NEW CORNER DETECTION ALGORITHM BY TANGENT AND VERTICAL AXES AND CASE TABLE

Soo-Chang Pei, Jian-Jiun Ding,
Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan, R.O.C
Email address: pei@cc.ee.ntu.edu.tw

ABSTRACT

In this paper, we introduce a new algorithm for corner detection. Instead of calculating the gradients along x and y-axes, which is the common step of many existed algorithms, we use the sum of differences to observe the variations along the adaptive vertical and tangent axes. We classify the variations into 36 types and use the 'case table' to determine whether a pixel is a corner. We do some experiments and show that our algorithm can detect almost all the corners of a complicated natural image (such as Lena image and Fruit image) successfully. In addition to corner detection, it is also possible to use our algorithm to detect the edges, ridges, valleys, isolated dots, saddles, and plain regions of a natural image.

1. INTRODUCTION

Corner detection is very important for image processing. To describe an object, the most economical way is to use corner features. It requires very less storage. Moreover, since the distances (or their ratios) among the corners are invariant to shifting, rotation, reflection, and scaling, the corner feature is also helpful for shifting, rotation, reflection, and scaling invariant pattern recognition.

In literature, there are some corner detection algorithms [1]-[5]. Although these algorithms are different, almost all of them have a common step: measuring the intensity gradients along *x*-axis and *y*-axis.

Although some corner detection algorithms have been developed, however, detecting the corner of a natural image is still a challenging work. At least, from what we know, until now no algorithm can well detect the corners of Lena image and some other popular test images. Due to some reasons, detecting the corners of a natural image is a hard work. First, it is inevitable that a natural image is affected by noise and distortion. Blurring decreases the gradient, and noise may increase or decrease the gradient. Both of them deteriorate the performance of corner detection. In addition, in nature, there are a variety of corners. Using the information obtained from the gradients along *x*- and *y*-axes is not enough to detect any type of corner.

In this paper, we develop a new algorithm for corner detection. The goal of our algorithm is to detect the corners of a complicated natural image successfully. Our algorithm is partially similar to the algorithm based on gradient measurement, but we use a more effective way to observe the intensity variations (illustrated in detail in Sec. 2 and Sec. 3). From the experiments in Figs. 4~6, we show that our algorithm can successfully detect the corners, even if the input image is as complicated as Lena image.

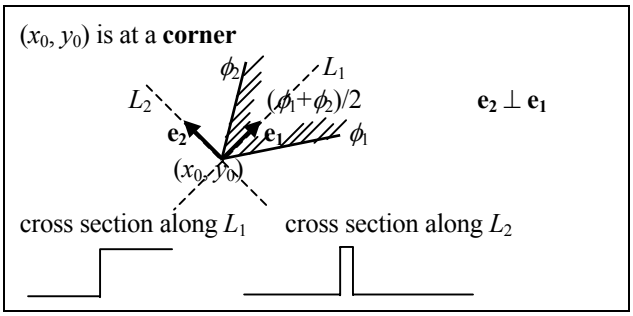

Fig. 1 Using the variations along vertical and tangent lines (L_1 & L_2) to determine whether a pixel is a corner.

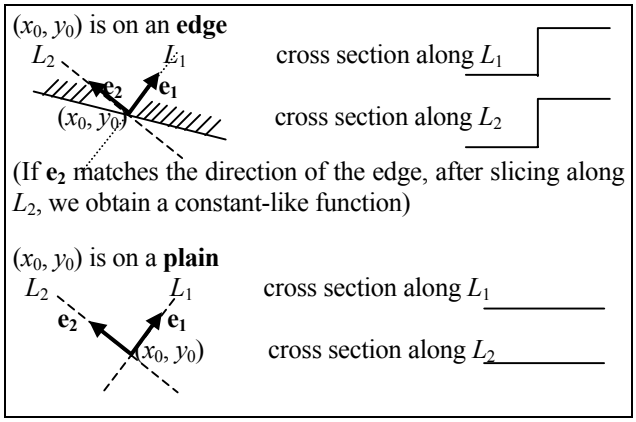

Fig. 2 If (x_0, y_0) is not a corner, after slicing, we can't obtain the results as in Fig. 1.

2. AN EFFECTIVE WAY TO DISTINGUISH AMONG CORNERS, EDGES, AND PLAINS

Our algorithm is based on the following concept. If $f(x, y)$ has a corner at (x_0, y_0), as in Fig. 1, then there exists an axis $\mathbf{e_1}$ such that

(1) If we slice $f(x, y)$ along the line $L_1 = (x_0, y_0) + a\mathbf{e_1}$ ($a \in R$), we obtain a step-like function.

(2) If we slice $f(x, y)$ along the line $L_2 = (x_0, y_0) + a\mathbf{e_2}$ ($a \in R$) where $\mathbf{e_2} \perp \mathbf{e_1}$, we obtain a ridge-like function.

0-7803-9134-9/05/$20.00 ©2005 IEEE

By contrast, if (x_0, y_0) is not a corner, when we slice $f(x, y)$ along the directions of $\mathbf{e_1}$ and $\mathbf{e_2}$, we can't obtain the above results. For example, if (x_0, y_0) is on an edge, after slicing, we obtain step-like functions for both the two directions (or obtain a step-like function along L_1 and obtain a constant-like function along L_2). If (x_0, y_0) is on a plain, after slicing, we obtain constant-like functions for both the two directions, as in Fig. 2. Thus using the variations along two perpendicular axes is a good way to conclude whether a pixel is a corner.

Our algorithm is similar to the algorithms based on gradient measurement [1]-[5]. However, there are some key differences. The existed algorithms calculate the first order partial differentiations with respect to x and y:

$$d_x = \partial f(x, y)/\partial x, \quad d_y = \partial f(x, y)/\partial y, \tag{1}$$

then use a function of d_x and d_y to measure the cornity of a pixel and conclude whether a pixel is a corner.

By contrast, in our algorithm,

(1) Instead of observing the variations along x and y-axes, we observe the variations along the directions of $\mathbf{e_1}$ and $\mathbf{e_2}$ and $\mathbf{e_1}$ and $\mathbf{e_2}$ varies with the localized character of the image. Using adaptive axes can improve the accuracy of corner detection, since in an image there are usually many corners whose orientations are far from 0 (the direction of x-axis) and $\pi/2$ (the direction of y-axis).

(2) Instead of doing the first order differentiation, we measure the variations along the directions of $+\mathbf{e_1}$, $-\mathbf{e_1}$, $+\mathbf{e_2}$, and $-\mathbf{e_2}$. If we do the first order differentiation of $\partial/\partial\mathbf{e_i}$ (i = 1 or 2), we can observe the variations along $\mathbf{e_i}$ (the variations $+\mathbf{e_i}$ and $-\mathbf{e_i}$ are not observed separately). However, to well distinguish between a corner and an edge, we suggest that the variations along $+\mathbf{e_i}$ and $-\mathbf{e_i}$ should be observed separately. If we do the first order differentiation, only three types of variations can be observed: ① rising, ② falling, and ③ remaining as a constant. The information is too less to well determine whether a pixel is a corner. However, if we observe the variations along $+\mathbf{e_i}$ and $-\mathbf{e_i}$, then, along the line L_i, ($L_i = [m, n] + \sigma\cdot\mathbf{e_i}$) there are 6 types of variation, there are 6 types of variation can be observed:

① $(-, -) \rightarrow$ ridge, ② $(-, 0) \rightarrow$ step (falling side)
③ $(-, +) \rightarrow$ step (middle) ④ $(0, +) \rightarrow$ step (rising side)
⑤ $(0, 0) \rightarrow$ constant, ⑥ $(+, +) \rightarrow$ valley.

We illustrated it in detail in subsec. 3-2. Since more types of variation L_i can be observed, the precision of corner detection. Using it together with the 'case table' (see Table 1), we can well determine whether a pixel is a corner.

(3) Doing differentiation may magnify the effect of noise. This is another reason why we don't use differentiation. Thus, instead of doing the differentiation, we use the sum of difference (see (12)), which is more resistant to noise, to measure the variations.

3. PROPOSED ALGORITHM

(**Step 1**) Use the phase mask to determine the vertical axis $\mathbf{e_1}$ and the tangent axis $\mathbf{e_2}$ for each of the pixels.
(**Step 2**) Then measure the variations along the directions of $+\mathbf{e_1}$, $-\mathbf{e_1}$, $+\mathbf{e_2}$, and $-\mathbf{e_2}$.
(**Step 3**) Use the results of Step 2 together with the 'case table' to determine whether the pixel can be treated as a candidate for corner.
(**Step 4**) Among the corner candidates obtained in Step 3, we select the ones whose 'scores' are larger than the neighboring corner candidates.

In the following four subsections, we illustrate each of the steps in detail.

3.1. Determining the vertical and tangent axes

Suppose that $f(x, y)$ has an ideal corner at (x_0, y_0) and surrounding the ideal corner $f(x, y)$ can be expressed as:

$$f(x, y) = k \quad \text{when } \phi_1 \le \phi \le \phi_2,$$
$$f(x, y) = 0 \quad \text{when } \phi < \phi_1 \text{ or } \phi > \phi_2, \tag{2}$$
$$\phi = \arg[(x-x_0)+j(y-y_0)],\ [(x-x_0)^2+j(y-y_0)^2]^{1/2} \le R.$$

That is, $f(x, y)$ is a corner function and the location of the corner is (x_0, y_0). If we convolve $f(x, y)$ with a phase mask:

$$f_1(x, y) = f(x, y) * M(x, y) \tag{3}$$

where $M(x, y) = -\exp(j\varphi)$ when $r \le R$, (4)
$M(x, y) = 0$ otherwise, $\varphi = \arg(x+jy)$, $r = (x^2+y^2)^{1/2}$,
then we can prove that

$$f_1(x_0, y_0) = \iint f(x_0 + x, y_0 + y)M(-x,-y)dxdy$$
$$= \int_0^R \int_{\phi_1}^{\phi_2} k\exp(j\varphi)2\pi r\,d\varphi dr = -j\pi kR^2\left(e^{j\phi_2} - e^{j\phi_1}\right)$$
$$= \pi kR^2 \sin((\phi_2 - \phi_1)/2)e^{j(\phi_1+\phi_2)/2}, \tag{5}$$

$$\arg[f_1(x_0, y_0)] = (\phi_1+\phi_2)/2. \tag{6}$$

Notice that $(\phi_1+\phi_2)/2$ is just the orientation of the corner. Thus, if we convolve an image with the phase-mask in (4) and take the phase part, at the corner pixel, we can explicitly obtain the orientation of the corner. Thus, in our algorithm, we treat $\arg[f_1(x, y)]$ as the vertical axis $\mathbf{e_1}$ and set the tangent axis $\mathbf{e_2}$ to satisfy $\mathbf{e_2} \perp \mathbf{e_1}$.

In digital implementation, the process of phase mask convolution can be rewritten as:

$$f_1[m, n] = f[m, n] * M[m, n], \tag{7}$$
$f[m, n]$: input image, $*$: convolution,
$$M[m, n] = -(m + jn)/(m^2+n^2)^{1/2} \tag{8}$$
$m, n, = -R \sim R$, R is some small integer.

For a 256×256 natural image, it is proper to choose R = 5~10. After $f_1[m, n]$ is computed, we can define $\mathbf{e_1}$ (the vertical axis) and $\mathbf{e_1}$ (the direction of the tangent axis) as:

$$\mathbf{e_1} = (\cos\theta_1, \sin\theta_1), \quad \mathbf{e_2} = (\sin\theta_1, -\cos\theta_1),$$
$$\theta_1[m, n] = \arg(f_1[m, n]). \tag{9}$$

3.2. Measure the variation along four directions

After $\mathbf{e_1}$ and $\mathbf{e_2}$ are determined, in Step 2, we measure the variations along the four directions: $\mathbf{e_1}$, $\mathbf{e_{-1}}$, $\mathbf{e_2}$, and $\mathbf{e_{-2}}$ ($\mathbf{e_{-1}} = -\mathbf{e_1}$ and $\mathbf{e_{-2}} = -\mathbf{e_2}$). First, we compute the coordinates of the

pixels on the directions of $\mathbf{e_i}$ ($i = \pm1, \pm2$). Suppose that, for each of the directions, we consider L pixels. For the pixel $[m, n]$, we use $[x_{h,i}[m, n], y_{h,i}[m, n]]$ ($i = \pm1, \pm2$, $h = 1\sim L$) to denote the coordinate of the pixel on the direction of $\mathbf{e_i}$. They can be computed from

$$x_{h,+1}[m,n] = m+p_h[m,n], \quad y_{h,+1}[m,n] = n+q_h[m,n],$$
$$x_{h,-1}[m,n] = m-p_h[m,n], \quad y_{h,-1}[m,n] = n-q_h[m,n],$$
$$x_{h,+2}[m,n] = m+q_h[m,n], \quad y_{h,+2}[m,n] = n-p_h[m,n],$$
$$x_{h,-2}[m,n] = m-q_h[m,n], \quad y_{h,-2}[m,n] = n+p_h[m,n], \quad (10)$$

where $p_h[m,n] = \text{round}\{h\cdot\cos\theta_1[m,n]/t[m,n]\}$, $h = 1\sim L$,

$$q_h[m,n] = \text{round}\{h\cdot\sin\theta_1[m,n]/t[m,n]\},$$
$$t[m,n] = \max\{|\cos\theta_1[m,n]|, |\sin\theta_1[m,n]|\}. \quad (11)$$

Then, we measure the variations along $\mathbf{e_i}$ ($i = \pm1, \pm2$) from:

$$V_i[m,n] = \sum_{h=1}^{L} G\left(f\left[x_{h,i}[m,n], y_{h,i}[m,n]\right] - f[m,n]\right), \quad (12)$$

where $G(\ )$ is some non-decreasing mapping function satisfying $G(a) = -G(-a)$. For example, we can choose it as:

$G(a) = a^r$ when $|a| < A$, $\quad G(a) = A^r$ when $a \ge A$,

$G(a) = -A^r$ when $a \le -A$, $\quad 0 < r \le 1$. (13)

Then we choose some threshold T ($T > 0$) and classify the status of variations into three types:

Case 1: $V_i[m,n] \ge T$ (denoted by +), (14)

$\rightarrow$ the gray-level grows larger along the direction of $\mathbf{e_i}$,

Case 2: $-T < V_i[m,n] < T$ (denoted by 0), (15)

$\rightarrow$ the gray-level changes less along the direction of $\mathbf{e_i}$,

Case 3: $V_i[m,n] \le -T$ (denoted by –), (16)

$\rightarrow$ the gray-level grows smaller along the direction of $\mathbf{e_i}$.

Thus, for each of the directions, there are three types of variations. We can combine the variations along $\mathbf{e_i}$ and $\mathbf{e_{-i}}$ (i = 1 or 2), then, along the line L_i, ($L_i = [m, n] + \sigma\cdot\mathbf{e_i}$) there are 6 types of variation:

① (–, –), ② (–, 0), ③ (–, +), ④ (0, +), ⑤ (0, 0), ⑥ (+, +).

Note that (0, –), (+, –), and (+, 0) are in fact the same as (–, 0), (–, +), and (0, +), respectively. (–, –) means that the cross section along L_i is a ridge-like function (since $V_i[m, n] \le -T$ and $V_{-i}[m, n] \le -T$). Moreover, for the case of (–, 0) or (–, +), we can conclude that the cross section along L_i is a step-like function.

3.3. Case table

We can use the status of variation obtained in subsection 3-2 to conclude whether a pixel is at a corner, on an edge, or on a plain. For example, from Fig. 1, we can see that, if at the pixel $[m, n]$, the status of variation is:

along L_1: (0, –), along L_2: (–, –), (17)

then $[m, n]$ is a corner. In addition, from Fig. 2, if the status of variation is $\{L_1$: (–, +), L_2: (–, +)$\}$, the pixel is on an edge. If the status of variation is $\{L_1$: (0, 0), L_2: (0, 0)$\}$, the pixel is on a plain. In subsection 3-2, we have illustrated that along the vertical or tangent line L_i (i = 1 or 2), there are 6 types of variation. Thus there are totally 36 types of variation. In Table 1, we show the 'case table'. With it and the status of variations obtained from subsection 3-2, we can conclude whether a pixel is at a corner, on an edge, on a ridge, at a saddle point, or on a plain.

In Table 1, the meaning of the junction type corner is shown in Fig. 3. It means that more than two regions intersect at the corner, as in Fig. 3. For the junction type corner, the phase $\theta_1[m, n]$ calculated from (9) will be the 'weighting average' of the principal axes of all the regions surrounding the junction. The variations along L_1 and L_2 will be (0, +) and (–, 0), respectively, as in Fig. 3. Note that all the three types of variation (+, 0, and –) appear. This is because there are three regions intersecting at $[m_0, n_0]$.

Table 1 Case table (L_1: vertical line, L_2: tangent line).

L_2 \ L_1	–, –	–, 0	–, +	0, 0	0, +	+, +
–, –	D	C	C	R	R	S
–, 0	D	E	C_J	P_E	C_J	R
–, +	Δ	C_J	E	Δ	C_J	Δ
0, 0	R	P_E	E	P	P_E	R
0, +	R	C_J	C_J	P_E	E	D
+, +	S	R	C	R	C	D

C: corner, C_J: junction type corner, D: dot
E: edge, R: ridge / valley S: saddle
P: plain P_E: plain (near edge) Δ: indeterminable

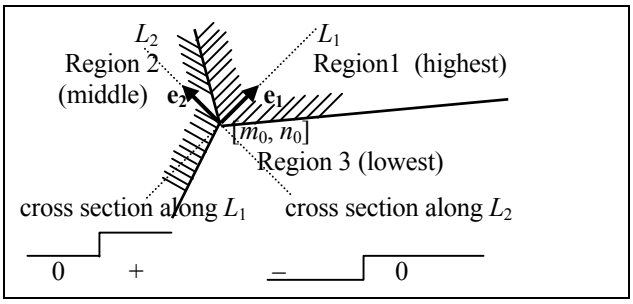

Fig. 3 The junction type corner (i.e., more than two regions are intersected at a pixel).

3.4. Finding the corners from candidate pixels

After looking for the case table, some pixels recognized as the corners. This is not the final result, since in the region surrounding a corner there is usually more than one pixel recognized as a corner. We should use some criterion to choose the corner among the corner candidates. There are several ways to define the criterion. For example, we can define $s[m, n]$ as the number of the surrounding pixels that satisfies $|f[m+\tau, n+\eta]-f[m, n]| >$ threshold, or define it as

$$s[m,n] = \sum_{i=\pm1,\pm2} |V_i[m,n]|, \quad (18)$$

then find the local peaks of it and choose them as corners.

4. ADDITIONAL ADVANTAGES

From the process introduced in Sec. 3, we can successfully find the corners of an image. In addition to detect the corner, our algorithm has some other additional advantages:

(1) The orientations of the corners can be determined at the same time. If $[m, n]$ is a corner, the direction of the vertical axis $\mathbf{e_1}$ is just the orientations of the corner.

(2) It is possible to conclude whether a corner is a junction type corner from our algorithm.

(3) The angle of the corner can be estimated by $\mathbf{e_1}$. If $[m, n]$ is a obtuse angle, the directions of $\mathbf{e_1}$ varies slowly around $[m, n]$. By contrast, for an acute angle, $\mathbf{e_1}$ varies fast.

(4) In addition to corner detection, from the case table (see Table 1), we can also use our algorithm with a little modification to do edge detection, ridge detection, saddle pixel detection, and plain region detection for an image.

5. EXPERIMENTS

In Figs. 4, 5, and 6, we do some experiments that use the algorithm we propose for corner detection. We use bright dots to show the corners we detect. In the four examples of Fig. 4, we successfully detect all the corners, even for the case where the noise exists. In Figs. 5, 6, the input is a natural image, Lena image and Fruit image. Although Lena image is very complicated and affected by blurring and distortion, however, we can still use our algorithm to detect almost all the corners of them. We also compare out method with Harris' algorithm in Fig. 6. It is shown that our proposed algorithm can reduce the probability of misunderstanding the pixels in the regions of edges, valleys, ridges, and local peaks as corners.

6. CONCLUSIONS

We introduced a new algorithm for corner detection. Our algorithm is based on determining the orientations of the adaptive vertical and tangent axes, $\mathbf{e_1}$ and $\mathbf{e_2}$, and observing the variations of brightness along $+\mathbf{e_1}$, $-\mathbf{e_1}$, $+\mathbf{e_2}$ and $-\mathbf{e_2}$. From the experiments, we show that the proposed algorithm can effectively detect the corners of images, even a complicated natural image. In addition, with some modification, we can also use our algorithm for edge detection, ridge detection, saddle pixel detection, and plain region detection.

7. REFERENCES

[1] C. Harris and M. Stephens, "A combined corner and edge detection", Proc. 4th Alvey Vision Conf., 1988, pp. 189-192.

[2] S. M. Smith and M. Brady, "SUSAN–a new approach to low level image processing", *Int. J. Computer Vision*, vol. 23, no. 1, 1997, pp. 45-78.

[3] A. Guiducci, "Corner characterization by differential geometry techniques", *Patt. Recogn. Lett.*, vol. 8, 1988, pp. 311-318.

[4] R. Mehrotra, S. Nichani, and N. Ranganathan, "Corner Detection", *Pattern Recog.*, vol. 23, 1990, pp. 1223-1233.

[5] H. Wang and M. Brady, "Real-time corner detection algorithm for motion estimation", *Image and Vision Computing*, vol. 13, no. 9, 1995, pp. 695-703.

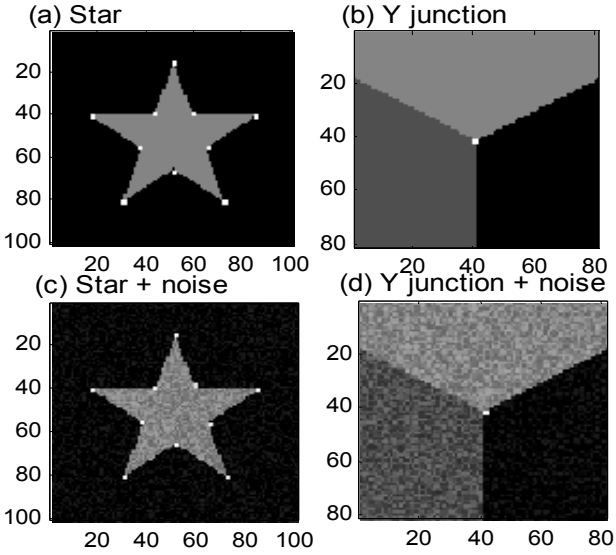

Fig. 4 The results of corner detection for (a) star, (b) Y junction, (c) star + noise, (d) Y junction + noise.

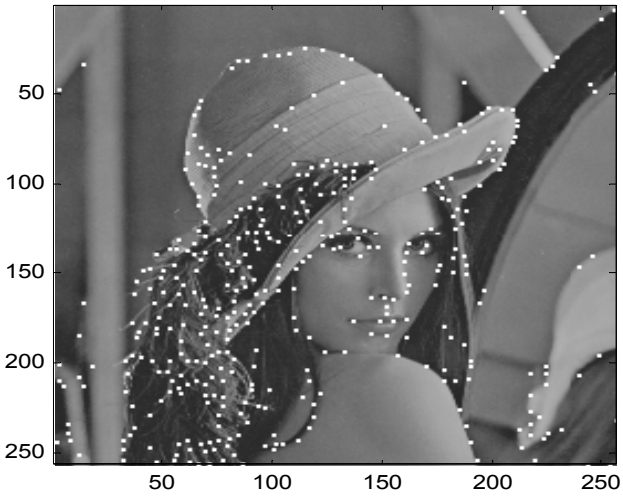

Fig. 5 Corners for Lena image (by the proposed algorithm).

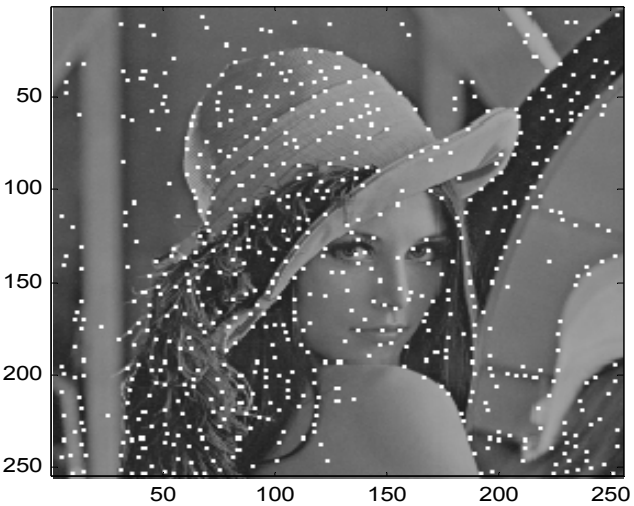

Fig. 6 Corners for Lena image (by Harris' algorithm).