MORPHOLOGICAL EDGE DETECTION AND CORNER DETECTION ALGORITHM USING CHAIN-ENCODING

Neeta Nain, Vijay Laxmi, Ankur Kumar Jain & Rakesh Agarwal
Department of Computer Engineering
Malaviya National Institute of Technology, Jaipur-302017Rajasthan, India
neetanain@yahoo.com, vlgaur@yahoo.co.in, ankur10sep@yahoo.com , raj_mnit222@yahoo.co.in
Telephone No: +91-141-2529078(O), +91-141-2529140(R) **Fax:** +91-141-2529029
Submitted to: IPCV'06

Abstract *- Edges and corners are regions of interest where there is a sudden change in intensity. These features play an important role in object identification methods used in machine vision and image processing systems. This paper presents a novel method for edge and corner detection in images. The approach used here is extracting Edges of the input image using morphological operator and then sending it for Chain Encoding. We are proposing a new morphological edge detector which returns a one pixel thick m-connected binary boundary image. This is followed by our chain encoding method to detect corners on the extracted edges. The algorithm works on all types of images (i.e. binary, gray level and color images). Since the proposed methods are based on morphological operations, these are very simple, efficient and fast. Experimental results on a variety of images identified all the prominent edges and significant corners efficiently.*

Index Terms*—Morphological operations, edge detection, corner detection, thresholding, neighborhoods etc. [1]*

1. Introduction

EDGE detection [1-4] and CORNER detection [7-9, 11] are essential tasks in various computer vision and image-understanding systems. Applications include motion tracking, object recognition, and stereo matching. The requirements of edge detector are that it should identify strong as well as weak edges. All the prominent intensity variations must be taken care of. Similarly corner detection should satisfy a number of important criteria:

1. All "true corners" should be detected.
2. No "false corners" should be detected.
3. Corner points should be well localized.
4. Detector should have a high repeatability rate (good stability).
5. Detector should be robust with respect to noise.
6. Detector should be computationally efficient.

This paper proposes a new *Morphological Edge Detection Method and Corner Detection Algorithm using Chain-Encoding*. The task of finding corners is formulated as a two step process. In the very first step one pixel thick noiseless boundary is extracted by various morphological edge extraction methods. For this, first of all image boundary is extracted by using erosion followed by thresholding, hitmiss transformation and thinning operations. And finally pruning is done to remove extra pixels which may be encountered as side effects. These extracted edges are m-connected. In the second step chain encoding procedures are applied on these extracted edges. Encoding is done using 8-way connectivity. Then abrupt changes are identified in encoding to detect potential corners. Finally optimization of corners to remove false positives by suppressing regular intensity changes is done.
The proposed method has been qualitatively evaluated over a number of images with different intensity gradation with excellent results. It identifies significant corners with minimal false positive rate.

2. Morphological Edge Detector in detail

In order to detect corner we need very fine image boundary. For that purpose we propose our own Morphological edge detector whose output is edge extracted one pixel thick binary image where each image segment is m-connected. The input of this edge detector is a binary or gray level image. If the input is a colored image it is converted into a gray level image. To extract edges of an image the following method is applied.

Step1. Boundary Extraction

Boundary of an image is extracted by the formula

"img = img - erode (img)"

The following set of masks are applied to erode the image.

Mask 1

1	1	1
1	1	1
1	1	1

Mask 2

0	1	0
1	1	1
0	1	0

If the input image is a gray level image then the boundary extracted image is in gray level. The image is required to be threshold. For thresholding the threshold value is calculated as

*"Threshold_Value = Average + k * Standard Deviation"*

Where *k >= 0.* This threshold value helps in identifying pixel locations with high intensity variations, also called edges [1, 10, 11, 12]. The quality of extracted edges can further be improved by adjusting the *Threshold_Value.* The result of this is a boundary extracted image which is two or more pixel thick image.

Step2. To Make Image One Pixel Thick In Horizontal and Vertical Direction

To change two or more pixel thick edge extracted image as in Figure1, into one pixel thick image in Figure2 the following set of 3 x 3 masks are used for horizontal direction. By rotating these masks by 90 degree the masks for vertical direction are obtained too.

Mask 1

1	1	1
0	1	1
0	-1	0

Mask 2

1	1	1
1	1	0
0	-1	0

Using these masks *Hitmiss Transform* [4] of the image is computed and the pixels searched by these masks are removed.

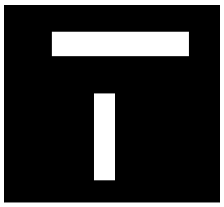
Figure1: Input image

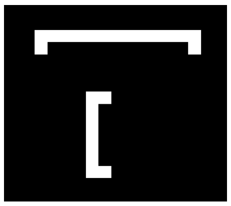
Figure2: Output of step2

Step3. To Remove the Noisy Pixels from the Image and Filling the Pixel Break in Regular Pattern

For making noise less and continuous boundary thinning masks are designed to handle the following cases [5] as shown in Figure 3(A-E).

i. Zero neighborhoods: Isolated pixel.
ii. One neighborhood: One pixel breaks in a straight horizontal or vertical line (one pixel thick).
iii. Two neighborhoods:
   a. Case 1: One pixel breaks in a straight horizontal line and there is only one pixel above the break.
   b. Case 2: One pixel thick diagonal line with one extra pixel in 4 -neighborhood of any of the pixel of this diagonal.
iv. Three neighborhoods:
   a. Case 1: One pixel thick straight line with one extra pixel above any of the pixels in the line.
   b. Case 2: A "T " made by a set of pixels.

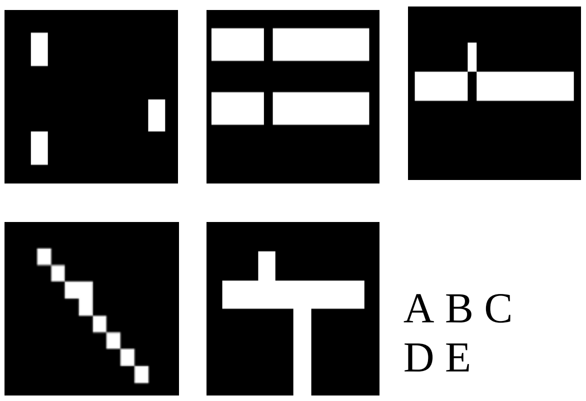

Figure3: A. Zero, B. One, C. Two Case1, D. Two Case2, E. Three Neighborhoods

To *Thin* [4] and to F*ill Gaps* in *the* binary image a set of masks (with rotation of 90 degree four times on each mask) are used.

Mask1

-1	-1	-1
-1	1	-1
-1	-1	-1

For zero neighborhood

Mask2

0	0	0	0	0
0	-1	-1	-1	0
1	1	-1	1	1
0	-1	-1	-1	0
0	0	0	0	0

For one neighborhood

	0	0	0	0	0
	0	-1	-1	-1	0
Mask3	0	-1	1	-1	0
	1	1	-1	1	1
	0	-1	-1	-1	0

Two neighborhood case 1

	0	0	0	0	0
	1	-1	-1	-1	0
Mask4	-1	1	1	-1	0
	0	-1	1	-1	0
	0	0	-1	1	-1

Two neighborhood case 2

	0	0	0	0	0
	0	0	-1	0	0
Mask5	0	-1	1	-1	0
	0	0	1	1	0
	0	0	0	0	0

Three neighborhood case 1

	0	0	0	0	0
	0	0	-1	0	0
Mask6	0	-1	1	-1	0
	0	1	1	0	0
	0	0	0	0	0

Three neighborhood case 2

The output after applying the above masks to Figure 3(A-E) is shown in Figure 4 (A-E):

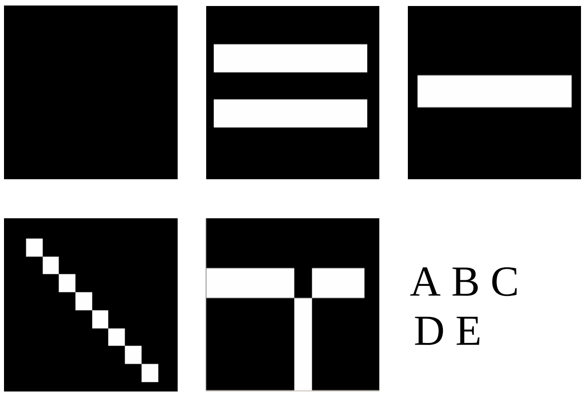

Figure4: Output of performing thinning operation on Figure3 respectively

Now use mask 7 (with four rotations of 90 degree) to obtain m-connected image from the above one pixel thick image.

	0	1	0
Mask 7	1	1	0
	0	0	0

Step4. Pruning

Images obtained from step 2 are having some extra pixels in edges. So in this step these extra pixels are removed using pruning [4]. The frequency of the pruning to remove the different length of extra edges may be varied. The output of this step is one pixel thick binary m-connected image without parasitic components.

Pruning procedure:

```
1. "X1 = thin (A, B)"
2. "X2 = union (hitmiss(X1, B))"
3. "X3 = intersection (dilation(X2, H), A)"
4. "X4 = union (X1, X3)"
```

Where H is a 3 x 3 mask as shown below

	1	1	1
H	1	1	1
	1	1	1

and B is a set of 3x3 masks: Mask 1& Mask 2.

	0	-1	-1
Mask 1	1	1	-1
	0	-1	-1

	1	-1	-1
Mask 2	-1	1	-1
	-1	-1	-1

And finally after pruning isolated pixels are removed using cleaning operation.

3. Experimental Results

Output Figures 5, 6, 7 (B, C) are obtained using our Edge Extraction method. Compare these with Figure 5, 6, 7(D), which Canny Edge Detector [1] gives with a lot of noise.

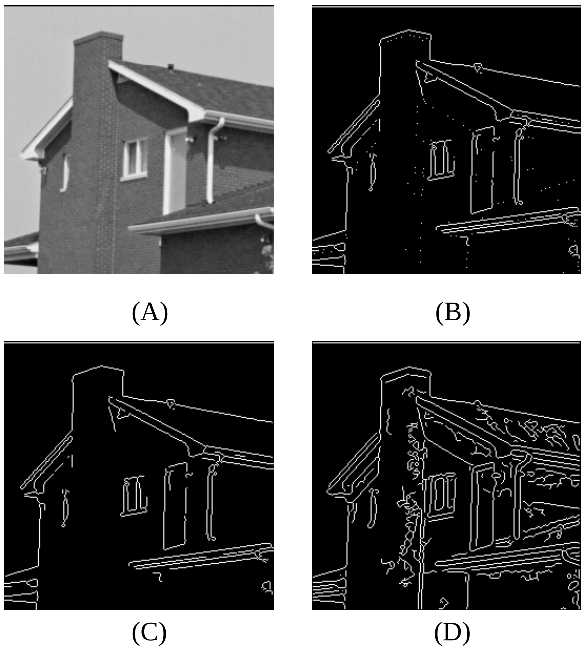

Figure5: A. Original Image (house.tif), B. Image after applying our thinning operation, C. Image after our cleaning isolated pixels and D. Image boundary using Canny Edge Detector [1].

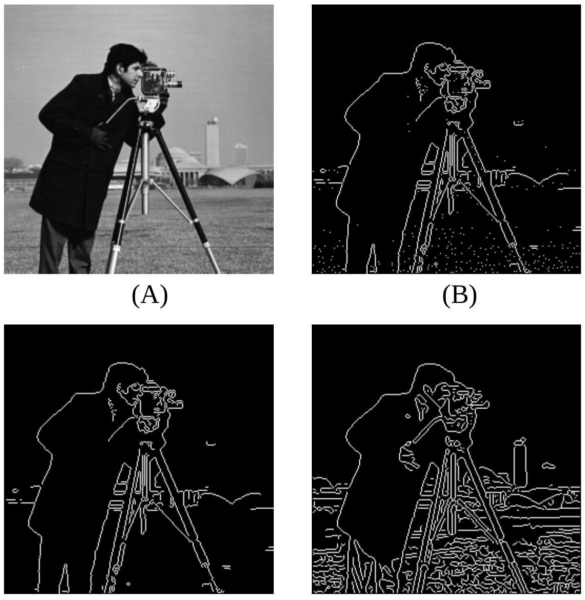

Figure6. A. Original Image (Cameraman.tif), B. Image after applying our thinning operation, C. Image after our cleaning isolated pixels and D. Image boundary using canny edge detector(detects straight edges as uneven edges).

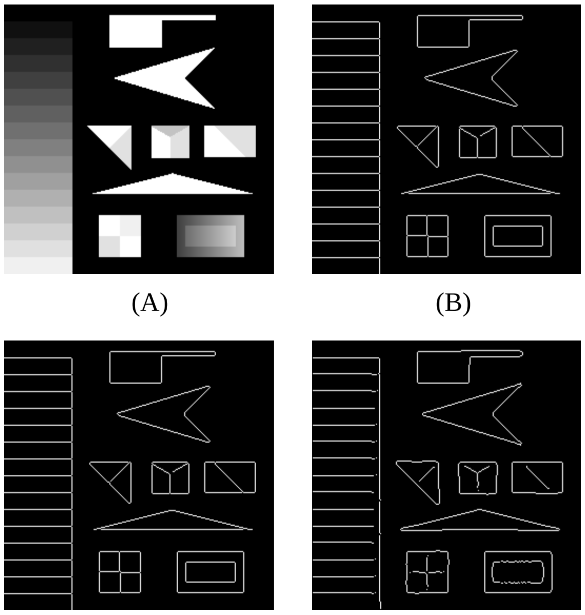

Figure7. A. Original Image (objects.gif), B. Image after applying our thinning operation, C. Image after our cleaning isolated pixels and D. Image boundary using canny edge detector.

To identify less dominant edges and in image reconstruction applications step C of cleaning isolated pixels, can be bypassed to further optimize the edge extraction algorithm. The week edges as shown in Figure 5-7(B) can be reconstructed using Hough transform.

4. Corner Detection Algorithm

In this proposed algorithm chain encoding [4] is performed on a binary extracted image which is obtained by the proposed edge extractor (section 2). On the encoded chain we search abrupt changes. The encoding could be done rotation invariant by using first order differences. In principle if we have two lines that are perpendicular to each other as in Figure 8 then encoded chain will be: 66666000000. So a change from 6 to 0 indicates that there is a corner at this point.

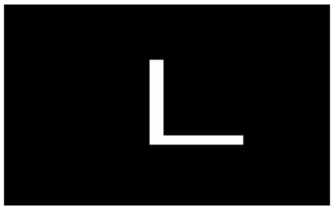

Figure8. Chain encoding of a perpendicular line

But in practical problems the search for an abrupt change is not so easy. Almost every figure has parts that are non linear. Even if only linear figures are considered, then also due to side effects of scan-conversion or aliasing effects from line drawing algorithms, a straight line in the image form or pixel form is not drawn as a straight line.

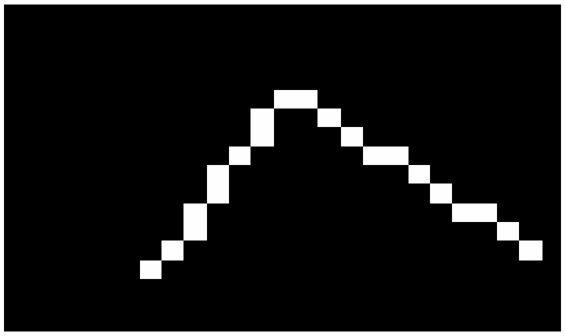

Figure9. Real world scenario of straight line segment

There are some regular changes in the drawn line called stair-step effects. For example consider Figure9, the encoded chain will be some thing like 56556565560777077707. In this chain regular changes are there which will generate false corners. We are interested in finding the abrupt changes and in this case it should identify only one corner placed at position where first 0 is there in the chain.

Also consider the following figures. In Figure10 there should not be any corner as the change here is not an abrupt change but this is a regular change which is a part of a line. Compare it with Figure11 the abrupt change can be seen and that is certainly a corner.

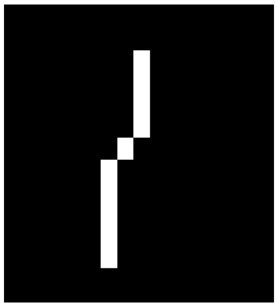
Figure10.

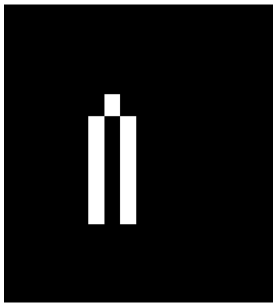
Figure11.

From the above discussion it can be concluded that the regular changes in any image have to be suppressed for detecting abrupt changes and hence significant corners.
To suppress the regular intensity changes we consider three encodes of the chain at a time (i.e. previous, current and next encodes). If the previous and the next encodes are same and the absolute difference between previous and current encodes on chain wheel is one then we substitute the current encode by previous encode. By this rule chain encode of Figure 9 becomes 555555555550777777777. Now abrupt changes can be detected very easily and hence the significant corner.
Isolated line segments of an image can be handled by considering the two end points of the line as corners. To handle this we take another parameter which is checked against the length of the encoded chain. If length is greater than the specified value of the parameter then end points of the line are taken as corners otherwise not.
In order to restrict number of corners it can be ensured that no two corners are closer than a predefined value (in neighborhood of 5*5 pixels).

5. Experimental Results

The test results of our corner detection algorithm on some of the standard images are shown below:

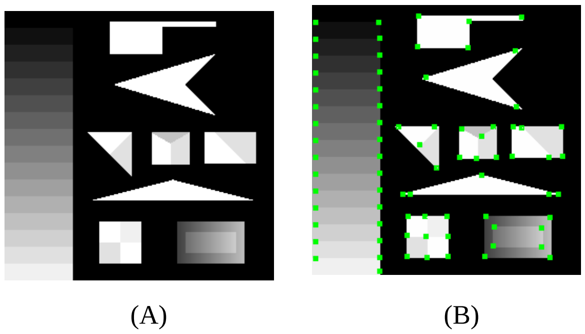
Figure12. A. Original image (objects.gif) and B. After applying corner detection algorithm, corner points are shown by green squares.

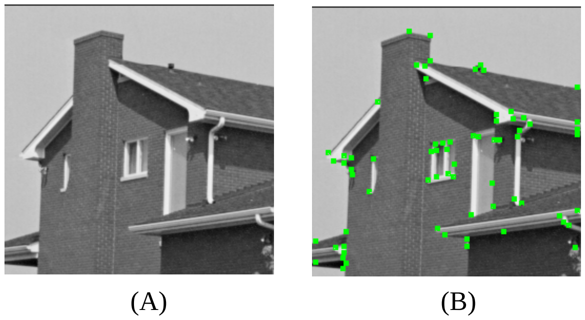
Figure13. A. Original image (houses.gif) and B. After applying corner detection algorithm, corner points are shown by green squares.

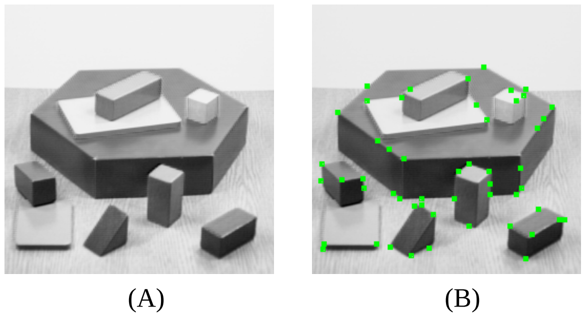
Figure14. A. Original image (hexa.gif) and B. After applying corner detection algorithm, corner points are shown by green squares.

6. Conclusion

In this paper we proposed a novel simple and efficient method for detection of edge or boundary and corner. The method works in two stages: in the first stage the edges of an image can be derived using morphological operators (section 2). In the second stage chain-encoding is performed to derive corners of an image (section 4). Then number of insignificant corner is reduced in order to optimize the corner detection. Compared to the existing Edge detectors like Prewitt, Sobel, Canny etc.,[1, 4] our algorithm extracts precise one pixel thick seamless, continuous (in a segment) image boundary which is very important to extract prominent and significant corners in images[2, 6, 10, 11, 12]. This method works on all types of images. We are further doing the quantitative analysis and generating detailed comparative metrics with the existing approaches as a future extension. This project is implemented using Matlab 7.0.

7. References

[1] J. Canny. "*A ComputationalApproach to Edge Detection*". PAMI, 8(6):679-698, 1986.

[2] Peter Kovesi, "*Phase Congruency Detects Corners and Edges*". The Australian Pattern Recognition Society Conference: DICTA 2003. December 2003. Sydney. pp 309-318.

[3] Rosenfeld and J.S. Weszka. "*An Improved Method of Angle Detection on Digital Curves*." *IEEE Trans. Computers*, 24:940-941, 1975.

[4] Rafael C. Gonzalez and Richard E. Woods, *"Digital Image Processing"*, 2e, PHI, 2004.

[5] P.Kumar, D. Bhatnagar, and P.S. Umapathi Rao. *"Pseudo One Pass Thinning Algorithm"*. Pattern Recognition Letters. 12:543-555, 1991.

[6] F. Mokhtarian and A.K. Mackworth. "*A theory of Multiscale, Curvature-Based Shape Representation for Planar Curves*". IEEE Trans. Pattern Analysis and Machine Intelligence, 14:789-805, 1992.

[7] H. Freeman and L.S. Davis, "*A Corner-Finding Algorithm for Chain-Coded Curves*," IEEE Trans. Computers, vol. 26, pp. 297-303, 1977.

[8] H.-C. Liu and M.D. Srinath. "*Corner Detection from Chain-Code*". Pattern Recognition, 23:51-68, 1990.

[9] Harris, C. and M. Stephens. "*A Combined Corner and Edge Detector*". Fourth Alvey Vision Conference, pp.147-151, 1988.

[10] X. C. He, N. H. C. Yung. *"Curvature Scale Space Corner Detector with Adaptive Threshold and Dynamic Region of Support,"* ICPR, pp. 791-794, 17th International Conference on Pattern Recognition (ICPR'04) - Volume 2, 2004.

[11] SUSAN – "*A New Approach to Low Level Image Processing*", Technical Report TR95SMS1c.

[12] Moravec, H. "*Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover*". Tech Report CMU-RI-TR-3, Carnegie Mellon University, Robotics Institute, Sept. 1980.