Open access • Proceedings Article • DOI:10.1117/12.644430

Finding corners in images by foveated search

Thomas L. Arnow, Alan C. Bovik

Institutions: University of Texas at Austin

Published on: 15 Jan 2006 - Visual Communications and Image Processing

Topics: Canny edge detector, Corner detection, Edge detection, Foveal and Feature detection (computer vision)

Finding corners in images by foveated search

Thomas L. Arnow[1] and Alan C. Bovik[2]
Laboratory for Image and Video Engineering (LIVE)
Department of Electrical and Computer Engineering
The University of Texas at Austin, Austin, TX 78712-1084, USA
[1]Email: arnowt@uthscsa.edu [2]Email: bovik@ece.utexas.edu

ABSTRACT

We develop a new approach to finding corners in images that combines foveated edge detection and curvature calculation with saccadic placement of foveal fixations. Each saccade moves the fovea to a location of high curvature combined with high edge gradient. Edges are located using a foveated Canny edge detector with spatial constant that increases with eccentricity. Next, we calculate a measure of local *corner strength*, based on a product of curvature and gradient. An inhibition factor based on previous visits to a region of the image prevents the system from repeatedly returning to the same locale. A long saccade is move thes fovea to previously unexplored areas of the image. Subsequent short saccades improve the accuracy of the location of the corner approximated by the long saccade. The system is tested on two natural scenes and the results compared against subjects observing the same test images through an eyetracker. Results show that the algorithm is a good locator of corners.

Keywords: Foveated Vision, corner detection, machine vision, feature detection,

1. INTRODUCTION

One of the most difficult and poorly designed problems in the field of image analysis is visual search. Does visual search mean an algorithm for finding and identifying a specific object or class of objects in an image? The extensive literature on automated target recognition (ATR) exemplifies this philosophy[1]. Or, does visual search imply a general framework for finding information from visual data without object-specific guidance? There is only a small literature on generic automated visual search. Methods include search based on contrast[2]; image and depth gradients[3]; other edge factors[4]; proximity between objects[5]; object similarity[6], and combinations of randomized saliency and proximity factors[7]. While both strategies have merit, great benefit would result from the development of basic principles guiding the design of visual search algorithms, applicable to a diversity of applications, and which would address some of the factors that have limited the success of visual search.

Primate and other animal visual systems are foveated, with high resolution near the center of gaze that falls off as a power function of eccentricity, the angle away from the center of gaze. In humans, the fovea is a circular region of tightly packed cones, roughly 1.5 mm in diameter[8]. This density decreases rapidly with eccentricity. In the central fovea, receptors are packed at a density of about 120/degree[9,10]. This corresponds to a resolution of .291 mm at a viewing distance of 100 cm. Foveated primate vision systems mechanically direct the fovea around a scene over time via very fast ballistic eye movements called saccades, resulting in series of static fixations[11,12]. Foveation is an effective compromise between the demand for high-resolution vision and the limited transmission and processing bandwidths of the optic nerve and subsequent brain regions. The amount of information flowing from retina to brain is far less than if the entire retina was sampled at foveal density. The brain uses peripheral, low-resolution information to decide on the next fixation point. This is accomplished quickly – the human eye typically makes more than 10,000 saccades per hour, ranging in distance from a few seconds of arc to up to over 70° [9,10,13,14]. Certainly the computation of new fixations must be fast, automatic, and image-driven Such processing is efficient for accomplishing visual search with active, mobile cameras or eyes [3,15].

Rather than processing a wide field of view (FOV) visual stream all at once, high-spatial-resolution search is conducted over a very small FOV (the image on the fovea) while wide-FOV search, rich with context but lacking detail, occurs over the peripheral field of view. Candidate discoveries in the periphery can be rapidly analyzed at high resolution via saccadic eye movements that redirect the candidate to the fovea; in the absence of candidates, eye movements may be sequentially deployed to enlarge the search space.

Visual information gathering and visual search is greatly augmented by deploying the foveation-fixation-scanpath process. We believe that this solution can, and should, be adapted into computational systems for visual information acquisition and processing. Most practical image processing systems, however, do not operate with mobile cameras. The foveation is accomplished in software rather then by camera or eye movement. One example is foveated image compression[15,16,17] which requires knowing where the spatial fixation on the image of the human observer is, and the distance of the observer from the image, so that the foveated fall-off can be matched to that of the observer's eye. This can be accomplished by eyetracking, headtracking and other physical measurements[17].

In foveated visual search systems, there is also the interesting central problem of fovea placement during the search process. In such systems, the goal of the placement is not to match the position of gaze, but rather, to optimize the gathering of information relevant to the object(s) being searched. Subsequent fixations should be chosen based on the available foveated data. The amount of information available to the search algorithm regarding the object(s) of interest is then determined by the proximity of the object to the current fixation.

Corners have long been recognized as rich bearers of visual information, and numerous algorithms have been proposed for detecting using them as features in basic visual tasks such as object recognition, stereo matching, shape analysis and optical flow computation[18-23]. In their early seminal work on computational vision, Marr and Hildrith[8,24], regarded corners as being of high visual saliency, and designated them as being members of the discontinuity class in the theory of the full primal sketch. They viewed corners as an important member of the class of image primitives that are used as building blocks for representing objects in image understanding systems, whether biological or computational. Other features exist, of course, such as edges. However, corners are more localized than edges, and as pointed out by Nobel[18], are superior to edges for defining the shapes of objects, since edges detectors only provide location information in a single direction (normal to the edge). Certainly corners present advantages as a discrete image feature, since they are simultaneously information-rich, yet require minimal description. Accurate corner information is not easy to acquire; for example, Mehrotra et al.[19] point out that edge detectors tend to perform poorly near corners, suggesting that corner detection by locating intersections between edges can lead to poor performance.

In this paper, we cast the problem of corner detection as a search process. We apply principles of foveated visual search and automated fixation selection in accomplishing the corner search. Thus, we approach the search process from a low level, searching for objects without requiring building blocks to represent them, since the objects being searched for are the same as the features. In this way, we hope to contribute by supplying a case study of both foveated search, and foveated feature detection. The result is a new algorithm for finding corners (viewed from the perspective of foveated feature detection), but which may also be considered as a corner-based algorithm for aiming computed visual fixations (along with a computated fovea), with the eventual goal of extracting information that is useful for more sophisticated object recognition system.

With this last interpretation in mind, as an interesting comparison study we also compare fixations generated by this algorithm with those of subjects viewing the same images, whose eye movements are being recorded by an eyetracker. The comparison is made using an information-theoretic measure.

2. COMPUTING EDGE AND CORNER FEATURES IN FOVEATED IMAGES

While foveation presents significant advantages for visual search via the efficient allocation of resources, it also presents new challenges for accomplishing low-resolution and spatially-varying object recognition, since each foveated view distorts the image away from fixation by reducing the resolution. Near the fixation point, fine features, such as edges and corners, are resolved well. Away from the fixation point, these fine details may be distorted or lost.

The overall approach that we will take towards searching for corners will involve foveating the image, deciding a most likely location of a corner, moving the fixation to that vicinity, refining the corner location estimate, identifying the corner – then choosing a next likely corner location, and so on. The details of the overall search methodology will be given later. In the following sections we describe the method we use to create foveated images; the method of edge detection we use on foveated images, and the method of corner detection we use on the detected foveated edge maps.

2.1 FOVEATION FILTERING

There are several possible methods for creating foveated images, the most popular of are either based on spatial-domain foveation filtering or wavelet-domain foveation. Foveation filtering is the most straightforward method[15,17,25] wherein a bank of low-pass filters is applied to the image on a pointwise basis, with bandwidths monotonically decreasing with eccentricity. The filters used are usually symmetric, unit-volume 2-D Gaussians of the form[15,17,25]

$$G_\sigma(x,y) = \frac{1}{\sqrt{2\pi}\,\sigma} e^{-\frac{x^2+y^2}{2\sigma^2}}. \quad (1)$$

The other popular approach is to selectively subsample and quantize the image data in the wavelet domain, leading to decreased resolution away from the fovea. Such techniques have proven very effective for image and video compression[16,17,25]. We choose to use the simple and direct method of foveation filtering with Gaussian filters, owing to the simplicity of the method, and since the artifacts that naturally arise in accomplishing wavelet-domain quantization might lead to spurious corner responses.

The cutoff frequencies of the Gaussian filters (1) can, in principle, be made to decrease continuously with eccentricity, to match the sampling grain of the image. It is also possible to more coarsely quantize the cutoff frequencies so that concentric rings of constant cutoff frequency are formed on the image surrounding the foveation point. This simplifies implementation while scarcely affecting the foveated image. The half magnitude cutoff frequency of the Gaussian (1) is

$$\omega_c = \frac{\sqrt{2 \cdot \ln(2)}}{\sigma}. \quad (2)$$

Hence one can implement foveation filtering by making σ an increasing function of eccentricity. Several approximations simplify implementation and improve performance. The support of a Gaussian is infinite but a good approximation can be made by centering it in an array of about 3σ pixels square. Instead of continuously varying σ with eccentricity, the image is divided into a series of n bands, B_i that are concentric about the fixation point. The innermost ring is a circle with center the the fixation point, while the outermost ring extends to the borders of the image.

We use a simple formula to determine the radius of each ring:

$$r_0 = 0, \quad r_i = (i+2)^{1.6}, \quad i = 3,4, \ \ldots n-1, \ r_n = \infty \quad (3)$$

where the radius of the i^{th} ring is r_i (pixels). The innermost ring has a radius of 5.8 pixels, and the distances between the rings increases with eccentricity. This formula provides a reasonable tradeoff between execution time and continuity at the ring boundaries. The image between the i^{th} and the $(i-1)^{st}$ ring is convolved with a Gaussian $G_{\sigma_i}(x,y)$, where s_i increases monotonically with eccentricity. Since convolving Gaussians with small values of σ takes less processing time than with larger values, efficiency is achieved by implementing larger Gaussian convolutions via repeated convolutions of smaller Gaussians. Repeated convolutions with Gaussians of spatial parameters designated $\sigma_i, i = 1,\ldots,k$ is equivalent to a single convolution with a Gaussian with spatial parameter

$$\sigma = \sqrt{\sum_{i=1}^{k} \sigma_i^2}. \quad (4)$$

The foveation filtered image is created by the following process: The input image is first convolved with a Gaussian of spatial parameter σ_1 and the results stored. This blurred image is next convolved with a Gaussian of spatial parameter σ_2 and the results stored. The process continues for the maximum possible number of bands. Later, when a fixation point is created, each band is filled from the appropriate stored image. Hence, the k^{th} filtered image is

$$I_k = I * \underset{i=1}{\overset{k}{G}}_{\sigma_i}(x,y), \quad (5)$$

where I is the original input image. The values of σ_i used in the algorithm are set to approximate the spatial frequency response of the human vision system (HVS) based on the following widely used formula[17,25,26]:

$$CT(f,e) = CT_0 e^{\alpha f \frac{e+e_2}{e_2}}. \quad (6)$$

Here $CT(f,e)$ is the contrast threshold expressed as a function of spatial frequency f (cycles/degree) and eccentricity e (degrees). CT_0 is the minimum contrast threshold, a a spatial frequency decay constant, and e_2 is the eccentricity in degrees at which the contrast threshold drops to one half of maximum. The half-magnitude spatial cutoff frequency f_c can be expressed a function of eccentricity by solving:

$$-\log(CT_0) = \alpha \; f_c \frac{e + e_2}{e_2} \tag{7}$$

which yields:

$$f_c = \frac{-\log(CT_0)e_2}{\alpha(e + e_2)} \tag{8}$$

In arriving at this formula, Geisler and Perry 24 fit (6) to various sets of experimental data taken from the vision literature. They found good consistency with the following parameter selections: $a = 0.106$, $e_2 = 2.3$, and $1/76 < CT_0 < 1/64$. Substituting these values into (8) and using an average for the high and low values for CT_0 yields a numeric relationship between cutoff spatial frequency and eccentricity:

$$f_c = \frac{92.024}{e + 2.3}. \tag{9}$$

The spread parameters of the Gaussians may then be found from (2).

2.3 FOVEATED EDGE DETECTION

In our approach to corner search, edges recovered from the foveated images are used as features input to a corner detection apparatus. In the approach given here, we will utilize the relatively simple and straightforward Canny edge detector for several reasons[27]. First, the Canny operator provides excellent localization in the edge detection results; second it is simple and naturally defined; third, it gives good performance where the edge curvature is high, and lastly, it does not require any kind of iterative processing, unlike anisotripic schemes. Given the framework of corner-finding via sequential fixations that we are presenting here, direct, locally-computed approaches appear to be a more natural choice, because of the need for rapid, localized processing. We briefly describe the Canny operator in the context of foveated edge detection. Given an image I(x, y) in which corners are to be searched, the usual method is to form the Gaussian smoothed image

$$S_\sigma(x,y) = G_\sigma(x,y) * I(x,y) \tag{10}$$

from which an estimate of the gradient ∇S_σ is computed. Of course, in our application, the image I is not convolved by a single Gaussian, but is instead smoothed by a space-variant Gaussian. In the Canny formulation, the unit vector in the gradient direction $\angle \nabla S_\sigma$ estimates the direction normal to the edge:

$$\mathbf{n}_\sigma(x,y) = \frac{\nabla S_\sigma(x,y)}{\left|\nabla S_\sigma(x,y)\right|}. \tag{11}$$

Putative edge locations are then marked by the zero crossings of the twice directional derivative in the direction of the normal (11):

$$\frac{\partial^2 S_\sigma(x,y)}{\partial \mathbf{n}_\sigma^2(x,y)} = \mathbf{n}(x,y) \cdot \nabla \left[\frac{\nabla S_\sigma(x,y) \cdot \nabla S_\sigma(x,y)}{\left|\mathbf{n}_\sigma(x,y)\right|^2} \right]. \tag{12}$$

It is easily shown[27] that the zero crossings of (12) are conveniently the same as those of

$$D(x,y) = \nabla S_\sigma(x,y) \cdot \nabla \left[\nabla S_\sigma(x,y) \cdot \nabla S_\sigma(x,y) \right]. \tag{13}$$

Discrete implementation of (13) is accomplished using space-varying discrete directional Gaussian derivatives $G_{\sigma,x}(x,y) = \partial / G_{\sigma,x}(x,y) / \partial x$ and $G_{\sigma,y}(x,y) = \partial / G_{\sigma,y}(x,y) / \partial y$ to compute the discrete gradient expressions $\nabla S_\sigma(i,j)$ at each discrete image coordinate (i, j).

2.4. Detection of Corners

Corners are generally regarded as points of high curvature, or of curvature discontinuity, along the countours of detected boundaries, edges, or local image intensity profiles. Of course, different definitions of curvature exist. A popular measure that we will use was proposed by Kitchen and Rosenfeld[28], who define curvature κ as the derivative of tangent angle, with respect to arc length, of a parametric curve $x = x(t),\ y = y(t)$:

$$\kappa \equiv \frac{d\phi}{ds} = \frac{\dfrac{d\phi}{dt}}{\sqrt{\left(\dfrac{dx}{dt}\right)^2 + \left(\dfrac{dy}{dt}\right)^2}} \tag{14}$$

where s is arc length, ϕ is the tangent angle, and where

$$\phi = \arctan\left(\frac{\frac{dy}{dt}}{\frac{dx}{dt}}\right). \tag{15}$$

Shortening the notation and taking the derivative gives:

$$\frac{d\theta}{dt} = \frac{x'y'' - x''y'}{1 + \left(\frac{y'}{x'}\right)^2}, \tag{16}$$

which substituted into (14) yields:

$$\kappa = \frac{x'y'' - y'x''}{\left(x'^2 + y'^2\right)^{3/2}}. \tag{17}$$

It can be easily shown from the definition that κ equals the reciprocal of the radius of curvature. The curvature measure (17) on a digitized curve is highly sensitive to noise because of the computed derivatives, so the curve is smoothed by fitting a low order polynomial is fit to it in a sliding window, and the derivatives of the polynomial are used in (17) to calculate κ at the center of the window. A local maximum of the absolute value of κ may indicate the presence of a sharp bend in the curve and hence a corner.

Mehrotra et al.[19] developed corner finders based on directional first and second derivatives of Gaussians, which can detect half-edges at any desired angle to each other. Flynn and Jain[20] describe a series of corner detectors based on a variety of curve fitting methods. They also mention the necessity for smoothing the curves. The Moravec interest operator is based on the response of a small averaging window to an image. If the window straddles an edge, then moving it parallel to the edge direction creates a small change in response whereas moving it normal to the edge creates a large response. If the window straddles a corner, however, moving it in any direction will cause a large response. The Moravec[29] detector declares a corner if the minimum change produced by a shift exceeds some threshold.

The Plessey corner finder[18,29] is based on a matrix M of products and squares of directional image derivatives. At points where two eigenvalues of M are large, small shifts of the window position in any direction will cause a large change in its average response, indicating that the point may be a corner. The SUSAN corner[30] detector applies a moving circular template to an image and declares a corner at points where the value at the center of the template is approximately equal to a small portion of the entire template. Mokhtarian and Suomela[31] developed a variable scale corner detector based on Kitchen's formula shown in (4) and the Canny edge detector. They initially convolve the image with a wide Gaussian, smoothing corners into broad curves. Locating the position of maximum curvature gives an estimate of the corner position, which they refine by narrowing the Gaussian, and by tracking the corner as it moves. While no definitive study conducted as yet indicates which corner detection algorithm is to be preferred, we chose the Rosenfeld-Kitchen algorithm for three reasons: first, it is computationally simple, it uses a very natural definition, and it is highly local in computation; second, it is widely used; and third, the use of smoothing in our application reduces the sensitivity of the operator to noise, making noise-reduction corner detection unnecessary.

3. SEQUENTIAL COMPUTATION OF FIXATION POINTS

The strategy of seeking high-interest image features, such as corners, by searching the image using a mobile, foveated tesselation requires the development of efficient algorithms for deciding and placing fixation points, *viz.,* allocating high resources to local image regions while maintaining reduced resources in the periphery. An early example of such an approach appears in Klarquist and Bovik[3], who describe a foveated active stereo vision system which automatically decides its fixation points in a 3-D scene. The system uses multiple criteria, including the image gradient, proximity in both 2-D and in 3-D, and local suppression to discourage repeat visits. More recently, Wang and Bovik[25] describe a foveated video compression system for video teleconferencing also incorporating automatic fixation selection. Fixation points are placed where there is a high likelihood of a human face, using measures of skin tone or motion.

In the approach taken here for searching for corners, the major control functions are the routines that locate each successive fixation point. The algorithm generates long and short saccades in succession, moving from fixation to fixation until a stopping point is reached. The search routine was designed under the following guidelines:
(*i*) Analyze only the foveated edge maps and images without accessing the original image.
(*ii*) Aim each fixation point at a potential corner, meaning a strong edge with high curvature. Since the data is foveated, corner locations are subject to blur ambiguity which increases with distance from current fixation.
(*iii*) At each current fixation, generate a long distance saccade followed by one or more short saccades. The long saccade approximately locates a potential corner. The smaller saccades locate the corner more accurately.
(*iv*) Inhibition controls the sequence of long and short saccades and prevents the fixation point from repeatedly visiting previously determined corner locations.

Initially, fixation is set to the center of the image. While this might seem arbitrary, there is no reason to select another initial location and moreover, there is a tendency in human vision to bring the eye position towards the central axis in the absence of stimuli (according to Kowler[9] "saccades often land in the center of the entire stimulus configuration"). A variable is set so that the initial saccade will be long. After calculating foveated edges from the image, edges smaller than 50 contiguous pixels are removed as being too small to contribute meaningful information.

A *fixation selection measure* $m_{i,j}$ is then computed over the entire edge map (as explained below), and the next fixation is placed at the pixel with the highest value of this measure. A new foveated edge map is created based on the new fixation position and the search algorithm is invoked to produce a short saccade, using a different calculation for the fixation selection measure than for a long saccade. A new foveated edge map is created and another short saccade generated. Short saccades are generated until a corner is deemed found, or until a corner is not found, which is assumed when short saccades are continually generated. If seven short saccades are generated in succession, or if a corner strength measure is sufficiently large to positively identify a corner, then the search is deemed to have failed and a long saccade is generated, which moves the fovea to a different region of the image.

We now describe the fixation selection algorithm in detail. At each fixation a foveated edge map is computed as described in Sections II-B and II-C. A curvature map is computed along the edge (zero-crossing) loci. In order to reduce the effects of noise on the derivative computations, a simple third-order polynomial is locally fit at each point on the zero-crossing contour. The Kitchen-Rosenfeld curvature (17) is then computed at each point (i, j) that lies on the smoothed zero-crossing contours. The curvature strength $\kappa_{i,j}$ is one of the multiplier factors in the fixation selection measure $m_{i,j}$. Figure 1 illustrates the calculation of a curvature map: Figure 1(a) depicts a contour with two points indicated: A and B. Figure 3(b) depicts a close-up of point (A) - a high curvature point – along with its local polynomial fit, while Fig. 3(c) shows the same for the low curvature point B. Finally, Fig. 3(d) shows the original contour in (a) with curvature coded by the intensity of the line (darker = higher curvature).

We believe that curvature alone is not a suitable measure for placement of subsequent fixation points for two reasons. First, very high-curvature locations would be visited repeatedly. Our goal is to successfully search for as many corners that are in the image as possible. Secondly, noise or low-contrast curves may create zero-crossing loci having high curvatures, thus attracting the fovea to uninteresting regions or artifacts in the image. To address the first of these problems, an array of history information is maintained and used to define a second multiplier factor in the fixation selection measure. Let

$$h_{i,j} = \sum_k e^{-\frac{(i-i_{fk})^2+(j-j_{fk})^2}{2\sigma_h^2}} . \qquad (18)$$

where the coordinates of the k^{th} fixation point are denoted. Whenever a fixation point is generated, a Gaussian, $e^{-\frac{x^2+y^2}{2\sigma^2}}$ which is centered about the coordinates of the fixation, is added to the history array. The spatial parameter of this Gaussian is set to 15 to prevent long saccades from coming too close to locations already visited. The history array $h_{i,j}$ is a component of the fixation selection measure $m_{i,j}$, as explained below.

Two observations motivate the next term used in the corner strength measure. First, corners that are further from the current fixation point (i_f, j_f) will be more severely blurred by foveation, and so the apparent curvature of distant corners

will be reduced. Secondly, once a corner is found at (i_f, j_f), a large-scale saccade is desired to cause the algorithm to scan the image more quickly. Hence the distance factor

$$d_{i,j} = \sqrt{(i - i_f)^2 + (j - j_f)^2} \cdot \tag{19}$$

is also used in the fixation selection measure $m_{i,j}$. This term compensates for the fact that corners away from the foveation point turn into broad curves by giving extra weight to curves far from the fovea. In addition, it forces the fixation point to move large distances between fixations, forcing it to scan the entire image more quickly.

We have also chosen to include an edge strength factor in the fixation selection measure. Our viewpoint is that corners having large edge magnitudes are more likely to be associated with significant image structure. Such an assumption is, of course, not necessarily applicable to every scenario and this term could be eliminated. The edge strength factor is simply the squared gradient magnitude of the Gaussian-smoothed space-variant image:

$$s_{i,j} = \left|\nabla S_\sigma (i, j)\right|^2 . \tag{20}$$

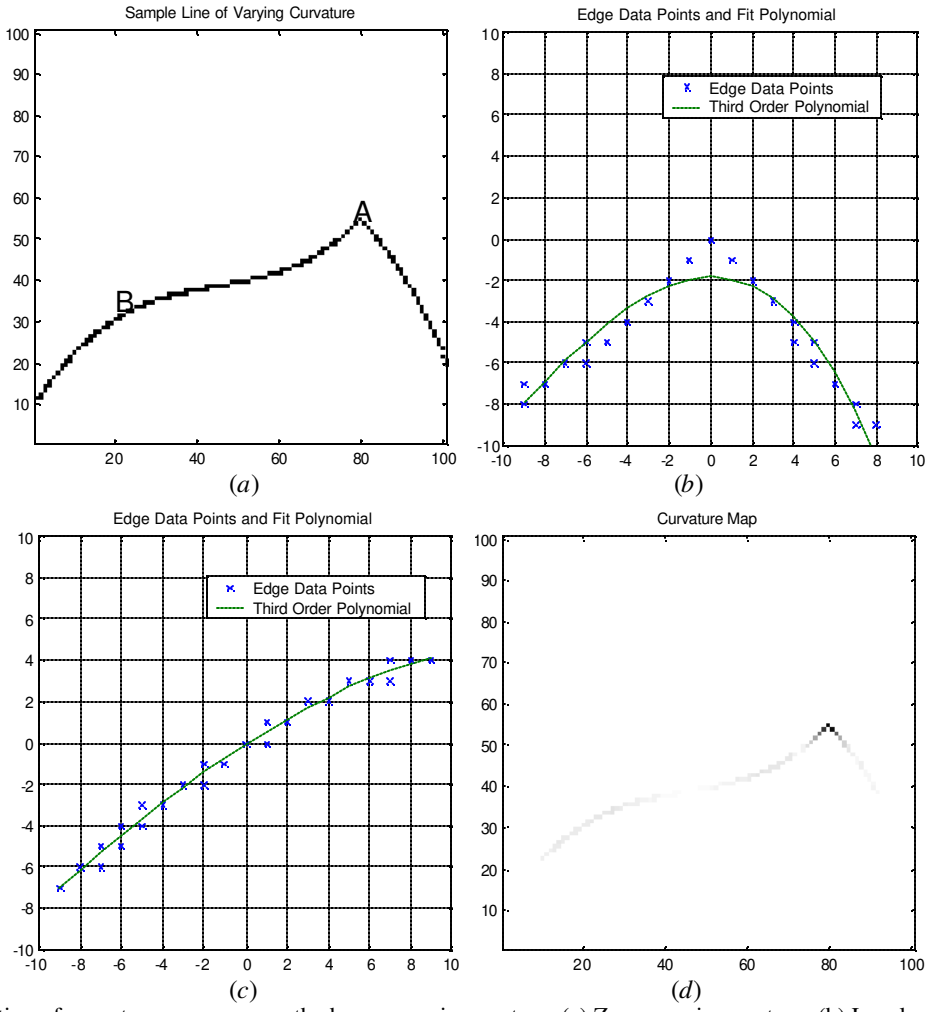


Figure 1. Generation of curvature map on smoothed zero-crossing contour. (a) Zero-crossing contour; (b) Local polynomial fit near the high-curvature point A. The axis coordinates are relative to A. (c) Local polynomial fit near the low-curvature point B. The axis coordinates are relative to B. (d) Zero-crossing contour in (a) with curvature coded as intensity (darker = higher curvature).

The use of this term introduces an additional problem. A high contrast edge of low curvature may attract the fovea to an uninteresting region of the image. To eliminate edges of low curvature, we apply a threshold t to the curvature data:

$$k_{i,j} = 0, if\ k_{i,j} < \tau \tag{21}$$

Regions of curvature less than the reciprocal of t in pixels will be eliminated. The introduction of thresholding requires a logic change due to the fact that the process of foveation reduces the curvature of corners, an effect that increases with distance. Thus, the curvature of a corner far from the fovea may be beneath t causing some corners of a large object to be ignored. Therefore, if no curvature exceeds t, t is temporarily set to zero and a new long saccade generated. After this one saccade, meant to move to a previously unexplored region of the image, t is reset to its initial value.

The overall fixation selection measure is then given by

$$m_{i,j} = \begin{cases} \dfrac{\kappa_{i,j}}{1+\kappa_{i,j}} \cdot \dfrac{s_{i,j}}{1+s_{i,j}} \cdot \dfrac{d_{i,j}}{20+d_{i,j}} \cdot \dfrac{1}{1+h_{i,j}}; C = 0 \\ \dfrac{\kappa_{i,j}}{1+\kappa_{i,j}} \cdot \dfrac{s_{i,j}}{1+s_{i,j}} \cdot \dfrac{d_{i,j}}{1+d_{i,j}}; C > 0 \end{cases} \tag{22}$$

where C controls the length of the saccade. When C = 0, a long saccade is to be generated, and when C > 0 a short saccade is to be generated according to the formula in (22). C is initially given a value of zero, and is incremented by one with each saccade, until it is reset to zero. There are two conditions under which C is reset to zero: (i) The measured curvature $\kappa_{(i_f, j_f)}$ at the current fixation point exceeds a threshold (0.9 in our algorithm), indicating the presence of a corner. (ii) Seven short saccades have been generated: C > 7.

Note that long saccades (C = 0) are discouraged from approaching previous saccades owing to the history term (18) in (22), but this is excluded for short saccades (C > 0) which attempt to zero in on the strongest local corner. The global maximum of $m_{i,j}$ provides the coordinates for the next fixation point. The saccade length control variable C is incremented from 0, producing a sequence of one long saccade, intended to explore a new region of the image, followed by a few short one, which aim to pinpoint the corner accurately. The use of sigmoidal terms prevents any one term from overly raising $m_{i,j}$ when the other terms are not particularly favorable.

We now illustrate the steps of the algorithm by example. Figure 2(a) is the image lighthouse. In each image, the fixation point is designated by the symbol "X." In this example, the current fixation point is presumed to be at the peak of the lighthouse, as indicated. Figure 2(b) shows a foveated version of lighthouse – although, of course, this image is not calculated by the algorithm, since (13) is discrete form is used to generate the zero crossings. Figure 2(c) depicts the foveated edge map calculated by the foveated Canny edge detector, and Figure 2(d) is the foveated Kitchen-Rosenfeld curvature map, with intensity made proportional to curvature.

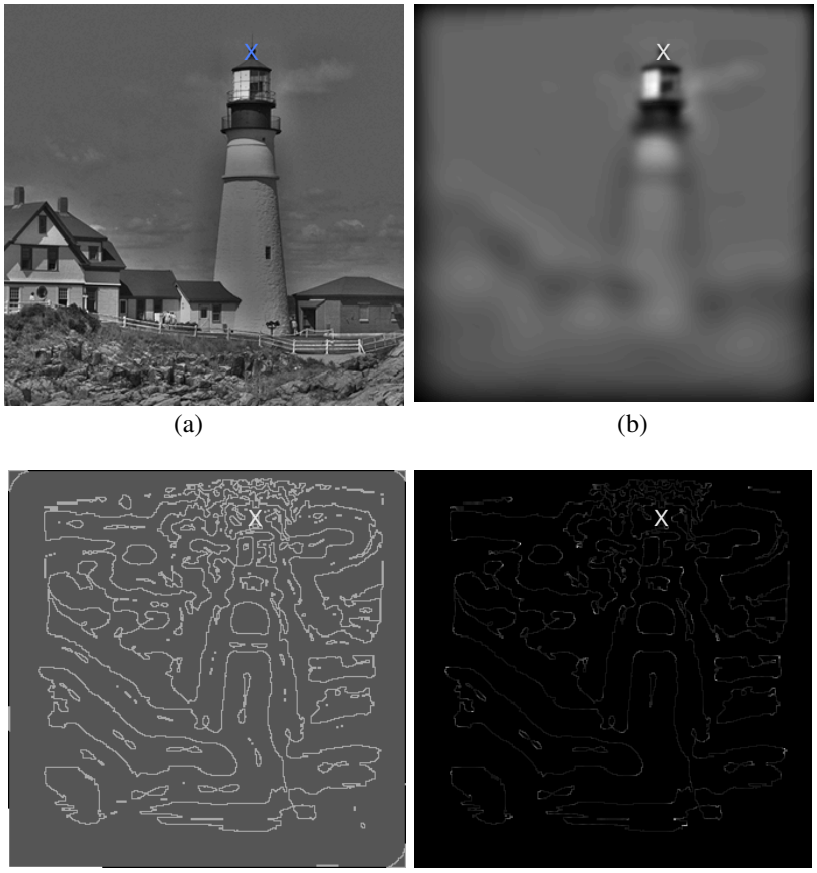
4. METHODOLOGY FOR COMPARING FIXATION POINTS.

We handpicked a set of 85 vertices in the polygon image to test the algorithm. It seemed impractical to handpick vertices for the lighthouse image because of its complexity. In addition, we have compared the algorithm generated fixations with those of seven human subjects viewing each test image through an eyetracker. The experimental protocol is described in the appendix.

The KLD is not a true distance function, since it is not symmetric and does not obey the triangle inequality. However, it is a convex function and $D(p \| q) = 0$, if and only if $p(x)=q(x)$. We use the symmetrical distance

$$DS(p \| q) = \frac{1}{\frac{1}{D(p \| q)} + \frac{1}{D(q \| p)}} \tag{24}$$

to quantify the distance between our fixation predictions and the recorded eye fixations (now represented by the pseudo-dense fixation map).



(a) (b)

(c) (d)

Figure. 2. Example of calculation of subsequent fixations. (a) Original image lighthouse. (b) Foveated version by space-varying Gaussian filtering. (c Foveated edge map by foveated Canny edge detection. (d) Foveated curvature map by foveated Kitchen-Rosenfeld curvature computation.

In order to evaluate the algorithm, we needed a methodology for comparing different sets of fixation points, based upon the fact that perfect matches are rare. We chose a method loosely based on that of Rajashekar, Cormack, and Bovik[4]. The first step is to replace each fixation point by a symmetrical 2D Gaussian envelope. Overlapping distributions are summed using the formula, $1-(1-p)(1-q)$, which represents the probability of either p or q taking place, assuming independence. This process is illustrated in figure (4a), which illustrates Gaussian envelopes centered about a set of fixation points, with overlapping envelopes summed. The variance of the Gaussian is chosen so that its full width at half-max (FWHM) reflects the diameter of the foveola, about 1º visual angle or 30 pixels. This approximation of a fixation point accounts for the uncertainty of its exact location. By spreading a fixation around, this approximation also encompasses any small errors in calibration, accuracy and/or precision of the eye movement measurement. Finally, replacing each fixation point by a 2D Gaussian envelope reflects the probability that a neighborhood region around the fixation could have been selected as a fixation point. Next, we compare pairs of probability density images using a modified Kullback Leibler distance (KLD) . The KLD is a measure of relative entropy between two functions

$$D(p \parallel q) = \Sigma_{x \in X}\, p(x) \log\left(\frac{p(x)}{q(x)}\right) \qquad (23)$$

.

5. RESULTS

Figure (3) shows lighthouse and polygon fixations for three subjects with the threshold, t is set to .2. The polygon results show 85 handpicked, 85 algorithmic and about 50 subject fixations. Lighthouse figures show 50 algorithmic fixations, and about 50 human subject fixations.

Table I provides KLD values of algorithm vs. eyetracker generated fixations for each of seven subjects. It also has a KLD value for polyhedron handpicked vs algorithm generated fixations. The handpicked points were generated by one of the authors before the algorithm was completed and have not been edited since. Values were calculated using the symmetricrical KLD shown in (33).

The KLD between 85 polygon algorithmic fixations and the handpicked was 3.621.

Each subject ran about fifty fixations on the eyetracker so the algorithm was run for fifty fixations to give the results shown in Table I. The lowest distance is between the polyhedron handpicked and algorithmic, demonstrating that the algorithm performs its intended task of finding corners well. The polyhedron handpicked vs. eyetracker comparison is usually worse than the handpicked vs algorithm. The eyetracker vs. algorithm is worse still, demonstrating that corners are incomplete predictors of visual fixations. Naturally, corners may be one of many different classes of features that attract fixations.

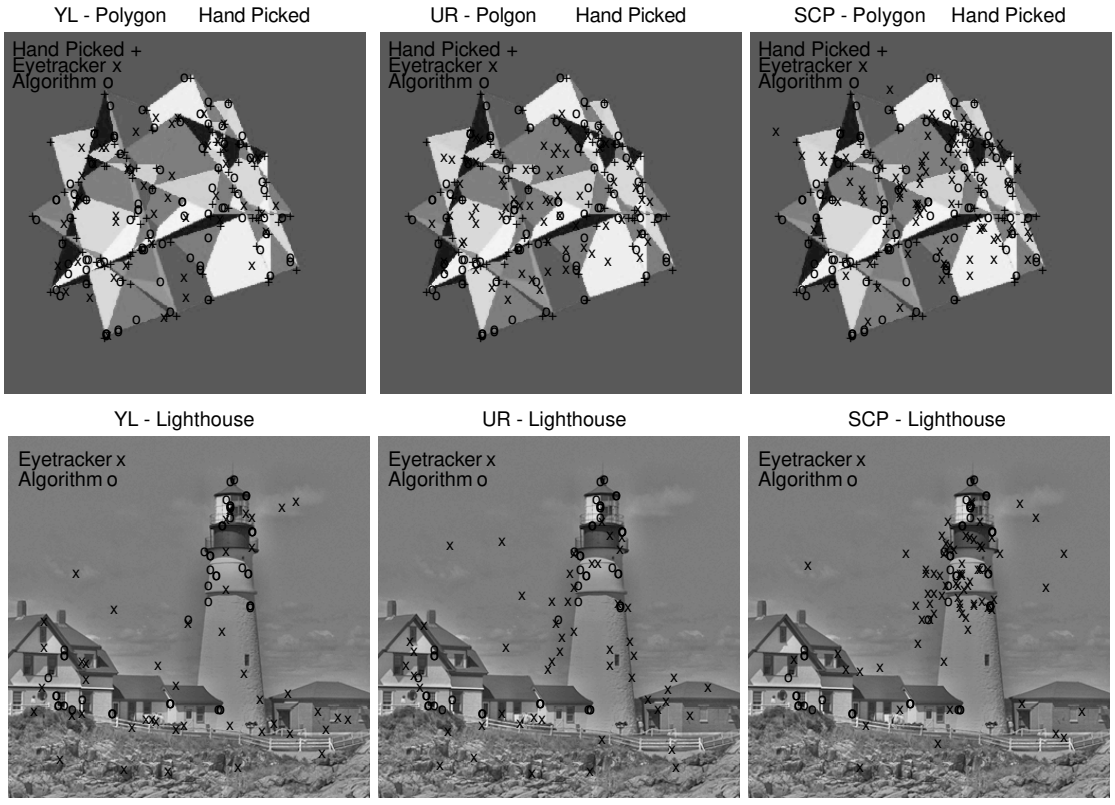


Figure 3. Fixation points for three subjects. Polygon figures show 85 handpicked fixations, 85 algorithmic fixations, and about 50 human subject fixations. Lighthouse figures show 50 algorithmic fixations, and about 50 human subject fixations.

Comparison			Subject				
	AT	BR	HHC	JSL	SCP	UR	YL
Polyhedron - Eyetracker vs. Hand Picked	10.225	7.894	7.420	8.569	9.411	10.339	7.170
Polyhedron - Eyetracker vs. Algorithm	11.069	8.903	7.496	9.513	9.753	9.621	5.903
Lighthouse - Eyetracker vs. Algorithm	10.180	12.356	16.551	10.137	13.081	15.506	15.017

Table I. KLD values for eyetracker vs. algorithm and eyetracker vs. handpicked 85 points.

To test the accuracy of the algorithm for finding corners, we calculated, for the polyhedron image, the distance from each algorithmic fixation to the nearest handpicked one, and repeated the process for all eyetracker fixations. If the minimum distance was less than or equal to a given tolerance, a "match" was declared. Figure (3b) show a comparison of matches as tolerance varies from zero to one degree. At zero tolerance, neither method shows matches. As tolerance increases, the handpicked matches increase much faster than the eyetracker ones. This further demonstrates that the algorithm locates corners far more accurately than human subjects.

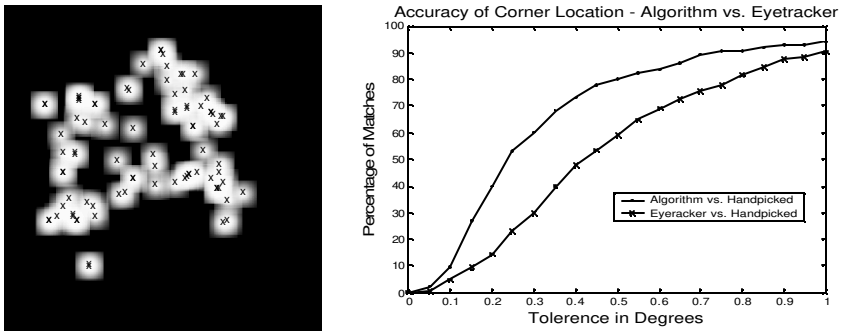


Figure (3). (a) Sample fixation points with Gaussian blur spots centered about them
(b) A comparison of the accuracy of corner locations, algorithm vs. eyetracker.

6. CONCLUSIONS

We have presented a new search strategy for locating corners in natural images. The method combines multiscale edge detection and space-variant, multiscale curvature detection with a multifixating, foveated search strategy. Such approaches to low-level vision problems that efficiently allocate visual resources according to local visual saliency have great promise in active vision, robotics, and general visual search.

APPENDIX. EXPERIMENTAL PROCEDURE

Seven observers, none of them familiar with the corner finding algorithm or the objectives of this work, were used for the experiment. All observers either had normal or corrected-to-normal vision. The stimuli consisted of two images: the lighthouse on the seashore, and the view of a polyhedron illustrated in Figure (1). The images were 1024 by 768 pixels and displayed on a 21 inch monitor pixels at a distance of 134 cm. from the observer. This set up corresponded to about 60 pixels/degree of visual angle, so the images extend 20.67 by 12.8 degrees. Observers were presented with each image for 30 seconds and instructed to view the displayed image as though they were observing a painting in a gallery. About 50 fixations were recorded for each observer. Human eye movements were recorded using an SRI Generation V Dual Purkinje eye tracker. It has an accuracy of < 10' of arc, precision of ~ 1' of arc and a response time of under 1ms. A bite bar and forehead rest was used to restrict the observer's head movements. The observer was first positioned in the eye tracker and a positive lock established onto the observer's eye. A linear interpolation on a 3 by 3 calibration grid was then done to establish the linear transformation between the output voltages of the eye tracker and the position of the observer's gaze on the computer display. The output of the eye tracker (horizontal and vertical eye position signals) was sampled at 200Hz and stored for offline data analysis.

ACKNOWLEDGEMENTS

Many thanks to Umesh Rajashekar for his help with the eyetracker and providing references.

REFERENCES

1. A. Srivatsava, M.I. Miller and U. Grenander, "Statistical models for bayesian object recognition," *The Handbook of Image and Video Processing* (A.C. Bovik, Ed.), pp. 1341-1354, 2005..
2. L.E. Wixson, Gaze Selection for Visual Search. PhD dissertation, University of Rochester, Computer Science Dept, 1994.
3. W. N. Klarquist and A. C. Bovik, "FOVEA: A foveated vergent active stereo vision system for dynamic three-dimensional scene recovery," *IEEE Trans. Robotics and Automation.,* vol. 14, no. 2, pp. 755-770, 1998.
4. C.M. Privitera and L.W. Stark, "Algorithms for defining visual regions-of-interest: comparison with eye fixations," *IEEE Trans. On Pattern Analysis and Machine Intelligence*, Vol. PAMI-22, pp. 970-982, Sept. 2000.
5. L.E. Wixson, "Using intermediate objects to improve the efficiency of visual search," *Int'l J. Computer Vision,* vol. 12, no. 2/3, pp. 209-230, 1994.
6. R.P.N. Rao and D.H. Ballard, "A multiscale filter bank approach to camera movement control in active vision systems," *SPIE* Volume 2354, pp. 105-116, 1994.
7. B. Moghaddam and A. Pentland, "Probabilistic visual learning for object detection," *Fifth Int'l Conf. Computer Vision*, Cambridge, MA, June 1995.
8. D. Marr, *Vision*. New York: W. H. Freeman and Company, 1982.
9. E. Kowler , "The role of visual and cognitive processes in the control of eye movement," in E. Kowler Ed *Eye Movements and Their Role in Visual and Cognitive Processes*, Amsterdam, Elsevier, pp. 1-63, 1990.
10. G. Sperling , "Comparison of perception in the moving and stationary eye," in E. Kowler Ed *Eye Movements and Their Role in Visual and Cognitive Processes*, Amsterdam, Elsevier, pp. 307-352, 1990.
11. K. Wiebe & A. Basu, "Modeling ecologically specialized biological visual systems," *Pattern Recognition* vol. 30 no. 10 pp. 1687-1703, 1997.
12. W. S. Geisler & M. S. Banks, "Visual performance," in *Handbook of Optics*, Vol. 1: Fundamentals, Techniques, & Design 2nd ed., M. Bass, Ed. New York: McGraw-Hill, 1995.
13. A. L. Yarbus, *Eye Movements and Vision*. New York: Plenum Press, 1967.
14. R, W, Rodieck, *The First Steps in Seeing*, Sunderland, MA: Sinauer Associates, Inc., 1998.
15. S. Lee and A. C. Bovik, "Fast algorithms for foveated video processing," *IEEE Trans. on Circuits and Systems for Video Technology*, 2003 pp. 149 -162, 2003.
16. P. T. Kortum and W. S. Geisler, "Implementation of a foveated image coding system for image bandwidth reduction," *SPIE* 2657, pp. 350-360, 1996.
17. Z. Wang, L. Lu and A. C. Bovik, "Foveation scalable video coding with automatic fixation selection," *IEEE Trans. on Image Processing*, vol. 12 pp: 243-254, 2003.
18. J. Alison Noble, Finding corners. *Image and Vision Computing*. vol 6, pp. 121-128, 1988.
19. R. Mehrotra, S. Nichani and N. Ranganathan, "Corner Detection," *Pattern Recognition*. vol. 23. pp.1223-1233. 1990.
20. J. Flynn and A.K. Jain, "On Reliable Curvature Estimation" , *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, San Diego, June 1989, pp. 110-116.
21. J Basak and D. Mahata "A Connectionist Model for Corner Detection in Binary and Gray Images," *IEEE Trans. on Neural Neworks*, vol. 11, pp 1124-1132, 2000.
22. H. Sossa and A. Palomino, "Model-based recognition of planar objects using geometric invariants," *Proceedings of the International Conference on Image Processing*, Lausanne, Switzerland, Sept. 16-19, 1996.
23. J. Qiang, R.M. Haralick, "Corner detection with covariance propagation," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, San Juan, Puerto Rico, June 17-19, 1997.
24. D. Marr and E. Hildreth. "Theory of edge detection," *Proc. R. Soc. Lond. B.*, vol. 207 pp. 187-217, 1980.
25. Z. Wang and A.C. Bovik, "Embedded foveation image coding," *IEEE Transactions on Image Processing*, vol. 10, no. 10, pp. 1397-1410, October 2001.
26. W. S. Geisler, and J. S. Perry, "A Real-time foveated multiresolution system for low-bandwidth video communication," *Proc. SPIE* vol. 3299, Human Vision and Electronic Imaging III; B. E. Rogowitz, T. N. Pappas; Eds. Jul 1998, pp 294-305.
27. J. F. Canny, "Finding Edges and Lines in Images," *AI Lab MIT Technical Report* AD-A130824, 1983.
28. L. Kitchen and A. Rosenfeld, "Gray level corner detection," *Pattern Recognition* Lett. vol. 1, pp. 95-102, 1982.
29. C. Harris and M. Stephens. "A combined corner and edge detector," *Fourth Alvey Vision Conference*, pp. 147-151, 1988.
30. S. M. Smith and J.M. Brady. "SUSAN - a new approach to low level image processing," *Int. Journal of Computer Vision*, vol. 23, pp. 45-78, 1997.
31. F. Mokhtarian and R. Suomela. "Robust Image Corner Detection Through Curvature Scale Space," *IEEE Trans. Pattern Anal. Mach. lntell*, vol. 20, pp. 1376-1381, 1998.
32. E. W. Weisstein. "Relative entropy." From MathWorld http://mathworld.wolfram.com/RelativeEntropy.