AN ABSTRACT OF THE DISSERTATION OF

Robin W. Hess for the degree of Doctor of Philosophy in Computer Science presented on June 14, 2012.

Title: Toward Computer Vision for Understanding American Football in Video

Abstract approved: _____

Alan P. Fern

In this work, I examine the problem of understanding American football in video. In particular, I present several mid-level computer vision algorithms that each accomplish a different sub-task within a larger system for annotating, interpreting, and analyzing collections of American football video. The analysis of football video is useful in its own right, as teams at all levels from high school to professional football currently spend thousands of dollars and countless human work hours processing video of their own play and the play of their opponents with the aim of developing strategy and improving performance. However, because football is an extremely challenging visual domain, with difficulties ranging from the chaotic motion and identical appearance of the players to the visual clutter on the field in the form of logos and other markings, computer vision algorithms developed towards the end goal of understanding American football are broadly applicable across a variety of visual problems.

I address four specific football-related problems in this thesis. First, I describe an approach for registering video with a static model (i.e. the football field in the American football domain) using a novel concept of locally distinctive invariant image feature matches. I also introduce a novel empirical registration transform stability test, which we use to initialize our registration procedure.

Second, I outline a novel method for constructing mosaics from collections of video. This method takes a greedy utility maximization approach to build mosaics that achieve user-definable mosaic quality objectives. While broadly applicable, our mosaicing approach accomplishes several tasks specifically relevant to the analysis of football video, including automatically con-

structing reference image sets for our video registration procedure and for computing background models for initial formation recognition and player tracking algorithms.

Third, I present an approach for recognizing initial player formations. This approach, called the Mixture-of-Parts Pictorial Structure (MoPPS) model, extends classical pictorial structures to recognize multi-part objects whose parts can vary in both type and location and for which an object part's location can depend on its type. While this model is effective in the American football domain, it is also broadly applicable.

Finally, I address the problem of tracking football players through video using a novel particle filtering formulation and an associated discriminative training procedure that directly maximizes filter performance based on observed errors during tracking. This particle filtering framework and training procedure are also broadly applicable.

For each of these algorithms, I also present a series of detailed experiments demonstrating the method's effectiveness in the American football domain. As a further contribution, I have made the data sets from most of these experiments publicly available.

Toward Computer Vision for Understanding American Football in Video

by

Robin W. Hess

A DISSERTATION

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Doctor of Philosophy

Presented June 14, 2012
Commencement June 2013

Doctor of Philosophy dissertation of Robin W. Hess presented on June 14, 2012.

APPROVED:

Major Professor, representing Computer Science

Director of the School of Electrical Engineering and Computer Science

---

Dean of the Graduate School

I understand that my dissertation will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my dissertation to any reader upon request.

Robin W. Hess, Author

ACKNOWLEDGEMENTS

After finishing the entirety of this dissertation, I have come back to write these acknowledgments, and it has just now struck me that this small part of this larger work may be the most challenging for me. The problem here is that there are countless people who deserve my gratitude and more for helping me to persist through—even to enjoy—the whole of my graduate school career, and I am wholly unconfident that I can, in just a few short paragraphs here, offer proper thanks to everyone who deserves them. Still, I will try, and I will ask the many folks who I shall certainly fail to acknowledge properly to please forgive me for being inarticulate and incomplete here.

First, I owe a debt of gratitude to my advisor, Dr. Alan Fern. Alan, I feel extremely lucky to have been able to work with you. I sincerely admire your insight and your ability to break down and fundamentally understand a problem. During my time working with you, I often struggled to live up to the example you set as a computer scientist, and, while I may never be as insightful as you or possess your analytical skill, struggling to match your standard has been a valuable endeavor in itself. I greatly appreciate the patience and support you've showed me as I have gone through the ups and downs of this struggle. In the end, I believe I have emerged as a better computer scientist who, while perhaps not of your stature, will still be able to make meaningful contributions to the field, and I attribute this in no small part to your skill as an advisor.

I am also extremely grateful to the other OSU Computer Science faculty members who have served on my PhD committee: Dr. Prasad Tadepalli, Dr. Tom Dietterich, Dr. Ron Metoyer, Dr. Sinisa Todorovic, and Dr. Eric Mortensen. Through working with each of you in courses, on committees, and on other projects, you've each been open and welcoming, and you've taught me a great deal along the way, as well. Thanks to each of you for that.

I would also like to thank Paul Paulson of the OSU Computer Science faculty for giving me the opportunity to work with him to transform the undergraduate Operating Systems course at OSU into something unique and, I believe, extremely worthwhile for students. Working on this course was unquestionably worthwhile for me, and I value this work as highly as the work I put into this dissertation. Thank you, Paul.

Many thanks also to my fellow graduate students in the Computer Science department. I am quite grateful for the many interesting discussions—on computer science and many other things—we have had over the years. You all helped make time at the office not only bearable, but enjoyable (though perhaps not always productive), and I appreciate that greatly. Thanks es-

pecially to Aaron Wilson, Aswin Raghavan, Avneet Sandhu, Chandan Sarkar, Crystal Ju, Ethan Dereszynski, Jana Doppa, Javad Azimi, Jervis Pinto, Jesse Hostetler, Neville Mehta, Sean McGregor, and Tuan Pham for this. Thanks, too, to Chris Chambers, David Piorkowski, Eric Walkingshaw, Jeremy Chen, Jun Yu, and the my other great teammates on the KEC Krew.

Apart from my work in Computer Science, my work with the Coalition of Graduate Employees, AFT Local 6069, has been invaluable to me. I can honestly say that no other experience in my life has helped me to grow as a human being in the way my work with CGE has, and I am indebted to the many other unionists and friends who worked alongside me during my time with CGE. Thanks especially to Allison Weinsteiger, Angela Brandt, Angela McClendon, Ashley Bromley, Barry Walker, Dennis Dugan, Jack Day, Joe Tyburczy, John Osborne, Katie Stofer, Lauren Atwell, Liz Dickinson, Matt Loewen, Michelle Marie, Michelle Zellers, Mindy Crandall, Sarah Cunningham, Sean McGregor, and Wren Keturi for your tireless work on behalf of OSU's graduate employees these past few years. Thanks also to the many other activists and officers whom I did not list here. Your work for CGE was no less valuable. I truly believe we all helped accomplish great things and that the organization we helped to build will continue to flourish long into the future.

I have also been fortunate to have a truly extraordinary group of friends here in Corvallis. I don't think it'd be appropriate to list you by name here, but each of you has not only been more fun and treated me better than any other group of friends I've ever had, but you have each challenged me in different ways that have helped me grow immensely as a person. I believe that thanks to you, I am more aware, more open, more understanding, more secure, and just generally a better human being, and I can't thank you enough for that. You all are some of the most genuine, open, and generous people I've ever been associated with, and you are also just incredibly goddamn fun. Thanks for everything.

Finally, I owe basically every good thing in my life to my family. Mom and Dad, I cannot imagine having better parents. You have both been so unfailingly loving and supportive, and you have participated in my life in a way for which I cannot even begin to express my appreciation. I try my best to be a good person, and if I at all succeed in that, it is because of the way you raised me. Thank you. I love you both.

Chris and Ryan, I also cannot imagine having better brothers. I am constantly awed at what you have done and what you are doing in your lives, and I feel that what I have achieved in my life is so small in comparison. You each have an approach to life that I admire. You both are also just pretty freaking cool. I hope some day I can be as cool and as generally admirable as each of

you. I love you both.

TABLE OF CONTENTS

Page

1 Introduction 1

1.1 Previous Related Work 5

2 Video Registration 10

2.1 Method 13
2.1.1 Detecting Invariant Image Features 13
2.1.2 Assembling a Set of Model Features 14
2.1.3 Matching Image Features 15
2.1.4 Computing Registration Transforms from Sets of Point Correspondences 16
2.1.5 Using Local Distinctiveness to Find Additional Correspondences . . . 17
2.1.6 Stability Test for Core Set Initialization 20

2.2 Experiments 21

2.3 Conclusions 25

3 Automatically Generating a Reference Set for Video Registration 27

3.1 Utility Maximization Framework 29

3.2 The Utility Function 31

3.3 Efficient Mosaic Estimation 34
3.3.1 Velocity and Location 34
3.3.2 Matchability from Mosaic Overlap 35
3.3.3 Matchability from Proximity to Unmatchable Frames 36
3.3.4 Final Mosaic Estimate 37

3.4 Results 37
3.4.1 Implementation Details 37
3.4.2 Single-Video Experiments 38
3.4.3 Multi-Video Experiments 41

3.5 Conclusion 42

4 Recognizing Initial Player Formations with Mixture-of-Parts Pictorial Structures 46

4.1 Pictorial Structures 49

4.2 Mixture-of-Parts Pictorial Structures 50
4.2.1 General MoPPS Model 50
4.2.2 The MoPPS Tree Representation 52

4.3 MoPPS Inference 53

TABLE OF CONTENTS (Continued)

Page

4.3.1 Branch-and-bound search 53
4.3.2 Lower bound computation 54
4.3.3 Upper bound computation 56
4.4 Experiments in American Football 56
4.4.1 Domain Description 57
4.4.2 MoPPS Model for Football Formations 58
4.4.3 Search strategies considered 60
4.4.4 Results 60
4.5 Summary and Future Work 62

5 Player Tracking with Discriminatively Trained Particle Filters 65
5.1 Related Work 66
5.2 Pseudo-Independent Log-Linear Filters 68
5.3 Error-Driven Discriminative Filter Training 71
5.4 Improving Training Computation Time 74
5.5 Experiments and Results 75
5.5.1 Football domain modeling 76
5.5.2 Results 78
5.6 Summary and Future Work 82

6 Conclusion 83
6.1 Recent Related Work on Multi-Object Tracking 84
6.2 Closing Thoughts 87

Bibliography 88

Appendices 96
A Our Recent Efforts in Multi-Object Tracking 97

LIST OF FIGURES

Figure		Page
1.1	The architecture of the four football video analysis procedures outlined in this thesis.	4
2.1	Non distinctive visual features in American football video.	12
2.2	A set of reference images from the American football domain registered with an overhead view of the field.	14
2.3	Image feature matches between video frames from the Amerian football domain.	15
2.4	Video frames with good and bad core set stability values.	21
2.5	Registration results from the American football domain.	22
2.6	Error plots comparing our registration method to naïve registration.	24
3.1	Mosaicing results for five single-video experiments.	40
3.2	Proportion of the in-bounds football field covered by the mosaic computed after each iteration of our greedy procedure.	43
3.3	A mosaic generated from a collection of 14 short videos.	44
3.4	Mosaics generated for the 15-video American football data set after 5, 10, 15, 20, and 25 iterations of our greedy procedure.	45
4.1	Significant variance between different football formations.	48
4.2	Subtle variations between different football formations.	57
4.3	MoPPS tree representation for initial player formations in American football.	58
4.4	Anytime behavior of BBS-based MoPPS inference.	63
5.1	A typical video frame from our dataset.	76
5.2	Comparison of mean squared error rates for different tracking approaches.	79
5.3	Comparison of failure rates for different tracking approaches.	80

LIST OF TABLES

Table Page

4.1 Summary of statistics of various search strategies for MoPPS inference. 61

5.1 Average run time comparison for tracking approaches. 82

LIST OF ALGORITHMS

Algorithm		Page
1	Stability Test for Core Set Initialization	20
2	Greedy procedure for multi-video mosaicing	31
3	Best-first branch-and-bound MoPPS tree search	55
4	Error-driven particle filter training	72

Chapter 1: Introduction

American football is at once chaotic and structured. Its chaos is evident: players run, cut, spin, jump, and dodge to evade each other; they crash into one another to block or tackle; the ball is blocked, or dropped, or intercepted, or fumbled; the wind blows; it snows or rains; it is foggy or muddy. Yet within all this chaos, structure exists: the game's rules dictate where players may line up and when and how they may move; players move in highly coordinated patterns; these players exhibit tendencies; coaches form detailed plans describing what plays the team will run in specific situations; these coaches also exhibit tendencies.

It is this duality—this structure within chaos—that makes football a worthwhile domain to study from a computer vision researcher's perspective. In particular, football's structure is both interesting and valuable. Teams at all levels from high school to professional football spend thousands of dollars and countless human work hours analyzing video of their own play and the play of their opponents. The goal of this significant investment of labor and capital is to uncover, analyze, and help create structure: to identify tendencies, to evaluate strategies, to develop plays and formations, and so forth, all with the hope of improving the team's performance. Because of the value of this information and because of the current state of the technology used to obtain it (nearly all of this work is currently done manually), there is a great opportunity to automate the annotation and analysis of football video and to develop methods to discover tendencies and other higher-level structure based on this annotation and analysis.

However, because of the chaotic nature of the game, the automatic analysis of football video is a very challenging computer vision problem. The complicating visual factors in football video are numerous. For example, in order to capture the game's action, the camera typically pans, tilts, and zooms rapidly to follow the evolution each play. In addition, because of the need to capture all the action in each play, the video teams use is often shot at a wide angle and hence captures players at rather low resolution. For this reason, players on the same team generally appear essentially identical in this video. Moreover, there is often a great deal of visual clutter and camouflage on the field in the form of large logos, numbers, yard lines, and other markings. Combine all this with the fact that players move erratically and interact with each other in very complex ways, and one can easily see why no successful system currently exists to perform the

automatic analysis of football video. On the other hand, these same factors make computer vision research on the American football problem quite useful, since methods that can achieve success in this domain will be tempered by fire, so to speak, and will typically be broadly applicable to many other visual domains.

With all this in mind, I set out in this thesis to make progress on designing computer vision methods for analyzing American football video. My work represents part of a longer-term project whose end goal is a complete vision system for understanding American football. In particular, my contributions to this system are four mid-level computer vision methods that accomplish specific sub-goals of the larger task of understanding football in video.

The first of these methods solves the problem of video registration, which is a necessary first step for solving nearly all later football video analysis goals. Video registration, specifically, is the process of transforming every frame of a video to align with a static coordinate system. In the case of American football, the coordinate system with which we wish to register is a 2D scale model of the football field. This is necessary for football analysis because, as mentioned above, football video is typically shot with a panning, tilting, and zooming camera, and thus, the raw image coordinates of objects within this video are not meaningful. For example, even a player who is standing still may appear to be moving in football video because of the motion of the camera. By registering this video with a model of the football field, we may instead reference object locations within the meaningful coordinate system of the football field. For example, we may say that a player is on the bottom hash at the home 35 yard line. This information is useful for later analysis tasks, such as recognizing initial player formations and tracking players, which I also address in this work.

The registration approach I describe in Chapter 2 uses invariant local image feature matching to compute transforms that map video frames to the model (i.e. the football field). However, because many frames of football video lack *globally* distinctive image features, which are the only features for which matches can typically be found using previous methods, I present a novel concept of *local* distinctiveness, in which features are matched only within a restricted spatial window. The concept of local distinctiveness can be employed to register football video even in the absence of globally distinctive image features. In addition, I outline a novel empirical stability test for registration transforms, which we use to help automatically initialize our registration procedure. I demonstrate through a set of detailed experiments on a data set of American football video that our registration method achieves results that are accurate to within only a few pixels, on average, and that it outperforms a benchmark method from the literature.

Second, I explore the problem of constructing a mosaic from a video collection, that is, automatically selecting a set of representative frames from that collection and registering them together in a common coordinate frame. In the American football domain, we use video mosaicing to automate tasks such as building reference image sets for our video registration procedure or for constructing background models for our initial formation recognition and player tracking algorithms.

Previous video mosaicing approaches looked only at the single-video mosaicing problem, and to our knowledge, we are the first to examine the problem of building mosaics from a *collection* of videos. Our multi-video mosaicing procedure, which I describe in Chapter 3, takes a utility maximization approach to achieve user-definable objectives for mosaic quality. Importantly, our method is efficient because it computes image features only in the handful of frames selected for inclusion into the final mosaic, whereas previous single-video methods computed image features in every video frame—a prohibitively expensive step when processing even a moderately-sized video collection. I also give an account of several experiments on both single videos and large video collections from American football and other domains showing that, using a simple utility function, our method can efficiently select a small subset of frames that achieve our stated goal of near-maximal scene coverage.

The third method I describe addresses the problem of recognizing initial player formations from the beginning of each football play. Initial player formations are a key piece of structure in a team's offensive strategy. In particular, not only does the initial formation set the team up to run a specific set of coordinated patterns (i.e. a play), it also forces the team's defensive opponent to line up in a way the play's designer hopes will introduce vulnerability into that opponent's defensive strategy (the same can be said for an initial defensive formation, as well). Thus, any useful system for football analysis must necessarily include formation recognition as one of its capabilities.

Automatically recognizing an initial formation entails determining both the types of players present on the field at the beginning of a play and the initial locations of those players. In Chapter 4, I describe an approach to this problem that views football formations as multi-part objects whose constituent parts are the players in the formation. Under this view, we employ a novel framework that extends the well-known pictorial structure model's ability to recognize object classes based on both the appearance and relative structure of their constituent parts to give that model the ability to recognize object classes whose parts can vary in both type and location (whereas classical pictorial structures assume a set of parts with fixed types). This

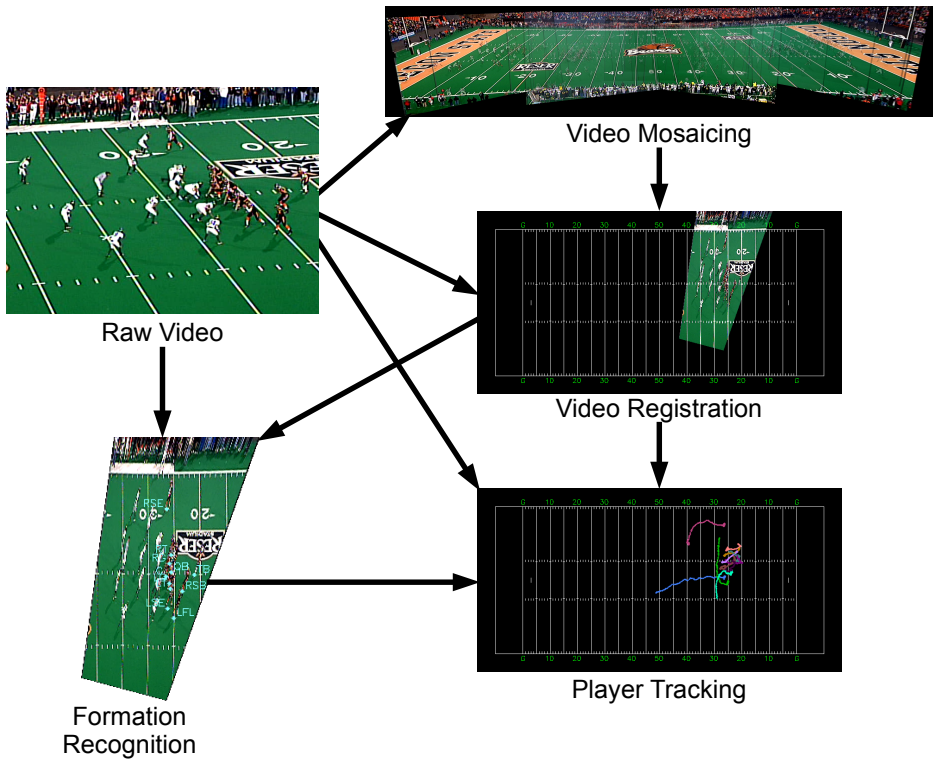


Figure 1.1: The architecture of the four football video analysis procedures outlined in this thesis. The results of the formation recognition and player tracking steps will be passed on to later stages of a larger football understanding system.

extended model is called the mixture-of-parts pictorial structure model (MoPPS). I further describe a restricted but reasonable MoPPS representation that allows for efficient inference using a branch-and-bound search procedure, and I present experimental results from the initial formation recognition problem in the American football domain to show that our MoPPS representation is both efficient and highly accurate.

Finally, I investigate the problem of tracking players through American football video (that is, determining their spatial trajectories through the video). Again, player trajectories are a key piece of the valuable structure we wish to unearth in football video because, once player trajectories are known, we can determine what specific plays a team runs (or attempts to run). This, in turn, opens up even further opportunities for analysis. However, player tracking is the likely the single problem in which football's chaos most manifests itself as an obstacle to

analysis. More simply put, for all the reasons discussed above, American football is one of the most challenging object tracking domains being studied today.

In Chapter 5, I describe a tracking method designed to attempt to overcome football's chaos. This method is based on the well known particle filter, but it extends the classical particle filtering framework in several key ways. First and foremost, we reformulate the particle filter as an undirected, log-linear probabilistic model (in contrast to the traditional directed, generative probabilistic model that is typically used). This formulation allows the user to define rich sets of (arbitrary) features that capture joint properties of a tracked object's state and the observations and is strictly more expressive than the classical generative model. Moreover, we describe a discriminative supervised training procedure for this new model (as well as an efficient approximation to this procedure) that attempts to learn an optimal relative weighting of the features based on observed errors during tracking. I also present a thorough set of experiments on American football video in which our particle filter is shown to outperform previous state-of-the art filtering approaches.

The overall architecture of the four procedures I describe in this thesis is depicted in Figure 1.1.

I conclude in Chapter 6 by discussing our more recent, preliminary work on player tracking in the football domain. Before entering into the rest of this thesis, however, I will review some of the recent literature that is most relevant to the problem of football video analysis.

1.1 Previous Related Work

Despite its research appeal, we are aware of very little work addressing the larger problem of analyzing American football video. One substantial earlier body of work on American football is by Intille and Bobick and is summarized in Intille's 1999 PhD thesis [31]. Indeed, though Intille and Bobick's work may be the most comprehensive to date on the football analysis problem, it might best be thought of as a series of interesting failures. To a large extent, this can be blamed on the fact that the state of the art in 1999 was simply not advanced enough to tackle many of the challenges set forth by the American football domain. In particular, at that time, technologies that have driven much of the success of our own research, such as invariant local image features, pictorial structure models, and particle filter-based object trackers, were either undiscovered or else obscure and little understood.

Indeed, Intille and Bobick attempted to solve many of the same problems addressed in this

thesis, but invariably, the solutions they proposed were unreliable to the point that their respective results could not be used in successive stages of their football understanding system. For example, in [30], Intille proposed a video registration method that tried to detect and track intersections of lines on the football field, a subset of which were then manually matched to the corresponding locations on a 2D football field model in order to initialize the computation of registration transforms for each video frame. However, this method proved ineffective and was abandoned in favor of tedious manual registration in Intille's thesis.

The main problem with this approach is its dependence on line intersections whose accurate detection is inherently unreliable, considering that a) most football fields are not, in fact, planar but are curved for drainage, so the yard lines on the field (i.e. the lines placed at five-yard intervals that span the width of the field) do not actually project as true lines in video; and b) there is not actually a set of lines that runs perpendicular to the yard lines on a football field, so Intille tried to induce lines perpendicular to the yard lines using the centroids of the hash marks (i.e. the series of short parallel lines that are placed at one-yard intervals). Ultimately, because of these factors, the noise inherent in the low-level image processing algorithms that underlay the detection of the intersections of these lines (and "lines") introduced unacceptable levels of error into their tracking and matching, which in turn led to unacceptable registration results.

Intille also attempted in his thesis to solve the problem of recognizing initial player formations using a SAT-like algorithm and a manually-constructed knowledge base of hard constraints—such as "near," "to the left of," and "bit of vertical space between"—on the relationships between player locations. However, not only did this approach require the manual input of players' locations in the initial formation—it's output was simply a set of player type tags associated with the input locations—but the results Intille achieved using it were again too poor to pass on to later stages of his football video analysis system.

The shortcomings of this approach were numerous. For instance, a tremendous amount of manual effort was required to make this approach work. In particular, not only did it need all initial player locations to be manually specified, it also required a large, manually-assembled knowledge base of hard constraints, the construction of which Intille himself admitted to be tedious and time-consuming. Perhaps most importantly, the use of hard constraints makes this approach quite inflexible. For example, because even identical formations typically exhibit slight differences in the relative locations of players, a database of discrete hard constraints engineered to operate on a particular data set of formations might have trouble recognizing new instances of the same formations. Moreover, incorporating new, previously-unseen formations into the

database would likely require hours of tedious constraint re-engineering.

Finally, much of Intille and Bobick's early efforts were devoted to player tracking [30, 32, 33]. The common thread running through these three tracking papers is the concept of "closed-world" tracking, in which small regions (i.e. closed worlds) are drawn around independent single players and interacting groups of players in the current video frame, appearance templates are computed for the players in each closed world (with the number and identities of the players in each closed world being used to determine how the templates are computed), and these templates are matched with the succeeding frame to determine each player's location there. It is now well known, however, that template-based tracking can be very unreliable, particularly for non-rigid objects like football players, and, primarily for this reason, Intille and Bobick's tracking results were never satisfactory for their later work on football play recognition. Thus, again, for this later stage of their football analysis system, Intille and Bobick were forced to use manually generated tracking data.

Still, despite much of their system's inability to produce reliable results, Intille and Bobick's work is still instructive. For example, the overall architecture we have employed in our own system to date has aligned very closely with the architecture designed by Intille and Bobick (i.e. video registration $\rightarrow$ initial formation recognition $\rightarrow$ player tracking $\rightarrow$ play recognition). In addition, some aspects of the individual components of Intille and Bobick's system—such as the computation of registration transforms from video-to-model point matches—are present in the corresponding components of our own system.

In addition to Intille and Bobick's work, there has been a good deal of more recent work on American football video. Liu *et al.* investigate the problem of extracting the portions of a television football broadcast that contain football play action and discarding non-action portions of the broadcast, such as commercials, etc. [48]. This work may have limited relevance to us because we are interested in working with pre-existing databases of video recorded by football teams themselves. This video is typically pre-segmented into single-play clips by the camera operator.

Several other efforts employ various means to address the problem of football play classification from video [72, 46, 67, 45]. Indeed, each of these papers attempts to perform essentially the same very coarse classification of football plays (i.e. left/right/middle run, or short/deep pass). Though we may wish to perform a more fine-grained analysis of football plays than is done in this work (e.g. recognizing specific passing patterns or blocking schemes), some of it may still be useful as we move to later stages of our own work. However, none of this work does much

to address the computer vision problems considered in this thesis. In particular, none of these papers explore the problems of video registration or initial player formation recognition, and two of them ([67, 45]) do not make use of any player tracking data at all, while one ([46]) uses only manually generated tracking data, and the other ([72]) resigns to using data from only partial, inexact trackers. It is important to note that, without accurate tracking data, it may not be possible to achieve some of the finer-grained play analysis we envision in later stages of our own system.

In addition to work on the American footall domain, there have been a number of research efforts on analyzing video from other sports domains that are worth noting here. For example, a research group led by James Little at the University of British Columbia is currently performing work on analyzing ice hockey video and has addressed the problems of video registration [61], player tracking [51, 7, 50, 62], and player action recognition [51, 50]. While most of this work is satisfactory, it still exhibits shortcomings that make it inappropriate for our efforts in the American football domain. For example, the video registration method Little's group proposes requires manual initialization of every video clip, which places an unnecessary burden on the end user. Little's group's tracking method, the boosted particle filter, is well known, but it does not provide mechanisms for reasoning about player types or identities, which is important in the football domain, since player type information often helps us cope with the ambiguity inherent in football video. Finally, Little's group focuses only on classifying individual player actions at a very coarse level (e.g. skating left/right or skating up/down). While action classifications of this type may be of some use in the football domain, a great deal of higher-level activity recognition will also be necessary for football.

A great deal of research work has also been done on soccer analysis, with a primary focus on player tracking. For example, Sullivan *et al.* propose a tracklet-based approach for tracking and identifying soccer players [57, 71]. Unfortunately this method relies on a panoramic video stream of the soccer pitch recorded by calibrated cameras to facilitate near-perfect background subtraction. Because we wish to design a system to work with the video football teams already use, this is simply not possible for us. Moreover, Sullivan *et al.*'s method uses a very simple nearest-neighbor data association algorithm for linking players from frame to frame, and this approach is far too simplistic to cope with the complex interactions between American football players.

A group of researchers from Disney Research and Georgia Tech have also explored soccer player tracking [22]. However, their tracking method assumes the action is being recorded by multiple cameras at different locations, and this assumption does not hold in our setting in

the football domain. This same research group also investigated a method for predicting the evolution of a soccer play, i.e. predicting the region on the pitch towards which the play is unfolding [39]. This method uses the tracks output by the method described in [22] to compute a motion field over the entire soccer pitch, and this motion field is then used to detect points of convergence. This method, while not directly relevant to the problems addressed in this thesis, is interesting, and it could be useful in the American football domain, for example, to coarsely classify the type of play being run (e.g. left/right run or short/deep pass) or to determine which player is carrying the ball.

Finally, Lu *et al.* recently presented a method for determining the identities of players in basketball video. This method learns an appearance-based classifier over player identities using SIFT and MSER image features and color histograms and then applies this classifier at test time on player regions determined using a player detection-based tracker. The role played by this procedure is essentially analogous to the one played by an initial player formation recognition procedure in the American football domain. However, because this method relies so heavily on the ability to distinguish players based on features of their appearance, it is not suitable for our American football video, in which the resolution is so low that all players on the same team have virtually indistinguishable appearances.

Before moving on, I will note that there is, of course, other work on sports analysis in the literature, but, aside from some additional research endeavors discussed in the chapters that follow, the ones mentioned here are the most salient for our work on American football video analysis.

Chapter 2: Video Registration[1]

An important first step in any system for interpreting American football video is to register the video with a the static coordinates of the football field. This is necessary because football video is typically recorded with a camera that rapidly pans and zooms to follow the action of the game, causing even a physically stationary player to appear to be moving as the video progresses and rendering raw player locations and trajectories meaningless from an interpretation standpoint.

Registering video frames with a static model is a common problem in many other computer vision domains as well, including robot localization [66, 40], augmented reality [21, 68], analysis of other sports[30, 61], and others [6]. In general, video registration is required whenever we need to know what part of an object or scene a video frame depicts or where an object in that frame is located relative to a fixed coordinate system.

The standard approach to the registration problem is to compute, for each frame in the video sequence, a set of point correspondences between that frame and the model. These correspondences are then used to numerically determine a registration transform that maps the video frame to the model. The problem of finding such sets of correspondences was investigated specifically within the American football domain by Intille [30], who hypothesized that since the football field is (approximately) planar, the registration of football video with a 2D football field model can be achieved by computing a planar homography mapping the video field surface to the model. A planar homography, which maps one plane to another, is a linear transform with eight degrees of freedom and can be computed from four or more 2D point correspondences [23]. Intille's approach to finding these correspondences involved locating, classifying and tracking line intersections on the field. Unfortunately, this method lacks generality, since many domains do not have such a precisely structured set of high level features as the lines on a football field. More importantly, because of the difficulty inherent in consistently detecting such high-level features, Intille's method proved to be unreliable and was abandoned in later work [31] in favor of tedious manual registration. In a set of informal experiments, we also found Intille's method to be ineffective, and we are unaware of any other successful demonstrations of robust registration of American football video.

[1]The work described in this chapter was published in [26].

Modern approaches to registration have taken advantage of recent breakthroughs in the detection [53] and description [55] of transform-invariant, local image features which are designed to facilitate consistent detection and easy, efficient matching between images. Using local feature techniques, the registration problem can be solved by assembling a set of reference images to represent the model and then detecting and matching local features between the model images and the video. [21], [66], and [40] are all examples of this type of approach.

Compared to Intille's method, local feature-based registration is attractive because of its generality and the proven robustness of finding reliable matches between distinctive local image features. Current local feature-based registration methods work well in domains with an ample supply of distinctive local features. It is important to note, though, that most current local feature-based methods rely *solely* on the presence of distinctive visual features for registration. Unfortunately, many domains can produce long segments of video without enough distinctive local features to robustly compute registration transforms, though there may still be many informative but non-distinctive features. In these domains, as we demonstrate in Section 2.2, relying completely on the presence of distinctive visual features for registration can result in crippling inaccuracy.

The American football domain is a prime example of one in which total reliance on the presence of *distinctive* visual features can prove to be disastrous. Here, important, distinctive visual features can be found at certain locations, such as within logos and around numbers on the field, but large regions of the field also exist that contain either no distinctive visual features at all or only a very small number of them. Often, video frames from the football domain depict only these latter regions of the field, making registration via distinctive feature matching either impossible or extremely unreliable. However, in video from the football domain, we can almost always guarantee the presence of *some* visual features, though they might be *non*-distinctive ones. For example, sets of identical hash marks, depicted in Figure 2.1, span the length of the football field, spaced one every yard. Such non-distinctive features convey a great deal of information about location on the field, and the ability to correctly match them to their corresponding model features would allow for robust computation of registration transforms. However, because these features are identical in appearance, they cannot be matched using common distinctive feature matching techniques.

The main contribution of this chapter is to develop a generic registration approach that can leverage modern invariant feature techniques in domains like American football, where distinctive image features are often scarce but non-distinctive features are plentiful. Our method, which

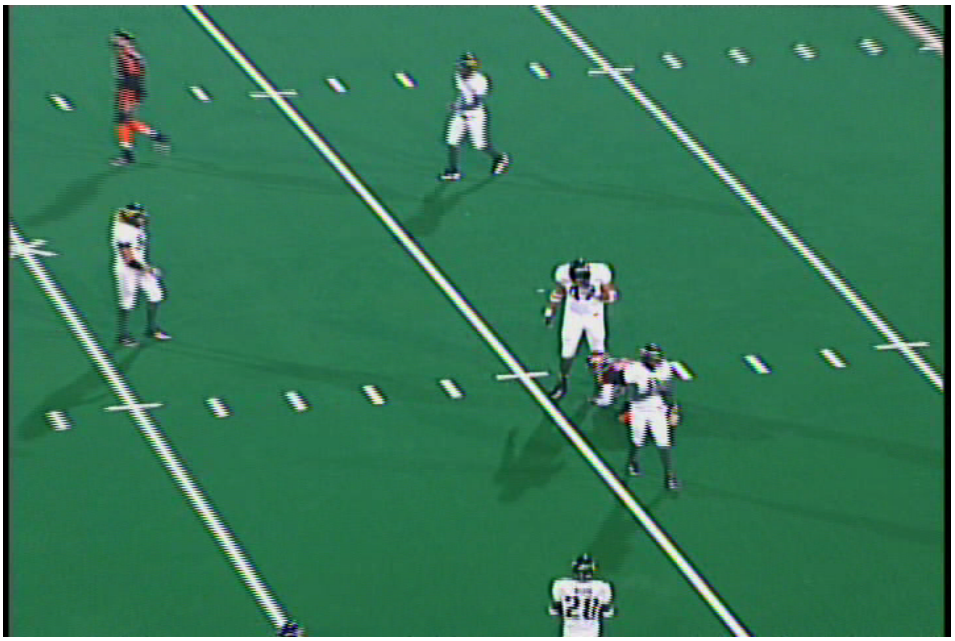


Figure 2.1: Some domains, such as the American football domain, shown here, produce images without distinctive visual features that can be easily matched. However, in these domains, the presence of *non*-distinctive visual features, such as the hash marks in the image above, can almost always be guaranteed.

is discussed in detail in Section 2.1, takes advantage of such non-distinctive visual features to register video, even in the absence of distinctive features. Specifically, we introduce a concept of *local* distinctiveness that enables us to find model matches for nearly all visual features in every video frame. In addition, we present a simple, empirical stability test that allows us to find a stable set of distinctive features with which to initialize the registration process, resulting in fully automatic registration.

Our approach is most similar in spirit to recent work by Okuma *et al.* [61]. Their approach avoids relying on distinctive features by utilizing generic point correspondences computed using the Kanade-Lucas-Tomasi tracking equation [74] along with edge-based model fitting. One major drawback of this approach is that it requires manual initialization for every video sequence to be registered. This can be cumbersome if a large collection of video must be processed, as is the case in our application domain. While Okuma *et al.*'s method is conceptually similar to the method we describe in this chapter, our use of invariant image features, in conjunction with our

initialization technique, allows for fully automatic operation.

Our empirical evaluation in Section 2.2, shows that, compared to distinctive feature-based approaches, our method is very effective in the challenging American football domain. We also note that a secondary contribution of the work described in this chapter is our substantial ground truth video dataset, which we hope can be used as a standard benchmarking tool for video registration methods.

2.1 Method

Our method registers a video sequence with a predefined, static model by finding point correspondences between the video and the model and using them to compute a registration transform for each frame. Under our method, video-to-model point correspondences are found by matching invariant image features in the video to a set of features assembled from reference images to represent the model. In what follows, we describe how invariant image features are detected in the video and reference images; how the set of model features is assembled; how distinctive image features are matched to form video-to-model point correspondences; how these correspondences are used to compute registration transforms; and how accurate transforms can be computed, even in the absence of distinctive image features, by finding matches between *non*-distinctive image features using a concept of local distinctiveness.

2.1.1 Detecting Invariant Image Features

In this chapter, we use the Harris-affine detector [54] and SIFT descriptor [49] to detect and describe image features. Given an image as input, the Harris-affine/SIFT operator computes a set of feature points, each represented by a set of parameters describing the affine region surrounding the feature as well as a 128-dimensional descriptor vector. These features are invariant in that, in theory, the same ones will be detected in each of two images of the same object related by a reasonable degree of affine transformation, including translation, scale, in-plane rotation, and, to a limited extent, out-of-plane rotation. In addition, corresponding features in the two images will have very similar descriptors.

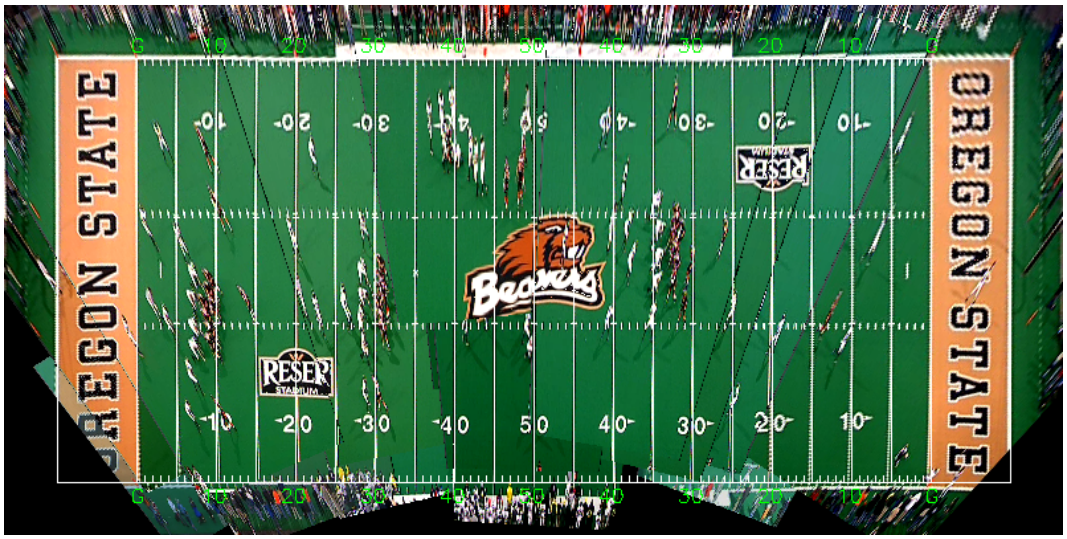


Figure 2.2: A set of reference images from the American football domain is registered with an overhead view of the field. Registering the set of reference images in this manner allows us to know the field coordinates of image features in the reference images and so to localize video frames on the field via feature matching.

2.1.2 Assembling a Set of Model Features

There are several possible ways to form the set model features, denoted below as Π. Our goal is to do so in such a way that the model coordinates of the features in Π are known, thereby allowing us to determine the model coordinates of video features via feature matching.

In some domains, where the locations of model features are only important in relation to each other, Π can be formed simply and automatically by iteratively registering a set of reference images to each other [6, 21]. In other domains, however, the locations of features in Π must be known in reference to a specific global coordinate frame, such as a particular view of the model. In the American football domain, for example, we want to know the field coordinates (e.g. bottom hash on the home 35-yard line) of each model feature so that video frames can be localized on the field and not just registered to an arbitrary view of it. This is achieved by registering the set of reference images to a known view of the field, as depicted in Figure 2.2.

It is, in general, difficult to automatically register a set of reference images with specific coordinate frame in a domain-independent manner. Fortunately, the amount of manual work required to do so is minimal. For each reference image in the football domain, for example,

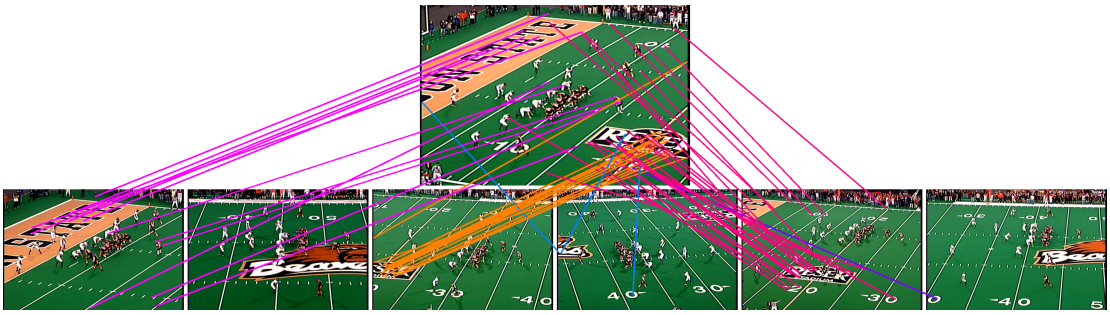


Figure 2.3: Image features from a video frame in the American football domain are matched using the 2NN heuristic to a portion of the model shown in Figure 2.2. There are very few false positive and many true positive ones. Most true positive matches, however are between distinctive image features, such as the field logo, with few correct matches between non-distinctive ones, such as the hash marks.

we must specify just one set of four point correspondences to compute a planar homography mapping that image to the desired view of the field, as was done to generate the model in Figure 2.2. It is also possible to automatically register the set of reference images with each other, as when we are not aligning them with a specific coordinate frame, and thus to reduce the number of manually specified point correspondences to a single set for all reference images instead of one set per reference image.

Note that, in the work described in this chapter, we use a manually constructed set of reference images. However, in Chapter 3, we describe a method for efficiently and automatically constructing a set of reference images from a video collection.

2.1.3 Matching Image Features

In practice, it is not always the case that two images of the same object will produce all of the same image features, nor is it true that two descriptors belonging to matching features will be identical. Therefore, it is necessary to have some way to compute feature matches in which we can be highly confident. To do so, we make use of the 2NN heuristic proposed by Lowe in [49]. Given a feature X from video, we find from the set Π of model features X's two nearest neighbors, $\pi_1(X)$ and $\pi_2(X)$, with respect to the Euclidean distance between descriptor vectors. The 2NN heuristic considers X and $\pi_1(X)$ to be a distinctive match if, for a fixed threshold

$\rho \in [0, 1]$,

$$\frac{\|d(X) - d(\pi_1(X))\|}{\|d(X) - d(\pi_2(X))\|} < \rho, \tag{2.1}$$

where $d(X)$ is the descriptor vector of feature X and $\| \cdot \|$ is the Euclidean norm. If (2.1) is not satisfied, X remains unmatched, even if X and $\pi_1(X)$ are, in fact, matching features. Figure 2.3 shows the results of matching features from a frame of football video to part of the model in Figure 2.2 using the 2NN heuristic.

By choosing ρ appropriately (we use $\rho = 0.6$), the 2NN heuristic yields a very small number of false positive matches. An unfortunate side-effect of this heuristic, however, is that it only finds matches between features whose descriptors are very different from those of the rest of the set of potential matching features. We call these features *globally distinctive*. The 2NN heuristic is generally incapable of establishing correspondences for features whose correct match in Π has a descriptor that is similar to many others in Π. Indeed, this is the case with any heuristic that attempts to minimize false positive matches while using only local context information for each feature. Unfortunately, as discussed above and as quantified in Section 2.2, relying solely on correspondences from globally distinctive features can impair our ability to successfully register video from some domains, such as American football. In Section 2.1.5, we discuss a method to overcome this difficulty, but first we describe briefly how registration transforms are computed from video-to-model correspondences.

2.1.4 Computing Registration Transforms from Sets of Point Correspondences

Having constructed a set of model features and found a set of correspondences between each video frame and the model, it is possible to compute a registration transform for each frame. In the football domain, we can analytically compute homographies from four or more correspondences via least squares. Specifically, given a set of $n \geq 4$ video-to-model correspondences $\left((x_v^i, y_v^i), (x_m^i, y_m^i)\right)_{i=1}^n$, where the (x_v^i, y_v^i) are image coordinates in the video frame and the (x_m^i, y_m^i) are the corresponding model coordinates, a least squares planar homography can be

computed by forming and solving the following linear system:

$$\begin{bmatrix} x_v^1 & y_v^1 & 1 & 0 & 0 & 0 & -x_m^1 x_v^1 & -x_m^1 y_v^1 \\ 0 & 0 & 0 & x_v^1 & y_v^1 & 1 & -y_m^1 x_v^1 & -y_m^1 y_v^1 \\ & \vdots & & & \vdots & & \vdots & \vdots \\ x_v^n & y_v^n & 1 & 0 & 0 & 0 & -x_m^n x_v^n & -x_m^n y_v^n \\ 0 & 0 & 0 & x_v^n & y_v^n & 1 & -y_m^n x_v^n & -y_m^n y_v^n \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \end{bmatrix} = \begin{bmatrix} x_m^1 \\ y_m^1 \\ \vdots \\ x_m^n \\ y_m^n \end{bmatrix}. \tag{2.2}$$

Here, the h_{ij} are the entries of the homography matrix. Because the homography is defined up to a scale factor, we may choose $h_{33} = 1$ [23]. We note that an alternative method for homography computation is the direct linear transform (DLT), which can handle the special case where $h_{33} = 0$. See [23] for a complete discussion of this topic.

Unfortunately, both least squares and DLT are very sensitive to outliers in the set of correspondences. In order to cope with the unavoidable presence of false correspondences, we use RANSAC [19] in conjunction with least squares to find a consistent set of inlier correspondences and a corresponding registration transform for each frame. We refer to the set of inliers as a frame's "core set", since it is comprised of correspondences in whose verity we are highly confident.

2.1.5 Using Local Distinctiveness to Find Additional Correspondences

The approach taken by current registration methods is to compute registration transforms using the procedure described between sections 2.1.1 and 2.1.4 (or some slight variation of it), using only correspondences between globally distinctive features found with the 2NN heuristic. As discussed above, and as demonstrated in section 2.2, this approach fares poorly in domains where many frames lack globally distinctive image features. Our method attempts to maintain registration through a sequence of these frames by inducing correspondences between globally non-distinctive features using a concept of local distinctiveness. Specifically, we say that a feature X is *locally distinctive* relative to a spatial region R in a model or image if it passes a

spatially restricted 2NN test,

$$\frac{\|d(X) - d(\pi_1^R(X))\|}{\|d(X) - d(\pi_2^R(X))\|} < \rho, \tag{2.3}$$

where $\pi_i^R(X)$ is the i^{th} nearest neighbor of feature X within region R of the model or image. Note that even if X is not globally distinctive it can be locally distinctive relative to a particular R. If the region R can be selected so that it is likely to contain a correct match for X, then a locally distinctive match is likely to be a correct one.

Matching via local distinctiveness plays two roles in our registration method. The first is to track image features between frames. Because video is sampled at very high rates—typically around 30 frames per second—the amount of change between any two consecutive frames is very small, and a given image feature is likely to move at most only a few pixels between those frames. We can take advantage of this fact by searching for a feature's locally distinctive match in the next frame within a small region R around the feature's spatial location in the current frame. By doing so, we essentially ensure our ability to find a correct match for that feature. The utility here lies in the fact that if a feature tracked to the current frame has previously been matched to the model, then that model match can be effectively carried over to the current frame. If the current frame does not have a sufficient set of globally distinctive matches, then these additional tracked matches, many of which may not be globally distinctive, can help produce a stable registration transform.

The second role of matching via local distinctiveness is to find new model matches for non-distinctive features. Specifically, if we can assume that we have found a core set of feature correspondences for the current frame that is sufficient for computing an accurate registration transform, we can use that transform to search for a locally distinctive model match for any unmatched feature X relative to a small region R around X's predicted model location. This is useful because it allows us to compute model correspondences for non-globally distinctive features whenever they appear in a video. These new matches can then be propagated through the video using the above tracking approach.

Our overall registration procedure uses the above two applications of matching via local distinctiveness as follows.

1. **Initialize.** Mark all frames as unprocessed and uninitialized. Select a frame for which the set of correspondences from globally distinctive features results in the "most stable" registration transform (see next section) after applying RANSAC. Initialize the core set of

this frame to be the set of globally distinctive matches, and mark this frame as initialized.

2. **Include Globally Distinctive Features.** Select an unprocessed, initialized frame. Add all correspondences from globally distinctive features in the frame to its initial core set and use RANSAC on the entire set to compute a new expanded core set and its associated registration transform. Note that this step will not affect the core set of the initial frame selected in step 1.
3. **Include Unmatched Features.** For the selected frame, compute, as discussed above, a set of correspondences from features that are locally distinctive relative to the frame's registration transform. Union these correspondences with the current core set and use RANSAC to compute a final core set and the associated final registration transform for the frame. Mark this frame as processed.
4. **Model Match Propagation.** For each neighboring frame of the selected frame that has not been processed (either 1 or 2 frames), use the approach described above to attempt to track each of the features in the core set to the neighbor, and propagate forward the model matches of successfully tracked features. Initialize the neighbor's core set to the set of correspondences determined by the propagated matches, and mark the neighbor as initialized.
5. **Loop.** If unprocessed frames remain, go to step 2.

This approach allows us to maintain a large core set of video-to-model correspondences and to add new correspondences to that set as new features appear in the video frame. In this way, as long as there are enough good features—either globally or locally distinctive—in the video frame to produce an accurate registration transform, the core set, once formed, is self-sustaining, since an accurate registration transform allows us to find model correspondences for all video features matchable via local distinctiveness.

All that remains then is to determine, in step 1, which frame to select as the initial frame to process. This should be a frame for which we are most certain that the set of correspondences from globally distinctive features is sufficient for computing an accurate registration transform. It is, in general, dangerous to assume that the first frame of video will always be such a frame. We therefore make the assumption that at least one such frame exists in the video and develop a test for finding one of them. This test is presented in the next section. If the single-good-frame assumption does not hold, we can revert to manual initialization.

Algorithm 1 Stability Test for Core Set Initialization

Input: $\mathbf{C}$: Potential initial core set
$\mathbf{K}$: User defined number of iterations
Output: $\mathbf{S}$: Output stability

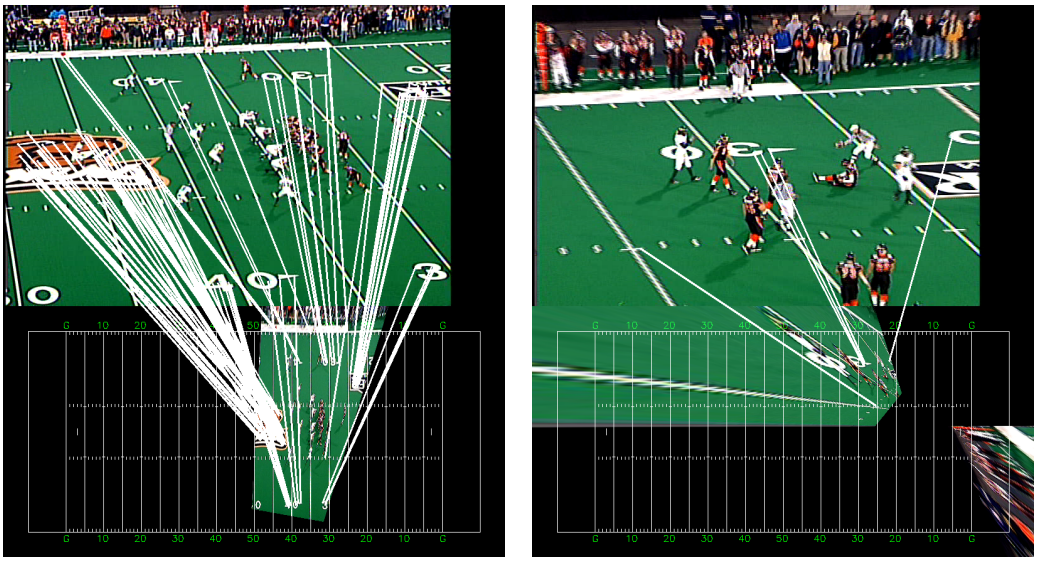
$\mathbf{T} \leftarrow$ Registration transform from $\mathbf{C}$
$\mathbf{L} \leftarrow$ Set of randomly sampled image locations
$\mathbf{L_T} \leftarrow$ Model coordinates of $\mathbf{L}$ via $\mathbf{T}$
$\mathbf{S} \leftarrow 0$
for $i \leftarrow 1..\mathbf{K}$ **do**
  $\widehat{\mathbf{C}} \leftarrow \mathbf{C}$ perturbed with $\mathcal{N}(0, \epsilon)$ noise
  $\widehat{\mathbf{T}} \leftarrow$ Registration transform from $\widehat{\mathbf{C}}$
  $\mathbf{L}_{\widehat{\mathbf{T}}} \leftarrow$ Model coordinates of $\mathbf{L}$ via $\widehat{\mathbf{T}}$
  $\mathbf{S} \leftarrow \mathbf{S} + \text{ERROR}(\mathbf{L}_{\widehat{\mathbf{T}}}, \mathbf{L_T})$
end for

2.1.6 Stability Test for Core Set Initialization

The stability of a set of correspondences can be computed analytically by forming the least squares system in (2.2) and measuring its conditioning using techniques from existing theory on the conditioning of least squares problems. This theory, discussed at length in [76], provides bounds on the error amplification factor that ensues from small perturbations in the input. Unfortunately, we have found that, in practice, this analytical approach often yields a poor choice of initial core set. In turn, we have developed an empirical stability test, which is outlined in Algorithm 1.

In general, sets of correspondences that produce stable transforms are large and widely distributed spatially. Sets of low cardinality whose correspondences are not well distributed, on the other hand, are very sensitive to small amounts of noise. Figure 2.4 helps to elucidate the difference between these two types of sets. Intuitively, our stability test identifies those sets of correspondences that are the most invulnerable to small amounts of noise.

The value $\mathbf{S}$ in Algorithm 1 represents a measure of how drastically the predicted model locations change with slight perturbations in the input set of correspondences. Sets that result in very low values of $\mathbf{S}$ are least sensitive to measurement noise and produce the most stable transforms. We initialize the registration process using the frame whose set of correspondences from globally distinctive features yields the lowest value of $\mathbf{S}$.



(a) $\mathbf{S} = 1.12 \times 10^2$ (b) $\mathbf{S} = 1.60 \times 10^7$

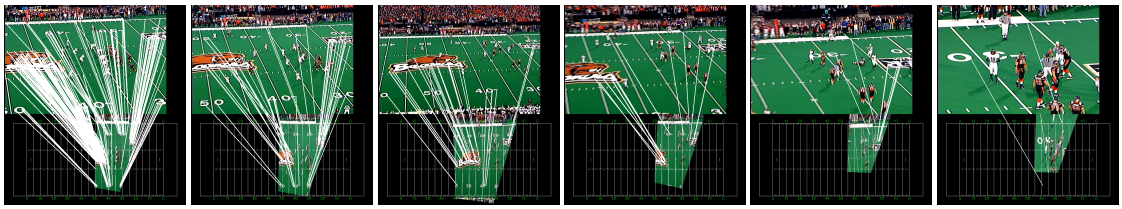
Figure 2.4: The large and widely distributed set of correspondences in (a) yields an acceptable registration transform, but the small, isolated set in (b) yields one that is worthless. The quality of each set as a core set initializer is reflected by the value **S** returned by the stability test outlined in Algorithm 1.

2.2 Experiments

We tested our method on a set of 25 video sequences from the American football domain[2], each between 280 and 500 frames in length. Sequences in this data set were selected from two different games to cover as much of the field surface as possible. Most of these sequences contain a significant number of frames without distinctive field features. However, almost all frames contain *some* field features, such as the hash marks depicted in Figure 2.1, for which a model match exists. Every tenth frame of every video in our data set has an associated set of hand-labeled ground truth video-to-model point correspondences. There are between 300 and 700 such hand-labeled correspondences for each video, for a total of about 12,000.

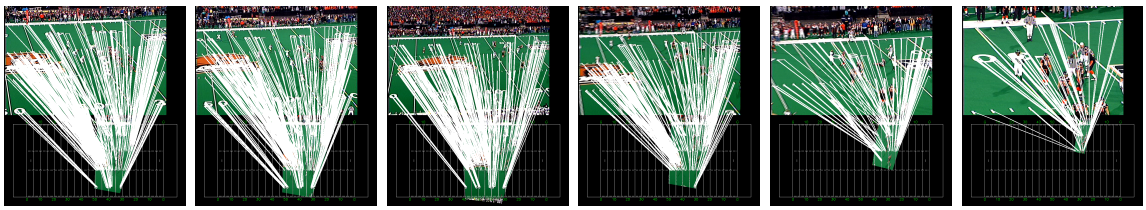
Our model was constructed from a set of 23 reference images as described in section 2.1.2 and as depicted in Figure 2.2 (note again that, while this reference set was constructed manually, we describe in Chapter 3 a method for doing this automatically). Reference images were selected

[2]Our dataset is available online at `http://eecs.oregonstate.edu/football/registration/dataset`.



Frame 1	Frame 150	Frame 250	Frame 300	Frame 325	Frame 407
232 Correspondences	50 Correspondences	31 Correspondences	15 Correspondences	10 Correspondences	2 Correspondences

(a) Registration with NR



Frame 1	Frame 150	Frame 250	Frame 300	Frame 325	Frame 407
525 Correspondences	480 Correspondences	390 Correspondences	333 Correspondences	210 Correspondences	125 Correspondences

(b) Registration with RIMLD

Figure 2.5: The sequences above illustrate typical registration results on a 407 frame video from the American football domain. For this video, NR maintains accuracy as long as the frame contains a stable set of correspondences from globally distinctive features. As early as frame 250, this set loses stability, and registration becomes slightly inaccurate. By frame 325, the set of correspondences, down to 10 and concentrated within a small region of the image, yields a transform that fails the sanity check, and registration reverts to the last good transform. On the same video, RIMLD maintains a stable set of at least 100 video-to-model correspondences throughout the run, and registration is accurate until the end of the video.

from video not included in the dataset unless achieving total field cover in the reference set required us to select a frame from the dataset.

For comparison, we tested two other methods using the same dataset and model. The first, which we call *naïve registration* (NR), uses only correspondences between globally distinctive features found using the 2NN heuristic (2.1). The second, *registration with uninitialized matching via local distinctiveness* (RUMLD), uses matching via local distinctiveness as described in Section 2.1.5 but always initializes the core set using the first frame instead of using the initialization technique described in Section 2.1.6. Our complete method is referred to below as *registration with initialized matching via local distinctiveness* (RIMLD). All three methods in-

clude a sanity check that reverts to the last good registration transform if the current frame's transform becomes grossly unacceptable.

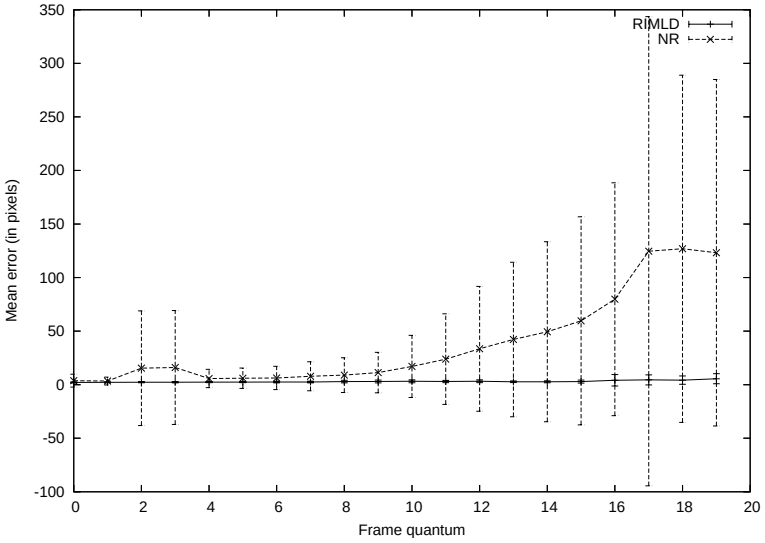
Figure 2.5 illustrates some registration results from NR and RIMLD that are representative for the American football domain. As is typically the case, both methods are accurate at the beginning of the video, where there is generally a large set of globally distinctive image features in the frame. However, as the video progresses, the set of correspondences used by NR to compute registration transforms gradually dwindles to the point of instability. At this point transforms computed by NR fail the sanity check, and NR reverts to the last known good transform, resulting in registration error that snowballs as the video proceeds to the end. RIMLD, on the other hand, maintains a large, stable set of correspondences throughout the length of the video, and registration is accurate to the end.

Registration accuracy for all three methods was quantified by computing the mean registration error for every tenth frame of every video in the dataset using the set of hand-labeled ground truth correspondences described above. Results were normalized to equal length by partitioning them into twenty quantums, where each quantum in a single video contains the same number of frames, and both the mean and maximum errors were computed for each quantum.

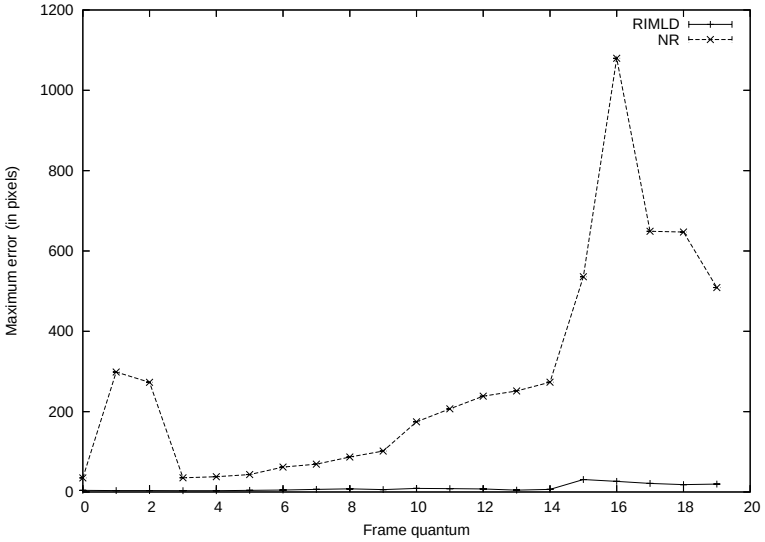
Interestingly, the results for RIMLD and RUMLD differ significantly on only a single video sequence. For this video, as might be expected, RUMLD's error rate is quite high at the beginning of the video, but it quickly reduces to nearly equal that of RIMLD after the core set has been expanded through a combination of locally and globally distinctive features. The similar performance of RIMLD and RUMLD can be explained by the fact that, with this one exception, the first frame of every video sequence in the dataset contains a large portion of one or more of the field logos or the end zones, where there are many distinctive features. Accordingly, these frames produce a stable enough set of correspondences with which to initialize registration.

The results for RIMLD and NR, however, *do* differ significantly on nearly all videos in the dataset. The error for these two methods over the entire dataset is summarized in Figure 2.6. To put these error rates into perspective, we note that six pixels in our model are equal to one yard on the football field. This means that, even during important parts of the football play, NR's average error rate approaches 10 yards—quite significant if these results are to be used in an interpretation system—while RIMLD maintains an average error of around one half yard.

The reason RIMLD is so much more accurate than NR is because RIMLD is able to maintain a model match for nearly every visual feature in the video frame for which a model match exists. Even at the end of a video sequence, the core set often contains on the order of one hundred or



(a) Mean registration error per frame quantum



(b) Maximum registration error per frame quantum

Figure 2.6: The above two plots depict (a) the mean and (b) the maximum registration error per frame quantum for RIMLD and NR over the entire dataset of 25 video sequences. The error bars in (a) indicate one standard deviation. By both measures, RIMLD is remarkably more accurate than NR, especially later in the video, when many frames contain few or no globally distinctive features. Note that six pixels in our model are equal to one yard on the football field.

more well distributed correspondences. By comparison, the set of correspondences determined by NR often shrinks to between 25 and 50, or even less, as early as halfway through the video, dwindling soon afterwards to ten or less. If such a small set of correspondences is not widely distributed, a small amount of error in either the video or model coordinates of the image features can become dramatically exaggerated in the resulting registration transform.

Besides raw registration error, another important gauge of registration quality is "smoothness." Because the appearance of a physical feature changes slightly in the video as time progresses, its feature descriptor changes also, and correspondences with a 2NN heuristic value near the threshold ρ may step back and forth over that threshold between frames. As the composition of the set of correspondences from globally distinctive features changes thus from frame to frame, registration with NR is prone to jitter. Because of the nature of the matching process in RIMLD, on the other hand, the composition of the core set of correspondences changes only slightly as the video moves from frame to frame, and registration with RIMLD is thus much smoother.

Of course, RIMLD is not perfect. As can be seen in Figure 2.6, RIMLD's error rate does also increase slightly towards the end of the video, reaching a maximum of around 30 pixels, or 5 yards in our model. The explanation for this slight increase is that, many times, in the last frames of the video sequence, the camera is zoomed so far in that there simply are not enough features in the frame, globally distinctive or otherwise, to robustly compute a registration transform from feature correspondences. This is an issue future registration methods—especially those that use only local image features—may need to address. However, in American football video, the important action in the play is usually over by the time RIMLD begins to show signs of inaccuracy, so we do not concern ourselves with this matter.

2.3 Conclusions

In this chapter, we introduced a method for video registration that uses invariant local image features in conjunction with a matching technique based on a concept of local distinctiveness that finds video-to-model correspondences between non-distinctive features. In addition, we presented a simple empirical stability test that provides a means by which our registration method can be fully automated under the assumption that at least one frame in the video—not necessarily the first—contains a stable set of correspondences from globally distinctive features. Our technique was shown to yield significantly more accurate registration results in the challenging

American football domain than methods that rely only on the presence of globally distinctive features for registration. Finally, we offered our significant ground truth video dataset to the community for use as a benchmarking tool for video registration methods.

Chapter 3: Automatically Generating a Reference Set for Video Registration

In Chapter 2, we investigated a method for registering video frames to a static model by matching local image features to a set of reference images representing that model. In the work described in that chapter, we constructed the set of reference images manually by hand selecting from a large video collection individual video frames that together covered the model nearly completely and then forming point correspondences by hand between each selected reference image and the model. Needless to say, this manual work was tedious, and would be a burden for users in a deployed football-understanding system. For this reason, we wish to be able to automate the process of selecting a reference image set for video registration.

In this chapter, we investigate a method for doing this via video mosaicing. More specifically, given a collection of videos, the problem we investigate in this chapter is how to select a set of representative frames from a large collection of videos (where criteria for representativeness may be user-defined) and to register those frames with one another in a common coordinate system. This is a difficult problem with applications in many areas besides football analysis, such as panorama construction [5], super-resolution [69, 35], summary and indexing [34], compression [36], etc. [85, 86].

The primary challenge of the multi-video mosaicing problem is computational. In particular, good methods already exist for generating accurate mosaics from an input set of still images (e.g. [5]). However, because these methods are typically quadratic in the number of input images, applying them out of the box to a collection of many videos, each potentially with hundreds or thousands of frames, is simply infeasible.

To our knowledge, the multi-video mosaicing problem has never before been investigated. Specifically, all research efforts on video mosaicing of which we are aware have focused on the single-video problem, which, even for videos of reasonable length, is still computationally challenging. Early approaches to this problem simply computed image-to-image transformations between contiguous video frames to register the video with the coordinate system defined by the first video frame [34, 36]. Unfortunately, when transformations are computed locally between frames, as in these approaches, instead of being optimized globally (e.g. using bundle

adjustment, as in [5]), small errors can compound and lead to severe inaccuracy.

In more recent single-video work, Steedly *et al.* [69] attempt to integrate bundle adjustment in a tractable way by using an image feature-based method to select keyframes in the video based on degree of overlap. Under this method, each keyframe is compared to all other keyframes during bundle adjustment. However, each intermediate frame is compared only to the first keyframes before and after it in the video, resulting in a reduction in the overall complexity of bundle adjustment compared to a naive approach that bundle adjusts all of the video frames. Specifically, Steedly *et al.*'s bundle adjuster has an overall time complexity of $O(K^2 + n - K)$, where n is the total number of frames in a video and K is the number of keyframes selected, whereas the naive approach has an overall complexity of $O(n^2)$.

Be that as it may, Steedly *et al.*'s method still involves detecting and matching image features in every video frame. In our experience, these operations are prohibitively expensive, even for a single video. For example, just computing SIFT features [49] in a single 500-frame, 720×480 video can take more than 20 minutes on a typical desktop machine. For a single video, this order of feature computation time dominates the bundle adjustment times reported in [69], and it is clear that, for even a moderately large video collection, computing image features for every video frame poses a huge computational burden.

In this chapter, we present a novel approach for constructing single- or multi-video mosaics that is designed to overcome the limitations of previous single-video approaches. In particular, our method takes advantage of the fact that it is rarely necessary to include every video frame (in a single video or in a collection) in the mosaic. Instead, our method attempts to select a minimal subset of frames that achieve user-defined objectives for mosaic quality (e.g. a specific degree of redundancy, anytime maximal scene area, complete connectedness of selected frames, etc.).

At the heart of our approach is a utility maximization procedure that allows us to compute image features in only a fraction of the frames in a video collection by maintaining an estimate of the location of each video frame with respect to the current mosaic and using those estimates to greedily build the mosaic one frame at a time, selecting video frames that achieve the user's quality objectives. Importantly, because we take an iterative greedy approach, our method exhibits strong anytime properties.

Once frames are selected by our method, their mosaic-registration transforms are optimized globally, using a bundle adjustment procedure similar to the one employed in [5]. Because we take a greedy approach to frame selection, this style of full bundle adjustment is typically still tractable, since at any given iteration of our procedure, the subset of selected video frames min-

imally meets the user-defined quality objectives. However, a fast approximate bundle adjuster like Steedly *et al.*'s [69] can also be employed in conjunction with our method if the user's quality objectives require the incorporation of a large number of frames into the mosaic.

In addition to allowing tractable and effective multi-video mosaicing, our approach also allows for faster single-video mosaicing, since it computes image features in only a fraction of the video frames. Indeed, we demonstrate our approach's effectiveness with a number of compelling single- and multi-video experiments for which we provide both qualitative and quantitative results. In one particular set of experiments, we use our approach to construct a reference image set for American football video registration. The reference image set thus automatically constructed is comparable to the one manually constructed for use in Chapter 2.

In what follows, we describe in detail our greedy utility maximization approach for frame selection (Section 3.1), we discuss possible forms of the utility function (Section 3.2), and we outline a function for quickly estimating mosaics without computing image features (Section 3.3). In Section 3.4, we present a full set of qualitative and quantitative experiments to demonstrate our approach's effectiveness, and we conclude by summarizing our work in Section 3.5.

3.1 Utility Maximization Framework

We model multi-video mosaicing as a utility maximization problem. In particular, let $V_1, \ldots, V_N$ be a collection of N videos, where each video V_i, $i = 1, \ldots, N$, is itself a set of n_i video frames, i.e. $V_i = \{v_1^{(i)}, \ldots, v_{n_i}^{(i)}\}$, and let $\mathcal{V} = V_1 \cup \cdots \cup V_N$, denote the set of all video frames. Let $\mathcal{M}$ denote the space of possible mosaics, and let $M : 2^{\mathcal{V}} \rightarrow \mathcal{M}$ be a function that computes a mosaic from a subset of the video frames in the collection $\mathcal{V}$ (we discuss the form of $M(\cdot)$ we use in Section 3.4.1). Finally, let $U : \mathcal{M} \rightarrow \mathbb{R}$ be a function that assigns utility values to mosaics. (In general, $U(\cdot)$ may be designed to capture arbitrary properties of mosaic quality as desired by the user. We discuss possible forms of $U(\cdot)$ in Section 3.2.) Then, our goal is to find the subset of video frames $S^* \subseteq \mathcal{V}$ that yields the highest mosaic utility:

$$S^* = \arg\max_{S \subseteq \mathcal{V}} U(M(S)). \tag{3.1}$$

It may be possible to maximize some utility functions $U(\cdot)$ by simply selecting $S^* = \mathcal{V}$. However, because the state of the art in mosaic construction (including the mosaic builder we use, described later in Section 3.4.1) involves computing image features in all of the input

images—an extremely costly operation, as stated above—we wish to avoid including all of $\mathcal{V}$ in the final mosaic.

While it is possible to embed preferences about the number of frames selected into most forms of utility function, additional computational issues arise, since, even given the temporal structure of each video, performing the maximization in (3.1) exactly is extremely difficult. In fact, this problem is closely related to the set cover problem, which is known to be $\mathcal{NP}$-complete [20]. On the other hand, it is well known that the greedy approximation to set cover is highly effective [8]. Inspired by this fact, we use a greedy approach to approximately optimize (3.1).

Specifically, our method greedily builds an approximation to S^* one video frame at a time by repeatedly selecting the frame that, when added to the current mosaic, yields the new mosaic with the greatest utility. In other words, given the approximation S_t from the current iteration of our greedy procedure, we select a video frame $v_t^* \in \mathcal{V} \setminus S_t$ such that

$$v_t^* \;=\; \arg\max_{v \in \mathcal{V} \setminus S_t} U\big(M(S_t \cup \{v\})\big), \tag{3.2}$$

and we use v_t^* to form a new approximation $S_{t+1} = S_t \cup \{v_t^*\}$. We repeat this step until some stopping criteria are met.

Again though, an exact search for v_t^* at each iteration of our greedy procedure (3.2) would require computing image features in every video frame $v \in \mathcal{V}$ in order to compute mosaics $M(S_t \cup \{v\})$. To avoid this, we further define a deterministic function $\widetilde{M}(v, S_t)$ that, without computing image features, estimates the mosaic that would result from adding frame v to S_t (we discuss our specific design for this function in Section 3.3).

Our final greedy procedure is summarized in Algorithm 2. Under this approach, we compute and match image features in only one frame per iteration (specifically, in v_t^*) in order to compute an accurate final mosaic in line 6. This design is efficient, and it exhibits strong anytime properties, since the set S_t at any given iteration minimally achieves the mosaic quality objectives specified by $U(\cdot)$.

In the next two sections, we discuss possible forms of the utility function $U(\cdot)$, and the specific mosaic estimator $\widetilde{M}(\cdot)$ we employ.

Algorithm 2 Greedy procedure for multi-video mosaicing

Input: $\mathcal{V} = V_1 \cup \cdots \cup V_n$ — Video collection

1: $S_0 = \emptyset$
2: $t = 0$
3: **repeat**
4: $\quad v_t^* \;=\; \arg\max_{v \in \mathcal{V} \setminus S_t} U\Big(\widetilde{M}(v, S_t)\Big)$
5: $\quad S_{t+1} \;=\; S_t \cup \{v_t^*\}$
6: $\quad$ Compute final mosaic $M(S_{t+1})$
7: $\quad t = t + 1$
8: **until** stopping criteria are met

3.2 The Utility Function

The form of the utility function $U(\cdot)$ can vary depending on the user's specific goals. For example, the utility function can be defined to attain a desired level of redundancy across the space of the generated mosaic, e.g. to enable super-resolution [69, 35] of the mosaic image, or it could be defined to attempt to select a disjoint set of frames that summarizes the video collection.

Because this work is the first to examine the the design space of utility functions for the multi-video mosaicing problem, we concentrate on a single class of utility functions that we believe is broadly applicable. In particular, we focus on the class of utility functions that attempt to cover as much of the area of the scene depicted in the video collection $\mathcal{V}$ as possible, while keeping the number of frames selected to a minimum.

Perhaps the most straightforward utility function in this class is

$$U(M(S)) \;=\; \frac{A(M(S))}{|S|}, \tag{3.3}$$

where $A(M(S))$ denotes the total pixel area of the mosaic generated from image set S. This utility function corresponds loosely to the set-cover interpretation of multi-video mosaicing, encouraging the creation of mosaics that cover the maximum area using the least number of video frames.

It is important to note, however, that the utility function in (3.3) does not consider the number of connected components present in the mosaic. In particular, because area is maximized when there is zero overlap between images in the mosaic, this utility function will cause our greedy

procedure to prefer to select video frames that are not connected with each other and to fill in gaps only when necessary. This behavior may be useful for some applications such as video collection summarization, but in many others, such as panorama construction or compression, the user may desire a fully-connected mosaic instead. For these applications, a utility function that encourages the selection of overlapping video frames is required.

One way to accomplish this objective is by penalizing the selection of frames that do not connect to the current mosaic. For example, if we let N_S denote the number of connected components in the mosaic $M(S)$ and $\{S^{(k)}\}_{k=1,\ldots,N_S}$ denote the collection of the sets of frames in those connected components, where $S = \bigcup_k S^{(k)}$ and $S^{(i)} \cap S^{(j)} = \emptyset, \forall i \neq j$, we can craft a utility function that penalizes the creation of new mosaic components:

$$U(M(S)) \;=\; \sum_{k=1}^{N_S} A\big(M\big(S^{(k)}\big)\big) + \lambda N_S. \tag{3.4}$$

Here, λ is a parameter that determines how much to penalize the creation of each new mosaic component. Intuitively, λ helps determine a threshold on the amount of additional area a video frame must contribute, at each iteration of our greedy procedure, to an already existing mosaic component. Below this threshold, the greedy procedure will choose to begin a new, unconnected component at the given iteration rather than add to an existing component.

It is straightforward to determine this threshold exactly. In particular, let $O(v, M(S^{(k)}))$ denote the proportion of video frame v that overlaps with mosaic component $S^{(k)}$, and let $A(v, M(S^{(k)}))$ denote the area of frame v within mosaic component $S^{(k)}$. Then, under (3.4), a frame will be added to an existing mosaic component at a given iteration of our greedy procedure instead of being used to begin a new, unconnected component if we can find a video frame v and a mosaic component $S^{(k)}$ such that

$$\begin{aligned} A\big(M\big(S^{(k)}\big)\big) + \big[1 - O\big(v, M\big(S^{(k)}\big)\big)\big]\; A\big(v, M\big(S^{(k)}\big)\big) + \lambda\, N_S & \\ \geq \quad A\big(M\big(S^{(k)}\big)\big) + A(v) + \lambda\,(N_S + 1). & \end{aligned}$$

This reduces to the following condition:

$$\big[1 - O\big(v, M\big(S^{(k)}\big)\big)\big]\; A\big(v, M\big(S^{(k)}\big)\big) \;\geq\; A(v) + \lambda. \tag{3.5}$$

In other words, under (3.4), if a frame v cannot be found that contributes *new area* of at least

the size of an untransformed video frame (i.e. $A(v)$) minus some slack ($-\lambda$) to any mosaic component $S^{(k)}$, our greedy procedure will instead choose to begin a new, unconnected mosaic component.

Because it may be difficult to characterize areas within the transformed mosaic space in order to determine λ, we can replace the absolute component area $A(M(S^{(k)}))$ in (3.4) with some form of normalized area $\widehat{A}(M(S^{(k)}))$. There are several ways in which we can choose to define the normalized area $\widehat{A}(M(S^{(k)}))$. A natural first choice might be to simply normalize the absolute area of the mosaic component by the sum of the individual areas of its constituent video frames, or, in other words, to compute the fraction of "effective area" contained within a mosaic component. Unfortunately, under this definition of normalized area, the condition corresponding to (3.5) requires that the effective area of a component be increased with each new frame added to it, and this is only achieved when the fraction of an added frame's non-overlapping area is *greater than* the component's fraction of effective area. In other words, under this definition of normalized area, in order to grow a mosaic component, we must be able to continually add frames that overlap less and less with the component. This can be difficult to achieve.

We can obtain more reasonable behavior by instead normalizing each component's absolute area by the *average* video frame area in that component, i.e. by defining

$$\widehat{A}\big(M\big(S^{(k)}\big)\big) \;=\; \frac{A\big(M\big(S^{(k)}\big)\big)}{\frac{1}{|S^{(k)}|}\sum_{v\in S^{(k)}} A\big(v, M\big(S^{(k)}\big)\big)}. \tag{3.6}$$

Intuitively, this form of normalized area corresponds to the number of average-sized video frames required to comprise the total absolute area of mosaic component $S^{(k)}$ without overlap.

If we replace the absolute area in (3.4) with the normalized area from (3.6), our greedy procedure will choose to add to an existing mosaic component whenever a video frame v and a component $S^{(k)}$ can be found such that

$$\widehat{A}\big(M\big(S^{(k)} \cup \{v\}\big)\big) \;\geq\; \widehat{A}\big(M\big(S^{(k)}\big)\big) + 1 + \lambda.$$

That is, a video frame will be added to an existing component as long as a frame v can be found such that the number of average-sized frames comprising (without overlap) the absolute area of mosaic component $S^{(k)} \cup \{v\}$ is at least $1 - (-\lambda)$ more than the number comprising component $S^{(k)}$.

This condition leads to an intuitively appealing interpretation of λ. Specifically, λ acts as

a threshold on overlap for video frames with average area. For example, if we set $\lambda = -0.5$, then a frame with average area can overlap with a component by at most 50% to be added to that component. In other words, by setting $\lambda = -0.5$, we encode a preference that each selected frame should increase the area of some existing component by an amount greater than or equal half the area of an average-sized frame. If no such frame and component can be found, our greedy procedure will begin a new component.

In the experiments reported in Section 3.4, we use a utility function that takes this last form, with normalized areas as in (3.6).

3.3 Efficient Mosaic Estimation

In order to avoid computing image features in every video frame in a collection, we use a deterministic function $\widetilde{M}(v, S)$ that quickly estimates the mosaic that would result from adding video frame v to set S. By using this function instead of the full mosaic builder $M(\cdot)$ (which computes image features in all input frames), we can rapidly assess the utility that results from adding each individual frame in the video collection to the current mosaic. This, in turn, allows us to make an efficient greedy choice at each iteration of our procedure in Algorithm 2.

The idea behind our mosaic estimator $\widetilde{M}(\cdot)$ is to maintain a set of variables that describe how each video frame relates to the current mosaic. These variables include the frame's velocity and location relative to the current mosaic, as well as additional variables that describe whether we believe the frame is matchable to any component in the current mosaic. We discuss each of these variables in turn below and then describe how they are combined to compute $\widetilde{M}(\cdot)$.

3.3.1 Velocity and Location

The key component of our mosaic estimator $\widetilde{M}(\cdot)$ is an estimate $\mathbf{l}$ of the location within the current mosaic of each video frame $v \in \mathcal{V}$. In order to compute these location estimates, we model each video's velocity relative to the current mosaic. Specifically, for each video frame $v_j^{(i)} \in \mathcal{V}$, we maintain an estimate $\boldsymbol{\delta}_j^{(i)}$ of that frame's velocity by estimating the speed and direction of each of the frame's four corners relative to the current mosaic (modeling the velocities of frames' corners allows us to account for effects such as zoom, rotation, etc.).

The velocity estimates $\boldsymbol{\delta}_j^{(i)}$ are calculated based on the actual mosaic locations of frames already selected for inclusion in the mosaic in previous iterations of our procedure. Specifically,

before our procedure's first iteration, all frames' velocity estimates are set to an empirically determined initial value. Once two frames $v_{j_1}^{(i)}$ and $v_{j_2}^{(i)}$, $j_1 < j_2$, from the same video are selected and assigned to the same mosaic component with no other intermediate frames also assigned to the same component (i.e. $\exists\, k$ s.t. $v_{j_1}^{(i)}, v_{j_2}^{(i)} \in S_t^{(k)}$ and $\nexists\, j$ s.t. $j_1 < j < j_2$ and $v_j^{(i)} \in S_t^{(k)}$), their velocities are computed using their exact known mosaic locations as $\mathbf{l}_{j_2}^{(i)} - \mathbf{l}_{j_1}^{(i)}/(j_2 - j_1)$ (where we assume $j_2 > j_1$). The velocities of all other frames are set equal to the velocity of the closest frame in the same video that is already included in the mosaic. In other words, for each frame $v_j^{(i)} \notin S_t$, if we let

$$v_{j^*}^{(i)} \;=\; \arg \min_{v_k \in S_t \cap V_i} |k - j|, \tag{3.7}$$

then we set $\boldsymbol{\delta}_j^{(i)} = \boldsymbol{\delta}_{j^*}^{(i)}$.

Once velocity estimates are computed, we can estimate each frame's location relative to the current mosaic. Specifically, if, for each frame $v_j^{(i)} \notin S_t$, $v_{j^*}^{(i)}$ is defined as in (3.7), then we set

$$\mathbf{l}_j^{(i)} \;=\; \mathbf{l}_{j^*}^{(i)} + \sum_{k=j^*}^{j} \boldsymbol{\delta}_k^{(i)} \tag{3.8}$$

(with the appropriate adjustments made if $j < j^*$), where we assume unit time between contiguous frames.

3.3.2 Matchability from Mosaic Overlap

In addition to just estimating the location of each frame within the current mosaic, we also need to determine whether the final mosaic builder will be able to match each frame to an existing component in the current mosaic instead of being forced to use that frame to begin a new mosaic component. In the end, a frame's final matchability to an existing mosaic component is determined by the mosaic builder's ability to find and match distinctive image features between that frame and the existing mosaic. However, since our goal here is to be able to estimate a mosaic without computing image features, we use two separate heuristics to determine matchability.

The first of these heuristics is based on the frame's estimated overlap with the existing mosaic. In particular, this heuristic measures matchability based on the assumption that frames with a greater degree of overlap with the existing mosaic are more likely to contain image features

that can be matched with the mosaic. Specifically, we define

$$o_j^{(i)} = \begin{cases} 1 & \exists\, k \text{ s.t. } \widetilde{O}\big(v_j^{(i)}, M\big(S_t^{(k)}\big)\big) > \theta \\ 0 & \text{otherwise,} \end{cases} \tag{3.9}$$

where $\widetilde{O}(\cdot)$ estimates the proportion of overlap between a video frame and a mosaic component using the frame's location estimate $\mathbf{l}_j^{(i)}$, and θ is a threshold on overlap below which it is likely that a frame will be unmatchable to the current mosaic due to a lack of matchable image features (we empirically set $\theta = 0.35$ in our experiments).

3.3.3 Matchability from Proximity to Unmatchable Frames

Our second matchability heuristic is based on the premise that it is likely possible to match frames from the same video into a single mosaic component, and the reason a frame may not be matchable to the rest of its video is because it does not contain enough distinctive image features to match with other frames in the video, as is the case, for example, when a portion of the video contains primarily repeated patterns. These types of frames will not be matchable to the existing mosaic no matter how much they overlap with it. Thus, our second matchability heuristic attempts to identify frames without enough matchable distinctive image features based on their temporal proximity to other similar frames, with the rationale that frames in the same temporal neighborhood of a video are likely to contain similar content.

To accomplish this for a given frame $v_j^{(i)}$, we define $v_{j^-}^{(i)}$ as the closest unmatchable frame from the same video, and we set a threshold τ on $v_j^{(i)}$'s temporal proximity to $v_{j^-}^{(i)}$, below which $v_j^{(i)}$ is also to be considered unmatchable. Specifically, we set

$$u_j^{(i)} = \begin{cases} 1 & \text{if } |\, j - j^- | > \tau \\ 0 & \text{otherwise} \end{cases} \tag{3.10}$$

In practice, we vary τ based on the distance between $v_{j^-}^{(i)}$ and the nearest frame in the same video that *is* matched to the current mosaic, since the presence of nearby matchable frames also indicates the presence of matchable image features.

3.3.4 Final Mosaic Estimate

Once the variables $\mathbf{l}$, o, and u have been computed for a frame v using (3.8), (3.9), and (3.10), respectively, the final mosaic estimate $\widetilde{M}(v, S_t)$ can be computed for that frame and the current mosaic. If $o = u = 1$ (i.e. we believe the frame can be matched to the current mosaic), we compute $\widetilde{M}(v, S_t)$ by simply transforming v to its estimated mosaic location $\mathbf{l}$ and adding it onto the appropriate mosaic component $M(S_t^{(k)})$. Otherwise, if either $o = 0$ or $u = 0$ (i.e. we believe the frame *cannot* be matched to the current mosaic), we compute $\widetilde{M}(v, S_t)$ by adding v to $M(S_t)$ as a new, separate component.

3.4 Results

In this section, we demonstrate our mosaicing approach's effectiveness by presenting qualitative and quantitative results from a number of single- and multi-video experiments, including experiments that test our method's ability to construct a set of reference images for video registration in the American football domain. We compare against the baseline methods of random frame selection or selection of every k^{th} frame. However, as our experiments show, these methods exhibit major shortcomings. For example, random frame selection can yield highly variable results, sometimes producing a reasonable mosaic, but often missing portions of the scene. Similarly, attempting to set k to select every k^{th} frame can be very difficult, since setting k too large can result in missing portions of the scene, and setting k too small can lead to the selection of far more frames than necessary, thus eliminating any potential efficiency gains. Essentially, our procedure can be seen as attempting to set k dynamically at each iteration based on observed properties of the videos and the current mosaic.

3.4.1 Implementation Details

In all of the experiments described below, our method uses a utility function that takes the form of (3.4) but uses the normalized component area from (3.6) instead of absolute component area. The final mosaic builder $M(S_t)$ we use is similar to the one described by Brown and Lowe in [5]. Specifically, it computes SIFT features[1] in each video frame in S_t and matches these between all pairs of frames in S_t (these computations are cached to avoid duplicating them for S_{t+1}).

[1] We compute SIFT features for this process using the author's open-source implementation [24]: `http://eecs.oregonstate.edu/~hess/sift.html`

Based on the number of feature matches between each pair of frames, frame-frame matches are formed to construct the set of connected mosaic components $\{S_t^{(k)}\}$, and homographies relating the frames in each component to each other are optimized jointly using a Levenberg-Marquardt-based bundle adjuster. Note that, because final mosaic construction is not our focus, we use a basic mosaic builder that assumes planar mosaics and does not perform any sophisticated blending. As a result, some of our visual results exhibit parallax and ghosting effects.

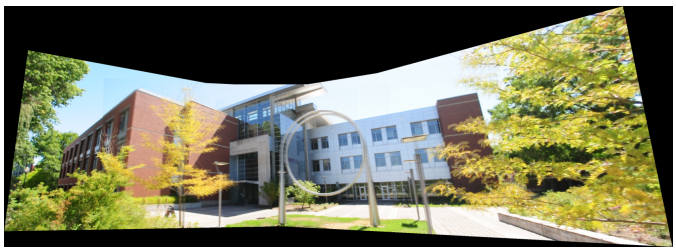
3.4.2 Single-Video Experiments

It is important to note that, because it selects a minimal subset of video frames to meet the user-defined goals encoded in the utility function, thereby avoiding the need to compute image features over the entire video, our mosaicing approach offers superior performance for both the single-video case and the multi-video case. For this reason single-video experiments are equally as necessary for assessing our approach's functionality and efficiency as multi-video experiments. Thus, we first tested our approach on a number of single videos.

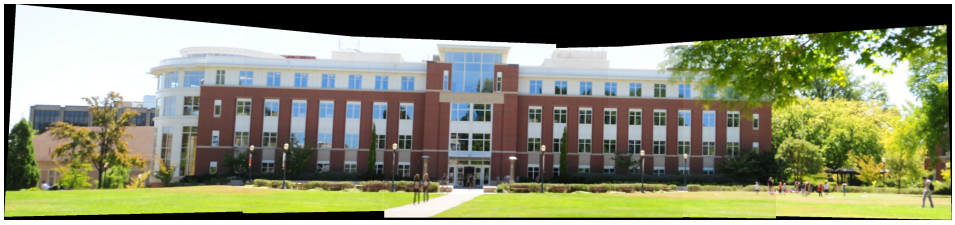
Each video in our single-video experiments was shot on one of two university campuses with a handheld digital camera, and, in order to test our approach's consistency under various conditions, each scene was shot using several different camera motion patterns (e.g. left-right, right-left, center-up-down, etc.). Since the results for each pattern were similar, we present here representative results for one video of each scene. The final mosaics and timing results for these videos are summarized in Figure 3.1. In each case, a high quality mosaic is constructed after only 4-5 iterations of our greedy procedure (Algorithm 2).

Since Steedly *et al.*'s algorithm [69] requires computing image features in every video frame, the time required to do so for each of our single-video examples is also reported in Figure 3.1 for reference. By comparison, our procedure's total running time is a fraction of this time alone. To get a better idea of the total running time for Steedly *et al.*'s algorithm, we can assume that the K frames selected by our procedure for each video are the same ones that would be selected as keyframes by Steedly *et al.*'s algorithm. Then the time spent in bundle adjustment in our algorithm would cancel with the quadratic component of their algorithm's bundle-adjuster, as would the time spent computing image features in the K selected frames, and, in addition to the time spent computing image features in the additional $n - K$ non-keyframes, their algorithm would also require an additional $O(n - K)$ operations to bundle adjust those frames.

(a) 288 frames; 1280 × 720 down-sampled to 711 × 400; shot center-down-up; 5 frames selected in 57 seconds vs. 401 seconds to compute SIFT features in all frames



(b) 96 frames; 1280 × 720 down-sampled to 711 × 400; shot right-left; 5 frames selected in 40 seconds vs. 159 seconds to compute SIFT features in all frames



(c) 192 frames; 1280 × 720 down-sampled to 711 × 400; shot left-right; 5 frames selected in 105 seconds vs. 303 seconds to compute SIFT features in all frames

(d) 1000 frames; 1280 × 720 down-sampled to 711 × 400; shot right-left; 4 frames selected in 75 seconds vs. 1085 seconds to compute SIFT features in all frames

(e) 750 frames; 1280×720 down-sampled to 711×400; shot nearly stationary at center with a fast pan downwards and then upwards in the middle of the video; 4 frames selected in 55 seconds vs. 1284 seconds to compute SIFT features in all frames

Figure 3.1 (caption on following page).

Figure 3.1: Mosaicing results for five single-video experiments. Depicted for each experiment is the final mosaic generated after the specified number of iterations of our greedy procedure outlined in Algorithm 2. In each case, high quality mosaics are constructed in only four or five iterations of Algorithm 2, and SIFT features are computed for only the corresponding number of frames. Included for reference in each subfigure caption is additional information for each experiment, including the total time required to compute SIFT features in every frame of the given video, which is necessary for Steedly et al.'s algorithm [69]. In each experiment, our procedure's running time is only a faction of the time required to compute SIFT features in all frames.

3.4.3 Multi-Video Experiments

Since our main focus is analysis of American football video, we assessed our procedure's ability to perform multi-video mosaicing on a data set containing 15 videos, each with between 400 and 600 720×480 frames, from the American football domain. The primary aim here is to be able to efficiently and automatically compute a set of reference images for the video registration procedure described in Chapter 2. The mosaics computed from football video collections are also useful for computing background models for player tracking and other tasks described in Chapters 4 and 5. The videos in the collection used for our mosaicing experiments are real videos used by NCAA coaches to analyze teams' performance. This data set poses a challenging test for our procedure, since it was shot with a camera that rapidly panned, tilted, and zoomed to follow the game's action. Moreover, because of the nature of the football field, there are many video frames that are difficult to match because they do not contain distinctive image features (recall from Section 3.3 that our approach explicitly models this phenomenon).

Nonetheless, our procedure was quite successful on this collection. As depicted in Figure 3.4, our greedy procedure from Algorithm 2 was able to select frames that covered the football field almost completely within 25 iterations, taking only about 41 minutes to do so. By comparison, simply computing image features in every frame of each of the 15 videos in the data set, as required by Steedly *et al.*'s algorithm [69], took over 4 hours. Similarly, manually constructing a comparable mosaic for use in [26], [28], and [25] took several hours of tedious labor.

Figure 3.2 details our procedure's progress per iteration in covering the football field scene depicted in the video collection by plotting the proportion of the in-bounds football field contained within the mosaic at each iteration. As a comparison, baseline results for the same data set are given for random frame selection and selection of every 100, 250, and 1000 frames. In the figure, the proportion of coverage was computed by registering the mosaic at each iteration with a scale model of the football field and computing the proportion of the field covered by registered mosaic pixels. As is clearly evident, our method comfortably outperforms the baseline methods. In addition, the strong anytime properties of our approach, noted above, are apparent in this example, with the connected scene coverage increasing at each iteration. By the 25^{th} iteration, our algorithm selects frames covering 96% of the in-bounds football field. This is even more impressive when considering that there are small portions of the football field that are never seen in the data set. In other words, within only 25 iterations, our procedure is able to select video frames covering essentially the entire scene depicted in a challenging 15-video collection.

In an additional multi-video experiment, we tested our method with a collection of 14 videos, seven of which contained portions of the same scene and seven of which were "noise" videos. Our greedy procedure from Algorithm 2 was able to identify the relevant videos and extract frames representing essentially the entire scene in only 15 iterations, totalling 2450 seconds. The resulting mosaic is depicted in Figure 3.3.

3.5 Conclusion

In this chapter, we presented a simple, flexible, and efficient method for building mosaics from a *collection* of videos. Our approach builds mosaics one frame at a time by greedily optimizing a utility function that can be defined to capture arbitrary properties of mosaic quality, as desired by the user. Its efficiency stems from the fact that it avoids computing image features in every video frame by making its greedy choices based on quickly estimated mosaics. Though we were motivated by the problem of building a reference image set for the video registration procedure described in Chapter 2, our mosaicing algorithm has a number of applications, and we demonstrated using a number of single- and multi-video experiments that our approach can build high-quality mosaics in a fraction of the time required to perform the operations undertaken by existing video mosaicing methods.

This work represents the first known exploration of the multi-video mosaicing problem, so there are obviously many potential directions for future work. In particular, in this work we focused only on the class of utility functions that attempt to maximize the connected scene area represented in the mosaic using the minimum possible number of frames. It would be interesting to explore different classes of utility functions, such as ones that attempt to achieve a desired degree of redundancy to allow for super-resolution, or ones that attempt to select a disjoint set of representative frames that succinctly summarizes the video collection.

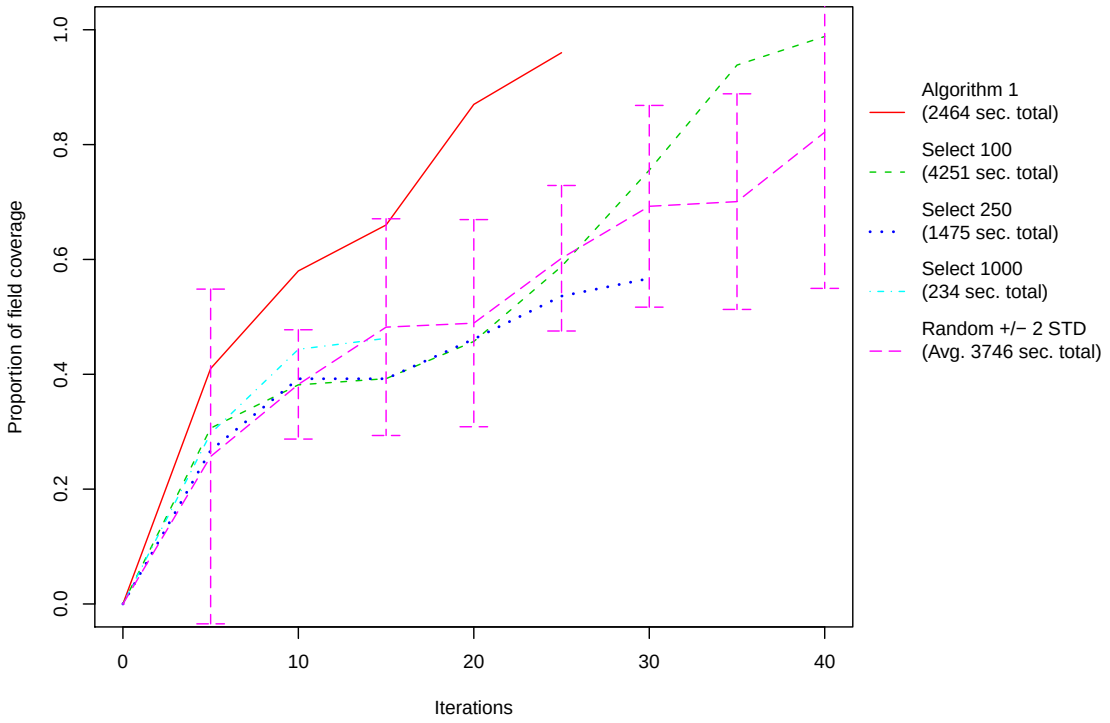
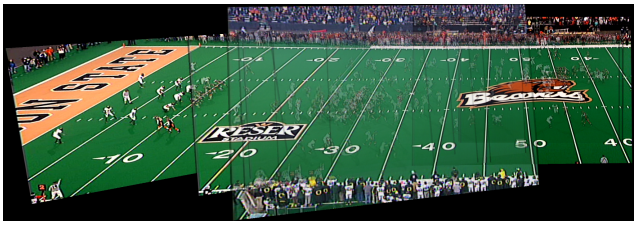


Figure 3.2: Proportion of the in-bounds football field covered by the mosaic computed after each iteration of our greedy procedure on the 15-video American football data set. Our approach exhibits strong anytime properties, with the connected scene coverage increasing at every iteration. Nearly the entire football field is covered by the mosaic after only 25 iterations of Algorithm 2, completed in 2464 seconds. For comparison the coverage attained from selection of every 100, 250, and 1000 frames is presented along with the average coverage attained over 5 runs of random frame selection (with error bars indicating 2 standard deviations). For each run of random frame selection and for selection of every 100 frames, a total of 40 frames were chosen, with runtimes totaling 3746 seconds (on average) and 4251 seconds, respectively. For selection of every 250 frames and every 1000 frames, the number of frames that could be selected was limited by the number of videos and their lengths, so only 30 and 15 total frames, respectively, were chosen for these methods, and they achieved total respective runtimes of 1475 seconds and 234 seconds. Though selection of every 100 frames eventually achieves similar coverage, our method from Algorithm 2 still clearly outperforms each of the baselines by a comfortable margin.

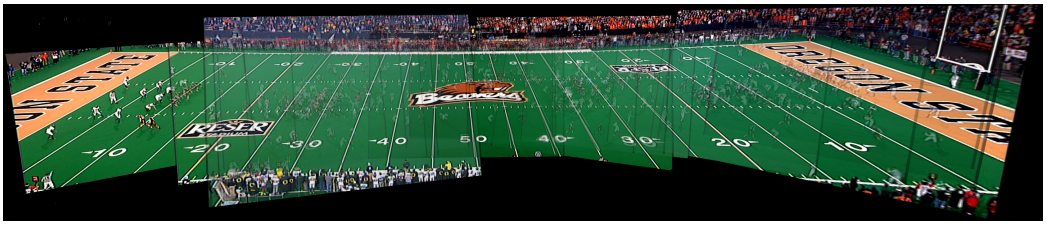
Figure 3.3: A mosaic generated from a collection of 14 short (100-200 frame), high-resolution videos, seven of which contain a portion of the depicted scene, and seven of which were "noise" videos. The mosaic contains essentially the entire scene and was constructed in 15 iterations of Algorithm 2 in a total of 2450 seconds.

(a) 5 iterations; 27 seconds

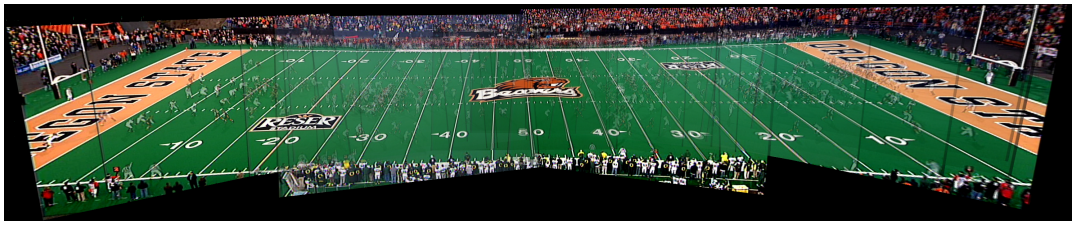


(b) 10 iterations; 256 seconds

(c) 15 iterations; 702 seconds



(d) 20 iterations; 1365 seconds



(e) 25 iterations; 2464 seconds

Figure 3.4: Mosaics generated for the 15-video American football data set after 5, 10, 15, 20, and 25 iterations of our greedy procedure outlined in Algorithm 2. The total run time after each iteration is also given (note that these do not scale linearly with the number of iterations because the utility function $U(\cdot)$ must be computed for additional videos as they are selected for observation by our procedure in later iterations). Nearly the entire football field is covered with a high-quality mosaic after only 25 iterations of our procedure, or only about 41 minutes. By comparison, just computing SIFT features in every frame of each video in the data set took 4 hours and 7 minutes.

Chapter 4: Recognizing Initial Player Formations with Mixture-of-Parts Pictorial Structures[1]

Recognizing initial formations of players is a fundamental task in the analysis of American football. Not only is initial formation information useful in its own right, as a component of a team's strategy, we can also use it to initialize a player tracker, as we do in Chapter 5. Recognizing initial football formations—that is, determining the starting location and type (QB, WR, DT, etc.) of each player—is not an easy problem. Because of the general clutter on the field and the inadequacy of current object detection methods, simply detecting players to determine their locations is infeasible. Moreover, player type is tied closely to player location in the initial formation, so much so that, when we consider that players on the same team appear nearly identical, the locations of players relative to one another is the single most informative piece of information for initial formation recognition. For this reason, we desire an algorithm that can leverage relative location information in recognizing initial football formations.

Pictorial structures are one such model. Pictorial structures are graphical models for representing and localizing objects with multiple spatially related parts. These models represent an object as a set of parts with local appearance models for each part and deformable connections between parts that describe their ideal relative locations. Given a pictorial structure, object recognition/localization corresponds to jointly assigning locations to all parts that minimize the combined local, appearance-based cost of each part plus the deformation cost based on the connections between parts. For restricted—but useful—classes of pictorial structures, efficient algorithms for performing this minimization have been developed that make recognition quite reasonable in practice [17]. By jointly reasoning about part appearances and relative positions, pictorial structures can provide more robust inference than approaches that reason about object parts in isolation, as has been demonstrated for a number of multi-part object recognition problems [12, 44, 18].

However, a fundamental assumption underlying pictorial structures is that each object instance contains the same set of parts with the same set of deformation constraints among those parts. Unfortunately, this assumption does not hold for many multi-part object classes for which

[1] The work described in this chapter was published in [28] and [27].

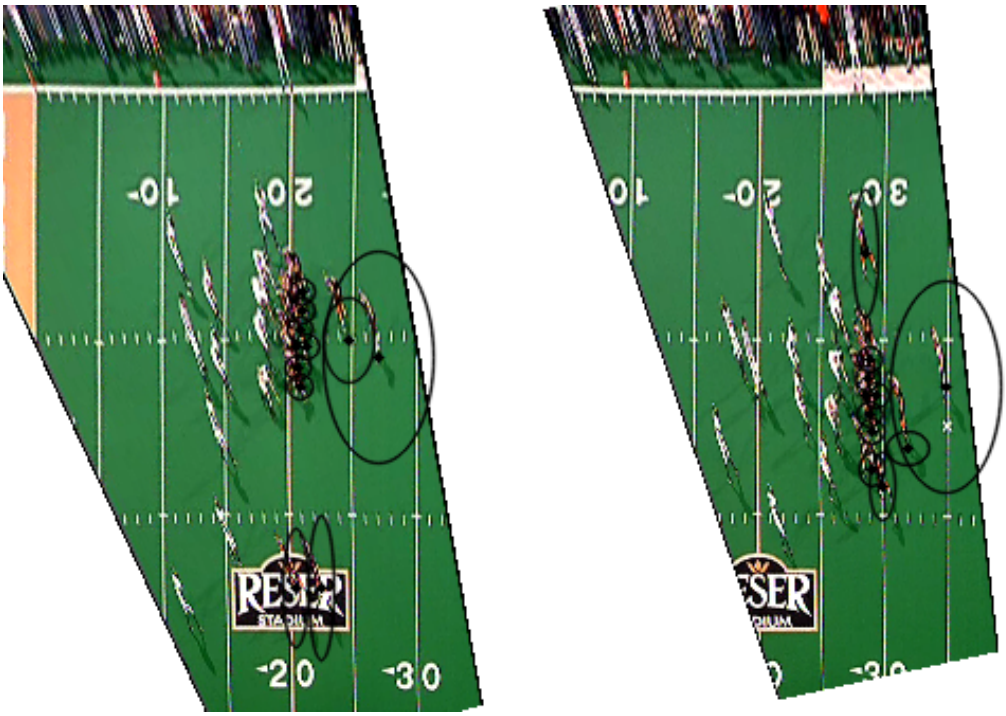
the set of parts can vary not only in location but also in type. Examples of this type of object class include furniture, such as chairs, which can have different types of arms, legs, backs, rockers, etc.; the human figure, along with accessories such as watches, hats, footwear, etc.; houses and other buildings; multi-agent sports scenes; car and airplane types; or any object class for which occlusion can be an issue.

More importantly, if we think of initial formations in American football as multi-part objects whose parts correspond to players, these are another object class whose parts can vary in both location and type. In particular, each football formation involves various subsets of players, each having a distinct player type corresponding to his role (e.g. left flanker, fullback, center, etc.), and, as mentioned above and as illustrated in Figure 4.1, the spatial constraints between players are determined largely by the particular subset of players in the formation.

While the pictorial structure model may seem *prima facie* useful for leveraging the spatial layout of players to recognize a football formation, there are complicating factors that make this difficult or impossible with pictorial structures out of the box. One of these factors is the rules of football, which enforce certain hard constraints on formations that restrict the number of certain types of players in the formation as well as their spatial configuration. These rules make it very difficult to formulate a single pictorial structure to recognize all possible football formations. In addition, because there are thousands of legal formations, formulating a pictorial structure model for each one is practically infeasible and would ignore the significant degree of common structure between similar formations.

In this chapter, we propose an extension of the classical pictorial structure model called the mixture-of-parts pictorial structure (MoPPS) model for recognizing multi-part objects with variable part sets (i.e. localizing and classifying the parts of these objects). The MoPPS model is characterized by three components: a set of available parts, a prior distribution on part subsets that assigns positive probability only to legal part sets, and a function that returns a pictorial structure for any legal part subset. Intuitively, a MoPPS model can be viewed as an implicit representation of a very large collection of pictorial structures that captures the possible variations of objects with variable part sets. Under a generative view of this model, a subset of parts is first drawn from the corresponding prior distribution, then, given the part subset, the corresponding pictorial structure is used to generate locations and appearances for each of the parts. Inference on a MoPPS model corresponds to jointly computing the most likely, or least cost, subset of parts and their locations.

In the absence of special structure, exact inference in MoPPS models is a hopelessly com-



(a) (b)

Figure 4.1: The configuration of players in an American football formation can vary drastically depending on the subset of players in the formation. Above are depicted, mapped to an overhead view, two very different formations containing different subsets of players. Player locations are marked along with confidence ellipses at two standard deviations based on distributions of the relative locations of players. Because player appearances are nearly identical, this variation in structure provides the necessary leverage point for formation recognition.

plex combinatorial optimization problem. Therefore, we describe a restricted but reasonable representation for MoPPS models that facilitates their easy specification as well as practically efficient inference. In particular, we represent MoPPS models in terms of a large pictorial tree structure involving all possible parts along with hard constraints on legal part subsets. This representation facilitates the computation of upper and lower cost bounds on part subsets that can be integrated into branch-and-bound style optimization.

To validate the MoPPS model and tree representation, we apply them to the American football formation recognition problem. In a previous attempt to solve this problem, Intille used a knowledge base of purely hard constraints along with a SAT-like procedure for inference [31].

Unfortunately, Intille's method was quite brittle, required significant human pre-processing, and performed poorly enough to be deemed unacceptable for use in later stages of his football understanding system. In contrast, our results show that MoPPS models facilitate accurate recognition and localization in a reasonable time frame without human preprocessing. To our knowledge, there have been no other attempts—in the football domain or otherwise—to solve the recognition problem for objects with variable part sets.

4.1 Pictorial Structures

Under the classical pictorial structure model, a class of objects is represented as a graph with n vertices $V = \{v_1, \ldots, v_n\}$ representing the parts of the object and a set of edges $E = \{(v_i, v_j)\}$ representing the connections between parts. Associated with each object class is also a set of model parameters Θ which includes part appearance parameters $A = \{a_1, \ldots, a_n\}$ and connection parameters $\Delta = \{\delta_{ij} \mid (v_i, v_j) \in E\}$ describing the ideal relative locations of connected parts. A particular instance of an object is represented as a set of locations of its parts $L = \{l_1, \ldots, l_n\}$.

Given an image I and a set of object model parameters Θ, the posterior distribution over the set of part locations is

$$p(L \mid I, \Theta) \;=\; \alpha\, p(I \mid L, \Theta)\, p(L \mid \Theta), \tag{4.1}$$

where α is a normalizing term, $p(I|L, \Theta)$ measures the likelihood of the image data given a particular configuration of the object, and $p(L|\Theta)$ is the prior distribution over object configurations.

Locating a single object in an image corresponds to maximizing (4.1), and Felzenszwalb and Huttenlocher have shown that if E, $p(I|L, \Theta)$, and $p(L|\Theta)$ satisfy certain, reasonable conditions, then efficient algorithms exist to perform this maximization exactly [17]. Specifically, if the edges in E form a tree and $p(I \mid L, \Theta)$ can be factored as a product of individual part appearance models, then the posterior distribution takes the form

$$p(L \mid I, \Theta) \;=\; \alpha \prod_{i=1}^{n} p(I \mid l_i, a_i) \frac{\prod_{(v_i, v_j) \in E} p(l_i, l_j \mid \delta_{ij})}{\prod_{i=1}^{n} p(l_i \mid \Theta)^{\deg(v_i)-1}}, \tag{4.2}$$

where the $p(I \mid l_i, a_i)$ are individual part appearance models, $p(l_i, l_j|\delta_{ij})$ are priors over relative locations of connected parts, $p(l_i|\Theta)$ are priors over individual part locations, and $\deg(v_i)$ is the

degree of vertex v_i.

Under this factorization, finding the optimal configuration L^* of an object corresponds to the following well known cost minimization problem:

$$L^* = \arg\min_L \left(\sum_{i=1}^{n} m_i(l_i) + \sum_{(v_i,v_j)\in E} d_{ij}(l_i, l_j) \right), \tag{4.3}$$

where $m_i(l_i) = -\log p(I|l_i, a_i) + (\deg(v_i) - 1) \log p(l_i|\Theta)$ is the local match cost for each part and $d_{ij} = -\log p(l_i, l_j|\delta_{ij})$ is the deformation cost between each pair of connected parts. If $p(l_i, l_j \mid \delta_{ij})$ is Gaussian, then (4.3) can be computed exactly in $O(hn)$ via distance transforms, where h is the number of possible part locations [17, 16].

4.2 Mixture-of-Parts Pictorial Structures

As discussed at the beginning of this chapter, the classical pictorial structure model's assumption of a static part set undermines its ability to recognize some multi-object classes whose parts can vary not only in location but also in type. To overcome this limitation, we introduce in the next two subsections an extension of classical pictorial structures called the mixture-of-parts pictorial structure (MoPPS) model and a specific, restricted MoPPS model representation that facilitates practically efficient inference.

4.2.1 General MoPPS Model

The MoPPS model is a triple $M = \langle \mathcal{V}, p_v, f \rangle$ where $\mathcal{V}$ is a finite set of parts, p_v is a probability distribution over $2^{\mathcal{V}}$ (i.e. subsets of $\mathcal{V}$), and f is a function that assigns a pictorial structure model to each subset $V \in 2^{\mathcal{V}}$ with $p_v(V) > 0$ (later, we discuss a particular representation for p_v and f). We use Θ_V to denote the parameters of the pictorial structure assigned to part set V and take the vertices and edges of the structure to be implicit in the parameters. Intuitively, a MoPPS model can be viewed as generating image data by first drawing a part subset V according to p_v and generating the image according the generative process dictated by the pictorial structure parameterized by Θ_V.

In the case of American football, the set of parts $\mathcal{V}$ corresponds to all possible players, each of which has a specific role (e.g. fullback, left flanker, shotgun quarterback, etc.). The probability

distribution p_v assigns non-zero probability only to those formations that contain exactly 11 parts (the number of players required in a formation) and that obey the formation constraints dictated by the rules of football (e.g. there must be 7 players on the line). Given a legal subset of players V, the corresponding pictorial structure Θ_V encodes the spatial constraints among the players in V along with local observation models for each player. Note that in this domain, the observation models for each player/part are identical since players have very similar appearances.

Given an image I and a MoPPS model $M = \langle \mathcal{V}, p_v, f \rangle$, we are interested in inferring the most likely part set V and the locations L of those parts. The joint posterior distribution over V and L is given by

$$p(L, V \mid I, M) \;=\; \alpha \, p(I \mid L, \Theta_V) \, p(L \mid \Theta_V) \, p_v(V), \tag{4.4}$$

where α is a normalizing term, $p(I|L, \Theta_V)$ measures the image data likelihood under the pictorial structure model for V, and $p(L|\Theta_V)$ is the corresponding prior distribution over part locations. Note that under this model the marginal probability of the image data can be viewed as a mixture distribution of pictorial structure components, with one per legal subset of parts; hence the name MoPPS.

Let $C(L \mid I, V) = -\log\left(p(I \mid L, \Theta_V) \, p(L \mid \Theta_V)\right)$ denote the cost assigned to locations L for parts V by pictorial structure Θ_V. We can then write our objective of finding the most likely locations and parts as computing

$$(L^*, V^*) = \arg\min_{(L,V)} C(L \mid I, V) - \log p_v(V). \tag{4.5}$$

Assuming all pictorial structures Θ_V allow for efficient minimization of $C(L \mid I, V)$, e.g. by assuming tree structures and Gaussian edge potentials, then the primary complexity in this minimization problem is the exponentially large set of part subsets that must be considered. An exhaustive enumeration of these will typically not be tractable. However, if one does not make any assumptions about the MoPPS model, then in the worst case exhaustive search is the best we can do (it is straightforward to show NP-completeness). To achieve practically efficient inference, therefore, we developed the MoPPS tree representation for a restricted class of MoPPS models. We present this representation in the next subsection. To simplify the discussion, we will assume for the remainder of the paper that p_v is a uniform distribution over all legal sets of parts. There are straightforward ways in which the inference procedure we describe later can

incorporate non-uniform priors.

4.2.2 The MoPPS Tree Representation

A MoPPS tree representation is a triple $\langle \mathcal{V}, \Theta, T \rangle$, where $\mathcal{V}$ is again a finite set of available parts, Θ is a tree-structured pictorial structure (the *global pictorial structure*) over the entire set of parts, and T is a boolean function that maps each part subset V to either **true** or **false** depending, respectively, on whether or not it is a legal part subset. We will denote by $\Theta|_V$ the projection of Θ onto V, which is just the subgraph of Θ induced by the part set V. Given a MoPPS tree representation the corresponding MoPPS model is given by $\langle \mathcal{V}, p_v, \Theta|_V \rangle$, where p_v is uniform over subsets V with $T(V) =$ **true**.

This representation can be viewed as compactly specifying $f(V) = \Theta|_V$ using the global pictorial structure by returning the projection of part set V onto this structure for any legal V. The set of pictorial structures allowed by this model is constrained so that the pictorial structures returned for any two part sets V and V' must be consistent for parts in $V \cap V'$. Furthermore the pictorial structures $\Theta|_V$ will all be tree structured. An important property of this representation utilized in the inference procedure described in the next section is the monotonicity of the pictorial structure cost function. In particular, if $C^*(I, V) = \min_L C(L \mid I, V)$ is the minimum pictorial structure cost for part set V, then for any part subsets (legal or illegal) V and V', if $V \subseteq V'$ then $C^*(I, V) \leq C^*(I, V')$.

Clearly MoPPS trees cover only a subclass of possible MoPPS models. Intuitively, they are unable to represent object classes for which the spatial relationships between parts are not pairwise independent. MoPPS trees also cannot represent models in which one legal part set is a subset of another because, due to the monotonicity property of MoPPS trees, the larger part set will always achieve a higher cost and so will never be selected as the best solution. However, despite these restrictions, MoPPS trees are rich enough to represent interesting object classes, as we demonstrate in Section 4.4, and they provide structure that can be leveraged to help achieve practically efficient inference. Extending to allow for richer subclasses while maintaining practical inference is an interesting direction for future work.

4.3 MoPPS Inference

Given a MoPPS model M represented as a MoPPS tree $\langle \mathcal{V}, \Theta, T \rangle$ we wish to solve the minimization problem defined in (4.5). Note that that if we know V^*, then we can efficiently compute L^* via the pictorial structure $\Theta|_{V^*}$. Thus, the fundamental problem here is to compute V^*. Under our assumption of a uniform p_v we can formulate the optimization problem as

$$V^* = \arg \min_{\{V:T(V)\}} C^*(I, V). \tag{4.6}$$

In other words, we simply wish to find a legal part set with minimum pictorial structure cost among other legal sets.

Our approach to solving this optimization problem is to cast it in the framework of branch-and-bound search (BBS) and to leverage the special structure of the MoPPS tree representation to efficiently compute informative upper and lower cost bounds as required by BBS.

4.3.1 Branch-and-bound search

Branch-and-bound search is a classical approach to combinatorial optimization that searches through a tree structure in which every node represents a subset of a space of combinatorial objects. Leaves of the BBS tree typically represent singleton sets or single combinatorial structures. As BBS proceeds, it continually expands new tree nodes and prunes any node from consideration whenever it can be proven that all structures it represents are suboptimal. Finding these nodes is done by computing both an upper and a lower bound on the cost of the combinatorial structures represented by each expanded node. A node can be pruned without sacrificing optimality if its lower bound is greater than any other node's upper bound.

In the case of MoPPS inference, the combinatorial objects of interest are legal part sets, and, hence, each node of the search tree represents collections of part sets. Each node is labeled by a set of parts V, indicating that the node represents all legal part sets V' that contain the parts in V. More formally, we assume that a search space $\langle V_0, s \rangle$ is available for a given MoPPS optimization problem, where V_0 represents the initial search node (V_0 is just a set of parts or possibly the empty set), and s is a successor function that for any node of the tree V returns $s(V) = \{V'_1, \ldots, V'_k\}$ where the V'_i are successor part sets of V and it is assumed that the space satisfies $V \subseteq V'_i$ for all successors.

For a particular application it is generally easy to hand-specify the search tree. However, it

is also relatively easy to automatically compile such a search from a MoPPS tree representation. In particular, we need only assume the availability of a function T' that returns **true** for a part subset V iff it is a subset of some legal part set. Given the function T' one can automatically specify a space by setting $V_0 = \emptyset$ and then having $s(V)$ return the set of all part sets V' that result from adding one part to V and such that $T'(V')$ is **true**. We take this latter approach in our application.

The other basic elements that must be specified to cast MoPPS inference as BBS are methods for computing an informative upper bound $c_u(V)$ and lower bound $c_l(V)$ on the cost of the set of part sets represented by a search node V.

Given a search space and upper and lower bound functions, we use a best-first search strategy for BBS, which additionally requires an ordering relation $<_o$ with which to maintain a priority queue of encountered search nodes. Under this strategy, each search step removes the first node from the priority queue, expands it according to s, and adds its successors to the priority queue according to $<_o$. Search stops when all search nodes have been eliminated except a single leaf node representing the optimal solution. In our experiments we consider two ordering relations: $<_l$, which orders nodes according to their lower bound, and $<_u$, which orders according to the upper bound. The effect of using different ordering relations is that a better ordering will expand good leaf nodes earlier than a poor one.

Algorithm 3 gives pseudocode for best-first BBS.

4.3.2 Lower bound computation

An important property resulting from the subset relationship maintained by the successor function s is that any descendent V' of a search node V is a superset of V and hence, due to the monotonicity of the MoPPS tree representation, we have $C^*(I, V) \leq C^*(I, V')$. In particular, the cost of a node V will never be greater than that of any leaf node (i.e. legal part set) under V. This means that to compute a lower bound on the cost of any complete part set represented by V, i.e. the any of the leaf nodes under V, we need only to compute $C^*(I, V)$, which can be done efficiently using the pictorial structure $\Theta|_V$. Thus, one choice for the lower bound is to take $c_l(V) = C^*(I, V)$.

This lower bound can be easily improved in cases where one can find out the minimum number of parts in any leaf node under V. This is straightforward in the football domain since each formation must contain exactly 11 players. In general, suppose that the minimum size leaf

Algorithm 3 Best-first branch-and-bound MoPPS tree search

Input: $\langle V_0, s \rangle$ – Input search space
$<_o$ – Ordering relation
c_l – Lower bound function
c_u – Upper bound function
I – Input image

1: $c^* \leftarrow \infty$ // initialize minimum cost
2: $Q \leftarrow$ **NIL**
3: ENQUEUE$(Q, V_0, <_o)$ // initialize priority queue with initial search node
4: **repeat**
5: $V \leftarrow$DEQUEUE(Q) // get best node on queue
6: **if** $s(V) = \emptyset$ **then**
7: **if** $C^*(I, V) < c^*$ **then**
8: $c^* \leftarrow C^*(I, V)$ // check for new minimum cost leaf node
9: $V^* \leftarrow V$
10: **end if**
11: **if** $\forall\ V'$ in Q, $c^* \leq c_l(V')$ **then**
12: RETURN V^*
13: **end if**
14: **else**
15: $\{V'_1, \ldots, V'_k\} \leftarrow s(V)$ // expand V
16: **for** $i \leftarrow 1..k$ **do**
17: ENQUEUE$(Q, V'_i, <_o)$
18: **end for**
19: PRUNE(Q, c_l, c_u) // prune dominated nodes in Q
20: **end if**
21: **until** forever

node has k additional parts beyond V, and let $C^*_v = \min_{v \notin V} C^*(I, \{v\})$ denote the minimum cost of any pictorial structure $\Theta|_{\{v\}}$, where v is a part that is not in V (note that each such cost is based only on the corresponding part's local match cost). It is straightforward to verify that in this case $c_l(V) = C^*(I, V) + k\, C^*_v$ is still a lower bound.

4.3.3 Upper bound computation

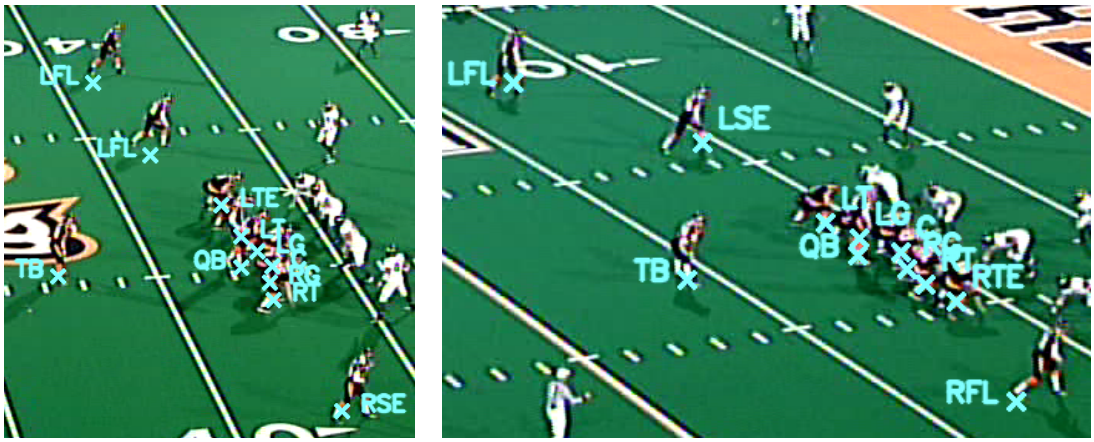
The main idea of our upper bound calculation is to quickly find a legal set of parts V_u that is a superset of the current node V and that we expect will have low (though perhaps not optimal) cost. If we can find such a set of parts, then we can use $C^*(I, V_u)$ as an upper bound on the cost of V. The key then is to quickly compute V_u, which we can do by leveraging the MoPPS tree representation.

In particular, prior to search, we use the global pictorial structure Θ to compute locations $\mathcal{L}$ for the entire set of parts $\mathcal{V}$. Then, to compute an upper bound on the cost of a node V during BBS, we select V_u as the minimum cost legal subset of $\mathcal{V}$ containing V with the location of each part in V_u fixed at the one specified in $\mathcal{L}$. That is, we select the V_u that minimizes $C(\mathcal{L}[V_u] \mid I, V_u)$ such that $V \subseteq V_u \subseteq \mathcal{V}$, $T(V_u) =$ **true**, and where $\mathcal{L}[V_u]$ is the set of locations in $\mathcal{L}$ for parts in V_u. We can then use $c_u(V) = C(\mathcal{L}[V_u] \mid I, V_u)$ as an upper bound on the cost of V. This upper bound may be tightened at the expense of an extra pictorial structure optimization by computing $c_u(V) = C^*(I, V_u)$.

The key to this upper bound is the fact that evaluating $C(\mathcal{L}[V_u] \mid I, V_u)$ for different subsets V_u is many orders of magnitude faster than computing V^*, which involves optimization over both locations and part sets. This permits for the search for the optimal V_u to be done via another branch-and-bound search or exhaustively, if computationally feasible. If exact optimization of V_u is still too costly, V_u may be approximated with a greedy, approximate hill-climbing search which at every step selects from the parts remaining in $\mathcal{V}$ the minimum cost part that does not make V_u an illegal part set. Such an approximation will typically yield a useful upper bound, though this will not always be the case. Ultimately, if a legal part set V_u has a low cost relative to $C(\mathcal{L} \mid I, \mathcal{V})$ (above) and $c_l(V)$ (below), it is likely that that V_u is a reasonably good part set.

4.4 Experiments in American Football

In this section, we demonstrate the capability of the MoPPS tree model by applying it to the challenging American football formation recognition problem. As described above, our goals in this domain are to classify the players that constitute the formation as well as to determine their locations.



(a) (b)

Figure 4.2: Some formations in American football differ only very subtly. The offensive formations depicted above (the orange and black players, with inferred locations and types overlaid) are two such ones. These formations differ by three players, but the differences between their spatial configurations are very slight and may be difficult even for an untrained human eye to detect. Still, as shown, the MoPPS tree model correctly locates and classifies all of the players in both images.

4.4.1 Domain Description

The dataset on which we tested the MoPPS tree model contains 25 images of the initial formations of American football plays.[2] Each formation consists of 11 players who may be one of 16 basic types. The rules of football impose certain restrictions on formations such as the requirement that there be exactly seven players on the line of scrimmage (the imaginary line between offense and defense), the requirement that the rest of the players be at least one yard behind the line of scrimmage, and the requirement that there be a quarterback and five down linemen.

The images in our dataset depict several formations, but as illustrated in Figure 4.2 (a) and (b), the differences between them are sometimes very subtle. However, because player appearances are very similar in our low-resolution imagery, these cannot be used as an indicator of player identity. Instead, we must rely on the relative spatial configuration of the players, which is determined by the particular subset of players that constitutes the formation.

As discussed above, classical pictorial structures cannot cope with the variation in player types in the class of American football formations. Intille attempted to solve the football for-

[2]This dataset is available at `http://eecs.oregonstate.edu/football/formations/dataset/`.

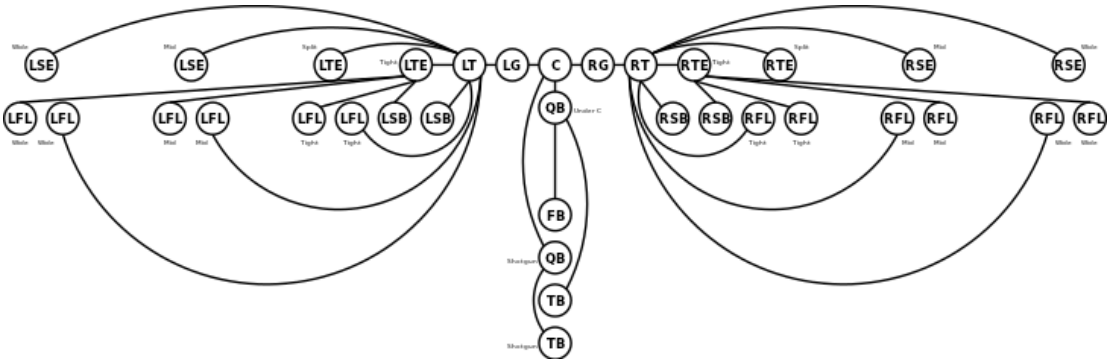


Figure 4.3: MoPPS tree representation for initial player formations in American football.

mation recognition problem [31], but his recognition system had many major shortcomings. For instance, whereas we attempt to jointly compute the most likely set of players and their locations, Intille's system took as input a set of manually specified player locations and attempted only to assign player type labels to those locations. To do this, Intille manually constructed a knowledge base of hard constraints, such as "near", "to the left of", and "bit of vertical space between", relating player locations—in itself an enormously time consuming and tedious task. He used this knowledge base to cast formation labeling as a SAT-like problem that was solved approximately using a number of search heuristics. In the end, the results of this system were poor, largely because of the strong numerical aspects of the problem, and could not be used in later stages of Intille's football understanding system.

4.4.2 MoPPS Model for Football Formations

For formation recognition, we use a MoPPS tree model with a total of 34 available parts corresponding to the 16 basic player types as well as several subtypes that capture different attributes of certain players (such as whether the quarterback is in shotgun formation or under center). These parts, subject to a set of hard constraints based on the rules of football, combine to form over 3200 legal formations. Our MoPPS tree for football formations is depicted in Figure 4.3.

Each image in our dataset is automatically registered to an overhead view of the football field, as depicted in Figure 4.1, using the technique described in Chapter 2, allowing us to model the relative locations of players in 2D football field coordinates. Specifically, the connection parameters δ_{ij} are the mean and diagonal covariance of a Gaussian distribution over each player's

ideal location in field coordinates relative to a "parent" player in the MoPPS tree. These parameters were manually set using a small set of training images.

Each player is treated as being identical in appearance, and the observation model $p(I|l_i, a_i)$ for players is a combination of two models: one, $p_b(I|l_i, a_i)$, based on background segmentation and another, $p_h(I|l_i, a_i)$, based on color histogramming.

To compute $p_b(I|l_i, a_i)$, we register a large collection of football video with the planar overhead field model and, for each pixel in the model, draw a set of samples uniformly from the set of all RGB values that register to that pixel. This sample set is used to compute a kernel density estimate of the field color distribution for the pixel under the (valid) assumption that the pixel exhibits it's true field color for all but a small fraction of the video frames in which it appears. This process is repeated for every pixel in the field model. Player likelihoods for the image I are computed using this model by projecting I into field model space and computing the probability of each pixel under its corresponding field color distribution. Pixels whose probability is below a manually specified threshold are considered foreground pixels, and all others are considered background pixels. The likelihood $p_b(I|l_i, a_i)$ is computed for each pixel in the original image space as the proportion of foreground pixels within a player-sized rectangular region anchored to that pixel at the bottom center (to put high likelihood at players' feet), and these likelihood values are projected back into field model space for compatibility with the structure model.

Because $p_b(I|l_i, a_i)$ does not differentiate between players on opposing teams, we also use an HSV histogram-based model $p_h(I|l_i, a_i)$ to help separate the players on the team of interest from the players on the other team. To compute $p_h(I|l_i, a_i)$, we use the method described by Pérez *et al.* in [63]. Specifically, we compute a reference histogram of HSV player color using a small set of manually segmented player regions. The likelihood $p_h(I|l_i, a_i)$ is computed at each pixel in the original image space based on the similarity between the reference histogram and the histogram defined by the player-sized rectangular region anchored to that pixel at the bottom center. These likelihoods are also projected back into field model space.

Unfortunately, the simple combination of these two models is imperfect because it can over count evidence. Some authors attempt to mitigate the effects of overcounting by applying a smoothing factor to the observation likelihood [17, 75]. However, we have found that this approach accentuates false peaks in the observation likelihood that are due to slight errors during registration with the field model. Instead, we apply a multiplicative reward term β_i to the observation likelihoods of players whose ideal locations make over counting the evidence associated with them unlikely. Thus, the final player likelihood $p(I|l_i, a_i)$ is the product of $p_b(I|l_i, a_i)$,

$p_h(I|l_i, a_i)$, and β_i, and the appearance parameters a_i for each player are the background color model, the HSV histogram model, and β_i.

4.4.3 Search strategies considered

In our experiments, we consider two different variants of best-first BBS. The first of these, referred to below as LB BBS, uses ordering relation $<_l$ and the lower bound function described in Section 4.3.2. Because a best-first search ordered by $<_l$ must consider all nodes V with $c_l(V) < C^*(I, V^*)$, the first leaf node drawn from the priority queue in LB BBS necessarily corresponds to V^*, and no nodes before V^* can be pruned. For this reason, we simply use constant ∞ as an upper bound function for LB BBS to avoid the cost of a more expensive computation. The second variant of BBS, referred to below as UB BBS, uses ordering relation $<_u$ and the lower and upper bound functions described in Sections 4.3.2 and 4.3.3, respectively. For comparison, we also consider exhaustive search and greedy hill-climbing as described at the end of Section 4.3.3.

4.4.4 Results

Table 4.1 summarizes the quantifiable statistics of the search procedures we consider for MoPPS inference. We use two different metrics to quantify error in predicted location, both of which compare the set of player types V^* and associated locations L^* inferred by the MoPPS model to a corresponding set of hand-labeled ground-truth player types and locations. The first metric, $e_c(V^*, L^*)$, computes the mean pixel distance between the locations of correctly classified players and the corresponding ground-truth locations. The second metric, $e_a(V^*, L^*)$, associates ground-truth locations with incorrectly classified players by finding the minimum matching between the locations of incorrectly classified players and ground-truth locations not associated with correctly classified players and then computes the mean pixel distance between the locations of all players and their associated ground-truth locations.

A far more important measure of performance in the football formation recognition domain is the percentage of correctly classified players. This is because the huge number of possible formations precludes naming all of them, so every football team uses their own language to describe formations. Player type information, however, can be translated into any team's formation language. Thus, the ability to correctly recognize which players are on the field along with special

Search Strategy	**Running time (min.)**			**Nodes Expanded**		
	Mean	Min.	Max.	Mean	Min.	Max.
LB BBS	4.35	0.48	11.18	392	51	988
UB BBS	9.00	1.44	24.14	412	57	1148
UB BBS, 1$^{\text{st}}$ Leaf	2.45	1.90	5.23	52	36	110
Greedy	0.57	0.53	0.63	11	11	11
Exhaustive	41.69	41.40	41.92	3264	3264	3264

Search Strategy	**% Correct Class.**	$e_c(V^*, L^*)$			$e_a(V^*, L^*)$		
		Mean	Std.	Max.	Mean	Std.	Max.
LB BBS	98.55 %	4.36	7.00	17.46	5.65	16.09	37.11
UB BBS	98.55%	4.36	7.00	17.46	5.65	16.09	37.11
UB BBS, 1$^{\text{st}}$ Leaf	92.00 %	4.72	7.96	27.20	9.36	23.27	66.07
Greedy	80.72 %	9.14	8.01	47.10	19.33	28.42	167.05
Exhaustive	98.55 %	4.36	7.00	17.46	5.65	16.09	37.11

Table 4.1: These tables summarizes the quantifiable statistics of various search strategies for MoPPS inference over the entire dataset of 25 images. Location error rates are in pixel units, six of which in our field model are equal to one yard on the football field. The optimal searches, LB BBS, UB BBS and exhaustive search yield excellent results in terms of both location error and classification rate. However, LB BBS provides a significant speedup over UB BBS, which is naturally much faster than exhaustive search. Halting UB BBS as soon as it encounters its first leaf node yields good results within a modest time frame, and greedy, approximate hill-climbing search yields reasonable results fairly quickly.

attributes of some (e.g. whether the quarterback is in shotgun formation) is akin to the ability to correctly recognize entire formations. This information, therefore, would be used to index plays in a coach's database.

Because best-first BBS is an optimal search, the location error rates and player classification rate for LB BBS and UB BBS are the same as those achieved through exhaustive search. By both measures, MoPPS inference with these methods is very accurate.

In particular, they achieve a mean location error rate of 4.36 pixels for correctly classified players. Considering that six pixels in our field model are equal to one yard on the football field, this error rate is quite good. Moreover, many of the higher individual errors we observed can be attributed to the fact that the low resolution of our imagery led the MoPPS inference method to locate some players at their waist instead of at their feet.

Even more striking is the 98.55% correct classification rate these methods achieved, repre-

senting a total of four misclassifications out of a possible 275 players. In each of of these cases, the misclassified player was placed either on a false peak in the observation likelihood around one of the several logos on the field, all of which are composed of colors identical to the ones in the players uniforms, or on a second peak in the likelihood generated by a single player.

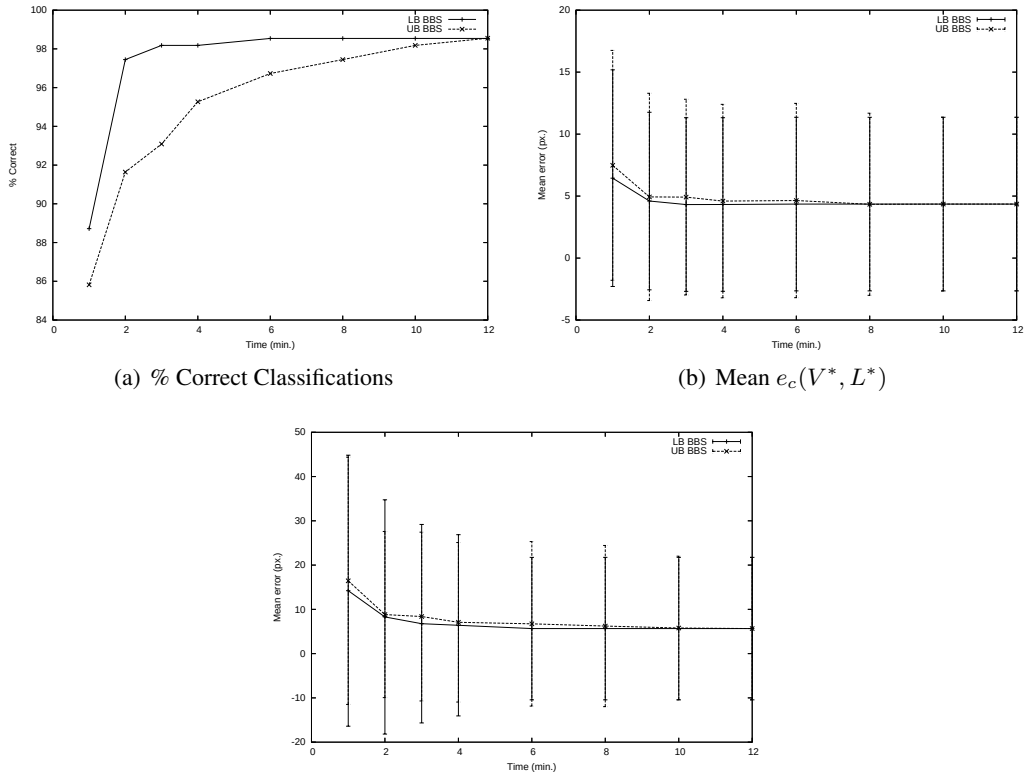
Of course, both LB BBS and UB BBS considerably outperform exhaustive search in terms of running time, with LB BBS beating UB BBS by about a factor of two. To obtain further insight on the running times of LB BBS and UB BBS, we measured their anytime behavior, which is plotted in Figure 4.4. Both search strategies perform similarly in terms of location error, achieving very low error rates with one minute of computation and near-optimal rates within two minutes. However, in terms of the percentage of correctly classified players, LB BBS consistently outperforms UB BBS for any given amount of computation time, again achieving near-optimal results within two minutes. This can be mostly attributed to the fact that UB BBS must spend additional processing time during the upper bound computation at every node searching for the optimal superset V_u and tightening the bound via pictorial structure minimization.

Overall, these results are very promising. While it is true that our dataset of 25 images directly represents only a small fraction of the over 3200 possible football formations, we are reassured by the fact that many other formations can be composed by combining correctly recognized pieces of our 25, suggesting the MoPPS model will likely work well for these other formations, too. In addition, informal evaluation on unlabeled images convinces us that the model is robust outside the test set.

4.5 Summary and Future Work

In this chapter we introduced the mixture-of-parts pictorial structure (MoPPS) model for recognizing object classes whose parts can vary in both location and type. We formulated a restricted but reasonable tree-structured representation of the MoPPS model and described how practically efficient inference could be performed on that model to jointly compute the most likely set of parts and their locations. Finally, we demonstrated the effectiveness of the model and inference procedure through experiments on recognizing initial player formations in American football.

We believe the MoPPS model will be generally useful whenever detailed internal object structure is needed and not just object existence/location. An important direction for future work will be to evaluate the merits of this model in terms of its expressiveness and computational speed on other recognition domains, such as furniture, which can be composed of various types



(a) % Correct Classifications

(b) Mean $e_c(V^*, L^*)$

(c) Mean $e_a(V^*, L^*)$

Figure 4.4: The plots above depict the anytime behavior of MoPPS inference with LB BBS and UB BBS over the entire dataset of 25 images in terms of (a) the percentage of correctly classified players and (b) & (c) the mean location error rates. For both strategies, a solution was computed using greedy, approximate hill-climbing search whenever a complete solution was not found in the alloted time. While both search strategies perform well in terms of location error, LB BBS clearly outperforms UB BBS in terms of the percentage of players it classifies correctly within a given amount of time. This is notable because classification accuracy the most important measure for our application.

of legs, arms, backs, rockers, etc.; the human figure, along with accessories such as hats, watches, footwear, etc., of which exponentially many combinations are possible; specific types of cars, which can have exponentially many combinations of different parts like spoilers, rims, etc.; and similarly for specific types of airplanes. In addition, we believe the MoPPS paradigm is particularly well suited for coping with occlusion during general object recognition and localization and would like to explore its capacity in this regard.

Many other directions for future work exist including extending MoPPS to richer representations than trees, such as k-fans [12]; incorporating richer sets of hard or nearly-hard constraints and logic-based reasoning; developing a part set prior that incorporates the image data to permit more efficient inference; and developing proposal distributions for MCMC sampling methods to allow for probabilistic queries (e.g. "what is the probability there is a tight end?"). There are also several opportunities to incorporate learning into the MoPPS paradigm, which we discuss in detail in [27].

Chapter 5: Player Tracking with Discriminatively Trained Particle Filters[1]

American football is one of the most challenging object tracking domains being studied today. There are several complicating factors in tracking football players, including the erratic movement of players, the complexity of interactions that sometimes involve upwards of ten players, a camera that rapidly moves to follow the action, clutter and camouflage on the football field, the near-identical appearance of players on the same team, and the strong dependence of player behavior on the player's type and the stage of the play. To date, there has been very limited success in developing tracking algorithms for American football.

In this chapter, we describe a conceptually simple particle filtering framework for multi-object tracking (Section 5.2), motivated by the problem of tracking players in American football. Particle filtering is a widely used framework for visual object tracking that is highly extensible and offers the flexibility to handle non-linearity and non-normality in the object models. In recent years, many new particle filter-based approaches have been proposed to solve difficult multi-object tracking problems [62, 38, 82]. However, most of this work has paid little attention to how to best tune the parameters of the proposed models and has instead relied on some combination of manual tuning and simple generative learning to set these parameters. These approaches, though, are often ineffective and/or extremely labor intensive.

The particle filtering framework we describe here is easy to extend and customize because it allows the user to define rich sets of features to capture essential properties of the tracking domain. Our main contribution is an error-driven, discriminative algorithm for training this model's parameters (Section 5.3). Our training is tightly integrated into the filtering process and attempts to optimize parameters in response to observed tracking errors. In addition, we describe an important practical approximation to this algorithm (Section 5.4) that can significantly reduce training time for domains where many objects must be tracked.

Our decision to use a discriminative approach is motivated by several factors. First, there is a growing body of empirical evidence from a number of fields [47, 1, 11, 83] along with theoretical results [56] suggesting that discriminative training outperforms generative training when enough data is available. Second, unlike generative training methods, our discriminative approach does

[1] The work described in this chapter was published in [25].

not make strong independence assumptions about the features, which would ignore important dependencies. Third, our discriminative approach is aimed at directly solving the problem we really care about: maximizing the accuracy of the learned filter. In contrast, generative training attempts to achieve this objective indirectly by maximizing the joint likelihood of the training data, which does not always relate well to filtering accuracy when the assumed model is wrong.

We apply our approach to tracking the 22 players and one referee in low-resolution American football video (Section 5.5). Our experiments show that some state-of-the-art multi-object tracking methods do not work well in this domain. In comparison, we show that filters trained using our method substantially outperform untrained filters and these other methods and, to the best of our knowledge, represent the state-of-the-art in tracking in the American football domain.

5.1 Related Work

Here we discuss related work on particle filter-based multi-object tracking and discriminative filter training.

Particle Filter-Based Multi-Object Tracking: Since particle filter-based visual object tracking was first proposed, much progress has been made on tracking single objects using particle filters. Tracking multiple objects, however, poses the challenge of dealing with the high-dimensionality of the state space, which grows with the number of objects. A naive solution is to use an independent single-object particle filter for each object, but this can break down when similar objects interact, leading to objects "hijacking" filters from other objects.

Some notable recent attempts to improve upon this naive approach include Okuma *et al.*'s boosted particle filter [62], which attempts to avoid hijacking by using a Haar-style object detector to terminate and resume tracks during and after an interaction; Khan *et al.*'s MCMC-based particle filter [38], which tracks objects in their joint state space and uses a Markov random field (MRF), built at each time step, that helps prevent hijacking by enforcing separation between nearby objects while allowing far apart objects to be tracked independently; and Yu and Wu's mean field Monte Carlo algorithm [81, 82], which also uses an MRF to enforce object separation but uses Monte Carlo variational inference.

While each of the above methods has been shown to outperform the naive approach, there is still much room for further improvement. For example, Okuma *et al.*'s method does not explicitly reason about object interactions, but rather attempts to improve on the purely naive approach by using more powerful proposal and observation distributions. The approach is prone to losing

object identities and locations when tracks are terminated. Khan *et al.*'s method tracks in the objects' joint state space and thus, in our experience, does not scale well when the number of objects is large and more uncertainty exists about objects' locations due, for example, to erratic object motion. Similarly, the joint inference approach employed in Yu and Wu's method suffers from quadratic complexity in the total number of particles used to track all of the objects. For both of these methods, it can be difficult to set the parameters of the interacting MRF model components to maximize accuracy, and they are currently set manually and/or learned generatively.

Discriminative Filter Training: There has been much recent work on discriminative training of sequential filters. For example, discriminatively trained conditional random fields (CRFs) for sequence data [43] have been shown to outperform generatively trained hidden Markov models (HMMs) in many domains. Collins has also proposed a generalized perceptron algorithm for sequential data [9] to train HMMs for natural language problems. This work has been further extended to large-margin discriminative training of HMM-style sequential models [73, 77]. Unfortunately, all of these methods assume small state spaces where exact, efficient filtering is possible, e.g. via dynamic programming, and hence are not directly applicable to object tracking.

In more recent work, discriminative training was used to set the noise parameters of continuous state extended Kalman filters (EKFs) for robot localization [1]. However, it is generally recognized that EKFs are not powerful enough for complex object tracking due to the normality assumptions they make. Our work can be seen as extending this approach to a more general class of process models.

In more closely related work, Limketkai *et al.* train large state-space CRFs for robot localization using Collins' perceptron algorithm within a particle filtering framework [47]. As far as we are aware, this is the only prior work to attempt any form of discriminative training of particle filters.

All of the above discriminative methods can be viewed as driving the learning process by iteratively using the current filter to perform inference on a set of training sequences and then updating parameters based on some measure of the disparity between the filter's output and the desired output. For example, Limketkai *et al.*'s approach computes a MAP state sequence for each training example using the current filter and then attempts to adjust filter parameters so that the ground truth sequences become more probable than the current MAP sequences. However because filters sometimes fail badly, often early in training, the filter parameter updates these approaches make can be dominated by the portion of the sequence that occurs after the failure

rather than focused on correcting the original point of failure. Our experience in multi-agent tracking suggests that such parameter updates can be counterproductive.

To address this issue, the perceptron algorithm has been extended to focus training directly on points of filter failure. For example, recent work has proposed performing updates based only on the part of the predicted sequence up to and including the first failure [10]. Later work has extended this idea by updating at successive points of failure [13, 80]. The learning approach we describe in Section 5.3 can be viewed as extending these failure-driven approaches to the particle filtering framework.

Finally, we note that there has been work that uses discriminative learning to produce certain individual components of particle filters, for example by learning object-detection classifiers to be included as part of the particle filter's proposal distribution [62]. However, such indirect learning approaches never take into account the actual filter performance and hence offer no guidance to improve the filter further after the original components have been learned. Our approach is complimentary, as such learned components could be included in a filter and the filter then further improved based on actual filtering performance.

5.2 Pseudo-Independent Log-Linear Filters

Below we first review the standard Bayesian particle filter, and then describe our specific particle filter-based multi-object tracking framework, which uses pseudo-independent filters parameterized by log-linear models.

Particle Filters: Particle filtering is a Monte Carlo approximation to the optimal Bayesian filter [15], which monitors the posterior probability of a first-order Markov process through the following formula:

$$p(\mathbf{x}_t|\mathbf{y}_{1:t}) \ = \alpha p(\mathbf{y}_t|\mathbf{x}_t) \int_{\mathbf{x}_{t-1}} p(\mathbf{x}_t|\mathbf{x}_{t-1}) p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}). \tag{5.1}$$

Here, $\mathbf{x}_t$ is the process state at time t, $\mathbf{y}_t$ is the observation, $\mathbf{y}_{1:t}$ is all of the observations through time t, $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ is the process dynamical distribution, $p(\mathbf{y}_t|\mathbf{x}_t)$ is the observation likelihood distribution, and α is a normalizing factor.

The integral in (5.1) does not have a closed form solution, except in the most basic cases, so particle filters are used to approximate (5.1) using a set of weighted samples $\{\mathbf{x}_t^{(i)}, \pi_t^{(i)}\}_{i=1,\ldots,n}$, where each $\mathbf{x}_t^{(i)}$ is an instantiation of the process state, known as a particle, and the $\pi_t^{(i)}$'s are the corresponding particle weights. Under this representation, the approximation to the Bayesian

filtering equation (5.1) is

$$p(\mathbf{x}_t|\mathbf{y}_{1:t}) \;\approx\; \alpha\, p(\mathbf{y}_t|\mathbf{x}_t) \sum_{i=1}^{n} \pi_{t-1}^{(i)}\, p(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)}). \tag{5.2}$$

To implement a standard particle filter, one must choose a state representation $\mathbf{x}_t$, which, in the case of object tracking might include object locations, scales, etc. In addition, one must design three distributions: the process dynamical distribution, $p(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)})$, which, in object tracking, describes how objects move between time steps; the proposal distribution, $q(\mathbf{x}_t^{(i)}|\mathbf{x}_{0:t-1}^{(i)}, \mathbf{y}_{1:t})$, which is sampled at each time step to update the particle distribution; and the observation likelihood distribution, $p(\mathbf{y}_t|\mathbf{x}_t^{(i)})$, which, in tracking, describes how objects appear within the video data, $\mathbf{y}_t$.

At each time step, given the previous particle set $\{\mathbf{x}_{t-1}^{(i)}, \pi_{t-1}^{(i)}\}$, a basic sequential importance resampling [15] particle filter updates the particles as follows:

1. Sample n particles $\mathbf{x}_{t-1}^{(i)}$ with replacement from current particle set according to probabilities $\pi_{t-1}^{(i)}$.

2. Generate an updated particle set by sampling from the proposal distribution,

$$\mathbf{x}_t^{(i)} \sim q(\mathbf{x}_t^{(i)}|\mathbf{x}_{0:t-1}^{(i)}, \mathbf{y}_{1:t}).$$

3. Reweight each particle according to the following formula and normalize so that the $\pi_t^{(i)}$ sum to 1:

$$\pi_t^{(i)} \propto \frac{p(\mathbf{y}_t|\mathbf{x}_t^{(i)})p(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)})}{q(\mathbf{x}_t^{(i)}|\mathbf{x}_{0:t-1}^{(i)}, \mathbf{y}_{1:t})}. \tag{5.3}$$

In tracking and and many other applications, it is typical to estimate the process state at each time step as the sample mean of the particles $\hat{E}[\mathbf{x}_t] = \sum_{i=1}^{n} \pi_t^{(i)} \mathbf{x}_t^{(i)}$.

Pseudo-Independent Filters: When multiple interacting objects must be tracked, the joint state space of those objects is high dimensional and renders the straightforward application of particle filtering impractical. As discussed in Section 5.1, extensions to the basic particle filtering framework have been explored that utilize more complex MRF dynamic models to capture important object interactions in a more tractable way. In contrast to those methods, we use a simple particle filtering framework for multi-object tracking. Specifically, we assign each object

of interest its own single-object particle filter. These filters are not completely independent, however, but are pseudo-independent, in the sense that each tracker estimates only a single object's state but has the previous state estimates of other trackers available as observations to use during inference.

Unlike the more sophisticated MRF approaches, our pseudo-independent approach does not increase the computational complexity of filtering over that of purely independent filters, but it still allows some amount of dependency between trackers through observations of one another's internal states from previous time steps. While the pseudo-independent approach is more restricted in the types of dependencies it can represent, we believe that it will be sufficient for many applications because the log-linear filtering framework described below enables straightforward representation of rich features of both individual objects and relations among objects, which the pseudo-independent filtering approach can exploit for improved performance.

Log-Linear Filters: To allow for maximum flexibility, we wish to allow the designer to devise arbitrary features that can capture joint properties of a tracked object's state, its observations, and the previous state estimates of other objects and to incorporate these features into the individual filters. For example, by including features that measure the distance between an object's proposed state and the predicted locations of other objects, it is possible to bias the filter against allowing two objects to occupy the same space.

While in principle one can attempt to define the dynamic and observation distributions in terms of such features, doing so is quite difficult in practice. In particular, one must decide how to weight the potentially many features against one another and/or make the assumption that features are independent in order to support traditional generative learning. In this work, we instead utilize a more flexible combination of features based on log-linear modeling.

From an algorithmic perspective, the main difference between our particle filtering framework and the traditional framework is in the way we compute particle weights. The traditional reweighting computation in equation (5.3) is the result of trying to account for dynamics, observations, and proposal bias all at once by combining their associated generative model terms. In contrast, we attempt to learn a single reweighting function defined in terms of arbitrary features, which can encompass all of those terms but is not restricted to the specific form of equation (5.3).

In particular, our particle weighting function takes the form of a log-linear combination of weighted features:

$$\pi_t^{(i)} \propto \exp\Big(\sum_j w_j f_j(\mathbf{y}_t, \mathbf{x}_t^{(i)})\Big). \tag{5.4}$$

Here the $f_j(\cdot)$'s are user-defined feature functions and the w_j's are the weights of the features, which parameterize the model. By including features that correspond to the logarithms of the process dynamics, observation likelihood, and proposal distribution terms in equation (5.3), the log-linear model can be made strictly more expressive than the traditional formulation. Note that, for our pseudo-independent filters, each $\mathbf{y}_t$ contains traditional observation data as well as information about the previous states of other filters to allow the features to model interactions between objects. In Section 5.5, we describe a number of feature functions for multi-object tracking in the American football domain.

There are two ways to view this log-linear filtering model. From an algorithmic perspective, it can be seen as a more flexible parameterization of the standard particle filtering algorithm. From a modeling perspective, it can be viewed as an undirected, log-linear probabilistic model over sequences—as in the work on CRF-filters [47]—for which a particle filtering inference procedure based on non-parametric belief propagation can be derived. In either case, the ultimate goal is to learn feature weights that optimize filter performance as described next.

5.3 Error-Driven Discriminative Filter Training

We take a supervised approach to training our individual log-linear particle filters. The training set contains examples of the form $(\mathbf{x}^*_{0:T}, \mathbf{y}_{1:T})$, where $\mathbf{y}_{1:T}$ is an observation sequence, for example, a video of a football play, and $\mathbf{x}^*_{0:T}$ is the ground-truth or target state sequence, for example, the trajectory of a particular player in the football play. The goal is to optimize the filter weights so that the filter output is as close to the target state sequences as possible.

The basic training approach iterates through the training examples and, for each one, calls Algorithm 4, which produces updated filter weights that are used as the initial weights for the next call. The iteration continues until a maximum number of steps is reached or performance no longer improves. In practice, we average the weights learned across iterations to arrive at the final weight vector, which is a common technique that has been shown to improve performance of online learning [9].

Algorithm 4 is identical to the log-linear filter of Section 5.2 with the addition of training mechanisms in lines 6 through 12. Each call to this algorithm monitors the filter's performance on the current example and tunes the feature weights each time the filter fails. Specifically, at each time step, a new particle set is proposed and reweighted according to the log-linear model. After particle reweighting, the filter's current state estimate $\hat{\mathbf{x}}_t$ is compared to the ground truth

Algorithm 4 Error-driven particle filter training

Input: $(\mathbf{x}^*_{0:T}, \mathbf{y}_{1:T})$ – Training sequence
$\mathbf{w}$ – Input feature weight vector
γ – Learning rate
λ_u – Update threshold
λ_r – Restart threshold ($\geq \lambda_u$)

1: Initialize particles: $\mathbf{x}_0^{(i)} = \mathbf{x}_0^*$ and $\pi_0^{(i)} = \frac{1}{n}$, $i = 1, \ldots, n$
2: **for** $t = 1$ to T **do**
3: Sample n particles $\mathbf{x}_{t-1}^{(i)}$ from current particle set
4: Sample new particle set from proposal distribution:
$\mathbf{x}_t^{(i)} \sim q(\mathbf{x}_t^{(i)} | \mathbf{x}_{0:t-1}^{(i)}, \mathbf{y}_{1:t})$
5: Reweight each particle using current weights:
$\pi_t^{(i)} \propto \exp\left(\sum_j w_j f_j(\mathbf{y}_t, \mathbf{x}_t^{(i)})\right)$
6: Compute $\epsilon = ||\mathbf{x}_t^* - \hat{\mathbf{x}}_t||$, where $\hat{\mathbf{x}}_t$ is the current state estimate
7: **if** $\epsilon > \lambda_u$ **then**
8: Update feature weights $\mathbf{w}$ (one of equations 5.5, 5.6, or 5.8)
9: **end if**
10: **if** $\epsilon > \lambda_r$ **then**
11: Reset particle filter to ground truth state at time t by setting all particle states to $\mathbf{x}_t^*$
12: **end if**
13: **end for**

state $\mathbf{x}_t^*$, and, if the distance between the two is above a user-specified threshold λ_u, a weight update is performed (see below). Additionally, if the distance between $\hat{\mathbf{x}}_t$ and $\mathbf{x}_t^*$ is greater than a second threshold λ_r, the filter is reset to the current ground truth state.

By updating the feature weights when an error is made during inference and by restarting the filter when a large enough error is made, the filter is allowed to focus its attention on meaningful points of failure, resulting in more purposeful weight updates. In addition, performing successive restarts can be seen as reducing training time by allowing the filter to encounter a variety of errors within a training iteration. If instead the filter is never reset for large errors, which is essentially Limketkai *et al.*'s approach [47], many weight updates would be performed on wildly diverging state estimates due to cascading errors. We have found this approach to work poorly in our tracking domain where filters often diverge badly in earlier training iterations.

Using the generic error-driven approach described above, we can optimize a number of discriminative objective functions by specifying appropriate weight update equations (line 8), along

with the type of state estimate $\hat{\mathbf{x}}_t$ to be used (line 6). In what follows, we specify these choices for three different objective functions.

Perceptron Updates: Collins' perceptron update [9] can be viewed as an approximation to the gradient of the conditional log-likelihood of the training data $\log p(\mathbf{x}^*_{0:T}|\mathbf{y}_{1:T})$ with respect to the feature weights, and thus, the perceptron algorithm can be thought of as attempting to maximize this probability via approximate gradient ascent.

The batch perceptron update of Collins can easily be modified into an iterative update for use in our error-driven training approach. The update to weight w_j takes the form:

$$w_j \;=\; w_j + \gamma \left(f_j(\mathbf{x}^*_t, \mathbf{y}_t) - f_j(\hat{\mathbf{x}}_t, \mathbf{y}_t)\right), \tag{5.5}$$

where γ is the learning rate. In this case, the state estimate $\hat{\mathbf{x}}_t$ used by the filter is taken to be the particle with the highest particle weight (i.e. the MAP state estimate). Thus, filters learned with the perceptron update aim at making the filter's MAP state close to ground truth at every time step.

Minimizing the Squared Residual of Mean: In the case of object tracking, the perceptron update is somewhat unsatisfactory because it places emphasis on the MAP particle instead of on the sample mean $\hat{E}[\mathbf{x}_t]$, which is typically used as the state estimate in particle filter-based object tracking and in our experience typically leads to more robust tracking. To remedy this, we can use an error-driven learning instantiation where the sample mean is taken as the state estimate (i.e. $\hat{\mathbf{x}}_t = \hat{E}[\mathbf{x}_t]$), and the objective function is to minimize the squared residual of the sample mean $||\hat{E}[\mathbf{x}_t] - \mathbf{x}^*_t||^2$. We update the weights upon each error in the gradient direction of this objective, which for weight w_j yields the following update:

$$w_j \;=\; w_j + \gamma \left(\hat{E}[\mathbf{x}_t] - \mathbf{x}^*_t\right)^{\mathsf{T}} \left(\hat{E}[\mathbf{x}_t f_j(\mathbf{x}_t, \mathbf{y}_t)] - \hat{E}[\mathbf{x}_t]\hat{E}[f_j(\mathbf{x}_t, \mathbf{y}_t)]\right). \tag{5.6}$$

Using this update, w_j converges when the state $\mathbf{x}_t$ is uncorrelated with feature $f_j(\cdot)$ according to the sample distribution. Using this update in our error-driven framework can be viewed as performing stochastic gradient descent on the truncated squared residual, which is set equal to zero when the residual is less than our update threshold λ_u.

Minimizing Mean Squared Error: The above update has the unfortunate property of ignoring the sample variance $\widehat{\mathrm{var}}(\mathbf{x}_t)$ of the state $\mathbf{x}_t$. In particular, we have

$$||\hat{E}[\mathbf{x}_t] - \mathbf{x}^*_t||^2 \;=\; \hat{E}\left[||\mathbf{x}_t - \mathbf{x}^*_t||^2\right] - \widehat{\mathrm{var}}(\mathbf{x}_t). \tag{5.7}$$

This shows that it is possible to minimize the mean's residual while both the mean squared error (MSE) and the variance take on very large values, which is quite undesirable. For example, in object tracking this would be akin to minimizing the residual to a ground truth location at the center of the video frame by placing an equal number of equally weighted particles at each of the four corners of the frame.

Thus, we here consider minimizing the MSE $\hat{E}\left[||\mathbf{x}_t - \mathbf{x}_t^*||^2\right]$ as a potentially more robust objective, which from (5.7), corresponds to minimizing the sum of the mean's squared residual and the sample variance. In tracking, minimizing this objective corresponds to having all of the particles bunched tightly around the ground truth location. Differentiating $\hat{E}\left[||\mathbf{x}_t - \mathbf{x}_t^*||^2\right]$ with respect to w_j yields the following gradient descending weight update:

$$w_j \;=\; w_j + \gamma\left(\hat{E}\left[||\mathbf{x}_t - \mathbf{x}_t^*||^2 f_j(\mathbf{x}_t, \mathbf{y}_t)\right] - \hat{E}\left[||\mathbf{x}_t - \mathbf{x}_t^*||^2\right]\hat{E}\left[f_j(\mathbf{x}_t, \mathbf{y}_t)\right]\right). \tag{5.8}$$

This update has the intuitively satisfying property of converging when the MSE is uncorrelated with feature $f_j(\cdot)$. Again, using this update, our error-driven training process can be viewed as stochastic gradient descent on a truncated version of MSE.

5.4 Improving Training Computation Time

Since each iteration of error-driven training involves running the particle filter once for each training example, this approach can become computationally expensive when each run of the particle filter has a non-trivial computational cost. This is particularly true in tracking domains in which a large number of objects must be tracked at once, necessitating many calls to feature functions involving expensive pixel-level operations on sizable regions of the video frame, which form a computational bottleneck.

In order to overcome the computational burden associated with performing many of these calculations repeatedly, we use an approximate method that operates on strategically drawn state samples with pre-computed values for these features. Specifically, we pre-process each training example $(\mathbf{x}_{0:T}^*, \mathbf{y}_{1:T})$ by drawing a large set of samples $\{\tilde{\mathbf{x}}_t^{(k)}\}_{k=1,\ldots,N}$ normally distributed around each of the ground truth states $\mathbf{x}_t^*$ with large variance and, for each of these samples, pre-computing the values of all features involving expensive pixel-level operations.

Then, during training, the particle filter is modified by only allowing the proposal distribution to propose states at time t that are represented in $\{\tilde{\mathbf{x}}_t^{(k)}\}_{k=1,\ldots,N}$. This can be implemented by

turning the continuous proposal distribution into a discrete distribution in a straightforward way. Specifically, given the state history $\mathbf{x}_{0:t-1}$ and the observation sequence $\mathbf{y}_{1:t}$ at time t, the filter makes a proposal $\acute{\mathbf{x}}_t^{(i)}$ for each particle $\mathbf{x}_{t-1}^{(i)}$ according to the normal proposal rules. Each $\acute{\mathbf{x}}_t^{(i)}$ is then compared to the samples in its own sample set and those in the sample sets of all objects within a certain distance, and the proposal is selected as the closest sample, i.e.

$$\mathbf{x}_t^{(i)} \;=\; \arg\min_{\tilde{\mathbf{x}}_t^{(k)}} ||\acute{\mathbf{x}}_t^{(i)} - \tilde{\mathbf{x}}_t^{(k)}||.$$

In this way, during the particle re-weighting step, the available pre-computed feature values are used for all of the particles, while all other features are computed online.

This approach allows training to proceed significantly faster than when using the full particle filter while still providing a close approximation to the way the full filter performs. Performance may suffer in tracking domains where most errors are made due to background clutter and other factors rather than to hijacking by other objects. However, in such domains, it is possible to use a slightly modified version of this method that generates sample sets during actual runs of the tracker and iteratively augments those sets after they have been used for several iterations of training. We are currently investigating such an approach.

5.5 Experiments and Results

We test our approach by tracking the 22 players and one referee in low-resolution videos of American football. Each video contains footage of a single football play shot from a panning, tilting, and zooming camera with a sideline view high above the center of the field. Figure 5.1 depicts a typical view from this camera. Every video is pre-processed to register frames to an overhead model of the football field using the method described in Chapter 2, thereby enabling us to determine players' locations in football field coordinates.

The football domain is an extremely challenging testbed for any multi-object tracking algorithm. The players being tracked in this domain move very erratically, and the characteristics of players' motion change substantially depending on the player's type and the time stage of the video. In the first several seconds of each video, for example, nearly all of the players stand virtually still, while perhaps one or two move only gradually. However, after the ball is snapped (a one-time event that occurs in every football play), all of the players begin to move very quickly, and the tracker must be able to adapt accordingly. In addition, football players interact in com-

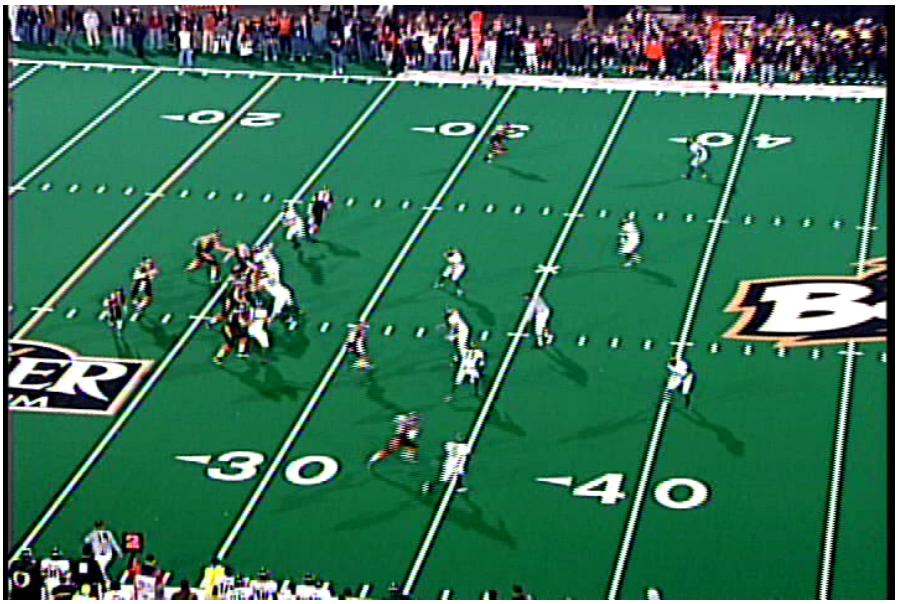


Figure 5.1: A typical video frame from our dataset.

plex ways, frequently in very large groups. For example, in every play, a group of five linemen on the offense stand shoulder to shoulder in a line and attempt to block a group of about three to five defensive players who, in turn, attempt to break through the offensive line. Many other complicated interactions take place between smaller groups of players throughout the course of a play. In what follows, we first describe the filter models and feature functions we use to capture these interactions and other salient aspects of the football domain. We then move on to present the results of our experiments.

5.5.1 Football domain modeling

In the football domain, we represent each player as a rectangular region defined by his location in football field coordinates and the scale of the region (i.e. $\mathbf{x} = \{x, y, s_x, s_y\}$). Players' motion is modeled using the second-order autoregressive dynamical model that is common in the tracking literature, in which $\mathbf{x}_t \sim \mathcal{N}(g(\mathbf{x}_{t-1}, \mathbf{x}_{t-2}), \Sigma_d)$, where $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$ is the normal distribution with mean $\boldsymbol{\mu}$ and covariance Σ,

$$g(\mathbf{x}_{t-1}, \mathbf{x}_{t-2}) \;=\; A_1\mathbf{x}_{t-1} + A_2\mathbf{x}_{t-2}, \tag{5.9}$$

and A_1 and A_2 define a constant acceleration model.

The proposal distribution we use is slightly non-standard. Specifically, it takes the form of the process dynamical distribution above but uses the previous sample means in place of the state values $\mathbf{x}_{t-1}$ and $\mathbf{x}_{t-2}$. In other words, the proposal distribution we use is,

$$\mathbf{x}_t \sim \mathcal{N}(g(\hat{E}[\mathbf{x}_{t-1}], \hat{E}[\mathbf{x}_{t-2}]), \Sigma_d), \tag{5.10}$$

where $g(\cdot)$ is as in (5.9). We have found that this form of proposal distribution works well in practice and is more stable than the motion model.

We use a variety of feature functions to describe different aspects of players' appearances, their motion, and the interactions between players. In order to model motion patterns that change depending on the time stage of the football play, each feature we describe below is replicated once for each possible time stage. All features that do not correspond to the current time stage take on zero values. In all, each filter uses 15 base features, described below, and 4 time stages for a total of 60 features.

Player motion features: We use 11 different features to describe player motion. The first of these is the logarithm of the Gaussian probability density value of the player state $\mathbf{x}_t$ evaluated under the proposal distribution. The second is the negative squared distance between the player's current state and the state estimate from the previous time step, i.e. $-||\mathbf{x}_t - \hat{E}[\mathbf{x}_{t-1}]||^2$. Intuitively, these features allow for a trade-off between rewarding a particle for being close to the proposal's prediction and not straying too far from the player's previous location.

We also use a set of binary features that indicate in which of the eight compass directions a player is moving. These are calculated by quantizing the player's motion vector $(\mathbf{x}_t - \hat{E}[\mathbf{x}_{t-1}])$. These features allow us to model motion tendencies that occur regularly across all football plays. For example, the quarterback nearly always moves backwards immediately after the ball is snapped.

Appearance features: We use two features to describe players' appearances. The first of these is an RGB histogram-based feature, which is calculated based on the Bhattacharyya coefficient-based histogram distance (as in [63]) between a reference histogram and the histogram of the region defined by $\mathbf{x}_t$. The second appearance feature is based on background modeling and is computed by using a pre-computed background color model to count the number of foreground pixels inside the player region defined by $\mathbf{x}_t$ to the total area of that region.

Player interaction features: The final two features describe interactions between players.

The first of these is similar to the MRF edge potential used in [38], which penalizes overlap between player regions in the video frame. The second is similar to the MRF edge potential used in [81], which penalizes proximity between players in state space. Intuitively, these features cause trackers to "repel" one another, thereby helping them to to better cope with interactions between players by avoiding hijacking.

5.5.2 Results

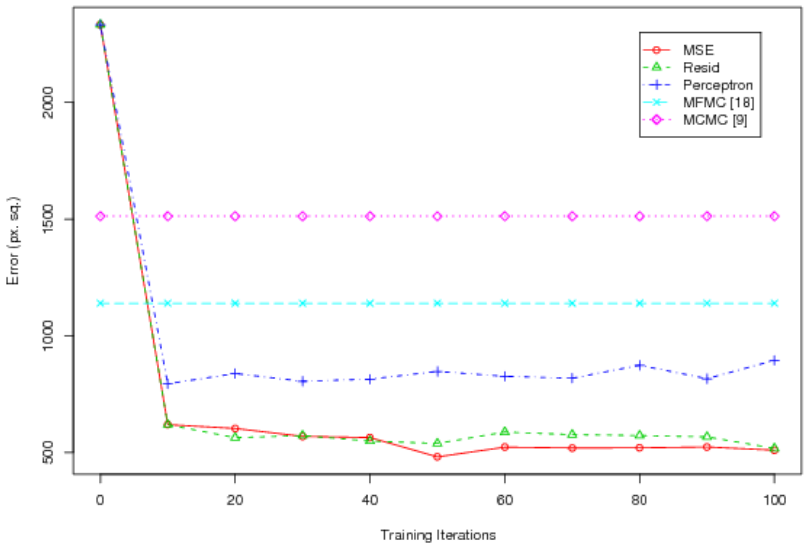
Our dataset contains 20 videos of different football plays, each around 400 frames long, along with hand-labeled ground truth data for every player in every frame[2].

We used a four-fold cross validation approach to evaluate our training method on this dataset. Specifically, we divided the entire set into four folds of five videos each and, for each fold, trained trackers on the other three folds over 100 iterations of the method described in Section 5.4 using each of the weight updates in equations (5.5), (5.6), and (5.8). To account for the different characteristics of various player types, we learned 13 separate sets of weights, one for each individual player type, with filters for players of the same type sharing the same set of weights in both tracking and learning.

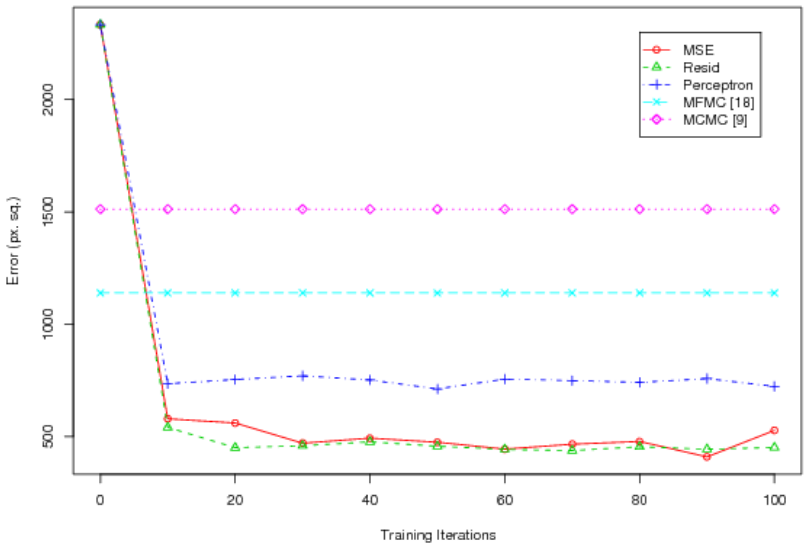
We evaluated each fold after every 10 training iterations using the full pseudo-independent log-linear filtering model with averaged feature weights for each player type. In addition, we also evaluated each video using Khan *et al.*'s MCMC-based particle filter [38] and Yu and Wu's mean field Monte Carlo algorithm [81], two state-of-the-art multi-object particle-filter trackers. For all of these methods, player types and initial locations are computed automatically using mixture-of-parts pictorial structures as described in Chapter 4. We also tried to use the CRF-Filter perceptron algorithm of Limketkai *et al.* [47] on our data set and found that it did not converge well in training. We believe that this is due to the non-incremental nature of the updates, which are based in large part on extremely erroneous filtering runs.

Figures 5.2 and 5.3 compare the performance of pseudo-independent log-linear filters trained using the weight updates in equations (5.5), (5.6), and (5.8) and of Khan *et al.*'s and Yu and Wu's methods. Figure 5.2 depicts the mean squared per-frame pixel error $(\hat{\mathbf{x}}_t - \mathbf{x}_t^*)^2$, while Figure 5.3 depicts the failure rate, which is the proportion of frames in which a player was considered to be lost (i.e. in which the tracker is at least 5 yards in error). Results are given averaged over

[2]Our entire hand-labeled tracking dataset is available at
`http://eecs.oregonstate.edu/football/tracking/dataset`.

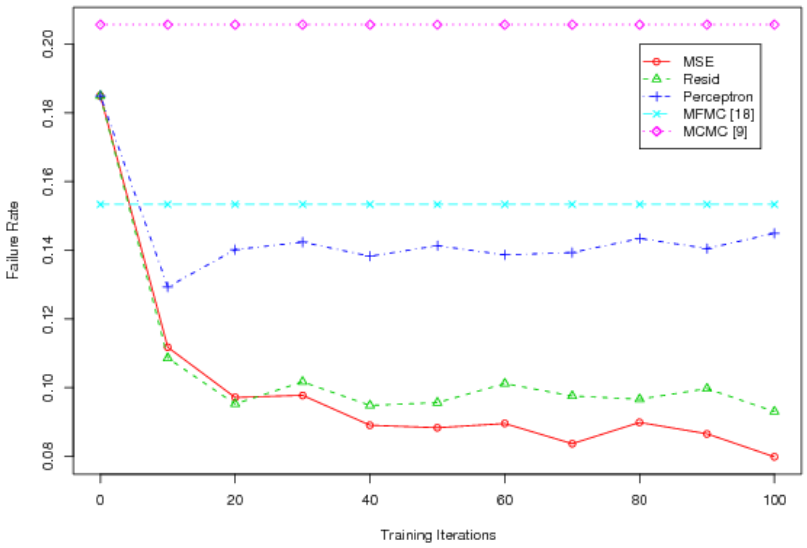


(a) Testing folds

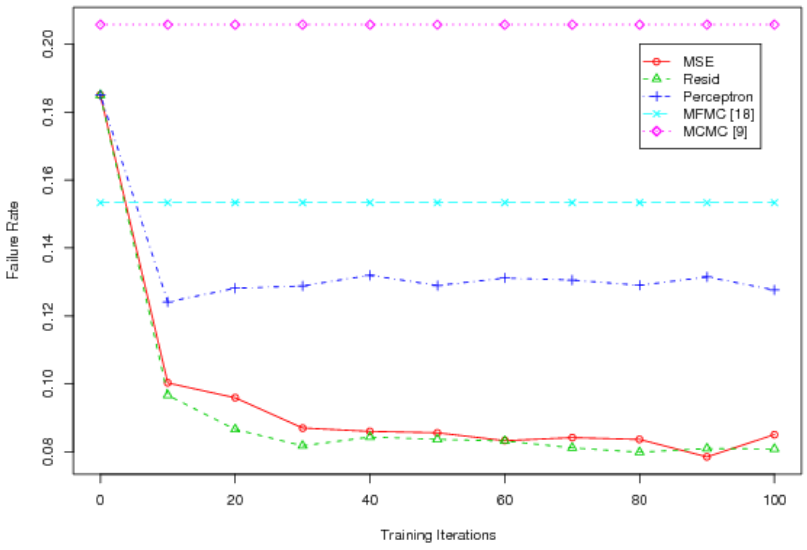


(b) Training folds

Figure 5.2: Mean squared per-frame pixel error at every 10 training iterations, averaged over all players in (a) the testing and (b) the training folds for each of the three weight update equations (5.5), (5.6), and (5.8). Included for reference are MSE values for Yu and Wu's method [81] and Khan *et al.*'s method [38].



(a) Testing folds



(b) Training folds

Figure 5.3: Failure rate at every 10 training iterations, averaged over all players in (a) the testing folds and (b) the training folds. Failure rate is computed as the proportion of a player's frames in which the player was lost by the tracker.

all players for both the training and testing folds.

First, these results make it clear that the training algorithm is able to significantly reduce the training error, indicating that the optimization approach is effective. Further, by both error measures, pseudo-independent log-linear filters trained using each of the weight updates achieve substantially better results than untrained pseudo-independent log-linear filters (iteration 0) and the previous state-of-the-art methods on the test set. It is especially important to note that the significantly lower lost-player rates attained by our filters indicate that our discriminative error-driven training approach is effective in achieving its goal of improving filter performance by learning to overcome failures. This also suggests that the approximate method described in Section 5.4 is a valid approximation to the full tracking process.

We also see that the MSE and residual updates perform better than the perceptron updates on this data, with the MSE update having an edge in peak accuracy. As is common for incremental learning methods, the error does not monotonically decrease. In practice one would use a validation set to select the best stopping point for training.

As an additional assessment of tracking performance, we also counted the number of players that were never lost by the tracker during testing. Interestingly, our untrained filters lost every player for at least a very small portion of the video, while Yu and Wu's and Khan *et al.*'s methods were successful on 174 and 119 players (out of a total of 440), respectively. In comparison, the best performing filters trained with the MSE update, the residual update, and the perceptron update were successful on 280, 264, and 223 players, respectively. Examining these players further, we found the best performing trained filters achieved an average per-frame MSE of 74.0 (MSE update), 76.3 (residual update), and 82.4 (perceptron update), while Yu and Wu's and Khan *et al.*'s methods respectively achieved average per-frame MSEs of 110.0 and 100.1. These results show that our trained filters can fully track a significantly larger number of players more accurately. In our model 6 pixels corresponds to one yard on the field, so the MSE and residual updates achieve an error of approximately 1.5 yards for these players. In our experience, this is more than enough to infer many types of player behavior.

Average run times for pseudo-independent log-linear filters, Yu and Wu's method and Khan *et al.*'s method are summarized in Table 5.1. We can see that, in addition to improved filter performance, pseudo-independent log-linear filters also offer faster tracking times.

	PILLFs	Yu and Wu	Khan *et al.*
Avg. run time	18.30 m	30.55 m	84.50 m

Table 5.1: Average run time in minutes for pseudo-independent log-linear filters, Yu and Wu's method [81], and Khan *et al.*'s method [38]. These are computed over the entire football dataset on a standard desktop computer.

5.6 Summary and Future Work

We have described a framework for multi-object tracking based on pseudo-independent log-linear particle filters. Our main contribution is an error-driven discriminative training algorithm for this model. We gave results in the domain of American football that show the effectiveness of the method in comparison to recently proposed multi-object trackers. To our knowledge, our learned trackers are state-of-the-art in the football domain. In future work, we plan to extend our approach to learn the parameters of more complex MRF-style models to allow for more significant joint inference about object interactions. In addition, we plan to study the theoretical convergence of our error-driven algorithms.

Chapter 6: Conclusion

In this dissertation, I described novel mid-level computer vision algorithms for accomplishing several sub-tasks within a larger system for understanding American football in video. In particular, I addressed the problems of video registration (Chapter 2), video mosaicing (Chapter 3), football formation recognition (Chapter 4), and football player tracking (Chapter 5). For video registration, I described a method based on localized matching of invariant image features and an empirical stability test that we use to initialize our registration procedure. For video mosaicing, I described a greedy, utility maximization-based approach and explored a particular class of utility functions that attempt to maximize the scene area covered by the mosaic. For formation recognition, I described mixture-of-parts pictorial structures (MoPPS), an extension of classical pictorial structures designed to recognize objects whose parts can vary in both location and type. Finally, for player tracking, I described a particle filter-based formulation and an error-driven discriminative training procedure. I demonstrated the effectiveness of each of these methods with in-depth experiments in the American football domain.

We believe that the video registration, video mosaicing, and formation recognition methods described here are suitable solutions to the research problems for which they were designed. Of course, more could always be done to improve upon any of these methods: our video registration approach could be extended to use line- and region-based features to handle cases where image gradient-based features, like those computed by SIFT, are unavailable; additional classes and instances of utility functions could be explored for our video mosaicing approach; or the MoPPS model could be converted into an undirected log-linear probabilistic model (i.e. a conditional random field) to allow a richer set of spatial and appearance features to be incorporated and a model over these features discriminatively learned, as we touched on in [27]. Indeed, in order to advance these methods out of the laboratory and into production, many details such as these will certainly need to be resolved.

In contrast to these other three methods, however, the tracking approach described in Chapter 5, while still representing a legitimate advance in the state of the art of multi-object tracking and likely an acceptable solution to many less challenging tracking problems, has not yielded results for our football video data set sufficient enough to call it an acceptable solution to the football

player tracking problem, and more work on this problem is necessary. Thus, our most recent efforts have been directed towards improving upon our tracking results in the American football domain.

Based on our reflection upon the results we've obtained with our particle filter-based tracking approach, we believe this approach's main failing lies in its short-sightedness. Specifically, because the particle filter is first-order Markovian, that is, because it computes player location estimates on a frame-to-frame basis and does not incorporate longer-term information about players' trajectories, it is prone to being "hijacked" by temporary distractions in the evidence, such as peaks in appearance-based features around field markings or other players. If such distractions persist long enough to occupy a player's tracker for several video frames, the player may move too far away in the meantime for the tracker to be able to find him and recover from its error after the distracting factors fade. Training our filters using the discriminative procedure we describe in Chapter 5 can help them overcome their short-sightedness to some degree, as evidenced by the significant improvement in their performance after training. However, even after training, hijacking errors are still not uncommon in our filters' results in the football domain.

We believe that, to succeed in the American football domain, we need a tracking approach that takes a longer-term view, one that can look forward (or backward) to reason through the situations that typically lead to particle filters being hijacked. Such an approach could either attempt to ignore the factors that lead to hijacking given enough evidence on either side of those factors, or it could attempt to recognize and recover from an instance of hijacking after it has occurred.

We have experimented with several approaches in an attempt to find a suitable long-view tracking algorithm. I discuss a few of these in Appendix A, including a promising approach based on limited-discrepancy search (LDS) [14], which attempts to identify errors made by out particle filter trackers and to fix those errors by placing location constraints on specific players' trackers. Before concluding, I will review some of the more recent efforts on multi-object tracking from the literature.

6.1 Recent Related Work on Multi-Object Tracking

There have been two interesting and significant thrusts in the recent literature on multi-object tracking. In particular, one major body of work addresses the problem of tracking in very crowded, unstructured scenes [41, 65, 52, 3, 2]. Another major body of work has explored

similar graph-based formulations of the general multi-object tracking problem [4, 79, 64, 42, 46, 84, 29, 37]. Aspects of both of these bodies of research are relevant to the football player tracking problem.

Some of the work on tracking in crowded scenes focuses only on the setting where a large crowd of people, such as runners in a marathon, are all moving in the same general direction [41, 3, 2]. While this work is interesting, it does not apply directly to the football domain, where players move in a more chaotic, irregular fashion. Other efforts on this topic, however, are more interesting in the context of football player tracking.

Mehran *et al.*, for example, investigated crowd behavior using a social force model that attempted to explain how people behave as they interact with one another in a crowd [52]. Mehran *et al.* use a method called particle advection, in which a grid of particles is overlaid on top of the video image, and each one is moved according to the average velocity of optical flow vectors in its surrounding neighborhood. They use differences between the local motion of individual particles and the average motion of a neighborhood of particles to compute a local interaction force that causes objects to move out of sync with the rest of the crowd. Rodriguez *et al.* also take an approach in which local optical flows are compared against learned crowd behaviors in order to make local predictions about object motion [65]. A method comparable to either of these latter two may be useful in the football domain for helping to model the interactions of individual players within larger groups.

A larger and perhaps more significant body of recent years' work has examined slightly different variations on a particular graph representation for the multi-object tracking problem. In particular, each of these recent papers [4, 64, 42, 46, 84, 29, 37] describes a method for tracking in which a graph is built on top of a set of object detections. Typically, nodes in these graphs correspond to the object detections, either individually or in sets, and edges correspond to potential transitions between detections/detection groups. Both nodes and edges in these graphs have some form of associated weight or cost to measuring the value of including the corresponding detection/detection group or transition in an object trajectory, and the goal is to find high-weight or low-cost paths through the graph. These paths are then output as object trajectories. Along with implementation details, the main differences between these methods lie in the way they perform graph inference.

Ram Nevatia's research group at University of Southern California has published several methods all utilizing the same basic graph structure for tracking [42, 46, 84, 29]. In this model, nodes correspond to object detections, with each node's weight representing an appearance-

based measure of the likelihood of the corresponding detection being truly generated by an object. Edges in this model link detections in contiguous frames between which an object could feasibly transition, and edge weights measure the plausibility of an object transitioning between the two linked detections. The method described in [84] determines object trajectories by finding paths through this graph using a network-flow algorithm, while the methods described in [42, 46, 29] all use an *ad hoc* inference procedure based on the Hungarian algorithm. The method described in [46] additionally uses an AdaBoost-based procedure to learn an edge weighting function, while [42] employs a different AdaBoost-based procedure to learn a node-weighting function.

Berclaz *et al.* [4], Pirsiavash *et al.* [64], and Jiang *et al.* [37] all use a graph formulation nearly identical to the one used by Nevatia's group. Jiang *et al.* [37] include extra nodes in the graph to facilitate occlusion reasoning and convert this graph into a linear program, which they solve using a standard LP solver. Pirsiavash *et al.* [64] cast the graph as a network flow problem and employ a dynamic programming-based successive shortest paths algorithm for inference. Berclaz *et al.* [4] propose both a linear programming formulation and a k-shortest paths formulation to inference in this graph structure and show that the k-shortest paths formulation is the less computationally complex of the two. The authors of all the papers mentioned in this paragraph and the previous one provide more-or-less compelling results in the pedestrian tracking domain.

Unfortunately, because all of these methods employ a common graph structure, they all share a common flaw that restricts their applicability to tracking American football video. Specifically, because each of these methods operates on a single graph structure, in which each node and each edge has a single associated weight/cost, it is extremely difficult to incorporate object type-specific characteristics into the overall trajectory scoring mechanism. In the American football domain, however, player type-specific characteristics are extremely important. In particular, because appearance is nearly identical among football players on the same team, player type-specific information—such as motion and interaction tendencies—is sometimes the only way to disambiguate between players after they engage in a complicated interaction. Thus, while each of these approaches satisfies our desire for long-view reasoning about trajectories, none of them are ultimately suitable for our work in American football tracking. Informal experiments we have conducted using some of these approaches has, in fact, borne this out.

In [79], Wu *et al.*, describe a tracking approach that uses a graph structure slightly different than the one used by the above methods. Wu *et al.*'s approach constructs tracklets by group-

ing object detections that unambiguously form spatio-temporally consistent chains across a set of contiguous video frames. These tracklets are detected using basic Kalman filter trackers. Specifically, Wu *et al.*'s method processes a video forward in time, and each time a detection is encountered that is not accounted for by an existing Kalman filter, a new Kalman filter is started at the location of that detection. Then, each existing Kalman filter makes a prediction into the next frame, and each detection that unambiguously matches a single Kalman filter's prediction is assigned to the tracklet corresponding to that filter.

The nodes in Wu *et al.*'s graph structure correspond to these tracklets. The edges in their graph, then, represent possible links between tracklets and are computed based on tracklet merges and splits detected during the Kalman filtering step. Inference in this graph entails linking tracklets together into trajectories. To achieve this, Wu *et al.* use a greedy set cover-based method in which they enumerate all possible paths through the graph, compute a cost for each path based on its spatio-temporal smoothness (they do not use an appearance-based term in their cost computation), and then greedily select paths in such a way as to achieve a maximal covering of the set of non-spurious detections that accumulates the least possible cost. They achieve fairly compelling results on a data set of infra-red videos of bats exiting a cave. Unfortunately the enumeration of all possible paths performed by this algorithm appears to be prohibitively expensive. In addition, because of the nature of the graph structure this method employs, this method, too, will suffer from the same inability to incorporate player type characteristics as the methods above.

6.2 Closing Thoughts

While we see promise in the LDS-based tracking approach described in Appendix A and believe it is ultimately capable of achieving much lower tracking error rates than the particle filters we describe in Chapter 5, it is still unclear whether the LDS approach will yield results that are acceptable for later use in our larger football understanding system. Indeed, concerns still remain with this approach. For example, as it is currently formulated, the LDS approach can only fix errors that involve a player being altogether lost by our trackers, thus leaving behind a chain of unaccounted-for player detections for our error recognition procedure to find. However, our particle-filter based trackers sometimes make errors such as swapping players, where two or more trackers may each end up tracking the wrong player without leaving behind any detection-based evidence that an error has been made. It is unclear whether our LDS method

can be extended to handle these kinds of tracking errors, and more work will need to go into investigating this issue.

In addition, it is unclear whether our LDS-based approach will be able to handle the most complicated player interactions in football video. For instance, in every football play, there are extremely complex interactions between the offensive and defensive linemen. These interactions often involve ten or more players. It may not be possible for our LDS-based approach to do more than simply identify that *one* of these players accounts for a given error without actually identifying *which* of the players accounts for the error, and lower-level processing, perhaps of the type described in the literature on tracking in unstructured, crowded scenes [65, 52, 3, 2, 41], might be required for this task.

Once the tracking task is completed, there is still much work to do on the larger problem of football video analysis. The next logical step in our own system is recognizing the actions taken by individual players (e.g. individual passing patterns, blocking assignments, or run specifications). Indeed, we have already looked briefly at the problem of recognizing passing patterns from player trajectories [70], but much more work is needed here. After that, work will be needed on recognizing entire offensive plays and defensive schemes based on observed individual actions. Particular attention will need to be paid here to being able to recognize and adapt to previously unseen plays and schemes. Finally, work must be done on taking the results of all of the previous analysis and mining it for tendencies, so teams can predict what their opponents will do in various game situations. Some day, perhaps, our system will even have the ability to identify teams' weaknesses and strengths and to suggest strategies to coaches to help them exploit this information.

Of course, much of this work is very far down the road from here. Regardless, the problem of football video analysis should continue to serve as a fertile proving ground for new, broadly-applicable computer vision research, as the game's chaos and structure continue to combine to provide interesting vision challenges.

Bibliography

[1] Pieter Abbeel, Adam Coates, Mike Montemerlo, Andrew Y. Ng, and Sebastian Thrun. Discriminative training of Kalman filters. In *Proc. Robitics: Science and Systems*, 2005.

[2] Saad Ali and Mubarak Shah. A Lagrangian particle dynamics approach for crowd flow segmentation and stability analysis. In *CVPR*, 2007.

[3] Saad Ali and Mubarak Shah. Floor fields for tracking in high density crowd scenes. In *ECCV*, 2008.

[4] Jérôme Berclaz, François Fleuret, Engin Türetken, and Pascal Fua. Multiple object tracking using k-shortest paths optimization. *TPAMI*, 33(9), 2011.

[5] Matthew Brown and David Lowe. Recognizing panoramas. In *ICCV*, 2003.

[6] Matthew Brown and David G. Lowe. Recognising panoramas. In *Proc. Intl. Conference on Computer Vision*, pages 1218–1225, 2003.

[7] Yizheng Cai, Nando de Freitas, and James Little. Robust visual tracking for multiple targets. In *ECCV*, 2006.

[8] Václav Chvátal. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 4(3), 1979.

[9] Michael Collins. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *EMNLP*, 2002.

[10] Michael Collins and Brian Roark. Incremental parsing with the perceptron algorithm. In *Proc. Annual Meeting for the Assoc. for Computational Linguistics*, 2004.

[11] Brooke Cowan, Ivona Kučerová, and Michael Collins. A discriminative model for tree-to-tree translation. In *EMNLP*, 2006.

[12] David Crandall, Pedro Felzenszwalb, and Daniel Huttenlocher. *Object Recognition by Combining Appearance and Geometry*, volume 4170/2006 of *LNCS*, pages 462–482. Springer, 2006.

[13] Hal Daume III and Daniel Marcu. Learning as search optimization: Approximate large margin methods for structured prediction. In *ICML*, 2005.

[14] Janardhan Rao Doppa, Alan Fern, and Prasad Tadepalli. Output space search for structured prediction. In *ICML*, 2012 (to appear).

[15] Arnaud Doucet, Nando de Freitas, and Neil Gordon, editors. *Sequential Monte Carlo Methods in Practice*. Springer, 2001.

[16] Pedro Felzenszwalb and Daniel Huttenlocher. Distance transforms of sampled functions. Technical Report TR2004-1963, Cornell Computing and Information Science, 2004.

[17] Pedro Felzenszwalb and Daniel Huttenlocher. Pictorial structures for object recognition. *International Journal of Computer Vision*, 61(1), 2005.

[18] R. Fergus, P. Perona, and A. Zisserman. A sparse object category for efficient learning and exhaustive recognition. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2005.

[19] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[20] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman, 1979.

[21] Iryna Gordon and David G. Lowe. Scene modelling, recognition and tracking with invariant image features. In *Proc. Intl. Symposium on Mixed and Augmented Reality*, pages 110–119, 2004.

[22] Raffay Hamid, Ram Krishan Kumar, Matthias Grundmann, Kihwan Kim, Irfan Essa, and Jessica Hodgins. Player localization using multiple static cameras for sports visualization. In *CVPR*, 2010.

[23] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2nd edition, 2004.

[24] Rob Hess. An open source SIFT library. In *ACM Multimedia*, 2010.

[25] Rob Hess and Alan Fern. Discriminatively trained particle filters for complex multi-object tracking. In *CVPR*, 2009.

[26] Robin Hess and Alan Fern. Improved video registration using non-distinctive local image features. In *CVPR*, 2007.

[27] Robin Hess and Alan Fern. Toward learning mixture-of-parts pictorial structures. In *The ICML 2007 Workshop on Constrained Optimization and Structured Output Spaces*, 2007.

[28] Robin Hess, Alan Fern, and Eric Mortensen. Mixture-of-parts pictorial structures for objects with variable part sets. In *ICCV*, 2007.

[29] Chang Huang, Bo Wu, and Ramakant Nevatia. Robust object tracking by heirarchical association of detection responses. In *ECCV*, 2008.

[30] Stephen Intille. Tracking using a local closed-world assumption: Tracking in the football domain. Technical Report 296, MIT Media Laboratory Perceptual Computing Section, 1994.

[31] Stephen Intille. *Visual Recognition of Multi-Agent Action*. PhD thesis, Massachusetts Institute of Technology, 1999.

[32] Stephen Intille and Aaron Bobick. Closed-world tracking. In *ICCV*, 1995.

[33] Stephen Intille, James Davis, and Aaron Bobick. Real-time closed-world tracking. In *CVPR*, 1997.

[34] Michal Irani and P. Anandan. Video indexing based on mosaic representations. *Proceedings of the IEEE*, 86(5), 1998.

[35] Michal Irani and Shmuel Peleg. Improving image resolution by image registration. *Graphical Models and Image Processing*, 53(3), 1991.

[36] Miuchal Irani, Steve Hsu, and P. Andan. Video compression using mosaic representations. *Signal Processing: Image Communication*, 7(4–6), 1995.

[37] Hao Jiang, Sidney Fels, and James Little. A linear programming approach for multiple object tracking. In *CVPR*, 2007.

[38] Zia Khan, Tucker Balch, and Frank Dellaert. MCMC-based particle filtering for tracking a variable number of interacting targets. *PAMI*, 27(11):1805–1819, 2005.

[39] Kihwan Kim, Matthias Grundmann, Ariel Shamir, Jessica Hodgins, and Irfan Essa. Player localization using multiple static cameras for sports visualization. In *CVPR*, 2010.

[40] Jana Košecká and Xiaolong Yang. Location recognition and global localization based on scale invariand keypoints. In *Proc. Workshop on Statistical Learning in Computer Vision, ECCV*, 2004.

[41] Louis Kratz and Ko Nishino. Tracking with local spatio-temporal motion patterns in extremely crowded scenes. In *CVPR*, 2010.

[42] Cheng-Hao Kuo, Chang Huang, and Ramakant Nevatia. Multi-target tracking by on-line learned discriminative appearance models. In *CVPR*, 2010.

[43] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, 2001.

[44] Xiangyang Lan and Daniel Huttenlocher. Beyond trees: common-factor models for 2D human pose recovery. In *Proc. IEEE Int'l Conf. on Computer Vision*, 2005.

[45] Mihai Lazarescu and Svetha Venkatech. Using camera motion to identify types of American football plays. In *ICME*, 2005.

[46] Ruonan Li, Rama Chellappa, and Shaohua Zhou. Learning multi-model densities on discriminative temporal interaction manifold for group activity recognition. In *CVPR*, 2009.

[47] Benson Limketkai, Dieter Fox, and Lin Liao. CRF-filters: Discriminative particle filters for sequential state estimation. In *ICRA*, 2007.

[48] Tie-Yan Liu, Wei-Ying Ma, and Hong-Jiang Zhang. Effective feature extraction for play detection in American football video. In *MMM*, 2005.

[49] David G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2), 2004.

[50] Wei-Lwun Lu and James Little. Simultaneous tracking and action recognition using the pca-hog descriptor. In *CRV*, 2006.

[51] Wei-Lwun Lu, Kenji Okuma, and James Little. Tracking and recognizing actions of multiple hockey players using the boosted particle filter. *Image and Vision Computing*, 27(1–2), 2009.

[52] Ramin Mehran, Alexis Oyama, and Mubarak Shah. Abnormal crowd behavior detection using social force model. In *CVPR*, 2009.

[53] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. van Gool. A comparison of affine region detectors. *Intlernational Journal of Computer Vision*, 65(1-2):43–72, 2005.

[54] Krystian Mikolajczyk and Cordelia Schmid. Scale & affine invariant interest point detectors. *Intl. Journal of Computer Vision*, 60(1):63–86, 2004.

[55] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1615–1630, 2005.

[56] Andrew Y. Ng and Michael Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *NIPS*, 2002.

[57] Peter nillius, Josephine Sullivan, and Stefan Carlsson. Multi-target tracking – linking identities using Bayesian network inference. In *CVPR*, 2006.

[58] Songhwai Oh, Stuart Russell, and Shankar Sastry. Markov chain Monte Carlo data association for general multiple-target tracking problems. In *CDC*, 2004.

[59] Songhwai Oh and Shankar Sastry. An efficient algorithm for tracking multiple maneuvering targets. In *CDC*, 2005.

[60] Songhwai Oh and Shankar Sastry. A polynomial time approximation algorithm for joint probabilistic aata association. In *ACC*, 2005.

[61] Kenji Okuma, James Little, and David Lowe. Automatic rectification of long image sequences. In *Proc. Asian Conf. on Computer Vision*, 2004.

[62] Kenji Okuma, Ali Taleghani, Nando de Freitas, James J. Little, and David G. Lowe. A boosted particle filter: multitarget detection and tracking. In *ECCV*, 2004.

[63] Patrick Pérez, Carine Hue, Jaco Vermaak, and Michel Ganget. Color-based probabilistic tracking. In *ECCV*, 2002.

[64] Hamed Pirsiavash, Deva Ramanan, and Charless Fowlkes. Globally-optimal greedy algorithms for tracking a variable number of objects. In *CVPR*, 2011.

[65] Mikel Rodriguez, Saad Ali, and Takeo Kanade. Tracking in unstructured crowded scenes. In *ICCV*, 2009.

[66] Stephen Se, David G. Lowe, and Jim Little. Global localization using distinctive image features. In *Proc. Intl. Conference on Intelligent Robots and Systems*, pages 226–231, 2002.

[67] Behjat Siddiquie, Yaser Yacoob, and Larry Davis. Recognizing plays in American football videos. Technical report, University of Maryland, 2009.

[68] G. Simon, A. Fitzgibbon, and A. Zisserman. Markerless tracking using planar structures in the scene. In *Proc. Intl. Symposium on Augmented Reality*, pages 120–128, 2000.

[69] Drew Steedly, Chris Pal, and Richard Szeliski. Efficiently registering video into panoramic mosaics. In *ICCV*, 2005.

[70] David Stracuzzi, Alan Fern, Kamal Ali, Robin Hess, Jervis Pinto, Nan Li, Tolga Könik, and Dan Shapiro. An application of transfer to american football: From observation of raw video to control in a simulated environment. *AI Magazine*, 32(2), 2011.

[71] Josephine Sullivan and Stefan Carlsson. Tracking and labelling of interacting multiple targets. In *ECCV*, 2006.

[72] Eran Swears and Anthony Hoogs. Learning and recognizing complex multi-agent activities with applications to American football plays. In *WACV*, 2012.

[73] B. Taskar, C. Guestrin, and D. Koller. Max-margin Markov networks. In *NIPS*, 2004.

[74] Carlo Tomasi and Kanade Takeo. Detection and tracking of point features. Technical Report CMU-CS-91-132, Carnegie Mellon University, 1991.

[75] Antonio Torralba, Kevin Murphy, William Freeman, and Mark Rubin. Context-based vision system for place and object recognition. In *Proc. IEEE International Conf. on Computer Vision*, 2003.

[76] Lloyd N. Trefethen and David Bau III. *Numerical Linear Algebra*. Society for Industrial and Applied Mathematics, 1997.

[77] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *ICML*, 2004.

[78] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR*, 2001.

[79] Zheng Wu, Thomas Kunz, and Margrit Betke. Efficient track linking methods for track graphs using network-flow and set-cover techniques. In *CVPR*, 2011.

[80] Yuehua Xu and Alan Fern. On learning linear ranking functions for beam search. In *ICML*, 2007.

[81] Ting Yu and Ying Wu. Collaborative tracking of multiple targets. In *CVPR*, 2004.

[82] Ting Yu and Ying Wu. Decentralized multiple target tracking using netted collaborative autonomous trackers. In *CVPR*, 2005.

[83] Luke Zettlemoyer and Michael Collins. Online learning of relaxed CCG grammars for parsing to logical form. In *EMNLP/CoNLL*, 2007.

[84] Li Zhang, Yuan Li, and Ramakant Nevatia. Global data association for multi-object tracking using network flows. In *CVPR*, 2008.

[85] Zhigang Zhu, Guangyou Xu, and Xueyin Lin. Constructing 3D natural scene from video sequences with vibrating motions. In *Virtual Reality Annual International Symposium*, 1998.

[86] Zhigang Zhu, Guangyou Xu, Edward M. Riseman, and Allen R. Hanson. Fast construction of dynamic and multi-resolution 360-degree panorama from video sequences. In *ICMCS*, 1999.

APPENDICES

Appendix A: Our Recent Efforts in Multi-Object Tracking

In this chapter, I describe some of our recent efforts to design a tracking algorithm for the American football domain that takes a longer-term view than our shortsighted particle filtering approach. In one of our first such attempts, we examined an MCMC-based data association algorithm similar to the one outlined in [58, 60, 59], in which we attempted to use an MCMC procedure to link player detections into tracklets and tracklets into complete trajectories. The MCMC procedure could propose steps such as linking a string of detections into a tracklet, linking two tracklets into a longer track/trajectory, unlinking a subset of the detections in a tracklet (up to unlinking all of the tracklet's detections), swapping portions of two tracklets, etc. We applied this method to player detections of various types, including ones from the well-known Viola-Jones-style object detector [78] and ones based on a particle advection-style approach as described in [52, 2]. However, in each case, inference over these detection sets—which, by their nature, contained many false positive detections, missing detections, and multiply detected players—was too complex, and the method achieved very poor results.

We also experimented with a Viterbi-like algorithm that attempted to link player detections into longer trajectories. In particular, we expanded Viterbi to perform a beam search, where, instead of finding a player's best single trajectory through the detection set, as the standard Viterbi algorithm would, our extended algorithm found the best b trajectories through the detection set. Our hope was that, by allowing the algorithm to maintain multiple trajectory hypotheses for each player, it would be able to continue to keep track of a player's true trajectory through a hijacking situation (i.e. while also maintaining the hijacked track). We also included a trajectory clustering step to avoid situations where a player's beam included two nearly-identical trajectories that differed by only a single detection. Unfortunately, allowing the Viterbi algorithm to maintain a beam for each player instead of a single trajectory made it very difficult to include player interaction features like the ones we used in our particle filter-based method, in which each tracker maintained only a single, known state. Thus, because it could not explicitly reason about player interaction, this Viterbi-based approach ended up being just as susceptible to hijacking as our particle filters (if not more so) and never yielded satisfactory results.

A.0.1 Limited-Discrepancy Search for Multi-Object Tracking

The most promising long-view tracking approach we have explored is one based on limited-discrepancy search (LDS) [14]. This work is preliminary, but it is worth discussing here at some length.

LDS is a search-based framework for structured prediction, in which a combinatorial search space is defined by the output of an imperfect but mostly-accurate recurrent structured classifier, i.e. a classifier that constructs structured outputs by making a series of decisions, each of which can depend on the input and the previously made decisions. The approach taken by LDS is to try to correct the errors made by the classifier, one at a time, and then to re-run the classifier with the fixed errors held constant. The idea is that by correcting individual errors, the performance of the classifier can be greatly improved, and, thus, only a very shallow search should be needed to find the correct structured output corresponding to a given input. Indeed, the LDS approach described in [14] learns a cost function over candidate error fixes (or *discrepancies*) to try to guide the search quickly to the correct output.

To illustrate more concretely how LDS works, consider an example from the visual word recognition problem. Let us assume we have a trained recurrent classifier that, given an image of a handwritten word, predicts the word depicted in the image, and assume that we pass as input to that classifier an image containing the word "search." It is quite possible that if that word is sloppily written, our classifier might mistake the 'e' for a 'c'. Once this decision is made, this error might propagate to further decisions causing our classifier to output "scared" instead of "search." However, if we correct the intial error of mistaking 'e' for 'c' and hold this constant while we re-run the classifier, it is likely that the classifier will be able to correct the rest of the word without further search.

For the tracking problem, we use the particle filters described in Chapter 5 as our recurrent classifier. The basic idea here, assuming we are operating in the American football domain, is as follows:

1. Run our particle filter trackers to obtain an initial prediction about the trajectories of the players.

2. Identify potential space-time locations where the trackers might have made an error (more on identifying potential errors below).

3. Identify which players could possibly fix the errors found in step 2. The player-error pairs

thus formed are our candidate discrepancies.

4. For each discrepancy formed in step 3, re-run the trackers with the constraint that the tracker whose player is associated with the discrepancy being examined is forced to fix the corresponding error. To make this step more efficient, we may duplicate the input tracking results until a player is involved in a discrepancy. At that point we activate that player's tracker and may continue to duplicate the input results for the other players until they are either involved in a discrepancy or interact with a player whose tracker is active.

5. Rank all of the tracking results from step 4 (more on ranking below). If running a greedy search, repeat at step 2 with the highest-ranked tracking results and the corresponding discrepancy. If running a beam search, repeat at step 2 with the b highest-ranked tracking results and the corresponding discrepancies. Repeat for a maximum number of iterations or until no more errors are available to be fixed.

In practice, re-running the trackers and ranking tracking results for all discrepancies can be time consuming, so we insert a filtering step prior to step 3 in which we heuristically approximate the tracking results for each discrepancy by assuming only the tracking results for the player involved in the discrepancy will differ from the input results. Then, we simply modify this player's tracking results to force the player to fix the error asociated with the discrepancy. We rank these heuristically approximated results for each discrepancy and pass only the k highest-ranked ones on to step 4.

We use player detections that are not accounted for by the tracking results to identify potential errors in step 2. Specifically, we identify temporal strings of unaccounted-for detections in which the spatial overlap between detections in contiguous video frames is above certain threshold, and we take all such strings of at least a given duration to be potential errors. Errors selected this way are reliable, with a few exceptions where a long string of detections is generated around a field logo. However, as we discuss below, the discrepancies corresponding to these erroneous detection strings can be filtered out using the ranking function by including ranking features that measure stationarity and proximity to field logos.

To determine what players might possibly explain a given error, we examine a short temporal window directly before the beginning of the error. In each frame within that temporal window, we look at a spatial window centered around the location of the beginning of the error and whose radius is determined based on the player dynamics equation (5.9). In particular, as the time until

the beginning of the error increases, the radius of the spatial window also increases so that the beginning location of the error can always be reached from the boundary of the spatial window by dynamics that are feasible under (5.9). Any player that falls within this combined spatio-temporal window is considered as potentially explaining the error in question, and a candidate discrepancy is generated for this player-error pair.

A.0.1.1 Ranking Function

We use a linear ranking function of the form $r(\mathbf{x}, \mathbf{y}) = \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, \mathbf{y})$, where $\mathbf{x}$ and $\mathbf{y}$ are the output tracks and input observation data, respectively, $\mathbf{f}(\cdot)$ is a feature vector over the tracks and observations, and $\mathbf{w}$ is the feature weight vector. The features we use in this function include the following:

- **Unexplained detections.** The proportion of player detections left unexplained by the tracks. This is used to encode a preference for tracks that explain more detections.
- **Double-counted detections.** The proportion of player detections that are accounted for by more than one track. This is used to encode a preference for tracks that double-count fewer detections.
- **Spatio-temporal histogram comparisons.** A histogram for each of the following spatio-temporal properties is computed for each player and compared to a distribution of reference histograms for players of the same type. These are used to try to encourage tracked player trajectories to be compatible with the distribution of trajectories players of the same type are known to take.
  - *Field region.* The field is divided into eight regions relative to the starting location of the ball, and player location at each time step is discretized into one of these regions.
  - *Motion direction.* Player motion at each time step is discretized into one of the eight compass directions.
  - *Motion magnitude.* The magnitude (in pixels) of player motion at each time step is discretized into bins.
  - *Distance from scrimmage.* Player distance from the line of scrimmage at each time step is discretized into bins.
- **Error-characterization features.** The errors in the selected discrepancies are quantified in several ways.

- *Time from player to error.* The number of frames between the beginning of the error in question and the location of the player when he is first constrained by the discrepancy. This is included to encourage the selection of discrepancies in which the player is closer to the error.
- *Error length.* The duration of the error in question, in frames, as a proportion of the total length of the video. This is included to encourage the selection of discrepancies representing longer errors.
- *Error frames near logo.* The proportion of the error in question's duration during which it is near a field logo. This is included to discourage the selection of discrepancies whose errors remain mostly within field logos. Such errors are likely to be false positives.

We learn weights for our ranking function using a procedure similar to the one described in [14]. Specifically, we run LDS search as described above with the untrained ranking function, and we record the tracking results and the set of candidate discrepancies at each search iteration. Then, when the search is complete, we order each set of candidate discrepancies by their true squared error, and we perform a pairwise perceptron update on the feature weights $\mathbf{w}$ between the best candidate discrepancy, according to squared error, and each other candidate that is ranked above it. In addition, for searches of more than one iteration, we keep track of the best candidate seen so far at each search iteration and perform weight updates if that candidate is ranked lower than any of the candidates at the current iteration.

A.0.1.2 Preliminary Experiments

We have run a number of preliminary experiments to test our LDS-based tracking approach on our American football data set, and I briefly describe them here. Though we have not achieved any significant improvement over our original particle filter-based trackers in these experiments, we have seen promising signs that the LDS-based method is worth continuing to pursue.

Our experiments with the LDS-based method are set up similarly to the ones we describe in Chapter 5. In particular, we use the same data set of 20 videos of American football plays, and we divide this data set into four folds of five videos each.

To date, we have only run experiments testing a single iteration of search, i.e. we have only used LDS to look for a fix to a single error in the original tracking results. For each of the folds in

the data set, we trained our ranking function on the single-iteration search results for the videos in the other three folds and then looked at the performance of the learned ranking function on the first-iteration search candidates in the testing fold.

The learned ranking function has given mixed results, and we are not convinced that any overall improvements in tracking error it has yielded are not just the result of random noise in the approach. In particular, while in multiple runs of these experiments we have seen the learned ranking function correctly select several true error-fixing discrepancies, we have also seen it occasionally select error-increasing discrepancies as well. Indeed, it is often the case that, for a given video, no error-fixing discrepancies survive our preliminary heuristic filtering step, and thus, in these cases, no error-fixing discrepancies are available for the ranking function to select. However, even when we hard code a guarantee that the best error-fixing discrepancy (i.e. the one that yields the lowest true tracking error) must survive the preliminary filtering step, the learned ranking function does not always select this discrepancy as the best one. We have also experimented with training the preliminary filtering function, but these experiments yielded similar results.

Nonetheless, despite their mixed results, we do see promising signs in these experiments. For example, while the preliminary filtering step often filters out any good discrepancies, we have noted that good, error-fixing discrepancies are always present before the preliminary filtering step. Often, discrepancies exist before the preliminary filtering step that reduce the overall tracking error for a single video by more than 50%. The consistent presence of these error-fixing discrepancies validates the error identification and player-error pairing steps of our procedure.

To explore the effectiveness of our ranking function learning procedure, we ran a set of experiments in which we included the true tracking error as a ranking feature. As expected, our training procedure learned a very large negative weight for the true error feature, signifying that the learning procedure was able to identify the fact that error is correlated with itself, and the overall learned ranking function was able to almost perfectly order the candidate discrepancies by increasing tracking error. The fact that the trained ranking function could successfully achieve the correct ordering when obviously discriminating information was present in the ranking features demonstrates the validity of our training procedure. Indeed, these results suggest that this overall approach may simply need better ranking features in order to be successful. This is likely also the case for the preliminary filtering function, since it uses the same features as the ranking function. Unfortunately, we have not yet had a chance to test any additional ranking features, but attempting to improve upon the information provided to the LDS-procedure in the form of

ranking features appears to be a logical next step in exploring this approach.