Journal of Visual Languages & Computing, Volume 29, August 2015, Pages 63–76,
doi:10.1016/j.jvlc.2015.03.003

Query-by-Gaming: Interactive Spatio-Temporal Querying and Retrieval Using Gaming Controller

Vineetha Bettaiah and Ramazan S. Aygun
Computer Science Department
University of Alabama in Huntsville
vineetha.bettaiah@gmail.com, aygunr@uah.edu

Abstract

Spatio-temporal querying and retrieval is a challenging task due to the lack of simple user interfaces for building queries despite the availability of powerful indexing structures and querying languages. In this paper, we propose *Query-by-Gaming* scheme for spatio-temporal querying that can benefit from gaming controller for building queries. By using *Query-by-Gaming*, we introduce our spatio-temporal querying and retrieval system named as GStar to interactively build *subsequent* spatio-temporal queries to determine if a state is directly reachable from current state and *eventual* spatio-temporal queries to know whether a spatial state is reachable from a current state. Queries are built using features of gaming controller by displaying the original video frames rather than on a graphical interface using a mouse or a keyboard. GStar has three main components: building the query, searching and retrieval of clips, and displaying query results. The queries are applied to an indexing structure called semantic sequence state graph (S^3G) and results of the query are displayed dynamically to provide timely feedback to the user. Experimental results and user interface are provided for a tennis video database. Users define desired game state (player and ball position) using an interactive interface at multiple points in time and GStar automatically retrieves all rallies that contain both states. Finally, the user interface evaluation comparing gamepad-based interface and mouse interface for spatio-temporal querying has been studied.

1. Introduction.

Spatio-temporal querying is the process of retrieving spatio-temporal objects and events that span space and time domains. Each object may change its position and its relationship with other objects continuously over time due to events or natural phenomena. As time and space are both continuous domains, the number of states in which a spatio-temporal system can exist might be infinite. Therefore, capturing and representing various states of the system and influencing events that cause state transitions along with semantics is challenging. The design of a good spatio-temporal query system should consider the following.

1) *Representation*. The methodology used to model the spatio-temporal system must be sophisticated enough to capture the semantic information into a representation of manageable size and structure.
2) *Query building.* Unlike traditional database queries, spatio-temporal queries request semantic information based on user's interpretation. Often, the query is complex consisting of several related events and interactions. Therefore, the system should allow the user to build the query methodically through an environment that will guide the user step-by-step to build a spatio-temporal query without knowing the internal representation of the data to retrieve a sequence of events which causes the specified action.

3) *Simplicity*. The spatio-temporal query system must provide a user friendly interface that is easy to use. The interface must be intuitive and must not require the user to know the internal representation of the semantic information. This allows changes and upgrades to the internal representation of the system without affecting the user interface.

1.1 Related Work

Spatio-temporal querying and retrieval has been studied by researchers. The research involves developing spatio-temporal querying languages and interfaces for building spatio-temporal queries. In addition, we briefly mention challenges of incorporating temporal constraints.

Spatio-Temporal Query Languages. Developing querying languages is important for experts to query the database. If visual interfaces are developed for building queries, these visual queries can be mapped to the querying language. There has been significant effort on querying spatio-temporal databases. The STQL (Spatio-Temporal Query Language) [3] is developed to demonstrate how SQL can be extended to query changes of spatial objects over time. It extends SQL by adding features and a mechanism to support complex spatio-temporal predicates (*disjoint*, *meet*, *overlap*, *coveredBy*, *covers*, *inside*, *contains*). STQL is based on *select-from-where* clause by integrating predicates into SQL. A survey of qualitative spatial reasoning is provided to represent key calculi for describing a set of spatial relationships related to direction, distance, shape, moving objects, and uncertainty [33]. Moving GeoPQL [23] is an extension of Geographical Pictorial Query language. Spatio-temporal queries are formed by adding temporal layers and defining temporal properties of spatial objects. The temporal SOLAP (GIS+online analytical system) includes a temporal language named TPiet-QL [26]. Although the proposed system in [26] has a powerful way of visualizing results, the application interface accepts queries in textual TPiet-QL format. Jain et al. [9] use 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.

User Interfaces for Spatio-Temporal Querying. Since learning about the query language is hard for traditional users who query and update the database using standard types of queries and updates called canned transaction [32], such users rather rely on a sophisticated user interface for building queries. The user interface for visual query languages usually uses icons to represent objects. The user selects objects and specifies the desired interactions by choosing the operators supported by the system. Penna et al. [12] propose a SQL-like SRQ (spatial relation query) by allowing users to arrange visual information (without temporal information). Liu et al. [13] describe a visual graph based query interface that has 4 rectangular boxes to define variables, conditions, video attributes and scenes, and 2 connections to link variable boxes or scene boxes. Such an interface helps to define textual queries with the user interface however declaring spatio-temporal component via variable connection may not be easy for users. Certo et al. propose a visual time automaton for building time queries. Although their system supports queries at different temporal granularities and simplifies query building, the users may need to learn about building basic automata [14]. The user interface in [17] supports multi-DST (dynamic spatio-temporal) queries and enables users to select multiple regions with corresponding intervals for a specific object to generate spatio-temporal queries. Naik et al. [10] provides a user interface for querying tennis video databases. The user chooses (or clicks) 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 since it requires both choosing player/ball objects and their locations, and such an interface does not provide an intuitive method of building queries. A query is built by providing a sequence of states (object-location pairs) but not necessarily consecutive and then linking those states by actions of objects. The output of a query is a clip that contains the sequence of states with corresponding actions. In [27], the queries are categorized into regional, fuzzy spatio-temporal, and trajectory queries and a button is provided for each type of a query (e.g., find regions of an object for a specific interval).

Query-by-Example. Since the users may not know the internal representation of database, the users may use a) similar examples by browsing the database, b) provide what the result should look like by generating a sample scenario using the interface or c) sketch the components and interactions of objects in a query. Papadias et al. [30] propose a pictorial query-by-example that helps to retrieve or define direction relations from symbolic images. The user is able to develop queries such as "retrieve all sub-images where object X is northeast of object Y." The QBIC (Query by Image Content) [30] supports querying based on image and video content based on color, shape, texture, and sketches. The QBIC system allows users to define their queries based on actual images from the database and a graphical interface that provides drawing and selecting. Query by sketch in QBIC allows user to draw main lines and edges in an image by freehand and template matching is applied based on the reduced-resolution edge map. For music retrieval, query-by-humming [31] enables users to query an audio database by representing the melodic information in a song as relative pitch changes. Similar query-by- concept was also applied for spatio-temporal querying. Query-By-Trace (QBT) [4], Visual Interactive Query Interface [6] and Visual Query system S-TVQL [5] are examples of visual interfaces for spatio-temporal databases. QBT [4] interactively and graphically produces a sketch from a derived spatio-temporal predicate. The user inserts objects freely (line or area) in an area and defines the movement of each object by sketching the path. A query includes several such traces which determine the spatial relations between the objects. Query-by-Trace allows users to define topological relationships for regions and trajectories on a 2D screen where x-axis indicates the spatial component and vertical axis indicates the temporal dimension. For example, a user may draw an evolving region such as a hurricane and then draw trajectory of flights that go through a hurricane for a query "find flights that go through hurricanes." The user interface of STVQL [6] presents the database in the form of maps and list of attributes. The user selects the element of the query by clicking on the desired attributes. Based on the selected attributes all valid operators are displayed in the interface. The underlying DBMS is relational with spatial capabilities. The valid-time temporal information of an object is stored as explicit time attributes (*from*, *to*). In [16], iconic visual querying allows user to drag 3D icons to a scene editor and provide relative positions of the objects. The object motions are captured using multi-track metaphor where the trajectory of an object is captured as the user moves the object. The motion of multiple objects are synchronized with respect to relative manual dragging speed of the user. Although such a scenario is beautiful, the user needs to build the query from scratch and the order of object motions may not be preserved due to rough synchronization among trajectories.

Temporal Constraints of Spatio-Temporal Queries. The interface of Mars [18] lets the user choose the spatial location on a map by using map functions and find microblogs in the last specific number of hours by providing the time value along with a weight between distance and time. Although statistical results can be provided in Mars, the interface only accepts a position of interest and a single time interval. Such an interface may not be effective if the number of spatio-temporal intervals is more than two. The spatio-temporal interface for analyzing taxi trips in New York City [19] enables selection of regions on a map and selecting either temporal interval or a hierarchy of temporal intervals (years, months, days, and times of days). Such interface does not support spatio-temporal querying for multiple moving objects simultaneously. The STEWARD [20] and Newsstand [22] systems execute spatial components of a spatio-temporal queries at the server side. The temporal component is executed by a slider that indicates the interval but it suffers from poor updates of the query results during interactivity if spatial results are not indexed properly. The VacationFinder system [21] helps to analyze where people are traveling based on twitter messages by choosing spatial location and choosing two temporal intervals. So this type of system limits spatial and temporal components of a query. Spatio-temporal Pseudo-Relevance Feedback system [24] accepts temporal condition in textual form. The spatio-temporal search of crowd sourced information [25] only accepts beginning and ending date for the temporal dimension of a spatio-temporal query. The Murmur project [28] supports querying time-varying attributes using a time travel facility for geo-databases. The time travel facility provides the map and thematic view at an instant and also supports animation of the map similar to a movie-map for a time interval. The continuous querying component of the PLACE (Pervasive Location-Aware Computing Environments) [29] proposes sliding-window concept to support temporal expiration. For example, an object that satisfies the temporal condition such as “within one hour” may not be part of the query result in future time. The PLACE system focuses on continuous querying and only one temporal interval is supported.

1.2 Our Method

One of the challenges in spatio-temporal querying is to provide spatio-temporal information about multiple objects in a synchronized or ordered way. The multiple trajectory-based queries, it may not check the concurrent progress of trajectories. For example, for a simple query like “ find videos where object A moves to X as object B moves to Y and then object A moves to Y as object B moves to Z” requires inputting partial temporal constraints of a query carefully. The order of events of multiple objects is important for spatio-temporal querying. In gaming, multiple objects can be controlled in a simple way by changing the object in control. Our proposed method utilizes linear temporal logic for building temporal constraints and benefits from gaming in the way spatio-temporal inputs are provided to video games.

Linear Temporal Logic. In Linear Temporal Logic (LTL), two important temporal operators are next (○) and eventually (◊). The ‘next’ operator allows checking whether a condition is true in a subsequent state whereas the ‘eventually’ operator allows checking whether a condition eventually becomes true. Based on these operators, we propose *subsequent* spatio-temporal querying for the ‘next’ operator and *eventual* spatio-temporal querying for the ‘eventually’ operator. We represent spatio-temporal data as a sequence of states, where each state is determined by a unique set of object-location pairs. An event causes the system to transition from present state to next state, and is represented as an arc between the two states. The *subsequent* queries involve building the query in a step-by-step manner and *eventual* queries involve specifying two states and retrieving all sequence of events that takes the video from the first state to the final state. In other words, *eventual* query checks whether a state is reachable from a current state.

Motivation for Using a Gaming Controller. We observe that an environment that gets spatio-temporal input from the users is video games. In these games, spatial and temporal movements of the elements are provided by the user using a gaming controller. Gaming controllers can be used with some basic knowledge about the rules of the game by a broad domain of users from 4-year old kids to 70-year old people. In this study, we adapt this idea for building spatio-temporal queries.

Query-by-Gaming. In this paper, we propose a novel spatio-temporal querying and retrieval system named as GStar using a methodology similar to the one in gaming by getting user interactions through a gaming controller. We name our querying scheme as *Query-by-Gaming.* GStar supports Query-by-Gaming and provides approximate querying for partial match of user queries. Our GStar is tested on tennis videos. GStar supports *subsequent* and *eventual* queries and has three components: building the query, searching and identification of clips, and displaying query results. For a tennis video, a 'clip' correspond to all events in a tennis point from serve until one of both player scores a point. Semantic sequence state graph (S^3G) [10] is used as an indexing structure to which the queries are applied. Our technique has an advantage over Query-By-Example systems [7] since the user does not need to search for a similar image or video to start the query and also has an advantage over sketch-based systems [8] since the user may not know how the actual video looks like or how sketches are mapped by the querying system. In this work, we choose gamepad as a gaming controller and compare it with the mouse interface.

The fundamental difference in our user interface from other interfaces is building the query. The main idea can be briefly summarized as follows:

1. The user will generate the query as if he or she plays a game.
2. As the user provides the commands to the system, the system works on two components:
    a. The system finds the best video clip and provides (shows and plays) it to the user instantly so that the user can continue to build query based on the current scene.
    b. The system searches for data that fits the user query and stores them to show to the user when the query is complete.

Our contributions can be summarized as follows:

- Use of a gaming controller for spatio-temporal querying,
- Support for *subsequent* and *eventual* queries,
- Approximate querying for spatio-temporal queries, and
- Interactive and incremental query building by presenting actual video (database) objects.

This paper is organized as follows. Background on spatio-temporal logic, semantic modeling and indexing is provided in Section 2. Section 3 provides the overview of our system. Section 4 presents the Query-by-Gaming in GStar. Section 5 presents illustrations of applying queries to tennis game video. Section 6 provides usability study of the user interface. The last section concludes the paper.

2. Background.

In this section, we provide information about the background of our GStar system. Our system uses spatio-temporal logic, semantic modeling and retrieval system (SMART) and our semantic sequence state graph (S^3G) for the retrieval of videos from a tennis video database. These are briefly described below.

2.1 Spatio-Temporal Logic

Temporal logic helps to specify the temporal properties of a system. If the temporal information can be described as a path of vertices such as $S_0, S_1, \ldots, S_n$ where each S_i corresponds to an instant in time, linear temporal logic (LTL) [15] can be used to analyze such a system. Our system is based on the spatio-temporal logic described in [16]. A state diagram Ξ is a finite set of states where each state is represented by a node and events that cause state transitions are represented by directed arcs. There is an event corresponding to each state transition. The temporal assertion, $\Psi := (\Xi, S) \models \Phi$, states that for the state diagram Ξ and the current state (or condition) S, the temporal formula Φ is satisfied. In other words, the system is evaluated to check if Φ holds or not with respect to the current state. The temporal formula can be generated using Boolean operators such as $\neg$, $\wedge$, $\vee$, $\rightarrow$ and temporal operators such as next ($\circ$), eventually ($\Diamond$),global ($\Box$), until (U), and releases (R). A temporal formula can be expressed as:

$$\Phi := \theta \mid \neg\,\theta \mid \theta_1 \vee \theta_2 \mid \theta_1 \wedge \theta_2 \mid \theta_1 \rightarrow \theta_2 \mid \Box\,\theta \mid \circ\,\theta \mid \Diamond\,\theta \mid \theta_1 U\, \theta_2 \mid \theta_1 R\, \theta_2$$

This study focuses on *next* and *eventually* operators. These are briefly described as follows:

- **Next.** The notation for the next operator is “X” or “○”. X**θ** or ○**θ** checks the correctness of the condition θ to be satisfied in the next state. If the current state is S_i, (Ξ, S_i) ╞ ○**θ** states that θ becomes true in the next state of the graph Ξ starting from the current state S_i.
- **Eventually/Finally.** The notation for the eventually operator is “F” or “◊”. F**θ** or ◊**θ** indicates that there exists a state reachable from the current state where the condition θ is satisfied. If the current state is S_i, (Ξ, S_i) ╞ ◊**θ** states that θ becomes eventually true in the graph Ξ starting from the current state S_i.

2.2 Semantic Modeling and Retrieval System (SMART).

The semantic content of a video represents high-level information in the video which is of interest to viewers. In general, viewers are interested in objects, events, sequence of events and the resulting spatio-temporal interactions among objects in the video. SMART [1,9] is a system developed for modeling and retrieval of semantic contents in a tennis video. SMART models the semantic contents of a tennis video 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 which are given in [1].

Objects: The set of objects Σ_O consists of 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 consists of 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 as legend in Figure 1. The set of locations Σ_L includes all these 13 regions:

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

Using the production rules, the semantic contents of the video are encoded as a set of strings, one string for each clip, and stored in the database.

2.3 S^3G - Semantic Sequence State Graph.

The semantic sequence state graph (S^3G) [2, 10] 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 theoretically. Each assignment pattern defines a unique state in S^3G, and the number of states in S^3G is less than 13^3 due to game constraints. An event from the set of all possible events $\Sigma = \{F_1$ ($player_1$ 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.

Example: Consider three clips 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] of a video. In these string representations of SMART, *A* indicates close-view camera and *C* indicates the court view. In M1, $Player_1$ serves an ace from location 7 as $Player_2$ moves to location 4 to receive. M7 begins with a good serve from $Player_1$ from position 7. $Player_2$ hits a backhand shot and sends the ball to location 5. $Player_1$ in location 7 returns the ball to location

4 by hitting a backhand shot. In M10 $player_2$ serves from location 10 to location 7. The S^3G in Figure 1 represents the semantic information in the three clips described above. For example, nodes S_1, S_4 and the state transition from $S_1 \rightarrow S_4$ represent the clip M1.

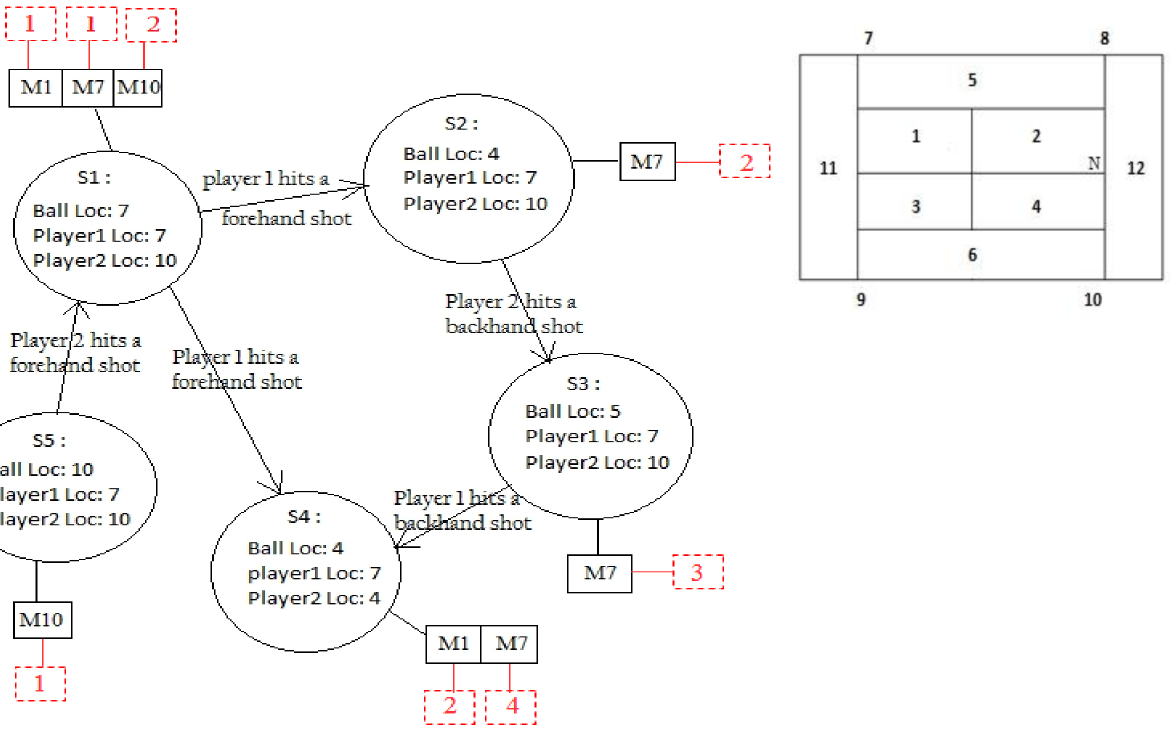

Figure 1: Construction of S^3G from SMART string data.

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. Each clip is attached a list of temporal orders (ranks) of the state as shown in Figure 1 by dotted red squares. These temporal orders or ranks should be checked to see if states appear consecutively in a clip. Firstly, clips common to all states involved in the query are selected. Then, clips in which states do not satisfy the temporal order constraints are deleted. Timings of these ranks are stored in the database. For example, time of S_1 in clip M1 may be at 127th second, time of S_4 in clip M1 may be at 129th second. Hence the event, $player_1$ hits a forehand shot from S_1 to S_4 starts from 127th second and ends at 129th second. These are represented as *StartTime* and *EndTime* for each event, respectively.

We have analyzed the performance of S^3G for retrieving as well as inserting into the index structure. Each node keeps the set of clips that has the corresponding state. Whenever the user provides a query for a specific node, the index structure is used similar to a finite state machine. Given a specific transition, the next state is reached instantly. The next state includes the sets of clips containing that state. The complexity of reaching the next state in S^3G is O(1). Locating a specific state has also time complexity of O(1) using hashing. We do not search the clips but we search the states of the index structure. Then the corresponding clips of the state are automatically retrieved. Given a state, getting the list of clips that contain the state is instantaneous since that list of clips are maintained at the node. Similarly, the set of clips that satisfy the next state can be determined instantaneously. The ranks of clips in consecutive states are checked to make sure that they appear back to back in the same clip. Checking ranks is equivalent to finding the intersection of sets. The complexity of finding intersections is based on the size of these small sets. If the sizes of two sets are m and k, then the complexity is $O(m+k)$ and if $m \geq k$ it can be stated as $O(m)$. Similar case applies for applying union of sets. If the intersection of sets is included, the time complexity becomes O(m); however, note that m is very small with respect to the total number of clips. Detailed discussion of experiments and time complexity analysis can be found in [10].

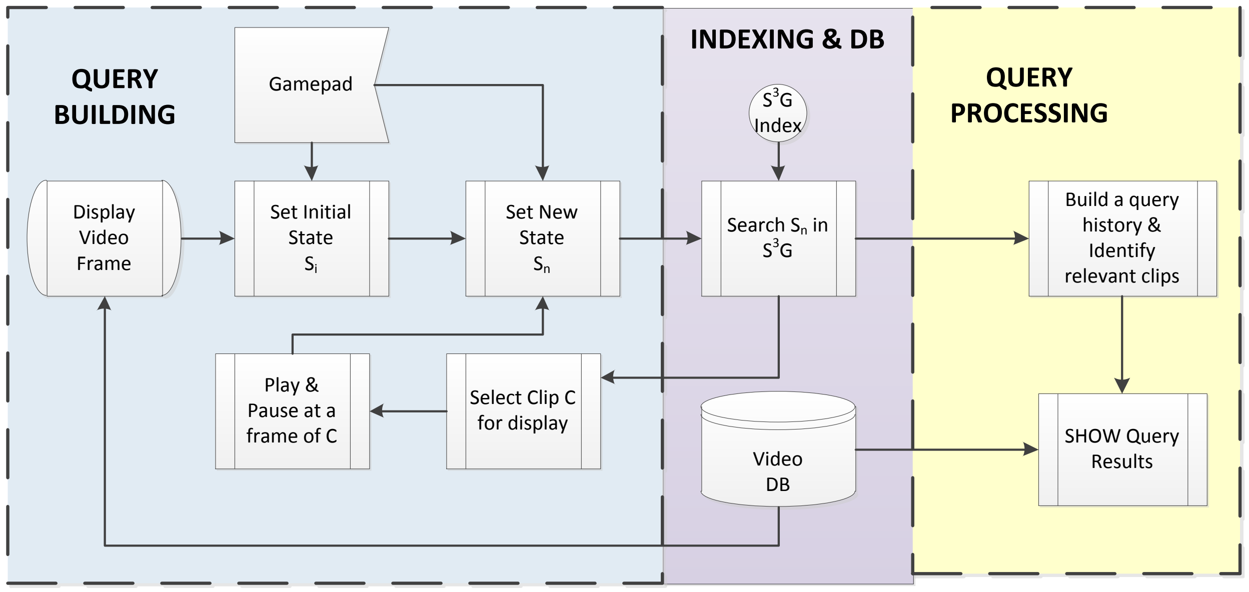

Figure 2: Components of GStar

3. System Overview

The two major parts of our system, query building unit and query processing unit, use the indexing structure and the database. The general framework of the system is provided in Figure 2. The main idea of our methodology is to map inputs from a gaming controller to a spatio-temporal query. The user controls or specifies the location of one object at a time. Using a gaming controller, the user may move object $\mathbf{o_i}$ from location $\mathbf{L_m}$ to $\mathbf{L_n}$. This implies that “find video clips where object $\mathbf{o_i}$ moves from location $\mathbf{L_m}$ to $\mathbf{L_n}$”**.** The query building block or unit of Figure 2, as soon as the system gets the new state information from the gaming controller, searches the database to determine whether there is any video clip in the database in which the specified transition occurs. If such clip or clips exist, the Indexing & DB unit provides one clip (or a fragment of a clip) to the query building unit. The query building unit plays the clip and pauses for the user to input the next subquery. The query building continues in this manner one subquery at a time. On the contrary, the query processing unit keeps track of all clips and sequences of clips that satisfy all subqueries specified so far. As the number of subqueries increases, the number of video clips maintained by the query processing unit decreases. When the user is done with building the query, the user may choose to see all clips that satisfy his or her query.

4. Query-By-Gaming in GStar.

In order to build spatio-temporal queries to retrieve video clips, it is important to provide a user friendly interface that is easy to use. A web-based platform independent interface [9] relies heavily on drop down menus to build SQL queries. A court-view and point-and-click approach is provided to specify the query in [2]. The annotated videos are maintained as SMART strings in our database. The annotation was initially done manually. We have later used a player template (a sample image of a player from the video) to track the location of players. Our GStar, which builds the queries using gaming controller to retrieve clips from tennis game video, looks similar to playing the game itself. Our approach is motivated by the following: a) gaming controllers are common user interface devices in gaming environments where user provides spatio-temporal inputs of the elements and b) gaming controllers with multiple buttons and different types of interface features like trigger, directional controls are, usually, more suitable to provide spatio-temporal inputs as compared to other devices like keyboard and traditional mouse.

We have chosen gamepad as a gaming controller in this research. The buttons or features of a gaming controller should be mapped to the semantic information in S^3G. After providing how a gaming controller such as a gamepad can be used for spatio-temporal queries, we explain how spatio-temporal queries are built.

4.1 Features of Gamepad.

The gamepad along with other features includes a cluster of 10 buttons $B = \{b_1, b_2, \ldots, b_{10}\}$, an 8-way switch *A*, a record button *R* and a record/search button *R/S* as shown in Figure 3. The features, switch *A*, record *R* and record/search *R/S* are used in GStar to build spatio-temporal queries.

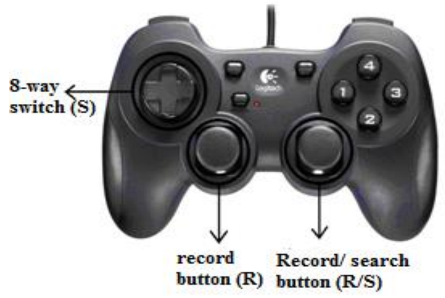

Figure 3: Features of gamepad

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

Mapping input obtained from the gamepad to the semantic information that the tennis video represents forms an integral part in retrieving clips that contain the queried events. Queries are provided using the features of the gamepad, and each feature (a button of *B*, 8-way switch *A*, etc.) of the gamepad generates a unique numerical input when pressed. Based on the game context these values are mapped appropriately to semantic information. For example, the input values from the 8-way switch are mapped to the direction of movement of the object on the tennis court while building the query. Thus GStar builds the query and applies as input to the indexing system, S^3G.

4.3 Building Spatio-Temporal Queries Using Gamepad.

A tennis match is composed of sets, and sets in turn are composed of games. Games are composed of points. In our database, each point corresponds to a clip. The process of building and retrieval of clips involves three main steps. First, spatio-temporal queries are built interactively using the gamepad. Secondly, a search process is initiated to find clips having interesting events using our graph based indexing structure (S^3G). And finally, the video clips identified by S^3G are fetched from the database and presented to the user. We explain how *subsequent* queries and *eventual* queries are implemented in GStar. GStar supports their combinations as well.

4.3.1 Subsequent Query.

The three main steps involved in building the *subsequent* query are detailed below.

Specifying States. The first step in building the query starts with a tennis game video being played in a window.

1. Query video initialization. The chosen (or default) video is played from designated start point and paused at the beginning of the first serve. The window used for playing the video is also used as the query window to position the ball and the players while building the queries.
2. Setting the initial state. GStar allows the user to select any state as an initial state, Ω_0, to start up the query process by selecting the "Set Initial State" feature on the user interface. Selection of this feature allows the user to select a specific location for ball and the two players. These locations taken together become Ω_0. L_1 (Player$_1$) location is selected first using switch ***A*** followed by pressing switch ***R*** to record its location. This is followed by selecting L_b (ball location) and L_2 (Player$_2$) in a similar way. Finally pressing switch R/S to record L_2 also triggers a search process in S^3G to verify the presence of the initial state, Ω_0.
3. Identify and set the initial and current states. If the initial state is not set explicitly, the location of the two players (L_1 and L_2) and the ball (L_b) are registered when the video is paused at the first serve. These values taken together (L_b, L_1, and L_2) represent the initial state, Ω_0. In other words, the current state of the game (Ω_C) is initially represented by Ω_0. The user can start building the query from Ω_0 or the user has the option to specify an arbitrary state as Ω_C.
4. Set the next state. After deciding on the current state (Ω_C), the user begins building the query by specifying the ball and player positions for the *next* state, Ω_N.
   a. Specifying the ball position. This is accomplished by moving the ball icon to the desired position with the help of switch *A*, and by using button *R* to record the ball's location in L_b. The user is given constant visual feedback of the movement of the ball by displaying ball icon in query window.
   b. Specifying the players' positions. After specifying the ball location, the gamepad features are used to specify the location of the player who receives the ball. Again, the visual feedback of the player's position on the court is provided to the user by displaying an icon that represents the player in the query window. After choosing the desired location, *R/S* switch is used to record the player location (L_1 or L_2).
   c. Initiate the search for the next state. The next state, Ω_N, is represented as (L_1, L_2, and L_b) and search for this state starts.

This completes the building of the first query, which specifies the desired ball and player location for the *subsequent* state (NextState).

Finding Clips Satisfying the State Sequence. Let Ω_C and Ω_N denote the current state and next state, respectively. Let C_i represent the clips in Ω_i and each clip $c_{i,m} \in C_i$has a rank denoted as $r(c_{i,m})$ that represents its relative order in its actual clip. $\mu(\Omega_i, \Omega_k)$ denotes the set of directed edges from state Ω_i to state Ω_k, and its cardinality is denoted with $|\mu(\Omega_i, \Omega_k)|$.

1. Check reachability of the next state from the current state. If $|\mu(\Omega_C, \Omega_N)| \geq 1$, then there might be a common clip between them. However, the presence of the common clip needs to be verified.
    a. Check the presence of common clip. The set of common clips between Ω_C and Ω_N can be found using $C_{C,N} = C_C \cap C_N$.
    b. Check whether the states appear back to back. After finding a common clip, it needs to be verified whether states Ω_C and Ω_N appear back to back in the clips using the rank of common clip(s) in Ω_C and Ω_N. The set of satisfying clips is found as

$$\overline{C_{C,N}} = \{c_p | ((c_p \in C_C) \wedge (c_p \in C_N)) \wedge (r(c_{p,N}) = r(c_{p,C}) + 1)\}$$

2. Return the query results. Assume that the states that are involved in a query are Ω_0, Ω_1, Ω_2, …, Ω_n. The output of the query returns

$$\mathbb{Q} = \bigcap_{k=0}^{n-1} \overline{C_{k,(k+1)}}$$

Besides returning the actual clips, GStar also returns the union of common clips with proper rank information from each state to the next state as the query is built:

$$\mho = \bigcup_{k=0}^{k=1} \overline{C_{k,(k+1)}}$$

For example, if $C_0 = [1, 7, 8, 9]$, $C_1 = [7, 8, 9, 13]$, and $C_2 = [8, 9, 13]$ then $\overline{C_{0,1}} = [7, 8, 9]$ and $\overline{C_{1,2}} = [8, 9, 13]$. Here, we just ignore the rank information for simplicity. Therefore, $\mathbb{Q} = [8,9]$ whereas $\mho = [7,8,9,13]$. The set $\mho$ gains importance when $\mathbb{Q}$ is empty. By using $\mho$, it may be possible to see a set of clips that satisfy the spatio-temporal query partially. Therefore, the set $\mho$ provides results for approximate querying.

Displaying Results. There are two aspects of displaying query results: query building and query outputs. As the user builds a query, the best clip that satisfies the query is chosen to be displayed to proceed building the query. When the query user completes the query, the user may want to see the list of all clips that satisfies his or her query. To achieve these two aspects of display, at all stages of the query process GStar displays a variety of information that may be of interest to the user. The *StartTime* and *EndTime* of every event in a clip are available in the database. Using this information the system retrieves and plays the events between the queried states in the query window easily. This allows the user to visualize each event immediately if the event is present.

- *Displaying a Sample Satisfying Clip for the Query Building Interface.* GStar automatically pauses the tennis game video at the *EndTime* of the event allowing the user to build the *subsequent* query. After the execution of each query, the next state (Ω_N) of the current query becomes the current state (Ω_C) for the *subsequent* query. The query process may continue until the user wishes or the query result ($\mathbb{Q}$) becomes empty. As the user develops the query by adding new states, $\mathbb{Q}$ becomes smaller and smaller. If the query process terminates since $\mathbb{Q} = \emptyset$, then there is no clip that contains the events between all the queried states. On the other hand, if $|\mathbb{Q}| \geq 2$, there is more than one clip that satisfies the query at the current stage. In order to maintain continuity and smoothness during the visualization of events while building the query, the system, if possible, plays the current event from the same clip from which the immediately previous event was played. If the current event is not present in the clip of the previous event then it is played from another clip chosen randomly from $\mathbb{Q}$. In this case, the continuity and smoothness may be lost.

- *Displaying Output Query Results.* GStar provides the list of clips in $\mathbb{Q}$ and $\mho$ in the user interface. Each clip in $\mathbb{Q}$ satisfies all the subqueries of the spatio-temporal query whereas each clip in the $\mho - \mathbb{Q}$ has events that satisfy some of the subqueries.

4.3.2 Eventual Query.

In our previous work, we present basics of *eventual* query [11]. It has similar steps as in building the *subsequent* query. Rather defining the *next* state that will happen right after the current state, the eventual query defines the *final* state that may be reachable after a series of states from the current state. Since the steps involved in building the *eventual* query are similar to the steps of *subsequent* query, they are just summarized below.

Specifying States. The *eventual* query is built by selecting "Eventually Query" feature on the UI and specifying an initial state *(Ω_0)* and a final state *(Ω_F)*. GStar displays icons for all three objects in locations corresponding to the current state. The user may select current state as the initial state and record $\mathbf{\Omega_0}$(L_b, L_1, L_2) by pressing *R/S* button or may specify an arbitrary initial state in a way similar using the "Set Initial State" feature. Similarly, the final state Ω_F(L_b, L_1, L_2) is also specified using switch *A*, buttons *R* and *R/S*.

Finding Clips Satisfying the State Sequence. The query built is executed to determine if there is a sequence of consecutive events from the current state, Ω_C, to the final state, Ω_F, in a single tennis point of S^3G. After confirming that both the states are present, the system finds all possible paths from Ω_C to Ω_F. Each clip in which the states satisfy the rank constraint is placed in $\mathbb{Q} = \bigcap_{k=0}^{n-1} \overline{C_{k,(k+1)}}$ as a clip that includes the path completely. The clips present in the list $\mho = \bigcup_{k=0}^{k=1} \overline{C_{k,(k+1)}}$ are identified and this list is called approximate query results.

Displaying Results. Similarly like in *subsequent* query, the events are played from *StartTime* of the initial state till the *EndTime* of the final state and paused at that point. The $\mathbb{Q}$ and $\mho$ are displayed in the UI which clips have the similar meaning as in *subsequent* query.

Journal of Visual Languages & Computing, Volume 29, August 2015, Pages 63–76,
doi:10.1016/j.jvlc.2015.03.003

5. Illustration of GStar.

We have developed a simple user interface (UI) shown in Figure 4 for evaluating our GStar. It consists of a query window where the actual tennis game video is played. The window supports the user to build spatio-temporal query using gamepad. The tennis court view is provided on the UI to guide the users in selecting positions for the ball and the players to build the query. The UI consists of features "Next Query" for *subsequent* query, "Set Initial State", "Query History", "Eventually Query" for *eventual* query, and drop down lists QueriedClipList and OutputClipList.

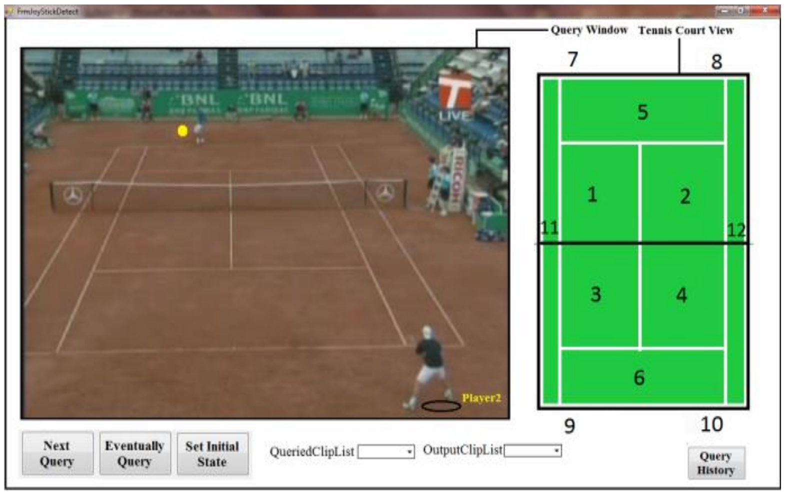

Figure 4: Snapshot of the User Interface,

Setting Initial and Current States. When GStar is invoked initially, the system is designed to automatically play the video in the query window from the designated start point and pause at the beginning of the first serve. For each video that is stored in the database, the player who serves first is designated as player$_1$. Figure 4 also shows the snapshot of the UI where the video is played and paused at the first serve. The icons of player$_2$ and the ball are displayed on the query window. In Figure 4, at the start of the first serve, player$_1$ and ball are in location 7, and player$_2$ is in location 10. These locations taken together define the default initial state, Ω_0, which is also the current state, Ω_C, for query. At this instant, GStar is ready to receive the locations of the ball and player$_2$ as inputs for *subsequent* query. The user has the option to change the default initial state by selecting "Set Initial State" feature.

Setting the Initial State Explicitly. As mentioned earlier, the user may choose any desired state as initial state by selecting "Set Initial State" feature. Figure 5 shows the snapshot of the UI after the user selects "Set Initial State" feature. The query window displays icons for the ball, player$_1$ and player$_2$. User can specify the initial state by moving the icons of the three icons of the objects to desired locations.

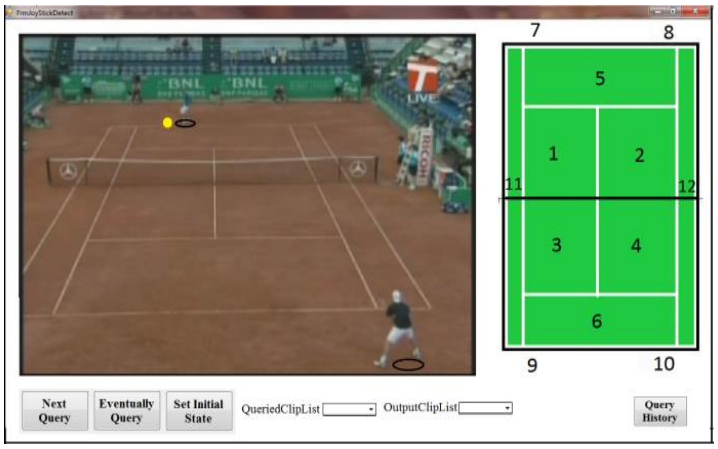

Figure 5: Snapshot after selecting the "Set Initial State" feature

Setting the Next State. Figure 6 shows the snapshot after the user specifies a *subsequent* query starting from the default initial state or current state. It can be seen that user has provided the ball location as 4 and $player_2$ location as 10 by moving their respective icons to those locations. The set of locations (7, 4, 10) defines the next state, Ω_N. The query is executed and clips containing the queried information are identified in S^3G and retrieved from the database. The query results are presented to the user ($\mathbb{Q}$ and $\mho$) as shown in Figure 6. After the execution of the query, (7, 4, 10) becomes the current state for the *subsequent* query to be built.

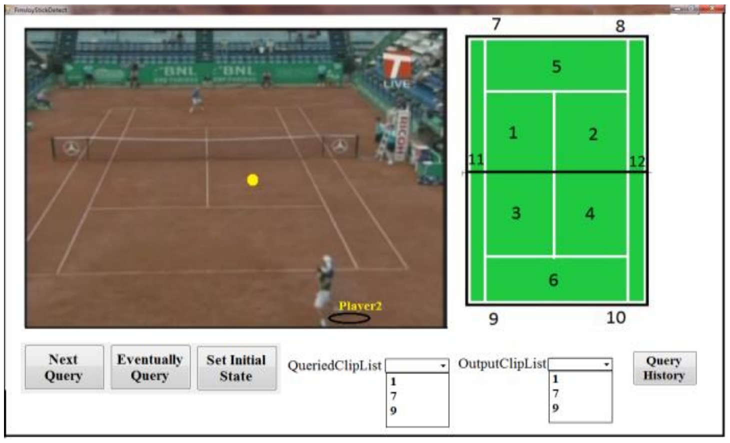

Figure 6: Snapshot after building first query

Query Results. Figure 7 shows the snapshot after executing a *subsequent* query which specifies (5, 8, 10) as the next state, Ω_N. Note that $\mho$ contains the union of clips retrieved by executing both queries (1, 7, 9, 13), and $\mathbb{Q}$ contains only those clips that are common to both queries (7, 9).

Journal of Visual Languages & Computing, Volume 29, August 2015, Pages 63–76,
doi:10.1016/j.jvlc.2015.03.003

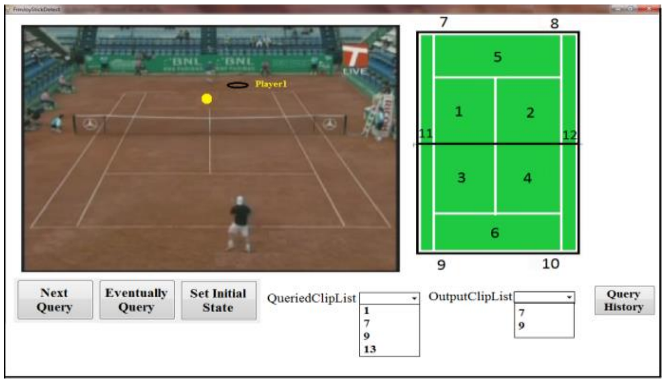

Figure 7: Snapshot after the execution of two queries starting from the default initial state

Setting the Final State for Eventual Query. For *eventual* query, the user selects "Eventually Query" feature on the UI. GStar displays icons for the players and the ball and allows the user to specify locations L_b, L_1 and L_2 for ball, $player_1$, and $player_2$ respectively, using the features of the gamepad. These locations are recorded as the initial state $\Omega_0(L_b, L_1, L_2)$. Figure 8 shows the snapshot in which the user has specified L_b as location 3, L_1 as location 8, and L_2 as location 6. Similarly, the final state Ω_F is specified as $\Omega_F(L_b, L_1, L_2) = (2, 5, 9)$ as shown in Figure 9. After the *eventual* query is executed the clips that contain the queried information appear in $\mathbb{Q}$ and $\mho$ as described before.

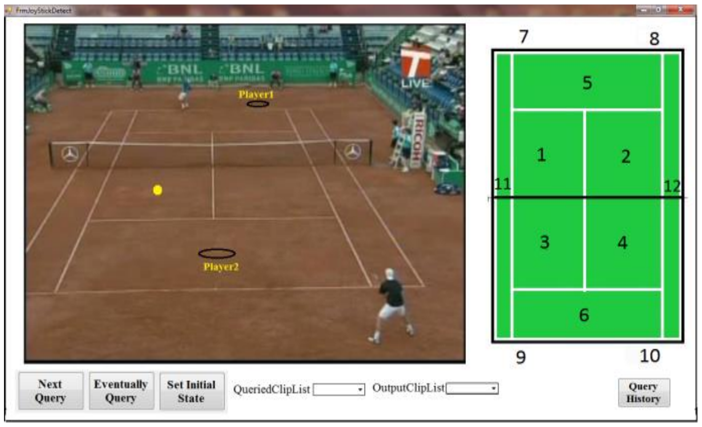

Figure 8: Snapshot after specifying the initial state for *eventual* query

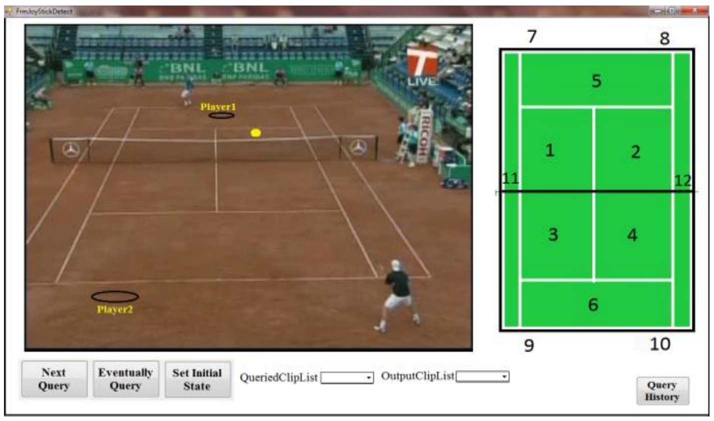

Figure 9: Snapshot after specifying the final state for *eventual* query.

The “Query History” feature on the UI helps the user visualize what has been queried and what to query next by providing a summarized view of the events being queried (Figure 10).

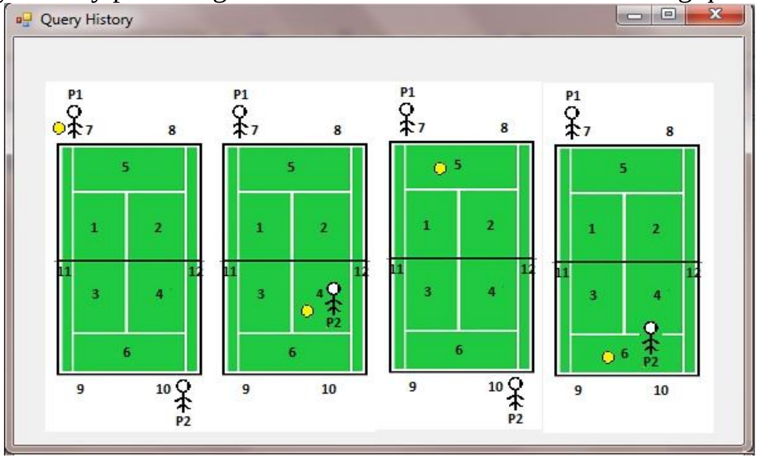

Figure 10: Snapshot of Query History window

6. Usability Study.

In this section, we evaluate the gamepad interface (as a gaming controller) whether it has an advantage over traditional mouse interface. The mouse interface (MI) uses point-and-click approach [2]. The underlying indexing structure is the same for both interfaces. The major differences are 1) the way inputs are taken from the users and 2) user interface for getting the inputs. The mouse interface provides a tennis court image with labels and enables selection of objects. The user first needs to select the object and the click the location on the court. To develop a state, this should be repeated for every object. On the other hand, GStar provides a small fragment of a tennis video to start and then the user provides input by controlling the objects (players and ball) in the video. The ISO standard (ISO, 1998) is used as the basis for the comparison study. The ISO defines usability of a product as the “extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction” (ISO, 1998).

6.1 The Usability Study Environment.

The study was conducted in the multimedia laboratory of the University of Alabama in Huntsville. To ensure that the study was fair and unbiased, ten students were randomly asked to participate in the study. The students had no prior knowledge about the two user interface systems or their developers. The familiarity with the gamepad or mouse interface was not considered while selecting users (students). There are three stages of our usability study: training, practicing, and testing.

Training. Two queries were developed for the purpose of training users so that they become familiar with the use of both user interface systems. First, each user was educated about spatio-temporal query, the two user interfaces, and the purpose of the study.

Practicing. After training, users were given two learning queries and help was given on an as needed basis to make them comfortable to use the user interfaces. A complex query is built by combining *subsequent* and *eventual* queries.

Testing. For testing purpose, five test queries of different complexity were developed and the result for each of the five queries was recorded by an expert. The complexity of the test queries ranged from a single *subsequent* query to a combination of *subsequent* and *eventual* queries. This information was used later to determine if the users were able to accurately query the database. Then the same test queries were given to each user and the user was asked to test both user interfaces. No help was provided about the use of the interfaces during testing. Each user was asked to use both UI in alternating order to minimize the bias due to the tendency of getting adjusted or tuned well to the user interface that was used first. All users were given unlimited amount of time to complete the task.

6.2 Usability Metrics.

The measures or metrics used to assess usability are grouped into three categories: effectiveness, efficiency and satisfaction. The effectiveness of a user interface is defined by ISO as the ability of the user to complete the specified task and obtain accurate results. Therefore, ability to complete task, accuracy of task completion (ability to retrieve correct clips), and quality-of-outcome which is the measure of extent to which the interface positively engages the user by providing good feedback during the querying process even before presenting the final result are selected as metrics for measuring effectiveness. The efficiency of a user interface may be defined as the resources such as time, cost, user's effort, etc. needed to achieve the goal. For the study, learning time (time for the new user to learn to use the interface), testing time (time to complete all 5 queries), and input rate (number of inputs provided by the user – number clicks, number of window closings, etc.) are chosen as metrics to measure efficiency. Satisfaction measures overall appreciation and opinion about the system. Preference (likelihood of using one interface over the other), ease-of-use (comfort in using the interface), and overall opinion are selected as metrics to measure user satisfaction.

6.3 Usability Study Results and Analysis.

This section presents the measurement data gathered for the assessment of effectiveness, efficiency, and satisfaction (the three measures of usability), and their analysis.

Effectiveness: In order to measure task completion ability, each user was given the same 5 queries. If the user completed the query, the outcome was recorded as "completed". Otherwise, the outcome was recorded

Journal of Visual Languages & Computing, Volume 29, August 2015, Pages 63–76,
doi:10.1016/j.jvlc.2015.03.003

as "not completed". The clips retrieved by the user were checked for accuracy and recorded as "correct" if it matched the stored result obtained by the expert. Otherwise, it was recorded as "incorrect". The results show that all users were able to complete the task and were able to retrieve correct clips using both interfaces.

Efficiency: Learning and testing time was measured in seconds. After the completion of all five queries, the user was asked to rank input rate on a scale from 1 to 5 (1 – very low, 2 – low, 3 – average, 4 – high, 5 – very high). The minimum, maximum, and the average learning time for the GI and MI were (300, 840, and 500) and (360, 1020, and 666) seconds, respectively. On an average, users took less time to learn gamepad interface over mouse interface. The minimum, maximum, and average testing time to complete all five queries using GI were (152, 780, and 331) and (198, 940, and 364) seconds, respectively. The average scores received for input rate by GI and MI were 2.2 and 4.1, respectively. Several features of the mouse interface contributed to high input rate. First, the user had to deal with many windows that popped-up during the query process. During the execution of complex queries which consist of several states, the user had to move from one interface window to another to build the query incrementally. Based on the measurement data, the users had better efficiency with the gamepad interface than the mouse interface.

Satisfaction: After completing all queries each user was asked to indicate his or her preference on a scale from 1 to 5 (1 – I definitely prefer MI, 2 – I prefer MI over GI, 3 – I have no preference, 4 – I prefer GI over MI, 5 – I definitely prefer 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. Finally, each user was asked to provide his or her opinion in his or her own words.

Nine users prefer GI over MI and one user had no preference. The average score for preference was 3.7/5.0 which shows that the users preferred the gamepad interface over the mouse interface. For ease-of-use, all users ranked GI high (4) and MI average (3), respectively. Since the sum of squares within groups is 0 and gamepad had better rating, the F-ratio becomes infinity for the ANOVA analysis. This rejects the null hypothesis and gamepad is considered better than mouse interface for ease-of-use. This indicates that the gamepad interface causes less user discomfort than the mouse interface. All users liked and appreciated immediate and automatic feedback of realistic result provided by the gamepad interface. One user said that GI provided smooth flow for building a query consisting of a sequence of states as there is no need to switch the interface window. Several users did not like the fact that too many windows pop-up while using the mouse interface. Therefore, GI provided higher level of satisfaction to users than MI.

6.4 Discussion.

Extendibility. GStar can easily be applied if the spatio-temporal database can be represented with S^3G. S^3G requires that the spatio-temporal database can be split into states and states are linked through transitions. The transitions may be annotated as in tennis example in this paper. Similar S^3G index can be built for sports games such as volleyball and table tennis, where players are restricted to a specific region. For other games, the discretization step is critical. Rather than building states for all players, states may be built for a team and ball object as in football (e.g., Team A at 49). For games like soccer, the video can be discretized based on when the game stops or based on specific actions of players.

Limitations of Mouse Interface. The current MI only provides a graphical layout of a tennis court to map objects to locations. It could have been improved by using the actual tennis footage. However, there is frequent camera motion in tennis videos. Therefore, the regions of locations change from one frame to another frame. This requires the alignment of frames or detection court lines in every frame. This was not required for the GI since there is a common region for a location in all frames. The gap between GI and MI interface may be reduced by also allowing MI to get inputs through the actual footage.

6. Conclusion.

This paper presents Query-by-Gaming with an innovative user friendly interface for retrieving the desired clips from tennis game video using a gaming controller. Our GStar allows the user to build spatio-temporal queries like subsequent and eventual. Spatio-temporal queries built are applied to a graph-based indexing structure called S^3G. GStar works efficiently in retrieving the clips by mapping the user input from gamepad to the corresponding semantic information of the video available in S^3G. GStar allows the user to continuously build the query and provides partial query results from the video immediately and automatically. GStar also maintains a history of all queries built by the user for the current session. A usability study was result showed majority of the users preferred the gamepad interface over mouse interface. Our experimental setting had 10 students and 5 queries per interface. The true evaluation will turn out from the experiences of real users of the interface.

Our usability study shows that Query-by-Gaming yields promising results for building spatio-temporal queries. As future work, Query-by-Gaming should be experimented on other types of spatio-temporal databases. Our index structure, S^3G, is appropriate for Query-by-Gamepad since sequence of states can be tracked instantly. The applicability of Query-by-Gaming should be studied on other spatio-temporal index structures. The spatio-temporal queries in GStar are not typical range queries (e.g., find objects at a specific region within an interval). Query-by-Gaming is especially useful for evaluating queries that can be represented with linear temporal logic. The usability study in our experimental setting provided promising results. As future work, the usability study should be conducted when such a system is deployed for real environments. Moreover, gaming controllers provide trigger buttons that can sense how hard the button is pressed. Such triggers can be used to build queries that may require a numeric attribute value (e.g., speed of a ball or a car). Future work may utilize triggers for more complex queries.

Acknowledgement

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

References

[1] Jain, V.; Aygun, R., "SMART: A grammar - based semantic video modeling and representation," *Southeastcon, 2008. IEEE* , vol., no., pp.247,251, 3-6 April 2008.

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

[3] Erwig, M. and Schneider, M., “Spatio-Temporal Predicates”. IEEE Trans. on Knowl. and Data Eng. vol. 14, no. 4 (Jul. 2002), pp. 881-901.

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

[5] Cavalcanti, V. M., Schiel, U., and de Souza Baptista, C, “Querying spatio-temporal databases using a visual environment”, In Proceedings of the Working Conference on Advanced Visual interfaces (Venezia, Italy, May 23 - 26, 2006). AVI '06. ACM, New York, NY, pp.412-419.

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

[7] T. Minka, “An image database browser that learns from user interaction,” M.I.T. Media Lab. Perceptual Computing Section, TR 365, 1996.

[8] K. Hirata and T. Kato, “Query by visual example, content based image retrieval,” in Advances in Database Technology—EDBT’92, A. Pirotte, C. Delobel, and G. Gottlob, Eds., Lecture Notes in Computer Science, vol. 580.

[9] Vani Jain, Ramazan S. Aygun, “Spatio-Temporal Quering of Video Content Using SQL for Quantizable Video Databases”, Journal of Multimedia. Vol 4, No 4 (2009), pp. 215-227, Aug 2009.

[10] Mitesh Naik, Madhav Sigdel, Ramazan Savas Aygün. Spatio-temporal querying recurrent multimedia databases using a semantic sequence state graph. Multimedia Syst. 18(3): 263-281 (2012)

[11] Vineetha Bettaiah, Ramazan Aygun, “Querying and retrieval of eventually type spatio-temporal query using gamepad”, Proc of MMEDIA 2011 - The Third International Conference on Advances in Multimedia, April 2011.

[12] Giuseppe Della Penna, Daniele Magazzeni, and Sergio Orefice. 2013. A general theory of spatial relations to support a graphical tool for visual information extraction. J. Vis. Lang. Comput. 24, 2 (April 2013), 71-87. DOI=10.1016/j.jvlc.2012.11.002

[13] Chenglang Lu; Mingyong Liu; Zongda Wu, "A visual graph-based query interface for video databases," Software Engineering and Service Science (ICSESS), 2014 5th IEEE International Conference on , vol., no., pp.577,580, 27-29 June 2014

[14] LuíS Certo, Teresa GalvãO, and José Borges. 2013. Time Automaton: A visual mechanism for temporal querying. J. Vis. Lang. Comput. 24, 1 (February 2013), 24-36.

[15] Moshe Y. Vardi. 2001. Branching vs. Linear Time: Final Showdown. In Proceedings of the 7th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2001), Tiziana Margaria and Wang Yi (Eds.). Springer-Verlag, London, UK, UK, 1-22.

[16] Alberto Del Bimbo, Enrico Vicario, and Daniele Zingoni. 1995. Symbolic Description and Visual Querying of Image Sequences Using Spatio-Temporal Logic. IEEE Trans. on Knowl. and Data Eng. 7, 4 (August 1995), 609-622.

[17] d'Acierno, A.; Leone, M.; Saggese, A.; Vento, M., "A system for storing and retrieving huge amount of trajectory data, allowing spatio-temporal dynamic queries," Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on , vol., no., pp.989,994, 16-19 Sept. 2012

[18] Magdy, A.; Aly, A.M.; Mokbel, M.F.; Elnikety, S.; Yuxiong He; Nath, S., "Mars: Real-time spatio-temporal queries on microblogs," *Data Engineering (ICDE), 2014 IEEE 30th International Conference on* , vol., no., pp.1238,1241, March 31 2014-April 4 2014

[19] Ferreira, N.; Poco, J.; Vo, H.T.; Freire, J.; Silva, C.T., "Visual Exploration of Big Spatio-Temporal Urban Data: A Study of New York City Taxi Trips," *Visualization and Computer Graphics, IEEE Transactions on* , vol.19, no.12, pp.2149,2158, Dec. 2013

[20] Rongjian Lan, Michael D. Lieberman, and Hanan Samet. 2012. The picture of health: map-based, collaborative spatio-temporal disease tracking. In *Proceedings of the First ACM SIGSPATIAL International Workshop on Use of GIS in Public Health* (HealthGIS '12). ACM, New York, NY, USA, 27-35.

[21] Jalal S. Alowibdi, Sohaib Ghani, and Mohamed F. Mokbel. VacationFinder: A Tool for Collecting, Analyzing, and Visualizing Geotagged Twitter Data to Find Top Vacation Spots. ACM SIGSPATIAL LBSN'14, November 4, 2014, Dallas, TX, USA

[22] Rongjian Lan, Marco D. Adelfio, and Hanan Samet. Spatio-Temporal Disease Tracking Using News Articles, 3rd ACM SIGSPATIAL International Workshop on the Use of GIS in Public Health (HealthGIS) 2014

[23] Arianna D'Ulizia, Fernando Ferri, and Patrizia Grifoni. Moving GeoPQL: a pictorial language towards spatio-temporal queries, Geoinformatica (2012) 16:357–389

[24] Takeuchi, S.; Akahoshi, Y.; Ong, B.T.; Sugiura, K.; Zettsu, K., "Spatio-temporal Pseudo Relevance Feedback for Large-Scale and Heterogeneous Scientific Repositories," *Big Data (BigData Congress), 2014 IEEE International Congress on* , vol., no., pp.669,676, June 27 2014-July 2 2014

[25] Laura Díaz, Carlos Granell, Joaquín Huerta, Michael Gould, Web 2.0 Broker: A standards-based service for spatio-temporal search of crowd-sourced information, Applied Geography, Volume 35, Issues 1–2, November 2012, Pages 448-459

Journal of Visual Languages & Computing, Volume 29, August 2015, Pages 63–76,
doi:10.1016/j.jvlc.2015.03.003

[26] Pablo Bisceglia, Leticia G´omez, and Alejandro Vaisman. Temporal SOLAP: Query Language, Implementation, and a Use Case, AMW 2012: 102-113

[27] Mesru Koprulu, Nihan Kesim Cicekli, Adnan Yazici, Spatio-temporal querying in video databases, Information Sciences, Volume 160, Issues 1–4, 22 March 2004, Pages 131-152

[28] C. Parent, S. Spaccapietra, E. Zimányi, The MurMur project: Modeling and querying multi-representation spatio-temporal databases, Information Systems, Volume 31, Issue 8, December 2006, Pages 733-769

[29] Mohamed F. Mokbel, Xiaopeng Xiong, Moustafa A. Hammad, and Walid G. Aref. Continuous Query Processing of Spatio-Temporal Data Streams in PLACE, GeoInformatica, December 2005, Volume 9, Issue 4, pp 343-365

[30] Flickner, M.; Sawhney, H.; Niblack, W.; Ashley, J.; Qian Huang; Dom, B.; Gorkani, M.; Hafner, J.; Lee, D.; Petkovic, D.; Steele, D.; Yanker, P., "Query by image and video content: the QBIC system," *Computer* , vol.28, no.9, pp.23,32, Sep 1995

[31] Asif Ghias, Jonathan Logan, David Chamberlin, and Brian C. Smith. 1995. Query by humming: musical information retrieval in an audio database. In *Proceedings of the third ACM international conference on Multimedia* (MULTIMEDIA '95). ACM, New York, NY, USA, 231-236.

[32] Ramez Elmasri and Shamkant B. Navathe. Fundamentals of Database Systems, 6/E, Addison-Wesley, 2011.

[33] J. Chen, A. G. Cohn, D. Liu, S. Wang, J. Ouyang, and Q. Yu. A survey of qualitative spatial representations. The Knowledge Engineering Review, FirstView:1-31, 11 2014