ABSTRACT

Title of dissertation: THREE DIMENSIONAL EDGE DETECTION USING WAVELET AND SHEARLET ANALYSIS

David A. Schug, Doctor of Philosophy, 2012

Dissertation directed by: Professor Dianne P. O'Leary
Dr. Glenn R. Easley
Applied Mathematics and Scientific and Statistical Computing Program

Edge detection determines the boundary of objects in an image. A sequence of images records a 2D representation of a scene changing over time, giving 3D data. New 3D edge detectors, particularly ones we developed using shearlets and hybrid shearlet-Canny algorithms, identify edges of complicated objects much more reliably than standard approaches, especially under high noise conditions. We also use edge information to track the position and velocity of objects using an optimization algorithm.

THREE DIMENSIONAL EDGE DETECTION USING WAVELET AND SHEARLET ANALYSIS

by

David Albert Schug

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2012

Advisory Committee:
Professor Dianne P. O'Leary, Co-Advisor
Dr. Glenn R. Easley, Co-Advisor
Professor John J. Benedetto
Professor William S. Levine
Professor David W. Jacobs

Acknowledgments

I owe my gratitude to all the those who have made this thesis possible and because of whom my graduate experience has been one that I will cherish forever.

First, I'd like to thank my co-advisor, Professor Dianne O'Leary for giving me an invaluable opportunity to work on challenging and extremely interesting projects over the years. It has been a pleasure to work with and learn from such an extraordinary individual.

I would also like to thank my co-advisor, Dr. Glenn Easley. Without his extraordinary theoretical ideas and computational expertise, this thesis would have been a distant dream.

Thanks are due to Professor John Benedetto, Professor David Jacobs, and Professor William Levine for agreeing to serve on my thesis committee and for sparing their invaluable time reviewing the manuscript.

I would like to acknowledge financial support from the Patuxent River Naval Air Station, in particular the Photogrammetrics group for all the projects discussed herein.

I owe my sincere gratitude to my wife who has stood by me and supported me through my career.

Lastly, deepest thanks go to the Lord Jesus Christ, Glory to God!

Table of Contents

List of Figures

Chapter 1

The Edge Detection Problem

1.1 Introduction

The edge detection problem is about discerning differences between objects in images. This problem is not new but it is very important and not completely well defined. Humans use our eyes to capture visual information and then pass that information to our minds to make sense of it. We open our eyes and clearly see differences between important objects and other background and clutter in a scene. As humans we can visually interpret many things but there are open questions about how we do it. The exact science and mathematics of how to process and organize image data is not clear. However applying mathematics to represent images intensities and compute the differences is a step in the right direction. At the heart of edge detection is the necessity to first compute the differences in image intensities. When the edge differences are combined we actually see the boundary of a silhouette of the three dimensional object being observed. The boundary could be curved or straight but in any case there is an implied direction for each edge pixel. If we consider a group of edge pixels as a batch, the direction can remain constant or change rapidly as our eye follows the boundary from pixel to pixel. The instantaneous snapshot gives us the outline of the object but what came before or after is somewhat of a mystery. Traditionally the two dimensional image itself

has been the focus of edge detection algorithms. This may be due to the fact that if we can't see features in two dimensions how can we visualize a complicated structured surface in three dimensions? Capturing image sequences and processing the volumetric data will provide more information for edge analysis. This work reviews the background of two dimensional edge detection and develops and applies three dimensional edge analysis.

Chapter 2 reviews basic two dimensional (2D) edge detection methods including Sobel, Canny, wavelet, and shearlet. For each method the necessary mathematical model is defined and integrated into its respective algorithm. The methods are also compared. Chapter 3 extends the traditional 2D edge detection approaches into three dimensions. The 3D edge detection methods developed include the 3D Canny, 3D wavelet, and the new 3D shearlet. The 3D methods are tested using a noisy spherical harmonic solid and a noisy solid spiraling disk. Also two hybrid methods are developed and implemented. The first hybrid method uses the 2D wavelet to process an image slice and then applies the Canny method along the time dimension. The second hybrid method is similar except it processes the image slice with a 2D shearlet edge detector. Chapter 4 describes the integration of 2D and 3D edge detection into a tracking application. The tracking algorithm is defined and tested using different shapes, noise levels, and illumination effects.

My research contributions are summarized by the following.

- Implement my own versions of 3D Canny and 3D Wavelet edge detectors.
- Design and implement new 3D Shearlet edge detectors and hybrid Canny-

wavelet and Canny-shearlet edge detectors.

- Develop test cases with corresponding metrics and analysis comparing 2D and 3D methods.

- Design and implement new tracking algorithms that use 2D and 3D Canny, wavelet, shearlet, and hybrid edge detectors.

- Develop test cases and provide corresponding analysis of tracking capabilities.

- Use 3D edge detection to compute object velocity.

Chapter 2

Two Dimensional Edge Detection

2.1 Introduction

Edge detection is the process of distinguishing the boundaries of different objects in an image. For two dimensional (2D) edge detection the input data is a single image. The image data is generated by a given intensity function

$$I : \mathbb{R}^2 \to \mathbb{R} \tag{2.1}$$

that assigns scalar valued intensities to discrete locations in the plane. The measured image $\mathbf{I}$ is an $m \times n$ array of samples of the intensity function I at evenly spaced grid points $\mathbf{P} = [1:m] \times [1:n]$. Each sample in the image is called a *pixel* or picture element consisting of an image intensity at a particular location in the plane. Given a threshold $h > 0$ the output of the edge detection process includes a set of points $\mathbf{P_E}$ and an image $\mathbf{I_E}$ defined by

$$\mathbf{I_E}(i,j) = \begin{cases} 1 & \text{if } (i,j) \in \mathbf{P}_E,\ i = 1,\ldots,m \text{ and } j = 1,\ldots,n \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

where

$$\mathbf{P}_E = \{\mathbf{p} \in \mathbf{P} : |\nabla I(\mathbf{p})| \geq h, \}\,. \tag{2.3}$$

The image $\mathbf{I}$ of a white ball on a black background is the disk on the left in Figure 2.1 and the corresponding edge image $\mathbf{I_E}$ is on the right.

Figure 2.1: Image $\mathbf{I}$ of solid disk (left) and its edge image $\mathbf{I_E}$ (right).

The gradient $\nabla\mathbf{I}(\mathbf{p})$ measures the change in the magnitude of image intensity from one pixel to another. However the gradient also reveals the direction of the intensity change. Therefore, it is important to consider both magnitude and direction when identifying an edge. In addition, we need to choose the threshold h so that the gradient identifies the true edge.

The process of finding edges is more complicated than it first appears. First, we need to approximate $|\nabla I|$, since we only have discrete samples of $\mathbf{I}$. Second, if the original object is simple, intensities are uniform on the interior of an object and dramatically different from the intensity of the background, as is the case with the white disk. However, if noise is added to the image, then the edges of the disk are much harder to detect. The differences in intensity might be too small to rise above the threshold. One solution to this problem is to smooth the image before edge detection to eliminate the effects of the noise. This will tend to make the interior more uniform and improve detection. A drawback of smoothing is that the true edges will be averaged with background pixels that have different magnitudes, blurring the true edge. Also, smoothing can distort true edges, especially for those edges that are close together, have high curvature, or change direction rapidly at a

corner point. The direction of noise gradients will often be different from true edges so a common practice is to search in different directions and suppress those gradient changes that are in completely different directions than the current edge direction.

This chapter will review several 2D edge detectors including Sobel, Canny, multiscale using wavelet, and multiscale combined with multidirectional using shearlet transforms. The different 2D methods will be compared with respect to their ability to address the complications of edge detection.

2.2 Basic Methods

The most fundamental operation when implementing edge detection is the estimation of the image gradient. There are two common approaches to find the gradient: convolution of the image with the second derivative of a 2D Gaussian function or with the Sobel central differencing operator. It is important to note that both approaches estimate horizontal and vertical directional derivatives, and other directions are not taken into account.

Consider a one-dimensional example with a sinusoidal function

$$f(x) = \sin(x). \tag{2.4}$$

For the first approach, define the standard Gaussian function

$$g_\sigma(x) = \exp\left(-(x^2)/(2 * \sigma^2)\right), \tag{2.5}$$

where σ is the standard deviation for the normal distribution of the x values about

the mean $\bar{x} = 0$. Compute the derivative

$$dg_{\sigma,x}(x) = -\frac{1}{\sigma^2} x \exp\left(-(x^2)/(2 * \sigma^2)\right). \tag{2.6}$$

The derivative of a Gaussian function is seen in Figure 2.2. The gradient of f is

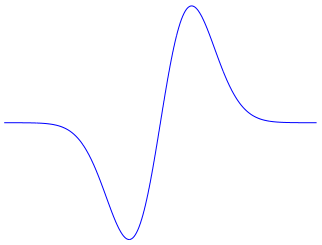


Figure 2.2: Derivative of the Gaussian using $\sigma = 2$

then estimated by the convolution

$$\nabla f \cong f * dg_{\sigma,x}. \tag{2.7}$$

Two results are displayed in Figure 2.3. Clearly, different choices of σ result in

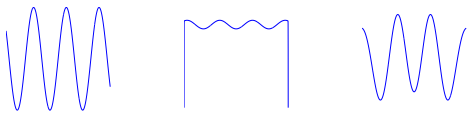


Figure 2.3: Sinusoidal signal (left) and convolution (2.7) at $\sigma = 0.01$ (middle) and $\sigma = 2$ (right).

different approximate derivatives. The derivative of $\sin(x)$ should be $\cos(x)$. Since the average value of $\sin(x)$ is zero, we can choose $\sigma = std(\sin(x)) = 0.6909$ to be

the standard deviation of the sine function. The result is much closer to the true derivative of $\cos(x)$ as seen in Figure 2.4. The convolution should be normalized to

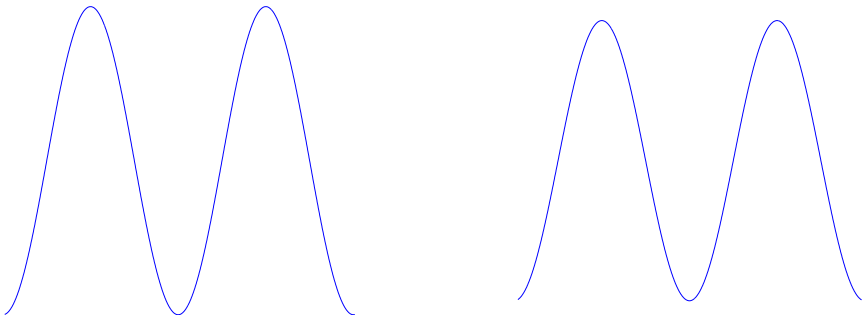


Figure 2.4: True derivative of the sinusoidal signal (left) and convolution approximation (right) using $\sigma = std(\sin(x)) = 0.6909$

one, but the frequency and wavelength of both signals is a closer match to that of the true cosine.

Consider a two dimensional Gaussian function

$$g_\sigma(x, y) = \exp\left(-(x^2 + y^2)/(2 * \sigma^2)\right), \tag{2.8}$$

with distribution standard deviation σ. The gradient of a function $f : [0, 1]^2 \to [0, 1]$ consists of the partial derivative of f with respect to both coordinates x and y:

$$dg_{\sigma,x} \triangleq \frac{\partial g_{\sigma,x}}{\partial x} = -\frac{x}{\sigma} \exp\left(-(x^2 + y^2)/(2 * \sigma^2)\right), \tag{2.9}$$

$$dg_{\sigma,y} \triangleq \frac{\partial g_{\sigma,y}}{\partial y} = -\frac{y}{\sigma} \exp\left(-(x^2 + y^2)/(2 * \sigma^2)\right). \tag{2.10}$$

For a given image $\mathbf{I}$ the horizontal image derivative can be estimated through convolution with $dg_{\sigma,x}$:

$$\nabla \mathbf{I}_{\sigma,x} = \mathbf{I} * dg_{\sigma,x}. \tag{2.11}$$

The vertical image derivative can be estimated similarly:

$$\nabla \mathbf{I}_{\sigma,y} = \mathbf{I} * dg_{\sigma,y}. \tag{2.12}$$

The magnitude of the gradient is then estimated by

$$|\nabla \mathbf{I}| = \sqrt{\nabla \mathbf{I}_{\sigma,x}{}^2 + \nabla \mathbf{I}_{\sigma,y}{}^2}, \tag{2.13}$$

and the direction is

$$\theta_{g_\sigma} = \arctan\left(\nabla \mathbf{I}_{\sigma,y} / \nabla \mathbf{I}_{\sigma,x}\right). \tag{2.14}$$

2.2.1 Sobel Edge Detection

The second approach to estimating the gradient uses the Sobel operator [5]. The Sobel horizontal derivative filter is given by

$$\mathbf{G}_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}, \tag{2.15}$$

and the vertical derivative filter is

$$\mathbf{G}_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}. \tag{2.16}$$

For a given image $\mathbf{I}$ the horizontal image derivative is estimated through convolution

$$\nabla \mathbf{I}_x = \mathbf{I} * \mathbf{G}_x, \tag{2.17}$$

and the vertical image derivative estimate is

$$\nabla \mathbf{I}_y = \mathbf{I} * \mathbf{G}_y. \tag{2.18}$$

The two quantities of interest for the gradient are magnitude and direction defined by (2.13) and (2.14). The Gaussian or Sobel edge detection method then finds the edge image $\mathbf{I_E}$ (2.2). The threshold h must be chosen carefully to minimize the loss of true edge information. The threshold h can be set from image statistics about the given image

$$h = s * mean(|\nabla \mathbf{I}|), \tag{2.19}$$

where s is a scale factor chosen to accept a fraction of the mean intensity change in the image.

2.2.2 Canny Edge Detection

The 2D Canny edge detection algorithm [3] is an improvement over Sobel because it takes into account both image intensity magnitude and direction information from the image gradient. The version we discuss is implemented in Matlab's *edge* function. The first step is to apply a smoothing operation to mitigate the effect of noise:

$$\mathbf{I}_s = \mathbf{I} * g_\sigma, \tag{2.20}$$

where g_σ is defined in equation (2.8). This is also accomplished by discrete convolution with

$$\mathbf{S} = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}. \tag{2.21}$$

Note that the goal of smoothing is to remove high frequency information and simplify the edge detection process. The smoothing averages the intensities of interior regions

so that the magnitudes are more uniform, setting them apart from the boundary. The variance σ^2 must be chosen carefully so as not to lose true edge information. Unfortunately, the boundaries of image objects are blurred by smoothing and obscure the actual location of the true edges. This problem occurs for images with high levels of noise where brightness or shade makes object intensity and background intensities about the same.

After smoothing, the 2D gradient is estimated using equations (2.11) and (2.12). The magnitude is then computed with (2.13) and normalized by the maximum intensity value in the gradient image. This is essentially very similar to the Sobel method up to this point.

The Canny algorithm improves upon the Sobel algorithm because Canny 2D accounts for directional information with a method called *nonmaximal suppression*. For nonmaximal suppression, each gradient magnitude is set to zero if it is not greater than at least one of its neighboring pairs of gradient magnitudes. Since the unit pixel is square, there are only four possible pairs, oriented by the compass directions N-S, E-W, NE-SW, NW-SE, to check. This means that only local information is considered for edge decision criteria. Only neighboring pixels that touch are considered for suppression. That may not account for nearby pixels that don't touch but still could be considered part of an edge. This would be the case for thick edges that are two or more pixels wide. To help overcome this problem nonmaximal suppression is combined with thresholding.

The Canny 2D algorithm improves the standard single thresholding approach by using a more sophisticated thresholding technique called *hysteresis* to determine

the true edge intensity magnitudes. Hysteresis uses two different thresholds t_{low} and t_{high} to help distinguish true edge intensity magnitudes that are just above or even below the intensity of the noise. A pixel is identified as a strong edge pixel if its intensity gradient magnitude is greater than t_{high}. A pixel is also marked as part of an edge if it is connected to a strong edge and its gradient magnitude is larger than t_{low} and larger than the magnitude of each of its two neighbors in at least one of the compass directions (N-S, E-W,NE-SW, NW-SE). The procedure is summarized for a single image in Algorithm 1.

Algorithm 1 The 2D Canny algorithm.

Input: $\mathbf{I}$, σ

Output: Estimate of $|\nabla \mathbf{I}|$

Smooth by forming $\mathbf{I}_s = \mathbf{I} * g_\sigma$.

Compute horizontal derivative $\nabla \mathbf{I}_x = \mathbf{I}_s * dg_{\sigma,x}$.

Compute vertical derivative $\nabla \mathbf{I}_y = \mathbf{I}_s * dg_{\sigma,y}$.

Compute gradient magnitude $|\nabla \mathbf{I}| = \sqrt{\nabla \mathbf{I}_x{}^2 + \nabla \mathbf{I}_y{}^2}$.

Perform nonmaximal suppression of gradient magnitudes.

Perform hysteresis thresholding.

2.2.3 Wavelet Edge Detection

The 2D wavelet edge detector [9] is a multi-scale method designed to address the single-scale smoothing limitations associated with the 2D Canny algorithm described above. The wavelet transform [18] partitions an image according to the frequency of image intensity changes. There will be an image for every different

scale where the transform is defined. The function $\psi = \nabla g$ is a wavelet known as the *first derivative Gaussian wavelet*. Then each image $I \in L^2(\mathbb{R}^2)$ for a fixed scale a satisfies:

$$I(\mathbf{x}) = \int_{\mathbb{R}^2} W_{\psi_a} I(a, \mathbf{y})\, \psi_a(\mathbf{x} - \mathbf{y})\, d\mathbf{y}, \tag{2.22}$$

where $\psi_a(\mathbf{x}) = a^{-1}\, \psi(a^{-1}\mathbf{x})$, and $W_{\psi_a} I(a, \mathbf{x})$ is the *wavelet transform* of I, defined by

$$W_{\psi_a} I(a, \mathbf{x}) = \int I(\mathbf{y})\, \psi_a(\mathbf{x} - \mathbf{y})\, d\mathbf{y} = I * \psi_a(\mathbf{x}). \tag{2.23}$$

The wavelet transform is useful because it provides a space-scale partitioning of the image I. The image $I \in L^2(\mathbb{R}^2)$ is mapped into the coefficients $W_{\psi_a} I(a, \mathbf{y})$ which depend on the location $\mathbf{y} \in \mathbb{R}^2$ and the scaling variable $a > 0$. It is also important to note that the wavelet transform of I is proportional to the gradient of the smoothed image

$$\nabla(I * g_a) = I * \nabla g_a(\mathbf{x}) = I * \psi_a(\mathbf{x}) = W_{\psi_a} I(a, \mathbf{x}), \tag{2.24}$$

so that a particular scale a matches the computation in Canny. The maxima of the magnitude of the gradient of the smoothed image I_a correspond exactly to the maxima of the magnitude of the wavelet transform $W_{\psi_a} I(a, \mathbf{x})$. Therefore the wavelet transform provides a natural mathematical framework for the multiscale analysis of edges [9, 10]. In particular, the multiscale wavelet representation sidesteps the problem of finding the appropriate standard deviation σ so that techniques based on it will improve those of the Canny algorithm defined above. Furthermore, there are very efficient numerical implementations of the wavelet transform [8].

Our implementation of the 2D wavelet edge detector first computes the hori-

zontal image derivative

$$\nabla \mathbf{I}_x = \mathbf{I} * \mathbf{G}_x \tag{2.25}$$

and the vertical image derivative

$$\nabla \mathbf{I}_y = \mathbf{I} * \mathbf{G}_y. \tag{2.26}$$

Next, for each scale $a_i, i = 1, \ldots, n$, we use matrix $\mathbf{S}$ in equation (2.21) to repeatly smooth the previous horizontal and vertical components:

$$\mathbf{G}_x^n = \mathbf{G}_x^{n-1} * \mathbf{S},\ \mathbf{G}_y^n = \mathbf{G}_y^{n-1} * \mathbf{S}. \tag{2.27}$$

This acts as a discretization of the wavelet dilation as the scale a changes. A weight is computed by

$$p_n = \frac{\max\left(\mathbf{G}_x^{n-1}\right)}{\max\left(\mathbf{G}_x^n\right)} \tag{2.28}$$

to scale these approximate dilations to guarantee that the maximum of the wavelet components does not change. The weight p_n is applied to compute the horizontal smoothed image derivative

$$\nabla \mathbf{I}_x^{(n)} = \nabla \mathbf{I}_x^{(n-1)} * p_n \mathbf{S} \tag{2.29}$$

and the vertical smoothed image derivative

$$\nabla \mathbf{I}_y^{(n)} = \nabla \mathbf{I}_y^{(n-1)} * p_n \mathbf{S}. \tag{2.30}$$

After each smoothing we store the dilated derivative $\mathbf{G}_x^n$ to compute p_n for the next scale. Finally a cumulative horizontal image derivative that re-enforces edges is given as

$$\left(\nabla \mathbf{I}_x^{(n)}\right)_i = \begin{cases} \left(\nabla \mathbf{I}_x^{(n-1)}\right)_i & \text{if } |\nabla \mathbf{I}_x^{(n-1)}|_i \leq |\nabla \mathbf{I}_x^{(n)}|_i \\ \left(\nabla \mathbf{I}_x^{(n)}\right)_i & \text{otherwise .} \end{cases} \tag{2.31}$$

The cumulative vertical image derivative as

$$\left(\nabla \mathbf{I}_y^{(n)}\right)_i = \begin{cases} \left(\nabla \mathbf{I}_y^{(n-1)}\right)_i & \text{if } |\nabla \mathbf{I}_y^{(n-1)}|_i \leq |\nabla \mathbf{I}_y^{(n)}|_i \\ \left(\nabla \mathbf{I}_y^{(n)}\right)_i & \text{otherwise .} \end{cases} \tag{2.32}$$

This step accumulates the contribution from each scaling into the image gradient. The last step is to compute the image gradient magnitude by summing the squares of the horizontal and vertical cumulative image derivatives:

$$|\nabla \mathbf{I}|^2 = \nabla \mathbf{I}_x{}^2 + \nabla \mathbf{I}_y{}^2. \tag{2.33}$$

The 2D wavelet algorithm is summarized in Algorithm 2.

Edge detection is complicated by the presence of noise and by edges that are close together, cross each other, or exhibit high curvature [20]. Difficulties with the wavelet approach are summarized by:

- *Difficulty in distinguishing close edges.* The isotropic Gaussian smoothing causes distinct edges that are close together to be blurred into a single curve.

- *Poor angular accuracy.* Sharp changes in curvature or crossing curves lead to an inaccurate detection of the edge direction. This affects the detection of corners and junctions.

To address these difficulties one has to deal with the anisotropic nature of the edge lines and curves. For example, in [13, 16, 4] it has been proposed to replace the scalable collection of isotropic Gaussian filters $g(x, y)$, $\sigma > 0$ in (2.8) with a family of steerable and scalable anisotropic Gaussian filters

$$G_{a_1,a_2,\theta}(x_1, x_2) = a_1^{-1/2}\, a_2^{-1/2} R_\theta\, G(a_1^{-1} x_1, a_2^{-1} x_2),$$

where $a_1, a_2 > 0$ and R_θ is the rotation matrix for angle θ. Unfortunately, the design and implementation of such filters is computationally involved. In addition, the justification for this approach is essentially intuitive and there is no proper theoretical model to indicate how to "optimize" this family of filters to best capture edges.

Image analysis that only considers gradient magnitude information is at a disadvantage because it does not account for any directional information for true edge decision criteria. Using gradient directional information is a critical step in improving edge detection capability. Unfortunately the wavelet is isotropic and does not account for intensity changes in different directions. In the next section we introduce a new directional transform.

2.2.4 Shearlet Edge Detection

In this section, we explain how an anisotropic and multiscale-based transform, known as the *shearlet transform*, can be used to detect and characterize edges. The shearlet characterization of image edges is more complete because the transform not only accounts for different scales of intensity changes, but also for the direction of the edge at each scale for a particular location in the image plane. The Canny method accounts for directional information by using nonmaximal suppression and hysteresis filtering, but only at a single scale, by considering neighboring pixels. The shearlet transform performs a convolution over a window of pixels with a multi-scale, multi-directional operator. The scale, direction, and size of the window helps

Algorithm 2 The 2D wavelet algorithm.

Input: Source image $\mathbf{I}$

Output: Estimate of $|\nabla \mathbf{I}|$

Compute basic horizontal derivative $\nabla \mathbf{I_x} = \mathbf{I} * \mathbf{G_x}$.

Compute basic vertical derivative $\nabla \mathbf{I_y} = \mathbf{I} * \mathbf{G_y}$.

for $i = 1 \rightarrow n$ **do**

Smooth current cumulative gradient operator.

Determine gradient scaling constant.

Smooth horizontal derivative scaled by gradient scaling factor.

Smooth vertical derivative scaled by gradient scaling factor.

Update smoothed gradient operator.

Accumulate horizontal derivative coefficients.

Accumulate vertical derivative coefficients.

end for

Compute gradient magnitude $|\nabla \mathbf{I}| = \sqrt{\nabla \mathbf{I}_x{}^2 + \nabla \mathbf{I}_y{}^2}$.

to capture edge information that doesn't touch a particular pixel but is still part of the true edge.

The shearlet operator, defined below, generates a partition of the frequency plane consisting of angular dependent regions. The number of directions is adaptive and can be set to match the complexity of the edge directional information in the image. Images whose edge directions change rapidly or in more complicated ways would require more shearlet directions while simple edge directions require fewer. It is necessary to first understand how the shearlet transform reveals image intensity

and directional information and then describe how the transform can be integrated into an edge detection system [19].

More specifically the shearlet transform is defined as the mapping

$$\mathcal{SH}_\psi f(a, s, \mathbf{t}) = \langle f, \psi_{as\mathbf{t}} \rangle,\ a > 0, s \in \mathbb{R}, \mathbf{t} \in \mathbb{R}^2,$$

where $\psi_{as\mathbf{t}}(\mathbf{x}) = |\det \mathbf{M}_{as}|^{-\frac{1}{2}} \psi(\mathbf{M}_{as}^{-1}(\mathbf{x} - \mathbf{t}))$ and $\mathbf{M}_{as} = \begin{pmatrix} a & s \\ 0 & \sqrt{a} \end{pmatrix}$. Each matrix $\mathbf{M}_{as}$ can be factored as $\mathbf{B}_s \mathbf{A}_a$, where $\mathbf{B}_s = \begin{pmatrix} 1 & -s \\ 0 & 1 \end{pmatrix}$ is a *shear matrix* and $\mathbf{A}_a = \begin{pmatrix} a & 0 \\ 0 & \sqrt{a} \end{pmatrix}$ is an *anisotropic dilation matrix*. The frequency support of $\psi_{as\mathbf{t}}(\mathbf{t})$ is given by a pair of trapezoids determined by the scale a and shear s. These trapezoids are symmetric with respect to the origin and oriented along a line of slope s. Figure 2.5 provides an illustration for various values of a and s.

The analyzing function ψ needs to be chosen appropriately in order for the transform to be invertible. In particular, for a point in the frequency domain $\boldsymbol{\xi} = (\xi_1, \xi_2) \in \mathbb{R}^2,\ \xi_1 \neq 0$, let the Fourier transform of ψ be given by

$$\hat{\psi}(\boldsymbol{\xi}) = \hat{\psi}(\xi_1, \xi_2) = \hat{\psi}_1(\xi_1)\, \hat{\psi}_2(\tfrac{\xi_2}{\xi_1}),$$

where $\psi_1 \in L^2(\mathbb{R})$ satisfies the Calderòn condition

$$\int_0^\infty |\hat{\psi}(a\boldsymbol{\xi})|^2 \frac{da}{a} = 1 \quad \text{for a.e. } \boldsymbol{\xi} \in \mathbb{R}^2,$$

and $\|\psi_2\|_{L^2} = \left(\int_{\mathbb{R}^2} (\psi_2(\mathbf{x}))^2\, d\mathbf{x}\right)^{\frac{1}{2}} = 1$. Then, for each $f \in L^2(\mathbb{R}^2)$, we have:

$$f(\mathbf{x}) = \int_{\mathbb{R}^2} \int_{-\infty}^{\infty} \int_0^\infty \langle f, \psi_{as\mathbf{t}} \rangle\, \psi_{as\mathbf{t}}(\mathbf{x}) \frac{da}{a^3}\, ds\, d\mathbf{t}.$$

The magnitudes in this representation have different behavior at edges than at smooth regions of the image. Knowing how the magnitudes at the edges will

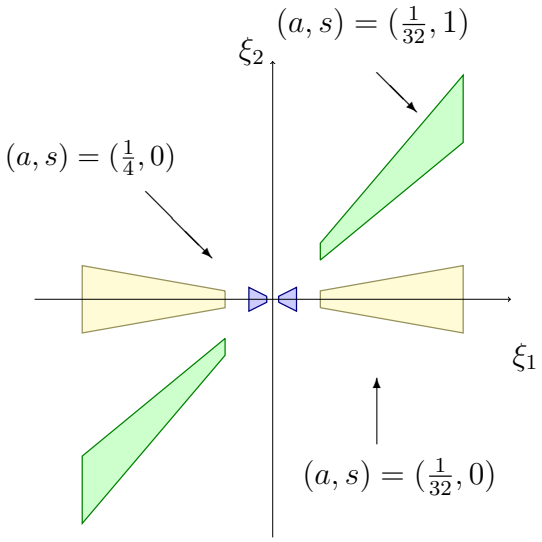


Figure 2.5: Frequency support of some representative shearlet analyzing functions $\hat{\psi}_{ast}$.

mathematically change in this representation with respect to scale a and shear s, a method for detecting edges and their orientation was given in [7]. This precise mathematical framework enables us to explicitly represent both large and small changes in both intensity magnitude and direction. The result is an improved capability to successfully detect subtle intensity differences and more complicated object shapes, especially when objects are moving rapidly. We avoid the problem of choosing the orientation of steering Gaussian filters [13] for wavelets or of using nonmaximal suppression at a single scale as in the Canny edge detector. The solution is to use a collection of directional filters that partitions the entire frequency space at different

scales rather than partitioning the spatial domain. This method has also proven to be very robust in correctly distinguishing edges over noise since the noise does not change with respect to scale in the same way.

To implement the discrete shearlet transform for $\boldsymbol{\xi} = (\xi_1, \xi_2) \in \hat{\mathbb{R}}^2$, scale $a < 1$, and shear $|s| \leq 1$, use the conjugate of the Fourier transform of the analyzing function ψ to define window functions

$$\hat{w}_{a,s}^{(0)}(\boldsymbol{\xi}) = a^{-\frac{1}{4}} \overline{\hat{\psi}_2 \left(a^{-\frac{1}{2}} \left(\frac{\xi_2}{\xi_1} - s \right) \right)} \chi_{\mathcal{D}_0}(\boldsymbol{\xi}), \tag{2.34}$$

$$\hat{w}_{a,s}^{(1)}(\boldsymbol{\xi}) = a^{-\frac{1}{4}} \overline{\hat{\psi}_2 \left(a^{-\frac{1}{2}} \left(\frac{\xi_1}{\xi_2} - s \right) \right)} \chi_{\mathcal{D}_1}(\boldsymbol{\xi}), \tag{2.35}$$

where $\mathcal{D}_0 = \left\{ (\xi_1, \xi_2) \in \hat{\mathbb{R}}^2 : \left| \frac{\xi_2}{\xi_1} \leq 1 \right| \right\}$, $\mathcal{D}_1 = \left\{ (\xi_1, \xi_2) \in \hat{\mathbb{R}}^2 : \left| \frac{\xi_1}{\xi_2} \leq 1 \right| \right\}$. For $a < 1$, $|s| \leq 1$, $\mathbf{t} \in \mathbb{R}^2$, in two dimensions $d = 0, 1$, the Fourier transform of the shearlets can be expressed as

$$\hat{\psi}_{a,s,\mathbf{t}}^{(d)}(\boldsymbol{\xi}) = a V^{(d)}(a\boldsymbol{\xi}) \overline{\hat{w}_{a,s}^{(d)}(\boldsymbol{\xi})} e^{-2\pi i \boldsymbol{\xi} \mathbf{t}}, \tag{2.36}$$

where $V^{(0)}(\xi_1, \xi_2) = \overline{\hat{\psi}_1(\xi_1)}$, $V^{(1)}(\xi_1, \xi_2) = \overline{\hat{\psi}_1(\xi_2)}$. The shearlet transform of $I \in L^2(\mathbb{R}^2)$ is:

$$\mathcal{SH}_{\psi}^{(d)} I(a, s, \mathbf{t}) = a \int_{\mathbb{R}^2} \hat{I}(\boldsymbol{\xi}) V^{(d)(a\boldsymbol{\xi})}(a\boldsymbol{\xi}) \hat{w}_{a,s}^{(d)}(\boldsymbol{\xi}) e^{2\pi i \boldsymbol{\xi} \mathbf{t}} \, d\xi, \tag{2.37}$$

where $d = 0, 1$ correspond to the horizontal and vertical directions respectively. Therefore from (2.37) we have

$$\mathcal{SH}_{\psi}^{(d)} I(a, s, \mathbf{t}) = v_a^{(d)} I * w_{a,s}^{(d)}(\mathbf{t}), \tag{2.38}$$

where

$$v_a^{(d)} I(\mathbf{t}) = \int_{\mathbb{R}^2} a \hat{\mathbf{I}}(\boldsymbol{\xi}) V^{(d)}(a\boldsymbol{\xi}) e^{2\pi i \boldsymbol{\xi} \mathbf{t}} \, d\boldsymbol{\xi}. \tag{2.39}$$

To develop a transform useful for edge detection we choose the functions $\hat{\psi}_1$ to be odd and $\hat{\psi}_2$ to be even. For $m, n \in \mathbb{N}$ an $m \times n$ image can be considered as a set of samples $\mathbf{I}[n_1, n_2] : n_1, n_2 = 0, \ldots, m-1$. Therefore by identifying the domain with $\mathbb{Z}_N^2$ we may view $l^2(\mathbb{Z}_N^2)$ as the discrete analog of $L^2(\mathbb{R}^2)$. The inner product of the images of $\mathbf{I}_1$ and $\mathbf{I}_2$ is defined to be

$$\langle \mathbf{I}_1, \mathbf{I}_2 \rangle = \sum_{n_1=0}^{m-1} \sum_{n_2=0}^{n-1} \mathbf{I}_1[n_1, n_2] \overline{\mathbf{I}_2[n_1, n_2]}. \tag{2.40}$$

For $m/2 \leq k_1, k_2 < n/2$, the 2D discrete Fourier transform (DFT) $\hat{\mathbf{I}}[k_1, k_2]$ is given by

$$\hat{\mathbf{I}}[k_1, k_2] = \frac{1}{m} \sum_{n_1=0}^{m-1} \sum_{n_2=0}^{n-1} \mathbf{I}[n_1, n_2] e^{-2\pi i \left(\frac{n_1}{m} k_1 + \frac{n_2}{n} k_2\right)}. \tag{2.41}$$

For notational purposes $[.,.]$ denotes arrays for indices and $(.,.)$ denotes function evaluations. The numbers $\hat{\mathbf{I}}[k_1, k_2]$ are samples from the trigonometric polynomial $\hat{I}(\xi_1, \xi_2) = \sum_{n_1=0}^{m-1} \sum_{n_2=0}^{n-1} \mathbf{I}[n_1, n_2] e^{-2\pi i \left(\frac{n_1}{m} \xi_1 + \frac{n_2}{n} \xi_2\right)}$. To implement the window functions $w_{a,s}^{(d)}$, we compute the DFT on a grid consisting of lines across the origin at various slopes called the *pseudo-polar grid* and then apply a band-pass filter with respect to this grid. Define the pseudo-polar coordinates $(\zeta_1, \zeta_2) \in \mathbb{R}^2$ by

$$(\zeta_1, \zeta_2) = (\xi_1, \frac{\xi_2}{\xi_1}) \qquad \text{if } (\xi_1, \xi_2) \in \mathcal{D}_0, \tag{2.42}$$

$$(\zeta_1, \zeta_2) = (\xi_2, \frac{\xi_1}{\xi_2}) \qquad \text{if } (\xi_1, \xi_2) \in \mathcal{D}_1. \tag{2.43}$$

Using this change of coordinates we get

$$\widetilde{\hat{v}_a^{(d)}} f(\zeta_1, \zeta_2) = \hat{v}_a^{(d)} f(\xi_1, \xi_2), \tag{2.44}$$

$$\widetilde{\hat{w}^{(d)}}(a^{-1/2}(\zeta_2 - s)) = \hat{w}_{a,s}^{(d)}(\xi_1, \xi_2). \tag{2.45}$$

The different directional components are obtained by translating the window function $\widetilde{\hat{w}^{(d)}}$. At the scale $a = 2^{-2j}$, $j \geq 0$, let $v_j^{(d)}\mathbf{I}[n_1, n_2]$ be the discrete samples of a function $v_{2^{-2j}}^{(d)} I(x_1, x_2)$ with Fourier transform $\hat{v}_a^{(d)} I(\xi_1, \xi_2)$. Also the discrete samples $\widetilde{\hat{v}_j^{(d)}}\mathbf{I}[k_1, k_2] = \widetilde{\hat{v}_{2^{-2j}}^{(d)}} I(k_1, k_2)$ are the values of the DFT of $v_a^{(d)}$ on the pseudo-polar grid by direct extraction using the Fast Fourier Transform (FFT). To discretize the window function use $\widetilde{w^{(d)}}$ where

$$\sum_{d=0}^{1} \sum_{l=2^j}^{2^j-1} \widetilde{\hat{w}^{(d)}}[2^j k_2 - l] = 1, \tag{2.46}$$

and the shearing variable is discretized as $s_{j,l} = 2^{-j} l$. Let Φ_p be a mapping function from the Cartesian grid to the pseudo polar grid. Then the discrete shearlet transform can be written in the discrete frequency domain as

$$\Phi_p^{-1}\left(\widetilde{\hat{v}_j^{(d)}}\mathbf{I}[k_1, k_2]\right) \Phi_p^{-1}\left(\hat{\delta}_p[k_1, k_2]\widetilde{\hat{w}^{(d)}}[2^j k_2 - l]\right), \tag{2.47}$$

where $\hat{\delta}_p$ is the discrete Fourier transform of the dirac delta function δ_p in the pseudo polar grid. Therefore, the discrete shearlet transform can be implemented as

$$\mathcal{SH}^{(d)}\mathbf{I}[j, l, k_1, k_2] = v_j^{(d)}\mathbf{I} * w_{j,l}^{(d)}[k_1, k_2], \tag{2.48}$$

where

$$\hat{w}_{j,l}^{(d)}[k_1, k_2] = \Phi_p^{-1}\left(\hat{\delta}_p[k_1, k_2]\overline{\widetilde{\hat{w}^{(d)}}[2^j k_2 - l]}\right). \tag{2.49}$$

To compute the discrete shearlet transform, let H_j and G_j be the low-pass and high-pass filters of a wavelet transform with $2^j - 1$ zeros inserted between consecutive coefficients of both filters. Given 1-D filters H and G define $\mathbf{I} * (H, G)$ to be the separable convolution of the rows and columns of $\mathbf{I}$ with H and G. The filter G is the

wavelet filter corresponding to $\hat{\psi}_1$. The function $\hat{\psi}_1$ must be odd and H represents the coarse scale. The window functions $\hat{w}^{(d)}$ are related to the functions $\hat{\psi}_2$ which are even. The $\hat{\psi}_2$ functions are implemented using a Meyer-type filter [8]. The Meyer wavelet is frequency band-limited with a Fourier transform that is smooth. Because it is smooth it has a much faster asymptotic decay rate over time. The 2D shearlet transform is summarized in Algorithm 4.

Algorithm 3 The 2D discrete shearlet algorithm.

Input: $\mathbf{I} \in l^2(\mathbb{Z}_N^2)$.

Output: Discrete shearlet transform $\mathcal{SH}^{(d)}\mathbf{I}[j,l,\mathbf{k}] = \mathbf{v_j^{(d)}}\mathbf{I} * \mathbf{w_{j,l}^{(d)}}[\mathbf{k}]$

$S_0\mathbf{I} = \mathbf{I}$.

for $j = 1 \to n$ **do**

Separable convolution for rows and columns of $S_j\mathbf{I} = S_{j-1}\mathbf{I} * [H_j, H_j]$, $j \geq 1$.

for direction $d = 0, 1$ **do**

The discrete shearlet transform is $\mathcal{SH}^{(d)}\mathbf{I}[j,l,\mathbf{k}] = v_j^{(d)}\mathbf{I} * w_{j,l}^{(d)}[\mathbf{k}]$

where $j \geq 0$, $-2^j \leq l \leq 2^j - 1$, $\mathbf{k} \in \mathbb{Z}^\mathbf{2}$ and

$v_j^{(0)}\mathbf{I} = S_j\mathbf{I} * [G_j, \delta]$,

$v_j^{(1)}\mathbf{I} = S_j\mathbf{I} * [\delta, G_j]$.

end for

end for

To implement the 2D shearlet edge detection algorithm first compute the horizontal and vertical image derivatives as in equations (2.25) and (2.26) for the wavelet. This approximates the components $v_j^{(d)}$. The directional windows $w_{j,\ell}^{(d)}$ are constructed as follows: The first step is to generate the $\mathbf{x}$ and $\mathbf{y}$ vectors containing

the polar coordinates to extract radial slices. Then for each direction compute the Meyer window of bandpass filters $\mathbf{w}$ of length $N \times L$ where N is the window size and L is the number of directions. Next reassemble each radial slice into a matrix of filters for each direction. The shearlet filters are computed off-line before processing begins. To begin processing, at each scale use (2.27) and (2.28) to compute the nth scaling weight p_n. Apply (2.29) and (2.30) to give the horizontal $\nabla \mathbf{I}_x$ and vertical $\nabla \mathbf{I}_y$ dilated image derivatives by smoothing with $\mathbf{S}$. After each smoothing store the dilated $\mathbf{G}_x^n$ to compute p_n for the next scale. To compute the horizontal shearlet directional image derivatives two ancillary quantities are necessary: the shearlet directional filter convolved with the basic image gradient without smoothing and the image gradient after smoothing. For each scale a and slope direction s, we use the shearlet directional filter to compute the basic horizontal shearlet image derivative

$$\nabla \mathbf{I}_{x,s} = \nabla \mathbf{I}_x * w_{a,s}. \tag{2.50}$$

Similarly, the basic vertical shearlet image derivative is

$$\nabla \mathbf{I}_{y,s} = \nabla \mathbf{I}_y * w_{a,s}. \tag{2.51}$$

The resultant shearlet coefficients serve as a reference to later accumulate the edge locations at the current dilation and above. The horizontal shearlet coefficients are

$$\nabla \mathbf{I}_{x,s}^{(n)} = \nabla \mathbf{I}_x^{(n)} * w_{a,s} \tag{2.52}$$

and the vertical shearlet coefficients are

$$\nabla \mathbf{I}_{y,s}^{(n)} = \nabla \mathbf{I}_y^{(n)} * w_{a,s}. \tag{2.53}$$

For each direction s, we retain those horizontal smoothed/scaled shearlet coefficients that are larger than the previous scaled horizontal shearlet coefficients

$$\nabla \mathbf{I}_{x_d}^{(n)} = \nabla \mathbf{I}_{x,s}^{(n)} .* (\left|\nabla \mathbf{I}_{x,s}^{(n)}\right| \geq \left|\nabla \mathbf{I}_{x,s}^{(n-1)}\right|) \tag{2.54}$$

and the vertical smoothed shearlet coefficients that are larger than the previous scaled vertical shearlet coefficients

$$\nabla \mathbf{I}_{y_d}^{(n)} = \nabla \mathbf{I}_{y,s}^{(n)} .* (\left|\nabla \mathbf{I}_{y,s}^{(n)}\right| \geq \left|\nabla \mathbf{I}_{y,s}^{(n-1)}\right|). \tag{2.55}$$

Add the larger horizontal shearlet coefficients to the cumulative horizontal shearlet directional coefficient sum

$$\nabla \mathbf{I}_{x_c}^{(n)} = \nabla \mathbf{I}_{x_c}^{(n-1)} + \nabla \mathbf{I}_{x_d}^{(n)}, \tag{2.56}$$

and also add the larger vertical shearlet coefficients to the cumulative vertical shearlet directional coefficient sum

$$\nabla \mathbf{I}_{y_c}^{(n)} = \nabla \mathbf{I}_{y_c}^{(n-1)} + \nabla \mathbf{I}_{y_d}^{(n)}. \tag{2.57}$$

After accumulating coefficients in each direction, add cumulative horizontal shearlet coefficients to the previous shearlet coefficients for the current scale estimate

$$\nabla \mathbf{I}_{x}^{(n)} = \nabla \mathbf{I}_{x}^{(n-1)} .* (\left|\nabla \mathbf{I}_{x}^{(n-1)}\right| \leq |\nabla \mathbf{I}_{x_c}|) + \nabla \mathbf{I}_{x_c}^{(n)} \tag{2.58}$$

and the cumulative vertical shearlet coefficients to the previous set of coefficients for the current scale

$$\nabla \mathbf{I}_{y}^{(n)} = \nabla \mathbf{I}_{y}^{(n-1)} .* (\left|\nabla \mathbf{I}_{y}^{(n-1)}\right| \leq |\nabla \mathbf{I}_{y_c}|) + \nabla \mathbf{I}_{y_c}^{(n)}. \tag{2.59}$$

The last step is to compute the image gradient magnitude by summing the squares of the horizontal and vertical cumulative image derivatives (shearlet coefficients)

$$|\nabla \mathbf{I}|^2 = \nabla \mathbf{I}_x{}^2 + \nabla \mathbf{I}_y{}^2. \tag{2.60}$$

The 2D shearlet based edge detection procedure for a single image is summarized in Algorithm 4.

2.3 Conclusions

We have presented a series of increasingly complicated 2D edge detection methods including Canny, wavelet, and shearlet. The Canny method mitigates noise by first smoothing the image to eliminate high frequency information. Canny also uses edge directional information to suppress gradient magnitudes that are non-maximal. The wavelet extends the ability to smooth at different scales and detect different bandwidths of intensity magnitudes using dilated waveforms. The shearlet transform extends the directional capability of edge detection through convolution with shearlet filters that form a finer partition of the frequency space to include more directions than the other methods. When object shapes and motions are simple then all the edge detectors perform about the same. However when edges run close together or noise increases the more complicated methods outperform simpler methods. None of these methods measure the magnitude or direction of edge changes from one image to the next, but these changes do indeed exist. The next chapter will extend the edge detection problem into the third dimension, time.

Algorithm 4 The 2D shearlet edge detection algorithm.

Input: Source image $\mathbf{I}$

Output: Estimate of $|\nabla \mathbf{I}|$

Compute basic horizontal derivative $\nabla \mathbf{I_x} = \mathbf{I} * \mathbf{G_x}$.

Compute basic vertical derivative $\nabla \mathbf{I_y} = \mathbf{I} * \mathbf{G_y}$.

Compute the shearlet filters.

for scale $s = 1 \rightarrow n_s$ **do**

Smooth current cumulative gradient operator.

Determine gradient scaling constant.

Smooth horizontal image derivative scaled by gradient scaling factor.

Smooth vertical image derivative scaled by gradient scaling factor.

Update smoothed gradient operator.

for direction $d = 1 \rightarrow n_d$ **do**

Compute shearing direction with smoothed horizontal derivative.

Compute shearing direction with smoothed vertical derivative.

end for

Accumulate horizontal derivative coefficients.

Accumulate vertical derivative coefficients.

end for

Compute gradient magnitude $|\nabla \mathbf{I}| = \sqrt{\nabla \mathbf{I}_x{}^2 + \nabla \mathbf{I}_y{}^2}$.

Chapter 3

Three Dimensional Edge Detection

3.1 Introduction

Traditional edge detection is performed with a single image. Edge detection in two dimensions highlights the boundaries of image features while setting the interior region to zero. The product of this process is an outline or silhouette of the outstanding image features. Two dimensional edge magnitude and direction shows local information about the orientation boundary for the current image only. However, for many applications there is a sequence of images available to capture information about an event. Therefore three dimensional (3D) edge detectors can operate on an entire movie consisting of a fixed number of image frames processed as a batch. What is new about 3D edge detection is the fact that two dimensional edge boundaries are linked together to form a hollow *edge surface* structure, revealing a three dimensional surface representing the trajectory of the edge image over time. Each *x-y* slice of the edge surface is still the image of the edges of the observed object at some time, but it is based on the nearby image neighbors from the block of images. Therefore more information is provided for each edge pixel because each edge surface pixel consists of an intensity and a three dimensional vector normal to the edge surface. Pixels inside the surface are set to zero, similar to the 2D case. The surface normal points toward the future location of the edge. Another new

and interesting consequence of 3D edge detection is better depth perception when observing objects whose motion is parallel to the line of sight. If the observer line of sight and motion become parallel, it is nearly impossible with 2D edge detection to detect that the object moved. For example , when the motion of a ball is directly lined up with the observer's line of sight the image sequence will be a collection of disks in the center of the image. No perceptible motion can be inferred from an image sequence from this viewpoint using 2D edge detection. However, if we process the block of images with 3D edge detection the derivative in the time direction records changes, and we know the object moved. In this case the object will change scale and edge slices will get larger if the object is moving towards the camera and smaller if it is moving away. The result is a cylindrical surface membrane that expands or contracts depending on the motion of the object with respect to the camera. This is significant information needed to estimate the motion along the line of sight. Note that no extra cameras are needed to detect the motion from another viewpoint normal to the motion of the object in such cases. The application of three dimensional edge detection to tracking will be studied further in Chapter 4.

Previous work in the area of 3D edge detection goes back to Monga and Deriche [12] , who extended the 2D Canny operator to three dimensions. They implemented 3D separable Gaussian masks computing the image derivative and smoothing for each orthogonal direction x, y, and t. This was one of the first extensions of the Canny method into 3D. Their method was applied to magnetic resonance and echographic volumetric data. Monga and Benayoun [11] extended the state of the art mathematically by using partial derivatives to treat the 3D image as a hypersurface

or three dimensional manifold in $\mathbb{R}^4$. They computed the curvatures at designated edge points using the partial derivatives of the image. In particular, surface characteristics such as maximum curvature and ridge lines were investigated using the second partial derivative known as the Laplacian of the image. The detectors based on the Laplacian provided edge localization and magnitude but did not provide directional information. Brejl and Sonka [1] suggested a directional 3D edge detector designed for anisotropic image data. The 3D wavelet has been used to detect cerebral vessels by Weiping and Hauzhong [17] in magnetic resonance angiogram (MRA).

The material in this chapter closely follows [14]. Section 3.2 presents data and background needed to implement and use different three dimensional edge detectors. Section 3.3 discusses the details of different edge detection approaches. Finally section 3.4 defines several different experiments, presents the results from different edge detection methods, and compares their respective advantages and disadvantages. Matlab implementations of these algorithms are available at *http:// www.cs.umd.edu/users/oleary/software*.

3.2 Edge Detection Characteristics of Wavelets and Shearlets

The desire to locate perceptible changes in image intensities or edges corresponds with the ongoing need to detect and estimate features in images. The fundamental component data for this problem is a single image frame. We assume a camera records frames at a particular frame rate f in units of [frames/sec.] over the

time period T seconds. The result of this observation is a discrete image sequence

$$\tilde{\mathbf{I}} = \{\mathbf{I}_0, \mathbf{I}_1, \ldots, \mathbf{I}_\ell\}\,. \tag{3.1}$$

Edge detection applications include imagery collected from medical sensors, surveillance video, and many others. Automating the extraction of image features and detecting small but important jumps in intensity has proven difficult with many problems yet to be resolved. Real world data often provides illumination changes, occlusions, complicated clutter, and noise that challenges the robustness of the latest technology. Even under ideal conditions, edge features are blurred or lost for edges that run close together. To mitigate the effects of noise, the image should first be smoothed with a low-pass averaging filter or a Gaussian filter to remove higher frequency image variations that are unlikely to be present in the true image. However it is important not to remove too much high frequency information that belongs to the actual image feature itself. The smoothing is accomplished by choosing a particular standard deviation σ for a Gaussian filter $\mathbf{g}_\sigma$ and convolving it with image $\mathbf{I}$ to generate a smoothed image

$$\mathbf{I}_\sigma = \mathbf{I} * \mathbf{g}_\sigma. \tag{3.2}$$

The difficulty in this step is choosing the correct image specific standard deviation. There is a delicate balance between removing unnecessary noise and possibly losing information. Loss of information can take place at a particular location when the magnitude of the image gradient drops below the set thresholds even though the pixel truly belongs to the structure of the image feature. Additive Gaussian noise further complicates the task. The noise by nature adds a constant distribution

of intensities to the image that tends to make the intensity magnitudes have less contrast from one pixel to the next. Also lighting could change from one frame to the next revealing a small gradient magnitude that is part of the image feature. The degree of smoothing would have to be changed to preserve the true edge feature. The previous standard deviation setting could cause the true edge to be lost when averaging the pixel values in that particular location even if noise is low. Therefore it is no surprise that the differences would tend to drop below constant thresholds and disappear.

Two-dimensional wavelet and shearlet approaches have proven effective in isolating edges at different scales. The 3D wavelet transform has been applied to medical images by Weiping and Hauzhong [17]. However, to the best of our knowledge, 3D shearlets have never been used for edge detection, so we now develop these ideas first for 2D and then for 3D.

The 2D *continuous wavelet transform* of image I is given by

$$W_\varphi I(\mathbf{M}, \tau) = \langle dgI, \varphi_{\mathbf{M},\tau} \rangle, \tag{3.3}$$

where $\mathbf{M}$ is a 2×2 invertible matrix. A common matrix to use is $\mathbf{M}$ equal to the 2×2 identity matrix and $a > 0$. The analysis functions

$$\varphi_{\mathbf{M},\boldsymbol{\tau}}(\mathbf{e}) = |\det \mathbf{M}|^{-\frac{1}{2}} \varphi(\mathbf{M}^{-1}(\mathbf{e} - \boldsymbol{\tau})) \quad \boldsymbol{\tau} \in \mathbb{R}^2 \tag{3.4}$$

are well localized waveforms that can decompose images $I \in L^2(\mathbb{R}^2)$ so that

$$I = \int_{\mathbb{R}^2} \langle I, \varphi_{\mathbf{M},\boldsymbol{\tau}} \rangle \, \varphi_{\mathbf{M},\boldsymbol{\tau}} \, d\boldsymbol{\tau}. \tag{3.5}$$

By analyzing the the magnitudes of $\langle I, \varphi_{\mathbf{M},\boldsymbol{\tau}} \rangle$ as a function of scale a, edges can be detected. Unfortunately, the wavelet approach does not isolate any directional information. The wavelet transform is isotropic since the dilation factor is the same in all coordinate directions. Therefore, the wavelet has poor angular accuracy for edges that cross or have sharp curvature. Multi-directional shearlet analysis, on the other hand, has demonstrated better success at isolating edges whose orientations change in complicated ways. Given the analyzing function

$$\varphi_{a,s,\boldsymbol{\tau}}(\mathbf{e}) = |\det \mathbf{M}|^{-\frac{1}{2}} \varphi(\mathbf{M}_{a,s}^{-1}(\mathbf{e} - \boldsymbol{\tau})), \tag{3.6}$$

where $\mathbf{M}_{as} = \begin{pmatrix} a & -\sqrt{a}s \\ 0 & \sqrt{a} \end{pmatrix}$, $a \in \mathbb{R}^+$, $s \in \mathbb{R}$, and $\boldsymbol{\tau} \in \mathbb{R}^2$, the *2D continuous shearlet transform* is defined as

$$\mathcal{SH}_\varphi : I \to \mathcal{SH}_\varphi I(a, s, \boldsymbol{\tau}) = \langle I, \varphi_{a,s,\boldsymbol{\tau}} \rangle . \tag{3.7}$$

The matrix $\mathbf{M}_{as}$ performs both the shearing and anisotropic dilation. Likewise, these directional waveforms decompose images in $L^2(\mathbb{R}^2)$ so that

$$I = \int_{\mathbb{R}^2} \int_{-\infty}^{\infty} \int_0^{\infty} \langle I, \varphi_{\mathbf{M}_{as},\boldsymbol{\tau}} \rangle \, \varphi_{\mathbf{M}_{as},\boldsymbol{\tau}} \frac{da}{a^3} \, ds \, d\boldsymbol{\tau}. \tag{3.8}$$

These analyzing functions can represent scale, location and orientation of important image features such as edges. The edges can be precisely characterized from the asymptotic decay of $\mathcal{SH}_\varphi I(a, s, \boldsymbol{\tau})$. Given the collection of edge locations $\mathbf{P}$, these characteristics outlined in [6] are:

- If $\boldsymbol{\tau} \notin \mathbf{P}$, then $\mathcal{SH}_\varphi I(a, s, \boldsymbol{\tau})$ decays rapidly as $a \to 0$ for each $s \in \mathbb{R}$. By rapid decay, we mean for any $N \in \mathbb{N}$ there is a $C_N > 0$ such that $|\mathcal{SH}_\varphi I(a, s, \boldsymbol{\tau})| \leq Ca^N$ as $a \to 0$.

- If $\boldsymbol{\tau} \in \mathbf{P}$ and $\mathbf{E}$ is smooth near $\boldsymbol{\tau}$, then $|\mathcal{SH}_{\varphi} I(a, s, \boldsymbol{\tau})|$ decays rapidly as $a \to 0$ for each $s \in \mathbb{R}$ unless $s = s_0$ is the normal orientation to $\mathbf{P}$ at $\boldsymbol{\tau}$ where $\mathcal{SH}_{\varphi} I(a, s, \boldsymbol{\tau}) \sim a^{\frac{3}{4}}$, as $a \to 0$.

- If $\boldsymbol{\tau}$ is a corner point of $\mathbf{P}$ and $s = s_0$, and $s = s_1$ are normal orientations to the $\mathbf{P}$ at $\boldsymbol{\tau}$, then $|\mathcal{SH}_{\varphi} I(a, s_0, \boldsymbol{\tau})|$, $|\mathcal{SH}_{\varphi} I(a, s_1, \boldsymbol{\tau})| \sim a^{\frac{3}{4}}$ as $a \to 0$. For all other orientations the asymptotic decay of $|\mathcal{SH}_{\varphi} I(a, s, \boldsymbol{\tau})|$ is faster.

Up to this point, attention has been focused on one image at a time without regard for the possibility of processing an image sequence as a whole. Our intent is to demonstrate the advantage of processing a block of data collectively instead of sequentially on a frame-by-frame basis. Extending wavelet edge detection to 3D data gives extra information that will likely help mitigate noise and improve identification. The multi-scale, multi-directional aspect of the 3D shearlet transform tracks edge information better than the 3D wavelet transform because of its added directional selectivity.

To understand the directional selectivity, one should realize that the 3D shearlet transform partitions the spatial frequency domain into a number of subdomains shaped like hyper-trapezoids as shown in Figure 3.1. Specifically, 3D shearlets are constructed by first restricting the subspace of $L^2(\mathbb{R}^3)$ to be $L^2(C^{(1)})^{\vee} = \{f \in L^2(\mathbb{R}^3) : \text{supp}\widehat{f} \subset C^{(1)}\}$, where $C^{(1)}$ is the horizontal cone in the frequency plane:

$$C^{(1)} = \{(\eta_1, \eta_2, \eta_3) \in \mathbb{R}^3 : |\eta_1| \geq 2, \left|\frac{\eta_2}{\eta_1}\right| \leq 1 \text{ and } \left|\frac{\eta_3}{\eta_1}\right| \leq 1\}. \tag{3.9}$$

We consider the shearlet group

$$\Lambda^{(1)} = \left\{ (\mathbf{M}_{as_1s_2,\mathbf{x}}) : 0 \leq a \leq \frac{1}{4}, -\frac{3}{2} \leq s_1 \leq \frac{3}{2}, -\frac{3}{2} \leq s_2 \leq \frac{3}{2}, \mathbf{x} \in \mathbb{R}^2 \right\}, \quad (3.10)$$

where $\mathbf{M}_{as_1s_2} = \begin{pmatrix} a & -a^{1/2}s_1 & -a^{-1/2}s_2 \\ 0 & a^{1/2} & 0 \\ 0 & 0 & a^{1/2} \end{pmatrix}$. Then the following (see proposition 2.1 from [6]) defines the conditions on the function $\varphi^{(1)}$ to generate a continuous shearlet transform on $L^2(C^{(1)})^\vee$ where the shearlet analyzing function is $\varphi^{(1)}_{as_1s_2\mathbf{x}}(\mathbf{y}) = |\det \mathbf{M}_{as_1s_2}|^{-\frac{1}{2}} \varphi^{(1)}(\mathbf{M}^{-1}_{as_1s_2}(\mathbf{y}-\mathbf{x}))$. For $\boldsymbol{\eta} = (\eta_1, \eta_2, \eta_3) \in \mathbb{R}^3, \eta_1 \neq 0$, let the function φ be such that

$$\hat{\varphi}^{(1)}(\boldsymbol{\eta}) = \hat{\varphi}^{(1)}(\eta_1, \eta_2, \eta_3) = \hat{\varphi}_1(\eta_1)\hat{\varphi}_2\left(\frac{\eta_2}{\eta_1}\right)\hat{\varphi}_2\left(\frac{\eta_3}{\eta_1}\right). \quad (3.11)$$

If $\varphi_1 \in L^2(\mathbb{R})$ satisfies the Calderòn condition

$$\int_0^\infty |\hat{\varphi}_1(a\boldsymbol{\eta})|^2 \frac{da}{a} = 1 \quad \text{for a.e.} \quad \boldsymbol{\eta} \in \mathbb{R}^3 \quad (3.12)$$

with supp $\hat{\varphi}_1 \subset \left[-2, -\frac{1}{2}\right] \cup \left[\frac{1}{2}, 2\right]$ and $\|\varphi_2\|_{L^2} = 1$ with supp $\hat{\varphi}_2 \subset \left[-\frac{\sqrt{2}}{4}, \frac{\sqrt{2}}{4}\right]$, then

$$f(\mathbf{u}) = \int_{\mathbb{R}^3} \int_{-\frac{3}{2}}^{\frac{3}{2}} \int_{-\frac{3}{2}}^{\frac{3}{2}} \int_0^{\frac{1}{4}} \langle f, \varphi^{(1)}_{as_1s_2\mathbf{x}} \rangle \varphi^{(1)}_{as_1s_2\mathbf{x}}(\mathbf{u}) \frac{da}{a^4}\, ds_1\, ds_2\, d\mathbf{x} \quad (3.13)$$

for all $f \in L^2(C^{(1)})$.

In the frequency domain a shearlet $\varphi^{(1)}_{as_1s_2\mathbf{x}}$ is defined by

$$\hat{\varphi}^{(1)}_{as_1s_2\mathbf{x}}(\eta_1, \eta_2, \eta_3) = a\hat{\varphi}_1(a\eta_1)\hat{\varphi}_2(a^{-\frac{1}{2}}(\frac{\eta_2}{\eta_1} - s_1))\hat{\varphi}_2(a^{-\frac{1}{2}}(\frac{\eta_3}{\eta_1} - s_2))e^{-2\pi i \boldsymbol{\eta}\mathbf{x}}. \quad (3.14)$$

Therefore, the functions $\hat{\varphi}^{(1)}_{as_1s_2x}$ have support in the sets:

$$\{(\eta_1, \eta_2, \eta_3) \mid \eta_1 \in \left[-\frac{2}{a}, -\frac{1}{2a}\right] \cup \left[\frac{1}{2a}, \frac{2}{a}\right], \left|\frac{\eta_2}{\eta_1} - s_1\right| \leq \frac{\sqrt{2}}{4}a^{\frac{1}{2}}, \left|\frac{\eta_3}{\eta_1} - s_2\right| \leq \frac{\sqrt{2}}{4}a^{\frac{1}{2}}\}. \quad (3.15)$$

The frequency support is a pair of hyper-trapezoids that are symmetric with respect to the origin with orientation determined by slope parameters s_1 and s_2 and that become elongated as $a \to 0$ as indicated in Figure 3.1. This construction is further extended to cover the entire space $L^2(\mathbb{R}^3)$ by forming similar components valid on complementary cone regions for the generating functions $\varphi^{(2)}$ and $\varphi^{(3)}$ (see [6] for more details). The superscript is then dropped and the notation φ is simply used to denote the combined generating functions.

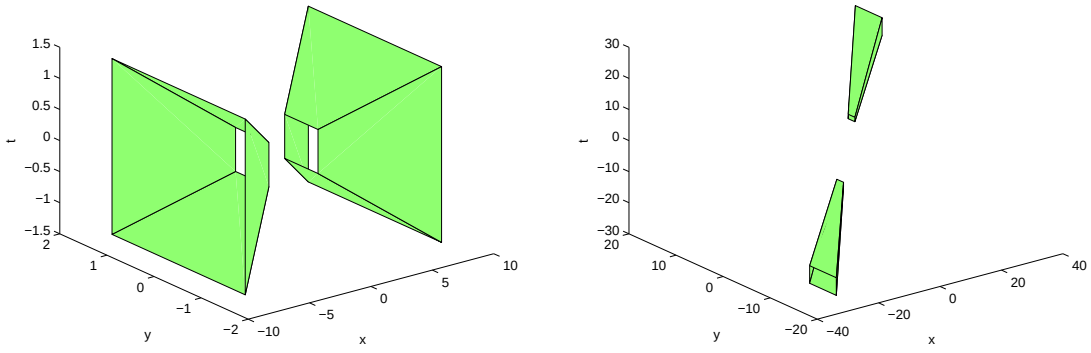


Figure 3.1: The support of a 3D shearlet $\hat{\varphi}_{as_1s_2x}$ in the frequency domain with $a = 1/4$ and $s_1 = s_2 = 0$ (left) and $a = 1/16$, $s_1 = 0.5$, and $s_2 = 0.7$ (right).

To characterize singularities (edge point locations), consider the example of the 3D Heaviside function $H(y_1, y_2, y_3) = \mathbf{1}_{\{y_1>0\}}(y_1, y_2, y_3)$ where $\mathbf{1}_Y$ denotes the characteristic function of the set Y. It is then known that the following are true for $\mathcal{SH}_\varphi H(a, s_1, s_2, \mathbf{x}) = \langle H, \varphi_{as_1s_2\mathbf{x}} \rangle$ [6]:

- If point $\mathbf{x} = (x_1, x_2, x_3)$, with $x_1 \neq 0$, then

$$\lim_{a\to 0^+} a^{-N}\mathcal{SH}_\varphi H(a, s_1, s_2, \mathbf{x}) = 0 \quad \text{for all} \quad N > 0.$$

- If $s_1 \neq 0$ or $s_2 \neq 0$, then

$$\lim_{a\to 0^+} a^{-N}\mathcal{SH}_\varphi H(a, s_1, s_2, \mathbf{x}) = 0 \quad \text{for all} \quad N > 0.$$

- If $x_1 = s_1 = s_2 = 0$, then

$$\lim_{a\to 0^+} a^{-1}\mathcal{SH}_\varphi H(a, \bar{s}_1, \bar{s}_2, \mathbf{x}) \neq 0.$$

This means the continuous shearlet transform of H has rapid asymptotic decay as $a \to 0$, unless point $\mathbf{x}$ is on the plane, then $y_1 = 0$ and slopes s_1, s_2 correspond to the normal direction to the plane; For planes with arbitrary orientation whose normal vector is given as $(\sin\phi\cos\theta, \sin\phi\sin\theta, \cos\phi)$, the continuous shearlet transform will have rapid decay, except for $\mathbf{x}$ on the plane and (s_1, s_2) satisfying $s_1 = \tan\theta$, $s_2 = \cot\phi\sec\theta$.

In general, let Ω be a region in $\mathbb{R}^3$ with boundary denoted by $\partial\Omega$. We assume its boundary is smooth and has positive Gaussian curvature at every point. If $B = \mathbf{1}_\Omega$, then we know [6]:

- If $\mathbf{x} \notin \partial\Omega$, then

$$\lim_{a\to 0^+} a^{-N}\mathcal{SH}_\varphi B(a, s_1, s_2, s_2, \mathbf{x}) = 0 \quad \text{for all } N > 0.$$

- If $\mathbf{x} \in \partial\Omega$ and (s_1, s_2) does not correspond to the normal direction of $\partial\Omega$ at $\mathbf{x}$, then

$$\lim_{a\to 0^+} a^{-N}\mathcal{SH}_\varphi B(a, s_1, s_2, s_2, \mathbf{x}) = 0 \quad \text{for all } N > 0.$$

- If $\mathbf{x} \in \partial\Omega$ and $(s_1, s_2) = (\bar{s}_1, \bar{s}_2)$ corresponds to the normal direction of $\partial\Omega$ at $\mathbf{x}$, then

$$\lim_{a \to 0^+} a^{-1} \mathcal{SH}_\varphi B(a, \bar{s}_1, \bar{s}_2, \mathbf{x}) \neq 0.$$

These background results establish that edge points can be located by analyzing the asymptotic rate of change of the magnitudes of the 3D shearlet coefficients as a function of scale. This means that the concepts developed in [19] for 2D can be extended and will be valid for 3D shearlets edge detection algorithms.

3.3 Edge Detection Algorithms

The first thing to consider before defining different three dimensional edge detectors is the input data that will be processed. With 2D methods, the edge detector operates on one image at time. Methods include Canny, Wavelet, and Shearlet based edge detectors. Three dimensional edge detectors will process a block of images as a batch. To extend the 2D approaches to three dimensions it is necessary to perform the analysis not only in the horizontal and vertical direction but also over time for a block of images. This section will present the algorithms defining the implementation for the 3D Canny, 3D Wavelet, and 3D Shearlet edge detectors.

3.3.1 3D Canny Edge Detection

The Canny edge detection algorithm begins with a smoothing step to reduce noise. This is accomplished with a 3D Gaussian low pass filter

$$\mathbf{g}_{\sigma}^{3D} = \exp\left(-(x^2 + y^2 + t^2)/(2 * \sigma^2)\right), \tag{3.16}$$

where σ is the variance of the distribution. For a given image sequence $\tilde{\mathbf{I}}$ the smoothed image sequence is

$$\tilde{\mathbf{I}}_s = \tilde{\mathbf{I}} * \mathbf{g}_{\sigma}^{3D}. \tag{3.17}$$

After smoothing to remove high frequency noise comes the important image gradient computation. The image gradient reveals the differences in intensity between neighboring pixels and is the foundational operation for edge detection. Our 3D Canny uses a derivative of the Gaussian function. The derivative in the x direction is

$$\mathbf{d}g_{\sigma,x}^{3D} = \frac{\partial \mathbf{g}_{\sigma}^{3D}}{\partial x} = Cx \exp\left(-(x^2 + y^2 + t^2)/(2 * \sigma^2)\right), \tag{3.18}$$

in the y direction is

$$\mathbf{d}g_{\sigma,y}^{3D} = \frac{\partial \mathbf{g}_{\sigma}^{3D}}{\partial y} = Cy \exp\left(-(x^2 + y^2 + t^2)/(2 * \sigma^2)\right), \tag{3.19}$$

and in the t direction is

$$\mathbf{d}g_{\sigma,t}^{3D} = \frac{\partial \mathbf{g}_{\sigma}^{3D}}{\partial t} = Ct \exp\left(-(x^2 + y^2 + t^2)/(2 * \sigma^2)\right) \tag{3.20}$$

where $C = -1/\sigma^2$ is a scalar. For a particular partial derivative, the exponential function will tend to dampen out or enhance the component in that direction according to the constant C. The constant C includes the variance σ^2 of the distribution

of intensities in its denominator. Therefore if we assume that the intensities in our image have a lot of variability from one pixel to the next, then the magnitudes of the derivatives will be smaller on average. This is the situation with high levels of Gaussian noise. Alternatively, if the variance in the image intensities is small, the magnitude of the gradient will tend to be larger on average. If the variance is chosen properly then the edges can be detected; otherwise the differences will be lost. Also the neighboring images in the block before or after may have a different intensity distribution than the current frame. Choosing the correct variance for edge detection is a fundamental problem for methods that are based on the derivatives computed with Gaussian functions.

The next step is to use the gradient components to compute the gradient magnitude

$$|\nabla\tilde{\mathbf{I}}| = \sqrt{\nabla\tilde{\mathbf{I}}_x^2 + \nabla\tilde{\mathbf{I}}_y^2 + \nabla\tilde{\mathbf{I}}_t^2}. \tag{3.21}$$

The 3D nonmaximal suppression step is then applied to each voxel in the sequence of the gradient magnitudes. Here a voxel is an element of the edge surface if it is greater in magnitude than at least one of its 13 neighboring pairs. A version of this was implemented but became impractical to use during testing because of the time to run the method. Also to save computational cost the same 2D hysteresis thresholding method described in Algorithm 1 is applied to each slice of the sequence $\nabla\tilde{\mathbf{I}}$ to compute the final edge result $\tilde{\mathbf{I}}_{\mathbf{E}}$. All of the edge detection methods use the thresholding procedure from Chapter 2. The edge detection procedure is summarized for one image sequence in Algorithm 5.

Algorithm 5 The 3D Canny algorithm.

Input: Raw image sequence $\tilde{\mathbf{I}}$

Output: Estimate of $|\nabla\tilde{\mathbf{I}}|$

Compute smoothed sequence by $\tilde{\mathbf{I}}_s = \tilde{\mathbf{I}}_s * \mathbf{g}^{3D}$.

Compute horizontal derivative with $\nabla\tilde{\mathbf{I}}_x = \tilde{\mathbf{I}}_s * \mathbf{dg}^{3D}_{\sigma,x}$.

Compute vertical derivative with $\nabla\tilde{\mathbf{I}}_y = \tilde{\mathbf{I}}_s * \mathbf{dg}^{3D}_{\sigma,y}$.

Compute time derivative with $\nabla\tilde{\mathbf{I}}_t = \tilde{\mathbf{I}}_s * \mathbf{dg}^{3D}_{\sigma,t}$.

Compute the gradient magnitude $|\nabla\tilde{\mathbf{I}}| = \sqrt{\nabla\tilde{\mathbf{I}}_x^2 + \nabla\tilde{\mathbf{I}}_y^2 + \nabla\tilde{\mathbf{I}}_t^2}$.

Apply nonmaximal suppression to gradient magnitudes.

Use hysteresis thresholding on each slice.

3.3.2 3D Wavelet and Shearlet Edge Detection

Our use of the 3D wavelet and 3D shearlet transforms for edge detection builds upon the algorithms developed in [19]. The horizontal, vertical, and time components of the transform are computed separately and then reassembled in the end. The first critical step for both the wavelet and shearlet processing is to compute the 3D gradient components of the image. The 3D gradient filter ($3 \times 3 \times 3$) is analogous to the Sobel filter that weights the central pixel the most. Stacking each plane of the operator side-by-side reveals the contents for the horizontal derivative

$$\mathbf{G}_x^{3D} = \left[\begin{array}{ccc|ccc|ccc} 0 & 0 & 0 & +1 & 0 & -1 & 0 & 0 & 0 \\ +1 & 0 & -1 & +2 & 0 & -2 & +1 & 0 & -1 \\ 0 & 0 & 0 & +1 & 0 & -1 & 0 & 0 & 0 \end{array}\right]. \tag{3.22}$$

The vertical 3D derivative is

$$\mathbf{G}_y^{3D} = \left[\begin{array}{ccc|ccc|ccc} 0 & +1 & 0 & +1 & +2 & +1 & 0 & +1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & -1 & -2 & -1 & 0 & -1 & 0 \end{array}\right], \tag{3.23}$$

and the derivative over time is

$$\mathbf{G}_t^{3D} = \left[\begin{array}{ccc|ccc|ccc} 0 & +1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 1 & 2 & 1 & 0 & 0 & 0 & -1 & -2 & -1 \\ 0 & +1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \end{array}\right]. \tag{3.24}$$

Our implementation of the 3D wavelet edge detector first computes the horizontal image derivative

$$\nabla\tilde{\mathbf{I}}_x = \tilde{\mathbf{I}} * \mathbf{G}_x^{3D}, \tag{3.25}$$

and the vertical image derivative

$$\nabla\tilde{\mathbf{I}}_y = \tilde{\mathbf{I}} * \mathbf{G}_y^{3D}, \tag{3.26}$$

and the image derivative over time

$$\nabla\tilde{\mathbf{I}}_t = \tilde{\mathbf{I}} * \mathbf{G}_t^{3D}. \tag{3.27}$$

The "continuous" scaling is then accomplished by convolving repeatedly a weighted average filter or mask $\mathbf{A} = \left(\begin{smallmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{smallmatrix}\right)$ extended into the third dimension to give more emphasis to the central pixels:

$$\mathbf{G}_a^{3D} = \left[\begin{array}{ccc|ccc|ccc} 1 & 1 & 1 & 1 & 2 & 1 & 1 & 1 & 1 \\ 1 & 2 & 1 & 2 & 4 & 2 & 1 & 2 & 1 \\ 1 & 1 & 1 & 1 & 2 & 1 & 1 & 1 & 1 \end{array}\right]. \tag{3.28}$$

Thus, we have

$$\mathbf{G}_x^{n,3D} = \mathbf{G}_x^{n-1,3D} * \mathbf{G}_a^{3D}. \tag{3.29}$$

A weight is computed by

$$p_n = \frac{\max\left(\mathbf{G}_x^{n-1,3D}\right)}{\max\left(\mathbf{G}_x^{n,3D}\right)} \tag{3.30}$$

to be multiplied by these approximate dilations to guarantee that the maximum of the wavelet components do not change. In summary, we multiply the smoothing operator $\mathbf{G}_a^{3D}$ by the weight p_n to compute the horizontal smoothed/scaled wavelet coefficients

$$\nabla\tilde{\mathbf{I}}_x^{(n)} = \nabla\tilde{\mathbf{I}}_x^{(n-1)} * p_n\mathbf{G}_a^{3D}, \tag{3.31}$$

the vertical scaled wavelet coefficients

$$\nabla\tilde{\mathbf{I}}_y^{(n)} = \nabla\tilde{\mathbf{I}}_y^{(n-1)} * p_n\mathbf{G}_a^{3D}, \tag{3.32}$$

and scaled wavelet coefficients over time

$$\nabla\tilde{\mathbf{I}}_t^{(n)} = \nabla\tilde{\mathbf{I}}_t^{(n-1)} * p_n\mathbf{G}_a^{3D}. \tag{3.33}$$

After each smoothing, store the dilated $\mathbf{G}_x^{n,3D}$ to compute p_n for the next scale. Finally, a cumulative horizontal component of the image gradient is estimated as

$$\left(\nabla\tilde{\mathbf{I}}_x^{(n)}\right)_i = \begin{cases} \left(\nabla\tilde{\mathbf{I}}_x^{(n-1)}\right)_i & \text{if } |\nabla\tilde{\mathbf{I}}_x^{(n-1)}|_i \leq |\nabla\tilde{\mathbf{I}}_x^{(n)}|_i \\ \left(\nabla\tilde{\mathbf{I}}_x^{(n)}\right)_i & \text{otherwise .} \end{cases} \tag{3.34}$$

The cumulative vertical component is estimated as

$$\left(\nabla\tilde{\mathbf{I}}_y^{(n)}\right)_i = \begin{cases} \left(\nabla\tilde{\mathbf{I}}_y^{(n-1)}\right)_i & \text{if } |\nabla\tilde{\mathbf{I}}_y^{(n-1)}|_i \leq |\nabla\tilde{\mathbf{I}}_y^{(n)}|_i \\ \left(\nabla\tilde{\mathbf{I}}_y^{(n)}\right)_i & \text{otherwise .} \end{cases} \tag{3.35}$$

and the cumulative component over time is estimated as

$$\left(\nabla\tilde{\mathbf{I}}_t^{(n)}\right)_i = \begin{cases} \left(\nabla\tilde{\mathbf{I}}_t^{(n-1)}\right)_i & \text{if } |\nabla\tilde{\mathbf{I}}_t^{(n-1)}|_i \leq |\nabla\tilde{\mathbf{I}}_t^{(n)}|_i \\ \left(\nabla\tilde{\mathbf{I}}_t^{(n)}\right)_i & \text{otherwise .} \end{cases} \tag{3.36}$$

This step accumulates the contribution from each scaling into the image gradient. The last step is to compute the image gradient magnitude by summing the squares of the horizontal, vertical, and time cumulative components

$$\left|\nabla\tilde{\mathbf{I}}\right|^2 = \nabla\tilde{\mathbf{I}}_x{}^{(n)^2} + \nabla\tilde{\mathbf{I}}_y{}^{(n)^2} + \nabla\tilde{\mathbf{I}}_t{}^{(n)^2}. \tag{3.37}$$

The 3D wavelet based procedure for a single image is summarized in Algorithm 6.

A different scale of the image is processed each time the image is convolved with the average filter and represents a different amount of smoothing. This is analogous to the Canny edge detection process of initially smoothing the image with a Gaussian function at a particular standard deviation. The drawback to the Canny method is choosing the correct standard deviation, as this can be image-specific. The 3D wavelet and 3D shearlet based methods, on the other hand, can accumulate the gradient information present at multiple scales and thus do not suffer from the same problems as a Canny detector. 3D shearlet directional filtering is accomplished by creating the appropriate frequency formed hyper-trapezoid filters. These directional components are specifically constructed by multiplying the 2D constructed directional components developed in [19] . The hypersphere is partitioned according to the number of search directions. For $m_d = 16$ directions, one instance of a partition is given in Figure 3.2. This filter directed along the vertical y-axis partitions the x-t plane at a given scale a. It is important to note that the extension of the 2D

Algorithm 6 The 3D wavelet algorithm.

Input: $\tilde{\mathbf{I}}$ **Raw image sequence**

Output: Estimate of $|\nabla\tilde{\mathbf{I}}|$

Compute basic horizontal derivative $\nabla\tilde{\mathbf{I}}_x$ with (3.25).

Compute basic vertical derivative $\nabla\tilde{\mathbf{I}}_y$ with (3.26).

Compute basic time derivative $\nabla\tilde{\mathbf{I}}_t$ with (3.27).

for scale $s = 1 \rightarrow n_s$ **do**

Compute the smooth current dilated gradient operator $\mathbf{G}_x^{n,3D}$ with (3.29).

Determine gradient scaling constant p_n using (3.30).

Smooth horizontal component $\nabla\tilde{\mathbf{I}}_x^{(n)}$ with (3.31).

Smooth vertical component $\nabla\tilde{\mathbf{I}}_y^{(n)}$ with (3.32).

Smooth time component $\nabla\tilde{\mathbf{I}}_t^{(n)}$ with (3.33).

Update smoothed gradient operator $\mathbf{G}_x^{n,3D}$ for next scale.

Accumulate horizontal component $\nabla\tilde{\mathbf{I}}_x^{(n)}$ with (3.34).

Accumulate vertical component $\nabla\tilde{\mathbf{I}}_y^{(n)}$ with (3.35).

Accumulate time component $\nabla\tilde{\mathbf{I}}_t^{(n)}$ with (3.36).

end for

Compute the gradient magnitude $\left|\nabla\tilde{\mathbf{I}}\right|$ using (3.37).

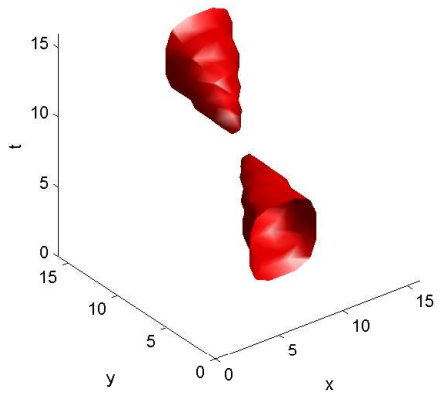


Figure 3.2: The horizontal partition for 3D shearlet filter.

methods into 3D presents some difficult issues. One of these issues is how to appropriately compensate for smoothing that is done in the horizontal, vertical, and time directions. We could follow the Canny algorithm strategy to isolate the edges by applying a thresholding step to both the wavelet and shearlet based edge nominations to further refine these nominations and help separate the edge information from noise. Next hysteresis thresholding could be applied to both wavelet and shearlet to further refine the edges and help separate edge information from noise. This process includes first setting all values above the high threshold T_2 to one and setting all values below the low threshold T_1 to zero. Those values between the high and low threshold are retained only if the pixel is connected to a pixel whose intensity is greater than the high threshold. This threshold method has proven satisfactory yet we are still in the process of developing better strategies.

Our implementation of the 3D shearlet edge detection algorithm first computes

the horizontal, vertical, and time components of the image gradients the same as equations (3.25), (3.26), and (3.27) for the wavelet. Before processing the image gradients, it is necessary to compute the 3D multi-scale, multi-directional shearlet filter bank. The first step is to generate the 2D horizontal shearlet filters at each scale according to the description in Chapter 2. The 3D shearlet filter for the t direction at a particular scale $a = 1$ partitioning the x-y plane is computed by stacking the 2D filters on top of each other over the time window size $n_w = 16$. The level one stacking for the t direction is given in Figure 3.3. Next stack the same y-t filters partition

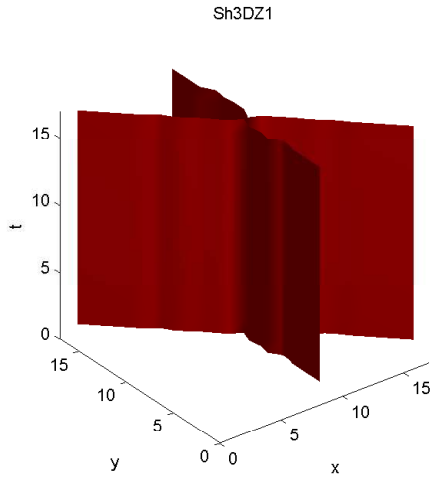


Figure 3.3: The x-y partition along the t-axis for 3D shearlet filter.

along the x-axis as shown in Figure 3.4. Finally to get one partition at a single scale a and slope s multiply element-by element to get the shearlet filter $w^{3D}_{a,s,t}$ at the first level in the t direction in Figure 3.5. The t oriented 3D shearlet filter partitions the x-y plane into pairs of cones at a particular scale over a window of size n_w . Larger scale values will result in a finer partition of the space into smaller cones. As the

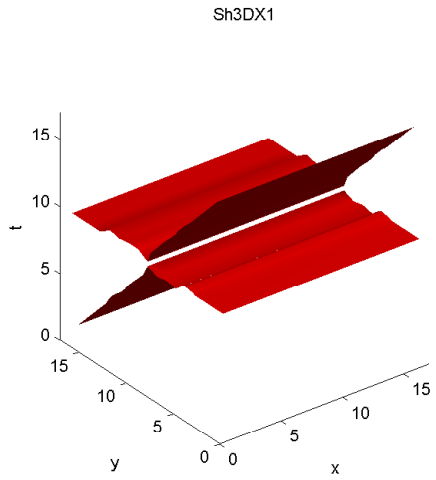


Figure 3.4: The y-t partition along the x-axis for 3D shearlet filter.

scale increases so does the number of directions and the frequency space partition is refined. This process must be repeated at each scale and slope direction for the x directed 3D shearlet filter $w^{3D}_{a,s,x}$, the y directed shearlet filter $w^{3D}_{a,s,y}$, and the t directed shearlet filter $w^{3D}_{a,s,t}$ before processing begins. To begin processing, at each scale use (3.29) and (3.30) to compute the nth scaling weight p_n. Apply (3.31), (3.32), and (3.33) to give the horizontal $\nabla\mathbf{I}_{x,a}$, vertical $\nabla\mathbf{I}_{y,a}$, and $\nabla\mathbf{I}_{t,a}$ dilated image derivatives by smoothing with $\mathbf{G}_{3D,a}$. After each smoothing store the dilated gradient $\mathbf{G}^n_{x,a}$ to compute p_n for the next scale. To compute the horizontal shearlet directional image derivatives two ancillary quantities are necessary including the shearlet convolved with the basic image gradient without smoothing and the image gradient after smoothing. For each scale a and slope direction s use the 3D horizontal $w^{3D}_{a,s,x}$, vertical $w^{3D}_{a,s,y}$, and time $w^{3D}_{a,s,t}$ directional shearlet filters to compute the basic

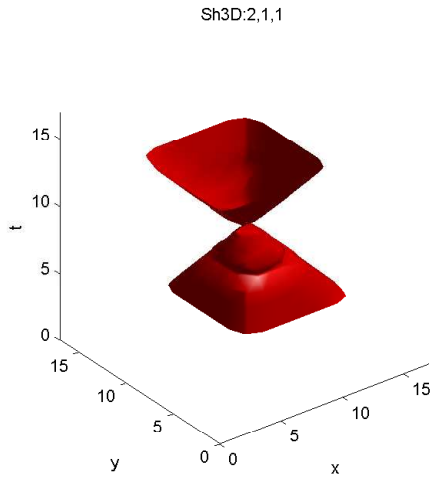


Figure 3.5: The level 1 partition for 3D shearlet filter along t-axis.

horizontal shearlet coefficients without scaling using

$$\nabla\tilde{\mathbf{I}}_{x,s} = \nabla\tilde{\mathbf{I}}_x * w^{3D}_{a,s,x}, \tag{3.38}$$

the basic vertical shearlet image derivative

$$\nabla\tilde{\mathbf{I}}_{y,s} = \nabla\tilde{\mathbf{I}}_y * w^{3D}_{a,s,y}, \tag{3.39}$$

and the basic time oriented shearlet image derivative

$$\nabla\tilde{\mathbf{I}}_{t,s} = \nabla\tilde{\mathbf{I}}_t * w^{3D}_{a,s,t}. \tag{3.40}$$

The resultant shearlet coefficients serve as a reference later to accumulate the edge locations at the current dilation and above. The smoothed horizontal shearlet image derivative is

$$\nabla\tilde{\mathbf{I}}_{x,a,s} = \nabla\tilde{\mathbf{I}}_{x,a} * w^{3D}_{a,s,x}, \tag{3.41}$$

the smoothed vertical shearlet image derivative is

$$\nabla\tilde{\mathbf{I}}_{y,a,s} = \nabla\tilde{\mathbf{I}}_{y,a} * w^{3D}_{a,s,y}, \tag{3.42}$$

and the smoothed over time shearlet image derivative is

$$\nabla\tilde{\mathbf{I}}_{t,a,s} = \nabla\tilde{\mathbf{I}}_{t,a} * w^{3D}_{a,s,t}. \tag{3.43}$$

For each direction s determine those horizontal smoothed shearlet coefficients that are larger than the basic horizontal shearlet coefficients

$$\nabla\tilde{\mathbf{I}}^{(n)}_{x_d} = \nabla\tilde{\mathbf{I}}^{(n)}_{x,a,s}. * (\left|\nabla\tilde{\mathbf{I}}^{(n)}_{x,a,s}\right| \geq \left|\nabla\tilde{\mathbf{I}}_{x,s}\right|), \tag{3.44}$$

the vertical smoothed shearlet coefficients that are larger than the basic vertical shearlet coefficients

$$\nabla\tilde{\mathbf{I}}^{(n)}_{y_d} = \nabla\tilde{\mathbf{I}}^{(n)}_{y,a,s}. * (\left|\nabla\tilde{\mathbf{I}}^{(n)}_{y,a,s}\right| \geq \left|\nabla\tilde{\mathbf{I}}_{y,s}\right|), \tag{3.45}$$

and the time smoothed shearlet coefficients that are larger than the basic time shearlet coefficients

$$\nabla\tilde{\mathbf{I}}^{(n)}_{t_d} = \nabla\tilde{\mathbf{I}}^{(n)}_{t,a,s}. * (\left|\nabla\tilde{\mathbf{I}}^{(n)}_{t,a,s}\right| \geq \left|\nabla\tilde{\mathbf{I}}_{t,s}\right|). \tag{3.46}$$

Add the larger horizontal shearlets to the cumulative horizontal shearlet directional sum

$$\nabla\tilde{\mathbf{I}}^{(n)}_{x_c} = \nabla\tilde{\mathbf{I}}^{(n-1)}_{x_c} + \nabla\tilde{\mathbf{I}}^{(n)}_{x_d}, \tag{3.47}$$

add the larger vertical shearlets to the cumulative vertical shearlet directional sum

$$\nabla\tilde{\mathbf{I}}^{(n)}_{y_c} = \nabla\tilde{\mathbf{I}}^{(n-1)}_{y_c} + \nabla\tilde{\mathbf{I}}^{(n)}_{y_d}, \tag{3.48}$$

and also add the larger time shearlets to the cumulative time shearlet directional sum

$$\nabla\tilde{\mathbf{I}}^{(n)}_{t_c} = \nabla\tilde{\mathbf{I}}^{(n-1)}_{t_c} + \nabla\tilde{\mathbf{I}}^{(n)}_{t_d}. \tag{3.49}$$

After accumulating coefficients in each direction, add the cumulative horizontal shearlet to the previous shearlet image derivative for the current scale

$$\nabla\tilde{\mathbf{I}}_x^{(n)} = \nabla\tilde{\mathbf{I}}_x^{(n-1)} .* (\left|\nabla\tilde{\mathbf{I}}_x^{(n-1)}\right| \leq \left|\nabla\tilde{\mathbf{I}}_{x_c}\right|) + \nabla\tilde{\mathbf{I}}_{x_c}^{(n)}, \tag{3.50}$$

the cumulative vertical shearlet image derivative for the current scale

$$\nabla\tilde{\mathbf{I}}_y^{(n)} = \nabla\tilde{\mathbf{I}}_y^{(n-1)} .* (\left|\nabla\tilde{\mathbf{I}}_y^{(n-1)}\right| \leq \left|\nabla\tilde{\mathbf{I}}_{y_c}\right|) + \nabla\tilde{\mathbf{I}}_{y_c}^{(n)}, \tag{3.51}$$

and the cumulative time shearlet image derivative for the current scale

$$\nabla\tilde{\mathbf{I}}_t^{(n)} = \nabla\tilde{\mathbf{I}}_t^{(n-1)} .* (\left|\nabla\tilde{\mathbf{I}}_t^{(n-1)}\right| \leq \left|\nabla\tilde{\mathbf{I}}_{t_c}\right|) + \nabla\tilde{\mathbf{I}}_{t_c}^{(n)}. \tag{3.52}$$

The last step is to compute the image gradient magnitude by summing the squares

$$\left|\nabla\tilde{\mathbf{I}}\right|^2 = \nabla\tilde{\mathbf{I}}_x{}^{(n)^2} + \nabla\tilde{\mathbf{I}}_y{}^{(n)^2} + \nabla\tilde{\mathbf{I}}_t{}^{(n)^2}. \tag{3.53}$$

The 3D shearlet based procedure for an image sequence is summarized in Algorithm 7.

Algorithm 7 The 3D shearlet algorithm.

Input: $\tilde{\mathbf{I}}$ Raw image sequence

Output: Estimate of $|\nabla\tilde{\mathbf{I}}|$

Compute horizontal,vertical, and time derivatives with (3.25), (3.26) and (3.27).

Compute shearlet filters before processing.

for $n = 1 \rightarrow n_s$ **do**

Smooth dilated gradient operator $\mathbf{G}_x^{n,3D}$ with (3.29).

Determine gradient scaling constant p_n using (3.30).

Compute $\nabla\tilde{\mathbf{I}}_{x,a}$ by smoothing horizontal component with (3.31).

Compute $\nabla\tilde{\mathbf{I}}_{y,a}$ by smoothing vertical component with (3.32).

Compute $\nabla\tilde{\mathbf{I}}_{t,a}$ by smoothing time component with (3.32).

Update smoothed gradient operator $\mathbf{G}_x^{n,3D}$ for the next scale.

for direction $d = 1 \rightarrow n_d$ **do**

Compute $\nabla\tilde{\mathbf{I}}_{x,s}$, $\nabla\tilde{\mathbf{I}}_{y,s}$, and $\nabla\tilde{\mathbf{I}}_{t,s}$ using (3.38), (3.39),and (3.40).

Compute smoothed $\nabla\tilde{\mathbf{I}}_{x,a,s}$, $\nabla\tilde{\mathbf{I}}_{y,a,s}$, $\nabla\tilde{\mathbf{I}}_{t,a,s}$ by (3.41), (3.42), and (3.43).

end for

Accumulate $\nabla\tilde{\mathbf{I}}_x^{(n)}$, $\nabla\tilde{\mathbf{I}}_y^{(n)}$,and $\nabla\tilde{\mathbf{I}}_t^{(n)}$ using (3.50),(3.51), and (3.52).

end for

Compute gradient magnitude $\left|\nabla\tilde{\mathbf{I}}\right|$ with (3.53).

3.3.3 Hybrid 3D Edge Detectors

The last two 3D algorithms to define are the hybrid wavelet and hybrid shearlet edge detectors. These methods use the more complicated 2D methods to process the image slices and the 3D Canny algorithm to process the image sequence over time. Our goal is to put more computational effort into the edge image in the x-y plane and save time by doing Canny 3D over time.

The hybrid 2D-3D wavelet method computes the horizontal 2D wavelet derivative $\nabla\tilde{\mathbf{I}}^{(n)}_{2D,x}$ and the vertical 2D wavelet derivative $\nabla\tilde{\mathbf{I}}^{(n)}_{2D,y}$ according to equations (2.25)-(2.32). The time image gradient $\nabla\tilde{\mathbf{I}}^{(n)}_{3D,t}$ is computed by first smoothing the image sequence with a Gaussian using equation (3.17). Then compute the derivative of the Gaussian in the time dimension using equation (3.20) like the Canny 3D algorithm. The final magnitude of the image gradient is

$$\left|\nabla\tilde{\mathbf{I}}\right|^2 = \nabla\tilde{\mathbf{I}}_x^{2D^2} + \nabla\tilde{\mathbf{I}}_y^{2D^2} + \nabla\tilde{\mathbf{I}}_t^{3D^2}. \tag{3.54}$$

The hybrid 2D-3D shearlet method computes the horizontal 2D shearlet derivative $\nabla\mathbf{I}^{(n)}_{2D,x}$ and the vertical 2D shearlet derivative $\nabla\mathbf{I}^{(n,2D)}_{y}$ according to the shearlet equations (2.29) and (2.30). The application of 2D shearlet filters to complete the computation of the shearlet image gradient components is the same as the 2D algorithm using equations (2.50)-(2.60). The time image gradient $\nabla\tilde{\mathbf{I}}^{(n,3D)}_{t}$ is computed by first smoothing the image sequence with a 3D Gaussian using equation (3.17). Then compute the derivative of the 3D Gaussian in the time dimension using equation (3.20) like the Canny 3D algorithm. The final magnitude of the image gradient

is

$$\left|\nabla\tilde{\mathbf{I}}\right|^2 = \nabla\tilde{\mathbf{I}}_x^{2D^2} + \nabla\tilde{\mathbf{I}}_y^{2D^2} + \nabla\tilde{\mathbf{I}}_t^{3D^2}. \tag{3.55}$$

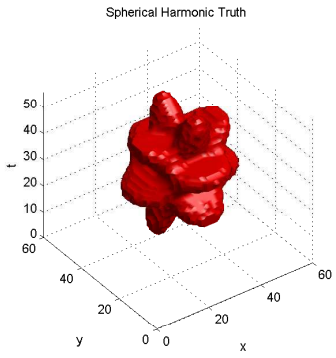


Figure 3.6: Spherical harmonic truth data

3.4 Experimental Results

In this section we compare the results of our 2D and 3D edge detection algorithms. In all experiments, only four scales were processed for wavelets and shearlet methods, as this proved adequate. Our experiments use synthetic data to better analyze the performance of the 3D shearlet transform compared to the 3D wavelet transform using known truth data.

3.4.1 Spherical Harmonic

Our first data set is a 3D image consisting of a solid spherical harmonic shape located in the center of a cube. The spherical harmonic functions are solutions of Laplace's equation

$$\nabla^2 V = 0 \tag{3.56}$$

for spherical coordinates. Our experiments used spherical harmonics of order 2 and degree 7, shown in Figure 3.6. Here there are 7 symmetrical structures with 2 lobes on each structure. These functions are useful because they describe rotation invariant structures for 3D surfaces and present good directionally oriented shapes to test the directional sensitivity of the routines.

The first experiment consists of using the 3D shearlet transform and the 3D wavelet edge detectors on the solid spherical harmonic, after adding identically distributed additive white Gaussian noise with a standard deviation $\sigma_n = 0.2$. We compare our results to those using the 2D edge detection algorithms. The results are displayed for the 2D and 3D Canny, wavelet, and shearlet edge detectors in Figure 3.7. At first glance it is apparent that the 3D methods produce a more complete representation of the surface with less missing information than the 2D methods. The 3D methods also have fewer spurious noise pixels far away from the true surface. At the highest noise level the 3D shearlet appears to best match the truth. The contour plots shown only display points with a resulting magnitude of 0.9 or greater. This presentation has a drawback in not displaying all of the noise present in the result.

The spherical harmonic truth data is known so it is possible to more precisely understand how well the methods compare with each other. Gaussian noise with the standard deviation $\sigma \in [0.0, 0.6]$ was added to the spherical harmonic. We ran the 2D and 3D edge detectors at the different noise levels and collected metrics to better measure detection performance. The metrics included the number of correctly identified edge pixels, the number of false positives, and the number of false negatives.

Method	Positive	False Positive	False negatives
Canny 2D	1392	4842	4906
Wavelet 2D	1434	3572	4864
Shearlet 2D	2234	3930	4064
Canny 3D	2062	2460	4236
Wavelet 3D	2259	2880	4039
Shearlet 3D	4741	6115	1557

Table 3.1: Edge statistics for 2D and 3D Canny, wavelet and shearlet edge detectors.

The truth contains 6,298 edge pixels. The results for $\sigma = 0.6$ are tabulated in Table 3.1.

The 3D methods identify more edge locations than the 2D methods due to the additional image gradient information over time. The statistics show that at high noise levels, both 2D and 3D shearlet methods have the largest number of correctly identified edges. Later we will see that shearlet edges are thicker, with fewer false positives far away from the truth than both the Canny and wavelet algorithms. To better see how the 2D and 3D methods compare for the spherical harmonic with increasing noise see Figure 3.8. The shearlet methods have the least number of false negative edges. The 2D Canny and 2D wavelet methods perform about the same, as do the 3D Canny and 3D wavelet methods.

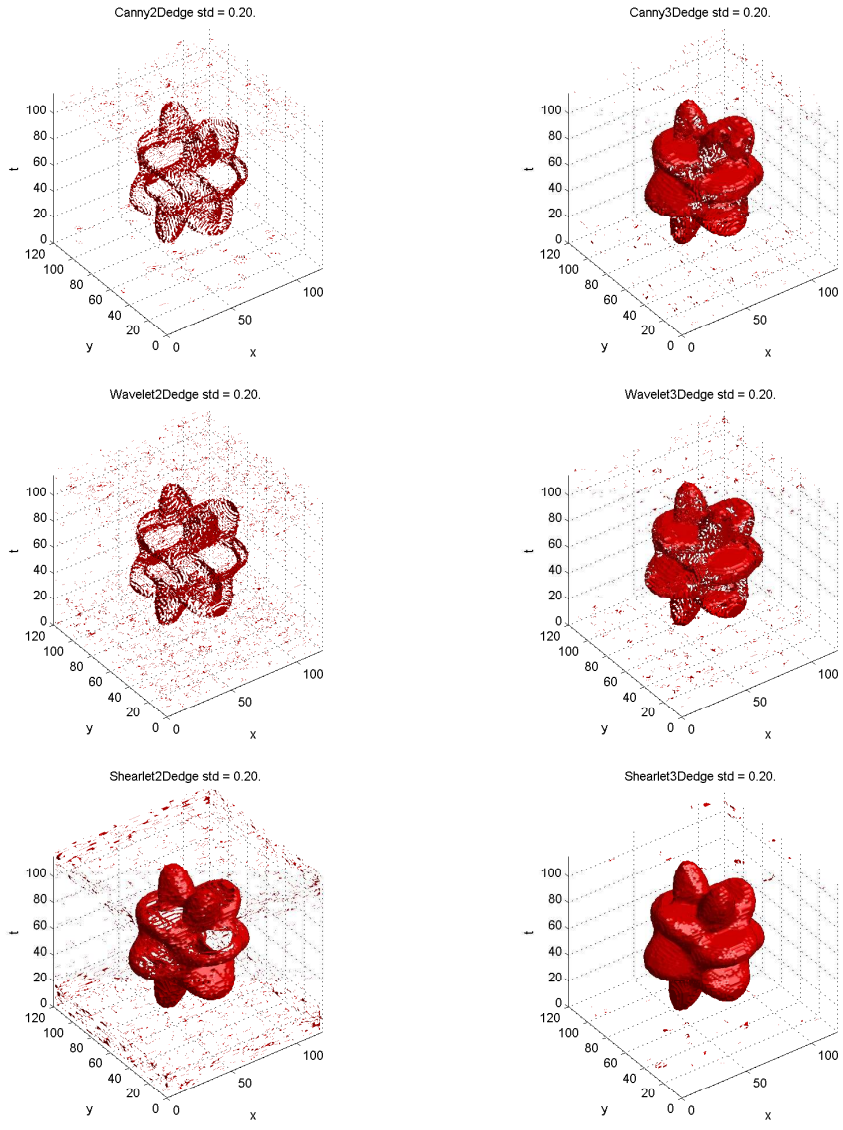


Figure 3.7: Noisy spherical harmonic for 2D (left) and 3D (right) algorithms.

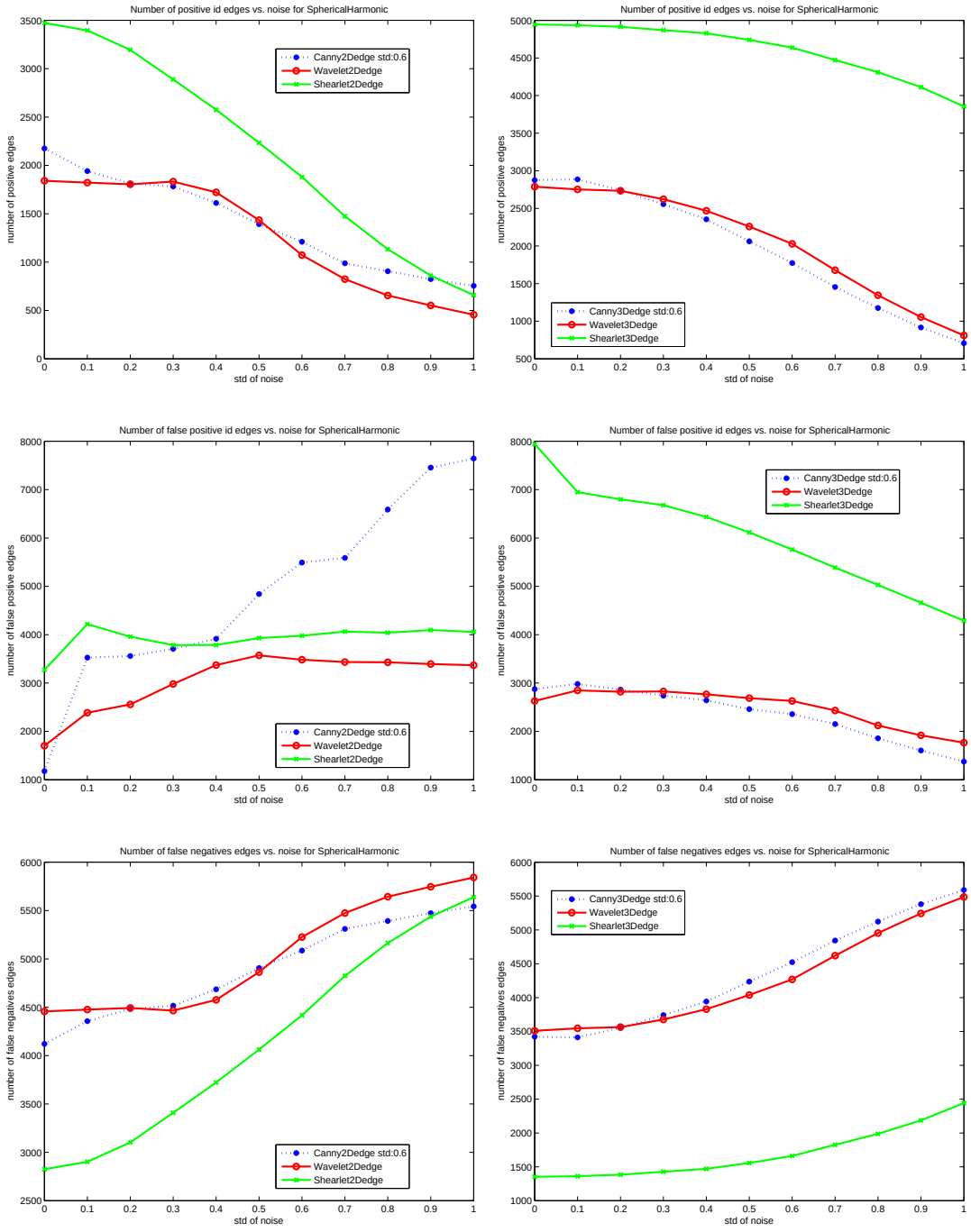


Figure 3.8: Positive identification of edges (top) false positives (middle) and false negatives(bottom) for 2D and 3D Canny, wavelet, and shearlet algorithms.

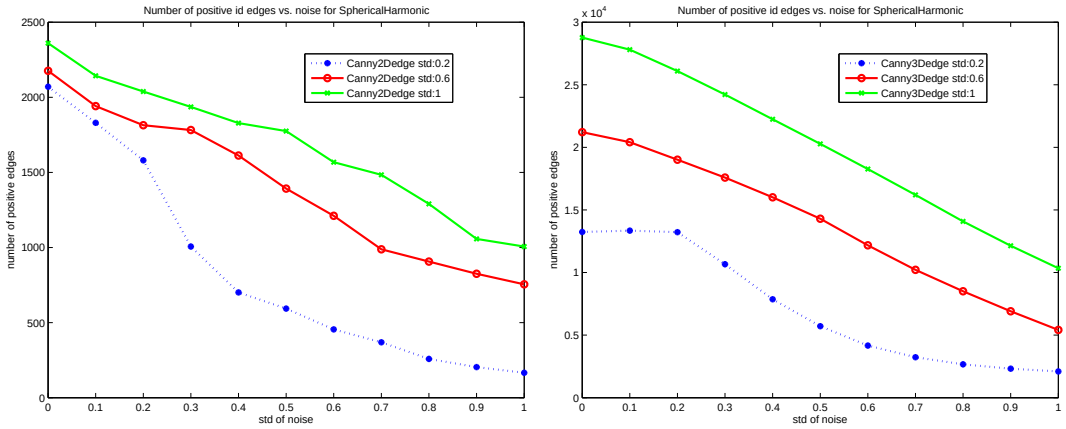


Figure 3.9: The performance of Canny using different smoothing levels, 2D (left) and 3D (right) on spherical harmonic. Average over 100 trials.

We also investigated the effect of the parameter σ_c used by the Canny methods to smooth the image. The results, averaged over 100 trials, are shown in Figure 3.9. Using a larger smoothing parameter helped.

To see how well the edges from the methods compare, consider three 2D slices from the spherical harmonic with noise $\sigma = 0.2$ in Figure 3.10. In this test, the results indicate that the 3D edge detection algorithms give a more complete representation of the surface in the presence of noise with fewer artifacts than their 2D counterparts. The shearlet transforms in particular perform better than the wavelet transforms as the noise level increases at the cusp points of the image slice. The Canny based edge detection methods have the most difficulty with the spherical harmonic test with noise. The 3D shearlet has the least number of spurious edge locations far from the true edge.

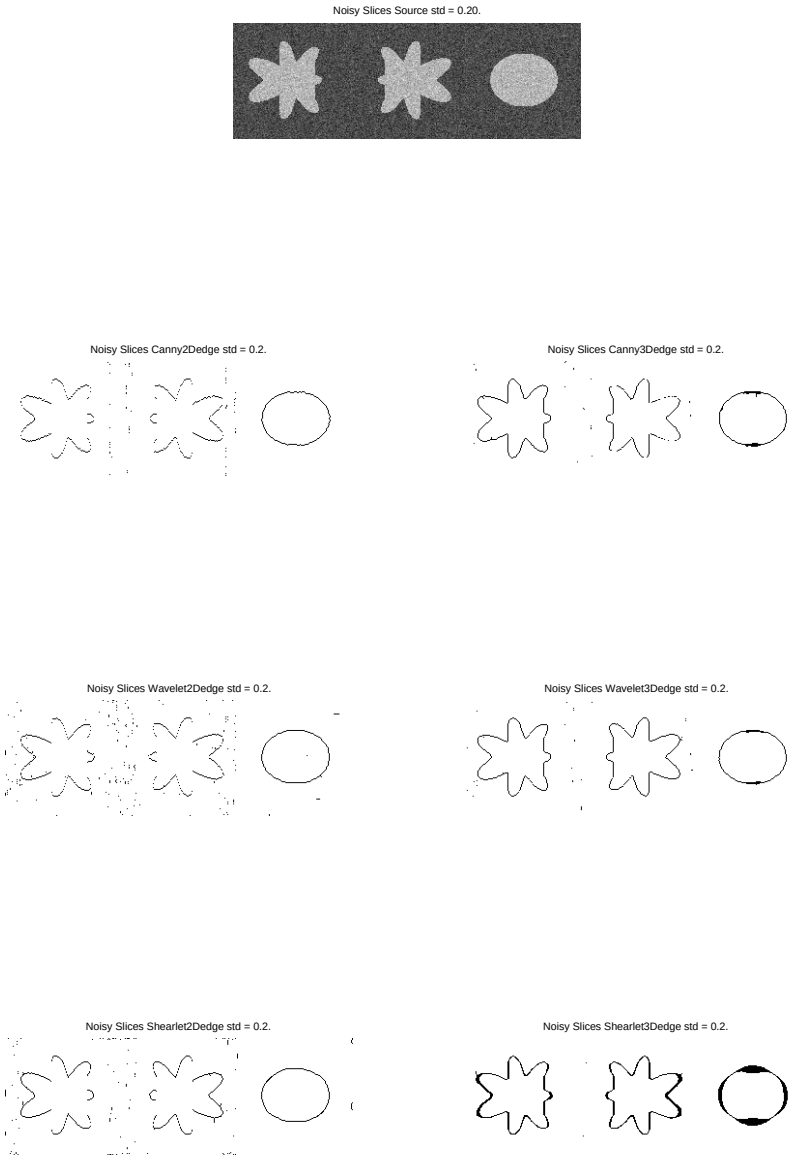


Figure 3.10: Slice spherical harmonics with noise for 2D (left) and 3D (right) for Canny (row 1), wavelet (row 2), and shearlet (row 3) routines.

The next experiment involves tracking a solid moving disk of known trajectory through an image sequence. This solid 2D disk spirals around the origin, with position

$$x_j = \lfloor r\cos(\alpha_j) \rfloor, \tag{3.57}$$

$$y_j = \lfloor r\sin(\alpha_j) \rfloor, \tag{3.58}$$

in frame j, where r is the radius of the spiral and α is an angle. A few frames of the image sequence are shown in Figure 3.11. Both 2D and 3D transforms are

Figure 3.11: Disk spiraling without noise

applied to this sequence of images to detect the edges and the results are displayed in Figure 3.12. The same spiraling disk with noise is given in Figure 3.13. The surfaces show again that the 3D algorithms provide better surface representation than the 2D methods. The 2D wavelet has more missing information than the other methods. The shearlet methods outperformed the wavelet methods.

Consider examining a slice from the 3D Canny detector with no noise given in Figure 3.14. Note the apparent spreading from the 3D Canny transform. At first glance, it is not apparent how this bulky edge could provide any useful information at all for tracking purposes. This is especially difficult if our rigid tracking requirement is to accurately determine the exact center of an image feature. To make sense of this problem, consider performing a 2D wavelet followed by a 3D Canny in the time

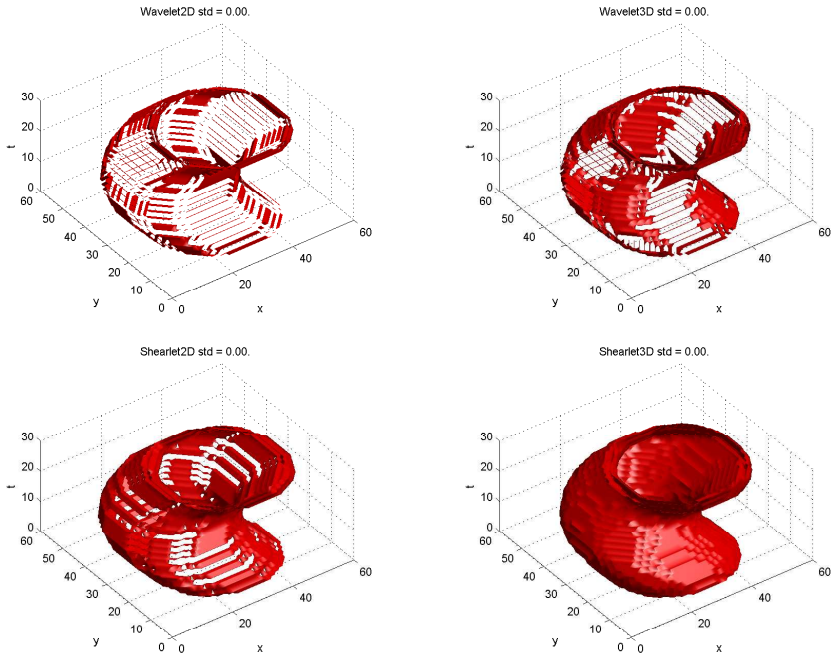


Figure 3.12: Results of disk spiraling surface detected for 2D (left) and 3D (right) for wavelet (row 1) and shearlet (row 2) algorithms *without* noise added to data.

direction given in Figure 3.15. This interesting figure provides insight into how the velocity component is automatically folded into the 3D edge detected result and helps to explain the directional spreading apparent in all 3D results. Summing the magnitudes of the image gradients resulted in the thick edge. What appeared to be an unsatisfactory bulge in the image is actually useful velocity information. The bright part of the Canny 3D image shows that the disk is moving in the southwest direction at a magnitude proportional to the thickness of the edge. Another thick edge accumulated over multiple scales by the 3D wavelet is given in Figure 3.16. The bright edge on the left and the darker edge on the right indicate that the disk is moving to the left at a magnitude proportional to the thickness of the edge.

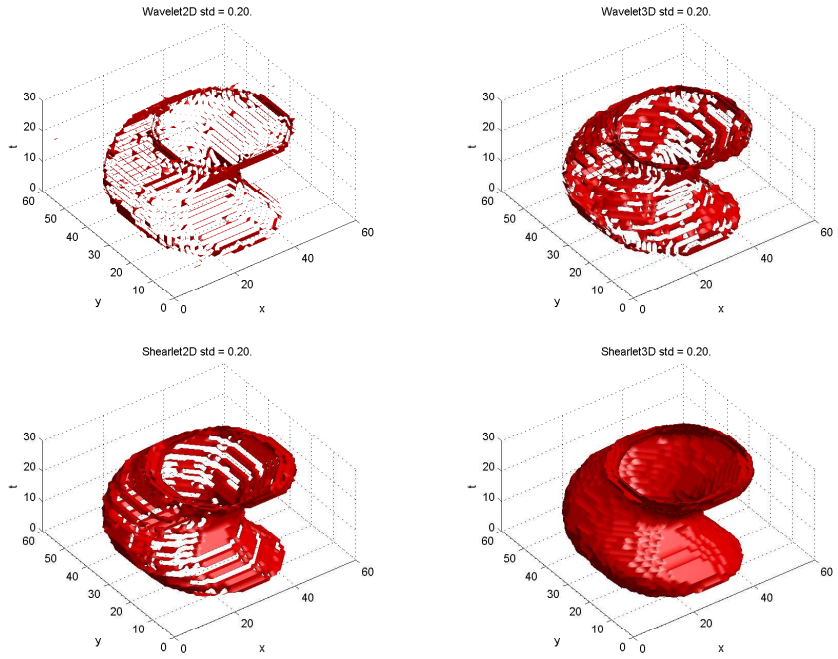


Figure 3.13: Results of disk spiraling surface detected for 2D (left) and 3D (right) for wavelet (row 1) and shearlet (row 2) algorithms *with* noise added to data.

The additional image velocity magnitude and directional information is not available from 2D edge detection algorithms and provides incentive to apply 3D edge detection to tracking in Chapter 4.

3.5 Conclusions

We have demonstrated the value of applying multi-scale and multi-directional transforms to detect edges in a sequence of images. Clearly extending the traditional 2D wavelet and shearlet transforms to 3D has provided more information and improved the detection performance. 3D edge detectors have more positively identified edges and higher PSNR values than 2D algorithms as noise increases. In

Figure 3.14: Slice from disk spiraling without noise for Canny 3D.

Figure 3.15: Slices from a disk spiraling without noise for Wavelet 2D x-y slice (left), Canny 3D over time (middle) and the sum of Wavelet 2D and Canny 3D (right).

particular, the 3D shearlet algorithm is especially effective for more complicated shapes because it takes edge direction into account. The 3D surface membrane detected is thicker and will be a more prominent feature compared to 2D results derived from noisy image sequences. The 3D detectors have thicker edges from the convolution over time that adds a velocity component to an edge slice. The image velocity is an attractive feature to use as an extra measurement for future integration into 3D Bayesian state estimation measurement models that currently only use edge locations. We anticipate further improvements in performance as we devise better compensation methods for the directional spread we noted earlier.

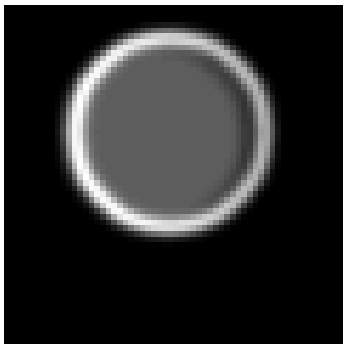


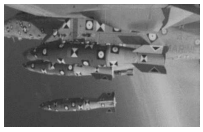
Figure 3.16: Slice from disk spiraling without noise for Wavelet 3D.

Chapter 4

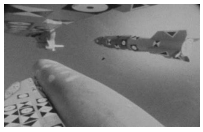
Tracking Objects Using Three Dimensional Edge Detection

4.1 Introduction

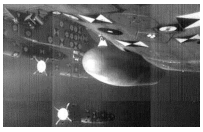
One important application of edge detection is segmentation of images to highlight patches of pixels called image features. This chapter will discuss algorithms to integrate edge detection into a tracking system for image feature estimation and explain what is new about the approach. Usually this problem arises when investigating an event that takes place in three dimensions while cameras record observations of objects that are moving in the field of view. The two fundamental factors that drive tracking complexity are complicated object shape and motion. For complicated shapes, multiple points on the object must be selected in order to accurately fit the geometry and represent motion through space. Each of these points, called targets, will correspond to an image feature as the object is tracked with a camera. An example is shown in Figure 4.1.



Wing tip camera



Pylon camera



Tail camera.

Figure 4.1: Images of track objects taken by cameras on an airplane

It is important to have enough targets uniformly distributed over the entire surface of the object to reliably represent the true structure of the 3D object. For example, if the object has fins or sharp curves, targets must be placed on the object closer together to produce good observations.

The three dimensional world also introduces noise, clutter, and illumination variations. These change the image intensities in complicated ways. Smoothing the image usually takes care of high frequency noise that is normally distributed throughout the image. However it is difficult to set the standard deviation of the filter to smooth noise without eliminating important edge information. Even when smoothing works for one image, the world can change, and the next image could require different smoothing. Also, not all noise is uniformly or normally distributed throughout the image, adding a further complication. What may eliminate noise in one region of the image may eliminate important information that is part of the track object in another region of the image. Next, background and dynamic clutter that mingles with true image edge features is not constant from frame to frame and must be accounted for to achieve accurate detection. Simple smoothing and image gradients are not enough to separate out important edge features in these cases. Edges by definition are defined by both magnitude and direction. Traditional edge detectors such as Sobel determine the magnitude of image gradients but do not determine the direction of the edge. More sophisticated algorithms like the Canny method account for only a few directions and have the same threshold choice problem. Illumination variations produce edges with gradients that are very difficult to discern. When most of the pixel intensities are near their average value, the

problem is particularly difficult. The changes in intensity from background to the track object is very small in shade or gets washed in bright spots. Areas of the image feature that fade gradually into the background are hard to detect. Even after noise is removed the directional edge information has small magnitude and can be thresholded out for certain directions.

In this chapter we present new 2D and 3D algorithms for tracking objects. What is new about our 2D tracking system is the use of multi-scale wavelet and multi-directional shearlet filter banks to get better feature detection in the presence of noise. What is truly different with our three dimensional tracking is the capability to process the track object over a fixed time interval to define a new surface detection method that is more robust to illumination change. Section 4.2 discusses the problem and the data. Section 4.3 presents our algorithms. Section 4.4 compares the different methods using metrics that help to evaluate the degree of success for different test cases. This presentation closely follows that in [15].

4.2 Problem Definition and Test Data

Developing a 2D tracking system using edge detection involves a description of the input data image sequence and how it relates to the problem to be solved. The tracking problem definition starts with the observed image sequence that captures 2D information about 3D objects that are in motion in the camera's field of view. At the lowest level, each image in the sequence provides a snapshot of a particular track object that moves from frame to frame. We present two test problems that

illustrate some of the difficulties.

Figure 4.2: Four frames from the spiraling ball movie.

Figure 4.3: Patch containing the disk (left) is inserted into a frame of the movie (right).

For our first test problem, suppose that a 3D ball spirals about a fixed axis for a period of several seconds. A camera located on that axis records a sequence of images, with the center of the images also positioned on the axis. Each image looks like a white disk on a black background, moved via translation. Four frames from the resulting spiraling ball movie are shown in Figure 4.2. The disk is chosen to have a diameter equal to an odd number of pixels so that in generating the data we can center it on the nearest pixel. The movie $\hat{\mathbf{I}}$ is stored in an $m \times m \times \ell$ array, with $m^2 = 157^2$ pixels per frame and $\ell = 30$ frames. The center (x_j, y_j) of the disk

at time t_j, relative to the center of the image, is defined to be

$$x_j = \lfloor r\cos(\alpha_j)\rfloor, \tag{4.1}$$

$$y_j = \lfloor r\sin(\alpha_j)\rfloor, \tag{4.2}$$

where r is the radius of the disk and α_j is the angle defining the position of the object relative to its position at time 0. We generate the frames of the movie by inserting a $(2r+3)\times(2r+3)$ patch of pixels containing the disk into a black (zero) frame of size $m \times m$, as shown in Figure 4.3. White noise (independent normally distributed samples for each pixel, with standard deviation $\sigma \in [0, 1.5]$) is added to the movie to create a noisy movie of the ball.

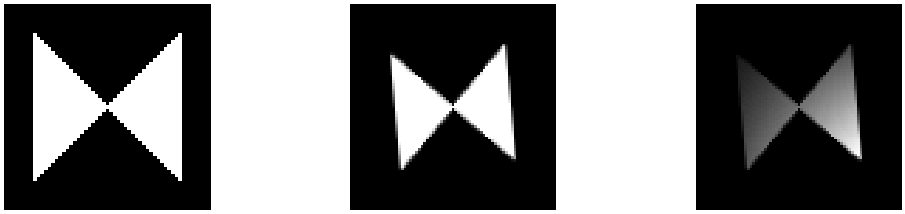


Figure 4.4: Patch containing a bow-tie (left), a rotated bow-tie (middle), and a shaded bow-tie (right).

Our second test problem is generated in a similar way, but uses a bow-tie patch, shown in Figure 4.4 (left), that spirals about the center of the image sequence but also rotates about its own center point, as illustrated in Figure 4.4 (middle). To perform rotation, we use Matlab's `imrotate`, which remaps each pixel in the patch to its rotated position using bilinear interpolation. We also use this example to investigate changes in illumination. This is accomplished by generating a row

Figure 4.5: Four frames from a bow-tie movie with spiraling movement, rotation, and illumination changes.

vector $\mathbf{g}$ of increasing values in the range $[0.05, 2]$ with dimension equal to that of the patch. The illumination matrix is then defined as

$$\mathbf{L} = \mathbf{g}^T \mathbf{g}. \tag{4.3}$$

The shaded object is obtained by elementwise multiplication of the patch $\mathbf{P}$ by $\mathbf{L}$:

$$\mathbf{S} = \mathbf{P}. * \mathbf{L}. \tag{4.4}$$

This is performed after rotation and produces a result like that shown on the right in Figure 4.4. To study the robustness of our algorithms, we add white noise to the bow-tie movie, because all of the tracking methods are able to match truth for data without noise. We will assume that we know the position of the object in the first frame of the movie. We use our methods to estimate the position of the center of the ball or bow-tie and, for the bow-tie, its rotation angle, as a function of time. It is useful (but more difficult) to estimate the velocity of the object, too.

4.3 Tracking Algorithm

A tracking algorithm must determine the trajectory of the track object as it moves from frame to frame. For simplicity, we consider translation first and discuss object rotation later. We assume that we are trying to determine the movement of the object between two particular frames, frame $i-1$ and frame i. We denote the displacement as Δx_i in the horizontal direction and Δy_i in the vertical direction and drop the subscript i when it is clear from context. Our first approach is to perform an exhaustive search for Δx and Δy by considering all possible positions of the patch and testing to see which trial position best matches the data from the movie. In practice, velocity bounds can be used to limit the search, and in this study we only test integer values between -2 and 2, giving 25 possible positions. For each trial position, we have two sets of data: $D(\tilde{\mathbf{I}})$, which is the data from the original movie $\tilde{\mathbf{I}}$, and $D(\tilde{\mathbf{I}}_p)$, where $\tilde{\mathbf{I}}_p$ is the movie $\tilde{\mathbf{I}}$ with the ith frame replaced by one with the patch in its trial position. The most obvious choices for the function D is $D(\tilde{\mathbf{I}}) = \tilde{\mathbf{I}}$. In this case we are interested in how the pixel values change when we replace the ith frame by the patched frame. This approach is quite sensitive to noise, however, and the preservation of important features, such as edges, is not guaranteed. We propose, therefore, that D denote the output from one of our edge detectors, 2D or 3D. In this case we are measuring how much the edges change between the original movie and the patched movie. We generate the patched frame in the same way we generate our test examples, by overwriting pixels in the ith frame by the patch, positioned by Δx and Δy relative to its position in the previous

frame. To determine the match between the trial position and the observed position, we use a cost function

$$f(\tilde{\mathbf{I}}_p) = \|D(\tilde{\mathbf{I}}) - D(\tilde{\mathbf{I}}_p)\|. \tag{4.5}$$

A natural choice is the Euclidean norm, but other choices are possible. Rather than running the edge detector on the complete movie for each trial position of the patch, we use a small number of frames surrounding Frame i. This reduces the cost of each trial. If the object is also rotating, then we need to measure the cost function at various values of Δx, Δy, and $\Delta\theta$, the change in rotation angle since the previous frame. In our experiments, we tested values $\Delta\theta = -2, -1, 0, 1, 2$ degrees, making a total of 125 possible positions and rotations per frame. Increasing the possible values of the Δ quantities quickly raises the expense of the exhaustive search algorithm. More sophisticated numerical optimization algorithms (steepest descent, Newton-like methods) can be used, but since our functions are noisy and highly nonconvex, we did not have much success with them. One advantage of our admittedly primitive optimization approach is that it is quite easy to parallelize. We found that our methods worked better if we added noise to the patch, comparable to that in the image sequence, before inserting it into the ith frame of the movie. We summarize our tracking method in Algorithm 8 .

From the computed Δ values, we can compute the magnitude of the planar velocity of the object at frame i,

$$|v_i| = \sqrt{\Delta x_i^2 + \Delta y_i^2}, \tag{4.6}$$

Algorithm 8 The tracking algorithm using edge detection.

Input: Image sequence $\tilde{\mathbf{I}}$, noise estimate, patch $\mathbf{P}$, and initial patch location.

Output: Estimates of patch position Δx, Δy, and $\Delta\theta$ for each frame.

Add noise to the patch $\mathbf{P}$.

for $n = 2 \rightarrow l$ **do**

Record $\Delta x_n = \Delta y_n = \Delta\theta_n = 0$ as the best guess so far.

for $d\theta = -2 : 1 : 2$ **do**

Rotate the patch by angle $d\theta$: $\mathbf{P}_r = \texttt{imrotate}(\mathbf{P}, d\theta)$.

Compute shaded patch $\mathbf{S} = \mathbf{P}_r . * \mathbf{L}$ from equation (4.3).

for $dx = -2 : 1 : 2$ **do**

for $dy = -2 : 1 : 2$ **do**

The current trial center is the patch center at frame $n-1$ plus (dx, dy).

Replace frame n of the image sequence $\tilde{\mathbf{I}}$ with a frame containing the patch $\mathbf{S}$ at the trial center, obtaining $\tilde{\mathbf{I}}_p$.

if $\|D(\tilde{\mathbf{I}}) - D(\tilde{\mathbf{I}}_p)\|$ is smaller than all previous values for frame n **then**

Set $\Delta x_n = dx$, $\Delta y_n = dy$, and $\Delta\theta_n = d\theta$.

end if

end for

end for

end for

Replace the patch $\mathbf{P}$ by rotating it by $\Delta\theta_n$.

end for

and the direction of the planar velocity,

$$\phi_i = \arctan\left(\frac{\Delta y_i}{\Delta x_i}\right). \tag{4.7}$$

The computed direction ϕ_i is quite sensitive to errors in the Δ values.

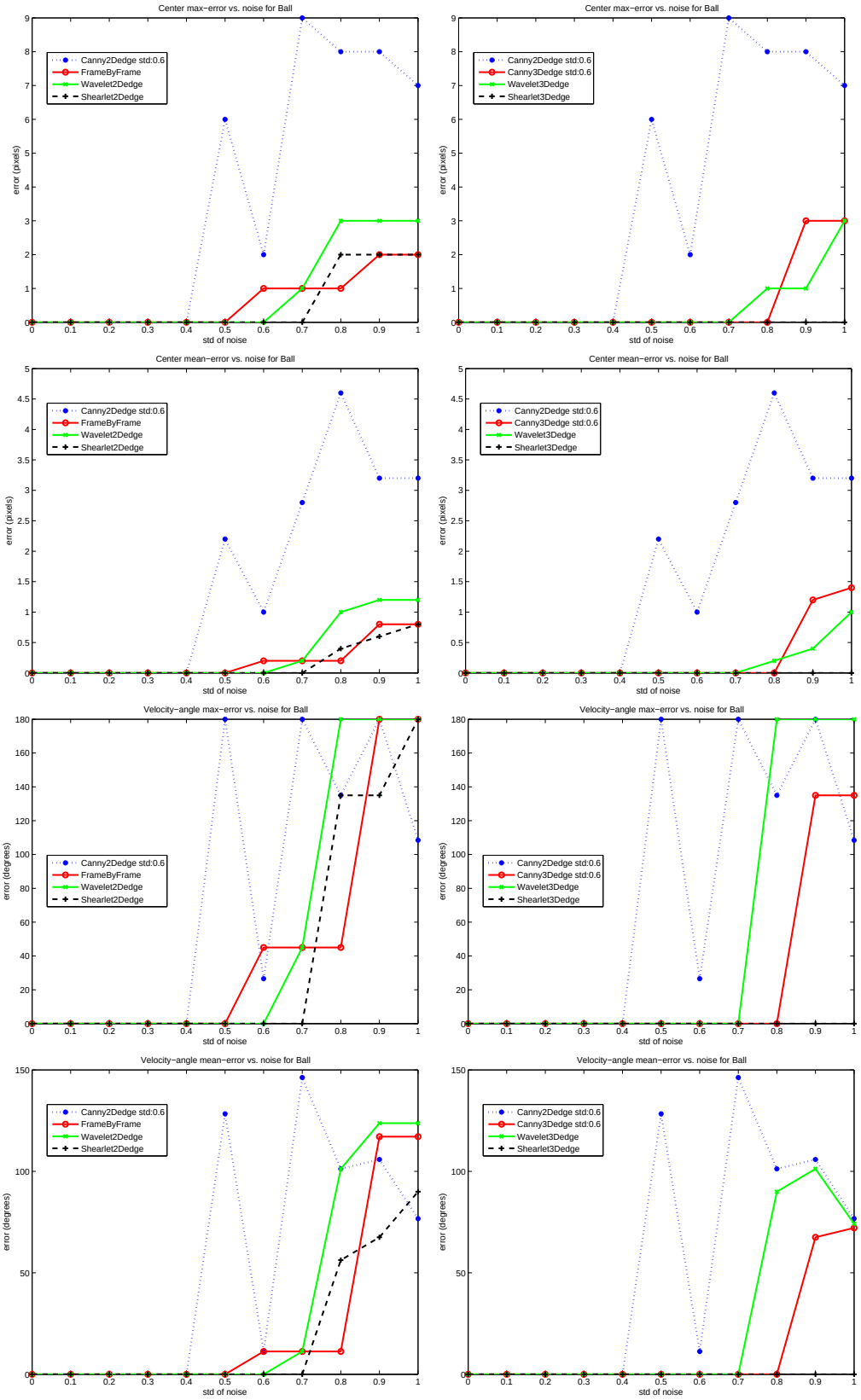


Figure 4.6: Results for spiraling ball with noise.

4.4 Experimental Results

Several experiments were conducted to help understand and characterize how edge detectors will work in the tracking context.

The first uses the spiraling ball movie (see Figure 4.2). In three dimensions this motion resembles a twisting tube or slinky. The optimization solves for the track object's estimated center location and velocity using the inserted artificial track object with and without edge detection methods. The spiraling ball movie is processed using the 3D Canny, 3D wavelet, 2D shearlet-Canny combination, and raw frame-by-frame without edge detection.

The exhaustive search is able to find the exact center and velocity of the track object for the frame-by-frame method without edge detection when no noise is present. Therefore a more difficult test is to compare methods by starting with no noise and gradually increasing the standard deviation $\sigma = [0.1, \ldots, 0.6]$ of the Gaussian noise until a particular method begins to stray from the truth data. This allows us to visualize the magnitude of the largest error as a function of increasing noise.

Our results compare the algorithms with respect to four metrics. The first metric, center max-error, is computed by finding the maximum over a set of frames of the one-norm of the difference between the true center and the computed center:

$$\max_{n=1,\ldots,\ell} \left\| \mathbf{x}_n^{true} - \mathbf{x}_n^{comp} \right\|_1 , \tag{4.8}$$

where $\mathbf{x}_n$ is the position of the patch in frame n. We chose $\ell = 5$. The second

metric, center mean-error, is

$$\frac{1}{\ell}\sum_{n=1}^{\ell}\left\|\mathbf{x}_n^{true}-\mathbf{x}_n^{comp}\right\|_1. \tag{4.9}$$

The errors in the estimated velocity magnitude of the track object can be inferred from these quantities. Our third metric, velocity-angle max-error, is the maximum error in the velocity angle of the track object:

$$\max_{n=1,\ldots,\ell}|\theta_n^{true}-\theta_n^{comp}|. \tag{4.10}$$

Similarly, velocity-angle mean-error is

$$\frac{1}{\ell}\sum_{n=1}^{\ell}\max|\theta_n^{true}-\theta_n^{comp}|. \tag{4.11}$$

Our results for the spiraling ball are shown in Figure 4.6. In this and subsequent figures, for clarity, we plot results for the 2D algorithms on the left and the 3D algorithms on the right. The plots show that for low error ($\sigma \leq 0.4$), all of the 2D and 3D algorithms perform perfectly. For error levels above that, Canny 2D breaks first, followed by the 2D wavelet at $\sigma = 0.5$. None of the 3D algorithms break track for this problem.

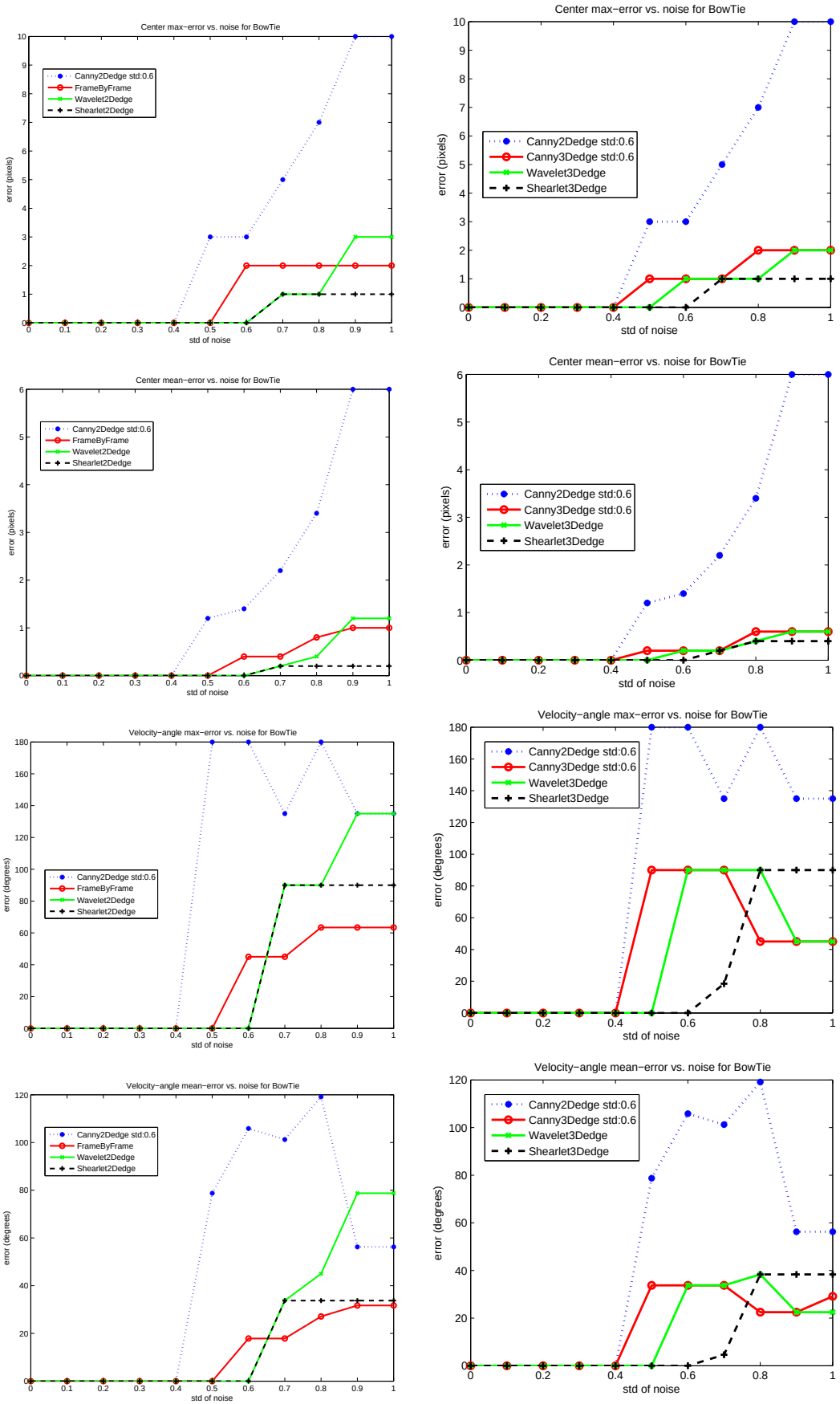


Figure 4.7: Results for spiraling bow-tie with noise.

Our second experiment evaluates how the edge detection algorithms perform on more complicated objects. Consider a black and white bow-tie target as shown in Figure 4.4. It has a corner point in the center where the image gradient changes direction rapidly over a very short distance and the edges run closer together. Our results for the bow-tie movie are displayed in Figure 4.7. For the bowtie problem the point of breaking track takes place for Canny 2D at noise standard deviation $\sigma = 0.4$, compared to the 2D wavelet at $\sigma = 0.6$. The 2D algorithms have larger error than the 3D algorithms.

Our third experiment considers a ball that has been shaded gradually from dark to light along the main diagonal, with results shown in Figure 4.8. Results are similar.

The fourth experiment considers a bow-tie that has been shaded gradually from dark to light along the main diagonal. The results are given in Figure 4.9. For this more complicated shaded bow-tie object, all 3D methods break track but maintain a center position tracking error of less than 3 pixels. The 2D shearlet generally has lower error than the 2D wavelet. The 3D shearlet and 3D wavelet perform slightly better than Canny3D.

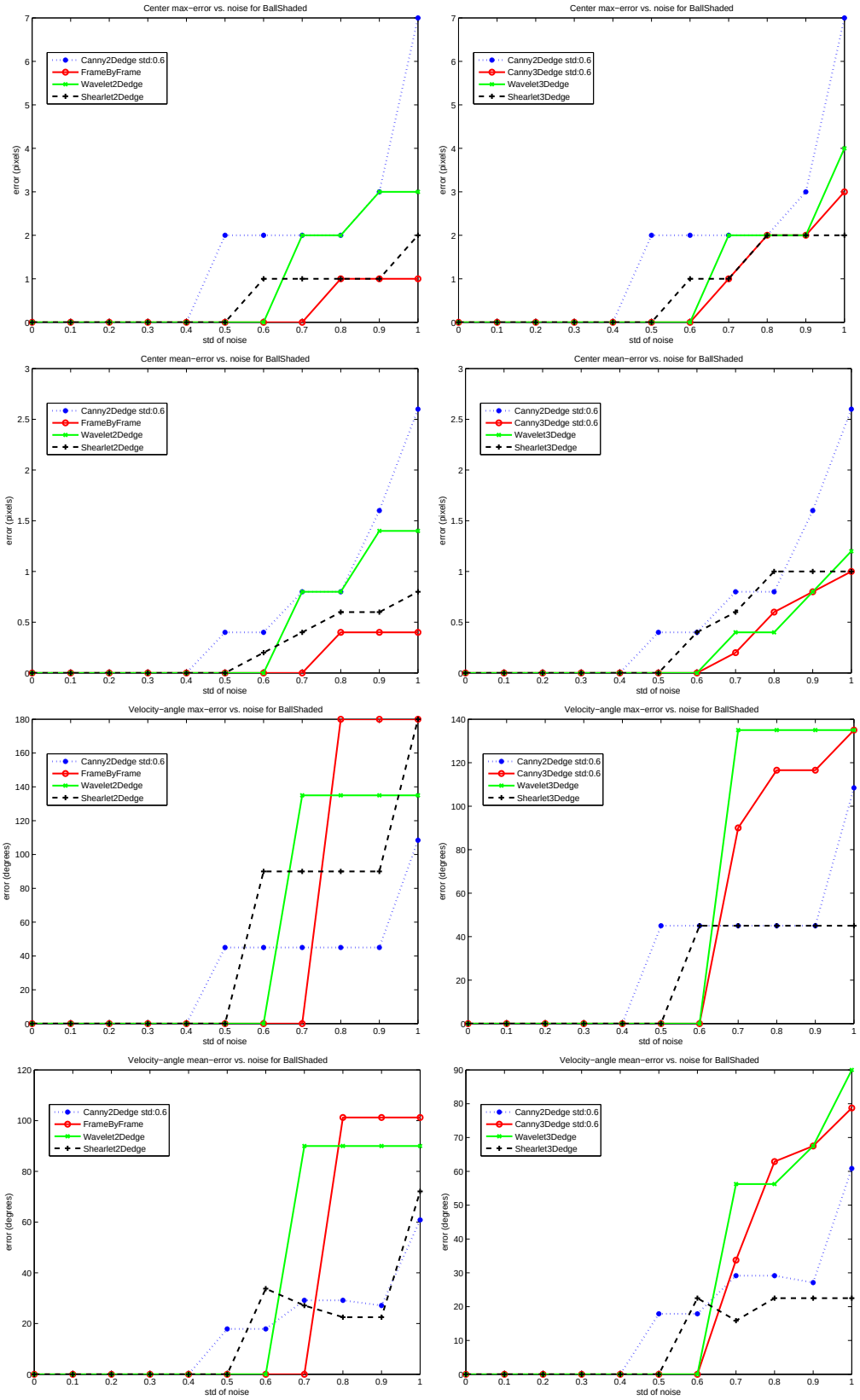


Figure 4.8: Results for the spiraling shaded ball with noise

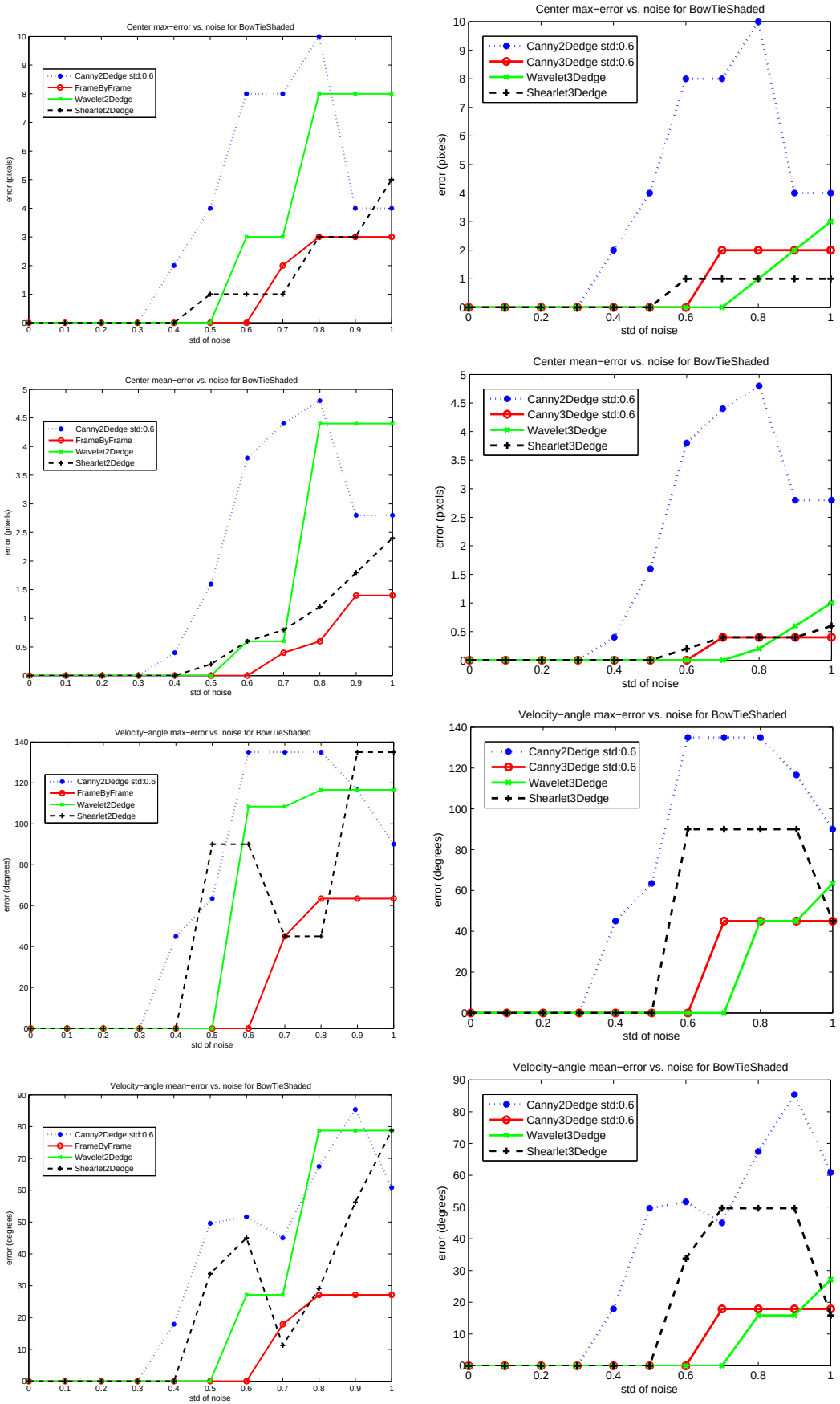


Figure 4.9: Results for the spiraling shaded bow-tie with noise.

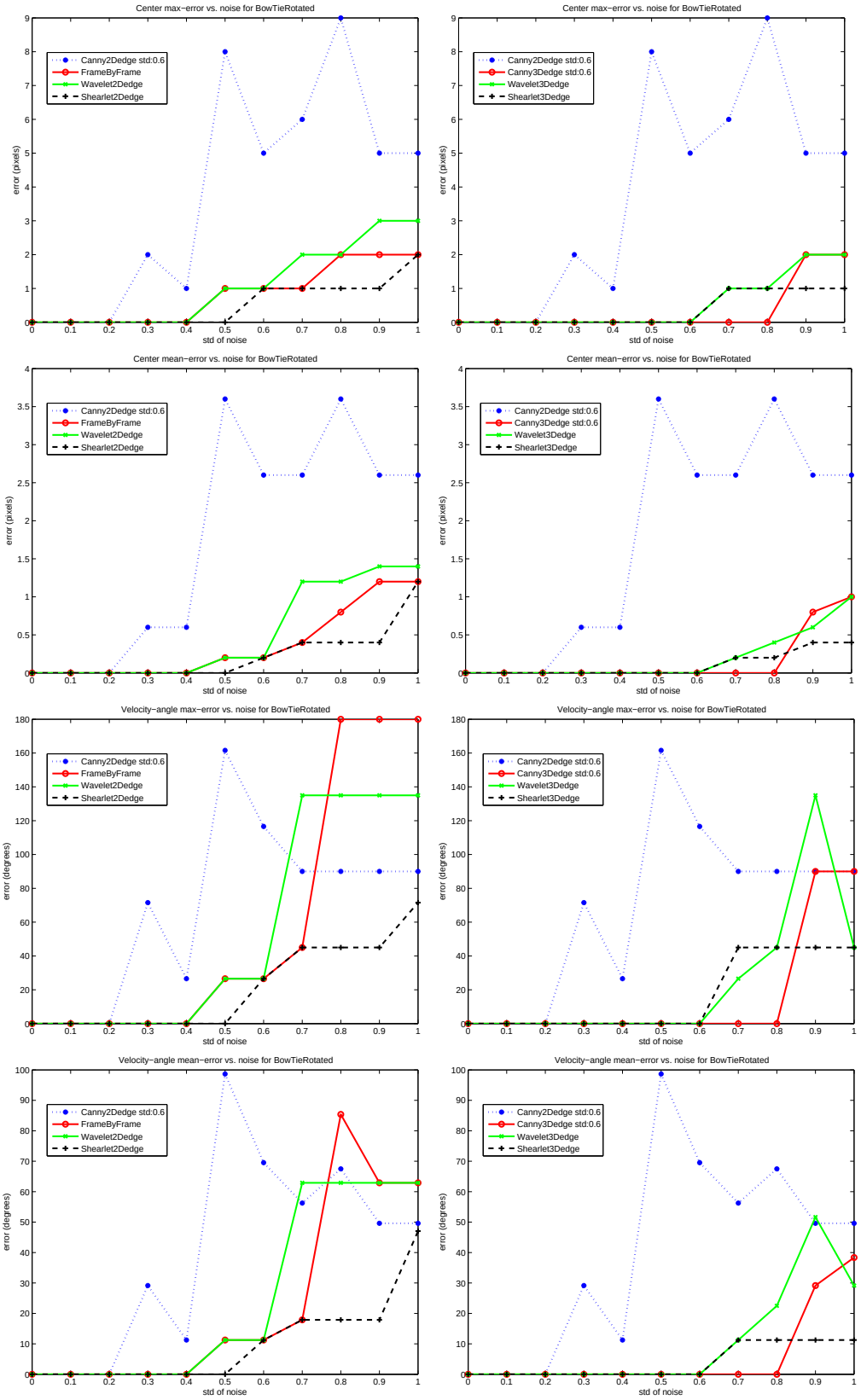


Figure 4.10: Results for spiraling rotating bow-tie with noise.

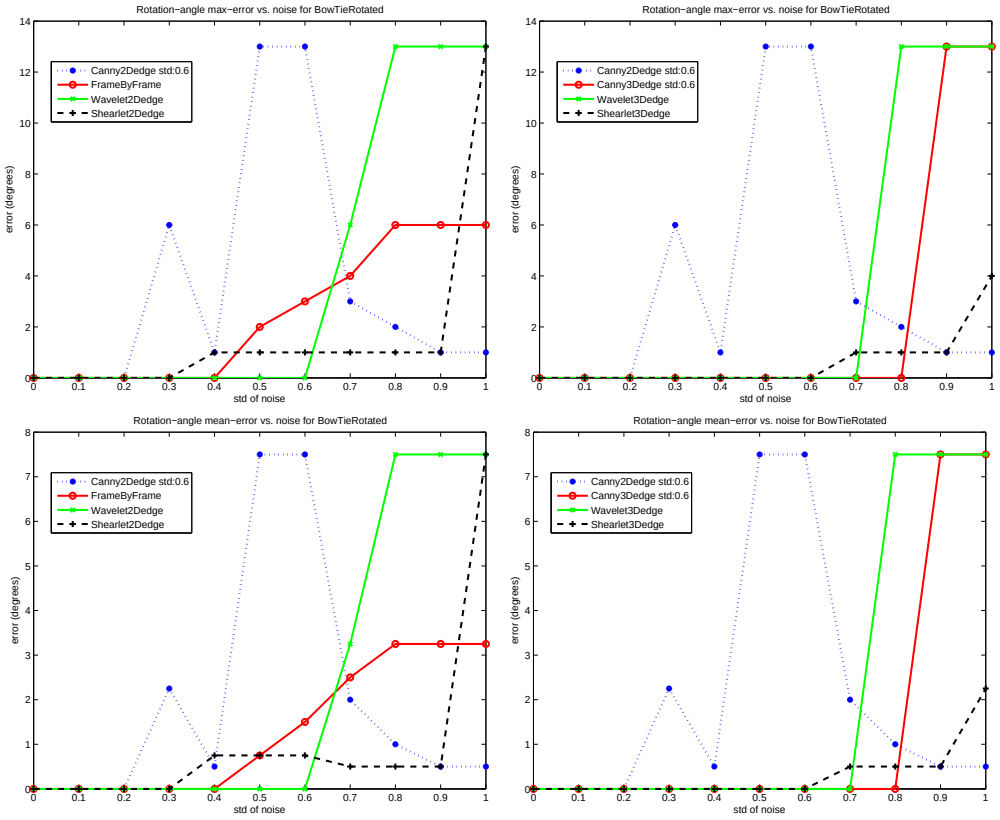


Figure 4.11: Rotation angle errors for spiraling rotating bow-tie with noise.

The fifth experiment adds rotation about the center of the object for a spiraling bowtie. The results are given in Figures 4.10 and 4.11. The 3D wavelet and shearlet algorithms are quite reliable in tracking position and rotation angle of this object.

The sixth experiment considers a bowtie that spirals and rotates and has been shaded across the diagonal of the object. Results are shown in Figures 4.12 and 4.13. The 3D methods again track position and rotation angle very well.

The last experiment evaluates the hybrid methods on a bow-tie that has been rotated and shaded gradually from dark to light along the main diagonal. Results are given in Figure 4.14. The hybrid algorithms are quite effective at lower cost than the other 3D algorithms.

4.5 Conclusions

We have presented 2D and 3D tracking algorithms based on edge detection. All of the methods perform with low error for simple objects in moderate noise. 2D methods are more likely to break track at higher noise levels. All of the 3D methods stay on track for both position and velocity metrics, even for complicated shapes. Shading was the first feature that noticably increased the errors for the 3D methods. Angular velocity errors are lower for the shearlet and wavelet than for the Canny algorithms. The 2D and 3D shearlets are the best performers for complicated objects. Noisy rotating shaded objects separate the 3D wavelet from the higher-quality 3D shearlet. When objects are simple, it is not necessary to use the 3D shearlet. Hybrid methods conserve computational resources by only

applying costly shearlet or wavelet transforms in the image planes where motion or object shape is more complicated. Additional velocity information provided by image gradients over time stabilizes tracking as noise increases.

Transformations other than translation and rotation could be included in future work. Expansions and contractions of the patch would account for movement toward and away from the camera. We could also allow for roll and yaw of a 3D target with known shape.

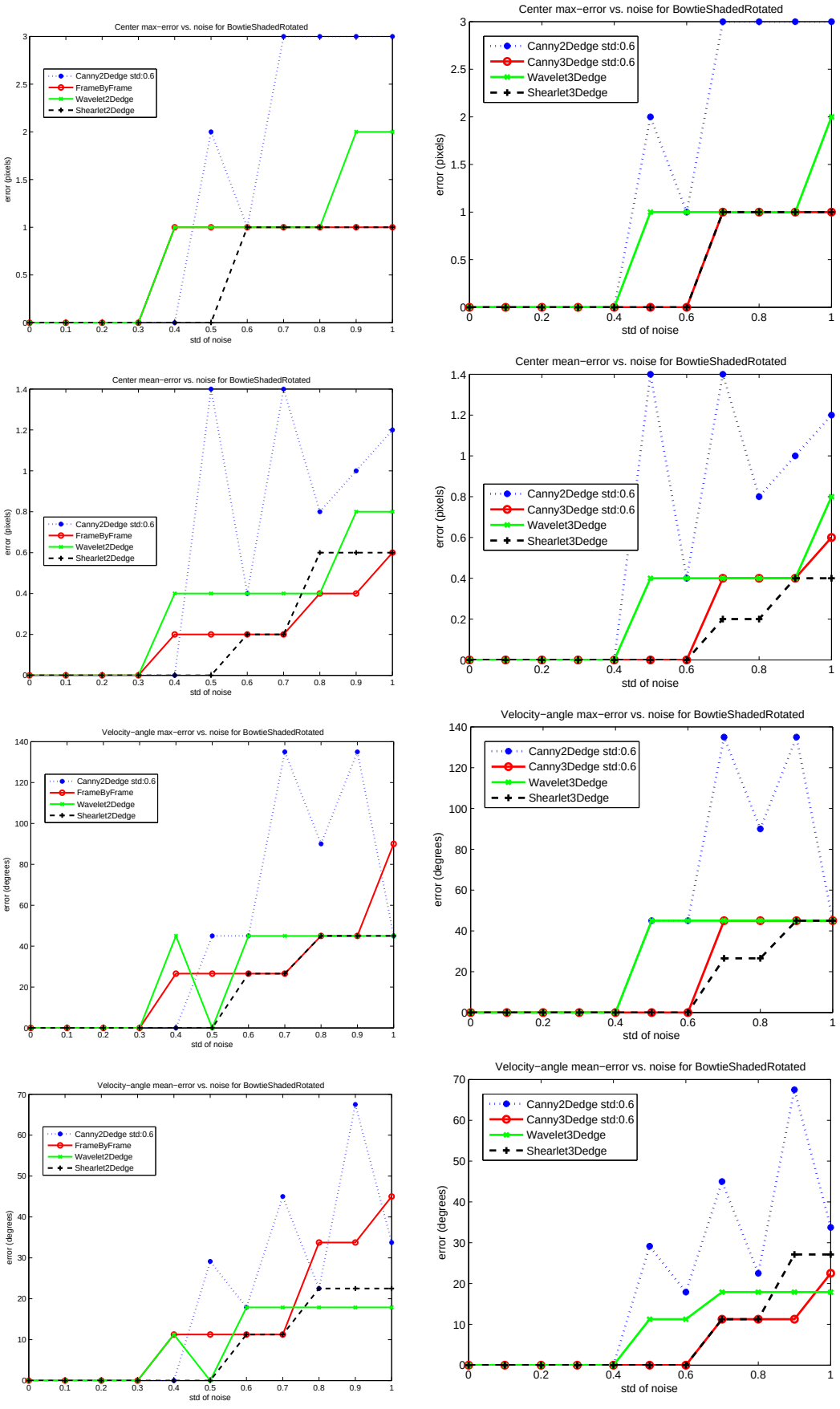


Figure 4.12: Results for spiraling rotating shaded bow-tie with noise.

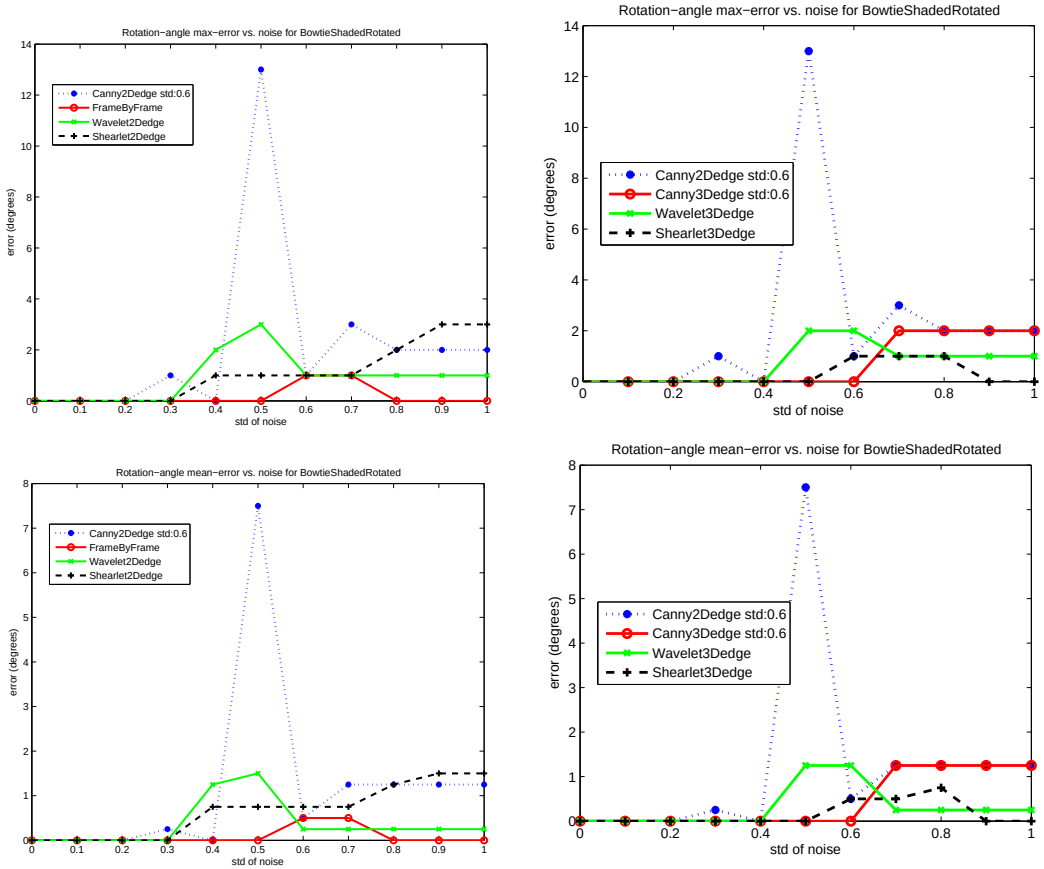


Figure 4.13: Rotation angle errors for spiraling rotating shaded bow-tie with noise.

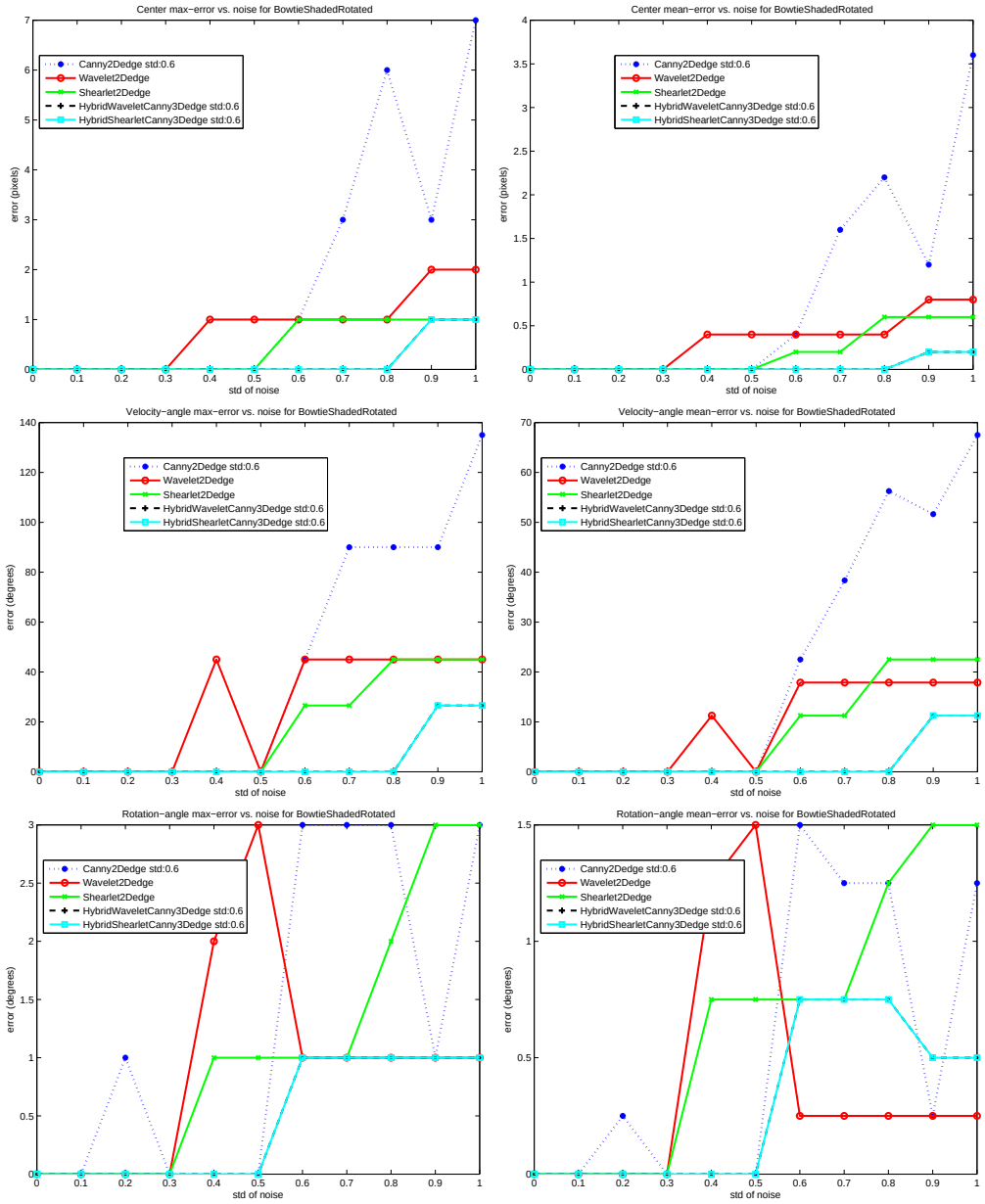


Figure 4.14: Maximum (left) and mean (right) errors in centers (top), velocity angles (middle), and rotation angles (bottom) for hybrid algorithm on a rotating spiraling bow-tie with noise.

Chapter 5

Summary and Future Work

5.1 Conclusion

This dissertation has presented the background to implement 2D and 3D edge detectors based on the Canny, wavelet, and shearlet analysis algorithms. In particular my contribution has been to design and implement 3D wavelet and 3D shearlet edge detectors. The edge detectors were also successfully integrated into a new tracking system as a practical application, using synthetic data. The 3D edge detectors provided better localization and velocity accuracy in the presence of noise. Several different test cases illustrated the performance of the 2D and 3D algorithms. The 2D algorithms are reliable for simple objects that follow smooth trajectories in moderate noise, but break down under high levels of noise. All of the 3D methods perform very well in the midst of high noise levels even for objects with more complicated shape. Shade is a problem that will degrade all of the 3D edge detectors' performance, but the multi-scale algorithms handle this case better than the Canny approach. The multi-scale, multi-directional shearlet algorithms are best for tracking complicated shapes and trajectories. The added benefit of 3D edge detection is the additional velocity information that was not previously available from 2D edge detection methods. The image gradient over time from the 3D algorithms provided better position and velocity results than the 2D algorithms. Hybrid 2D-3D methods

reduce computational cost but retain the benefit of estimating image velocity.

5.2 Future Work

The most important work I would like to pursue in the near future is the integration of edge detection into a photogrammetric tracking system. When collecting data for tracking purposes, the best measurements come from cameras with a line of sight normal to the object's plane of motion. Sometimes it is not possible to place the camera at the correct line of sight. Therefore multiple cameras collect images from different points of view to perform scene analysis. If the track object is moving quickly, high frame rates and more images are necessary to reconstruct the object's trajectory correctly. Long image sequences make automation of tracking a priority. Noise and illumination further complicate the tracking process. Reliable 3D edge detection will help to provide faster more robust 2D tracking. Future work includes these research projects:

- Integrate 2D and 3D edge detectors into a 3D photogrammetric tracking system that works with real-world data.
- Use the image velocity information from 3D edge detectors as measurements to improve the accuracy of 3D Bayesian state estimation techniques that currently use image position data.
- Prepare an edge detection toolbox.
- Continue testing 3D edge detectors with different problems.

- Improve and automate the choice of scales and directions for multi-scale, multi-directional filter banks.
- Design new multi-scale, multi-directional partitioning of 3D frequency space.
- Refine and automate 3D thresholding and 3D non-maximal suppression.
- Improve the capability of the edge detection based tracking system to track more complicated targets.
- Add image acceleration to the 2D tracking problem.
- Parallelize the edge detection and tracking process.
- Apply 3D shearlets to denoise and restore volumetric data sets.

Bibliography

[1] M. Brejl, and M. Sonka, "Directional 3D Edge Detection in Anisotropic Data: Detector Design and Performance Assessment", *Computer Vision and Image Understanding*, vol. 77, pp. 84–110, 1999.

[2] J. F. Canny, "A Computational Approach to Edge Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8(6), 679-698, 1986.

[3] J. F. Canny, "Finding Edges and Lines in Images ," *Master's Thesis*, MIT, 1983.

[4] J. Geusebroek, A. W. M. Smeulders, and J. van de Weijer, "Fast Anisotropic Gauss Filtering", *IEEE Transactions on Image Processing*, vol. 8, pp. 938-943, 2003.

[5] R. C. Gonzalez, and R. E. Woods, *Digital Image Processing* (Prentice Hall, Upper Saddle River, 2002).

[6] K. Guo, D. Labate, "Analysis and Detection of Surface Discontinuities Using the 3D Continuous Shearlet Transform", *Applied and Computational Harmonic Analysis*, vol. 30(2), pp. 231–249, 2010.

[7] D. Labate, and W.-Q. Lim, G. Kutyniok, and G. Weiss, "Sparse multidimensional representation using shearlets", *Wavelets XI* , SPIE Proc. vol. 5914, 254–262, SPIE, Bellingham, WA, 2005.

[8] S. A. Mallat, *A Wavelet Tour of Signal Processing* (Academic, San Diego, 1998).

[9] S. A. Mallat and W. L. Hwang, "Singularity Detection and Processing with Wavelets", *IEEE Transactions on Information Theory*, vol. 38(2), pp. 617-643, 1992.

[10] S. A. Mallat and S. Zhong, "Characterization of Signals from Multiscale Edges", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14(7), pp. 710-732, 1992.

[11] O. Monga, S. Benayoun, "Using Partial Derivatives of 3D Images to Extract Typical Surface Features", *INRIA Report research*, 1992.

[12] O. Monga, R. Deriche,, "3D Edge Detection Using Recursive Filtering: Application to Scanner Images", *CVGIP: Image Understanding*, San Diego, vol. 53(1), pp. 76-87, January 1991.

[13] P. Perona, "Steerable-scalable Kernels for Edge Detection and Junction Analysis", *Image and Vision Computing*, vol. 10, pp. 663-672, 1992.

[14] D. A. Schug, G. R. Easley, and D. P. O'Leary, "Three-dimensional Shearlet Edge Analysis", *SPIE: Defense, Security, and Sensing*, Orlando, April 25-29, 2011.

[15] D. A. Schug, G. R. Easley, and D. P. O'Leary, "Tracking Objects Using Three Dimensional Edge Detection", in preparation, 2012.

[16] J. Weickert, "Foundations and Applications of Non-linear Anisotropic Diffusion Filtering", *Zeitschrift für Angewandte Mathematik und Mechanik.*, vol. 76, pp. 283-286, 1996.

[17] Z. Weiping, S. Hauzhong, "Detection of Cerebral Vessels in MRA Using 3D Steerable Filters", *Engineering in Medicine and Biology 27 Annual Conference*, Shanghai, September 1-4, 2005.

[18] S. Yi, D. Labate, G. R. Easley, and H. Krim, "Edge Detection and Processing Using Shearlets", *Proceedings IEEE International Conference on Image Processing*, San Diego, October 12-15, 2008.

[19] S. Yi, D. Labate, G. R. Easley, and H. Krim, "A Shearlet Approach to Edge Analysis and Detection", *IEEE Transactions on Image Processing*, vol. 18(5), pp. 929–941, 2009.

[20] D. Ziou, S. Tabbone, "Edge Detection Techniques An Overview", *International Journal of Pattern Recognition and Image Analysis* , vol. 8(4), pp. 537–559, 1998.