EAI Endorsed Transactions
on Creative Technologies

Research Article ICST.ORG

Using Video Analysis and Machine Learning for Predicting Shot Success in Table Tennis

Lukas Draschkowitz[1,*], Christoph Draschkowitz[1], Helmut Hlavacs[1]

1 University of Vienna, Faculty of Computer Science, Research Group Entertainment Computing

Abstract

Coaching professional ball players has become more and more difficult and requires among other abilities also good tactical knowledge. This paper describes a program that can assist in tactical coaching for table tennis by extracting and analyzing video data of a table tennis game. The here described application automatically extracts essential information from a table tennis match, such as speed, length, height and others, by analyzing a video of that game. It then uses the well known machine learning library "Weka" to learn about the success of a shot. Generalization is tested by using a training and a test set. The program then is able to predict the outcome of shots with high accuracy. This makes it possible to develop and verify tactical suggestions for players as part of an automatic analyzing and coaching tool, completely independent of human interaction.

Keywords: machine learning, sports video analysis, ball tracking, video processing, video information retrieval, video mining, multimedia data mining

Received on 11 August 2014, accepted on 26 March 2015, published on 20 October 2015

doi: 10.4108/eai.20-10-2015.150096

1. Introduction

In the past, numerous attempts have been made to analyze sports games, here mainly using historical data for statistical analysis. In fast paced ball games like table tennis, it may be interesting to understand why some shots are successful, while others are not. This of course highly depends on the respective players, and every player generally will have a unique portfolio of successful and unsuccessful shots. We believe that in this case, machine learning can be employed to understand the individual behavior, and use this for training and tactical analysis for individual players.

Furthermore, player and ball tracking algorithms are getting introduced into literally any kind of ball sports. So far most of tactical game analysis systems like the well known "Hawk Eye" [13] are produced for audiences rather than coaching assistance tools. If not fully automatically done by a computing system, analyzing a game and creating a decent game plan and tactics is very time consuming and quality highly depends on the coach's expert level. In addition to that, statistical analysis demands high computational effort. This brought up the initial idea of our work, which is to fully automatically give tactical advice to players, based on the analysis of their games. This means that the program must be aware of shots and points that are taken and has to be able to extract and make use of as many attributes as possible. Then an algorithm has to detect patterns to make suggestions for the ongoing game.

We introduce a simple algorithm that allows analysis of a table tennis game by two calibrated off-the-shelf USB webcams, and a laptop using Java and the "OpenCV"[1] and "Weka" [23] libraries. The whole program is therefore implemented in Java to be consistent. We show that this low-cost setup already results in excellent results, and it may be expected that professional equipment would result in even higher accuracy. (Figure 1) pictures our approach.

The main source of our approach is given by videos of table tennis games, in our case captured by standard webcams. For particular players, we extract relevant features like length and direction of strokes from the videos and train classifiers from the "Weka" library, discriminating between success and failure. After training, these classifiers are capable of predicting the success of strokes for a particular game and player. By analyzing the respective classifiers, trainers and players can understand which tactical patterns have higher success rates and should therefore be preferred by the players.

*Corresponding Author. E-mail: lukas.draschkowitz@chello.at

[1] http://opencv.org

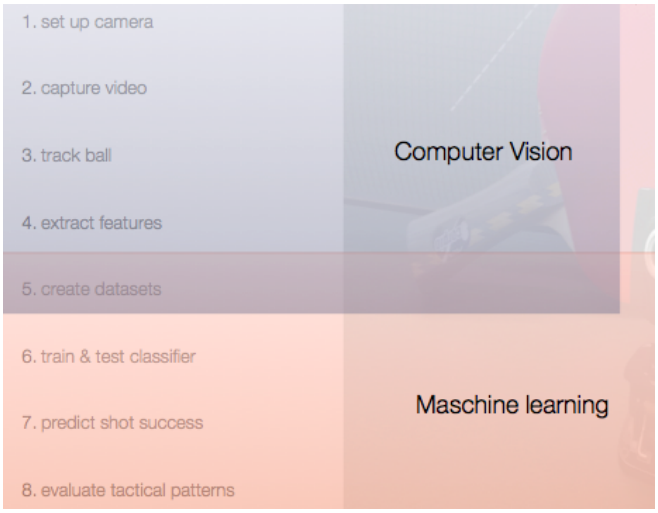

Figure 1. Overview over our approach

Results show a high potential in that area and also lead to new ideas of how to analyze the game and how to use such algorithms in similar but slightly different ways, since we extract a lot of information about the game. This is part of the discussion in the conclusion section.

2. Related Work

An general overview of sports informatics technologies can be found in [1]. Our work relates to various works that detect and track certain objects in sports, e.g., tennis [25], table tennis [6], soccer [10, 17, 24], basketball [3], volleyball [4, 5], or even pool billiard [18, 19]. The best known multi camera technology that works with player and ball tracking is probably the "Hawk Eye" [13], widely used in tennis sports, not only to aid referees but also to collect statistical data (mostly for audiences and TV). [5] analyze volleyball sequences with a 3D model from a single camera input. [15] points out that match analysis is an undervalued coaching tool. A key to obtain an exact position of a detected object is the right camera calibration. [5, 8, 19] show how to extract an object's exact position by detecting court lines in any sport and compare them to models of those courts. [6] extracts data without calibration and uses table and player tracking instead. [4] uses a very similar process to ours of subtracting a background picture to avoid noise. [18] follows our approach in pool billiard and differences between balls by color. On the other hand [9, 10] show different approaches of finding objects with classifiers that work with positive and negative image examples. [5, 25] show how to filter object candidates in various situations and sports. We provided an algorithm that extracts all data automatically, which is quite similar to the requirements of robotics. [20] shows a detection and tracking program for robotic soccer games also implemented using "OpenCV".

In [22] different tactical patterns are assigned to filmed tennis matches and archived for future queries. In [14] table tennis games are analyzed via a stochastic performance diagnostic concept using various state-transition-models. [16] gives an overview of the field of data mining in sports. [21] can be seen as a summary of video data mining, while [2] describes the problems and solutions in multimedia data mining from feature extraction to current multimedia data mining systems. [12] delivers an approach to identify soccer strategies in multi-camera videos.

3. Analyzing Table Tennis Shots

3.1. Optical Ball Tracking and Feature Extraction

The goal was to extract as many attributes as possible from a table tennis game. At the same time, we tried to keep everything as simple and low-cost as possible, without making compromises in terms of quality and accuracy. For recording we used two standard Logitech USB webcams, connected to a notebook. The requirements we set for the cameras were different. Since the room we recorded in was pretty small, we had to use wide angle cameras. Furthermore, we needed an appropriate frame rate since table tennis is a fast sport and balls can reach as much as 180 km/h [11]. That means in worst case the ball moves 80 cm in a second so we can only detect the ball three times on each side of the table with our 60 frames/second frame rate. That is on the lower limit and the system would of course profit a lot on higher frame rates. Up to that we want the program to run really stable and therefore be forgiving whenever the algorithm is unable to detect the ball on a single frame for any reason. That could be simply because a player gets between the ball and the cameras or even be caused by light effects etc. Not detecting the ball is of course in general not bad, since we would not be able to detect it a lot of times simply because the ball is not in the game! Still in our experiments the program was able to deal with very low frame rates as well. On the other hand, the amount of pixels was secondary, since it turned out that high resolutions do not necessarily improve the accuracy of our predictions, but have a negative effect on the analysis performance and might be scaled down anyway. The main camera was located at the side of the table with light coming in from a window behind and an additional light source we installed. Since the height of the ball relative to the table top is an important feature when extracting information from the video, we had to make sure that this camera was at one level with the table so that the table plate seems 2-dimensional. Therefore there is no difference whether the ball bounces on the right or left side of a players side of the table later in the game and extracting the ball's height is quite

easy independent from the occasion. The second camera was fixed on the ceiling above the table, with lights also coming from the top of the room (Figure 2) and was used for depth information to determine the side of the table on which the ball bounces during the game. One of these cameras should be centered to detect the net. In our case it was the side camera we centered, since for economic reasons we only fetched pictures from the upper camera whenever an event was detected to get additional information for the extracted features. The experiments showed that this is sufficient and saves a lot of computing power as well. Still this leads to some smaller imprecisions and analyzing each frame from the upper camera can further increase the robustness of the system and will generally result in more accurate speed attributes.

Figure 2. Red marked are both of the cameras as well as the computer and an additional light source

Due to the fact that the implemented ball detection is based on color, we used an orange ball which is an official competition ball and easier to detect with most backgrounds than white ones, especially with respect to contrast (Figure 3).

In terms of software we used "EvoCam 3.6.9" in a free trial version, which made it possible to start both cameras synchronously. To read and handle the videos and extract the required information we used the well known "OpenCV" library in the fairly new Java Desktop version. By now there might not be all functions translated from the original "OpenCV" since the first release of the "Java" version was in February 2013[2]. Basically our software can be split up into five components. "ReadVideo" extracts frame by frame from the given input videos. "ConvertImage" converts these frames into an easily usable and economic format.

[2]http://opencv.org/opencv-java-api.html

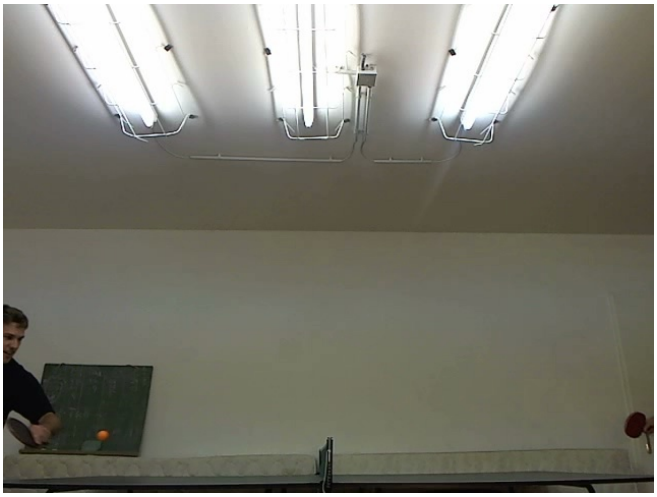

Figure 3. Orange ball tracking. Mostly good contrast with an official game ball of ITTF.

"Detection" detects the ball object, "Data Extraction" extracts all the features and attributes and sends it to the "Weka" component, which in turn is responsible for statistical analysis. The ball detection algorithm works as follows:

Since the ball can change color through speed, light effects, shadows and similars [10], it is not advisable to search for a specific RGB color. Therefore we transform the images to the HSV color space, which is defined by a channel for color, light intensity and color saturation.

The very first frame (acting as the reference frame) is seen as the background picture and should neither contain the ball nor any players. Each of the following frames is compared to to this reference frame and similar objects that are detected in a certain range of the HSV model are subtracted since those contours are obvisously not the ball. Of course, for economic reasons, only the HSV subtracted black and white image of the background picture is saved and then compared to the following substracted black and white frames. The result is a black and white image containing only the ball and some noise, both presented by white pixels.

Those small contours caused by noise (Figure 4) not being part of the actual ball object, are then filtered by size (Figure 5) since most contours appear to be either too big or really small and can therefore by eleminated as ball candidates. Should there still be another ball-like object left (Figure 6), the program then takes the one object that is nearest to the last known ball position. This has proven to work reliable, although there would also be certain algorithms in opencv, to estimate the ball position relative to the last known position if needed.

We then compute the ball position within the image with the help of *image moments* that return the center of the white pixels (representing the ball object). The next step is to extract the game play fully automatically.

Figure 4. All contours, including noise from background.

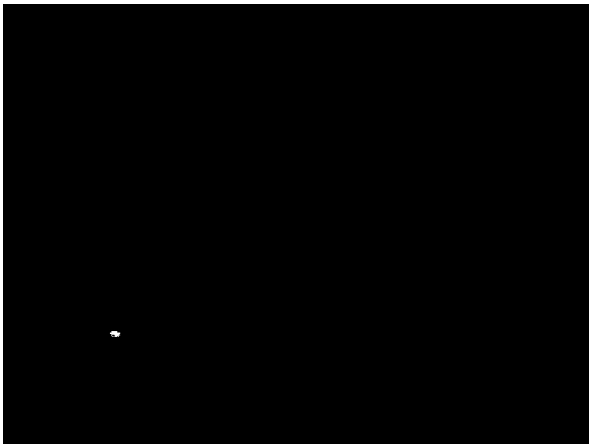

Figure 5. Ball position after filtering, extracted by using image moments.

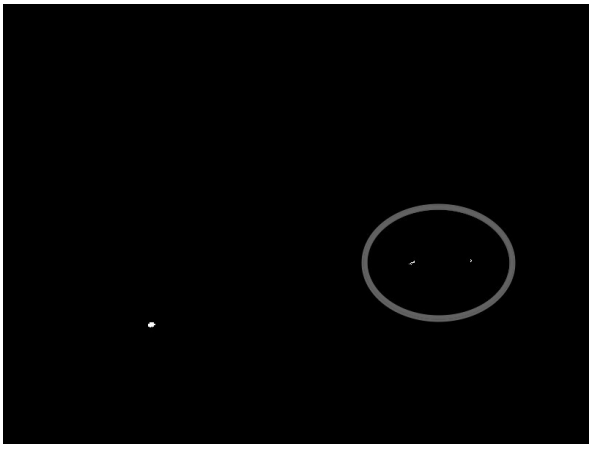

Figure 6. Occurrence of noise, can be filtered through size or prediction.

Therefore the program has two states that define whether or not the ball is in the game at a certain time point. For ongoing rallies the program saves the additional information on whether the ball just moved upwards or downwards, whether it moves left to right or right to left and the ball position itself. For upwards/ downwards detection we are only interested in the case when the ball moves downwards and then changes its vertical direction. Still we also want to know if the ball has already moved upwards before to avoid detection of the same bounce twice (due to high frame rates). This is enough information to detect all of the features defined in Table 1.

Event	**is Ball in game?**	**Feature**
change of *horizontal* direction	yes	stroke, ball in net
change of *vertical* direction	yes	bounce, bounce on wrong side, second bounce
change of *vertical* direction	no	could be service
timeout	yes	winner, out

Table 1. Detecting events

Whenever the ball is not in the game we wait for the algorithm to detect a change of vertical direction. Since the players are forced by the official rules to throw the ball in the air before a service, it can be identified easily. We then change our flag to register that the ball is in the game from now on. If, after that, we detect the ball going downwards and then changing to go upwards, we know there was a bounce. Now we have to decide whether it was a normal bounce or an error, for example a second bounce or the bounce happened to be on the wrong side of the table. As mentioned it is important to know that the ball was once going downwards before we detect the bounce since we do not want the same bounce to be recognized twice. Whenever we detect a horizontal change we have to decide whether we have a new stroke or if the ball has bounced of the net. If we do not detect any event for a certain amount of time we can assume that the last player touching the ball has hit a winner if the ball has already bounced on the other side or an out ball if not. If the end of a stroke is detected (meaning a vertical directional change, a timeout, or an error) all attributes of the last stroke are calculated. Length, winner (=last stroke) or error and direction (which at the same time reveals the player of that stroke) are fairly straight forward to calculate. Additionally we can already tell when a service (=first stroke) and return stroke is coming up. For the side on which the ball bounces up, we use the picture from the upper camera which we handle the same way as for the side camera. Calculating the ball speed, however, is more challenging and incorporates the Euclidean distance between two

ball objects (only 2-dimensional), the distance between camera and table, and the time between the two frames. After calibrating the camera we marked to edges of the table (in orange again), saved the picture and calculated the distance between camera and table at the beginning of each analysis by comparing the picture with the actual (real) table length. Still we could also just work with constant distances between camera and table.

Figure 7. Pictures of a bounce - left: the main camera located on the side - right: the upper camera.

3.2. Machine Learning

By now we extracted the following features for each shot: *hits*, *quality*, *length*, *service*, *speed*, *direction*, *player*, and *point*.

Most of the feature composition is straight forward and features can be represented by numeric attribute values. For binary features we used numeric attributes as well by assigning 0 or 1 in these cases. The "hits" variable tells us how many strokes have been played up to this moment of the rally, starting with zero for the service. If it shows significance in the dataset, it could tell us to shorten up the rallies and to try to force the point decision as quickly as possible. Or on the contrary to keep the ball in the game. Another thing that can influence the success of a table tennis stroke is the previous stroke's *quality*. The quality value is computed from both length and speed of the stroke previous to the one described, by summing up the normalized values. This should quantify the impact of the previous stroke on the current one. Very often a good quality stroke leads to a stroke of poor quality or maybe even a fault. Certainly the flow of the game will also depend on previous strokes.

The length attribute specifies the length of the stroke observed. The service attribute only specifies whether this stroke is a service or not because those strokes will show great differences in their other attribute values. The speed attribute refers to the speed of the ball when hit. "Direction" stores the direction of a shot in the meaning of left or right. This is especially relevant in table tennis where players are very often vulnerable on their backhands, which is what [14] summarize in their results.

Of course each player will behave differently in a game and you want to be able to make tactical suggestions for each of them. Thus strokes are assigned to one of the players. In the end, what we want to know about a given stroke is whether it leads to a point (thus is the last valid stroke in that rally) or not. Therefore the class attribute values to be evaluated are "point" or "no point". This means we want to assign each stroke either to the class of no point instances or to the class of point instances.

We generate data sets where every stroke is represented by an instance described by these seven features. Each of them is further assigned to one of the two class variables. Thus we can work with every classifier that is capable of handling classification problems with binary class values, which in fact applies to most of the standard classifiers in "Weka" [7].

In our experiments we included as many attributes as possible, and later let the classifier decide whether they are relevant or not. Since matches vary, it is generally unknown beforehand, which features will be good discriminators. For example, if the service is an important factor in one game, it could be of no relevance in the next one. As a consequence we tried to obtain as many attributes as possible to include every aspect of the game and make the application work as precise as possible.

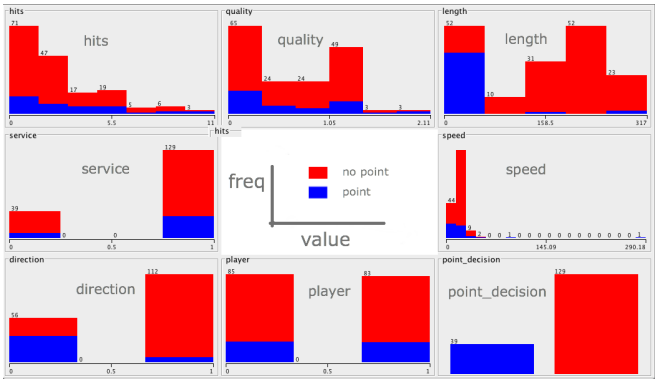

Figure 8. Attribute visualisation for 178 strokes

What we essentially generated at this point are detailed statistics about the captured game. In Figure 8 we can see the strokes captured in our experiment, where the darker (blue) areas indicate point strokes and lighter (red) areas indicate no-point strokes. What we can learn from these statistics is that a high number of point strokes are played with little length and were played to the left hand side. Whereas in this experiment the number of hits in a rally or the speed of a stroke are secondary. Naturally this matter will vary from game to game.

Nonetheless it is not possible to tell whether the above is true for both players by just looking at the statistics, although it seems to be like that in our experiment. We want to know a lot more than we can learn from statistics directly. So what we did, was using

the generated statistics to train a supervised machine learning classifier that analyzes the game automatically.

After splitting the dataset into a training set (66.6 %) and a testing set (33.3 %), we trained a classifier and were able to evaluate unlabeled data, which means we were able to predict the outcome of a stroke by knowing about its features.

4. Experimental Results

We tested our approach by analyzing various videos, the longest being 15 minutes and consisting of 39 rallies and around 160 strokes in total. The Computer Vision component was able to detect all of them, extract the attributes and provide them to the Machine Learning component. Our setup worked with a number of classifiers implemented by "Weka". Although exact numbers depend on the dataset and respectively on the input video, various trees, "Naive Bayes" and "kStar" [23] always classified well over 80 % of the instances in the test set correctly (see Figure 9). In our experiments we even measured 89 % correct classifications for the "Decision Table" classifier.

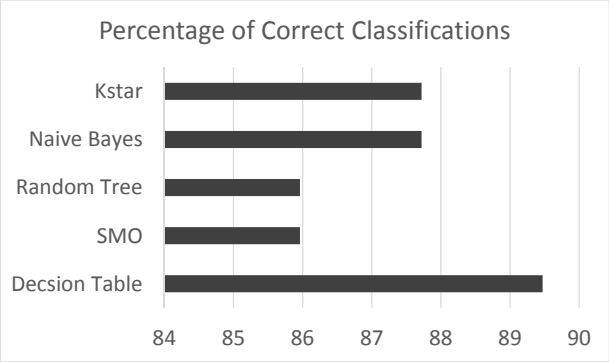

Figure 9. Percent of correctly classified instances for different classifier using 66 % of the dataset as the training set - from a total of 57 instances

Figure 10 shows the percentage of correctly and wrongly classified instances when using a "Random Tree". A fact that makes our classification problem harder to solve is that obviously only a fraction of the strokes will be point strokes.

The "Random Tree" algorithm, a so called "white box" [23] classifier, seems to be a good choice when trying to understand what the classifier has learned. It is a tree that considers n randomly chosen attributes at each node. The decision process can be easily followed by visually inspecting the tree that is the result of the learning process (see Figure 11). This classifier can already be used for real-time situations, where e.g. an on-going game is analyzed on the fly, yielding important tactical information for adapting game tactics in the

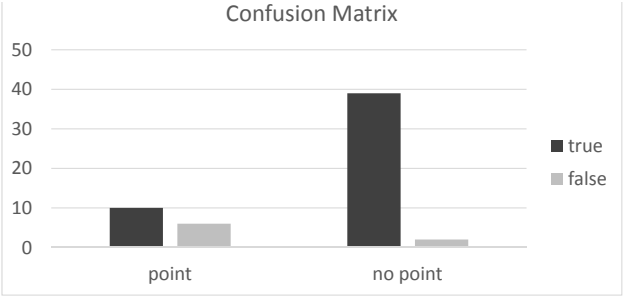

Figure 10. Confusion matrix of Random Tree classifier - dark areas are correctly classified

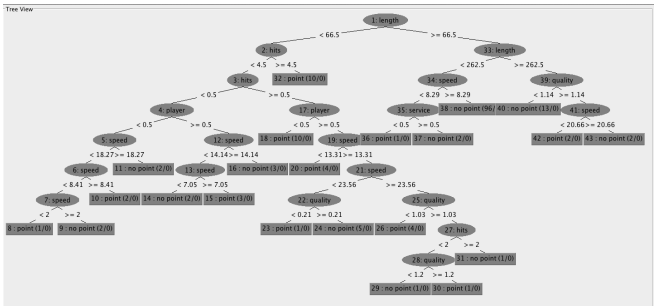

Figure 11. Decision tree (RandomTree): 57 strokes. Size of the tree: 43 nodes.

right away. In many cases however, the tree could be hard to understand. So we still need to find an appealing way to give tactical advice to the players.

With the possibility to evaluate strokes we were able to verify the success for predefined strokes. Each of these strokes would correspond to a certain tactical pattern or game plan. One pattern for instance suggests to play shots of a certain length to the backhand, with high speed, whilst another one suggests to play strokes short and slow to the right side of the table.

This process could be done incrementally as well, in case a real time application is required. In our experiment we processed the video first and classified it afterwards in an off-line manner. To generate artificial strokes that correspond to certain tactical patterns, the mean, the maximum value and the minimum value for each attribute are computed. Thus we take the highest speed observed and leave the rest of the values to be the mean values. Therefore they should have no affect on the stroke success. This way we can generate different patterns by setting some values to max or min and look at their effects. The machine learning system will tell us, if it is likely to become a point stroke or not.

By validating the stroke success for the generated strokes we validated the respective tactical patterns at the same time. We are free to generate and verify an unlimited number of different tactical patterns in variable complexity as long as they are expressible

through our selected set of features. This system is scalable as new game patterns can be added easily. In our experiment the program suggested to "Play short shots to the left side". This is a good strategy because it is what we already saw in our statistics shown in Figure 8. Obviously adding more patterns would have resulted in more suggestions. Another thing that can be thought of, is creating collections of patterns that, if validated all together, represent an entire game plan. This way we could offer more complete suggestions to the players.

Another adaption that can be made, is changing the class value type of the used classifier to numeric. On the one hand this would result in a reduced number of classifiers that can be used. Obviously trees can not handle numeric class values. On the other hand this makes it possible to not only decide whether a game plan is good or not but to tell you precisely in numbers how good they work. So you would have more options to choose from.

Ultimately, analyzing sport games fully automatically makes a huge difference to conventional sport analysis methods. It could be used in practice by coaches and athletes before, during or after competition and also in practice, by giving tactical advice on the fly [7].

5. Conclusions and Future Work

To develop tactical suggestions for table tennis players during a game we analyzed a video and extracted stroke-relevant features. Then we built a model in "Weka" and trained a classifier to predict the outcome of each shot. We showed that 85 % of the shots have been correctly classified in our test set. To make tactical suggestions we let our classifier evaluate different stroke types and verify them by calculating the probability of being a point stroke. In the end we can offer tactical suggestions for each player.

Naturally they will likely not hold for other match-ups between different players or sometimes even for future matches of the same players, because in sports like table tennis, players can have days when they feel good or bad about some kind of shots and one can never infer from one game to the sport as a whole.

Nevertheless more general tactical patterns like "reduce return errors" are likely to improve their universal validity and can therefore be helpful in general and not only in the context of the analyzed game. So, a way to go would be to try to be more universal and to focus on (maybe few) aspects of only one player which corresponds to finding strength and weakness of that player. In order to get accurate universal results though, you will need an universal training set including more than one match. Of course the more general the predictions are, the less accurate they will get and above all over-generalization will lead to uninformative outputs like "play winners" or rather the same score for all precise patterns. Also the removal of attributes in the machine learning system would lead to generalization.

Another helpful idea could be to attach a sensor to a player's racket like it is already done in golf and in "Babolat" tennis rackets. [3] These could give additional information on the ball impact location or the swing speed.

Plenty of these approaches could possibly be added to our machine learning component. The algorithm could also take several of the last strokes or rallies into account, which is similar to the idea of the quality attribute we introduced.

This of course is only a small excursion to the huge amount of possibilities that tactical computer aided coaching can have an impact on.

There is also lot's of room for improvement in terms of quality and calibration of the cameras, although the algorithm performs very well for low-quality equipment already. If we would be able to record the video in a bigger room, and therefore capture not only the table but also the room behind we could introduce another attribute spin, by calculating the ratio of the speed between the beginning of the stroke and its bounce and the speed between the bounce and the next stroke since the ball would increase its speed with topspin and decrease due to slices. This would make the analysis even more accurate since spin is a central element of the game. Additionally we could calculate the speed more accurately by taking both pictures and a 3-dimensional ball movement into account. Obviously we could then use the second camera to detect the ball whenever it is not detected by the main camera, for example when there is some object between camera and ball. This would help making the program even more stable and correct. As already mentioned a higher frame rate would also contribute to better results.

A way to actually generate and evaluate only the right patterns would be to select the two most significant attributes prior, which can be done with the ranker algorithm in "Weka", and create patterns varying only these two attributes. Incremental classification could be used to make the application work as a real-time application. Execution on smart phones would be possible as the application works for low quality videos. Generally more tactical patterns are to be added as well as there is the possibility to add more high level algorithms in the computer vision component to filter noise or predict ball movement better. Furthermore our results are likely to be improved if more attributes (e.g. spin) are added.

Overall, the good results and possible improvements lead us to believe that this kind of game analysis could

[3] http://www.babolat.com/product/tennis/racket/babolat-play-pure-drive-102188

help trainers as well as players to identify strengths and weaknesses or simply help to adjust their game in the tactical level. This might be true for other racket sports as well.

References

[1] Arnold Baca, Peter Dabnichki, Mario Heller, and Philipp Kornfeind. Ubiquitous computing in sports: A review and analysis. *Journal of Sports Sciences*, 27(12):1335–1346, 2009.

[2] Chidansh Amitkumar Bhatt and Mohan S. Kankanhalli. Multimedia data mining: state of the art and challenges. *Multimedia Tools and Applications*, 51(1):35–76, 2011.

[3] B. Chakraborty and S. Meher. Real-time position estimation and tracking of a basketball. In *Signal Processing, Computing and Control (ISPCC), 2012 IEEE International Conference on*, pages 1–6, 2012.

[4] B. Chakraborty and S. Meher. A trajectory-based ball detection and tracking system with applications to shot-type identification in volleyball videos. In *Signal Processing and Communications (SPCOM), 2012 International Conference on*, pages 1–5, 2012.

[5] Hua-Tsung Chen, Wen-Jiin Tsai, Suh-Yin Lee, and Jen-Yu Yu. Ball tracking and 3d trajectory approximation with applications to tactics analysis from single-camera volleyball sequences. *Multimedia Tools and Applications*, 60(3):641–667, 2012.

[6] Wei Chen and Yu-Jin Zhang. Tracking ball and players with applications to highlight ranking of broadcasting table tennis video. In *Computational Engineering in Systems Applications, IMACS Multiconference on*, volume 2, pages 1896–1903, 2006.

[7] Lukas Draschkowitz. Extracting table tennis shots from videos and predicting shot success using machine learning algorithms with weka. B.sc. thesis, University of Vienna, 2013.

[8] Dirk Farin, Jungong Han, and Peter H. N. De. Fast camera-calibration for the analysis of sports sequences. In *In Proceedings IEEE International Conference on Multimedia and Expo (ICME)*, pages 482–485, 2005.

[9] Roque Nimrod Cruz Gómez and Amparo Palomino Merino. Object recognition and tracking in a video sequence.

[10] Dawei Liang, Yang Liu, Qingming Huang, and Wen Gao. A scheme for ball detection and tracking in broadcast soccer video. In Yo-Sung Ho and HyoungJoong Kim, editors, *Advances in Multimedia Information Processing - PCM 2005*, volume 3767 of *Lecture Notes in Computer Science*, pages 864–875. Springer Berlin Heidelberg, 2005.

[11] Thomas Matzke. Auswirkungen der regeländerungen im tischtennis unter besonderer berücksichtigung der medialen wirksamkeit des sports. Master's thesis, Ernst-Moritz-Arndt-Universität Greifswald, 07.09.2008.

[12] Yukihiro Nakamura, Shin Ando, Kenji Aoki, Hiroyuki Mano, and Einoshin Suzuki. Strategy diagram for identifying play strategies in multi-view soccer video data. In Ljupčo Todorovski, Nada Lavrač, and KlausP. Jantke, editors, *Discovery Science*, volume 4265 of *Lecture Notes in Computer Science*, pages 173–184. Springer Berlin Heidelberg, 2006.

[13] N. Owens, C. Harris, and C. Stennett. Hawk-eye tennis system. In *Visual Information Engineering, 2003. VIE 2003. International Conference on*, pages 182–185, 2003.

[14] M. Pfeiffer, H. Zhang, and A. Hohmann. A markov chain model of elite table tennis competition. *International Journal of Sport Science and Coaching*, 5(2):205–222, 2010.

[15] Attilio Sacripanti and Antonio Pasculli. Match analysis an undervalued coaching tool. *arXiv preprint arXiv:1004.1051*, 2010.

[16] Robert P. Schumaker, Osama K. Solieman, and Hsinchun Chen. Sports data mining: The field. In *Sports Data Mining*, volume 26 of *Integrated Series in Information Systems*, pages 1–13. Springer US, 2010.

[17] Xiaofeng Tong, Tao Wang, Wenlong Li, Yimin Zhang, Bo Yang, Fei Wang, Lifeng Sun, and Shiqiang Yang. A three-level scheme for real-time ball tracking. In *Proceedings of the 2007 international conference on Multimedia content analysis and mining*, MCAM'07, pages 161–171, Berlin, Heidelberg, 2007. Springer-Verlag.

[18] Hideaki Uchiyama and H. Saito. AR display of visual aids for supporting pool games by online markerless tracking. In *Artificial Reality and Telexistence, 17th International Conference on*, pages 172–179, 2007.

[19] Hideaki Uchiyama and Hideo Saito. Position estimation of solid balls from handy camera for pool supporting system. In Long-Wen Chang and Wen-Nung Lie, editors, *Advances in Image and Video Technology*, volume 4319 of *Lecture Notes in Computer Science*, pages 393–402. Springer Berlin Heidelberg, 2006.

[20] Wouter Verlinden and David Cnoops. Robocup vision system and fast object tracking with opencv. VAMK (University of Applied Sciences) - Finland, 2009.

[21] V. Vijayakumar and R. Nedunchezhian. A study on video data mining. *International Journal of Multimedia Information Retrieval*, 1(3):153–172, 2012.

[22] Jenny R. Wang, Nandan Prameswaran, Xinguo Yu, Changsheng Xu, and Qi Tian. Archiving tennis video clips based on tactics information. In Kiyoharu Aizawa, Yuichi Nakamura, and ShinÄôichi Satoh, editors, *Advances in Multimedia Information Processing - PCM 2004*, volume 3332 of *Lecture Notes in Computer Science*, pages 314–321. Springer Berlin Heidelberg, 2005.

[23] Ian H. Witten, Eibe Frank, and Mark A. Hall. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, Amsterdam, 3 edition, 2011.

[24] Sunghoon Choi Yongduek, Sunghoon Choi, Yongduek Seo, Hyunwoo Kim, and Ki-sang Hong. Where are the ball and players? soccer game analysis with color-based tracking and image mosaick. In *Proc. ICIAP*, pages 196–203, 1997.

[25] Xinguo Yu, Chern-Horng Sim, J.R. Wang, and Loong-Fah Cheong. A trajectory-based ball detection and tracking algorithm in broadcast tennis video. In *Image Processing, 2004. ICIP '04. 2004 International Conference on*, volume 2, pages 1049–1052 Vol.2, 2004.