IMPROVING EDGE DETECTION USING INTERSECTION CONSISTENCY

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

SERDAR ÇİFTÇİ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

SEPTEMBER 2011

Approval of the thesis:

IMPROVING EDGE DETECTION USING INTERSECTION CONSISTENCY

submitted by **SERDAR ÇİFTÇİ** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen _____
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Adnan Yazıcı _____
Head of Department, **Computer Engineering**

Prof. Dr. Fatoş Tünay Yarman-Vural _____
Supervisor, **Computer Engineering Department, METU**

Assist. Prof. Dr. Sinan Kalkan _____
Co-supervisor, **Computer Engineering Department, METU**

Examining Committee Members:

Assist. Prof. Dr. Ahmet Oğuz Akyüz
Computer Engineering Department, METU _____

Prof. Dr. Fatoş Tünay Yarman-Vural
Computer Engineering Department, METU _____

Assist. Prof. Dr. Sinan Kalkan
Computer Engineering Department, METU _____

Dr. Onur Pekcan
Civil Engineering Department, METU _____

Dr. Ahmet Sayar
Space Technologies Research Institute, TÜBİTAK _____

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: SERDAR ÇİFTÇİ

Signature :

ABSTRACT

IMPROVING EDGE DETECTION USING INTERSECTION CONSISTENCY

Çiftçi, Serdar

M.Sc., Department of Computer Engineering

Supervisor : Prof. Dr. Fatoş Tünay Yarman-Vural

Co-Supervisor : Assist. Prof. Dr. Sinan Kalkan

September 2011, 44 pages

Edge detection is an important step in computer vision since edges are utilized by the successor visual processing stages including many tasks such as motion estimation, stereopsis, shape representation and matching, etc. In this study, we test whether a local consistency measure based on image orientation (which we call Intersection Consistency - IC), which was previously shown to improve detection of junctions, can be used for improving the quality of edge detection of seven different detectors; namely, Canny, Roberts, Prewitt, Sobel, Laplacian of Gaussian (LoG), Intrinsic Dimensionality, Line Segment Detector (LSD). IC works well on images that contain prominent objects which are different in color from their surroundings. IC give good results on natural images that have especially cluttered background. On images involving human made objects, IC leads to good results as well. But, depending on the amount of clutter, the loss of true positives might be more crucial. Through our comprehensive investigation, we show that approximately 21% increase in f-score is obtained whereas some important edges are lost. We conclude from our experiments that IC is suitable for improving the quality of edge detection in some detectors such as Canny, LoG and LSD.

Keywords: Edge Detection, Improving Edge Detection, Intersection Consistency.

ÖZ

KESİŞİMLERİN TUTARLILIĞI KULLANILARAK KENAR BULMAYI İYİLEŞTİRME

Çiftçi, Serdar
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi : Prof. Dr. Fatoş Tünay Yarman-Vural
Ortak Tez Yöneticisi : Yrd. Doç. Dr. Sinan Kalkan
2011 Eylül, 44 sayfa

Kenar bulma; hareket kestirimi, derinlik algılaması, şekil temsili ve eşleştirme gibi birçok görsel işlemler, kenar bulmanın akabinde yapıldığından, bilgisayar görmesinin yararlandığı önemli bir adımdır. Bu çalışmada, daha önce kesişimlerin tespitinin iyileştirilmesinde kullanılan görüntü yönelim tabanlı yerel tutarlılık ölçümünün (Kesişimlerin Tutarlılığı, KT olarak adlandırıyoruz), yedi farklı kenar tespit edicinin sırasıyla, Canny, Roberts, Prewitt, Sobel, Gauss Filtrelenmiş Laplace (GFP), Esas Boyutluluk, Doğru Parça Bulucusu (DPB) kalitesinin arttırılıp arttırılmayacağı test edildi. KT, içerisinde renkleri bakımından çevrelerinden farklı olan belirgin nesne bulunduran resimlerde iyi çalışmaktadır. KT, özellikle dağınık zeminleri olan doğal resimlerde iyi sonuçlar vermektedir. İnsan yapımı nesnelerde de KT iyi sonuçlar vermesine öncülük edebilir. Fakat, dağınıklığın miktarına bağlı olarak gerçek pozitif kayıpları daha önemli olabilir. Kapsamlı araştırmalarımız sonucunda, f-ölçümünde %21 civarlarında bir artış elde ettiğimizi gösterdik, bunun yanında bazı önemli kenarlar da kayboldu. Araştırmalarımız sonucunda, KT'nin Canny, GFP ve DPB gibi bazı kenar bulucularında kenar bulma kalitesini iyileştirebileceği sonucuna vardık.

Anahtar Kelimeler: Kenar Bulma, Kenar Bulmayı İyileştirme, Kesişimlerin Tutarlılığı.

To my family
for being the
sufficient condition of existence...

ACKNOWLEDGMENTS

I would like to thank my supervisor Prof. Dr. Fatoş Tünay Yarman-Vural for her support and patience. Without her help and motivating approach this study would not have been completed. She always listened me, and believed in me. She has contributed to my development not only as an academician but also as a human being.

I am deeply grateful to my co-supervisor Assist. Prof. Dr. Sinan Kalkan. His guidance and help on this study have been invaluable. His friendly approach and kindness have helped me to pursue this study.

I would like to thank my thesis committee members, Assist. Prof. Dr. Ahmet Oğuz Akyüz, Dr. Ahmet Sayar, for their feedback on this study. In addition to his feedback on this study, Dr. Onur Pekcan has also provided precious guidance on thesis writing and LaTeX.

I would like to thank to my friends in our department, Levent, Gülşah, Özlem, Özcan, Mine, Hilal, Çelebi, Can and Murat. In addition to her comradeship, my friend Aslı was most helpful on MATLAB and other technical issues. I also would like to thank to Göker, Serdar, Gökhan and Çağaçan. Especially the support of my friends Faruk, Salih and Eyyüp has helped me to make it through my hard times.

Finally, I would like to thank to my family, whom I dedicate this study. My sisters Aysel, Gülşen, Mülkiye, and Necla, and my brothers Nihat and Adnan; thank you all for being by me all the time. My father, Mehmet, and my love, my mother, Sultan; this study is for you. Thank you for supporting me and believing in me. I am very grateful to you all.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

An edge can be defined as a significant change in local intensities of spatially neighboring pixels. Edge detection is the process of finding the edges in images that determines whether a pixel is an edge or not and assigns a confidence value to its detection.

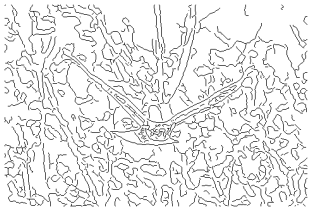
Edge detection is an important step in computer vision as suggested by Oskoei et al. [26], Ziou et al. [42] and by Marr [22] in his Vision Theory (see also section 2.2) since many further steps of computer vision are based on edges that are basic structures that constitute an object. Due to its importance, the process of detection should be robust, for the following steps that rely on the detected edges to be robust.

In this study, we test whether a local consistency measure based on image orientation (which we call Intersection Consistency - IC), which was previously shown to improve detection of junctions, can be used for improving the quality of edge detection of seven different detectors; namely, Canny [6], Roberts [30], Prewitt [29], Sobel [33], Laplacian of Gaussian (LoG) [23], Intrinsic Dimensionality (iD) [21], Line Segment Detector (LSD) [38]. The IC is a regularity measure which checks whether pixels in the image patch point towards the center of the patch or not. Our motivation for testing IC on different edge detectors comes from the observation that IC removes small outliers (e.g., see Figure 1.1). In order to find the best result, we test IC with various window size and power values.

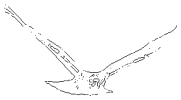
We used the *Berkeley Image Segmentation Dataset (BSD500)* [4] that has ground-truth values. We evaluate the improvement of IC on seven different edge detectors using the precision, recall and f-score measures. We also run the BSD500 [4] benchmark tests.

IC works well on images that contain prominent objects which are different in color from their

Original Image



Canny Operator (σ=1)



IC on Canny

Figure 1.1: Improving edge detection using IC.

surroundings. IC give good results on natural images that have especially cluttered background. On images involving human made objects, IC leads to good results as well. But, depending on the amount of clutter, the loss of true positives might be more crucial. Through our comprehensive investigation, we show that approximately 21% increase in f-score is obtained whereas some important edges are lost. We conclude from our experiments that IC is suitable for improving the quality of edge detection in some detectors such as Canny, LoG and LSD.

In Chapter 2, we provide a literature survey on the existing edge detection methods and studies for improving edge detection. We also introduce the basics of IC. In Chapter 3, we propose the IC for edge detection. In Chapter 4, we provide the results of experiments. In Chapter 5, we conclude and give the future work.

CHAPTER 2

BACKGROUND

Edge detection is an important step in computer vision as suggested by Oskoei et al. [26], Ziou et al. [42] and by Marr [22] in his Vision Theory. Many further steps of computer vision are based on edges that are basic structures that constitute an object. Edge-like structures are detected in human visual system by starting with orientation sensitive cells in primary visual cortex (V1) [14], and biological and machine vision systems depend on their reliable extraction and utilization [20, 22]. As it is shown in Figure-2.1, edges are adequate for identifying objects in perception. This shows that, an edge is an important feature for distinguishing the object from others. Due to its importance, the process of detection should be robust, thus the following steps that rely on the detected edges can be robust. This chapter covers, edge definition, a subset of basic edge detection algorithms, some related works for improving edge detection, and the Intersection Consistency *(IC)* [18].

2.1 Edges

An edge can be defined as a significant change in local intensities of neighboring pixels. The more abrupt the changes of local intensities, the more strong the edges. Edges generally occur at the boundary of between two different regions of pixels. From these, we can infer that an edge is a distinctive point which can be used as a salient feature for several problems in computer vision.

Edges can be in various forms such as *step*, *line*, *ramp*, and *roof* [15] which are shown in Figure-2.2. If local intensity between neighboring pixels changes abruptly, then it is called *step* edge. If local intensity changes abruptly and after a short while the intensity value returns

back to its starting value, then it is called *line* edge. *Step* and *line* edges are in the form of sharp changes. In Figure-2.2, step and line edges are shown to be sharp but in real images, they are not as sharp due to quantization, imaging errors and smoothing performed by image sensing devices. In real images, *step* edges become *ramp* edges and *line* edges become *roof* edges.

Figure 2.1: Edges for perception research, taken from [35].

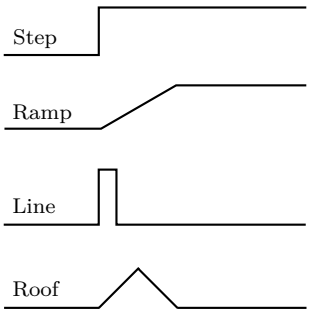


Figure 2.2: Edges in various forms, adapted from [15].

Edges can be used for various purposes in computer vision applications such as image enhancement, recognition, restoration, registration, retrieval, and etc. [26].

2.2 Marr's Paradigm

Marr [22] defined the vision problem as a complex information processing task that is constrained by the physical properties of the machine that should be understood at different, interrelated levels [28]. These levels are: a) computational theory, b) representation and algorithm, and c) hardware implementation. The importance of the computational theory is stressed by Marr as this, to understand the perception only from the neurological way is as fruitless as trying to understand bird flight from the study of feathers [37]. First aerodynamics should be understood, then the structure of feathers make sense. This level describes the logic of the strategy that performs the task. Representation and algorithm level describes how the computation may be, includes information representations and algorithms to manipulate them. Hardware implementation level describes the physical realization of the algorithm that necessitate programs and hardwares [34].

Marr, made his main contribution on the representation level. He formulated a framework for this level. The visible world is based on this framework that has three main representations [28]. These three representations are:

1. **The Primal Sketch:** This representation mainly concerns with the intensity changes. This intensity variations may correspond to physical realities like object boundaries [28]. Primitives for primal-sketch are: blobs, edge-segments, groups, and boundaries [22]. The representation and analysis of local geometric structures take place in this sketch.

2. ***2.5-D* Sketch:** It is a viewer-centered description of rough depth of the visible surfaces, and contours of discontinuities. Primitives for *2.5-D* are: local surface orientation, distance from viewer, discontinuities in depth, and discontinuities in surface orientation [22].

3. ***3-D* Model Representation:** It is an object-centered description of shapes and their spatial organization. This representation uses modular hierarchical representation includes volumetric and surface primitives. The goal in this representation is both handling and recognition of objects [28]. Primitives for *3-D* are: 3-D models that are arranged hierarchically, each one based on a spatial configuration of a few sticks or axes, to which volumetric or surface shape primitives are attached [22].

The framework consists of interrelated representations. The information in primal sketch affects the *2.5-D* sketch which affects the *3-D* model representation. The primal sketch plays an important role for the successor steps. Due to its importance, we aimed to focus our study on the primal sketch representation areas, improving edge detection.

2.3 Edge Detection Algorithms

Edge detection is the process of finding the edges in images, that determines whether a pixel is an edge or not and assigns a confidence value to its detection. Edge detection algorithms usually run in 3 steps namely *smoothing*, *differentiation*, and *localization* [42]. In smoothing, noise in the image are reduced, thus getting rid of unimportant small details. Smoothing removes noise but decreases sharp edges too. After smoothing, differentiation operation is applied in which way the changes in gray values are detected. Lastly, localization operation is applied by that the detected changes in gray values are compared with a threshold value for determining whether the changes at that point corresponds to an edge or not.

Edge detectors have been studied extensively. The edge detectors are usually split into two main categories: *contextual* and *autonomous* detectors [42]. Contextual detectors use a *priori* knowledge which are scene edges and structures. Contextual detectors have limited capabilities and they are adapted to detect precise objects in images. Unlike contextual detectors, autonomous detectors are not limited to detect precise objects. However, the task of determining an edge is based only its neighboring pixels. In this study, the focus is to test improvement on autonomous detectors. By the following sections a subset of edge detector algorithms are described.

2.3.1 Roberts Operator

The Roberts [30] operator makes an approximation to the gradient magnitude which is simple and runs quick [15]. It is computed as:

$$G\left[f\left[i,j\right]\right] = |G_x| + |G_y| \tag{2.1}$$

where G_x and G_y are the image gradient that are the first derivatives of the digital image, according to its x and y axis respectively and calculated by the following masks:

$$G_x = \begin{array}{|c|c|}\hline 1 & 0 \\ \hline 0 & -1 \\ \hline \end{array} \qquad G_y = \begin{array}{|c|c|}\hline 0 & -1 \\ \hline 1 & 0 \\ \hline \end{array} \tag{2.2}$$

which are equal to:

$$G\left[f[i,j]\right] = \left|f[i,j] - f[i+1,j+1]\right| + \left|f[i+1,j] - f[i,j+1]\right|. \tag{2.3}$$

The Roberts operator makes an approximation to the continuous gradient at point $[i+1/2, j+1/2]$. The results of the Roberts edge detector are shown in Figure-2.3

2.3.2 Sobel Operator

To have a gradient value for the exact position instead of the interpolated point, the Sobel [33] operator can be used which computes the gradient by using a 3x3 window. Arrangement of the operator mask around the pixel $[i, j]$ is shown in Figure-2.4. The Sobel operator computes the magnitude of the gradient by:

$$M = \sqrt{s_x^2 + s_y^2}, \tag{2.4}$$

where the partial derivatives s_x and s_y are usually as follows

$$s_x = (a_2 + ca_3 + a_4) - (a_0 + ca_7 + a_6) \tag{2.5}$$

$$s_y = (a_0 + ca_1 + a_2) - (a_6 + ca_5 + a_4) \tag{2.6}$$

and the constant c is usually 2. The gradient operators s_x and s_y are usually implemented by the following masks:

$$s_x = \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array}, \qquad s_y = \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array}. \tag{2.7}$$

Sobel operator puts an emphasis on the pixels that are closer to the center of the window [15]. The results of the Sobel edge detector are shown in Figure-2.3.

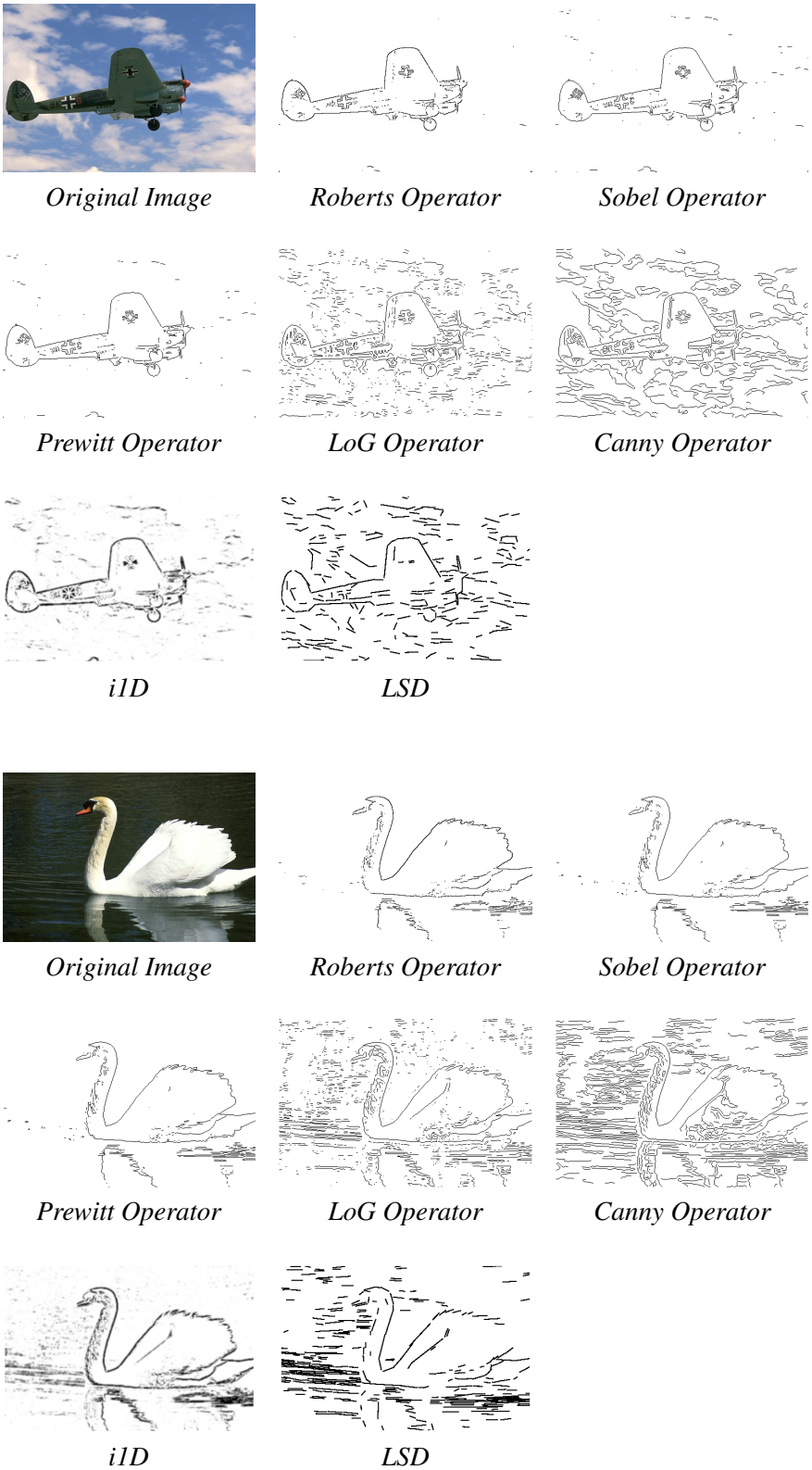


Figure 2.3: Some Basic Edge Detector Results.

a_0	a_1	a_2
a_7	$[i, j]$	a_3
a_6	a_5	a_4

Figure 2.4: Labels of neighboring pixels used in Sobel and Prewitt Opertors, adapted from [15].

2.3.3 Prewitt Operator

The Prewitt [29] operator runs the same way as the Sobel operator except for the constant $c = 1$. This operator does not put emphasis on the pixels that are closer to the center of the window [15]. The Prewitt operator can be implemented by the following masks:

$$s_x = \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline \end{array}, \qquad s_y = \begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -1 & -1 \\ \hline \end{array}. \tag{2.8}$$

The results of the Prewitt edge detector are shown in Figure-2.3.

2.3.4 Laplacian of Gaussian (LoG) Operator

The algorithms described earlier compute the first derivatives of the image to obtain the gradient, and if the magnitude of the gradient is above a threshold it is assumed to be an edge. This results with too many edge points. To avoid this situation, pixels that have local maxima in their gradient are assumed to be edges which means their first derivative will make a peak and equivalently there will be a zero-crossing in their second derivative [15].

To find the zero-crossings, the second derivatives of the image are computed. Laplacian operator, $\nabla^2 f = (\partial^2 f/\partial x^2) + (\partial^2 f/\partial y^2)$, is used for detecting the second derivatives. This operator does not used much in vision applications because even a small peak in the first derivative will result in zero-crossings in the second derivatives which makes Laplacian operators sensitive to noise. To avoid the noise effect, Laplacian operator is combined with the Gaussian smoothing [23]. This Laplacian of Gaussian(LoG) has these properties:

1. Uses Gaussian filtering for smoothing.

2. Computes the second derivatives of the image.

3. Zero-crossing in the computed second derivative, which is local maximum in the first derivative, is assumed to be edge.

4. Edge location can be an approximated by using linear interpolation.

It is shown in Figure 8.3 in [12] that as the σ parameter for the Gaussian filter increases the detail is suppressed. From that figure we can see that $\sigma = 2$ preserves details. The computer vision toolboxes [1, 2] take $\sigma = 2$ as default value. In our experiments we took $\sigma = 2$ too.

The second derivatives of a function along x and y direction can be approximated by the following differences:

$$\begin{aligned}
\frac{\partial^2 f}{\partial x^2} &= \frac{\partial G_x}{\partial x} && (2.9)\\
&= \frac{\partial f\,[i, j+1] - f\,[i, j]}{\partial x} && (2.10)\\
&= \frac{\partial f\,[i, j+1]}{\partial x} - \frac{\partial f\,[i, j]}{\partial x} && (2.11)\\
&= \Big(f\,[i, j+2] - f\,[i, j+1]\Big) - \Big(f\,[i, j+1] - f\,[i, j]\Big) && (2.12)\\
&= f\,[i, j+2] - 2f\,[i, j+1] + f\,[i, j] && (2.13)
\end{aligned}$$

To get the partial second order derivatives, central difference method can be used. Which takes difference between the centered one, with the two neighbors $(f(x_1) - 2f(x_0) + f(x_{-1}))$. From the central difference method, this approximation can be seen that it is centered at pixel $[i, j+1]$. By replacing j with $j-1$, we get

$$\frac{\partial^2 f}{\partial x^2} = f\,[i, j+1] - 2f\,[i, j] + f\,[i, j-1] \tag{2.14}$$

which is centered at pixel $[i, j]$. Similarly,

$$\frac{\partial^2 f}{\partial y^2} = f\,[i+1, j] - 2f\,[i, j] + f\,[i-1, j]\,. \tag{2.15}$$

The Laplacian operator can be approximated by combining equations 2.14 and 2.15 with the mask given below:

$$\nabla^2 \approx \begin{array}{|c|c|c|}\hline 0 & 1 & 0 \\ \hline 1 & -4 & 1 \\ \hline 0 & 1 & 0 \\ \hline \end{array}\,. \tag{2.16}$$

To implement the LoG, the Laplacian operator need to be combine with Gaussian, $g(x, y)$. The result of LoG operator, $h(x, y)$, is obtained by convolution operation,

$$h(x, y) = \nabla^2\Big[(g(x, y) \star f(x, y)\Big]. \tag{2.17}$$

Derivative rule for convolution,

$$h(x,y) = \left[\nabla^2(g(x,y)\right] \star f(x,y), \tag{2.18}$$

where

$$\nabla^2\Big(g(x,y)\Big) = \left(\frac{x^2+y^2-2\sigma^2}{\sigma^4}\right) e^{-(x^2+y^2)/2\sigma^2}. \tag{2.19}$$

An LoG mask is shown in Figure-2.5. For the purpose of the work, the size of the mask can be different. The results of the LoG edge detector are shown in Figure-2.3.

0	0	-1	0	0
0	-1	-2	-1	0
-1	-2	16	-2	-1
0	-1	-2	-1	0
0	0	-1	0	0

Figure 2.5: 5x5 Laplacian of Gaussian mask.

2.3.5 Canny Operator

Canny [6] operator uses the first derivative of a Gaussian image and closely approximates the optimum signal-to-noise ratio and localization [15]. It assigns a pixel as an edge, if magnitude of the gradient of the pixel is larger than the pixels that are at both sides of the pixel, in the direction of the maximum intensity change [8]. According to Canny, the aim of good detector is:

1. **Good detection:** Getting true edges should be maximized and false edges should be minimized,
2. **Good localization:** The detected edges should be localized as in the real edges,
3. **Single response:** A real edge in the image should be marked once.

Canny operator runs in the following steps [15]:

1. Applying Gaussian filter for smoothing the image.
2. Partial derivatives are used for computing the gradient magnitude and orientation.

3. Non-maxima suppression is applied to the gradient magnitude.

4. For obtaining the potential edges, double thresholding and hysteresis tracking is applied.

The σ parameter for the Gaussian filter can be in the range of 0.5-5.0 [32]. Nadernajad et al. [25] took σ as 1. In our experiments we also compute Canny with parameter $\sigma = 1$.

Non-maxima suppression, double thresholding, and hysteresis tracking steps can be mentioned briefly.

Non-maxima Suppression (NMS) , is a thinning operation. The maximum gradient magnitude is chosen, in the gradient direction. NMS runs as follows:

- Round the gradient angle to the closest 45 degrees.
- Traverse a 3x3 window across the magnitude array.
- If the gradient magnitude of the centered element in the window is larger than the neighbored elements along the gradient direction, then it is assumed to be an edge element, otherwise the centered element is set to zero.

Double Thresholding

Selecting a proper threshold value is difficult. Choosing too low or too high threshold values may cause redundant (false positive) or missing (false negative) edges. To avoid this causes double thresholding is a solution. Double thresholding uses two threshold values t_1 and t_2. The edge pixels that are higher than t_2 called *strong* edges, lower than the t_1 are suppressed and edge pixels between two thresholds are leaved as *weak* edges.

Hysteresis Edge Tracking

The edges that are found as *strong* edges in the double thresholding are accepted as edges. The edges that are found as *weak* edges are need to be checked once again. If any of these *weak* edges are connected to a *strong* edge then these weak edges are assumed to be an edge, otherwise not.

The results for the Canny edge detector are shown in Figure-2.3.

2.3.6 Intrinsic Dimensionality (*iD*)

iD is a continuous representation of intrinsic dimension of an image patch in terms of its local spectrum or, same as, its gradient field [10]. It assigns three confidence measures, homogeneous, edge-like and junction-like structures to each patch and they are namely classified by *iD* as *intrinsically zero dimensional (i0D), intrinsically one dimensional (i1D) and intrinsically two dimensional (i2D)*. By observing the spectral representation of a local patch in Figure 2.6, we see that the energy of an i0D signal is accumulated in the origin, the energy of an i1D signal is accumulated along a line, and energy of an i2D signal varies in more than one dimension. The structure of the iD can be shown by a triangle that is spanned by two measures: origin variance and line variance. Origin variance shows the deviation of the energy from a accumulation at the origin while line variance shows the deviation from a line structure. The corner points of the triangle represent the 'ideal' situations of *iD*. The surface of the triangle equivalent to signals that carry aspects of the three 'ideal' situations, and the distance from the corners of the triangle shows the similarity or dissimilarity to *ideal* i0D, i1D and i2D signals [21].

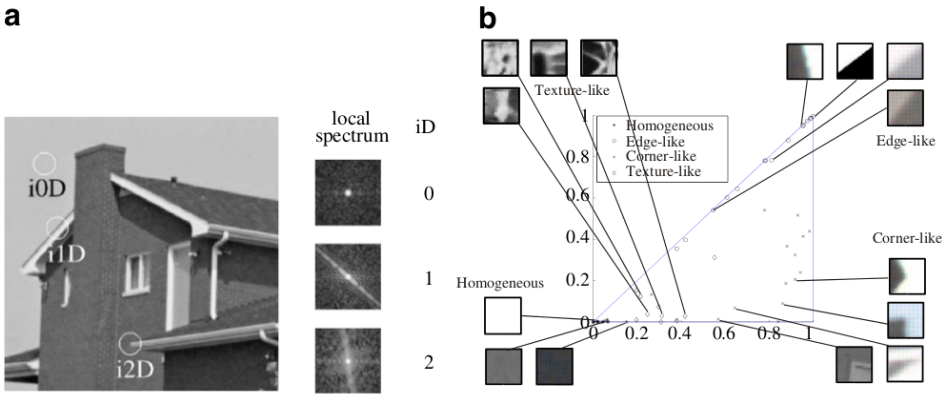


Figure 2.6: Intrinsic Dimensionality, taken from [10]. (a) Three different intrinsic dimensionalities are indicated. The other three images show local spectra of these images. (b) Different image structures and their position in the iD triangle.

2.3.7 Line Segment Detector (*LSD*)

The algorithm is improved by Grompone et al. [38]. It is a linear time detector, require no parameter tuning and controls the false detections. The algorithm is based on Burns et al. [5]

line segment detector and results are improved by inspiring from Desolneux et al. [7] validation criterion that use of gradient orientation and a new framework to deal with parameter setting. The Burns et al. algorithm extracts line segments in following steps:

1. The image is partitioned into line-support regions by grouping connected pixels that share the same gradient angle up to a defined tolerance.

2. Determine line segment that best approximates each line-support region.

3. Validate or not each line segment according to the information in the line-support region.

The Grompone et al. algorithm uses the main ideas of steps 1 and 2, with some improvements. In step 3 they used Desolneux et al. *contrario* method in different form. In step 1 each region starts with just one pixel and the region angle set to the level-line angle at that pixel. They test the pixels in 8-connected pixel neighborhood adjacent to the region. They add the ones with level-line orientation equal to the region angle up to a certain tolerance to the region. At each iteration, the region angle is updated. In step 2 the line-support regions must be associated with a line segment. A line segment is determined by its end points and its width or, equivalently, its center, angle, length, and width. In LSD (the idea was first proposed by Kahn et al. in [17], [16]), they use center of mass to select the rectangle orientation. They use gradient magnitude as the pixel's mass. They chose length and the width in such a way as to cover the line-support region. In step-3 the Desolneux et al. algorithm gives many parallel detections and it requires a lot of effort to go back to a correct interpretation. To deal with this problem, they used rectangles (line segment with a certain width) instead of line segments. The complete LSD algorithm is shown in Figure 2.7,
where ρ: is a threshold, points with gradient magnitude smaller than ρ are discarded; τ is the angular tolerance, ε is the number of meaningful rectangles, *nfa* is the number of false alarms. *Grad* function computes the image gradient and returns three outputs: the level-line angles, the gradient magnitude, and an ordered list of pixels. An image *Status* where pixels used by other regions are marked. *RegionGrow* is used to obtain a line-support region. *RectApprox* gives a rectangle approximation of the region. *ImproveRect* tries several perturbations to the initial approximation in order to get better approximation.

Algorithm : LSD: LINE SEGMENT DETECTOR
input: An image I, parameters ρ, τ and ε.
output: A list *out* of rectangles.

```
(LLAngles, GradMod, OrderedListPixels) ← Grad(I, ρ);
Status(allpixels) ← NotUsed;
foreach pixel P in OrderedListPixels do
    if Status(P) = NotUsed then
        region ← RegionGrow(P, τ, Status);
        rect ← RectApprox(region);
        nfa ← NFA(rect);
        nfa ← ImproveRect(rect);
        if nfa < ε then
            Add rect to out;
            Status(region) ← Used;
        else
            Status(region) ← NotIni;
        end
    end
end
```

Figure 2.7: Line Segment Detector algorithm, taken from [38].

In their experiments they obtained good results and algorithm runs faster than compared methods. It works better on straight objects, does not give good results on curved ones.

In Table 2.3.7 the advantages and disadvantages of the mentioned methods are shown.

Table 2.1: Comparison of Edge Detectors.

Detector	**Advantages**	**Disadvantages**
Canny	Good localization and response.	Complex computations, need to determine the parameter values.
Sobel, Prewitt	Simple computations.	Sensitive to noise.
Roberts	Simple computation, good to detect diagonal edges.	Sensitive to noise.
LoG	Good localization.	Sensitive to noise, need to determine the parameter values.
i1D	A continuous representation for the intrinsic dimensionality of local image structures.	Need post processing for final decision.
LSD	Good detection of lines, faster.	Not good on curves.

2.4 Studies on Improving Edge Detection

He and Yung [13] suggest that, a meaningful edge has long length *(continuity)*, smooth curvature *(smoothness)*, and significant magnitude value. Based on this, they propose that an edge detector should utilize all those values. Edge-likelihood Index *(ELI)*, utilizing the gradient, length and curvature of edges. The method is described as follows. A raw edge map is obtained from any traditional gradient-based edge detector. An 8-connected component labeling operation is applied to split this edge map into contours $(C_1, C_2, ..., C_m)$. $Gradient_i$ is mean gradient of each $contour_i$. $Length_i$ is number of pixels in each $contour_i$. $Curvature_i$ is mean curvature of each contours. Curvature of contours is computed by Curvature Scale Space *(CSS)* method [24]. The properties of contours are combined in *ELI* as follows:

$$ELI_i = \frac{(gradient_i)^p \times (length_i)^q}{(curvature_i)^r}, \tag{2.20}$$

where i is the index for each contour and p, q, and r are variables for optimizing the edge detection. The normalized ELI *(NELI)* is obtained by the following step:

$$NELI_i = \frac{ELI_i}{max(ELI)}. \tag{2.21}$$

The final edge-map *(FEM)* for a pixel $(x, y) \in C_i$ is obtained by:

$$FEM(x, y) = \begin{cases} 1 & \text{if } NELI_i \geq T, \\ 0 & \text{if } NELI_i < T, \end{cases} \tag{2.22}$$

where T is the threshold value and determined empirically or automatically. They compared their results with *Canny* [6] and got relatively better results. For instance, a pixel is detected as an edge even it has low gradient magnitude, by utilizing its continuity and smoothness. Even some edge pixels have reasonable gradient magnitude, they could not detected due to their short length and high curvature. Detecting edges are proportional to the gradient magnitude and length of contours, and inversely proportional to the mean curvature of contours.

Wang and Xue [40] suggest a new method is based on *Maximizing Objective Function (MOF)* [19]. The MOF method is described as follows. An edge map and direction map is obtained by using four different 3x3 direction masks. Each direction mask, divide the pixels in the mask, into two different sets S_0 and S_1. If the interset distance between S_0 and S_1 is large and intraset distances of S_0 and S_1 are small then the compactness of sets are high and edge intensity is large. Non-maxima suppression is applied by using the edge and direction map.

Wang and Xue add four more direction masks. By using these eight direction masks they reduced noise.

Yang et al. [41] suggest an improving edge detection method based on the Rothwell [31] method. Rothwell method uses topological relations for edge detection, it is same as Canny method but there are some differences: Rothwell does thinning as post edge detection process and instead of hysteresis, dynamic thresholding is used. Yang et al. [41] used 5x5 distance transform in the thresholded gradient map. They approximate the boundary object by using B-Spline curve fitting for sub-pixel interpolation. In their experiment they eliminate some spurious edges but the execution time got higher.

Fan et al. [9] suggest a new method fuses each color space component outputs. They used *YUV* color space because chrominance components *(U and V)* separated from luminance *(Y)* efficiently, and it is used mostly in image and video processing applications thus avoided from computation of the format transformation. The method is described as follows. Firstly, four different pattern of second-order neighborhood masks, that are detailed in the paper, namely horizontal *HOE(x,y)*, vertical *VOE(x,y)*, north-east diagonal *NOE(x,y)*, and north-west diagonal *SOE(x,y)* are applied to each *YUV* component. After applying these masks, the *local maximum edge strength (MOE)* of the pixel (x, y) is computed:

$$MOE(x, y) = max\{HOE(x, y), VOE(x, y), NOE(x, y), SOE(x, y)\}. \tag{2.23}$$

For classifying the pixels on the luminance component whether the pixel is an edge or not, an optimal threshold $\bar{T}_y$ is applied to *local maximum edge strength*:

$$Y_E(x, y) = \begin{cases} 1, \text{edge pixel} & \text{if } MOE(x, y) \geq \bar{T}_y, \\ 0, \text{non-edge pixel} & \text{if } MOE(x, y) < \bar{T}_y, \end{cases} \tag{2.24}$$

where $Y_E(x, y)$ stands for edges on Y (luminance) component and threshold $\bar{T}_y$ is determined by *fast entropic thresholding technique*. This thresholding technique is developed by the authors and it is efficient for the binary classification. For reducing the search burden of the optimal threshold value, a recursive iteration is done on re-normalized part repeatedly. The steps given above are repeated for U and V components too. The final edge map is obtained

as follows:

$$E(x,y) = \begin{cases} 1, \text{edge pixel} & \text{if } Y_E(x,y) = 1 \text{ or,} \\ & U_E(x,y) = 1 \text{ or,} \\ & V_E(x,y) = 1, \\ 0, \text{non-edge pixel} & otherwise. \end{cases} \tag{2.25}$$

They detect the potential edges and by using fast entropic thresholding the computation cost is reduced. There is no quantitative information for their results.

Wang and Fan [39] suggest a new method to improve *Canny* [6] method. The method is described as follows. For removing noise, they used adaptive filtering instead of Gaussian, that can select the weight adaptively according to the gray values jumps. Determining the proper threshold value is difficult. Instead of using double thresholding they used gradient direction. They take an 8-neighborhood around a pixel. They decide whether the centered pixel is in the direction of the neighborhood ones. Finally, to avoid multi responses, they used mathematical morphology to thin the detected edges. They compared their results with the traditional Canny method and obtained better results but the computational time also increased.

2.5 Intersection Consistency (*IC*)

The IC is a regularity measure which checks whether pixels in the image patch (a sliding window) point towards the center of the patch or not [18]. This decision is made by the distance between the center $\mathbf{p}_c$ and the line passing through the pixel in the patch. The line is defined by the position of the pixel $\mathbf{p}$ and the computed orientation information θ_p. The IC at $\mathbf{p}_c$ is defined by the weighted average of these distances:

$$IC(\mathbf{p}_c) = \int [c_{i1D}(\mathbf{p})]^2 \left[1 - \frac{d(l^{\mathbf{p}}, \mathbf{p}_c)}{d(\mathbf{p}, \mathbf{p}_c)}\right] d\mathbf{p}, \tag{2.26}$$

where

$\mathbf{p}$: is the index of the pixels in image patch P,

$c_{i1D}(\mathbf{p})$: is the confidence for $i1D$ of pixel $\mathbf{p}$,

$l^{\mathbf{p}}$: is the line passing through pixel $\mathbf{p}$ defined according to the orientation θ_p,

$d(l^{\mathbf{p}}, \mathbf{p}_c)$: is the distance between $l^{\mathbf{p}}$ and $\mathbf{p}_c$,

$d(\mathbf{p}, \mathbf{p}_c)$: is the distance between $\mathbf{p}$ and $\mathbf{p}_c$. This variables are shown in Figure 2.8.

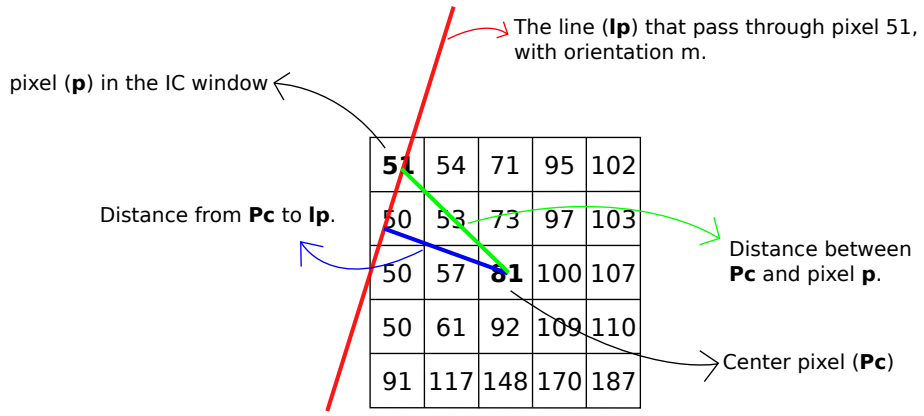


Figure 2.8: IC window (5x5) and its variables.

In order to give equal weights to every pixel according to their closeness to the center, $d(l^{\mathbf{p}}, \mathbf{p}_c)$ is normalized by $d(\mathbf{p}, \mathbf{p}_c)$. IC that Kalkan et al. [18] used is similar to work of Parida et al. [27] observation of junction position and Förstner [11] regularity function. For edgeness measure, both Parida et al. and Förstner are utilized local image gradient whereas Kalkan et al. used intrinsic-one-dimensionality (c_{i1D}).

Kalkan et al. used IC for improving the detection of junction positions, looking for the intersection of edges. In this study, we test IC for improving the detection of edges.

2.6 Conclusion

Due to its importance, edge detection and improvement of detection are studied intensively. A subset of edge detection and improvement of edge detection methods are mentioned in this chapter. There is no single method that detects edges completely on every real image. In this study, we test whether a local consistency measure based on image orientation IC, which was previously shown to improve detection of junctions, can be used for improving the quality of edge detection by seven different detectors namely, Canny, Roberts, Prewitt, Sobel, Laplacian of Gaussian, Intrinsic Dimensionality, Line Segment Detector.

CHAPTER 3

IMPROVING EDGE DETECTION USING INTERSECTION CONSISTENCY

The importance of the edge detection was mentioned earlier. In this study, we test Intersection Consistency *IC* [18] to show whether it can be used for improving the quality of edge detection. We test IC with various window size and power values.

3.1 Proposed Method

This study is motivated from Kalkan et al. [18]. They used IC for improving the localization of junctions. They obtained good results for detecting junctions. We test IC with various window sizes and power values. The formula we test is:

$$IC(\mathbf{p}_c) = \sum_{\mathbf{p} \in P} [ec(\mathbf{p})^{power}] \left[1 - \frac{d(l^{\mathbf{p}}, \mathbf{p}_c)}{d(\mathbf{p}, \mathbf{p}_c)} \right], \tag{3.1}$$

where $ec(\mathbf{p})$ is edge-like confidence *(edge-map, edge-strength)* for pixel $\boldsymbol{p}$, and the other variables are introduced in following sections.

3.1.1 Patch Pixels *(P)*

It is a set of pixels in the image. This image patch is obtained by sliding a window on the digital image. The size of the window is determined after experiments. 3x3 IC window gives best result. The effect of the window size is shown in Chapter 4.

3.1.2 Obtaining the Line Equation Passing Through Pixel p

The line equation is obtained by using the pixel coordinates (x, y) and the pixel orientation θ_p. Orientation is obtained by $tan^{-1}\left(G_y/G_x\right)$ where G_x and G_y are the image gradient that are the first derivatives of the digital image, according to its x and y axis respectively. By using these parameters we can obtain the line l_p equation which passes through **p**(x,y):

$$y = mx + n, \tag{3.2}$$

where $m = \theta_p$ and n is a constant.

3.1.3 Obtaining the Point-Line Distance

A simple line equation is given by:

$$Ax + By + C = 0, \tag{3.3}$$

and if we adapt (3.2) to (3.3) we get:

$$-mx + y - n = 0, \tag{3.4}$$

thus, the coefficients are equal to $A = -m$, $B = 1$, and $C = -n$, where $n = y - mx$. The distance between a line and a point is computed by:

$$d = \frac{|Ax + By + C|}{\sqrt{A^2 + B^2}}\ . \tag{3.5}$$

After getting the parameters A, B, and C, we can obtain the point-line distance formula. According to the value of slope*(m)*, there are four cases for this computation.

Case 1: $G_x = 0, G_y = 0$

Division zero by zero is not a number, for this case we assumed to take the parameter values as:

$$A = 0, B = 0, C = 0. \tag{3.6}$$

Case 2: $G_x = 0, G_y \neq 0$

Division any number to zero which equals to infinite number, this case corresponds to a vertical line slope. The line equation is $x = n$. Adapt the $x = n$ to general line equation form

$x - n = 0$, thus

$$A = 1, B = 0, C = -n. \tag{3.7}$$

Case 3: $G_x \neq 0, G_y = 0$

Division zero to any number which equals to zero, this case corresponds to a horizontal line slope. The line equation is $y = n$. Adapt the $y = n$ to general line equation form $y - n = 0$, thus

$$A = 0, B = 1, C = -n. \tag{3.8}$$

Case 4: $G_x \neq 0, G_y \neq 0$

The slope is equal to arctan of division of G_y to G_x. For this case

$$m = \tan^{-1}\left(\frac{G_y}{G_x}\right), \tag{3.9}$$

$$n = y - mx, \tag{3.10}$$

$$A = -m, B = 1, C = -n, \tag{3.11}$$

steps are followed.

3.1.4 Obtaining the Distance Between Pixels p and p_c

Distance between any pixel $\mathbf{p}$ in the patch P and pixel $\mathbf{p}_c$ at the center of the patch P is computed by the Euclidean distance:

$$d(\mathbf{p}, \mathbf{p}_c) = \sqrt{(\mathbf{p}_x - \mathbf{p}_{cx})^2 + (\mathbf{p}_y - \mathbf{p}_{cy})^2}\,. \tag{3.12}$$

After obtaining all the parameters that IC needs, the computation will be as given in Algorithm-1. IC is normalized with the window size, thus a coherent information is obtained. Experiment results are shown in Chapter 4.

3.2 Testing IC

In order to find the best result, we test the original IC formula with various window size and power values. To observe the affect of the power values,

$$IC(\mathbf{p}_c) = \sum_{\mathbf{p} \in P} [ec(\mathbf{p})^{power}] \left[1 - \frac{d(l^{\mathbf{p}}, \mathbf{p}_c)}{d(\mathbf{p}, \mathbf{p}_c)}\right], \tag{3.13}$$

Algorithm 1 Test Intersection Consistency for Improving the Quality of Edge Detection

input : Gray-level Image *(im)*, Edge-like Confidence *(ec)*, Window-Size *(w)*

output: Intersection Consistency Map *(IC)*

$[numRows, numCols] \leftarrow size(im)$; $[F_x, F_y] \leftarrow Gradient(im)$

for $i \leftarrow w{+}1$ **to** $numRows{-}w$ **do**

  for $j \leftarrow w{+}1$ **to** $numCols{-}w$ **do**

    if $ec(i,j){=}\,0$ **then**
      skip this round;
    end

    for $k \leftarrow i - w$ **to** $i + w$ **do**

      for $l \leftarrow j - w$ **to** $j + w$ **do**

        if $F_x(k,l) = 0$ **and** $F_y(k,l) = 0$ **then**
          $A = 0; B = 0; C = 0;$
        end

        if $F_x(k,l) = 0$ **and** $F_y(k,l) \neq 0$ **then**
          $n = k; A = 1; B = 0; C = -n;$
        end

        if $F_x(k,l) \neq 0$ **and** $F_y(k,l){=}\,0$ **then**
          $n = l; A = 0; B = 1; C = -n;$
        end

        if $F_x(k,l) \neq 0$ **and** $F_y(k,l) \neq 0$ **then**
          $m = \tan^{-1}\left(\frac{G_y}{G_x}\right); n = l - m \times k; A = -m; B = 1; C = -n;$
        end

        $EuclideanDistance = \sqrt{((i-k)^2 + (j-l)^2)};$

        $pointLineDistance = \frac{|A \times k + B \times l + C|}{\sqrt{A^2 + B^2}};$

        $IC(i,j) = \sum [ec(k,l)]\left[1 - \frac{pointLineDistance}{EuclideanDistance}\right].$

      end

    end

    $IC(i,j) = \frac{IC(i,j)}{(2 \times w + 1)^2}$

  end

end

we test IC with six different power values that are 0.25, 0.5, 1, 1.5, 2, and 3. We experiment IC for 3x3, 5x5, 7x7, and 9x9 IC window sizes. Apart from window size and power value, we also test IC for weighting the IC with image gradient,

$$IC(\mathbf{p}_c) = \sum_{\mathbf{p} \in P} [(mag)^{power}] \left[1 - \frac{d(I^{\mathbf{p}}\mathbf{p}_c)}{d(\mathbf{p}, \mathbf{p}_c)} \right]. \tag{3.14}$$

Results are shown in Chapter 4.

CHAPTER 4

EXPERIMENTS & RESULTS

In this chapter, the IC results are compared with the traditional edge detectors. We run our experiments on NAR multi-core high performance computer, which is located in our department. We used MATLAB environment for computing IC and doing comparisons. For the computation, we used gray-level values of images since their computation are easier than the colored ones. This chapter covers an introduction for the dataset that is used for the experiments, evaluation metrics for the benchmarking, experiment results, and discussion on results.

4.1 Dataset

For the experiments, the *Berkeley Image Segmentation Dataset (BSD500)* [4] was used. The dataset contains 500 various natural images which are manually segmented. This dataset is prepared for image segmentation and boundary detection purposes. For each image, there are segmented regions and boundary units of ground-truth values. This dataset was segmented by various human subjects and the annotations that are done by the subjects serve as a ground-truth for image segmentation and boundary detection. For each image there are various number of annotations, it is not fixed. The dataset separated into disjoint *training*, *validation*, and *test* subsets. For comparisons, the experiments are done on *test* folder as authors of the dataset specified that comparison should be with the *test* folder. The test folder contains 200 natural images. The reason of using this dataset is that, it has ground-truth, it is popular for image segmentation and boundary detection, and it has its own benchmarking codes. In Figure 4.1 some images and their ground-truth values are shown.

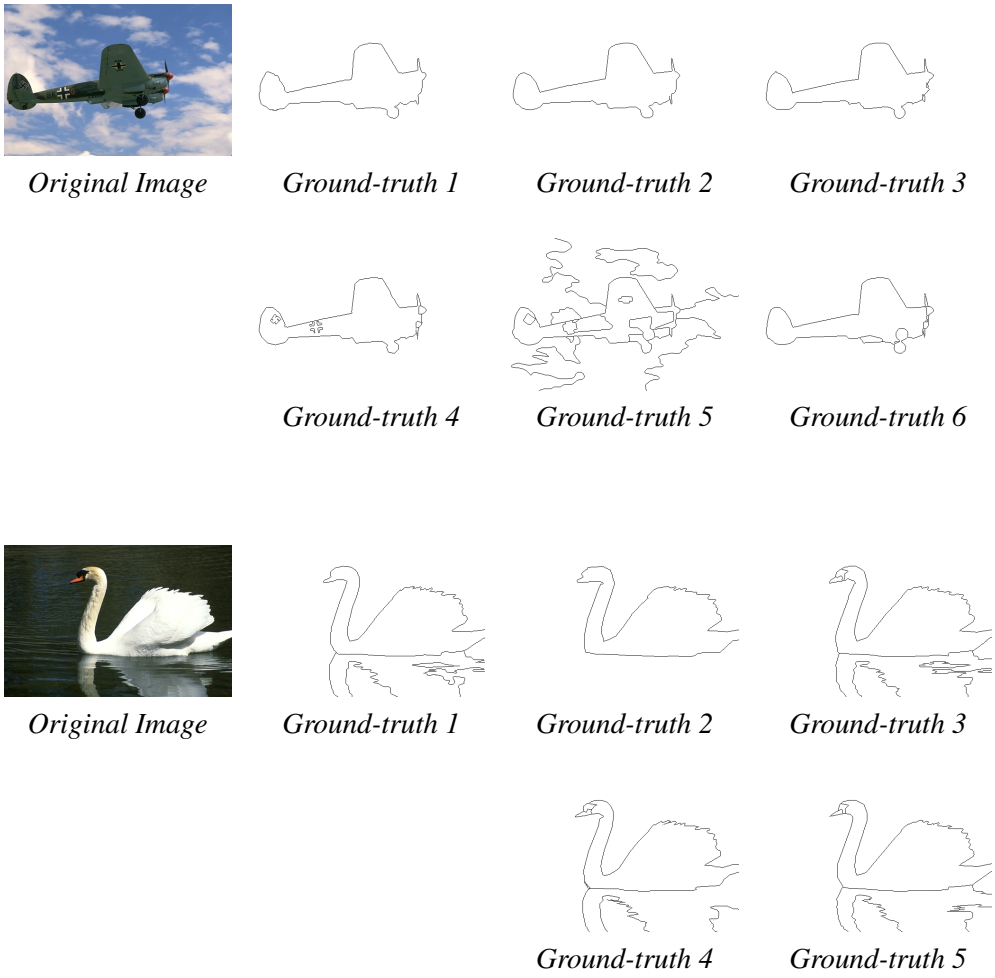


Figure 4.1: Ground-truth of some images from BSD500 [4] dataset.

4.2 Experiments

For the experiments; Prewitt, Roberts, Sobel, Canny, Laplacian of Gaussian(LoG), i1D, Line Segment Detector and probabilistic Canny (pbCanny) [4] methods are used for edge-like confidence. All edge detection methods, except pbCanny, are mentioned earlier. PbCanny does not give a hard edge map, it gives a probability for the edge-like confidence by using Canny edge detection.

To test IC, various window size are experimented. The effect of the window size can be seen in Figure-4.2. Choosing big window size will lose edge-like pixels. The experiments show that using 3x3 gives the best results, see Figure 4.3.

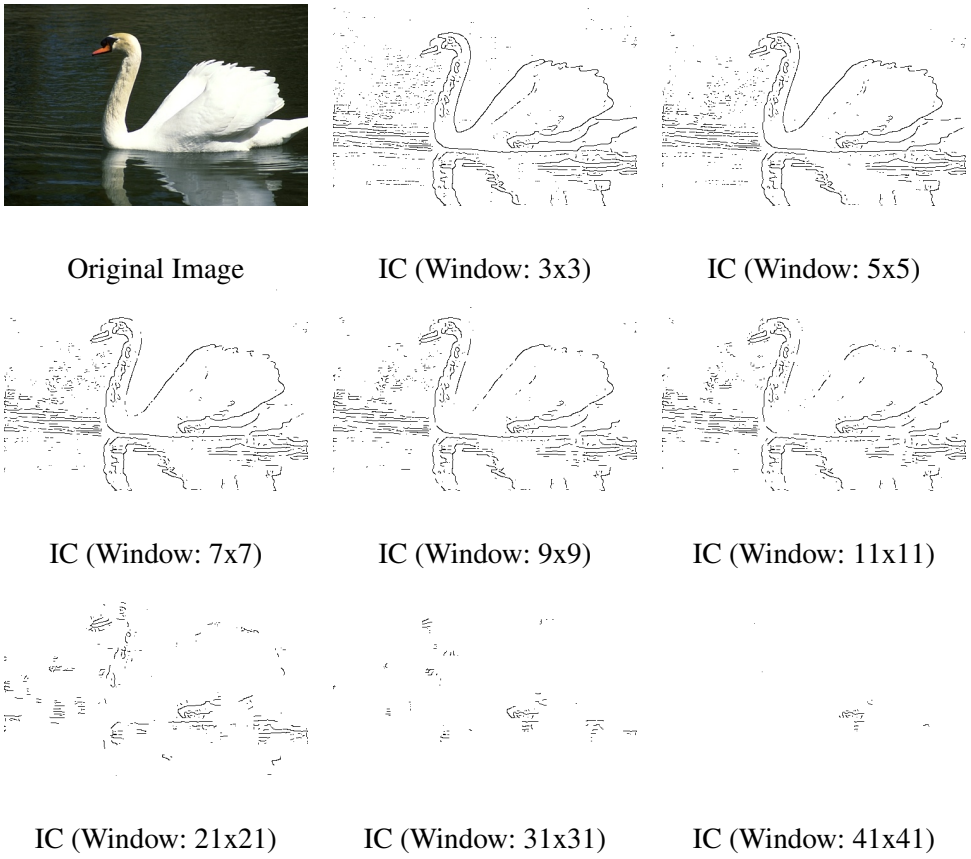


Figure 4.2: IC experiments, using different window sizes.

4.3 Performance Measures

For an objective evaluation, a quantitative measurement is needed. For that purpose, *precision (prec)*, *recall (rec)*, and *F-score (F)* [36] metrics and BSD500 [4] their own metrics are used. Before describing these metrics some basic terms should be defined:

True Positive (TP): A pixel found as an edge, that is in fact an edge.

False Positive (FP): A pixel found as an edge, that is in fact not an edge.

True Negative (TN): A pixel found as a non-edge, that is in fact not an edge.

False Negative (FN): A pixel found as non-edge, that is in fact an edge.

Based on these terms:

Precision (prec): Proportion of the retrieved relevant edge pixels to the whole retrieved edge pixels.

$$prec = \frac{TP}{TP + FP} \tag{4.1}$$

Recall (rec): Proportion of the retrieved relevant edge pixels to the whole relevant edge pixels.

$$rec = \frac{TP}{TP + FN} \tag{4.2}$$

F-score (F): It is defined as a harmonic mean of precision and recall. Since it takes precision and recall into account, it is more balanced.

$$F = 2 \cdot \frac{prec \cdot rec}{prec + rec} \tag{4.3}$$

BSD500 [4] has its own metrics that are based on F-score, precision and recall:

Optimal Dataset Scale (ODS): The best F-score on the dataset for a fixed scale.

Optimal Image Scale (OIS): The aggregate F-score on the dataset for the best scale in each image.

Average Precision (AP): The average precision on the full recall range which equals the area under the precision-recall curve.

4.4 Validation Tests

To obtain the best result, IC window size and power values are determined by the experiments on training dataset. 3x3 window size and around 1.5 power value gives best IC result, see Figure 4.3. The IC result is scaled to the gray-level values that are varies between 0 and 255. To obtain the final edge map (FEM), the IC result is thresholded with the gray-level values that are obtained from training. From training, all IC results are thresholded between 20 and 220 values. The value which gives the best result is accepted as the threshold value for the testing. Each FEM that are obtained from thresholding, compared with the ground-truths corresponding to each image. For each image, an average precision and recall is computed from that comparisons. Finally, the best threshold value is chosen that generates the best FEM according to the final f-score. The computation of the comparison is shown in Algorithm-2.

The human subjected ground-truths and computer detected FEMs do not match exactly. For that reason, to make the comparison we developed an algorithm. We slide a window on each ground-truth that is centered on the exact location of the FEM's location. If any pixel from FEM is an edge then the corresponding sliding window on the ground-truth is observed. If also any pixel in the window is an edge then the pixel in the FEM is assumed to be an edge, otherwise not. For comparison we decide to slide 5x5 window. It is shown in Figure 4.4 - 4.5 - 4.6 that IC removes some spurious edges.

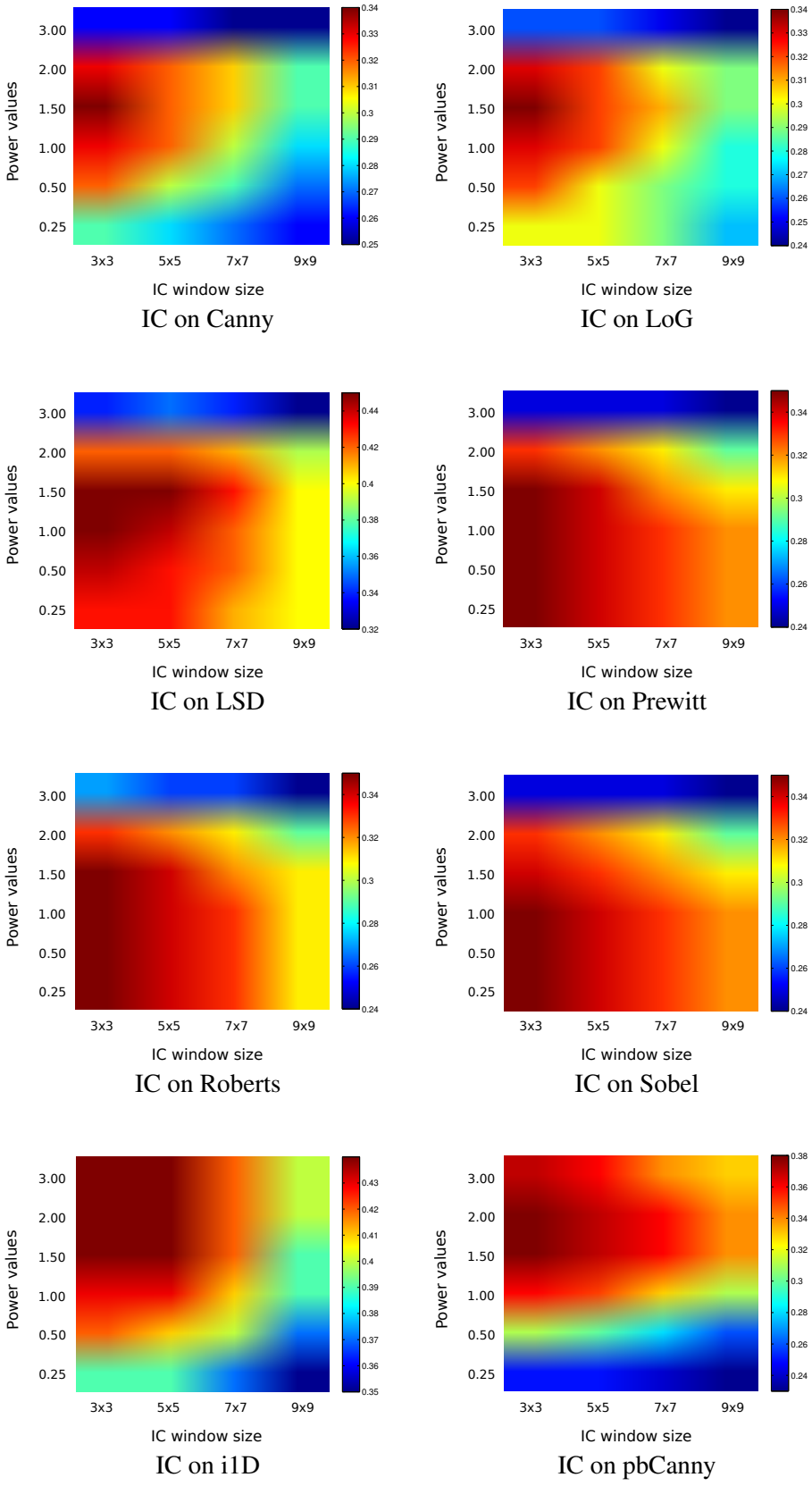


Figure 4.3: IC results in f-score, with respect various window size and power values.

Algorithm 2 Comparison method.

input : Intersection Consistency Map *(IC)*, Ground-Truth Edge Maps *(GTs)*

Gray-Level Value for Thresholding *(ThrVal)*, Window-Size (w)

output: Precision *(Prec)*, Recall *(Rec)*, F-Measure (F), True Positive Pixels *(TP)*,

False Positive Pixels *(FP)*, False Negative Pixels (FN)

Average Precision *(AvgPrec)*, Average Recall *(AvgRec)*

$FEM \leftarrow IC > ThrVal$; FEM pixels valued with 1 when conforms to the condition, otherwise 0.

N $\leftarrow$ number of ground-truths for each image; $[NumRows, NumCols] \leftarrow Size(FEM)$

for $i \leftarrow 1$ **to** N **do**

  [TP,FP,FN] $\leftarrow$ **getPixelClass**$(FEM, GTs(\text{i}), w)$;

  $Prec(\text{i}) \leftarrow (TP)/(TP + FP)$; $Rec(\text{i}) \leftarrow (TP)/(TP + FN)$

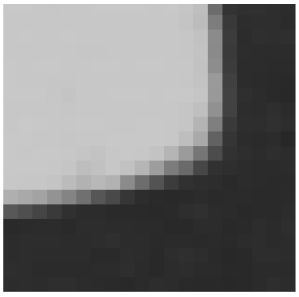
end

$$AvgPrec \leftarrow \frac{\sum_{i=1}^{N} Prec(i)}{N}; \quad AvgRec \leftarrow \frac{\sum_{i=1}^{N} Rec(i)}{N}; \quad F \leftarrow \frac{2 \cdot AvgPrec \cdot AvgRec}{(AvgPrec + AvgRec)}$$

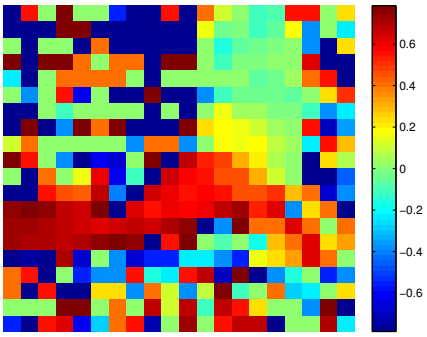
function getPixelClass: $(FEM, GTs, w) \rightarrow (TP, FP, FN)$ **begin**

```
for i ← w+1 to NumRows − w do
    for j ← w+1 to NumCols + w do
        if FEM(i, j) = 1 then
            if FEM(i, j) = GTs(i, j) then
                TP ← TP + 1;
            else
                sumOfOnes = Σ_{i−w}^{i+w} Σ_{j−w}^{j+w} GTs(i, j);
                if sumOfOnes > 1 then
                    TP ← TP + 1;
                else
                    FP ← FP + 1;
                end
            end
        else
            if FEM(i, j) ≠ GTs(i, j) then
                FN ← FN + 1;
            end
        end
    end
end
```

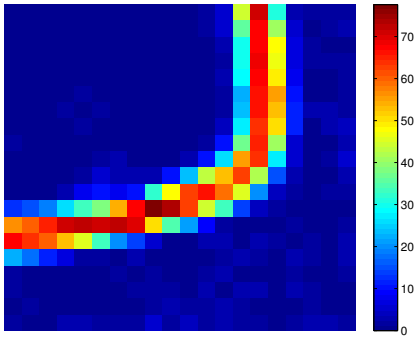
end



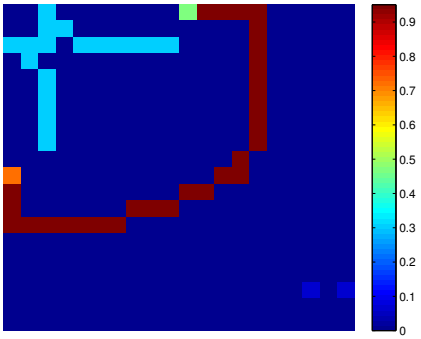
a) Original Image



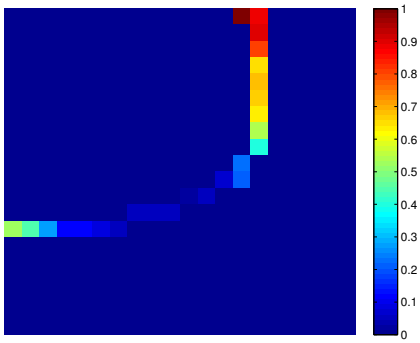
b) Orientation Map



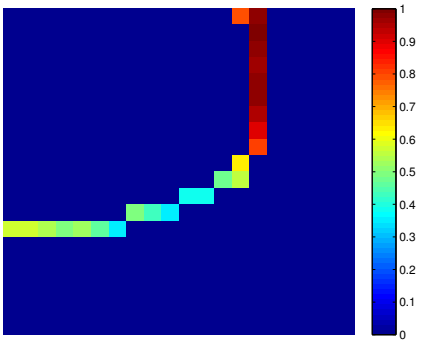
c) Gradient Magnitude



d) pbCanny

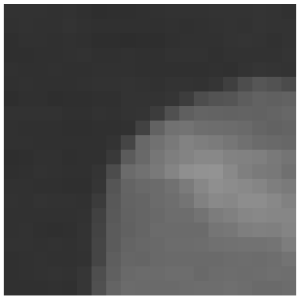


e) IC (weighted by ec)

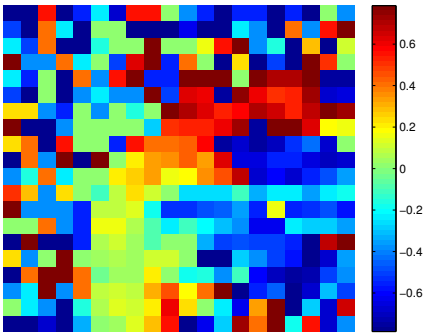


f) IC (weighted by mag)

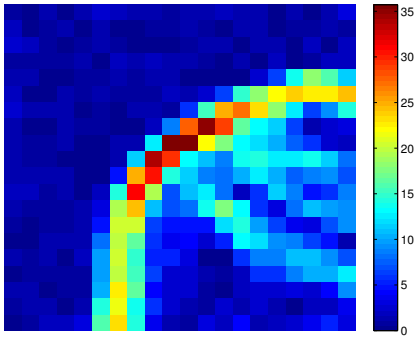
Figure 4.4: IC test-1. Using IC window size:5x5, power:1, and pbCanny as edge-like confidence: a) A 20x20 image patch cropped from WoodBlock [3] dataset, b) edge orientation map of the image patch, scaled in radiance, c) gradient magnitude of the image patch, d) pbCanny result for the image patch, the scale shows the edge-like confidence, e) IC result, weighted by edge-like confidence, f) IC result, weighted by image gradient magnitude.



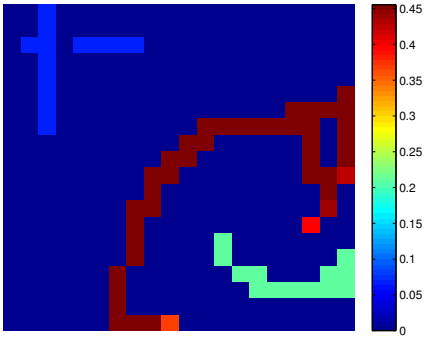
a) Original Image



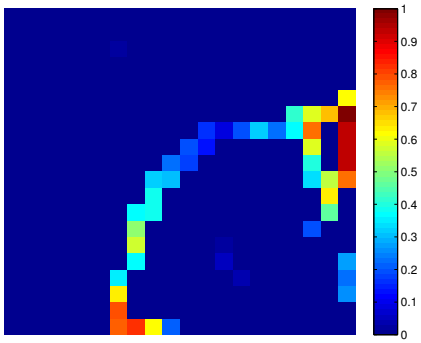
b) Orientation Map



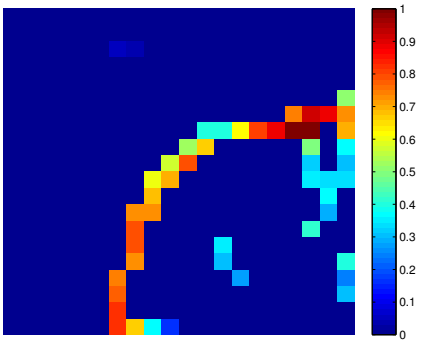
c) Gradient Magnitude



d) pbCanny



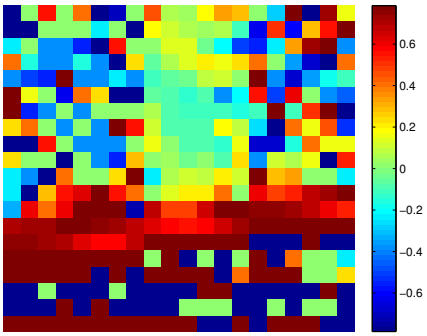
e) IC (weighted by ec)



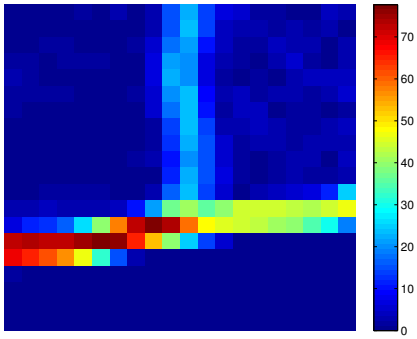
f) IC (weighted by mag)

Figure 4.5: IC test-2. Using IC window size:5x5, power:1, and pbCanny as edge-like confidence: a) A 20x20 image patch cropped from WoodBlock [3] dataset, b) edge orientation map of the image patch, scaled in radiance, c) gradient magnitude of the image patch, d) pbCanny result for the image patch, the scale shows the edge-like confidence, e) IC result, weighted by edge-like confidence, f) IC result, weighted by image gradient magnitude.

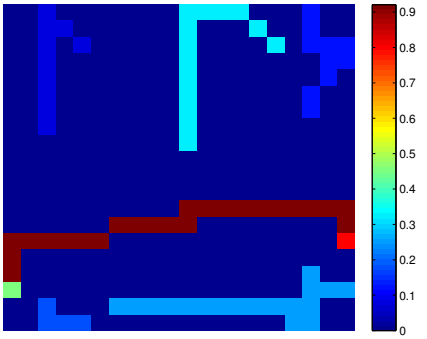
a) Original Image



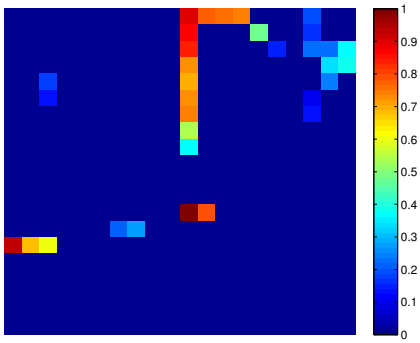
b) Orientation Map



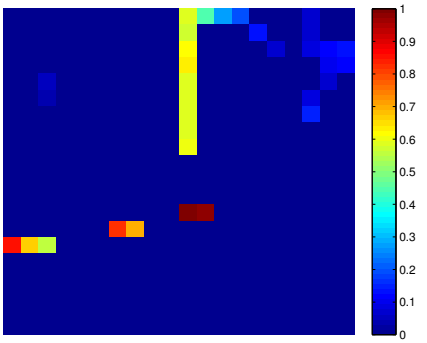
c) Gradient Magnitude



d) pbCanny



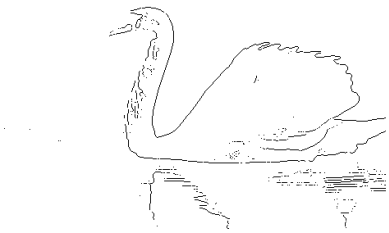
e) IC (weighted by ec)



f) IC (weighted by mag)

Figure 4.6: IC test-3.Using IC window size:5x5, power:1, and pbCanny as edge-like confidence: a) A 20x20 image patch cropped from WoodBlock [3] dataset, b) edge orientation map of the image patch, scaled in radiance, c) gradient magnitude of the image patch, d) pbCanny result for the image patch, the scale shows the edge-like confidence, e) IC result, weighted by edge-like confidence, f) IC result, weighted by image gradient magnitude.

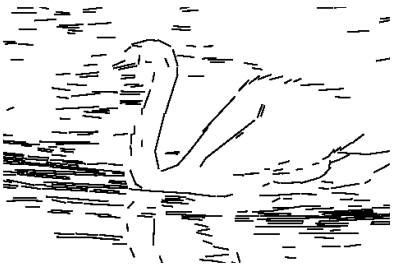
a) Canny (σ=1)



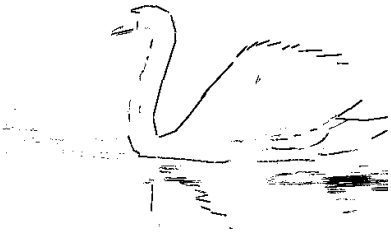
b) IC (edge confidence: Canny)

c) LoG (σ=2)

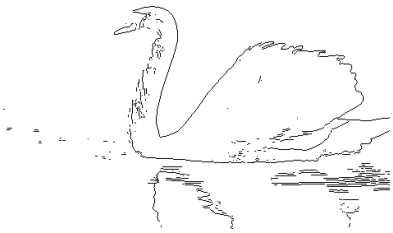
d) IC (edge confidence: LoG)



e) LSD



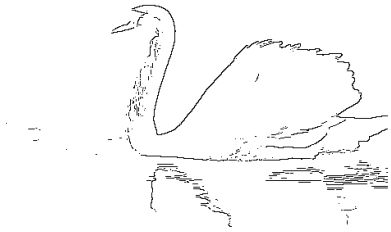
f) IC (edge confidence: LSD)



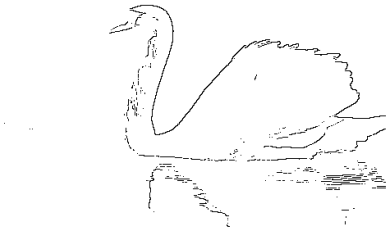
e) Prewitt

f) IC (edge confidence: Prewitt)

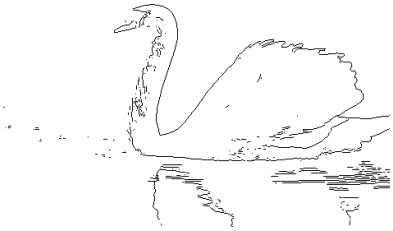
Figure 4.7: IC Results.



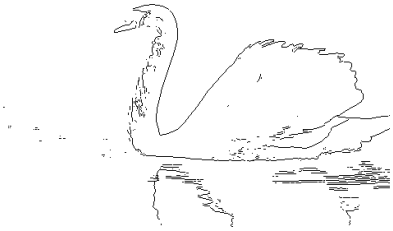
a) Roberts



b) IC (edge confidence: Roberts)



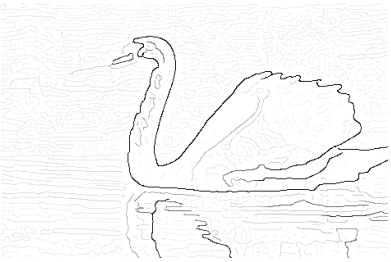
c) Sobel



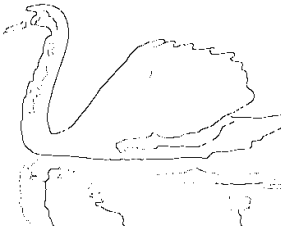
d) IC (edge confidence: Sobel)

e) i1D

f) IC (edge confidence: i1D)



e) pbCanny



f) IC (edge confidence: pbCanny)

Figure 4.8: IC Results.

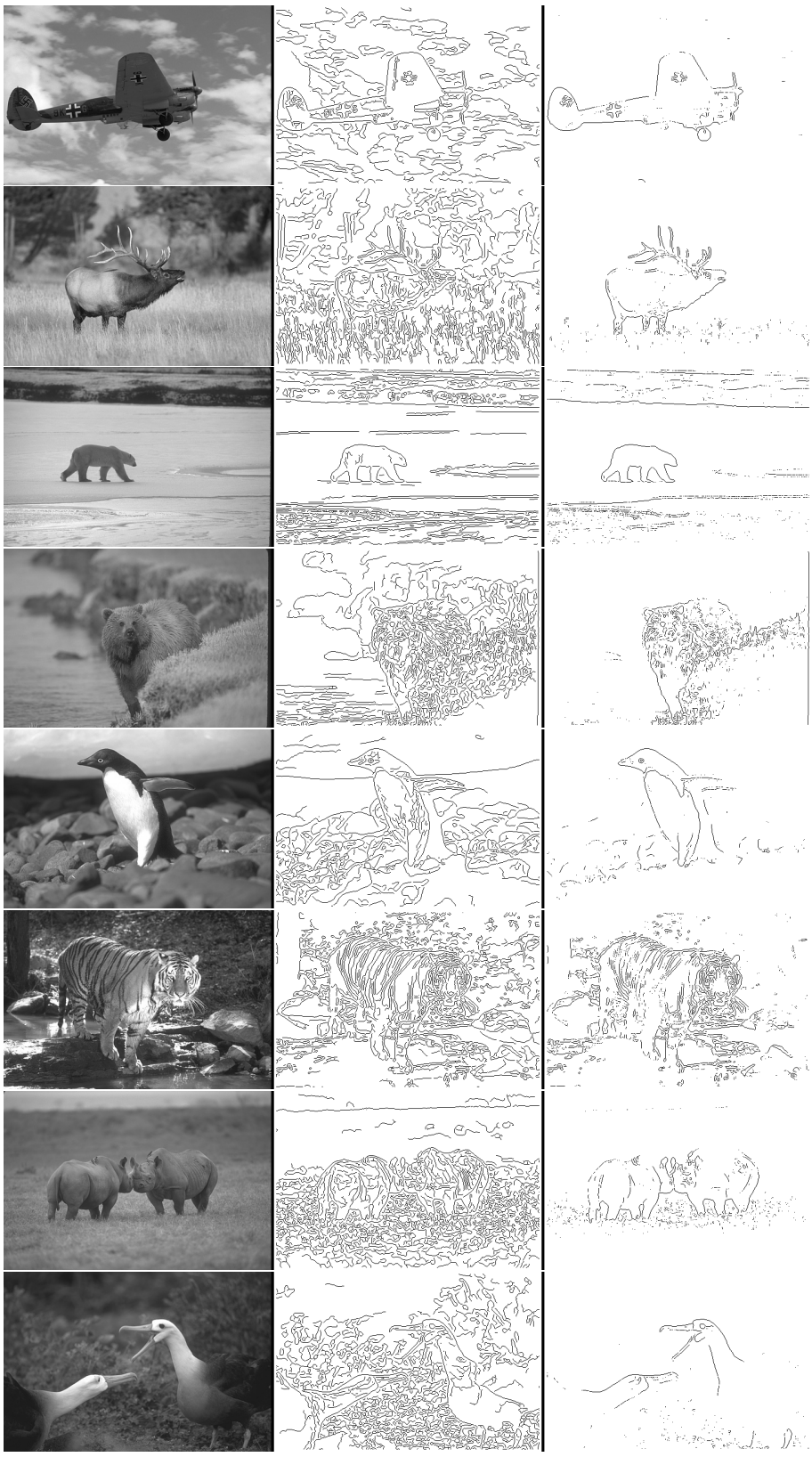


Figure 4.9: IC Results, images are ordered namely as original image, Canny, and IC on Canny.

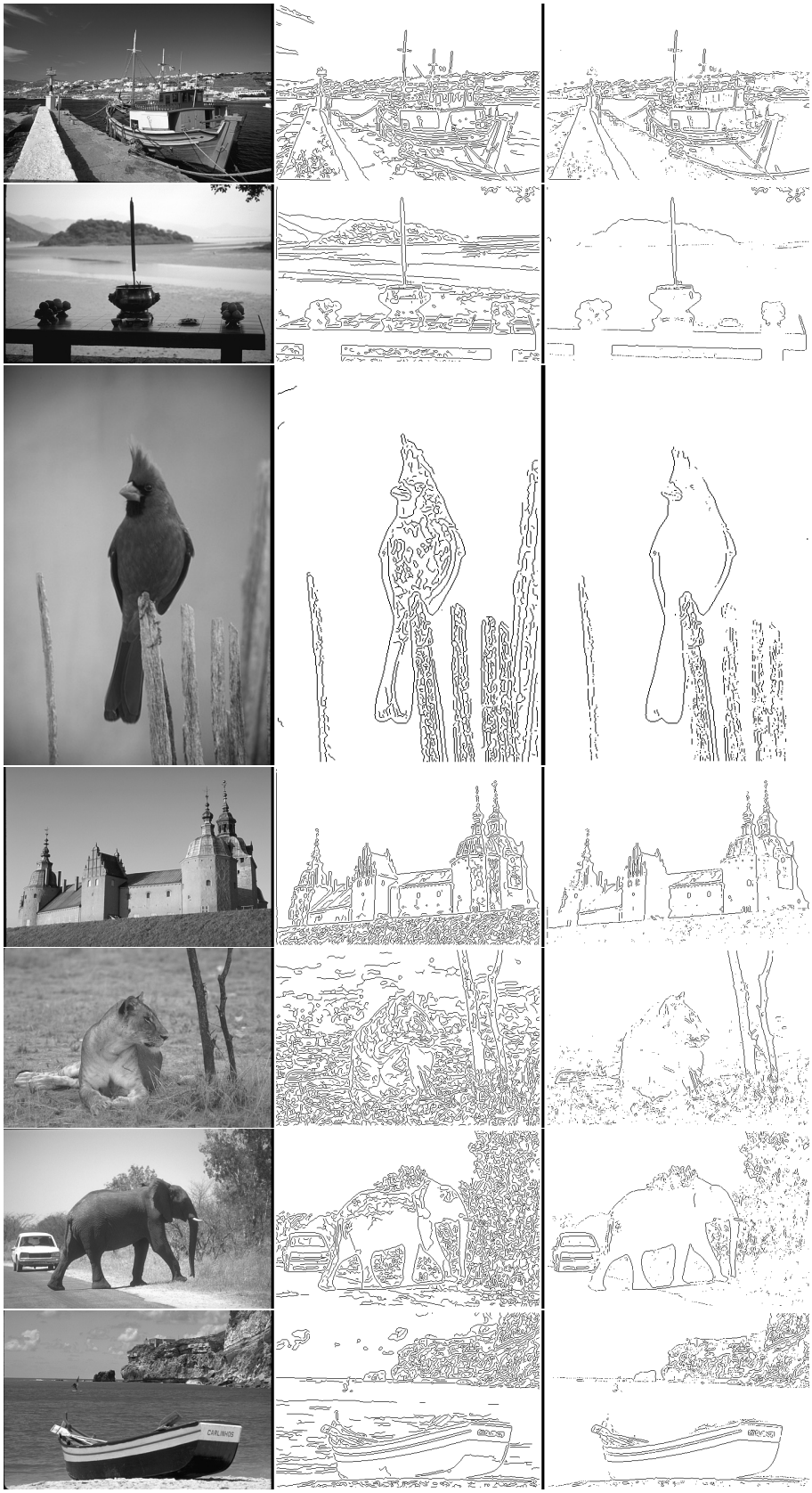


Figure 4.10: IC Results, images are ordered namely as original image, Canny, and IC on Canny.

Table 4.1: IC results by using our comparison method given in Algorithm 2, provided different edge confidences. σ: standard deviation of Gaussian filter, w: IC window size, p: IC edge confidence power, weight: IC edge confidence, can be ec or mag (ec: edge-confidence,edge-maps; mag: image gradient magnitude), t:threshold-value, Prec: Precision, CIP: Change In Precision, Rec: Recall, CIR: Change In Recall, F: F-score, CIF: Change In F-score.

	Prec	**CIP(%)**	**Rec**	**CIR(%)**	**F**	**CIF(%)**
Canny (σ=1,t=100) **IC** (w=3,p=1.5,weight=mag,t=25)	0.18 **0.29**	**61.11**	0.56 **0.40**	**-28.57**	0.28 **0.34**	**21.43**
LoG (σ=2,t=100) **IC** (w=3,p=1.5,weight=mag,t=25)	0.21 **0.30**	**42.86**	0.51 **0.38**	**-25.49**	0.30 **0.34**	**13.33**
LSD (t=100) **IC** (w=3,p=1.5,weight=mag,t=20)	0.33 **0.43**	**30.30**	0.62 **0.50**	**-19.35**	0.43 **0.46**	**6.98**
Prewitt (t=100) **IC** (w=3,p=1.5,weight=mag,t=20)	0.31 **0.32**	**3.23**	0.40 **0.38**	**-5.00**	0.35 **0.35**	**0.00**
Roberts (t=100) **IC** (w=3,p=1.5,weight=mag,t=20)	0.34 **0.35**	**2.94**	0.37 **0.35**	**-5.41**	0.35 **0.35**	**0.00**
Sobel (t=100) **IC** (w=3,p=1,weight=mag,t=20)	0.31 **0.31**	**0.00**	0.40 **0.40**	**0.00**	0.35 **0.35**	**0.00**
i1D (t=95) **IC** (w=3,p=1.5,weight=ec,t=35)	0.33 **0.32**	**-3.03**	0.66 **0.72**	**9.09**	0.44 **0.44**	**0.00**
pbCanny (t=75) **IC** (w=3,p=1.5,weight=ec,t=20)	0.36 **0.37**	**2.78**	0.45 **0.39**	**-13.33**	0.40 **0.38**	**-5.00**

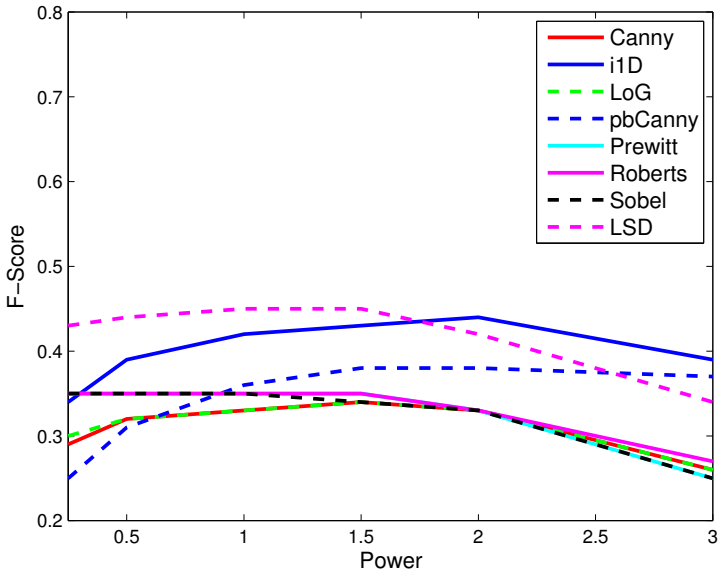


Figure 4.11: Change in IC (with 3x3 IC window) by using various edge-like confidence power. Results are obtained by using Algorithm 2. For edge-like confidence, the output of edge detectors that are shown in legend are used.

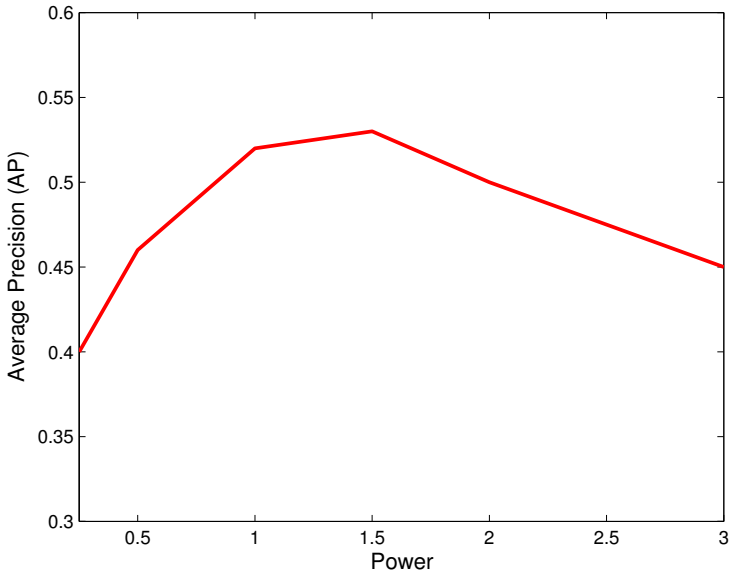


Figure 4.12: Change in IC (with 3x3 IC window) by using various edge-like confidence power. Results are obtained by using BSD [4] benchmarking, for edge-like confidence pbCanny is used.

4.5 Discussion

It is shown in Figure-4.11 and in Figure-4.12 that, changing the edge-like confidence power effects the results. This power value generally gives its best result by around 1.5. Using 3x3 IC window gives the best results, see Figure 4.3. Weighting IC with image gradient magnitude generally gives better results than weighting with the edge-confidence. IC works well on images that contain prominent objects which are different in color from their surroundings. IC give good results on natural images that have especially cluttered background, see Figure 4.9 and 4.10. On images involving human made objects, IC leads to good results as well. But, depending on the amount of clutter, the loss of true positives might be more crucial. Through our comprehensive investigation, we show that approximately 21% increase in f-score is obtained whereas some important edges are lost. The ratio of false positives that are removed is greater than the removed true positive ones. For that reason, the ratio of increase in precision is greater than the decrease in recall. We conclude from our experiments that IC is suitable for improving the quality of edge detection in some detectors such as Canny, LoG and LSD.

CHAPTER 5

CONCLUSION

In this study, we test whether a local consistency measure based on image orientation (which we call Intersection Consistency - IC), which was previously shown to improve detection of junctions, can be used for improving the quality of edge detection of seven different detectors; namely, Canny, Roberts, Prewitt, Sobel, Laplacian of Gaussian, Intrinsic Dimensionality, Line Segment Detector. In order to find the best result, we experiment the original IC with various window sizes and power values.

We tried IC with 3x3, 5x5, 7x7 and 9x9 window sizes. We found that the window size of IC effects the results. Choosing big window size remove noise but caused loss in true positive edges. The experiments showed that choosing 3x3 for IC window gives the best result in all edge detectors.

We test IC with various edge-like confidence power. We experimented with values 0.25, 0.5, 1, 1.5, 2, and 3 as the power in Equation 3.1. The results showed that IC gives better results almost in all detectors around power of 1.5 except Sobel detector which is around power of 1.

Lastly, we experiment IC with weighting by image gradient instead of edge-like confidence value. i1D and pbCanny give better results with weighting by edge-like confidence value but the rest detectors give better results by weighting with image gradient.

IC works well on images that contain prominent objects which are different in color from their surroundings. IC give good results on natural images that have especially cluttered background. On images involving human made objects, IC leads to good results as well. But, depending on the amount of clutter, the loss of true positives might be more crucial. Through our comprehensive investigation, we show that approximately 21% increase in f-score is obtained

whereas some important edges are lost. We conclude from our experiments that IC is suitable for improving the quality of edge detection in some detectors such as Canny, LoG and LSD.

As a future work, IC can be experimented with weighting by an inverse Gaussian in one dimension, according to the Euclidean distance between center pixel of the patch and pixels in the patch. By this experiment, the points that are far from the center pixel are assumed to be as edges. Moreover, IC can be tried by discarding the Euclidean distance normalization, to see whether closeness of the pixels in the patch to the center pixel is important or not. Another point that can be pursued is to determine the parameters just window size, power value by histogram of edges, and threshold value by adaptive thresholding. Finally IC can be tested with different databases and edge detectors (e.g., Steerable filters, Edge likelihood index method, and Fan et al. [9] Isotropic color edge detection method.)

REFERENCES

[1] HALCON, Vision Toolbox. http://www.mvtec.com/download/reference/laplace_of_gauss.html. [Last accessed on 09-09-2011].

[2] MATLAB, Image Processing Toolbox. http://www.mathworks.com/help/toolbox/images/ref/edge.html. [Last accessed on 09-09-2011].

[3] WoodBlock Dataset. http://figment.csee.usf.edu/edge/sfm/#Dataset. [Last accessed on 08-08-2011].

[4] Arbelaez, P., Maire, M., Fowlkes, C., and Malik, J. Contour Detection and Hierarchical Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 33*, 5 (May 2011), 898–916.

[5] Burns, J. B., Hanson, A. R., and Riseman, E. M. Extracting Straight Lines. *IEEE Transactions on Pattern Analysis and Machine Intelligence 8* (1986), 425–455.

[6] Canny, J. A Computational Approach to Edge-detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence 8*, 6 (Nov 1986), 679–698.

[7] Desolneux, A., Moisan, L., and Morel, J. M. *From Gestalt Theory to Image Analysis: A Probabilistic Approach*. Springer, 2008.

[8] Ding, L. J., and Goshtasby, A. On the Canny Edge Detector. *Pattern Recognition 34*, 3 (Mar 2001), 721–725.

[9] Fan, J., Aref, W., Hacid, M., and Elmagarmid, A. An Improved Automatic Isotropic Color Edge Detection Technique. *Pattern Recognition Letters 22*, 13 (Nov 2001), 1419–1429.

[10] Felsberg, M., Kalkan, S., and Kruger, N. Continuous Dimensionality Characterization of Image Structures. *Image and Vision Computing 27*, 6 (May 4 2009), 628–636.

[11] Forstner, W. A Framework for Low Level Feature Extraction. In *ECCV* (1994), pp. B:383–394.

[12] Forsyth, D. A., and Ponce, J. *Computer Vision: A Modern Approach*. Prentice Hall, 2003.

[13] He, X., and Yung, N. Performance Improvement of Edge Detection Based on Edge Likelihood Index. In *Visual Communications and Image Processing 2005, Pts 1-4* (2005), vol. 5960 of *Proceedings of the Society of Photo-Optical Instrumentations Engineers (SPIE)*, pp. 1664–1673.

[14] Hubel, D. H., and Wiesel, T. N. Anatomical Demonstration of Columns in the Monkeys Striate Cortex. *Nature 221* (1969), 747–750.

[15] Jain, R. C., Kasturi, R., and Schunck, B. G. *Machine vision*. McGraw-Hill, 1995.

[16] Kahn, P., Kitchen, L., and Riseman, E. A Fast Line Finder for Vision-Guided Robot Navigation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 12*, 11 (Nov 1990), 1098–1102.

[17] Kahn P., K. L., and E.M., R. Real-Time Feature Extraction: A Fast Line Finder for Vision-Guided Robot Navigation. Tech. Rep. 87-57, COINS, 1987.

[18] Kalkan, S., Shi, Y., Pilz, F., and Krüger, N. Improving Junction Detection by Semantic Interpretation. In *VISAPP (1)* (2007), pp. 264–271.

[19] Kang, C.-C., and Wang, W.-J. A Novel Edge Detection Method Based on the Maximizing Objective Function. *Pattern Recognition 40*, 2 (Feb 2007), 609–618.

[20] Konderink, J., and Vandoorn, A. The Shape of Smooth Objects and the Way Contours End. *Perception 11*, 2 (1982), 129–137.

[21] Krüger, N., and Felsberg, M. A Continuous Formulation of Intrinsic Dimension. In *Proceedings of the British Machine Vision Conference* (2003).

[22] Marr, D. *Vision*. W.H. Freeman Company, New York, 1982.

[23] Marr, D., and Hildreth, E. Theory of Edge Detection. *Proceedings of the Royal Society of London Series B-Biological Sciences 207*, 1167 (1980), 187–217.

[24] Mokhtarian, F., and Mackworth, A. A Theory of Multiscale, Curvature-Based Shape Representation for Planar Curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence 14*, 8 (Aug 1992), 789–805.

[25] Nadernajad, E., Sharifzadeh, S., and Hassanpour, H. Edge Detection Techniques: Evaluation and Comparisons. *Applied Mathematical Sciences 2*, 31 (2008), 1507–1520.

[26] Oskoei, M. A., and Hu, H. A Survey on Edge Detection Methods. Tech. Rep. CES-506, School of Computer Science Electronic Engineering, University of Essex, United Kingdom, February 2010.

[27] Parida, L., Geiger, D., and Hummel, R. Junctions: Detection, Classification, and Reconstruction. *IEEE Transactions on Pattern Analysis and Machine Intelligence 20*, 7 (1998), 687–698.

[28] Poggio, T. Marr's Approach to Vision. Tech. Rep. AIM-645, MIT Artificial Intelligence Laboratory, Aug. 6 1981.

[29] Prewitt, J. M. S. *Object Enhancement and Extraction*. Academic Press, 1970, pp. 75–149.

[30] Roberts, L. *Machine Perception of 3-D Solids*. MIT Press, 1965, pp. 159–197.

[31] Rothwell, C. A., Mundy, J. L., Hoffman, W., and Nguyen, V.-D. Driving Vision by Topology. In *IEEE International Symposium on Computer Vision* (1995), pp. 395–400.

[32] Shin, M. C., Goldgof, D. B., and Bowyer, K. W. Comparison of Edge Detector Performance through Use in an Object Recognition Task. *Computer Vision and Image Understanding* (2001), 160–178.

[33] Sobel, I. E. *Camera Models and Machine Perception*. PhD thesis, Stanford, CA, USA, 1970. AAI7102831.

[34] Sonka, M., Hlavac, V., and Boyle, R. Marr's Theory. http://homepages.inf.ed.ac.uk/cgi/rbf/ CVONLINE/entries.pl?TAG370. [Last accessed on 08-07-2011].

[35] van Diepen, P. M. J., and De Graef, P. Line-drawing Library and Software Toolbox. Tech. Rep. 165, Laboratory of Experimental Psychology, University of Leuven, Belgium., 1994.

[36] van Rijsbergen, C. J. *Information Retrieval*, 2 ed. Butterworths, London, 1979.

[37] Vernon, D. *Machine Vision*. Prentice hall, Englewood Cliffs, 1991.

[38] von Gioi, R. G., Jakubowicz, J., Morel, J.-M., and Randall, G. LSD: A Fast Line Segment Detector with a False Detection Control. *IEEE Transactions on Pattern Analysis and Machine Intelligence 32*, 4 (2010), 722–732.

[39] Wang, B., and Fan, S. An Improved Canny Edge Detection Algorithm. *International Workshop on Computer Science and Engineering 1* (2009), 497–500.

[40] Wang, X., and Xue, H. An Improved Edge Detection Method for Image Corrupted by Gaussian Noise. In *Computer and Computing Technologies in Agriculture II, VOLUME 2* (2009), vol. 295 of *International Federation for Information Processing*, pp. 1153–1159.

[41] Yang, Y., Li, Z., and Li, H. An Improved Edge Detection Method Based on Topology. *MIPPR 2009: Multispectral Image Acquisition and Processing 7494*, 1 (2009).

[42] Ziou, D., and Tabbone, S. Edge Detection Techniques - An Overview. *International Journal of Pattern Recognition and Image Analysis 8* (1998), 537–559.