in distance test images. The separation in pixels between the two lines increase from (a) to (d).



(a) $\sigma = 0$ (b) $\sigma = 10$

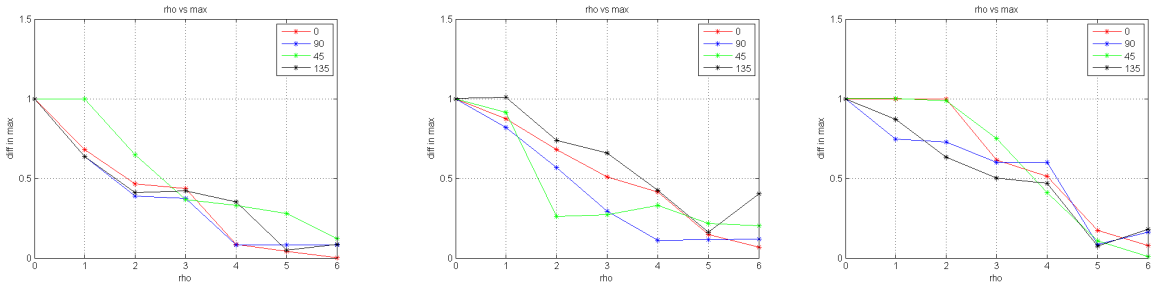
(c) $\sigma = 20$ (d) $\sigma = 30$

Figure 9: Variation in noise test images. The standard deviation σ of the added noise increase from (a) to (d).



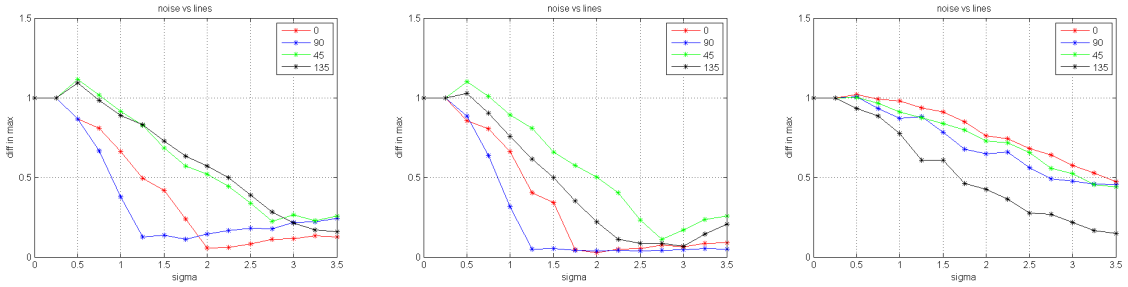
(a) CT (b) GT (c) NN

Figure 10: Decay of difference in peaks with difference in angle between two line segments for the 3 different implementations. As expected, as the line segments became more distinct, the peaks obtained from the units became more distinct, driving their difference to zero.



(a) CT (b) GT (c) NN

Figure 11: Decay of difference in peaks with difference in distance between two line segments for the 3 different implementations. Once again, all implementations showed a similar trend in behavior as the lines became more distinct.



(a) CT (b) GT (c) NN

Figure 12: Decay of difference in peaks with addition of noise for the 3 different implementations. With added noise, multiple peaks appeared in the cells, making the unit with the maximum value indistinguishable from others.

3. Human Experiments

An important argument in this paper as stated in Hypothesis 2 is that the human visual system response to straight lines is comparable to that of an accumulator-array network. That is, straight line detection is a hard-wired process, as opposed to a process involving the higher areas of the visual cortex. From the human experiments, we expected additional support by means of a number of results.

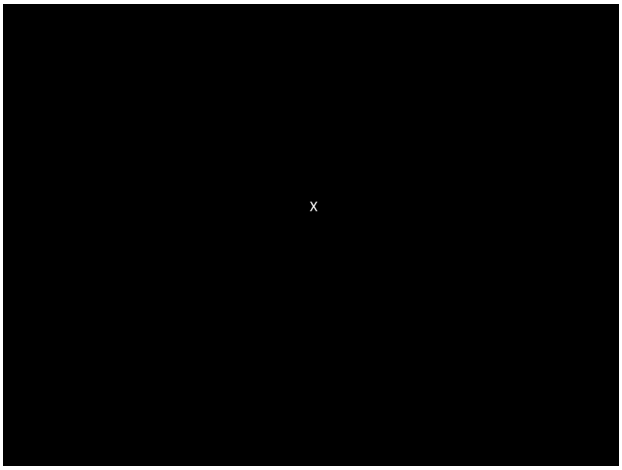
First, we expected the peripheral and foveal responses to straight lines in human subjects to match, in terms of latency and accuracy. If peripheral and foveal responses match, it would imply that straight line detection does not require the full attention of the observer, and hence is a process happening in the early stages of the visual cortex. It would add weight to the "hard-wired" idea.

We also expected the observers to detect lines much faster than contours, aligned or non-aligned. If straight lines are detected faster than contours, it would once again add weight to the before mentioned ideas. This is because contours give a weaker response than straight lines in an accumulator-array structure.

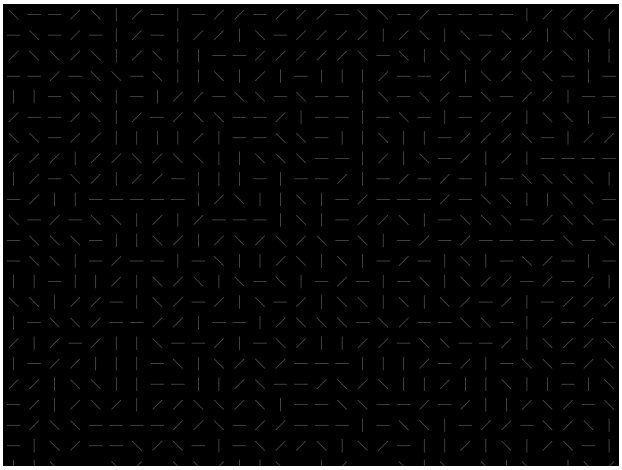
An additional hypothesis is that better alignment of line segments in terms of ρ and θ as well as reduced/low noise conditions should improve accuracy as well as latency of human observers. If the responses of the human observers match that of the Hough Transform implementation/Neural Network under 1) various alignment conditions for line segments and 2) presence of noise, it would add further weight to our key hypothesis.

In our experiments, we presented various test images to human observers on a screen, and asked them to respond to questions that would appear on the screen after the images. Details on the experiment set up and participants are discussed next. We followed the institutional review board (IRB) guidelines for our experiments.

Procedure

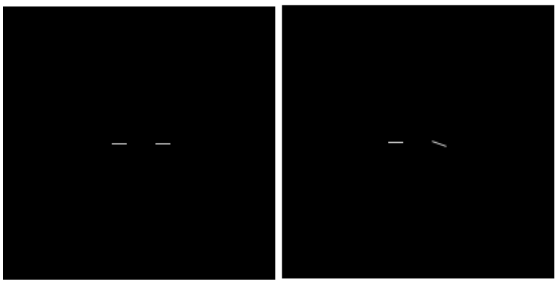


(a) Focus Screen

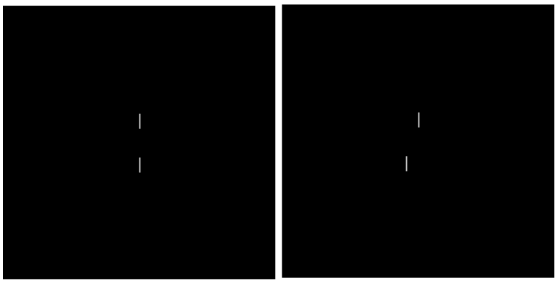


(b) Mask Image

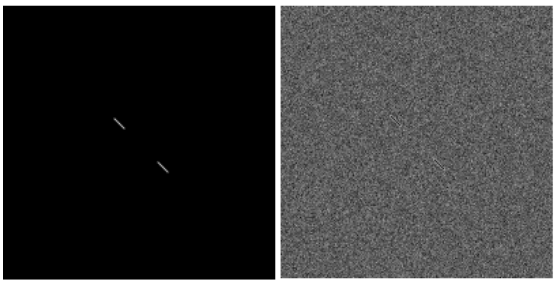
Figure 13: Focus Screen and Mask Image.



(a) Variation in $\theta = 0$ and $\theta = 15$ degrees cases, for horizontal lines.

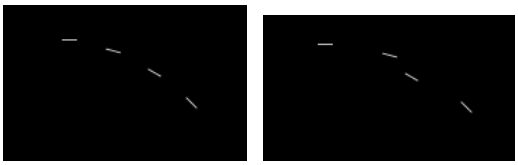


(b) Variation in $\rho = 0$ and $\rho = 9$ pixels cases, for vertical lines.



(c) Variation in $\sigma = 0$ and $\sigma = 0.6$ cases, for inclined lines.

Figure 14: Aligned, Non-Aligned, and Noise Examples.



(a) Aligned Contour (b) Non Aligned Contour

Figure 15: Aligned and Non-Aligned Contour Examples.

An outline of the experiment setup is as follows: the human observer is presented with a number of test images, each test image for an interval of a hundred milliseconds. The observer is then asked to answer a couple of questions that appear on the screnn based on what s/he has seen, as quickly as possible. The answers are to be recorded by clicking on a radio button, next to their option of choice.

Each participant completed a supervised practice session of 5 trials before the experiment. The user would indicate his/her response to questions by clicking on the corresponding option on screen. For example, to indicate that a line was aligned, (s)he would click on the radio button appearing next to the word "Aligned" appearing on screen.

Before every test image, the observer will be presented with a focus screen, informing the observer where to direct his or her gaze, Fig.13.a. After every test image, a masking image will be shown, Fig.13.b. The display screen is at a distance of 50 cm from the observer. The screen resolution is 1920×1080. The observer position is stabilized using a chin rest. The test images will have line segments that may or may not be aligned, with or without noise, lying along a contour or along a straight line. Sample images are similar to Fig.14, 15. The program for displaying the images and recording user data was created using QT [37].

Participants 15 university undergraduates (6 men, 9 women; age range 18 - 24 years) participated in Experiments 1-3 for course credit. All participants had normal to corrected vision. Each participant completed the 3 experiments

in a single session.

15 university undergraduates (8 men, 7 women; age range 18 - 24 years) participated in Experiments 4-7 for course credit. All participants had normal to corrected vision. Each participant completed the 4 experiments in a single session.

Experiments 1 - 3: Line Detection in the Fovea

This set of experiments examine how detection of lines in the fovea is affected by misalignment in orientation (variation in θ, Experiment 1), misalignment in distance (variation in ρ, Experiment 2), and the presence of noise (variation in noise σ, Experiment 3). The line segments always appeared within 4° of the fovea. The line segments were each 10 pixels long. The orientation of the line segments could be one of $4 : 0^\circ$, 45°, 90°, and 135°.

Stimuli:

In Experiment 1, the misalignment between the line segments is in the orientation of the two line segments. The difference in θ could be one of five - 0°, 5°, 10°, 15°, and 20°. For the 4 orietations mentioned before, and 5 differences in θ for each orientation, 15 test images were generated - amounting to 300 images in total. The observer was asked to indicate whether the line segments were aligned or not, and if they were aligned, to indicate one of the four possible orientations.

In Experiment 2, the misalignment between the line segments is in the placement of the two line segments. The difference in ρ in pixels could be one of five - 0, 3, 6, 9, 12. Once again, 15 test images were generated for 4 orientations and 5 differences in ρ - resulting in 300 images in total. The observer was asked to indicate whether the line segments were aligned or not, and if they were aligned, to indicate one of four possible orientations.

In Experiment 3, the line segments are perfectly aligned, but embedded in noise. The noise added was Gaussian, with 5 possible values of standard deviation, σ - 0, 0.2, 0.4, 0.6, 0.8. 15 test images were generated for each of 4 orientations with 5 possible values of σ, resulting in 300 images. The observer was asked to indicate whether the line segments were detected or not, and if

they were detected, to indicate one of four possible orientations.

Experiments 4 - 6: Line Detection in the Periphery

In these experiments, the line segments always appeared within 10° and 20° of the fovea. The line segments were 40 pixels long. The orientation of the line segments could be one of 4 - 0°, 45°, 90°, and 135°. The conditions tested were the same as before : variation in θ, variation in ρ, and variation in noise σ.

Experiment 7: Detection of Contours

In these experiments, lines as well as contours consisting of line segments, see [14], that are aligned or not aligned, are presented to the observers. The observers are asked if the line segments were aligned or not, and if they were aligned, whether they were aligned along a contour or along a line.

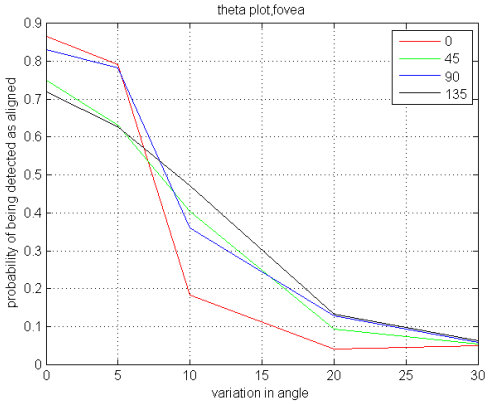
Results

The results obtained from the fovea are shown in Fig.16 for variation in θ, ρ, and σ of noise. The results from periphery are shown in Fig.17.

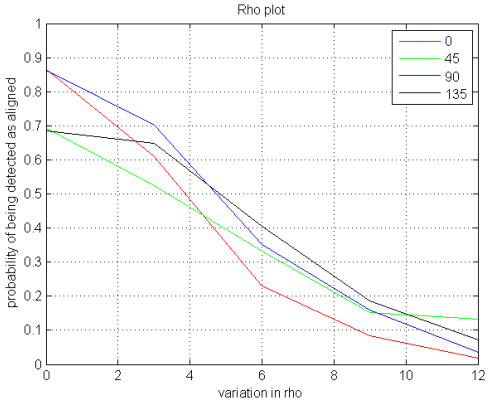
As expected, in both fovea and periphery, the human subjects detected the lines as aligned with a higher probability when the variation in θ and ρ values were low. The drop in probability is low for periphery, for the variation in ρ case. In presence of noise, as the noise increased, the subjects failed to detect the lines, no matter the orientation, in the fovea as well periphery. The overall variation in the curves follow the same trend for both fovea and periphery and supports the idea that better alignment of line segments in terms of ρ and θ as well as reduced noise conditions improves accuracy of human observers. No marked variation in latency was observed in these experiments.

The response times for the users for both fovea and periphery was recorded, and the average response time in both cases was found to match - Fig.18. Combined with the previous results, it adds support to the hypothesis that peripheral and foveal responses match.

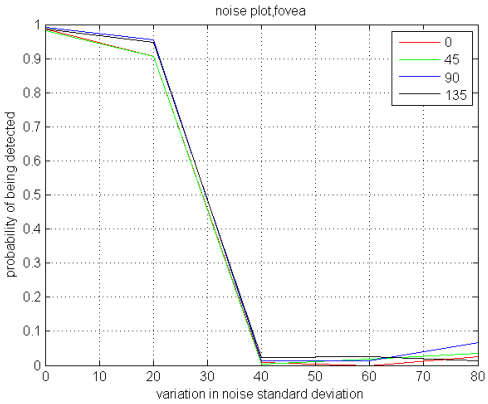
The human subject responses to the contour vs straight line experiment are shown in Fig.19. The responses to aligned straight lines and contours matched in terms of accuracy and response time. This contradicted our hypothesis that lines would be detected faster than contours. However, non-aligned straight



(a) Variation in θ

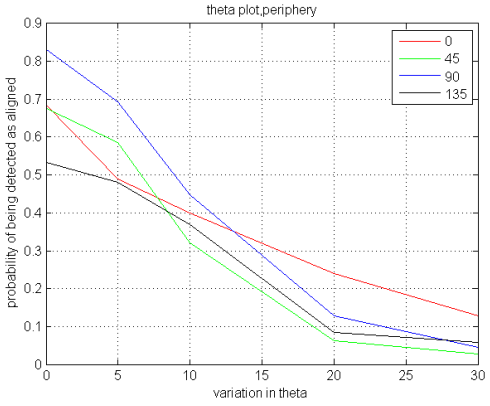


(b) Variation in ρ

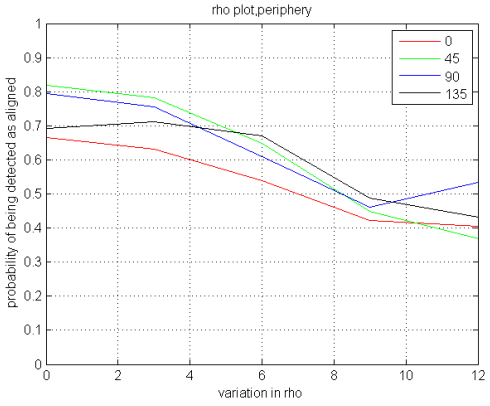


(c) Variation in noise σ

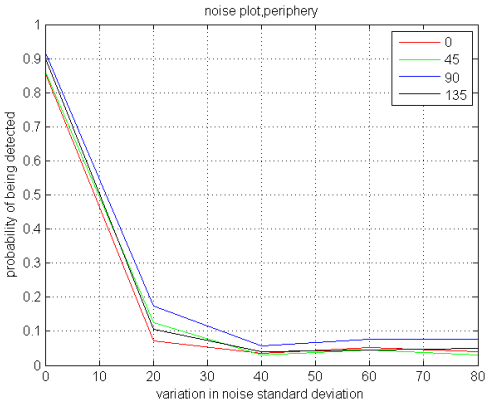
Figure 16: Plots showing the variation in detection of the lines by the human observers, for all 4 angles when θ, ρ, and noise σ were varied, in fovea.



(a) Variation in θ



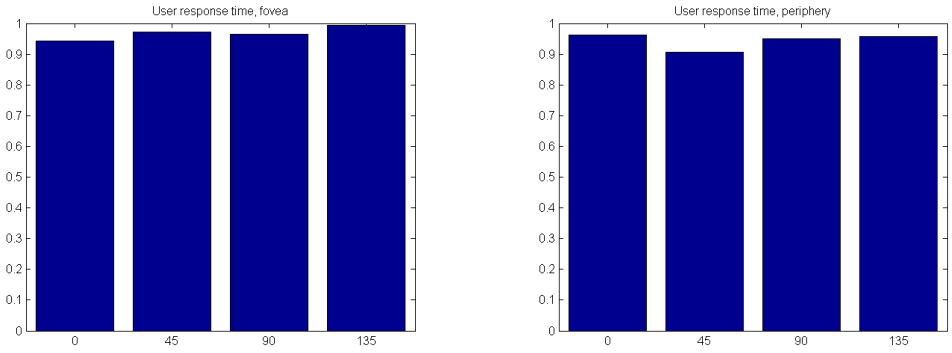
(b) Variation in ρ



(c) Variation in noise σ

Figure 17: Plots showing the variation in detection of the lines by the human observers, for all 4 angles when θ, ρ, and noise σ were varied, in periphery.

lines were detected as misaligned more accurately, and with a shorter response time, than the non-aligned contour. Since the non-aligned straight lines had two alternate line segments aligned along a line, while the non-aligned contour had two alternate line segments aligned along a contour, both cases had comparable degrees of misalignment. It could be argued that humans are better at detecting lines as misaligned than contours in short intervals of time - given sufficient time to examine the test images, the subjects could have easily determined the misaligned contours easily. However, for the very short times the test images were displayed, the subjects incorrectly identified contours as aligned a lot more often than lines. This could still imply that straight line detection is a hard wired process, not needing the full attention of the observer. The matching responses for both contours and lines in perfectly aligned cases could be because the aligned line segments give rise to comparable responses while viewing contours as well as straight lines.



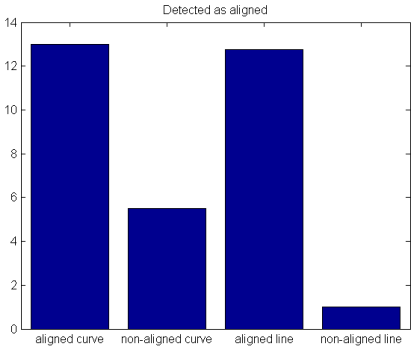
(a) Average user response time in seconds, Fovea

(b) Average user response time in seconds, Periphery

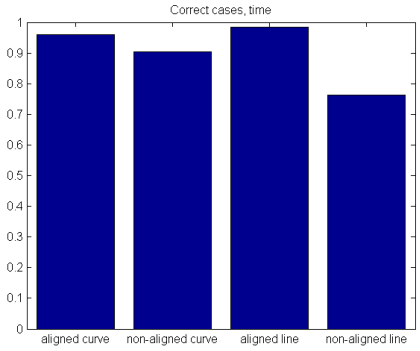
Figure 18: Comparison and fovea and periphery response times.

4. Conclusion

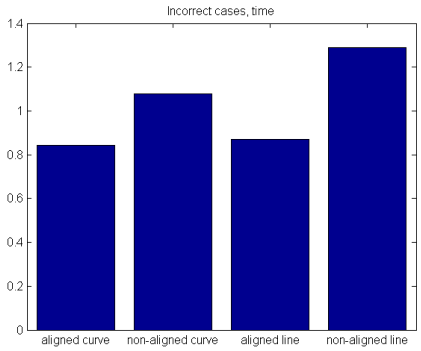
An artificial neural network that learned to detect straight lines using the same parameterization as the Hough Transform was presented. The network



(a) No:of times out of 16, when each of the cases was detected as aligned.



(b) Average reaction time in seconds when the user response was correct



(c) Average reaction time in seconds when user response was incorrect

Figure 19: Results from the contour experiments.

output to various test images with straight lines was compared to the output from various implementations of the Hough Transform. The network was found to give a comparable performance. Responses of human subjects to similar test images were also evaluated. Results similar to the neural network and other Hough Transform implementations were obtained. It can be concluded that the network responds in a manner similar to human visual cortex for detection of straight lines.

A theoretical examination of the operation of the network, in terms of minimization of an objective function, will be detailed in the future.

The network may be further extended by adding additional layers. Along with straight lines, the training set can now be extended by adding simple shapes like triangles and squares, and then more more complex shapes. The idea is to built a hierarchical model where deeper layers learn to represent more complex objects, by pooling from previous layers for simpler shapes.

References

[1] M. Bar, K. S. Kassam, A. S. Ghuman, J. Boshyan, A. M. Schmid, A. M. Dale, M. S. Hamalainen, K. Marinkovic, D. L. Schacter, B. R. Rosen, and E. Halgren. Top-down facilitation of visual recognition. *Proc. Natl. Acad. Sci. U.S.A*, 103:449–454, 2006.

[2] J. Basak. Learning hough transform: A neural network model. *Neural Computation*, 13:651–676, 2001.

[3] A. J. Bell and T. J. Sejnowski. The independent components of natural scenes are edge filters. *Vision Res.*, 37(23):3327–3338, 1997.

[4] E. Borenstein and S. Ullman. Combined top-down/bottom-up segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(12):2109 – 2125, 2008.

[5] F. M. G. da Costa and L. da F. Costa. Straight line detection as an optimization problem: An approach motivated by the jumping spider visual

system. In *Biologically Motivated Computer Vision, Lecture Notes in Computer Science*. Springer, 2000.

[6] P.M. Daniel and D Whitteridge. The representation of the visual field on the cerebral cortex in monkeys. *J. Physiol*, 159:203–221, 1961.

[7] E. Duan, M. Xie, Q. Mo, Z. Han, and Y. Wan. An improved hough transform for line detection. *International Conference on Computer Application and System Modeling (ICCASM)*, 2010.

[8] E. Erwin, K. Obermayer, and K. Schulten. Models of orientation and ocular dominance columns in the visual cortex: a critical comparison. *Neural Computation*, 7(3):425 – 468, 1995.

[9] P. Foldiak. Forming sparse representations by local anti-Hebbian learning. *Biol. Cybern.*, 64:165–170, 1990.

[10] P. Foldiak. Learning invariance from transformation sequences. *Neural Computation*, 3:194–200, 1991.

[11] P. Foldiak. *Models of sensory coding*. PhD thesis, University of Cambridge, 1991.

[12] K. Fukushima. Neocognitron: A hierarchial neural network capabe of visual pattern recognition. *Neural Networks*, 1:119 – 130, 1988.

[13] C. Fyfe. *Hebbian Learning and Negative Feedback Networks*. Springer, 2005.

[14] W. S. Geisler, J. S. Perry, B. J. Super, and D. P. Gallogly. Edge co-occurence in natural images predicts contour grouping performance. *Vision Research*, 41:711–724, 2001.

[15] D. O. Hebb. *The Organization of behaviour: A Neuropsychological Theory*. Psychology Press, 2002.

[16] G. Hinton and T. Sejnowski. Optimal perceptual inference. *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 448 – 453, 1983.

[17] G. Hinton and T. Sejnowski. Learning and relearning in boltzmann machines. In D. RumelHart and J. McClelland, editors, *Parallel Distributed Processing: Explorations in Microstructure of Cognition*. MIT Press, 1986.

[18] D. H. Hubel. *Eye, Brain and Vision*. Scientific American Library Series, Available online: http://hubel.med.harvard.edu/index.html, 1995.

[19] D. H. Hubel and T. N. Wiesel. *Brain and Visual Perception*. Oxford University Press, 2005.

[20] D.H. Hubel and T.N. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *J.Physiol.*, 160:106–154, 1962.

[21] D.H. Hubel and T.N. Wiesel. Receptive fields and functional architecture of monkey striate cortex. *J.Physiol.*, 195:215–243, 1968.

[22] D.H. Hubel and T.N. Wiesel. Sequence regularity and geometry of orientation columns in the monkey striate cortex. *J.comp.Neurol.*, 158:267–293, 1974a.

[23] D.H. Hubel and T.N. Wiesel. Uniformity of monkey striate cortex:a parallel relationship between field size, scatter, and magnification factor. *J.comp.Neurol.*, 158:295–302, 1974b.

[24] T. Jacob and W. Snyder. Learning rule for associative memory in recurrent neural networks. *International Joint Conference on Neural Networks (IJCNN)*, 2015.

[25] T. Kohonen. Emergence of invariant-feature detectors in the adaptive-subspace self-organizing map. *Biol. Cybern.*, 75:281–291, 1996.

[26] T. Kohonen. *Associative Memory: A System Theoretical Approach*, volume 17. Springer Science and Business Media, 2012.

[27] T. Kohonen and E. Oja. Fast adaptive formation of orthogonalizing filters and associative memory in recurrent networks of neuron-like elements. *Biol. Cybern.*, 21(2):85–95, 1976.

[28] M. Köppen, A Soria-Frisch, and Vicente-Garcia. Neurohough: A neural network for computing the hough transform. In *Artificial Neural Nets and Genetic Algorithms*. Springer, 2001.

[29] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86:2278 – 2324, 1998.

[30] R. Linsker. Self-organization in a perceptual network. *Computer*, 21(3):105–117, 1988.

[31] R. Linsker. Local synaptic learning rules suffice to maximize mutual information in a linear network. *Neural Computation*, 4(5):691–702, 1992.

[32] S. Maji and J. Malik. Object detection using a max-margin hough transform. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[33] D. McLaughlin, R. Shapley, M. Shelley, and D. J. Wielaard. A neuronal network model of macaque primary visual cortex (v1): Orientation selectivity and dynamics in the input layer 4c. *Proceedings of the National Academy of Sciences*, 97(14):8087 – 8092, 2000.

[34] N. Murshed. A neural network structure for detecting straight line segments. *International Joint Conference on Neural Networks (IJCNN)*, 1999.

[35] E. Oja. A simplified neuron model as a principal components analyzer. *J. Math. Biol.*, 15:267–273, 1982.

[36] B. A. Olshausen and D. J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381:607–609, 1996.

[37] QT Project. `http://qt-project.org`.

[38] R. P. N. Rao and D. H. Ballard. Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nat. Neuroscience*, 2(1):79–87, 1999.

[39] N. Razavi, J. Gall, P. Kohli, Z. Han, and L. V. Gool. Latent hough transform for object detection. *European Conference on Computer Vision (ECCV)*, 2012.

[40] M. Riesenhuber and T. Poggio. Hierarchial models of object recognition in cortex. *Nat. Neuroscience*, 2(11), 1999.

[41] E. L. Schwartz. Spatial mapping in the primate sensory perception:analytic structure and relevance to perception. *Biol. Cybern.*, pages 181 – 194, 1977.

[42] T. Serre, M. Kouh, C. Cadieu, U. Knoblich, G. Kreiman, and T. Poggio. A theory of object recognition: Computations and circuits in the feedforward path of the ventral stream in primate visual cortex. *AI Memo 2005-036BCL Memo 259, MIT*, 2005.

[43] Q. Wu, T. M. McGinnity, L. Maguire, G. D. Valderrama-Gonzalez, and J. Cai. Detection of straight lines using a spiking neural network model. *Fifth International Conference on Natural Computation (ICNC)*, 2009.

Supplementary Material

Preliminary Results

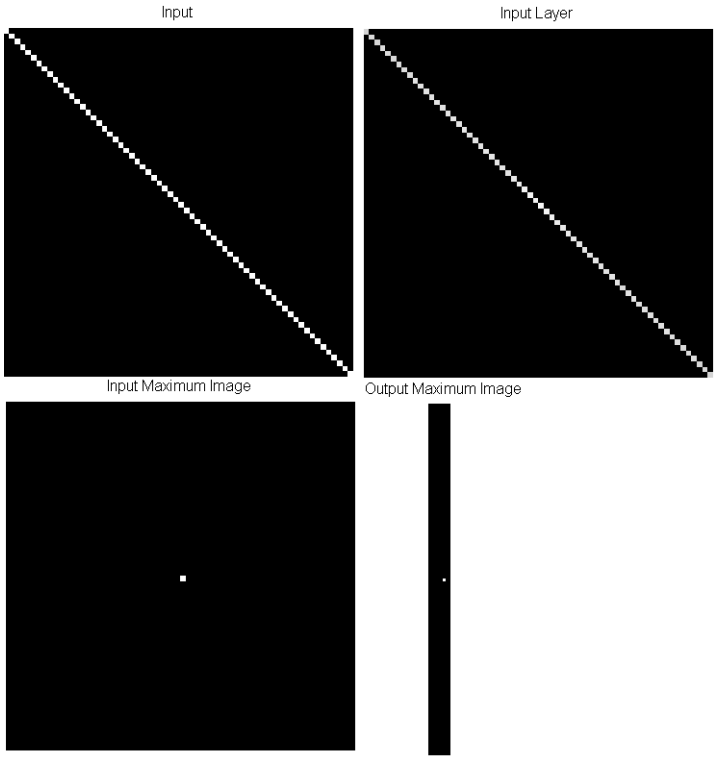


Figure 20: Full diagonal line.

Fig.20 and Fig.22 show the results obtained from a network trained with 135° lines. In Fig. 20, a diagonal line was given as the input. The same line was represented in the input array. We obtained the maximum in the input array at the centroid of the line, and the maximum in the output array at the row corresponding to the line. In Fig.22, the same diagonal line, but occluded, was given as the input. The input array filled up the occluded portion. The maximum in the input array is now shifted as the input line is incomplete. However, we continue to obtain the same maximum in the output array as the full line, because the output array managed to identify the full line from the segments. In Fig.21, we see a shifted diagonal line. The input array values,

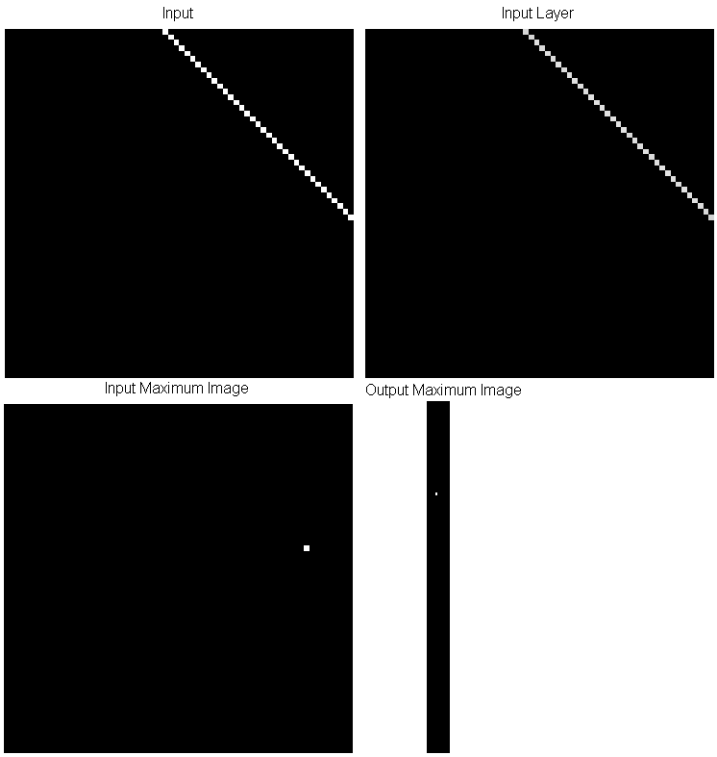


Figure 21: Full diagonal line, different position.

as well as the positions of maximum values in the input array and output are shifted as well.

Fig.23 shows the results from a network trained with horizontal line segments. When a particular line segment was presented, the input array filled up part of the full line. The maximum in the input array and the output array both correspond to the particular line segment.

Fig.24 are the results obtained when a network trained with diagonal lines was presented with an image with clutter. The input layer spread out all of the line segments along the diagonal direction. As there were aligned diagonal line segments along the origin, the maximal output corresponded to a diagonal line along the origin.

The results obtained are in agreement with Gestalt principles. Fig.22 can be

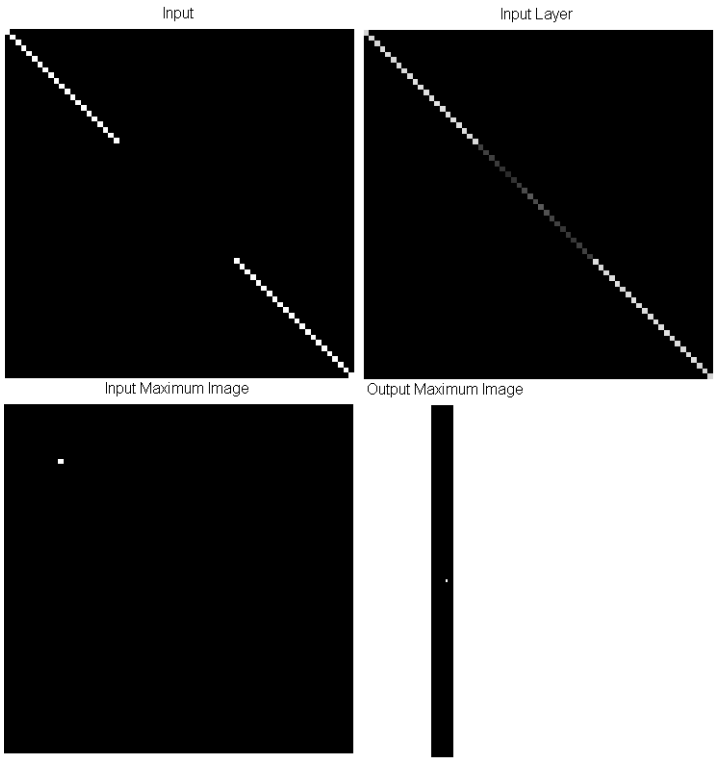


Figure 22: Occluded diagonal line.

thought of as following the Gestalt law of closure, while Fig.24 can be thought of as following Gestalt law of continuity.

Algorithmic Representation of HT implementations

Algorithm 1 Conventional Hough Transform, for an Image $\mathcal{I}$ of size $\mathcal{M} \times \mathcal{N}$.

for $\theta = 0$ to π **do**
  for $x = 1$ to $\mathcal{M}$ **do**
    for $y = 1$ to $\mathcal{N}$ **do**
      $\rho = x\cos(\theta) + y\sin(\theta)$
      $\mathcal{A}(\rho, \theta) = \mathcal{A}(\rho, \theta) + \mathcal{I}(x,y)$
    end for
  end for
end for

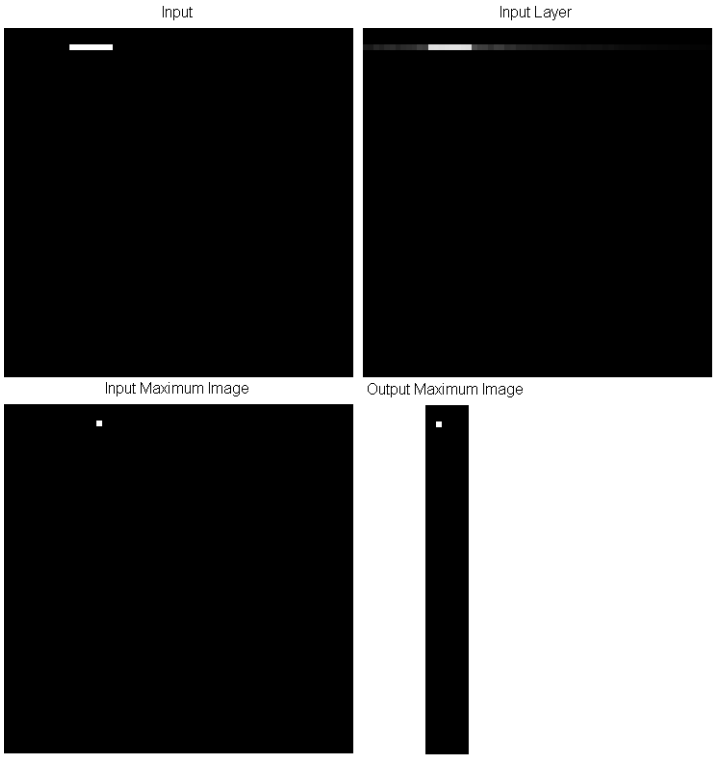


Figure 23: Horizontal line segment.

Algorithm 2 Gabor Filter Hough Transform, for an Image $\mathcal{I}$ of size $\mathcal{M} \times \mathcal{N}$.

for $\theta = 0$ to π **do**
  for $x = 1$ to $\mathcal{M}$ **do**
    for $y = 1$ to $\mathcal{N}$ **do**
      $\rho = x\cos(\theta) + y\sin(\theta)$
      $\mathcal{A}(\rho, \theta) = \mathcal{A}(\rho, \theta) + \mathcal{I}(x,y)\ gf(x, y, \theta)$
    end for
  end for
end for

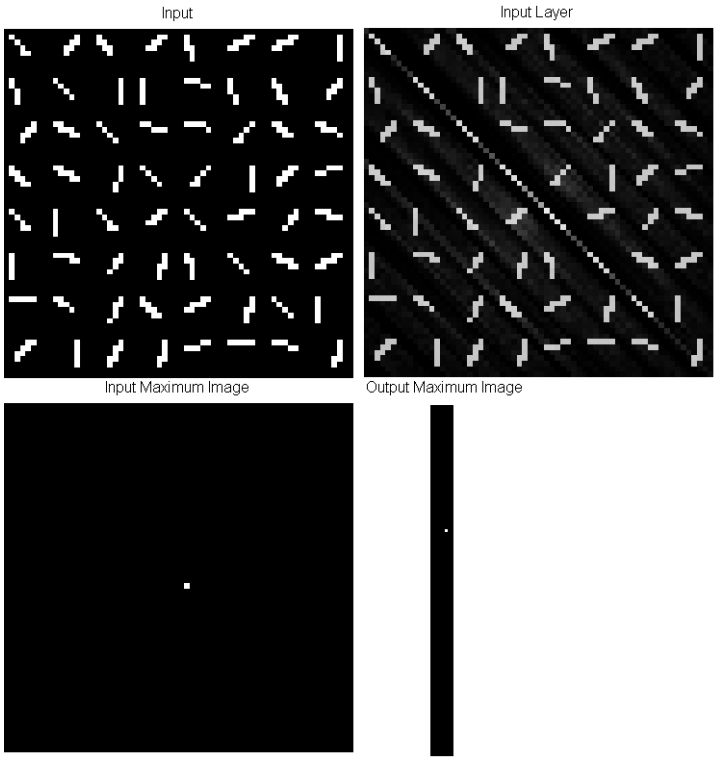


Figure 24: Network trained of diagonal line, presented with a clutter.

Extension to all orientations

The architecture outlined can be extended to detect other orientations in two ways -

1. Build and train separate networks for lines of separate orientations, or
2. Estimate the "actual" orientation from the peak outputs of the 4 networks already trained.

The first approach is straightforward, but time consuming. In the second approach, the outputs of 4 networks already trained are considered. The networks were trained separately with lines of the following orientations - 0°, 45°, 90°, and 135°. The outputs of these networks (after training), for each of the 4 orientations, are then considered.

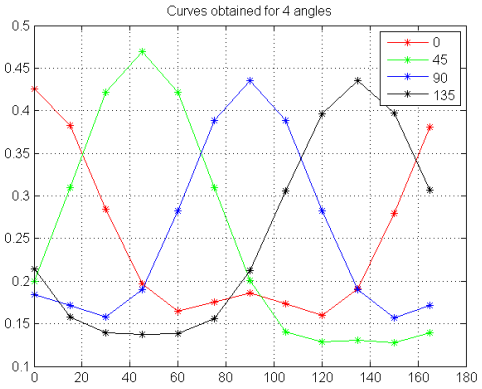
As expected, each network responds maximally to the orientation with which it was trained, Fig.25a. For example, after training, the 0° network responds maximally to a horizontal line, while the 90° responds minimally to it. A curve is then fitted to pass through the output values of each of the networks, for each of the orientations, using a standard interpolation algorithm. We used an inbuilt MATLAB interpolation algorithm using the Fast Fourier Transform(FFT). The points are transformed to the Fourier domain using FFT, and then transformed back to obtain more points.

Four curves are obtained after the interpolation process. These 4 curves are all centered, Fig.25b and averaged, Fig.25c, to obtain a single curve which captures the variation in outputs from each of the trained networks. Let this curve be called the "output" curve, $f(\theta)$, with a peak at $\theta = 90^\circ$. The curve is then shifted to peak at $\theta = 0^\circ$.

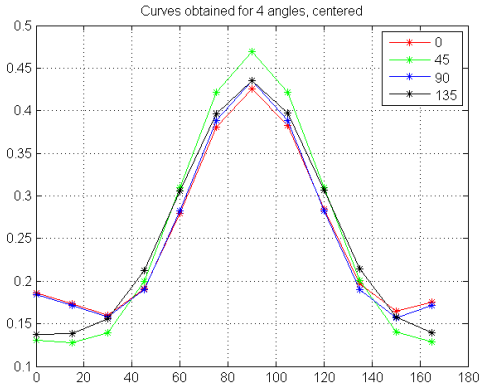
A test line (of any orientation) is then presented to the 4 networks, and their outputs are considered. Once again, a curve is fitted through the 4 outputs. Let the resulting curve be $g(\theta)$, where θ varies from 0 to π. The "output" curve is made to slide over the resulting cuve (much like in a convolution), and the square of the error between both curves are computed. The angle corresponding to the location of minimum error is then declared as the angle of the test line.

$$\phi^* = \underset{\phi}{\operatorname{argmin}}\left(\sum_{\theta} \left(f(\theta) - g(\phi - \theta)\right)^2\right) \tag{10}$$

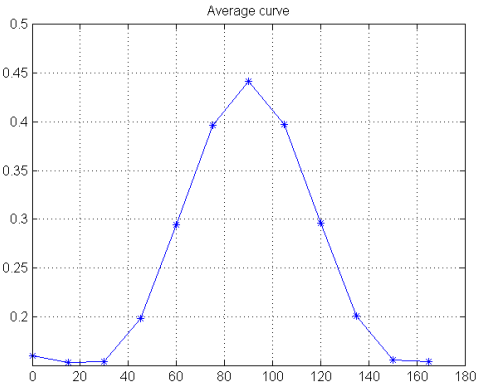
The approach outlined above gave us the correct angle in the all of the test lines (with orientations varying from 0 to π) considered.



(a) Curves obtained for the four angles.



(b) The four curves centered.



(c) Single curve after averaging.

Figure 25: Extension to all orientations by curve estimation.