Unified point-edgelet feature tracking

A Thesis
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
Electrical Engineering

---

by
Kalaivani Sundararajan
May 2011

Accepted by:
Dr. Stan Birchfield, Committee Chair
Dr. Robert Schalkoff
Dr. John Gowdy

Abstract

Feature tracking algorithms have conventionally tracked 'corner' features or windows with high spatial frequency content. However, this conventional point feature representation of scenes would be inappropriate for poorly textured image sequences like indoor image sequences. To overcome this problem, we propose a feature tracking algorithm which tracks point features and edgelets simultaneously. Edgelets are straight line approximations of intensity edges in an image. Hence, a combination of point features and edgelets provides a better representation of untextured sequences with the point features and edgelets complementing each other. We show that this property results in more robust tracking.

Tracking edgelets is challenging due to the inherent aperture problem. This thesis proposes an optical flow-based tracking method to track both point features and edgelets in a combined fashion. The aperture problem of the edgelets is overcome using Horn-Schunck regularisation to penalize flow vector deviations from those of the neighboring features. This method uses a translational motion model for tracking individual features and hence only the change in displacement of the point features and edgelets is computed.

The point features are detected using the Shi-Tomasi method. The edgelets are detected using the Canny edge map and Douglas-Peucker polyline approximation algorithm. It is assumed that motion will be constant in a neighborhood around the point feature and for all edgels in an edgelet. The point features and edgelets are tracked by minimizing an energy function consisting of the optical flow constraint and sum of negative gradient magnitude of edgels. Thus the edgelets which are typically attracted to the nearby intensity edges will be guided by the optical flow constraint equation and by the motion of the neighboring features.

We have also implemented a pyramidal implementation of our algorithm with the uppermost pyramidal level representing the original image and the lower pyramidal levels consisting of the

downsampled images. The unified feature tracking method utilizes a pyramidal implementation which respects the scale at which the features are visible. Hence, the motion vector computation at lower pyramidal levels is reliable and improves the tracking robustness. Moreover, the average flow vector due to the neighboring features is computed by fitting an affine motion model to the neighboring features. The neighboring features are weighted based on their distance from the feature and the pyramidal level at which they are visible.

Dedication

I dedicate this thesis to my parents, Gowri, Mekhala and Ganesh who have been a great support throughout.

Acknowledgments

I would like to thank my advisor, Dr. Stan Birchfield, for his valuable guidance and insight which made this thesis possible. He has been really supportive, encouraging and inspiring. I would also like to thank Dr.Robert Schalkoff and Dr.John Gowdy for being part of my committee and for their helpful suggestions.

Special thanks to Vidya Murali and all students in our research group for their support during the research. I would also like to acknowledge the immense support and encouragement given by all friends and colleagues at Soliton Technologies. Finally, I am thankful to all my friends for their support.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Feature tracking methods track different low-level features and involve minimization of an energy function pertaining to the feature attributes. Feature tracking can be used to track a target object in an image sequence and finds its application in object recognition, surveillance and human computer interaction. These applications require detection of object motion in an image sequence. Feature tracking methods are also useful for applications like 3D reconstruction, augmented reality and SLAM. These applications require feature correspondences between image frames to determine the scene geometry and camera motion. Feature tracking methods use different types of primitive features like points, lines, and contours to represent the target. However, a single feature type may not efficiently represent the image content and a combination of different feature types would provide a better representation.

1.1 Tracking primitive features

The features are chosen such that they provide a good representation of the target. Conventionally, textured images are represented using point features, man-made environments using lines, and deformable objects using contours. The feature tracking methods depends on the features used for tracking and the features themselves are chosen based on the application.

Point features are used to represent textured portions in an image with two-dimensional intensity changes. These point features are typically corners, points of occlusion, discontinuities, or regions of curvature maxima. The point features are detected using corner detectors like Harris

corner detector [12] or Shi-Tomasi method [23]. These features have sufficient information in their local neighborhood to enable localization and hence can be tracked reliably. Point feature trackers include optical flow based methods like Kanade-Lucas-Tomasi (KLT) feature tracker [25] and its variations [1].

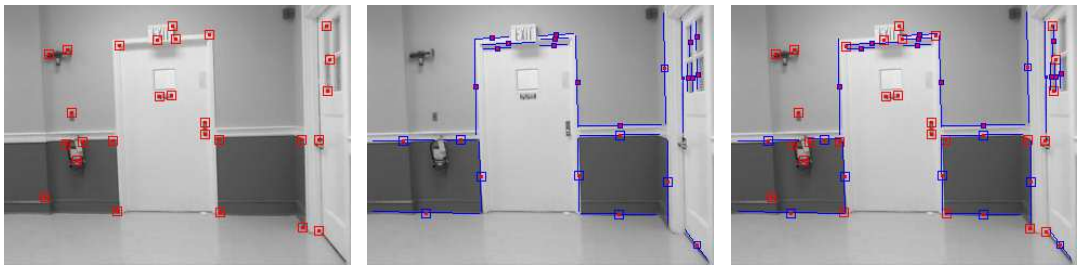
Lines and contour features are used to represent edges with one-dimensional intensity changes. Various edge detectors exist which detect edges as maxima of first order derivatives [6] or zero-crossings of second-order derivatives of image intensities. Tracking edges has been very limited due to the inherent *aperture problem*. Edges can be localized only in one dimension and hence we can obtain only the motion perpendicular to the edge. Line features are typically tracked by one dimensional search in the direction of edge gradient. For many applications, this method of tracking works reasonably well if the scene is uncluttered and displacement between frames is incremental.

Contour features represent edges with arbitrary curvature and are used to represent object boundaries. These features are used when the object shape undergoes deformation in subsequent frames. Contour tracking methods include snakes [15, 14] and level sets [18, 7, 19]. Snakes can track open and closed contours but require initialization very close to the actual edge. Level sets based tracking is used to track only closed contours and does not require initialization very close to the edge.

1.2 Why a combination of primitive features?

A combination of primitive features often provides a better representation of the image content. Each primitive feature has its own distinctive property and hence a fusion of different features provides a rich description of the image. Moreover a single primitive feature might not work well under all conditions and hence a fusion of multiple primitive features proves to be more robust. This thesis proposes a combination of point features and edgelets for tracking. Edgelets are straight line approximations of intensity edges in an image.

Point features represent textured regions in an image, can be localized easily, and can be tracked reliably. They provide a compact representation of informative regions in an image. Both motion vector components of point features can be computed reliably. However, the presence of such textured regions in an untextured environment is limited and hence the sparse representation using point features is a poor geometric representation of the scene content. Moreover, the point features



(a) Point feature representation (b) Edgelet representation (c) Combination of point features and edgelets

Figure 1.1: Different feature representations of an untextured image

are described by their local neighborhood and may undergo substantial change over multiple frames due to change in camera position.

In contrast, edgelets prove to be more useful to represent images with poor texture. Image sequences of man-made environments will have more edgelets than point features. They also provide a better insight into the geometric structure of the scene content. Edgelets are represented by one-dimensional intensity change and can be quite robust to lighting and scale changes and to motion blur. However, the invariance to all these conditions also makes them less discriminative. Moreover, when the scene is cluttered or textured, edgelets may not provide a good scene representation. Tracking edgelets under such conditions proves to be difficult due to the ambiguity in edgelet correspondences. Moreover, edgelets can be tracked reliably only in the direction of edgelet gradient and not in the direction perpendicular to the edgelet gradient.

Hence, both the point features and edgelets provide complementary information about a scene. Point features are distinctive and but undergo changes with viewpoint, while edgelets are robust to viewpoint changes but lack discrimination. Hence a combination of both edgelets and point features provide a much better representation of the scene. Tracking a combination of both these features proves to be more robust and informative regardless of whether the image content is rich in texture or poorly textured.

Figures 1.1a and 1.1b shows the point features and edgelets detected in an indoor image. It can be seen that each representation by itself does not provide complete information about the image content. Figure 1.1c shows both the point features and edgelets detected in an image. It can be observed from this figure that point features and edgelets complement each other and provide a rich description of the actual image content.

1.3 Unified feature tracking and its applications

This thesis proposes an optical-flow based feature tracking method to track point features and edgelets in an unified manner. In optical flow algorithms, point features can be tracked reliably since both flow vector components can be determined accurately. However, edgelets suffer from the aperture problem and hence only the motion vector component in the direction of the edgelet gradient can be computed reliably. The motion vector component tangential to the edgelet cannot be computed reliably.

The unified feature tracking approach overcomes the aperture problem of the edgelets by letting the point features and edgelet features guide each other's motion. The unified tracking approach makes use of the motion coherence property which ensures that features close to each other will have similar motion. This motion coherence property enables smoother flows for features, and also guides features which might otherwise stray away due to occlusion/disocclusion or image noise.

The point features are tracked by minimizing an energy function whose data term represents the optical flow constraint. The edgelets are tracked by minimizing an energy function with two data terms: the sum of the negative gradient magnitude of the edgelets, and optical flow constraint for the edgelets. The former term helps to snap on to any intensity edges in the proximity while the latter term guides the edgelets towards nearby intensity edges with matching gradient orientation. The former term also makes it robust to lighting changes.

The inherent aperture problem of the edgelets is overcome using a regularization term which penalizes large deviations in flow vector components of the neighboring features. Therefore, the tangential motion component of the edgelets is filled in from those of the neighboring features. Moreover, the point features and edgelets interact and guide each other in the motion vector computation, thereby providing smoother flow.

The unified feature tracking approach should be valuable for many computer vision applications that use feature tracking. This method provides relatively denser and better geometric representation of scene content compared to feature tracking methods which are either point-based or line-based. This method does not require any prior model to describe the edges or to assist in edge correspondences. The unified tracking method provides reliable edge correspondences inherently. Structure from motion applications which use a combination of straight line and point features

can directly benefit from this approach. Moreover, object tracking algorithms will also benefit since this approach provides a better representation of the object.

1.4 Thesis outline

The next chapter provides a brief summary of various unified feature tracking methods in the literature, along with their advantages and limitations. Chapter 3 provides a detailed explanation of our unified feature tracking approach and related concepts. Chapter 4 discusses the experimental results of our approach, effect of various algorithm parameters, and benchmark results with respect to standard feature tracking methods. Chapter 5 gives an overall summary of the thesis and the scope for future work.

Chapter 2

Related Work

While most feature tracking methods have focused upon tracking point features, some attention has been paid to tracking intensity edges. Most notably is the work on contour-based tracking [14], in which a closed parameterized contour is tracked using primarily intensity edge information. Other research on contour analysis aims to compute contour flow between two images [16]. Various authors have looked at the problem of matching line segments [22, 26] outside the context of tracking.

Schmid and Zisserman [22] proposed a line matching method given the fundamental matrix of image pairs or trifocal tensors of image triplets. Given two images, they propose using epipolar constraints to obtain point correspondences between line segments. For short range motion, the match score is computed using the average of neighborhood correlation for every edgel. For long range motion, the neighborhood around edgels might have undergone transformations. Hence, the neighborhood around each edgel is warped using a homography computed using the fundamental matrix. The average correlation of corrected neighborhood for all edgels is computed as match score. In contrast, our method does not use any a priori information for line matching.

Liu et al. [16] proposed a contour tracking method for tracking objects with no visible texture. They disregard point features since spurious features might mislead the contours. Hence, they propose a global motion estimation technique which utilizes information from three levels of contour analysis: edgelets, boundary fragments and contours. Boundary fragments are detected in the first frame and are broken in short edgelets. The probabilities of motion vector of every edgelet is computed by searching for the edgelet in the neighborhood. The boundary fragments are then

grouped into contours using a graphical model and importance sampling to resolve uncertainities in motion vector computation. In our method, we compute the motion vectors of edgelets using local matching and motion vectors of the neighboring features. Hence, any ambiguity in the motion vector computation of edgelets is resolved using optical flow of neighboring features.

Wang et al. [26] proposed a line matching method using mean-standard deviation line descriptor (MSLD). They use no a priori information in the line matching method. For every pixel on the line segment, they define a pixel support region (PSR) to define its neighborhood. The gradient information of each sub-region of PSR is encapsulated as gradient description matrix (GDM). MSLD is built by computing the mean and standard deviation of GDM column vectors. The line correspondences are made by matching the MSLD of line segments. This method works well only if there is sufficient texture in the PSR of edgels. However, for untextured image sequences, line segments cannot be distinguished due to their inherently poor discriminative property. In contrast, edgelets used in our method will be guided by optical flow and by the motion of neighboring features therby reducing the number of erroneous edge correspondences.

Various model-based tracking approaches [21],[28],[20] have been proposed using edgelets. These methods require a known 3D edge model of target for edgelet tracking or matching. Given a prior target pose, the 3D edges are projected onto the image plane, and edgelet correspondences are searched for in the vicinity. Multiple hypotheses of matching edgelets are then resolved using maximum likelihood formulation in order to recover the pose in the current image frame.

Some recent structure-from-motion research, based on the early work of [27, 9], reconstruct scenes using lines as the primitives. Eade and Drummond [11] propose a method to use edge landmarks in an environment for SLAM. 3D edgelets are projected onto the image plane using the camera pose, and the edgelet locations are updated using the Kalman filter. Smith et al. [24] combine FAST point features and Sobel edgelets for real-time SLAM in an Extended Kalman filter framework, but the edgelets are only allowed if they connect two point features. Bartoli et al. [3] perform structure from motion from lines using Plücker coordinates and manually entered line correspondences. All of these approaches assume a static scene.

Crowley et al. [9] proposed a method to obtain structure from motion by using edgelet correspondences. They use a parametric representation of edgelets comprising of edgelet center, orientation and half-length of the edgelet and perpendicular distance of line segment from origin. The edgelet correspondences between images is obtained using the predict-match-update cycle of

a Kalman filter. These edgelet correspondences are then used to obtain sturcture and motion parameters.

Eade and Drummond [11] proposed a method to use edge landmarks in an environment for SLAM. Their system uses a single calibrated camera to update camera pose and landmark estimates. The edgelets are represented as short straight line segments with two 3D vectors representing the edgelet center and edgelet direction. The edgelets are detected as group of edgels with gradient magnitude maximal along the gradient direction in image blocks of 16×16 pixels. To obtain edgelet correspondences, the edgelet locations are predicted using the camera pose which projects 3D edgelets onto the image plane. Given the predicted location, the correspondence is searched for in the vicinity and matching is done using an approximation of Hough transform. If clutter is present in the edgelet neighborhood, there might be multiple correspondence hypotheses. To disambiguate the correspondence, all hypotheses are incorporated into the maximum likelihood data association phase. As the edgelets are observed in subsequent image frames, the edgelet center and direction are updated using Kalman filter.

Smith et al. [24] proposed a method to use combination of point features and edgelets for real-time SLAM. They detect point features using FAST feature detector and edgels are detected using Sobel edge detector. All possible point feature pairs are tested for presence of edgelet between them using the edgels. This allows combination of point features and edgelets to be used in real-time SLAM. The point features and edgelets are then incorporated into Extended Kalman filter framework to obtain SLAM.

The purpose of this work is to detect and track line features (edgelets) in a unified manner along with point features, without making strong assumptions about the scene (such as a prior model, or a static scene). The focus is upon a practical, nearly real-time system applicable to long image sequences. In this regard, it is similar in spirit to the earlier work of Chiba and Kanade [8] which accomodates broken or closely spaced lines in a pyramidal optical flow implementation.

Chiba and Kanade [8] proposed a line tracking method which accomodates broken or closely spaced lines. They used pyramidal optical flow implementation to get a good prediction of line segments. They do not use point features for tracking but use the flow vector information of textured regions to fill-in for regions with unidirectional or homogeneous texture. This was done by thresholding the optical flow field for reliable flow vectors and dilation of flow vector field to fill-in the missing flow vector components. After the line segment locations are predicted, they use line gra-

dient direction to distinguish between closely spaced lines and a similarity measure which allows broken lines to be matched. In our method, instead of dilation of optical flow fields, an affine motion model is fit to the neighboring features to compute the expected displacement for features whose flow vector components are missing.

As discussed above, it can be observed that many methods using edgelets for better scene representation have assumed a priori model or epipolar constraints. Methods which do not assume any model have used edgelets for reconstruction of static scenes only. In contrast, the proposed method aims at tracking point features and edgelets without any prior model assumption or constraints. Moreover, this method can also be used for tracking features in sequences with moving objects.

Chapter 3

Approach - Unified point-edgelet feature tracking

This chapter discusses our unified point-edgelet feature tracking algorithm. We provide a brief discussion of optical flow, aperture problem, and the two optical flow paradigms: sparse and dense optical flow. We then discuss the joint tracking method proposed by Birchfield and Pundlik [4]. Following that, we describe our unified point-edgelet feature tracking algorithm and the choice of various algorithm parameters.

3.1 Optical flow and aperture problem

Optical flow has been widely used in computer vision for motion estimation. This section describes some of the differential methods for optical flow computation.

3.1.1 Optical flow

Optical flow is the apparent motion of brightness pattern in an image sequence. This apparent motion corresponds to the 2D projection of object motion in the 3D world. The optical flow computation involves three basic assumptions:

- **Brightness constancy** - Image intensities in small regions will remain the same although their location may change.

- **Spatial coherence** - Neighboring pixels in an image typically belong to the same surface and hence have similar motions.
- **Temporal persistence** - Image motion of a surface patch changes gradually over time.

Optical flow methods are used to determine the motion between two consecutive frames taken at time t and $t + \delta t$ respectively. Consider a pixel *(x,y)* in an image taken at time t with intensity *I(x,y,t)*. The pixel will have moved by δx, δy within a time interval δt. Using the brightness constancy assumption, the image constraint equation can be given as,

$$I(x, y, t) = I(x + \delta x, y + \delta y, t + \delta t). \tag{3.1}$$

The differential methods used for optical flow computation use Taylor series expansion to linearize $I(x + \delta x, y + \delta y, t + \delta t)$. Assuming that the image motion is small, $I(x + \delta x, y + \delta y, t + \delta t)$ can be linearized as

$$I(x + \delta x, y + \delta y, t + \delta t) = I(x, y, t) + \frac{\partial I}{\partial x}\partial x + \frac{\partial I}{\partial y}\partial y + \frac{\partial I}{\partial t}\partial t + H.O.T. \tag{3.2}$$

Using the equations 3.1 and 3.2,

$$\frac{\partial I}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial I}{\partial y}\frac{\partial y}{\partial t} + \frac{\partial I}{\partial t} = 0. \tag{3.3}$$

Thus the resulting optical flow equation is given by,

$$I_x u + I_y v + I_t = 0, \tag{3.4}$$

where I_x, I_y and I_t are partial derivatives of image intensity with respect to x, y, and t respectively. u and v are the x and y components of optical flow:

$$u = \frac{\partial x}{\partial t} \quad v = \frac{\partial y}{\partial t}.$$

3.1.2 Aperture problem

It can be observed that (3.4) represents a single equation with two unknowns, u and v. This underconstrained nature of the problem is called aperture problem in optical flow algorithms. The intuitive explanation is that, when the motion of a certain pixel is viewed through a small aperture, we would be able to recover only the flow vector component in the direction of pixel gradient and not the component in the perpendicular direction.

In order to compute the motion completely, additional constraints need to be added to (3.4). This gives rise to two paradigms in optical flow computation - sparse and dense optical flow. The sparse optical flow methods [17] are local methods and compute optical flow only for pixels that have sufficient texture in the neighborhood. These pixels do not suffer from aperture problem and hence the complete motion information can be obtained reliably. These pixels are identified as *corner* features by Harris and Stephens [12] and Shi-Tomasi features [23]. The dense optical flow methods [13] are global methods and compute optical flow for every pixel in the image.

Lucas and Kanade in [17] used regression to impose additional constraints. They assumed that all pixels in an image patch will undergo same motion. The optical flow constraint equations of these pixels are solved using least squares method to obtain the flow vector components. However, this method can be used only for pixels with sufficient texture in the neighborhood ([12],[23]). The unknown motion vector, $\vec{\mathbf{u}} = [u, v]^T$, is assumed to be constant in a certain neighborhood with sufficient texture. The motion vector $\vec{\mathbf{u}}$ is computed by iteratively minimizing the energy function

$$E_{LK}(u, v) = K_\rho * (I_x u + I_y v + I_t)^2 , \tag{3.5}$$

where K_ρ is denotes convolution with an integration window of size ρ.

Horn and Schunck in [13] use regularization to impose additional constraints. They impose a global smoothness term assuming that neighboring pixels have similar motion. They determine global flow fields, *u(x,y)* and *v(x,y)*, which minimizes the global energy functional

$$E_{HS}(u, v) = \int_\Omega (I_x u + I_y v + I_t)^2 + \lambda(|\nabla u|^2 + |\nabla v|^2) dx dy, \tag{3.6}$$

where $\lambda > 0$ is the regularization parameter and Ω is the image domain. This method enables a 'filling-in' effect for pixels with unidirectional or no texture in the neighborhood since one or none of

the flow vectors can be computed reliably for these pixels. The regularizer $|\nabla u|^2 + |\nabla v|^2$ penalizes large deviations in flow vector components from that of the neighboring pixels and hence fills in the missing flow vector components from the neighborhood.

3.2 Joint tracking of features and edges

Both local and global optical flow estimation techniques have their own merits and demerits. The global methods like Horn and Schunck approach yields dense optical flow field but is not robust to noise. The local methods like Lucas-Kanade approach yields sparse flow field but is robust to noise. Research efforts have gone into combining the ideas of Lucas-Kanade and Horn-Schunck as in [5] and [4]. Bruhn et al. in [5] proposed a method that combines the optical flow estimation of Horn-Schunck and the Lucas-Kanade thereby yielding a dense optical flow field that are robust to noise. Inspired by this idea, Birchfield and Pundlik [4] proposed a joint tracking method to track corner features and edges wherein the motion of every feature is guided by its neighboring features. In this method, both corner features, and edges with unidirectional texture are represented as point features. The feature detection in this method does not disregard edges for their unidirectional texture and chooses point features which lie on the intensity edges. Hence, the features which are tracked in this method represent only point features that lie on textured regions and intensity edges.

Sparse optical flow methods have conventionally tracked point features independently thus neglecting motion coherence which serves an important role in determining the motion of a feature. The joint tracking method utilizes the motion coherence property by combining the optical flow methods of Lucas-Kanade [17] and Horn-Schunck[13]. In the Horn-Schunck method, the optical flow of every pixel is influenced by the neighboring pixels. Following the same idea, in the joint tracking method, the point features are not tracked independently but the motion of every feature is influenced by that of the neighboring features.

3.2.1 Feature detection

The joint tracking method [4] uses a feature detection technique which is different from the Shi-Tomasi feature detection method. Shi-Tomasi technique chooses features such that the minimum eigenvalue of gradient covariance matrix is greater than a threshold. This ensures that both the eigenvalues of the gradient covariance matrix are large and features with multidirectional

texture are chosen. The joint tracking method however does not disregard edges for its unidirectional texture. The features are selected using $max(e_{max}, \eta e_{min})$, where e_{max} and e_{min} are the maximum and minimum eigenvalues of the gradient covariance matrix, $\eta < 1$ is a scale factor. This method of feature selection allows point features with unidirectional texture also to be chosen. Hence, the point features representing textured regions and edges are treated uniformly in this tracking method.

3.2.2 Feature tracking

The joint feature tracking method [4] combines the optical flow methods proposed by Lucas-Kanade and Horn-Schunck. The data term is similar to that of the Lucas-Kanade method. A regularization term, similar to that in Horn-Schunck method, is added to penalize the deviation of displacement of a feature from its expected value. The expected value of feature displacement is computed by fitting a motion model to the displacement of its neighboring features. The addition of regularization term enables features with unidirectional texture also to be tracked reliably. This is due to the fact that the regularization term provides for the missing flow vector components from the flow vectors of the neighboring features. Hence point features representing textured regions and edges are treated uniformly for tracking in the joint tracking method. The features and edges are tracked by minimizing the energy functional,

$$E_{JLK} = \sum_{i=1}^{N} E_D(i) + \lambda_i E_S(i), \tag{3.7}$$

where N is the number of point features and the data and smoothness terms are given by,

$$E_D(i) = K_\rho * (I_x u + I_y v + I_t)^2 \tag{3.8}$$

$$E_S(i) = ((u_i - \hat{u}_i)^2 + (v_i - \hat{v}_i)^2). \tag{3.9}$$

In these equations, the energy of the feature i is determined by how well its displacement $(u_i, v_i)^T$ matches the local image data, as well as how far the displacement deviates from the expected displacement $(\hat{u}_i, \hat{v}_i)^T$. Because the features are sparse, a significant difference in the motion of neighboring features is not uncommon. Hence the expected displacement cannot be obtained by simply averaging the displacment of the neighboring features. The expected displacement $(\hat{u}_i, \hat{v}_i)^T$

of a feature is predicted by fitting an affine motion model to the displacements of neighboring features weighted according to their distance to the feature i.

A $2N \times 2N$ sparse matrix is formed in which every feature i is represented by a 2×2 diagonal block as,

$$Z_i \mathbf{u_i} = \mathbf{e_i}, \tag{3.10}$$

where

$$Z_i = \begin{bmatrix} \lambda_i + K_\rho * I_x^2 & K_\rho * (I_x I_y) \\ K_\rho * (I_x I_y) & \lambda_i + K_\rho * I_y^2 \end{bmatrix}$$

$$e_i = \begin{bmatrix} \lambda_i \hat{u}_i - K_\rho * (I_x I_t) \\ \lambda_i \hat{v}_i - K_\rho * (I_y I_t) \end{bmatrix}$$

This sparse system of equations can be solved using Jacobi iterations of the form,

$$\tilde{u}_i^{(k+1)} = \tilde{u}_i^k - \frac{J_{xx}\tilde{u}_i^k + J_{xy}\tilde{v}_i^k + J_{xt}}{\lambda_i + J_{xx} + J_{yy}} \tag{3.11}$$

$$\tilde{v}_i^{(k+1)} = \tilde{v}_i^k - \frac{J_{xy}\tilde{u}_i^k + J_{yy}\tilde{v}_i^k + J_{yt}}{\lambda_i + J_{xx} + J_{yy}} \tag{3.12}$$

where, $J_{xx} = K_\rho * I_x^2, J_{xy} = K_\rho * (I_x I_y), J_{yy} = K_\rho * (I_y^2), J_{xt} = K_\rho * (I_x I_t)$ and $J_{yt} = K_\rho * (I_y I_t)$.

3.2.3 Pyramidal implementation

A pyramidal implementation of the joint tracking method [4] is used to handle large displacements between frames. Since the motion of features influence each other, the displacement of all the features is computed at every pyramidal level for every iteration. The updated motion vectors of the neighboring features is then used to guide the motion vector computation of a feature in the next iteration. The joint feature tracking algorithm is summarized below:

Algorithm:

For each feature i,

1. Initialize $u_i = (0, 0)^T$
2. Initialize λ_i

For pyramid level n-1 to 0 step -1,

(a) For each feature i, compute Z_i

(b) Repeat until convergence:

For each feature i,

i. Determine $\hat{u}_i$

ii. Compute the difference I_t between the first image and the shifted second image:

$$I_t(x, y) = I_1(x, y) - I_2(x + u_i, y + v_i)$$

iii. Compute e_i

iv. Solve $Z_i u_i' = e_i$ for incremental motion u_i'

v. Add incremental motion to overall estimate: $u_i := u_i + u_i'$

3. Expand to the next level: $u_i := \alpha u_i$, where α is the pyramid scale factor

This method uses the same regularization parameter λ_i for all features with multidirectional and unidirectional texture even though some velocity components of these features can be determined reliably without regularization.

3.3 Unified point-edgelet feature tracking

Inspired by the joint feature tracking method [4], we have developed the unified point-edgelet feature tracking method to simultaneously track point features and edgelets. This method provides a much better representation of the image content as compared to the joint tracking method since the image intensity edges are represented as linear segments called edgelets. In the method proposed by Birchfield and Pundlik in [4], the intensity edges are still represented as point features with unidirectional texture in the neighborhood. Therefore, multiple such point features would be needed to represent an intensity edge. Moreover, each of these features will yield similar but different flow vectors for different edgels belonging to the same edge. In this thesis, the edges are approximated as edgelets and it is assumed that all edgels of the edgelet will be displaced by the same amount. Hence, the features which are tracked in this method include point features representing textured regions, and edgelets representing intensity edges.

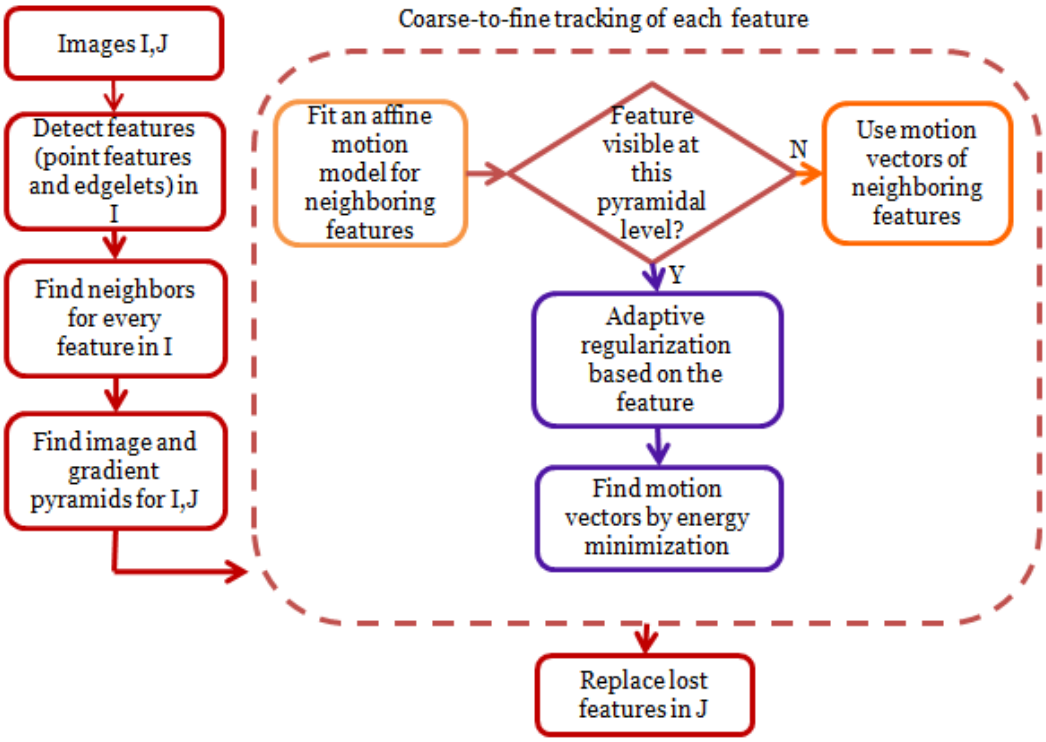


Figure 3.1: Unified point-edgelet feature tracking - An overview.

Moreover, for both point features and edgelets, at least one of the flow vector components can be determined reliably without additional constraints. Hence, a adaptive regularization is used to provide missing flow vectors only for the flow vector components which suffer from aperture problem. We use a pyramidal implementation of our algorithm to handle large displacements. The original image represents the uppermost pyramidal level and the downsampled images are represented in the lower pyramidal levels. We also propose tracking these features in scale space to ensure robust tracking since some of these features may be smoothed away at the lower pyramidal levels.

Figure 3.1 shows a brief overview of our approach given two image frames I and J. The point features and edgelets are detected in frame I and the lowest pyramidal level at which they can be detected is computed. The neighboring features for every feature in I is determined based on the distance between the features. The neighboring features are weighted based on their distance to the feature and the lowest pyramidal level at which they can be detected. The features are then tracked starting from the lowest pyramidal level. At every pyramidal level, for every feature, the affine motion model of the neighboring features is computed. If the feature is visible at that pyramidal level, then it is tracked by minimizing an energy function with adaptive regularization based on

the feature. If the feature is not visible at that pyramidal level, then it follows the motion of its neighboring features. The motion vectors computed at every pyramidal level serve as initialization for the motion vectors computed at the next higher pyramidal level.

3.3.1 Detection of point and edgelet features

The point features and edgelets are detected in the first frame and tracked over the subsequent frames. The point features are pixel locations with multidirectional texture in its neighborhood and are detected using the method proposed by Shi and Tomasi [23]. The edgelets are straight line segments with unidirectional texture in its neighborhood. The edgelets are detected using Canny edge map and Douglas-Peucker polyline approximation [10].

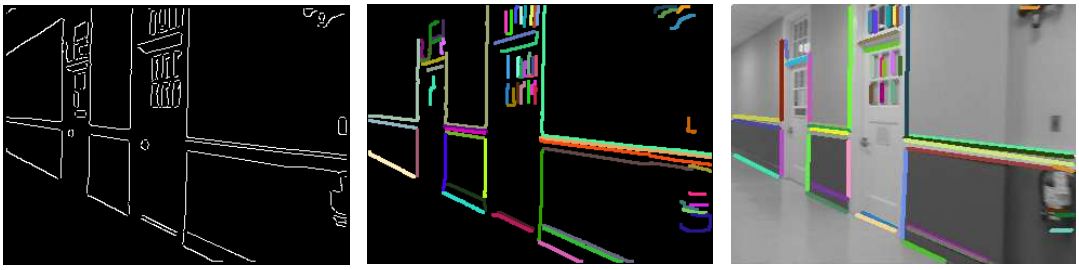
3.3.1.1 Point feature detection

The set of Shi-Tomasi point features is represented as $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^{N_P}$, where $\mathbf{x}_i = (x_i, y_i)$ is the image location of the ith point feature, and N_P is the number of point features. The Shi-Tomasi features are detected by computing the gradient covariance matrix for every pixel and its neighborhood. Pixels, with gradient covariance matrices whose minimum eigenvalue is greater than a predefined threshold, are accepted as features. The gradient covariance matrix, Z_i, for pixel i is given by,

$$Z_i = \begin{bmatrix} K_\rho * I_x^2 & K_\rho * (I_x I_y) \\ K_\rho * (I_x I_y) & K_\rho * I_y^2 \end{bmatrix}$$

where I_x and I_y are image intensity gradients with respect to x and y, K_ρ denotes convolution with integration window of size ρ.

The gradient covariance matrix of a pixel must be above the image noise level and well-conditioned to ensure reliable tracking. The noise requirement requires that both eigenvalues of Z_i must be large while the conditioning requirement means that they cannot differ by several orders of magnitude. Two small eigenvalues denote a textureless homogeneous region in the image. One large and one small eigenvalue indicates unidirectional texture. Two large eigenvalues denote corners and other textured portions in an image. In practice, when the smaller eigenvalue is large enough to meet the noise requirements, the matrix Z_i is also well-conditioned. Since the intensity variations in a window are bounded by the maximum allowable pixel value, the greater eigenvalue cannot be



(a) Canny edge map with suppressed corner edgels

(b) Linking connected edgelets

(c) Edgelets: Output of Douglas-Peucker algorithm

Figure 3.2: **Left:** Canny edge map. Note that some corner edgels show up even after suppression. **Center:** Connected edgels are linked into a single edgel chain. Each linked edgel chain is denoted by a unique color. **Right:** The Douglas-Peucker line approximation recursively splits the linked edgel chain into edgelets. Each edgelet is denoted by a unique color. Edgelets with length ≤ 15 have been suppressed.

arbitrarily large. Hence, if λ_1 and λ_2 are the two eigenvalues of Z_i, we accept a window as a trackable feature if,

$$\min(\lambda_1, \lambda_2) > \tau,$$

where τ is a predefined threshold.

3.3.1.2 Edgelet detection

The edgelets are straight line segments which approximate the intensity edges in an image. The edgelets are obtained from the binary edge map produced by the Canny edge detector. Since corners, junctions, and other textured point features have already been found using Shi-Tomasi method, we suppress the edgels which exhibit such property. The edgels, with gradient covariance matrices having minimum eigenvalue greater than a predefined threshold, are considered as corners and junctions. These corners and junctions are then suppressed in the binary edge map leaving out only the edgels with unidirectional texture.

The remaining edgels are grouped based on 4- or 8-connectivity with their neighboring edgels and a labelled edge map is produced. The labelled edgels are then collated into an ordered chain of edgels. This process is called *edge linking*. Figure 3.2b shows the labelled edge map of linked edgels. All the linked edgels are represented by a unique color.

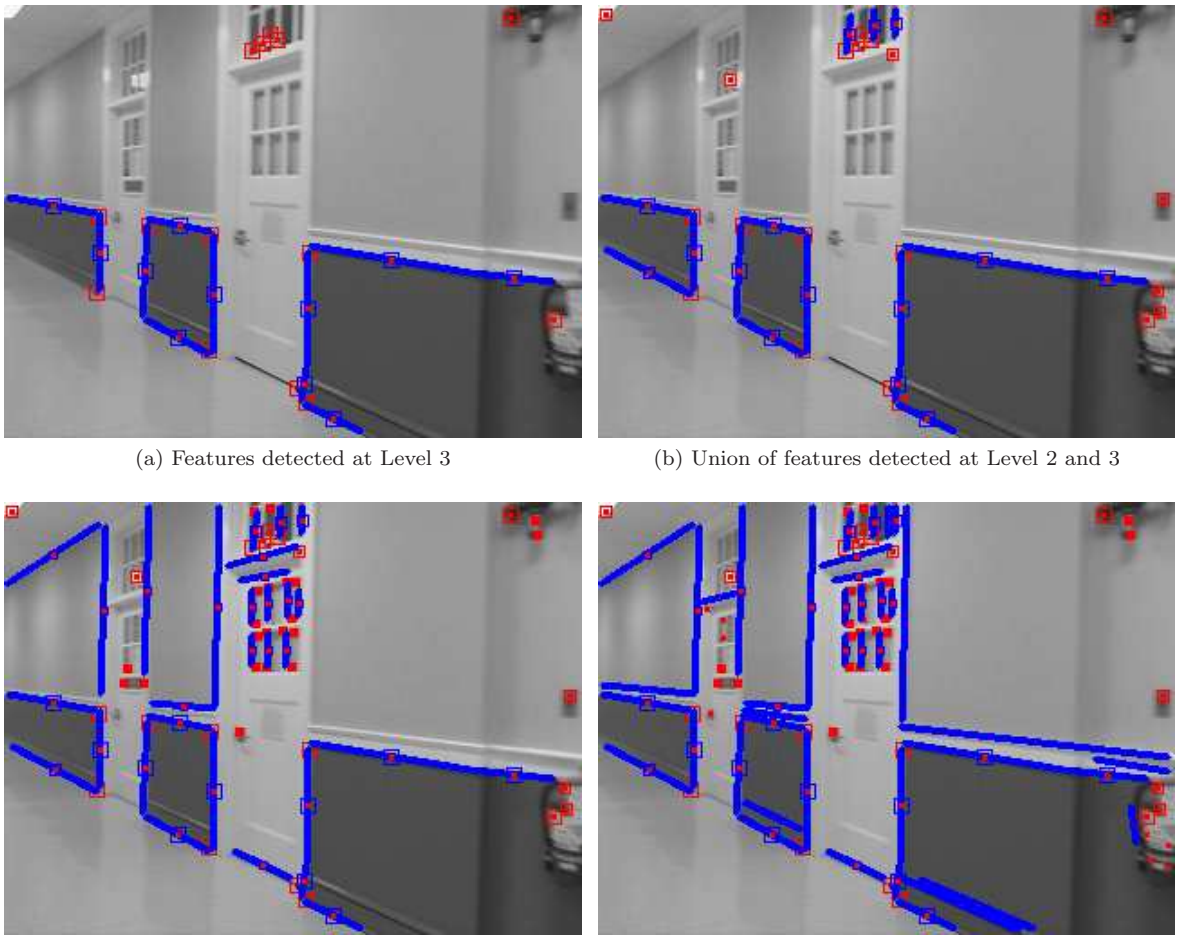
The linked edgels may represent a continuous curve and can be further split into shorter straight line segments called edgelets. The Douglas-Peucker algorithm is applied recursively to

the linked edgels to obtain a polyline approximation of the linked edgels. The Douglas-Peucker algorithm provides points of maximum curvature at which the curve can be split. The curve can now be represented as multiple edgelets and these points of maximum curvature form the endpoints of edgelets. The center (x_c, y_c), the orientation θ and length l of every edgelet is then computed using the endpoints of the edgelet. The set of edgelets $\mathcal{E} = \{\mathbf{e}_j\}_{j=1}^{N_E}$, where N_E is the number of edgelets, is the union of the line segments from all the polylines found by Douglas-Peucker subject to a minimum length $\ell_{min} = 15$ pixels. The jth edgelet is represented by $\mathbf{e}_j = (x_{cj}, y_{cj}, \theta_j, \ell_j)$, which contains its center (x_{cj}, y_{cj}), angle $0 \leq \theta_j < \pi$ (clockwise from the positive x axis), and length $\ell_j > 0$ in pixels. Figure 3.2c shows the linked edgels represented as edgelets.

3.3.1.3 Scale space detection

In the pyramidal implementation, let the original image represent the uppermost pyramidal level (level 0) and the downsampled images represent the lower pyramidal levels (level1, level2, ..). In the pyramidal implementation of the existing feature tracking algorithms, all the features are tracked at all pyramidal levels. If some features are smoothed away at a certain pyramidal level, tracking those features at that level could lead to erroneous flow vector computation. Moreover, since the flow vectors computed at lower pyramidal levels are used to initialize flow vectors at higher pyramidal levels, any error in the flow vector computation at the lower pyramidal level would propagate to the higher pyramidal levels as well. In the unified feature tracking method, the point features and edgelets influence each other in their motion computation. Hence, when the features are tracked in the pyramidal implementation, the motion vectors computed at the lower pyramidal levels must be reliable. Therefore, when we detect the point features and edgelets, we also compute the lowest pyramidal level at which they can be detected.

The image pyramid is computed by successively smoothing images using a 3×3 Scharr kernel and resampling images by a factor of 2. The point features and edgelets are detected at the uppermost pyramidal level as described in Section 3.3.1.1 and 3.3.1.2. To compute the lowest pyramidal level at which the point features can be detected, the closest pixel at every pyramidal level is computed. The minimum eigenvalue of the gradient covariance matrix for the closest pixel at that pyramidal level is computed. If the minimum eigenvalue of the gradient covariance matrix at the closest pixel is greater than a predefined threshold, the point feature is said to be visible at that pyramidal level. Similarly, the lowest pyramidal level at which an edgelet can be detected is



(a) Features detected at Level 3

(b) Union of features detected at Level 2 and 3

(c) Union of features detected at Level 1,2,and 3

(d) Union of features detected at Level 0,1,2, and 3

Figure 3.3: Features detected at four different pyramidal levels. Level 0 represents the original image with the union of features detected at all pyramidal levels.

computed by finding the closest pixel of edgelet center at every pyramidal level. The Canny edge detector is applied to every pyramidal level. If the closest pixel lies on the Canny edge map (± 1 pixel), then the edgelet is said to be visible at that pyramidal level.

Figure 3.3 shows features visible at four different pyramidal levels. The point features are represented in red and edgelets in blue. The size of the square denotes the lowest pyramidal level at which the point features and edgelets can be detected. The bigger the square, the lower the pyramidal level at which they can be detected.

3.3.2 Tracking of point and edgelet features

The point features and edgelets are tracked by minimizing an energy function similar to (3.7). In the unified feature tracking method, the point features are tracked by minimizing an energy function whose data term represents the optical flow constraint. The edgelets are tracked by minimizing an energy function consisting of two data terms: the optical flow constraint, and the sum of inverse gradient magnitude of edgels. Hence, the edgelets which are typically attracted to the nearest edges will also be guided by the optical flow constraint equation. A smoothness term is added to the energy function for both point features and edgelets to enforce regularization. Hence, the motion of point features and edgelets are guided by that of the neighboring features. This leads to more robust tracking of point features, and edgelets in particular which suffer from the aperture problem.

The goal of tracking is to compute the displacements $U_P = \langle (u_{Pi}, v_{Pi}) \rangle_{i=1}^{N_P}$ of the point features and the displacements $U_E = \langle (u_{Ej}, v_{Ej}) \rangle_{j=1}^{N_E}$ of the edgelets. The displacements are measured with respect to the previous image frame I and the current frame J, where $I(\mathbf{x})$ and $J(\mathbf{x})$ are the intensities of the pixel $\mathbf{x} = (x, y)$ in the two frames, respectively. We propose to minimize the following energy functional:

$$E_{UPE}(U_P, U_E; \mathcal{X}, \mathcal{E}, I, J) = \sum_{i=1}^{N_P} (E_{DP}(i) + E_{SP}(i)) + \sum_{j=1}^{N_E} (E_{DE}(j) + E_{SE}(j)). \tag{3.13}$$

The point features are tracked using the optical flow constraint, while edgelets are tracked using both the optical flow constraint and the gradient magnitude, as seen in the two data terms:

$$E_{DP}(i) = \sum_{\mathbf{x} \in \mathcal{W}_P(i)} \left(f(\mathbf{x}, u_{Pi}, v_{Pi}; I, J)\right)^2 \tag{3.14}$$

$$E_{DE}(j) = \sum_{\mathbf{x} \in \mathcal{W}_E(j)} \left(f(\mathbf{x}, u_{Ej}, v_{Ej}; I, J)\right)^2 + \left(g(\mathbf{x}, u_{Ej}, v_{Ej}; J)\right)^2, \tag{3.15}$$

where (u_{Pi}, v_{Pi}) is the displacement of the ith point feature, (u_{Ej}, v_{Ej}) is the displacement of the jth edgelet, $\mathcal{W}_P(i)$ is an integration window of size $m \times m$ around the ith point feature, and $\mathcal{W}_E(j)$ is an integration window of size $m \times \ell_j$ along the jth edgelet. We set $m = 7$ for all experiments. The functions f and g are the optical flow constraint equation and the inverse gradient magnitude,

respectively:

$$f(\mathbf{x}, u, v; I, J) = I_x(\mathbf{x})u + I_y(\mathbf{x})v + I_t(\mathbf{x}) \tag{3.16}$$

$$g(\mathbf{x}, u, v; J) = G(x + u, y + v), \tag{3.17}$$

where $G(x, y) = m(J) - \|\nabla J(x, y)\|$, and $m(J) = \max_{\mathbf{x}} \|\nabla J(\mathbf{x})\|$ is the maximum gradient magnitude over the whole image used to ensure that $G(x, y) \geq 0$. The spatial derivatives are $I_x(\mathbf{x}) = \partial I(\mathbf{x})/\partial x$ and $I_y(\mathbf{x}) = \partial I(\mathbf{x})/\partial y$, and the temporal derivative is $I_t(\mathbf{x}) = \partial I(\mathbf{x})/\partial t \approx J(\mathbf{x}) - I(\mathbf{x})$.

As in the joint tracking approach [4], smoothness terms enforce regularization to allow the motion of features to be guided by those of neighboring features. This leads to more robust tracking of both point features and edgelets, particularly for the latter which suffer inherently from the aperture problem that prevents the tangential flow vector component along the edge from being computed by local information alone. The smoothness terms are

$$E_{SP}(i) = \lambda_{Pui}(u_{Pi} - \hat{u}_{Pi})^2 + \lambda_{Pvi}(v_{Pi} - \hat{v}_{Pi})^2 \tag{3.18}$$

$$E_{SE}(j) = \lambda_{Euj}(u_{Ej} - \hat{u}_{Ej})^2 + \lambda_{Evj}(v_{Ej} - \hat{v}_{Ej})^2, \tag{3.19}$$

where $(\hat{u}_{Pi}, \hat{v}_{Pi})$ is the expected displacement of the ith point feature based on the displacements of its neighbors, and similarly for $(\hat{u}_{Ej}, \hat{v}_{Ej})$. These values are computed by fitting an affine motion model to the neighboring features using a constant radius neighborhood around the point features and endpoints / centerpoints of edgelets. The affine model is computed using weighted least squares, in which the neighboring features are weighted based on two criteria: their distance from the feature for which the expected displacement is being computed, and the coarsest scale at which they are detected. Features detected at coarser pyramidal levels are assigned higher weights since they yield more reliable flow vectors. In addition, the coarse-to-fine implementation ignores features that are only detected at scales finer than the current scale.

Differentiating (3.13) with respect to the unknown displacements yields a $2N \times 2N$ sparse matrix equation, where $N = N_P + N_E$ is the total number of features. By convention, we stack the point features above the edgelets in the equation, so that the $(2i - 1)$th and $(2i)$th rows for point

features are $Z_{Pi}\mathbf{w}_{Pi} = \mathbf{r}_{Pi}$, where

$$Z_{Pi} = \begin{bmatrix} \lambda_{Pui} + \sum_{\mathbf{x}\in\mathcal{W}_P(i)} I_x^2(\mathbf{x}) & \sum_{\mathbf{x}\in\mathcal{W}_P(i)} I_x(\mathbf{x})I_y(\mathbf{x}) \\ \sum_{\mathbf{x}\in\mathcal{W}_P(i)} I_x(\mathbf{x})I_y(\mathbf{x}) & \lambda_{Pvi} + \sum_{\mathbf{x}\in\mathcal{W}_P(i)} I_y^2(\mathbf{x}) \end{bmatrix} \tag{3.20}$$

$$\mathbf{w}_{Pi} = \begin{bmatrix} u_{Pi} \\ v_{Pi} \end{bmatrix} \qquad \mathbf{r}_{Pi} = \begin{bmatrix} \lambda_{Pui}\hat{u}_{Pi} - \sum_{\mathbf{x}\in\mathcal{W}_P(i)} I_x(\mathbf{x})I_t(\mathbf{x}) \\ \lambda_{Pvi}\hat{v}_{Pi} - \sum_{\mathbf{x}\in\mathcal{W}_P(i)} I_y(\mathbf{x})I_t(\mathbf{x}) \end{bmatrix}. \tag{3.21}$$

For edgelets the $(2N_P + 2j - 1)$th and $(2N_P + 2j)$th rows are $Z_{Ej}\mathbf{w}_{Ej} = \mathbf{r}_{Ej}$, where

$$Z_{Ej} = \begin{bmatrix} \lambda_{Euj} + \sum_{\mathbf{x}\in\mathcal{W}_E(j)} (I_x^2(\mathbf{x}) + G_x^2(\mathbf{x})) & \sum_{\mathbf{x}\in\mathcal{W}_E(j)} (I_x(\mathbf{x})I_y(\mathbf{x}) + G_x(\mathbf{x})G_y(\mathbf{x})) \\ \sum_{\mathbf{x}\in\mathcal{W}_E(j)} (I_x(\mathbf{x})I_y(\mathbf{x}) + G_x(\mathbf{x})G_y(\mathbf{x})) & \lambda_{Evj} + \sum_{\mathbf{x}\in\mathcal{W}_E(j)} (I_y^2(\mathbf{x}) + G_y^2(\mathbf{x})) \end{bmatrix} \tag{3.22}$$

$$\mathbf{w}_{Ej} = \begin{bmatrix} u_{Ej} \\ v_{Ej} \end{bmatrix} \qquad \mathbf{r}_{Ej} = \begin{bmatrix} \lambda_{Eui}\hat{u}_{Ej} - \sum_{\mathbf{x}\in\mathcal{W}_E(j)} (I_x(\mathbf{x})I_t(\mathbf{x}) + G_x(\mathbf{x})G(\mathbf{x})) \\ \lambda_{Evi}\hat{v}_{Ej} - \sum_{\mathbf{x}\in\mathcal{W}_E(j)} (I_y(\mathbf{x})I_t(\mathbf{x}) + G_y(\mathbf{x})G(\mathbf{x})) \end{bmatrix}. \tag{3.23}$$

In these expressions $\lambda_{Pui}, \lambda_{Pvi}, \lambda_{Euj}, \lambda_{Evj} > 0$ are regularization parameters governing the amount of smoothing. The computation of these parameters will be discussed momentarily. In practice, the energy functional is minimized using Gauss-Seidel iterations or successive over-relaxation in a Horn-Schunck manner, as in [4].

3.3.2.1 Pyramidal implementation

The pyramidal implementation of unified feature tracking is novel compared to Section 3.2.3. All the features are not tracked at all pyramidal levels. Only the features that are detectable at a certain pyramidal level or lower are tracked at that pyramidal level. If they are not detectable, they just follow the neighboring features and use $(\hat{u}_{Pi}, \hat{v}_{Pi})$ or $(\hat{u}_{Ej}, \hat{v}_{Ej})$ as their motion vector. The pyramidal implementation for unified tracking can be summarized as follows:

Algorithm:

For each point feature $\mathbf{x_i}$ or edgelet $\mathbf{e_j}$,

1. Initialize $\mathbf{w}_{Pi} = \mathbf{w}_{Ej} = (0,0)^T$

   For pyramid level n-1 to 0 step -1,

   (a) Repeat until convergence:

   For each feature,

   i. Determine $(\hat{u}_{Pi}, \hat{v}_{Pi})$ for point feature or $(\hat{u}_{Ej}, \hat{v}_{Ej})$ for edgelet

   ii. If point feature $\mathbf{x_i}$ or edgelet $\mathbf{e_j}$ is visible at this pyramidal level

   A. For point feature, compute Z_{Pi} with $\lambda_{Pui} = 0$ and $\lambda_{Pvi} = 0$. For edgelet, compute Z_{Ej} with $\lambda_{Euj} = 0$ and $\lambda_{Evj} = 0$

   B. Compute eigenvalues and eigenvectors of Z_{Pi} or Z_{Ej} to determine regularization factors

   C. Substitute λ_{Pui} and λ_{Pvi} in Z_{Pi} or λ_{Euj} and λ_{Evj} in Z_{Ej}

   D. Compute $\mathbf{r}_{Pi}$ or $\mathbf{r}_{Ej}$

   E. Solve $Z_{Pi}\mathbf{w}'_{Pi} = \mathbf{r}_{Pi}$ or $Z_{Ej}\mathbf{w}'_{Ej} = \mathbf{r}_{Ej}$ for incremental motion $\mathbf{w}'_{Pi}$ or $\mathbf{w}'_{Ej}$ respectively.

   iii. Else if feature not visible at this level, $\mathbf{w}'_{Pi} = (\hat{u}_{Pi}, \hat{v}_{Pi})^T$ or $\mathbf{w}'_{Ej} = (\hat{u}_{Ej}, \hat{v}_{Ej})^T$

   iv. Add incremental motion to overall estimate: $\mathbf{w}_{Pi} := \mathbf{w}_{Pi} + \mathbf{w}'_{Pi}$ or $\mathbf{w}_{Ej} := \mathbf{w}_{Ej} + \mathbf{w}'_{Ej}$

2. Expand to the next level: $\mathbf{w}_{Pi} := \alpha\mathbf{w}_{Pi}$ or $\mathbf{w}_{Ej} := \alpha\mathbf{w}_{Ej}$, where α is the pyramid scale factor

3.3.2.2 Flow vector computation of neighboring features

The expected displacmenets of every point feature or edgelet, $(\hat{u}_{Pi}, \hat{v}_{Pi})$ or $(\hat{u}_{Ej}, \hat{v}_{Ej})$, is not computed by simply averaging the flow vectors of the neighboring features. Since sparse features are used, the neighboring features may all not have the same flow vectors. Hence, an affine motion model is fit to the neighboring features to obtain the expected displacments. Hence, only those neighboring features which are detectable at that pyramidal level or lower level should be taken into account while computing $(\hat{u}_{Pi}, \hat{v}_{Pi})$ or $(\hat{u}_{Ej}, \hat{v}_{Ej})$. This ensures that the values computed for the expected displacements are reliable. Moreover, the flow vectors of the neighboring features are weighted while computing the expected displacements. The neighboring features are weighted not just based on their distance from the point feature/ edgelet but also based on the scale at which they are detectable. The weights of neighboring point features or edgelets are calculated based on

the function,

$$wt(\mathbf{x}_i) = \exp\left(\frac{s(\mathbf{x}_i)}{N_{PL}} - \gamma\frac{d(\mathbf{x}_i)^2}{2\sigma^2}\right), \tag{3.24}$$

$$wt(\mathbf{e}_j) = \exp\left(\frac{s(\mathbf{e}_j)}{N_{PL}} - \gamma\frac{d(\mathbf{e}_j)^2}{2\sigma^2}\right), \tag{3.25}$$

where $s(\mathbf{x}_i)$ or $s(\mathbf{e}_j) \in \{0, 1, .., N_{PL} - 1\}$ are the lowest pyramidal level at which the features are detectable, N_{PL} is the number of pyramidal levels, $d(\mathbf{x}_i)$ or $d(\mathbf{e}_j)$ is the Euclidean distance between the neighboring feature and the current feature for which the expected displacment is computed, $\sigma = 10$ is the standard deviation of Gaussian function used in (3.24) and (3.25) and γ is the relative weight between the two criteria. $\gamma = 0.5$ has been chosen in this work. $d(\mathbf{x}_i)$ or $d(\mathbf{e}_j)$ is computed using the location of point features and center/endpoints of the edgelets. It can be noted that from (3.24) and (3.25), the lower the pyramidal level at which the neighboring feature is detectable, the higher the weight. This is based on the assumption that features detectable at lower pyramidal levels give reliable flow vectors in the pyramidal implementation.

3.3.2.3 Choice of regularization parameter

For both point features and edgelets, both or at least one of the motion vector components can be computed reliably. For edgelets, only the flow vector component in the direction of edgelet gradient can be computed reliably. The flow vector component tangential to the edgelet cannot be determined reliably. Hence, the estimation of flow vector component in that direction has to be regularized more than the direction in which the flow vector can be computed reliably.

Instead of using fixed regularization parameters, as is commonly done, we use adaptive regularization in the motion estimation of both point features and edgelets. For point features, if the minimum eigenvalue of the gradient covariance matrix is greater than the threshold used for Shi-Tomasi [23] detection, then minimal regularization is used: $\lambda_{Pui} = \lambda_{Pvi} = 0.01$. Otherwise, more regularization is used: $\lambda_{Pui} = \lambda_{Pvi} = 50$.

For edgelets, the regularization is not only adaptive but also non-isotropic to force the smoothing to be applied primarily in the direction perpendicular to the gradient. The regularization values are set as $\lambda_{Euj} = \max(|50\ell_j \cos\theta_j|, 0.01)$ and $\lambda_{Evj} = \max(50\ell_j \sin\theta_j, 0.01)$. The length ℓ_j of

the edgelet is needed since the regularization parameter is added to summations over $\mathcal{W}_E(j)$ values that depend on the length, in (3.22) and (3.23). The absolute value is needed in case $\pi/2 < \theta_j < \pi$, and the minimum value of 0.01 ensures that some regularization is always applied. The dependency on θ_j causes the smoothing to be applied primarily along the edgelet rather than across the edgelet. For example, a vertical edgelet is pulled vertically by its neighbors, but its horizontal motion is determined primarily by local image data. Similarly, a horizontal edgelet is pulled horizontally by its neighbors, but its vertical motion is determined primarily by local image data.

Chapter 4

Results

The unified point-edgelet feature tracking algorithm was implemented using the Blepo computer vision library and Microsoft Visual C++. The feature tracking algorithm was tested on the standard Middlebury optical flow datasets, and the average angular error and average endpoint error were measured. The feature tracking algorithm was also tested on man-made indoor environments like hallways and rooms and on a few natural sequences. These images were acquired using hand-held camera or camera mounted on a robot with significant scale changes and large displacements. The unified feature tracking algorithm was tested on Intel dual-core 2.2GHz processor with 3GB RAM.

The features were tracked well over multiple frames if there are sufficient features in the neighborhood to guide the motion computation. It was observed that isolated features, especially edglets, suffer from aperture problem. The features also get straddled when objects undergo occlusion/disocclusion. The feature tracking algorithm was tested for various algorithm parameters, and the results have been presented in this chapter. The point features are represented by red dots, the edgelets by blue/green lines. The yellow lines indicate the direction and magnitude of motion vectors. The square around the point feature or edgelet centre represents the scale at which the features are visble. The bigger the square, the lower the pyramidal levels at which the features are visible.

4.1 Comparison with ground truth and other methods

The unified point-edgelet feature tracking algorithm was tested on five Middlebury optical flow datasets: Rubberwhale, Venus, Dimetrodon, Urban2, and Urban3. The algorithm was compared with the ground truth results given in the Middlebury optical flow site [2]. The algorithm was also compared with two similar feature tracking algorithms: standard Lucas-Kanade feature tracking method, and the joint tracking of features and edges. We also tested these algorithms on a synthetic image sequence of a hallway. Frame 2 of the hallway sequence was obtained by shifting Frame 1 to right by 7 pixels and down by 5 pixels. Rubberwhale, Venus and Dimetrodon sequences have sufficient texture and contain more point features than edgelets. Urban2, Urban3 and Hallway sequences represent man-made environments and contain more edgelets than point features.

All the algorithms were tested with three pyramidal levels and twenty maximum iterations. The neighborhood features are chosen using a radius of 30 pixels. Each of the three feature tracking methods use a different feature detection method. Hence, we choose features such that each feature detection method accounts for few thousand pixels in the image. Therefore, the feature detection methods for standard Lucas-kanade and joint tracking method would detect few thousand point features. The feature detection for unified feature tracking would detect point features and edgelets such that the point features and edgels in the edgelets sum upto few thousand pixels. The images with the tracked features for all these tracking methods are shown in Figures 4.1a to 4.3f.

For all the three feature tracking methods, we obtained two performance metrics: average endpoint error and average angular error. For the point features, we obtain endpoint error and angular error by comparing the ground truth results at that pixel location. However, for the edgelet features, we follow a different method. Since we use a parametric representation for the edgelets, the edgels can be obtained only by discretization. Edgelets may sometimes represent motion boundaries and some of the discrete edgels may lie on the other side of the motion boundaries. So we obtain endpoint error and angular error by comparing the ground truth results at that pixel location and its eight neighbors. The average endpoint error and average angular error for these sequences has been presented in Table 4.1.

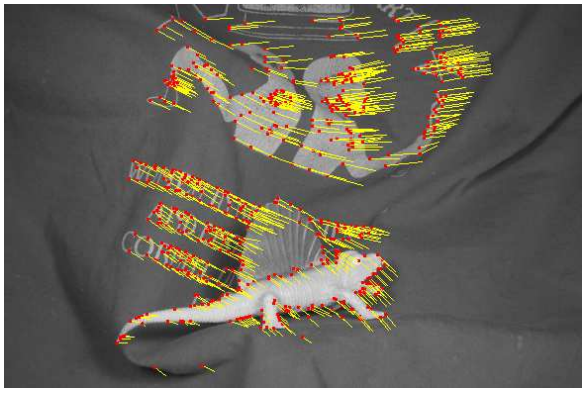
As can be observed from Table 4.1, the unified point-edgelet feature tracking method provides far better results as compared to the standard Lucas-Kanade method for all the image sequences. Moreover, the results are comparable to that of the joint feature tracking method for

Error		Average endpoint error			Average angular error		
Image	Pixels	LK	JFT	UPE	LK	JFT	UPE
Venus	3000	1.1	0.8	0.5	11.5	14.3	9.6
Rubberwhale	2000	0.4	0.2	0.3	6.4	6.4	12.6
Dimetrodon	2000	0.2	0.2	0.1	3.6	2.5	2.7
Urban2	5000	4.6	1.5	1.6	11.8	11.4	5.2
Urban3	5000	6.9	2.2	2.2	17.7	11.3	2.7
Hallway	2000	1.8	0.9	0.3	3.3	2.8	1.3

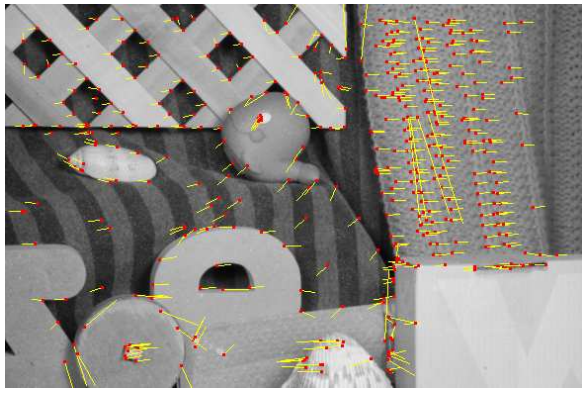
Table 4.1: Average Endpoint Error (in pixels) and Average Angular Error (in degrees) for five Middlebury image pairs. The tracking methods used were standard Lucas-Kanade (LK), joint feature tracking (JFT), and unified point-edgelet feature tracking (UPE). All three algorithms were tested with three pyramid levels and twenty maximum iterations.

Dimetrodon, Venus and Rubberwhale sequences. For the Urban2, Urban3 and Hallway sequences, the unified feature tracking method performs better than the joint tracking method. This is due to the choice of neighboring features. In the joint tracking method, all features within a certain radius of a feature are chosen as neighboring features. The expected displacements of point features and edgelets is computed by fitting an affine motion model to these features. However, all these features are point features and do not capture the geometry of the image content completely. In the unified feature tracking method, we choose the neighboring features based on the geometry of the point features and edgelets. All point features and edgelets with endpoints within a certain radius of the point feature and all point features and edgelets with endpoints within a certain radius of the edgelet centre and endpoints are chosen as neighboring features. The expected displacements of point features and edgelets is computed by fitting an affine motion model as in the joint feature tracking method. Hence, it can be seen that more neighboring features are chosen based on feature geometry and hence the flow vector computation is more regularized.

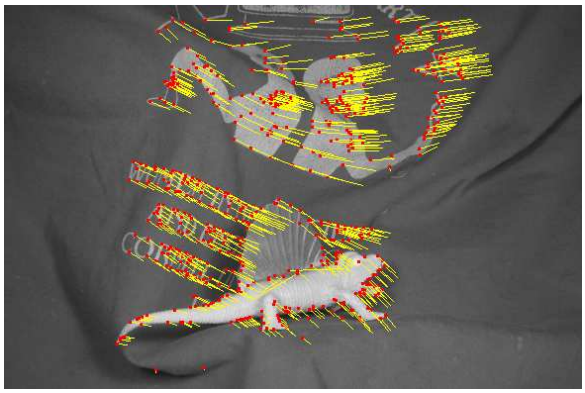
It can also be observed from Figures 4.2f, 4.3e, and 4.3f that a combination of edgelets and point features provide better representation of man-made environments. It can also be observed that these images contain multiple closely spaced edgelets which are tracked reliably using the unified feature tracking method. The edgelets which are typically attracted to nearby edges are guided by optical flow and regularization, and hence provide reliable flow estimates for these edgelets. It can also be observed from Figure 4.2f that the unified feature tracking algorithm provides a meaningful representation of poorly textured image sequences. The unified feature tracking algorithm currently does not handle errors caused due to motion discontinuities. Hence, edgelets which lie



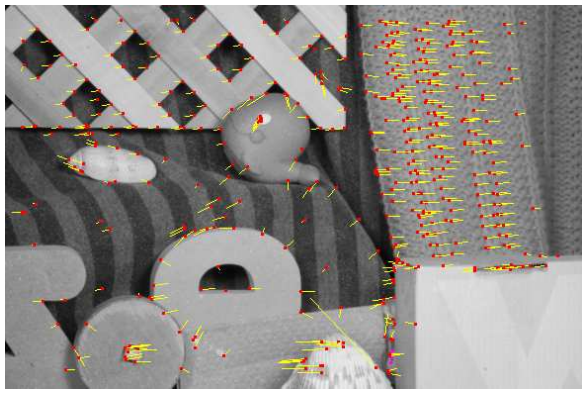
(a) Dimetrodon: Standard Lucas-Kanade



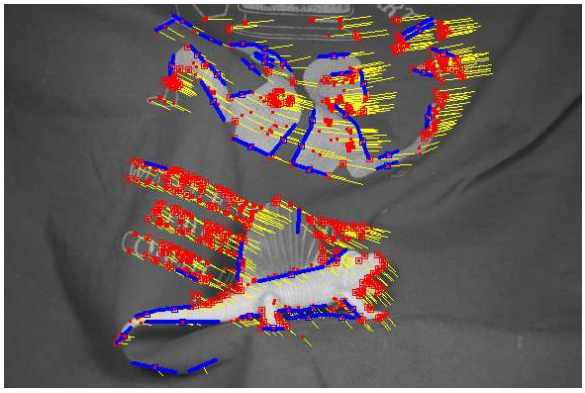
(b) Rubberwhale: Standard Lucas-Kanade



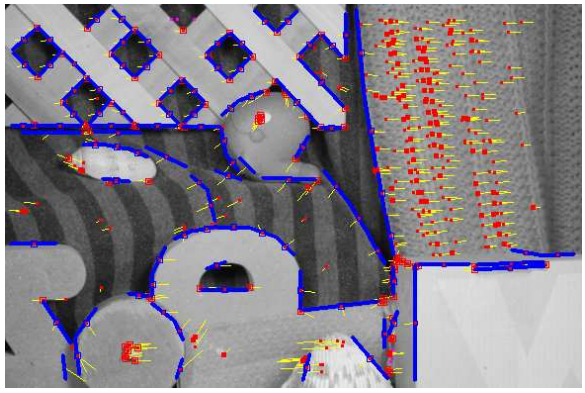
(c) Dimetrodon: Joint Lucas-Kanade



(d) Rubberwhale: Joint Lucas-Kanade

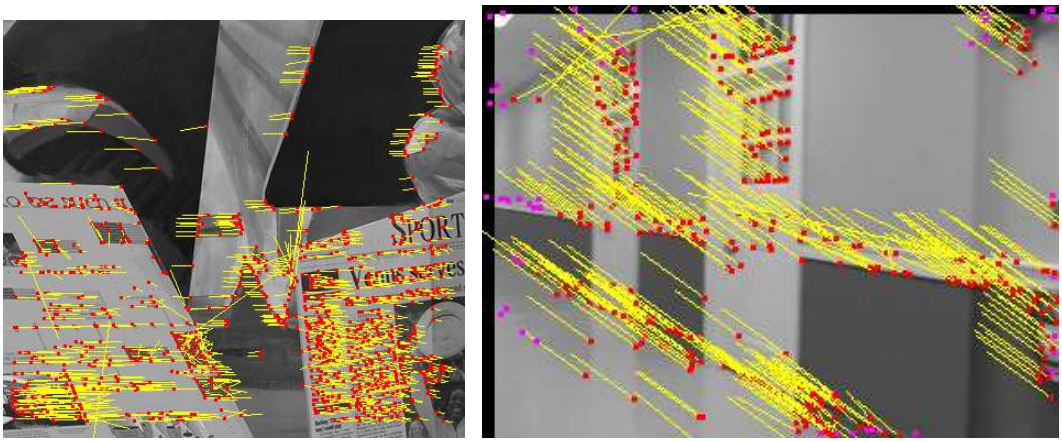


(e) Dimetrodon: Unified feature tracking



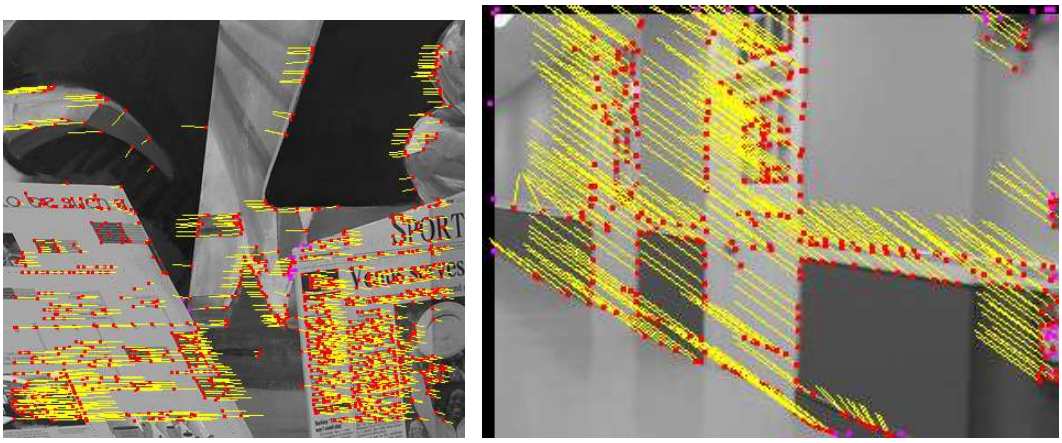
(f) Rubberwhale: Unified feature tracking

Figure 4.1: Results of different feature tracking methods for Dimetrodon and Rubberwhale.



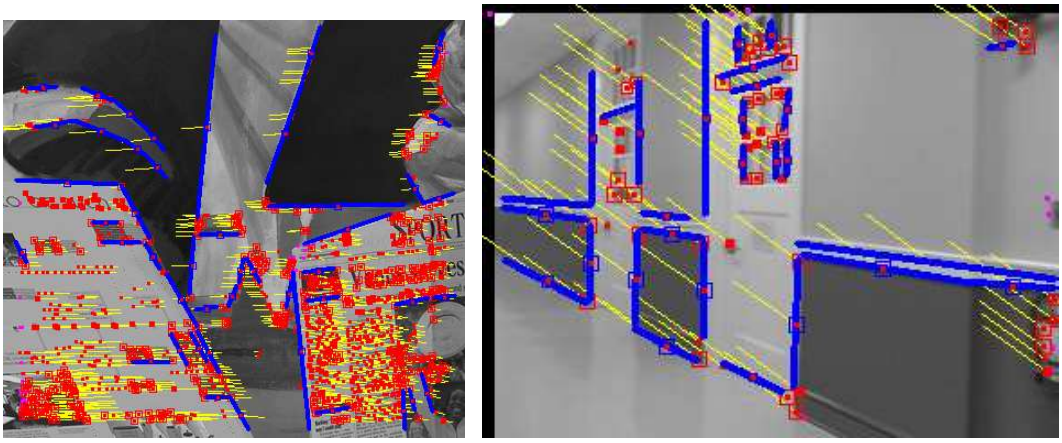
(a) Venus: Standard Lucas-Kanade

(b) Hallway: Standard Lucas-Kanade



(c) Venus: Joint Lucas-Kanade

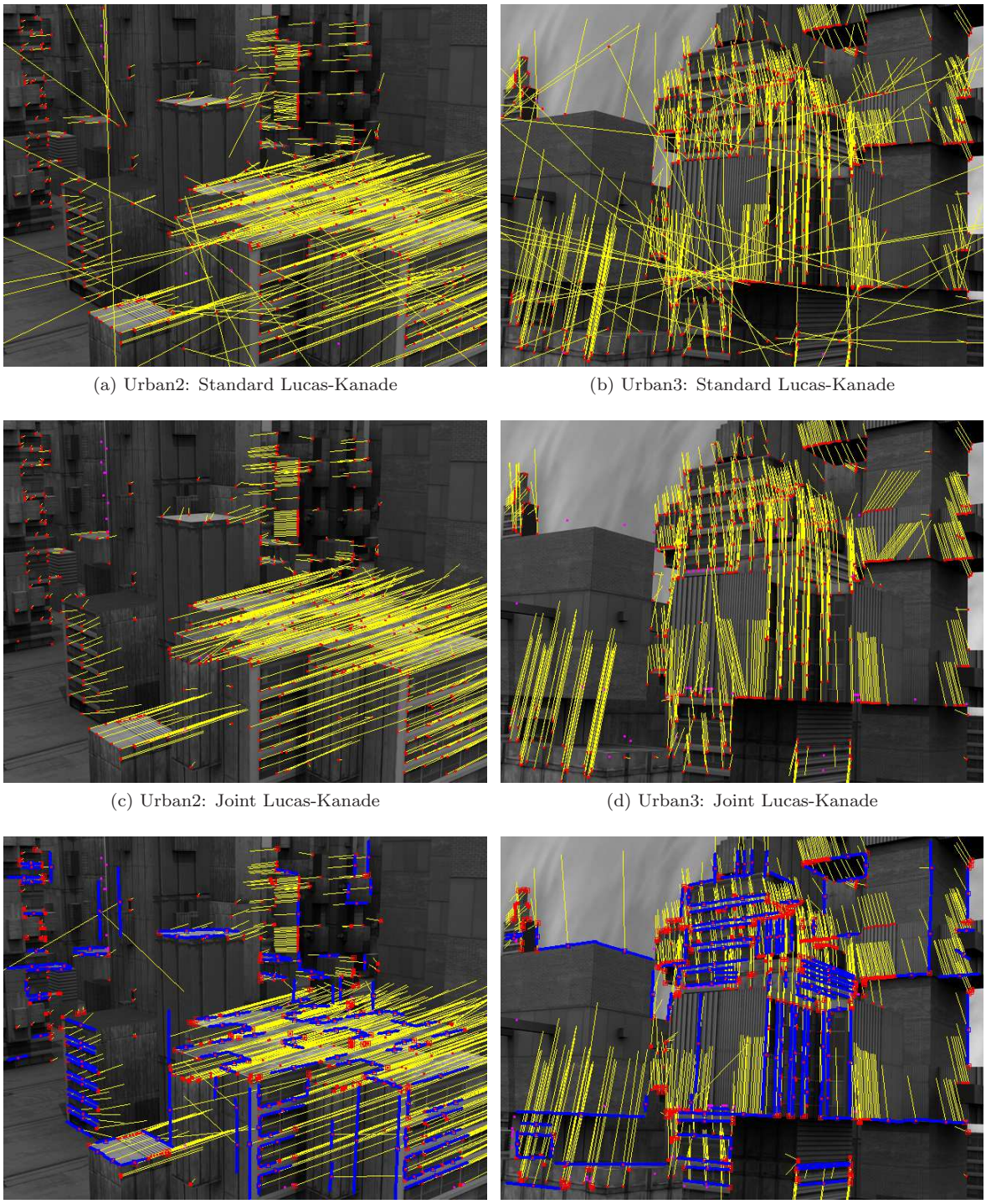
(d) Hallway: Joint Lucas-Kanade



(e) Venus: Unified feature tracking

(f) Hallway: Unified feature tracking

Figure 4.2: Results of different feature tracking methods for Venus and Hallway.



(a) Urban2: Standard Lucas-Kanade

(b) Urban3: Standard Lucas-Kanade

(c) Urban2: Joint Lucas-Kanade

(d) Urban3: Joint Lucas-Kanade

(e) Urban2: Unified feature tracking

(f) Urban3: Unified feature tracking

Figure 4.3: Results of different feature tracking methods for Urban2 and Urban3.

on motion boundaries are influenced erroneously by their neighboring features. This can be observed from Figures 4.1f and 4.3e where the edgelets stray away due to motion discontuinuities and occlusion/disocclusion.

4.2 Dense feature representation

The unified feature tracking method tracks point features and edgelets using regularization. Since this method obtains optical flow information for all pixels with at least unidirectional texture, it produces a dense optical flow field when compared to the standard Lucas-Kanade method. The joint tracking method and unified feature tracking method can be used to compute dense optical flow fields when the number of point features and edgelets are increased. We performed several tests on Middlebury optical flow datasets to evaluate the performance of the unified feature tracking method for dense optical flow computation.

We measured the error metrics, average endpoint error and average angular error, for standard Lucas-Kanade method, joint tracking method and unified feature tracking method by increasing the number of features. For the standard Lucas-Kanade method and joint feature tracking method, the first N good point features were chosen, where N is the number of pixels to be represented using features. For the unified feature tracking method, the edgelets were chosen such that the sum of edgels would account for 0.8*N and the point features would account for 0.2*N. For dense optical flow computation, we take into consideration all the edgels in an edgelet. It was assumed that all edgels in an edgelet will undergo same motion and hence have same optical flow information. The edgels that constitute an edgelet were obtained using edgelet parameters and Bresenham line algorithm. The point features and edgels were compared with the ground truth results and the error metrics were measured for the three feature tracking methods. A plot of the average endpoint/angular error with respect to the number of pixels with valid optical flow information was obtained. Figures 4.5a to 4.5d show the performance of the three feature tracking methods when the optical flow field becomes dense.

Venus, Rubberwhale and Dimetrodon image sequences have sufficient texture and hence point feature tracking algorithms like standard Lucas-Kanade and joint feature tracking method perform well when the optical flow field grows denser. It can be observed from Figures 4.5a and 4.5c that the average endpoint error and angular error are comparable for all three methods even

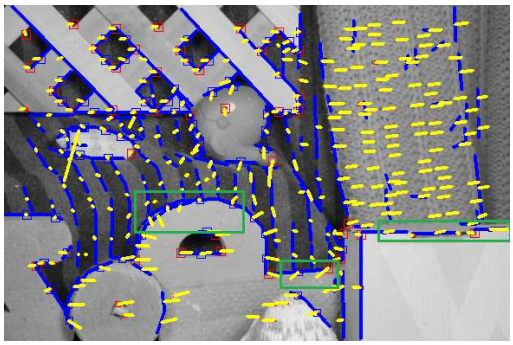
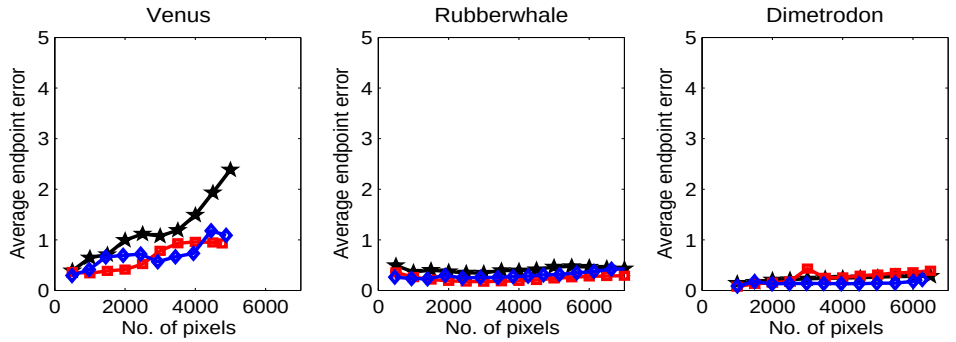


Figure 4.4: The yellow lines denote the motion vectors. The box and shell at the bottom right move to the right while the textured cloth behind them moves to the left. Similarly the objects in the lower portion move to the left while the blanket behind them moves to the right. Notice that edgelets of these objects within the green boxes are moving in the wrong direction since they lie on motion boundaries. This is due to the influence of the neighboring features which move in the opposite direction.

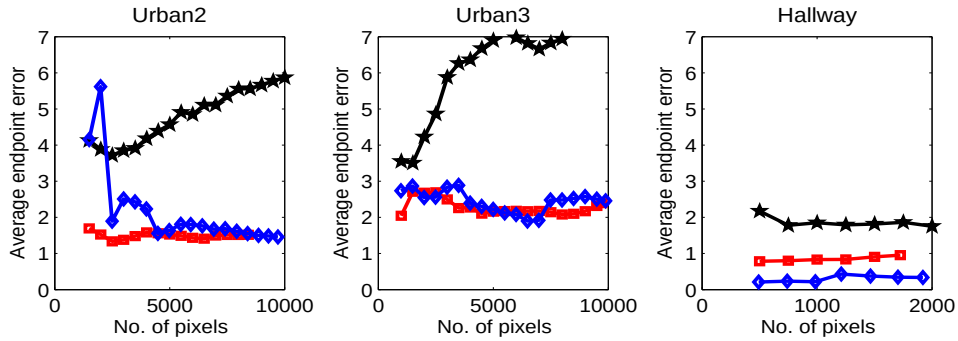
when the number of pixels with valid optical flow information increases. The angular error of unified feature tracking method for the Rubberwhale sequence is greater when compared to the other two methods. This is due to the fact that some of the edgelets which lie on motion boundaries are influenced by the erroneous motion of their neighboring features as seen in Figure 4.4. The features shown within the green boxes are moving in the wrong direction since they lie on motion boundaries. Hence, all the edgels that constitute these edgelets are steered in the wrong direction thereby causing high angular error.

Urban2, Urban3 and Hallway sequences represent man-made environments and can be meaningfully represented using point features and edgelets. It can be observed from Figures 4.5b and 4.5d that the standard Lucas-Kanade method performs poorly on these sequences. The joint tracking method and unified feature tracking method perform comparably on these sequences. In the untextured Hallway sequence, the unified feature tracking algorithm performs better than the joint tracking method. It was also observed from these images that the unified feature tracking method provides the better results compared to the other feature tracking methods when motion discontinuities are absent as in Urban3 or Hallway sequence.

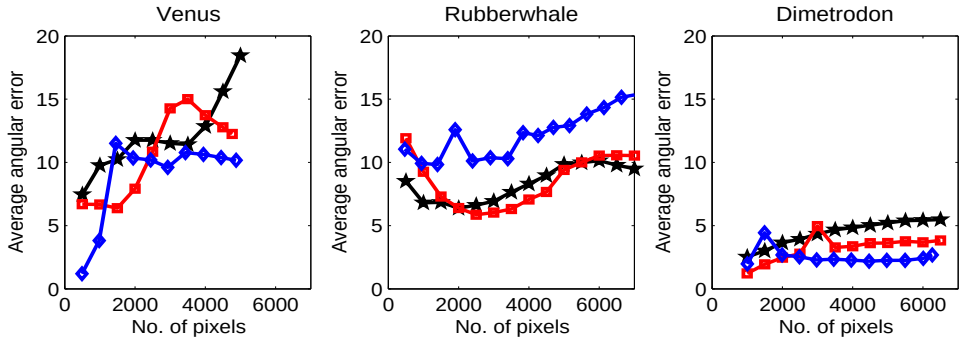
The dense feature representation of the three feature tracking methods for the Rubberwhale sequence has been shown in Figures 4.6a to 4.6l. Each figure shows the features representing 1000, 3000, 5000 and 7000 pixels in the image. It can be inferred from these images that the unified feature tracking method provides a meaningful representation of image content as compared to the other



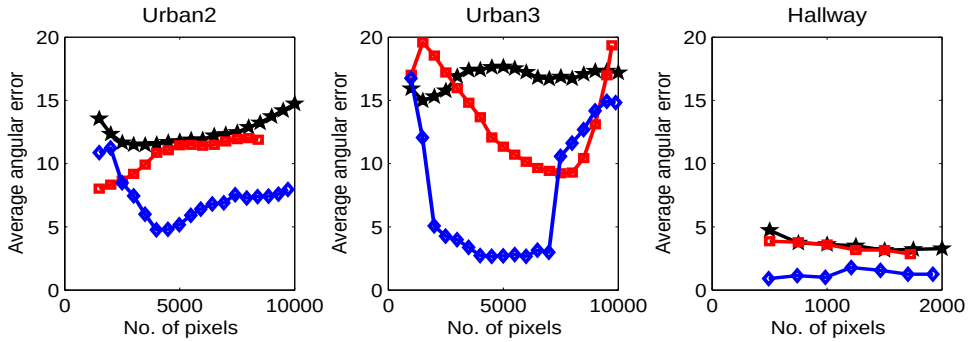
(a) Average endpoint error: Hidden texture



(b) Average endpoint error: Synthetic sequences



(c) Average angular error: Hidden texture



(d) Average angular error: Synthetic sequences

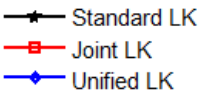
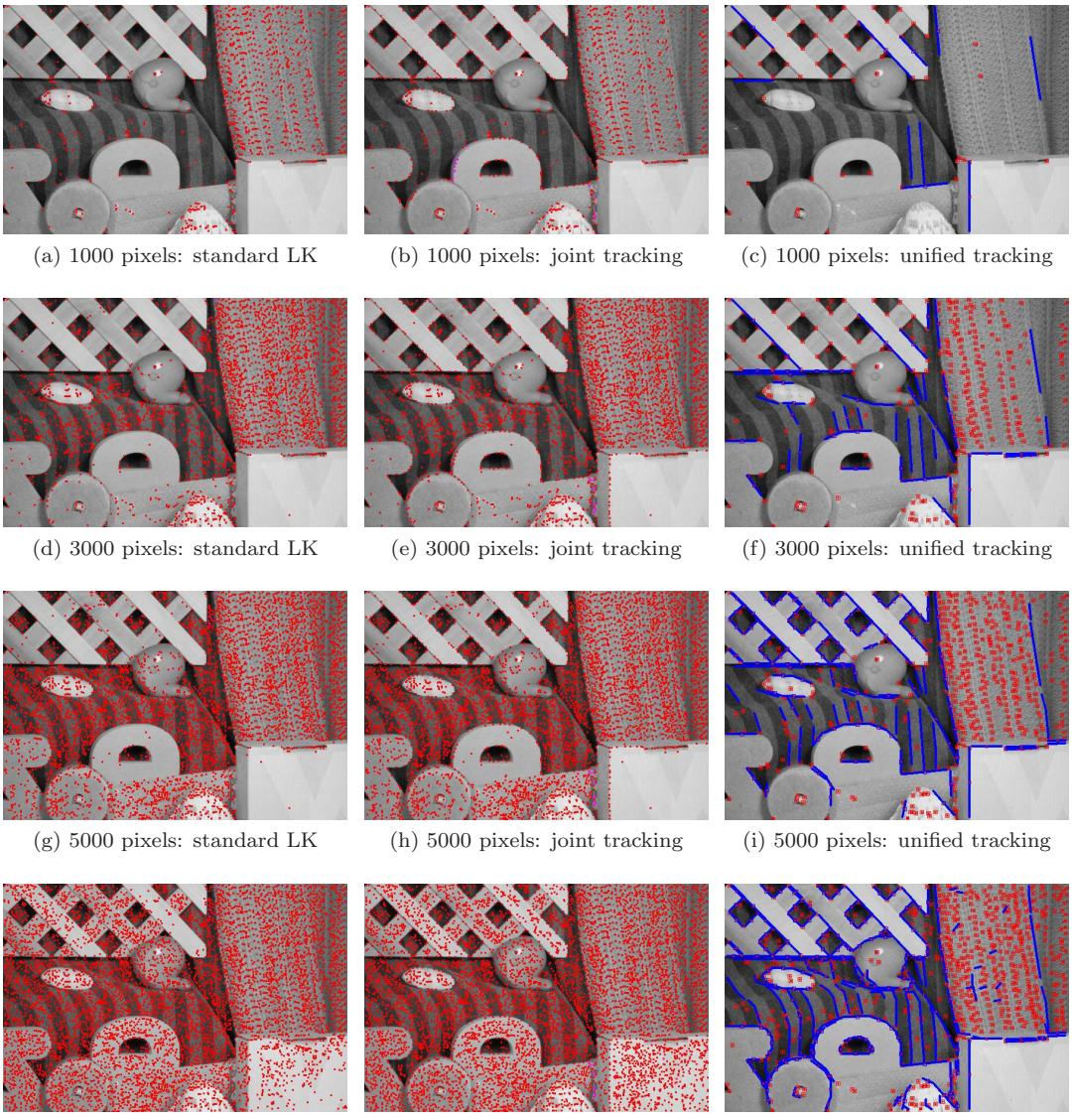


Figure 4.5: Average endpoint and angular error for dense optical flow.



(a) 1000 pixels: standard LK (b) 1000 pixels: joint tracking (c) 1000 pixels: unified tracking

(d) 3000 pixels: standard LK (e) 3000 pixels: joint tracking (f) 3000 pixels: unified tracking

(g) 5000 pixels: standard LK (h) 5000 pixels: joint tracking (i) 5000 pixels: unified tracking

(j) 7000 pixels: standard LK (k) 7000 pixels: joint tracking (l) 7000 pixels: unified tracking

Figure 4.6: Notice that the unified feature tracking method provides a meaningful representation of the scene as compared to the other two methods for dense feature representation.

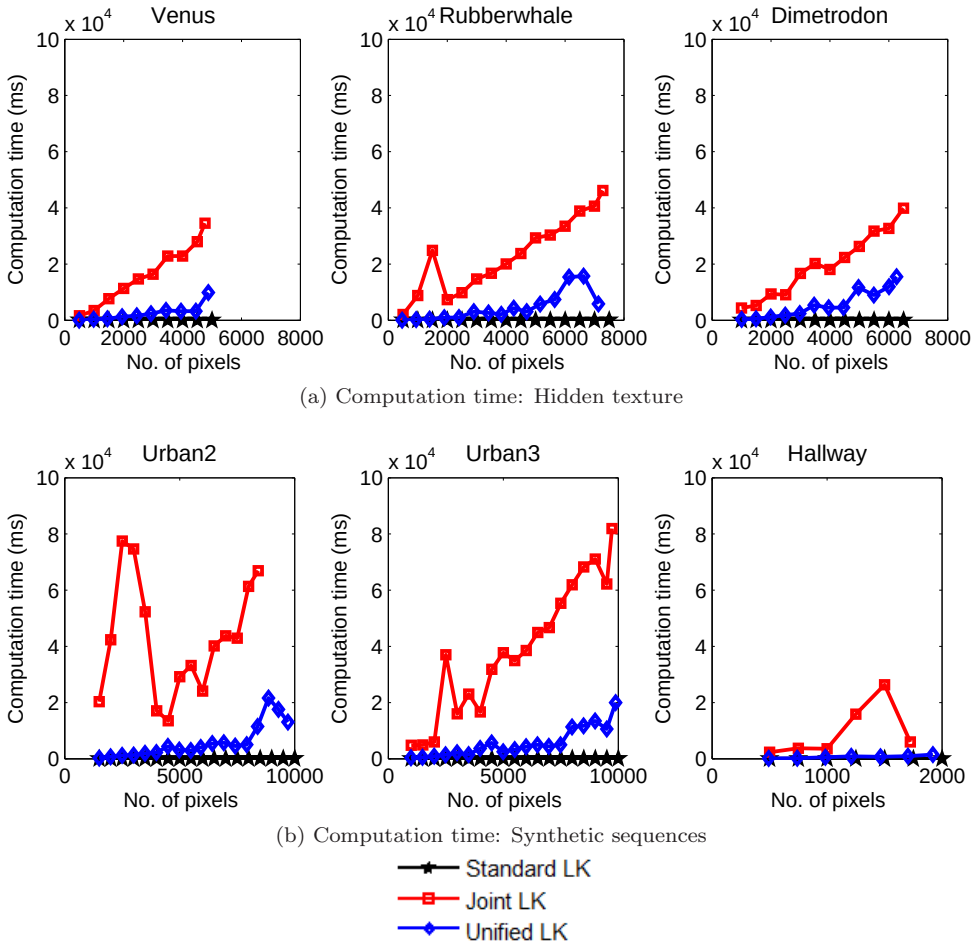


Figure 4.7: Computation time for different feature tracking methods.

feature tracking methods.

4.3 Computation time

The computation time of the standard Lucas-Kanade method, joint tracking method and unified feature tracking method was obtained for features representing different number of pixels. Features represent only point features for Lucas-Kanade method and joint tracking method. Both point features and edgelets are taken into account for unified feature tracking method. Figures 4.7a and 4.7b show the computation time for all the three feature tracking methods.

It can be observed from Figures 4.7a and 4.7b that the Lucas-Kanade method takes only 100ms on an average even for a large number of features. However, the joint tracking method and

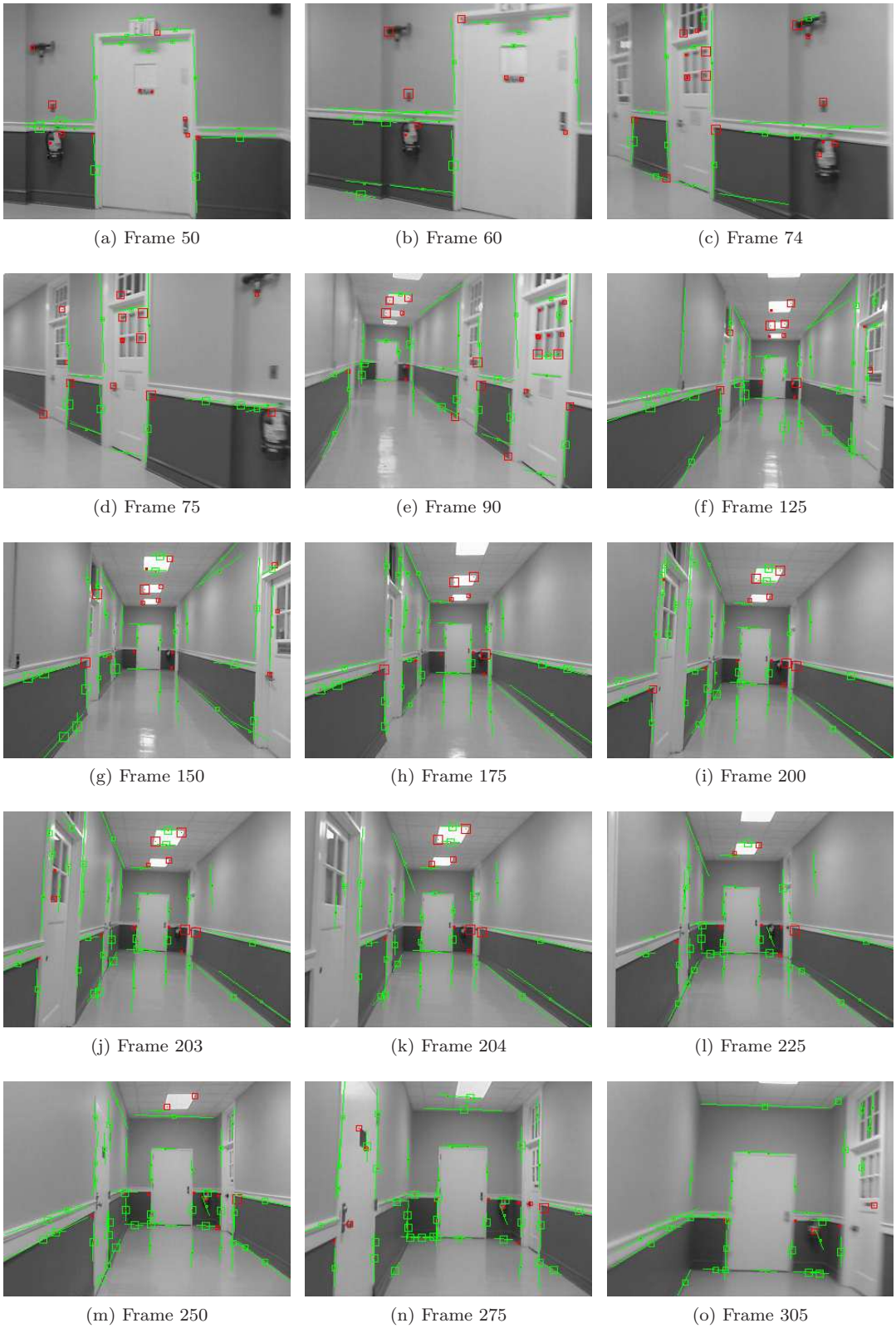
unified feature tracking method take atleast few hundreds of milliseconds and the computation time increases to few seconds as the number of features increase. This is due to the fact that both these methods rely on the motion vector computation of the neighboring features whereas the Lucas-Kanade method tracks features independently. It should also be noted that even though the unified feature tracking method tracks both point features and edgelets, its computation time is much lesser than that of the joint tracking method. This is due to the efficient feature representation in the unified feature tracking method. In the unified feature tracking method, all the edgels in a edgelet are tracked as a single feature. In the joint tracking method, each edgel would be tracked as a distinct point feature. Hence the unified feature tracking method, besides providing meaningful description of an image, can also be used for nearly real-time tracking.

4.4 Indoor image sequences

4.4.1 Hallway sequence

The unified feature tracking algorithm was tested using an image sequence of hallway taken using a handheld camera. The image sequence consisted of 300 frames. These images represent typical indoor image sequences with very less texture and relatively more edgelets. The feature tracking results for few frames have been shown in Figures 4.8a to 4.8o. This image sequence has various scale changes and large motion between successive frames. Hence, we use up to five pyramidal levels to handle large motion and maximum of 20 iterations. Since this image sequence has very few features, we choose any feature within a radius of 70 pixels as a neighboring feature. This ensures that there are sufficient features to provide flow vector regularization. Utilization of feature scale in this method has shown very reliable results. The scale of the tracked features was also updated in every frame.

It can be observed from Figures 4.8c and 4.8d and Figures 4.8j and 4.8k that these subsequent frames show large displacment. However, our feature tracking method has been able to track features on these frames reliably. It can also be observed that we use a translational model for our feature tracking method and hence only the change in displacement of the edgelets is computed. Any change in orientation or length of the edgelets is not computed. Hence, when image frames have large viewpoint changes over a few frames, the edgelets tracked using this method will not align exactly with the image edges.



(a) Frame 50 (b) Frame 60 (c) Frame 74

(d) Frame 75 (e) Frame 90 (f) Frame 125

(g) Frame 150 (h) Frame 175 (i) Frame 200

(j) Frame 203 (k) Frame 204 (l) Frame 225

(m) Frame 250 (n) Frame 275 (o) Frame 305

Figure 4.8: Unified point-edgelet feature tracking on Hallway sequence.

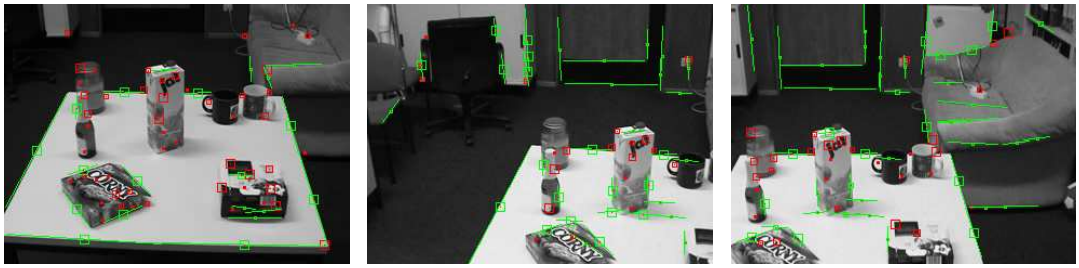
4.4.2 BoBoT sequence - C

The unified feature tracking algorithm was tested using sequence C of Bonn Benchmark on Tracking (BoBoT). This sequence consists of 404 frames of a room obtained using a moving camera. This sequence represents another man-made environment with relatively more edgelets and texture due to presence of multiple objects. This sequence has large direction and scale changes and our feature tracking method has provided reliable results on this sequence. In order to handle large direction changes, we use up to four pyramidal levels and maximum of 20 iterations. Since this sequence has reasonably more point features and edgelets, any feature within a radius of 30 pixels is chosen as a neighboring feature. The feature tracking results have been shown in Figures 4.9a to 4.9o.

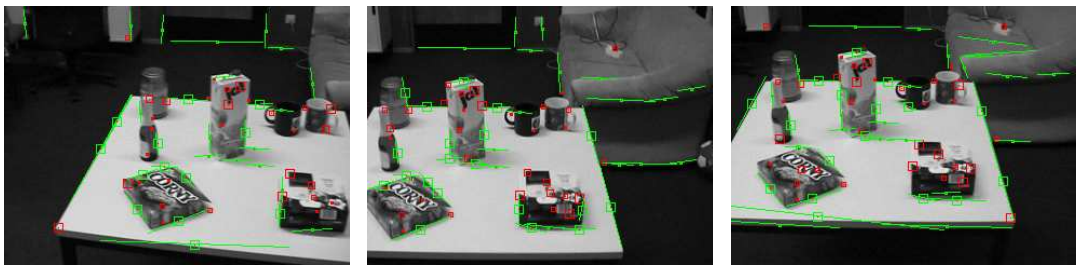
4.5 Effect of scale in tracking

As described in Sections 3.3.1.3 and 3.3.2.1, the scale at which the point features and edgelets can be detected is determined. The features are tracked only from the pyramidal level at which they can be detected. At lower pyramidal levels at which they cannot be detected, they follow the motion of the neighboring features. This ensures that the features do not give erroneous motion vectors at lower pyramidal levels at which they cannot be detected. This proves really very effective when there is large displacement between frames and multiple pyramidal levels are used.

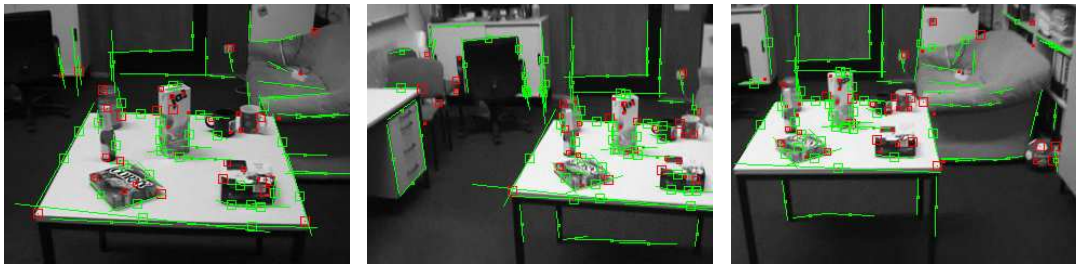
Figures 4.10a and 4.10b shows two images from the hallway sequence where the point features have been tracked without taking into account the scale at which these features can be detected. Figures 4.10c and 4.10d shows the same images where the features have been tracked giving respect to the scale at which they can be detected. The number of pyramidal levels, iterations and other parameters used in both the cases were the same. It can be observed from these images that some edgelets which represent the door frame stray away when tracked at all pyramidal levels. This is due to the fact that these edges have weak gradients and hence vanish at lower pyramidal levels. Hence, the motion vectors computed at lower pyramidal levels are erroneous and serve as wrong intialization vectors for upper pyramidal levels. When the features are tracked from the pyramidal levels at which they can be detected, they provide more reliable flow vectors. Figures 4.10c and 4.10d show that these edgelets which represent the door frame are visible only at the uppermost pyramidal level. Hence, these edgelets follow the neighboring features' motion vectors at the lower



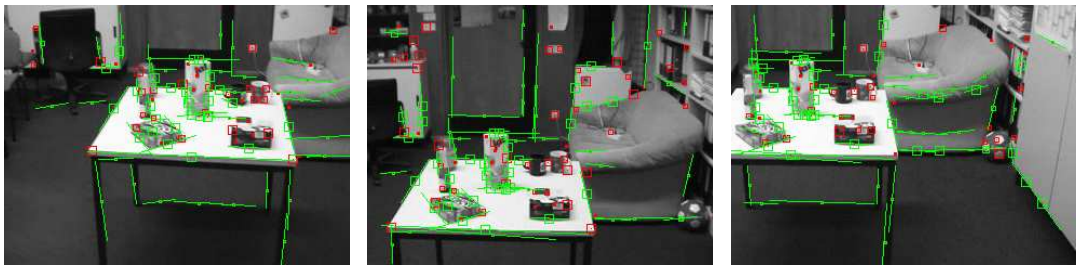
(a) Frame 2 (b) Frame 34 (c) Frame 55



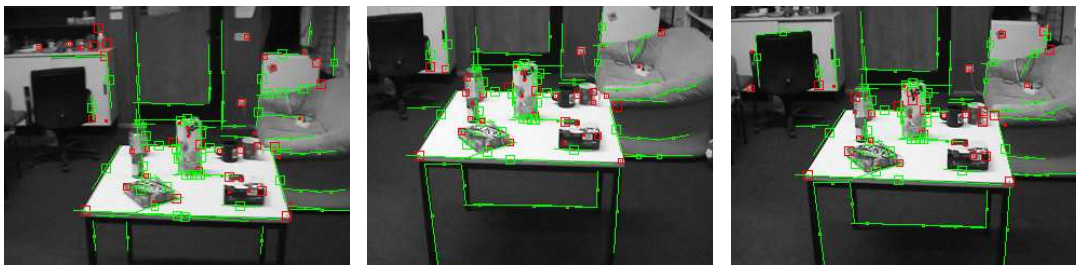
(d) Frame 75 (e) Frame 100 (f) Frame 150



(g) Frame 172 (h) Frame 195 (i) Frame 227



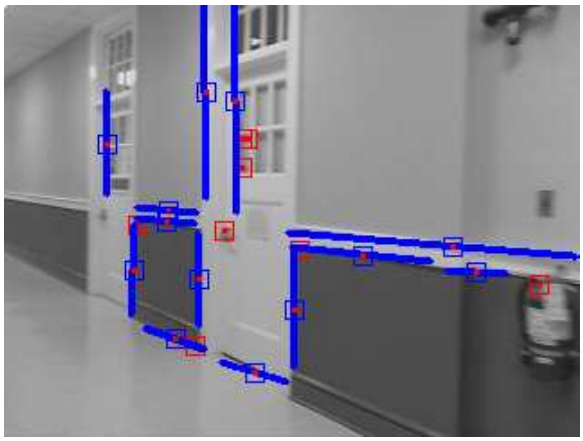
(j) Frame 242 (k) Frame 275 (l) Frame 300



(m) Frame 350 (n) Frame 396 (o) Frame 404

Figure 4.9: Unified point-edgelet feature tracking on BoBoT sequence C.

(a) Frame 74 - Features detected with no scale information



(b) Frame 75 - Features tracked at all pyramidal levels

(c) Frame 74 - Features detected in scale space

(d) Frame 75 - Features tracked in scale space

Figure 4.10: Note that the features tracked in scale space are able to handle features due to weak gradients (edgelets on the door frame) better than their counterpart which tracks all features at all pyramidal levels.

pyramidal levels. These serve as correct initialization vectors for the uppermost pyramidal level and hence are tracked reliably.

Chapter 5

Conclusions and Discussion

In this thesis, we have proposed a unified feature tracking algorithm to track point features and edgelets simultaneously. We have shown that the combination of point features and edgelets provide a better representation of image content. We have provided the results of the ground truth comparison for the Middlebury optical flow datasets. We have also compared the results with that of existing feature tracking algorithms like the Lucas-Kanade method and the joint feature tracking method.

It has been shown that the unified feature tracking method uses fewer features to provide meaningful representation of the image as compared to the other two methods. Moreover, the results of the unified feature tracking method is better than that of the standard Lucas-Kanade method or Joint tracking method for images of man-made environment. Also, the computation time of the unified feature tracking algorithm is in order of hundred milliseconds for few hundred features and is comparable to the standard Lucas-Kanade method. This would work reasonably well for the real-world scenario where the image content can be meaningfully represented using few point features and edgelets. Hence, this method will prove to be invaluable for nearly real-time feature tracking applications besides providing meaningful representation of the image content.

The unified point-edgelet feature tracking algorithm has been tested on few indoor image sequences with poor texture. It has been demonstrated that the unified feature tracking algorithm efficiently tracks the point features and edgelets over few hundreds of frames. The use of scale space for feature tracking aids in reliable feature tracking even in cases of large displacement.

There is much room for improvement in this algorithm. This algorithm currently assumes

a translational motion model for the features. Hence, only the displacement of the point features and edgelets are computed. However, if the image undergoes affine transformations over frames, the length and orientation of the edgelets will undergo change. The current algorithm does not handle changes in length and orientation of the edgelets but can be extended to incorporate these changes.

The edgelets may sometimes represent regions of motion discontinuities. If this property could be utilized for intelligent regularization of features, it would yield a robust feature tracking algorithm and accurate motion segmentation. Also, the unified feature tracking algorithm currently tracks only point features and edgelets and can be easily extended to include other parametric curves.

Appendices

Appendix A Derivation: Unified point-edgelet tracking

This section describes the energy formulation for unified point-edgelet feature tracking and the derivation for computing the motion vector components. The point features are tracked by minimizing an energy function whose data term represents the optical flow constraint. The edgelets are tracked by minimizing an energy functional consisting of two data terms: the optical flow constraint, and the sum of inverse gradient magnitude of edgels. A smoothness term is added to the energy function of both point features and edgelets to avoid large flow vector deviations from the neighboring features.

Let N_P be the number of point features and N_E be the number of edgelets. We need to compute the displacements $U_P = \langle (u_{Pi}, v_{Pi}) \rangle_{i=1}^{N_P}$ of the point features and the displacements $U_E = \langle (u_{Ej}, v_{Ej}) \rangle_{j=1}^{N_E}$ of the edgelets. The displacements are measured with respect to the previous image frame I and the current frame J, where $I(\mathbf{x})$ and $J(\mathbf{x})$ are the intensities of the pixel $\mathbf{x} = (x, y)$ in the two frames, respectively. The displacements of the point features and edgelets are determined by minimizing the following energy functional:

$$E_{UPE}(U_P, U_E; \mathcal{X}, \mathcal{E}, I, J) = \sum_{i=1}^{N_P} (E_{DP}(i) + E_{SP}(i)) + \sum_{j=1}^{N_E} (E_{DE}(j) + E_{SE}(j)). \tag{1}$$

The point features are tracked using the optical flow constraint, while edgelets are tracked using both the optical flow constraint and the gradient magnitude, as seen in the two data terms:

$$E_{DP}(i) = \sum_{\mathbf{x} \in \mathcal{W}_P(i)} (f(\mathbf{x}, u_{Pi}, v_{Pi}; I, J))^2 \tag{2}$$

$$E_{DE}(j) = \sum_{\mathbf{x} \in \mathcal{W}_E(j)} (f(\mathbf{x}, u_{Ej}, v_{Ej}; I, J))^2 + (g(\mathbf{x}, u_{Ej}, v_{Ej}; J))^2, \tag{3}$$

where (u_{Pi}, v_{Pi}) is the displacement of the ith point feature, (u_{Ej}, v_{Ej}) is the displacement of the jth edgelet, $\mathcal{W}_P(i)$ is an integration window of size $m \times m$ around the ith point feature, and $\mathcal{W}_E(j)$ is an integration window of size $m \times \ell_j$ along the jth edgelet, ℓ_j is the length of the edgelet. The functions f and g are the optical flow constraint equation and the inverse gradient magnitude, respectively:

$$f(\mathbf{x}, u, v; I, J) = I_x(\mathbf{x})u + I_y(\mathbf{x})v + I_t(\mathbf{x}) \tag{4}$$

$$g(\mathbf{x}, u, v; J) = G(x + u, y + v), \tag{5}$$

where $G(x,y) = m(J) - \|\nabla J(x,y)\|$, and $m(J) = \max_{\mathbf{x}} \|\nabla J(\mathbf{x})\|$ is the maximum gradient magnitude over the whole image used to ensure that $G(x,y) \geq 0$. The spatial derivatives are $I_x(\mathbf{x}) = \partial I(\mathbf{x})/\partial x$ and $I_y(\mathbf{x}) = \partial I(\mathbf{x})/\partial y$, and the temporal derivative is $I_t(\mathbf{x}) = \partial I(\mathbf{x})/\partial t \approx J(\mathbf{x}) - I(\mathbf{x})$.

Using Taylor series expansion on (5),

$$g(\mathbf{x}, u, v; J) \;=\; G(x+u, y+v) \;=\; G(\mathbf{x}) + G_x(\mathbf{x})u + G_y(\mathbf{x})v, \tag{6}$$

where $G_x(\mathbf{x}) = \partial G(\mathbf{x})/\partial x$ and $G_y(\mathbf{x}) = \partial G(\mathbf{x})/\partial y$ are the spatial derivatives of $G(\mathbf{x})$.

The smoothness terms for point features and edgelets are

$$E_{SP}(i) \;=\; \lambda_{Pui}(u_{Pi} - \hat{u}_{Pi})^2 + \lambda_{Pvi}(v_{Pi} - \hat{v}_{Pi})^2 \tag{7}$$

$$E_{SE}(j) \;=\; \lambda_{Euj}(u_{Ej} - \hat{u}_{Ej})^2 + \lambda_{Evj}(v_{Ej} - \hat{v}_{Ej})^2, \tag{8}$$

where $(\hat{u}_{Pi}, \hat{v}_{Pi})$ is the expected displacement of the ith point feature based on the displacements of its neighbors, and similarly for $(\hat{u}_{Ej}, \hat{v}_{Ej})$.

A.1 Displacements of point features

The displacements of the point features are determined by minimizing the energy functional,

$$\begin{aligned} E_{UPE}(U_P; \mathcal{X}, I, J) \;=\; & \sum_{\mathbf{x} \in \mathcal{W}_P(i)} \left(I_x(\mathbf{x})u_{Pi} + I_y(\mathbf{x})v_{Pi} + I_t(\mathbf{x})\right)^2 \\ & + \lambda_{Pui}(u_{Pi} - \hat{u}_{Pi})^2 + \lambda_{Pvi}(v_{Pi} - \hat{v}_{Pi})^2 \end{aligned} \tag{9}$$

Differentiating (9) with respect to the unknown displacements and equating to zero, we obtain

$$\frac{\partial E_{UPE}}{\partial u_{Pi}} \;=\; \sum_{\mathbf{x} \in \mathcal{W}_P(i)} \left(I_x(\mathbf{x})^2 u_{Pi} + I_x(\mathbf{x})I_y(\mathbf{x})v_{Pi} + I_x(\mathbf{x})I_t(\mathbf{x})\right) + \lambda_{Pui}(u_{Pi} - \hat{u}_{Pi}) = 0, \tag{10}$$

$$\frac{\partial E_{UPE}}{\partial v_{Pi}} \;=\; \sum_{\mathbf{x} \in \mathcal{W}_P(i)} \left(I_x(\mathbf{x})I_y(\mathbf{x})u_{Pi} + I_y(\mathbf{x})^2 v_{Pi} + I_y(\mathbf{x})I_t(\mathbf{x})\right) + \lambda_{Pvi}(v_{Pi} - \hat{v}_{Pi}) = 0. \tag{11}$$

Equations (10) and (11) for $\{\mathcal{X}\}_{i=1}^{N_P}$ yields $2N_P$ equations which are stacked in a $2N \times 2N$ sparse

matrix equation, where $N = N_P + N_E$. By convention, we stack the point features above the edgelets in the equation, so that the $(2i-1)$th and $(2i)$th rows for point features are $Z_{Pi}\mathbf{w}_{Pi} = \mathbf{r}_{Pi}$, where

$$Z_{Pi} = \begin{bmatrix} \lambda_{Pui} + \sum_{\mathbf{x}\in\mathcal{W}_P(i)} I_x^2(\mathbf{x}) & \sum_{\mathbf{x}\in\mathcal{W}_P(i)} I_x(\mathbf{x})I_y(\mathbf{x}) \\ \sum_{\mathbf{x}\in\mathcal{W}_P(i)} I_x(\mathbf{x})I_y(\mathbf{x}) & \lambda_{Pvi} + \sum_{\mathbf{x}\in\mathcal{W}_P(i)} I_y^2(\mathbf{x}) \end{bmatrix} \tag{12}$$

$$\mathbf{w}_{Pi} = \begin{bmatrix} u_{Pi} \\ v_{Pi} \end{bmatrix} \qquad \mathbf{r}_{Pi} = \begin{bmatrix} \lambda_{Pui}\hat{u}_{Pi} - \sum_{\mathbf{x}\in\mathcal{W}_P(i)} I_x(\mathbf{x})I_t(\mathbf{x}) \\ \lambda_{Pvi}\hat{v}_{Pi} - \sum_{\mathbf{x}\in\mathcal{W}_P(i)} I_y(\mathbf{x})I_t(\mathbf{x}) \end{bmatrix}. \tag{13}$$

A.2 Displacement of edgelets

The displacements of edgelets are determined by minimizing the energy functional,

$$\begin{aligned} E_{UPE}(U_E;\mathcal{E},I,J) \;=\; & \sum_{\mathbf{x}\in\mathcal{W}_E(j)} \left(I_x(\mathbf{x})u_{Ej} + I_y(\mathbf{x})v_{Ej} + I_t(\mathbf{x})\right)^2 \\ & + \left(G_x(\mathbf{x})u_{Ej} + G_y(\mathbf{x})v_{Ej} + G(\mathbf{x})\right)^2 \\ & + \lambda_{Euj}(u_{Ej} - \hat{u}_{Ej})^2 + \lambda_{Evj}(v_{Ej} - \hat{v}_{Ej})^2 \end{aligned} \tag{14}$$

Differentiating (14) with respect to the unknown displacements and equating to zero, we obtain

$$\begin{aligned} \frac{\partial E_{UPE}}{\partial u_{Ej}} \;=\; & \sum_{\mathbf{x}\in\mathcal{W}_E(j)} \left(I_x(\mathbf{x})^2 u_{Ej} + I_x(\mathbf{x})I_y(\mathbf{x})v_{Ej} + I_x(\mathbf{x})I_t(\mathbf{x})\right) \\ & + \left(G_x(\mathbf{x})^2 u_{Ej} + G_x(\mathbf{x})G_y(\mathbf{x})v_{Ej} + G_x(\mathbf{x})G(\mathbf{x})\right) \\ & + \lambda_{Euj}(u_{Ej} - \hat{u}_{Ej}) = 0, \end{aligned} \tag{15}$$

$$\begin{aligned} \frac{\partial E_{UPE}}{\partial v_{Ej}} \;=\; & \sum_{\mathbf{x}\in\mathcal{W}_E(j)} \left(I_x(\mathbf{x})I_y(\mathbf{x})u_{Ej} + I_y(\mathbf{x})^2 v_{Ej} + I_y(\mathbf{x})I_t(\mathbf{x})\right) \\ & + \left(G_x(\mathbf{x})G_y(\mathbf{x})u_{Ej} + G_y(\mathbf{x})^2 v_{Ej} + G_y(\mathbf{x})G(\mathbf{x})\right) \\ & + \lambda_{Evj}(v_{Ej} - \hat{v}_{Ej}) = 0. \end{aligned} \tag{16}$$

Equations (15) and (16) for $\{\mathcal{E}\}_{i=1}^{N_E}$ yields $2N_E$ equations which are stacked below point features in

a $2N \times 2N$ sparse matrix equation, where $N = N_P + N_E$. For edgelets the $(2N_P + 2j - 1)$th and $(2N_P + 2j)$th rows are $Z_{Ej}\mathbf{w}_{Ej} = \mathbf{r}_{Ej}$, where

$$Z_{Ej} = \begin{bmatrix} \lambda_{Euj} + \sum_{\mathbf{x}\in\mathcal{W}_E(j)} (I_x^2(\mathbf{x}) + G_x^2(\mathbf{x})) & \sum_{\mathbf{x}\in\mathcal{W}_E(j)} (I_x(\mathbf{x})I_y(\mathbf{x}) + G_x(\mathbf{x})G_y(\mathbf{x})) \\ \sum_{\mathbf{x}\in\mathcal{W}_E(j)} (I_x(\mathbf{x})I_y(\mathbf{x}) + G_x(\mathbf{x})G_y(\mathbf{x})) & \lambda_{Evj} + \sum_{\mathbf{x}\in\mathcal{W}_E(j)} (I_y^2(\mathbf{x}) + G_y^2(\mathbf{x})) \end{bmatrix} \tag{17}$$

$$\mathbf{w}_{Ej} = \begin{bmatrix} u_{Ej} \\ v_{Ej} \end{bmatrix} \qquad \mathbf{r}_{Ej} = \begin{bmatrix} \lambda_{Eui}\hat{u}_{Ej} - \sum_{\mathbf{x}\in\mathcal{W}_E(j)} (I_x(\mathbf{x})I_t(\mathbf{x}) + G_x(\mathbf{x})G(\mathbf{x})) \\ \lambda_{Evi}\hat{v}_{Ej} - \sum_{\mathbf{x}\in\mathcal{W}_E(j)} (I_y(\mathbf{x})I_t(\mathbf{x}) + G_y(\mathbf{x})G(\mathbf{x})) \end{bmatrix}. \tag{18}$$

In these expressions $\lambda_{Pui}, \lambda_{Pvi}, \lambda_{Euj}, \lambda_{Evj} > 0$ are regularization parameters governing the amount of smoothing.

Bibliography

[1] S. Baker and I. Matthews. Lucas-Kanade 20 years on: A unifying framework. *International Journal of Computer Vision*, 56(3):221–255, 2004.

[2] S. Baker, D. Scharstein, J.P. Lewis, S. Roth, M.J. Black, and R. Szeliski. A database and evaluation methodology for optical flow. In *Proceedings of the IEEE International Conference on Computer Vision*, 2007.

[3] A. Bartoli and P. Sturm. Structure-from-motion using lines: Representation, triangulation and bundle adjustment. *Computer Vision and Image Understanding*, 100(3):416–441, 2005.

[4] S.T. Birchfield and S.J. Pundlik. Joint tracking of features and edges. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[5] A. Bruhn, J. Weickert, and C. Schnorr. Lucas-Kanade meets Horn-Schunck: Combining local and global optic flow methods. *International Journal of Computer Vision*, 61(3):211–231, 2005.

[6] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.

[7] T.F. Chan and L.A. Vese. Active contours without edges. *IEEE Transactions on Image Processing*, 10(2):266–277, 2001.

[8] N. Chiba and Takeo Kanade. A tracker for broken and closely spaced lines. In *Proceedings of the 1998 International Society for Photogrammetry and Remote Sensing Conference (ISPRS '98)*, pages 676–683, 1998.

[9] J. Crowley, P. Sterlmaszyk, T. Skordas, and P. Puget. Measurement and integration of 3-D structure by tracking edge lines. *International Journal of Computer Vision*, 8(1):29–52, July 1992.

[10] David Douglas and Thomas Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *The Canadian Cartographer*, 10(2):112–122, 1973.

[11] E. Eade and T. Drummond. Edge landmarks in monocular SLAM. In *Proceedings of the British Machine Vision Conference*, 2006.

[12] C. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of the 4th Alvey Vision Conference*, 1988.

[13] B.K.P. Horn and B.G. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[14] Michael Isard and Andrew Blake. CONDENSATION – Conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998.

[15] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1988.

[16] C. Liu, W.T. Freeman, and E.H. Adelson. Analysis of contour motions. In *NIPS*, 2006.

[17] 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*, pages 674–679, 1981.

[18] R. Malladi and J.A. Sethian. Level set and fast marching methods in image processing and computer vision. In *Proceedings of International Conference on Image Processing*, 1996.

[19] N. Paragios and R. Deriche. Geodesic active regions and level set methods for motion estimation and tracking. *International Journal of Computer Vision*, 97(3):259–282, 2005.

[20] Gerhard Reitmayr and Tom W. Drummond. Going out: Robust model-based tracking for outdoor augmented reality. In *Proceedings of the IEEE and ACM International Symposium on Mixed and Augmented Reality*, 2006.

[21] E. Rosten and T. Drummond. Fusing points and lines for high performance tracking. In *International Conference on Computer Vision*, pages 1508–1515, October 2005.

[22] C. Schmid and A. Zisserman. Automatic line matching across views. In *IEEE Transactions on Computer Vision and Pattern Recognition*, pages 666–671, June 1997.

[23] Jianbo Shi and Carlo Tomasi. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994.

[24] P. Smith, I. Reid, and A. Davison. Real-time monocular SLAM with straight lines. In *Proceedings of the British Machine Vision Conference*, volume 1, pages 17–26, 2006.

[25] C. Tomasi and T. Kanade. Detection and tracking of point features. *CMU Technical report*, 1991.

[26] Zhiheng Wang, Fuchao Wu, and Zhanyi Hu. MSLD: A robust descriptor for line matching. *Pattern Recognition*, 42(5):941–953, 2009.

[27] J. Weng, T. S. Huang, and N. Ahuja. Motion and structure from line correspondences: Closed form solution, uniqueness and optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(3):318–336, March 1992.

[28] Harald Wuest, Florent Vial, and Didier Stricker. Adaptive line tracking with multiple hypotheses for augmented reality. In *Proceedings of the IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 62–69, October 2005.