SportSense: User Interface for Sketch-Based Spatio-Temporal Team Sports Video Scene Retrieval

Lukas Probst, Ihab Al Kabary, Rufus Lobo, Fabian Rauschenbach, Heiko Schuldt
Dept. of Mathematics and Computer Science
University of Basel, Switzerland
firstname.lastname@unibas.ch

Philipp Seidenschwarz, Martin Rumo
Centre for Technologies in Sports and Medicine
Bern University of Applied Sciences, Switzerland
firstname.lastname@bfh.ch

ABSTRACT
In the last years, various sports have seen significant efforts in collecting large volumes of video, mainly for analytical purposes. These videos are tagged with events that include spatial and temporal information. In order to avoid that game analysts have to manually analyze large video collections, appropriate user interfaces are needed for finding characteristic video scenes. SPORTSENSE is a novel sketch-based video retrieval system tailored to the needs of sports video analysts which include spatio-temporal queries such as player movements, ball trajectories, or interactions between players. We present the user interface of SPORTSENSE that allows users to draw sketches of spatio-temporal events on the field and that visualizes the retrieved video scenes. We introduce the data model of SPORTSENSE, show the sketch-based spatio-temporal retrieval on the annotated videos, and present a qualitative evaluation exhibiting the effectiveness of the SPORTSENSE UI.

ACM Classification Keywords
H.3.3. Information Storage and Retrieval: Information Search and Retrieval; H.5.2. Information Interfaces and Presentation (e.g. HCI): User Interfaces

Author Keywords
Sketch Interfaces, Spatio-Temporal Video Retrieval, Sports

INTRODUCTION
The analysis of videos is more and more becoming important in a large variety of team sports. For this, videos on previous matches are tagged, either manually or automatically, with events that are characterized by spatial (relative to the field) and/or temporal information. Conventionally, analyzing tactical elements of teams in sports videos is a tedious, manual activity of game analysts with the objective of understanding the strengths and weaknesses of the next opponent for optimal preparation and for taking specific tactical decisions. Software packages such as Sportscode[1], myDartfish[2] or Viz Libero[3] help segmenting videos through tagging and help enhancing video content through manually added schematic video overlays. Still, the information is not obtained from a systematic analysis but stays anecdotal.

Yet, in order to facilitate the analysis of large video collections, appropriate user interfaces are needed for advanced video retrieval to find characteristic video scenes. In this paper, we present SPORTSENSE, a novel sketch-based video scene retrieval system tailored to the needs of sports video analysts which include spatio-temporal queries such as the movement of a player, the trajectory of the ball, and/or interactions between players. Given that tracking and event data will become a commodity, an information retrieval tool such as SPORTSENSE will help to bridge the gap between videos and events, and also between anecdotal information / simple statistics and more sophisticated analysis results.

The contribution of this paper is threefold: (i) we introduce a generic model for spatio-temporal tracking data, events, statistics, and match metadata, independent of concrete sports; (ii) we show the user interfaces of SPORTSENSE for sketch-based video scene retrieval in football and ice hockey; and (iii) we present the results of user studies showing the effectiveness and user friendliness of SPORTSENSE.

The remainder of the paper is structured as follows: Section 2 introduces the architecture of SPORTSENSE and Section 3 its generic data schema for video annotations. Section 4 summarizes the spatio-temporal query types of SPORTSENSE and Section 5 presents the UIs for football and ice hockey. The results of our usability study are discussed in Section 6. Section 7 presents related work and Section 8 concludes.

ARCHITECTURE
In this section, we present the architecture of SPORTSENSE (see Figure 1) which consists of three components: a Web client, a REST proxy, and a database backend.

The database stores all available tracking data (e.g., positions of a player), events (e.g., passes) and statistics (e.g., pass statistics). In addition, it stores metadata for all matches.

[1] https://www.hudl.com/elite/sportscode
[2] http://www.dartfish.com/Products
[3] http://www.vizrt.com/products/viz_libero

UISTDA '18, March 11, 2018, Tokyo, Japan

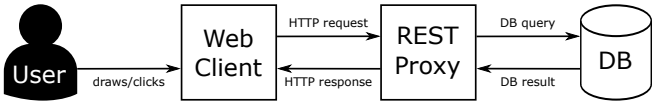

Figure 1. SportSense Architecture[4]

The user can use the sketch-based interface of the Web client for different types of spatio-temporal queries to the database. Moreover, the Web client visualizes the query results (events and associated video scenes) for the user in an intuitive way.

Communication between the Web client and the database is handled via a REST proxy which allows to query the database using a restful API. It receives HTTP requests from the Web client and transforms them into database queries. Subsequently, on receiving the query results from the database, it returns HTTP responses to the Web client.

DATABASE

Spatial databases [4] focus on storing and querying spatial data like GPS tracks. They provide two or three-dimensional spatial index structures and support spatial query operators like distance, intersection, or containment.

The database component of SPORTSENSE is used to store temporal and spatial information, especially tracking and event data. While every tracking data item contains exactly one position, an event data item may even contain multiple positions. For instance, a data item that reflects a pass contains two positions – the start and end location of the pass – which form a line. As we are interested in events that took place in a specific region, SPORTSENSE uses a database with support for two-dimensional containment queries.

The objective of SPORTSENSE is not only to enable a user to retrieve video scenes of a single match, but also to retrieve video scenes of hundreds of matches (e.g., from several seasons) for analytical purposes. Therefore, we need to make sure that the database scales with the volume of tracking and event data. Based on comparisons of spatial database systems by Schmid et al. [7] and Agarwal et al. [1], we have decided to use MongoDB[5] because its feature set supports containment queries with arbitrary polygons and it exhibits a satisfactory performance and scaling behavior. MongoDB is a document database belonging to the NoSQL database family. It stores collections of documents which are binary JSON objects. In contrast to relational databases, MongoDB does not enforce a static schema. Nevertheless, we define three schemata for SPORTSENSE as this simplifies spatio-temporal queries.

Tracking Data and Event Schema

First, we define a generic schema for the tracking and event data. This schema is presented in Listing 1. The *type*, *matchId*, *ts*, *videoTs*, *xyCoords*, *zCoords*, *playerIds* and *teamIds* fields are mandatory. The *type* specifies the type of the event or tracking data, resp. The *matchId* assigns the data item to a match and *ts* denotes the time (in ms) at which the event took place or a tracking position has been captured. Additionally, *videoTs* specifies the video offset (in s). The *xyCoords* array contains

[4] User icon made by Freepik from https://www.flaticon.com

[5] https://www.mongodb.com

```
{  "type" : <type>,
   "matchId": <matchId>,
   "ts": <ts>,
   "videoTs": <videoTs>,
   "xyCoords": [[<x_1>,<y_1>], ... ,[<x_n>,<y_n>]],
   "zCoords": [<z_1>, ... ,<z_n>],
   "playerIds": [<playerId_1>, ... ,<playerId_i>],
   "teamIds": [<teamId_1>, ... ,<teamId_j>],
   "additionalInfo": {
      <key_1>: <value_1>,
      ...
      <key_k>: <value_k> } }
```

Listing 1. Generic Tracking Data and Event Schema

```
{  "type" : "position",
   "matchId": "510350",
   "ts": 1405278313179,
   "videoTs": 313,
   "xyCoords": [[2.367,17.739]],
   "zCoords": [0.0],
   "playerIds": ["A8"],
   "teamIds": ["A"],
   "additionalInfo": {
      "v": [0.409,-1.136,0.0],
      "vabs": 1.207  }   }
```

Listing 2. Sample Tracking Data. Player A8 was located at $\langle 2.367, 17.739\rangle$, had a velocity of $0.409\,^{m}/_{s}$ in x, $-1.136\,^{m}/_{s}$ in y and $0.0\,^{m}/_{s}$ in z direction and an absolute velocity of $1.207\,^{m}/_{s}$.

```
{  "type" : "passEvent",
   "matchId": "510350",
   "ts": 1405279175498,
   "videoTs": 1175,
   "xyCoords": [[-6.041,7.888],[4.816,3.682]],
   "zCoords": [0.0,0.0],
   "playerIds": ["A3","A5"],
   "teamIds": ["A"],
   "additionalInfo": {
      "packing": 3.0  }   }
```

Listing 3. Sample Event. Successful pass from player A3 located at $\langle -6.041, 7.888\rangle$ to player A5 located at $\langle 4.816, 3.682\rangle$ with a packing value of 3.

the planar tracking position or the position(s) where the event took place. The *zCoords* array contains the corresponding z coordinates. The *playerIds* and *teamIds* array contain the involved players and teams.

There are two varying factors in the mandatory part of the schema. The first factor is the number of positions. The *xyCoords* array can contain arbitrarily many positions. Every position in this array is a so called legacy coordinate pair[6], that is, a point in a planar two-dimensional coordinate system. A containment query returns an event if at least one[7] of its positions is contained in the specified region. In order to accelerate the containment queries, we leverage MongoDB's two-dimensional index structure. Since MongoDB does not support three-dimensional indexes, the z coordinate is outsourced to the *zCoords* array and thus separated from the x and y coordinates. However, this is not an issue since the z coordinate is not required for SPORTSENSE's queries as the user interfaces support only two-dimensional sketches. The second factor is the varying number of player and team identifiers in the *playerIds* and *teamIds* array. For instance, a tracking data item involves exactly one player and one team (see Listing 2), a pass event involves two players from a single team (see

[6] https://docs.mongodb.com/manual/geospatial-queries/

[7] It is possible to specify spatial constraints for a specific position in the array by means of leveraging the dot operator (e.g., *xyCoords.2*).

```
{  "type" : <type>,
   "matchId": <matchId>,
   "ts": <ts>,
   "videoTs": <videoTs>,
   "playerIds": [<playerId_1>, ... ,<playerId_i>],
   "teamIds": [<teamId_1>, ... ,<teamId_j>],
   "additionalInfo": {
      <key_1>: <value_1>,
      ...
      <key_k>: <value_k> } }
```

Listing 4. Generic Statistics Schema

```
{  "type" : "passStatistic",
   "matchId": "510350",
   "ts": 1405279825763,
   "videoTs": 2425,
   "playerIds": ["B8"],
   "teamIds": ["B"],
   "additionalInfo": {
      "numPasses" : 12,
      "numInterceptions" : 3,
      "avgPacking" : -0.34  }   }
```

Listing 5. Sample Statistics. Pass statistics for player B2.

Listing 3), an interception event involves two players and two teams and an offside trap event contains only a team but no specific player. Although, the number of positions, players, and teams depend on the *type* their structure remains static.

The *additionalInfo* field enables specifying more event and tracking data type specific information. For instance, the *additionalInfo* field for player position tracking data contains the velocity of the player (see Listing 2) while the *additionalInfo* field for pass events contains the packing value which denotes the number of players of the opposing team that are outplayed by this pass (see Listing 3).

While the fixed structure of the mandatory information enables consistent accesses to common information (e.g., retrieve all events which are located in a specific region of the field by means of a single database query) and the definition of indexes for efficient access, the *additionalInfo* field makes the schema flexible and thus extensible towards arbitrary new event types.

Statistics Schema

Second, we define a generic schema for storing statistics (see Listing 4). The sole but crucial difference to the tracking data and event schema is that the statistics schema has no *xyCoords* and *zCoords* field, i.e., contains no position information, as a statistics data item is not associated with certain positions on the field but corresponds to aggregated values.

The actual statistics values as well as other statistic type specific information (if necessary also positions) are stored in the *additionalInfo* field. For instance, the *additionalInfo* field of pass statistics data items contains the number of passes and interceptions as well as the average packing of a certain player (see Listing 5) or team (if *playerIds* is empty).

Match Metadata Schema

Third, we define a generic match metadata schema depicted in Listing 6 and exemplified in Listing 7. This schema is used to store information about matches. There is exactly one match metadata item for every match.

In contrast to the other two schemata, the generic match metadata schema has a strict structure and no field for flexible

```
{  "matchId": <matchId>,
   "sport": <sport>,
   "fieldSize": [<x>,<y>],
   "date": <date>,
   "competition": <competition>,
   "venue": <venue>,
   "homeTeamId": <homeTeamId>,
   "awayTeamId": <awayTeamId>,
   "homeTeamPlayerIds": [<homeTeamPlayerId_1>, ...],
   "awayTeamPlayerIds": [<awayTeamPlayerId_1>, ...],
   "homeTeamName": <homeTeamName>,
   "awayTeamName": <awayTeamName>,
   "homeTeamPlayerNames": [<homeTeamPlayerName_1>, ...],
   "awayTeamPlayerNames": [<awayTeamPlayerName_1>, ...],
   "videoPath": <videoPath>,
   "homeTeamColor": <homeTeamColor>,
   "awayTeamColor": <awayTeamColor>  }
```

Listing 6. Generic Match Metadata Schema

```
{  "matchId": "510350",
   "sport": "football",
   "fieldSize": [110.0,75.0],
   "date": "2014-07-13T19:00:00Z",
   "competition": "FIFA World Cup 2014",
   "venue": "Maracana",
   "homeTeamId": "A",
   "awayTeamId": "B",
   "homeTeamPlayerIds": ["A1", ...],
   "awayTeamPlayerIds": ["B1", ...],
   "homeTeamName": "Germany",
   "awayTeamName": "Argentina",
   "homeTeamPlayerNames": ["Manuel Neuer", ...],
   "awayTeamPlayerNames": ["Sergio Romero", ...],
   "videoPath": "./path/to/video.mp4",
   "homeTeamColor": "white",
   "awayTeamColor": "blue"  }
```

Listing 7. Sample Match Metadata. FIFA World Cup final 2014.

information. The *matchId* is the identifier of the match and referred in the tracking, event, and statistics data items. The *sport* field specifies the discipline and *fieldSize* defines the size of the sports field. The subsequent fields inform when (*date*), in which context (*competition*) and where (*venue*) the match took place. The *videoPath* specifies the path where a video of the match can be found. The *homeTeamColor* and the *awayTeamColor* specify colors which can be used for visualizations. The remaining fields can be used to map the player and team identifiers used in the tracking, event, and statistics data items to player and team names. For instance, in Listing 7 team identifier "A" is mapped to "Germany" and player identifier "A1" is mapped to "Manuel Neuer".

SPATIO-TEMPORAL QUERIES

In this section, we present four types of sketch-based spatio-temporal queries for retrieving team sports video scenes. All query types have in common that they are not tailored to one sport discipline, but can be applied to different team sports.

Region Queries

The most straightforward query type is to retrieve all events that are located in a specific region on the field [2]. In football, this query type can be used for instance if the user is interested in all shots on goal that originated from outside of the penalty box. The region for such a query can be defined by sketching a free form on the field (see Figure 2).

The results of such a region query can be further refined by specifying the event type, the player, and/or the team. For spatio-temporal queries, also a time interval is specified.

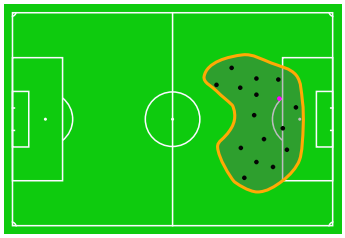

Figure 2. Sample Region Query Sketch in Football

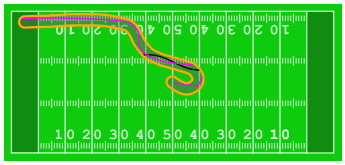

Figure 3. Sample Motion Query Sketch in American Football

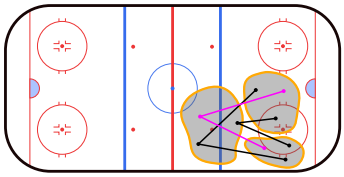

Figure 4. Sample Forward Event Cascade Query Sketch in Ice Hockey

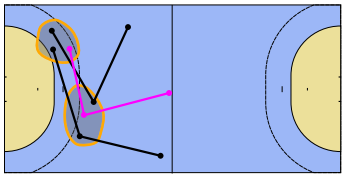

Figure 5. Sample Reverse Event Cascade Query Sketch in Handball

Motion Queries

Another query type is to retrieve video scenes in which the movement of the ball or a player follows a specific path (i.e., is within a certain area) [2]. In American football, this query type can be used for instance to retrieve a certain play that has led to a touchdown after a long run. Such a query can be defined by drawing the movement of the ball, i.e., its motion path, on the field (see Figure 3).

Forward Event Cascade Queries

Assume that a user wants to retrieve a video scene based on a specific sequence of events she remembers, such as a sequence of passes in the bottom right part of an ice hockey field. In this case, using the region query would not be sufficient. Instead, the user has to specify a query that includes the temporal sequence of events in the form of a forward event cascade [2]. In order to specify such a query, the user can subsequently draw a region for every event in the sequence (see Figure 4).

Reverse Event Cascade Queries

Assume that a user wants to analyze which sequence of events has led to a specific event she remembers. For instance, in handball a coach might want to analyze which pass patterns have led to a goal from the right back area. In such cases, the user does usually not know the full sequence of events in advance. Instead, the event cascade has to be built incrementally and interactively in reverse order. To perform such a reverse event cascade query, the user can draw the region of the last event and retrieve the preceding events by expanding all results [3]. This can be continued by repeating the expand action. To shrink the set of event cascades, the user can define additional regions prior to every expand action (see Figure 5).

USER INTERFACES

The Web client of SPORTSENSE and thus its user interface is different for every sport discipline. Currently, we provide an UI for football and one for ice hockey.

SportSense-Football

SPORTSENSE-FOOTBALL is the football UI of SPORTSENSE. This Web client is implemented in TypeScript and uses Bootstrap[8], Bootstrap Multiselect[9], jQuery[10], jQuery UI[11], DataTables[12] and heatmap.js[13]. Figure 6 shows a screenshot captured after a successful motion query. The user interface can be partitioned into four areas: The navigation, the field, the result list, and the video pane.

The navigation on the left hand side contains various drop-downs and sliders grouped into four categories and adapts to the current query type. The drop-downs and sliders can be used to set filters, to change between the query types, to set query type specific parameters, and to induce queries.

The football field at the bottom serves two purposes. In drawing mode, the user can sketch on the field in order to specify a region for a query. SPORTSENSE-FOOTBALL supports drawing rectangles, circles, and free form sketches. Moreover, SPORTSENSE-FOOTBALL allows to mirror the sketch for region and motion queries to handle different playing directions. Besides sketching, the field is used to visualize the query results. In selection mode, the user can select a result of a region or cascade query by clicking on the corresponding visualization. The user can switch between the modes by clicking on the buttons above the field.

Alternatively, the user can select a result in the result list at the right hand side. The result list enables the user to sort the results by different criteria (e.g., by the *matchId*, the duration and the covered distance in case of a motion query).

Once a user has selected a result on the field or in the result list, the video pane at the top starts playing the corresponding scene. The slider below the video can be used to specify how much ahead of the result the video should start.

Besides the four spatio-temporal query types introduced before, SPORTSENSE-FOOTBALL supports two purely temporal queries. More precisely, the user can retrieve the heatmap for a certain player and interval (e.g., for player A3 between minute 10 and 25) as well as the pass statistics at a certain time (e.g., at minute 42). The time boundaries for the heatmap interval and the point of time for the pass statistics can be specified by means of a slider.

[8] http://getbootstrap.com
[9] https://github.com/davidstutz/bootstrap-multiselect
[10] https://jquery.com
[11] https://jqueryui.com
[12] https://datatables.net
[13] https://www.patrick-wied.at/static/heatmapjs

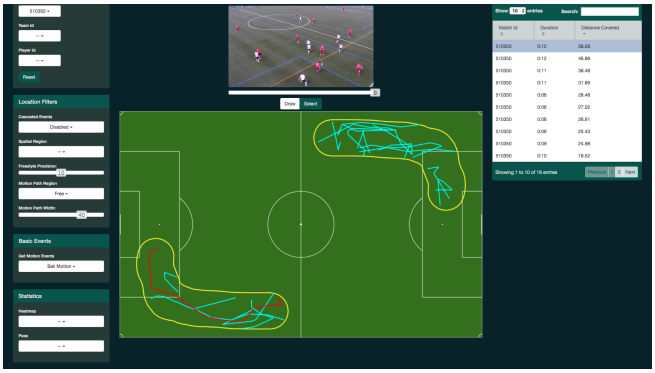

Figure 6. Mirrored Motion Query in SportSense-Football

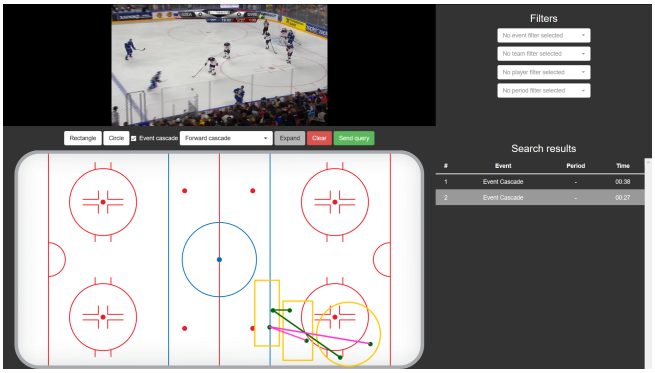

Figure 7. Forward Event Cascade Query in SportSense-Icehockey

SportSense-Icehockey

The ice hockey Web client of SPORTSENSE, called SPORTSENSE-ICEHOCKEY, is implemented in TypeScript and uses Bootstrap, Bootstrap Select[14], fabric.js[15], jQuery and video.js[16]. A screenshot showing the result of a successful forward event cascade query is given in Figure 7.

The user interface is partitioned into five areas. Three of them – the field, the result list, and the video pane – are very similar to those in SPORTSENSE-FOOTBALL. However, so far, SPORTSENSE-ICEHOCKEY lacks a few features that have already been implemented in SPORTSENSE-FOOTBALL. The field does not yet support free sketches and the mirror function, the result list cannot be sorted and there is no slide to control the video offset. Moreover, since tracking data, statistics and heatmaps for ice hockey are not yet available in both quantity and detail as in football, motion, heatmap and statistics queries are not supported but will eventually be added.

In contrast to a large navigation area with many drop-downs and sliders as in SPORTSENSE-FOOTBALL, SPORTSENSE-ICEHOCKEY has a dedicated filter area at the top right. The actual navigation for specifying the query type, entering the drawing mode and inducing queries is between the video pane and the field area. This navigation contains much less elements and hides elements that are not required for the current query type. For instance, the drop-down for switching between the forward and the reverse event cascade query type is only visible if the event cascade box is ticked. If further input is required (e.g., to select the types for expanding a reverse event cascade query) a pop-up is shown to the user.

EVALUATION

We have run separate user studies for SPORTSENSE-FOOTBALL and SPORTSENSE-ICEHOCKEY to evaluate the usability of both user interfaces. The group of participants in both user studies has been different but not completely disjoint. SPORTSENSE-FOOTBALL has been evaluated by 20 participants and SPORTSENSE-ICEHOCKEY has been evaluated by twelve participants. In both user studies, the participants have been guided by the developer of the system.

To evaluate a user interface, each participant had to rate the different query types as well as the overall application w.r.t. several questions on an evaluation sheet (see left part of Figure 8). The rating schema we have used was: 1 for "very bad", 2 for "bad", 3 for "fair", 4 for "good" and 5 for "very good". Alternatively, a participant has been able to abstain from answering a question. Unanswered questions are ignored in the result interpretation. Moreover, the participants have been able to give additional feedback as free text. The evaluation sheet for SPORTSENSE-ICEHOCKEY has been a subset of the evaluation sheet for SPORTSENSE-FOOTBALL as SPORTSENSE-ICEHOCKEY only supports a subset of the query types of SPORTSENSE-FOOTBALL.

For every spatio-temporal query type, all participants were given a task (such as, for instance, retrieve all video scenes with a goal shot by a certain player) they had to complete before answering the corresponding questions. The tasks have been the same for all participants but different in SPORTSENSE-FOOTBALL and SPORTSENSE-ICEHOCKEY.

Figure 8 shows the average rating as well as the standard deviation for all answers. The majority of the average ratings is between 4 (good) and 5 (very good). This confirms the high level of user satisfaction in both interfaces.

The two outliers for SPORTSENSE-ICEHOCKEY – the ease of use (*2.917*) and the spatial aspects (*3.5*) for the forward cascade – can be explained by the fact that the task for the forward event cascade query type has been quite hard. The participants had to sketch very narrow and nearby regions for an event sequence – while these regions have not been allowed to overlap[17]. We suppose that this is also the reason why SPORTSENSE-ICEHOCKEY has a lower overall ease of use rating (*3.75*) than SPORTSENSE-FOOTBALL (*4.0*).

Although the overall ease of use rating in the SPORTSENSE-FOOTBALL user study is good at large, the fact that the ease of use ratings for the spatio-temporal query types (*4.0*, *3.8*, *3.3* and *3.95*) and the overall intuitiveness (*3.55*) are among the lowest results in the SPORTSENSE-FOOTBALL user study indicates that there is still potential for further improvement. This is also reflected in the free text feedback provided by the participants. While in the feedback comments on the SPORTSENSE-FOOTBALL user study *45 %* of the participants

[14] https://github.com/silviomoreto/bootstrap-select
[15] http://fabricjs.com
[16] http://videojs.com

[17] We have fixed this issue already in the latest version.

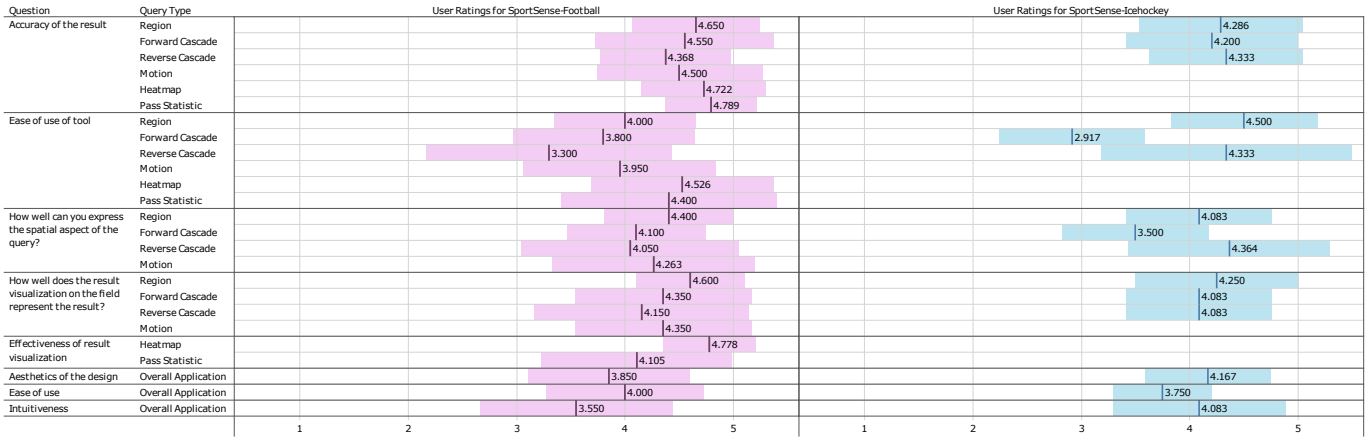

Figure 8. Evaluation Results. The graph shows the average and the standard deviation.

criticized the intuitiveness or ease of use, only *25 %* of the participants did so in the SPORTSENSE-ICEHOCKEY user study. Moreover, *30 %* of the participants raised issues with the scrolling in the navigation of SPORTSENSE-FOOTBALL[18]. In consequence, we deduce that SPORTSENSE users prefer a more lightweight navigation concept like in SPORTSENSE-ICEHOCKEY. For that reason, we plan to implement a more lightweight navigation in our future user interfaces.

RELATED WORK

Systems for visually analyzing matches in team sports rely on event data that is either provided manually (e.g., by companies like Opta [5]) or automatically. The latter is done by extracting raw position data from sensors deployed on the players [10] or from videos and analyzing theses position data, i.e., detecting events and calculating statistics, offline or in real-time [6].

The work of Stein et al. [9] focuses on the analysis of player and ball motion on the field and the visualization of the analysis results, as an overlay to the video of the match. Chalkboarding [8] is a spatio-temporal approach that focuses on retrieving particular plays in sports. The system offers the following query types: (i) exemplar-based queries, (ii) manipulated (by a user) exemplar-based queries, and (iii) drawing-based queries. Since Chalkboarding does not rely on explicit event data, event cascades like in SPORTSENSE cannot be provided.

CONCLUSION & FUTURE WORK

In this paper, we have introduced SPORTSENSE, a system for retrieving video scenes of team sports matches by means of different types of spatio-temporal queries. Moreover, we have defined generic schemata for storing tracking data, events, statistics, and match metadata of arbitrary team sports. We have presented two user interfaces, SPORTSENSE-FOOTBALL and SPORTSENSE-ICEHOCKEY, tailored to particular sports, that enable the user to intuitively define spatio-temporal queries to retrieve video scenes by drawing sketches on the field. Our user studies have confirmed the effectiveness of the sketch-based retrieval approach in sports and a high level of user satisfaction in both user interfaces.

In our future work, we plan to leverage the insights acquired from the evaluation to develop a unified SPORTSENSE Web client that combines the best of SPORTSENSE-FOOTBALL and SPORTSENSE-ICEHOCKEY and that will serve as a user interface for arbitrary team sports. Moreover, we aim to introduce new sketch-based spatio-temporal query types for retrieving video scenes based on (multi-player) events, trajectories, and specific team behavior patterns (i.e., the team's tactics).

[18] The UI has been evaluated with a lower resolution than in Figure 6.

ACKNOWLEDGMENTS

This work has been partly supported by the Hasler Foundation (project StreamTeam, contract no. 16074).

REFERENCES

1. S. Agarwal and KS Rajan. 2015. Performance Analysis of MongoDB Vs. PostGIS/PostGreSQL Databases For Line Intersection and Point Containment Spatial Queries. In *Proc. of FOSS4G*.
2. I. Al Kabary and H. Schuldt. 2013. Towards Sketch-based Motion Queries in Sports Videos. In *Proc. of ISM*.
3. I. Al Kabary and H. Schuldt. 2014. Enhancing Sketch-based Sport Video Retrieval by Suggesting Relevant Motion Paths. In *Proc. of SIGIR*.
4. R. Güting. 1994. An Introduction to Spatial Database Systems. *VLDB Journal* 3, 4 (1994).
5. Opta Sports. 2017. The Data Collection Process. http://www.optasports.com/about/how-we-do-it/the-data-collection-process.aspx. (2017).
6. L. Probst, et al. 2017. Demo: Real-Time Football Analysis with StreamTeam. In *Proc. of DEBS*.
7. S. Schmid, et al. 2015. Performance investigation of selected SQL and NoSQL databases. In *Proc. of AGILE*.
8. L. Sha, et al. 2016. Chalkboarding: A New Spatiotemporal Query Paradigm for Sports Play Retrieval. In *Proc. of IUI*.
9. M. Stein, et al. 2018. Bring It to the Pitch: Combining Video and Movement Data to Enhance Team Sport Analysis. *Trans. Vis. Comput. Graphics* 24, 1 (2018).
10. A. Stelzer, et al. 2004. Concept and Application of LPM – A Novel 3-D Local Position Measurement System. *Trans. Microw. Theory Techn.* 52, 12 (2004).