CONVOLUTION KERNELS FOR CALCULATING INTENSITY GRADIENTS IN DIGITAL IMAGES DERIVED FROM SAMPLED FIRST-DERIVATIVE GAUSSIAN FUNCTIONS

MATTI KARILUOMA

Abstract. This article demonstrates the use of particular convolution kernels in accurately detecting the edge orientations in a digital image, and shows a generalized method for finding such kernels.

1. Introduction

The problem we will explore concerns the detection of an edge's direction in digital images. We note that the edge of an object in a digital image corresponds to an abrupt change in the intensity of nearby pixels. If we consider a digital image as a two-dimensional signal of intensities, we can instead find the derivative of this signal and search for local maximum of this first-derivative signal; each local maxima will correspond to a point in the image where an abrupt change of intensity occurs. Call this mapping of change of intensities the *image gradient*.

Mapping the change of intensities is typically achieved through the convolution of the image with some function kernel, a *step edge operator*. Canny [2] concerned himself with the derivation of an optimal step edge operator; he found that a first-derivative Gaussian function closely approximates his optimal step edge operator.

However, the first-derivative Gaussian function is asymmetrical, and when extended to two-dimensions, exhibits a directional component. In a two-dimensional image, this is typically dealt with by calculating the gradients in the directions $\theta = 0$ and $\frac{\pi}{2}$, then approximating the image gradient from a combination (see Section 3.1) of these two directional estimates. [1] [3] [5]

This choice of directional kernels used to calculate the image gradient will be the topic of this paper. We will discuss the motivation and criteria for constructing such kernels in directions other than $\theta = 0$ and $\frac{\pi}{2}$, and in particular we will construct kernels in the directions $\theta = \frac{\pi}{8}, \frac{\pi}{4}, \frac{3\pi}{8}, \frac{5\pi}{8}, \frac{3\pi}{4}$, and $\frac{7\pi}{8}$.

Date: May 10, 2010.
1991 *Mathematics Subject Classification.* Primary 94A08; Secondary 68U10.
Key words and phrases. image processing, computer vision, edge detection.

2. Definitions

Pixel A discrete element, p, representing a unit square with coloring.

Pixels of one-dimension are of the form $p = \{p_0\}$ where $p_0 \in \mathbb{N}$ represents the amount of white coloring in the pixel.

Pixels of three-dimensions are of the form $p = \{p_1, p_2, p_3\}$ where $p_1, p_2, p_3 \in \mathbb{N}$ represent the amount of red, green, and blue coloring in the pixel.

Intensity A mapping $\Phi : p \to \mathbb{N}$, defined by (where p is a pixel of n dimensions):

$$\Phi[p] = \left\lfloor \frac{1}{n} \sum_{i=0}^{n} p_i \right\rfloor .$$

Image An image $\mathbf{I}$ is represented by an $m \times n$ matrix of pixels:

$$\mathbf{I} = \begin{bmatrix} p_{0,0} & p_{0,1} & \cdots & p_{0,n} \\ p_{1,0} & p_{1,1} & \cdots & p_{1,n} \\ \cdots & \cdots & \cdots & \cdots \\ p_{m,0} & p_{m,1} & \cdots & p_{m,n} \end{bmatrix}$$

Noise In an image domain, noise is defined as variation from the true intensity of an object in an image resulting from the addition of intensity from sources not the object.

Convolution In an image domain, a convolution of the image $\mathbf{I}$ and a discrete *kernel* function f is denoted by the operator $*$:

$$(I * f)[x, y] = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} f[i, j] I[x - i, y - j].$$

Note : I, f are usually represented as finite matrices, and f is often much smaller than I.

We make convention here that f has an odd number of rows and an odd number of columns, often a square matrix, with $f[0, 0]$ the index of the center element.

We make convention here that any undefined index of f is zero, and that any undefined index of $\mathbf{I}$ takes the value of the nearest, or "border", value.

Intensity Gradient The rate of change of intensity in a given direction at each pixel in the image $\mathbf{I}$.

Each local maxima of an intensity gradient corresponds to a large change in intensity near the pixel in a given direction and often indicate the edge of some object.

3. Background

3.1. **The Process of Edge Detection.** We may consider edge detection to be a three-step process:

- Reduce noise in the image in order to improve the validity of our results.
- Calculate the intensity gradient of the image.

- Search the intensity gradient for local maxima.

Noise Reduction. The image must first have any noise reduced. We expect the distribution of noise throughout the image to follow the normal distribution, therefore we convolve the image with a normalized Gaussian function.

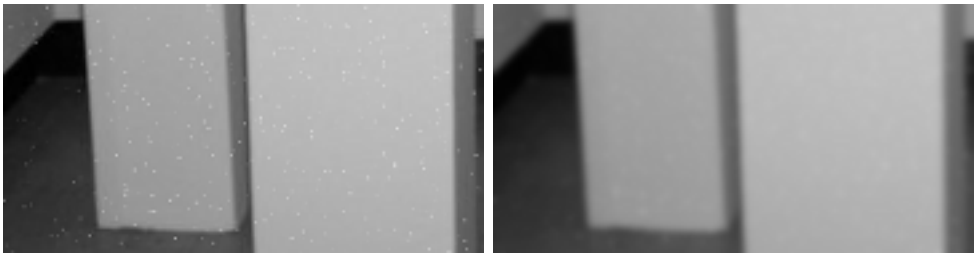
A discretized version of this Gaussian function is typically used to process images. Since the Gaussian function rapidly approaches zero as we move from the origin, we may sample the the Gaussian function at even intervals and truncate the remaining values. Call this sampled, truncated Gaussian function a *discrete Gaussian kernel*, usually represented as a square matrix.

This convolution with a Gaussian kernel reduces the effect that any erroneous pixels will have in the detection of local maxima in the intensity gradient by averaging the intensity of each pixel dependent on the intensity of neighboring pixels.

The standard deviation σ of the Gaussian function determines how quickly the function approaches zero as we leave the origin, and therefore determines the dimensions of the matrix representing our Gaussian kernel.

In Figure 1 we illustrate the convolution with a 3×3 Gaussian kernel with $\sigma = \frac{1}{2}$.

$$\frac{1}{94}\begin{bmatrix} 4 & 11 & 4 \\ 11 & 30 & 11 \\ 4 & 11 & 4 \end{bmatrix} * \mathbf{I} \tag{1}$$

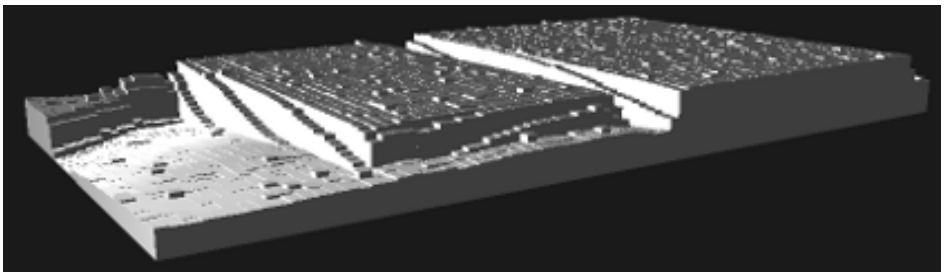


(A) A noisy photograph of two square, white pillars.

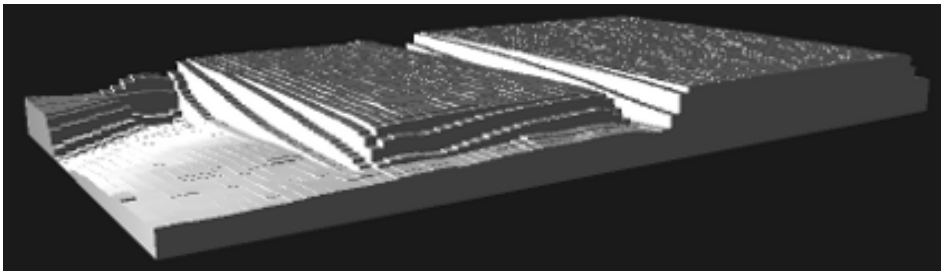
(B) A Gaussian filter with $\sigma = \frac{1}{2}$ convolved with 1a.

FIGURE 1. The extraneous white marks (noise) in Figure 1a are almost completely removed by the convolution with a small Gaussian filter in Figure 1b.

Intensity Gradient. The image gradient ($\mathbf{I}_G$) of our original image $\mathbf{I}$ is typically estimated through the use of a first-order difference operator, such as the Sobel operators [5]:



(A) A 3D representation of Figure 1a.



(B) A 3D representation of Figure 1b.

FIGURE 2. In the above graphs, the z-axis maps to the intensity of the pixels. We can see how the convolution with a Gaussian filter reduces "false" local maxima.

$$\mathbf{I}_{G_x} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * \mathbf{I} \quad \text{and} \quad \mathbf{I}_{G_y} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{I}$$

$$\mathbf{I}_G \simeq \sqrt{\left(\frac{\partial}{\partial x}\mathbf{I}\right)^2 + \left(\frac{\partial}{\partial y}\mathbf{I}\right)^2} \tag{2}$$

$$\simeq \sqrt{(\mathbf{I}_{G_x})^2 + (\mathbf{I}_{G_y})^2} \tag{3}$$

where $\mathbf{I}_{G_x}$ corresponds to a step edge detector with $\theta = \frac{\pi}{2}$ and $\mathbf{I}_{G_y}$ to $\theta = 0$.

We can then estimate the gradient, θ, of a potential edge by considering the values of both $\mathbf{I}_{G_x}$ and $\mathbf{I}_{G_y}$ at each pixel:

$$\theta = \arctan\left(\frac{p_{G_y}}{p_{G_x}}\right) \tag{4}$$

where $p_{G_y} \in \mathbf{I}_{G_y}$ and $p_{G_x} \in \mathbf{I}_{G_x}$ both occupy the same index as $p \in \mathbf{I}$.

This use of a first-order difference operator leads to a great amount of error [3] in the detection of a pixel's gradient, and therefore only allows for the accurate detection of gradients $\theta = \{0, \frac{\pi}{4}, \frac{\pi}{2}, \frac{3\pi}{4}\}$ [1].

Edge Detection. A typical algorithm [1] will search among the local maxima of gradients by using two threshold parameters to determine whether a collection of local maxima is an actual edge or just noise in the image.

Any of the local maxima that are above the high threshold will be considered to correspond to edge pixels, a search is then carried out from these selected pixels in the direction of their gradients.

Each pixel encountered that has the same gradient and is above the low threshold is added to the edge. The search concludes when all local maxima above the high threshold have been inspected.

3.2. **First-Derivative Gaussian Functions.** Kroon [3] has shown that first-order difference operators, such as the Sobel operators (Section 3.1), are unsuitable for accurately detecting the gradient of an edge. Kroon tested various step edge operators, and found that two-dimensional first-derivative Gaussian kernels outperformed other kernels.

Kroon considered two directions of $n \times n$ first-derivative Gaussian kernels for $\theta = 0$ and $\frac{\pi}{2}$, and found that kernels with larger n performed best.

4. Rotated First-Derivative Gaussian Kernels

Without an accurate detection of gradient, the edge detection stage is limited in directions of search. This affects not only the number of edges found, but will also effect whether or not an edge is found as a whole or as a collection of smaller edges.

We will expand the capacity of an edge detection algorithm to accurately detect smaller graduations of angle through the use of kernels constructed using a model of the two-dimensional first-derivative Gaussian function and a coordinate system of unit squares.

Method. Say we have a two-dimensional grid of unit squares on a discrete-valued Cartesian x, y plane. We define a two-dimensional first-derivative Gaussian function on the continuous-valued coordinate system r, s such that $r = -s$ corresponds to a one-dimensional first-derivative Gaussian:

$$g'_\sigma(r,s) := \nabla g_\sigma(r,s) \tag{5}$$

$$= \frac{\partial}{\partial r} g_\sigma(r,s) + \frac{\partial}{\partial s} g_\sigma(r,s) \tag{6}$$

$$= -\frac{r+s}{\sigma^3\sqrt{2\pi}} e^{-\frac{r^2+s^2}{2\sigma^2}} \tag{7}$$

We set our r, s coordinates such that its origin maps to the origin of the x, y plane and the line $r = 0$ is rotated $\theta - \frac{3\pi}{4}$ from the line $x = 0$. We then sample our continuous function on r, s on the discrete x, y plane, represented by a matrix. This matrix is our step edge operator for the chosen θ.

The utility of the two-dimensional first-derivative Gaussian function is two-fold; not only does it follow Canny's [2] optimal step edge detector when $r = -s$, but it also takes into account neighboring pixels when $r \neq -s$.

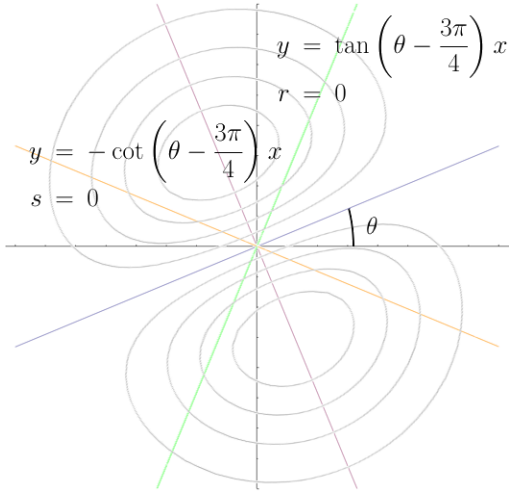


FIGURE 3. A sketch of the r, s plane rotated $\theta - \frac{3\pi}{4}$ from the x, y plane

In order to represent the r, s coordinates in terms of x and y coordinates, we express the lines $r = 0$ and $s = 0$ as

$$r = 0 \quad \Rightarrow \quad y = \tan\left(\theta - \frac{3\pi}{4}\right) x = \cot\left(\frac{\pi}{4} - \theta\right) x, \tag{8}$$

$$s = 0 \quad \Rightarrow \quad y = -\cot\left(\theta - \frac{3\pi}{4}\right) x = -\tan\left(\frac{\pi}{4} - \theta\right) x. \tag{9}$$

We then calculate the shortest straight-line distance from each discrete point of the x, y plane to the lines $s = 0$ and $r = 0$, thereby performing a change of coordinates.

Given a point $P(x, y)$ on the x, y plane we know the shortest straight line distance between P and a line L to be the line that is orthogonal to L and passes through P. Call this orthogonal line L^T. Then it follows that $L \cdot L^T = 0$, where $\cdot$ is the dot product of two vectors.

We can now derive an equation for determining the distance from a point $P(x, y)$ to a point $P_r(x_r, y_r)$ on the line $r = 0$ and $P_s(x_s, y_s)$ on the line $s = 0$:

$$1(x - x_r) + \cot\left(\frac{\pi}{4} - \theta\right)(y - y_r) = 0, \tag{10}$$

$$1(x - x_s) - \tan\left(\frac{\pi}{4} - \theta\right)(y - y_s) = 0 \tag{11}$$

but since we know that the point P_s lies on the line $s = 0$ and P_r lies on $r = 0$ we have:

$$y_r = \cot\left(\frac{\pi}{4} - \theta\right) x_r, \tag{12}$$

$$y_s = -\tan\left(\frac{\pi}{4} - \theta\right) x_s. \tag{13}$$

$$\begin{aligned} 1(x - x_r) + \cot\left(\frac{\pi}{4} - \theta\right)\left(y - \cot\left(\frac{\pi}{4} - \theta\right) x_r\right) &= 0, \\ 1(x - x_s) - \tan\left(\frac{\pi}{4} - \theta\right)\left(y + \tan\left(\frac{\pi}{4} - \theta\right) x_s\right) &= 0 \end{aligned}$$

$$\begin{aligned} x + \cot\left(\frac{\pi}{4} - \theta\right) y &= x_r + \cot^2\left(\frac{\pi}{4} - \theta\right) x_r, \\ x - \tan\left(\frac{\pi}{4} - \theta\right) y &= x_s + \tan^2\left(\frac{\pi}{4} - \theta\right) x_s \end{aligned}$$

$$x_r = \frac{x + \cot\left(\frac{\pi}{4} - \theta\right) y}{1 + \cot^2\left(\frac{\pi}{4} - \theta\right)}, \tag{14}$$

$$x_s = \frac{x - \tan\left(\frac{\pi}{4} - \theta\right) y}{1 + \tan^2\left(\frac{\pi}{4} - \theta\right)}. \tag{15}$$

Then the shortest distance from any point $P(a, b)$ to the line $r = 0$ is given by

$$|r| = \sqrt{(x - x_s)^2 + (y - y_s)^2}, \tag{16}$$

and, from $P(a, b)$ to $s = 0$,

$$|s| = \sqrt{(x - x_r)^2 + (y - y_r)^2}. \tag{17}$$

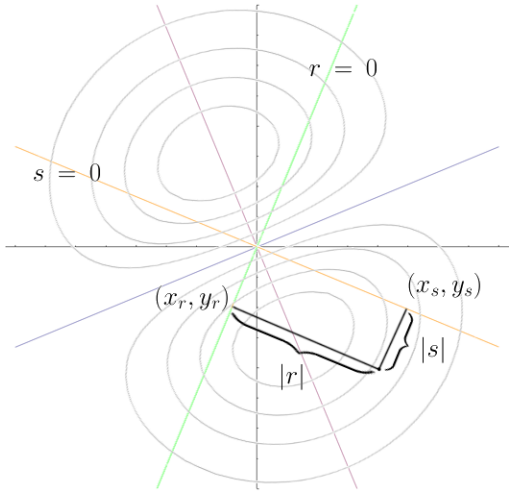


FIGURE 4. The points of intersection on $s = 0$ and $r = 0$, and the distance to them from some point $P(x, y)$.

Having found the magnitudes of r and s, it remains to calculate their sign, and hence which quadrant of the r, s plane we are considering.

$$r = \begin{cases} r & \text{if } y > \cot\left(\frac{\pi}{4} - \theta\right) x \\ -r & \text{otherwise.} \end{cases} \tag{18}$$

$$s = \begin{cases} s & \text{if } y > -\tan\left(\frac{\pi}{4} - \theta\right) x \\ -s & \text{otherwise.} \end{cases} \tag{19}$$

We then evaluate g'_σ at (r, s) and set this value to the index at (x, y) in our matrix representation of our kernel.

4.1. **Example Kernels.**

Example Kernel for Edge Direction $\frac{\pi}{4}$. With $\theta = \frac{\pi}{4}$ we sample the first-derivative Gaussian function on r, s to the x, y plane below and scale the values to whole integers. The value of σ will determine the size of resulting the matrix.

Here we choose $\sigma = \frac{5}{6}$, resulting in a a 5×5 matrix:

$$\frac{1}{562}\begin{bmatrix} 0 & -3 & -8 & -3 & 0 \\ -1 & -129 & -259 & 0 & 1 \\ -8 & -259 & 0 & 259 & 8 \\ -1 & 0 & 259 & 129 & 1 \\ 0 & 3 & 8 & 3 & 0 \end{bmatrix}$$

We can then flip this kernel (by exchanging the first, last rows and also the second, fourth rows) to get an equivalently derived kernel for $\theta = \frac{3\pi}{4}$:

$$\frac{1}{562}\begin{bmatrix} 0 & 3 & 8 & 3 & 0 \\ -1 & 0 & 259 & 129 & 1 \\ -8 & -259 & 0 & 259 & 8 \\ -1 & -129 & -259 & 0 & 1 \\ 0 & -3 & -8 & -3 & 0 \end{bmatrix}$$

Example Kernel for Edge Direction $\frac{\pi}{8}$. For $\theta = \frac{\pi}{8}$ we illustrate a kernel with $\sigma = \frac{2}{5}$, as a 5×5 matrix:

$$\frac{1}{576}\begin{bmatrix} -3 & -27 & -55 & -23 & -2 \\ -17 & -154 & -287 & -109 & -8 \\ -9 & -49 & 0 & 49 & 9 \\ 8 & 109 & 287 & 154 & 17 \\ 2 & 23 & 55 & 27 & 3 \end{bmatrix}$$

We then flip the kernel in order to generate an equivalently derived kernel for $\theta = \frac{7\pi}{8}$:

$$\frac{1}{576}\begin{bmatrix} 2 & 23 & 55 & 27 & 3 \\ 8 & 109 & 287 & 154 & 17 \\ -9 & -49 & 0 & 49 & 9 \\ -17 & -154 & -287 & -109 & -8 \\ -3 & -27 & -55 & -23 & -2 \end{bmatrix}$$

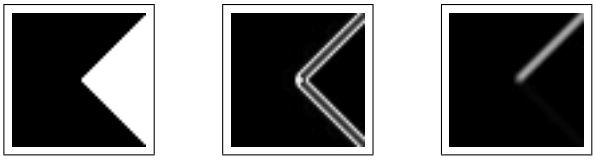
Example Kernel for Edge Direction $\frac{3\pi}{8}$. For $\theta = \frac{3\pi}{8}$ we illustrate a kernel with $\sigma = \frac{2}{5}$, as a 5×5 matrix:

$$\frac{1}{869}\begin{bmatrix} -5 & -32 & -78 & 3 & 2 \\ -43 & -262 & -405 & 104 & 28 \\ -32 & -168 & 0 & 168 & 32 \\ -28 & -104 & 405 & 262 & 43 \\ -2 & -3 & 78 & 32 & 5 \end{bmatrix}$$

We then flip the kernel in order to generate an equivalently derived kernel for $\theta = \frac{5\pi}{8}$:

$$\frac{1}{869}\begin{bmatrix} -2 & -3 & 78 & 32 & 5 \\ -28 & -104 & 405 & 262 & 43 \\ -32 & -168 & 0 & 168 & 32 \\ -43 & -262 & -405 & 104 & 28 \\ -5 & -32 & -78 & 3 & 2 \end{bmatrix}$$

5. Comparison to Other Kernels



(A) (B) (C)

FIGURE 5. A synthetic image with edges of direction $\theta = \frac{\pi}{4}$ and $\frac{3\pi}{4}$ and the result of convolution the Sobel kernel in 5b, and our kernel in 5c.

(A) (B) (C)

FIGURE 6. A synthetic image for $\theta = \frac{3\pi}{8}$ and $\frac{5\pi}{8}$. The convolution with the Sobel kernel (6b), and our kernel (6c).

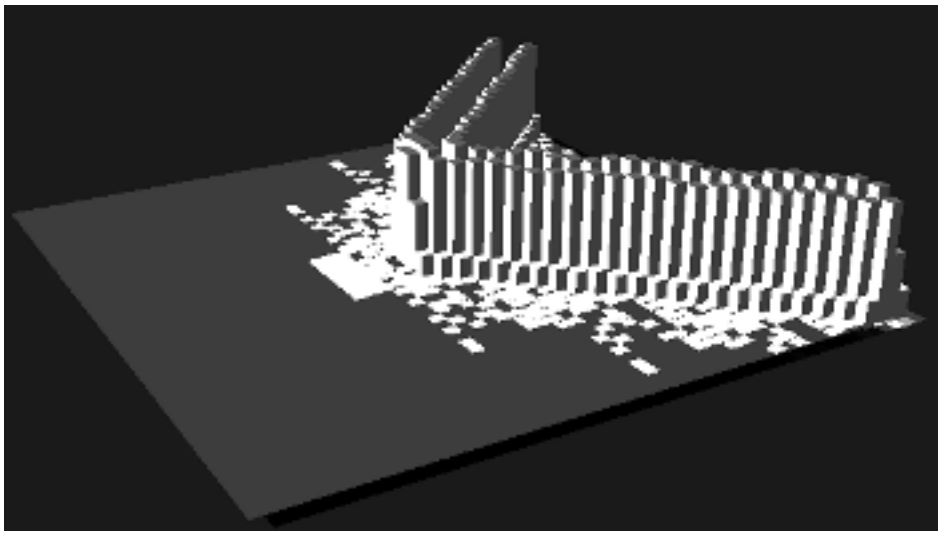


FIGURE 7. A 3D intensity map of 5b

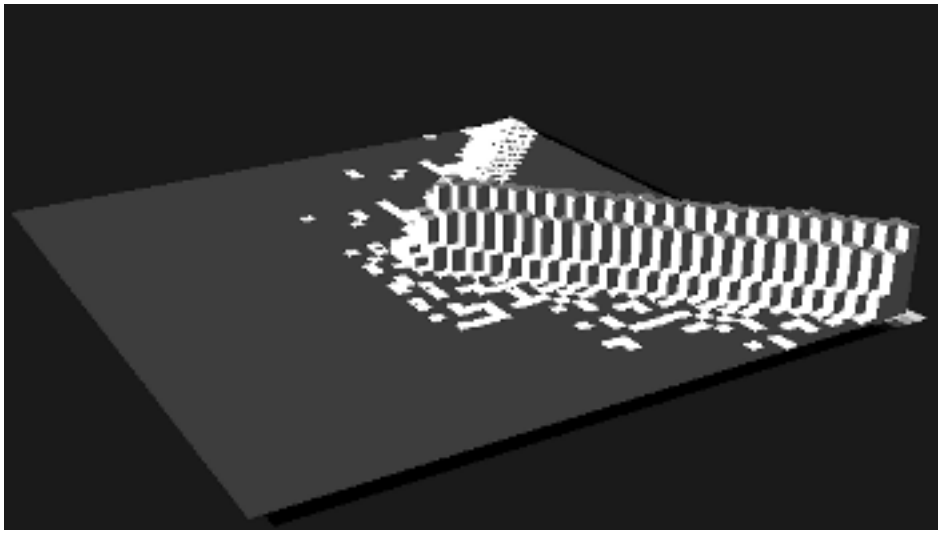


FIGURE 8. A 3D intensity map of 5c

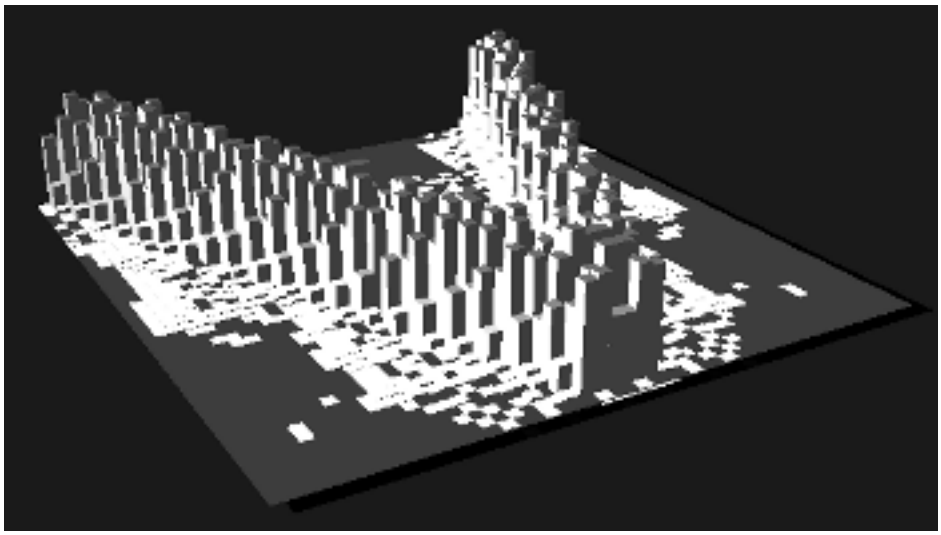


FIGURE 9. A 3D intensity map of 6b

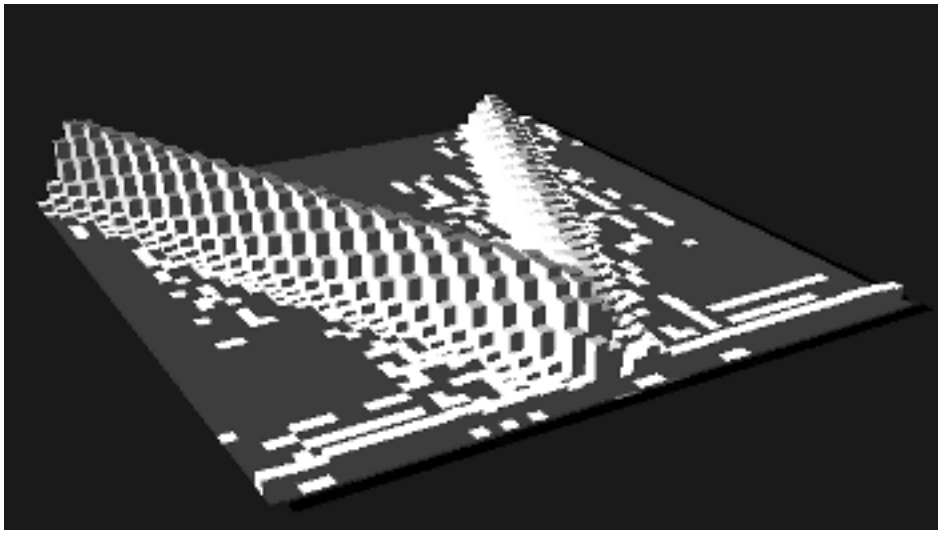


FIGURE 10. A 3D intensity map of 6c

6. DISCUSSION

Summary. Our method has been shown to increase the detection of angular edges in digital images. This increased angular granularity allows us to make better decisions about the composition of the edges in an image.

Further Work. The choice of σ for the first-derivative Gaussian function was loosely guided; others [3] have employed numerical optimization techniques in order to find an optimal σ. No such methods were employed here. Variations in σ affect the relative weights of nearby to further pixels in our kernels, the size of the kernel, and to a certain degree how well the overlying Gaussian function is approximated.

We only considered a simple scheme for sampling the overlaying Gaussian function. Other methods can be considered, including averaging the area of the region near the sample point; another method for a discrete approximation of the Gaussian function, using integral Bessel functions, has been put forth by Lindeberg [4].

For the purposes of algorithmic implementation, the number of different kernels one must apply as well as the manner to interpret their output remains to be explored. Such a decision will no doubt involve a discussion of the number of kernels to consider, and hence the accuracy of the operation, versus the time efficiency of such operations.

References

[1] Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly, Cambridge, MA, 2008.
[2] F. John Canny. A Computational Approach to Edge Detection. 8(6):679–698, 1986.
[3] Dirk-Jan Kroon. Numerical optimization of kernel based image derivatives. Short Paper, University of Twente, 2009.
[4] Tony Lindeberg. Scale-space for discrete signals. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:234–254, 1990.
[5] Irwin Sobel and Gary Feldman. A 3x3 isotropic gradient operator for image processing. Presented at a talk at the Stanford Artificial Project, 1968.

Department of Mathematics, North Dakota State University, Fargo, North Dakota
E-mail address: matti.m.kariluoma@ndsu.edu