

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Vision and Knowledge Representation Methodologies for Game Analysis

Catarina Maria Brito de Noronha Santiago



Thesis submitted in partial fulfilment of the requirements for the degree of Doctor in
Informatics Engineering of the University of Porto

Supervisor: Professor Dr. Armando Jorge Miranda de Sousa

Co-Supervisor: Professor Dr. Luís Paulo Reis

November, 2015

Vision and Knowledge Representation Methodologies for Game Analysis

Catarina Maria Brito de Noronha Santiago

Thesis submitted in partial fulfilment of the requirements for the degree of
Doctor in Informatics Engineering of the University of Porto

November, 2015

Resumo

Nos últimos anos tem-se assistido a um crescente interesse por parte da sociedade em aprofundar o conhecimento relativo ao desporto, bem como uma crescente necessidade em extrair informação útil de eventos desportivos. O interesse neste tipo de informação está patente em diversos sectores, quer para melhorar a experiência dos adeptos face aos jogos, quer para melhorar o conhecimento ao nível do desempenho físico e tático dos jogadores/ das equipas por forma a ter mecanismos que permitam um treino adequado e mais focado ou ainda como ferramenta para auxílio na educação de futuros profissionais do desporto. Para colmatar tais limitações, esta dissertação foi estruturada com o intuito de estudar, desenhar e implementar metodologias que permitam o desenvolvimento de um sistema de visão capaz de realizar análise complexa de jogos desportivos.

Com vista a alcançar este objectivo foi proposto e testado um sistema com múltiplas câmaras aplicado ao caso específico do andebol. A modularidade da solução proposta permitiu a sua validação em diferentes eventos com diferentes requisitos em termos de número de câmaras.

Numa primeira fase é efectuada a gravação dos jogos, que são posteriormente processados para extrair as posições dos jogadores em coordenadas do campo recorrendo a metodologias de detecção e seguimento.

A detecção dos jogadores é feita através de um modelo das cores do equipamento dos jogadores baseado em Lógica Difusa. Este modelo é inicialmente obtido com a ajuda de um operador, sendo posteriormente actualizado automática e dinamicamente no decorrer do jogo para garantir que a cada instante reflecte as cores das equipas que são afectadas pelas condições de iluminação. A natureza Difusa do modelo é dupla, na medida em que permite atribuir aos pixéis graus de pertença a cada equipa, mas também iniciar e controlar a adaptabilidade dos modelos de cor.

A detecção dos jogadores propriamente dita começa pela detecção dos pixéis que constituem o "primeiro plano" através da subtracção do fundo. Este fundo resulta de uma imagem vazia do campo que é constantemente actualizada. Ao efectuar esta subtracção inicial consegue-se limitar a área de campo a ser pesquisada em mais detalhe e como tal reduzir consideravelmente o tempo de processamento.

As áreas que resultam da subtracção do fundo são depois processadas usando os modelos de cor, e a cada pixel é atribuído um grau de pertença a cada equipa. Pixéis adjacentes são então agrupados e as zonas resultantes filtradas através de restrições de tamanho. A detecção é ainda melhorada através de Filtros de Kalman, um filtro por cada jogador.

Dos passos anteriores resultam as posições dos jogadores em coordenadas do campo, que são depois utilizadas para extrair informação de baixo nível relativa ao desempenho físico dos jogadores (distância percorrida e velocidade), bem como à identificação de áreas preferenciais de ataque e defesa, ou a corredores usados para transição de fase.

Esta informação base é ainda usada por algoritmos mais complexos que permitem identificar a fase do jogo (ataque, defesa, transição ofensiva-defensiva e transição defensiva-ofensiva) e a tática defensiva adoptada (defesa a uma linha, duas linhas ou três linhas).

A fase de jogo é obtida através de um classificador Difuso do tipo Mandani que tendo em conta a fase de jogo anterior, a velocidade e posição longitudinais da equipa, determina a fase de jogo actual. Neste classificador, as entradas são primeiro fuzzificadas para então passarem por um conjunto de regras "SE-ENTÃO" que determinam a fase de jogo actual. A defesa táctica é identificada por um método de minimização de erro que aproxima as posições dos jogadores a círculos à volta da área de baliza.

Por último, é proposto um modelo do jogo de andebol baseado numa rede de Petri hierárquica e colorida que inclui três camadas que incrementalmente adicionam mais detalhe ao modelo. A primeira camada traduz uma vista genérica e indica se o jogo está a decorrer ou se está parado, a segunda camada pormenoriza cada um destes modos de jogo e a terceira modela as interacções entre jogadores durante situações de ataque e defesa.

Testes efectuados durante uma competição oficial de andebol demonstraram a validade da arquitectura proposta, com taxas de exactidão de detecção de jogadores acima de 95%. Adicionalmente, os testes validaram que o uso de espaços de cor adaptativos baseados na metodologia Difusa proposta conseguem caracterizar melhor as propriedades de cor dos equipamentos dos jogadores, o que contribui para taxas de seguimento mais elevadas.

Os testes efectuados ao classificador Difuso demonstraram que é capaz de identificar alterações de fase de jogo com uma exactidão acima de 86% e o início de fase tem um desalinho temporal de em média 1.9s. Por outro lado, a estratégia defensiva é detectada com uma exactidão acima dos 85%.

Para além desta validação, a metodologia proposta foi ainda usada para aferir o esforço físico de árbitros de andebol durante competições oficiais, bem como para monitorizar atletas durante sessões de treinos de basquetebol.

Abstract

Over the last years, society has increased the interest for having more knowledge and useful information from sports events. The interest on such information can arise from different sectors, either to improve the fans' immersion into the game, to enhance the knowledge from a performance or tactical point of view in order to better train the athletes or even to provide a more adequate background when teaching students. Having these concerns in mind, this thesis was formulated with the purpose of studying, designing and implementing methodologies to support the development of a vision based system to perform high level game analysis.

In order to achieve this goal, a multiple camera architecture applied to the specific handball case was proposed and tested in different sports halls. The modularity of the proposed system was validated under different configurations (number of cameras).

On the proposed system, the games are first recorded and only then processed in order to extract the players' coordinates on the field using detection and tracking methodologies.

Player detection is achieved using a Fuzzy inspired model of the players' vests colours. An initial model is obtained with the aid of an operator intervention. This model is then dynamically and automatically adapted to the light conditions that affect the colour properties. The Fuzzy nature of the model is twofold: it allows categorizing pixels with a given belonging degree, but also to trigger and control the adaptability of the colour models.

The players' detection starts by initially identifying foreground pixels by means of background subtraction. This background model results of an initial empty image of the court that is dynamically updated. By first removing the background it is possible to narrow the possible areas where players can be, which accelerates the overall processing time.

The obtained foreground areas are then scanned using the aforementioned colour model and each pixel is categorized into each of the teams with a given belonging degree. Adjacent pixels are afterwards grouped and the generated areas are filtered using area constraints. Detection is further enhanced with tracking using a vector of Kalman filters, one filter per player.

The players' positions on the field, obtained from this detection and tracking methodology, are then used to obtain low level information that allows to evaluate the players' physical effort during the match (distance travelled and speeds achieved), as well as to define preferable areas of attack, defence and transitional corridors.

More complex algorithms are used to identify the game phase (attack, defence, defensive offensive transition and offensive defensive transition) and the defensive team formation (one defensive line, two defensive lines or three defensive lines).

The game phase algorithm is based on a Mandani Fuzzy Logic classifier that uses as inputs the previous game phase and the longitudinal velocity and position of the teams to evaluate the current game phase. These inputs are fuzzified and afterwards pass through an "IF-THEN" rule database in order to determine the current game phase. The defensive system is identified based on an error minimization algorithm, which approximates the players' positions to circles around the goal area.

Finally, a handball game model based on a Hierarchical Coloured Petri Net is proposed. The model includes three layers that gradually add more detail. On the first layer it is possible to have a generic view of whether the game is stopped or on, the second layer details each of these modes and the deepest layer provides information about the players' interactions during attack and defensive situations.

Tests conducted during a handball professional competition proved the architecture concept with an accuracy for player tracking above 95%. Moreover, tests validated that by using adaptive colour subspaces based on the Fuzzy inspired methodology it was possible to better define the teams' colour properties, which contributed to the high tracking rates.

Tests on the phase detection methodology showed that the Fuzzy Logic classifier correctly identifies phase change events with an accuracy above 86% and a time misalignment of 1.9s. On the other hand, the defensive strategy is classified with an accuracy above 85%.

Besides the initial player study purpose, the system was also used to assess the physical effort of handball referees during competitions and athletes during basketball training sessions.

Acknowledgements

First of all, there are no words to express my gratitude towards my supervisor, Prof. Dr. Armando Jorge Sousa, for his support, patience, knowledge, interesting discussions and above all friendship. I am also grateful to my co-advisor, Prof. Dr. Luís Paulo Reis, whose advice, knowledge and support helped me during this process.

I would also like to thank Prof. Luísa Estriga and Marco Guimarães from the Faculty of Sports of the University of Porto for their knowledge and support, as well as their valuable aid on gathering videos and classifying them for comparison with the automatic methods. This work could not have been done without the support from the Académico Futebol Clube and the Portuguese Handball Federation, which allowed the recording of several official games.

Many thanks to my laboratory colleagues, Luísa Vaz Matos and Paulo Costa, for their support, funny discussions and coffee breaks during these years. To my PRODEI colleagues, especially the ones who followed my work in a more closed way and became good friends.

A special thanks to Prof. Adriano Carvalho for allowing me to settle in his laboratory.

To Fundação Calouste Gulbenkian by the support given through the PhD scholarship (ref. 104410).

To my family, without whom I could have never completed this phase. For your unconditional support, encouragement and guidance throughout my life.

Finally, to all of those, that during this journey crossed my life and helped me, one way or another, carry this *giant* despite the rough path.

Catarina B. Santiago

*“Our passion for learning ...
is our tool for survival.”*

Carl Sagan in *Cosmos* (1980, 1985), 230-231

Contents

1	Introduction	1
1.1	Motivation and Goals	2
1.2	Thesis Statement and Objectives	2
1.3	Scientific Contributions	3
1.4	Document Structure	3
2	Methodologies and Fundamentals	5
2.1	Video Segmentation	5
2.1.1	Spatial Video Segmentation	5
2.1.2	Object Tracking	11
2.2	Image Recognition	19
2.2.1	Bayesian Classifiers	20
2.2.2	Non-Bayesian Classifiers	20
2.2.3	Multiple Classifiers	26
2.3	Knowledge Modelling and Representation	27
2.3.1	Knowledge Representation Formalisms	28
2.3.2	Ontologies	29
2.3.3	Petri Nets	31
2.3.4	CPN Tools	32
2.4	Summary and Conclusions	37
3	Related Research	39
3.1	Player Detection and Tracking	40
3.1.1	Intrusive Systems	40
3.1.2	Non-Intrusive Systems	44
3.1.3	Conclusions	51
3.2	Ball Detection and Tracking	54
3.2.1	Intrusive Systems	54
3.2.2	Non-Intrusive Systems	55
3.2.3	Conclusions	58
3.3	Game Analysis	60
3.3.1	Hand Annotation	60
3.3.2	Automatic Analysis	61
3.3.3	Conclusions	64
3.4	Summary and Conclusions	65

4	Methodology	67
4.1	Proposed Architecture	67
4.1.1	Acquisition System	68
4.1.2	Processing System	69
4.1.3	Visualizer and Annotator	70
4.2	Player Detection	71
4.2.1	Colour calibration	72
4.2.2	Background Subtraction	73
4.2.3	Team identification	74
4.3	Player Tracking	78
4.4	High Level Game Analysis	79
4.4.1	Game Phase Analysis	80
4.4.2	Defensive Systems Analysis	86
4.5	Game Model	88
4.5.1	Game ON Subnet	92
4.5.2	Game Stop Subnet	96
4.6	Summary and Conclusions	99
5	Experimental Validation	101
5.1	Player Detection and Tracking	101
5.1.1	Processing Time	102
5.1.2	Measurement Accuracy	102
5.1.3	Detection	102
5.1.4	Sensitivity Analysis	106
5.1.5	Detection with Shared Colours	113
5.1.6	Player Tracking	116
5.2	Ball Detection and Tracking	118
5.3	Game Analysis	120
5.3.1	Game Phase Detection	120
5.3.2	Team Formation Detection	123
5.4	Results Visualization and Annotation	124
5.4.1	Metrics	125
5.4.2	Maps	129
5.5	Software Interface	133
5.5.1	Acquisition Module	133
5.5.2	Processing Module	134
5.5.3	Visualizer Module	141
5.6	Game Model Validation	145
5.6.1	First Hierarchical Level	145
5.6.2	Second Hierarchical Level	149
5.6.3	Third Hierarchical Level	156
5.7	Specific Case Studies	162
5.7.1	Evaluate players' effort during official competition	162
5.7.2	Assessing referees physical effort	163
5.7.3	Assessing Physical Activity Intensity	165
5.8	Summary and Conclusions	166

6	Conclusions	169
6.1	Research Findings	169
6.2	Implementation Results	171
6.3	Publications	171
6.4	Guidelines for Future Work	171
6.5	Final Remarks	172
A	Game Model - Appendix	173
	References	179

List of Figures

2.1	Overview of image segmentation techniques (adapted from Monteiro (2008) and Vantaram and Saber (2012)).	6
2.2	Block diagram of CAMSHIFT (adapted from Bradski (1998)).	12
2.3	Examples of graphs: (a) simple graph (b) weighted graph (c) directed weighted graph.	13
2.4	Complete cycle of the Kalman Filter (adapted from Welch and Bishop (2002)).	15
2.5	Complete cycle of the Extended Kalman Filter (adapted from Welch and Bishop (2002)).	15
2.6	Complete life cycle of the Unscented Kalman Filter.	16
2.7	Pseudo-code for SIS particle filter in Arulampalam et al. (2002) .	18
2.8	Pseudo-code for resampling algorithm in Arulampalam et al. (2002) .	18
2.9	Overview of types of classifiers.	19
2.10	A neural network node (adapted from Russell and Norvig (1995)).	22
2.11	Support vectors.	23
2.12	Fuzzification step (adapted from Negnevitsky (2001)).	25
2.13	Fuzzy inference step (adapted from Negnevitsky (2001)).	26
2.14	Defuzzification step (adapted from Negnevitsky (2001)).	26
2.15	Example of two PN and their firing sequence.	31
2.16	Example of the main screen of CPN Tools.	33
2.17	Instances provided by CPN Tool software.	34
2.18	Inscriptions available for places on CPN Tool software: place name, colour set and initial marking.	34
2.19	Inscriptions available for transitions on CPN Tool software: transition name, guard, time, code segment and priority.	34
2.20	CPN Tool arc inscriptions: (a) single element Boolean arc inscription (b) multiset arc inscription with two elements (c) multiset arc inscription with seven elements.	35
2.21	CPN Tool transition firing: (a) Petri net state prior to the transition takes place and (b) Petri net state after to the transition takes place.	36
2.22	CPN Tool substitution transition: (a) substitution transition superpage (b) substitution transition subpage.	37
3.1	Apidis typical images in Apidis (2009) .	49
3.2	Cairos Technologies and Adidas goal detection system (Cairos Technologies AG).	54
3.3	Camera distribution in D'Orazio et al. (2009) .	57
3.4	Camera distribution in Ren et al. (2009) .	58
3.5	Aspogamo model hierarchy from Beetz et al. (2009) .	63
3.6	Passing accuracy of the game Italy vs. Spain (Duch et al. (2010)).	64
3.7	Time evolution of the performance of players and teams (Duch et al. (2010)).	64

4.1	System's architecture (from Santiago et al. (2013))).	67
4.2	Data flow information.	68
4.3	Images collected from the two camera's system.	69
4.4	(a) Player effort in tabular form (b) positional map indicating the zones occupied by the player (c) tactical map of both teams during a time frame defined by the user.	71
4.5	(a) and (b) left image before and after removing the barrel effect distortion. (c) and (d) right image before and after removing the barrel effect.	78
4.6	Fuzzy inference engine for game phase classification.	81
4.7	Membership functions for $v_x = \{ HighNegVx, LowNegVx, StopVx, LowPosVx, HighPosVx \}$	81
4.8	Membership functions for $p_x = \{ Close2OwnGoal, MiddleFieldNeg, MiddleFieldPos, Close2OppGoal \}$	82
4.9	Membership functions for $prev_{phase} = \{ Defence, DefOff, OffDef, Attack \}$	82
4.10	Membership belonging degrees when: (a) $v_x = -1.8m/s$ (b) $p_x = 7.8m$ and (c) $phase = Attack$	83
4.11	Most common defensive systems, from top right to bottom left 6:0, 5:1, 4:2, 3:3, 3:2:1 and 1:5.	87
4.12	Example of how the minimum distance between the goal and the players is determined.	88
4.13	Model hierarchy overview.	89
4.14	First layer of the game model.	90
4.15	Subnet with the model of the Game On subnet.	91
4.16	Model of the Defence subnet.	93
4.17	Model of the Attack subnet.	95
4.18	Model of the Game Stop subnet.	97
5.1	Images from a 3 cameras' system.	101
5.2	Measurement error. Dots represent real coordinates and bars represent the measure obtained using the estimated homography.	102
5.3	(a) Initial colour subspaces for green and red teams. Lighter dots are seed colours (C_S), intermediate are team colour (C_F) and the darkest resemble the team colour (C_L). (b) Examples of players from red (up) and green teams (down).	103
5.4	Final colour subspaces after 34 auto-expansions.	103
5.5	Evolution of the colour triplets that belong to each colour subspace and the respective belonging degree. The expansion is performed at every 30 frames ($t_{exp} = 30$).	104
5.6	Number of miss detected field players in each frame with and without colour auto-expansion: (a) red and (b) green team. Each point was obtained by passing an averaging filter of size 9 to the original points.	105
5.7	Results of the player detection at frame 671: (a) without auto-expansion and (b) with auto-expansion. Green and red crosses indicate correct detection while blue crosses indicate the position was predicted by the Kalman filter.	106
5.8	Comparison of miss detection rates using different t_{exp} : 15, 30, 45 and 60 frames (each point on the graphs was obtained by passing an averaging filter of size 9 to the original points).	108

5.9	Comparison of miss detection rates (bottom) between the original video and video with artificially changed brightness. Original video and corresponding detection deficiencies are shown in the "Original" series. Video with long term added brightness step is shown on "Step" series. Video with added ramp brightness is shown on "Ramp" series. Video with added noise of Gaussian distribution is shown on "Gauss" series. Top graph indicates the average brightness of the video (results were collected with $t_{exp} = 30$ and each point on the graphs was obtained by passing an averaging filter of size 9 to the original points).	109
5.10	Example of images obtained by applying a brightness step of -20 (top), 0 (middle) and +20 (bottom) to the original videos.	110
5.11	Comparison of miss detection rates between two different initial sets (Experiment A and Experiment B). All results were collected with $t_{exp} = 30$ and each point on the graphs was obtained by passing an averaging filter of size 9 to the original points.	112
5.12	Comparison of miss detection rates using original image and down sampled image with a scale factor of 1/2: (a) examples of the two images (b) and plot showing the detection rates along the time. All results were collected with $t_{exp} = 30$ and each point on the graphs was obtained by passing an averaging filter of size 9 to the original points.	114
5.13	Comparison of tracking rates using different temporal windows (TPW = 1, 5, 10 and 20 frames) for the predictive stage of the Kalman Filter (results were collected with $t_{exp} = 30$ and each point on the graphs was obtained by passing an averaging filter of size 9 to the original points).	115
5.14	(a) Colour subspaces for two teams with a common colour (white) and (b) an example of a processed image.	116
5.15	Comparison of miss detection rates with different setups: without auto-expansion and without Kalman filter (noExp noKalman); with auto-expansion and without Kalman filter (Exp noKalman); without auto-expansion and with Kalman filter (noExp Kalman); with auto-expansion and with Kalman filter (Exp Kalman). The Kalman Filters used a $TPW = 5$ frames and each point on the graphs was obtained by passing an averaging filter of size 9 to the original points.	117
5.16	Processed video showing the tracking of the ball.	118
5.17	Images provided by the visualizing application when tracking the ball during a pass between two players.	119
5.18	Images provided by the visualizing application when tracking the ball during a dribble.	119
5.19	Evolution of the belonging degrees and output of the Fuzzy classifier from frame 1435 to frame 1786.	122
5.20	Results comparison between expert and automatic defensive team formation labelling for team A.	123
5.21	Results comparison between expert and automatic defensive team formation labelling for team B.	123
5.22	Images provided by the visualizing application: (a) image of Game 1, (b) image of Game 2.	124
5.23	Positional maps of the 10 minutes for team A players during Game 1: (a) player P_0 , (b) player P_1 , (c) player P_2 , (d) player P_3	129
5.24	Positional maps of the 10 minutes for team B players during Game 1: (a) player P_6 , (b) player P_7 , (c) player P_8 , (d) player P_{11}	130

5.25	Positional maps for the 11 minutes of players from team A during Game 2:(a) player P_0 , (b) player P_1 , (c) player P_4 , (d) player P_5	131
5.26	Positional maps for the 11 minutes of players from team B during Game 1:(a) player P_9 , (b) player P_{10}	131
5.27	Velocity maps for P_1 (team A) during Game 1 (10 minutes of game):(a) speed below 2m/s, (b) speed between 2m/s and 4m/s and (c) speed above 4m/s.	132
5.28	Speed maps for P_4 (team A) during Game 2 (11 minutes of game):(a) speed below 2m/s, (b) speed between 2m/s and 4m/s and (c) speed above 4m/s.	132
5.29	Positional maps for two game situations of Game 1: (a) team A (in red) attacking and team B (in green) defending (from frame to 620 to frame 1478), (b) team A defending and team B attacking (from frame 1712 to frame 2589).	132
5.30	Positional maps for two game situations of Game 2: (a) team A attacking and team C defending (from frame to 600 to frame 1160), (b) team C defending and team A attacking (from frame 1650 to frame 2120).	133
5.31	Acquisition System software print screen.	133
5.32	Processing Software Module main window with the videos browse menu highlighted.	134
5.33	Processing Software Module showing the videos of a two cameras' system. . . .	135
5.34	Processing Software Module main window with the synchronization and advance configuration menus highlighted.	135
5.35	Example of how the videos are shown when the user: (a) checks the <i>show proc</i> checkbox and (b) when it does not.	136
5.36	Processing Software Module main window with the colour configuration, player identification, input and output configuration files.	136
5.37	Example of a configuration file used on the Processing Module.	137
5.38	Example of an output log file.	138
5.39	Advanced tab of the Processing Software Module.	139
5.40	Example of points selection of two lines used to determine the barrel distortion coefficients $(k_1, k_2, k_3, x_c, y_c)$	139
5.41	Windows provided by the application for the user to introduce the real world co-ordinates when performing the Homography calibration.	140
5.42	Debug tab of the Processing Software Module.	141
5.43	Visualizer Software Module showing the undistorted final image with the players highlighted, along with the Visualizer and Analyser windows.	142
5.44	Visualizer Software Window with the different areas highlighted.	142
5.45	Example of the log file generated by the Visualizer Module.	143
5.46	Global video window showing: (a) original image, (b) positional map with players highlighted.	144
5.47	Analyser Software Window.	145
5.48	Temporal chart of the Game Mode Petri Net states and transitions during the analysed game period.	146
5.49	(a) Petri Net state for the Game Mode level (the green highlight around the Game Stop substitution transition indicates the Petri Net state) and (b) players' distribution on the field at frame 619 (20.6s).	146
5.50	Petri Net state for Game Mode when the game starts at frame 620.	147
5.51	Players' distribution on the field at frame 2620 (87.3s).	147

5.52	Petri Net state for Game Mode when the game stops at frame 2620 (87.3s): (a) while the net is still evaluating the cause of the exit, (b) while the game is already stopped.	148
5.53	Petri Net state for Game Mode while the team is preparing for the Free Throw when the game resumes back to the Game On substitution transition at frame 3001 (100s).	149
5.54	Players' distribution on the field at frame 3001 (100s).	149
5.55	Temporal chart of the Game ON Petri Net states and transitions during the analysed game time period.	150
5.56	Temporal chart of the Game Stop Petri Net states and transitions during the analysed game time period.	150
5.57	Game Stop Petri net at: (a) frame 618 (20.6s) and (b) frame 619 (20.6s) (partial view of the nets).	151
5.58	Game ON Petri net at frame 619 (20.6s).	151
5.59	Game ON Petri net at frame 620 (20.6s).	152
5.60	Game ON Petri net at frame 620 (20.6s) after team reaching the Defence state.	152
5.61	Game ON Petri net at frame 752 (25s), when <i>Team</i> is attacking and <i>TOpp</i> is defending.	153
5.62	Game ON Petri net at frame 1496 (49.8s) after the defending team intercepting the ball.	153
5.63	Game ON Petri net at frame 2620 (87.3s), after a defensive foul.	154
5.64	Petri Net state for (Game Stop) when the defending team performs a foul (partial views of the net).	154
5.65	Game ON Petri net at frame 2844 (94.8s), when teams are preparing to perform a Free Throw	155
5.66	Game Stop Petri net at frame 2844 (94.8s), when teams are preparing to perform a Free Throw.	155
5.67	Players' distribution on the field at frame 3135 (104.5s), when the defending team commits a foul.	156
5.68	Temporal chart of the Attack Petri Net states and transitions during the first attack, from frame 620 (20.6s) to frame 1496 (49.8s).	157
5.69	Temporal chart of the Defence Petri Net states and transitions during the first defence from frame 620 (20.6s) to frame 1496 (49.8s).	157
5.70	Attack subnet sequence of transitions when the game starts at frame 620 (20.6s) (partial views of the net): (a) before reaching the Attack state, (b) and (c) when reaching an attacking position, (d) decision on whose player has the ball.	158
5.71	Petri Net sequence of a ball pass between players 5 and 0 during frames 779 and 789 (partial views of the net). (a) ball pass start (b) ball pass end.	159
5.72	Global view of the field during a ball pass. (a) Player 5 passes the ball at frame 779 (b) and player 0 receives the ball at frame 789.	159
5.73	Partial view of the field when player 1 is dribbling the ball during frames 1251 to 1312(a) at frame 1262, (b) at frame 1267, (c) at frame 1271 and (d) at frame 1275.	160
5.74	(Defence) subnet sequence of states and transitions when the game starts at frame 620 (20.6s): (a) before game starts, (b) when team is entering the Defensive state and (c) when all players start zone defending.	160
5.75	Defence Petri net at frame 754 (25.11s) (partial view of the net).	161
5.76	Global view of the field at frame 754 (25.11s), when players 8 and 10 are performing a man-to-man defence.	161

5.77	(a) Defence Petri net at frame 932 (partial view of the net) and (b) the partial view of the field.	162
5.78	Defence Petri Net sequence of transitions at the end of the first attack (partial views of the net): (a) the attacking player performs the ball throw, (b) the defending goalkeeper intercepts the ball ($dInterceptBall = true$), (c) players start moving towards the opponent goal (Def Off state).	163
5.79	Attack Petri Net sequence of transitions at the end of the first attack (partial views of the net): (a) before the throw to the goal, (b) while the ball is being thrown, (c) when the ball is intercepted by the defending team goalkeeper.	164
5.80	Positional map of each referee during one half of the game (in Estriga et al. (2013)).	165
A.1	Temporal chart of the Attack Petri Net states and transitions during the second attack: from frame 1759 (58.6s) to frame 2580 (86s).	174
A.2	Temporal chart of the Defence Petri Net states and transitions during the second attack: from frame 1759 (58.6s) to frame 2580 (86s).	175
A.3	Temporal chart of the Attack Petri Net states and transitions during the third attack: from frame 3001 (100s) to frame 3136 (104.5s).	176
A.4	Temporal chart of the Defence Petri Net states and transitions during the third attack: from frame 3001 (100s) to frame 3136 (104.5s).	176
A.5	Temporal chart of the Attack Petri Net states and transitions during the fourth attack: from frame 3436 (114.5s) to frame 3639 (121.3s).	177
A.6	Temporal chart of the Defence Petri Net states and transitions during the fourth attack: from frame 3436 (114.5s) to frame 3639 (121.3s).	177

List of Tables

3.1	Summary of systems based on intrusive devices.	43
3.2	Summary of NIS features ([D] – detection, [T] – tracking, P/R – precision/recall, [M] – measurement accuracy, * - values calculated based on the information provided by the authors).	45
3.3	Summary of methodologies used on non-intrusive systems.	52
3.4	Summary of Ball Detection and Tracking Systems.	59
3.5	Taxonomy and summary regarding player detection techniques.	65
4.1	Mapping of B_{Sc} to Fuzzy belonging μ	73
4.2	Fuzzy inference associative matrix.	75
4.3	Belonging degrees to each membership term.	83
4.4	Game phase Fuzzy inference rule matrix when the Previous Phase is Defence. . .	84
4.5	Game phase Fuzzy inference rule matrix when the Previous Phase was <i>Attack</i> . . .	84
4.6	Game phase Fuzzy inference rule matrix when the Previous Phase was <i>DefOff</i> . . .	85
4.7	Game phase Fuzzy inference rule matrix when the Previous Phase is <i>OffDef</i>	85
4.8	Defined colour sets.	89
5.1	Foreground detection using different learning constants ($\alpha = 0.02$, $\alpha = 0.08$, $\alpha = 0.16$ and $\alpha = 0.20$) at frames 31, 82 and 120. The pixels classified as background were darkened.	111
5.2	Colour subspaces evolution depending on the initial colour seeds. Lighter dots are seed colours (C_S), intermediate are team colour (C_F) and the darkest resemble the team colour (C_L).	113
5.3	Tracking rates of all the players (6 field players per team) during 11000 frames analysed ($\approx 370s$).	118
5.4	Comparison of phase detection between a sport's expert categorization and the developed automatic methodology for a 10 minute's period of Game 1.	120
5.4	Comparison of phase detection between a sport's expert categorization and the developed automatic methodology for a 10 minute's period of Game 1.	121
5.5	Player statistics of Game 1 during approximately 10 minutes of game.	125
5.6	Player statistics of Game 2 during approximately 11 minutes of game.	127
5.7	Results of the Yates Chi-square between methodologies (in Silva et al. (2015)). .	166

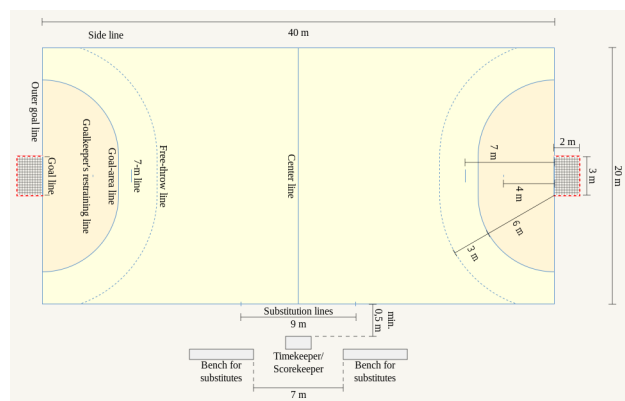
Abbreviations and Symbols

3D	3 dimensions
AAAI	Association for the Advancement of Artificial Intelligence
AC Milan	Associazione Calcio Milan (Football Association of Milan)
AOA	Angle of Arrival
APIDIS	Autonomous Production of Images based on Distributed and Intelligent Sensing
ASIR	Auxiliary Sampling Importance Resampling
CAMSHIFT	Continuously Adaptive Mean Shift
CHT	Circle Hough Transform
CPU	Central Processing Unit
DAML	DARPA Agent Markup Language
DARPA	Defense Advanced Research Projects Agency
ECTS	European Credit Transfer and Accumulation System
EKF	Extended Kalman Filter
EM	Expectation Maximization
FADEUP	Faculty of Sports of the University of Porto
FC	Fußball-Club (Football Club)
FIFA	Fédération Internationale de Football Association (International Federation of Association Football)
FLogic	Frame Logic
fps	frames per second
GMM	Gaussian Mixture Model
GPS	Global Positioning System
HOT	Higher Order Terms
HT	Hough Transform
HTML	HyperText Markup Language
ID3	Iterative Dichotomiser 3
IEEE	Institute of Electrical and Electronics Engineers
KIF	Knowledge Interchange Format
LDA	Linear Discriminant Analysis
LISP	List Processing Language
LPM	Local Position Measurement

MC	Monte Carlo
MOG	Mixture of Gaussians
MPEG	Moving Picture Experts Group
MUMIS	Multimedia Indexing and Searching Environment
OCML	Operational Conceptual Language
OIL	Ontology Interchange Language
OKBC	Open Knowledge Base Connectivity
OWL	Web Ontology Language
PCA	Principal Component Analysis
PSV	Philips Sport Vereniging (Philips Sports Union)
RAM	Random Access Memory
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
RFID	Radio Frequency Identifiers
RGB	Red, Green, Blue
ROI	Region of Interest
RPF	Regularized Particle Filter
SGML	Standard General Markup Language
SHOE	Simple HTML Ontology Extension
SIR	Sampling Importance Resampling
SIS	Sequential Importance Sampling
SVM	Support Vector Machine
SWAN	Semantic Web Annotator
TDOA	Time Difference of Arrival
TV	Television
UEFA	Union of European Football Associations
UKF	Unscented Kalman Filter
UT	Unscented Transform
UWB	Ultra Wide Band
W3C	World Wide Web Consortium
XML	eXtensible Markup Language
XOL	XML-Based Ontology Exchange Language
YIQ	luma, in-phase, quadrature

Handball Terminology

Handball Court



Handball field (in [Wikipedia \(2015\)](#))

Throw-In	A throw-in is given when the ball crosses the side line or when crosses the outer goal line after being touched by a defending player. In case the ball goes outside by the side line the throw-in is performed on the spot where the ball left the court, otherwise the throw-in is taken at the intersection of the side line with the outer goal line.
Throw-Off	A throw-off is awarded at the beginning of each half and also after scoring a goal. This throw is performed at the centre of the court.
Free-Throw	A free-throw is awarded when the team in possession of the ball commits a foul or when the opponent team commits a foul that causes the team in possession of the ball to lose it. A free-throw is taken from the place where the infraction took place.
7-Meter Throw	A 7-meter throw is awarded when a clear chance of scoring is destroyed by the defending team with a foul. The execution of a 7-meter throw is taken as a shot on goal from behind the 7-meter line.
Goalkeeper-Throw	A goalkeeper-throw is awarded when: a player enters the goal area and wins an advantage; when the goalkeeper controls the ball inside the goal area or when the ball goes outside the field by the outer goal line after being touched either by the goalkeeper or by an attacker. The goalkeeper-throw is performed by the goalkeeper from the goal area to the playing court.
Team time-out	Each team is entitled to 1-minute time-out in each half of the game. The team may only request the time-out if in possession of the ball.

Chapter 1

Introduction

Sports have always played an important role in our society, first as a mean to prepare men for hunting or fighting, then as a leisure activity or even as part of a competition.

Due to its importance in our society, sports have embraced technology in order to provide a more immersive experience to fans by allowing different views, zooms of specific situations and even the recording for posterior visualization.

On the other hand, there is also interest by the sports community on bringing the benefits of scientific and technological development into the sports domain in order to provide mechanisms for performing a more accurate and systematic analysis of the game, as well as to create new mechanisms to aid training and decision making.

At the amateur level, users have now at their disposal several gadgets that allow localization and effort evaluation (which includes speed, distance, motion information and time spent in practice).

Concerning high level competitions a small advantage of one team regarding another can be of great importance and decisive for a match or even a complete championship. Therefore, at this level there is great care in trying to benefit from all sources of information in order to gain advantage over an opponent.

On a more academic level, sport teachers are also showing interest on having mechanisms to aid during teaching and keep track of the evolution of the taught/trained issues.

An automatic tool able to perform a high level analysis of sport games could provide useful material to all of the aforementioned groups. Such tool should be as non-intrusive as possible so that it would not interfere in the game environment, allowing players to move as freely as possible without any extra weight.

Pursuing this aim, the Faculty of Engineering of the University of Porto (FEUP) in collaboration with the Faculty of Sports of the University of Porto (FADEUP) have developed a project mostly focused on handball to address problems related with player tracking and game analysis.

1.1 Motivation and Goals

As already referred, sports are taking considerable importance in our society, not only due to the increasing interest by fans, but also due to the amount of economic transactions that are at stake.

These two factors have contributed for a quest from most professional sport teams to rely on high level game analysis. The focus of this analysis can either be on the players as an individual person or as part of a team.

Nowadays, this analysis is mainly done through visual observation from experts since video is the least intrusive technology. However the existing software usually requires considerable user intervention which delays the process of obtaining results on useful time.

An indoor sport is played in a sports hall that has very specific characteristics. Usually the same sports hall can be used for several sports such as handball, basketball or volleyball. Further complexity on the handball and basketball cases arises from the fact that they are invasion sports, where teams are not segregated on the two halves of the field, but actively compete with each other throughout the field.

Team sports have in common the fact that players wear similar equipment, move very rapidly, are constantly changing their trajectory, interact with each other which often causes occlusions or merging making it even more difficult to track the players.

Of course, each of these sports has its own specificities, for example handball is a very fast game, with the main action zone near the 6 meter line. It is a high contact game where occlusion between players of the same team and different teams often occurs. Statistics indicate that an handball player can achieve velocities of around 7m/s and the ball 28m/s.

Taking into account this need from the sports community, and given the complexity adjacent to the game itself, the focus of this thesis was to study and explore algorithms and techniques to develop a non-invasive automatic image processing system for team sports with application on indoor sports that is able to detect and track the players and perform game and tactical analysis.

1.2 Thesis Statement and Objectives

Concerning the motivation and goals behind this thesis it is defended that:

"It is possible to perform high level game analysis for indoor sports combining the information obtained from a vision system with game models."

From this statement some questions arise, namely:

- Q1** What architecture should be used to achieve this goal?
- Q2** What information should be extracted from the vision system?
- Q3** How to extract that information?
- Q4** How to model game concepts?

Q5 How to extract metrics from the models?

In order to answer these questions the following objectives were outlined:

- propose a non-invasive image acquisition system adequate for an indoor sport's hall
- devise methodologies to perform player detection and tracking given the proposed non-invasive acquisition system
- develop algorithms for detecting the game concepts and extracting metrics associated with them
- define a model for the handball game, able of modelling the game flow, but also the players interactions
- validate the proposed methodologies in real game situations, by collecting data at official competitions

1.3 Scientific Contributions

During the course of this thesis the following scientific contributions were performed:

- Design and definition of a modular architecture able to collect video footages from different setups (number of cameras), process the video information and provide useful metrics to the final user.
- Development of an adaptable colour calibration method based on a region growing method combined with Fuzzy categorization, which is able to absorb the positional and temporal colour variations of the teams' equipment.
- Development of an indoor player tracking system, which was used to track not only the handball players, but also to assess the referees effort during official competitions and evaluate the classification of traditional observation methods.
- Study and development of methodologies to classify the handball game phase, as well as the defensive team formation.
- Conceptualization and development of a handball game model based on a Hierarchical Coloured Petri Net. The hierarchical nature of the module allows it to be modular and easily adaptable to other indoor sports, such as basketball.

1.4 Document Structure

This document is organized into six chapters. Each chapter comprises a small introduction describing its contents followed by the content itself.

Chapter 1, Introduction, is subdivided into 4 sections that introduce the subject of this thesis, including the motivation and the goals inherent to it. It also provides the context of the problem and the thesis statement.

Chapter 2, Methodologies and Fundamentals, provides a theoretical background regarding methodologies and techniques commonly used in image processing systems as well as image recognition and knowledge representation. The aim of this chapter is to provide a solid theoretical background to better understand the subsequent chapters. Emphasis is given to image segmentation, object recognition and tracking as well as modelling languages and formalisms. A brief overview on CPN Tools, a tool to handle Coloured Petri Nets, is also provided.

Chapter 3, Related Research, presents a literature review on systems that have been devised to tackle the player detection and tracking problem, as well as the issue of high level game analysis. This chapter also highlights the main open issues.

Chapter 4, Methodology, explains the proposed solution, including the algorithms developed to perform the player detection and tracking. This chapter also presents a game model based on Hierarchical Coloured Petri Nets and the approach used to perform high level game analysis.

Chapter 5, Experimental Validation, describes the experiments taken in order to validate the proposed methodologies.

Chapter 6, Conclusions, summarizes the main conclusions and contributions of this thesis. It also launches possible directions for future developments.

Chapter 2

Methodologies and Fundamentals

This chapter intends to present some of the methodologies and concepts that are usually adopted in this kind of systems, which come from different areas such as image processing, data mining, machine learning and artificial intelligence.

2.1 Video Segmentation

Video segmentation is the first step, and probably the most critical, in any video (image) processing system, because the quality of the final result is highly dependent on a good segmentation. There are two main categories of video segmentation:

- Temporal segmentation - Segments the video into meaningful temporal sequences. It is usually used as the first step of video annotation and segments the video taking into account similarities/dissimilarities between successive frames ([Koprinska and Carrato \(2001\)](#)).
- Spatial segmentation - Aims to divide the content of each frame into homogeneous regions that correspond to independent objects.

Due to the nature of this thesis, we are more concerned with spatial segmentation, therefore, for more details on temporal segmentation please refer to [Koprinska and Carrato \(2001\)](#).

2.1.1 Spatial Video Segmentation

Spatial video segmentation inherits many of the methodologies used for image segmentation, with the added benefit of allowing the incorporation of the temporal characteristics inherent to video.

Spatial segmentation methods are based on two basic properties of the image, discontinuity and similarity ([Gonzalez and Woods \(2002\)](#)). Discontinuity methods are characterized by partition the image based on the detection of intensity changes such as edges, while similarity methods, segment the image into regions that are similar in some predefined characteristics.

According to [Monteiro \(2008\)](#) and [Vantaram and Saber \(2012\)](#), colour image segmentation techniques can be divided into two main categories, feature domain and image domain. These categories can be further refined into Clustering, Histogram Thresholding, Region Based, Boundary Based and Motion Based methodologies (Figure 2.1).

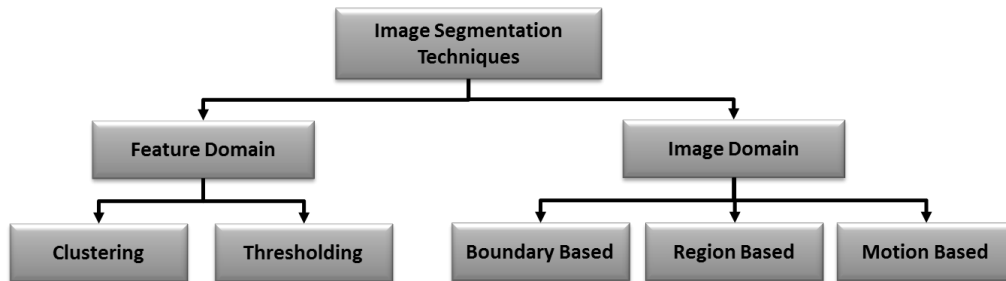


Figure 2.1: Overview of image segmentation techniques (adapted from [Monteiro \(2008\)](#) and [Vantaram and Saber \(2012\)](#)).

Feature domain methods, also named spatially blind based methods ([Vantaram and Saber \(2012\)](#)) do not take into account the spatial relation among pixels, instead they segment regions based on a feature/attribute space. On the other hand, image based (or spatially guided) methods perform segmentation by analysing the spatial relation among pixels.

Initially, authors used only one method to segment the images, but nowadays, the tendency is to aggregate techniques from different categories in order to achieve better results. A typical example of this is the JSEG algorithm ([Deng and Manjunath \(2001\)](#)) that initially clusters colours into several representative classes, afterwards replaces each pixel with its corresponding colour class label and only then applies a region growing process directly to the class map.

2.1.1.1 Clustering

Clustering in image processing is a classification methodology for grouping the image feature space into a set of meaningful groups or classes called clusters, where data belonging to the same cluster possess a higher degree of similarity, when compared to data from other clusters. Usually the feature space is based on intensity, colour or texture and ignores the spatial relation between pixels on the image.

As defined in [Kotsiantis and Pintelas \(2004\)](#), there are five main approaches for clustering:

- hierarchical clustering (also known as graph based segmentation methods) represented by a hierarchical decomposition of the set of data (or objects) using a specific criteria. These methods, depending on how the hierarchical decomposition is performed, either bottom-up or top-down, can be further classified into agglomerative or divisive, respectively;
- partitioning clustering methods divide the data into k clusters based on some evaluation criteria;

- density-based partitioning methods allow clusters to grow into arbitrary shapes by using density and connectivity functions;
- grid-based methods perform the clustering on a grid that is obtained by first quantizing the clustering space into a number of finite cells;
- model-based methods try to optimize the fit between a model and the data.

Hierarchical clustering groups data objects into an hierarchy, and can be divided into (Forsyth and Ponce (2003)): divisive methods, where the entire data set is regarded as a cluster and then clusters are recursively split to yield a good clustering; and agglomerative methods where each data item is regarded as a cluster and clusters are recursively merged to yield a good clustering.

Agglomerative methods use an inter-cluster distance to merge neighbour clusters, while divisive clusters use this distance to split clusters. A good way to represent hierarchical clustering is using a tree structure called dendrogram.

Partitioning clustering includes the well-known K-means (Hard Clustering) (Hartigan (1975)), K-Medoids (Kaufman and Rousseeuw (1987)) and Fuzzy C-means (Fuzzy C-Clustering) (Bezdek (1981)).

K-means is one of the simplest unsupervised clustering methods and starts with a random k initial partition of the feature space (with j points). Each partition (i) is characterized by its centroid (mean c_i) and the objective is to minimize the within-cluster sum of squares of the data points x_j (equation 2.1), by exchanging, in each iteration, points to the cluster with the nearest centroid.

$$f(i, j) = \sum_{i=1}^k \sum_{j=1}^{data} (x_j - c_i)^T (x_j - c_i) \quad (2.1)$$

Fuzzy C-clustering is quite similar to K-means but allows pixels to belong to multiple classes with different degrees of membership.

Density-based clustering algorithms are able to find clusters with arbitrarily shapes, handle noisy data and only require one scan in order to find a solution. Examples of this class of algorithms are DBSCAN (Ester et al. (1996)) (Density-Based Spatial Clustering of Applications with Noise) or BRIDGE (which combines DBSCAN with K-means) (Dash et al. (2001)).

Grid-based clustering is especially suitable for very large datasets, due to the considerable data reduction achievement obtained from quantization. Grid-based clustering WaveCluster algorithm was used to perform mammographic segmentation (Barnathan (2012)).

Model-based clustering allows including knowledge about the domain and contrarily to K-means offers more flexibility about the clusters models. A commonly used algorithm for model-based clustering is the Expectation-Maximization algorithm (EM). This algorithm was first proposed by Dempster et al. (1977) and consists in an iterative maximum-likelihood procedure for parameter estimation from missing or incomplete data.

2.1.1.2 Thresholding

As defined in (Sezgin and Sankur (2004)), thresholding techniques can be further subdivided into six categories considering the information used to perform the segmentation: histogram shape, clustering, entropy, object attribute similarity, spatial and local methods.

Histogram shape based methods exploit the histogram shape either by searching histogram peaks and valleys, analysing the histogram concavities via the convex hull or approximating the histogram shape by functions.

Histogram clustering based methods separate the grey-level samples into two classes (background and foreground). Approaches include:

- iteratively searching for the threshold value until finding an optimal point where the new value is sufficiently near the previous one (Iterative Thresholding);
- assuming that an image can be characterized by a mixture distribution of foreground and background pixels and solving a minimum error Gaussian density fitting problem or minimizing the total misclassification error (Minimum Error Thresholding);
- using the cross-over point of fuzzy membership functions (Fuzzy Clustering Thresholding);
- or the most used histogram thresholding method, the Otsu method (Otsu (1979)), which computes the optimal threshold by determining the intensity value that minimizes the weighted within class variances of the two classes (Clustering Thresholding).

Entropy based thresholding methods determine the threshold value based on the grey level distribution entropy, either by maximizing the entropy of the thresholded image, minimizing the cross-entropy between the input grey-level image and the output binary image or using fuzzy memberships to characterize how strongly a pixel belongs to the background or foreground and minimizing the sum of fuzzy entropies.

Object attribute similarity methods determine the threshold value by measuring the similarity of a specific characteristic (such as edges, grey-level moments or texture) between the original image and the binarized image.

The main drawback of histogram thresholding is that not always histograms have obvious peaks and the segmented regions may not be contiguous if spatial information is not taken into consideration (Monteiro (2008)). Therefore, spatial thresholding methods (Sezgin and Sankur (2004)) incorporate information about the neighbouring pixels such as local average grey levels, quadtree thresholding or co-occurrence probabilities, among others.

Since images may not have uniform luminance (for example due to shadows or non-uniform illumination), global threshold techniques like the ones previously referred may not be suitable. Therefore, locally adaptive threshold techniques have emerged which try to surpass this limitation by computing individual thresholds for each pixel using information from the neighbourhood of the pixel.

Histogram thresholding can be quite straightforward in grey scale images as long as the intensity of pixels belonging to the background and foreground are somewhat different, however on colour images this procedure is more complex and computationally costly because of the multi-dimensionality of the colour space.

In order to overcome these problems, authors have developed methods for storing and dealing with information on the 3D colour space, namely by using binary trees, where each node contains the number of pixels with a given RGB (red, green, blue) range of values or by projecting the 3D space onto a lower dimensional space (Cheng et al. (2001)).

2.1.1.3 Region Based

Region-Based techniques intend to split the image domain by progressively fitting statistical models to the image properties (colour, intensity, texture or motion) and are based on the fact that adjacent pixels in a same region have similar visual features. There are four types of region-based algorithms: region growing, region splitting, region merging and hybrid algorithms.

Region growing algorithms start with predefined seed pixels and gradually agglomerate pixels around these seeds that satisfy a given criteria of intensity, colour or texture. This growth process stops when no more pixels can be agglomerated into the regions. One advantage of these methods is that the regions obtained are spatially connected and compact, however they are quite dependent on a good seed choice (Monteiro (2008)).

On the other hand, region splitting methods start with a non-homogeneous image segmentation result and continuously split the regions until the desired homogeneity criteria is obtained. Quadtree and Watershed transform are examples of splitting methods (Beucher and Lantuejoul (1979)).

In Watershed transform, an image is seen as a topographic image immersed in a lake, with holes pierced in local minima. The method starts by filling up basins with water starting at these local minima. In the process different basins can merge with each other forming dams. The stopping condition is achieved when the water level reaches the highest peak in the image. As a result, the image is segmented into regions or basins separated by dams that correspond to the watershed lines.

Region merging methods objective is to merge regions that satisfy a given homogeneity criteria.

A very well-known algorithm that is considered hybrid (both growing and merging) is JSEG (Deng and Manjunath (2001)). In JSEG colour segmentation and spatial segmentation are decoupled, so initially colours are quantized into several representative classes and pixels are replaced by their corresponding colour class labels and only then a region growing process is applied directly on this class map. Afterwards, the obtained regions are merged using colour similarity criteria.

2.1.1.4 Boundary Based

As the name indicates, boundary based methods segment the image based on edges by locating pixels where the intensity (colour) changes when compared to its neighbours. Edge-based methods segment the image by finding the edges of each region using one of the well-known edge detectors (Canny [Canny \(1986\)](#), Sobel [Sobel \(1970\)](#), Roberts [Roberts \(1963\)](#)). Afterwards it is usually necessary to perform some edge linking in order to complete the boundaries.

Deformable models are also boundary based methods and can be classified ([Monteiro \(2008\)](#)) based on the implementation into:

- parametric, which represent curves and surfaces explicitly in their parametric form during the deformation such as snakes;
- geometric, that represent curves and surfaces implicitly as a level set of a higher dimensional scalar function. Active contours or level sets are well known methods of geometric deformable models.

2.1.1.5 Motion Based

On videos, contrary to static images, besides the two physical components (x and y) there is also the time component. Using this property it is possible to segment images based on motion along time.

In order to perform this task there are two main approaches: background subtraction and optical flow.

Background subtraction can be used in cases where a more or less fixed background can be assumed. Several background subtraction techniques have been proposed in literature. The main issue with these methods is to obtain a good estimate of the background.

The simplest method to model the background is to use a single static image without objects. However this approach works rather poorly, because it does not take into account changes that may occur in the background (for example light effects). More robust methods include estimating the background model using a moving average ([Heikkila and Silvén \(1999\)](#)), median or even a mixture of Gaussians ([Grimson et al. \(1998\)](#)).

Optical flow ([Barron and Thacker \(2005\)](#)) is based on the fact that when an object moves in front of a camera, there is a corresponding change in the image. Therefore if a point p_0 in an object moves v_0 , then the image point p_i can be assigned a vector v_i to indicate its movement on the image plane.

In order to compute the optical flow of an image two assumptions are made: there is colour constancy, and pixels represented by $I(x, y, t)$ perform a small motion $(\delta x, \delta y)$ in δt time. From these two constraints equation 2.2 is obtained.

$$I(x, y, t) = I(x + \delta x, y + \delta y, t + \delta t) \quad (2.2)$$

Since the displacement $(\delta x, \delta y)$ is small enough it is possible to perform a 1st order Taylor expansion of equation 2.2 into 2.3.

$$I(x + \delta x, y + \delta y, t + \delta t) = I(x, y, t) + \frac{\partial I}{\partial x} \delta x + \frac{\partial I}{\partial y} \delta y + \frac{\partial I}{\partial t} \delta t + HOT \quad (2.3)$$

,where HOT represents the higher order terms which are small and can be ignored.

Further exploring equation 2.3 it is possible to obtain 2.4:

$$\begin{aligned} \frac{\partial I}{\partial x} \delta x + \frac{\partial I}{\partial y} \delta y + \frac{\partial I}{\partial t} \delta t &= 0 \Leftrightarrow \frac{\partial I}{\partial x} \frac{\delta x}{\delta t} + \frac{\partial I}{\partial y} \frac{\delta y}{\delta t} + \frac{\partial I}{\partial t} \frac{\delta t}{\delta t} = 0 \\ \Leftrightarrow \frac{\partial I}{\partial x} v_x + \frac{\partial I}{\partial y} v_y + \frac{\partial I}{\partial t} &= 0 \\ \text{assuming } I_x = \frac{\partial I}{\partial x}, I_y = \frac{\partial I}{\partial y} \text{ and } I_t = \frac{\partial I}{\partial t} \text{ then} & \\ (I_x, I_y) \cdot (v_x, v_y) = -I_t \Leftrightarrow \nabla I \cdot v = -I_t & \end{aligned} \quad (2.4)$$

This equation has two unknowns and consequently does not have a unique solution. This is the mathematical consequence of the aperture problem, which means that this first order system can only detect motion perpendicular to the orientation of the contour that is moving.

To add an additional constraint, Horn uses a global regularization calculation (Horn and Schunk (1981)). Their work assumes that images consist in objects undergoing rigid motion, and so, over relatively large areas the optical flow will be smooth. Then they minimize the square of optical flow gradient magnitude using 2.5.

$$\int_D (\nabla I \cdot \vec{v} + I_t)^2 + \lambda^2 \left[\left(\frac{\partial v_x}{\partial x} \right)^2 + \left(\frac{\partial v_x}{\partial y} \right)^2 + \left(\frac{\partial v_y}{\partial x} \right)^2 + \left(\frac{\partial v_y}{\partial y} \right)^2 \right] dx dy \quad (2.5)$$

,where $\nabla I = (\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y})$ is the spatial intensity gradient, I_t the pixel colour intensity, $\vec{v} = (v_x, v_y)$ the image velocity at point (x, y) and λ a regularization constant.

In contrast, Lucas uses a local least squares calculation to provide the constraint of equation 2.6 (Lucas (1981)).

$$\sum_{x,y \in \Omega} W^2(x, y) [\nabla I(x, y, t) \cdot \vec{v} + I_t(x, y, t)]^2 \quad (2.6)$$

Where $W^2(x, y)$ is a weighting function that attributes more weight to the central pixels, $\nabla I(x, y, t)$ the spatial intensity gradient and $\vec{v} = (v_x, v_y)$ the image velocity at point (x, y) and time instant t .

2.1.2 Object Tracking

Tracking is the process of locating the position (state) of a moving object(s) in time. The tracking algorithm analyses the video frames and outputs the estimate of the position (state) of the moving target(s) within each video frame.

Tracking is different from detection in a way that detection explores all possible object's positions configurations since there is no prior knowledge on the state, while in tracking the objective is to evaluate the existence of the object near a predicted location.

The following subsections explain some of the methodologies that have been proposed to deal with the tracking problem, which include CAMSHIFT, different types of graphs, Kalman Filters and Particle Filters.

2.1.2.1 CAMSHIFT

The Continuously Adaptive Mean Shift (CAMSHIFT) algorithm ([Bradski \(1998\)](#)) is based on the Mean Shift Algorithm ([Comaniciu and Meer \(1997\)](#)), a robust non-parametric iterative technique for finding the mode of probability distributions. The block diagram of CAMSHIFT can be seen in Figure 2.2. In this method for each video frame, the raw image is converted into a colour probability distribution image via a colour histogram model of the colour being tracked. The colour probability distribution is computed through a Histogram Back-Projection method. Histogram Back-Projection is a primitive operation that associates the pixel values in the image with the value of the corresponding histogram bin. The probability image is computed by comparing the target histogram with any consecutive histogram and each pixel value is characterized by the probability of the input pixel belonging to the target histogram.

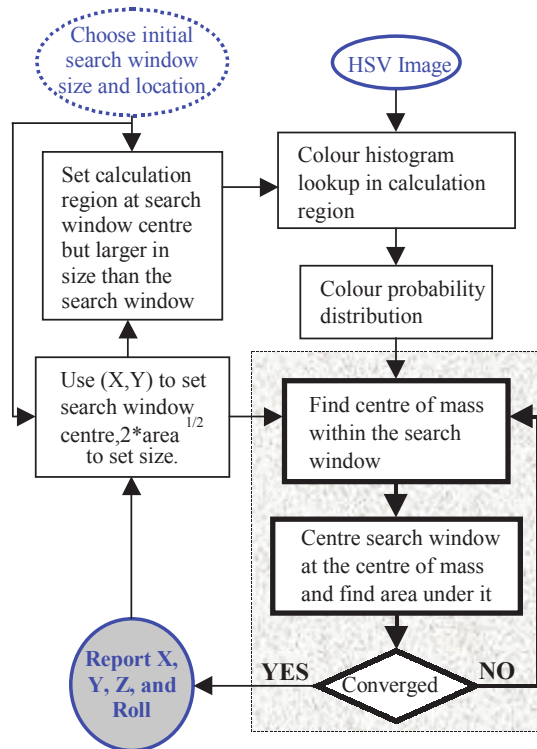


Figure 2.2: Block diagram of CAMSHIFT (adapted from [Bradski \(1998\)](#)).

Afterwards the centroid of the area is computed using the Mean Shift algorithm. The Mean Shift uses the $zero^{th}$ and first moments of the colour probability histogram. The search window in

the next video image is then centred over this centroid and the size is adjusted as a function of the $zero^{th}$ moment. This algorithm has real practical applications for skin colour object tracking.

2.1.2.2 Graphs

Although most graphs fail on the initial definition of tracking (they do not limit the search, but rather consist in a matching problem of the detected objects), they are commonly used on the tracking phase.

There are several types of graphs depending on their characteristics. A simple graph, G , is defined as $G = (\mathfrak{V}, A)$, where \mathfrak{V} is a finite set of vertices of G , while A is a set of edges that connect elements of \mathfrak{V} .

A weighted graph, $G = (\mathfrak{V}, A)$, is a graph where each arc (i, j) has an associated weight w_{ij} . The weight can represent concepts such as affinity, distance or connection cost.

A directed weighted graph is similar to a weighted graph, however each arc has a well-established direction.

Figure 2.3 illustrates these three types of graphs.

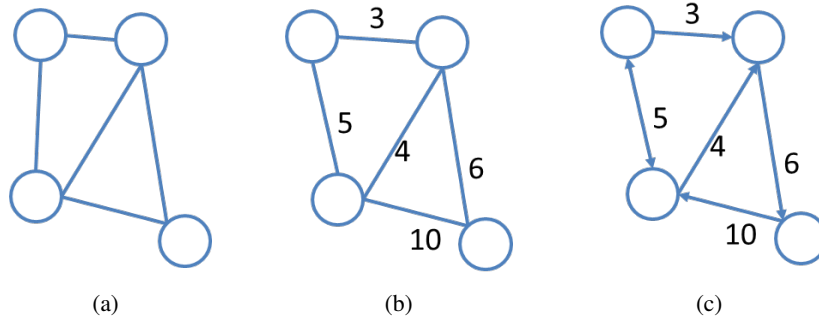


Figure 2.3: Examples of graphs: (a) simple graph (b) weighted graph (c) directed weighted graph.

2.1.2.3 Kalman Filter

The Kalman Filter (Welch and Bishop (2002)) was developed by Rudolf Kalman (Kalman (1960)) and consists in an efficient recursive filter that estimates the state of a dynamic system from a series of incomplete and noisy measurements. It consists of two alternating operations: predicting the new state and its uncertainty, and correcting it with the new measurement.

The state estimation $x \in \mathfrak{R}^n$ of a discrete-time process is modelled by the linear stochastic difference equation 2.7 with a measure $z \in \mathfrak{R}^m$ (Equation 2.8).

$$x_k = Ax_{k-1} + Bu_k + w_{k-1} \quad (2.7)$$

$$z_k = Hx_k + v_k \quad (2.8)$$

Where A is the state model matrix, B is the input model matrix, H is the observation model matrix, u_k is the known input vector state at step k and the random variables w_k and v_k represent the process and measurement noise. They are assumed to be independent (of each other), white, and with normal probability distributions (Equation 2.9).

$$p(w) \sim N(0, Q)p(v) \sim N(0, R) \quad (2.9)$$

The predicting operation of the Kalman Filter is responsible for projecting forward (in time) the current state (\hat{x}_{k-1}) and error covariance (P_{k-1}) estimates to obtain the *a priori* estimates (\hat{x}_k^- and P_k^-) for the next time step according to equations 2.10 and 2.11.

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_k \quad (2.10)$$

$$P_k^- = AP_{k-1}A^T + Q \quad (2.11)$$

The correcting operation, (equation 2.12 and 2.13), incorporates a new measurement (z_k) into the *a priori* estimate to obtain an improved *a posteriori* estimate (\hat{x}_k) and error covariance (P_k). K_k is the Kalman Gain and is obtained according to 2.14.

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) \quad (2.12)$$

$$P_k = (I - K_kH)P_k^- \quad (2.13)$$

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1} \quad (2.14)$$

The complete cycle of the Kalman Filter is illustrated by Figure 2.4.

One of the great limitations of the Kalman Filter is that it assumes the system is linear. In order to overcome this limitation the Extended Kalman Filter (EKF) or the Unscented Kalman Filter (UKF) can be used instead.

The Extended Kalman Filter linearizes the state around the current mean and covariance using the partial derivatives of the process and measurement functions to compute estimates and is governed by the non-linear stochastic equations 2.15 and measure 2.16 that are linearized into equations 2.17 and 2.18, respectively.

$$x_k = f(x_{k-1}, u_k, w_{k-1}) \quad (2.15)$$

$$z_k = h(x_k, x_k) \quad (2.16)$$

$$x_k \sim \tilde{x}_k + A(x_{k-1} - \hat{x}_{k-1}) + Ww_{k-1}, \tilde{x}_k = f(\hat{x}_{k-1}, u_k, 0) \quad (2.17)$$

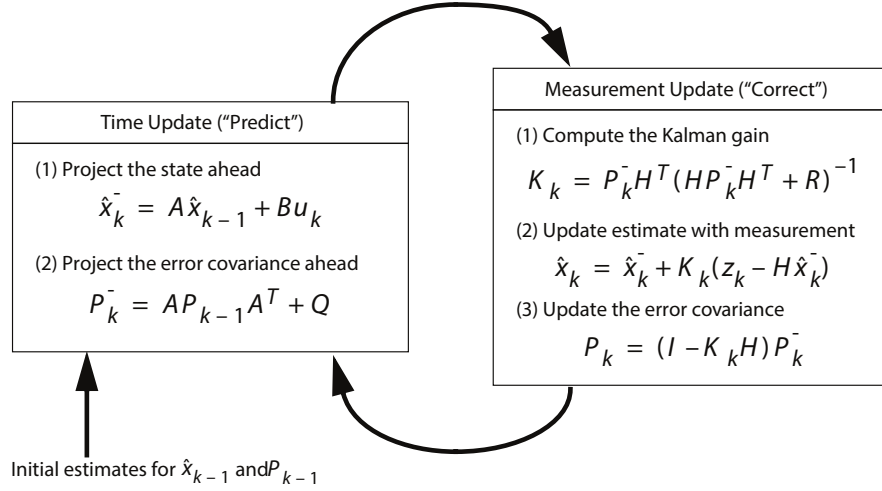


Figure 2.4: Complete cycle of the Kalman Filter (adapted from Welch and Bishop (2002)).

$$z_k \sim \tilde{z}_k + H(x_{k-1} - \tilde{x}_{k-1}) + Vv_k, \tilde{z}_k = h(\hat{x}_k, 0) \quad (2.18)$$

In equations 2.17 and 2.18, x_k is an *a posteriori* estimate of the state at step k , w_k and v_k represent the process and measurement noise, A is the Jacobian matrix of partial derivatives of f with respect to x , W is the Jacobian matrix of partial derivatives of f with respect to w , H is the Jacobian matrix of partial derivatives of h with respect to x and V is the Jacobian matrix of partial derivatives of h with respect to v .

These premises lead to different equations as illustrated in Figure 2.5.

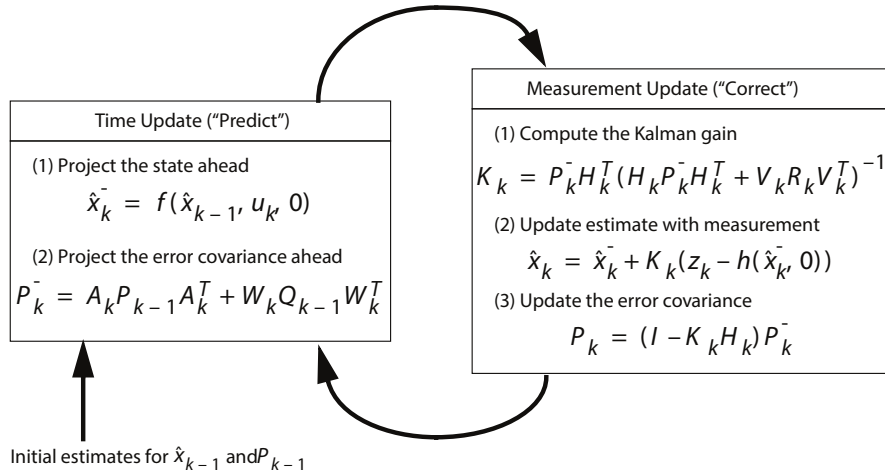


Figure 2.5: Complete cycle of the Extended Kalman Filter (adapted from Welch and Bishop (2002)).

In the EKF, the state distribution is approximated through the first-order linearization of the

nonlinear system which can introduce large errors in the true *a posteriori* mean and covariance values and lead to sub-optimal performance and sometimes divergence of the filter.

The Unscented Kalman Filter (Julier and Uhlmann (1997)) addresses this problem by using a deterministic sampling technique known as the Unscented Transform (UT) to pick a minimal set of sample points (called sigma points) around the mean. These sigma points are then propagated through the non-linear functions, from which the mean and covariance of the estimate are then recovered.

The result is a filter that more accurately captures the true mean and covariance with no extra computational complexity. The usage of UT results in approximations that are accurate to the third order for Gaussian inputs and at least to the second-order for non-Gaussian inputs. The cycle of the Unscented Kalman Filter is presented in Figure 2.6.

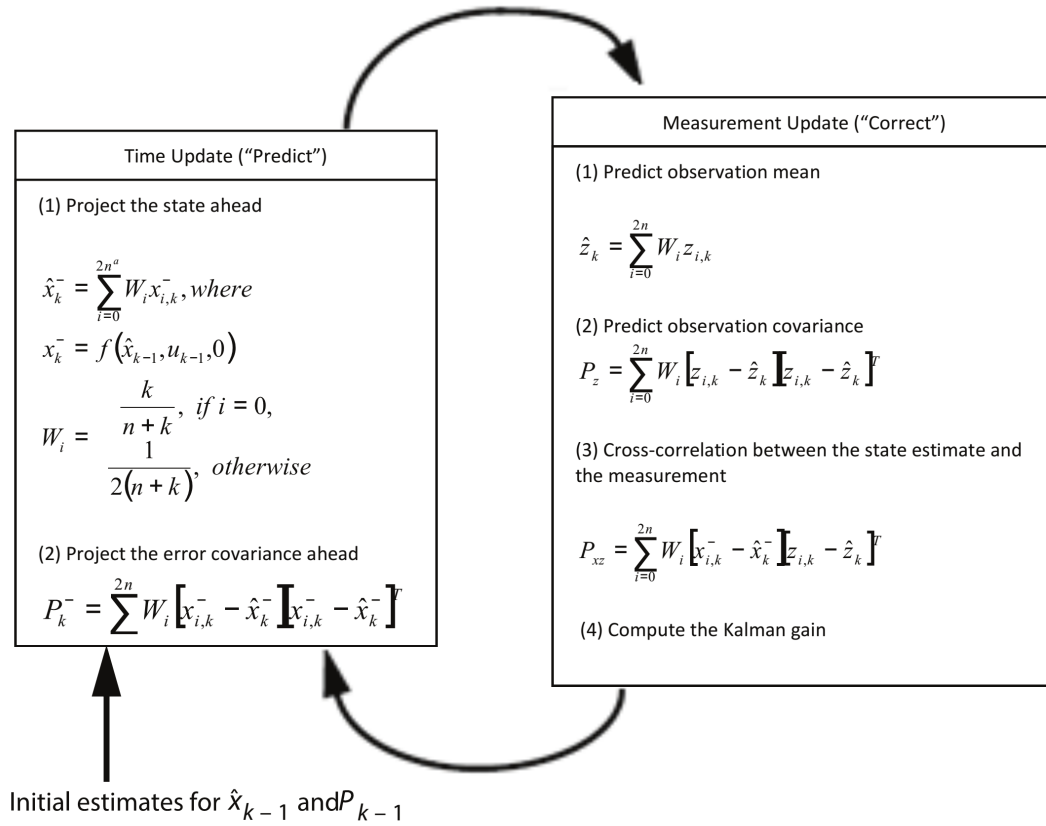


Figure 2.6: Complete life cycle of the Unscented Kalman Filter.

2.1.2.4 Particle Filter

Like the EKF and UKF, Particle Filters (Arulampalam et al. (2002)) are able to deal with non-linear systems with the advantage of dealing with non-Gaussian distributions. However, it is important to notice that if the assumptions for Kalman filters are valid, no Particle Filter can outperform them,

additionally depending on the dynamic model, Unscented Kalman filters or Extended Kalman filters may produce satisfactory results at lower computational costs (Arulampalam et al. (2002)).

The Sequential Importance Sampling (SIS) algorithm also known as Bootstrapping Filtering, Condensation Algorithm, Particle Filtering or survival of the fittest, is a technique for implementing a recursive Bayesian filter by Monte Carlo (MC) simulations. The required state transition posterior density function ($p(x_{0:k}|z_{1:k})$) is represented by a set of random samples with associated weights (Equation 2.19), where the random measure is represented by $\{x_{0:k}^i, w_k^i\}_{i=1}^{N_S}$, and consists of a set of support points $\{x_{0:k}^i, i = 1, \dots, N_S\}$ with associated weights $\{w_k^i, i = 1, \dots, N_S\}$. $x_{0:k} = \{x_j, j = 0, \dots, k\}$ is the set of all states up to time k , while δ is the Dirac delta function.

$$p(x_{0:k}|z_{1:k}) \approx \sum_{i=1}^{N_S} w_k^i \delta(x_{0:k} - x_{0:k}^i) \quad (2.19)$$

As the number of samples becomes very large, this MC characterization becomes an equivalent representation of the usual functional description of the posterior density function and the SIS filter approaches the optimal Bayesian estimate.

Usually it is not possible to draw samples x_k^i directly from the posterior density function $p(x_{0:k}^i|z_{1:k})$, therefore samples are obtained directly from a (different) density function $q(x_{0:k}^i|z_{1:k})$ that obeys to equation 2.20 and can be chosen freely.

$$w_k^i \propto \frac{p(x_{0:k}^i|z_{1:k})}{q(x_{0:k}^i|z_{1:k})} \quad (2.20)$$

If the importance density function is chosen so that it obeys to equation 2.21, then the new samples $x_{0:k}^i \sim q(x_{0:k}|z_{1:k})$ can be obtained by augmenting each of the existing samples $x_{0:k}^i \sim q(x_{0:k-1}|z_{1:k-1})$ with the new state $x_k^i \sim q(x_k|x_{0:k-1}, z_{1:k})$ where the weight update follows equation 2.22.

$$q(x_{0:k}^i|z_{1:k}) = q(x_k|x_{0:k-1}, z_{1:k})q(x_{0:k-1}|z_{1:k-1}) \quad (2.21)$$

$$w_k^i \propto w_{k-1}^i \frac{p(z_k|x_k^i)p(x_k^i|x_{k-1}^i)}{q(x_k^i|x_{0:k-1}^i, z_{1:k})} \quad (2.22)$$

Furthermore, if q is only dependent on the last state and on the observation then the posterior filtered density function can be approximated by equation 2.23.

$$p(x_k|z_{1:k}) \approx \sum_{i=1}^{N_S} w_k^i \delta(x_k - x_k^i) \quad (2.23)$$

The SIS algorithm is then a recursive propagation of weights and support points as each measurement is received. Figure 2.7 represents the pseudo code for the SIS particle filter.

The main problem with the Sequential Importance Sampling algorithm is that after a few iterations, the weights are concentrated on a few particles (degeneracy problem). There are some

 ALGORITHM 1: SIS PARTICLE FILTER

```

 $[\{\mathbf{x}_k^i, w_k^i\}_{i=1}^{N_s}] = \text{SIS} [\{\mathbf{x}_{k-1}^i, w_{k-1}^i\}_{i=1}^{N_s}, \mathbf{z}_k]$ 
• FOR  $i = 1 : N_s$ 
  – Draw  $\mathbf{x}_k^i \sim q(\mathbf{x}_k | \mathbf{x}_{k-1}^i, \mathbf{z}_k)$ 
  – Assign the particle a weight,  $w_k^i$ , according to (48)
• END FOR

```

Figure 2.7: Pseudo-code for SIS particle filter in [Arulampalam et al. \(2002\)](#).

options to overcome this problem, namely the usage of many samples, a good choice of the importance density function or resampling.

The basic idea behind resampling is to eliminate particles that have small weights and to concentrate on particles with large weights "Survival of the fittest". The resampling step involves generating a new set $\{\mathbf{x}_k^{i*}\}_{i=1}^{N_s}$ by resampling (with replacement) N_s times from an approximated discrete representation of $p(x_k | z_{1:k})$ (Equation 2.23) so that $P(x_k^{i*} = x_k^j) = w_k^j$. The pseudo-code for the resampling algorithm is shown in Figure 2.8.

 ALGORITHM 2: RESAMPLING ALGORITHM

```

 $[\{\mathbf{x}_k^{j*}, w_k^j, i^j\}_{j=1}^{N_s}] = \text{RESAMPLE} [\{\mathbf{x}_k^i, w_k^i\}_{i=1}^{N_s}]$ 
• Initialise the CDF:  $c_1 = 0$ 
• FOR  $i = 2 : N_s$ 
  – Construct CDF:  $c_i = c_{i-1} + w_k^i$ 
• END FOR
• Start at the bottom of the CDF:  $i = 1$ 
• Draw a starting point:  $u_1 \sim \mathcal{U}[0, N_s^{-1}]$ 
• FOR  $j = 1 : N_s$ 
  – Move along the CDF:  $u_j = u_1 + N_s^{-1}(j - 1)$ 
  – WHILE  $u_j > c_i$ 
    *  $i = i + 1$ 
  – END WHILE
  – Assign sample:  $\mathbf{x}_k^{j*} = \mathbf{x}_k^i$ 
  – Assign weight:  $w_k^j = N_s^{-1}$ 
  – Assign parent:  $i^j = i$ 
• END FOR

```

Figure 2.8: Pseudo-code for resampling algorithm in [Arulampalam et al. \(2002\)](#).

The resampling algorithm eliminates the degeneracy problem, however it continuously selects particles with high weight which can cause loss of diversity or sample impoverishment. In order to counteract these problems some new special cases of the SIS algorithm have been proposed. These special cases can be derived from the SIS algorithm by an appropriate choice of importance

sampling density and/or modification of the resampling step. Sampling Importance Resampling (SIR), Auxiliary Sampling Importance Resampling (ASIR) or Regularized Particle Filter (RPF) are just some examples.

2.2 Image Recognition

Image recognition is the problem of determining whether or not the image contains/represents some specific object, scene or activity. This task is usually very simple for humans, nevertheless it is still an open issue in computer vision where most image recognition methods can solve this problem only for specific situations.

A typical image recognition system is composed of two main blocks: one that is responsible for extracting the features of interest (Feature Extraction) and another that performs the classification relying on the features extracted (Classifier).

Features can be classified into low-level, mid-level and high-level features (Coimbra (2013)). Low-level features include specific image and video features such as colour, texture, shape and motion. Mid-level features contain some subjectivity and usually are the result of a segmentation process or identification. High-level features already have a semantic interpretation and knowledge.

Sometimes the dimensionality of the extracted features can be too high and some of the features may be correlated, therefore, features dimensionality can be reduced by redesigning the features and selecting an appropriate subset among the existing features (feature selection) or transforming them into a different feature space either by using Principal Components Analysis (PCA) or Linear Discriminant Analysis (LDA) (Bishop (2006)). PCA seeks a projection that best represents the data in a least squares sense, while LDA seeks a projection that best separates the data in a least squares sense.

The purpose of the Classifier is to transform low-level features into high-level features that can be understood by humans and used to take some kind of decision or action. Several classifiers have been proposed as denoted in Figure 2.9.

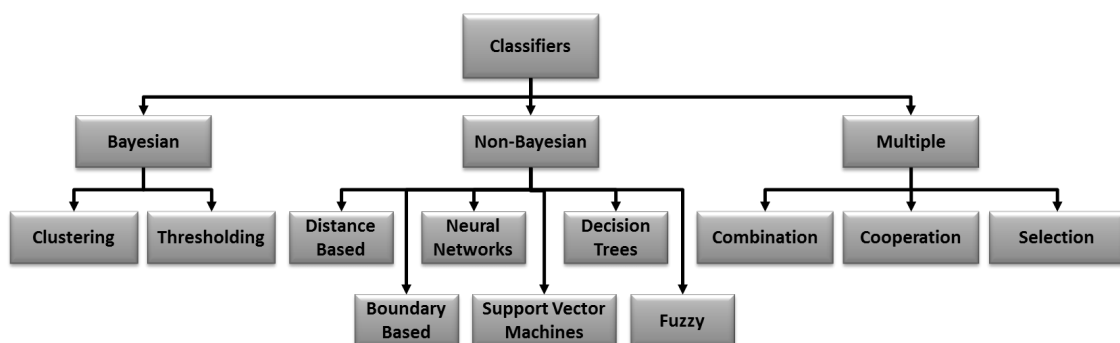


Figure 2.9: Overview of types of classifiers.

These classifiers will be discussed in the following subsections.

2.2.1 Bayesian Classifiers

Bayesian classifiers have their foundations on the probability theory and, on average, these classifiers yield the lowest probability of performing classification errors (Gonzalez and Woods (2002)). In fact, the average loss of assigning a sample x , to a given class w_j when it came from w_k is given by equation 2.24, where L_{kj} represents the loss due to the misclassification and $p(w_k | x)$ the probability that sample x comes from class w_k .

$$r_j(x) = \sum_{k=1}^w L_{kj} p(w_k | x) \quad (2.24)$$

Applying the Bayes theorem, $P(A | B) = P(B | A)P(A)/P(B)$, we get equation 2.25, where $p(x | w_k)$ is the probability density function of the samples from class w_k and $P(w_k)$ is the probability of occurrence of class w_k .

$$r_j(x) = \frac{1}{p(x)} \sum_{k=1}^w L_{kj} p(x | w_k) P(w_k) \quad (2.25)$$

The classifier has W possible classes to choose and this choice, for a Bayesian Classifier, is made based on the class that minimizes the total average loss. Thus, sample x will be assigned to class w_i if $r_i(x) < r_j(x)$ for $j=1,2,\dots,W, j \neq i$.

2.2.2 Non-Bayesian Classifiers

Contrary to Bayesian classifiers, non-Bayesian classifiers use training data to learn the classifiers directly without estimating any probabilistic structure. There are two main groups of non-Bayesian classifiers: distance based and boundary based.

2.2.2.1 Distance Based Classifiers

Distance based classifiers use some kind of distance measure in order to classify a given sample.

The Nearest Neighbour Classifier partitions the feature space into cells consisting of all points closer to a given training point than to any other training point. It classifies a sample by assigning it the label associated with its closest neighbour, given a distance function. All points in such a cell are labelled by the class of the training point, forming a Voronoi tessellation of the feature space.

The k-Nearest Neighbour Classifier assigns to the sample the label most frequently represented among the k nearest samples, so the decision is based on examining the labels of the k -nearest neighbours and taking a vote.

The closeness on these two classifiers is defined by some kind of distance measure such as the Euclidean, Manhattan or Minkowski distance.

2.2.2.2 Boundary Based Classifiers

Boundary based methods define a boundary between classes. An example of a boundary based classifier is the Linear Discriminant Classifiers (Bishop (2006)). This classifier is based on discriminant functions that take as input a vector x with the sample features and assign it to one of K classes, denoted C_k . The linear discriminant analysis method consists in searching, some linear combinations of selected variables (by using weights - w_k), which provide the best separation between the considered classes (Equation 2.26).

$$y_k(x) = w_k^T x + \omega_{k_0} \quad (2.26)$$

Linear Discriminant Analysis can also be seen as a way to perform dimension reduction, so the input vector may have D-dimensions but it is projected into a one dimension space. In general, the projection onto one dimension leads to a considerable loss of information and usually classes that are well separated in the original D-dimensional space may become strongly overlapped in one dimension. Therefore Fisher proposed a function that maximizes the separation between the projected class means while also giving a small variance within each class, thereby minimizing the class overlap. Fisher formulates equation 2.26 as 2.27 and defines the weights (w) as 2.28.

$$y_k(x) = w_k^T x \quad (2.27)$$

$$\begin{aligned} w_k &\propto S_{W_k}^{-1}(m_{k+1} - m_k), \text{ where} \\ m_k &= \frac{1}{N_k} \sum_{n \in C_k} x_n ; m_{k+1} = \frac{1}{N_{k+1}} \sum_{n \in C_{k+1}} x_n \\ S_w &= \sum_{n \in C_k} (x_n - m_k)(x_n - m_k)^T + \sum_{n \in C_{k+1}} (x_n - m_{k+1})(x_n - m_{k+1})^T \\ &\text{, where } N_{k(+1)} \text{ is the number of samples of class } C_{k(+1)} \end{aligned} \quad (2.28)$$

2.2.2.3 Neural Networks

A neural network, from a classification perspective, can be seen as a mapping function where a d-dimensional input x is submitted to the network and a M-vector network output y is obtained to make the classification decision. A neural network is composed of a number of nodes connected by links that have associated a weight (Russell and Norvig (1995)).

Each node receives signals from its inputs and computes a new activation level that sends to each of its output links (Figure 2.10).

The output computation comprises two components: the first component consists in a weighted ($W_{j,i}$) sum of the unit's input values (a_j) represented in 2.29 and the second on an activation function, g , which transforms the weighted sum into the final values and serves as the unit's

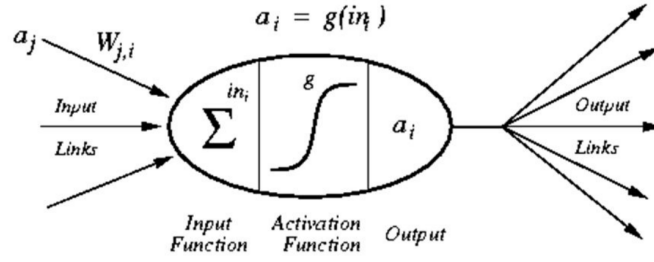


Figure 2.10: A neural network node (adapted from [Russell and Norvig \(1995\)](#)).

activation value, a_i , for the following level. Different functions can be used for g , namely the step, sign or sigmoid functions.

$$in_i = \sum_{j=0}^n W_{j,i} a_j = \mathbf{W}_i \cdot \mathbf{a}_i \quad (2.29)$$

There are two main groups of network structures, the feed-forward and the recurrent networks. In feed-forward networks, links are unidirectional and do not contain cycles, while in recurrent networks the links do not have restrictions ([Russell and Norvig \(1995\)](#)).

Feed-forward networks do not have an internal state and in order to build the network it is only necessary to define the structure and compute the weights associated to each link. Usually this is done using a back-propagation learning algorithm.

Recurrent networks are much more complex and have internal state stored in the activation levels. Examples of recurrent networks are the Hopfield networks ([Sathasivam and Abdullah \(2008\)](#)) and the Boltzmann machines ([Ackley et al. \(1985\)](#)).

2.2.2.4 Support Vector Machines

A Support Vector Machine (SVM) ([Burges \(1998\)](#)) performs classification by constructing a N-dimensional hyper plane that optimally separates the data into two categories. SVM models are closely related to neural networks. In fact, a SVM model using a sigmoid kernel function is equivalent to a two-layer, perceptron neural network.

Considering the training data x_i, y_i , $i=1..l$, $y_i \in -1, 1$, $x_i \in R^d$ and supposing a hyper plane that separates the data in two classes $(-1, 1)$ the points x which lie on the hyper plane satisfy $w \cdot x + b = 0$ and all data must satisfy the following constraints:

$$\begin{aligned} \mathbf{x}_i \cdot \mathbf{w} + b &\geq 1 \text{ for } y_i = 1 \\ \mathbf{x}_i \cdot \mathbf{w} + b &\leq -1 \text{ for } y_i = -1 \end{aligned} \quad (2.30)$$

Let d_+ (d_-) be the shortest distance from the separating hyper plane to the closest positive (negative) example and $d_+ + d_-$ the "margin" of a separating hyper plane, then for the linearly

separable case, the support vector algorithm simply looks for the separating hyper plane with the largest margin, the "maximum margin hyper plane". The instances that are closest to the maximum hyper plane are called support vectors. There is at least one support vector for each class but there can be more.

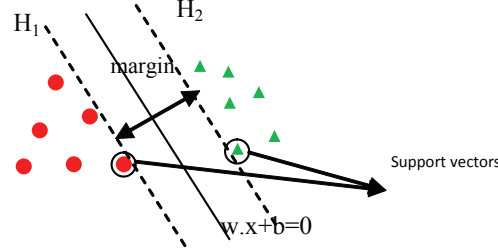


Figure 2.11: Support vectors.

Furthermore, d_+ and d_- are equal to $1/\|w\|$, therefore it is possible to find the hyper plane which gives the maximum margin by minimizing $\|w\|^2$. To solve this problem the Lagrangian multipliers method is used as denoted on equation 2.31.

$$L_p = \frac{1}{2}\|w\|^2 - \sum_{i=1}^l \alpha_i y_i (\mathbf{x}_i \cdot \mathbf{w} + b) + \sum_{i=1}^l \alpha_i \quad (2.31)$$

Equation 2.31 must be minimized with respect to w and b , and simultaneously require that the derivatives of L_p with respect to all the α_i vanish, all subject to the constraint $\alpha_i \geq 0$.

For the case where classes are not separable, SVM can be extended by incorporating positive slack variables (ζ). To incorporate these slacks, the constraints established in equation 2.30 must be changed according to 2.32 and so it is necessary to minimize $\frac{\|w\|^2}{2} C \sum_i \zeta_i$ which yields minimizing Lagrangian of equation 2.33. In this equation C is a parameter chosen by the user that is directly proportional to the penalty errors intended, while μ_i are the Lagrange multipliers introduced to enforce positivity to x_i .

$$\begin{aligned} \mathbf{x}_i \cdot \mathbf{w} + b &\geq 1 - \zeta_i \text{ for } y_i = 1 \\ \mathbf{x}_i \cdot \mathbf{w} + b &\leq -1 + \zeta_i \text{ for } y_i = -1 \end{aligned} \quad (2.32)$$

$$L_p = \frac{1}{2}\|w\|^2 + C \sum_{i=1}^l \alpha_i \{y_i (\mathbf{x}_i \cdot \mathbf{w} + b) - 1 + \zeta_i\} + \sum_{i=1}^l \mu_i \zeta_i \quad (2.33)$$

2.2.2.5 Decision Trees

Decision Trees (Mitchell (1997)) are hierarchical decision systems in which conditions are sequentially tested until a class is accepted. To this end, the feature space is split into unique regions, corresponding to the classes, in a sequential manner. Upon the arrival of a feature vector, the

searching of the region to which the feature vector will be assigned is achieved via a sequence of decisions along a path of nodes of an appropriately constructed tree.

In the decision tree representation, each internal node tests an attribute, each branch corresponds to an attribute value and each leaf assigns a classification. The basic decision tree learning algorithm, ID3 (Iterative Dichotomiser 3), employs a top-down, greedy search through the space of possible decision trees selecting the attribute that is most useful for classifying examples. In order to measure the worth of an attribute, a statistical property known as information gain is defined. This property measures how well a given attribute separates the training examples, according to their target classification.

The information gain takes into account an entropy measure (Equation 2.34) and corresponds to the expected reduction in entropy caused by partitioning the examples according to it (Equation 2.35).

$$E(s) = -p_+ \log_2 p_+ - (-p_- \log_2 p_-) \quad (2.34)$$

$$G(s, a) = E(s) - \sum_{v \in V(a)} \frac{|S_v|}{|S|} E(s_v) \quad (2.35)$$

Where, p_+ and p_- represent the proportion of positive and negative samples of S , respectively, S_v is a subset of S for which the attribute a takes the value v , $|S_v|$ and $|S|$ are the number of elements in S_v and S .

2.2.2.6 Fuzzy Classifiers

As defined by Kuncheva (2000), "a fuzzy classifier is any classifier which uses fuzzy sets either during its training or during its operation".

Fuzzy theory was proposed by Lofti Zadeh in 1965 (Zadeh (1965)), as a way of dealing with vagueness and imprecision in pattern classification and information processing areas.

Considering the space of objects X , traditional set theory attributes to each object, x , a belong ($\mu_A(x) = 1$) or no belong ($\mu_A(x) = 0$) degree to a given class A . On the other hand, on fuzzy theory each object has a given belonging degree (or membership) $\mu_A(x)$ in the interval $[0;1]$ to a fuzzy set (class). In case of the object not belonging to the set then $\mu_A(x) = 0$, if the object fully belongs to the set then $\mu_A(x) = 1$ and if $0 < \mu_A(x) < 1$, then x is considered to be a fuzzy member of A . In addition, an object can belong to different fuzzy sets with different belonging degrees (for example, we could have $\mu_A(x) = 0.5$ and $\mu_B(x) = 0.3$). This methodology is the basis for the already mentioned Fuzzy C-Means algorithm.

Other Fuzzy classifiers use the entire fuzzy engine (Fuzzy if-then systems) that include the fuzzification, fuzzy inference and finally the defuzzification steps as highlighted by Kuncheva (2000).

On these classifiers, n inputs ($x = [x_1, \dots, x_n]^T \in \Re^n$) are first fuzzified, which correspond to determining the degree of membership of each input to each fuzzy logic membership function.

These membership functions establish the relationship between the values of an element and its degree of membership in a set.

To better understand each of these steps, let's consider the example given in [Negnevitsky \(2001\)](#) to classify projects according to the project funding and staffing. The sets defined for the project funding are **inadequate**, **marginal** and **adequate**, while the sets for project staffing are **small** and **large** and take the membership functions defined in Figure 2.12.

According to the characteristics of the project, its risk can be classified as high, normal or low according to the following rules:

- RULE1: IF *project funding* is adequate OR *project staffing* is small THEN *risk* is low
- RULE2: IF *project funding* is marginal AND *project staffing* is large THEN *risk* is normal
- RULE3: IF *project funding* is inadequate THEN *risk* is high

Considering a given project with a project funding of 35% and project staffing of 60%, the fuzzification step results in a membership degree of 0.5, 0.2 and 0.0 to the inadequate, marginal and adequate input sets, respectively, and a degree of membership of 0.1 and 0.7 to the small and large output sets, respectively.

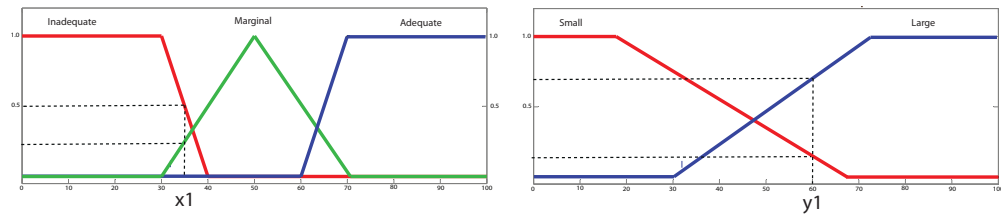


Figure 2.12: Fuzzification step (adapted from [Negnevitsky \(2001\)](#)).

Once the membership degrees have been determined it is possible to perform the rule evaluation, that consists in evaluating the antecedents of the fuzzy rules, which are then applied to the consequent membership function as illustrated in Figure 2.13. (Note: this is a Mandani-Assilian model, where both the antecedents as well as the consequents are represented by linguistic terms. On the Takagi-Sugeno fuzzy systems the consequents are a function, usually a polynomial).

The defuzzification step is the last one and obtains a crisp value out of the aggregated output fuzzy set. One of the most common methods, consists in determining the centre of gravity of the output fuzzy set (COG), according to equation 2.36.

$$COG = \frac{\int_a^b \mu_A(x) x dx}{\int_a^b \mu_A(x) dx} \quad (2.36)$$

In the example given the outcome of the defuzzification is 64.7 (Figure 2.14), which indicates that the risk associated with the project is of 64.7%.

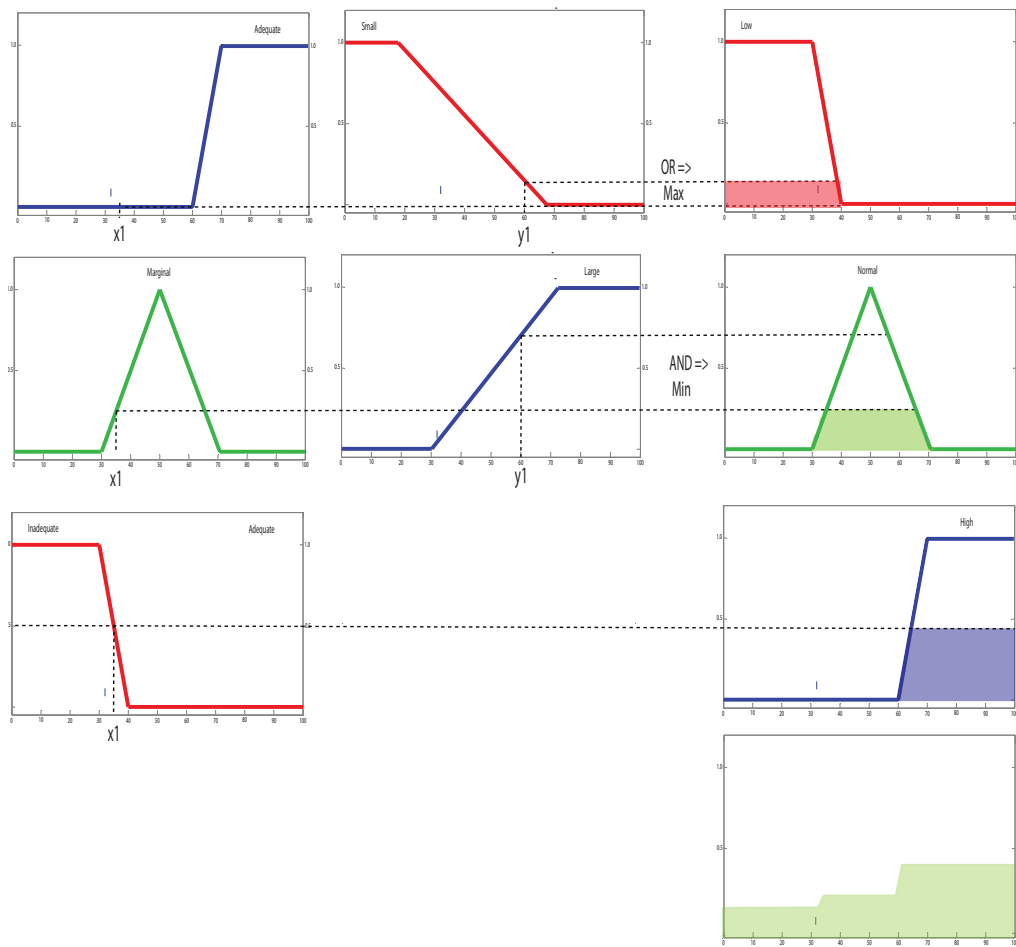


Figure 2.13: Fuzzy inference step (adapted from [Negnevitsky \(2001\)](#)).

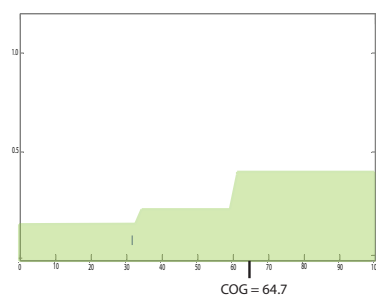


Figure 2.14: Defuzzification step (adapted from [Negnevitsky \(2001\)](#)).

2.2.3 Multiple Classifiers

As stressed by [Kuncheva \(2005\)](#) "By combining classifiers we are aiming at a more accurate classification decision at the expense of increased complexity."

According to [Gunes et al. \(2010\)](#) there are three categories of classification systems based on the type of operation among classifiers: combination, cooperation or selection classifiers.

On combination classifiers each classifier is applied in parallel and the resulting outputs are combined using voting methods, unanimous consensus, maximum/minimum/mean/product rules, Bayesian theory, Possibility theory, Belief Theory or Fuzzy Sets Theory.

Combination classifiers can only be effective if the outputs of the different classifiers are decorrelated, which can be achieved by using different feature spaces, training sets or classification methods.

Two widely used techniques, Bagging ([Breiman \(1996\)](#)) and Boosting ([Schapire et al. \(1998\)](#)), have been used to guarantee this decorrelation on the training sets space.

Boosting attributes weights to each instance in the training set. These weights are updated at every training cycle according to the performance of the classifier. Initially, all weights are set equally, but on each round, the weights of incorrectly classified samples are increased so that on the next rounds the classifier effort will focus on the miss classified examples in the training set. This method also assigns weights to each classifier prior to the combination phase.

Bagging is based on creating several individual classifiers by using the same classifier but different subsets of the training set. These subsets are obtained using the bootstrap technique which aims to reduce the error of statistical estimators.

Cooperation classifiers use the information from one classifier to aid other classifiers making a decision. Usually cooperative systems can use different types of classifiers in the classification process because the information transmitted between them is the decision vector.

Classifiers used in cooperation can be specialized in one the following three characteristics: feature space, decision space or representation space. Neural networks are a good example of classifiers cooperating.

Algorithms based on a selection of classifiers can be static or adaptive, depending if the system of classifiers used at any k moment is that of the learning stage (first case) or is the one that best fits the characteristics of the pattern being classified (second case).

2.3 Knowledge Modelling and Representation

According to the Free On-line Dictionary of Computing ([Foldoc \(1994\)](#)), knowledge representation is:

"The subfield of artificial intelligence concerned with designing and using systems for storing knowledge - facts and rules about some subject.

A body of formally represented knowledge is based on a conceptualisation - an abstract view of the world that we wish to represent. In order to manipulate this knowledge we must specify how the abstract conceptualisation is represented as a concrete data structure. Ontology is an explicit specification of a conceptualisation."

Knowledge representation appears in different forms, first order predicate logic, frames, property lists, semantic nets, direct representation, procedure/subroutine and procedure/production systems (Davis and Yen (1998)).

As stated on the Free On-line Dictionary of Computing this conceptualisation can be achieved through ontologies, in fact using ontologies it is possible to specify knowledge in a conceptual way in terms of symbols that represent concepts and their relations. These symbols and relations are defined using a formal language. There are various ontology languages, based on different knowledge representation formalisms.

2.3.1 Knowledge Representation Formalisms

First-order predicate logic is commonly used in mathematics to prove theorems. It uses qualifiers and logical operators to describe objects, properties, situations, and relationships. It is a simple formalism, flexible and modular, however lacks organizational principles and has problems on representing procedural or heuristic knowledge.

Frames look much like modern classes, without the methods and have two main parts: slots to hold variable data (properties or attributes) and fixed parts to hold static data. Data inside the frame can be of any kind, including procedures. Frames organize knowledge for easy retrieval, reference and maintenance.

Property lists organize data into named values and lists of properties (or attributes) to describe the state of the world using several object types. All appropriate properties for an object are grouped into a list, and lists are easily structured in LISP (List Processing Language), a popular expert system programming language. However, property lists are not very effective for inference-oriented operations.

On semantic nets, knowledge consists of a collection of objects, object properties, concepts and events represented by nodes connected by arcs (relationships) that allow inheritance of properties. With a semantic net, important associations and relationships can be described explicitly and the inheritance hierarchy is easy to understand and revise. However establishing an inheritance hierarchy is a difficult task and finding a specific piece of information can be inefficient (heuristic inadequacy).

Direct representation also called analogical representation is based on analogy and can represent knowledge about certain aspects of the world in natural ways. Representations such as maps, models, diagrams and music sheets are direct representations of knowledge. Direct representation facilitates searching because the important constraints are already known. However it tends to represent specific instances inappropriately when generality is needed.

In the procedure/subroutine approach, knowledge is contained in procedures, small programs that know how to do specific things. This formalism is good for representing heuristic knowledge, modelling complex meta-problems, performing extended logical inferences and reasoning. Nevertheless, procedures and subroutines are difficult to verify or change and the information needed to control the subroutines can limit or even exclude significant alternatives or information.

Procedural knowledge is related to the procedure of carrying out an action. In this case, knowledge is represented by a collection of loosely coupled procedures, which may be organized into sets. Procedural knowledge allows easily adding, removing or updating information and is consistent with compiler design. Because of the complexity of the logic, program execution is inefficient and the process is constrained by the predetermined control flow imposed by the program.

2.3.2 Ontologies

Besides knowledge representation, ontologies have also been used in data bases, software engineering, electronic commerce and semantic web, among others. Therefore ontology specification languages are usually divided into classical and web-based ([Corcho and Gomez-Perez \(2000\)](#)).

Classical languages include Ontolingua, Open Knowledge Base Connectivity (OKBC), Operational Conceptual Language (OCML), Frame Logic (FLogic), CycL and LOOM and web-based include eXtensible Markup Language (XML), Resource Description Framework (RDF), XML-Based Ontology Exchange Language (XOL), Simple HyperText Markup Language (HTML) Ontology Extension (SHOE), Ontology Interchange Language (OIL), Defense Advanced Research Projects Agency (DARPA) Agent Markup Language (DAM) and Web Ontology Language (OWL).

Ontologies are composed of concepts, relations and instances, which according to [Grimm et al. \(2007\)](#), can be defined as:

- Concepts represent nodes in semantic networks, unary predicates in logic or concepts in description logics. They represent the ontological categories that are relevant in the domain of interest;
- Relations symbolize arcs in semantic networks, binary predicates in logic or roles in description logics. They semantically connect concepts, as well as instances, specifying their interrelations;
- Instances map to individual nodes in semantic networks, or to constants in logic. They represent the named and identifiable concrete objects in the domain of interest.

2.3.2.1 Classical Ontology Specification Languages

Ontolingua ([Farquhar et al. \(1997\)](#)) from the Stanford University is based on KIF (Knowledge Interchange Format) ([Genesereth and Fikes \(1994\)](#)) and Frame Ontology. It is the ontology building language used by the Ontolingua Server. KIF provides definition of objects, functions and relations and is based on the first-order predicate calculus, with a prefix notation. It allows the representation of meta-knowledge and non-monotonic reasoning rules.

OKBC ([Chaudhri et al. \(1998\)](#)), Open Knowledge Base Connectivity, specifies a protocol for accessing knowledge bases stored in frame knowledge representation systems and it is considered complementary to language specifications developed to support knowledge sharing. It supports network connections such as the direct access to knowledge databases.

OCML ([Domingue et al. \(1999\)](#)) stands for Operational Conceptual Modelling Language and was developed at the Knowledge Media Institute (United Kingdom). It provides mechanisms for expressing items such as relations, functions, rules (with backward and forward chaining), classes, instances and logical mechanisms for efficient reasoning. It is compatible with Ontolingua.

FLogic ([Kifer et al. \(1995\)](#)) or Frame Logic integrates frame-based languages and first-order predicate calculus. It can represent most of the structural aspects of object-oriented and frame-based languages.

LOOM ([MacGregor \(1991\)](#)) is a high-level programming language and environment intended for constructing expert systems. It supports a "description" language for modelling objects and relationships, and an "assertion" language for specifying constraints on concepts and relations, and to assert facts about individuals.

2.3.2.2 Web-Based Ontology Specification Languages

XML ([Bray et al. \(2008\)](#)), eXtensible Markup Language, derives from Standard General Markup Language (SGML). It envisions that it is easier making implementation and interoperability with both SGML and HTML. XML allows users to define their own tags and attributes, define data structures, extract data from documents and develop applications which test the structural validity of a XML document. XML itself has no special features for the specification of ontologies, as it just offers a simple but powerful way to specify a syntax for an ontology specification language.

RDF ([Lassila and Swick \(1999\)](#)), Resource Description Framework, has a strong relationship with XML and was developed by the W3C (World Wide Web Consortium) for the creation of metadata describing Web resources. One of the goals of RDF is to make it possible to specify semantics for data, based on XML in a standardized and interoperable manner. The RDF data model does not provide itself mechanisms for defining the relationships between properties (attributes) and resources, this is done by RDFS (RDF Schema) ([Brickley and Guha \(2004\)](#)). RDFS is a declarative language used for the definition of RDF schemas. It is based on some ideas from knowledge representation (semantic nets, frames and predicate logic).

XOL ([Karp et al. \(1999\)](#)) was developed by Pangea Systems Inc. and SRI International. XOL stands for XML-Based Ontology Exchange Language. It was designed to provide a format for exchanging ontology definitions among a set of interested parties by acting as an intermediate language for transferring ontologies among different database systems, ontology development tools or application programs.

SHOE ([Luke and Heflin \(2000\)](#)), Simple HTML Ontology Extension, was developed by the University of Maryland. SHOE is a HTML extension and its objective is to incorporate in Web documents semantic knowledge and provide specific tags for ontology representation.

OIL ([Fensel et al. \(2000\)](#)), Ontology Interchange Language, is a standard proposal for describing and exchanging ontologies. It has been designed to provide most of the modelling primitives commonly used in frame-based and description logic ontologies. It is compatible with RDF Schema and allows defining concepts, relations, functions and axioms.

DAML (Ouellet and Ogbuji (2002)) (DARPA Agent Markup Language) is an extension of XML and RDF languages. It consists in a simple language for expressing more sophisticated RDF class definitions than RDFS.

OWL (McGuinness and van Harmelen (2004)), Web Ontology Language, is recommended by W3C when information needs to be processed both by applications and people. This language can be used to explicitly represent concepts and their relationships. OWL is better in expressing concepts than XML, RDF or RDFS.

2.3.3 Petri Nets

Petri Nets (Petri (1966)), as highlighted by Bobbio (1990), are a very powerful tool for representing complex logical interactions between physical components or activities in a system. Their advantages include an intuitive graphical formalism allied to a strong mathematical background which allows simulating and demonstrating the system. PN have been used in the most varied fields which range from communication protocols, manufacturing systems, software development, chemical processes (Zurawski and Zhou (1994)) to sports (Perše et al. (2010)), (Bai et al. (2009)), (Marin et al. (2010)).

A classical PN (Petri (1966)) consists of three types of components: places (circles), transitions (rectangles) and arcs (arrows). Places represent possible states of the system, transitions are events or actions which cause the change of state and arcs connect a place to a transition and vice-versa.

The state of a PN is defined by its marking (marked PN), which is accomplished by additional components named tokens (black dots). The PN marking is defined by the set of tokens residing in the different places. Moreover the PN marking allows defining its possible transitions (future states). In fact, a transition can only be performed when all its input places contain at least one token. The action performed by a transition is to remove a token from each input place and add a token to each output place (see Figure 2.15).

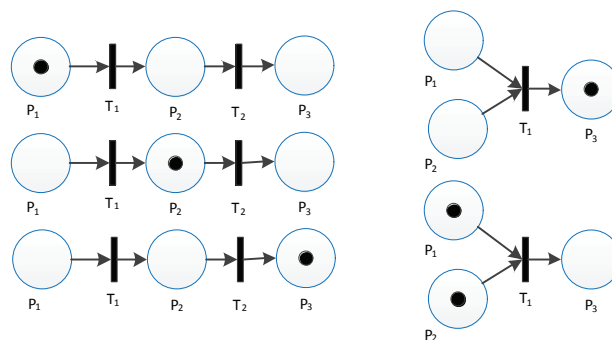


Figure 2.15: Example of two PN and their firing sequence.

Most systems cannot, however, be represented by classical PN, due to their complexity. Therefore, high level PNs have emerged along the years:

- Deterministic Timed Petri Nets (DTPNs) have a deterministic firing time or time interval associated with each transition, place or arc (Wang (1998));
- Stochastic Timed Petri Nets (STPNs) are similar to DTPNs, but each transition is associated with a random firing time (Wang (1998));
- Coloured Petri Nets (CPNs) (Jensen (1991)) include in each token a data value named token colour. These colours can be used and modified by occurring transitions;
- Hierarchical Coloured Petri Nets (HCPNs) (Jensen (1992)) allow to construct a large model by combining a number of small CP-nets into a large net by using substitution or place transitions. Along with the definition of HCPNs, Jensen (1992) have also launched the basis for a tool – CPN Tools – that allows for the implementation, simulation and analysis of Hierarchical Coloured Petri Nets.

2.3.4 CPN Tools

This subsection intends to explain how the CPN Tools software works, so that the model defined on chapter 4 (4.5) can be more easily understood.

CPN Tools is a software tool developed by the CPN group at Aarhus University in Denmark from 2000 until 2010. Since then it has been managed by the AIS Group at the Eindhoven University of Technology. The first software version was developed by Kurt Jensen as part of his PhD thesis and published in 1981 on Jensen (1981).

A typical image of CPN Tools is shown in Figure 2.16. The left side bar allows defining several colour sets and variables as will be seen below, while on the main screen tab it is possible to draw the HCPN with the respective transitions, arcs, places, among other instances.

The first step when defining a Petri Net on CPN Tools, consists in defining the colour sets and variables. A colour set (**colset**) defines the colour properties of a token and can be classified as simple or compound.

A simple colour set is only based on native colour sets, such as integers, strings, Booleans, unit, enumerations and index. The syntax declaration for simple colour sets is given below:

colset name = **type** [**with** user definitions];

, where the text between square brackets is optional. Using this notation it is possible to define the following colour sets:

- **colset** PlayerID = **int** with 0..100;
- **colset** Team = **with** Team|TOpp;
- **colset** Ball = **bool**;

A compound colour set uses previously defined colour sets either by combining them as a product of colour sets; a record, which is identical to the Cartesian product of the values; a list

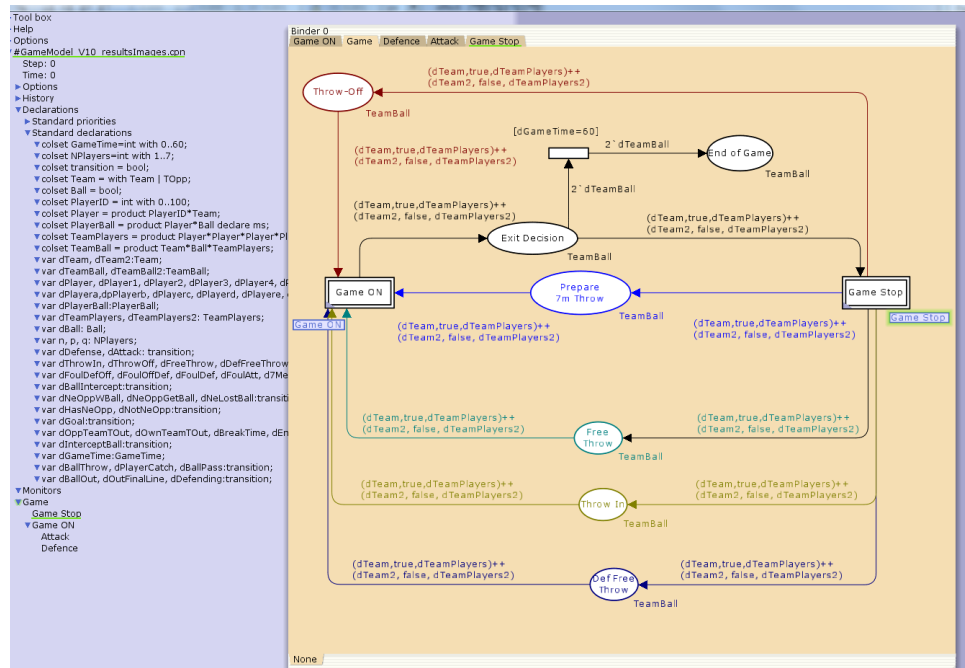


Figure 2.16: Example of the main screen of CPN Tools.

that allows to define a variable length colour set; an union to obtain a disjoint union of previously defined colour sets or a subset of colour sets, according to the following syntax:

- **colset** name = **product** name₁ * name₂ * ... * name_n;
- **colset** name = **record** id₁ : name₁ * id₂ : name₂ * ... * id_n : name_n;
- **colset** name = **list** name₀[**with** int – exp1...int – exp2];
- **colset** name = **union** id₁[: name₁] + id₂[: name₂] + ... + id_n[: name_n];
- **colset** name = **subset** name₀withsubset – list;

With this syntax the following colour sets can be defined:

- **colset** Player = **product** PlayerID*Team;
- **colset** TeamPlayers = **product** Player*Player*Player*Player*Player*Player*Player;
- **colset** TeamBall = **product** Team*Ball*TeamPlayers;

Once the several colour sets are defined, it is possible to define the variables that will be used throughout the HCPN. In CPN Tools a variable is an identifier whose value can change during the execution of the model. A variable is defined as illustrated below:

var id₁, id₂, ..., id_n : cs_{name};

,where cs_{name} is the name of an already defined colour set.

With the previous defined colour sets it is possible to define variables such as:

- **var** dTeam, dTeam2: Team;
- **var** dPlayer, dPlayer1, dPlayer2, dPlayer3, dPlayer4, dPlayer5, dPlayer6, dPlayer7, dPlayer8: Player;
- **var** dTeamPlayers, dTeamPlayers2: TeamPlayers;

For readability purposes, all variables are preceded by a "d" in order to indicate that they refer to data. This nomenclature allows to easily differentiating between a colour set and the respective variable.

Besides the declarations already defined, the CPN Tools software also provides instances for places, transitions and arcs as illustrated in Figure 2.17.



Figure 2.17: Instances provided by CPN Tool software.

Each place can have three types of inscriptions: place name, colour set and initial marking. The place name is not mandatory, but it is useful to better understand the network. The colour set inscription is very important because it determines the types of tokens that can be put in that place. Like the place name, the initial marking inscription is optional and defines the set of tokens that are assigned to the place. Figure 2.18, illustrates one place with the three available inscriptions.

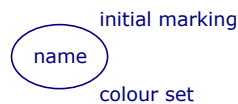


Figure 2.18: Inscriptions available for places on CPN Tool software: place name, colour set and initial marking.

Transitions can have up to five optional inscriptions as depicted in Figure 2.19: transition name, guard, time, priority and code segment. Like for places, it is not mandatory for a transition to have a name, however it is recommended for readability reasons.

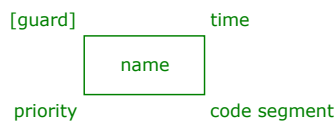


Figure 2.19: Inscriptions available for transitions on CPN Tool software: transition name, guard, time, code segment and priority.

The guard inscription is used for tests on the input arc inscription variables, which allows enabling transitions with restrictions.

Time inscriptions allow to delay the transition activation by the amount defined on the inscription, the time delay is preceded by the @+ expression. Time inscriptions can be formulated also

as an expression that depends on the token's value. Transitions can also have different priorities in order to give priority to a transition over another and code segments which are executed when the transition is fired. A transition code segment is composed of an input, an output and an action field.

Arcs can only have one inscription that is the arc inscription. An arc inscription can evaluate a multiset or a single element, and the arc expression must match the colour set of the destination place in case of an output arc, or the source place in case of an input arc.

When using arc inscriptions, it means that a transition is only enabled if there is a binding so that each input arc expression evaluates to one or more colours that are present on the corresponding input place, in other words a transition is only enabled if there are tokens matching the values of the arcs inscriptions and the guard of the transition evaluates to true.

Expressions on arcs can be as simple as a single true or false expression defining a single element (Figure 2.20(a) describes a single element expression based on a Boolean colour set). However, more complex expressions must use multisets as the ones illustrated in Figures 2.20(b) and 2.20(c).

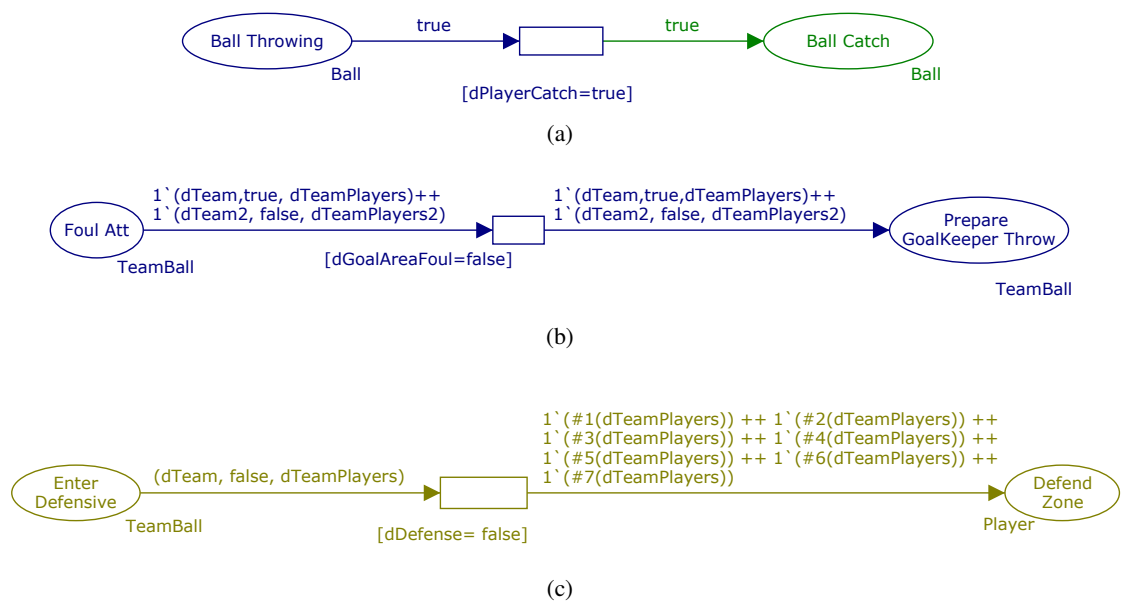


Figure 2.20: CPN Tool arc inscriptions: (a) single element Boolean arc inscription (b) multiset arc inscription with two elements (c) multiset arc inscription with seven elements.

In order to define multiset arc expressions it is necessary to use the multiset constructor that is the back-quote operator (```). This operator allows defining how many instances of a given colour set are necessary to perform the binding.

Using Figure 2.20(b) as an example, it is possible to verify that for the arc expression to be valid there must be an instance of $(dTeam, true, dTeamPlayers)$ and another of $(dTeam2, false, dTeamPlayers2)$. These two instances are of colour set TeamBall which is the product of a Team

(dTeam or dTeam2), a Ball (true or false) and a set of players (dTeamPlayers and dTeamPlayers2 are variables of colour set TeamPlayers).

Figure 2.20(c) illustrates an example which has access to the components of a compound colour set. By using the operator #, each player of the dTeamPlayer variable can be accessed. This transition also illustrates how a transition between two places with different colour sets can be triggered.

When a transition takes place, tokens from the input arc (represented by the green circle and the green comment) are transferred from the input place to the output place as illustrated in Figure 2.21. This transition is only enabled because both the arc inscription ((dTeam, false, dTeamPlayers)), as well as the transition guard ([dDefence=false]) evaluate to true.

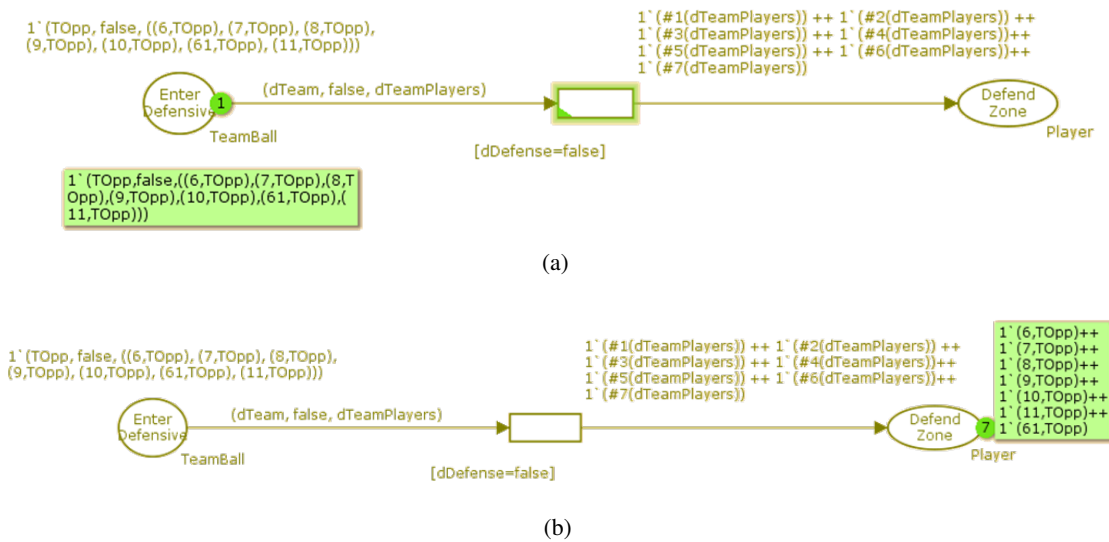


Figure 2.21: CPN Tool transition firing: (a) Petri net state prior to the transition takes place and (b) Petri net state after to the transition takes place.

Has already been referred, large networks can become too complex and hard to read, therefore CPN Tools provides mechanisms to simplify a network by allowing to define multiple layers by means of substitution transitions.

Substitution transitions represent an entire piece of a portion of a net structure. Therefore, a substitution transition does not represent a transition, but an entire network with other transitions and places, connected via arcs.

The net represented by the substitution is stored in what is called a subpage, additionally the net on the subpage is called a subnet or a sub-model.

Figure 2.22, illustrates an example of a superpage with a substitution transition (2.22(a)) and the respective subpage (Figure 2.22(b)) with the corresponding subnet.

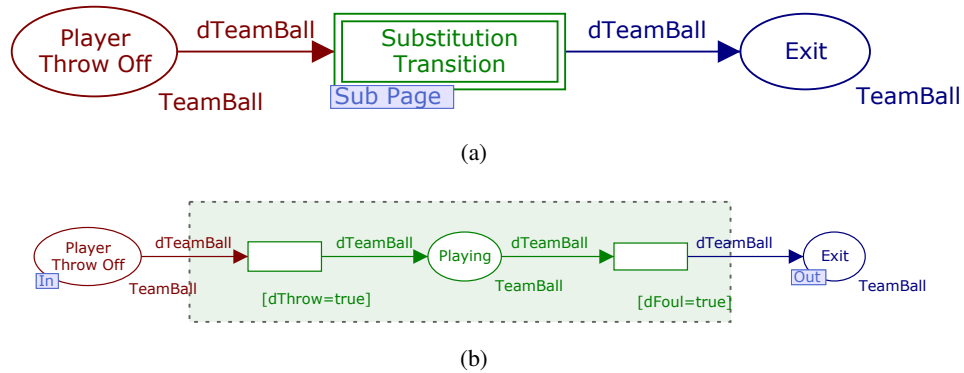


Figure 2.22: CPN Tool substitution transition: (a) substitution transition superpage (b) substitution transition subpage.

Places on the superpage that have a connection with the substitution transition are called sockets (places **Player Throw Off** and **Exit** in Figure 2.22(a)). These sockets have a corresponding counterpart on the subnet, which are called ports. Each port has an associated tag-type, which indicates if the port is an input, an output or both an input and output.

Figure 2.22(b), shows two ports, one that is the input for the subnet, *Player Throw Off*, and another that is the output of the subnet, *Exit*.

This subsection only described the concepts that are needed for Chapters 4 and 5, for a more exhaustive description of the tool please refer to CPN Tools webpage (Tools (2015)) or to Jensen and Kristensen (2009).

2.4 Summary and Conclusions

This chapter presented some of the most important methodologies and fundamentals used for video segmentation, image recognition and knowledge representation.

Video segmentation is divided into two main groups, one based on the traditional detection methods used by image segmentation, spatial video segmentation, and another that is more focused on object tracking. The main difference between these two groups is related with the fact that the first explores all the possibilities existing in the image/frame, while the latter uses prior knowledge to predict the object next location, restricting the area to be searched.

Due to its long existing history, image segmentation techniques have a wide spectrum of methods, ranging from thresholding until motion based methods. On the other hand, object tracking techniques are more recent and so, the variety of methods is less abundant.

The presented image recognition methods are subdivided into three main categories: Bayesian, Non-Bayesian classifiers and the combination of different classifiers (multiple classifiers). Image recognition is an evolving field of investigation, since most image recognition methods can only provide good results on specific situations. Nevertheless, studies point that combining multiple classifiers seems to be the right direction to be followed.

This chapter also introduced the fundamentals of knowledge representation as well as several modelling languages and formalisms. A more detailed section was devoted to CPN Tools, as it will be widely explored on Section Game Mode ([4.5](#)).

The scope of this chapter was more focused on the concepts that will be used throughout this document.

Chapter 3

Related Research

This chapter presents an overview on what has been done on the areas of player detection and tracking, ball tracking and high level game analysis.

For several years, sports experts have performed hand annotation of games in order to collect metrics and game statistics ([Sampaio et al. \(2006\)](#) and [Spencer et al. \(2005\)](#)). These techniques usually require a huge time effort to collect the raw data, and generate enormous amounts of information that are subjective to the operator and are only moderately reliable ([Duthie et al. \(2003\)](#) and [O'Donoghue \(2010\)](#)). The use of technology as a way of minimising the aforementioned problems ([Franks et al. \(1987\)](#)) was proposed in the late 1980s. Initially, it consisted of a support to hand annotations where it would be possible to store information in databases, perform statistics and record matches so that they could later be analysed by experts. Furthermore, recent advances have led to systems that are able to perform Motion Analysis in an automated and time efficient way, which means that all the positions of the game elements are extracted automatically, with added benefits such as high repeatability and accuracy.

Motion Analysis in sports consists of knowing the location of all relevant elements in a sports game. This is a complex task due to concerns with human safety and the fact that there are many elements to track, often at high speeds and with frequent changes of direction. An additional problem, is the presence of the ball, a somewhat small shared object that both teams compete to manipulate in accordance to the specific game rules.

Two main technology categories are used for automatic Motion Analysis: (i) Intrusive Systems, where special tags or sensors are placed on the targets, and (ii) Non-Intrusive System, where there are no extra objects inside the game, only on the surrounding environment.

Intrusive Systems use well known, stable, theoretical concepts that have already been widely addressed, and most limitations are concerned with technological issues, namely with noise and interference with signals (accuracy and dependability), battery life time and transmitter/receiver size.

For Non-Intrusive Systems, vision is the interesting approach, which is developing quickly by delivering better images at higher rates and at affordable prices. The processing of such massive amounts of information still presents a significant challenge due to the high dynamics in sports.

Due to the differences inherent to each of the two game elements (players and ball), namely in terms of size and physical behaviour and to the necessity of applying either different image processing techniques or different hardware devices, authors usually deal with them separately. Therefore it seems appropriate to separate them into the following sections.

3.1 Player Detection and Tracking

3.1.1 Intrusive Systems

Intrusive systems make use of wireless sensors or tags. This property makes them sensitive to degradation and interferences either by other objects or by Radio Frequency (RF) signal collisions, among other detection problems. Additionally, and since RF signals are electromagnetic waves, they lose signal quality when crossing different materials and suffer additional problems with metals (reflected waves) and water (absorbed waves), as a consequence of basic physics – Maxwell Laws ([Alonso and Finn \(1992\)](#)). Such issues can be mitigated if the technology is carefully implemented. However, there will always be limitations inherent to the basic principles of the signals.

RF energy is regulated by legislation due to health concerns and interference between RF devices. Localisation systems typically use the so-called industrial, scientific and medical frequency part of the spectrum that is shared among many applications, such as Wi-Fi networks or Bluetooth devices ([Committee \(2014\)](#) and [NTIA \(2011\)](#)).

Several technologies are available: Global Positioning Systems (GPS), Radio Frequency Identifiers (RFID), Ultra Wide Band (UWB), among others.

3.1.1.1 Detection

GPS Systems It is generally accepted that pure Global Positioning Systems (GPS) cannot be used indoor due to signal degradation problems which occur when crossing walls and ceilings. Therefore their main usage is devoted to outdoor sports such as rugby, soccer and football.

The position of a GPS receiver is calculated by measuring its distance to several GPS satellites. This distance is obtained using trilateration ([Wikipedia \(2011\)](#)), and so GPSs are able to determine the player's position, speed and elevation.

Despite only being able to work outdoor, GPS systems represent a significant slice of the available intrusive commercial systems: Minimax ([Catapult Innovations \(2009\)](#)), SPI Pro HPU ([GPSports, 2015](#)) and VIS Track – GPS ([IMPIRE AG \(2015\)](#)).

Minimax uses a small tag (19x50x88mm and 67g), that besides providing the player position may, depending on the model, also include accelerometers, gyroscopes, magnetometers and a heart rate detector. Models may provide measures at 5Hz or 10Hz frequencies. The manufacturer

indicates errors of 40cm in 10m and 40m sprints while independent studies ([Castellano et al. \(2011\)](#)) indicate that errors on distance covered can be as high as 3.9m (with a deviation of 1.1m) for 15 and 30 meter distance. This manufacturer provides a dedicated software so that the end users can analyse the obtained information (information available includes distance, velocity, real time data and alarms, accelerations/decelerations, heart rate zones, tactical animations, among others). This system is used by several soccer (Liverpool, Everton, and Blackburn Rovers) and rugby (Canterbury Crusaders, Brisbane Broncos, Melbourne Storm) teams. Scientific studies have also been conducted on Australian football ([Farrow et al. \(2008\)](#)) and field hockey ([Gabbett \(2010\)](#)), among others.

SPI Pro X works at 15Hz and includes accelerometers and heart rate measurement devices. Each tag measures 48x20x87mm and weights 76g. It has been used by several European soccer clubs (Barcelona, Real Madrid, Chelsea, Liverpool), rugby (Wallabies, English Rugb, West Tigers) and Australian football (Geelong, Brisbane). Although accuracy studies do not include the 15Hz model, older models of 1Hz and 5Hz showed accuracies within 1% and 5% for straight-line runs of 50m, respectively ([Gastin and Williams \(2010\)](#)). Studies using this system have been performed in the Australian football league ([Wisbey et al. \(2010\)](#)) and rugby ([Venter et al. \(2011\)](#)).

VIS Track-GPS is the lightest system with 40g of weight and a size of 55x45mm. It operates at 10Hz and provides a software for visualizing the raw data and performing data analysis. It has been used by soccer teams (Borussia Dortmund, Sandhurst FC).

Using the same principles as the GPS, but for indoor environments, it is possible to find Pseudolite (pseudo-satellites) based systems ([Wang \(2002\)](#)). A European project ([European Space Agency \(2013\)](#); [Väkevä et al. \(2004\)](#)) developed and tested a Pseudolite based system composed of static devices (Pseudolites, reference receivers and master control unit) that generate the navigation signals and mobile receivers that are capable of calculating their own positions. Even though no tests have been made on real sports activities, the results for indoor environments (which include a sports hall) show promising results (distance root mean square as low as 2 cm).

Tag plus Antennae Systems RF localisation systems are based on nodes/tags placed on the objects being tracked, which broadcast a signal that is picked up by a set of fixed nodes placed at known points ([Sathyan et al. \(2011\)](#)), or antennae placed around the field ([InMotio \(2015\)](#); [ISS \(2015a\)](#); [Ubisense \(2012\)](#); [Wadell et al.](#); [ZXY \(2012\)](#)).

The tag's position is not determined by the tag itself (contrary to GPS based systems), but the information collected by the antennae is combined, depending on the system, using the signal Time of Arrival (TOA), the Angle of Arrival (AOA) or Local Positioning Measurement technology ([Stelzer et al. \(2004\)](#)) to determine its 3D position. If a tag is capable of "self-locating", which means that localisation data is available "internally" in the tag of the tracked object, then, most likely, there will be the need to record or transmit the data for latter manipulation. External localisation (the environment has the positions of the tags to track) will probably make the tags cheaper and lighter (but such tags are still battery operated).

Due to the nature of electromagnetic waves, the systems' behaviour may be different depending on the environment conditions. However, only [Sathyan et al. \(2011\)](#) highlight these concerns by providing accuracies both for indoor (50cm) and outdoor (15cm) environments. The smaller errors in outdoor environments can be explained by the inexistence of structures that interfere with the signals, although rain or fog will also probably cause problems.

Despite its small use on the sports environment it is worth mentioning the Ubisense system, which is a generic localization system that uses UWB frequencies and is able to detect the 3D position of objects or persons within an accuracy of 15cm. This system has been used under the Tennissense project ([Conaire et al. \(2009\)](#)).

Table 3.1 summarises the systems presented. It is important to highlight that most of these systems are commercial and therefore the information provided about the system itself is based on the manufacturer statements that most of the times does not include detailed information or scientific studies about validity, objectivity or reliability, an issue already raised by [Carling et al. \(2008\)](#).

Table 3.1: Summary of systems based on intrusive devices.

System	Oper. Freq. (Hz)	Strategy	Update Frequency (Hz)	Accuracy	Localization	Comm.
Minimax (Catapult Sports)	– ¹	Trilateration	2, 5, 10 (depending on the model)	0.40m (10Hz) ² 0.1m to 3.9m (10Hz) ³	Ext.	✓
SPI HPU ((GPSports, 2015))	15	Trilateration	15	1	Ext.	✓
VIS Track - GPS (IMPIRE AG (2015))	– ¹	Trilateration	10	1	Ext.	✓
Pseudolites (European Space Agency (2013))	1.58	Triangulation	1-5	2cm	Self	✓
RedFir (ISS (2015a), Nandan (2013))	2.4	TOA and AOA	200 (player) 2000 (ball)	"Few cm" [sic]	Ext.	✓
Trakus (Wadell et al.)	2.4	TOA	30	– ¹	Ext.	✓
Wasp (Sathyan et al. (2011))	5.8	TOA	10	50 ⁴ 15 ⁵	Ext.	×
InMotio (InMotio (2015))	5.8	LPM	1000	±3cm	Ext.	✓
Ubisense (Ubisense (2012))	5.8 to 7.2 ⁶	TOA and AOA	160 ⁷	15	Ext.	✓
ZXY (ZXY (2012))	2.45 -5.2	Radio wave analysis	40	Centimetre [sic]	Ext.	✓

¹information not available²Catapult Innovations (2009)³Castellano et al. (2011)⁴indoor⁵outdoor⁶EU licensed operating frequencies⁷depending on firmware version

3.1.1.2 Tracking and Filtering

With Intrusive Systems, tracking is likely to be straightforward since each tag is always tracked independently. Filtering methodologies such as the ones mentioned in section 3.1.2.2 will further improve the measurements' accuracy by reducing interferences and limiting effects of temporarily unavailable data. However, data quality is generally good and simple formulations provide adequate tracking.

3.1.2 Non-Intrusive Systems

One of the major problems of the aforementioned intrusive systems is that usually regulations do not allow their usage on official games since they may interfere with the game, hence they tend to be used only during training sessions or non-official games, therefore several works have been developed using non-intrusive systems.

The most interesting non-intrusive systems are based on vision methodologies which require advanced image processing methodologies in order to find and individualise players due to the high contact and speeds involved in invasion sports. Table 3.2 provides an overview of the presented methodologies and systems.

3.1.2.1 Detection

Two main sources for acquiring video streams are used: (i) television broadcast cameras and (ii) dedicated cameras.

Besides the systems described in scientific literature it is possible to find systems that are completely commercial (Barris and Button (2008)), mainly on the soccer domain, such as Pro-Zone/Amisco (Prozone (2015)) or eAnalyse Pro (Espor (2015)), however they require great user intervention to correct the players' tracking and identify important events (Setterwall (2003)) and usually the information provided by the manufacturer is scarce. Moreover, no similar solutions seem to exist for indoor sports.

Broadcast Vision Systems Television broadcast systems are usually based on a single broadcast camera focused on restricted areas where the most important events take place (generally the goal area). Nevertheless, they are a good source of information because they are common and the resulting streams can be used for multiple purposes.

Kasiri-Bidhendi and Safabakhsh (2009) use TV footage to track the players and the ball in indoor soccer games, however they provide qualitative results (image coordinates).

Beetz et al. (2009); Hayet et al. (2005); Pallavi et al. (2008a); Tong et al. (2011); Zhu et al. (2009) take advantage of television footages on the soccer domain. Frequently authors use one single camera, however this approach has the negative aspect of not providing an entire view of the playing area and only gives useful information on long shot images, therefore, Beetz et al. (2009); Hayet et al. (2005) use multiple broadcast camera systems.

Table 3.2: Summary of NIS features ([D] – detection, [T] – tracking, P/R – precision/recall, [M] – measurement accuracy, * - values calculated based on the information provided by the authors).

Sports	System	World Coord.	Ball	Results			Vision System Configuration	Dataset size
				D[P/R]	T[P/R]	M		
Indoor Soccer	Kasiri-Bidhendi and Safabakhsh (2009)	×	✓	Qualitative Results	-	-	Broadcast 720x480 pixels	Not mentioned
	Needham and Boyle (2001)	×	×	-	-56%	1.16m	Dedicated 320x240 pixels	[T]:835 frames
	Hu et al. (2011)	✓	×	91%/91%	-	-	Broadcast 720x480 pixels 29.97fps	[T]:390s
Basketball	Lu et al. (2011)	×	×	96%/67%	98%/81%	-	Broadcast	[DT]:21 clips of 24.1s each
	Monier et al. (2009)	×	×	99%*	-	-	2 dedicated 1932x1040 pixels 30fps	[T]:11 clips from 12.16-25.43 min
	Alahi et al. (2009)	✓	×	72%/77%	-	-	Apidis dataset	Not mentioned
	Chen et al. (2011); Delannay et al. (2009)	×	✓	-90%	-	-	Apidis dataset	[D]:180 frames
	Chen et al. (2012)	✓	×	-	89%/89%	91%	Broadcast 640x352 pixels 29.97fps	40 test clips [sic]
Handball, Basketball	Kristian et al. (2009); Perše et al. (2009)	✓	×	99%* ⁹	0.3-0.5m ¹⁰	-	2 dedicated 384x288pixels 25fps	[T]:12.2s-50.28s

Continued on Next Page...

Table 3.2 – Continued

Sports	System	World Coord.	Ball	Results			Vision System Configuration	Dataset size
				D[P/R]	T[P/R]	M		
Handball	Barros et al. (2011)	✓	×	-82%	-84%	0.07-0.28m	2 dedicated 720x480pixels 30fps	[D]:30 min [T]:3 min
Ice Hockey	Okuma et al. (2004)	×	×	Qualitative results		-	Broadcast	205 frames
	Lu et al. (2009)	✓	×	Qualitative results		0.1413 ¹¹	Broadcast 320x240 pixels	1314 frames
	Tong et al. (2011)	✓	✓ ¹²	90%/90%	98%/93%	Qual.	Broadcast 720x576 25fps	[D]:1133-5927 [T]:100-250 frames
	Pallavi et al. (2008a)	×	×	91%/96.5%	98%/93%	-	Broadcast	[T]:60–317 frames
Soccer	Beetz et al. (2009)	✓	✓			0.05-1m	Broadcast	
	Hayet et al. (2005)	✓	×	Qualitative results		-	4 Broadcast	[T]:400 frames
	Iwase and Saito (2003)	✓	×	Qualitative results		-	8 Dedicated cameras 720x480 15fps	450 and 150 frames
	Figuerola et al. (2006)	✓	×	-	-94%	-	4 dedicated static cameras 720x480 pixels	4500 frames
	Zhu et al. (2009)	×	✓	-	83%/88%	-	Broadcast Camera 704x506 pixels	64 matches

⁹(Kristan et al. (2009))
¹⁰(Perše et al. (2009))
¹¹fraction of width of the ground truth box
¹²(Tong et al. (2007))

The usage of television footages makes the homography computation more difficult, because the cameras are constantly moving and there may not be reference points on the image that allow its computation. To deal with this, [Tong et al. \(2011\)](#) use SIFT and RANSAC methods to estimate the camera motion and deduce the new homography matrix values. The player detection precision ranges from 88.65% to 92.38% while the tracking can reach 98.47%. The homography matrix allows converting players' positions from image coordinates into either a common reference or into real world coordinates.

In order to only deal with meaningful frames, [Pallavi et al. \(2008a\)](#) first classify frames based on colour features to detect long shot images (wide field of view). The results obtained on short video sequences show high detection rates (average of 93%), however in some occlusion situations the detection rate can drop down to 47%.

[Hayet et al. \(2005\)](#) use several broadcast cameras with rotating and zooming characteristics, and so are much concerned with grouping all the data into a common reference frame, therefore they continuously keep an updated homography matrix for each video stream in order to map the points from the original frames into the reference frame. Although the results presented are qualitative and for short periods of time, they show that their methodology is effective even under severe occlusion of players from the same team.

One of the most interesting systems, Aspogamo, is presented by [Beetz et al. \(2007, 2009\)](#). This system is able to analyse sports games using ontology models of the game with players' positions, motion trajectories and ball actions as primitives.

Another quite complete work is the one from [Zhu et al. \(2009\)](#). They are able to provide information about ball trajectory, offensive players' and defensive players' trajectories and other higher level information as will be discussed on Section 3.3. Player detection precision is around 83%.

[Chen et al. \(2012\)](#); [Hu et al. \(2011\)](#); [Lu et al. \(2011\)](#) use TV basketball footage. [Hu et al. \(2011\)](#) work provides a reliable mapping between pixel and world coordinates (accuracy around 97%) and the players' tracking is performed with precision and recall of around 91% (most problems are related with players of the same team merging with each other and interference from the audience). Additionally, and using the extracted players' trajectories, they formalise high level concepts such as wide-open players and one-to-one or zone defence. [Lu et al. \(2011\)](#) are also capable of tracking the players with 98% precision and 81% recall. By using a Logistic Regression classifier that combines maximum stable extremal regions (MSER), scale invariant features transform (SIFT) and interest points on colour histograms, they can uniquely identify each player (accuracy: 82-85%), which is important in games where players can be replaced multiple times. Finally, [Chen et al. \(2012\)](#) are able to determine the real world coordinates based on extracting the court lines and determining the camera homography matrix with 91.47% of frames being well calibrated. Players are tracked with average precision and recall rates of 89.71% and 89.20%, respectively. The players' trajectories are mapped into a real world court model for screen pattern recognition.

[Okuma et al. \(2004\)](#) applied a boosted particle filter for tracking and a cascaded AdaBoost algorithm to learn hockey player characteristics. Results are presented as bounding boxes on images

and therefore quantitative values are not available. In a more recent work [Lu et al. \(2009\)](#), have enhanced the detection and tracking algorithms and included players' action recognition (skating up, down, right and left) using a Sparse Multimodal Logistic Regression Classifier. Actions were recognized with an accuracy of 76.37%, and tracking with an average centre error distance of 0.1413 (this value does not have unit since it corresponds to the fraction of width of the ground truth box).

Dedicated Cameras Dedicated camera systems place the cameras at strategic locations of the sports halls, which allows a better and complete view of the field. Additionally, they make it easier to have multi-camera systems. Such systems usually provide higher resolution (better accuracy) comparatively to the previous ones and overlapped image regions that minimise occlusion and merging problems. At the same time, using stereo vision it is possible to perform 3D localisation. However, their setup (and calibration) can be somewhat complex.

There are various system configurations regarding the number and camera location: [Needham and Boyle \(2001\)](#) use a single camera placed at the bottom line of an indoor soccer field. Their results are compared with hand annotated values and the associated error is quite high (around 1 meter).

With multiple-camera systems this error can be reduced, so [Barros et al. \(2011\)](#) use two cameras placed along the side line of a handball field with a high overlapped region, which reduces the measurement errors down to less than 28 cm. Overlapped regions also enable high detection (82%) and tracking (84%) rates because occlusion usually occurs only on one camera.

[Kristan et al. \(2009\)](#) developed the Sagit, which uses two fixed cameras placed at the top of the sports hall to provide a "bird's eye" view. This configuration makes it possible to reduce occlusion and merging problems. The authors show that failure rates per player and per minute were quite low - less than 1.1 in the worst test case - and positional accuracy values were estimated between 10 to 30 cm in squash games ([Vučković et al. \(2010\)](#)). Tests were performed both in handball as well as in basketball matches.

[Monier et al. \(2009\)](#) also use dedicated ceiling mounted cameras to analyse basketball matches. According to their results, a human operator needs to correct tracking data approximately 6.7 times per each player in each 1000 frames.

[Figueroa et al. \(2006\)](#) use four cameras placed on the side lines which allows overlapped regions and are able to compute the world coordinates of the players. The percentage of automatically tracked frames in case of no occlusions achieved 78.5% and 15.5% in case of occlusions.

In order to further minimize occlusion effects [Iwase and Saito \(2003\)](#) propose a dedicated 8 camera system. Each camera is treated as an independent system and, whenever a player is not detected, either due to occlusions, not detections or by being outside the angle of view a multi camera process is activated and, in most of the cases, is able to correctly identify the players. The geometrical relationships between the cameras are calculated based on planar homography. Their system has a drawback since it only covers the penalty area and therefore they are not able to give a good insight of the entire game.

[Alahi et al. \(2009\)](#) and [Delannay et al. \(2009\)](#) use Apidis (Autonomous Production of Images Based on Distributed and Intelligent Sensing [Apidis \(2008\)](#)). This dataset results from a project in which a basketball court was equipped with a network of cameras and microphones. An example of the footages recorded can be seen in Figure 3.1.



Figure 3.1: Apidis typical images in [Apidis \(2009\)](#).

Taking advantage of the setup, flexibility and cameras, [Alahi et al. \(2009\)](#) were able to minimise occlusion and merging problems. Moreover, they are able to determine the players' 3D positions in world coordinates. Results show that using more than one camera can tremendously increase both the precision and recall rates for the player detection, from 62% and 57% with a single camera system to 72% and 76% simply by adding an omnidirectional camera.

[Delannay et al. \(2009\)](#) were able to detect not only the players (accuracy above 90%), but also to identify their numbers, with a recognition accuracy of 73%. More recently, [Chen et al. \(2011\)](#) included the ball tracking (80% detection rate) and semantic analysis in this work.

Image Processing Techniques The typical image processing system starts by distinguishing background from foreground, so that foreground features (game elements) can be detected. Several methodologies have been applied, ranging from clustering where pixels are grouped together based on similar features such as colours, to dynamic methods that adapt to illumination changes. In these dynamic methods, each pixel of the empty field (the background) is modelled as Gaussian variables that vary over time according to pixel colour change, a technique known as Mixture of Gaussians, MOG ([Bouwman et al. \(2008\)](#)).

The usage of adaptive methodologies is important because most sports halls have windows. However, they are characterised by a learning constant that must be well tuned since it is responsible for determining how fast it is able to adapt to illumination changes, to incorporate objects which are moved, inserted or that were moving but have stopped in the background ([Bouwman et al. \(2008\)](#)).

Results from foreground/background detection must then be grouped into regions known as blobs. Blobs may be further filtered using: physical clues, colour clustering, boosting classifiers that use a series of weak classifiers to produce a stronger classifier, regions colour histograms

(team colour, total area) or template matching (similar to comparing candidate players against a given image template).

Alahi et al. (2009); Delannay et al. (2009) refine the foreground objects based on occupancy maps generated from the multiple camera views. Both authors prove that multiple cameras greatly increase the system's precision, from 62% to 72% by adding an extra omnidirectional camera (Alahi et al. (2009)).

Lu et al. (2011) besides detecting the players, also recognise them using Deformable Part Models (that use a set of small known images for body parts – templates) and a Logistic Regression Classifier (with MSER, SIFT and colour histograms).

Template matching and classifiers require training, which means that prior to processing the game, a set of representative image samples must be collected and classified by a human operator. Techniques based on background subtraction usually speed up the processing time, since only a few regions need to be analysed with more complex methodologies.

3.1.2.2 Tracking and Filtering

After detection, targets must be tracked over time in order to produce usable trajectories. This is especially challenging because Vision systems identify the presence of an "object", but generally cannot identify the target(s) that may be merged together in that detected "object".

Methodologies such as Kalman Filtering (KF) (Kalman (1960)), Particle Filters (PF) (Gordon et al. (1993)) or Markov Chain Monte Carlo based methodologies are often chosen because the several existing variants are capable of dealing with non-linear movement, missing and noisy measurements. KF is theoretically optimal and corresponds to a very efficient filter where all variables are Gaussian probabilistic functions. The filter estimates state variables (such as positions) based on movement equations and noisy measurements models. The PF samples, enable combinations to find the most likely one (using Bayesian probability models).

Additional filtering techniques include CamShift (Bradski (1998)), weighted graphs, the Munkres (or Hungarian) (Munkres (1957)) algorithm, level set contours or more naïve methodologies based on velocity constraints. CamShift is based on the Mean Shift algorithm (Comaniciu and Meer (1997)), which is capable of determining the change in the player's centre of mass over time. Weighted graphs are composed of nodes with spatial information (size, shape and colour) and edges with time information. The blob assignment is performed using minimal path searching. The Munkres algorithm assigns each blob to a given player, minimising the assignment cost and level set contours follow the players' boundaries instead of the blob itself (for further information on this subject, please refer to section 2.1.2).

The filtering and tracking methodologies play an important role when dealing with occlusion and merging situations. Therefore, CamShift, for example, cannot perform well when players from the same team merge together, even if for short periods of time (Hu et al. (2011)). Probabilistic models are appropriate not only because the players' movements on the field are non-linear, but also because they can handle occlusion and merging situations, as long as they are not too persistent (Yilmaz et al. (2006)).

3.1.3 Conclusions

Intrusive Systems major problems are the usual interdiction in official matches, mounting and calibration requirements, and attaching tags to targets, which likely restricts data collection to home players at home matches. Drawbacks also include the systems' accuracy dependence on environmental conditions (for instance, metal parts, water vapour and human bodies can cause problems with RF signals).

Their strengths include 3D localisation, high localisation rates (2000 per second), accuracies as low as ± 5 cm in large spaces (InMotio (2015)) and the ball tracking possibility (ISS (2015a), Nandan (2013)).

Although GPS systems are quite common on rugby, soccer and Australian football, they are not suitable for indoor sports due to the poor reception of GPS transmitters inside buildings. The other two technologies, tag plus antennae and Pseudolites, can be used on indoor environments, however only InMotio (2015) seems to have been used in a real handball sports match (Hökelmann et al. (2008)) and the details available are scarce.

On non-intrusive systems, the raw data is provided in image coordinates which may not be useful for performing more high level analysis or even extract statistical indicators. Therefore, systems must compute the "homography", so that they can translate image positions into world coordinates, however not all systems have this concern as demonstrated previously on Table 3.2. Additional features include individually recognition of each player (Lu et al. (2011)) or shirt number identification (Delannay et al. (2009)). The ball position also plays an important role in game analysis, but only Beetz et al. (2009); Chen et al. (2011); Kasiri-Bidhendi and Safabakhsh (2009); Tong et al. (2011); Zhu et al. (2009) address this problem.

Most systems can deal with temporary data loss. However, user intervention is sometimes needed to complete partial information or tracking mistakes. Tables 3.2 (shown previously) and 3.3 provide an overview of the methodologies and technologies used by each author. There, it is possible to confirm that non-intrusive systems achieve detection rates ranging from 56% to 99%, while measurement accuracies range from 0.05 m to 1.6 m when compared with ground truth positions.

Table 3.3: Summary of methodologies used on non-intrusive systems.

Sports	System	Filtering/Pre-processing	Detection Method	Tracking Method
Indoor Soccer	Kasiri-Bidhendi and Safabakhsh (2009)	Background model using colour clustering	Area and roundness constraints applied to the resulting areas	Level set contour
	Needham and Boyle (2001)	Static background/foreground segmentation using HIS colour space clustering	Bounding box fitting	Multiple object condensation algorithm (family of Particle Filters) with a Kalman filter predictive stage
Basketball	Hu et al. (2011)	Background MOG modelling Camera parameters	K-means colour clustering	CamShift
	Lu et al. (2011)	Classifier training	Deformable Part Models Logistic Regression Classifier	Kalman Filter
	Monier et al. (2009)	Dynamic background model	Template matching	Fixed ROI
	Alahi et al. (2009)	Basic background subtraction Camera parameters: homography	Foreground point projection to ground occupancy map	The maximum speed of a player defines a disk of future areas
	Chen et al. (2011); Delannay et al. (2009)	Background modelling using GMM	Foreground point projection to occupancy maps	Munkres algorithm
	Chen et al. (2012)	Background modelling using GMM Camera parameters: homography	Colour K-means clustering of foreground regions	Kalman Filter
Handball, Basketball	Kristan et al. (2009); Perše et al. (2009)	Dynamic background model	Colour histogram model and image mask filtering	Particle filter and world partition into Voronoi cells
Handball	Barros et al. (2011)	Adaptive background segmentation Camera parameters: perspective projection matrix	AdaBoost classifier Numerical descriptors obtained from integral images	Minimal path search on a graph using spatial and temporal information

Continued on Next Page...

Table 3.3 – Continued

Sports	System	Filtering/Pre-processing	Detection Method	Tracking Method
Ice Hockey	Okuma et al. (2004)	Cascaded AdaBoost classifier	Cascaded AdaBoost classifier 1D colour histograms	Boosted Particle Filter
	Lu et al. (2009)	Cascaded AdaBoost classifier and Sparse Multinomial Logistic Regression Classifier	Cascaded AdaBoost classifier 1D and 2D colour histograms Histograms of Oriented Gradient (for shape)	Boosted Particle Filter
	Tong et al. (2011)	Colour segmentation, shape and size constraints.	Boosted cascaded of Haar Features detector.	Markov Chain Monte Carlo data association
Soccer	Pallavi et al. (2008a)	Shot dominant colour ratio Removal of non-player regions	Segmentation based on area threshold	Directed weighted graph
	Beetz et al. (2009)	Cam. parameters: model based localization Blobs through colour segmentation	Segmentation based on players' colour template matching	Rao-Blackwellized Resampling Particle Filter
	Hayet et al. (2005)	Camera parameters: homography	Local descriptors characterized by interest points	Point Distribution Models and Kalman filter for camera merging
	Iwase and Saito (2003)	Background subtraction, noise removal and smoothing Camera parameters: homography	Areas resulting from the pre-processing process	Moving distance, area and colour of the extracted player Multi-camera information fusion
	Figuerola et al. (2006)	Adaptive background segmentation Camera parameters: perspective projection matrix	Image binarization and morphological filtering	Minimal path search on a graph with spatial and temporal information
	Zhu et al. (2009)	Background GMM modelling Physical constraints for the ball	Support vector classification Weighted graph and Viterbi algorithm	Support vector regression Particle Filter Particle Filter and template matching

3.2 Ball Detection and Tracking

Although some of the previous referred works already deal with the ball detection and tracking problem, the high complexity associated with it impels researchers to develop specific methodologies and techniques. Therefore it seems pertinent to devote a section to detail a little bit what has been done on this field.

3.2.1 Intrusive Systems

Since the ball has very small dimensions, compared with players, and is constantly being kicked or grabbed it is very difficult to place a sensor on it. Therefore fewer systems exist when compared with player tracking.

A goal detection line system (GLT) was developed by Cairos Technologies in conjunction with Adidas ([Cairos Technologies AG](#)). This system is composed of a magnetic field placed underneath the penalty area and behind the goal line, and a sensor placed in the ball (Teamgeist II) that measures the magnetic fields and transmits the ball location to a computer which analyses the data and determines if a goal was scored (Figure 3.2). In case of goal a signal is sent to the referee.



Figure 3.2: Cairos Technologies and Adidas goal detection system ([Cairos Technologies AG](#)).

Two of the player detection and tracking systems mentioned on Section 3.1.1 also offer ball detection and tracking devices. Besides Minimax GPS sensors for player tracking, Catapult also offers a special small tag with 15g ([Catapult Sports](#)), called e-tag (non GPS technology), that is placed inside the ball and is detected by nearby Minimax devices.

The other supplier is Redfir, which according to Fraunhofer website ([ISS \(2015b\)](#)), uses tags small enough that can be included on the balls.

Another system is being developed by Carnegie Mellon University in conjunction with Yinz-Cam to be applied in American Football ([Seabrook \(2008\)](#)). A special small microchip was designed and includes a small GPS and an accelerometer. Their prototype transmits information at every second and has an accuracy of around 9 meters (30feet), however they are developing a new prototype to transmit information four times per second and combine the GPS information with information from fixed GPS receivers near the field in order to have more precise measures.

Like already explained on Section 3.1.1.2, on Intrusive Systems the tracking method becomes quite straightforward due to the unique tag placed on the ball.

3.2.2 Non-Intrusive Systems

On vision systems, like in the previous section, researchers have at disposal two main sources of image information: TV broadcast images and images from dedicated cameras that are placed at very specific locations.

Usually cameras used for TV broadcast have low frame rates and high shutter times which can cause motion blur when the ball is moving too fast and even miss to capture some important events. On the other hand, dedicated cameras are usually chosen with specific characteristics, namely frame rate and sensor resolutions and placed at specific locations to provide a good view of the field.

[Liang et al. \(2005\)](#) use images provided by TV broadcast. As mentioned before the lower frame rates of these cameras and the long shutter times compels them to take into consideration the motion blur caused by the ball speed and make use of the colour properties of the image.

Under the assumption that the ball is nearly white in long view shots, white pixels are first segmented and candidate regions are defined using physical restrictions. They use candidate regions from a set of consecutive frames to construct a weighted graph where nodes represent ball candidates. The optimal path (which represents the true location of the ball) is extracted using the Viterbi algorithm and a prediction stage is performed with a Kalman filter. Results show that occlusion and motion blur problems are not well handled and the ball can be confused with the players' socks due to the colour similarity. Precision achieved is around 89%.

Images from broadcast TV images are also used by [Pallavi et al. \(2008b\)](#). Like in [Pallavi et al. \(2008a\)](#), initially each frame is categorized into long, medium or close shot based on the ratio between the grass pixels (green colour) and the field region using the YIQ colour space (allows to minimize the effects caused by light changes). They only deal with long and medium shot images since close shot images usually represent the face of a player and therefore do not contain useful information. Afterwards, the first ball candidates are determined using the Circle Hough Transform (CHT) with some radius restrictions in order to eliminate the ones that are less likely to be a ball.

Due to the geometry of the ball it seems appropriate to use the CHT in order to perform its detection, in fact this technique is widely used in applications where the objects to be detected present a round shape which frequently happens in industrial and medical applications ([Chaichana et al. \(2008\)](#); [Huang et al. \(2008\)](#); [Staroveski et al. \(2008\)](#)). However, the CHT requires a huge computational effort and is very time consuming, which is not affordable in real time applications. Therefore, like in this case, authors use it as basis but perform a few adjustments or include some restrictions in order to fasten it.

Once the ball candidates are detected through the CHT, two paths are followed depending on the type of image (long or medium shot).

In case of a medium shot view, the velocity of every ball candidate is calculated through the optical flow velocity method of Horn and Schunck. The candidate having the highest velocity is labelled as being the ball while the others are discarded.

For long shot views they apply the opposite philosophy and instead of searching for the best candidate try to eliminate the less probable candidates. To do so, a series of filters based on previous defined heuristics are used in order to eliminate the weakest candidates. The remnant ones will be used to build a directed weighted graph where the longest path represents the ball trajectory.

The usage of the CHT and of predefined heuristics although providing good results can lead to miss detections if the ball is partially occluded or if by some chance the ball behaves contrarily to the heuristics. The time it takes to process a single frame is too long which makes impossible to have real time processing. Also the usage of optical flow does not seem appropriate since in optical flow it is assumed that the displacement is small, nevertheless results can achieve a precision of 83.1% for medium shots and 94.2% for long shots.

Like in [Pallavi et al. \(2008b\)](#), [Tong et al. \(2007\)](#) also start by classifying the type of view (global view, medium view, close-up and out of view) based on the image dominant colour and only deal with global views. Afterwards, a three level process is started which includes object processing, intra-trajectory and inter-trajectory levels. At the object level, an image segmentation based on the dominant colour is performed in order to extract non-field objects like ball, players, line-marks and others objects in the scene. To discard non-probable candidates several filters are applied based on the Hough transform, shape and size constraints and a colour clues support vector machine.

Afterwards, several possible trajectories are generated at the intra-trajectory level through linking the adjacent candidates in spatial-temporal domain with the aid of a predictive Kalman filter stage. The unconfident trajectories are removed through trajectory filtering using length and timeline relationship restrictions. To identify the true trajectories, the inter-trajectory level, constructs a graph (nodes represent a trajectory and edges the distance between pairs of trajectories) where the optimal path is found using the Dijkstra algorithm. In order to smooth and link up the final ball trajectory, a cubic spline is used to interpolate the gap between two adjacent trajectories.

They performed tests in the FIFA 2006 championship and could achieve an accuracy of 80.3%. Although their intention was to provide a real time framework they state that "The processing ... is beyond real-time with two threads on Intel Core 2 Dual machine with 2.4G CPU, 2.0G RAM."

On the other side there are authors that use dedicated cameras. An example is given by [D'Orazio et al. \(2004\)](#) who developed a system based on one dedicated camera which is able to recognize the ball in soccer images with the intention of aiding referees to verify a goal event.

They are truly focused on providing a real time system that is able to recognize the ball in real images where light conditions can vary and the background is constantly changing.

Two different techniques are combined, a fast circle detection based on directional CHT to identify regions of interest (ROI) and a three layer neural network classifier to evaluate these ROIs in order to accept or discard the hypothesis. The inputs of the neural network are the grey scale

and a 3-level Haar Transform image of the ROIs. The usage of constraints based on a priori knowledge of the ball dimensions and the influence on the shape of the ball due to light conditions helps fastening and improving the algorithm.

Although their results seem very promising they consider that the ball is not visible if more than 50% of its surface is occluded, which seems a very low value. Detection rates of 92% were achieved with artificial light and 97% with natural light. More recently their work has evolved from a single camera system to a four high frame rate camera system (D’Orazio et al. (2009)) (Figure 3.3). Having two cameras pointed to the ball allows them to detect its 3D position through homography. The tracking is performed using simple constraints of ball velocity. Results are still very scarce but seem promising on detecting goal events.

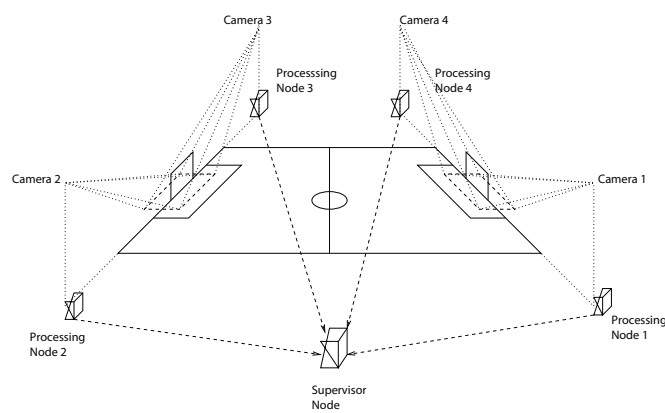


Figure 3.3: Camera distribution in D’Orazio et al. (2009).

The usage of cameras with such a high frame rate provides a more reliable picture of the ball movement and minimizes the effect of motion blur. However, these cameras are highly expensive and such high frame rates leave little time to perform the image processing.

In their new approach, first a moving object segmentation based on background subtraction is performed in order to detect the ROIs and then apply their fast CHT and the neural network classifier. This results in a much faster algorithm since the Hough Transform (HT) is not applied to the entire image but only to the ROIs. Nevertheless, they only detect and track the ball near the goal areas and cannot achieve real time processing.

Ren et al. (2009) also use a dedicated system based on cameras with 25 frames per second that is able to see the entire field with 8 cameras (Figure 3.4).

For detecting and tracking moving objects they model the background using a two-step algorithm. In the first step the background is modelled using a Gaussian Mixture Model and afterwards is continuously updated using a faster running average algorithm. Afterwards, they apply an image-plane Kalman tracker to filter noisy measurements and split merged objects.

From the previous step, result several moving objects that are assigned a likelihood based on ground plane velocity, longevity, normalized size and colour features. This likelihood measure is further refined using occlusion and backtracking reasoning. They are also able to extract the 3D

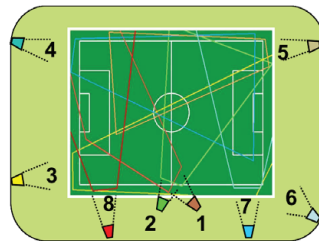


Figure 3.4: Camera distribution in [Ren et al. \(2009\)](#).

ball position and use this information to improve the ball detection. Results show that backtracking is a key feature of their algorithm and that the buffer size influences the accuracy achieved.

3.2.3 Conclusions

It is possible to verify that several methodologies have been used to detect the ball, and most of them are based on different techniques that are not commonly used in the player case. Another important aspect is that the usage of higher frame rate cameras ([D'Orazio et al. \(2009\)](#)) can give a more reliable picture of the ball position, however the complexity of the methodologies applied makes it hard to process the frames in the available time. Table 3.4 gives an overview of these methodologies.

In the ball detection case it is possible to verify that non-intrusive systems do not have a high expression due to the difficulties on placing sensors on the ball. Therefore, authors base their research on vision systems, which can already achieve very high detection rates even with low frame rate (25 fps) cameras ([Ren et al. \(2009\)](#)). However, tests have been performed in small data samples.

Table 3.4: Summary of Ball Detection and Tracking Systems.

Technology	System	Configuration	Objective	Techniques	Results
Intrusive System	Cairos (Cairos Technologies AG)	Sensor and Antennae	Detect goal events	Magnetic detection	Highly accurate
	Catapult (Catapult Sports)	Tag and GPS devices	Ball position compared to players	unavailable info.	unavailable info.
Vision System	Liang et al. (2005)	TV Broadcast	Detect and track the ball	Colour segmentation Viterbi combined with a graph Kalman prediction stage	89%
	Pallavi et al. (2008b)		Detect and track the ball Circle Hough Transform	Optical flow and heuristics Weighted graph	83% (m.) 95% (l)
	Tong et al. (2007)		Detect and track the ball	Colour segmentation, shape and size constraints Predictive Kalman filter Weighted graph combined with Dijkstra algorithm	80.26%
	D’Orazio et al. (2004, 2009)		Detect goal events 3D ball position	Background subtraction Circle Hough Transform Neural Network classifier (Haar transform)	92%-97%
	Ren et al. (2009)		Detect and track the ball 3D position	MoG Kalman tracker	80%

3.3 Game Analysis

High level sports analysis has been performed for a long time using manual methodologies such as hand annotation, however this is a time consuming task, subjective to the person and methods used to extract the metrics (provide low spatial and temporal resolution ([Barros et al. \(2011\)](#))). The usage of automatic methodologies allows not only to speed up the process (all players can be analysed simultaneously), but also reduces the error since the metrics are not extracted based on averages or user subjectivity. Moreover, the effects of tiredness do not affect the process outcome.

This section provides an overview of the most common hand annotation techniques and presents the existing automatic systems that are emerging.

3.3.1 Hand Annotation

Notational analysis as stated by [Hughes and Franks \(2008\)](#) "...is an objective way of recording performance so that critical events in that performance can be identified and quantified in a consistent and reliable manner" and has been used since the early days of sport. Although it has started mainly on soccer and squash ([Franks and Hughes \(2004\)](#)), nowadays it is performed in nearly every sport. Also according to the same authors, the first attempt to create and use a notational system able to analyse multiple players is credited to [Messersmith and Bucher \(1939\)](#), who tried to notate the distance traversed by big players during a basketball match.

Initially, notational analysis was based on hand annotation and several research works were performed making use of it (for further details please refer to [Franks and Hughes \(2004\)](#)), however this technique usually requires a huge time effort to collect the raw data and generates enormous amounts of information that are subjective to the operator, somewhat difficult to handle and moderately reliable ([Duthie et al. \(2003\)](#)).

Advances on technology opened the possibility of introducing new computerized tools to aid sport experts during this process. The benefits of these tools include the possibility to deal with huge amount of data more easily, in a shorter period of time and also with higher sampling frequencies which translate into higher accuracies. Nevertheless, the developed solutions must be carefully validated as highlighted by [Franks and Hughes \(2004\)](#).

For the specific case of handball, there is still not much data available and most works are based on simple assumptions that make it impossible to analyse the behaviours and interactions that can occur in a handball game. More in depth investigations, using hand annotation include the study of attack organizations to assess their success on specific situations.

[Yiannakos et al. \(2005\)](#) estimated the players' level of stamina using information about the type of attack, the outcome of the shot and the offensive errors. Their test sample consisted of 15 matches with 1503 attacks.

[Gruic et al. \(2006\)](#) analysed the elements of situational attack efficiency, such as 9m shots scored, 9m shots taken but missed, 6m shots scored, among others, in 60 handball matches and the contribution of standard performance parameters to the final score. Besides a global analysis of the 24 teams playing the championship, the study involved an inter-group analysis based on four

qualification groups. Their results confirmed the statistically significant contribution of the chosen predictor variables to the successfulness of the teams, however the structure of contributions differed among the groups. More recently, the same methodology was applied to a women championship and the same conclusions were drawn. Additionally, the authors highlighted the need to modify the methodology since, as they state, only fragments of the complexity of the handball game were analysed and they obtained a considerable variability on the parameters studied. Once again, they pointed out that situation efficiency models are different between teams and almost each match.

Although notational systems are intensively used on high level competition, especially to evaluate and study opponent teams, this information is usually kept in secret and therefore little information is available.

3.3.2 Automatic Analysis

As demonstrated on the previous sections ([Barros et al. \(2011\)](#); [Kristan et al. \(2009\)](#)) have developed vision based systems able to automatically track handball players without interfering with the game itself. Their analysis is still based on physiological parameters such as distance travelled and players' velocity. However, such systems enable the possibility to perform a more in depth analysis of the game, not only because they provide information with higher sampling frequencies, but also because they are able to extract the players' positions on the field without being subjective to an operator judgement.

When talking about automatic game analysis two main groups arise: semantic analysis and high level game analysis. An interesting comparison between these two concepts is proposed by [Zhu et al. \(2009\)](#) "Semantic analysis aims at detecting and extracting information that describes "facts" ... tactic analysis ... aims to recognize and discover tactic patterns and match strategies that teams or individual players used in the game."

Most of the existing approaches on sport's video analysis are concentrated on semantic analysis by annotating and indexing video footages, which is generally oriented to audience ([Bai et al. \(2009\)](#); [Buitelaar et al. \(2008\)](#); [Duan et al. \(2005\)](#); [Wang et al. \(2008\)](#)). Although semantic analysis can also be useful to coaches since it has the ability to delimitate specific events such as goal situations or attack events, it lacks on the ability to provide further information related with tactics and the game itself.

Recently, some groups with more or less complexity are exploring this high level game/tactics analysis ([Beetz et al. \(2009\)](#); [Chin et al. \(2005\)](#); [Kang et al. \(2006\)](#); [Perše et al. \(2009\)](#); [Taki et al. \(1996\)](#); [Zhu et al. \(2009\)](#)).

[Taki et al. \(1996\)](#) evaluate the cooperative movement of players and the ability of the team to make space using two concepts: minimum moving time pattern and dominant region, that assent on low level features such as player's position, velocity and acceleration. Additionally, the ball dominant regions can also be calculated which allows detecting high quality passes. Despite, providing two interesting metrics to evaluate teamwork, the way these two concepts are evaluated

on real game situations is lightly addressed on the paper and it would be important to compare these results with hand annotated information.

Chin et al. (2005) model basketball defensive strategies as matrices that represent the spatial-temporal relationships among players, additionally they provide a similarity measure which allows detecting similar defensive strategies of clips stored in a database. In order to use this system, the video under analysis has to be clipped by a user and each clip must have a similar number of frames. It would be interesting if the system could automatically detect defence clips and similar defensive strategies even for clips with considerably different sizes.

In order to overcome this static property, Perše et al. (2009) use a template based recognition method. Templates of the team activities are stored prior in a database (using a graphical interface) and are represented as semantic descriptions (chains of symbols), which confers them more flexibility. Additionally, they are able to segment basketball games according to its phases (offense, defence and timeouts) using GMM. Tests on real games gave phase rates detection from 88% to 94% and tests on a specific test sample gave activity recognition of 100% (screen, move and player formation).

More recently, their work has evolved (Perše et al. (2010)) and the activity templates were used to create Petri Nets (PN) which express not only the sequential relations among actions (between several players), but also encode temporal information concerning each action by learning from training data. With this methodology they are able to determine the activity the team performed (as long as the template is in the database) and at which stage it ended. Nevertheless, tests were only made for specific projected situations and not in real game situations, where the random factors cannot be so well controlled.

Beetz et al. (2009) perform game analysis using ontology models of the game with players' positions, motion trajectories and ball actions as primitives. One of the strongest and innovative points of this system is the definition of an hierarchical ontology of game models (Figure 3.5), that contains a static part independent of the class of soccer games, and a dynamic part that is learnt using machine learning techniques (decision/regression trees, clustering). This dynamic property makes the system very flexible and allows it to be used in different contexts (e.g. Junior vs. Senior leagues). They are able to define tactical set-ups, assess models for situations, tactical behaviours, and identify attack and defensive systems. FIFA soccer world championship 2006 was their test data set.

Zhu et al. (2009) use ball and players' positions to perform tactical analysis on soccer attack events (that are detected using web-casting) by formalizing route patterns using field partitioning and interaction patterns such as cooperative attack (unhindered or interceptive attack) and individual attack (direct or dribbling attack). The correct identification of these concepts ranges from 70% for dribbling attack to 93% for cooperative attack. Despite formalizing and classifying attack situations, they do not provide metrics in order to assess its performance, for example it would be interesting to evaluate which attack is more effective for a given team.

Kang et al. (2006) propose a model to quantitatively express the strategic performance of soccer players. Their model is based on the trajectories of all game elements, including the ball,

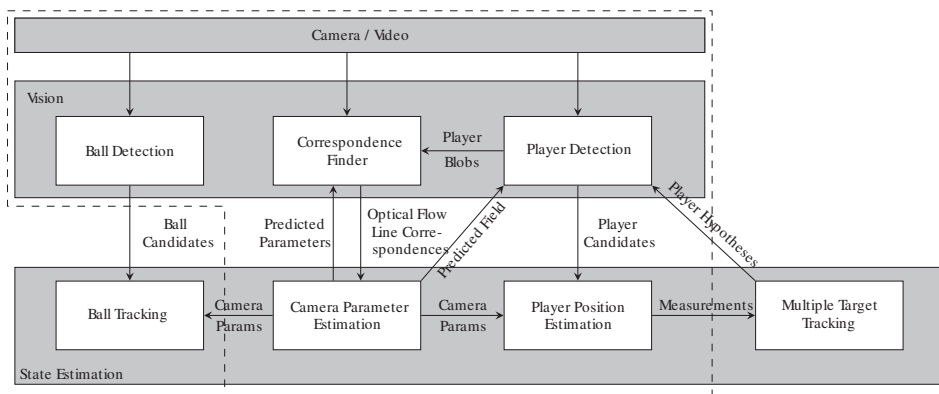


Figure 3.5: Aspogamo model hierarchy from [Beetz et al. \(2009\)](#).

and defines/formalizes several concepts such as kicks and receives, possible regions, catchable regions, and safe/competing regions. Associated with these concepts they define metrics that allow evaluating and comparing players' performance. However, the validation of these concepts was not performed on real game data but rather on data from a simulated soccer game.

[Bai et al. \(2009\)](#) propose the use of perception concepts (either aural, visual or motion) and Petri Nets to semantically describe the match and detect events. They were able to model complex concepts such as scored goal, yellow and red cards in soccer; fast breaks and fouls in basketball and try in rugby with a precision that ranges from 71.4% to 100%.

Still on the Petri Nets field, [Marin et al. \(2010\)](#) implemented a soccer model based on a Hierarchical Coloured Petri Net. Their model defines at any time, the efficiency and also the errors that occurred per team – number of good passes, efficiency of the shots at the goal, fouls taken, fouls given, among others.

Another interesting work is presented by [Duch et al. \(2010\)](#). Their analysis is based on the statistical information provided by UEFA for the European Cup 2008 soccer tournament. Using methods from social network analysis they were able to quantify the performance of players and teams. Their analysis is not concentrated on a single game but rather on a series of games.

They used a directed network of "ball flow" between players. In this network, nodes represent players and arcs are weighted according to the number of passes successfully completed between two players. Two additional nodes are included to represent "shots to goal" and "shots wide". A player node is connected to these two nodes by arcs weighted according to the number of shots.

The team's performance is inferred from the players' performance. They are able to extract metrics such as passing accuracy, shooting accuracy, flow centrality (probability that each path definable on the network finishes with a shot) and match performance (normalized value of the logarithm of the player's flow centrality in the match) to determine the players and team performances.

Figures 3.6 and 3.7 illustrate the kind of information provided by this work. In Figure 3.6 G letter represents "shots to goal", W "shots wide" ([Duch et al. \(2010\)](#)).

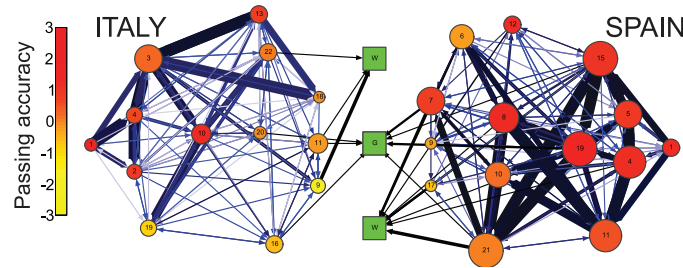


Figure 3.6: Passing accuracy of the game Italy vs. Spain (Duch et al. (2010)).

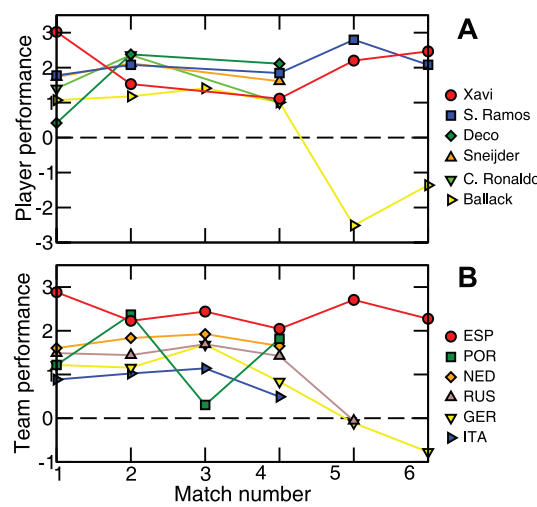


Figure 3.7: Time evolution of the performance of players and teams (Duch et al. (2010)).

Another interesting area is the one that explores robotic soccer. For this specific case, the players' positions as well as the ball are known and so events such as successful/missed passes, end line shots, intercepted shots, goal, outsides among other can be determined using specific algorithms such as the ones explored on Abreu et al. (2012, 2010); Almeida et al. (2012).

On the area of Robotic Soccer there is also a large amount of work on areas related with game modelling, such as coordination and strategy definition (Reis and Lau (2000); Reis et al. (2000)), coaching (Reis and Lau (2001)), setplays (Cravo et al. (2014); Mota and Reis (2007)) and formation/tactical analysis and recognition (Abreu et al. (2014); Faria et al. (2010)).

3.3.3 Conclusions

Most indoor systems are only concerned with assessing the players' effort (distance covered, velocity) and usually do not take the following step of performing high level game analysis. Moreover, the few works that truly perform game and tactical analysis are still in early stages and usually focused on very specific set plays, rely on user video clipping or were tested either on specific situations or simulated data.

3.4 Summary and Conclusions

In order to perform motion analysis, it is necessary to have each player position over time which can be obtained by two different approaches: Intrusive Systems and Non-Intrusive Systems (comparison summarised in Table 3.5).

Table 3.5: Taxonomy and summary regarding player detection techniques.

Intrusive	GPS	Cannot be used on indoor environments Sensor complexity and weight
	Tag+Antennae	Likely to have problems with interferences, metals, fog, rain, other RF systems (cellular antennas, ...)
Non-Intrusive	Broadcast	Affordable if games are broadcasted (very expensive otherwise) Partial view of the field
	Dedicated cameras	Dedicated setup (not easy to relocate and may not be allowed in the other teams' sports halls) Usually cover the entire playing area Multi-cameras minimise occlusion and enable 3D localisation

Intrusive systems are more insensitive to crowded fields, contact between players and can provide 3D measurements at high rates. However, nearby objects, structures and apparatuses that operate at similar frequencies can cause interferences and therefore their use in specific sports halls may be compromised. Also their usage in official competitions may be restricted or dependent on agreements, and most players do not like to wear devices that may interfere with their performance.

Recent advances show that Vision based non-intrusive systems can be very promising, not only due to the costs involved (intrusive systems are usually more expensive), but also because they can be used to track the visitor team without restrictions, even at official matches.

One of their major drawbacks is that they require sophisticated methodologies to extract the players' positions. Due to this limitation, most existing systems were either tested on limited datasets or require user intervention in order to correct the tracking and, therefore, a totally automated application is still not available. Moreover, not all systems provide the players' positions in real world coordinates, which is of the utmost importance if game analysis is to be performed, because the spatial relations between players cannot be truly assessed with image coordinates. Systems from both categories are not easily portable and require mounting and calibration.

Another aspect that must be highlighted is the relevance of knowing the ball position since it improves how game analysis and team performance evaluation is performed, and from what has been seen is an issue that is far from being solved.

With motion information from all game elements it is possible to characterise the players' effort by extracting measurements such as covered distance, preferred field zones (Barros et al. (2011); Monier et al. (2009)) or speed profiles (Monier et al. (2009)).

Chapter 4

Methodology

4.1 Proposed Architecture

A three module software system is proposed, one responsible for acquiring the images from the multiple camera system (Acquisition System) and another for the off-line processing of the video streams (Processing System). This last one detects and tracks the players and generates a log file with the players' positions, so that they can be used to perform game analysis and infer game statistics. Finally, the last module (Visualizer and Annotator) is able to merge the video streams and the log file to create a global image of the field with the players highlighted, as well as to perform high level game analysis. Figure 4.1 illustrates the architecture described.

The interaction between the different modules is performed by means of logs that assure all the information necessary is passed between them. Figure 4.2, clarifies these interactions.

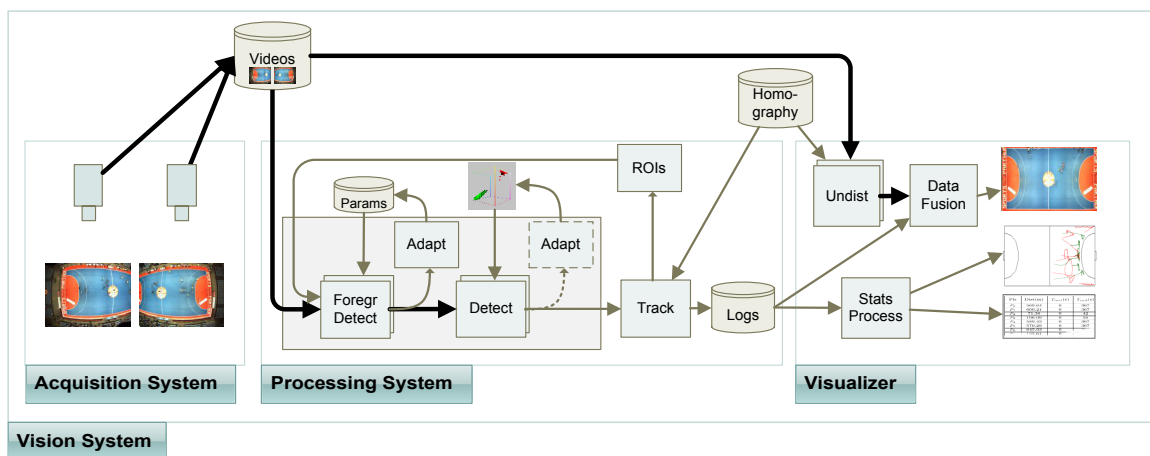


Figure 4.1: System's architecture (from [Santiago et al. \(2013\)](#)).

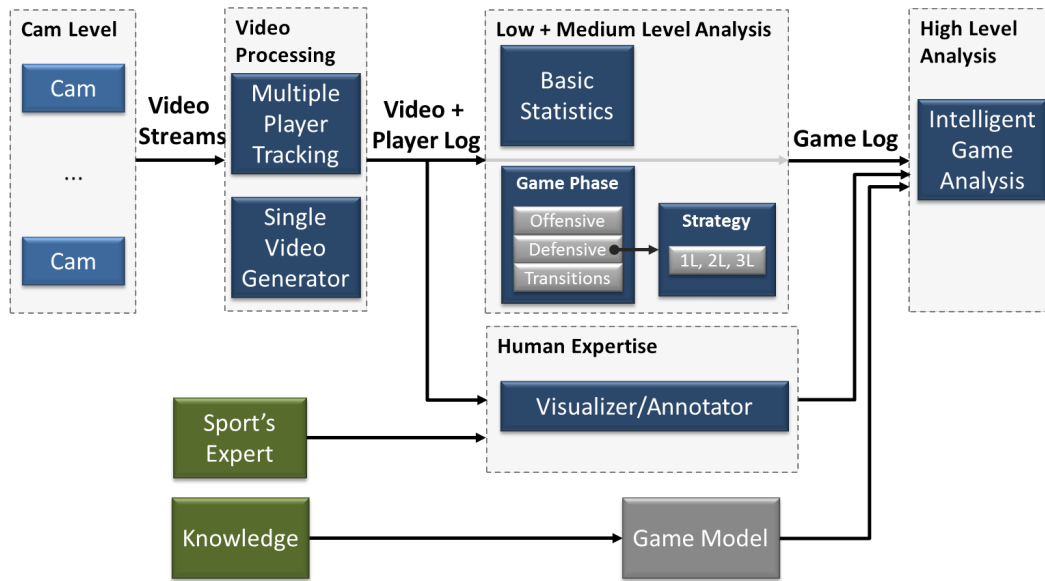


Figure 4.2: Data flow information.

4.1.1 Acquisition System

The challenges of defining a vision system able to identify and track the game elements in an invasion team game are huge due to the dynamic and spatial characteristics of the game itself. In fact, handball is played in an area of 20 by 40 meters and it is quite a dynamic game, with high physical contact among players and rapid movements (a player can achieve velocities higher than 5m/s). These characteristics impose a careful choice on the system's architecture, which includes choosing not only the cameras and their disposition but also defining the software design.

The system must cover the entire handball field including the extra border, the obtained image should have enough resolution to correctly detect all players and capture images with a frame rate adequate to the involved speeds. The problem's particularities and the sport's hall characteristics place the ceiling as the best spot to set the camera system, since there is no interference from the crowd, a single player never fills the entire field of view of the camera and a bird's eye perspective usually means less occlusion and/or merging situations (this solution was also adopted by [Kristan et al. \(2009\)](#) and [Monier et al. \(2009\)](#)).

On the other hand, placing the cameras on the ceiling generally forces the usage of multi-camera systems for additional resolution. Although this choice may result in a more complex system, it also carries the advantage that some parts of the field are covered by more than one camera, which provides two views of the same portion of the field that can be used to overcome or minimize occlusion situations.

Given the enumerated characteristics, the proposed system uses, depending on the configuration, 1, 2 or 3 Gigabit Ethernet cameras DFK 31BG03.H model from Imaging Source. Resolution is 1024 x 768 pixels and the camera can deliver up to 30 frames per second. The used lenses

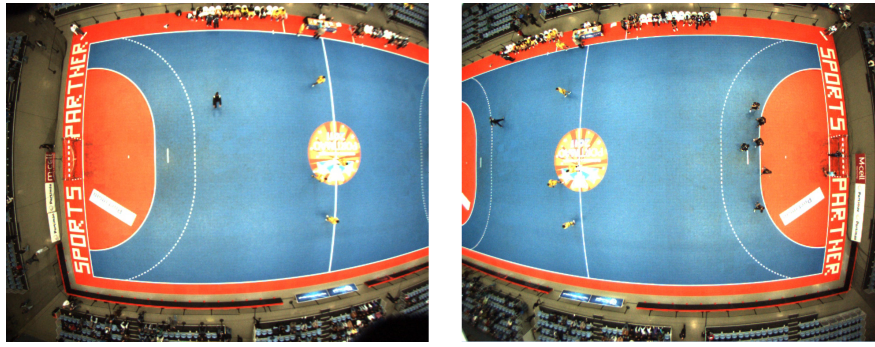


Figure 4.3: Images collected from the two camera's system.

are Computar T2Z1816CS Vari-focal lens with focal distances ranging from 1.8mm-3.6mm (wide angle lens).

When compared with other implementations, the presented architecture and hardware choice allow an "easy" set-up that can be transferred between sports halls. The choice for industrial grade Gigabit Ethernet interfaced cameras allows reliability with digital quality, high data rate and low cost: no frame grabber, common hardware, low cable costs while still allowing large distances. The chosen Vari-focal lenses have an interesting price and allow (manual) "zooming", and therefore cope with different pavilions, that have ceilings at different heights. These advantages come at the expense of an important image distortion.

Figure 4.3 shows the images collected from a two camera's system. It can be seen that the barrel effect is wide due to the usage of the aforementioned inexpensive lenses. The mentioned combination offered excellent performance at an interesting cost at the time of the start of the project, in 2010. Fig. 5.1, illustrates the same system using three cameras at the Académico's Sports Hall in 2012.

4.1.2 Processing System

The processing system is responsible for analysing the videos collected by the Acquisition System, detecting the players on the field and tracking them.

The first step on this system consists in identifying which areas correspond to foreground and which areas correspond to background. In order to make this process as independent from the light conditions as possible, background is modelled using an adaptive method able to cope with light changes.

Afterwards, the detected foreground zones are analysed based on their colour properties in order to identify the players. Having in mind the same adaptive line of thought, a Fuzzy categorization methodology is used that allows to continuously model the players' colour properties.

Finally the tracking is accomplished by converting the players' positions into real world coordinates by means of the cameras' homographies and using a vector of Kalman Filters.

The result of the Processing System is a log file with the players' real world positions during the game as well as their velocity.

The detailed description of the methodologies and algorithms used during the processing step are given on Sections 4.2 and 4.3.

4.1.3 Visualizer and Annotator

The Visualizer and Annotator system is an important tool for the end user, since it allows visualizing the videos from the cameras in a single undistorted and meaningful image, observing player and team statistics, detecting game events and annotating the game.

4.1.3.1 Video Visualization

The images coming from the cameras (as can be seen in Figure 4.3) are deeply affected by distortion due to the used lenses and contain a high overlapped region, and therefore constitute an awkward view of the field. In order to provide an image that is useful for the user it is necessary to build a unified image that shows a natural view of the scene.

Since the cameras homographies have already been computed (to convert the players coordinates into real world coordinates) they can be used to generate this image, by first converting the input video streams into a common reference, which is the real world and afterwards convert them into one common reference on the image plane. The visualization tool also allows the user to navigate throughout the video by providing forward, backward and go to options.

4.1.3.2 Player and Team Statistics

One of the first and most straightforward information that can be extracted from motion analysis is the player and team statistics concerning position and speed which allows characterizing not only the players' effort but also preferable playing areas and attack corridors. This information is shown in different ways:

- Tabular form with total distance covered and time spent on the field, maximum and average velocities (Figure 4.4(a));
- Positional and speed maps. On positional maps, colours indicate the frequency the player occupies a given area, while on speed maps it is possible to choose the range of speeds to be shown (Figure 4.4(b));
- Tactical maps of both teams during a user defined time frame (Figure 4.4(c)).

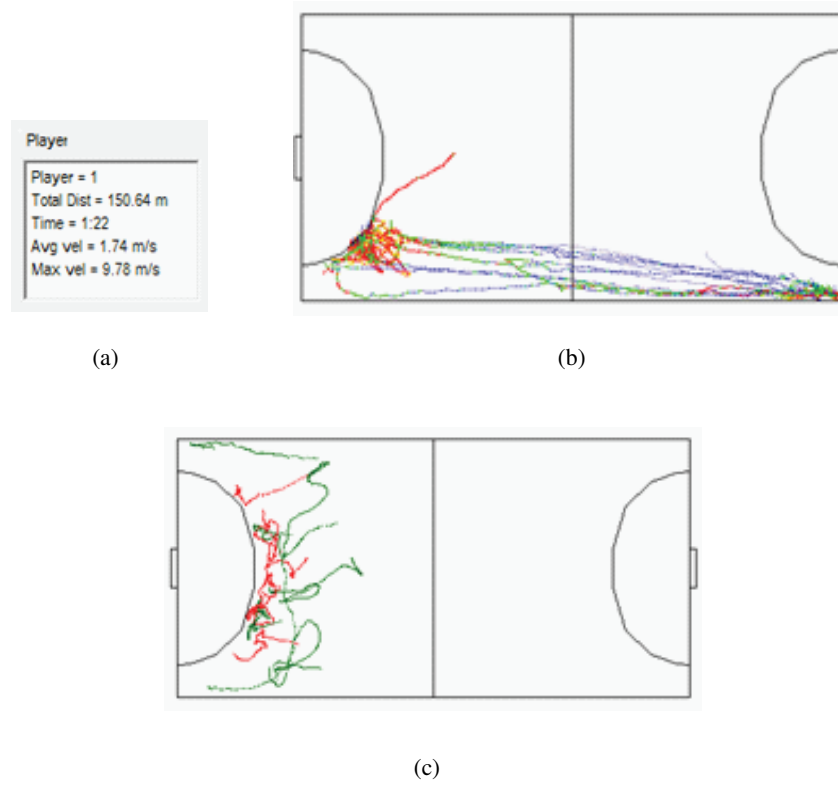


Figure 4.4: (a) Player effort in tabular form (b) positional map indicating the zones occupied by the player (c) tactical map of both teams during a time frame defined by the user.

4.2 Player Detection

Player detection is achieved through colour identification and is composed of three steps. The first step (Section 4.2.1) consists in the user based colour calibration of each team, using a region growing method allied with a Fuzzy categorization methodology. This calibration is responsible for subdividing the colour space into subspaces, which are not necessarily disjoint since there may be colours common to both teams (for example, it is common to have teams with white stripes).

The second step (Section 4.2.2) consists in detecting foreground regions through a dynamic background subtraction method, which uses an empty image of the field and a dynamic threshold that is continuously and locally updated at each new frame.

After the foreground pixels are identified, their colour is compared against the colour subspaces and classified into one of the teams (Section 4.2.3). In case there is a belonging tie between teams, information of adjacent pixels is used in order to break the tie. Additionally, the teams' colour subspaces are updated with new information.

Finally, pixels are aggregated to form blobs and categorized into player or no player (noise), according to size and density restrictions. The centre of mass of the blob is considered the player's position that is afterwards transformed into court coordinates using the cameras' homographies.

The usage of simple, robust and parallelizable methodologies is intentional, so that, making use of parallel technologies (multi-threading, Open Multi-Processing (OPENMP) (Chapman et al. (2007))), Graphics Processing Unit (GPU) (Luebke and Humphreys (2007)) and code optimization, real time processing can be achieved.

4.2.1 Colour calibration

The colour calibration is performed under user supervision and is achieved using a region growing method allied with a Fuzzy categorization methodology.

Let us define colour subspace S_c as the set of RGB colour triplets that are tagged as having the colours of the vests of team c . The initial colour seeds $C(x_s, y_s)$ for each colour subspace S_c are set manually using the mouse to click on the objects that will be segmented. Afterwards, the surrounded pixels' colours $C(x_a, y_a)$ are agglomerated around these seeds using colour distance criteria. Colour expansion is performed on the HSL (Hue, Saturation and Luminance) colour space in order to minimize the effects of shadows and light variations.

Regions growth is performed in all directions (using a 8 neighbour mask n_8) in a recursive way until reaching a pixel that, in terms of colour, is more than a global threshold (C_{ThresG}) away from the seed or more than a local threshold (C_{ThresL}) away from its previous neighbour $C(x_p, y_p)$, according to the following definition (both thresholds are user definable):

Rule 0

$$C(x_a, y_a) \in S_c \Leftrightarrow \forall (x_a, y_a) \in (n_8(x_p, y_p) \wedge \Delta(C(x_a, y_a), C(x_p, y_p)) < C_{ThresL} \wedge \Delta(C(x_a, y_a), C(x_s, y_s)) < C_{ThresG})$$

Where, $C(x, y)$ is the HSL colour of pixel at location (x, y) , $n_8(x, y)$ are the eight neighbours of the pixel at location (x, y) and $\Delta(C_1, C_2)$ represents the colour distance on the HSL colour space between colours C_1 and C_2 .

During the colour expansion process, each colour value is attributed a given belonging degree to the subspace being calibrated. This value is stored in a lookup table that contains, for each colour triplet, the belonging degree to each subspace. Despite the expansion being performed on the HSL colour space, the colour lookup table is built on the RGB (red, green, blue) colour space, which is the format provided by the camera. Therefore, in execution time, labelling is performed via this lookup table, which allows fastening the process.

The Fuzzy belonging degree (μ) of the colour C of a pixel P of coordinates (x_p, y_p) to a given colour subspace S_c is $\mu_{S_c}(C(x_p, y_p))$ and can assume four levels, according to Table 4.1. By default, and before the calibration takes place, all the colours are categorized with no belonging degree to every subspace.

In order to determine the belonging degree of the colour triplet to the respective subspace, the following rules (Rules 1, 2 and 3) are sequentially applied during the colour calibration region growing process. To make it easier to understand the categorization rules, each colour belonging degree has a corresponding alias (B_{S_c}).

Table 4.1: Mapping of B_{Sc} to Fuzzy belonging μ .

Colour	B_{Sc}	μ_{Sc}
Not the colour	C_0	0
Resembles the colour	C_L	0.5
Is the colour	C_F	1
Is a seed colour	C_S	1

Rule 1

$$B_{Sc}(C(x_a, y_a)) = C_S \Leftrightarrow C(x_a, y_a) \in (S_c \wedge D((x_a, y_a), (x_s, y_s)) < D_{Low} \wedge \Delta((x_a, y_a), (x_s, y_s)) < C_{SThres})$$

If the pixel was assigned to the subspace, is physically quite close to the initial seed pixel ($D((x_a, y_a), (x_s, y_s)) < D_{Low}$) and the colour distance to the initial seed pixel is less than a small threshold (C_{SThres}), then it is also assumed to be a seed pixel with a full belonging degree.

Rule 2

$$B_{Sc}(C(x_a, y_a)) = C_F \Leftrightarrow C(x_a, y_a) \in (S_c \wedge \Delta(C(x_a, y_a), C(x_s, y_s)) < C_{MThres})$$

If the colour distance to the initial seed pixel is less than a medium threshold (C_{MThres}) for the growing process, then the pixel is categorized with a full belonging degree but without being a seed.

Rule 3

Otherwise, and in case the pixel obeys to the region growing conditions (Rule 0), it is categorized with a low belonging degree (C_L).

By the end of the calibration process, the colour space is subdivided into subspaces, which are not necessary disjoint since the same colour can belong to different subspaces, with different belonging degrees. The motivation for allowing non-disjoint subspaces is that teams frequently share colours, for example uniforms with white stripes are common and, thus, the exact same well known colour belongs to the two opposing teams.

As will be seen later, the belonging degrees assigned to each colour triplet, will allow breaking ties but also to generate dynamic subspaces that can adapt, either grow or shrink, during the game. Subspaces do not have, nor ever create, any predefined specific shape as they are created from user selected seeds on the image and based on the frames characteristics.

4.2.2 Background Subtraction

Since the background is more or less static, due to the semi-controlled environment of an indoor game, the subtraction is performed using an empty image of the viewed scene recorded prior to

each competition and only the threshold used to distinguish between foreground and background pixels is allowed to vary. This threshold is specific for each pixel.

Background subtraction is performed on the RGB colour space, because tests showed that, for some pixels, a small difference between the RGB colour components of the background and the processed images corresponded to a large difference on the Hue component (HSL colour space). In fact, non-linear colour spaces suffer from the non-removable singularity problem as stated by [Cheng et al. \(2001\)](#).

Also, in order to make the processing time shorter, the subtraction is executed locally and not to the entire image. In other words, only predefined regions, which are defined by the Kalman Filter predictive stage, suffer this process (Section 4.3).

The threshold applied to each pixel is only updated if the pixel is classified as background, otherwise its value remains unchanged. The update obeys to Equation 4.1 and the value is never allowed to go below 4% or above 23.5% of the entire colour range (0-255) for each colour component (these values were obtained experimentally by trial and error).

$$\sigma_{t+1}^c(x,y) = \begin{cases} \alpha |I_t^c(x,y) - B^c(x,y)| + (1 - \alpha) \sigma_t^c(x,y), & \text{if } I_t(x,y) \in B(x,y) \\ \sigma_t^c(x,y), & \text{otherwise} \end{cases} \quad (4.1)$$

Where σ is the threshold of the pixel at position (x,y) , time $t+1$ and colour component c , I_t^c is the colour intensity of the pixel at position (x,y) , time t and colour component c , B^c is the background colour intensity of the pixel at position (x,y) and colour component c , and α is a learning constant, that for our specific case was set to 0.02.

Pixels whose colour difference from the background image is less than the respective threshold are labelled as background and the threshold updated, the other pixels are labelled as foreground.

4.2.3 Team identification

After the foreground pixels are identified, their colour is compared against the colour lookup table that resulted from the calibration process (Section 4.2.1) and then classified into one of the subspaces.

Since the same colour can belong to different subspaces (due to a shared colour among teams) it may occur that a pixel is classified into more than one subspace. To obtain a crisp value of the team the pixel belongs to (S_c), the Fuzzy inference model uses not only the belonging degree itself (μ), but also information about adjacent pixels that have already been classified, or more precisely the proportion of pixels belonging to each subspace ($\frac{n_A}{n_B}$) according to the inference associative matrix defined on Table 4.2 (δ is a small constant defined by the user). This table illustrates the process when the colour is shared among two colour subspaces, however it can be easily transposed for a three case problem by including the remaining team(s) on the denominator:

$$\frac{n_A}{n_B + n_C} > 1 + \delta \rightarrow S_A.$$

Table 4.2: Fuzzy inference associative matrix.

$S_c(x, y) =$		μ_{S_B}		
		0	0.5	1
μ_{S_A}	0	ϕ	S_B	S_B
	0.5	S_A	$\begin{cases} \frac{n_A}{n_B} > 1 + \delta \rightarrow S_A \\ \frac{n_A}{n_B} < 1 + \delta \rightarrow S_B \\ \text{else} \rightarrow \phi \end{cases}$	$\begin{cases} \frac{n_A}{n_B} \gg 1 \rightarrow S_A \\ \text{else} \rightarrow S_B \end{cases}$
	1	S_A	$\begin{cases} \frac{n_B}{n_A} \gg 1 \rightarrow S_B \\ \text{else} \rightarrow S_A \end{cases}$	$\begin{cases} \frac{n_A}{n_B} > 1 + \delta \rightarrow S_A \\ \frac{n_A}{n_B} < 1 + \delta \rightarrow S_B \\ \text{else} \rightarrow \phi \end{cases}$

Using this Fuzzy inference model it is possible that, although the belonging degree of a pixel to a subspace based on the colour calibration information is higher than the belonging degree to the other subspace, it may be the winner due to the neighbourhood characteristics.

Additionally, if the winning subspace has a full belong to that colour triplet and corresponds to a seed colour (C_S), then a region growing process is triggered, and the colour lookup table that contains the information concerning the colour subspaces is updated. This auto expansion is more restrictive than the one performed during the manual initialization and is triggered at time intervals (t_{exp}), that can be defined by the user.

In order for this update to add not only colour triplets to the subspaces but also to remove them, each colour triplet has associated a persistence ($p_{S_c}(R, G, B)$) to that subspace. Colours with lower belonging degrees have lower persistence and colours with higher belonging degrees have higher persistence. The initial persistence given to the colour is proportional to the time between auto expansions, according to Equation 4.2.

$$\begin{cases} p_{S_c}(R, G, B) = C_{Low} \times t_{exp} & , \text{if } B_{S_c}(R, G, B) = C_L \\ p_{S_c}(R, G, B) = C_{Med} \times t_{exp} & , \text{if } B_{S_c}(R, G, B) = C_F \\ p_{S_c}(R, G, B) = C_{High} \times t_{exp} & , \text{if } B_{S_c}(R, G, B) = C_S \end{cases} \quad (4.2)$$

The persistence is maximum, with the values defined in Equation 4.2, when the colour is added to the subspace and decreases whenever it is not detected in a frame. When the persistence value reaches zero, the colour triplet is removed from the subspace.

With the introduction of this dynamical behaviour (the update of the look up table), it is possible to have mutable colour subspaces that adapt to light changes, either occurring at different regions of the same frame or between frames.

At the same time the foreground pixels are classified, they are also aggregated horizontally to form Run Length Encoding (RLE) structures characterized by the y , x_{min} and x_{max} positions of the RLE. An outline of the algorithm to generate these RLE structures is presented on Algorithm 1.

Where the *endSeg* function is described by Algorithm 2. This last algorithm demonstrates how the end of the RLE structure is obtained. During this process a filtering is performed and adjacent

Algorithm 1 *findRLEs*

```

/* Searches the input ROI and returns the existing RLEs */
for  $y \in ROI$  do
  for  $x \in ROI$  do
     $curSubspaceN \leftarrow getCrispSubspace(C(x,y))$ 
    if  $OK(curSubspaceN)$  then
       $TmpRLE \leftarrow endSeg(curSubspaceN, ROI, x, y)$ 
      if  $OK(TmpRLE)$  then
         $RLEs.add(TmpRLE)$ 
      end if
    end if
  end for
end for

```

RLE structures are merged together. The conditions that dictate this merging are related with the distance between two structures and if they belong to the same colour subspace.

Algorithm 2 *endSeg($curSubspaceN, ROI, xInit, y$)*

```

/* Identifies the characteristics ( $x_{min}, x_{max}$ ) of the RLE*/
 $x \leftarrow xInit$ 
for  $x \in ROI$  do
  while  $getCrispSubspace(C(x,y)) = curSubspaceN$  do
     $x \leftarrow x + 1$ 
  end while
   $xEnd \leftarrow x$ 
  while  $getSubspace(C(x,y)) \neq curSubsp$  do
     $x \leftarrow x + 1$ 
  end while
  if  $length(xEnd - x) > interRLEMinDist$  then
    break
  end if
end for
if  $length(TmpRLE) > minRLESize$  then return  $TmpRLE$ 
elsereturn  $-1$ 
end if

```

Finally, the RLEs are merged vertically to form blobs. Again, if the distance between two of these lines is small, and they belong to the same colour subspace, they are considered as being part of the same blob and are connected together.

The blobs resulting from this pixel aggregation are further refined, according to size and colour density constraints. Therefore, blobs that are too small or too large or blobs that have low colour density are discarded as being players. The colour density is measured as the percentage of pixels inside the rectangular bounding box of the blob that belong to the subspace divided by the total number of pixels. The remaining blobs are considered players that belong to a given subspace (team) (S_c) and have an (x, y) position on image and world coordinates.

The position on image coordinates is calculated as the blob's centre of mass, according to Equation 4.3. This is a weighted centre of mass calculation because it uses the belonging degrees

defined on Table 4.1. This way, pixels that are seeds or fully belong to the subspace (C_S or C_F) have a higher contribution to the final result.

$$(x_{cm}, y_{cm}) = \left(\frac{\sum_x \sum_y \mu_{Sc}(C(x, y))x}{\sum_x \sum_y \mu_{Sc}(C(x, y))}, \frac{\sum_x \sum_y \mu_{Sc}(C(x, y))y}{\sum_x \sum_y \mu_{Sc}(C(x, y))} \right) \quad (4.3)$$

Where c is the team the blob belongs to, $C(x, y)$ is the colour of pixel at location (x, y) and μ_{Sc} is the Fuzzy belonging degree assigned during the colour calibration phase.

The world coordinates are obtained by first removing the barrel effect produced by the lens (only radial effect was considered, since the tangential component was found to be insignificant) using Equation 4.4. The unknowns in this equation system (k_1 , k_2 , k_3 , x_c and y_c) are determined using the information extracted from the field lines.

$$\begin{cases} x_u = x_d + (x_d - x_c)(k_1 r^2 + k_2 r^4 + k_3 r^6) \\ y_u = y_d + (y_d - y_c)(k_1 r^2 + k_2 r^4 + k_3 r^6) \end{cases} \quad (4.4)$$

Where: $r^2 = (x_d - x_c)^2 + (y_d - y_c)^2$, (x_u, y_u) are the undistorted coordinates, (x_d, y_d) are the distorted coordinates, (x_c, y_c) are the coordinates of the centre of distortion of the lens and k_1, k_2 and k_3 are the radial coefficients for barrel distortion.

Figure 4.5 illustrates the images before and after removing the barrel effect for a two camera's system.

Once the barrel effect is removed from the images, it is possible to apply the pinhole camera model in order to obtain the world coordinates. This model uses intrinsic parameters (K) and extrinsic parameters (R and T) to map image coordinates (X) into world coordinates (x), according to Equation 4.5, a process known as homography.

$$x = K[R|T]X \Leftrightarrow x = H_w X \quad (4.5)$$

The H matrix is defined according to Equation 4.6.

$$H_w = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R_{T_{11}} & R_{T_{12}} & R_{T_{13}} & R_{T_{14}} \\ R_{T_{21}} & R_{T_{22}} & R_{T_{23}} & R_{T_{24}} \\ R_{T_{31}} & R_{T_{32}} & R_{T_{33}} & R_{T_{34}} \end{bmatrix} \quad (4.6)$$

Where $R_{T_{11}} = \cos \phi \cos \alpha$, $R_{T_{21}} = \cos \phi \sin \alpha$, $R_{T_{31}} = -\sin \phi$, $R_{T_{12}} = \sin \omega \sin \phi \cos \alpha - \cos \omega \sin \alpha$, $R_{T_{22}} = \sin \omega \sin \phi \sin \alpha + \cos \omega \cos \alpha$, $R_{T_{32}} = \sin \omega \cos \phi$, $R_{T_{13}} = \cos \omega \sin \phi \cos \alpha + \sin \omega \sin \omega \sin \alpha$, $R_{T_{23}} = \cos \omega \sin \phi \sin \alpha - \sin \omega \cos \alpha$, $R_{T_{33}} = \cos \omega \cos \phi$, $R_{T_{14}} = T_x$, $R_{T_{24}} = T_y$, $R_{T_{34}} = T_z$, f is the focal length, c_x and c_y are the coordinates of the optical centre, ϕ , ω and α are the rotations around the x , y and z axis, respective and finally T_x , T_y , T_z are the translations on x , y and z directions.

The distortion and homography matrices coefficients are determined using the handball field lines. In order to perform this calculation, the user is requested to select points on the field lines and provide the corresponding real world coordinates.

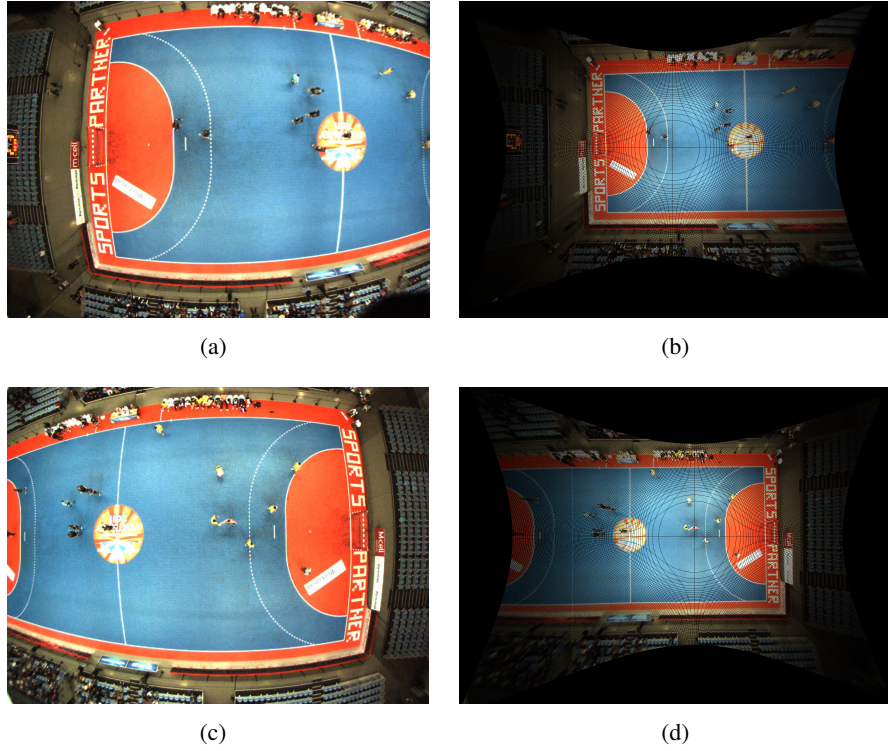


Figure 4.5: (a) and (b) left image before and after removing the barrel effect distortion. (c) and (d) right image before and after removing the barrel effect.

Since we are only interested in the players centre of mass position, only these coordinates are converted and not the entire field. Additionally, these coordinates are projected at an average's player height, which allows a more correct measure of the players' positions and enables better information fusion between cameras, which will be crucial for the tracking methodology when a player passes from one camera to the other.

4.3 Player Tracking

Player tracking is based on a vector of Kalman Filters (Kalman (1960); Welch and Bishop (2002)), one per player, with state x_k (Equation 4.7), measure z_k (Equation 4.8) and input u_k (Equation 4.9) at instant time k .

$$x_k = [x \ y]^T \quad (4.7)$$

$$z_k = [x \ y]^T \quad (4.8)$$

$$u_k = [v_x \ v_y]^T \quad (4.9)$$

Where x and y are the player's centre of mass position in real world coordinates, v_x and v_y are the player's velocity in real world coordinates.

The player's movement is modelled according to the following linear stochastic difference equations (Equations 4.10 and 4.11).

$$x_k = Ax_{k-1} + Bu_k + w_{k-1} \quad (4.10)$$

$$z_k = Hx_k + v_k \quad (4.11)$$

Where A represents the state model matrix, B the control input model and H is the observation model matrix. These matrices correspond to an identity matrix of size 2×2 . The random variables w_k and v_k represent the process and measurement noise.

The input to the system (u_k) is determined based on the players' subsequent positions according to Equation 4.12.

$$u_k(x, y) = \left(\frac{x_k - x_{k-1}}{\Delta t}, \frac{y_k - y_{k-1}}{\Delta t} \right) \quad (4.12)$$

The usage of real world coordinates allows a transparent tracking between the several video streams. Moreover, in the overlapped regions, two measures can be extracted from both images. The Kalman Filter deals with these two measures in a straightforward way because they are in real world coordinates.

Whenever the user indicates a player (with the mouse), a new Kalman Filter is added to the vector with the player's real world position and a default velocity of 0m/s. Afterwards, the players' locations on the subsequent frames are predicted using Equation 4.10. The area around the predicted measure corresponds to a region of interest (ROI) that is searched, according to the process explained on Section 4.2.3, to generate a measure (z_k) to update the estimate.

By predicting the position of the players on the subsequent frames it is possible to reduce the computational cost because only a few regions of the entire image are searched for players.

For the cases when the tracking is lost beyond a given configurable threshold (called Tracking Prediction Window - TPW), the system prompts the user to locate the player in the field. If subsequent tracking is successful, no further user actions are necessary and subsequent tracking is linked to previous history as a normal result of the Kalman Filter.

4.4 High Level Game Analysis

Using the outcome of the player tracking step it is possible to extract simple metrics that characterize the players' effort and preferred areas on the field. Based on this information, it is possible to combine knowledge and data to find patterns of higher levels of complexity.

Phase detection provides important information to evaluate how a team behaves, not only by demonstrating which team performs more attacks but also to understand how an attack is effective in terms of time.

The two following subsections describe the proposed methodologies used to detect the game phase and to characterize the defensive system a team is adopting.

4.4.1 Game Phase Analysis

The handball game is divided into four main phases: attack, defence, offensive-defensive transition and defensive-offensive transition ((Fuertes, 2010)). The attack and defence phases are considered positional game, while the offensive-defensive and defensive-offensive phases are considered transitional game and perform the connection between the two first phases.

In the attack phase the team which is in possession of the ball tries to create finalization situations that can lead to goal, while in the defence phase the defending team intends to take possession of the ball and avoid the opposing team of progressing and scoring.

As highlighted on (Perše et al., 2009), each game phase is characterized by distinct velocities and field positioning of the team's centre of mass. Transitional phases are characterized by higher speeds and few passes between players (Clanton and Dwight, 1997) and the game success can greatly depend on how fast players can perform a counter attack (defensive-offensive transitional phase) (Burger et al., 2013).

From a more systematic point of view on attack and defence situations, the team's overall velocity in the x direction, v_x , is relatively small, because the high displacements occur in the y direction, v_y , since the attacking team tries to break the defensive "barrier" while the defending team tries to keep it. Additionally, the teams are both concentrated near one of the goals.

On the other hand, during transitional phases the team's x position, p_x , can be in any zone between the two goals, but the team's longitudinal velocity, v_x , is characterized by a high module and a specific direction depending if the transition is between the defence and attack or the other way around.

Using the aforementioned premises a Fuzzy Logic (Zadeh, 1965) based classifier was developed in order to classify the game phase of an handball match. The choice for such a methodology is intrinsically related with the straightforward relation that exists between how a sports professional perceives the game which can be mapped into fuzzy "IF-THEN" rules, the inherent uncertainties which can be translated into the membership degrees and finally the easiness in changing the knowledge database.

The system uses as inputs the team's velocity on the x direction, v_x , the team's centre of mass (position) on the x direction, p_x , and the result of the previous detected phase, $prev_{phase}$. The classifier output indicates the current instant game phase based on an if-then inference engine as exemplified by Figure 4.6.

Using the previous phase as input into the classifier allows having an hysteresis and therefore the classifier output is more stable and more robust to temporary changes on the other two parameters.

The team's velocity and position on the x direction are determined based on the information stored on the log files and obeys to Equations 4.13 and 4.14, respectively.

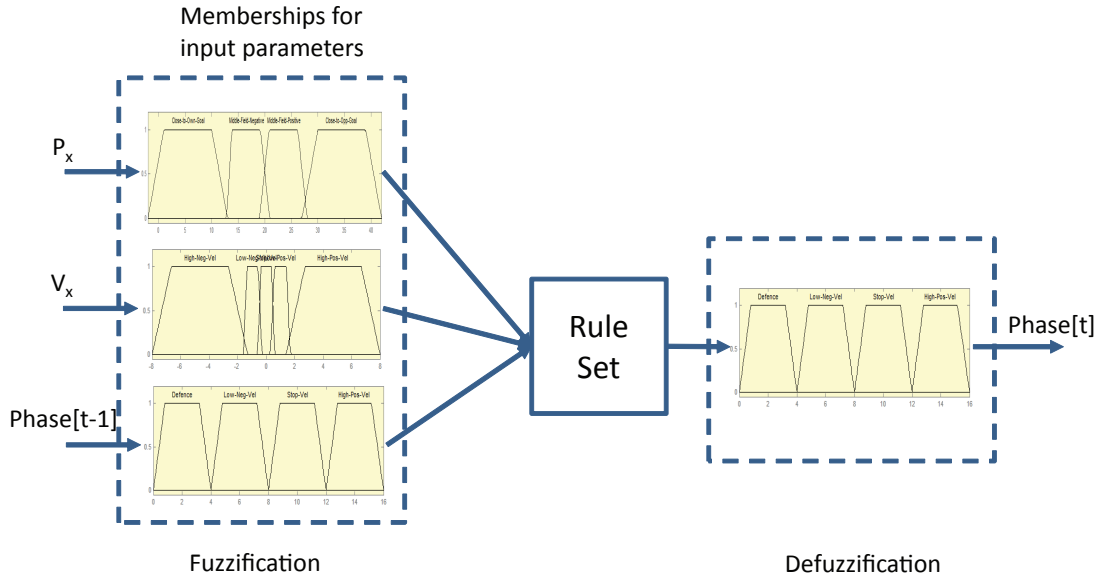


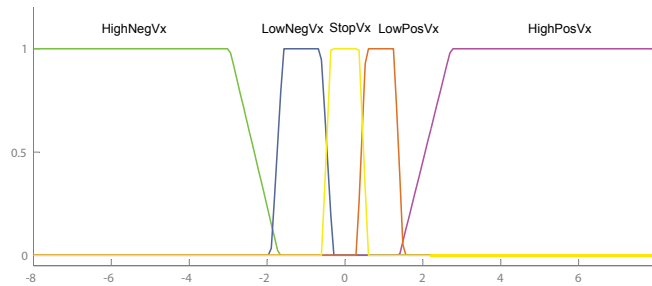
Figure 4.6: Fuzzy inference engine for game phase classification.

$$v_x = \frac{1}{N} \sum_{n=1}^N v_n x \quad (4.13)$$

$$p_x = \frac{1}{N} \sum_{n=1}^N p_n x \quad (4.14)$$

,where N is the number of field players currently playing.

The linguistic variable team's velocity (v_x) can be classified into five linguistic terms named *HighNegVx*, *LowNegVx*, *StopVx*, *LowPosVx* and *HighPosVx*, which are represented by the membership functions indicated in Figure 4.7.

Figure 4.7: Membership functions for $v_x = \{ \text{HighNegVx}, \text{LowNegVx}, \text{StopVx}, \text{LowPosVx}, \text{HighPosVx} \}$.

The linguistic terms *HighNegVx* and *HighPosVx* represent a large longitudinal displacement in a relatively short period of time and therefore represent high velocities, on the other hand the linguistic term *StopVx* corresponds to a very small velocity which means the team is stopped. *LowPosVx* and *HighPosVx* represent small team velocities.

The linguistic variable team's x position (p_x) can be classified into four linguistic terms named *Close2OwnGoal*, *MiddleFieldNeg*, *MiddleFieldPos* and *Close2OppGoal*, which are represented by the membership functions indicated in Figure 4.8.

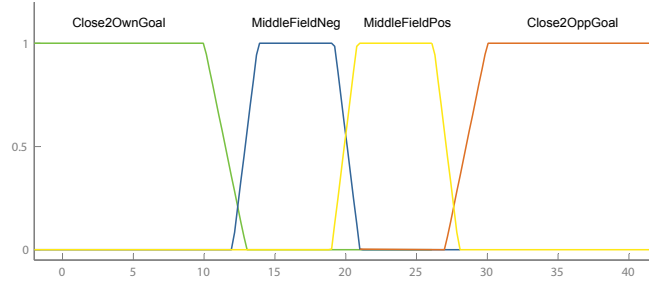


Figure 4.8: Membership functions for $p_x = \{ \textit{Close2OwnGoal}, \textit{MiddleFieldNeg}, \textit{MiddleFieldPos}, \textit{Close2OppGoal} \}$.

As the names of the linguistic terms indicate and the previous image illustrates (Figure 4.8), *Close2OwnGoal* and *Close2OppGoal* represent field areas near each team goal, while *MiddleFieldNeg* and *MiddleFieldPos* represent areas near the middle field line.

The membership functions parameters for the two previous linguistic variables were determined using information obtained from an excerpt of a game that was previously tagged by a sports expert.

Finally the linguistic variable corresponding to the previous phase ($prev_{phase}$) can be classified into four linguistic terms that correspond to the four game phases: *Defence*, *DefOff*, *OffDef* and *Attack*, which are represented by the membership functions indicated in Figure 4.9.

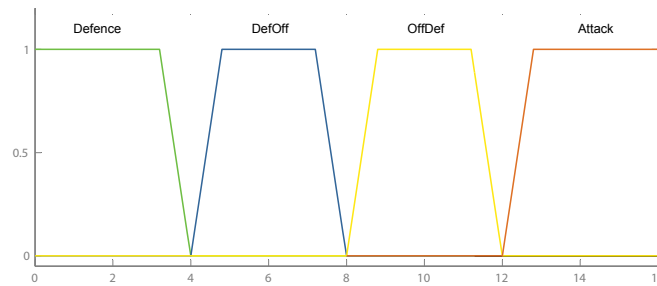


Figure 4.9: Membership functions for $prev_{phase} = \{ \textit{Defence}, \textit{DefOff}, \textit{OffDef}, \textit{Attack} \}$.

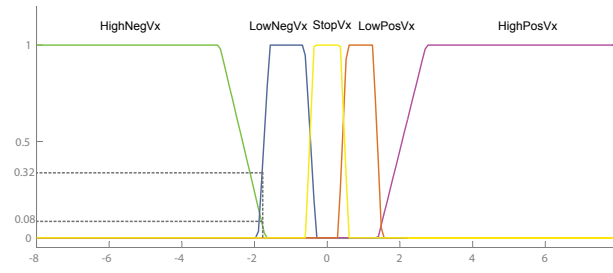
The first step of the Fuzzy-Ruled based classifier consists in the fuzzification of each input. Therefore, each crisp input (team's position and velocity on the x direction and previous game phase) is fuzzified in order to obtain its membership degree to each membership function.

Considering the team position $p_x = 27.5m$, velocity $v_x = -1.8m/s$ and that previously the team was in Attack phase, the following membership belonging degrees are obtained (Table 4.3).

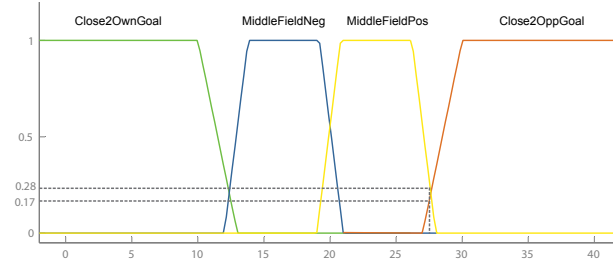
Table 4.3: Belonging degrees to each membership term.

v_x	p_x	phase
$\mu_{v_x=HighNegVx}(-1.8) = 0.08$	$\mu_{p_x=Close2OwnGoal}(27.5) = 0.0$	$\mu_{phase=Defence}(Attack) = 0.0$
$\mu_{v_x=LowNegVx}(-1.8) = 0.32$	$\mu_{p_x=MiddleFieldNeg}(27.5) = 0.0$	$\mu_{phase=DefOff}(Attack) = 0.0$
$\mu_{v_x=StopVx}(-1.8) = 0.0$	$\mu_{p_x=MiddleFieldPos}(27.5) = 0.28$	$\mu_{phase=OffDef}(Attack) = 0.0$
$\mu_{v_x=LowPosVx}(-1.8) = 0.0$	$\mu_{p_x=Close2OppGoal}(27.5) = 0.17$	$\mu_{phase=Attack}(Attack) = 1.0$
$\mu_{v_x=HighPosVx}(-1.8) = 0.0$		

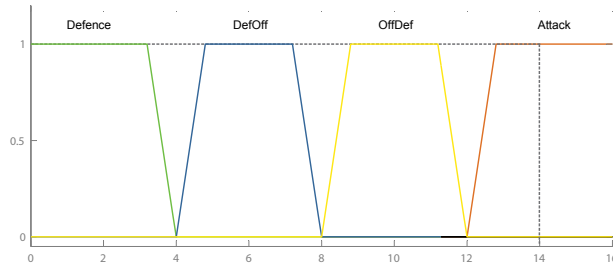
The following image (Figure 4.10) illustrates how these values were obtained.



(a)



(b)



(c)

Figure 4.10: Membership belonging degrees when: (a) $v_x = -1.8m/s$ (b) $p_x = 7.8m$ and (c) $phase = Attack$.

Afterwards, each input is combined according to the Fuzzy "IF-THEN" rules defined on Tables 4.4, 4.5, 4.6 and 4.7, which were obtained taking into account the sports expert knowledge. Due to the large number of rules they were subdivided into groups so that they can be more easily read.

Table 4.4 contains the rules that are applied when the previous phase is *Defence*. In this case, the output of the rules can only be *Defence* or *DefOff*, since due to imposed physical restrictions a team cannot pass from *Defence* to *Attack* without passing through *DefOff* phase. In case of a fast counter-attack then the time spent by the team on the *DefOff* phase will be very small.

Table 4.4: Game phase Fuzzy inference rule matrix when the Previous Phase is Defence.

Rule	PrevPhase	TeamPosX	TeamVelX	Phase
1.1	<i>Defence</i>	<i>Close2OwnGoal</i>	<i>HighNegVx</i> OR <i>LowNegVx</i> OR <i>StopVx</i> OR <i>LowPosVx</i>	<i>Defence</i>
1.2			<i>HighPosVx</i>	<i>DefOff</i>
1.3		<i>MiddleFieldNeg</i>	<i>HighNegVx</i> OR <i>LowNegVx</i> OR <i>StopVx</i> OR <i>LowPosVx</i>	<i>Defence</i>
1.4			<i>HighPosVx</i>	<i>DefOff</i>
1.5		<i>MiddleFieldPos</i>	<i>HighNegVx</i> OR <i>LowNegVx</i> OR <i>StopVx</i>	<i>Defence</i>
1.6			<i>LowPosVx</i> OR <i>HighPosVx</i>	<i>DefOff</i>

Similarly, when at the previous time instant the team is attacking (*Attack* phase), depending on the values of the team's velocity and position on the longitudinal direction of the field only two phases are allowed, either *Attack* or *OffDef*. The rules in this case are summarized on Table 4.5.

Table 4.5: Game phase Fuzzy inference rule matrix when the Previous Phase was Attack.

Rule	PrevPhase	TeamPosX	TeamVelX	Phase
1.7	<i>Attack</i>	<i>Close2OppGoal</i>	<i>HighNegVx</i>	<i>OffDef</i>
1.8			<i>LowNegVx</i> OR <i>StopVx</i> OR <i>LowPosVx</i> OR <i>HighPosVx</i>	<i>Attack</i>
1.9		<i>MiddleFieldPos</i>	<i>HighNegVx</i> OR <i>LowNegVx</i> OR <i>StopVx</i>	<i>OffDef</i>
1.10			<i>LowPosVx</i> OR <i>HighPosVx</i>	<i>Attack</i>
1.11		<i>MiddleFieldNeg</i>	<i>HighNegVx</i>	<i>OffDef</i>
1.12			<i>LowNegVx</i> OR <i>StopVx</i> OR <i>LowPosVx</i>	<i>Attack</i>
1.13			<i>HighPosVx</i>	<i>DefOff</i>

When the team is currently in a transitional phase then the number of rules increases because on these situations the team's next phase can be any of the other three.

Following the same line of thought, the game can only evolve to a phase that is adjacent to it, physically speaking. Therefore if the previous phase is *OffDef* it is not allowed for the game to pass immediately to *Attack*, first it must pass through a *DefOff* phase, even if only for a small time period and afterwards it can naturally evolve to the *Attack* phase. For the *DefOff* case the same stands, so the team must first switch to the *OffDef* phase and only then to the *Defence* phase.

The two following tables, Tables 4.6 and 4.7, show the "IF-THEN" rules when the previous phase is *DefOff* or *OffDef*, respectively.

Table 4.6: Game phase Fuzzy inference rule matrix when the Previous Phase was *DefOff*.

Rule	PrevPhase	TeamPosX	TeamVelX	Phase
1.14	<i>DefOff</i>	<i>Close2OwnGoal</i>	<i>HighNegVx</i> OR <i>LowNegVx</i>	<i>OffDef</i>
1.15			<i>StopVx</i> OR <i>LowPosVx</i> OR <i>HighPosVx</i>	<i>DefOff</i>
1.16		<i>MiddleFieldNeg</i>	<i>HighNegVx</i>	<i>OffDef</i>
1.17			<i>LowNegVx</i> OR <i>StopVx</i> OR <i>LowPosVx</i> OR <i>HighPosVx</i>	<i>DefOff</i>
1.18		<i>MiddleFieldPos</i>	<i>HighNegVx</i>	<i>OffDef</i>
1.19			<i>LowNegVx</i> OR <i>StopVx</i> OR <i>LowPosVx</i>	<i>Attack</i>
1.20			<i>HighPosVx</i>	<i>DefOff</i>
1.21		<i>Close2OppGoal</i>	<i>HighNegVx</i>	<i>OffDef</i>
1.22			<i>LowNegVx</i> OR <i>StopVx</i> OR <i>LowPosVx</i>	<i>Attack</i>
1.23			<i>HighPosVx</i>	<i>DefOff</i>

Table 4.7: Game phase Fuzzy inference rule matrix when the Previous Phase is *OffDef*.

Rule	PrevPhase	TeamPosX	TeamVelX	Phase
1.24	<i>OffDef</i>	<i>Close2OwnGoal</i>	<i>HighNegVx</i>	<i>OffDef</i>
1.25			<i>LowNegVx</i> OR <i>StopVx</i> OR <i>LowPosVx</i>	<i>Defence</i>
1.26			<i>HighPosVx</i>	<i>DefOff</i>
1.27		<i>MiddleFieldNeg</i>	<i>HighNegVx</i> OR <i>LowNegVx</i> OR <i>StopVx</i> OR <i>LowPosVx</i>	<i>OffDef</i>
1.28			<i>HighPosVx</i>	<i>DefOff</i>
1.29		<i>MiddleFieldPos</i>	<i>HighNegVx</i> OR <i>LowNegVx</i> OR <i>StopVx</i>	<i>OffDef</i>
1.30			<i>LowPosVx</i> OR <i>HighPosVx</i>	<i>DefOff</i>
1.31		<i>Close2OppGoal</i>	<i>HighNegVx</i> OR <i>LowNegVx</i>	<i>OffDef</i>
1.32			<i>StopVx</i> OR <i>LowPosVx</i> OR <i>HighPosVx</i>	<i>DefOff</i>

Once the inputs have been fuzzified and their membership degrees determined, each rule antecedent is evaluated using the minimum for the AND operator and the maximum for the OR operator. The result of the antecedent evaluation can then be applied to the consequent using a clipping method.

Finally the outcome of every rule is aggregated and the defuzzification is accomplished using the mean of the maximum (MOM) value obtained in the consequent.

For the cases with only one maximum value then the output crisp value will be the centre of the corresponding membership function: 2 for Defence, 6 for DefOff, 10 for OffDef and 14 for Attack.

In case of more than one maximum, it is possible that the output of the classifier is not a valid class (for example a value of 4). For those cases the current phase is considered to be the previous one. The result of the classifier is still filtered using a median filter of size 9.

Phase categorization is only the first step on the game analysis process. Once the phases are determined a wide range of more in depth analysis is opened which may include classification of the defensive/offensive formations, event detection, team tactics, among others.

4.4.2 Defensive Systems Analysis

Defensive systems can be classified into three major groups: man-to-man, zone and combined (Czerwinski and Taborsky, 2005).

On man-to-man defensive systems each defender is assigned to a specific opponent player, on the other hand on zone defensive systems each player is responsible for guarding a given zone from any attacker. Combined defensive systems include aspects of both man-to-man and zone defensive systems, which means that one or more players perform man-to-man defence, usually to guard either the pivot or the opponent's team strongest player while the rest of the team plays in zone defence and is responsible for defending a particular area.

The most popular zone defence systems are (Gomes, 2008): 6:0, 5:1, 1:5, 4:2, 3:3 and 3:2:1, while on combined systems teams usually choose 5+1 or 4+2.

Figure 4.11 illustrates the characteristics of the most common zone defence systems.

As it is possible to verify, each system receives its name according to the disposition of the players around their own goal area. So, for 6:0 systems all the 6 fielders form a line right after the 6 meters line, while for 4:2 systems, 4 fielders form a line right after the 6 meters line, while the other 2 form a second line near the 9 meters line.

Taking into account the characteristic highlighted, a methodology for classifying the team's defensive system was developed. This methodology consists in determining which system best fits the team's current position on the field, therefore, for each defensive system the radius of every defensive line is determined.

The lines' radius for each system are determined by averaging the minimum distance of the players to their own goal. Since the players disposition is not a perfect circle around the goal, but rather two semi-circumferences and a straight line, this minimum distance is determined according to the following equation (Equation 4.15):

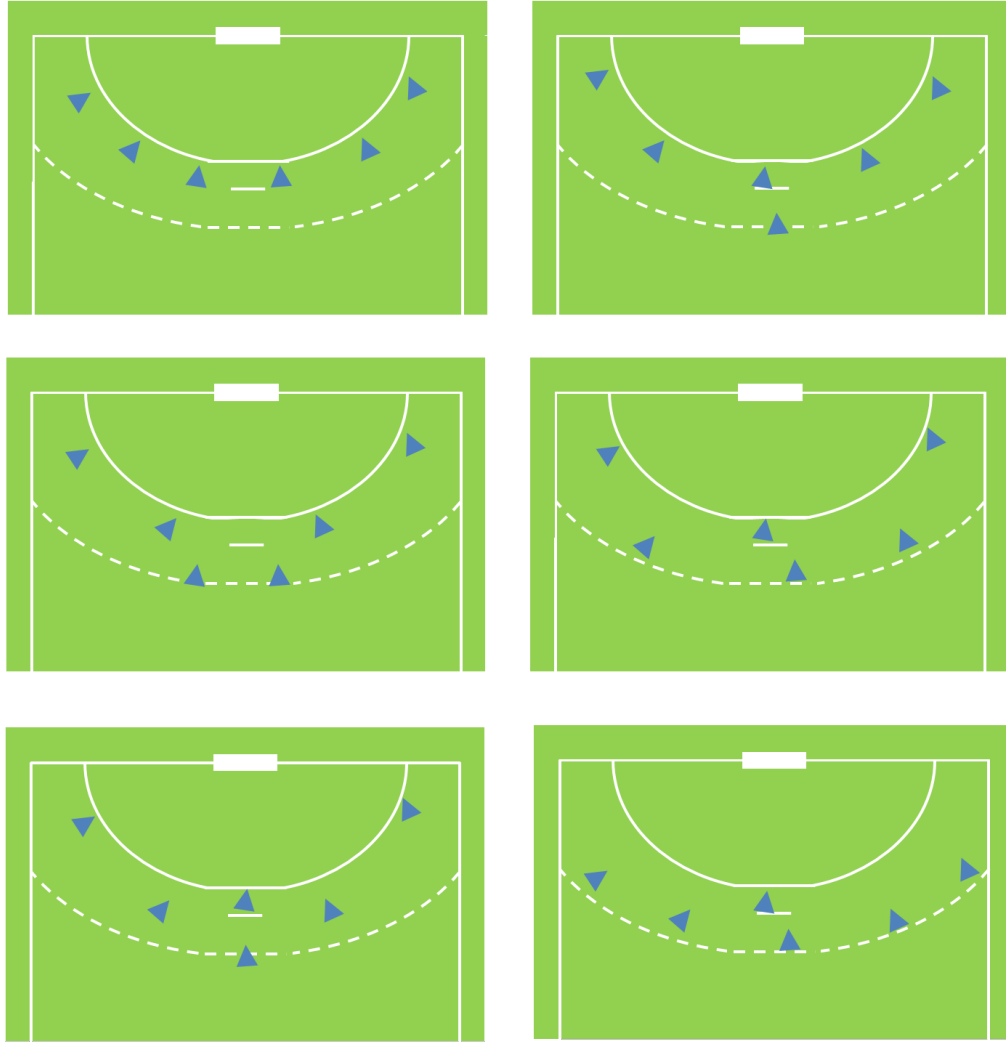


Figure 4.11: Most common defensive systems, from top right to bottom left 6:0, 5:1, 4:2, 3:3, 3:2:1 and 1:5.

$$p_{min_{dist}} = \begin{cases} \sqrt{(p_x - p_{Goalx})^2 + (p_y - 1.5)^2}, & \text{if } p_y > 1.5 \\ \sqrt{(p_x - p_{Goalx})^2 + (p_y - (-1.5))^2}, & \text{if } p_y < -1.5 \\ abs(p_x - p_{Goalx}) & \end{cases} \quad (4.15)$$

Where, p_x and p_y are the players' coordinates on the field and p_{Goalx} is the x coordinate of the goal (-20m or +20m).

Therefore if the player vertical position p_y on the field is not within the goal limits (-1.5m and 1.5m) the distance is calculated to the nearest pole of the goal otherwise it is calculated as only the horizontal distance as shown in Figure 4.12.

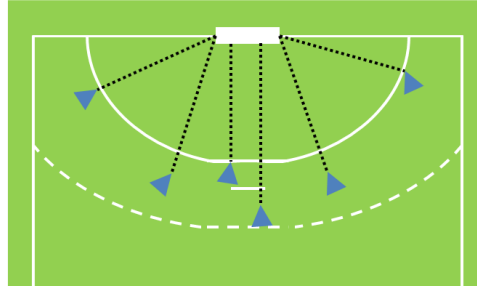


Figure 4.12: Example of how the minimum distance between the goal and the players is determined.

For systems with only one defensive line (6:0) the radius of the line is determined by averaging all the six fielders' minimum distances to the goal. For systems with two or three defensive lines, players are first ordered by their minimum distance to the goal, afterwards players farther away from the goal are grouped into a same line, for example, in the 5:1 system this group will be composed of a single player, while on 1:5 systems it will be composed of 5 players. Players close to the goal will be grouped into the other line.

On three line systems (3:2:1) an intermediate line is also determined and so, the player farther away from the goal forms the 1 line player, the 2 subsequent players form the 2 line player and the remainder form the 3 line player.

Once the players are grouped into their respective lines, the several line radius are determined by averaging the minimum distance of the players corresponding to each line.

To determine which system the team is currently adopting, the next step on the algorithm consists in analysing the root mean square error between each system line and the corresponding players' positions according to Equation 4.16.

$$MSE_S = \sqrt{\frac{1}{N} \sum_{l=1}^{L_{max}} \sum_{j=0, j \in l}^N (minDis_j - radius_l)^2} \quad (4.16)$$

Where $minDis_j$ is the minimum distance of the player to the goal and $radius_l$ is the radius of the obtained defensive line.

Finally, the systems are ordered by their root mean square error, and the one with the least mean square error that assures a minimum distance of lines of at least 1.5m is the one that categorizes the current distribution of the players on the defensive situation.

4.5 Game Model

Under the scope of this thesis, the game model is considered a mathematical representation of the game that allows to characterize the interactions between the parties involved (players and ball) throughout game evolution.

The complex characteristics of the handball game and the different levels of detail make Hierarchical Coloured Petri Nets an appropriate modelling language. Moreover, the colour property

of the net allows tokens to have properties such as player number plus team identification, which reduces the number of nets that need to be implemented, and the hierarchical structure allows not only to have nets with multiple layers of detail where a simplified net gives a broader view of the system and the other subnets provide more detailed information but also to easily include/remove extra detail to the net.

The proposed Hierarchical Coloured Petri Nets is composed of 3 layers with different levels of abstraction according to Figure 4.13. The first layer, **Game Mode**, gives a generic overview of the handball game; the second layer specifies the characteristics of the game when the ball is on game, **Game On**, and when the game is stopped, **Game Stop**, either due to a foul or to the specific rules of the game. The deepest layer is more concerned with the players' state, and models the interactions among players, as well as between them and the ball.

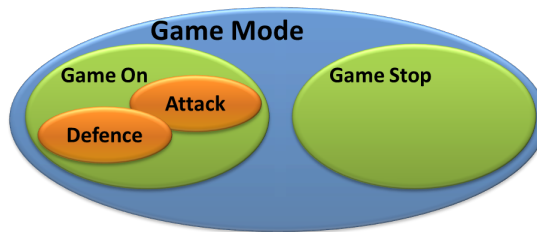


Figure 4.13: Model hierarchy overview.

The colour sets defined are illustrated on Table 4.8.

Table 4.8: Defined colour sets.

Colour Set Definition	Description
colset GameTime = int ;	Colour set that represents the game time.
colset Transition = bool ;	Colour set that represents a variable used to enable/disable a transition.
colset Team = with Team TOpp;	Colour set that indicates the team. It can assume the values Team or TOpp.
colset Ball = bool ;	Colour set that represents the ball. It can assume two values <i>true</i> or <i>false</i> .
colset PlayerID = int ;	Colour set that represents the players' IDs (note: the ID will most likely not correspond to the number on the player's vest).
colset Player = product PlayerID * Team;	Compound colour set to represent a player. This colour set is composed of a player number (ID) and a team.
colset PlayerBall = product Player * Ball declare ms;	Compound colour set, composed of a player and a ball.
colset TeamPlayers = product Player * Player * Player * Player * Player * Player;	Compound colour set that represents the active players of a team.
colset TeamBall = product Team * Ball * TeamPlayers;	Compound colour set that represents a team with their players and ball.

The hierarchical model is achieved using substitution places (Huber et al. (1991)). Depending on the detail level of each layer, the tokens involved are represented differently. Therefore, layers at the second level include tokens of colour sets *Team* or *TeamBall*. Tokens on the third level, besides the colour sets from the second level, can also be of colour sets *Ball*, *Player* or *PlayerBall*.

Figure 4.14 corresponds to the first layer of the hierarchical coloured Petri model. The handball game starts with a **Throw-Off**, which leads to the subnet represented by the substitution transition **Game ON**.

The game will remain on the **Game On** subnet (details are given on Section 4.5.1) as long as the two teams are fighting over the ball. Whenever there is a foul, a goal, the ball goes outside the game area, a time out is requested or the match reaches the break the state will change into the **Game Stop** subnet represented by the **Game Stop** substitution transition (Section 4.5.2). As soon as the corresponding throw is performed the ball is again being fought over and the game changes into the **Game On** subnet.

When the game reaches the full time, which is monitored by variable *dGameTime*, the game state moves into the **End of Game** state.

The inscriptions on the arcs indicate that every transition moves two teams from one place to another. As described on Table 4.8, a *TeamBall* colour set is composed of a *Team*, which can assume one of two values (Team or TOpp), that are stored on the *dTeam* and *dTeam2* variables; a *Ball* that can assume a true or false value depending if the respective team is in possession of the ball or not and a set of *TeamPlayers* that is characterized by the product of seven *Players*, that have a *PlayerID* and belong to a given *Team*.

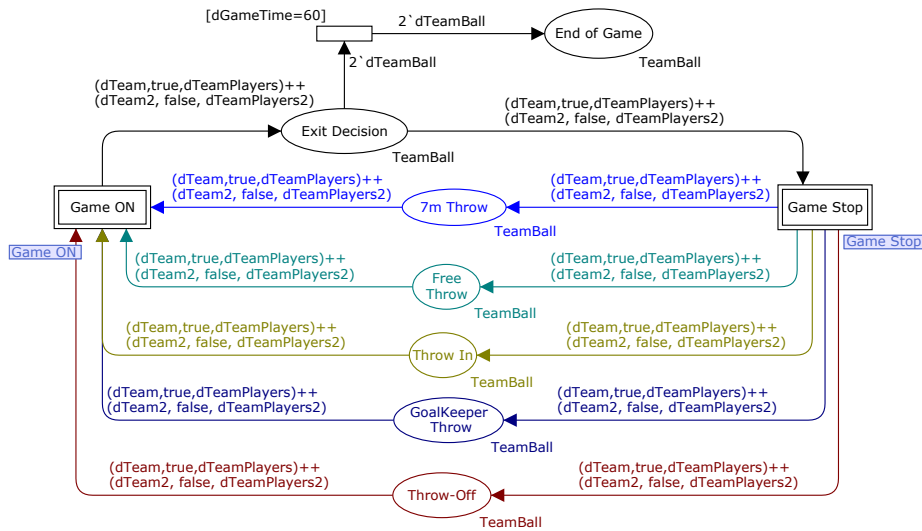


Figure 4.14: First layer of the game model.

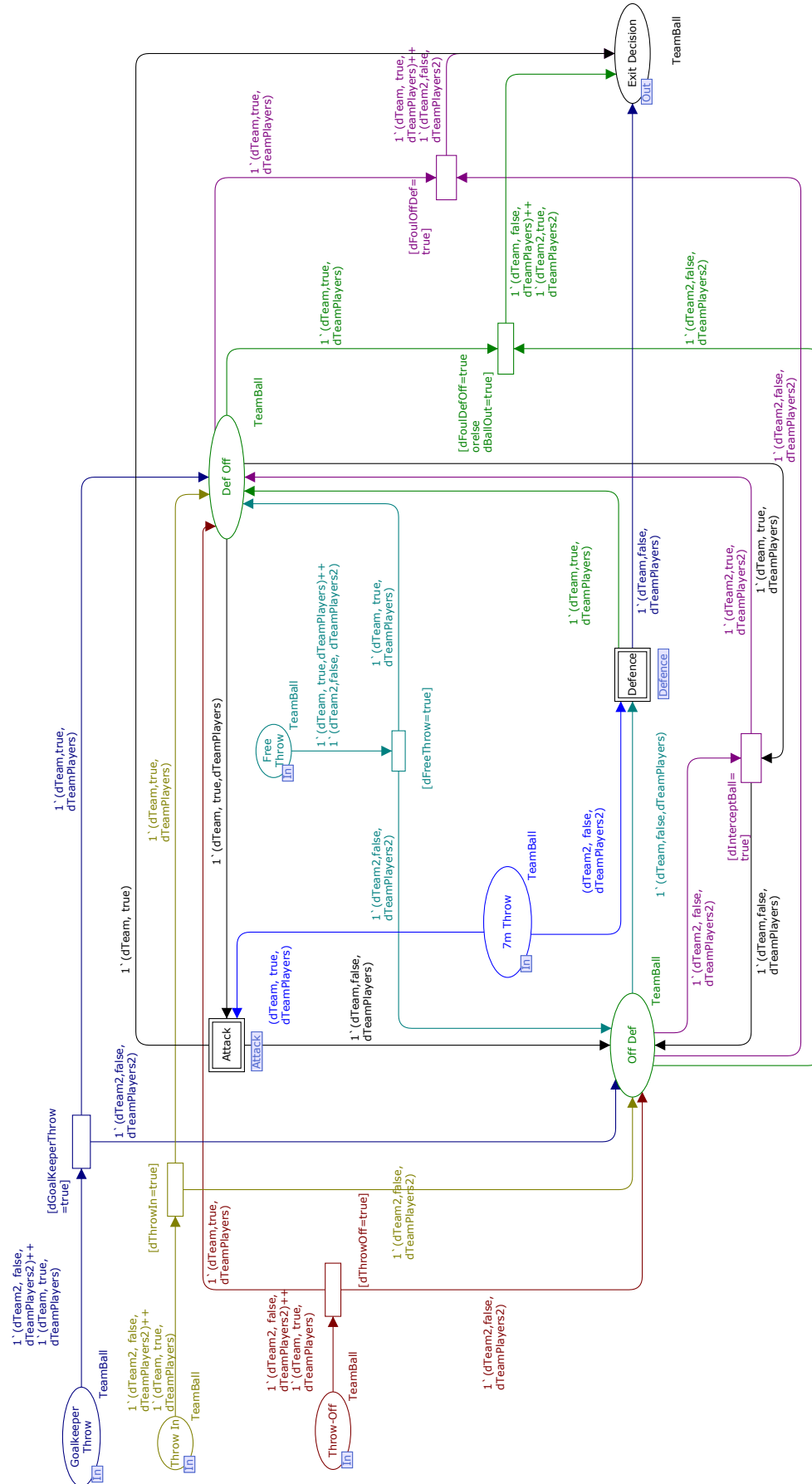


Figure 4.15: Subnet with the model of the Game On subnet.

4.5.1 Game ON Subnet

As stated, the **Game On** subnet is active whenever the ball is being fought over by the teams. This subnet models the behaviour of the global team, and so it provides information about the team game phase, which is represented by two substitution transitions, **Attack** and **Defence**, and two states, **Off Def** and **Def Off**, among other states and transitions as illustrated in Figure 4.15.

The transition between game phases is ruled not only by the team's ball possession, but also by the position of the players on the field.

Whenever the team is in possession of the ball and assumes an attack position near the opponent goal, it transits to the **Attack** game phase.

During the attack phase two major situations may occur: either the opponent team intercepts the ball and therefore goes into a transitional phase (**Off Def** state) or an event may lead the game to stop (**Game Stop** subnet) and the team may either remain in the **Attack** game phase or switch into the **Off Def** phase. This decision is performed in the **Game Stop** subnet (Section 4.5.2), via the output port **Exit Decision**.

In case the team passes into the **Off Def** state, it will remain on it until it reaches a defensive position (**Defence** substitution transition), it intercepts the ball and passes immediately into the **Def Off** state or, again, a **Game Stop** state is reached.

In the **Defence** phase, the only two possible states to move on are the **Def Off** or the **Game Stop** state via the **Exit Decision** output port. The team will move into the **Def Off** state in case it intercepts the ball or to the **Exit Decision** output port in case of a foul, the ball goes outside the field's limits, among other options that will be addressed on Section 4.5.1.1.

Once in the **Def Off** state, the team can transit into the **Off Def** state in case the other team intercepts the ball or into an **Attack** position if it reaches the goal area. Like on the **Off Def** state it can also move into the **Exit Decision** state.

Additionally, there are other states that correspond to entry points (input ports) to this subnet from the **Game Stop** subnet, that are **Prepare 7m Throw**, **Free Throw**, **Throw Off**, **Throw In** and **GoalKeeper Throw**. As already referred, the **Exit Decision** output port connects the **Game On** subnet to the **Game Stop** subnet.

The two following sections provide more details about the **Attack** and **Defence** substitution transitions.

4.5.1.1 Defence Subnet

As explained on Section 4.5, the third layer of the hierarchical model details the most important interactions between players. At each time only one team can be in the **Defence** state, therefore all tokens in this state must belong to the same team, which means that variable *dTeam* and variables of colour set *Player* (*dPlayer1..7*) or colour set *TeamPlayers* (*dTeamPlayers*) can only assume the colour *Team* or *TOpp*.

The **Defence** sub state describes the interactions when the players assume a defensive position as can be seen in Figure 4.16. When a player is defending he/she can be in three different states:

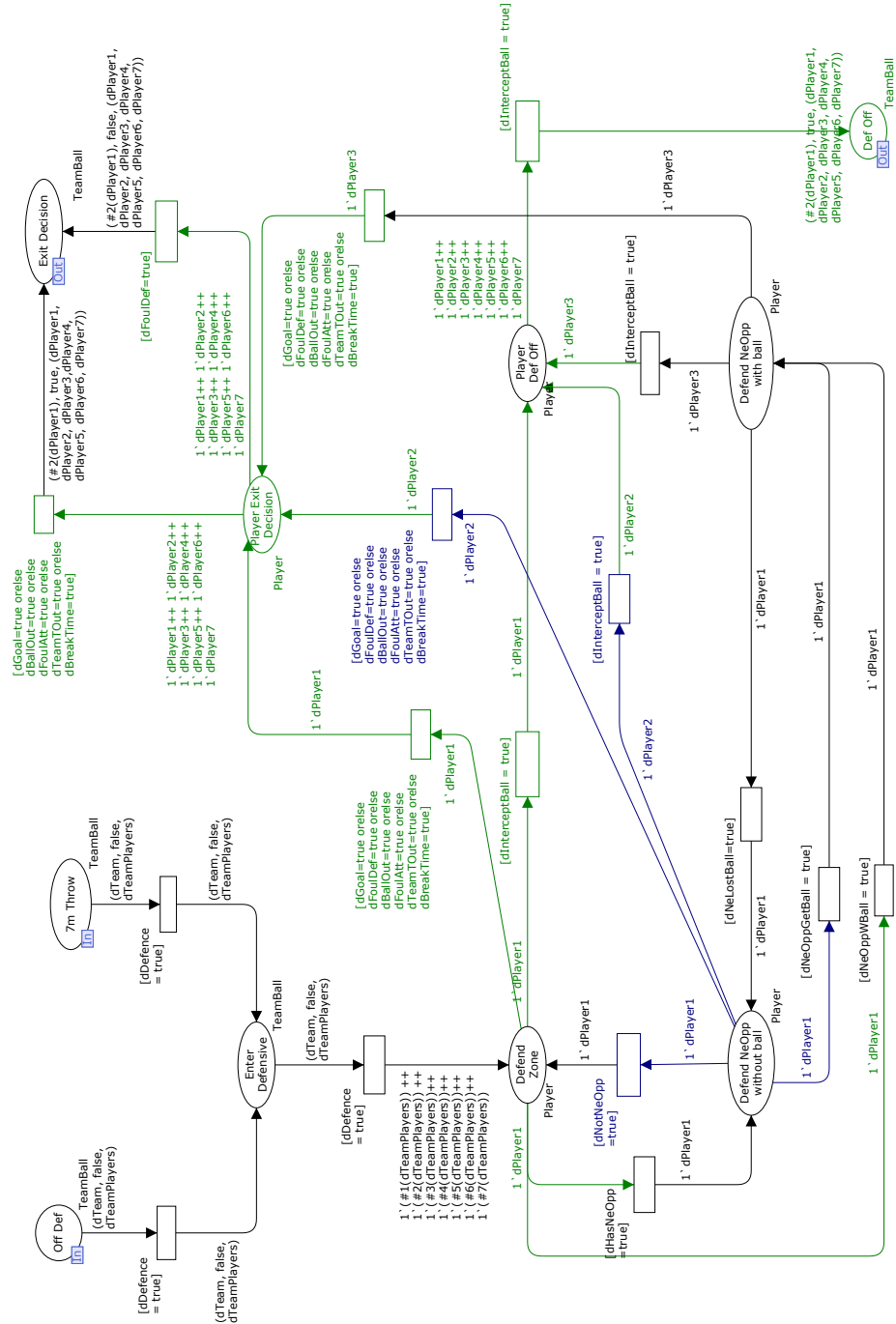


Figure 4.16: Model of the Defence subnet.

- **Defend Zone** – defending an area where there is no near opponent;
- **Defend NeOpp Without Ball** - defend the goal against an opponent that is near;
- **Defend NeOpp With Ball** - defending the goal and pass lines against an opponent that is in possession of the ball.

The transitions between these states are dependent on the movements of the players from both teams and the ball.

There are two input ports that allow reaching the **Defence** state, the **Off Def**, in case the opponent team intercepts the ball or when there was a foul performed by this same team that leads to the team losing the ball and the opponent team performing a **Prepare 7m Throw**.

Since there is more than one entry point an intermediate state needed to be created (**Enter Defensive**), that is reached whenever the *dDefence* variable is true. This variable can in future be automatically updated using the information obtained from the game phase detection method (Section 4.4.1). Once the team reaches this state, all players are considered to be defending an area and therefore, are all transferred into the **Defend Zone** state.

As soon as a player starts blocking the attack of an opponent without ball, which is detected by variable *dHasNeOpp* getting the *true* value, the respective player (characterized by a team and an ID) is moved to state **Defend NeOpp Without Ball**. In case the defender starts defending an opponent that has the ball (pointed out by variable *dNeOppWBall*), then the movement will be into the **Defend NeOpp with Ball** state.

In case the opponent that is being guarded is no longer in possession of the ball (variable *dNeLostBall* changed to true), the player that is in the **Defend NeOpp with Ball** will change into the **Defend NeOpp Without Ball**. Moreover, she/he can further move into the **Defend Zone** state if she/he is no longer actively defending the goal against an opponent (variable *dNotNeOpp*).

During the course of a defensive state, any defender can intercept the ball (variable *dInterceptBall*) and so all the defender players will move, temporarily into the **Player Def Off** state and afterwards into the output port **Def Off**.

In case there is a goal by the opponent team (*dGoal*), a foul (*dFoulDef* or *dFoulAtt*), the ball goes outside the playing area (*dBallOut*), a time out is requested by the opponent team (*dTeamTOut*) or a break time is reached (*dBreakTime*) the game passes to the **Player Exit Decision**, which depending on the action that triggered the transition defines if the team moves into the **Exit Decision** output port in possession of the ball or not. This state is the link to the **Game Stop** subnet (Section 4.5.2).

4.5.1.2 Attack Subnet

The Attack sub state (Figure 4.17) can be entered via two input ports. The first one is after a defensive offensive state (**Def Off**) that results in a ball progression such that the team reaches the opponent's goal which is detected by the *dAttack* variable. The other port results from the **Prepare 7m Throw** state.

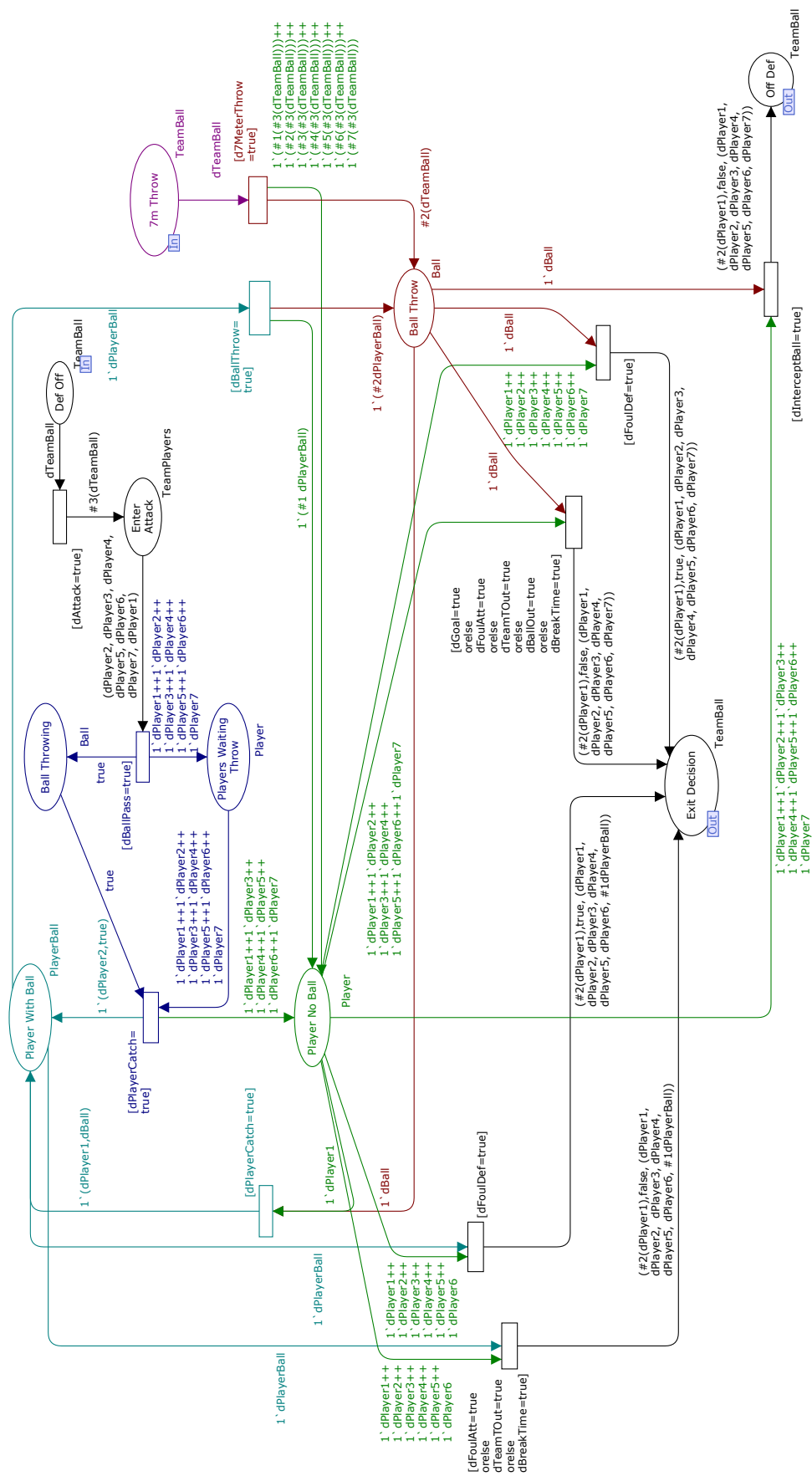


Figure 4.17: Model of the Attack subnet.

The proposed model includes the interactions with the ball which leads to players reaching the **Player With Ball** state in case they own the ball or the **Player No Ball** state otherwise. Only one player can be at any given time in the **Player With Ball** state, which is guaranteed by the input arcs guards ($1'(dPlayer1, dBall)$ and $1'(dPlayer2, true)$) and by the **Ball Throwing** and **Ball Throw** states.

When the player with the ball launches it (which can either be as a trigger for a pass to a team mate or for a throw to the opponent goal), the ball token goes into the **Ball Throw** state while the player that performed the launch goes into the **Player No Ball** state.

While the ball is in the **Ball Throw** state three different events may occur that lead to a change on the net state:

- a team mate successfully intercepts the ball and variable $dPlayerCatch$ gets true. This triggers a transition and the player who caught the ball along with the ball token moves into the **Player With Ball** state;
- the ball is intercepted by an opponent player (variable $dInterceptBall$) and all the players from the team pass from the **Player No Ball** state to the **Off Def** state. The ball exits the **Ball Pass** state to the **Def Off** state as can be seen on the **Defence** subnet (Figure 4.16);
- the ball launch consisted in a throw that resulted on a goal ($dGoal$), a player from either teams performed a foul ($dFoulAtt$ or $dFoulDef$), the attacking team required a time out ($dTeamTOut$), the ball went outside the allowed playable area ($dBallOut$), or a break time was reached ($dBreakTime$), which requires a decision about the next state. In this case the team passes into the **Exit Decision** state that is dealt at the **Game Stop** subnet (please refer to Section 4.5.2).

Finally, while the ball is being carried by a player other events may occur that also lead to the **Exit Decision** state. These events include a foul ($dFoulAtt$ or $dFoulDef$), the team requested a time out ($dTeamTOut$) or the time to a break was reached ($dBreakTime$).

4.5.2 Game Stop Subnet

The **Game Stop** subnet is more complex because, depending on the condition that lead the game to the **Exit Decision** input port, several paths may be followed. Figure 4.18 illustrates the game model for this situation.

The game always starts on the **Break** state and the $dEndBreak$ variable switching to true signals the consent to perform the **Throw-Off** that will lead the game to the **Game On** state.

When the game evolves again to the **Game Stop** substitution transition the following may occur:

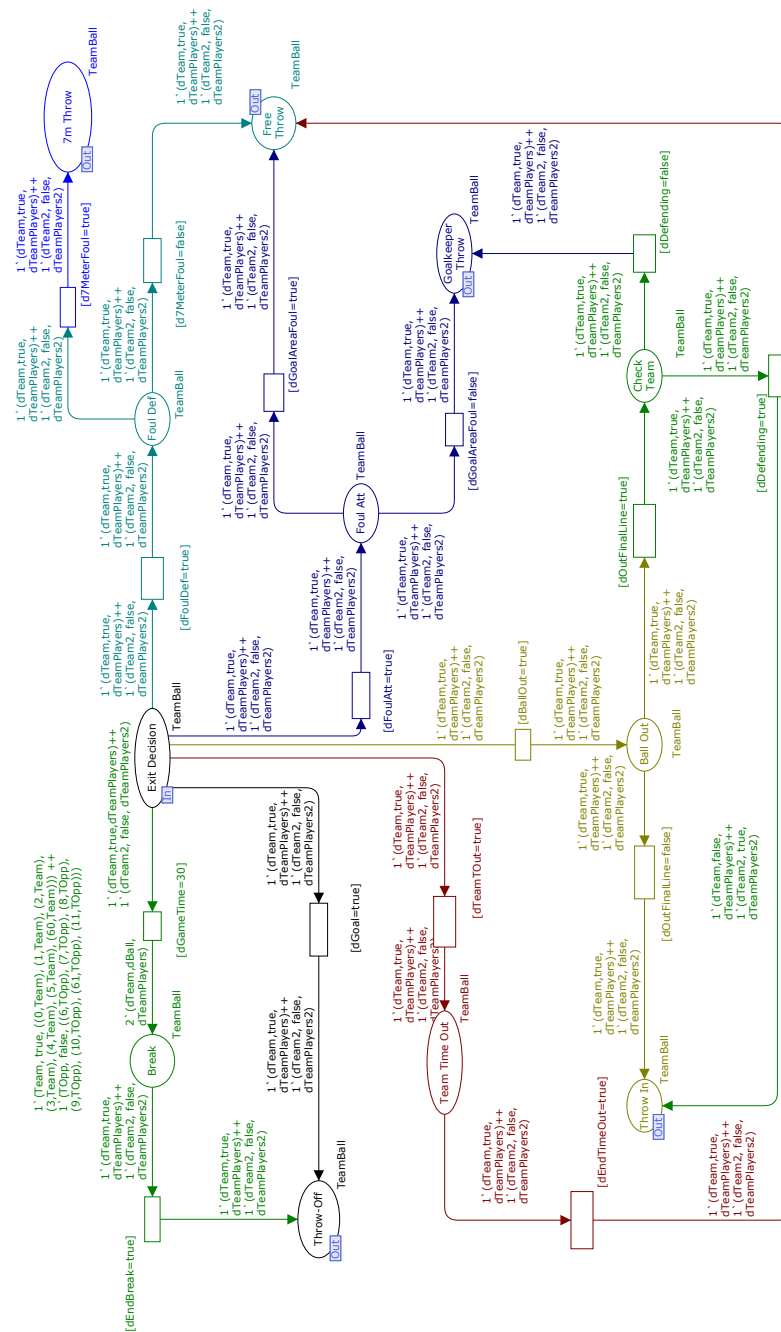


Figure 4.18: Model of the Game Stop subnet.

- In case one of the teams scores a goal, indicated by the *dGoal* variable, both teams must move into their own half of the court and so they pass to the **Prepare for Throw Off** state. Once the teams are ready and the referee whistles the game passes to the **Game On** subnet via the **Throw-Off** output port. As already seen on Section 4.5.1, the team with the ball steps into the **Def Off** state, while the other team steps into the **Off Def** state.
- In case of a defensive foul, both teams will go into the **Foul Def** state so that a decision can be made on whether it was a minor or a major foul. If a major foul was performed, then variable *d7MeterFoul* is set to true and both teams must prepare themselves for a 7 meter throw (**Prepare 7m Throw** state), otherwise they prepare for a free throw (**Prepare Free Throw** state), where both teams keep their previous game phase (recall Figure 4.15) and exit to the **Game On** subnet via the **Free Throw** output port.

Major fouls imply that the defensive team performs a foul that clearly destroys the chance for the attacking team to score, the goalkeeper carries the ball back into the team's own goal area, a court player intentionally plays the ball back to goalkeeper that is still in the goal area, or a defensive player enters the goal area to gain an advantage over an attacking player in possession of the ball.

- Attacking fouls can also be evaluated based on whether the player committed the foul inside the opponent's goal area or not. In case the foul is committed by the attacking player entering the opponent's team goal area, or touching the ball inside the opponent's goal area then variable *dGoalAreaFoul* is true and both teams must prepare for a goalkeeper throw (*Prepare GoalKeeper Throw*).

The **Goalkeeper Throw** is taken by the goalkeeper from the goal area out over the goal area line and is considered to be finished when the ball completely crosses the goal area line.

In case the foul does not justify a goalkeeper throw (variable *dGoalAreaFoul* is false), then both teams prepare themselves for a free throw (**Free Throw**).

- If the ball goes outside the court, then variable *dBallOut* is true and the next state for both teams is **Ball Out**. In case the ball exits the court through a lateral line then a throw is awarded to the opponent team and both teams move into the **Prepare for Throw In** state. A **Throw In** is also issued in case the ball crosses the team's goal line but the last player to touch it belongs to the defending team.

In case the ball crosses the goal line, after being touched by the defending goalkeeper or an attacking player then a goalkeeper throw is awarded (**Goalkeeper Throw** state).

- If the game stops due to a time out request (*dTeamTOut*), then both teams take a pause of 1 minute (**Team Time Out**). During the team time out, players and team officials remain at the level of their substitution areas. The game is resumed via a **Free Throw** from the place where the ball was before the interruption.

- The game can also stop if the break time is reached ($dBreakTime=30$). Teams will stay on the **Break Time** state for 10 minutes and resume game via a **Throw-Off**.

4.6 Summary and Conclusions

This chapter described the proposed methodologies used to tackle the problem highlighted on the Introduction (Chapter 1).

Player detection is based on an initial manual colour calibration that, during the processing stage, is able to dynamically and automatically adapt to the light conditions that influence the colour (different field zones, shadows, influence from outside conditions due to the presence of windows, among others). The methodology adopted includes the identification of foreground pixels, using dynamic background subtraction, and the notion of team colour subspaces, using a Fuzzy inspired dynamic model to detect players based on the colour properties of their uniforms. Due to the Fuzzy colour classification, a given colour may be shared among teams. During this classification phase the colour models are constantly being updated in order to accommodate changes resulting from different light conditions and shadows.

Once the players are detected, they are then followed using a vector of Kalman Filters, one Kalman Filter per player. The player tracking is performed on real world coordinates which allows the usage of measures from any image source since they are referred to the same coordinates referential. Moreover, it allows each Kalman Filter to integrate more than one measure in case the player is being detected in more than one image source (on overlapped regions).

From the two previous phases, detection and tracking, it is possible to extract very useful information about the game, namely the players' individual and team's performance in terms of travelled distance, average speed, preferred field areas, among others.

Moreover, the visualisation tool allows for a better understanding of the teams' behaviour by providing a global, undistorted view of the field, as well as schematic views concerning the players' movements and teams' interactions.

These initial metrics correspond to low level information, which are then fed into more complex data analysis methodologies to extract the game phase and the defensive tactics.

The game phase is obtained with the aid of a Fuzzy classifier that uses the teams' longitudinal velocity and position along with the previous phase as inputs. These inputs are fuzzified and passed through a series of "IF-THEN" rules, which are aggregated and defuzzified using the mean of maximum method.

Whenever one of the teams is defending, the tactics is automatically determined and classified into a one line, two lines or three lines defensive system by an error minimization based classifier.

Finally, and in order to formalize the handball game, a model based on a Hierarchical Coloured Petri Net was proposed. The usage of such a Hierarchical Petri Net allowed dividing the game into three abstraction layers which highly contributes to better understanding the model.

Additionally, the usage of the colour property of such a model is quite suitable to differentiate both teams as well as to differentiate players inside the same team.

Chapter 5

Experimental Validation

This chapter presents the experimental validation of the proposed architecture as well as of the developed and proposed methodologies.

In order to accomplish this validation, the system was placed at a public sports hall to capture the official games of the Handball Portuguese SuperCup, year 2011. Additionally, extra tests were performed in a local sport's club (Académico Futebol Clube) which allowed to validate the flexibility of it, by providing footages with different camera configurations (1, 2 and 3 cameras' systems), and at the same time guarantee that the entire field is covered with good resolution and large overlapped areas as shown in Figures 4.3 and 5.1.

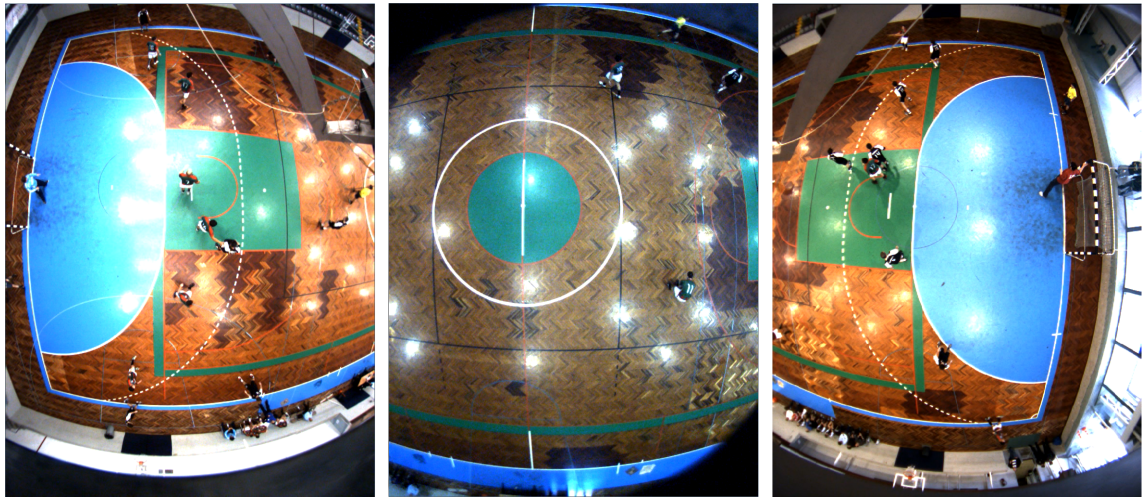


Figure 5.1: Images from a 3 cameras' system.

5.1 Player Detection and Tracking

The presented results relate to: system's accuracy; player detection and tracking rates. Also, an in-depth sensitivity analysis was performed in order to evaluate the robustness of the approach.

5.1.1 Processing Time

The system takes, on average, about 160ms to process each frame using an Intel Core i7-2630QM@ (2.00-2.90) GHz computer operating on Windows7 and can go up to 1s during an auto-expansion process. In order to obtain real time processing (a frame should be able to be processed in less than 33ms - cameras operate at 30fps) a speed up of around 5 times would be required. This result can be achieved using techniques such as Graphics Processing Unit (GPU), parallel processing or smart cameras which enable speeding-up the processing time.

5.1.2 Measurement Accuracy

Tests conducted on the measurement accuracy show that the error is less than 35cm for a 2 camera's system. This error is comparable with other works devoted to handball [Barros et al. \(2011\)](#) (7-28cm) and [Kristan et al. \(2009\)](#) (30-50cm). The following image (Figure 5.2) illustrates the errors at known points of the field. The green dots indicate the real positions, while the bars represent the distances to the measures obtained with the calculated homographies.

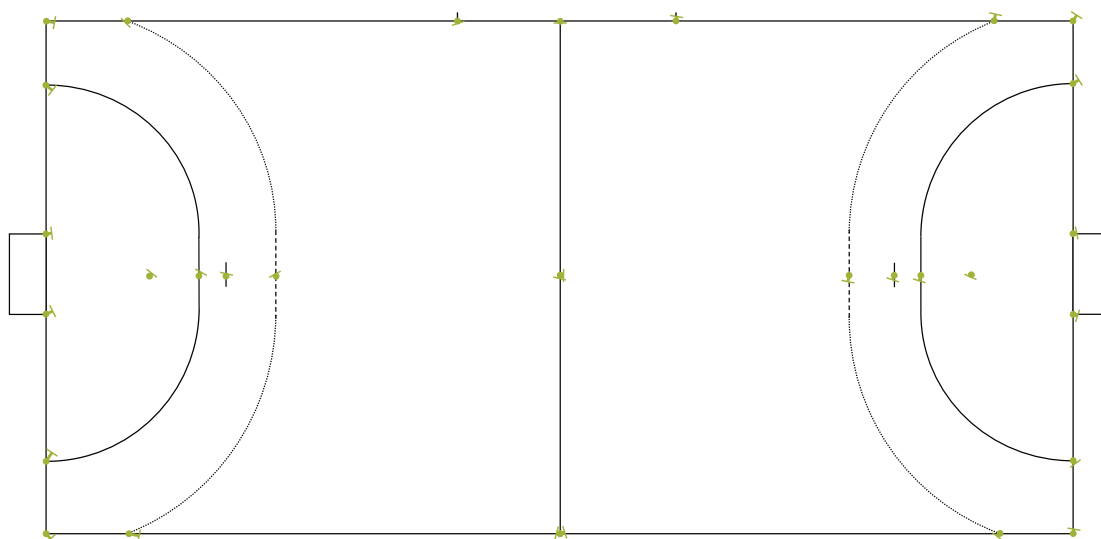


Figure 5.2: Measurement error. Dots represent real coordinates and bars represent the measure obtained using the estimated homography.

The error obtained is satisfactory given the characteristics of the implemented system: high area to be covered, the distortion induced by the wide angle lenses, and the system's inherent resolution (3.6cm, 3.7cm). Additionally, the maximum errors are achieved in the field periphery and not in critical areas for the game, such as the goal area.

5.1.3 Detection

In order to validate the Fuzzy methodology, two distinct teams that we named green and red teams (examples of both teams can be found in Figure 5.3(b)) were calibrated as explained on section

4.2.1. The original seeds were selected by clicking on the players of both teams, which resulted on the initial colour subspaces illustrated in Figure 5.3(a).

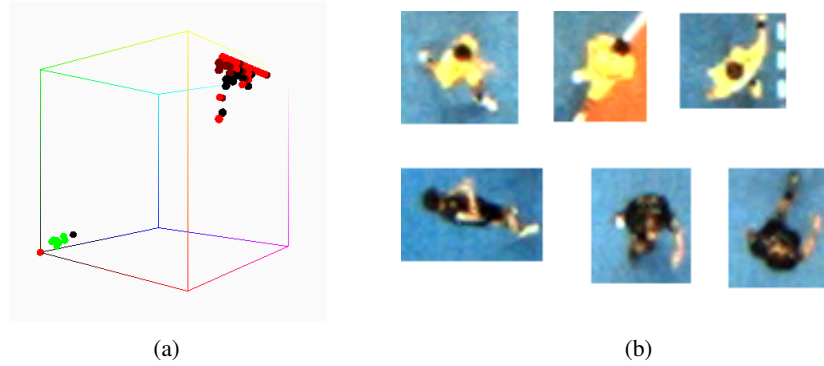


Figure 5.3: (a) Initial colour subspaces for green and red teams. Lighter dots are seed colours (C_S), intermediate are team colour (C_F) and the darkest resemble the team colour (C_L). (b) Examples of players from red (up) and green teams (down).

After 1020 frames and a time between expansions of 30 frames ($t_{exp} = 30$, which corresponds to 34 expansions), it is possible to confirm that the teams' colour subspaces have updated and dynamically changed from the original colour subspaces (Figure 5.3(a)) into the new colour subspaces of Figure 5.4. It is convenient to disable the auto-expansion process when the colour subspaces become stable and the detection rates acceptable because the overhead time induced by the auto-expansion process is high and can increase the processing time.

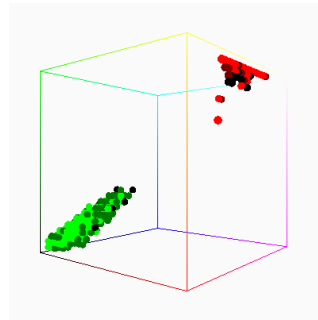


Figure 5.4: Final colour subspaces after 34 auto-expansions.

Figure 5.5 provides an overview of the colour subspaces evolution during the auto calibration process. It is possible to verify that, from the initial colour subspaces (Figure 5.3(a)) to the final ones (Figure 5.4), the colour triplets that belong to each class are adapting.

On the red team, the initial colour subspace defined by the user consisted in colour triplets that in fact do not belong to it. By making use of the persistence property of the colour triplets, the auto-expansion process adaptively "pruned" them which resulted on a slight more condensed and stable form.

On the other hand, the initial green team calibration did not reflect well the characteristics of the team's uniform. Therefore, it is possible to verify that the colour seed triplets (C_S) increase

greatly during the auto-expansion, the C_F colours triplets increase less due to their smaller persistence and colour triplets that resemble the colour, C_L , stay quite stable at low values.

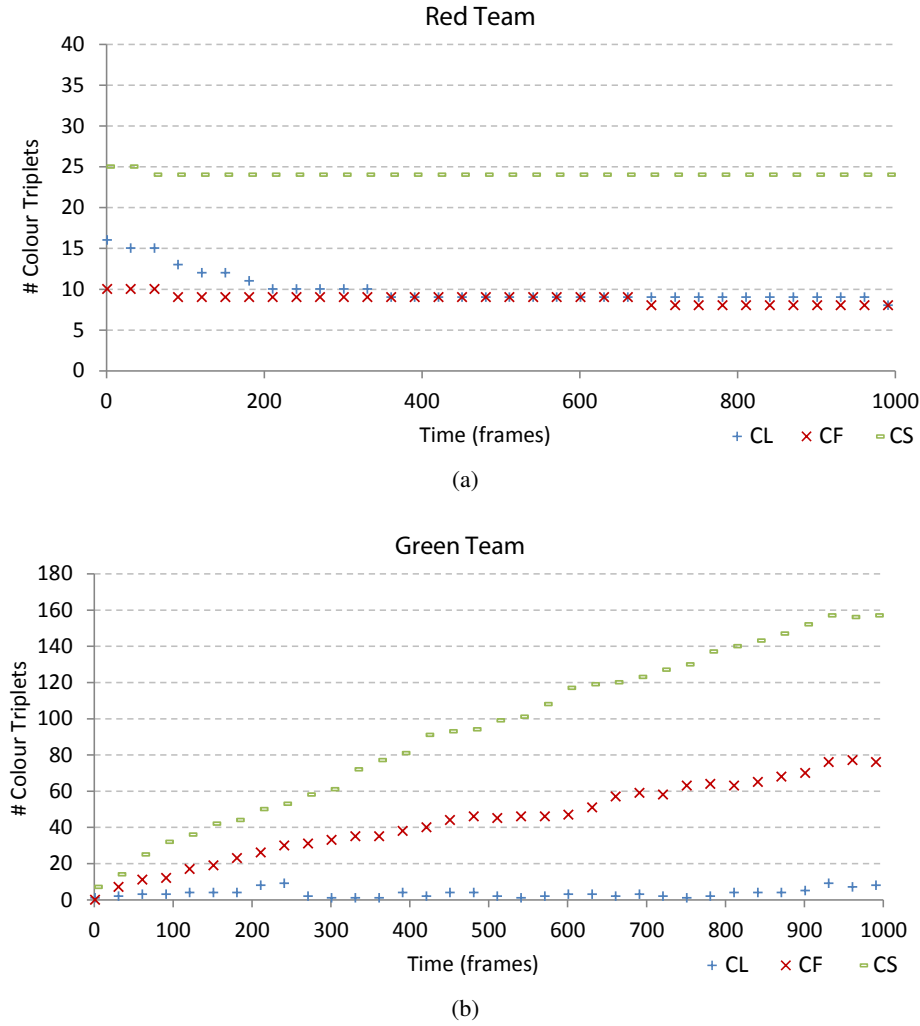


Figure 5.5: Evolution of the colour triplets that belong to each colour subspace and the respective belonging degree. The expansion is performed at every 30 frames ($t_{exp} = 30$).

From what has been said, it is important to highlight that the initial seed choice (a well-known problem of region growing methods) as well as the specific characteristics of the team's uniform will influence on how well the colour subspace adapts to the environment conditions. In fact, the initial seeds for the red team resulted in a faster adaptation: the colour subspace adaptation is very quick at the initial process on pruning colour triplets (C_F) that were miss categorized by the user. On the other hand, for the green team the stabilization seems to occur more at the end of the process.

Comparing the number of not detected players in each frame with and without the Fuzzy model of colour expansion, it is possible to verify that the overall player detection achieves better results with the mutable colour subspaces, as depicted in Figure 5.6 (each handball team is composed of 6 field players).

The usage of the auto-expansion methodology greatly improves the detection rate for the green team, in fact the number of miss detected players per frame, after frame 100, sporadically gets higher than 1 but never higher than 2, and most of the time is 0, while with the initial colour subspace continuously oscillates between 4 and 6 miss detections.

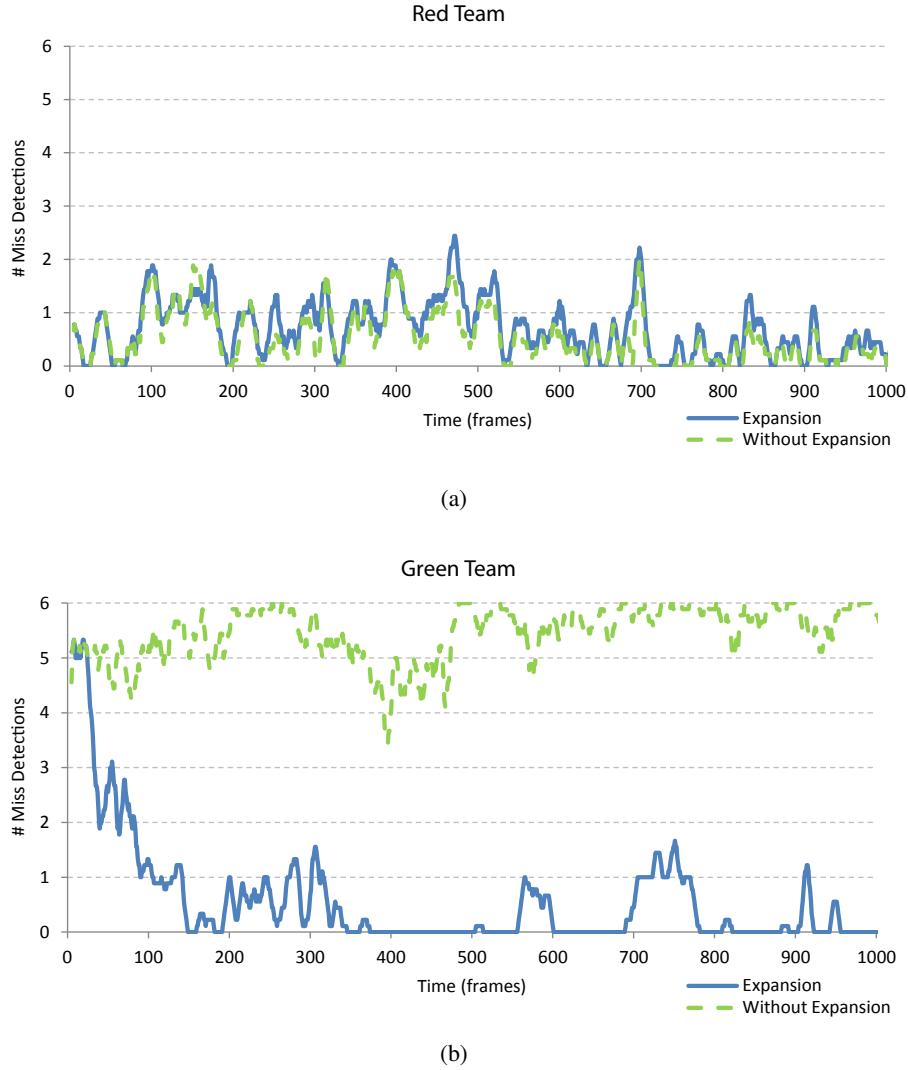


Figure 5.6: Number of miss detected field players in each frame with and without colour auto-expansion: (a) red and (b) green team. Each point was obtained by passing an averaging filter of size 9 to the original points.

For the red team the behaviour is somewhat different, and the number of miss detections is similar for both cases because the initial colour subspace already reflected well the team's characteristics. An important aspect to highlight is that, despite some pruning, the auto-expansion process was able to maintain the colour triplets that really belong to the team.

The results for the two teams indicate that the auto-expansion is extremely important and can greatly improve the results when the initial colour subspace does not reflect well the team

characteristics. Furthermore, during the game if the light conditions change, the colour subspace will also adapt in order to accommodate the new colour triplets and remove the ones that no longer belong to the team.

Figure 5.7 shows a zoom around the 6 meters' line of the players' detection at frame 671, where the green/red light pixels correspond to pixels that were labelled as belonging to green/red team, the green/red crosses correspond to players detected from green/red team, while the blue crosses indicate that the player's position was predicted by the Kalman filter and, therefore, it was not detected. Analysing the two images, it is possible to verify that, using the Fuzzy auto-expansion model, all players from both teams were detected (Figure 5.7(b)), while using the initial colour subspaces (Figure 5.3(a)) only one player from the green team was detected (Figure 5.7(a)). Additionally, the detected area of the players is higher with the Fuzzy model (visible on player 9) which allows having a better measure of the player's centre of mass.

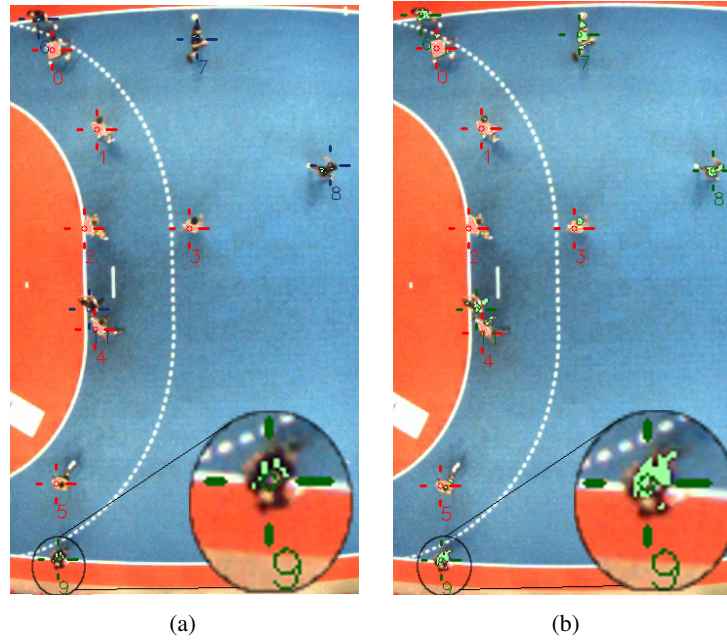


Figure 5.7: Results of the player detection at frame 671: (a) without auto-expansion and (b) with auto-expansion. Green and red crosses indicate correct detection while blue crosses indicate the position was predicted by the Kalman filter.

5.1.4 Sensitivity Analysis

In order to better assess the proposed global methodology, systematic tests were carried out by varying configuration parameters and operation conditions. A sensitivity analysis of the following parameters was performed: time between auto-expansions (t_{exp}), background learning constant (α), tracking prediction window (TPW) and initial seeds choice. Additionally, robustness tests to changes in brightness and image resolution were performed.

5.1.4.1 Expansion Time

As explained on Sections 4.2.1 and 4.2.3 the evolution of the colour subspaces is governed by the t_{exp} constant. This dependency is two-fold, first because the auto-expansion process is triggered at intervals of t_{exp} and secondly the triplets' persistence is also dependent on the t_{exp} according to Equation 4.2.

The choice of the t_{exp} plays an important role in the overall detection, because a too low value although providing a fast response, will also increase the processing time since auto-expansions occur more often. On the other hand, higher time between expansions will make the system adaptation slower and a good calibration cannot be achieved so fast. Figure 5.8 shows the miss detection rates using different t_{exp} values. It is possible to verify that for lower values (15 and 30) the system quickly adapts, while for $t_{exp} = 60$ it takes around 400 frames to achieve similar rates.

The overall values indicate that, for this specific case, the best performance is achieved with $t_{exp} = 30$ which has 6.95% global average of miss detections. Other similar value is 7.72% for $t_{exp} = 15$.

5.1.4.2 Illumination

The illumination's impact on the system's performance was tested by artificially changing the brightness of the video, similarly to what is expected to happen in a sudden illumination change. Figure 5.9 shows the miss detection rates for the original behaviour ("Original" series) compared with the ones obtained when the videos' brightness is changed by:

- applying a step of brightness of -20 at frame 600 and changing it to a step of +20 (above baseline) at frame 1200 ("Step" series)
- applying a ramp of brightness from frame 500 until frame 700 of -20 and a ramp from frame 1100 to 1400 of +40 ("Ramp" series)
- applying Gaussian noise at every video frame ("Gauss" series)

The visual difference between the brightness of the several test limits (-20, +20) and the original image can be seen in Figure 5.10.

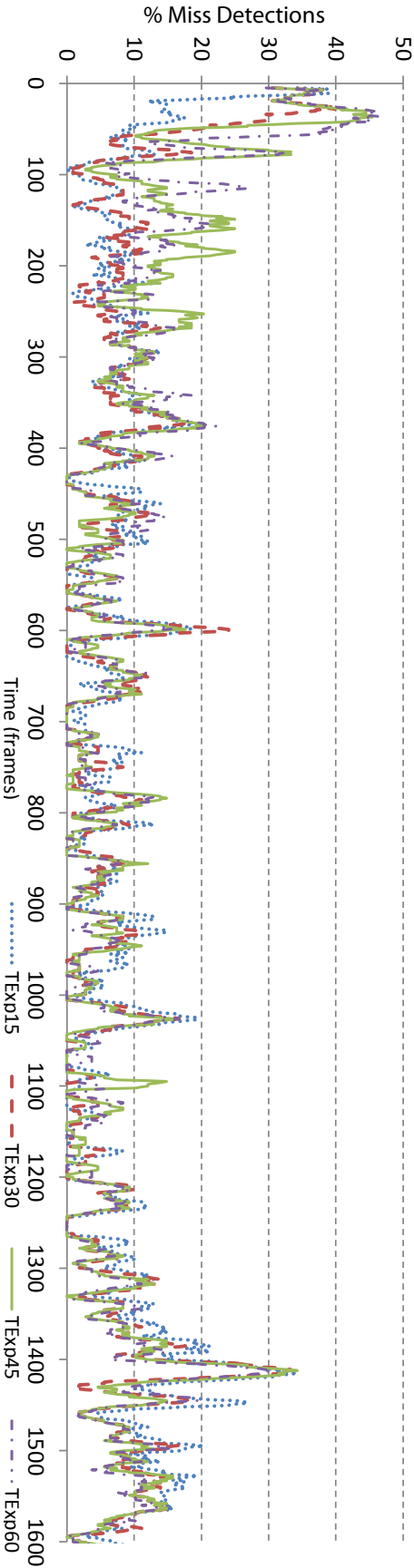


Figure 5.8: Comparison of miss detection rates using different t_{exp} : 15, 30, 45 and 60 frames (each point on the graphs was obtained by passing an averaging filter of size 9 to the original points).

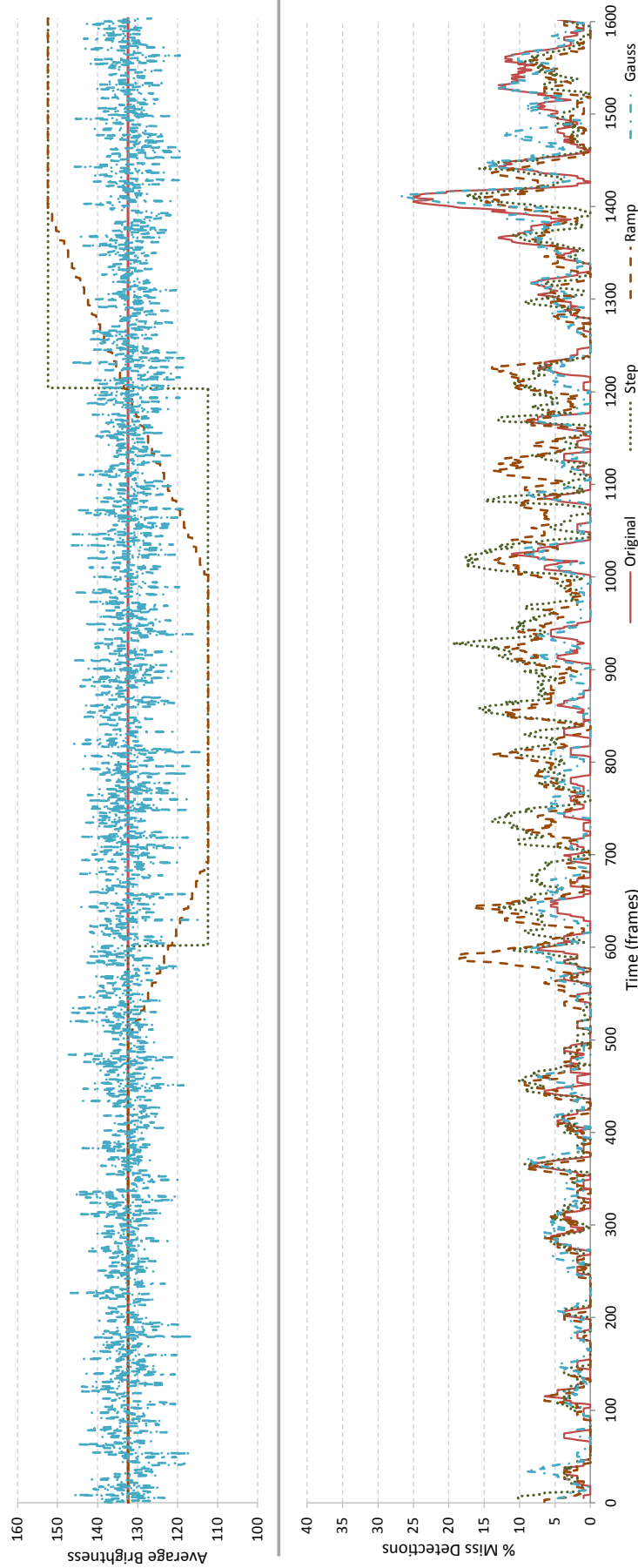


Figure 5.9: Comparison of miss detection rates (bottom) between the original video and video with artificially changed brightness. Original video and corresponding detection deficiencies are shown in the "Original" series. Video with long term added brightness step is shown on "Step" series. Video with added ramp brightness is shown on "Ramp" series. Video with added noise of Gaussian distribution is shown on "Gauss" series. Top graph indicates the average brightness of the video (results were collected with $t_{exp} = 30$ and each point on the graphs was obtained by passing an averaging filter of size 9 to the original points).

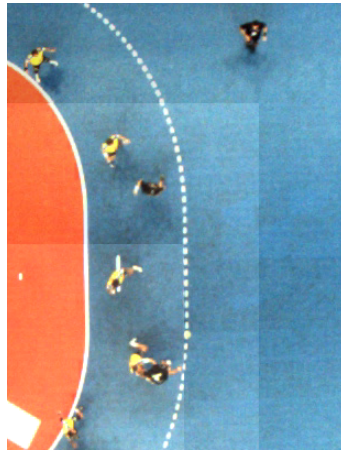


Figure 5.10: Example of images obtained by applying a brightness step of -20 (top), 0 (middle) and +20 (bottom) to the original videos.

Analysing Figure 5.9 it is possible to verify that the system behaves well when Gaussian noise is applied. Although the miss detection rate is slightly higher than in the original case, the overall performance is very similar. For the Step and Ramp tests, the miss detection rate increases specially when applying the -20 brightness delta at frame 600.

5.1.4.3 Background Learning Constant

Concerning the background learning constant, the choice of its value must take into account that high values tend to induce fast absorption of foreground features into background that cause miss detections, while low values make the background detection slower and therefore more pixels are considered foreground.

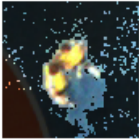





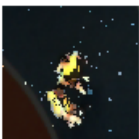


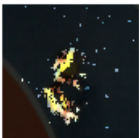


Table 5.1 illustrates how the background detection behaves using four different learning constant values (0.02, 0.08, 0.16 and 0.20), when two players stay still for a long period of time.

High learning constants (0.16 and 0.20) rapidly absorb foreground features and the two static players are miss detected from frame 82 onwards. A smaller constant (0.08) has a better behaviour, however on frame 82 only one of the players is detected, the other was included on the background. The smallest tested constant (0.02) proved to have better results, since it is able to completely absorb the players shadows in a relatively short period of time (51 frames) while keeping the detection rates high.

5.1.4.4 Seed Pixels Choice

Another important aspect to take into consideration is the initial colour seeds choice that, as previously mentioned, impacts heavily on detection rates. Figure 5.11 illustrates how the detection rate may be influenced depending on the initial colour calibration. Therefore experiment B, due to a poorer choice of the initial seeds presents higher miss detection rates particularly at the initial phase, only getting similar values after frame 600.

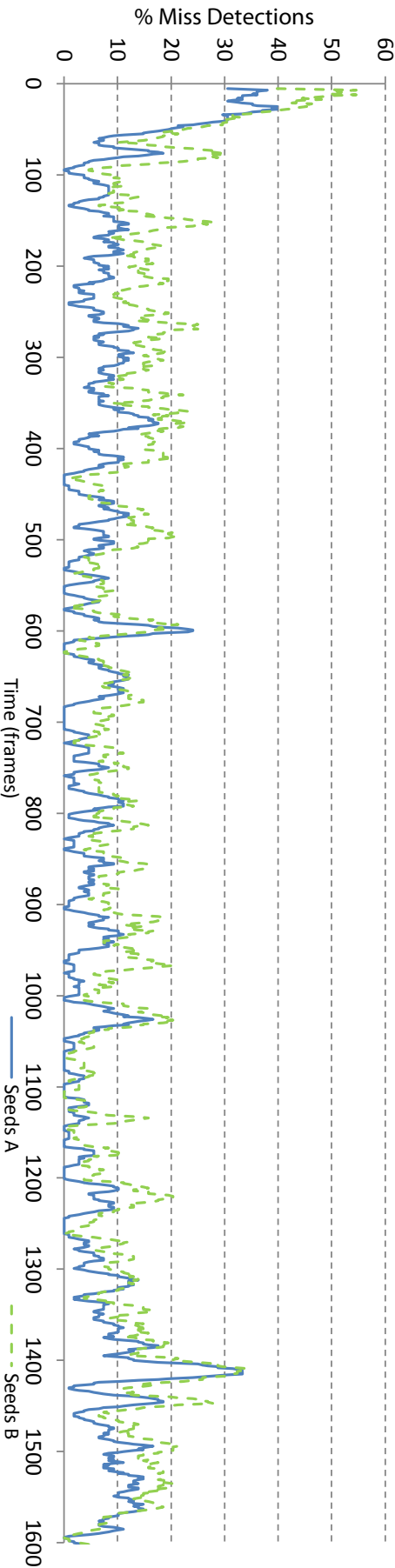
Table 5.1: Foreground detection using different learning constants ($\alpha = 0.02$, $\alpha = 0.08$, $\alpha = 0.16$ and $\alpha = 0.20$) at frames 31, 82 and 120. The pixels classified as background were darkened.

		Frame Number		
		31	82	120
Background Learning Constant	0.02			
	0.08			
	0.16			
	0.20			

It is also possible to verify that the dynamics of the colour subspaces behaves differently as illustrated on Table 5.2. This difference is more noticed for the red team, which for Experiment B results in a more condensed colour subspace compared with the one from Experiment A on frame 1600. For the green team, despite the differences on the initial subspaces the final ones are very similar. Nevertheless it is possible to verify that both colour subspaces show a tendency towards a similar shape.

5.1.4.5 Video Resolution

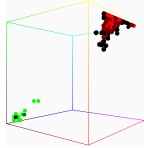
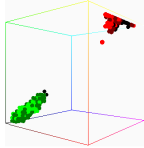
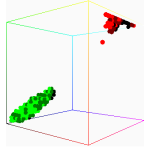
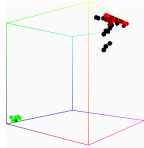
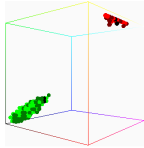
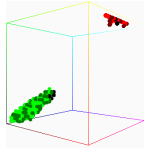
A final sensitivity test on the detection methodology consisted in evaluating how the camera resolution would influence the miss detection rate. The videos were down sampled by a factor of two and the resulting miss detection rates can be seen in Figure 5.12. For the down sampled video, the parameters concerning the minimum area allowed for a blob to be considered a player, were adjusted accordingly. Down sampling the video proved to induce a higher miss detection rate,



(a)

Figure 5.11: Comparison of miss detection rates between two different initial sets (Experiment A and Experiment B). All results were collected with $t_{exp} = 30$ and each point on the graphs was obtained by passing an averaging filter of size 9 to the original points.

Table 5.2: Colour subspaces evolution depending on the initial colour seeds. Lighter dots are seed colours (C_S), intermediate are team colour (C_F) and the darkest resemble the team colour (C_L).

		Frame Number		
		0	800	1600
Experiment A				
Experiment B				

which is justified by the smaller number of pixels that compose each player as can be seen in Figure 5.12(a).

5.1.4.6 Tracking Prediction Window (TPW)

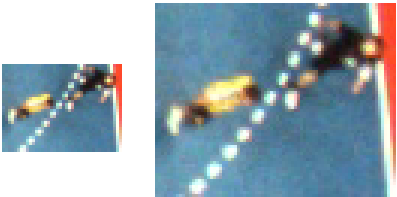
Normal Kalman filtering transforms the sequence of detections into tracking. In case of miss detections, the system is able to make a limited prediction in time in order to avoid user intervention. However, if a given player is not detected beyond a given configurable threshold called TPW, the user is prompted to locate the player or indicate a special game circumstance.

The TPW plays an important role in the tracking rate, therefore this experiment evaluated how the tracking rate is affected using TPWs of 1, 5, 10 and 20 frames. The results are shown in Figure 5.13.

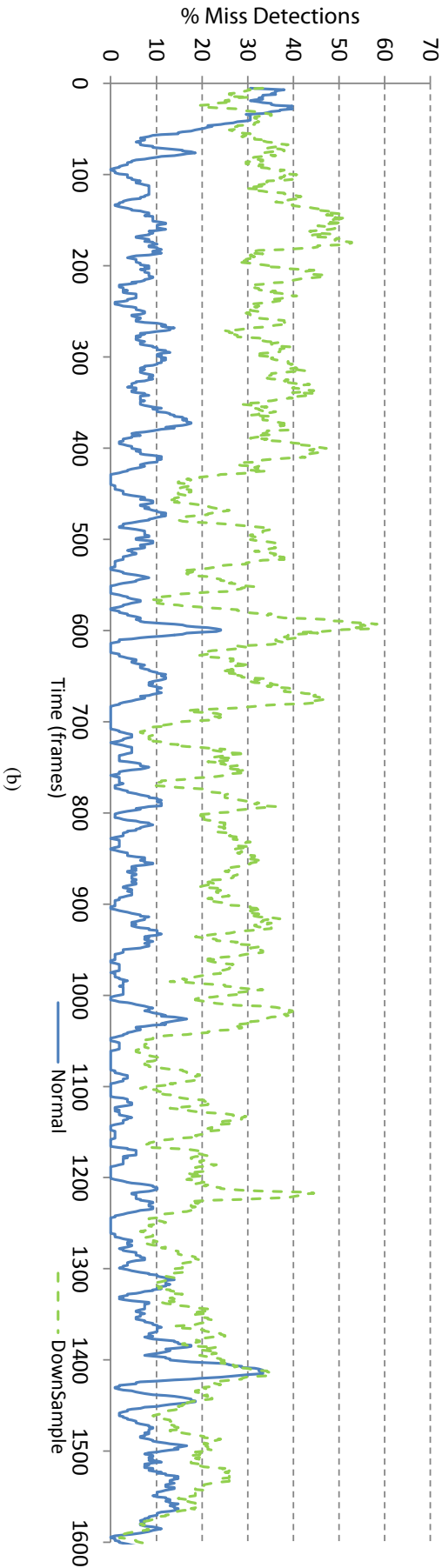
As can be seen, with the smallest TPW, which corresponds to 1, the tracker behaves like a pure filtered tracker where the results are based solely on the detection and therefore the tracking rate is lower (around 99.44%). Increasing the TPW allows for higher tracking rates (around 99.70% for TPW=20 frames), because the user is prompted to correct the tracker less often. However, the operator must pay more attention to the entire process, otherwise the tracker may be lost and continue to rely on a miss leading prediction.

5.1.5 Detection with Shared Colours

The proposed Fuzzy methodology enables having the aforementioned auto-expansion process, but also the possibility of having shared colours among both teams. To test it, a game where two teams (team A and team B) have the colour white on their uniforms was recorded and processed. The colour subspaces are shown in Figure 5.14(a) and the resulting process in Figure 5.14(b).



(a)



(b)

Figure 5.12: Comparison of miss detection rates using original image and down sampled image with a scale factor of 1/2: (a) examples of the two images (b) and plot showing the detection rates along the time. All results were collected with $t_{exp} = 30$ and each point on the graphs was obtained by passing an averaging filter of size 9 to the original points.

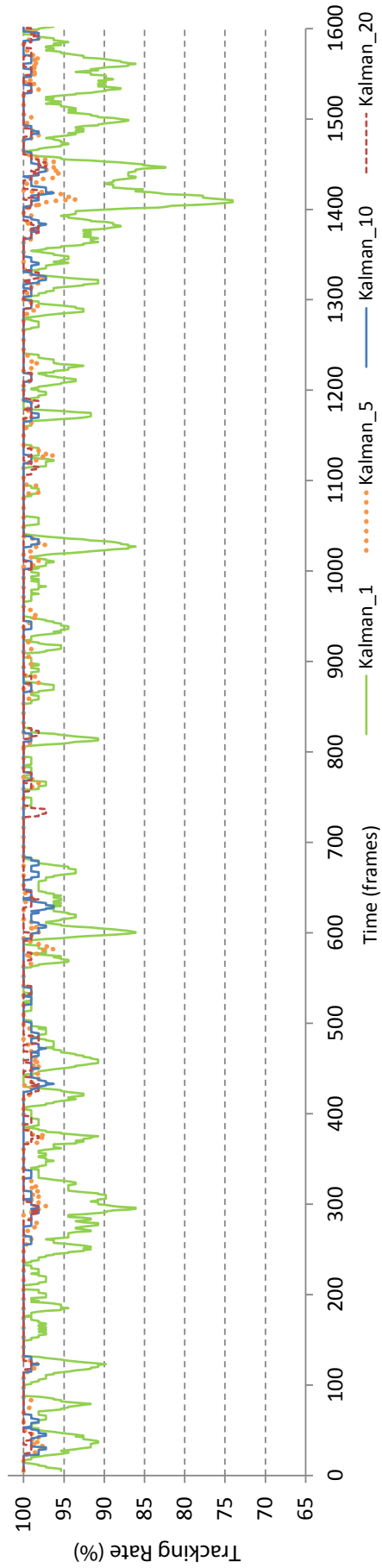


Figure 5.13: Comparison of tracking rates using different temporal windows (TPW = 1, 5, 10 and 20 frames) for the predictive stage of the Kalman Filter (results were collected with $t_{exp} = 30$ and each point on the graphs was obtained by passing an averaging filter of size 9 to the original points).

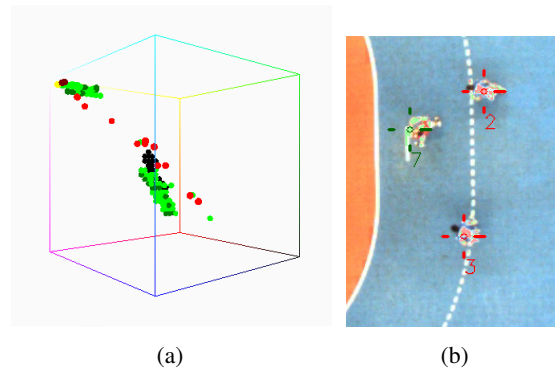


Figure 5.14: (a) Colour subspaces for two teams with a common colour (white) and (b) an example of a processed image.

Despite the fact that the colour white is common to both uniforms (shown by the yellow triplets on the colour subspaces of Figure 5.14(a)), the processing is able to identify the neighbouring pixels and correctly label the pixel under analysis. So for player 7 from team A, the white pixels are labelled as belonging to team A, because they are near red pixels that only belong to the team A, while for players 2 and 3 most of the white pixels are labelled as belonging to team B, due to their neighbourhood to blue pixels that only belong to team's B colour subspace.

5.1.6 Player Tracking

The miss detections achieved with the auto-expansion process are not consistent and persistent on time, so they are compensated by the predictive stage of the Kalman filter (Equation 2.7), and the benefits of the method are evident on the results obtained for the green team.

Taking into account the results of section 5.1.4.6, tests were performed with a TPW=5 frames, which allows having a good tracking rate as well as an accurate measure, because in case the tracker is lost this fact will be highlighted to the operator sooner.

Figure 5.15 compares the miss detection rates with and without the auto-expansion as well as with and without the Kalman filter. As expected, the worst case is when no Kalman filter or auto-expansion is performed. The usage of the auto-expansion method greatly improves the overall detection rates, which are further enhanced with the aid of the Kalman filter. Results for the non auto-expansion case are only plotted until frame 1000, because the human operator effort of correcting the tracking is very high, and more data does not provide extra information for what we are demonstrating.

Numerical results for each player on the tracking rate during the auto-expansion process can be seen on Table 5.3 (this data was obtained by manual validation performed by an expert during the tracking process, and combines miss detections resultant from the TPW parameter, as well as corrections made by the operator in case the prediction was wrong). As expected, despite some miss detections, the tracking achieved very good rates, having a success that ranges from 95.4% to 99.9% (and a corresponding average of 98.8%).

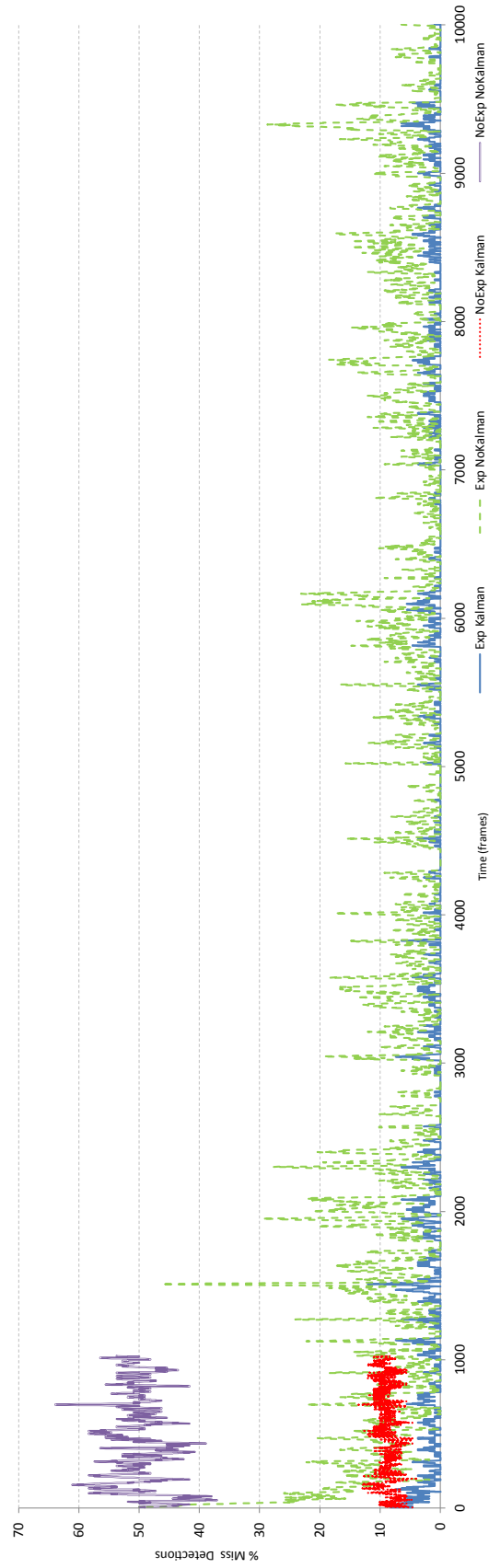


Figure 5.15: Comparison of miss detection rates with different setups: without auto-expansion and without Kalman filter (noExp noKalman); with auto-expansion and without Kalman filter (Exp noKalman); without auto-expansion and with Kalman filter (noExp Kalman); with auto-expansion and with Kalman filter (Exp Kalman). The Kalman Filters used a $TPW = 5$ frames and each point on the graphs was obtained by passing an averaging filter of size 9 to the original points.

Table 5.3: Tracking rates of all the players (6 field players per team) during 11000 frames analysed (≈ 370 s).

Player		Player	
ID	Rate(%)	ID	Rate(%)
P_0	99.44	P_6	99.78
P_1	99.61	P_7	99.62
P_2	97.87	P_8	98.83
P_3	96.95	P_9	99.90
P_4	99.32	P_{10}	99.67
P_5	99.04	P_{11}	95.44

Compared with other approaches these results prove to be similar to the ones obtained with the fast Rao-Blackwellized Resampling particle filter proposed by [Beetz et al. \(2007\)](#) ($> 90\%$), the Condensation particle filter proposed by [Kristan et al. \(2009\)](#) (ranging from 99.12% to 99.57%) or the directed weighted graphs of [Pallavi et al. \(2008a\)](#) (around 93.26%).

5.2 Ball Detection and Tracking

In order to evaluate the performance of the proposed methodology applied to the ball tracking case, a short sequence was processed using the same Fuzzy based methodology and a single Kalman Filter. Figure 5.16 shows the final video processed when tracking only the ball.



Figure 5.16: Processed video showing the tracking of the ball.

To have a better overview of how the ball behaves while it is being passed, hold by players or dribbled, Figures 5.17 and 5.18 show sub images of the entire field for each of these cases.

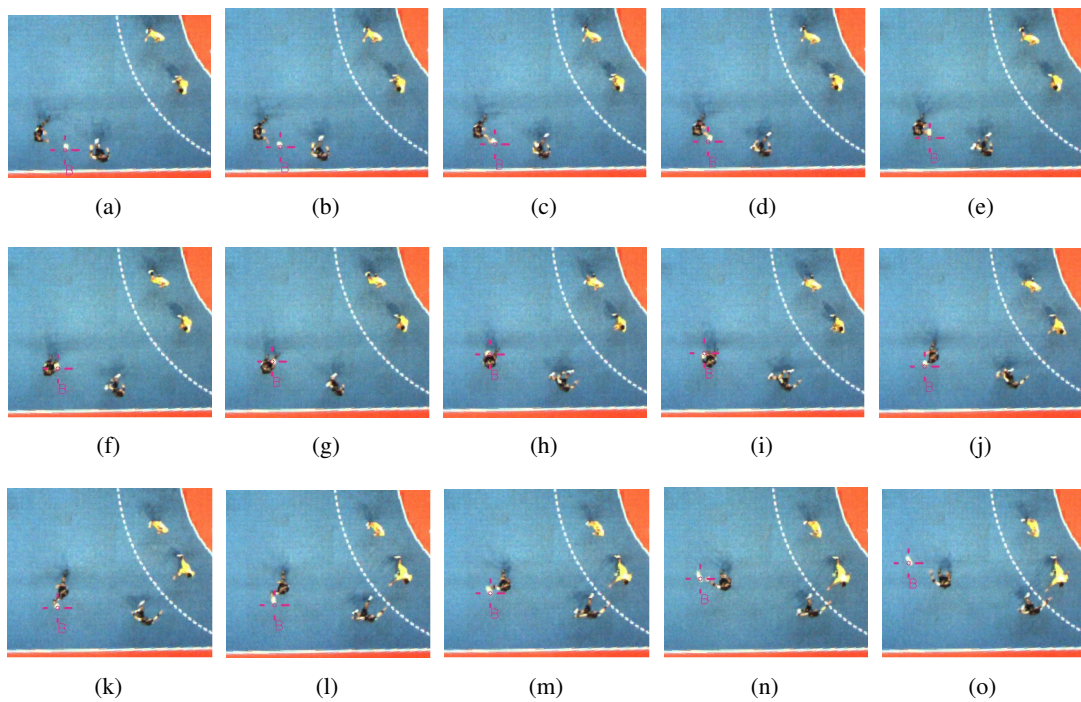


Figure 5.17: Images provided by the visualizing application when tracking the ball during a pass between two players.

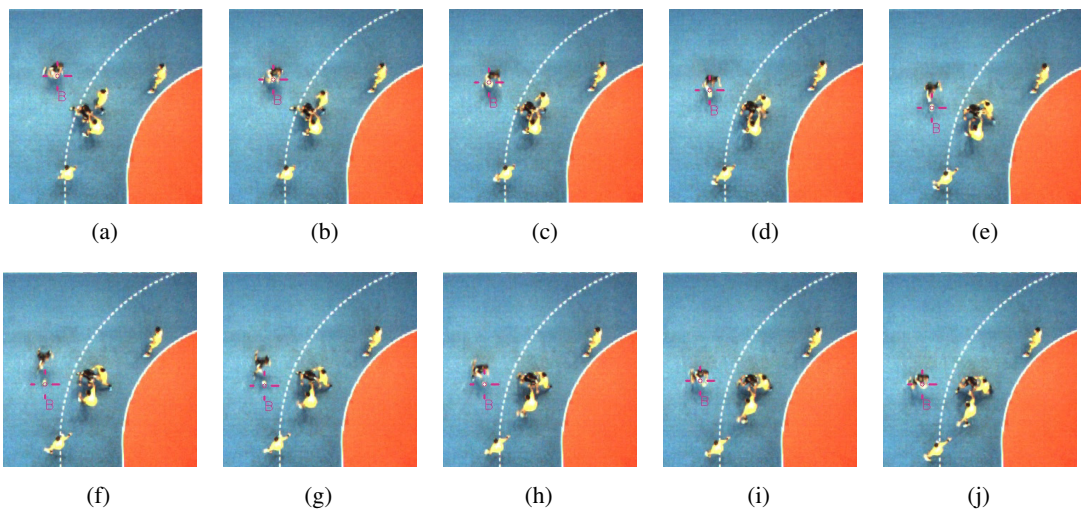


Figure 5.18: Images provided by the visualizing application when tracking the ball during a dribble.

From the tests, it is possible to verify that the methodology is not robust enough and the ball is lost quite often. The main reasons for this are:

- the high velocity at which the ball can travel is too high for the frame rate used on the cameras, in order to obtain better results it would be important to increase the cameras frame rate;

- on handball, players can grab the ball and carry it with them, which can occlude the ball under the players' torso, additionally the ball trajectory can be changed either by the player or when it is being dribbled. Long occlusions and unpredictable trajectory changes make the used Kalman Filter unsuitable.

In order to overcome these two issues the algorithm should be improved to take into account that a ball can be *part of the player*, and after being caught by the player can be thrown in any direction.

5.3 Game Analysis

In order to analyse the game itself, higher level analysis must be performed as already explained on section 4.4.

The two following subsection validate the methodology employed to detect the phase of the game as well as to determine the type of defence being used by the defending team.

5.3.1 Game Phase Detection

The game phase detection validation was performed by cross checking the results obtained from the proposed methodology with the labelling performed by a sport's expert.

As already referred, the proposed automatic methodology is based on a Fuzzy classifier that has as inputs the team x position, p_x , velocity in the x direction, v_x , and the information from the previous phase, $Phase[t - 1]$.

Figure 5.19 illustrates the evolution of the belonging degrees for each input parameter to each membership function as well as the evolution of the Fuzzy classifier output during the following sequence of phases of Game 1: *Attack* (from frame 1435 to frame 14956)), *Offensive Defensive* (from 1495 to frame 1750) and *Defence* (from frame 1750 to frame 1785).

The classification performed by the sport's expert was done with the aid of the Visualizer tool, therefore an expert was asked to visualize 10 minutes of one game and annotate the current game phase for each team.

Table 5.4 compares the results obtained by the automatic labelling and the manual labelling performed by the sport's expert.

Table 5.4: Comparison of phase detection between a sport's expert categorization and the developed automatic methodology for a 10 minute's period of Game 1.

Team A				Team B			
Game Phase	Auto	Expert	$\nabla frame$	Game Phase	Auto	Expert	$\nabla frame$
Def	575	575	0	Def Off	637	×	×
Def Off	1566	1550	16	Attack	752	585	167
Attack	1759	1850	-91	Off Def	1496	1550	-54

Continued on Next Page...

Table 5.4: Comparison of phase detection between a sport's expert categorization and the developed automatic methodology for a 10 minute's period of Game 1.

Team A				Team B			
Game Phase	Auto	Expert	$\nabla frame$	Game Phase	Auto	Expert	$\nabla frame$
Off Def	3643	3643	0	Def	1722	1850	-128
Def Off	3923	3880	43	Def Off	3656	3643	13
Attack	4084	4140	-56	Off Def	3907	3880	27
Off Def	4938	4890	48	Def	4083	4140	-57
Def	5123	5300	-177	Def Off	4973	4890	83
Def Off	5845	×	×	Attack	5193	5300	-107
Attack	6077	5775	302	Off Def	5800	×	×
Off Def	6863	6850	13	Def	5989	5775	214
Def Off	7012	6949	63	Def Off	6873	6850	23
Off Def	7213	7227	-14	Off Def	×	6949	×
Def Off	7523	7500	23	Def	7078	×	×
Attack	7719	7737	-18	Def Off	7251	7227	24
Off Def	8615	8613	2	Off Def	7506	7500	6
Attack	9100	8760	340	Def	7715	7737	-22
Off Def	9595	9570	25	Def Off	8619	8613	6
Def	9798	9870	-72	Attack	×	8760	×
Def Off	11794	11770	24	Def	8892	×	×
Attack	11999	×	×	Def Off	9592	9570	22
Off Def	12135	12117	18	Attack	9820	9870	-50
Attack	12570	12260	310	Off Def	11783	11770	13
Off Def	14147	14130	17	Def	11974	×	×
Def Off	14365	14350	15	Def Off	12158	12117	41
Off Def	14549	14490	59	Def	12387	12260	127
Def Off	14696	14679	17	Def Off	14147	14130	17
Off Def	14878	14840	38	Off Def	14379	14350	29
Def	15034	15070	-36	Def Off	14544	14490	54
Def Off	16373	16359	14	Off Def	14692	14679	13
Off Def	16561	16520	41	Def Off	14904	14840	64
Def	16729	16743	-14	Attack	15042	15070	-28
				Off Def	16369	16359	10
				Attack	16786	16743	43

It can be seen that most of the phases are correctly detected by the proposed methodology, however and in order to evaluate the proposed methodology, three different metrics were chosen.

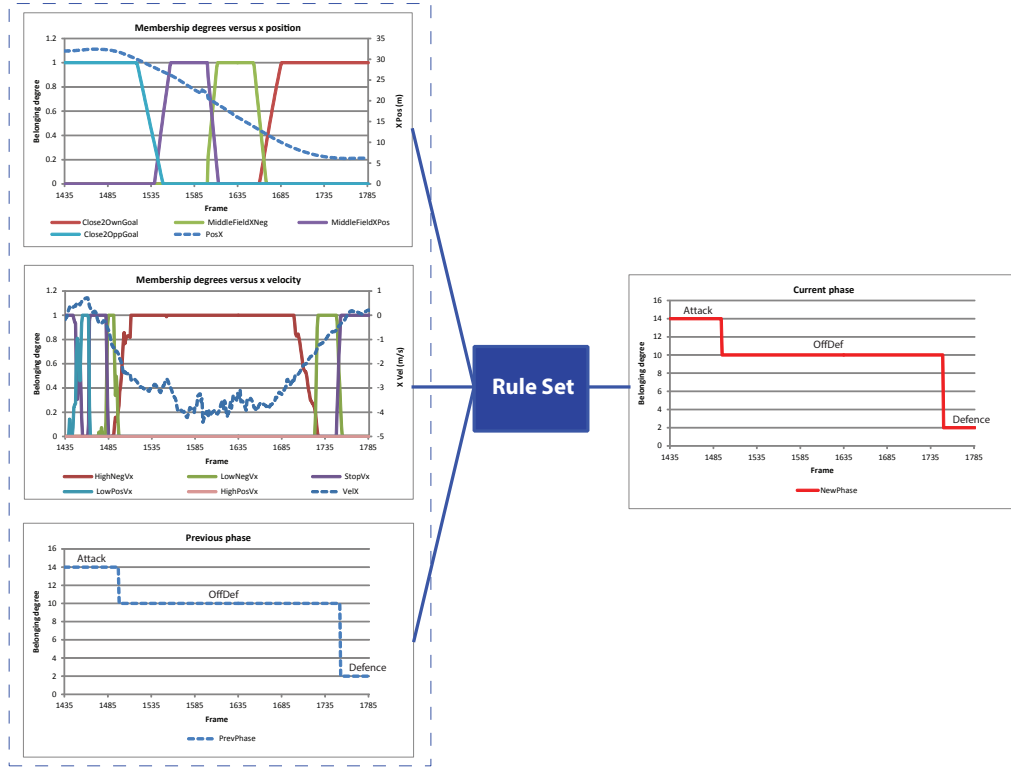


Figure 5.19: Evolution of the belonging degrees and output of the Fuzzy classifier from frame 1435 to frame 1786.

Each of the proposed metrics evaluates different parameters such as the phase detection, the phase consistency with manual labelling and the phase start alignment.

The first metric, F-measure (Santiago et al. (2012b)) is determined using Equation 5.1, and allows evaluating how well the game phases are detected. This measure compares the number of correct events c (*i.e.*, phases that were also categorized by the sport's expert) with the number of false positives, f^+ (*i.e.*, phases that were categorized by the application but were not categorized by the sport's expert) and false negatives, f^- (*i.e.*, phases that were categorized by the sports expert but were not categorized by the application):

$$F = \frac{c}{c + f^+ + f^-} \quad (5.1)$$

Applying this measure to the results of Table 5.4 we obtained 86.76% of game phases correctly detected for approximately 17000 frames analysed.

The second metric, P_{time} , measures the percentage of time during which both the automatic as well as the expert categorization match. Equation 5.2 demonstrates how this measure is determined.

$$P_{time} = \frac{\sum_{t=0}^{t_{total}} \mathbb{1}_{ph_{auto}=ph_{exp}}}{t_{total}} \quad (5.2)$$

This measure was applied independently to each team and the results ranged from 84.8% for team A to 82.6% for team B (during the 17000 frame period).

Finally, the third metric, T_{align} , allows evaluating how miss aligned are the different phases determined by the application when compared with the phase start determined by the sport's expert. Equation 5.3 illustrates how this measure is obtained.

$$T_{align} = \frac{\sum_{ph=0}^{ph=ph_{total}} |t_{ph_{auto}} - t_{ph_{exp}}|}{ph_{total}} \quad (5.3)$$

The analysed time period has a phase time misalignment of 1.89s, when compared with the expert's opinion.

5.3.2 Team Formation Detection

The validation of the defensive team formation labelling was performed through comparing the results obtained automatically with the ones labelled by a sports' expert.

While the expert was categorizing the game phase he was also requested to indicate which defensive system the teams were using and label it into one defensive line, two defensive lines or three defensive lines. The same game period was analysed with the developed methodology and the results comparison is shown in Figures 5.20 and 5.21.

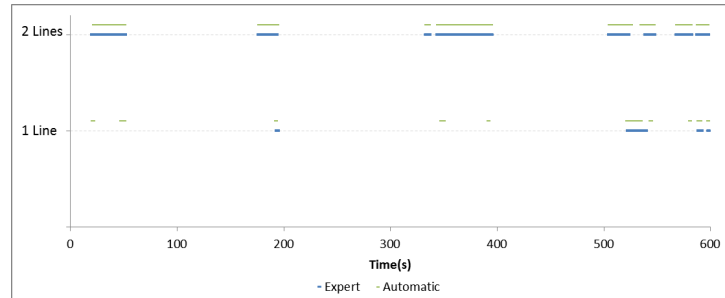


Figure 5.20: Results comparison between expert and automatic defensive team formation labelling for team A.

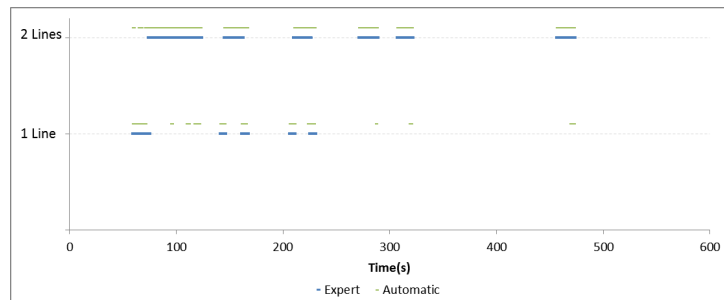


Figure 5.21: Results comparison between expert and automatic defensive team formation labelling for team B.

The automatic method achieved 89.92% for team A and 85.97% for team B of accuracy, when compared with the expert's data.

5.4 Results Visualization and Annotation

The log file generated during the game processing contains the players' positions and velocities and, as explained in chapter 4.1.3, can be used by the Visualizer not only to see the two images fused into a single image with the tracked players highlighted, but also to extract statistics of the players' behaviour.

The results analysis was performed for two games, Game 1 and Game 2. Game 1 was played between Team A and Team B, while Game 2 was played between Team C and Team A. For both games, a period of approximately 10 minutes was taken into consideration.

Figure 5.22 illustrates how the Visualizer shows the field and the players highlighted to the end user.

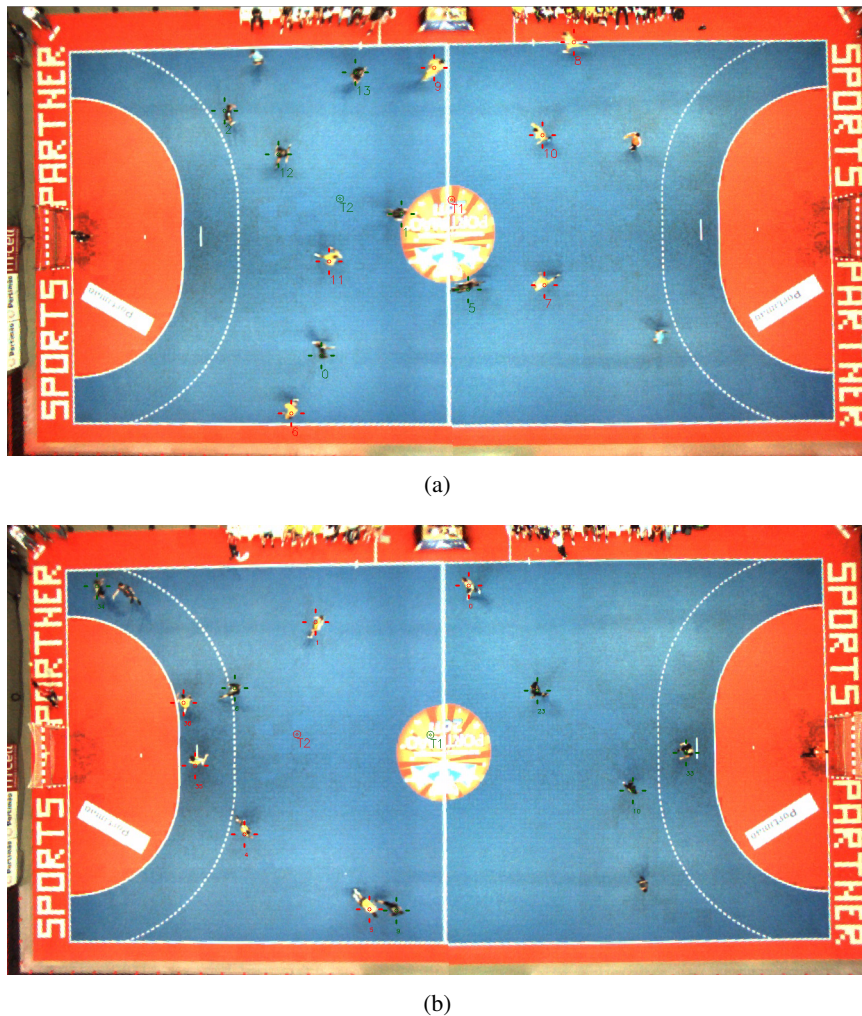


Figure 5.22: Images provided by the visualizing application: (a) image of Game 1, (b) image of Game 2.

This global image is obtained by first converting each pixel coordinate into real world coordinates and then converting them back to a common coordinate system (in this case the right image coordinate system).

The global view also shows the players' centre of mass (crosses above the players), their IDs and the teams' centre of mass (the two concentric circles with T_x).

Additionally, the application provides statistics for each player, positional and speed maps per player and the team's tactical map. These two topics are deeper explored on the two following sections (Sections 5.4.1 and 5.4.2).

5.4.1 Metrics

The defined metrics are more player centred, and therefore include players' maximum and average speeds, as well as the distance covered during the analysed period.

Table 5.5 shows the statistics for the players on Game1, while Table 5.6 shows the statistics for Game 2, during the two periods under analysis. It is important to mention that handball is a game that allows unlimited substitutions and, therefore, players are constantly being replaced, and so the table also includes the time period at which each player started ($T_i(s)$) and stopped playing ($T_e(s)$).

Table 5.5: Player statistics of Game 1 during approximately 10 minutes of game.

T	P	D(m)	$T_i(s)$	$T_e(s)$	Avg speed(m/s)	Max speed(m/s)
A	P_0	1001.57	0	603	1.65	9.22
	P_1	862.28	0	603	1.43	8.93
	P_2	1001.85	0	603	1.64	10.32
	P_3	82.05	0	53	1.55	7.2
		35.89	125	134	3.95	8.98
		63.13	168	196	2.18	8.89
		61.59	244	262	3.62	6.99
		106.91	323	395	1.47	7.75
		256.38	477	703	2.03	7.71
	P_4	74.15	0	54	1.39	8.40
		36.51	182	196	2.49	6.30
		77.14	334	396	1.24	8.33
		153.69	509	603	1.63	9.35
	P_5	858.55	0	603	1.42	7.76
	P_{12}	82.29	53	125	1.15	6.07
		51.93	134	168	1.55	5.03
		64.53	196	245	1.38	5.46
		70.75	262	323	1.16	6.46
		97.02	395	477	1.16	5.86

Continued on Next Page...

T	P	D(m)	$T_i(s)$	$T_e(s)$	Avg vel(m/s)	Max vel(m/s)
B	P_{13}	202.76	54	181	1.59	7.03
		210.57	197	334	1.53	7.59
		184.48	396	510	1.62	6.90
	P_6	1017.63	0	603	1.68	9.50
	P_7	867.93	0	603	1.44	8.37
	P_8	961.01	0	603	1.58	9.40
	P_9	376.67	0	245	1.54	6.78
		99.54	263	324	1.61	5.12
		81.28	435	476	1.98	5.29
	P_{10}	108.40	0	66	1.65	5.10
		43.35	126	140	3.09	7.69
		39.22	169	198	1.37	4.06
		47.88	244	262	2.66	7.14
		168.97	324	435	1.50	7.11
		233.57	476	603	1.85	9.37
	P_{11}	930.06	0	603	1.54	8.38
	P_{14}	90.95	66	126	1.52	6.73
		58.73	140	169	2.03	6.72
		528.41	198	603	1.30	8.30

Analysing Table 5.5 it is possible to verify that:

- players P_0 , P_1 , P_2 and P_5 from team A, and players P_6 , P_7 , P_8 and P_{11} from team B played during the entire analysed period, while the other players were constantly replaced, depending if the team was attacking or defending,
- for the same amount of time, players P_0 , P_2 and P_6 had covered more distance (above 1000m) than players P_1 , P_5 , P_7 , P_8 or P_{11} (all these players had been on the field for the same amount of time (603s)),
- the average speeds of the players ranged from 1.15 m/s (player P_{12}) to 3.95 m/s (player P_3). For players that played during the entire analysed time period the values ranged from 1.42m/s (player P_1) to 1.68m/s (player P_6),
- the maximum instantaneous speed ranged from 4.06 m/s (player P_{10}) to 10.32 m/s (player P_2),
- players that were replaced, played on average 32% of the time, while the average maximum speed was 7.4m/s and the global average speed of the game was 1.77m/s.

Table 5.6: Player statistics of Game 2 during approximately 11 minutes of game.

T	P	D(m)	$T_i(s)$	$T_e(s)$	Avg speed(m/s)	Max speed(m/s)
C	P_0	545.93	0	389	1.39	7.59
	P_1	996.16	0	669	1.48	8.77
	P_2	54.85	0	42	1.27	5.35
		53.72	72	107	1.50	4.79
		75.50	119	182	1.20	4.20
		62.18	208	243	1.75	5.06
		98.00	315	412	1.03	7.17
		40.81	442	468	1.56	7.44
		78.89	499	571	1.11	5.32
		52.58	629	669	1.30	6.03
		80.68	0	57	1.41	10.25
		164.16	73	186	1.45	7.69
	P_3	70.36	210	250	1.73	7.08
		90.45	317	421	0.87	5.02
		170.82	444	590	1.34	6.36
		47.90	635	669	1.38	6.09
	P_4	977.40	0	669	1.45	9.52
	P_5	941.70	0	669	1.40	9.71
	P_{12}	61.49	42	72	2.07	5.81
		32.00	107	119	2.74	5.88
		54.29	182	207	2.12	5.38
		145.17	243	315	2.02	8.37
		37.41	571	635	2.01	4.68
		46.77	57	73	2.89	7.32
	P_{14}	54.72	186	210	2.28	7.15
		127.40	250	317	1.92	8.80
		19.44	421	426	2.92	4.92
		112.53	590	629	2.80	8.01
		64.98	412	444	2.01	7.26
	P_{16}	70.59	468	499	2.03	6.64
		214.94	426	442	2.52	9.87
	P_{17}	247.76	516	669	1.58	8.72
		740.22	0	467	1.58	8.31
	P_6	131.21	522	584	2.11	9.04
		122.50	609	669	1.99	6.49

Continued on Next Page...

T	P	D(m)	$T_i(s)$	$T_e(s)$	Avg vel(m/s)	Max vel(m/s)
		120.26	0	79	1.50	6.44
	P_7	34.24	108	122	2.55	5.88
		53.07	178	210	1.70	5.95
		116.37	241	289	2.37	8.92
		311.47	466	669	1.51	10.14
	P_8	99.94	0	51	1.94	7.10
		189.07	74	178	1.80	9.25
		59.27	210	240	1.97	5.46
		315.90	321	522	1.54	7.07
		59.29	584	609	2.45	8.15
	P_9	960.21	0	669	1.43	7.98
	P_{10}	921.77	0	669	1.38	12.14
		129.41	0	50	2.13	13.12
		61.09	79	108	2.04	7.41
	P_{11}	105.10	125	178	1.95	6.48
		66.67	212	241	2.27	7.77
		45.08	446	466	2.28	8.20
		176.70	522	669	1.97	8.02
	P_{13}	108.11	52	108	1.95	6.32
		787.57	122	669	1.42	7.42
		43.48	51	74	1.98	7.92
		35.32	108	125	2.18	5.07
	P_{15}	51.29	178	212	1.52	5.47
		89.13	240	320	1.10	10.62
		54.60	467	522	0.96	7.49
		136.44	582	642	2.33	8.80

For Game 2 the data in Table 5.6 indicates that:

- players P_1 , P_4 and P_5 from team C, and players P_9 and P_{10} from team A played during the entire analysed time period, while the other players were constantly being replaced, depending if the team was attacking or defending,
- the distance travelled by the players that stayed on the game during the analysed time period (669s) ranged from 921.77m for player P_{10} until 996.16m for player P_1 ,
- the average speeds of the players during this game ranged from 0.87 m/s (player P_3) to 2.92 m/s (player P_{14}). For the players that were playing during the analysed time period the values ranged from 1.38m/s (player P_{10}) to 1.48m/s (player P_1),

- the maximum instantaneous speed ranged from 4.20 m/s (player P_2) to 11.54 m/s (player P_{11}),
- players that have been replaced play on average 12% of the time, while the average maximum speed was 7.4m/s and the global average speed of the game was 1.81m/s.

Comparing the information from both games, it is possible to verify that the second game was more dynamic, not only by the amount of players' substitutions, but also by the slightly higher average speed (1.81m/s versus 1.74m/s)

This statistical information can be very useful for sports experts since it allows them to have perception of the preferred field areas of each player as well as the effort spent during the game (distance travelled, average speed and peaks of speed). Moreover, tactical maps allow inferring methodologies and strategies followed by the teams as demonstrated on the following subsection (5.4.2).

5.4.2 Maps

The positional and speed maps constitute a useful source of information, not only by the content itself, but because they provide the information on a visual manner which is easier to analyse.

Figure 5.23 shows the positional maps of some of the players of team A during Game 1, while Figure 5.24 shows the same maps for players of team B (only the players who played during the entire game were chosen, because they give a better overview of the areas where the players tend to be more often).

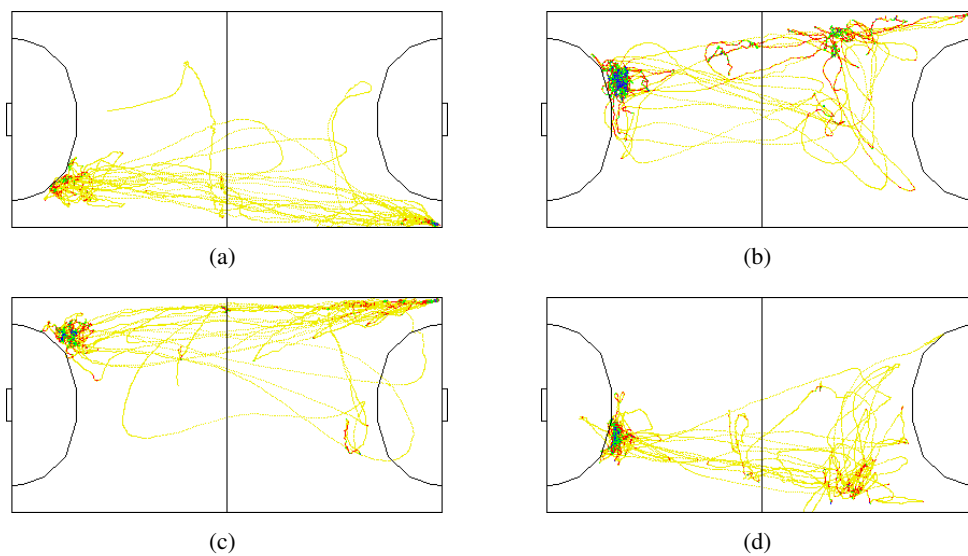


Figure 5.23: Positional maps of the 10 minutes for team A players during Game 1:(a) player P_0 , (b) player P_1 , (c) player P_2 , (d) player P_5 .

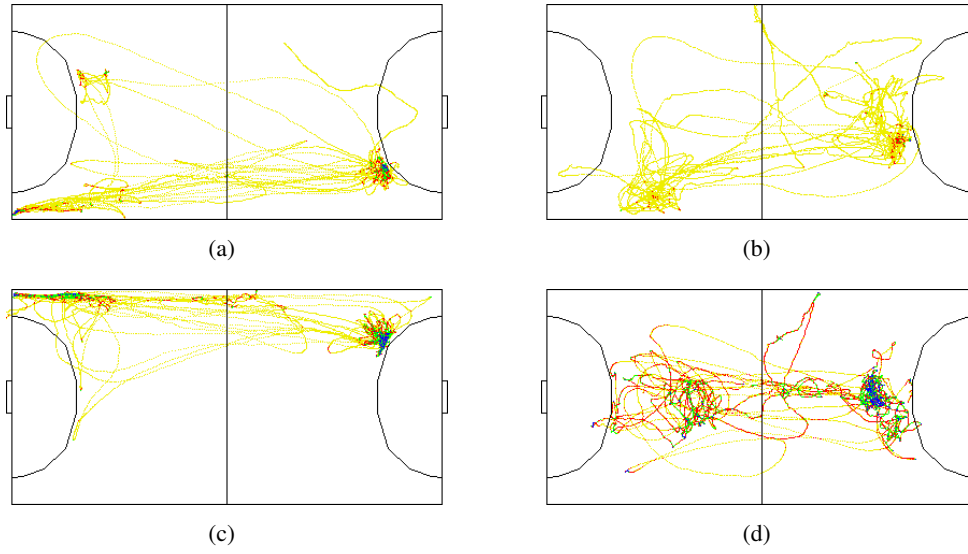


Figure 5.24: Positional maps of the 10 minutes for team B players during Game 1:(a) player P_6 , (b) player P_7 , (c) player P_8 , (d) player P_{11} .

Another information provided by these positional maps is how often a player is in each part of the field, which is indicated by the colour. The measure chosen to determine this frequency obeys to the Equation 5.4.

$$freq_{(x,y)} = \frac{time_{(x,y)}}{time_{max}} \times 100 \quad (5.4)$$

Where $time_{(x,y)}$ is the time the player spent on position (x,y) and $time_{max}$ is the time spent on the position where the player stood longer.

Using Equation 5.4, the following colours were defined to pass this information: yellow means $freq_{(x,y)} < 40\%$; red means $40\% \leq freq_{(x,y)} < 80\%$; green means $80\% \leq freq_{(x,y)} < 90\%$ and blue means $freq_{(x,y)} \geq 90\%$.

The positional maps show the field areas covered by each player and allow inferring which areas are preferred and even determine the position the player occupies within the team (lateral versus central player, attacker versus defender).

Players P_0 and P_2 of team A play more (either defending or attacking) on the lateral side of the field (Figures 5.23(a) and 5.23(c)), while players P_1 and P_5 (Figures 5.23(b) and 5.23(d)) defend more near the centre and use more space to attack.

Concerning team B, players P_6 and P_8 play more on the laterals, player P_7 is clearly a player that defends and attacks on the middle and player P_{11} has a behaviour that is characteristic of a pivot.

Also, and as part of the game itself, players spend less time on the middle of the field (yellow colour) and are most of the time either defending near the 6 meter line or attacking between the 6 meter and 9 meter lines (blue colour).

For Game 2 the information regarding the players' positional maps can be seen in Figures 5.25 and 5.26.

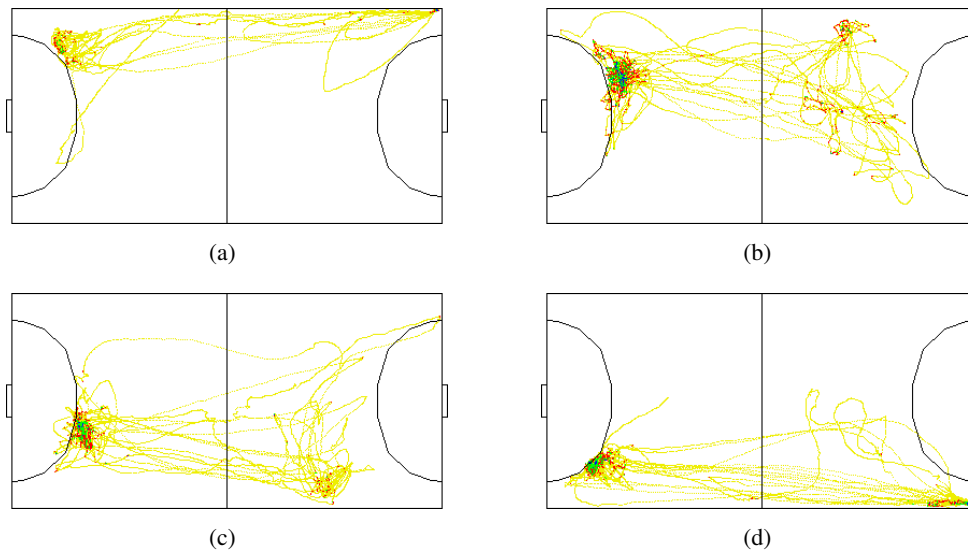


Figure 5.25: Positional maps for the 11 minutes of players from team A during Game 2:(a) player P_0 , (b) player P_1 , (c) player P_4 , (d) player P_5 .

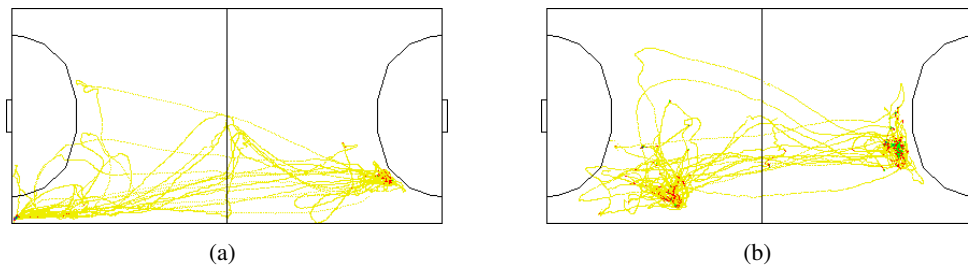


Figure 5.26: Positional maps for the 11 minutes of players from team B during Game 1:(a) player P_9 , (b) player P_{10} .

Like in the previous game, on this game there are players that play more on the lateral side of the field: players P_0 and P_5 of team C and player P_9 of team B, while the others occupy a slightly more central position.

Another type of map that can be useful is the speed map. This type of map allows visualizing the areas of the field where the players have a speed between two user defined speed intervals. The following figures, Figures 5.27 and 5.28, exemplify the speed of one player for each game for three different speed intervals: below $2m/s$, between $2m/s$ and $4m/s$ and above $4m/s$.

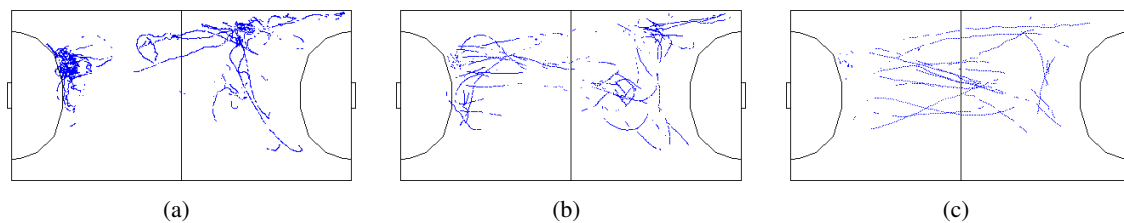


Figure 5.27: Velocity maps for P_1 (team A) during Game 1 (10 minutes of game):(a) speed below 2m/s, (b) speed between 2m/s and 4m/s and (c) speed above 4m/s.

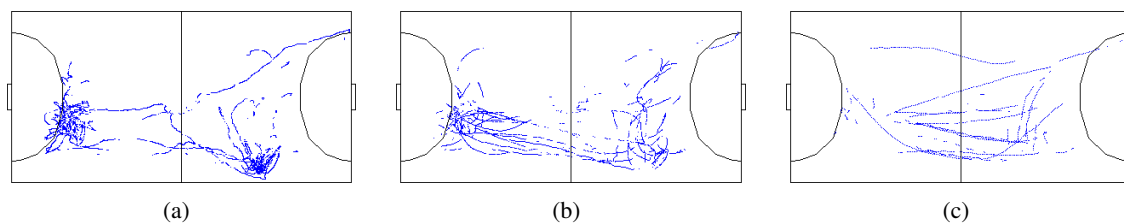


Figure 5.28: Speed maps for P_4 (team A) during Game 2 (11 minutes of game):(a) speed below 2m/s, (b) speed between 2m/s and 4m/s and (c) speed above 4m/s.

The speed maps show that players tend to have higher speeds during transitional phases (when players change from an attacking position into a defending one or vice versa), while the lowest speeds are when they are defending or attacking.

Finally, another source of information are the teams' tactical maps, which demonstrate the teams' positions on the field during specific time periods. The Visualizer allows the user to interact and choose specific time frames to be displayed.

Figures 5.29 and 5.30 show these maps for two attack situations of each game.

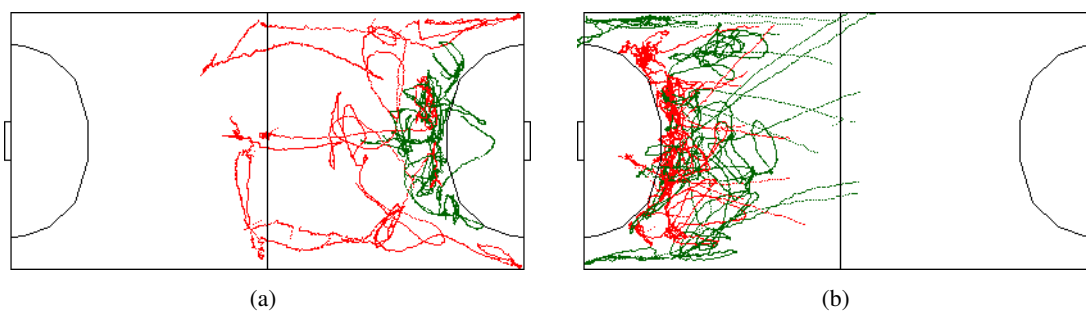


Figure 5.29: Positional maps for two game situations of Game 1: (a) team A (in red) attacking and team B (in green) defending (from frame to 620 to frame 1478), (b) team A defending and team B attacking (from frame 1712 to frame 2589).

The two previous images indicate that the first attack performed by team A was very fast and players tend to stay on the central area of the field (Figure 5.29(a)), while the second attack took longer and players were more spread around the 6 meter line (Figure 5.29(b)).

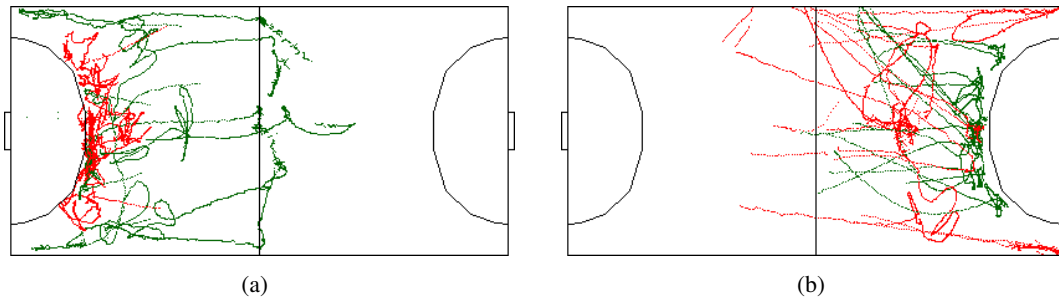


Figure 5.30: Positional maps for two game situations of Game 2: (a) team A attacking and team C defending (from frame 600 to frame 1160), (b) team C defending and team A attacking (from frame 1650 to frame 2120).

The two attacks of the second game seem more similar, nevertheless team A seems to be able to perform the attack nearer to the 6 meter line which also makes the defence to be more aggressive (Figure 5.30(b)) to better cover the 6 meter line. The attack from team C is performed in a more open way (Figure 5.30(a)) and therefore the defence is not so "glued" to the 6 meter line.

5.5 Software Interface

As described on section 4.1, the proposed architecture is composed of three main modules: the Acquisition System, the Processing System and finally by the Visualizer.

The following subsections provide an overview of these three modules.

5.5.1 Acquisition Module

The first module, the Acquisition System, uses the software provided by the manufacturer "IC Capture" (Figure 5.31) which allows configuring several parameters of the camera, such as the frame rate, exposure time, contrast, brightness and saturation, among others.

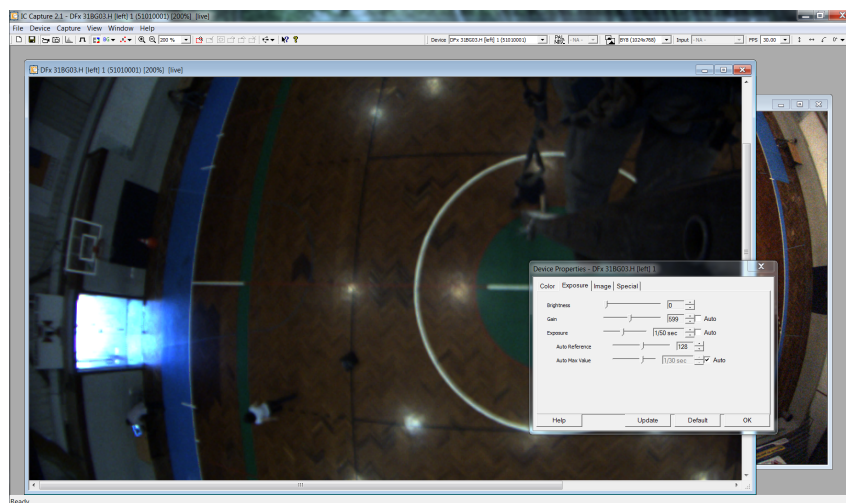


Figure 5.31: Acquisition System software print screen.

Additionally, this software provides the possibility of either using codecs to compress the video information or several uncompressed data ('Y800', 'RGB24', 'RGB32' and 'UYVY', 'BY8') to store the video.

An handball game lasts 60 minutes (two halves of 30 minutes), which at a frame rate of 30 fps and two cameras implies 216000 frames per game. In order to preserve the original image information and minimize the disk space used, the uncompressed BY8 format was chosen to store the videos.

5.5.2 Processing Module

The Processing Module is responsible for analysing the video sequences, identifying and tracking the players and generating a report with their positions along the time.

The application is composed of three different tabs: Play Video, Advanced and Debug, which will be described in the next sections.

5.5.2.1 Play Video Tab

To use this module it is necessary to select the videos that contain the games as well as to provide an empty image of the field as illustrated in Figure 5.32.

The application is prepared to analyse up to three video camera systems. In order to define the number of cameras that compose the system, the user must check the checkboxes on the right side of the respective video filename textbox.

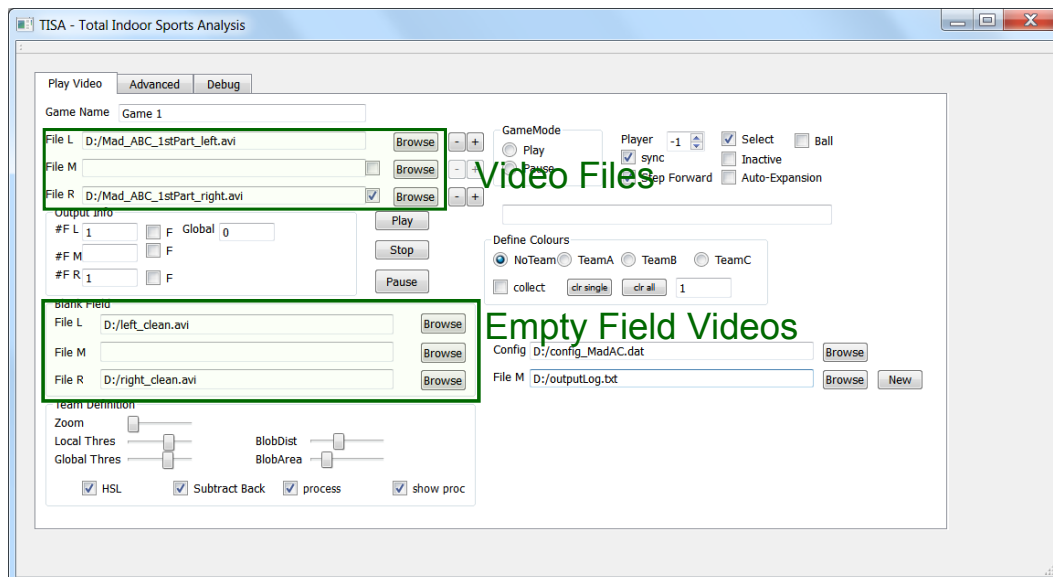


Figure 5.32: Processing Software Module main window with the videos browse menu highlighted.

Once the "Play" button is pressed, the video streams appear in different windows, along with a zoom window as can be seen in Figure 5.33. This figure illustrates a two cameras' system.



Figure 5.33: Processing Software Module showing the videos of a two cameras' system.

The zoom window magnifies the area surrounding the mouse, therefore if a more detailed information is required of a given area of the field, the user must position the mouse on that area and see the result on this window.

On this tab the user can also perform the video synchronization either by freezing a given video stream while it is being played using the check boxes before the "F", or using the '+' and '-' buttons when the *sync* checkbox is pressed as illustrated in Figure 5.34.

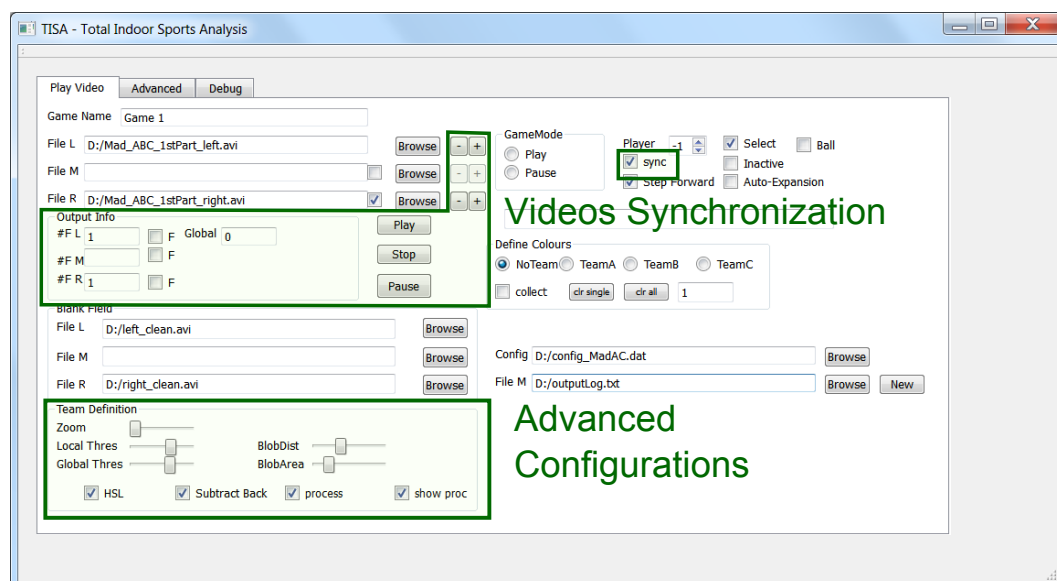


Figure 5.34: Processing Software Module main window with the synchronization and advance configuration menus highlighted.

Figure 5.34 also highlights advanced configurations that the user may adjust. The C_{ThresG} and C_{ThresL} parameters of the colour calibration process (Rule 0, subsection 4.2.1) can be tuned on the *Global Thres* and the *Local Thres* sliders, respectively, while the *interRLEMinDist* and *minRLESize* parameters of the blobs definition algorithm (Algorithm 2) can be adjusted by the *Blobdist* and *BlobArea* sliders, respectively.

Additionally, the advanced configuration region allows the user to select analysing the data in HSL format instead of RGB, to perform background subtraction or not, to process or not the video frame and to show it processed or not (Figure 5.35).

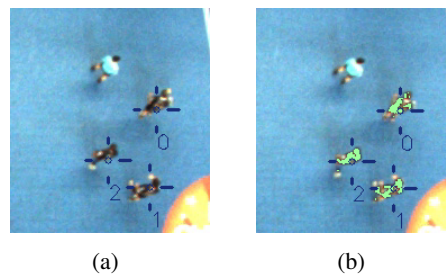


Figure 5.35: Example of how the videos are shown when the user: (a) checks the *show proc* checkbox and (b) when it does not.

Finally, this tab allows the user to perform the initial seeds configuration, select each player to be tracked, upload previous defined configurations (which contain the cameras' homographies and previous obtained colour subspaces of the teams) and define an output log file as highlighted in Figure 5.36.

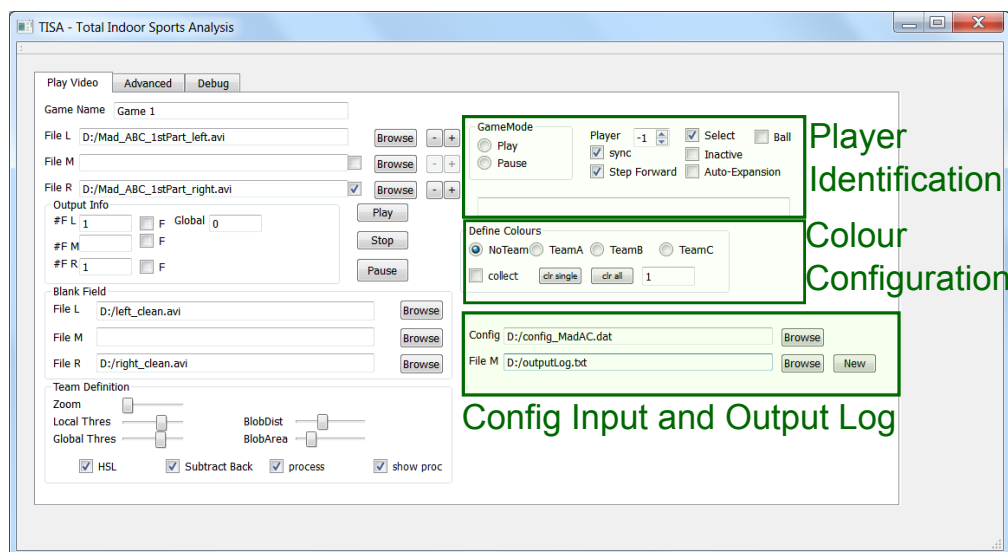


Figure 5.36: Processing Software Module main window with the colour configuration, player identification, input and output configuration files.

The Player Identification area allows the user to select a player number under the *Player* spin box or the ball in the *Ball* checkbox, and then click on a player/ball on the image. This process allows tagging a player which is then automatically tracked by the tracking algorithm.

The *Inactive* checkbox is used when a player is replaced by another and stays on the bench. Once the player is again on the field, the user must select again the player's number and click on the image to initiate the tracker.

The *Step Forward* checkbox allows the user to control the video streams using the '+' key, therefore the video streams will only move forward to the next frames in case the user presses this key. The *Auto-Expansion* checkbox allows enabling/disabling the auto-expansion process as defined on Sections 5.1.3 and 4.2.3.

On the *GameMode* area the user can select if the game is either in the *Game Stop* or *Game On* states (Section 4.5), by selecting the *Pause* or *Play* radio buttons, respectively.

On the Colour Configuration area, the user can select to collect or not more colours triplets by checking/unchecking the *collect* checkbox. In case the user checks it and clicks on any pixel of the video frames, the colour triplet will be added into the selected team colour subspace. The colour subspace can be chosen using the radio buttons with the *Team A*, *Team B* and *Team C* indications.

The user can also reset either one of the teams colour subspace using the *clr single* button and the teams radio buttons or all the colour subspaces using the *clr all* button. The text box at the bottom of the highlighted zone allows defining how many frames are skipped from being shown (the higher this value the fastest the application will be).

Finally, this tab provides the user the possibility to load an already defined configuration file with the colour calibration subspaces and the homography configuration. Figure 5.37 illustrates the content of one of these files for a three cameras' system.

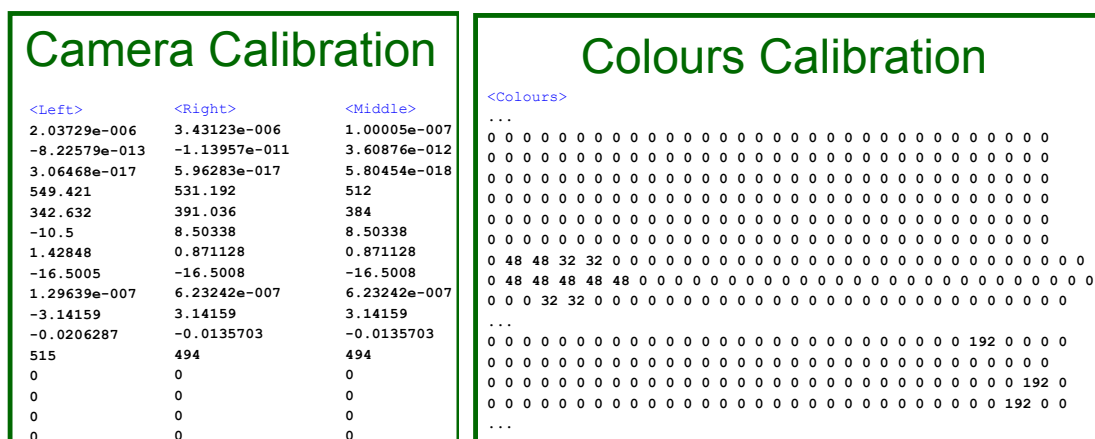


Figure 5.37: Example of a configuration file used on the Processing Module.

The configuration file contains first the information concerning the cameras' homographies which is used by Equations 4.4 and 4.6 and then the colour subspaces information. In order to reduce this file size, the colour information is stored in 2 bits which allows the 4 belonging degrees as defined on Table 4.1.

```

<Game> Game 1 </Game>
<!--
<Frame> cam#    global_tStamp    specific_tStamp -->
<Frame> 0        1                2
<!--    <Player>    ID        Px        Py        Vx        Vy        Team Active
-->
</Player>
<Player> 0    -11.2127    0.912046    1.39295    -0.525019    3    1
</Player>
<Player> 1    -2.88691    4.12677    -3.26187    -0.26696    3    1
</Player>
<Player> 2    -4.42996    1.85421    -2.31672    0.212003    3    1
</Player>
<Player> 3    -3.20239    2.27114    0.348744    0.0232526    3    1
</Player>
<Player> 4    -0.291359    0.609242    0.002909    -0.0116144    3    1
</Player>
<Player> 5    -2.9505    -0.439706    -3.03646    0.47782    3    1
</Player>
<!--    <PlayMode> STOP|ON    </PlayMode>    -->
<PlayMode> 1    </PlayMode>
</Frame>
<Frame> 1        1                72
<Player> 6    5.33601    6.53989    -0.142    -0.525437    2    1    </Player>
<Player> 7    8.81735    4.69969    -0.00759417    -0.718002    2    1    </Player>
<Player> 8    14.3989    4.0443    -0.372518    0.21348    2    1    </Player>
<Player> 9    13.0291    -0.47483    -3.957    -0.732669    2    1    </Player>
<Player> 10    11.1782    0.072656    0.987784    -0.943403    2    1    </Player>
<Player> 11    11.232    -0.361878    0    0    2    1    </Player>
<PlayMode> 1    </PlayMode>
</Frame>
...
<Frame> 0        18686            18687
<Player> 2    -2.16027    4.4989    -4.52146    -1.45046    3    1    </Player>
<Player> 39    -6.08549    -1.866    -3.68739    -0.235844    3    1    </Player>
<Player> 6    -6.34658    -6.70417    -6.63165    -1.64484    2    1    </Player>
<PlayMode> 1    </PlayMode>
</Frame>
<Frame> 1        18686            18758
<Player> 1    7.93262    3.73783    -4.63842    0.905377    3    1    </Player>
<Player> 5    8.8442    6.7194    -3.85849    -1.02883    3    1    </Player>
<Player> 0    -1.49455    -4.478    -6.41218    -0.0894739    3    1    </Player>
<Player> 7    11.8513    1.75785    -2.40042    -0.565399    2    1    </Player>
<Player> 11    6.43811    -6.13074    -4.23924    -1.31699    2    1    </Player>
<Player> 27    5.06404    -3.40299    -4.00836    -1.63895    2    1    </Player>
<Player> 8    11.7073    8.26436    0.0775286    -0.171691    2    1    </Player>
<Player> 38    3.2829    7.29432    -2.9559    0.977861    2    1    </Player>
<Player> 42    1.82865    3.98962    -3.08908    3.66488    3    1    </Player>
<PlayMode> 1    </PlayMode>
</Frame>

```

Figure 5.38: Example of an output log file.

Finally, the user can define either an existing log file to append information by pressing on the *Browse* button or create a new log file by pressing the *New* button. The log file will store information concerning the players' position (P_x and P_y), velocity (V_x and V_y), as well as the player ID, team (T) and if his/her state on the field is active (1) or inactive (0). Additionally, this file stores information to be used for cameras' synchronization (cam#, global and specific) and if the game is in Play Mode (1) or Pause (0). Figure 5.38 shows an example of an output log file.

5.5.2.2 Advanced Tab

On the Advanced Tab it is possible to perform the camera calibration, which includes determining the parameters for the camera homography (Equation 4.6) and barrel distortion (Equation 4.4). The following image (Figure 5.39) shows a print screen of this tab.

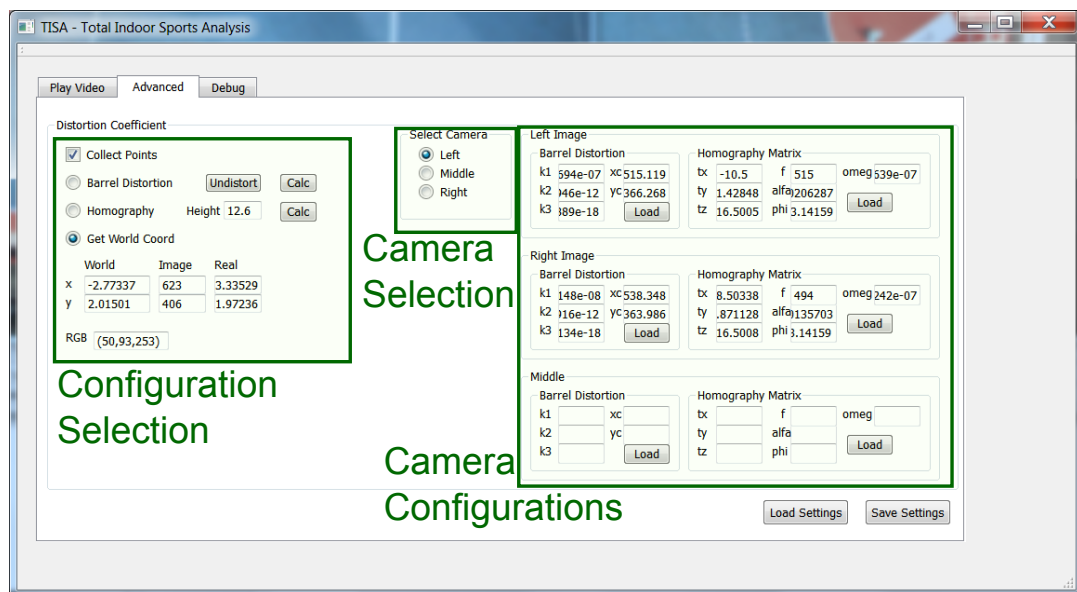
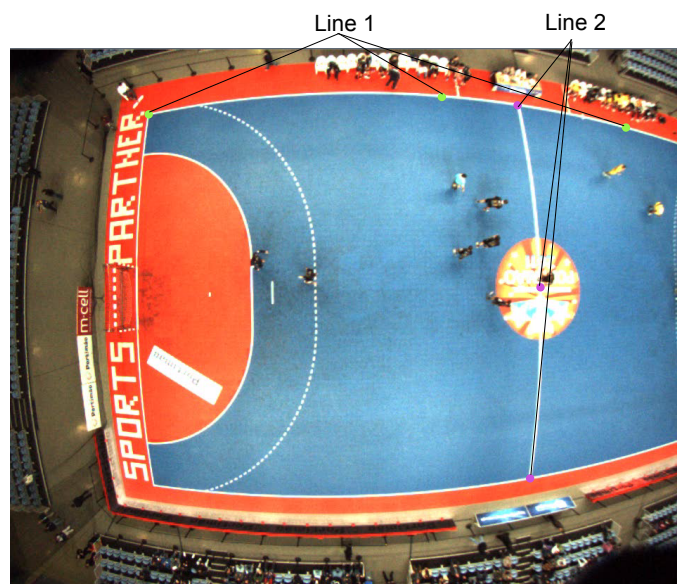


Figure 5.39: Advanced tab of the Processing Software Module.

On the Camera Selection zone the user can select which camera will be configured. Once the camera is chosen, the user can, afterwards, select between three types of operations as highlighted on the Configuration Selection area.

If the user selects the option *Barrel Distortion* the application will consider the points selected by the user for the calculation of the barrel distortion coefficients (k_1, k_2, k_3, x_c, y_c). When selecting the points the user must be careful in choosing straight lines and providing three points for each line (two at the edges and one at the centre), as illustrated in Figure 5.40.

Figure 5.40: Example of points selection of two lines used to determine the barrel distortion coefficients (k_1, k_2, k_3, x_c, y_c).

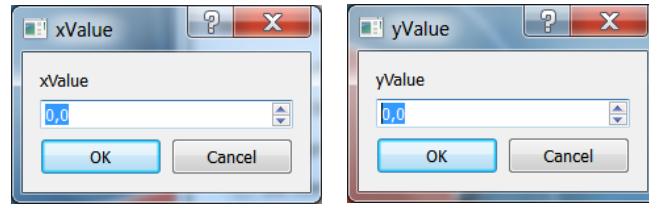


Figure 5.41: Windows provided by the application for the user to introduce the real world coordinates when performing the Homography calibration.

Once the user finishes defining all the lines it is possible to determine the barrel distortion coefficients by pressing on the *calc* button next to the *Barrel Distortion* radio button. The result of the calculated coefficients can be seen by pressing on the *Undistort* button, which will provide the images displayed in Figures 4.5(b) and 4.5(d). The barrel distortion coefficients are determined using the Levenberg-Marquardt (Marquardt (1963)) optimization algorithm.

In case the user intends to determine the Homography coefficients (f , (c_x, c_y) , ϕ , ω , α , (T_x, T_y, T_z)) the *Homography* radio button must be selected.

The procedure to determine the coefficients is similar to the one used to determine the barrel distortion coefficients, so the user must select a point on the field and the application prompts two subsequent windows where the real world coordinates must be introduced (Figure 5.41). The third coordinate missing, the height of the camera, is introduced on the *Height* textbox.

Like before, the user can start the process of determining the coefficients by pressing on the *calc* button next to the *Homography* radio button.

The Homography coefficients are also determined via the Levenberg-Marquardt optimization algorithm.

The final radio button, *Get World Coord*, allows the user to obtain, for a given pixel on the image, the real world coordinates. In case the user selects this option and picks a pixel on any of the video streams, the lower text boxes will be filled in with the image coordinates, the world coordinates on the ground level and the real player coordinates at 1.2m from the ground.

The *Load Settings* and *Save Settings* buttons allow the user to reload the parameters that were stored on the configuration file, or update the new values into the configuration file.

Finally, the Camera Configuration shaded area shows the calculated coefficients for each camera.

5.5.2.3 Debug Tab

The Debug tab allows the user to visualize each team colour subspaces through a 3D RGB cube (Figure 5.42). This RGB cube was implemented using OpenGL (Group (2015)).

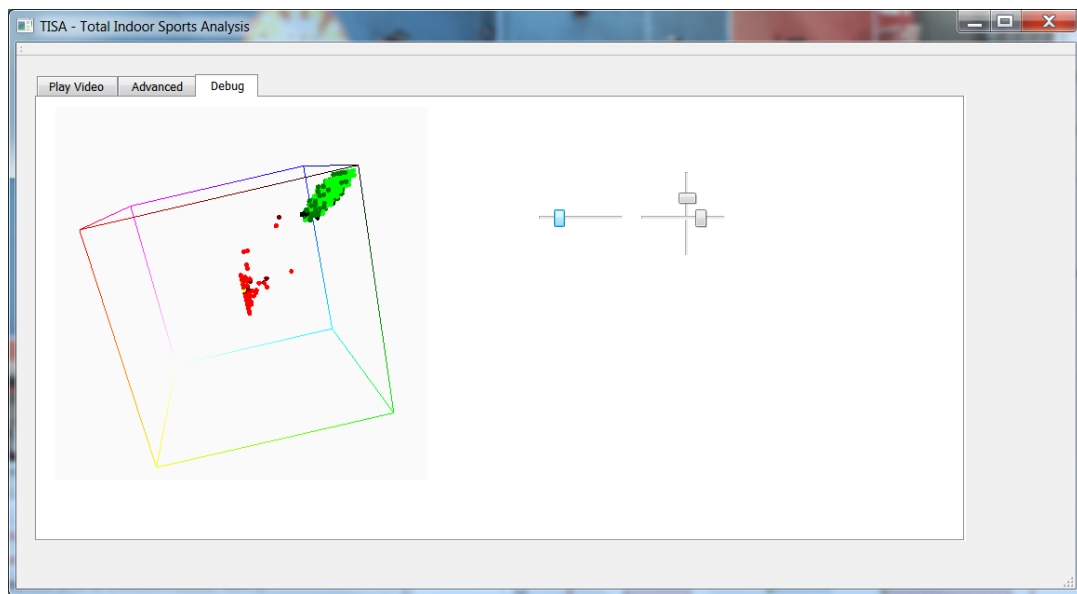


Figure 5.42: Debug tab of the Processing Software Module.

The sliders on the left side allow rotating the cube in the x, y and z directions.

5.5.3 Visualizer Module

The Visualizer Module allows the user to see the final video, with the two video streams aggregated to form a single image as the one shown in Figure 5.22. This application also allows the user to create a log file with certain game events that can be chosen by the user, enable the automatic detection of the game phase and defence formation.

Despite the previous modules allowing configurations up to 3 cameras, this module, at the moment, is only able to upload information from a system composed of two cameras. Nevertheless, and given the files configuration and the fact that the players' positions are in real world coordinates, this add-on is quite straightforward.

This module is composed of two windows plus the complete undistorted image as shown in Figure 5.43.

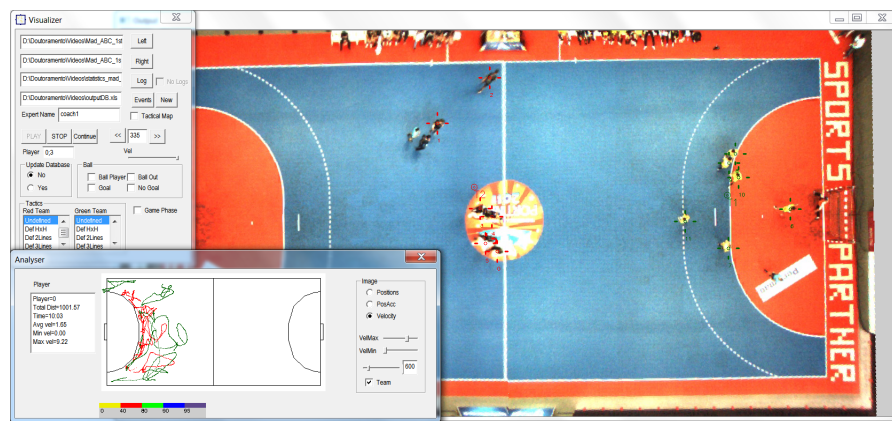


Figure 5.43: Visualizer Software Module showing the undistorted final image with the players highlighted, along with the Visualizer and Analyser windows.

5.5.3.1 Visualizer Window

On the Visualizer Window the user is able to select the two video streams to be shown by choosing the *Left* and *Right* buttons on the User Files Input area as highlighted on the following image (Figure 5.44).

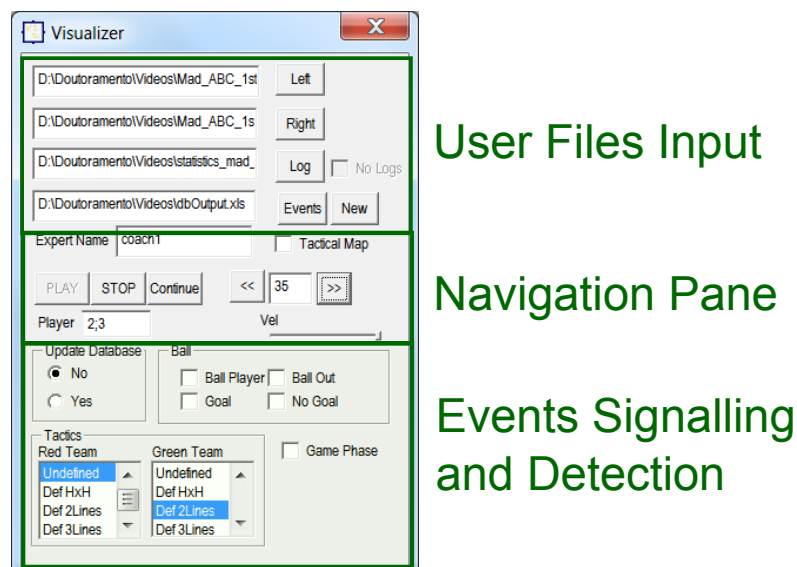


Figure 5.44: Visualizer Software Window with the different areas highlighted.

The *Log* button allows selecting the log file generated by the Processing Module (Section 5.5.2). In case the user does not require a log and just wants to see the two video streams merged it is possible by checking the *No Logs* checkbox.

Although it is possible not to use a Log file it is advisable, since the application uses the camera information to perform the video synchronization. Additionally, with the log file it is possible to see the players highlighted on the field as well as the teams' centre of mass.

On the *Events* and *New* buttons, the user is able to store the players' positional information in an *.xls format along with the events. An example of a file is shown in Figure 5.45

The *Events* button allows to load an existing file to append more information, while the *New* button allows to generate a new log file.

EXPERT	FRAME	REFGOALX	REFGOALY	P1NUMBER	P1TEAM	P1EVENT	P1DIST	P1ANG	P2NUMBER	P2TEAM	P2EVENT	P2DIST	P2ANG	P3NUMBER	P3TEAM	P3EVENT	P3DIST	P3ANG
coach1	46	20	0	0	3	:Transition	26.892	3.077	1	3	:Transition	23.307	2.934	2	3	:Transition	24.811	2.992
coach1	47	20	0	0	3	:Transition	26.816	3.076	1	3	:Transition	23.289	2.931	2	3	:Transition	24.834	2.991
coach1	48	20	0	0	3	:Transition	26.743	3.075	1	3	:Transition	23.287	2.929	2	3	:Transition	24.849	2.99
coach1	49	20	0	0	3	:Transition	26.66	3.074	1	3	:Transition	23.298	2.929	2	3	:Transition	24.854	2.989
coach1	50	20	0	0	3	:Transition	26.583	3.073	1	3	:Transition	23.294	2.928	2	3	:Transition	24.832	2.987
coach1	51	20	0	0	3	:Transition	26.514	3.072	1	3	:Transition	23.304	2.927	2	3	:Transition	24.831	2.985
coach1	52	20	0	0	3	:Transition	26.451	3.071	1	3	:Transition	23.294	2.926	2	3	:Transition	24.829	2.985
coach1	53	20	0	0	3	:Transition	26.388	3.071	1	3	:Transition	23.316	2.926	2	3	:Transition	24.827	2.983
coach1	54	20	0	0	3	:Transition	26.331	3.07	1	3	:Transition	23.344	2.926	2	3	:Transition	24.809	2.981
coach1	55	20	0	0	3	:Transition	26.271	3.069	1	3	:Transition	23.393	2.927	2	3	:Transition	24.787	2.98
coach1	56	20	0	0	3	:Transition	26.212	3.068	1	3	:Transition	23.417	2.926	2	3	:Transition	24.79	2.978
coach1	57	20	0	0	3	:Transition	26.146	3.067	1	3	:Transition	23.45	2.925	2	3	:Transition	24.794	2.977
coach1	58	20	0	0	3	:Transition	26.079	3.066	1	3	:Transition	23.5	2.926	2	3	:Transition	24.792	2.975
coach1	59	20	0	0	3	:Transition	26.014	3.065	1	3	:Transition	23.506	2.925	2	3	:Transition	24.767	2.972
coach1	60	20	0	0	3	:Transition	25.965	3.064	1	3	:Transition	23.531	2.925	2	3	:Transition	24.764	2.971

P11TEAM	P11EVENT	P11DIST	P11ANG	P12NUMBER	P12TEAM	P12EVENT	P12DIST	P12ANG	BALLSTATUS	BALLDIST	BALLANG	TEAMANUMBER	TEAMAFORM	TEAMBNUMBER	TEAMBFORM
2	:Transition	7.577	3.114	11	2	:Transition	8.954	3.181	P1_T3	23.307	2.93	3	Und		2 Und
2	:Transition	7.575	3.096	11	2	:Transition	8.968	3.183	P1_T3	23.289	2.93	3	Und		2 Und
2	:Transition	7.553	3.079	11	2	:Transition	9.006	3.185	P1_T3	23.287	2.92	3	Und		2 Und
2	:Transition	7.513	3.076	11	2	:Transition	9.014	3.185	P1_T3	23.298	2.92	3	Und		2 Und
2	:Transition	7.482	3.085	11	2	:Transition	9.007	3.186	P1_T3	23.294	2.92	3	Und		2 Und
2	:Transition	7.415	3.094	11	2	:Transition	8.974	3.185	P1_T3	23.304	2.92	3	Und		2 Und
2	:Transition	7.393	3.091	11	2	:Transition	8.931	3.179	P1_T3	23.294	2.92	3	Und		2 Und
2	:Transition	7.417	3.067	11	2	:Transition	8.92	3.173	P1_T3	23.316	2.92	3	Und		2 Und
2	:Transition	7.399	3.085	11	2	:Transition	8.926	3.166	P1_T3	23.344	2.92	3	Und		2 Und
2	:Transition	7.376	3.079	11	2	:Transition	8.965	3.165	P1_T3	23.393	2.92	3	Und		2 Und
2	:Transition	7.333	3.074	11	2	:Transition	9.008	3.166	P1_T3	23.417	2.92	3	Und		2 Und
2	:Transition	7.309	3.065	11	2	:Transition	9.032	3.165	P1_T3	23.45	2.92	3	Und		2 Und
2	:Transition	7.271	3.055	11	2	:Transition	9.045	3.164	P1_T3	23.5	2.92	3	Und		2 Und
2	:Transition	7.246	3.043	11	2	:Transition	9.047	3.166	P1_T3	23.506	2.92	3	Und		2 Und
2	:Transition	7.238	3.037	11	2	:Transition	9.05	3.17	P1_T3	23.531	2.92	3	Und		2 Und

Figure 5.45: Example of the log file generated by the Visualizer Module.

The generated file contains the name of the person who made the categorization, the global frame number, the reference goal, since the players' coordinates are given in polar coordinates (it was a request from the sport's experts), each player number, team and positions on the field, along with an extra column that in the future will store events performed by the players that have been automatically detected.

The final columns contain information about the ball (player that is holding it and in that case also its position on the field) and the team formation in case it is either in attack or defence. For this specific log, this information is supplied by the user so that it can be afterwards compared with the one obtained automatically as defined on Section 4.4.

The user can navigate through the video using the *Stop*, *Start* and *Pause* buttons on the Navigation Pane highlighted in Figure 5.44.

The “«” and “»” buttons allow moving backwards/forwards while the video is paused, nevertheless the user can also go directly to a specific frame by editing the text box between these two buttons. The *Vel* slider controls the speed at which the video is shown.

In this area the user can also introduce an identifier on the *Expert Name* text box, which will be stored on the first column of the log file (Figure 5.45). The *Player* text box indicates the player number and team of the player selected by the user (the user can select a player by clicking on it on the global video). Finally the *Tactical Map* checkbox allows commuting between the real image and a tactical map with the players' positions as displayed in Figure 5.46.

On the bottom area (Events Signalling and Detection) the user is able to define some events related with the ball and the team formation.

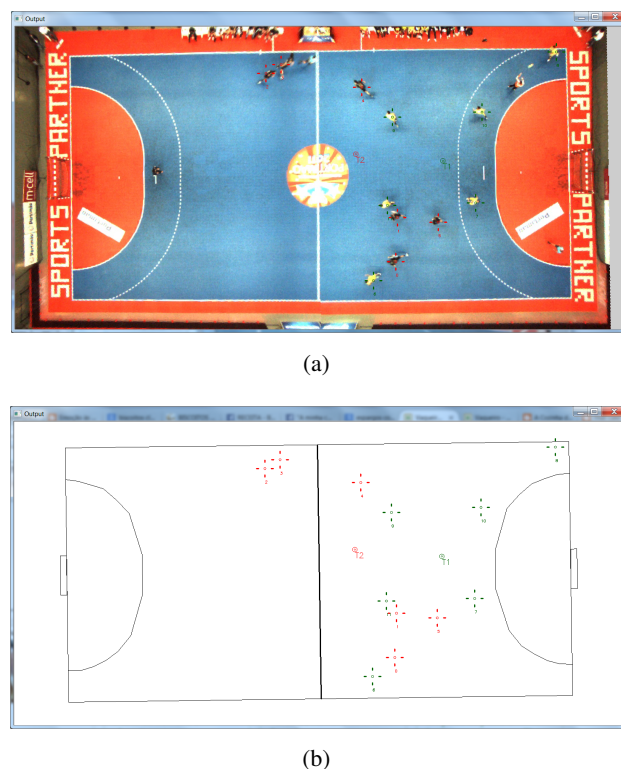


Figure 5.46: Global video window showing: (a) original image, (b) positional map with players highlighted.

On the *Ball* group the user can signal that a player is holding the ball (*Ball Player*), the ball went out of the field (*Ball Out*) or that there was a successful or unsuccessful goal trial (*Goal* or *No Goal*). These events will be stored into the ball columns of the output log file.

On the *Update Database* group the user may choose to whether overwrite or not values that were already stored on the log file using the radio buttons.

The *Tactics* group is composed of two lists, one for each team, where the user is able to define not only the game phase for each team, but also the tactic that is being used.

The last element, the *Game Phase* checkbox allows enabling the automatic game phase and defence team formation categorization according to the methodology defined on Section 4.4.

This application is still on a development phase and therefore needs some improvements, namely the possibility to visualize up to three video streams so that it can be compatible with the Processing module and also the possibility to include the camera calibration via a file instead of being hard coded.

5.5.3.2 Analyser Window

The second window of the Visualizer Module, allows the user to verify the players' statistics and obtain the maps illustrated on Section 5.4. Figure 5.47 shows the information displayed on this window.

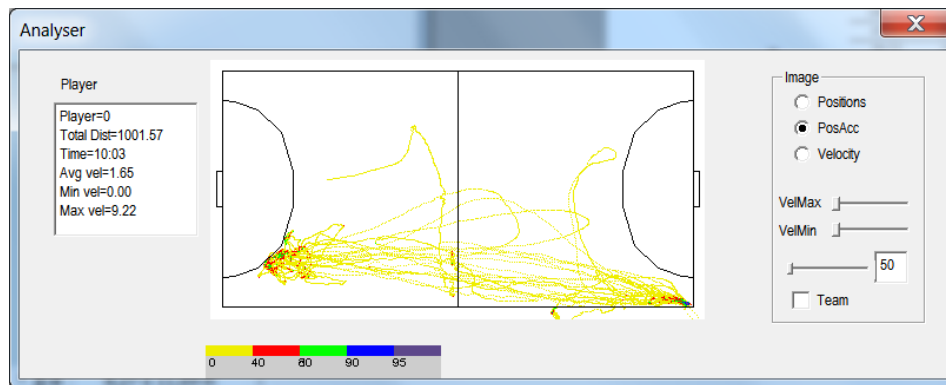


Figure 5.47: Analyser Software Window.

On the Player area the user can have access to the statistics of the player that was selected, while on the radio buttons the user has the possibility to change the type of information that is shown on the map.

The possible maps include the positions along the time (*Positions* radio button), the accumulated positions (*PosAcc* radio button) which indicate in which area of the field the players stayed longer or the speed (*Velocity* radio button) of a player during the game. The sliders next to the *VelMax* and *VelMin* tags allow choosing between which values the speeds are shown on the map.

Finally, the bottom slider, text box and *Team* can be used to draw the positional maps of both teams. To enable this option the user must check the *Team* checkbox and define the starting time on the slider and the desired period duration on the text box.

5.6 Game Model Validation

This section presents the validation of the proposed game model using the information from one of the already analysed games. This validation consisted in visualizing the video footage with the Visualizer module (Section 5.5.3) during 3600 frames of Game 1 (which corresponds to around 2 minutes of game) and reflecting the game state on the several layers of the Hierarchical Coloured Petri Net developed on the CPN Tools software.

The next subsections illustrate these results following the model's hierarchy. Whenever pertinent the results also illustrate the players' positions using video frames.

5.6.1 First Hierarchical Level

As already explained, from a higher perspective, the handball game is defined by two substitution transitions: **Game ON** and **Game Stop** and several states that form the bridge between these two transitions.

For the analysed time period the **Game Mode** subnet assumes the states defined in Figure 5.48. This chart only presents the active states and transitions during this time period, because otherwise it would become too confusing and would not add extra information.

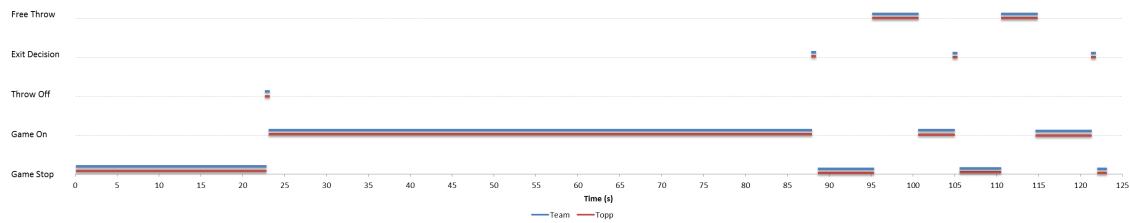
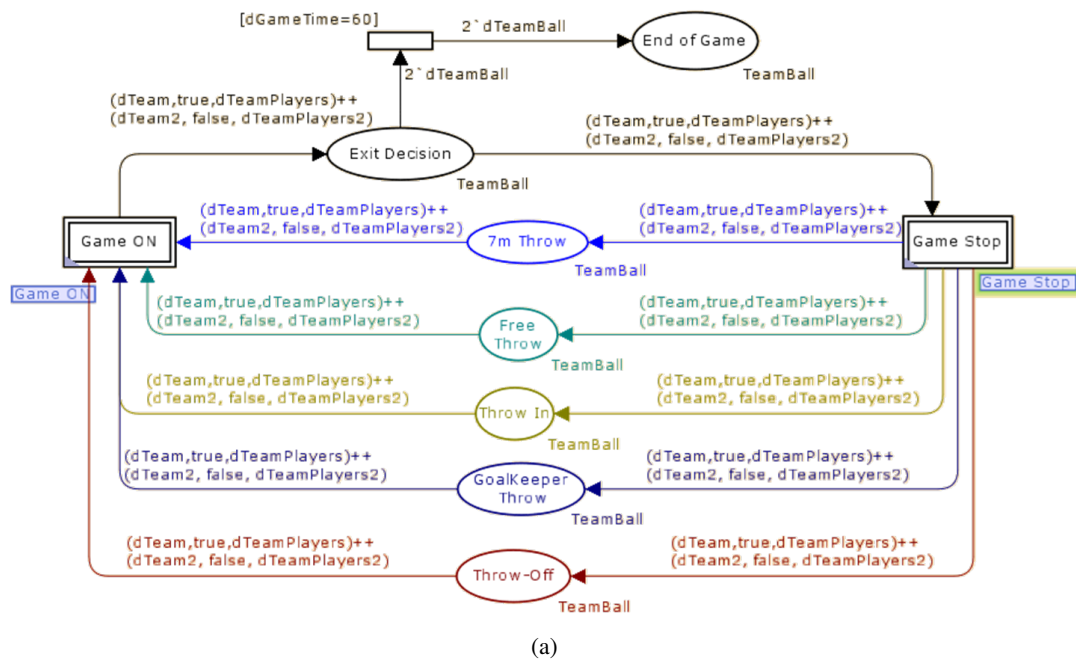
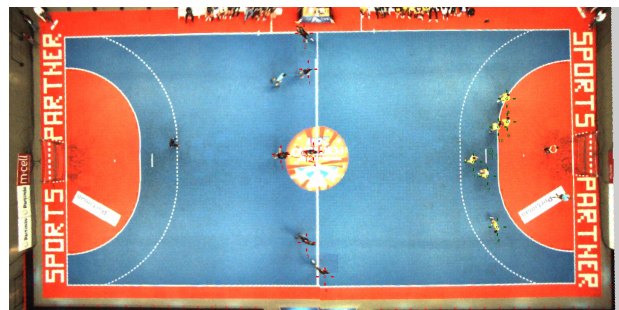


Figure 5.48: Temporal chart of the **Game Mode** Petri Net states and transitions during the analysed game period.

Analysing the chart it is possible to verify that the game started at frame 620 (20.6s), after a **Throw-Off**. This way, at frame 619 the Petri Net state is the one defined in Figure 5.49(a), while the players' positions on the field are shown in Figure 5.49(b).

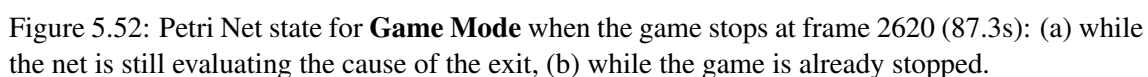


(a)



(b)

Figure 5.49: (a) Petri Net state for the **Game Mode** level (the green highlight around the **Game Stop** substitution transition indicates the Petri Net state) and (b) players' distribution on the field at frame 619 (20.6s).



The game is again resumed at frame 3001 (100s) with the attacking team performing a **Free-Throw** (Figure 5.53) which then leads the game back into the **Game On** substitution transition. The teams' positions on the field at this frame can be seen in Figure 5.54. Both teams remain in the **Free-Throw** from frame 2844 until frame 3001.

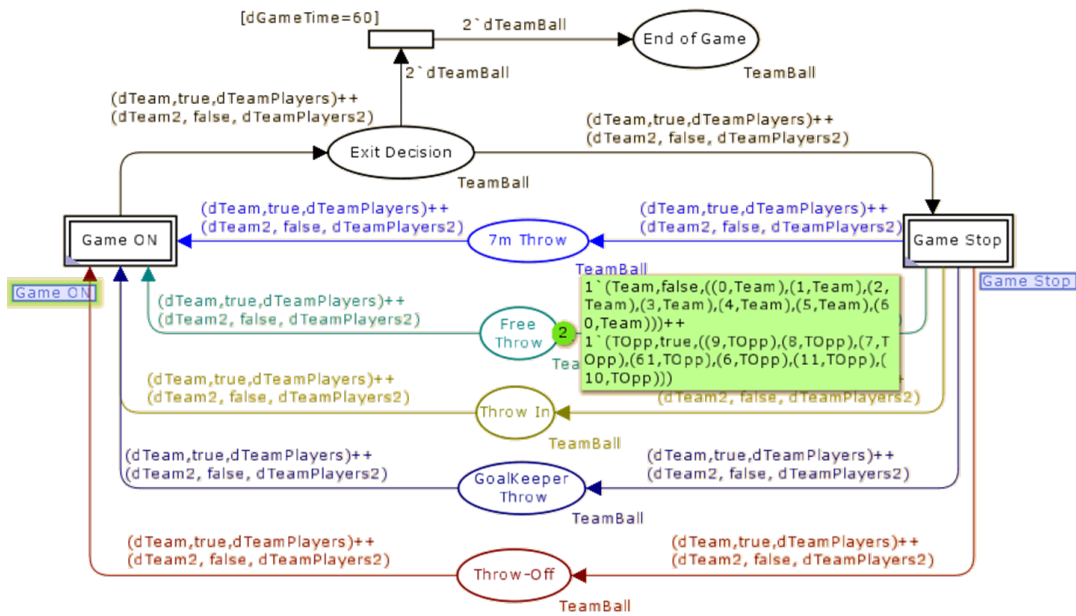


Figure 5.53: Petri Net state for **Game Mode** while the team is preparing for the **Free Throw** when the game resumes back to the **Game On** substitution transition at frame 3001 (100s).



Figure 5.54: Players' distribution on the field at frame 3001 (100s).

On frame 3136 (104.5s) the defending team makes a foul and the Petri Net state changes back to the **Game Stop** substitution transition after passing by the **Exit Decision** state. The game is restarted again at frame 3437 after passing by the **Free Throw** state.

Finally the attacking team is able to score a goal which also leads to the **Game Stop** substitution transition at frame 3640 (121.3s).

5.6.2 Second Hierarchical Level

The second hierarchical level of the proposed Petri Net details the two substitution transitions of the first level: **Game On** and **Game Stop** transitions.

As already seen on the previous subsection, 5.6.1, for the analysed time period there are four phases of **Game Stop** substitution transition and three phases of **Game ON** substitution transition. The two following images (Figures 5.55 and 5.56) detail the several states by which each substitution transition evolves during this time period.

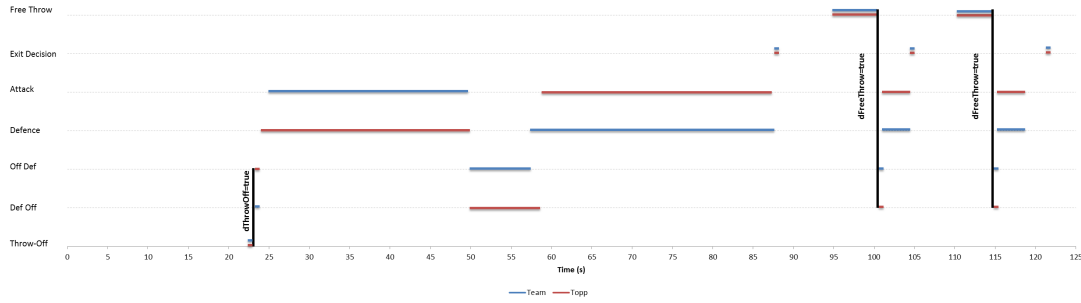


Figure 5.55: Temporal chart of the **Game ON** Petri Net states and transitions during the analysed game time period.

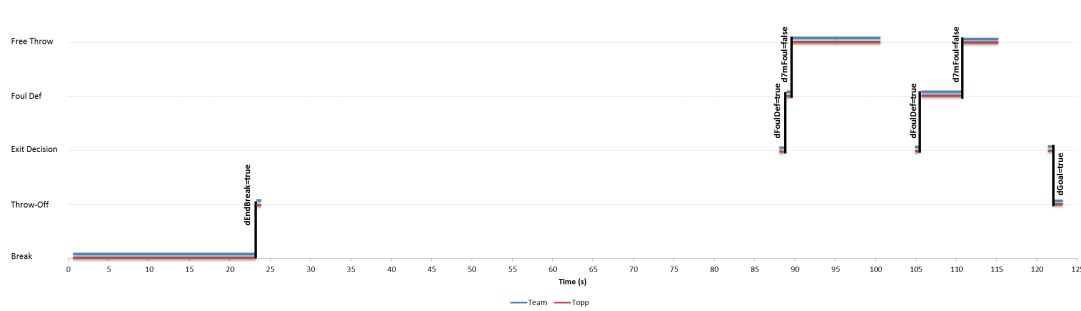


Figure 5.56: Temporal chart of the **Game Stop** Petri Net states and transitions during the analysed game time period.

Before the game starts at frame 620 (20.6s), the Game Stop Petri net has both teams, *Team* and *Topp*, on state **Break** and only one transition can be fired, which happens when variable `dEndBreak` gets the truth value. This transition leads both teams to state **Throw-Off**, as illustrated in Figures 5.57 and 5.58.

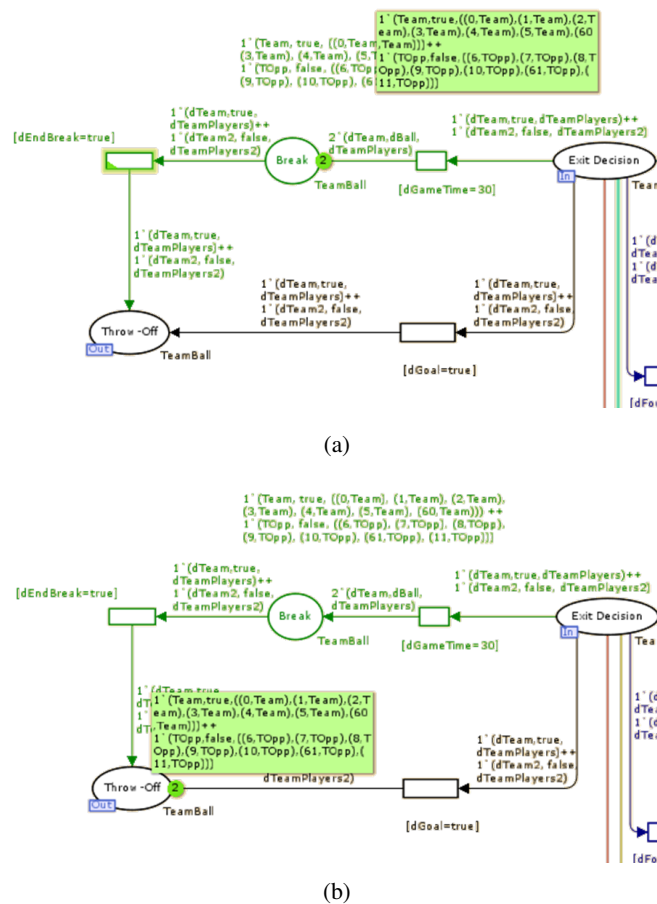


Figure 5.57: Game Stop Petri net at: (a) frame 618 (20.6s) and (b) frame 619 (20.6s) (partial view of the nets).

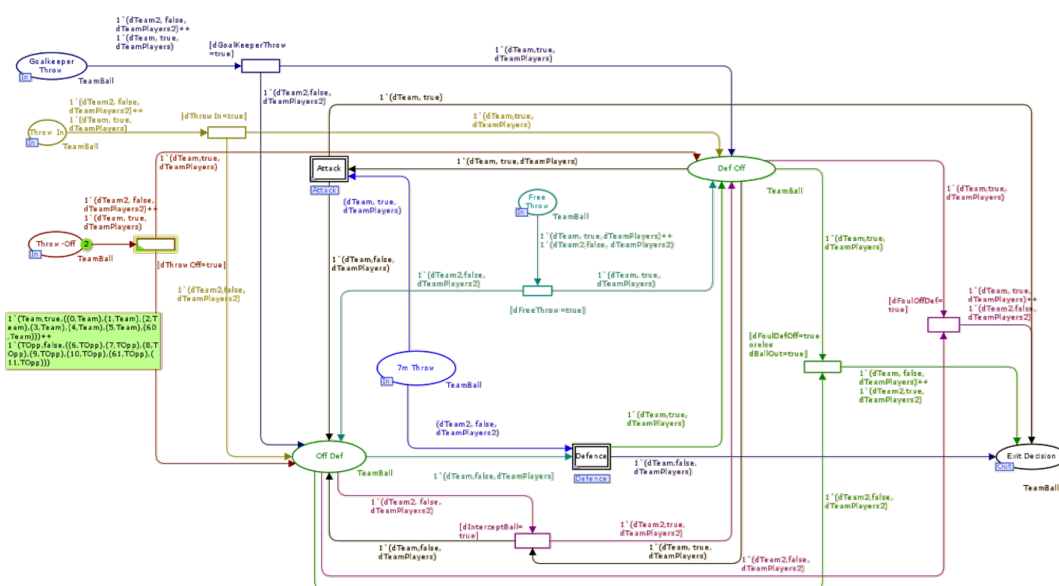


Figure 5.58: Game ON Petri net at frame 619 (20.6s).

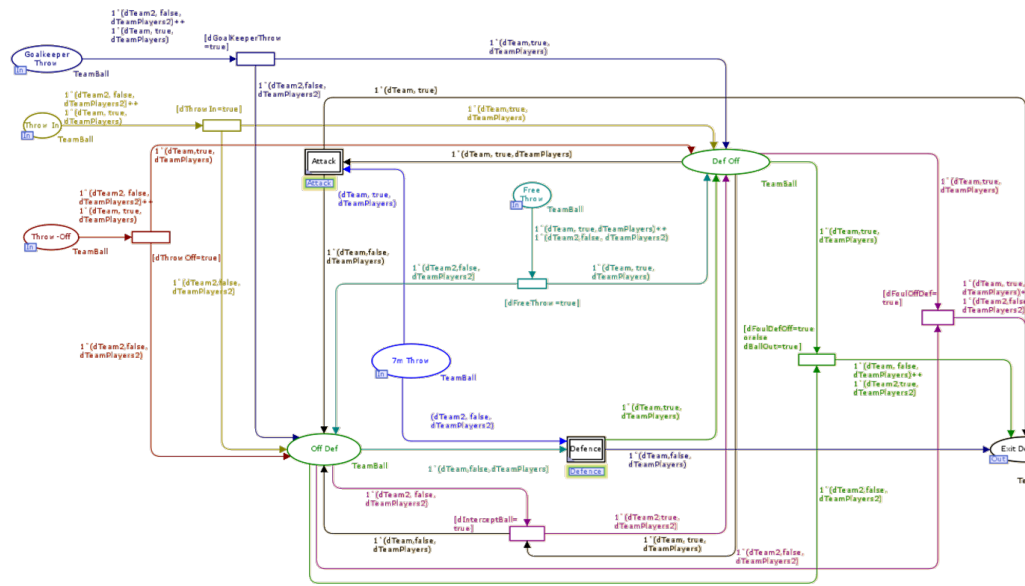


Figure 5.61: Game ON Petri net at frame 752 (25s), when *Team* is attacking and *TOpp* is defending.

At frame 1496, the defending team is able to intercept the ball and players from both teams automatically start moving towards their new positions assuming the **Def Off** and **Def Off** states, as denoted in Figure 5.62.

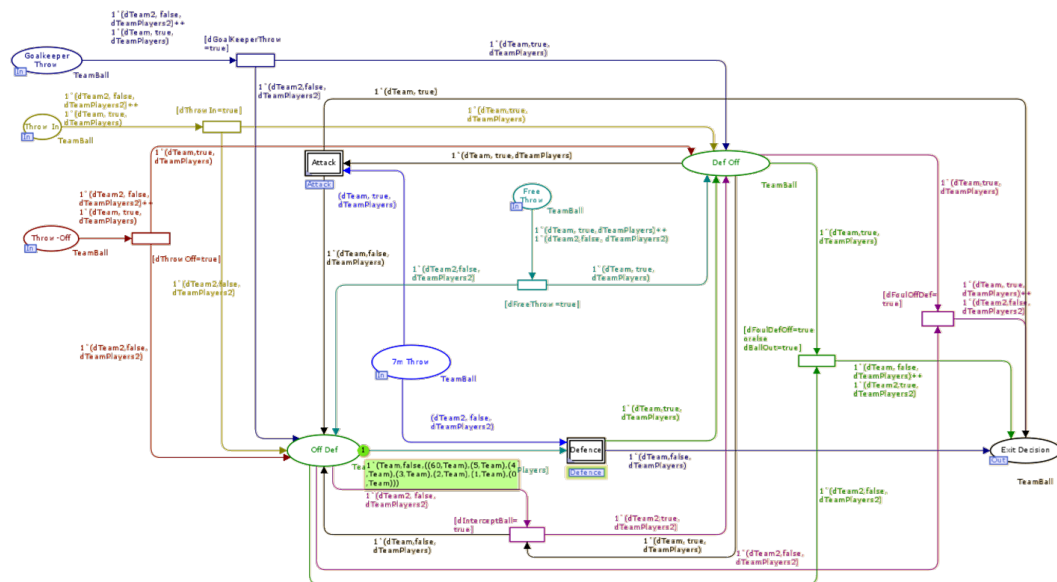


Figure 5.62: Game ON Petri net at frame 1496 (49.8s) after the defending team intercepting the ball.

This transitional phase lasts until frame 1759, at which, both teams assume their attacking and defending positions.

At frame 2620 (87.3s) a defensive player performs a foul, which leads the **Game ON** subnet into the **Exit Decision** state (Figure 5.63).

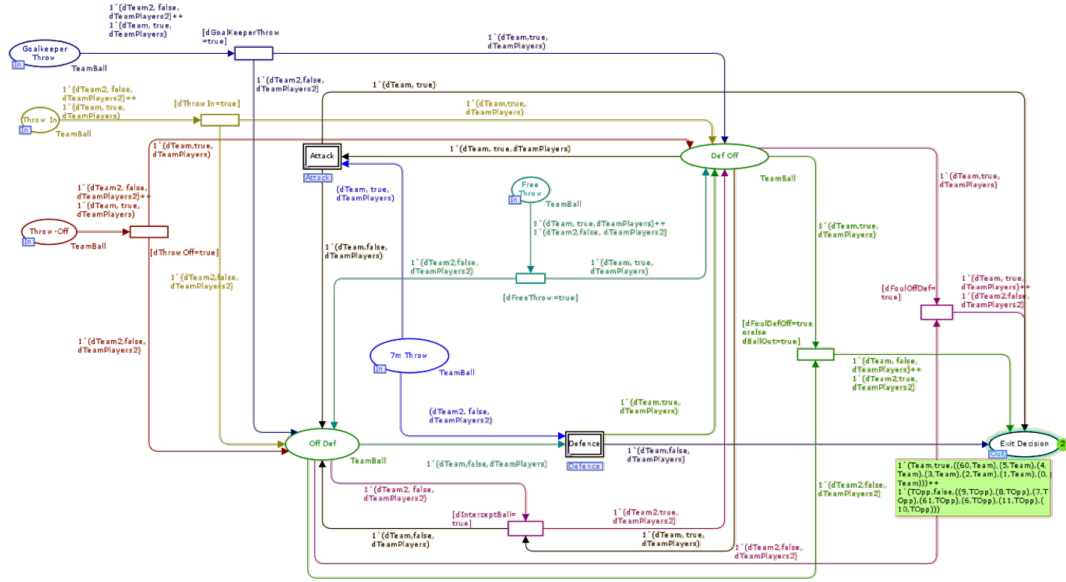


Figure 5.63: **Game ON** Petri net at frame 2620 (87.3s), after a defensive foul.

On the **Game Stop** subnet the teams' states evolve from the **Exit Decision** state to the **Foul Def** state, since variable $dFoulDef$ is true. These states are shown in Figure 5.64.

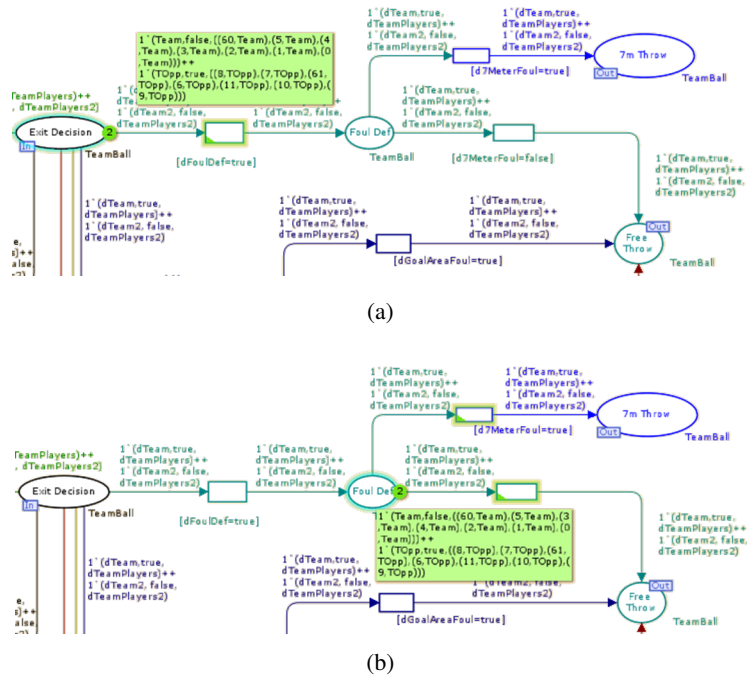


Figure 5.64: Petri Net state for (**Game Stop**) when the defending team performs a foul (partial views of the net).

On frame 2844, both teams start preparing for the throw and since it was the defending team who committed the foul a **Free Throw** is awarded as indicated by Figures 5.65 and 5.66.

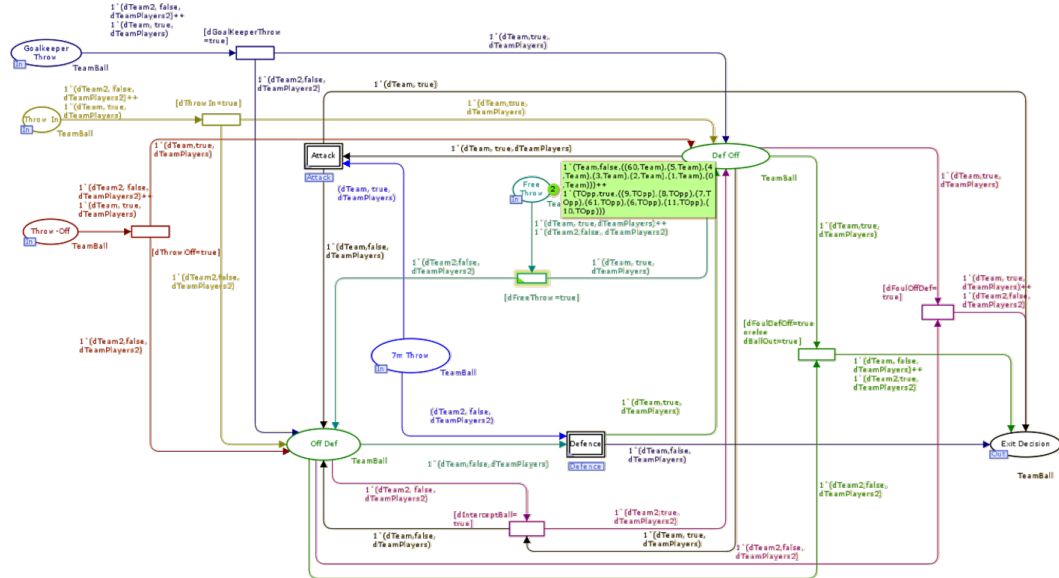


Figure 5.65: **Game ON** Petri net at frame 2844 (94.8s), when teams are preparing to perform a **Free Throw**.

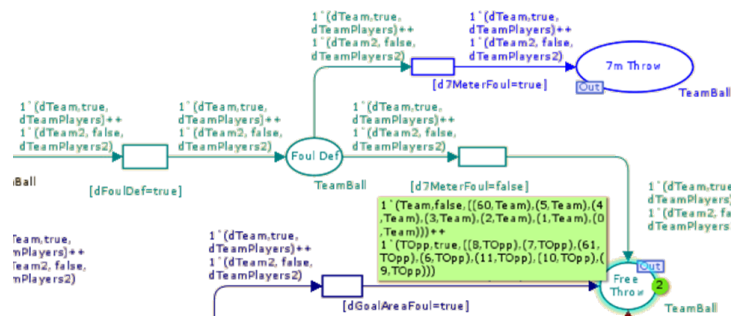


Figure 5.66: **Game Stop** Petri net at frame 2844 (94.8s), when teams are preparing to perform a **Free Throw**.

The game stays stopped until frame 3001 (100s), when the attacking team performs the Free Throw (variable $dFreeThrow=true$) and the game restarts again.

At time 104.5s (frame 3135) the defending team commits again a defending foul by entering on its own area to gain advantage over the attacking team (Figure 5.67), which causes the game to stop again.



Figure 5.67: Players' distribution on the field at frame 3135 (104.5s), when the defending team commits a foul.

The *Free Throw* is executed at time 114.5s making the game go into the **Game On** state. A few seconds later (121.3s) it stops again with the attacking team scoring a goal, which leads to a **Throw-Off**.

5.6.3 Third Hierarchical Level

The third hierarchical level of the game model includes the most detailed information, namely the interactions between the players as well as between the players and the ball.

Figure 5.68 illustrates the first attack, where the *Team* attacks the *TOpp* team, while Figure 5.69 illustrates the defending team behaviour.

As can be seen, once the players reach the attacking position, three transitions are fired and the players pass by the **Enter Attack** and **Players Waiting Throw** states, before occupying one of the two possible attacking states: **Player with Ball** or **Player No Ball**. On the other hand, the ball passes by the **Enter Attack** and **Ball Throwing** states, until reaching the **Player with Ball** state.

In this case, it is player number 5 who gets the ball. The following sequence of images (Figure 5.70) shows the several states and transitions that lead to the two main attack states after the team reaches the attack.

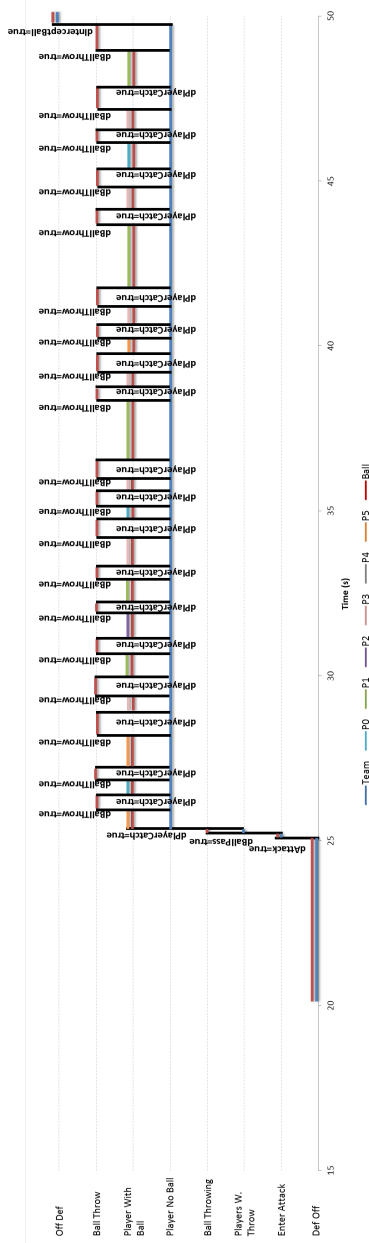


Figure 5.68: Temporal chart of the Attack Petri Net states and transitions during the first attack, from frame 620 (20.6s) to frame 1496 (49.8s).

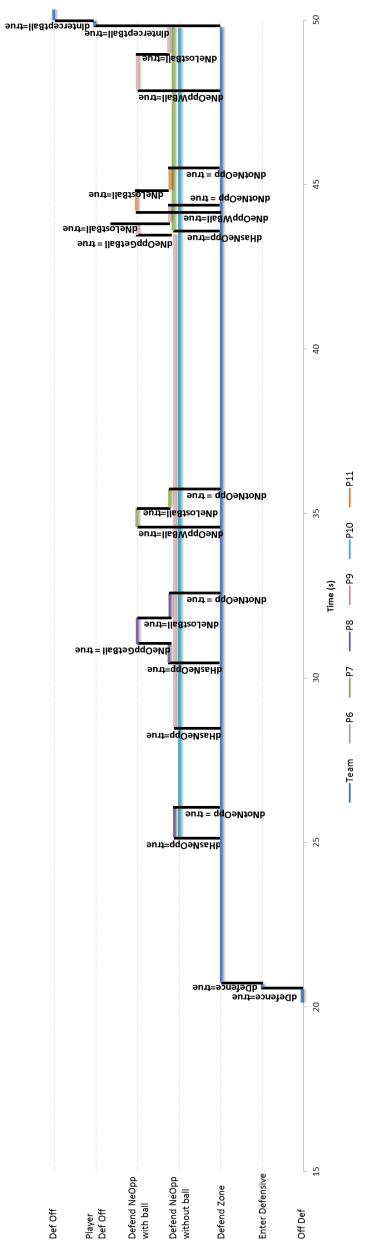


Figure 5.69: Temporal chart of the Defence Petri Net states and transitions during the first defence from frame 620 (20.6s) to frame 1496 (49.8s).

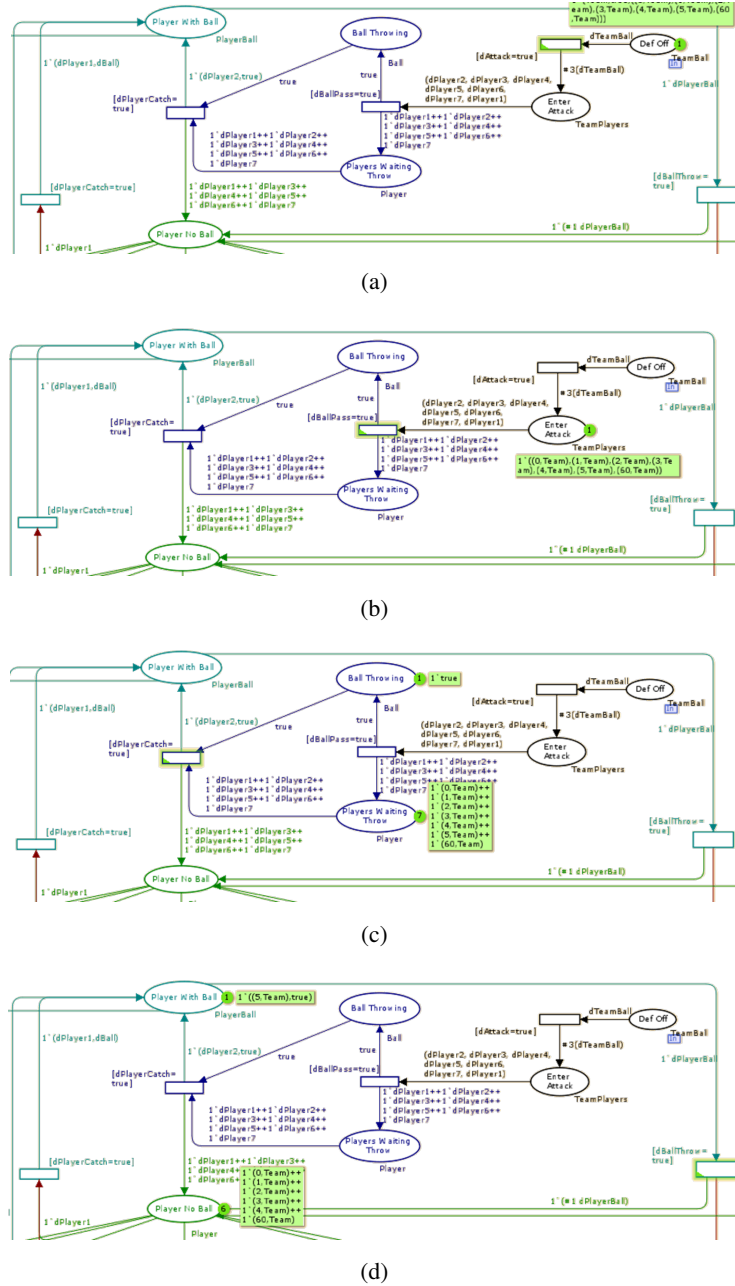


Figure 5.70: **Attack** subnet sequence of transitions when the game starts at frame 620 (20.6s) (partial views of the net): (a) before reaching the **Attack** state, (b) and (c) when reaching an attacking position, (d) decision on whose player has the ball.

On this attack it is also possible to verify that there are several ball passes, 19 in total. Figure 5.71 illustrates the pass that occurs from frame 779 (25.5s), when player 5 passes the ball ($dBallThrow=true$) to frame 789 (26.3s) when player 0 catches it ($dPlayerCatch=true$). During a pass all players are on the **Player No Ball** state, while the ball stays in the **Ball Throw** state until being caught by another player.

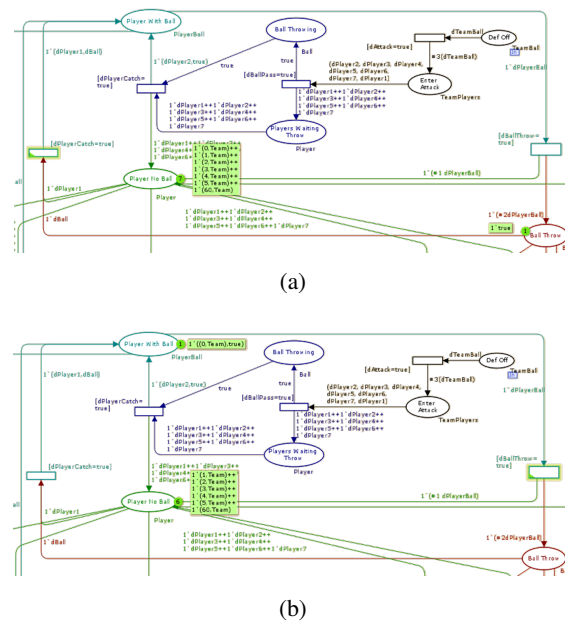


Figure 5.71: Petri Net sequence of a ball pass between players 5 and 0 during frames 779 and 789 (partial views of the net). (a) ball pass start (b) ball pass end.

The global images of the players' positions during this pass can be seen in Figure 5.72.

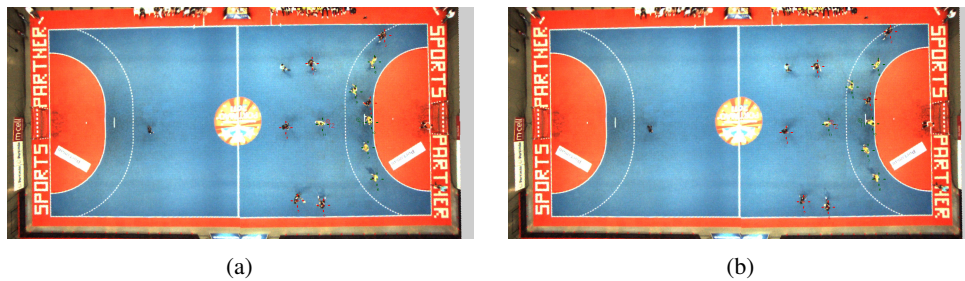


Figure 5.72: Global view of the field during a ball pass. (a) Player 5 passes the ball at frame 779 (b) and player 0 receives the ball at frame 789.

Analysing the players' ball possession it is also possible to verify that the team moves the ball throughout the entire attack zone, since most players, with the exception of player 4, had the ball.

Additionally, looking at the time each player had the ball, it is possible to infer which players dribbled the ball. So, for instance player 1 held the ball for around 2 seconds on two different occasions, from frame 1100 to 1152 (1.73 seconds) and from frame 1251 to 1312 (2.03 seconds), which is out of the average of the ball possession for this attack, around 0.8 seconds.

A closer look at these two time periods shows that in fact player 1 dribbled the ball, as shown in Figure 5.73.

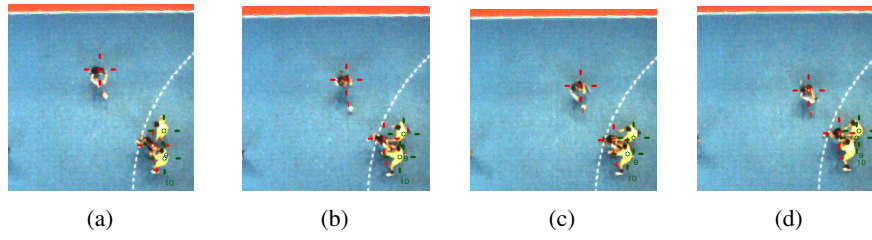


Figure 5.73: Partial view of the field when player 1 is dribbling the ball during frames 1251 to 1312(a) at frame 1262, (b) at frame 1267, (c) at frame 1271 and (d) at frame 1275.

Analysing the defence phase during the same time period it is possible to verify that players 7, 8, 9 and 10 are more actively defending since they are usually defending an opponent, while players 6 and 11 are usually defending a zone of the defence area.

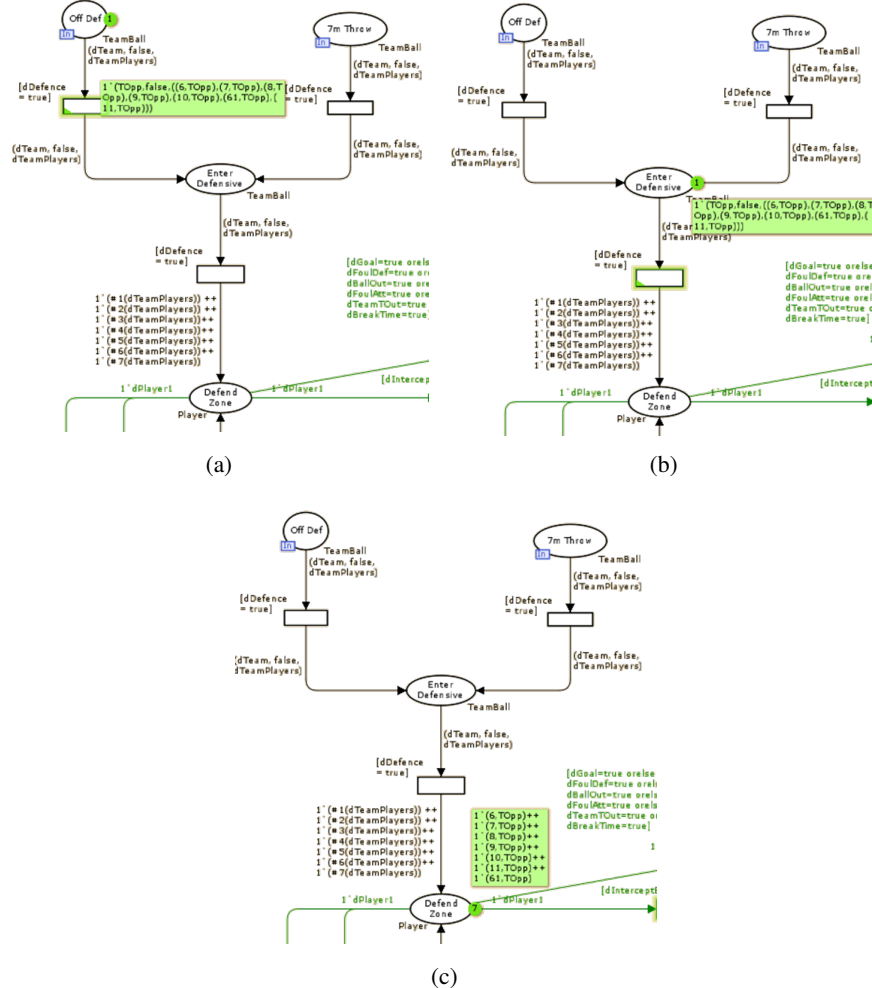


Figure 5.74: **(Defence)** subnet sequence of states and transitions when the game starts at frame 620 (20.6s): (a) before game starts, (b) when team is entering the **Defensive** state and (c) when all players start zone defending.

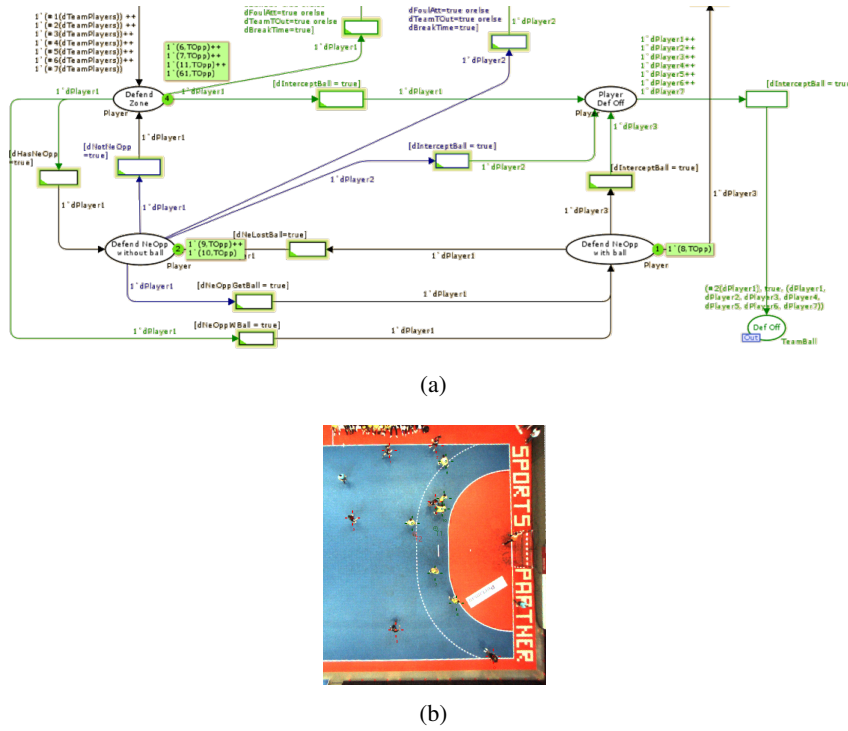


Figure 5.77: (a) Defence Petri net at frame 932 (partial view of the net) and (b) the partial view of the field.

This first attack ends with a player from the attacking team performing a throw that is intercepted by the defending team goalkeeper.

This sequence of events is described in the two following images. Figure 5.78 illustrates how the defence evolves, while Figure 5.79 describes the attack.

5.7 Specific Case Studies

The developed methodologies and algorithms were applied to different study cases as demonstrated in the following sections.

5.7.1 Evaluate players' effort during official competition

The primary goal of this thesis was to perform game analysis, and therefore it was possible to evaluate the players' physical effort, obtain the information regarding their movements on the field and perform high level game analysis which included determining the game phase and classifying the adopted defensive system.

This analysis was performed on professional games that took place during the Portuguese SuperCup handball competition in 2011.

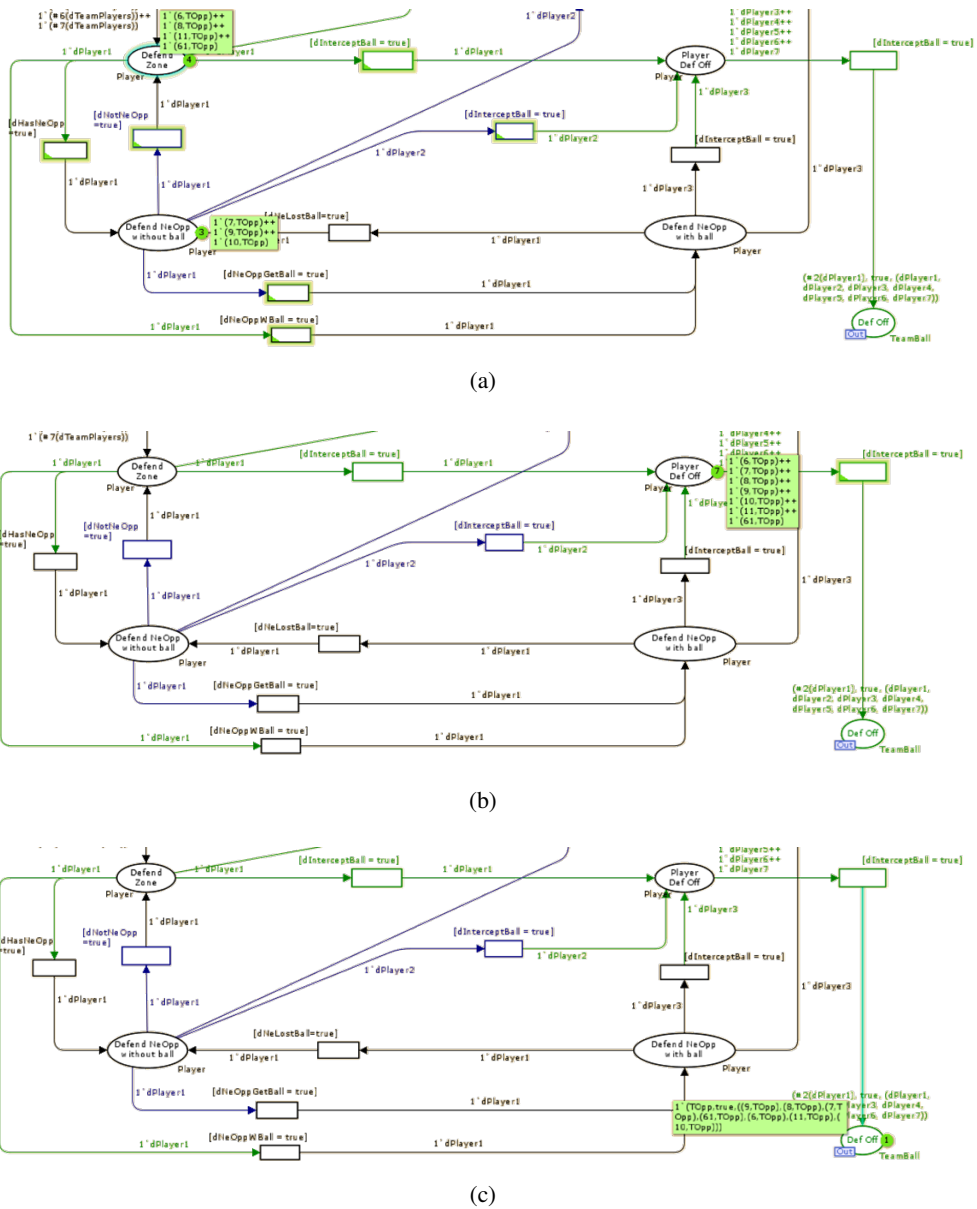


Figure 5.78: **Defence** Petri Net sequence of transitions at the end of the first attack (partial views of the net): (a) the attacking player performs the ball throw, (b) the defending goalkeeper intercepts the ball ($dInterceptBall = true$), (c) players start moving towards the opponent goal (**Def Off** state).

5.7.2 Assessing referees physical effort

The developed platform was also used to perform a more broad study of the physical effort referees spend during official games (Estriga et al. (2013)).

The vision system was based on three cameras placed on the ceiling of the Académico's Sports Hall (Figure 5.1) during a male second division official game.

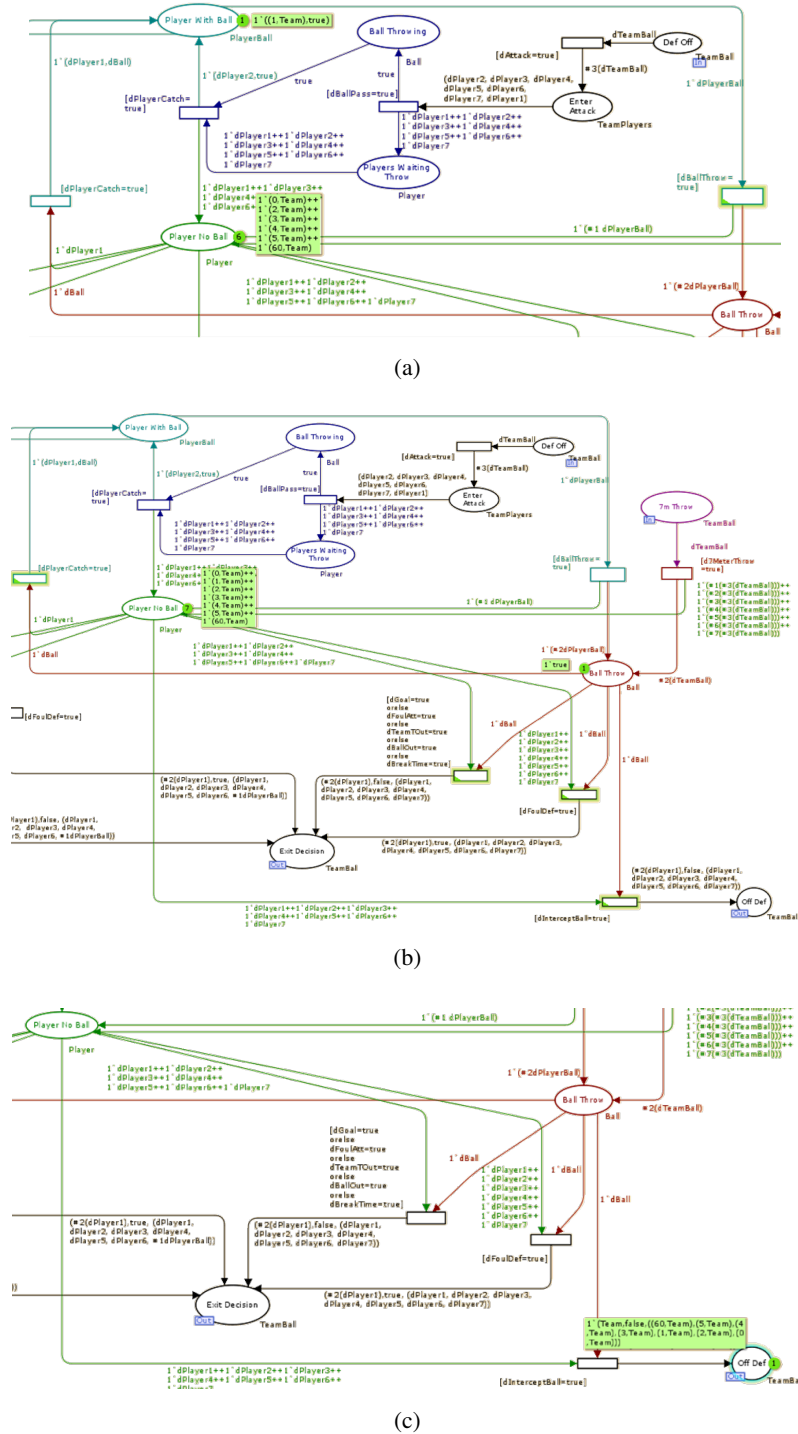


Figure 5.79: **Attack** Petri Net sequence of transitions at the end of the first attack (partial views of the net): (a) before the throw to the goal, (b) while the ball is being thrown, (c) when the ball is intercepted by the defending team goalkeeper.

Besides the vision system, the overall study included an oximeter to evaluate the oxygen consumption of one of the referees during the match, a tri-axial accelerometer and an ECG (electrocardiogram) device.

The vision system was able to supply cinematic information, namely the positional maps of the referees during the game (Figure 5.80) as well as the global distance travelled by the referees ($4.2\text{km} \pm 1\%$).

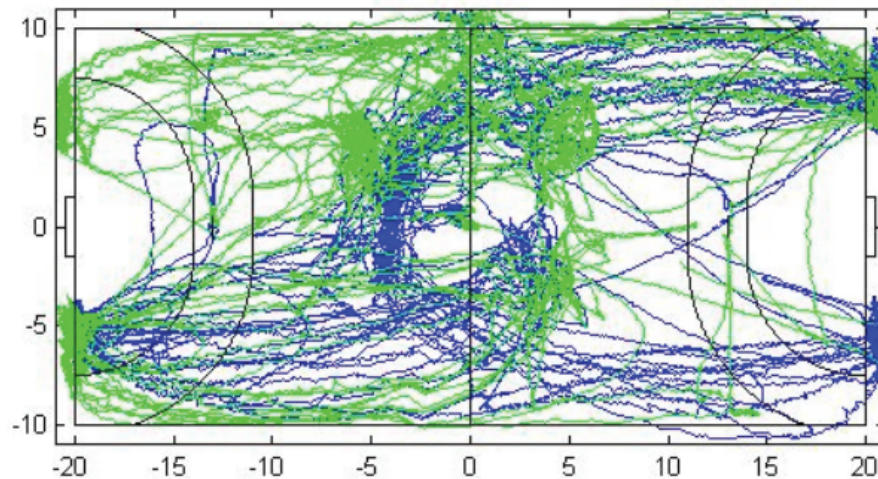


Figure 5.80: Positional map of each referee during one half of the game (in Estriga et al. (2013)).

5.7.3 Assessing Physical Activity Intensity

Another study, (Silva et al. (2015)), used the developed application to evaluate the physical activity intensity during a young players' basketball practice. This study included a comparison between different tracking techniques.

The vision tracking system was compared with a direct observation method (SOPLAY) and with accelerometry data (Actigraph GT3X+) during a 20 minute basketball session.

SOPLAY (McKenzie (2002)) is a system developed by McKenzie for observing play and leisure activity. The method is based on samplings that are performed at periodic times, where separate scans are made for males and females. At each scan the type of activity and intensity (Sedentary, Walking and Very Active) is recorded. This specific study was performed by two observers.

Actigraph (ActiGraph (2015)) is an accelerometer's based system, where acceleration can be measured and recorded on the x-y-z axes. The data is then processed using a proprietary software (Actilife) that allows using different data analysis methodologies.

This study revealed that the vision based system is more correlated with the Actigraph GT3X+ system than the traditional direct observation performed by two independent persons as demonstrated by the Chi-Square tests in Table 5.7.

From this study it was possible to validate that a video tracking system has the potential to provide reasonable estimates of physical activity intensities without being intrusive to the players.

Table 5.7: Results of the Yates Chi-square between methodologies (in [Silva et al. \(2015\)](#)).

Yates Chi-Square		Expected		
		GT3X+	SOPLAY2	SOPLAY1
Observed	CAM	24.18	158.22	110.90
	SOPLAY1	119.55	27.0	
	SOPLAY2	144.44		

5.8 Summary and Conclusions

In this chapter we described the methods and study cases used to validate all the steps of the proposed methodologies. This validation included tests on samples obtained during professional handball games that took place during the Portuguese SuperCup handball competition in 2011.

As it was possible to verify the complete system architecture was fully validated using different acquisition systems that ranged from 1 camera to 3 camera systems placed in two different sports halls.

Results obtained for the player detection and tracking case validated the proposed system and indicate that it is possible to obtain high tracking rates (above 95%) using simple clues, such as colour and physical constraints aided by a robust tracking method (Kalman Filter). The usage of adaptive colour subspaces generated by the Fuzzy inspired methodology allowed to better define the teams' colour properties during the game and increased the overall detection rates, minimizing the user intervention.

The adaptiveness of the proposed player detection methodology also proved effective when subject to a sensitivity analysis by varying the expansion time, illumination conditions, colour learning rate, initial seeds choice and video resolution.

Concerning the usage of the same methodology to track the ball, preliminary tests indicate that the detection and tracking rates are too low due to the high velocity the ball can achieve, and the fact it can be carried through the field by the players.

The information provided by the detection and tracking phases was then used as inputs to algorithms able to determine the game phase and the defensive strategy adopted by the teams. The integration of these measures into the algorithms to extract high level information is straightforward since the tracking is performed in real world coordinates.

Tests on the phase detection proved that the presented Fuzzy Logic classifier is able to correctly identify the phase change events with a percentage of 86.76% and a time misalignment of 1.9s. On the other hand, the defensive strategy is classified with an accuracy above 85%.

Once all the raw data has been processed (by raw data we mean video streams), the final outcome includes not only the aforementioned high level analysis, but also low level information pertaining the players playing time, occupied areas on the field (with an error less than 35cm) and velocity during the game.

Concerning the game itself, we believe the proposed game model is able to address most of the game situations as was proved by the analysis of 3600 frames. Moreover the hierarchical nature of

it, allows having different details of the game which is adequate for different audiences. Another positive aspect of this model choice, is related with the visual nature of a Petri Net that allows an immediate perception of what can be the game next states and provides a user friendly interface for the end user.

Although most of the states are already addressed by the model, we are aware it lacks on providing a way to address substitutions and fouls.

Finally, it is possible to verify how the importance of studying and proposing solutions as was done in this thesis is crucial and necessary to perform other related studies, being either the assessment of the physical effort of referees or the physical activity during practices.

Chapter 6

Conclusions

This chapter summarizes the main conclusions and achievements of this thesis (apart from what was already mentioned on section 1.3) and provides some guidelines for future work based on the current limitations. The implementation of these guidelines will, for sure, enrich the work developed so far.

6.1 Research Findings

The major findings of this thesis are focused on two main areas:

Player detection and tracking Player detection and tracking is achieved using a Fuzzy inspired model of the team's colours. This model divides the colour space into subspaces, one per team, by attributing to each colour a Fuzzy belonging degree. The Fuzziness of the colour subspaces allow colours to belong to more than one team and adapt (change shape) during the game to different lighting conditions.

Game modelling and metrics extraction The hierarchical nature of the proposed handball game model (Hierarchical Coloured Petri Net), allows it to be modular, to easily add new game concepts or even adapt it to other indoor sports.

The players' coordinates on the real world (resulting from the player detection and tracking phase) constitute the basis for extracting more complex game metrics such as the game phase or the defensive team formation. The methodologies used are based on Fuzzy classifiers, for the game phase, and on an error minimization algorithm, for the team formation.

With the proposed methodologies it was possible to answer the questions raised at section 1.2 and validate the initial hypothesis. Moreover, the complete solution was tested in real game situations with results that satisfied the sport's experts.

Q1 : What architecture should be used to achieve this goal?

Due to the vast area to be covered and the need to minimize occlusion issues a multi-camera system was proposed. Videos from each camera are recorded and, afterwards, each video is

analysed offline in order to detect the players. Once the players are detected, their centre of mass is converted into real world coordinates using the cameras' homographies and barrel distortion parameters.

The usage of real world coordinates makes the cameras information aggregation straightforward by means of a vector of Kalman filters.

Additionally, the information obtained from the players can be visualized using a visualizing module that provides a global and undistorted view of the field with the players highlighted and schematic views with speed and positional information.

Q2 : What information should be extracted from the vision system?

The information necessary to be extracted from the vision system are the players' positions in real world coordinates as well as their velocities.

Q3 : How to extract that information?

This information is extracted using robust detection methodologies that are able to adapt to changing light conditions. Players are detected using a Fuzzy inspired model of the players' vests colours, that dynamically and automatically adapts to the light conditions that affect the colour properties.

At the first stage, foreground pixels are detected using background subtraction, afterwards the obtained pixels are categorized with a specific belonging degree into one of the teams by comparing their colour with the ones defined on the colour model. Finally adjacent pixels are grouped into blobs that are further refined using area and pixels density constraints.

The detection is enhanced using a tracking algorithm based on a vector of Kalman Filters.

Q4 : How to model game concepts?

A game model was then built on top of the detection and tracking module. This model combines a Hierarchical Coloured Petri Net to define the game main states, with specific models that are able to classify the game phase (defensive, attack and transitional) and the defensive tactic system.

Q5 : How to extract metrics from the models?

The output of the tracking module can then be used to extract metrics that range from simple statistics such as speed or travelled distance to more complex information, which is obtained by feeding the players' information into the defined models to obtain the required metrics.

Two complex game models were defined, the first one represents the game phase using a Fuzzy classifier with the teams' longitudinal velocity and position as inputs and the current game phase as output. The second model categorizes the defensive team system through an error minimization based classifier.

6.2 Implementation Results

The proposed methodology was validated with real game situations during the Handball Portuguese SuperCup, year 2011. Tests on these footages proved the concept for the acquisition system and validated the proposed methodology. The results showed an accuracy for player tracking above 95%, obtained by the combination of a detection method, a Fuzzy inspired methodology, and a set of Kalman filters for tracking.

The usage of the Fuzzy Logic classifier for phase detection proved effective with an accuracy above 86% and a time miss alignment of 1.9s. On the other hand, using a minimization algorithm that models the defensive strategy as semi-circles around the goal it was able to correctly classify the defensive system with an accuracy above 85%.

6.3 Publications

The work developed during this thesis has led to several publications in conferences, as well as in journals and book chapters.

[Santiago et al. \(2010a\)](#) provides an initial research on the state of art concerning the detection and tracking techniques applied to the sports domain, while [Santiago et al. \(2011b\)](#) describes the first sketch of the proposed methodology for player detection using region based and clustering methods.

Still on an early phase, some studies were performed for the detection and tracking of a ball using colour features under a controlled and laboratorial environment [Santiago et al. \(2010b\)](#).

The initial proposed player detection methods were empowered with the Fuzzy methodology and presented on [Santiago et al. \(2012c\)](#).

Meanwhile, an exploratory work was developed to adapt the official RoboCup Simulation League Simulator ([Chen et al. \(2002\)](#)) to the specific case of handball, [Santiago et al. \(2011a\)](#). Although not referred, this platform could be of great use if a simulator was to be integrated into the proposed final solution.

Finally, the proposed solution was fully described on [Santiago et al. \(2012a\)](#) chapter, and a detailed sensitivity analysis as well as a more in depth testing was presented on [Santiago et al. \(2013\)](#).

Additionally, the developed work served as basis to assess referees effort during official competitions [Estriga et al. \(2013\)](#) and assessing physical activity intensity when compared with observational and GPS based methods [Silva et al. \(2013\)](#) and [Silva et al. \(2015\)](#).

6.4 Guidelines for Future Work

Although the obtained results validated the proposed methodology and answered the initial questions and hypothesis, there are still some open points that are worth investigating:

Player Detection and Tracking As was already highlighted, the proposed methodology has not proven to be effective on tracking the ball, therefore the overall solution would benefit if other methodologies for tracking the ball were explored. Given the ball movement characteristics, we believe an Extended Kalman filter would produce better results, which could be complemented with specific heuristics to include the fact that the ball can be carried and, consequently, severely occluded by a player.

Another interesting path could be to use information from the teams' disposition and movements on the field and combine them with data mining techniques in order to evaluate and predict the ball position.

High Level Game Analysis The proposed algorithms for performing high level game analysis were narrowed to detecting the game phase and classifying the defensive tactic, however, to fully benefit of the proposed game model, one major opportunity of improvement would be to study and develop algorithms that could identify each transition and state so that the game analysis could be fully automated and directly linked to the proposed Hierarchical Coloured Petri Net model.

Game Model Concerning the game model we believe it could be valuable to include other features of the handball game, such as the players' substitutions and the attribution of yellow and red cards. Moreover, it would also be interesting to explore its modularity and adapt it to other indoor sports, such as basketball and hockey.

Software Application Currently the three software modules are spread into three different applications that communicate via specific video or text files, and so an integrated solution would be easier to the end users. It would also be important to integrate the HCPN game model into the final application, instead of using the CPN Tools software.

Include an online module, that could allow the user to add specific comments and notes while the game is being recorded, would also be interesting.

6.5 Final Remarks

The proposed methodologies proved to be effective on answering the research questions raised at the beginning of this thesis. Moreover, tests on real game situations during official games and training sessions achieved tracking rates of 95% and phase change events detection above 86%.

The importance of this work is clear by the measure of satisfaction of the informal tests with the sports experts that appraised the automatic annotation that in turn allowed for automatic high level game analysis.

Appendix A

Game Model - Appendix

The second attack and defence periods are described on the following figures (Figures [A.1](#) and [A.2](#)).



Figure A.1: Temporal chart of the **Attack** Petri Net states and transitions during the second attack: from frame 1759 (58.6s) to frame 2580 (86s).



The third attack and defence are described on Figures A.3 and A.4.

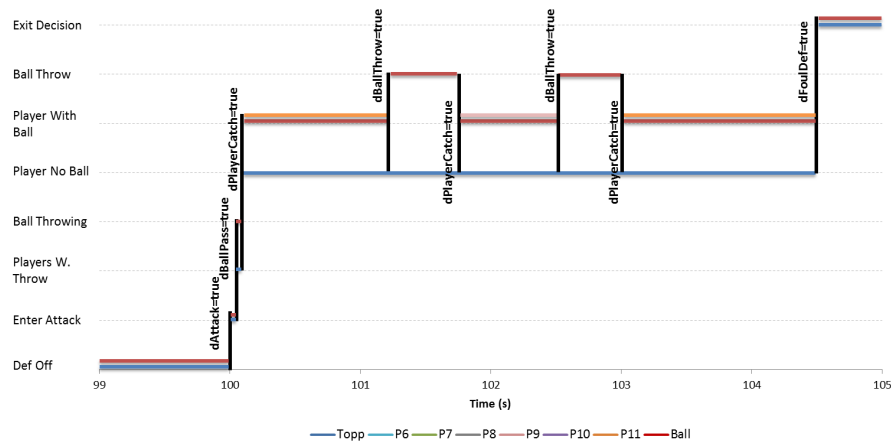


Figure A.3: Temporal chart of the **Attack** Petri Net states and transitions during the third attack: from frame 3001 (100s) to frame 3136 (104.5s).

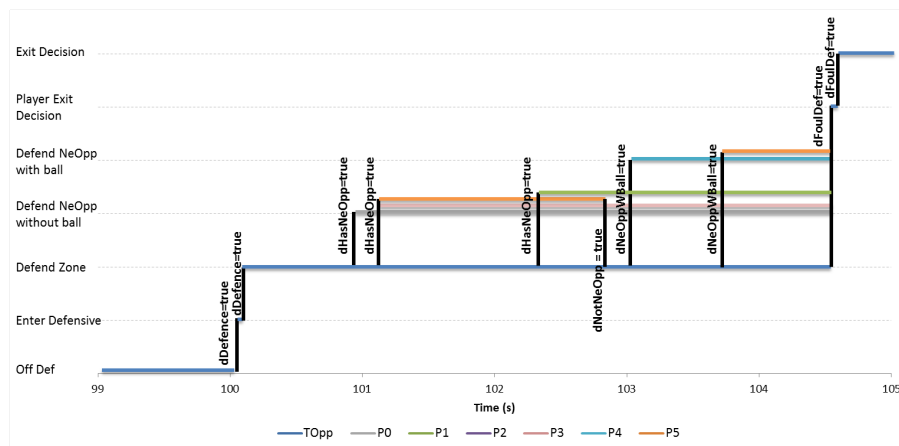


Figure A.4: Temporal chart of the **Defence** Petri Net states and transitions during the third attack: from frame 3001 (100s) to frame 3136 (104.5s).

The final attack and defence sequences (the fourth) of the analysed time period are defined on Figures A.5 and A.6.

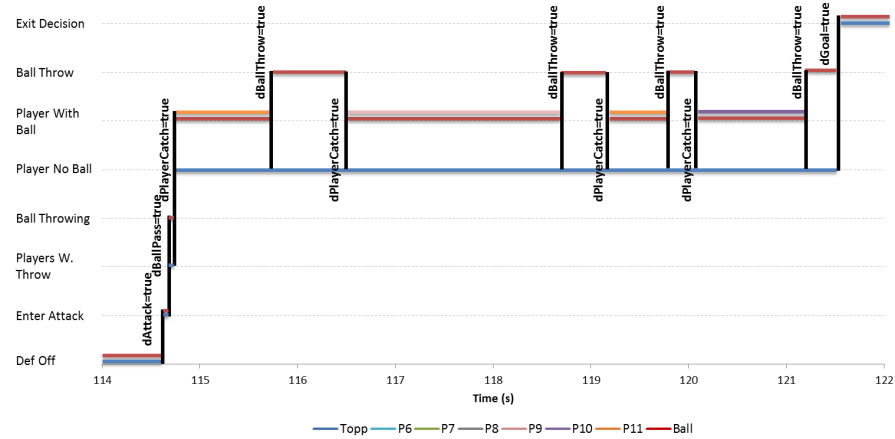


Figure A.5: Temporal chart of the **Attack** Petri Net states and transitions during the fourth attack: from frame 3436 (114.5s) to frame 3639 (121.3s).

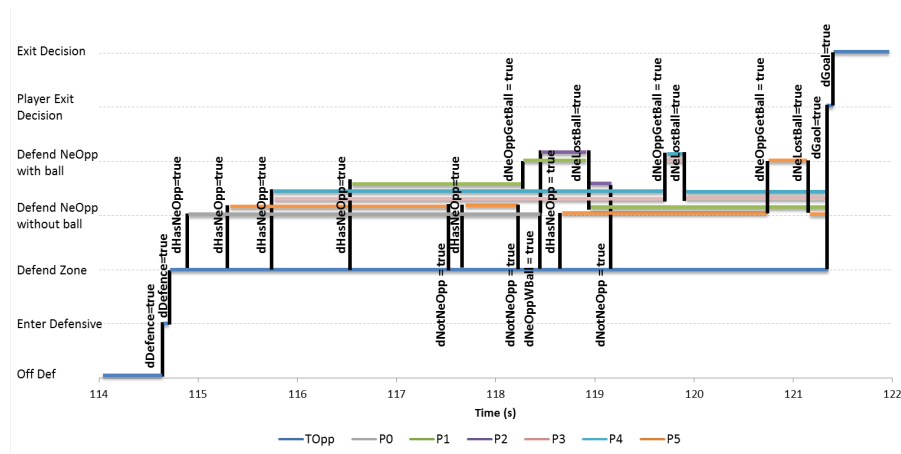


Figure A.6: Temporal chart of the **Defence** Petri Net states and transitions during the fourth attack: from frame 3436 (114.5s) to frame 3639 (121.3s).

References

- Pedro Abreu, José Moura, Daniel C. Silva, Luís P. Reis, and Júlio Garganta. Performance analysis in soccer: a cartesian coordinates based approach using robocup data. *Soft Computing*, 16(1): 47–61, 2012.
- Pedro Abreu, José Moura, Daniel C. Silva, Luís P. Reis, and Júlio Garganta. Football Scientia - An Automated Tool for Professional Soccer Coaches. In *2010 IEEE Conference on Cybernetics and Intelligent Systems, Cis 2010*, pages 126–131, 2010.
- Pedro H. Abreu, Daniel C. Silva, J. Portela, J. Mendes-Moreira, and Luís P. Reis. Using model-based collaborative filtering techniques to recommend the expected best strategy to defeat a simulated soccer opponent. *IDA – Intelligent Data Analysis*, 18(5):973–991, 2014.
- David H. Ackley, Geoffrey E. Hinton, and Terrence J. Sejnowski. A learning algorithm for boltzmann machines. *Cognitive Science*, pages 147–169, 1985.
- ActiGraph. GT3X+ | Actigraph, 2015. URL <http://www.actigraphcorp.com/support/devices/gt3xplus/>.
- Alexandre Alahi, Yannick Boursier, Laurent Jacques, and Pierre Vandergheynst. Sport Players Detection and Tracking With a Mixed Network of Planar and Omnidirectional Cameras. In *Third ACM/IEEE International Conference on Distributed Smart Cameras*, pages 1–8. IEEE, 2009. doi: 10.1109/ICDSC.2009.5289406.
- Fernando Almeida, Pedro H. Abreu, Nuno Lau, and Luís P. Reis. An automatic approach to extract goal plans from soccer simulated matches. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, 17(5):835–848, 2012.
- Marcelo Alonso and Edward Finn. *Physics*. Addison Wesley Longman, 1992. ISBN 0201565188.
- Apidis. apidis-overview, 2008. URL <http://www.apidis.org/index.htm>.
- Apidis. Apidis basket ball dataset1, 2009. URL <http://www.apidis.org/Dataset/>.
- M. Sanjev Arulampalam, Simon Maskell, Neil Gordon, and Tim Clapp. A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking. *IEEE Transactions on Signal Processing*, 50(2):174–202, 2002.
- Liang Bai, Songyang Lao, Alan F. Smeaton, Noel E. O’Connor, David Sadlier, and David Sinclair. Semantic Analysis of Field Sports Video using a Petri-Net of Audio-Visual Concepts. *The Computer Journal*, 52(7):808–823, 2009.
- Michael Barnathan. Mammographic Segmentation Using WaveCluster. *Algorithms*, 5(3):318–329, 2012.

- Sian Barris and Chris Button. A review of vision-based motion analysis in sport. *Sports Medicine*, 38(12):1025–1043, 2008.
- John L. Barron and Neil A. Thacker. Tutorial : Computing 2D and 3D Optical Flow. *Biomedical Engineering*, 2004-012:1–12, 2005. URL <http://www.tina-vision.net/docs/memos/2004-012.pdf>.
- Ricardo Barros, Rafael Menezes, Tiago Russomanno, Milton Misuta, Bruno Brandao, Pascual Figueroa, Neucimar Leite, and Siome Goldenstein. Measuring handball players trajectories using an automatically trained boosting algorithm. *Computer Methods in Biomechanics and Biomedical Engineering*, 14(1):53–63, 2011.
- Michael Beetz, Suat Gedikli, Jan Bandouch, Bernhard Kirchlechner, Nicolai von Hoyningen-Huene, and Alexander Perzylo. Visually Tracking Football Games Based on TV Broadcasts. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2066–2071, 2007.
- Michael Beetz, Nicolai von Hoyningen-Huene, Bernhard Kirchlechner, Suat Gedikli, Francisco Siles, Murat Durus, and Martin Lames. ASPOGAMO: Automated Sports Game Analysis Models. *International Journal of Computer Science in Sport*, 8(1):4–12, 2009.
- Serge Beucher and Christian Lantuejoul. Use of Watersheds in Contour Detection. In *International Workshop on Image Processing, Real-Time Edge and Motion Detection/Estimation*, pages 17–21, 1979.
- James C. Bezdek. *Pattern recognition with fuzzy objective function algorithms*. Advanced applications in pattern recognition. Plenum Press, New York, 1981. ISBN 0-306-40671-3. URL <http://opac.inria.fr/record=b1090684>.
- Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer Science+Business Media, LLC, New York, USA, 2006.
- Andrea Bobbio. *System Modelling with Petri Nets*, volume 6 of *ISPRA Courses*. Springer Netherlands, 1990. ISBN 978-94-010-6777-5. doi: 10.1007/978-94-009-0649-5_6.
- Thierry Bouwmans, Fida El Baf, and Bertrand Vachon. Background Modeling using Mixture of Gaussians for Foreground Detection - A Survey. *Recent Patents on Computer Science*, 1(3): 219–237, 2008.
- Gary R. Bradski. Computer Vision Face Tracking For Use in a Perceptual User Interface. *Intel Technology Journal*, Q2, 1998.
- Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, and François Yergeau. Extensible Markup Language (XML) 1.0 (Fifth Edition), 2008. URL <http://www.w3.org/TR/2008/REC-xml-20081126/>.
- Leo Breiman. Bagging predictors. *Journal of Machine Learning*, 24(2):123–140, August 1996. ISSN 0885-6125. doi: 10.1023/A:1018054314350.
- Dan Brickley and Ramanathan V. Guha. RDF Vocabulary Description Language 1.0: RDF Schema, 2004. URL <http://www.w3.org/TR/2004/REC-rdf-schema-20040210/>.

- Paul Buitelaar, Thierry Declerck, Jan Nemrava, and David Sadlier. Cross-Media Semantic Indexing in the Soccer Domain. In *International Workshop on Content-Based Multimedia Indexing, 2008*, pages 296–301. IEEE, 2008.
- Ante Burger, Nenad Rogulj, Nikola Foretić, and Marijana Čavala. Analysis of Rebounded Balls in a Team Handball Match. *SportLogia*, 9(1):53–58, 2013. ISSN 1986-6119.
- Christopher J.C. Burges. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.
- Cairos Technologies AG. Sitemap | Cairos Technologies AG. URL http://www.cairos.com/WP-cairos/?page_id=22.
- John Canny. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6), 1986.
- Christopher Carling, Jonathan Bloomfield, Lee Nelsen, and Thomas Reilly. The Role of Motion Analysis in Elite Soccer. *Sports Medicine*, 38(10):839–862, 2008.
- Julen Castellano, David Casamichana, Julio Calleja-González, Jaime San Román, and Sergej M. Ostojic. Reliability and Accuracy of 10Hz GPS Devices for Short-Distance Exercise. *Journal of Sports Science and Medicine*, 10:233–234, 2011.
- Catapult Innovations. MinimaxV4 GPS performance - sprints (white paper). Technical report, 2009.
- Catapult Sports. Catapult Sports Ball Tracking Tech to Deliver Game Changing Data Boost for AFL Teams | Catapult USA. URL <http://www.catapultsports.com/us/media/catapult-sports-ball-tracking-tech-to-deliver-game-changing-data-boost-for-afl-teams>.
- Thanapong Chaichana, Sarat Yoowattana, Zhonghua Sun, Supan Tangjikusolmun, Supot Sookpotharom, and Manas Sangworasil. Edge Detection of the Optic Disc in Retinal Images Based on Identification of a Round Shape. In *International Symposium on Communications and Information Technologies*, pages 670–674, 2008.
- Barbara Chapman, Gabriele Jost, and Ruud Van Der Pas. *Using OpenMP: Portable Shared Memory Parallel Programming*. The MIT Press, 2007.
- Vinay K. Chaudhri, Adam Farquhar, Richard Fikes, Peter D. Karp, and James P. Rice. Open Knowledge Base Connectivity 2.0.3. Technical report, Stanford University, 1998.
- Fan Chen, Damien Delannay, and Christophe De Vleeschouwer. An Autonomous Framework to Produce and Distribute Personalized Team-Sport Video Summaries: a Basketball Case Study. *IEEE Transactions on Multimedia*, (99):1381–1394, 2011.
- Hua-Tsung Chen, Chien-Li Chou, Tsung-Sheng Fu, Suh-Yin Lee, and Bao-Shuh P. Lin. Recognizing Tactic Patterns in Broadcast Basketball Video Using Player Trajectory. *Journal of Visual Communication and Image Representation*, 23(6):932–947, August 2012. ISSN 10473203. doi: 10.1016/j.jvcir.2012.06.003.
- Mao Chen, Ehsan Foroughi, Fredrik Heintz, ZhanXiang Huang, Spiros Kapetanakis, Kostas Kostiadis, Johan Kummeneje, Itsuki Noda, Oliver Obst, Pat Riley, Timo Steffens, Yi Wang, and Xiang Yin. *Robocup Soccer Server: User Manual*. RoboCup Federation, 2002.

- Hengda D. Cheng, X.H. Jiang, Y. Sun, and Jingli Wang. Color Image Segmentation: Advances and Prospects. *Pattern Recognition*, 34(12):2259–2281, 2001.
- Su-Lin Chin, Chun-Hong Huang, Chia-Yong. Tang, and Jason Hung. An Application Based on Spatial-Relationship to Basketball Defensive Strategies. In *Embedded and Ubiquitous Computing – EUC 2005 Workshops*, pages 180–188. Springer, 2005.
- Reita Clanton and Mary Phil Dwight. *Team Handball: Steps to Success*. Human Kinetics, 1997. ISBN 9780873224116.
- Miguel T. Coimbra. Computer Vision 2011/2012 Lecture 13 - Pattern Recognition, 2013. URL http://www.dcc.fc.up.pt/~mcoimbra/lectures/VC_1112/VC_1112_T13_PatternRecognition.pdf.
- Dorin Comaniciu and Peter Meer. Robust Analysis of Feature Spaces: Color Image Segmentation. *Computer Vision and Pattern Recognition*, pages 750 – 755, 1997.
- Electronic Communications Committee. The European Table of Frequency Allocations and Applications in the Frequency Range 8.3kHz to 3000 GHz (ECA Table). Technical report, Electronic Communications Committee, 2014. URL <http://www.ero-docdb.dk/Docs/doc98/official/pdf/ercprep025.pdf>.
- Ciarán Ó. Conaire, Philip Kelly, Damien Connaghan, and Noel E. O’Connor. Tennissense: A Platform for Extracting Semantic Information from Multi-Camera Tennis Data. In *16th International Conference on Digital Signal Processing*, pages 1–6. IEEE, 2009.
- Oscar Corcho and Asuncion Gomez-Perez. Evaluating Knowledge Representation and Reasoning Capabilities of Ontology Specification Languages. In *Proceedings of the ECAI 2000 Workshop on Applications of Ontologies and Problem-Solving Methods*, 2000.
- João Cravo, Fernando Almeida, Pedro H. Abreu, Luís. Reis, Nuno Lau, and Luís Mota. Strategy planner: Graphical definition of soccer set-plays. *Data Knowledge Engineering*, 94, Part A: 110 – 131, 2014. ISSN 0169-023X. doi: <http://dx.doi.org/10.1016/j.datak.2014.10.001>.
- Janusz Czerwinski and Frantisek Taborsky. *Basic Handball: Methods/Tactics/Technique*. AVIS-Werbung, Vienna, Austria, 2005.
- Manoranjan Dash, Huan Liu, and Xiaowei Xu. ‘1+1>2’: Merging distance and density based clustering. In *Proceedings of the Seventh International Conference on Database Systems for Advanced Applications*, pages 32–39, 2001.
- William S. Davis and David C. Yen. *The Information System Consultant’s Handbook: Systems Analysis and Design*. CRC Press, 1998. ISBN 978-1-4200-4910-7.
- Damien Delannay, Nicolas Danhier, and Christophe De Vleeschouwer. Detection and Recognition of Sports (Wo)men from Multiple Views. In *Proceedings of the International Conference on Distributed Smart Cameras (ICDSC)*, pages 15–21, Como, Italy, August 2009. doi: 10.1109/ICDSC.2009.5289407.
- Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38, 1977.

- Yining Deng and B. S. Manjunath. Unsupervised Segmentation of Color-Texture Regions in Images and Video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(8):800–810, 2001.
- John Domingue, Enrico Motta, and Oscar Corcho Garcia. Knowledge Modelling in WebOnto and OCML: A User Guide Knowledge. Technical report, The Knowledge Media Institute, The Open University, Milton Keynes, UK, 1999.
- Tiziana D’Orazio, Cataldo Guaragnella, Marco Leo, and Arcangelo Distanto. A New Algorithm for Ball Recognition Using Circle Hough Transform and Neural Classifier. *Pattern Recognition*, 37(3):393–408, 2004.
- Tiziana D’Orazio, Marco Leo, Paolo Spagnolo, Massimiliano Nitti, Nicolo Mosca, and Arcangelo Distanto. A Visual System for Real Time Detection of Goal Events During Soccer Matches. *Computer Vision and Image Understanding*, 113(5):622–632, 2009.
- Ling-Yu Duan, Min Xu, Qi Tian, and Chang-Sheng Xu. A Unified Framework for Semantic Shot Classification in Sports Video. *IEEE Transactions on Multimedia*, 7(6):1066–83, 2005.
- Jordi Duch, Joshua S. Waitzman, and Luís A. Nunes Amaral. Quantifying the Performance of Individual Players in a Team Activity. *PloS one*, 5(6), 2010.
- Grant Duthie, David Pyne, and Sue Hooper. The Reliability of Video Based Time Motion Analysis. *Journal of Human Movement Studies*, 44:259–271, 2003.
- Espor. espor - about | facebook, 2015. URL https://www.facebook.com/espor.tv/info?tab=page_info.
- Martin Ester, Hans-Peter Kriegel, Joerg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In Evangelos Simoudis, Jiawei Han, and Usama M. Fayyad, editors, *Second International Conference on Knowledge Discovery and Data Mining*, pages 226–231. AAAI Press, 1996.
- Luísa Estriga, Catarina Santiago, and João Carvalho. Avaliação do esforço físico de Árbitros de andebol em competição. *Revista Mineira de Educação Física*, (9):854–860, 2013.
- European Space Agency. Indoor navigation solution, 2013. URL <http://www.esa-tec.eu/space-technologies/from-space/indoor-navigation-solution/>.
- Brígida M. Faria, Luís P. Reis, Nuno Lau, and Gladys Castillo. Machine Learning Algorithms Applied to the Classification of Robotic Soccer Formations and Opponent Teams. In *2010 IEEE Conference on Cybernetics and Intelligent Systems, Cis 2010*, pages 344–349, 2010.
- Adam Farquhar, Richard Fikes, and James Rice. The Ontolingua Server: A Tool for Collaborative Ontology Construction. *International Journal of Human-Computer Studies*, 46(6):707–727, 1997.
- Damien Farrow, David Pyne, and Tim Gabbett. Skill and Physiological Demands of Open and Closed Training Drills in Australian Football. *International Journal of Sports Science and Coaching*, 3(4):489–499, 2008.
- Dieter Fensel, Ian Horrocks, Frank van Harmelen, Stefan Decker, Michael Erdmann, and Michel Klein. *OIL in a Nutshell*, volume 1937 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2000. ISBN 978-3-540-41119-2. doi: 10.1007/3-540-39967-4_1. URL http://dx.doi.org/10.1007/3-540-39967-4_1.

- Pascual J. Figueroa, Neucimar J. Leite, and Ricardo M. L. Barros. Tracking Soccer Players Aiming Their Kinematical Motion Analysis. *Computer Vision and Image Understanding*, 101(2):122–135, 2006.
- Foldoc. Knowledge Representation, 1994. URL <http://foldoc.org/knowledge+representation>.
- David Forsyth and Jean Ponce. *Computer Vision: A Modern Approach*. Prentice Hall, London, London, 2003.
- Ian M. Franks and Mike Hughes. *Notational Analysis of Sport: Systems for Better Coaching and Performance in Sport*. Psychology Press, 2004.
- Ian M. Franks, G. E. Wilson, and David Goodman. Analysing a Team Sport With the Aid of Computers. *Canadian Journal of Sport Sciences*, 12(2):120–25, 1987.
- Xavier Pascual Fuertes. *La Tactica Individual Dentro de los Sistemas de Juego*. Girona, 2010. ISBN 978-84-8458-306-6.
- Tim J. Gabbett. GPS Analysis of Elite Women’s Field Hockey Training and Competition. *Journal of Strength and Conditioning Research*, 24(5):1321–1324, 2010.
- Paul Gastin and Kylie Williams. Accuracy of 1 Hz versus 5 Hz GPS Devices to Measure Movement Patterns in Team Sport Activities. *Journal of Science and Medicine in Sport*, 13(1):e32, 2010.
- Michael R. Genesereth and Richard E. Fikes. Knowledge Interchange Format Version 3.0. Technical report, Stanford University, 1994.
- Fernando P. Gomes. *Análise do Jogo no Andebol: Caracterização do Processo Defensivo, em Situação de 6x6, dos Três Primeiros Classificados no Campeonato da Europa 2006, Seniores Masculinos*. PhD thesis, Universidade Técnica de Lisboa Faculdade de Motricidade Humana, 2008.
- Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Prentice Hall, London, London, UK, 2nd edition edition, 2002.
- Neil J. Gordon, David J. Salmond, and Adrian F. M. Smith. A novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proceedings on Radar and Signal Processing*, 140(2):107–113, 1993.
- GPSports. Hardware - GPSports | Global Positioning Systems, 2015. URL <http://gpsports.com/spi-hpu/>.
- Stephan Grimm, Pascal Hitzler, and Andreas Abecker. Knowledge Representation and Ontologies, 2007.
- William E. L. Grimson, Chris Stauffer, Raquel Romano, and Lily Lee. Using Adaptive Tracking to Classify and Monitor Activities in a Site. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 22–29. IEEE, 1998.
- Khronos Group. OpenGL - The Industry Standard for High Performance Graphics, 2015. URL <https://www.opengl.org/>.
- Igor Gruic, Dinko Vuleta, and Dragan Milanovic. Performance Indicators of Teams at the 2003 Mens World Handball Championship in Portugal. *Kinesiology*, 38(2):164–175, 2006.

- Veyis Gunes, Michel Ménard, and Simon Petitrenaud. *Multiple Classifier Systems: Tools and Methods*. World Scientific, 2010.
- John A. Hartigan. *Clustering algorithms*. Wiley series in probability and mathematical statistics: Applied probability and statistics. Wiley, 1975. ISBN 9780471356455. URL <https://books.google.pt/books?id=NznvAAAAMAAJ>.
- Jean-Bernard Hayet, T. Mathes, Jacek Czyz, Justus Piater, Jacques Verly, and Benoît Macq. A Modular Multi-Camera Framework for Team Sports Tracking. In *Proceedings of the IEEE Conference on Advanced Video and Signal Based Surveillance*, pages 493–498, 2005.
- Janne Heikkilä and Olli Silvén. A Real-Time System for Monitoring of Cyclists and Pedestrians. In *Second IEEE Workshop on Visual Surveillance*, pages 74–81. IEEE, 1999.
- Anita Hökelmann, Klaus Richter, Sebastian Scholz, and Matthias Kempe. New Perspectives for Performance Analysis in Sport. In *Proceedings of the World Congress of Performance Analysis of Sport VIII*, pages 152–159, Magdeburg, Germany, 2008.
- Berthold K. P. Horn and Brian G. Schunk. Determining Optical Flow. *Artificial Intelligence*, 17(1-3):185–203, 1981.
- Min-Chun Hu, Ming-Hsiu Chang, Ja-Ling Wu, and Lin Chi. Robust Camera Calibration and Player Tracking in Broadcast Basketball Video. *IEEE Transactions on Multimedia*, 13(2):266–279, 2011. ISSN 1520-9210.
- Hua Huang, Chao Chen, Yulan Jia, and Shuming Tang. Automatic detection and recognition of circular road sign. In *International Conference on Mechatronic and Embedded Systems and Applications*, 2008, pages 626–630, Beijing, China, 2008.
- Peter Huber, Kurt Jensen, and Robert M. Shapiro. Hierarchies in Coloured Petri Nets. *Lecture Notes in Computer Science: Advances in Petri Nets 1990*, 483:313–341, 1991.
- Mike Hughes and Ian M. Franks. *The Essentials of Performance Analysis: an Introduction*. Routledge, 2008.
- IMPIRE AG. VIS.TRACK live: IMPIRE AG, 2015. URL <http://www.bundesliga-datenbank.de/en/91/>.
- InMotio. Technologylinmotio, 2015. URL <http://www.inmotio.eu/en-GB/5/technology.html>.
- Fraunhofer ISS. RedFIR, 2015a. URL <http://www.iis.fraunhofer.de/en/ff/lok/proj/redfir.html>.
- Fraunhofer ISS. RedFir, 2015b. URL <http://www.iis.fraunhofer.de/en/ff/lok/proj/redfir.html#tabpanel-2>.
- Sachiko Iwase and Hideo Saito. Tracking Soccer Players Based on Homography Among Multiple Views. In *Proceedings of SPIE, Visual Communications and Image Processing*, volume 5150, pages 283–292, Lugano, Switzerland, 2003. Citeseer.
- Kurt Jensen. Coloured petri nets and the invariant-method. *Theoretical Computer Science*, 14(3): 317 – 336, 1981. ISSN 0304-3975.
- Kurt Jensen. Coloured Petri Nets: A High Level Language for System Design and Analysis. 483: 342–416, 1991. doi: 10.1007/3-540-53863-1_31.

- Kurt Jensen. *Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use, Vol 1*, volume 1. Springer-Verlag, 1992.
- Kurt Jensen and Lars M. Kristensen. *Coloured Petri Nets: Modelling and Validation of Concurrent Systems*. Springer, 2009.
- Simon J. Julier and Jeffrey K. Uhlmann. A new extension of the kalman filter to nonlinear systems. In *AeroSense'97*, pages 182–193, 1997.
- Rudolph E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME—Journal of Basic Engineering*, 82(Series D):35–45, 1960.
- Chan-Hyun Kang, Jung-Rae Hwang, and Ki-Joune Li. Trajectory Analysis for Soccer players. *Sixth IEEE International Conference on Data Mining Workshops*, pages 377–381, 2006.
- Peter D. Karp, Vinay K. Chaudhri, and Jerome F. Thomere. XOL: An XML-Based Ontology Exchange Language, 1999.
- Soudeh Kasiri-Bidhendi and Reza Safabakhsh. Effective Tracking of the Players and Ball in Indoor Soccer Games in the Presence of Occlusion. In *Proceedings of the 14th International CSI Computer Conference*, pages 524–529. IEEE, 2009.
- Leonard Kaufman and Peter Rousseeuw. *Clustering by Means of Medoids*. Reports of the Faculty of Mathematics and Informatics. Faculty of Mathematics and Informatics, 1987. URL <https://books.google.co.in/books?id=HK-4GwAACAAJ>.
- Michael Kifer, Georg Lausen, and James Wu. Logical Foundations of Object-Oriented and Frame-Based Languages. *Journal of the Association for Computing Machinery*, 42(4):741–843, 1995.
- Irena Koprinska and Sergio Carrato. Temporal Video Segmentation: A Survey. *Signal Processing: Image Communication*, 16(5):477–500, 2001.
- Sotiris B. Kotsiantis and Panagiotis Pintelas. Recent advances in clustering: A brief survey. *WSEAS Transactions on Information Science and Applications*, 1(1):73–81, 2004.
- Matej Kristan, Janez Perš, Matej Perše, and Stanislav Kovačič. Closed-World Tracking of Multiple Interacting Targets for Indoor-Sports Applications. *Computer Vision and Image Understanding*, 113(5):598–611, May 2009. ISSN 10773142. doi: 10.1016/j.cviu.2008.01.009.
- Ludmila I. Kuncheva. *Fuzzy Classifier Design*, volume 49 of *Studies in Fuzziness and Soft Computing*. Springer, 2000. ISBN 978-3-7908-2472-8.
- Ludmila I. Kuncheva. Diversity in Multiple Classifier Systems. *Information Fusion*, 6:3–4, 2005.
- Ora Lassila and Ralph R. Swick. Resource Description Framework (RDF) Model and Syntax Specification, 1999.
- Dawei Liang, Yang Liu, Qingming Huang, and Wen Gao. A Scheme for Ball Detection and Tracking in Broadcast Soccer Video. In *Advances in Multimedia Information Processing - PCM2005*, volume 3767, pages 864–875, 2005.
- Wei Lwun Lu, Kenji Okuma, and James J. Little. Tracking and Recognizing Actions of Multiple Hockey Players using the Boosted Particle Filter. *Image and Vision Computing*, 27(1-2):189–205, January 2009. ISSN 02628856. doi: 10.1016/j.imavis.2008.02.008.

- Wei Lwun Lu, Jo-Anne Ting, Kevin P. Murphy, and James J. Little. Identifying Players in Broadcast Sports Videos using Conditional Random Fields. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3249–3256, Providence, RI, 2011.
- Bruce D. Lucas. An Iterative Image Registration Technique with an Application to Stereo Vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, volume 2, pages 674–679, Vancouver, Canada, 1981. Morgan Kaufmann Publishers Inc.
- David Luebke and Greg Humphreys. How GPUs Work. *IEEE Computer*, 40(2):96–100, 2007.
- Sean Luke and Jeff Heflin. SHOE 1.01: Proposed Specification, 2000.
- Robert M. MacGregor. Inside the LOOM Description Classifier. *ACM Sigart Bulletin*, 2(3):88–92, 1991.
- Popa Marin, Dragan Mihaita, and Popa Mariana. Modeling and Simulation of a Team Game with Coloured Petri Nets. In *Proceedings of the 14th WSEAS International Conference on Computers*, volume 2, pages 516–520, 2010.
- Donald W. Marquardt. An Algorithm for Least-Squares Estimation of Nonlinear Parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11(2):431–441, 1963.
- Deborah L. McGuinness and Frank van Harmelen. OWL Web Ontology Language Overview, 2004.
- Thomas L. McKenzie. System for observing play and leisure activity in youth (soplay). *San Diego State University, San Diego, CA*, 2002.
- Lloyd L. Messersmith and Clum C. Bucher. The Distance Traversed by Big Ten Basketball Players. *Research Quarterly. American Association for Health, Physical Education and Recreation*, 10(3):61–62, 1939.
- Tom M. Mitchell. *Machine Learning*. McGraw-Hill, New York, London, 1997.
- Emad Monier, Per Wilhelm, and Ulrich Ruckert. Template Matching Based Tracking of Players in Indoor Team Sports. In *Third ACM/IEEE International Conference on Distributed Smart Cameras, 2009*, pages 1–6. IEEE, August 2009. doi: 10.1109/ICDSC.2009.5289408.
- Fernando C. Monteiro. *Region-Based Spatial and Temporal Image Segmentation*. PhD thesis, Faculty of Engineering of the University of Porto, 2008.
- Luís Mota and Luís P. Reis. Setplays: Achieving coordination by the appropriate use of arbitrary pre-defined flexible plans and inter-robot communication. In *Proceedings of the 1st International Conference on Robot Communication and Coordination, RoboComm '07*, pages 13:1–13:7, Piscataway, NJ, USA, 2007. IEEE Press. ISBN 978-963-9799-08-0.
- James Munkres. Algorithms for the Assignment and Transportation Problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1):32–38, 1957.
- Naveen Nandan. Live Analytics on High Velocity Sensor Data Streams using Event-based Systems. In *Journal of Industrial and Intelligent Information*, volume 1, pages 81–85, 2013.
- Chris J. Needham and Roger D. Boyle. Tracking Multiple Sports Players Through Occlusion, Congestion and Scale. In *British Machine Vision Conference*, pages 93–102, Manchester, UK, 2001.

- Michael Negnevitsky. *Artificial Intelligence: A Guide to Intelligent Systems*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 2001. ISBN 0201711591.
- NTIA. U.S. Frequency Allocation Chart. Technical report, U.S. Department of Commerce: National Telecommunications and Information Administration Office of Spectrum Management, 2011. URL http://www.ntia.doc.gov/files/ntia/publications/spectrum_wall_chart_aug2011.pdf.
- Peter O'Donoghue. *Research Methods for Sports Performance Analysis*. Taylor & Francis LTD, 2010. ISBN 9780415496230.
- Kenji Okuma, Ali Taleghani, Nando De Freitas, James J. Little, and David G. Lowe. A Boosted Particle Filter : Multitarget Detection and Tracking. In *Proceedings of ECCV*, pages 1–11, 2004.
- Nobuyuki Otsu. A Threshold Selection Method from Gray-Level Histograms. *IEEE Transaction on Systems, Man and Cybernetics*, 9(1):62–66, 1979.
- Roxane Ouellet and Uche Ogbuji. Introduction to DAML: Part I, 2002. URL <http://www.xml.com/pub/a/2002/01/30/daml1.html>.
- V. Pallavi, Jayanta Mukherjee, Arun K. Majumdar, and Shamik Sural. Graph-based Multiplayer Detection and Tracking in Broadcast Soccer Videos. *IEEE Transactions on Multimedia*, 10(5): 794–805, 2008a.
- V. Pallavi, Jayanta Mukherjee, Arun K. Majumdar, and Shamik Sural. Ball Detection from Broadcast Soccer Videos Using Static and Dynamic Features. *Journal of Visual Communication and Image Representation*, 19(7):426–436, 2008b.
- Matej Perše, Matej Kristan, Stanislav Kovačič, Goran Vučkovič, and Janez Perš. A Trajectory-Based Analysis of Coordinated Team Activity in a Basketball Game. *Computer Vision and Image Understanding*, 113(5):612–621, 2009.
- Matej Perše, Matej Kristan, Janez Perš, Gašper Mušič, Goran Vučkovič, and Stanislav Kovačič. Analysis of Multi-Agent Activity Using Petri Nets. *Pattern Recognition*, 43(4):1491–1501, 2010.
- Carl A. Petri. Communication with Automata. *Applied Data Research, Princeton, AF*, 1966.
- Prozone. Pioneers in sport performance analysis and data insight - Prozone Sports, 2015. URL <http://www.prozonesports.com/>.
- Luís P. Reis and Nuno Lau. FC portugal team description: Robocup 2000 simulation league champion. In *RoboCup 2000: Robot Soccer World Cup IV*, pages 29–40, 2000. doi: 10.1007/3-540-45324-5_2.
- Luís P. Reis and Nuno Lau. Coach Unilang - A Standard Language for Coaching a (Robo)Soccer Team. In Andreas Birk 0002, Silvia Coradeschi, and Satoshi Tadokoro, editors, *Robocup 2001: Robot Soccer World Cup V*, volume 2377 of *Lecture Notes in Computer Science*, pages 183–192, 2001.
- Luís P. Reis, Nuno Lau, and Eugenio Oliveira. Situation based strategic positioning for coordinating a team of homogeneous agents. In Markus Hannebauer, Jan Wendler, and Enrico Pagello, editors, *Balancing Reactivity and Social Deliberation in Multi-Agent Systems*, volume 2103 of *Lecture Notes in Computer Science*, pages 175–197. Springer, 2000. ISBN 3-540-42327-3.

- Jinchang Ren, James Orwell, Graeme A. Jones, and Ming Xu. Tracking the soccer ball using multiple fixed cameras. *Computer Vision and Image Understanding*, 113(5):633–642, 2009.
- Lawrence G. Roberts. *Machine Perception of Three-Dimensional Solids*. PhD thesis, Massachusetts Institute of Technology, 1963.
- Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, London, 1995.
- Jaime Sampaio, Manuel Janeira, Sergio Ibáñez, and Alberto Lorenzo. Discriminant Analysis of Game-Related Statistics Between Basketball Guards, Forwards and Centres in Three Professional Leagues. *European Journal of Sport Science*, 6(3):173–178, 2006.
- Catarina B. Santiago, Armando Sousa, Luís P. Reis, Maria L. Estriga, and Martin Lames. Survey on team tracking techniques applied to sports. In *Proceedings of International Conference on Autonomous and Intelligent Systems*, Póvoa de Varzim, Portugal, 2010a.
- Catarina B. Santiago, Armando Sousa, Luís P. Reis, Maria L. Estriga, and Martin Lames. Ball detection and tracking using color features. In *Proceedings of the 5th Doctoral Symposium in Informatics Engineering*, volume 5150, pages 215–226, Porto, Portugal, 2010b.
- Catarina B. Santiago, Luís P. Reis, Rosaldo J. Rossetti, and Armando Sousa. Foundations for creating a handball sport simulator. In *Proceedings of the 3rd Workshop on Intelligent Systems and Applications (WISA), 6th Iberian Conference on Information Systems and Technologies*, Chaves, Portugal, 2011a.
- Catarina B. Santiago, Armando Sousa, Luís P. Reis, and Maria L. Estriga. Real time colour based player tracking in indoor sports. 19:17–35, 2011b. doi: 10.1007/978-94-007-0011-6_2.
- Catarina B. Santiago, Lobinho Gomes, Armando Sousa, Luís P. Reis, and Maria L. Estriga. Tracking players in indoor sports using a vision system inspired in fuzzy and parallel processing. *Cutting Edge Research in New Technologies*, pages 117–140, 2012a.
- Catarina B. Santiago, João L. Oliveira, Luís P. Reis, Armando Sousa, and Fabien Gouyon. Overcoming motor-rate limitations in online synchronized robot dancing. *International Journal of Computational Intelligence Systems*, 5(4):700–713, 2012b. doi: 10.1080/18756891.2012.718120.
- Catarina B. Santiago, Armando Sousa, and Luís P. Reis. Pseudo fuzzy colour calibration for sport video segmentation. In *Proceedings of VipIMAGE 2011 - 3rd ECCOMAS Thematic Conference on Computational Vision and Medical Image Processing*, pages 361–366, Olhão, Portugal, 2012c.
- Catarina B. Santiago, Armando Sousa, and Luís P. Reis. Vision system for tracking handball players using fuzzy color processing. *Machine Vision and Applications*, 24(5):1055–1074, 2013. doi: 10.1007/s00138-012-0471-z.
- Saratha Sathasivam and Wan A. Abdullah. Logic learning in hopfield networks. *Modern Applied Science*, 2(3), 2008.
- Thuraiappah Sathyan, David Humphrey, and Mark Hedley. WASP : A System and Algorithms for Accurate Radio Localization Using Low-Cost Hardware. *Most*, 41(2):211–222, 2011.

- Robert E. Schapire, Yoav Freund, Peter Bartlett, and Wee S. Lee. Boosting the Margin: A New Explanation for the Effectiveness of Voting Methods. *The Annals of Statistics*, 26(5):1651–1686, 1998. ISSN 00905364. doi: 10.2307/120016.
- Andrea Seabrook. GPS Units Get A Fix On The Football : NPR, 2008. URL <http://www.npr.org/templates/story/story.php?storyId=98564245>.
- Daniel Setterwall. *Computerised Video Analysis of Football – Technical and Commercial Possibilities*. PhD thesis, Royal Institute of Technology, 2003.
- Mehmet Sezgin and Bulent Sankur. Survey Over Image Thresholding Techniques and Quantitative Performance Evaluation. *Journal of Electronic imaging*, 13(1):146–165, 2004.
- Pedro Silva, Catarina B. Santiago, Luís P. Reis, Armando Sousa, João Mota, and Greg Welk. Assessing Physical Activity Intensity Using a Video Based Approach. In *3rd International Conference on Ambulatory Monitoring of Physical Activity and Movement*, 2013.
- Pedro Silva, Catarina B. Santiago, Luís Reis, Armando Sousa, Jorge Mota, and Greg Welk. Assessing physical activity intensity by video analysis. *Physiological Measurement*, 36(5):1037–1046, 2015.
- Irwin E. Sobel. Camera Models and Machine Perception. Technical report, DTIC Document, 1970.
- Matt Spencer, Claire Rechichi, Steven Lawrence, Brian Dawson, David Bishop, and Carmel Goodman. Time-Motion Analysis of Elite Field Hockey During Several Games in Succession: a Tournament Scenario. *Journal of Science and Medicine in Sport*, 8(4):382–391, 2005.
- Tomislav Staroveski, Dubravko Majetic, Toma Udiljak, and Viktor Mihaljevic. Machine Vision System for Seam Weld Detection in Longitudinally Welded Piped. In *Annals of Daaam for 2008 & Proceedings of the 19th International Daaam Symposium*, pages 1303–1304, 2008.
- Andreas Stelzer, Klaus Pourvoyeur, and Alexander Fischer. Concept and Application of LPM - A Novel 3-D Local Position Measurement System. *IEEE Transactions on Microwave Theory Techniques*, 52(12):2664–2669, 2004.
- Tsuyoshi Taki, Jun-ichi Hasegawa, and Teruo Fukumura. Development of Motion Analysis System for Quantitative Evaluation of Teamwork in Soccer Games. In *Proceedings of the International Conference on Image Processing*, volume 3, pages 815–818. IEEE, 1996.
- Xiaofeng Tong, Tao Wang, Wenlong Li, Yimin Zhang, Bo Yang, Fei Wang, Lifeng Sun, and Shiqiang Yang. A Three-Level Scheme for Real-Time Ball Tracking. In *Proceedings of the 2007 International Conference on Multimedia Content Analysis and Mining*, pages 161–171, 2007.
- Xiaofeng Tong, Jia Liu, Tao Wang, and Yimin Zhang. Automatic Player Labeling, Tracking and Field Registration and Trajectory Mapping in Broadcast Soccer Video. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(2):15, 2011.
- CPN Tools. CPN Tools Homepage, 2015. URL <http://cpntools.org/start>.
- Ubisense. Real-time location systems (RTLS) and geospatial consulting - Ubisense, 2012. URL <http://www.ubisense.net/en/>.

- Minna Väkevä, Kimmo Rautkoski, Mårten Ström, and F. von Schoultz. NAVIndoor 2 Final Report. Technical report, Space Systems, 2004.
- Sreenath R. Vantaram and Eli Saber. Survey of Contemporary Trends in Color Image Segmentation. *Journal of Electronic Imaging*, 21(4), 2012.
- Rachel E. Venter, Eben Opperman, and Simon Opperman. The Use of Global Positioning System (GPS) Tracking Devices to Assess Movement Demands and Impacts in Under-19 Rugby Union Match Play. *African Journal for Physical, Health Education, Recreation and Dance (AJPHERD)*, 17(1):1–8, 2011.
- Goran Vučković, Janez Perš, Nic James, and Mike Hughes. Measurement error associated with the SAGIT / Squash computer tracking software. *European Journal of Sport Science*, 10(2): 129–140, March 2010. ISSN 1746-1391. doi: 10.1080/17461390903311927. URL <http://www.tandfonline.com/doi/abs/10.1080/17461390903311927>.
- Brian C. Wadell, Robert J. McCarthy, Eric L. Gravengaard, Eric Spitz, and Vahe Katros. Local area multiple object tracking system.
- J. Wang. Pseudolite Applications in Positioning and Navigation: Progress and Problems. *Journal of Global Positioning Systems*, 1:48–56, 2002.
- Jiacun Wang. *Timed Petri Nets: Theory and Application*. Kluwer Academic PublisherGroup, 1998.
- Jinjun Wang, Changsheng Xu, Engsiong Chng, Hanqing Lu, and Qi Tian. Automatic Composition of Broadcast Sports Video. *Multimedia Systems*, 14(4):179–93, 2008.
- Greg Welch and Gary Bishop. An Introduction to the Kalman Filter. Technical Report 1, Department of Computer Science University of North Carolina at Chapel Hill, EUA, North Carolina, 2002.
- Wikipedia. Trilateration - Wikipedia, the free encyclopedia, 2011. URL <http://en.wikipedia.org/wiki/Trilateration>.
- Wikipedia. Handball - wikipedia, the free encyclopedia, 2015. URL <http://en.wikipedia.org/wiki/Handball>.
- Ben Wisbey, Paul G. Montgomery, David B. Pyne, and Ben Rattray. Quantifying Movement Demands of AFL Football using GPS Tracking. *Journal of Science and Medicine in Sport*, 13(5):531–536, 2010.
- Athanasios Yiannakos, P. Sileoglou, Vassilis Gerodimos, Panagiota Triantafillou, Vasilis Armatas, and Spyros Kellis. Analysis and Comparison of Fast Break in Top Level Handball Matches. *International Journal of Performance Analysis in Sport*, 5(3):62–72, 2005.
- Alper Yilmaz, Omar Javed, and Mubarak Shah. Object Tracking: A Survey. *ACM Computing Surveys (CSUR)*, 38(4), 2006.
- Lotfi A. Zadeh. Fuzzy Sets. *Information and Control*, 8:338–353, 1965.
- Guangyu Zhu, Changsheng Xu, Qingming Huang, Yong Rui, Shuqiang Jiang, Wen Gao, and Hongxun Yao. Event Tactic Analysis Based on Broadcast Sports Video. *IEEE Transactions on Multimedia*, 11(1):49–67, 2009.

Richard Zurawski and MengChu Zhou. Petri Nets and Industrial Applications: A Tutorial. *IEEE Transactions on Industrial Electronics*, 41(6):567–583, 1994.

ZXY. Welcome to ZXY Sport Tracking A, 2012. URL <http://www.zxy.no/>.