A User Interface for Spatio-Temporal 'Eventually' Queries using Gamepad

Vineetha Bettaiah
University of Alabama in Huntsville
Department of Computer Science
Huntsville, USA
vb0003@cs.uah.edu

Ramazan S. Aygün
University of Alabama in Huntsville
Department of Compute Science
Huntsville, USA
raygun@cs.uah.edu

***Abstract*— Video databases have both spatial and temporal components. Querying and retrieval of spatio-temporal content is a challenging task due to lack of simple user interfaces. In this paper, we propose a system to allow the user to interactively build "eventually" queries in video databases. In eventually queries, the user just needs to provide the starting state (or information) and the ending state without providing the intermediate states. This helps the user setup queries without knowing all details. Our system uses a methodology similar to the one in gaming. Queries are built by displaying natural videos based on gamepad commands rather than on a graphical interface using a mouse or a keyboard. The system uses a semantic sequence state graph (S^3G) to search the database. The system is applied on a tennis video database. This paper, proposes a novel spatio-temporal query and retrieval system with user friendly interface for developing "eventually" type spatio-temporal queries using gamepad.**

Keywords-Video querying and retrieval; interactive query interface; eventually queries.

I. INTRODUCTION

The video databases have both spatial and temporal dimension. The process of retrieving spatio-temporal objects and events that span space and time domains is known as spatio-temporal query. The design of a good spatio-temporal query system should consider representation of the spatio-temporal information, query building, and simplicity. The *representation* to model the spatio-temporal system must be sophisticated enough to capture the semantic contents. Such a system should be able to represent objects, events, and changes in the data. It may be hard to build a spatio-temporal query instantaneously. Therefore, the system should allow the user to build the spatio-temporal queries incrementally to retrieve one or a sequence of many events which, cause the specified action. The spatio-temporal query system must also be simple enough to be used by a general-purpose user and should not require them to know the internal representation of the database.

Significant effort has been made on querying spatio-temporal databases and many of the approaches are based on developing new languages or extending the existing query languages such as SQL [11] or developing interfaces for the user to build a spatio-temporal query. STQL (Spatio-Temporal Query Language) [7] demonstrates how SQL can be extended to query spatial objects that change over time. It extends SQL by adding features like a set of spatio-temporal predicates such as *disjoint, meet, overlap, coveredBy, covers, inside, contains*, and *equal*. 2198 predicates are identified between two evolving regions. Such a large number of predicates make it practically impossible to name the predicates as well as their utilization by the user. Jain *et al.* [1] uses pattern matching properties of SQL to express spatio-temporal queries. Since the data is represented as strings based on a grammar, it is possible to apply pattern matching techniques. In [3], "conceptual-neighborhood-graph" (or "closest-topological-relationship-graph") is developed based on spatio-temporal relationships like overlap, meet. This graph is used to retrieve spatial objects that changes over time [4].

Besides query languages based on SQL, visual query languages have also been proposed to query spatio-temporal data since the data has at least spatial component. Icons are usually used to represent objects. Lvis supports querying moving objects [5][6]. Query-By-Trace [8], Visual Interactive Query Interface [10] and Visual Query system S-TVQL [9] are other examples of visual querying interfaces for spatio-temporal content. All the above approaches present difficulty in analyzing the query for the novice user. Naik [2] provides a user interface for querying tennis video databases. The user chooses (or click) the locations of players and the ball on the available interface for each instance in the query. However, point-and-click approach using a graphical court view is tedious and does not provide an intuitive method of building queries.

In this paper, we focus on the *eventually* operator in temporal logic. If the user is interested in the next available state from a current state, we basically call it as a *'next'* query. If the user is interested in whether a state is reachable from a current state, we call it as an *'eventually'* query. Eventually type query result allows the user to visualize all intermediate steps to reach the given state. These types of query allow the user to specify two states and view all intermediate events and states between them and also relieve the user from trying to recollect every possible next event to query in case of "next" query. In other words, the user does not need to specify all intermediate steps. It is possible that the user may not know or not interested in intermediate steps.

Since video databases may require spatio-temporal queries that include three dimensions, it is hard to build such queries without a proper user interface. Especially,

 ISBN: 978-1-61208-129-8

incorporating temporal dimension is difficult. We observe that one of the common environments in which users provide spatio-temporal inputs to the system is the environment of video games. In these eniveronments, a player (or a user) provides spatio-temporal inputs of objects using a gamepad. In our system, the queries are built by displaying natural videos based on gamepad commands rather than on a graphical interface. There are three components in the system: building the query, searching and retrieval of clips, and displaying query result. Semantic sequence state graph (S^3G) is used to search the database. A query is built incrementally as a sequence of queries. Though this paper describes the query building process using "Eventually" type queries, the process is applicable to build other types of spatio-temporal queries. We illustrate the system on tennis videos.

Our paper is organized as follows. The following section provides background about the database and indexing. Section III describes how a gamepad is used for "eventually" queries. Our examples and illustrations are provided in Section IV. The last section concludes our paper.

II. BACKGROUND

In this section, we provide information about our semantic modeling and retrieval system (SMART) and our semantic sequence state graph (S^3G) for indexing and retrieval of videos from a tennis video database.

A. SMART

The semantic content of a video corresponds to high-level information in the video. SMART [1] models objects, events, sequence of events and the resulting spatio-temporal interactions among objects in the video. A sample application on tennis videos that utilizes SMART is developed for modeling and retrieval of semantic contents in a tennis video. The semantic contents of a tennis video are modeled using a set of objects, a set of events, a set of locations on the court besides a set of camera views and a set of production rules (grammar) which, are given in [1].

Objects: The set of objects Σ_O contains three objects: the ball b, the first player U and the second player V:

$\Sigma_O = \{U, V, b\}$.

Events: The set of events Σ_E contains two distinct events: the forehand shot F, and the backhand shot B:

$\Sigma_E = \{F, B\}$.

Locations: The tennis court is divided into 13 non overlapping regions including the net N as shown in Figure 1. The set of locations Σ_L includes all these 13 regions:

$\Sigma_L = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, N\}$

The production rules are used to encode the semantic contents of the videos as a set of strings. Each video clip is represented with one string.

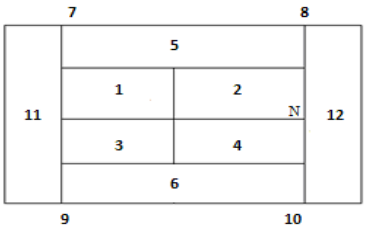

Figure 1. Regions of the tennis court.

B. S^3G - Semantic sequence state graph.

While SMART [1] represents the semantic information in a video as a set of strings, the semantic sequence state graph (S^3G) [2] represents the same information in the form of a graph. In tennis video, each object (ball, $player_1$, $player_2$) can be in any of the 13 possible locations. Therefore, theoretically, there are a maximum of 13^3 patterns of assigning 3 objects to 13 locations. Each assignment pattern defines a unique state in S^3G, and the maximum number of states in S^3G is less than 13^3 due to game constraints. S^3G also reduces the number of states by maintaining only states that are present in the video database. An event from the set of all possible events Σ = {F_1 ($player_1$ hits forehand), F_2 ($player_2$ hits forehand), B_1 ($player_1$ hits backhand), B_2 ($player_2$ hits backhand)} makes the objects move causing state-to-state transitions. Note that S^3G may have cycles as a state may be visited many times during the game. Thus, in S^3G, the semantic information of a clip is represented by a sequence of states and transitions, starting from one of the 8 possible states (four serve locations and two players). The semantic information of all clips, together, represents the semantic information of the video.

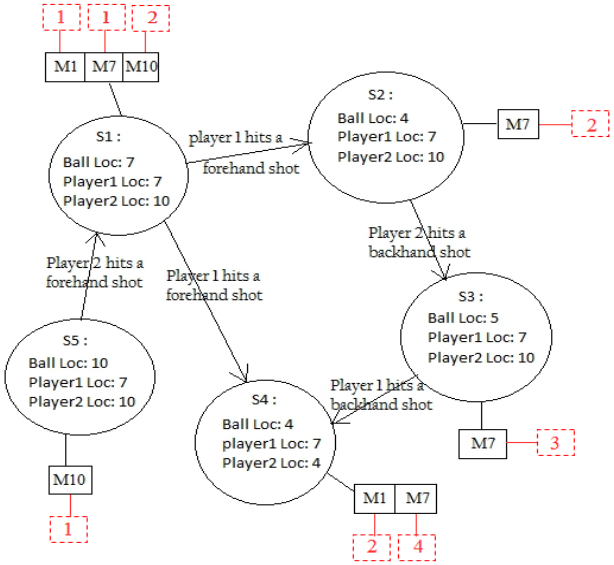

Figure 2. Construction of S^3G from SMART string data.

Example: S^3G in Figure 2 is built for three clips of a video: M1 = A[U] C[U7b7V10 b4 V4], M7 = A[U] C[U7b7V10 b4 BV10 b5 BU7 b4 V4], and M10 = A[V] C[U7b10V10 b7

]. The letter *A* represents a close-view of a player, while letter *C* represents court-view. The sub-string for court-view is indexed by S^3G. In M1, $Player_1$ (U) serves an ace from location 7 as Player2 moves to location 4 to receive, but $Player_2$ fails. In M7, $Player_1$ serves from position 7 again; Player2 responds from location 5 with a backhand shot and the ball hits location 5; and $Player_1$ hits with a backhand shot at location 7 and the ball goes to location 4. In M10, $Player_2$ serves the ball from location 10 and the ball hits location 7. For example, nodes S_1, S_2, S_3, S_4 and the state transition from $S_1 \rightarrow S_2$, $S_2 \rightarrow S_3$, $S_3 \rightarrow S_4$, represent M7.

The semantic sequence state graph, as described above, has a limitation. From the string representation of M1, it is clear that the temporal order of states in M1 is (S_1, S_4) indicating that S_1 is the first state and S_4 is the second state. Similarly, temporal orders of states in M7 and in M10 are (S_1, S_2, S_3, S_4) and (S_5, S_1), respectively. The initial S^3G did not contain temporal orders of states in various clips. The lack of temporal order information could lead to the retrieval of clips that do not satisfy the criteria specified by the query. For example, if a query specifies a direct transition from S_1 to S_4 through a forehand shot from $player_1$, the system will retrieve two clips M1 and M7 because both clips are attached to S_1 as well as S_4. However, note that in M7 there is no direct transition from S_1 to S_4. Retrieval of incorrect clips also occurs when a state is visited multiple times during graph traversal. This is possible due to the occurrence of several instances of a same state in a single clip. This limitation can easily be resolved by attaching to each clip a list of temporal orders (ranks) of the state as shown in Figure 2 by dotted red squares. With the enhanced S^3G, the retrieval of clips is a two step process. In step 1, clips common to all states involved in the query are selected. In step 2, clips in which states do not satisfy the temporal order constraints are deleted. In addition, timings of these ranks are stored in the database. For example, time of S_1 in clip M1 may be at 127^{th} second, time of S_4 in clip M1 may be at 129^{th} second. Hence the event, $player_1$ hits a forehand shot from S_1 to S_4 starts from 127^{th} second and ends at 129^{th} second. These are represented as *StartTime* and *EndTime* respectively for each event

III. INTERACTIVE RETRIEVAL OF VIDEO CLIPS USING GAMEPAD.

A user-friendly interface is needed to get spatio-temporal input from the user. There were some approaches in the past to handle spatio-temporal queries. For example, Jain *et al.* [1] developed a user interface with drop-down menus to get input from the user. These inputs are first used to develop a SQL string pattern that can be mapped to spatio-temporal expressions. Then a SQL query is built. On the other hand, Naik *et al.* [2] develop a graphical user interface with a mouse point-and-click approach using court-view for tennis. Basically, the user needs to choose an object and then click where the object should be on the court. After a state is built, the next or future states are built in a similar fashion. In this paper, we propose a better user interface than the previous approaches: a) we provide a court-view (from a tennis video) to the user for interactions and b) the inputs are obtained using a gamepad. The gamepad provides multiple buttons that enable switching objects and locating them on the court-view.

A. Features of Gamepad

The gamepad provides a set of interactions as shown in Figure 3 to build spatio-temporal queries: an 8-way switch (A) for directional controls, a set of 10 buttons, B={b_1, b_2, …, b_{10}}, a record (R) button to store the current information, and a record/search (R/S) button.

Figure 3. Gamepad features

B. Mapping Gamepad Input to Semantic Information in S^3G

The critical part of querying is to map the input from the gamepad to semantic information for tennis video to be used for retrieval. Each feature or button (e.g., button b_1) of gamepad produces a numerical input when pressed. These numerical inputs are mapped to semantic information. For example, the 8-way switch has eight active positions corresponding to eight directions. The switch generates unique code for each direction. Thus it is used to input the direction of movement of the object on the tennis court while building the query. Based on the input from switch A and other features of the gamepad, our system builds the query and provides as input to the retrieval system based. Every node in S^3G, represents particular semantic information about ball-player location. The search process matches the semantic information specified by the query with the semantic information of the video represented by S^3G. If a match is found, clips attached to the matching nodes are retrieved from the database and displayed to the user.

C. Algorithm for Eventually Query

The system retrieves the desired video clips from the database using an approach that has three main steps. First, spatio-temporal queries are built interactively using the gamepad. Secondly, a search process is initiated where clips that include the states of interest are identified by applying queries to our graph based indexing structure

 ISBN: 978-1-61208-129-8

(S^3G) and fetched from the database. Finally, the system provides a visual display of the query results by displaying all relevant events in real time using original video in the database. An implementation of "Eventually" query is given by following algorithm.

```
void Eventually_Query ( )
begin
 // I_Lb, I_L1, I_L2 represents location of ball, player_1, player_2
respectively of the InitialState I_S
// F_Lb, F_L1, F_L2 represents location of ball, player_1, player_2
respectively of the FinalState F_S
// S.clipList denotes the list of clips associated with state S
  I_Lb ← Initial_ball_loc
  I_L1 ← Initial_player_1_loc
  I_L2 ← Initial_player_2_loc

  I_S ← (I_Lb, I_L1, I_L2)

  F_Lb ← Final_ball_loc
  F_L1 ← Final_player_1_loc
  F_L2 ← Final_player_2_loc
  F_S ← (F_Lb, F_L1, F_L2)

  QueriedClipsList ← Ø   // List of clips having subset of the
intermediate events between I_S and F_S
  OutputClipList ← Ø   // List of clips having all the events
between I_S and F_S

  I_S_Present = SearchState (I_S)
  F_S_Present = SearchState (F_S)
  if  I_S_Present = = false  OR  F_S_Present = = false  then
          Print "Query state does not exist"
  else
     Im_S = SelectConnectedStates( I_S, F_S)
                                        // Im_S - IntermediateState
if  Im_S = = NULL
         Print "Cannot reach the FinalState from InitialState"
     else
       for i in range( 0 to Im_S.size) :
         QueriedClipsList=QueriedClipsList U Im_S[i].clipList
          for k in range ( 0 to Im_S[i].clipList.size)
              for l in range ( 0 to Im_S[i+1].clipList.size)
                  if Im_S[i].clipList[k] == Im_S[i+1].clipList[l]
                    if Im_S[i].clipList[k].order ==
                       Im_S[i+1].clipList[l]+1
                        OutputClipList = OutputClipList ∩
                                                Im_S[i].clipList[k]
                     endif
                  endif
              end
           end
       end
     endif
  endif
end
```

"Eventually Query" button on the UI in Figure 4. The system displays icons for all three objects in locations corresponding to the current state. An "Eventually" query requires the specification of an *InitialState* and a *FinalState*. The user may select current state as the InitialState and record (I_{Lb}, I_{L1}, I_{L2}) by pressing *R/S* button or may specify an arbitrary InitialState by using the gamepad. Switch *A* is used to move and position objects on the tennis court in a pre-determined order (ball, $player_1$ and $player_2$) by moving their icons. The position of each icon is constantly displayed on the UI window. The record button *R* is used to record the locations of the ball and $player_1$ (I_{Lb}, I_{L1}) for the *InitialState*. After positioning $player_2$, *R/S* button is used to record its location (I_{L2}) to complete the information needed to fully specify the *InitialState*. Similarly, the *FinalState* (F_{Lb}, F_{L1}, F_{L2}) is also specified using switch *A*, buttons *R* and *R/S*. However, this time, when *R/S* button is pressed, it not only records the location of $player_2$, but also initiates the search process.

In the second step, the query built is executed to determine if there is a sequence of consecutive events that takes the game from *InitialState(S_i)* to *FinalState(S_j)* in S^3G. Note thst S_i and S_j are used to denote initial and final state instead of I_S and F_S for convenience. If S_i or S_j is not present in S^3G, the algorithm terminates saying that queried events are not present in the tennis video. If both the states are present, the system finds all possible paths from S_i to S_j using a graph-traversal algorithm. One clip may completely include a path or a path may be spanned by a sequence of successive clips. Let the set of clips associated with S_k be C_k for i<= k <= j. The clips present in the list $\{C_i \cap C_{i+1} \cap C_{i+2} \ldots \cap C_j\}$ are identified, and each clip in which, the states satisfy the order constraint is placed in OutputClipList as a clip that includes the path completely. The clips present in the list $\{C_i \cap C_{i+1} \cup C_{i+1} \cap C_{i+2} \cup C_{i+2} \cap C_{i+3} \ldots \cup C_{j-1} \cap C_j\}$ are identified and this list is called QuriedClipList. A virtual clip which takes the game from S_i to S_j is created by reordering these clips in increasing order.

As the query is built, the clips that satisfy the conditions specified by the user are made available to the user. Each clip has *StartTime* and *EndTime* that determines the timings of the beginning and ending of a clip, respectively. The user query may not involve retrieving back to back clips from the same video. Therefore, the clip is played for the user, and it is paused automatically at *FinalState* to let the user define the next query. Also the current QueriedClipList and the OutputClipList are displayed in the UI. OutputClipList contains the set of clips that satisfy the user query. However, it may be possible that when the user reaches a step in query building process, there might not be any clip that satisfies all the conditions specified by the user so far. Our system also maintains QueriedClipList that maintains all clips that satisfied all sub-queries. The user may check either list to see the relevant clips.

 ISBN: 978-1-61208-129-8

IV. ILLUSTRATION

Figure 4 displays the user interface for building a query. It has three components: tennis video display, the court view, and buttons for functionalities and drop boxes for query results. The tennis video display is the major component for building a query. As the user builds a query, a corresponding clip is shown to the user. The tennis court view helps the user to associate objects with locations. There are three buttons available: "New Query" to start a new query, "Eventually Query" to skip some states during query building process, and "Query History" to visualize the query built so far. The drop boxes are used to see the list of clips that satisfy the conditions during a query building process.

The three steps involved in the query process is supported by the UI. The first step of building the query is done in the query window, "searching step" is done in the background and search results are appropriately displayed into two lists mentioned. And the last step is done by displaying the queried event in the query window. The black ring icons represent each of the player's location (L_1 and L_2) and yellow icon represents the ball location (L_b).

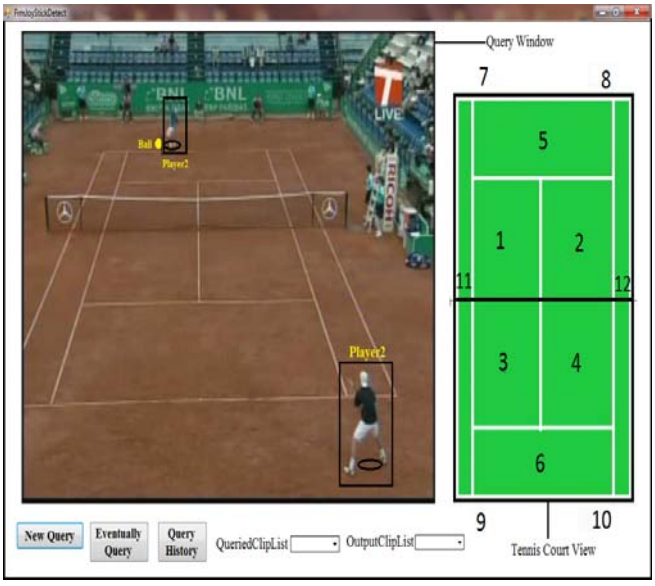

Figure 4. Snapshot of the User Interface

When the "Eventually Query" button is pressed any time during the query process, the icons for the players and the ball appear in the query window as shown in Figure 4. This allows the user to provide the location L_1, L_2 and L_b for the Player1, Player2 and Ball using the features of gamepad. These locations are recorded as InitialState I_S (I_{L1}, I_{L2}, I_{Lb}). As shown in Figure 5 I_{L1}= location 8, I_{L2} = location 6 and I_{Lb} = location 3. Similarly, FinalState F_S is provided by moving the corresponding icons using features of gamepad as shown in Figure 6. Thus F_S (F_{L1}, F_{L2}, F_{Lb}) = (5, 9, 2). Figure 6 also shows the snapshot after providing the eventually query have been executed. QueriedClipList contains four clips that match any intermediate event between I_S and F_S. OutputClipList shows one clip that has the entire events between I_S and F_S.

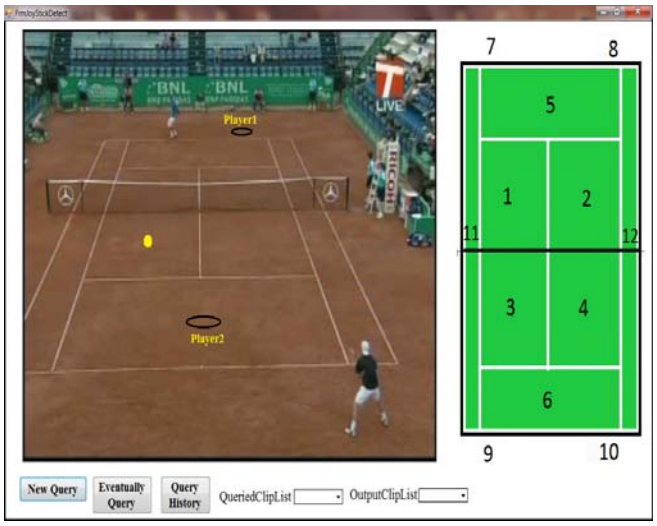

Figure 5. Snapshot after providing then InitialState I_S

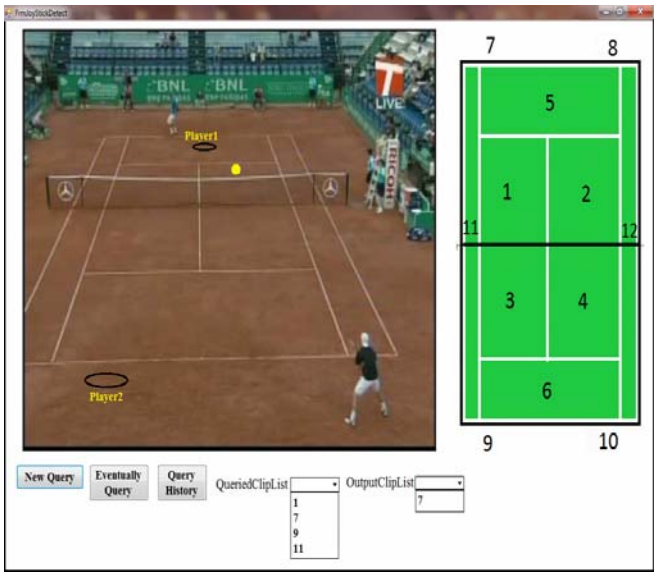

Figure 6. Snapshot showing the queried FinalState F_S and results after executing the "Eventually" query

V. USABILITY STUDY

A usability study was conducted to compare our gamepad user interface (GI) with the mouse interface (MI) that uses point-and-click approach developed by Naik [2] satisfaction as metrics (ISO, 1998). Ten users who were almost randomly selected to participate in the study were trained to use both interfaces and then were asked to build five test queries of varying complexity to take

 ISBN: 978-1-61208-129-8

measurements to assess the three metrics. The environment was designed to ensure that the study was fair and unbiased.

The user satisfaction was measured using preference and ease-of-use. Preference is a measure that indicates the likelihood of using one interface over the other. After completing all queries each user was asked to indicate his or her preference on a scale from 1 to 5 (1 – I definitely choose MI, 2 – I prefer MI over GI, 3 – I have no preference, 4 – I prefer GI over MI, 5 – I definitely choose GI). The metric *ease-of-use* was also ranked on a scale of 1 to 5 (1 – very low, 2 – low, 3 – average, 4 – high, 5 – very high) for both user interfaces. Based on the user data, it was concluded that the users overwhelmingly (9 out of 10) preferred GI over MI with preference receiving an average score of 3.7/5.0. For the metric ease-of-use, all users ranked GI high (score 4) and MI average (score 3), respectively. This clearly indicates the gamepad interface causes less user discomfort than the mouse interface. The overall opinion of users also favored GI over MI.

VI. CONCLUSION

This paper presented an innovative user friendly system for retrieving the desired clips from tennis game video using a gamepad. The system allows user to build spatio-temporal 'eventually' queries. Eventually query is an important type of query since it is usually difficult to have a proper user interface but it is very important since the user does not need to provide all the details about a query. We have used S^3G to build eventually queries. We have developed an interface that gives the feeling of playing a game as the inputs are received through a gamepad. As future work, we look into other types of temporal queries. We also plan to specify the type of shot (forehand and backhand) while the user builds a query.

Though the indexing capability of S^3G is described for tennis game videos it can easily be used for indexing general videos like other games and news events. In all videos objects interact because of events caused by objects or natural phenomena in a limited space over time. The only differences among different types of videos are number of objects, events, spatial layout, and the associated semantics. Therefore, S^3G can be used to index general video. However, the number of states and the number of arcs (transitions) may become very large if the video involves too many objects and events. If the designer takes sufficient care to minimize the number of states and transitions based on the number of active objects and relevant events then S^3G can be used effectively. For example, in a basketball game, there are ten players on the court and one ball. Passing the ball, dribbling from one location to another, shooting are examples of events (state transitions).

At present, S^3G is built from the string representation of the video manually generated by SMART [1]. Future research should focus on automating the generation of the string data using video analysis techniques. Also work for automatic feature extraction for building content based image retrieval is going on in parallel in our group. It is suggested that experiments be conducted in an environment of a collection of videos. Minor modifications are needed to S^3G to accommodate retrieval of selected clips from multiple videos. However, the optimization for searching S^3G is limited since we are interested in all paths for an eventually query to retrieve all relevant clips. In the future, the probabilistic relationship between states through transitions can also be studied.

ACKNOWLEDGMENT

This material is based upon work supported by National Science Foundation under Grant No. 0812307.

REFERENCES

[1] Jain, V. and Aygun, R. S., "SMART: A grammar -based semantic video modeling and representation," IEEE Southeastcon, 2008, pp. 247-251.

[2] Naik, M., Jain, V., and Aygun, R. S., "S3G: A Semantic Sequence State Graph for Indexing Spatio-temporal Data - A Tennis Video Database Application," IEEE International Conference on Semantic Computing, 2008, pp. 66-73.

[3] Erwig, M. and Schneider, M., "Spatio-Temporal Predicates," IEEE Trans. on Knowledge and Data Eng., vol. 14, no. 4, July 2002, pp. 881-901.

[4] Erwig, M. and Schneider, M., "Developments in spatio-temporal query languages," Tenth International Workshop, Database and Expert Systems Applications, 1999, pp. 441-449.

[5] Bonhomme, C., Trépied, C., Aufaure, M., and Laurini, R., "A visual language for querying spatio-temporal databases," Proceedings of the 7th ACM international Symposium on Advances in Geographic information Systems, 1999, pp. 34-39.

[6] Sourina O., "Visual 3D Querying of Spatio-Temporal Data," International Conference on Cyberworlds, 2006, pp. 147-156.

[7] Erwig, M. and Schneider, M., "Spatio-Temporal Predicates," Technical Report, FernUniversit at Hagen, 1999.

[8] Erwig, M. and Schneider M., "Query-by-Trace. Visual Predicate Specification in Spatio-Temporal Databases," Proceedings of the 5th IFIP Conf. on Visual Databases, 2000, pp. 199-218.

[9] Cavalcanti, V. M., Schiel, U., and de Souza Baptista, C., "Querying spatio-temporal databases using a visual environment," Proceedings of the Working Conference on Advanced Visual interfaces, 2006, pp. 412-419.

[10] Li, X. and Chang, S. K., "An Interactive Visual Query Interface on Spatial/temporal Data," Proceedings of the Tenth International Conference on Distributed Multimedia Systems, 2004, pp. 257-262.

[11] Silberschatz, A., Korth, H. F., and Sudarshan, S., Database System Concepts, 3rd Ed., McGraw Hill, 1997.

 ISBN: 978-1-61208-129-8