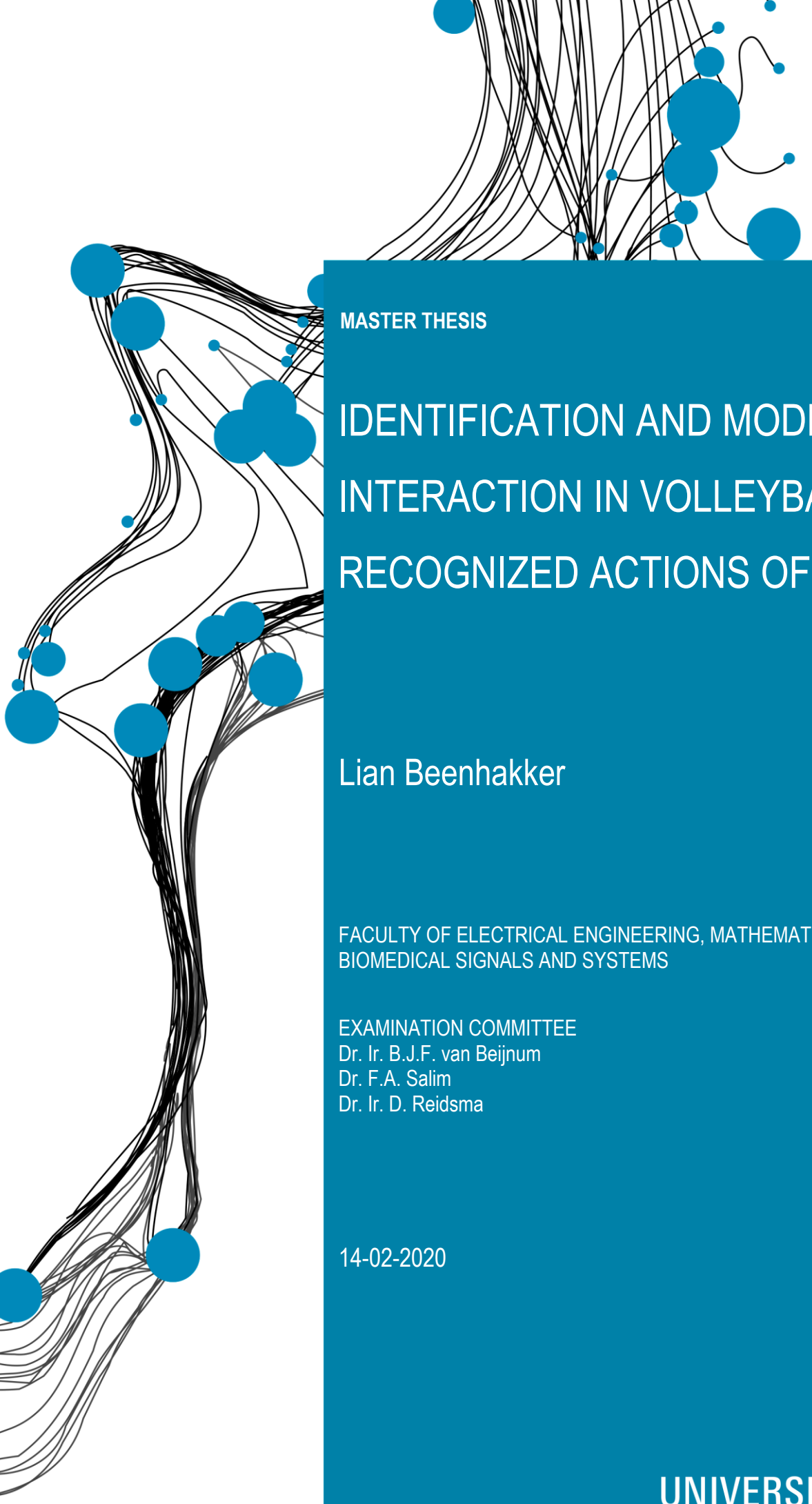


MASTER THESIS

IDENTIFICATION AND MODELLING OF INTERACTION IN VOLLEYBALL USING RECOGNIZED ACTIONS OF PLAYERS

Lian Beenhakker

FACULTY OF ELECTRICAL ENGINEERING, MATHEMATICS AND COMPUTER SCIENCE
BIOMEDICAL SIGNALS AND SYSTEMS

EXAMINATION COMMITTEE
Dr. Ir. B.J.F. van Beijnum
Dr. F.A. Salim
Dr. Ir. D. Reidsma

14-02-2020

UNIVERSITY OF TWENTE.

Preface

With this document, I present my master thesis: *Identification and modelling of interaction in volleyball using recognized actions of players* and this document also defines the end of my master Biomedical Engineering. When I started with my thesis in May 2019, I relatively quickly found the idea of volleyball complexes in literature and the more I learned about those complexes, the more I also learned about volleyball. To model these complexes, I have tried many different approaches, from machine learning on more than 1000 features to Hidden Markov Models to predict the next complex. In the end, Labelled Transition Systems turned out to give a realistic representation of the complexes and I think these models are also interpretable to coaches.

This thesis is part of the Smart Sports Exercises (SSE) project which combines technology with volleyball. I really like the fact that this project (and thus my thesis) is sports related as usually Biomedical Engineering is more related to clinical problems. As part of the SSE project, there were weekly meetings to discuss all aspects of the project. These were really useful to keep the overview over the project as a whole, instead of only focussing on my own part.

My final words of this preface I want to spend on thanking a few people who supported and helped me throughout my thesis. First of all, I want to thank Bert-Jan and Fahim for all meetings and all discussions we have had on the different approaches of modelling the interaction. Especially the few meetings we had to go into further detail have been helpful throughout the process. Also, over the last few weeks, both of your feedback has been really useful to improve my report to what it has become. Next to that, I also want to thank Dennis for being external member of my examination committee and for all the feedback given on my report, which has been really relevant on improving my thesis.

As mentioned, my graduation project was part of the SSE project and so I was allowed to attend the weekly project meetings. As they were very interesting and useful, I want to thank Bert-Jan, Dees, Dennis, Fahim and Robby for letting me join these meetings and for all the small tips and tricks that came along sometimes.

In the end, I want to thank my friends for all the support and interest in my thesis and my coffee- and lunch buddies for all the breaks in which we talked about our theses for new inspiration and for all the breaks in which we just played card games to get our mind off of our theses. Also, I want to thank my family for all the interest you had in my thesis and even though you might not always have understood what I was doing, I still appreciated it very much. Last but not least, I want to thank Joost for all the support you gave, for all endless moments where I could just tell you what I was doing to give myself new ideas, for all proofreading of my thesis over and over and for just being there for me.

I hope you enjoy reading my thesis, as I have had a good time writing it.

Lian Beenhakker

Enschede, February 2020

Summary

Volleyball is the 5th most popular sport in the world with more than 100,000 players in the Netherlands alone. During a volleyball training, a coach cannot keep an eye on all players and therefore cannot give feedback to everyone on the performance of actions, both on individual level as well as team play/interaction level. This can be solved by using technology and giving sensors to players to recognize actions. These can then be used to recognize team play. This is also the goal of the Smart Sports Exercises (SSE) project as this project aims to support training and coaching in volleyball using an interactive floor and sensors worn by players. Using sensor data, individual actions can already be recognized. Therefore, this thesis focusses on the next step (team play) with as main research question: *How can the interaction between volleyball players be identified and modelled based on recognized player actions?*

In context of this thesis, interaction in volleyball is linked to team play and sequences of actions. These sequences of actions can be used to divide a rally into six different complexes. The complexes represent different parts of a match and have previously been used in combination with Social Network Analysis (SNA) to analyse volleyball matches. As SNA has already been used in volleyball analysis, this thesis introduces Labelled Transition Systems (LTS) as a different approach to analyse volleyball.

The LTS is created either by design or data-driven. Data collected during two measurement sessions is used as input for any LTS to update the weights and show an LTS graph to a coach. Models created by design can be used for rallies or training exercises linked to the complexes. With the current approach, models created data-driven can be used if a coach wants to practise an exercise that does not fit within the framework of complexes.

Actions are recognized by a previously developed action recognizer which has an Unweighted Average Recall of 67.87%. Errors made by the action recognizer influence the output of the LTS. Using different confusion matrices, the influences of different types of errors are identified and analysed. This results in the requirement of the action recognizer to recognize so-called non-Freeball actions with a recall of at least 95% at the cost of Freeball actions, which can be recognized with a recall of 80%.

Using the LTS in the SSE project is an effective way to keep track of the complexes and actions performed by players. The main reason is the fact that there are two ways to create a model and therefore these models can be used to analyse interaction in both training settings as well as during matches. The interactive floor, which is part of the SSE project, can be used to give feedback about the complexes to the players, but there is some delay in recognizing the current state.

Future research can focus on improving the LTS to accommodate to more actions like an action in which players try to block (but fail) or an action like a fake smash. Other directions can focus on feedback given to players and coaches or the improvement of the action recognizer.

Samenvatting

Volleybal is de vijfde meest populaire sport ter wereld en in Nederland zijn er meer dan 100.000 spelers. Tijdens een training is het lastig voor een coach om alle spelers in te gaten te houden en terugkoppeling te geven over hoe ze oefeningen uitvoeren, zowel op individueel niveau als op groepsniveau. Dit kan verholpen worden door spelers sensoren te geven zodat de acties die ze uitvoeren herkend kunnen worden. Dit is ook het doel van het Smart Sports Exercises (SSE) project, naast het geven van feedback via een interactieve vloer. Op basis van de sensor data kunnen individuele acties herkend worden, daarom richt dit onderzoek zich op de volgende stap, namelijk het herkennen van acties op groepsniveau/interactie. De hoofdvraag is dan ook: *Hoe kan de interactie tussen volleybal spelers herkent en gemodelleerd worden op basis van herkende individuele acties van spelers?*

In context van dit onderzoek is de term interactie in volleybal verbonden acties die op groepsniveau worden uitgevoerd en reeksen van acties. Deze reeksen kunnen opgedeeld worden in zes verschillende complexen en elk complex staat voor een ander deel van een rally. Het analyseren van een volleybalwedstrijd met behulp van deze complexen is eerder gedaan door Social Network Analysis (SNA) toe te passen. Gezien dit al gebruikt is als model, wordt in dit onderzoek gekeken naar Labelled Transition Systems (LTS) als een andere manier om volleybal te modelleren.

De LTS wordt gemaakt door hem te ontwerpen en door de data van trainingen te gebruiken. Tijdens twee meetsessies is data verzameld dat als input gebruikt kan worden voor de modellen. Op basis van deze data kan een waarde aan elke transitie worden gehangen die aangeeft hoe vaak de transitie voorkomt. Dit kan dan gebruikt worden om aan een coach te laten zien. Modellen die ontworpen zijn kunnen in wedstrijden gebruikt worden of in trainingen waarbij oefeningen gedaan worden die met de complexen te maken hebben. Modellen die aan de hand van data gemaakt zijn, kunnen worden gebruikt in trainingen waarbij een coach een nieuw soort oefening wil doen die niet per se gekoppeld is aan de complexen.

Het herkennen van individuele acties gaat op dit moment met een eerder ontwikkelde classifier met een ongewogen gemiddelde sensitiviteit van 67.87%. Fouten die gemaakt worden bij deze herkenning hebben invloed op het uiteindelijke LTS dat als output wordt laten zien. Door de gelabelde acties op verschillende manieren random te veranderen, is deze invloed onderzocht. Hier is uitgekomen dat de classifier het beste acties die geen zogenoemde vrije bal veroorzaken met minstens 95% sensitiviteit moeten worden herkend zolang de acties die wel een vrije bal veroorzaken met minstens 80% worden herkend.

Het gebruik van een LTS model in het SSE project is een effectieve manier om bij te houden welke sequenties van acties de spelers uitvoeren. Dit komt doordat er twee manieren zijn om het model te creren, waardoor de modellen toepasbaar zijn tijdens trainingen en tijdens wedstrijden. De interactieve vloer van het SSE project, kan worden gebruikt om feedback te geven aan de spelers, maar er is vertraging in het herkennen van de huidige toestand.

Toekomstig onderzoek kan zich richten op het verder uitbreiden van het model. Dit kan worden gedaan door meer acties toe te voegen, zoals bijvoorbeeld een block dat mislukt of een schijnsmash. Andere richtingen zijn het geven van feedback van spelers of het verbeteren van de classifier die acties herkent.

Contents

1 Introduction

Volleyball is the 5th most popular sport in the world [1] with more than 100,000 players in the Netherlands alone [2]. In a volleyball match, teams play six-against-six to get the ball on the floor within the boundaries of the opponent's field. The most important rule is that teams can play the ball at most three times when attacking. Therefore, a widely used attack strategy is to use the first action to defend the service/attack of the opponents, by e.g. a ForearmPass. The ball is then passed to the setter who gives a set up, usually with an OverHeadPass, so that one of the attackers can perform a Smash to finish the attack.

During a match, especially at higher levels, teams are expected to have a video scout to label all actions performed by players during a match by hand [3]. This data, including video images of the match, is shared with all possible opponents. In that way, every team can have the ultimate preparation for every match as they can adjust their own strategy based on the opponent's expected strategy.

Further match preparation is done by performing different exercises during training sessions. Coaches come up with these exercises to practise situations that might come along during a match. A simple example of an exercise related to the above attack strategy is as follows: first, a player serves to another player who has to receive the ball. A third player gives a set up so a fourth player can place the attack after which the exercise ends. An exercise like this can be repeated and practised over and over while taking turns in serving, receiving, setting and attacking.

During the execution of an exercise, a coach wants to give all players feedback on how they can improve their performance of actions. It is not only important that players improve their own performance, it is even more important that team play is performed to perfection. As an example, the defending ForearmPass does not need to be ideal as long as it does lead to the best possible Smash for the attackers. For a coach, it is thus not only necessary to focus on individual actions, but also focus on how the team as a whole performs sequences of actions.

Unfortunately, it is difficult for a coach to keep an eye on everyone during a training session. Next to that, video scouting is currently very labour-intensive as a video scout has to annotate all actions by hand. This is where technology might come in useful to give a helping hand to both the coach as well as the video scout and with that also the volleyball players.

Players can be given sensors to measure their arm movement and this data can be used to recognize actions performed. This information can help the video scout in annotating all actions during a match, but more importantly, these recognized actions can also be used to identify team play and sequences of actions performed. This information can help a coach in giving feedback to players.

This is where the Smart Sports Exercises (SSE) project comes in. The SSE project aims to support training and coaching in volleyball using an interactive floor [4]. This interactive floor has pressure sensors to possibly identify player position whilst it can also display interactive images as feedback to the players. Next to the usage of the floor, players are asked to wear sensors to recognize actions performed.

The approach described above can be visually presented in a layered approach as given in figure 1. The first layer consists of the measurement of the arm motion of individual players by any type of sensors. This sensor data can be used to identify individual actions of players at the second layer. At the third layer, these actions can be used to recognize patterns in the sequences of actions. Once team play is identified, output can, in any form, be presented to the coach or volleyball players. From here on, all terms that describe this third layer, like team play, sequences of actions and patterns in actions, are all covered using the single term *interaction*.

There is one thing to keep in mind when using the actions recognized from the arm movements. As these actions follow from sensor data, it can be that there are mistakes in the classification of actions. This might influence the recognition of the interaction.

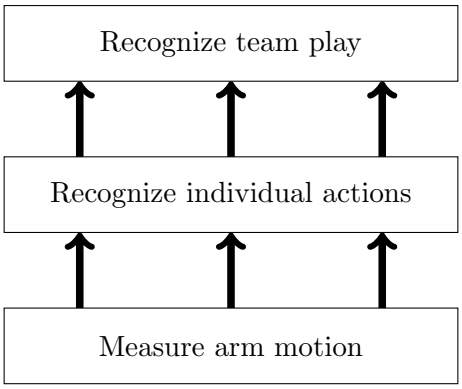


Figure 1: Visualization of a layered approach: Arm movements are measured with sensors which is used to recognize individual actions. These actions can be used to find sequences of actions which can be seen as team play. The output can be shown in any form to coaches or volleyball players.

In the SSE project, research has already been done to identify actions based on the arm movements of players [5, 6]. The next step is to use these actions to identify interaction. Next to being able to identify interaction, it is also important to create a model which can represent the interaction to show as output to e.g. coaches. The main research question of this thesis is therefore: *How can the interaction between volleyball players be identified and modelled based on recognized player actions?*

This research question can be divided into some smaller sub-questions. By answering these sub-questions, the main research question can also be answered. This first sub-question is: *What is interaction in volleyball and how can it be described?* As mentioned already above, the term interaction is used to cover all terms defining team play, but more literature research needs to be done to further understand the term interaction and what it represents in volleyball.

Once it is known how interaction can be described, a closer look can be taken into available models. Therefore, the second sub-question is: *What type of model can be used to represent the interaction?* This question can be answered by looking into state-of-the-art models for volleyball. It might be that these state-of-the-art models are already useful for the SSE project, but it can also be that another type of model is more compatible within the context of figure 1. The type of model found by this question can also be used as one of the ways to present the output of the interaction layer to coaches.

As a model has to be valuable to the SSE project, it is important to look into different ways to construct a model, even though it might be that a model has already been created in previous studies. It is important that the model represents the structural patterns that occur in volleyball so that these can be tracked. Therefore, the third sub-question is: *How can a model be created that represents the structural patterns in actual volleyball interactions?*

Once one or more structural models are created, these can be used to represent information on what happens during a training session or match. This information consists of the sequences of actions performed by players. The output model containing all this information can then be presented as output to a coach. This leads to the fourth sub-question: *How can individual actions that occur in a volleyball session be used to update information presented by the model?*

The final sub-question is linked to the previous as the actions performed by players are recognized using a classification algorithm. These actions and sequences of actions are used to update information in the model, however, the information can differ if errors occur due to the action recognizer. It is important to see how realistic the output is, given different classification errors. The last sub-question is therefore: *What is the influence of recognition errors when automatic action classification is used to update the model for interaction?*

Combining these sub-questions should make it possible to answer the main research question. To answer the sub-questions, this thesis is divided into different chapters. The first two sub-questions are addressed in the background in chapter 2. This is done by taking a closer look into interaction and state-of-the-art modelling in volleyball. In chapter 3, the methods are described by giving details on how a model can be created. Next to that, the fourth sub-question is addressed on how the information in a model can be updated and a method is described to examine the influence of errors in action classification. The results are presented in chapter 4 and in the subsequent discussion chapter, these results are examined in detail. Chapter 5 also focusses on the influence of errors in the action recognizer by giving a requirement for the action recognizer. Finally, in chapter 6, this thesis is concluded and all information is wrapped up to answer the main research question. Also, some future directions are pointed out.

2 Background

As volleyball is a popular sport, it is also an interesting topic to study. A review by Silva, Marcelino, Lacerda and João [7] discusses 34 articles which are about skills and how these are related to success, player position and match phase. This chapter consists of a literature study about interactions in volleyball to go into the first sub-question: *What is interaction in volleyball and how can it be described?* The second sub-question (*What type of model can be used to represent the interaction?*) is also addressed by describing the state-of-the-art models in volleyball and new ways to model interaction.

2.1 (Inter)actions in volleyball

When picturing volleyball, actions that come to mind immediately are the serve, the pass, the set-up and the smash. These are also actions most often mentioned in literature, together with the dig, block and free ball [8, 9]. Instead of looking at specific actions, performed actions can also be divided into the three types of actions that can be performed: receiving, setting and spiking [10].

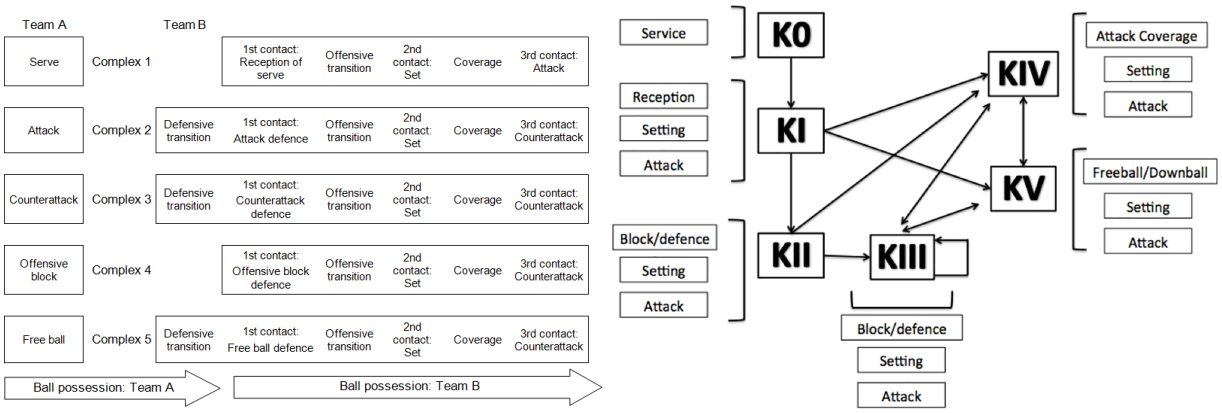
As described in the introduction, it is not only interesting to look at individual player actions, it is also useful to look at the interaction between players. The term interaction was introduced in this thesis to cover all terms related to team play. In the Cambridge dictionary, the definition for interaction is *'An occasion when two or more people or things communicate with or react to each other'* [11]. Interaction as a term to cover all terms related to team play therefore makes sense as players communicate with each other about who performs what action and react on actions performed by others.

This is also described by Beniscelli, Tenenbaum, Schinke and Torregrose [12] as they write that *'Interaction is related to the level of coordination among players'*. Analysing the interaction between players can give information on who can perform what actions best (in terms of receiving, setting and spiking) and thus who should perform these actions during a match [13]. Looking at the interaction in a broader way could even give information on which strategies might work for a team, but also which do not work [8]. Part of interaction is also the communication about tactical decisions either verbal or non-verbal [14]. An example of non-verbal communication is using hand gestures by the setter to communicate about the tactics. By this, players not involved in this specific attack strategy can perform fake smashes to distract the opponent. Furthermore, if one player fails to perform an action successfully, team mates might solve this by performing a non-scripted/non-expected action to still try and save the ball [12].

This already gives some insights for the first sub-question (*What is interaction in volleyball and how can it be described?*), as the term interaction can indeed be used to cover all terms related to team play. Following on this information on what interaction in volleyball is, a closer look can be taken into a way to describe interaction. One way to look at it is by dividing a rally into separate complexes [15]. Hileno and Buscá [15] described a tool to methodologically observe these complexes in a rally. Using this tool, information of a rally is saved. This information is about where in the field what type of action is performed. Next to this, other variables such as the number of blockers and the tempo of the attack can also be described with this tool. The tool has showed to be successful as several studies have used (an adjusted version of) this tool [16, 17, 18, 19, 20]. It differs per study which variables are taken into account.

A rally is thus divided into different phases depending on what has happened and how long the rally has lasted. In total, there are six complexes, counting from K0 to K5. Figure 2 shows the description of the complexes (figure 2(a)) and how they relate to one another (figure 2(b)). As these complexes are used to identify interaction, more elaboration is given on the different complexes including the name of the complex used in literature.

Starting with the serving complex (K0, serve) and going to the reception of service (K1, side-out), the first two complexes are clearly defined. In the second (K2, side-out transition)



(a) Actions performed in the individual complexes. Actions on the left are performed by the opponents and the set of actions on the right is then performed in the given complex. Adapted from [15].

(b) Relations between the different complexes. The arrows show which complexes can follow each other. As K0 is the serving complex, the rally starts in K0. Adapted from [20].

Figure 2: Overview of complexes as defined in literature. Each complex is a different phase of a rally depending on the actions performed by the opponents.

as well as in the third complex (K3, transition of transition), the opponent's attack is defended and a new attack is created. The main difference between these complexes is that in K2 the first attack of the rally is defended, whereas in K3, any next attack is defended (so a counter attack).

In some studies complexes are combined with each other to make analysis easier [19, 20], however, this is not preferable. K2 and K3 should be kept separate as the phase of the rally between K2 and K3 differs. During serve, players have to stand on specific positions determined by the rules of volleyball and this influences K2, whereas from K3 on, both teams have already attacked at least once and already moved around to other positions. Next to that, in K3 players are more fatigued which influences the attack tempo of K3 compared to K2 [19].

K4 (attack coverage) and K5 (freeball and downball) do not follow from an attack or counter attack and therefore occur when players do not perform the complex in the three steps as shown in figure 2(a). In K4, the opponent successfully blocked the attack placed, meaning the team that has just attacked has to defend the block immediately. This complex differs from K2/K3 as there is less time to get ready to defend. In K5, the opponent failed to attack in a desired manner, sending out a so-called freeball or downball. A freeball is an easy ball caused by any action but a smash and a downball is a smash-like action performed in standing position (without jumping) which makes it easier to defend. The fact that these balls are easy to defend makes that a team can put more focus on creating their attack, which might therefore be more successful.

As each of these complexes represent a part of the rally, a team might be interested in what complexes they can perform well and which are difficult to them during rallies. Based on this information, coaches can come up with exercises to train specific aspects of complexes to improve the overall performance of a team.

2.2 Recognition of individual actions

As mentioned in the introduction, in the SSE project, volleyball players are asked to wear sensors to identify actions based on arm movements [5, 6]. In these specific studies, players wore two Intertial Measurement Unit (IMU) sensors, one for each wrist. IMUs contain an accelerometer, a gyroscope and a magnetometer which respectively measure acceleration, angular velocity and the Earth's magnetic field. As the second layer shown in figure 1 is about recognizing individual actions, some background about action recognition in volleyball is given.

In (beach)volleyball, action recognition has been performed using machine learning on IMU

data. This has already led to some interesting results, for example in skill recognition [21], serve type recognition [22] and action recognition [23]. Recently, two papers were published about action recognition in volleyball using two machine learning models [5, 6]. The first paper describes a classifier which labels IMU data as action or non-action [5]. In the second paper, the IMU data that is classified as action is used as input for a second classifier which identifies the specific action performed [6]. The first classifier recognizes actions/non-actions for windows of 0.5 seconds with an overlap of 50% [5]. The second classifier combines the data of all windows over which an action reaches to determine the action performed [6]. For each individual player, it is thus known what action is performed every 0.25 sec.

In both studies, the data sets are unbalanced. Therefore, instead of using accuracy as performance measure, the Unweighted Average Recall (UAR) is used [5, 6]. This measure uses the recall for all individual classes and averages these, without taking the number of samples into account. The UAR of the first model reaches up to 86.87% [5], whereas the second model reaches only to 67.87% [6]. The actions that are recognized best are ForearmPass (81.6%), OverHeadPass (87.3%), Serve (71.9%), Smash (80.7%) and UnderHandServe (93.9%). The actions with which the model has more difficulties are Block (37.5%) and OneHandPass (22.2%). This can most likely be explained because these actions do not occur that often [6].

2.3 Models to represent interaction

With the possibility to recognize individual player actions, the next step is to go from these actions to interaction in terms of complexes. This is also shown in the layered approach of figure 1. There are different types of models that can be created to represent the interaction, in this section some details on different models are presented.

The tool for complexes [15] is used in different studies [17, 18, 19, 20] to create a Social Network (SN). An example SN is shown in figure 3. Social Network Analysis (SNA) was introduced to the field of volleyball in 2016 [17] using graph theoretical approaches (GT) [20]. In GT, a graph is created consisting of nodes and edges [24]. The edges are links between the nodes and can be either undirected (two nodes are linked) or directed (as a one-way path) [24].

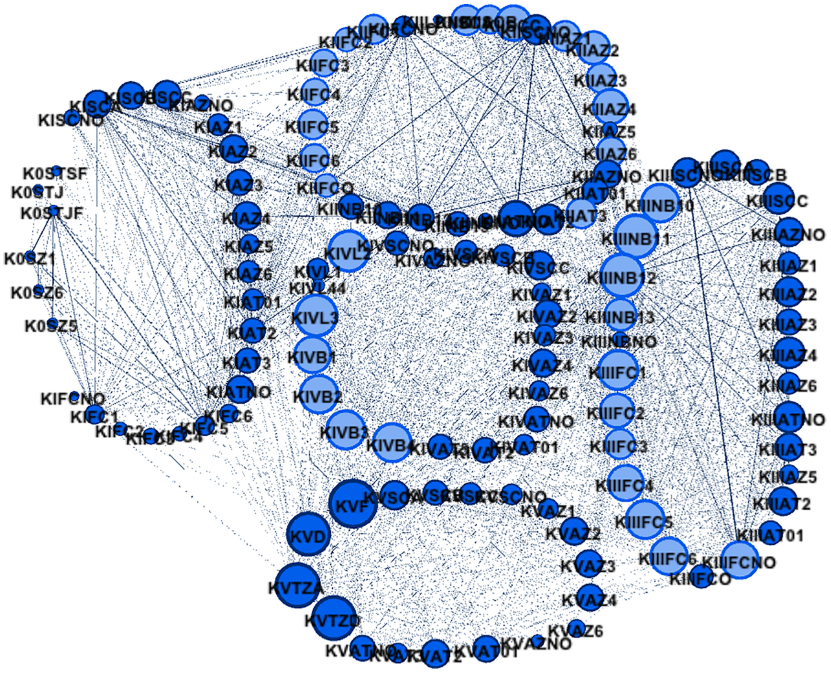


Figure 3: Example of a Social Network with Eigenvector Centrality to determine the importance of different nodes. The nodes are sorted per complex and nodes that occur simultaneously or consecutively are connected. Adapted from [20].

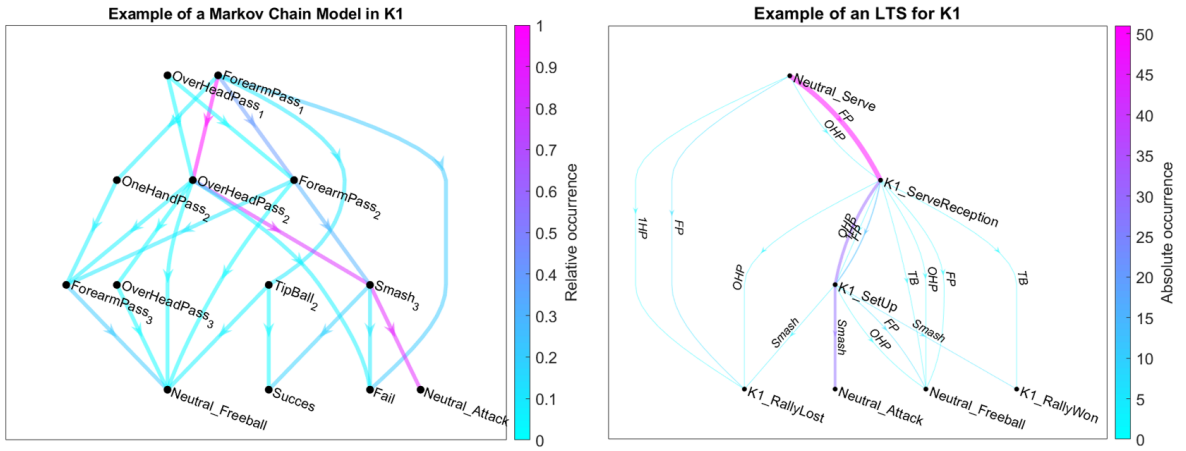
In the SN created for SNA, variables like number of blockers, tempo of attack and serve type are used as nodes [20]. For each variable, there are a number of options possible and thus several nodes. For example, in K0, there are two variables, serve zone (SZ) and serve type (ST). Both these variables have three options; SZ is either zone 1, zone 5 or zone 6, whereas ST has either float jump serve (JF), jump serve (J) or standing serve (SF) [20]. There are thus six nodes representing K0: K0SZ1, K0SZ5, K0SZ6, K0STJF, K0STJ and K0STSF. This can also be seen in the example SN given in figure 3 [20].

Variables that occur simultaneously or consecutive are connect with each other by an edge. This means that for every time a serve occurs in zone 1, the node K0SZ1 is connected to either K0STJF, K0STJ or K0STSF, depending on the serve type that occurs simultaneously. This also means that the tempo of attack of e.g. K2 is connected to the number of blockers in K3 as they follow consecutively [20].

Analysis is done by calculating the centrality of every node using Eigenvector Centrality [20]. Centrality is a measure of how important a node is within a network [25]. In Eigenvector Centrality this importance is determined not only by the number of neighbouring nodes, but it also takes into account how well these neighbours are connected [25]. This is displayed in figure 3 by the different sizes of nodes.

In the analysis of the volleyball matches, for each variable (number of blockers, tempo of attack, etc.), the possible options are compared. This is done to determine which of these nodes has the highest Eigenvector Centrality and thus occurred most [20]. This shows that for the SN in figure 3 the categories K0STJF (jump-float serve) and K0SZ1 (zone 1) have the highest centrality compared to the other serve types and serve zones [20]. This means that K0 is performed most often in zone 1 and the jump-float serve is the most used serve-type.

Next to creating an SN as graph, there might be different ways to use GT for modelling interaction in volleybal. Instead of defining nodes as described above, nodes can also be labelled based on actions performed. The edges then show the likeliness to go from one action to another in terms of relative occurrence. Such a graph is known as a First Order Markov Chain model showing what transitions between actions happen most often [9]. Spencer Best [9] used a Markov Chain model in which he displayed the transitions of actions between two teams. Using this model he could predict how likely a team is to score a point at the end of a rally depending on the action performed. An example of a Markov Chain Model in which actions are represented by the nodes is shown in figure 4(a).



(a) Example of a Markov Chain Model shown for K1. Transition likeliness is shown for consecutive actions (shown as subscript).

(b) Example of an LTS shown for K1. Thickness and colour of the transitions is determined by the absolute number of occurrences per transition.

Figure 4: Examples of both a Markov Chain model as well as an LTS for comparison of the different approaches of GT.

Another way to define the graph is by using the actions as labels for the edges. The nodes then represent different states within a rally. This results in a so-called Labelled Transition System (LTS) [26] in which it is possible to go from one state to another by performing a certain action. All possible states are given by S, all possible transitions are given by $\rightarrow$ and all possible labels of actions are given in Λ. Using information about complexes, it is possible to set up information of all states, all transitions and all labels. The transitions $\rightarrow$ can be noted down as (p, a, q) or $p \xrightarrow{a} q$ with $p \in S$, $q \in S$ and $a \in \Lambda$ [26].

Next to labelling a transition with only an action, an extra edge label can be added representing the weight of the transition. This weight shows the occurrence of the transition and can be updated based on performed actions. An example of an LTS in which actions are used as labels of the transitions is shown in figure 4(b). Labelled Transition Systems are seen in process theory [26], but not yet in combination with the process of a volleyball match or volleyball training.

To summarize, there are three options to use as model for interaction, all follow from graph theoretical approaches. Each option has both advantages as disadvantages related to using that specific approach in the SSE project. This is of importance as the input of the interaction layer consists of the individual actions performed by players as follows from the layered approach in figure 1.

The first option is the SN using Eigenvector Centrality as analysis method. This approach gives an extensive analysis with a lot of information on the rally, like the number of blockers available, attack tempo and serve type. Unfortunately, this information does not follow automatically from the information available in the SSE project. Next to that, SNA has already been used to analyse volleyball matches, so it is not a new approach in volleyball.

The second option is the Markov Chain model in which transitions show the likeliness of one action following a next. This information can follow directly from the SSE project as individual actions are used as input which is definitely an advantage. However, as with SNA, this is not a new approach within volleyball.

The third option is an LTS with nodes representing different states of volleyball with actions to transit from one state to the next. As with the Markov Chain model, the information needed consists of the individual actions performed, which follows from the layered approach of the SSE project. Using an LTS is a new approach in volleyball, which makes it interesting to see if this approach also works.

3 Methods

The advantages and disadvantages of the three types of models that can be used are outlined at the end of section 2.3. For the remainder of this thesis, a choice has to be made which of these models is used to represent interaction. As it is already known that SNA and Markov Chain models work in volleyball, this thesis focusses further on LTS models to see if this approach of GT also works in volleyball.

Now that it is clear LTS models can be used to represent volleyball interaction, this chapter focusses on the third and fourth sub-questions: *How can a model be created that represents the structural patterns in actual volleyball interactions* and *How can individual actions that occur in a volleyball session be used to update information presented by the model?* From these questions, it also follows what output can be shown to a coach as part of the SSE project.

Figure 5 gives an overview of the different steps to take before output can be shown to a coach. First of all, data is needed from a volleyball session to generate output. The volleyball session is recorded on video (details in section 3.1.1) and is annotated by hand to make sure all actions are labelled correctly (details in section 3.1.2).

On the left side of figure 5, it can be seen that there are two ways to create an LTS, either by design or data-driven. When a model is created by design, the model is constructed based on the rules of volleyball and the information about complexes given in section 2.1. In this way, the model contains all possible transition, but the model has to be created before it can be used. This is addressed in section 3.2.1. When a model is created data-driven, the model is build 'on-the-go,' based on actions performed by players. Using this approach, it does not matter

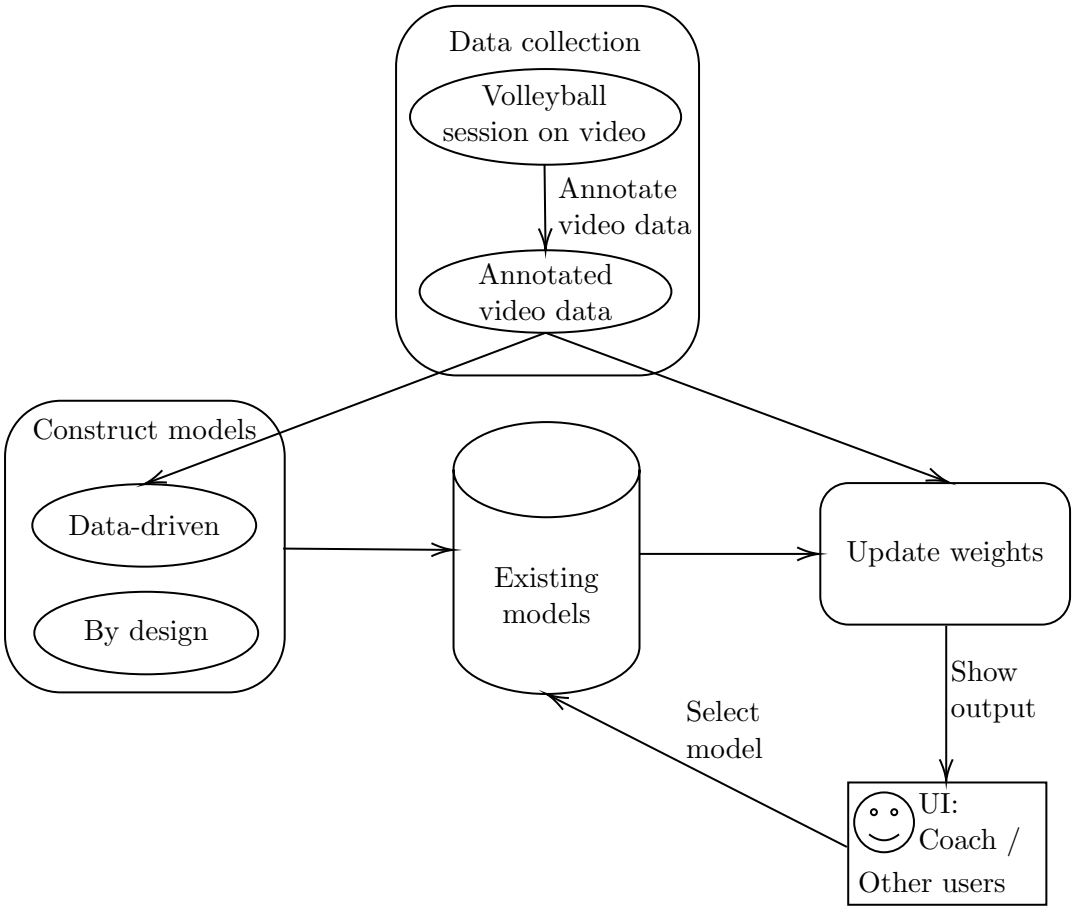


Figure 5: Overview of the different parts of the methods. Constructed models are assembled and the coach can select a model he wants to use to update the weights. Data is needed for both the data-driven construction of models as well as the updating of the weights.

that it is unclear what happens beforehand, because the model includes all transitions anyway. Section 3.2.2 focusses on how transitions can be added when creating a model data-driven.

All constructed models are assembled from which the coach can select any of the models he wants to use. The original model remains in the collection for a possible next time the coach wants to use the same model. Once the model is selected, the actions performed by players are used to traverse the model and update the weights of the transitions that belong to these performed actions (details in section 3.3). This updated model can be shown to the coach to show what sequences of actions have been performed by the team and he can use this information to adjust training sessions. The output displayed to the coach is addressed in chapter 4 by showing the results of different measurement sessions performed.

In figure 5, it is shown that a volleyball session is recorded on video to annotate the actions performed by hand. To save time, an action recognizer as described in section 2.2 can be used to classify actions performed based on sensor data and replace the annotation by hand. However, there can be errors in the action classification and the fifth sub-question focusses on this: *What is the influence of recognition errors when automatic action classification is used to update the model for interaction?* In section 3.4, it is described how these errors can be simulated and studied.

3.1 Data collection

This section focusses on data collection to have annotated data that can be used to construct the data-driven model or update the weights as shown in figure 5. This section is divided into two subsections. First, volleyball sessions are recorded on video during a measurement session. After that, actions performed by players during this session have to be annotated to get the annotated data.

3.1.1 Measurement session

To be able to collect data in a structured way, a protocol is set up. This protocol includes information for the researchers to make sure no details are forgotten as well as exercises that are performed by the players, so they know what is expected of them. The measurement sessions are not only used to record video data, but players are also asked to wear IMU sensors. The IMU data can be used for the recognition of individual actions as shown in the layered approach of figure 1.

The protocol consists of several steps. Before the measurement can start, some preparations have to be done. These consist of letting players sign consent forms and giving all players IMU sensors. Once the video recording has started, players first have to do a calibration procedure for the IMU sensors after which so-called *Drills for reference* are performed. These drills consist of an exercise per complex, performed at least ten times, to have data for each complex. Once these exercises are completed, players are asked to play a friendly game in which all complexes can occur again. The detailed protocol is given in appendix A.

The ethical committee of the faculty of Electrical Engineering, Mathematics and Computer Science of the University of Twente gave approval to perform several measurement sessions within the SSE project. The information brochure for volleyball players and the informed consent they had to sign are given in appendix B.

Using the described protocol, two measurement sessions were done, both with ladies teams. These two sessions are referred to as session 1 and session 2. Each session had a different realization of the protocol, mainly due to the fact that all coaches are unique. Each coach might have a different interpretation of the protocol or adjust exercises to make them alike to exercises the players are used to. For each session, adjustments made by the trainers are given:

- **Session 1**
  - There were nine players (instead of preferably twelve) and as the trainer wanted everyone to participate, some of the exercises were not done with six players, but with three, four or five players, depending on how it worked out best.
  - The exercises as described in *Drills for reference* are almost fully played-out with the following exceptions:
    * K0 and K1 (ses1_K0K1) are trained in smaller groups of three taking turns serving, receiving, setting and attacking. The field is divided in half so two groups can perform the exercise simultaneously. Figure 6 shows how the actions are divided over the players.
    * K2 and K3 (ses1_K2K3) are trained by playing four against five. One team always served and so K0 and K1 are also included in this part.
    * K4 (ses1_K4) is played by having a team of six receive a serve and create an attack (so perform K1). Three players are positioned behind the net to throw a ball over the net simulating a block. This means the team has to perform K4 immediately after their attack. It depends on where the team attacks which of the three players simulates the blocks. Figure 6 shows the positions of players.
    * K5 (ses1_K5) is played by having three players positioned behind the net as shown in figure 6. One by one, the three players behind the net throw in a ball to simulate a FreeBall.
  - The friendly game (ses1_game) was played in a three-against-three-against-three setting. After each rally, the team that lost the point had to get out and the team getting in had to take turn in serving.

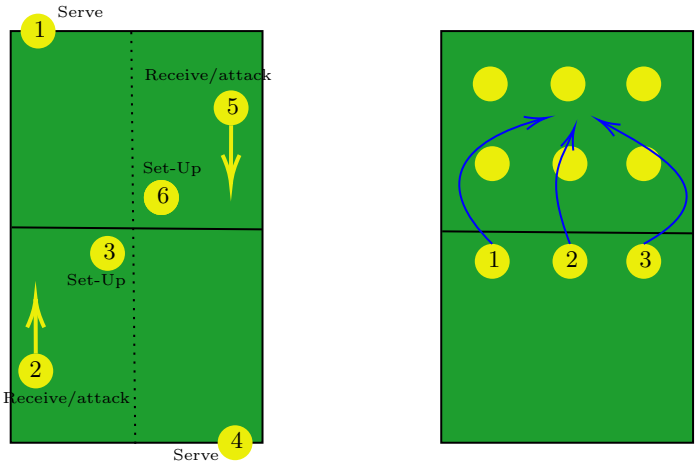


Figure 6: Schematic overview of how the exercises were performed in session 1. On the left, the exercise for K0 and K1 is shown, whereas on the right, the positions of players in the exercise for K4 and K5 is shown.

- **Session 2**
  - There were twelve players attending this training session, so they were able to form two teams and play six-against-six (team A and team B).
  - The trainer did not want to follow the protocol as described, he had his own way of performing certain complexes. Instead of performing the exercises as given by the protocol, he used three alternating options (ses2_exercise): 1) Team B has to serve, so Team A can practice K1, 2) Team A gets a free ball to train K5, 3) Team B gets a

free ball to train K5. After the alternating starts, the teams had to play a rally until one of them scored a point.

- The trainer did not like the exercise given in the protocol for K4. He does not train K4 with a drill, he just tells players to be ready if this complex arises during training and so also during a match.
- After the three alternating exercises, the second part of the training consisted of a friendly game between the two teams (ses2_game). The team that scored a point was allowed to take the next serve.

3.1.2 Annotation

To use the data from the measurement sessions to construct models or update weights, the video recording has to be annotated with actions performed by each individual player. For this, the annotation tool ELAN 5.6-FX is used. In ELAN, it is possible to include a so-called tier for every player in which all actions are annotated over time. The annotation is done by different annotators to spread the amount of work. Once an annotator has started labelling a specific player, the annotator has to finish annotating that player. In that way, only between players differences can occur in annotations, but not within an individual player.

To make sure all annotators are on the same page while annotating, a protocol is set up for this. In this protocol, rules are noted down about when an action starts and when it ends and what possible labels can be used for specific actions. In total, there are thirteen possible action labels that can be used to annotate actions. The detailed annotation protocol is given in appendix C.

To keep the model that is shown as output clear and structured, the number of possible actions that can be performed is kept low. Therefore, some adjustments are made to the possible labels to annotate. This is done after all annotation was completed, but before the labels are used in any of the models. Of the original labels, OneHandTouch, LeftHandPass, RightHandPass and OneHandPass are combined into OneHandPass as they are all actions of the same type, namely a pass performed with one hand. TipOverNet and TipBall are combined into TipBall as these actions are also of the same type. Last, the action TryBlock, in which players try to block, but fail to touch the ball, is removed from the possible labels in the model as this action does not change the current state of a rally. This leads to the following actions that can be used as transition labels in any of the models:

- Block (B)
- ForearmPass (FP)
- OneHandPass (1HP)
- OverHeadPass (OHP)
- Serve
- Smash
- TipBall (TB)
- UnderHandServe (UHS)

For some of the exercises, players were divided into smaller groups. It is important to know what players belong together within an exercise. For this, an extra tier in ELAN is added. This tier contains annotations with information on which players perform exercises together. By extracting this information, it is known which action labels form a sequence and which action labels belong to another sequence.

The same holds for exercises with which the coach/other players influence the exercise, by e.g. throwing in a FreeBall. For this, also an extra tier in ELAN is added. This contains annotations with actions of the coach (or players) which influenced the rally. The annotation protocol in appendix C also describes how starting- and ending points are defined for these labels.

3.2 Construct models

This section focusses on how models can be created. As shown in figure 5, models can be constructed in two ways, either by design or data-driven. Both approaches are described below.

3.2.1 By design

The first way to create a model is by design. This means all states S are known as well as all transitions $\rightarrow$ with the labels Λ. It depends on the application of the model what states and transitions are added. Rally models can be created to keep track of what sequences of actions are performed based on the information of complexes. Alternatively, exercise models can be created that might be linked to these complexes, but are different from the rally model.

For an LTS that can be used as rally model, states follow from the complexes as described by Hileno and Buscá [15]. Transitions follow from the rules of volleyball to decide which transitions are and are not allowed. Figure 2(a) shows the complexes and using this figure, the names of the states are determined.

The first state is part of K0 and is 'StartRally.' This is the starting state of the graph and this state can only be left with a serve action. With a serve, either a point is scored by one of the teams, or the rally continues. In the latter case, the next state is 'NeutralServe,' being the start of K1. This state is named Neutral as no point was scored, meaning the team was neither successful nor unsuccessful.

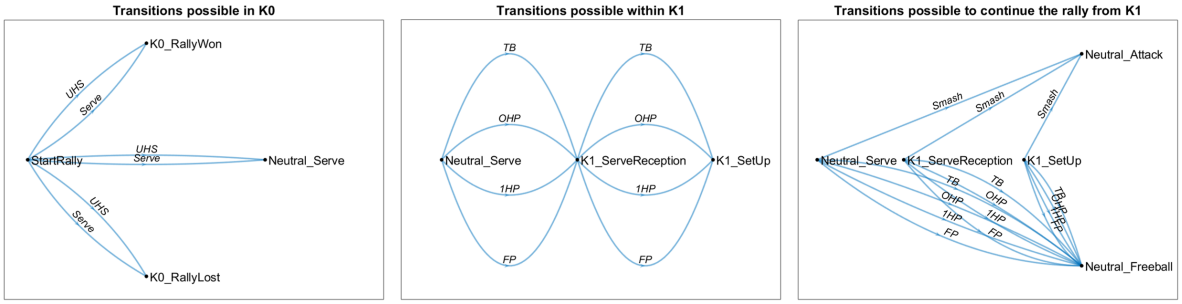
For the complexes K1 to K5, the name of the states follow from figure 2(a). All complexes start with a neutral performance of the action given on the left of this figure (NeutralServe, NeutralAttack, etc.) followed by a defensive state (ServeReception, AttackDefence, etc). After the defensive state, the attack is prepared in the SetUp state. The SetUp state is either followed by scoring/losing a point or, in case the attack is neutral, by the start of the next complex.

This leads to a total of 28 states. Between these different states, different transitions are possible. From the annotated labels described in section 3.1.2, there are eight possible actions to transit between different states. Based on the rules of volleyball, it is known that certain actions can only be performed between specific states. For example, a serving action (Serve and UnderHandServe) can only be performed starting from 'StartRally.' This is shown in figure 7(a). Next to that, a Smash is only performed when going from one complex to the next, as it does not make sense to smash the ball to a team mate to go to e.g. the SetUp state. This also means that actions to go from one state to the next within a complex are the ForearmPass, OneHandPass, OverHeadPass and TipBall. This is shown in figure 7(b).

Preferably, an attack consists of three actions, but it is also possible that a team already attacks after the first action or the defending action is performed in such a way that the ball is immediately passed over the net again. These transitions also have to be included in the model.

The action performed to play the ball over the net influences what the next complex is. If the ball gets over the net with a smash, this means an attack occurred and either 'Neutral_Attack' or 'Neutral_Counterattack' is the next state (depends on the phase of the rally). If the ball gets over the net with one of the other actions (ForearmPass, OneHandPass, OverHeadPass or TipBall), this is an easier ball to defend and thus 'Neutral_Freeball' is the next state. If a Block is used to defend an attack and the ball is sent back over the net again, the next state is 'Neutral_Block', the start of K4. In K1, it is not possible to perform a block, so only 'Neutral_Attack' or 'Neutral_Freeball' can be next states when the rally continues. This is shown in figure 7(c).

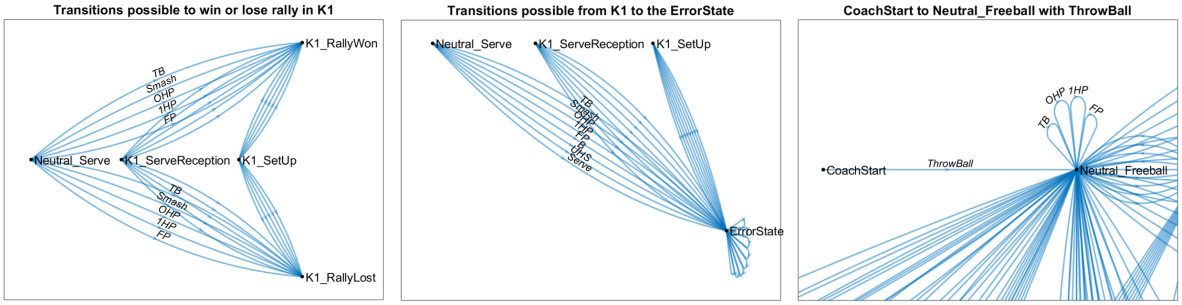
It can also be that the action performed results in the end of the rally. This means that the team who played the ball either goes to RallyWon or RallyLost. Figure 7(d) shows that it is possible to win or lose the rally from any of the states with all actions possible in that state. In some cases, it does not follow from the data whether a team won or lost the rally. Therefore, it is also possible to change RallyWon and RallyLost to RallyEnd (per complex one ending state



(a) Snippet showing all transitions in K0.

(b) Snippet showing all transitions within K1.

(c) Snippet showing all transitions to finish K1.



(d) Snippit showing all transitions to end the rally in K1.

(e) Snippet showing all transitions from K1 to the ErrorState.

(f) Snippet showing a transition from 'CoachStart' to 'Neutral_Freeball.'

Figure 7: In the subfigures, snippits of the full graph are included to show important parts of the graph. The snippits showing transitions to next states within a complex or transitions to the next complex are focussed on K1, however, these transitions can also be made in the other complexes. For the overview, only these snippits are shown.

instead of two). The states RallyWon, RallyLost or RallyEnd are final states and there are no outgoing transitions from these states.

Considering these states and set of possible actions as transitions, this leads to a total of 304 transitions between the 28 identified states. An overview of all states and transitions as a list is given in appendix D.

As actions can be recognized incorrectly by the action recognizer [6], a transition can be recognized that is not possible with the given LTS. As an example, an OverHeadPass used as SetUp can be recognized incorrectly as a Smash. This means that the transition made in the volleyball match is recognized incorrectly and cannot be represented by the model as the transition is not part of the LTS.

To address this in the model, the ErrorState is added. Next to that, transitions to the ErrorState are included. These transitions start in every possible state with outgoing transitions and are labelled with all possible actions. Once the transition to the ErrorState is made, next transitions loop from the ErrorState to the ErrorState, unless a Serve or UnderHandServe is performed. With these two actions, the rally starts at 'StartRally' again. Figure 7(e) shows the transitions to the ErrorState for the states of K1.

The reason transitions labelled with all possible actions are added is to be sure it is always possible to make a transition. There might be some states (e.g. K1_SetUp) in which it is already always possible to perform a certain action, e.g. ForearmPass. However, still a transition from K1_SetUp to the ErrorState is included labelled ForearmPass to capture the case in which it is not possible (even though it seems like it will never occur). Next to that, the alternative is to go over all states, check all outgoing transitions and add transitions to the ErrorState in case an action is not possible in a specific state. In that approach of adding the ErrorState, it might be that transitions are missed.

The model described above can be used as a rally model. It is also possible to create an exercise model by design. This exercise model can be related to the complexes or not at all. In case the model is related to the complexes, the steps described above can be used to include the complexes. Extra states or transitions can be included that make the model an exercise model. For example, if the coach throws in the ball to start K5, an extra starting state 'CoachStart' can be included with a transition labelled 'ThrowBall' to 'Neutral_Freeball.' This is shown in figure 7(f).

3.2.2 Data-driven

The other option is to create a model data-driven. This means that the states S are unknown beforehand and can only be created when annotated data is given as input. From the annotated data, transition labels between the different states can be extracted to know the transitions $\rightarrow$ with their labels Λ.

This means the model is created 'on-the-go' while players perform actions repeating an exercise. Two nodes are always included, 'State1' which is the starting node of the exercise and 'EndExercise' which is the ending node without outgoing transitions. As there is no information on the name of the states, next states are named 'State2,' 'State3,' etc. Based on actions performed, transitions are included between these states. The only information that is necessary to know beforehand is the action that is used as first action in an exercise. This can e.g. be a Serve if it is an exercise to practise K0 and K1.

Below, a stepwise overview of the algorithm to create a data-driven model is given. The input needed for this algorithm consists of a data array over time showing the starting time of each action. Appendix E describes how this data array can be conceived from either the annotated data or the recognized actions.

1. Start in 'State1' and determine the current and next action performed. Set the state counter i to 2.
2. If the next action is not the starting action, repeat the following steps:
   (a) Create a new state 'State i,' unless this state is already present.
   (b) Add a transition between the current state and 'State i' labelled with the current action, unless this transition was added previously.
   (c) Update the current state to 'State i.'
   (d) Update the state counter i to i+1.
3. If the next action is the starting action, add a transition between the current state and ending state 'EndExercise,' labelled with the current action. Update the current state to 'State1' and reset the state counter i to 2. Repeat step 2.
4. Once no more data is available, the ErrorState is included, with transitions to this state.
5. The model with ErrorState is added to the collection of existing models.

For some steps in this algorithm, more elaboration is needed to understand why it is an important (part of a) step. In step 1, it is mentioned that both the current and next action are needed to update the state and add a transition. The current state is needed as label of the transition, whereas the next action is needed to determine whether to update to the next state or the EndExercise node.

Step 4 describes that the ErrorState is added once no more data is available. This can be due to the fact that the coach decides that the model is ready or that a pre-recorded session is used and the end of the session is reached. The ErrorState is included with transitions to this state and this is done in the same way and with the same reasoning as with the model created by design.

With the algorithm as described above, it follows that transitions are only made to consecutive states ('State3' to 'State4') or to the ending state 'EndExercise.' This implicates that it is not possible to make a transition from 'State2' to 'State4.' Therefore, constructing a model data-driven for a rally does not give a useful model to create output for a coach. Constructing a model by design has the possibility to create both an exercise model as well as a rally model.

3.3 Update weights

Once several models are created, they are collected in 'Existing models' as shown in figure 5. For a specific volleyball session, the coach can select any of the models to keep track of the interaction during that session. This is based on the actions performed by all players and done by updating the weights of the transitions in that specific model. These weights are extra labels of each edge, next to the action label. They consist of a general counter for total occurrence of the transition and a counter per player to keep track of how often a certain player performs a specific transition.

The input for the update weight algorithm is a data array over time with the starting point of each action marked. A same data array is included with the player numbers of the player who performed the given action. Appendix E describes how these data arrays can be conceived from either the annotated data or the recognized actions. The graph is traversed using the data arrays with a loop over time, going over all possible actions, performing the following steps:

1. Start in the starting state (e.g. State1, StartRally or StartCoach) and determine the current and next action performed and the time between those actions.
2. The next action determines which state is reached and the current action determines what transition label has to be updated.
    - If the next action is performed by a player of the same team, the next state within the complex is reached.
    - If the next action is performed by a player of the opponents, the next complex is reached, depending on the current action (Smash to start K2/K3, Block to start K4, Freeball action to start K5).
3. Check if the transition between the two states with the given action label is present in the graph (leaving out the ErrorState).
    - If this is the case, the transition label is updated by adding 1 to the general counter and 1 to the specific player counter
    - If this is not the case, the next state is the ErrorState and this transition to the ErrorState is updated by adding 1 to the general counter and 1 to the specific player counter. This transition can always be updated by how the ErrorState is defined.
4. Steps 2 and 3 are repeated unless, depending on the model:
    - If in ErrorState, the state is reset to the starting state if the starting action occurs. The starting state is specific for a model and depends on the starting action.
    - If the time between two consecutive actions is more than 5 sec, the next state is an ending state of the model. After the transition to the ending state is updated, the current state is reset to the starting state. From there on, steps 2-3 are repeated.

In the step 4, it is mentioned that if there is more than 5 sec between two consecutive actions, the rally has ended and the current state is updated to the starting state. This definition of when a rally ends is chosen, because if this depends on the starting action, it can be that the rally ends at moments that it is clear the rally did not end. For example, if a Smash is recognized as a Serve, the model should update a transition to the ErrorState instead of starting at StartRally again. The value of 5 sec is an assumption made as it does not make sense to perform a consecutive action within a rally after more than 5 sec.

Once the loop over time is done, all performed actions are used to update one of the labels. Using the information of the general counter, the LTS can be plotted to show as output to the coach. The value of this counter is used to show the absolute occurrence of the transition by thickness and colour. In that way, it is possible to compare transitions to each other and determine the most used sequence of actions in an exercise.

Another way to show some output is by using the specific player counter to compare players in a table. In the results chapter, for different types of models the coach can select, the output shown to the coach is presented. This is done based on the different exercises performed during the measurement sessions.

3.4 Automatic action classification

In the above sections, more information has been given on the different building blocks in the overview given in figure 5. These sections focussed on the third and fourth sub-questions: *How can a model be created that represents the structural patterns in actual volleyball interactions?* and *How can individual actions that occur in a volleyball session be used to update information presented by the model?* However, a different approach is needed to be able to answer the fifth sub-question: *What is the influence of recognition errors when automatic action classification is used to update the model for interaction?*

Instead of annotating the actions by hand, a classification algorithm can be used to recognize actions. However, such an algorithm can make mistakes in classification and this can influence the output shown to a coach in two ways. First, as the annotated data is used to create a data-driven model, the model a coach selects looks different than it would have if the the actions were annotated by hand. The second option where incorrect recognized actions can influence the output is while updating the weights of the transition labels. In this section, a closer look is taken into the latter possibility.

In case an action is recognized as another action, this can result in two things. The first is that the action still makes sense within the rules of volleyball and is thus represented by the model. This is the case if e.g. a Smash is recognized as a ForearmPass. The transition to a next state is still possible with a ForearmPass, but the transition of which the weight is updated is a different transition. The model can still keep track of the states and complexes, but there might be an error compared to what players perform in their rally. The second option is that the action does not make sense any more. This is the case if e.g. a Smash is recognized as a Serve. Within a rally, it is not possible to suddenly perform a Serve. In this case, the model thus updates a transition to the ErrorState as the transition with Serve cannot be made in the model without the ErrorState.

To demonstrate what happens to the output shown to the coach if actions are incorrectly recognized, the action recognition is simulated using different confusion matrices. Using the simulated recognized actions, a rally model created by design is updated to compare the weights of transitions to the original weights found in an error free model. This error free model is updated using actions annotated by hand.

Per confusion matrix, 100 randomization are performed to find an average weight for each transition label. Due to the fact that 100 randomizations are performed, the assumption of normal distribution of the mean follows from the central limit theorem [27]. For each transition, the mean and standard deviation of the adjusted model can be compared to the value found in the error free model. This comparison can be done using a Z-test with a significance level of $\alpha = 5\%$ [27]. Depending on whether the amount of transitions increased or decreased, either of the below probabilities can be calculated to find if the transition occurred significantly more or less:

$$P(X > x_{\text{true}}) = P(Z > \frac{x_{\text{true}} - \mu}{\sigma/\sqrt{100}}) < \alpha$$
$$P(X < x_{\text{true}}) = P(Z < \frac{x_{\text{true}} - \mu}{\sigma/\sqrt{100}}) < \alpha$$

As a rally model is used to demonstrate the influence of the action recognizer, there are three types of transitions that can change significantly and for each type of transition that changes significantly, there are different consequences for the output shown to the coach:

- A transition between two states linked to complexes can change significantly, for example, the number of ForearmPasses to receive a Serve decreases while the number of OverHeadPasses used increases. This does not influence the model dramatically, because while updating the weights, the current state is not updated to the ErrorState. The coach should keep in mind that the values of the weights might not be the exact occurrence of the transitions.
- A transition between a state linked to complexes and the ErrorState can change significantly. This is a worse error, as it means that the current state of the LTS (the ErrorState) cannot reflect what happens in the match.
- A transition looping in the ErrorState is the worst type of transition that can change significantly. If this transition changes significantly, several transitions labelled the same action occurred while in the ErrorState. This means that probably the transition to the ErrorState was made early in a rally as relatively many transitions still occurred.

As mentioned above, different confusion matrices are used to simulate the classification of actions. Next to using the confusion matrix found in literature with a UAR of 67.87%, it is interesting to see how other pre-defined confusion matrices influence the output of rally model. These results can be used to give a recommendation of how well the action recognizer should perform before the LTS can realistically be used to represent statistics of players.

The confusion matrices used to randomly adjust the action labels are given below. For each confusion matrix, additional information is given on why it is interesting to see the influence that specific confusion matrix. The confusion matrices given all have a recall of 95% for different subsets of actions. When testing the models, these confusion matrices are also created with a recall of 90% and 80%, these are presented in appendix F.

The first confusion matrix used is given in table 1. The confusion matrix has a recall of 95% for all actions and so also an overall accuracy of 95%. The 5% of the times the action is not recognized correctly is divided evenly over all other actions to what actions the label is changed. This matrix gives an overall idea of what the accuracy of the action recognizer should be.

Table 1: Confusion matrix with overall accuracy of 95%.

	B	FP	1HP	OHP	Serve	Smash	TB	UHS
B	95	0.71	0.71	0.71	0.71	0.71	0.71	0.71
FP	0.71	95	0.71	0.71	0.71	0.71	0.71	0.71
1HP	0.71	0.71	95	0.71	0.71	0.71	0.71	0.71
OHP	0.71	0.71	0.71	95	0.71	0.71	0.71	0.71
Serve	0.71	0.71	0.71	0.71	95	0.71	0.71	0.71
Smash	0.71	0.71	0.71	0.71	0.71	95	0.71	0.71
TB	0.71	0.71	0.71	0.71	0.71	0.71	95	0.71
UHS	0.71	0.71	0.71	0.71	0.71	0.71	0.71	95

The next confusion matrices gives more information about the influence of type of actions. In table 2, the four actions that get confused are the so-called FreeBall actions (ForearmPass, OneHandPass, OverHeadPass and TipBall). In case the ball is not played over the net, these actions cause a continuation within a complex, whereas in case the ball is played over the net with one of these actions, the opponent receives a FreeBall and starts in K5. If any of these actions is recognized as another Freeball action, it is expected that the model is still able to keep track of the states.

The other confusion matrix is given in table 3 and confuses the non-FreeBall actions (Block, Serve, Smash and UnderHandServe). In contrast to the reasoning above, these actions cannot happen in all cases (e.g. a Serve can only occur in the 'StartRally' state) and confusing these actions might cause a transition to the ErrorState.

Table 2: Confusion matrix which mixes up the FreeBall actions with a recall of 95%.

	B	FP	1HP	OHP	Serve	Smash	TB	UHS
B	1	0	0	0	0	0	0	0
FP	0	95	1.67	1.67	0	0	1.67	0
1HP	0	1.67	95	1.67	0	0	1.67	0
OHP	0	1.67	1.67	95	0	0	1.67	0
Serve	0	0	0	0	1	0	0	0
Smash	0	0	0	0	0	1	0	0
TB	0	1.67	1.67	1.67	0	0	95	0
UHS	0	0	0	0	0	0	0	1

Table 3: Confusion matrix which mixes up the non-FreeBall actions with a recall of 95%.

	B	FP	1HP	OHP	Serve	Smash	TB	UHS
B	95	0	0	0	1.67	1.67	0	1.67
FP	0	1	0	0	0	0	0	0
1HP	0	0	1	0	0	0	0	0
OHV	0	0	0	1	0	0	0	0
Serve	1.67	0	0	0	95	1.67	0	1.67
Smash	1.67	0	0	0	1.67	95	0	1.67
TB	0	0	0	0	0	0	1	0
UHS	1.67	0	0	0	1.67	1.67	0	95

In these two confusion matrices, it is assumed that a defined subset of actions is confused, whereas all other actions are recognized perfectly. As this is not a realistic representation of an action recognizer, the two confusion matrices are combined into one in which the specific subsets are confused at the same time. This results in the confusion matrix in table 4 which is a combination of the confusion matrices in tables 2 and 3.

Table 4: Confusion matrix which confuses both the Freeball actions as well as the non-FreeBall actions, both with a recall of 95%.

	B	FP	1HP	OHP	Serve	Smash	TB	UHS
B	95	0	0	0	1.67	1.67	0	1.67
FP	0	95	1.67	1.67	0	0	1.67	0
1HP	0	1.67	95	1.67	0	0	1.67	0
OHP	0	1.67	1.67	95	0	0	1.67	0
Serve	1.67	0	0	0	95	1.67	0	1.67
Smash	1.67	0	0	0	1.67	95	0	1.67
TB	0	1.67	1.67	1.67	0	0	95	0
UHS	1.67	0	0	0	1.67	1.67	0	95

Another way to create a confusion matrix is by looking at the nature of the actions performed. Some of the actions are performed with arms low (ForearmPass, OneHandPass and UnderHandServe), whereas other actions are performed with the arms high (Block, OverHeadPass, Serve, Smash and TipBall). As sensor data is used to recognize actions, these actions might get confused easily. The confusion matrices that confuse these actions are shown in tables 5 and 6. Again, these confusion matrices can be combined into one confusion matrix that confuses both subsets of actions. This confusion matrix is shown in table 7.

Table 5: Confusion matrix which confuses actions performed with arms down with a recall of 95%.

	B	FP	1HP	OHP	Serve	Smash	TB	UHS
B	1	0	0	0	0	0	0	0
FP	0	95	2.5	0	0	0	0	2.5
1HP	0	2.5	95	0	0	0	0	2.5
OHP	0	0	0	1	0	0	0	0
Serve	0	0	0	0	1	0	0	0
Smash	0	0	0	0	0	1	0	0
TB	0	0	0	0	0	0	1	0
UHS	0	2.5	2.5	0	0	0	0	95

Table 6: Confusion matrix which confuses actions performed with arms up with a recall of 95%.

	B	FP	1HP	OHP	Serve	Smash	TB	UHS
B	95	0	0	1.25	1.25	1.25	1.25	0
FP	0	1	0	0	0	0	0	0
1HP	0	0	1	0	0	0	0	0
OHP	1.25	0	0	95	1.25	1.25	1.25	0
Serve	1.25	0	0	1.25	95	1.25	1.25	0
Smash	1.25	0	0	1.25	1.25	95	1.25	0
TB	1.25	0	0	1.25	1.25	1.25	95	0
UHS	0	0	0	0	0	0	0	1

Table 7: Confusion matrix which confuses both actions performed with arms down as well as actions performed with arms up, both with a recall of 95%.

	B	FP	1HP	OHP	Serve	Smash	TB	UHS
B	95	0	0	1.25	1.25	1.25	1.25	0
FP	0	95	2.5	0	0	0	0	2.5
1HP	0	2.5	95	0	0	0	0	2.5
OHP	1.25	0	0	95	1.25	1.25	1.25	0
Serve	1.25	0	0	1.25	95	1.25	1.25	0
Smash	1.25	0	0	1.25	1.25	95	1.25	0
TB	1.25	0	0	1.25	1.25	1.25	95	0
UHS	0	2.5	2.5	0	0	0	0	95

The final confusion matrices that are used for testing are based on the confusion matrix found in literature. Salim et al. [6] describe three confusion matrices in their paper and the best of these can be used to adjust action labels. This confusion matrix is given in table 8. The confusion matrix given is a little adjusted as Salim et al. [6] did not include the TipBall action in their recognition. Therefore, the confusion matrix is extended a little by the adding the action TipBall. As no information on TipBall is available, perfect recognition is assumed for this action.

This confusion matrix has a UAR of 67.87% [6], which is lower than the confusion matrices used for testing (recall of 80%, 90% and 95% for defined actions). To see how this confusion matrix performs with a higher UAR, while keeping the same ratios between the incorrect recognized actions, the best recognized actions in the confusion matrix of table 8 are improved by doubling the values in the diagonal for these actions. This improved confusion matrix is given in table 9.

Table 8: Confusion matrix from [6] with the addition of the perfect recognition of TipBall.

	B	FP	1HP	OHP	Serve	Smash	TB	UHS
B	9	2	2	10	1	0	0	0
FP	14	169	5	14	2	3	0	0
1HP	4	9	6	1	0	6	0	1
OHP	4	7	2	103	2	0	0	0
Serve	1	11	0	2	69	10	0	3
Smash	1	7	2	6	5	88	0	0
TB	0	0	0	0	0	0	1	0
UHS	3	0	0	0	1	0	0	61

Table 9: Confusion matrix from [6] with improvement of recognition of well recognized actions.

	B	FP	1HP	OHP	Serve	Smash	TB	UHS
B	9	2	2	10	1	0	0	0
FP	14	338	5	14	2	3	0	0
1HP	4	9	6	1	0	6	0	1
OHP	4	7	2	206	2	0	0	0
Serve	1	11	0	2	138	10	0	3
Smash	1	7	2	6	5	176	0	0
TB	0	0	0	0	0	0	1	0
UHS	3	0	0	0	1	0	0	122

3.5 Possible models

As shown in figure 5, the coach has a UI to select any of the possible models and this interface also shows output based on the updated weights. In section 3.1.1, the different exercises and rallies (ses1_K0K1, ses1_K2K3, etc) performed during the measurement sessions are described. As the current collection of models is still empty, for each of these exercises and rallies, first a model has to be created. For the rally model, this is preferably done by design (section 3.2.1), whereas for the exercises it depends on the exercise if the model is created by design or data-driven (section 3.2.2). Once these models are part of the possible models, any of them can be selected to update the weights based on the actions performed during the exercises (section 3.3).

In the results chapter, graphs presented that can be shown to the coach as output. These are divided into models created by design and models created data-driven. For each of these models, it is then described what information a coach can derive from the graph. Next to that, the results of the simulations done with the confusion matrices are shown. These are based on the actions performed in ses2_game. In this session, all players always used Serve and never UnderHandServe as serving action. Therefore, compared to the given confusion matrices, UnderHandServe is excluded from the analysis.

4 Results

In the methods chapter, the third and fourth sub-questions (*How can a model be created that represents the structural patterns in actual volleyball interactions?* and *How can individual actions that occur in a volleyball session be used to update information presented by the model?*) are addressed by describing how models can be created and how the weights of the transitions can be updated. To further elaborate on that, this chapter shows the implementation of the algorithms using data of the measurement sessions. Next to that, this chapter also shows the results of the carried out simulations to focus further on the fifth sub-question in the discussion chapter: *What is the influence of recognition errors when automatics action classification is used to update the model for interaction?*

4.1 Possible models

As described in section 3.5, the coach can select different models based on the exercise performed. In this section, for these different models, either created by design or created data-driven, the updated LTS is shown. This relates directly to both the third as well as the fourth sub-question as both the construction of models as well as the algorithm to update weights are addressed.

4.1.1 By design

In the measurement sessions, five exercises are performed that can be used to update weights of a model created by design. As it is possible to create both a rally model by design as well as exercise models, this subsection is divided into these different created models.

Rally model

The first type of model that can be constructed by design is a rally model in which all complexes are represented. This model can be used in for either a rally during a training session or the rally of a match. Exercises ses1_K2K3, ses1_game and ses2_game are performed in a rally-like situation during a training session and for these exercises, the rally model can be selected by the coach. The results of the latter exercise are used in section 4.2 in the simulation of the automatic action classification. For the other two exercises, the graphs with updated weights are shown here.

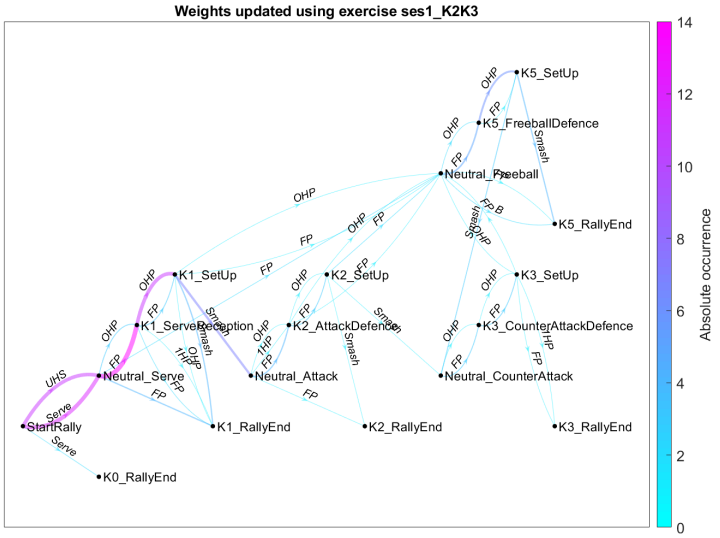


Figure 8: Rally model updated with actions performed during exercise ses1_K2K3. It can be seen that the most common path starts in 'StartRally' with either Serve or UnderHandServe, followed by a ForearmPass, OverHeadPass and a Smash in K1 to go to Neutral_Attack of K2.

In figure 8, the output LTS of ses1_K2K3 is shown. The thickness and colour of the of the transitions show the absolute occurrence of the transitions. The rally starts on the left with 'StartRally' as starting state and from there, both Serve and UnderHandServe occur most often to go to Neutral_Serve, starting K1. In K1, the most common path is ForearmPass, OverHeadPass and Smash to start K2. The coach can decide for himself if this is the best path or that he wants to see another sequence of actions.

For exercise ses1_game, the same rally model is selected, but the weights are updated with other actions performed to give another graph as output. Figure 9 shows this graph zoomed in on K0 and K1. The most common path is the same as in figure 8 and it can be seen that a Smash is also performed to go to 'RallyEnd' instead of 'Neutral_Attack.'

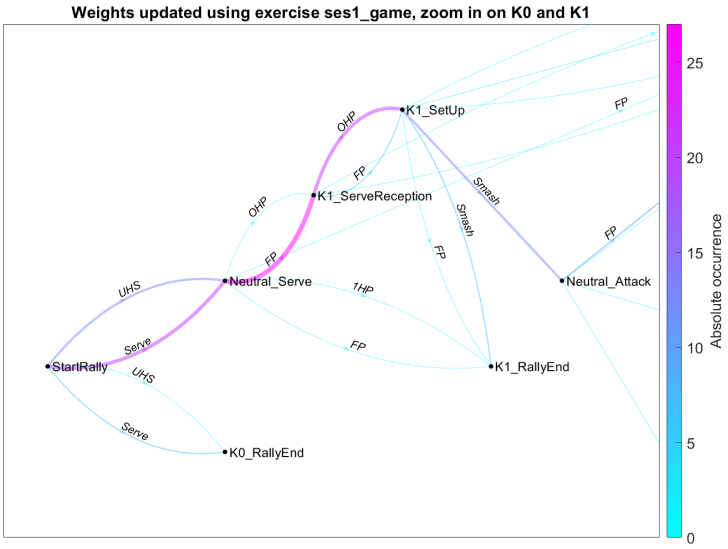


Figure 9: Zoom in on K0 and K1 of the rally model updated with the actions performed during exercise ses1_game. It can be seen that the most common path starts in 'StartRally' with preferably Serve, followed by a ForearmPass, OverHeadPass and a Smash in K1 to go to Neutral_Attack of K2.

Instead of showing a graph as output as in figures 8 and 9, it is also possible to show the values of the counters as output. These counters are part of the transition label and keep track of the total occurrence per transition. Different counters divide the occurrence also over the individual players. For example, in figure 9, K1_SetUp has two outgoing transitions with the label Smash. These transitions can be compared to the other outgoing transitions to see what other actions are used and if the Smash is indeed performed most often as is expected from the graph.

Table 10: Values of the counters used as transition labels for all outgoing transitions of K1_SetUp

StartNode	EndNode	Label	Total count	Team 1			Team 2			Team 3		
				P1	P5	P9	P2	P6	P7	P3	P4	P8
K1_SetUp	K1_RallyEnd	FP	1	0	0	1	0	0	0	0	0	0
K1_SetUp	K1_RallyEnd	Smash	7	4	1	0	0	0	1	0	0	1
K1_SetUp	Neutral_Attack	Smash	13	4	1	0	4	0	2	1	0	1
K1_SetUp	Neutral_FreeBall	FP	1	0	0	0	0	0	0	1	0	0
K1_SetUp	Neutral_FreeBall	OHP	4	1	0	2	0	0	1	0	0	0
K1_SetUp	Neutral_FreeBall	TB	1	1	0	0	0	0	0	0	0	0

In table 10, all outgoing transitions from K1_SetUp are shown with the values of the different counters. The players are sorted per team to see differences between the three teams. This table shows that after SetUp, a Smash is the most performed action. It also shows that a Smash is performed most often to continue the rally with a transition to Neutral_Attack. Further analysis can be done when looking at the count values per team. Team 1 has performed most actions starting in K1_Setup, whereas team 3 has the fewest. Most smashes are performed by P1, however, P2 is more stable in the outcome of her smashes (always to Neutral_Attack).

Exercise models

Instead of creating a rally model by design, it is also possible to create an exercise model by design. This is most useful if the exercise is linked to the complexes. Exercises ses1_K4 and ses2_exercise are examples from the measurement session in which players were asked to perform complexes with influence of the coach.

In figure 10, the output LTS of ses1_K4 is shown. In the model created by design the states of complexes are included as well as a transition to start K4 with 'SimB' (simulated block).

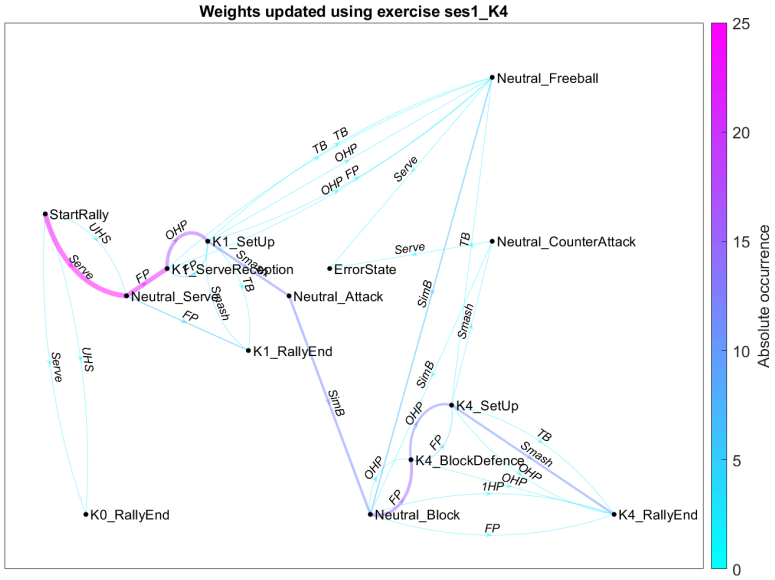


Figure 10: Model created by design including a transition 'SimB' to start K4. The most common path in K4 is ForearmPass, OverHeadPass, Smash and it can be seen that this is also the most common path in K1.

Figure 10 also shows that something goes wrong in the implementation of the update weights algorithm as described in section 3.3. There are two transitions to the ErrorState with the label Serve. This probably means that the algorithm assumed the rally continued while actually a transition had to be made to 'RallyEnd' to start with Serve again.

What else can be seen in figure 10 is that in most cases, K1 is finished with a Smash to go to 'Neutral_Attack' which is then blocked subsequently to go to K4. However, in some cases, the actions performed cause a transition to Neutral_Freeball, from which the block is simulated. In a match-like situation, this would probably not occur as a FreeBall action is not an action that is blocked. However, in the exercise, the block was always simulated, even if no attack was placed.

In figure 11, the output LTS of ses2_exercise is shown. In the model created by design all states of complexes are included as well as the state 'StartCoach' with transition 'Throwball' to start K5. The figure shows a zoom in on K5 and the 'StartCoach' state to show its implementation.

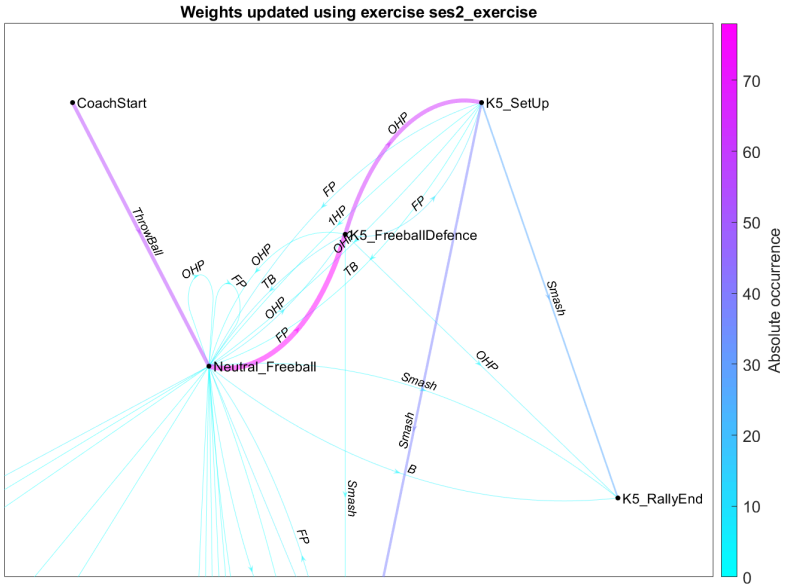


Figure 11: Model created by design including the state 'StartCoach' and transition 'ThrowBall' to start K5 by the coach. The most common path in K5 after ThrowBall is ForearmPass, OverHeadPass followed by a Smash to either continue or end the rally.

As mentioned in section 3.1.1, there are three options to start the rally in ses2_exercise: 1) Serve by team B, 2) 'ThrowBall' to team A or 3) 'ThrowBall' to team B. This means that in the full LTS, which is only partly shown in figure 11, there is another starting node, 'StartRally' next to the 'StartCoach' which is displayed. Depending on the action performed ('ThrowBall' or 'Serve'), it is determined where to start updating the weights of transitions.

The start by the coach is to practise K5 and try to end this complex with a Smash. However, it follows from figure 11 that K5 can also be followed by K5. For example, it is possible to go from K5_SetUp to Neutral_FreeBall with a ForearmPass. This shows that it is not always possible to create an attack, even though the team starts with the 'easy' FreeBall. In table 11, the statistics are shown for the incoming transitions for Neutral_Freeball that started in K5 and all outgoing transitions for K5_SetUp. This shows that relatively few times K5 is followed by K5.

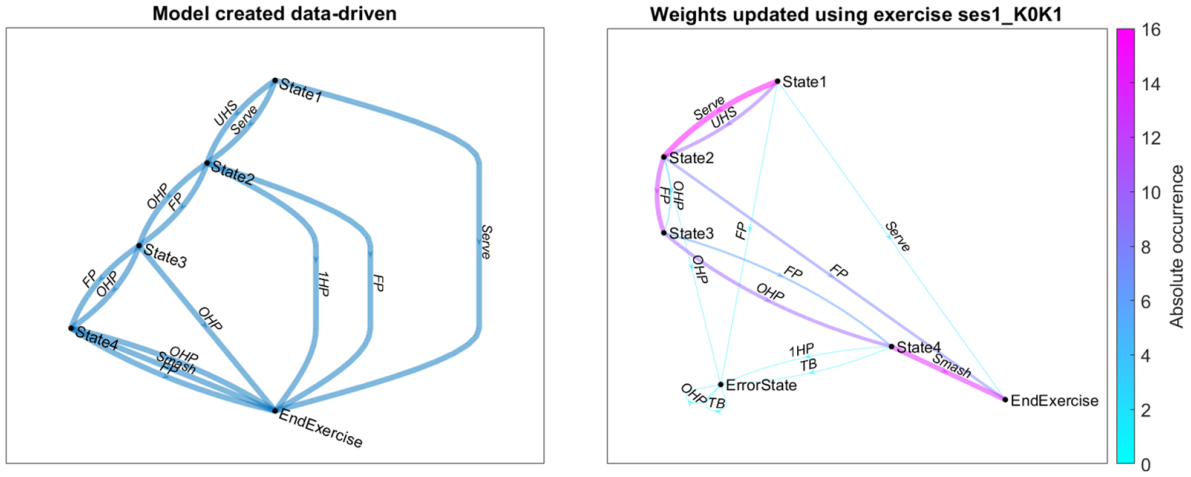
Table 11: Incoming transitions for Neutral_Freeball starting in K5 and all outgoing transitions for K5_SetUp.

StartNode	EndNode	Label	Count
CoachStart	Neutral_FreeBall	ThrowBall	57
Neutral_FreeBall	Neutral_FreeBall	FP	1
Neutral_FreeBall	Neutral_FreeBall	OHP	1
K5_FreeballDefence	Neutral_FreeBall	OHP	1
K5_FreeballDefence	Neutral_FreeBall	TB	5
K5_SetUp	Neutral_FreeBall	FP	4
K5_SetUp	Neutral_FreeBall	1HP	1
K5_SetUp	Neutral_FreeBall	OHP	2
K5_SetUp	Neutral_FreeBall	TB	1
K5_SetUp	K5_RallyEnd	Smash	26
K5_SetUp	Neutral_CounterAttack	Smash	38

4.1.2 Data-driven

In the measurement sessions, two exercises are performed for which a data-driven exercise model is required before the weights can be updated to show any output to a coach. In this subsection, the constructed model is shown with all transitions added while creating the model using the algorithm described in section 3.2.2. Next to that, the same LTS is shown in which the weights are updated after the coach has selected the model, using the algorithm to update the weights described in section 3.3.

In figure 12(a), the model is shown after it is constructed with half of the data of ses1_K0K1. This model is added to the collection of models from which the coach can select the model to update the weights. In figure 12(b), the weights of the model are updated based on the other half of the data of the session.



(a) LTS created data-driven using half of the actions performed in exercise ses1_K0K1. The model can be selected by the coach to update the weights.

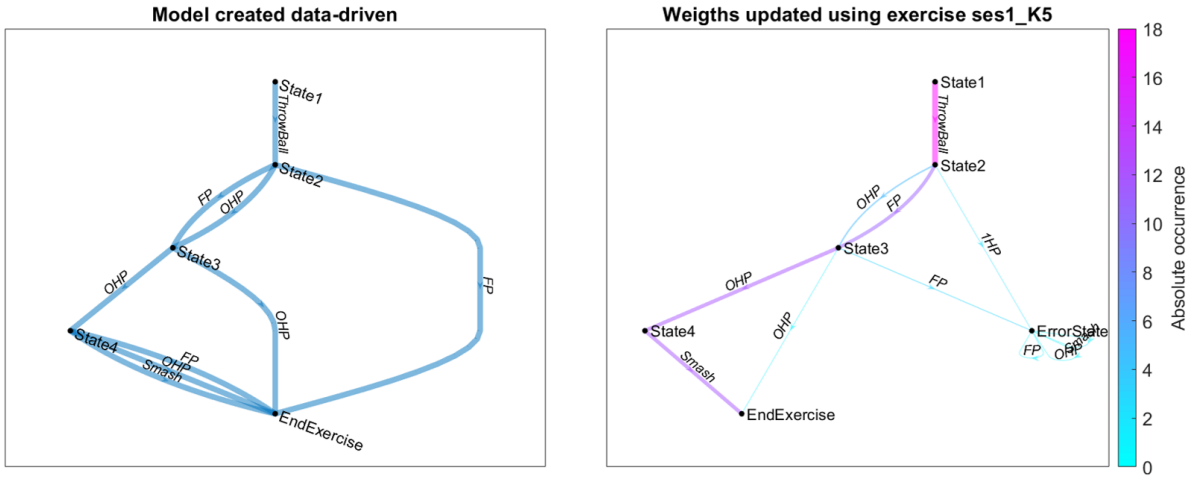
(b) The weights of the model are updated using the other half of actions performed in exercise ses1_K0K1. It can be seen that the most common path for this exercise is Serve, ForearmPass, OverHeadPass and Smash.

Figure 12: LTS of exercise ses1_K0K1. The model is created data-driven first after which the weights are updated.

As can be seen in figure 12(b), there are a few transitions to the ErrorState. In figure 12(a), it can be seen that the transitions from State4 to EndExercise are labelled OverHeadPass, Smash or ForearmPass. Figure 12(b) shows that also TipBall and OneHandPass are performed as final action. As these are not included in the original model, a transition is updated to the ErrorState, whereas these are actually valid actions to end the exercise with.

For ses1_K5, the same figures are shown. In figure 13(a), the model is shown after constructing the model with half of the data of the session and in figure 13(b), the other half of the data of exercise ses1_K5 has been used to update the weights. As in figure 12, there are a few transitions to the ErrorState while these might be valid actions within the exercise.

As with model created by design, per transition label, a counter is used to keep track of the occurrence of transitions. This means that an overview like table 10 can also be shown for these exercises. These tables are easier to interpret as there are not as many states and thus fewer transitions possible to show.



(a) LTS created data-driven using half of the actions performed in exercise ses1_K5. The model can be selected by the coach to update the weights.

(b) The weights of the model are updated using the other half of the actions performed in exercise ses1_K5. It can be seen that the most common path after the ball has been thrown in is ForearmPass, OverHeadPass and Smash.

Figure 13: LTS of exercise ses1_K5. The model is created data-driven first after which the weights are updated.

4.2 Automatic action classification

In this section, the results of the simulations with confusion matrices are shown to focus on the fifth sub-question: *What is the influence of recognition errors when automatic action classification is used to update the model for interaction?* In figure 14, the weights of the rally model as described in section 4.1.1 are updated using the actions performed during ses2_game, creating an error free graph. The actions of ses2_game are also randomly confused as if they were recognized by an action classifier. The graph shown as output using these actions to update the weights is shown in figure 15.

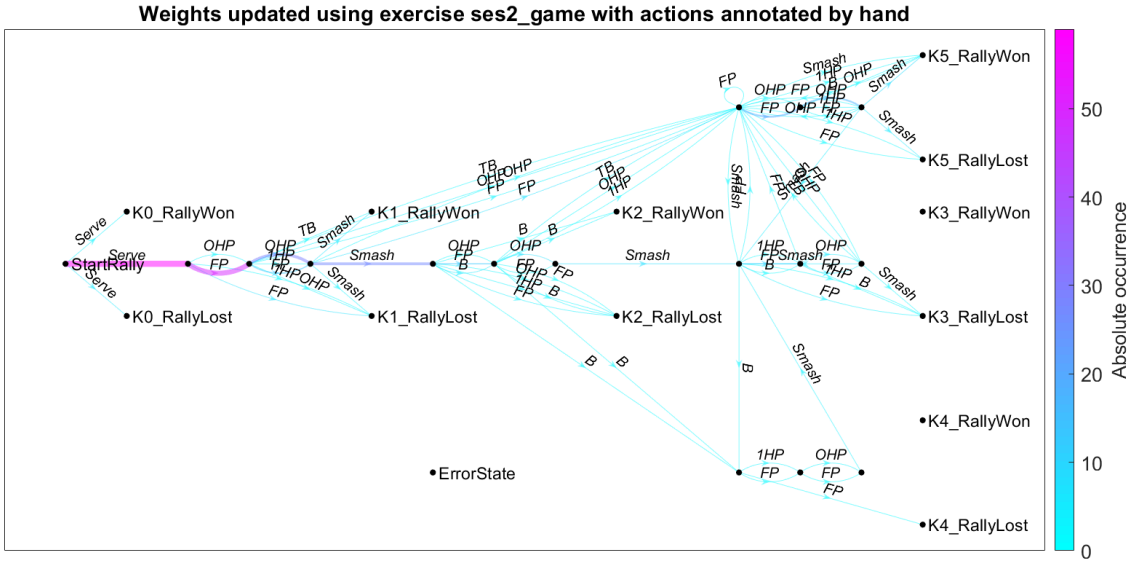


Figure 14: Rally model updated with annotated actions performed during exercise ses2_game. It can be seen that the most common path starts at 'StartRally' with Serve, followed by a ForearmPass, OverHeadPass and Smash in K1.

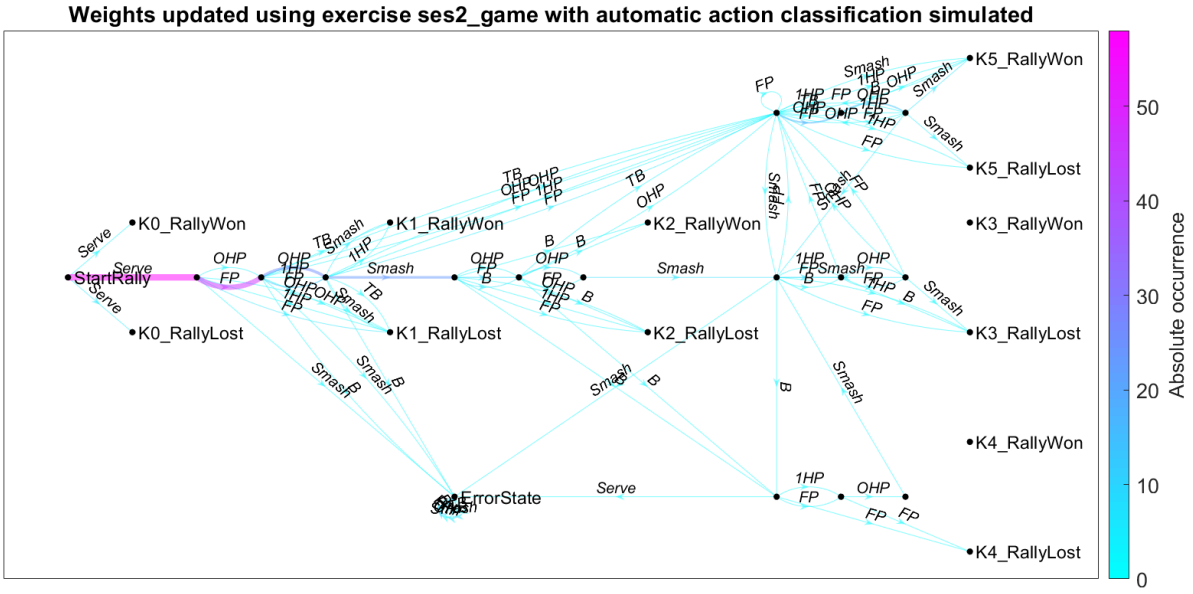


Figure 15: Rally model updated with randomly confused actions performed during exercise ses2_game. It can be seen that the most common path starts at 'StartRally' with Serve, followed by a ForearmPass, OverHeadPass and a Smash in K1. There are also transitions that go to the ErrorState.

The first thing that can be noted immediately when comparing these graphs are the transitions to the ErrorState. In the graph using the actions annotated by hand, there is no transition to the ErrorState, whereas when using the actions randomly confused, there are transitions from different states to the ErrorState. Further analysis and comparison of these graphs is not done visually, but statistically by looking at occurrence of each of the transitions. As mentioned in section 3.4, the Z-test can be used to compare the occurrence of a transition in the error free graph to the mean and average occurrence of this transition when errors are simulated using different confusion matrices. In table 12, these values are given for the first few transitions.

For each row, the Z-test can be performed and this can be used to determine what percentage of the total number of the transitions changed significantly. This is done for the three types of transitions that can change, a transition between two states linked to complexes, a transition to the ErrorState or a transition within the ErrorState. In the bar graph in figure 16, these percentages are given for all simulated scenarios, divided over the three options.

Table 12: Overview of the occurrence of the first few transitions in both the error free LTS and the LTS with simulated errors.

StartNode	EndNode	Label	Error free LTS	Simulated errors in LTS
StartRally	K0_RallyWon	Serve	2	1.95 ± 0.297
StartRally	K0_RallyLost	Serve	5	4.84 ± 0.369
StartRally	Neutral_Serve	Serve	59	56.22 ± 1.580
Neutral_Serve	K1_PointLost	ForearmPass	7	6.41 ± 0.818
Neutral_Serve	K1_PointLost	OneHandPass	1	1.02 ± 0.492
Neutral_Serve	K1_ServeReception	ForearmPass	49	44.85 ± 2.190

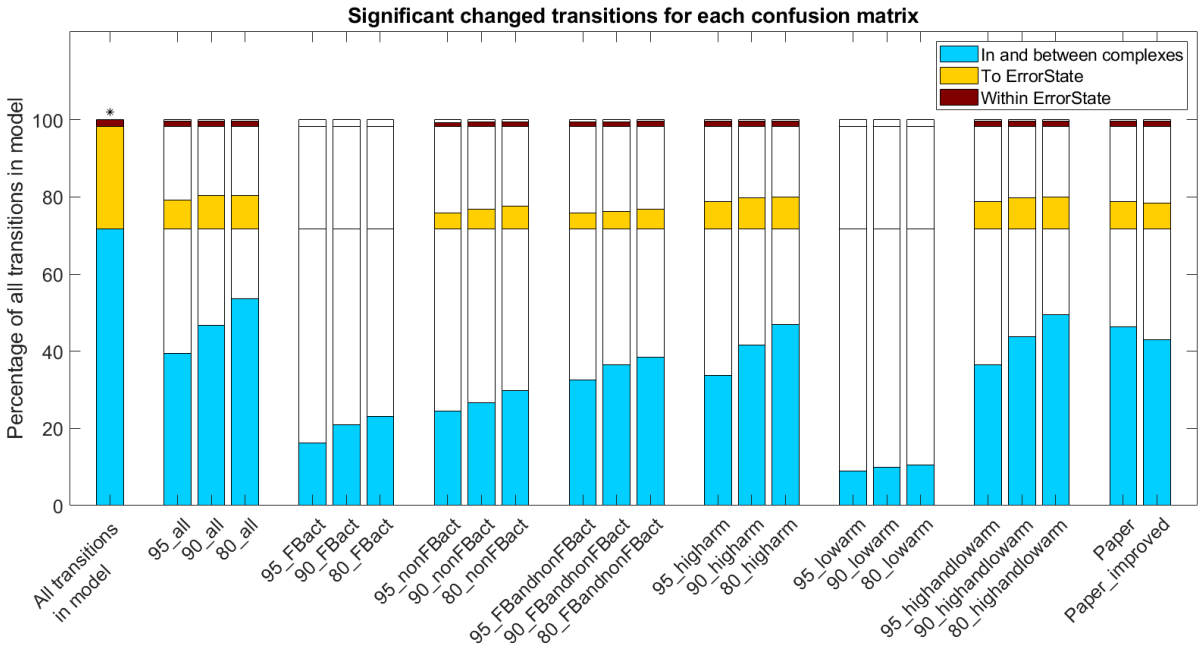


Figure 16: Bar graph showing the percentage of transitions that changed significantly per confusion matrix. The left-most bar (tagged *) is used as reference, showing how the transitions are divided over the three options in the full LTS. On the x-axis, the confusion matrices used are set out, the numbers refer to the recall of the actions and the names to the different confusion matrices.

The pattern seen in figure 16 is that when the recall of actions improves (from 80% to 95%, the number of significant transitions drops. This pattern is seen for all types of confusion matrices. The highest number of significant transitions is seen when actions can be randomly changed into any of the actions ('all') and the lowest when only actions performed with arms low (ForearmPass and OneHandPass) are confused with one another.

The results of two separate confusion matrices (e.g. FBact (table 2) and nonFBact (table 3)) can be compared to the result of the combined confusion matrix (table 4). This shows that the percentage of significant transitions for the combined confusion matrix is less than the sum of the two separate matrices.

Another thing that can be seen is that using the confusion matrix found in literature [6] with the UAR of 67.87%, less transitions are significant than when using the confusion matrices with a recall of 80% and 90% for all actions (giving a UAR of 80% and 90%).

5 Discussion

In the method chapter, algorithms were described and in the results, it was shown what output can be generated using these algorithms. This is done to address the third and fourth sub-questions: *How can a model be created that represents the structural patterns in actual volleyball interactions?* and *How can individual actions that occur in a volleyball session be used to update information presented by the model?* In this chapter, remarkable outputs are discussed and a possible explanation for these results is described. This is done to further elaborate on the algorithms and their implementation.

The results chapter also showed the results of the simulations done for recognition errors. These results are addressed to look into detail at the fifth sub-question: *What is the influence of recognition errors when automatic action classification is used to update the model for interaction?*

Next to discussing the results presented, this chapter also compares the LTS to the SNA performed in earlier research [17, 18, 19, 20] as described in section 2.3. In the last section of this chapter, some final points are discussed on how the LTS can help coaches in training session and give feedback through the SSE project.

5.1 Possible models

In this section, the graphs presented in section 4.1 are discussed by addressing the things that were noted when describing these graphs. There graphs are the results of the algorithms of sections 3.2 and 3.3 and therefore give more insights related to the third and fourth sub-questions. This section is divided the same way as section 4.1. First, the models created by design are discussed, followed by the models created data-driven. A final sub-section is included to discuss the different ways of constructing a model and how these options differentiate from one another.

5.1.1 By design

In total, there are four graphs shown based on models created by design. Twice, the rally model is updated with two parts of the measurement sessions and two exercise models were designed.

Rally model

The results of figures 8 and 9 show that the implementation of both the algorithm to create the model by design as well as the update weights algorithm is successful. In both exercises, the players did not play six-against-six, but four-against-five (ses1_K2K3, figure 8) and three-against-three-against-three (ses1_game, figure 9). This shows that the update weights algorithm is indeed independent of the number of players due to the preprocessing done (described in appendix E). The LTS can even track actions performed in the more complex situation in which teams played three-against-three-against-three.

Next to presenting these results as a graph, it is also useful to present the results as a table like table 10. This table is created by looking at all outgoing transitions from a single node, but it is also possible to create such a table for incoming transitions or transitions with a specific action label. The table shows that team 1 has finished K1 most often and that P1 has performed most smashes. This does not give information on how often each team has played K1, as they could have ended the complex after two touches of the ball, never reaching K1_SetUp.

When updating weights, both a counter of total occurrence of the transition is used as well as the occurrence for each individual player. Using this information, percentages can be given on who performed which transition most often or who performed what action most often. The example shown in table 10 is one of the options to represent this data, but it might be best to interview coaches or players about what output they would like to see.

Exercise models

The results of figures 10 and 11 show that an exercise model can be constructed by design. In figure 10, it was seen that even though actions annotated by hand were used, a transition to the ErrorState was made twice, both times labelled Serve. These transitions start in Neutral_Freeball and Neutral_CounterAttack and a transition is also made to these states from one of the K4 states. As described in step 4 of the algorithm to update weights (section 3.3), the assumption is made that a rally ends if there is at least 5 seconds between two consecutive actions. The results of figure 10 show that this is not a perfect assumption as transitions to the ErrorState are made. By checking the data, it was found that these errors indeed occur because there was less than 5 seconds between the actions. Therefore, it might be better to adjust the assumption so that it is not violated in this exercise.

The assumption can be adjusted in several ways, however, each option has its own disadvantages. A first adjustment would be to make the threshold of ending a rally shorter, so set it to e.g. 4 seconds between actions to start a new rally. However, this leads to more problems, because then the state might be updated to StartRally if still a Smash comes along. This also triggers a transition to the ErrorState which is undesired. Another solution is to state that if a Serve comes along, a new rally starts. In that case, when a Serve comes by 1 second after an OverHeadPass, it is more likely that the Serve is incorrectly recognized and should have been a Smash than that a new rally started. Based on the disadvantages of other assumptions, the threshold was set at 5 seconds accepting the fact that this can still cause errors. In the graphs of figure 8 and 9, a transition to the ErrorState is not seen, meaning the assumption is not violated is these situations.

In the LTS of figure 11, an extra start state is added named 'StartCoach.' This means this LTS has two starting nodes. As mentioned in section 2.3, a graph consists of nodes and edges. Next to that, only one of these nodes can be the starting node, meaning that every time the graph is traversed, the same node is the starting point. Therefore, the way the graph in figure 11 is implemented is not correct. A way to solve this is by combining the two starting nodes ('StartRally' and 'StartCoach') into one starting node ('StartExercise') from which both Serve, UnderHandServe and ThrowBall can be the labels of the outgoing transitions.

Figure 11 also shows that teams were not always able to create an attack from an easy FreeBall. Even though this is not preferred by the coach as he probably wants his team to attack in all cases, it is not something to trouble him. As every rally is unique, there might be more aspects that influence the possibility to attack then only the easy FreeBall. This is also shown in table 11 as starting K5 from K5 only happens in a few cases and more often, players do succeed to attack with a Smash.

5.1.2 Data-driven

For the models created data-driven, two exercises were used to show what the graphs shown as output can look like. Figures 12(a) and 13(a) show what the models look like after the models have been created using the algorithm in section 3.2.2. This already shows that not all actions that can be used to go from one state to the next are included during construction of the model. This is a drawback of this way to create the model as there is limited data to construct the models.

This is also shown once the weights are updated in figures 12(b) and 13(b). Some of the transitions made to the ErrorState are actually valid actions to go to the next state, e.g. OneHandPass and TipBall to go from State4 to EndExercise in figure 12(b). However, in some cases, the transition to the ErrorState is made because the action performed does not make sense. This is seen in figure 12(b) when the transition between State1 and the ErrorState is made labelled ForearmPass. As State1 is the starting state of this exercise, only a Serve or UnderHandServe can be performed.

The graphs in figures 12 and 13 are constructed with a very small set of actions. Coaches are expected to use more data when constructing the model. The short time to create the model has most certainly influenced the transitions to the ErrorState, because if all data of the session is used to create the model, these actions transitioning to the ErrorState would have been added as valid actions. However, to show that the data-driven approach works, it was decided to split the data in half. Next to that, there can always be an action that is not included during creation of the model, so it is a good demonstration of this drawback as well.

5.1.3 Comparison construction of models

There are two ways to create a model, either by design or data-driven and these created models can be a rally model or an exercise model. In the results, it was shown that when a model is created by design, it can indeed be either a rally model or an exercise model. However, for the models created data-driven, only exercise models are shown. No rally models are created with the data-driven approach because it will give a completely different model compared to the rally model created by design. A rally model created data-driven does not represent the rally in a correct way.

This is due to the algorithm used to create the model data-driven. In section 3.2.2, it is mentioned that the algorithm only includes transition from the current state to the consecutive state, e.g. State2 to State3. With the current method of constructing a model data-driven, it is not possible to include transition from e.g. State3 to State5. This is the main reason only for relatively simple exercises (serve, receive, set, attack) models are created data-driven. When using data from a rally (like ses1_game) to create a model data-driven, the result will not look at all like the rally model with the updated weights in figure 9.

A way to improve this is by extending the algorithm to create models data-driven. In that way, it can be made possible that the model learns more from the data than only sequences of actions. With the given algorithm, only the actions are used as input, but by including information on teams and player numbers, may be more information can be extracted from the data. It can be that the model is then able to transit from State3 to State5, by e.g. having states per team (State2, State3, State4 are related to actions performed by team A, whereas State5, State6 and State7 are related to actions of team B). However, more research needs to be done to see if this is a useful approach of creating the rally model data-driven.

5.2 Automatic action classification

In section 4.2, the results are shown of the simulations done with confusion matrices. These results are discussed in this section to answer the fifth sub-question: *What is the influence of recognition errors when automatic action classification is used to update the model for interaction?* In section 3.4, the three types of transitions that can change significantly are described. These are either transitions between states linked to complexes, transitions to the ErrorState and transitions within the ErrorState. In figure 16, the transitions that changed significantly are divided over these three types.

One of the things described in the results is that if only actions with arms low (table 5) are randomly recognized incorrectly, the least number of transitions change significantly. This can be explained by the fact that the UnderHandServe was left out for ses2_game and thus only ForearmPass and OneHandPass are randomly confused with one another. These two actions are both so-called FreeBall actions and in any state it is possible to perform this action. In case the next action is performed by a team mate, the current complex is continued. In case the next action is performed by an opponent, the Forearmpass/OneHandPass is used to go to the next complex, which is K5 (Neutral_Freeball). This means that all recognition errors might result in a significant change of transitions between states linked to complexes. This is not too bad as it is still possible to keep track of the states. This explanation also holds for the bars in

figure 16 named 'FBact' (confusion matrix of table 2). These bars are higher as more actions are confused (ForearmPass, OneHandPass, OverHeadPass and TipBall), but also no transitions to the ErrorState are made.

A second thing mentioned in the results is that combining two confusion matrices does not sum the amount of transitions changed significantly. This means that transitions that change due to one confusion matrix (FBact) also change due to the other confusion matrix (nonFBact). This can be explained by giving two examples, one for each confusion matrix.

First, for nonFBact (table 3): when a Smash is changed into a Serve, this has two consequences. First of all, the original transition with label Smash cannot be updated, instead the transition to the ErrorState labelled Serve is updated. Second, if the next action after the Smash is a ForearmPass, the original transition labelled ForearmPass cannot be updated because the transition to the ErrorState is made. Instead, the loop within the ErrorState labelled ForearmPass is updated. If this happens often within one simulation, all these transitions can change significantly compared to the value in the error free model.

Secondly, for FBact (table 2): when the consecutive ForearmPass mentioned is changed into an OverHeadPass, the same transition labelled ForearmPass decreases due to the error in action classification. Again, if this happens often within one simulation, this transition can change significantly. So even though different actions are confused (Smash to Serve or ForearmPass to OverHeadPass), the same transition labelled ForearmPass can change significantly in both cases. Therefore, when combining confusion matrices, the total number of significant transitions does not sum.

The last thing mentioned in the results that needs some further elaboration is the fact that in figure 16 the bar of the confusion matrix found in literature (table 8) [6] is lower than the bars with overall confusion of all actions (table 1). This might be caused by the fact that in table 8, the recognition of the action TipBall is assumed perfect. However, this probably changes the percentages only slightly as TipBall is a Freeball action. The differences in percentages show that the overall accuracy of a confusion matrix is not the only thing of important to the results. It also matters into what actions incorrect recognized actions are changed and the ratio between confused actions. This can also be seen by looking at the values of the other confusion matrices. For example, when only Freeball actions are confused, no transitions to the ErrorState are made.

The results thus show that the action recognizer should still be improved as the number of transitions that change significantly is high. The results also show that an error in a non-Freeball action is worse than an error in a Freeball action, as the first might trigger a transition to the ErrorState. Therefore, the action recognizer should be improved by increasing the recall for the non-Freeball actions, even at cost of the recognition of the Freeball actions. This can be done using a loss function penalizing the incorrect recognition of non-Freeball actions more than Freeball actions [28]. The recall for the non-Freeball actions should go up to at least 95%, as figure 16 shows that the decrease in significant transitions drops most between 90% and 95%. The recall of the Freeball actions can drop even to around 80%, as long as the actions are confused with other Freeball actions.

5.3 Social Network Analysis

LTS is one way of using GT in volleyball matches. In the background chapter, another way of using GT in volleyball rallies is described, namely SNA [17, 18, 19, 20]. In this section, similarities and difference of both approaches are described to compare both implementations of GT in volleyball.

As described in the background in section 2.3, the nodes of an SN represent different variables of complexes, like number of blockers, tempo of attack and serve type [20]. The edges show what variables occurr simultaneously or consecutively. For example, serve type and serve zone are linked as they happen simultaneously. In an LTS, the nodes represent the states within the complexes and the edges represent transitions (labelled with actions) between these states. This

clearly illustrates the distinction between the lay-outs of the graphs, as the nodes represent different things, the edges also describe other connections between the nodes.

The analysis done with both graphs also differs. For the SNA, the centrality of the nodes is determined using Eigenvector Centrality to show which nodes are most important in the graph [17, 18, 19, 20]. One of these studies showed that in K0, the serve was performed most often in zone 1 and the jump-float serve was the most used serve type [20]. The analysis done with the LTS focusses on what transitions are performed most often and especially which actions are linked to these transitions.

Both types of analysis can give useful insights to the coach. It might be that the LTS is more interpretable to a coach as he probably understands actions more easily than the variables that can occur within a complex. However, this cannot be known for sure based on this current research, so it might be useful to take this into account with the recommendations.

An advantage of the SNA is that the variables of the complexes also take into account the position in the field in which attacks or defences take place. If a video scout (or someone else in the supporting staff of a team) is willing to understand how the complexes work and how they relate to one another, using SNA might give useful insights for match preparation. As the positions of attacks and defences are taken into account, weak defence spots of opponents can be determined to know where to attack as a team.

The reason why this thesis focussed on the LTS is because the LTS also has an advantage over the SN. The data available from the SSE project consists of the actions performed by players recognized from sensor data. These actions are the only input the LTS needs to be able to construct the model data-driven and update the weights of models. The SN used in previous studies also needs position data to determine in what zone actions are performed and more information to determine e.g. attack tempo and number of blockers available.

These parameters are hard to extract from the current available data. The attack tempo is related to the moment of impact of an action, but the recognized action labels are not necessarily centred around the moment of impact. The number of blockers could follow from the floor data, but players with a good position to block might not have the possibility to block and thus visual feedback is needed to determine this variable.

Within volleyball, it is known that the SN can be used to analyse volleyball matches or match-like situations. However, the graph has not yet been used in training session to see if SNA can be applied in those situations as well. This thesis showed that the LTS can be used in both match-like situations as well as training sessions. The reason for this is the possibility to create models by design as well as data-driven.

5.4 SSE project

The layered approach as shown in figure 1 is used in the SSE project. This thesis focusses on the third layer, the recognition of team play using the recognized actions of players as input. As described in the background chapter, the recognition of actions happens using two classifiers [5, 6]. As first step, a classifier separates actions from non-actions [5], as second step, a next classifier recognizes the performed action [6]. To see what the influence is of errors at this second step, simulations were performed as described in section 3.4. However, errors can already be introduced at the first step, when dividing actions from non-actions.

Errors originating at the first step are not taken into account in this thesis but they can have a big influence. As the precision for the action class is only 25.4 % [5], many non-actions are included as actions. These extra actions probably influence the graphs shown as output even more than errors in the second step of the action recognizer. This probably leads to more transitions to the ErrorState and more significant transitions than shown in the bar graph of figure 16. Once the LTS models are included in the SSE project in combination with automatic action recognition, this is something that definitely has to be taken into account.

The layered approach of figure 1 also shows that once team play is recognized, output has to be created that can be shown to the coach or players. Two ways are shown already in this thesis, either showing the results as a graph as in figure 14 or showing the statistics in a table as in table 10. However, there are other ways of showing the output to the players.

In the SSE project, an interactive floor is used to give feedback to the players. The floor also contains pressure sensors to determine player position. Unfortunately, this functionality of the floor is still inoperative. Still, feedback can be given to the players by showing what complex they perform. One way of doing this is by adjusting the background colour of the floor based on the current complex. In a first concept, the floor could be either green or black, where green means that the model still knows it is in some complex and once changed to black, the model reached the ErrorState due to some mistake. However, before this approach of showing the results can work, first, the delay of the model has to be determined to see if the feedback can be given real-time.

As described in section 3.3, the model can only update the current state once the next action is known. For example, if in K1_SetUp a Smash is performed, the model can only decide what the next state is based on the consecutive action. If the Smash is defended, the next state after the Smash is Neutral_Attack, however, if the next action is a Serve, the Smash ended the rally and K1_RallyEnd was reached with the Smash. This means the model is always at least one action lagging when determining the state. This probably influences the feedback given with colours, but further research has to be carried out to see if it is doable.

While annotating, as described in section 3.1.2, two additional tiers of information are added in ELAN to keep track of what players perform exercises together and whether or not the coach influences the exercise. Once the LTS models are included in the SSE project in combination with automatic action recognition, this information does not follow from ELAN any more. Therefore, another way has to be found to know what players perform actions together and when the coach performs an action.

The first problem, linking players together, can be solved by letting players high five each other before performing the exercise. This can be identified in the sensor data linking players together. Another way is to use the floor data once available and by knowing the positions of players, it can be deducted what players perform exercises together.

For the second problem (determining when a coach performs an action to influence the exercise), solutions are a bit different and might ask some multitasking from the coach. As the coach has a UI, he can use this interface to let the system know when he throws in a ball, which can be used when updating weights. Another option to let the system know is by stamping on the interactive floor once the floor is operative. Instead of multitasking, the coach can wear a sensor as well, but this probably causes too much errors in classifying the data.

6 Conclusion and recommendations

In this final chapter, this thesis is wrapped up by looking back at the main research question. Next to that, recommendations for future research are given for further implementation in the SSE project.

6.1 Conclusion

The main research question of this thesis is: *How can the interaction between volleyball players be identified and modelled based on recognized player actions?* This question is divided into smaller sub-questions which are answered throughout the different chapters.

The first sub-question is: *What is interaction in volleyball and how an it be described?* and the question is answered in the background chapter. It was found that interaction can be used to cover all terms related to team play and this team play is seen in sequences of actions as well as communication about tactical decisions made in a rally. Using complexes (K0-K5), interaction can be described based on sequences of actions.

The background chapter also addressed the second sub-question: *What type of model can be used to represent interaction?* A Labelled Transition System can be used to represent the different states of complexes and actions are used to transit from state to state.

In the method chapter, it was shown that a model can be constructed either by design or data-driven, focussing on the third sub-question: *How can a model be created that represents the structural pattern in actual volleyball interactions?*. The algorithm to update weights relates to the fourth sub-question: *How can individual actions that occur in a volleyball session be used to update information presented by the model?* Graphs that can be shown as output to a coach (after constructing the model and updating its weights) are displayed in the results section. The algorithms together with the graphs presented in the results combine into the answer of both sub-questions.

The final sub-question is included to look into the effect of possible errors in action classification: *What is the influence of recognition errors when automatic action classification is used to update the model for interaction?* In the method chapter, it is described that errors in action classification can be simulated using different confusion matrices. The results showed that the percentage of transitions of which the updated weight value changes significantly depends on the confusion matrix used and the recall for different actions. In the discussion chapter, a requirement for the action recognizer is given: Non-Freeball actions have to be recognized with at least a recall of 95%, at cost of the recognition of Freeball actions. The recall of Freeball actions can drop to around 80% when only confused with other Freeball actions. This can be implemented using a loss function with different losses for each action.

Combining all, this thesis shows that interaction between volleyball players can adequately be modelled using an LTS within the framework of complexes using a model created by design. When the model is created data-driven, it is shown that the LTS can also be used outside the structure of complexes to extend the usage of an LTS. Therefore, in the SSE project, the LTS can be used to analyse training sessions and matches in an effective way, however, the action recognizer has to improve before it can replace the annotation of actions by hand.

6.2 Recommendations

The first recommendation is already very clear from the discussion and conclusion, which is the improvement of the action recognizer. The action recognizer has to classify non-Freeball actions with a recall of at least 95% at the cost of the classification of Freeball actions, which recall can drop to 80%. This can be realised by introducing a loss function with different losses for distinct actions.

As described in section 3.1.2, all actions labelled TryBlock were removed from the performed actions. In TryBlock, players try to block the ball, but fail to touch the ball and therefore this action does not influence the state of the game. However, in some cases, it can influence further development of the game. The players who performed TryBlock focus on the blocking and might not be able to perform a next action. It can also be the case that a block fails, meaning the ball is touched, but the opponents do not go to K4. In this cases, the team still has three actions to complete the attack, even though four actions are recognized. With the current models constructed, the LTS goes into the ErrorState as the third action is counted as the fourth action.

The LTS model can be improved by extending the model and implement the actions Block and TryBlock in case the opponents do not go to K4, but the next action is performed on the same side of the field. One way to do this is by implementing a transition from the current state to the current state with the actions Block and TryBlock creating a loop. In this way, the model does not go to the ErrorState after the second action, but the Block/TryBlock action is included in the model. It might also be nice to recognize feint smashes and include these in the LTS. These are important to recognize when determining tactics before a rally. There might be several options to include these actions in the model but this has to be studied further.

Two ways of showing the output to coaches and players are addressed (visually in a graph or values in a table showing statistics), but there might be other ways to present the output to coaches. For future research, interviews can be done with coaches to see what their preference is in presenting the output. Some first options can be shown already to give some incentive but probably coaches have different suggestions for this.

As described in section 5.4, the floor can have a role in showing output to players, e.g. by showing colours linked to complexes, but only if it is possible to use the model in real-time. Therefore, further research needs to be done to determine the delay of the model. It is expected that this delay is at least one action as it should be known if the rally continues or not. To solve this, the model can make a first guess on what the next state might be, based on which state is the most likely. Once it is known what the next state is, the feedback given can be changed if needed.

Once the delay is known and a solution is found if the delay is too much, the concept of changing the background colour based on the current complex can be studied. The first concept mentioned is using green for all complexes and black for the ErrorState, but this can be extended to having a different colour for every complex.

Models created data-driven have the disadvantage that valid actions can lead to a transition to the ErrorState if these actions were not included during the construction of the model. This could be solved by giving the coach the opportunity to add additional transitions which were not added already. This requires more effort from the coach, so it should stay optional. In the same way, he should also have the opportunity to remove transitions he does not like. The latter option can motivate players if the model is linked to real-time feedback on the floor. For example, if the coach wants players to perform a Smash as final action, he can remove all transitions with other actions. The feedback on the floor can then motivate players to always end with a Smash as this results in positive feedback.

Another drawback of the algorithm used to create models data-driven is that the model only adds transitions to consecutive states (State2 to State3). Therefore, data of a rally used to create a model data-driven will never give the rally model created by design. Further research can be done in extending the algorithm of constructing a model data-driven to see if the model can learn more from the data so that it is able to add transitions from e.g. State3 to State5.

Once the UI for the coach as shown figure 5 is ready to use in the SSE project, no models are constructed yet for the coach to select. It is however convenient to have already some models included, so coaches have an example graph and thus something to start off with. For this, all models already created for this thesis can be included. When interviewing coaches about the

output of the model, they can also be asked for some standard exercises they prefer to have included as a model already.

A final recommendation is based on the two models used in recognizing actions from the sensor data. Right now, only a closer look is taken at the second classifier, determining the type of action. However, it might also be interesting to look into influences of errors caused by the first classifier (action vs. non-action). A first step to do this is to look into what causes so many non-actions to be recognized as actions. This could be because of fake smashes or high-fives. Once that is known, additional actions can be added to the data, both randomly and based on this analysis. These new actions can then be used as input for the model to see what influence this first model has on the statistics of the LTS.

References

[1] VolleyCountry. Volleyball - the increasingly popular sporting choice. https://volleycountry.com/news/volleyball-the-increasingly-popular-sporting-choice. Visited on 18-12-2019.

[2] Nederlandse Volleybalbond. https://www.nevobo.nl/over-ons/organisatie/. Visited on 18-12-2019.

[3] Nederlandse Volleybal Bond. Opzet eredivisie zaalvolleyball 2016+. https://www.nevobo.nl/cms/download/3121/20171218%20Opzet%20Eredivisie%202016+.pdf. Visited on 17-01-2020.

[4] Smart Sports Exercises. http://smart-sports-exercises.nl/. Visited on 18-11-2019.

[5] Fasih Haider, Fahim Salim, Sena Busra Yengec, Vahid Naghashi, Izem Tengiz, Kubra Cengiz, Dees Postma, Robby van Delden, Dennis Reidsma, Bert-Jan van Beijnum, and Saturnino Luz. Evaluation of dominant and non-dominant hand movements for volleyball action modelling. In *Adjunct of the 2019 International Conference on Multimodal Interaction on - ICMI '19*, 2019.

[6] Fahim Salim, Fasih Haider, Dees Postma, Robby van Delden, Dennis Reidsma, Saturnino Luz, and Bert-Jan van Beijnum. Evaluation of dominant and non-dominant hand movements for recognising different types of volleyball actions. 2019.

[7] Miguel Silva, Rui Marcelino, Daniel Lacerda, and Paulo João. Match analysis in volleyball: a systematic review. *Montenegrin Journal of Sports Science and Medicine*, 5:35–46, 03 2016.

[8] Jan Van Haaren, Horesh Ben Shitrit, Jesse Davis, and Pascal Fua. Analyzing volleyball match data from the 2014 world championships using machine learning techniques. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*, pages 627–634. ACM Press, 2016.

[9] Spencer Best. Using markov chains to analyze a volleyball rally, 2012. https://dspace.carthage.edu/handle/123456789/406.

[10] Chien Hsun Chen, Chin Fa Chen, Ming Hua Hsu, and Iuon Chang Lin. A blocking prediction for volleyball using neural networks. *Applied Mechanics and Materials*, 373-375:1224–1230, aug 2013.

[11] Cambride dictionary. https://dictionary.cambridge.org/. Visited on 18-01-2020.

[12] Violeta Beniscelli, Gershon Tenenbaum, Robert Joel Schinke, and Miquel Torregrosa. Perceived distributed effort in team ball sports. *Journal of Sports Sciences*, 32(8):710–721, January 2014.

[13] Ana Paulo, Frank T.J.M. Zaal, Ludovic Seifert, Sofia Fonseca, and Duarte Araújo. Predicting volleyball serve-reception at group level. *Journal of Sports Sciences*, 36(22):2621–2630, 2018.

[14] Gaetano Raiola and Alfredo Di Tore. Non-verbal communication and volleyball: A new way to approach the phenomenon. *Mediterranean Journal of Social Sciences*, 3(2):347–356, 2012.

[15] Raúl Hileno and Bernat Buscà. Observational tool for analyzing attack coverage in volleyball. *Revista Internacional de Medicina y Ciencias de la Actividad Fisica y del Deporte*, 12:557–570, 10 2012.

[16] Lorenzo Laporta, Pantelis Nikolaidis, Luke Thomas, and José Afonso. The importance of loosely systematized game phases in sports: The case of attack coverage systems in high-level women's volleyball. *Montenegrin Journal of Sports Science and Medicine*, 4:19–24, 03 2015.

[17] Marta Hurst, Manuel Loureiro, Beatriz Valongo, Lorenzo Laporta, T. Pantelis Nikolaidis, and Jos Afonso. Systemic mapping of high-level womens volleyball using social network analysis: The case of serve (k0), side-out (ki), side-out transition (kii) and transition (kiii). *International Journal of Performance Analysis in Sport*, 16(2):695–710, 2016.

[18] Marta Hurst, Manuel Louriero, Beatrix Valongo, Lorenzo Laporta, Pantelis Nikolaidis, and José Afonso. Systemic mapping of high-level women's volleyball using social network analysis: The case of attack coverage, freeball, and downball. *Montenegrin Journal of Sports Science and Medicine*, 6:57–64, 03 2017.

[19] Manuel Loureiro, Marta Hurst, Beatriz Valongo, Pantelis Nikolaidis, Lorenzo Laporta, and José Afonso. A comprehensive mapping of high-level men's volleyball gameplay through social network analysis: Analysing serve, side-out, side-out transition and transition. *Montenegrin Journal of Sports Science and Medicine*, 6(2):35–41, September 2017.

[20] Lorenzo Laporta, José Afonso, and Isabel Mesquita. Interaction network analysis of the six game complexes in high-level volleyball through the use of eigenvector centrality. *PLOS ONE*, 13(9):e0203348, sep 2018.

[21] Yufan Wang, Yuliang Zhao, Rosa H. M. Chan, and Wen J. Li. Volleyball skill assessment using a single wearable micro inertial measurement unit at wrist. *IEEE Access*, 6:13758–13765, 2018.

[22] L. Ponce Cuspinera, Sakura Uetsuji, F. J. Ordonez Morales, and Daniel Roggen. Beach volleyball serve type recognition. In *Proceedings of the 2016 ACM International Symposium on Wearable Computers - ISWC '16*. ACM Press, 09 2016.

[23] Thomas Kautz, Benjamin H. Groh, Julius Hannink, Ulf Jensen, Holger Strubberg, and Bjoern M. Eskofier. Activity recognition in beach volleyball using a deep convolutional neural network. *Data Mining and Knowledge Discovery*, 31(6):1678–1705, feb 2017.

[24] Robin J. Wilson. *Introduction to Graph Theory*, chapter 1: Introduction, What is a graph?, pages 1–7. Longman Group, 4th edition, 1996.

[25] Frank Emmert-Streib. A brief introduction to complex networks and their analysis. In Matthias Dehmer, editor, *Structural Analysis of Complex Networks*, pages 1–26. Birkhäuser Boston, 2011.

[26] R.J. van Glabbeek. *Handbook of Process Algebra*, chapter The Linear Time - Branching Time Spectrum I.* The Semantics of Concrete, Sequantial Processes, pages 1–99. Elsevier Science, 2001.

[27] Deborah Rumsey. *Statistics Essentials for Dummies*. Wiley Publishing, 2010.

[28] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[29] Youtube. Volleyball coverage drill - russ rose. https://www.youtube.com/watch?v=NfI1fnxw9uc. Visited on 20-06-2019.

A Measurement protocol

For the measurement sessions, the following protocol is created. As every coach is unique, each coach has his own interpretation of the exercises described. Adjustments made during measurement sessions are described in the method chapter.

Before measurement

- Let all players sign the consent form (also players without sensors because they are on film).
- Make photos of the players with sensors to know everyone's player number.
- Put the IMU sensors on the posterior side of the wrists of the players. Players can replace the sensors a bit so that they are not hindering performance.
- Start the video recording and start the measurement of the sensor on the video recording (necessary for alignment of the labels and data).

Measurements

Calibration

- Get all players with sensors in line for the calibration, parallel to the net, facing the net. This can be assured by letting them stand on one of the lines in the field. The calibration consists of the following:
  - Stand in N-pose.
  - Move arms to shoulder height in front of the body (keep elbows stretched). Shoulder joint makes an angle of about 90 degrees.
  - Mover arms back to N-pose.
  - Move arms to T-pose.
  - Bend spine forward (try to keep the back stretched and keep arms in position). Hip joint makes an angle of about 90 degrees.
  - Move back to T-pose.
  - Move back to N-pose.
- For every time data collection is stopped and started again, the calibration has to be redone. It is important to tell all players to align with the net carefully for orientation purposes. It might even be useful to put a sensor on the net to know the orientation of the net.
- To keep relatively short sessions for labelling, after each set of actions, restart the video and redo the calibration.

Drills for reference

- As the different complexes play an important role for the data collection, some drills will be done to have enough data as reference for the complexes. These are as follows:

  0. Serve
     This complex can be trained by serving ten times.
     If enough sensors are available, this can be combined with the drill of K1 and K2.

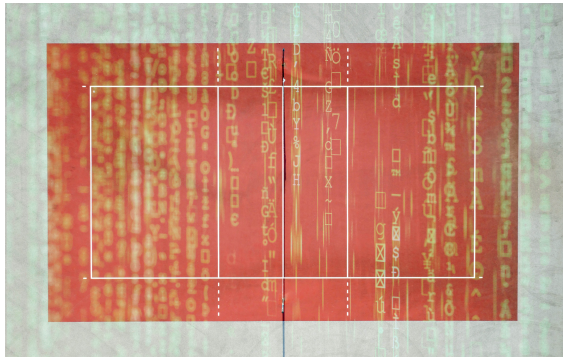
1. Reception of serve + give attack
   This complex can be trained by letting someone serve and give the team with sensors the possibility to attack from this serve.
   If enough sensors are available, this can be combined with the drill of K0 and K2.
2. Defend attack + give counterattack
   This complex can be trained by letting the opponent attack from K1, so that the team with sensors has to defend the attack, but also give a counter attack.
   If enough sensors are available, this can be combined with the drill of K0 and K1.
3. Defend counterattack + give counterattack
   This complex can be trained by letting the opponent attack, so that the team with sensors has to defend that attack, but also give a counter attack.
   If enough sensors are available, this can be combined with the drill of K5.
4. Defend block + give counterattack
   This complex is hard to train with a drill, as it is hard to smash into a block with as goal to let the rally continue. It is possible to train it by having someone to throw in the ball immediately after the attack took place [29]. This makes players prepared if the opponent places a successful block.
5. Defend freeball + give counterattack
   This complex can be trained by throwing a ball in the field by the trainer or other players and create an attack from there.
   If enough sensors are available, this can be combined with the drill of K3.

Friendly game

- After the drills of the specific complexes, the players can play a friendly game. From this friendly game, complexes can be recognized and labelled.
- Depending on the time available, the number of sets to be player will be determined. Of course, the more the better, as that means more data will be available.

INFORMATION BROCHURE

Interaction Technology in Volleyball
University of Twente



University of Twente
Human Media Interaction
Dennis Reidsma, Assistant Professor
d.reidsma@utwente.nl

This brochure contains information about the research project: *Smart Sports Exercises in Volleyball*. This brochure helps you to reach a decision on whether or not you want to participate in this research project.

Background
Technology plays an ever-greater role in our daily lives. Technology helps us to live healthier lives and get the best out of ourselves. The same is true in sports. We rely on *wearables*, *apps* and other technological advancements to aid our performance in sports.

The University of Twente, in collaboration with the Windesheim University of Applied Sciences, is investigating the influence of technology in sports. Specifically, the present research project examines the potential benefits for a 'smart' gym floor in volleyball.

LedGo, a company specialized in LED-technology, developed a high-tech gym floor. The system is comprised of pressure-sensors and LED-technology which makes it possible to measure player position over time and provide real-time feedback about training and sports performance.

The current research project, led by the University of Twente and the Windesheim University of Applied Sciences, is aimed at developing suitable training methods for this new form of technology. Ultimately to enhance player performance in volleyball.

Procedure
To establish what the potential benefit of the 'smart' gym floor might be in the context of volleyball, research data need to be acquired. This data might consist of, but is not limited to: interviews, observations and video-recordings.

In this folder, we will explain what it means for you to participate in this research project. Your participation in this research is entirely voluntary. If you have any remaining questions after reading this brochure, please contact the University of Twente (Dennis Reidsma, for contact information see title page).

Participation
Your participation in this research is entirely voluntary. You are always free to change your mind and stop participating, even if you agreed earlier. No reason for doing so will need to be disclosed. When you decide to participate, you only have to provide consent once, thereafter your consent will be valid for the remainder of this academic year (until 31-08-2019). Data will be stored for at least 10 years, conform the NVSU-guideline. However, you do have the right to be forgotten, which means that you can request all personal data to be erased from the research-project database.

What do the research activities entail?
All research activities will predominantly serve one of three purposes: 1) to better understand volleyball in general 2) to better understand the challenges that players and trainers face in practicing volleyball 3) to gain insight into the potential benefits from using a 'smart' gym floor in volleyball. Research activities will include, but are not limited to: observations, interviews, video-recordings, ethnographic research and taking measurements using non-obtrusive sensors (e.g. Inertial Measurement Units). In the future, some players, teams and trainers might also be asked to test early prototypes of smart sports exercises for volleyball.

Which data will be obtained?
Over the course of the research project, video-recordings, observations, interviews, questionnaires and sensor-data (IMU-data) will be obtained. All data will be handled with extreme care and will be anonymized. Data will be treated and stored according to the rules and regulations of the AVG and the GDPR. All data will be stored for a minimum of 10 years, following the NVSU-guideline.

Who has access to the data?
All data, including the video-recordings, will only be accessible to researchers involved in this research project. A list of people that have access to the data is available and can be requested from Dennis Reidsma. None of the data will be made public and none of the data will be used for promotional purposes without explicit consent.

Additional information and independent advice.
If you wish to seek independent advice or file a complaint, you can contact the secretary of the ethics committee of the University of Twente (ethics-comm-ewi@utwente.nl).

For any additional questions regarding this research, please contact Dennis Reidsma or any of the other researchers involved in this research project (affiliated with the University of Twente or the Windesheim University of Applied Sciences).

Partners

ZonMw

Windesheim
UNIVERSITY OF APPLIED SCIENCES

UNIVERSITY OF TWENTE.

sport service veenendaal

LED GO

INNOSPORTLAB
SPORT & BEWEEG!
INNOVATIE IN BREEDTESPORT

Human Media Interaction.

BIOMEDICAL SIGNALS AND SYSTEMS
UNIVERSITY OF TWENTE.

INFORMED CONSENT FORM

Purpose of the research
This research, led by the University of Twente and the Windesheim University of Applied Sciences, aims to develop a suite of smart sports exercises that can be used with a next-generation interactive floor in volleyball.

Research: Interaction Technology in Volleyball
I hereby declare that I am fully informed about the purpose of the research, the research procedure and the relevant research methodology. I have read and I understand the provided information and have had the opportunity to ask questions. I give my consent for the collection of anonymized data, the kind of which has been described in the information brochure. I give my consent for recording audio and video for research and evaluation purposes.

Any (video) data will solely be accessible to researchers affiliated with this research and will never be made public. None of the research data will be used for promotional purposes without explicit consent. All data will be treated and stored according to the rules and regulations of the AVG and the GDPR. All data will be stored for a minimum of 10 years, conform the NVSU-guideline. You have the 'right to be forgotten', which means that you can request your data to be erased from the research database if you wish. Your consent will be valid for the remainder of the academic year (until 31-08-2019).

I understand that my participation is voluntary and that I am free to withdraw at any time, without giving a reason and without cost.

☐ Hereby, I also grant permission for sharing the video recordings with my trainer / coach. Other data, like interviews and sensor data, may also be shared but only after consultation.

City:

Date:

Full name:

Signature:

……

Principal investigators:
Dennis Reidsma[1], Robby van Delden[1], Dees Postma[1], Bert-Jan van Beijnum[2], Fahim Salim[2], Ivo van Hilvoorde[3], Jeroen Koekoek[3] and Wytse Walinga[3]
[1]University of Twente | Electrotechniek, Wiskunde en Informatica | Human Media Interaction
[2]University of Twente | Electrotechniek, Wiskunde en Informatica | Biomedical Signals and Systems
[3]Windesheim University of Applied Sciences | Human Movement, School and Sports

Contact information
If you have any questions concerning this research, please contact Dennis Reidsma (d.reidsma@utwente.nl), any of the researchers on site or the secretary of the Ethics Committee (ethics-comm-ewi@utwente.nl). The Ethics Committee is composed of highly trained research-professionals affiliated with the University of Twente. The Ethics Committee is available for any questions or complaints you might have regarding the research.

C Annotation protocol

For annotation, the annotation tool ELAN 5.6-FX can be used. To make sure results are reproducible, at first, labels should be reproducible and for this, an annotation protocol is created.

Action labelling

In ELAN, it is possible to create a label tier for each individual player. This tier contains all actions performed by that player. For labelling actions, it is important to define the starting and ending point of an action. When performing no action, players have their arms in a neutral rest position, either with their hands at hip height or with their hands little higher to be ready for an action. The start of an action is defined as the moment a player moves the arms from this neutral position. The end of an action is defined at the moment the arms are back at rest position.

The actions are labelled as one of the following actions:

- Block (at least touch the ball)
- TryBlock (try a block, but fail to touch the ball)
- Serve (performed overhead, like a smash)
- Underhand Serve
- ForearmPass
- OneHandTouch
- OneHandPass
- LeftHandPass
- RightHandPass
- OverHeadPass
- Smash
- TipBall
- TipOverNet

Depending on the application for which the actions are used, labels can be combined or removed during processing. For example, in action recognition, it might be better to remove the labels with low occurrence to improve the machine learning model. It is also possible to combine all the passes performed with a single hand (either left or right).

Player labelling

During some of the exercises, the combination of players performing exercises might change during the training. Therefore, it is also important to include a tier with information on when players perform exercises together. For both sides of the field an extra tier is added in which is labelled what players perform exercises together.

The label contains the player numbers of all players who are part of the exercise. This also included players that do not perform an action over the time of the exercise, because they could have performed an action if the ball was played different. The starting point of the player label is defined at least one second before the first action performed by one of the players. The labels ends at least one second after the last action performed.

Coach actions

In some exercises, the coach also influences the model of the exercise by e.g. throwing in the ball. This also has to be labelled for the model to know when instead of one of the players, the coach performs an action. As the action of the coach is only used to know when he influenced the exercise and not how long the action took, the labels are randomly placed around the time the coach influences the exercise.

As the time windows for action recognition are defined at 0.5 sec with 50% overlap, the duration of the label by the coach has to be at least 0.75 sec. This value is found as the label has to cover at the full time window to be extracted as label. This means that a label of 0.5 sec might not be always extracted as label, but from 0.75 sec on, this is the case.

D States and Transtions in the LTS

Table 13: All states S present in the Labelled Transition System

StartRally	K3_PointWon	K1_ServeReception	K3_SetUp
K0_PointWon	K3_PointLost	K1_SetUp	Neutral_Block
K0_PointLost	K4_PointWon	Neutral_Attack	K4_BlockDefence
K1_PointWon	K4_PointLost	K2_AttackDefence	K4_SetUp
K1_PointLost	K5_PointWon	K2_SetUp	Neutral_Freeball
K2_PointWon	K5_PointLost	Neutral_CounterAttack	K5_FreeballDefence
K2_PointLost	Neutral_Serve	K3_CounterAttackDefence	K5_SetUp

Table 14: All transitions $\rightarrow$ with labels Λ present in the Labelled Transition System

StartNode	**EndNode**	**Label**
StartRally	K0_PointWon	Serve
StartRally	K0_PointWon	UnderHandServe
StartRally	K0_PointLost	Serve
StartRally	K0_PointLost	UnderHandServe
StartRally	Neutral_Serve	Serve
StartRally	Neutral_Serve	UnderHandServe
Neutral_Serve	K1_PointWon	ForearmPass
Neutral_Serve	K1_PointWon	OneHandPass
Neutral_Serve	K1_PointWon	OverHeadPass
Neutral_Serve	K1_PointWon	Smash
Neutral_Serve	K1_PointWon	TipBall
Neutral_Serve	K1_PointLost	ForearmPass
Neutral_Serve	K1_PointLost	OneHandPass
Neutral_Serve	K1_PointLost	OverHeadPass
Neutral_Serve	K1_PointLost	Smash
Neutral_Serve	K1_PointLost	TipBall
Neutral_Serve	K1_ServeReception	ForearmPass
Neutral_Serve	K1_ServeReception	OneHandPass
Neutral_Serve	K1_ServeReception	OverHeadPass
Neutral_Serve	K1_ServeReception	TipBall
Neutral_Serve	Neutral_Attack	Smash
Neutral_Serve	Neutral_Freeball	ForearmPass
Neutral_Serve	Neutral_Freeball	OneHandPass
Neutral_Serve	Neutral_Freeball	OverHeadPass
Neutral_Serve	Neutral_Freeball	TipBall
K1_ServeReception	K1_PointWon	ForearmPass
K1_ServeReception	K1_PointWon	OneHandPass
K1_ServeReception	K1_PointWon	OverHeadPass
K1_ServeReception	K1_PointWon	Smash
K1_ServeReception	K1_PointWon	TipBall
K1_ServeReception	K1_PointLost	ForearmPass
K1_ServeReception	K1_PointLost	OneHandPass
K1_ServeReception	K1_PointLost	OverHeadPass
K1_ServeReception	K1_PointLost	Smash
K1_ServeReception	K1_PointLost	TipBall
K1_ServeReception	K1_SetUp	ForearmPass

StartNode	EndNode	Label
K1_ServeReception	K1_SetUp	OneHandPass
K1_ServeReception	K1_SetUp	OverHeadPass
K1_ServeReception	K1_SetUp	TipBall
K1_ServeReception	Neutral_Attack	Smash
K1_ServeReception	Neutral_Freeball	ForearmPass
K1_ServeReception	Neutral_Freeball	OneHandPass
K1_ServeReception	Neutral_Freeball	OverHeadPass
K1_ServeReception	Neutral_Freeball	TipBall
K1_SetUp	K1_PointWon	ForearmPass
K1_SetUp	K1_PointWon	OneHandPass
K1_SetUp	K1_PointWon	OverHeadPass
K1_SetUp	K1_PointWon	Smash
K1_SetUp	K1_PointWon	TipBall
K1_SetUp	K1_PointLost	ForearmPass
K1_SetUp	K1_PointLost	OneHandPass
K1_SetUp	K1_PointLost	OverHeadPass
K1_SetUp	K1_PointLost	Smash
K1_SetUp	K1_PointLost	TipBall
K1_SetUp	Neutral_Attack	Smash
K1_SetUp	Neutral_Freeball	ForearmPass
K1_SetUp	Neutral_Freeball	OneHandPass
K1_SetUp	Neutral_Freeball	OverHeadPass
K1_SetUp	Neutral_Freeball	TipBall
Neutral_Attack	K2_PointWon	Block
Neutral_Attack	K2_PointWon	ForearmPass
Neutral_Attack	K2_PointWon	OneHandPass
Neutral_Attack	K2_PointWon	OverHeadPass
Neutral_Attack	K2_PointWon	Smash
Neutral_Attack	K2_PointWon	TipBall
Neutral_Attack	K2_PointLost	Block
Neutral_Attack	K2_PointLost	ForearmPass
Neutral_Attack	K2_PointLost	OneHandPass
Neutral_Attack	K2_PointLost	OverHeadPass
Neutral_Attack	K2_PointLost	Smash
Neutral_Attack	K2_PointLost	TipBall
Neutral_Attack	K2_AttackDefence	Block
Neutral_Attack	K2_AttackDefence	ForearmPass
Neutral_Attack	K2_AttackDefence	OneHandPass
Neutral_Attack	K2_AttackDefence	OverHeadPass
Neutral_Attack	K2_AttackDefence	TipBall
Neutral_Attack	Neutral_CounterAttack	Smash
Neutral_Attack	Neutral_Block	Block
Neutral_Attack	Neutral_Freeball	ForearmPass
Neutral_Attack	Neutral_Freeball	OneHandPass
Neutral_Attack	Neutral_Freeball	OverHeadPass
Neutral_Attack	Neutral_Freeball	TipBall
K2_AttackDefence	K2_PointWon	Block
K2_AttackDefence	K2_PointWon	ForearmPass
K2_AttackDefence	K2_PointWon	OneHandPass
K2_AttackDefence	K2_PointWon	OverHeadPass

StartNode	EndNode	Label
K2_AttackDefence	K2_PointWon	Smash
K2_AttackDefence	K2_PointWon	TipBall
K2_AttackDefence	K2_PointLost	Block
K2_AttackDefence	K2_PointLost	ForearmPass
K2_AttackDefence	K2_PointLost	OneHandPass
K2_AttackDefence	K2_PointLost	OverHeadPass
K2_AttackDefence	K2_PointLost	Smash
K2_AttackDefence	K2_PointLost	TipBall
K2_AttackDefence	K2_SetUp	Block
K2_AttackDefence	K2_SetUp	ForearmPass
K2_AttackDefence	K2_SetUp	OneHandPass
K2_AttackDefence	K2_SetUp	OverHeadPass
K2_AttackDefence	K2_SetUp	TipBall
K2_AttackDefence	Neutral_CounterAttack	Smash
K2_AttackDefence	Neutral_Block	Block
K2_AttackDefence	Neutral_Freeball	ForearmPass
K2_AttackDefence	Neutral_Freeball	OneHandPass
K2_AttackDefence	Neutral_Freeball	OverHeadPass
K2_AttackDefence	Neutral_Freeball	TipBall
K2_SetUp	K2_PointWon	Block
K2_SetUp	K2_PointWon	ForearmPass
K2_SetUp	K2_PointWon	OneHandPass
K2_SetUp	K2_PointWon	OverHeadPass
K2_SetUp	K2_PointWon	Smash
K2_SetUp	K2_PointWon	TipBall
K2_SetUp	K2_PointLost	Block
K2_SetUp	K2_PointLost	ForearmPass
K2_SetUp	K2_PointLost	OneHandPass
K2_SetUp	K2_PointLost	OverHeadPass
K2_SetUp	K2_PointLost	Smash
K2_SetUp	K2_PointLost	TipBall
K2_SetUp	Neutral_CounterAttack	Smash
K2_SetUp	Neutral_Block	Block
K2_SetUp	Neutral_Freeball	ForearmPass
K2_SetUp	Neutral_Freeball	OneHandPass
K2_SetUp	Neutral_Freeball	OverHeadPass
K2_SetUp	Neutral_Freeball	TipBall
Neutral_CounterAttack	K3_PointWon	Block
Neutral_CounterAttack	K3_PointWon	ForearmPass
Neutral_CounterAttack	K3_PointWon	OneHandPass
Neutral_CounterAttack	K3_PointWon	OverHeadPass
Neutral_CounterAttack	K3_PointWon	Smash
Neutral_CounterAttack	K3_PointWon	TipBall
Neutral_CounterAttack	K3_PointLost	Block
Neutral_CounterAttack	K3_PointLost	ForearmPass
Neutral_CounterAttack	K3_PointLost	OneHandPass
Neutral_CounterAttack	K3_PointLost	OverHeadPass
Neutral_CounterAttack	K3_PointLost	Smash
Neutral_CounterAttack	K3_PointLost	TipBall
Neutral_CounterAttack	Neutral_CounterAttack	Smash

StartNode	EndNode	Label
Neutral_CounterAttack	K3_CounterAttackDefence	Block
Neutral_CounterAttack	K3_CounterAttackDefence	ForearmPass
Neutral_CounterAttack	K3_CounterAttackDefence	OneHandPass
Neutral_CounterAttack	K3_CounterAttackDefence	OverHeadPass
Neutral_CounterAttack	K3_CounterAttackDefence	TipBall
Neutral_CounterAttack	Neutral_Block	Block
Neutral_CounterAttack	Neutral_Freeball	ForearmPass
Neutral_CounterAttack	Neutral_Freeball	OneHandPass
Neutral_CounterAttack	Neutral_Freeball	OverHeadPass
Neutral_CounterAttack	Neutral_Freeball	TipBall
K3_CounterAttackDefence	K3_PointWon	Block
K3_CounterAttackDefence	K3_PointWon	ForearmPass
K3_CounterAttackDefence	K3_PointWon	OneHandPass
K3_CounterAttackDefence	K3_PointWon	OverHeadPass
K3_CounterAttackDefence	K3_PointWon	Smash
K3_CounterAttackDefence	K3_PointWon	TipBall
K3_CounterAttackDefence	K3_PointLost	Block
K3_CounterAttackDefence	K3_PointLost	ForearmPass
K3_CounterAttackDefence	K3_PointLost	OneHandPass
K3_CounterAttackDefence	K3_PointLost	OverHeadPass
K3_CounterAttackDefence	K3_PointLost	Smash
K3_CounterAttackDefence	K3_PointLost	TipBall
K3_CounterAttackDefence	Neutral_CounterAttack	Smash
K3_CounterAttackDefence	K3_SetUp	Block
K3_CounterAttackDefence	K3_SetUp	ForearmPass
K3_CounterAttackDefence	K3_SetUp	OneHandPass
K3_CounterAttackDefence	K3_SetUp	OverHeadPass
K3_CounterAttackDefence	K3_SetUp	TipBall
K3_CounterAttackDefence	Neutral_Block	Block
K3_CounterAttackDefence	Neutral_Freeball	ForearmPass
K3_CounterAttackDefence	Neutral_Freeball	OneHandPass
K3_CounterAttackDefence	Neutral_Freeball	OverHeadPass
K3_CounterAttackDefence	Neutral_Freeball	TipBall
K3_SetUp	K3_PointWon	Block
K3_SetUp	K3_PointWon	ForearmPass
K3_SetUp	K3_PointWon	OneHandPass
K3_SetUp	K3_PointWon	OverHeadPass
K3_SetUp	K3_PointWon	Smash
K3_SetUp	K3_PointWon	TipBall
K3_SetUp	K3_PointLost	Block
K3_SetUp	K3_PointLost	ForearmPass
K3_SetUp	K3_PointLost	OneHandPass
K3_SetUp	K3_PointLost	OverHeadPass
K3_SetUp	K3_PointLost	Smash
K3_SetUp	K3_PointLost	TipBall
K3_SetUp	Neutral_CounterAttack	Smash
K3_SetUp	Neutral_Block	Block
K3_SetUp	Neutral_Freeball	ForearmPass
K3_SetUp	Neutral_Freeball	OneHandPass
K3_SetUp	Neutral_Freeball	OverHeadPass

StartNode	EndNode	Label
K3_SetUp	Neutral_Freeball	TipBall
Neutral_Block	K4_PointWon	ForearmPass
Neutral_Block	K4_PointWon	OneHandPass
Neutral_Block	K4_PointWon	OverHeadPass
Neutral_Block	K4_PointWon	Smash
Neutral_Block	K4_PointWon	TipBall
Neutral_Block	K4_PointLost	ForearmPass
Neutral_Block	K4_PointLost	OneHandPass
Neutral_Block	K4_PointLost	OverHeadPass
Neutral_Block	K4_PointLost	Smash
Neutral_Block	K4_PointLost	TipBall
Neutral_Block	Neutral_CounterAttack	Smash
Neutral_Block	K4_BlockDefence	ForearmPass
Neutral_Block	K4_BlockDefence	OneHandPass
Neutral_Block	K4_BlockDefence	OverHeadPass
Neutral_Block	K4_BlockDefence	TipBall
Neutral_Block	Neutral_Freeball	ForearmPass
Neutral_Block	Neutral_Freeball	OneHandPass
Neutral_Block	Neutral_Freeball	OverHeadPass
Neutral_Block	Neutral_Freeball	TipBall
K4_BlockDefence	K4_PointWon	ForearmPass
K4_BlockDefence	K4_PointWon	OneHandPass
K4_BlockDefence	K4_PointWon	OverHeadPass
K4_BlockDefence	K4_PointWon	Smash
K4_BlockDefence	K4_PointWon	TipBall
K4_BlockDefence	K4_PointLost	ForearmPass
K4_BlockDefence	K4_PointLost	OneHandPass
K4_BlockDefence	K4_PointLost	OverHeadPass
K4_BlockDefence	K4_PointLost	Smash
K4_BlockDefence	K4_PointLost	TipBall
K4_BlockDefence	Neutral_CounterAttack	Smash
K4_BlockDefence	K4_SetUp	ForearmPass
K4_BlockDefence	K4_SetUp	OneHandPass
K4_BlockDefence	K4_SetUp	OverHeadPass
K4_BlockDefence	K4_SetUp	TipBall
K4_BlockDefence	Neutral_Freeball	ForearmPass
K4_BlockDefence	Neutral_Freeball	OneHandPass
K4_BlockDefence	Neutral_Freeball	OverHeadPass
K4_BlockDefence	Neutral_Freeball	TipBall
K4_SetUp	K4_PointWon	ForearmPass
K4_SetUp	K4_PointWon	OneHandPass
K4_SetUp	K4_PointWon	OverHeadPass
K4_SetUp	K4_PointWon	Smash
K4_SetUp	K4_PointWon	TipBall
K4_SetUp	K4_PointLost	ForearmPass
K4_SetUp	K4_PointLost	OneHandPass
K4_SetUp	K4_PointLost	OverHeadPass
K4_SetUp	K4_PointLost	Smash
K4_SetUp	K4_PointLost	TipBall
K4_SetUp	Neutral_CounterAttack	Smash

StartNode	EndNode	Label
K4_SetUp	Neutral_Freeball	ForearmPass
K4_SetUp	Neutral_Freeball	OneHandPass
K4_SetUp	Neutral_Freeball	OverHeadPass
K4_SetUp	Neutral_Freeball	TipBall
Neutral_Freeball	K5_PointWon	Block
Neutral_Freeball	K5_PointWon	ForearmPass
Neutral_Freeball	K5_PointWon	OneHandPass
Neutral_Freeball	K5_PointWon	OverHeadPass
Neutral_Freeball	K5_PointWon	Smash
Neutral_Freeball	K5_PointWon	TipBall
Neutral_Freeball	K5_PointLost	Block
Neutral_Freeball	K5_PointLost	ForearmPass
Neutral_Freeball	K5_PointLost	OneHandPass
Neutral_Freeball	K5_PointLost	OverHeadPass
Neutral_Freeball	K5_PointLost	Smash
Neutral_Freeball	K5_PointLost	TipBall
Neutral_Freeball	Neutral_CounterAttack	Smash
Neutral_Freeball	Neutral_Block	Block
Neutral_Freeball	Neutral_Freeball	ForearmPass
Neutral_Freeball	Neutral_Freeball	OneHandPass
Neutral_Freeball	Neutral_Freeball	OverHeadPass
Neutral_Freeball	Neutral_Freeball	TipBall
Neutral_Freeball	K5_FreeballDefence	Block
Neutral_Freeball	K5_FreeballDefence	ForearmPass
Neutral_Freeball	K5_FreeballDefence	OneHandPass
Neutral_Freeball	K5_FreeballDefence	OverHeadPass
Neutral_Freeball	K5_FreeballDefence	TipBall
K5_FreeballDefence	K5_PointWon	Block
K5_FreeballDefence	K5_PointWon	ForearmPass
K5_FreeballDefence	K5_PointWon	OneHandPass
K5_FreeballDefence	K5_PointWon	OverHeadPass
K5_FreeballDefence	K5_PointWon	Smash
K5_FreeballDefence	K5_PointWon	TipBall
K5_FreeballDefence	K5_PointLost	Block
K5_FreeballDefence	K5_PointLost	ForearmPass
K5_FreeballDefence	K5_PointLost	OneHandPass
K5_FreeballDefence	K5_PointLost	OverHeadPass
K5_FreeballDefence	K5_PointLost	Smash
K5_FreeballDefence	K5_PointLost	TipBall
K5_FreeballDefence	Neutral_CounterAttack	Smash
K5_FreeballDefence	Neutral_Block	Block
K5_FreeballDefence	Neutral_Freeball	ForearmPass
K5_FreeballDefence	Neutral_Freeball	OneHandPass
K5_FreeballDefence	Neutral_Freeball	OverHeadPass
K5_FreeballDefence	Neutral_Freeball	TipBall
K5_FreeballDefence	K5_SetUp	Block
K5_FreeballDefence	K5_SetUp	ForearmPass
K5_FreeballDefence	K5_SetUp	OneHandPass
K5_FreeballDefence	K5_SetUp	OverHeadPass
K5_FreeballDefence	K5_SetUp	TipBall

StartNode	EndNode	Label
K5_SetUp	K5_PointWon	Block
K5_SetUp	K5_PointWon	ForearmPass
K5_SetUp	K5_PointWon	OneHandPass
K5_SetUp	K5_PointWon	OverHeadPass
K5_SetUp	K5_PointWon	Smash
K5_SetUp	K5_PointWon	TipBall
K5_SetUp	K5_PointLost	Block
K5_SetUp	K5_PointLost	ForearmPass
K5_SetUp	K5_PointLost	OneHandPass
K5_SetUp	K5_PointLost	OverHeadPass
K5_SetUp	K5_PointLost	Smash
K5_SetUp	K5_PointLost	TipBall
K5_SetUp	Neutral_CounterAttack	Smash
K5_SetUp	Neutral_Block	Block
K5_SetUp	Neutral_Freeball	ForearmPass
K5_SetUp	Neutral_Freeball	OneHandPass
K5_SetUp	Neutral_Freeball	OverHeadPass
K5_SetUp	Neutral_Freeball	TipBall

E Preprocessing individual action labels

After annotation or the recognition of action by a classifier, an extra processing step has to be made to prepare the data as input for the LTS models. As mentioned in section 2.2, actions are recognized with windows of 0.5 sec with 50% overlap. In case annotated data is used, the labels are extracted using this same windowing. This means that for every 0.25 sec, it is known if and what action every player performs.

However, as input for the model, only two data arrays are needed, one showing the starting window of each action and the second contains the player numbers to know who performs the action. Therefore, an extra step is needed to transform the data. This is shown in figure 17, the preprocessing step has to be done before team play is recognized, but it does not matter if it happens as last step of the action recognition, as first step of the team play recognition or in between.

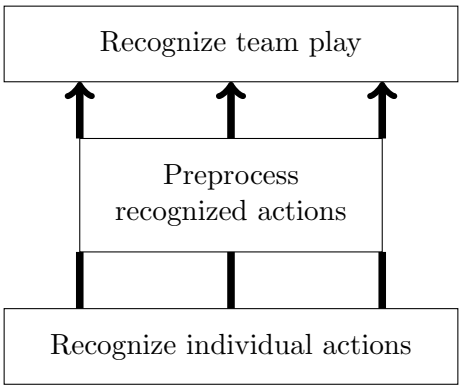


Figure 17: Visualization of part of the layered approach showing the extra preprocessing step. This step has to be performed after the actions are recognized, but before the team play is recognized. It does not matter if this happens as last step of the action recognition or as first step of the team play recognition, therefore it is shown between those layers.

Depending on what exercise is performed by the players, one or two steps need to be taken. In case an exercise is performed in a small group without interference of other groups, one step is needed. If there are two groups interfering, e.g. in a rally-like situation, two steps are performed.

In the first step, the actions performed within a group of players is combined into two data arrays, with information on the actions performed and information on who performs these actions. In the second step, the actions performed by two groups of players are combined into two new data arrays, combining the information of the two groups. This is shown in figure 18 and these steps are described in detail below.

1. In step 1, actions of players are combined in a loop over time. If a new action occurs compared to the previous data window, it is added in the single data array with all actions performed by all players. If there is no change in actions performed compared to the previous data window, the time instance is labelled 'Other' to be able to keep track of time between actions.
   It could be the case that within one window, two new actions occur, e.g. a ForearmPass and an OverHeadPass. In that case, a standard order in which actions occur is used to determine which action was performed first. This standard order is as follows: Serve, UnderHandServe, Block, ForearmPass, OneHandPass, OverHeadPass, Smash and TipBall. So in the given example, the first action is ForearmPass and the second is OverHeadPass.

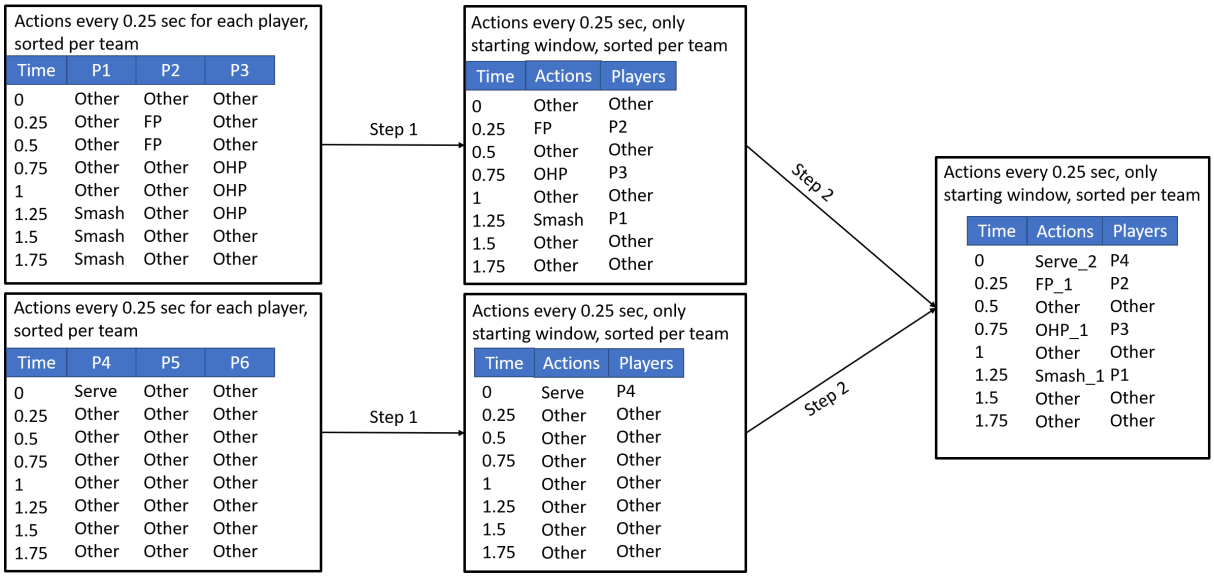


Figure 18: Visualization of the preprocessing steps that need to be taken before the action labels can be used to traverse an LTS model. There are two steps involved, depending on whether two groups of players interfere or the groups perform exercises in one group.

2. In step 2, actions of teams are combined in a loop over time. If a new action occurs, it is added in the new single data array with a notion of the team number. If there is no change, the time instance is labelled 'Other,' to keep track of time between actions.
   It could be that within one window, both teams perform an action. If this happens, it is assumed that the fist action is performed by the same team as the previous action.

F Confusion matrices

Table 15: Confusion matrix with overall accuracy of 90%.

	B	FP	1HP	OHP	Serve	Smash	TB	UHS
B	90	1.43	1.43	1.43	1.43	1.43	1.43	1.43
FP	1.43	90	1.43	1.43	1.43	1.43	1.43	1.43
1HP	1.43	1.43	90	1.43	1.43	1.43	1.43	1.43
OHP	1.43	1.43	1.43	90	1.43	1.43	1.43	1.43
Serve	1.43	1.43	1.43	1.43	90	1.43	1.43	1.43
Smash	1.43	1.43	1.43	1.43	1.43	90	1.43	1.43
TB	1.43	1.43	1.43	1.43	1.43	1.43	90	1.43
UHS	1.43	1.43	1.43	1.43	1.43	1.43	1.43	90

Table 16: Confusion matrix which mixes up the FreeBall actions with a recall of 90%.

	B	FP	1HP	OHP	Serve	Smash	TB	UHS
B	1	0	0	0	0	0	0	0
FP	0	90	3.33	3.33	0	0	3.33	0
1HP	0	3.33	90	3.33	0	0	3.33	0
OHP	0	3.33	3.33	90	0	0	3.33	0
Serve	0	0	0	0	1	0	0	0
Smash	0	0	0	0	0	1	0	0
TB	0	3.33	3.33	3.33	0	0	90	0
UHS	0	0	0	0	0	0	0	1

Table 17: Confusion matrix which mixes up the non-FreeBall actions with a recall of 90%.

	B	FP	1HP	OHP	Serve	Smash	TB	UHS
B	90	0	0	0	3.33	3.33	0	3.33
FP	0	1	0	0	0	0	0	0
1HP	0	0	1	0	0	0	0	0
OHV	0	0	0	1	0	0	0	0
Serve	3.33	0	0	0	90	3.33	0	3.33
Smash	3.33	0	0	0	3.33	90	0	3.33
TB	0	0	0	0	0	0	1	0
UHS	3.33	0	0	0	3.33	3.33	0	90

Table 18: Confusion matrix which confuses both the Freeball actions as well as the non-FreeBall actions, both with a recall of 90%.

	B	FP	1HP	OHP	Serve	Smash	TB	UHS
B	90	0	0	0	3.33	3.33	0	3.33
FP	0	90	3.33	3.33	0	0	3.33	0
1HP	0	3.33	90	3.33	0	0	3.33	0
OHP	0	3.33	3.33	90	0	0	3.33	0
Serve	3.33	0	0	0	90	3.33	0	3.33
Smash	3.33	0	0	0	3.33	90	0	3.33
TB	0	3.33	3.33	3.33	0	0	90	0
UHS	3.33	0	0	0	3.33	3.33	0	90

Table 19: Confusion matrix which confuses actions performed with arms down with a recall of 90%.

	B	FP	1HP	OHP	Serve	Smash	TB	UHS
B	1	0	0	0	0	0	0	0
FP	0	90	5	0	0	0	0	5
1HP	0	5	90	0	0	0	0	5
OHP	0	0	0	1	0	0	0	0
Serve	0	0	0	0	1	0	0	0
Smash	0	0	0	0	0	1	0	0
TB	0	0	0	0	0	0	1	0
UHS	0	5	5	0	0	0	0	90

Table 20: Confusion matrix which confuses actions performed with arms up with a recall of 90%.

	B	FP	1HP	OHP	Serve	Smash	TB	UHS
B	90	0	0	2.5	2.5	2.5	2.5	0
FP	0	1	0	0	0	0	0	0
1HP	0	0	1	0	0	0	0	0
OHP	2.5	0	0	90	2.5	2.5	2.5	0
Serve	2.5	0	0	2.5	90	2.5	2.5	0
Smash	2.5	0	0	2.5	2.5	90	2.5	0
TB	2.5	0	0	2.5	2.5	2.5	90	0
UHS	0	0	0	0	0	0	0	1

Table 21: Confusion matrix which confuses both actions performed with arms down as well as actions performed with arms up, both with a recall of 90%.

	B	FP	1HP	OHP	Serve	Smash	TB	UHS
B	90	0	0	2.5	2.5	2.5	2.5	0
FP	0	90	5	0	0	0	0	5
1HP	0	5	90	0	0	0	0	5
OHP	2.5	0	0	90	2.5	2.5	2.5	0
Serve	2.5	0	0	2.5	90	2.5	2.5	0
Smash	2.5	0	0	2.5	2.5	90	2.5	0
TB	2.5	0	0	2.5	2.5	2.5	90	0
UHS	0	5	5	0	0	0	0	90

Table 22: Confusion matrix with overall accuracy of 80%.

	B	FP	1HP	OHP	Serve	Smash	TB	UHS
B	80	2.86	2.86	2.86	2.86	2.86	2.86	2.86
FP	2.86	80	2.86	2.86	2.86	2.86	2.86	2.86
1HP	2.86	2.86	80	2.86	2.86	2.86	2.86	2.86
OHP	2.86	2.86	2.86	80	2.86	2.86	2.86	2.86
Serve	2.86	2.86	2.86	2.86	80	2.86	2.86	2.86
Smash	2.86	2.86	2.86	2.86	2.86	80	2.86	2.86
TB	2.86	2.86	2.86	2.86	2.86	2.86	80	2.86
UHS	2.86	2.86	2.86	2.86	2.86	2.86	2.86	80

Table 23: Confusion matrix which mixes up the FreeBall actions with a recall of 80%.

	B	FP	1HP	OHP	Serve	Smash	TB	UHS
B	1	0	0	0	0	0	0	0
FP	0	80	6.67	6.67	0	0	6.67	0
1HP	0	6.67	80	6.67	0	0	6.67	0
OHP	0	6.67	6.67	80	0	0	6.67	0
Serve	0	0	0	0	1	0	0	0
Smash	0	0	0	0	0	1	0	0
TB	0	6.67	6.67	6.67	0	0	80	0
UHS	0	0	0	0	0	0	0	1

Table 24: Confusion matrix which mixes up the non-FreeBall actions with a recall of 80%.

	B	FP	1HP	OHP	Serve	Smash	TB	UHS
B	80	0	0	0	6.67	6.67	0	6.67
FP	0	1	0	0	0	0	0	0
1HP	0	0	1	0	0	0	0	0
OHV	0	0	0	1	0	0	0	0
Serve	6.67	0	0	0	80	6.67	0	6.67
Smash	6.67	0	0	0	6.67	80	0	6.67
TB	0	0	0	0	0	0	1	0
UHS	6.67	0	0	0	6.67	6.67	0	80

Table 25: Confusion matrix which confuses both the Freeball actions as well as the non-FreeBall actions, both with a recall of 80%.

	B	FP	1HP	OHP	Serve	Smash	TB	UHS
B	80	0	0	0	6.67	6.67	0	6.67
FP	0	80	6.67	6.67	0	0	6.67	0
1HP	0	6.67	80	6.67	0	0	6.67	0
OHP	0	6.67	6.67	80	0	0	6.67	0
Serve	6.67	0	0	0	80	6.67	0	6.67
Smash	6.67	0	0	0	6.67	80	0	6.67
TB	0	6.67	6.67	6.67	0	0	80	0
UHS	6.67	0	0	0	6.67	6.67	0	80

Table 26: Confusion matrix which confuses actions performed with arms down with a recall of 80%.

	B	FP	1HP	OHP	Serve	Smash	TB	UHS
B	1	0	0	0	0	0	0	0
FP	0	80	10	0	0	0	0	10
1HP	0	10	80	0	0	0	0	10
OHP	0	0	0	1	0	0	0	0
Serve	0	0	0	0	1	0	0	0
Smash	0	0	0	0	0	1	0	0
TB	0	0	0	0	0	0	1	0
UHS	0	10	10	0	0	0	0	80

Table 27: Confusion matrix which confuses actions performed with arms up with a recall of 80%.

	B	FP	1HP	OHP	Serve	Smash	TB	UHS
B	80	0	0	5	5	5	5	0
FP	0	1	0	0	0	0	0	0
1HP	0	0	1	0	0	0	0	0
OHP	5	0	0	80	5	5	5	0
Serve	5	0	0	5	80	5	5	0
Smash	5	0	0	5	5	80	5	0
TB	5	0	0	5	5	5	80	0
UHS	0	0	0	0	0	0	0	1

Table 28: Confusion matrix which confuses both actions performed with arms down as well as actions performed with arms up, both with a recall of 80%.

	B	FP	1HP	OHP	Serve	Smash	TB	UHS
B	80	0	0	5	5	5	5	0
FP	0	80	10	0	0	0	0	10
1HP	0	10	80	0	0	0	0	10
OHP	5	0	0	80	5	5	5	0
Serve	5	0	0	5	80	5	5	0
Smash	5	0	0	5	5	80	5	0
TB	5	0	0	5	5	5	80	0
UHS	0	10	10	0	0	0	0	80