Indexing and Behaviour Modelling of Team Sports

By

Andrew Hume

Submitted in accordance with the requirements
for the degree of Doctor of Philosophy.

The University of Leeds
School of Computing

March, 2012

Abstract

With the steady reduction in the price of storage, and increasing availability of high quality recording devices, much effort has been invested in investigating methods to index large collections of high dimensional datasets. Archives of sporting events are well represented within this set of large datasets. Most efforts to index sport related data have concentrated on the indexing of collections of audio/video data. This thesis presents and evaluates several novel methods to index football matches based on the underlying trajectory of the ball and players, rather than the raw video. This allows for the potential of very expressive indexing systems. The second strand of this thesis explores the use of the underlying trajectory data to build behavioural models of players. A promising hierarchical approach is undertaken, whereby the behaviour of individual players is influenced by the cliques of players they associate with, as well as the team as a whole. Although both the indexing and behavioural modelling aspects of this thesis use data from football as the basis for the work, in principle the approaches taken are general enough to apply to any team based game.

Acknowledgements

I would like to thank the Leeds Vision Group, and particularly my supervisor Derek Magee for many illuminating discussions. Thanks also go to Phil Tordoff and Dan Mason as my contacts within ProZone for their assistance with data procurement. Finally thank you to my parents John and Judith Hume for their continued support over this time. Sleep well Dad.

Contents

Figures

Tables

Glossary

- **Autoregression** : A predictive process which uses past states of a modelled system to predict the next state.
- **Clique** : A completely connected sub-section of an undirected graph.
- **CP (Chebyshev Polynomials)** : A sequence of orthogonal polynomials which can be used to approximate/compress arbitrary functions or data time series.
- **Indexing system** : A system which enables navigation/searching within a (usually large) dataset by generating a mapping from a compressed representation of the data to its uncompressed form. Google search is one example of indexing for Internet sites, the indexing systems presented within this thesis are another example in the domain of football.
- **MAM (motion activity maps)** : 2D histograms which record the spatial extent of motion activity over a predefined period of time.
- **MAS (Multi Agent System)** : A system composed of multiple interacting agents, in which each agent is aware of and reactive to (a subset of) the other agents in the system. The degree to which each agent is aware of the other agents is a function of its behavioural sophistication : it may view them as simply non-static elements of the environment and attempt to learn their behaviour, it may communicate with other agents to negotiate behaviour or it may hold internal 'mental' models of other agents and use this to anticipate behaviour.
- **MBB (minimum bounding box)** : Given an n-dimensional space containing m points, it is the smallest n-dimensional hypervolume which can contain all m points. Reduces to a rectangular parallelepiped and a rectangle in 3 and 2 dimensions respectively.
- **MDL (minimum description length)** : A model selection process which posits that the best models to select are those which are the most parsimonious.
- **NN (Neural Network)** : Computational model inspired by biological neural networks. Able to learn any arbitrary function (given enough internal connectivity).
- **OLS (Orthogonal Least Squares) :** An algorithm for performing linear regression on a dataset.

- **OLS pruning :** An algorithm for reducing the number of connections within a neural network with the goal of making it better generalise learnt functions and also remove redundant input data. Uses OLS to measure the variance of node activity, the idea being that more variance equates with useful information (and no or little variance equates with redundant information).
- **PCA (Principal Component Analysis)** : A mathematical operation on an n-dimensional dataset to project it into an m-dimensional space (where $m < n$) using an orthogonal linear transformation which aims to preserve as much of the variation in the original dataset as possible in the m principal components.
- **PI (Partition Index)** : The portion of the indexing system that allows very efficient division of the indexed data during queries via levering database index techniques.
- **SOP (Segment Of Play)** : A short play sequence from a football match which cover all players on the pitch
- **SOPREF (Segment Of Play Reference)** : A 3-tuple reference to a SOP consisting of an ID for the match, half, and time within the half during which the SOP starts.
- **RM (Ranking Metadata)** : The portion of the indexing system that allows the results returned during the PI section of the query to be ranked w.r.t. similarity with the query.
- **SQL (Structured Query Language)** : The standard language used to define and program modern day relational databases.
- **Vector quantization** : a lossy data compression method which generates prototypical vectors (in a codebook) from an underlying dataset, the prototype vectors being representative of the distribution of data within the underlying dataset.
- **Voronoi diagram** : A decomposition of a metric space containing a number of generator points into a collection of cells, such that each cell represents the volume closest to each generator point.

1 Introduction

1.1 Aims and Motivation

With the steady reduction in the price of storage, and increasing availability of high quality recording devices, many professional football clubs have taken to augmenting coaching sessions with digital video match footage coupled with detailed coverage of the statistical and movement characteristics of the players. The club itself may supply this information, but more often, an outside company supplies it.

One such company is the Leeds based ProZone [1]. They supply client clubs with multi-angle digital match videos, and corresponding collections of player trajectories and events transcribed from the videos. The trajectory/events collections allow coaches an in-depth view of an individual player's performance during a match; both from a fitness and a tactical/skill perspective (see Figure 1.1). Currently this information is packaged up in the form of one or more DVDs, and then used by the coaching staff to identify problems from the previous game and help prepare for the upcoming match. Outside this time window around the current match, historical information concerning players, particular the trajectory and event information is rarely used. The information is kept by ProZone however, and they currently have an archive containing at least five years worth of trajectory and event information for premiership clubs/players.

The aim of this thesis is to see if this archival data can be used in any interesting and novel fashions. Specifically two problem domains will be looked at; that of indexing the archival data with a view to making it easily queriable, and that of using the archival player data as a basis to build behavioural models of the players. As ProZone is a CASE research partner for this thesis, one last aim (or perhaps constraint) of this thesis is that any work developed should be at least in some sense practically realisable (by which is meant implementable with reasonable computing resources).

Figure 1.1 – Examining the trajectory of a single player over a segment of play using the Prozone3 software.

1.2 The Problem domains

At the most abstract level, this thesis examines whether trajectory data obtained from tracking players involved in a team game can be used to index the tracked games and examines if it is possible to build player models from analysis of the player trajectories.

1.2.1 Indexing

Two problem domains are covered in this thesis. The first is that of indexing football matches, specifically indexing football matches via short segments of play (SOP). Many interesting scenarios that occur during a football game can be characterised by a surprisingly short SOPs (see Figure 1.2). Obvious examples are corners and free kicks, but there are also less obvious free moving examples such as signature passing moves, or a critical possession loss resulting in attack breakdowns.

Indexing involves describing a collection of objects in a compressed form, then allowing similarity searches to be performed over the compressed collection. The ability to index archived games using the SOPs contained within them would

allow coaching staff a powerful method of consulting the past when making decisions about current tactical situations (i.e. how successful have team X been in the past when attempting a free kick using player Y?)

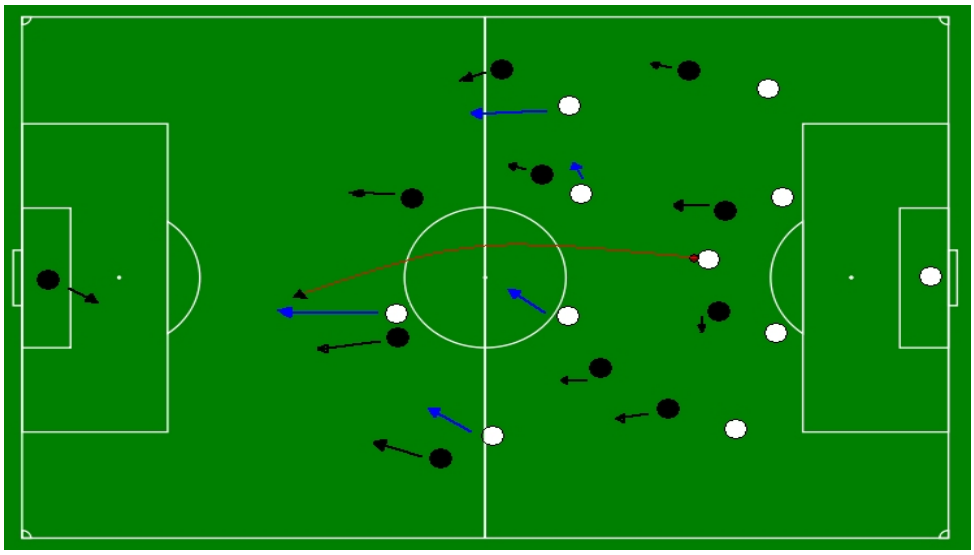

Figure 1.2 – SOP depicting longball attack tactic by the white team and associated off the ball movement of players during the segment.

1.2.2 Behaviour modelling

The second problem domain is behavioural modelling. The overall aim is to attempt to discover systematic patterns of movement that will allow general models of player movement to be realised. There are at least two potential scenarios that such a collection of models could be used for. The first is simulation of hypothetical game situations, which would obviously be a boon for coaches. Modelling of players allows for clustering of players who exhibit similar behaviour together; which would be very useful for scouts and the transfer market (i.e. replacing an injured player with a player with a similar behaviour model).

1.2.3 Wider applicability

In principle, the techniques explored in this thesis could be applied to any other sport that also satisfies the following constraints:

(1) Two distinct teams of identifiably unique players.
(2) Two-dimensional bounded playing surface[1].
(3) Playing ball (or puck or similar).
(4) The ability to track/record player trajectories.

Accordingly, games such as Rugby, American Football and Hockey could all be fruitful areas to reapply the research contained within this thesis.

1.3 Thesis overview

- **Chapter 2** – Firstly provides general background information concerning ProZone and the nature of the data they provide and its procurement, then reviews previous work in the fields of indexing and behavioural modelling.

- **Chapter 3** – Presents a general framework for indexing SOPs, and then several novel indexing approaches are introduced; either based on the trajectory of players or the trajectory of the ball.

- **Chapter 4** – Work covering the aggregation of query results from two different indexing systems is presented. A formal experiment is undertaken to compare the proposed indexing approaches, as well as result aggregation, by eliciting subjective ratings for query results from human experimental subjects.

[1] A number of the indexing schemes presented in this thesis implicitly impose a further restriction of requiring the playing surface to be rectangular, as they project rectangular overlays onto the playing surface during indexing (see sections 3.6, 3.7, 3.8, 3.9 for details).

- **Chapter 5** – A novel approach to behaviour modelling of players is presented, player behaviour being modelled in a hierarchical fashion, from individual player, to cliques of players associated by proximity/movement direction, to the overall team behaviour. The resultant models are evaluated, firstly against a random walk model w.r.t. predicting the next time step movement (0.1s), and then over an extended simulation period of 25s against a random walk model, and a linear predictor model. The results of the evaluations demonstrate the potential usefulness of the models produced.

- **Chapter 6** – Presents the conclusions of the thesis and highlights possible futures avenues of research.

2 Background and previous work

This thesis covers two separate areas; that of indexing and behaviour modelling. To reflect this, this chapter is divided into two main sections covering previous research on indexing and behaviour modelling respectively. Preceding the two main sections are background information on ProZone, the source of the trajectory and event data used in this thesis, and a brief primer on databases.

2.1 Prozone

ProZone [1] is a Leeds based company that provides client football clubs with the ability to record and analyse their matches both in video form, and in the form of annotated player trajectories/events (see Figure 2.1). The fusion of match video with trajectory / event information allows a more comprehensive post match analysis to be performed than would be the case with video alone. Clubs are able to collect detailed statistics about player performance during the match and to perform rudimentary indexing of the video at the level of events (i.e. find all goals or corners in a game). Deeper analysis of player performance is also possible by collating and comparing player statistics over a number of matches, enabling trends in player performance to be studied.

Given that this thesis will use collections of the trajectory and event data both in the indexing and behavioural modelling research, it is prudent to describe how the data is procured, its fidelity/accuracy, and how the data is structured. The video data is captured from a collection of eight static cameras strategically situated around the football pitch, such that no area of the pitch is left unobserved (as opposed to normal TV coverage that naturally follows the movement of the ball). Player trajectory data is generated by manual operators tasked with inspecting the raw video, identifying individual players[2] and tracking their movements throughout the game by marking

[2] Each player is assigned a unique immutable player ID, which is used in any match they are tracked in.

their positions on the raw video. The raw video screen coordinates of players are transformed into ground plane positions on the pitch, and coupled with unique player ids form the trajectory dataset for the match. The trajectory transcription process has been shown to exhibit a high correlation with independent trajectory measurements using timing gates [2] for both long player trajectories and shorter sprint trajectories[3].

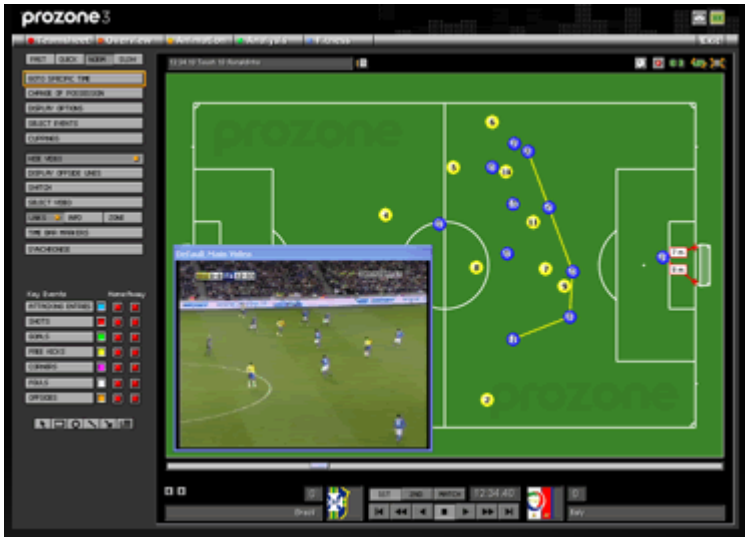

Figure 2.1 – ProZone Match viewer with synchronised match video

Events are recorded by the manual operators as and when they occur with the particular player being tracked. The type of event, the player(s) involved, and the event location are all recorded (a complete list of valid events is available in section 7.2). It should be noted that the manual transcription of all trajectories/events is a very labour intensive task; typically, one operator is only assigned one player per match to track. To achieve timely processing of all captured matches, a large team of manual operators is required. ProZone is attempting to semi-automate some of the trajectory tracking by the use of vision-based player tracking techniques [3], but as of September 2009, this technology is still under testing/review.

[3] For 60m & 50m trajectories correlation of ProZone tracking with ground truth is: {r=0.999, total error=0.05, limits of agreement=0.23}, for 15m sprints correlation is : {r=0.970, total error=0.23, limits of agreement=0.85}

All trajectories and events are recorded to a temporal fidelity of 0.1 seconds; however, trajectories are not recorded continuously in a temporal sense. Gaps exist in the trajectory data because player positions are only recorded when the player being tracked exhibits significant movement on the pitch, and this position is updated with a frequency of approximately 0.5-1.0 seconds (depending on player motion). Thus in the raw state it is not guaranteed that for a particular time instant all the players' positions will be immediately available, so the missing player positions must be interpolated if required. The trajectory of the ball is not recorded at all during the transcription process. If required it can be interpolated from events (such as ball touches) and player position.

2.2 Indexing review

The following section initially presents on overview of indexing, or to give it its more general title information retrieval [8]. There then follows two subsections which review indexing research carried out at two different levels of analysis; starting with context based indexing then moving onto content based indexing.

2.2.1 Basics of indexing

A centuries old example of indexing is available in the back pages of virtually all academic textbooks. In this index section, a list of subjects is displayed together with associated page numbers on which the subjects are referenced. The index is ordered alphabetically, thus allowing the reader very fast access to relevant subject pages. This is an early version of what is now technically known as an Inverted Index [9].

The subjects available in the index are chosen manually by the author/editor of the book, and represent key concepts or features that the book addresses. During compilation of the index, each page of the book is analysed for occurrences of the predefined features, and their presence is incrementally added into the index. This is the first vital component of any indexing system; namely, the ability to analyse a set of objects (book pages in this example) and to decompose each object into a collection of features which represent it in a more abstract/compressed sense.

Subject	Page
Agent	1,4,6,9
Emergence	2,3,9
Entropy	4,5,6
Indexing	7,8
Scalability	9

Table 2.1 – Example inverted index associating subjects with the pages they are found within

Table 2.1 shows an example of an inverted index from a very small notional book, with five subjects of interest. The obvious function that this index provides is to quickly locate relevant pages covering subject matter of interest to the reader. In other words, the reader of the book is able to perform queries using the index and the predefined set of subjects. This is the second vital component of any indexing system, namely the ability for queries to be constructed from the set of index features and then applied to the index to return a set of matching results (in this case page numbers, but in general they will be references to the indexed objects). A less obvious function that the index provides becomes clear if the list of pages numbers attributed to each subject are treated as sets. Then more complex queries can be performed using set algebra, such as:

$$(Agents \cup Entropy) \cap Scalability^{C} \tag{2.1}$$

Equation (2.1) asks what set of pages reference either $Agents$ or $Entropy$, but do not cover $Scalability$? The results of this query, which is the set {1,4,5,6}, attaches equal importance to each page returned. However, the terms $Agents$ and $Entropy$ occur simultaneously on pages 4 and 6, and only appear alone on pages 1 and 5. Since the reader is interested in both terms, pages 4 and 6 are likely to be more important to the reader than pages 1 and 5. If instead of sets, the list of page numbers are viewed as bags, then the result of the query becomes {1,4,4,5,6,6}. Now pages 4 and 6 are elevated above pages 1 and 5 by virtue of their greater representation in the results bag. This is the third and final vital component of any indexing system; namely, the ability to rank results of index queries in terms of

relevance to the query itself. Therefore, in summary, the basic requirements of any indexing system are:

(1) Given a set of features (or the ability to generate a set), be able to analyse a set of objects and decompose them into a compressed form representing the features present, and map the compressed representation to the original object.
(2) The ability to pose suitably structured queries using the features, and then resolve the query by matching against the index.
(3) The ability to rank results from a query in order of relevance to it.

These basic principles will help frame the following discussions covering indexing research.

2.2.2 Context indexing

This review of indexing research is structured in such a way that as the review continues approaches will be examined which analyse indexed objects at deeper levels than those in previous sections. However to begin with, approaches which do not analyse the objects to be indexed at all will be reviewed. These approaches treat the objects to be indexed as atomic, and instead examine the context in which the objects are situated and use this as the basis for indexing.

Perhaps the best-known example of this approach is Google image search [10]. Images embedded within web pages have image filenames, hypertext links and hypertext metadata associated with them. This contextual information associated with the image is used by Google as a surrogate object to be indexed in place of the more complex image. The surrogate object is parsed to check for occurrences of words in the Google lexicon (built up from previously encountered words on web bot cached pages), and those which are present are added into an inverted index which references the image instead of the surrogate object. Since the image is now indexed as a collection of lexicon entries, it can be queried by those same lexicon entries, enabling users to search for images simply by entering keywords. Ranking of returned image query results is performed as a combination of how many matches

the search terms made against the index and the pagerank of the page in which the image is embedded. The pagerank algorithm [11], promotes pages which are 'popular' (pages with many links to them, or pages with links from other popular pages), and its general form is shown in equation (2.2). The total collection of page ranks for all pages indexed forms a probability distribution.

$$P(u) = \sum_{v \in B_u} \frac{P(v)}{L(v)} \tag{2.2}$$

Where:

$P(u)$ is the page rank of page u (expressed as a probability),

B_u is the set of pages which link to page u,

$P(v)$ is the page rank of a particular page v which links to page u,

$L(v)$ is the total number of links which page v contains

Whilst this is a simple and elegant query interface, it does produce variable results. As an example, the search terms 'crab fishing north sea' returned the image in Figure 2.2 as the #2 search result – an image that is not visually relevant to the search terms used. Examination of the web page[4] in which the image is embedded reveals that there is a hypertext link to a story entitled 'Crab fishing in the North Sea is a dangerous profession', which presumably has been included in the context information for the image. This highlights the major drawback of using context information that is not checked for consistency, namely, there is no way of knowing if the context that surrounds the object forms a representative description.

[4] http://planetsave.com/category/war-conflict/page/2/ image reproduced under the Creative Commons licence.

Figure 2.2 – 'crab fishing north sea' image query result from Google image search

2.2.2.1 Manually entered labels

One approach to overcoming this weakness would be to explicitly solicit relevant image labels from humans, and then use these labels as the basis of the context on which to index the images. This is the approach taken in [12], where a short online game is proposed that shows a series of randomly selected images to teams of two players. Each player is asked to enter words they think the other player is also entering, and since the only shared experience the users have is being shown the image (players cannot see each other's entered labels), the labels will likely describe some aspect of it. Once players agree on a label, the next image is shown. Evaluation of the labels produced show that they perform well as the basis for a context based image index, and the labels received high subjective relevancy ratings from experimental participants. The conclusion that this game with a pool of 5000 players could label all of the images indexed by Google in a matter of weeks at first seems to be massively overoptimistic. Interestingly however, Google made public an almost identical game, the Google image labeller[5], from 2006 until 2011.

2.2.2.2 Attached annotations/captions

Within a more restricted domain however, using manual labelling of images to form the basis of an indexing system is a much more viable approach. In [13]

[5] http://images.google.com/imagelabeler/

annotations attached to crime scene photographs by the police are first analysed using parts of speech tagging [14] to assign words into lexical categories. The marked up annotation is then further analysed to extract relational triples representing additional semantic information between objects in the annotation. These relational triples are more discriminating than the underlying words in the annotation, and are used to form the index. A particularly gruesome example from the paper is the extracted relation triple 'Blood Around Body'. Searching the annotations for occurrences of the terms blood and body will match against photographs containing either, as well as photographs where the blood is not surrounding the body, whereas searching for the triple 'Blood Around Body' will only return photographs were indeed the blood surrounds the body. The use of relations between objects also allows a richer description to be produced (as there are potentially $n^2 - n$ relations between n objects assuming asymmetric relations).

Guglielmo [15] adopts a similar approach, using text captions associated with military photographs as the basis for indexing a collection of over 100,000 photographs. The indexing is implemented in two phases : a coarse phase, involving indexing purely on the nouns and verbs in the caption (based on a domain specific lexicon), and then a fine phase which involves involving indexing on logical form records, which capture the semantic sense of the caption in a collection of case grammar records [16]. Queries are also broken down into the two phases, and matching first occurs with the coarse phase, then results from this are matched against the fine phase. Evaluation of the system, using the precision and recall metrics (see equations (2.3) and (2.4)), showed a 30% improvement in precision and a 50% improvement in recall over a baseline system simply indexing on keywords. The major drawback of the system is that the use of a limited domain lexicon means that free form text queries must often be restated in the specific lexicon in order to generate acceptable results, implying that the system requires some training/experience to use successfully.

$$P = \frac{|R_e \cap R_r|}{R_r} \tag{2.3}$$

$$R = \frac{|R_e \cap R_r|}{R_e} \tag{2.4}$$

Where:

P = Precision metric,

R = Recall metric,

R_e = Set of relevant (to query) documents which exist,

R_r = Set of retrieved (by query) documents

ANVIL [17] is another image indexing system that uses short attached captions in order to index the images. Again, natural language parsing techniques are used to analyse the caption to extract dependency structures, which are tree-like structures representing the relationships between words (for instance that adjectives depend on the nouns that they describe). Textual queries are similarly decomposed into dependency structures, being matched against the dependency structures of captions that contain at least one of the keywords in the query. The dependency structure matching gives a ranking to the result of the query. The advantages and disadvantages of ANVIL both spring from the fact that it uses very short captions (mean length 9 words). The main advantage is that a more comprehensive analysis of the text is possible in a reasonable time (i.e. dependency structures). The main disadvantage is that using such short captions reduces the discriminatory power of the indexing system.

2.2.3 Content based indexing

Content Based Information Retrieval systems (CBIR) go directly to the heart of the matter, and use the data contained in the object itself to extract indexing information. The most well established area in this field is CBIR as applied to text documents. The World Wide Web (WWW) constitutes the largest collection of documents that has ever existed [18]. Internet search engines such as Google, Bing, Yahoo, Ask etc, all attempt to index this vast collection, and the primary means by which this is achieved is by viewing web pages as text documents, and indexing them based on the textual information they contain. Google [10] analyses web pages

for occurrences of keywords from its internal lexicon (which is built up by word extraction from previously indexed pages), and records keyword occurrences using a full inverted index, which is an augmentation of the standard inverted index in that it permits the position of words in a document to be recorded as well as their presence.

The use of a full inverted index allows the word order of a query to be taken into account, thereby enabling phrase queries and well as simple keyword queries. Internally the full inverted matrix is also used to partially contribute rank to results by examining the proximity of query keywords in the indexed documents. The final rank of a query result is a combination of how well the keywords were represented in the web page, how closely they were grouped together, and how popular the web page is (as denoted by the Pagerank algorithm [11]). The full inverted index is at the heart of all current commercial Internet search engines, and is arguably the most used indexing method for text documents.

An alternative to the (full) inverted index is the vector space model (VSM) [19] (also known as the bag of words model). In this model, the document is represented as a vector, where each component of the vector represents the presence (non-zero), or absence (zero) of a particular keyword (or even a phrase as in [20]). The length of the vector is set to the size of the lexicon, so each document is described in a fixed length format. At its most naïve, VSM will simply assign a 1 or 0 to each component to denote presence/absence of keywords, but more often the values assigned to those keywords which do appear represent the level of importance the word has in the document. This is the case with the term frequency inverse document frequency model (TF-IDF [21]), which uses the following equation (2.5) to estimate component values:

$$v_t = f_t . log\left(\frac{D}{d_t}\right) \tag{2.5}$$

Where :

v_t = t^{th} component of the vector v

f_t = Frequency of term t in the current document

D = total number of documents

d_t = number of documents containing the term t

The TF-IDF correctly gives a weight of 0.0 to terms which do not occur in the current document, and modifies the raw frequency of the terms which are present in the document by how widespread the term is in the document set (more coverage = greater importance). As with all indexing systems, queries are broken down in the same fashion as the indexed objects, and in the case of VSM it is simpler to use a document as a query template, analysing this, and using the resultant VSM as the query. This is a query by example, and is the more usual form of querying used when either the underlying object is complex or as in this case the raw query form can be complex/cumbersome (which a long vector would be). Ranking of queries is given by the angle between the query vector and each of the results vectors.

Latent Semantic Indexing (LSI) [22] is based on a similar idea to that of VSM, in that it uses fixed length vectors to denote the occurrence of keywords. Instead of treating each resultant vector as an independent entity, all vectors produced are gathered into matrix form (the term document), with rows representing keywords and columns representing documents. The matrix is decomposed into three matrices (term, concept and document) via a rank reducing variation of Singular Value Decomposition (SVD) (see equation (2.6)). The reduced rank chosen typically is very much smaller than that of the original matrix (reduction to 100-500 is usual).

$$X = T * C * D^T \tag{2.6}$$

Where :

$\boldsymbol{X}$ = m_xn term document (m=number of keywords, n=number of documents)

$\boldsymbol{T}$ = m_xr term matrix (r=reduced rank)

$\boldsymbol{C}$ = r_xr concept matrix

$\boldsymbol{D}$ = n_xr document matrix (the transpose $\boldsymbol{D^T}$ is r_xn)

SVD decomposition highlights hidden/latent semantic concepts (stored in the singular value matrix) which connect keywords to documents. The index is formed primarily of the document matrix, with the term and concept matrix as necessary ancillary structures. Queries by example (a vector) are initially transformed into a pseudo-document *P* via equation (2.7):

$$P = Q^T * T * C^{-1} \quad (2.7)$$

Where:

P=PseudoDocument

Q^T=Query vector (transposed)

T=term matrix

C^{-1}=concept matrix (inverse)

Ranked results of indexing similarity are produced by applying a suitable similarity metric (Euclidean distance, angle between vectors etc) to the pseudo-document and each row vector in the document matrix.

2.2.3.1 Images

Some approaches to the indexing of images were examined in the section on context indexing, but the data content of the image itself provides the richest and most accurate description of it. Content Based Image Retrieval (CBIMR) aims to use this data to provide accurate indexing information that does not require additional external cues. The vast majority of CBIMR uses the query by example model.

Many approaches to CBIMR use global statistical properties of the image as an index. One popular property is colour. One of the earliest approaches by Swaine *et al*. [23] involves decomposing images into histograms of the opponent colour space. Query images are similarly decomposed, and a direct comparison between histograms is made via histogram intersection. Indexes on features-vectors produced from HSV colour space histograms extracted from images are utilised in [24]. Queries are ranked by the cosine of the angle between the query feature-vector and indexed feature-vectors. A similar approach is taken by [25] where a structure named a Correlogram distils the spatial correlation of colours within the image; ranking is via the L_1 distance (Manhattan distance) between the query and indexed Correlograms. A variation on the colour histogram is presented in [26], where the

HVC colour space[6] and a HVC distance metric are used to cluster the colours within an image together to form an effective index.

Another useful second order global image feature is that of texture. Liu *et al*. [27] use a 2D Wold decomposition[7] to extract three orthogonal measures of texture from images: harmonic, evanescent, and indeterministic, which form the basis of the index feature vector for the image. Highly structured query images (those which posses a high harmonic rating due to repeated patterns) are compared purely using harmonic peak matching as an efficiency measure. All other query images are compared using the evanescent / indeterministic components. Rotation and scale invariant Gabor filters (achieved by using collections of Gabor filters at differing orientations and scales) are used in [28] to extract robust texture feature vectors from images that are used as the index. The distance between query and index feature vectors is used for ranking results.

2.2.3.2 Local image features

An alternative approach to CBIMR uses collections of local features extracted from the image as a means to classify/index the image. A progression from the purely textual context image indexing systems [29] uses a collection of manually entered semantic image annotations together with a hierarchical multi-resolution decomposition of the image to model image categories using two-dimensional multi-resolution Hidden Markov Models (2DMHMM). Once fitted to the training data, these 2DMHMMs can be used to find the best fitting collection of semantic index annotations for arbitrary images presented to them. This system has the potential to bridge the semantic gap between low level image features and high level semantic concepts, which will allow users to query for abstract image attributes (such as 'blue sky'), rather than keywords or example images. Developing a bottom up approach to

---

[6] The colour space which represents colours along the human colour perceptual dimensions

[7] A Wold decomposition is a method of decomposing time-series data into uncorrelated components.

categorisation/indexing, [30] extracts Harris affine regions[8] [31] from the image, transforms then into SIFT descriptors[9] [32], assigns these descriptors to feature clusters via vector quantisation (forming keypoints), then constructs a feature vector composed of the number of features assigned to each cluster (this is known as a bag-of-keypoints). The bag-of-keypoints can then be used to determine what predetermined category the image is best described/indexed by. [33] adopts a similar approach in that it breaks images down into low-level keypoint features; however, it adds hierarchical levels of structure above the lowest keypoint level, with probabilistic spatial relations linking higher parts of the structure to the lower level elements. Hierarchical structuring of spatial relations is also a key feature of [34], which uses quadtrees to partition the image, and form the basis of the indexing. Queries are made more efficient by the ability of the representation to filter out unsuitable matches quickly by matching at the coarsest levels of the hierarchy first.

2.2.3.3 Music

Music is another complex and ubiquitous data source that would benefit from reliable indexing. Music is often manually categorised in terms of style of music, and the work in [35] automatically assigns styles to collections of audio. The method involves extracting 6-second clips from each audio source, creating a time-invariant representation of it by Fourier transforms, and then allowing a self-organising-map (SOM) [36] to cluster the collection of time invariant representations. The clusters allow the music to be broadly indexed, and for new music to be categorised, although the categories do not have any clear semantic meaning. [37] Uses a polyphonic multi-pitch detector to record all the notes in a piece of music into a pitch-histogram, which is then transformed into a small feature vector which records the frequency of the most dominant folded[10] pitch, the time-coverage of that dominant folded pitch, the most dominant raw pitch, and the histogram distance between the most dominant and second most dominant pitch. Manually categorised

---

[8] Images regions which are invariant w.r.t. translation, scaling, and rotation; are partially invariant to illumination changes and robust to local geometric distortion.

[9] Scale-Invariant Feature Transform

[10] Folded here meaning all notes are transformed to lie within one octave

music (5 categories) is similarly decomposed into feature vectors and used as cluster seeds, and then k-nearest-neighbour classification can assigned unseen music into its closest category. Musipedia [38] is a website that offers a query-by-humming service, music is indexed by contour strings [39], which record only the change in pitch in a monophonic melody, disregarding time/tempo. User queries (in the form of humming, or midi files) are similarly decomposed into contour strings and matched against music in the index by the edit distance between the two strings.

2.2.3.4 Video

Video offers another challenging dimension to indexing, that of time (since video is essentially a temporal sequence of images), and multi-modality (accompanying audio tracks, (tele)text, captions etc). Temporal segmentation/indexing of video is a well researched area, most techniques using some variation of scene change[11] detection such as using inter frame difference metrics to detect abrupt scene changes and block based motion analysis to detect gradual fades/dissolves [40,41]. The temporal segmentation allows a video to be decomposed into a sequential number of segments and then representative key frames are selected for each segment and indexed into an image database[12].

More sophisticated approaches make use of the temporal evolution of the video. [42] adopts a unimodal approach, indexing based on the global motion present with video. It produces motion activity maps (MAM), which are basically 2D histograms (see Figure 2.3) which capture the magnitude and spatial distribution of motion within a video clip by summing then quantising the motion vector fields over a series of video frames. This allows video content to be indexed over the spatial distribution of motion, which is useful in applications such a video surveillance which are typically only interested in activity in specific regions (such as doors, windows).

---

[11] such as fades, dissolves, abrupt scene changes etc

[12] The author is not aware of any instance of videos being indexed on their entire global content, at least not from a content analysis angle.

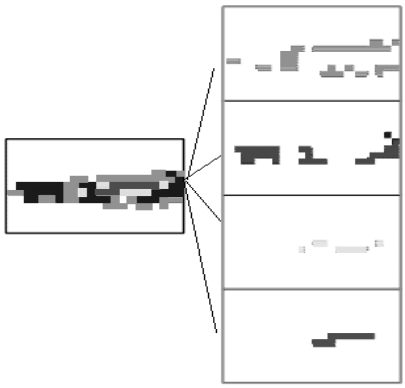

Figure 2.3 – Differing areas of motion activity captured using a motion activity map

The approach seems similar to that of Motion History Images (MHI) [43], applied to video indexing rather than visual movement classification, the main difference being that MAMs do not have any notion of direction of movement, recording only magnitude and spatial information. In the paper, the MAMs are only used as a means of interactive video navigation, rather than a full blown information retrieval model, but using the MAMs as index seems perfectly feasible, as demonstrated by the other CBIR models covered here which are histogram based.

As with images, video indexing can make use of collections of local features as well as global characteristics. [44] indexes half-second video clips by first segmenting out pixel regions of consistent motion content and colour from the 3D pixel space produced by stacking the half-second of video images. From the consistent regions, 7D spatio-temporal descriptors are produced which summarise the motion and colour of each region. Thus, each half-second clip is decomposed into a variable number of these 7D descriptors, and this set is manually labelled with the originating video clip. Queries also take the form of half-second video clips, are similarly transformed into a set of 7D descriptors, and then compared to the indexed video clips using k-NN to find the video(s) which are most similar to the query.

[45] adopts a multi-modal approach to the indexing of association football videos, taking into account video, audio and textual information. Interesting events to look for are predefined using fuzzy Allen Time Interval Relations [46], which is a propositional logic like language allowing binary temporal statements to be made (such as X precedes Y). This enables events to be composed of features from different modalities that do not occur simultaneously in time. The features

themselves are extracted from text (detection and video optical character recognition of closed captions), video (type of camera work, face detection, close-ups,) and audio (excitement level of commentator). Given the events rules and analysed video modalities, the video can then be indexed on successful firing of the event rules. This system forms the basis of the association football video search engine, Goalgle, which allows users to quickly search association football videos based on the predefined event rules (such as goal, yellow card, substitution).

2.2.3.5 Trajectories

Perhaps of more direct relevance to the data available to this thesis are approaches based on trajectory data. Chen [47] uses wavelets to smooth and segment trajectories into component parts (the parts being sections of the trajectory during which acceleration does not change quickly). Each sub-trajectory is transformed into a feature vector recording the acceleration, initial velocity, and the trajectory shape; each sub-trajectory feature vector being indexed as belonging to the parent trajectory. Query trajectories are similarly decomposed into parts, and the index is searched for sub-trajectories which match each part, the Mahalanobis metric being used for similarity rating. The results are consolidated and ranked by the number and quality of sub-part matches.

Extracting trajectories from video of street surveillance, the work described in [48] normalises and splits the trajectories into their separate x and y components (essentially time-series). The components are decomposed using Haar wavelets, and the first eight coefficients of each component projection form the feature vector for indexing. Query trajectories are similarly decomposed, and ranking is achieved by the Euclidean distance between feature vectors. [49] Presents a novel way to index spatio-temporal trajectories using Chebyshev Polynomials(CP) [50][13], which are a set of orthogonal polynomials, defined recursively as (see Equations (2.8) (2.9) (2.10)):

[13] Specifically Chebyshev polynomials of the first kind.

$$T_0(x) = 1 \tag{2.8}$$

$$T_1(x) = x \tag{2.9}$$

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x) \tag{2.10}$$

The first six Chebyshev polynomials are shown in Figure 2.4.

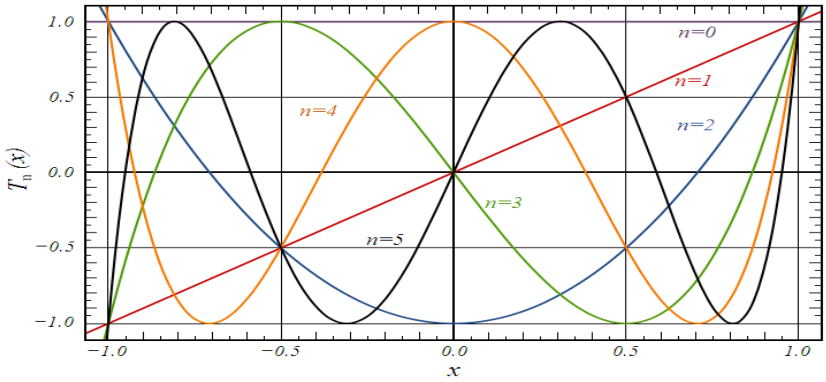

Figure 2.4 – The first six Chebyshev polynomials over the interval [-1,1]

One use of CPs is for function approximation [51], where an arbitrary data time series or function can be approximated by the summation of the first N Chebyshev polynomials, multiplied by Chebyshev coefficients $[a_0, \dots, a_n]$ (Equation (2.11)):

$$p(x) = \sum_{n=0}^{N} a_n T_n(x) \tag{2.11}$$

Numerical methods exist which can calculate the optimal fit for the coefficients in order to best approximate an arbitrary time series of data (the paper uses the Gauss-Chebyshev quadrature). The paper uses this function approximation to create approximate versions of trajectories, a relatively low number of coefficients provides a good trajectory approximation. All trajectories are converted into CPs, and the coefficients form the index for the trajectories. Queries on trajectories take the form of a target trajectory, which itself is converted into CP

coefficient form, and then a similarity metric (see Equations (2.12), (2.13), (2.14)) is used to calculate the distance between pairs of CPs (one being the query CP, the other CP index entries). With the specific metric presented, similarity in CPs corresponds to similarity in the original trajectory.

$$\vec{C}_1 = [a_0, \dots, a_m] \tag{2.12}$$

$$\vec{C}_2 = [b_0, \dots, b_m] \tag{2.13}$$

$$D(\vec{C}_1, \vec{C}_2) = \sqrt{\frac{\pi}{2}\sum_{i=0}^{m}(a_i - b_i)^2} \tag{2.14}$$

Where:

$\vec{C}_1$=query-vector

$\vec{C}_2$=index-vector

$D(\vec{C}_1, \vec{C}_2)$=distance between the query-vector and the index vector

The work presented in [52] breaks down trajectories into subparts, segmenting based on the sharpness of the 2D curve of the trajectory, normalising and resampling the sub-trajectories (so that they all have the same size), and storing them in a matrix. PCA [53] is applied to the matrix, and the most significant components form a feature matrix. Two variations of indexing are presented, in the first query trajectories are similarly decomposed into a feature matrix, and matching/ranking is achieved via summing the minimum Euclidean distances between sub-trajectories PCA coefficients of the query and indexed trajectories. This appears to be very computationally expensive. The second variation uses spectral clustering [54] to assign categories to the indexed sub-trajectories, clusters are assigned a letter/code, and the trajectory is described as string. Query trajectories are also decomposed and

transformed into strings, and matching/ranking is perform via the Edit distance metric for strings [55].

Whilst the previous approaches only index individual trajectories, [56] is able to index collections of trajectories simultaneously. Multiple trajectories are represented as a tensor[14]. A method of tensor decomposition known as parallel factor analysis (PARAFAC) [57] decomposes the tensor into the three loading vectors. The tensor is then multiplied by the two most significant vectors to produce a matrix of coefficients. This matrix (plus the two vectors as metadata) forms the index for the multiple trajectories. Multiple trajectory queries, also represented as tensors, are multiplied by each indexed tensors loading vectors to produce a coefficient matrix. Ranking of results is via the Euclidean distance between the matrices.

A related field to that of pure trajectory indexing is the study of moving object databases (MOD), where the spatial coordinates of collections of entities (such as vehicles, mobile phones and other location aware devices) are continually logged to form extended spatio-temporal trajectories. Queries are likewise spatio-temporal in nature, such as “how many objects were within range R of location X at time T”. MODs are typically very large in terms of data volume, and as such efficient means of indexing their spatio-temporal content are required. Early approaches used the R-Tree [7] and its variants [58,59] to spatially partition the data into collections of MBBs, each one covering an objects complete spatial distribution over its lifetime (see section 7.8 for a brief overview of database use and terminology). This partitioning allows the database to efficiently preselect a subset of the data that is spatially relevant to the query, before more expensive (distance) similarity operations are performed on the subset of data. One problem with this early approach is that it does not take into account time; long lived objects may have large MBBs (as they have had time to move significantly) and as such considerable overlaps could exist between object MBBs, severely reducing the effectiveness of the data partitioning.

Later approaches incorporated the time dimension explicitly into the indexing; the multi-version R-tree (MVR) [60] introduced a series of time-interval labelled

---

[14] which is a generalisation of the vector (1-tensor) and matrix(2-tensor)

MBBs which evolve over the lifetime of a tracked object, each individual MBB being more compact than the equivalent MBB encompassing the entire history of the object, thus allowing more discrimination partitioning (when a time interval is explicitly queried at least). [61] improves on the basic MVR approach by proposing a number of greedy algorithm heuristics which can break down a single MBB covering the entire lifetime of an object into a collection of k smaller variable time-length MBBs which approximately minimise the volume covered (see Figure 2.5).

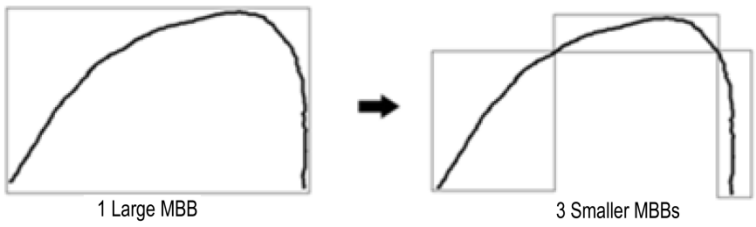

Figure 2.5 – minimising MBB volume coverage by using a collection of smaller MBBs to cover the trajectory rather than one large MBB.

Moving objects such as trains and cars do not exhibit unconstrained movement, rather they move within predefined spatial networks. This constraint can be used to further boost the efficiency of the indexing, as in [62] where the positions of trains and cars are indexed in a two-stage fashion. A MBB is constructed around the route taken by the tracked entity in the first stage, allowing an optimal MBB to be constructed, as the spatial layout of the network is known *a priori*. Since travel along a route is essentially movement along a line, the progress along the route can simply by represented by a point on the interval [0,1] (where 0=beginning of route, 1=end of route), and can be easily indexed by a 1D R-Tree (essentially performing straight line segmentation).

2.3 Behaviour modelling review

A good dictionary definition of behaviour [63] is as follows:

"The actions or reactions of a person or animal in response to external or internal stimuli."

The first step in modelling behaviour is to choose a paradigm in which the various approaches to behaviour modelling can be viewed. The obvious paradigm to choose is that which views the entities that are to be modelled as agents [64]. Just as Object Orientation [65] was an important paradigm shift in software engineering, allowing modelling to be centred around clusters of data exhibiting strong cohesion together, so the Agent based approach is a natural fit for behaviour modelling, allowing modelling to be centred around entities which cluster cohesive behaviour [66]. As a useful modelling approach, Agent usage became popular from the 1990s onwards in such fields as economic theory [67,68], artificial life research [69,70] and network optimisation [71,72].

There are many dimensions in which a review of behaviour modelling from an agent perspective could be taken. How the behaviour is learnt (Hand crafted, induced from data, learnt online), how the behaviour is represented (Production Rule System, State Machine, Hidden Markov Model, Neural Network etc), whether the modelling involves a single agent or can be considered a multi-agent system. For the purposes of this thesis, the first approach of how the behaviour is learnt will be taken.

2.3.1 Hand crafted behaviour models

This thesis defines handcrafted behaviour models as those in which either:

(1) The rules of behaviour for the system being modelled have been explicitly defined.
(2) The rules of behaviour for the system being modelled have been implicitly defined by explicitly defining fitness functions for this behaviour (and allowing the actual rules to be generated to fit this ideal model).

The Agent paradigm has actually been active since the late 1940s when John von Neumann introduced the idea of the von Neumann machine [73], which was a notional machine capable of reproducing itself. The idea was developed further in concert with Stanisław Ulam, to produce the idea of Cellular Automata, where agents are fixed within an n-dimensional grid only seeing and being able to affect

automata in neighbouring cells. The classic example of Cellular Automata is John Conway's Game of Life [74] in which extremely simple pre-programmed cellular automata are nonetheless able to produce complex emergent patterns on a 2D grid.

Rodney Brooks, an opponent of the symbolic computation direction that artificial intelligence took in the 1970-80's was at the forefront of a push to investigate biologically inspired agent architectures [75,76] (his agents were physical robots). His central thesis was that true intelligence should be built from the ground up, emerging from the interactions of simple pre-programmed behavioural modules, which were modelled as Augmented Finite State Machines (AFSM). The separate AFSMs were accommodated within what Brooks termed a Subsumption Architecture [77], which is a bottom up behavioural architecture where the modules at the lowest level (say obstacle avoidance) can influence modules at higher levels of the architecture (say robot movement), but not vice-versa. The behaviour of the whole robot is due to the emergent behaviour caused by the interactions of the behaviour modules.

Another example of complex emergent behaviour resulting from interactions of simple behavioural agents is the work of Reynolds [78], who uses a Multi-Agent System (MAS) of very simple agents in order to simulate flocking in birds (and by extension other creatures which also move in a mass emergent pattern such as fish). Each agent (a boid) has simple, predefined behaviour, captured as ranked set of production rules. These rules capture the boids urge to join the flock alongside its aversion to colliding with other members of the flock. In decreasing order of importance, the flock rules are:

- **Collision Avoidance**: avoid collisions with nearby flockmates
- **Velocity Matching**: attempt to match velocity with nearby flockmates
- **Flock Centring**: attempt to stay close to nearby flockmates

The boids are given only a limited perception of their environment, only able to see local neighbouring boids. However even with these simple behavioural rules and limited environmental knowledge, the interactions that occur between the boids allows complex behaviour to emerge from these interactions. A few things to note

about the boids. Firstly they are stateless, i.e. they have no memory of the past, the simply react to their environment. In the case of the boids the spatial environment in which they are embedded is empty/isotropic, so what they are really reacting to is the perception of other nearby agents; in effect the other agents are non-stationary environmental features. The degree to which a particular agent within a MAS is aware of the other agents is addressed in [79], which lists three levels of agent awareness, which are in order of increasing complexity:

- **Level 1** – Agents are not explicitly aware of other agents, but instead view them as (non-stationary) parts of the environment. This implies that level 1 agents respond to other agents via environmental sensing.
- **Level 2** – Agents are explicitly aware of each other and can interact directly together via exchange of messages.
- **Level 3** – Agents are explicitly aware of each other and possess the ability to generate internal models of other agents which they can use to predict agent behaviour.

Under this classification, boids are clearly level 1 agents. Social insects are a rich area of research for agents at level 1, and are grouped under the general term Swarm Intelligence (SI) [80]. The behaviour of ants within colonies is the archetypal example of SI, and central to any model of ant behaviour is the concept of Stigmergy [81], which is the indirect communication between agents via the environment (in the case of ants it is via pheromones). Stigmergy allows collections of ants to discover surprisingly optimal routes to and from food sources, even though the ants themselves possess only a limited repertoire of behaviour (which can be easily be modelled as a Probabilistic Finite State Machine). Ant Colony Optimisation (ACO) is the area of research which abstracts the behaviour of ants and directs it to solving general optimisation problems as diverse as scheduling the movement of partially built cars in an assembly line [82], discovering rule based classifiers during data mining [83] and performing binary thresholding on images [84].

Sport is an area where behaviour modelling is of great interest, as a tool for analysis as well as what if scenarios and team selection. American Football is in

many ways a much simpler game than its English namesake, and this relative simplicity allowed [85] to identify a finite set of American football plays, which were hand encoded using a Temporal Structure Description (TSD). These TSD encodings describe the permitted temporal behaviour of the set of players involved in any particular play. This has the advantage that the behaviour modelled is explicitly multi-agent. The main disadvantage obviously is that they are hand encoded, which would be intractable for the domain of learning association football behaviour from actual data. If a method could be discovered which could automatically generate the TSDs, then this would be a large step forward in team behavioural modelling.

Robocup is a yearly competition that pits teams of robot footballers against each other. As this competition involves many research areas, such as object tracking/identification and robotics, many of the robotic agents either utilise hand crafted behaviour models [86,87], or models generated via evolutionary techniques [88,89].

Dee [90] builds behaviour models based on the perceived intentions/goals of tracked entities traversing a known scene. Two variations of this principal are presented; the first produces a static behaviour model based upon psychologically valid models of human navigation. These high-level behaviour models have the advantage that they are quite robust to change because of their generality. Another advantage is that the goals/rules produced by the behaviour model are readily understandable by humans (which is often not the case with purely statistical models without the use of visualisation techniques). The second produces an ephemeral behaviour model of the tracked entity, by continuously updating the conjectured intentions of the entity being tracked as it moves. Being dynamic in nature this variation has the advantage of being able to adapt to new behaviour exhibited by tracked entities. This second variation is actually an example of learning behaviour from data, and so leads into the next review section.

2.3.2 Behaviour models learnt from data

This thesis defines “induced from data behaviour models” as those in which there is an available body of behavioural data. From analysis of this data, actual behaviour models can be built, and the resultant models should be a good fit to the

actual behaviour displayed within the data. In other words, the behaviour model(s) are used to build an accurate simulation of the system under study.

Statistical approaches have been used successfully within the vision community to model behaviour. Johnson [91] uses Vector Quantisation (VQ) techniques coupled with a Neural Network (NN) to statistically model the typical trajectories of pedestrians crossing a car park. VQ is first used to generate a set of state prototypes, which represent the most statistically significant object locations and associated velocities at these locations. These state prototypes are then used in the training of a NN, with a specialised dissipative or 'leaky' layer of neurons that is able to encode the temporal nature of trajectory. Further VQ on the output of the NN generates a set of trajectory prototypes, capturing the most likely behaviour of pedestrians. These prototypes can then be used either in a recognition role to spot usual/unusual behaviour, or in a generative role to provide realistic pedestrian trajectories. Later Johnson describes experiments in modelling behaviour between objects, firstly using the statistical co-occurrence of events between objects, and then modelling the joint behaviour of interacting objects is explored. Both of the techniques as stated are limited to the binary interactions.

Markov Models, Hidden Markov Models (HMM) and their variations are often used to model temporal behaviour / sequences of events from data. Galata [92] uses one variation on the HMM called the Variable Length Markov Model (VLMM) to represent sequences of atomic behaviours. The atomic behaviours are discovered from the data by searching for minima in the velocity of the object being modelled, the atomic behaviours identified as existing within the interval between two time adjacent velocity minima. More complex behaviour can be modelled by again using VLMMs to sequence together VLMMs containing only atomic behaviours. This hierarchical modelling approach allows richer object behaviour to be modelled.

Magee [93] uses another variation of the Hidden Markov Model called a Multi Stream Cyclic Hidden Markov Model (MSCHMM) to model the movement of cows' legs, primarily in order to automatically detect lameness in cattle. The MSCHMM is trained on cycles of archetypal cow leg movements automatically extracted from video of cattle walking past a static camera. Once trained, the MSCHMM can be used to predict cattle leg movements given a particular initial pose.

Brogan [94] involves analysing data archived from Robocup football matches [95] in order to model realistic agent behaviour over the team as a whole from logs of previous Robocup matches. He simplifies the data by generating Presence Density Maps (PDM) for each team, at each time interval. These PDMs are 2D Gaussian global representations of player locations at each instance. Behaviour modelling is achieved by predicting future PDMs from current PDMs, the future PDMs indicate the areas of greatest density which the agents should gravitate towards in order to replicate the observed behaviour in the original game. Even very simple agents could exhibit realistic movement behaviour given this scheme, assuming that the correct PDMs are predicted.

In Kaminka *et al.* [96], logs of previously played Robocup matches are analysed using a domain-dependent logic to generalise actions, creating time series of atomic actions from the raw data. These time series are then further analysed to identify repeated sub-sequences of actions. The facility to reject sequences that are frequent, but due to random co-occurrence rather than deliberate action, is also demonstrated.

At first predicting the next sequence in a time series may not seem to be a example of behaviour modelling, but many time series do represent human behaviour either in abstract form such as stock indices (arrived at by the mass participation of human traders), or trajectories represented as one or more time series. [97] looks at the use of Support Vector Machines (SVM) and NNs to model time series in an non-linear autoregressive manner (that is using the past information of the time series as a means to predicting its future value), drawing the conclusion that both SVMs and NNs can provide robust accuracy of prediction when used in this manner. The use of autoregression to model the movement behaviour of a housefly is covered in [98], where a linear autoregressive approach is taken and was able to identify two distinct components of the fly behaviour, that of velocity and angular motion which appeared to be independently controllable.

2.3.3 Behaviour models learnt experientially

This thesis defines experientially learnt behaviour models as those in which the participants begin with no behaviour model, but learn behaviour models by observation/interaction with the environment and/or other agents.

Reinforcement learning is a classic online modelling method. In overview it involves the agent viewing its environment as being in one of a finite number of states, each state has one or more actions associated with it that will cause the agent to transition into a new state. Each state also has a scalar reward associated with it, so moving from one state to another can result in gaining or losing rewards. The ultimate aim is to learn the optimal policy for the set of states, that is the actions at each state that will maximise reward for the agent. The collection of states, actions and rewards can be represented by a Markov Decision Process (MDP).

Back within the domain of Robocup, [99] uses a combination of Reinforcement Learning and Semi-Markov Decision Processes (SMDP) to enable a team of Robocup agents to learn to play keepaway association football, where the aim of the game for the team in possession of the ball to keep possession of the ball for as long as possible. SMDPs, which generalise MDPs, are used to enable handling extended actions that last longer than one state transition.

A more general case of robotic acquisition of behaviour is given in [100], where a specific variant in reinforcement learning, called Q-learning, is used in conjunction with a NN which has the capacity to constructively grow. The main advantage of Q-learning is that it does not require a model of the environment; it couples states and actions together and maps them onto a reward value (which are by default stored in a table). Over the course of the online learning, providing the learning converges, this table becomes closer and closer to the optimal policy. The NN is used as a function approximator, enabling the Q-learning system to not have to use a table (which can get very large if there are a large number of states and/or actions). At the start of learning, the Q-learning table has identical rewards for all entries (unless biased with *a priori* knowledge).

2.4 Summary

This chapter has reviewed work in the information retrieval domain for indexing large datasets of varying composition and complexity. The concept of indexing was introduced, that is the creation of a mapping from dataset entries to compressed/simplified keys (the index), and then two general approaches to indexing were reviewed: context based indexing and content based indexing. Of the two, context based indexing is the simplest and uses information which is associated with the indexed data, or into which the indexed data is embedded, as the basis for

the index. Google image search is a good example of context indexing, with the hypertext into which images are embedded providing the context. The second approach of content based indexing is more popular, both in general use and in research, because it is a more powerful approach which analyses the nature of the data itself to generate indices. It can be categorised into approaches which utilise global properties of the data being analysed (such as a colour histogram of an image), or those which utilise a collection of local properties (such as a collection of SIFT descriptors for an image), with both approaches being well represented in the literature. Examples of content based indexing were covered spanning WWW pages, text corpora, images, music, video and trajectory data. With regards to this thesis, current sport based indexing was shown to be predominately in the video domain, which gives scope for this thesis to extend this research into the trajectory domain, which is covered in chapters 3 and 4.

A review of behavioural modelling was also undertaken, with literature from the fields of manually-created, dataset induced and experientially learnt behaviour models being covered. Whilst all three fields are well represented, and can each produce effective models, it is the level of autonomy of learning which separates them. Manually created behaviour models are most suitable if a definite goal or set of behaviours can be expressed (or is desired), and may require extensive human intervention to create the model. Data induced behaviour modelling is most suitable for replication/simulation of recorded behaviour, and requires human attention to prepare and present the dataset to the learning process. Experientially learnt behaviour models are most suitable if a definite environment to operate in is known, but a specific set of required behaviour is unknown or mutable, and requires the least intervention after the initialisation of the learning process. For the purposes of the behaviour modelling aspect of this thesis, which is to produce accurate behaviour models of players from captured player trajectory data, the modelling via induction from data approach is the most suitable, and is this undertaken in chapter 5.

3 Indexing

This chapter presents a framework for indexing segments of football play, using a data partitioning indexing model which allows a two stage index query process to occur : the first stage efficiently partitions the total data into a smaller subset of relevant indexed objects; the second stage ranks indexed objects in the subset using additional indexing metadata. Within this general indexing model, multiple novel indexing representations are developed to cover player trajectories individually, as groups of players, and as a team mass, and to index the trajectory of the ball. The competing representations are then evaluated.

3.1 Introduction

Football is a complex game, but it can be characterised at a tactical level by collections of relatively short periods of play. Set pieces such as corners and free kicks, and free flowing moves such as wing attacks can all be viewed as segments of play, with at least approximate start, duration and end times (this is somewhat subjective in the case of free flowing moves). Once matches are decomposed into a series of segments of play, the possibility of comparing matches at this level emerges. The ability to find similar segments of play in games by presenting an example (either archetypal or actual) would be a powerful tool in the hands of football coaches. As covered in chapter 2, this type of functionality is what is offered by indexing systems; namely the ability to characterise a collection of objects, and then to allow similarity searches over this collection by presenting a query object of the same type. This is the problem being addressed in this chapter; namely can an indexing system be devised that will accurately describe segments of football play in a compressed manner and allow the discovery of similar segments of play when presented with a query Segment Of Play (SOP).

3.2 Formal problem statement

Expanding on the introduction section, the required functionality for the indexing of SOPs is:

(1) The indexing system should be able to index SOPs, at any point in a match, down to a fidelity of 0.1s, matching the fidelity of the tracking data.

(2) The user query should take the form of an arbitrary initial player configuration and at least a general notion of how this configuration evolves in the near future. This could take the form of an actual example SOP, or it could be produced as the result of a more complex operation where the user individually specifies types of team movement he/she is interested in finding.

(3) The indexing system should analyse the user query and use it to locate similar segments of play from all available indexed segments of play, whilst simultaneously rejecting bad configurations (Figure 3.1).

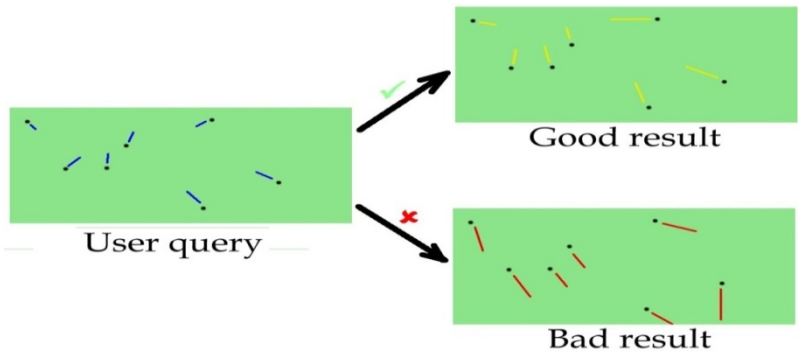

Figure 3.1 – Indexers should find similar results whilst rejecting the majority of dissimilar results.

(4) Trajectories on the pitch are spatially context sensitive. In Figure 3.2 even though the trajectories A and B are identical in relative movement terms, they are very different within the context of a football match. Any indexing system should be able to cope with this context sensitivity.

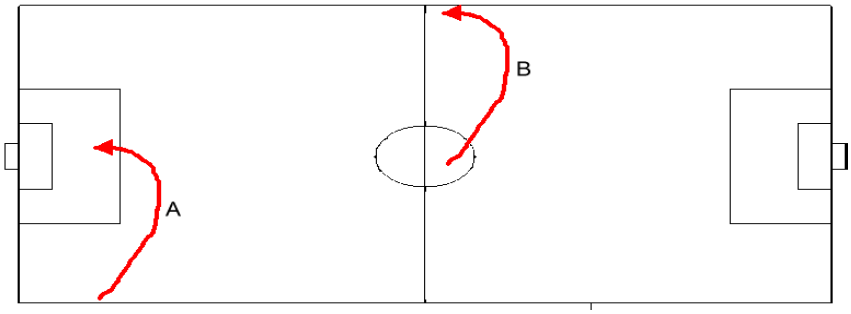

Figure 3.2 – Spatially context sensitive trajeories. Although trajectories A and B have the same shape, they have different semantic content within the game.

3.3 General Indexing Model

Two broad approaches to indexing emerge when it is viewed from the perspective of how queries are satisfied. The first approach applies a similarity metric between the query (which is firstly decomposed into a suitable index format if it is query by example) and each individual indexed object entry, such that given n indexed objects a set of n similarity ratings are produced. This approach is used by the majority of the indexing work reviewed in chapter 2 [19,20,22,23,24,25,26,27,28,30,37,47,48,49,52,56]. It has the advantage that the resultant ranking produced is complete, and assuming the object indexes are representative of the object and the similarity metric is valid, will produce an accurate ordered list of the most similar objects matching the query. The disadvantage is that all indexed object entries must be operated upon by the similarity metric as there exists no means to preselect subsets of the index before application of the similarity metric. This could become prohibitive computationally if the similarity metric is very complex and/or if the volume of indexed objects becomes very large.

The second approach involves partitioning the indexed data prior to any queries, based upon the contents of the index (or subparts thereof). Queries, which are themselves decomposed into an indexed form, can then be used to preselect subsets of data which are relevant to the query based upon which data partitions match with (portions of) the query index. The resultant subsets can then be ranked by applying the similarity metric to each member of the subset. This approach has the advantage that it can efficiently deal with large volumes of data and/or complex

similarity metrics by quickly selecting a smaller subset of data. For this reason it is the favoured approach of systems which do have large volumes of data such as search engines [10] which use full inverted indexes to partition (realised as B-trees), or moving object databases [60,61,62] which use the R-Tree and variants to partition. The disadvantage of this approach is that either duplicate results can exist if index data partitions overlap (also the efficiency of using partitioning decreases as overlap increases), or if partitions are exact then potential good results may be lost if they exist in unselected partitions.

With respect to this thesis, each individual football match has the potential to generate approximately 54,000 unique index objects (football play segments are allowed to be indexed to a 0.1s fidelity and there are 5,400 seconds in 90 minutes). Given that the volume of archival football data available is already large (5+ years worth of tracked matches), and that it is open ended, it is the view of the author that the potential volume of index data is sufficiently large to justify adopting the indexing approach of partitioning the indexing data prior to any queries.

To be compatible with the data partitioning approach, the proposed general index structure will be a composite structure (see Figure 3.3). When a given SOP is indexed, three structures are generated: the Partition Index (PI), the SOP Reference (SOPREF) and the Ranking Metadata (RM). The PI is used to partition the indexed SOPs into subsets. The PI should be abstract enough such that multiple indexed SOPs will have identical PIs[15], thereby grouping together SOPs that are viewed as approximately similar within the indexing scheme (cf. clustering, indeed the PI could be as simple as a cluster ID). Each unique PI is associated with a set of indexed SOPs; each set member is a composite object consisting of a SOPREF and RM. The SOPREF is a 3-tuple which specifies which match the SOP occurred in, the half of the specific match, and the time within the specific half at which the SOP started.

[15] ideally each partition should have approximately equal numbers of members.

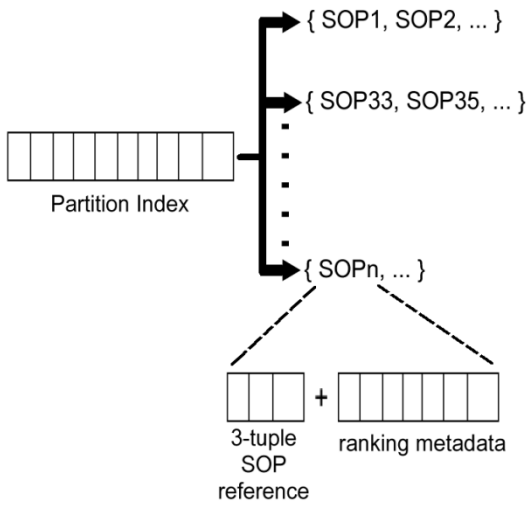

Figure 3.3 – General Indexing Structure proposed for all indexing systems – the PI is used as a key to quickly select the relevant subset of SOP data, and points to one or more SOPREF+RM pairs. The RM is used to sort the resultant set of SOPREF+RM pairs if the PI points to more than one pair.

The RM contains additional information about the SOP not referenced in the PI[16], and is used to order results. To understand its use, it is necessary to describe the general method of query satisfaction. A query by example model is taken, meaning an example SOP is used as a query and the indexing system must find SOPs that resemble it. The query SOP is broken down into a PI, and RM. The query PI is used to quickly locate the subset of indexed SOPs that are approximately similar to it. The will result in a set of SOPs which are all equally similar to the query SOP w.r.t. the PI. The query RM is then compared to each of the set members RM using a similarity metric to give a ranked list of SOPs.

In indexing systems, which utilise data partitioning, it is sometimes necessary to relax an initial query in order to encompass a larger subset of the indexed objects. For instance, web search engines often use synonyms of keywords to relax strict keyword searches, and moving objects databases have the capability to expand search regions. The ability to relax a query will be included in the indexing schemes proposed. This relaxation should be an iterative process, i.e. it should gradually expand the number of valid partitions available, until the most relaxed search effectively includes all entries in the indexing scheme (i.e. it is now functionally

[16] or if referenced, then in a non-abstract form

equivalent to the first approach of applying the similarity metric to all indexed objects). Query results will contain not only the similarity score, but also the relaxation level they were discovered at. The exact details of the indexing schemes, PIs, RM, and search relaxations will be dependent on the indexing scheme it is based upon, and so will be left until specific indexing representations are explored (see sections 3.5 – 3.12).

3.3.1 Multiple Indexing approaches

With the most trivial of indexing systems, the question of similarity is almost a moot point, an index reference in a book either matches precisely or it does not match at all. When the domain being indexed becomes more complex, the question of similarity can become as much subjective as objective. This is most certainly the case for football. To the best knowledge of the author, there exists no body of data that provides similarity ratings between segments of play/team movements/set pieces at different times/in different matches. Since there is no empirical information available addressing similarity, a number of different approaches to indexing, each employing a different representation but all fitting within the general indexing model proposed, will be taken. The multiple indexing approaches will be formally evaluated against each other via an experiment with human volunteers, who will provide their subjective similarity ratings. It is hoped that analysis of the similarity ratings will then be able to rank competing indexing systems against each other. In that spirit the next sections of this chapter will outline the differing indexing methods explored.

3.4 Preliminaries

This section introduces some necessary details concerning the data, and how some aspects of it are transformed, which is pertinent to the indexing sections that follow. The pitch coordinate system origin is the centre spot of the pitch (Figure 3.4). Pitches within the data obtained were not of identical dimensions. To normalise the player positional data over all matches, each player coordinate was scaled in the following manner:

$$x' = \frac{x}{p} \tag{3.1}$$

$$y' = \frac{y}{p} \tag{3.2}$$

Equations (3.1) & (3.2) perform the pitch coordinate normalisation; where x and y are the raw pitch coordinates of a player in the current match; x' and y' are the pitch normalised coordinates of a player in the current match; p = ½ pitch length of current match pitch. The y' term in Equation (3.2) is normalised with p, rather than the distance from the centre point to the sideline, as doing this would result in y extending from -1.0 to +1.0 leading to deformations when measuring the Euclidean distance on the pitch (when using normalised coordinates). The author feels it is more important in this case to preserve Euclidean distances than to have an absolute y location of the sideline.

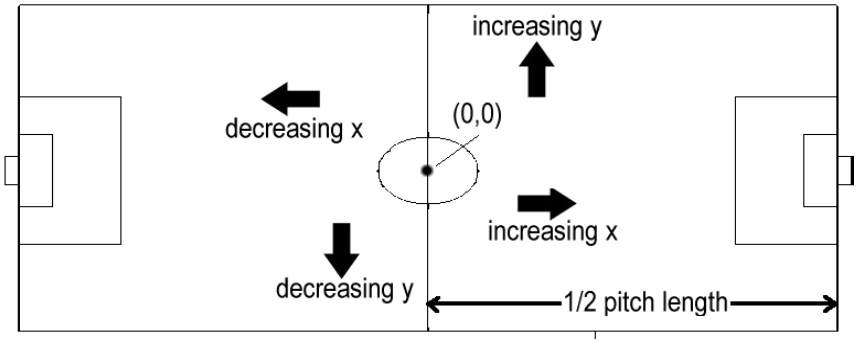

Figure 3.4 – pitch coordinate system (normalised to ½ pitch length)

Each player is assigned a unique player ID that persists over the entire lifetime of the player, even if he changes clubs. Within the data, each players ID is associated with a club ID in each match. This makes it easy to differentiate players of opposing teams, but some method of abstracting teams is required for the indexing process. The method chosen involves characterising each team by which half of the pitch the team is currently attacking. At the beginning of each half (for each game), it is easy to determine in which section of the pitch the two opposing clubs reside by calculating the mean x position of each team. If the mean x position is $-ve$ then the club is attacking the goal in the $+ve$ portion of the pitch and the club is labelled the '+1' club. If the mean x position is $+ve$ then the team is attacking the

goal in the $-ve$ portion of the pitch and it is labelled the '-1' club. A mapping of clubs to abstract teams can be generated for each (match, half) pair.

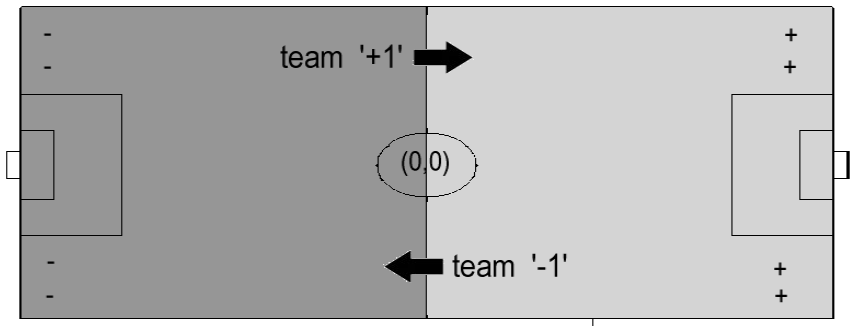

Figure 3.5 – team abstraction allows teams to be anonomously differentiated by direction of play

3.5 Context indexing with player cliques

The two main approaches to obtaining indexing information from objects covered in chapter 2 were Context indexing and Content based indexing. Whilst the content-based approach is certainly suitable, given the complex data that exists within a SOP, this section examines whether the context based approach, which uses information that surrounds/is attached to the object to be indexed is suitable for segments of football play.

There exists global information that could be viewed as the context around a SOP; items such as the venue, playing home/away, half/direction of play, score and time. However, the author doubts there is any significant and systematic correlation between this information, and what is occurring during a particular SOP, and so using the pre-existing global context information available as the basis of context indexing will not be pursued.

Manual labelling by humans viewing segments of play is a theoretical possibility, although it is difficult to imagine encapsulating useful information about a SOP in a single word (over and above what is already available via the recorded player events). A phrase would have more capability of capturing at least some aspect of what was occurring (i.e. cross from a midfielder from the top-left corner), but the use of phrases would make it difficult to match user entries if the model of

[12] were used (which is the most feasible validated manual labelling approach given the volume of data). It would be possible to increase the likelihood of matching phrases by markedly restricting both the lexicon and the syntax of the manually entered phrases, but this would entail either training users in the lexicon/syntax, or a complex conversion of free text to the restricted syntax. For these reasons, the manual labelling approach will not be pursued.

From within the game itself, one reasonable candidate for the context around a SOP could be the initial and final positions of players at the beginning and end of a SOP. This is a basic notion of context, but as [14,15,17] demonstrate, more discriminating indexing features can be discovered if there exist higher-level semantic relationships. This thesis posits two such higher-level semantic relationships, which are:

(1) Proximity – Players close to each other can be considered associated together.
(2) Direction of movement – Players demonstrating extended movement in approximately the same direction can be considered associated together.

The proximity criteria are motivated by the fact that players close to each other (a) tend to be aware of each other and so can synchronise behaviours on an ad hoc basis, and (b) tend to be part of the same team formation (i.e. a defensive subgroup) and so are trained to move in a synchronous manner. Likewise, the direction of movement criteria in which a group of players are moving in approximately the same direction suggests some degree of synchronisation amongst players, whether that be purely because they are all following the ball, or as some learned tactical move. Therefore, to clarify, the subgroups of interest w.r.t. context are:

(1) Subgroups of players associated by proximity at the beginning of the play segment.
(2) Subgroups of players associated by similar extended direction of motion at the end of a play segment.

3.5.1 Discovering cohesive player subgroups

Four methods of discovering subgroups of players were investigated, which were: k-means clustering, agglomerative clustering, graph partitioning and clique detection. For the purposes of this section proximity is defined as the Euclidean distance between two players, and direction of movement is defined as the angular component of the velocity needed to move the player from his initial position at the start of the play segment to his final position at the end of the play segment (see Figure 3.6).

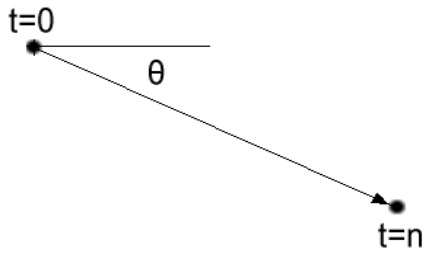

Figure 3.6 – interpolated direction of movement from the beginning of a SOP to the end.

3.5.1.1 Standard clustering algorithms

The k-means clustering algorithm [101] had a fundamental mismatch for this particular problem, in that it requires as a constraint the number of clusters *k* it is to produce. Given that the number of player subgroups during a game is variable, and not know at any particular instant *a priori,* this rules out using k-means for this particular task. Agglomerative clustering [102], is a deterministic, bottom up approach to clustering which builds up clusters from elements/sub-clusters according to the following logic:

(1) Initially, put each element in its own cluster.
(2) From all current clusters, pick the two clusters with the smallest distance between them providing it is below a given threshold.
(3) Merge these two clusters together.
(4) Repeat steps (2) + (3) until no more cluster merging is possible.

This is a suitable clustering algorithm, as it makes no assumptions on the numbers of clusters that will emerge at the end of the clustering process. An inter cluster distance metric is required by agglomerative clustering, and the average linkage metric was used, which defines cluster distance as the average distance between elements in the two clusters[17]. Agglomerative clustering produced stable, repeatable clusters of players, using both the proximity metric and the direction of movement metric.

3.5.1.2 Graph Partitioning

Graph partitioning [103] involves the division of an initial large graph into smaller disconnected sub graphs by removal of graph edges. The raw player data can easily be converted into graph form, where the vertices of the graph represent players (expressed as unique player IDs from the raw data) and the weighted edges represent the distance between players (either Euclidean or angular).

Many standard graph partitioning algorithms (as typified by the Metis package [104]) although they are very efficient suffer from the same drawback as the k-means algorithm, namely they require a target number of sub-graphs to produce from the initial graph. They also typically try to cleave the graph into equally sized sub-graphs, removing the least number of edges possible, which is not what is required for this problem. However, a bespoke approach to graph partitioning can be taken, whereby given a threshold value all edges within the graph above the threshold are removed, and then the graph is analysed to identify all disconnected sub-graphs. This is approximately the same overall computational complexity as agglomerative clustering.

Comparing the results to that achieved from agglomerative clustering, it was noted that the graph partitioning on average produced fewer but larger subgroups. After further examination, the reason behind this was discovered to be the use of

[17] The alternatives are to use the maximum distance between two clusters (complete linkage) or the minimum distance between two clusters (single linkage).

average linking for the cluster distance metric. If single linkage[18] had been used then the results would have been identical (in terms of sub-group membership), however average linking has the effect of breaking associations between players which are below the threshold level based on their current cluster associations. These cases highlight that some players really belong in more than one subgroup at any one instance. There are variants of k-means that allow fuzzy cluster membership [105], enabling elements to belong partially to two or more clusters, however they suffer the same drawback of requiring the initial number of clusters parameter.

3.5.1.3 Cliques

An alternative approach from graph theory/social network analysis is the concept of the clique [106]. A clique is defined a maximally connected sub-graph, that is all members of the sub-graph are directly connected to each other. Figure 3.7 shows an example of the cliques existing within a graph of 8 vertices/11 edges.

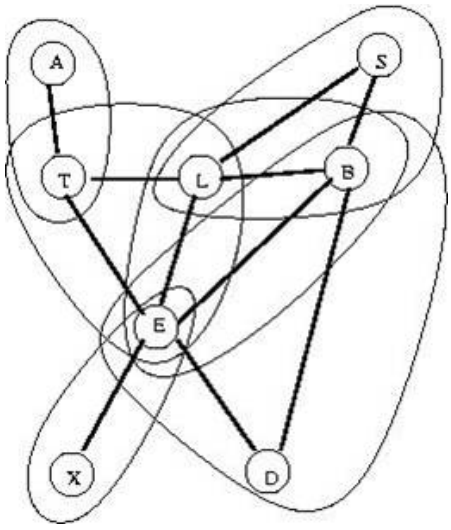

Figure 3.7 – 8 vertices decomposed into 6 cliques: {A,T} {B,E} {E,X} {B,D,E} {B,L,S} {E,L,T}

Clique discovery proceeds in an identical fashion to the graph partition algorithm in that it removes all edges above the threshold level. The clique discovery algorithm has to find the maximally connected cliques in the graph, and as

[18] The single linkage cluster distance metric is defined as the closest distance between any elements in the two clusters.

such is somewhat more computationally expensive than merely locating disconnected sub-graphs. The actual clique discovery implemented uses a brute force approach to enumerate through each vertex in the graph, then enumerating through all directly connected neighbour vertices to discover the cliques that include the current vertex. Whilst this algorithm was conceptually simple, for large graphs this would be computationally prohibitive (more complex efficient solutions are enumerated in [107]). However since the graphs is question will never exceed eleven vertices (the maximum number of players allowed on a team), this never actually becomes prohibitively expensive.

As can be seen from Table 3.1, which enumerates clique membership in Figure 3.7, using cliques allows vertices to be members of more than one group. This is what was missing in the clustering and graph partitioning approaches. Application of clique discovery to the player data confirmed that players were allowed to appear in more than one clique (both for proximity and for direction of movement). Thus, cliques provide a richer description of the team subgroups than either clustering or graph partitioning, and are therefore the preferred method of player subgroup discovery.

Vertex	# of cliques vertex member of
A	1
B	3
D	1
E	4
L	2
S	1
T	2
X	1

Table 3.1 – Clique membership count for the network shown in Figure 3.7

For the implementation details of the clique indexing system, see section 7.9

3.6 Team mass indexing with 2D histogram

The indexing scheme proposed in this section posits that when judging similarity between two segments of play, humans do not concentrate on the movement of individual players as much as they concentrate on the movement of the bulk of the team. It is a content based indexing approach, and is inspired by the MAMs model [42], where motion activity over a period of time in a video is recorded in a 2D histogram covering the entire spatial extent of the video.

In this indexing scheme a 2D histogram overlaid onto the football pitch (one per team) records the positions of players during the SOP, resulting in a spatially quantised representation of player position/density over the play segment. Given that in the indexing context we would like some level of abstraction, recording the accumulated motion of players on a very fine grained 2D histogram would likely result in an indexing system which was far too specific. Therefore a coarse 2D histogram of only 24 bins was used, divided into a 6x4 configuration of labelled bins (see Figure 3.8), which fits precisely over the football pitch.

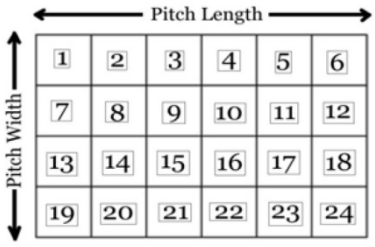

Figure 3.8 – labelled player density 2D histogram with 24 bins covering the entire rectangular playing area.

The histogram records the spatial location of players over the SOP, allowing both the trajectories of moving players and relatively stationary players such as goalkeepers to be recorded. To further increase the abstraction the histogram bins are sorted in order of descending magnitude and the top six bins are used as an indicative measure of where the bulk of the team was located during the SOP (see Figure 3.9).

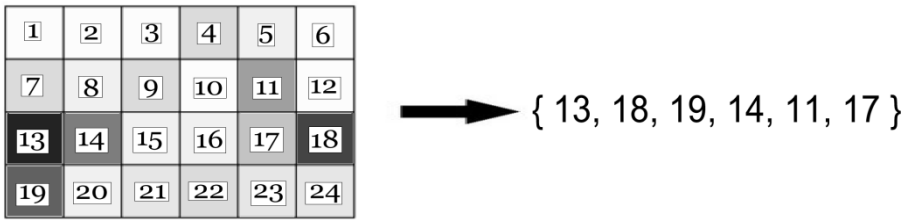

Figure 3.9 – conversion of 2D histogram to index via selection of the six bins which record the highest player densities.

For the implementation details of the Team mass indexing with 2D histogram system, see section 7.10

3.7 Team mass indexing with multi-resolution 2D histograms

Another feature of the MAMs model [42] is that there exists a hierarchy of motion activity maps from coarse to fine (the hierarchy is actually a tree), used to characterise the motion in the analysed video at multi-resolutions. A similar approach was adopted for the next indexing scheme, but simplifying the hierarchy from a tree dividing the area studied to a nesting of finer grained histograms. In this scheme a series of three coarser-to-finer 2D labelled histograms records the spatial positions of players in a manner identical to the flat 2D histogram scheme detailed above. The hierarchy of 2D histograms follow the size progression of 3x2 then 6x4 and finally 9x6 (see Figure 3.10).

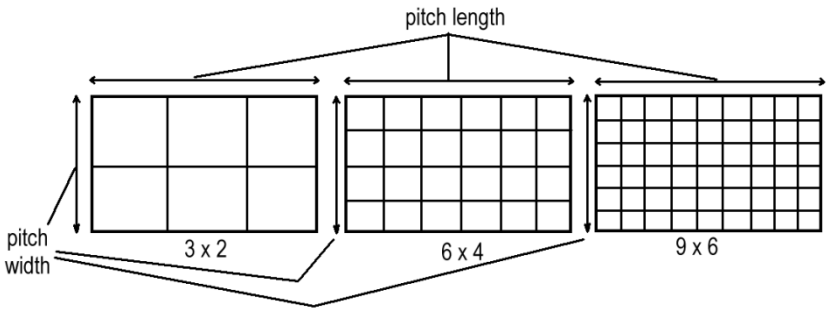

Figure 3.10 – Multi-resolution 2D histograms at three resolutions of 3x2, 6x4 and 9x6

The 2D histogram bins are not sorted in this scheme, rather the mean bin density for each histogram in the hierarchy is calculated and for each histogram those bins which are at or above this density are labelled with a '1', the bins below this mean density are labelled with a '0' (see Figure 3.11). The histograms are effectively recording all areas of above normal player concentration at three levels of resolution.

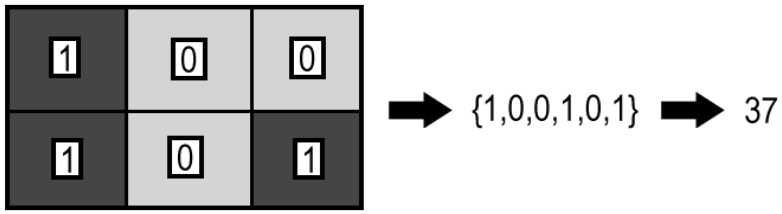

Figure 3.11 – Conversion of 2D histogram template to integer via labelling all bins above the mean density as '1' (otherwise '0'), then transforming the bins into a binary number.

To convert the histograms into a useable index, each one can be viewed as representing a binary number of 6, 24, and 54 bits respectively (Figure 3.11). These binary numbers, represented as integers, are used as the index for each team.

For the implementation details of the Team mass indexing with multi-resolution 2D histograms system, see section 7.11

3.8 Team mass indexing with local high entropy features

In pursuit of an efficient indexing scheme how well the PI vectors actually separate the data which is being indexed is an important consideration. In an extreme degenerate case, indexing may result in each SOP being mapped to the same index partition vector, which renders the indexing scheme useless. In the ideal case, assuming the number of unique index vectors is n, then the data can be split into n equally sized categories and index searching will be optimally efficient. Whilst the preceding three schemes, from manual inspection of the indexes they do produce, do not exhibit degenerate levels of indexing, this is not assured at the time

of index creation. The next two indexing schemes attempt to approach the optimal level of indexing by introducing the concept of entropy. Entropy can be a confusing concept, particularly because the concept appears both in thermodynamics and information theory. The best definition of entropy is actually the basic entropy equation which is:

$$S = -\sum_{i} P_i \ln(P_i) \tag{3.3}$$

Where:

S=the total entropy of the system under examination

P_i= the probability that the ith state of the system has of occurring.

In order to introduce the concept of entropy into indexing, if one considers a feature that may or may not be present within a SOP (and is checkable), given a collection of such segments say of size N, then one can say with certainty after checking each configuration in the collection that the features occurs M times, where $0 \leq M \leq N$. If one were to take a random sample from the configuration collection, then the probability that this feature would be present would be M/N, where $0 \leq M/N \leq 1$. If a system is envisaged with only two states, that of the feature being present and that of the feature being absent, then the total entropy S of this system is:

$$S = -(P_p \ln P_p + P_a \ln P_a) \tag{3.4}$$

Where:

P_p is the probability that the feature is present in a SOP

P_a is the probability that the feature is absent in a segment.

Note that since $P_a = (1.0 - P_p)$, the equation above can be rewritten as:

$$S = -(P_p \ln P_p + (1 - P_p) \ln(1 - P_p) \quad (3.5)$$

Now consider if one discovers a feature that is in exactly half of the collection of match configurations. This would be a perfect binary indexing feature, as it would neatly cut the data into two equal halves. The probability P_p would be 0.5 (as would the probability P_a). It can be shown that the maximum value for S in equation 3 occurs when P_p takes the value 0.5. Therefore, the maximum entropy of the system occurs when the feature is the optimal data separator.

Of course, one binary feature alone would not make a very good indexing system as it could best cut the data into two equal halves. If another optimal binary feature could be found, then this would allow the data to be separated into four equal parts. Three optimal binary features would allow the data to be separated into eight equal parts, and so on following (3.6):

$$N = 2^F \quad (3.6)$$

Where:

N = the number of parts the data could be equally cleaved into

F = the number of optimal binary features used.

The effectiveness of the indexing system grows exponentially with the number of optimal binary features used. Except that it may not. The reason it may not is that although each binary feature is optimal in its own right, they may not be optimal when combined. This can be seen clearly if one considers two distinct optimal binary features. Each on its own splits the data into two. However, they both split the data in the same manner, so using both of them is as effective as using any single one of them. One of the features is completely redundant. Entropy can again be introduced to detect whether two binary features work well together, this time via the concept of joint entropy, which is shown in:

$$H(X,Y) = -\sum_{x,y} P_{x,y} \ln(P_{x,y}) \tag{3.7}$$

Where:

$H(X,Y)$ is the joint entropy of systems X and Y

$P_{x,y}$ is the joint probability of state x of system X occurring with state y of system Y.

Which simplifies to the following equation for two binary features:

$$J = -PX_pY_p \ln(PX_pY_p) - PX_pY_a \ln(PX_pY_a) - PX_aY_p \ln(PX_aY_p) - PX_aY_a \ln(PX_aY_a) \tag{3.8}$$

Where:

J = joint entropy of the two binary features,

PX_pY_p = joint probability of feature X and feature Y being present,

PX_pY_a = joint probability of feature X being present and feature Y being absent,

PX_aY_p = joint probability of feature X being absent and feature Y being present,

PX_aY_a = joint probability of feature X being absent and feature Y being absent

The entropy will be maximised when the two binary features share as little in common as possible. To see why this is consider Figure 3.12 which symbolises the entire dataset as the complete circle. Two binary features X and Y are represented by two lines bisecting the circle (separating it optimally into two equal parts). If X and Y were both either horizontal or vertical then one of them would be redundant. To most effectively partition the circle into four equal parts, the X and Y lines must be orthogonal to each other. The joint entropy of more than two binary features can be calculated by summing the pair-wise joint entropies of all the binary features under consideration. As shown, maximising the joint entropy equates to maximising the indexing efficiency, and this gives a good metric to select sets of binary features in order to create an index.

The previous two indexing scheme used 2D histograms to reveal global features of the players movement/density over the SOP. This indexing scheme will

likewise use 2D histograms to record the player density over the SOP, but will select high entropy local features from these histograms. Five resolution levels of histogram (3x2, 6x4, 9x6, 12x8, 15x10) will be used to select the local features from (see Figure 3.13). It should be noted that on average the highest player densities will occur around the central line perpendicular to the goal line (as is the case will all of the approaches which uses regularly spaced 2D histograms), with the 'wings' of the pitch exhibiting the least average density. This may be a problem for the scheme in section 3.6 which indexes on the top six densest 2D bins. It may be slightly less of a problem for the scheme in 3.7 as this extends over multiple resolutions and indexes on all bins at or above the mean density. However, it should not be a problem for this scheme as the use of entropy should ensure that sub-areas of consistently high player density (or consistently low player density) are less likely to be chosen to be a template then sub-areas which experience variable player density.

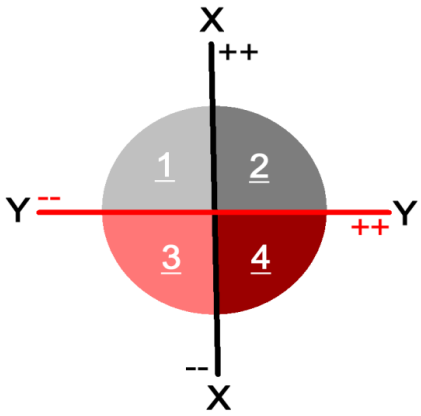

Figure 3.12 – Graphical joint entropy of two binary features splitting a space into four sections

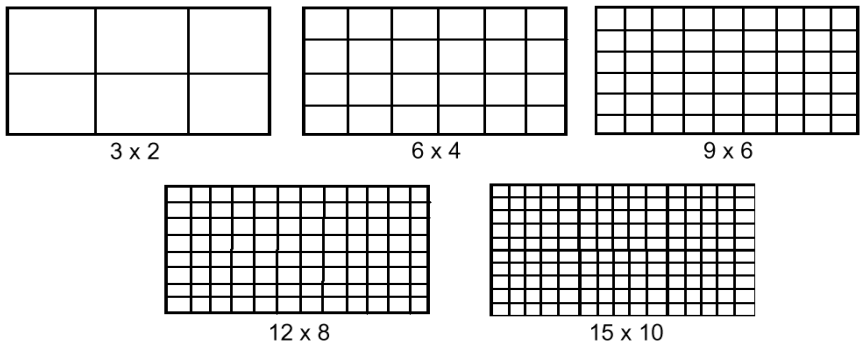

Figure 3.13 – 5 level histogram hierarchy from coarsest resolution of 3x2 to finest at 15x10.

The creation of random local feature detectors involves selecting one of the five histograms in the hierarchy, and then mapping a sub-area of that histogram (minimum area 2 bins, maximum area the number of bins in the histogram) onto the feature, as shown in Figure 3.14. The mapped bins in the feature are randomly assigned as either 1 or 0. The meaning of this assignment involves comparison with that same area of histogram once the histogram has recorded the player density over a SOP. For the feature to be present the bins on the feature that have been labelled as '1' must correspond to the highest levels of player density in the subarea of the histogram, the bins on the feature labelled as '0' must correspond to the lowest levels of player density in the subarea. Within the '1' and '0' areas no order is imposed on bin player density. Figure 3.15 show an example feature, with dark areas representing the '1' region where player density should be highest, and the light area representing where player density should be lowest.

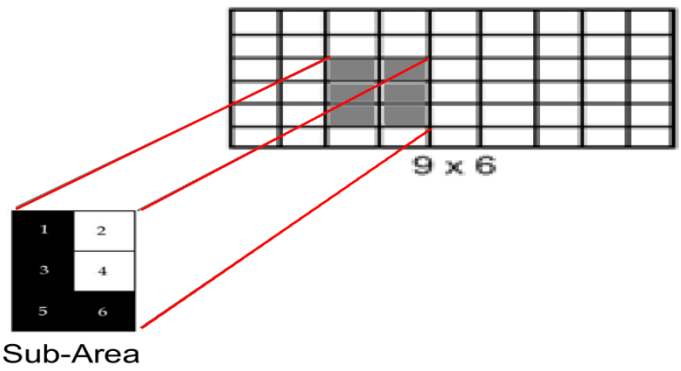

Figure 3.14 – Selection of sub-area from a 2D histogram which becomes a new histogram local feature template

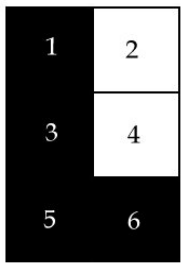

Figure 3.15 – histogram local feature template with the darker regions representing which bins should have the highest denstity if the template is be a considered a match

In order to discover an efficient collection of high entropy local features, firstly the individual local features themselves must be discovered. One set of features must be discovered per team (i.e. the '+' and '-' abstract teams). The algorithm for their discovery is as follows:

(1) Randomly sample N segments of play from the pool of those available
(2) For each of these random segments of play, generate player density histograms at all five levels of resolution for a particular team ('+' or '-')
(3) Generate a random local feature by selecting one of the five histogram levels, selecting a sub-area from that histogram, and then randomly assign the sub-area bins '1' or '0' (with meaning explained above)
(4) For each of the randomly sampled segments of play, test whether the feature is present or not. This will produce a probability of presence.
(5) The probability of presence will give the entropy of the random local feature. Reject local features with entropy of 0 (always present or never present in the random samples)
(6) Continue (3)-(5) until M random local features are discovered.

Once a pool of M randomly discovered local features is available, the 'optimal' collection of K of them can be discovered (again for a particular abstract team) via the following algorithm:

(1) Randomly select K local features from the pool of M features
(2) Calculate the joint entropy of the entire collection by summing the pairwise joint entropies over the collection of K features
(3) Randomly select one unused local feature F from the pool
(4) For each of the current K features, temporarily replace it with F, and calculate the resultant joint entropy.
(5) Replace the feature which produced the highest joint entropy when swapped out with F iff the joint entropy was higher than that calculated in (2)

(6) Continue (2)-(5) until either a threshold joint entropy value is achieved or a threshold number of attempts is exceeded.

A reasonable value for K in the above algorithm is 20 features per team, as this will result in an index that can optimally partition the data into 2^{20} sections (although in practice it is likely that only an approximation to the optimal will be attained) whilst still maintaining a compact PI.

For the implementation details of the Team mass indexing with local high entropy features system, see section 7.12

3.9 Team mass indexing with hierarchical high entropy features

The previous indexing scheme utilised a flat collection of high entropy local features that approached the optimal set by maximising the joint entropy of the collection. An alternative approach to organising a collection of high entropy local features is explored in the following indexing scheme, which structures the features in a binary tree-like fashion.

The flat collection of features operated on the assumption that each local feature was in competition with all other local features, partitioning the indexing space in a parallel fashion. An alternative approach is to view the partitioning as a serial process. If one local feature which can best partition the whole indexing space is chosen first (the feature with the highest individual entropy over the whole space), then this feature partitions the space into two (nearly) equal subsets; one in which it is present (+) and another in which it is not present (-). These two subsets will also have a corresponding local feature which best splits them (has the highest entropy w.r.t. the subset), although this will obviously not be the initial feature as it should have 0 entropy in either subset. The splitting of the two subsets into (+) and (-) smaller subsets will then identify four more suitable local features. Thus, the local features are becoming arranged in a binary tree like structure (see Figure 3.16), where the nodes of the tree represent local features, and the branches represent either the presence or absence of any particular local feature (at the immediate parent

node). This process can continue until the subsets become too small to make any reasonable determination about how well local features split them.

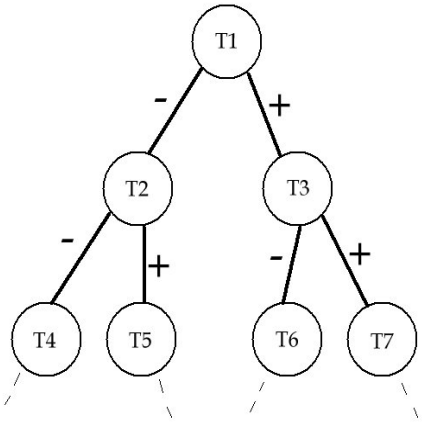

Figure 3.16 – High entropy local features decision tree cleaves the search space further at each tree level

The tree structure which emerges (along with the selected local features) can form the basis of an indexing structure. To index a SOP, one simply starts by looking for the root local feature of the tree. If it is present, then the next local feature which is checked is the (+) child, otherwise the next local feature checked is the (-) child, and so on down the tree. If each local feature in the tree is assigned a unique ID, then the tree traversal can be represented as an ordered binary vector of N components (where N is the depth of the tree). Figure 3.17 shows the initial section of two possible traversals of the tree defined in Figure 3.16.

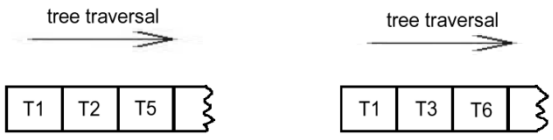

Figure 3.17 – 2 possible local feature tree traversals

As with the previous scheme, the first stage is to identify a suitable collection of local features, which can be done as follows:

(1) Randomly sample N segments of play from the pool of those available
(2) For each of these random segments of play, generate player density histograms at all five levels of resolution for a particular team ('+' or '-')
(3) Generate a random local feature by selecting one of the five histogram levels, selecting a sub-area from that histogram, and then randomly assign the sub-area bins '1' or '0' (with meaning explained above)
(4) For each of the randomly sampled segments of play, test whether the feature is present or not. This will produce a probability of presence.
(5) The probability of presence will give the entropy of the random local feature. Reject local features with entropy of 0 (always present or never present in the random samples)
(6) Continue (3)-(5) until M random local features are discovered.

Generating the tree and discovering the most suitable local feature for each node proceeds as follows:

(1) The root of the tree is the local feature that has the individually highest entropy over the entire sample set.
(2) This feature will split the sample set into two partitions, the (+) partition in which the feature is present, and the (-) partition in which the feature is absent. As the feature has high entropy, these two sets will be approximately the same size.
(3) Select the local feature which has the highest entropy w.r.t. the (+) partition
(4) Select the local feature which has the highest entropy w.r.t. the (-) partition
(5) The features discovered form (+) and (-) child nodes to the parent node defined in (2).
(6) For each child node, if its subset size is above a sufficient threshold and the current tree depth is below a given threshold T, then recursively apply steps (2)-(5), otherwise terminate tree building.

The output of this algorithm is a tree structure, with references to specific local features at each node. Both the tree and the referenced local features should be recorded for future use. Actual indexing of a SOP involves first recording the player

density at the 5 resolution levels (once for each team), and then is simply a matter of traversing the tree from root to a leaf, recording the sequence of local features which were applied at each level. If the tree is T layers deep, then this will produce a binary vector that has T components (each component representing the unique ID of a local feature).

For the implementation details of the Team mass indexing with hierarchical high entropy features system, see section 7.13

3.10 Semantically augmented ball trajectories

The previous indexing schemes have all dealt with indexing players during a SOP, either as the context surrounding a segment, or as global/local features of the two teams as a whole during the segment. One key component of any SOP that is so far notable by its absence is the ball trajectory. As covered in chapter 2, the trajectory of the ball is not recorded during the trajectory/event transcription process, but it can be reconstructed for any particular game/segment based on events and player positions (see section 7.7 for ball trajectory reconstruction algorithm).

The form the reconstructed ball trajectory takes is a collection of temporally linked line segments. The line segments represent the linear interpolation at 0.1s fidelity of the ball position between the nearest two points where it was known to be (i.e. from events such as passes, shots and ball touches). As events are connected with specific players, it is possible to attach additional semantic information to each line segment in the reconstructed ball trajectory; specifically whether the ball is currently in play or not, and if it is in play which player/club is nominally in possession of it (i.e. which player last touched the ball). Thus, the full form of the ball trajectory is as shown in Figure 3.18.

There are indexing methods that deal with individual trajectories covered in chapter 2. Of these [48,49] treat trajectories as a whole, and form a compressed representation of them (using Haar wavelets and Chebyshev polynomials respectively). [47,52] break down trajectories into a number of subparts (based on the acceleration and 2D curvature characteristics respectively), and use these

subparts as the basis of the indexing system. Although treating the trajectory as a whole as [48,49] would yield a fixed length format for the ball trajectory, it omits the semantic information of club/player possession which is attached to each line segment, and this would have to be separately encoded, which seems counterproductive as this information is already suitably connected to the relevant line segments.

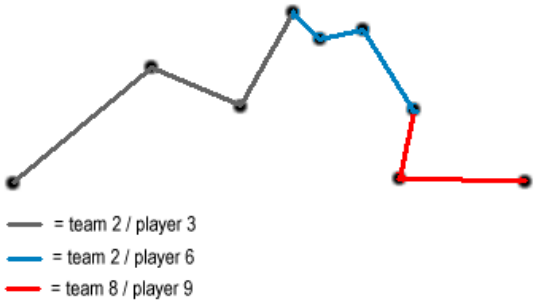

Figure 3.18 – labelled ball trajectory segments

The approach of indexing on subparts of the trajectory as in [47,52] is a better fit for the ball trajectory, although as it is already composed of line segments, there is no need to break it down initially. A drawback of these methods, in this particular situation, is they do not address the spatial position of the trajectory/sub-trajectory parts, rather they describe them in spatially independent characteristics. [47] uses features such as initial velocity, acceleration, and trajectory shape. These features are not useful for the ball trajectory as it is linearly interpolated; each line segment is a straight line, it has no acceleration and hence it maintains its initial velocity. [52] Spectrally clusters sub-trajectories, and then uses the clusters to categorise sub-trajectories, but the clustering as described in the paper does not take into account spatial characteristics.

Of course the general approach of both [47,52] could be modified to take into account both spatial characteristics and the additional semantic ball possession information. Of the two general approaches the author prefers the explicit representation of sub-trajectories in [47] over the implicit representation via clusters in [52], as clustering over multiple sources of information (spatial, direction, semantic) would require combining metrics for each source, and at least one source, the semantic ball possession, has no obvious metric. Given that the indexing scheme

will be based around line segments, the following information can be usefully included to augment the indexing scheme: spatial information about the beginning/end of line segments, ball speed across a line segment (or alternatively time taken as the velocity is constant over line segments), the semantic information detailing ball possession, and the temporal order of line segments with the whole ball trajectory spanning the segment.

The PI should ideally contain information which separates quantitatively different line segments from one another (such as those with markedly different beginning or end points or different teams in possession during the line segment), in order to efficiently partition the data. However, it should also be abstract enough to allow multiple members within each partition. A reasonable candidate for this is a combination of the spatial beginning/end of a line segment together with the semantic possession information for the segment, both suitably abstracted. The RM will be the repository of the additional information covering the time across a line segment, the temporal order of the time segment within the whole trajectory, and the exact beginning/ends of the line segment. Since the semantic information covers Club ID / Player ID, an exact version of this makes no sense as it cannot be usefully compared and so will be limited to an abstracted version in the PI.

3.10.1 Abstract spatial coordinates

A component of the PI will cover spatial information detailing the beginning/end points of line segments. This information requires a level of abstraction, and so a reasonable method must be devised to achieve this abstraction.

Spatial abstraction is covered in chapter 2 in the work on moving object databases, where the prevailing method of abstraction is the use of minimum bounding boxes (MBB) contained within R-trees (and variants), points and extended objects being grouped together if they occur within the same MBB. Whilst the more sophisticated variants of MBB usage [60,61,62] do allow efficient spatial indexes to be produced, these methods do not naturally admit additional non-spatial information (in this case the semantic information), so a bespoke R-Tree variant would be required, which unfortunately would not be mappable to a standard database R-Tree (and hence would involve creating a new database structure).

An alternative spatial abstraction scheme would be to divide the pitch into equally spaced points (as in Figure 3.19 with 12 points), with each point assigned a label, and describing beginning/end points of line segments by the nearest labelled point (thereby converting a purely spatial coordinate into a nominal category). This is a basic example of vector quantisation, and the set of labelled points form what is known as a codebook. The approach is used in [30] to form feature clusters from SIFT descriptors, and it is essentially what the 2D histograms are achieving in the team mass indexing schemes.

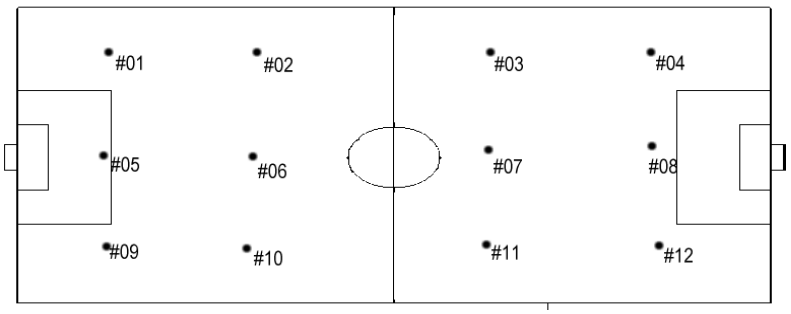

Figure 3.19 – equidistant spatial prototypes overlaid onto a football pitch

Whilst this is basically a sound approach (given an adequate density of prototype spatial points), it is perhaps not the most efficient method of spatially partitioning the pitch, as some areas of the pitch will be more heavily trafficked than others (for instance the midfield area will likely have more players present during a game than the corner regions), which will result in some spatial partitions containing many more members than others. In his PhD thesis Johnson [91], whilst researching the modelling of individual pedestrians crossing a car park used a version of vector quantisation he coined Altruistic Vector Quantisation (AVQ see section 7.1 for details of the algorithm) to generate a fixed set of prototypical spatial positions for the pedestrians. Crucially the location of these prototypes was dependent not only on the spatial positions of the pedestrians, but also influenced by the frequency of occurrence.

So the use of AVQ on the trajectories on players across the pitch (remember all ball trajectories are interpolated from player positions), has the capability to produce a non-uniform distribution of spatial prototypes. An example of such a collection of prototypes is shown in Figure 3.20, where the spatial prototypes are

shown (black points), along with a Voronoi diagram (red regions) highlighting the areas which are closest to each point. The areas of each Voronoi cell is approximately inversely proportional to the frequency that a player may be within this cell, and hence will give at least an approximately equal partition membership to all spatial prototypes.

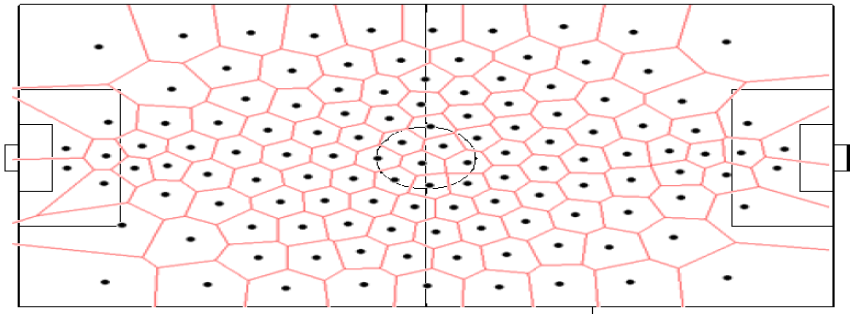

Figure 3.20 – Non-uniform spatial prototypes forming the basis of a Voronoi cell tessellation covering the football pitch

To generate a set of non-uniform spatial prototypes, AVQ requires as an initial condition the number of spatial prototypes it is to use. A method is required to generate ideally an optimal number of them. This situation is an example of model selection, and one well-known method of model selection is the Minimum Description Length (MDL) approach [108].

MDL posits that amongst models competing to represent a body of data, the best model to use is the model that achieves the best compression of the data. Crucially this compression must also take into account the size of the model itself, which stops models over fitting the data by producing fantastically complex models. Additionally any data which is unable to be represented by the model to a given fidelity[19] must remain uncompressed, and as such penalises models which would drastically under fit the data by producing trivial representations.

An example is shown in Figure 3.21, where three models H_1, H_2, H_3 exist. The classic thought experiment for MDL is that a body of data D must be communicated from a sender to a receiver. The receiver has access to all possible models which

[19] Which is an initial condition/variable of the selection process

may be used (in this case H_1, H_2, H_3). To successfully send D to the receiver[20], the sender must first indicate which model is to be used, then send the parameter settings for that model and finally send the data as it is represented under the model in use. Any data in D which cannot be expressed in sufficient fidelity under the model is sent in its original form as residual data. The cost of sending the model choice $L(H_x)$, the parameters of that model $L(w_x^*|H_x)$ and the data under the model $L(D|w_x^*, H_x)$ are measured in bits, where the operator $L(x)$ returns the length of x in bits. As shown graphically in Figure 3.21, model H_2 achieves the best compression of the message producing the shortest combined length in bits of model parameters and body of data (the cost of sending the model choice is constant and so can be ignored), and so would be the favoured model under MDL.

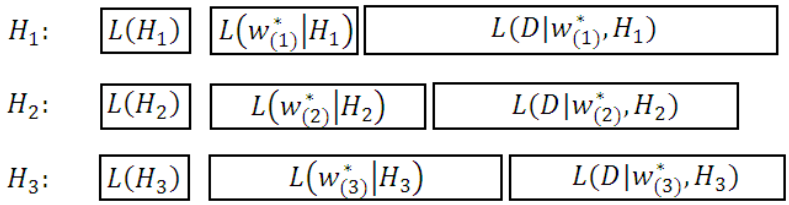

Figure 3.21 – comparing competing models under MDL

In the case of selecting the best set of spatial prototypes, the data to be transmitted D is a random sample of player positions over all games of size n. These positions could be sent uncompressed, but if a codebook of prototypical positions exist, then it could be used to record the distribution of positions across the codebook. This distribution could then be sent in place of the data. However, positions that did not fit the codebook well would have to be sent individually to allow all the data to be reconstructed without loss.

Each model will be a codebook produced by AVQ operating on D. Each model will utilise a different number of spatial prototypes, such that if there are x models, then the number of spatial prototypes used will range from $1 \ldots x$

[20] Who must be able to reconstruct the body of data *without loss*

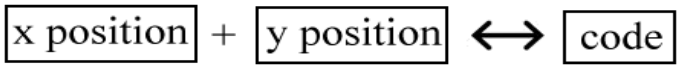

Figure 3.22 – codebook for the spatial prototypes

Figure 3.22 shows the format of one entry in the codebook. If there are x entries in codebook C_x then the cost in bits of C_x is:

$$L(C_x) = x(2R + \lceil(\log_2 x)\rceil) \tag{3.9}$$

Where R is the number of bits used to represent a real value[21], and $\lceil(\log_2 x)\rceil$ represents the minimum number of bits needed to uniquely identify each entry in the codebook. The cost of sending the body of data D of size n using codebook C_x is:

$$L(D|C_x) = \lceil(\log_2 n)\rceil x + \sum_{i=1}^{n} if\ K(C_x, D_i) \geq f\ then\ 2R \tag{3.10}$$

The first term $\lceil(\log_2 n)\rceil x$ is the cost of sending the distribution over C_x, the second term is the cost of sending individual positions which do not fit the model to fidelity f (a distance threshold), and $K(C_x p)$ is a function which takes a codebook C_x and a 2D coordinate point p, locates the nearest spatial prototype to p in C_x, and then returns the Euclidean distance from p to that prototype. Therefore if we have a representative body of data D containing player positions, N AVQ codebooks generated from D, and a suitable fidelity threshold f, then we can find the optimal number of spatial prototypes via MDL by performing :

$$min_{i=1}^{N}\big(L(C_i) + L(D|C_i)\big) \tag{3.11}$$

---

[21] This is another initial condition, and is often set or at least limited by the computing platform in practice.

3.10.2 Abstract semantic possession information

The second part of the PI that must be abstracted is the semantic possession information. This comprises the player currently in possession of the ball and the club to which the player belongs. Abstracting the club has already been dealt with in section 3.3, each club is defined as either the '+1' club or the '-1' club, depending on the direction of attack of a specific club during the SOP.

Two approaches to abstracting players were followed. The first method produces a gross archetype for each player by first estimating the mean x position (pitch normalised) of each player, players' positions being transformed (rotated 180 degrees) as necessary so that each play is a member of the '+1' club (i.e. the club which is attacking the goal in the positive x section of the pitch). The mean x positions are then clustered using agglomerative clustering into 4 clusters, which approximately partition players into goalkeeper, defence, midfield and attack positions. Specific Player IDs are then mapped onto one of these four clusters to give an approximate gross abstract role for the player. The second approach to abstracting players again uses MDL, this time to find the best number of static player archetypes to use. Before the details of the MDL selection process are covered, the abstract player model used must be explained.

Spatial prototype codebooks will be used again, but to normalise for all players must be made asymmetric (see Figure 3.23). This is achieved by randomly sampling player positions, however if a player is sampled during a match/half where it is a member of the '-1' abstract club, then its position is rotated 180 degrees about the origin. This has the effect of making each player sampled a (temporary) member of the '+1' abstract club (i.e. the club attacking to the $+ve\ x$ portion of the pitch).

Given a particular asymmetric codebook of size n, A_n, if a random sample of a particular players positions over all games is taken (again made asymmetric as above), and each position mapped to the nearest spatial prototype in A_n, then a discrete probability distribution can be calculated over the codebook prototypes indicating which prototypes the player is most likely to be found at. This probability distribution will have n entries for A_n. This can be repeated for all m players resulting in m probability distributions of length n. The probability distribution for a

particular player can be transformed into an ordered list of prototypes, if the list is sorted by the probability of visitation for each prototype (Figure 3.24).

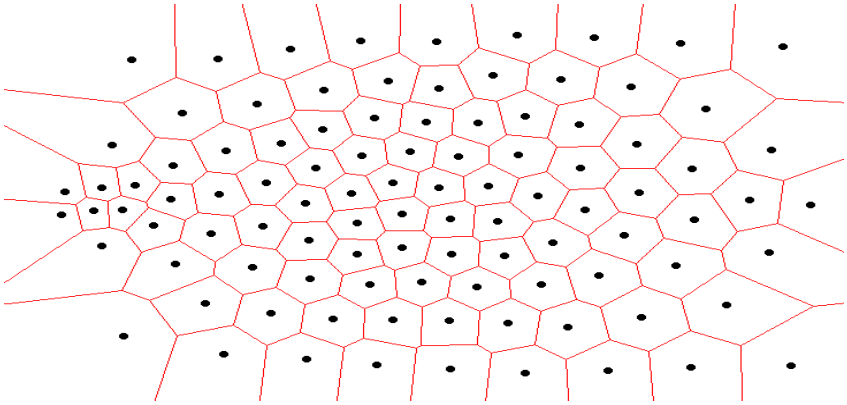

Figure 3.23 – asymmetric codebook representing average player positions built up from the motion of all available player trajectories (modified so all attacking to the right of diagram)

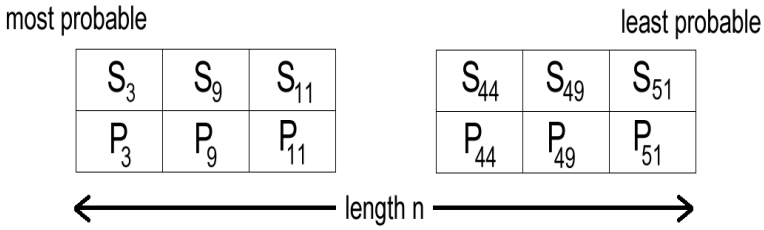

Figure 3.24 – spatial prototypes sorted by probability of visitation

Given a sorted list l of length n where $l = [(s_{l,1}, p_{l,1}), \dots, (s_{l,n}, p_{l,n})]$ and $s_{l,x}$ is the prototype and $p_{l,x}$ is the probability at position x in the list l, then the following distance metric can be imposed between the two lists l and m of length n w.r.t. the codebook over which they are created A_n:

$$D(l, m, A_n) = \frac{1}{n}\sum_{i=1}^{n} \frac{p_{l,i} + p_{m,i}}{2} E(s_{l,i}, s_{m,i}, A_n) \tag{3.12}$$

$$E(s_1, s_2, A_n) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{3.13}$$

Where $(x_1, y_1)\ and\ (x_2, y_2)$ are the coordinates of the spatial prototypes $s_1\ and\ s_2$ in the codebook A_n.

What the metric $D(l, m, A_n)$ is measuring is the mean spatial similarity of the two lists $l\ and\ m$, modulated by the mean probability of visitation. The distance metric allows the ordered lists to be clustered. If there exists a cluster of m lists of length n, then to create a truncated list T of length nc, where $nc \leq n$, which averages the cluster of lists the following algorithm can be used:

(1) Create a map X that associates each prototype in A_n with a probability. Initially each probability will be 0.
(2) Iterate over all m lists in cluster
(3) Iterate over all n positions in list l_m
(4) At position n in list l_m the spatial prototype $s_{l_m,n}$ has a probability of $p_{l_m,n}$
(5) Add $\frac{1}{m} p_{l_m,n}$ to the map entry for $s_{l_m,n}$ in X
(6) Continue (2)-(5) until all m lists are examined
(7) Extract the nc highest probabilities from X in order, along with their associated spatial prototypes and place in list T

The list T covers the nc spatial prototypes with the highest mean probability of being visited by the players represented in the cluster. In a sense, it is a model of the average player that the cluster implicitly represents. The notion of an average player allows the details of the MDL selection process to be introduced. The data D to be sent is a collection of n player histories, such that D_i is the ith player history in D. A player history is defined as a randomly sampled collection of points from the player trajectories over all games (made asymmetric). Each player history is of equal length m. As with the section 3.10.1, data must be sent with predefined fidelity f. The cost of sending D as real coordinates is given by equation (3.14). If a spatial prototype codebook of size x, A_x, is generated via AVQ from D, then the cost of transmitting the data as spatial prototypes, ignoring for the moment the required fidelity threshold, is given by equation (3.15).

$$L(D) = nm2R \tag{3.16}$$

$$L(D|A_x) = nm \log_2 x \tag{3.17}$$

If y average player models of length z are generated using the player histories over A_x, then each player will be associated with one average player model (depending on which cluster it was a member of). Figure 3.25 shows an average player model AV of size 3, which is defined over a codebook A_x. As the length of the player histories is known to be m, T can be used to generate an estimate of the number of the z most significant spatial prototypes in the player history of any player which is assigned this average player model (see equations (3.18), (3.19), (3.20))

S_3	S_9	S_{11}
P_3	P_9	P_{11}

Figure 3.25 – simple average player model (with $P_3 \geq P_9 \geq P_{11}$)

$$|s_3| \approx m.p_3 \tag{3.21}$$

$$|s_9| \approx m.p_9 \tag{3.22}$$

$$|s_{11}| \approx m.p_{11} \tag{3.23}$$

Thus, the average player model can be sent in place of the estimated number of spatial prototypes. This estimate may be larger or smaller than the real number in any particular player history, so under and over-estimate corrections will have to be included after the average player model. The process of transmitting one player history using the above model (codebook + average player models) is:

(1) The player history is mapped onto codebook A_x to form a bag of spatial prototypes. Any points which cannot be mapped to fidelity f must be sent as real coordinates.
(2) The player history belongs to a player, and that player is associated with one of y average player models of length z.
(3) The average player model estimates the number of most significant spatial prototypes in the player history. These prototypes are removed from the player history bag, but additional under and over-estimate information may need to be sent.
(4) The spatial prototypes in the player history bag not covered by the average player model are sent.

Formally the cost of sending one player history P using model $M_{x,y,z}$ where x is the number of prototypes in A_x, and y is the number of average player models of length z is :

$$L(P|M_{x,y,z}) = L(T) + L(O) + L(N) + L(F) \tag{3.24}$$

$$L(T) = \lceil(\log_2 y)\rceil \tag{3.25}$$

$$L(O) = o\lceil(\log_2 x)\rceil \tag{3.26}$$

$$L(N) = n\lceil(\log_2 x)\rceil \tag{3.27}$$

$$L(F) = f2R \tag{3.28}$$

Here $L(T)$ represents the cost of indicating which of the y average player models will be used to compress a portion of the player history. $L(O)$ represents the cost of sending o overestimation corrections as spatial prototypes. $L(N)$ represents the cost of sending the n spatial prototypes not covered by the average model. $L(F)$

represents the cost of sending f real coordinates which could not be mapped to the codebook because of the fidelity threshold. The total cost sending all player histories is therefore:

$$L(D|M_{x,y,z}) = \sum_{i=1}^{n} L(D_i) \tag{3.29}$$

The cost of the model $M_{x,y,z}$, comprising a codebook A_x of size x, and y average player models of length z, $G_{y,z}$ is:

$$L(M_{x,y,z}) = L(A_x) + L(G_{y,z}) \tag{3.30}$$

$$L(A_x) = \left((2xR) + (x\lceil(\log_2 x)\rceil)\right) \tag{3.31}$$

$$L(G_{y,z}) = yz(\lceil(\log_2 x)\rceil + R) \tag{3.32}$$

Here the term $\lceil(\log_2 x)\rceil + R$ in $L(G_{y,z})$ represents the cost of an individual spatial prototype and associated probability (as a real of size R). Therefore the best model $M_{x,y,z}$ to use given data D (at fidelity f) is :

$$min_{x=1,y=1,z=1}^{X,Y,Z}\left(L(M_{x,y,z}) + L(D|M_{x,y,z})\right) \tag{3.33}$$

The best model will include y average player models, hereafter known as the fine player archetypes. As with the gross player archetypes, Player IDs can be mapped onto the fine player archetypes to produce an abstract player type.

For the implementation details of the Semantically augmented ball trajectories system, see section 7.14

3.11 Semantically augmented individual player trajectories

The previous approaches to indexing SOP using the players and their trajectories has focussed on either treating each team as a mass, or using the players' initial position and coordinated movement to form an abstract clique-based context around the SOP. In this indexing scheme, individual player trajectories will be used as the basis for indexing.

Figure 3.26(a) shows actual individual trajectories of two teams of players over a short period of time equivalent to that of a SOP (approximately 10s). One approach to indexing these trajectories is offered in [56], which proposes a system that deals with collections of trajectories simultaneously. The trajectories are packaged into a tensor, which is then projected into a lower dimensional space via PARAFAC [57] to form a compressed matrix representation of the trajectories (and this is what is indexed). Multiple trajectory queries are likewise compressed, and similarity is based on the matrix distance between the query and each indexed object. Whilst this approach is at first appealing as there are two distinct collections of trajectories for each SOP (i.e. each team), it does not take into account the extra information that is available from the players; their individual spatial position and player type(s). This extra information allows for better discrimination between SOPs than using just the trajectory of players. For this reason, [56] was not pursued.

Figure 3.26 – (a) Two teams of individual player trajectories (b) close up of one trajectory from the collection

As discussed in section 3.10, there are trajectory-indexing approaches that operate on the level of individual trajectories, namely [47,48,49,52]. They either produce a compressed version of the whole trajectory [48,49], or they segment the trajectory and use the sub-parts as the indexing basis [47,52]. It was beneficial in the

case of the ball trajectory to take the sub-part approach, as the ball was composed of line segments to begin with, and there was a semantic theme through their temporal progression (i.e. changes of possession). In the case of individual player trajectories (Figure 3.26(b)), the trajectory exists as an ordered collection of points that does not have any additional semantic information attached to over and above the identity of the player it belongs to and its spatial position. For this reason, the extra effort involved in segmenting a player's trajectory would not produce any additional value over treating the trajectory as a whole, so [47,52] were not pursued.

Of the two remaining approaches, [48,49] are quite similar in many respects. The both separate the trajectory into x and y components (Figure 3.27) and compress each component individually (using Haar wavelets and Chebyshev polynomials respectively). The compressed x and y components then form the basis of the indexing system. Both are multi-resolution; Haar wavelets explicitly and Chebyshev polynomials implicitly (as the wavelengths of the polynomials decrease as their order increases). As they are similar, the author used Occam's Razor to select, and as the Chebyshev polynomials approach is somewhat simpler (in the author's view) this was the approach taken to represent compressed trajectories.

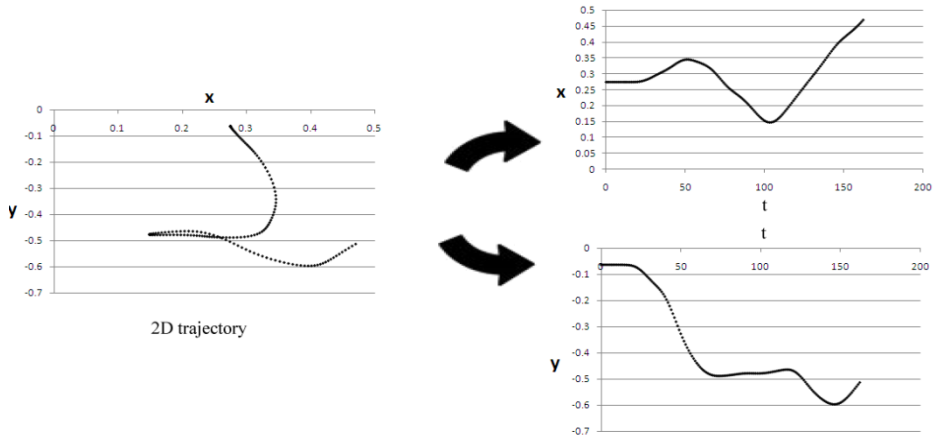

Figure 3.27 – separation of trajectory into x and y components

In detail, the Chebyshev method first separates the trajectory into x and y components (Figure 3.27) varying with time. For each of these components the following approximation process is performed on it. The Chebyshev polynomials (of the first kind) are given by the recursive definition:

$$T_0(x) = 1 \tag{3.34}$$

$$T_1(x) = x \tag{3.35}$$

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x) \tag{3.36}$$

The process of Chebyshev function approximation states that for any function $f(x)$ (suitably scaled such that x lies in the interval [-1,1]), an approximation of it using N Chebyshev polynomials can be made such that :

$$f(x) \approx \sum_{k=0}^{N-1} c_k T_k(x) - \frac{1}{2} c_0 \tag{3.37}$$

The necessary coefficients c_0 ... c_{N-1} can be calculated as follows :

$$c_j = \frac{2}{N} \sum_{k=1}^{N} f\left(\cos\left(\frac{\pi\left(k-\frac{1}{2}\right)}{N}\right)\right) \cos\left(\frac{\pi j\left(k-\frac{1}{2}\right)}{N}\right) \tag{3.38}$$

It is these coefficients that form the basis of the compressed representation of the trajectory. The higher the value of N, the more precise the fit to the function will be. It is recommended for approximations to N coefficients that a higher number are initially fitted and then only the first N coefficients used. As it stands this approximation method deals with continuous functions, but it can be adapted to work with discrete functions (such as the trajectories) by using linear interpolation between the discrete trajectory points in order to simulate a continuous function.

The other attributes of an individual player's trajectory (Figure 3.26(b)) which are useful for indexing purposes, namely its start and end positions and the type of player undertaking the trajectory can easily be extracted from the SOP. These attributes can be abstracted using the spatial and player abstractions developed in 3.10.

For the implementation details of the Semantically augmented individual player trajectories system, see section 7.15

3.12 Semantically augmented context indexing with player cliques

The final indexing scheme revisits the first indexing scheme that was a context based scheme which use player cliques to form the index. This variation uses the spatial and player abstractions developed in 3.10 to augment the clique descriptions.

In the previous clique based indexing approach, only the clique sizes (as a distribution), the type of clique (implicitly by using different distributions for differing clique types), the abstract team the clique belongs to (again implicitly) and the clique centroid (averaged in the metadata) were used to describe the cliques. These features will be kept in this scheme, but now will be used explicitly to index on the collection of individual cliques within the SOP. With the addition of spatial prototypes the centroid of a clique ('C' in Figure 3.28(a)) can be expressed in a dual sense, both as an exact coordinate and as the nearest spatial prototype.

The abstract team, and the type of clique are initial parameters for the clique search algorithm, so they are trivially available for each clique. The clique size is the number of players in the clique, and this again is trivially available as the size of the set describing the clique (composed of player IDs).

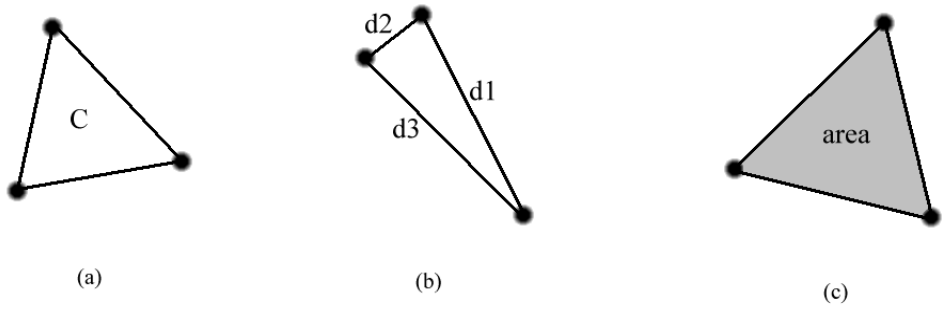

Figure 3.28 – basic clique properties of (a) centroid (b) player distances (c) clique area

The clique centroid is easily calculated, given that the clique has a membership of n players $(P_0, \ldots, P_n)$, and that each player has an associated x and y coordinate (P_x, P_y) then the clique centroid (C_x, C_y) is:

$$\overline{C_x} = \frac{1}{n}\sum_{t=1}^{n} P_{x_t} \tag{3.39}$$

$$\overline{C_y} = \frac{1}{n}\sum_{t=1}^{n} P_{y_t} \tag{3.40}$$

Since individual cliques are being indexed in this scheme, it is now possible to also include some additional information about the composition of each clique. Three additional features will be included, which abstractly describe the shape/extent of the clique. Figure 3.28(b) shows the distances between players in a 3-clique. The mean distance between players in the clique of n players, $\overline{D}$ is defined as:

$$\overline{D} = \frac{1}{n^2 - n}\sum_{i=0,j=0,i\neq j}^{n-1.n-1} \sqrt{(P_{x_i} - P_{x_j})^2 + (P_{y_i} - P_{y_j})^2} \tag{3.41}$$

The minimum (N) and maximum (M) player distance in the clique can be calculated, and from this the ratio of minimum to maximum distance, R, can be defined :

$$M = \max\left(\sum_{i=0,j=0,i\neq j}^{n-1,n-1} \sqrt{(P_{x_i} - P_{x_j})^2 + (P_{y_i} - P_{y_j})^2}\right) \tag{3.42}$$

$$N = \min\left(\sum_{i=0,j=0,i\neq j}^{n-1,n-1} \sqrt{(P_{x_i} - P_{x_j})^2 + (P_{y_i} - P_{y_j})^2}\right) \tag{3.43}$$

$$R = \frac{N}{M} \tag{3.44}$$

The final shape/extent feature extracted is the interior area of the clique (Figure 3.28(c)) which can be calculated for a clique of n players as follows:

$$A_C = \frac{1}{2}\sum_{i=0}^{n-1}\left(P_{x_i}P_{y_{i+1}}\right) - \left(P_{x_{i+1}}P_{y_i}\right) \tag{3.45}$$

In addition to the basic properties of the clique, and the abstract properties covering shape/extent, the gross and fine player archetypes can be used to enumerate the clique membership in an abstract sense (remember the previous clique scheme could not use the raw Player IDs as they are semantically meaningless). Assuming that the gross/fine player archetypes have been discovered for each available player (which is a vital prerequisite to indexing with schemes 3.10-3.12), then it is simply a matter of iterating through the clique set of player IDs and generating associated multi-sets[22] which contain the corresponding gross/fine player archetypes (Figure 3.29)

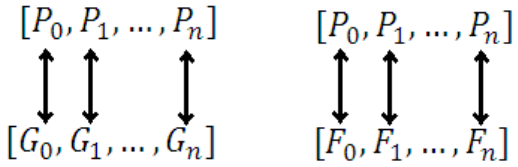

Figure 3.29 – mapping player IDs to gross/fine archetypes

For the implementation details of the Semantically augmented context indexing with player cliques system, see section 7.16

3.13 Summary

This chapter introduced eight SOP indexing schemes. Of these one was based on a semantically augmented ball trajectory, describing the ball trajectory as a temporally linked series of line segments to which were attached semantic information describing possession. This indexing scheme required the development

[22] Multi-sets are required as the use of player archetypes will generate duplicate entries which must be preserved.

of abstract prototypes both for spatial coordinates and for types of players. MDL was used in both cases to select the best model. Two of the player based indexing schemes also made use of the spatial and player abstractions: the player trajectories and augmented cliques approach. Of the remaining five approaches, four used some variation of measuring player density (two used global player density, two used high entropy local patches of player density). The final approach used a very abstract clique based approach to describe the context around a SOP.

4 Combining indexing results

This chapter presents a framework for combining indexing search results from two sources, that of ball indexing and of player indexing (the two being viewed as complementary approaches), to produce a composite ranking of results. The concept of the *context* of a query is introduced in the hope that it may serve as useful additional information in producing accurate ranking of results. Finally, a formal experiment involving multiple experimental subjects is undertaken to ascertain the usefulness of this approach, as well as that of the underlying indexing schemes introduced in chapter 3.

4.1 Introduction

In chapter 3 multiple approaches were taken to indexing segments of play. At the broadest level, these can be categorised into player-centric indexing and ball-centric indexing approaches. In reality during a SOP, both the players and the ball are active components simultaneously. It is the central claim of this chapter that whilst there is a causal connection between the movement of players and the movement of the ball, the two are not mutually redundant views of a SOP. If this premise is accepted, then the next reasonable step is to investigate methods by which, using the same SOP query, the results from a player-centric indexing system and a ball-centric indexing system modulate each other to produce a more accurate combined ranking.

4.2 The problem

The general form of the problem is shown in Figure 4.1. Two indexing systems exist (source1 and source2), both of which have indexed the same collection of N objects using different approaches. An identical query is posed to both indexing systems. Each system produces a list of n results, (where $n \ll N$) , which is ranked by a similarity measure unique to that indexing system.

The trivial case occurs when each list holds the same results in the same ranking order; the lists are effectively identical (although the similarities internal to each list may differ) and so either can be returned as the combined result. If this is not the case then the following problems exist if these two lists are to be combined into a single, ranked results list:

(1) The two lists may contain the same results, but in a different ranking order.
(2) The two lists may contain different results, resulting in a combined unique number of results of between $n + 1$ and $2n$ depending on the mismatch between results lists. The mismatched result items will not have ranking/similarity information from both sources (Figure 4.2(a)).
(3) The two sources may not be identically proficient at producing accurate results. A simple scenario could involve one source being always better at finding similar results than the other. In this case, the merging can be weighted in favour of the more accurate source. In addition, the proficiencies of the sources may depend on the nature of the query; which would then require some indication of the type/class of the query and a record of how the sources performed for that query class before a merge could be accurately made.

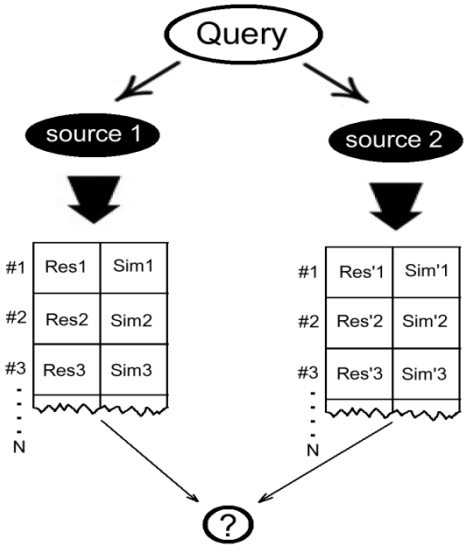

Figure 4.1 – Combining rankings from separate sources into one list

The general solution to this problem is to use some form of rank aggregation, which can be classified in two main categories. The first category examines only the order of the rank associated with each result, not the similarity. Several simple methods exist such as the Borda count [109,110], in which a ranked list of n items are assigned scores based on the distance from the top of the rank (top rank assigned n, bottom rank assigned 1). Under Borda count, aggregation simply involves totalling the score for each result in each list, the new ranking being given by the combined Borda count. This approach is degraded in the context of (2), in the worst case of no results overlap defaulting to interleaving the results (#1 from list1,#1 from list 2,#2 from list1 etc). Median rank aggregation [111] assumes d lists to be aggregated, thereby giving every result item d ranks, from which the combined rank of the item is the median value in rankings. This is not suitable in the case of only two lists as a median cannot be defined. Markov chains are used in [112], where members of the result lists are modelled as states, the transition probabilities are heuristically assigned depending on the current state (four variations of moving to 'better' higher ranking states are offered) and the aggregate ranking emerges as the Markov chain ordering. This method has the advantage that it can explicitly work with partial lists (i.e. result lists that do not share all members).

The second category of rank aggregation examines the similarity scores present in the ranked lists. Several simple methods of score fusion are presented in [113], where the unweighted minimum, maximum or sum of each result items normalised score is used to perform the rank aggregating. [114] utilises a weighted linear combination of similarity scores to rank results, the weights being determined via linear regression on training data. This is based on the assumption that similarity scores will always be available, which may not be the case if (2) holds of course. [115] models the score/similarity distributions of search engine results as normal/exponential distributions for relevant/non-relevant results respectively, then uses these distributions to map result scores onto a probability of being relevant/non-relevant to the query, allowing result lists from different search engines to be combined using the probabilistic relevance score.

In the case of this thesis, both rank and similarity scores are available in the individual result lists. Since the score implicitly defines the rank, and it also gives more information about the relative quality of results within a list (real vs ordinal), it is the preferred aggregating attribute. It is highly likely that the result lists will not

contain exactly the same results, so some result items will be missing similarity measures from one indexer (Figure 4.2(a)). The aggregator should either be able to cope with this missing information, or employ a method to reconstitute it (Figure 4.2(b)). The latter approach is preferred; as the aggregator will be part of the same system as the indexers, it should be possible to achieve this relatively efficiently.

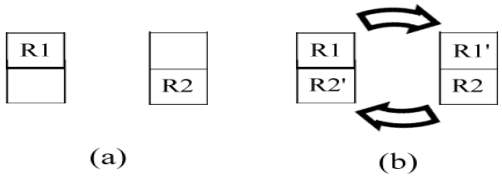

Figure 4.2 – Either coping with incomplete information (a) or filling in result similarity blanks (b) is a necessary step in merging two result lists

Comparison/combination of scores from different result lists requires some form of normalisation as they are produced by different similarity metrics. Actual normalisation is used by [113], whereas ground truth is used by [114,115] to produce suitable weightings/distributions respectively in order to combine scores. For the SOPs there is no pre-existing function which can objectively and accurately compare SOPs to produce a similarity ground truth (if there were then this function would be the obvious choice to use in the indexing system(s)). Nor do any pre-existing datasets contain previously compared and vetted SOP similarities. However a subjective ground truth can be supplied by the author, by manually comparing SOPs and assigning similarities to each compared pair. This is the preferred approach, with the caveat that the author can produce reasonably consistent similarity comparisons. A NN will be used as a general, trainable non-linear means of mapping from sets of similarity scores to estimated ground truths, which can then be used to rank the aggregated list. Training using ground truth will implicitly rate the indexers' global proficiency (i.e. the ground truth may indicate that the ball indexer produces more accurate results on average than the player indexer), however the author suspects that the type/context of the query may have some impact on the proficiency of the indexers on a per query basis, so a means of representing the context of a query should be included as additional information to be used during aggregation.

4.3 General model

Figure 4.3 shows the general model that will be used to combine the results output from a ball-trajectory based indexer B and a player-based indexer P. The process of results aggregation is as follows (also see Figure 4.3):

(a) Prior to any results aggregation, ground truth is located for P and B. A NN is trained with the available ground truth.
(b) An identical query is sent to both the indexing systems (the player and ball indexers respectively). Each indexing system decomposes the example into the specific query format used by that indexing system, and then performs the query. This process generates two result lists sorted internally by similarity to the query.
(c) The result lists are matched up w.r.t. the result items (SOPREFs) to produce a unique list of results. Missing similarity information is generated for the mismatched results.
(d) For each unique result, the two similarity scores, along with a characterisation of the query context are fed into the trained NN to produce an estimated ground truth-value for the result.
(e) Once all results have generated estimated ground truths, they are placed into a results list and ranked on their assigned estimated ground truth.

4.4 Implementation

The general form of results produced by the SOP indexing systems is shown in Figure 4.4 (covering five results). $s_1 \ldots s_5$ represents the query similarity measure, $o_1 \ldots o_5$ represents the SOPREFs associated with each result and $r_1 \ldots r_5$ represent the search relaxation level at which the result was discovered. The ranking is produced by sorting on the similarity column. A combination of the relaxation level and the similarity score will be used as a composite similarity for use with the NN ground truth estimation, as it supplies additional information about the how difficult each result was to find in the index.

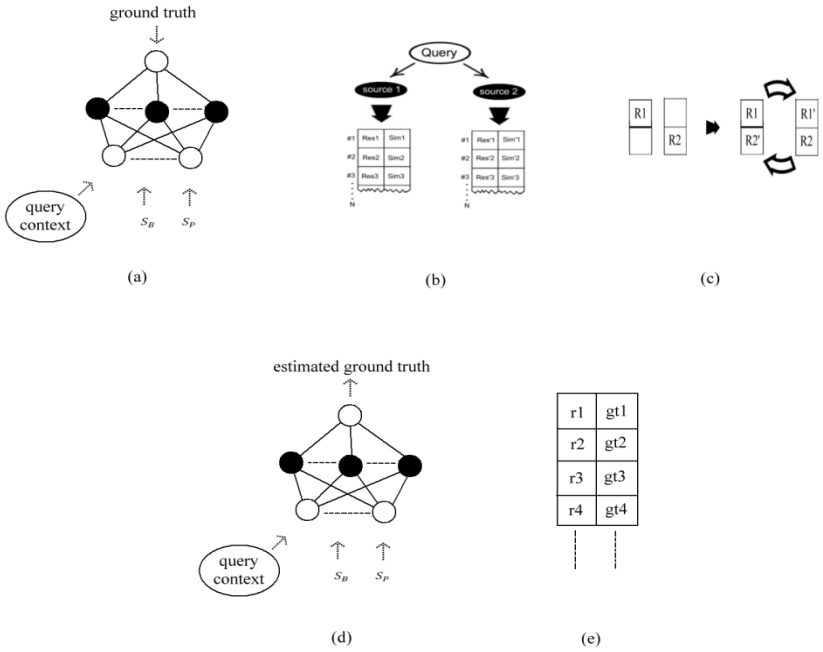

Figure 4.3 – General model used to combine results involves pretraining NN with similarity ground truths, obtaining two results lists from the complementary indexing systems, filling in any missing information in the results lists, and then using the NN to generate estimated similarities which are used to combine the two result lists into one final sorted list.

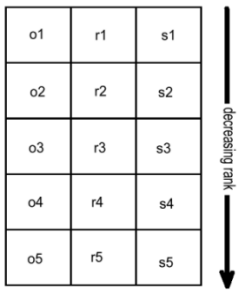

Figure 4.4 – General form of indexer results comprising a ranked list of (similarity score , relaxation level and SOPREF) 3-tuples

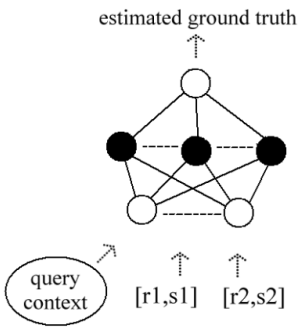

Figure 4.5 – Composite similarity includes relaxtion level to describe how difficult result was to find and a query context to describe what class of query has been initiated.

4.4.1 Supplying missing information

Consider the example in Figure 4.6 that represents the top five results generated by the same SOP query to the ball-centric indexing system and a player-centric indexing system.

ball results

01	br1	bs1
02	br2	bs2
03	br3	bs3
04	br4	bs4
05	br5	bs5

player results

02	pr1	ps1
04	pr2	ps2
06	pr3	ps3
07	pr4	ps4
08	pr5	ps5

Figure 4.6 – two result lists with some mismatch between the lists

If the result sets are aligned to match up identical SOPREFs (see Figure 4.7), it can be seen that in some cases the similarity/relaxation information is available from both indexing sources $\{o_2, o_4\}$, in other cases it is not $\{o_1, o_3, o_5, o_6, o_7, o_8\}$. In the cases where similarity/relaxation information is not available from both indexers, it must be generated.

ball results

01	br1	bs1
02	br2	bs2
03	br3	bs3
04	br4	bs4
05	br5	bs5

player results

02	pr1	ps1
04	pr2	ps2
06	pr3	ps3
07	pr4	ps4
08	pr5	ps5

Figure 4.7 – result list alignment (with resultant void spaces which require filling)

Consider an example in which two indexing systems, P and B, are given the same query (by example). P generates query Q_p, B generate query Q_b. Both queries are performed on their respective indexes and generate two result lists. There exist mismatches between the result lists, and one such mismatched result O has relaxation/similarity information from indexer P but does not have

relaxation/similarity information from indexer B (the missing information R_m and S_m). Assuming both indexing systems have indexed identical collections of SOPs[23] (of course from different perspectives), the SOPREF associated with O can be used to locate the indexing information for the SOP in the indexing data of B[24]. This indexing information consists of a set of PIs P_b and a set of (SOPREF, RM) pairs[25] R_b. Once located these two sets (P_b and R_b) can be used to generate the missing relaxation/similarity information R_m and S_m as detailed in the next two subsections.

4.4.1.1 Generating missing similarity measures

The query object Q_b has an associated set of PIs, P_q, and an associated set of RM R_q, which are generated from the example SOP given as a query. Given P_q , R_q , P_b and R_b, then if:

(1) The indexing system B has a query similarity metric, M_B, which only uses the RM, then $S_m = M_B(R_q , R_b)$

(2) The indexing system B has a query similarity metric, M_B, which uses both the PIs and the RM then $S_m = M_B (P_q , R_q, P_b , R_b)$.

4.4.1.2 Generating missing relaxation information

The query object Q_b has an associated set of PIs P_q, which are generated from the example SOP given as a query. The following algorithm can be used with P_q and P_b to generate the missing relaxation information R_m:

---

[23] Which the author would recommend as the default approach to indexing

[24] note this is **not** a query, rather a very specific index lookup using a SOP reference

[25] For the indexing schemes proposed in sections 3.5,3.6,3.7,3.8 and 3.9 these sets will only contain one member, but in the case of the indexing schemes proposed in sections 3.10, 3.11 and 3.12 the sets will contain multiple members (as they deal with multiple ball trajectory segments, player trajectories and augmented cliques respectively), but they are all associated with the same SOP reference. This fact does not affect the relaxation/similarity generation, it is only for additional clarification.

(1) Initial relaxation level $r = 0$
(2) Using the PI matching logic of B at relaxation level r, if any members of P_q match any members of P_b, then $R_m = r$
(3) If no matches are found between P_q and P_b, if $r + 1$ does not exceed the maximum relaxation of B, then $r = r + 1$ and return to (2)
(4) If $r + 1$ exceeds the maximum relaxation of B, then $R_m = r$

The purpose of the algorithm is to determine what relaxation level under B the original query would have been discovered at. The fourth clause is the base case of this recursive algorithm; ensuring that it always terminates at the maximum relaxation (which essentially means that the PIs match every potential search result).

4.4.2 Query context

It is another claim of this chapter that the type of query being performed may influence the quality of results returned from an indexing system. To test this claim, some form of categorisation of the type of query will be required in order to be coupled with the similarity information as input for the NN ground truth estimator.

The available information associated with a query is the PI, RM and the SOPREF. Of the three, the SOPREF does not hold any information about the actual content of a SOP, only when it begins, so is not a good candidate for categorisation of the query. Of the remaining two, the PI seems the logical choice as a source of categorisation information, as its function is already to approximately categorise an indexed collections of SOPs into subsets of similar SOPs.

Depending on the indexing scheme in question the PI is either a fixed length vector (for sections 3.5, 3.6, 3.7, 3.8, 3.9) or a variable number[26] of fixed length vectors (for sections 3.10,3.11,3.12). The ground truth estimating NN accepts two compound similarities (four values in total) and then a categorisation of the query context from the perspective of both indexing systems. If the context section of the input is larger, in terms of input nodes, than the compound similarity section then there exists a danger that the NN will be preferentially sensitive to the query context over the similarity input (particularly if the input values in the context section are

[26] in which case the resultant vector length is assumed to be the longest possible length with vector padding applied if necessary for smaller resultant vectors

highly uncorrelated). As the purpose of the context information is to allow the NN to process the simililarity in a more nuanced manner, its domination of the input is undersirable. Therefore in order for the NN not to overfit on the query context section, it's size in terms of input nodes should be equal to the size of the similarity input section, namely four inputs (two nodes per PI context) as in Figure 4.8. Now even if the context inputs are highly uncorrelated, they should not dominate. There now exists the problem of how to select what constitutes the two inputs per PI context, as this can represent a drastic reduction in input size. It is important that when reducing the dimensionality to two, as much significant information is preserved as possible.

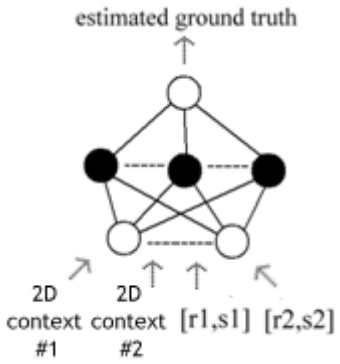

Figure 4.8 – neural network with context + similarity terms detailed

Two alternative methods of reducing the dimensionality of the PIs down into a compact 2D space were used. The first method used was the linear transformation provided by Principal Component Analysis (PCA) [116]. PCA is used to transform (possibly correlated) N-dimensional data into a new coordinate system of equal dimension N, but where the new orthogonal dimensions of the transformed space are ranked by how much variation they capture from the original data. The transformed data can then be projected into a lower dimension of n (where $n < N$), by only selecting the n most significant dimensions of the data in the transformed space. As an example consider Figure 4.9, which shows a collection of 2D values plotted on the (x, y) coordinate system. Under PCA a new coordinate system emerges (u, v); the u dimension being the principal dimension of the data, that is the dimension which exhibits the greatest variation in the data, and is the single most discriminating feature of the data. Hence the data can be transformed can be

projected into 1-dimension by only recording the value of the data on the u dimension.

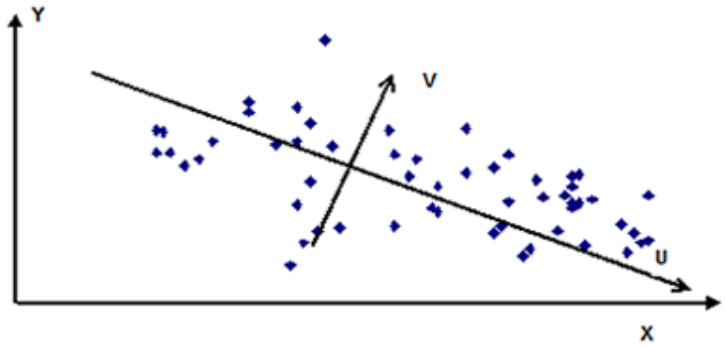

Figure 4.9 – PCA transformation from (x,y) to (u,v)

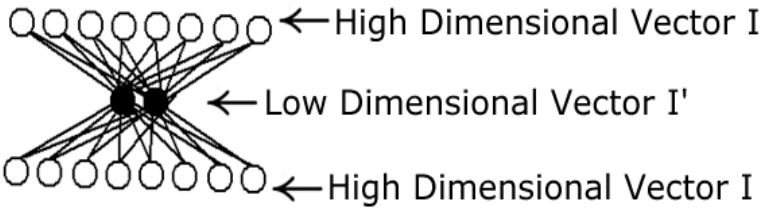

Figure 4.10 – auto-associative neural network mapping high dimensional vector I onto interior hidden nodes representing a lower dimsional vector I', effectively compressing I into I' (although the compression is likely to be lossy).

The second method of obtaining a compact 2D representation of the query context used was an auto-associative NN. These type of networks are essentially trained to perform an identity function over a data set, the central layer hidden nodes ('bottle neck' nodes) forming a compressed representation of the input vector. Figure 4.10 gives an example, where a network has been trained to perform the identity function on vectors of length 8 (vector I). However the two hidden nodes within the network form a compressed representation of I, I' which has only two dimensions. This form of neural compression has been used previously in research into the lossy compression of images [117], and has the potential advantage over PCA of being a non-linear process (as opposed to the linear PCA).

4.5 Formal evaluation

In order to properly evaluate the eight competing indexing schemes, as well as evaluate the efficacy of index results combination[27], an experiment was required. Human experimental volunteers were asked to rate the similarity between pairs of SOPs; these pairs either being generated as the result of a query, or randomly generated as a control. Competing indexing schemes were then evaluated, both in a relative and in an absolute sense, by the combined ratings supplied by the human volunteers for the index query results they produced.

4.5.1 Database selection and data pre-processing

Before the evaluation of the indexing schemes could be carried out, a suitable database system had to be selected and the data had to be conditioned to assure that it was consistent and complete. The database chosen for use was MySql 5.0 [118,119], which has the dual advantages of being free and being well optimised (for the purposes of this thesis only the MyISAM storage engine was used as transactional DB features available in innoDB were not required and they consume a lot of disk/memory resources).

Sixty-three matches of trajectory and event data were obtained from ProZone for the purposes of indexing/evaluation. This constitutes approximately 5,700 minutes of play, of which 55% is active play (the other 45% being stoppages such as ball out of play, fouls etc). The original ProZone trajectory data records player positions down to a time resolution of 0.1 seconds, however players are not recorded consistently in a temporal sense, rather they are recorded whenever they exhibit significant movement on the pitch, and this position is updated with a frequency of approximately 0.5-1.0 seconds (depending on player motion). Thus in the raw state it is not guaranteed that for a particular time instant all the players positions will be immediately available, so the missing player positions must be interpolated. All player positions were interpolated during pre-processing rather than being performed ad-hoc. During interpolation, the trajectories were also smoothed with a

[27] Of which there are 21 variants in total, (a) 7 variants of player indexer with ball indexer but with no query context, then (b) with a query context supplied via neural net compression, then (c) with a query context supplied via PCA.

time window of +/- 0.5 seconds. This smoothing is consistent with the standard ProZone data pre-processing performed before shipment of data to client clubs (as communicated during an early data format/integrity meeting with ProZone staff).

A series of annotated events for each match is supplied with the trajectory dataset. They record event type, player(s) involved in the event, and the position of the event (if this cannot be inferred from the player(s) involved). These are discrete temporal events and as such require no interpolation. The trajectory of the ball is not recorded in the raw data at all, rather it may be interpolated (as line segments) from a combination of player positions and associated events. Again, the interpolation was performed in the pre-processing stage rather than ad-hoc as an efficiency measure. Added to the ball position at any given instant was which player/club (if any) is currently in possession of the ball, and if the ball is currently in play or out of play. When the ball is out of play it remains static at the last position it was in play at until it is put back into play.

4.5.2 Indexing structure discovery

Several of the indexing methods require optimal parameters / local features / abstractions to be extracted from the data (or a representative sample thereof). The following sections briefly outline this process. Each of the indexing schemes were mapped onto a MySQL table, with the PI structured using a B-Tree, and (database) indexes placed on the columns constituting the PI in the table.

4.5.2.1 Optimal clique threshold discovery

Both the clique (section 3.5) and augmented clique (section 3.12) indexing schemes require threshold values for proximity and angular distance in order to produce the cliques from the players' positions/direction of movement. The best value for these thresholds is considered by the author to be those that produce, on average, the highest number of cliques – as this will result in the richest description. Since there is only one variable to take into account in each case, this reduces into a line-search over the range of permitted threshold values.

Five thousand segments of play were randomly sampled from the available data, and a line-search was performed over a predefined interval of threshold values

for the Euclidean proximity metric to determine the mean number of cliques produced. Figure 4.11 shows the results for the proximity cliques, where the threshold value represents player separation in normalised pitch units (see section 3.4). From the line search between the values of 0.0 and 2.0, the maximal cliques value occurs when the threshold is 0.51. Note that when the threshold is 0.0, no cliques are discovered (as no players are exactly coincident on the pitch), and values approaching and at 2.0 the number of cliques become 1.0 (i.e. there is exactly one clique and it encompasses the whole team).

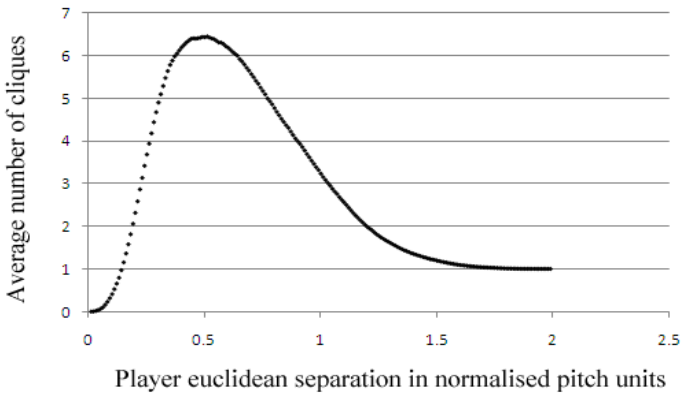

Figure 4.11 – locating 'richest' proximity clique threshold

The same procedure was undertaken for cliques produced by direction separation, here the threshold is expressed as a separation of direction measured in radians, and the line was searched between the values 0.0 and π. Once again, a definitive answer is available as shown in Figure 4.12, producing the maximum mean number of cliques at a threshold value of 0.3 radians (approximately 17 degrees separation). In this case, it is possible, if improbable, that two or more players are moving in exactly the same direction, so the mean cliques discovered for a threshold of 0.0 is a non-zero value. Values at π and above produced exactly one clique, as this covers the full range of rotational separation.

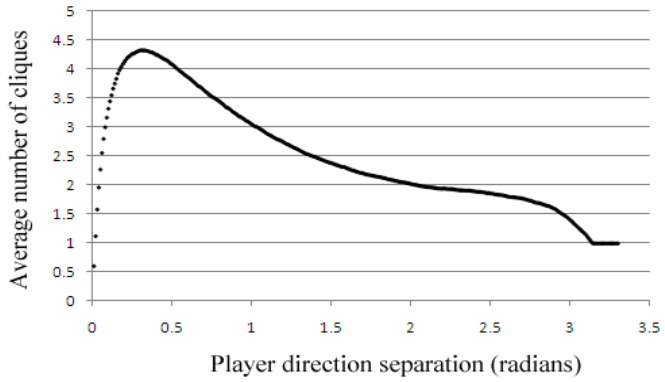

Figure 4.12 – locating 'richest' player separation clique threshold

4.5.2.2 High entropy local features discovery

Five thousand randomly selected segments of play were used in conjunction with the individual local feature discovery and local feature set discovery algorithms described in section 3.8 (see Figure 4.13 and Figure 4.14). This resulted in a collection of 20 local features per abstract team. Details of the local features discovered are available in section 7.5

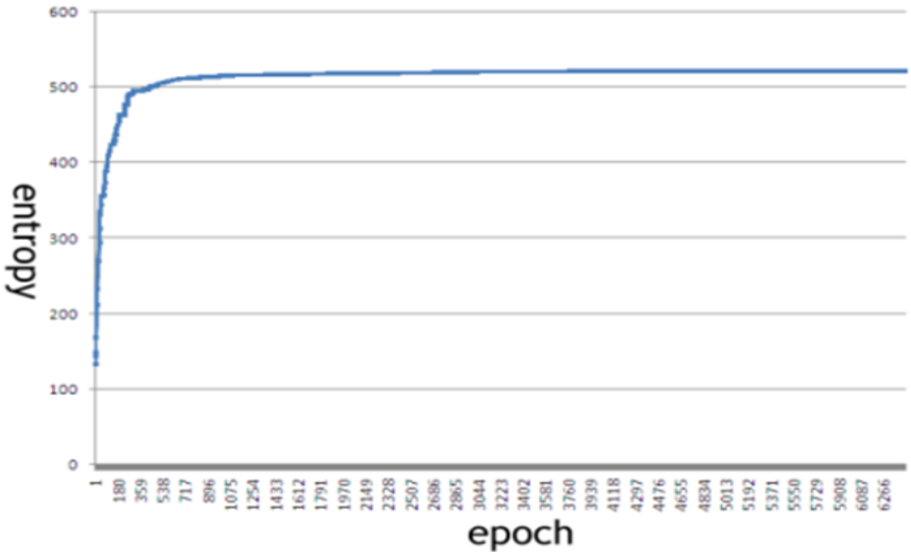

Figure 4.13 – convergence of search for high entropy local feature combinations with team '-1'

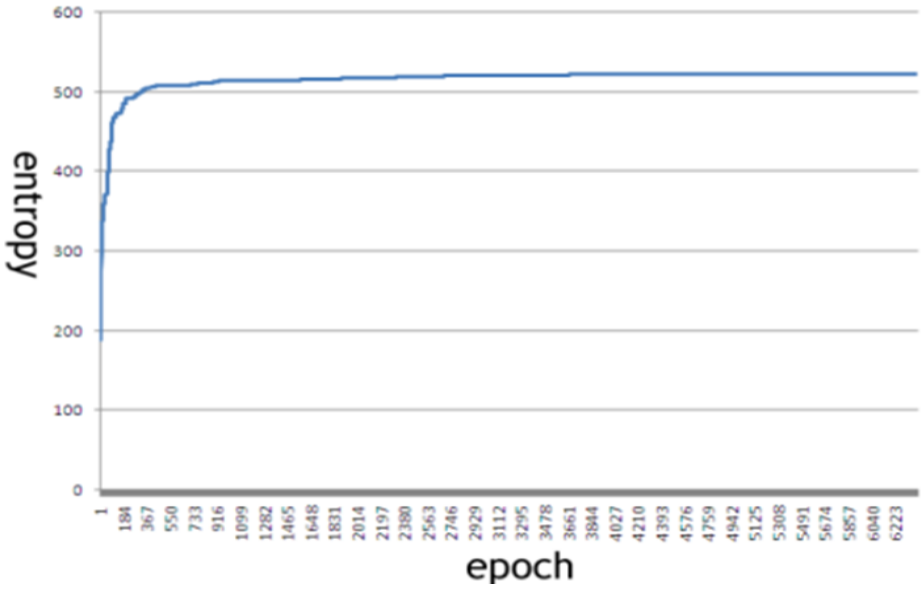

Figure 4.14 – convergence of search for high entropy local feature combinations with team '+1'

Five thousand randomly selected segments of play were used in conjunction with the individual local feature discovery and local feature tree generation algorithms described in section 3.9. This resulted in one local feature tree per team, each tree being 10 levels deep with 311 unique features in the '-1' team tree, 327 unique features in the '+1' tree.

4.5.2.3 Optimal spatial prototypes discovery

A random sample of five thousand player positions together with a fidelity threshold of 0.07 (pitch normalised, approximately 4m) was used with the MDL model selection algorithm for the optimal number of spatial prototypes between 1 and 255 (as described in section 3.10.1). The optimal number of spatial prototypes was 112 (Figure 4.15), and the resultant spatial distribution of prototypes along with an overlaid Voronoi diagram is shown in Figure 4.16. The prototypes are roughly mirror symmetrical (see Figure 4.17) about two orthogonal lines with their origins at the centre of the pitch. The symmetry across the vertical red line is due to the presence of the two opposing teams, averaged across the permitted team formations (such as 4-4-2, 5-3-2 etc).

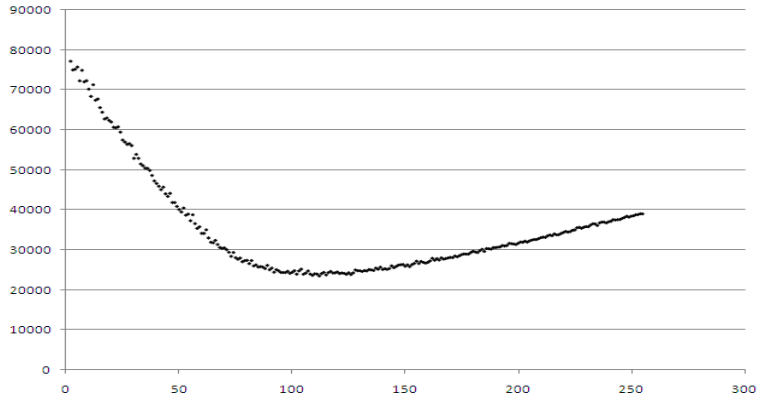

Figure 4.15 – Locating the optimal MDL derived values for spatial prototypes

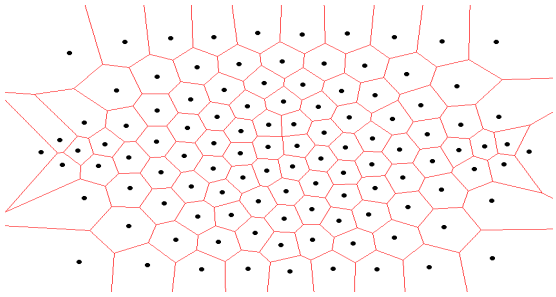

Figure 4.16 – Optimal spatial prototypes based on player movement

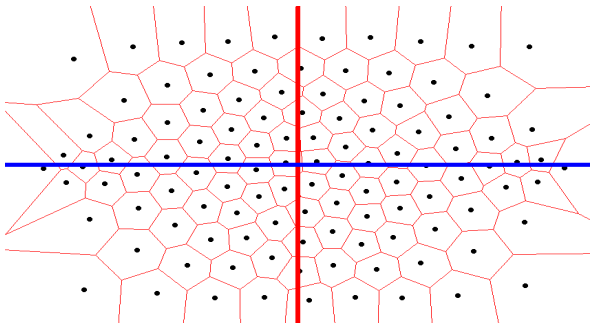

Figure 4.17 – Approximate mirror symmetry evident across two orthogonal lines originating at the centre of the pitch

4.5.2.4 Optimal gross/fine player archetypes discovery

The mean x position of each available player was agglomeratively clustered, the players clustering into the four gross archetypes shown in Figure 4.18, with (a) being the goalkeepers, (b) being the defenders, (c) were the midfielders and (d) were the attackers. A random sample of five thousand positions was taken from the

trajectories of each available player. Together with a fidelity threshold of 0.07 (pitch normalised, approximately 4m) this data was used with the MDL model selection algorithm (as described in section 3.10.2) for the optimal number of player prototypes. Agglomerative clustering was used to generate the average player models throughout the modelling process, as this produced predictable, deterministic clusters. A parameter space of spatial prototypes ranging from 100 to 200, number of player models ranging from 22 to 44, and player model length ranging from 1 to 50 was used. This space was searched using the stochastic search method Simulated Annealing (SA) [120] (as it was computationally prohibitive to brute force search the whole space). A total of five runs of SA, each starting at a random point in the space, were run with a maximum number of 1000 steps for each run. In an addition to the standard SA algorithm, each point in the space visited during the five runs was recorded in a set alongside its energy level (which in this case is the cost in bits), this record also functioned as a cache if a point was revisited, but otherwise did not influence the SA search. At the end of the five runs, the resultant set members were sorted by their energy level, the top ten lowest energy levels being recorded, and the lowest energy point being selected as the optimal solution.

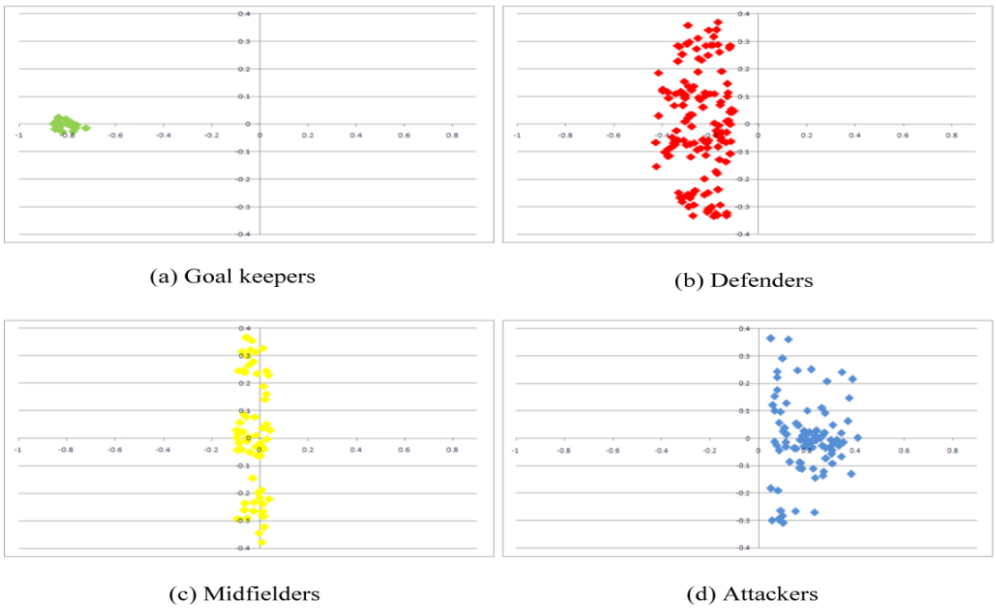

Figure 4.18- Gross player archetypes discovered by clustering

The optimal number of player models emerged as 22 with a model length of 30, on a backdrop of 117 spatial prototypes. Figure 4.19 shows three such discovered fine player archetypes (with only the most significant 16 positions in the model shown). A graphical depiction of all fine player archetypes found is available in section 7.6. The top ten results from the stochastic search are shown in Table 4.1.

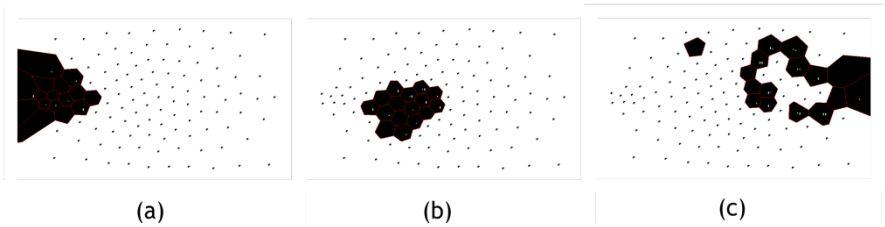

Figure 4.19 – 3 fine archetypes which approximately map to (a) a goalkeeper, (b) a defender, (c) an attacker

Spatial prototypes	Player models	Model length	Cost in bits
117	22	30	691274
113	22	31	697066
122	22	28	697994
104	23	32	699503
111	24	31	699796
104	22	31	700298
121	24	28	700628
116	22	29	700842
113	22	28	700906
110	22	32	701482

Table 4.1 – MDL stochastic search for fine player archetypes (top ten smallest models + data)

4.5.3 Experimental design

Sixty-three matches of trajectory and event data were obtained from ProZone and pre-processed (see section 4.5.2 for details). Each of the sixty three pre-processed matches were indexed at intervals of 5 seconds, with a default SOP length of 10s, starting from the beginning of each half, using each of the competing index schemes (resulting in eight versions of SOP indexes). The choice of a default SOP length of 10s is justified as follows. Firstly it ensures that the indexing process for the experiment is uniform and unbiased, both in the generation of the indices and in the eventual rating of the SOPs by the experimental subjects (for instance it is conceivable that subjects may be better at judging similarity over shorter rather than longer SOPs). Secondly, although the general indexing model as introduced in sections 3.2 and 3.3 has no intrinsic notion of time/duration of player movement, and most of the proposed indexing systems likewise have no intrinsic time/duration constraints, the semantically augmented ball trajectory indexing system (see sections

3.10 and 7.14 for details) does have a notion of duration in its RM and this is used to compare semantically augmented trajectories in a pair-wise fashion. The use of a standard SOP length therefore ensures no bias during the experiment for or against the ball indexing system. Thirdly, for the purposes of the experiment, the SOP length was chosen specifically to be 10s because the author perceived this to be roughly the amount of time most set plays such as corners, throw-ins, free kicks take either to terminate or to transform into free play.

4.5.3.1 Training context nets/obtaining PCA projections

For each of the eight individual indexing systems, ten thousand indexed SOPs were randomly sampled, and their PIs collected. These were used to train an auto-associative NN and to fit a PCA projection in order to produce compressed context representation in the 2D plane. Table 4.2 shows the extent of variability preserved (or information lost) from the original PIs by using PCA to create the 2D contexts[28]. The PIs which were the most successfully compressed where those that are essentially bit-fields (HTI and HTTI), the least successful was the clique indexing PI (CI) which is an integer vector based on the distribution of clique sizes within a team. Table 4.2 also shows the final convergent mean squared error (MSE) for each of the trained auto-associative NNs used to generate the alternative 2D compressions. Although the MSE is a more opaque metric than the PCA variability value to illuminate information loss due to compression, a lower bound of an MSE of 0.0 would indicate a perfect lossless compression, and increasing values above 0.0 indicate increasing information loss due to incomplete auto-associative modelling of the input vector. Interestingly even though the PI of the CI context exhibits the most information loss, the 2D projections of both the PCA and NN compressions as show in Figure 4.20 would seem to indicate that the resultant two variables have very little correlation with each other (a positive or negative correlation would show evidence of a line grouping of the variables). The low correlation will ensure that the 2D representation is making effective use of both of the NN input nodes allocated to it, albeit with the associated information loss. Section 7.4 shows the remainder of the 2D NN and PCA projections, with most of them showing little or no correlation between the compressed variables.

---

[28] Where ACI=Augmented cliques, AFI=player trajectories, BI=ball trajectories, CI=cliques, HPI=team mass multi-resolution 2D histograms, HTI=flat collection of high entropy features, HTTI=tree collection of high entropy local features, PI=team mass 2D histogram

PI Type	2D variability preserved	Converged MSE
CI	18.61%	0.874
PI	21.38%	0.819
HPI	25.09%	0.785
HTI	50.82%	0.449
HTTI	87.13%	0.282
BI	70.45%	0.343
AFI	36.83%	0.576
ACI	62.39%	0.327

Table 4.2 – Variability preserved in the compressed 2D context by PCA and convergent MSE of the auto-associative NN by indexing system

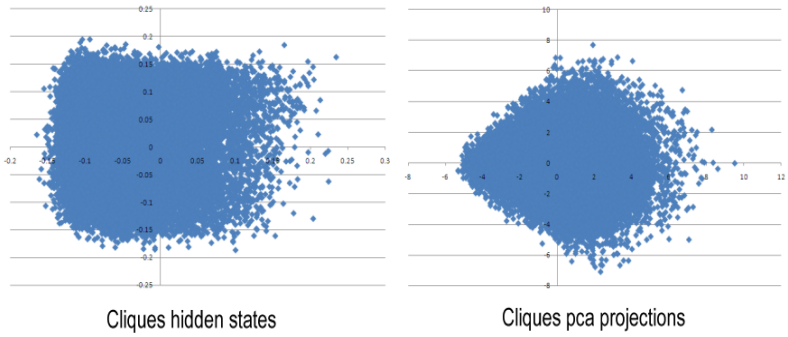

Figure 4.20 – compressed 2D context projections for cliques via NN (hidden states) and PCA projection

4.5.3.2 Providing bootstrapping ground truth for network training

As previously stated in section 3.3.1, to the best knowledge of the author, there exists no body of data that provides similarity ratings between segments of play/team movements/set pieces within football matches (and specifically within the ProZone data). There also isn't a pre-existing objective SOP similarity function to compare SOP pairs (if there were then this would be the obvious choice for an indexing system). Subsequently any similarity ratings which are required to initially train the respective indexing systems have to be generated in a bespoke manner by

an ‘expert’ in the field. As the author has spent some considerable time immersed in the study of SOP indexing and similarities, it did not seem an unreasonable step for the author to generate this initial body of bootstrap similarity ground truth. Since all the indexing systems will be trained using the same supplied similarity data (albeit from their own internal indexing perspective/representation), any potential biases within the training data will be equally shared between indexing systems. There will be no commonality between the set of SOPs used to generate the bootstrap similarities, and the set of SOPs the experimental subjects rate, so only generalised knowledge learnt from the bootstrap training data will be tested, not the recall of specific SOP similarity ratings. It is hoped that if the indexing work is pursued beyond this thesis, then an iterative process can be put it place where the experimental ratings data is folded back into retraining the indexing systems, which can then be further tested, generating even more ratings data and hence incrementally improving indexing performance. This being said, prior to beginning the experiment, the author rated 1425 pairs of SOPs, using the subjective Likert rating scale defined in Table 4.3. These pairs were divided approximately into the four cases covered by each subjective similarity category of *very unsimilar / quite unsimilar / quite similar / very similar* (in the subjective view of the author).

As the SOPs subjectively rated by the author were not the result of a direct query, the relaxation/similarity information was initially missing. This information (along with the compressed context) is required for training the NN ground truth estimators. However, it can be generated, as per section 4.4.1 (as can the PIs for the compressed context) as both SOPREFs are available.

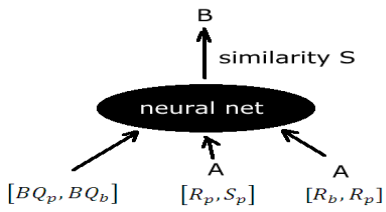

Figure 4.21 – filling in missing information for bootstrap similarity ratings

Consider two such SOPs, A and B as shown in Figure 4.22. Since similarity is a symmetric operation, any similarity attributed from A to B can be mirrored and applied from B to A. In this way the number of training cases was double by using the symmetric nature of similarity (see Figure 4.23)

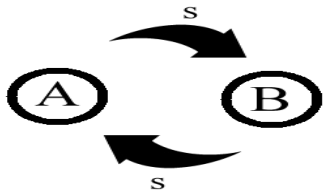

Figure 4.22 – symmetric similarity between two compared SOPs

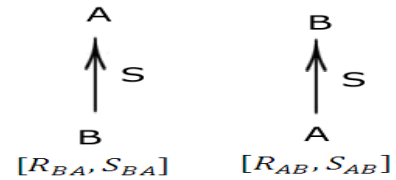

Figure 4.23 – mirroring similarity from B->A onto A->B

These initial ratings (2850 with similarity mirroring) were used to train the estimated ground truth NNs within each of the twenty one variants of the combined indexing systems (seven player indexer + ball indexer; seven player indexer + ball indexer with neural compressed context; seven player indexer + ball indexer with PCA compressed context).

4.5.3.3 Obtaining SOP similarity ratings from experimental subjects

Twenty seed segments of play were selected from the available matches, which broadly covered the different aspects of football play (corner, kick-off, goal kick, throw-in, direct free kick, indirect free kick, and segments from moving play). These seed segments were used as queries by example for each of the eight underlying indexing schemes, and the top ten results from each index query were recorded. Ten segments were selected at random for each seed to act as a control. Thus ninety results were associated with each seed, and in total eighteen hundred results were recorded. The twenty-one variants of the index combining system were used to combine the query results from their associated underlying indexing systems; resulting in twenty-one ranked result lists (whose length varied from ten to twenty depending on result list overlap, however only the top ten results from each system were evaluated).

An experimental application was created (see Figure 4.24) which was able to display two segments of play simultaneously in two adjoining display panes. The

segments of play were shown graphically from an overhead perspective, with opposing teams clearly delineated by colour (red and green teams). The segments of play were synchronised temporally, and could be played/stopped and fast-forwarded/reversed by use of on screen controls. The eighteen hundred results and their associated seeds were randomly sorted, and then sequentially displayed to the human subject in a blind fashion (i.e. no indication was given as to whether a particular pane contained a seed or a result, and the seed/result was randomly assigned to the left or right segment pane for each rating).

Figure 4.24 – SOP similarity evaluation application which enables the user to view two SOPs and then submit a similarity rating for the two SOPs

After viewing the two segments, the subject was invited to rate the similarity of the two segments. The similarity rating was a forced 4-point Likert scale[29], whose available options are enumerated in Table 4.3. This is a *forced* Likert scale because it removes the central non-committal answer of *don't know/unsure* and *forces* the rater to give a definite opinion on the similarity. Likert scales fall within the ordinal level of measurement; the response categories have a rank order, but the intervals between categories cannot be assumed to be equal. Common descriptive statistics such as the mean and standard deviation are inappropriate because of the discrete, ordinal nature of the data, as are common parametric analysis methods such as t-test, ANOVA and numerical regression. In their place, the median may be used instead of the mean to give a measure of the central tendency, and non-parametric analysis

[29] Likert scales are frequently used in questionnaires to elicit responses which cover a subjective qualitative scale.

methods such as Chi-Squared, and Spearman's rho (amongst others) may be used to further compare/analyse the data.

Score	Meaning
0.0	Very unsimilar
0.33	Quite unsimilar
0.67	Quite similar
1.0	Very similar

Table 4.3 – Four point Likert Similarity scale covering the interval [0,1] with associated semantic meaning

After rating one pair of segments, the rating score is associated with the pair and recorded, and the next randomly selected pair is shown to the subject. After all SOP pairs have been displayed and rated at least once, the 'to be rated' list is repopulated and randomised, and then rating continues as normal.

4.5.4 Results

A total of sixteen volunteers each submitted ratings for approximately one hour, in total generating 2593 ratings. This covered all of the test cases once[30], with an overflow of 631 ratings (uniformly randomised across all test cases). After each rating session, the volunteer was asked to attempt to write in English what constituted 'similarity' with respect to segments of play. The replies are reproduced in section 7.3.

A summary of the experimental ratings appears in Table 4.4. The sixteen raters can be clustered into three groups via their median rating given, with the rating of quite unsimilar (0.33) giving the largest cluster. There exists a surprising range in the number of ratings given during the experimental sessions, all of which extended for approximately one hour. The overall distribution of ratings submitted during all the experimental sessions is given in Table 4.5 (where LS1=very unsimilar(0.0), LS2=quite unsimilar(0.33), LS3=quite similar(0.66), LS4=very similar(1.0)). The

[30] with duplicates results from differing indexing schemes factored in

most common rating given is quite unsimilar, matching with the largest cluster of individual rater median ratings.

Rater	Ratings	Median
1	238	0.33
2	177	0.33
3	183	0.67
4	119	0
5	109	0
6	283	0.33
7	73	0.33
8	80	0.33
9	123	0.33
10	154	0.33
11	266	0
12	100	0.33
13	129	0.33
14	84	0.33
15	133	0.33
16	342	0.33

Table 4.4 – Summary of experimental data giving ratings and median similarity score per rater

LS1	LS2	LS3	LS4
0.345	0.38	0.209	0.066

Table 4.5 – Distribution of Likert ratings over entire experiment

In order to determine if the overall distribution of ratings has captured any general feature of raters, a series of Chi-Square Tests was performed to test whether the distribution of ratings for each rater (summarised in Table 4.6) was independent of the overall distribution of ratings (Table 4.7).

Rater	Ratings	LS1	LS2	LS3	LS4
1	238	0.303	0.294	0.29	0.113
2	177	0.169	0.373	0.345	0.113
3	183	0.169	0.301	0.372	0.158
4	119	0.63	0.319	0.05	0
5	109	0.615	0.303	0.073	0.009
6	283	0.244	0.597	0.141	0.018
7	73	0.342	0.233	0.247	0.178
8	80	0.437	0.375	0.15	0.038
9	123	0.244	0.276	0.26	0.22
10	154	0.325	0.37	0.286	0.019
11	266	0.647	0.32	0.023	0.011
12	100	0.19	0.46	0.21	0.14
13	129	0.426	0.209	0.341	0.023
14	84	0.298	0.238	0.274	0.19
15	133	0.414	0.248	0.308	0.03
16	342	0.249	0.602	0.14	0.009

Table 4.6 – Summary of experimental rating distributions over the four interval Likert scale

The results of the Chi Square tests are summarised in Table 4.7 and Table 4.8. They show that all raters apart from rater number eight and rater number four are statistically independent of the overall ratings distribution. A series of 256 Chi Square tests were performed between the individual rater distributions in an attempt to discover any similar clusters of raters. The full results table is too cumbersome to include within this thesis, but the raters which could not be shown to be significantly independent of each other (and therefore possibly having a shared subjective view of football similarity) are shown in Table 4.9.

	R01	R02	R03	R05	R06	R07	R08
Chi-Square	21.438	37.329	66.251	39.612	59.738	18.547	4.306
df	3	3	3	3	3	3	3
Asymp. Sig.	.000	.000	.000	.000	.000	.000	.230

Table 4.7 – Chi square test against overall distribution of Likert values (R01 – R08). Chi square test not possible for rater R04 as this rater did not submit any very similar (LS4) ratings.

	R09	R10	R11	R12	R13	R14	R15	R16
Chi-Square	52.575	9.610	129.009	16.945	26.706	26.406	16.770	78.398
Df	3	3	3	3	3	3	3	3
Asymp. Sig.	.000	.022	.000	.001	.000	.000	.001	.000

Table 4.8 – Chi square test against overall distribution of Likert values (R09 – R16)

Cluster	Raters
1	1,7,9,14
2	2,3
3	6,16
4	13,15

Table 4.9 – Clusters of similar raters w.r.t. their ratings distributions

Table 4.10 gives the median ratings assign to the top ten results supplied by the competing dual-indexing systems. Unfortunately, apart from demonstrating that all of the competing indexing systems are more effective than the random control, the results do not allow any additional comparisons to be made between the competing indexing systems.

Rank correlation is a more sophisticated class of non-parametric analysis methods that can be used to compare the ranking order of two identically sized ordered lists with a one-to-one mapping between elements of each list. The output of the dual indexing systems has a ranking score (the estimated similarity) with each search result, and as such when coupled with the associated user supplied similarity rating (which can also be used to rank) the resultant dataset is suitable for analysis by rank correlation methods. Two rank correlation methods were used[31], Kendall Tau –b and Spearman's rho. Both methods produce results in the range -1.0 to +1.0, where +1.0 indicates perfect correlation between the ordered lists, -1.0 indicates perfect inverse correlation and 0.0 indicates no correlation whatsoever. Each method produces an associated significance value with the correlation score, giving an indication of the probability that the correlation measured is a random effect.

Table 4.11[32] contains the both the Kendall Tau-b and the Spearman's Rho correlation scores for the competing indexing systems (and the random control) together with their associated significance levels. By convention, a significance level

[31] Testing was accomplished via SPSS as both methods were easily accessible.

[32] Where KTB=Kendall Tau-b correlation, KSIG=Kendall Tau-b significance level, SMR=Spearman's Rho correlation, SSIG=Spearman's Rho significance level

of at least 0.05 is usually required before a result can be viewed as significant. Correspondingly, two indexing systems can be said to have exhibited a weak, but significant correlation with the user similarity ratings.

Type	MEDIAN
PI_BI	0.33
AFI_BI	0.33
PCA_AFI_BI	0.33
CI_BI	0.33
NN_CI_BI	0.33
PCA_CI_BI	0.33
NN_PI_BI	0.33
PCA_PI_BI	0.33
HPI_BI	0.33
PCA_HPI_BI	0.33
NN_HTI_BI	0.33
HTI_BI	0.33
NN_HTTI_BI	0.33
HTTI_BI	0.33
NN_HPI_BI	0.33
PCA_HTI_BI	0.33
NN_AFI_BI	0.33
ACI_BI	0.33
NN_ACI_BI	0.33
PCA_ACI_BI	0.33
PCA_HTTI_BI	0.33
RND	0

Table 4.10 – Median ratings for underlying indexing schemes. The PCA prefix denotes a dual indexing system using Principal Component Analysis as the means to derive the query context, and the NN prefix denotes the use of an auto-associative neural network to derive the query context.

Overall, even though two indexing systems do provide a weakly significant correlation, the results are somewhat disappointing. In hindsight, there were a number of problems with the experiment that lead to the limited results achieved, and which should be rectified if the experiment is repeated. The first problem, and probably the most serious, is the use of a Likert scale, particularly one with such a limited number of responses. The ordinal nature of the Likert scale requires the use of non-parametric analysis methods, but with the number of responses limited to four, the median and rank-correlation methods have difficulty producing clear results. A repeat of the experiment should ideally opt for a continuous interval value for similarity (say between 0.0 and 1.0). The second problem with the experiment is

that it would have benefitted from a larger pool of raters. Table 4.6 demonstrates the considerable variability attached to the subjective similarity ratings, even over only sixteen volunteers using a coarse four-point Likert scale. Working with a larger pool of ratings should help lessen the overall subjective nature of the ratings (especially if the experiment opts to use a continuous interval value for similarity).

TYPE	KTB	KSIG	SMR	SSIG
PCA_HTI_BI	0.115	0.013	0.158	0.01
NN_HPI_BI	0.092	0.053	0.125	0.047
ACI_BI	0.085	0.07	0.108	0.083
PCA_ACI_BI	0.076	0.105	0.098	0.112
HPI_BI	0.071	0.129	0.095	0.127
NN_ACI_BI	0.071	0.13	0.093	0.133
PCA_PI_BI	0.067	0.152	0.086	0.168
NN_CI_BI	0.052	0.265	0.072	0.247
PCA_HPI_BI	0.052	0.265	0.068	0.272
AFI_BI	0.042	0.366	0.058	0.35
NN_HTI_BI	0.035	0.45	0.045	0.467
PCA_HTTI_BI	0.033	0.484	0.044	0.48
PCA_CI_BI	0.03	0.519	0.04	0.52
CI_BI	0.024	0.615	0.032	0.612
NN_HTTI_BI	0.021	0.647	0.03	0.626
NN_PI_BI	0.014	0.769	0.017	0.786
RND	0.011	0.801	0.018	0.743
PI_BI	0.006	0.896	0.007	0.913
PCA_AFI_BI	-0.005	0.913	-0.007	0.906
HTI_BI	-0.019	0.679	-0.026	0.673
NN_AFI_BI	-0.019	0.683	-0.024	0.701
HTTI_BI	-0.062	0.185	-0.083	0.181

Table 4.11 – Rank correlations scores for competing indexing systems

4.6 Summary

This chapter dealt with aggregating results from a combined system of a ball indexer and a player indexer. The general problems encountered when fusing two results lists together were covered, and a score based aggregation approach was decided upon (rather than a rank based approach). This chapter also detailed the evaluation experiment for both the individual indexing systems, and the composite

combined indexing system. The best performing combined indexing systems were identified, and enhancements to the experiment were recommended should it be required to be repeated in the future.

5 Behaviour modelling

This chapter presents a novel hierarchical behaviour model for football players, combining behaviour models at three levels of team, cliques of players associated via proximity/velocity direction, and individual players themselves. The modelling process is evaluated by comparison of generated models with real player behaviour and that of simple experimental control models.

5.1 Introduction

The behaviour exhibited by players within a football match is both rich and complex. Players do not operate in a vacuum, but make decisions based on the disposition of their team mates, the disposition of the players of the opposing team, and taking into account the global state of the game (location of ball, current score, time remaining etc).

The second strand of this PhD involves research into how to capture and model the behaviour of real football players working within a team and against an opposing team. The raw behaviour is represented by a large amount of data representing player positions, ball location (interpolated) and player events (such as touching the ball, passing, shooting etc), all indexed by time. Successful behaviour modelling should result in realistic movement and actions, both at the level of the individual player and at the team level, and produce at least a good approximation of actual team play.

5.2 Required features of a player behaviour model

As stated in the introduction, football players operate cooperatively as a member of a team, competing against another set of players. Any behaviour model of players should be able to capture this notion of a player being influenced by groups of other players (be they team mates or opponents).

The notion of behaviour of players in groups suggests that there exists a higher level of behavioural organisation than merely the player level. A behavioural modelling approach which utilises such higher level structures is known as hierarchical behaviour modelling. [92] utilises a hierarchical model to model human body movements; initially modelling small atomic movements, before modelling more complex movements via combinations of the previously modelled atomic movements.

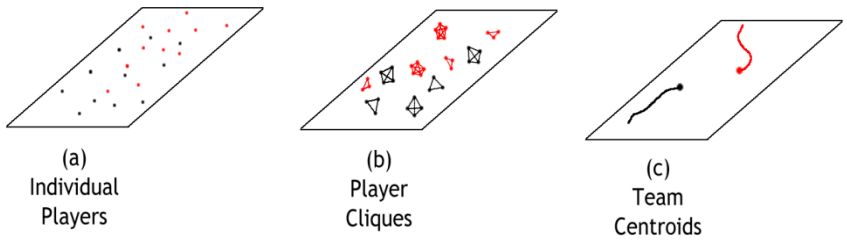

Figure 5.1 – The three hierarchical levels of behaviour from the most concrete players (a), to the more abstract cliques (b), to the most abstract team centroids (c).

To utilise a hierarchical behaviour model within a football context, distinct levels of behaviour must be identified with a team. The obvious level of behaviour is located at the player level (Figure 5.1 (a)). A second level of behaviour is that displayed by groups of players (Figure 5.1 (b)). As demonstrated in Chapter 3, the use of cliques is a good way of extracting groups of associated players from the raw data, allowing players to be members of multiple cliques simultaneously. In the case of behaviour modelling, we not only want intrateam cliques, but also interteam cliques as player behaviour is certainly influenced by the opposing team players as well as teammates. A third level of behaviour is exhibited by the team as a whole (Figure 5.1 (c) as represented by the team centroid).

The team centroid is defined as the mean position of the team. As such, its behaviour is driven by the mean movement of the team. Team formations are generally mirror symmetrical across the pitch (grey line in Figure 5.2), and tend to fill the entire width of the pitch, so centroid movement in this direction is limited. In contrast, in the orthogonal direction they are asymmetrically arranged into defence/midfield/attack (red line in Figure 5.2), and depending on the flow of the game (attacking or defending), the team formation can move backwards or forward

in a coordinated fashion. Hence, the centroid captures the general attacking/defending flow of the game, but will be less sensitive to movement in the orthogonal direction. The team centroid will also be less sensitive to some movement of team formations that give greater importance to few or even a single player. As an example the off the ball movement of the single forward in the 5-4-1 will have little effect on the team centroid.

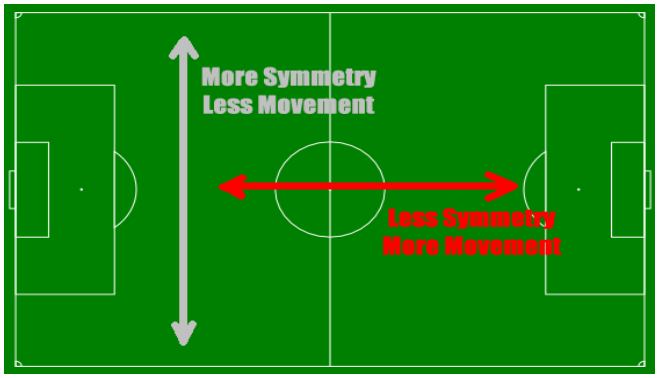

Figure 5.2 – Team centroid dynamics exhibit an asymmetrical preference for movement perpendicular to the goal lines

Models may either have some form of memory of previous states, or possess no memory of previous states. Which approach is most useful for the hierarchical player models had to be decided. Figure 5.3 shows the clique ABC in two different contexts of approaching players. If the players within the clique exhibit some form of shared behaviour, and this behaviour model is has no memory, then the players in scenarios (a) and (b) will exhibit the same behaviour (or exhibit the same behaviour probabilities if the model is non-deterministic).

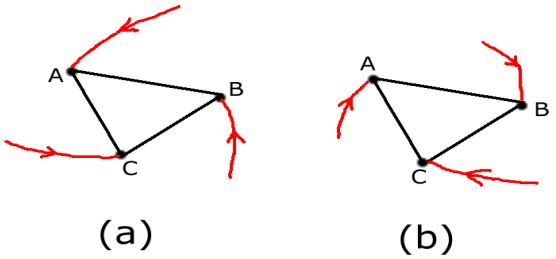

Figure 5.3 – Two identical spatial configurations with differing histories

It is the view of the author that this is not the case in real football, and as such, some portion of the history of interacting players/cliques/centroids must be admitted

into the modelling scheme. Auto-regression [98] is a modelling technique whereby the past states of a system can be used to predict future states. In order to include past states of other entities, the auto-regression model can be widened to the Nonlinear AutoRegressive eXogenous model (NARX), which is often implemented using either SVMs or NNs.

5.3 General approach for behaviour modelling of players

A hierarchical modelling approach was taken, where the behaviour of a player is represented at three levels (Figure 5.4). The history of players, as represented by their recent trajectory is used as context for the behaviour model. The highest level is that of the behaviour of the entire team. The behaviour of the team (as represented by its centroid) is influenced by its own recent movement, the recent movement of the opposing team centroid and the ball, as well as the general context of the match at the current time (such as which team is in possession, which team is winning, how much time is left to play etc). The behaviour of the team exerts an influence on the behaviour models below it in the hierarchy (i.e. the clique and player models).

The intermediate level of behaviour model for a player is a collection of clique behaviour models. These models simulate how a player operates within each clique of a given type and size. The behaviour of each clique model is influenced by the recent movement history of the player, the recent movement history and team affiliation of the other clique members, the recent movement history of the ball and which team is currently in possession of it. The behaviour of cliques is also influenced by how the entire team will likely move in the next time interval (i.e. it is influenced by the team centroid model at the higher model level).

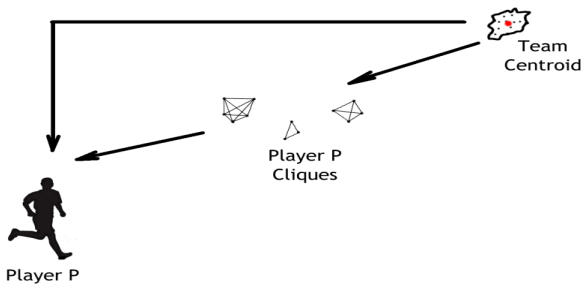

Figure 5.4 – The hierarchical player behaviour model will include movement influences from the more abstract player clique and team centroid levels

The lowest level of behaviour is that which is exhibited by the player himself. This is influenced by the player's recent movement history, the recent movement history of the ball and which team is currently in possession of it. The player model is also influenced by the likely next move of the team as a whole, and by the likely next moves of each clique in which the player is a member of at the current time.

5.4 Implementation

The behaviour models will attempt to predict the next move of the player (or team centroid) for the next time instance (0.1s into the future). Movement from the current position in the behaviour models will be represented as relative polar coordinates from the current position rather than as the more traditional Cartesian displacement vector.

The polar coordinates of the move are quantised. The angular component of the polar coordinate θ, is quantised into sixteen equidistant angular increments (each separated by 22.5 degrees). The polar distance r is quantised into ten increments, each quantisation representing 0.1 metres distance (meaning the highest distance quantisation will correspond to sprinting at approx 9m/s or above). The choice of sixteen angular components and ten distance components is a compromise by the author between player movement fidelity and model complexity; allowing the player a reasonably realistic fidelity of movement. The two quantised polar components are mapped onto a twenty-six component vector (Figure 5.5), the first sixteen components representing the quantised angular direction (D), and the last ten components representing the quantised distance covered (S). This twenty-six component vector represents the movement that the model is attempting to predict.

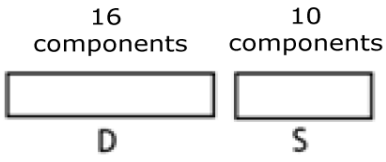

Figure 5.5 – mapping quantised direction and speed onto vectors of length 16 and 10

Each behavioural model will use a NN to map its input vector (see later sections for details) to the predicted movement output vector. The NN will

effectively try to learn to classify each move as a combination of one chosen direction (from D) and one chosen distance (from S). To enable this classification to be done in a probabilistic manner, each NN will use two softmax output functions [121], covering the first sixteen components representing the direction (D), and then last ten components representing the distance moved (S). The use of softmax activation functions over these two sub-sections of the output vector means that each subsection will become a separate probability distribution (each subsection will sum to 1.0).

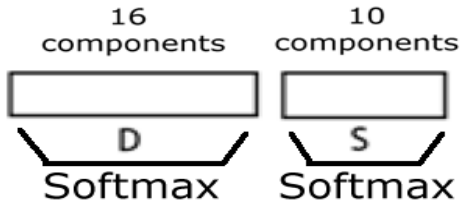

Figure 5.6 – softmax probability distributions over the D and S result in $\sum_{i=0}^{i=15} D_i \approx 1.0$ *and* $\sum_{i=0}^{i=9} S_i \approx 1.0$

Where trajectory histories of players/ball at time current time t, extending n instances into the past[33] are used as the input to NNs, the original trajectory T (5.1) will be transformed into T^p via (5.2). The transformed trajectory T^p maintains T's current position, but all of T 's previous positions are relative to its current position.

$$T = [(X_t, Y_t), (X_{t-1}, Y_{t-1}), \dots, (X_{t-n}, Y_{t-n})] \tag{5.3}$$

$$T^p = [(X_t, Y_t), (X_{t-1} - X_t, Y_{t-1} - Y_t), \dots, (X_{t-n} - X_t, Y_{t-n} - Y_t)] \tag{5.4}$$

Player cliques come in four main types:

[33] So T will have $n + 1$ members in total (1 current position + n historical positions)

- **Intrateam proximity cliques** – players of the same team, grouped together by spatial proximity
- **Intrateam direction of movement cliques** – players of the same team, group together by the similarity of the direction of motion component of their instantaneous velocity. Note this is different from the direction of motion cliques in chapter 3, which used the future positions of players to work out the mean direction of motion into the future. In this case, the future is inaccessible, so either the instantaneous velocity or the mean recent historical velocity may be used. Since the behaviour model is autoregressive in nature, the influence of past states should already have an influence on future behaviour, so the simpler option of instantaneous velocity was chosen
- **Interteam proximity cliques** – players of either team, grouped together by spatial proximity
- **Interteam direction of movement cliques** – players of either team grouped together by the similarity of the direction of motion component of their instantaneous velocity.

In addition to these four main types, cliques are also described by how many members are within them. Thus in total there are sixty-two possible clique types (ten for intrateam proximity/direction and twenty-one for interteam proximity/direction). Details of the NN for each level of the model follow in the next sections.

5.4.1 Team centroid

The team centroid behaviour model is shared by each member of the team it is modelling. In general, the input into the model covers the past movement of the team centroid, the past movement of the opposing team centroid, the past movement of the ball, and a collection of inputs that together represents the current match context. All input values are of a similar magnitude (+/-1.0 to +/- 10.0). The pruning method used in the training of the model will ensure that any irrelevant sections of the input vector will be discarded. The input/output vectors are:

INPUT

- Abstract mapping of team (A) ('+1' or '-1')
- Team Trajectory over past n time instances (T)
- Opposing team trajectory over past n time instances (O)
- Ball trajectory over past n time instances, and current team in possession ('+1' or '-1') (B)
- Game Context (C) composed of
    - score difference w.r.t. team (+/- value)
    - time from half (expressed at $\frac{current\ time\ in\ half\ as\ 0.1s\ increments}{27{,}000}$)
    - half (either 1 or 2)

OUTPUT

- Quantised future direction of team centroid (D)
- Quantised future speed of team centroid (S)

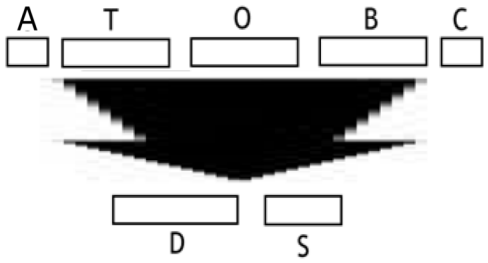

Figure 5.7 – neural network model for team centroid produces softmax probability distributions over D (movement direction) and S (movement speed)

5.4.2 Player cliques

Each player owns a collection of clique behaviour models, representing the types of cliques that the player is likely to be a member of. Each clique is described a combination of its type and it membership size. All input entries are of the same order of magnitude (+/- 1.0). The input/output vectors are:

INPUT

- Abstract mapping of team (A) ('+1' or '-1')
- Team centroid predicted movement (C) – This is the twenty-six component predicted movement vector output from the team centroid behaviour model.
- Clique centroid trajectory over past n time instances (P)
- Ball trajectory over past n time instances, and current team in possession ('+1' or '-1') (B)
- Clique members ($M_1 \ldots M_n$) covering
    - Team affiliation ('+1' or '-1')
    - Member trajectory over past n time instances

OUTPUT

- Quantised future direction of player P (D)
- Quantised future speed of player P (S)

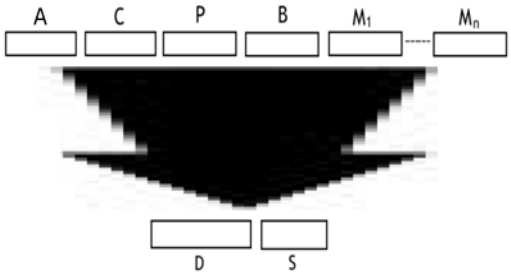

Figure 5.8 – neural network model for clique produces softmax probability distributions over D (movement direction) and S (movement speed)

5.4.3 Player

Each player has his own player level behaviour model. This model is influenced by the team centroid model and all clique models that the player is a member of at the time the player movement is predicted. Details of the input/output vectors are:

INPUT

- Team centroid predicted movement (C) – This is the twenty-six component predicted movement vector output from the team centroid behaviour model.
- Average cliques predicted movement (A) – This is a combination of all the twenty-six component predicted movement vector outputs from all the cliques models that the player is currently a member of at the time of the prediction. Each of the twenty-six components is summed across all clique outputs and divided by the number of active clique components. If the relative accuracy of the cliques models was known for each particular player configuration, then a more sophisticated weighted mean could be taken, However as this information is not currently available the uniform mean is the only justifiable approach.
- Player trajectory over past n time instances (P)
- Ball trajectory over past n time instances, and current team in possession ('+1' or '-1') (B)

OUTPUT

- Quantised future direction of player P (D)
- Quantised future speed of player P (S)

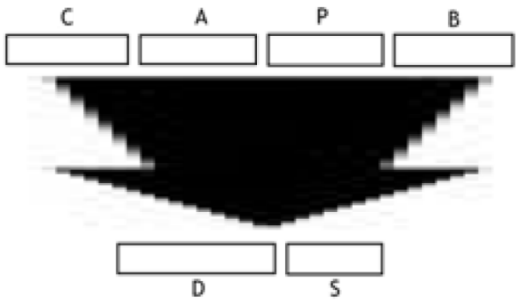

Figure 5.9 – neural network model for player produces softmax probability distributions over D (movement direction) and S (movement speed)

5.5 Generalisation/Feature selection

Selecting the correct topology for a NN is not a trivial task. Large networks will tend to overfit the available data, and small networks may not adequately capture the underlying relationships within the data. One approach to automatically selecting a good network topology is to initially train a deliberately large network, and then progressively prune irrelevant weights/nodes from it. A second advantage to pruning NNs, is that irrelevant input variables can be identified (by becoming disconnected from the network), and hence pruning becomes a method of automatically selecting a relevant subset of features from an initial larger set, possibly containing irrelevant or redundant information.

A method of network pruning based on orthogonal least squares [122] was located. This method compares favourably with other well known pruning methods such as Optimal Brain Damage [123] and Optimal Brain Surgeon [124], not only in parsimonious pruning, but in computational resources, as it works more efficiently on larger networks than either OBD or OBS. The OLS pruning approach was used in the training of the NN to produce network with better generalisation and reduced input vectors. An overview of the approach in [122] is given in the following section.

5.5.1 Overview of OLS pruning approach

5.5.1.1 Orthogonal Least Squares for linear regression

This section gives an overview of the Orthogonal Least Squares algorithm, and specifically how it is used by the pruning algorithm to assign importance to network weights. The standard form of a series of linear equations (over determined if $n > m$) is:

$$z(t) = \sum_{i=1}^{m} P_i(t)\theta_i \ \ (t = 1, \dots, n) \tag{5.5}$$

(5.5) can be written more succinctly in matrix form:

$$Z = P\theta \tag{5.6}$$

Given values for Z and P, to find the optimum values in the parameter matrix $\hat{\theta}$ that gives the best fit (i.e. the lowest sum of squared residuals) the normal equation (5.7) is used.

$$\hat{\theta} = (P^T P)^{-1} P^T Z \tag{5.8}$$

Solving (5.8) directly on computers can pose problems due to limitations of accuracy of number representation, but it can be solved indirectly. One method is the Cholesky decomposition (5.9).

$$P^T P = A^T D A \tag{5.10}$$

Using the decomposition given in (5.10) as a substitution in (5.8) allows the optimum parameter matrix $\hat{\theta}$ to be numerically calculated (5.11).

$$\hat{\theta} = (A^T D A)^{-1} P^T Z \tag{5.11}$$

The pruning algorithm builds on this approach to the Orthogonal Least Squares algorithm. Using matrix A from (5.10), (5.6) can be rewritten as (5.12).

$$Z = Bg \tag{5.12}$$

Where $B = PA^{-1}$ and $g = A\theta$

(5.12) can be rewritten, after a lengthy transformation (see [122] page 5459) as (5.13):

$$\frac{1}{n}Z^T Z = \sum_{i=1}^{m}\left[{g_i}^2 \frac{1}{n}{b_i}^T b_i\right] + \sigma_\varepsilon^2 \tag{5.13}$$

Here $\frac{1}{n}Z^T Z$ is the variance in the matrix Z, and thus the variance of the matrix Z is equal to the sum of the variances of the m terms of the linear equations set (plus the modelling error variance). This variance information is used by the pruning algorithm to determine the relative importance of weights during the pruning process (weights which transmit more variance are more important to the network).

5.5.1.2 Application of OLS to neural network pruning

Now consider either a hidden or an output node in a NN (R in Figure 5.10). The receiving node R aggregates the signals of the connected nodes $n_1 \dots n_4$ (and then non-linearly 'squashes' it). The aggregation process ($n_1 w_1 + n_2 w_2 + n_3 w_3 + n_4 w_4$) is a set of linear equations (see (5.5)). So given a set of n patterns, a NN can be trained on those n patterns. Once trained for each hidden/output node in the network, the equation $Z = P\theta$ represents the signals of each node (i.e. Z is the matrix of aggregated totals before non-linear squashing) over all input patterns P (values of $n_1 \dots n_4$ in Figure 5.10) given the weight matrix θ ($w_1 \dots w_4$ in Figure 5.10).

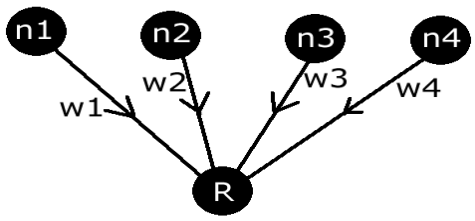

Figure 5.10 – Aggregating a neural signal in node R involves linearly summing the activation of each input node multiplied by the connecting weight, and then squashing the linear sum into a predefined range (usually either $[0, +1]$ *or* $[-1, +1]$)

Transforming (5.6) into (5.12) allows the variances associated with each weight in θ to be calculated via a further transformation into (5.13). For each layer in the network (ignoring the input layer), the total variance can be calculated via the summation of $\frac{1}{n}Z^TZ$ for each node in the layer. Then the relative importance of each weight (w.r.t. the variance it carries) arriving into the layer can be calculated.

The network is pruned by keeping only the weights which account for $X\%$ of the total variability exhibited by the each layer in the network[34]. Completely disconnected nodes can be removed from the network (so input nodes can be deleted). The training/pruning cycle continues until either no more pruning is possible, or the network has reached an acceptable level of generalisation. Not only does the resultant network exhibit better generalisation properties, but irrelevant input nodes are also removed (by elimination of all weights leading from them).

5.6 Evaluation

Two experiments were devised to test the efficacy of the behaviour modelling approach. The first experiment dealt with testing the next step (0.1 seconds into the future) predictions of each component of the behaviour model (centroid, cliques, player) against a random control. The second experiment examined how well simulated player trajectories compare to real player trajectories (given an identical starting configuration). A random walk simulation and a first-order linear predictor are included in the second experiment as controls.

5.6.1 Data acquisition and pre-processing

Twenty-seven matches of trajectory and event data were obtained from ProZone for the purposes of behavioural modelling. One team[35] (say team E) was present in each of the matches (the opposing team varied between matches). This constitutes approximately two thousand four hundred minutes of play, of which approximately 58% is active play (the other 42% being stoppages such as ball out of

[34] The paper recommends a high value for X such as 99.99%, thereby only eliminating the most irrelevant network weights in each round of pruning

[35] The identity of this premiership club cannot be divulged because of ProZone privacy concerns

play, fouls etc). The 42% of play which represented stoppages was rejected (as out of play behaviour is not being modelled), and the remaining 58% was segmented into uninterrupted sequences of play (i.e. the temporal boundaries of each sequence were recorded). Player positions and the ball trajectory were interpolated down to fidelity of 0.1s prior to any behaviour modelling (as described in section 4.5.1). The centroid of each team was also interpolated down to a temporal fidelity of 0.1s. for each relevant section of all twenty seven matches.

5.6.2 Optimal clique threshold discovery

In a similar fashion to section 4.5.2.1, the optimal threshold values for four types of cliques (intrateam proximity, intrateam direction, interteam proximity, and interteam direction) must be discovered before cliques can be extracted from the raw player data. A line-search was performed over the interval [0,2] for the proximity cases (normalised pitch units) and [0, 3.2] for the direction cases (radians). Figure 5.11 shows the results of the line search for the optimal threshold value in each case. Specifically: Interteam proximity = 0.51, Interteam direction= 0.48, Intrateam proximity=0.5, Intrateam direction=0.45.

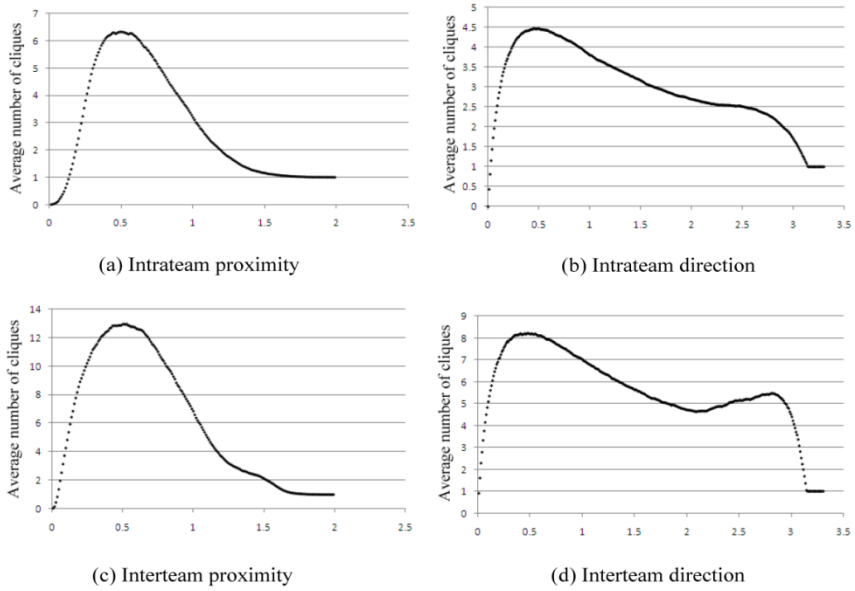

Figure 5.11 – locating the optimal threshold values for the four types of cliques

5.6.3 Team centroid behaviour model training

Twenty-one games were randomly chosen to provide training data for the NNs. The remaining six games were used as a source of testing data. Five thousand random temporal locations within the active sequences of the twenty-one games were chosen. At each of these temporal locations, training data was generated for the team centroid, the exact nature of the training data is given in section 5.4.1.

The initial topology of the NN trained was two fully connected hidden layers, each containing two hundred nodes. In informal tests, this size was found to be large enough to exhibit overfitting in data of the magnitude used in the experiment (thousands of examples), via training then validation. The initial overfitting is an important precondition of employing the training/pruning processes (if the network underfits the data, there is no scope to identify irrelevant weights, but no means to add extra weights/nodes). The NN was initially trained using this collected training set, and then pruned/retrained as per the algorithm in section 5.4. Two thousand random samples were taken from the remaining six games, and used to create testing data for the team centroid behaviour model. The format of the data was identical to the training data described above.

5.6.4 Clique and player behaviour models

Five players were preselected from team E, having good coverage over the whole twenty seven games. These players P_1, P_2, P_3, P_4, P_5 mapped to the approximate roles of {*attacker, defender, goalkeeper, defender, midfielder*} respectively. The same twenty-one games selected in section 5.6.3 were used to provide training data for the NNs. The remaining six games were used as a source of testing data.

5.6.4.1 Clique behaviour model training

For each player $\{P_1, P_2, P_3, P_4, P_5\}$ five thousand random temporal locations within the active sequences of the twenty-one games were chosen. At each of these temporal locations, the four distinct clique sets for the player in question were generated, resulting in a distribution over the five thousand samples of the most encountered clique types for the player in question. The most represented fifteen

clique types for each player were recorded (at maximum, it is possible that there will be less than fifteen in total).

For each of the represented cliques for each player $\{P_1, P_2, P_3, P_4, P_5\}$, it was calculated where in the matches temporally it occurred, and then five thousand random samples were taken from this distribution and used to provide training data for the clique. The exact nature of the training data is given in section 5.6.1. A large NN[36] for each clique was initially trained using this collected training set, and then pruned/retrained as per the algorithm in section 5.5.1. Two thousand random samples were taken from the remaining six games, and used to create testing data for the each of the clique behaviour models (for each of the players). The format of the data was identical to the training data described above.

5.6.4.2 Player behaviour model training

For each player $\{P_1, P_2, P_3, P_4, P_5\}$ five thousand random temporal locations within the active sequences of the twenty-one games were chosen. At each of these temporal locations the cliques which the player was a member of was recorded. Those cliques not in the top fifteen for the player were rejected. The exact nature of the training data is detailed in section 5.6.1. As before a large NN for each player was initially trained using this collected training set, and then pruned/retrained as per the algorithm in section 5.5.1 Two thousand random samples were taken from the remaining six games, and used to create testing data for the each of the player behaviour models. The format of the data was identical to the training data described above.

5.6.5 Experiment One Evaluation Setup

Once all NN behaviour models are trained (team centroid, cliques and player models), the models can be evaluated as follows. A random walk model (Figure 5.12) is available for each level of the behavioural model. The behaviour of the random walk model is very simple, for its next 'predicted' move it simply:

[36] Two hidden layers, 200 nodes per layer

- Randomly chooses one of the sixteen quantised movement directions, and assigns it 1.0 (the rest are 0.0)
- Randomly chooses one of the ten quantised speed increments and assigns it 1.0 (the rest are 0.0)

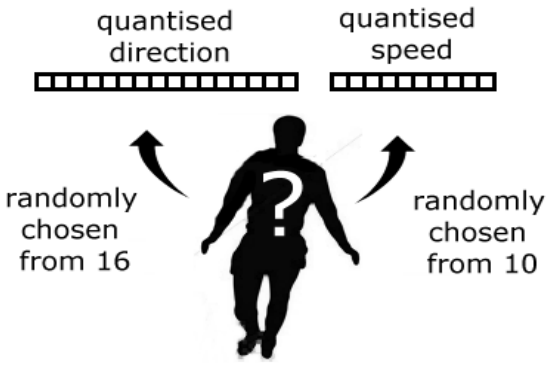

Figure 5.12 – The random walk model randomly selects one of the sixteen possible directions and one of the ten possible speeds at each simulated time step

For each of the components of the behavioural model, testing data exists which was extracted from the six games which were not used as a source of training data. Each block of testing data holds two thousand instances of correct input, output pairings as observed in the real data. The input from the test data can be fed through the relevant NN behaviour model to produce its predicted movement output, and the random walk model can produce its 'predicted' output. Thus for each of the two thousand test cases, there exist three movement 'predictions':

- The actual movement exhibited by the centroid/clique/player (A)
- The behaviour model prediction, from behaviour model B, for the movement of the centroid/clique/player (P)
- The random walk model 'prediction' for the movement of the centroid/clique/player (R)

The mean accuracy of prediction of the direction of movement, $\overline{D_B}$ given by behaviour model B, and the mean accuracy of the prediction of movement $\overline{D_R}$ given by the random walk model R are defined by equations (5.14) and (5.15)

respectively. The mean accuracy of prediction of the speed $\overline{S_B}$, given by behaviour model B and the mean accuracy of prediction of the speed $\overline{S_R}$, given by the random walk model R are defined by the equations (5.16) and (5.17) respectively. Finally, the accuracy of j^{th} component of the movement/speed prediction $\overline{B_J}$, of behaviour model B, and the mean accuracy of j^{th} component of the movement/speed prediction $\overline{R_J}$, of random walk model R are defined by the equations (5.18) and (5.19) respectively.

$$\overline{D_B} = \frac{1}{2000}\sum_{i=1}^{2000}\frac{1}{16}\sum_{j=1}^{16}A_{i,j}P_{i,j} \tag{5.14}$$

$$\overline{S_B} = \frac{1}{2000}\sum_{i=1}^{2000}\frac{1}{16}\sum_{j=17}^{26}A_{i,j}P_{i,j} \tag{5.15}$$

$$\overline{D_R} = \frac{1}{2000}\sum_{i=1}^{2000}\frac{1}{16}\sum_{j=1}^{16}A_{i,j}R_{i,j} \tag{5.16}$$

$$\overline{S_R} = \frac{1}{2000}\sum_{i=1}^{2000}\frac{1}{10}\sum_{j=17}^{26}A_{i,j}R_{i,j} \tag{5.17}$$

$$\overline{B_J} = \frac{1}{2000}\sum_{i=1}^{2000}A_{i,j}P_{i,j} \tag{5.18}$$

$$\overline{R_J} = \frac{1}{2000}\sum_{i=1}^{2000}A_{i,j}R_{i,j} \tag{5.19}$$

5.6.6 Experiment One Results

5.6.6.1 Team centroid model

For the single team centroid behavioural model of team E, $\overline{D_B}, \overline{S_B}, \overline{D_R}$ *and* $\overline{S_R}$ are shown in Figure 5.13, $\overline{B_J}$ *and* $\overline{R_J}$ covering the quantised direction (1…16) are shown in Figure 5.14 and $\overline{B_J}$ *and* $\overline{R_J}$ covering the quantised speed (17…26) are shown in Figure 5.15. In the three result sets, the behaviour model easily outperforms the random control, which gives predictably poor results in all circumstances. The behavioural model does well over the entire range of quantised directions, demonstrating that there are not directions in which the centroid is less predictable than others (at least not covering projections 0.1s into the future). Across the range of quantised speeds the model performs well on all but the slowest quantised speed. The most likely explanation for the increased unpredictability at low speed is that it occurs when a team is essentially static (rather than expanding in an isotropic manner), and as such one team member changing speed can unduly influence the centroid speed (but they do not have a preferred direction in which to accelerate to).

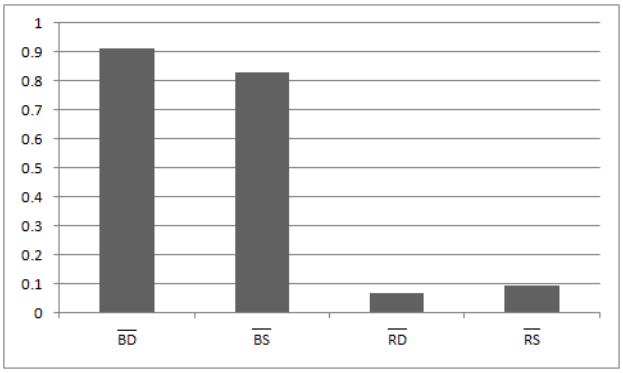

Figure 5.13 – $\overline{D_B}, \overline{S_B}, \overline{D_R}, \overline{S_R}$ for team centroid model

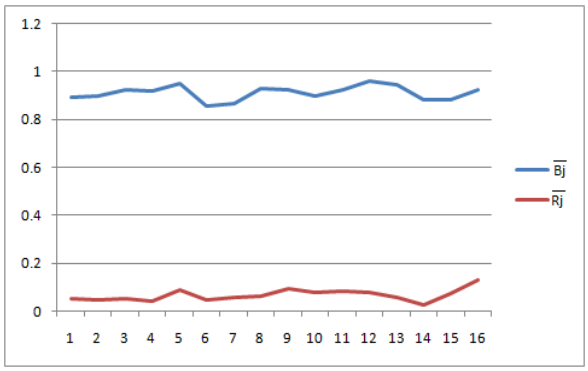

Figure 5.14 – $\overline{B_j}$, $\overline{R_j}$ for j=1…16 for team centroid model

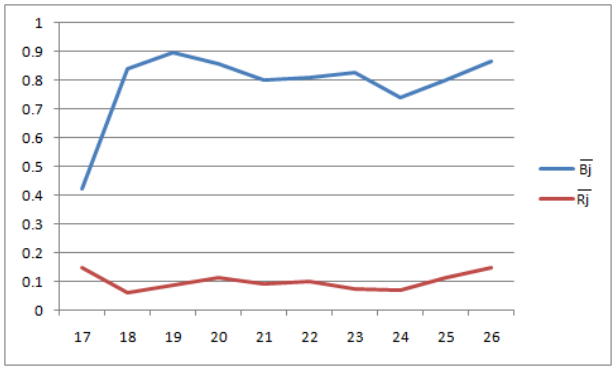

Figure 5.15 – $\overline{B_j}$, $\overline{R_j}$ for j=17…26 for team centroid model

5.6.6.2 Mean clique results

The number of clique models is large compared to the number of other models (60 for each player maximally). Therefore, only the mean performance of the clique models over all players is recorded. For all clique behavioural models of team E, $\overline{D_B}, \overline{S_B}, \overline{D_R}$ *and* $\overline{S_R}$ are shown in Figure 5.16, $\overline{B_j}$ *and* $\overline{R_j}$ covering the quantised direction (1…16) is shown in Figure 5.17 and $\overline{B_j}$ *and* $\overline{R_j}$ covering the quantised speed (17…26) are shown in Figure 5.18.

As with the centroid behaviour model, the mean clique behaviour is much better than the random control, and the behaviour model appears to perform better at predicting the direction of movement rather than the associated speed. There is a significant drop in speed predictability from the lowest speed (except for the fastest sprinting speed). The most likely explanation is that the faster speeds are maintained for less time (and ended more abruptly) than the lower speeds. The anomalous rise

in predictability for the highest speed could be because situations in which the clique-associated player must break into a sprint are distinct from those in which the speed is more modest.

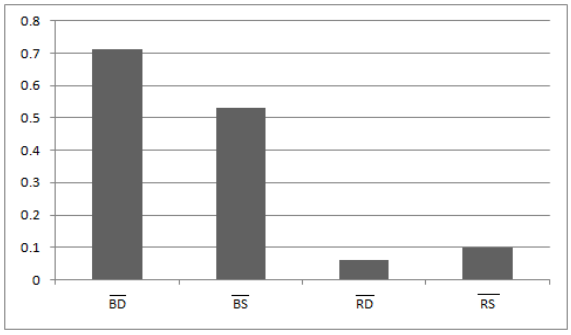

Figure 5.16 – $\overline{D_B}, \overline{S_B}, \overline{D_R}, \overline{S_R}$ mean over all cliques

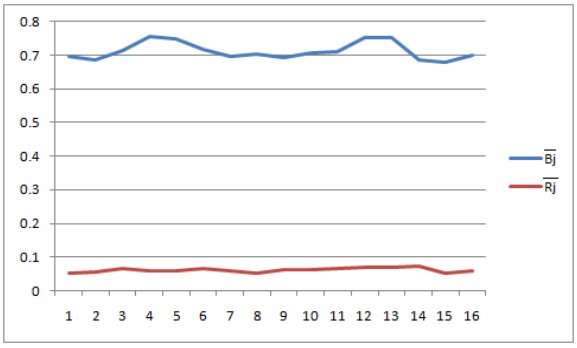

Figure 5.17 – $\overline{B_j}$, $\overline{R_j}$ for j=1…16 mean over all cliques

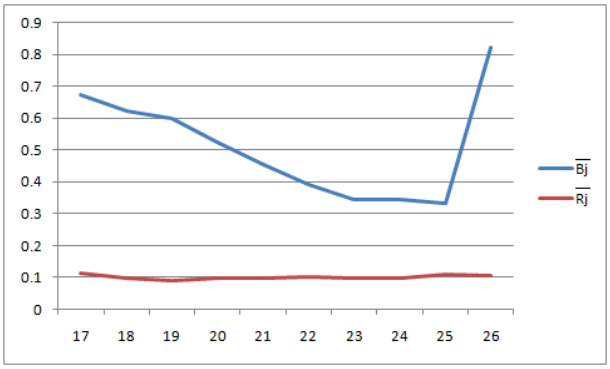

Figure 5.18 – $\overline{B_j}$, $\overline{R_j}$ for j=17…26 mean over all cliques

5.6.6.3 Mean results over all players

Over all player models for $\{P_1, P_2, P_3, P_4, P_5\}$, the mean results for $\overline{D_B}, \overline{S_B}, \overline{D_R}$ *and* $\overline{S_R}$ are shown in Figure 5.19, $\overline{B_j}$ *and* $\overline{R_j}$ covering the quantised direction (1…16) are shown in Figure 5.20 and $\overline{B_j}$ *and* $\overline{R_j}$ covering the quantised speed (17…26) are shown in Figure 5.21. As with both the centroid and the clique models, the performance over the quantised direction is higher and more uniform than the performance covering the quantised speed. The shape of the model performances in Figure 5.18 and Figure 5.21 are quite similar, and the author suggests that the reasons for this specific performance profile are the same as for the clique models; namely that higher speeds are maintained for less time and ended more abruptly, apart from the very fastest speeds which are initiated in more predictable circumstances.

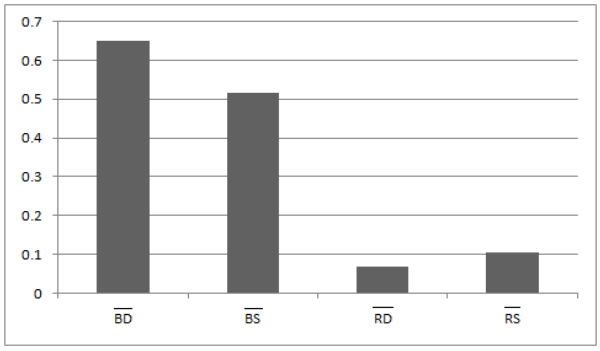

Figure 5.19 – $\overline{D_B}, \overline{S_B}, \overline{D_R}$ *and* $\overline{S_R}$ mean over all players

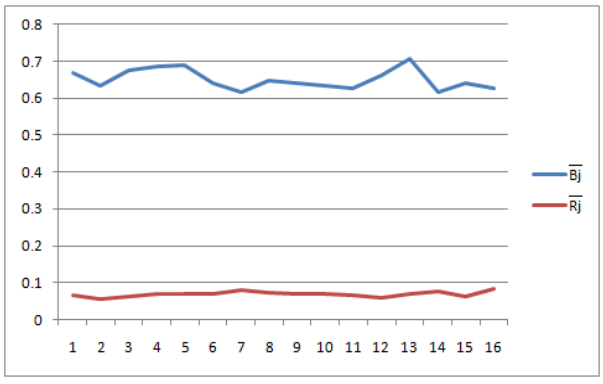

Figure 5.20 – $\overline{B_j}$, $\overline{R_j}$ for j=1…16 mean over all players

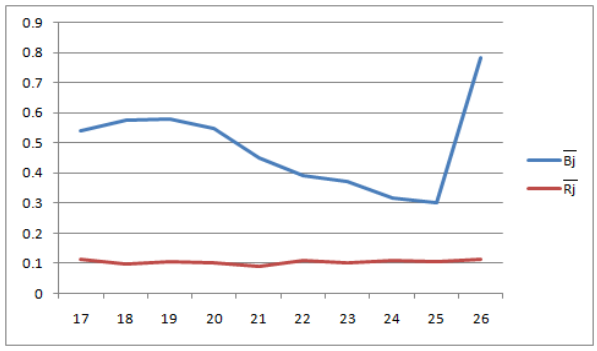

Figure 5.21 – $\overline{B_j}$, $\overline{R_j}$ for j=17…26 mean over all players

5.6.7 Experiment Two Evaluation Setup

The purpose of the second experiment was to examine how well simulated players match trajectories with the real players they model. For each of the five players modelled, twenty-five starting configurations each (total 125) were identified in the testing data in which the player had at least twenty-five seconds of uninterrupted movement. The real player trajectory was then ground truth. For each player configuration, the appropriate player model was used to generate twenty-five seconds of simulated player movement. During the simulation, the ball and the other players on the pitch moved exactly as they did during the real player's movement. Acting as a control comparison, the random walk model of section 5.6.5 was used to generate a twenty-five second random walk trajectory. As an additional control, a first-order linear predictor model was used to extrapolate the initial instantaneous velocity of the real player twenty-five seconds into the future, essentially predicting the straight line the player would take if his initial velocity were constant. In order to compare how well the models performed against the ground truth and each other, the follow procedure was followed:

(1) All ground truth trajectories are collected for the 125 distinct test configurations.
(2) Random walk, first order extrapolation and neural behaviour model trajectories are generated for each of the 125 test configurations.

(3) Each trajectory consists of 25 seconds worth of movement at 0.1 seconds fidelity – 250 points in total. To compare trajectories, the Euclidean distance between corresponding points is measured (see Figure 5.22), resulting in 250 difference readings.

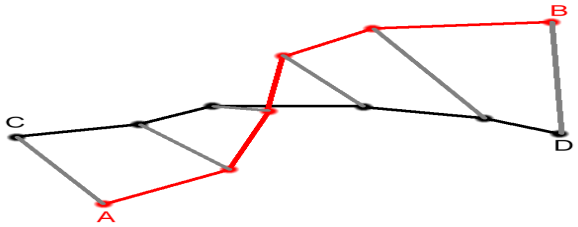

Figure 5.22 – Calculating the Euclidean distance between each corresponding set of points in trajectories AB and CD

(4) All 125 trajectories generated by each model type (neural behavioural, random walk, and first order extrapolation) are compared to the corresponding ground truth trajectories resulting in 125x250 difference readings for each method.

(5) For each model type, over a set of thresholds distances (0.05,0.1,0.15,0.25,0.5 in pitch normalised dimensions), the proportion of difference readings which are at or below a specific threshold are calculated. This results in 250 proportion figures for each model type/threshold value.

5.6.8 Experiment Two Results

Figure 5.23 shows the three competing models performance under the tightest threshold of 0.05. In the this and subsequent figures the labelling is RW=random walk, FD=first-order predictor and NN=Neural Behavioural model. Overall the neural behaviour model matches closest to the real trajectories, but the accuracy of all three fall off quickly, with the neural model only being within the threshold 50% of the time after approximately two seconds. All models become indistinguishable after approximately eight seconds.

At a threshold of 0.1 (Figure 5.24), a clear separation is becoming apparent between the neural behaviour model and the other control methods, with the first-order linear predictor model being marginally worse than the random walk model.

The next three threshold values of 0.15 (Figure 5.25), 0.25 (Figure 5.26), and 0.5 (Figure 5.27) show the widening separation between the models with the first-order linear predictor being shown to be the most inaccurate model, and the neural mode the most accurate of the three.

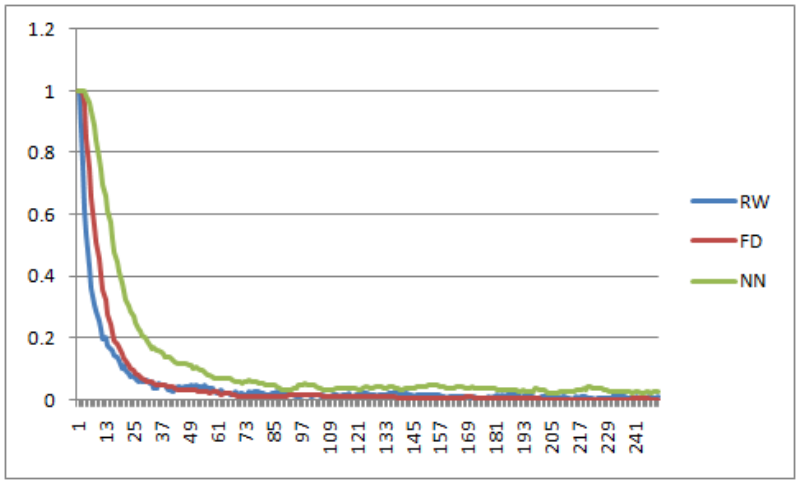

Figure 5.23 – 0.05 distance threshold results over twenty-five seconds

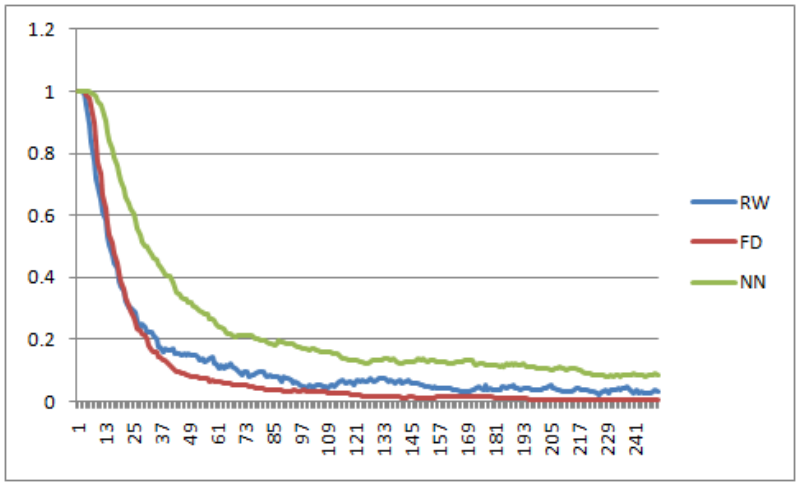

Figure 5.24 – 0.1 distance threshold results over twenty-five seconds

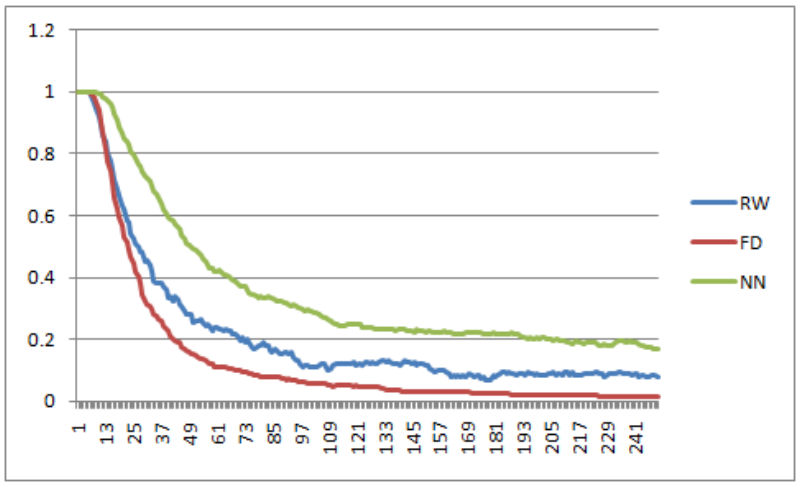

Figure 5.25 – 0.15 distance threshold results over twenty-five seconds

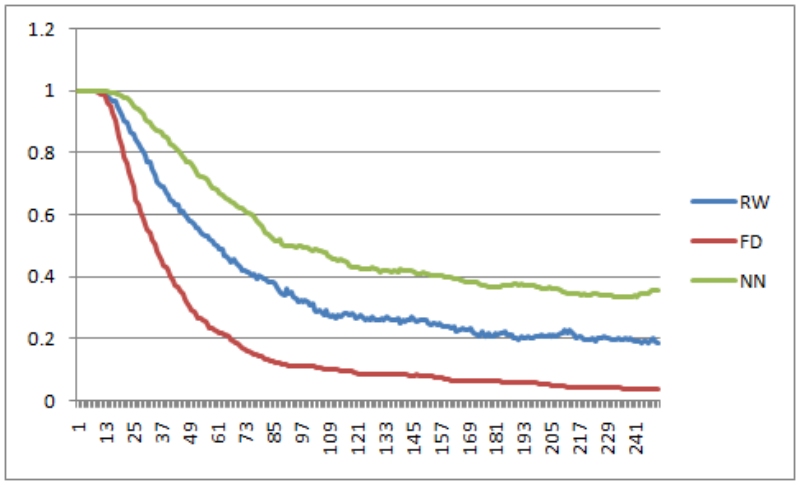

Figure 5.26 – 0.25 distance threshold results over twenty-five seconds

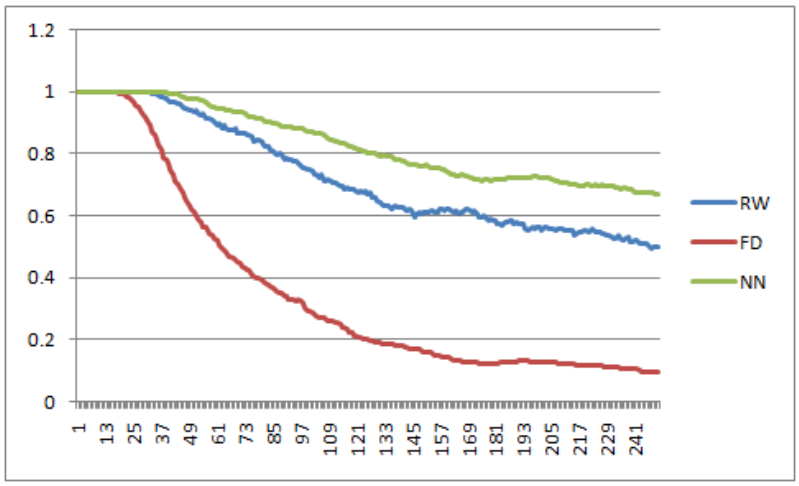

Figure 5.27 – 0.5 distance threshold results over twenty-five seconds

The fact that the first-order linear predictor performs so poorly suggest that football players do not tend to travel in straight lines over more than a few seconds. Random walking[37] will probabilistically hover around the origin point, so the fact that this model is more accurate than the first-order linear predictor suggests that players tend to stay in the same location more often than they travel in straight lines.

Three examples of actual simulated trajectories generated by the neural behaviour model show its strong and weak points. The author would characterise the generated trajectory in Figure 5.28 as quite successful, especially in the earlier sections of the trajectory. The generated trajectory has the ability to turn (although perhaps a little too 'loopy') whilst still maintaining definite direction. One problem appears to be a tendency to favour slow speeds (resulting in shorter trajectories and tying in with the speed inaccuracies noted in experiment one). The general modelling pitfall of errors feeding back into the model and amplifying over time is also apparent. This may suggest an over-sensitivity in the model to the simulated player as opposed to surrounding players and the ball.

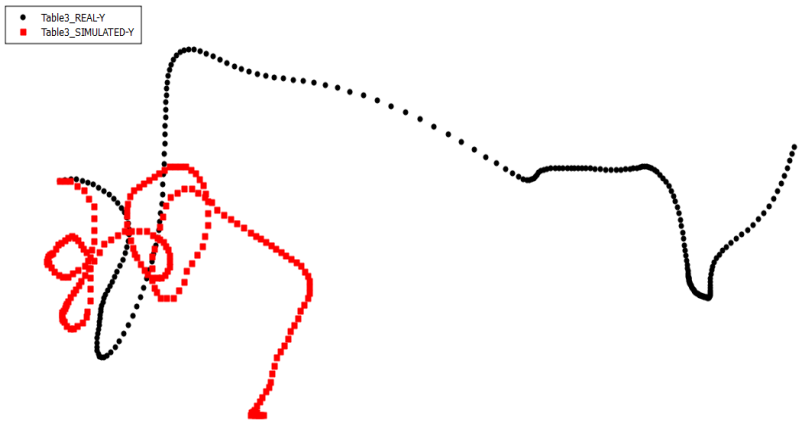

Figure 5.28 – Simulated (red) Vs Real (black) player trajectories #1

Figure 5.29 demonstrates that the model can generate larger area turns (which seem to be a common feature in real player trajectories), but once again the model does not do well to match the initial higher speed trajectory section, and eventually becomes disconnected from the real trajectory. Figure 5.30 demonstrates a poor simulated trajectory, which diverges almost immediately from the real trajectory, is slow in comparison, and eventually backtracks on itself.

[37] At least in two dimensions

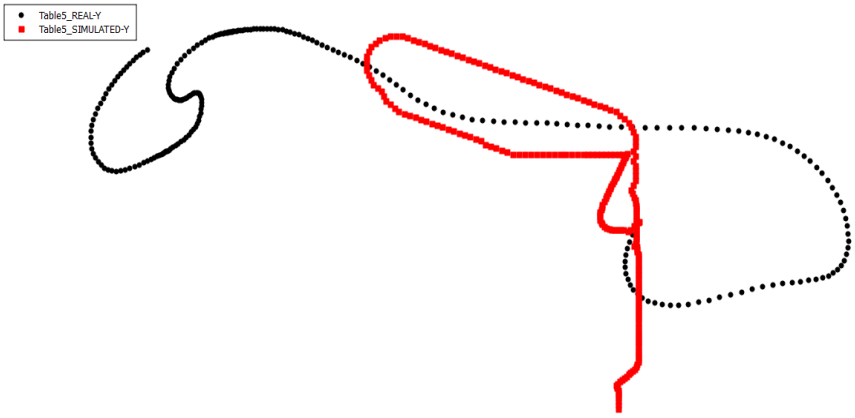

Figure 5.29 – Simulated (red) Vs Real (black) player trajectories #2

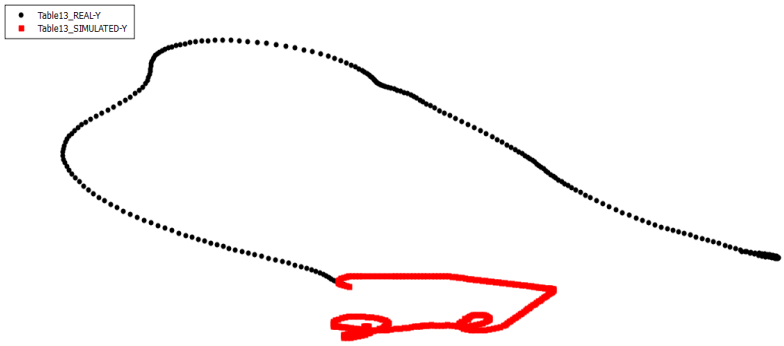

Figure 5.30 – Simulated (red) Vs Real (black) player trajectories #3

5.7 Discussion

Whilst this is at the moment a limited approach to behaviour modelling, covering only the motion of the players not their actions, the preliminary results do seem promising. The models at all levels seem to be able to predict the direction of motion for the next time instance with reasonable accuracy over all quantised directions. Experiment two, which involved an extended simulation for 25s, demonstrated that the models can produce reasonable accuracy for approximately the first 3s of the simulation, but after this accuracy drops off (in part due to building errors because of the autoregressive nature of the models).

The predictions for the quantised speed are somewhat more problematic, there seem to be two accurate outliers for most of the results, which cover both extremes

of quantised speed. This problem modelling speed became more evident in the time extended experiment two. The simulated trajectories have difficulty replicating real trajectories which exhibit higher speeds. More sophisticated evaluation of the models, which ideally could separate the feedback errors associated with autoregression from any intrinsic modelling errors, may shed more light on these problems.

5.8 Summary

The chapter explored developing behaviour models of players using a hierarchical approach, where groups of players were associated together in cliques depending on their relative proximity or similar direction of movement, as well as the mean motion of the entire team. The generated models performed quite well when asked to predict the next move of modelled players, and performed very well in comparison to a random walk player model used as a control. Analysis of the performance of the models over extended time periods again demonstrated that they can outperform random and linear extrapolation trajectory models, and have the capability to generate varied (and sometimes quite accurate) trajectories over a short time window (less than ten seconds).

6 Conclusions

The aim of the work in this thesis is to investigate the possibility of indexing an archive of games (described as collections of player trajectories and events) by using short segments of play, and also to investigate the possibility of building player behaviour models which can capture the systematic behaviour of players as evidenced by their trajectories. To this end the following work was undertaken:

6.1 Summary of work

Chapter 3 introduced eight SOP indexing schemes. Of these one was based on a semantically augmented ball trajectory, describing the ball trajectory as a temporally linked series of line segments to which were attached semantic information describing possession. This indexing scheme required the development of abstract prototypes both for spatial coordinates and for types of players. MDL was used in both cases to select the best model. Two of the player based indexing schemes also made use the spatial and player abstractions: the player trajectories and augmented cliques approach. Of the remaining five approaches, four used some variation of measuring player density (two used global player density, two used high entropy local patches of player density). The final approach used a very abstract clique based approach to describe the context around a SOP.

Chapter 4 dealt with aggregating results from a combined system of a ball indexer and a player indexer. The general problems encountered when fusing two results lists together were covered, and a score based aggregation approach was decided upon (rather than a rank based approach). This chapter also detailed the evaluation experiment for both the individual indexing systems, and there composite combined indexing system.

Chapter 5 covered the work undertaken in behaviour modelling using the player trajectory data available. A hierarchical based modelling approach was taken, whereby the behaviour of a player is influenced by the players associated with him (by proximity or a similar direction of movement), as well as the mean motion of the

entire team. The generated models performed quite well when asked to predict the next move of modelled players, and performed very well in comparison to a random walk player model used as a control. Testing involving extended trajectories of 25s in duration revealed that the model tends to reasonably accurate over the short term (<3s), but becomes progressively more inaccurate as the simulation duration increases, almost certainly in part because the autoregressive nature of the model amplifies earlier errors in later simulation steps. The extended duration experiment also highlighted some problems with correctly predicting medium player speeds (which were also evident to a lesser degree in the first experiment).

6.2 Contributions

The main contributions of this thesis are:

- A successful SOP indexing scheme using the semantically augmented ball trajectory.
- Several successful SOP indexing schemes based on player movements.
- The development of useful spatial and player prototypes which could be incorporated into other work.
- A query results aggregation system, with the ability to learn to map similarities to estimated ground truth values.
- A hierarchical behaviour model for player motion, which demonstrated reasonable agreement with reality.

6.3 Future research

With regard to indexing, one area of future research could be to attempt to combine the player and ball trajectory approaches within the same indexing scheme. From the results of the experiment in chapter 4, it is apparent that they are not completely mutually redundant descriptions of a SOP as the aggregation of player trajectory and ball trajectory results achieved a higher mean rating than either the ball trajectory or player trajectory mean ratings individually.

The combined aggregators used only one variant of player indexing with the ball indexer, future research could look into the effect of having two or more player indexing systems in concert with the ball indexer.

The concept of the context of a query was introduced to see if could improve the performance of results aggregation, by providing additional information to the ground truth which would allow the NN to bias the score combination on a per query type basis (provided a systematic effect was evident in the data). Although the top performing combined indexer system did use the PCA variant of the query context, overall the results were inconclusive as to whether this was a useful feature or not. The author suspects the compression performed to achieve a 2D context was too severe, and perhaps a less compressed version of the context could show superior performance. Alternatively, the RM could be combined with the PI in order to provide a richer description to compress.

A larger similarity gathering experiment would be useful to allow a more significant judgement to be made amongst the competing indexing systems. The current experiment using only sixteen raters with a four point Likert scale resulted in noisy data. The experiment could be usefully rerun with either a more nuanced rating scale (perhaps even a continuous variable for similarity), a panel of 'expert' raters (so that the subjective variation in similarities is reduced), or a much larger pool of normal raters (so the subjective variations can be averaged out over a large number of ratings).

There is much scope to research further into behaviour modelling of players. Constructing a player model which does not rely on the players' own trajectory, but only of surrounding players and the ball would make useful comparison as it would not be subject to feedback error amplification. Clustering is a possible approach for players with associated behaviour models, by observing and comparing their behaviour in exactly the same situations. Clusters of behaviour models leads to the concept of an average behaviour model. The current models only address the movement of the player; they do not address actions undertaken by the player at all, so this is one avenue of research, including questions such as can an action model coexist with a movement model, and could they use the same clique based data? Finally, can the behaviour models of players be used as the beginnings of a semantic approach to context indexing segments of play? The behaviour models having probabilistic outputs can potentially describe everything that could occur from a certain starting configuration of players and ball, so it might be possible to describe (and attach a probability to) all possible futures in a SOP, and then perhaps index the segment on a number if the most likely outcomes.

7 Appendices

7.1 The AVQ Algorithm

- Randomly place the ***k*** prototypes $\boldsymbol{m_i}$ (**i=1…k**) in the feature space
- Let ***x(t)*** be the feature vector for epoch ***t***
- Define:

$$\alpha(t) = 1 - \frac{t}{T} \tag{7.1}$$

  a monotonically decreasing gain coefficient
- To ensure the prototypes are representatively distributed, each node $\boldsymbol{m_i}$ has an associated sensitivity $\boldsymbol{S_i}$ which is initially zero
- Set a value for $\boldsymbol{\beta}$, the node sensitivity adjustment parameter. This must be small in comparison to the feature space.
- Train for ***T*** epochs, at each epoch ***t :***
  - Find the prototype $\boldsymbol{m_c(t)}$ which is nearest to this input:

$$c = \arg min_i(|x(t) - m_i(t)| - S_i(t)) \tag{7.2}$$

  - For each ***i*** update the prototypes and sensitivities:
    - If $i = c$

$$m_c(t+1) = m_c(t) + \alpha(t)|x(t) - m_c(t)| \tag{7.3}$$

$$S_c(\mathrm{t}+1) = \mathrm{S_c(t)} - \beta \tag{7.4}$$

    - If $i \neq c$

$$m_c(\mathrm{t}+1) = \mathrm{m_c(t)} \tag{7.5}$$

$$S_c(\mathrm{t}+1) = \mathrm{S_c}(\mathrm{t}) + \frac{\beta}{\mathrm{k}-1} \tag{7.6}$$

7.2 Valid Football Events

EventID	Event Type
1	Touch (of ball)
2	Dribble (of ball)
3	Header
4	Tackle
5	Cross
6	Clearance
7	Post Deflection
8	Crossbar Deflection
9	Shot
10	Header Shot
11	Goal
12	Own Goal
13	Start Of Half
14	End Of Half
15	Kick Off
16	Drop Ball
17	Stoppage
18	Ball Out Of Play
19	Goal Kick
20	Throw In
21	Foul Throw
22	Corner Pass

23	Corner Cross
24	Goalkeeper Save
25	Goalkeeper Punch
26	Goalkeeper Catch
27	Goalkeeper Throw
28	Goalkeeper Fumble
29	Goalkeeper Save Catch
30	Goalkeeper Pick Up
32	Goalkeeper Kick
33	Goalkeeper Drop Catch
34	Foul
35	Handball
36	Direct Free Kick Pass
37	Direct Free Kick Shot
38	Direct Free Kick Cross
39	Penalty Shot
40	Offside
41	Indirect Free Kick Pass
42	Indirect Free Kick Cross
43	Yellow Card
44	Red Card
45	Substitution
46	Block
47	Goalkeeper Foul
48	Pass
49	Deflection

Table 7.1 – Event IDs as used by ProZone to describe events occurring during a football game

7.3 User similarity opinions

7.3.1 User opinion #1

- By looking at any patterns, i.e. number of passes, directions etc to see if there are any similarities.

7.3.2 User opinion #2

- Patterns of trajectory
- Position of goalkeepers
- Location of ball
- Direction of play
- # of players in each half
- Which half of pitch the ball is in

7.3.3 User opinion #3

- Has a team 'scored psychological points' against the other?
- Has the ball changed hands?
- Was the play in the same third of the pitch?
- How much passing?

7.3.4 User opinion #4

- Main play, i.e. cross, short-passing
- If both attacking/defending
- Successful phase of play or not in both?
- Loosing/Keeping possession.

7.3.5 User opinion #5

- I compared which team had possession, what area of the pitch they were in. whether the passes were long or short and the eventual outcome of the move (shot, loss of possession, free kick, etc).

7.3.6 User opinion #6

- Motion of the players, whether attacking or defending
- Speed at which ball was played
- Number of players involved in an action
- Position of ball at that time
- The passes played, whether they were played long or short.

7.3.7 User opinion #7

- (mostly) ignored player types
- Ignored side of pitch and often if majority of play was in back half or front half of pitch. Didn't ignore this if I rated this 'v.similar'
- Possession seemed to influence my decision significantly
- Generally it had to do be the same kind of play to get a v.similar (i.e. corner kick, shot on goal, etc).
- Speed of play had a very slight influence
- Range of play (amount of pitch covered) had a slight influence

7.3.8 User opinion #8

- I tried to look at the movement of the ball, what I thought were the intended passess, which players passed it to who, and the movement of the players.
- When seeing two that looked similar, I would perhaps consider the formations and positions of the different players (attackers, defenders etc);

7.3.9 User opinion #9

- Ball as main reference point and possession

7.3.10 User opinion #10

- Goalkeeper activity/inactivity
- Ball going out of play
- Ball trajectory
- Defensive/attacking manoeuvre
- Change of possession

- Involvement of different categories of player

7.3.11 User opinion #11

- I tried to describe what was going on similar to that of a commentator, then compared the descriptions

7.3.12 User opinion #12

- Position of ball and it's play
- Position of players

7.3.13 User opinion #13

- The general direction of play
- Ball position
- Teams being defensive/attacking

7.3.14 User opinion #14

- Positions of players + player type
- Position of ball in play i.e. corner/goal shot/cross
- Direction of movement of players
- Team in possession + attacking/defending
- Thinking about what's happening and what might happen (attack / tackle / defend / goal etc), eg attack _+ goal can be very similar to attack + no goal.

7.3.15 User opinion #15

- If the same team is in possession at approx. the same time, the plays were more likely to be regarded as similar.
- If the outcome is similar, i.e. goal, defend with possession, goal kick then the plays were more likely to be regarded as similar.
- The player's positions were also used. The closer the player positions are to matching each other, the more likely to be regarded as similar the plays were.
- Roughly:
    - If all 3 apply, then very similar

- If 2 apply, quite similar
- If 1 applies then quite unsimilar
- If 0 applies then very unsimilar

7.3.16 User opinion #16

- Movement of ball w.r.t. movement of the players; that is the general direction of the ball in relation to the players movement around it. So, if in both cases the ball is being passed diagonally with surrounding players running after it then this would classify as being rather similar. Moreover, this must be consistent throughout the clip.
- The above coupled with a bijection between the timing of both events leads to a higher degree of similarity between clips.
- The grouping of players throughout the duration of the clip is also paramount. For there to be a high degree of similarity there should be a correlation (approximate at least) between the disposition of the players around the pitch. The players should also correspond in their respective positions (i.e. defenders are situated in similar positions in both clips).
- A minor method for selection of similarity is also the speed (or change thereof – acceleration) of the ball. Large differences in the speed or change of speed lead to less similarity between clips.

7.4 Query context – low dimensional projections from PCA and NN compression

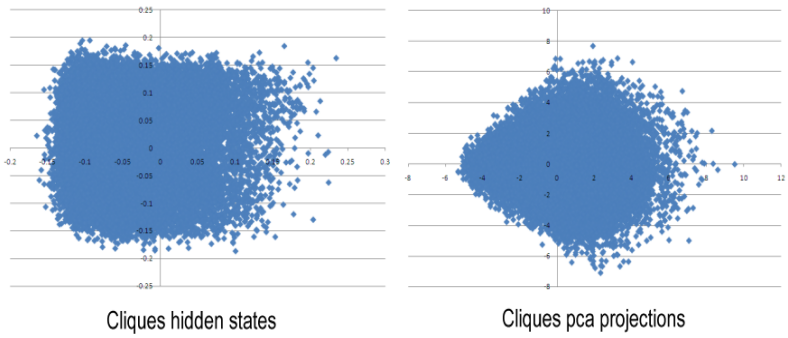

Figure 7.1 – 2D context projections for clique indexing system

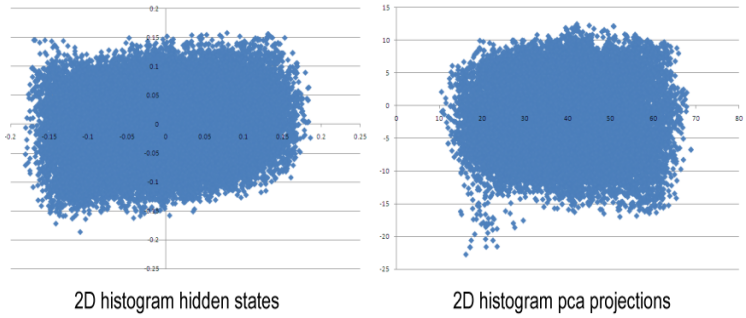

Figure 7.2 – 2D context projections for 2D histograms indexing system

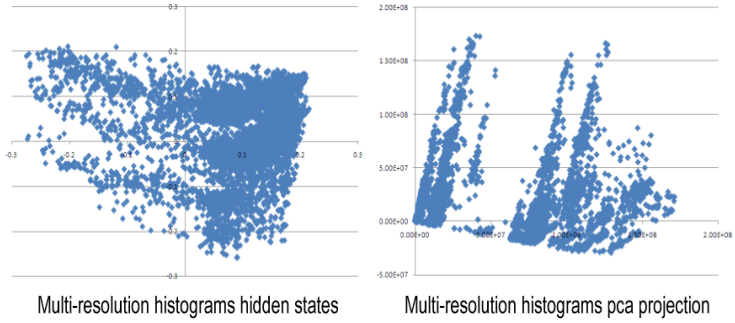

Figure 7.3 – 2D context projections for multi-resolution histograms indexing system

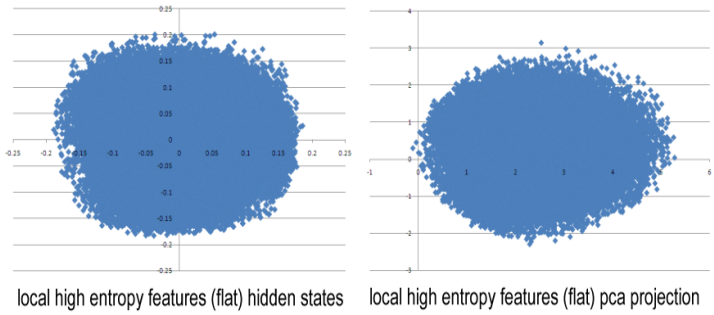

Figure 7.4 – 2D context projections for local features (flat) indexing system

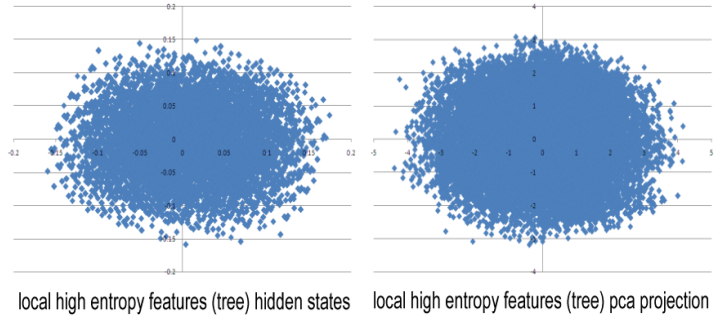

Figure 7.5 – 2D context projections for local features (tree) indexing system

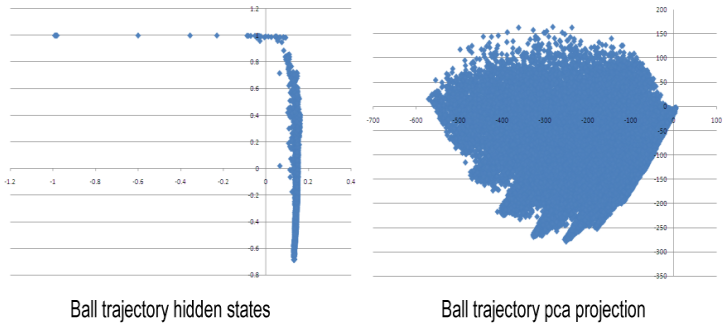

Figure 7.6 – 2D context projections for ball trajectory indexing system

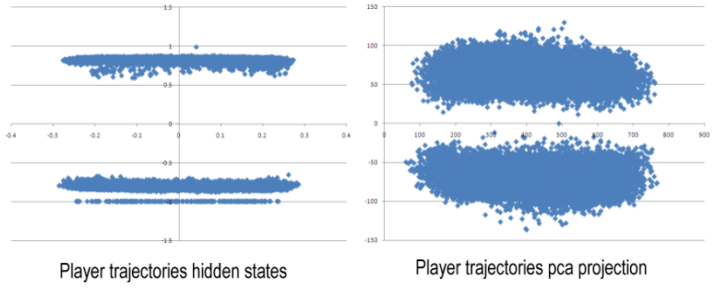

Figure 7.7 – 2D context projections for player trajectories indexing system

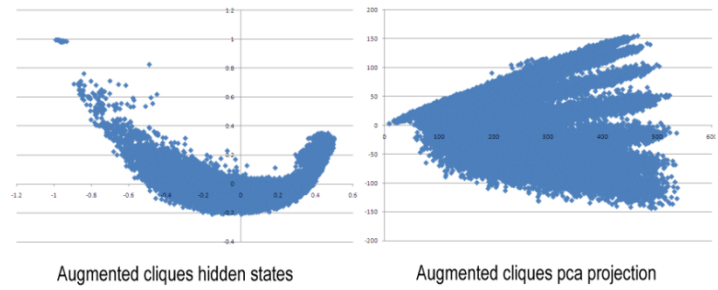

Figure 7.8 – 2D context projections for augmented cliques indexing system

7.5 Generated local feature histograms for entropy indexing systems

id	posx	posy	extentx	extenty	level	entropy	rank
981	2	0	1	3	2	0.693147	0
4340	5	4	1	2	2	0.693123	1
185	0	1	3	1	1	0.693114	2
1655	0	2	3	1	1	0.693043	3
4093	4	4	2	1	2	0.692999	4
2610	2	1	1	2	1	0.692914	5
771	2	0	3	2	1	0.692772	6
859	5	3	1	2	2	0.692528	7
3961	0	1	1	3	1	0.692425	8
2710	5	0	1	3	2	0.692387	9
4343	3	1	1	3	1	0.692347	10
4096	4	0	1	2	1	0.692265	11
254	0	0	3	1	1	0.692214	12
1566	2	6	1	3	5	0.691763	13
1899	3	3	2	1	2	0.691499	14
3048	1	2	2	1	2	0.691346	15
4153	5	1	4	1	2	0.690164	16
992	4	2	3	1	2	0.689912	17
2535	5	4	1	2	3	0.689715	18
2537	1	2	3	1	1	0.689564	19

Table 7.2 – Top 20 generated histograms for abstract team '-1'

Id	posx	posy	extentx	extenty	level	entropy	rank
512	3	4	2	1	2	0.693147	0
2892	5	2	1	2	2	0.693022	1
2535	5	4	1	2	3	0.692927	2
178	5	2	2	1	3	0.69291	3
1655	0	2	3	1	1	0.692825	4
4153	5	1	4	1	2	0.692788	5
378	3	2	3	1	1	0.692744	6
2442	5	4	4	2	2	0.692679	7
1204	7	1	1	4	2	0.692528	8
3054	6	0	1	3	2	0.69244	9
382	5	1	1	3	1	0.692231	10
2265	3	0	1	3	1	0.692033	11
4404	2	1	1	2	1	0.690347	12
2196	2	4	2	1	2	0.688798	13
3320	8	4	2	1	4	0.688038	14
2111	0	0	5	2	2	0.687936	15
2497	9	1	1	3	3	0.686929	16
3249	2	3	2	1	2	0.686098	17
4033	7	3	2	2	2	0.684115	18
533	6	6	2	1	4	0.683898	19

Table 7.3 – Top 20 generated histograms for abstract team '+1'

7.6 Fine player archetypes

For all archetype figures displayed in this section, the home team goal is on the left hand side of the figure, and the away team goal is on the right hand side of the figure. In general terms, the more an archetype is placed towards the right the more attacking it is, and *vica versa*. The archetypes can also be generally classed as either left-wing, central or right-wing depending on where the archetype occupies on the axis parallel to the goal line. The author has attempted to assign a known football position to each of the archetypes within the brackets in each figure caption.

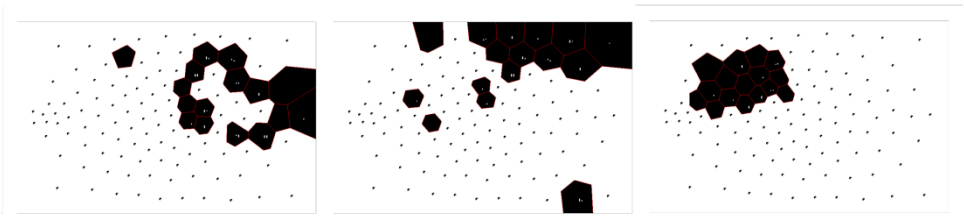

Figure 7.9 – Fine player archetypes set #1 (striker, left midfielder/forward, left fullback)

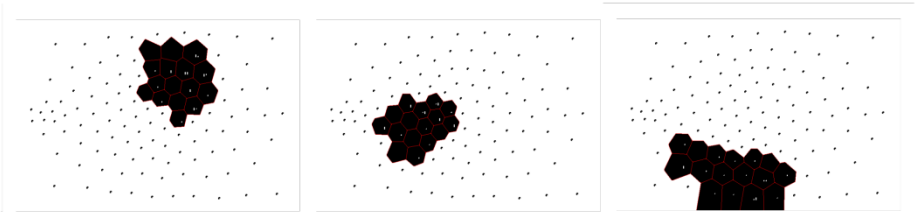

Figure 7.10 – Fine player archetypes set #2 (left midfielder, centre back, right fullback)

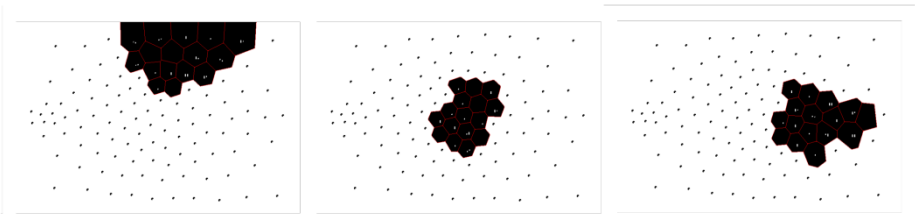

Figure 7.11 – Fine player archetypes set #3 (left midfielder, midfielder, striker)

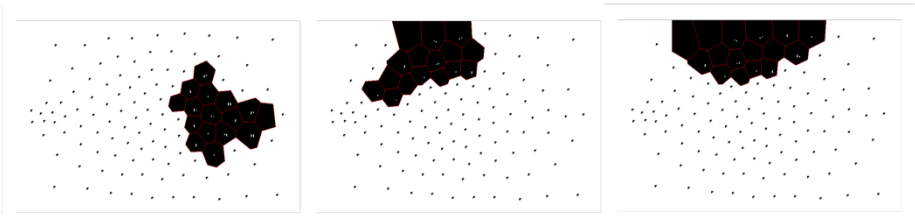

Figure 7.12 – Fine player archetypes set #4 (second striker, left fullback/midfielder, left midfielder)

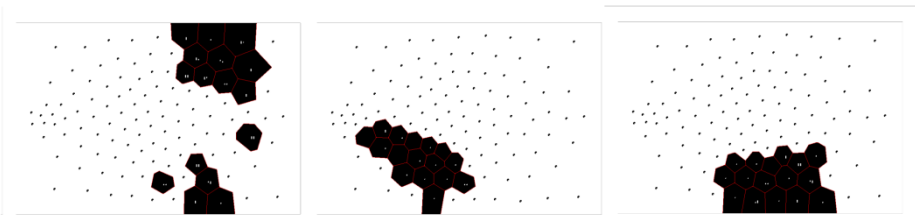

Figure 7.13 – Fine player archetypes set #5 (second striker, right fullback, right midfielder)

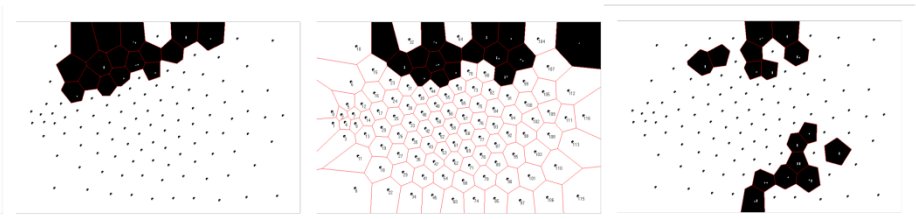

Figure 7.14 – Fine player archetypes set #6 (left fullback, left midfielder, midfielder)

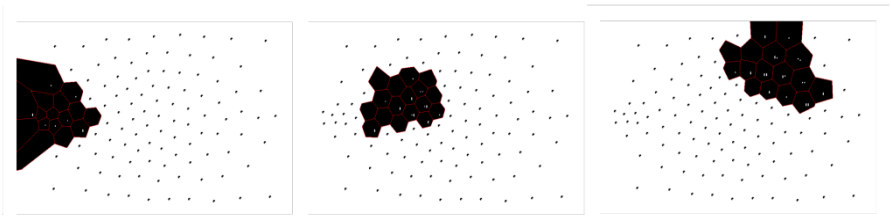

Figure 7.15 – Fine player archetypes set #7 (goalkeeper, sweeper, left forward)

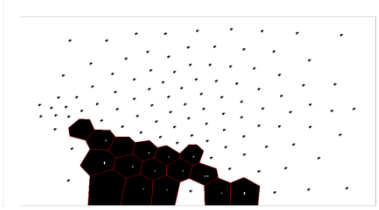

Figure 7.16 – Fine player archetypes set #8 (right fullback/midfielder)

7.7 Ball following algorithm

(1) Given a game ID M, for each half in the game the initial ball position is located at the centre point.

(2) Move through the current match events for game M in the relevant half until a ball-encounter class event is reached. From the playerID associated with the event, calculate the current position of the ball.

(3) Knowing the current position of the ball, the current time, the last known position of the ball and the time it was there, interpolate a straight-line trajectory between the two points giving the ball a constant velocity.

(4) Go back to (2) until the events from both halves of game M are exhausted.

7.8 Database overview

The volume of trajectory and event data available (approximately 7GB in all) for use in this thesis requires the use of a database in order to efficiently store and analyse it, and also to store the football play segment indexing structures. As that is the case, an overview of their basic functionality of databases is prudent.

The vast majority of database systems currently available are based on the relational model [4], whereby data is structured in terms of tables, rows and columns (see Figure 7.17).

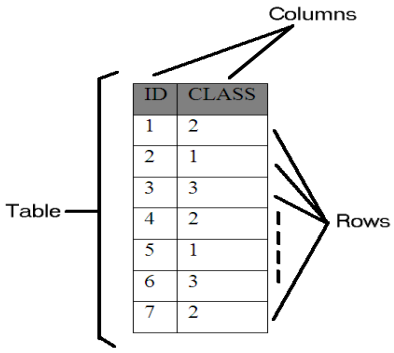

Figure 7.17 – Relational model used widely in modern day databases

SQL (Structured Query Language) [5], is the de facto language used with relational databases to perform data creation, manipulation and query. Its syntax is predicate logic based, and results of SQL queries are represented as sets of column values. Figure 7.18 shows an example SQL query performed on the table defined in Figure 7.17.

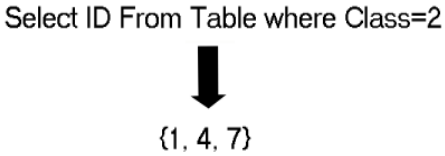

Figure 7.18 – A simple example of a Select SQL query on a database table

At the most basic level, performing queries on one or more database tables involves examining each row in the table(s), and selecting only those that satisfy the

conditions of the query. The computational cost of performing the queries grow in direct proportion to the size of the tables using the basic query method. For a large table this can be an extremely costly operation.

Database indexes offer a way to ameliorate the performance of queries performed on large datasets. The approach is analogous to that taken when a simple binary tree is used to increase the efficiency of searching a linear list (see Figure 7.19). By partitioning the data, the binary tree requires less steps (on average) to locate any value than the equivalent linear list. A perfectly balanced binary tree will reduce the search complexity of a linear list of $O(n)$ to that of $O(log_2\ n)$.

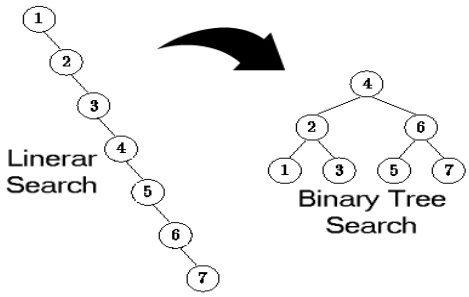

Figure 7.19 – Increasing search efficiency by restructing data elements

The index data structure most commonly used in database systems is that of the B-Tree [6] (see Figure 7.20), a more complex version of the binary tree which allows multiple scalar values to be stored within the tree nodes, thereby enabling each node to have more than 2 children. When used in database indexing, the interior nodes of the B-Tree hold values of the indexed entity, allowing a rapid traversal of the tree to the leaves, which hold the actual references to rows in the database that match the index key of the immediate parent node.

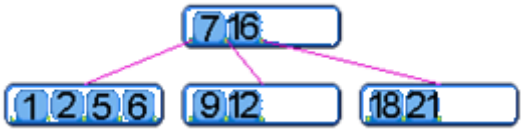

Figure 7.20 – B Tree (general purpose database structure)

Another common database indexing structure is the R-Tree [7] (see Figure 7.21), which is specialised to efficiently handle spatial information (it is frequently used in Geographic Information Systems). Indexes are based on a n-dimensional space model (where n is arbitrary and at the users discretion) , and each interior node of the tree holds one or more minimum bounding boxes (MBB), which effectively partition the n-dimensional space into discrete cuboid n-dimensional volumes (or rectangular areas if n=2). The collection of MBBs within an R-Tree will be such that the tree is as balanced as possible. Spatial queries fall into one or more MBBs at each level[38] of the tree (MBBs may overlap), the MBBs getting progressively smaller as the tree is traversed. The leaves of the tree hold references to database rows whose spatial index exists within the immediate parents MBB.

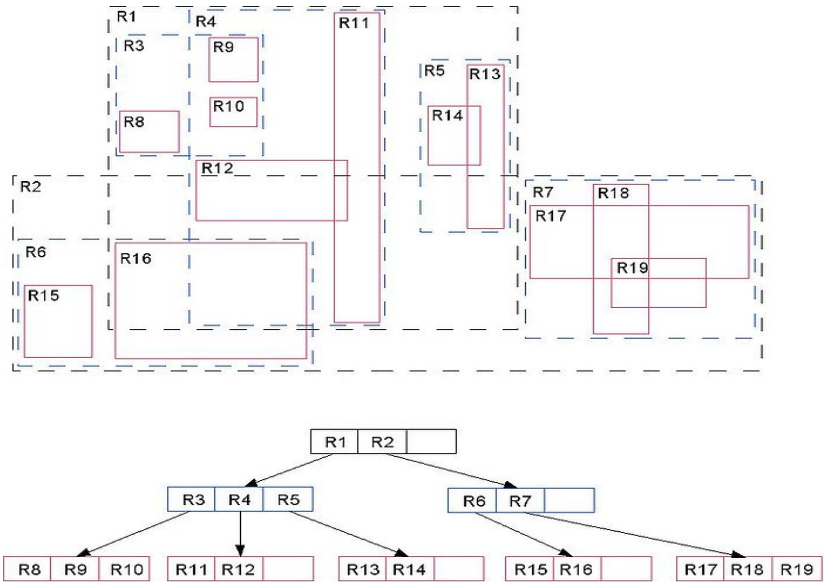

Figure 7.21 – R Tree (specialised to hold geographic data)

Indexes are based on the column values of tables (the key of the index), so in the case of the table defined in Figure 7.17 an index could be constructed from the ID column, the Class column, or a combination of the two. The index associates values of the key of the index with that of a set of rows in the database that share the same values for the key, allowing much quicker access to relevant rows of data. Given that the purpose of a database index is to speed up queries, knowledge of

[38] Or at least are closest to one MBR

which queries are most probable, and specifically which table columns will be referenced in those queries, is essential in order to construct a useful index. If the query in Figure 7.18 were the most probable query to be performed on the table in Figure 7.17, then the most efficient column to construct an index on would be the Class column.

7.9 Implementation details for Context indexing with player cliques

7.9.1 Implementation of indexing scheme

Clique discovery is applied to the initial player positions, and to the players' overall direction of movement for each SOP considered, to produce two sets of cliques per team (Figure 7.22), that is four sets of cliques in total. The contents of the cliques are raw player IDs, and as such are semantically meaningless in themselves. However, the distribution of the sizes of the cliques is potentially semantically meaningful. In a similar fashion to the bag of words model [19], a count is kept of clique size distribution resulting from each of the four set of cliques (for an example distribution see Figure 7.23). Each distribution has ten values, representing the valid range of sizes of cliques within a team of eleven players maximum.

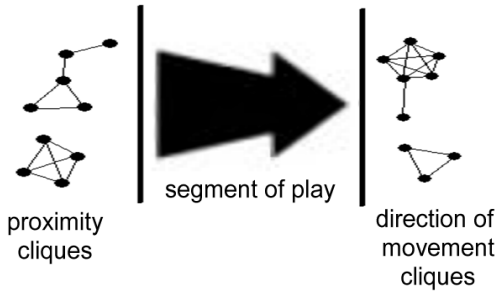

Figure 7.22 – Clique context around the beginning and end of a SOP

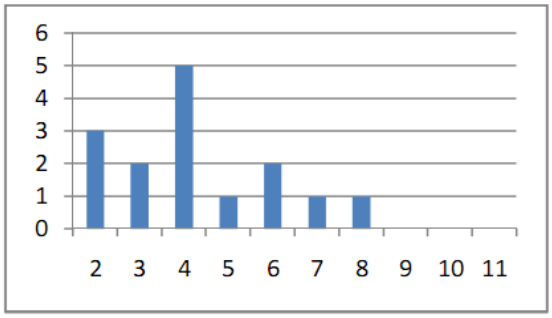

Figure 7.23 – an example clique size distribution

This transforms the four sets of cliques into four integer vectors (each with ten components). These four vectors form the basis of the PI, which is composed of the concatenation of the vectors as shown in Figure 7.24, where: A = Team '+1's proximity distribution, B = Team '+1's movement distribution, C = Team '-1's proximity distribution, D = Team '-1's movement distribution.

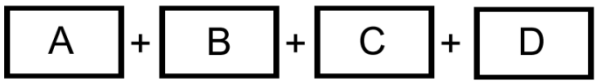

Figure 7.24 – Clique PI composed of the two teams clique distributions

Each clique has a spatial centroid associated with it, which is dependent on the player positions within the cliques. For a proximity clique of size n, this centroid $(\overline{CP_x}, \overline{CP_y})$ is given by:

$$\overline{CP_x} = \frac{1}{n}\sum_{t=1}^{n} Px_t \quad \overline{CP_y} = \frac{1}{n}\sum_{t=1}^{n} Py_t \tag{7.7}$$

Where Px_t and Py_t are the pitch normalised initial x and y coordinates of the t^{th} player in the clique. The spatial centroid for a direction of movement clique of size n, $(\overline{CD_x}, \overline{CD_y})$ is given by:

$$\overline{CD_x} = \frac{1}{n}\sum_{t=1}^{n} Px_t + \frac{(Fx_t) - (Px_t)}{2} \tag{7.8}$$

$$\overline{CD_y} = \frac{1}{n}\sum_{t=1}^{n} Py_t + \frac{(Fy_t) - (Py_t)}{2} \tag{7.9}$$

Where Px_t and Py_t are the pitch normalised initial x and y coordinates of the t^{th} player in the clique. Fx_t and Fy_t are the pitch normalised final x and y coordinates of the t^{th} player in the clique. Since the number of cliques (both proximity and distance) is variable, the RM will be composed of the mean centroid of each of the four sets of cliques. The mean centroid ($\overline{C_x}$, $\overline{C_y}$) of a set of cliques of size m is given by:

$$\overline{C_x} = \frac{1}{m}\sum_{t=1}^{m} Cx_t \tag{7.10}$$

$$\overline{C_y} = \frac{1}{m}\sum_{t=1}^{m} Cy_t \tag{7.11}$$

Where Cx_t and Cy_t are the pitch normalised centroid x and y coordinates of the t^{th} clique in the clique set. These four sets of clique centroids are concatenated into an eight dimensional vector that forms the RM component of the indexing scheme. Its form is shown in Figure 7.25, where E = Team '+1's proximity cliques centroid, F = Team '+1's movement cliques centroid, G = Team '-1's proximity cliques centroid, H = Team '-1's movement cliques centroid.

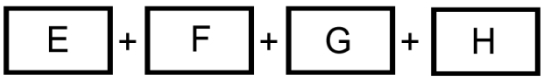

Figure 7.25 – Clique RM composed of the two teams clique centroids

The RM is associated with the 3-tuple SOPREF, and both are associated with the PI (see Figure 7.26).

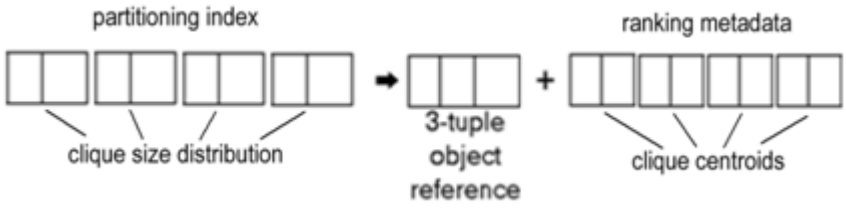

Figure 7.26 – clique size distribution indexing scheme

7.9.2 Query matching

Queries by example (i.e. a SOP) are broken down into a clique set PI and RM as detailed in the previous section. Query matching is done in two stages. In the first stage all (3-tuple, metadata) pairs which are associated with a PI identical to the query PI are collated. The second stage ranks the results of the first stage by the Euclidean distance between the query metadata feature vector and the metadata feature vector of each of the first stage results.

7.9.3 Query relaxation

Queries initially have no relaxation associated with them and proceed as described in the previous section. However queries may be relaxed gradually, and the relaxation scheme proceeds as follows:

(1) Initial relaxation $r = 0$, each subsequent relaxation level is $+1$ (i.e. $0 \rightarrow 1 \rightarrow 2 \ldots \rightarrow n$)

(2) At relaxation level r each component of the PI (i.e. each distribution value) is allowed to vary by $+/- r$ and still match the corresponding component of the query PI (Figure 7.27)

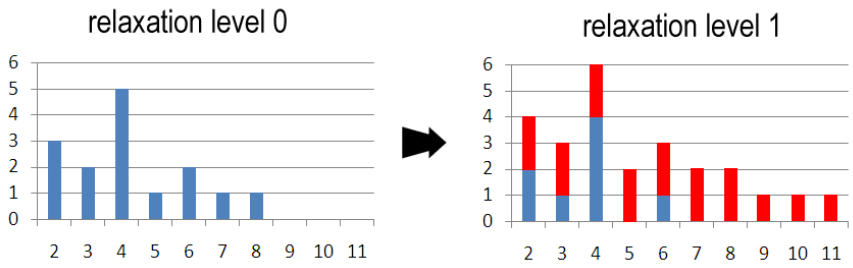

Figure 7.27 – clique indexing query relaxation process

7.10 Implementation details for Team mass indexing with 2D histogram

7.10.1 Implementation of indexing scheme

One histogram is used per team, so in total 12 integers describe the position of the bulk of players in both teams. These integers form the basis of the PI (see Figure 7.30), and are arranged as shown in Figure 7.28, where A = team '+1's top 6 densest histogram bins B = team '-1's top 6 densest histogram bins.

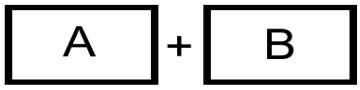

Figure 7.28 – 2D histogram PI composed of the top six histograms bins for each team

The RM consists of the mean centroid for each team over the whole SOP. Each team centroid $(\overline{C_x}, \overline{C_y})$ is defined as follows:

$$\overline{C_x} = \frac{1}{nm}\sum_{t=1}^{n}\sum_{i=1}^{m} Px_{i,t} \tag{7.12}$$

$$\overline{C_y} = \frac{1}{nm}\sum_{t=1}^{n}\sum_{i=1}^{m} Py_{i,t} \tag{7.13}$$

Where $Px_{i,t}$ and $Py_{i,t}$ are the pitch normalised x and y coordinates of the i^{th} player in the team of m players, at time position t in the play segment, which has a total number of time positions n. The RM is arranged to form a 4D feature vector as in Figure 7.29, Where C = team '+1's mean centroid over the SOP, D = team '-1's mean centroid over the SOP

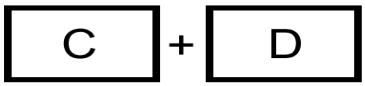

Figure 7.29 – 2D histograms RM containing the team centroid of each team

The RM is coupled with the object reference composed of a 3-tuple {MatchID, HalfID, Matchtime}, and both are associated with the PI (Figure 7.30).

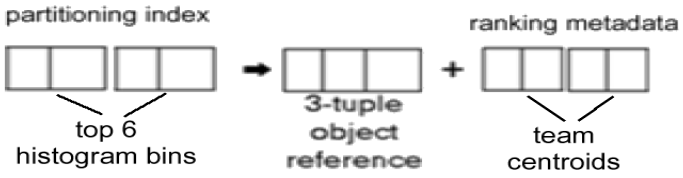

Figure 7.30 – 2D histogram indexing scheme

7.10.2 Query matching

Queries by example are broken down into a PI and RM as detailed in the previous section. Query matching is done in two stages. In the first stage all (3-tuple, metadata) pairs which are associated with a PI identical to the query PI are collated. The second stage ranks the results of the first stage by the Euclidean distance between the query metadata feature vector and the metadata feature vector of each of the first stage results.

7.10.3 Query relaxation

Queries initially have no relaxation associated with them and proceed as described in the previous section. However queries may be relaxed gradually, and the relaxation scheme proceeds as follows:

(1) Initial relaxation $r = 0$, each subsequent relaxation level is $+1$, up to a maximum of 6.

(2) At relaxation level r, only the top $(6 - r)$ bins for each team are compared against the query PI. The maximum relaxation that can be reached is 6, at which time all indexed objects trivially match the query PI.

7.11 Implementation details for Team mass indexing with multi-resolution 2D histograms

7.11.1 Implementation of indexing scheme

In total the integer vector component of the PI comprises 6 integers (3 for each team) represent coarse-to-fine measures of player density, and is arranged as in Figure 7.31, where A = team '+1's 3x2 histogram represented as an integer, B = team '+1's 6x4 histogram represented as an integer, C = team '+1's 9x6 histogram represented as an integer, D = team '-1's 3x2 histogram represented as an integer, E = team '-1's 6x4 histogram represented as an integer, F = team '-1's 9x6 histogram represented as an integer.

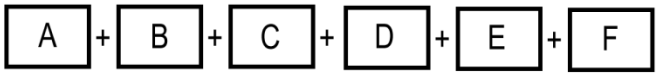

Figure 7.31 – multi-resolution histograms PI containing the three levels of histograms (each represented as in integer) for both team

The RM consists of the mean centroid for each team over the whole SOP. Each team centroid $(\overline{C_x}, \overline{C_y})$ is defined as follows:

$$\overline{C_x} = \frac{1}{nm} \sum_{t=1}^{n} \sum_{i=1}^{m} Px_{i,t} \tag{7.14}$$

$$\overline{C_y} = \frac{1}{nm}\sum_{t=1}^{n}\sum_{i=1}^{m} Py_{i,t} \tag{7.15}$$

Where $Px_{i,t}$ and $Py_{i,t}$ are the pitch normalised x and y coordinates of the i^{th} player in the team of m players, at time position t in the play segment, which has a total number of time positions n. The RM is arranged to form a 4D feature vector as in Figure 7.32, Where G = team '+1's mean centroid over the SOP, H = team '-1's mean centroid over the SOP.

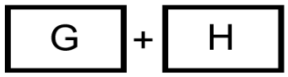

Figure 7.32 – Multi-resolution histograms RM containing the team centroid of each team

The RM is coupled with the object reference composed of a 3-tuple {MatchID, HalfID, Matchtime}, and both are associated with the PI (Figure 7.33).

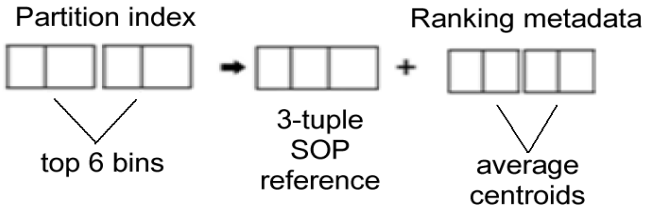

Figure 7.33 – multi-resolution histograms indexing scheme

7.11.2 Query matching

Queries by example are broken down into a PI and RM as detailed in the previous section. Query matching is done in two stages. In the first stage all (3-tuple, metadata) pairs which are associated with a PI identical to the query PI are collated. The second stage ranks the results of the first stage by the Euclidean distance between the query metadata feature vector and the metadata feature vector of each of the first stage results.

7.11.3 Query relaxation

Queries initially have no relaxation associated with them and proceed as described in the previous section. However queries may be relaxed gradually, and the relaxation scheme proceeds as follows:

(1) Initial relaxation $r = 0$, each subsequent relaxation level is $+1$, up to a maximum of 3.
(2) At relaxation level 1 , only the 3x2 and 6x4 histograms for each abstract team are matched against the query PI
(3) At relaxation level 2 , only the 3x2 histogram for each abstract team is matched against the query PI
(4) At relaxation level 3 , all indexed objects match against the query partition object.

7.12 Implementation details for Team mass indexing with local high entropy features

7.12.1 Indexing scheme

The 40 best features discovered (20 per team) are assigned unique ids and recorded for future use. Each feature has its own entropy value attached to it, to allow them to be ordered within each collection (in order of highest individual entropy first). The indexing partition structure is thus two sets of 20 features, each of which can be either present(1) or absent(0) for any particular SOP, giving a vector of twenty integers per team. The two vectors are concatenated into the PI as shown in Figure 7.34, Where *A* = team '+1's local features, *B* = team '-1's local features

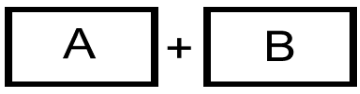

Figure 7.34 – high entropy local features PI containg bitfields for both teams indicating the presence/absence of a set of multi-resolution features

The RM consists of the mean centroid for each team over the whole SOP. Each team centroid ($\overline{C_x}$, $\overline{C_y}$) is defined as follows:

$$\overline{C_x} = \frac{1}{nm}\sum_{t=1}^{n}\sum_{i=1}^{m} Px_{i,t} \tag{7.16}$$

$$\overline{C_y} = \frac{1}{nm}\sum_{t=1}^{n}\sum_{i=1}^{m} Py_{i,t} \tag{7.17}$$

Where $Px_{i,t}$ and $Py_{i,t}$ are the normalised x and y coordinates of the i^{th} player in the team of m players, at time position t in the play segment, which has a total number of time positions n. The RM is arranged to form a 4D feature vector as in Figure 7.35, where C = team '+1's mean centroid over the SOP, D = team '-1's mean centroid over the SOP.

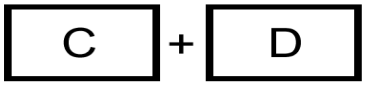

Figure 7.35 – High entropy local features RM containing the team centroid of each team

The RM is coupled with the object reference composed of a 3-tuple {MatchID, HalfID, Matchtime}, and both are associated with the PI (Figure 7.36).

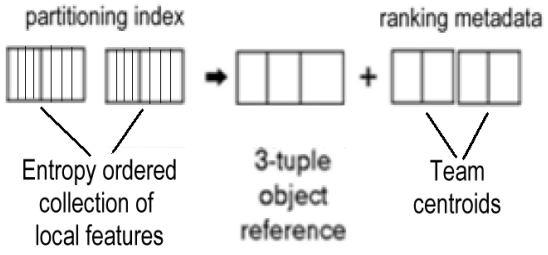

Figure 7.36 –high entropy local features indexing scheme

7.12.2 Query matching

Queries by example are broken down into a PI and RM as detailed in the previous section. Query matching is done in two stages. In the first stage all (3-tuple, metadata) pairs which are associated with a PI identical to the query PI are collated. The second stage ranks the results of the first stage by the Euclidean distance between the query metadata feature vector and the metadata feature vector of each of the first stage results.

7.12.3 Query relaxation

Queries initially have no relaxation associated with them and proceed as described in the previous section. However queries may be relaxed gradually, and the relaxation scheme proceeds as follows:

(1) Initial relaxation $r = 0$, each subsequent relaxation level is +1, up to a maximum of 20.
(2) At relaxation level r, only the $(20 - r)$ highest individual entropy features for each abstract team are compared against the query PI. The maximum relaxation that can be reached is 20, at which time all indexed objects trivially match the query PI.

7.13 Implementation details for Team mass indexing with hierarchical high entropy features

7.13.1 Indexing scheme

The PI structure is shown in Figure 7.37, Where A = team '+1's tree traversal vector, B = team '-1's tree traversal vector.

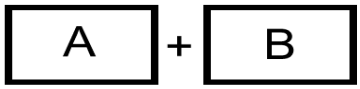

Figure 7.37 – tree structured high entropy local features PI containing bitfields describing tree traversals for both teams

The RM consists of the mean centroid for each team over the whole SOP. Each team centroid ($\overline{C_x}$, $\overline{C_y}$) is defined as follows:

$$\overline{C_x} = \frac{1}{nm}\sum_{t=1}^{n}\sum_{i=1}^{m} Px_{i,t} \tag{7.18}$$

$$\overline{C_y} = \frac{1}{nm}\sum_{t=1}^{n}\sum_{i=1}^{m} Py_{i,t} \tag{7.19}$$

Where $Px_{i,t}$ and $Py_{i,t}$ are the normalised x and y coordinates of the i^{th} player in the team of m players, at time position t in the play segment, which has a total number of time positions n. The RM is arranged to form a 4D feature vector as in Figure 7.38, where C = team '+1's mean centroid over the SOP, D = team '-1's mean centroid over the SOP.

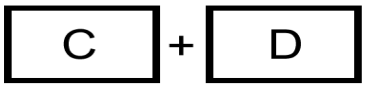

Figure 7.38 – tree structured high entropy local features RM containing the team centroid of each team

The RM is coupled with the object reference composed of a 3-tuple {MatchID, HalfID, Matchtime}, and both are associated with the PI (Figure 7.39).

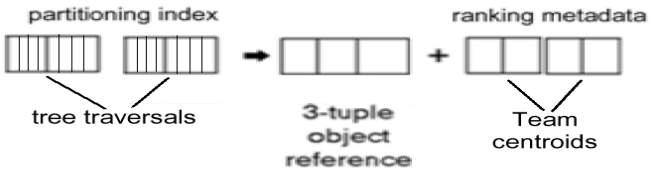

Figure 7.39 – tree structured high entropy local features indexing scheme

7.13.2 Query matching

Queries by example are broken down into a PI and RM as detailed in the previous section. Query matching is done in two stages. In the first stage all (3-tuple, metadata) pairs which are associated with a PI identical to the query PI are collated. The second stage ranks the results of the first stage by the Euclidean distance between the query metadata feature vector and the metadata feature vector of each of the first stage results.

7.13.3 Query relaxation

Queries initially have no relaxation associated with them and proceed as described in the previous section. However queries may be relaxed gradually, and the relaxation scheme proceeds as follows:

(1) Initial relaxation $r = 0$, each subsequent relaxation level is $+1$, up to a maximum of N, where N is the depth of the tree.

(2) At relaxation level r, only the first $(N - r)$ tree traversal features for each abstract team are compared against the query PI. The maximum relaxation that can be reached is N, at which time all indexed objects trivially match the query PI.

7.14 Implementation details for Semantically augmented ball trajectories

7.14.1 Indexing scheme

Given a SOP, the ball trajectory within it is reconstructed from the player positions and events as a series of line segments. Line segments which cross the boundaries of the SOP are truncated as shown in Figure 7.40.

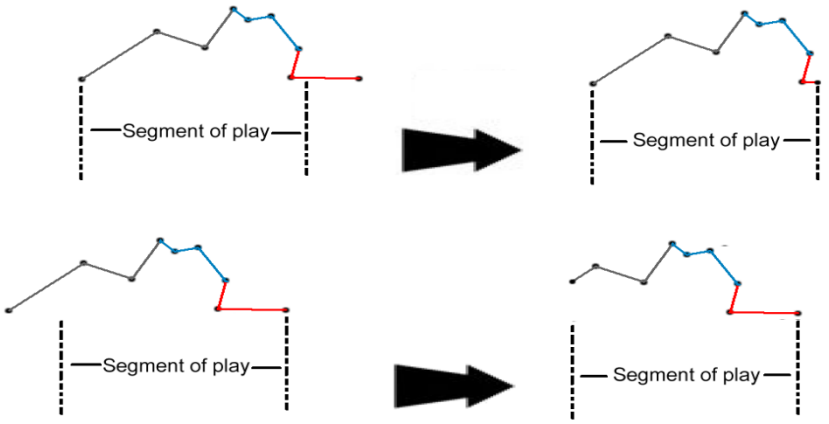

Figure 7.40 – Ball trajectory truncation to lie within a SOP

The beginning and end of each line segment is mapped to the nearest spatial prototype in the codebook. Each line segment is associated with a club ID and a player ID. The club ID is mapped onto the abstract club '+1'/'-1'. The player ID is mapped onto the corresponding gross player archetype and fine player archetype. A line segment is represented in the PI as shown in Figure 7.41, where B = Nearest spatial prototype to exact beginning of line segment, E = Nearest spatial prototype to exact end of line segment, A = Abstract team in possession during line segment (either +1 or -1), G = Gross player archetype corresponding to player in possession during line segment, F = Fine player archetype corresponding to player in possession during line segment.

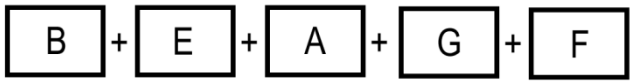

Figure 7.41 – Ball trajectory segment PI containing beginning and end spatial prototypes and the team and type of player in possession

The RM for each line segment in the ball trajectory is structured as shown in Figure 7.42, where: T = temporal order of line segment in ball trajectory (1…n) of n segments, D = time duration of line segment in multiples of 0.1s, P = exact beginning coordinates of line segment (pitch normalised), Q = exact ending coordinates of line segment (pitch normalised).

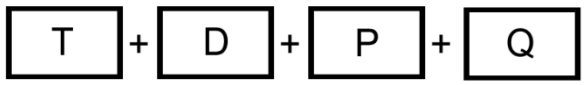

Figure 7.42 – Ball trajectory segment RM containing details of one particular line segment within the ball trajectory (one or more are required to desribe ball trajectory over entire SOP)

The RM is coupled with the SOPREF composed of a 3-tuple {Matchid, Halfid, Matchtime}, and both are associated with the PI (see Figure 7.43).

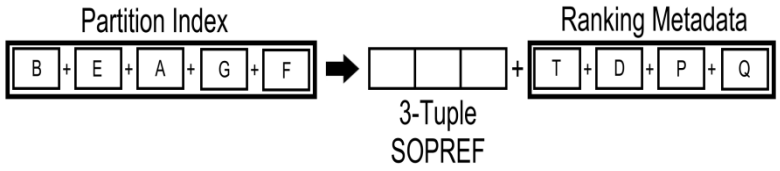

Figure 7.43 – Semantically augmented ball trajectory indexing scheme

7.14.2 Query matching

Queries by example are broken down into N PIs and N RM collections representing the query ball trajectory of N segments, as described in the previous section. The aim is to find all ball trajectories that share at least something in common with the query ball trajectory. To accomplish this, each line segment within the query ball trajectory is viewed as a separate sub-query.

Each individual line segment from the query is posed as a sub-query and matches those indexed line segments which exactly share it's PI (at level 0 relaxation). The result of the sub-query is a set of SOPREF/RM pairs. For the purposes of the sub-queries, only the unique SOPREFs in each set are kept. After each line segment has been sub-queried, there exists a collection of sets containing SOPREFs. These sets are merged, and the result is a set of unique SOPREFs.

As each SOPREF is associated with a set of line segments representing the ball trajectory over the SOP, the set of unique SOPREFs which result from the sub-queries represent all indexed ball trajectories which share at least some part in common with the query ball trajectory. However, there is no ranking associated with the set of SOPREFs at this stage, they are all equally similar to the query. A simple approach to similarity ranking could be to count how many line segments each SOPREF has in common with the query, however this will only allow a broad ranking of the results as there will be at most N ranking positions available for the results (from $1 \ldots N$ segments in common with the query). A more complex approach would be to compare the entire trajectory of the query with that of the trajectory associated with each SOPREF.

For any given line segment, either in the query or in the indexed line segments, there exists information relating to its exact beginning and end coordinates, and its duration (in multiples of 0.1s). This information is sufficient to interpolate coordinates between the beginning and end points of the line segments at a fidelity of 0.1s. Each line segment also has an indication of its order within the trajectory. Therefore using interpolation of individual line segments together with the order in which they occur in the ball trajectory, the entire trajectory of the ball over a SOP can be interpolated (to a fidelity of 0.1s). As semantic information is attached to each line segment (abstract team, gross and fine player archetypes), it is also possible to associate this information to the coordinates resulting from line segment interpolation, and by extension to the entire interpolated trajectory. Thus, the collection of line segments in the query, or those associated with any given SOPREF can be expanded into an ordered list of coordinates, associated with a list of equal length detailing the abstract possession information for that coordinate, as in Figure 7.44, where (xn, yn) = the n^{th} interpolated coordinate in the expanded trajectory, Tn = the abstract team possession for the nth coordinate, G_n = the gross archetype for the n^{th} coordinate, F_n = the fine archetype for the n^{th} coordinate.

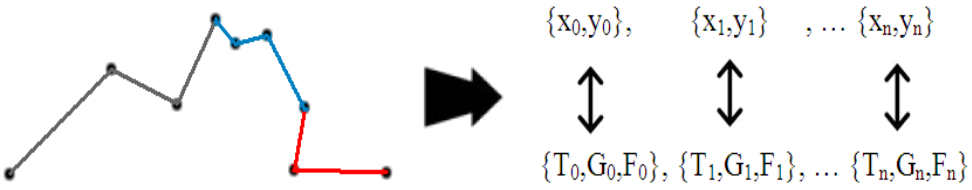

Figure 7.44 – Generating a semantic possession list which mirrors the interpolted ball trajectory

With the capability to expand a collection of line segments into an interpolated list of coordinates (together with associated possession information), it is now possible to consider exactly how to compare trajectories. The proposed similarity metric will rely on the Euclidean distance between temporally matching interpolated coordinates of each trajectory (modified by the associated semantic information). Its form is as follows. Consider the two portions of trajectories; trajectory AB and trajectory CD shown in Figure 7.45. Each trajectory portion consists of n coordinate points (six shown):

$$AB = \{(x_0, y_0) \dots (x_n, y_n)\} \tag{7.20}$$

$$CD = \{(p_0, q_0) \dots (p_n, qy_n)\} \tag{7.21}$$

Each set of coordinate points has a corresponding set of semantic associations:

$$AB_{sem} = \{(T_0^{AB}, G_0^{AB}, F_0^{AB}) \dots (T_n^{AB}, G_n^{AB}, F_n^{AB})\} \tag{7.22}$$

$$CD_{sem} = \{(T_0^{CD}, G_0^{CD}, F_0^{CD}) \dots (T_n^{CD}, G_n^{CD}, F_n^{CD})\} \tag{7.23}$$

Where T_n^{TRJ} is the abstract team, G_n^{TRJ} is the gross player archetype and F_n^{TRJ} is the fine player archetype associated with the n^{th} point in the interpolated coordinates of trajectory TRJ.

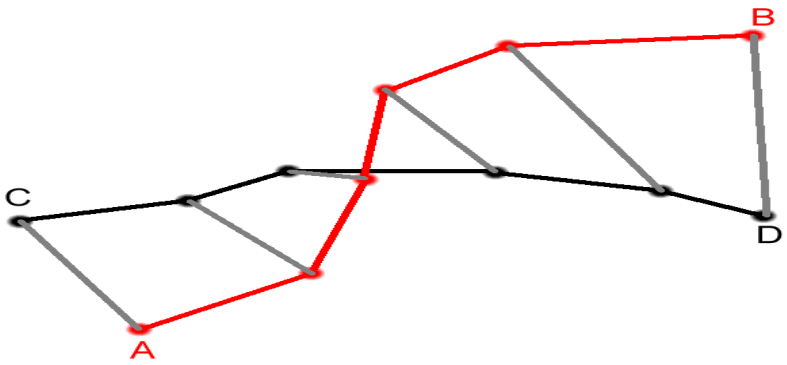

Figure 7.45 – comparison of two ball trajectories via corresponding trajectory points

The Euclidean distance between the i^{th} coordinate of AB and the corresponding ith coordinate of CD is:

$$E_i = \sqrt{(x_i - p_i)^2 + (y_i - q_i)^2} \tag{7.24}$$

The mean Euclidean distance over the two trajectory portions is:

$$\bar{E} = \frac{1}{n}\sum_{i=1}^{n} E_i \tag{7.25}$$

The associated semantic possession information associated with each coordinate have no distance metrics associated with them; they either match or they do not. A fixed penalty term for mismatches between the elements of the semantic information could be used, but this would have to be carefully weighted so as not either dominate or be dominated by the Euclidean distance. A method of variably weighting the mismatches could be to apply a multiplier penalty term to the E_i term as follows:

(1) Initial multiplier penalty $M = 1.0$
(2) For the i^{th} components of the associated semantic information
(3) If $T_i^{AB} \neq T_i^{CD}$ then increase M by P_T
(4) If $G_i^{AB} \neq G_i^{CD}$ then increase M by P_G
(5) If $F_i^{AB} \neq F_i^{CD}$ then increase M by P_F
(6) $E_i^{mod} = E_i * M$

The values assigned to P_T, P_G and P_F should be chosen to reflect how important mismatches in the team, the gross archetype and the fine archetype are when retrieving results[39]. The modified mean Euclidean distance, taking into account the semantic information is :

$$\overline{E^{mod}} = \frac{1}{n}\sum_{i=1}^{n} E_i^{mod} \tag{7.26}$$

This measure will be used to compare the interpolated trajectories of the query with that of indexed SOPs found during the sub-queries.

So to summarise the query process:

(1) The query by example is broken down into line segments
(2) Each line segment in the query is used to perform a sub-query, returning all indexed SOPs that have a similar line segment in them.
(3) The results of all sub-queries are consolidated, generating a set of SOPREFs that contain at least one similar line segment to the query.
(4) The trajectory of the query is interpolated into a semantically annotated list of coordinates
(5) For each SOPREF returned in (3), the corresponding set of line segments that belong to it are interpolated into a semantically annotated list of coordinates.

[39] For the purposes of the implementation the author chose P_T,=1, $P_G = \frac{2}{3}$, and $P_F = \frac{1}{3}$, reflecting the ordinal ranking of mismatching teams is worse than mismatching gross player prototypes which is worse than mismatching fine player prototypes.

(6) The list of annotated coordinates from (4) and (5) are operated on by the similarity metric $\overline{E^{mod}}$ to produce a similarity rating
(7) Repeat (5)-(6) until all SOPREFs have been dealt with
(8) Rank the similarity of the set of SOPREFs by their associated $\overline{E^{mod}}$

7.14.3 Search relaxation

Queries initially have no relaxation associated with them and proceed as described in the previous section. However queries may be relaxed gradually, and the relaxation scheme proceeds as follows:

(1) Initial relaxation $r = 0$, each subsequent relaxation level is $+1$, up to a maximum of $N - 1$, where N is the number of spatial prototypes in the codebook.
(2) At relaxation level 1 and higher, the constraint that both the gross player archetype and the fine player archetype need match the gross player archetype and fine player archetype of the query are relaxed to a constraint that the query can match either the gross player archetype or the fine player archetype (or both, it is not exclusive-or).
(3) At relaxation level r, where $r > 0$, the constraint that the abstract spatial beginning/end points of the indexed line segments must match precisely with the beginning/end points of the query line segment is relaxed to the beginning/end points of the indexed line segments must be a member of the set of beginning/end points formed by locating the closest r spatial prototypes to the original spatial prototype. See Figure 7.46 which shows the levels of relaxation against a prototype '111' which represents the abstract beginning of a line segment in a notional codebook. Since there are N spatial prototypes in the codebook, the maximum relaxation level possible is $N - 1$.

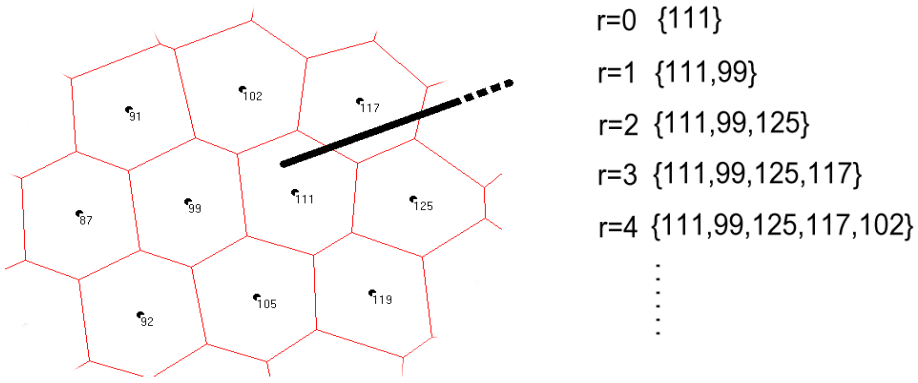

Figure 7.46 – spatial prototypes relaxation allows matching to prototypes increasingly further away from original query prototype

It should be noted that as this indexing scheme decomposes the SOP into multiple PIs (to represent individual ball trajectory line segments), each PI in the query is relaxed to the same level simultaneously.

7.15 Implementation details for Semantically augmented individual player trajectories

7.15.1 Implementation of indexing scheme

Given a SOP, each individual player trajectory can be extracted. For each trajectory the player ID can be mapped onto the corresponding gross and fine archetypes, the club ID can be mapped onto the corresponding abstract team, and the beginning and end points can be mapped onto the close spatial prototypes in the codebook. This information forms the PI (see Figure 7.47), where B = spatial prototype closest to the beginning position of the trajectory, E = spatial prototype closest to the end position of the trajectory, G = gross archetype of the player, F = fine archetype of the player, A = abstract team to which the player belongs to.

Figure 7.47 – Player trajcectory PI containg the beginning and end spatial prototypes of the player trajectory and the team and type of player indexed

The trajectory is split into x and y components, and each component is approximated using Chebyshev approximation (to a required accuracy of C_N coefficients[40]). The exact beginning and end points of the trajectory are recorded. This information forms the RM for the player trajectory (see Figure 7.48), where X = Chebyshev coefficients for x dimension of player trajectory, Y = Chebyshev coefficients for y dimension of player trajectory, P = Exact beginning coordinates of player, Q = Exact end coordinates of player. The RM is coupled with the SOPREF composed of a 3-tuple {Matchid, Halfid, Matchtime}, and both are associated with the PI (see Figure 7.49).

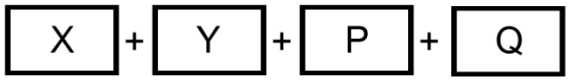

Figure 7.48 – Player trajectories RM containing Chebyshev coefficients describing the shape of the player trajectory and the exact beginning and end coordinates of the player over the SOP

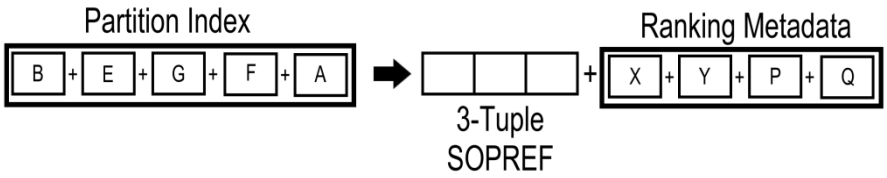

Figure 7.49 – Player trajectories indexing scheme

7.15.2 Query matching

In overview, the query should match indexed SOPs that have at least one player exhibiting a similar trajectory. The ranking of results should be contingent on both the similarity of individual player trajectory similarities between query and indexed SOPs, and on the number of such similar player trajectories. Initially queries by example are broken down into N PIs and N RM collections representing the N player trajectories present within the query SOP, as described in the previous section.

---

[40] Where C_N can be chosen by the index implementer to balance accuracy and index size.

The first stage of the query proceeds in a similar fashion to that of ball trajectory query. Each individual player trajectory from the query is posed as a sub-query and matches those indexed player trajectories that exactly share its PI (at level 0 relaxation). The result of the sub-query is a set of SOPREFs. After each player trajectory has been sub-queried, there exists a collection of sets containing SOPREFs. These sets are merged, and the result is a set of unique SOPREFs. The set of SOPREFs represents those SOPs that have at least one similar player trajectory within them. No ranking is associated with the set of SOPREFs at this stage; they are all equally similar to the query.

Consider the two player trajectories belonging to player A and B in Figure 7.50. They both have definite exact start and end points (A_s, A_e and B_s, B_e respectively) available in the RM. A suitable similarity metric for comparing the beginning/end points of two trajectories is the mean Euclidean distance function $\overline{E(A,B)}$ defined as:

$$if\ A_s = (x, y)\ and\ B_s = (p, q)$$

$$d(A_s, B_s) = \sqrt{(x-p)^2 + (y-q)^2} \tag{7.27}$$

$$\overline{E(A,B)} = \frac{d(A_s, B_s) + d(A_e, B_e)}{2} \tag{7.28}$$

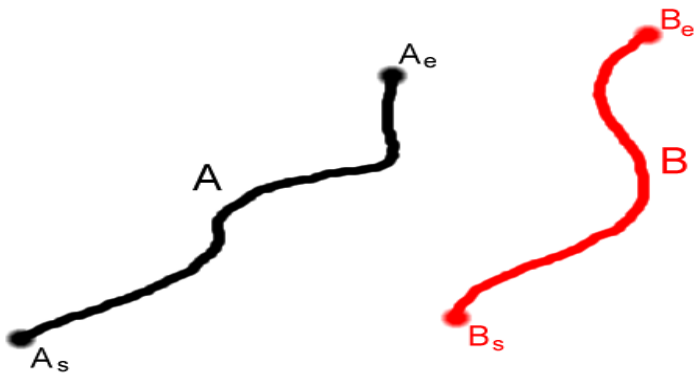

Figure 7.50 – Two player trajectories whose similarity may be determined by a suitable similarity metric

The two trajectories both have a characteristic trajectory shape, which is represented as two sets of Chebyshev coefficients per trajectory, also available in the RM. Each set of corresponding Chebyshev coefficients can be operated on by the D similarity metric defined in [49], which is:

$$C_1 = [a_0, \dots, a_m] \; and \; C_2 = [b_0, \dots, b_m]$$

$$D(C_1, C_2) = \sqrt{\frac{\pi}{2} \sum_{i=0}^{m} (a_i - b_i)^2} \tag{7.29}$$

For two trajectories A and B, the two sets of Chebyshev coefficients for each are $A_x, A_y \; and \; B_x, B_y$. Thus, the mean distance between them is given by:

$$\overline{D(A.B)} = \frac{D(A_x, B_x) + D(A_y, B_y)}{2} \tag{7.30}$$

Finally, the players themselves each are assigned an abstract team, and a gross and fine archetype which are all available in the PI. There exists no similarity metric to smoothly differentiate between player archetypes or abstract teams, so a mismatch penalty is applied as follows. If $A_g, A_f \; and \; B_g, B_f$ are the gross and fine player

archetypes for players A and B respectively, and A_t and B_t are the abstract teams for A and B then the mismatch penalty $M(A,B)$ is:

$$initial\ mismatch\ penalty\ M = 1 \tag{7.31}$$

$$If\ A_g \neq B_g\ then\ M + P_G \tag{7.32}$$

$$If\ A_f \neq B_f\ then\ M + P_F \tag{7.33}$$

$$If\ A_t \neq B_t\ then\ M + P_T \tag{7.34}$$

The constants P_G, P_F, and P_T should be chosen to reflect the relative seriousness of mismatching in gross player prototypes, fine player prototypes and teams respectively[41]. Combining all three measure into a single scalar value $sim(A,B)$ is defined as:

$$Sim(A.B) = \frac{\overline{E(A,B)} + \overline{D(A,B)}}{2} * M(A,B) \tag{7.35}$$

It should be noted that if a large number of Chebyshev coefficients are used to approximate the trajectory, then the $\overline{D(A,B)}$ term will dominate in (7.35) (as the Euclidean distance measured by $E(A,B)$ is limited to 2D space, whereas the $\overline{D(A,B)}$ term operates in a C_N-dimensional space (where C_N is the number of Chebyshev polynomials used). Weighting could be employed if this is undesirable behaviour. Consider the comparison of m player trajectories in a query (collectively Q) with that of n indexed player trajectories associated with a SOP (collectively I) as shown in Figure 7.51. In this situation, m and n need not be equal (but both are between 1 and 22). The mean similarity of the query trajectories to the indexed

[41] For the purposes of the implementation, the author chose P_G=1, $P_F = 0.5$, and $P_T = 2$ to reflect that view that mismatches in teams is worse than mismatches in gross player prototypes which is worse than mismatches in fine player prototypes.

trajectories is given by (7.36). This captures the idea that each query trajectory must match at least one indexed trajectory, but the best match for each individual trajectory is preferred. $\overline{Sim(Q,I)}$ is applied between the query and every retried index set of trajectories to produce a ranked results list.

.

$$\overline{Sim(Q,I)} = \frac{1}{m}\sum_{i=1}^{m} \min\left(\sum_{j=1}^{n} Sim(Q_i, I_j)\right) \tag{7.37}$$

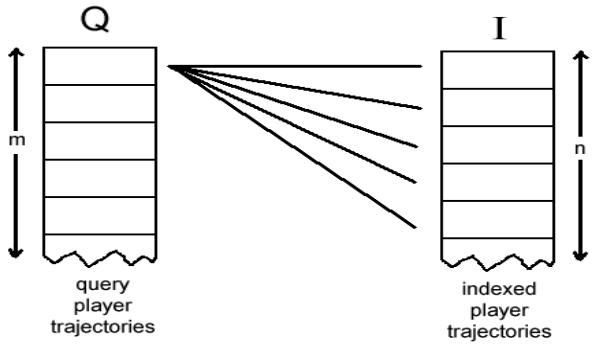

Figure 7.51 – comparison of query trajectories against all relevant indexed trajectories allows the best match to be selected for each query

7.15.3 Query relaxation

Queries initially have no relaxation associated with them and proceed as described in the previous section. However queries may be relaxed gradually, and the relaxation scheme proceeds as follows:

(1) Initial relaxation $r = 0$, each subsequent relaxation level is $+1$, up to a maximum of $N - 1$, where N is the number of spatial prototypes in the codebook.

(2) At relaxation level 1 and higher, the constraint that both the gross player archetype and the fine player archetype need match the gross player archetype and fine player archetype of the query are relaxed to a constraint that the query can match either the gross player archetype or the fine player archetype (or both, it is not exclusive-or).

(3) At relaxation level r, where $r > 0$, the constraint that the abstract spatial beginning/end points of the indexed player trajectory must match precisely with the beginning/end points of the query player trajectory is relaxed to the beginning/end points of the indexed player trajectory must be a member of the set of beginning/end points formed by locating the closest r spatial prototypes to the original spatial prototype (See Figure 7.46). Since there are N spatial prototypes in the codebook, the maximum relaxation level possible is $N - 1$.

It should be noted that as this indexing scheme decomposes the SOP into multiple PIs (to represent each player trajectory), each PI in the query is relaxed to the same level simultaneously.

7.16 Implementation details for Semantically augmented context indexing with player cliques

7.16.1 Implementation of Indexing scheme

As with the original clique indexing method, the first stage of indexing is to use clique discovery on the initial player positions, and the player's overall direction of movement for each SOP considered, producing two sets of cliques per abstract team, four sets of cliques in total.

For each clique it is already known which abstract team it belongs to (A), and what type of clique it is (T – either proximity=1 or velocity=2). The number of players in the clique is extracted (N), and the centroid of the clique (C) is calculated. The nearest spatial prototype in the codebook (S) is found. These four features form the PI for the clique (Figure 7.52).

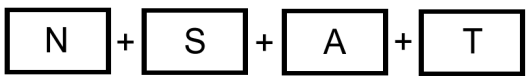

Figure 7.52 – Augmented cliques PI containing the number of players in the clique, the spatial prototype nearest to its centroid, the type of clique and the abstract team to which it belongs

The clique shape/extent properties $\bar{D}, R \; and \; A_C$ are extracted from the clique, along with two length n multi-sets containing the gross and fine archetypes corresponding to the players in the clique. This collection forms the RM (Figure 7.53)

Figure 7.53 – Augmented cliques RM containg real number attributes of the clique such as area, ration of maximum to minmum span and mean player distance, together with types of player which make up the clique

The RM is coupled with the SOPREF composed of a 3-tuple {Matchid, Halfid, Matchtime}, and both are associated with the PI (see Figure 7.54).

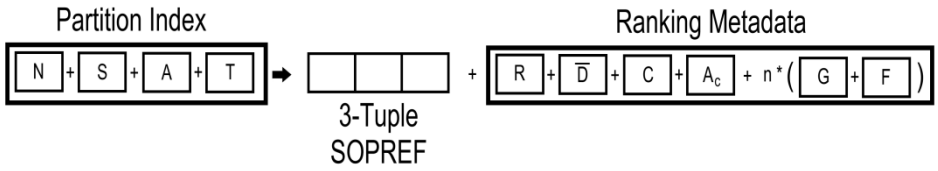

Figure 7.54 – Augmented cliques indexing scheme

7.16.2 Query matching

A valid augmented clique query should match indexed SOPs which have at least one similar clique within them (as defined by the information in the PI). The ranking of results should be contingent on both the similarity of cliques between query and indexed SOPs, and on the number of such similar cliques. Initially queries by example are broken down into N PIs and N RM collections representing the N augmented cliques present within the query SOP, as described in the previous section.

The first stage of the query locates all indexed SOPs that contain at least one similar augmented cliques to that of the query. Each individual augmented clique from the query is posed as a sub-query and matches those indexed augmented

cliques which exactly share it's PI (at level 0 relaxation). The result of the sub-query is a set of SOPREFs. The results of all sub-queries are consolidated into a single set of SOPREFs. No ranking is associated with the set of SOPREFs at this stage; they are all equally similar to the query.

To assert a ranking onto the results, similarity metric(s) must be applied between the query and each result. Consider the comparison of two cliques (C and D), as in Figure 7.55. The information describing them, which is readily available in the metadata/PI, covers the size of each clique and its abstract membership, the type of clique[42], the abstract team the clique belongs to, the exact centroid of the clique, and various statistics covering the shape of the clique[43]. All of this information will be used.

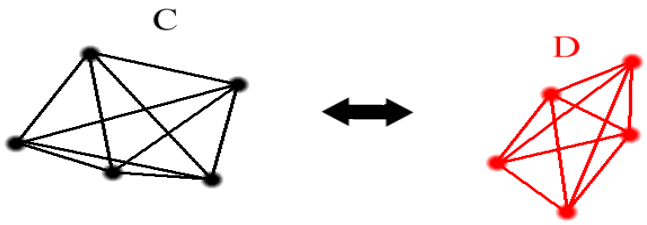

Figure 7.55 – Comparing two cliques uilising the real number clique attributes minimum to maximum player distance ratio, clique internal area and mean player distance in the clique

The *minimum to maximum player distance ratio* (C_r, D_r) the *clique internal area* (C_a, D_a) and the *mean player distance in the clique* (C_d, D_d) are all compared with the similarity metric $S(C, D)$ which is defined as :

$$S(C,D) = 1 - \frac{\min(C,D)}{\max(C,D)} \quad if\ \max(C,D) > 0 \quad else\ 1 \tag{7.38}$$

The scalar similarity produced by $S(C, D)$ is guaranteed to be in the interval [0,1]. The centroid coordinates use the Euclidean distance as the preferred similarity metric, with centroids C_c, D_c as centroids of C and D:

---

[42] Proximity or movement clique

[43] Ratio of minimum to maximum player distance, clique internal area, average distance between players

$$C_c = (x, y) \tag{7.39}$$

$$D_c = (p, q) \tag{7.40}$$

$$d(C_c, D_c) = \sqrt{(x-p)^2 + (y-q)^2} \tag{7.41}$$

The remaining attributes of *clique size*, *abstract team* and *clique type* are aggregated together to form a mismatch penalty M. If C_t , D_t are the *abstract teams*, C_s , D_s are the *clique sizes*, C_{pm} , D_{pm} are the *clique types*, C_g , D_g are the *gross player archetypes* (represented as multi-sets/bags) and C_f , D_f are *fine player archetypes* (again as multi-sets) of C and D respectively then $M(C, D)$ is defined as:

$$initial\ mismatch\ penalty\ M = 1 \tag{7.42}$$

$$If\ C_t \neq D_t\ then\ M + P_T \tag{7.43}$$

$$M{+}= |C_s - D_s| \tag{7.44}$$

$$M{+}= \frac{|C_g \cup D_g| - |C_g \cap D_g|}{|C_g \cup D_g|} \tag{7.45}$$

$$M{+}= \frac{|C_f \cup D_f| - |C_f \cap D_f|}{2|C_f \cup D_f|} \tag{7.46}$$

The constants P_T and P_{pm} should be chosen to reflect the seriousness of mismatches in the clique team and clique type respectively. The full similarity metric between two cliques $sim(C, D)$ combines all these terms:

$$sim(C,D) = \frac{S(C_r,D_r) + S(C_a,D_a) + S(C_d,D_d) + d(C_c,D_c)}{4} * M(C,D) \tag{7.47}$$

As with the player trajectory case, consider the comparison of m augmented cliques in a query (collectively Q) with that of n indexed augmented cliques associated with a SOP (collectively I) as shown in Figure 7.56.

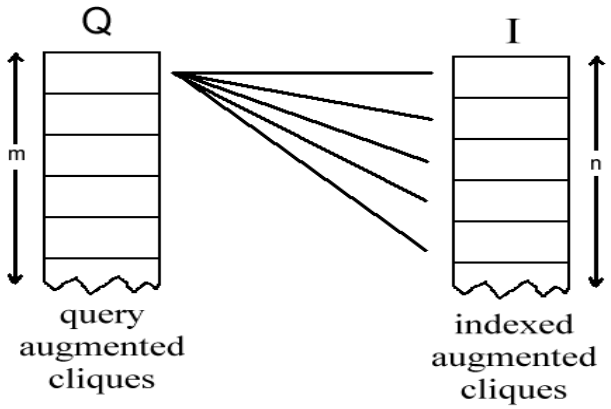

Figure 7.56 – comparison of query augmented cliques against all relevant indexed augmented cliques allows the best match to be selected for each query

The mean similarity of the query augmented cliques to the indexed augmented cliques is given by:

$$\overline{Sim(Q,I)} = \frac{1}{m}\sum_{i=1}^{m} \min\left(\sum_{j=1}^{n} sim(Q_i,I_j)\right) \tag{7.48}$$

$\overline{Sim(Q,I)}$ is applied between the query and every retrieved index set of augmented cliques to produce a ranked results list.

7.16.3 Query relaxation

Queries initially have no relaxation associated with them and proceed as described in the previous section. However queries may be relaxed gradually, and the relaxation scheme proceeds as follows:

(1) Initial relaxation $r = 0$, each subsequent relaxation level is $+1$, up to a maximum of $N - 1$, where N is the number of spatial prototypes in the codebook.

(2) At relaxation level r, where $r > 0$, the constraint that the abstract spatial centroid of the indexed player clique must match precisely with the centroid of the query player clique is relaxed to the centroid of the indexed player clique must be a member of the set of clique centroids formed by locating the closest r spatial prototypes to the original spatial prototype (See Figure 7.46). Since there are N spatial prototypes in the codebook, the maximum relaxation level possible is $N - 1$.

It should be noted that as this indexing scheme decomposes the SOP into multiple PIs (to represent each clique), each PI in the query is relaxed to the same level simultaneously.

8 Bibliography

[1]. **ProZone Home Page:** [Accessed 10th February 2012] http://www.prozonesports.com/index.html

[2]. **V. Salvo, A. Collins, B. McNeill and M. Cardinale.** Validation of ProZone ®: A new video-based performance analysis system, *International Journal of Performance Analysis in Sport*, Volume 6, Number 1, pages 108-119, University of Wales 2006.

[3]. **D. Hart, C. Needham and D. Magee.** Following your team to new extremes. *University of Leeds newsletter* Issue 508, 2005. [Accessed 10th February 2012] http://reporter.leeds.ac.uk/508/s8.htm

[4]. **E. F. Codd.** A relational model of data for large shared data banks, *Communications of the ACM,* Volume 26, Number 1, pages 64-69, 1983.

[5]. **J. Groff and P. Weinberg.** SQL: The Complete Reference, Second Edition, McGraw-Hill Osborne Media 2002.

[6]. **R. Bayer and E. McCreight.** Organization and maintenance of large ordered indexes, *Acta Informatica,* Volume 1, Number 3, pages 173-189, Springer 1972.

[7]. **A. Guttman.** R-Trees: A dynamic index structure for spatial searching, In *Proc. International Conference on Management of Data*, pages 47-57, ACM 1984.

[8]. **K. Järvelin.** Frameworks, Models and Theories in Lab IR, *In Proc. ACM SIGIR 2005 Workshop on Information Retrieval in Context*, pages 10-13, 2005.

[9]. **J. Zobel and A. Moffat.** Inverted files for text search engines, *ACM Computing Surveys*, Volume 38, Number 2, pages 6-es, 2006.

[10]. **S. Brin and L. Page.** The Anatomy of a Large-Scale Hypertextual Web Search Engine. *Computer Networks and ISDN Systems,* Volume 30, Number 1, pages 107-117, Elsevier 1998.

[11]. **L. Page, S. Brin, R. Motwani and T. Winograd.** The PageRank Citation Ranking: Bringing Order to the Web. *Technical Report 422*, Stanford University InfoLab, 1999.

[12]. **L. Ahn and L. Dabbish.** Labeling images with a computer game. *In Proc. SIGCHI conference on Human factors in computing systems*, pages 319-326, 2004.

[13]. **K. Pastra, H. Saggion and Y. Wilks.** Extracting relational facts for indexing and retrieval of crime-scene photographs. *Applications and Innovations in Intelligent Systems*, Volume 16, Number 5, pages 121-134, Springer 2002.

[14]. **M. Hepagesle.** Independence and commitment: assumptions for rapid training and execution of rule-based POS taggers. *In Proc. 38th Annual Meeting of the Association for Computational Linguistics,* pages 278-281, 2000.

[15]. **E.J. Guglielmo and N.C. Rowe.** Natural-language retrieval of images based on descriptive captions. *ACM Transactions on Information Systems*, Volume 14, Number 3, pages 237-267, 1996.

[16]. **S. Haas.** A feasibility study of the case hierarchy model for the construction and porting of natural language interfaces. *Information Processing & Management*, Volume 26, Number 5, pages 615–628, Elsevier 1990.

[17]. **T. Rose, D. Elworthy, A. Kotcheff and A. Clare.** ANVIL: a system for the retrieval of captioned images using NLP techniques. *In Proc. CIR2000, 3rd UK Conference in Image Retrieval*, BCS 2000.

[18]. **A. Gulli and A. Signorini.** The indexable web is more than 11.5 billion pages. *In Proc. WWW '05: Special interest tracks and posters of the 14th international conference on World Wide Web*, pages 902-903, ACM 2005.

[19]. **R. Baeza-Yates and B. Ribeiro-Neto.** *Modern Information Retrieval*, Addison Wesley 1999.

[20]. **W Mao and W Chu.** Free-text medical document retrieval via phrase-based vector space model. *In Proc. AMIA Symposium,* pages 489-493, 2002.

[21]. **A. Aizawa.** An information-theoretic perspective of tf—idf measures. *Information Processing and Management*, Volume 39, Number 1, pages 45-65, Elsevier 2003.

[22]. **C. Kumar, A. Gupta, M. Batool and S. Trehan.** Latent semantic indexing-based intelligent information retrieval system for digital libraries. *Journal of Computing and Information Technology*, Volume 13, Number 3, pages 191-196, University of Zagreb 2006.

[23]. **M. Swain and D. Ballard.** Color indexing. *International Journal of Computer Vision*, Volume 7, Number 1, pages 11-32, 1991.

[24]. **S. Sural, G. Qian and S. Pramanik.** A histogram with perceptually smooth color transition for image retrieval. *In Proc. Fourth Int. Conf. on Computer Vision, Pattern Recognition and Image Processing*, pages 664-667, 2002.

[25]. **J. Huang, S.R. Kumar, M. Mitra, W. Zhu and R. Zabih.** Image Indexing Using Color Correlograms. *In Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition,* pages 762-768 , 1997.

[26]. **Y. Gong, G. Proietti and C. Faloutsos.** Image Indexing and Retrieval Based on Human Perceptual Color Clustering. *In Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 578-583, 1998.

[27]. **F. Liu and R. Picard.** Periodicity, directionality and randomness: Wold features for image modelling and retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Volume 18, Number 7, pages 722-733, 1996.

[28]. **J. Han and K. Ma.** Rotation-invariant and scale-invariant Gabor features for texture image retrieval. *Image and Vision Computing*, Volume 25, Number 9, pages 1474–1481, Elsevier 2007.

[29]. **J. Li and J.Z. Wang.** Automatic linguistic indexing of pictures by a statistical modeling approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Volume 25, Number 9, pages 1075-1088, 2003.

[30]. **G. Csurka, C.R. Dance , L. Fan , J. Willamowski and C. Bray.** Visual categorization with bags of keypoints. *ECCV Workshop on Statistical Learning in Computer Vision*, pages 1-22, 2004.

[31]. **K. Schmid and C. Mikolajczyk.** An Affine Invariant Interest Point Detector. *In Proc. 7th European Conference on Computer Vision*, pages 1-7, 2002.

[32]. **D.G. Lowe.** Object recognition from local scale-invariant features.**.** *In Proc. International Conference on Computer Vision*, pages 1150–1157, 1999.

[33]. **G. Bouchard and B. Triggs.** Hierarchical part-based visual object categorization. *In Proc. IEEE Computer Vision and Pattern Recognition,* pages 710-715, 2005.

[34]. **I. Ahmad and W.I. Grosky.** Indexing and retrieval of images by spatial constraints. *Journal of Visual Communication and Image Representation*, Volume 14, Number 3, pages 291–320, Elsevier 2003.

[35]. **A. Rauber, E. Pampalk and D. Merkl.** Content-based music indexing and organization. *In Proc. 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 409-410, 2002.

[36]. **T. Kohonen.** The self-organizing map, *Proceedings of the IEEE*, Volume 78, Number 9, pages 1464-1480, 1990.

[37]. **G. Tzanetakis, A. Ermolinskyi and P. Cook.** Pitch Histograms in Audio and Symbolic Music Information Retrieval. *In Proc. Third International Conference on Music Information Retrieval: ISMIR*, pages 31-38, 2002.

[38]. **R. Typke, R. C. Veltkamp and F. Wiering:.** Searching notated polyphonic music using transportation distances, *In Proc. ACM Multimedia Conference*, pages 128-135, 2004.

[39]. **L. Prechelt and R. Typke.** An interface for melody input. *ACM Transactions on Computer-Human Interaction*, Volume 8, Number 2, pages 133-149, 2001.

[40]. **S. Porter, M. Mirmehdi and B. Thomas.** Detection and classification of shot transitions. *In Proc. 12th British Machine Vision Conference*, pages 73–82, 2001.

[41]. **J. Meng, Y. Juan and S. Chang.** Scene Change Detection in a MPEG Compressed Video Sequence. *In Proc. IS&T/SPIE '95 Digital Video Compression: Algorithms and Technologies*, pages 14-25, 1995.

[42]. **W. Zeng, W. Gao and D. Zhao.** Video indexing by motion activity maps. *In Proc. IEEE International Conference on Image Processing*, pages 912-915, 2002.

[43]. **J. Davis.** Recognizing movement using motion histograms. *Technial Report 487*, MIT Media Lab, 1999.

[44]. **D. DeMenthon and D. Doermann.** Video Retrieval using Spatio-Temporal Descriptors. *In Proc. of the eleventh ACM international conference on Multimedia*, pages 508-517, 2003.

[45]. **C. Snoek and M. Worring.** Multimedia Event-Based Video Indexing using Time Intervals. *IEEE Transactions on Multimedia*, Volume 7, Number 4, pages 638-647, 2005.

[46]. **J.F. Allen,** Maintaining knowledge about temporal intervals. *Communications of the ACM*, Volume 26, Number 11, pages 832–843, 1983.

[47]. **W. Chen and S.F. Chang.** Motion Trajectory Matching of Video Objects. *In Proc. SPIE Storage and Retrieval for Media Databases*, pages 544-553, 2000.

[48]. **E. Sahouria and A. Zakhor.** A Trajectory Based Video Indexing System For Street Surveillance. *In Proc. IEEE International. Conference on Image Processing*, pages 24-28, 1999.

[49]. **Y. Cai and R. Ng.** Indexing Spatio-Temporal Trajectories with Chebyshev Polynomials. *In Proc. 2004 ACM SIGMOD international conference on Management of data*, pages 599-610, 2004.

[50]. **E.W. Weisstein.** Chebyshev Polynomial of the First Kind. *From MathWorld: A Wolfram Web Resource.* [Accessed 10th February 2012] http://mathworld.wolfram.com/ChebyshevPolynomialoftheFirstKind.html.

[51]. **E.W. Weisstein.**. Chebyshev Approximation Formula. *From MathWorld: A Wolfram Web Resource.* [Accessed 10th February 2012] http://mathworld.wolfram.com/ChebyshevApagesroximationFormula.html .

[52]. **F.I. Bashir. A.A. Khokhar and D. Schonfeld.** Real-Time Motion Trajectory-Based Indexing and Retrieval of Video Sequences. *IEEE Transactions on Multimedia*, Volume 9, Number 1, pages 58-65, 2007.

[53]. **I.T. Jolliffe,** Principal Component Analysis Second Edition, Springer 2002.

[54]. **A.Y. Ng, M.I. Jordan and Y. Weiss.** On Spectral Clustering: Analysis and an algorithm, *Advances in Neural Information Processing Systems*, pages 849-856, MIT Press 2001.

[55]. **G. Navarro.** A guided tour to approximate string matching, *ACM Computing Surveys*, Volume 33, Number 1, pages 31-88, 2001.

[56]. **X. Ma, F. Bashir, A. Khokhar and D. Schonfeld.** Tensor-Based Multiple Object Trajectory Indexing and Retrieval, *In Proc. IEEE International Conference on Multimedia*, pages 341-344 , 2006.

[57]. **R.A. Harshman and M.E. Lundy**. PARAFAC: Parallel factor analysis, *Computational Statistics & Data Analysis*, Volume 18, Number 1, pages 39-72, Elsevier 1994.

[58]. **N. Beckmann, H. Kriegel, R. Schneider and B. Seeger.** The R*-Tree: An effecient and robust access method for points and rectangles. *In Proc. ACM Management of Data (SIGMOD),* pages 220-231, 1990.

[59]. **T.K. Sellis, N. Roussopoulos and C. Faloutsos.** The R+-Tree: A dynamic index for multi-dimensional objects. *In Proc. IEEE International Conference on Very Large Data Bases*, pages 507-518, 1987.

[60]. **Y. Tao and D. Papadias.** MV3R-Tree: A spatio-temporal access method for timestamp and interval queries. *In Proc. IEEE International Conference on Very Large Data Bases*, pages 431-440, 2001.

[61]. **M. Hadjieleftheriou, G. Kollios, V. Tsotras and D. Gunopulos.** Indexing Spatio Temporal Archives. *The VLDB Journal*, volume 15, Number 2, pages 143-164, Springer 2006.

[62]. **V. Almeida and R. Güting.** Indexing the Trajectories of Moving Objects in Networks. *Geoinformatica*, Volume 9, Number 1, pages 33-60, Springer 2005.

[63]. Defn: Behavior, Answers.com. The American Heritage® Dictionary of the English Language, Fourth Edition, Houghton Mifflin Company 2004 [Accessed 10th February 2012]. http://www.answers.com/topic/behavior.

[64]. **M. Wooldridge and N.R. Jennings.** Intelligent Agents: Theory and Practice. *Knowledge Engineering Review*, Volume 10, Number 2, pages 115-152, Cambridge University Press 1995.

[65]. **M. Kifer, G. Lausen and G. . Wu.** Logical Foundations of Object-Oriented and Frame-Based Languages. *Journal of the ACM*, Volume 42, Number 4, pages 741-843, 1995.

[66]. **N.R. Jennings.** On agent-based software engineering. *Artificial Intelligence*, Volume 117, Number 2, pages 277-296, Elsevier 2000.

[67]. **J. H. Holland and J.H. Miller.** Artificial Adaptive Agents in Economic Theory. *The American Economic Review*, Volume 81, Number 2, pages 365-371, AEA 1991.

[68]. **W.B. Arthur.** On designing economic agents that behave like human agents. *Journal of Evolutionary Economics*, Volume 3, Number 1, pages 353-359, Springer 1993.

[69]. **P. Maes.** Artificial life meets entertainment: lifelike autonomous agents. *Communications of the ACM*, Volume 38, Number 11, pages 108-114., 1995.

[70]. **R. Moller, D. Lambrinos, R. Pfeifer, T. Labhart and R. Wehner.** Modeling Ant Navigation with an Autonomous Agent. *In Proc. 5th Int. Conf. Simulation of Adaptive Behavior*, pages 185-194, 1998.

[71]. **R. Schoonderwoerd, O. Holland and J. Bruten.** Ant-like agents for load balancing in telecommunications networks. *In Proc. first international conference on Autonomous agents*, pages 209-216, 1997.

[72]. **E. Bonabeau, F. Henaux, S. Guérin, D. Snyers, P. Kuntz and G. Theraulaz.** Routing in telecommunications networks with ant-like agents. *In Proc. second international workshop on Intelligent agents for telecommunication applications*, pages 60-71, 1999.

[73]. **J. Neumann.** Theory of Self-Reproducing Automata, University of Illinois Press 1966.

[74]. **M. Gardner.** Mathematical Games: The fantastic combinations of John Conway's new solitaire game "life", *Scientific American*, Issue 223, pages 120-123, 1970.

[75]. **R. Brooks.** Intelligence Without Reason. *Computers and Thoughts – IJCAI '91*, pages 569-595, Morgan Kaufmann 1991.

[76]. **R. Brooks**. Intelligence without representation. *Artificial Intelligence*, Volume 47, Number 1, pages 139-159, Elsevier 1991.

[77]. **R. Brooks**. New approaches to robotics, *Science*, Volume 253, Number 5025, pages 1227-1232, AAAS 1991.

[78]. **C.W. Reynolds.** Flocks, herds, and schools: a distributed behavioral model. *ACM SIGGRAPH Computer Graphics*, Volume 21, Number 4, pages 25-34, 1987.

[79]. **H.Guerra and S. Fallah-Seghrouchni.** Learning in BDI Multi-agent Systems. *In Proc. CLIMA IV – Computational Logic in Multi-Agent Systems*, pages 39-44, 2004.

[80]. **M. Dorigo, E. Bonabeau and G. Theraulaz.** Swarm intelligence: from natural to artificial systems, Oxford University Press 1999.

[81]. **M. Dorigo, E. Bonabeau and G. Theraulaz.** Ant algorithms and stigmergy. Future Generation Computer Systems, Volume 16, Number 9, pages 851-871, Elsevier 2000.

[82]. **C. Gagne, M. Gravel and W. Price.** Solving real car sequencing problems with ant colony optimization. *European Journal of Operational Research*, Volume 174, Number 3, pages 1427-1448, Elsevier 2006.

[83]. **D. Martens, M. Haesen, R. Vanthienen, J. Snoeck and M. Baesens.** Classification With Ant Colony Optimization. *IEEE Transactions on Evolutionary Computation*, Volume 11, Number 5, pages 651-665, 2007.

[84]. **A.R. Malisia and H.R. Tizhoosh.** Applying Ant Colony Optimization to Binary Thresholding. *In Proc. IEEE International Conference on Image Processing*, pages 2409-2412, 2006

[85]. **S.S. Intille and A.F. Bobick.** A framework for recognizing multi-agent action from visual evidence. *In Proc. AAAI '99 national conference on Artificial Intelligence*, pages 518-525, 1999.

[86]. **R. Nakanishi, J. Bruce, K. Murakami, T. Naruse and M. Veloso.** Cooperative 3-Robot Passing and Shooting in the RoboCup Small Size League. *Lecture Notes In Artificial Intelligence*, Volume 4434, pages 418-425, Springer 2006.

[87]. **M.Lotzsch, J. Bach, H. Burkhard and M. Jungel.** Designing Agent Behavior with the Extensible Agent Behavior Specification Language XABSL. *In Proc. RoboCup-2002 Symposium*, pages 114-124, 2002.

[88]. **A. Pietro, L. While and L. Barone.** Learning In RoboCup Keepaway Using Evolutionary Algorithms. *In Proc. Genetic and Evolutionary Computation Conference*, pages 1065-1072, 2002.

[89]. **T. Nakashima, M. Takatani. M. Udo and H. Ishibuchi.** An evolutionary apagesroach for strategy learning in RoboCup soccer. *In Proc. IEEE International Conference on Systems, Man and Cybernetics*, pages 2023–2028, 2004.

[90]. **H. Dee.** Explaining Visible Behaviour. PhD thesis, University of Leeds 2005.

[91]. **N. Johnson.** Learning Object Behaviour Models. PhD thesis, University of Leeds 1998.

[92]. **A. Galata, N. Johnson and D. Hogg.** Learning Variable Length Markov Models of Behaviour. *Computer Vision and Image Understanding*, Volume 81, Number 3, pages 398–413, Elsevier 2001.

[93]. **D.R. Magee.** Machine Vision Techniques for the Evaluation of Animal Behaviour, PhD thesis, University of Leeds 2000.

[94]. **D.C. Brogan and Y. Loiti`ere.** Data-Driven Generation of Simulated Soccer Behaviors. *In Proc. AAMAS '02 first international joint conference on Autonomous agents and multiagent systems*, pages 1391-1392, 2002.

[95]. **H. Kitano, M. Tambe, P. Stone, M. Veloso, S. Coradeschi, E. Osawa, H. Matsubara, I. Noda and M. Asada** The RoboCup Synthetic Agent Challenge**.** *Lecture Notes in Computer Science*, Volume 1395, pages 62-73, Springer 1998.

[96]. **G. Kaminka, M. Fidanboylu, A. Chang and M. Veloso.** Learning the sequential coordinated behavior of teams from observations. *In Proc. RoboCup-2002 Symposium*, pages 111-125, 2002.

[97]. **S. Crone, J.Guajardo and R. Weber.** A study on the ability of Support Vector Regression and Neural Networks to Forecast Basic Time Series Patterns, *Artificial Intelligence in Theory and Practice*, Volume 217, pages 149-158, Springer 2006.

[98]. **H.Takahashi, N. Horibe,M. Shimada and T.Ikegami.** Analyzing the House Fly's Exploratory Behavior with Autoregression Methods. *Journal of the Physical Society of Japan*, Volume 77, Number 8, pages 084802.1-084802.6, The Physical Society of Japan 2008.

[99]. **P. Sutton and R.S. Stone.** Scaling reinforcement learning toward RoboCup. *In Proc. 18th International Conf. on Machine Learning,* pages 537-544, 2001.

[100]. **J. Li A. Lilienthal, T. Martinez-Marin and T. Duckett.** Q-RAN: A Constructive Reinforcement Learning Apagesroach for Robot Behavior Learning. *In Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems,* pages 2656-2662, 2006.

[101]. **K. Alsabti.** An efficient k-means clustering algorithm. *In Proc. IPPS/SPDP Workshop on High Performance Data Mining*, pages 881-892, 1998.

[102]. **T. Kurita.** An Efficient Agglomerative Clustering Algorithm for Region Growing, *In Proc. IAPR Workshop on Machine Vision Applications*, pages 210-213, 1991.

[103]. **P. Fjallstrom.** Algorithms for graph partitioning: A survey. *Articles in Computer and Information Science,* Volume 3, Number 10, Linköping University Electronic Press 1998.

[104]. **G. Karypis and V. Kumar.** A Parallel Algorithm for Multilevel Graph Partitioning and Sparse Matrix Ordering. *In Proc. 10th Intl. Parallel Processing Symposium*, pages 314-319, 1996.

[105]. **M. Hung and D. Yang.** An efficient Fuzzy C-Means clustering algorithm. *In Proc. 2001 IEEE International Conference on Data Mining,* pages 225-232 , 2001.

[106]. **S. Wasserman, K. Faust, D. Iacobucci and M. Granovetter.** Social Network Analysis: Methods and Applications.Cambridge University Press 1994.

[107]. **D. Du and P. Pardalos.** Handbook of Combinatorial Optimization, Springer 1999.

[108]. **M. Hansen and B. Yu.** Model Selection and the Principle of Minimum Description Length. *Journal of the American Statistical Association*, Volume 96, Number 454, pages 746-774, 2001.

[109]. **M. Erp and L. Schomaker.** Variants of the Borda Count Method for Combining Ranked Classifier Hypotheses. *In Proc. 7th International Workshop on Frontiers in Handwriting Recognition*, pages 443-452, 2000.

[110]. **X. Olivares, M. Ciaramita and R. Zwol.** Boosting image retrieval through aggregating search results based on visual annotations. *In Proc. 16th ACM international conference on Multimedia*, pages 189-198, 2008.

[111]. **R. Fagin, R. Kumar and D. Sivakumar.** Efficient Similarity Search and Classification via Rank Aggregation. *In Proc. 2003 ACM SIGMOD International Conference on Management of Data*, pages 301-312, 2003.

[112]. **C. Dwork, Ravi Kumar, M. Naor and D. Sivakumar.** Rank aggregation methods for the Web. *In Proc. 10th international conference on World Wide Web*, pages 613-622, 2001.

[113]. **J. Lee.** Analysis of multiple evidence combination. *In Proc. 20th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 267-276, 1997.

[114]. **C. Vogt and G. Cottrel.** Fusion via a Linear Combination of Scores. Information Retrieval, Volume 1, Number 3, pages 151-173, Springer 1999.

[115]. **R. Manmatha, T. Rath and F. Feng.** Modeling Score Distributions for Combining the Outputs of Search Engines. *In Proc. 24th Annual International ACM SIGIR Conference on Research and Development in information*, pages 267-275, 2001.

[116]. **L. Smith.** A tutorial on Principal Components Analysis. *Systems Neurobiology Laboratory, Salk Institute for Biological Studies*, 2005.

[117]. **D. Tzovaras and M. Strintzis.** Use of nonlinear principal component analysis and vector quantization for image coding. *IEEE Transactions on Image Processing*,. Volume 7, Number 8, pages 1218–1223, 1998.

[118]. **Various.** MySQL 5.0 Reference Manual. [Accessed 10th February 2012] http://dev.mysql.com/doc/refman/5.0/en/.

[119]. **M. Giacomo.** MySQL: Lessons Learned on a Digital Library. *IEEE Software* Volume 22 , Number 3, pages 10–13, 2005.

[120]. **S. Kirkpatrick, C.D. Gelatt and M.P.Vecchi.** Optimization by Simulated Annealing, *Science*, Volume 220, Number 4598, pages 671-680, AAAS 1983

[121]. **R. Dunne and N. Campbell.** On the pairing of the softmax activation and cross-entropy penalty functions and the derivation of the softmax activation function. *In Proc. 8th Australian conference on neural networks*, pages 181-185, 1997.

[122]. **H. Henrique, E. Lima and D. Seborg.** Model structure determination in neural network models. *Chemical Engineering Science,* Volume 55, Number 22, Pages 5457-5469, Elsevier 2000.

[123]. **Y. Cun, J. Denker and S. Solla.** Optimal brain damage. *Advances in neural information processing systems*, Volume 2, pages 598-605, Morgan Kaufmann 1990.

[124]. **B. Hassibi, D. Stork and G. Wol.** Optimal brain surgeon and general network pruning. *Technical Report 9235*, RICOH California Research Center, Menlo Park, CA, 1992.