Using Spatio-Temporal Patches for Simultaneous Estimation of Edge Strength, Orientation, and Motion

Andrew N. Stein* and Martial Hebert
The Robotics Institute, Carnegie Mellon University
Pittsburgh, Pennsylvania, USA
anstein@cmu.edu, hebert@ri.cmu.edu

Abstract

We describe an extension to ordinary patch-based edge detection in images using spatio-temporal volumetric patches from video. The inclusion of temporal information enables us to estimate motion normal to edges in addition to edge strength and spatial orientation. The method can handle complex edges in clutter by comparing distributions of data on either half of an extracted patch, rather than modeling the intensity profile of the edge. An efficient approach is provided for building the necessary histograms which samples candidate edge orientations and motions. Results are compared to classical spatio-temporal filtering techniques.

1. Introduction

The use of 2D patches in images has become common practice for object recognition tasks ([2, 7, 16, 29] to name a few), but recently several researchers have begun to explore the utility of spatio-temporal, *volumetric* patches extracted from video data, mostly in the context of event/action detection and recognition [12, 13, 23]. In this work we investigate the use of spatio-temporal patches in an important low-level vision task: simultaneous detection of edges *and* their motion. Our approach extends to the spatio-temporal domain ideas from existing spatial (2D) patch-based edge detectors, which offer improved performance in textured/cluttered regions versus classical filtering techniques.

Edges are known to contain significant information useful for understanding scene content. In particular, those edges which correspond to physical boundaries of objects and structures in a scene, often referred to as occlusion boundaries, provide some of the most useful information (as compared to edges due simply to appearance/texture) [5]. Such edges provide cues for reasoning about the geometric structure of a scene, and they are the basic cue for shape, both of which can be used for object recognition. Indeed, for many objects, shape is the only distinctive characteristic, precluding the use of popular appearance-based features, such as SIFT [16].

Because edges have long been understood to contain such essential information for image understanding, their detection is one of the oldest problems in computer vision and image processing. Low-level edge detectors typically report edge strength and (optionally) orientation for each position in an image. Higher-level reasoning about low-level edge information relies on the ability to group this local edge information into more global structure, usually by looking for long chains of similarly-oriented edges which may form a partial or closed boundary. This problem is at the heart of perceptual grouping and Gestalt theory [30].

There are two main approaches to edge detection: filtering techniques and patch-based methods, with the former being the most prevalent by far. At their most basic level, filtering approaches model an edge as a step function of image intensity. Therefore, edges can be detected by looking for positions in the image with high derivative (or zero second derivative), e.g. using the response of the image to a derivative-of-Gaussian filter. Other more complicated filters have also been designed which can handle more complex models of edges [3, 21]. By using steerable filters [8], it is also possible to look for peak response with respect to orientation. Exploiting the use of separable filters, all of these approaches can be made very efficient. Unfortunately, performance is poor at edges which do not conform to the particular intensity profiles for which the filters were designed. For example, this is a problem at boundaries bordering textured or cluttered regions.

One can also take a slightly more general, non-model-based view of edges. In this case, the profile of the edge is not assumed to have any particular form. Locally, an edge is instead defined to be a line segment which divides a patch of data into halves which contain significantly different *distributions* of some property (*e.g.* brightness, color, texture, etc.). By comparing histograms computed in each half of a circular patch, such approaches have been shown to produce

*Part of this research was performed under the Intelligent Robotics Development Program, a 21st Century Frontier R & D Program funded by the Korean Ministry of Commerce, Industry, and Energy. Support was also provided by the National Science Foundation.

good results even on textured/clutterd data [18, 19, 22].

When a sequence of frames is available, in mobile robotic applications for example, we can leverage the extra temporal dimension of the data for edge detection. At the very least, this extra data can be used to help filter out spurious detections due to noise by exploiting temporal redundancy. More interestingly, however, is that the temporal component allows us to add a new, information-rich attribute to local edge detections: their motion. Motion could provide powerful additional cues for grouping and higher-level reasoning. But how can we estimate it?

One approach would be to attack this problem from a frame-to-frame matching or tracking standpoint [25]. Edges could be detected in one frame and then matched to detections in the next frame by solving a correspondence problem or using tracking approaches like those used in feature tracking [28]. Unfortunately, tracking (which typically relies on local data) is notoriously difficult at edges and even more so at occlusion boundaries. Tracking at any edge suffers from the so-called aperture problem, which limits us to seeing only motion normal to the edge when considering only local information. More global reasoning could help resolve resulting ambiguities, but not without substantial computational machinery. If we try to group edges into more easily trackable chains [25] – a hard problem in its own right – we can suffer from an early commitment problem.

At occlusion boundaries, the problem is even worse. By definition, the data on either side of the boundary belongs to different surfaces, which can move independently and confound any tracking approach which assumes the local data moves contiguously. The same problem hampers matching techniques that use local appearance information, but simple line segments are not distinctive enough to be matched reliably without again resorting to a more global approach such as [14]. Dealing with occlusion boundaries in any patch-based technique, from stereo [10, 11, 27], to layered motion segmentation [24], to recognition [26], is arguably one of the most difficult challenges in their application. Methods to explicitly model occlusion boundaries and detect them via particle filters have also been explored [5, 20], but the number of required particles grows out of hand for the resulting high degree-of-freedom generative models.

On the other hand, consider treating the video data as a spatio-temporal volume, rather than processing individual frames separately. A moving edge traces out an oriented path along the temporal dimension of the data. The angle of this path corresponds to its speed. So applying an oriented edge detector to a temporal slice of data would detect edges not at some spatial orientation in the image, but at speeds corresponding to the orientation of the detector. Combining detectors for spatial orientation and motion, we can design a 3D detector which simultaneously detects edges at a given spatial orientation *and* a given speed of motion normal to the edge. The spatial orientation and speed of such a detector will correspond to the orientation of a spatio-temporal *plane* in the volume, rather than an oriented line in an single frame or temporal slice. We have also completely circumvented the occlusion boundary problem – appearance/texture edges as well as occlusion boundaries can both be handled without resorting to any higher-level reasoning.

Following this idea, 3D spatio-temporal filters were designed in the classical work of [1, 9], and their potential for efficient implementation via separability was recently reported [6]. As discussed above, these filtering approaches will likely suffer from analogous problems in three dimensions that cause problems for the 2D filtering (*e.g.* moving edges against a textured background). Naturally, one would like a 3D, spatio-temporal version of the 2D patch-based approaches to combat these problems. Thus, the efficient extension of the patch-based edge detection strategy discussed above to spatio-temporal detection of oriented, *moving* edges is the focus of the remainder of this paper.

2. Patch-based Edge Detection in Images

First, we will describe the patch-based approach to edge detection in single images in more detail before extending the method to video in the next session.

Consider a circular patch of pixels extracted from an image. The radius of this patch is related to the scale of the edges we wish to detect. To evaluate the possibility that an edge passes through the center of this patch at a particular orientation, we split the patch in half along a line at that orientation. We estimate the distribution of the data – be it raw intensity, color, textons, etc. – on either side of the proposed line by binning each half of the patch into two histograms. The more these two histograms differ from one another, the more likely an edge exists in this patch, at this orientation. By rotating the dividing line about the patch center, we can evaluate edge strength for various orientations.

There exist several possible methods of measuring the difference between two distributions. The simplest is to compute Euclidean distance between them by evaluating bin-wise count differences. Quantization effects make this approach a poor choice. A better approach would be to use the Mallow's or Earth Mover's Distance (EMD) [15, 17] as in [22] or dynamic time warping (DTW) as in [19], since each takes the distance between bins into account. While DTW can be made significantly more efficient than EMD [19], both come at an unacceptable computation cost, especially once we transition to video.

The compromise suggested by [18] is to use the χ^2-Distance,

$$\chi^2(g,h) = \frac{1}{2}\sum_{i=1}^{N_{bins}} \frac{(g_i - h_i)^2}{g_i + h_i} \tag{1}$$

between binned kernel density estimates g and h. The use of the χ^2 metric normalizes the difference between the bins by their total count. Using a small Gaussian kernel when binning the data helps nearby bins interact when differencing,

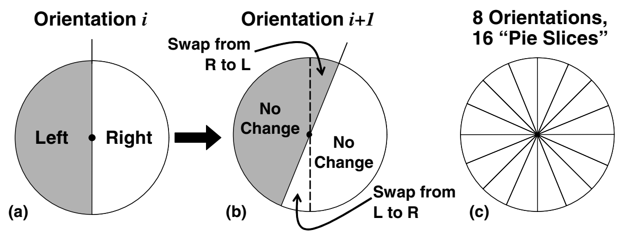

Figure 1. Circular patch used for patch-based edge detection. (a) The left and right halves of the patch at one orientation. (b) Only a fraction of the pixels in the patch will change halves as we rotate to the next orientation. (c) "Pie slices" for an 8-orientation detector.

thus approximating EMD to some degree. Note that instead of using a kernel each time we add a pixel to a histogram, we can keep simple integer counts and smooth the result with that kernel. Thus, we can replace the numerator and denominator of (1) by smoothed versions of the histogram difference and sum, $k_\sigma * (g - h)$ and $k_\sigma * (g + h)$, respectively. Here, k_σ is a Gaussian smoothing kernel with standard deviation σ.

Implemented naïvely, there is substantial redundancy in binning data from each half of the circular patch into two histograms. As shown in Figure 1 (a)-(b), a substantial portion of each half covers the same data when we rotate the dividing line from one orientation to the next. Furthermore, the slice which should be added to one half is the same slice which must be removed from the other half, and vice versa. Therefore we can first divide the patch into a "pie" of slices, as in (c), and bin the data in each slice individually. Then we can sum the histograms of appropriate slices for each half, swapping slices between halves as we rotate the dividing line.

3. Extension to Video

For video, we will consider 3D spatio-temporal patches of data instead of simple 2D patches as in the previous section. By extending the patch into the temporal dimension, we end up with a sphere of voxels rather than a disc of pixels[1]. We will split the patch into two hemispheres using an oriented plane. The intersection of this plane with image plane is a line which corresponds to the proposed edge's spatial orientation, as in the previous section. The projection of the dividing plane's normal onto a second plane which is normal to the orientation line corresponds to the motion of the proposed edge in the direction normal to its orientation. This is illustrated in Figure 2. Thus we have two degrees of freedom to specify the dividing plane for the sphere: the spatial orientation of the edge followed by the normal motion of the edge with respect to that orientation.

[1]The radius of our patch in the temporal dimension is not restricted to be the same as that used for the spatial dimensions. Thus, we may in fact have an ellipsoidal patch of data rather than sphere. But we will consider a spherical volume for simplicity in the discussion, without loss of generality.

Using the same χ^2 distance metric discussed above, we can now compare histograms computed from data on either side of a set of proposed planes through our spatio-temporal patch. The set of dividing planes will correspond to edges at various spatial orientations moving at various speeds normal to their orientation.

Note that there are two special cases of dividing planes. First, planes which are orthogonal to the image (xy-plane) represent a non-moving edge at some spatial orientation. Such planes are a direct extension of simple oriented edge detection to video and are essentially using the extra temporal information as additional support for determining oriented strength while combating imaging noise. Secondly, and more interestingly, planes which are parallel to the image plane correspond to local, instantaneous occlusion. For a spatio-temporal patch to be optimally divided along such a plane, an edge must have moved all the way across the patch volume within the time window captured by the patch. Thus, one surface completely occluded another at this position. Note that these occlusion planes have completely ambiguous spatial orientation.

This method samples spatial edge orientations, θ_s, directly. The normal motion speed can be computed from dividing plane's temporal rotation angle, θ_t, about this orientation axis. The speed is simply $tan(\theta_t)$.

Once again, computing the histograms of data in each hemisphere for each plane orientation would involve significant redundancy and inefficiency (especially now that we are working with volumetric data). Unfortunately, directly extending the "pie-slice" idea from the 2D case to this new 3D problem is non-trivial. It is not a simple matter to divide the sphere into a regular set of sub-volumes which can easily be combined to form the set of hemispheres we want. It may be possible to use an icosahedron to uniformly sample normals from the unit sphere for our dividing planes (rather than specifying a set of spatial orientations and normal motions) [4]. Then a tesselation of the sphere's volume could potentially supply the solid angle sub-volumes analogous to our 2D pie slices. These are themselves difficult problems, albeit ones that could certainly be pre-computed.

We have instead chosen a simpler approach. We choose a vertical edge as a canonical, initial orientation. For no motion, we would divide our sphere in half along the y-t plane to detect this edge. Rotating this dividing plane by θ_t about the y-axis will sample various (horizontal) motions of this vertically-oriented edge. Here we can use a direct extension of the 2D slices into 3D: we divide the spherical patch into sections like those of an orange, oriented along this vertical edge, as in the left diagram of Figure 2. Combinations and swaps of these sections as were performed with the pie slices in the 2D case can be used to construct any hemisphere, allowing us to sample motions *for this spatial orientation*.

For the next spatial orientation, we rotate this set of volume sections by θ_s about the t-axis, as in the middle of Figure 2. This results in a new set of sections that can be

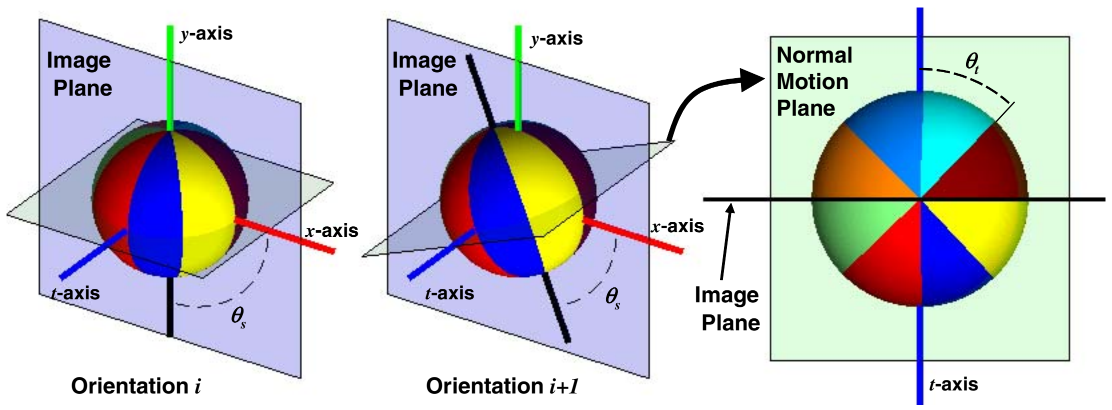

Figure 2. Our volumetric patch can be divided into histogram sections and oriented to produce a detector for a given set of spatial orientations and normal motions. See text for details.

used to determine the motion strength *for this new spatial orientation*. We can pre-compute which voxels change section membership due to the spatial rotation by θ_s, allowing us to efficiently update each section from one spatial orientation to the next. As long as we always evaluate the set of orientations in the same order, as is naturally the case in practice, we can save substantial redundant binning – particularly as the radius of the sphere increases.

4. Results

In this section we compare our results for determining instantaneous edge orientation and motion using histograms built over a spatio-temporal patch to a classical filtering approach.

4.1. Preliminaries

For the filter-based detection, we use a steerable, oriented edge filter which can detect composite intensity profiles [8, 21], but extended to a 3D spatio-temporal version [1, 9]. Its form is the sum of squared responses of a quadrature pair, where the even filter is the second derivative of the Gaussian (with the derivative taken across the putative moving edge), and the odd filter is its Hilbert transform. The filter kernel is set to the same size as the spatio-temporal patch used for our method, and the standard deviations are chosen to provide $\pm 3\sigma$ of support along each axis. The patch sizes used below are all approximately 4% of the minimum image dimension, which results in spatial radii of eight to ten pixels. The temporal radius we consider is three to five frames.

The selected orientations (both spatially and temporally) used for the patch-based and filtering approaches are precisely the same. We first sample n_s evenly-spaced spatial orientations, $\{\theta_{s_i}\}_{i=1}^{n_s}$, and then n_t samples of temporal orientation, $\{\theta_{t_j}\}_{j=1}^{n_t}$ (relative to each θ_{s_i}). Recall that normal motion speed is given by $tan(\theta_t)$ if θ_t is measured with respect to the temporal axis, as in Figure 2. Starting at the sample pair $(\theta_{s_i}, \theta_{t_j})$ which results in the maximum edge strength, we use interpolation to arrive at more accurate subsample spatial orientation and normal speed estimates.

We also suppress non-local maxima of edge strength within the image (with respect to the edge normal) and then fit parabolas to edge strengths along the edge normal to determine the edge locations with sub-pixel precision [21]. Finally, we use a hysteresis threshold to remove noisy detections. We scale the edge strengths such that the maximum is one and use the same thresholds for either approach.

In the experiments that follow, $n_s = n_t = 8$, corresponding respectively to spatial orientation samples at every $22.5°$ and normal motion speeds non-linearly sampled between 0 and ± 2.4 pixels/frame (and also including "infinite" motion, *i.e.* occlusion, as discussed earlier). Note that the use of interpolation allows us to recover speeds greater than 2.4 as well.

4.2. Synthetic and Semi-Synthetic Data

Because of the difficulty of obtaining ground truth for arbitrary video data, we will first compare results on synthetic data where ground truth is known. Here "ground truth" implies that we know the exact motion in the image. The concept of edge strength is qualitative. For simple horizontal and vertical edges, we know the true orientation, but pixelwise ground truth orientation for arbitrary curves is harder to obtain – though fairly easy to evaluate qualitatively.

The first sequence, whose middle frame is shown on the left side of Figure 3, consists of eight frames depicting a textured square translating up and to the right in front of a differently-textured background. The square moves exactly 2 pixels along each axis between every pair of frames. Note that this speed was intentionally chosen *not* to be in the set of sampled speeds. In Figure 4, edge strength maps are shown (after non-local maxima suppression and hysteresis thresholding) for filtering (top) and our method (bottom). Clearly, only our approach can find the square's boundaries against the textured background. In the center we show corresponding maps of absolute normal speed, where we see that only those edges from the square are found to be

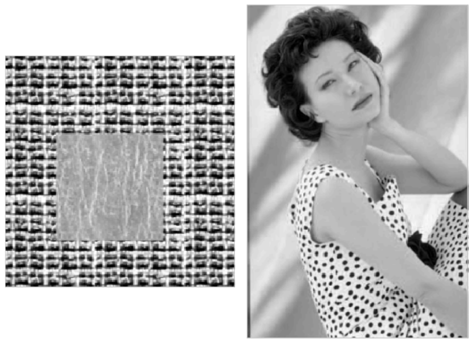
Figure 3. Representative frames for the synthetic and semi-synthetic sequences.

moving. On the right we provide a close-up view of the corner of the square displaying oriented edgelets and normal motion vectors overlaid on a lightened version of the original data. (The color of the edgelets corresponds to their strength: magenta is high, cyan is low.)

We see that both approaches respond to some of the horizontal and vertical structure of the background texture, but the filtering approach totally misses the edges of the square itself, which are arguably more useful in this sequence. Both methods correctly label the background edges as not moving and the orientations are consistent. Our method, however, also has a strong response at the square's edges and estimates their motion and orientation quite well. Note that this is entirely due to the use of histograms; there is no explicit preprocessing to handle the texture (*e.g.* using textons). For the square's edges, our method reports a mean normal speed of 1.8 (versus the correct value of 2.0) with a standard deviation of 0.2. The only comparison we can make for the filtering approach is for the few texture edges it found on the interior of the square. Since they are approximately vertical, their normal speed should also be near 2.0. The filtering approach reports a mean speed of 2.1 with standard deviation 0.16 for those edges.

The second sequence, a frame of which is provided on the right side of Figure 3, is a real image (obtained from [18]) which we synthetically translated to the right one pixel per frame for six frames, *e.g.* somewhat simulating camera motion. Its results, are shown in Figure 5. For the face and hair, both approaches do a good job detecting edges, but our approach better estimates the motion, particularly on the bridge of the nose. (Recall that we are estimating *normal* motion of the edges, thus the motion vectors should not all point to the right as one might expect in a typical optical flow result.) We also see that the polka-dots of the woman's dress prevent the filtering approach from performing well at all; edges of her body and arms are missed almost completely. For our histogram result, notice the strong edges and consistent motion along the dress and arms.

If we assume the orientation of the edge detections are correct – and they qualitatively appear to be fairly accurate – we do know what the normal speed at each edgelet should be given the known image translation. Considering only the area around the woman's hair, for which strong edges are detected by each method, our mean absolute error in normal speed is 0.10 pixels/frame, with a standard deviation of 0.08. For the same region, the filtering approach has an average error of 0.23 with a standard deviation of 0.41, making its estimation of normal speed less precise and less accurate than using our proposed patch-based method. Even more important, however, is that the patch-based approach is able to detect (and thus meaningfully estimate the motion of) many more useful edges in the scene. Note that the speed error statistics for *all* edgelets in the scene found using our method do not differ significantly from those reported for the region around the hair. The same cannot be said for the filtering result: the errors are clearly much worse on the woman's dress and body.

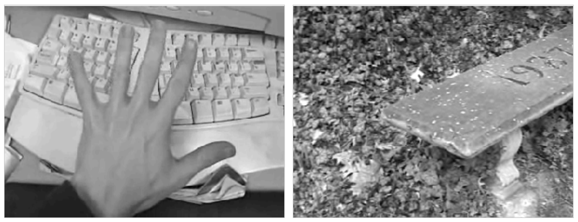
Figure 6. Representative frames for the real image sequences.

4.3. Real Data

Results for a real video sequence are provided in Figure 7. This sequence depicts a hand translating from right to left over a keyboard, as seen by a handheld, *nearly*-stationary camera (see Figure 6 for a representative frame). We expect to see relatively large normal motion for those edges perpendicular to the hand's motion and small motion vectors for the keyboard and background, due only to slight camera movements. For the simple, high contrast edges, either method performs fairly well. But the histograms used by our method are better able to distinguish the boundaries of the fingers seen against the clutter of the keys. This type of complex edge created by an approximately uniform foreground against cluttered background is not modeled well by the filtering approach. Also note that clean motion boundaries of this type could be quite useful for motion layer segmentation where feature-based approaches will likely fail on such a uniform foreground object.

Note that improving the filtering approach's results is not simply a matter of choosing a better threshold on edge strength since approximately the same strength is reported for the boundaries of the fingers as the clutter. In fact, we consistently found in all our experiments that our histogram-based method was much less sensitive to choice of threshold than the filtering technique. It also appears that using filtering over-estimates the speed of the hand slightly. For the edges detected on the keys, the orientations and speeds are completely wrong, whereas our approach's

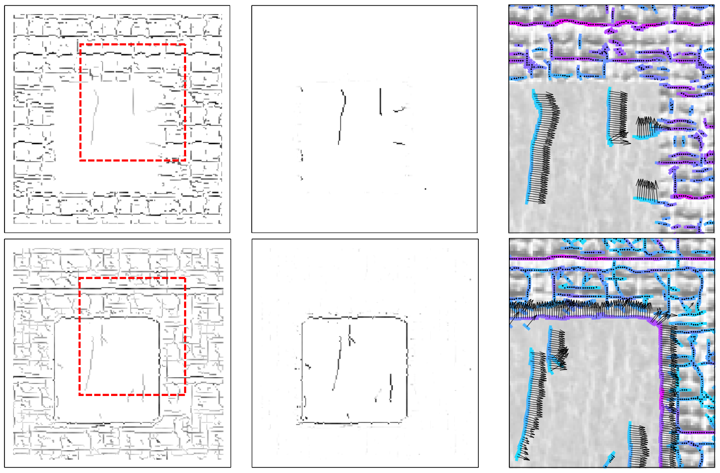

Figure 4. Results for the synthetic data. For standard filtering (top) and the proposed method (bottom), the columns depict non-local maxima suppressed edge strength (left), speed of edge motion normal to the edge (center), and a closeup view of oriented edgelets, color-coded by edge strength, with normal motion vectors for the region of the image indicated by the red dashed rectangle (right). Only our approach finds and estimates correct normal motion for the square's boundaries.

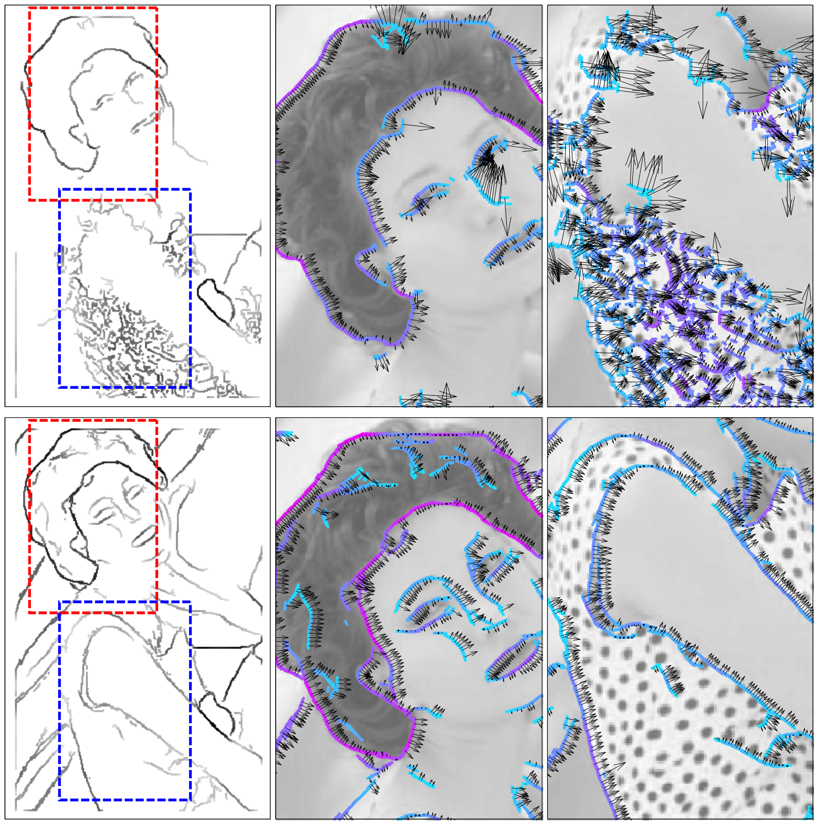

Figure 5. Results for the semi-synthetic data. The columns depict the edge strength and two close-up views of color-coded oriented edgelets and their normal motion vectors, overlaid on the data. The filtering approach (top) makes motion errors on the face and cannot handle the edges between the polka-dotted dress and smooth regions. Using our method (bottom) finds those edges, as well as ones from the face, and correctly estimates their normal motion, while ignoring spurious texture edges from the dots.

results are much more reasonable, in spite of the clutter. Finally, significantly better results were also not possible simply by selecting a better patch size (*i.e.* scale) for the filtering. Performance in some regions improved at the expense of others, depending on the chosen scale. Once again, the histogram method also appeared to be more stable

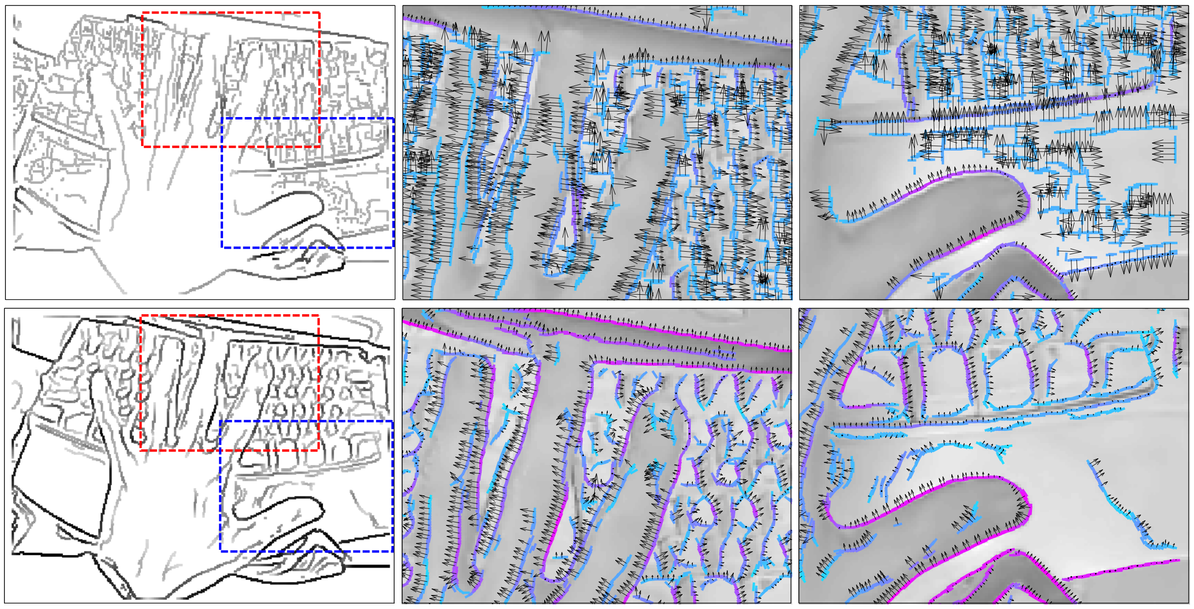

Figure 7. Results for the real sequence from the "hand" sequence of Figure 6. Our approach (bottom) finds stronger, more distinguishable edges for the fingers in front of the clutter of the keys, and its normal speed estimates are more consistent with the motion observed in the sequence, particularly for the background edges.

with respect to selection of patch size.

The second real sequence is shown on the right of Figure 6. It depicts a concrete bench in front of ivy clutter. In this sequence, the camera translates approximately in the direction of the long axis of the bench, while rotating slightly around the bench's center. The results, shown in Figure 8, again demonstrate our method's ability to detect edges bordered by a cluttered texture. For example, we detect a strong, consistent edge along the top side of the bench, while filtering performs very poorly there. We also see fewer bogus motion estimates by our method, *e.g.* around the markings on the surface of the bench and in the clutter. For the long edges of the bench, our method correctly estimates little to no normal motion (recall the camera translates approximately *along* these edges). At the close end of the bench, the correct normal motion is also shown. Even in the clutter, the motion estimates are consistent with the actual camera motion. Some of the motion estimates by the filtering approach do appear to be reasonable, for example those along the underside of the bench, where a strong edge is resolved. But in general, the result using our method is far superior.

5. Conclusions and Future Work

We have presented a spatio-temporal extension of an existing histogram-based edge detection strategy. Our approach estimates the best dividing plane for a volumetric patch of video data. The orientation of this plane directly corresponds to the spatial orientation and normal motion speed of a detected edge. Our results indicate that the same sorts of benefits of patch-based approaches for edge detection in images extend to the detection of *moving* edges in video, as compared to classical filtering methods. Our approach better finds edges bordering cluttered regions, and estimates more accurate and more consistent motion of those edges.

We have also presented an efficient method of constructing the necessary volumetric histograms which avoids unnecessary redundant binning. In the future we hope to also leverage the fact that the histogram "sectons" change relatively little as our patch shifts from site to site in an image (or frame to frame in a sequence). We should therefore be able to achieve further efficiency by pre-computing those changes, as we currently do for each change in spatial orientation of the volume.

In this work, we show results for intensity data only, but we hope to extend the approach to use color and possibly texture-analyzed data, as in [18]. Noting that our approach involves entirely local processing, a next step will be to use edges' motion cues for higher level reasoning, including perceptual grouping as well as occlusion boundary detection.

References

[1] E. H. Adelson and J. R. Bergen. Spatiotemporal energy models for the perception of motion. *Journal of the Optical Society of America A*, 2(2):284–299, February 1985.

[2] S. Agarwal, A. Awan, and D. Roth. Learning to detect objects in images via a sparse, part-based representation. *PAMI*, 26(11):1475–1490, 2004.

[3] S. Baker, S. K. Nayar, and H. Murase. Parametric feature detection. *IJCV*, 27(1):27–50, 1998.

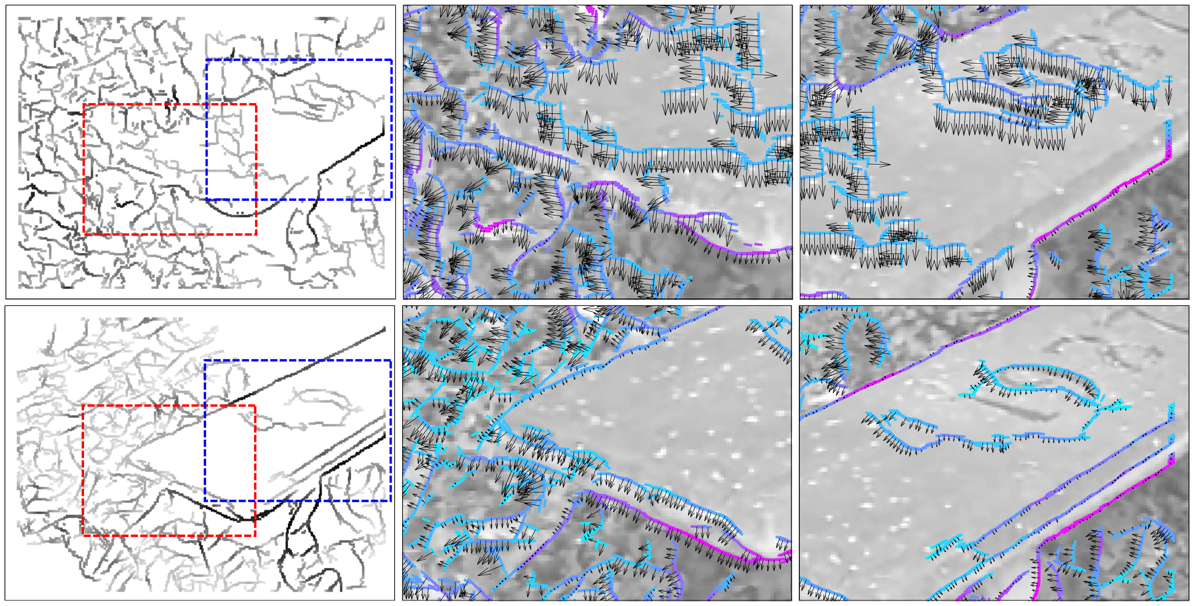
Figure 8. Results for the real sequence from the "bench" sequence in Figure 6. Our approach (bottom) finds the boundaries of the bench much more consistently despite the clutter of the ivy, and the normal speed vectors are consistent with the camera motion. The filtering approach proposes bogus motion estimates on the surface of the bench and in the background clutter.

[4] D. H. Ballard and C. M. Brown. *Computer Vision*. Prentice Hall, 1982.

[5] M. J. Black and D. J. Fleet. Probabilistic detection and tracking of motion discontinuities. *IJCV*, 38(3):231–245, 2000.

[6] K. G. Derpanis and J. M. Gryn. Three-dimensional nth derivative of gaussian separable steerable filters. In *ICIP*, volume III, pages 553–556, 2005.

[7] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *CVPR*, pages 264–271, 2003.

[8] W. T. Freeman and E. H. Adelson. The design and user of steerable filters. *PAMI*, 13(9):891–906, September 1991.

[9] D. J. Heeger. Optical flow using spatiotemporal filters. *IJCV*, 1:270–302, 1988.

[10] H. Hirschmüller, P. R. Innocent, and J. Garibaldi. Real-time correlation-based stereo vision with reduced border errors. *IJCV*, 47(1-3):229–246, April-June 2002.

[11] T. Kanade and M. Okutomi. A stereo matching algorithm with an adaptive window: Theory and experiment. *PAMI*, 16(9):920–932, September 1994.

[12] Y. Ke, R. Sukthankar, and M. Hebert. Efficient visual event detection using volumetric features. In *ICCV*, October 2005.

[13] I. Laptev and T. Lindeberg. Space-time interest points. In *ICCV*, 2003.

[14] M. Leordeanu and M. Hebert. A spectral technique for correspondence problems using pairwise constraints. In *ICCV*, 2005.

[15] E. Levina and P. Bickel. The earth mover's distance is the mallows distance: Some insights from statistics. In *ICCV*, volume 2, pages 251–256, 2001.

[16] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, January 2004.

[17] C. Mallows. A note on asympototic joint normality. *Annals of Mathematical Statistics*, 43(2):508–515, April 1972.

[18] D. R. Martin, C. C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *PAMI*, 26(5):530–549, May 2004.

[19] B. A. Maxwell and S. J. Brubaker. Texture edge detection using the compass operator. In *BMVC*, volume II, pages 549–558, September 2003.

[20] O. Nestares and D. J. Fleet. Probabilistic tracking of motion boundaries with spatiotemporal predictions. In *CVPR*, pages 358–365, 2001.

[21] P. Perona and J. Malik. Detecting and localizing edges composed of steps, peaks, and roofs. In *ICCV*, pages 52–57, 1990.

[22] M. Ruzon and C. Tomasi. Color edge detection with the compass operator. In *CVPR*, pages 160–166, June 1999.

[23] E. Shechtman and M. Irani. Space-time behavior based correlation. In *CVPR*, 2005.

[24] J. Shi and J. Malik. Motion segmentation and tracking using normalized cuts. In *ICCV*, 1998.

[25] P. Smith, T. Drummond, and R. Cipolla. Layered motion segmentation and depth ordering by tracking edges. *PAMI*, 26(4):479–494, April 2004.

[26] A. Stein and M. Hebert. Incorporating background invariance into feature-based object recognition. In *WACV*, 2005.

[27] J. Sun, Y. Li, S. B. Kang, and H.-Y. Shum. Symmetric stereo matching for occlusion handling. In *CVPR*, 2005.

[28] C. Tomasi and T. Kanade. Detection and tracking of point features. Technical Report CMU-CS-91-132, Carnegie Mellon University, April 1991.

[29] S. Ullman, M. Vidal-Naquet, and E. Sali. Visual features of intermediate complexity and their use in classification. *Nature Neuroscience*, 5(7):1–6, 2002.

[30] S. Wang, T. Kubota, J. M. Siskind, and J. Wang. Salient closed boundary extraction with ratio contour. *PAMI*, 27(4):546–561, April 2005.