Beyond Pixels: Exploring New Representations and Applications for Motion Analysis

by

Ce Liu

Submitted to the Department of Electrical Engineering and Computer Science in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Electrical Engineering and Computer Science
at the Massachusetts Institute of Technology

June 2009

© 2009 Massachusetts Institute of Technology
All Rights Reserved.

Signature of Author: _____

Department of Electrical Engineering and Computer Science
May 21, 2009

Certified by: _____

William T. Freeman, Professor of Computer Science
Thesis Supervisor

Certified by: _____

Edward H. Adelson, John and Dorothy Wilson Professor of Vision Science
Thesis Supervisor

Accepted by: _____

Terry P. Orlando, Professor of Electrical Engineering
Chair, Department Committee on Graduate Students

To my parents

Beyond Pixels: Exploring New Representations and Applications for Motion Analysis

by

Ce Liu

Submitted to the Department of Electrical Engineering
and Computer Science on May 21, 2009
in Partial Fulfillment of the Requirements for the Degree
of Doctor of Philosophy in Electrical Engineering and Computer Science

Abstract

The focus of motion analysis has been on estimating a flow vector for every pixel by matching intensities. In my thesis, I will explore motion representations beyond the pixel level and new applications to which these representations lead.

I first focus on analyzing motion from video sequences. Traditional motion analysis suffers from the inappropriate modeling of the grouping relationship of pixels and from a lack of ground-truth data. Using layers as the interface for humans to interact with videos, we build a human-assisted motion annotation system to obtain ground-truth motion, missing in the literature, for natural video sequences. Furthermore, we show that with the layer representation, we can detect and magnify small motions to make them visible to human eyes. Then we move to a contour presentation to analyze the motion for textureless objects under occlusion. We demonstrate that simultaneous boundary grouping and motion analysis can solve challenging data, where the traditional pixel-wise motion analysis fails.

In the second part of my thesis, I will show the benefits of matching local image structures instead of intensity values. We propose SIFT flow that establishes dense, semantically meaningful correspondence between two images across scenes by matching pixel-wise SIFT features. Using SIFT flow, we develop a new framework for image parsing by transferring the metadata information, such as annotation, motion and depth, from the images in a large database to an unknown query image. We demonstrate this framework using new applications such as predicting motion from a single image and motion synthesis via object transfer. Based on SIFT flow, we introduce a nonparametric scene parsing system using label transfer, with very promising experimental results suggesting that our system outperforms state-of-the-art techniques based

on training classifiers.

Thesis Supervisor: William T. Freeman
Title: Professor of Computer Science

Thesis Supervisor: Edward H. Adelson
Title: John and Dorothy Wilson Professor of Vision Science

Acknowledgments

This thesis is not possible without the support, guidance and love from so many people around me.

I want to especially thank my thesis advisors, Prof. William T. Freeman and Prof. Edward H. Adelson. In his patience and wisdom, Bill has spent great effort to guide me during my doctoral thesis, from mathematical equations to how to give professional presentations. His words of encouragements and guidance have helped me through many obstacles during my thesis. I have learned from him not only how to conduct high-quality research, but also how to be a man of humility, integrity, kindness and patience. With his broad knowledge in science and engineering, Ted has shown me his genuine motive and profound wisdom for pursuing scientific truth. I deeply appreciate the words he shared with me when our ICCV submission got rejected: "We are to discover the truth of God's creation. It is the process of pursuing the truth that we ought to enjoy, not the recognition that comes with the discovery." I am very thankful to both Bill and Ted for their generous support.

My deep gratitude also goes to my other thesis committee members, Prof. Antonio Torralba and Dr. Richard Szeliski. Having been working on several projects together, Antonio is a great mentor and friend to me personally. I am always encouraged by his attitude towards research, especially when we were frustrated. One of his famous sayings "bugs are good" (because bugs may indicate that the algorithm we designed still has hope) motivates me to debug even late at night. It was a very memorable summer when I interned at Microsoft Research in Redmond with Dr. Richard Szeliski. Already exceptionally knowledgeable in computer vision and computer graphics, Rick is till very passionate about new technologies. He also gave me many insightful comments on my thesis.

I want to thank other professors and researchers with whom I worked together during my thesis: Dr. Harry Shum, Prof. Frédo Durand, Prof. Yair Weiss, Prof. Marshall Tappen, Prof. Rob Fergus, Dr. Sing Bing Kang, Dr. Larry Zitnick, Dr. Josef Sivic, Dr. Bryan Russell, Jenny Yuen and Dr. Sylvain Paris. Frédo has generously shared his knowledge about photography as well as his equipments to allow me to experience the fantastic world of digital photography. Yair taught me a very unique perspective towards research. Once he decided not to make an

ICCV submission because the preliminary results he got was too good to comprehend why. So he would rather believe that there was something he still did not understand, than thinking he made some great discoveries. Marshall has been a very good friend to me since we were labmates at MIT. I really cherish the time we spent together in Cambridge, UK.

I feel very privileged to share office with Bryan Russell, Michael Siracusa, Biliana Kaneva, and Clare Poynton. We have altogether shared so much joy and laughter. I am particularly grateful to Bryan who has helped me generously on things including Linux, MATLAB and English writing since I came to MIT. We have spent great time together at conferences in NYC, Beijing, Minneapolis, Pasadena, Marseille, and Paris.

I want to thank Prof. Randall Davis, my academic advisor, and other MIT faculty, Prof. Victor Zue, Prof. Eric Grimson, Prof. Gilbert Strang, Prof. Polina Golland, and Prof. Seth Teller. I am also thankful to my labmates (including previous members) in vision and graphics at CSAIL: Xiaogang Wang, Xiaoxu Ma, Jiawen Chen, Robert Wang, Wanmei Ou, Thomas Yeo, Erik Sudderth, Gerald Dalley, Biswajit Bose, Mario Christoudias, Peter Sand, Soonmin Bae, Tilke Judd, Sara Su, Taeg Sang Cho, Anat Levin, Hyun Sung Chang, Kevin Murphy, Barun Singh, Leo Zhu, Sam Hasinoff, Roger Grosse, Kimo Johnson, Lavanya Sharan, Yuanzhen Li and Alvin Raj. In addition, Microsoft Research and Xerox Incorporation have kindly granted me fellowships to support my graduate study.

This thesis is impossible without my parents' love and sacrifice. My gratitude to them is beyond what I can express in words. Lastly, I want to thank my fiancée, Irene Liaw, for her love and support.

Contents

Abstract **5**

Acknowledgments **7**

List of Figures **13**

1 Introduction **29**

1.1 Thesis Theme 31

1.2 Thesis Overview 33

1.3 Other Work not in the Thesis 36

2 Human-Assisted Motion Annotation **37**

2.1 Introduction 37

2.2 Related Work 39

2.3 Human-Assisted Motion Annotation System 41

2.3.1 Semiautomatic Layer Segmentation with Occlusion Handling 41

2.3.2 Objective function 42

2.3.3 Linearization and Optimization 42

2.3.4 Multi-channel, multi-resolution image representation 45

2.3.5 Occlusion handling 46

2.3.6 Layer-wise Optical Flow Estimation 48

2.3.7 Semiautomatic Motion Labeling 50

2.3.8 System Design and Human Judgement 52

2.4 Methodology Evaluation 52

2.5 A Human-Annotated Motion Ground-Truth Database 54

2.6 Conclusion 57

3 Layer and Contour Representations for Motion Analysis 59
3.1 Motion Magnification . . . 59
3.1.1 Related Work . . . 60
3.1.2 Overview . . . 61
3.1.3 Robust Video Registration . . . 64
3.1.4 Robust Computation and Clustering of Feature Point Trajectories . . . 65
3.1.5 Segmentation: Assignment of Each Pixel to a Motion Cluster . . . 70
3.1.6 Magnification and Rendering . . . 73
3.1.7 Experimental Results . . . 73
3.1.8 Applications . . . 77
3.1.9 Conclusion . . . 80
3.2 Contour Motion Analysis . . . 81
3.2.1 Boundary Fragment Extraction . . . 83
3.2.2 Edgelet Tracking with Uncertainties . . . 84
3.2.3 Boundary Fragment Grouping and Motion Estimation . . . 85
3.2.4 Experimental Results . . . 89
3.3 Conclusion . . . 90

4 SIFT Flow: Dense Correspondence Across Scenes 93
4.1 Introduction . . . 93
4.2 Related Work . . . 96
4.3 The SIFT Flow Algorithm . . . 98
4.3.1 Dense SIFT descriptors and visualization . . . 98
4.3.2 Matching Objective . . . 100
4.3.3 Coarse-to-fine matching scheme . . . 101
4.3.4 Neighborhood of SIFT flow . . . 104
4.3.5 Scene matching with histogram intersection . . . 104
4.4 Experiments on Video Retrieval . . . 105
4.4.1 Results of video retrieval . . . 105
4.4.2 Evaluation of the dense scene alignment . . . 109
4.5 Applications . . . 109
4.5.1 Predicting motion fields from a single image . . . 111
4.5.2 Quantitative evaluation . . . 111
4.5.3 Motion synthesis via object transfer . . . 115

4.6 Experiments on Image Alignment and Face Recognition 116
4.6.1 Image registration of the same scene 116
4.6.2 Face recognition 120
4.7 Discussions 122
4.8 Conclusion 123

5 Nonparametric Scene Parsing via Dense Scene Alignment 125
5.1 Introduction 125
5.2 Scene Parsing through Label Transfer 127
5.3 Experiments 129
5.4 Conclusion 134

6 Conclusion 139

A Estimating Optical Flow 143
A.1 Two-Frame Optical Flow Computation 143
A.1.1 Problem formulation 143
A.1.2 Iterative Reweighted Least Squares (IRLS) 144
A.1.3 Multi-channel and Lucas-Kanade 146
A.2 Temporal Constraints: Multiple Frames 146
A.2.1 Constant velocity model 147
A.2.2 Second-order data term 148

B The Equivalence between Variational Optimization and IRLS 149
B.1 Introduction 149
B.2 Variational Optimization 150
B.3 Iterative Reweighted Least Square (IRLS) 152
B.4 Variational Inference and IRLS are Identical 152

Bibliography 153

List of Figures

1.1 **Human-assisted motion analysis**. In Chapter 2, we designed a system to allow the user to specify layer configurations and motion hints (b). Our system uses these hints to calculate a dense flow field for each layer. We show that the flow (c) is repeatable and accurate. (d): The output of a representative optical flow algorithm [22], trained on the Yosemite sequence, shows many differences from the labeled ground truth for this and other realistic sequences we have labeled. This indicates the value of our database for training and evaluating optical flow algorithms. 31

1.2 **Motion magnification** is a visual system to magnify small motions in video sequences [67]. See details in Chapter 3.1. 32

1.3 **Analysis of contour motions** [61]. Column (a): boundary fragments are extracted using our boundary tracker. The red dots are the edgelets and the green ones are the boundary fragment ends. Column (b): boundary fragments are grouped into contours and the flow vectors are estimated. Each contour is shown in its own color. Column (c) & (d): the illusory boundaries are generated for the first and second frames. See details in Chapter 3.2. 33

1.4 **SIFT flow for dense scene alignment**. In Chapter 4, we extend the concept of "motion" to the correspondence across scenes. (a) and (b) show images of similar scenes. (b) was obtained by matching (a) to a large image collection. (c) shows image (b) warped to align with (a) using the estimated dense correspondence field. (d) Visualization of pixel displacements using the color coding scheme of [7]. Note the variation in scene appearance between (a) and (b). The visual resemblance of (a) and (c) demonstrates the quality of the scene alignment. 34

1.5 **Nonparametric scene parsing**. For a query image (a), we designed a system to find the top matches (b) (three are shown here) using a coarse-to-fine SIFT flow matching algorithm. The annotations of the top matches (c) are transferred and integrated to parse the input image as shown in (d). For comparison, the ground-truth user annotation of (a) is shown in (e). See Chapter 5 for details. . 35

2.1 We designed a system to allow the user to specify layer configurations and motion hints (b). Our system uses these hints to calculate a dense flow field for each layer. We show that the flow (c) is repeatable and accurate. (d): The output of a representative optical flow algorithm [22], trained on the Yosemite sequence, shows many differences from the labeled ground truth for this and other realistic sequences we have labeled. This indicates the value of our database for training and evaluating optical flow algorithms. 38

2.2 A weighting function, $r_k(q)$, a.k.a. region of support, for each feature point is used in the data term of the contour tracking. The weight function is a product of whether a pixel being inside the contour and a Gaussian modulation. Left: the user specified the contour of a van. Right: the weighting function of each key point. The darker the pixel value, the higher the weight. 43

2.3 Multi-channel and multi-resolution representation for tracking. We use four channels for image presentation, where the first three are RGB (a), and the fourth one is the Laplacian filtering response of the brightness image (b). A Gaussian pyramid is built for initialization and acceleration. Notice that the Laplacian channel has to be obtained from each level of the RGB image, as described in [23]. The Laplacian channel becomes blurred and useless if the pyramid is directly built upon the finest level Laplacian image. 46

2.4 The system allows the user to dynamically change the depth of the object. The user selects the key frames to specify the value of depth, and the system automatically interpolates to the rest of the frames. We use color in HSV space to indicate the depth, by fixing S and V as 255, and letting H (hue) reflect the depth value. Intuitively, warmer color (red) indicates smaller value of depth, closer to the camera, whereas colder color (blue) indicates larger value of depth, further to the camera. 47

2.5 Occlusion reasoning is crucial for the correct tracking. Top row: the depth of the person is specified to be less than that of the car. The tracking was correct even though the car becomes occluded. Bottom row: the depth of the person is specified to be greater than that of the car. The tracking becomes wrong at the occlusion because the appearance of the person was mistakenly used for the appearance of the feature points of the car. 47

2.6 A screenshot of our motion annotation system. From (a) to (e) is the main window, depth controller, magnifier, flow field viewer and control panel. 51

2.7 Clockwise from top left: the image frame, mean labeled motion, mean absolute error (red: high error, white: low error), and error histogram. 51

2.8 For the RubberWhale sequence in [7], we labeled 20 layers in (b) and obtained the annotated motion in (c). The "ground-truth" motion from [7] is shown in (d). The error between (c) and (d) is 3.21° in AAE and 0.104 in AEP, excluding the outliers (black dots) in (d). (e): The color encoding scheme for flow visualization [7]. 53

2.9 The marginal ((a)∼(h)) and joint ((i)∼(n)) statistics of the ground-truth optical flow in our database (log histogram). 54

2.10 Some frames of the ground-truth motion database we created. We obtained ground-truth flow fields that are consistent with object boundaries, as shown in column (3), the horizontal component of flow, and column (4), flow colorization using Figure 2.8 (f). In comparison, the output of an optical flow algorithm [22] is shown in column (5). The error between the ground-truth motion (4) and flow estimation (5) is as follows (AAE, AEP), (a): 8.996°, 0.976; (b): 58.904°, 4.181; (c): 2.573°, 0.456; (d): 5.313°, 0.346; (e) 1.924°, 0.085; (f): 5.689°, 0.196; (g): 5.2431°, 0.3853; and (h): 13.306°, 1.567. Most of the errors are significantly larger than the errors with the Yosemite sequence (AAE 1.723°, AEP0.071) . The parameter of the flow algorithm in column (5) is tuned to generate the best result for each sequence. 55

3.1 Summary of motion magnification processing steps. 61

3.2 Learned regions of support allow features (a) and (b) to reliably track the leaf and background, respectively, despite partial occlusions. For feature (b) on the stationary background, the plots show the x (left) and y (right) coordinates of the track both with (red) and without (blue) a learned region of support for appearance comparisons. The track using a learned region of support is constant, as desired for feature point on the stationary background. 66

3.3 Top images show a feature point on the stationary background layer becoming occluded during frames 6 and 7. Below are the x- and y- coordinates of the tracked feature, showing outlier positions. These can be identified from the inlier probabilities shown as a bar plot (repeated for comparison with each plot) and replaced with smoothed values. 68

3.4 Layered representation for the swing sequence. Only the background and two layers (out of six) are shown. 72

3.5 (a) Outlier locations, not well described by our model. Pixel intensities from these locations are passed through from the registered video into the rendered model (b) to yield the final composite output sequence, (c). The user specifies at which depth layer the outliers belong, in this case, above the background and below the other layers. 72

3.6 The magnification result (right) for a handstand (left). The motion magnified sequence exaggerates the small postural corrections needed to maintain balance. 75

3.7 Under hand pressure, the bookshelf (on aluminium supports) undergoes motions that are made visible when magnified by a factor of 40 using motion magnification. User editing of reference frame masks refined the upper boundary between the books and the shelf. Also, the background layer required manual texture completion as little background is visible during the sequence. 75

3.8 Section of the x, y, t volume from the original sequence (left) and the sequence after motion magnification. The detail shows a vertical section of the beam. The volume illustrates the amplification of the oscillation and also the filling of the texture behind the beam. Notice that after magnification, the motion is almost periodic despite the noise. So, the real motion of the beam is not just one single harmonic but a mixture. The original motion is amplified by a factor of 40. 76

3.15 The local motion vector is estimated for each contour in isolation by selectively comparing orientation energies across frames. (a) A T-junction of the two bar example showing the contour orientation for this motion analysis. (b) The other frame. (c) The relevant orientation energy along the boundary fragment, both for the 2nd frame. A Gaussian pdf is fit to estimate flow, weighted by the oriented energy. (d) Visualization of the Gaussian pdf. The possible contour motions are unaffected by the occluding contour at a different orientation and no spurious motion is detected at this junction. . . 84

3.16 A simple example illustrating switch variables, reversibility and fragment chains. The color arrows show the switch variables. The empty circle indicates end 0 and the filled indicates end 1. (a) Shows three boundary fragments. Theoretically $\mathbf{b}_1^{(0)}$ can connect to any of the other ends including itself, (b). However, the switch variable is *exclusive*, i.e. there is only one connection to $\mathbf{b}_1^{(0)}$, and *reversible*, i.e. if $\mathbf{b}_1^{(0)}$ connects to $\mathbf{b}_3^{(0)}$, then $\mathbf{b}_3^{(0)}$ should also connect to $\mathbf{b}_1^{(0)}$, as shown in (c). Figures (d) and (e) show two of the legal contour groupings for the boundary fragments: two open contours and a closed loop contour. 86

3.17 An illustration of local saliency computation. (a) Without loss of generalization we assume the two ends to be $\mathbf{b}_1^{(0)}$ and $\mathbf{b}_2^{(0)}$. (b) The KL divergence between the distributions of flow vectors are used to measure the motion similarity. (c) An illusory boundary γ is generated by minimizing the energy of the curve. The sum of square curvatures are used to measure the curve smoothness. (d) The means of the local patches located at the two ends are extracted, i.e. h_{11} and h_{12} from $b_1^{(0)}$, h_{21} and h_{22} from $b_2^{(0)}$, to compute contrast consistency. 87

3.18 Input images for the non-synthetic examples of Figure 6. The dancer's right leg is moving downwards and the chair is rotating (note the changing space between the chair's arms). 90

3.19 Experimental results for some synthetic and real examples. The same parameter settings were used for all examples. Column (a): Boundary fragments are extracted using our boundary tracker. The red dots are the edgelets and the green ones are the boundary fragment ends. Column (b): Boundary fragments are grouped into contours and the flow vectors are estimated. Each contour is shown in its own color. Columns (c): the illusory boundaries are generated for the first and second frames. The gap between the fragments belonging to the same contour are linked exploiting both static and motion cues in Eq. (3.14). 91

4.1 Image alignment resides at different levels. Researchers used to study image alignment problem at pixel level (a) where the two images are taken at the same scene with slightly different time or perspective [98]. Recently, correspondence has been established at object level (b) for object recognition [12]. We are interested in image alignment at scene level, where two images come from the same scene category but different instances. As shown in (c), scene alignment is a challenging problem; the correspondence between scenes is not as obvious to human eyes as the correspondence at pixel and object levels. SIFT flow is proposed to align the examples in (c) for scene alignment. 95

4.2 **Mapping from SIFT space to RGB space**. To visualize SIFT images, we compute the top three principal components of SIFT descriptors from a set of images, and then map these principal components to the principal components of the RGB space. . . . 99

4.3 **Visualization of SIFT images**. We compute the SIFT descriptors on a regular dense grid. For each pixel in an image (a), the descriptor is a 128-D vector. The first 16 components are shown in (b) in a 4×4 image grid, where each component is the output of a signed oriented filter. The SIFT descriptors are quantized into visual words in (c). In order to improve the clarity of the visualization, cluster centers have been sorted according to the first principal component of the SIFT descriptor obtained from a large sample of our dataset. A visualization of SIFT image using the mapping function in Figure 4.2 is shown in (d). We will use (d) as our visualization of SIFT descriptors for the rest of the chapter. 99

4.4 An illustration of coarse-to-fine SIFT flow matching on pyramid. The green square is the searching window for $\mathbf{p}_k$ at each pyramid level k. For simplicity only one image is shown here, where $\mathbf{p}_k$ is on image s_1, and $\mathbf{c}_k$ and $\mathbf{w}(\mathbf{p}_k)$ are on image s_2. See text for details. 100

4.5 We generalized distance transform function for truncated L1 norm [36] to pass message between neighboring nodes that have different offsets (centroids) of the searching window. 102

4.6 Coarse-to-fine SIFT flow not only runs significantly faster, but also achieves lower energies most of the time. In this experiment, we randomly selected 10 samples in the test set and computed the lowest energy of the best match with the nearest neighbors. We tested both the coarse-to-fine algorithm proposed in this chapter and the ordinary matching scheme in [69]. Except for sample #8, coarse-to-fine matching achieves lower energy than the ordinary matching algorithm. 103

4.7 SIFT flow for image pairs depicting the same scene/object. (a) shows the query image and (b) its densely extracted SIFT descriptors. (c) and (d) show the best (lowest energy) match from the database and its SIFT descriptors, respectively. (e) shows (c) warped onto (a). (f) shows the warped SIFT image (d). (g) shows the estimated displacement field with the minimum alignment energy shown to the right. 105
4.8 SIFT flow computed for image pairs depicting the same scene/object category where the visual correspondence is obvious. 106
4.9 SIFT flow for challenging examples where the correspondence is not obvious. 107
4.10 Some failure examples with semantically incorrect correspondences. Although the minimized SIFT flow objectives are low for these samples (compared to those in Figure 4.9), the query images are rare in the database and the best SIFT flow matches do not belong to the same scene category as the queries. However, these failures can be overcome through increasing the size of the database. 107
4.11 Alignment typically improves ranking of the nearest neighbors. Images enclosed by the red rectangle are the top 10 nearest neighbors found by histogram intersection, displayed in a scan-line order (left to right, top to bottom). Images enclosed by the green rectangle are the top 10 nearest neighbors ranked by the minimum energy obtained by the alignment algorithm. The warped nearest neighbor image is displayed to the right of the original image. Note how the returned images are re-ranked according to the size of the depicted vehicle by matching the size of the bus in the query. 108
4.12 For an image pair such as row (1) or row (2), a user defines several sparse points in (a) as "+". The human annotated matchings are marked as dot in (b), from which a Gaussian distribution is estimated and displayed as an ellipse. The correspondence estimated from SIFT flow is marked as "$\times$" in (b). 110
4.13 The evaluation of SIFT flow using human annotation. Left: the probability of one human annotated flow lies within r distance to the SIFT flow as a function of r (red curve). For comparison, we plot the same probability for direct minimum L1-norm matching (blue curve). Clearly SIFT flow matches human perception better. Right: the histogram of the standard deviation of human annotation. Human perception of scene correspondence varies from subject to subject. 110
4.14 **Multiple motion field candidates**. A still query image with its temporally estimated motion field (in the green frame) and multiple motion fields predicted by motion transfer from a large video database. 112

4.15 **Evaluation of motion prediction**. (a) and (b) show normalized histograms of prediction rankings (result set size of 15). (c) shows the ranking precision as a function of the result set size. 113

4.16 **Motion instances where the predicted motion was not ranked closest to the ground truth**. A set of random motion fields (blue) together with the predicted motion field (green, ranked 3rd). The number above each image represents the fraction of the pixels that were correctly matched by comparing the motion against the ground truth. In this case, some random motion fields appear closer to the ground truth than our prediction (green). However, our prediction also represents a plausible motion for this scene. . . 113

4.17 **Motion from a single image**. The (a) original image, (b) matched frame from the video data set, (c) motion of (b), (d) warped and transferred motion field from (b), and (e) ground truth for (a). Note that the predicted motion in (d) is inferred from a single input still image, i.e. no motion signal is available to the algorithm. The predicted motion is based on the motion present in other videos with image content similar to the query image. 114

4.18 **Motion synthesis via object transfer.** Query images (a), the top video match (b), and representative frames from the synthesized sequence (c) obtained by transferring the moving objects from the video to the still query image. 115

4.19 **SIFT flow can be applied to aligning satellite images**. The two Mars satellite images (a) and (b) are taken at four years apart with different local appearances. The results of sparse feature detection and matching are shown in (c) to (g), whereas the results of SIFT flow are displayed in (h) to (k). The mean absolute error of the sparse feature approach is 0.030, while the mean absolute error of SIFT flow is 0.021, significantly lower. Visit webpage http://people.csail.mit.edu/celiu/SIFTflow/NGA/ for the animations of the warping. 117

4.20 **SIFT flow can be applied to image registration of the same scene but under different lighting and imaging conditions**. Column (a) and (b) are some examples from [130]. Even though originally designed for scene alignment, SIFT flow is also able to align these challenging pairs. Visit webpage http://people.csail.mit.edu/celiu/SIFTflow/NGA/ for the animations of the warping. 118

4.21 SIFT flow can serve as a tool to account for pose, expression and lighting changes for face recognition. (a): Ten samples of one subject in ORL database [93]. Notice pose and expression variations of these samples. (b): We select the first image as the query, apply SIFT flow to aligning the rest of the images to the query, and display the warped images with respect to the dense correspondence. The poses and expressions are rectified to that of the query after the warping. (c): The same as (b) except for choosing the fifth sample as the query. 119

4.22 **SIFT flow is applied for face recognition.** The curves in (a) and (b) are the performance plots for low-res and high-res images in the ORL face database, respectively. SIFT flow significantly boosted the recognition rate especially when there are not enough training samples. 121

5.1 For a query image (a), our system finds the top matches (b) (three are shown here) using a modified, coarse-to-fine SIFT flow matching algorithm. The annotations of the top matches (c) are transferred and integrated to parse the input image as shown in (d). For comparison, the ground-truth user annotation of (a) is shown in (e). 126

5.2 For a query image, we first find a K-nearest neighbor set in the database using GIST matching [82]. The nearest neighbors are re-ranked using SIFT flow matching scores, and form a top M-voting candidate set. The annotations are transferred from the voting candidates to the query image. 128

5.3 Above: the per-pixel frequency counts of the object categories in our dataset (sorted in descending order). The color of each bar is the average RGB value of each object category from the training data with saturation and brightness boosted for visualization. Bottom: the spatial priors of the object categories in the database. White means zero and the saturated color means high probability. 130

5.4 System overview. Our algorithm computes the SIFT image (b) of an query image (a), and uses GIST [82] to find its K nearest neighbors in our database. We apply coarse-to-fine SIFT flow to align the query image to the nearest neighbors, and obtain top M as voting candidates ($M = 3$ here). (c) to (e): the RGB image, SIFT image and user annotation of the voting candidates. (f): the inferred SIFT flow. From (g) to (i) are the warped version of (c) to (e) with respect to the SIFT flow in (f). Notice the similarity between (a) and (g), (b) and (h). Our system combines the voting from multiple candidates and generates scene parsing in (j) by optimizing the posterior. (k): the ground-truth annotation of (a). 131

5.5 The per-class recognition rate of our system and the one in [106]. (a) Our system with the the parameters optimized for pixel-wise recognition rate. (b) Our system with $\alpha = \beta = 0$, namely, with the Markov random field model turned off. (c) The performance of the system in [106] also with the conditional random field turned off, trained and tested on the same data sets as (a) and (b). (d) Our system, but matching RGB instead of matching SIFT. We may observe that the our optimized system (a) is biased towards "stuff", large-region objects such as *sky*, *building*, *mountain*, and *tree*, because of the biased prior distribution of the classes as illustrated in Figure 5.3. When we sacrifice the overall performance by turning off the Markov random field model in Eqn. 5.1, our system performs better for small-size objects. To compare, we downloaded the code of a state-of-the-art object recognition system [106] based on training per-pixel classifiers using texton features, and ran the code on our dataset with the results shown in (c). Notice that the conditional random field model described in [106] is not available in the publicly available code. The fair comparison of (b) and (c) (both with spatial regularization turned off) suggests that our system, which is established upon dense scene alignment, outperforms [106], which relies on training pixel-wise classifiers. Lastly, we modified our system by matching RGB instead of SIFT without changing anything else, and showed the results in (d). Clearly, SIFT flow (a) results in better performance than optical flow (d), although optical flow produces reasonable results compared to [106]. . . . 132

5.6 (a): Recognition rate as a function of the number of nearest neighbors K and the number of voting candidates M. (b): recognition rate as a function of the number of nearest neighbors K. Clearly, prior and spatial smoothness help improve the recognition rate. 134

5.7 The ROC curve of each individual pixel-wise binary classifier. Red curve: our system after being converted to binary classifiers; blue curve: the system in [31]. We used convex hull to make the ROC curves strictly concave. The number (n, m) underneath the name of each plot is the quantity of the object instances in the test and training set, respectively. For example, (170, 2124) under "sky" means that there are 170 test images containing sky, and 2124 training images containing sky. Our system obtains reasonable performance for objects with sufficient samples in both training and test sets, *e.g.*, sky, building, mountain and tree. We observe truncation in the ROC curves where there are not enough test samples, *e.g.*, field, sea, river, grass, plant, car and sand. The performance is poor for objects without enough training samples, *e.g.*, crosswalk, sign, boat, pole, sun and bird. The ROC does not exist for objects without any test samples, *e.g.*, desert, cow and moon. In comparison, our system outperforms or equals [31] for all object categories except for grass, plant, boat, person and bus. The performance of [31] on our database is low because the objects have drastically different poses and appearances. 135

5.8 Some scene parsing results output from our system. (a): query image; (b): the best match from nearest neighbors; (c): the annotation of the best match; (d): the warped version of (b) according to the SIFT flow field; (e): the inferred per-pixel parsing after combining multiple voting candidates; (f): the ground truth annotation of (a). The dark gray pixels in (f) are "unlabeled". Notice how our system generates a reasonable parsing even for these "unlabeled" pixels. . . 136

5.9 Some scene parsing results output from our system. (a): query image; (b): the best match from nearest neighbors; (c): the annotation of the best match; (d): the warped version of (b) according to the SIFT flow field; (e): the inferred per-pixel parsing after combining multiple voting candidates; (f): the ground truth annotation of (a). The dark gray pixels in (f) are "unlabeled". Notice how our system generates a reasonable parsing even for these "unlabeled" pixels. . . 137

5.10 Our system fails when no good matches can be retrieved in the database. Since the best matches do not contain river, the input image is mistakenly parsed as a scene of grass, tree and mountain in (e). The ground-truth annotation is in (f). . 138

A.1 The factor graph of optical flow estimation. Notice that the flow fields are coupled in (b) and (c) in different ways. 147

B.1 Both power function $\phi(x^2) = (x^2)^\alpha$ and Lorentzian function $\phi(x^2) = \log(1 + x^2)$ can be upper-bounded by quadratic functions. 151

List of Tables

4.1 SIFT flow outperforms the state-of-the-art [24] when there are only few (one or two) training samples. 122

Chapter 1

Introduction

We are interested in motion analysis, namely to study the projected motions of objects in digital images. Our goal is to make computers understand the motion of the visual world as humans perceive it.

Motion perception is an important task in our daily lives. When we stand on a street, we can perceive how fast cars move and predict their trajectories to avoid being hit. As we walk, we constantly perceive the motion of the environment and we can locate where we are in the environment as well as know the direction that we are moving in. There is sufficient evidence that motion perception plays an important role in the early stage when humans start to know and interact with the visual world [8].

Humans are experts at perceiving motion. Our visual system is more sensitive to moving objects than still objects. We are good at predicting the direction and velocity of moving objects. In fact, motion perception provides reliable feedback for us to interact with the world. Moreover, humans can easily perceive the boundary and shape of moving objects. Human motion perception is also very robust to transparency, shadow, noise, occlusion, lighting changes, *etc.*

Because of the importance of motion perception for humans, motion analysis has been one of the central topics in computer vision. From the early work of optical flow [71, 51], researchers have developed a variety of models and computational tools to analyze motion, especially from video sequences. Preliminary success has been achieved for estimating pixel-wise flow, tracking features, and segmenting moving objects. The motion information of a video sequence is the key for video compression, 3D geometry reconstruction [116], object tracking [95], segmentation [30] and advanced video editing [94].

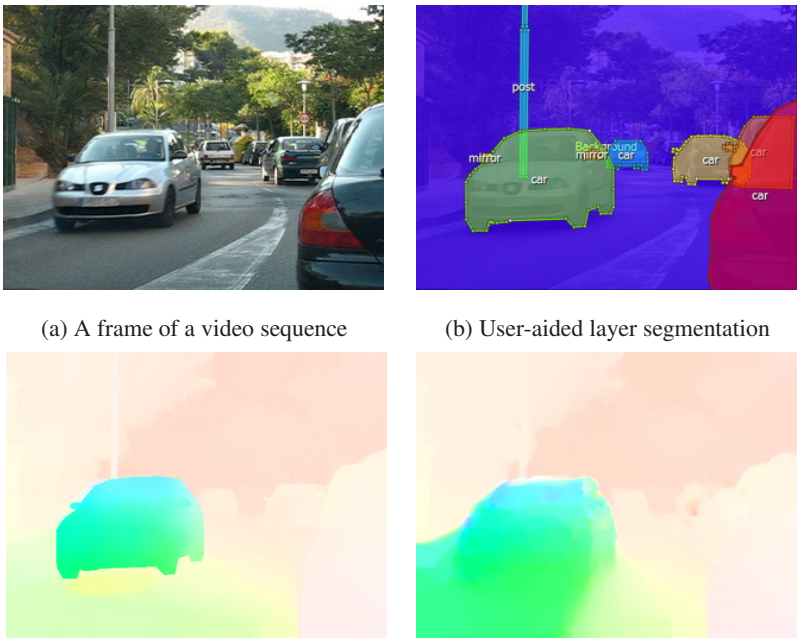
However, it is a nontrivial problem for computers to understand motion as humans do. The true motion is ambiguous from only a local analysis, an effect referred to as the aperture problem [51]. In order to reduce the ambiguity, motion estimation was constrained through

spatial regularization, namely neighboring pixels are likely to move together. But it is not clear which pixels in a neighborhood should move together merely from local information. Furthermore, in motion analysis, the brightness constancy assumption that pixel values are invariant from one frame to another along with the motion, often breaks because of illumination changes and noise. Although illumination invariant features such as image gradient are used for motion estimation [20], it is still challenging to estimate motion (or correspondence) for images with very different appearances.

Therefore, it is important to analyze motion beyond pixel levels. People have proposed models such as layers [123, 127] to explicitly model the grouping relationship of pixels in motion analysis. But automatic layer analysis remains a hard problem, and the current local motion-based layer analysis framework cannot effectively estimate motion for even seemingly simple cases, *e.g.*, , textureless objects under occlusion. Other forms of motion analysis such as motion particles [95] have been proposed, but they often rely on traditional motion analysis algorithms such as optical flow, and therefore inherit the same drawback that motion is analyzed without the right grouping.

Moreover, most motion analysis algorithms suffer from a lack of training datasets providing ground-truth motions for challenging, real-world videos. The video sequences for which we have ground-truth data are limited to controlled, indoor environments and constrained motions [7]. Algorithm training on and evaluation of these datasets do not generalize well to videos of uncontrolled, real-world environments. But obtaining the ground-truth motion for real-world videos is a challenging problem.

Another field with great attention in computer vision is object recognition and scene parsing. Traditional approaches to object recognition begin by specifying an object model, such as template matching [121, 31], constellations [37, 35], bags of features [107, 57, 44, 109], shape models [10, 12, 34], *etc.* These approaches typically work with a fixed-number of object categories and require training generative or discriminative models for each category given training data. Is it possible to build an object recognition system without training any generative models or classifiers? Object recognition and motion analysis used to be disjoint fields in computer vision, but we are curious about how motion analysis algorithms can be integrated into building an effective object recognition and scene parsing system.



(a) A frame of a video sequence (b) User-aided layer segmentation

(c) User-annotated motion (d) Output of a flow algorithm [22]

Figure 1.1. Human-assisted motion analysis. In Chapter 2, we designed a system to allow the user to specify layer configurations and motion hints (b). Our system uses these hints to calculate a dense flow field for each layer. We show that the flow (c) is repeatable and accurate. (d): The output of a representative optical flow algorithm [22], trained on the Yosemite sequence, shows many differences from the labeled ground truth for this and other realistic sequences we have labeled. This indicates the value of our database for training and evaluating optical flow algorithms.

■ 1.1 Thesis Theme

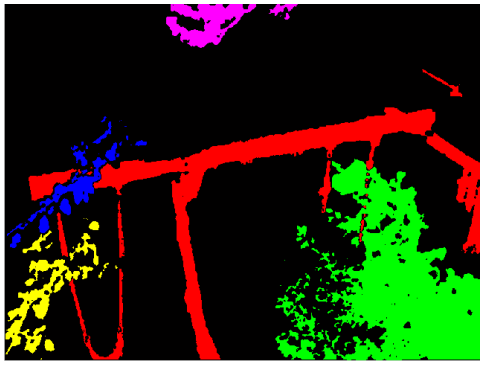
Instead of improving existing models to achieve more accurate motion estimation, we feel that it is important to look at motion analysis in a bigger picture. In particular, we want to expand the frontier of motion analysis to beyond the pixel level in the following three aspects:

(a) **Obtaining ground truth for real-world videos.** We seek to measure the ground-truth motion of real-world videos not only to benchmark algorithms and obtain the statistics of the real-world motion, but also to learn how to make humans and computers work together, a process to make automatic vision algorithms more mature and robust.

(b) **Proposing new representations.** We feel that motion analysis is not merely a correspondence problem, but is rather entangled with groupings. We explore new representations

(a) Registered input frame

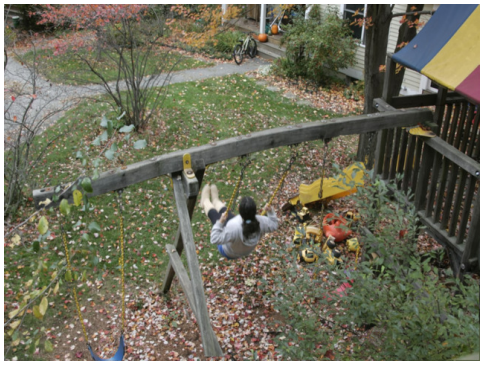
(b) Clustered trajectories of tracked features



(c) Layers of related motion and appearance

(d) Motion magnified, showing holes

(e) After texture in-painting to fill holes

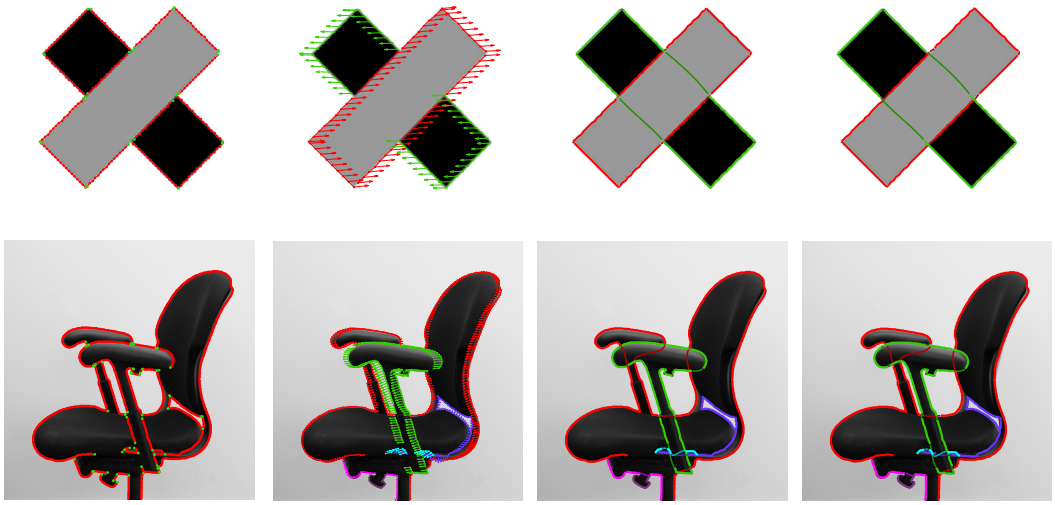


(f) After user's modifications to segmentation map in (c)

Figure 1.2. Motion magnification is a visual system to magnify small motions in video sequences [67]. See details in Chapter 3.1.

such as layers and contours [1] to analyze motion under some special conditions. We also

[1] We are not claiming that we invent layer or contour representations for motion analysis. Layer and contours are "new" representations compared to the pixel-wise flow field representation. However, the algorithms to extract layers and contours are new in this thesis.



(a) Extracted boundaries (b) Estimated flow (c) Frame 1 (d) Frame 2

Figure 1.3. Analysis of contour motions [61]. Column (a): boundary fragments are extracted using our boundary tracker. The red dots are the edgelets and the green ones are the boundary fragment ends. Column (b): boundary fragments are grouped into contours and the flow vectors are estimated. Each contour is shown in its own color. Column (c) & (d): the illusory boundaries are generated for the first and second frames. See details in Chapter 3.2.

try to match local image structures instead of image intensities, and obtain correspondences between seemingly impossible examples.

(c) **Exploring new applications.** The representations beyond pixel-level lead to many new applications that traditional pixel-wise algorithms cannot achieve. For example, we use the layer representation to detect and magnify small motions, and a contour representation to analyze the motion of textureless objects under occlusion. Through matching image structures, we are able to establish dense scene correspondence, which further enables a large database-driven framework for image analysis and synthesis.

■ 1.2 Thesis Overview

The thesis consists of several projects I explored during my PhD.

We first explore obtaining motion ground truth for real-world videos. In Chapter 2 *Human-Assisted Motion Annotation* [63], we designed a human-computer interactive system to allow users to annotate pixel-wise flow information for a video sequence (Figure 1.1). Our system uses layers as the interface for humans to interact with computers. We used robust object track-

Figure 1.4. SIFT flow for dense scene alignment. In Chapter 4, we extend the concept of "motion" to the correspondence across scenes. (a) and (b) show images of similar scenes. (b) was obtained by matching (a) to a large image collection. (c) shows image (b) warped to align with (a) using the estimated dense correspondence field. (d) Visualization of pixel displacements using the color coding scheme of [7]. Note the variation in scene appearance between (a) and (b). The visual resemblance of (a) and (c) demonstrates the quality of the scene alignment.

ing and layer-wise optical flow estimation algorithms to obtain pixel-wise flow. Our system also allows users to annotate motion using sparse feature points when optical flow estimation fails due to shadow, transparency and noise. The user feedback helps our system to accurately quantify human perception of motion as ground-truth annotation. We have used our system to annotate the ground-truth motion of a number of real-world videos, which can be further used for benchmarking and obtaining motion statistics.

We are interested in exploring novel representations beyond pixel level, *e.g.*, layers and contours, for automatic motion analysis. In Chapter 3.1 we present a *Motion Magnification* system [67] to magnify small motions to make them salient to human eyes (Figure 1.2). The core of our motion magnification system is accurate, automatic layer motion analysis. After a video sequence is parsed into layers and the user selects a particular layer, the system is able to magnify the motion of the layer and fill in the missing pixels in the background. We demonstrated broad applications of motion magnification in digital entertainment, surveillance,

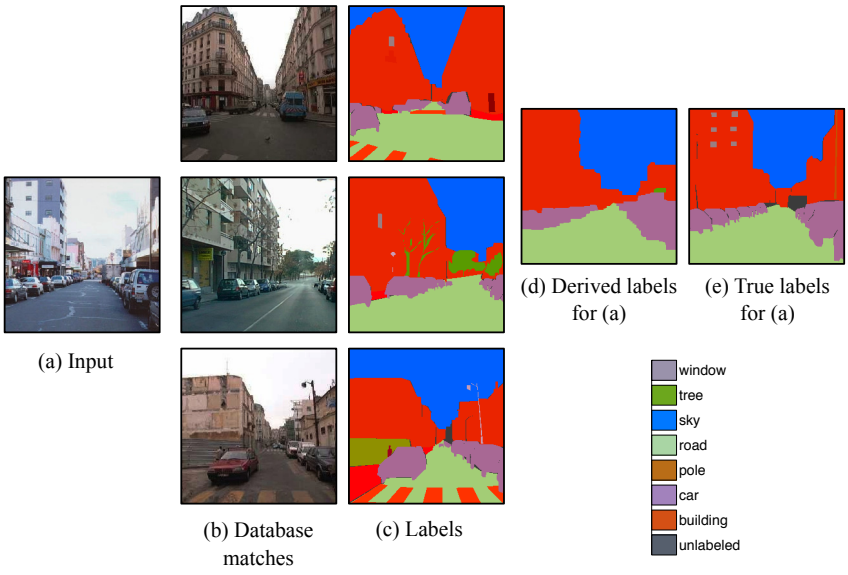


Figure 1.5. Nonparametric scene parsing. For a query image (a), we designed a system to find the top matches (b) (three are shown here) using a coarse-to-fine SIFT flow matching algorithm. The annotations of the top matches (c) are transferred and integrated to parse the input image as shown in (d). For comparison, the ground-truth user annotation of (a) is shown in (e). See Chapter 5 for details.

and visualization, as a tool to reveal the underlying physical mechanism of the visual world.

In Chapter 3.2 we introduce a *Contour Motion Analysis* system [61][2] to analyze motion for textureless objects under occlusion (Figure 1.3). Traditional motion analysis algorithms such as optical flow fail miserably for analyzing the motion of textureless objects under occlusion because only one flow vector is allowed for one pixel, and pixels are grouped uniformly with neighboring pixels. Stimuli such as T-junctions and illusory boundaries would confuse the algorithms based on this single flow vector representation. We propose to analyze motion on contours to deal with the aperture problem: local motion ambiguity is modeled during the bottom-up process, and then resolved during the contour forming process. The success of our contour motion analysis model on some challenging examples illustrates the importance of the right representation for motion analysis.

Although much has been studied, the realm of motion analysis was limited to the correspondence between adjacent frames in video sequences. In Chapter 4, we extend the concept

[2]The paper received the *Outstanding Student Paper Award* from *Advances in Neural Information Processing Systems* (NIPS), 2006.

of "motion" to general images through proposing *SIFT Flow* (SIFT stands for Scale-Invariant Feature Transform [70]), a method to align images across scenes (Figure 1.4). The SIFT flow algorithm consists of matching densely sampled, pixel-wise SIFT features between two images, while preserving spatial discontinuities. Through many examples, we demonstrate that SIFT flow is able to establish semantically meaningful correspondence between drastically different images. We propose novel applications such as predicting motion fields from a single image and motion synthesis via object transfer based on SIFT flow. We also show how SIFT flow can be applied to traditional image alignment problems such as satellite image registration and face recognition.

There is a rich set of applications of dense scene alignment for computer vision. In Chapter 5 we study *Nonparametric Scene Parsing* [68][3] using the dense scene alignment model developed in Chapter 4 (Figure 1.5). Once we establish the dense correspondence between a query, unlabeled image to the images in a large, labeled database, we are able to transfer the user annotation of objects from the images in the database to the query image for scene parsing. Our method is easy to implement, does not require training classifiers, has few parameters, and outperforms the state-of-the-art object recognition algorithms on a challenging database.

■ 1.3 Other Work not in the Thesis

During my doctorial training, I have also visited problems other than motion analysis. With Dr. Richard Szeliski, Dr. Sing Bing Kang, Dr. Larry Zitnick and Prof. William T. Freeman, I studied automatically estimating noise level functions (NLFs) from a single color image [64] and applying NLFs for automatic noise removal [66]. Under the supervision of Dr. Heung-Yeung Shum and Prof. William T. Freeman, I designed the first practical face hallucination system [65] to detect, align and super-resolve small faces in photos. Prof. Marshall Tappen and I co-developed a general framework for learning Gaussian conditional random fields for low-level vision [115]. I also participated in object recognition by scene alignment [90] with Dr. Bryan Russell, Prof. Antonio Torralba, Prof. Rob Fergus and Prof. William T. Freeman. Interested readers can refer to these publications for details.

[3] The paper received the *Best Student Paper Award* from *IEEE Conference on Computer Vision and Pattern Recognition* (CVPR), 2009.

Chapter 2

Human-Assisted Motion Annotation

When you can measure what you are speaking about and express it in numbers, you know something about it; but when you cannot measure it, when you cannot express it in numbers, your knowledge is of the meager and unsatisfactory kind.

–Lord Kevin

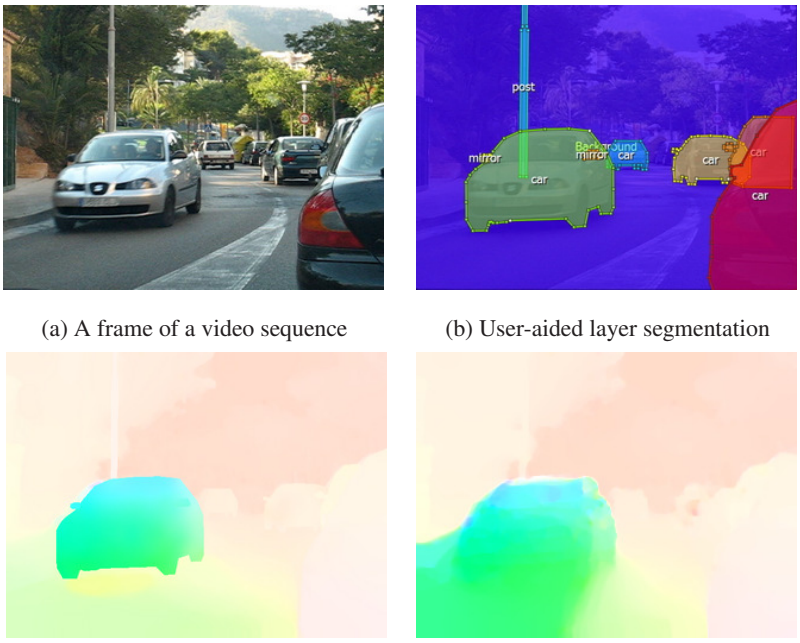
2.1 Introduction

Motion analysis has been an essential component for many computer vision applications such as structure from motion, object tracking/recognition, and advanced video editing. A variety of computational models, *e.g.*, optical flow fields [51, 71], layers [124] and boundaries [61], have been developed to estimate the motion from a video sequence. It is important to evaluate the performance of various motion analysis algorithms to gain insight and design better models.

However, little has been done for evaluating motion analysis algorithms compared to the tremendous effort put into developing these algorithms. For example, researchers have tried to optimize optical flow algorithms based on a synthetic sequence *Yosemite*, an overly simplified example compared to real-life videos [6]. As a result, the performance of the optical flow algorithms optimized for the Yosemite sequence may deteriorate significantly when sensor noise, motion blur, occlusion, shadow, reflection, auto white balance and compression artifacts occur in a sequence.

While it is important to have a ground-truth optical flow database consisting of real-world videos, annotating the motion for real-life videos can be challenging. Recently, Baker *et al.* designed a new database for optical flow evaluation [7]. Although the ground-truth flow of several sequences was measured, their methodology of painting fluorescent materials to track motion is limited to indoor scenes and artificial motion.

To address this problem, we propose a human-in-loop methodology to annotate ground-



(a) A frame of a video sequence (b) User-aided layer segmentation

(c) User-annotated motion (d) Output of a flow algorithm [22]

Figure 2.1. We designed a system to allow the user to specify layer configurations and motion hints (b). Our system uses these hints to calculate a dense flow field for each layer. We show that the flow (c) is repeatable and accurate. (d): The output of a representative optical flow algorithm [22], trained on the Yosemite sequence, shows many differences from the labeled ground truth for this and other realistic sequences we have labeled. This indicates the value of our database for training and evaluating optical flow algorithms.

truth motion for arbitrary real-world videos. Our approach to obtaining motion ground truth is based on three observations. First, humans are experts at segmenting layers in a video sequence because we can easily recognize the moving objects and their relative depth relationships. Second, we are sensitive to any differences between two images when these two images are displayed back and forth. Third, humans have a knowledge of the smoothness and discontinuities of the motion that a moving object undergoes. In fact, computer vision researchers have implicitly used these observations to inspect the accuracy of motion estimates when the ground truth is missing, and even to verify the correctness of ground-truth annotation.

Labeling motion pixel by pixel and frame by frame can be tedious, and we designed a computer vision system to ease the labeling process. Typically, it takes four steps to label the motion of a video sequence with our system:

(a) **Semi-automatic layer segmentation**. The user specifies the contour and relative depth of an object in one frame and the system automatically tracks the contour for the rest of

the frames with appropriate occlusion handing.

(b) **Automatic layer-wise flow estimation**. The user specifies a set of parameters for optical flow estimation for each of the layers, and selects the flow field that both produces the best visual match and agrees with the smoothness and discontinuity of a layer.

(c) **Semi-automatic motion labeling**. If the user is not satisfied with the flow estimation of a layer in step (b), the user can specify sparse correspondence and the system automatically fits parametric motion or interpolates a dense flow field until the user is satisfied with the motion.

(d) **Automatic compositing**. The system automatically composes the labeled motion of each layer into a full-frame flow field.

To evaluate our methodology, we provide quantitative evidence that human annotated motion is significantly closer to the real ground truth than any of the existing computer vision algorithms, and that human annotations made using our system are consistent across subjects.

Using our system, we have obtained and carefully labeled 10 video sequences with 341 total frames for both indoor and outdoor scenes (and we continue to label sequences). We also reported the performance of current optical flow algorithms on this database; we hope our labeled dataset will provide the feedback necessary to improve the state of the art for motion analysis in realistic video sequences. An additional byproduct of our system is ground-truth layer segmentation that can be used for evaluating layer segmentation and occlusion boundary detection algorithms. We make available the source code, labeling tool and database online for public evaluation and benchmarking `http://people.csail.mit.edu/celiu/motion/`.

■ 2.2 Related Work

Modeling and estimating dense optical flow fields have been intensively studied in the literature. Starting with Horn & Schunck [51] and Lucas & Kanade [71], researchers have developed a variety of models for effective flow computation, including some recent work such as incorporating robustness functions [14], integrating gradient information [20], estimating a symmetric flow field [4], combining local and global flow [22], and reasoning about occluded/disoccluded pixels (outliers) [95]. The success of optical flow estimation has been shown on the Yosemite sequence, with the average angular error (AAE) within 2°, but this success does not reveal many examples where optical flow algorithms may fail. One motivation of our work is to have a ground-truth motion database with diverse examples to reveal these failures and understand the cause.

Researchers have argued that higher level representations such as layers and contours should be used to analyze motion. Since Adelson proposed the layer representation for videos [1], many computational models have been developed for automatic layer estimation [124, 127, 117, 128, 67]. Recently, preliminary success was obtained using a boundary representation for motion analysis [61]. Although most of the layer and boundary motion work focused on obtaining the correct segmentation, these higher level representations should also result in better full-frame motion estimation. Unfortunately, the goal of producing more accurate motion is missing from the existing work. We want to obtain a database to evaluate layer/boundary analysis algorithms, as well.

Closest to our goal is Baker *et al.* 's recent work [7], where the authors designed an elegant approach to obtaining ground truth optical flow. They painted fluorescent patterns onto objects in an indoor experimental setup and used a computer-controlled stage to generate motion. By taking two pictures simultaneously under ambient and UV light, they were able to capture both a natural image and an image with rich textures from the fluorescent patterns. A feature tracking system was designed to produce the ground-truth motion. Although the motion of a few sequences were successfully measured, their methodology is limited to controlled materials, motion and lighting conditions. In parallel, Roth and Black [87] obtained a synthetic ground-truth motion database using an existing depth database. In contrast, our approach is able to generate ground-truth motion for real-world videos under much broader imaging conditions.

Image annotation has been explored in the literature, such as the Berkeley image segmentation database [75] and LabelMe object annotation database [91]. These ground-truth databases have led to not only algorithm evaluation but also many other vision applications. One of our goals is to provide a broad ground-truth motion database for the computer vision community. Nevertheless, compared to these static image annotation tools which are relatively easy to design, designing a tool for motion annotation is nontrivial because accurate motion annotation and temporal consistency are required.

Interactive object segmentation in video sequences has recently been explored in computer graphics, including contour-based video rotoscoping [3], soft alpha matting [27], over-segmentation [122] and 3D graph cut [59]. We use the contour-based model [3] in our system because contours are intuitive to interact with. We allow the user to specify depth information so that contours can be tracked appropriately under occlusion. The goal of our system is to obtain both segmentation and motion, but our focus is motion, whereas these interactive segmentation tools focused on segmentation.

■ 2.3 Human-Assisted Motion Annotation System

Accurately labeling every pixel in every frame of a video sequence is a nontrivial task, and we designed a computer vision based system to allow the user to annotate motion in a reasonable amount of time. We assume that a video sequence can be decomposed into layers, each of which has a binary mask and smooth flow field. There are three main components in our system: *semiautomatic layer segmentation*, *automatic optical flow estimation* and *semiautomatic motion labeling*. The model that we designed for each of the components will be explained in Sect 2.3.1 through 2.3.7. The human interaction and the graphical user interface will be briefly described in Sect 2.3.8.

We used the state-of-the-art computer vision algorithms to design our system. Many of the objective functions in contour tracking, flow estimation and flow interpolation have $L1$ norm for robustness concerns. Techniques such as iterative reweighted least squared (IRLS) [22, 20], pyramid-based coarse-to-fine search [14, 22] and occlusion/outlier detection [95] were intensively used for optimizing these nonlinear objective functions.

One core technique of our system is optical flow estimation. The details of deriving optical flow using iterative reweighted least square (IRLS) can be found in Appendix A. This IRLS algorithm is the same as the ones derived from traditional Euler-Lagrange equations [22, 20], but derivation using IRLS is more succinct. In Appendix B, we show that IRLS is also equivalent to variational optimization [115]. Therefore, we choose IRLS as the main optimization tool for our system. In Sect 2.3.1, we shall go through the details of IRLS for contour tracking. We omit the details of deriving a symmetric optical flow in Sect 2.3.6 as similar procedure can be found in Appendix A.

■ 2.3.1 Semiautomatic Layer Segmentation with Occlusion Handling

In this module, the user labels the contour of a moving layer in one frame, and the system automatically tracks the contour for the rest of frames. By allowing the user to specify a time-varying depth value for each layer, our system can handle contour tracking under occlusion. The user can correct the tracking, and the system automatically propagates the correction to other frames.

We feel that realtime performance is more important than accuracy for this user-interactive tracker. Therefore, we did not choose slow but accurate trackers such as particle filtering [52]. Our contour tracker is designed based on the models for optical flow estimation with occlusion handling incorporated.

Suppose the user defined the contour of an object using landmarks, a set of points. Computer's job is to track the contour across the rest of the frames. We do not expect the computer can do this tracking job perfectly. The user can always correct the tracking and let the computer re-track the contour. However, we shall focus on the mathematical model of tracking contours. We shall particularly focus on two-frame matching problem in this section for the basic model.

■ 2.3.2 Objective function

Let polygon $\mathcal{L} = \{z_k : z_k \in \mathbb{R}^2\}_{k=1}^N$ be the user-defined landmarks for frame I_1. We are interested at where $\{z_k\}$ would be at frame I_2, and use $\{w_k\}$ to denote the flow vector from I_1 to I_2 associated with each landmark. Intuitively, we want the contour to move coherently and match local image features. The objective function is thus

$$E(\{w_k\}) = \sum_{k=1}^{N} \sum_{q \in N_k} r_k(q) \|I_2(z_k + w_k + q) - I_1(z_k + q)\|^2 + \alpha \sum_{k=1}^{N} h_k \|w_k - w_{k+1}\|^2, \quad (2.1)$$

where we have circular definition $w_{N+1} = w_1$. The weight h_k is used to take into account the distance between point z_k and z_{k+1}. It is defined as

$$h_k = \frac{\beta}{d_k + \beta}, \quad (2.2)$$

where $d_k = \|z_k - z_{k+1}\|$, and $\beta = \overline{d_k}$. N_k is the neighborhood for point z_k, which is defined as a square patch centered at each landmark. r_k is the weighting function for each landmark, defined as

$$r_k(q) = \mathbb{1}[q + z_k \in \mathcal{L}] \cdot \exp\{-\frac{\|q\|^2}{2\sigma_r^2}\}, \quad (2.3)$$

where $\mathbb{1}[q + z_k \in \mathcal{L}]$ equals 1 if $q + z_k$ is inside the polygon $\mathcal{L}$, and 0 otherwise. This weighting function is also called region of support in tracking. An illustration of weighting function is shown in Figure 2.2. In Eqn. (2.1), α is the coefficient for the regularization.

■ 2.3.3 Linearization and Optimization

The objective function in Eqn. (2.1) is highly nonlinear. Similar to many of the optical flow algorithms, we choose a coarse-to-fine searching scheme and iteratively update the flow flow vectors $\{w_k\}$. Suppose we already have an initial guess of $\{w_k\}$ and our goal is to compute the optimal adjustment $\{dw_k : dw_k = [du_k, dv_k]^T\}$ for each feature point. The new objective

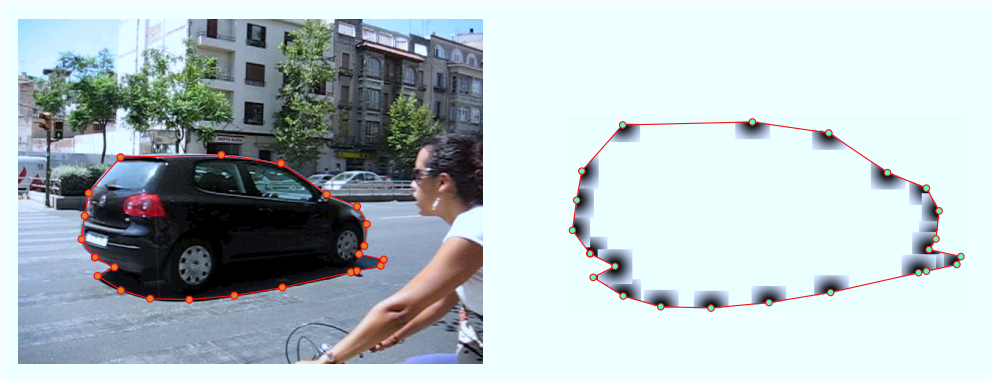


Figure 2.2. A weighting function, $r_k(q)$, a.k.a. region of support, for each feature point is used in the data term of the contour tracking. The weight function is a product of whether a pixel being inside the contour and a Gaussian modulation. Left: the user specified the contour of a van. Right: the weighting function of each key point. The darker the pixel value, the higher the weight.

function becomes

$$E(\{w_k\}) = \sum_{k=1}^{N} \sum_{q \in N_k} r_k(q) \| I_2(z_k + w_k + dw_k + q) - I_1(z_k + q) \|^2 + \alpha \sum_{k=1}^{N} h_k \| w_k - w_{k+1} + dw_k - dw_{k+1} \|^2, \quad (2.4)$$

The data term of the objective function is still nonlinear. We can linearize the data term using Taylor expansion

$$I_2(z_k+w_k+dw_k+q) - I_1(z_k+q) = I_x(z_k+w_k+q)du_k + I_y(z_k+w_k+q)dv_k + I_t(z_k+w_k+q), \quad (2.5)$$

where each of the terms is defined as

$$I_x(z_k+w_k+q) = \frac{\partial I_2(z_k+w_k+q)}{\partial x} \quad (2.6)$$

$$I_y(z_k+w_k+q) = \frac{\partial I_2(z_k+w_k+q)}{\partial y} \quad (2.7)$$

$$I_t(z_k+w_k+q) = I_2(z_k+w_k+q) - I_1(z_k+q) \quad (2.8)$$

We can now rewrite the data term in quadratic form:

$$
\begin{aligned}
&\sum_{k=1}^{N}\sum_{q\in N_k} r_k(q)\big\|I_2(z_k+w_k+dw_k+q)-I_1(z_k+q)\big\|^2\\
&=\sum_{k=1}^{N}\sum_{q\in N_k} r_k(q)\big\|I_x(z_k+w_k+q)du_k+I_y(z_k+w_k+q)dv_k+I_t(z_k+w_k+q)\big\|^2\\
&=\sum_{k=1}^{N}\sum_{q\in N_k} r_k(q)\left\|\begin{bmatrix}I_x(z_k+w_k+q) & I_y(z_k+w_k+q)\end{bmatrix}\begin{bmatrix}du_k\\ dv_k\end{bmatrix}+I_t(z_k+w_k+q)\right\|^2\\
&=\sum_{k=1}^{N}\sum_{q\in N_k} r_k(q)\left\{[du_k\ dv_k]\begin{bmatrix}I_x^2(z_k+w_k+q) & I_{xy}(z_k+w_k+q)\\ I_{xy}(z_k+w_k+q) & I_y^2(z_k+w_k+q)\end{bmatrix}\begin{bmatrix}du_k\\ dv_k\end{bmatrix}+\right.\\
&\quad\left.2[du_k\ dv_k]\begin{bmatrix}I_{xt}(z_k+w_k+q)\\ I_{yt}(z_k+w_k+q)\end{bmatrix}\right\}+\mathrm{C}\\
&=\sum_{k=1}^{N}\left\{[du_k\ dv_k]\begin{bmatrix}\Psi_{xx}^{(k)} & \Psi_{xy}^{(k)}\\ \Psi_{xy}^{(k)} & \Psi_{yy}^{(k)}\end{bmatrix}\begin{bmatrix}du_k\\ dv_k\end{bmatrix}+2[du_k\ dv_k]\begin{bmatrix}\Psi_{xt}^{(k)}\\ \Psi_{yt}^{(k)}\end{bmatrix}\right\}+\mathrm{C}\\
&=[dU^T dV^T]\begin{bmatrix}\boldsymbol{\Psi}_{xx} & \boldsymbol{\Psi}_{xy}\\ \boldsymbol{\Psi}_{xy} & \boldsymbol{\Psi}_{yy}\end{bmatrix}\begin{bmatrix}dU\\ dV\end{bmatrix}+2[dU^T dV^T]\begin{bmatrix}\boldsymbol{\Psi}_{xt}\\ \boldsymbol{\Psi}_{yt}\end{bmatrix}+\mathrm{C},
\end{aligned}
\tag{2.9}
$$

where C is a constant. $\Psi_{xx}^{(k)}$ is defined as

$$\Psi_{xx}^{(k)}=\sum_{q\in N_k} r_k(q)I_x^2(z_k+w_k+q). \tag{2.10}$$

$\Psi_{xy}^{(k)}$, $\Psi_{yy}^{(k)}$, $\Psi_{xt}^{(k)}$ and $\Psi_{yt}^{(k)}$ are defined similarly. $I_{xy}(z_k+w_k+q)$ is a shorthand of the product $I_x(z_k+w_k+q)I_y(z_k+w_k+q)$, and so for $I_{xt}(z_k+w_k+q)$ and $I_{yt}(z_k+w_k+q)$. The diagonal matrix $\boldsymbol{\Psi}_{xx}$ is defined as

$$\boldsymbol{\Psi}_{xx}=\mathrm{diag}(\Psi_{xx}^{(1)},\cdots,\Psi_{xx}^{(N)}), \tag{2.11}$$

and $\boldsymbol{\Psi}_{xy}$, $\boldsymbol{\Psi}_{yy}$, $\boldsymbol{\Psi}_{xt}$ and $\boldsymbol{\Psi}_{yt}$ are defined similarly.

Now let us visit the smoothness term in the objective function. The smoothness term can

also be rewritten in quadratic form

$$\begin{aligned}
&\sum_{k=1}^{N} h_k \|w_{k+1} - w_k + dw_{k+1} - dw_k\|^2 \\
&= \sum_{k=1}^{N} h_k \left[(u_{k+1} - u_k + du_{k+1} - du_k)^2 + (v_{k+1} - v_k + dv_{k+1} - dv_k)^2 \right] \\
&= \sum_{k=1}^{N} h_k \left[(du_{k+1} - du_k)^2 + 2(u_{k+1} - u_k)(du_{k+1} - du_k) + (dv_{k+1} - dv_k)^2 + \right. \\
&\quad \left. 2(v_{k+1} - v_k)(dv_{k+1} - dv_k) \right] + \mathrm{C}_1 \\
&= dU^T \mathbf{D}^T \mathbf{H} \mathbf{D} dU^T + dV^T \mathbf{D}^T \mathbf{H} \mathbf{D} dV^T + 2\mathbf{D}^T \mathbf{H} \mathbf{D} U + 2\mathbf{D}^T \mathbf{H} \mathbf{D} V + \mathrm{C}_1
\end{aligned} \tag{2.12}$$

where matrix $\mathbf{D}$ is a circular difference matrix, defined as

$$\mathbf{D} = \begin{bmatrix} -1 & 1 & & & \\ & -1 & 1 & & \\ & & \ddots & \ddots & \\ & & & -1 & 1 \\ 1 & & & & -1 \end{bmatrix}. \tag{2.13}$$

$\mathbf{H}$ is a diagonal matrix $\mathbf{H} = \mathrm{diag}(h_1, \cdots, h_N)$.

Therefore, we need to solve the following linear system to minimize the objective function

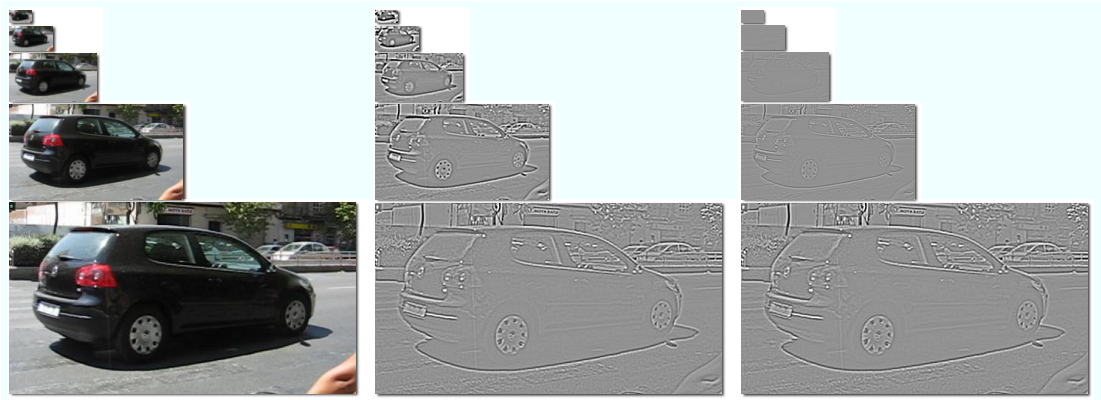
$$\begin{bmatrix} \mathbf{\Psi}_{xx} + \alpha \mathbf{D}^T \mathbf{H} \mathbf{D} & \mathbf{\Psi}_{xy} \\ \mathbf{\Psi}_{xy} & \mathbf{\Psi}_{yy} + \alpha \mathbf{D}^T \mathbf{H} \mathbf{D} \end{bmatrix} \begin{bmatrix} dU \\ dV \end{bmatrix} = - \begin{bmatrix} \mathbf{\Psi}_{xt} + \alpha \mathbf{D}^T \mathbf{H} \mathbf{D} U \\ \mathbf{\Psi}_{yt} + \alpha \mathbf{D}^T \mathbf{H} \mathbf{D} V \end{bmatrix} \tag{2.14}$$

Clearly, in this linear system $\mathbf{A}x = b$ matrix $\mathbf{A}$ is positive definite. So we use conjugate gradient method to solve the linear system, which converges in $2N$ iterations. The model we developed here is similar to optical flow estimation [51, 20].

■ 2.3.4 Multi-channel, multi-resolution image representation

It has been shown in [20] that the gradient information can help improve optical flow algorithm. We also want to use the full RGB channels to present the image I_1 and I_2. We extend image channels to include Laplacian-filtering response of the brightness image, and rewrite Eqn. (2.10) as

$$\Psi_{xx}^{(k)} = \sum_{q \in N_k} \sum_{l=1}^{L} r_k(q) I_x^2(z_k + w_k + q, l). \tag{2.15}$$



(a) Image Pyramid of RGB channels (b) Laplacian channel (c) The incorrect way of obtaining Laplacian channel

Figure 2.3. Multi-channel and multi-resolution representation for tracking. We use four channels for image presentation, where the first three are RGB (a), and the fourth one is the Laplacian filtering response of the brightness image (b). A Gaussian pyramid is built for initialization and acceleration. Notice that the Laplacian channel has to be obtained from each level of the RGB image, as described in [23]. The Laplacian channel becomes blurred and useless if the pyramid is directly built upon the finest level Laplacian image.

where $I_x(z, l)$ indicates the lth channel of image I_x.

The objective function in Eqn. (2.1) can only be linearized to Eqn. (2.4) when the two images I_1 and I_2 are close enough. This is not true in general for two consecutive frames. Similar to optical flow algorithms, we take a hierarchical, multi-resolution approach. Image pyramids are built for I_1 and I_2, and at the coarse level the two images are indeed close enough. We optimize the linearized objective function for a number of iterations, and then propagate the results to the next finer level.

This image representation is illustrated in Figure 2.3. Notice that we did not compute the Laplacian image for the finest level and build the pyramid of Laplacian image on top of the finest Laplacian image. This will make the Laplacian channel useless at the coarse level, as shown in Figure 2.3 (c). Instead, we first build the standard image pyramid, and then compute the Laplacian image for each level, as in (b).

■ 2.3.5 Occlusion handling

The tracking can go wrong unless we model the occlusion appropriately. We designed a UI to allow the users to specify the depth of each object at some key frames, and the system automatically interpolates a smooth depth function based on the user specification. A snapshot of this function is illustrated in Figure 2.4. We use HSV color space to visualize depth. We fix S (saturation) and V (value) to be maximum, and let H (hue) to be the depth value. From the

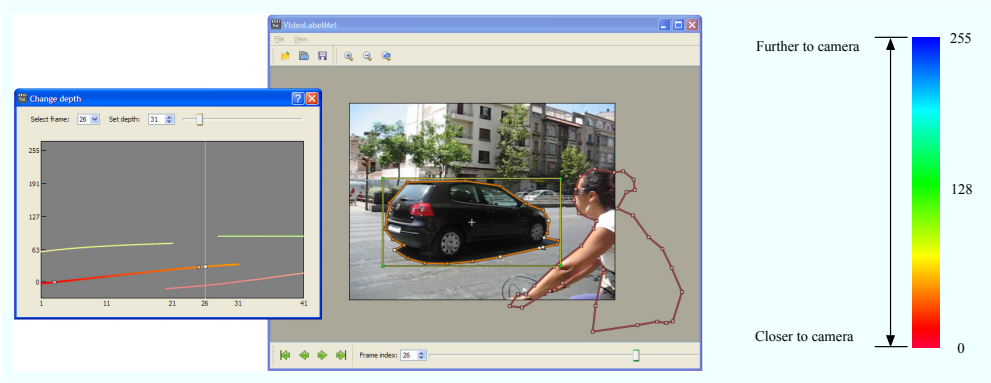


Figure 2.4. The system allows the user to dynamically change the depth of the object. The user selects the key frames to specify the value of depth, and the system automatically interpolates to the rest of the frames. We use color in HSV space to indicate the depth, by fixing S and V as 255, and letting H (hue) reflect the depth value. Intuitively, warmer color (red) indicates smaller value of depth, closer to the camera, whereas colder color (blue) indicates larger value of depth, further to the camera.

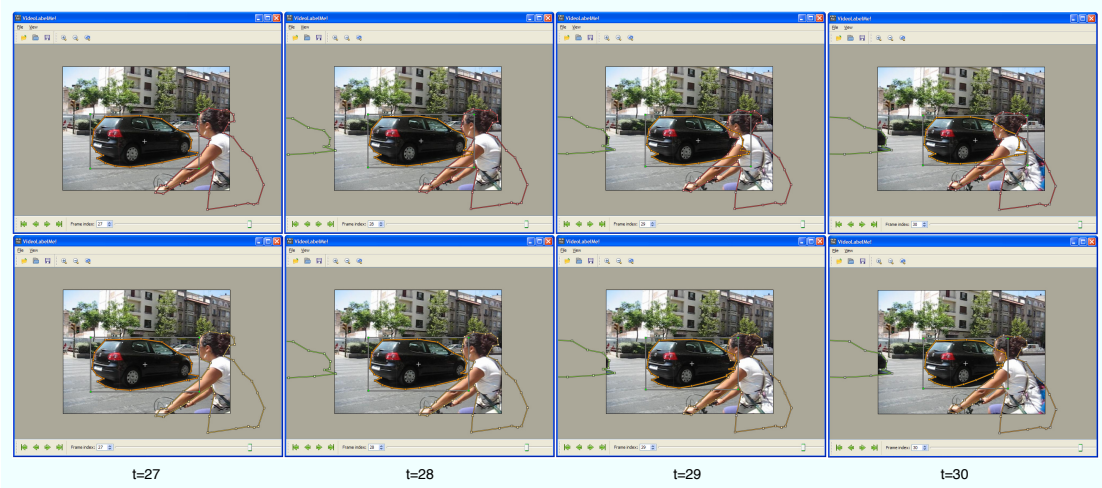


Figure 2.5. Occlusion reasoning is crucial for the correct tracking. Top row: the depth of the person is specified to be less than that of the car. The tracking was correct even though the car becomes occluded. Bottom row: the depth of the person is specified to be greater than that of the car. The tracking becomes wrong at the occlusion because the appearance of the person was mistakenly used for the appearance of the feature points of the car.

definition, warmer color (red) indicates smaller value of depth, closer to the camera, whereas colder color (blue) indicates larger value of depth, further from the camera.

We used 1D membrane model to interpolate the depth curve. Suppose the user specified the depth at frame $t_1, t_2, \cdots, t_n$ in the range $\{1, 2, \cdots, T\}$. Let $x_1 \in \mathbb{R}^M$ be the set of the depths that have been specified, and $\mathbf{T}_1 \in \mathbb{R}^{T \times M}$ be the matrix to transform x_1 to x. The ith

column of $\mathbf{T}_1$ is 1 at the t_ith row, and is zero elsewhere. Likewise, let $x_2 \in \mathbb{R}^{T-M}$ be the depth to interpolate and $\mathbf{T}_2$ be the matrix to transform x_2 to x. So we have $x = \mathbf{T}_1 x_1 + \mathbf{T}_2 x_2$. We want to x_2 to be interpolated so that the gradient energy is minimized

$$\begin{aligned} x_2 &= \arg\max (\mathbf{T}_1 x_1 + \mathbf{T}_2 x_2)^T \mathbf{K}^T \mathbf{K} (\mathbf{T}_1 x_1 + \mathbf{T}_2 x_2) \\ &= -(\mathbf{T}_2^T \mathbf{K}^T \mathbf{K} \mathbf{T}_2)^{-1} \mathbf{T}_2^T \mathbf{K}^T \mathbf{K} \mathbf{T}_1 x_1 \end{aligned} \tag{2.16}$$

where matrix $\mathbf{K}$ is a differential matrix corresponding to the first order derivative. We can also extend $\mathbf{K}$ to other higher order differential matrices.

The system can infer the occlusion relationship with the depth curve obtained from user specification and automatic interpolation. During the tracking, we use a 1st order motion model to predict the position of the contour in the next frame. For each point of the predicted contour, we run occlusion reasoning, i.e. to check whether the point is occluded by other contours with smaller depth value. We define a point to be occluded by a contour if the local weighting function remains less than 90% after running `inpolygon` function for the neighboring pixels. If the kth feature point is occluded, we set the corresponding weighting function $r_k(q)$ to be 0, so that the data term of this feature point is not counted in the tracking.

In Figure 2.5, the user can specify the right (top row) and wrong (bottom row) depth for the person. If the person is specified to be behind the car, equivalent to the person not being specified because a selected contour is assumed to be in front of other pixels, the tracking of the car gets affected by the person. If the person is specified to be in front of the car, then the tracking is correct even for the occluded feature points. Therefore, it is important to label objects in the order of their occlusion relationship: foreground first, background second.

■ 2.3.6 Layer-wise Optical Flow Estimation

The key difference between layer-wise optical flow estimation and traditional full-frame optical flow estimation is a mask indicating the visibility of each layer. Only the pixels falling into this mask are used for matching. Because the shape of the mask can be arbitrary and fractal, outlier detection is performed to excluded occlusions in the flow computation.

We use the optical flow algorithms in [22, 20] as the baseline model for optical flow estimation, while symmetric flow computation [4] is also incorporated to improve the accuracy. Let M_1 and M_2 be the visible mask of a layer at frame I_1 and I_2, (u_1, v_1) be the flow field from I_1 to I_2, and (u_2, v_2) the flow field from I_2 to I_1. The objective function for estimating layer-wise optical flow consists of the following terms. First, a data term is designed to match the two

images with the visible layer masks:

$$E^{(1)}_{\text{data}} = \int g * M_1(x,y)|I_1(x+u_1, y+v_1) - I_2(x,y)|, \tag{2.17}$$

where g is a Gaussian filter. The data term $E^{(2)}_{\text{data}}$ for (u_2, v_2) is defined similarly. Notice that $L1$ norm is used here to account for outliers in matching. Second, smoothness is imposed by

$$E^{(1)}_{\text{smooth}} = \int \left(|\nabla u_1|^2 + |\nabla v_1|^2\right)^{\eta}, \tag{2.18}$$

where η varies between 0.5 and 1. Lastly, symmetric matching is required:

$$E^{(1)}_{\text{sym}} = \int \begin{array}{l} \big|u_1(x,y) + u_2(x+u_1, y+v_1)\big| + \\ \big|v_1(x,y) + v_2(x+u_1, y+v_1)\big|. \end{array} \tag{2.19}$$

The objective function is the sum of the above three terms:

$$E(u_1, v_1, u_2, v_2) = \sum_{i=1}^{2} E^{(i)}_{\text{data}} + \alpha E^{(i)}_{\text{smooth}} + \beta E^{(i)}_{\text{symtr}}. \tag{2.20}$$

We use IRLS, equivalent to the outer and inner fixed-point iterations proposed in [20], combined with coarse-to-fine search and image warping to optimize this objective function. After the flow computation at each pyramid level, we update the visible layer mask M_1 based on the estimated flow:

- If $M_2(x+u_1, y+v_1) = 0$, then set $M_1(x,y) = 0$;
- If the symmetry term in Eqn. (2.19) is beyond a threshold at (x,y), then set $M_1(x,y) = 0$.

The same rule is also applied to update M_2. As the algorithm runs from coarse to fine, we obtain both bidirectional flow fields and trimmed visible layer masks that reflect occlusion.

We allow the user to adjust α, β and η in Eqn. (2.20) for each layer to handle different elasticities. Intuitively, a larger α, β or η results in a smoother flow field, but it is smooth in different ways. The user typically does not know which parameter setting produces the best flow. Therefore, our system allows the user to specify a list of different parameter settings for each layer, and the system computes the dense optical flow field for each parameter setting of each layer at each frame. This computation is done offline.

Although our layer-wise optical flow estimation works well in general, we observe failures where no parameter setting generates reasonable flow. The failures are mainly caused by the following factors.

- Specularity, shadow, noise and blurriness in real-life videos cause the brightness or boundary preservation assumption to break.
- Due to occlusions, the visible layer mask may be very small or irregularly shaped, making the global motion difficult to capture.

In such cases when optical flow estimation fails, we rely on semiautomatic motion labeling.

2.3.7 Semiautomatic Motion Labeling

When optical flow estimation fails, the user can specify sparse correspondence between two frames using feature points, and our system automatically generates a parametric motion or interpolates a dense optical flow field based on the sparse correspondence. The sparse correspondence can be specified either with a computer's assistance for efficiency, or manually, to give the user full control of motion annotation.

When the user specifies one feature point in one frame, the system automatically searches for the best match in the next frame using mininum-SSD matching and the Lucas-Kanade transform [71] for sub-pixel accuracy. Based on the number of specified feature points, the system automatically determines the mode of parametric motion−translation, affine transform or homography−and estimates the motion parameters accordingly [113]. The user can also select these modes, and even choose to produce a smooth flow field interpolated using the preconditioned conjugate gradient algorithm described in [112].

However, specifying corner-like feature points [105] can be difficult for some sequences when only line structures are present in the layer. To address this problem, we incorporate uncertainty matching and probabilistic parametric motion estimation so that the user can freely choose any pixel for correspondence. In uncertainty matching, the system generates a probability map $p_k(x)$ for the matching of feature point k at location $q_k \in \mathbb{R}^2$. This probability map $p_k(x)$ is approximated by a mean μ_k and covariance matrix Σ_k. In probabilistic motion estimation, the system iterates between the following two steps. In the first step, motion is estimated using the current estimate of mean and covariance. Mathematically, let $h(q_k;\theta)\colon \mathbb{R}^2 \mapsto \mathbb{R}^2$ be a parametric motion applied to q_k. The motion parameter is estimated by

$$\theta^* = \arg\min_{\theta} \sum_k \big(h(q_k;\theta) - \mu_k\big)^T \Sigma_k \big(h(q_k;\theta) - \mu_k\big). \tag{2.21}$$

In the second step, the mean and covariance are estimated using a new probability map reweighted by the current motion

$$\{\mu_k, \Sigma_k\} \leftarrow p_k(x)\mathcal{N}(h(q_k;\theta^*), \sigma^2\mathbf{I}). \tag{2.22}$$

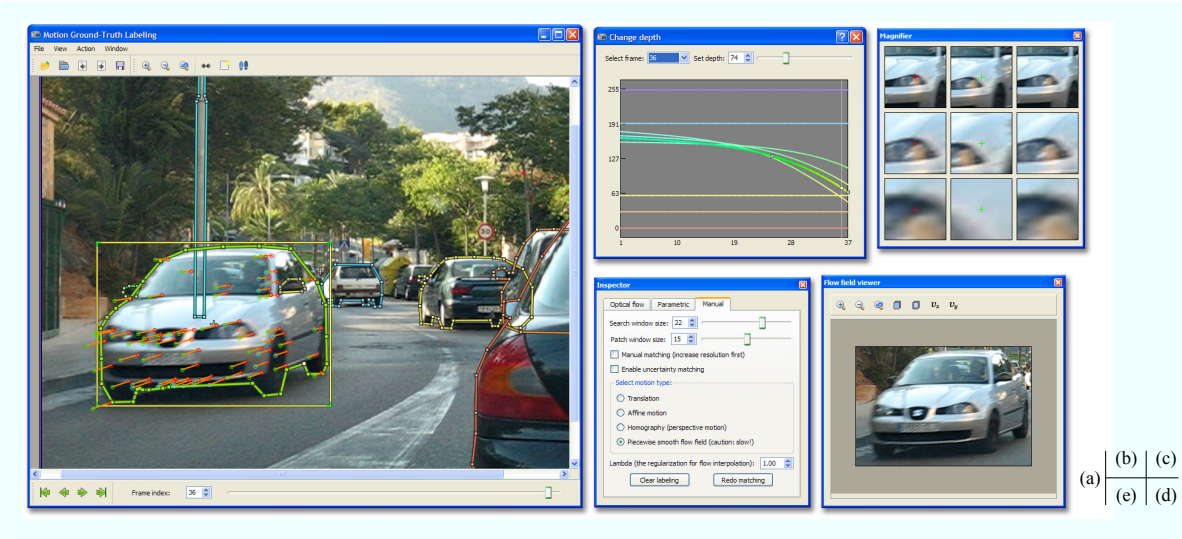


Figure 2.6. A screenshot of our motion annotation system. From (a) to (e) is the main window, depth controller, magnifier, flow field viewer and control panel.

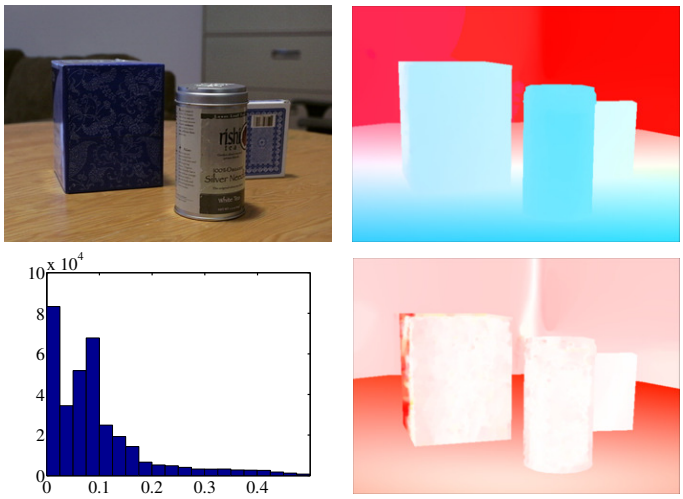


Figure 2.7. Clockwise from top left: the image frame, mean labeled motion, mean absolute error (red: high error, white: low error), and error histogram.

This algorithm converges within a few iterations. The motion $h(q_k;\theta)$ can also be a dense optical flow field (*i.e.* θ). In addition, the feature points that the user labeled can be used for the next frame. To perform the semiautomatic motion labeling, the user interacts with these tools through an interface, described next.

2.3.8 System Design and Human Judgement

A graphical user interface (GUI) is a key component of our system since intensive user interaction is needed for motion annotation. Our system is developed in C++ with Qt^{TM} 4.3 for the GUI. A screenshot of the GUI is shown in Figure 2.6. Our system allows the user to label contour and specify correspondence in the main window (a), change depth in a dialog window (b) and modify parameters in an inspector window (e). We also provide a magnifier to see image details (c) and a flow field viewer to check optical flow field and matching (d).

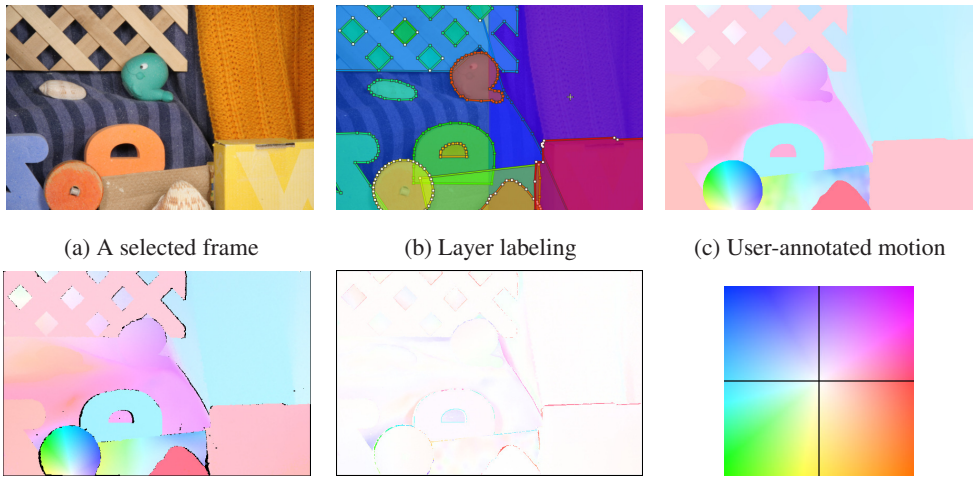
We found that two criteria must be satisfied for a computer-generated flow field, either from dense optical flow estimation or from sparse correspondence, to be accepted by a human as the right motion for a layer. First, the matching has to be correct. The flow field viewer, as shown in Figure 2.6 (d), is used to display the current frame and the warped next frame based on the flow field back and forth, to check whether this flow field produces the right match. Second, the smoothness and discontinuities of the flow field must match the rigidity and boundary of the object. This property can also be inspected in the flow field viewer.

The user interaction time depends on the number of frames, the number of layers and the shape of the objects. Typically, it takes 10 to 40 minutes to label five or six objects in a 30-frame sequence, depending on the shape complexity and the amount of occlusion. It takes several hours of computation time to compute the dense flow field for every flow parameter setting, every layer and every frame. It takes one minute or more to specify the sparse correspondence for a layer, and fitting parametric motion in realtime.

2.4 Methodology Evaluation

We conducted two experiments to examine our methodology. First, we applied our system to obtaining motion for the sequences that have been carefully annotated using other methods. We downloaded the ground-truth motion of a sequence RubberWhale from the webpage of [7]. We labeled the sequence using 20 layers, generated the layer-wise motion, and showed the results in Figure 2.8. Most of the disagreement lies along the occluding boundaries of the objects, as shown in Figure 2.8(e). The error between our annotation and their "ground-truth" flow is 3.21° in average angular error (AAE) and 0.104 in average endpoint error (AEP). For comparison, the best optical flow algorithm achieves merely 11.09° in AAE for a similar sequence Seashell [7]. Notice that the authors in [7] generated the flow on high-res images and then down-sampled the flow to produce the ground truth, whereas we directly worked on the low-res images.

Second, we tested the consistency of multiple users' labelings. We asked nine subjects to



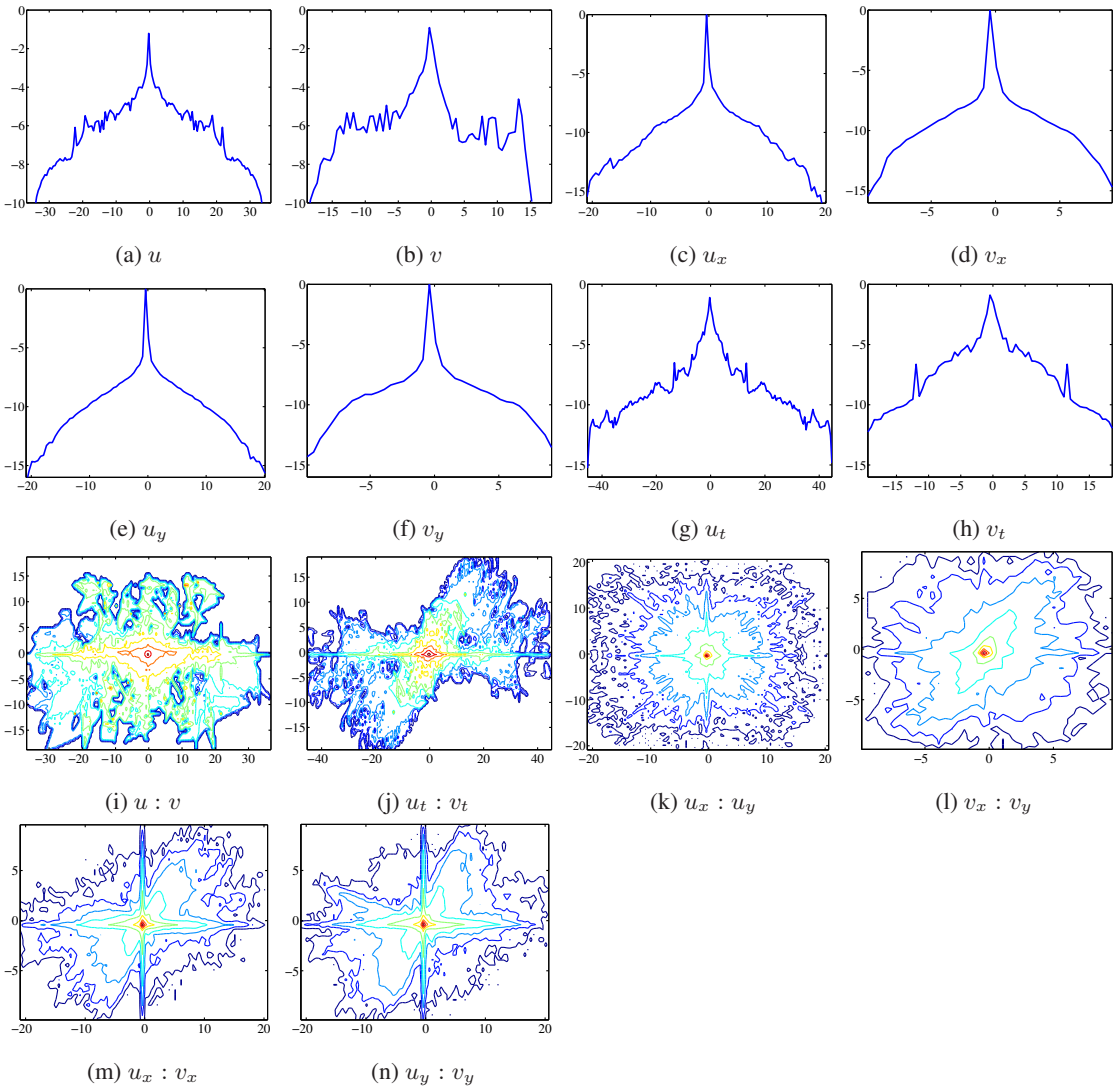
(a) A selected frame (b) Layer labeling (c) User-annotated motion

(d) Ground-truth from [7] (e) Difference (f) Color map

Figure 2.8. For the RubberWhale sequence in [7], we labeled 20 layers in (b) and obtained the annotated motion in (c). The "ground-truth" motion from [7] is shown in (d). The error between (c) and (d) is 3.21° in AAE and 0.104 in AEP, excluding the outliers (black dots) in (d). (e): The color encoding scheme for flow visualization [7].

use our tool to annotate motion for the sequence in Figure 2.10 (a). Since the subjects agreed with the layer segmentation of the sequence, we labeled the layers and generated ten flow fields for each layer by varying the smoothness parameters. The subjects were asked to choose the best flow field for each layer, and they could label the sparse correspondence if they were not satisfied with any of the flow fields. We found some subjects preferred smooth flow fields over good matching, while others preferred good matching over smoothness, but all the subjects tried to balance between smoothness and correct matching. The subjects were unanimously unsatisfied with the flow field of the table, and all of them labeled the motion of the table using feature point matching.

The per-pixel mean and standard deviation of these nine subjects' annotations are shown in Figure 2.7. The mean of the standard deviation is 0.0934 pixel. Most of the disagreement is at the table area, where some subjects labeled more than 10 points and some only labeled five. We also measured the error between each annotation and the mean flow field. The mean error is 0.989° in AAE and 0.112 in AEP. This AEP value is consistent with the previous experiment: the accuracy of human annotation is around 0.1 AEP.



(a) u (b) v (c) u_x (d) v_x

(e) u_y (f) v_y (g) u_t (h) v_t

(i) $u : v$ (j) $u_t : v_t$ (k) $u_x : u_y$ (l) $v_x : v_y$

(m) $u_x : v_x$ (n) $u_y : v_y$

Figure 2.9. The marginal ((a)∼(h)) and joint ((i)∼(n)) statistics of the ground-truth optical flow in our database (log histogram).

■ 2.5 A Human-Annotated Motion Ground-Truth Database

We collected video sequences of both indoor and outdoor scenes using a Canon EOS-1D (low frame rate, medium resolution) and a Canon SD870 (high frame rate, middle resolution), and carefully labeled the motion using our annotation tool. Some sequences captured by the Canon EOS-1D are shown in Figure 2.10 (a) to (d). We observe noisy and blurry backgrounds in (a)

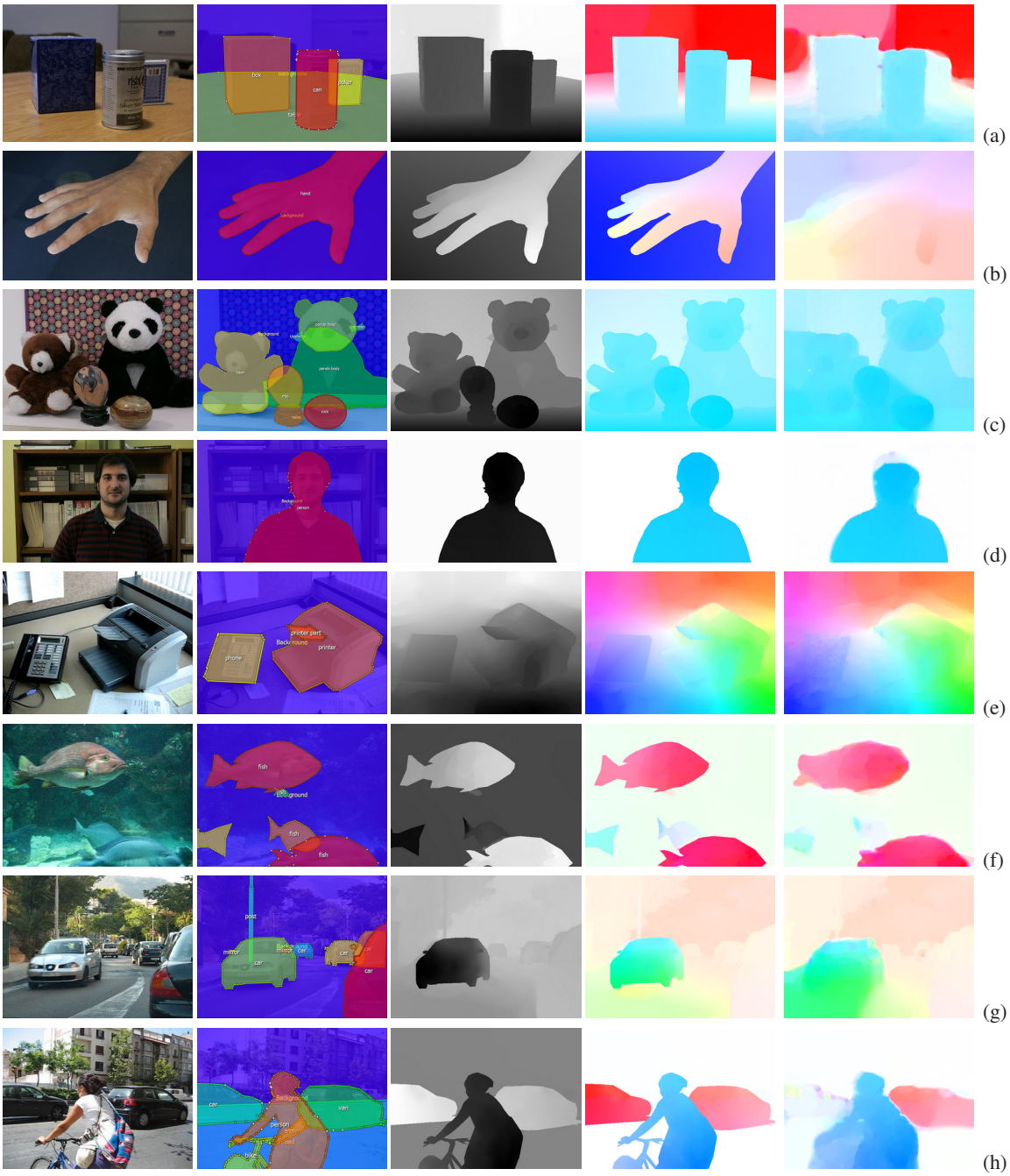


Figure 2.10. Some frames of the ground-truth motion database we created. We obtained ground-truth flow fields that are consistent with object boundaries, as shown in column (3), the horizontal component of flow, and column (4), flow colorization using Figure 2.8 (f). In comparison, the output of an optical flow algorithm [22] is shown in column (5). The error between the ground-truth motion (4) and flow estimation (5) is as follows (AAE, AEP), (a): 8.996°, 0.976; (b): 58.904°, 4.181; (c): 2.573°, 0.456; (d): 5.313°, 0.346; (e) 1.924°, 0.085; (f): 5.689°, 0.196; (g): 5.2431°, 0.3853; and (h): 13.306°, 1.567. Most of the errors are significantly larger than the errors with the Yosemite sequence (AAE 1.723°, AEP0.071) . The parameter of the flow algorithm in column (5) is tuned to generate the best result for each sequence.

and (b) because of the shallow depth of field. Some of the sequences captured by the Canon SD870 are displayed in Figure 2.10 (e) to (h). A typical frame of the selected sequence is shown in column (1), the corresponding layer labeling in column (2) and horizontal motion in column (3). To compare the annotated flow field and the result of a state-of-the-art flow algorithm [22], we used the colorization scheme in [7] (Figure 2.8(f)) to visualize these two flow fields in column (4) and (5).

Our human-annotated motion is significantly better than what the flow algorithm could achieve. Based on the criteria of the visual inspection of motion, the discontinuities of the annotated flow fields align well with the object boundaries, and the smoothness reflects the rigidity of the objects. In comparison, the flow fields computed by the flow algorithm often fail to capture the object boundary and the correct smoothness. The flow computation can have large errors for blurry and noisy regions, such as sequence (a) and (b) where the background motion is affected by the foreground in flow computation. In (c), (d), (f), (g) and (h), ambiguous boundary ownership causes the flow to mis-propagate across occluding boundaries. The mean AAE and AEP errors of each sequence are listed in the caption of Figure 2.10. Most of these errors are significantly larger than those for the Yosemite sequence. This suggests that our motion ground-truth database is more challenging for motion estimation algorithms and can be useful for developing better algorithms.

Now that we have sufficient realistic, ground-truth motion data, as a side effect, we can learn the statistics of realistic motion fields. These statistics can lead to more accurate prior of flow fields and help to improve flow estimation algorithms [87]. We computed the marginal and joint statistics of the ground-truth flow in our database and displayed the log histograms in Figure 2.9. In (a) and (b), the marginal of u (horizontal flow) is flatter than that of v (vertical flow), indicating that horizontal motion dominates vertical. As shown in (b) and (i), the marginal of v is asymmetric, and there are more pixels falling down than going up (due to gravity). The marginals of the 1st-order derivatives of the flows are sparse, as shown in (c) to (f). Unlike the marginals of synthetic flow fields [87], our statistics show that the vertical flow is sparser than the horizontal flow, consistent with the fact that horizontal motion has a larger range. The temporal derivatives of the flow are not as sparse as the spatial ones, as depicted in (g) and (h). The joint histogram in (j) suggests that horizontal and vertical motion are likely to increase or decrease together temporally. The joint histograms in (k) and (l) reveal that the discontinuities of the flow are isotropic. At motion discontinuities, the change of vertical motion may dominate the change of horizontal motion, and vice versa, as shown in (m) and (n).

2.6 Conclusion

Motion analysis algorithms have been in use for decades, but little has been done to obtain the ground-truth motion for real-world videos. We presented a methodology and system to obtain ground-truth motion through human annotation. Our system is built upon several state-of-the-art motion analysis algorithms that allow the annotation of every pixel and every frame. A special graphical user interface is designed to allow the user to label layers, inspect motions, select parameters, and specify sparse correspondences. Our methodology is validated by comparison with the ground-truth motion obtained through other means and by measuring the consistency of human annotation. Using our system, we collected a motion ground-truth database consisting of challenging real-world videos for algorithm evaluation and benchmarking. We hope this database and our annotation code will lead to improved algorithms for optical flow and layered motion analysis.

Chapter 3

Layer and Contour Representations for Motion Analysis

In Chapter 2 we introduced a human-assisted motion annotation system to obtain layer representation for a video sequence. Even though with human labor this system is able to achieve high-precision layer segmentation and motion measurement, the system is not fully automatic. We want to explore automatic layer analysis from video sequences. Moreover, layers are not the only middle-representation for video sequences. In this chapter, we will also discuss alternative representations such as contours.

Analyzing forms such as layers and contours from video sequences leads to many applications that cannot be achieved through mere pixel-level analysis. In particular, we are interested in detecting and magnifying small motions, as well as analyzing motion for textureless objects under occlusion.

■ 3.1 Motion Magnification

Visual motion can occur at different amplitudes, and over different temporal and spatial frequency scales. Small motions are difficult to observe, yet may reveal important information about the world: small deformations of structures, minute adjustments in an equilibrium process, or the small movements of a system in response to some forcing function. We want a machine which will reveal and clarify those motions, much as a microscope can reveal small and invisible structures.

We have developed a technique, called Motion Magnification, which acts like a microscope for motion in video sequences. The algorithm analyzes the motions of an input video sequence, allowing a user to specify a cluster of pixels to be affected, and how much their motions are to be magnified. Typically, small motions are amplified and large motions are left unchanged.

The final stage of motion magnification is to render the sequence with the desired motions magnified by the specified amount.

While motion magnification is conceptually simple, performing it without objectionable artifacts requires considerable attention to detail. We introduce techniques to analyze motion robustly, verifying estimated motions as well as their regions of support. The selection of motions to be amplified is made simple and intuitive by an automatic grouping process, where a prespecified number of pixel clusters are found, based on their similarity in position, intensity, and motion characteristics. For this, we introduce a measure of affinity that groups points based on their trajectories over time, not just the similarities of their instantaneous velocities. The user specifies which cluster's motions should be amplified and by how much. Holes revealed by amplified motions are filled using texture synthesis methods.

We demonstrate motion magnification with several proof-of-concept examples, magnifying small-amplitude motions in videos of structures or people. We envision potential applications ranging from engineering diagnosis or instruction to comedic amplification of ordinary expressions.

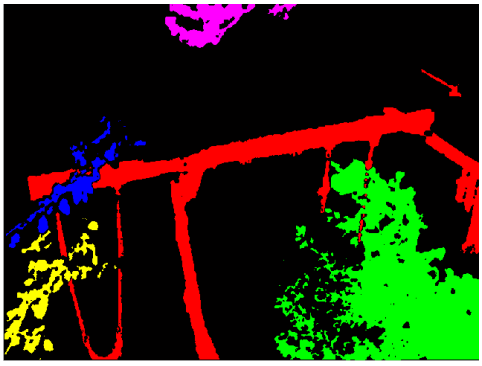
■ 3.1.1 Related Work

Motion magnification analyzes and redisplays a motion signal, and thus relates to research on manipulating and redisplaying motion capture data, such as modifications to create new actions from others [5, 58, 84, 60], and methods to alter style [17, 42, 119]. Of course, our problem is in a different domain; we manipulate video data, not marker positions, and thus have a more difficult analysis task, but also have the richer synthesis possibilities of video.

Several other projects have used video motion analysis in order to synthesize an output. Brostow and Essa [19] tracked frame-to-frame motion of objects, then integrated the scene's appearance as it changed over a synthetic shutter time to simulate motion blur. Video textures [100] analyzes the overall similarity of all frames to identify candidate temporal jumps in a modified playback of the video. Several researchers have used a layered motion analysis to synthesize a modified video [123, 53], where the modification typically involves removing individual layers. However, many of the techniques used to group pixels of similar motions into layers would not work for our difficult case of analyzing very small motions. While the above projects relate at the general level of motion analysis followed by synthesis, we are not aware of any previous work addressing motion magnification.

(a) Registered input frame

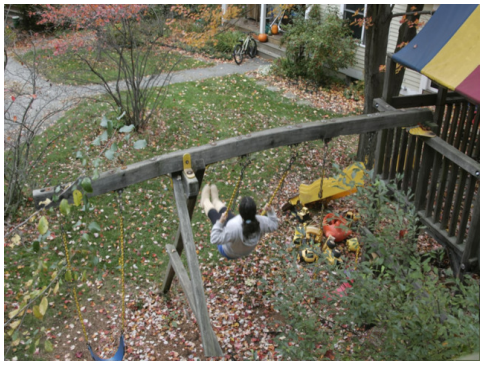
(b) Clustered trajectories of tracked features



(c) Layers of related motion and appearance

(d) Motion magnified, showing holes

(e) After texture in-painting to fill holes



(f) After user's modifications to segmentation map in (c)

Figure 3.1. Summary of motion magnification processing steps.

■ 3.1.2 Overview

We want to find small motions in a video and magnify them. We model the appearance of the input video as trajectories (translations) of the pixel intensities observed in a reference frame. Naïvely, this sounds like one simply needs to (a) compute the translation from one pixel to the

next in each frame, and (b) re-render the video with small motions amplified. Unfortunately, such a naïve approach would lead to artifactual transitions between amplified and unamplified pixels within a single structure. Most of the steps of motion magnification relate to reliably estimating motions, and to clustering pixels whose motions should be magnified as a group. While we ultimately estimate a motion at every pixel, we begin by analyzing and grouping the motions of feature points, local intensity configurations that are promising candidates for finding reliable motion trajectories. Below we motivate and summarize each step of the motion magnification processing. The processing steps are illustrated with the swing set images in Fig. 3.1.

Register input images

When we magnify small motions, it is essential to begin by registering the frames, to prevent amplifying the inevitable small motions due to camera shake. For this step, we assume that the input image sequence depicts a predominantly static scene. We perform an initial tracking of detected feature points and find the affine warp which best removes the motions of the set of tracked feature points, ignoring outliers. After intensity normalization for any exposure variations, the resulting registered images are ready for motion analysis.

Cluster feature point trajectories

In order that the motion magnification not break apart coherent objects, we seek to group objects that move with correlated (not necessarily identical) motions. To achieve this, we robustly track feature points throughout the sequence, then cluster their trajectories into K sets of correlated motions. One special cluster of feature points with no translation over frames is the background cluster. An important contribution of this work is the computation of trajectory correlation in a manner invariant to the overall scale of the motions, thereby allowing very small motions to be grouped with larger motions to which they are correlated. For example, the left extremity of the beam in Fig. 3.1 has larger motion magnitude than the points attached to the vertical beam, and some points along the beam move in opposite phase, yet, they should all be assigned to the same motion layer because they belong to a "common cause". The motions are all specified as translations from the feature point position in a reference frame.

Segmentation: layer assignment

From the clustered feature point trajectories, we want to derive motion trajectories for each pixel of the reference frame. We interpolate a dense motion field for each motion cluster, giving us

K possible motion vectors at each pixel. We need to assign each pixel of every frame to one of the clusters or motion layers.

It is possible, in principle, to perform segmentation using motion alone [123, 53], but reliable segmentation requires the use of additional features. We use pixel color, position, as well as motion to estimate the cluster assignment for each pixel, defining a Markov random field which we solve using graph cuts [16]. To impose temporal consistency, we then assign each pixel trajectory to its most commonly assigned cluster over all time frames.

This gives us a layered motion representation such as that proposed by Wang and Adelson [123], but generalizing layer membership to include correlated motions, and not just similar ones. Our model of the video is a set of temporally constant pixel intensities, clustered into layers, which translate over the video sequence according to interpolated trajectories that are unique to each pixel. The layer ordering can be specified by the user, or computed using the methods of Brostow and Essa [18]. In practice, it is sufficient to randomly assign the ordering of non-background layers if the magnified layer has minimal occlusions with other layers, as is often the case. At each stage, pixels which do not fit the model are relegated to a special "outlier layer". The other layer that is treated specially is the background layer. Regions of the background layer which were never seen in the original video sequence may be made visible by amplified motions of motion layers above the background. We thus fill-in all holes in the background layer by the texture synthesis method of Efros and Leung [33].

Magnify motions of selected cluster

After the layers are determined, the user specifies a layer for motion magnification. Presently, the magnification consists of amplifying all translations from the reference position by a constant factor, typically between 4 and 40, but more general motion modification functions are possible.

Render video

Following motion magnification, we render the modified video sequence. The background layer is constant for all frames and we render its pixels first. Then the pixels assigned to the outlier layer are copied as they appeared in the registered input frames. Finally, the pixels of the remaining layers are written into the output sequence. The intensities are those of the reference frame; the displacements are those of the measured or magnified motions, as appropriate to the layer. In the following sections, we describe each processing step of motion magnification in detail.

■ 3.1.3 Robust Video Registration

Since we magnify small motions, we need to be very careful that stationary pixels are not classified as moving. Because of inevitable small camera motions, even with a tripod, almost all pixels are moving in the input sequences. To address this problem, we devised a fully automatic system to align the input images.

The main idea comes from recent work [94]. Instead of registering images frame to frame, our algorithm finds a reliable set of feature points that are classified as "still". Then an affine motion is estimated from the matched feature points. All frames are registered to a reference frame, typically the first frame in our system.

We detect corners at different scales in the reference frame using a hierarchical version of Harris corner detector [48], with a modification from page 45 of Nobel's thesis [80]. Then we compute a flow vector for each feature point from the reference frame to each of the other frames based on the minimum sum of squared differences (SSD) over a small patch. The precision of the flow vector is further refined to sub-pixel level based on a local Lucas-Kanade algorithm [71, 105].

Before describing the affine motion estimation, we introduce some notation conventions of the chapter. We measure N feature points over K frames. The n^{th} ($n = 1 \cdots N$) feature point in frame k ($k = 1 \cdots K$) is denoted as (n, k). The coordinate of feature point (n, k) is (x_{nk}, y_{nk}). Likewise, the flow vector from the reference frame is denoted as (v^x_{nk}, v^y_{nk}). For similarity measurements, we consider a window or patch B_{nk} of size $2w \times 2w$ around each feature point (n, k). $B_{nk}(p, q)$ is the pixel at relative coordinates (p, q) from the centroid of patch B_{nk}. Note that when we use sub-pixel coordinate (x_{nk}, y_{nk}), we interpolate the patch using bicubic reconstruction.

A global affine motion $A_k \in \mathbb{R}^{2\times 3}$ is estimated from the reference frame to frame k with a weight depending on the quality of the local appearance match. The probability that a feature point (n, k) participates in this affine motion is estimated as

$$\Pr_{nk} = \exp\{-\|A_k[x_{nk}\ y_{nk}\ 1]^T - [v^x_{nk}\ v^y_{nk}]^T\|^2/(2\sigma_k^2)\} \tag{3.1}$$

where variance σ_k is estimated as the mean reconstruction error $\sigma_k = \frac{1}{n}\sum_n \|A_k[x_{nk}\ y_{nk}\ 1]^T - [v^x_{nk}\ v^y_{nk}]^T\|^2$. We treat the ego motion of the camera as random noise, and therefore the probability for feature point n contributing to the global affine motion over all frames is the product of the probability for each frame: $\Pr_n = \prod_k \Pr_{nk}$.

Finally, the stable feature points are selected by their probability relative to that of the most

probable feature point:

$$\Pr_n > \alpha^K \cdot \max_i \Pr_i. \tag{3.2}$$

We find $\alpha = 0.3$ works well for all the sequences we have tested. By this procedure, unstable feature points, such as those on occluding boundaries, in disoccluded regions and at rapidly moving objects, are discarded. Only the feature points that consistently contribute to a global affine motion across all the frames are selected for registration.

When the stable feature points are selected, we redo the SSD matching and local Lucas-Kanade refinement from the reference frame to each of the rest of the frames. The rest of the frames are all registered to the reference frame from a global affine warp $\tilde{A}_k$ estimated from the matching upon this stable feature point set. In this step we also perform histogram equalization for each frame to the reference frame to remove illumination or exposure changes.

■ 3.1.4 Robust Computation and Clustering of Feature Point Trajectories

In the previous section, we computed feature trajectories for the static background in order to stabilize the sequence. We now turn to the computation of trajectories for feature points over the whole image and to their clustering into motions that are correlated.

Variable region feature point tracking

Once the images have been registered, we find and track feature points for a second time. The goal of this feature tracking is to find the trajectories of a reliable set of feature points to represent the motions in the video. As before, the steps consist of feature point detection, SSD matching and local Lucas-Kanade refinement. For simplicity, we use only those features detected in the first frame.

To achieve reliable feature point matching near occlusion boundaries, [97] displaced the rectangular region of support away from the boundary edge. Here we introduce a method to find regions of support of general shape, tailored to each feature point. We compute a weight or support map for each feature patch that characterizes how pixels should be weighted in the SSD similarity measures. For features near occlusion boundaries, this increases reliability by only comparing pixel intensities with others on the same side of the boundary. This map, shown in Fig. 3.2, also lets us assess the validity of each feature trajectory, useful in both the SSD matching and Lucas-Kanade refinement. We call this method *variable region feature point tracking*.

We use an Expectation Maximization (EM) algorithm [32] to learn the weight map asso-

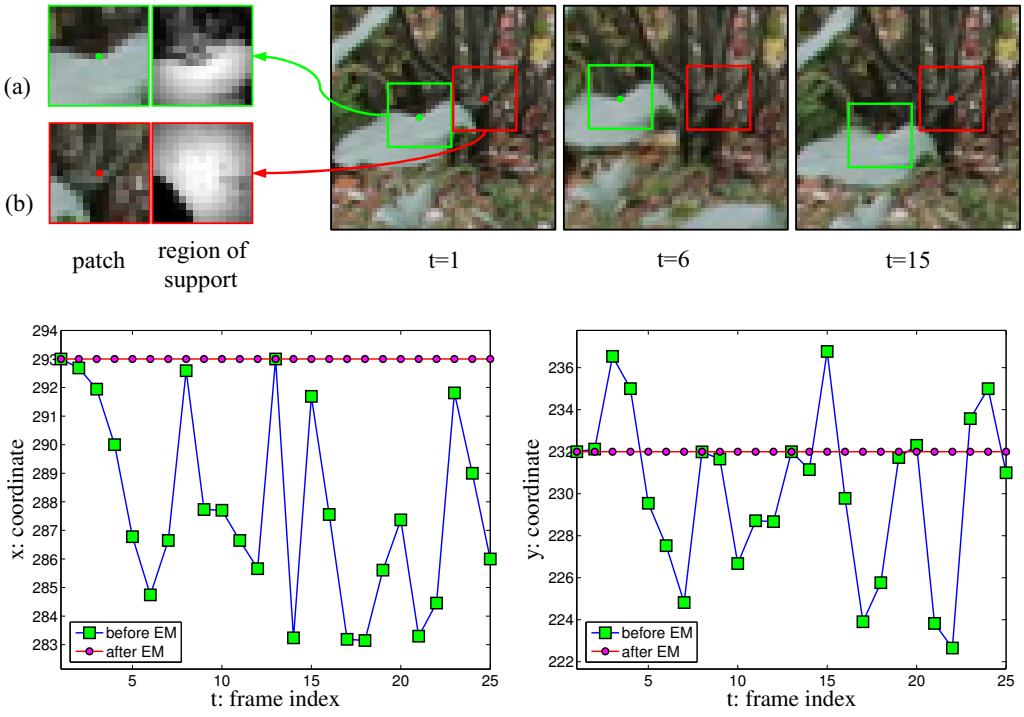


Figure 3.2. Learned regions of support allow features (a) and (b) to reliably track the leaf and background, respectively, despite partial occlusions. For feature (b) on the stationary background, the plots show the x (left) and y (right) coordinates of the track both with (red) and without (blue) a learned region of support for appearance comparisons. The track using a learned region of support is constant, as desired for feature point on the stationary background.

ciated with each feature point. The EM algorithm alternates between an "E-step", when the weight map (region of support) is estimated for an assumed feature translation, and an "M-step", when the feature translation is updated, using the weight map just estimated.

E-step

We first estimate the probability that each feature point trajectory is reliable. We call reliable trajectories inliers and unreliable ones outliers. Two conditions must be met for the inliers: (1) the SSD matching error at each position of the feature point (compared with the appearance in the reference frame) must be small, and (2) there must be nearby feature point positions from other times along the trajectory. To compute the second term, we evaluate the mean distance to the N nearest feature points in the same trajectory. The tracking inlier probability of feature n

at frame k ($k > 1$) is computed as

$$\Pr_{nk} = \exp\{-\frac{\mathrm{SSD}_{nk}}{2 \min\limits_{1<i\leqslant K} \mathrm{SSD}_{ni}} - \frac{d_{nk}}{2 \min\limits_{1\leqslant i\leqslant K} d_{ni})}\}, \tag{3.3}$$

where SSD_{nk} is the SSD of feature n at frame k, and d_{nk} is the mean distance of feature n at frame k to the N-nearest feature points in the same trajectory.

The weight map is estimated from the average reconstruction error for the pixel (p, q) ($-w \leqslant p, q \leqslant w$) relative to the feature point. We use a patch size $w = 7$ for all the examples. The weight is therefore computed as

$$\Phi_n(p,q) = \exp\{-\frac{p^2+q^2}{2s^2} - \frac{\sum_{k=2}^{K} \|B_{nk}(p,q) - B_{n,1}(p,q)\|^2 \Pr_{nk}}{2\sigma_n^2 \sum_{k=2}^{K} \Pr_{nk}}\} \tag{3.4}$$

where $s = w/2$ and σ_n^2 is the mean variance over frames k's and positions (p, q). Intuitively, the neighboring pixels that are close to the feature point and have less variation in matching should have high weight.

M-step

The M-step is the same as the previous SSD matching and local Lucas-Kanade refinement except that the weight map Φ_n is used. The use of this map, indicating the region of support for each feature, results in more reliable feature point tracking, illustrated in Fig. 3.2.

Experimentally, we found that after 10 iterations of EM most of the feature points converge to a reliable estimate of the weight map as well as the local flow vector. However, some feature points do not yield valid trajectories and must be pruned. We use a number of criteria to prune feature points:

- **Minimum matching error** Some feature points may appear in the reference frame but not in others. For such cases, the minimum matching error remains high. We remove these spurious trajectories by setting an upper bound on the minimum matching error; this threshold is set so that feature points which appear only in the first frame are removed.

- **Inlier probability** Some feature points reappear, but seldom, indicating an unreliable trajectory. We use the average inlier probability $\frac{1}{K}\sum_k \Pr_{nk}$ as a metric for trajectory n. A trajectory is removed if the average inlier probability is below a threshold, set to be 30% of the mean inlier probability for the trajectory.

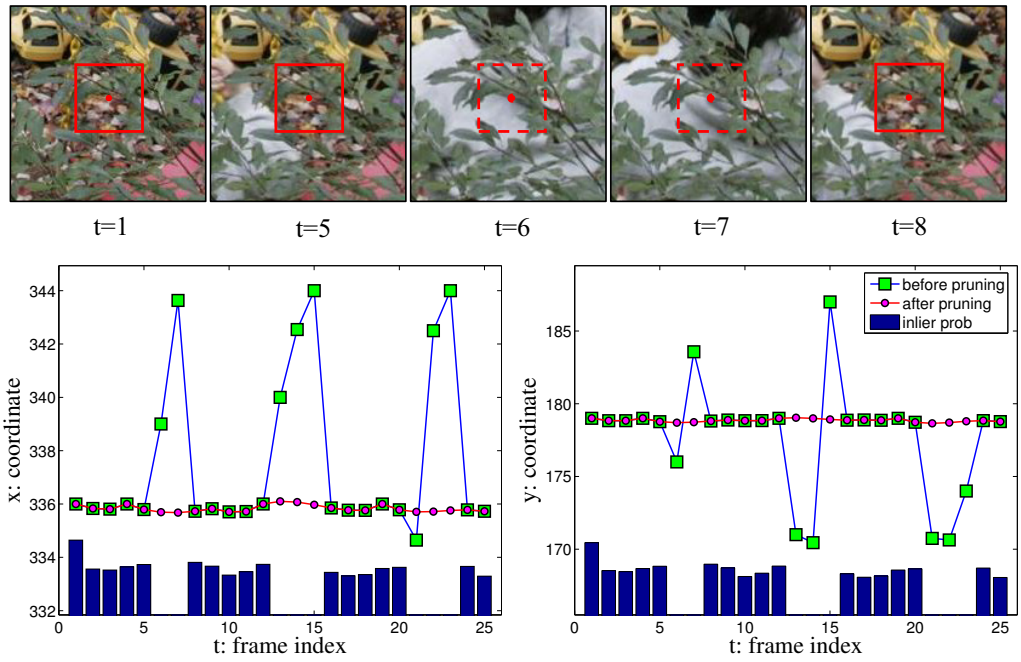


Figure 3.3. Top images show a feature point on the stationary background layer becoming occluded during frames 6 and 7. Below are the x- and y- coordinates of the tracked feature, showing outlier positions. These can be identified from the inlier probabilities shown as a bar plot (repeated for comparison with each plot) and replaced with smoothed values.

- **Matching outlier detection, removal and filling** Finally, we must smooth some of the remaining inlier feature point trajectories because of occlusions at some frames, which generates impulsive noise in the trajectory and a corresponding increase in the matching error, as shown in Fig. 3.3. These events are detected from the inlier probability at each time frame and removed. We fill in the missing feature point positions by minimizing the second derivative energy of the trajectory over time, by summing the squared responses from filtering with $[-1\ 2\ -1]$. The resulting least squares problem is solved by a standard conjugate gradient method, [108].

The first two criteria are used both before and after the EM algorithm, and the third criterion is applied only after EM. The output of the feature point tracking is a set of feature points, their regions of support, reliabilities, and trajectories over time. These trajectories are output to the next module for clustering.

Clustering by coherent motion

We seek to group related feature point trajectories into clusters, which will form the basis for our assignment of pixels to motion layers. Using motion, we can group regions of varying appearance or without spatial continuity (due to occlusions). Our goal in the clustering is that motions with a common cause be grouped together, even though the motions may be in different directions.

To do this, we use the entire motion trajectory of each feature point, not just instantaneous motions. Motions caused by the same physical source tend to covary and have common modes of resonance [131]. We introduce the use of *normalized correlation* between the trajectories as a measure of their similarity. Composing the x and y components of the velocities into a complex motion vector, the correlation index $\rho_{n,m}$ between the trajectories of two feature points n and m is:

$$\rho_{n,m} = \left| \frac{\sum_k \left(v_{nk}^x + j v_{nk}^y\right)\left(v_{mk}^x + j v_{mk}^y\right)}{\sqrt{\left(\sum_k (v_{nk}^x)^2 + (v_{nk}^y)^2\right)\left(\sum_k (v_{mk}^x)^2 + (v_{mk}^y)^2\right)}} \right| \tag{3.5}$$

with $j = \sqrt{-1}$.

The normalized correlation between complex velocities is invariant to both the direction of the motion trajectories, and their magnitudes. In the swingset example, the motions of the beam and of the chain attached to the beam show a high correlation index even though they have very different magnitudes and are moving in different directions (because of the normalization and absolute value operations in Eq. (3.5)). The normalized correlation is close to zero for feature points that belong to objects with independent motions. For instance, the trajectories of points belonging to the swing structure and the blowing leaves have very small correlation when evaluated over 25 frames.

Using the normalized correlation, we construct a similarity matrix between all feature tracks, then use spectral clustering [104] to group them into K clusters, a number selected by the user to give physically reasonable groupings. Before clustering, feature points having fewer than 5 neighbors with $\rho > 0.9$ are removed and considered as outliers. Fig. 3.1(c) shows the clustering result for the swing sequence using 6 clusters.

Dense optic flow field interpolation

From each group of feature tracks, we seek to interpolate a dense optic flow field over all pixels; we will then assign each pixel to one motion group to form a layered motion representation.

Because the tracked objects can be non-rigid, we interpolated using locally weighted linear regression to estimate an affine motion for the query pixel, following the approach of [94]. To speed the computation, we apply this interpolation on a sparse lattice of every fourth pixel. Then a bicubic interpolation is applied to obtain the dense flow field for all pixels. This two-step approach reduced the complexity by one order of magnitude with little cost in precision. We denote $M_{ik}^{(l)}$ the dense flow field from frame i to frame k for layer l.

■ 3.1.5 Segmentation: Assignment of Each Pixel to a Motion Cluster

We seek to assign every pixel to one of the motion clusters (layers). We do so using three cues: motion likelihood, color likelihood, and spatial connectivity. In the subsections that follow, we construct grouping probabilities for each cue to form a probability for a given layer assignment that we optimize by graph cuts. Finally, we impose a temporal coherence constraint to add poorly fitting pixels to the outlier layer.

Motion likelihood

The likelihood for pixel $I_k(x, y)$ at frame k to be generated by layer l is computed from reconstruction error

$$\Pr_M(I_k(x,y)|l) = \exp\{-\sum_{i=k-u}^{k+u} \frac{\|I_k(x,y) - I_i(M_{ik}^{(l)}(x,y))\|^2}{2\sigma_M^2}\}. \tag{3.6}$$

Where u is the number of neighboring frames and σ_M^2 is the variance. Often, motion segmentations are based on two sequential frames, but we find that a large u, such as 10, makes motion likelihood very reliable. We can compute this since we keep the trajectories of each feature point. We assign pixels of low likelihood to the outlier layer.

Color likelihood

Color information has been widely used in interactive image editing, such as [88, 92]. We also use color to help propagate the motion cue to ambiguous (flat) regions. Similar to [88], we use a Gaussian mixture model to compute the likelihood for each pixel generated by layer l

$$\Pr_C(I_k(x,y)|l) = \sum_{i=1}^{N_C} \alpha_i^{(l)} G(I_k(x,y); \mu_i^{(l)}, \sigma_i^{(l)}) \tag{3.7}$$

where $\{\alpha_i^{(l)}, \mu_i^{(l)}, \sigma_i^{(l)}\}$ are estimated from a previous layer assignment, and N_C is the number of mixtures (this term is only used after a first iteration of segmentation).

Spatial connectivity

We use a compatibility function to encourage layer assignment changes at spatial discontinuities of pixel intensity. We choose the following compatibility function, which is widely used [16, 88, 128]

$$V(I_k(x,y), I_k(x+p,y+q), l_1, l_2) = (p^2+q^2)^{-\frac{1}{2}}\delta[l_1 \neq l_2] \cdot \exp\{-\beta\|I_k(x,y) - I_k(x+p,y+q)\|^2\} \tag{3.8}$$

where $-1 \leqslant p, q \leqslant 1$, or $I_k(x,y)$ and $I_k(x+p,y+q)$ are neighboring pixels. l_1 and l_2 are the label assignment to the two pixels, respectively. Parameter β is estimated as described in Rother et al. [88].

Segmentation by energy minimization

Once we have set up and learned the parameters for each of the models, we use graph cuts [16] to minimize the total energy defined on the label assignment:

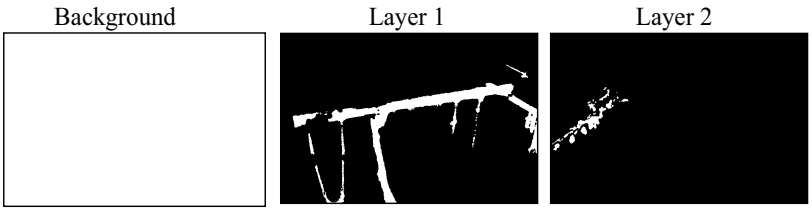
$$\begin{aligned} L^* = \arg\min_L & -\sum_{(x,y)} \log \Pr_M(I_k(x,y)|L(x,y)) \\ & -\xi \sum_{(x,y)} \log \Pr_C(I_k(x,y)|L(x,y)) \\ & +\gamma \sum_{(x,y)} \sum_{(p,q)\in N_{(x,y)}} V(I_k(x,y), I_k(x+p,y+q), L(x,y), L(x+p,y+q)) \end{aligned} \tag{3.9}$$

We follow Rother et al. [88] to set $\gamma = 50$ and $\xi = 2$, which works well for our test examples.

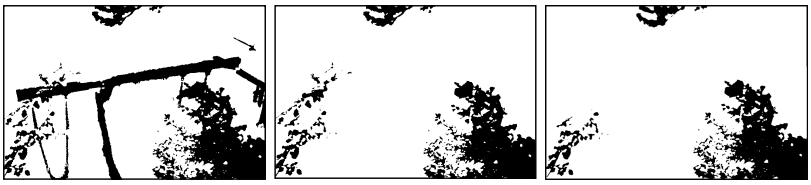
In each iteration of the graph cut algorithm the color distribution for each layer is re-estimated. The energy function drops fastest in the first three iterations, so we applied graph cuts three times to find the layer assignment for each frame.

Final layered representation of the sequence

The energy minimization segmentation is carried out for every frame independently, which inevitably introduces changes in layer assignment from frame to frame. To build our final representation of the sequence, we project each pixel back to a reference frame using the estimated motions. Each reference frame location which projects to a motion trajectory with 80% consistent layer assignments over all frames is assigned to that layer, Fig. 3.4(a). (Note that reference frame pixels may be assigned to more than one motion layer). The pixel intensity for a layer at each position is set to the median of the pixel intensities assigned to that layer along the



(a) Locations assigned to each layer



(b) Pixels occluded (shown in black) during the whole sequence for each layer

(c) Appearance of each layer before texture filling-in

(d) Appearance after texture filling-in

(e) Appearance after user intervention

Figure 3.4. Layered representation for the swing sequence. Only the background and two layers (out of six) are shown.

(a) (b) (c)

Figure 3.5. (a) Outlier locations, not well described by our model. Pixel intensities from these locations are passed through from the registered video into the rendered model (b) to yield the final composite output sequence, (c). The user specifies at which depth layer the outliers belong, in this case, above the background and below the other layers.

trajectory, Fig. 3.4(c). Since motion magnification will reveal occluded regions, we mark with occlusion masks regions where texture in-painting [33] needs to be applied, Fig. 3.4(d).

Finally, we need to account for the outliers. In some sequences, outliers might correspond to important elements for which the algorithm failed to build a model. In the case of the swing sequence, the person on the swing is not tracked, due to the fast motion, and is considered as an outlier. Fig. 3.5(b) shows one frame of the sequence rendered without including outliers. Fig. 3.5(c) shows the final result when the registered input pixels from the outlier locations, Fig. 3.5(a), are composited into the rendered frame (above the background and below the other layers). The outlier region is the union of the outliers computed for all the frames, as described in section 3.1.5.

In summary, the final representation of the sequence consists in a set of N_l layers plus one outlier layer (not always required). Each layer is defined by a segmentation mask (Fig. 3.4(a)), and its appearance (Fig. 3.4(d)).

User interaction

While the automatic segmentation results are very good, the bottom-up analysis inevitably makes small errors that can lead to artifacts in the synthesized video. To address this, we allow the user to modify the layer segmentation on the reference frame, as shown in Fig. 3.4(d). In this example, user modifications to the segmentation and appearance maps for layer 1 removed some pixels attached to that layer in the canvas awning, completed holes in the beam, and marked the swing back leg, Fig. 3.4(e). Only maps in the reference frame need to be edited.

■ 3.1.6 Magnification and Rendering

The user selects the motion layer for which the motion is to be magnified, and the displacements of each pixel in the cluster are multiplied by the selected factor. In our experiments the magnification factor was in the range between 4 and 40.

The depth ordering for each layer and the outliers can be assigned manually, or, for the non-outliers, computed through occlusion reasoning [18]. We render the pixels of each motion layer from back to front.

■ 3.1.7 Experimental Results

The accompanying video shows motion magnification results for three video sequences, *handstand*, *bookshelf*, and *swingset*. The first two sequences were taken with a JVC digital HD video camera JY-HD10, which can capture images at 1086×720 resolution, 30 Hz, progressive scan.

The last sequence was acquired at 8 frames per second with a Canon EOS 1D Mark II, which records images at 3500×2200 resolution. We downsample the input images for processing to 866×574 for the JVC and 1750×1100 for Canon. The precise motion computations and grouping operations of the motion magnification algorithm take 10 hours, end-to-end processing, for the swingset sequence, in a combination of C++ and Matlab code.

Handstand shows a single figure with visually perceptible balancing motions to maintain vertical posture. The magnified sequence amplifies the left-to-right corrective motion of the torso. In order to reveal small body motions without too much distortion of the human figure, we applied a saturating non-linear amplification to the motions of the magnified layer . The magnification was close to linear for amplified displacements below 40 pixels, with a compressive saturation for amplified displacements above that. Fig. 3.6 shows a frame from the original and motion magnified sequences. In this sequence we used a model with two layers and no outlier layer, and made no manual edits.

Bookshelf magnifies the response of a thick plywood bookshelf on aluminium supports to pressure from a hand. The original response is barely perceptible in the original sequence, and was motion-amplified 40 times to be clearly visible in the motion magnified sequence. Fig. 3.7 show frames from the original and motion magnified sequences. Notice the droop of the bookshelf close to the point at which force is applied. In this sequence we used a model with two layers and no outlier layer, and made user edits as described in the figure caption.

Swingset is the most challenging sequence of the three. In this sequence we used a model with six layers and one outlier layer. The sequence has periodic motion (the swingset structure), very fast motion (the person), random motion (the leaves) all within a complicated, textured scene. The motion magnified video (stills in Fig. 1) reveals the imperceptible deformations of the swingset supports in response to the person swinging. Note that the beams and swings of the swingset are grouped into a single motion layer, based on the *correlations* of their motions, not based on the uniformity of the motions. This allows pixels with a common motion cause to be motion magnified together as a group.

Our model of the translations of each pixel over time allows us to perform post-processing steps unrelated to motion magnification, as well. To compensate for the low frame rate of the digital still camera images, we used our motion layer model of the sequence to interpolate missing frames, synthetically achieving a higher frame rate. For pixels of the outlier layer, we have no motion model, so we sample-and-hold replicated those pixels within the interpolated frames. This can be seen from single-stepping through the output motion magnified video, which also shows which pixels were assigned to the outlier layer.

Figure 3.6. The magnification result (right) for a handstand (left). The motion magnified sequence exaggerates the small postural corrections needed to maintain balance.

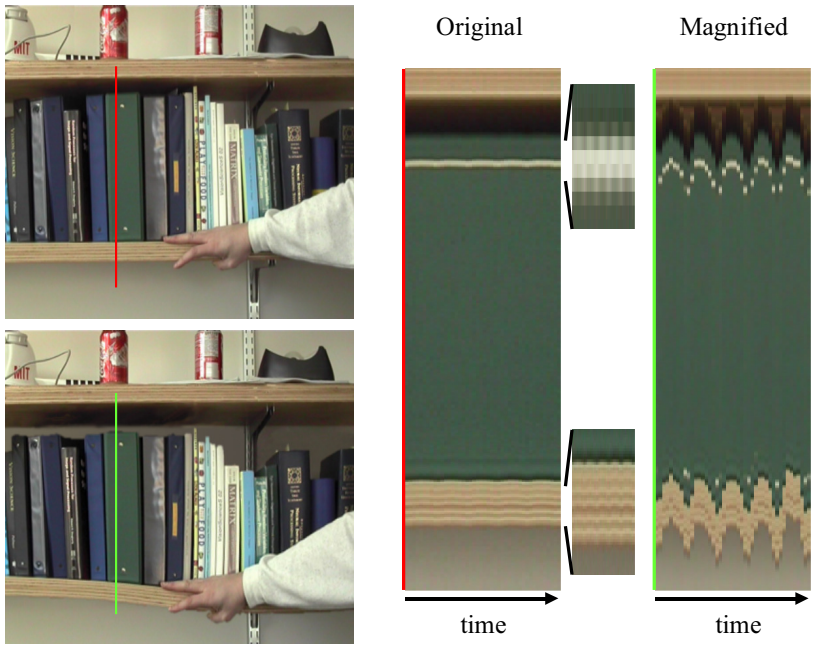


Figure 3.7. Under hand pressure, the bookshelf (on aluminium supports) undergoes motions that are made visible when magnified by a factor of 40 using motion magnification. User editing of reference frame masks refined the upper boundary between the books and the shelf. Also, the background layer required manual texture completion as little background is visible during the sequence.

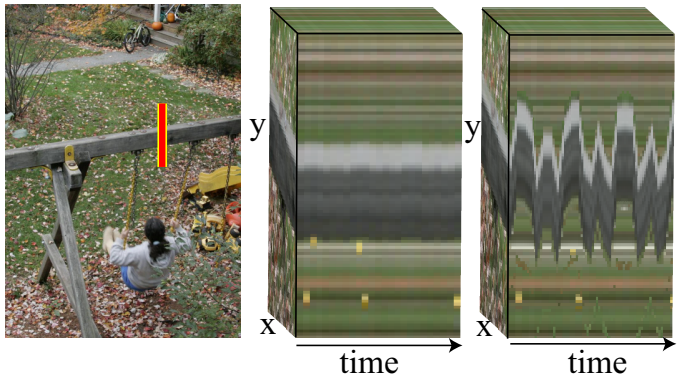


Figure 3.8. Section of the x, y, t volume from the original sequence (left) and the sequence after motion magnification. The detail shows a vertical section of the beam. The volume illustrates the amplification of the oscillation and also the filling of the texture behind the beam. Notice that after magnification, the motion is almost periodic despite the noise. So, the real motion of the beam is not just one single harmonic but a mixture. The original motion is amplified by a factor of 40.

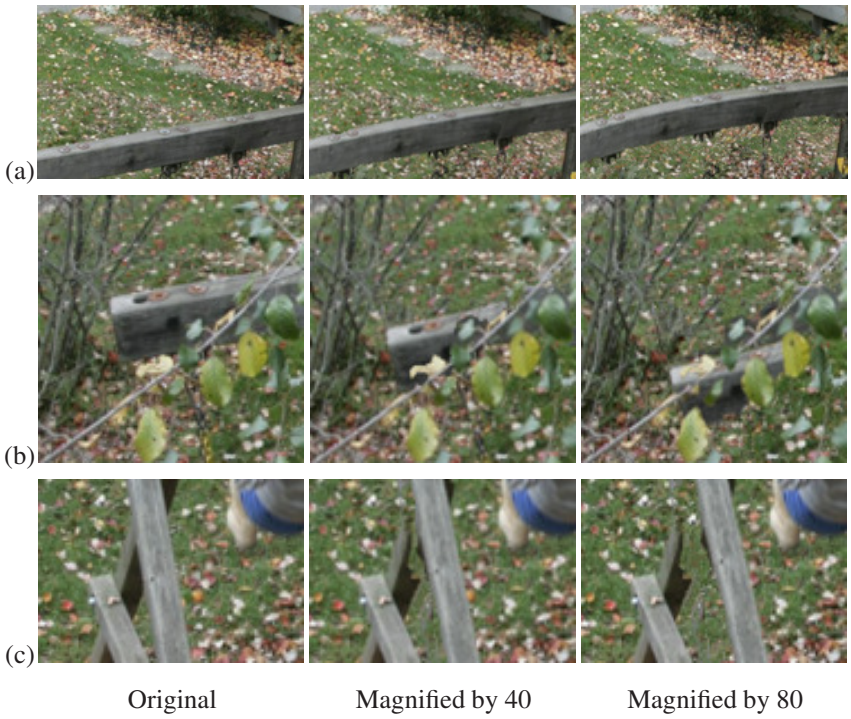


Figure 3.9. Details from frames of original and motion magnified swingset sequence, showing (a) beam curvature, (b) proper handling of occlusions, and (c) an artifact from imperfect automatic segmentation (before correction by the user).

Fig. 3.8 shows an x-y-t slice through part of the video volume of the swingset example, before and after motion magnification. The amplified beam motions are visible, as well as the textural filling-in of the background holes behind the displaced beam.

Fig. 3.9 shows details of the original and output swingset sequence, without user intervention, for motion magnifications of 40 and 80 times. Row (a) shows the bending of the support beam revealed by the magnified motion. Row (b) shows the leaf occlusions handled correctly in the synthesized output sequence. Row (c) shows a break artifact that occurred (before user editing) because the dark, occluded far leg of the swingset was not put in the same motion layer as the rest of the swingset.

■ 3.1.8 Applications

Since the publication of our work [67], the motion magnification system has been applied to several scientific and engineering fields.

In [41][1], the authors used our system to magnify wave motion along the ear's tectorial membrane at nanometer-scale displacement to make it more apparent. As shown in Figure 3.10 (a) and (c), the motion of the ear membrane is almost unnoticeable in the original sequence. Our motion magnification system, however, is able to magnify the small motion and make it visible, as shown in (b) and (d). Notice that for this sequence we only use one layer to estimate and magnify the motion.

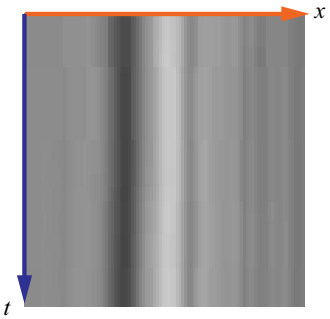
We also apply our system to magnifying the deformation of underground faults to help locate oil. One of the six frames of the underground fault at the same location taken in the past two decades were show in Figure 3.11 (a). We can barely perceive any motion from this sequence. In (b), the motion of the faults is magnified to help geologists to find oil underground. However, because the motion field of the faults is more complicated than the motion in previous examples, simply magnifying the displacement of each pixel individually may result in unexpected pixel distortions. We introduce spatial regularity, *i.e.* a truss model for pixels, to overcome this problem. In the truss model, the magnified displacement for each pixel is only a reference; the pixels should also move together with neighboring pixels as much as possible. The results of using this truss model are displayed in (c) and (d), where the layers of the faults move more consistently after the magnification.

Lastly, we use motion magnification to magnify the difference between the motions of a car passing through a speed bump with a normal load and a heavier load, as shown in Figure 3.12 (a) and (b), respectively. The car with heavier load tends to go down more after passing

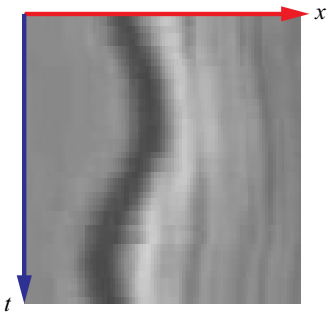
[1] See also http://web.mit.edu/newsoffice/2007/hearing-1010.html

(a) The 18th frame from the original sequence (orange arrow is where the slice in (c) is taken)

(b) The 18th frame from the magnified sequence (red arrow is where the slice in (d) is taken)

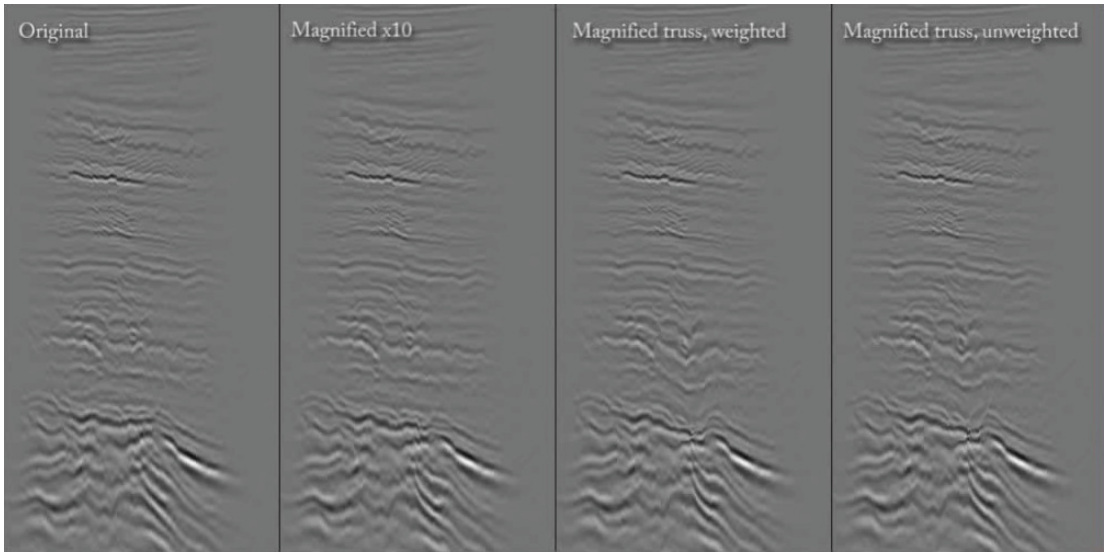


(c) The *x*-*t* slice of (a)



(d) The *x*-*t* slice of (b)

Figure 3.10. Motion magnification is applied to revealing the almost unnoticeable motion of ear's tectorial membrane at nanometer-scale displacement. The 18th frame of the original and magnified sequences is displayed in (a) and (b), respectively. To visualize the motion magnification effect, we take a slice at the same location, shown as orange and red line in (a) and (b), and plot how the line evolves with respect to time in (c) and (d), respectively. The motion becomes much more visible in (d) than in (c).

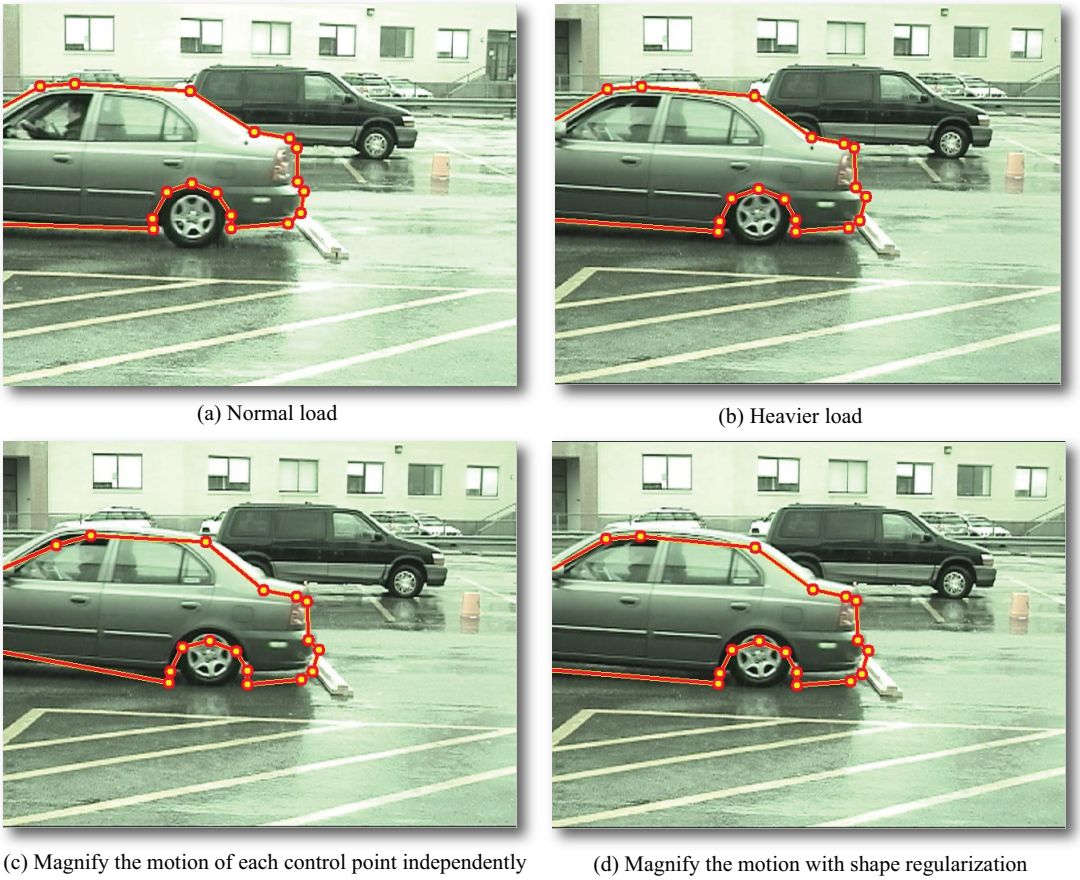


(a) (b) (c) (d)

Figure 3.11. Motion magnification is applied to magnify the temporal deformation of faults underground to help find oils. The original sequence (a) contain six frames taken over the past two decades from Shell Research. We can magnify the motion of each pixel individually (b), or fit a truss model to the pixels so that the spatial regularity is respected in the magnification, as shown in (c) and (d).

through the speed bump, but this difference is subtle to human eyes. Because the motion of the wheel is complicated and different from the body of the car, we use the motion annotation tool developed in Chapter 2 to label the contour, which is the representation of the motion we will work on (note that this is different from the pixel-wise motion we have been working on so far). The two sequences are not temporally aligned as the sequence with heavier load is slower with more frames. So we first apply a 1D version of the Lucas-Kanade algorithm [71] to aligning the two sequences according to the y-displacement of one feature point. As shown in Figure 3.13, the two signals are temporally registered after this process.

We magnify the difference between the two trajectories of each feature point and show the results in Figure 3.12 (c). Because the motion of each feature point is magnified independently, the contour of the car is deformed and breaks the overall car shape. To overcome this inconsistency, we impose shape regularity by projecting the individually magnified feature points to the rigid car shape, and obtain shape-consistent magnification results in Figure 3.12 (d). Using our motion magnification tool, we are able to visualize the subtle difference of the car load and make it more visible to inspectors.



(a) Normal load

(b) Heavier load

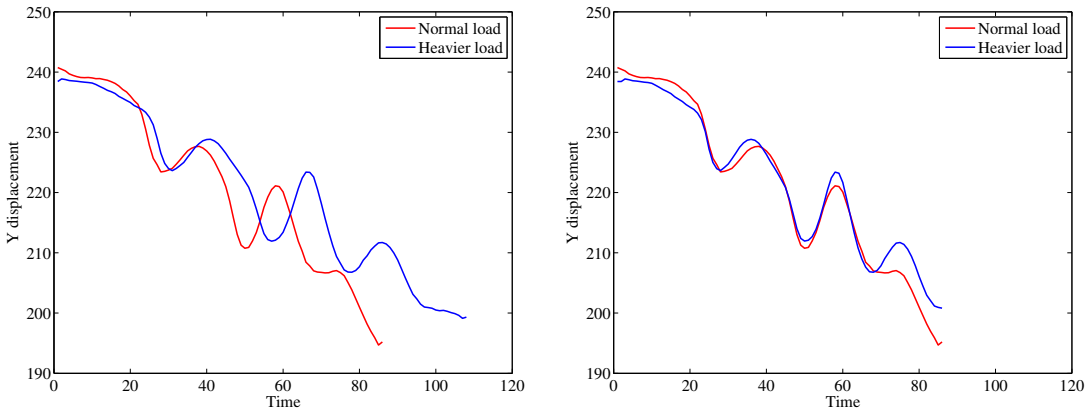
(c) Magnify the motion of each control point independently

(d) Magnify the motion with shape regularization

Figure 3.12. We use the human-assisted motion annotation in Chapter 2 to label the contour of a moving car running through a speed bump, with a normal load (a) and a heavier load (b), respectively. If we only magnify the difference between the vertical axis of each point, we can the magnified result as shown in (c), where the motion is magnified but inconsistent with respect to the shape of the car. After a shape regularity is imposed, we obtain magnified motion effect and respect the shape of the car in (d).

■ 3.1.9 Conclusion

We have presented a new technique, motion magnification, that reveals motions that would otherwise be invisible or very difficult to see. The input is a sequence of images from a stationary camera. The system automatically segments a reference frame into regions of "common fate", grouped by proximity, similar color, and correlated motions. Analogous to focussing a microscope, the user identifies the segment to modify, and specifies the motion magnification factor. The video sequence is then re-rendered with the motions of the selected layer magnified as desired. The output sequence allows the user to see the form and characteristics of the magnified



(a) Before temporal alignment (b) After temporal alignment

Figure 3.13. Before running the motion magnification system for the samples in Figure 3.12, we need to first align the two sequences because the sequences with heavier load is slower and has more frames. Let us look at the y-displacement of one feature point for the two sequences. Before we do anything, the two signals are not aligned as shown in (a). After applying a 1D version of the Lucas-Kanade algorithm [71], the two signals are aligned and ready for magnification.

motions in an intuitive display, as if the physical movements themselves had been magnified, then recorded.

■ 3.2 Contour Motion Analysis

Motion magnification requires reliable motion analysis, especially at occlusion boundaries. One type of occlusion boundary that we found particularly difficult to analyze is the boundaries of featureless objects. In fact, humans can reliably analyze visual motion under a diverse set of conditions, including textured as well as featureless objects. However, computer vision algorithms have focussed on conditions of texture, where junction or corner-like image structures are assumed to be reliable features for tracking [71, 51, 105]. But under other conditions, these features can generate spurious motions. T-junctions caused by occlusion can move in an image very differently than either of the objects involved in the occlusion event [78]. To properly analyze motions of featureless objects requires a different approach.

The spurious matching of T-junctions has been explained in [127] and [76]. We briefly restate it using the simple two bar stimulus in Figure 3.14 (from [127]). The gray bar is moving rightward in front of the leftward moving black bar, (a). If we analyze the motion locally, i.e. match to the next frame in a local circular window, the flow vectors of the corner and line points

are as displayed in Figure 3.14 (b). The T-junctions located at the intersections of the two bars move downwards, but there is no such a motion by the depicted objects.

One approach to handling the spurious motions of corners or T-junctions has been to detect such junctions and remove them from the motion analysis [127, 81]. However, T-junctions are often very difficult to detect in a static image from local, bottom-up information [76]. Motion at occluding boundaries has been studied, for example in [15]. The boundary motion is typically analyzed locally, which can again lead to spurious junction trackings. We are not aware of an existing algorithm that can properly analyze the motions of featureless objects.

In this section, we use a boundary-based approach which does not rely on motion estimates at corners or junctions. We develop a graphical model which integrates local information and assigns probabilities to candidate contour groupings in order to favor motion interpretations corresponding to the motions of the underlying objects. Boundary completion and discounting the motions of spurious features result from optimizing the graphical model states to explain the contours and their motions. Our system is able to automatically detect and group the boundary fragments, analyze the motion correctly, as well as exploit both static and dynamic cues to synthesize the illusory boundaries (c).

We represent the boundaries at three levels of grouping: *edgelets*, *boundary fragments* and *contours*, where a fragment is a chain of edgelets and a contour is a chain of fragments. Each edgelet within a boundary fragment has a position and an orientation and carries local evidence for motion. The main task of our model is then to group the boundary fragments into contours so that the local motion uncertainties associated with the edgelets are disambiguated and occlusion or other spurious feature events are properly explained. The result is a specialized motion tracking algorithm that properly analyzes the motions of textureless objects.

Our system consists of four conceptual steps, discussed over the next three sections (the last two steps happen together while finding the optimal states in the graphical model):

(a) **Boundary fragment extraction**: Boundary fragments are detected in the first frame.

(b) **Edgelet tracking with uncertainties**: Boundary fragments are broken into edgelets, and, based on local evidence, the probability distribution is found for the motion of each edgelet of each boundary fragment.

(c) **Grouping boundary fragments into contours**: Boundary fragments are grouped, using both temporal and spatial cues.

(d) **Motion estimation**: The final fragment groupings disambiguate motion uncertainties and specify the final inferred motions.

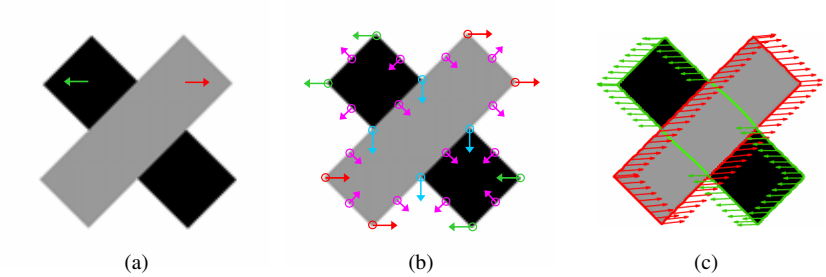


Figure 3.14. Illustration of the spurious T-junction motion. (a) The front gray bar is moving to the right and the black bar behind is moving to the left [127]. (b) Based on a local window matching, the eight corners of the bars show the correct motion, whereas the T-junctions show spurious downwards motion. (c) Using the boundary-based representation our system is able to correctly estimate the motion and generate the illusory boundary as well.

We restrict the problem to two-frame motion analysis though the algorithm can easily be extended to multiple frames.

■ 3.2.1 Boundary Fragment Extraction

Extracting boundaries from images is a nontrivial task by itself. We use a simple algorithm for boundary extraction, analyzing oriented energy using steerable filters [39] and tracking the boundary in a manner similar to that of the Canny edge detector [25]. A more sophisticated boundary detector can be found in [74]; occluding boundaries can also be detected using special cameras [85]. However, for our motion algorithm designed to handle the special case of textureless objects, we find that our simple boundary detection algorithm works well.

Mathematically, given an image I, we seek to obtain a set of fragments $\mathbf{B} = \{\mathbf{b}_i\}$, where each fragment $\mathbf{b}_i$ is a chain of *edgelets* $\mathbf{b}_i = \{e_{ik}\}_{k=1}^{n_i}$. Each edgelet $e_{ik} = \{p_{ik}, \theta_{ik}\}$ is a particle which embeds both location $p_{ik} \in \mathbb{R}^2$ and orientation $\theta_{ik} \in [0, 2\pi)$ information.

We use H4 and G4 steerable filters [39] to filter the image and obtain orientation energy per pixel. These filters are selected because they describe the orientation energies well even at corners. For each pixel we find the maximum energy orientation and check if it is local maximum within a slice perpendicular to this orientation. If that is true and the maximum energy is above a threshold T_1 we call this point a *primary* boundary point. We collect a pool of primary boundary points after running this test for all the pixels.

We find the primary boundary point with the maximum orientation energy from the pool and do bidirectional contour tracking, consisting of *prediction* and *projection* steps. In the pre-

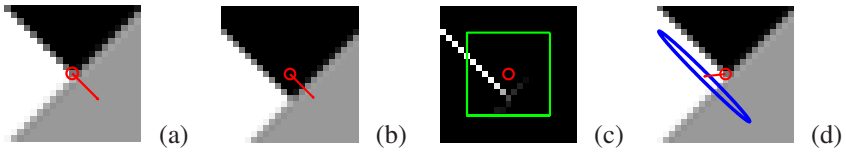


Figure 3.15. The local motion vector is estimated for each contour in isolation by selectively comparing orientation energies across frames. (a) A T-junction of the two bar example showing the contour orientation for this motion analysis. (b) The other frame. (c) The relevant orientation energy along the boundary fragment, both for the 2nd frame. A Gaussian pdf is fit to estimate flow, weighted by the oriented energy. (d) Visualization of the Gaussian pdf. The possible contour motions are unaffected by the occluding contour at a different orientation and no spurious motion is detected at this junction.

diction step, the current edgelet generates a new one by following its orientation with a certain step size. In the projection step, the orientation is locally maximized both in the orientation bands and within a small spatial window. The tracking is stopped if the energy is below a threshold T_2 or if the turning angle is above a threshold. The primary boundary points that are close to the tracked trajectory are removed from the pool. This process is repeated until the pool is empty. The two thresholds T_1 and T_2 play the same roles as those in Canny edge detection [25]. While the boundary tracker should stop at sharp corners, it can turn around and continue tracking. We run a postprocess to break the boundaries by detecting points of curvature local maxima which exceed a curvature threshold.

■ 3.2.2 Edgelet Tracking with Uncertainties

We next break the boundary contours into very short edgelets and obtain the probabilities, based on local motion of the boundary fragment, for the motion vector at each edgelet. We cannot use conventional algorithms, such as Lucas-Kanade [71], for local motion estimation since they rely on corners. The orientation θ_{ik} for each edgelet was obtained during boundary fragment extraction. We obtain the motion vector by finding the spatial offsets of the edgelet which match the orientation energy along the boundary fragment in this orientation. We fit a Gaussian distribution $\mathcal{N}(\mu_{ik}, \Sigma_{ik})$ of the flow weighted by the orientation energy in the window. The mean and covariance matrix is added to the edgelet: $e_{ik} = \{p_{ik}, \theta_{ik}, \mu_{ik}, \Sigma_{ik}\}$. This procedure is illustrated in Figure 3.15.

Grouping the boundary fragments allows the motion uncertainties to be resolved. We next discuss the mathematical model of grouping as well as the computational approach.

■ 3.2.3 Boundary Fragment Grouping and Motion Estimation

Two Equivalent Representations for Fragment Grouping

The essential part of our model is to find the connection between the boundary fragments. There are two possible representations for grouping. One representation is the connection of each end of the boundary fragment. We formulate the probability of this connection to model the *local* saliency of contours. The other equivalent representation is a chain of fragments that forms a contour, on which *global* statistics are formulated, e.g. structural saliency [101]. Similar local and global modeling of contour saliency was proposed in [86]; in [73], both edge saliency and curvilinear continuity were used to extract closed contours from static images. In [96], contour ends are grouped using loopy belief propagation to interpret contours.

The connections between fragment ends are modeled by switch variables. For each boundary fragment $\mathbf{b}_i$, we use a binary variable $\{0,1\}$ to denote the two ends of the fragment, i.e. $\mathbf{b}_i^{(0)} = e_{i1}$ and $\mathbf{b}_i^{(1)} = e_{i,n_i}$. Let switch variable $S(i,t_i) = (j,t_j)$ denote the connection from $\mathbf{b}_i^{(t_i)}$ to $\mathbf{b}_j^{(t_j)}$. This connection is *exclusive*, i.e. each end of the fragment should either connect to one end of the other fragment, or simply have no connection. An exclusive switch is further called *reversible*, i.e.

$$\text{if } \; S(i,t_i) = (j,t_j), \; \text{then } \; S(j,t_j) = (i,t_i),$$

or in a more compact form

$$S(S(i,t_i)) = (i,t_i). \tag{3.10}$$

When there is no connection to $\mathbf{b}_i^{(t_i)}$, we simply set $S(i,t_i) = (i,t_i)$. We use the binary function $\delta[S(i,t_i) - (j,t_j)]$ to indicate whether there is a connection between $\mathbf{b}_i^{(t_i)}$ and $\mathbf{b}_j^{(t_j)}$. The set of all the switches are denoted as $\mathbf{S} = \{S(i,t_i) | i = 1:N, t_i = 0,1\}$. We say $\mathbf{S}$ is reversible if every switch variable satisfies Eqn. (3.10). The reversibility of switch variables is shown in Figure 3.16 (b) and (c).

From the values of the switch variables we can obtain contours, which are chains of boundary fragments. A fragment chain is defined as a series of the end points $\mathbf{c} = \{(\mathbf{b}_{i_1}^{(x_1)}, \mathbf{b}_{i_1}^{(\overline{x}_1)}), \cdots, (\mathbf{b}_{i_m}^{(x_m)}, \mathbf{b}_{i_m}^{(\overline{x}_m)})\}$. The chain is specified by fragment label $\{i_1, \cdots, i_m\}$ and end label $\{x_1, \cdots, x_m\}$. It can be either open or closed. The order of the chain is determined by the switch variable. Each end appears in the chain at most once. The notation of a chain is not unique. Two open chains are identical if the fragment and end labels are reversed. Two closed chains are identical if they match each other by rotating one of them. These identities are guaranteed from the reversibility of the switch variables. A set of chains $\mathbf{C} = \{c_i\}$ can be uniquely extracted based

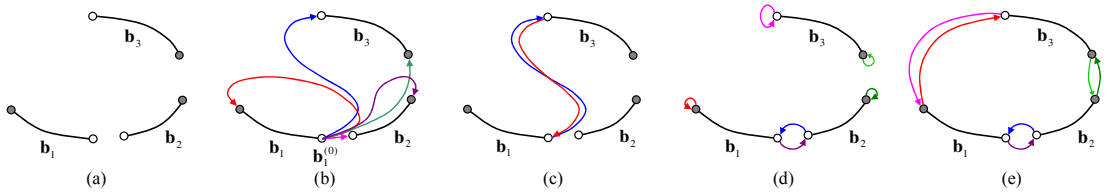


Figure 3.16. A simple example illustrating switch variables, reversibility and fragment chains. The color arrows show the switch variables. The empty circle indicates end 0 and the filled indicates end 1. (a) Shows three boundary fragments. Theoretically $\mathbf{b}_1^{(0)}$ can connect to any of the other ends including itself, (b). However, the switch variable is *exclusive*, i.e. there is only one connection to $\mathbf{b}_1^{(0)}$, and *reversible*, i.e. if $\mathbf{b}_1^{(0)}$ connects to $\mathbf{b}_3^{(0)}$, then $\mathbf{b}_3^{(0)}$ should also connect to $\mathbf{b}_1^{(0)}$, as shown in (c). Figures (d) and (e) show two of the legal contour groupings for the boundary fragments: two open contours and a closed loop contour.

on the values of the reversible switch variables, as illustrated in Figure 3.16 (d) and (e).

The Graphical Model for Boundary Fragment Grouping

Given the observation O, the two images, and the boundary fragments $\mathbf{B}$, we want to estimate the flow vectors $\mathbf{V} = \{\mathbf{v}_i\}$ and $\mathbf{v}_i = \{v_{ik}\}$, where each v_{ik} associates with edgelet e_{ik}, and the grouping variables $\mathbf{S}$ (switches) or equivalently $\mathbf{C}$ (fragment chains). Since the grouping variable $\mathbf{S}$ plays an essential role in the problem, we shall first infer $\mathbf{S}$ and then infer $\mathbf{V}$ based on $\mathbf{S}$.

We use two equivalent representations for boundary grouping, switch variables and chains. We use $\delta[S(S(i,t_i)) - (i,t_i)]$ for each end to enforce the reversibility. Suppose otherwise $S(i_1,t_{i_1}) = S(i_2,t_{i_2}) = (j,t_j)$ for $i_1 \neq i_2$. Let $S(j,t_j) = (i_1,t_{i_1})$ without loss of generality, then $\delta[S(S(i_2,t_{i_2})) - (i_2,t_{i_2})] = 0$, which means that the switch variables are not reversible.

We use a function $\lambda(S(i,t_i);\mathbf{B},O)$ to measure the distribution of $S(i,t_i)$, i.e. how likely $\mathbf{b}_i^{(t_i)}$ connects to the end of other fragments. Intuitively, two ends should be connected if

- ◇ **Motion similarity** the distributions of the motion of the two end edgelets are similar;
- ◇ **Curve smoothness** the illusory boundary to connect the two ends is smooth;
- ◇ **Contrast consistency** the brightness contrast at the two ends consistent with each other.

We write $\lambda(\cdot)$ as a product of three terms, one enforcing each criterion. We shall follow the example in Figure 3.17 to simplify the notation, where the task is to compute $\lambda(S(1,0) =$

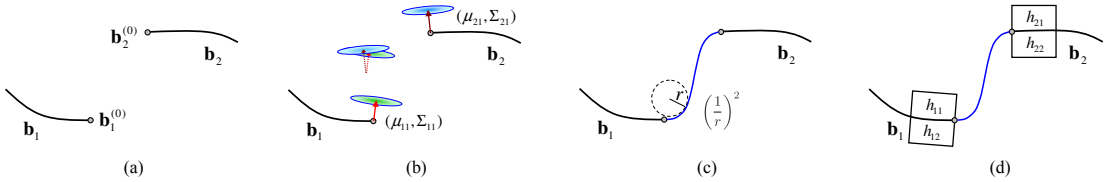


Figure 3.17. An illustration of local saliency computation. (a) Without loss of generalization we assume the two ends to be $\mathbf{b}_1^{(0)}$ and $\mathbf{b}_2^{(0)}$. (b) The KL divergence between the distributions of flow vectors are used to measure the motion similarity. (c) An illusory boundary γ is generated by minimizing the energy of the curve. The sum of square curvatures are used to measure the curve smoothness. (d) The means of the local patches located at the two ends are extracted, i.e. h_{11} and h_{12} from $b_1^{(0)}$, h_{21} and h_{22} from $b_2^{(0)}$, to compute contrast consistency.

$(2,0)$). The first term is the KL divergence between the two Gaussian distributions of the flow vectors

$$\exp\{-\alpha_{KL} KL(\mathcal{N}(\mu_{11},\Sigma_{11}),\mathcal{N}(\mu_{21},\Sigma_{21}))\}, \tag{3.11}$$

where α_{KL} is a scaling factor. The second term is the local saliency measure on the illusory boundary γ that connects the two ends. The illusory boundary is simply generated by minimizing the energy of the curve. The saliency is defined as

$$\exp\left\{-\alpha_\gamma \int_\gamma \left(\frac{d\theta}{ds}\right)^2 ds\right\}, \tag{3.12}$$

where $\theta(s)$ is the slope along the curve, and $\frac{d\theta}{ds}$ is local curvature [101]. α_γ is a scaling factor. The third term is computed by extracting the mean of local patches located at the two ends

$$\exp\left\{-\frac{d_{max}}{2\sigma_{max}^2} - \frac{d_{min}}{2\sigma_{min}^2}\right\}, \tag{3.13}$$

where $d_1 = (h_{11}-h_{21})^2$, $d_2 = (h_{12}-h_{22})^2$, and $d_{max} = \max(d_1,d_2)$, $d_{min} = \min(d_1,d_2)$. $\sigma_{max} > \sigma_{min}$ are the scale parameters. h_{11}, h_{12}, h_{21}, h_{22} are the means of the pixel values of the four patches located at the two end points. For self connection we simply set a constant value: $\lambda(S(i,t_i)=(i,t_i))=\tau$.

We use a function $\psi(\mathbf{c}_i;\mathbf{B},O)$ to model the structural saliency of contours. It was discovered in [77] that convex occluding contours are more salient, and additional T-junctions along the contour may increase or decrease the occlusion perception. Here we simply enforce that a contour should have no self-intersection. $\psi(\mathbf{c}_i;\mathbf{B},O)=1$ if there is no self intersection and $\psi(\mathbf{c}_i;\mathbf{B},O)=0$ otherwise.

Thus, the (discrete) graphical model favoring the desired fragment grouping is

$$\Pr(\mathbf{S}; B, O) = \frac{1}{Z_S} \prod_{i=1}^{N} \prod_{t_i=0}^{1} \lambda(S(i,t_i); \mathbf{B}, O)\delta[S(S(i,t_i)) - (i,t_i)] \cdot \prod_{j=1}^{M} \psi(\mathbf{c}_j; \mathbf{B}, O), \quad (3.14)$$

where Z_S is a normalization constant. Note that this model measures both the switch variables $S(i,t_i)$ for local saliency and the fragment chains $\mathbf{c}_i$ to enforce global structural saliency.

Gaussian MRF on Flow Vectors

Given the fragment grouping, we model the flow vectors $\mathbf{V}$ as a Gaussian Markov random field (GMRF). The edgelet displacement within each boundary fragment should be smooth and match the observation along the fragment. The probability density is formulated as

$$\varphi(\mathbf{v}_i; \mathbf{b}_i) = \prod_{k=1}^{n_i} \exp\{-(v_{ik} - \mu_{ik})^T \Sigma_{ik}^{-1} (v_{ik} - \mu_{ik})\} \prod_{k=1}^{n_i - 1} \exp\{-\frac{1}{2\sigma^2} \|v_{ik} - v_{i,k+1}\|^2\}, \quad (3.15)$$

where μ_{ik} and Σ_{ik} are the motion parameters of each edgelet estimated in Sect 3.

We use $\mathbf{V}(i,t_i)$ to denote the flow vector of end t_i of fragment $\mathbf{b}_i$. We define $\mathbf{V}(S(i,t_i)) = \mathbf{V}(j,t_j)$ if $S(i,t_i) = (j,t_j)$. Intuitively the flow vectors of the two ends should be similar if they are connected, or mathematically

$$\phi(\mathbf{V}(i,t_i), \mathbf{V}(S(i,t_i))) = \begin{cases} 1 & \text{if} S(i,t_i) = (i,t_i), \\ \exp\{-\frac{1}{2\sigma^2} \|\mathbf{V}(i,t_i) - \mathbf{V}(S(i,t_i))\|^2\} & \text{otherwise.} \end{cases} \quad (3.16)$$

The (continuous) graphical model of the flow vectors is therefore defined as

$$\Pr(\mathbf{V}|\mathbf{S}; \mathbf{B}) = \frac{1}{Z_V} \prod_{i=1}^{N} \varphi(\mathbf{v}_i; \mathbf{b}_i) \prod_{t_i=0}^{1} \phi(\mathbf{V}(i,t_i), \mathbf{V}(S(i,t_i))) \quad (3.17)$$

where Z_V is a normalization constant. When $\mathbf{S}$ is given it is a GMRF which can be solved by least squares.

Inference

Having defined the graphical model to favor the desired motion and grouping interpretations, we need to find the state parameters that best explain the image observations. The natural decomposition of $\mathbf{S}$ and $\mathbf{V}$ in our graphical model

$$\Pr(\mathbf{V}, \mathbf{S}; \mathbf{B}, O) = \Pr(\mathbf{S}; \mathbf{B}, O) \cdot \Pr(\mathbf{V}|\mathbf{S}; \mathbf{B}, O), \quad (3.18)$$

(where $\Pr(\mathbf{S};\mathbf{B},O)$ and $\Pr(\mathbf{V}|\mathbf{S};\mathbf{B},O)$ are defined in Eqn. (3.14) and (3.17) respectively) lends itself to performing two-step inference. We first infer the boundary grouping $\mathbf{B}$, and then infer $\mathbf{V}$ based on $\mathbf{B}$. The second step is simply to solve least square problem since $\Pr(\mathbf{V}|\mathbf{S};\mathbf{B},O)$ is a GMRF. This approach does not globally optimize Eqn. (3.18) but results in reasonable solution because $\mathbf{V}$ strongly depends on $\mathbf{S}$. The density function $\Pr(\mathbf{S};\mathbf{B},O)$ is not a random field, so we use importance sampling [72] to obtain the marginal distribution $\Pr(S(i,t_i);\mathbf{B},O)$. The proposal density of each switch variable is set to be

$$q\left(S(i,t_i)=(j,t_j)\right) \propto \frac{1}{Z_q}\lambda\left(S(i,t_i)=(j,t_j)\right)\lambda\left(S(j,t_j)=(i,t_i)\right) \tag{3.19}$$

where $\lambda(\cdot)$ has been normalized to sum to 1 for each end. We found that this bidirectional measure is crucial to take valid samples. To sample the proposal density, we first randomly select a boundary fragment, and connect to other fragments based on $q(S(i,t_i))$ to form a contour (a chain of boundary fragments). Each end is sampled only once, to ensure reversibility. This procedure is repeated until no fragment is left. In the importance step we run the binary function $\psi(\mathbf{c}_i)$ to check that each contour has no self-intersection. If $\psi(\mathbf{c}_i)=0$ then this sample is rejected. The marginal distributions are estimated from the samples. Lastly the optimal grouping is obtained by replacing random sampling with selecting the maximum-probability connection over the estimated marginal distributions. The number of samples needed depends on the number of the fragments. In practice we find that n^2 samples are sufficient for n fragments.

■ 3.2.4 Experimental Results

Figure 5.9 shows the boundary extraction, grouping, and motion estimation results of our system for both real and synthetic examples[2]. All the results are generated using the same parameter settings. The algorithm is implemented in MATLAB, and the running time varies from ten seconds to a few minutes, depending on the number of the boundary fragments found in the image.

The two-bar examples in Figure 1(a) yields fourteen detected boundary fragments in Figure 5.9(a) and two contours in (b). The estimated motion matches the ground truth at the T-junctions. The fragments belonging to the same contour are plotted in the same color and the illusory boundaries are synthesized as shown in (c). The boundaries are warped according to the estimated flow and displayed in (d). The hallucinated illusory boundaries in frame 1 (c) and 2 (d) are plausible amodal completions.

[2]The results can be viewed online http://people.csail.mit.edu/celiu/contourmotions/

(a) Dancer frame 1 (b) Dancer frame 2 (c) Chair frame 1 (d) Chair frame 2

Figure 3.18. Input images for the non-synthetic examples of Figure 6. The dancer's right leg is moving downwards and the chair is rotating (note the changing space between the chair's arms).

The second example is the Kanizsa square where the frontal white square moves to the right bottom. Twelve fragments are detected in (a) and five contours are grouped in (b). The estimated motion and generated illusory boundary also match the ground truth and human perception. Notice that the arcs tend to connect to other ones if we do not impose the structural saliency $\psi(\cdot)$.

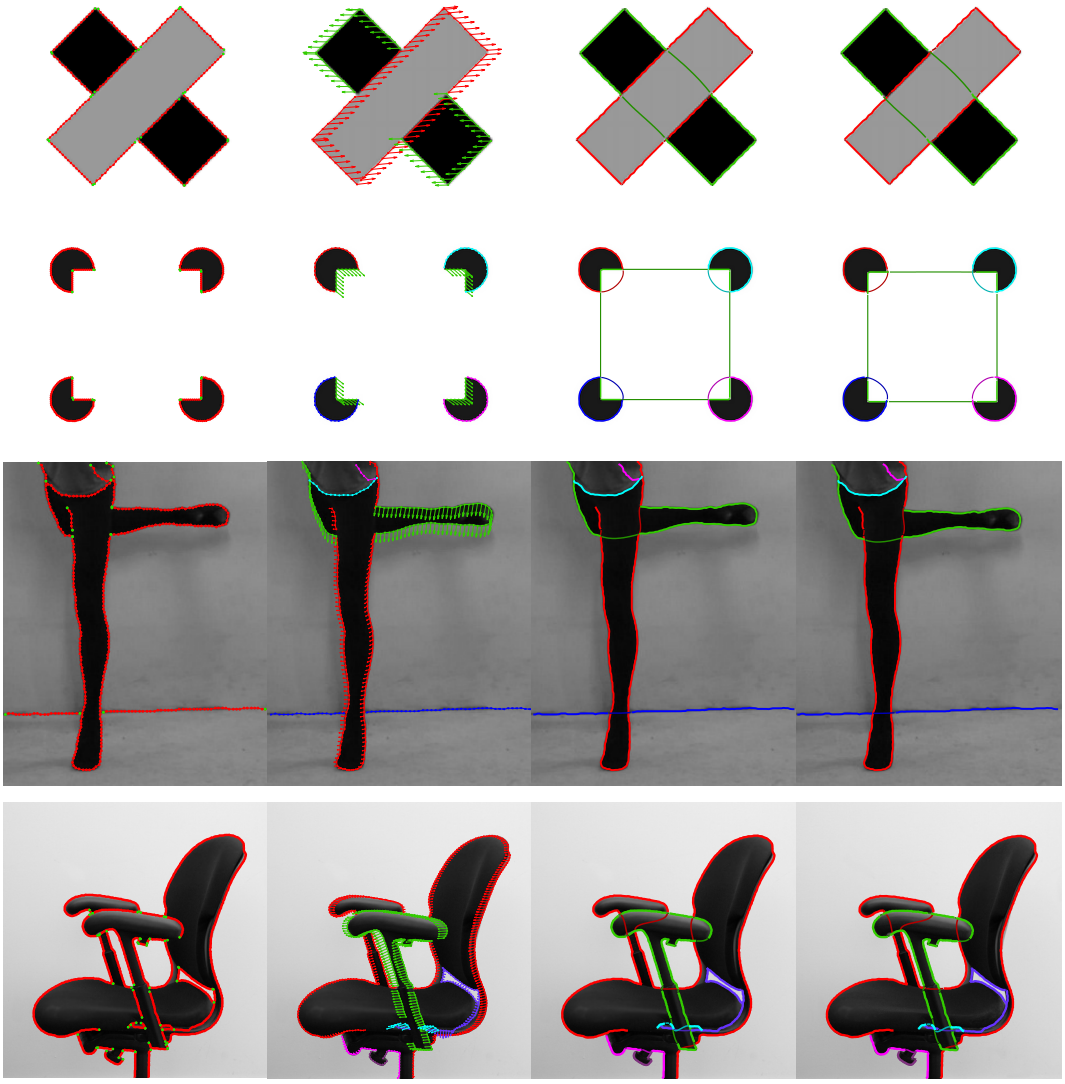
We apply our system to a video of a dancer (Figure 3.18 (a) and (b)). In this stimulus the right leg moves downwards, but there is weak occluding boundary at the intersection of the legs. Eleven boundary fragments are extracted in (a) and five contours are extracted in (b). The estimated motion (b) matches the ground truth. The hallucinated illusory boundary in (c) and (d) correctly connect the occluded boundary of the right leg and the invisible boundary of the left leg.

The final row shows challenging images of a rotating chair (Figure 3.18 (c) and (d)), also showing proper contour completion and motion analysis. Thirty-seven boundary fragments are extracted and seven contours are grouped. To complete the occluded contours of this image would be nearly impossible working only from a static image. Exploiting motion as well as static information, our system is able to complete the contours properly.

Note that the traditional motion analysis algorithms fail at estimating motion for these examples (see supplementary videos) and would thus also fail at correctly grouping the objects based on the motion cues.

■ 3.3 Conclusion

We propose a novel boundary-based representation to estimate motion under the challenging visual conditions of moving textureless objects. Ambiguous local motion measurements are resolved through a graphical model relating edgelets, boundary fragments, completed contours,



(a) Extracted boundaries (b) Estimated flow (c) Frame 1 (d) Frame 2

Figure 3.19. Experimental results for some synthetic and real examples. The same parameter settings were used for all examples. Column (a): Boundary fragments are extracted using our boundary tracker. The red dots are the edgelets and the green ones are the boundary fragment ends. Column (b): Boundary fragments are grouped into contours and the flow vectors are estimated. Each contour is shown in its own color. Columns (c): the illusory boundaries are generated for the first and second frames. The gap between the fragments belonging to the same contour are linked exploiting both static and motion cues in Eq. (3.14).

and their motions. Contours are grouped and their motions analyzed simultaneously, leading to the correct handling of otherwise spurious occlusion and T-junction features. The motion cues help the contour completion task, allowing completion of contours that would be difficult or impossible using only low-level information in a static image. A motion analysis algorithm such as this one that correctly handles featureless contour motions is an essential element in a visual system's toolbox of motion analysis methods.

Chapter 4

SIFT Flow: Dense Correspondence Across Scenes

So far we have been discussing motion analysis in video sequences. The concept of "motion" has been limited to temporal consistent frames in videos. However, we can generalize object motion to very different images. For example, there are two street-scene images that both contain cars, buildings, roads and pedestrians. Although flipping back and forth these two images would not make humans perceive any object motion, we can argue that the cars in one image "move" to the cars in the other image even though these cars can be of different type and color. From this chapter on, we will interchangeably use *motion* and *correspondence* because the idea of image alignment falls into the traditional image correspondence regime[1].

4.1 Introduction

Image alignment and registration is a central topic in computer vision. For example, aligning different views of the same scene has been studied for the purpose of image stitching [113] and stereo matching [98], *e.g.*, Figure 4.1 (a). The considered transformations are relatively simple (*e.g.*, parametric motion for image stitching and 1D disparity for stereo), and images to register are typically assumed to have the same pixel value after applying the geometric transformation.

The image alignment problem becomes more complicated for dynamic scenes in video sequences, as is the case of optical flow estimation [51, 71, 22]. The correspondence problem between two adjacent frames in the video is often formulated as an estimation of a 2D flow field. The extra degree of freedom (from 1D in stereo to 2D in optical flow) introduces an ad-

[1] In my opinion, *motion* has some physical meanings, *i.e.* an object moves physically from one image to the other. In contrast, *correspondence* is about how pixels in one image correspond to the pixels in the other. I prefer to use *correspondence* in this chapter because there is no physical object movement in scene alignment.

ditional level of complexity. Typical assumptions in optical flow algorithms include brightness constancy and piecewise smoothness of the pixel displacement field [9, 14].

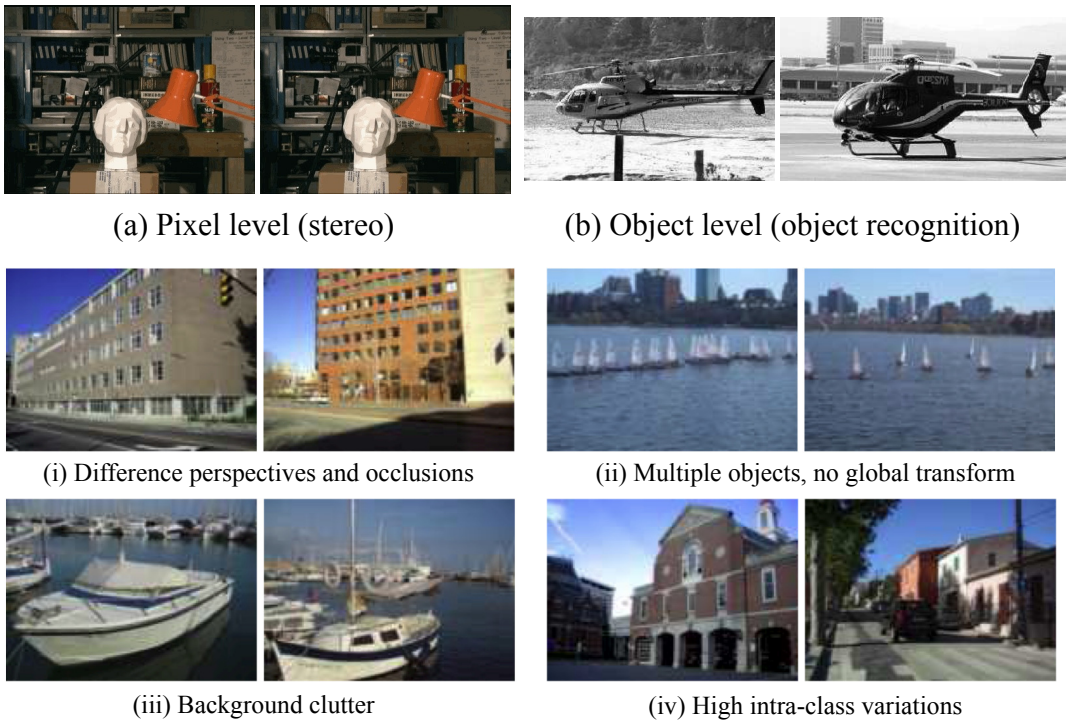
Image alignment becomes even more difficult in the object recognition scenario, where the goal is to align different instances of the same object category, as illustrated in Figure 4.1 (b). Sophisticated object representations [10, 12, 34, 129] have been developed to cope with the variations in objects' shape and appearance. However, the methods still typically require objects to be salient and large, visually very similar and with limited background clutter.

In this work, we are interested in a seemingly impossible task of aligning images depicting different instances of the same *scene category*. Image alignment at scene level is thus called *scene alignment*. As illustrated in Figure 4.1 (c), the two images to match may contain object instances captured from different viewpoints, placed at different spatial locations, or imaged at different scales. The two images may contain different quantities of the same objects. In addition, some objects present in one image might be missing in the other. Due to these issues the scene alignment problem is extremely challenging. Ideally, in scene alignment we want to build the correspondence at the semantic level, *i.e.* buildings correspond to buildings, windows to windows, sky to sky, and sailboats to sailboats. But the current object detection and recognition techniques are not robust enough to detect and recognize all objects in images.

We take a different approach for scene alignment by matching local, salient and transform-invariant image structures. We hope that semantically meaningful correspondences can be established through matching these image structures. Moreover, we want to have a simple, effective, object-free model to align all the pairs in Figure 4.1 (c).

Inspired by optical flow that is able to produce dense, pixel-to-pixel correspondence between two images, we propose *SIFT flow* by matching SIFT descriptors rather than raw pixel values and adopting the computational framework of optical flow. In the SIFT flow, a SIFT descriptor [70] is extracted at each pixel to characterize local image structures and encode contextual information. A discrete, discontinuity preserving, flow estimation algorithm is used to match the SIFT descriptors between two images. The use of SIFT features allows robust matching across different scene/object appearances and the discontinuity-preserving spatial model allows matching of objects located at different parts of the scene. A coarse-to-fine matching scheme is designed to significantly accelerate the flow estimation process.

Optical flow is only applied to two adjacent frames in a video sequence in order to obtain meaningful correspondences; likewise, we need to define the *neighborhood* for SIFT flow. Motivated by the recent progress in large image database methods [49, 91], we define the neighbors of SIFT flow as the top matches queried from a large database. The chance that some



(a) Pixel level (stereo)

(b) Object level (object recognition)

(i) Difference perspectives and occlusions

(ii) Multiple objects, no global transform

(iii) Background clutter

(iv) High intra-class variations

(c) Scene level (scene alignment)

Figure 4.1. Image alignment resides at different levels. Researchers used to study image alignment problem at pixel level (a) where the two images are taken at the same scene with slightly different time or perspective [98]. Recently, correspondence has been established at object level (b) for object recognition [12]. We are interested in image alignment at scene level, where two images come from the same scene category but different instances. As shown in (c), scene alignment is a challenging problem; the correspondence between scenes is not as obvious to human eyes as the correspondence at pixel and object levels. SIFT flow is proposed to align the examples in (c) for scene alignment.

of the nearest neighbors share the same scene category with the input image becomes higher as the database grows larger, and SIFT flow is able to obtain some semantically meaningful correspondences between a query image and its nearest neighbors.

We apply SIFT flow to two original applications, which both rely on finding and aligning images of similar scenes in a large collection of images or videos. The first application is motion prediction from a single static image, where a motion field is hallucinated for an input image using a large database of videos. The second application is motion transfer, where we animate a still image using object motions transferred from a similar moving scene. We also apply SIFT flow back to the regime of traditional image alignment. We study satellite image registration,

where two images were taken several years apart with different local appearances. We also use SIFT flow for face recognition, especially the scenario where there are not sufficient samples for training. Through these examples we demonstrate the potential of SIFT flow for broad applications in computer vision and computer graphics.

The rest of the chapter is organized as follows: after reviewing the related work in Sect. 4.2, we introduce the concept of SIFT flow and the inference algorithm in Sect. 4.3. In Sect. 4.4, we apply SIFT flow to video retrieval with many examples of correspondences between different scenes. In Sect. 4.5, we show how to infer the motion field from a single image, and how to animate a still image, both with the support of a large video database and scene alignment. We further apply SIFT flow back to image alignment at pixel level through satellite image registration, and at object level through face recognition in Sect. 4.6. After briefly discussing how SIFT flow fits in the literature of image alignment in Sect. 4.7, we conclude the chapter in Sect. 4.8.

■ 4.2 Related Work

Image alignment (or image registration, correspondence) is a broad topic in computer vision, computer graphics and medical imaging, covering stereo, motion analysis, video compression, shape registration, and object recognition. It is beyond the scope of this chapter to give a thorough review on image alignment. Please refer to [113] for a comprehensive review on image alignment. In this section we want to review the image alignment literature focusing on *what* to align, or the features that are consistent across images, *e.g.*, pixels, edges, descriptors; *which* way to align, or the representation of the alignment, *e.g.*, sparse vs. dense, parametric vs. nonparametric; and *how* to align, or the computational methods to obtain alignment parameters. Moreover, correspondence can be established between two images, or between an image and image models such as active appearance models [28] . We will focus on image to image correspondence.

In image alignment we must first define the features based on which image correspondence is established: the invariant features that do not change from one image to another. In stereo [47] and optical flow [71, 51], arguably the first areas of image alignment in computer vision, brightness constancy assumption was often made for building the correspondence between two images. But soon researchers came to realize that pixel values are not reliable for image matching due to lighting, perspective and noise [45]. Features such as phase [38], filter banks [54], mutual information [120] and gradient [20] are used to match images since they

are more reliable across frames. But these low-level features are not able to cope with drastic changes between two images. Middle-level representations such as scale-invariant feature transform (SIFT) [70], shape context [11, 12], histogram of oriented gradients (HOG) [31] have been introduced to account for variations within object categories. Although these middle-level representations have been widely used for object detection and recognition, little has been investigated for exploring features to establish correspondence at scene level.

The representation of the correspondence is another important aspect of image alignment. One can utilize the information of every pixel to obtain a dense correspondence, or merely use sparse feature points. The form of the correspondence can be pixel-wise displacement such as a 1-D disparity map (stereo) and a 2-D flow field (optical flow), or parametric models such as affine and homography. Although a parametric model can be estimated from matching every pixel [13], and a dense correspondence can be interpolated from sparse matching [126], typically, pixel-wise displacement is obtained through pixel-wise correspondence, and parametric motion is estimated from sparse, interest point detection and matching [99]. In between the sparse and dense representation is correspondence on contours [125, 62], which has been used in tracking objects and analyzing motion for textureless objects. However, the fact that the underlying motion between scenes is complicated and unclear, and detecting contours from scenes is unreliable, leads us to seek for dense, pixel-wise correspondence for scene alignment.

Estimating dense correspondence between two images is a nontrivial problem with spatial regularity, *i.e.* the displacements (flow vectors) of neighboring pixels tend to be similar. When the feature values of the two images are close and temporally smooth, this displacement can be formulated as a continuous variable and the estimation problem is often reduced to solving PDE's using Euler-Lagrange [51, 20]. When the feature values are different, or other information such as occlusion needs to be taken into account, one can use belief propagation [40, 110] and graph cuts [16, 56] to optimize objective functions formulated on Markov random fields. The recent studies show that optimization tools such as belief propagation, tree-reweighted belief propagation and graph cuts can achieve very good local optimum for these optimization problems [114]. In [103], a dual-plane formulation is proposed to apply tree-reweighted BP to estimating optical flow fields. These advances in inference on Markov random fields provide us with optimization tools for dense scene matching.

Scene retrieval, parsing and recognition has become an important research direction to understand images at scene level [82, 118]. Scene recognition and retrieval has proposed global image representations in order to find similar images at a global level. Some of the most popular representations for scene matching are color histograms [111], texture models [43], segmented

regions [26], GIST descriptors [82], bag of words and the spatial pyramid [57], among many others. Common to all these representation is that there is no attempt in establishing precise meaningful correspondences across different image regions between the input and retrieved images. Our approach relates to the task of co-segmentation [89] which tries to segment simultaneously the common parts of an image pair and to the problem of shape matching [11] used in the context of object recognition. Our goal is to find correspondences across different across all the structures that compose a full scene and not just the elements of an object. Scenes have less defined regularities across images and therefore, the task of alignment can be more challenging.

Inspired by the recent advances in image alignment and scene parsing, we propose SIFT flow to analyze the correspondence between images from the same scene category but different instances.

4.3 The SIFT Flow Algorithm

4.3.1 Dense SIFT descriptors and visualization

SIFT is a local descriptor to characterize local gradient information [70]. In [70] SIFT descriptor is a sparse feature representation that consists of both feature extraction and detection. In this chapter, however, we only use the feature extraction component. For every pixel in an image, we divide its neighborhood (*e.g.*, 16×16, 32×32) into a 4×4 cell array, quantize the orientation into 8 bins in each cell, and obtain a $4\times4\times8=128$-dimensional vector as the SIFT representation for a pixel. We call this per-pixel SIFT descriptor *SIFT image*.

To visualize SIFT images, we compute the top three principal components of SIFT descriptors from a set of images, and then map these principal components to the principal components of the RGB space, as shown in Figure 4.2. Through projecting a 128D SIFT descriptor to a 3D subspace, we are able to compute the SIFT image from an RGB image in Figure 4.3 (a) and visualize it in (d). In this visualization, the pixels that have similar color may imply that they also share similar SIFT descriptors, and hence have similar local structures.

Notice that even though this SIFT visualization may look blurry as shown in Figure 4.3 (d), SIFT image indeed has very high spatial resolution. For two neighboring pixels lying across an image boundary, for example, their SIFT descriptors can be drastically different as the boundary information may shift from one set of cells to other cells. The visualization blurriness comes from the projection where significant amount of information is lost. This projection is only for visualization; in SIFT flow, the whole 128 dimensions are used for matching.

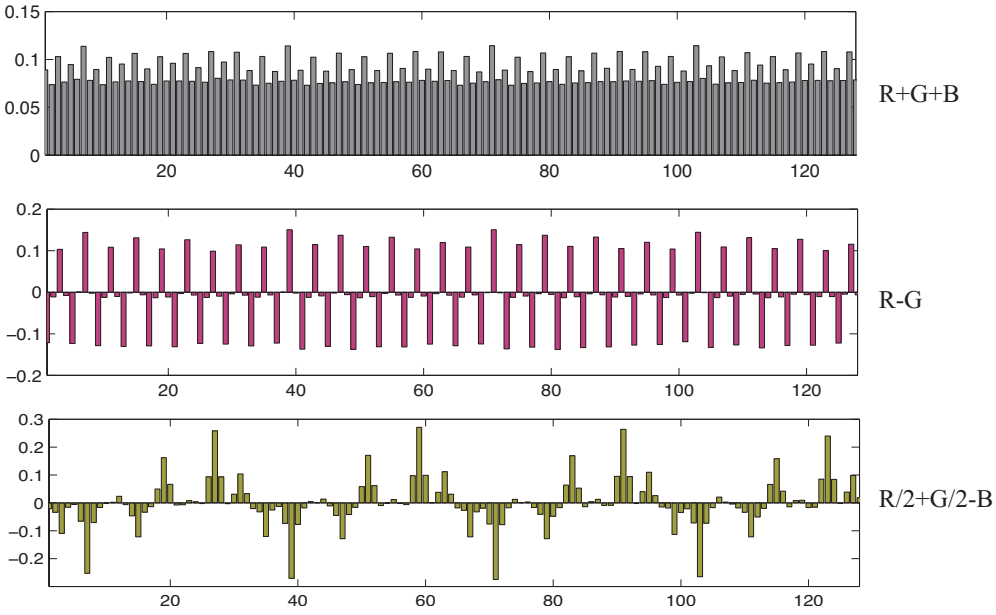


Figure 4.2. Mapping from SIFT space to RGB space. To visualize SIFT images, we compute the top three principal components of SIFT descriptors from a set of images, and then map these principal components to the principal components of the RGB space.

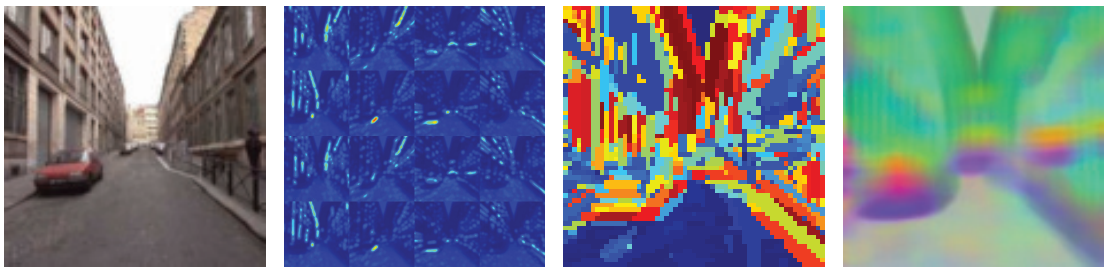


Figure 4.3. Visualization of SIFT images. We compute the SIFT descriptors on a regular dense grid. For each pixel in an image (a), the descriptor is a 128-D vector. The first 16 components are shown in (b) in a 4×4 image grid, where each component is the output of a signed oriented filter. The SIFT descriptors are quantized into visual words in (c). In order to improve the clarity of the visualization, cluster centers have been sorted according to the first principal component of the SIFT descriptor obtained from a large sample of our dataset. A visualization of SIFT image using the mapping function in Figure 4.2 is shown in (d). We will use (d) as our visualization of SIFT descriptors for the rest of the chapter.

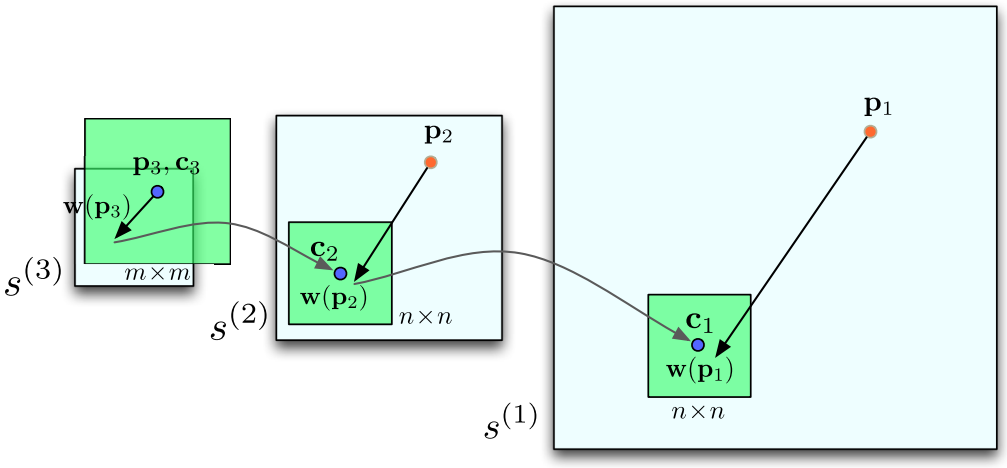


Figure 4.4. An illustration of coarse-to-fine SIFT flow matching on pyramid. The green square is the searching window for $\mathbf{p}_k$ at each pyramid level k. For simplicity only one image is shown here, where $\mathbf{p}_k$ is on image s_1, and $\mathbf{c}_k$ and $\mathbf{w}(\mathbf{p}_k)$ are on image s_2. See text for details.

Now that we have per-pixel SIFT descriptor for two images, our next task is to build dense correspondence based on these descriptors.

■ 4.3.2 Matching Objective

Similar to the brightness constancy assumption in optical flow, we assume the SIFT descriptor is constant along the flow vectors. At the same time, the flow field should be piecewise smooth to agree with object boundaries. The objective function of SIFT flow is formulated as follows. Let $\mathbf{p} = (x, y)$ contain the spatial coordinate of a pixel, and $\mathbf{w}(\mathbf{p}) = (u(\mathbf{p}), v(\mathbf{p}))$ be the flow vector at $\mathbf{p}$. Denote s_1 and s_2 as the per-pixel SIFT feature [70] for two images, and ε contains all the spatial neighborhood (a four-neighbor system is used). The energy function for SIFT

flow is defined as:

$$E(\mathbf{w}) = \sum_{\mathbf{p}} \min\left(\left\|s_1(\mathbf{p}) - s_2(\mathbf{p}+\mathbf{w}(\mathbf{p}))\right\|_1, t\right) + \sum_{\mathbf{p}} \eta\Big(|u(\mathbf{p})| + |v(\mathbf{p})|\Big) + \sum_{(\mathbf{p},\mathbf{q})\in\varepsilon} \min\Big(\alpha|u(\mathbf{p}) - u(\mathbf{q})|, d\Big) + \sum_{(\mathbf{p},\mathbf{q})\in\varepsilon} \min\Big(\alpha|v(\mathbf{p}) - v(\mathbf{q})|, d\Big). \quad (4.1)$$

In this objective function, truncated L1 norms are used in both the data term and the smoothness term to account for matching outliers and flow discontinuities, with t and d as the threshold, respectively. An L1 norm is also imposed on the magnitude of the flow vector as a bias towards smaller displacement when no other information is available. Notice that in [69] only an L1 norm is used for the data term and the small displacement biased is formulated as an L2 norm. This energy function can be directly optimized by running sequential Belief Propagation (BP-S) [114] on a dual plane setup [103].

■ 4.3.3 Coarse-to-fine matching scheme

However, directly optimizing Eqn. (4.1) may scale poorly with respect to the image size. In SIFT flow, a pixel in one image can literally match to any other pixel in another image. Suppose the image has h^2 pixels, then the time and space complexity of the BP algorithm to estimate the SIFT flow is $O(h^4)$. As reported in [69], the computation time for 145×105 images with an 80×80 searching neighborhood is 50 seconds. The original implementation of SIFT flow [69] would require more than two hours to process a pair of 256×256 images in our database with a memory usage of 16GB to store the data term.

To address the performance drawback, we designed a coarse-to-fine SIFT flow matching scheme that significantly improves the performance. The basic idea is to roughly estimate the flow at a coarse level of image grid, then gradually propagate and refine the flow from coarse to fine. The procedure is illustrated in Figure 4.4. For simplicity, we use s to represent both s_1 and s_2. A SIFT pyramid $\{s^{(k)}\}$ is established, where $s^{(1)} = s$ and $s^{(k+1)}$ is smoothed and downsampled from $s^{(k)}$. At each pyramid level k, let $\mathbf{p}_k$ be the coordinate of the pixel to match, $\mathbf{c}_k$ be the offset or centroid of the searching window, and $\mathbf{w}(\mathbf{p}_k)$ be the best match from BP. At the top pyramid level $s^{(3)}$, the searching window is centered at $\mathbf{p}_3$ ($\mathbf{c}_3 = \mathbf{p}_3$) with size $m\times m$, where m is the width (height) of $s^{(3)}$. The complexity of BP at this level is $O(m^4)$. After BP

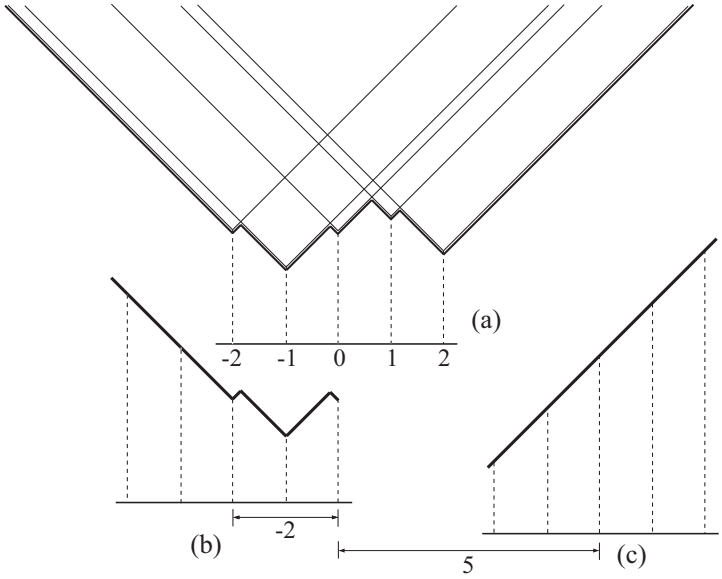


Figure 4.5. We generalized distance transform function for truncated L1 norm [36] to pass message between neighboring nodes that have different offsets (centroids) of the searching window.

converges, the system propagates the optimized flow vector $\mathbf{w}(\mathbf{p}_3)$ to the next (finer) level to be $\mathbf{c}_2$ where the searching window of $\mathbf{p}_2$ is centered. The size of this searching window is fixed to be $n \times n$ with $n = 11$. This procedure iterates from $s^{(3)}$ to $s^{(1)}$ until the flow vector $\mathbf{w}(\mathbf{p}_1)$ is estimated. The complexity of this coarse-to-fine algorithm is $O(h^2 \log h)$, a significant speed up compared to $O(h^4)$.

When the matching is propagated from an coarser level to the finer level, the searching windows for two neighboring pixels may have different offsets (centroids). We modify the the distance transform function developed for truncated L1 norm [36] to cope with this situation, with the idea illustrated in Figure 4.5. To compute the message passing from pixel $\mathbf{p}$ to its neighbor $\mathbf{q}$, we first gather all other messages and data term, and apply the routine in [36] to compute the message from $\mathbf{p}$ to $\mathbf{q}$ assuming that $\mathbf{q}$ and $\mathbf{p}$ have the same offset and range. The function is then extended to be outside the range by increasing α per step, as shown in Figure 4.5 (a). We take the function in the range that $\mathbf{q}$ is relative to $\mathbf{p}$ as the message. For example, if the offset of the searching window for $\mathbf{p}$ is 0, and the offset for $\mathbf{q}$ is 5, then the message from $\mathbf{p}$ to $\mathbf{q}$ is plotted in Figure 4.5 (c). If the offset of the searching window for $\mathbf{q}$ is -2 otherwise, the message is shown in Figure 4.5 (b).

Using the proposed coarse-to-fine matching scheme and modified distance transform func-

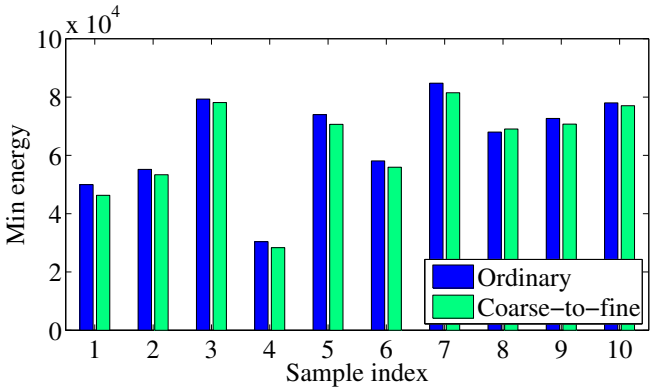


Figure 4.6. Coarse-to-fine SIFT flow not only runs significantly faster, but also achieves lower energies most of the time. In this experiment, we randomly selected 10 samples in the test set and computed the lowest energy of the best match with the nearest neighbors. We tested both the coarse-to-fine algorithm proposed in this chapter and the ordinary matching scheme in [69]. Except for sample #8, coarse-to-fine matching achieves lower energy than the ordinary matching algorithm.

tion, the matching between two 256×256 images takes 31 seconds on a workstation with two quad-core 2.67 GHz Intel Xeon CPUs and 32 GB memory, in a C++ implementation. Further speedup (up to 50x) can be achieved through GPU implementation [29] of the BP-S algorithm since this algorithm can be parallelized. We leave this as future work.

A natural question is whether the coarse-to-fine matching scheme can achieve the same minimum energy as the ordinary matching scheme (only one level without coarse-to-fine) [69]. We randomly selected 10 pairs of images to estimate SIFT flow, and check the minimum energy obtained using coarse-to-fine scheme and ordinary scheme (non coarse-to-fine), respectively. For these 256×256 images, the average running time of coarse-to-fine SIFT flow is 31 seconds, whereas it takes 127 minutes in average for the ordinary matching. The coarse-to-fine scheme not only runs significantly faster, but also achieves lower energies most of the time compared to the ordinary matching algorithm [69] as shown in Figure 4.6. This is consistent with what has been discovered in the optical flow community: coarse-to-fine search not only speeds up computation but also leads to lower energy. This can be caused by the inherent self-similarity nature of SIFT features across scales: the correspondence at a coarser level is a good prediction for the correspondence at a finer level.

■ 4.3.4 Neighborhood of SIFT flow

In theory, we can apply optical flow to two arbitrary images to estimate a correspondence, but we may not get anything meaningful if the two images come from two different videos. In fact, even when we apply optical flow to two adjacent frames in a video sequence, we assume dense sampling in time so that there is significant overlap between two neighboring frames. Similarly, in SIFT flow we define the neighborhood of an image as the nearest neighbors when we query a large database with the input. Ideally, if the database is large enough to contain almost every possible image in the world, the nearest neighbors will be visually similar to the query image. This motivates the following analogy with optical flow, where correspondence is sought between temporally adjacent (and thus visually similar) video frames:

$$\begin{aligned} \text{Dense sampling in time} &: \text{optical flow} :: \\ \text{Dense sampling in the space of all images} &: \text{SIFT flow} \end{aligned}$$

In other words, as optical flow assumes dense sampling of the time domain to enable tracking, SIFT flow assumes dense sampling in (some portion of) the space of natural images to enable scene alignment. In order to make this analogy possible, we collect a large database consisting of 102,206 frames from 731 videos. Analogous to the time domain, we define the "temporal frames" to a query image as the N nearest neighbors in this database. The SIFT flow is then established between the query image and the N nearest neighbors. How to obtain nearest neighbors will be discussed in the next subsection.

■ 4.3.5 Scene matching with histogram intersection

We use a fast indexing technique in order to retrieve nearest neighbors that will be further aligned using the SIFT flow algorithm to match the query image. As a fast search we use spatial histogram matching of quantized SIFT [70] features [57]. First, we build a dictionary of 500 visual words [107] by running K-means on 5000 SIFT descriptors randomly selected out of all the video frames in our dataset. Then, the visual words are binned using a two level spatial pyramid [57, 44].

The similarity between two images is measured by the histogram intersection. For each input image, we select the top 20 nearest neighbors. Matching is performed on all the frames from all the videos in our dataset. There is at most one best matching frame from each video. We then apply SIFT flow between the input image and the top 20 candidate neighbors and re-rank the neighbors based on the alignment score. This approach is well matched to the

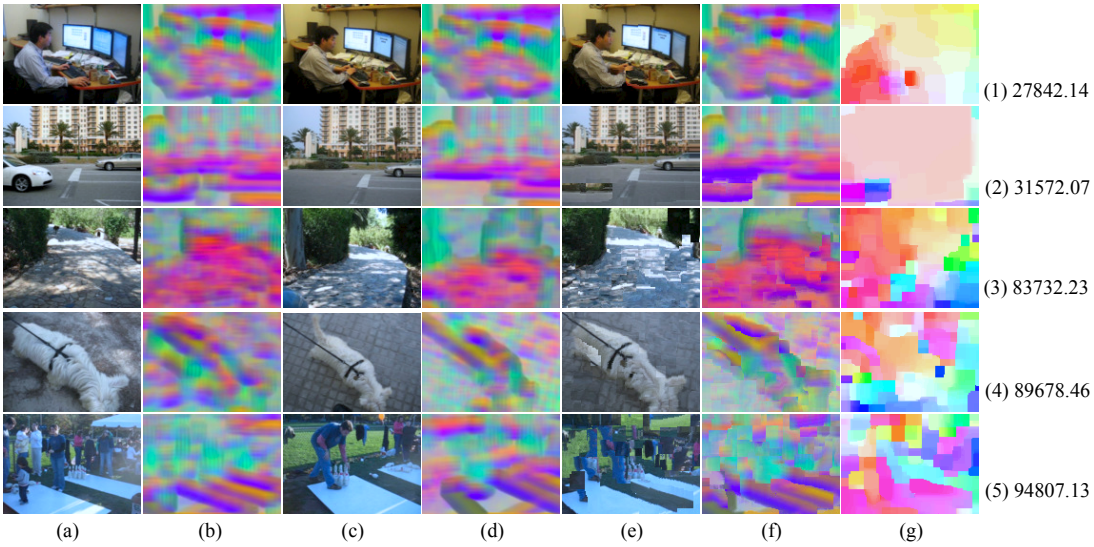


Figure 4.7. SIFT flow for image pairs depicting the same scene/object. (a) shows the query image and (b) its densely extracted SIFT descriptors. (c) and (d) show the best (lowest energy) match from the database and its SIFT descriptors, respectively. (e) shows (c) warped onto (a). (f) shows the warped SIFT image (d). (g) shows the estimated displacement field with the minimum alignment energy shown to the right.

similarity obtained by SIFT flow as it uses the same basic features (SIFT descriptors) and spatial information is loosely represented (by means of the spatial histograms).

Other scene metrics such as GIST [82] can also be used for retrieving nearest neighbors [68]. It has been reported that various nearest matching algorithms do not result in much difference in obtaining the nearest neighbors for matching [90].

■ 4.4 Experiments on Video Retrieval

■ 4.4.1 Results of video retrieval

We conducted several experiments to test the SIFT flow algorithm on our video database. One frame from each of the 731 videos was selected as the query image and histogram intersection matching was used to find its 20 nearest neighbors, excluding all other frames from the query video. The SIFT flow algorithm was then used to estimate the dense correspondence (represented as a pixel displacement field) between the query image and each of its neighbors. The

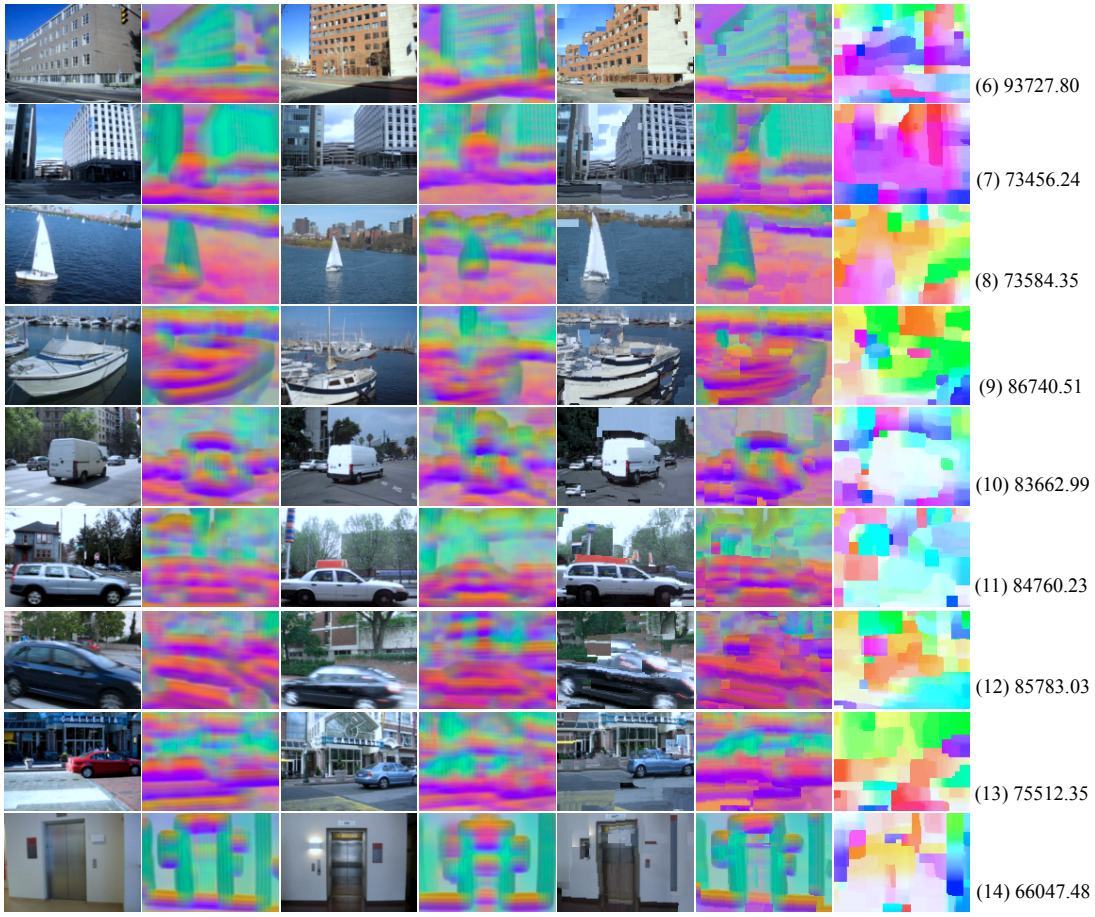


Figure 4.8. SIFT flow computed for image pairs depicting the same scene/object category where the visual correspondence is obvious.

best matches are the ones with the minimum energy defined by (4.1). Alignment examples are shown in Figure 4.7–4.9. The original query image and its extracted SIFT descriptors are shown in columns (a) and (b). The minimum energy match (out of the 20 nearest neighbors) and its extracted SIFT descriptors are shown in columns (c) and (d). To investigate the quality of the pixel displacement field, we use the computed displacements to warp the best match onto the query image. The warped image and warped SIFT descriptor image are shown in columns (e) and (f). The visual similarity between (a) and (e), and (b) and (f) demonstrates the quality of the matching. Finally, the displacement field is visualized using color-coding adapted from [7]

Figure 4.9. SIFT flow for challenging examples where the correspondence is not obvious.

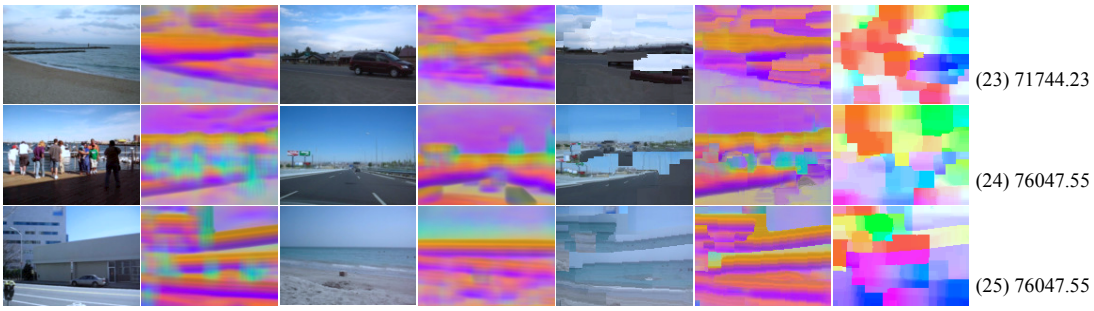


Figure 4.10. Some failure examples with semantically incorrect correspondences. Although the minimized SIFT flow objectives are low for these samples (compared to those in Figure 4.9), the query images are rare in the database and the best SIFT flow matches do not belong to the same scene category as the queries. However, these failures can be overcome through increasing the size of the database.

Figure 4.11. Alignment typically improves ranking of the nearest neighbors. Images enclosed by the red rectangle are the top 10 nearest neighbors found by histogram intersection, displayed in a scan-line order (left to right, top to bottom). Images enclosed by the green rectangle are the top 10 nearest neighbors ranked by the minimum energy obtained by the alignment algorithm. The warped nearest neighbor image is displayed to the right of the original image. Note how the returned images are re-ranked according to the size of the depicted vehicle by matching the size of the bus in the query.

in column (g) with the minimum alignment energy shown to the right.

Figure 4.7 shows examples of matches between frames coming from exactly the same scene, but different video sequence. The almost perfect matching in (1) and (2) demonstrates that SIFT flow reduces to classical optical flow when the two images are temporally adjacent frames in a video sequence. In (3)–(5), the query and the best match are more distant within the video sequence, but the alignment algorithm can still match them reasonably well.

Figure 4.8 shows more challenging examples, where the two frames come from different videos while containing the same type of objects. The alignment algorithm attempts to match the query image by transforming the candidate image. Note the significant changes in viewpoint between the query and the match in examples (8), (9), (11), (13), (14) and (16). Note also that some discontinuities in the flow field are caused by errors in SIFT matching. The square shaped discontinuities are a consequence of the decoupled regularizer on the horizontal and vertical components of the pixel displacement vector.

Figure 4.9 shows alignment results for examples with no obvious visual correspondence. Despite the lack of direct visual correspondence, the scene alignment algorithm attempts to rebuild the house (17), change the shape of the door into a circle (18) or reshuffle boats (20).

Some failure cases are shown in Figure 4.10. Typically, these are caused by the lack of visually similar images in the video database. Note that, typically, alignment improves ranking of the K-nearest neighbors. This is illustrated in Figure 4.11.

■ 4.4.2 Evaluation of the dense scene alignment

After showing some examples of scene alignment, we want to evaluate how well SIFT flow performs in matching structures across different images and how it compares with human selected matches. Traditional optical flow is a well-defined problem and it is straightforward for humans to annotate motion for evaluation [63]. In the case of SIFT flow, however, there may not be obvious or unique best pixel-to-pixel matching as the two images may contain different objects, or the same object categories with very different instances.

To evaluate the matching obtained by SIFT flow, we performed a user study where we showed 11 users image pairs with preselected sparse points in the first image and asked the users to select the corresponding points in the second image. This process is explained in Figure 4.12. As shown on the right of Fig. 4.13 , user annotation can be ambiguous. Therefore, we use the following metric to evaluate SIFT flow: for a pixel $\mathbf{p}$, we have several human annotations $\mathbf{z}_i$ as its flow vector, and $\mathbf{w}(\mathbf{p})$ as the estimated SIFT flow vector. We compute $\Pr\big(\exists \mathbf{z}_i, \|\mathbf{z}_i - \mathbf{w}(\mathbf{p})\| \leq r | r\big)$, namely the probability of one human annotated flow is within distance r to SIFT flow $\mathbf{w}(\mathbf{p})$. This function of r is plotted on the left of Fig. 4.13 (red curve). For comparison, we plot the same probability function (blue curve) for minimum L1-norm SIFT matching, *i.e.* SIFT flow matching without spatial terms. Clearly SIFT flow matches better to human annotation than minimum L1-norm SIFT matching.

■ 4.5 Applications

In this section we demonstrate two applications for the proposed scene matching algorithm: (1) motion field prediction from a single image using motion priors, and (2) motion synthesis via transfer of moving objects common in similar scenes.

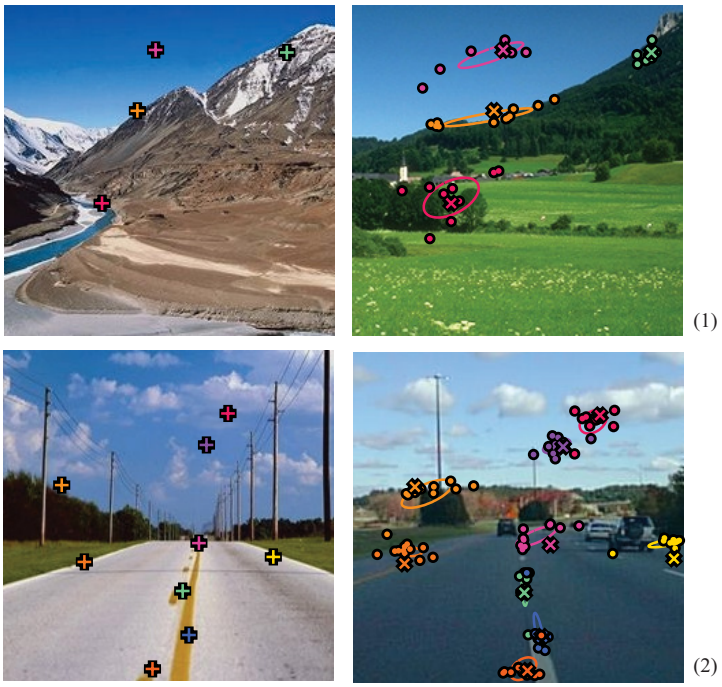


Figure 4.12. For an image pair such as row (1) or row (2), a user defines several sparse points in (a) as "+". The human annotated matchings are marked as dot in (b), from which a Gaussian distribution is estimated and displayed as an ellipse. The correspondence estimated from SIFT flow is marked as "×" in (b).

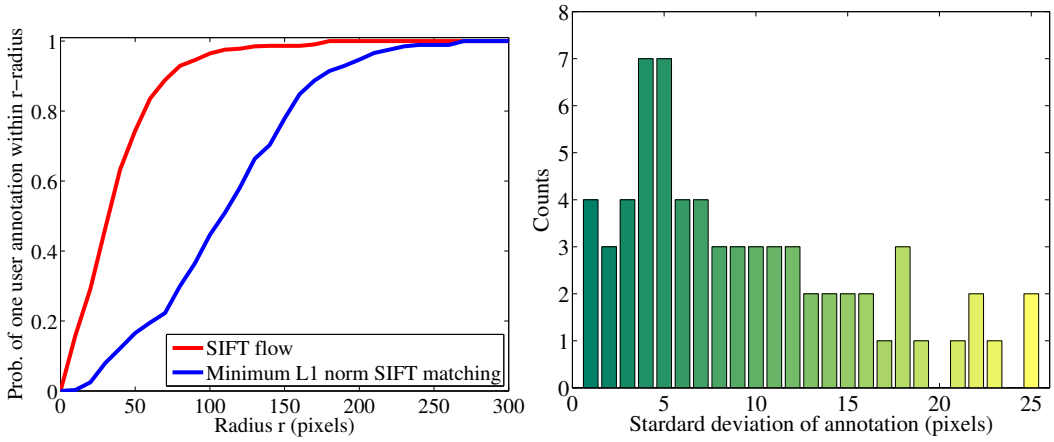


Figure 4.13. The evaluation of SIFT flow using human annotation. Left: the probability of one human annotated flow lies within r distance to the SIFT flow as a function of r (red curve). For comparison, we plot the same probability for direct minimum L1-norm matching (blue curve). Clearly SIFT flow matches human perception better. Right: the histogram of the standard deviation of human annotation. Human perception of scene correspondence varies from subject to subject.

■ 4.5.1 Predicting motion fields from a single image

The goal is, given a single static image, to predict what motions are plausible in the image. This is similar to the recognition problem, but instead of assigning labels to each pixel, we want to assign possible motions.

We built a scene retrieval infrastructure to query still images over a database of videos containing common moving objects. The database consists of sequences depicting common events, such as cars driving through a street and kids playing in a park. Each individual frame was stored as a vector of word-quantized SIFT features, as described in Sect. 4.3.5. In addition, we store the temporal motion field estimated using [22] between every two consecutive frames of each video.

We compare two approaches for predicting the motion field for the query still image. The first approach consists of directly transferring the motion of the closest video frame matched in the database. Using the SIFT-based histogram matching in Sect. 4.3.5, we can retrieve very similar video frames that are roughly spatially aligned. For common events such as cars moving forward on a street, the motion prediction can be quite accurate given enough samples in the database. The second approach refines the coarse motion prediction described above using the dense correspondences obtained by the alignment algorithm Sect. 4.3.3). In particular, we compute the SIFT flow from the retrieved video frame to the query image and use the computed correspondence to warp the temporally estimated motion of the retrieved video frame. Figure 4.17 shows examples of predicted motion fields directly transferred from the top 5 database matches and the warped motion fields. Note that in simple cases the direct transfer is already quite accurate and the warping results only in minor refinements.

While there are many improbable flow fields (*e.g.*, a car moving upwards), each image can have multiple plausible motions : a car or a boat can move forward, in reverse, turn, or remain static. In any scene the camera motion can generate motion field over the entire frame and objects can be moving at different velocities. Figure 4.14 shows an example of 5 motion fields predicted using our video database. Note that all the motions fields are different, but plausible.

■ 4.5.2 Quantitative evaluation

Due to the inherent ambiguity of multiple plausible motions for each still image, we design the following procedure for quantitative evaluation. For each test video, we randomly select a test frame and obtain a *result* set of top n inferred motion fields using our motion prediction method. Separately, we collect an *evaluation* set containing the temporally estimated motion

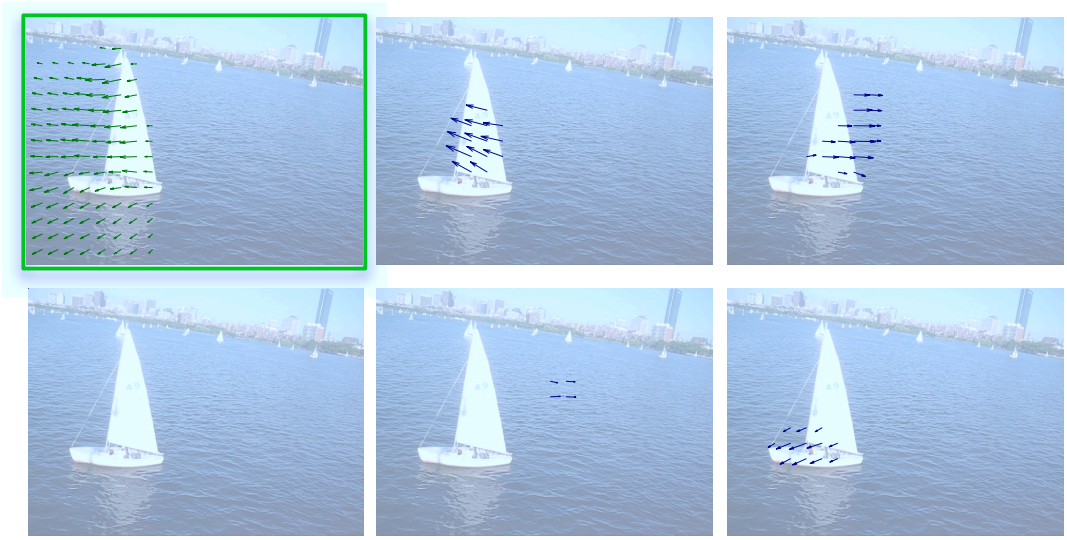


Figure 4.14. Multiple motion field candidates. A still query image with its temporally estimated motion field (in the green frame) and multiple motion fields predicted by motion transfer from a large video database.

(from video) for the test frame (the closest to a ground truth we have) and 11 random motion fields taken from other scenes in our database, acting as distracters. We take each of the n inferred motion fields from the result set and compute their similarity (defined below) to the set of evaluation fields. The rank of the ground truth motion with respect to the random distracter motions is an indicator of how close the predicted motion is to the true motion estimated from the video sequence. Because there are many possible motions that are still realistic, we do this comparison with each of the top n motion fields within the result set and keep the highest ranking achieved. Finally, we repeat this evaluation ten times with a different randomly selected test frame for each test video and report the median of the rank score across the different trials.

For this evaluation, we represent each motion field as a regular two dimensional motion grid filled with 1s where there is motion and 0 otherwise. The similarity between two motion fields is defined then as

$$S(\mathbf{M},\mathbf{N}) \stackrel{\text{def}}{=} \sum_{(x,y)\in G} \Big(\mathbf{M}(x,y) = \mathbf{N}(x,y)\Big) \tag{4.2}$$

where $\mathbf{M}$ and $\mathbf{N}$ are two rectangular motion grids of the same size, and (x,y) is a coordinate pair within the spatial domain G of grids $\mathbf{M}$ and $\mathbf{N}$.

Figure 4.15 (a) shows the normalized histogram of these rankings across 720 predicted

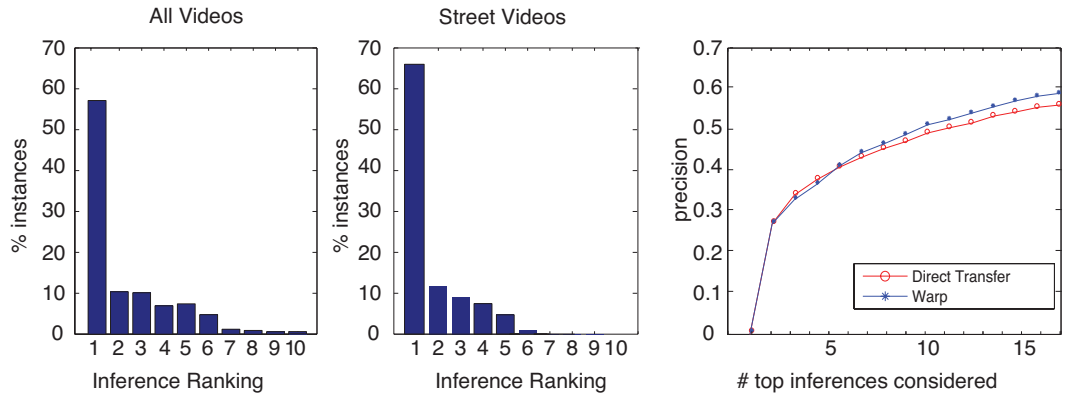


Figure 4.15. Evaluation of motion prediction. (a) and (b) show normalized histograms of prediction rankings (result set size of 15). (c) shows the ranking precision as a function of the result set size.

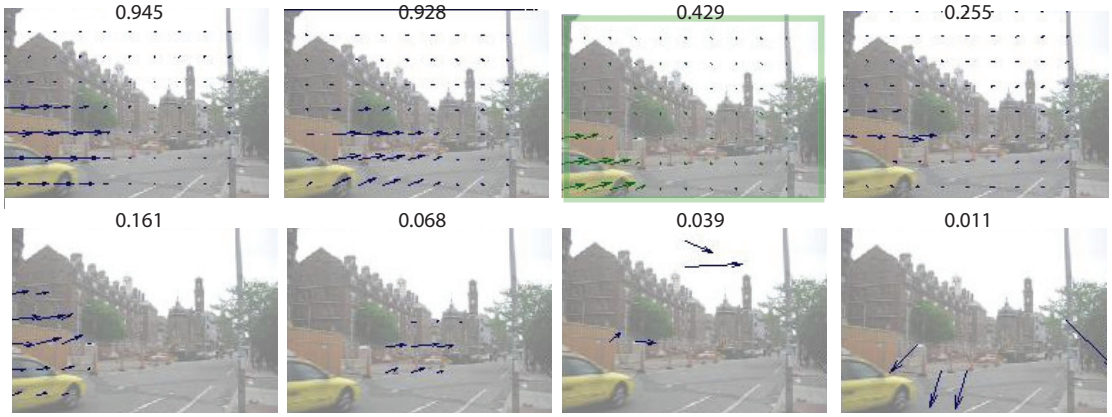
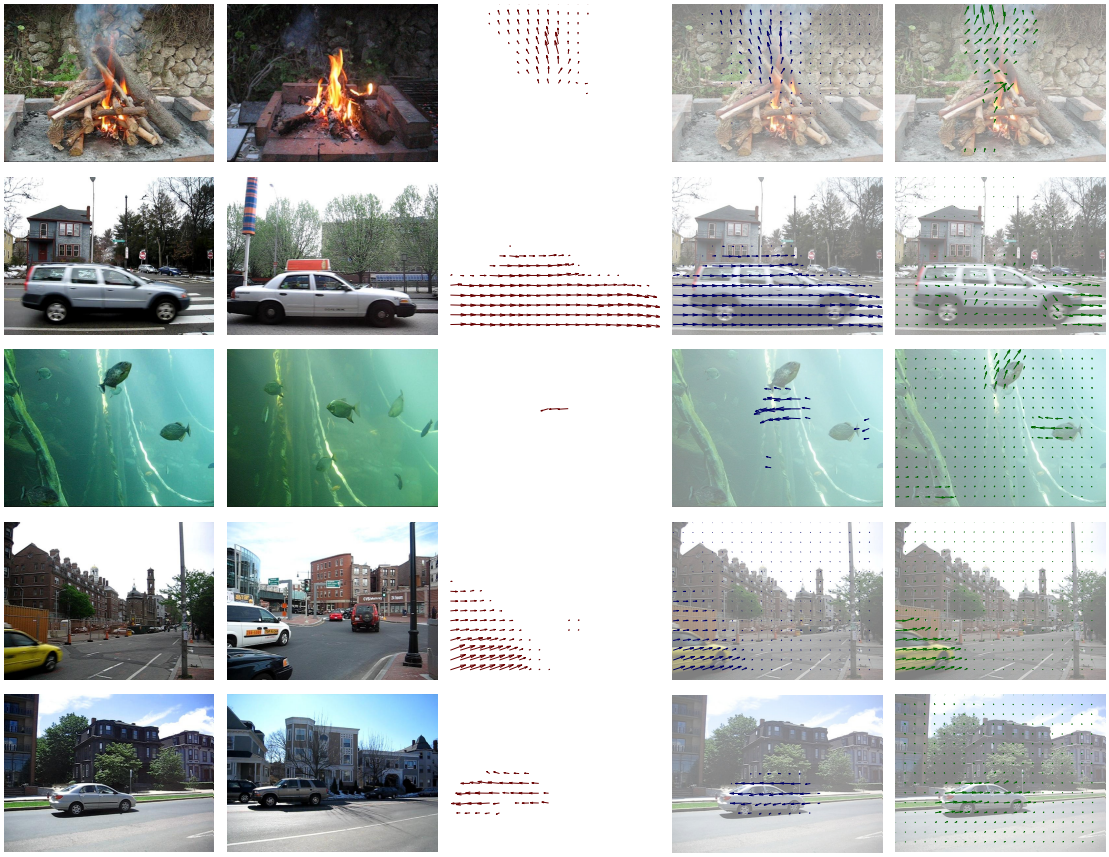


Figure 4.16. Motion instances where the predicted motion was not ranked closest to the ground truth. A set of random motion fields (blue) together with the predicted motion field (green, ranked 3rd). The number above each image represents the fraction of the pixels that were correctly matched by comparing the motion against the ground truth. In this case, some random motion fields appear closer to the ground truth than our prediction (green). However, our prediction also represents a plausible motion for this scene.

motion fields from our video data set. Figure 4.15 (b) shows the same evaluation on a subset of the data that includes 400 videos with mostly streets and cars. Notice how, for more than half of the scenes, the inferred motion field is ranked the first suggesting a close match to the temporally-estimated ground truth. Most other test examples are ranked within the top 5. Focusing on roads and cars gives even better results with 66% of test trials ranked 1st and even



(a) (b) (c) (d) (e)

Figure 4.17. Motion from a single image. The (a) original image, (b) matched frame from the video data set, (c) motion of (b), (d) warped and transferred motion field from (b), and (e) ground truth for (a). Note that the predicted motion in (d) is inferred from a single input still image, i.e. no motion signal is available to the algorithm. The predicted motion is based on the motion present in other videos with image content similar to the query image.

more test examples ranked within the top 5. Figure 4.15 (c) shows the precision of the inferred motion (the percentage of test examples with rank 1) as a function of the size of the result set, comparing (i) direct motion field transfer (red circles) and (ii) warped motion field transfer using SIFT flow (blue stars).

While histograms of ranks show that the majority of the inferred motions were ranked 1st, there are still a significant number of instances with lower rank. Figure 4.16 shows a false negative example, where the inferred motion field was not ranked top despite the reasonable output. Notice how the top ranked distracter fields are quite similar to our prediction showing

Figure 4.18. Motion synthesis via object transfer. Query images (a), the top video match (b), and representative frames from the synthesized sequence (c) obtained by transferring the moving objects from the video to the still query image.

that, in some cases, where our prediction is not ranked the first, we still produce realistic motion.

■ 4.5.3 Motion synthesis via object transfer

We described above how to predict the direction and velocity of objects in a still image. Having a prior on what scenes look like over time also allows us to infer what objects (that might not be part of the still image) can possibly appear. For example, a car moving forward can appear in a street scene with an empty road; or a fish can start swimming in a fish tank scene.

Based on this idea, we propose a method for synthesizing motion from a still image. The goal is to transfer moving objects from similar video scenes to a static image. In particular, given a still image q that is not part of any video in our database D, we identify and transfer moving objects from videos in D into q as follows:

1. Query D using the SIFT-based scene matching algorithm to retrieve the set of closest video frame matches $F = \{f_i | f_i$ is the ith frame from a video in $D\}$ given the query image q.

2. For each frame $f_i \in F$, we can synthesize a video sequence based on the still image q. The kth frame of the synthesized video is generated as follows:

(a) Densely sample the motion from frame f_{i+k} to f_{i+k+1}

(b) Construct frame q_k by transferring non-moving pixels from q and moving pixels from f_{i+k}.

(c) Apply poisson editing [83] to blend the foreground (pixels from f_{i+k}) into the background composed of pixels from q.

Figure 4.18 shows examples of still query images, their corresponding retrieved video sequences from our database, and representative frames from the synthesized video sequence created by transferring the moving objects from the video sequence into the still image. Notice the variety of region sizes transferred and the seamless integration of objects into the new scenes.

Some of the biggest challenges in creating realistic composites lie in estimating the correct size and orientation of the objects to introduce in the scene. Our framework inherently takes care of these constraints by retrieving sequences that are visually similar to the query image. This enables creating realistic motion sequences from still images using a straightforward transfer of moving objects.

■ 4.6 Experiments on Image Alignment and Face Recognition

We have demonstrated that it is possible to establish semantically meaningful matches between images across scenes through matching salient local image structure SIFT. In this section, we apply SIFT flow to traditional image alignment problems, and demonstrate that SIFT flow is also able to handle challenging image registration and recognition problems in these areas.

■ 4.6.1 Image registration of the same scene

Image registration of the same physical scene can be a very challenging problem when there is little overlap between two images, or local appearances change drastically from one image to the other due to the change of season, different imaging conditions (angle, lighting, sensor), geographical deformations, and human activities such as new buildings and destruction by wars. Sparse feature detection and matching is the standard way for image matching problem in this category [130], but we want to see how SIFT flow would perform even though it is not specifically designed for this task.

Let us first look at two satellite images[2] of the same location on Mars as shown in Figure

[2]Image source: http://www.msss.com/mars_images/moc/2006/12/06/gullies/sirenum_crater/index.html

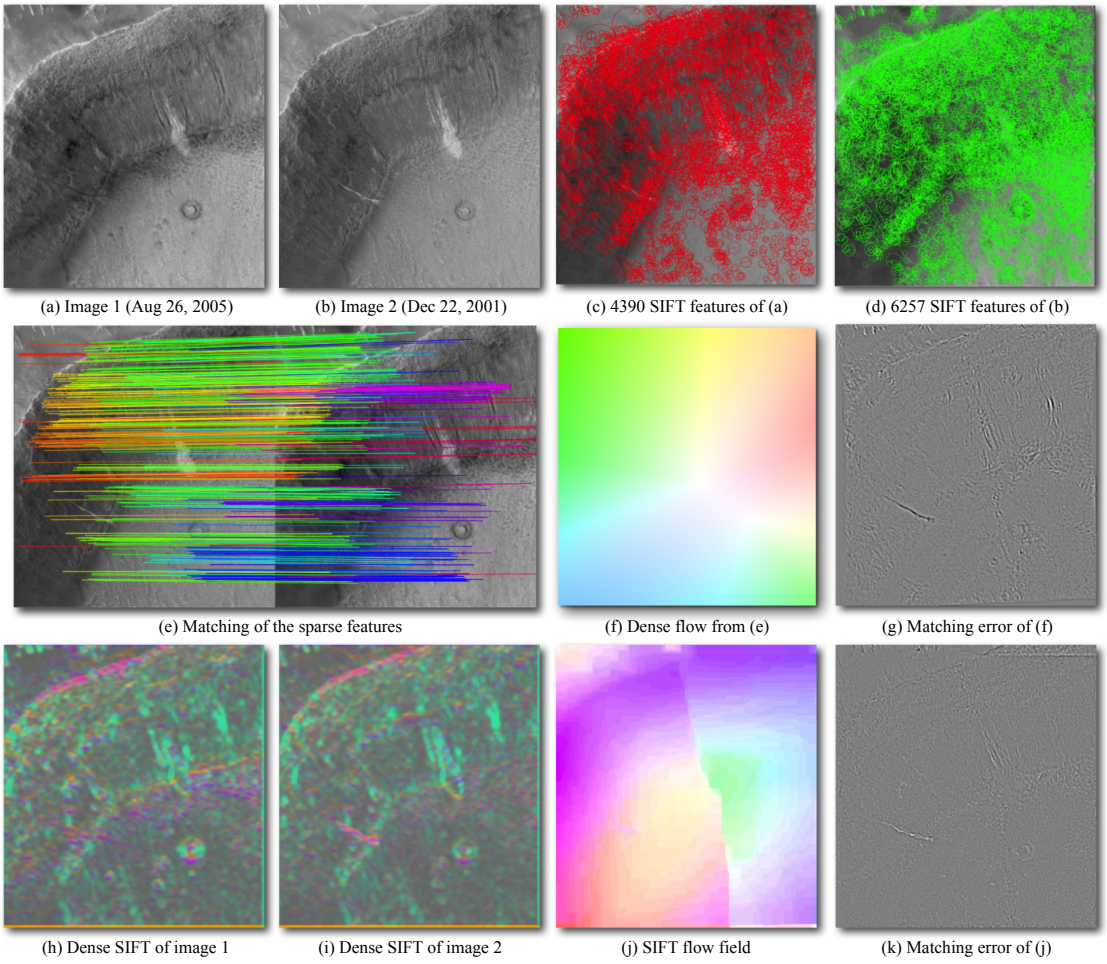


Figure 4.19. SIFT flow can be applied to aligning satellite images. The two Mars satellite images (a) and (b) are taken at four years apart with different local appearances. The results of sparse feature detection and matching are shown in (c) to (g), whereas the results of SIFT flow are displayed in (h) to (k). The mean absolute error of the sparse feature approach is 0.030, while the mean absolute error of SIFT flow is 0.021, significantly lower. Visit webpage http://people.csail.mit.edu/celiu/SIFTflow/NGA/ for the animations of the warping.

4.19 (a) and (b). Because they were taken four years apart, the intensities and even features vary between the two images. We first used sparse SIFT feature detection [70] to detect SIFT feature points on both images ((c) and (d)), and a sparse correspondence is established through minimum SSD matching (e). This sparse correspondence is further interpolated to form a dense

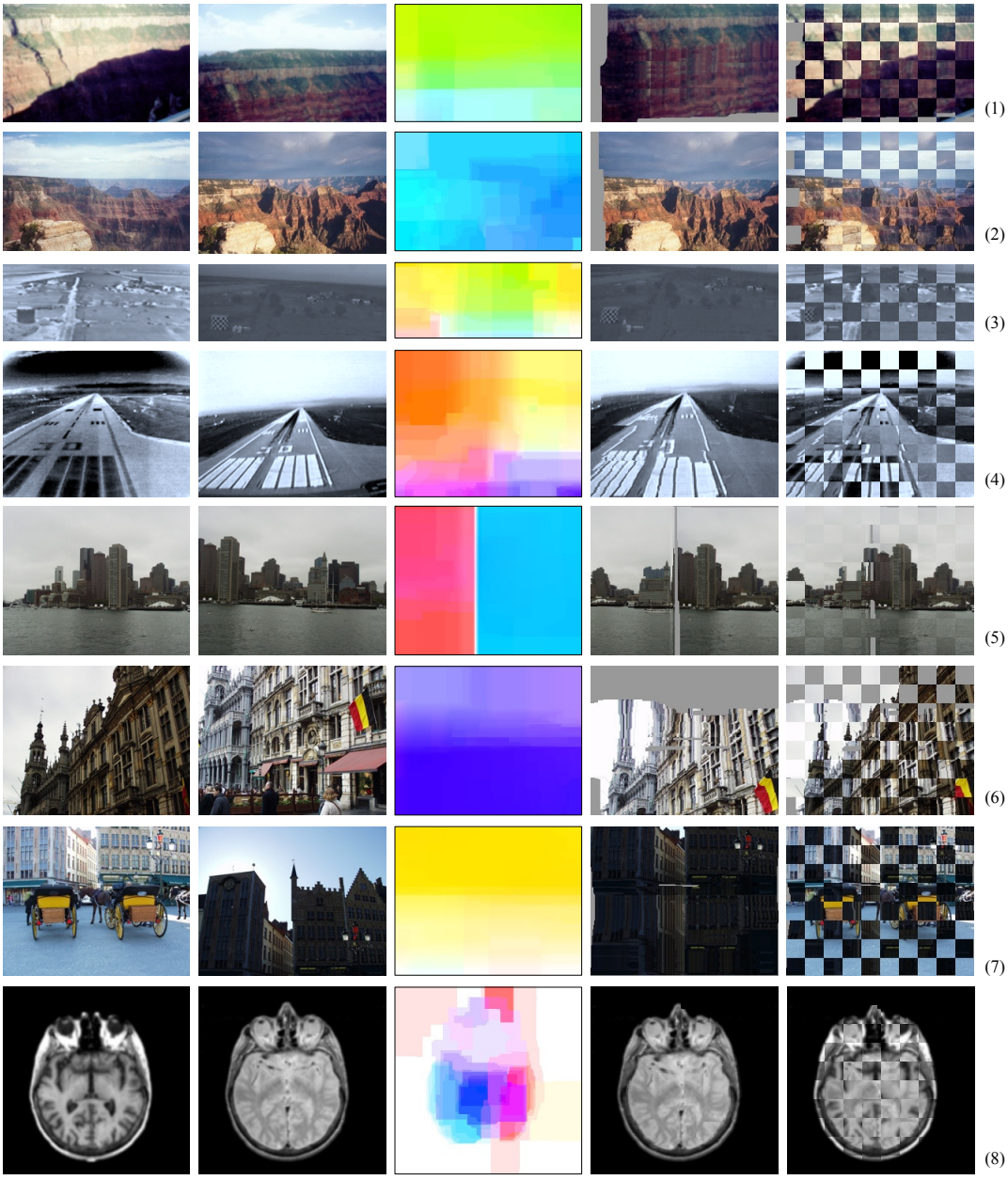


Figure 4.20. SIFT flow can be applied to image registration of the same scene but under different lighting and imaging conditions. Column (a) and (b) are some examples from [130]. Even though originally designed for scene alignment, SIFT flow is also able to align these challenging pairs. Visit webpage http://people.csail.mit.edu/celiu/SIFTflow/NGA/ for the animations of the warping.

Figure 4.21. SIFT flow can serve as a tool to account for pose, expression and lighting changes for face recognition. (a): Ten samples of one subject in ORL database [93]. Notice pose and expression variations of these samples. (b): We select the first image as the query, apply SIFT flow to aligning the rest of the images to the query, and display the warped images with respect to the dense correspondence. The poses and expressions are rectified to that of the query after the warping. (c): The same as (b) except for choosing the fifth sample as the query.

flow field as shown in (f). To understand how good this interpolated dense correspondence is, we warp image 2 to image 1 according to the dense flow field and display the difference between image 1 and warped image 2 in (g). The mean absolute error of this correspondence is 0.030 (the pixel value is between 0 and 1). Clearly, the structural correspondence between the two Mars images are not captured in (g). We then applied SIFT flow to aligning these two images. The SIFT flow field is displayed in (j), and the difference between image 1 and warped image 2 according to the SIFT flow field is displayed in (k). The mean absolute error decreases to 0.021 for SIFT flow, and visually we can see that misalignment has been significantly reduced. To our surprise, there is a fracture in the estimated SIFT flow field in (j). This could be caused by an underlying geographical deformation over the four years, or simply a stitching artifact of satellite images.

We further applied SIFT flow to aligning some challenging examples in [130] (the algorithm proposed in [130] is able to handle these examples well) and the results are displayed in Figure 4.20, where column (1) and (2) are pairs of images to align. The correspondences between some pairs, *e.g.*, row (1), (3), and (7) are even not obvious to human visual system. The dense correspondences estimated from SIFT flow are displayed in column (c). For visualization we again warp image 2 to image 1 according to the flow field and display the warped image 2 in column (d). To inspect the quality of the flow, we superimpose warped image 2 to image 1 on

a checkerboard, as shown in column (e). From these results it is clear that SIFT flow is able handle these image registration problems despite drastically different local image appearances and large displacement.

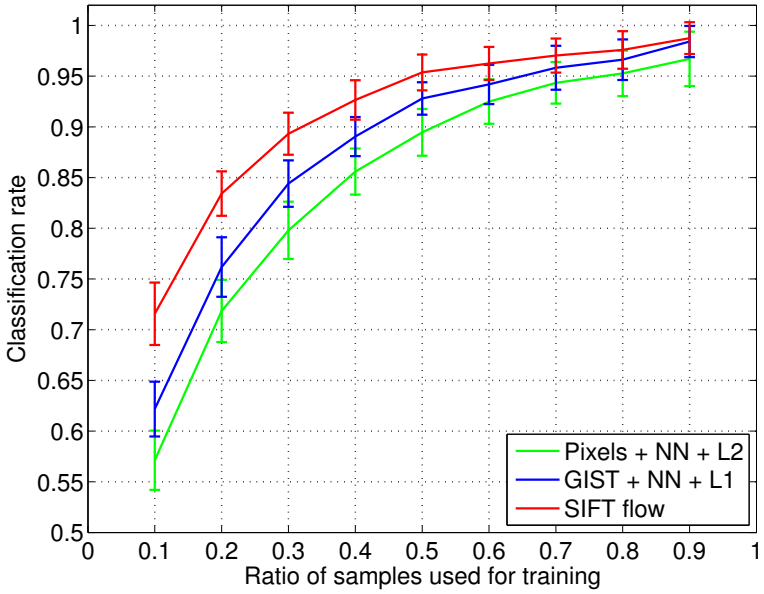
■ 4.6.2 Face recognition

Face recognition has been a key application of computer vision and pattern recognition. It has been integrated in consumer-end applications such as Apple iPhoto and Google Picassa. Face recognition turns to be a challenging problem when there are large pose and lighting variations for a large corpus of subjects. We apply SIFT flow for face recognition to see how image alignment can address classical pattern recognition problems.

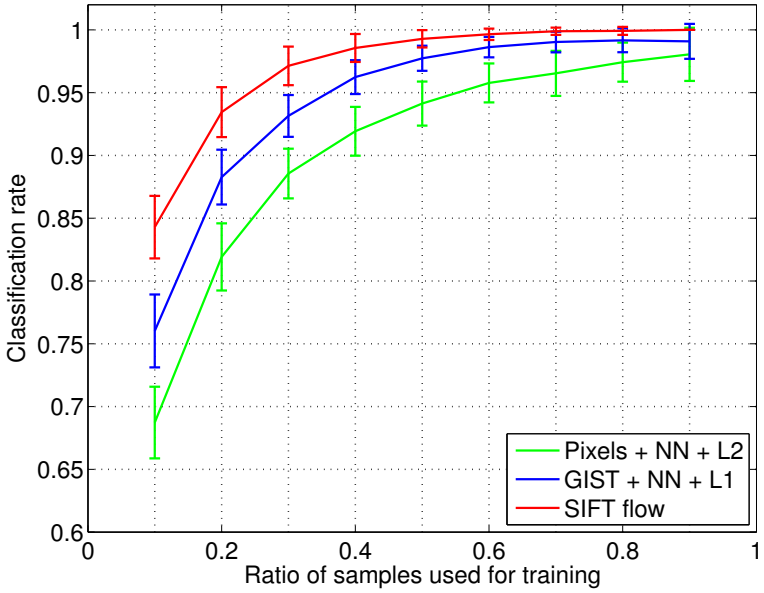
We use the ORL database [93], which contains 40 subjects and 10 images for each subject with some pose and expression variations, as the data for the experiment. In Fig. 4.21, a female sample is selected as an example to demonstrate how dense registration can deal with pose and expression variations. We first select the first image as the query, apply SIFT flow to aligning the rest of the images to the query, and display the warped images with respect to the estimated dense correspondence. As shown in Fig. 4.21 (b), the poses and expressions are rectified to that of the query after the warping. Distances established amongst the samples in (b) would exclude pose and expression variations, which may play an important factors for the distances computed amongst the samples in (a). We can also choose a different sample as query and align the rest of the images to this query, as demonstrated in (c).

In order to compare to the state-of-the-art, we conducted experiments for both original size and downsampled 32×32 images. We random split γ ($\gamma \in (0, 1)$) portion of the samples for each subject for training, and the rest $1-\gamma$ is used for test. For a test image, we first retrieve the top nearest neighbors (maximum 20) from the training database using GIST matching [82], and then apply SIFT flow to find the dense correspondence from the test to each of its nearest neighbors by optimizing the objective function in Eqn. (4.1). We assign the subject identity associate with the best match, *i.e.* the match with the minimum matching objective, to the test image. This is essentially a nearest neighbor approach using SIFT flow score as the distance metric for face recognition.

The experimental results are shown in Figure 4.22. We use the nearest neighbor classifier based on the Euclidian distance (Pixels + NN + L2) and nearest neighbor classifier using the L1-norm distance between GIST features (GIST + NN + L1) as the benchmark. Clearly, GIST features outperform raw pixel values since GIST feature is invariant to lighting changes. SIFT flow further improves the performance as SIFT flow is able to align images across different



(a) 32×32



(b) 92×112

Figure 4.22. SIFT flow is applied for face recognition. The curves in (a) and (b) are the performance plots for low-res and high-res images in the ORL face database, respectively. SIFT flow significantly boosted the recognition rate especially when there are not enough training samples.

poses. We observe that SIFT flow boosts the classification rate significantly especially when there are not enough samples, *e.g.*, the ratio of training sample is 0.1 or 0.2. We compare the

performance of SIFT flow and the state-of-the-art [24] for few training samples in Table 4.1. Clearly, SIFT flow outperforms the state-of-the-art when there are only one or two samples for training because SIFT flow is able to handle pose variation through establishing dense correspondence between different poses.

Test errors	1 Train	2 Train	3 Train
S-LDA [24]	N/A	17.1 ± 2.7	8.1 ± 1.8
SIFT flow	28.4 ± 3.0	16.6 ± 2.2	8.9 ± 2.1

Table 4.1. SIFT flow outperforms the state-of-the-art [24] when there are only few (one or two) training samples.

■ 4.7 Discussions

There have been two schools for image alignment, dense or sparse correspondence. In the sparse representation, images are summarized as feature points such as Harris corners [48], SIFT [70], and many others [99]. Correspondence is then established through matching these feature points. The algorithms based on the sparse representations are normally efficient, and are able to handle large displacements. In the dense representation, however, correspondence is established at pixel levels[3] in the two images, *e.g.*, optical flow field. Because of the spatial regularity, it is nontrivial to estimate a flow field from two images. There has been tremendous work since the early explorations [51, 71]; and there are advances in both continuous [20] and discrete domains [103]. Yet the advantage of the dense representation is the potential for image warping and computing image-domain metrics (for recognition).

SIFT flow inherits the characteristics of the dense representation by obtaining pixel-to-pixel correspondences, and at the same time inherits the scale, lighting, and pose-invariant feature of SIFT. We have demonstrated that SIFT flow is able to align very different images across scenes through scene retrieval and satellite image alignment, where the images contain objects of different appearances. Traditional dense correspondence models such as optical flow often fail to capture correspondences for this scenario. On the other hand, the dense representation of

[3] It does not mean the correspondence is *bijective* (nor *injective* or *surjective*). In the flow field representation, there is a flow vector for every pixel in image A indicating which pixel it corresponds to in image B.

SIFT flow is able to warp the information from the target image to the source, *e.g.*, predicting motion fields from a single image, and this provides a rich set of applications for image parsing [68]. Moreover, SIFT flow is able to discover a crack in satellite images of Mars by minimizing the SIFT flow objective function that contains truncated L1 norms for spatial regularity. These results can be hardly achieved from sparse representations.

An important direction for improving the algorithm is speed. The current system cannot be used for real-time image or video retrieval and matching. One direction is the GPU implementation [29] of the BP-S algorithm, which can get up to 50x speedup. However, we feel that there could be essential speedup from the sparse matching. The bottleneck of SIFT flow is the large search window size as the locations of objects may change drastically from one image to the other. The sparse, independent matching provides good, approximate matching for sparse points, and this correspondence can be propagated by abiding by the spatial regularities.

■ 4.8 Conclusion

We introduced the concept of scene alignment: to find the dense correspondence between images across scenes. We proposed SIFT flow to match salient local image structure SIFT under the conventional optical flow framework, and demonstrated that SIFT flow is able to establish semantically meaningful correspondence across complex scenes despite significant differences in appearance and spatial layout of matched images. Because the search space in SIFT flow is much larger than the search space in optical flow, we designed a coarse-to-fine matching scheme for matching large-size images.

We have demonstrated SIFT flow in video retrieval, motion estimation from a single image and video synthesis via transferring moving objects with the support of large databases. We further applied SIFT flow to traditional image alignment problems such as satellite image registration and face recognition. The preliminary success on these experiments suggested that scene alignment techniques such as SIFT flow would be useful tools for various applications in computer vision and computer graphics.

Chapter 5

Nonparametric Scene Parsing via Dense Scene Alignment

In Chapter 4, we have introduced SIFT flow, a dense scene alignment algorithm to align images across scenes. We showed many new applications, such as motion estimation from a single image and video synthesis via transferring moving objects, both with the support of large databases. In this chapter, we want to apply dense scene alignment to scene parsing.

5.1 Introduction

Scene parsing, or recognizing and segmenting objects in an image, is one of the core problems of computer vision. Traditional approaches to object recognition begin by specifying an object model, such as template matching [121, 31], constellations [37, 35], bags of features [107, 57, 44, 109], or shape models [10, 12, 34], etc. These approaches typically work with a fixed-number of object categories and require training generative or discriminative models for each category given training data. In the parsing stage, these systems try to align the learned models to the input image and associate object category labels with pixels, windows, edges or other image representations. Recently, context information has also been carefully modeled to capture the relationship between objects at the semantic level [46, 50]. Encouraging progress has been made by these models on a variety of object recognition and scene parsing tasks.

However, these learning-based methods do not, in general, scale well with the number of object categories. For example, to expand an existing system to include more object categories, we need to train new models for these categories and, typically adjust system parameters. Training can be a tedious job if we want to include thousands of object categories for a scene parsing system. In addition, the complexity of contextual relationships amongst objects also increases rapidly as the quantity of object categories expands.

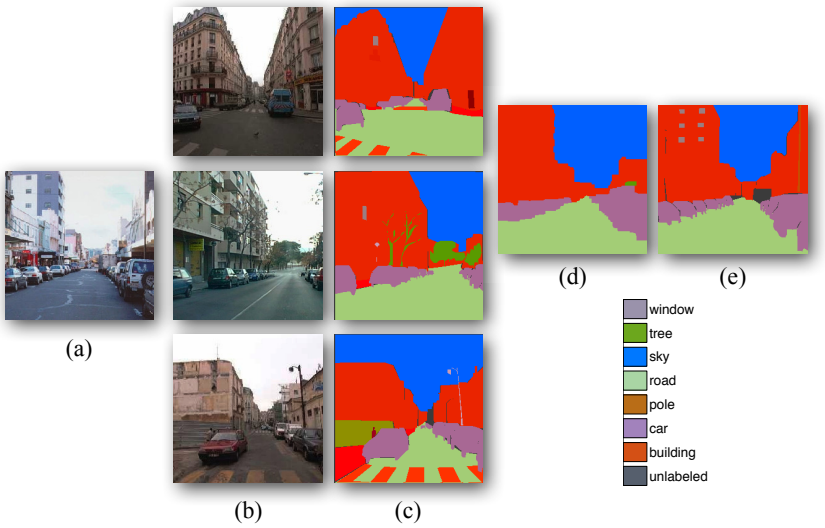


Figure 5.1. For a query image (a), our system finds the top matches (b) (three are shown here) using a modified, coarse-to-fine SIFT flow matching algorithm. The annotations of the top matches (c) are transferred and integrated to parse the input image as shown in (d). For comparison, the ground-truth user annotation of (a) is shown in (e).

Recently, the emergence of large databases of images has opened the door to a new family of methods in computer vision. Large database-driven approaches have shown the potential for nonparametric methods in several applications. Instead of training sophisticated parametric models, these methods try to reduce the inference problem for an unknown image to that of matching to an existing set of annotated images. In [102], the authors estimate the pose of a human relying on 0.5 million training examples. In [49], the proposed algorithm can fill holes on an input image by introducing elements that are likely to be semantically correct through searching a large image database. In [90], a system is designed to infer the possible object categories that may appear in an image by retrieving similar images in a large database [91]. Moreover, the authors in [118] showed that with a database of 80 million images, even simple SSD match can give semantically meaningful parsing for 32×32 images.

Motivated by the recent advances in large database-driven approaches, we designed a non-parametric scene parsing system to transfer the labels from existing samples to annotate an image through dense scene alignment, as illustrated in Figure 5.1. For a query image (a), our system first retrieves the top matches in the LabelMe database [91] using a combination of GIST matching [82] and SIFT flow [69]. Since these top matches are labeled, we transfer the annotation (c) of the top matches to the query image and obtain the scene parsing result in (d).

For comparison, the ground-truth user annotation of the query is displayed in (e). Our system is able to generate promising scene parsing results if images from the same scene category are retrieved in the annotated database.

However, it is nontrivial to build an efficient and reliable scene parsing system using dense scene alignment. The SIFT flow algorithm proposed in [69] does not scale well with image dimensions. Therefore, we propose a flexible, coarse-to-fine matching scheme to find dense correspondences between two images. To account for the multiple annotation suggestions from the top matches, a Markov random field model is used to merge multiple cues (*e.g.*, likelihood, prior and spatial smoothness) into reliable annotation. Promising experimental results are achieved on images from the LabelMe database [91].

Our goal is to explore the performance of scene parsing through the transfer of labels from existing annotated images, rather than building a comprehensive object recognition system. We show, however, that the performance of our system outperforms existing approaches [31, 106] on our dataset.

■ 5.2 Scene Parsing through Label Transfer

Now that we have a large database of annotated images and a technique of establishing dense correspondences across scenes, we can transfer the existing annotations to a query image through dense scene alignment. For a given query image, we retrieve a set of *K-nearest neighbors* in our database using GIST matching [82]. We then compute the SIFT flow from the query to each nearest neighbor, and use the achieved minimum energy (defined in Eqn. 4.1) to re-rank the K-nearest neighbors. We further select the top M re-ranked retrievals to create our voting candidate set. This voting set will be used to transfer its contained annotations into the query image. This procedure is illustrated in Figure 5.2.

Under this setup, scene parsing can be formulated as the following label transfer problem. For a query image I with its corresponding SIFT image s, we have a set of voting candidates $\{s_i, c_i, \mathbf{w}_i\}_{i=1:M}$, where s_i, c_i and w_i are the SIFT image, annotation, and SIFT flow field (from s to s_i) of the ith voting candidate. c_i is an integer image where $c_i(\mathbf{p}) \in \{1, \cdots, L\}$ is the index of object category for pixel $\mathbf{p}$. We want to obtain the annotation c for the query image by transferring c_i to the query image according to the dense correspondence $\mathbf{w}_i$.

We build a probabilistic Markov random field model to integrate multiple labels, prior information of object category, and spatial smoothness of the annotation to parse image I. Similar

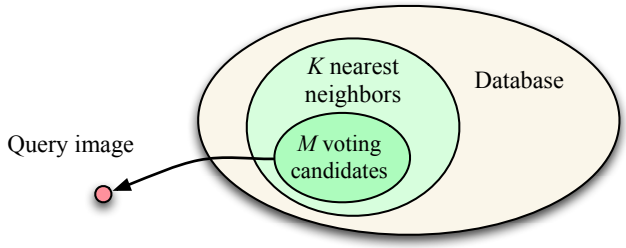


Figure 5.2. For a query image, we first find a K-nearest neighbor set in the database using GIST matching [82]. The nearest neighbors are re-ranked using SIFT flow matching scores, and form a top M-voting candidate set. The annotations are transferred from the voting candidates to the query image.

to that of [106], the posterior probability is defined as:

$$-\log P\big(c|I,s,\{s_i,c_i,\mathbf{w}_i\}\big)=\sum_{\mathbf{p}}\psi\big(c(\mathbf{p});s,\{s'_i\}\big)+$$
$$\alpha\sum_{\mathbf{p}}\lambda\big(c(\mathbf{p})\big)+\beta\sum_{\{\mathbf{p},\mathbf{q}\}\in\varepsilon}\phi\big(c(\mathbf{p}),c(\mathbf{q});I\big)+\log Z, \tag{5.1}$$

where Z is the normalization constant of the probability. This posterior contains three components, *i.e.* likelihood, prior and spatial smoothness.

The *likelihood* term is defined as

$$\psi\big(c(\mathbf{p})=l\big)=\begin{cases}\min\limits_{i\in\Omega_{\mathbf{p},l}}\|s(\mathbf{p})-s_i(\mathbf{p}+\mathbf{w}(\mathbf{p}))\|, & \Omega_{\mathbf{p},l}\neq\emptyset\\ \tau, & \Omega_{\mathbf{p},l}=\emptyset\end{cases} \tag{5.2}$$

where $\Omega_{\mathbf{p},l}=\{i;c_i(\mathbf{p}+\mathbf{w}(\mathbf{p}))=l\}$ is the index set of the voting candidates whose label is l after being warped to pixel $\mathbf{p}$. τ is set to be the value of the maximum difference of SIFT feature: $\tau=\max_{s_1,s_2,\mathbf{p}}\|s_1(\mathbf{p})-s_2(\mathbf{p})\|$.

The *prior* term is $\lambda(c(\mathbf{p})=l)$ indicates the prior probability that object category l appears at pixel $\mathbf{p}$. This is obtained from counting the occurrence of each object category at each location in the training set.

$$\lambda\big(c(\mathbf{p})=l\big)=-\log\mathrm{hist}_l(\mathbf{p}) \tag{5.3}$$

where $\mathrm{hist}_l(\mathbf{p})$ is the spatial histogram of object category l.

The *smoothness* term is defined to bias the neighboring pixels into having the same label if no other information is available, and the probability depends on the edge of the image: the stronger luminance edge, the more likely that the neighboring pixels may have different labels.

$$\phi\big(c(\mathbf{p}),c(\mathbf{q})\big)=\delta[c(\mathbf{p})\neq c(\mathbf{q})]\left(\frac{\epsilon+e^{-\gamma\|I(\mathbf{p})-I(\mathbf{q})\|^2}}{\epsilon+1}\right) \tag{5.4}$$

where $\gamma = (2 < \|I(\mathbf{p}) - I(\mathbf{q})\|^2 >)^{-1}$ [106].

Notice that the energy function is controlled by four parameters, K and M that decide the mode of the model, and α and β that control the influence of spatial prior and smoothness. Once the parameters are fixed, we again use BP-S algorithm to minimize the energy. The algorithm converges in two seconds on a workstation with two quad-core 2.67 GHz Intel Xeon CPUs.

A significant difference between our model and that in [106] is that we have fewer parameters because of the nonparametric nature of our approach, whereas classifiers where trained in [106]. In addition, color information is not included in our model at the present as the color distribution for each object category is diverse in our database.

■ 5.3 Experiments

We used a subset of the LabelMe database [91] to test our system. This data set contains 2688 fully annotated images, most of which are outdoor scenes including street, beach, mountains, fields and buildings. From these images we randomly selected 2488 for training and 200 for testing. We chose the top 33 object categories with the most labeled pixels. The pixels that are not labeled, or labeled as other object categories, are treated as the 34th category: "unlabeled". The per pixel frequency count of these object categories in the training set is shown at the top of Figure 5.3. The color of each bar is the average RGB value of the corresponding object category from the training data with saturation and brightness boosted for visualization. The top 10 object categories are, in descending order *sky*, *building*, *mountain*, *tree*, *unlabeled*, *road*, *sea*, *field*, *grass*, and *river*. The spatial priors of these object categories are displayed at the bottom of Figure 5.3. White means zero probability and saturated color means the highest probability. We observe that sky occupies the upper part of image grid and field occupies the lower part. Notice that there are only limited numbers of samples for the objects such as *sun*, *cow*, *bird*, and *moon*.

Our scene parsing system is illustrated in Figure 5.4. The system retrieves a K-nearest neighbor set for the query image (a), and further selects M voting candidates with the minimum SIFT matching score. For the purpose of illustration we set $M = 3$ here. The RGB image, SIFT image, and annotation of the voting candidates are shown in (c) to (e), respectively. The SIFT flow field is visualized in (f) using the same visualization scheme as in [69]. After we warp the voting candidates into the query with respect to the flow field, the warped RGB (g) and SIFT image (h) are very close to the query (a) and (b). Combining the warped annotations in (i), the system outputs the parsing of the query in (j), which is close to the ground-truth annotation in

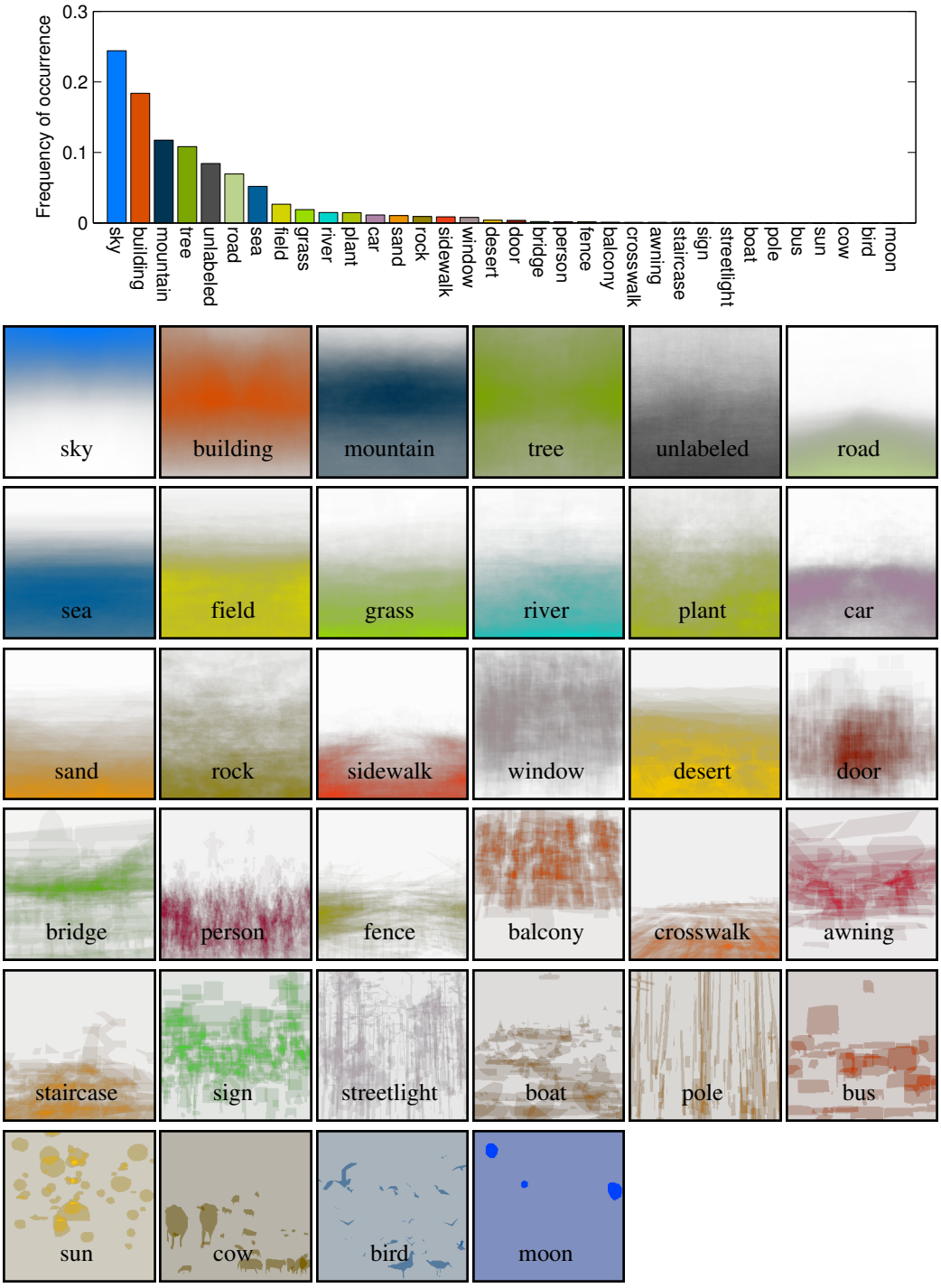


Figure 5.3. Above: the per-pixel frequency counts of the object categories in our dataset (sorted in descending order). The color of each bar is the average RGB value of each object category from the training data with saturation and brightness boosted for visualization. Bottom: the spatial priors of the object categories in the database. White means zero and the saturated color means high probability.

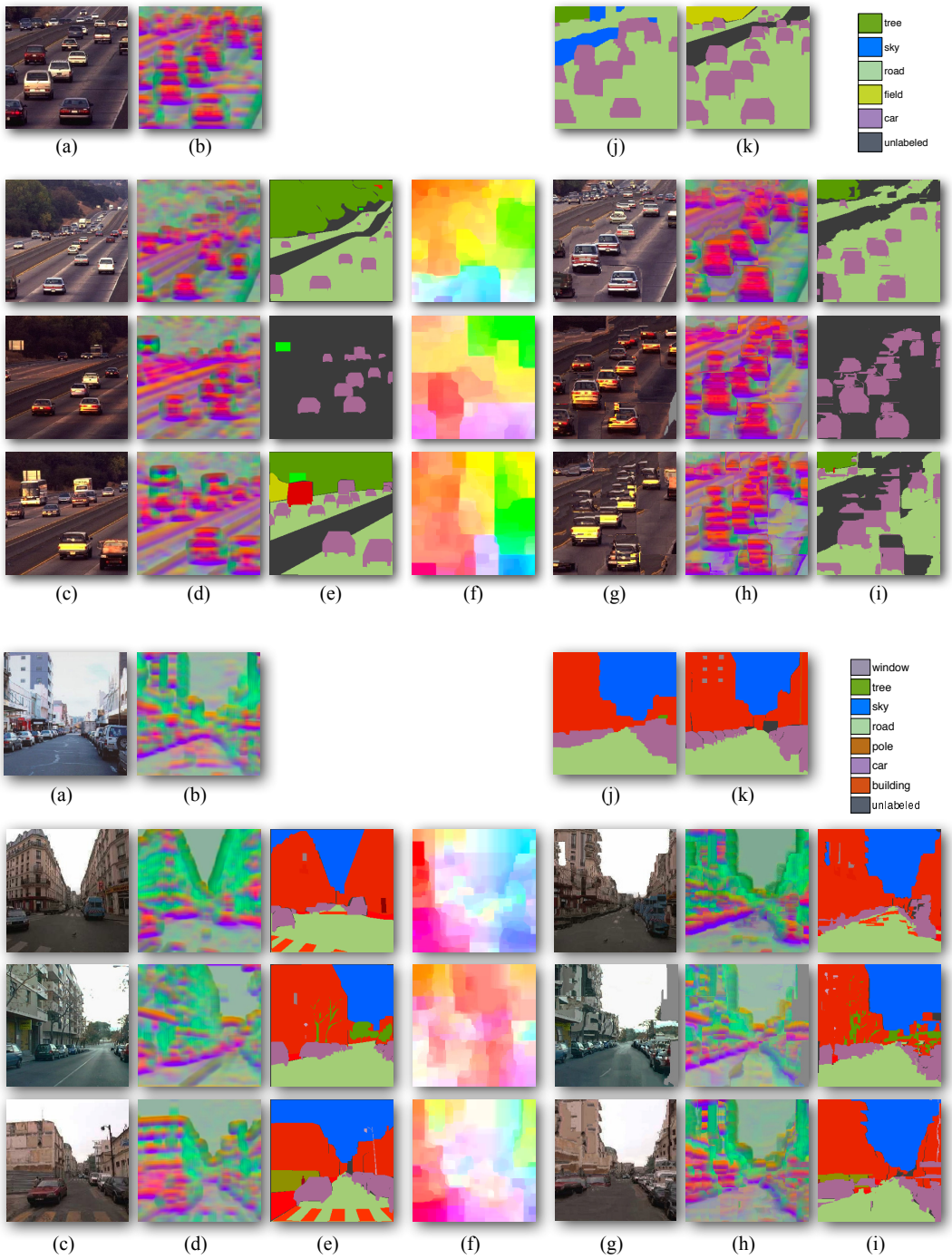


Figure 5.4. System overview. Our algorithm computes the SIFT image (b) of an query image (a), and uses GIST [82] to find its K nearest neighbors in our database. We apply coarse-to-fine SIFT flow to align the query image to the nearest neighbors, and obtain top M as voting candidates ($M = 3$ here). (c) to (e): the RGB image, SIFT image and user annotation of the voting candidates. (f): the inferred SIFT flow. From (g) to (i) are the warped version of (c) to (e) with respect to the SIFT flow in (f). Notice the similarity between (a) and (g), (b) and (h). Our system combines the voting from multiple candidates and generates scene parsing in (j) by optimizing the posterior. (k): the ground-truth annotation of (a).

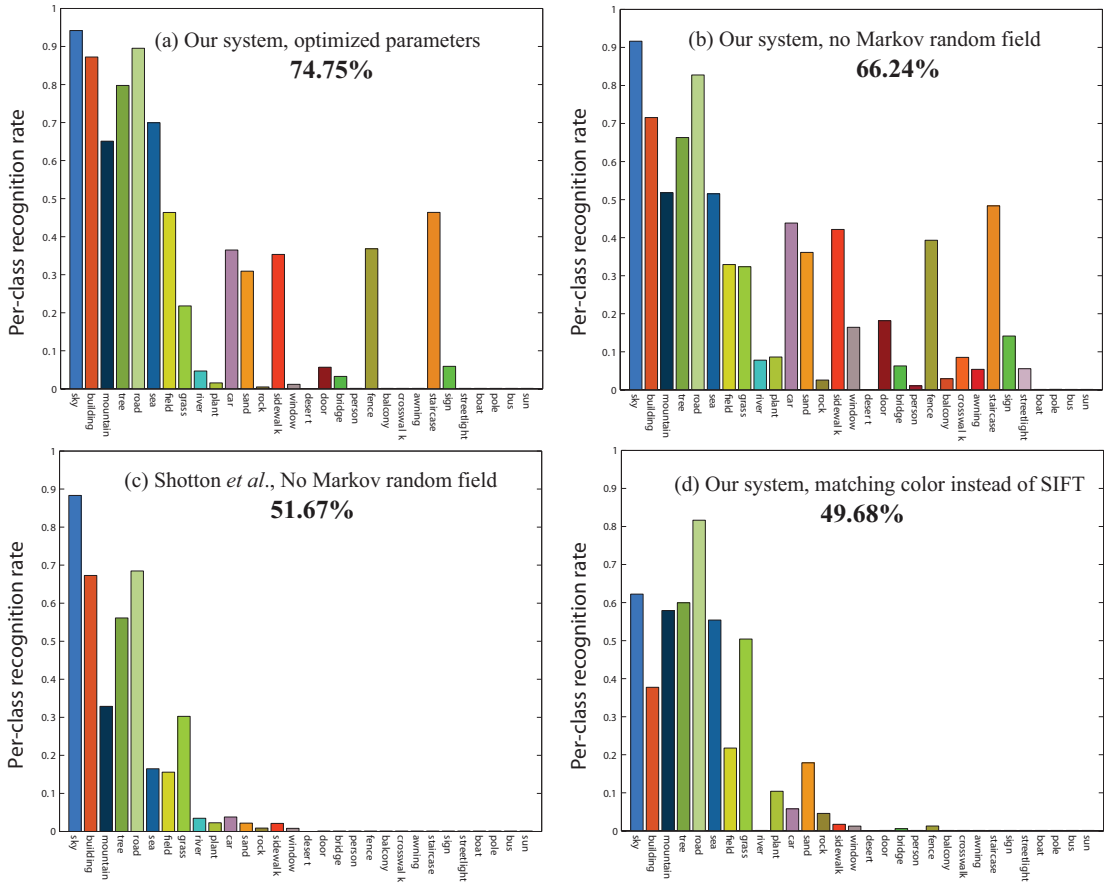


Figure 5.5. The per-class recognition rate of our system and the one in [106]. (a) Our system with the the parameters optimized for pixel-wise recognition rate. (b) Our system with $\alpha = \beta = 0$, namely, with the Markov random field model turned off. (c) The performance of the system in [106] also with the conditional random field turned off, trained and tested on the same data sets as (a) and (b). (d) Our system, but matching RGB instead of matching SIFT. We may observe that the our optimized system (a) is biased towards "stuff", large-region objects such as *sky*, *building*, *mountain*, and *tree*, because of the biased prior distribution of the classes as illustrated in Figure 5.3. When we sacrifice the overall performance by turning off the Markov random field model in Eqn. 5.1, our system performs better for small-size objects. To compare, we downloaded the code of a state-of-the-art object recognition system [106] based on training per-pixel classifiers using texton features, and ran the code on our dataset with the results shown in (c). Notice that the conditional random field model described in [106] is not available in the publicly available code. The fair comparison of (b) and (c) (both with spatial regularization turned off) suggests that our system, which is established upon dense scene alignment, outperforms [106], which relies on training pixel-wise classifiers. Lastly, we modified our system by matching RGB instead of SIFT without changing anything else, and showed the results in (d). Clearly, SIFT flow (a) results in better performance than optical flow (d), although optical flow produces reasonable results compared to [106].

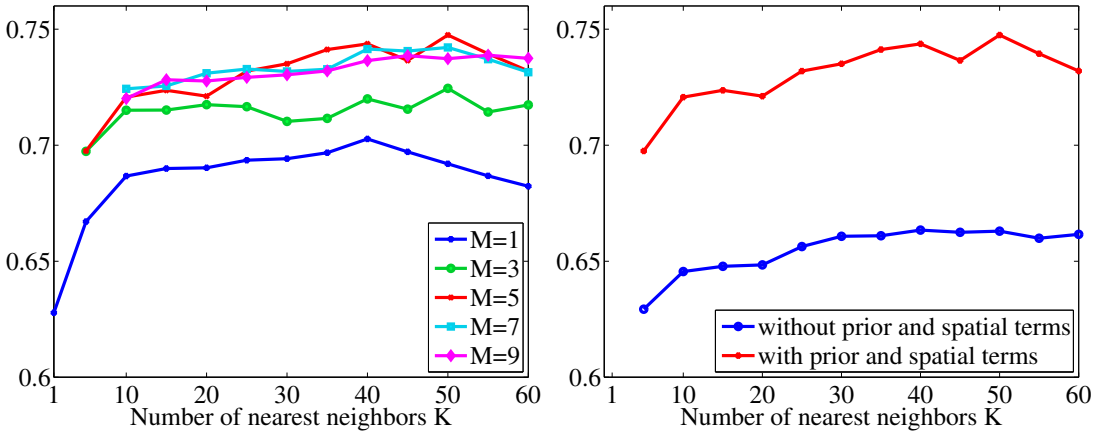
(k).

Some label transferring results are displayed in Figure 5.9. The input image from the test set is displayed in column (a). We show the best match, its corresponding annotation, and the warped best match in (b), (c) and (d), respectively. Even though our system takes the top M matches as voting candidates, due to lack of space we only show the best match to demonstrate how the system parses the query. Again, the warped image (d) looks similar to the input, indicating that SIFT flow successfully matches image structures. The scene parsing results output from our system are listed in column (e) with parameter setting $K = 50,\ M = 5,\ \alpha = 0.1, \beta = 70$. The ground-truth user annotation is listed in (f). Notice that the gray pixels in (f) are "unlabeled", but our system does not generate "unlabeled" output. For sample 1, 5, 6, 8 and 9, our system generates reasonable predictions for the pixels annotated as "unlabeled". The pixel-wise recognition rate of our system is **74.75%** by excluding the "unlabeled" class. A failure example for our system is shown in Figure 5.10 when the system fails to retrieve images with similar object categories to the query.

For comparison, we downloaded and ran the code from [106] using the same training and test data with the conditional random field (CRF) turned off (the downloaded code does not include the CRF part). The overall pixel-wise recognition rate of their system on our data set is **51.67%**, and the per-class rates are displayed in Figure 5.5 (c). For fairness we also turned off the Markov random field model in our framework by setting $\alpha = \beta = 0$, and plotted the corresponding results in Figure 5.5 (b). Clearly, our system outperforms [106] in terms of both overall and per-class recognition rate. We also modified our system by matching RGB instead of SIFT with everything else unchanged, and showed in the results in Figure 5.5 (d). The comparison of (a) and (d) suggests that SIFT flow results in better performance than optical flow, although optical flow produces reasonable results compared to [106].

Overall, our system is able to predict the right object categories in the input image with a segmentation fit to image boundary, even though the best match may look different from the input, *e.g.*, 2, 11, 12 and 17. If we divide the object categories into *stuff* (*e.g.*, sky, mountains, tree, sea and field) and *things* (*e.g.*, cars, sign, boat and bus) [2, 50], our system generates much better results for stuff than for things. The recognition rate for the top 7 object categories (all are "stuff") is 82.72%. This is because in our current system we only allow one labeling for each pixel, and smaller objects tend to be overwhelmed by the labeling of larger objects. We plan to build a recursive system in our future work to further retrieve things based on the inferred stuff.

We investigate the performance of our system by varying the parameters K, M, α and β. First, we fix $\alpha = 0.1,\ \beta = 70$ and plot the recognition rate as a function of K in Figure 5.6



(a) (b)

Figure 5.6. (a): Recognition rate as a function of the number of nearest neighbors K and the number of voting candidates M. (b): recognition rate as a function of the number of nearest neighbors K. Clearly, prior and spatial smoothness help improve the recognition rate.

(a) with different M. Overall, the recognition rate increases as more nearest neighbors are retrieved ($K\uparrow$) and more voting candidates are used ($M\uparrow$) since, obviously, multiple candidates are needed to transfer labels to the query. However, the recognition drops as K and M continue to increase as more candidates may include noise to label transfer. The maximum performance is obtained when $K = 50$ and $M = 5$. Second, we fix $M = 5$, and plot the recognition rate as a function of K by turning on prior and spatial terms ($\alpha = 0.1,\ \beta = 70$) and turning them off ($\alpha = \beta = 0$) in Figure 5.6 (b). Prior and spatial smoothness increase the performance of our system by about 7 percentage.

Lastly, we compared the performance of our system with a classifier-based system [31]. We downloaded their code and trained a classifier for each object category using the same training data. We converted our system into a binary object detector for each class by only using the per-class likelihood term. The per-class ROC curves of our system (red) and theirs (blue) are plotted in Figure 5.7. Except for five object categories, *grass*, *plant*, *boat*, *person* and *bus*, our system outperforms or equals theirs.

■ 5.4 Conclusion

We presented a novel, nonparametric scene parsing system to transfer the annotations from a large database to an input image using dense scene alignment. Using the dense scene corre-

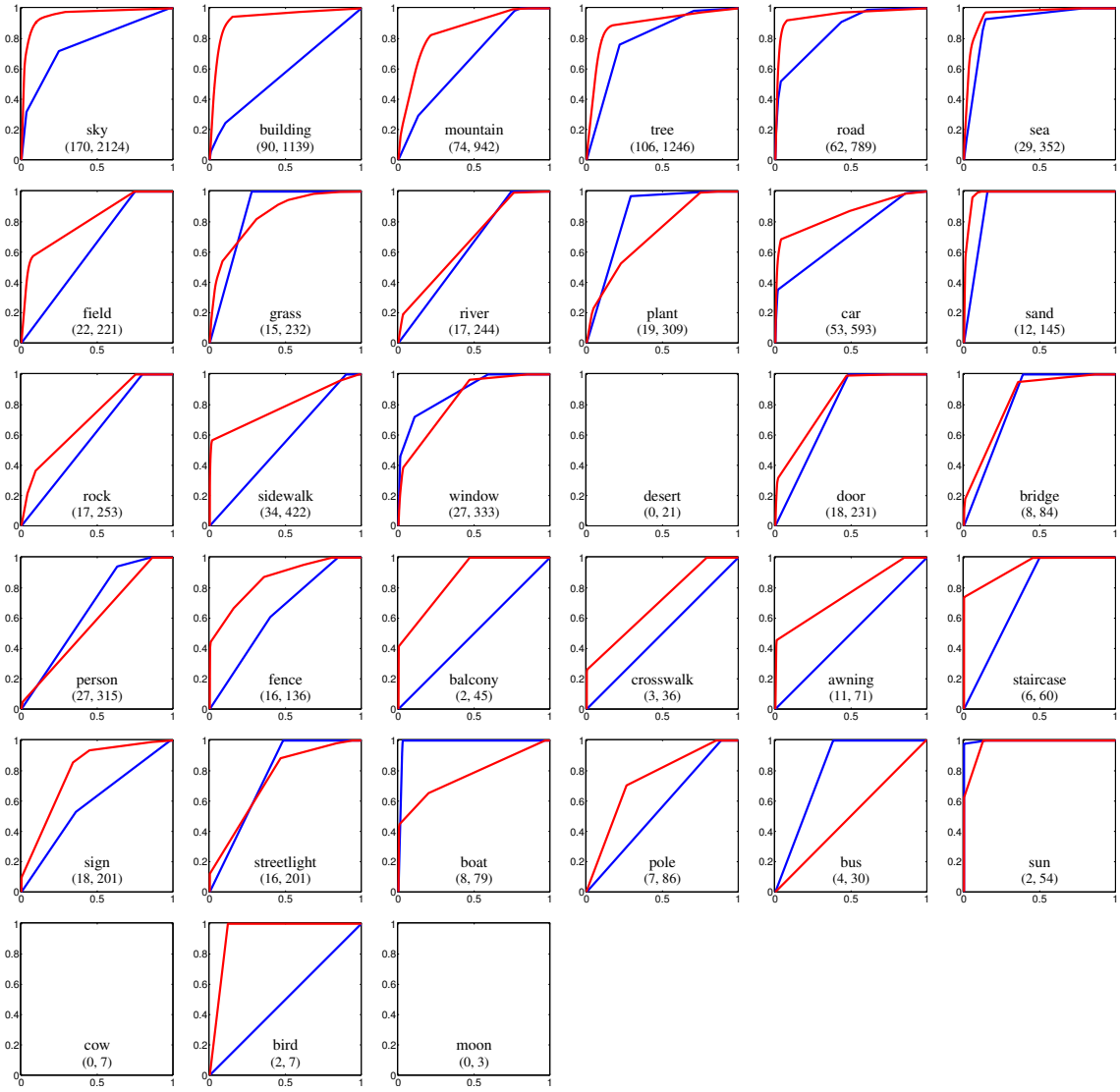


Figure 5.7. The ROC curve of each individual pixel-wise binary classifier. Red curve: our system after being converted to binary classifiers; blue curve: the system in [31]. We used convex hull to make the ROC curves strictly concave. The number (n, m) underneath the name of each plot is the quantity of the object instances in the test and training set, respectively. For example, (170, 2124) under "sky" means that there are 170 test images containing sky, and 2124 training images containing sky. Our system obtains reasonable performance for objects with sufficient samples in both training and test sets, *e.g.*, sky, building, mountain and tree. We observe truncation in the ROC curves where there are not enough test samples, *e.g.*, field, sea, river, grass, plant, car and sand. The performance is poor for objects without enough training samples, *e.g.*, crosswalk, sign, boat, pole, sun and bird. The ROC does not exist for objects without any test samples, *e.g.*, desert, cow and moon. In comparison, our system outperforms or equals [31] for all object categories except for grass, plant, boat, person and bus. The performance of [31] on our database is low because the objects have drastically different poses and appearances.

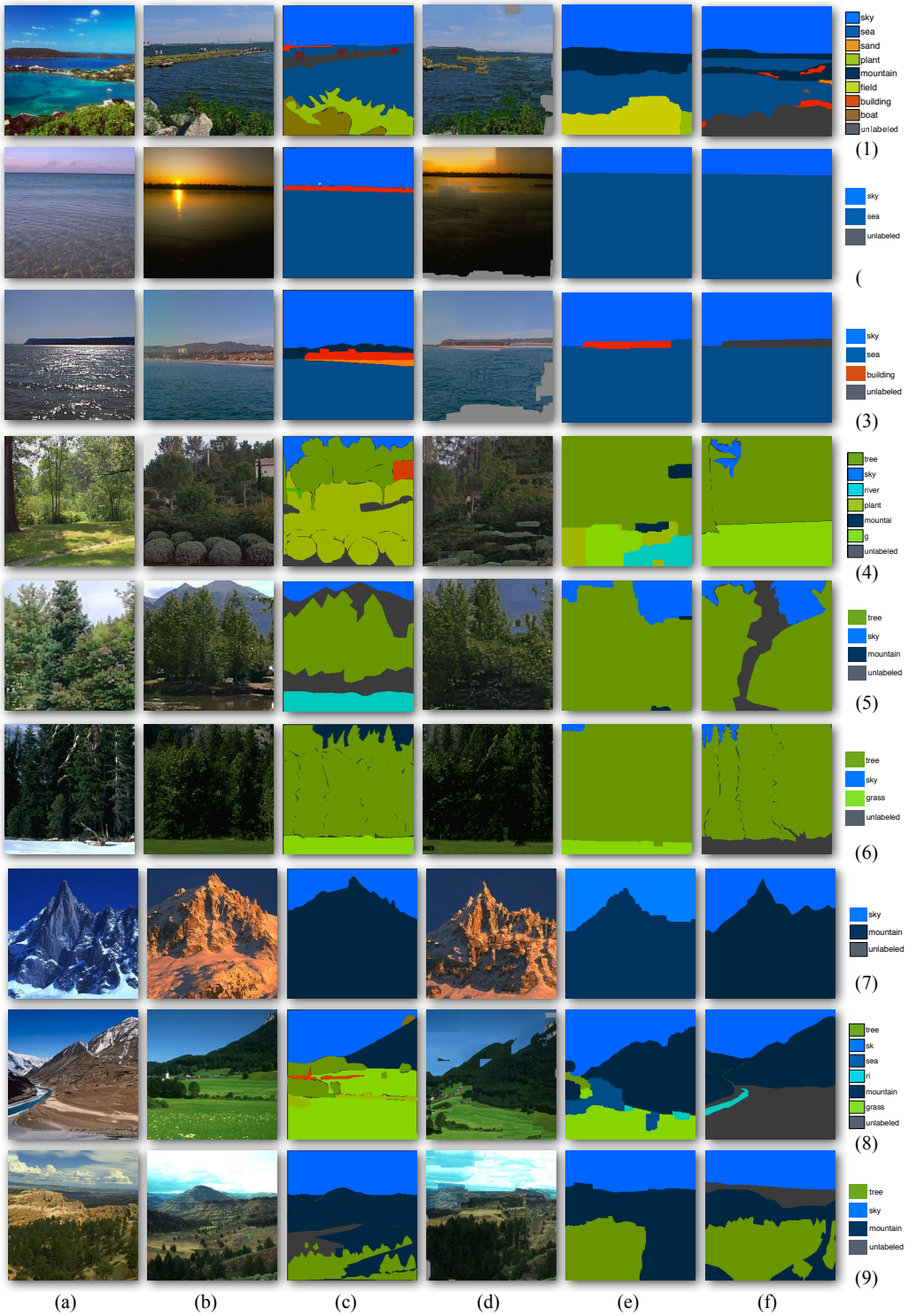


Figure 5.8. Some scene parsing results output from our system. (a): query image; (b): the best match from nearest neighbors; (c): the annotation of the best match; (d): the warped version of (b) according to the SIFT flow field; (e): the inferred per-pixel parsing after combining multiple voting candidates; (f): the ground truth annotation of (a). The dark gray pixels in (f) are "unlabeled". Notice how our system generates a reasonable parsing even for these "unlabeled" pixels.

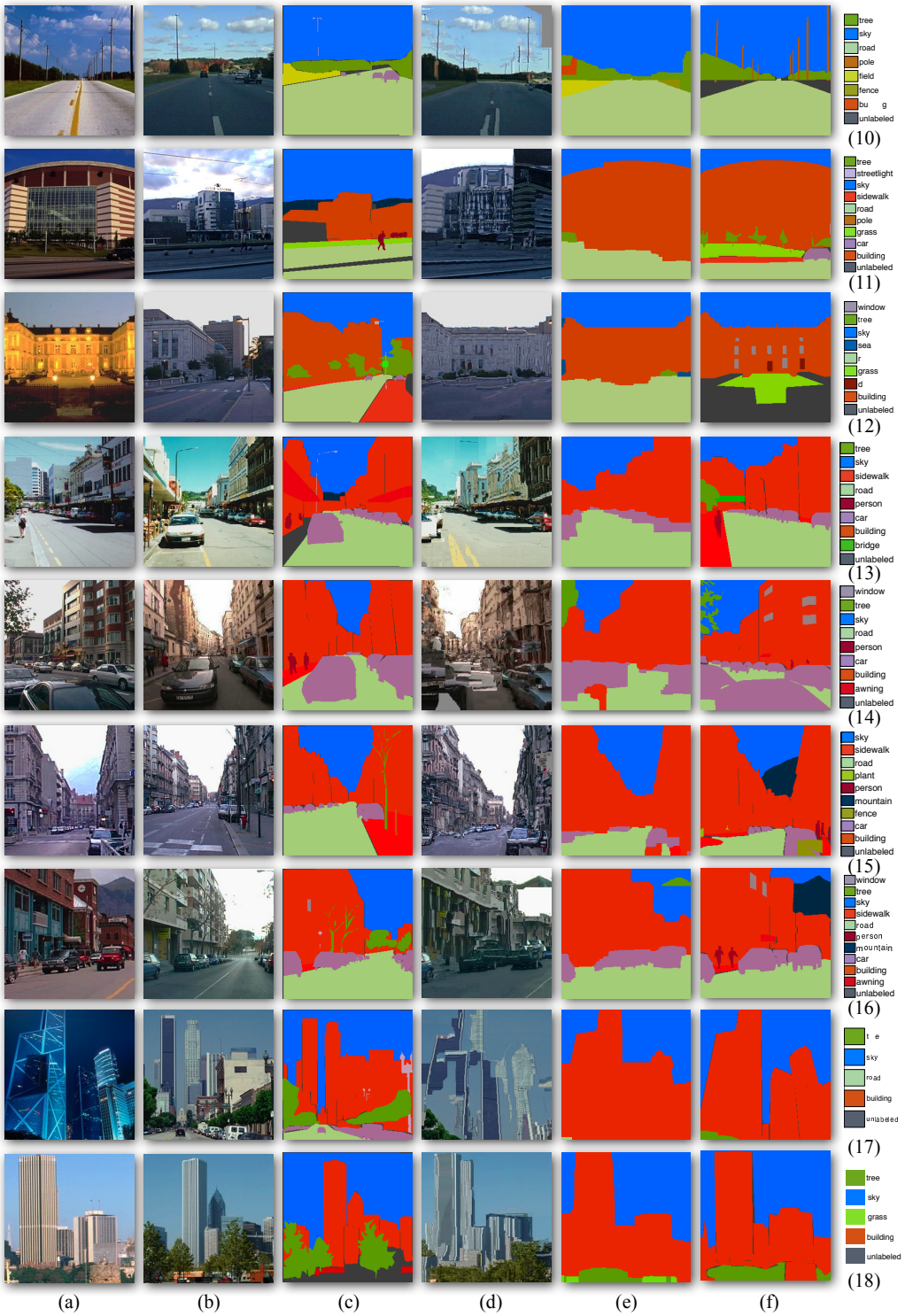
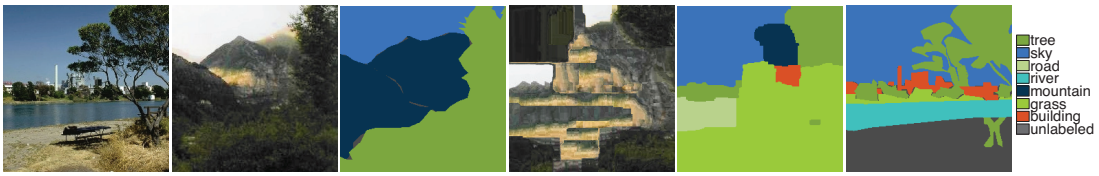


Figure 5.9. Some scene parsing results output from our system. (a): query image; (b): the best match from nearest neighbors; (c): the annotation of the best match; (d): the warped version of (b) according to the SIFT flow field; (e): the inferred per-pixel parsing after combining multiple voting candidates; (f): the ground truth annotation of (a). The dark gray pixels in (f) are "unlabeled". Notice how our system generates a reasonable parsing even for these "unlabeled" pixels.



(a) (b) (c) (d) (e) (f)

Figure 5.10. Our system fails when no good matches can be retrieved in the database. Since the best matches do not contain river, the input image is mistakenly parsed as a scene of grass, tree and mountain in (e). The ground-truth annotation is in (f).

spondences, we warp the pixel labels of the existing samples to the query. Furthermore, we integrate multiple cues to segment and recognize the query image into the object categories in the database. Promising results have been achieved by our scene alignment and parsing system on a challenging database. Compared to existing approaches that require training for each object category, our nonparametric scene parsing system is easy to implement, has only a few parameters, and embeds contextual information naturally in the retrieval/alignment procedure.

Chapter 6

Conclusion

In the previous chapters we have been through several topics that expand the frontier of motion analysis beyond the pixel level along three directions, obtaining motion ground truth for real-world videos, proposing new representations and exploring new applications. Below, we will summarize the contributions of my thesis.

In Chapter 2, *Human-Assisted Motion Annotation*, we designed a system to integrate the state-of-the-art computer vision algorithms and human expertise of motion perception to annotate ground-truth motion for real-world video sequences at every pixel and every frame. A special graphical user interface was designed to allow the user to label layers, inspect motions, select parameters, and specify sparse correspondences. Our methodology was validated by comparing with the ground-truth motion obtained through other means and by measuring the consistency of human annotation. Using our system, we collected a motion ground-truth database consisting of challenging real-world videos for algorithm evaluation and benchmarking. Our human-in-the-loop system is able to obtain much more accurate motion estimates than automated methods. We hope that our database and annotation code will improve both the training and evaluation of algorithms for optical flow and layered motion analysis. The human-assisted motion annotation system can be useful by itself for advanced video analysis and editing tasks.

In *Motion Magnification* (Chapter 3.1), we presented a system to reveal motions that would otherwise be invisible or be very difficult for humans to see. Our automatic layer analysis system is able to robustly extract layer and motion information from a video sequence even when the magnitude of the motion is very small. We proposed to estimate the motion of a video sequence through tracking feature points with an *adaptive region of support* that represents the local grouping relationship of pixels. Iteratively estimating regions of support and tracking feature points help the system to overcome occlusions. We also proposed *normalized complex correlation* to group the trajectory of feature points according to the same physical cause. Fea-

ture points that exhibit similar frequency characteristics are grouped to form layer primitives. Dense layer segmentation is obtained through a hypothesis-testing framework combining spatial, temporal and color cues. We demonstrated that our approach is able to reliably estimate layers in challenging sequences.

In general, motion magnification can be a useful tool for visualizing and measuring small, subtle deformation of objects of interest. We have applied motion magnification to revealing almost invisible motion of eardrum membranes, breathing of babies, deformation of faults, and to exaggerating the motion of cars under different loads.

In *Analysis of Contour Motions* (Chapter 3.2), we proposed a novel boundary-based representation to estimate motion under the challenging visual conditions of moving textureless objects. We introduced a hierarchical representation of *edgelets*, *boundary fragments*, and *contours* for textureless objects. Ambiguous local motion measurements are resolved through a graphical model relating these edgelets, boundary fragments, completed contours, and their motions. Contours are grouped from boundary fragments and their motions analyzed simultaneously, leading to the correct handling of otherwise spurious occlusion and T-junction features. The motion cues help the contour completion task, allowing completion of contours that would be difficult or impossible using only low-level information in a static image. Our system has been successfully applied to analyzing motion for challenging motion stimuli, such as Kanizsa Square and the rotating chair sequence, where conventional optical flow algorithms fail miserably.

In Chapter 4, we proposed *SIFT Flow* to establish correspondence between images across scenes. We followed the computational framework of optical flow, but instead of matching pixel values, we matched the transform-invariant feature SIFT. A coarse-to-fine Belief Propagation algorithm was designed to reduce the computational complexity from $O(n^4)$ to $O(n^2 \log n)$. Through many examples, we demonstrate that SIFT flow is able to establish semantically meaningful correspondences across complex scenes despite significant differences in appearance and spatial layout of the matched images. We also demonstrated SIFT flow in video retrieval, motion estimation from a single image and video synthesis via transferring moving objects with the support of large databases. We further applied SIFT flow to traditional image alignment problems, such as satellite image registration and face recognition. The preliminary success of these experiments suggests that scene alignment techniques such as SIFT flow can be useful tools for various applications in computer vision and computer graphics.

In Chapter 5, we further applied SIFT flow to *Nonparametric Scene Parsing*. Using the dense scene correspondences, we warped the pixel labels of the existing samples in a large

database to an unlabeled query image. Based on the warped annotations, we integrated multiple cues to segment and recognize the query image in the object categories present in the database. Promising results have been achieved by our scene alignment and parsing system for a challenging database. Compared to existing approaches, which require training generative models or classifiers for each object category, our nonparametric scene parsing system is easy to implement, has only a few parameters, and embeds contextual information naturally in the retrieval/alignment procedure. We demonstrated that our system outperforms the state-of-the-art object recognition systems that rely on classifiers.

For the future work, we plan to explore more, innovative representations for motion analysis with solid computational models to provide more robust tools for computer vision, such as video editing, tracking, object recognition, and visual learning.

Appendix A

Estimating Optical Flow

Optical flow estimation is one of the corner stones of this thesis, especially in human-assisted motion annotation and motion magnification. In this Appendix, we will show how to derive optical flow in a different framework than the classical ways in the literature.

■ A.1 Two-Frame Optical Flow Computation

Our first task is to build a framework for flow estimation. While the mathematical derivation appears complicated, using IRLS we can make it easy and straightforward. We shall focus on two-frame flow estimation in this section, and move to multi-frame in next section.

■ A.1.1 Problem formulation

Let the image lattice be $\mathbf{p}=(x,y,t)$ and the underlying flow field be $\mathbf{w}(\mathbf{p})=(u(\mathbf{p}),v(\mathbf{p}),1)$, where $u(\mathbf{p})$ and $v(\mathbf{p})$ are the horizontal and vertical components of the flow field, respectively. For simplicity, we shall omit $\mathbf{p}$ when it is clear from the context. Under the brightness constancy assumption [51], the pixel value should be consistent along the flow vector, and the flow field should be piecewise smooth [14, 20]. This results in an objective function in the continuous spatial domain

$$E(u,v) = \int \psi(|I(\mathbf{p}+\mathbf{w})-I(\mathbf{p})|^2)+\alpha\phi(|\nabla u|^2+|\nabla v|^2)d\mathbf{p}, \tag{A.1}$$

where $\psi(\cdot)$ and $\phi(\cdot)$ are robust functions [14], ∇ is the gradient operator $|\nabla u|^2 = u_x^2+u_y^2$ ($u_x = \frac{\partial}{\partial x}u,\ u_y=\frac{\partial}{\partial y}u$), and α weights the regularization. In this paper we restrict the robust functions to be the L1 norm, which results in a piecewise smooth flow field, *i.e.* $\psi(x)=\sqrt{x^2+\varepsilon^2}$, $\phi(x)=\sqrt{x^2+\varepsilon^2}$ ($\varepsilon=0.001$), but it can be any function as shown in Fig. B.1. Notice that this objective function is highly non-convex, and it has been shown that a coarse to fine refining scheme on a dense Gaussian pyramid (*e.g.*, with downsampling rate 0.8) with image warping can avoid

getting stuck at bad local minima of the function [20].

Although most of the existing optical flow work was derived based on Eq. (A.1) or some variations using *Euler-Lagrange* variational approach, the mathematical derivation is rather complicated since a function in the continuous spatial domain needs to be optimized. Alternatively, one can derive the optical flow algorithm directly from a discrete version of Eq. (A.1). Under the incremental flow framework, it is assumed that an estimate of the flow field is known as $\mathbf{w}$, and one needs to estimate the best increment $d\mathbf{w}$ ($d\mathbf{w}=(du,dv)$). The objective function in Eq. (A.1) is then changed to

$$E(du,dv)=\int \psi(|I(\mathbf{p}+\mathbf{w}+d\mathbf{w})-I(\mathbf{p})|^2)+\alpha\phi(|\nabla(u+du)|^2+|\nabla(v+dv)|^2)d\mathbf{p}. \tag{A.2}$$

Let

$$I_z(\mathbf{p})=I(\mathbf{p}+\mathbf{w})-I(\mathbf{p}),\quad I_x(\mathbf{p})=\frac{\partial}{\partial x}I(\mathbf{p}+\mathbf{w}),\quad I_y(\mathbf{p})=\frac{\partial}{\partial y}I(\mathbf{p}+\mathbf{w}), \tag{A.3}$$

and $I(\mathbf{p}+\mathbf{w}+d\mathbf{w})-I(\mathbf{p})$ can be linearized by a first-rder Taylor expansion

$$I(\mathbf{p}+\mathbf{w}+d\mathbf{w})-I(\mathbf{p})\approx I_z(\mathbf{p})+I_x(\mathbf{p})du(\mathbf{p})+I_y(\mathbf{p})dv(\mathbf{p}). \tag{A.4}$$

In the above equation we add $\mathbf{p}$ into du and dv to emphasize that du and dv are indexed by space and time.

Since we focus on two-frame flow in this section, we shall assume that t is fixed and omit t in $\mathbf{p}$. We vectorize $u,\ v,\ du,\ dv$ into $U,\ V,\ dU,\ dV$. Let $\mathbf{I}_x=\mathrm{diag}(I_x)$ and $\mathbf{I}_y=\mathrm{diag}(I_y)$ be diagonal matrices where the diagonals are the images I_x and I_y. We use $\mathbf{D}_x$ and $\mathbf{D}_y$ to denote the matrix corresponding to x- and y-derivative filters, *i.e.* $\mathbf{D}_xU=u*[0\ \ -1\ 1]$. We introduce column vector $\delta_\mathbf{p}$ that has only one nonzero (one) value at location $\mathbf{p}$, *e.g.*, $\delta_\mathbf{p}I_z=I_z(\mathbf{p})$ and $\delta_\mathbf{p}I_x=I_x(\mathbf{p})$. The continuous function in Eq. (A.2) can be discretized as

$$\begin{aligned}E(dU,dV)=\sum_\mathbf{p}\quad & \psi\Big(\ \big(\delta_\mathbf{p}^T(I_z+\mathbf{I}_xdU+\mathbf{I}_ydV)\big)^2\Big)+\\ & \alpha\phi\Big(\ \big(\delta_\mathbf{p}^T\mathbf{D}_x(U+dU)\big)^2+\big(\delta_\mathbf{p}^T\mathbf{D}_y(U+dU)\big)^2+\\ & \quad\big(\delta_\mathbf{p}^T\mathbf{D}_x(V+dV)\big)^2+\big(\delta_\mathbf{p}^T\mathbf{D}_y(V+dV)\big)^2\Big).\end{aligned} \tag{A.5}$$

■ A.1.2 Iterative Reweighted Least Squares (IRLS)

The main idea of iterative reweighted least squares (IRLS) [79] is to find dU,dV so that the gradient $[\frac{\partial E}{\partial dU};\frac{\partial E}{\partial dV}]=0$. Let

$$\begin{aligned}g_\mathbf{p} &= \big(\delta_\mathbf{p}^T\mathbf{D}_x(U+dU)\big)^2+\big(\delta_\mathbf{p}^T\mathbf{D}_y(U+dU)\big)^2+\big(\delta_\mathbf{p}^T\mathbf{D}_x(V+dV)\big)^2+\big(\delta_\mathbf{p}^T\mathbf{D}_y(V+dV)\big)^2,\\ f_\mathbf{p} &= \big(\delta_\mathbf{p}^T(I_z+\mathbf{I}_xdU+\mathbf{I}_ydV)\big)^2.\end{aligned} \tag{A.6}$$

We can derive

$$\frac{\partial E}{\partial dU} = \sum_{\mathbf{p}} \psi'(f_{\mathbf{p}}) \frac{\partial f_{\mathbf{p}}}{\partial dU} + \alpha \phi'(g_{\mathbf{p}}) \frac{\partial g_{\mathbf{p}}}{\partial dV} \tag{A.7}$$

$$= 2 \sum_{\mathbf{p}} \psi'(f_{\mathbf{p}}) \big(\mathbf{I}_x \delta_p \delta_p^T \mathbf{I}_x dU + \mathbf{I}_x \delta_p \delta_p^T (I_z + \mathbf{I}_y dV) \big) + \alpha \phi'(g_{\mathbf{p}}) (\mathbf{D}_x^T \delta_{\mathbf{p}} \delta_{\mathbf{p}}^T \mathbf{D}_x + \mathbf{D}_y^T \delta_{\mathbf{p}} \delta_{\mathbf{p}}^T \mathbf{D}_y)(dU + U) \tag{A.8}$$

$$= 2 \Big((\mathbf{\Psi}' \mathbf{I}_x^2 + \alpha \mathbf{L}) dU + \mathbf{\Psi}' \mathbf{I}_x \mathbf{I}_y dV + \mathbf{\Psi}' \mathbf{I}_x I_z + \alpha \mathbf{L} U \Big). \tag{A.9}$$

From Eq. (A.7) to Eq. (A.8) we used $\frac{d}{dx} x^T \mathbf{A} x = 2\mathbf{A}x$, $\frac{d}{dx} x^T b = b$, where x and b are vectors and $\mathbf{A}$ is a matrix. From Eq. (A.8) to Eq. (A.9) we used the fact that $\sum_{\mathbf{p}} \delta_{\mathbf{p}} \delta_{\mathbf{p}}^T$ is the identity matrix, and $\mathbf{I}_x$, $\mathbf{I}_y$ are diagonal matrices. We also defined the vector $\psi' = [\psi'(f_{\mathbf{p}})]$ and $\phi' = [\phi'(g_{\mathbf{p}})]$, and the diagonal matrix $\mathbf{\Psi}' = \mathrm{diag}(\psi')$, $\mathbf{\Phi}' = \mathrm{diag}(\phi')$. A generalized Laplacian filter $\mathbf{L}$ is defined as

$$\mathbf{L} = \mathbf{D}_x^T \mathbf{\Phi}' \mathbf{D}_x + \mathbf{D}_y^T \mathbf{\Phi}' \mathbf{D}_y. \tag{A.10}$$

Then one can show

$$\frac{\partial E}{\partial dV} = 2 \Big(\mathbf{\Psi}' \mathbf{I}_x \mathbf{I}_y dU + (\mathbf{\Psi}' \mathbf{I}_y^2 + \alpha \mathbf{L}) dV + \mathbf{\Psi}' \mathbf{I}_y I_z + \alpha \mathbf{L} V \Big). \tag{A.11}$$

Both Eq. (A.9) and (A.11) contain nonlinear function $\psi'(f_{\mathbf{p}})$ and $\phi'(g_{\mathbf{p}})$, and solving $[\frac{\partial E}{\partial dU}; \frac{\partial E}{\partial dV}] = 0$ can be performed in the following fixed-point iterations:

(a) Initialize $dU = 0,\ dV = 0$.

(b) Compute the "weight" $\mathbf{\Psi}'$ and $\mathbf{\Phi}'$ based on the current estimate dU and dV.

(c) Solve the following linear equation

$$\begin{bmatrix} \mathbf{\Psi}' \mathbf{I}_x^2 + \alpha \mathbf{L} & \mathbf{\Psi}' \mathbf{I}_x \mathbf{I}_y \\ \mathbf{\Psi}' \mathbf{I}_x \mathbf{I}_y & \mathbf{\Psi}' \mathbf{I}_y^2 + \alpha \mathbf{L} \end{bmatrix} \begin{bmatrix} dU \\ dV \end{bmatrix} = - \begin{bmatrix} \mathbf{\Psi}' \mathbf{I}_x I_z + \alpha \mathbf{L} U \\ \mathbf{\Psi}' \mathbf{I}_y I_z + \alpha \mathbf{L} V \end{bmatrix} \tag{A.12}$$

(d) If dU and dV converge, stop; otherwise, goto (b).

This algorithm is called iterative reweighted least squares because it iterates between (re)computing the weight, the nonlinear term $\mathbf{\Psi}'$ and $\mathbf{\Phi}'$ in step (b), and solving a least squares problem in step (c). Notice that the matrix in the linear system Eq. (A.12) is positive definite, and we use conjugate gradient method to solve this linear system.

Although this algorithm is shown to be identical to the two fixed-point iterations in [20] in Appendix A, IRLS has the following two advantages over the traditional Euler-Lagrange method.

- $\mathbf{D}_x$ and $\mathbf{D}_y$ are not limited to derivative filters. The filters learnt from ground-truth flows such as in [87] can be easily incorporated under IRLS framework to capture more sophisticated characteristics of the flow fields.
- It is easier to handle large-magnitude flows for the temporal constraint. This will be shown in Sect 3.

We also show that IRLS is equivalent to the variational upper-bound optimization method [55] in Appendix B. Therefore, IRLS is guaranteed to converge to a local minimum since the variational method is guaranteed to do so.

A.1.3 Multi-channel and Lucas-Kanade

We further assume $I(\mathbf{x}) \in \mathbb{R}^d$ to cover multi-channel images (*e.g.*, RGB), or add more features to the original image (*e.g.*, gradient image, so that gradient matching in [20] is included). Meanwhile, a spatial smoothing kernel can be applied to the data term for robustness as a Lucas-Kanade term [71, 22]. These additions can be easily achieved under IRLS framework by slightly modifying Eq. (A.12). Notice that in Eq. (A.12) $\mathbf{\Psi}'\mathbf{I}_x\mathbf{I}_y = \mathrm{diag}([\psi'_{\mathbf{p}} I_x(\mathbf{p}) I_y(\mathbf{p})])$. For a multi-channel image, $I_x(\mathbf{p})$, $I_y(\mathbf{p})$, and $I_z(\mathbf{p})$ all become vectors. We define

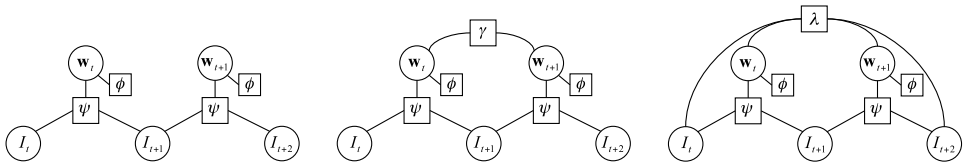
$$\mathbf{\Psi}'_{xy} = \mathrm{diag}(g * [\psi'_{\mathbf{p}} I_x^T(\mathbf{p}) I_y(\mathbf{p})]), \tag{A.13}$$

where g is a small spatial Gaussian kernel (*e.g.*, with standard deviation 1). Similarly we define $\mathbf{\Psi}'_{xx}$, $\mathbf{\Psi}'_{yy}$, $\mathbf{\Psi}'_{xz}$ and $\mathbf{\Psi}'_{yz}$. Eq. (A.12) can be rewritten as

$$\begin{bmatrix} \mathbf{\Psi}'_{xx} + \alpha\mathbf{L} & \mathbf{\Psi}'_{xy} \\ \mathbf{\Psi}'_{xy} & \mathbf{\Psi}'_{yy} + \alpha\mathbf{L} \end{bmatrix} \begin{bmatrix} dU \\ dV \end{bmatrix} = - \begin{bmatrix} \mathbf{\Psi}'_{xz} + \alpha\mathbf{L}U \\ \mathbf{\Psi}'_{yz} + \alpha\mathbf{L}V \end{bmatrix}. \tag{A.14}$$

A.2 Temporal Constraints: Multiple Frames

It is natural to use multiple frames to improve the flow estimation. In [20, 21] a 3D gradient ∇_3 is introduced to impose the temporal constraint. This is, however, inadequate for many video sequences containing dramatic object movements. We need a more sophisticated temporal constraint to let the flow estimates agree over multiple frames. IRLS can help simplify the derivation.



(a) Two-frame flow (b) Constant velocity (c) 2nd-order data term

Figure A.1. The factor graph of optical flow estimation. Notice that the flow fields are coupled in (b) and (c) in different ways.

■ A.2.1 Constant velocity model

We assume that the velocity is constant from frame to frame along with the flow vectors. Recall that $\mathbf{p} = (x, y, t)$ and $\mathbf{w}(\mathbf{p}) = (u(\mathbf{p}), v(\mathbf{p}), 1)$. We can write the new flow estimate over the whole sequence

$$\begin{aligned} E\big(\{u(\mathbf{p}), v(\mathbf{p})\}\big) \quad &= \quad \sum_t \int g * \psi(|I(\mathbf{p}) - I(\mathbf{p}+\mathbf{w})|^2) + \\ &\alpha\phi(|\nabla u(\mathbf{p})|^2 + |\nabla v(\mathbf{p})|^2) + \beta\gamma(|\mathbf{w}(\mathbf{p}) - \mathbf{w}(\mathbf{p}+\mathbf{w})|^2) d\mathbf{p}, \end{aligned} \quad \text{(A.15)}$$

where $\mathbf{w}(\mathbf{p}+\mathbf{w}) = (u(\mathbf{p}+\mathbf{w}), v(\mathbf{p}+\mathbf{w}), 1)$ is the warped flow field at time $t+1$ to time t according to flow $\mathbf{w}$. $\gamma(\cdot)$ is the same robust function as $\psi(\cdot)$ and $\phi(\cdot)$.

It would be difficult to estimate the flow fields simultaneously. Instead, we assume $\{U_t, V_t\}$ has been computed and we need to compute $\{dU_t, dV_t\}$. Except for the first and last frames, $\{dU_t, dV_t\}$ is affected by the temporal term at both $t-1$ and t. We need to modify step (b) and (c) accordingly in IRLS. In step (b) we compute an additional weight

$$\gamma'_t = \gamma'(|\mathbf{w}(\mathbf{p}) + d\mathbf{w}(\mathbf{p}) - \mathbf{w}(\mathbf{p}+\mathbf{w})|^2) \quad \text{(A.16)}$$

Let $\mathbf{\Gamma}'_t = \mathrm{diag}(\gamma'_t)$. The iterative reweighted least square problem for solving Eq. (A.15) can be derived as

$$\begin{aligned} &\begin{bmatrix} \mathbf{\Psi}'_{xx} + \alpha\mathbf{L} + \beta\mathbf{M}_t & \mathbf{\Psi}'_{xy} \\ \mathbf{\Psi}'_{xy} & \mathbf{\Psi}'_{yy} + \alpha\mathbf{L} + \beta\mathbf{M}_t \end{bmatrix} \begin{bmatrix} dU_t \\ dV_t \end{bmatrix} = \\ &- \begin{bmatrix} \mathbf{\Psi}'_{xz} + \alpha\mathbf{L}U_t + \beta\big(\mathbf{M}_t U_t - \mathbf{\Gamma}'_t\mathbf{H}_t U_{t+1} - \mathbf{H}^T_{t-1}\mathbf{\Gamma}'_{t-1}U_{t-1}\big) \\ \mathbf{\Psi}'_{yz} + \alpha\mathbf{L}V_t + \beta\big(\mathbf{M}_t V_t - \mathbf{\Gamma}'_t\mathbf{H}_t V_{t+1} - \mathbf{H}^T_{t-1}\mathbf{\Gamma}'_{t-1}V_{t-1}\big) \end{bmatrix}, \end{aligned} \quad \text{(A.17)}$$

where $\mathbf{M}_t = \mathbf{\Gamma}'_t + \mathbf{H}^T_{t-1}\mathbf{\Gamma}'_{t-1}\mathbf{H}_{t-1}$, and $\mathbf{H}_t$ is the bilinear warping matrix corresponding to the flow field (U_t, V_t). There is no need to generate $\mathbf{H}_t$ and compute the transpose $\mathbf{H}^T_t$. Instead, we can stack the interpolation coefficients at each pixel and treat them as a filter when applying $\mathbf{H}^T_t$.

A.2.2 Second-order data term

Another possible temporal constraint is a long-range match, *e.g.*, to match frame t to frame $t+2$, $t+3$, $\cdots$ as well. Even though in theory the frames can be matched at any temporal distance, we restrict the matching to the second frame by adding the matching between t and $t+2$. The objective function becomes

$$\begin{aligned} E\big(\{u(\mathbf{p}),v(\mathbf{p})\}\big) \quad = \quad & \sum_t \int g * \Big(\psi(|I(\mathbf{p})-I(\mathbf{p}+\mathbf{w})|^2)+ \\ & \xi\lambda(|I(\mathbf{p})-I(\mathbf{p}+\mathbf{w}+\mathbf{w}(\mathbf{p}+\mathbf{w}))|^2)\Big)+\alpha\phi(|\nabla u(\mathbf{p})|^2+|\nabla v(\mathbf{p})|^2) \end{aligned} \tag{A.18}$$

where $\lambda(\cdot)$ is also a robust function. Similar to Eq. (A.3) we have the following abbreviations for 2nd-order matching

$$\begin{aligned} I_{x2} &= \tfrac{\partial}{\partial x} I(\mathbf{p}+\mathbf{w}+\mathbf{w}(\mathbf{p}+\mathbf{w})), \\ I_{y2} &= \tfrac{\partial}{\partial y} I(\mathbf{p}+\mathbf{w}+\mathbf{w}(\mathbf{p}+\mathbf{w})), \\ I_{z2} &= I(\mathbf{p}+\mathbf{w}+\mathbf{w}(\mathbf{p}+\mathbf{w}))-I(\mathbf{p}). \end{aligned} \tag{A.19}$$

Noting that the 2nd-order matching is essentially the same as the 1st-order matching problem, we can modify Eq. (A.13) for 2nd-order matching

$$\lambda'_t = \lambda'(|I_{z2}|^2), \quad \mathbf{\Lambda}'_{xyt} = \mathrm{diag}(g * \lambda'_t I_{x2}^T I_{y2}). \tag{A.20}$$

$\mathbf{\Lambda}'_{xxt}$, $\mathbf{\Lambda}'_{yyt}$, $\mathbf{\Lambda}'_{xzt}$ and $\mathbf{\Lambda}'_{yzt}$ are defined in a similar way. The IRLS for solving Eq. (A.18) is then

$$\begin{bmatrix} \mathbf{\Psi}'_{xx}+\xi\mathbf{P}_{xxt}+\alpha\mathbf{L} & \mathbf{\Psi}'_{xy}+\xi\mathbf{P}_{xyt} \\ \mathbf{\Psi}'_{xy}+\xi\mathbf{P}_{xyt} & \mathbf{\Psi}'_{yy}+\xi\mathbf{P}_{yyt}+\alpha\mathbf{L} \end{bmatrix} \begin{bmatrix} dU_t \\ dV_t \end{bmatrix} = - \begin{bmatrix} \mathbf{\Psi}'_{xz}+\xi\mathbf{P}_{xzt}+\alpha\mathbf{L}U_t \\ \mathbf{\Psi}'_{yz}+\xi\mathbf{P}_{yzt}+\alpha\mathbf{L}V_t \end{bmatrix} \tag{A.21}$$

where $\mathbf{P}_{xxt} = \mathbf{\Lambda}'_{xxt} + \mathbf{H}_{t-1}^T \mathbf{\Lambda}'_{xx,t-1} \mathbf{H}_{t-1}$, and similarly for $\mathbf{P}_{xyt}$ and $\mathbf{P}_{yyt}$. $\mathbf{P}_{xzt} = \mathbf{\Lambda}'_{xzt} + \mathbf{H}_{t-1}^T \mathbf{\Lambda}'_{xz,t-1}$, and similarly for $\mathbf{P}_{yzt}$.

We show the factor graphs of the constant velocity model and 2nd-order data term in Figure A.1. Clearly these two terms both add temporal constraints to the flow fields but in different ways. It is thus natural to combine them by adding Eq. (A.15) and (A.18) together. The IRLS solution becomes the linear system combining Eq. (A.17) and (A.21).

Appendix B

The Equivalence between Variational Optimization and IRLS

In this Appendix, we will show that variational optimization and iterative reweighted least square (IRLS) are equivalent in optimizing a particular family of object function that appears frequently in computer vision.

■ B.1 Introduction

In computer vision, we often need to minimize the following object function

$$Q(z) = \sum_{i=1}^{n} \phi\big((a_i^T z + b_i)^2\big) \tag{B.1}$$

where $z,\ a_i \in \mathbb{R}^d$, $b_i \in \mathbb{R}$, and $\phi : \mathbb{R} \rightarrow \mathbb{R}$. Normally $\phi(\cdot)$ is a nonnegative, monotonic function satisfying

$$\phi(0) = 0, \phi'(x^2) > 0. \tag{B.2}$$

If $\phi(x^2) = x^2$, then the objective function in Eqn. (B.1) is quadratic and the solution is straightforward. However, for many applications a robust function is used, for example, Lorentzian function $\phi(x^2) = \log(1 + x^2)$, $L1$ norm $\phi(x^2) = |x|$ and truncated quadratic function $\phi(x^2) = \min(x^2, c)$ (this function is not continuously differentiable and therefore difficult to optimize). The question is how to effectively optimize this objective function with these robust functions. In this document we focus on continuous differentiable functions.

Notice that we write the robust function in the form $\phi(x^2)$ instead of $\phi(x)$ to enforce the function to be an evenly symmetric function. Therefore, the notion $\phi'(x^2)$ is equal to $2\phi'|_{x^2}x$.

■ B.2 Variational Optimization

For a function $\phi(x^2)$, we define a quadratic upper bound as a quadratic function in x with coefficient determined by λ:

$$h(x, \lambda) = a(\lambda)x^2 + b(\lambda)x + d(\lambda) \tag{B.3}$$

The coefficient functions $a(\lambda)$, $b(\lambda)$ and $d(\lambda)$ are chosen by imposing the following constraints on $h(\cdot)$:

$$\begin{aligned} h(\lambda_0, \lambda_0) &= \phi(\lambda_0^2), \\ h'(\lambda_0, \lambda_0) &= \phi'(\lambda_0^2), \\ h'(0, \lambda_0) &= 0. \end{aligned} \tag{B.4}$$

Solving these equations leads to

$$a(\lambda) = \phi'(\lambda^2),\ b(\lambda) = 0,\ d(\lambda) = \phi(\lambda^2) - \phi'(\lambda^2)\lambda^2. \tag{B.5}$$

Based on this result we plotted the family of quadratic upper bound for power function $\phi(x^2) = (x^2)^\alpha$, $\alpha \in (0,\ 1]$ and Lorentzian function $\phi(x^2) = \log(1 + x^2)$ in Figure B.1.

Figure B.1 suggests that function $\phi(x^2)$ can be rewritten as

$$\phi(x^2) = \min_\lambda \left(\phi'(\lambda^2)x^2 + \phi(\lambda^2) - \phi'(\lambda^2)\lambda^2\right), \tag{B.6}$$

and the minimum is obtained when $\lambda = \pm x$. This can be verified that by solving

$$\frac{\partial}{\partial \lambda}\left(\phi'(\lambda^2)x^2 + \phi(\lambda^2) - \phi'(\lambda^2)\lambda^2\right) = 0 \tag{B.7}$$

$$\lambda\phi''(\lambda^2)(x^2 - \lambda^2) = 0 \tag{B.8}$$

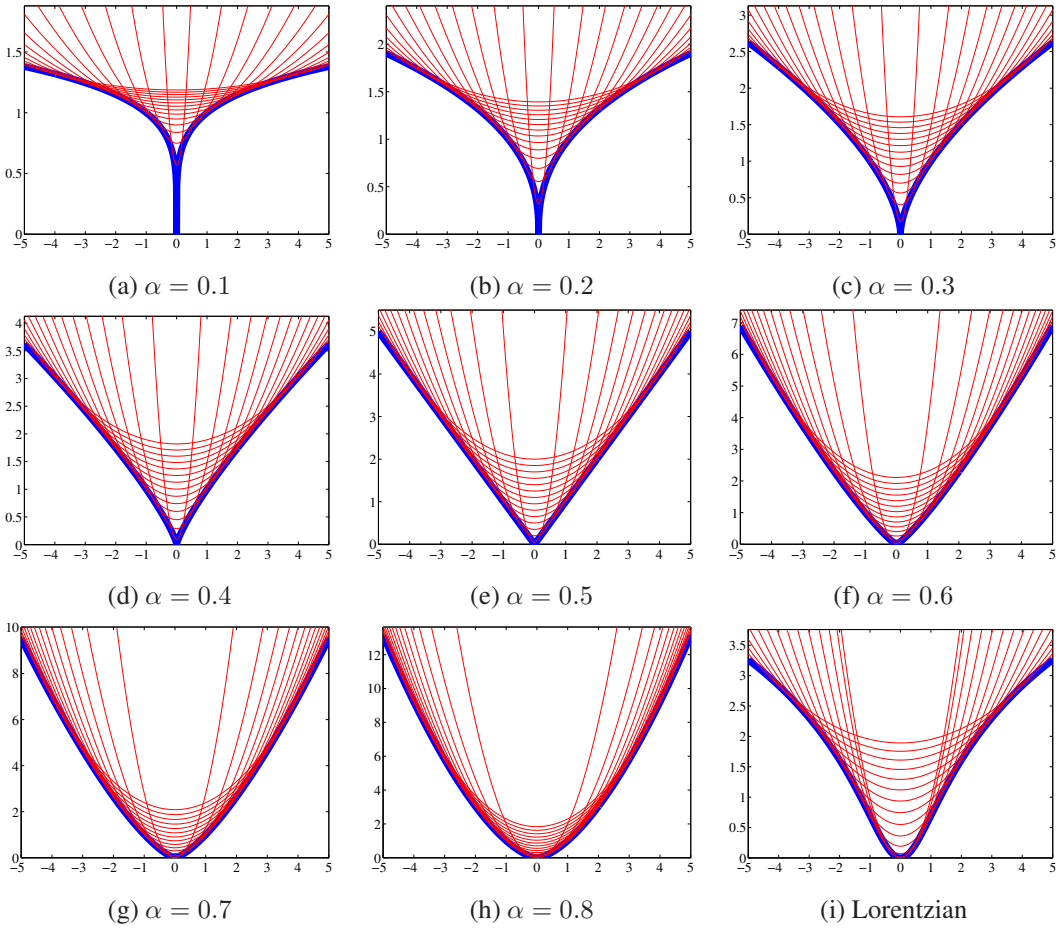
we obtain three roots, $\lambda = \pm x$ and $\lambda = 0$. The second derivative is only positive when $\lambda = \pm x$.

Based on the quadratic upper bounds, we can rewrite the objective function

$$\min_z \sum_{i=1}^n \phi\left((a_i^T z + b_i)^2\right) \tag{B.9}$$

$$= \min_z \sum_{i=1}^n \min_{\lambda_i} \left(\phi'(\lambda_i)^2(a_i^T z + b_i)^2 + \phi(\lambda^2) - \phi'(\lambda^2)\lambda^2\right). \tag{B.10}$$

The new form suggests a two-step coordinate descent algorithm for finding the optimal z.



(a) $\alpha = 0.1$ (b) $\alpha = 0.2$ (c) $\alpha = 0.3$

(d) $\alpha = 0.4$ (e) $\alpha = 0.5$ (f) $\alpha = 0.6$

(g) $\alpha = 0.7$ (h) $\alpha = 0.8$ (i) Lorentzian

Figure B.1. Both power function $\phi(x^2) = (x^2)^\alpha$ and Lorentzian function $\phi(x^2) = \log(1 + x^2)$ can be upper-bounded by quadratic functions.

1. Assume that we already have a value of z. We can minimize Eqn. (B.10) in terms of the variational parameter λ_i. From Eqn. (B.6), the optimal λ_i is:

$$\lambda_i = a_i^T z + b_i, \quad i = 1, \cdots, n. \tag{B.11}$$

2. Threat all the variational parameters λ_i constant and minimize Eqn. (B.10) in terms of z (least square):

$$\mathbf{A}\mathbf{W}\mathbf{A}^T z = -\mathbf{A}\mathbf{W}b \tag{B.12}$$

where $\mathbf{A} = [a_1, \cdots, a_n]$, $\mathbf{W} = \text{diag}\left(\phi'(\lambda_1^2), \cdots, \phi'(\lambda_n)\right)$, $b = [b_1, \cdots, b_n]^T$.

These two steps are iterated until convergence.

■ B.3 Iterative Reweighted Least Square (IRLS)

We can also minimize the objective function in Eqn. (B.1) from a different angle. Set the derivative of the objective function to be zero

$$\sum_i \phi'\left((a_i^T z + b_i)^2\right)\left(a_i^T z + b_i\right) a_i = 0. \tag{B.13}$$

There is a nonlinear term $\phi'\left((a_i^T z + b_i)^2\right)$ in the above equation. This suggest a fixed-point iteration for the optimization. We treat the nonlinear terms as weights of a least square problem– the problem is a least square problem if the nonlinear term is regarded as constant. The fixed-point iteration consists of the following two iterations.

---

1. Compute the weight

$$w_i = \phi'\left((a_i^T z + b_i)^2\right), \quad i = 1, \cdots, n. \tag{B.14}$$

2. Perform a weighted least square to solve z:

$$\mathbf{AWA}^T z = -\mathbf{AW}b \tag{B.15}$$

where $\mathbf{A} = [a_1, \cdots, a_n]$, $\mathbf{W} = \mathrm{diag}\,(w_1, \cdots, w_n)$, $b = [b_1, \cdots, b_n]^T$.

This fixed-point iteration is called iterative reweighted least square because we try to solve a least square problem, but with different weight at every iteration. The weight comes from the solution in the previous step.

■ B.4 Variational Inference and IRLS are Identical

Clearly, for the objective function in Eqn. (B.1) variational inference and IRLS generate exact the same algorithm. The variational parameter in variational optimization and the weight in IRLS are connected by

$$w_i = \phi'(\lambda_i^2). \tag{B.16}$$

However, the derivation using IRLS is more straightforward and succinct.

Bibliography

[1] E. H. Adelson. Layered representations for image coding. Vision and Modeling Technical Report 181, MIT Media Laboratory, 1991. 40

[2] E. H. Adelson. On seeing stuff: the perception of materials by humans and machines. In *SPIE, Human Vision and Electronic Imaging VI*, pages 1–12, 2001. 133

[3] A. Agarwala, A. Hertzmann, D. H. Salesin, and S. M. Seitz. Keyframe-based tracking for rotoscoping and animation. *ACM SIGGRAPH*, 23(3):584–591, 2004. 40

[4] L. Alvarez, R. Deriche, T. Papadopoulo, and J. Sánchez. Symmetrical dense optical flow estimation with occlusions detection. In *European Conference on Computer Vision (ECCV)*, pages 721–735, 2002. 39, 48

[5] O. Arikan and D. A. Forsyth. Synthesizing constrained motions from examples. *ACM Transactions on Graphics*, 21(3):483–490, July 2002. 60

[6] I. Austvoll. *A Study of the Yosemite Sequence Used as a Test Sequence for Estimation of Optical Flow*, pages 659–668. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2005. 37

[7] S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski. A database and evaluation methodology for optical flow. In *Proc. ICCV*, 2007. 13, 15, 30, 34, 37, 40, 52, 53, 56, 106

[8] B. Balas and P. Sinha. Observing object motion induces increased generalization and sensitivity. *Perception*, 37:1160–1174, 2008. 29

[9] J. L. Barron, D. J. Fleet, and S. S. Beauchemin. Systems and experiment performance of optical flow techniques. *International Journal of Computer Vision (IJCV)*, 12(1):43–77, 1994. 94

[10] S. Belongie, J. Malik, and J. Puzicha. Shape context: A new descriptor for shape matching and object recognition. In *Advances in Neural Information Processing Systems (NIPS)*, 2000. 30, 94, 125

[11] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 24(4):509–522, 2002. 97, 98

[12] A. Berg, T. Berg., and J. Malik. Shape matching and object recognition using low distortion correspondence. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005. 19, 30, 94, 95, 97, 125

[13] J. R. Bergen, P. Anandan, K. J Hanna, and R. Hingorani. Hierarchical model-based motion estimation. In *European Conference on Computer Vision (ECCV)*, pages 237–252, 1992. 97

[14] M. J. Black and P. Anandan. The robust estimation of multiple motions: parametric and piecewise-smooth flow fields. *Computer Vision and Image Understanding*, 63(1):75–104, January 1996. 39, 41, 94, 143

[15] M. J. Black and D. J. Fleet. Probabilistic detection and tracking of motion boundaries. *International Journal of Computer Vision*, 38(3):231–245, 2000. 82

[16] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 23(11):1222–1239, 2001. 63, 71, 97

[17] M. Brand and A. Hertzmann. Style machines. In *Proceedings of ACM SIGGRAPH 2000*, pages 183–192, July 2000. 60

[18] G. Brostow and I. Essa. Motion-based video decompositing. In *IEEE International Conference on Computer Vision (ICCV '99)*, pages 8–13, 1999. 63, 73

[19] G. Brostow and I. Essa. Image-based motion blur for stop motion animation. In *Proceedings of ACM SIGGRAPH 2001*, pages 561–566, 2001. 60

[20] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. In *European Conference on Computer Vision (ECCV)*, pages 25–36, 2004. 30, 39, 41, 45, 48, 49, 96, 97, 122, 143, 144, 146

[21] A. Bruhn, J. Weickert, T. Kohlberger, and C. Schnörr. A multigrid platform for real-time motion computation with discontinuity-preserving variational methods. *International Journal of Computer Vision (IJCV)*, 70(3):257–277, 2006. 146

[22] A. Bruhn, J. Weickert, and C. Schnörr. Lucas/Kanade meets Horn/Schunk: combining local and global optical flow methods. *International Journal of Computer Vision (IJCV)*, 61(3):211–231, 2005. 13, 14, 15, 31, 38, 39, 41, 48, 55, 56, 93, 111, 146

[23] P. J. Burt and E. H. Adelson. The laplacian pyramid as a compact image code. *IEEE Transactions on Communications*, COM-31,4:532–540, 1983. 14, 46

[24] D. Cai, X. He, Y. Hu, J. Han, and T. Huang. Learning a spatially smooth subspace for face recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007. 27, 122

[25] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 8(6):679–698, Nov 1986. 83, 84

[26] C. Carson, S. Belongie, H. Greenspan, and J. Malik. Blobworld: Color- and Texture-Based Image Segmentation Using EM and Its Application to Image Querying and Classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 24(8):1026–1038, 2002. 98

[27] Y.-Y. Chuang, A. Agarwala, B. Curless, D. H. Salesin, and R. Szeliski. Video matting of complex scenes. In *ACM SIGGRAPH*, 2002. 40

[28] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. In *European Conference on Computer Vision (ECCV)*, volume 2, pages 484–498, 1998. 96

[29] N. Cornelis and L. V. Gool. Real-time connectivity constrained depth map computation using programmable graphics hardware. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1099–1104, 2005. 103, 123

[30] A. Criminisi, G. Cross, A. Blake, and V. Kolmogorov. Bilayer segmentation of live video. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006. 29

[31] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005. 24, 30, 97, 125, 127, 134, 135

[32] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. R. Statist. Soc. B*, 39:1–38, 1977. 65

[33] A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In *IEEE International Conference on Computer Vision*, pages 1033–1038, Corfu, Greece, September 1999. 63, 73

[34] P. Felzenszwalb and D. Huttenlocher. Pictorial structures for object recognition. *International Journal of Computer Vision (IJCV)*, 61(1), 2005. 30, 94, 125

[35] P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008. 30, 125

[36] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient belief propagation for early vision. *International Journal of Computer Vision (IJCV)*, 70(1):41–54, 2006. 19, 102

[37] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2003. 30, 125

[38] D. J. Fleet, A. D. Jepson, and M. R. M. Jenkin. Phase-based disparity measurement. *Computer Vision, Graphics and Image Processing (CVGIP)*, 53(2):198–210, 1991. 96

[39] W. T. Freeman and E. H. Adelson. The design and use of steerable filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 13(9):891–906, Sep 1991. 83

[40] W. T. Freeman, E. C. Pasztor, and O. T. Carmichael. Learning low-level vision. *International Journal of Computer Vision (IJCV)*, 40(1):25–47, 2000. 97

[41] R. Ghaffari, A. J. Aranyosi, and D. M. Freeman. Longitudinally propagating traveling waves of the mammalian tectorial membrane. *Proceedings of the National Academy of Sciences of the United States of America*, 104(42):16510–16515, Oct. 2007. 77

[42] M. Gleicher. Retargetting motion to new characters. In *Proceedings of ACM SIGGRAPH 98*, pages 33–42, July 1998. 60

[43] M. M. Gorkani and R. W. Picard. Texture orientation for sorting photos at a glance. In *IEEE International Conference on Pattern Recognition (ICPR)*, volume 1, pages 459–464, 1994. 97

[44] K. Grauman and T. Darrell. Pyramid match kernels: Discriminative classification with sets of image features. In *IEEE International Conference on Computer Vision (ICCV)*, 2005. 30, 104, 125

[45] W. E. L. Grimson. Computational experiments with a feature based stereo algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 7(1):17–34, 1985. 96

[46] A. Gupta and L. S. Davis. Beyond nouns: Exploiting prepositions and comparative adjectives for learning visual classifiers. In *European Conference on Computer Vision (ECCV)*, 2008. 125

[47] M. J. Hannah. *Computer Matching of Areas in Stereo Images*. PhD thesis, Stanford University, 1974. 96

[48] C. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of the 4th Alvey Vision Conference*, pages 147–151, 1988. 64, 122

[49] J. Hays and A. A Efros. Scene completion using millions of photographs. *ACM SIGGRAPH*, 26(3), 2007. 94, 126

[50] G. Heitz and D. Koller. Learning spatial context: Using stuff to find things. In *European Conference on Computer Vision (ECCV)*, 2008. 125, 133

[51] B. K. P. Horn and B. G. Schunck. Determing optical flow. *Artificial Intelligence*, 17:185–203, 1981. 29, 37, 39, 45, 81, 93, 96, 97, 122, 143

[52] M. Isard and A. Blake. Condensation – conditional density propagation for visual tracking. *International Journal of Computer Vision (IJCV)*, 29(1):5–28, 1998. 41

[53] N. Jojic and B. Frey. Learning flexible sprites in video layers. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'01)*, pages 199–206, Kauai, December 2001. 60, 63

[54] D. G. Jones and J. Malik. A computational framework for determining stereo correspondence from a set of linear spatial filters. In *European Conference on Computer Vision (ECCV)*, pages 395–410, 1992. 96

[55] M. Jordan. *Leanring in Graphical Models*. Cambridge MA: MIT Press, 1999. 146

[56] V. Kolmogorov and R. Zabih. Computing visual correspondence with occlusions using graph cuts. In *IEEE International Conference on Computer Vision (ICCV)*, pages 508–515, 2001. 97

[57] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume II, pages 2169–2178, 2006. 30, 98, 104, 125

[58] J. Lee, J. Chai, P. S. A. Reitsma, J. K. Hodgins, and N. S. Pollard. Interactive control of avatars animated with human motion data. *ACM Transactions on Graphics*, 21(3):491–500, July 2002. 60

[59] Y. Li, J. Sun, and H.-Y. Shum. Video object cut and paste. In *ACM SIGGRAPH*, 2005. 40

[60] Y. Li, T. Wang, and H.-Y. Shum. Motion texture: A two-level statistical model for character motion synthesis. *ACM Transactions on Graphics*, 21(3):465–472, July 2002. 60

[61] C. Liu, W. T. Freeman, and E. H. Adelson. Analysis of contour motions. In *Advances in Neural Information Processing Systems (NIPS)*, 2006. 13, 33, 35, 37, 40

[62] C. Liu, W. T. Freeman, and E. H. Adelson. Analysis of contour motions. In *Advances in Neural Information Processing Systems (NIPS)*, 2006. 97

[63] C. Liu, W. T. Freeman, E. H. Adelson, and Y. Weiss. Human-assisted motion annotation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008. 33, 109

[64] C. Liu, W. T. Freeman, R. Szeliski, and S. B. Kang. Noise estimation from a single image. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 901–908, 2006. 36

[65] C. Liu, H. Y. Shum, and W. T. Freeman. Face hallucination: theory and practice. *International Journal of Computer Vision (IJCV)*, 75(1):115–134, 2007. 36

[66] C. Liu, R. Szeliski, S. B. Kang, C. L. Zitnick, and W. T. Freeman. Automatic estimation and removal of noise from a single image. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 30(2):299–314, 2008. 36

[67] C. Liu, A. Torralba, W. T. Freeman, F. Durand, and E. H. Adelson. Motion magnification. In *ACM SIGGRAPH*, pages 519–526, 2005. 13, 32, 34, 40, 77

[68] C. Liu, J. Yuen, and A. Torralba. Nonparametric scene parsing: Label transfer via dense scene alignment. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009. 36, 105, 123

[69] C. Liu, J. Yuen, A. Torralba, J. Sivic, and W. T. Freeman. SIFT flow: dense correspondence across different scenes. In *European Conference on Computer Vision (ECCV)*, 2008. 19, 101, 103, 126, 127, 129

[70] D. G. Lowe. Object recognition from local scale-invariant features. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1150–1157, Kerkyra, Greece, 1999. 36, 94, 97, 98, 100, 104, 117, 122

[71] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 674–679, 1981. 17, 29, 37, 39, 50, 64, 79, 81, 84, 93, 96, 122, 146

[72] D. Mackay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003. 89

[73] S. Mahamud, L. Williams, K. Thornber, and K. Xu. Segmentation of multiple salient closed contours from real images. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 25(4):433–444, 2003. 85

[74] D. Martin, C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 26(5):530–549, May 2004. 83

[75] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *IEEE International Conference on Computer Vision (ICCV)*, pages 416–423, 2001. 40

[76] J. McDermott. Psychophysics with junctions in real images. *Perception*, 33:1101–1127, 2004. 81, 82

[77] J. McDermott and E. H. Adelson. The geometry of the occluding contour and its effect on motion interpretation. *Journal of Vision*, 4(10):944–954, 2004. 87

[78] J. McDermott and E. H. Adelson. Junctions and cost functions in motion interpretation. *Journal of Vision*, 4(7):552–563, 2004. 81

[79] P. Meer. Robust techniques for computer vision. *Emerging Topics in Computer Vision*, pages 107–190, 2004. 144

[80] A. Nobel. *Descriptions of Image Surfaces*. PhD thesis, Oxford University, Oxford, UK, 1989. 64

[81] S. J. Nowlan and T. J. Sejnowski. A selection model for motion processing in area mt primates. *The Journal of Neuroscience*, 15(2):1195–1214, 1995. 82

[82] A. Oliva and A. Torralba. Modeling the shape of the scene: a holistic representation of the spatial envelope. *International Journal of Computer Vision (IJCV)*, 42(3):145–175, 2001. 22, 23, 97, 98, 105, 120, 126, 127, 128, 131

[83] P. Pérez, M. Gangnet, and A. Blake. Poisson image editing. *ACM SIGGRAPH*, 22(3):313–318, 2003. 116

[84] K. Pullen and C. Bregler. Motion capture assisted animation: Texture and synthesis. *ACM Transactions on Graphics*, 21:501–508, July 2002. 60

[85] R. Raskar, K.-H. Tan, R. Feris, J. Yu, and M. Turk. Non-photorealistic camera: depth edge detection and stylized rendering using multi-flash imaging. *ACM Trans. Graph. (SIGGRAPH)*, 23(3):679–688, 2004. 83

[86] X. Ren, C. Fowlkes, and J. Malik. Scale-invariant contour completion using conditional random fields. In *Proceedings of International Conference on Computer Vision*, pages 1214–1221, 2005. 85

[87] S. Roth and M. Black. On the spatial statistics of optical flow. *International Journal of Computer Vision (IJCV)*, 74(1):33–50, 2007. 40, 56, 146

[88] C. Rother, V. Kolmogorov, and A. Blake. Interactive foreground extraction using iterated graph cuts. In *Proceedings of ACM SIGGRAPH 2004*, pages 309–314, July 2004. 70, 71

[89] C. Rother, T. Minka, A. Blake, and V. Kolmogorov. Cosegmentation of image pairs by histogram matching - incorporating a global constraint into mrfs. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 993–1000, 2006. 98

[90] B. C. Russell, A. Torralba, C. Liu, R. Fergus, and W. T. Freeman. Object recognition by scene alignment. In *Advances in Neural Information Processing Systems (NIPS)*, 2007. 36, 105, 126

[91] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman. LabelMe: a database and web-based tool for image annotation. *International Journal of Computer Vision (IJCV)*, 77(1-3):157–173, 2008. 40, 94, 126, 127, 129

[92] M. Ruzon and C. Tomasi. Alpha estimation in natural images. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'00)*, pages 24–31, 2000. 70

[93] F. Samaria and A. Harter. Parameterization of a stochastic model for human face identification. In *IEEE Workshop on Applications of Computer Vision*, 1994. 22, 119, 120

[94] P. Sand and S. Teller. Video matching. In *Proceedings of ACM SIGGRAPH 2004*, pages 592–599, 2004. 29, 64, 70

[95] P. Sand and S. J. Teller. Particle video: long-range motion estimation using point trajectories. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 2195–2202, 2006. 29, 30, 39, 41

[96] E. Saund. Logic and MRF circuitry for labeling occluding and thinline visual contours. In *Advances in Neural Information Processing Systems 18*, pages 1153–1160, 2006. 85

[97] H. Sawhney, Y. Guo, K. Hanna, R. Kumar, S. Adkins, and S. Zhou. Hybrid stereo camera: An ibr approach for synthesis of very high resolution stereoscopic image sequences. In *Proceedings of ACM SIGGRAPH 2001*, pages 451–460, 2001. 65

[98] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision (IJCV)*, 47(1):7–42, 2002. 19, 93, 95

[99] C. Schmid, R. Mohr, and C. Bauckhage. Evaluation of interest point detectors. *International Journal of Computer Vision (IJCV)*, 37(2):151–172, 2000. 97, 122

[100] A. Schodl, R. Szeliski, D. Salesin, and I. Essa. Video textures. In *Proceedings of ACM SIGGRAPH 2000*, pages 489–498, 2000. 60

[101] A. Shahua and S. Ullman. Structural saliency: the detection of globally salient structures using a locally connected network. In *Proceedings of International Conference on Computer Vision*, pages 321–327, 1988. 85, 87

[102] G. Shakhnarovich, P. Viola, and T. Darrell. Fast pose estimation with parameter sensitive hashing. In *IEEE International Conference on Computer Vision (ICCV)*, 2003. 126

[103] A. Shekhovtsov, I. Kovtun, and V. Hlavac. Efficient MRF deformation model for non-rigid image matching. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007. 97, 101, 122

[104] J. Shi and J. Malik. Motion segmentation and tracking using normalized cuts. In *Proceedings of International Conference on Computer Vision*, pages 1154–1160, 1998. 69

[105] J. Shi and C. Tomasi. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994. 50, 64, 81

[106] J. Shotton, J. Winn, C. Rother, and A. Criminisi. Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *European Conference on Computer Vision (ECCV)*, 2006. 23, 127, 128, 129, 132, 133

[107] J. Sivic and A. Zisserman. Video Google: a text retrieval approach to object matching in videos. In *IEEE International Conference on Computer Vision (ICCV)*, 2003. 30, 104, 125

[108] G. Strang. *Introduction to Applied Mathematics*. Wellesley-Cambridge Press, 1986. 68

[109] E. Sudderth, A. Torralba, W. T. Freeman, and W. Willsky. Describing visual scenes using transformed dirichlet processes. In *Advances in Neural Information Processing Systems (NIPS)*, 2005. 30, 125

[110] J. Sun, N. Zheng, , and H. Shum. Stereo matching using belief propagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 25(7):787–800, 2003. 97

[111] M. J. Swain and D. H. Ballard. Color indexing. *International Journal of Computer Vision (IJCV)*, 7(1), 1991. 97

[112] R. Szeliski. Fast surface interpolation using hierarchical basis functions. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 12(6):513–528, 1990. 50

[113] R. Szeliski. Image alignment and stiching: A tutorial. *Foundations and Trends in Computer Graphics and Computer Vision*, 2(1), 2006. 50, 93, 96

[114] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother. A comparative study of energy minimization methods for markov random fields with smoothness-based priors. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 30(6):1068–1080, 2008. 97, 101

[115] M. Tappen, C. Liu, W. T. Freeman, and E. H. Adelson. Learning Gaussian conditional random fields for low-level vision. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2007. 36, 41

[116] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: A factorization method. *International Journal of Computer Vision (IJCV)*, 9(2):137–154, Nov 1992. 29

[117] P. H. S. Torr, R. Szeliski, and P. Anandan. An integrated Bayesian approach to layer extraction from image sequences. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 23(3):297–304, 2001. 40

[118] A. Torralba and W. T. Freeman R. Fergus. 80 million tiny images: a large dataset for non-parametric object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 30(11):1958–1970, 2008. 97, 126

[119] M. Unuma, K. Anjyo, and R. Takeuchi. Fourier principles for emotion-based human figure animation. In *Proceedings of ACM SIGGRAPH 95*, pages 91–96, July 1995. 60

[120] P. Viola and W. Wells III. Alignment by maximization of mutual information. In *IEEE International Conference on Computer Vision (ICCV)*, pages 16–23, 1995. 96

[121] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001. 30, 125

[122] J. Wang, P. Bhat, A. Colburn, M. Agrawala, and M. Cohen. Interactive video cutout. In *ACM SIGGRAPH*, 2005. 40

[123] J. Y. A. Wang and E. H. Adelson. Representing moving images with layers. *IEEE Trans. Image Processing*, 3(5):625–638, 1994. 30, 60, 63

[124] J. Y. A. Wang and E. H. Adelson. Representing moving images with layers. *IEEE Transactions on Image Processing (TIP)*, 3(5):625–638, 1994. 37, 40

[125] Y. Weiss. Interpreting images by propagating bayesian beliefs. In *Advances in Neural Information Processing Systems (NIPS)*, pages 908–915, 1997. 97

[126] Y. Weiss. Smoothness in layers: Motion segmentation using nonparametric mixture estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 520–527, 1997. 97

[127] Y. Weiss and E. H. Adelson. Perceptually organized EM: A framework for motion segmentaiton that combines information about form and motion. Technical Report 315, M.I.T Media Lab, 1995. 17, 30, 40, 81, 82, 83

[128] J. Wills, S. Agarwal, and S. Belongie. What went where. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 37–44, 2003. 40, 71

[129] J. Winn and N. Jojic. Locus: Learning object classes with unsupervised segmentation. In *IEEE International Conference on Computer Vision (ICCV)*, pages 756–763, 2005. 94

[130] G. Yang, C. V. Stewart, M. Sofka, and C.-L. Tsai. Registration of challenging image pairs: Initialization, estimation, and decision. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 29(11):1973–1989, 2007. 21, 116, 118, 119

[131] L. Zelnik-Manor and M. Irani. Degeneracies, dependencies and their implications in multi-body and multi-sequence factorizations. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'03)*, pages 287–293, 2003. 69