Precise State Tracking Using Three Dimensional Edge Detection

David A. Schug[1], Glenn R. Easley[2], and Dianne P. O'Leary[3]

[1] NAWCAD
[2] MITRE, 7515 Colshire Drive, McLean, VA 22102
[3] Computer Science Department and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742

Abstract. An important goal in applications such as photogrammetry is precise kinematic state estimation (position, orientation, and velocity) of complex moving objects, given a sequence of images. Currently, no method achieves the precision and accuracy of manual tracking under difficult real-world conditions. In this work, we describe a promising new direction of research that processes the 3D datacube formed from the sequence of images and uses edge detectors to validate position hypotheses. We propose a variety of new 3D edge/surface detectors, including new variants of wavelet- and shearlet-based detectors and hybrid 3D detectors that provide computational efficiency. The edge detectors tend to produce broad edges, increasing the uncertainty in the state estimates. We overcome this limitation by finding the best match of the edge image from the 3D data to edge images derived from different state hypotheses. We demonstrate that our new 3D state trackers outperform those that only use 2D information, even under the challenge of changing lighting conditions.

1 Introduction

When cameras record a sequence of observations of an object moving in the field of view, we can try to track that object precisely. Complex object motion and complicated shape increase the difficulty of this type of tracking. Multiple target features on the object can be selected, as shown in Fig. 1, in order to accurately fit the geometry and represent motion through space. It is important to have enough features to reliably represent the object and the motion, which means, for example, additional features if the object has fins or sharp curves. Other factors, including noise, clutter, and illumination variations, also make tracking difficult, since these change image intensities in complicated ways. In addition, low contrast image intensities make it difficult to discern an image feature from the background.

david.schug@navy.mil, geasley@mitre.org, oleary@cs.umd.edu

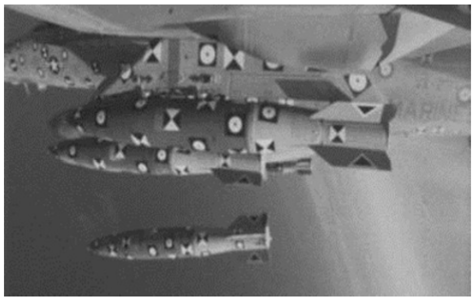


Fig. 1. Circular and bowtie-shaped features on objects to be tracked.

Very precise tracking is needed in photogrammetry, where the goal is to estimate the three-dimensional rigid-body kinematics of objects. With precision, we are concerned with a deviation from the target center that is relatively constant from frame to frame. A nearly constant deviation is more precise. Estimation accuracy will refer to the particular magnitude of deviation from a chosen standard. For real-world data, the chosen standard will depend on how well a human can estimate the image feature's center point. Typical tolerances require the object state parameters to be measurable to within one inch for Cartesian position and within one degree for Euler rotations. Tracking in 2D image space must therefore be accurate to within 2 pixels on average. Tracking accuracy will also depend on the camera's resolution influenced by the distance and orientation of the image feature with respect to the camera's field of view, and the camera's specific performance characteristics.

1.1 Previous Work in Tracking

Most methods that aim to estimate the kinematic state are feature-based searches that minimize a cost function that generates the sum of squared distances between chosen projected points and their corresponding observations from a particular image sequence (see [24] for more details). In practice, a two dimensional feature tracker such as the Kanade-Lucas tracker [16] or other correlation based methods are used to collect observations while maintaining required correspondences with the chosen locations on the three dimensional object.

Good techniques for general tracking include those provided by Lee et al. [14] and subspace methods for face tracking such as those in [1], [4], and [28]. Video-based tracking methods such as those provided in [32],[11], and [15] are also effective. Despite substantial progress in tracking algorithms, however, no single method has achieved the precision of manual tracking in photogrammetry. Most approaches to this particular kind of tracking problem have made use of edge detectors. Meaningful changes in edges are the fundamental criteria for distinguishing image features from the background. This is because the tracker can use the boundary of the feature to precisely register its orientation and position. At first glance, this may seem to be a complete solution to this tracking problem, but edge detectors can produce an estimated edge that is nonuniform in thickness, making it difficult to estimate critical attributes of the feature, such as its center and its velocity. In this work we use the edge detector to *validate* position estimates rather than to infer them.

1.2 Previous Work in Edge Detection

Traditional edge detection is performed with a single image I recorded at positions $\mathcal{P} = \{(i,j), i = 1,\dots,m,\ j = 1,\dots,n\}$. Given a threshold $h > 0$, the output of the edge detection process is a set of edge locations $\mathbf{P_E} \subseteq \mathcal{P}$ and an edge image $\mathbf{I_E}$ defined by

$$\mathbf{I_E}(i,j) = \begin{cases} 1 & (i,j) \in \mathbf{P}_E, \\ 0 & (i,j) \in \mathcal{P} \backslash \mathbf{P}_E. \end{cases} \tag{1}$$

Mathematically, we define

$$\mathbf{P}_E = \{\mathbf{p} \in \mathcal{P} : |\nabla I(\mathbf{p})| \geq h\}\,. \tag{2}$$

The image gradient $\nabla I(\mathbf{p})$ cannot be computed exactly, and edge detection algorithms differ in how they approximate this quantity.

There are many approaches to edge detection, including one based on computing eigenvalues of an autocovariance matrix to discover when the power spectrum of a function is slowly decreasing [5]. Perhaps the most widely used 2D edge detector is that due to Canny [3]. It is based on approximating the image gradient through convolution of the image with a filter formed from the derivative of a Gaussian. The algorithm depends on a single parameter σ_c, the standard deviation of the Gaussian

$$g_{\sigma_c}(x,y) = \exp\left(-(x^2+y^2)/(2\sigma_c^2)\right), \tag{3}$$

which effectively sets the scale for the smoothing of the image. The Canny algorithm is quite fast and is effective for simple shapes if σ_c is tuned to the image and the noise conditions, but this is not always possible. Fig. 2 shows the results of applying MATLAB's `edge` algorithm using the Canny edge detector with various choices of σ, to the noisy image shown in the top center. Smoothing at a single scale, i.e., with a single value of σ, makes it difficult to balance noise suppression with feature loss.

A further complication comes from choosing the correct thresholds that separate image features from the background or noise. Often one must manually adjust parameters until the method produces helpful results. The end

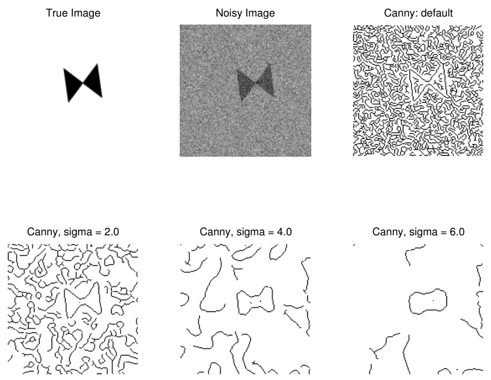


Fig. 2. Results of MATLAB's `edge` algorithm using the Canny edge detector with various choices of `sigma`. The pixel values in the true image are in the range $[0, 1]$, and the standard deviation of the added noise is 0.4.

result can be acceptable, but the level of human intervention is not satisfactory for automatic tracking applications.

A 2D wavelet edge detector [18, 19] overcomes the single-scale smoothing limitation of the Canny algorithm. The wavelet transform [30] provides a representation of an image using a set of basis functions that resolve detail at multiple scales and thus can reveal edge information at various levels of smoothing. Very efficient numerical implementations of the wavelet transform [17] make it practical. It can be much more effective than the Canny algorithm for complicated shapes, but because the wavelets are isotropic, it still has difficulty distinguishing close edges, crossing edges, and sharp changes in edge curvature. To resolve such cases, basis functions that have a sharp directional orientation are needed. One solution is to replace the scalable collection of isotropic Gaussian filters g_{σ_c} used in the Canny algorithm with a family of steerable and scalable anisotropic Gaussian filters [8]

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

where $a_1, a_2 > 0$, R_θ is the rotation matrix for angle θ, and G is a Gaussian basis function parametrized by $a_1^{-1}x_1$ and $a_2^{-1}x_2$. The design and implementation of such filters is computationally involved, and the justification is essentially intuitive, lacking theory to prescribe parameters that best capture edges.

The shearlet transform provides this theory [13]. It provides a sparse directional representation [13, 29] key to obtaining a good edge detector. It is an invertible transform that relies on a set of analyzing functions (directionally oriented filters) that partition the frequency space at different scales and orientations. Knowing from mathematical analysis how the magnitudes at the edges change in this representation with respect to scale and shear (see [12] and [10]), a method for detecting edges and their orientation was given in

[29]. The result is an improved capability to successfully detect subtle intensity differences and complicated object shapes.

All of these 2D methods have 3D counterparts that can be used for movies, sequences of images. Monga and Deriche [21] developed one of the first 3D Canny detectors, using separable Gaussians for smoothing. They applied their method to magnetic resonance and echographic volumetric data. Monga and Benayoun [20] extended the state of the art mathematically by using partial derivatives to treat the 3D image as a hypersurface in 4-dimensional space. They computed the curvatures at designated edge points using the partial derivatives of the image but did not obtain directional information. Brejl and Sonka [2] designed a directional 3D edge detector. Weiping and Hauzhong [27] used 3D wavelets to detect cerebral vessels in magnetic resonance angiograms (MRA). In [25] and [23], we proposed using the 3D shearlet transform for edge/surface detection and demonstrated its advantages over other methods in distinguishing high-curvature edges in low SNR conditions. It should be pointed out that this early 3D shearlet-based edge detector [25] is different from the one proposed in this work, which makes more explicit use of the analytic properties of the shearlet transform at surfaces and edges.

We believe that edge detection is much improved by including the time dimension, so we use 3D detectors. Previously, 3D detectors for tracking have been investigated in [6] and [31]. However, these are only for point features and are not suitable for our purpose. Another breakthrough in our approach is to make use of robust sparse and directional filtering concepts such as those provided in [24], [6], and [31].

1.3 Outline and Contributions

In Section 2, we discuss the tracking problem and the data. We then make several contributions.

- Section 3 focusses on new 3D edge detectors.
  - We develop new variants of 3D wavelet- and shearlet-based edge detection algorithms. We present a new 3D shearlet transform algorithm that is better suited to edge/surface detection than the version developed in [22]. This new algorithm exploits the theory provided in [9] and contains extensions as well as important improvements over the algorithms developed in [29] for the purpose of feature tracking.
  - With efficiency in mind, we devise inexpensive but effective hybrids, combining results of 2D wavelet, shearlet, or Canny edge detectors on $x-y$, $y-t$, and $x-t$ slices, where x and y are spacial dimensions and t denotes time.
  - We demonstrate the effectiveness of these edge detection algorithms, but show that they are not adequate for precise tracking.
- In Section 4 we propose a totally new approach to tracking, using edge detectors to *validate* rather than *generate* state hypotheses, thus avoiding

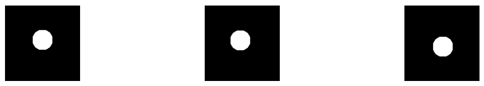


Fig. 3. Three frames from the orbiting ball movie.

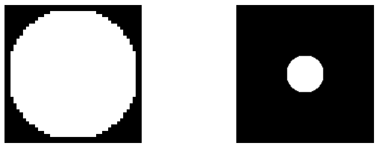


Fig. 4. Patch containing the disk (left) is inserted into a black background to create a noise-free frame of the movie (right).

the uncertainty imposed by broad estimates of edges, and in Section 5 we demonstrate the effectiveness of our tracking ideas.

We draw conclusions in Section 6.

The edge detection algorithms described here were used by us in [26]. MATLAB software implementing these algorithms is available at `http://www.cs.umd.edu/users/oleary/software`.

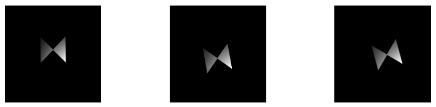


Fig. 5. Three frames from an orbiting bow-tie movie with rotation and illumination changes.

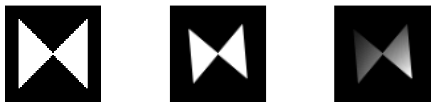


Fig. 6. Patches containing a bow-tie (left), a rotated bow-tie (middle), and a shaded rotated bow-tie (right).

2 The Data

The tracking problem starts with an observed image sequence (movie) that captures 2D snapshot information about 3D objects moving in the camera's field of view. We present two test problems that illustrate some of the difficulties in tracking a single feature from Fig. 1. Tracking of multiple features can be done in parallel.

For our first test problem, a camera records a movie of a 3D ball of radius r whose position in the sequence of 2D images describes a circular orbit at radius R about the center pixel. Each image frame in the movie looks like a white disk on a black background, moved via translation, with the center (x_j, y_j) of the disk at time t_j, relative to the center of the image, given by

$$x_j = \lfloor R\cos(\alpha(t_j))\rfloor, \tag{4}$$

$$y_j = \lfloor R\sin(\alpha(t_j)\rfloor, \tag{5}$$

where $\alpha(t_j)$ is the angle defining the position of the object at time t.

Three frames from the resulting orbiting ball movie are shown in Fig. 3. We chose the diameter of the disk to be an odd number of pixels so that in generating the data we can center it on the nearest pixel. The movie $\tilde{\mathbf{I}}$ is stored in an $m \times m \times \ell$ array, with $m^2 = 157^2$ pixels per frame and $\ell = 30$ frames. 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 Fig. 4.

Our second test problem, with sample frames shown in Fig. 5, is generated in a similar way, but uses a bow-tie patch, shown in Fig. 6 (left), that orbits about the center of the frame but also rotates about its own center point, as illustrated in Fig. 6 (middle). To perform rotation, we remap 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 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 $\mathbf{L} = \mathbf{g}^T\mathbf{g}$. The shaded object is obtained by elementwise multiplication of the patch $\mathbf{P}$ by $\mathbf{L}$:

$$\mathbf{S} = \mathbf{P}. * \mathbf{L}\ . \tag{6}$$

This is performed after rotation and produces a result like that shown on the right in Fig. 6.

We assume that we know the position of the object in the first frame of the movie. To study the robustness of our algorithms, we add white noise (independent normally distributed samples for each pixel) to the frames. 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. In the next section we introduce 3D edge detectors that provide an important tool in making these estimates.

3 3D Edge Detectors

In this section we provide a brief description of how the 3D Canny, 3D wavelet, new 3D shearlet, and the new hybrid edge detectors identify edges.

The edge estimates produced by any of these algorithms can be refined by two well-known methods discussed in standard texts. *Nonmaximal suppression* labels a voxel as an element of the edge surface if its estimated gradient magnitude is at least h and if the magnitude is greater than at least one of its neighboring pairs. This can be applied in 3D or, to save time, in 2D, comparing each pixel to its neighboring pairs in the compass directions N-S, E-W, NE-SW, and NW-SE. *Hysteresis thresholding* identifies a voxel as a *strong* edge voxel if its gradient magnitude is greater than a threshold h. It is also identified as an edge voxel if it is connected to a strong edge and its gradient magnitude is larger than a threshold h_{low} and larger than the magnitude of each of its two neighbors in at least one of the compass directions. This, too, can be applied in 2D or 3D.

3.1 3D Canny Edge Detection

The 3D Canny algorithm (Algorithm 1) makes use of the 3D Gaussian low pass filter

$$\mathbf{g}_\sigma^{3D} = \exp\left(-(x^2+y^2+t^2)/(2\sigma^2)\right), \tag{7}$$

where σ is the standard deviation for the Gaussian. After convolution with this filter, it then uses convolution with a discretization of the partial derivatives $\mathbf{d}g_{\sigma,x}^{3D}$, $\mathbf{d}g_{\sigma,y}^{3D}$, and $\mathbf{d}g_{\sigma,t}^{3D}$ to estimate derivatives of the smoothed image sequence. It operates at a single scale, determined by the choice of σ, and produces estimates of the derivatives at the voxel centers.

Algorithm 1 The 3D Canny Edge Detection Algorithm.

```
1: Input: Raw image sequence $\tilde{\mathbf{I}}$ and parameter $\sigma$.
2: Output: $(\nabla\tilde{\mathbf{I}}_x, \nabla\tilde{\mathbf{I}}_y, \nabla\tilde{\mathbf{I}}_t)$, estimates of the gradient for each pixel in the input.
3: Compute the smoothed sequence $\tilde{\mathbf{I}}_s = \tilde{\mathbf{I}} * \mathbf{g}_\sigma^{3D}$.
4: Compute horizontal derivative estimate $\nabla\tilde{\mathbf{I}}_x = \tilde{\mathbf{I}}_s * \mathbf{d}g_{\sigma,x}^{3D}$.
5: Compute vertical derivative estimate $\nabla\tilde{\mathbf{I}}_y = \tilde{\mathbf{I}}_s * \mathbf{dg}_{\sigma,y}^{3D}$.
6: Compute time derivative estimate $\nabla\tilde{\mathbf{I}}_t = \tilde{\mathbf{I}}_s * \mathbf{dg}_{\sigma,t}^{3D}$.
```

3.2 3D Wavelet Edge Detection

A wavelet representation is a multiscale representation that allows us to overcome the problem of choosing an appropriate scale parameter σ. Given a function f in $L^2(\mathbb{R}^3)$ and an appropriate well-localized *mother wavelet* function $\psi \in L^2(\mathbb{R}^3)$, we define the continuous wavelet transform of f to be

$$\mathcal{W}_\psi f(a,t) = a^{-1} \int_{\mathbb{R}^3} f(x)\, \psi\left(a^{-1}(x-t)\right) dx,$$

where $a > 0$ and $t \in \mathbb{R}^3$. The analysis functions (wavelets) are $\psi_{a,t}(x) = a^{-1}\psi\left(a^{-1}(x-t)\right)$.

If f on $\mathbb{R}^3$ is smooth except for a discontinuity at $x_0 \in \mathbb{R}^3$, the wavelet transform $\mathcal{W}_\psi f(a,t)$ decays rapidly as $a \to 0$ everywhere, except where t is near x_0. Hence, the wavelet transform is able to signal the location of the singularity of f through its asymptotic decay at fine scales.

We discretize the wavelet transform and write $\tilde{\psi}_a(x) = a^{-1}\psi(-x/a)$, so that the wavelet transform can be expressed as the convolution product $\mathcal{W}_\psi f(a,t) = f * \tilde{\psi}_a(t)$. As a mother wavelet, we use a Sobel-like filter (instead of the Gaussian derivative used in Canny) to estimate the gradient and repeatedly apply a smoothing matrix to effectively dilate $\tilde{\psi}$ by an amount dependent on the number of iterations ℓ, obtaining $\tilde{\psi}_\ell$. This means we can concisely write the implementation as $f * \tilde{\psi}_\ell$ for $\ell = 1, 2, \ldots, n_\ell$.

Stacking each plane of the filter side-by-side reveals the contents for the horizontal, vertical, and time filters:

$$\mathbf{G}_x^{3D} = \left[\begin{array}{ccc|ccc|ccc} 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 \\ 1 & 0 & -1 & 2 & 0 & -2 & 1 & 0 & -1 \\ 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 \end{array}\right], \tag{8}$$

$$\mathbf{G}_y^{3D} = \left[\begin{array}{ccc|ccc|ccc} 0 & 1 & 0 & 1 & 2 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & -1 & -2 & -1 & 0 & -1 & 0 \end{array}\right], \tag{9}$$

$$\mathbf{G}_t^{3D} = \left[\begin{array}{ccc|ccc|ccc} 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 1 & 2 & 1 & 0 & 0 & 0 & -1 & -2 & -1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \end{array}\right]. \tag{10}$$

After the image gradient is estimated using these filters, the wavelet edge detection algorithm repeatedly rescales by convolving the gradient components with a weighted average filter

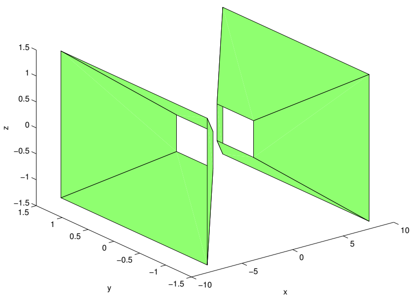
$$\mathbf{G}_a^{3D} = \left[\begin{array}{ccc|ccc|ccc} 1 & 1 & 1 & 1 & 2 & 1 & 1 & 1 & 1 \\ 1 & 2 & 1 & 2 & 4 & 2 & 1 & 2 & 1 \\ 1 & 1 & 1 & 1 & 2 & 1 & 1 & 1 & 1 \end{array}\right], \tag{11}$$

choosing to save either the previous estimate or the new one, depending on which has smaller magnitude. This is summarized in Algorithm 2. Note that notation in the description of the algorithm emphasizes the approximate gradient aspects rather than the wavelet aspects.

The conditional re-enforcement in steps 7 and 8 emphasizes edge locations based on the change between adjacent scales. For both wavelets and shearlets, the local Lipschitz regularity of a point determines the decay of the magnitude of the response as a function of scale [17], [7]. The coefficients are likely to increase slowly with ℓ when the Lipschitz regularity is positive (i.e. an edge point) and increase rapidly when the Lipschitz regularity is negative (i.e. a noise point). Thus the choice is meant to strengthen the influence of edges and weaken the influence of noise.

Algorithm 2 The 3D Wavelet Edge Detection Algorithm.

```
1: Input: Raw image sequence $\tilde{\mathbf{I}}$ and number of levels $n_\ell$.
2: Output: $(\nabla\tilde{\mathbf{I}}_x^{(n_\ell)}, \nabla\tilde{\mathbf{I}}_y^{(n_\ell)}, \nabla\tilde{\mathbf{I}}_t^{(n_\ell)})$, estimates of the gradient for each pixel in the input.
3: Compute the basic horizontal derivative estimate $\nabla\tilde{\mathbf{I}}_x^{(0)} = \tilde{\mathbf{I}} * \mathbf{G}_x^{3D}$, the basic vertical derivative estimate $\nabla\tilde{\mathbf{I}}_y^{(0)} = \tilde{\mathbf{I}} * \mathbf{G}_y^{3D}$ and the time derivative estimate $\nabla\tilde{\mathbf{I}}_t^{(0)} = \tilde{\mathbf{I}} * \mathbf{G}_t^{3D}$.
4: for level $\ell = 1, \ldots, n_\ell$ do
5:     Compute the horizontal derivative estimate $\nabla\tilde{\mathbf{I}}_x^{(\ell)} = \nabla\tilde{\mathbf{I}}_x^{(\ell-1)} * \mathbf{G}_a^{3D}$.
6:     Similarly, compute the vertical and time derivative estimates $\nabla\tilde{\mathbf{I}}_y^{(\ell)}$ and $\nabla\tilde{\mathbf{I}}_t^{(\ell)}$.
7:     Modify the horizontal derivative estimate $\nabla\tilde{\mathbf{I}}_x^{(\ell)}$ by choosing (for each point in $\mathcal{P}$) the minimum magnitude component from either $\nabla\tilde{\mathbf{I}}_x^{(\ell-1)}$ or from the smoothed estimate $\nabla\tilde{\mathbf{I}}_x^{(\ell)}$.
8:     Similarly, compute the vertical and time derivative estimates $\nabla\tilde{\mathbf{I}}_y^{(\ell)}$ and $\nabla\tilde{\mathbf{I}}_t^{(\ell)}$.
9: end for
```



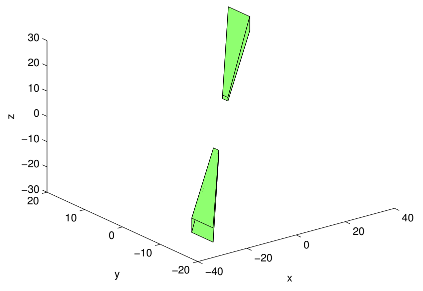


Fig. 7. The support of a 3D shearlet in the frequency domain with $a = 1/4$ and $s_1 = s_2 = 0$ (left) and $a = 1/16$, $s_1 = 0.5$, and $s_2 = 0.7$ (right).

3.3 3D Shearlet Edge Detector

In shearlet analysis, we refine the wavelet analysis by, at each level, identifying components corresponding to different regions in frequency space.

The 3D shearlet transform implementation we have developed, like the 3D wavelet transform developed here, repeatedly rescales the gradient components, but at each scale it also partitions the frequency domain into a number of subdomains

$$\left\{(\eta_1, \eta_2, \eta_3) \mid \eta_1 \in \left[-\frac{2}{a}, -\frac{1}{2a}\right] \cup \left[\frac{1}{2a}, \frac{2}{a}\right], \right.$$
$$\left. \left|\frac{\eta_2}{\eta_1} - s_1\right| \leq \frac{\sqrt{2a}}{4}, \ \left|\frac{\eta_3}{\eta_1} - s_2\right| \leq \frac{\sqrt{2a}}{4}\right\}. \tag{12}$$

Each subdomain, illustrated in Fig. 7, is a pair of hyper-trapezoids, symmetric with respect to the origin, oriented according to the slope parameters s_1 and s_2, and more elongated as $a \to 0$.

Shearlet analyzing functions are defined as

$$\psi_{a,s_1 s_{,2},t}(x) = |\det M_{as_1s_2}|^{-\frac{1}{2}} \psi(M_{as_1s_2}^{-1}(x-t))$$

where

$$M_{as_1s_2} = \begin{pmatrix} a & -a^{1/2}s_1 & -a^{-1/2}s_2 \\ 0 & a^{1/2} & 0 \\ 0 & 0 & a^{1/2} \end{pmatrix},$$

and the mapping

$$\mathcal{SH}_\psi f(a, s_1, s_2, t) = \langle f, \psi_{a,s_1 s_{,2},t} \rangle$$

defines the *continuous shearlet transform* of f for $a > 0$ and $t \in R^3$. (See [9] for a complete description as the technical issues in the construction are extensive.) The matrix $M_{as_1s_2}$ is a product of a dilation matrix dependent on a and shearing matrices. Creating filters $w_{d,\ell}^{3D}$ whose frequency response produces the appropriate hyper-trapezoidal restrictions when combined with a wavelet filtering is done by extending the corresponding 2D filters $w_{d,\ell}$ constructed in [29]. The subscript d is an index used to replace the dependency of the window function on s_1 and s_2. The integer d ranges between 1 and n_d, where n_d indicates the total number of directional components for a scale we index by ℓ. An additional weighting correction is applied to each $w_{d,\ell}^{3D}$ to guarantee that the summation of all n_d components is a delta function. The continuous shearlet transform of f can then essentially be calculated as $f * (\tilde{\psi}_\ell * w_{d,\ell}^{3D})$ for $\ell = 1, \ldots, n_\ell$ and $d = 1, \ldots, n_d$. Thus, we can extend our wavelet transform algorithm to be a shearlet transform algorithm by doing an additional loop with a convolution dependent on $w_{d,\ell}^{3D}$.

Let Ω be a region in $\mathbb{R}^3$ with boundary denoted by $\partial\Omega$ and let γ_j, $j = 1, 2, \ldots, m$ be the smooth boundary segments of $\partial\Omega$, assumed to have positive Gaussian curvature at every point. If B is a function that is one for every point in Ω and zero elsewhere, then we know from [9] that:

- If $t \notin \partial\Omega$, then

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

- If $t \in \partial\Omega \backslash \cup_{j=1}^{m} \gamma_j$ and (s_1, s_2) does not correspond to the normal direction of $\partial\Omega$ at t, then
$$\lim_{a \to 0^+} a^{-N} \mathcal{SH}_\psi B(a, s_1, s_2, s_2, t) = 0 \quad \text{for all } N > 0.$$
- If $t \in \partial\Omega \backslash \cup_{j=1}^{m} \gamma_j$ and $(s_1, s_2) = (\bar{s}_1, \bar{s}_2)$ corresponds to the normal direction of $\partial\Omega$ at t or $t \in \cup_{j=1}^{m} \gamma_j$ and (s_1, s_2) corresponds to one of the two normal directions of $\partial\Omega$ at t, then
$$\lim_{a \to 0^+} a^{-1} \mathcal{SH}_\psi B(a, s_1, s_2, t) \neq 0.$$
- If $p \in \gamma_j$ and (s_1, s_2) does not correspond to the normal directions of $\partial\Omega$ at t, then
$$|\mathcal{SH}_\psi B(a, s_1, s_2, t)| \leq C a^{3/2}.$$

The above result essentially says that the continuous shearlet transform of a bounded region with piecewise smooth boundary has rapid decay everywhere, except when the location variable t is on the surface and the shearing variables correspond to the normal orientation, in which case it decays like $O(a)$ as $a \to 0$.

Our idea is that, at each level, for each shearlet region, we reinforce components that seem to be decaying at the proper rate by checking if the magnitude at a given position is between α and α^{-1} times the previous value. This comparison method is a significant improvement over the simple comparison concept originally conceived in [29] as this reduces thickening of the nominated edge points and increases efficiency.

Our shearlet edge detection algorithm, summarized in Algorithm 3, resembles the wavelet algorithm, but there is an inner loop at each scale that enhances the estimated derivative magnitude when an edge aligns with a shearlet filter $w_{d,\ell}^{3D}$.

3.4 3D Hybrid Wavelet and Shearlet Edge Detectors

Our experimental results show that the 3D wavelet and 3D shearlet edge detectors are quite effective but quite expensive. Therefore, we developed 3D hybrid wavelet-Canny and 3D hybrid shearlet-Canny edge detectors. These methods use 2D wavelet or 2D shearlet methods to process each image, producing estimates of the x and y derivatives. The time derivative estimate is then taken from the 3D Canny algorithm. We summarize this process in Algorithm 4. There may be an advantage in averaging the Canny estimate for each horizontal and vertical derivative with the wavelet or shearlet estimate; if not, then the algorithm can be made more efficient by omitting the horizontal and vertical computations in the 3D Canny step.

Similarly, we developed edge detectors based on repeated use of 2D wavelet (or shearlet) edge detectors to compute 3D estimates as shown in Algorithm 5.

All of these algorithms are economical, and some proved quite effective.

Algorithm 3 The 3D Shearlet Edge Detection Algorithm.

```
1: Input: Raw image sequence $\tilde{\mathbf{I}}$ and number of levels $n_\ell$ and parameter $\alpha \in [0,1]$.
2: Output: $(\nabla\tilde{\mathbf{I}}_x, \nabla\tilde{\mathbf{I}}_y, \nabla\tilde{\mathbf{I}}_t)$, estimates of the gradient for each pixel in the input.
3: Compute the basic horizontal derivative estimate $\nabla\tilde{\mathbf{I}}_x = \tilde{\mathbf{I}} * \mathbf{G}_x^{3D}$.
4: Similarly, compute the basic vertical and time derivative estimates $\nabla\tilde{\mathbf{I}}_y = \tilde{\mathbf{I}} * \mathbf{G}_y^{3D}$
   and $\nabla\tilde{\mathbf{I}}_t = \tilde{\mathbf{I}} * \mathbf{G}_t^{3D}$.
5: Precompute the shearing filters $w_{d,\ell}^{3D}$.
6: for level $\ell = 1, \ldots, n_\ell$ do
7:    Compute the horizontal derivative estimate $\nabla\tilde{\mathbf{I}}_x^{(+1)} = \nabla\tilde{\mathbf{I}}_x * \mathbf{G}_a^{3D}$.
8:    Similarly, compute the vertical and time derivative estimates $\nabla\tilde{\mathbf{I}}_y^{(+1)}$ and
      $\nabla\tilde{\mathbf{I}}_t^{(+1)}$.
9:    Initialize $\Delta_x = \Delta_y = \Delta_t = \mathbf{0}$.
10:   for direction $d = 1, \ldots, n_d$ do
11:      Add into $\Delta_x$ any element of $\nabla\tilde{\mathbf{I}}_x * w_{d,\ell}^{3D}$ whose magnitude is between $\alpha$ and
         $\alpha^{-1}$ times $\nabla\tilde{\mathbf{I}}_x^{(+1)} * w_{d,\ell}^{3D}$.
12:      Update $\Delta_y$ and $\Delta_t$ similarly.
13:   end for
14:   Add $\Delta_x$ to $\nabla\tilde{\mathbf{I}}_x$.
15:   Update $\nabla\tilde{\mathbf{I}}_y$ and $\nabla\tilde{\mathbf{I}}_t$ similarly.
16: end for
```

Algorithm 4 The 3D Hybrid Wavelet-Canny (or Shearlet-Canny) Edge Detection Algorithm.

```
1: Input: Raw image sequence $\tilde{\mathbf{I}}$ and number of levels $n_\ell$.
2: Output: $(\nabla\tilde{\mathbf{I}}_x, \nabla\tilde{\mathbf{I}}_y, \nabla\tilde{\mathbf{I}}_t)$, estimates of the gradient for each pixel in the input.
3: Compute horizontal and vertical derivative estimates $\nabla\tilde{\mathbf{I}}_x$ and $\nabla\tilde{\mathbf{I}}_y$ by applying
   the 2D wavelet (or 2D shearlet) edge detector to each frame in the sequence $\tilde{\mathbf{I}}$
   using $n_\ell$ levels.
4: Compute a time derivative estimate $\nabla\tilde{\mathbf{I}}_t$ by applying the 3D Canny edge detec-
   tion algorithm to $\tilde{\mathbf{I}}$.
```

3.5 Performance of the Edge Detectors

To illustrate the potential of these edge detectors for tracking moving objects, we do a simple comparison of their performance. Figs. 8–10 show results for the fifth frame of the rotating bow-tie movie with various noise levels. The movie frame is labeled FbF (frame-by-frame), and the results from Canny-2D (C2), Canny-3D (C3), Wavelet (W2 and W3), Shearlet (S2 and S3), and the hybrid algorithms (HWC, HSC, HWW, HSS) are also shown. We see that all of the algorithms are reliable when the SNR is high, but for low SNR, the 3D wavelet and shearlet algorithms are most reliable. Unfortunately, the algorithms can produce very broad edge estimates and can lose detail at sharp corners. This makes it very difficult to determine the precise location of a feature from the raw output of the edge detectors, so next we consider how to use this output more effectively in tracking.

Algorithm 5 The 3D Hybrid Wavelet-Wavelet (or Shearlet-Shearlet) Edge Detection Algorithm.

```
1: Input: Raw image sequence $\tilde{\mathbf{I}}$ and number of levels $n_\ell$.
2: Output: $(\nabla\tilde{\mathbf{I}}_x, \nabla\tilde{\mathbf{I}}_y, \nabla\tilde{\mathbf{I}}_t)$, estimates of the gradient for each pixel in the input.
3: Apply the 2D wavelet (or 2D shearlet) edge detector to each frame ($xy$ slice) in the sequence $\tilde{\mathbf{I}}$ using $n_\ell$ levels to obtain horizontal and vertical derivative estimates $\boldsymbol{h}_{1x} = \nabla\tilde{\mathbf{I}}_x$ and $\boldsymbol{h}_{1y} = \nabla\tilde{\mathbf{I}}_y$.
4: Similarly, apply the 2D wavelet (or 2D shearlet) edge detector to each $xt$ slice in the sequence to obtain horizontal and time derivative estimates $\boldsymbol{h}_{2x} = \nabla\tilde{\mathbf{I}}_x$ and $\boldsymbol{h}_{2t} = \nabla\tilde{\mathbf{I}}_t$.
5: Apply the 2D wavelet (or 2D shearlet) edge detector to each $yt$ slice in the sequence to obtain vertical and time derivative estimates $\boldsymbol{h}_{3y} = \nabla\tilde{\mathbf{I}}_y$ and $\boldsymbol{h}_{3t} = \nabla\tilde{\mathbf{I}}_t$.
6: Compute derivative estimates $\nabla\tilde{\mathbf{I}}_x = \frac{1}{2}(\boldsymbol{h}_{1x} + \boldsymbol{h}_{2x})$, $\nabla\tilde{\mathbf{I}}_y = \frac{1}{2}(\boldsymbol{h}_{1y} + \boldsymbol{h}_{3y})$, and $\nabla\tilde{\mathbf{I}}_t = \frac{1}{2}(\boldsymbol{h}_{2t} + \boldsymbol{h}_{3t})$.
```

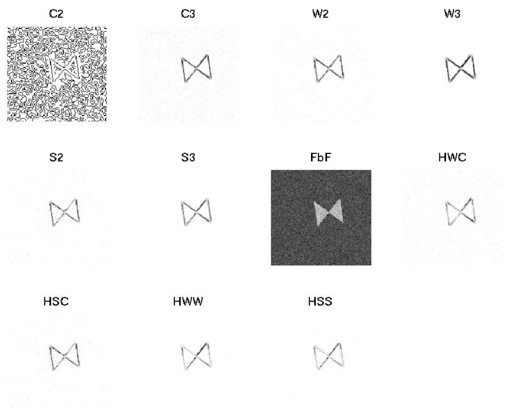


Fig. 8. Results of edge detectors on the rotating bow-tie movie, with standard deviation of noise 0.2. The methods are denoted as Canny (C), Wavelet (W), and Shearlet (S), 2D and 3D, the original frame (FbF), and Hybrid (H).

4 From Edge Detection to Tracking

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 tailor our algorithm to our photogrammetric application; since we have an image of the object to be tracked, we can make use of this information to avoid any need to identify features, and this is a significant advantage.

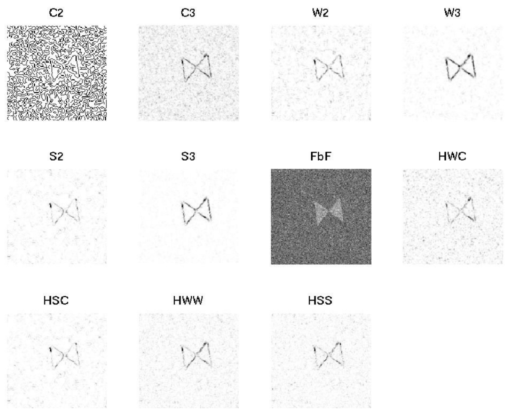


Fig. 9. Results of edge detectors on rotating the bow-tie movie, with standard deviation of noise 0.6.

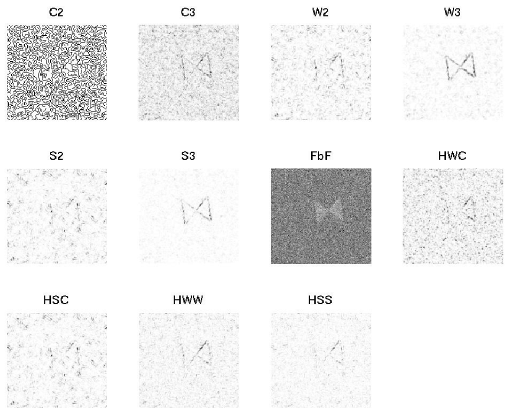


Fig. 10. Results of edge detectors on the rotating bow-tie movie, with standard deviation of noise 1.0.

Assume that we are trying to determine the movement of the object between two particular frames, frame $k-1$ and frame k. Assume that the center of the object in frame $k-1$ is (i, j). We denote the displacement as Δx_k in the horizontal direction and Δy_k in the vertical direction and drop the subscript k 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 of pixels defining the object, 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 displacement values between -2 and 2 for i and j, 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 kth frame replaced by one with the patch in its trial position. To find the correct position of the feature, we want to minimize the difference between the two sets of data, so we use a cost function

$$f(\tilde{\mathbf{I}}_p) = \|D(\tilde{\mathbf{I}}) - D(\tilde{\mathbf{I}}_p)\|. \tag{13}$$

A natural choice of norm is the square root of the sum of squares of the elements, but other choices are possible.

We also have a choice of the function D. The most obvious choice is $D(\tilde{\mathbf{I}}) = \tilde{\mathbf{I}}$. In this case f measures how the pixel values change when we replace the kth frame by the patched frame. Preservation of (noisy) pixel values is not our objective, however; we want to preserve edges. We propose, therefore, that D denote the edge image sequence produced by one of our edge detectors. In this case, f measures 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 kth frame by the patch positioned at $(i + \Delta x, j + \Delta y)$.

If the object is also rotating, then we need to measure the cost function at various values of $\Delta\theta$, the change in rotation angle since the previous frame, as well as Δx and Δy. 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. We found that our methods worked better if we added noise to the patch, comparable to that in the original movie, before inserting it into the kth frame of the movie.

We summarize our tracking method in Algorithm 6. There are two important observations to be made concerning the cost of the algorithm.

First, increasing the number of 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 non-differentiable 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.

Second, there is a very important cost savings to be made. Rather than running the edge detector on the full image sequence, we can use a smaller *submovie* formed from a limited number of frames around frame k and a limited number of pixels within each frame, those near (i, j), since the effects on the edge detectors due to introducing the patch are primarily local. Making use of the submovie greatly reduces the cost of each trial.

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

Algorithm 6 Tracking Using Edge Detection.

```
1: $D$ denotes the output of one of our edge detectors.
2: **Input**: Image sequence $\tilde{\mathbf{I}}$ with $\ell$ frames, noise estimate, patch $\mathbf{P}$, shading vector
   $\mathbf{g}$, and initial patch location.
3: **Output**: Estimates of patch motion $\Delta x$, $\Delta y$, and $\Delta\theta$ for each frame.
4: Initialize $\Delta x_1 = \Delta y_1 = \Delta\theta_1 = 0$.
5: Add noise to the patch $\mathbf{P}$.
6: for $k = 2 : l$ do
7:    Record $\Delta x_k = \Delta y_k = \Delta\theta_k = 0$ as the best guess so far.
8:    for $d\theta = -2 : 1 : 2$ do
9:       Rotate the patch by angle $d\theta$: $\mathbf{P}_r = \texttt{imrotate}(\mathbf{P}, d\theta)$.
10:      Compute shaded patch $\mathbf{S} = \mathbf{P}_r . * (\mathbf{g}^T\mathbf{g})$.
11:      for $dx = -2 : 1 : 2$ do
12:         for $dy = -2 : 1 : 2$ do
13:            The current trial location is the patch location at frame $k - 1$ plus
               $(dx, dy)$.
14:            Replace frame $k$ of the image sequence $\tilde{\mathbf{I}}$ with a frame containing the
               patch $\mathbf{S}$ at the trial location, obtaining $\tilde{\mathbf{I}}_p$.
15:            if $\|D(\tilde{\mathbf{I}}) - D(\tilde{\mathbf{I}}_p)\|$ (calculated using the relevant subimage sequence)
               is smaller than all previous values for frame $k$ then
16:               Set $\Delta x_k = dx$, $\Delta y_k = dy$, and $\Delta\theta_k = d\theta$.
17:            end if
18:         end for
19:      end for
20:   end for
21:   Replace the patch $\mathbf{P}$ by rotating it by $\Delta\theta_k$.
22: end for
```

$$|v_k| = \sqrt{\Delta x_k^2 + \Delta y_k^2}, \tag{14}$$

and the direction of the planar velocity,

$$\phi_k = \arctan\left(\frac{\Delta y_k}{\Delta x_k}\right). \tag{15}$$

However, ϕ_k is quite sensitive to errors in Δx_k and Δy_k.

5 Experimental Results

Experiments were conducted to help understand and characterize how well our edge detectors work in tracking.

We used the difficult case of a spiraling shaded bow-tie with various amounts of noise added. For each standard deviation of the noise, we replicated the experiment 4 times, measuring the average error in single-frame tracking for frames 2 through 26. The results are shown in Figs. 11–13. If the standard

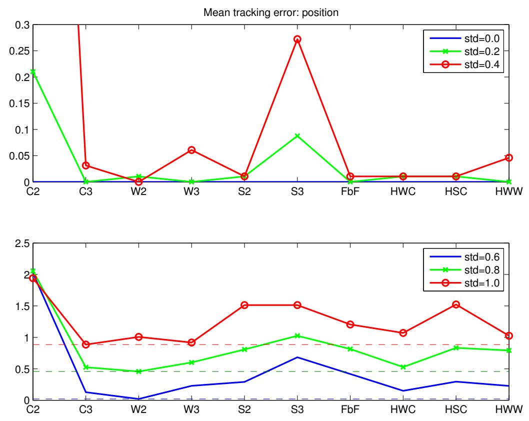


Fig. 11. Average error in position estimate (unit = pixel) for tracking a spiraling, rotating, shaded bowtie with noise. The dotted lines indicate the minimum error for each noise level.

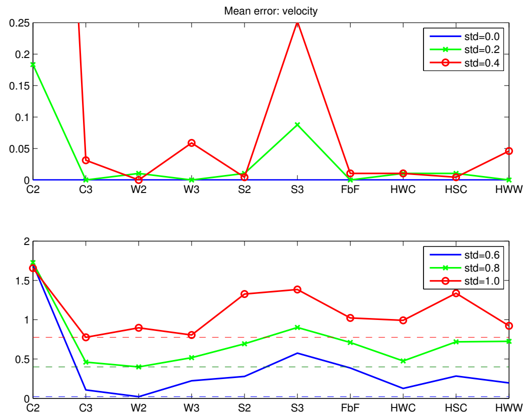


Fig. 12. Average error in velocity estimate (unit = pixel / frame) for tracking a spiraling, rotating, shaded bowtie with noise.

deviation of the noise is 0.0, all of the edge detectors yielded perfect tracking. As the noise level increased, all algorithms except Canny-2D performed quite well, but the most reliable algorithms were Canny-3D, Wavelet-3D, Hybrid Wavelet-Canny, and Hybrid Wavelet-Wavelet for our particular set of experiments.

6 Conclusions

We have developed new variants of 3D wavelet- and shearlet-based edge detectors and new hybrid detectors that provide 3D information using only 2D

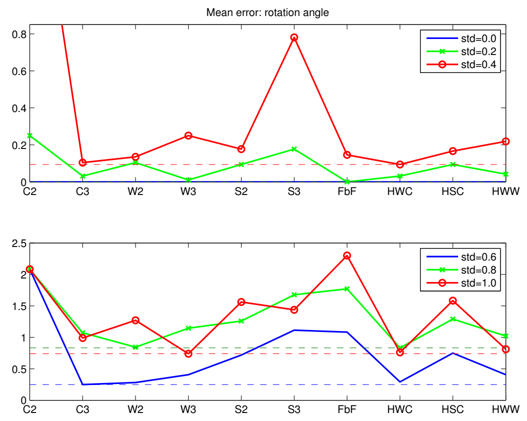


Fig. 13. Average error in rotation angle estimate (unit = degree) for tracking a spiraling, rotating, shaded bowtie with noise.

wavelet or shearlet transforms. We demonstrated the effectiveness of these algorithms for edge detection. Our wavelet edge detectors try to filter noise from the gradient estimates, while our shearlet detectors reinforce gradients that change with scale at the expected rate. A variety of other implementations are possible, and some may perform better than these. All of these methods could be improved by tuning parameters and by applying standard post-processing techniques such as nonmaximal suppression or hysteresis thresholding.

We then developed algorithms for tracking objects moving under translation and rotation, using edge detectors to validate position estimates. All of the methods tested, except Canny-2D, give low error in position, velocity, and rotation angle estimates in moderate noise. These methods are well adapted to particular applications involving rigid motion and flat backgrounds.

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 feature with known shape. Also, by fitting the patch to the object, our assumption of flat background could be removed.

Acknowledgments. D.P.O acknowledges partial support from the National Science Foundation under grant NSF DMS 1016266.

References

1. S. Birchfield. Elliptical head tracking using intensity gradients and color histograms. In *Computer Vision and Pattern Recognition, 1998. Proceedings. 1998 IEEE Computer Society Conference on*, pages 232–237. IEEE, 1998.
2. M. Brejl and M. Sonka. Directional 3D edge detection in anisotropic data: Detector design and performance assessment. *Computer Vision and Image Understanding*, 77:84–110, 1999.

3. J. F. Canny. Finding edges and lines in images. Master's thesis, MIT, 1983.
4. Y. Chen, Y. Rui, and T. S Huang. JPDAF based HMM for real-time contour tracking. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–543. IEEE, 2001.
5. W. Czaja and M. V. Wickerhauser. Singularity detection in images using dual local autocovariance. *Appl. Comput. Harmon. Anal.*, 13(1):77–88, 2002.
6. D. Davies, P. Palmer, and M. Mirmehdi. Detection and tracking of very small low contrast objects. In *Proceedings of the British Machine Vision Conference*, pages 60.1–60.10. BMVA Press, 1998. doi:10.5244/C.12.60.
7. G. Easley, K. Guo, D. Labate, et al. Analysis of singularities and edge detection using the shearlet transform. In *SAMPTA'09, International Conference on Sampling Theory and Applications*, 2009.
8. J. Geusebroek, A. W. M. Smeulders, and J. van de Weijer. Fast anisotropic Gauss filtering. *IEEE Transactions on Image Processing*, 8:938–943, 2003.
9. K. Guo and D. Labate. Analysis and detection of surface discontinuities using the 3D continuous shearlet transform. *Applied and Computational Harmonic Analysis*, 30(2):231–242, 2011.
10. K. Guo, D. Labate, and W.-Q. Lim. Edge analysis and identification using the continuous shearlet transform. *Applied and Computational Harmonic Analysis*, 27(1):24–46, 2009.
11. V. Kruger and S. Zhou. Exemplar-based face recognition from video. *Lecture Notes In Computer Science*, pages 732–746, 2002.
12. G. Kutyniok and D. Labate. Resolution of the wavefront set using continuous shearlets. *Transactions of the American Mathematical Society*, 361(5):2719–2754, 2009.
13. D. Labate, W.-Q. Lim, G. Kutyniok, and G. Weiss. Sparse multidimensional representation using shearlets. In *Wavelets XI*, volume 5914, pages 254–262. SPIE Proc., Bellingham, WA, 2005.
14. K.-C. Lee, J. Ho, M.-H. Yang, and D. Kriegman. Visual tracking and recognition using probabilistic appearance manifolds. *Computer Vision and Image Understanding*, 99(3):303–331, 2005.
15. X. Liu and T. Cheng. Video-based face recognition using adaptive hidden Markov models. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 1, pages I–340. IEEE, 2003.
16. B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial intelligence*, 1981.
17. S. A. Mallat. *A Wavelet Tour of Signal Processing.* Academic, San Diego, 1998.
18. S. A. Mallat and W. L. Hwang. Singularity detection and processing with wavelets. *IEEE Transactions on Information Theory*, 38(2):617–643, 1992.
19. S. A. Mallat and S. Zhong. Characterization of signals from multiscale edges. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(7):710–732, 1992.
20. O. Monga and S. Benayoun. Using partial derivatives of 3D images to extract typical surface features. *Computer Vision and Understanding*, 61(2):171–189, 1995.
21. O. Monga and R. Deriche. 3D edge detection using recursive filtering: Application to scanner images. *CVGIP: Image Understanding*, 53(1):76–87, 1991.

22. P. S. Negi and D. Labate. 3-d discrete shearlet transform and video processing. *Image Processing, IEEE Transactions on*, 21(6):2944–2954, 2012.
23. D. A. Schug. *“Three Dimensional Edge Detection Using Wavelet and Shearlet Analysis.* PhD thesis, Applied Mathematics and Statistics and Scientific Computing Program, University of Maryland, 2012.
24. D. A. Schug and G. R. Easley. Three dimensional Bayesian state estimation using shearlet edge analysis and detection. In *Communications, Control and Signal Processing (ISCCSP), 2010 4th International Symposium on*, pages 1–4. IEEE, 2010.
25. D. A. Schug, G. R. Easley, and D. P. O’Leary. Three-dimensional shearlet edge analysis. In *SPIE: Defense, Security, and Sensing*, Orlando, Florida, April 25-29, 2011.
26. D. A. Schug, G. R. Easley, and D. P. OLeary. Wavelet–shearlet edge detection and thresholding methods in 3D. In R. Balan, M. J. Begué, J. J. Benedetto, W. Czaja, and K. A. Okoudjou, editors, *Excursions in Harmonic Analysis, Volume 3*, pages 87–104. Springer, 2015.
27. Z. Weiping and S. Hauzhong. Detection of cerebral vessels in MRA using 3D steerable filters. In *Engineering in Medicine and Biology 27 Annual Conference*, Shanghai, September 1-4,2005.
28. Y. Wu and T. S Huang. A co-inference approach to robust visual tracking. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 2, pages 26–33. IEEE, 2001.
29. S. Yi, D. Labate, G. R. Easley, and H. Krim. A shearlet approach to edge analysis and detection. *IEEE Transactions on Image Processing*, 18(5):929–941, 2009.
30. S. Yi, D. Labate, G. R. Easley, and H. Krim. Edge detection and processing using shearlets. In *Proceedings IEEE International Conference on Image Processing*, San Diego, CA, October 12-15, 2008.
31. A. Yilmaz, K. Shafique, and M. Shah. Target tracking in airborne forward looking infrared imagery. *Image and Vision Computing*, 21(7):623–635, 2003.
32. S. Zhou, V. Krueger, and R. Chellappa. Probabilistic recognition of human faces from video. *Computer Vision and Image Understanding*, 91(1):214–245, 2003.