EDGE DETECTION USING BAYESIAN PROCESS CONVOLUTIONS

By

Yanda Lang, B.A.

A Project Submitted in Partial Fulfillment of the Requirements

for the Degree of

Master of science
in

Statistics

University of Alaska Fairbanks

May/ 2017

Abstract

This project describes a method for edge detection in images. We develop a Bayesian approach for edge detection, using a process convolution model. Our method has some advantages over the classical edge detector, Sobel operator. In particular, our Bayesian spatial detector works well for rich, but noisy, photos. We first demonstrate our approach with a small simulation study, then with a richer photograph. Finally, we show that the Bayesian edge detector performance gives considerable improvement over the Sobel operator performance for rich photos.

Key words: Bayesian statistics; spatial process convolution; image edge detection; image processing; Markov chain Monte Carlo.

Table of Contents

Table of Figures

Section 1

Introduction

Image edges are the most basic feature of an image. They represent the boundaries of objects within images. Edges contain significant variation between regions. Detecting edges of objects in images is necessary for image processing, image segmentation, feature detection and feature extraction. An edge detection process for brightness or intensity detects the location of edges, which occur where there are sharp changes of intensity. There are many different edge detection methods, and different detectors work better under different circumstances. The performance of a detector is usually evaluated subjectively.

We mention some specific edge detection methods here. The best-known classical algorithm is the Sobel operator. The Sobel operator applies two 3×3 kernels to an image, computing an approximation of the gradient of the image intensity function [1]. Canny [3] proposed a "Laplacian of Gaussian" function method, which causes the algorithm to be slightly more sensitive to weak edges. A Bayesian approach to edge detection was proposed by Santis [7], based on a linear stochastic signal model derived from a physical image description [2]. Research in this area has a lot of diversity; other algorithms rely on mathematical methods [3]-[6], statistical methods [7]-[9], and machine learning methods [10]-[11].

In most of these examples, there are two basic criteria relevant to edge detection performance. One is that edges that occur in the image should not be missed. If they are, this can be described as an edge detection error. The other is that the detection should be well localized; that is, the distance between points marked by the detector and the center of the true edge should be minimized [3]. In other words, if there is an edge in the image, and the detector creates a line that describes this edge, then the thickness of this line should be minimized.

In this project, we will develop a Bayesian spatial model that attempts to perform well under these two criteria in the image intensity space. The basic idea is to apply the Sobel operator to get an approximation of the gradient, perform Bayesian smoothing by using a Bayesian process convolution model, then apply thresholding to identify edges. A Bayesian process convolution model smooths a noisy image and reduces the speckles produced by the Sobel operator. We use

Markov chain Monte Carlo to obtain thresholded samples from the posterior distribution, creating an edge detection image. Several examples of the Bayesian spatial detector's performance on real images will be given.

The rest of the paper is organized as follows. In Section 2, the image gradient, the classical Sobel operator, the data and the concept of image intensity are introduced. In Section 3, a Bayesian spatial convolution model is introduced and examples in one-dimension and two-dimensions are used to illustrate the basic model concept. In Section 4, a Bayesian spatial process convolution model for predicting image edges is developed. A simulation study can be found in Section 5 to evaluate the model's performance. Examples of the Bayesian spatial detector performance on rich images, comparisons with the Sobel operator detector, and Markov chain Monte Carlo results can be found in Section 6. Discussion and future work can be found in Section 7.

Section 2

Image Gradient, Sobel Operator, and Data

2.1 Image Gradient

An image gradient is a directional change in the intensity or color in an image. At each image point ("pixel"), the magnitude of the gradient vector corresponds to the maximum rate of change in some direction. Pixels with large gradient values occur where significant variations occur; these become possible edge pixels. To get an image gradient, the discretized first order derivative of the image in the horizontal and vertical directions is computed.

Define $f(x, y)$ as the intensity of image at location $(x, y), x \in \mathbb{R}, y \in \mathbb{R}$. The gradient function of f is given by

$$\bigtriangledown f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right)$$

where $\partial f/\partial x$ and $\partial f/\partial y$ are the partial derivatives in the x and y directions, respectively. In our image, (x, y) will be the center of the pixel in column x, row y. These center locations are discrete, so we must approximate the gradient, as follows,

$$\frac{\partial f}{\partial x} \approx \mathbf{F}_x * \mathbf{A} \quad \text{and} \quad \frac{\partial f}{\partial y} \approx \mathbf{G}_y * \mathbf{A},$$

where $\mathbf{F}_x = \begin{bmatrix} 1 & 0 & -1 \end{bmatrix}$ and $\mathbf{G}_y = \begin{bmatrix} 1 & 0 & -1 \end{bmatrix}^T$ are two 1-dimensional filters, and $\mathbf{A}$ is an $n \times m$ matrix whose entries provide data observations. Here $*$ denotes the signal processing convolution operation. We provide examples below to illustrate these concepts.

Note that $\mathbf{F}_x * \mathbf{A}$ and $\mathbf{G}_y * \mathbf{A}$ are also $n \times m$ matrices, whose (i, j) entries are given by

$$(\mathbf{F}_x * \mathbf{A})_{ij} = (-1)\mathbf{A}_{i,j-1} + (0)\mathbf{A}_{i,j} + (1)\mathbf{A}_{i,j+1}$$

and

$$(\mathbf{G}_y * \mathbf{A})_{ij} = (-1)\mathbf{A}_{i-1,j} + (0)\mathbf{A}_{i,j} + (1)\mathbf{A}_{i+1,j}$$

At each point, the image gradient approximations can be combined to give the gradient magnitude,

$$|| \bigtriangledown f||_{ij} = \sqrt{[(\mathbf{F}_x * \mathbf{A})_{ij}]^2 + [(\mathbf{G}_y * \mathbf{A})_{ij}]^2}$$

Also, the gradient direction at the (i, j) pixel is given by

$$\tan^{-1} \frac{(\mathbf{G}_y * \mathbf{A})_{ij}}{(\mathbf{F}_x * \mathbf{A})_{ij}}$$

The image gradient has directionality. However, no matter which direction it has, a large gradient value still corresponds to a location where significant variation occurs. As a result, we will only consider the gradient length or magnitude in edge detection.

To illustrate the concepts, we calculate the image gradient for a 4×4 matrix $\mathbf{A}$,

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 2 & 4 \\ 4 & 6 & 0 & 2 \\ 2 & 1 & 3 & 1 \\ 0 & 2 & 1 & 1 \end{bmatrix}$$

For the entry $(2, 2)$ as an example,

$$(\mathbf{F}_x * \mathbf{A})_{22} = \begin{bmatrix} 1 & 0 & -1 \end{bmatrix} * \begin{bmatrix} 4 & 6 & 0 \end{bmatrix} = (-1)4 + (0)6 + (1)0 = -4,$$

$$(\mathbf{G}_y * \mathbf{A})_{22} = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} * \begin{bmatrix} 1 \\ 6 \\ 1 \end{bmatrix} = (-1)1 + (0)6 + (1)1 = 0,$$

therefore

$$|| \bigtriangledown f||_{22} = \sqrt{[(\mathbf{F}_x * \mathbf{A})_{22}]^2 + [(\mathbf{G}_y * \mathbf{A})_{22}]^2} = \sqrt{(-4)^2 + 0^2} = 4.$$

Its gradient matrix is then

$$\begin{bmatrix} - & - & - & - \\ - & 4.00 & 4.12 & - \\ - & 4.12 & 1.00 & - \\ - & - & - & - \end{bmatrix}$$

We note that values for the edge entries of this gradient matrix can be generated by allowing the image to wrap around, but we simply restrict ourselves to the inner entries and work with a $(m - 2) \times (n - 2)$ gradient matrix.

2.2 Sobel Operator Detection

The Sobel operator [1] is another classical algorithm for image edge detection, which is widely used within edge detection algorithms. Instead of using 3×1 filters for horizontal and vertical directions, two 3×3 filters are applied:

$$\frac{\partial f}{\partial x} \approx \mathbf{M}_x * \mathbf{A} \quad \text{and} \quad \frac{\partial f}{\partial y} \approx \mathbf{M}_y * \mathbf{A},$$

where

$$\mathbf{M}_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \quad \text{and} \quad \mathbf{M}_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Both $\mathbf{M}_x * \mathbf{A}$ and $\mathbf{M}_y * \mathbf{A}$ are $n \times m$ matrices with (i, j) entries given by

$$\begin{aligned} (\mathbf{M}_x * \mathbf{A})_{ij} &= (-1)\mathbf{A}_{i-1,j-1} + (0)\mathbf{A}_{i-1,j} + (1)\mathbf{A}_{i-1,j+1} \\ &+ (-2)\mathbf{A}_{i,j-1} + (0)\mathbf{A}_{i,j} + (2)\mathbf{A}_{i,j+1} \\ &+ (-1)\mathbf{A}_{i+1,j-1} + (0)\mathbf{A}_{i+1,j} + (1)\mathbf{A}_{i+1,j+1} \end{aligned}$$

$$\begin{aligned} (\mathbf{M}_y * \mathbf{A})_{ij} &= (-1)\mathbf{A}_{i-1,j-1} + (-2)\mathbf{A}_{i-1,j} + (-1)\mathbf{A}_{i-1,j+1} \\ &+ (0)\mathbf{A}_{i,j-1} + (0)\mathbf{A}_{i,j} + (0)\mathbf{A}_{i,j+1} \\ &+ (1)\mathbf{A}_{i+1,j-1} + (2)\mathbf{A}_{i+1,j} + (1)\mathbf{A}_{i+1,j+1} \end{aligned}$$

The same procedure as in Section 2.1 is followed to calculate the gradients:

$$|| \bigtriangledown f||_{ij} = \sqrt{[(\mathbf{M}_x * \mathbf{A})_{ij}]^2 + [(\mathbf{M}_y * \mathbf{A})_{ij}]^2}$$

Applying the filters on three rows (columns) instead of one row (columns) gives us an average gradient for these three rows (columns), which will make the gradient image less sensitive to noisy data. We will use Sobel operator in our Bayesian spatial edge detection model.

Following is the calculation of the Sobel operator for the 4×4 matrix $\mathbf{A}$ defined earlier. For the $(2, 2)$ element of $\mathbf{A}$ as an example,

$$(\mathbf{M}_x * \mathbf{A})_{22} = \begin{bmatrix} 1 & 1 & 2 \\ 4 & 6 & 0 \\ 2 & 1 & 3 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} =$$
$$(-1)1 + (0)1 + (1)2 + (-2)4 + (0)6 + (2)0 + (-1)2 + (0)1 + (1)3 = -6,$$
$$(\mathbf{M}_y * \mathbf{A})_{22} = \begin{bmatrix} 1 & 1 & 2 \\ 4 & 6 & 0 \\ 2 & 1 & 3 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} =$$
$$(-1)1 + (-2)1 + (1)2 + (0)4 + (0)6 + (0)0 + (-1)2 + (-2)1 + (1)3 = -2,$$

therefore

$$|| \bigtriangledown f||_{22} = \sqrt{[(\mathbf{M}_x * \mathbf{A})_{22}]^2 + [(\mathbf{M}_y * \mathbf{A})_{22}]^2} = \sqrt{(-6)^2 + (-2)^2} = 6.32.$$

Its Sobel operator matrix is then:

$$\begin{bmatrix} - & - & - & - \\ - & 6.32 & 5.09 & - \\ - & 11.05 & 5.83 & - \\ - & - & - & - \end{bmatrix}$$

An example of the Sobel operator applied to a sample intensity image is shown in Figure 1. R code for applying the Sobel operator to an image is given in Appendix I.

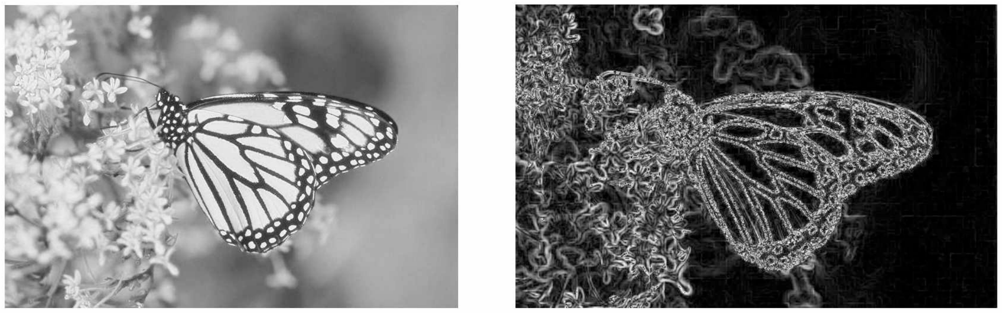

FIGURE 1. Monarch butterfly image 621 by 391 pixels (left). The Sobel operator applied to this image (right).

2.3 Data Description

This project focuses on the intensity data calculated from images. To obtain these data, we download jpeg files from a Public-Domain Test Images website, and the intensity can be calculated using,

$$\text{intensity} = \frac{\max(\text{red},\text{green},\text{blue})}{255}$$

where red, green, and blue are from 0 to 255. The intensity is ranging from pure black (intensity = 0) to pure white (intensity = 1). The image is then treated as an intensity matrix, which is the two-dimensional data used in this project. The original jpeg images are usually 1000 by 1000 pixels. We cannot work with such big images, so we extract small portions of these images, usually 40 by 50 pixels. The R code for calculating an intensity matrix from a raw jpeg file is given in Appendix II.

Section 3

Bayesian Spatial Process Convolution Models

Since the goal of this study is to construct Bayesian spatial models that can detect edges, the noise in images may affect the correctness of detection. In this section, we describe a spatial model that uses process convolutions to smooth an image, whether it's a raw image or a matrix of gradient values that we treat as our image. We will describe these models first for 1-dimensional data, then for 2-dimensional data.

3.1 Bayesian Spatial Process Convolution Model

A spatial process convolution model is a continuous spatial model constructed by convolving knot values with a kernel. A continuous spatial Gaussian process $z(s)$ can be defined using kernel convolutions.

To define our model, we start with evenly spaced spatial locations (knot locations) $\{w_1, w_2, ..., w_J\}$. We specify knot values $\{x_1, x_2, ..., x_J\}$ associated with these knot locations. The knot values $\{x_j\}$ are independent random variables distributed as Normal$(0, \sigma_x^2)$; and $k(d;\lambda)$ is a smoothing kernel function. The kernel function $k(d;\lambda)$ is a nonnegative function. We will use a Gaussian kernel, which can be written as,

$$k(d;\lambda) = c \times \exp\left(-\frac{\lambda}{2}d^2\right), \quad d \geq 0$$

where $\lambda > 0$, and c is $\sqrt{\lambda/2\pi}$ for one-dimensional data, or $\lambda/2\pi$ for two-dimensional data.

The process convolution $z(s)$ at spatial locations is defined by the convolution of the knot values $\{x_j\}$ and the kernel function $k(d;\lambda)$:

$$z(s) = \sum_{j=1}^{J} x_j k(||s - w_j||; \lambda), \quad s \in \mathbb{R}^2$$

where s is an arbitrary spatial location, as in the original paper by Higdon et al. (1998) [12]. Since the x_j's are normally distributed, $z(s)$ is a Gaussian process.

We link observations $\{Y_i\}$ at spatial locations $\{s_i,\ i = 1, 2, ..., n\}$ with the underlying spatial process $z(s)$ through a likelihood function,

$$Y_i | \mathbf{x}, \sigma_x^2, \sigma_y^2 \sim N(z(s_i), \sigma_y^2)$$

where $\mathbf{x} = (x_1, x_2, ..., x_J)$ and $z(s_i) = \sum_{j=1}^{J} x_j k(||s_i - w_j||; \lambda)$.

We assign prior distributions to the remaining parameters, σ_x^2 and σ_y^2. Prior distributions can be denoted as $\pi(\sigma_x^2), \pi(\sigma_y^2)$, and the prior distributions are independent.

To choose a reasonable value of λ, a rule of thumb is,

$$\text{sd(kernel)} \geq 1.6||w_j - w_{j+1}||$$

Then

$$\lambda = \left(\frac{1}{\text{sd(kernel)}}\right)^2$$

After selecting λ in this way, the posterior distribution then becomes,

$$\begin{aligned} p(\sigma_x, \sigma_y, \mathbf{x} \mid data) &\propto L(\sigma_x, \sigma_y, \mathbf{x}) \pi(\sigma_x, \sigma_y, \mathbf{x}) \\ &\propto \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{1}{2\sigma_y^2}(y_i - z(s_i))^2\right) \times \\ &\quad \prod_{j=1}^{J} \frac{1}{\sqrt{2\pi\sigma_x^2}} \exp\left(-\frac{1}{2\sigma_x^2} x_j^2\right) \times \pi(\sigma_x^2) \times \pi(\sigma_y^2) \end{aligned}$$

The R code and *Stan* code to fit this model is given in Appendices III and IV. *Stan* obtains samples from the joint posterior distribution using MCMC (Specifically, Hamiltonian Monte Carlo). Finally, the posterior mean of $z(s_l^*)$ at the gth MCMC iteration can be calculated at spatial locations s_l^* using,

$$z^{(g)}(s_l^*) = \sum_{j=1}^{J} x_j^{(g)} k(||s_l^* - w_j||; \lambda), \quad l = 1, 2, ..., L,$$

where $x_j^{(g)}$ is the value of the jth knot on the gth iteration.

We use $\hat{z}(s_l^*)$ at each pixel location to represent the smoothed image, calculated using,

$$\hat{z}(s_l^*) = \frac{1}{G} \sum_{g=1}^{G} z^{(g)}(s_l^*)$$

This procedure produces a smoothed version of whatever data it starts with - a raw image, or, in our eventual setting, an “image” of magnitudes of the gradient.

3.2 Simple One-Dimensional Example

We first illustrate the basic concept in a 1-dimensional setting. To simulate these data, we first pick J evenly spaced knot locations $\{w_1, ..., w_J\}$; we simulate $\{x_1, ..., x_J\}$ associated with these knot locations, $x_j \overset{ind}{\sim} N(0, 1^2)$; then we calculate $\{z(s_i),\ i = 1, 2, ..., n\}$ where $z(s_i) = \sum_{j=1}^{J} x_j k(||s_i - w_j||; \lambda = 0.4)$. In this example, we simulate $n = 200$ locations, $\{s_i\}$, using a Uniform(0,50) distribution, and each data y_i can be simulated as,

$$Y_i|x_1, ..., x_J \overset{ind}{\sim} N(z(s_i), 0.1^2)$$

These simulated data $\{(s_i, y_i)\}$ are shown in Figure 2. The vertical lines here represent the knot values we used to generate data. We will use these data to predict the posterior distribution of $z(s)$ and compare it with the original $z(s)$.

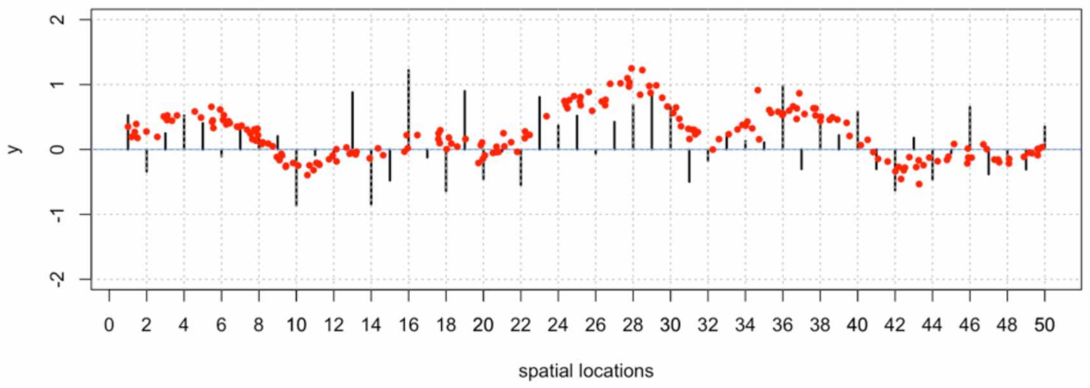

Figure 2. A one-dimensional simulated spatial dataset.

Fit model using these data (using flat priors for σ_x and σ_y) and calculate posterior mean using $\hat{z}(s_l^*)$ where $\{s_1^* = w_1,\ s_2^* = w_2, ...,\ s_J^* = w_J\}$.

The posterior mean, $\hat{z}(s_l^*)$, can be calculated at spatial locations s_l^* using ,

$$\hat{z}(s_l^*) = \sum_{j=1}^{J} x_j k(||s_l^* - w_j||; \lambda), \quad l = 1, 2, ..., L$$

Figure 3 displays a comparison between the prediction using posterior means, $\hat{z}(s_l^*)$, and $z(s_l^*)$ (the true values of z in our simulated dataset). Note that the two curves almost coincide, which means our Bayesian model smooths the noisy data well, and reconstructs the true underlying spatial realization.

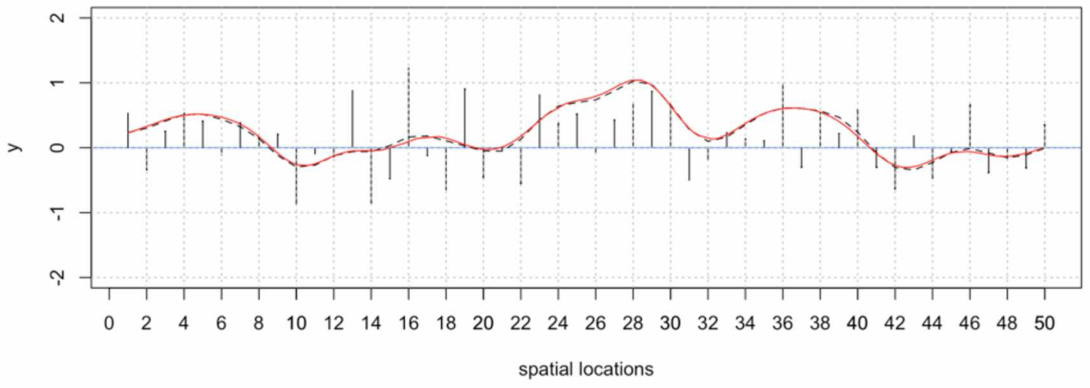

FIGURE 3. Posterior means of predicted values $\hat{z}(s_l^*)$ (solid line) and of $z(s_l^*)$ (dashed line).

3.3 Process Convolution Model in 2D

We used 1-dimensional data to illustrate the concept of Bayesian process convolution model. Now we extend our model to 2-dimensions.

We will use Bayesian process convolution to smooth a raw image, as a precursor to edge detection. For two-dimensional spatial data, we can obtain the intensity data by processing an image as described earlier. For our model, we will use knot locations that are the same as the pixel locations. The model is almost the same as in the 1D process convolution model; the only thing that changes is the distance matrix calculation.

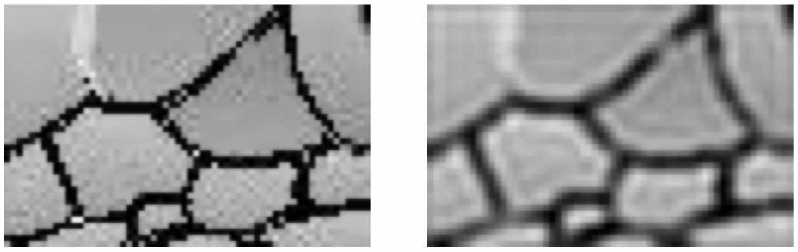

FIGURE 4. Intensity image (left). Posterior mean image (right).

In order to illustrate the process convolutions in 2D, we smooth a small image, 40 by 60 pixels, in the left panel of Figure 4. We used *Stan* to fit model with 2 chains, 500 MCMC iterations. The posterior mean is in the right panel of Figure 4. Comparing the MCMC result with original intensity of this image, our Bayesian model smooths the image well, while leaving out most of the noise. So far we are just illustrating how we can smooth a raw image. In the next section, we will discuss the edge detection.

Section 4

Bayesian Spatial Process Convolution Model for Edge Detection

Edges occur where there is locally a big change in intensity, so we are tempted to look for big changes in the derivative of the intensity. But we are working with 2-dimentional spatial data, so we will look at the magnitudes of the (discretized) gradients of the intensity image. If the magnitude is large, we will conclude there is an edge.

We simulated an image with some noise as an example to illustrate the concept of our edge detection algorithm, as shown in Figure 5, discussed in following paragraphs.

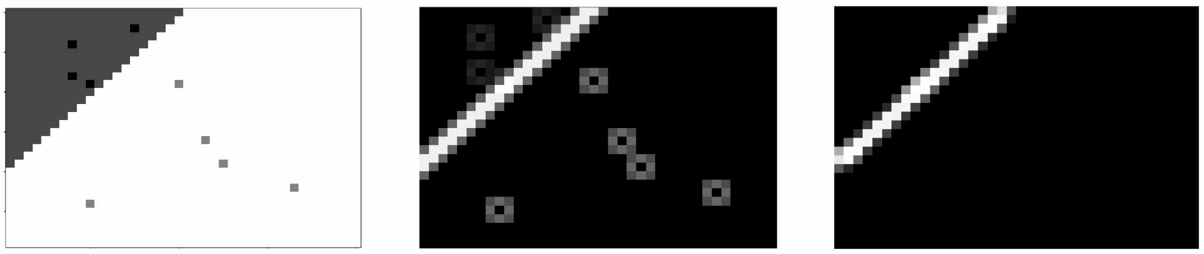
Figure 5. Applying Bayesian edge detection algorithm. Original image (left), after applying the Sobel operator (middle), and posterior probability (right).

Instead of putting raw data (intensity) into the Bayesian model, we start with the gradient values of a raw image. We start by applying the Sobel operater to get the gradient magnitude for each data location, as shown in the middle panel of Figure 5. The edges between black and white in the raw image become a single white line, which indicates edges. Recall that with our color scheme, large values correspond to lighter colors. So white represents the highest possible gradient magnitude. However, after applying the Sobel operator, the noisy data in the raw image changes into some speckles. In order to diminish these speckles, we fit a Bayesian process convolution model using the value of (i, j) pixel in the gradient image, so that they are less likely to be designated as edges.

Recall that in the earlier model in Section 3, Y_i and z_i represented intensity values. Now, Y_i and z_i are magnitudes of the gradient for our edge detection model. We fit a Bayesian process convolution model using these gradient values.

The likelihood function of our model is now given by

$$Y_i|\mathbf{x}, \sigma_x^2, \sigma_y^2 \sim N(z(s_i), \sigma_y^2)$$

where $\mathbf{x} = (x_1, x_2, ..., x_J)$ and $z(s_i) = \sum_{j=1}^{J} x_j k(||s_i - w_j||; \lambda)$.

Prior distributions are assumed for all model parameters,

$$x_j \overset{ind}{\sim} N(0, \sigma_x^2),\ j = 1, 2, ..., J$$

$$\sigma_x \sim \text{Uniform}(0, 2)$$

$$\sigma_y \sim \text{Uniform}(0, 2)$$

As before, the knot locations are the same as the image locations, which are the centers of the pixels. Define the posterior prediction values $z(s)$, and set up a cutoff, c, for $z^*(s)$, so that

$$z^*(s) = \begin{cases} 1 & \text{if } z(s) > c, \\ 0 & \text{o.w.} \end{cases}$$

In this way, we transform all posterior prediction values into either 0 or 1 for each location for each iteration: $z^*(s) = 1$ represents an edge occurring when the gradient is greater than the threshold, and $z^*(s) = 0$ represents the absence of an edge when the gradient is less than the threshold. Then we calculate the posterior probability of an edge at each spatial location,

$$\text{pr}(\{\text{edge at location } s\}) \approx \frac{\sum_{g=1}^{G} z_g^*(s)}{G}$$

Here, G is the number of MCMC iterations.

We display these posterior probabilities in the right panel of Figure 5. Probabilities at or near 1 are close to white. When we apply our Bayesian edge detection algorithm, the speckles that were in the Sobel operator-generated image are gone. In our resulting image, the brighter the color, the more likely there is an edge.

Section 5

Simulation Study

To evaluate the performance of the edge detection algorithms, it is important that the detector has few edge detection errors. The edges occurring in the image should not be missed. We will perform a simulation study in this section, applying our edge detection algorithm to several test images: one without edges, one with sharp edges and one with weak edges. We will then compare the results from our model to those produced by the Sobel operator.

5.1 An Image without Edges

We construct an 30×40 image with pure white (a value of 255) at the left edge, pure black (a value of 0) at the right edge, and which transitions smoothly in intensity as the pixels go from left to right. Such an image is shown in the left panel of Figure 6. There is a linear relationship between the horizontal locations and the intensity values, as shown in right panel of Figure 6. In this case, there are no edges in the image.

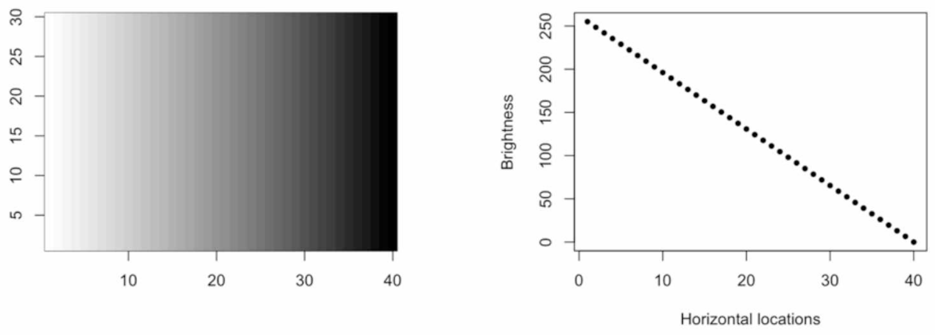

Figure 6. Image from pure white to pure black (left). Its intensity values change linearly with horizontal locations (right).

Since the distance matrix d is a large 1200×1200 matrix, and large matrix calculations are quite time-consuming, we only use 500 iterations with 2 chains for the MCMC. It took 2 hours on a Macbook pro (2.7GHz dual-core Intel Core i5 processor) machine.

As before, we set $\lambda = 0.4$. Figure 7 shows the trace plots for two parameters σ_x and σ_y, for the resulting MCMC output. Note that the traceplots do not show convergence of the MCMC because of the small number of MCMC iterations. However, these plots are acceptable for our purposes.

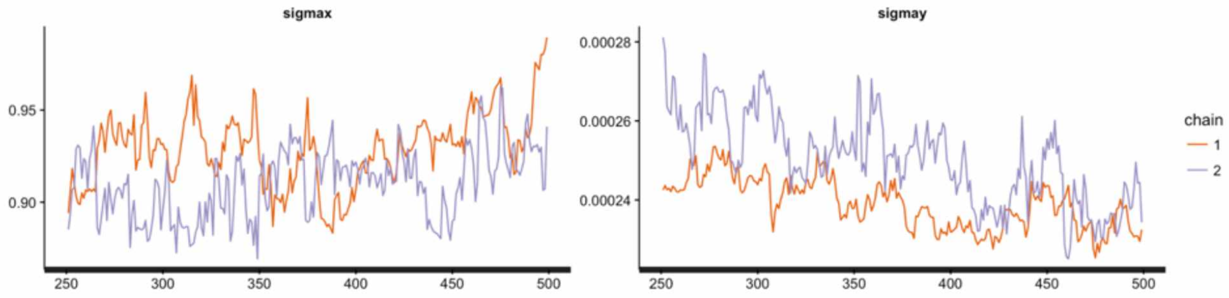

FIGURE 7. Trace plots for MCMC from a simulation image with a smooth intensity change.

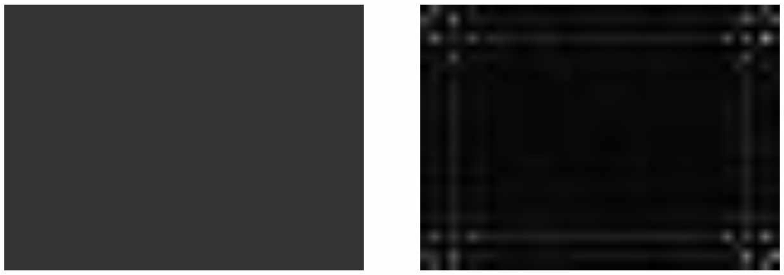

FIGURE 8. Sobel operator (left) and Bayesian method (right) edge detection results from a simulation image with smoothly varying intensity.

Figure 8 presents the edge detection results from the Sobel operator and our Bayesian edge detection model. Both results show that there are no edges in the image, as expected. We note that there is some ghosting (faint speckles) in the corners of the image generated by the Bayesian edge detection method. (This may be edge effects, or simply due to the small number of MCMC iterations.)

5.2 Images with Edges

5.2.1 An Image with A Sharp Edge along The Diagonal

An image with an edge along the diagonal can be constructed using pure white (a value of 255) in the upper left and pure black (a value of 0) in the lower right. A single sharp edge cuts across the image along the diagonal, as shown in the left panel of Figure 9. The image size is 40 by 40 pixels.

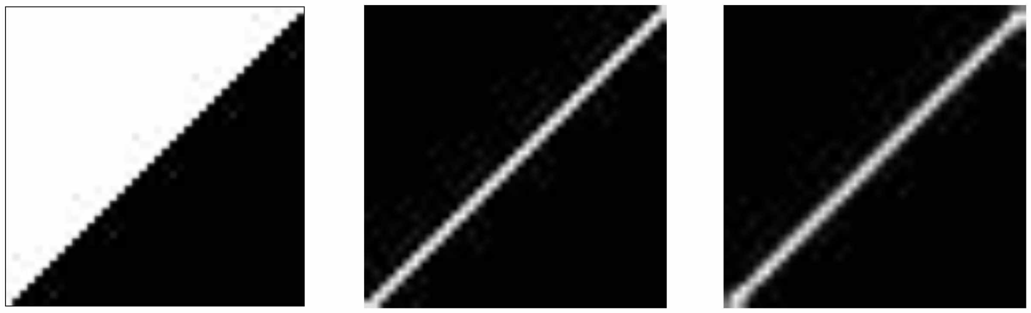

FIGURE 9. An image with a sharp edge along the diagonal (left). Sobel operator detection results (middle) and Bayesian edge detection results (right) are shown.

We again used 500 iterations with 2 chains for the MCMC. Here, we set a cutoff value of $c = 0.95$. Figure 10 shows the trace plots for two parameters σ_x and σ_y for the Bayesian edge dection method. The overlapping trajectories indicate the convergence of each chain this time.

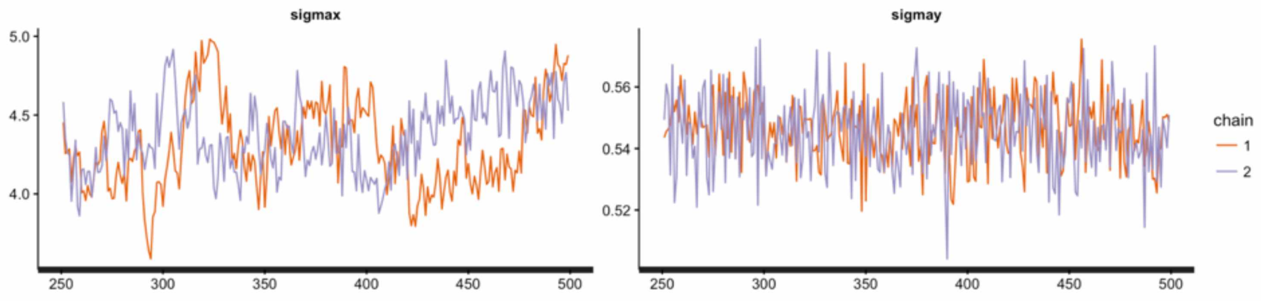

FIGURE 10. Trace plots for Bayesian method from an image with a sharp edge along the diagonal.

Results are shown in the middle and right panels of Figure 9; both plots indicate that an edge exist on the diagonal of the original image. The gray in the Sobel operator detection results represent the magnitude of the gradient; while the gray in the Bayesian edge detection results represent the probability of an edge at that spatial location.

5.2.2 An Image with A Weak Edge along The Diagonal

An image with a weak edge along the diagonal was constructed using pure white (a value of 255) in the upper left and light gray (a value of 190) in the lower right. A single weak edge cuts across the image along the diagonal, as shown in the left panel of Figure 11. The image size is 40 by 40 pixels.

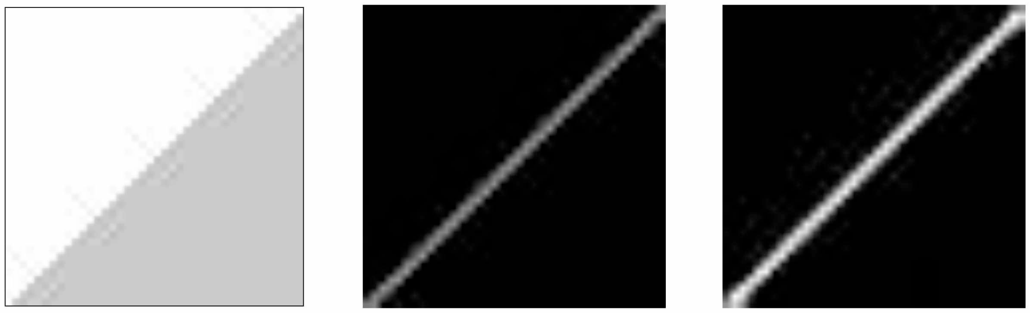

FIGURE 11. An image with a weak edge along the diagonal (left). Sobel operator detection results (middle) and Bayesian edge detection results (right) are shown.

We again use 500 iterations with 2 chains for the MCMC, and we again use a 0.95 cutoff value. Each of the edge detection methods does a satisfactory job and detects the edge on the diagonal from the original image, as shown in the middle and right panels of Figure 11. The Bayesian edge detection method gives a brighter edge.

Section 6

Rich Photos

In this section, we explore the performance of the edge detection algorithms using some richer photos containing more details. We chose a rich photo with sharp edges, one with weak edges in a bright image, and one with narrowly spaced edges.

6.1 A Rich Photo with Sharp Edges

As an example, we select a portion of an image that is 40 by 50 pixels from the image *Frymire* [17], and calculate its intensity. The original intensity image is shown in the left panel of Figure 12. The edges look sharp; and the images are quite clean, but they are not as clean as the simulation images; there is some noise in the white area.

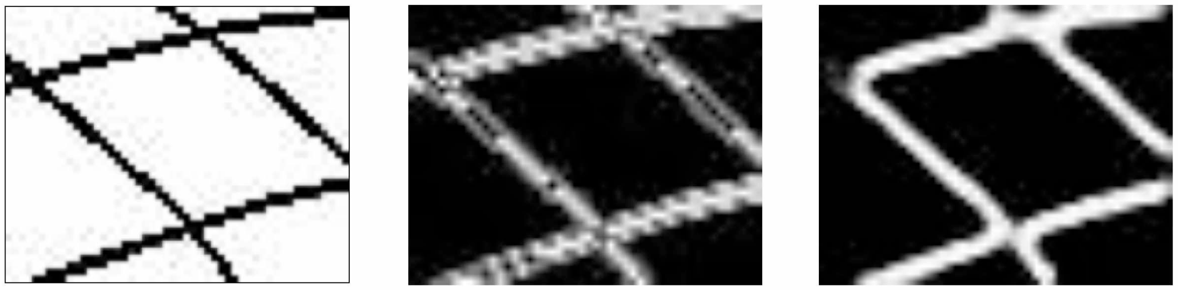

Figure 12. A rich photo with sharp edges (left). Sobel operator detection results (middle) and Bayesian edge detection results (right) are shown.

We ran 2 chains using 500 iterations for the MCMC. The traceplots for σ_x and σ_y are somewhat problematic, but still deemed satisfactory for our purposes. Here, we set a 0.80 cutoff value.

Both models detect edges well, as shown in the middle and right panels of Figure 12. The speckling in the original image is completely gone in the Sobel operator image and the Bayesian model output. However, the Sobel operator gives us a blurry edge image with unusual artifacts along the diagonal line segments. The Bayesian model gives clean edges without any artifacts. As we compare these rich photo results with the Sobel operator results from clean images in the

Section 5, the noise from the rich photos affect the Sobel operator detector greatly. In this case, the Bayesian model detects the edges really well, and it gives us an accurate depiction of the edges.

6.2 A Bright Photo with Weak Edges

We select a portion of a bright image that is 40 by 50 pixels from the image *Arctichare* [17], and calculate its intensity matrix. The original intensity image is shown in the left panel of Figure 13. The image is bright; faint edges occur between the light gray and the white.

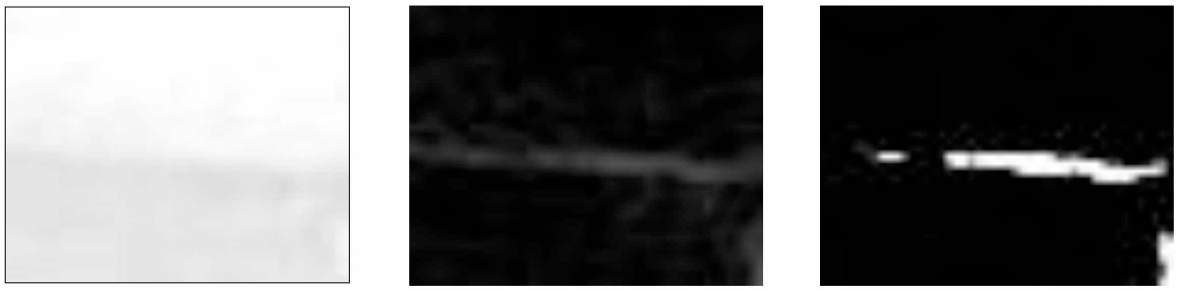

FIGURE 13. A bright image with edges (left). Sobel operator detection results (middle) and Bayesian edge detection results (right) are shown.

We ran 2 chains using 500 iterations for the MCMC. Trace plots (omitted) indicate convergence nicely this time. We set a 0.95 cutoff value.

Results from two methods are shown in the middle and right panels of Figure 13, the result from the Sobel operator detection can barely be considered as an edge. The Bayesian method detected the faint edge better in this case. The locations are accurate, but the Bayesian method makes the edge appear thicker than the actual edge.

6.3 A Rich Photo with Narrowly Spaced Edges

We select another piece of a high-contrast image with 40 by 50 pixels from the image *Frymire* [17] again, and calculate its intensity matrix, as shown in the left panel of Figure 14. The edges in the image are narrowly spaced. Edges occur between the gray and the black.

We fit our Bayesian model with 500 iterations with 2 chains for the MCMC. Trace plots (omitted) indicate the MCMC has converged. Since there are many edges in a same image, we choose a smaller threshold value 0.65 for this image.

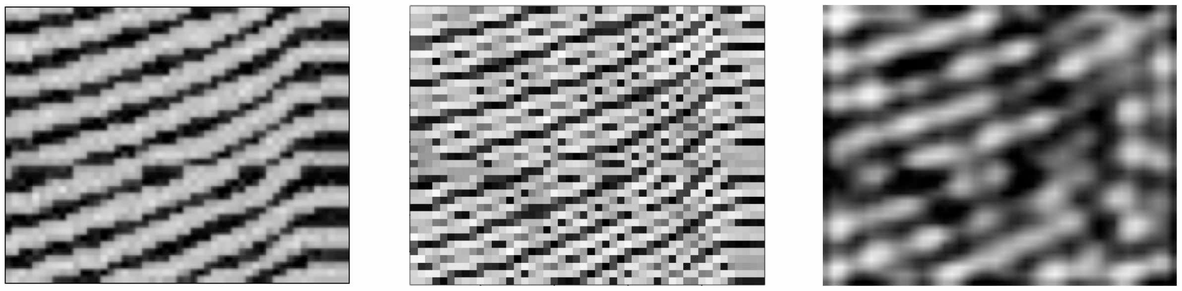

FIGURE 14. An image with narrow band of brightness (left). Sobel operator detection results (middle) and Bayesian edge detection results (right) are shown.

Results from both methods are shown in the middle and right panels of Figure 14. The locations of the edges in the Sobel operator image are not correct. The Bayesian method can detect edges well, but it gives us a very blurry image.

Section 7

Discussion and Future Work

7.1 Discussion

We have described a Bayesian process convolution method for edge detection. One advantage of the Bayesian approach include ease of interpretation. The Bayesian edge detection results represent the probability of an edge at that spatial location; while the Sobel operator detection results represent the magnitude of the gradient. The Bayesian process convolution model gives us a reliable interpretation.

We have considered two models: a Bayesian process convolution model and a Sobel operator model. It is necessary to discuss the two criteria mentioned in the introductory section: few edge detection errors and the localized edges, in order to capture the intuition of good edge detection.

Different detectors work better under different circumstances. Specifically, the Sobel operator works better when the image is totally clean. Even though the Sobel operator usually satisfies both of the two criteria, the noise in actual photos affects the performance of the Sobel operator greatly. In addition, the Sobel operator does not work well if the intensity changes quickly, as when there are edges occurring every few pixels. Thus, the Sobel operator detector does not always work well for rich photos with very fine details.

The Bayesian process convolution model can work well for noisy images. The location of edges is usually accurate, but the result edges look thick and blurry most of the time. This method satisfies the first criterion; it has few edge detection errors. However, it does not satisfy the second criterion, the localized edges, because of the thick edges.

7.2 Future Work

The Sobel operator is one of the classical detectors. However, it is sensitive to noise, so we can use some less sensitive edge detectors to perform the gradient calculation within our Bayesian spatial model. We can apply 1×3 and 3×1 filters, or we can apply an improved Sobel operator with two 5×5 filters.

Moreover, we only worked on the intensity of images, which are grayscale images. We can consider the other two components, hue and saturation, from the HSB (hue/saturation/brightness) color model in order to work on color images.

Last but not least, because fitting the Bayesian spatial model is extremely time-consuming, we used a small number of MCMC iterations, and sometimes the MCMC did not converge (although it was still marginally acceptable). We tried our best to improve the code efficiency, and avoid loops as much as possible. But, for a small portion of an image, 40×50 pixels, our improved coding still takes at least two hours to run only 2 chains and 500 iterations in *Stan*. If we can improve the coding efficiency more and give the MCMC more time to run, or if we try to use fewer knot locations (although this will make the edges thicker), we can use more MCMC iterations to improve our results.

Acknowledgement

I would like to thank my committee chair, Margaret Short, PhD, of University of Alaska Fairbanks for a large amount of help for this project, as well as my committee members, Ron Barry, Scott Goddard, Julie McIntyre for coding and course work. The images in this project can be downloaded from [17].

Appendix

Below is a part of R code and *Stan* code that produces the results in the project.

I. Sobel Operator

```
> img.bri.gradient = matrix(NA,nrow=knot.y,ncol=knot.x)
> temp.img = img.bri
> img.row = nrow(temp.img)-1
> img.col = ncol(temp.img)-1
> for (i in 2:img.row) {
+   for (j in 2:img.col) {
+     matrix_x = matrix(c(-1,-2,-1,0,0,0,1,2,1), nrow=3, ncol=3)
+     matrix_y = matrix(c(-1,0,1,-2,0,2,-1,0,1), nrow=3, ncol=3)
+     matrix_a = matrix(c(temp.img[i-1,j-1],temp.img[i,j-1],temp.img[i+1,j-1],
      temp.img[i-1,j],temp.img[i,j],temp.img[i+1,j],temp.img[i-1,j+1],
      temp.img[i,j+1],temp.img[i+1,j+1]), nrow=3, ncol=3)
+     gx = matrix_x[1,1]*matrix_a[3,3]+matrix_x[1,2]*matrix_a[3,2]+
      matrix_x[1,3]*matrix_a[3,1]+matrix_x[2,1]*matrix_a[2,3]+
      matrix_x[2,2]*matrix_a[2,2]+matrix_x[2,3]*matrix_a[2,1]+
      matrix_x[3,1]*matrix_a[1,3]+matrix_x[3,2]*matrix_a[1,2]+
      matrix_x[3,3]*matrix_a[1,1]
+     gy = matrix_y[1,1]*matrix_a[3,3]+matrix_y[1,2]*matrix_a[3,2]+
      matrix_y[1,3]*matrix_a[3,1]+matrix_y[2,1]*matrix_a[2,3]+
      matrix_y[2,2]*matrix_a[2,2]+matrix_y[2,3]*matrix_a[2,1]+
      matrix_y[3,1]*matrix_a[1,3]+matrix_y[3,2]*matrix_a[1,2]+
      matrix_y[3,3]*matrix_a[1,1]
+     mag = sqrt(gx^2+gy^2)
+     img.bri.gradient[i,j] = mag
+   }
+ }
```

II. Image Intensity

```
> imgbri = function(x){
+   img255 = 255*x
+   n = length(img255[,1,1])
+   m = length(img255[1,,1])
+   bri.x = matrix(NA, nrow=n, ncol=m)
+   for (i in 1:m) {
+     bri.x[,i] = RGB2HSV(img255[,i,])[,3]
+   }
+   return(bri.x)
+ }
```

III. Stan Codel for Bayesian Process Convolution

```
data{
      int<lower=1> N;
      int<lower=1> J;
      real y[N];
      matrix[N,J] K;
}

parameters{
      real<lower=0> sigmax;
      real<lower=0> sigmay;
      vector[J] x;
}
```

```
model{
      vector[N] kx;
      matrix[N,J] K;
      sigmax ~ uniform(0,2);
      x ~ normal(0,sigmax);
      sigmay ~ uniform(0,2);
      kx = K * x;
      y ~ normal(kx,sigmay);
}
```

IV. R Codel for Bayesian Process Convolution

```
> knot.x = ncol(img.bri.gradient)
> knot.y = nrow(img.bri.gradient)
> J = knot.x*knot.y
> N = J
> y = c(img.bri.gradient)
> d = matrix(NA,nrow=N,ncol=J)
> for(i in 1:nrow(knot.location)){
+   dt = knot.location[i,]
+   d[i,] = apply(data.location, 1, function(x){dist(rbind(as.numeric(x),as.numeric(dt)))})
+ }
> distsq  = d^2
> K = 0.4/6.2832*exp(-0.4*distsq/2)
> iter.times = 500
> output = "mcmc_bri"
> iter.times = 500
> myfit.bri = stan(file="2-D convolution.stan",data=c("N","J","y","K"),
  iter=iter.times,chains=2,sample_file=output,cores=4)
```

V. Predicted Values

```
> chain1.bri = read.csv("mcmc_bri_chain1.csv")
> chain2.bri = read.csv("mcmc_bri_chain2.csv")
> post.bri.x = rbind(chain1.bri,chain2.bri)
> rr = iter.times*2
> post.bri.mu = matrix(NA,nrow=rr,ncol=N)
> for(i in 1:rr){
+   post.bri.mu[i,] = t(K%*%t(post.bri.x[i,]))
+ }
```

VI. Bayesian Edge Detection

```
> cutoff = quantile(post.bri.mu,0.95)
> post.bri.zstar = (post.bri.mu>cutoff)*1
> post.prob.bri.vector = matrix(NA,nrow=1,ncol=N)
> for(i in 1:N){
+   post.prob.bri.vector[,i] = sum(post.bri.zstar[,i])/(2*iter.times)
+ }
> post.prob.bri = matrix(post.prob.bri.vector,nrow=knot.y,ncol=knot.x)
```

References

[1] I. Sobel, G. Feldman, “A 3x3 Isotropic Gradient Operator for Image Processing,” presented at the Stanford Artificial Intelligence Project (SAIL) in 2015 (1968).

[2] V. Rani, “A Study of Edge Detection Methods,” International Journal of Science, Engineering and Technology Research, Volume 1, Issue 6, 2012.

[3] J. Canny, “A Computational Approach to Edge Detection, IEEE Transactions On Pattern Analysis and Machine Intelligence, VOL. PAMI-8, NO.6, 1984

[4] D. Marr, E. Hildreth, “Theory of Edge Detection,” Proceedings of the Royal Society of London B: Biological Sciences 207.1167 (1980): 187-217.

[5] A. Yuille, T. Poggio, “Scaling Theorems for Zero Crossings,” IEEE Transactions on Pattern Analysis and Machine Intelligence 1 (1986): 15-25.

[6] P. Perona, J. Malik. “Scale-space and Edge Detection Using Anisotropic Diffusion,” IEEE Transactions on Pattern Analysis and Machine Intelligence 12.7 (1990): 629-639.

[7] A. Santis, C. Sinisgalli, “A Bayesian approach to edge detection in noisy images,” in IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, vol. 46, no. 6, pp. 686-699, Jun 1999.

[8] S. Konishi, et al. “Statistical Edge Detection: Learning and Evaluating Edge Cues,” IEEE Transactions on Pattern Analysis and Machine Intelligence 25.1 (2003): 57-74.

[9] J. Bezdek, R. Chandrasekhar, Y. Attikouzel. “A Geometric Approach to Edge Detection,” IEEE Transactions on Fuzzy Systems 6.1 (1998): 52-75.

[10] C. Lin, C. Lee, “Neural-fuzzy Systems: A Neuro-fuzzy Synergism to Intelligent Systems,” Prentice-Hall, Inc., 1996.

[11] J. Zhang, Y. Chen, X. Huang, “Edge detection of images based on improved Sobel operator and genetic algorithms, in 2009 International Conference on Image Analysis and Signal Processing, Taizhou, 2009, pp. 31-35.

[12] D. Higdon, “Space and Space-Time Modeling using Process Convolutions, in Anderson C.W., Barnett V., Chatwin P.C., El-Shaarawi A.H. (eds) Quantitative Methods for Current Environmental Issues. Springer, London, 1998

[13] A. Hanbury, “Circular Statistics Applied to Colour Images, in 8th Computer Vision Winter Workshop,Austria, 2003

[14] A. Gelman, J. Carlin, H. Stern, D. Dunson, A. Vehtari, and D. Rubin, “Bayesian Data Analysis, 3rd edition. CRC Press. 2014.

[15] Stan Development Team, “Stan Modeling Language User’s Guide and Reference Manual, Version 2.12.0, 2016

[16] H. Wickham, “Advanced R, Chapman and Hall/CRC, 2014

[17] Website: `https://homepages.cae.wisc.edu/~ece533/images/`