Description, Analysis and Prediction of Player Actions in Selected Hockey Game Situations

by

Fahong Li

B.E., Xi'an Jiaotong University 1996

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
Master of Science
in
THE FACULTY OF GRADUATE STUDIES
(Department of Computer Science)

We accept this thesis as conforming
to the required standard

The University of British Columbia
April 2004
© Fahong Li, 2004
Library Authorization

In presenting this thesis in partial fulfillment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the head of my department or by his or her representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

FAHONG LI
Name of Author (please print)

21/04/2004
Date (dd/mm/yyyy)

Title of Thesis: Description, Analysis and Prediction of Player Actions in Selected Hockey Game Situations

Degree: Master of Science Year: 2004

Department of Computer Science

The University of British Columbia
Vancouver, BC Canada
Abstract

Motion is an important cue to the intentions of active agents in environments involving collaboration and competition. We demonstrate this in the domain of ice hockey. We develop a framework to represent and reason about hockey behaviors using as input actual player motion trajectory data tracked from game video and supported by knowledge of hockey strategy, game context and specific player profiles. The raw player motion trajectory data consists of space-time point sequences of forward/backward skating registered to rink coordinates. This is augmented with knowledge of possession of the puck and specific player attributes (e.g., shoots left, shoots right).

We focus on the analysis of three clearly identifiable situations: 2-on-1 offensive attacks, defensive zone breakouts and power play shots from the point. We use a Finite State Machine (FSM) model to represent our total knowledge of a given situation and develop evaluation functions for primitive hockey behaviors (e.g., pass, shot). Based on the augmented trajectory data, the FSMs and the evaluation functions, we describe what happened in each identified situation, assess the outcome, estimate when and where key play choices were made, and attempt to predict whether better alternatives were available to achieve understood goals. A textual natural language description and a simple 2D graphic animation of the analysis are produced as the output. The graphic animation is useful for interactive visualization and debugging. The textual description also provides potentially useful annotation for large databases of player motion trajectories.

The framework is flexible to allow the substitution of different analysis modules and extensible to allow the inclusion of additional hockey situations. We expect that the methodology and the framework can be generalized and applied in other domains, such as soccer, basketball, traffic flow control and people surveillance.
Contents

Abstract ii

Contents iii

List of Tables vi

List of Figures vii

Acknowledgements ix

Dedication x

1 Introduction 1

1.1 The problem and our motivation 2

1.2 The IRIS TRA project 6

1.3 The framework and a hockey scenario analysis 7

1.4 Contributions 15

1.5 Outline of the following chapters 16

2 Relevant work 17

2.1 Sports video analysis 17

2.1.1 Trajectory acquisition 18
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1.2</td>
<td>Event recognition</td>
<td>20</td>
</tr>
<tr>
<td>2.1.3</td>
<td>Play analysis</td>
<td>22</td>
</tr>
<tr>
<td>2.2</td>
<td>Knowledge representation</td>
<td>27</td>
</tr>
<tr>
<td>2.3</td>
<td>Qualitative reasoning and collision detection</td>
<td>29</td>
</tr>
<tr>
<td>2.4</td>
<td>Summary of relevant work</td>
<td>30</td>
</tr>
<tr>
<td>3</td>
<td>Design and assumptions</td>
<td>32</td>
</tr>
<tr>
<td>3.1</td>
<td>Architectural overview</td>
<td>34</td>
</tr>
<tr>
<td>3.2</td>
<td>Augmented trajectory data</td>
<td>36</td>
</tr>
<tr>
<td>3.3</td>
<td>Play events and evaluation functions</td>
<td>37</td>
</tr>
<tr>
<td>3.4</td>
<td>Game situations, play knowledge and analysis strategies</td>
<td>42</td>
</tr>
<tr>
<td>3.4.1</td>
<td>Definitions of specific game situations</td>
<td>42</td>
</tr>
<tr>
<td>3.4.2</td>
<td>Modeling play knowledge with FSMs</td>
<td>45</td>
</tr>
<tr>
<td>3.4.3</td>
<td>Situation analysis modules</td>
<td>48</td>
</tr>
<tr>
<td>3.5</td>
<td>Main modules</td>
<td>49</td>
</tr>
<tr>
<td>4</td>
<td>Implementation</td>
<td>51</td>
</tr>
<tr>
<td>4.1</td>
<td>Input data</td>
<td>52</td>
</tr>
<tr>
<td>4.1.1</td>
<td>Data structures and major classes for the input data</td>
<td>52</td>
</tr>
<tr>
<td>4.1.2</td>
<td>Acquisition of the ATD and SPEs from video clips</td>
<td>63</td>
</tr>
<tr>
<td>4.1.3</td>
<td>Textual FSMs for the selected game situations</td>
<td>64</td>
</tr>
<tr>
<td>4.2</td>
<td>Event evaluation functions</td>
<td>70</td>
</tr>
<tr>
<td>4.3</td>
<td>Configuration</td>
<td>74</td>
</tr>
<tr>
<td>4.4</td>
<td>Description, Analysis and Prediction</td>
<td>74</td>
</tr>
<tr>
<td>4.4.1</td>
<td>Description function</td>
<td>77</td>
</tr>
<tr>
<td>4.4.2</td>
<td>Analysis function</td>
<td>78</td>
</tr>
</tbody>
</table>
List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Main storage items</td>
<td>35</td>
</tr>
<tr>
<td>3.2</td>
<td>Main modules</td>
<td>35</td>
</tr>
<tr>
<td>3.3</td>
<td>A team-centric hierarchy of hockey game behaviors</td>
<td>38</td>
</tr>
<tr>
<td>3.4</td>
<td>Primitive play events and the involved roles</td>
<td>40</td>
</tr>
<tr>
<td>3.5</td>
<td>Definition of 2-on-1 offensive attack</td>
<td>43</td>
</tr>
<tr>
<td>3.6</td>
<td>Definition of power play shot from the point</td>
<td>44</td>
</tr>
<tr>
<td>3.7</td>
<td>Definition of defensive zone breakout</td>
<td>45</td>
</tr>
</tbody>
</table>
List of Figures

1.1 Key image frames in clip1 ............................................. 9
1.2 The FSM for simple 2-on-1 offensive attacks ...................... 10
1.3 Analysis of a clip of 2-on-1 offensive attack in the framework .... 11
1.4 Applying one of the predefined trajectory perturbations to the anal­ysis in Figure 1.3 .................................................... 12
2.1 Block diagram of the tennis video analysis system, from [31] ...... 21
2.2 The layered hierarchy of the COBRA video data model, from [25] 23
2.3 The architecture of the SOCCER system, from [1] .................. 24
2.4 Part of the plan hierarchy in REPLAI, from [26] .................... 26
2.5 Plan schema (decomposition) for “solo”, from [26] .................. 26
2.6 The architecture of the REPLAI component, from [26] .......... 27
2.7 The group/region hierarchy, from [15] ............................. 30
3.1 Architectural overview ................................................. 33
3.2 A player-centric hierarchy of hockey game behaviors .............. 38
3.3 FSM model for 2-on-1 offensive attack .............................. 46
3.4 FSM model for the power play shot from the point ................. 47
3.5 FSM model for the defensive zone breakout ........................ 47
4.1 Data structures for the Augmented Trajectory Data ............... 53
4.2 Data structures for the Sorted Play Events .................. 53
4.3 Wrapper classes for TClip, TGame and TScoring .............. 56
4.4 Wrapper classes for TTrajectory, TPrimEvent, TShotOnGoal and TPlayer ............................................. 57
4.5 Wrapper classes for TPuckPoss, TSkateMode and TCrossBL ... 58
4.6 Data structures for the Finite State Machines ................. 59
4.7 Wrapper class for THFSM, TEvent, TState and TTransition ... 60
4.8 The FSM for simple 2-on-1 offensive attacks .................. 66
4.9 The FSM for 5-on-4 power play shots from the point ........... 67
4.10 The FSM for simple defensive zone breakouts .................. 68
4.11 Illustration for EvShootV2 ..................................... 73
4.12 The Configure dialog ........................................... 75
4.13 The structure TClipSituDAC and its wrapper class ............ 76
4.14 Visualizing information stored in pClip ........................ 81
5.1 Key image frames in clip2 ....................................... 84
5.2 Analysis of clip2, an instance of 2-on-1 offensive attack ....... 86
5.3 Applying one of the predefined trajectory perturbations to the analysis in Figure 5.2 ......................... 87
5.4 Key image frames in clip3 ....................................... 89
5.5 Analysis of clip3, an instance of power play shot from the point ... 90
5.6 Key image frames in clip11 ..................................... 92
5.7 Analysis of clip11, an instance of defensive zone breakout .... 93
Acknowledgements

I would like to thank my advisor, Dr. Bob Woodham, for his guidance, patience, and encouragement to me. Without him, this work could not be possible. During the weekly meetings, I was often inspired by his deep vision, keen insight and excellent analogies. His humors entertained us a lot too. It was a really good time for me to watch the ice hockey games at UBC with Bob and Nancy. Special thanks for his detailed comments on the first draft, which even include correcting my spelling and grammar errors.

I also thank Dr. David Lowe for being the second reader of this thesis. Thanks for his comments on the presentation style and other valuable suggestions and kind encouragement to me.

Thanks to other members in the IRIS TRA project, including Dr. Raymond Ng, Dr. Jim Little, Michael Zhang, Kenji Okuma, Xiaojing Wu and Yuhan Cai for their guidances and/or cooperations.

Thanks to all LCI members, who make my academic life in UBC much more interesting; and last but not least, thank-you to all my friends in Vancouver, who make me enjoy staying in this beautiful city so much.

Fahong Li

The University of British Columbia
April 2004
To my dear parents, for their common but great love to me.
Chapter 1

Introduction

What are important research problems in computer vision? To what degree will society benefit from solutions to these problems? Researchers give different answers according to their own knowledge, experience and motivation. Researchers delve into the problems they think important. Some will succeed and prove their original ideas important and correct. Others will fail and change direction. Both success and failure provide valuable lessons to those that follow.

With this thought in mind, we choose the domain of ice hockey to demonstrate that motion is an important cue to the intentions of multiple active agents in a context involving both collaboration and competition. Thus players' motion trajectory data, augmented with their personal profiles and knowledge of possession of the puck, could help coaches and players analyze games and improve play in specific game situations. The first section in this chapter presents the problem in detail and addresses our motivation. In the second section, we introduce the IRIS TRA project, from which the problem is derived, and which provides the focus for the thesis. The third section uses a typical hockey scenario to briefly illustrate the framework designed to solve the problem. We summarize our contributions in the
fourth section and outline the following chapters in the last section.

1.1 The problem and our motivation

Coaching is crucial in competitive sports, such as ice hockey. Many books on hockey coaching use diagrams and textual descriptions to represent and explain play formations and strategies (e.g., [24] and [29]). Thanks to electronic technology, instructional hockey videos (as listed in [35] and [38]) came into being and present vivid drill or game clips to the audience. Recently, some computer software (e.g., [36]) has been developed to show play plans with graphic animation.

All these media (books, videos, software) help coaches and players improve play to some degree. However, to our knowledge, there is no framework which automatically or semi-automatically integrates play knowledge accumulated since the origin of ice hockey, the vast and ever increasing amount of hockey video data acquired in recent decades, and computer graphic animation to demonstrate how to play hockey effectively in various game situations, such as 2-on-1 offensive attacks. One can imagine, in such a framework, that users (coaches, players or fans) can easily browse the diagrams and textual descriptions on how to play effectively in various game situations, can watch the actual play in video clips classified as instances of a particular situation, and can simulate/alter the play with graphic animation for more detailed analysis based on trajectory data automatically extracted from the video.

It is possible to build such a framework, since the key task (reasoning about play effectiveness according to augmented players' trajectory data) is supported by a fundamental scientific observation: motion is an important cue to the intentions of active agents. In a context having both collaboration and competition, one can
interpret the motion of those agents in order to:

- describe their actions, i.e., their choices and the results of their choices,
- analyze the effectiveness of their actions towards achieving understood goals, and
- predict alternative actions which are likely to be more successful in achieving understood goals.

For example, in a 2-on-1 offensive attack, we can clearly define the start, the end and the outcome (either good, bad or neutral) of the situation. After detecting the start, we can check the two attackers' trajectory data, which is augmented with information on possession of the puck and players' profiles (e.g., shoots left, shoots right), to see whether they skate wider to set up a pass to beat the defenceman or whether the puck carrier just skates directly for a shot towards the goal. If they skate wider for a pass, we can calculate the feasibility of passing the puck, based on the geometric configuration of these three players' positions and their profiles. If the puck carrier shoots towards the goal, we can evaluate the effectiveness of shooting at that particular point, considering the distance between the point and the goal, the shooting angle, the position of the goalie, and the positions of the other two players. We can also explore possible better outcomes by assuming the puck carrier makes the choice (pass or shot) at a slightly different timing (sooner or later) or at a different place (e.g., nearer to the goal).

In order to build such a framework, a series of primary questions need to be answered first. The questions include (but are not limited to) the following:

1. How to classify or define hockey game situations (including their start, end and outcomes), such as a 2-on-1 offensive attack, according to hockey domain
knowledge? And how to segment video data into individual instances of different game situations?

2. How to represent play knowledge so that it can be used to describe, evaluate and reason about any particular play element which is defined in the framework?

3. How to describe a particular instance of a game situation?

4. How to evaluate and reason about a particular play element (i.e., prove that playing in one way is likely to be better than another way)?

5. What information on the actual play is necessary to do the analysis required by users? And how to get it (e.g., extract it from video clips or record it with augmented instruments)?

6. How to extend the framework's knowledge about hockey play either by direct input or by learning?

Obviously some of these questions are open ended. They are too broad and complicated to answer in a master's thesis project. As a first step, we focus on building what we see as the core parts of the framework, including representation of general hockey play knowledge and augmented players' trajectory data, description and evaluation of particular play elements, and exploration of better alternatives to the actual play observed. We limit ourselves to three well-defined game situations: a 2-on-1 offensive attack, a power play shot from the point and a defensive zone breakout. We make the following assumptions and simplifications:

- The video data have been segmented and classified into individual clips belonging to one of the three identified game situations. We can do this manually.
Papers on video analysis addressing similar problems automatically in other sports have been published (see Chapter 2 for relevant work).

- For each video clip, players' (and the puck's) trajectory data have been accurately acquired and registered to a rink model. We use tracking software developed by others in the IRIS TRA project [23]. It is also feasible to get trajectory data by instrumenting players, the puck and the rink itself, as done in FoxTrax[37].

- Trajectory data have been (manually or semi-automatically) augmented with players' profiles and knowledge of players' skating forward/backward, falling down, and possession of the puck.

- High-level play events (such as pass, shot) have been recognized and extracted from the video clips. We do this manually. See Chapter 2 for relevant work to recognize higher-level play events automatically in video of other sports.

- The knowledge on how to play in the three identified game situations is provided as input to the framework.

We provide a flexible architecture which allows the input of additional analysis modules. Thus we can analyze an actual play from different perspectives and at different levels of detail. In addition, the architecture is extensible in that other hockey situations can be analyzed once knowledge about play in those situations and associated augmented trajectory data are provided.

In summary, we develop the core of a flexible and extensible system to reason about hockey play effectiveness which takes as input players' trajectory data, profiles and knowledge of their skating forward/backward, falling down and possession
of the puck. The core includes a Finite State Machine representation for general play knowledge, data structures for augmented trajectories, and a mechanism for generating play descriptions, for analyzing the actual play and for predicting alternative actions in the play. The design is generalizable not only to other hockey situations but ideally also to other domains, such as soccer, basketball, traffic flow control, and people surveillance.

1.2 The IRIS TRA project

The problem presented above arises as part of the IRIS TRA project [39]. We use tools already developed in the project in our implementation, and contribute ourselves to additional objectives of the project. The IRIS TRA project has four overall technical objectives:

- Trajectory acquisition and measurement: to build object appearance models automatically by tracking features and solving for their 3D structure; to develop a scheme for acquiring accurate trajectories of multiple, similar objects in various circumstances including cross-overs; to construct a common frame of reference for images captured from multiple cameras/sensors.

- Trajectory representation: to develop a new representation language supporting the recognition of key motion patterns over the range of spatial-temporal scales and viewpoints associated with the recognition/interpretation; to develop new kinetic data structures for tracking objects whose motion paths are not known a priori.

- Trajectory querying: to develop new storage and retrieval schemes for masses of spatio-temporal data, including an expressive query language and interface,
new indexing schemes, etc.

- Trajectory analysis and prediction: to identify commonly occurring sub-trajectories and patterns from a database of typical motion trajectories; to predict in a short time interval what can happen in the immediate future; to identify observable human motion pre-cursors.

We choose the domain of ice hockey to explore some of the ideas proposed in the project. Ice hockey is a very high-speed team sport, full of players’ cross-overs, involving both collaboration and competition. In Canada, hockey is very popular and we have access to masses of hockey game videos. Two of the project’s principal investigators have expertise in playing and coaching hockey.

As part of the project, we focus on trajectory representation, analysis and prediction. We use an existing tracking module to acquire players’ trajectory data. We expect that the output of our analysis and prediction modules can support the querying and indexing.

1.3 The framework and a hockey scenario analysis

We assume we can extract accurate players’ trajectory data, players’ identity, players’ skating mode (i.e., forward, backward or falling down), and information on possession of the puck from the video clips. Given player identity, we assume we have access to ancillary player profile data which include individual attributes such as shoots left, shoots right, maximum speed, etc. We name these data as Augmented Trajectory Data (ATD). We also assume there is some event recognizer which can identify key higher-level play events (such as shoot, pass, etc.) as they happen in the video clips, and we define the list of these play events as Sorted Play Events (SPEs).
We design a variety of functions (called Event Evaluation Functions, or EEFs for short) to evaluate the effectiveness (feasible or not) of players' taking primitive actions (play events) under particular geometric configurations during some time interval. Play knowledge in specific game situations is represented with Finite State Machines (FSMs). Situation Analysis Modules (SAMs) are developed for each FSM to support a puck carrier centered analysis strategy.

In our framework, the meaning of the words description, analysis and prediction is given as follows. Description is a parse of an SPE according to the FSM selected by the user. The result is the transition path augmented with time information indicating when each transition in the path occurs in the video clip. Analysis identifies feasible alternative paths in the FSM which start from the same initial state as the path parsed in the description and end up in a final state whose outcome is as good as or better than the outcome in the actual path. Prediction follows the same transition path as in the description but varies the players' spatio-temporal trajectories and checks whether there are more favorable outcomes resulting from these trajectory perturbations. Thus prediction is limited to determining whether a slightly different positioning or timing of what actually happened might have led to a more favorable outcome.

We visualize the ATD (either the original one or one altered in the prediction process) in one window using a simplistic 2D graphic animation, synchronized with the video clip displayed in another window. The riskiness of passing and shooting can also be visualized in the animation upon the user's request. Meanwhile, textual descriptions of the SPEs and the paths generated in the description and analysis process also are output.

Consider one video clip of a 2-on-1 offensive attack as an example. In this clip
the 2-on-1 situation starts from Frame #72 (Figure 1.1(a)), when the attacker (Player #2) on the right begins to carry the puck forward in the neutral zone and the other attacker (Player #3) follows up on the left. The opposite defenceman (Player #1) keeps skating backward between these two attackers until Frame #152 (Figure 1.1(d)), when he falls down in front of the goal to block the potential pass from Player #2 to Player #3. It ends when the offensive team loses possession of the puck at Frame #164 (Figure 1.1(f)).

With tools built in the IRIS TRA project, we acquired the players’ trajectories, registered them to the rink coordinate frame, and augmented them with
necessary information to form the ATD for this clip. Then we manually identified
and sorted the higher-level play events (SPEs) of interest from the ATD. The FSM
(Figure 1.2) representing play knowledge in the simple 2-on-1 offensive attack is
provided as input, in a formatted text file. Description of the clip parses the SPEs

![Figure 1.2: The FSM for simple 2-on-1 offensive attacks](image)

as a sequence of transitions in the FSM and outputs a corresponding natural lan­
guage account as text. Analysis finds alternative transition paths in the FSM which
start from the same initial state as the path matched in the description but end
up with good or better outcomes. Prediction considers the actual path in the de­
scription but perturbs both the timing of key play events and the players’ geometric
configurations to check whether these perturbations might result in more favorable
outcomes.

Figure 1.3 is part of the final output. The interface has 4 windows: the left
window on the top shows the original video clip; the middle window on the top
displays textual output; the right window on the top contains control buttons; the
window at the bottom visualizes the ATD and the evaluation results.

Major textual components of the description include:
Figure 1.3: Analysis of a clip of 2-on-1 offensive attack in the framework

In the bottom animation window, areas (i.e., possible passing/shooting paths) filled with green horizontal lines represent intervals during which it is feasible to pass/shoot, areas filled with yellow diagonal crossing lines represent intervals risky to pass/shoot, and areas filled with red vertical lines represent unwise to pass/shoot.
Figure 1.4: Applying one of the predefined trajectory perturbations to the analysis in Figure 1.3
At Frame #120, Player #2 crossed the blue line.

At Frame #152, Player #1 fell down.

At Frame #164, the offensive team (Player #2) lost possession of the puck.

The clip goes through the path:

1. q0[Frame #72]: The two attackers are outside their offensive blue line.
2. q1[Frame #120]: The puck carrier is inside his offensive zone.
3. q2[Frame #164]: The puck carrier lost possession of the puck.

The outcome is bad for the offensive team.

Major textual components of the analysis include:

According to the selected FSM and the return values of evaluation functions, i.e.,

Since Frame #73, EvPassV1() returns:1.
Since Frame #114, EvPassV1() returns:0.
Since Frame #115, EvPassV1() returns:1.
Since Frame #116, EvPassV1() returns:0.
Since Frame #118, EvPassV1() returns:1.
Since Frame #119, EvPassV1() returns:0.
Since Frame #120, EvPassV1() returns:1.
Since Frame #123, EvPassV1() returns:0.
Since Frame #130, EvPassV1() returns:1.
Since Frame #130, EvShootV2() returns:3.
Since Frame #133, EvPassV1() returns:0.
Since Frame #134, EvPassV1() returns:1.
Since Frame #136, EvPassV1() returns:0.
Since Frame #137, EvPassV1() returns:1.
Since Frame #138, EvPassV1() returns:0.
Since Frame #139, EvPassV1() returns:1.
Since Frame #141, EvPassV1() returns:0.
Since Frame #156, EvShootV2() returns:2.
Since Frame #160, EvPassV1() returns:-1.

the offensive team could have played the following path for a better outcome:

1. q0[Frame #72]: The two attackers are outside their offensive blue line.
2. q1[Frame #120]: The puck carrier is inside his offensive zone.
3. q5[Frame #130--155]: The puck carrier shot towards the goal in his offensive zone.

In the analysis process, two evaluation functions, i.e., EvPassV1() and EvShootV2(), are used to estimate possible passing/shooting path(s) and check the feasibility of taking the two primitive play actions/events (Pass and Shoot) from the offensive team's perspective.

- EvPassV1() returns 1, 0 or -1, indicating it is feasible, risky or unwise to pass respectively.

- EvShootV2() returns a value between 1 and 5, indicating to which extent it is feasible to shoot (a bigger value means a greater extent).

The evaluation results are visualized (frame by frame if applicable) and the visualization effect in every frame is kept from the start to the end in the bottom window of Figure 1.3. The possible pass path area is filled with horizontal lines in green, diagonal crossing lines in yellow, or vertical lines in red to represent the case of feasible, risky or unwise to pass respectively. Similarly, the possible shot path area is filled with one of the specific line patterns in one color to represent the specific
extent(s) of feasible to shoot, i.e., green horizontal lines for the extents of 4 and 5, yellow diagonal crossing lines for the extents of 2 and 3, and red vertical lines for the extent of 1. The analysis results show the offensive team could have shot towards the goal since Frame #130 (Figure 1.1(c)), instead of making a potential pass, in order for a more favorable outcome.

As for prediction, we apply several predefined perturbations to the two attackers' trajectories and redo the analysis. Figure 1.4 is the output after applying one of the perturbations. By comparing it with Figure 1.3, we can see that the feasible region (filled with green horizontal lines) for a pass has grown larger and longer and that the unwise region (filled with red vertical lines) for a pass has disappeared. This suggests a better outcome than what actually occurred for the offensive team. The animated versions of Figure 1.3 and Figure 1.4 are available through http://www.cs.ubc.ca/nest/lci/thesis/fhli/videoclips. We have not come up with a general algorithm to automatically calculate optimum perturbations to players' original positions in order for them to achieve more favorable outcomes. This topic will be addressed further in the future.

1.4 Contributions

We develop a flexible and extensible framework to describe, analyze and predict player actions in selected hockey game situations. We design data structures to store players' augmented trajectory data, define evaluation functions to check the feasibility and effectiveness of primitive play events/actions under various conditions, build FSM models to represent play knowledge in specific game situations, and implement a simple FSM based, puck carrier centered analysis strategy. The augmented trajectory data and the analysis results are visualized with a simple 2D
graphic animation. The framework provides a foundation for development of more elaborated analysis modules for hockey and for other applications.

1.5 Outline of the following chapters

In Chapter 2, we survey relevant work and position the thesis among them. Chapter 3 presents the overall design and assumptions made for simplification. Chapter 4 provides the implementation details, and Chapter 5 the results and evaluation. In Chapter 6, we draw conclusions and point out directions for future work.
Chapter 2

Relevant work

We survey relevant work in this chapter. The first section reviews sports video analysis, including tracking players, recognizing events, and analyzing play processes. The second section refers to five roles played by knowledge representation and introduces several relevant formalisms. The third section is on qualitative reasoning and geometry computation. We summarize these works in the fourth section.

2.1 Sports video analysis

Sports video analysis is an active research area in computer vision. Topics in this area include: segmenting video sequences into camera shots, tracking players (and the ball), mapping low-level features to high-level events (i.e., classification and indexing), generating natural language description, recognizing players' intentions, analyzing the play process, etc.

For instance, (Boreczky and Rowe, 1996)[4] presented a comparison of several algorithms and their variations for detecting and classifying video shot boundaries. (Koprinska and Carrato, 2001)[17] surveyed existing segmentation techniques for
both compressed and uncompressed video, as well as algorithms for camera motion recognition. [URL][34] is a project focusing on automatic analysis of soccer video data. It tries to use domain-specific knowledge and statistical or deterministic decision rules to locate and extract interesting events from the video data. The long-term goal of the project is to derive high-level semantics from low-level media features.

We focus on reviewing work to acquire players’ (and the ball’s) trajectories, to recognize high-level play events, and to understand/analyze the play.

2.1.1 Trajectory acquisition

[Intille, 1994][14] summarizes traditional object tracking algorithms and proposes a new one, “closed-world” tracking for tracking players in the football domain. A “closed-world” is defined as a spatial-temporal image region about which we know some contextual information, such as the objects that always exist within that region, and all the other objects that might enter into or leave from that region. Based on the contextual information, the algorithm analyzes the region locally, selects context-specific features, and adapts the manually initialized player template for tracking. The boundaries of the closed-worlds around players are manually assigned in the first frame, and the new closed-world regions for the next frame are computed based on motion difference blobs and the tracked positions of the players. All these computations are performed in the image frames registered to the football field model after camera motions are removed. The input video data were captured with a pan and zoom camera above the football field. Though the tracking results are not accurate enough to calculate players’ velocities and accelerations, they do tell where the players are. This work shows that high-level domain knowledge can
be used to improve low-level feature tracking.

(Needham and Boyle, 2001)[22] uses a CONDENSATION based stochastic approach to track multiple players through occlusion, congestion and scale. The aim is to automatically track sports players for positional behavior analysis. Their tracking results are usable, i.e, 56% of the players’ trajectories in indoor soccer games are within one meter of the hand marked-up ones.

(Misu et al., 2002)[20] uses a ladder structure integrating color, texture and motion features to track soccer players through occlusion, deformation and congestion. The structure consists of observation units and prediction units, and each of the observation units with different pattern matching algorithms is executed step-by-step to update the state vector measuring the reliability of the observation. The algorithm could detect tracking failures and could be easily extended by adding more features for tracking into the ladder structure.

(Okuma, 2003)[23] acquires players’ trajectories from ice hockey game videos. The approach registers each frame in a video sequence to a globally consistent rink model by automatically detecting/tracking rink features in the frame, fitting them to the model and then solving the homography matrix. A color-based sequential Monte Carlo method is used to track players in the video sequence. The tracked positions in each image frame are registered to the rink model with the solved homography matrix. This approach works well if each key frame in the input video sequence has enough detectable widely distributed rink features. The registered trajectories serve as major components of the augmented trajectory data used in the thesis.
2.1.2 Event recognition

(Gravila, 1999)[7] discussed promising application domains of “Looking at people” technology and reviewed works on visual analysis of gestures and whole-body movement, classified as 2-D approaches with or without explicit shape models or 3-D approaches, as well as techniques used in recognizing human actions based on the features extracted with those approaches.

(Gong et al., 1995)[8] proposes an approach to classify broadcasting soccer video sequences into different categories, such as shot at left goal, top-left corner kick, play in right penalty area, in midfield, etc, based on a priori model composed of the soccer pitch, the ball, the players and the motion vectors. It assumes that the input soccer video has been segmented into individual camera shots, detects line marks, motion, the ball and players in representative frames, and derives those semantic indexes from the play positions, play movements, ball positions and its motion vectors, and players’ uniform colors.

Similarly, (Zhong et al., 2000)[40] described a framework for sports video structure discovery and event detection by using domain-specific knowledge and generic machine learning techiques. They explored the well-defined temporal structure in sports video and focused on generating event indexes on-line.

(Sudhir et al., 1998)[31] present a system for automatic classification of tennis video, i.e., extracting and mapping low-level visual features to high-level play events, such as baseline rallies, passing shots, net games and serve-and-volley games. See Figure 2.1 for its components.

(Tan et al., 2000)[32] developed a technique to estimate camera motion directly from MPEG-encoded video data. They applied this technique to analyze and annotate compressed basketball video, based on the assumption that camera
Figure 2.1: Block diagram of the tennis video analysis system, from [31]
motion in basketball video could be used to derive high-level video content such as wide-angle and close-up views, fast breaks, probable shots at the basket, etc (Drew, 1997)[28].

(Zhou et al., 2000)[41] proposes a rule-based video semantic classification system for on-line basketball video indexing. Through supervised learning, the system builds a decision tree composed of if-then rules to associate high-level video events, such as left fast-break, right dunk and close-up shots, with low-level image features, such as motion, color and edge. Then the decision tree is used to index new basketball video clips on-line.

(Petokvic and Jonker, 2001)[25] proposes a framework to automatically infer semantics from raw video data by using a layered description model COBRA (COntent-Based RetrievAl) in line with MPEG-7. Figure 2.2 shows the structure of the COBRA video data model. The framework integrates two approaches for mapping low-level audiovisual features to high-level concepts/events (e.g., player near the net, forehand volley, etc.): rule-based spatio-temporal formalization of events and stochastic (Hidden Markov Models) recognition of events. Normal TV Broadcast tennis video was tested in the experiment, with promising results.

2.1.3 Play analysis

The System SOCCER (Andre et al., 1988)[1] automatically and simultaneously generates natural language description from soccer game video sequences (recorded with a static TV-camera) for listeners who were not watching the game. Figure 2.3 from [1] is an overview of the system. It takes as input geometrical description of the scene (Geometrical Scene Description, see (Herzog, 1995)[11] for an explanation), including a stationary background and trajectories of dynamic objects. The trajectories
Figure 2.2: The layered hierarchy of the COBRA video data model, from [25]
Figure 2.3: The architecture of the SOCCER system, from [1]
are provided by a vision system with a special trajectory editor. Propositions about what is happening at the moment are produced by the incremental event recognition module. The selection/linearization component chooses relevant propositions, sorts them and passes them on to the encoding processes, which further transform non-verbal information in those propositions into ordered written or spoken words in German. All the processes in the system run in parallel. One limitation of the SOCCER system is that it cannot recognize collaborations among several players, such as a team’s attack.

Players’ intentions are added into the output description of the SOCCER system by another component, REPLAI (REcognition of PLans And Intentions), as presented in (Retz-Schmidt, 1988)[26]. It argues that intention is the uncompleted parts of a plan that has been started. The knowledge about standard goals and plans in REPLAY is then formalized as a hierarchy. There are two distinct parts in that representation, i.e., specialization hierarchy as shown in Figure 2.4[26], and decomposition hierarchy, shown in Figure 2.5[26]. Figure 2.6[26] shows the architecture of the whole REPLAI component. Refer to (Herzog and Rohr, 1995)[12] and (Herzog, 1995)[11] for more information about the background project VITRA (VI­su­al TRAnslator), which deals with the relationship between natural language and vision, with the aim of developing systems to describe image sequences in natural language.

(Lashkia et al., 2003)[18] presents a software tool for assisting team play analysis in soccer. Based on a probability model using color information, it detects the ball and players in video sequences recorded with a single digital video camera, registers their camera coordinates to a field model, simulates the image scenes with 2D and 3D graphic animation, and then analyzes dominant areas of each team. As
Figure 2.4: Part of the plan hierarchy in REPLAI, from [26].

AKO means "a kind of" (specialization), p: stands for "precondition", is: represents "intended state", and sch. means "schema".

Figure 2.5: Plan schema (decomposition) for "solo", from [26]
Lashkia et al. point out, this tool does not integrate players' personal abilities and needs to improve the accuracy of detecting players and the ball when it is off the ground.

2.2 Knowledge representation

Knowledge representation is crucial in Artificial Intelligence. (Davis et al., 1993)[5] argues that "the notion can be best understood in terms of five distinct roles it plays", as quoted below:

- A knowledge representation (KR) is most fundamentally a surrogate, a substitute for the thing itself, used to enable an entity to determine consequences by thinking rather than acting, i.e., by reasoning about the world rather than taking action in it.

- It is a set of ontological commitments, i.e., an answer to the question: In what terms should I think about the world?
• It is a *fragmentary theory of intelligent reasoning*, expressed in terms of three components: (i) the representation's fundamental conception of intelligent reasoning; (ii) the set of inferences the representation sanctions; and (iii) the set of inferences it recommends.

• It is a *medium for pragmatically efficient computation*, i.e., the computational environment in which thinking is accomplished. One contribution to this pragmatic efficiency is supplied by the guidance a representation provides for organizing information so as to facilitate making the recommended inferences.

• It is a *medium of human expression*, i.e., a language in which we say things about the world.

(Little and Gu, 2001)[19] presents a novel representation for trajectories: path curve and speed curve, which separates the positional information from the temporal information. They derive a local geometric description of the curves which is invariant under scaling and rigid motion. Two curves are matched by warping their feature vectors with dynamic programming. They use R-trees to index the multidimensional features for improving search efficiency in a large database of trajectories.

(Harel, 1987)[9] proposes a visual formalism called statecharts for specifying and designing complex discrete-event systems (e.g., digital-control units). The author claims a complex system cannot be beneficially described in conventional finite state machines and their corresponding state-transition diagrams (or state-diagrams for short), "because of the unmanageable, exponentially growing multitude of states, all of which have to be arranged in a 'flat' unstratified fashion, resulting in an unstructured, unrealistic, and chaotic state diagram." [9] Statecharts
extend state-diagrams "by AND/OR decomposition of states together with interlevel transitions, and a broadcast mechanism for communication between concurrent components", enhance their capabilities in dealing with the notions of hierarchy, concurrency and communication, and "transform the language of state diagrams into a highly structured and economical description language"[9]. The extension is shown in the following equation (from [9]):

\[
\text{statecharts} = \text{state-diagrams} + \text{depth} + \text{orthogonality (i.e., concurrency)} + \text{broadcast-communication}.
\]

(Arens and Nagel, 2002)[2] presents a behavior knowledge representation for planning and plan-recognition in a cognitive vision system (Nagel, 2001)[21]. They use Situation Graph Trees (SGTs) to provide necessary conceptual knowledge about vehicle behavior for interpreting image sequences of road traffic scenes. A planning formalism, Hierarchical Task Networks (HTNs), is compared with SGTs to check whether and to what extent the conceptual knowledge represented with SGTs may be recast with HTNs, which can be used in a driver assistance system to generate synthetic images from textual descriptions.

2.3 Qualitative reasoning and collision detection

(Kawashima et al., 1994)[15] proposed an algorithm to qualitatively analyze group behavior in soccer video sequences. The algorithm uses the histogram backprojection to detect and classify each player in a color image. A group hierarchy (see Figure 2.7[15]) is formed by applying Gaussian filters to the output image of the histogram backprojection. Finally an interpretation to the group behavior is derived from the temporal aspect of qualitative relations (e.g., disconnected, overlaped, or
The relations among regions form a tree structure. (a) A simulated example of smoothed regions. (b) The structure of the regions of (a).

part of) among groups. The interpretation does not include exact motion information such as velocity, direction, or its distribution. See (Forbus, 1997)[6] for an introduction to qualitative representations and qualitative reasoning techniques.

(Basch, 1999)[3] proposes a general approach to solve problems interwining discrete and continuous aspects in modeling objects in space, e.g., detecting collisions between moving objects. A Kinetic Data Structure (KDS) is introduced to compute and keep track of discrete attributes, such as the closest pair, the convex hull and the minimum spanning tree. Based on the KDS, (Speckmann, 2001)[30] proposes a simple and compact structure to detect collisions between moving polygonal objects in a 2D plane.

2.4 Summary of relevant work

We discussed relevant work in the literature, including:
• Sports video analysis, which addresses acquisition of players’ trajectories, recognition of higher-level play events, description of image sequences and analysis of players’ intentions;

• Knowledge representation, which provides design rules and useful formalisms to represent trajectories and domain knowledge; and

• Qualitative reasoning and collision detection, which help us reason about play effectiveness and predict better alternative actions according to play knowledge and augmented players’ trajectory data, which might not be as accurate as needed.

Many research problems in sports video analysis, such as tracking players (and the ball), are still open ended. Suggested partial solutions to them often take advantage of specific domain knowledge and/or use statistical methods and succeed under some circumstances, leaving much room to improve. Finite state machines and statecharts [9] are widely used as efficient formalisms to represent knowledge/processes in various domains. Efficient representation for massive spatio-temporal data (e.g., 3D or 4D trajectories) is highly in need too. With qualitative reasoning techniques, we can infer correct conclusions from not-so-accurate input data, which means qualitatively correct assumptions can be made to simplify reasoning processes.
Chapter 3

Design and assumptions

In this chapter, we present our overall design and the assumptions we make. Section 1 introduces the structural overview of a hockey game video analysis system and establishes our foci. Section 2 explains the trajectory data and augmented information which supports the later analysis. In Section 3, we describe primitive play events and evaluation functions attached to these events. In Section 4, we present definitions of selected game situations, FSM (Finite State Machine) models for play knowledge in these situations and situation analysis strategies. The last section introduces major modules of the design.
Figure 3.1: Architectural overview
EEFs: Event Evaluation Functions, FSMs: Finite State Machines, SAMs: Situation Analysis Modules
3.1 Architectural overview

The goal is to develop a semi-automatic system (as introduced in Chapter 1) for hockey game video analysis. We propose the primary architecture depicted in Figure 3.1, where rectangles with dashed sides stand for storage items such as databases, data structures, and function libraries; rectangles with solid sides mean activities/modules; and solid arrow lines are flows of information items (Harel, et al 1990)[10].

Input includes hockey game videos and hockey domain knowledge. All the videos tested in our framework are recorded TV broadcast professional matches, including examples from the Canada Cup in 1987 and recent NHL games. Our domain knowledge is that which generally applies in professional matches. In our analysis and prediction, we assume that each player possesses the necessary skills to play at a high level. For example, certain passes or shots that we deem feasible might not be so at a novice level.

After digitizing a video clip into image sequences, we either manually or with the help of any available automatic player tracker, play event recognizer and game situation detector, build the necessary ATD information for later analysis. We assume we have a proper rink model, definitions of each primitive play event (the action and roles associated with it), definitions of specific game situations and play knowledge in the situations, and analysis strategies.

We focus on designing modules and storage items identified within the large box with dotted sides in Figure 3.1. The main storage items are listed in the left column of Table 3.1, with brief descriptions in the right column. Similarly Table 3.2 lists the main modules. We explain these modules and storage items in the following sections.
### Table 3.1: Main storage items

<table>
<thead>
<tr>
<th>Storage Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ATD</strong></td>
<td>augmented trajectory data, including information on trajectories, possession of the puck, skating forward/backward, shooting left/right, etc.</td>
</tr>
<tr>
<td><strong>SPEs</strong></td>
<td>sorted primitive play events, with associated roles defined for each event/action</td>
</tr>
<tr>
<td><strong>ImgSeqInfo</strong></td>
<td>information on the image sequence such as frequency, players' positions in each image, etc.</td>
</tr>
<tr>
<td><strong>ClipInfo</strong></td>
<td>information on a video clip, including ATD, SPEs, ImgSeqInfo, etc.</td>
</tr>
<tr>
<td><strong>EEFs</strong></td>
<td>event evaluation functions, evaluating effectiveness of play events</td>
</tr>
<tr>
<td><strong>FSMs</strong></td>
<td>finite state machines, representing specific game situations</td>
</tr>
<tr>
<td><strong>SAMs</strong></td>
<td>situation analysis modules, implementing various analysis strategies</td>
</tr>
<tr>
<td><strong>Options</strong></td>
<td>various options</td>
</tr>
<tr>
<td><strong>IIS</strong></td>
<td>illustrated image sequences, showing process results</td>
</tr>
<tr>
<td><strong>SGA</strong></td>
<td>simple 2D graphic animation, showing process results</td>
</tr>
<tr>
<td><strong>TE</strong></td>
<td>textual explanation, showing process results</td>
</tr>
</tbody>
</table>

### Table 3.2: Main modules

<table>
<thead>
<tr>
<th>Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Configure</strong></td>
<td>configure Options and which EEFs, FSMs, SAMs to be used while processing the video clip</td>
</tr>
<tr>
<td><strong>Visualize</strong></td>
<td>visualize process results stored in main storage items</td>
</tr>
<tr>
<td><strong>Describe</strong></td>
<td>describe the video clip by calling specified situation analysis modules</td>
</tr>
<tr>
<td><strong>Analyze</strong></td>
<td>analyze the video clip by calling specified situation analysis modules</td>
</tr>
<tr>
<td><strong>Predict</strong></td>
<td>make prediction about the video clip by calling specified situation analysis modules</td>
</tr>
</tbody>
</table>
3.2 Augmented trajectory data

One basis for reasoning about play in some game situation is the raw trajectory of each player involved in the situation. For example, in a 2-on-1 offensive attack, we need the trajectories of the two attackers and the opponent defenceman, recorded from the start of the situation to the end of it. Currently we only deal with reasoning in the 2D space of rink coordinates.

We take a player's trajectory to be a sequence of temporal-spatial tuples \((T, X, Y)\), where \(T\) is the time passed since the start time of the situation, and \((X, Y)\) are the coordinates of the player's position in the rink frame at time \(T\). We simplify a player's position to be one point on the rink surface where the player's left or right foot contacts the ice. We further assume the trajectory data is accurate enough for our analysis.

Currently we can not extract accurate trajectories of the puck from the video clips. Although it is crucial to know the puck's exact position in order to analyze play under some circumstances such as judging offside violations (see [13] for details), the puck's trajectory is assumed to be the same as that of the puck carrier (i.e., the player who has possession and control of the puck or the player who has possession of the puck after the puck becomes loose, see [13] for details), or more reasonably, we assume the puck's position is fixed relative to the puck carrier. This assumption about the puck's trajectory does not affect the qualitative analysis of the situations we include in this thesis.

We augment a player's raw trajectory data with his individual profile and with information on his skating forward/backward or falling down, and his possession of the puck in the play. Currently, each player's profile includes only two attributes:
1. **SHOOTS**: i.e., whether the player shoots left or right (we do not exclude the case that some players can shoot both left and right), which influences his playing (attacking and defending) coverage; and

2. **MaxSpeed**: i.e., how rapidly the player can skate in the rink, which is a major factor affecting the perturbation to his position in the prediction module.

Other attributes such as height, weight, team position, acceleration/deceleration, turning abilities and statistical tendencies could be included in future versions. Players' skating forward/backward or falling down affects their play coverage too. We put this into a category called **MODE**. With information on possession of the puck, we know who is the current puck carrier (this player normally is the focus of the analysis), which player passed the puck to a teammate or lost the puck to his opponent, and which players could receive a pass from the current puck carrier. Thus at any time $T$, we can get a tuple

$$(T, X, Y, \text{SHOOTS}, \text{MaxSpeed}, \text{MODE}, \text{PuckPossession})$$

for each player visible in the video clip. With a sequence of these tuples, we also can estimate the player's instantaneous velocity, as required. The representation for the augmented trajectory data is flexible (e.g., can be separated into path curves and speed curves (Little and Gu, 2001)[19]) to satisfy requirements in the analysis and the prediction.

### 3.3 Play events and evaluation functions

Hockey game behaviors are hierarchical in various ways. For example, Figure 3.2 shows a player-centric hierarchy. While from a team's perspective, the behaviors
Figure 3.2: A player-centric hierarchy of hockey game behaviors

Table 3.3: A team-centric hierarchy of hockey game behaviors

<table>
<thead>
<tr>
<th>Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Behaviors of individual players, such as pass, shoot, cross the blue line, skate forward/backward, fall down, lose possession of the puck, etc.</td>
</tr>
<tr>
<td>2</td>
<td>Offensive/defensive behaviors in specific game situations, such as 2-on-1 offensive attacks, power play shots from the point, defensive zone breakouts, etc.</td>
</tr>
<tr>
<td>3</td>
<td>Game strategies. Play styles change as a function of current game score, time remaining, and other relevant factors.</td>
</tr>
<tr>
<td>4</td>
<td>Team strategies. Play styles change as a function of player personnel, standings and time in the season.</td>
</tr>
<tr>
<td>5</td>
<td>Franchise strategies. Teams develop new strategies over the course of seasons in multiple years.</td>
</tr>
</tbody>
</table>
can be classified into 5 levels, shown in Table 3.3, forming another hierarchy.

The framework we develop currently represents and reasons about some behaviors in the first two levels, shown in Table 3.3. It is extensible in the sense that more behaviors at the same or higher levels can be added. It also allows current reasoning modules (SAMs) to be superseded with more sophisticated ones.

We choose a set of behaviors in Level 1 (Table 3.3) as primitive play events (mainly players’ actions), naming them as PASS, SHOOT, SKATEForward, SKATEBackward, FALLDown, STANDUp, CROSSBlueLine, CROSSCenterLine, CROSSGoalLine, ENTERPlay, LEAVEPlay, DUMPuck, PICKupPUCK, and LOSEPUCKPossession. These behaviors are sufficient for us to process all the video clips used in this thesis. One can add other behaviors, such as SKATEWIDER, into this set as the system evolves to deal with more complicated video clips or with the same video clips at a more detailed granularity. However, all the behaviors in the set are primitive in the sense that any of them can be included in any situation analysis if it makes sense and if the user wants to do so. For instance, in both a 2-on-1 offensive attack and a power play shot from the point, users can evaluate the effectiveness of the puck carrier’s shooting the puck towards the goal and/or the riskiness of passing the puck to a teammate. A user can also evaluate the puck carrier’s CROSSBlueLine in those two situations.

Each primitive play event has two components, the action it implies and the player roles associated with it. The action is obvious, as suggested by the name of the event. The roles include players involved in the action and any relevant marks in the rink model, if needed. For example, roles in CROSSBlueLine are the player who crossed the blue line and the particular blue line in the rink model. Although it is important to know the puck’s exact position at this moment if we want to
judge offside violations, we make the assumption that the puck is in a reasonable neighborhood of the puck carrier, and its position does not quantitatively affect the analysis results (i.e., the results match what actually happened in the video clip). The direction of crossing is important too, but it can be computed from the trajectory data, so we do not explicitly represent it with another component. Table 3.4 summarizes the roles involved in the primitive play events.

Table 3.4: Primitive play events and the involved roles

<table>
<thead>
<tr>
<th>Event/Action</th>
<th>Role(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PASS</td>
<td>Puck Carrier, Receiver (the puck carrier passes the puck to the receiver)</td>
</tr>
<tr>
<td>SHOOT</td>
<td>Puck Carrier, Goal (the puck carrier shoots the puck towards the goal)</td>
</tr>
<tr>
<td>SKATERFORWARD</td>
<td>Player (who just started skating forward)</td>
</tr>
<tr>
<td>SKATEBACKWARD</td>
<td>Player (who just started skating backward)</td>
</tr>
<tr>
<td>FALLDOWN</td>
<td>Player (who just fell down)</td>
</tr>
<tr>
<td>STANDUP</td>
<td>Player (who just stood up)</td>
</tr>
<tr>
<td>CROSSBLUELINE</td>
<td>Player, Blue Line (the player just crossed the blue line)</td>
</tr>
<tr>
<td>CROSSCENTERLINE</td>
<td>Player (the player just crossed the center line)</td>
</tr>
<tr>
<td>CROSSGOALLINE</td>
<td>Player, Goal Line (the player just crossed the goal line)</td>
</tr>
<tr>
<td>ENTERPLAY</td>
<td>Player (who just entered the play)</td>
</tr>
<tr>
<td>LEAVEPLAY</td>
<td>Player (who just left the play)</td>
</tr>
<tr>
<td>DUMPPUCK</td>
<td>Player (who just dumped the puck)</td>
</tr>
<tr>
<td>PICKUPPUCK</td>
<td>Player (who just picked up the puck)</td>
</tr>
<tr>
<td>LOSERPUCKPOSSESSION</td>
<td>Puck Carrier, Opponent Player (the puck carrier just lost possession of the puck to the opponent player)</td>
</tr>
</tbody>
</table>

Note that PASS occurs when the puck carrier passes the puck to his teammate, while LOSERPUCKPOSSESSION occurs when a player from the opponent team gets the puck, and PICKUPPUCK occurs whenever a player from either the same team or the opponent team gets the puck. In fact, PASS consists of two actions: DUMPPUCK and PICKUPPUCK, i.e., the puck carrier dumped the puck first and
then his teammate picked up the puck. In current implementation, without affecting the qualitative analysis results, we assume the DUMPPUCK and the PICKUPPUCK happen as the same event and take these two players as roles involved in the PASS primitive play event. Similarly, this assumption applies to the primitive play event LOSEPuckPossession. It is difficult to give quantitative definitions to distinguish DUMPPUCK and SHOOT in all cases. We assume some meta event recognizer would provide enough information to make this distinction if it were needed. Some primitive play events listed in Table 3.4, including STANDUP and LEAVEPLAY, are provided here just for completeness. They are not part of the analysis of any of the situations currently considered.

Each primitive play event is associated with an evaluation function to analyze the action's effectiveness in different ways. All evaluation functions have the same interface, i.e., they take as input parameters a pointer to the augmented trajectory data, a time interval for evaluating the action, a pointer to custom data such as roles involved in the event, and output a value representing the evaluation result. The output can be as simple as one of the three values, “feasible”, “risky” and “unwise”, suggesting respectively that if the action occurs during the time interval the outcome will be favorable, will be risky (even though it still could be favorable), and will be unwise. See formal definitions of these evaluation functions in Section 4.2. Of course, for some events such as LOSEPuckPossession, the output will be always “unwise”. The key idea is to make multiple evaluation functions available for each primitive play event. Users then can analyze the event from different perspectives and to different granularities. One hockey maxim is that shooting at the net is never a bad thing. A simple evaluation function for SHOOT might always return “feasible”. On the other hand, as with the situation of a power play shot from the point, there
are some shooting situations that are more favorable than others. A more complex
evaluation function for \texttt{SHOOT} is given in Section 4.2.

We do not attempt to recognize automatically play events from image se­
quences. Instead, we assume the existence of an event recognizer that outputs a list
of the primitive play events that happened in the image sequence in time order into
the storage item \textit{ClipInfo}. This list, called Sorted Play Events (\textit{SPEs}), will serve
later analysis of the video clip. Currently, this list is provided manually. The im­
plementation details of these primitive play events, their evaluation functions, and
the sorted list of those events will be presented in Chapter 4.

3.4 Game situations, play knowledge and analysis strate­
gies

3.4.1 Definitions of specific game situations

In the team-centric hierarchy of hockey game behaviors, above the level of individual
player's behaviors (primitive play events) is the level of offensive/defensive behaviors
in specific game situations. We focus on representing and reasoning about simple
behaviors in three clearly identifiable game situations: 2-on-1 offensive attack, power
play shot from the point, and defensive zone breakout. As the system evolves, we
expect to add more behaviors and game situations to it.

The definition of a specific game situation has three parts, describing its
start, its end and an evaluation for each end respectively. The start defines under
what circumstance the situation starts, including information on possession of the
puck, the play event just occured, and the geometrical configuration of players with
respect to each other and possibly to the location in the rink. The end describes
what condition or conditions bring the situation to the end. There might be more than one end for each game situation. We provide a comparative evaluation of each end, as “good”, “bad”, or “neutral”, from either the offensive or the defensive team’s perspective. Here, we use natural language definitions of situation starts and ends and identify them manually in video clips. The definitions are formalizable and, in principle, can be used to detect the start and end automatically.

Definitions for 2-on-1 offensive attack, power play shot from the point and defensive zone breakout are given in Table 3.5, 3.6 and 3.7 respectively. The definitions here are as simple as possible (see the notes in the tables).

<table>
<thead>
<tr>
<th>Situation</th>
<th>2-on-1 offensive attack</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start</td>
<td>Two players attack (one of them has possession of the puck) and there is only one opposite defenceman closer to the goal than the two attacking players.</td>
</tr>
<tr>
<td>End</td>
<td>E1: the puck carrier shoots towards the goal in their attacking zone. E2: another offensive player enters the play. E3: the offensive team loses possession of the puck. E4: another defender enters the play. E5: the puck carrier shoots towards the goal outside their offensive blue line.</td>
</tr>
<tr>
<td>Evaluation</td>
<td>Perspective: offensive E1, E2 are good; E3, E4 and E5 are bad.</td>
</tr>
<tr>
<td>Notes</td>
<td>The outcome of a 2-on-1 offensive attack is never neutral since the offence begins with a distinct advantage. The offensive side should be able to get a shot on goal in this situation. E2 is good since the 2-on-1 changed into a 3-on-1. E5 is unwise because such a long shot is easy for the goalie to stop.</td>
</tr>
</tbody>
</table>
Table 3.6: Definition of power play shot from the point

<table>
<thead>
<tr>
<th>Situation</th>
<th>power play shot from the point</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start</td>
<td>In a 5-on-4 power play setting, all the 5 attackers are in the attacking zone, and the attacker in one of the point positions has possession of the puck.</td>
</tr>
</tbody>
</table>
| End       | E1: the offensive team loses possession of the puck.  
|           | E2: the puck carrier passes the puck to a teammate who is not in the other point position.  
|           | E3: the puck carrier shoots the puck towards the goal. |
| Evaluation| Perspective: offensive  
|           | E1 is bad, E2 is neutral, and the outcome for E3 is whatever the selected evaluation function attached with SHOOT (one of the primitive play events) outputs. |
| Notes     | 1. We only deal with one case of power play, i.e., 5 attackers vs. 4 opposite players, and mainly analyze available options for the puck carrier in the point area.  
|           | 2. The outcome of pass to a teammate who is at the other point position is handled as an internal loop with roles (the puck carrier and the other point player) exchanged. Alternatively, the situation could end with another instance begins.  
|           | 3. Any shot on net is a good thing. Thus the default shot evaluation function returns “good”. The framework allows using more elaborate shot evaluation functions which may include analysis of the shooter’s position and the positions of other players, both offence and defence. |
Table 3.7: Definition of defensive zone breakout

<table>
<thead>
<tr>
<th>Situation</th>
<th>Definition of defensive zone breakout</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start</td>
<td>A player has possession of the puck behind the net in the player’s defensive zone.</td>
</tr>
<tr>
<td>End</td>
<td>E1: the team advances the puck beyond their defensive blue line while maintaining possession of the puck. E2: the team loses possession of the puck in their defending zone.</td>
</tr>
<tr>
<td>Evaluation</td>
<td>Perspective: offensive E1 is good and E2 is bad.</td>
</tr>
<tr>
<td>Notes</td>
<td>A team actually becomes offensive once it has possession of the puck. In some video clips on hand, not all the players on ice are visible in the image sequence, and we only analyze options for the players whose trajectory data are available.</td>
</tr>
</tbody>
</table>

3.4.2 Modeling play knowledge with FSMs

We use an FSM (Finite State Machine) model to represent specific game situations.

An FSM $M$ is defined as a quintuple:

$$M = (A, Q, I, J, R)$$

where

- $A$: a set of primitive play events, as described in Section 3.3,
- $Q$: a set of states,
- $I$: a subset of $Q$ assigned as initial states,
- $J$: a subset of $Q$ defined as final states, and
- $R$: a state transition map $Q \times A \rightarrow Q$.

We can represent play knowledge about one specific game situation with several different FSMs. Each would represent the play from the perspective of either
the offence or the defence, or possibly even more specifically, from the viewpoint of a particular player role defined in the game situation, such as the puck carrier in the point position for a 5-on-4 power play. Figure 3.3, 3.4 and 3.5 are diagrams of the standard FSMs for the three game situations: 2-on-1 offensive attack, power play shot from the point and defensive zone breakout, which are defined in Table 3.5, 3.6 and 3.7 respectively. Details on how to input and store information represented by an FSM will be discussed later in Section 4.1.3.

Figure 3.3: FSM model for 2-on-1 offensive attack
OP: the puck carrier; O: the other offensive player; \(q_i\) \((i = 0, \ldots, 5)\): states; \(q_0\): the initial state; \(q_i\) \((i = 2, \ldots, 5)\): final states; \(a_i\) \((i = 0, \ldots, 4)\): primitive play events.

Each state in \(Q\) represents a state which the specific situation can start in, reach, or end up in. When defining the states, we take into account three factors: previous primitive play events/actions, current players’ geometric configuration, and possession of the puck. By default, except for the final states, any state defined in an FSM could be an initial state. But, one can explicitly assign a subset of \(Q\) as the initial state set \(I\). Having a state \(q_i \in I\) means the FSM can deal with any particular instance of the specific situation which starts from that state \(q_i\). For each final state
Figure 3.4: FSM model for the power play shot from the point
OP: the offensive puck carrier; \( q_0 \): the initial state; \( q_i \) \((i = 1, \ldots, 3)\): final states; \( a_i \) \((i = 0, \ldots, 3)\): primitive play events.

Figure 3.5: FSM model for the defensive zone breakout
\( q_0 \): the initial state; \( q_2, q_3 \): final states; \( a_i \) \((i = 0, \ldots, 6)\): primitive play events.
in J, there is at least one evaluation function attached with it. These functions judge whether the final state, as one possible end of the specific game situation, is good, bad, neutral or some other value specified to a higher level of granularity, from a particular perspective. They call evaluation functions attached with the primitive play event which caused the transition to the final state. The state transition map \( R \) defines the relationships among states and primitive play events with sequences of 3-tuple \((q_i, a_j, q_k)\), where \( q_i, q_k \in Q \) and \( a_j \in A \), meaning that \( a_j \) causes the state changes from \( q_i \) to \( q_k \).

3.4.3 Situation analysis modules

There are different objectives in the consideration of play in specific game situations. We implement three distinct levels of consideration:

1. Description: describe what primitive play events happened in the actual play based on a parse of the sorted list of these play events as a path in an FSM of the game situation,

2. Analysis: find alternative feasible paths in the FSM which have the same initial state as the original path and final states with better evaluation results (returned by the evaluation functions attached with the final states), and

3. Prediction: alter the play events in the original path with respect to timing (e.g. shoot earlier) and positioning (e.g. skate wider) and check whether these perturbations might result in better outcome evaluation.

Other potential considerations include using multiple FSMs for a given game situation, using different evaluation functions for primitive play events, and focusing on different elements or perspectives in the process of analysis and prediction. Though
these approaches were not implemented in the thesis, we do provide a mechanism to allow inclusion of these considerations (i.e., building additional situation analysis modules offline and assigning online which modules to use in a particular situation).

A situation analysis module consists of functions in three categories, description functions, analysis functions and prediction functions. They respectively describe the actual play, analyze possible better alternative actions, and predict results of potential changes. Functions in each category have the same signature (i.e., number, order and data types of input/output parameters), so they could be called by the module Describe, Analyze, or Predict. Within one module which implements one analysis strategy, a prediction function can access results produced by either description functions or analysis functions in the same analysis module, if needed. Similarly, an analysis function can use results produced by description functions which belong to its analysis module.

3.5 Main modules

As shown in the architectural overview (Figure 3.1), there are 5 major modules: Configure, Describe, Analyze, Predict, and Visualize. The Configure module provides the user interface to assign which versions of evaluation functions should be attached to the corresponding primitive play events, which FSMs should be used for the specific game situations, and which analysis module should be used. It also calls subfunctions provided by the selected analysis module to configure necessary parameters and options for the description functions, the analysis functions and the prediction functions in the module, upon users' request. In addition, the Configure module allows the user to decide which visualization functions to use and sets relevant options for them.
After the user selects which analysis module and associated internal functions to use, the *Describe* module calls the selected description functions, the *Analyze* module calls the selected analysis functions, and the *Predict* module calls the prediction functions.

The *Visualize* module renders the output. It mainly simulates the actual play with simple 2D animation of figures representing players in a rink model, plots color lines or areas representing analysis results such as riskiness of alternative actions (pass or shot), and outputs the textual version. It also displays the original image sequence.
Chapter 4

Implementation

In this chapter, we present the implementation details. We use Qt (a multiplatform C++ GUI toolkit, Unix/X11 version) to develop the user interface, including the Visualize module. The major data structures for the input data including Augmented Trajectory Data (ATD), Sorted Play Events (SPEs) and Finite State Machines (FSMs), are described in Section 1. We consider the potential requirement to integrate the system with databases. In future, a large volume of input data may be stored in database tables or views. Thus we specify some particular fields in our data structures to communicate with database systems. Section 2 describes 4 evaluation functions for three primitive play events: PASS, SHOOT and LOSEPUCKPOSSESSION. We present the Configure module in Section 3, followed by the Describe, Analyze and Predict modules in Section 4, and the Visualize module in Section 5.
4.1 Input data

As described in the architectural overview (Figure 3.1), the input data include hockey game videos and specific hockey domain knowledge. In the previous chapter, we introduced the concepts of players’ ATD, video clips' SPEs, and FSMs for play knowledge in specific game situations. Now we present the detailed data structures used in the current implementation, the acquisition of ATD and SPEs from video clips, and the textual FSMs for 2-on-1 offensive attack, power play shot from the point and defensive zone breakout.

4.1.1 Data structures and major classes for the input data

First we define a C++ struct for each important entity which in future could be stored in database tables/views (currently they are stored as formatted text files), and then we define a wrapper C++ class for each struct to provide operations on this struct. All the structures are named with 'T' as the first letter (adding 'P' before 'T' represents a pointer pointing to this struct), and the wrapper classes are named starting with the letter 'C'.

We represent related information on a hockey game video clip with the following structures: TClip, TGame, TRink, TScoring, TPlayer, TTrajectory, TTrajectory, TPrimEvent, TShotOnGoal, TPuckPoss, TSkateMode, TCrossBL, TPass, and TLosePuckPossession, as listed in Figure 4.1 and Figure 4.2. Some other typical descriptions (such as penalties) are not included in the current implementation.

In these two figures, ClipID, GameID, RinkID, PlayerID (including PuckCarrierID, ReceiverID and OpponentPlayerID) and EventID identify clips, games, rinks, players and prim-
Figure 4.1: Data structures for the Augmented Trajectory Data

Figure 4.2: Data structures for the Sorted Play Events
itive play events respectively.

*Period, TimeStart* and *TimeEnd* represent the period (1st, 2nd, 3rd or overtime) in which the clip occurred and the starting and ending times.

*ImgSeqStart, FirstFrame, ImgSeqFrames* and *FrameRate* record information on the image sequence exported from the clip (by using Adobe Premiere 6.5), i.e., the path to the first image file in the sequence, the frame (number) from which the situation starts, the total number of image files and the frame rate, e.g.,

- *ImgSeqStart = "~/thesis/videoclips/2on1/van_cal/van_cal072.bmp",
- *FirstFrame = 72,
- *ImgSeqFrames = 97,
- *FrameRate = 30 (frames/second).*

*HSituID* is the heuristic situation identification (representing 2-on-1 offensive attack, power play shot from the point or defensive zone breakout), and *HDescription* is the heuristic description to the clip.

*Status* means whether the description for this clip has been generated.

*TeamA* and *TeamB* stand for the two teams in the game. *ScoreA* and *ScoreB* are their respective scores earned in the game.

Although rinks may have different sizes (see [13] for details), the rink model we used in the implementation is fixed (200 feet in length and 85 feet in width).

*T, X, Y, PuckPoss* and *SkaMode* represent a player's position coordinates *(X, Y)*, whether he has possession of the puck, and his current skating mode (backward, forward, falling down, etc.) at timing *T* (frame number).
TPuckPoss, TSkateMode, TCrossBL, TPass and T LosePuckPossession are tables built upon the augmented trajectory data, representing primitive play events such as PICKUPPUCK, SKATEFOWARD, SKATEBACKWARD, FALLDOWN, CROSSBLUELINE, CROSSCENTERLINE, CROSSGOALLINE, PASS and LOSEPUCKPOSSESSION, occurred in the clip. TShotOnGoal is a separate table storing information on the primitive play event SHOOT. TPrimEvent is a table built on those tables mentioned above, representing general primitive play events occurred in the clip.

PEvtInfo is a castable pointer to TShotOnGoal, TSkateMode, TCrossBL, TPass or T LosePuckPossession, which is indicated by EventID.

CSt and NSt are used in the later description process, recording current state in the FSM and the next state caused by the primitive play event.

Mark, Side and Direction tell which line (blue line, center line or goal line) in which side (left: A; right: B; center: C) and towards what direction (A to B, or B to A) is crossed by the player.

Wrapper classes for these data structures are listed in Figure 4.3, 4.4 and 4.5. All the classes except CPrimEvtList provide an operation init() to initialize relevant lists from formatted text files. The list of primitive play events (i.e., SPEs) happened in the clip is built by the operation CClip::setPrimEvtList(), which will be discussed in Section 4.1.2. Other operations provide read or write access to the data stored in the structures. The data structure TClipSituDAC and its wrapper class CClipSituDACList for recording the description, analysis and prediction are listed in Figure 4.13.

For the FSMs representing play knowledge in specific game situations, the
Figure 4.3: Wrapper classes for TClip, TGame and TScoring
Figure 4.4: Wrapper classes for TTrajectory, TPrimEvent, TShotOnGoal and TPlayer
Figure 4.5: Wrapper classes for TPuckPoss, TSkateMode and TCrossBL
data structures are listed in Figure 4.6, and Figure 4.7 is the wrapper class for them.

Figure 4.6: Data structures for the Finite State Machines

TEvent defines the data structure for related information on primitive play events and their evaluation functions.

- *Exp1* is a one-word explanation to the event, i.e., its action, as listed in Table 3.4.

- *Exp2* is a one-sentence explanation to the event, which includes format-control strings. These strings will be instantiated as actual *Roles* in the textual descriptions of video clips.

- *EEFs* store names of evaluation functions attached with this event. *EEF-Exps* are corresponding explanations to these functions.
Figure 4.7: Wrapper class for \textit{THFSM}, \textit{TEvent}, \textit{TState} and \textit{TTransition}
• Though in the definition the sizes of these string arrays are fixed (i.e., 4-by-32 bytes for Roles and 8-by-256 bytes for EEFs and EEFExps), we could dynamically allocate memory space to these fields if needed, so that the number of roles or evaluation functions will be changeable.

• EEFDefault indicates which evaluation function (i.e., the function's name is stored in EEFs[EEFDefault]) is the default one to be used in any analysis to the event. FSMEEF defines which evaluation function (i.e., the function's name is stored in EEFs[FSMEEF]) should be used in this particular FSM; it could be configured by the user during the analysis process.

• For instance, for the primitive play event PASS:
  
  - ID = “a0”,
  - Exp1 = “Pass”,
  - Exp2 = “%s passed the puck to his teammate %s.”,
  - EEFs[0] = “EvPassV1”,
  - EEFExps[0] = “Evaluate riskiness of passing the puck to other teammates, i.e., whether the pass will be defended by the defenders.”,
  - EEFDefault = 0, and FSMEEF = 0.

If in some video clip, the puck carrier is player “A1” and the receiver is “A2”, then Exp2 will be instantiated as “A1 passed the puck to his teammate A2.” in the description of this clip.

THFSM defines the data structure for Finite State Machines representing hockey play knowledge in specific game situations.
• SituID and Version together identify a unique FSM.

• Perspective indicates for which team (offensive or defensive) the situation's outcome is good, bad or neutral.

• Explanation briefly explains this FSM's characteristics.

• EventIDs, StateIDs and TransitionIDs store the FSM's events, states and transitions respectively.

TState defines the states in the FSM.

• ID identifies a unique state within the scope of the current FSM.

• Exp is a brief explanation of this state.

• BFinal indicates whether this state is a final state and if it is, Evf further points out the evaluation of it, either as "good", "bad" or "neutral", or the name of an evaluation function which returns some other value specified to a higher level of granularity.

• BVisited and PPath serve the analysis process and will be discussed in Section 4.4.

TTransition defines the transitions in the FSM. ID identifies a unique transition within the scope of the current FSM. HStateID, EventID and TStateID identify the head state, the event caused this transition and the tail state, respectively.

Operations defined in CHFSM initialize the FSM, including the primitive play events, from formatted text files, provide read or write access to the data encapsulated, and serve the description and analysis process.
4.1.2 Acquisition of the ATD and SPEs from video clips

We manually extract instances of three specific game situations (i.e., 2-on-1 offensive attack, power play shot from the point and defensive zone breakout) from hockey game video tapes as the source video clips. These clips are digitized into image sequences with the frame rate 30 frames per second. We use a semiautomatic tracking tool developed in the IRIS TRA project ([23], [39]) to track players' positions in the image frame, identify correspondence points between the rink features in the images and the features in the rink model, calculate the homography matrices and then register the players' positions to the rink frame. Hence we get the players' raw trajectory data. We also detect shots on goal, changes of possession of the puck, and each player's changes of skating mode (i.e., among skating backward, skating forward and falling down) in the image sequences and save these events in formatted text files (in future versions, they may be stored in database tables/views). Augmenting a player's raw trajectory data with information on possession of the puck and skating mode, we get the player's Augmented Trajectory Data.

From the list of changes of possession of the puck, we derive the play events PASS (if the current puck carrier is from the same team as the previous puck carrier) and LOSEPuckPossession (otherwise). We build the list of the puck carrier's crossing goal line, crossing blue line and crossing center line if it is meaningful in the analysis to the situation, which is decided by the FSM used in the analysis. Similarly, the PASS events could be further classified into pass between two players behind the goal line, pass between one player (current puck carrier) behind the goal line and the other player (receiver) inside the defensive zone, etc. These classifications are automatically done in the operation CClip::setPrimEvtList(), which builds the list of play events, sorted by their occurring times, according to the clip's situation (i.e.,
4.1.3 Textual FSMs for the selected game situations

Based on the fields defined in \textit{TEvent}, we built the formatted text file for primitive play events as follows. Words after "/" in the same line are comments.

\begin{verbatim}
// File: playevents.def
// Purpose: define play events and their evaluation functions
// Format:
//   Event: ID, Exp1 (Explanation 1, i.e., Action), Exp2, Roles
//   (optional) EEF: index, EEF Name, EEF Description
//   (optional) DefaultEEF: index

[Event]
a0 Pass "\%s passed the puck to his teammate \%s." (Puck Carrier, Receiver)
  0 EvPassV1 "Evaluate riskiness of passing the puck to other teammates, i.e., whether the pass will be defended by the defenders."
  DefaultEEF: 0

[Event]
a1 Shoot "\%s shot the puck toward the goal \%s." (Puck Carrier, Goal)
  0 EvShootV1 "Simply evaluate a shot on net as good."
  1 EvShootV2 "Take into account the distance, the shot angle and the players in front of the goal."
  DefaultEEF: 0

[Event]
a2 SkateForward "\%s started skating forward." (Player)

[Event]
a3 SkateBackward "\%s started skating backward." (Player)

[Event]
a4 FallDown "\%s fell down." (Player)

[Event]
a5 StandUp "\%s stood up." (Player)

[Event]
a6 CrossBlueLine "\%s crossed the blue line \%s." (Player, BlueLine) // in side A or B
\end{verbatim}
According to the IDs defined in the text file for primitive play events, we get the FSM depicted in Figure 4.8 for simple 2-on-1 offensive attacks, by relabelling the events in the FSM depicted in Figure 3.3 and refining the final states. The information the updated FSM represents is stored in the following formatted text file, which is provided as input.

Formatted text file for the FSM of simple 2-on-1 offensive attacks —
// File: tra2on1fsm.def
// Purpose: define the FSM for 2-on-1 offensive attack
// M = {A, Q, I, J, R};
// A: primitive events/actions; Q: states;
// I: initial states; J: final states;
// R: state transition map.

[Identification]
SituID: 1  // 0: unknown; 1: 2-on-1 offensive attack;
          // 2: power play shot from the point;
          // 3: defensive zone breakout
Version: 0.1.0
Perspective: 0  // 0: offensive; 1: defensive
Explanation: "Represent play knowledge under simple 2-on-1 offensive
attacks from the offensive perspective."

[Events] // ID only
a0 // Pass
a6 // CrossBlueLine
a12 // LosePuckPossession
a1_0 // Shoot (outside the blue line)
a1_1 // Shoot (inside the offensive zone)
a9_0 // EnterPlay (another offensive player entered)
a9_1 // EnterPlay (another defensive player entered)

[States] // ID, Exp, BFinal, Evf
q0 "The two attackers are outside their offensive blue line." 0 Null
q1 "The puck carrier is inside his offensive zone." 0 Null
q2 "The puck carrier lost possession of the puck." 1 Bad
q3 "The puck carrier shot towards the goal outside his offensive blue line." 1 Bad
q4 "Another offensive player entered the play." 1 Good
q5 "The puck carrier shot towards the goal in his offensive zone." 1 Good
q6 "Another defensive player entered the play." 1 Bad

[Transitions] // ID, HStateID, EventID, TStateID
r0 q0 a0 q0
r1 q0 a6 q1
r2 q0 a12 q2
r3 q0 a1_0 q3
r4 q0 a9_0 q4
r5 q0 a9_1 q6
r6 q1 a6 q0
r7 q1 a0 q1

Figure 4.8: The FSM for simple 2-on-1 offensive attacks
Similarly, Figure 4.9 and Figure 4.10 are the FSMs for 5-on-4 power play shots from the point and simple defensive zone breakouts respectively, followed by the formatted text files storing the information represented by these FSMs.

![FSM Diagram]

Figure 4.9: The FSM for 5-on-4 power play shots from the point

```
. Formatted text file for the FSM of 5-on-4 power play shots from the point
// File: trappfsm.def
// Purpose: define the FSM for power play shot from the point
// M = {A, Q, I, J, R};
//   A: primitive events/actions;   Q: states;
//   I: initial states;           J: final states;
//   R: state transition map.

[Identification]
SituID: 2 // 0: unknown; 1: 2-on-1 offensive attack;
          // 2: power play shot from the point;
          // 3: defensive zone breakout
Version: 0.1.0
Perspective: 0 // 0: offensive; 1: defensive
Explanation: "Represent play knowledge under 5-on-4 power play shots
from the point, from the offensive perspective."

[Events]   // ID only
a0_0      // Pass (to the other point player)
a0_1      // Pass (to a player not in the point)
```
\textbf{a1} \hspace{1em} // Shoot (from the point)
\textbf{a12} \hspace{1em} // LosePuckPossession

\textbf{[States]} \hspace{1em} // ID, Exp, BFinal, Evf
q0 "The point player has possession of the puck." 0 \hspace{1em} Null
q1 "The puck carrier passed the puck to a player not in the point." 1 \hspace{1em} Neutral
q2 "The puck carrier lost possession of the puck." 1 \hspace{1em} Bad
q3 "The puck carrier shot towards the goal from the point." 1 \hspace{1em} EvShootV2

\textbf{[Transitions]} \hspace{1em} // ID, HStateID, EventID, TStateID
r0 q0 a0_0 q0
r1 q0 a0_1 q1
r2 q0 a12 q2
r3 q0 a1 q3

\begin{figure}
\centering
\includegraphics[width=\textwidth]{fig4-10.jpg}
\caption{The FSM for simple defensive zone breakouts}
\end{figure}
[Identification]
SituID: 3 // 0: unknown; 1: 2-on-1 offensive attack;
// 2: power play shot from the point;
// 3: defensive zone breakout
Version: 0.1.0
Perspective: 0 // 0: offensive; 1: defensive
Explanation: "Represent play knowledge under simple defensive zone
breakouts, from the offensive perspective."

[Events] // ID only
a0_0 // Pass (between two players behind their goal line)
a0_1 // Pass (between one player behind his goal line and
// the other player in their defending zone)
a0_2 // Pass (between two players in their defending zone)
a0_3 // Pass (from one player in his defending zone to his
// teammate in the neutral zone)
a6 // CrossBlueLine
a8 // CrossGoalLine
a12 // LosePuckPossession

[States] // ID, Exp, BFinal, Evf
q0 "The puck carrier is behind his goal line." 0 Null
q1 "The player in his defending zone has possession of the puck."
  0 Null
q2 "The offensive team lost possession of the puck."
  1 Bad
q3 "The player in the neutral zone has possession of the puck."
  1 Good

[Transitions] // ID, HStateID, EventID, TStateID
r0 q0 a0_0 q0
r1 q0 a0_1 q1
r2 q0 a8 q1
r3 q0 a12 q2
r4 q1 a0_1 q0
r5 q1 a8 q0
r6 q1 a0_2 q1
r7 q1 a12 q2
r8 q1 a0_3 q3
r9 q1 a6 q3

End of file
4.2 Event evaluation functions

Evaluation functions for play events are used in the analysis process to judge whether alternative actions are feasible during specified time intervals. We define the following function type:

```c
typedef int (EvFunc)(CClip* pClip, int t1, int t2, void* pCookie);
```

for all the evaluation functions. It takes 4 arguments:

- `pClip` is a pointer to `CClip` through which information (such as ATD) on the clip in analysis is accessed;
- `t1` is the time (frame number) from which the evaluation starts;
- `t2` is the time (frame number) at which the evaluation ends;
- `pCookie` is a pointer to any custom data required or provided by the evaluation function;

and returns an integer score as the evaluation result. Currently we implement 4 evaluation functions, i.e., `EvPassV1`, `EvShootV1`, `EvShootV2` and `EvLosePuckPossV1`, for 3 play events: PASS, SHOOT and LOSEPUCKPOSSESSION.

`EvPassV1` evaluates the feasibility of passing the puck to some teammate during the interval 
\[ [t_1, t_2] \]. It takes into account the players’ current geometric configuration, the defensive players’ defending coverages and the potential receiver’s most likely next position after a short fixed time interval. In the version we implemented, the custom data include the puck carrier’s `PlayerID` ([input]), the potential receiver’s `PlayerID` ([input]) and the earliest time point \( t \) ([output]) at which passing is feasible. It uses the following algorithm to calculate the output:
1. Build a list of the puck carrier's opponent players; if none exists, set 
\( t = t_1 \) and return 1, which means it is feasible to pass at \( t_1 \). Here we 
ignore other factors which may affect the pass, such as it may be blocked 
by the referees or nets.

2. For each frame \( t' \in [t_1, t_2] \), estimate the possible passing paths and check 
whether they could be defended by opponent players. We first estimate 
the potential receiver's next position after some \( \delta_t \), say 0.5 seconds, ac­
cording to his previous trajectory, and take the area (a triangle or a line 
segment) formed by the receiver's current position point, his estimated 
next position point and the puck carrier's current position point as the 
estimated passing path(s). Assuming without being defended or inter­
fered, the puck carrier has the skills to pass the puck to the receiver 
along the estimated passing path(s) within the period \( \delta_t \). We define a 
player's defending coverage as a circle area with a 4-foot radius, centered 
at the player's current position. Though we could take into account the 
player's shoots, skating mode and other factors to refine his defending 
area, in current implementation, we ignore these details.

2.1. If the estimated passing path(s) do not intersect any defending player's 
coverage area, set \( t = t' \) and return 1, which means it is feasible to 
pass at frame \( t' \);

2.2. If \( t_1 = t_2 \) and the path(s) contain at least one defending player's 
coverage, set \( t = t_1 \) and return -1, which means it is unwise to pass 
at frame \( t_1 \);

2.3. If \( t_1 = t_2 \) and the path(s) intersect at least one of the defending 
players' coverages but do not contain any of them, set \( t = t_1 \) and
return 0, which means it is risky to pass at frame \( t_1 \);

2.4. otherwise continue to deal with next frame.

3. Set \( t = -1 \) and return \(-1\), which means it is unwise to pass the puck during the interval \([t_1, t_2]\).

\(EvShootV1\) takes any “shot towards the goal” as a good thing for the offensive team. It simply returns 1, suggesting that shot is feasible during the interval \([t_1, t_2]\).

\(EvShootV2\) further analyzes the “shot” by taking into account the distance \( d \) between the puck carrier and the center of the goal line, the shot angle \( \alpha \) (i.e., the acute angle between the goal line \( l_g \) and the line \( l_s \) decided by the puck carrier’s position and the center point of the goal line), and positions of both offensive and defensive players in front of the goal. The custom data include the puck carrier’s \( PlayerID \) ([input]), the goal to shoot \( GoalMark \) ([input], either side ‘A’ or side ‘B’) and the earliest feasible time \( t \in [t_1, t_2] \) ([output]) to shoot. It uses the following steps to decide the output.

1. Build a list \( (L_d) \) of defensive players excluding the goalie, and a list \( (L_o) \) of offensive players excluding the puck carrier;

2. Let \( t = t_1 \) and \( nRet = 1 \);

3. For each frame \( t' \in [t_1, t_2] \):

   3.1. let \( nRet' = 1 \);

   3.2. calculate the shot distance \( d \), if \( d < 50 \) (in feet), \( nRet' = nRet' + 1 \);

   3.3. calculate the shot angle \( \alpha \), if \( \alpha > \pi/4 \), \( nRet' = nRet' + 1 \);

   3.4. compute the in-front-of-net area \( S_3 \), i.e., the intersection of the quadrilateral \( S_1 \) (formed by the left face-off spot \( E \), the left goal post \( B \),
the right goal post C and the right face-off spot F) and the triangle $S_2$ (formed by the puck carrier $A$, the left goal post $B$ and the right goal post $C$), as depicted in Figure 4.11;

![Diagram](image)

Figure 4.11: Illustration for $EvShootV2$

$S_1$ is the quadrilateral $EBCF$, $S_2$ is the triangle $\triangle ABC$, and $S_3$ is the intersection of $S_1$ and $S_2$. The shot distance $d$ is the distance between the puck carrier $A$ and the goal center $D$. The shot angle $\alpha$ is the acute angle $\angle ADB$.

3.5. for all the defending players in the list $L_d$, if none of them is inside the area $S_3$, $nRet' = nRet' + 1$;

3.6. for all the offensive players in the list $L_\alpha$, if at least one of them is
inside the area $S_3$, $nRet' = nRet' + 1$;

3.7. If $nRet' == 5$, set $t = t'$ and return 5;

3.8. If $nRet' > nRet$, set $t = t'$ and $nRet = nRet'$;

4. Return $nRet$.

$EvLosePuckPossV1$ returns $-1$, suggesting it is bad or unwise for the offensive team to lose possession of the puck.

### 4.3 Configuration

The $Configure$ module provides users the interface (Figure 4.12) to dynamically assign general system options. For each of the built-in situations, users can choose which FSM to use in the analysis process and which evaluation function to call for each of the primitive play events. Users can also assign which description, analysis and prediction functions and their parameters (if any) to be used for the selected analysis strategy (analysis module). Options for the $Visualize$ module include whether to analyze and display in each frame the riskiness of passing the puck to the puck carrier's teammates and the feasibility extent of shooting the puck towards the goal.

### 4.4 Description, Analysis and Prediction

As described in Section 3.4.3, we implement an FSM based, puck carrier centered analysis strategy. Description is done by parsing the sorted list of primitive play events and generating the transition path bound with time information, according to the FSM selected for analyzing the clip's situation. Analysis identifies alternative paths in the FSM which may result in better outcomes than the final state of the path.
Figure 4.12: The Configure dialog
generated in the description step. Prediction perturbs the puck carrier's position and timing again to see if better outcomes might result at key play events.

The module for this strategy uses the structure `TClipSituDAC` and its wrapper class `CClipSituDACList` (Figure 4.13) to record the textual description, analysis and prediction generated by corresponding functions, i.e., `describeV1`, `analyzeV1` and `predictV1`. These three functions have the following function type:

```c
typedef int (SamFunc)(CClip* pClip, CHFSM* pFSM, char** ppPath);
```

where

- `pClip` is a pointer to `CClip` through which information on the clip such as the
sorted list of primitive play events is accessed;

- \textit{pFSM} is a pointer to \textit{CHFSM} which stores information about the selected FSM;

- \textit{ppPath} keeps the generated transition path(s) in the analysis process.

4.4.1 Description function

descrive\_V1 uses the following algorithm to generate the transition path and the description:

1. Set the FSM's initial state as the current/start state (\textit{pCSt}, we assume all the clips start from the FSM's initial state, though the start state could be any non-final state calculated from the players' augmented trajectory data);

2. For each item (\textit{pEvt}) in the clip's sorted list (\textit{pClip->mpPrimEvtList}) of primitive play events:

   2.1. If the current state is a final state, return;

   2.2. Instantiate the current event's explanation (\textit{Exp2}) and insert the result string together with the time information (\textit{Frame}) as one piece of description to that frame into the clip's description, analysis and prediction list (\textit{pClip->mpCSDACHead});

   2.3. If the FSM does not include a transition which takes the current state as the head state\textsuperscript{1} and the current event as the action, i.e., this event does not result in state change in current state, set both the current and next state.

\textsuperscript{1}If a transition starts from state \(q_h\) and ends up with state \(q_t\), we call \(q_h\) the transition's head state and \(q_t\) the tail state.
states of the current event \( (pEvt->mpPrimEvent->CSt, pEvt->mpPrimEvent->NSt) \) as the current state \( (pCSt) \);

2.4. Otherwise, set \( pEvt->mpPrimEvent->CSt \) and \( pEvt->mpPrimEvent->NSt \) as the transition's head state \( q_h \) and tail state \( q_t \) respectively; append this transition ID together with Frame number into the transition path \( pp-Path \); and set the current state \( pCSt \) as \( q_t \).

### 4.4.2 Analysis function

The `analyzeV1` function takes as input the time bound transition path \( (ppPath) \) generated by `describeV1`, identifies feasible alternative paths with better results in the FSM and inserts relevant textual analysis into the clip's description, analysis and prediction list \( (pClip->mpCSDACHead) \). It has the following major steps:

1. Copy the actual path \( ppPath \) to a temporary work path \( tmpPath \);

2. Check the outcome of \( tmpPath \) by calling the evaluation function or retrieving the constant evaluation result defined for the final state in that path. If the return value or the constant value, called as \( nActualResult \), is less than 0, which means the actual path's outcome is bad, then set \( ppPath \) as NULL;

3. Set current state \( pCSt \) as the final state in \( tmpPath \) and mark it as visited;

4. Trace one transition backward from the end of \( tmpPath \). If no transition is available, return; otherwise, set current state \( pCSt \) as the head state of this transition, extract the time information (i.e., the Frame number in which this transition occurred) enclosed in the transition path, and trim this transition from \( tmpPath \);
5. Get all the transitions which have the current state $pCSt$ as their head states. If all these transitions' tail states have been visited, mark current state $pCSt$ as visited and go to Step 4; otherwise put those transitions in a set $R$;

6. For each transition $r_i \in R$ whose tail state $q_i$ has not been visited:

   6.1. If the evaluation function attached to $r_i$'s action $a_i$ has been defined, call it with the input parameter $t_1 = Frame$ and $t_2$ being the clip's last frame number; if the function returns a value greater than 0 and stores a valid frame $t$ in $pCookie$, which means the action $a_i$ is feasible at time $t \in [t_1, t_2]$, then append $r_i$ together with $t$ to $tmpPath$;

   6.2. If no evaluation function is assigned for $r_i$'s action $a_i$, append $r_i$ together with $Frame$ to $tmpPath$ directly;

   6.3. If $r_i$ has been appended to $tmpPath$, call $pFSM->searchPath()$ to retrieve subpaths (transition sequences) which start from $q_i$ and end up with final states whose evaluation results are not worse than the actual outcome, i.e., their evaluation functions return values not less than $nActualResult$; for each of these subpaths $subPath$, form an alternative path $aPath$ by concatenating $tmpPath$ with $subPath$, append $aPath$ to $ppPath$; instantiate $aPath$ by expanding its actions' $Exp2$ fields and insert the expanded $aPath$ as pieces of analysis into the clip's description, analysis and prediction list ($pClip->mpCSDACHHead$);

7. Set the current state $pCSt$ as visited and go back to Step 4.

### 4.4.3 Prediction function

`predictV1` is highly related with the underlying FSM. We have not developed a gen-
eral algorithm (see discussion of prediction in Chapter 6) to do the prediction in various game situations. In this version, we only provide several predefined perturbations to deal with the situation 2-on-1 offensive attacking. The basic idea is to perturb the two attackers' positions so that both passing the puck between them and shooting towards the goal are likely to be more successful. While generating perturbations, we also take into account some heuristic constraints such as the player's maximum speed, current speed, etc. After applying these perturbations to the players' trajectory data, we redo the analysis.

The Describe module calls the description function describeV1, the Analyze module calls the analysis function analyzeV1, and the Predict module calls the prediction function predictV1 to update the clip's description, analysis and prediction list (pClip->mpCSDACHead).

4.5 Visualization

The Visualize module provides the user interface and shows relevant information stored in the class CClip pointed by pClip, as depicted in Figure 4.14. The upper left window displays the clip's image sequence frame by frame and the upper middle window displays the clip's textual description, analysis and prediction list. The upper right window contains several play control buttons and the bottom window visualizes frame by frame (synchronized with the image window) the augmented trajectory data and the riskiness of the puck carrier's passing and/or shooting if the corresponding options are selected by the user.

The Visualize module does not do additional analysis of its own. It just visualizes the current description, analysis or prediction result frame by frame. If the event/action evaluation function used in the current analysis process returns
Figure 4.14: Visualizing information stored in pClip
a value indicating it is feasible for the offensive or defensive player(s) to take that event/action, then the relevant area (e.g., the estimated passing/shooting paths) will be filled in with green horizontal lines. If the return value indicates it is risky to take that event/action, the relevant area will be filled in with yellow diagonal crossing lines. Red vertical lines will be filled in the relevant area if the return value indicates it is unwise to take that event/action.

In this chapter we presented a simple implementation. The focus is to make the framework flexible and extensible. Accordingly, the hockey accuracy of the implemented Event Evaluation Functions (EEFs) and the implemented Finite State Machines (FSMs) are not the major concerns at this point. Instead, the implemented functions and FSMs serve as template to demonstrate what is possible. Given a flexible and extensible framework, one can easily imagine that EEFs and FSMs supersede the ones given in terms of their hockey accuracy and sophistication.
Chapter 5

Results

We present analyses of three more clips (beyond the one presented in section 1.3). The video clips and the animated analysis results are available through: http://www.cs.ubc.ca/nest/lci/thesis/fhli/videoclips.

5.1 Instances of 2-on-1 offensive attacking

Figure 5.2 and Figure 5.3 are from the output of analysis and prediction applied to clip2, an instance of 2-on-1 offensive attack. In the clip, the situation starts at Frame #1 (Figure 5.1(a)) after the attacker on the right (Player #4) receives a pass from a teammate in the neutral zone. Another attacker (Player #5) is on the left and in front of Player #4. Only one opposite defenceman (Player #6) faces these two attackers. In Frame #74 (Figure 5.1(c)), the puck carrier (Player #4) crosses the blue line, followed by the other attacker (Player #5) immediately. At Frame #124 (Figure 5.1(e)), the puck carrier (Player #4) passes the puck to Player #5 and the latter shoots the puck towards the goal at Frame #144 (Figure 5.1(f)) and the offensive team scores.
Figure 5.1: Key image frames in clip2

(a): the situation starts but the defenceman is invisible in the video; (b): the defenceman shows up in the video since Frame #14; (c): the puck carrier crosses the blue line; (d): the puck carrier could have shot sooner since this frame; (e): the puck carrier passes the puck to a teammate; (f): the puck carrier shoots towards the goal.
Major textual descriptions of the clip include:

At Frame #74, Puck carrier (#4) crossed the blue line.

At Frame #124, Puck carrier (#4) passed the puck to his teammate (#5).

At Frame #144, The puck carrier (#5) shot the puck towards the goal.

The clip goes through the path:

1. q0[Frame #1]: The two attackers are outside their offensive blue line.
2. q1[Frame #74]: The puck carrier is inside his offensive zone.
3. q1[Frame #124]: The puck carrier is inside his offensive zone.
4. q5[Frame #144]: The puck carrier shot towards the goal in his offensive zone.

The outcome is good for the offensive team.

Major textual analyses include:

According to the selected FSM and the return values of evaluation functions, i.e.,

Since Frame #43, EvPassV1() returns:0.
Since Frame #44, EvPassV1() returns:1.
Since Frame #79, EvShootV2() returns:2.
Since Frame #96, EvShootV2() returns:3.
Since Frame #124, EvShootV2() returns:2.
Since Frame #126, EvShootV2() returns:3.
Since Frame #128, EvPassV1() returns:0.
Since Frame #132, EvPassV1() returns:1.
Since Frame #133, EvPassV1() returns:0.
Since Frame #136, EvPassV1() returns:-1.
Since Frame #137, EvPassV1() returns:0.
Since Frame #143, EvPassV1() returns:1.
Figure 5.2: Analysis of clip2, an instance of 2-on-1 offensive attack
the offensive team could have played the following path for a better outcome:

1. \( q_0[\text{Frame } #1] \): The two attackers are outside their offensive blue line.
2. \( q_1[\text{Frame } #74] \): The puck carrier is inside his offensive zone.
3. \( q_5[\text{Frame } #96--123] \): The puck carrier shot towards the goal in his offensive zone.

It tells that Player #4 could have shot sooner since Frame #96 (Figure 5.1(d)) instead of passing the puck to Player #5 at Frame #124 (Figure 5.1(e)) in order to get a more favorable outcome.

Figure 5.3: Applying one of the predefined trajectory perturbations to the analysis in Figure 5.2

The predefined perturbation does change the effectiveness of shooting and
passing, i.e., the return values of their evaluation functions,

Since Frame #40, PosPert=3, EvPassV1() returns:0.
Since Frame #44, PosPert=3, EvPassV1() returns:1.
Since Frame #79, PosPert=3, EvShootV2() returns:2.
Since Frame #100, PosPert=3, EvShootV2() returns:3.
Since Frame #107, PosPert=3, EvShootV2() returns:2.
Since Frame #112, PosPert=3, EvShootV2() returns:3.
Since Frame #121, PosPert=3, EvShootV2() returns:4.
Since Frame #124, PosPert=3, EvShootV2() returns:3.
Since Frame #127, PosPert=3, EvPassV1() returns:0.
Since Frame #135, PosPert=3, EvPassV1() returns:-1.
Since Frame #136, PosPert=3, EvPassV1() returns:0.
Since Frame #143, PosPert=3, EvPassV1() returns:1.

as we can see from the difference between Figure 5.2 and Figure 5.3.

5.2 Instances of power play shooting from the point

Figure 5.5 is from the output of analysis applied to clip3, an instance of power play shot from the point. In this clip, the situation starts from one point player (Player #8) has possession of the puck at Frame #1 (Figure 5.4(a)) in a 5-on-4 power play formation. The puck carrier (Player #8) passes the puck to the other point player (Player #7) at Frame #90 (Figure 5.4(d)). The situation ends when the puck carrier (Player #7) shoots the puck towards the goal at Frame #181 (Figure 5.4(f)).

Major textual descriptions of the clip include:

At Frame #90, Puck carrier (#8) passed the puck to his teammate (#7).
At Frame #181, The puck carrier (#7) shot towards the goal in his offensive zone.
(a): the situation starts; (b), (c), (e): frames where the return value of `EvShootV2()` changes; (d): the puck carrier passes the puck to a teammate; (f): the puck carrier shoots towards the goal.
Figure 5.5: Analysis of clip3, an instance of power play shot from the point
The clip goes through the path:

1. q0[Frame #1]: The point player has possession of the puck.
2. q0[Frame #90]: The point player has possession of the puck.
3. q3[Frame #181]: The puck carrier shot towards the goal from the point.

The outcome is dependent on EvShootV2().

Major textual analyses include:

According to the selected FSM and the return values of evaluation functions, i.e.,

Since Frame #37, EvShootV2() returns: 3.
Since Frame #41, EvShootV2() returns: 2.
Since Frame #177, EvShootV2() returns: 1.
Since Frame #181, EvShootV2() returns: 2.

the offensive team could have played the following path for a better outcome:

1. q0[Frame #1]: The point player has possession of the puck.
2. q3[Frame #37–40]: The puck carrier shot towards the goal from the point.

It tells that the point player (Player #8) could have shot since Frame #37 (Figure 5.4(b)) instead of passing the puck to Player #7 at Frame #90 (Figure 5.4(d)) in order to get a more favorable outcome.

No perturbation/prediction is applied to this clip.

5.3 Instances of defensive zone breakout

Figure 5.7 is from the output of analysis applied to clip11, an instance of defensive zone breakout. In this clip, the situation starts after Player #1101 behind the net has possession of the puck at Frame #38 (Figure 5.6(a)). At Frame #46 (Figure 5.6(b)),

91
he passes the puck to a teammate (Player #1105) in their defending zone. Player #1105 then passes the puck to his teammate Player #1104 in their defending zone at Frame #126 (Figure 5.6(h)). The situation ends after Player #1104 carries the puck and crosses the blue line into the neutral zone at Frame #132 (Figure 5.6(i)).

![Figure 5.6: Key image frames in clip11](image)

(a) Frame #38  (b) Frame #46  (c) Frame #72  
(d) Frame #85  (e) Frame #98  (f) Frame #105  
(g) Frame #112  (h) Frame #126  (i) Frame #132

Figure 5.6: Key image frames in clip11

(a): the situation starts; (b),(h): the puck carrier passes the puck to a teammate; (c),(d),(e): two offensive players cross over; (f),(g): pass process; (i): the puck carrier crosses his defensive blue line.

Major textual descriptions to the clip include:

At Frame #46, Puck carrier (#1101) passed the puck to his teammate (#1105).
Figure 5.7: Analysis of clip11, an instance of defensive zone breakout
At Frame #126, Puck carrier (#1105) passed the puck to his teammate (#1104).

At Frame #132, Puck carrier (#1104) crossed the blue line.

The clip goes through the path:

1. \(q_0\) [Frame #38]: The puck carrier is behind his goal line.

2. \(q_1\) [Frame #46]: The player in his defending zone has possession of the puck.

3. \(q_1\) [Frame #126]: The player in his defending zone has possession of the puck.

4. \(q_3\) [Frame #132]: The player in the neutral zone has possession of the puck.

The outcome is good for the offensive team.

No evaluation function or perturbation/prediction is applied to this clip.
Chapter 6

Conclusion, discussion and future work

6.1 Conclusion

We design a flexible and extensible framework to describe, analyze and predict player actions in selected game situations, based on Augmented Trajectory Data, Sorted Play Events, Event Evaluation Functions, and Finite State Machines which represent play knowledge in those situations. The results show that augmented player trajectory data support description, analysis and prediction in hockey game situations. Although our FSM model does not represent the complete range of feasible geometric arrangements of players in each situation, it is adequate to represent specific game situations and the discrete events/actions that occur in those situations. With event evaluation functions, we can identify alternative paths in the FSM which might result in better outcomes than the actual one if the players played the situation by following those paths. Analysis of alternative geometric positioning of the players can predict more advantageous outcomes for a given path through the FSM.
This analysis is limited in depth in that we do not predict how players on the other team might react to the alternative positioning proposed. The framework provides a foundation for development of more elaborated hockey analysis applications.

6.2 Discussion

6.2.1 Acquiring input data

It proved difficult for us to extract accurate players' trajectories from video clips, given that our clips are from broadcast TV and we cannot control the cameras that record the games. When the tracking tool [23] developed in the project could not find enough rink features in key image frames and thus could not automatically register these images to the globally consistent rink model, we manually identified necessary corresponding points between the original image and the rink model. Based on the geometric properties that exist among those identified correspondences, we extrapolate additional useful correspondences (if possible) which may be beyond the original image borders. For instance, if in the original image, we can only see part of the center line $l_c$ and part of the left border line $l_b$ and thus miss their intersection point $P$, and we have identified 4 correspondence pairs, i.e., $A, B$ in $l_c$ and $C, D$ in $l_b$ corresponding to $A_m, B_m, C_m$ and $D_m$ in the rink model, then we can compute another correspondence pair $P$ and $P_m$ even though $P$ is invisible in the original image.

We also manually recognize play events that occur in the clips, since we have not developed an event recognizer which can determine the higher-level play events of interest automatically from video clips. Play events are sorted, stored in a formatted text file and included in the input. Based on this sorted play events list,
the framework could have identified the start and the end of the three given game situations automatically from video clips; however, for simplification we manually identify the three game situations’ start and end as well.

When defining event evaluation functions, we make assumptions according to general cases, without considering players’ particular skills. We have data structures to store players’ basic profiles including attributes such as shoots left, shoots right, maximum speed, etc. It is feasible to build a database storing players’ individual skating, defending and/or attacking skills in various game situations to enhance the system. This would help to make more accurate evaluations of events/actions taken by an individual player or a specific group of players. As a specific example, in current implementation, we do not take into account the difference between two players’ defending or attacking coverages caused by the difference between their handedness.

Different FSMs can be defined for any given game situation. This is desirable since description, analysis and prediction in specific game situations depend on many factors including the skill level of the players and the particular coaching style and strategies employed. Each FSM is one view of a real game situation. It is not expected that analysis based on a single FSM will be relevant to all parties. At the same time, users can easily build their own FSMs so that a given video clip can be analyzed from various perspectives and at different levels of granularity.

6.2.2 Prediction

The basic idea of making prediction is to find a better spatio-temporal configuration for the offensive (or defensive, whichever is the analysis perspective) players to complete their understood goals, under some heuristic constraints such as maintaining
offensive players' current speed though they could change their directions. In an ideal case, i.e., if the evaluation functions are continuous, we can find an improved configuration by estimating the spatio-temporal gradients of these functions. The difficulty lies in that we do not know what play formations the players are trying to achieve and how they would react to opponent players' interference. Nevertheless, we can perturb the puck carrier's (or one of the key players') current position to somewhere nearby that can be reached by the player after a short time interval $\delta_t$, given her or his maximum skating speed and meanwhile obeying the physical laws; assuming other players keep still or moving in their own previous velocities, now we get a new players' geometric configuration after $\delta_t$ and we can call the evaluation functions to redo the prediction. How to decide the perturbations? One possible solution is to introduce uncertainty and use statistical methods to decide the most likely positions the player will move to based on the player's history of play behavior. Another one is to allow the user to adjust players' positions interactively and show the evaluation results on the fly in the 2D graphic animation. In current implementation, we just define several fixed perturbation modes and each of them perturbs key players' positions according to heuristic factors which play a role in computing the evaluation functions.

6.3 Future work

First, we need to improve the accuracy of trajectory data. We need to improve the methods used in tracking hockey players and in solving the homography matrices between key image frames and the globally consistent rink model. We might consider smoothing the raw trajectory data in both spatial and temporal dimensions. In addition, we could synthesize players' trajectory data by treating players
as intelligent artificial lives [27] who are collaborating and/or competing together to achieve understood goals.

Second, as other members in the project [39] are designing higher-level event recognizers, we would like to integrate these recognizers into the framework and develop more powerful/realistic event evaluation functions in the future. Improved trajectory data and event evaluation functions will make the analysis results more persuasive and more helpful.

Third, the Predict module needs general algorithms to do the prediction in various game situations, as discussed in Section 6.2.2. A program which can predict a player's next action on the fly will interest more people.

Fourth, we would also like to develop more analysis strategies such as using multiple FSMs in parallel to deal with complicated game situations. Statecharts [9] may be introduced to supersede FSMs when the multitude of states grows exponentially and/or when the notions of hierarchy, concurrency and communication among game situations need to be dealt with to a much higher level of granularity.

Last but not the least, on one hand, we want to enhance the current user interface by various means, such as providing user-friendly tools for users to input, debug and summarize hockey domain knowledge, illustrating analysis/prediction results on image frames which are sequentially displayed in the video window, and using advanced 3D graphic animation in the animation window; on the other hand, we would like to recombine key modules in the framework to create a program without graphic user interface to deal with massive input data and output the description, analysis and prediction results into a database.
Bibliography


