Video Motion: Finding Complete Motion Paths for Every Visible Point

by

Susanna Maria Ricco

Department of Computer Science
Duke University

Date: __________________________
Approved:

____
Carlo Tomasi, Supervisor

____
Ronald Parr

____
Mauro Maggioni

____
Svetlana Lazebnik

Dissertation submitted in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in the Department of Computer Science
in the Graduate School of Duke University
2013

Abstract

Video Motion: Finding Complete Motion Paths for Every Visible Point

by

Susanna Maria Ricco

Department of Computer Science
Duke University

Date: ____________________

Approved:

____
Carlo Tomasi, Supervisor

____
Ronald Parr

____
Mauro Maggioni

____
Svetlana Lazebnik

An abstract of a dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Computer Science in the Graduate School of Duke University
2013

Abstract

The problem of understanding motion in video has been an area of intense research in computer vision for decades. The traditional approach is to represent motion using optical flow fields, which describe the two-dimensional instantaneous velocity at every pixel in every frame. We present a new approach to describing motion in video in which each visible world point is associated with a sequence-length video motion path. A video motion path lists the location where a world point would appear if it were visible in every frame of the sequence. Each motion path is coupled with a vector of binary visibility flags for the associated point that identify the frames in which the tracked point is unoccluded.

We represent paths for all visible points in a particular sequence using a single linear subspace. The key insight we exploit is that, for many sequences, this subspace is low-dimensional, scaling with the complexity of the deformations and the number of independent objects in the scene, rather than the number of frames in the sequence. Restricting all paths to lie within a single motion subspace provides strong regularization that allows us to extend paths through brief occlusions, relying on evidence from the visible frames to hallucinate the unseen locations.

This thesis presents our mathematical model of video motion. We define a path objective function that optimizes a set of paths given estimates of visible intervals, under the assumption that motion is generally spatially smooth and that the appearance of a tracked point remains constant over time. We estimate visibility based on global properties of all paths, enforcing the physical requirement that at least one tracked point must be visible at every pixel in the video. The model assumes the existence of an appropriate path motion basis; we find a sequence-specific basis through analysis of point tracks from a frame-to-frame tracker. Tracking failures caused by image noise, non-rigid deformations, or occlusions complicate the problem by introducing missing data. We update standard trackers to aggressively reinitialize points lost in earlier frames. Finally, we improve on standard Principal Component Analysis with missing data by introducing a novel compaction step that associates these relocalized points, reducing the amount of missing data that must be overcome. The full system achieves state-of-the-art results, recovering dense, accurate, long-range point correspondences in the face of significant occlusions.

To Mac, who makes things better.

Contents

List of Tables

List of Figures

List of Algorithms

List of Abbreviations and Symbols

Symbols

Throughout this document, we represent vector-valued quantities with bold, lowercase symbols; scalar values are not bold. Matrices are both bold and uppercase. Whenever possible, we use f or t to refer to frame indices; other letters index tracked points.

Here we collect the most important quantities for video motion.

$\boldsymbol{x}_p(t)$	The location of point p in frame t as defined by the video motion path.
$\nu_p(t)$	The binary visibility state of point p in frame t; equal to one if visible.
$(\boldsymbol{u}_p, \tau_p)$	The anchor point for point p. τ_p is the point's anchor frame. The path must pass through $\boldsymbol{u}_p$ in this frame, that is, $\boldsymbol{x}_p(\tau_p) = \boldsymbol{u}_p$.
$\boldsymbol{\varphi}_k$	A path basis function.

Abbreviations

SFM	Structure-from-motion.
MRF	Markov Random Field.

Acknowledgements

A complete list of those that deserve my thanks would surely double the length of this document. Most I will thank in person, but a few deserve special recognition.

Thank you to my advisor, Carlo Tomasi, for the years of support and encouragement. Carlo taught me to ask big questions, to embrace the struggle for answers without giving up, and to value deep insights over incremental advances.

Thank you to my committee: Ron Parr, for adopting me as an almost-member of his research group; Svetlana Lazebnik, for her extraordinarily detailed comments on early drafts; and Mauro Maggioni, for stepping in on short notice to allow for a non-empty intersection of busy summer schedules.

I have been blessed with outstanding teachers at every school I have attended. I would especially like to thank Zach Dodds and Christine Alvarado for introducing me to computer science and encouraging me to pursue research in computer vision.

Thank you to my fellow graduate students for making Duke a wonderful place to spend these many years. In particular, thank you to Neeti Wagle, for complaining about the rain with me, and Gavin Taylor, for giving us a hard time about our complaining.

Thank you to my officemates past and present. Cassi Carley and her remote-controlled ghost computer have kept me company on many a late night. Jason Pazis graciously surrendered his desk to my expanding disorganization.

Thank you to the departmental staff for organizing travel to conferences and generally keeping all the gears turning. Special thanks go to Pam Spencer for the advance notice any time there was extra food to be found.

None of my accomplishments would have been possible without the unwavering support of my family, both old and new. Thank you to my sister for reminding me not to take myself too seriously. Thank you to my mother, the scientist, for being a source of inspiration all my life, and to my father, the recovering scientist, for keeping me grounded.

Finally, my thanks to the following agencies that have provided financial support for my work: Duke University, the National Science Foundation (through an NSF Graduate Research Fellowship, grant IIS-0534897, and grant IIS-1017017), and the Army Research Office grant no. W911NF-10-1-0387.

1 Introduction

Chances are good that the next person I meet will have a video camera within easy reach (although it will likely be disguised as a cell phone). People use these devices constantly to document life's events, from the momentous to the mundane. The amount of data being generated is staggering, with recent estimates suggesting over 72 hours of video are uploaded to YouTube *every minute* (YouTube, 2013). Associated Press photographer Michael Sohn's now-viral image of the faithful gathered to witness the introduction of the newly elected Pope Francis testifies to the ubiquitous presence of cameras in our modern world. The photo of the crowd blurs into a sea of bright lights, as if each individual holds a candle in the night. In fact, the lights are the screens of thousands of smartphones, capturing the historic moment.

For the computer vision researcher, the explosion of video data presents a challenge: understand the rich and complex motions contained within video sequences of even moderate length. For decades, understanding motion meant extracting optical flow. An optical flow field describes the (apparent) instantaneous velocity at every pixel in a given frame. Each pixel stores a vector pointing to its matching location in the next frame.

But typical videos consist of more than just two frames. As a scene is imaged over a longer period of time, the projection of a single world point traces a curve in the image plane. The shape of this curve is determined by the evolving position of the point, the position of the camera, and the intrinsic properties of the camera such as focal length and radial distortion. In the absence of occlusions, the points on the curve are the images of the tracked point in each frame. An occlusion occurs when another world point is located closer to the camera on the same projection ray. When this happens, the image of the occluded point is hidden from view and only the image of the visible point appears. In some frames, the projection of the scene point may fall outside the boundaries of the image. The scene point will not appear in these frames even if it is not occluded by another physical object. From here on, we refer to these curves traced by moving points as *paths*. We will formalize the definition mathematically in Chapter 3.

The goal of video motion estimation is to extract sequence-length paths for a dense sampling of the visible surfaces in a scene. This separates the appearance of the objects from their motion, permitting further analysis that disregards irrelevant variation in appearance. Extracted paths should be robust to transient occlusions so that points on objects that are momentarily occluded are connected across the occlusion. The sampling of surfaces in the scene should be dense enough so that there is some visible path within a small neighborhood of each pixel.

Simply applying established methods for optical flow computation fails to generate the desired paths. Paths built by concatenating optical flow vectors between consecutive pairs of frames cannot follow points across occlusions because optical flow fields do not capture the motion of points that are not visible in the source frame. Paths created by computing optical flow from the first to every subsequent frame ($1 \rightarrow 2$, $1 \rightarrow 3$, $1 \rightarrow 4$, etc.) could span occlusions, but optical flow estimation fails when too much motion accumulates between distant frames. Truly reliable

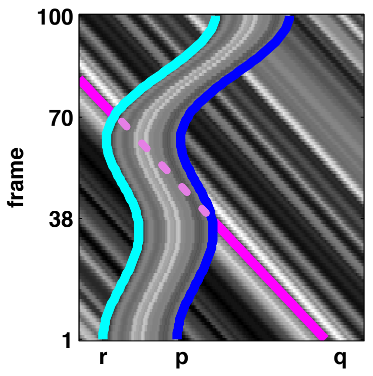


FIGURE 1.1: An illustration of the video motion concepts. We show a slice through a single row of a synthetic video sequence that contains only horizontal motion. Video motion paths follow lines of constant intensity; we highlight three in color. Paths of occluded points (magenta) extend through detected occluded intervals. The full solution includes a visible path through every pixel in every frame.

recovery of long-range video motion paths requires new solution techniques.

This dissertation proposes a solution to the problem of video motion estimation with particular focus on the treatment of occlusions. Many motion estimation techniques treat occlusions as noise. In contrast, we promote occlusions to first-class citizens and model visible intervals explicitly for each path. Paths extend through occlusion regions and allow for hallucinating the location of momentarily occluded points. Because occlusions are actually quite common in general video, correctly handling occlusions is crucial.

Figure 1.1 illustrates the key concepts in our work. For simplicity, we consider a hypothetical sequence that contains only horizontal motion. Paths are curves in the image plane, parameterized by frame. We can visualize these paths by lifting them into space-time, so that the evolution over time is drawn explicitly in an added spatial dimension. Because our toy sequence does not include any vertical motion, each of these lifted curves is contained in an xt-slice through the space-time volume; one xt-slice is shown in the figure, with time increasing in the vertical direction. Here, a translating occluder temporarily obscures a portion of a moving background.

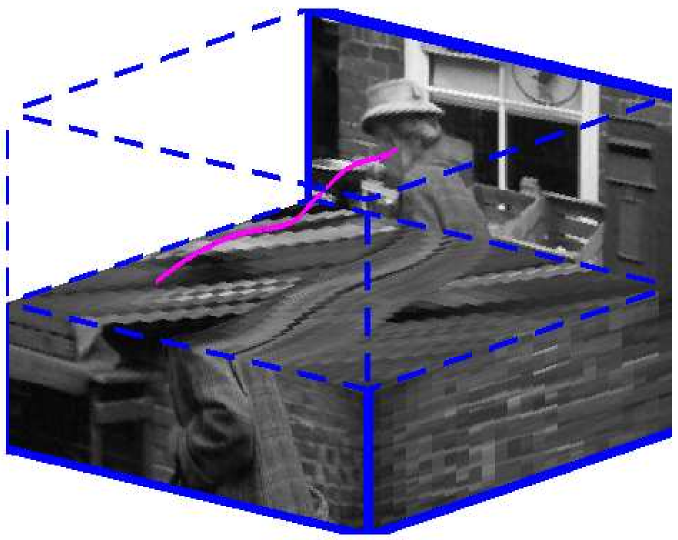


FIGURE 1.2: An example path extracted from general video, including both horizontal and vertical motion. In this sequence, the camera pans to follow Miss Marple as she walks from right to left. The path drawn in magenta follows Miss Marple's ear. The cube shows the bottom half of the first frame connected to the top half of the last frame by an xt-slice. Note the significant occlusions visible in the crossing intensity contours in this slice. This sequence will be used as a running example throughout this dissertation.

We single out the paths of three particular scene points: p, q, and r. An occlusion occurs in frame 38, where the paths for p (blue) and q (magenta) intersect. The point p is on the right edge of the occluding object, while q is occluded. We denote this by drawing the path q with a dashed rather than solid line over the next few frames. The point r (cyan) is on the opposite side of the translating occluder. Its path intersects the path of point q in frame 70, after which q reappears. This corresponds to a disocclusion event. After frame 70, the path q is again drawn with a solid line to indicate that the point is visible. Note that the two visible intervals are connected by the estimates of q's location during the occlusion. A potential dense sampling of the visual surfaces in this example scene generates the rest of the paths drawn in shades of gray. Note that every point in this slice has a visible path nearby.

Because real sequences do contain non-zero vertical motion, the paths we solve for live in a three-dimensional space-time volume. Figure 1.2 shows just one of the more than 100,000 video motion paths we extract from one of our test sequences. This figure also shows an xt-slice in context within the space-time volume. Note the significant occlusions visible in the xt-slice. Nearly every background point is either occluded or out of the field of view in at least one frame.

If we had complete knowledge of the scene, consisting of a geometric model, its evolution over time, the relative location of the camera in each frame, and the camera's intrinsic properties, we would be able to construct paths and reason about occlusions through computer graphics, rendering the current scene at each frame. Of course, it is entirely unreasonable to assume we have access to all this hidden information. Our method bypasses scene reconstruction and instead extracts paths directly from input video. Although we do not attempt reconstruction, the multiframe correspondences we find could be used as input to a structure-from-motion routine to recover 3D structure if desired.

The dense multiframe correspondences that video motion paths supply support many other high-level vision applications. Paths provide information about the long-term interactions between objects that can be used to segment a scene. The extended time frames allow for reasoning about occlusion relationships and create more discriminative features by integrating small instantaneous differences to form very distinct paths. Segmenting video eases object recognition by eliminating confusing clutter. If recognition is successful in any single frame, the resulting labels (or other human annotations) can be propagated along the path to any frame in the sequence. In other domains, the motion itself may be a key discriminative feature for recognition. Examples include medical imaging, gesture recognition, and event or activity recognition. Long-term correspondences are beneficial when the events to recognize have non-trivial temporal extent.

Our approach to video motion estimation relies on a few fundamental assumptions that we outline here.

Appearance is constant.

We assume that the appearance of a world point does not change over time. Because we work with grayscale video, this means that the intensity sampled at points along the visible portions of a trajectory's path should be constant. There are many situations where this assumption is not valid: it rules out specular surfaces, ignores shadows, and disallows global lighting changes that affect the average illumination of the scene. Nonetheless, we find this assumption is reasonable for many videos lasting as long as a few seconds. Unmodeled brightness changes are included as part of the image noise process and dealt with using a robust penalty function.

Motion discontinuities are continuous.

We assume that motions and occlusions are spatially smooth. Nearby points on a single object will tend to follow similar paths because the distance between the two paths in any particular frame is constrained by the real-world distance between the points. As a result, two points visible in neighboring pixels in a particular frame will tend to follow essentially parallel paths. Because the paths are similar, they are likely to be occluded or visible at the same time. This assumption breaks down at when points on different objects happen to appear in neighboring pixels for a fleeting moment. We accommodate these rare motion discontinuities by again using robust penalty functions.

Surfaces are opaque.

We assume that there are no semi-transparent objects in the scene. Semi-transparent objects would allow a single pixel to contain more than one visible object each of which would follow potentially very different paths.

Paths are correlated.

The crucial assumption we make is that the paths of all visible points in the scene are highly correlated. Paths for any sequence lasting F frames can be represented using vectors of length $2F$. We assume that all paths for the given sequence actually lie within a K-dimensional linear subspace where K is much smaller than $2F$. This assumption makes it computationally feasible to estimate paths for long sequences by decreasing the dimensionality of the search space for paths from $2FN$ to KN (where N is the number of visible points being tracked). This assumption is also what makes it possible to extend paths through brief occlusions. Evidence from frames where the point is visible can be used to constrain its location during occlusions.

1.1 Motion Estimation Procedure

This section provides a high-level overview of our approach to extracting video motion by finding paths and visibility estimates. It also outlines the remainder of the document.

We begin with a review of past work in motion estimation in Chapter 2. Chapter 3 describes our mathematical model of video motion. Our formulation improves upon our initial proposal in Ricco and Tomasi (2012a). We represent paths using a low-dimensional path basis and attach them to world points by requiring that they each pass through a different reference pixel in the video sequence while visible. By selecting enough of these anchor points, we can ensure that all pixels in the video sequence are covered by a nearby visible path. We define an objective function that penalizes changes in the appearance of tracked points as well as differences in the motion of nearby points. The best paths minimize this objective function.

The penalty for changes in appearance only applies when a path is considered visible. Our visibility estimates are based on global properties of all paths. We determine an initial estimate of the visible path at each pixel by measuring brightness constancy in a small patch aligned with the motion of each path over a few frames. The most consistent path at each pixel is considered visible, as are neighboring paths that differ only slightly over the course of the sequence. This initial estimate is refined using a global objective function formulated as a Markov Random Field (MRF) that encourages spatial smoothness (nearby points with similar paths get occluded at the same time) and temporal smoothness (changes in visibility along a path are rare). We require at least one visible path near every pixel by forcing the most visually consistent path at each pixel to be visible in the final solution.

Chapters 4 and 5 describe the process of solving for motion given this model. Given an input video, the first step is to determine the sequence-specific subspace we will use to parameterize paths. We track a representative set of points using a frame-to-frame tracker that aggressively attempts to relocalize points lost to occlusions in earlier frames. We supplement these tracks with tracks found using frame-to-frame optical flow to improve coverage in low-texture areas. We find a low-dimensional linear subspace that can adequately reconstruct the frame-to-frame tracks using Principal Component Analysis (PCA) with missing data. This process was presented in Ricco and Tomasi (2012b) and is described in detail in Chapter 4.

With the path basis fixed, we concatenate frame-to-frame flow fields into extended tracks and project these onto the path subspace to create initial path estimates. We attach these path estimates to an initial set of anchor points. We then alternate between estimating which points are visible in each frame (assuming our estimate of their paths is correct) and refining the estimates of the paths (assuming our visibility estimates are correct). When paths do not change between iterations, we reconsider the points we selected, adding or deleting points to track until the video coverage is satisfactory.

Our path objective function is non-convex and our initialization is often poor, especially near occlusion boundaries. To alleviate this problem, we apply a heuristic update after a fixed number of descent steps. The update replaces paths and visibility estimates with alternatives from nearby points if the change results in better brightness constancy. Chapter 5 details the complete optimization routine from initialization to convergence.

The remainder of this document contains our experimental results (Chapter 6), a discussion of topics that warrant further investigation (Chapter 7), and some concluding remarks (Chapter 8).

1.2 Summary of Contributions

The main technical contributions of this thesis are as follows:

1. We present a formal definition of long-range video motion through paths with explicit treatment of occlusions.

2. We develop a technique to improve the estimation of the path subspace for sequences with significant occlusions. The technique can be fully automated or can run with limited user interaction. The extracted subspace improves extrapolation of paths into distant frames and increases robustness to incorrect estimates of the dimensions of the subspace.

3. We introduce a technique to determine the visibility of points based on global properties of all paths. The estimates are causal, identifying the occluder responsible for each occlusion.

4. We provide a complete pipeline that extracts full-length paths from input video, guaranteeing dense coverage of all visible points and achieving state-of-the-art performance.

2

Prior Work in Motion Estimation

Motion estimation is, in general, the problem of solving for pixel-level correspondences between images evolving in time. Given a sequence of images $I_1, I_2, \ldots, I_T$, the goal is to associate particular locations in I_{t_1} with the matching locations in I_{t_2}. Approaches differ in what and how many locations are matched between images, the number of images expected in the sequence, and the techniques used to find the correspondences themselves. It is assumed, however, that the images to match are provided in temporal order and that the difference in time between two consecutive images is small. This assumption distinguishes motion estimation from the broader field of image registration, where the goal is to find correspondences between pixels in images captured of similar but not necessarily the same scene or object without any knowledge of the temporal relationship between the images.

2.1 The Brightness Change Constraint Equation

The main assumption in motion estimation is that the appearance of a point in space does not change as a result of motion. That is, if we assume that the images we see are a discrete sampling of a continuous function $I(x, y, t)$, the brightness constancy assumption states that the total derivative with respect to time should be zero:

$$\frac{dI}{dt} = I_x \frac{dx}{dt} + I_y \frac{dy}{dt} + I_t = 0 \ . \tag{2.1}$$

In the second expression, the result of applying the chain rule to compute the total derivative, subscripts denote partial differentiation. It is possible to estimate the partial derivatives I_x, I_y, and I_t from the image sequence using finite differences. The values $u = \frac{dx}{dt}$ and $v = \frac{dy}{dt}$ describe the 2D image motion of points in the scene and thus specify pixel correspondences. Specifically, for small motions between frames, the location (x, y) in frame t approximately matches location $(x + u, y + v)$ in frame $t+1$.

It is important to note that equation (2.1) is almost never exactly satisfied, even for infinitesimally small motions between frames. First, the reflectance at a point on a Lambertian surface (and therefore the observed brightness at this point) is a function of the angle between the surface normal and the illuminating light source, so a point's appearance can change under motions that change this angle. For non-Lambertian surfaces, the appearance can also change due to specular reflection. A point can pass through a shadow, resulting in a change of appearance due to a change in the intensity of the illuminating light source. Occlusions (and disocclusions) result in points not merely changing their appearance but actually disappearing from view (or reappearing). Finally, the imaging process itself introduces noise, caused by errors in measurement as well as discretization and quantization. In particular, using discrete approximations of the derivatives introduces significant error when the motion is large relative to the frame rate. In spite of all these potential sources of error, equation (2.1) is still approximately satisfied in many real sequences.

Equation (2.1) is a single equation with two unknowns and is therefore not enough to recover the full motion at any given point. Instead, it constrains the component of the velocity in the direction of the image gradient at each pixel. Any values of u and

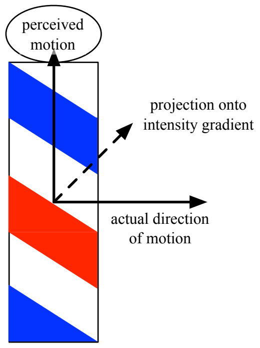

perceived motion
projection onto intensity gradient
actual direction of motion

FIGURE 2.1: A barbershop pole demonstrates the aperture problem. Although the objective motion is purely horizontal, any motion with the same component normal to the isointensity contours is indistinguishable. Humans tend to perceive motion parallel to the dominant direction of the surrounding aperture (in this case, vertical).

v that differ by motions along isocontours of image intensity are indistinguishable. This is referred to as the *aperture problem*, and occurs in the human visual system as well. A well-known example is the barbershop pole illusion (Guilford, 1929), depicted in Figure 2.1, in which a pattern of diagonal lines is actually translating horizontally as the pole rotates. However, the viewer perceives that the lines are translating vertically, parallel to the vertically elongated aperture created by the curvature of the pole (Wallach, 1935). The two vectors, pure horizontal motion and pure vertical motion, have the same projection onto the line in the direction of the gradient. Interestingly, if one views a barbershop pole through a pinhole, one will tend to perceive diagonal motion in the direction of the gradient only. Mathematically, there would be no reason to prefer horizontal, diagonal, or vertical motion over the others without additional constraints on the solution.

2.2 KLT Tracking

One technique for constraining the solution to equation (2.1) is to assume that the motion field is constant within a small spatial neighborhood. This technique was proposed in 1981 and is still the basis for the widely used KLT tracker (Lucas and

Kanade, 1981; Tomasi and Kanade, 1991). The goal is to find the displacement u and v that minimizes

$$\sum_{x,y \in \Omega} \left[u I_x(x,y,t) + v I_y(x,y,t) + I_t(x,y,t) \right]^2, \tag{2.2}$$

where Ω defines a spatial neighborhood. This is equivalent to solving this system of equations

$$\begin{bmatrix} I_{x1} & I_{y1} \\ I_{x2} & I_{y2} \\ \vdots & \vdots \\ I_{xn} & I_{yn} \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -I_{t1} \\ -I_{t2} \\ \vdots \\ -I_{tn} \end{bmatrix} \tag{2.3}$$

in the least-squares sense, where I_{xi} is the partial derivative with respect to x evaluated at pixel i in the neighborhood Ω. I_{yi} and I_{ti} are the corresponding values when the partial derivative is taken with respect to y and t. The solution is simply

$$\begin{bmatrix} u \\ v \end{bmatrix} = (\boldsymbol{A}^T \boldsymbol{A})^{-1} \boldsymbol{A}^T \boldsymbol{b} \tag{2.4}$$

with

$$\boldsymbol{A}^T \boldsymbol{A} = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}, \qquad \boldsymbol{A}^T \boldsymbol{b} = \begin{bmatrix} -\sum I_x I_t \\ -\sum I_y I_t \end{bmatrix}. \tag{2.5}$$

The solution is unique when the 2×2 matrix $\boldsymbol{A}^T \boldsymbol{A}$, called the *structure tensor*, is invertible. Thus, rather than tracking arbitrary image locations, a KLT tracker is usually restricted to tracking only *good features to track* (Shi and Tomasi, 1994), defined as points where the second singular value of $\boldsymbol{A}^T \boldsymbol{A}$ exceeds a defined threshold. These locations correspond to highly textured regions like corners.

A complete KLT tracker includes two additional steps. First, because equations (2.1) and (2.2) hold for infinitesimal displacements but are used for finite motions, the displacement is found through an iterative process. At each iteration, the temporal derivatives I_{ti} are recomputed after warping the second image according to the current estimate of motion (u, v). The solution to equation (2.4) is then $(\delta u, \delta v)$, a small update to the current motion estimate. Second, the quality of proposed matches between the patches centered at (x, y) in frame t and $(x + u, y + v)$ in frame $t+1$ is monitored by measuring the sum of squared differences between the two patches after they are aligned with the optimal affine transformation between them. Features with residuals exceeding a given threshold are considered lost and the correspondence is rejected. Features are often lost either because the assumption that motion is constant across the patch is violated (that is, the patch could straddle an object boundary or there could simply be too much non-rigid deformation occurring), or because the feature becomes occluded. Lost features are replaced by new points according to the good features to track criteria.

We use a modified KLT tracker when finding a path basis for each sequence. We improve on the standard feature reinitialization procedure by adding a step that actively searches for and reacquires lost features.

2.3 Optical Flow

While KLT tracking provides motion estimates for a discrete set of selected points with sufficient texture, optical flow defines the displacement between two consecutive images using *continuous* flow fields $u(x, y)$ and $v(x, y)$. The estimates of motion at different pixels (with non-overlapping neighborhoods) in a KLT tracker are independent of each other. In optical flow, motion estimates at distant locations are coupled through a spatial regularizer that eliminates the ambiguity in the solution caused by the aperture problem. Our definition of video motion replaces optical flow with a model that includes sequence-length correspondences. Like optical flow, we assume brightness constancy holds and enforce spatial smoothness to address the aperture problem.

In the original optical flow formulation, Horn and Schunck (1981) assume that the desired motion field varies smoothly in space. They minimize an energy functional

$$E = \iint (I_x u + I_y v + I_t)^2 + \lambda \left[\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2 \right] dx\, dy \ , \quad (2.6)$$

where λ is a non-negative regularizing parameter, balancing the relative importance of the spatial smoothness of the flow field versus the data term that penalizes violations of the brightness change constraint equation.

The minimizing optical flow field is found using the calculus of variations by solving the corresponding Euler-Lagrange equations with natural boundary conditions. The Euler-Lagrange equations for this functional are

$$\begin{aligned} I_x(I_x u + I_y v + I_t) - \lambda^2 \Delta u &= 0 \\ I_y(I_x u + I_y v + I_t) - \lambda^2 \Delta v &= 0 \ , \end{aligned} \quad (2.7)$$

where $\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$ is the Laplacian operator. Approximating partial derivatives with finite differences results in a sparse (banded) system of linear equations that can be solved using an iterative scheme such as the Jacobi method or successive over-relaxation.

Hundreds of researchers have followed in Horn and Schunck's footsteps, refining the original formulation to address its limitations. One of the main problems with the formulation is its sensitivity to outliers. Unmodeled changes in brightness due to shadows or specular reflections cause large errors in the data term. Occlusions result in outliers for both the data term and the smoothness term. Occlusions occur precisely at those locations where two neighboring objects in the image follow different motions, causing large violations of the smoothness assumption. Although occluded points by definition have no corresponding points in the second frame, the optical flow field forces a correspondence, which is almost certain to violate the brightness constancy assumption. In a recent paper, Sun et al. (2010b) present a concise review of many of the proposed advances and evaluate their impact. Most modern methods replace the quadratic penalties on both the data and smoothness term with robust penalty functions, add new data terms penalizing changes in the color or image gradient along with changes in brightness, and use structure-texture decomposition (Aujol et al., 2006) as a preprocessing step to remove shadows.

Two remaining areas of research involve finding techniques for dealing with large motions and improving the efficiency of the optimization procedure to enable real-time computation. One example of a modern optical flow method that improves on Horn and Schunck's original formulation in all three areas is Large Displacement Optical Flow (LDOF) (Brox et al., 2009; Brox and Malik, 2011). A fast, GPU-accelerated version (Sundaram et al., 2010) is publicly available as an executable distributed by the authors. We review the details here as a case study and because we use its optical flow fields to provide an initial solution for our path estimates.

Before LDOF, the standard approach for dealing with large motions within the variational framework introduced by Horn and Schunck was to estimate optical flow using a coarse-to-fine pyramid (Anandan, 1989; Bergen et al., 1992; Black and Anandan, 1996; Brox et al., 2004). The images are successively down-sampled to form a scale-space pyramid, and optical flow is estimated at the coarsest scale first. At the coarsest level, large displacements at full resolution become small – small enough that the local linear approximation used in the data term is reasonable – and good estimates of the motion can be found starting from an initial estimate of zero motion. The solution at a coarse scale is used to initialize motion estimates at the next level of the pyramid, which are then refined using the new details revealed at the finer scale. As with KLT tracking, the solution is found iteratively, with the temporal derivative recomputed after each iteration according to the current motion estimates. The process continues until motion estimates are propagated back up to and refined at the original resolution of the images.

This process can recover large displacements for large objects, but it can miss smaller objects that are smoothed away at the lower resolutions. To solve this problem, LDOF introduces a term into the objective function that encourages final optical flow estimates to agree with sparse correspondences estimated by matching regions. Regions could be matched by estimating correspondences between keypoints detected by SIFT (Lowe, 2004), SURF (Bay et al., 2008), BRISK (Leutenegger et al., 2011), or other similar algorithms. LDOF proposes using descriptors based on histograms of oriented gradients sampled on a dense grid in both images. Tentative correspondences are proposed between nearest neighbors in descriptor space and then refined by running Horn-Schunck optical flow between small image patches centered at the proposed matches and measuring image difference between the aligned patches. Mutual best matches between frames are accepted as long as the matched patches contain enough texture; each match is given a confidence $c(\boldsymbol{x}_i)$ based on how much better the best match is than the second best. An indicator function $\delta(\boldsymbol{x})$ is set to one at the locations where successful matches are found.

These sparse matches are integrated into an objective function of the form

$$E(\boldsymbol{d}) = E_{\text{color}}(\boldsymbol{d}) + \lambda E_{\text{gradient}}(\boldsymbol{d}) + \alpha E_{\text{smooth}}(\boldsymbol{d}) + \beta E_{\text{match}}(\boldsymbol{d}, \boldsymbol{d}_1) \;, \tag{2.8}$$

with

$$\begin{aligned}
E_{\text{color}}(\boldsymbol{d}) &= \int_\Omega \rho(|I_2(\boldsymbol{x} + \boldsymbol{d}(\boldsymbol{x})) - I_1(\boldsymbol{x})|^2) d\boldsymbol{x} \\
E_{\text{gradient}}(\boldsymbol{d}) &= \int_\Omega \rho(|\nabla I_2(\boldsymbol{x} + \boldsymbol{d}(\boldsymbol{x})) - \nabla I_1(\boldsymbol{x})|^2) d\boldsymbol{x} \\
E_{\text{smooth}}(\boldsymbol{d}) &= \int_\Omega \rho(|\nabla u(\boldsymbol{x})|^2 + |\nabla v(\boldsymbol{x})|^2) d\boldsymbol{x} \\
E_{\text{match}}(\boldsymbol{d}, \boldsymbol{d}_1) &= \int_\Omega \delta(\boldsymbol{x}) c(\boldsymbol{x}) \rho(|\boldsymbol{d}(\boldsymbol{x}) - \boldsymbol{d}_1(\boldsymbol{x})|^2) d\boldsymbol{x} \;.
\end{aligned} \tag{2.9}$$

The optical flow field is $\boldsymbol{d} = (u, v)$ (with u and v functions of image locations $\boldsymbol{x}$), $\rho(s^2) = \sqrt{s^2 + \epsilon^2}$ is a robust penalty function, and Ω is the image domain. The auxiliary flow "field" $\boldsymbol{d}_1$ is defined only at those points with $\delta(\boldsymbol{x}) > 0$ as $\boldsymbol{d}_1(\boldsymbol{x}_i) = \boldsymbol{x}_j - \boldsymbol{x}_i$, the displacement defined by the sparse match $(\boldsymbol{x}_i, \boldsymbol{x}_j)$.

Once sparse matches are computed, the energy function is optimized using the standard coarse-to-fine approach, solving the Euler-Lagrange equations at each level of the scale-space pyramid. This formulation differs slightly from the original Horn-Schunck formulation because the linear approximation of the image intensities in the data term has yet to be applied. The resulting Euler-Lagrange equations are non-linear; the solution is found using nested fixed-point iterations (Brox et al., 2004).

Although LDOF computes reasonable flow fields for realistic sequences, it is no longer among the highest performing published algorithms, at least on benchmark datasets. The standard benchmark is the Middlebury dataset (Baker et al., 2011). The benchmark contains eight sequences with publicly available ground truth flow for a pair of frames. Another eight sequences, with hidden ground truth flow, are used as a held-out test set to evaluate the performance of competing algorithms. The current ranking of published methods (and anonymous submissions under review) can be found at `http://vision.middlebury.edu/flow/`. By default, algorithms are ranked based on average endpoint error, defined as the average of the Euclidean distance between the estimated flow vector (u, v) and the ground truth flow vector (u_{GT}, v_{GT}) at each pixel. The benchmark has proved to be an extremely useful scientific tool, but it has its limitations. For example, the longest sequence has only eight frames, and ground truth is provided for only a single pair of frames per sequence. The sequences contain mostly rigid motion of relatively small magnitude. Occlusions, which cause problems for flow algorithms, are rare events. Finally, the performance of competing algorithms on the benchmark sequences is converging. Of the top 20 algorithms, only one has an average endpoint error greater than one pixel on any sequence.

The MPI-Sintel Flow dataset (Butler et al., 2012) is a challenging new benchmark for optical flow containing long sequences of up to 50 frames with large non-rigid motions complete with motion blur, significant occlusions, and numerous violations of the brightness constancy assumption caused by shadows and specular reflections. Although the sequences are synthetic, the creators of the dataset argue that both the image and flow statistics are representative of real sequences. Ground truth flow for *every* frame is provided for a training set and hidden for the held-out test set used to rank submitted algorithms. Because the sequences are computer generated, the true flow at a pixel can be computed by ray-tracing into the known scene geometry, propagating that point according to the known 3D motion, and then computing the projection of the point in the next frame. An interesting feature of this dataset is that this process also computes ground truth flow for pixels that are occluded in the following frame, and these pixels are included in the evaluation. The current ranking for algorithms tested on MPI-Sintel can be found at `http://sintel.is.tue.mpg.de/results`. The current results clearly demonstrate that, despite decades of progress, optical flow remains an unsolved problem. Algorithms achieving subpixel accuracy on the Middlebury benchmark report results with average endpoint errors between 7 and 14 pixels on the new sequences.

Sadly, neither dataset provides the ground truth we need to fully evaluate our results. The ground truth for MPI-Sintel comes closest, but the data at occlusions are insufficient. MPI-Sintel only follows points into the frame immediately following an occlusion; it does not follow the points until they reappear. In contrast, we return long-range correspondences across occlusions.

2.4 Occlusion Detection

Our definition of video motion includes an explicit occlusion detection step. Although detection of occluded pixels and the computation of optical flow are intimately connected, occlusion handling often seems to be added as an afterthought to motion estimation. In many optical flow algorithms – even advanced ones such as LDOF – occlusions are handled only indirectly through the use of robust metrics for the data and smoothness terms. Unsurprisingly, these methods can fail catastrophically when faced with significant occlusions. For instance, the average endpoint error of LDOF in unmatched regions of the MPI-Sintel dataset exceeds 42 pixels.

Algorithms that do explicitly detect occlusions base their estimates primarily on local properties of the computed flow field. One simple technique is to set a threshold on the maximum error in the brightness change constraint equation. Pixels with a large intensity change between frames are marked as occluded and pay a penalty defined by an occlusion penalty function instead of the penalty from the data term (*e.g.*, Xiao et al., 2006a; Ayvaci et al., 2012). For particular settings of the occlusion penalty function, the joint estimation of occlusion and optical flow is equivalent to using a properly chosen robust function for the data term without occlusions (Black and Rangarajan, 1996). By making the occlusion detection process explicit, it is possible to define more complicated diffusion processes for filling in correspondences in the occlusion region. Techniques often ignore occluded pixels while smoothing over a larger region using a weighted median (Sun et al., 2010b) or bilateral (Xiao et al., 2006a) filter.

Another local cue commonly used to detect occlusion is the local divergence of the flow field, $\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}$ (*e.g.*, Sand and Teller, 2008; Sun et al., 2010b). The idea is that negative divergence indicates the presence of multiple points in the first image mapping to the same location in the second image, which occurs during an occlusion. Other techniques count the number of pixels in the first frame that map to the same pixel in the second frame (Xu et al., 2012b) and associate occlusions to counts greater than 1. Finally, some have proposed computing bidirectional flow fields, from frame 1 to frame 2 *and* from frame 2 back to frame 1, and declaring locations where the two flows disagree to be occlusions (*e.g.*, Alvarez et al., 2007; Ince and

Konrad, 2008). In all cases, the optical flow within the occluded region is filled in by relying on the spatial smoothness of the flow field. These hallucinated flow vectors are not necessarily any more accurate than those generated by methods that do not explicitly handle occlusions. For example, Xu et al. (2012b) average almost one pixel less endpoint error than Brox and Malik (2011) in the matched (unoccluded) regions of the MPI-Sintel dataset but perform worse in the unmatched (occluded) regions.

Our method uses these traditional cues (specifically, violations of brightness constancy and multiple points mapping to a single pixel) but in a multiframe setting. The local techniques used in traditional optical flow are hampered by treating each consecutive pair of images as an independent problem. We can more accurately predict the location of a point during an occlusion because the extrapolation is based on the pattern of motion observed over many frames. Optical flow methods can only rely on the smoothness term to predict the location of an occluded point. Finally, accumulating evidence from multiple frames makes it easier to distinguish between violations of brightness constancy due to image noise and violations resulting from occlusions.

It is possible in some cases to detect occlusions without recovering the exact motion in the scene. Feldman and Weinshall (2008) propose detecting occlusion (or disocclusion) boundaries by thresholding the smallest eigenvalue of the 3×3 structure tensor matrix $\sum_{\Omega} \nabla I (\nabla I)^T$. Zero should be an eigenvalue of this matrix if the motion is constant over the window and there are no occlusions. The method can miss occlusions in regions with insufficient texture. Apostoloff and Fitzgibbon (2005) find occlusions without explicitly estimating motion by training a classifier to detect T-junctions in space-time patches that collect the appearance of a single row of an image over time. Like our method, this approach can benefit from incorporating evidence from multiple frames. However, it assumes that motion is locally linear and primarily horizontal, while our approach works for all types of motion.

2.5 Layered Models

Occlusions result in high values of the smoothness term as well as the data term. Layered optical flow methods define the motion field as a composition of K layers, with a smoothness penalty between neighboring pixels in different layers that does not depend on the difference in the flow field across the boundary. Motion within a single layer may be parametric (*e.g.*, Jepson et al., 2002; Jojic and Frey, 2001; Kumar et al., 2008; Wang and Adelson, 1994), or non-parametric (*e.g.*, Weiss, 1997; Sun et al., 2010a, 2012, 2013) with a regularization term penalizing spatial variation, possibly with respect to an underlying parametric representation. By collecting evidence over multiple frames (Darrell and Fleet, 1995; Sun et al., 2012), it is possible to reason about the depth ordering of the layers and to detect occlusions when layer assignment changes between frames that are aligned according to the detected optical flow. The number of layers used may vary during optimization or can be fixed at some low value; two or three layers are popular choices, although methods that require that motion fit a parametric form typically require more layers to capture the complexity of real-world motions.

Layered techniques can be used to follow points through occlusions because the flow field for each layer is defined over the entire frame, even at points where the layer is occluded from view. However, the resulting correspondences are only correct when the assumption that the motion is spatially parametric holds. Layered techniques ignore evidence from surrounding frames when computing the flow field for a given layer at a particular frame; temporal smoothness constraints are included for the layer assignment but not for the flow fields. Our approach uses evidence from all frames to predict the motion of occluded points, and our parameterization of motion is temporal rather than spatial.

Layered techniques end up segmenting the video while simultaneously solving for the motion in the sequence. Ristivojevic and Konrad (2006) also propose to detect occlusions by segmenting video, although they do not place themselves within the layered model framework. They assume a foreground object moves according to a parametric motion model and occludes portions of the background, which moves according to a different parametric motion model. The task is to extract 3D space-time surfaces corresponding to (1) the evolving boundary of the foreground object (the "object tunnel"), (2) the boundary of the background region that changes from visible to occluded (the "background occlusion volume"), and (3) the boundary of the background region that changes from occluded to visible during the sequence (the "background exposed volume"). If there are multiple moving foreground objects that occlude each other, the object tunnel may be split into object occlusion volumes and object exposed volumes. A point is likely to be assigned to a an occluded volume if it exhibits small variation in intensity up to some frame t_j but large variations in frames after t_j. A point is likely to be assigned to an exposed volume if the reverse is true – *i.e.*, the intensity variance is large up to some frame t_k and small for the remainder of the sequence. The method encourages compact volumes by penalizing the total surface area of each boundary surface. The reliance on parametric motion fields is limiting, and it is unclear how well the approach would generalize to real-world sequences, as the published results are primarily toy synthetic examples.

2.6 Multiframe Constraints and Structure-from-Motion

When optical flow fields between frames are coupled, it is typically done by assuming smooth motion and adding a penalty that increases with the magnitude of the partial derivative of the flow with respect to time to the regularization term that already penalizes large spatial derivatives (Weickert and Schnörr, 2001). More advanced constraints on the motion of points collected over many frames trace back to the work in structure-from-motion (SFM).

Tomasi and Kanade (1992) introduced the factorization method for solving SFM. A static scene is captured by a moving camera; the goal is to recover both the unknown locations of the camera in each frame and the 3D structure of the scene. To begin, feature points are tracked using a frame-to-frame tracker (see Section 2.2), and the locations (u_{fp}, v_{fp}) of all P points in all F frames are collected into a *measurement matrix* $\boldsymbol{W}$, a $2F \times P$ matrix where each column corresponds to the track of a single point. The first F rows contain the horizontal location of each point in a given frame; the last F rows contain the vertical location of each point. The *registered measurement matrix* $\widetilde{\boldsymbol{W}}$ is constructed by subtracting the mean from each row of $\boldsymbol{W}$.

Under orthographic projection (and in the absence of noise) the registered measurement matrix can be factored as $\widetilde{\boldsymbol{W}} = \boldsymbol{RS}$, where $\boldsymbol{R}$ is a $2F \times 3$ matrix describing the orientation of the camera over time and $\boldsymbol{S}$ is a $3 \times P$ matrix defining the scene structure. Although the observed registered measurement matrix is corrupted by noise, the motion of the camera and the structure of the scene can be recovered by solving for the best rank three approximation of $\widetilde{\boldsymbol{W}}$. Assuming all entries of the matrix are observed and "best approximation" is measured in terms of the Frobenius norm, the solution can be found through the singular value decomposition (SVD).

Static-scene SFM is now typically computed through bundle adjustment (Triggs et al., 1999), a non-linear optimization that recovers intrinsic and extrinsic camera parameters as well as the 3D structure of the scene by minimizing the total error between the predicted image location of a 3D point under perspective projection and the observed location. However, there has been a significant amount of work over the last 20 years to extend the factorization method to apply to scenes with multiple rigid objects (Costeira and Kanade, 1998; Vidal et al., 2008; Rao et al., 2010), varying camera models (*e.g.*, Poelman and Kanade, 1997; Christy and Horaud, 1996;

Sturm and Triggs, 1996), different noise models (Ke and Kanade, 2005), and even articulated (Yan and Pollefeys, 2008) or non-rigid objects (*e.g.*, Brand, 2005; Bregler et al., 2000; Xiao et al., 2006b; Akhter et al., 2011). In the non-rigid case, the standard assumption is that deformation can be expressed as a time-varying linear combination of a set of basis shapes or, equivalently, that each point follows a trajectory in 3D that can be expressed as a linear combination of a set of basis trajectories. Sometimes, priors are included to encourage smoothness, both in the recovered shape and the deformations. For example, Torresani et al. (2008) propose a hierarchical generative probabilistic model of shape and motion and infer the parameters using a expectation-maximization approach. We use the general factorization method as a basis for our work on extracting a sequence-specific path basis.

A big challenge for factorization methods is missing data in the measurement matrix resulting from lost tracks in the initial KLT tracker. Some previous approaches can handle missing data, but they may require that at least some tracks be complete (Rao et al., 2010) or that missing data not be too prevalent. For example, performance on the motion segmentation task in Vidal et al. (2008) rapidly degrades when over 20% of the entries in the measurement matrix are missing. By recognizing that columns of the matrix can be merged to reduce missing data, we perform successful factorization with significant (up to 85%) missing data, without requiring any complete tracks. We discuss the problem of missing data in much more detail in Chapter 4.

Any SFM technique can be used to determine image motion by taking the recovered structure and motion and computing the projected image locations according to the camera model. In this view, SFM methods simply denoise pre-computed image motion of a sparse set of trackable points by projection into a low-dimensional subspace. Our approach to video motion avoids the difficult reconstruction step and instead searches for multiframe correspondences within a defined linear subspace from the start. We are not restricted to points that can be tracked frame to frame but can recover video motion paths for all points in the scene.

2.7 Long-Range Motion Trajectories

Recent efforts in video segmentation and analysis have demonstrated the importance of recovering motion estimates that can provide information about multiframe interactions while maintaining the density available from optical flow methods. Video segmentation (*e.g.*, Brox and Malik, 2010; Fragkiadaki et al., 2012; Lezama et al., 2011) can use information about the multiframe motion of points to group objects based on the principle of common fate (Koffka, 1935). Small differences in motion in any single frame accumulate over time to form noticeable differences after many frames, so collecting information over multiple frames often simplifies the clustering problem. Multiframe motion trajectories are also popular features for gait analysis (Perbet et al., 2009) and action recognition (Wang et al., 2011; Matikainen et al., 2009; Prest et al., 2013).

Our technique provides the desired dense multiframe correspondences. The approaches we review in this section come the closest to computing the desired information but do not properly account for occlusions.

The simplest technique for constructing multiframe motion trajectories that are more spatially dense than those provided by a KLT tracker is to concatenate flow vectors computed using a two-frame optical flow method. The point starting at (x_t, y_t) is tracked to the next frame using the flow field in the current frame (u_t, v_t) as

$$(x_{t+1}, y_{t+1}) = (x_t, y_t) + (u_t(x_t, y_t), v_t(x_t, y_t)) \ . \tag{2.10}$$

Both Sundaram et al. (2010) and Wang et al. (2011) propose constructing trajectories based on concatenated flow vectors. The two methods differ in how they handle occlusions and how they interpolate to find the optical flow vector at real-valued positions (x_t, y_t). Wang et al. (2011) do not detect occlusions, but simply terminate all trajectories after 15 frames. They compute the flow vector at (x_t, y_t) using a median filter with a 3×3 neighborhood on the flow field computed by the OpenCV (Bradski, 2000) implementation of the technique described by Farnebäck (2003).

The more sophisticated and more popular method of Sundaram et al. (2010) uses bilinear interpolation to compute the optical flow vector at (x_t, y_t) from LDOF flow fields. Trajectories extend until an occlusion is detected based on two criteria. First, occlusions are detected by looking for regions where the forward flow (from t to $t+1$) and the backward flow (from $t+1$ to t) disagree. Second, trajectories terminate at all motion boundaries. While these terminations do not necessarily correspond to occlusions, they are essential for keeping trajectories from drifting across a boundary from a foreground to a background object or vice versa.

While both techniques can create a trajectory for every pixel in a video sequence, in practice they only initialize trajectories in textured regions. When pixels in later frames have enough surrounding texture but do not have any trajectories within a small neighborhood, new trajectories are initialized and propagate forward in time.

These methods introduce little computational overhead beyond that required to compute the initial optical flow fields. However, they assume that the optical flow fields are accurate, even though each flow field is computed based only on local information. The local occlusion detection can be similarly inaccurate. Neither method is capable of associating points across occlusions because the flow fields they use do not estimate motion of occluded points.

The Particle Video method from Sand and Teller (2008) also starts with concatenated optical flow vectors but refines these *post facto* by optimizing a cost function that includes multiframe data and smoothness terms. The data term penalizes trajectories based on the difference between the appearance in a single frame and the running average of the appearance over a few frames, computed by smoothing with a Gaussian kernel with a standard deviation of five frames. The smoothness term penalizes differences in motion between neighboring trajectories. Neighbors are defined by computing a Delaunay triangulation in frames $t-1$, t, and $t+1$ and are weighted according to the average difference in motion between the two trajectories over all frames in the sequence. High-cost trajectories are terminated at suspected occlusions. As with the other methods, new trajectories are initialized to fill holes. Compared to Sundaram et al. (2010), this method is significantly more expensive. Sadly, it does not result in more accurate trajectories, possibly because it relies on slightly older, less advanced optical flow techniques to create the initial trajectories. Like the two previous methods, it cannot associate the trajectories of reappearing points across occlusions.

The final methods we review are the most closely related to the concept of video motion we present in this thesis because they use the rank constraints from SFM to return sequence-length correspondences akin to our definition of paths. However, they do not account for occlusions.

Rank constraints for dense motion were first introduced by Irani (2002), but her method only applied to rigid scenes. Torresani and Bregler (2002) track a sparse set of reliable points through all frames using a KLT tracker and find a basis for these tracks through PCA. Non-rigid motions are allowed as long as a suitably low rank basis can be found to approximate them. The correspondences for untracked points are found by estimating coefficients for each basis track using factored sampling (Grenander et al., 1991). The likelihood distribution that is used models changes in image intensity along a trajectory as independent and identically distributed Gaussian random variables. The inferred coefficients correspond to the expected value of the posterior distribution for each trajectory. There is no additional regularization beyond the low rank assumption. This method makes two assumptions that are often violated: that a KLT tracker will return enough full-length tracks of textured points, and that the posterior distribution is unimodal. Our method makes no such assumptions.

Garg et al. (2010) solve for motion using a variational framework similar to standard optical flow estimation but using a low-rank basis found as in Torresani and Bregler (2002). The unknowns in the variational objective function are reconstruction coefficients for trajectories located on a uniform grid in a reference frame. The objective function combines a data term with a quadratic penalty on changes in intensity along a trajectory with a variational smoothness term with a quadratic penalty on the spatial gradient of the coefficient fields. The solution is found using a direct extension of the nested fixed point iterations from variational optical flow. Because occlusions are ignored, trajectories are unfairly penalized for changes in appearance when they are not visible. (The method was developed to solve for the multiframe registration of deforming surfaces in domains where occlusions are rare.) In an extension, Garg et al. (2011) soften the hard subspace constraints in their earlier approach to create a prior on image motion. The new method includes robust penalty functions to provide some robustness to occlusions, but there is no explicit detection of occlusion events. Garg et al. also experiment with using the DCT basis for trajectories in place of the sequence-specific PCA basis.

In Chapter 6, we compare our video motion paths to the results from Sundaram et al. (2010) and Garg et al. (2011). We use results from Sundaram et al. (2010) to initialize our solution. We improve on their formulation by considering evidence across all frames during optimization and maintaining correspondence across occlusions. We improve on the results from Garg et al. (2011) by explicitly modeling occlusions and reporting motion for all points, not just those visible in a privileged reference frame.

3

Video Motion

The key contribution of this thesis is the definition of and a solution for *video motion*. We begin with a formal definition. Consider a single world point, identified by the index p, appearing in a video containing F frames. The image projection of the point moves along a *path*, denoted $\boldsymbol{x}_p : \{1, \ldots, F\} \rightarrow \mathbb{R}^2$, where $\boldsymbol{x}_p(t)$ is the location at which the point would appear in frame t if it were visible. We mark visible frames using the visibility flag $\nu_p : \{1, \ldots, F\} \rightarrow \{0, 1\}$. The value $\nu_p(t)$ is one if the point is visible in frame t and zero otherwise. Both $\boldsymbol{x}_p(t)$ and $\nu_p(t)$ are unknowns to be estimated for all paths in a given video sequence. Video motion consists of the paths and visibility flags for a dense sampling of each visible surface in a scene. In the next sections we develop a model that allows for video motion to be extracted automatically from a few seconds of video.

3.1 Selecting Scene Points for Paths

Points sampled on visible surfaces form a set of *anchor points*

$$\{(\boldsymbol{u}_1, \tau_1), \ldots, (\boldsymbol{u}_p, \tau_p), \ldots, (\boldsymbol{u}_P, \tau_P)\} , \tag{3.1}$$

where $\boldsymbol{u} = (u, v)$ is a position in the image plane at frame τ. We create one path per anchor point and constrain the path to pass through its anchor point while visible, *i.e.*, $\nu_p(\tau_p) = 1$ and $\boldsymbol{x}_p(\tau_p) = \boldsymbol{u}_p$.

A set of anchor points is sufficient to explain the motion in the video if it results in paths that can explain every pixel in the video. For each pixel (u, v, t), we want some path p such that $\nu_p(t) = 1$ and $||(u, v) - \boldsymbol{x}_p(t)|| < \Delta$ for some small distance Δ that defines a spatial neighborhood. An efficient representation of motion would divide world surfaces into small patches and use a single anchor point for each unique visible patch. This would guarantee full coverage provided that all the paths and visibility flags can be found. However, it is difficult to identify unique world points accurately without first knowing the motion that we are trying to find. Placing an anchor point at every pixel in every frame of the sequence would guarantee that every pixel is explained but would result in multiple paths for a single world point. For example, a point visible in T frames would be associated with T distinct paths when only one would suffice.

An alternative to selecting every pixel as an anchor point is to first select a few privileged reference frames and to locate anchor points at every pixel in each of these selected frames (*e.g.*, Garg et al., 2010, 2011; Ricco and Tomasi, 2012a). Unfortunately, there is no guarantee that a deterministic selection of a few frames will result in a set of paths that explain all pixels in the sequence because objects cannot be guaranteed to be visible in any subset of frames. Our definition allows anchor points to be located in any frame of the video sequence and there is no requirement that nearby world points have paths anchored in the same frame.

3.2 Path Parameterization

A path is defined relative to its anchor point as a linear combination of a finite set of basis paths, $\{\boldsymbol{\varphi}_1, \ldots, \boldsymbol{\varphi}_K\}$, up to a shift. Specifically, we have

$$\boldsymbol{x}_p(t) = \boldsymbol{u}_p + \sum_{k=1}^{K} c_{pk} \left(\boldsymbol{\varphi}_k(t) - \boldsymbol{\varphi}_k(\tau_p) \right) \ . \tag{3.2}$$

It is easy to see that any setting of $\boldsymbol{c}_p = (c_{p1}, \ldots, c_{pK})$ satisfies $\boldsymbol{x}_p(\tau_p) = \boldsymbol{u}_p$, so that a path cannot become dislodged from the world point it follows. It would be equivalent to use a linear model in standard form

$$\boldsymbol{x}_p(t) = \sum_{k=1}^{K+2} c_{pk} \boldsymbol{\varphi}_k(t) \ , \tag{3.3}$$

with the last two basis functions set to the constant functions $\boldsymbol{\varphi}_{K+1}(t) = (1, 0)$ and $\boldsymbol{\varphi}_{K+2}(t) = (0, 1)$ and the coefficients $c_{p,K+1}, c_{p,K+2}$ defined as functions of $\boldsymbol{u}_p, \boldsymbol{\varphi}_k(\tau_p)$, and c_{pk} for all other k, computed so as to satisfy the constraint that $\boldsymbol{x}_p(\tau_p) = \boldsymbol{u}_p$. The formulation in (3.2) is used to eliminate these constrained variables, keeping only the free variables. Note that we use the same set of basis paths to express the paths of all visible points in a given sequence.

Representing all paths parametrically using a single finite set of basis functions does not, in principle, restrict the variety or complexity of motions that can be represented. Rather than representing paths as functions, we can write them as vectors of length $2F$. The ith entry is the x-coordinate in frame i and the $(i + F)$th entry is the y-coordinate in frame i. From this representation we can see that a basis for $\mathbb{R}^{2F}$ would be able to describe any possible path lasting F frames.

One particular setting of $\boldsymbol{\varphi}_k$ connects paths to optical flow. Consider the basis where $\boldsymbol{\varphi}_k(t) = (\mathbb{1}_{t=k}, 0)$ and $\boldsymbol{\varphi}_{k+F}(t) = (0, \mathbb{1}_{t=k})$. The symbol $\mathbb{1}_{t=k}$ denotes the indicator function taking the value 1 when $t = k$. Represented as vectors, these functions form the standard basis for $\mathbb{R}^{2F}$. The values $(c_{pk}, c_{p(k+F)})$ with this basis correspond to the flow at $\boldsymbol{u}_p$ in the optical flow field computed between frame τ_p and frame k.

A better choice for a basis could be the DCT basis, which has been a popular choice in recent related work (Garg et al., 2011; Akhter et al., 2011). These basis functions consist of cosine functions of increasing frequency. Each function contains support in every frame, which makes the process of estimating paths (by estimating coefficients c_{pk}, given evidence from the video) more robust to noise in a single frame. Additionally, if it is known that motions are sufficiently smooth, only the low-order basis functions need to be used. This approach decreases the dimension of the search space when estimating paths for long sequences and guarantees that the expected smoothness properties hold.

Rather than use these pre-defined bases, we can select a sequence-specific basis. For a particular sequence, it is often possible to find a much more compact basis that can still express all possible paths with reasonable accuracy than would be possible using a fixed basis for all sequences. A sequence-specific basis can provide dimensionality reduction that is equivalent to what can be achieved by using only the low-order DCT basis functions without requiring that the motion be temporally smooth. Instead, the basis provides strong temporal regularization that forces extracted paths to be consistent with all other paths in the sequence.

The theoretical justification for the existence of these compact, sequence-specific bases traces back to the factorization method for SFM (Tomasi and Kanade, 1992) we reviewed in Chapter 2. We postpone further discussion of how to find a sequence-specific basis for an input sequence to Chapter 4 and assume for the moment that the basis is given.

3.3 Optimal Paths

Let us assume that, in addition to having specified a basis, we have already selected a set of anchor points and have estimated the visibility of each point in each frame. The path for each point is determined by specifying the coefficients of the linear combination $\boldsymbol{c}_p$. How do we determine the likelihood that this guess represents the true motion of the scene?

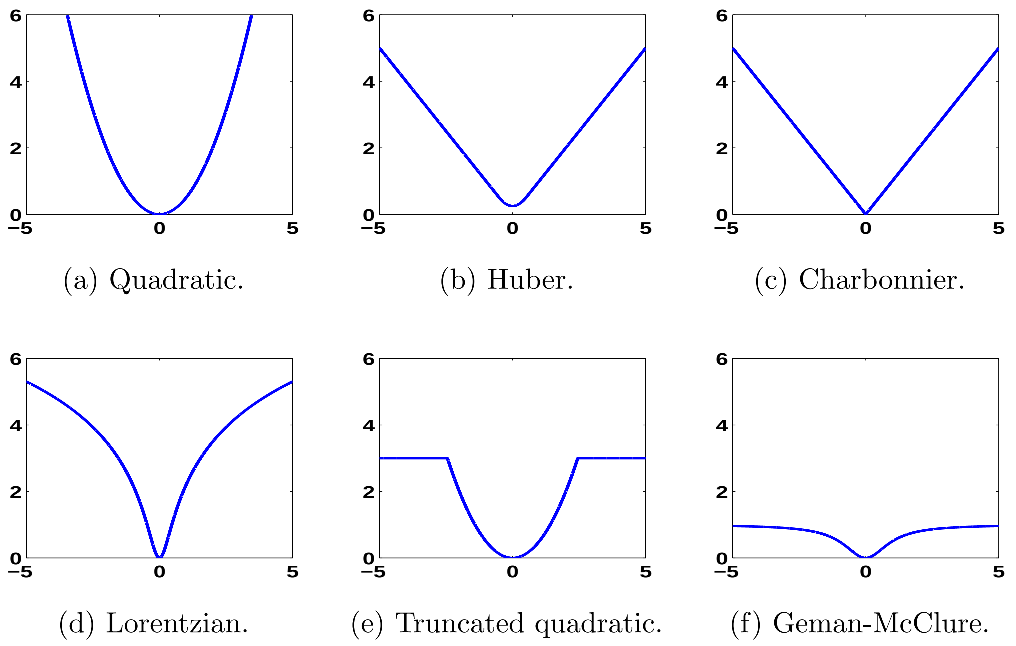
First, we assume that the appearance of a point does not change over time. This means that, if we sample the appearance of the video at points along a path in frames where the point is visible, we expect constant measurements in the absence of noise. The reference appearance of each point is given by the appearance of video at the anchor point. In the context of this thesis, we work with grayscale video and define the appearance of a point $(\boldsymbol{u}_p, \tau_p)$ as the intensity at that location, $I(\boldsymbol{u}_p, \tau_p)$. We use interpolation to recover intensity values at real-valued locations $\boldsymbol{u}_p$.

The value $\Delta I_p(t) = I(\boldsymbol{x}_p(t), t) - I(\boldsymbol{u}_p, \tau_p)$ measures the difference in intensity sampled along the path in frame t and the reference intensity at the anchor point in frame τ_p. The total penalty for changes in intensity along a path is given by

$$E_D(\boldsymbol{c}_p) = \sum_{t=1}^{F} \nu_p(t)\rho(\Delta I_p(t)) \ . \tag{3.4}$$

This quantity is a function of the path coefficients $\boldsymbol{c}_p$, which determine the shape of the path $\boldsymbol{x}_p(t)$. The function ρ is a symmetric penalty function, a non-decreasing function of the absolute value of its argument, and multiplying by $\nu_p(t)$ ensures that the penalty is levied only on visible points.

We use $\rho(s) = \sqrt{s^2 + \epsilon^2}$ (with $\epsilon^2 = 0.001$ as is standard), which is a differentiable approximation of the function $\rho(s) = |s|$. When compared to the standard quadratic penalty $\rho(s) = s^2$, our penalty function is less sensitive to the large changes in appearance that can occur (even along the correct path) as a result of unmodeled



(a) Quadratic. (b) Huber. (c) Charbonnier.

(d) Lorentzian. (e) Truncated quadratic. (f) Geman-McClure.

FIGURE 3.1: Examples of common penalty functions (see Black and Rangarajan (1996) for more details). The quadratic penalty function, $\rho(s) = s^2$, is sensitive to outliers. The Huber function combines a quadratic penalty for small errors with a linear penalty for large errors. It is defined as $\rho(s) = \frac{s^2}{2\epsilon} + \frac{\epsilon}{2}$ for $|s| < \epsilon$ and $\rho(s) = |s|$ for $|s| \geqslant \epsilon$. The Charbonnier function, $\rho(s) = \sqrt{s^2 + \epsilon^2}$, is a differentiable approximation of the absolute value. The non-convex Lorentzian, $\rho(s) = \log\left(1 + \frac{1}{2}\left(\frac{s}{\sigma}\right)^2\right)$, is differentiable and further reduces the effect of outliers. The truncated quadratic, $\rho(s) = \min(s^2, \beta)$, assesses a constant penalty for outliers but is not differentiable. The Geman-McClure function, $\rho(s) = \frac{s^2}{1+s^2}$, is differentiable and robust to outliers but is not convex. We use (c).

lighting changes or incorrectly detected occlusions. The quadratic penalty would be optimal if the changes in intensity due to noise were normally distributed. Many other choices for ρ exist; see Figure 3.1 for a sample of popular functions. Some of these functions are non-convex, and some, *e.g.*, the Geman-McClure function, have $\lim_{s\to\infty} \rho'(s) = 0$ so that the penalty assessed for gross outliers does not increase without bound. Using such functions improves robustness at the cost of (possibly additional) non-convexity, which complicates minimization.

We sum together the penalties for intensity changes along all paths to get the first term in our path objective function. This data term cannot distinguish between paths that move along isocontours of intensity within the video sequence. This limitation is the multiframe extension of the aperture problem from optical flow estimation. We address this problem by adding spatial regularization based on our assumption that nearby points should follow essentially parallel paths. Our final objective function to be minimized is a weighted combination of the data term and a spatial smoothness term penalizing differences in the path coefficients between pairs of paths. The full objective function is

$$E(\boldsymbol{c}_1, \ldots, \boldsymbol{c}_P) = \sum_{p \in \mathcal{P}} E_D(\boldsymbol{c}_p) + \frac{\lambda}{2} \sum_{p \in \mathcal{P}} \sum_{q \in \mathcal{P}} E_S(\boldsymbol{c}_p, \boldsymbol{c}_q) \ , \tag{3.5}$$

with

$$E_S(\boldsymbol{c}_p, \boldsymbol{c}_q) = \alpha_{pq} \sum_{k=1}^{K} \rho(c_{pk} - c_{qk}) \ . \tag{3.6}$$

We use the same robust function here as in the data term, although this is not a requirement. The multiplier α_{pq} couples nearby paths that have similar reference appearance and is equal to

$$\alpha_{pq} = \exp\left(-\frac{(I(\boldsymbol{u}_p, \tau_p) - I(\boldsymbol{u}_q, \tau_q))^2}{\sigma^2}\right) \tag{3.7}$$

if $p \neq q$ and the path p is both visible in the anchor frame of path q (that is, $\nu_p(\tau_q) = 1$) and close enough to the anchor of q (that is, $||\boldsymbol{x}_p(\tau_q) - \boldsymbol{u}_q|| < \Delta$). Otherwise, $\alpha_{pq} = 0$. Both the use of a robust penalty function and the weight α_{pq} allow discontinuities at object boundaries (as object boundaries tend to coincide with changes in intensity that decrease the strength of α_{pq} across the boundary).

The neighbor relationships defined by α_{pq} are not known *a priori* and are instead a function of the current estimate of the paths. The only relationships that can be determined independent of the path coefficients are relationships between paths with anchor points in neighboring pixels in a single frame. These are the relationships exploited by variational formulations of the smoothness term in methods with reference frames defined *a priori* (Garg et al., 2010; Ricco and Tomasi, 2012a). Because our placement of anchor points is unrestricted, there is no guarantee that there will be sufficient neighbors to constrain the solution based on anchor points alone.

The parameter λ allows for adjustment of the relative strength of the data and smoothness term. Note that the relative weight between the two terms also depends on the magnitude of the coefficients $\boldsymbol{c}_p$. The choice of basis paths will affect the scaling of $\boldsymbol{c}_p$, as one could multiply a basis path by some non-zero scalar a and multiply the corresponding coefficient of each path by the reciprocal $\frac{1}{a}$ without changing the final paths. To make the range of $\boldsymbol{c}_p$ consistent across sequences, we choose to scale each basis path independently so that the average motion between consecutive frames is one pixel. Other choices, such as making each basis path have unit norm in vector form, are possible.

Although our standard penalty functions are convex functions of their arguments, our full objective function is non-convex in the unknowns $\boldsymbol{c}_p$ because $\Delta I_p(t)$ is not guaranteed to be convex. We describe our approach to constructing an initial solution and searching for a local minimum in Chapter 5.

3.4 Optimal Visibility

The visibility flags $\nu_p(t)$ for each path and frame were treated as known in the data term of the path objective function. However, these values are unknown in general; they must also be estimated from data.

We model visibility flags as an MRF whose structure depends on the current estimates $\boldsymbol{x}_p(t)$ of the paths $p \in \mathcal{P}$. Formally, an MRF is a graphical model $G = (V, E)$ that defines a probability distribution over a set of random variables taking values from a specified label set. Each random variable is associated with a node in the graph, and edges in the graph represent dependencies. The graph structure is defined so that the random variable at node v_i is conditionally independent of all other nodes, given its immediate neighbors $\{v_j|(v_i, v_j) \in E\}$. Consequently, the joint distribution can be written as a (sufficiently normalized) product of potential functions defined over cliques in the graph. For general MRFs, most inference tasks are NP-hard (Veksler, 1999; Kolmogorov and Zabih, 2004).

In computer vision, the most common MRFs have maximum cliques of size two and are often defined in terms of their energy function, written as

$$E(\boldsymbol{x}) = \sum_{p \in V} D(x_p) + \sum_{p,q \in E} V_{pq}(x_p, x_q)\ . \tag{3.8}$$

The value of $\boldsymbol{x}$ that minimizes E is the *maximum a posteriori* (MAP) estimate given some observations $\boldsymbol{o}$ when

$$P(\boldsymbol{o}|\boldsymbol{x}) \propto \prod e^{-D(x_p)} \tag{3.9}$$

is the likelihood and

$$P(\boldsymbol{x}) \propto \prod_{p,q \in E} e^{-V_{pq}(x_p, x_q)} \tag{3.10}$$

is the prior distribution. The functions $D(x_p)$ are referred to as data terms, and the prior terms $V_{pq}(x_p, x_q)$ are usually called smoothness terms because "smoothness" is a very common prior in vision problems.

Our MRF has one node for each point $\boldsymbol{v}_p(t) = (\boldsymbol{x}_p(t), t)$ along some path, for $t = 1, \ldots, F$, and one binary random variable $\nu_p(t)$ per node. The neighborhood of $\boldsymbol{v}_p(t)$ consists of the set of points $\boldsymbol{v}_q(t)$ with $q \neq p$ and $||\boldsymbol{v}_p(t) - \boldsymbol{v}_q(t)|| \leqslant \Delta$ for some small fixed Δ plus the two points $\boldsymbol{v}_p(t-1)$ and $\boldsymbol{v}_p(t+1)$. The latter two points are the nodes temporally adjacent to $\boldsymbol{v}_p(t)$ along the same path p and form the temporal neighborhood. The other nodes in the neighborhood are spatial neighbors.

Each node in the MRF is associated with a binary *observed visibility* flag $\hat{\nu}_p(t)$. This value is computed from the video using a less sophisticated model of spatial and temporal smoothness than the MRF provides. The purpose of $\hat{\nu}_p(t)$ is to provide an initial estimate of visibility to ground the estimate from the MRF in the evidence from the video based on our assumptions that visible points do not change their appearance over time and that a single object is visible at each video pixel.

Consider the video pixel with coordinates (x, y, t). We denote the set of paths intersecting this video pixel as $Q(x, y, t) = \{q \mid ||\boldsymbol{x}_q(t) - (x, y)|| < \Delta\}$. We may have $|Q| > 1$ for two reasons. First, there may be an occlusion, which occurs when distant points in 3D project to the same location in the image. In this case, we expect the intersecting paths to have different motions in the surrounding frames. One of these motions corresponds to the movement of the visible surface; the others correspond to estimated motions of the occluded surfaces. If we take a small image patch and transport it along the different paths within a small temporal window, we expect minimal intensity changes in the patches that follow the motion of the visible surface. Because the appearance of the video and the motion of the occluded object are uncorrelated, we expect larger changes in brightness when following paths of occluded surfaces.

We measure the patch brightness constancy for each path p in each frame t as follows. Let

$$C(p, t) = \Upsilon(p, t, t) + \Upsilon(p, t, \tau_p) \ , \tag{3.11}$$

with

$$\Upsilon(p, t, t') = \iiint_{-\infty}^{\infty} w(\boldsymbol{\xi}, \tau - t) |I(\boldsymbol{x}_p(\tau) + \boldsymbol{\xi}, \tau) - I(\boldsymbol{x}_p(t') + \boldsymbol{\xi}, t')| d\boldsymbol{\xi} d\tau \tag{3.12}$$

and

$$w(\boldsymbol{\xi}, \tau) \propto e^{\frac{-(||\boldsymbol{\xi}||^2 + \tau^2)}{\sigma^2}} \ . \tag{3.13}$$

The value $\Upsilon(p, t, t)$ measures the Gaussian-weighted absolute image differences between the image patch centered at $\boldsymbol{x}_p(t)$ in frame t and the volume of values obtained by transporting the patch through the motion $\boldsymbol{x}_p(t)$. The parameter τ indexes frames, while $\boldsymbol{\xi}$ is a spatial offset. The value $\Upsilon(p, t, \tau_p)$ compares the transported volume to the reference patch for point p, located in the frame of its anchor point, τ_p. The surface may have deformed some between τ_p and t, so we often have $\Upsilon(p, t, \tau_p) > \Upsilon(p, t, t)$. Including both terms allows for robustness to some deformation while still penalizing too much drift from the reference patch.

Thus, $C(p, t)$ measures the severity of violations of our brightness constancy assumption for path p in the vicinity of frame t. As stated, if the point p is visible at time t, we expect $C(p, t)$ to be low and high otherwise. Crucially, if p is occluded, there must be some *other* point $q \in Q$ that is visible at time t and hides p so that $C(q, t) < C(p, t)$. Let

$$p^*(x, y, t) = \arg \min_{q \in Q(x,y,t)} C(q, t) \tag{3.14}$$

be the index of the path with the best patch brightness constancy in the neighborhood of video pixel (x, y, t). We call this path a *controlling path* and set $\hat{\nu}_{p^*}(t) = 1$.

A video pixel may have multiple paths intersecting it without an occlusion occurring if the same scene point is associated with multiple anchor points (and therefore multiple paths). This frequently occurs when temporarily occluded points reappear in a scene. To handle this case, we measure the average distance between paths $p, q \in Q$ as

$$\bar{d}_{pq} = \frac{1}{F} \sum_{t=1}^{F} ||\boldsymbol{x}_p(t) - \boldsymbol{x}_q(t)|| \; . \tag{3.15}$$

Paths that are essentially parallel to the controlling path are also likely to be visible in this frame. We set $\hat{\nu}_p(t) = 1$ if $\bar{d}_{pp^*} < d$ for some small threshold d, say $d = 4$ pixels. Here, p^* is the controlling path for the video pixel intersected by path p at time t. (Note that this definition sets $\hat{\nu}_{p^*}(t) = 1$ automatically because $\bar{d}_{p^*p^*} = 0$.)

Figure 3.2 illustrates these concepts with a concrete example. The paths drawn all

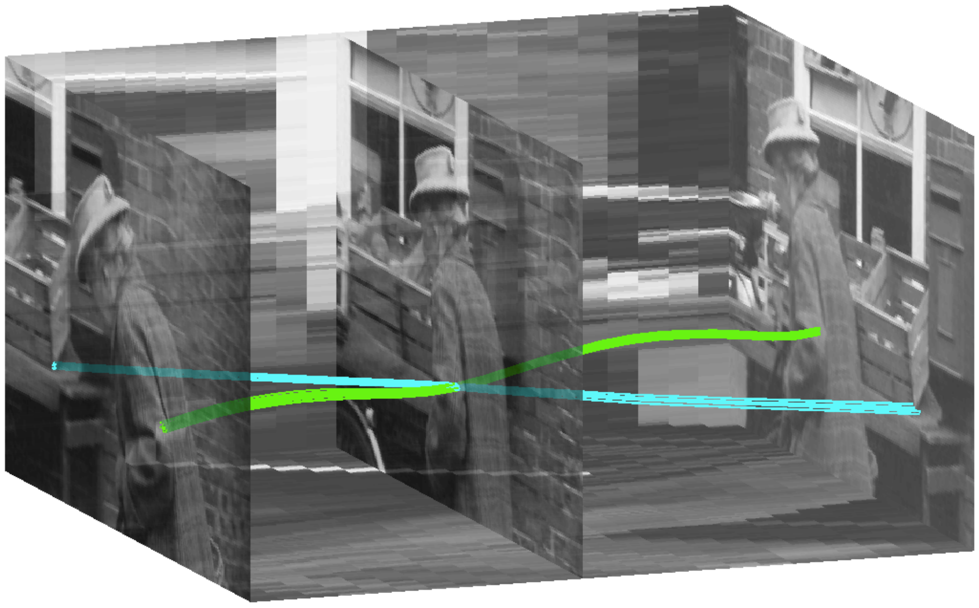
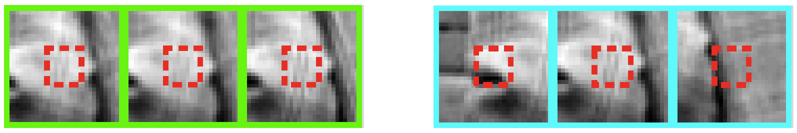



FIGURE 3.2: The space-time cube for our example sequence with vertical slices showing frames 1, 12, and 25. Time increases to the right. The corner of the crate (cyan) is occluded by Miss Marple's arm (green) in frame 12. A small patch (red dashed squares) is transported along each path and monitored for consistent appearance. The arm patch (bottom left, outlined in green) is most consistent, making this a controlling path. Points along paths that either coincide with or are substantially parallel to a nearby controlling path have their observed visibility flag $\hat{\nu}_p(t)$ set to 1. The observed visibility flags for the cyan paths are all 0 in this frame. Observed flags feed into the data term in the visibility MRF. Visibility flags for controlling paths are fixed at 1 in the final solution and removed from the set of unknowns in the MRF, ensuring at least one path is visible at every pixel.

pass within a radius of one pixel of $(109, 123)$ in frame 12. The repeated miniatures show transported patches for the controlling path (outlined in green), as well as for the least consistent path according to $C_p(t)$ (outlined in cyan). All paths drawn in green are substantially parallel to the controlling path and so are marked as visible in this frame in $\hat{\nu}_p(t)$. The paths drawn in cyan follow substantially different paths than the controlling path and so are marked as occluded.

The values of the observed visibility flags $\hat{\nu}_p(t)$ influence the hidden visibility flags $\nu_p(t)$ through a data term in the MRF. Recall the data term of our path objective function, which includes a term

$$\sum_{t=1}^{F} \nu_p(t)\rho(\Delta I_p(t)) \tag{3.16}$$

equal to the total penalty incurred as a result of changes in brightness in the visible frames along the path. The values $\nu_p(t)$ are unknown, but we can replace them with the known observed visibility values and compute the *average* penalty along the observed visible portion of path p:

$$\Delta_p = \frac{\sum_{t=1}^{F} \hat{\nu}_p(t)\rho(\Delta I_p(t))}{\sum_{t=1}^{F} \hat{\nu}_p(t)} \; . \tag{3.17}$$

For correctly estimated paths, this measure reflects variations of intensity caused by unmodeled effects such as image noise or global illumination changes, rather than by occlusions between points with intersecting paths.

The data term of the MRF is defined as follows:

$$\begin{aligned} D(\nu_p(t) = 1) &= \rho(\Delta I_p(t)) + \lambda_L(1 - \hat{\nu}_p(t)) \\ D(\nu_p(t) = 0) &= \Delta_p + \lambda_L \hat{\nu}_p(t) \; . \end{aligned} \tag{3.18}$$

The terms with multiplier λ_L bias the estimated visibility values $\nu_p(t)$ toward the observed values $\hat{\nu}_p(t)$. Declaring a point visible incurs an additional charge $\rho(\Delta I_p(t))$, which increases with the magnitude of the change in intensity between the anchor and current point. Marking the point as occluded incurs the additional charge Δ_p that accounts for the fact that intensity variations may have causes other than occlusions. Figure 3.3 shows $\rho(\Delta I_p(t))$ and Δ_p for one of the paths drawn in cyan in Figure 3.2. The point is occluded from frame 9 to frame 19.

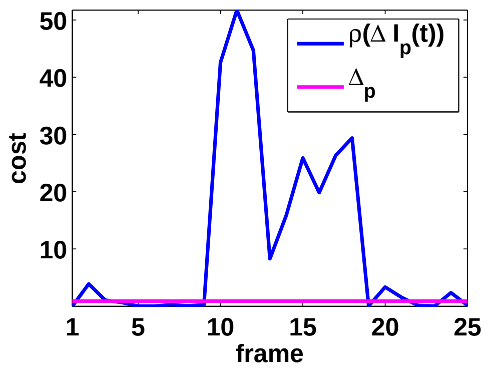


FIGURE 3.3: The penalty $\rho(\Delta I_p(t))$ (in blue) for a path drawn in cyan in Figure 3.2 compared to the penalty Δ_p (in magenta). The tracked point is occluded from frame 9 to frame 19, when the cost for the occlusion penalty Δ_p is much lower than the actual difference in intensity compared to the path's reference appearance.

The weights on edges between the nodes of the MRF encourage both temporal and spatial consistency among visibility values. Specifically, a penalty

$$V_T(\nu_p(t), \nu_p(t+1)) = \lambda_T |\nu_p(t) - \nu_p(t+1)| \tag{3.19}$$

is added between temporally adjacent neighbors to discourage frequent changes in visibility along a path. The weight on the edge between spatial neighbors

$$V_S(\nu_p(t), \nu_q(t)) = w_{pq}(t) \lambda_S |\nu_p(t) - \nu_q(t)| \tag{3.20}$$

adds an additional weight that depends on the paths p and q and the frame t. This weight is equal to

$$w_{pq}(t) = \frac{e^{-\left(\frac{\Delta I_{pq}(t)^2 + \Delta I_{pq}^2}{\sigma^2}\right)}}{\bar{d}_{pq} + \epsilon} \ , \tag{3.21}$$

where $\epsilon > 0$ prevents division by zero. In this expression,

$$\begin{aligned} \Delta I_{pq}(t) &= I(\boldsymbol{x}_p(t), t) - I(\boldsymbol{x}_q(t), t) \\ \Delta I_{pq} &= I(\boldsymbol{u}_p, \tau_p) - I(\boldsymbol{u}_q, \tau_q) \ . \end{aligned} \tag{3.22}$$

In words, $\Delta I_{pq}(t)$ measures the difference in appearance between paths in the frame in which visibility is being estimated, and ΔI_{pq} measures the difference in appearance between paths at their anchor points. The combined effect of these two terms is to push discontinuities in visibility closer to intensity boundaries.

The full energy function we wish to minimize is then

$$E_{\mathrm{MRF}} = \sum_p \sum_t D(\nu_p(t)) + \sum_q \sum_t V_T(\nu_q(t), \nu_q(t+1)) + \sum_{pq} \sum_t V_S(\nu_p(t), \nu_q(t)) \ . \quad (3.23)$$

Note that both V_T and V_S satisfy

$$V(0,0) + V(1,1) \leqslant V(0,1) + V(1,0) \ .$$

Therefore, this energy function is graph-representable as Kolmogorov and Zabih (2004) show, and it is possible to find the global minimum efficiently. However, there is no guarantee that this global minimum satisfies the physical requirement that each video pixel must have some visible object that is responsible for the appearance at that pixel. We enforce this constraint by fixing the visibility of the controlling paths. If p^* is a controlling path in frame t, we require that $\nu_{p^*}(t) = 1$ in the final solution. As long as there are enough paths for at least one path to pass through every video pixel, this guarantees at least one *visible* path through every video pixel.

This formal definition of video motion satisfies our objectives. Each visible world point is associated with a path that extends for the length of the sequence, defining a correspondence in every frame in which the point is visible. The flexibility of our definition of anchor points enables us to associate paths with world points regardless of when those points become visible in the scene. Occlusions are given first-class status, with each path reporting whether its associated point is visible in any given frame. The occlusion model incorporates evidence from every frame and allows occlusions only when an occluder can be found. Most importantly, our model is practical; as the remainder of this dissertation demonstrates, we have successfully used it to recover accurate video motion in realistic sequences.

4

Finding a Path Basis

In this chapter, we present our method for generating a sequence-specific path basis. While it is possible to use a pre-defined basis, a sequence-specific basis can more accurately and compactly represent the motion in a sequence, providing stronger regularization during the path inference process and improving the computational efficiency of the inference step. The work described here was first published in Ricco and Tomasi (2012b).

4.1 Factorization in the Presence of Missing Data

Recall from our parameterization of video motion that we wish to find a set of K basis paths $\{\boldsymbol{\varphi}_1, \ldots, \boldsymbol{\varphi}_K\}$ that span the space of allowable paths in a sequence, up to a shift. The size of the basis will vary for each sequence. Here we will use the equivalent formulation to search for a basis of size $K + 2$ that spans the space of allowable paths without a shift. The two constant basis functions $\boldsymbol{\varphi}_{K+1}(t) = (1, 0)$ and $\boldsymbol{\varphi}_{K+2}(t) = (0, 1)$ can be separated out from the first K through Gram-Schmidt orthogonalization during finalization.

We start by considering an ideal case. Let us assume that there exists a set of representative points that can be reliably tracked through the *entire* image sequence. Let (u_{fp}, v_{fp}) be the tracked location of point p in frame f returned by the frame-to-frame tracker. Our assumption is that these tracks lie close to a low-dimensional linear subspace so that we can write

$$\boldsymbol{M} \approx \boldsymbol{L}\boldsymbol{R}^T \ , \tag{4.1}$$

where the columns of $\boldsymbol{M}$ (a $2F \times P$ matrix) collect the tracked locations of each point, with u_{fp} in the first F rows and v_{fp} in the second F rows, columns of $\boldsymbol{L}$ (a $2F \times K$ matrix) correspond to the vector representation of the basis paths, and rows of $\boldsymbol{R}$ (a $P \times K$ matrix) contain the coefficients of the reconstructions of each track. Equality is only approximate because of tracking noise and other unmodeled non-linear effects. These points are representative if the track of any visible point in the sequence could be expressed with equivalent accuracy using the same basis $\boldsymbol{L}$.

The process of finding the optimal $\boldsymbol{L}$ and $\boldsymbol{R}$ for some matrix $\boldsymbol{M}$ is called matrix factorization. If the errors in the entries of $\boldsymbol{M}$ are assumed to come from an independently and identically distributed isotropic Gaussian noise process, so that the Frobenius norm is the appropriate error metric, then the best basis can be found by solving the optimization problem

$$\boldsymbol{L}^* = \arg\min_{\boldsymbol{L}} \min_{\boldsymbol{R}} ||\boldsymbol{M} - \boldsymbol{L}\boldsymbol{R}^T||_F^2 \ . \tag{4.2}$$

The global optimum can be computed as the first K left singular vectors in the singular value decomposition (SVD) of the matrix $\boldsymbol{M}$. If K is not known *a priori*, it can be selected by, for example, requiring that the basis captures a certain percentage of the variation in the original tracks. This process is equivalent to Principal Component Analysis except for the missing centering step. It is also identical to the initial steps of the work in factorization for SFM (again, disregarding centering), although we do not assign any physical meaning to $\boldsymbol{L}$ or $\boldsymbol{R}$, which in SFM correspond to the

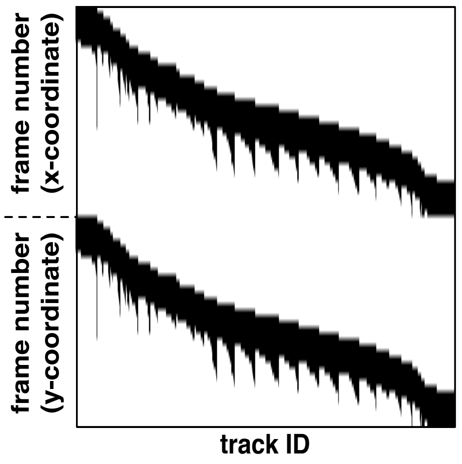


FIGURE 4.1: A typical fill pattern showing the prevalence of missing data in matrices collecting frame-to-frame tracks. Black entries are known; white entries are missing. Tracks follow points on a small toy dinosaur rotating on a turntable (see Buchanan and Fitzgibbon (2005)). The sequence is 36 frames long; we include only those tracks that survived for at least 6 frames, sorted by the frame in which they were initialized. The second 36 rows of the fill matrix (corresponding to y-coordinates of the tracks) are duplicates of the first 36 (corresponding to x-coordinates), as a single observation always contains both coordinates.

camera parameters and scene structure, respectively.

Of course, the assumption that a tracker will be able to follow enough features through the entire sequence is completely unrealistic. In practice, tracking point features is brittle because the appearance within a feature window changes as a result of many factors including image noise, variations in lighting or viewpoint, or object deformations. Even under perfect conditions, a track will be lost if the feature becomes occluded. When features are lost, new features are initialized to replace them, creating new columns of $\boldsymbol{M}$ with missing entries prior to initialization. Similarly, the columns of $\boldsymbol{M}$ corresponding to features lost in frame t have missing entries in the rows corresponding to all frames after frame t. Figure 4.1 shows a typical fill pattern for $\boldsymbol{M}$. Black regions correspond to known entries; missing entries are white. Only 20% of the entries are known in this example.

With missing entries in the matrix, the SVD cannot be used unless imputed values are used in place of the missing elements. Many researchers have addressed this problem within the framework of factorization for SFM. Tomasi and Kanade (1992) proposed imputing missing entries greedily, starting by solving the original problem with a full submatrix of $\boldsymbol{M}$ and then predicting missing entries that overlap the partial solution, thereby creating a larger submatrix that can be used to continue the process.[1] If imputed values are not used, the optimization problem to solve becomes

$$\min_{\boldsymbol{L},\boldsymbol{R}} ||\boldsymbol{F} \odot \left(\boldsymbol{M} - \boldsymbol{L}\boldsymbol{R}^T\right)||_F^2 \ , \tag{4.3}$$

where $\boldsymbol{F}$ is a binary fill matrix with $f_{ij} = 1$ if and only if m_{ij} is known and '$\odot$' is the Hadamard (*i.e.*, entry-by-entry) product. The solution is typically found via alternation (Wiberg, 1976), where one iterates between solving for $\boldsymbol{L}$ with $\boldsymbol{R}$ fixed and vice versa, or through direct non-linear numerical optimization. Buchanan and Fitzgibbon (2005) provide an excellent summary of many proposed alternation techniques and present a damped Newton method to perform direct non-linear optimization.

Interest in this type of problem is by no means restricted to the computer vision community. In particular, the problem of low-rank matrix completion is a widely studied problem recently popularized by the Netflix Prize, a \$1 million challenge to improve movie recommendations for Netflix users through collaborative filtering (Bennett and Lanning, 2007). The noise-free matrix completion problem solves

$$\begin{aligned} &\textit{minimize} \ \operatorname{rank}(\boldsymbol{X}) \\ &\textit{subject to} \ \mathcal{P}_\Omega(\boldsymbol{X}) = \mathcal{P}_\Omega(\boldsymbol{M}) \ , \end{aligned} \tag{4.4}$$

where Ω identifies the known entries of the matrix $\boldsymbol{M}$ and the function $\mathcal{P}_\Omega(\cdot)$ replaces unknown entries of its argument with zeros. Minimizing the rank (equal to the number of non-zero singular values of $\boldsymbol{X}$) results in an NP-hard problem, which

[1] Unfortunately, the problem of finding the largest submatrix with which to start is NP-hard (by reduction from max-clique).

is often replaced with its convex relaxation, the minimization of the nuclear norm $||\boldsymbol{X}||_*$ (equal to the sum of the singular values of $\boldsymbol{X}$). If entries of the matrix $\boldsymbol{M}$ are corrupted by noise, then the problem becomes

$$\begin{aligned} &\textit{minimize}\ ||\boldsymbol{X}||_* \\ &\textit{subject to}\ ||\mathcal{P}_\Omega(\boldsymbol{X}-\boldsymbol{M})||_F \leqslant \delta \end{aligned} \tag{4.5}$$

for some $\delta > 0$. Candès and Recht (2009) proved that (with high probability) the unique solution to this optimization problem with $\delta = 0$ (the noise-free case) perfectly predicts the missing entries of an $n_1 \times n_2$ matrix of rank r, provided that $m \geqslant Cn^{1.25} r \log n$ where m is the number of known entries and $n = \max(n_1, n_2)$. For this theorem to apply, the matrix must satisfy certain incoherence properties. The intuition behind the incoherence requirement is that it is not possible to recover a matrix with non-zero entries concentrated in some small neighborhood unless those entries are sampled. As a concrete example, consider trying to recover the matrix that contains a single non-zero element in the upper-left corner (all other entries are zero). This matrix has rank 1, but even if every other element of the matrix is known, it is impossible to recover the non-zero element. Candès and Plan (2010) considered the noisy case under similar assumptions and proved that recovery is stable, with errors proportional to the noise level δ. We refer interested readers to the cited works for the exact statements of these and other related theorems.

Unfortunately, neither of these theoretical guarantees (nor later improvements) applies to our problem because both rely on one crucial assumption in addition to the incoherence property: the assumption that the known entries of the matrix are sampled uniformly at random (UAR). UAR sampling requires that whether the entry at (i, j) is known or unknown is independent of both the value of the indices i and j and the value of the entry itself. This is clearly violated in the tracking case, where entries are known over intervals of the frame index. The switch from known to unknown often results from either an occlusion or track loss at the image boundary, processes that depend on the value of the first missing entry. Because of this, and as can be seen in Figure 4.1, the pattern of known elements is highly structured. As a result, imputed entries are often of lower quality than would be predicted by the theoretical results.

In our problem, the imputed entries extrapolate from the limited length tracks available from a frame-to-frame tracker into full-length video motion paths. When finding the optimal paths given the basis, we are able to leverage information about the appearance of the video along the path that is not used in the traditional matrix completion problem. Nevertheless, geometric information is valuable; if we can find a basis that can accurately impute values based on the geometric information alone, our task becomes that much easier.

We improve imputation of missing entries by recognizing that, although every newly initialized feature creates a new column of the matrix $\boldsymbol{M}$, in fact many of these new features follow points that were tracked before. If we can recognize these re-identified features, we can collapse the respective columns of $\boldsymbol{M}$ into a single column, forming a compact matrix $\hat{\boldsymbol{M}}$. The new matrix will have a greater percentage of observed elements and the distribution of these observed entries will have less structure than in the original matrix (although the sampling process will still not be independent of the value of the elements). In the next sections, we present a history-sensitive feature reinitialization procedure that actively works to identify previously lost features in a frame-to-frame tracker. These tracks feed into an optimization routine that simultaneously searches for the best low-rank factorization and compacts the matrix being factored by merging columns that correspond to the same world point. At convergence, the path basis $\{\boldsymbol{\varphi}_1, \ldots, \boldsymbol{\varphi}_K\}$ is computed by factoring out the contribution of the constant basis functions from the columns of the matrix $\boldsymbol{L}$.

Surprisingly, the opportunity for matrix compaction during factorization is rarely considered in the SFM literature. When correspondences provided by the tracker are considered as untrustworthy, it is most often within the framework of robust factorization in which individual correspondences may be flagged as outliers and discarded (*e.g.*, Olsen and Bartoli, 2008; De la Torre and Black, 2001; Ke and Kanade, 2005). This is the reverse of the problem we focus on, as we look for tracks that should have been merged and were not rather than associations that were made and should not have been. The work in Dellaert et al. (2000) is a noticeable exception to this general rule. The authors assume that no correspondences are known at all and propose a Monte Carlo approach for sampling possible correspondences. They solve for perspective SFM to verify proposals. Our compaction approach differs in three ways. First, we assume that some, but not all, correspondences are known; that is, we assume that a point-feature tracker often works. Second, we use a more general low-rank matrix model in place of the rigidity assumed in perspective SFM, making our approach applicable to multi-body or non-rigid motions. Finally, we allow the appearance of tracked points to inform predicted associations while the approach taken by Dellaert et al. uses geometric information only. The "track gluing" experiment of Olsen and Bartoli (2008) performs compaction, but it considers merging tracks only after factorization is complete and does so based on geometric information only. In contrast, we propose to interleave matrix compaction with matrix factorization and include the appearance of tracks in the decision about compaction.

4.2 Tracking with History-Sensitive Feature Snapping

It is standard practice to initialize new tracks in a KLT tracker to replace lost tracks based on a stateless analysis of the current frame. Points surrounded by sufficient texture are found, and a predefined distance threshold is used to retain points that are far enough from each other and from the live points still being tracked. Similarly, optical flow-based trackers such as Sundaram et al. (2010) or Wang et al. (2011) also sample points with significant texture, although they may not necessarily filter based on the distance between points. Since this approach to reinitialization only looks at the current frame, new features may be detected at image locations that are close but not necessarily identical to those of features that had been lost earlier and have now reappeared. For example, a previously tracked feature will presumably once again pass the texture threshold. However, new features are initialized at integer pixel coordinates even though true motion between frames is generally real-valued. If we were to merge two such columns, the tracks between the two would be slightly misaligned, and doing so would introduce artificial jitter into the space of paths.

To address this problem, we introduce a history-sensitive form of feature reinitialization that positions new tracks preferentially at points that have been previously seen but were lost for a short amount of time. We call this procedure *feature snapping*. To support feature snapping, we maintain a record of the temporally-averaged image patch around each live feature. When a track dies, we move its patch record to a catalog of lost patches. At reinitialization time, we first form a list of feature candidates with the traditional, stateless method. We then compute the affine motion between each candidate in the list and the features in the catalog to adjust the initial position of the candidate feature to better align with the historical appearance of a previously tracked patch. We compute the photometric distance between the aligned patches by computing the sum of squared differences in intensity. If this distance is below a set threshold, we use the location of the center of the patch after alignment as the coordinates of the new feature instead of the initial pixel coordinates.

The catalog of lost features can grow quickly, particularly in sequences with frequent occlusions or significant non-rigid deformations. In this case, we may not want to search the entire catalog for every new candidate feature. We can limit our search by using geometric information. For example, we can augment the record of the feature in the catalog with its last known location and only try to snap to features that were last seen within some fixed distance. This assumes that features do not move (much) when they are not seen and biases the system toward tracks that are only briefly lost, *e.g.*, due to image noise rather than true occlusions. If motion in the scene is significant or occlusions are of considerable duration, we can use more sophisticated models of motion to predict the location of a lost feature in the current frame. For example, we could assume that velocity is constant and predict locations based on the most recent (or average) observed velocity of each feature in the catalog. The size of the search window centered on this predicted location grows based on the amount of observed variation in the velocity during the tracked interval and the time since the point was last observed. Of course, both these proposals are simply heuristics used to reduce the amount of computation required. More complicated prediction strategies could be used. Recall that we will eventually be predicting the tracks of lost features using a general low-rank assumption that incorporates information about the motion of all tracks in the scene, but that model is not available at this point in the computation.

Importantly, when snapping occurs, we do not assume that the new feature and the one it is snapped to should have their columns merged. We leave those decisions to the compaction algorithm, which is able to use the stronger constraints from rank, geometry, and photometric appearance to determine if merging is appropriate.

Our work here addresses only the feature reinitialization procedure, not the process for tracking points once initialized. For that, we use a standard KLT tracker or an optical flow-based tracker. The algorithm by Sundaram et al. (2010) is better at tracking through large motions and can be used to add tracks in low-texture areas for sequences without many otherwise trackable features. In both cases, we tune parameters of the trackers to terminate tracks very conservatively (*i.e.*, earlier than one might normally terminate a track). We do this to eliminate as many outlier tracks as possible because the quadratic penalty in the matrix completion step is sensitive to outliers. Early terminations result in more missing elements in the original (uncompact) matrix $\boldsymbol{M}$; we rely on our compaction step to compensate.

4.3 Matrix Compaction

Once feature tracks have been computed, we proceed with the compaction-and-factorization step. Let $\boldsymbol{C}$ be an $n \times n$ binary symmetric *compaction matrix* with entries $c_{ij} = 1$ if and only if track i and track j correspond to the same world point and therefore should be merged into the same column. Because the compaction matrix defines an equivalence relationship between track fragments, its entries are required to obey the transitive property: If $c_{ij} = c_{jk} = 1$, then $c_{ik} = 1$. The compact version $\hat{\boldsymbol{M}}$ of $\boldsymbol{M}$ consists of the unique columns of the matrix $\boldsymbol{MC}$ when we fill missing elements of $\boldsymbol{M}$ with zeros. We encourage compaction by adding a term to the standard factorization objective function (4.2) that is proportional to the number of zeros in $\boldsymbol{C}$. Recall that we wish to minimize this objective.

We include additional heuristics and constraints to ensure that the recovered compaction is plausible. First, two tracks i and j are *temporally consistent* if they do not overlap in time. Only temporally consistent tracks may be merged because a single point cannot appear in two locations in a single frame. The fill matrix $\boldsymbol{F}$ indicates which tracks are observed in which frames; the entry at (f, i) of the fill matrix after compaction $\boldsymbol{FC}$ counts the number of times a point associated with track i is observed in frame f. Recall that row f and row $2f$ of $\boldsymbol{F}$ are identical, so the entry at $(2f, i)$ stores the same information. The requirement that associated tracks must be temporally consistent can be enforced mathematically by adding the constraint $||\boldsymbol{F}_x\boldsymbol{C}||_\infty = 1$ where $\boldsymbol{F}_x$ denotes the matrix consisting of the first F rows of $\boldsymbol{F}$ (corresponding to the locations of observed x-coordinates). This constraint is linear in the unknown variables c_{ij}.

Two tracks are *photometrically consistent* if small image windows around them look similar. Specifically, let I_i be the average image over the frames in which track i was observed, with individual frames aligned according to the motion of the track. That is, with the tracked point located at (x_{fi}, y_{fi}) in frame f, and tracked between frames f_1 and f_2,

$$I_i(x,y) = \frac{1}{f_2 - f_1 + 1} \sum_{f=f_1}^{f_2} I(x + x_{fi} - x_{f_1 i}, y + y_{fi} - y_{f_1 i}, f) \ . \tag{4.6}$$

We use bilinear interpolation to compute intensities at real-valued image locations and replace missing values (out of the field of view of the frame) with zeros. Let I_j be this average image for track j. Note that the patch from I_i centered at $\boldsymbol{x}_i = (x_{f_1 i}, y_{f_1 i})$ is the patch stored in the lost feature catalog during tracking. As before, we solve for the affine deformation that best aligns the correct patches from the two images and store the image difference in an $n \times n$ lower-triangular *photometric discrepancy matrix* $\boldsymbol{\Phi}$ with entries

$$\phi_{ij} = \min_{\boldsymbol{A} \in \mathbb{R}^{2\times 2},\, \boldsymbol{d} \in \mathbb{R}^{2\times 1}} \sum_{\Omega} ||I_i(\boldsymbol{x} - \boldsymbol{x}_i) - I_j(\boldsymbol{A}(\boldsymbol{x} - \boldsymbol{x}_j) + \boldsymbol{d})||_F^2 \quad \text{for} \quad i > j \ . \tag{4.7}$$

Because the same point should look similar in all the frames in which it is tracked, merged tracks should be photometrically consistent. We encourage photometric consistency by adding to the penalty function (4.2) a term proportional to $\boldsymbol{\phi}^T \boldsymbol{c}$ where $\boldsymbol{\phi} = \text{vec}(\boldsymbol{\Phi})$ and $\boldsymbol{c} = \text{vec}(\boldsymbol{C})$. The expression vec($\boldsymbol{A}$) for a matrix $\boldsymbol{A}$ denotes the column vector obtained by listing all the entries of $\boldsymbol{A}$ in an arbitrary but fixed order.

Finally, track fragments that are merged must be *geometrically consistent*, meaning that the same reconstructed track (*i.e.*, full length path) can approximate the observations from both track fragments well. We can enforce this by adding a constraint that $c_{ij}(r_{ik} - r_{jk}) = 0$ for all i, j, k with $i > j$. In words, either track i and j

are kept separate so that $c_{ij} = 0$, or they are merged, in which case the corresponding rows of $\boldsymbol{R}$ in equation (4.3) must be identical.

In summary, simultaneous compaction and factorization solves the following optimization problem:

$$\begin{aligned} \min_{\boldsymbol{L},\boldsymbol{R},\boldsymbol{C}} \ & ||\boldsymbol{F} \odot \left(\boldsymbol{M} - \boldsymbol{L}\boldsymbol{R}^T\right)||_F^2 + \lambda_1 \mathbf{1}^T(\mathbf{1} - \boldsymbol{c}) + \lambda_2 \boldsymbol{\phi}^T \boldsymbol{c} \\ s.t. \quad & \boldsymbol{C} = \boldsymbol{C}^T \\ & c_{ij} + c_{jk} \leqslant c_{ik} + 1 \quad \forall i, j, k \\ & ||\boldsymbol{F}_x \boldsymbol{C}||_\infty = 1 \\ & c_{ij}(r_{ik} - r_{jk}) = 0 \quad \forall i, j, k \end{aligned} \tag{4.8}$$

with $\boldsymbol{L} \in \mathbb{R}^{2F \times K}$, $\boldsymbol{R} \in \mathbb{R}^{n \times K}$, and $\boldsymbol{C} \in \{0, 1\}^{n \times n}$. The first two constraints ensure that $\boldsymbol{C}$ is symmetric and satisfies the transitive property. The last two enforce temporal and geometric consistency, respectively.

4.3.1 Optimization

The mixed integer quadratically constrained nonlinear program (4.8) is intractable; solving it exactly is impractical for typical problem sizes. We find an approximate solution by alternating between performing matrix factorization with a fixed compaction and finding a new candidate compaction matrix.

With compaction fixed, only the first term of the objective function and the last constraint are relevant. We enforce the constraint by collapsing $\boldsymbol{M}$ into $\hat{\boldsymbol{M}}$ and performing factorization on the smaller matrix. We treat the actual factorization of the compacted matrix as a black box. In our current implementation, we use the factorization algorithm of Gotardo and Martinez (2011). This method supports adding priors on the temporal smoothness of the motion by restricting the recovered basis to be a linear combination of the low-frequency DCT basis vectors. We run the algorithm with the complete DCT basis, so no smoothness is enforced. The solution to this step is a factorization $\hat{\boldsymbol{M}} \approx \boldsymbol{L}\hat{\boldsymbol{R}}^T$. We recover $\boldsymbol{R}$ by duplicating rows of $\hat{\boldsymbol{R}}$ according to the associations defined in the compaction matrix.

It is highly unlikely that factorization will recover identical entries in two columns of $\boldsymbol{R}$ that have yet to be compacted. The entries in $\boldsymbol{R}$ are real valued and are estimated from disjoint track fragments that have each been corrupted by noise. In a two-step process, where a new estimate of $\boldsymbol{C}$ must use a fixed estimate of $\boldsymbol{L}$ and $\boldsymbol{R}$, requiring strict equality for compacted track fragments would likely rule out all correct associations. Instead, we want to relax the constraint so that two columns may be compacted if they are similar but not identical. We replace the constraint with a heuristic that measures how well the full-length reconstruction of one track fits the observed entries of a different track. The idea is that if the reconstruction of track i fits the observations of track j nearly as well as the current (unconstrained) reconstruction of track j does, then compacting these two columns is reasonable.

We define a lower-triangular *geometric discrepancy matrix* $\boldsymbol{G}$ to have entries

$$g_{ij} = \min(D(i,j), D(j,i)) \quad \text{for} \quad i > j \; , \tag{4.9}$$

where

$$D(i,j) = ||\boldsymbol{f}_j \odot \left(\boldsymbol{L}\boldsymbol{r}_i^T - \boldsymbol{m}_j\right)||_2^2 \; . \tag{4.10}$$

In this expression, $\boldsymbol{f}_j$ is the jth column of $\boldsymbol{F}$, which identifies the frames in which track j is observed; $\boldsymbol{r}_i^T$ is the ith column of $\boldsymbol{R}^T$ so $\boldsymbol{L}\boldsymbol{r}_i^T$ is the reconstruction of track i; $\boldsymbol{m}_j$ is the jth column of $\boldsymbol{M}$, which contains the observed locations of track j. In words, the distance term $D(i,j)$ is the total distance between the imputed version of track i and the observed version of track j in the frames in which track j is visible. The minimum of $D(i,j)$ and $D(j,i)$ yields a symmetric distance function and represents the additional geometric fitting penalty that would be paid if the factorization procedure selected the best of the two current reconstructions to use for both track fragments. This quantity is an upper bound on the actual additional penalty that would be paid in the original objective function if factorization were allowed to search for the best single track to use for both tracks i and j. In this way, the value measures the predicted cost of enforcing the strict equality constraint.

To find a new compaction, we solve the integer linear program (IP)

$$\begin{aligned} \min_{\boldsymbol{C}} \quad & \boldsymbol{g}^T \boldsymbol{c} + \lambda_1 \mathbf{1}^T (\mathbf{1} - \boldsymbol{c}) + \lambda_2 \boldsymbol{\phi}^T \boldsymbol{c} \\ s.t. \quad & \boldsymbol{C} = \boldsymbol{C}^T \\ & c_{ij} + c_{jk} \leqslant c_{ik} + 1 \quad \forall i, j, k \\ & ||\boldsymbol{F}_x \boldsymbol{C}||_\infty = 1 \; . \end{aligned} \tag{4.11}$$

In these expressions, $\boldsymbol{g} = \text{vec}(\boldsymbol{G})$. We iterate between proposing a compaction and factoring with the constraints imposed by that compaction until the proposed compaction matrix stabilizes. The solution with the lowest cost according to the objective function from (4.8) encountered during the iteration is returned as the final solution. Because of the approximation used in place of the geometric consistency constraint, some compactions may be missed. We greedily perform single compactions (in which a single pair of columns of $\hat{\boldsymbol{M}}$ are merged) until none are left that decrease the value of the full objective function. Although this procedure does not guarantee that we reach a global minimum of our original objective, we achieve good results in practice, often recovering very close to the ground truth compaction.

The algorithm as described assumes that we know the rank of the matrix during factorization. In practice, K is often not known. We select its value for a particular sequence by slowly increasing K until the observed entries of $\boldsymbol{M}$ are reconstructed to within a set threshold. More advanced model selection techniques could be used to estimate K. We have observed that adding compaction during factorization tends to make the solution more robust to overestimates of the rank. By this we mean that the accuracy with which we can impute missing entries is stable for values of K that slightly overestimate the true rank. It may be possible to correct for an overestimated rank by examining the magnitude of the singular values from the SVD of the imputed version of $\boldsymbol{M}$ at convergence. We present the details of these experiments in Section 4.4.3.

4.3.2 Pre-solving and constraint generation

A naive implementation of the integer program (4.11) solves for n^2 variables, but over half can be immediately eliminated because of the first set of constraints that force $c_{ij} = c_{ji}$. Thus, we only actually need to solve for c_{ij} with $i > j$. Even more unknowns can be eliminated in a pre-solve step based on the max-norm constraint; we know that $c_{ij} = 0$ for any pair of tracks visible in the same frame. This pre-solve step often reduces the number of unknowns for sequences with approximately 100 tracks per frame to the point where the IP can be solved using standard off-the-shelf techniques. In particular, we use the Gurobi Optimizer (Gurobi Optimization, Inc., 2013), which solves IPs using linear programming-based branch-and-bound.

For larger problem sizes, the number of constraints needed to enforce the transitivity property becomes prohibitive. However, very few of these constraints are tight at the final solution. Instead of constructing all constraints from the start, we use constraint generation to enforce only the constraints we believe will be tight. We first solve using only the constraints from $||\boldsymbol{F}_x\boldsymbol{C}||_\infty = 1$. (The equality constraints $\boldsymbol{C} = \boldsymbol{C}^T$ have already been eliminated as described above.) We then check for any violations of the transitivity property, that is, for $i \neq j \neq k, i \neq k$ such that $c_{ij} = c_{jk} = 1$ but $c_{ik} = 0$. If no violated constraints are found, the solution found is an optimal solution to the problem with all constraints imposed. If violations are found, we add the violated constraints and re-solve.

In Figure 4.2, we report the time to solve one iteration of the IP for synthetic datasets of various sizes. The naive implementation quickly becomes prohibitively expensive; pre-solving decreases the running time, but both versions require more

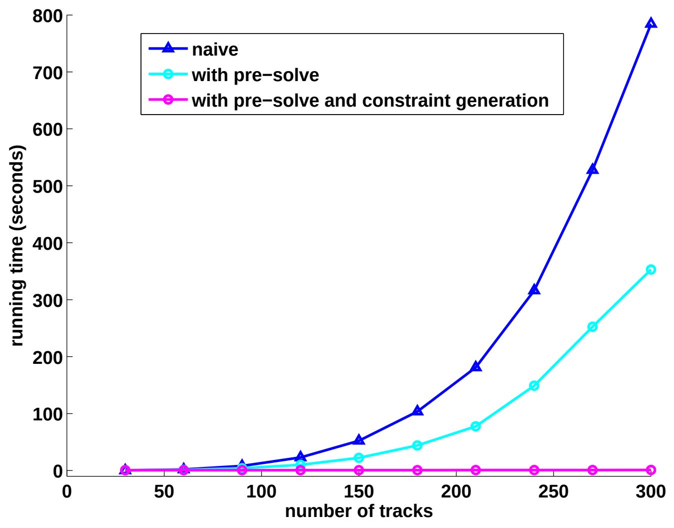
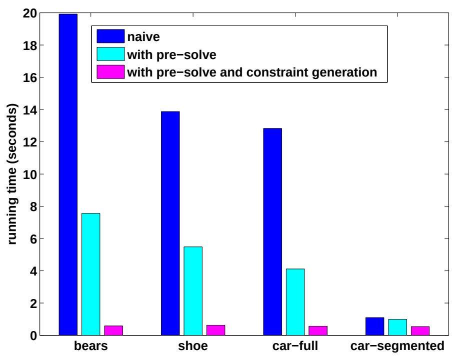
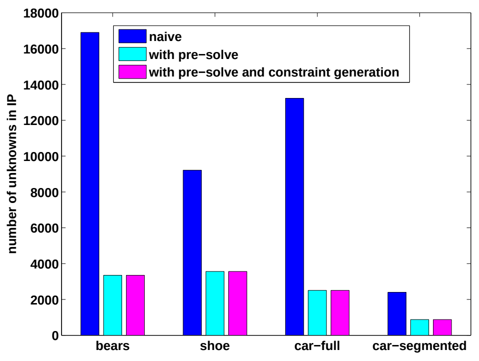


FIGURE 4.2: Time to compute a single compaction proposal on synthetic datasets with increasing numbers of tracks. A set of tracks of size $3N$ is generated by creating N full tracks, breaking them into five random pieces, and considering the first, third, and last segment of each track as separate observed track fragments. The computation scales to large problem sizes when pre-solving and constraint generation are included (magenta). Pre-solving without constraint generation (cyan) results in a noticeable speedup compared to the naive implementation (blue), but the improvement is not enough to scale to sequences that generate many track fragments.

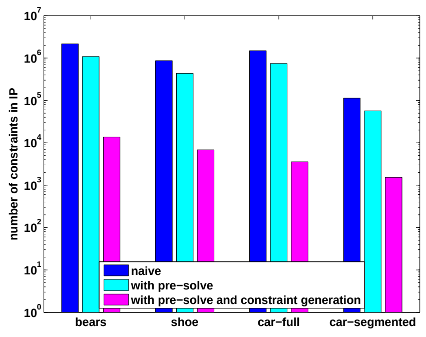
than 64GB of memory to solve for compactions with 600 track fragments. Adding constraint generation greatly improves the situation: running times and memory requirements scale gracefully, and the Gurobi Optimizer can compute solutions for 3000 tracks in just 45 seconds. (With the naive implementation, it takes 45 seconds to solve a problem with just 150 tracks.) Behavior on real sequences is similar, as demonstrated in Figure 4.3. Figure 4.3a shows the time required for computing a proposed compaction using the naive implementation, pre-solving, or pre-solving with constraint generation. Figure 4.3b shows the number of unknowns in the resulting IP for the different methods; Figure 4.3c shows the number of constraints enforced for each.



(a) Time to compute a compaction proposal.



(b) Number of unknowns in the IP.

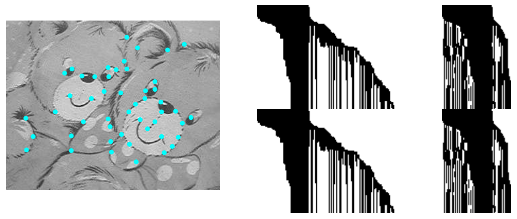


(c) Number of generated constraints.

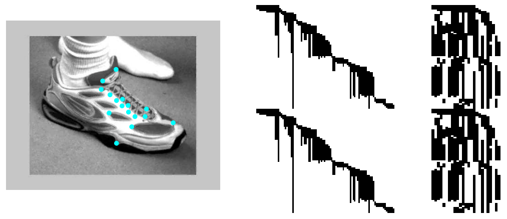
FIGURE 4.3: Comparison of the naive implementation (blue), the integer program after pre-solving (cyan), and the integer program with pre-solving and constraint generation (magenta) on four real sequences. Note the logarithmic scale in the y-axis of (c). See Section 4.4 for details on the sequences tested.

4.4 Results

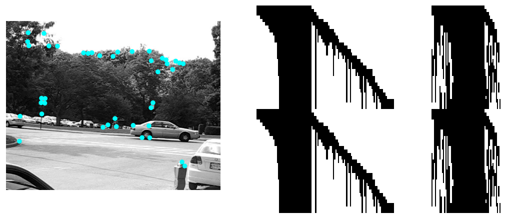
Because our video motion technique relies so heavily on the existence of a good sequence-specific basis, we present the results for basis extraction before continuing to describe the rest of the optimization procedure. In order to use our full system as designed, we need to find reliable bases for sequences without any full-length frame-to-frame tracks. We prove that we can accomplish this task here. The results for the



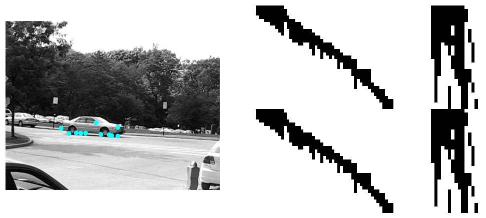
(a) bears



(b) shoe



(c) car-full



(d) car-segmented

FIGURE 4.4: Example frames, with tracked features marked, fill matrices before compaction, and fill matrices after the correct compaction is applied for the four real sequences we test. The matrices follow the layout of Figure 4.1. The 105-frame *bears* sequence generates 130 tracks from 68 unique features. Before compaction, 62% of the entries in $\boldsymbol{M}$ are missing. With the ground truth compaction applied, 28% of the entries are missing. The 71-frame *shoe* sequence generates 96 observations of 25 unique features. $\boldsymbol{M}$ is 85% missing before compaction and 42% missing under perfect compaction. Two car sequences track points through the same 31-frame sequence. In *car-full*, there are 115 tracked features, 60 of which are unique. Perfect compaction reduces the amount of missing data from 61% to 25%. Manually discarding the tracks on the background leaves 49 tracks in the *car-segmented* sequence, 14 of which are unique. Perfect compaction reduces the amount of missing data from 84% to 45%.

full video motion system can be found in Chapter 6.

We test simultaneous compaction and factorization on four sets of tracks. See Figure 4.4 for details of each set. The tracks are generated using a KLT tracker augmented to use history-sensitive feature reinitialization. We do not artificially split any tracks; rather, column splitting is a result of feature loss and automatic reacquisition. We determine the ground truth associations for the recovered tracks and generate a ground truth measurement matrix $\boldsymbol{M}^*$ by hand tracking the acquired points throughout the sequence. We use fixed parameters $\lambda_1 = 30, \lambda_2 = 1$ for all runs. For the first two experiments, we assume we know the desired rank K.

4.4.1 Accuracy of compaction

Our first experiments test how well compaction is able to recover correct associations between track fragments. We can evaluate performance using standard metrics for clustering algorithms as ideally the compaction matrix partitions the columns of $\boldsymbol{M}$ into clusters corresponding to unique world points. Given two partitions, $\mathcal{P}$ and $\mathcal{Q}$, of a set of n points, the Rand index (Rand, 1971) is defined as

$$\mathrm{RI} = \frac{a+d}{\binom{n}{2}} \ , \tag{4.12}$$

where a counts the pairs of points in which the two elements of the pair are assigned to the same cluster in both $\mathcal{P}$ and $\mathcal{Q}$ and d counts the pairs of points whose elements are assigned to different clusters in both $\mathcal{P}$ and $\mathcal{Q}$. In words, the numerator counts the pairs on which the two partitions agree, and the denominator counts all possible pairs. This value can range between 0 and 1, but a value of 0 only occurs when $\mathcal{P}$ groups all points in a single cluster and $\mathcal{Q}$ leaves every point in its own individual cluster (or vice versa). For any other assignment, there will be many pairs of points correctly placed in different clusters in $\mathcal{P}$ and $\mathcal{Q}$ even when both partitions are completely random. The adjusted Rand index of Hubert and Arabie (1985) corrects for the effects of chance, creating an index that ranges from -1 to 1 and is expected to take values close to zero when comparing two random partitions. The formula for this metric is

$$\mathrm{ARI} = \frac{\sum_{ij}\binom{n_{ij}}{2} - \left[\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}\right]/\binom{n}{2}}{\frac{1}{2}\left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\right] - \left[\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}\right]/\binom{n}{2}} \ , \tag{4.13}$$

where n_{ij} is the total number of points in cluster i in $\mathcal{P}$ and in cluster j in $\mathcal{Q}$, $a_i = \sum_j n_{ij}$ is the number of points in cluster i in $\mathcal{P}$, and $b_j = \sum_i n_{ij}$ is the number of points in cluster j in $\mathcal{Q}$.

TABLE 4.1: Accuracy of compaction, measured using the Adjusted Rand Index (ARI) between the computed and ground truth associations. Our technique returns a more accurate estimate of the track fragments that should be associated than techniques that consider geometric or photometric information in isolation.

Sequence	**Adjusted Rand Index** (estimated vs. true)			
	Geo-only	Photo-only	Photo+Geo	Compaction
car-segmented	0.1659	0.3267	0.4632	**1.0000**
car-full	0.0205	0.5120	0.5570	**0.9146**
bears	0.0384	0.5078	0.1784	**0.9637**
shoe	0.0416	0.2920	0.3886	**0.9825**

We compare to three baseline data association techniques: one that uses only geometric evidence, one that uses only photometric evidence, and one that uses photometric evidence to prime the geometric-only technique. The geometric-only alternative runs factorization without compaction and then merges temporally consistent tracks with geometric consistency g_{ij} below a threshold corresponding to an average geometric error of one pixel per observation. This method is modeled on the track gluing procedure of Olsen and Bartoli (2008). The photometric-only alternative associates two tracks if they are respective best-matches according to the photometric discrepancy matrix Φ (*i.e.*, $i = \arg\min_j \phi_{ij}$ and $j = \arg\min_i \phi_{ij}$) with ϕ_{ij} below a threshold. We use the parameter λ_1 as the threshold. The photometric-geometric bootstrapping technique uses the photometric-only step to compute associations which are used during factorization. The geometric-only technique is then applied to these reconstructed tracks to determine the final associations. As Table 4.1 shows, compaction recovers clusters of track fragments that are much more similar to the ground truth than any of the baseline techniques.

4.4.2 Accuracy of recovered paths

Next, we examine whether the added constraints provided by compaction improve the ability to extrapolate from a frame-to-frame track into a full-length path based on geometric information alone. This is in contrast to the results we present in Chapter 6, where we find video motion paths for all points, not just those initially tracked with a frame-to-frame tracker, using photometric evidence and accounting for occlusions.

Recall that $\boldsymbol{M}^*$ collects the ground truth locations of the tracked points in every frame. $\boldsymbol{M}$ is a noisy version of $\boldsymbol{M}^*$ with only some entries observed. The total error in the extrapolated paths is $||\boldsymbol{M}^* - \boldsymbol{L}\boldsymbol{R}^T||_F^2$. Rather than including the errors in all entries in a single metric, we group the entries based on the temporal distance to the nearest observation of the corresponding track. At the coarsest level, this grouping separates errors in reconstruction of the observed entries, which measure the denoising capacity of the low-rank assumption, from errors in imputation of the unobserved entries, which measure the ability to generalize, *i.e.*, the degree to which the recovered basis $\boldsymbol{L}$ can constrain an entire path based on only a limited number of observations. In general, we expect the accuracy of imputed entries to decrease as the temporal distance to the nearest observation of the tracked point increases, so we consider *generalization error* as a function of this temporal distance. That is, if track j is tracked from frame 10 to frame 15, we expect better imputation performance for the entries corresponding to frame 17 (a temporal distance of 2) than the entries corresponding to frame 1 (a temporal distance of 9). Let $\boldsymbol{F}(t)$ be a $2F \times P$ binary matrix with non-zero entries identifying the corresponding entries of $\boldsymbol{M}^*$ with temporal distance t. $\boldsymbol{F}(0)$ is the original fill matrix used in our objective function. Figure 4.5 shows examples with increasing values of t. Generalization error is computed as

$$\mathrm{GE}(t) = \sqrt{\frac{||\boldsymbol{F}(t) \odot (\boldsymbol{M}^* - \boldsymbol{L}\boldsymbol{R}^T)\,||_F^2}{||\boldsymbol{F}(t)||_F^2}}\,. \tag{4.14}$$

The numerator computes the sum of the squared errors in all the entries at a par-

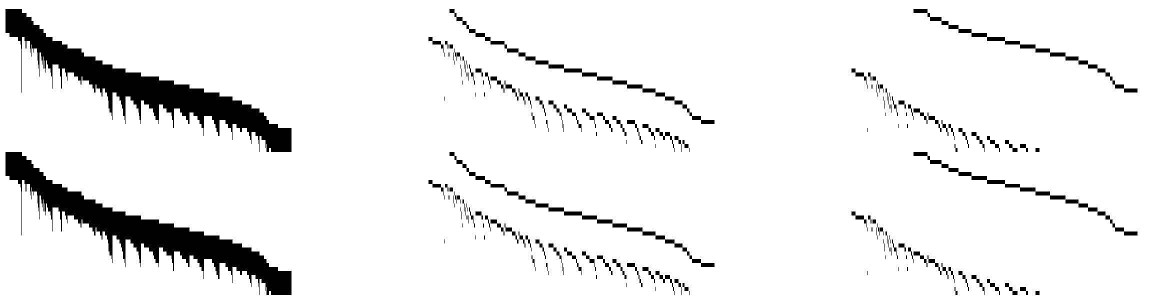

(a) $\boldsymbol{F}(0)$ (b) $\boldsymbol{F}(2)$ (c) $\boldsymbol{F}(10)$

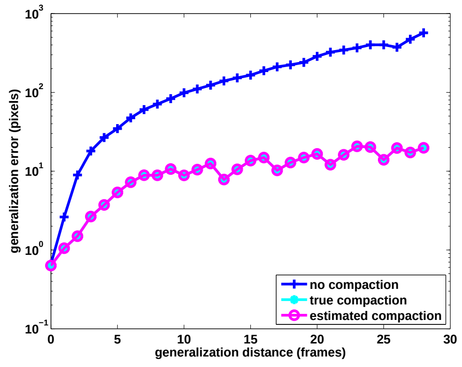
FIGURE 4.5: Selected temporal distance masks for the fill pattern in Figure 4.1.

ticular temporal distance; the denominator computes the total number of entries at that temporal distance.

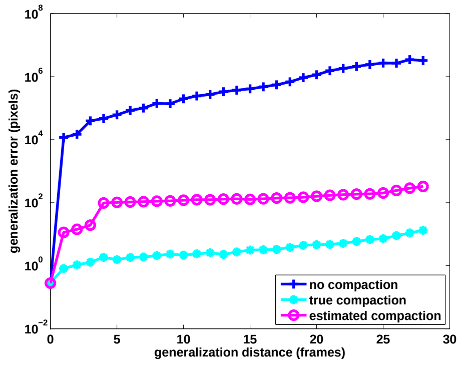
Note that, although similar at first glance, GE(0) is not a function of the quantity we minimize during factorization. Both measure error in the same entries of the reconstructed matrix $\boldsymbol{L}\boldsymbol{R}^T$, but here the errors are computed with respect to the ground truth locations, not to the observed locations in $\boldsymbol{M}$.

We compare our results to two baseline methods. The first demonstrates the overall difficulty of the imputation task by using the true compaction matrix to form $\hat{\boldsymbol{M}}$ and then factoring (*true compaction*). The opposite extreme is a standard method that performs no compaction before factoring (*no compaction*). Generalization performance should be best for the true compaction method because the added geometric consistency constraint forces the reconstruction to be close to the observed location of a point any time it is observed, which is potentially many frames removed from the last observation of one of the corresponding fragments. Our results are summarized in Figure 4.6, with the performance of our algorithm shown in magenta. In every case, our technique significantly increases imputation accuracy over the standard approach.

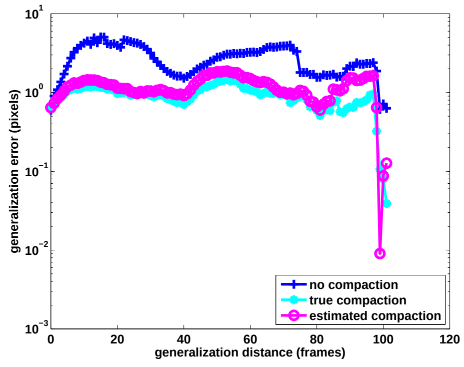
The increase in imputation accuracy is the result of two factors. First, impu-



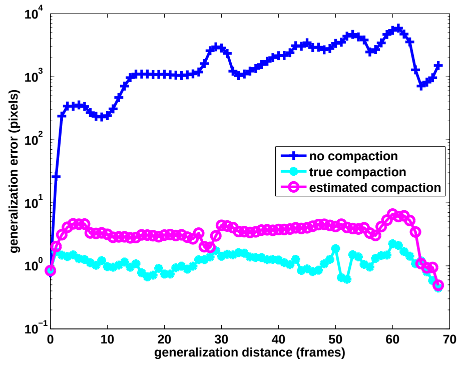
(a) car-segmented, $K = 3$



(b) car-full, $K = 6$



(c) bears, $K = 4$



(d) shoe, $K = 4$

FIGURE 4.6: The generalization error for the four test sequences with and without compaction. Blue shows the results of factorization without any compaction. Our algorithm (magenta) uses the estimated compaction during factorization and recovers missing entries better than standard factorization. The oracle algorithm, shown in cyan, achieves the best performance by using the ground truth associations to compact the matrix before factorization. Our results are identical to the oracle for the car-segmented sequence because we exactly recover the true associations.

tation improves simply because of the geometric consistency constraint; the same reconstructed track must fit all associated track fragments equally well, so imputed locations are highly accurate when associations are correctly recovered, regardless of the basis. Second, the recovered basis $\boldsymbol{L}$ is more reliable with compaction included. Figure 4.7 isolates the effects of these two different components of the solution. The blue and magenta lines are copies of the same from Figure 4.6a; the magenta line is

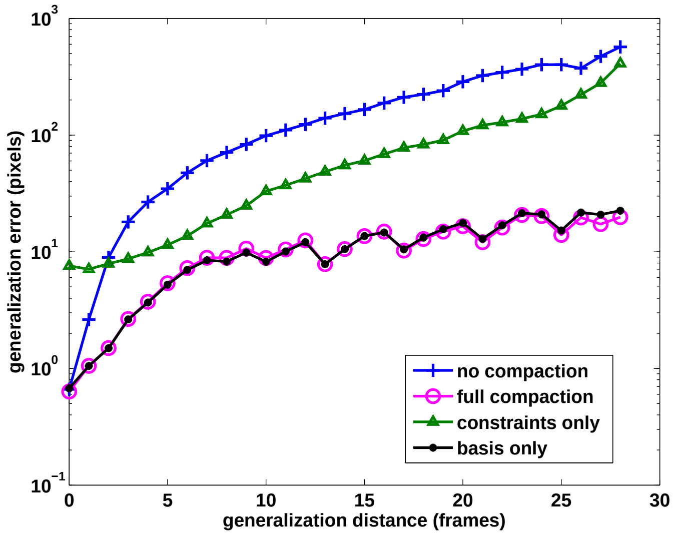


FIGURE 4.7: Evaluation of the cause of the observed improvement in generalization performance. The blue and magenta lines show generalization error on the *car-segmented* sequence with (magenta) and without (blue) compaction. The green line uses the basis $\boldsymbol{L}$ found without compaction, but computes final reconstruction coefficients $\boldsymbol{R}$ using the added constraints from the estimated compaction. The added constraints improve performance slightly, but the poor quality of the basis is apparent. Of particular interest is the generalization error at a generalization distance of zero. The no-compaction basis does not span the correct motion space and so cannot accurately reconstruct the *observed* entries when the extra constraints are included. The black line uses the basis estimated with compaction, but recomputes the final reconstruction coefficients in $\boldsymbol{R}$ without using the extra constraints implied by the inferred data association. The excellent generalization performance demonstrates that the basis alone provides sufficient regularization to make the additional constraints virtually redundant.

the error for the full algorithm, while the blue line is the error with no compaction performed. The remaining two lines correspond to hybrid versions. The green line shows generalization error when the basis $\boldsymbol{L}$ is taken from the no compaction results, but the reconstruction coefficients $\boldsymbol{R}$ are replaced by ones computed with the geometric consistency constraints from the estimated compaction applied. The black line is the one we are most interested in for the larger video motion application; it shows generalization error when using $\boldsymbol{L}$ computed with compaction but replacing

$\boldsymbol{R}$ with reconstruction coefficients computed using only the observed entries of the original track, without the added constraints from the associated track fragments.

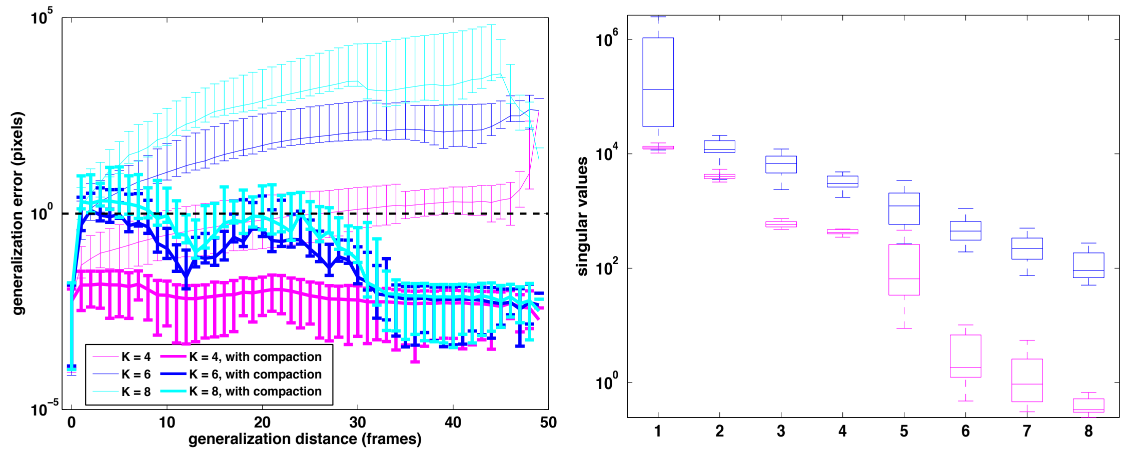
Generalization performance is nearly as good in this last case as it is with the constraints applied (compare black vs. magenta), showing that the basis we find spans the correct motion space and provides sufficient regularization to make it possible to generalize to full tracks from the individual track fragments. This experiment validates our approach to initializing dense video motion (described in detail in Section 5.1), which consists of extrapolating to full-length paths from short tracks built from concatenated optical flow vectors.

4.4.3 Performance with incorrect rank estimation

Finally, we test our performance in situations where the rank of the true measurement matrix is unknown and may be overestimated. We shatter synthetic tracks into five track fragments, putting the first, third, and last fragment into individual columns of a measurement matrix and discarding the remaining two. Each original track is associated with a random patch from an image. We corrupt both the shattered measurement matrix and the local image patches with Gaussian noise. Figure 4.8 shows the generalization performance over 20 random datasets, both with and without compaction. Adding compaction makes the reconstruction robust to an estimated rank that is double the true rank. It also allows for recovery of the correct rank through inspection of the singular values of the reconstructed matrix.

4.5 Implementation for Video Motion Estimation

We make two modifications to the basic compaction algorithm when it runs as part of the full video motion estimation system. First, recall that feature snapping works to ensure that two tracks that will be compacted follow *exactly* the same world point. However, feature snapping places the new track at the best possible location



(a) (b)

FIGURE 4.8: Compaction adds robustness to overestimation of the rank of the true matrix. We create 20 random track matrices with a true rank of 4 and run our algorithm with an estimated rank of 4 (magenta), 6 (blue), and 8 (cyan). The average motion between frames is approximately 2 pixels and the average noise added to the tracks is 0.01 pixels. In (a), we plot the median of $GE(t)$ for each generalization distance over the 20 samples; error bars show the 25th and 75th percentile. Without compaction, the missing entries cannot be recovered when an incorrect rank estimate is provided. Compaction recovers the ground truth associations in every instance, and reconstructs the full matrix with errors on the order of the added noise when given the correct rank. Even when the rank is overestimated by a factor of two, most entries can be recovered with sub-pixel accuracy. The decrease in error after a generalization distance of 30 frames is due to successful compaction. These entries are observed but not correctly associated in the original matrix. Most entries with a generalization distance less than 30 are truly unobserved. In (b), we show the non-zero singular values of the reconstructed matrix when the rank is 8, without (blue) and with (magenta) compaction. Boxes contain the 25th-75th percentiles over the 20 samples. Based on the true rank, the last 4 singular values shown should be zero. While they are still non-zero with compaction, they are consistently and significantly lower than the first four, making it possible to correct the overestimate of the rank.

given the history of one track and *the first frame* of the new track. Once tracking is complete, we can achieve more robust alignment by comparing the average patches of the two tracks, thereby decreasing the effects of noise in the single frame. When collapsing track j into track i, we allow it to shift by the translation that best aligns the two average patches. The shift for each track pair is already computed while building the photometric consistency matrix.

The second modification addresses a limitation of our compaction technique. The factorization procedure we use is very sensitive to outliers, so it is important to ensure that there are no incorrect associations corrupting the motion space during the final factorization. Our technique can generate incorrect associations in sequences with many missing observations and repetitive structures. For example, in the *car-full* sequence, tracks on the front wheel of the car at the very end of the sequence are matched to earlier tracks on the back wheel of the car (see Figure 4.9). For challenging sequences, we can run compaction in an interactive mode, where a human verifies proposed associations and can mark pairs of tracks as being different world points that should never be compacted. This negative feedback is easy to provide; it is, for instance, much simpler than manually specifying correct associations. For the hardest sequences, it can be helpful to manually specify true associations as well instead of repeatedly flagging incorrect associations. However, we stress that the initial tracking and the computation of the basis from associated tracks is always done without human input.

User input is also required to select the rank of the matrix. When the rank is too low, the initial factorization step will fail to reconstruct the observed entries of the matrix with sufficient accuracy. Numerous incorrect associations can indicate that the selected rank is too large. As discussed, we can often correct for slight overestimates of the rank by inspecting the singular values of the final reconstructed matrix.

In a final step, we remove the contribution of the constant basis functions by shifting each reconstructed track so that it passes through $(0, 0)$ in the first frame. A path basis of size K is formed by scaling the first K left singular vectors of the shifted matrix, as described in Section 3.3.

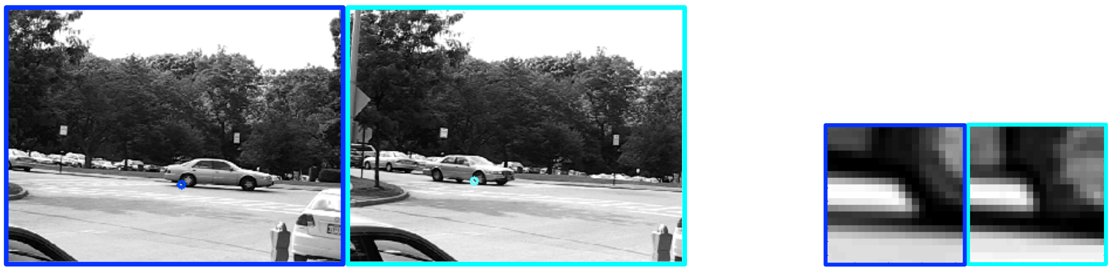


FIGURE 4.9: Example of an incorrect association that can be rejected with an interactive system. The two large images show the location of two track fragments from *car-full* in the first frame in which they are tracked. We also show the corresponding patches used to compute the photometric consistency between this pair of tracks. Although the tracks are on different wheels, the local patches appear nearly identical, causing problems for compaction. A human supervisor can quickly review the results and disallow this association.

5

Extracting Paths from Video

Once we have defined a path basis according to the procedure described in Chapter 4, we are ready to extract complete video motion paths for a new sequence according to the model defined in Chapter 3. Given a selection of anchor points, the remaining unknowns are a mix of continuous and discrete variables, and the path objective function is non-convex. Finding the best anchor points is a difficult problem in its own right, as we hope to densely sample each visible surface *before* we know the correspondences between surfaces visible in different frames of the video.

We proceed by considering the discrete visibility flags separately from the continuous path coefficients, using a local search in the continuous space that finds a local optimum of the objective function. Initial path estimates consisting of the anchor points and path coefficients are found through an analysis of two-frame optical flow fields. For any set of paths, we can find the optimal visibility flags by minimizing the MRF energy

$$E_{\mathrm{MRF}} = \sum_{p}\sum_{t} D(\nu_p(t)) + \sum_{q}\sum_{t} V_T(\nu_q(t), \nu_q(t+1)) + \sum_{pq}\sum_{t} V_S(\nu_p(t), \nu_q(t)) \ . \quad (5.1)$$

From our initial estimate, we minimize the path objective function

$$E(\boldsymbol{c}_1, \ldots, \boldsymbol{c}_P) = \sum_{p \in \mathcal{P}} \underbrace{\sum_{t=1}^{F} \nu_p(t) \rho(\Delta I_p(t))}_{E_D(\boldsymbol{c}_p)} + \frac{\lambda}{2} \sum_{p \in \mathcal{P}} \sum_{q \in \mathcal{P}} \underbrace{\alpha_{pq} \sum_{k=1}^{K} \rho(c_{pk} - c_{qk})}_{E_S(\boldsymbol{c}_p, \boldsymbol{c}_q)} \tag{5.2}$$

with respect to the path coefficients, updating the visibility estimates after a fixed number of descent steps. We add new paths to fill unexplained regions if necessary, converging to a solution that covers every pixel in the video sequence with a visible path. (Equations (5.1) and (5.2) are identical to equations (3.23) and (3.5), respectively. We restate them here for the reader's convenience.)

5.1 Initialization

Initial paths are defined by their anchor points and coefficients. We want the number of paths we extract to be of the same order as the number of visible points in the sequence, where a "visible point" covers more or less the extent of a pixel. Placing anchor points at every pixel in the first and last frames in the sequence at worst overestimates the true number of anchor points needed by a factor of two. However, we also need to account for portions of the visible scene that happen to be occluded in these two particular frames. We search for regions not known to correspond to scene points covered by the first- or last-frame anchors by following a procedure inspired by Sundaram et al. (2010).

We use the method in Brox et al. (2009) to solve for bi-directional pairwise optical flow fields. We concatenate optical flow vectors into tracks, breaking the tracks when the optical flow fields fail a forward-backward consistency check or when the tracks pass too close to a motion boundary (according to equations (5) and (6) from Sundaram et al. (2010), respectively). To prevent merging foreground and background tracks, we create a thin empty buffer around the regions where tracks terminate. We initialize a new track whenever we find a pixel that is at a distance greater than one pixel from any track propagated from the previous frame, including at pixels with low texture that would be discarded by Sundaram et al. (2010). The track fragments that start in the first frame of the sequence are converted into paths, with anchor points in the first frame. If the track fragments are long enough, the coefficients for the paths are determined by projecting the fragments onto the path basis. Specifically, consider the track fragment that starts at $\boldsymbol{u} = (u, v)$ in the first frame, passes through points (x_f, y_f), and is terminated in frame t. The anchor point is $(\boldsymbol{u}, 1)$. We form the $2t \times 1$ vector $\boldsymbol{m}$ and $2t \times K$ matrix $\boldsymbol{A}$ as

$$\boldsymbol{m} = \begin{bmatrix} x_1 - u \\ x_2 - u \\ \vdots \\ x_t - u \\ y_1 - v \\ \vdots \\ y_t - v \end{bmatrix} \quad \text{and} \quad \boldsymbol{A} = \begin{bmatrix} \boldsymbol{\varphi}_1^x(1) - \boldsymbol{\varphi}_1^x(1) & \cdots & \boldsymbol{\varphi}_K^x(1) - \boldsymbol{\varphi}_K^x(1) \\ \vdots & \ddots & \vdots \\ \boldsymbol{\varphi}_1^x(t) - \boldsymbol{\varphi}_1^x(1) & \cdots & \boldsymbol{\varphi}_K^x(t) - \boldsymbol{\varphi}_K^x(1) \\ \boldsymbol{\varphi}_1^y(1) - \boldsymbol{\varphi}_1^y(1) & \cdots & \boldsymbol{\varphi}_K^y(1) - \boldsymbol{\varphi}_K^y(1) \\ \vdots & \ddots & \vdots \\ \boldsymbol{\varphi}_1^y(t) - \boldsymbol{\varphi}_1^y(1) & \cdots & \boldsymbol{\varphi}_K^y(t) - \boldsymbol{\varphi}_K^y(1) \end{bmatrix}. \tag{5.3}$$

The notation $\boldsymbol{\varphi}_k^x(f)$ represents the x-coordinate of the kth basis path, evaluated in frame f; $\boldsymbol{\varphi}_k^y(f)$ is the y-coordinate of the basis path in that frame. The initial coefficients are the solutions of $\boldsymbol{Ac} = \boldsymbol{m}$, as long as $2t > K$ so that $\boldsymbol{A}$ is full rank. If the track fragment is too short and $\boldsymbol{A}$ is rank deficient, we select coefficients by testing the coefficients from other nearby track fragments. We pick the coefficients that create the path with the best brightness constancy measured over a few frames.

Track fragments that reach the last frame of the sequence are converted into paths anchored at points in the last frame using the same procedure. For all other track fragments, we place a (possible) anchor point in the frame in which they are initialized and construct initial coefficients using the same technique as before. We iterate through these potential paths and only add those that differ from already included paths by more than an average of 2 pixels per frame. Figure 5.1a shows the anchor points of the initial paths for one of our test sequences. Anchor points are drawn in color on top of the frames of the sequence; colors correspond to the value of c_{p1}, the coefficient of the first basis path.

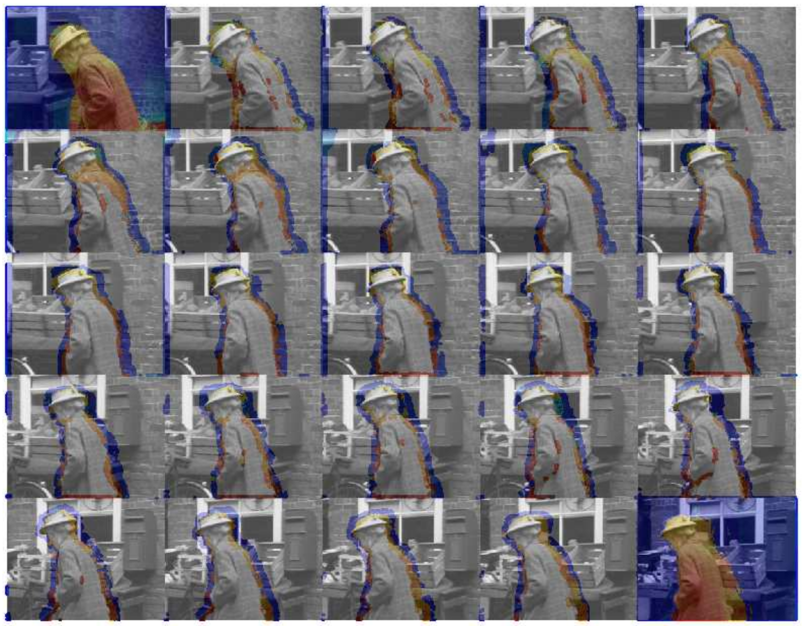
The anchor points are generally fixed during optimization, although they can be deleted and new ones can be added to ensure complete coverage of the video sequence. We describe this process in Section 5.3. Optimization primarily changes the values of the coefficients for the paths emanating from the anchor points. Figure 5.1b shows the color-coded anchor points for the test sequence after optimization is complete.

5.2 Optimization

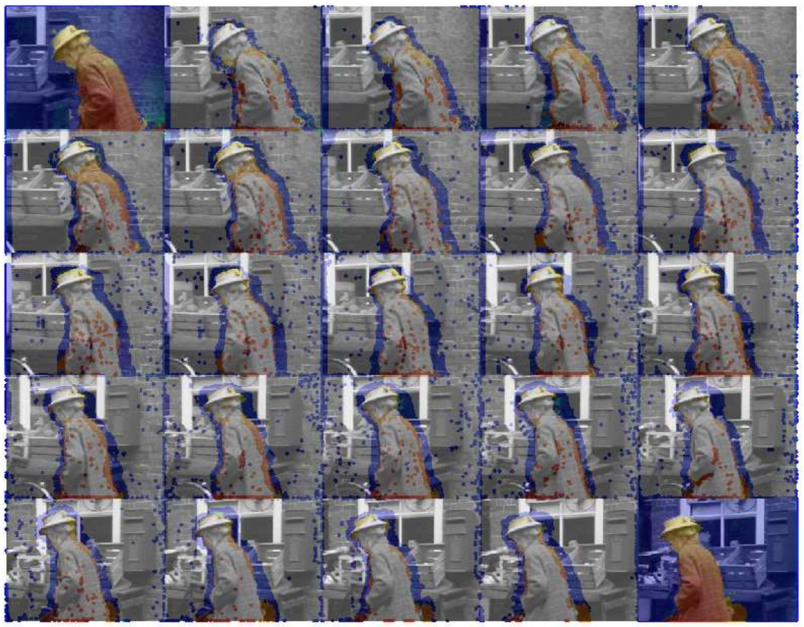
Starting with the initial path estimates, we interleave two steps: a combinatorial optimization step finds visibility flags $\nu_p(t)$ for the current path estimates, and a continuous optimization step updates path coefficients $\boldsymbol{c}_p$ given the current visibility estimates. The initial path estimates are often poor along occlusion boundaries because visibility was not properly accounted for during initialization. We use a heuristic procedure to take large steps in the search space, replacing the parameters of a path with the parameters from a nearby path if this replacement reduces the overall data cost.

5.2.1 Estimating visibility for current paths

We can find the best values for the visibility flags for a set of paths by minimizing the MRF energy function using graph cuts (Boykov and Kolmogorov, 2004; Kolmogorov and Zabih, 2004). However, we must first compute the spatial neighbors for each MRF node and the observed visibility flags $\hat{\nu}_p(t)$. We find the controlling path at each video pixel in $O(PF)$ time for P paths over F frames by storing the path with the best brightness constancy seen at each pixel. We walk along each path, computing its patch brightness constancy at each frame and replacing the stored value at the



(a) At initialization.



(b) At convergence.

FIGURE 5.1: Anchor points selected during initialization and at convergence. Anchor points are drawn in false color on the frames of the sequence (time increases right-to-left, top-to-bottom). Colors are chosen according to the value of the first path coefficient; similar colors (in the jet colormap) denote similar sets of path coefficients. Note the improved segmentation of Miss Marple after convergence.

pixels it passes through if necessary. On a second pass, we check whether the current path is the controlling path at the nearest pixel to $\boldsymbol{x}_p(t)$ in frame t.

Actually computing $\hat{\nu}_p(t)$ for all p and t takes $O(PF^2)$ time because we need to compute the average distance between the path p and the controlling path p^* when the two are not equal. This takes an extra F operations for every new pair, and the controlling path may change at every time step along the current path. In practice, we can save a lot of computation by caching this value so that we do not have to recompute the distance when the same pair appears in different frames.

It is possible to compute the spatial neighbors by storing the set

$$Q(x, y, t) = \{q \mid ||\boldsymbol{x}_q(t) - (x, y)|| < \Delta\} \tag{5.4}$$

while finding the controlling path at each video pixel (x, y, t). If (x, y) is the nearest video pixel to $\boldsymbol{x}_p(t)$, then all neighbors of p are listed in $Q(x', y', t)$ for $|x - x'| \leqslant 1$ and $|y - y'| \leqslant 1$.

Remember that we enforce the constraint that controlling paths are visible in the final solution. We do this by eliminating these nodes from the MRF, incorporating the smoothness terms for the incident edges into the data terms for the remaining vertices. We also eliminate the nodes for all frames and paths where $\boldsymbol{x}_p(t)$ is outside the field of view of the camera because we know these must have $\nu_p(t) = 0$.

The strength of the connection between two spatial neighbors is determined by the weight $w_{pq}(t)$ defined in equation (3.21). This weight decreases rapidly as the geometric and photometric distances between the two paths increase. For very large problems, we can limit the memory requirements by setting a threshold on the weights and only including edges with large enough $w_{pq}(t)$. We found that only including an edge when $w_{pq}(t) > 0.01$ eliminates many edges without changing the optimal solution significantly. For example, on the *marple7* sequence (our running example), we discard nearly one third of the edges in the MRF, reducing computation time from 310 seconds to 90 seconds. The optimal solution in the reduced MRF differs in fewer than 0.1% of the unknown visibility flags.

5.2.2 Non-linear minimization to improve paths

With visibility estimates fixed, we update path coefficients by minimizing the energy function (3.5) via trust-region Newton Conjugate Gradients optimization (Nocedal and Wright, 2006). The pseudocode for the optimization scheme is included in Appendix A. The idea is to find each descent step by solving the Newton step approximately using conjugate gradients (CG) on each iteration. This requires computing vectors of the form $\boldsymbol{Hv}$, where $\boldsymbol{H}$ is the Hessian, but it does not require computing (or inverting) $\boldsymbol{H}$ itself. The Hessian of our objective function is very large but sparse. The sparsity pattern changes over time because the coupling coefficients α_{pq} depend on the path coefficients. When computing successive conjugate gradients, we treat the terms α_{pq} as constants – a good approximation for small path perturbations – and recompute them between full descent steps. The derivations of the gradient and the Hessian are included in Appendix A.3.

5.2.3 Heuristic update

After a fixed number of descent steps, we allow a path to copy its coefficients and visibility flags from one of its neighbors if doing so improves the path's fit to data. Specifically, path p copies from path q if:

- q is visible in the anchor frame for p: $\nu_q(\tau_p) = 1$,
- q is anchored in a different frame: $\tau_p \neq \tau_q$,
- q is a neighbor of p in frame τ_p: $||\boldsymbol{x}_q(\tau_p) - \boldsymbol{u}_p|| < \Delta$,
- q is visible for enough frames: $\sum_t \nu_q(t) \geqslant \frac{F}{2}$, and

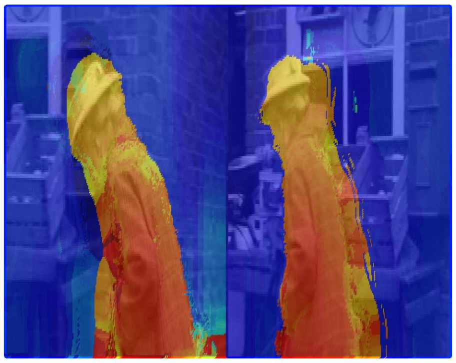
- copying parameters improves the data fit for p the most out of all possible neighbors.

Replacing the parameters of the path always decreases the data term, but it is not guaranteed to be a descent step in the full objective function because the smoothness term is ignored. However, the large steps this procedure permits improve the final solution by letting the optimization routine jump out of bad local optima. Figure 5.2 illustrates the benefits of this step on our test sequence. Without the heuristic update, the solution is stuck in an undesirable local optimum, where the incorrect initial estimates of the motion of the background persist in the final solution. The background points are initialized with the foreground motion because they are occluded within just a few frames of their anchor frame. With the heuristic update, these wrong initial estimates have been replaced by correct estimates that propagated from paths of similar points anchored in frames farther away from the occlusion.

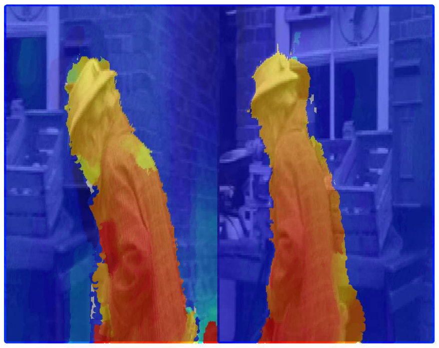
5.2.4 Anchor management

The most efficient description of video motion would use a single anchor point for each unique world point visible in the scene. According to this metric, our selection of anchor points is suboptimal. We start out with multiple anchor points for each point. For example, even completely static surfaces that are never occluded will be associated with two paths per point: one anchored in the first frame and one anchored in the last frame of the sequence. However, this slight oversampling is actually beneficial in our approach because it allows us to quickly correct from poor initial estimates as described in Section 5.2.3. We have not addressed the problem of merging paths to eliminate unnecessary duplicates.

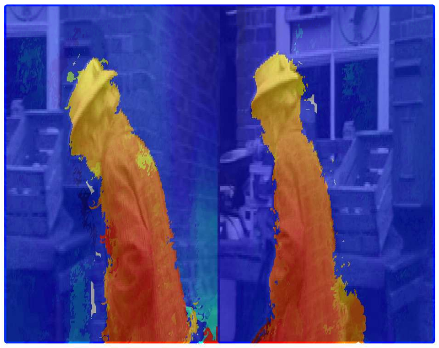
We also have not considered the problem of selecting the optimal anchor frame for each path. The anchor frame determines the reference appearance of each path, which plays a unique role in the objective function. It may be possible to reduce the



(a)



(b)



(c)

FIGURE 5.2: Effect of the heuristic update step. Motion estimates are shown for paths anchored in the first and last frames only, using the same color scheme as in Figure 5.1. In (a), we show the initial path estimates derived from the concatenated optical flow vectors. Note the significant errors in regions affected by occlusions (directly in front of Miss Marple in frame 1 and behind her in frame 25). The other two image pairs show the solution at convergence with the heuristic update either ignored or included. Without the heuristic update (b), the solution cannot escape the poor local optimum nearest to this incorrect initialization. The heuristic update (c) compensates for the poor initialization, achieving a much better final solution.

objective function without actually changing any path by simply sliding the anchor points along their respective paths to the frame that minimizes $E_D(\boldsymbol{c}_p)$. However, predicting the full effect of this change is complicated because the anchor frames are also used to define neighbors for the smoothness term. The anchor frames also play an important role when computing the controlling paths because patch brightness constancy (3.11) includes a term measuring total image distance from a patch centered on the point in its anchor frame. Selecting an anchor frame that is as far as possible from any occlusion events could be beneficial because it would decrease the chance that the reference patch includes portions of an occluding object. We have not fully investigated what impact allowing paths to change anchor frames would have on the overall convergence.

5.3 Termination

After one round of Newton-CG descent steps and before we recompute visibility, we check for convergence by computing the maximum change in any path in any frame in which it is visible. Convergence occurs when the maximum change from the solution at step zero of this minimization round is less than one pixel.

We then test the solution for adequate coverage of the video. We first eliminate paths that are within the field of view of the camera for multiple frames but are only marked as visible in their anchor frame. These one-point paths occur when visibility estimation correctly identifies an outlier with an incorrect path estimate. Next, we compute the distance from each video pixel to the nearest visible path in its frame. An unexplained pixel has a distance of greater than one pixel to the nearest visible path. If fewer than 0.5% of the pixels are unexplained, we accept the solution and terminate optimization. If too many unexplained pixels are detected, we add paths by creating new anchor points at every unexplained pixel and resume optimization. In principle, we could continue optimization as long as *any* unexplained pixels are found. Eventually we would terminate after, in the worst case, adding an anchor point at every pixel. We selected our chosen threshold to balance coverage with computational burden.

The parameters for newly created paths are set by interpolating from the existing paths. We implemented the simplest possible nearest-neighbor interpolation where the path with anchor point $(\boldsymbol{u}_{\text{new}}, \tau_{\text{new}})$ copies its initial parameters from the closest visible path $q = \arg\min_{p \mid \nu_p(\tau_{\text{new}})=1} ||\boldsymbol{x}_p(\tau_{\text{new}}) - \boldsymbol{u}_{\text{new}}||$. Another option would be to average over multiple nearby neighbors, weighted by the similarity of the appearance of the various points, as is commonly done to fill in occluded regions in optical flow computation. It is only necessary to initialize the path coefficients, as the optimal visibility flags can be recomputed quickly once the coefficients are known.

6

Evaluation

The experiments in this chapter validate our approach to computing video motion. Our experiments demonstrate that we can successfully recover sequence-length correspondences for all visible points, with world points correctly associated before and after identified occlusion intervals.

6.1 Test Sequences

Although there exist extensive benchmark datasets for the problem of optical flow estimation, these sequences are not well suited for our problem. The Middlebury sequences (Baker et al., 2011) are too short and lack sufficient occlusions. Ground truth optical flow is available between a single pair of frames per sequence, but no information is provided about the motion in other frames. The MPI-Sintel sequences (Butler et al., 2012) are long enough and include very challenging, realistic motions. In fact, the sequences are so challenging that current frame-to-frame trackers struggle to maintain tracks, limiting our ability to find an appropriate path basis. Current trackers allow us to recover bases for the motion of the backgrounds, but they fail to track points on the foreground objects for more than a few frames. A benefit of using synthetic sequences like those in MPI-Sintel is the availability of ground truth. However, even though MPI-Sintel provides ground truth optical flow between each consecutive pair of frames, the information to connect points across occlusions is not provided. This lack of long-term information is unfortunate, as the ability to maintain correspondence across occlusions is one of the key features we wish to test.

Both Sand and Teller (2008) and Sundaram et al. (2010) evaluated their techniques on a set of contrived sequences formed by temporally mirroring short video clips so that frame $F - f + 1$ is a copy of frame f. The length of the sequences ranges from 50 to 70 frames (corresponding to 25 to 35 unique frames). The purpose of mirroring the sequences is so that performance can be measured by comparing the starting and ending positions of extracted point trajectories that extend throughout the entire sequence. This metric does not necessarily accurately represent the quality of the solution. Consider an algorithm that always returns zero motion with no occlusions. This algorithm would be unbeatable with respect to the "return-to-start" metric but could not be called a successful video motion algorithm. Furthermore, recall that both algorithms terminate trajectories at suspected occlusion boundaries, meaning that the most difficult points to track are never evaluated. In fact, the criteria for labeling a point as occluded in Sand and Teller (2008) is a fixed threshold on a smoothed version of the particle's energy, causing all potential tracking failures to be removed from the set of full trajectories. The temporally mirrored motion is itself a problem because it is in no way representative of real motions one would observe.[1]

We test our approach on five real sequences with increasingly complex motion. Each sequence was selected because it contains significant but temporary occlusion of a textured background, allowing us to test the novel components of our algorithm

[1] Peter Sand acknowledges the limitations of his experiment. His dissertation describes his evaluation as "for descriptive purposes only" and recommends that the reported results "not be used to compare the algorithm with future particle video algorithms." (Sand, 2006)

by measuring how well we maintain correspondences across the occlusion interval. Selected frames from each sequence are shown in Figure 6.1.

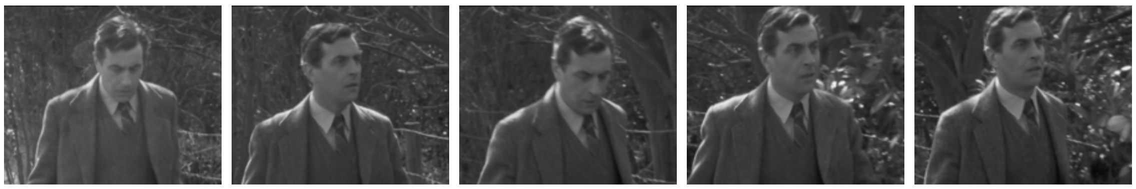
The first sequence we include is the popular MPEG *flowerbed* sequence, containing a total of 29 frames. This sequence was used in Wang and Adelson (1994). The rigid motion in the sequence is the result of a panning camera imaging a static scene. A foreground tree occludes a significant portion of the background, with the average occlusion lasting approximately six frames. The second sequence, called *truck*, is a new sequence collected for this work. It captures a truck driving behind a bus stop sign, viewed from an almost stationary camera. This sequence contains 33 frames; the start and end frame were selected so that a significant portion of the truck is occluded in both frames. The bus stop sign occludes portions of the truck for an average of 12 frames. The truck occludes much of the background, but these occluded regions do not reappear before the end of the sequence.

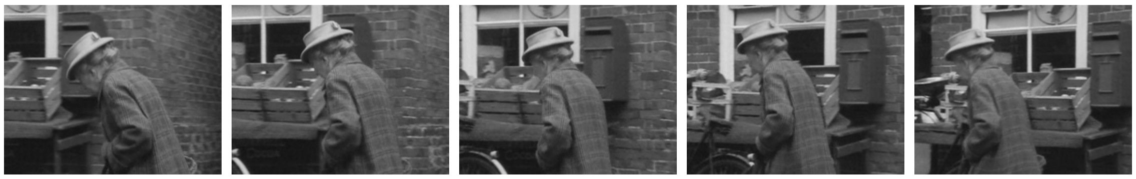
The remaining three sequences are clips from the Berkeley motion segmentation dataset (Brox and Malik, 2010) containing significant occlusions caused by a non-rigid foreground object (a person). We use 25 frames from *marple7*, 60 frames from *marple1*, and 72 frames from *marple8*. We have been using the marple7 sequence as our running example. Miss Marple occludes regions of the background for an average of ten frames. The two longest sequences are significantly more difficult than the other three. Marple1 contains many shadows that violate the brightness constancy assumption; marple8 includes a moving object in the background that is almost entirely occluded by Miss Marple in frame 32. Background points are occluded for approximately 14 frames.

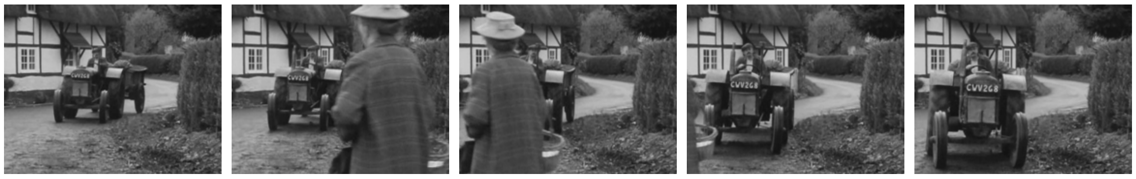
We also include experiments on the 60-frame synthetic *flag* sequence introduced in Garg et al. (2011). This sequence was created by deforming a textured surface according to motion-capture data from a flag waving in the wind. Because the sequence is synthetic, ground-truth, full-length paths are available starting at every

(a) Flowerbed.

(b) Truck.


(c) Marple1.


(d) Marple7.



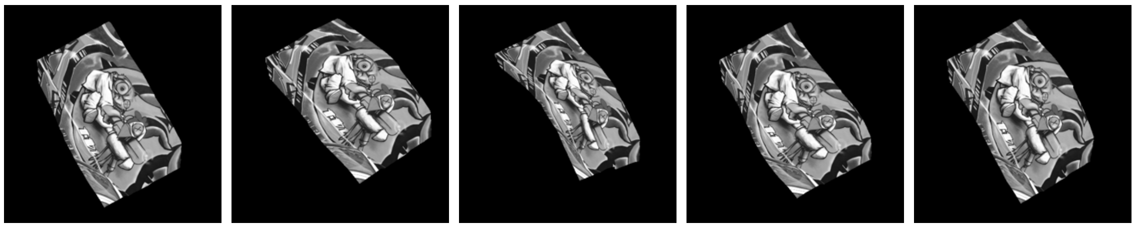
(e) Marple8.

FIGURE 6.1: Selected frames from our test sequences.

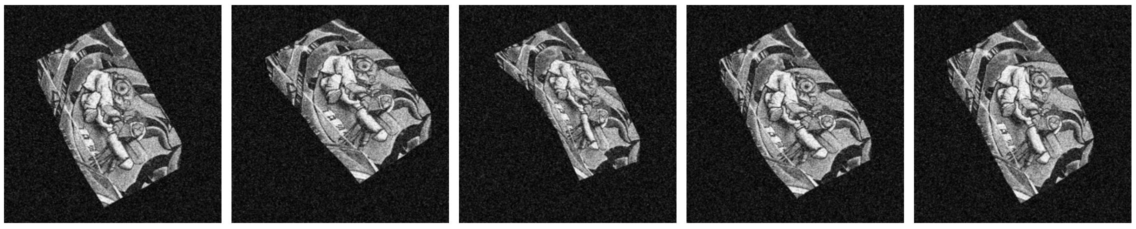
pixel in the first frame. The sequence does not contain any significant occlusions (although a version of the sequence with extremely unrealistic synthetic occlusions is available). Despite the lack of occlusions, the sequence is challenging because the motion is highly non-rigid. As a result, it challenges our assumption that all paths can be represented in a single low-dimensional linear subspace. Figure 6.2 shows selected frames from the original sequence, the versions with added Gaussian or salt-and-pepper noise, and the version with toy occlusions.

We compare our results to trajectories from Sundaram et al. (2010), using the executable provided by the authors. We call these LDOF trajectories.[2] Another natural comparison is to the work from Garg et al. (2010) and Garg et al. (2011). There are published results for the latter method on the flag sequence, so we can compare our method directly on that sequence. Unfortunately, the authors have not provided a public implementation of their algorithm, so a direct comparison on the real sequences is difficult. We use an earlier version of our work (Ricco and Tomasi, 2012a) as a proxy for these sequences. Our earlier method was developed in parallel with and shares many of the features of the approaches from Garg et al. We call the results of our earlier method Lagrangian Motion Estimation (LME) paths. LME paths are computed using a variational framework that finds paths for the visible pixels in both the first and last frames of a sequence. Similarly, Garg et al. compute motion for points visible in a single selected reference frame. LME paths include occlusion detection, while the other methods do not. However, the LME occlusion model is less sophisticated than the one we use for video motion paths and can miss occlusions caused by objects not visible in the first or last frame. When occlusions are caused by these objects, the method functions similarly to one that does not explicitly detect occlusions. Although the methods from Garg et al. have not been

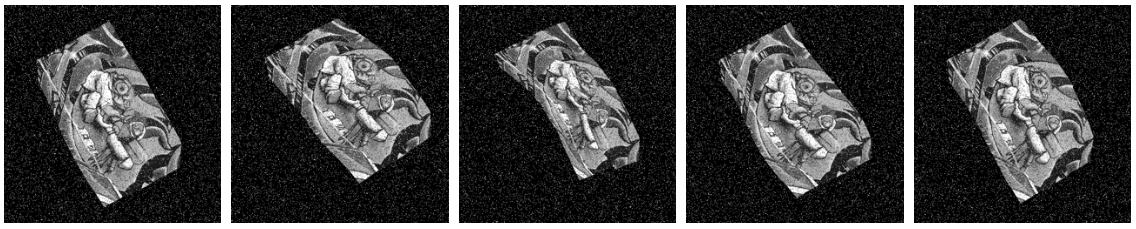
[2] The results from Sundaram et al. (2010) are *trajectories* not *paths* because they do not extend through the full length of the sequence but instead terminate at detected occlusions.



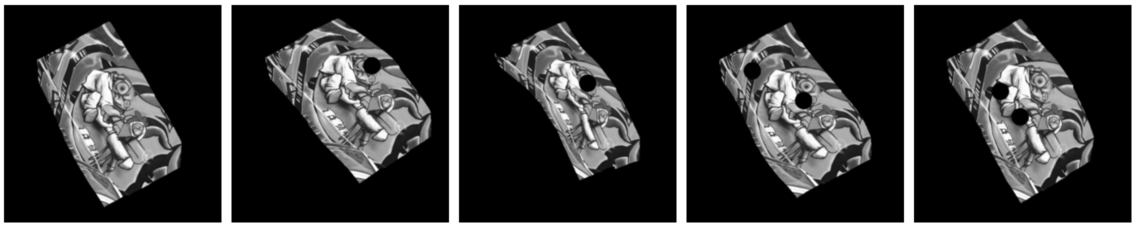
(a) original



(b) Gaussian noise



(c) salt-and-pepper noise



(d) occlusions

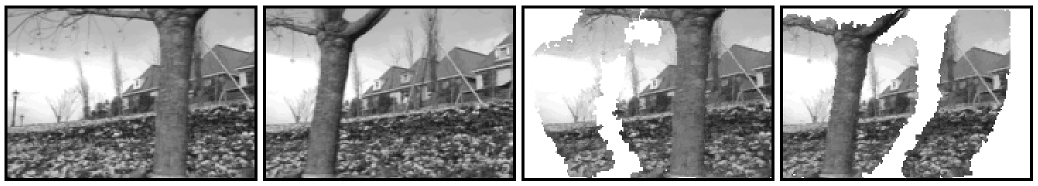
Figure 6.2: Selected frames from the flag sequence. Four versions are available: original (noise-free), added Gaussian noise, added salt-and-pepper noise, and noise-free with toy occlusions.

tested on sequences with significant occlusions, we believe that LME paths represent a reasonable approximation of the results one might expect.

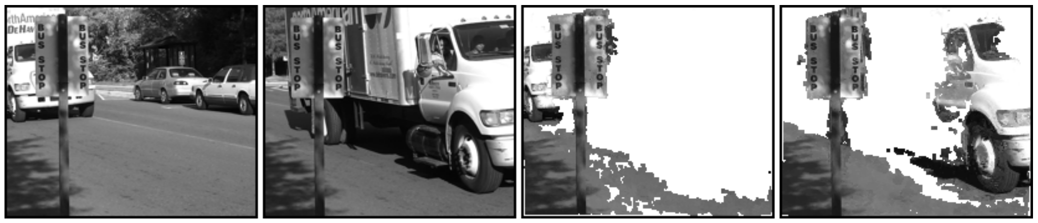
We tried constructing paths by computing optical flow independently between all pairs of frames, but the results were so poor that they are not worth including in detail. We do not compare against Particle Video trajectories from Sand and Teller (2008) directly because LDOF trajectories have been shown to be significantly and consistently more accurate.

6.2 Qualitative Evaluation

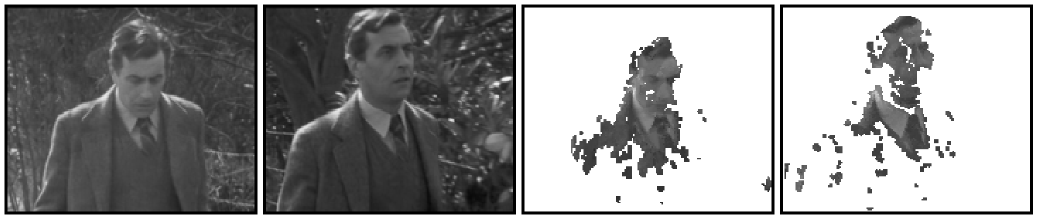
For a qualitative evaluation, we use computed motion to warp all frames of a sequence to align with a selected reference frame. For example, to warp all frames back to the first frame, we first find the closest path to every pixel in that frame. This path defines the location of the corresponding point in the second frame, the third, and so on. Let us assume that the selected path through pixel (i, j) has location (x, y) in the second frame. We render the second frame of the warped sequence by coloring the pixel at (i, j) with the intensity at (x, y) (computed through bilinear interpolation). When the closest path to pixel (i, j) is marked as occluded in the frame we are rendering, we do not use the appearance of the pixel at (x, y) because that would be the appearance of the occluding object. Instead, we search for another path that is within half a pixel of (i, j) and visible in both the reference frame and the frame to render. If no such path exists, we paint the pixel (i, j) white to mark it as occluded. (To follow this procedure with motion computed via LDOF trajectories, missing correspondences are treated as detected occlusions.) This process is repeated for all pixels in all frames to create a full-length warped sequence. The result is a motion-compensated video that should appear static except for the white regions replacing occluded pixels.



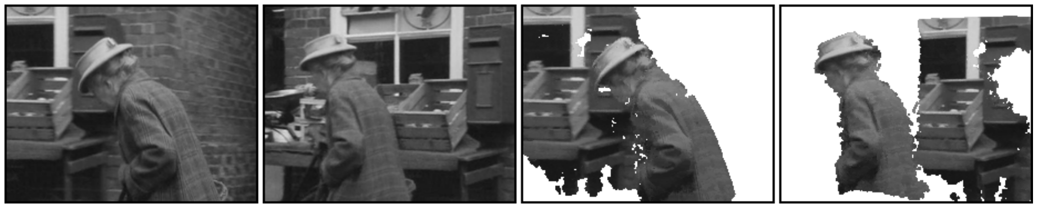
(a) Flowerbed. Two basis functions.



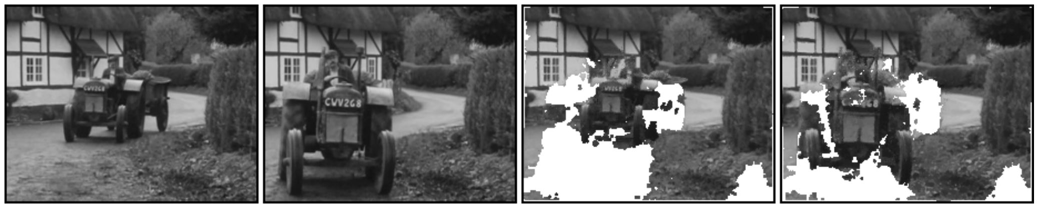
(b) Truck. Four basis functions.



(c) Marple1. Eight basis functions. Shadows cause problems for occlusion detection, so not all of the man is recognized as visible in both frames.

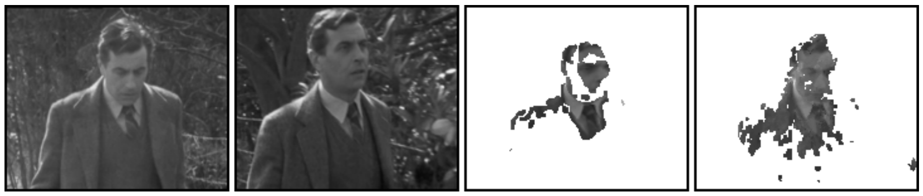


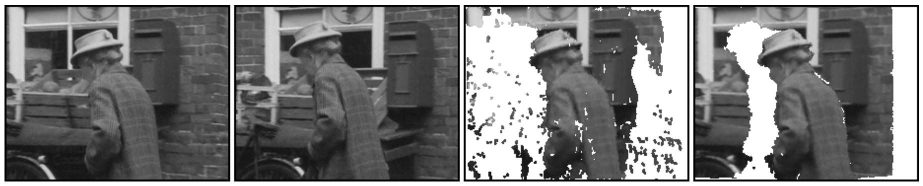
(d) Marple7. Five basis functions.



(e) Marple8. Eight basis functions. Miss Marple (off-screen in both the first and last frames) walks past during the sequence, completely occluding the entire scene. Intermediate frames with Miss Marple visible can be seen in Figure 6.1e.

FIGURE 6.3: Results of our method. For each sequence, we show the first and last frames, followed by the last frame warped to align with the first frame, and vice versa. Regions detected as occluded in the source frame of the warp are marked in white.


(a) Warp of last frame to first frame for marple1.


(b) Warp from frame 18 to frame 13 of marple7.

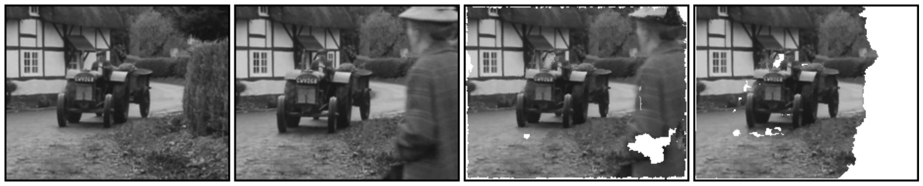

(c) Warp from frame 13 to frame 1 of marple8.

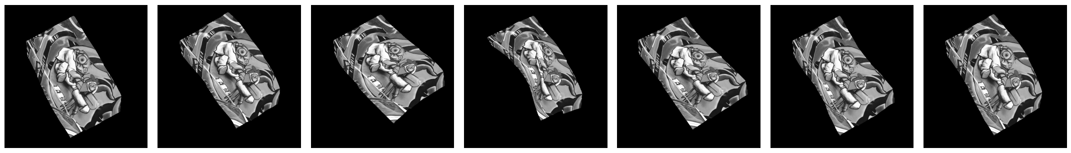
FIGURE 6.4: Examples of mistakes made by alternative methods. Results from video motion paths (rightmost column) are consistently higher quality. The left two columns show the source and target frames of the warp. Correct warps should match the first image in each row. The third column shows the inferior result from a competing method. In (a), few correspondences survive with LDOF trajectories. Surviving trajectories suffer from drift. In (b), LME paths rely on reference frames, so they lack correspondences for the portions of the scene that are occluded in the first and last frames. In (c), LME paths do not track Miss Marple because she is off-screen in the first and last frame. As a result, this method cannot reliably detect the occlusion she causes. Video motion paths track every visible point and correctly mark the background as occluded.

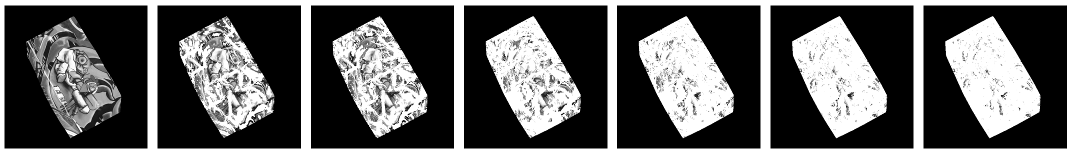
We cannot include the complete videos here. However, Figure 6.3 shows the last frame of the five real sequences aligned to the first frame, as well as the first aligned to the last, using the computed video motion paths. Figure 6.4 shows examples of common mistakes made by the competing algorithms. LDOF trajectories are unable to bridge occlusions, resulting in very few correspondences between distant frames. They also suffer from drift, which manifests in the misalignment of the man's face in

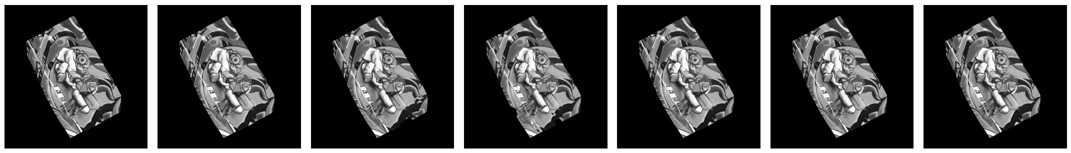
Figure 6.4a. The LME paths maintain correspondences across occlusions, but points that are not visible in the first or the last frame are not tracked. This limitation causes two types of artifacts. First, these untracked points must be treated as always being occluded (because they have no correspondences defined), resulting in holes in the warped frames (see Figure 6.4b). Also, because the points are not tracked, they cannot cause occlusions, so any points that they do occlude are mistakenly considered visible. These points are rendered in warped frames using the appearance of their occluder, instead of being marked in white (see Figure 6.4c). Video motion paths give better results. Correspondences are correctly maintained across occlusions and the detected occlusion regions are plausible.

Figure 6.5 shows the result of warping every tenth frame of the flag sequence back to the first frame. We also show the corresponding warped images from the LDOF trajectories for this sequence and the warped images from paths created by computing optical flow from the first image to all the rest using LDOF (Brox et al., 2009) without any occlusion detection. Because the absolute displacement from the first frame does not grow too large, first-to-rest optical flow did not suffer from the extreme errors we observed on the real sequences.

The complexity of the motion in this sequence required 30 path basis functions; the largest basis used for any of the first five sequences contained eight path basis functions. The sequence is also much higher resolution than our other five sequences. As a result, computing video motion was much slower. For efficiency, we downsampled the images by 1/2 and subsampled the initial anchor points. We recover paths for the full resolution sequence through linear interpolation. LME paths could not be computed for this sequence due to the problem size.


(a) Original frames.


(b) Warps from LDOF trajectories. (White pixels mark missing correspondences.)


(c) Warps from LDOF optical flow between first and rest.

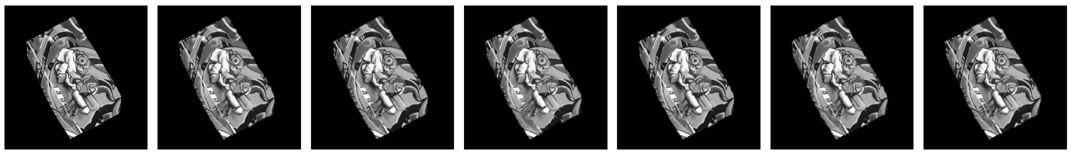

(d) Warps from video motion.

FIGURE 6.5: Estimated motion in the flag sequence. In (a), we show every tenth frame of the sequence. The remaining rows are the corresponding frames warped backward to align with the first frame of the sequence. With perfect motion estimates, every image would be a copy of the leftmost image. Warps are computed using (b) LDOF trajectories, (c) LDOF optical flow computed between frame 1 and frame f for all f, and (d) video motion paths.

6.3 Quantitative Evaluation

The flag sequence contains ground truth paths starting at every pixel in the first frame. For this sequence, we can measure geometric error between the estimated location of each of these points in each frame and the ground truth location. Table 6.1 reports the root mean squared (RMS) error between the predicted and true locations of each point. We compare our results to results from LDOF trajectories, results from LDOF first-to-rest flow, and the published results from Garg et al. (2011). Our method forces all computed paths to lie within a subspace with 30 dimensions while the other methods are nonparametric.

Garg et al. (2011) uses a motion subspace as a prior to regularize the solution. They report results using a full-rank DCT basis or a 75-dimensional basis computed via PCA on tracks from Pizarro and Bartoli (2012). To isolate the effect of errors in our extracted basis from other sources of error, we also ran our method using a 30-dimensional basis constructed from the ground truth. This basis can reconstruct the ground truth paths with an RMS error of 0.4 pixels and a maximum error of 4.8 pixels. To recover the ground truth motion exactly we would need a higher-dimensional representation.

There is no reason we could not use our video motion paths to initialize a non-parametric method that would solve for small deviations from the motion subspace. We would expect this step to improve our results in this sequence with due to its highly non-rigid motion. The focus of our work is on improving this initial solution,

Table 6.1: Root Mean Squared (RMS) pixel error for computed paths on the flag sequence. We report results on all four versions of the sequence: no noise added (orig), Gaussian noise added (GN), salt-and-pepper noise added (SPN), and synthetic occlusions added (occ). Note that the good performance of the LDOF trajectories is somewhat misleading. The results are missing over 70% of the correspondences for the original sequence due to terminated trajectories (see the warps in Figure 6.5 for examples); the coverage for the noisy sequences is even sparser. The method from Garg et al. (2011) uses a full-rank DCT basis (120-dimensional) or a 75-dimensional basis computed via PCA on tracks from Pizarro and Bartoli (2012). Our basis for video motion is computed automatically as described in Chapter 4. The last row reports errors if we are allowed to use the optimal 30-dimensional basis constructed from the ground truth tracks. The bases we use for the version with occlusions include the motion of the synthetic occluder, which Garg et al. (2011) do not track.

Method	**RMS endpoint error**				**99th percentile error**			
	orig.	GN	SPN	occ.	orig.	GN	SPN	occ.
Garg et al. (2011), DCT basis	1.06	2.78	2.29	1.72	6.70	7.92	8.53	5.18
Garg et al. (2011), PCA basis	0.98	2.28	1.84	1.33	3.08	8.33	7.09	4.92
LDOF trajectories	1.29	1.34	1.26	1.33	4.53	4.29	4.20	4.74
LDOF first-to-rest flow	1.71	4.35	5.05	2.01	3.72	18.15	20.35	6.63
Video Motion	1.81	3.08	2.88	3.18	5.96	10.83	9.63	11.09
Video Motion, true PCA basis	1.18	2.25	2.20	2.56	3.81	7.41	7.34	8.44

particularly in sequences with significant occlusions that Garg et al. neglect.

To measure geometric error for the real sequences, we would have to resort to using hand-tracked paths as a surrogate for ground truth. Manual tracking for real sequences is both painstaking and unreliable, particularly for complex motions or low-texture regions.

Instead, we measure the degree to which intensities remain constant along computed paths as a proxy for geometric accuracy. We use the median intensity value along the estimated visible portion of each path as the expected appearance of the corresponding point. Let $\hat{I}_p$ denote this value, computed as

$$\hat{I}_p = \arg\min_a \sum_t \nu_p(t) |I(\boldsymbol{c}_p, t) - a| \ . \tag{6.1}$$

We use a summary statistic instead of the appearance at the path's anchor point so that our metric is invariant to the selection of the anchor frame and only varies if the actual computed motion or visibility changes. We use the median intensity to increase robustness to errors in the visibility estimates. The *all-path interpolation error* (APIE) is the absolute deviation from the expected appearance of the appropriate path, averaged over all visible frames for all paths. Mathematically, this quantity is expressed as

$$\text{APIE} = \frac{\sum_p \sum_t \nu_p(t) |I(\boldsymbol{c}_p, t) - \hat{I}_p|}{\sum_p \sum_t \nu_p(t)} \ . \tag{6.2}$$

When computing this metric (and the ones that follow) for the LDOF trajectories, we treat missing correspondences as if they were marked occlusions.

In using this metric, we assume that the brightness constancy assumption holds. Every realistic sequence will have some violations of brightness constancy, even along the correct paths, so we should not expect an APIE equal to zero. However, in general, lower values for APIE indicate better performance.

TABLE 6.2: Solution quality metrics. APIE measures average intensity constancy along estimated paths (smaller is better, assuming the brightness constancy assumption holds). Path length is the number of frames in which a path is reported as visible (in general, longer is better). Pixel distance measures path density by reporting the distance to the nearest visible path for each pixel. We report the 50th, 95th, and 99th percentiles (smaller is better). Better values for each sequence are highlighted in bold.

Sequence	**Method**	**APIE**	**Path length**		**Pixel distance to closest path**		
			Mean	Std. dev.	50th	95th	99th percentile
Flowerbed	LDOF traj.	4.54	11.2	10.5	0.47	8.5	15.2
	LME paths	3.65	**23.9**	7.3	0.31	0.79	1.3
	Video Motion	**2.59**	23.1	7.4	**0.29**	**0.66**	**0.85**
Truck	LDOF traj.	5.40	6.8	7.5	1.2	47.0	70.0
	LME paths	5.97	**23.4**	7.4	0.39	1.9	4.4
	Video Motion	**3.56**	20.9	9.1	**0.28**	**0.67**	**0.91**
Marple7	LDOF traj.	2.50	6.7	6.4	0.47	6.9	13.3
	LME paths	2.64	**15.9**	7.5	0.43	5.7	9.7
	Video Motion	**2.27**	14.7	6.8	**0.30**	**0.68**	**0.87**
Marple1	LDOF traj.	4.57	9.4	11.4	0.51	14.7	26.7
	LME paths	4.11	**25.9**	19.4	0.62	9.1	18.9
	Video Motion	**2.61**	11.21	14.7	**0.32**	**0.84**	**1.0**
Marple8	LDOF traj.	3.70	14.9	14.7	0.47	7.4	16.4
	LME paths	5.17	**59.9**	15.7	0.35	1.9	6.0
	Video Motion	**2.79**	29.5	25.3	**0.24**	**0.65**	**0.90**

One way to decrease the APIE is to recover many more paths that are flagged as occluded in all but very few frames. Such a solution will tend to have lower APIE simply because the brightness constancy assumption is more likely to hold over short intervals. We want to discourage this type of solution because recovering long-range correspondences between distant frames is one of our driving goals in video motion estimation. We ensure that our solution is not overwhelmed by numerous short tracks by examining the distribution of the visible lengths of computed paths. The visible length of path p is simply $\sum_t \nu_p(t)$. We report the cumulative distribution as well as the mean and standard deviation of the visible lengths of the paths.

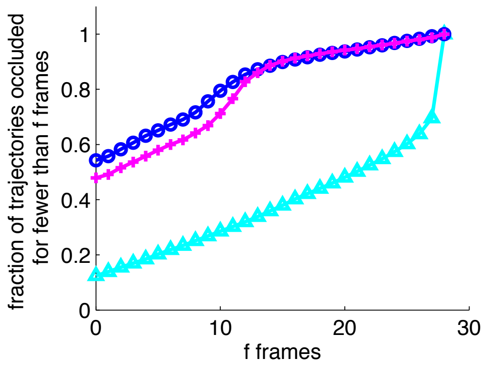
Finally, we measure the density of the computed paths within the sequence. For each video pixel, we compute the distance to the closest visible path in its same frame. We examine the cumulative distribution of this pixel distance measurement and report the 50th, 95th, and 99th percentile.

Our results are summarized in Table 6.2. Figures 6.6 and 6.7 show the cumulative distribution functions for path lengths and pixel distance for each sequence. Video motion paths report high-quality motion estimates for nearly all pixels.

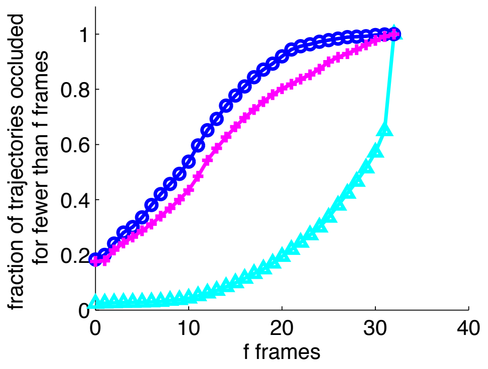
6.4 Sensitivity to Parameters

Our objective functions for both visibility and path optimization include parameters that can be tuned to increase the importance of spatial smoothness relative to brightness constancy. We used the same setting of these parameters for every sequence that we tested. We set $\lambda = 1$ in (3.5), and $\sigma = 50$ in (3.7), with intensity values in the range $[0, 255]$. We tried varying λ in the range $[0.25, 4]$ and achieved similar results for the flowerbed sequence in all cases.

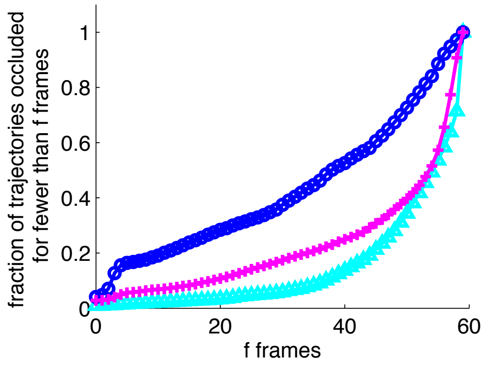
The system is much more sensitive to the parameters in the visibility MRF. We use $\lambda_L = 0.75$, $\lambda_T = 0.5$, and $\lambda_S = 0.25$, and we scale intensity values to $[0, 1]$. These values were tuned manually to achieve the best performance for the flowerbed sequence and were then used for all other sequences. Decreasing λ_L and increasing λ_S usually results in missed occlusions. Figure 6.8 shows an example of the detected occlusions in the flowerbed sequence when we use $\lambda_L = 0.5$ and $\lambda_S = 0.75$ instead of the default values.



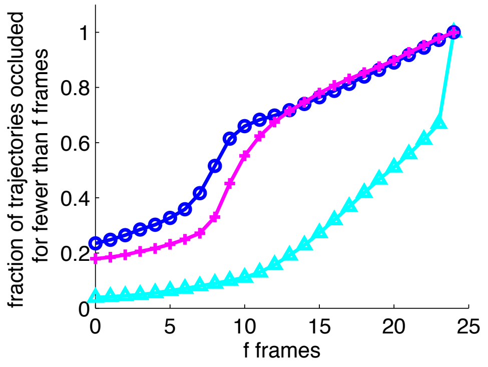
(a) Flowerbed (29 frames).



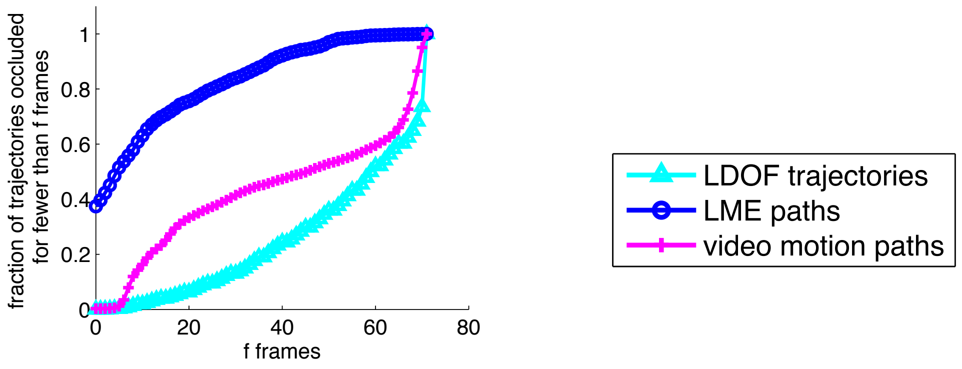
(b) Truck (33 frames).



(c) Marple1 (60 frames).

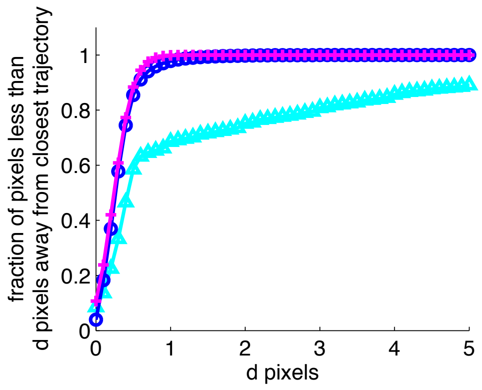


(d) Marple7 (25 frames).

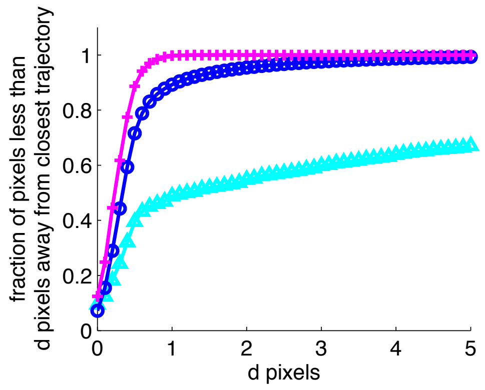


(e) Marple8 (72 frames).

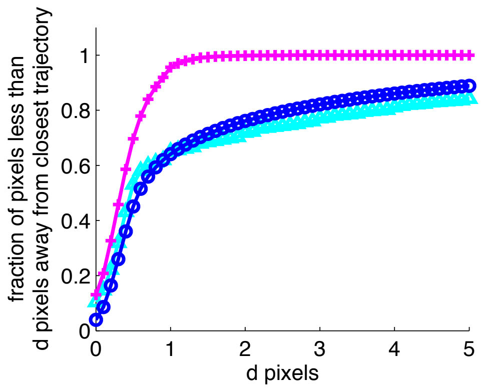
FIGURE 6.6: Cumulative distribution functions for path length. Video motion paths track points that are occluded in the LME reference frames, allowing for detection of true occlusions that LME paths miss. (For example, there should be no unoccluded points in marple8 but 40% of the LME paths are mistakenly marked as always visible.) LDOF trajectories are comparatively very short for all sequences.



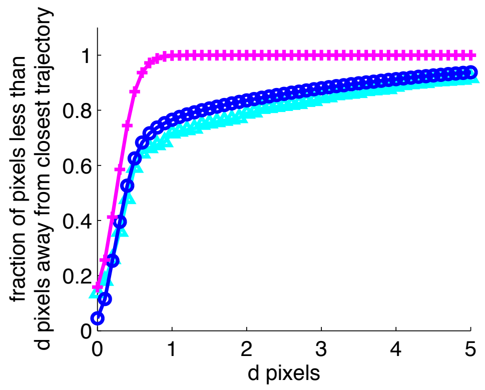
(a) Flowerbed.



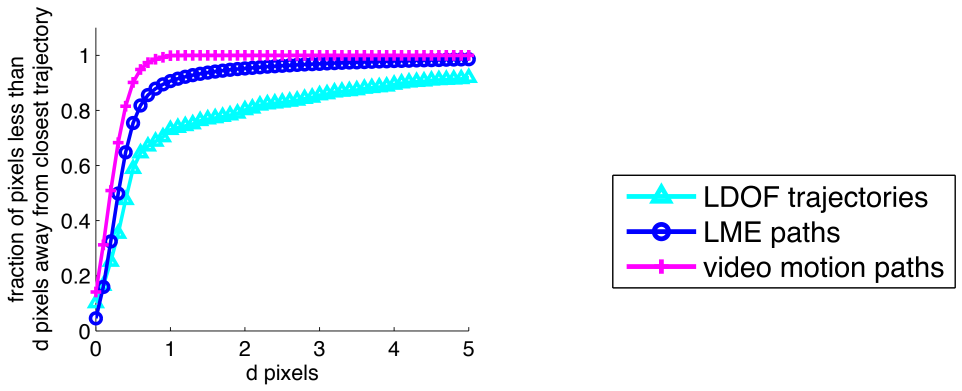
(b) Truck.



(c) Marple1.



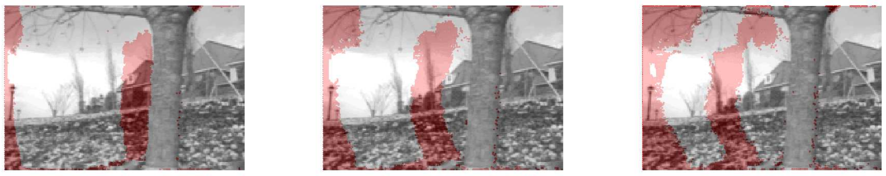
(d) Marple7.



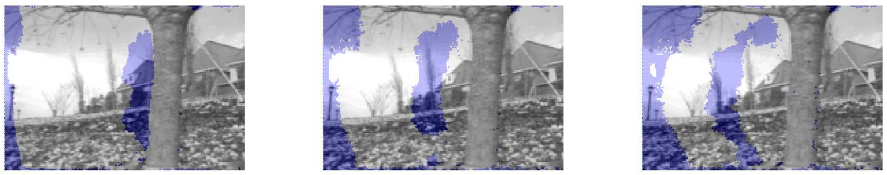
(e) Marple8.

FIGURE 6.7: Cumulative distribution functions for pixel distance. LDOF trajectories are only located in areas with enough texture. LME paths miss points not visible in the first or last frames. Video motion paths provide much more complete coverage, explaining virtually all pixels in any frame of the sequence.

(a) Original frames 10, 20, and 29.



(b) Points detected as occluded (in red) with default parameters.



(c) Points detected as occluded (in blue) with $\lambda_L = 0.5$ and $\lambda_S = 0.75$.

FIGURE 6.8: The visibility MRF can be sensitive to its parameters. In (b), we show the detected occlusion region in three frames (10, 20, and 29, respectively) with the default parameter settings used for all previous experiments. In (c), we show the detected occlusion region with larger λ_S and smaller λ_L, increasing the importance of spatial smoothness and decreasing the importance of the causal signal. Many occlusions are missed with this parameter setting. Detected occlusions are drawn as colored dots superimposed on the points in the first frame. For visual reference, (a) shows the frames in which the occlusions are detected.

7 Proposed Extensions

Our work advances the state of the art in video motion estimation. To scale our work to larger and longer sequences, there remain unanswered questions. In this chapter, we enumerate a few research directions we consider most important.

First, the creation of a good benchmark for video motion estimation would be an invaluable contribution to the scientific community. The Middlebury sequence has had a tremendous impact on research in optical flow in just a few short years. It introduced a standard set of metrics, allowing for quantitative analysis of competing ideas. An equivalent set of sequences for video motion estimation should include realistic rigid and non-rigid motion and should contain significant occlusion. The accompanying ground truth information should include correspondences across ephemeral occlusions. Computer graphics makes creation of such a dataset possible, but unfortunately both the creation and maintenance of the dataset amounts to a significant amount of work that is often not adequately rewarded in the research community.

In addition to improving our motion estimation results, we would like to use the extracted motion information to support higher-level computer vision tasks. Video segmentation is one obvious application. We expect the shapes of video motion paths, or of paths collected into video motion tubes, to be useful features for activity or event recognition and the related problem of content-based video retrieval. For all these applications, we would need to develop ways to describe and compare video motion paths, perhaps through their representation in the path motion basis.

The main limitation of our work is the computation time required with the current implementation, as we have focused primarily on accuracy rather than speed. We report running time for each sequence (after initialization) in Table 7.1. One iteration of the solver corresponds to 40 trust-region descent steps to adjust path estimates, visibility estimation using graph cuts, and the heuristic path update. The number of iterations required depends on the quality of the initial solution from the optical flow fields. Generally, sequences with more occlusion require more iterations to converge because we spend considerable computational effort recovering from errors inherited from the frame-to-frame optical flow fields. Developing techniques for initializing without estimates of optical flow would be useful. One possible approach would be to adapt the coarse-to-fine scheme used in traditional optical flow methods, starting from an initial estimate of all zero path coefficients. Because we also use optical flow

Table 7.1: Running time after initialization for video motion estimation. Our algorithm was implemented in a combination of Matlab and C++ and ran on shared machines using dual Intel 8-core Xeon processors (2.53GHz or 2.66GHz) with between 48GB and 96GB RAM. Marple1 was terminated after 91 calls to the trust-region optimizer. At this point, the path estimates had converged, but there remained slightly too many unexplained pixels as a result of shadows causing paths to be marked as occluded.

Sequence	**Size**	**Iterations completed**	**Video motion time**	**LDOF traj. time**
Flowerbed	$120 \times 175 \times 29$	8	1.4 hours	32 seconds
Truck	$167 \times 200 \times 33$	15	5.8 hours	71 seconds
Marple7	$175 \times 225 \times 25$	15	10.5 hours	54 seconds
Marple1	$144 \times 175 \times 60$	91	150.0 hours	88 seconds
Marple8	$175 \times 225 \times 72$	33	89.4 hours	171 seconds
Flag	$250 \times 250 \times 60$	7	29.1 hours	163 seconds
Flag (full res.)	$500 \times 500 \times 60$	–	–	2089 seconds

to estimate the number and location of anchor pixels, we would need to reconsider that step as well.

We could decrease running times by using more efficient numerical optimization techniques that do not require knowledge of the exact gradient. The fastest optical flow techniques (including the technique implemented in Sundaram et al. (2010)) leverage the parallel computing power of the GPU to speed computation. The challenge in porting our algorithms to the GPU lies in the variable nature of our neighborhood relationships. Unlike optical flow, the spatial neighbors of a path are not known *a priori* and do not form a lattice. Similarly, the graph representation of the visibility MRF does not have the regular connectivity that is common in computer vision problems.

It may also be possible to increase computation speed by adapting our mathematical model to incorporate a hierarchical approach. Currently, each path is essentially one pixel thick; we use a number of paths proportional to the area of the visible surfaces in the scene, regardless of the spatial complexity of the motion. We could instead define a tube-like object with non-trivial spatial thickness that could describe the motion of a larger surface patch with fewer parameters. Techniques for extracting superpixels in a single image (Ren and Malik, 2003) or intensity-based video segmentation (Grundmann et al., 2010; Xu et al., 2012a) could give an idea of the cross-section of each tube. Each tube would be associated with a single path or parametric family of paths, reducing the complexity of the model but ideally not its expressive power.

If we moved to a tube model, we would need to decide what it means for a tube to be partially occluded. In our current model, the visibility state of a path is binary; a world point is either completely occluded in a frame or completely visible. If we instead describe the motion of larger regions, we will need to correctly model the cost of a partial occlusion of an object. Rather than use our point-based model of occlusions, it would presumably be better to model the occlusion surfaces directly. Occlusion surfaces partition the space-time video cube into regions corresponding to individual objects. The point-based occlusion events could be recovered by detecting intersections between the extracted surfaces and paths.

We want to extend our approach to much longer sequences containing hundreds of frames. One way to approach this problem would be to break the larger sequence down into more manageable temporal chunks and compute video motion separately in each chunk. These partial solutions would need to be stitched together to form the full solution. We could stitch together paths for points that are visible in overlapping frames, but stitching together paths for occluded points is more difficult. There may be many layers of occluded points that project to the same pixel, so simply attaching two co-located occluded paths may not be correct. Paths on points that are never again visible do not need to be continued into neighboring chunks (and indeed should not be, as it is unlikely that the motion basis in the other chunks will correctly represent the motion of independent objects that are completely occluded). However, identifying these paths that may be terminated is a non-trivial problem.

Our path objective function assumes that brightness constancy holds along the entire length of a path. As we scale our approach to longer sequences, this assumption will become less and less reliable. It would be straightforward to change our objective function to evaluate color constancy or gradient constancy. However, we may want to include descriptors that are more likely to be invariant to appearance changes that are independent of motion in the scene. For example, we could use region descriptors (*e.g.*, SIFT (Lowe, 2004), SURF (Bay et al., 2008), etc.) in the data term, particularly in conjunction with the hierarchical approach that tracks regions. We would likely want to retain some pinpoint features such as the actual pixel intensities because most region descriptors are not designed to localize features with subpixel accuracy as is desired in motion estimation. These changes to the data term would impact the choice of numerical optimization routine. For example, it is not immediately clear that it would be possible to compute the analytic gradient of a descriptor with respect to the path coefficients.

Longer sequences are more likely to contain multiple independently moving objects, increasing the size of the required path basis. However, if the motion of the different objects is sufficiently independent, there should exist a basis for which only a few path coefficients are non-zero at each point. We could encourage this by adding a sparsity-inducing regularization term on the path coefficients (Costeira and Kanade, 1998; Rao et al., 2010). We would need to incorporate this term into the factorization algorithm that finds the initial basis as well. Our initial experiments suggest that combining motion estimation with motion segmentation through this term would significantly improve results. For example, we computed motion for the flag sequence with occlusions using a basis that included both the motion of the flag and the motion of the occluder and got an RMS endpoint error of 2.56 pixels for the paths on the flag. When we used a basis that included only the motion of the flag, without the motion of the occluder, our error dropped to 1.31 pixels.

Interestingly, the recovered paths include many that track the occluder for just a few frames (and are marked as visible in only those few frames) before being forced to diverge from the occluder's motion by the constraints of the subspace. Figure 7.1 shows the anchor points associated with paths that are visible for less than half of the frames in the sequence. The majority of these points are located along the linear tracks of the two synthetic occluders. These paths do not accurately report the motion of specific points on the occluders (because the true motion lies outside the path subspace), but they could be used to carve out the space-time volume that corresponds to the occluding object.

At its core, our method assumes that it is possible to find a path basis for a sequence. The experiment described above suggests that our technique may degrade

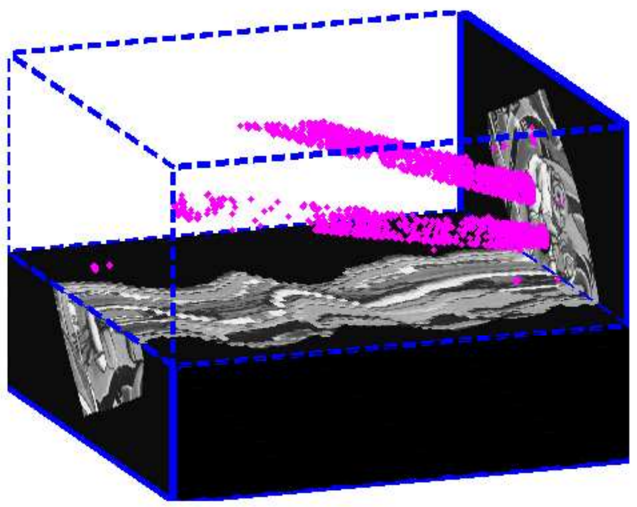


FIGURE 7.1: Using an incorrect motion subspace can still provide useful information. Here, we recover paths for the flag sequence with occlusions but provide a basis that represents the motion of the flag but not the occluder. Magenta points mark the location of anchor points for all paths marked as visible for fewer than 30 of the 60 frames of the sequence. These points are almost all located on the two synthetic occluders, which are small black disks that move with constant velocity through the scene.

gracefully as long as the basis represents most of the motion in the scene. (Note that the basis we used above accurately captured the motion of the dominant object.) While we have made significant progress in finding path bases, there are sequences for which it is currently too difficult to track enough points from frame to frame to make factorization possible, even with compaction working to decrease the amount of missing data. We encountered this problem when attempting to extract video motion for the MPI-Sintel sequences. Current frame-to-frame trackers could maintain enough tracks to model the motion of the background of many scenes but could not track foreground points for more than a few frames at a time. It may be that we could run our algorithm with a basis that fits the motion of the background only and still recover some useful information about foreground object without explictly recovering its motion. We could also represent the motion of the foreground object nonparametrically and use nonparametric regularization to encourage similarity to frame-to-frame track fragments when they are available.

Ultimately, one should consider not just the level of detailed motion information that can be recovered from a video sequence but also the level of detail that is required to support desired higher-level applications. In our work, we aim for the most complete representation of motion available without actually reconstructing the scene. We believe our results can be improved by considering the extensions outlined above. However, for some applications, it may not be necessary to carefully track every point through every frame of the sequence. Instead, it may only be necessary to recover the occluding surfaces corresponding to object boundaries, and this may be possible without ever explicitly solving for the motion of point features. If that is the case, then solving for full video motion would be overkill for those applications.

8

Summary and Conclusions

Understanding motion in video is key to meeting numerous challenges, from forensic analysis of public safety surveillance video to tracking cells in scientific experiments, and from management of personal video collections to autonomous navigation. In all of these applications, important questions can be answered by analyzing the motion of objects over significant temporal intervals. Our work addresses the problem of extracting this long-range motion information from general video sequences.

The defining feature of our work is our treatment of occlusions. Most frame-to-frame motion estimation techniques treat occlusions as the result of some spatially sparse but otherwise unknowable noise process. Occluded points are abandoned, as if swallowed up by a black hole, never to be seen again. This approach makes it difficult to analyze motions over long timespans because tracks keep getting destroyed by sometimes brief occlusions.

In contrast, we model occlusions as physical events with limited temporal extent. We recognize that it is often possible to associate points to each other across brief occlusions, and even to infer their locations during the intervening frames, by analyzing their motion during visible intervals in relation to the motion of other nearby points. To this end, we define video motion as a set of sequence-length paths covering the visible surfaces in the scene, representable in a single low-dimensional subspace and coupled with visibility flags that mark the frames in which each point is occluded. We find a sequence-specific basis from a representative set of frame-to-frame tracks. We do not require that these tracks be full length. Instead, we introduce a novel compaction-and-factorization algorithm that associates track fragments from rediscovered points to reduce the amount of missing data during the factorization step. We augment traditional frame-to-frame trackers with a history-sensitive feature initialization routine to ensure that points are rediscovered after being lost.

We use the estimated basis to compute paths, ensuring that nearly every pixel in the video is covered by some visible path. We simultaneously estimate the visible intervals for each path so that motion estimates are not corrupted by the appearance of occluders. Our occlusion model requires that some point be visible at every pixel and detects occlusions based on global analysis of the geometric properties of extracted paths and the appearance of the video when tracing each path. Our results on real sequences are state of the art, successfully tracking points on moving objects through significant occlusions.

The idea that motion can be accurately predicted through and across occlusions is still controversial. For example, the new MPI-Sintel dataset penalizes errors in optical flow estimates in "unmatched" (occluded) regions. Some researchers consider this unfair, protesting "How could you possibly estimate the location of the point when it isn't visible?" Our work suggests that there exist realistic sequences where this may be possible. Although we are not yet at the level of processing the MPI-Sintel data, we can track points through occlusions by relying on photometric evidence during visible intervals combined with geometric constraints provided by other visible points.

Appendix A

Trust-Region Newton-CG

The algorithms included here are reproduced from Nocedal and Wright (2006).

A.1 Trust-Region Optimization

The objective function to minimize is f, with $f_k = f(\boldsymbol{x}_k)$ the value at the solution found on step k. At each step, the objective function is approximated by the quadratic function

$$m_k(\boldsymbol{p}) = f_k + \boldsymbol{g}_k^T \boldsymbol{p} + \frac{1}{2}\boldsymbol{p}^T \boldsymbol{B}_k \boldsymbol{p} \ , \tag{A.1}$$

where $\boldsymbol{g}_k = \nabla f(\boldsymbol{x}_k)$ is the gradient of the function and $\boldsymbol{B}_k = \nabla^2 f(\boldsymbol{x}_k)$ is the Hessian, both evaluated at the current solution $\boldsymbol{x}_k$.

The ratio

$$\rho_k = \frac{f(\boldsymbol{x}_k) - f(\boldsymbol{x}_k + \boldsymbol{p}_k)}{m_k(\boldsymbol{0}) - m_k(\boldsymbol{p}_k)} \tag{A.2}$$

is the fraction of the predicted improvement (the denominator) that would be actually realized after the update proposed by $\boldsymbol{p}_k$. The update is found by solving

$$\boldsymbol{p}_k = \min_{||\boldsymbol{p}||<\Delta_k} m_k(\boldsymbol{p}) \ . \tag{A.3}$$

Here, Δ_k determines the size of the *trust region*, where we expect the approximation m_k to be close to the function f. If $\rho_k \ll 1$, then the quadratic approximation is poor and the size of the trust region should be decreased. If $\rho_k < 0$, the approximation is so poor that the predicted descent step is not a descent in the actual objective function and so should not be accepted.

The following algorithm implements the minimization procedure:

```
input: $\hat{\Delta} > 0, \Delta_0 \in (0, \hat{\Delta}), \eta \in [0, \frac{1}{4})$, and initial guess $\boldsymbol{x}_0$
for $k = 0, 1, 2, \ldots$ do
    Determine $\boldsymbol{p}_k$ from (A.3)
    Evaluate $\rho_k$ according to (A.2)

    // Change trust region size.
    if $\rho_k < \frac{1}{4}$ then
        // Approximation is poor.
        $\Delta_{k+1} = \frac{1}{4}\Delta_k$
    else if $\rho_k > \frac{3}{4}$ and $||\boldsymbol{p}_k|| = \Delta_k$ then
        // Progress impeded by size of trust region.
        $\Delta_{k+1} = \min(2\Delta_k, \hat{\Delta})$
    else
        $\Delta_{k+1} = \Delta_k$
    end

    // Update solution.
    if $\rho_k > \eta$ then
        $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{p}_k$
    else
        $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$
    end

    if converged then return $\boldsymbol{x}_k$
end
```

ALGORITHM A.1: Trust region optimization.

A.2 Trust-Region Newton-CG

The trust-region Newton-CG algorithm uses conjugate gradient iterations to solve for $\boldsymbol{p}_k$ approximately rather than exactly on line 2 of Algorithm A.1. The version presented here is from Steihaug (1983).

```
   input: trust region size $\Delta_k$, tolerance $\epsilon_k > 0$, gradient $\nabla f_k$, and Hessian $\boldsymbol{B}_k$
1  Initialize $\boldsymbol{z}_0 = \boldsymbol{0},\ \boldsymbol{r}_0 = \nabla f_k,\ \boldsymbol{d}_0 = -\boldsymbol{r}_0$
2  if $||\boldsymbol{r}_0|| < \epsilon_k$ then return $\boldsymbol{p}_k = \boldsymbol{0}$
3  for $j = 0, 1, 2, \ldots$ do
4      if $\boldsymbol{d}_j^T \boldsymbol{B}_k \boldsymbol{d}_j \leqslant 0$ then
           // Non-positive curvature found.
           // Minimum occurs on trust region boundary.
5          Find $\tau$ such that $\boldsymbol{p}_k = \boldsymbol{z}_j + \tau \boldsymbol{d}_j$ minimizes $m_k(\boldsymbol{p}_k)$ with $||\boldsymbol{p}_k|| = \Delta_k$.
6          return $\boldsymbol{p}_k = \boldsymbol{z}_j + \tau \boldsymbol{d}_j$
7      end
8      $\alpha_j = \frac{\boldsymbol{r}_j^T \boldsymbol{r}_j}{\boldsymbol{d}_j^T \boldsymbol{B}_k \boldsymbol{d}_j}$
9      $\boldsymbol{z}_{j+1} = \boldsymbol{z}_j + \alpha_j \boldsymbol{d}_j$
10     if $||\boldsymbol{z}_{j+1}|| \geqslant \Delta_k$ then
           // Next search step is outside trust region.
           // Min must be on boundary.
11         Find $\tau \geqslant 0$ such that $\boldsymbol{p}_k = \boldsymbol{z}_j + \tau \boldsymbol{d}_j$ satisfies $||\boldsymbol{p}_k|| = \Delta_k$.
12         return $\boldsymbol{p}_k = \boldsymbol{z}_j + \tau \boldsymbol{d}_j$
13     end
14     $\boldsymbol{r}_{j+1} = \boldsymbol{r}_j + \alpha_j \boldsymbol{B}_k \boldsymbol{d}_j$
       // Check convergence within tolerance.
15     if $||\boldsymbol{r}_{j+1}|| \leqslant \epsilon_k$ then return $\boldsymbol{p}_k = \boldsymbol{z}_{j+1}$
16
17     $\beta_{j+1} = \frac{\boldsymbol{r}_{j+1}^T \boldsymbol{r}_{j+1}}{\boldsymbol{r}_j^T \boldsymbol{r}_j}$
18     $\boldsymbol{d}_{j+1} = -\boldsymbol{r}_{j+1} + \beta_{j+1} \boldsymbol{d}_j$
19 end
```

Algorithm A.2: CG-Steihaug procedure for solving the Newton step (A.3).

A.3 Derivation of Gradient and Hessian

The algorithm described above requires the gradient and the Hessian of the objective function (3.5). The entries in the gradient are

$$\frac{\partial E}{\partial c_{pk}} = \frac{\partial}{\partial c_{pk}} E_D(\boldsymbol{c}_p) + \frac{\lambda}{2}\left(\sum_q \frac{\partial}{\partial c_{pk}} E_S(\boldsymbol{c}_p, \boldsymbol{c}_q) + \sum_q \frac{\partial}{\partial c_{pk}} E_S(\boldsymbol{c}_q, \boldsymbol{c}_p)\right) . \qquad \text{(A.4)}$$

Let us begin with $\frac{\partial}{\partial c_{pk}} E_S(\boldsymbol{c}_p, \boldsymbol{c}_q)$:

$$\frac{\partial}{\partial c_{pk}} E_S(\boldsymbol{c}_p, \boldsymbol{c}_q) = \alpha_{pq} \sum_{l=1}^{K} \frac{\partial}{\partial c_{pk}} \rho(c_{pl} - c_{ql}) \qquad \text{(A.5)}$$

$$= \alpha_{pq} \frac{c_{pk} - c_{qk}}{\sqrt{(c_{pk} - c_{qk})^2 + \epsilon^2}} . \qquad \text{(A.6)}$$

Here we use the fact that $\rho(s) = \sqrt{s^2 + \epsilon^2}$ so $\rho'(s) = \frac{s}{\sqrt{s^2+\epsilon^2}}$. The last term simplifies to the same expression:

$$\frac{\partial}{\partial c_{pk}} E_S(\boldsymbol{c}_q, \boldsymbol{c}_p) = \alpha_{qp} \sum_{l=1}^{K} \frac{\partial}{\partial c_{pk}} \rho(c_{ql} - c_{pl}) \qquad \text{(A.7)}$$

$$= \alpha_{qp} \frac{c_{qk} - c_{pk}}{\sqrt{(c_{qk} - c_{pk})^2 + \epsilon^2}}(-1) \qquad \text{(A.8)}$$

$$= \alpha_{pq} \frac{c_{pk} - c_{qk}}{\sqrt{(c_{pk} - c_{qk})^2 + \epsilon^2}} . \qquad \text{(A.9)}$$

The last line follows from the fact that our definition of α_{pq} is symmetric and that $(-x)^2 = x^2$. Note that we treat α_{pq} as a constant with respect to c_{pk}.

The contribution of the first term is more complicated. Recall that $E_D(\boldsymbol{c}_p) = \sum_{t=1}^{F} \nu_p(t)\rho(\Delta I_p(t))$. By the chain rule, we have

$$\frac{\partial}{\partial c_{pk}} E_D(\boldsymbol{c}_p) = \sum_{t=1}^{F} \nu_p(t) \frac{\Delta I_p(t)}{\sqrt{\Delta I_p(t)^2 + \epsilon^2}} \left(\frac{\partial}{\partial c_{pk}} \Delta I_p(t)\right) . \qquad \text{(A.10)}$$

$\Delta I_p(t)$ is a difference of two terms: $I(\boldsymbol{x}_p(t), t)$, which depends on c_{pk} because $\boldsymbol{x}_p(t)$ is a function of c_{pk}, and $I(\boldsymbol{u}_p, \tau_p)$, which is a constant with respect to c_{pk}. Let $\boldsymbol{x}_p(t) = (x_p, y_p)$, $\boldsymbol{u}_p = (u_p, v_p)$, and $\boldsymbol{\varphi}_k(t) = (\phi^x_k(t), \phi^y_k(t))$. From our parameterization,

$$x_p = u_p + \sum_{l=1}^{K} c_{pl} \left(\phi^x_l(t) - \phi^x_l(\tau_p)\right) \ , \tag{A.11}$$

so,

$$\frac{\partial}{\partial c_{pk}} x_p = \phi^x_k(t) - \phi^x_k(\tau_p) \ . \tag{A.12}$$

Similarly, we can see that

$$\frac{\partial}{\partial c_{pk}} y_p = \phi^y_k(t) - \phi^y_k(\tau_p) \ . \tag{A.13}$$

So, by application of the chain rule again, we have

$$\frac{\partial}{\partial c_{pk}} E_D(\boldsymbol{c}_p) = \sum_{t=1}^{F} \nu_p(t) \frac{\Delta I_p(t)}{\sqrt{\Delta I_p(t)^2 + \epsilon^2}} \left[\ I_x \left(\phi^x_k(t) - \phi^x_k(\tau_p)\right) + I_y \left(\phi^y_k(t) - \phi^y_k(\tau_p)\right) \ \right] \ , \tag{A.14}$$

where I_x and I_y are the partial derivatives of the image $I(x, y, t)$, evaluated at $\boldsymbol{x}_p(t)$.

Combining all three terms, we have:

$$\begin{aligned} \frac{\partial E}{\partial c_{pk}} &= \sum_{t=1}^{F} \nu_p(t) \frac{\Delta I_p(t)}{\sqrt{\Delta I_p(t)^2 + \epsilon^2}} \left[\ I_x \left(\phi^x_k(t) - \phi^x_k(\tau_p)\right) + I_y \left(\phi^y_k(t) - \phi^y_k(\tau_p)\right) \ \right] \\ &\quad + \lambda \sum_q \alpha_{pq} \frac{c_{pk} - c_{qk}}{\sqrt{(c_{pk} - c_{qk})^2 + \epsilon^2}} \ . \end{aligned} \tag{A.15}$$

To compute the Hessian, we must take another derivative. Each element can be computed through applications of the chain rule and the product rule following the example of the gradient computation. The entry for $\frac{\partial}{\partial c_{pk}} \frac{\partial}{\partial c_{ql}} E$ is non-zero only if

$p = q$ or if $p \neq q$ but $k = l$ and $\alpha_{pq} > 0$. In the first case, the entry includes a contribution from the data term $E_D(\boldsymbol{c}_p)$ and from the smoothness term if $k = l$. If $p \neq q$, there is only a contribution from the smoothness term.

Bibliography

Akhter, I., Sheikh, Y., Khan, S., and Kanade, T. (2011), “Trajectory space: a dual representation for nonrigid structure from motion,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33, 1442–1456.

Alvarez, L., Deriche, R., Papdopoulo, T., and Sánchez, J. (2007), “Symmetrical dense optical flow estimation with occlusions detection,” *International Journal of Computer Vision*, 75, 371–385.

Anandan, P. (1989), “A computational framework and an algorithm for the measurement of visual motion,” *International Journal of Computer Vision*, 2, 283–310.

Apostoloff, N. and Fitzgibbon, A. (2005), “Learning spatiotemporal T-junctions for occlusion detection,” in *Computer Vision and Pattern Recognition (CVPR).*

Aujol, J.-F., Gilboa, G., Chan, T., and Osher, S. (2006), “Structure-texture image decomposition – modeling, algorithms, and parameter selection,” *International Journal of Computer Vision*, 67, 111–136.

Ayvaci, A., Raptis, M., and Soatto, S. (2012), “Sparse occlusion detection with optical flow,” *International Journal of Computer Vision*, 97, 322–338.

Baker, S., Scharstein, D., Lewis, J., Roth, S., Black, M. J., and Szeliski, R. (2011), “A database and evaluation methodology for optical flow,” *International Journal of Computer Vision*, 92, 1–31.

Bay, H., Ess, A., Tuytelaars, T., and Van Gool, L. (2008), “Speeded-up robust features (SURF),” *Computer Vision and Image Understanding*, 110, 346–359.

Bennett, J. and Lanning, S. (2007), “The Netflix Prize,” in *KDD Cup and Workshop.*

Bergen, J. R., Anandan, P., Hanna, K. J., and Hingorani, R. (1992), “Hierarchical model-based motion estimation,” in *European Conference on Computer Vision (ECCV).*

Black, M. J. and Anandan, P. (1996), “The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields,” *Computer Vision and Image Understanding*, 63, 75–104.

Black, M. J. and Rangarajan, A. (1996), "On the unification of line processes, outlier rejection, and robust statistics with applications in early vision," *International Journal of Computer Vision*, 19, 57–91.

Boykov, Y. and Kolmogorov, V. (2004), "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26, 1124–1137.

Bradski, G. (2000), "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*.

Brand, M. (2005), "A direct method for 3D factorization of nonrigid motion observed in 2D," in *Computer Vision and Pattern Recognition (CVPR)*.

Bregler, C., Hertzmann, A., and Biermann, H. (2000), "Recovering non-rigid 3D shape from image streams," in *Computer Vision and Pattern Recognition (CVPR)*.

Brox, T. and Malik, J. (2010), "Object segmentation by long term analysis of point trajectories," in *European Conference on Computer Vision (ECCV)*.

Brox, T. and Malik, J. (2011), "Large displacement optical flow: descriptor matching in variational motion estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33, 500–513.

Brox, T., Bruhn, A., Papenberg, N., and Weickert, J. (2004), "High accuracy optical flow estimation based on a theory for warping," in *European Conference on Computer Vision (ECCV)*.

Brox, T., Bregler, C., and Malik, J. (2009), "Large displacement optical flow," in *Computer Vision and Pattern Recognition (CVPR)*.

Buchanan, A. M. and Fitzgibbon, A. (2005), "Damped Newton algorithms for matrix factorization with missing data," in *Computer Vision and Pattern Recognition (CVPR)*.

Butler, D. J., Wulff, J., Stanley, G. B., and Black, M. J. (2012), "A naturalistic open source movie for optical flow evaluation," in *European Conference on Computer Vision (ECCV)*.

Candès, E. and Plan, Y. (2010), "Matrix completion with noise," *Proceedings of the IEEE*, 98, 925–936.

Candès, E. and Recht, B. (2009), "Exact matrix complection via convex optimization," *Foundations of Computational Mathematics*, 9, 717–772.

Christy, S. and Horaud, R. (1996), "Euclidean shape and motion from multiple perspective views by affine iterations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18, 1098–1104.

Costeira, J. P. and Kanade, T. (1998), "A multibody factorization method for independently moving objects," *International Journal of Computer Vision*, 29, 159–179.

Darrell, T. and Fleet, D. (1995), "Second-order method for occlusion relationships in motion layers," Tech. Rep. 314, Massachusetts Institute of Technology.

De la Torre, F. and Black, M. J. (2001), "Robust principal component analysis for computer vision," in *International Conference on Computer Vision (ICCV)*.

Dellaert, F., Seitz, S., Thorpe, C. E., and Thrun, S. (2000), "Structure from motion without correspondence," in *Computer Vision and Pattern Recognition (CVPR)*.

Farnebäck, G. (2003), "Two-frame motion estimation based on polynomial expansion," in *Image Analysis*.

Feldman, D. and Weinshall, D. (2008), "Motion segmentation and depth ordering using an occlusion detector," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30, 1171–1185.

Fragkiadaki, K., Zhang, G., and Shi, J. (2012), "Video segmentation by tracing discontinuities in a trajectory embedding," in *Computer Vision and Pattern Recognition (CVPR)*.

Garg, R., Pizarro, L., Rueckert, D., and Agapito, L. (2010), "Dense multi-frame optic flow for non-rigid objects using subspace constraints," in *Asian Conference on Computer Vision (ACCV)*.

Garg, R., Roussos, A., and Agapito, L. (2011), "Robust trajectory-space TV-L1 optical flow for non-rigid sequences," in *Energy Minimization Methods in Computer Vision and Pattern Recognition*.

Gotardo, P. F. and Martinez, A. M. (2011), "Non-rigid structure from motion with complementary rank-3 spaces," in *Computer Vision and Pattern Recognition (CVPR)*.

Grenander, U., Chow, Y., and Keenan, D. M. (1991), *Hands: a pattern theoretic study of biological shapes*, Springer-Verlag New York, Inc., New York.

Grundmann, M., Kwatra, V., Han, M., and Essa, I. (2010), "Efficient hierarchical graph-based video segmentation," in *Computer Vision and Pattern Recognition (CVPR)*.

Guilford, J. P. (1929), "Illusory movement from a rotating barber pole," *The American Journal of Psychology*, 41, 686–687.

Gurobi Optimization, Inc. (2013), "Gurobi Optimizer Reference Manual," `http://www.gurobi.com`.

Horn, B. and Schunck, B. (1981), "Determining optical flow," *Artificial Intelligence*, 17, 185–203.

Hubert, L. and Arabie, P. (1985), "Comparing partitions," *Journal of Classification*, 2, 193–218.

Ince, S. and Konrad, J. (2008), "Occlusion-aware optical flow estimation," *IEEE Transactions on Image Processing*, 17, 1443–1451.

Irani, M. (2002), "Multi-frame correspondence estimation using subspace constraints," *International Journal of Computer Vision*, 48, 173–194.

Jepson, A. D., Fleet, D. J., and Black, M. J. (2002), "A layered motion representation with occlusion and compact spatial support," in *European Conference on Computer Vision (ECCV).*

Jojic, N. and Frey, B. J. (2001), "Learning flexible sprites in video layers," in *Computer Vision and Pattern Recognition (CVPR).*

Ke, Q. and Kanade, T. (2005), "Robust L_1 norm factorization in the presence of outliers and missing data by alterative convex programming," in *Computer Vision and Pattern Recognition (CVPR).*

Koffka, K. (1935), *Principles of Gestalt Psychology*, Harcourt, New York.

Kolmogorov, V. and Zabih, R. (2004), "What energy functions can be minimized via graph cuts?" *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26, 147–159.

Kumar, M. P., Torr, P. H., and Zisserman, A. (2008), "Learning layered motion segmentations of video," *International Journal of Computer Vision*, 76, 301–319.

Leutenegger, S., Chli, M., and Siegwart, R. Y. (2011), "BRISK: Binary robust invariant scalable keypoints," in *International Conference on Computer Vision (ICCV).*

Lezama, J., Alahari, K., Sivic, J., and Laptev, I. (2011), "Track to the future: spatio-temporal video segmentation with long-range motion cues," in *Computer Vision and Pattern Recognition (CVPR).*

Lowe, D. G. (2004), "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, 60, 91–110.

Lucas, B. and Kanade, T. (1981), "An iterative image registration technique with an application to stero vision," in *7th International Joint Conference on Artificial Intelligence (IJCAI).*

Mason, J., Ricco, S., and Parr, R. (2011), "Textured occupancy grids for monocular localization without features," in *Proceedings of the 2011 IEEE International Conference on Robotics and Automation*.

Matikainen, P., Hebert, M., and Sukthankar, R. (2009), "Trajectons: Action recognition through the motion analysis of tracked features," in *Proceedings of the ICCV Workshop on Video-Oriented Object and Event Classification*.

Nocedal, J. and Wright, S. J. (2006), *Numerical Optimization*, Springer Series in Operations Research, Springer Verlag, 2nd edn.

Olsen, S. and Bartoli, A. (2008), "Implicit non-rigid structure-from-motion with priors," *Journal of Mathematical Imaging and Vision*, 31, 233–244.

Perbet, F., Maki, A., and Stenger, B. (2009), "Correlated probabilistic trajectories for pedestrian motion detection," in *International Conference on Computer Vision (ICCV)*.

Pizarro, D. and Bartoli, A. (2012), "Feature-based deformable surface detection with self-occluison reasoning," *International Journal of Computer Vision*, 97, 54–70.

Poelman, C. J. and Kanade, T. (1997), "A paraperspective factorization method for shape and motion recovery," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19, 206–218.

Prest, A., Ferrari, V., and Schmid, C. (2013), "Explicit modeling of human-object interactions in realistic videos," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35, 835–848.

Rand, W. M. (1971), "Objective criteria for the evaluation of clustering methods," *Journal of the American Statistical Association*, 66, 846–850.

Rao, S., Tron, R., Vidal, R., and Ma, Y. (2010), "Motion segmentation in the presence of outlying, incomplete, or corrupted trajectories," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32, 1832–1845.

Ren, X. and Malik, J. (2003), "Learning a classification model for segmentation," in *International Conference on Computer Vision (ICCV)*.

Ricco, S. and Chen, M. (2009), "Classification of scan location in retinal optical coherence tomography," in *IEEE International Symposium on Biomedical Imaging (ISBI '09)*.

Ricco, S. and Tomasi, C. (2009), "Fingerspelling recognition through classification of letter-to-letter transitions," in *Asian Conference on Computer Vision (ACCV)*.

Ricco, S. and Tomasi, C. (2012a), "Dense Lagrangian motion estimation with occlusions," in *Computer Vision and Pattern Recognition (CVPR).*

Ricco, S. and Tomasi, C. (2012b), "Simultaneous compaction and factorization of sparse image motion matrices," in *European Conference on Computer Vision (ECCV).*

Ricco, S., Chen, M., Ishikawa, H., Wollstein, G., Xu, J., and Schuman, J. (2009), "Correcting motion artifacts in retinal spectral domain optical coherence tomography via image registration," in *International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI).*

Ristivojevic, M. and Konrad, J. (2006), "Space-time image sequence analysis: object tunnels and occlusion volumes," *IEEE Transactions on Image Processing*, 15, 364–376.

Sand, P. (2006), "Long-range video motion estimation using point trajectories," Ph.D. thesis, Massachusetts Institute of Technology.

Sand, P. and Teller, S. (2008), "Particle Video: long-range motion estimation using point trajectories," *International Journal of Computer Vision*, 80, 72–91.

Shi, J. and Tomasi, C. (1994), "Good features to track," in *Computer Vision and Pattern Recognition (CVPR).*

Steihaug, T. (1983), "The conjugate gradient method and trust regions in large scale optimization," *SIAM Journal on Numerical Analysis*, 20, 626–637.

Sturm, P. and Triggs, B. (1996), "A factorization based algorithm for multi-image projective structure and motion," in *European Conference on Computer Vision (ECCV).*

Sun, D., Sudderth, E. B., and Black, M. J. (2010a), "Layered image motion with explicit occlusions, temporal consistency, and depth ordering," in *Advances in Neural Information Processing Systems 23.*

Sun, D., Roth, S., and Black, M. (2010b), "Secrets of optical flow estimation and their principles," in *2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

Sun, D., Sudderth, E. B., and Black, M. J. (2012), "Layered segmentation and optical flow estimation over time," in *Computer Vision and Pattern Recognition (CVPR).*

Sun, D., Wulff, J., Sudderth, E. B., Pfister, H., and Black, M. J. (2013), "A fully-connected layered model of foreground and background flow," in *Computer Vision and Pattern Recognition (CVPR).*

Sundaram, N., Brox, T., and Keutzer, K. (2010), "Dense point trajectories by GPU-accelerated large displacement optical flow," in *European Conference on Computer Vision (ECCV)*.

Tomasi, C. and Kanade, T. (1991), "Detection and tracking of point features," Tech. Rep. CMU-CS-91-132, Carnegie Mellon University.

Tomasi, C. and Kanade, T. (1992), "Shape and motion from image streams under orthography: a factorization method," *International Journal of Computer Vision*, 9, 137–154.

Torresani, L. and Bregler, C. (2002), "Space-time tracking," in *European Conference on Computer Vision (ECCV)*.

Torresani, L., Hertzmann, A., and Bregler, C. (2008), "Nonrigid structure-from-motion: Estimating shape and motion with hierarchical priors," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30, 878–892.

Triggs, B., McLauchlan, P. F., Hartley, R., and Fitzgibbon, A. (1999), "Bundle adjustment – a modern synthesis," in *Proceedings of the International Workshop on Vision Algorithms: Theory and Practice*.

Veksler, O. (1999), "Efficient graph-based energy minimization methods in computer vision," Ph.D. thesis, Cornell University.

Vidal, R., Tron, R., and Hartley, R. (2008), "Multiframe motion segmentation with missing data using PowerFactorization and GPCA," *International Journal of Computer Vision*, 79, 85–105.

Wallach, H. (1935), "Über visuell wahrgenommene Bewegungsrichtung," *Psychologische Forschung*, 20, 325–380.

Wang, H., Kläser, A., Schmid, C., and Liu, C.-L. (2011), "Action recognition by dense trajectories," in *Computer Vision and Pattern Recognition (CVPR)*.

Wang, J. Y. and Adelson, E. H. (1994), "Representing moving images with layers," *IEEE Transactions on Image Processing*, 3, 625–638.

Weickert, J. and Schnörr, C. (2001), "Variational optic flow computation with a spatio-temporal smoothness constraint," *Journal of Mathematical Imaging and Vision*, 14, 245–255.

Weiss, Y. (1997), "Smoothness in layers: motion segmentation using nonparametric mixture estimation," in *Computer Vision and Pattern Recognition (CVPR)*.

Wiberg, T. (1976), "Computation of principal components when data are missing," in *Second Symposium on Computational Statistics*, pp. 229–326.

Xiao, J., Cheng, H., Sawhney, H., Rao, C., and Isnardi, M. (2006a), "Bilaterial filtering-based optical flow estimation with occlusion detection," in *European Conference on Computer Vision (ECCV)*.

Xiao, J., Chai, J., and Kanade, T. (2006b), "A closed-form solution to non-rigid shape and motion recovery," *International Journal of Computer Vision*, 67, 233–246.

Xu, C., Xiong, C., and Corso, J. J. (2012a), "Streaming hierarchical video segmentation," in *European Conference on Computer Vision (ECCV)*.

Xu, L., Jia, J., and Matsushita, Y. (2012b), "Motion detail preserving optical flow estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34, 1744–1757.

Yan, J. and Pollefeys, M. (2008), "A factorization-based approach for articulated nonrigid shape, motion, and kinematic chain recovery from video," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30, 865–877.

YouTube (2013), "Statistics – YouTube," `http://www.youtube.com/yt/press/statistics.html`.

Biography

Susanna Maria Ricco was born February 1, 1984 in Albuquerque, New Mexico, where she learned to appreciate green chile and hate rain. She attended Harvey Mudd College in Claremont, CA, where she was seduced into studying computer science by an artificial Connect Four player and an arrow-following, scavenger-hunting robot named Twitchy. She graduated with a B.S. in Computer Science - Mathematics in May 2006. She obtained an M.S. in Computer Science from Duke University in December 2009 and a Ph.D. in September 2013.

Ricco was the recipient of an NSF Graduate Research Fellowship. In addition to her work on video motion estimation (Ricco and Tomasi, 2012a,b), she has published work on fingerspelling recognition (Ricco and Tomasi, 2009), analysis of and automatic artifact removal in retinal images (Ricco and Chen, 2009; Ricco et al., 2009), and robot localization with a single camera (Mason et al., 2011).

She joined Google Research in September 2013.