Fast Edge Detection by Center of Mass

Bo Li[a,*], Aleksandar Jevtic[b], Ulrik Söderström[a], Shafiq Ur Réhman[a], Haibo Li[c]
[a]Umeå University, X, Håken Gullessons väg 20, Teknikhuset, Umeå 90187, Sweden
[b]Robosoft, Technopole d'Izarbel - F-64210 Bidart, France, France
[c]KTH Royal Institute of Technology, Kungl Tekniska Högskolan, Stockholm 10044, Sweden

*Corresponding Author: boli@umu.se

Abstract

In this paper, a novel edge detection method that computes image gradient using the concept of Center of Mass (COM) is presented. The algorithm runs with a constant number of operations per pixel independently from its scale by using integral image. Compared with the conventional convolutional edge detector such as Sobel edge detector, the proposed method performs faster when region size is larger than 9×9. The proposed method can be used as framework for multi-scale edge detectors when the goal is to achieve fast performance. Experimental results show that edge detection by COM is competent with Canny edge detection.

Keywords: edge detection, center of mass, integral image, multi-scale, fast computing.

1. Introduction

Edge detection as a fundamental image processing method has been studied for decades[1-7]. In general, the aim of edge detection is to significantly reduce the amount of data in an image, while retaining the structural properties to be used for further image processing. The state-of-the-art gradient-based edge detectors lack scalability in the filter size. Small-scaled filters are sensitive to edge signals but also prone to noise, whereas large-scaled filters are robust to noise but can filter out fine details[7]. A small step in this direction has been accomplished by using multiple detectors and multiple scales as described in literatures[5,7]. The characterization of signals from multi-scale edges has been studied extensively in the literature[5]. Multi-scale edge detection face a runtime issue that when scale increases there will be a linear or quadratic increase of time consumption[16]. The general idea of edge detection is to first convolve the input image with a filter to obtain gradient. The complexity of this operation per pixel is usually $O(n)$ or $O(n^2)$, where n is the filter width. For example, Sobel edge detector which runs in $O(n)$ time takes 45 ms on 512×512 image with filter size of 31×31. This is not applicable for real-time algorithms. The time consumption issue will be fully analyzed in Section 4.

In our paper, we present a non-convolutional method using the concept of center of mass which provides competent result with Canny edge detector[1,17]. COM edge detector uses integral image (also known as a summed-area table[9]) to compute center of mass in constant time $O(1)$. We propose a linear equation which transforms center of mass to gradient in a simple fashion.

In section 2, we will give an overview of conventional edge detection method. Section 3 presents the proposed edge detection method using the concept of center of mass. Section 4 explains how to achieve fast COM edge detection using integral image, followed by speed analysis given in Section 5. Experimental results and discussion are provided in Section 6. Finally, conclusions are given in Section 7.

2. Background

Edges are normally representations of changes in intensity functions of an image; i.e., image intensity variations such as steps, lines and junctions[7]. The widespread edge detection methods detect edges by finding local maxima of first-order derivative function or zero-crossing of second-order derivative function of the intensity profile of given image. In practice, image gradients are estimated by convoluting images with first-order derivative operators (also known as kernels), such as Robert's cross operator, Prewitt operator and Sobel operator[2]. The kernel convolution finds the abrupt changes in intensity of the image. In case of Sobel kernel both horizontal and vertical changes are approximated. Formally if $I(x, y)$ is intensity

© 2013 The Institute of Industrial Applications Engineers, Japan.

level at a point in a given source image, and *Gx*, *Gy* are horizontal and vertical derivative approximations, then the gradient magnitude is given as:

$$G = Gx2 + Gy2 \quad (1)$$

Once the gradient magnitude is computed, the next step is to apply a threshold, to decide whether edges are present or not at an image point. Appropriate threshold will filter out most noise and keep edge points. After this step, the resulting edge is still thick. Canny[1] introduced the notion of non-maximum suppression (NMS) to find edges with one-pixel thickness by modifying Sobel's Method; which gives the gradient directions as:

$$\theta = \arctan(GyGx) \quad (2)$$

Non-maximum suppression is implemented by comparing the values of the gradient magnitude in the estimated gradient directions. Canny also introduce a hysteresis method to further reduce noise. Hysteresis uses the upper threshold to find the start of an edge. Then edge is traced from the start point, marking an edge whenever gradient value is above the lower threshold. Hysteresis step deletes the weak edge points that are not connected to a strong edge.

3. Edge Detection by Center of Mass

The definition of edges is abrupt changes in intensity functions of an image. If we calculate a local COM of non-edge locations within a region of certain size, the center of mass will be very close to the center of that region. Fig. 1 shows a 1D case of step edge whose intensity is presented on the vertical axes. It can be noted that the center of mass of the region *a* to *b* is far from the region center and COM of region *g* to *f* coincides with the region center. The distance between COM and center of region can indicate the change of intensity function. This allows the possibility of using COM to design a new edge detector.

The location of COM is given by the equation[8]:

$$XCOM = mixim_i \quad (3)$$

This is a vector equation that represents each of the three object dimensions in the physical world. We first try to estimate the gradient of image intensity by the distance between COM and the region center:

$$Gx' = c(XCOM - xc) \quad (4)$$

where c is a constant parameter and x_c is region center.

The intensity gradient based on equation (4) in a 1D case, is shown in Fig. 2. The gradient estimated by COM reflects step changes of image intensity. However, the local maxima of gradient are not accurately located on the step edge. For example, as shown in Fig. 3, (b) should yield larger gradient than (a) and (c). But due to the nature of COM, Fig. 3 (a) yields a bigger value and Fig. 3 (c) yields a smaller value. Is it possible to find a parameter which can convert equation (3) to a more accurate gradient

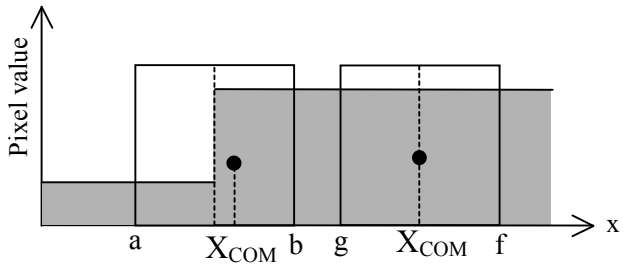


Fig. 1. Center of mass of 1D case.

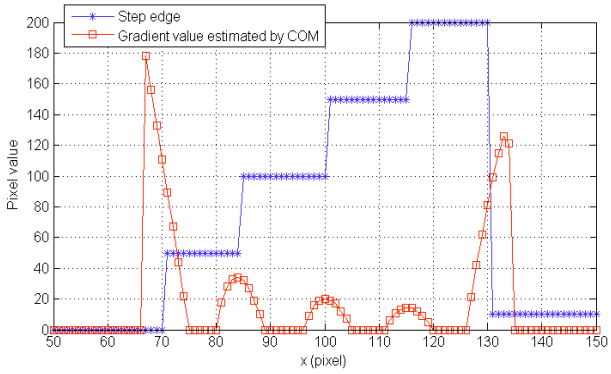


Fig. 2. Gradient estimation of 1D case using equation (4).

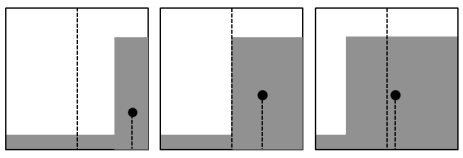


Fig. 3. Different case of center of mass.

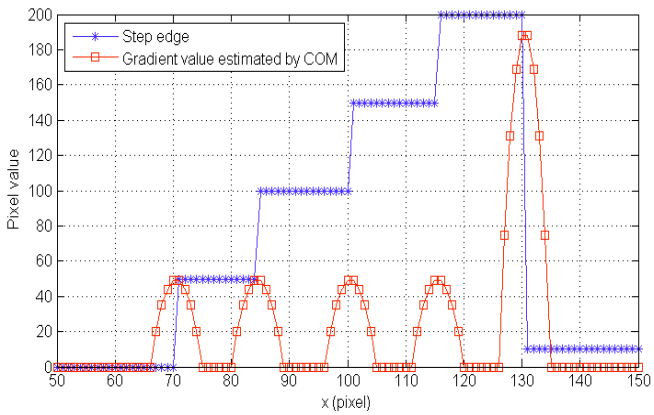
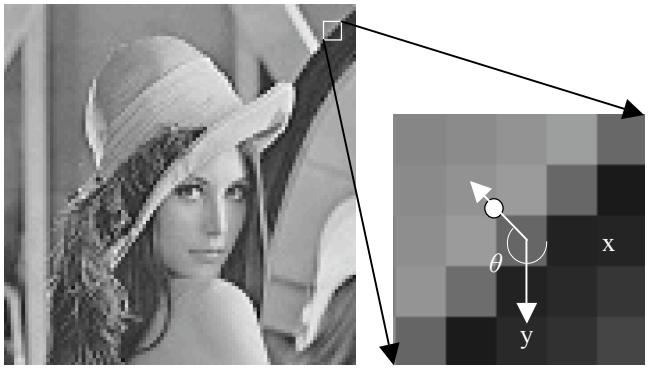


Fig. 4. Gradient estimation of 1D case using equation (5).



(a) (b)

Fig. 5. (a) Lena image; (b) enlarged part of image, where red circel denotes center of mass.

estimation? We realize that the total mass (or intensity) shown in Fig. 3 (a) is smaller than the one in Fig. 3 (b) and the total mass of the region shown in Fig. 3 (c) is larger than the one shown in Fig. 3 (b). We modify equation (3) by multiplying it with the total of mass in order to balance its inaccuracy:

$$Gx=aGx'mi \qquad (5)$$

where a is a constant parameter. In Fig. 4, it can be seen that by applying equation (5), the estimated gradient perfectly reflects the step change with respect to the gradient magnitude and location.

Fig. 1 to 4 above illustrates the 1D case of COM. Nevertheles, the COM based method can be applied to a 2D image. Digital image is discrete object where each pixel as a particle has its own mass. In image processing, such mass is referred as intensity. From equations (3) to (5), directional gradient of an image is calculated as:

$$Gx=a(I(x,y)xI(x,y)-xc)I(x,y) \qquad (6)$$

$$Gy=a(I(x,y)yI(x,y)-yc)I(x,y) \qquad (7)$$

where $I(x,y)$ is image intensity, (x_c, y_c) is the center of the region (called kernel in conventional edge detection method), f all denotes $x1< x\leq x2,\quad y1< y\leq y2f$, x_1, x_2, y_1, y_2 are the boundary of calculated region. In Fig. 5 (b), a 5×5-pixel region from Lena image is shown. Pixel intensity of a gray scale image can obtain a range of values from 0 to 255. Brighter pixels have heavier mass, i.e. they are heavier. Therefore, COM is closer to brighter area, as shown by the white circle. Inheriting from conventional edge detection method[16], image gradient and gradient direction are calculated as:

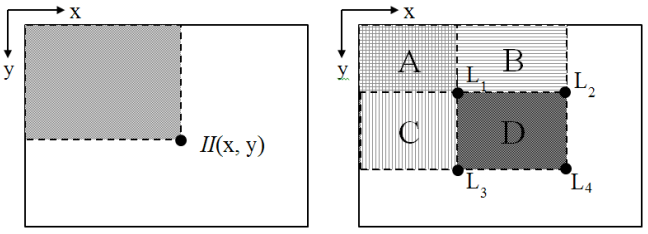
$$G= Gx2+Gy2 \qquad (8)$$

$$\theta=arctan(GyGx) \qquad (9)$$

In Fig. 5 (b) θ is -135°. Having close look at equation (6) and (7), it involves calculating sum of functions which can be computed rapidly by integral image.

4. Fast Computing by Integral Image

An integral image (also known as a summed-area table[9]) is a tool that can be used whenever we have a function from pixels to real numbers $f(x, y)$ (for instance, pixel intensity), and we wish to compute the sum of this function over a rectangular region of the image[12]. Some research work that applied integral images include texture mapping[9], face detection in images[10], and stereo correspondence[11]. Without using integral image, the sum



(a) (b)

Fig. 6: Integral image. (a) integral image; (b) region D can be computed summing the following four array references: $L4+L1-(L2+L3)$.

is computed in quadratic time per rectangle by calculating the value of the function for each pixel individually. However, by using the integral image, sum can be calculated in constant number of operations per rectangle if we need to compute the sum for every pixel. Following liturature[12], the integral image, denoted $II(x,y)$, at location (x,y) contains the sum of the pixel values above and to the left of (x,y) (see Fig. 6 (a)), formally,

$$IIx,y=x'\leq xy'\leq yIx',y' \quad (10)$$

where $II(x,y)$ is the input image. The integral image can be computed in single run over the image using the following recurrence relation:

$$ss,y=sx,y-1+Ix,y \quad (11)$$

$$IIx,y=IIx-1,y+s(x,y) \quad (12)$$

where $s(x,y)$ denotes the cumulative row sum and $sx,-1=II(-1,y)\equiv 0$.

Given an integral image, the sum of pixel values within a rectangular region of the image aligned with the coordinate axes can be computed with four array references (i.e., constant time). For example, the region D in Fig. 6 (b) can be computed using the following four array references: $L4+L1-(L2+L3)$ [12]. To compute an integral image, we need 4 times of operation for each pixel. To calculate a summed area, we need four times operation for each pixel.

To solve equation (6) and (7), we need tree integral images:

$$IIx,y=x'\leq xy'\leq yIx',y' \quad (13)$$

$$IIxx,y=x'\leq xy'\leq yIxx',y'x' \quad (14)$$

$$IIyx,y=x'\leq xy'\leq yIyx',y'y' \quad (15)$$

Equation (14) and (15) use one operation of multiplication for each pixel. Therefore, to get these integral images, we need 14 operations for each pixel. Combining with equation (6) and (7), we get:

$$Gx(xc,yc)=a(BA-xc)A \quad (16)$$

$$Gy(xc,yc)=a(CA-yc)A \quad (17)$$

Where A denotes $IIxc+w2,yc+w2+IIxc-w2,yc-w2-IIxc-w2,yc+w2-IIxc+w2,yc-w2$,
B denotes $IIxxc+w2,yc+w2+IIxxc-w2,yc-w2-IIxxc-w2,yc+w2-IIxxc+w2,yc-w2$,
and C denotes $IIyxc+w2,yc+w2+IIyxc-w2,yc-w2-IIyxc-w2,yc+w2-IIyxc+w2,yc-w2$.

where w is the width of square region. To calculate equation (16) and (17), we need totally 22 operations for each pixel. Together with integral image building time, to get two directional gradients, each pixel needs 36 operations. The time consumption is invariant to region size because we use integral image. For multi-scale edge detection, the scales size can be up to 31×31. Our proposed algorithm provides a good methodology to speed up multi-scale detection. In next section, we compare the time consumption of edge detection between COM method and Sobel edge detection through different scales.

5. Time Consumption

The reason that we compare our method with Sobel edge detector in the sense of speed is because Sobel is widely used in image processing and it is a separable filter which can be calculated rapidly in a linear time rather than quadratic. Therefore, Sobel filter is a good reference to evaluate the speed of proposed algorithm.

Example of 3×3 separable Sobel filter is explained in reference[13,14]. The 3×3 Sobel operator employs two 3×3 kernels to convolve with the original image and approximate the derivatives – one in horizontal direction and other in vertical. Formally, let $\boldsymbol{I}$ be the source image, then the directional derivatives are:

$$Gx=+10-1+20-2+10-1*I \quad (18)$$

$$Gy=+1+2+1000-1-2-1*I \quad (19)$$

Sobel kernel is a separable filter which can be decomposed as the products of an averaging and a differentiation kernel. Horizontal and vertical filter can be written as:

$$+10-1+20-2+10-1=121+10-1 \quad (20)$$

$$+1+2+1000-1-2-1=\\+10-1121 \quad (21)$$

and Gx and Gy can be estimated as:

$$Gx=121+10-1*I \quad (22)$$

$$Gy=+10-1*I121 \quad (23)$$

Generally, a two-dimensional convolution filter including Sobel filter requires $O(n^2)$ operataions for each output pixel. Separable filter requires $O(n)$ operataions for each output pixel. For example, a 9×9 separable Sobel filter requires 18 operations. OpenCV[16] has implemented extended separable Sobel filter from size 3×3 to 31×31.

We have tested COM edge detector and Sobel edge detector with OpenCV Library using a 2.70Hz CPU. As shown in Fig. 7 (a), proposed algorithm consumes approximately 15 ms invariant to filter size. Sobel filter consumes 6 ms on region size of 3×3 and 46 ms on region size of 31×31. The intersection point of two curves is between filter size of 9×9 and 11×11. Single scale edge detector usually use filter size of 3×3 to 7×7. However, multi-scale edge detector[4-7] can use filter size up to 31×31. Our proposed algorithm provides a good method to speed up multi-scale detection. Edge detection contains thinning step and threholding step. In our time consumption test these steps are not included.

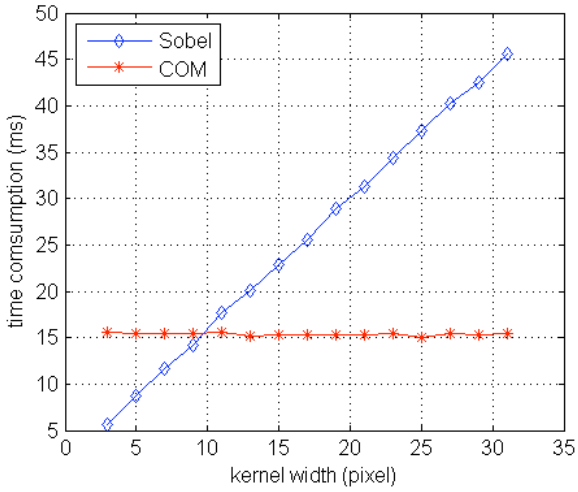
Integral image bring another benefit to multi-scale edge detector regarding efficiency. Multi-scale edge detector needs to compute gradient on several different scales. Because integral image is reusable for different scales, it is only needed to be calculated once. As shown in Fig. 7 (b), we compare the accumulated time consumption between Sobel and COM. The accumulated time consumption of Sobel is calculated as:

$$T=i=wwmaxTi \qquad (24)$$

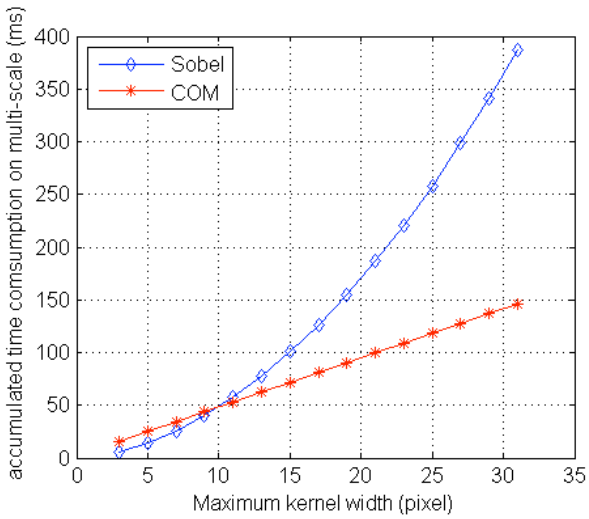
where Ti is time consumption when filter width is w. The accumulated time consumption of COM is calculated as:

$$T=Tintegral+i=wwmaxTi \qquad (25)$$

where $Tintegral$ is integral image building time and Ti is gradient calculation time when region width is w. We can see that the accumulated time consumption of Sobel presents a quadratic increase and COM presents linear increase.



(a)



(b)

Fig. 7. (a) Time consumption of COM and Sobel. Horizontal axis denotes filter width. (b) Accumulated time consumption.

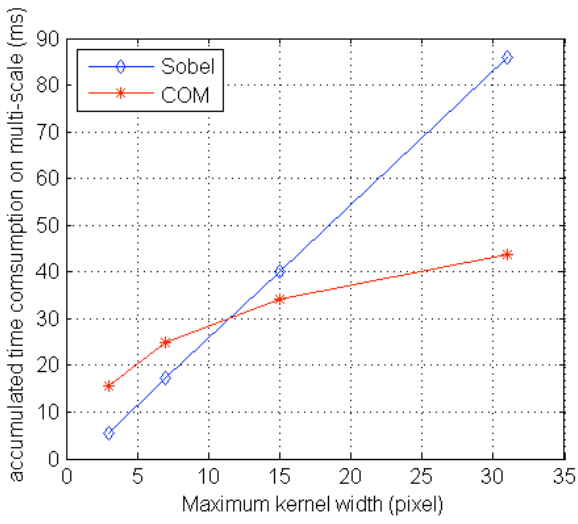
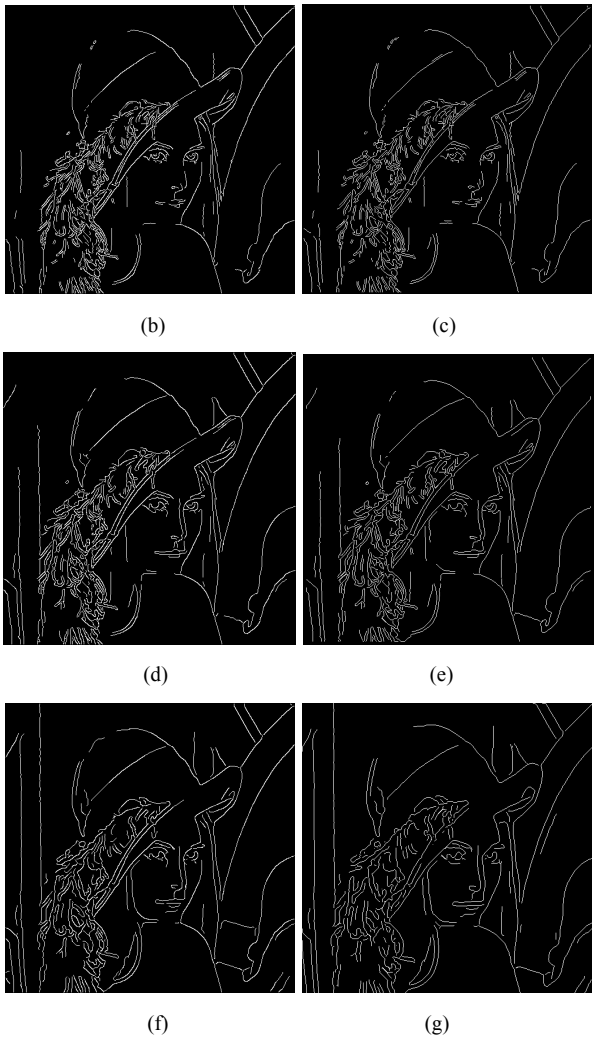


Fig. 8. Accumulated time consumption.

However multi-scale edge detector usually does not process throughout all filter size. Fig. 8 presents a realistic case where the selection of filter size has an interval scale of 2. When using multi-scale filters of scale 3×3, 7×7, 15×15 and 31×31, the accumulated time consumption of COM is 43 ms (i.e., half of the time consumption of Sobel (86 ms)).

6. Edge Detection Result

In this section, we show that COM edge detector provide competitive edge detection result. We choose Canny edge detector as a competitor because Canny edge detector is implemented using Sobel filter in OpenCV[16] and it is always a standard reference comparing various methods in edge detection research. The other reason is that Canny edge detector is the most popular edge detector in real applications because it is fast and easy to be implemented. We aim to develop a fast and concise multi-scale edge detector based on COM in the future. Our experiment shows that COM edge detector presents competitive results. Fig. 9 shows edge detection result of Sobel and COM on different scales. Result shows that proposed method generates very similar results compared with Canny on different scales. Fig. 10 provides supplemental result with different source image.



(b) (c) (d) (e) (f) (g)

Fig. 9. (a) Lena image. (b), (d) and (f): Canny edge detection result with filter size of 3×3, 5×5 and 9×9. (c), (e) and (g): COM edge detection result with filter size of 3×3, 5×5 and 9×9.

7. Conclusions

In this work, we present a fast edge detection method which computes image gradient using COM. The proposed method uses integral image to achieve fast computing. COM edge detection runs with a constant number of operations per pixel independently from its scale. Our experiments show that COM has faster performance than Sobel filter when region size is larger than 9×9. The COM edge detector can be used as a framework for multi-scale edge detection. Our experiment tests accumulated time consumption of multi-scale filters

(a)

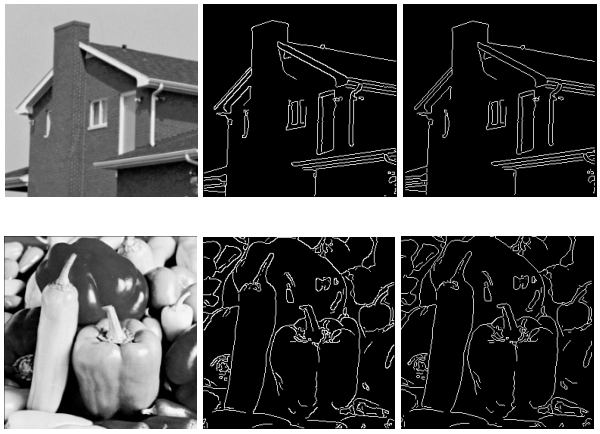


Fig. 10. First column: cameraman, house, pepper. Second column: Canny edge detection result. Third column: COM edge detection result. Both Canny and Sobel here use filter size of 3×3.

with size 3×3, 7×7, 15×15 and 31×31. It shows that COM filter is one time faster than Sobel. The quality of the COM edge detector is competitive to Canny edge detector. COM edge detector is easy to be implemented so it can be flexibly replaced or combined with Sobel filter. In high speed demand task, we can even combine COM filter and Sobel filter to achieve best speed performance. That is to use Sobel for filter size smaller than 9×9 and COM for filter size bigger than 9×9. As part of the future work, we will implement a quantitative evaluation of the COM edge detector and we aim to develop a fast and concise multi-scale edge detector based on COM in the future.

References

(1) Canny, John. "A computational approach to edge detection", *IEEE Trans. Pattern Anal. Mach. Intell.*, PAMI-8(6):679 – 698, 1986.

(2) R. C. Gonzalez and R. E. Woods. "*Digital Image Processing", 3rd ED.* Addison-Wesley, Reading, MA, 2008.

(3) M. Maire, P. Arbelaez, C. Fowlkes, and J. Malik. "Using contours to detect and localize junctions in natural images", CVPR, 2008.

(4) Paul Bao, Lei Zhang, and Xiaolin Wu. "Canny edge detection enhancement by scale multiplication". *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(9):1485 – 1490, 2005.

(5) J. I. Toriwaki J I. Hasegawa, H. Kubota. "Automated construction of image processing procedures by sample-figure presentations". In *Proc. 8th Int. Conf. on Pattern Recogn.*, pages 586 – 588, 1986.

(6) Giuseppe Papari and Nicolai Petkov. "Edge and line oriented contour detection: State of the art", *Image and Vision Computing*, 29(2 – 3):79 – 103, 2011.

(7) D. Ziou and S. Tabbone. "Edge detection techniques: An overview", *Int. J. Pattern Rec. and Image A.*, 8(4):537–559, 1998.

(8) Feynman, Richard Phillips, Robert Benjamin Leighton, and Matthew Linzee Sands. "The Feynman lectures on physics", Mainly mechanics, radiation, and heat. Vol. 1. Basic Books, 2011. P. 19.1 – 19.3.

(9) Crow, Franklin C. "Summed-area tables for texture mapping." ACM SIGGRAPH Computer Graphics. Vol. 18. No. 3. ACM, 1984.

(10) Veksler, Olga. "Fast variable window for stereo correspondence using integral images", Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on. Vol. 1. IEEE, 2003.

(11) Viola, Paul, and Michael J. Jones. "Robust real-time face detection", International journal of computer vision 57.2 (2004): 137-154.

(12) Derpanis K. G. "Integral image-based representations", Department of Computer Science and Engineering York University Paper, 2007, 1(2): 1-6.

(13) Podlozhnyuk V." Image convolution with CUDA". NVIDIA Corporation white paper, June, 2007, 2097(3).

(14) Kroon D. "Numerical optimization of kernel based image derivatives". Short Paper University Twente, 2009.

(15) Pham T. Q., Van Vliet L J. "Separable bilateral filtering for fast video preprocessing", Multimedia and Expo, 2005. ICME 2005. IEEE International Conference on. IEEE, 2005: 4 pp.

(16) G. Bradski, The OpenCV Library, http://docs.opencv.org/doc/tutorials/imgproc/imgtrans/canny_detector/canny_detector.html, accessed 2013-07-03.