SCS&E Report 9318
December, 1993

Designing a Video Rate Edge Detection ASIC

Mehdi N. Fesharaki and Graham R. Hellestrand

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
THE UNIVERSITY OF NEW SOUTH WALES

Abstract

A method of generating a video rate edge maps, and thereby segmenting an image into regions based on the Kolmogorov-Smirnov test is presented. By applying this test and comparing cumulative distribution functions of the intensities in the neighbourhood of a given pixel, the pixel can be accurately classified as either an edge pixel on the boundary between regions, or as a pixel belonging to a particular type of region. It is shown that a custom VLSI design for this algorithm using parallel pipelined architecture is realisable. The outline of this design is presented and then the critical path modules are simulated.

Contact: mehdi@vast.unsw.edu.au
hell@vast.unsw.edu.au

Designing a Video Rate Edge detection ASIC

1. Introduction

Edge detection is a critical element in computer vision, since edges contain a major fraction of image information. For example, according to Marr [11], stereo vision operates not on a pair of images *per se* but on the tokens that arise during, or as the results of, edge detection. The function of edge detection is to identify the boundaries of homogeneous regions in an image based on properties such as intensity and texture.

Despite two decades of research, computer vision is only recently being exploited by the industry. One of the major reasons is that for large images (typically $\geq 512 \times 512$ pixels) with many operating to be performed on each pixel at video rate (25 or 50 frame per second), a huge volume of operations need to be carried out in a limited time. To realise real-time vision systems, a trade off between algorithmic complexity and feasible fast hardware structures is required. In general, more robust algorithms need higher complexity operations, and therefore, there is less chance for these to be performed in real time.

For different levels of support for computer vision, different ASIC architectures have been proposed [4], in which, the various algorithmic properties allow realisation in silicon having different properties of speed, power dissipation, area, and I/O configurations. The simplest way to realise a real time edge detection chip is to use binary images [1], with the loss of considerable information in the image. An alternative is to use grey scale images. Many edge detection algorithms have been developed based on the computation of the intensity gradient vector of grey images. The Sobel and compass operators are examples. These algorithms, in general, are sensitive to noise and speckle in the image but their computations are simple. Plessey's CMOS edge detector chip, the PDSP 16401, implements the compass operator for 10-bit grey scale images. This chip supports the basic operations and also delivers a 13-bit word that gives a measure of how sharp the edges are [2]. Kanopoulos et al [9] describe the implementation of the Sobel operator. To overcome the inherent problems in the gradient based operators, Lee et al [10] store the gradient amplitudes in off-chip memories and then, in a recursive manner, detect and trace the edges.

An alternative to the use of ASIC design for edge detection, is the use of more general purpose chips designed for image processing. For example Ruetz and Brodersen [12] designed a set of eight chips consisting of a convolver, a sorting filter, etc. Ruff's implementation [13] of Canny's edge detector operating at video rate is an example of using these chips. His use of buffered, pipelined processors is a typical architecture in early vision processing. Ruff demonstrated how a state-of-the-art edge operator can be realised using special-purpose parallel hardware.

An alternative to gradient based techniques is a statistical approach, where the distribution of intensity values in the neighbourhood of a given pixel is computed to determine if the pixel is to be classified as an edge. In the literature, less attention has been paid to statistical approaches. However, this method has been approached by some researchers [3, 15]. The general scheme of statistical edge detection algorithms is usually based on comparing two adjacent samples obtained from the image data. Then the following hypotheses are formulated:

H_0: Both samples belong to a homogeneous region.
H_1: The samples belong to more than one homogeneous region.

If the null hypothesis is rejected, an edge is declared between the two samples.

We have developed a new edge detection algorithm based on non-parametric statistical methods. It has been shown [5] that this algorithm is robust in countering additive noise, including Gaussian and impulse noise. Another advantage of this algorithm is that the simple mathematical operations used in this algorithm, lend themselves to a parallel pipelined architecture. In this paper, we present the algorithm and how it can be realised as an ASIC for real time edge detector design. Real time edge detection for 8-bit grey scale images occurring at 25 frames per second, requires a 7 MHz chip. To show that this is achievable, first, we realise the algorithm as a parallel pipelined architecture and then design the critical path modules and simulate them prior to realisation. It is shown that a real time edge detector chip with a good margin for increase in speed is realistic. In the next section, the edge detection algorithm is explained briefly, and in the following sections, the implementation of this algorithm is detailed.

2. The edge detection algorithm

To implement the edge detection algorithm, a 5×5 window is scanned over the image. Since there is no *a priori* knowledge about the direction of edges, the window is partitioned into two parts (X and Y samples) in four different orientations shown in Figure 1, in order to identify an edge.

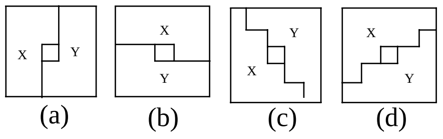

Figure 1. The four orientations used for declaring edge detection

In each window, we obtain N = m + n observations $X_1, \cdots, X_m$ and $Y_1, \cdots, Y_n$. Based on the Kolmogorov-Smirnov test, the null hypothesis that the X and Y samples are from the same population against the alternative that they are from different population is tested. For this purpose, we may write the null hypothesis as

$$H_0 : P(X \leq a) = P(Y \leq a), \quad \text{for all } a.$$

To verify this hypothesis, the cumulative distribution function, *cdf,* of two samples are compared. To achieve this, the N observations from the two samples are ordered to form the set $\{Z_{(i)}$: i = 1,...,N}, where $Z_{(1)} \leq Z_{(2)} \leq \cdots \leq Z_{(N)}$. The statistic J is defined as follows [7]:

$$J = \frac{mn}{d} \max_{i=1,\ldots,N} \left\{ \left| F_m(Z_{(i)}) - G_n(Z_{(i)}) \right| \right\} \qquad (1)$$

where m and n are as before, d is the greatest common divisor of m and n, and

$$F_m(a) = \frac{\textit{number of X's} \le a}{m} \quad \text{and} \quad G_n(a) = \frac{\textit{number of Y's} \le a}{n}$$

are considered to be the empirical cumulative distribution functions for two random samples of sizes m and n from two distributions with *cdf F(x)* and *G(y).* We can simplify the calculation of J, and eliminate sorting by partitioning the range of the N observed data into ρ equal sub-ranges with length τ, as follows:

$$\tau = \frac{Z_{(N)} - Z_{(1)}}{\rho}.$$

Supposing ρ = 4, the following set can be defined:

$$\theta = \left\{ Z_{(1)} + \tau \ , \ Z_{(1)} + 2\tau \ , \ Z_{(1)} + 3\tau \right\} \quad \textit{(2)}$$

With a 5×5 window divided into two equal sub-regions, and excluding the center pixel, we get m = n = d = 12. Now Equation (2) may be rewritten as follows:

$$J = \max_{\theta_k \in \theta} \left\{ \left| F_m(\theta_k) - G_n(\theta_k) \right| \right\}, \qquad (3)$$

where k = 0, 1, 2, and $F_m(\theta_k)$ and $G_n(\theta_k)$ are defined as follows:

$$F_m(\theta_k) = \textit{number of X's} \le \theta_k \qquad (4)$$

$$G_n(\theta_k) = \textit{number of Y's} \le \theta_k \qquad (5)$$

From Figure 1, if an edge exists in the window, at least one of the partitions is matched with the direction of the edge and therefore, results in the greatest *J*. For each pair shown in Figure 1, the statistic *J* is computed in parallel. Then all *J*'s are ordered to form the set $\{J_{(i)}$: i = 1,...,4}, where $J_{(1)} \le \cdots \le J_{(4)}$. Therefore, for the two-sided Kolmogorov-Smirnov α-level test (viz H_0 versus the broad alternatives that H_0 is not true)

$$\begin{array}{r l} \textit{reject } H_0 \ \text{ if } & J_{(4)} \ge j(\alpha,m,n) \\ \textit{accept } H_0 \ \text{ if } & J_{(4)} < j(\alpha,m,n) \end{array}$$

where $j(\alpha,m,n)$ is extracted from the Table A_{23} in [7]. Due to the nondecreasing nature of the cumulative distribution function, noise and speckle are suppressed significantly. However, for further improved noise suppression, the first and second greatest of the Kolmogorov-Smirnov statistics may be tested against desired thresholds.

3. *Hardware Realisation*

To implement a real time edge detection chip, the Kolmogorov-Smirnov algorithm is realised in a pipelined fashion. To obtain real-time processing, each stage has its time constraints which affects the architecture, logic, and technology used to realise the chip. Therefore, we need to exploit the parallelism implicit in the algorithm to overcome the video frame refresh constraint of 40ms per frame. The hardware schematic of the ASIC is shown in Figure 2.

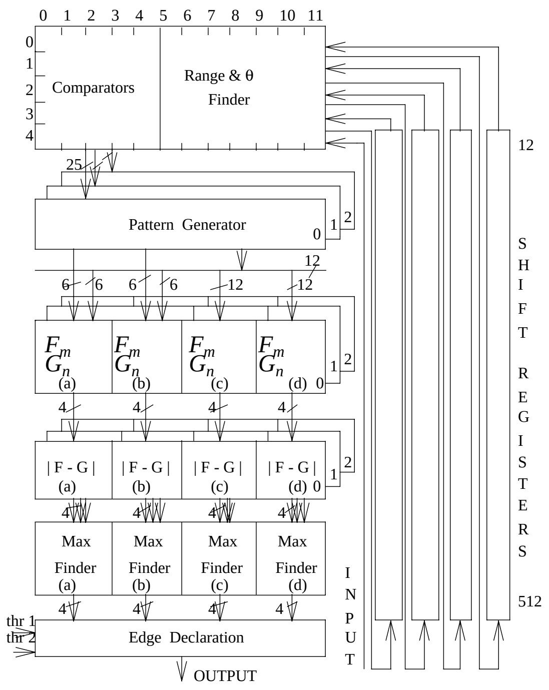

Figure 2. The overal scheme of the hardware design

The basic principle of a pipelined operation is to buffer the minimum amount of data and to perform all computations based on one data sample before the next input sample arrives. Since a 5×5 window is used to implement this algorithm, four rows of the image are stored in shift registers as shown in the right part of Figure 2. As illustrated, the algorithm is mapped to the various stages of a pipeline. The basic operations of each stage involve mainly addition, subtraction, comparison, and logic decision. These operators are repeated to form the required functions.

3.1 Stage 1: Range & θ *Finding*

All cells (registers) are indexed by their rows and columns as $C_{i,j}$. At each frame time (t_0), the procedure of edge detection begins within the 5×5 window centered at $C_{2,2}$. The computation determines whether the center pixel is an edge or non-edge element. The first step in implementing this algorithm is to calculate the range of data within the 5×5 window and thereby calculate the θ set in Equation (2). To achieve video rate edge detection, the θ set needs to be ready at time t_0, therefore, the computation of the range needs to be started before t_0. This block, named *Range &* θ *Finder,* is located at the top of Figure 2. The detailed operations of this block are shown in Figure 3.
As seen from this figure, the operations start from column 11 at time t_{-11}, and range calculation is completed at time t_{-1}. At time t_{-1}, the maximum and minimum observations are stored in the max and min registers and then, the set $\theta = \{\theta_0, \theta_1, \theta_2\}$ in Equation (2) is computed at time t_{-1} as follows:

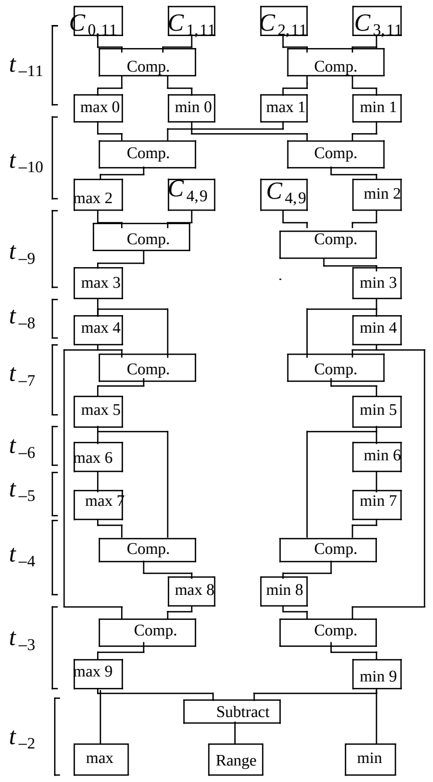

Figure 3. The schematic diagram of Range computaion

$$t_{-1}\left[\begin{array}{l}\theta_0 = Range \gg 2 + min\,;\\ \theta_1 = Range \gg 1 + min\,;\\ \theta_2 = \max - Range \gg 2.\end{array}\right.$$

where the notation $'' >> i\,''$ means shift right operation of i times.

3.2 Stage 2: Comparators

In this stage, all observations within the 5×5 window at time t_0 are compared with the set θ in parallel. Therefore, for each cell, three outputs are generated labeled γ_{ijk}, where i = 0, ..., 4, j = 0, ..., 4, and k = 0, 1, 2, and γ_{ijk} is defined as follows:

$$\gamma_{ijk} = \begin{cases} 1 & \text{if } C_{ij} < \theta_k \\ 0 & \text{if } C_{ij} \geq \theta_k \end{cases}$$

The output of this stage consists of 25 buses, each three bits wide.

3.3 Stage 3: Pattern Generation

By specifying γ_{ijk} in the previous stage, to compute the $F_m(\theta_k)$ and $G_n(\theta_k)$ of Equations (4) and (5) respectively, the γ_{ijk} set for the two samples of X and Y are counted (added) in parallel. The result of this counting (addition) shows how many of the pixels of those samples are less than the θ_k, where k = 0, 1, 2. To compute $F_m(\theta_k)$ and $G_n(\theta_k)$, for each orientation shown in Figure 1, the volume of operations can be reduced significantly by using the common patterns used to compute the above two functions for each orientation. To achieve this, based on sub-patterns consisting of three pixels as shown in Figure 4.a, four patterns used commonly in all partitionings in Figure 1, are constructed. These patterns are shown in Figure 4.b-e.

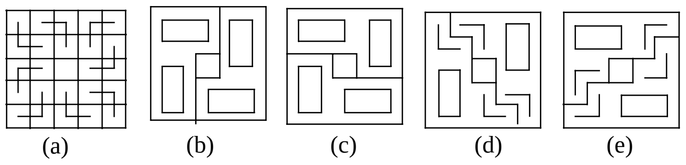

Figure 4. The patterns used commonly in all partitionings

3.4 Stage 4: F and G calculation

In this stage, $F_m(\theta_k)$ and $G_n(\theta_k)$ are computed in parallel. For this purpose, depending on the type of orientation, a through d, the related patterns shown in Figure 4 are added together to generate $F_m(\theta_k)$ and $G_n(\theta_k)$.

3.5 Stage 5: $|F(\theta_k) - G(\theta_k)|$ computation

According to Equation (3), the difference between the two cumulative distribution functions needs to be computed first. At the previous stage, the *cdf* of each sample X and Y is computed, and in this stage their difference is calculated using each member of the θ set and for all four orientations. These computations are done in parallel.

3.6 Stage 6: Maximum difference calculation

According to Equation (3), the maximum difference between the two cumulative distribution functions is the criteria for the homogeneity of the two samples. Therefore, in this stage, the maximum differences between the cdf of the two samples is computed for each partitioning shown in Figure 1.

3.7 Stage 7: Edge declaration

In the final stage, the responses to the different partitionings of the window are sorted. Obviously, the partitioning with the highest response shows the direction of the edge. In order to enhance noise suppression, we need to know the second highest response as well. Therefore, all responses are sorted based on shuffle sorting logic, and the first and second maximum responses are tested against the thresholds chosen at the desired significance level. Also, the direction of the highest response gives the direction of the edge.

4. VLSI Implementation

The implementation takes a two phase approach. First, this algorithm was described using a hardware description language developed in our Laboratory, named MODAL [6, 8], and simulated with an event-driven logic simulator. Simulation tests two aspects of the designed hardware, functional and timing. In addition to the accuracy of the functionality of the proposed hardware, the result of this simulation shows that the algorithm will run with a 100 MHz clock cycle, independent of the wiring delays and other fabrication issues. The longest delay in this model corresponds to the 8-bit adder / comparator modules, and when real delays are considered, it is clear that the speed estimate is optimistic. In the second phase, the floor plan of the chip is presented and then some modules are simulated.

4.1 Floor planning

The floor plan of the chip is shown in Figure 5.

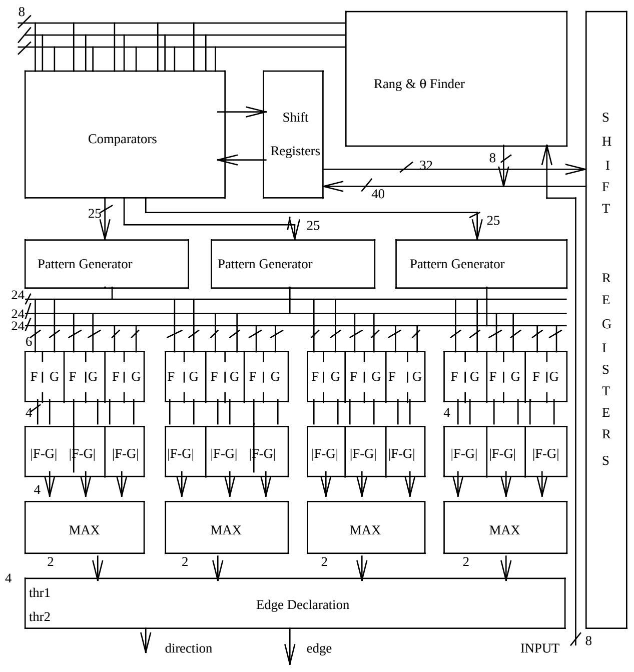

Figure 5. The floor plan of the edge detector chip

Since the algorithm can be implemented completely in a pipelined manner, the floor plan is straight forward. The output of each stage is directly connected to the input of the next stage. This reduces significantly the problem of routing which is one of the major contributors to delays in VLSI chips. In each stage, a simple module is involved which can be repeated. This is an added advantage of designing, simulating, and testing the chip. The use of iteration to form arrays of identical cells is an example of exploiting of regularity in IC design which simplifies the design process [14].

Across the whole pipelined design, the worst stage with respect to timing issues is the 8-bit adder / comparator. The rest of stages which involve a maximum of 4-bit operations, incur less delays.

4.2 8-bit Adder / Comparator

To realize the 8-bit adder / comparator, a domino 4-bit carry look ahead circuit is used. The two 8-bit inputs are divided up into two equal 4 bits. The input carry to the higher significance 4-bits is generated by a domino CMOS gate. The 8-bit comparator is realised by cascading two 4-bit domino carry look ahead circuits. The output carry of the second stage is interpreted as the greater than or equal to (≥) signal. To realise the comparator block shown in Figure 5, the floor plan shown in Figure 6, based on each comparator cell is used. This block was designed and simulated using a double metal, 2μ, n-well CMOS technology. The result of the analog simulation shows a worst case 10ns signal delay time for this block. The geometric layout of this block is shown in Figure 7. For the adder / subtractor, the domino carry look ahead circuit is used as well.

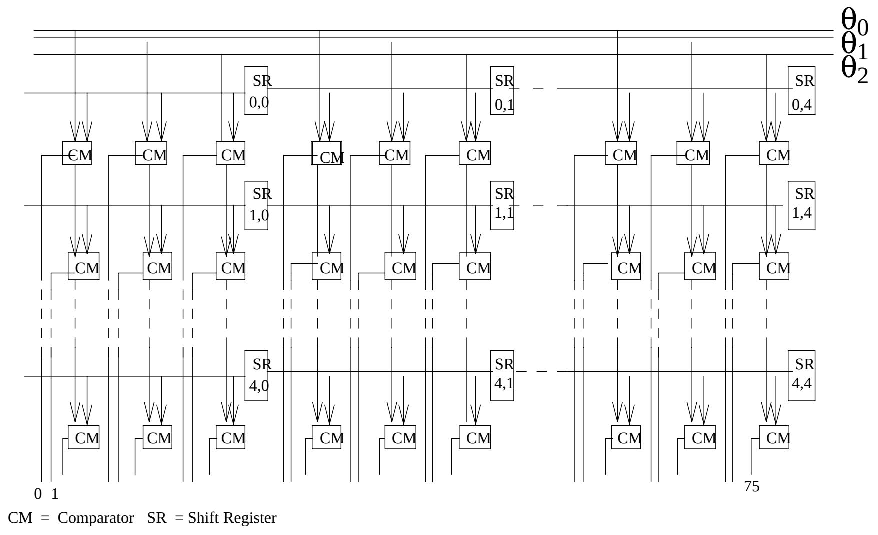

Figure 6. The floor planning of the comparator module

The postscript of the geometric layout of this report (Figure 7) requires too much memory to be placed here. If some one is interested to get a copy of the file, please contact the authors.

Figure 7. The geometric layout of the comparator module

4.3 Investigation

The result of the simulation for the 8-bit comparator block shows a 10ns signal, which is the slowest of the pipelined stages. To achieve an edge map for an image with a 512×512 resolution refreshing at 25 frames per second, in real time, requires a 150ns throughput for each pixel, which dictates a 7MHz period for the running of the chip. The simulation of the designed circuit will easily meet this target. Migrating the design to a state-of-the-art, sub-micron technology will provide considerable, further speed enhancement.

5. Conclusion

This paper presents a parallel pipelined architecture for the detection of edges in 8-bit grey scale images with 512×512 resolution refreshing at video rate. The procedure of edge declaration for each pixel starts as soon as the pixel values are available within a 5×5 window surrounding the pixel. This window is partitioned so that it provides two samples in four orientations in order to detect edges. The Kolmogorov-Smirnov test is applied to each orientation in parallel and the first and second greatest Kolmogorov-Smirnov statistics are compared with thresholds values determined by the desired significance level. Since this algorithm lends itself to a parallel, pipelined hardware implementation, and no complex operations are needed, a video rate ASIC edge detector design is realisable. The worst case timing in the pipeline is due to the adder / comparator module. The analog simulations of these two phase clocked modules, realised using a double metal, n-well 2μ CMOS process shows a worst case signal delay of less than 10ns. The presented design can produce edge maps for 512×512 images at the desired frame rate. The critical components of this design have been scheduled for fabrication.

REFERENCES

1. I. Agi, P.J. Hurst, and A.K. Jain, ‘‘An expandable vlsi processor array approach to contour tracing,’’ *Proc. IEEE Int. conf. Acous., speech, signal processing*, pp. 1969-1972 (1988).

2. M. Beedie, ‘‘Image IC detects edges in real time,’’ *Electronic Design*, pp. 57-58 (1986).

3. A.C. Bovik, T.S. Huang, and D.C.M. JR, ‘‘Nonparametric tests for edge detection in noise,’’ *Pattern Recognition* **19**, pp. 209-219 (1986).

4. F. Catthoor and H. DeMan, ‘‘Customized architectureal methodologies for high-speed image and video processing,’’ *Proc. IEEE Int. conf. Acous., speech, signal processing*, pp. 1985-1988 (1988).

5. M.N. Fesharaki and G.R. Hellestrand, ‘‘A comparative approach towards statistical based edge detection,’’ *Technical Report*, Univ. of NSW, Australia (1993).

6. G.R. Hellestrand, ‘‘MODAL: A system for digital hardware description and simulation,’’ *j. Digital Systems*, pp. 241-303 (1979).

7. M. Hollander and D.A. Wolfe, *Nonparametric statistical methods,* Jhon-Wiely (1973).

8. M.C. Kam, ‘‘Tutorial on Using the MODAL Compiler and Simulation System,’’ *Internal Document, VLSI and Systems Technology Laboratory, UNSW.* (1988).

9. N. Kanopoulos, N. Vasanthavada, and R. Baker, ‘‘Design of an edge detection filter using the Sobel operator,’’ *IEEE J. Solid-State Circuits* **23** (1988).

10. C. Lee, F. Catthoor, and H.J. DeMan, ‘‘An efficient ASIC architecture for real-time edge detection,’’ *IEEE Trans. Circuits and Systems* **36**, pp. 1350-1359 (1989).

11. D. Marr, *Vision : a computational investigation into the human representation and processing of visual information,* Freeman (1982).

12. P.A Ruetz and R.W. Brodersen, ‘‘Architectures and design techniques for real-time image processing IC’s,’’ *IEEE J. Solid-State Circuits* **sc-22**, pp. 233-250 (1987).

13. B.P.D. Ruff, ‘‘A pipelined architecture for a video rate Canny operator used at the initial stage of a stereo image analysis system,’’ *Parallel Architectures and Computer Vision Edited by I. Page,* pp. 171-185, Clarendon Press (1988).

14. N. Weste and K. Eshraghian, *Principles of CMOS VLSI Design,* Addison-Wesley (1985).

15. Y. Yakimovsky, ‘‘Boundary and object detection in real world images,’’ *j. ACM* **23**, pp. 599-618 (1976).