MotionFlow: Visual Abstraction and Aggregation of Sequential Patterns in Human Motion Tracking Data

Sujin Jang, Niklas Elmqvist, *Senior Member, IEEE*, and Karthik Ramani

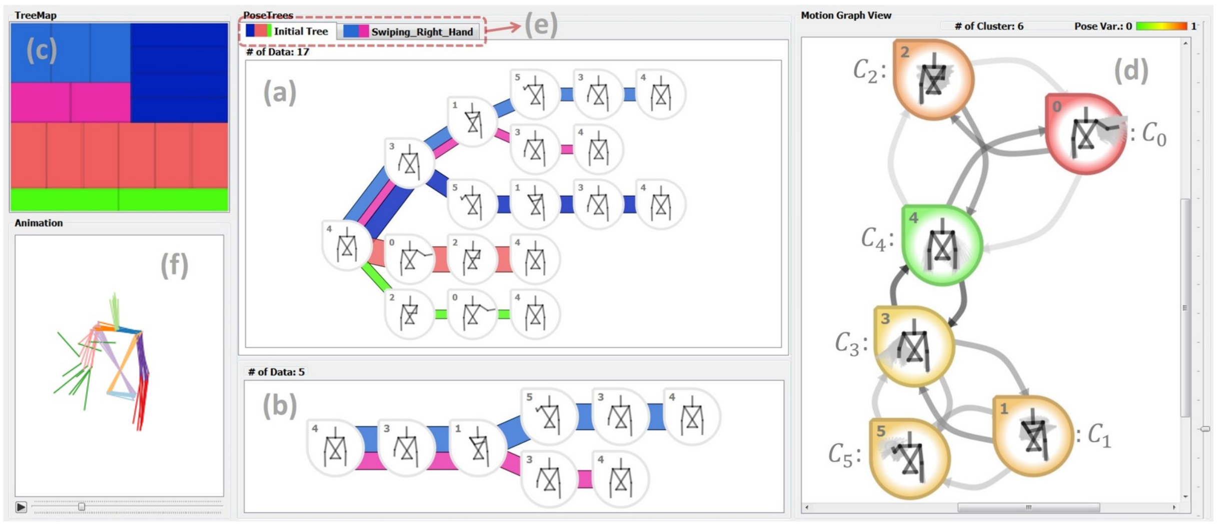


Fig. 1. MotionFlow for pattern analysis of human motion data. (a) Pose tree: a simplified representation of multiple motion sequences aggregating the same transitions into a tree diagram. (b) A window dedicated to show a subtree structure based on a query. (c) Space-filling treemap [32] representation of the motion sequence data using slice-and-dice layout. (d) Node-link diagram of pose clusters (nodes) and transitions (links) between them. This view supports interactive partition-based pose clustering. (e) Multi-tab interface for storing unique motion patterns. (f) Animations of single or multiple selected human motions.

Abstract—Pattern analysis of human motions, which is useful in many research areas, requires understanding and comparison of different styles of motion patterns. However, working with human motion tracking data to support such analysis poses great challenges. In this paper, we propose *MotionFlow*, a visual analytics system that provides an effective overview of various motion patterns based on an interactive flow visualization. This visualization formulates a motion sequence as transitions between static poses, and aggregates these sequences into a tree diagram to construct a set of motion patterns. The system also allows the users to directly reflect the context of data and their perception of pose similarities in generating representative pose states. We provide local and global controls over the partition-based clustering process. To support the users in organizing unstructured motion data into pattern groups, we designed a set of interactions that enables searching for similar motion sequences from the data, detailed exploration of data subsets, and creating and modifying the group of motion patterns. To evaluate the usability of MotionFlow, we conducted a user study with six researchers with expertise in gesture-based interaction design. They used MotionFlow to explore and organize unstructured motion tracking data. Results show that the researchers were able to easily learn how to use MotionFlow, and the system effectively supported their pattern analysis activities, including leveraging their perception and domain knowledge.

Index Terms—Human motion visualization, interactive clustering, motion tracking data, expert reviews, user study.

1 Introduction

Recent development of reliable and low-cost sensing technologies now enable accurate and convenient acquisition of human motion tracking data. Human motion tracking data typically consists of temporal sequences of human poses that are defined by a set of 3D joint positions (e.g., head, hands, elbows, and knees). These data support researchers in performing detailed and comprehensive analysis of a variety of complex human motions. In particular, analyzing common and irregular motion patterns across subjects and environments can help researchers better understand human behavior in real-world situations. For example, comparative analysis of motion patterns may indicate differences between novice and professional dancers [2]. In learning environments, studying gesture patterns of students can lead to deeper understanding of the role of gestures in the learning process [7]. Similarly, the analysis of repetitive and irregular motion patterns can be used in a rehabilitation process to diagnose and correct patients' undesired movement [46]. This style of analysis requires understanding motion sequence data, identifying main trends, and detecting anomalies by properly aggregating similar sequences into pattern groups.

However, such analysis is challenging for many reasons: (1) human motion data is multivariate with an almost infinite range of spatial and temporal variations; (2) motion sequences commonly consist of hundreds or thousands of consecutive frames; and (3) different individuals often exhibit small variations in their motions representing the same

- *Sujin Jang and Karthik Ramani are with Purdue University in West Lafayette, IN, USA. E-mail: {jang64, ramani}@purdue.edu.*
- *Niklas Elmqvist is with University of Maryland in College Park, MD, USA. E-mail: elm@umd.edu.*

Manuscript received 31 Mar. 2015; accepted 1 Aug. 2015; date of publication xx Aug. 2015; date of current version 25 Oct. 2015. For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

general patterns. While visual analytics has previously been applied to this problem—e.g. MotionExplorer [5] and GestureAnalyzer [18]—these solutions are not sufficiently flexible, scalable, and visually compact to fully support motion pattern analysis.

To meet this need, we designed *MotionFlow*, an interactive visual analytics system that supports pattern analysis of sequential motion data. Compared to state-of-the-art systems [5, 18], it supports efficient but effective visualization of multiple motion patterns, and reflects the context of data and human perception of pose similarity into pattern analysis. MotionFlow defines each motion sequence as a series of transitions between pose states. Then, it combines the motion sequences sharing the same pose transitions into a *pose tree*: a tree diagram where each node represents a specific pose state and edges define transitions between pose states. We adopt a *flow* visualization technique to provide an overview of the pose transition patterns in the pose tree. In this visualization, the frequency of pose transition is encoded using the thickness of edges. To define representative pose states, MotionFlow supports interactive partition-based pose clustering. In this approach, each pose cluster is represented by a *poselet* where poses included in the cluster are displayed inside of a tear-drop like glyph. Transitions between pose clusters are drawn in a directed cyclic graph where the user can explore pairwise similarity among clusters, internal cluster similarity, and transition frequency. Users can use such information along with their intuition and perception of pose similarity during clustering. The structure of graph is driven by the user by manipulating pose clusters through both local and global interaction techniques such as *split*, *merge*, *lock* of cluster nodes (local) and *partition number control* (global). Users can also select and investigate individual or group of motion data in patterns through a *treemap* and *animation* view. As a result of pattern analysis, MotionFlow generates an organized motion tracking database where similar motion sequences are aggregated into a set of patterns, and stored in multi-tab windows. In summary, the key contributions in this paper are as follows:

- An approach to visualize sequential motion patterns that preserves spatio-temporal information and supports visual analysis of transitions between pose states;
- A set of interaction techniques for visual abstraction, exploration, and organization of unstructured sequential motion data into a set of patterns; and
- Results of an expert review conducted on MotionFlow.

In this paper, we present MotionFlow in the context of *gesture pattern studies*, wherein comparative pattern analysis of natural human gestures is enabled. In order to understand the usefulness of our approach in practical analytics tasks, we evaluated MotionFlow with six domain experts, focusing on the role of visualization components.

2 Related Work

Below, we review topics related to the analysis of human motion tracking data, visualization of sequential data, and interactive clustering.

2.1 Visual Analytics for Human Motion Tracking Data

So far, only a few visual analytics systems have been proposed to help researchers better understand and analyze human motion tracking data. To our best knowledge, Bernard et al. was the first to introduce such a system, *MotionExplorer* [5], which supports interactive search and retrieval of target motion sequences based on an interactive hierarchical clustering of motion tracking database. Although MotionExplorer helps users generate and analyze meaningful subsets of a large database, it does not fully support aggregation of similar sequences into a set of pattern groups. To understand style variations in the retrieved sequences, this system requires manual exploration and comparison of multiple motion data. In this context, manual analysis can be challenging. The interactive clustering technique introduced in MotionExplorer is helpful in providing an overview of data structure in a tree diagram by aggregating similar pose data. However, the tree structure is purely decided by the pose similarity measure, and users are only able to control the global aggregation level (i.e., adjusting the number of clusters). Thus, the users cannot locally manipulate any aggregations that conflict with their perception of pose similarity. For example, there could be an unnecessary splitting of pose clusters during adjusting the global level control, while there also could be undesired merging of dissimilar pose clusters. Further, MotionExplorer employs a motion graph [20] approach to visualize motion patterns. The motion graph is a node-link diagram where nodes represent static human poses while links define sequences of human poses. Although, it provides a simplified overview of multiple motion patterns, when the motion patterns being displayed become complex and diverse, it is hard to recognize complete transitions (i.e., order of node occurrence). In short, it does not provide a clear start and end to each sequence.

The closest existing work to MotionFlow is *GestureAnalyzer* [18], a visual analytics system for interactive aggregation of sequential motion data based on time-series similarity. However, similar to MotionExplorer, GestureAnalyzer uses a fixed cluster structure, so there is no way to change the aggregation of motion sequences at a local level. This system provides an overview of multiple sequential motion data through small-multiple visualization. Since each motion data is represented by a single row, there could be redundant pose block across the entire visualization. Thus, this approach suffers from information overload when it tries to display large amount of sequential motions. Also, GestureAnalyzer does not allow users to define representative human poses that reflect their perception of pose similarity and representativeness for an effective overview of motion trends. In contrast, MotionFlow supports comparative pattern analysis of motion sequences, and allows for organizing them into meaningful pattern groups while leveraging human perception and domain knowledge. Furthermore, MotionFlow enables users to directly control aggregation through a set of direct manipulation techniques (e.g., splitting and merging clusters) when defining representative poses. Finally, MotionFlow formulates sequential motion patterns into a state transition diagram in a tree form to eliminate redundant information, and provide more concise and simple overview of multiple sequential motions.

2.2 Sequential Pattern Visualization

A sequential pattern is commonly defined by a series of temporal events (e.g., A $\rightarrow$ B $\rightarrow$ C where A, B, and C stand for an event occurrence at a given time). There exists a vast body of work in the visualization of sequential patterns for different application areas. The most relevant approach to our system is a flow visualization. Here, different quantities of flows are represented by the width of edges that connect nodes where specific states of data is represented. Riehmann et al. [28] introduced interactive *Sankey diagrams* to visualize different flow quantities through the size of edges. Similar techniques have been applied in visualizing various contexts of data such as social media [45], storytelling [22], text [11], and temporal events [43].

The tree diagram is one of the common approaches to visualize multiple pathways. Word trees [40] visualize a series of words appearing in multiple sentences by spatially aggregating and locating them in a tree diagram, providing an effective way to search and explore specific path ways from a complex text-based database. Timeline trees [10] provide an overview of multiple sequential transaction data through a hierarchical tree diagram and a timeline visualization. eSeeTrack [34] takes a similar approach to visualize sequential patterns of eye fixations. LifeFlow [44] provides an overview of multiple sequential patterns by aggregating the same pathway into a tree diagram.

While our visualization design is inspired by both tree diagrams and flow visualizations, we define each node with representative human pose states defined by users' perception and domain knowledge. To visualize multiple sequential motion patterns, MotionFlow aggregates the pose states into a tree structure, providing an effective but efficient overview of multiple sequential motion patterns.

2.3 Visual Abstraction of Sequential Data

Motion tracking data is a series of multivariate, continuous human poses where infinite spatial and temporal variations can exist. To provide an overview of multiple sequential motions, we need to consider how to properly abstract and simplify such highly variable data. There exists work in simplifying sequential event data that is discrete and concurrent (e.g., medical event, transaction history). OutFlow [43] applied a hierarchical clustering method to reduce the number of states in the flow visualization by combining similar event states based on a user-defined merging threshold. Monroe et al. [23] introduced a simplification of sequential event data by replacing and aligning longer and complex event sequences with respect to a set of user-defined sub-sequences. Similarly, DecisionFlow [14] provides an overview of high-dimensional sequential event data with several representative events. The approach provides an abstraction of a complex data set, and then lets the user progressively explore details of the sequence.

A state transition graph is an effective way to visualize multiple pathways of sequential events by considering temporal changes as state transitions [8]. Ham et al. [36] introduced a visualization for state transitions in a 3D space tree structure. Here, to reduce visual complexity, state nodes sharing the same ancestor nodes are clustered into a sub-tree structure. However, this approach does not provide users a direct control of state definitions as they are automatically defined. Pretorius et al. [26] proposed a bar tree to abstract multivariate state transitions into 2D space. Blaas et al. [6] introduced an approach to explore higher-order state (i.e., more than three states) transition patterns by aggregating transitions into spline bundles in a 2D graph layout. These approaches provide an overview and detail of specific transition patterns. However, they do not provide capability to compare multiple transition patterns based on time.

In this paper, we formalize pattern analysis of human motion into an interactive state transition visualization. Unlike existing work in state visualization, we provide the users direct manipulation of states to properly abstract complex and massive sequential motion into a manageable overview, then progressively define each transition pattern at a local level through an interactive clustering approach.

2.4 Interactive Data Clustering

Data clustering techniques cater to a wide number of research questions from a broad set of disciplines. Traditionally, clustering algorithms are considered an unsupervised machine learning where an unknown data set is clustered into a set of meaningful groups (see Berkhin et al. [4] for a review). However, there has been considerable efforts to turn clustering algorithms into supervised learning by integrating users domain knowledge into the clustering process.

In the data mining community, researchers have introduced the concept of constrained clustering providing a role for the user in the clustering process [12, 38, 39]. In these efforts, users define pairwise cluster constraints such as "must be" and "must not be" in the same cluster. Balcan and Blum [3] discussed interactive feedback in data clustering process. They described a pair of natural clustering processes—*split* and *merge*—and provide algorithmic requirements with bounded number of split and merge. However, these approaches do not support dynamic manipulation of the clustering process due to the lack of user interactions. Also, in addition to Balcan and Blum's local split/merge clustering metaphor, we also provide a global clustering control.

In visual analytics, clustering algorithms have been actively integrated with interactive visualizations to provide a control over the clustering process. Hierarchical clustering generates aggregation results in the form of tree diagrams, and most existing work allows users to steer the aggregation level by adjusting the depth of the tree structure [15, 31]. gCLUSTO [27] is a software package for testing various algorithms through a visual investigation. Ahmed et al. [1] integrated partition-based clustering with multiple coordinate visualizations. Schultz et al. [30] proposed an interactive spectral clustering approach to segment 3D imagery data. These approaches incorporate experts into data clustering processes, and require users to have background knowledge to properly control algorithmic parameters.

To our best knowledge, Hossain et al. [17] proposed a concept most similar to our work, with interaction techniques that enable users with limited expertise in clustering algorithms to directly control the aggregation process through *scatter* and *gather* interactions. These interaction techniques support iterative modification of clustering results through a user-defined constraint matrix. However, such tabular input is less intuitive than direct cluster manipulation.

Our work focuses on providing an interactive clustering technique that supports users who have limited expertise in data mining to adeptly insert their domain knowledge and perception into the pose clustering process. To provide an effective interaction techniques in manipulating the clustering process, MotionFlow features both global and local cluster controls.

3 Analysis of Human Motion Tracking Data

The use-case driving the design of MotionFlow is gesture pattern studies in human-computer interaction (HCI). After reviewing this background below, we then identify requirements and rationale for the system design based on close collaboration with domain experts.

3.1 Background: Gesture Pattern Studies

In HCI, a *gesture* is commonly defined by *"a motion of the body that contains information"* [21]. Human gestures are actively and increasingly being used to support *gesture-based interaction* (e.g., mobile interaction [29], virtual 3D object design [37], and automobile interaction [25]). In general, development of such interaction systems requires deep understanding of natural aspects in human gestures to increase naturalness and intuitiveness of the interactions, and reflecting this knowledge into gesture recognition [41]. Thus, researchers in gestural interaction design must be experts in either one or both of HCI and pattern recognition. Such gesture interaction researchers are the target audience for our work in this paper.

To observe and understand natural patterns in human gestures, researchers commonly perform gesture pattern analysis through *elicitation studies* [42]. Gesture elicitation studies collect natural and common trends in human gestures by categorizing similar gestures into a set of groups based on perception, background knowledge, and interaction contexts. Motion tracking data is central to this analysis, and researchers apply automated computational methods to aggregate gestures based on computational similarity [18]. For example, when a researcher wants to study how people express gestures to control a virtual object on a big screen, he/she may capture natural gestures elicited from the candidate user groups. Through exploration and comparative analysis of gesture patterns, the researcher can answer questions such as, "What is the most common and similar gesture pattern expressed among the users?", or "How much variation is observed in the gesture patterns?". This kind of analysis requires annotation and grouping of similar gestures based on interaction context. Researchers also need to iteratively generate and modify gesture categorization criteria.

3.2 Method: Requirements Analysis

To collect requirements for a system based on the analysis procedures and tasks of gesture pattern studies, we collaborated with three HCI researchers with long experience in gestural interaction design, including one of co-authors of this paper. We had discussions with the collaborators every week, and demonstrated the initial design of MotionFlow to gather their opinions and feedback. We also shared visual design of the most relevant existing system, GestureAnalyzer [18] and MotionExplorer [5], so that the collaborators are able to understand limitations of the system, and reflect them in generating requirements.

3.3 Design Requirements and Rationale

During the requirements analysis process, we identified a list of analytical requirements that MotionFlow should support in order to scaffold pattern analysis of human gestures in HCI research:

- R1 Visualizing *multiple motion data sequences* to support exploration and understanding of gesture motion patterns;
- R2 Selecting *interesting gesture patterns* for detailed analysis;
- R3 Comparing gestures to *identify similar and dissimilar gestures*;
- R4 Supporting users to *reflect their domain knowledge and interaction contexts* into aggregating similar gesture data; and
- R5 Generating pattern analysis results in a form so that *findings and insights can be shared* with other researchers and domains.

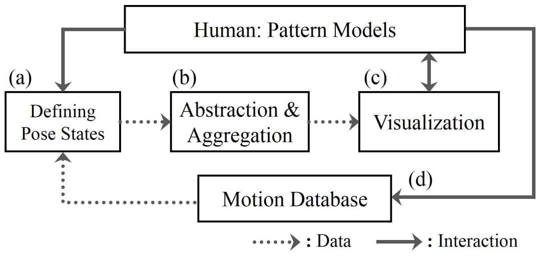


Fig. 2. System workflow. (a) Users first define representative pose states. (b) The system simplifies the motion data into sequences of the states, and aggregates them into pattern visualizations. (c) Users interact with visualizations to explore motion data, and iteratively specify pattern models involving similarity of motions. (d) Based on the pattern models, users organize the database. The cycle is repeated as needed.

In particular, all domain experts expressed the need to have a global overview of multiple gesture data, and then be able to decide which portion of data should be further explored and analyzed (R1, R2, R3). Although tools such as GestureAnalyzer [18] provide a detailed view of multiple gesture datasets, combining them into a single visualization in an efficient way is not a trivial problem. MotionExplorer [5] features a motion graph view to integrate multiple motion data into a directed graph. However, it suffers from edge crossings and lacks a complete view of gesture sequences. To resolve such issues, we should consider how to efficiently eliminate and effectively simplify overlapping information. In practical scenarios, categorizing gestures cannot be formulated as a single exploration process. Instead, it commonly involves iterative refinement and modification of the current aggregation based on domain knowledge and the interaction context (R4). Similarly, the search and retrieval of subsets of motion data provided in MotionExplorer [5] does not support flexible and user-driven pattern analysis to generate motion patterns. To support such aggregation, a new system should consider progressive and dynamic modification of gesture pattern groups. The domain experts also expressed a desire to have a summary of the analysis results (R5), so that the value of results can be transferred to support other researchers and their work, e.g. designing pattern classifiers for identified gestures.

4 The MotionFlow System

MotionFlow consists of four key components combined to support the workflow in Figure 2: (1) user-driven pose states clustering, (2) a tree diagram with flow visualization, (3) progressive organization of motion patterns, and (4) user interactions. Here, we first define the data model used in the system. Then, we provide details on the visual and interaction design, and how each analytical requirement is supported.

4.1 Data Model

MotionFlow is based around a gesture database recorded during elicitation studies with mid-air hand gestures [18]. This database consists of a variety of natural human gestures, each representing a single trial of meaningful human activity (i.e., a trial defined by a distinct starting and ending pose). We use such a dataset as a source of motion data for evaluating our system in supporting practical gesture analysis tasks. This database includes a total of 816 clips recorded by 17 participants while performing 48 gestures as inputs to gestural interaction systems. A Microsoft Kinect[1] camera was used to capture 3D coordinates of 11 upper-body joints (hands, elbows, shoulders, head, neck, chest, and pelvis) at 30 frames per second. A collection of motion data is loaded to the system as a binary file without annotating gesture style.

Motion Data. We define a *motion*, $\vec{J} = (id, [j_1, j_2, ..., j_n])$ as an identifier of human subject *id* and a sequence of *n* human poses where j_i is a pose at *i*th frame.

Feature Descriptor of Human Poses. We define the feature descriptor of poses $\vec{X} \in \mathbb{R}^{3*d}$ as $[(x_1,y_1,z_1),(x_2,y_2,z_2),...,(x_d,y_d,z_d)]$ where d is the number of tracked body joints and (x_i,y_i,z_i) is a 3D position of i-th body joint. Here, we are not concerned with improving clustering results through improved feature descriptors, but rather with involving human perception in the clustering process. For this purpose, we use a simple distance measure, Euclidean distance, to evaluate similarity between poses.

[1]https://www.microsoft.com/en-us/kinectforwindows/

Pose States. As a result of pose clustering (Section 4.2), we obtain a set of representative human poses, $\vec{C} = [c_1, c_2, c_3, ..., c_m]$ where c_j is a *pose state* representing the j-th pose cluster centroid.

Motion Sequences. Each motion pose belongs to a pose state, and we partition the motion data into k segments with respect to the frame where the pose state changes. Then, we replace each motion data segment with the corresponding pose state. We call this simplified data representation a *motion sequence*, $\vec{S} = (id, [s_1, s_2, ..., s_k])$ where s_i is a pose state representing the i-th pose segment and *id* as its identifier.

Motion Patterns. Multiple individual motion sequence can be aggregated into a *motion pattern*, $\vec{P} = \left\{\vec{S}_1, \vec{S}_2, \vec{S}_3, ...\vec{S}_l\right\}$ where l sequences are grouped together. Each motion sequence has a unique identifier *id*, and shares pose states $[s_1, s_2, ..., s_k]$ with other sequences.

4.2 User-Driven Pose State Clustering

To provide an overview of the gesture data, we formalize motion data with sequential events where each event represents a human pose (R1). As seen in Figure 2, users begin by analyzing motion data to generate representative pose states. Users are involved in the clustering process by subjectively surveying all pose states and their similarities.

4.2.1 Partition-Based Clustering

Our target users do not necessarily have significant background in data mining techniques. So, we aim at providing an easy-to-understand and intuitive-to-control pose clustering process. There exist substantial work in clustering multivariate data such as hierarchical, spectral, density-based, and partition-based clustering (see [4] for a review). Even though our user-driven clustering approach can be applied in most of the state-of-the-art clustering methods, we decided to use a partition-based clustering, K-Means, in the interactive clustering. The K-Means clustering approach is simple enough to understand and it is easy-to-manipulate the output clusters by simply adjusting the number of partitions. Hierarchical clustering also provides a simple way of adjusting clustering process; however, the structure of clustering is pre-determined, and the users cannot alter it progressively. We initialize centroids of K-Means to be uniformly distributed over the pose space having maximized sum of Euclidean distance.

4.2.2 Poselets

We define a *poselet* to be the visual entity encoding a pose cluster in MotionFlow (Figure 3). Poselets are a glyph-based approach consisting of a stick-figure representation of a human pose [5]. The stick figure is centered in a circle, and other similar poses are displayed as semi-transparent ghosts around the center pose. Cluster IDs are given in the top-left corner of poslets. The variance of pose clusters is normalized into $[0,1]$, and visually encoded using a gradient color scheme (green-to-yellow-to-red). For example, in Figure 3, pose state c_2 has a higher variance (red boundary), while c_3 indicates a lower variance (green boundary). To annotate and classify human body poses, Bourdev and Malik [9] defined a poselet as an encoding of 3D joint information of human body into a 2D representation. Projecting a 3D pose into 2D space involves a loss of depth information. Each poselet has

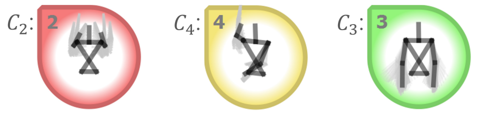


Fig. 3. *Poselets* of human pose clusters. A tear-drop like shape is used as the outline of poselet. At the center of a poselet, the centroid pose is drawn. The other poses are displayed using semi-transparent strokes.

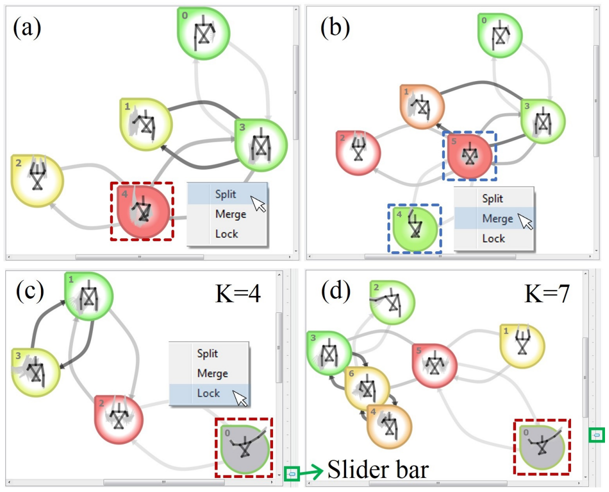


Fig. 4. Interactions on pose clustering. (a) A pose cluster (red box) is selected for a split, and (b) is separated into two pose clusters (blue boxes). They can be merged back to retrieve the original cluster in (a). (c) Dragging a slider-bar (green box) adjusts the number of clusters (K) while a pose cluster (red box) is locked. (b) The locked cluster remains unchanged while the other clusters are regenerated.

specific perspectives where the spatial information is well-reflected into 2D space. However, it is hard to automatically generate such a view for a general pose. To mitigate this issue, the user can control the view of poselets by dragging the mouse around each poselet.

4.2.3 Pose States Graph

Figure 1(d) shows an interface for user-driven pose states definition. Here, we provide an overview of pose clustering, and let users progressively manipulate the pose clusters. For an overview of pose clusters and their sequential relationship, we use a *motion graph*, a directed graph where nodes represent pose clusters and edges encode motion sequences [20]. Within the graph layout, we consider distances among cluster nodes to encode pairwise similarity among clusters. To visually encode such information, we employed a force-directed graph layout algorithm [19] where the dynamics among cluster nodes is defined by a spring force model. When a dataset is loaded, MotionFlow computes the Euclidean distance between pose clusters, then the pairwise distances define the forces between the nodes in the spring model. As shown in Figure 1(d), we use spatial proximity to encode the similarity of pose clusters (e.g., c_1, c_3, c_5 are close to each other while c_2 and c_5 are not). The edges in the motion graph visually encode the directed transition frequency between two pose states using a color gradient. This color-coding is intended to highlight frequent transitions (e.g., dark gray colored edges encodes frequent transitions between c_3 and c_4, while infrequent transitions are observed between c_3 and c_5).

4.2.4 Interactive Cluster Control

Since the pose states are directly used in expressing sequential motion data, it is important to extract appropriate key pose states that do not conflict with the users' perception and domain knowledge. We designed three interaction techniques to support dynamic and progressive refinement of pose clustering guided by the users.

Split, Merge, and Lock. MotionFlow provides the interactions *split* and *merge*, a pair of dynamic manipulation methods to control clustering at a local level. Split partitions an overcrowded pose cluster into two partitions, while merge combines multiple clusters into a single node (Figure 4(a),(b)). This pair of cluster manipulations enables users to dynamically correct local clustering results, and directly reflect their perception of human pose similarity. Such manipulations change the layout of motion graph since new nodes are generated. To help users to keep track of such dynamic changes, we adopt animated transitions among node positions before and after manipulation. Often, we want to preserve some cluster nodes, while other nodes are manipulated. To support such cases, MotionFlow provides *lock* of the node in interests indicating user intention of no further split, merge, nor re-clustering of the node (Figure 4(c)). As seen in Figure 4, users can select individual or multiple nodes, and right-clicking on the node activates a pop-up menu listing possible manipulations.

Adjusting Partition Numbers. MotionFlow also allows the user to control the number of clustering partitions by adjusting a vertical slider (Figure 4(c),(d)). When the slider is changed, a new set of pose clusters is generated, and previous nodes are replaced with a new one.

4.3 Visual Motion Concordance: Pose Tree

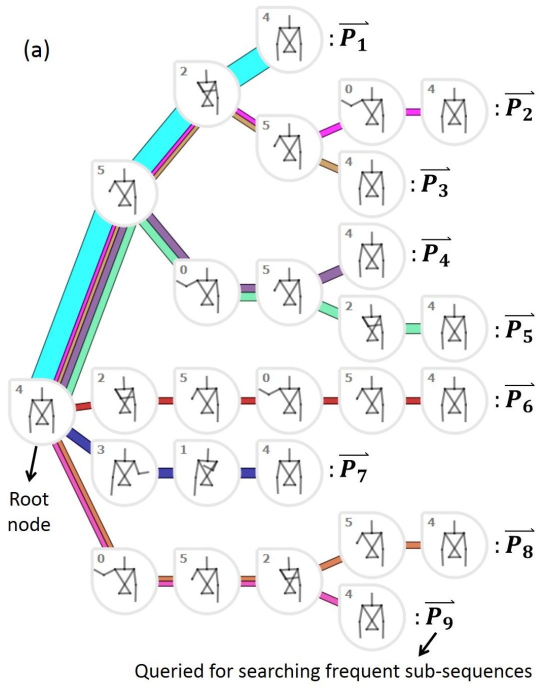
After users define pose states, the system simplifies and aggregates the motion data into a pattern visualization. Here, we provide an overview of multiple motion data clearly showing common pathways of sequences (R1) and support exploratory analysis of motion patterns (R2, R3). In Section 2.1, we discussed limitations of existing techniques for visualizing motion patterns using a graph-based approach [5] and small-multiples [18]. Considering both analytical requirements and the limitation of existing approaches, we decided to use a tree layout design. The main advantage of the tree-based approach is to allow a complete view of motion including clear indication of start, end, and intermediate motion trends. However, trees have a potential scalability issue when considering the visualization of large motion patterns. To alleviate this issue, we provide interactive sub-sequence searching; and navigation of tree structure and a separate view dedicated to show the selected sub-tree structure (Section 4.3.5). Thus, inspired by the Word Tree [40] visualization technique, we designed the *Pose Tree*, a visual motion pattern concordance (Figure 5). In the Pose Tree, nodes represent the pose states and edges encode transitions between them.

4.3.1 Abstraction and Aggregation

The first step in generating the pose tree is to simplify each motion data as a sequence of representative pose states. The pose states being used in this step are defined through the clustering process described in Section 4.2. Based on this visual abstraction, we consider the problem of visualizing sequential motion data as state transition of the pose states. The next step is to aggregate the motion sequences into a set of motion patterns. Our aggregation procedure is equivalent to creating a *prefix tree* [13] where each child node has only one prefix, a series of parent nodes. The aggregation starts with finding the pose states from which each motion sequence starts and then generating a list of root nodes where pose trees start to branch off. Then, it recursively searches the motion sequences, and adds non-existing child nodes to a corresponding prefix. The result is the pose tree structure where each leaf node stands for the last pose state of a unique motion pattern. Figure 5(a) shows an example of the pose tree visualization. Here, pose states are given by $\vec{C} = [c_0, c_1, c_2, c_3, c_4, c_5]$ where six distinct representative poses are defined. There is a single root node representing c_4 state (a natural standing pose). By expanding a pose tree from the root node, we get nine motion patterns ($\vec{P}_1 \sim \vec{P}_9$).

4.3.2 Tree Layout

In the pose tree layout, motion patterns are ordered from left to right representing temporal occurrence. To draw the pose tree, we first define the position of a leaf node in each pattern. We then vertically and horizontally partition the pose tree into m and n layers, respectively, where m is equal to the highest order of state transitions and n is the number of motion patterns. In the i-th vertical layer, leaf nodes having i-th order of state transitions are horizontally positioned. In Figure 5(a), the leaf node of the motion pattern $\vec{P}_4$ is a 4th-order state transition, so it is horizontally positioned at the 4th vertical layer. Similarly, leaf nodes are vertically positioned based on the order of their motion patterns. The motion patterns are then sorted based on frequency. For example, $\vec{P}_1$ is the most frequent pattern, so it is vertically positioned on the first horizontal layer. Once all leaf node positions are defined, the positions of all remaining nodes are recursively decided based on the position of their child nodes.



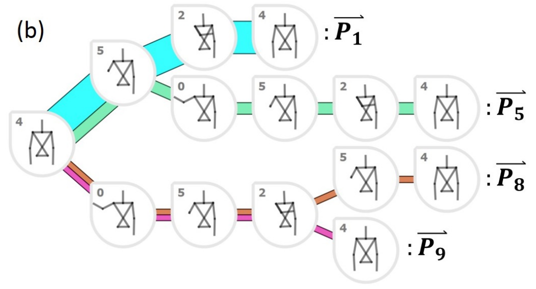


Fig. 5. Pose Tree visualizations. (a) Visualization of sequential motion patterns. Here, 9 unique patterns are identified, and colors are assigned to each pattern. The link thickness encodes the frequency of transitions between two poses. (b) Similar sub-sequence search result.

4.3.3 Edge: Flow Visualization of Pose Transitions

To provide an overview of the frequency of motion patterns, we take a flow visualization approach in visual encoding the pose tree edges. The edges connect the tree nodes, and each connection involves a certain number of motion data which is equivalent to the frequency of state transitions. The width of edges is proportional to the number of associated motion data over the number of the entire motion data. For example, in Figure 5(b), motion pattern $\vec{P}_1$ and $\vec{P}_5$ consists of six and two motion data respectively. So, the width of edges in $\vec{P}_1$ is three times thicker than the edges in $\vec{P}_5$. To distinguish the patterns identified in the pose tree, we assign different colors to each pattern.

4.3.4 Alternative View: Motion Pattern Treemap

To support exploration of individual or groups of motions (R2), MotionFlow employs a *treemap* as an alternative representation of the pose tree using an ordered space-filling layout [32]. This is to preserve the order and hierarchical relationship in the pose tree structure using a treemap layout. In this view, individual rectangles represent a single motion data, and the vertical and horizontal adjacency of rectangles encode hierarchical relationship among motion data. Figure 6 shows a treemap layout of pose tree in Figure 5(a). The root node is split into four child nodes, so the original rectangle in the tree map is first horizontally divided into four motion pattern groups; $G_1 = [\vec{P}_1, \vec{P}_2, \vec{P}_3, \vec{P}_4, \vec{P}_5]$, $G_2 = \vec{P}_6$, $G_3 = \vec{P}_7$, and $G_4 = [\vec{P}_8, \vec{P}_9]$. In the next step, G_1 is further vertically divided into two pattern groups.

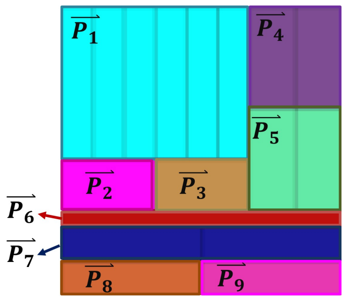


Fig. 6. A treemap layout for an alternative representation of pose tree in Figure 5(a). Each rectangle has the same color with corresponding motion pattern in the pose tree.

$G_{11} = [\vec{P}_1, \vec{P}_2, \vec{P}_3]$ and $G_{12} = [\vec{P}_4, \vec{P}_5]$. Similarly, the rest of space are divided into sub-rectangles inside of the tree map.

Animating Motion Data. Selecting individual or multiple items animates human motion in the animation view (Figure 1(f)). Animating complex human body motion may yield clutter when representing the body limbs as 3D lines. To address this issue, we used categorical qualitative color scales informed by ColorBrewer [16], and applied different colors to the body limbs to help users discern individual limbs and joints during the animation.

4.3.5 Searching for Similar Motion Patterns

The pose tree visualization can significantly reduce the visual space and information overload by aggregating coincident motion sequences into a motion pattern set. However, if two perceptually similar motion data have different pose state transitions at the initial motion, they would not be aggregated into the same motion pattern even if they share the same intermediate motion sequences. This is one of the main concerns in using a tree layout to provide an efficient overview of motion patterns. Based on this observation, we implement a sequential pattern mining technique to support pattern selection and exploration (R1, R2). In particular, we implemented PreFixSpan [24] with constraints on having non-intervals between pose states. In this approach, individual motion sequence is defined by a string of pose states, and frequent sub-sequences are generated based on their frequency and transition length [24]. We first generate a bag of frequent sub-sequential patterns (FSPs) from the motion sequences. When the users query a specific motion pattern, a list of associated FSPs is extracted from the bag of FSPs. Then, we search the extracted FSPs to identify sub-sequences matching to the queried motion pattern. As a result, a list of candidate similar motion sequences are generated. In our application, we restrict the minimum length of FSP as three successive pose states regarding the length of motion data.

Clicking a tree node selects all pathways passing through the node, and right-clicking the node toggles a pop-up menu, enabling similar pattern searching. In Figure 5(a), a motion pattern $\vec{P}_9$ is queried to the original pose tree. The queried motion pattern is defined by five pose states $[c_4, c_0, c_5, c_2, c_4]$. As a result of the search, three similar motion patterns are retrieved by mutual sub-sequences; $[c_5, c_2, c_4]$ of $\vec{P}_1$ and $\vec{P}_5$, $[c_4, c_0, c_5, c_2]$ of $\vec{P}_8$. MotionFlow provides a window dedicated to show the search result, represented in a sub pose tree structure (Figure 1-b). Here, the users are able to group similar sequential motion data distributed in different location of the pose tree. To support detailed and focused display of specific region of interest in the pose tree, we also provide zooming and panning capabilities.

4.4 Exploration and Organization of Motion Patterns

After motion sequences are aggregated into a pose tree, users investigate this overview for detailed exploration. To support such analysis, MotionFlow allows users to dynamically create and modify a group of motion data using tabbed windows (R4).

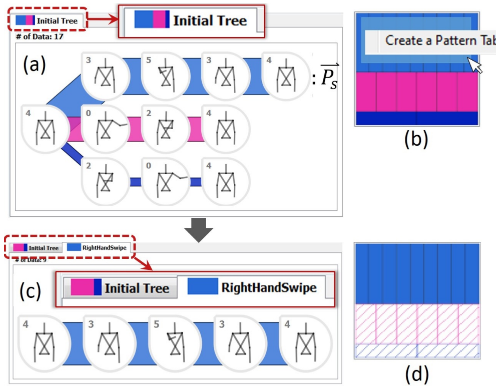


Fig. 7. (a) A motion pattern $\vec{P_s}$ with the thickest edge in the initial pose tree is considered as a potential frequent pattern. (b) The data included in $\vec{P_s}$ is selected on the treemap to create a new pattern tab and window. (c) *RightSwiping* tab only containing $\vec{P_s}$ is created. (d) The data not associated with the pattern $\vec{P_s}$ is unhighlighted in the treemap.

4.4.1 Motion Pattern Tabs

In the process of organizing motion patterns, users can progressively create and modify pattern tabs by adding or removing motion sequences. Using this dynamic and iterative refinement of pattern tabs, users can categorize patterns based on criteria such as their perception, and motion pattern context. Figure 7 illustrates how users interact with the pattern tab interface. Users select a motion pattern ($\vec{P_s}$ as the most frequent) from the initial pose tree (a), then create a pattern tab for storing the selected sub-pattern. (b) is a treemap view of the initial tree in (a). Users can select motion data included in $\vec{P_s}$ on the treemap (individuals) or the pose tree (groups). Right clicking on the selected data on treemap activates a pop-up context menu including a list of possible operations (create, add, and remove). (c) shows a new tabbed window storing the selected pattern $\vec{P_s}$. Pattern names can be customized so that the user can remember what type of motion pattern is aggregated in a certain tab. (a) and (c) show zoomed-in views of the pattern tab names (red box). A colored box icon is placed on the left side of each tab name in order to link the sequential patterns contained in each tab to the pose tree. These boxes can be dynamically changed as the users manipulate the data organization. The size of subordinate color boxes is proportional to the frequency of the corresponding motion pattern. For example, the Initial Tree tab in (a) consists of three sub-patterns; light blue, pink, and dark blue (from left to right order of appearance in the color-box). As we can see from the color-box icon, the dark blue motion pattern (left) has the largest portion in the pattern tab, while the other sub-patterns have smaller contributions in creating the pattern. As seen in (d), the treemap view reflects current selection of tabbed windows by representing non-associated data using hatching.

4.4.2 Specifying Motion Pattern Models

After creating the pattern tab containing interesting patterns, the users can further explore and organize them into a meaningful set of sub-patterns. Figure 8 illustrates a workflow of specifying pattern models of the subset of motion data ($\vec{P_s}$ in Figure 7). When users explore a subset of motion patterns in a tabbed window, only associated pose states are highlighted, while the others (green boxes in Figure 8(a)) are grayed out in the motion graph. Users can identify a cluttered pose state (red box), and split the pose state in the motion graph (blue boxes in Figure 8(a)). In return, the pose tree and the tree map change their structure showing hidden sub-patterns as seen in Figure 8(b). Only motion data included in the pattern tab are highlighted in the treemap view (Figure 8(b),(c)). The selective highlighting in the motion graph and the tree map supports the users in focusing on relevant information in analyzing and refining a specific group of motion data. By investigating the sub-patterns, the user can further specify pattern models, and organize the data based on the models as shown in Figure 8(c).

4.4.3 Creating an Organized Motion Database

After the analysis phase, users are able to save the gesture pattern organization to form a gesture database for later presenting and sharing insights with other researchers. Also, the visual representation of each gesture pattern (i.e., pose tree, pose state graph) can be further exploited to generate a statistical model informed by natural human gestures (R5). Users can project their domain knowledge and the context of human gestures directly into this output gesture database.

5 Evaluation: Expert Review

We performed an expert review [33] to evaluate the usability of MotionFlow in supporting pattern analysis of human motion tracking data. The goals of our study were to (1) evaluate how domain experts use MotionFlow in practical analysis tasks, and (2) understand the role of visual components in supporting the sensemaking process.

5.1 Design

We recruited six Ph.D. students, three of which had extensive experience in the design of gestural interactions, denoted as E1, E2, and E3. The other three participants (E4, E5, E6) indicated that they had expertise in pattern analysis of motion tracking data. All six participants had no prior experience with MotionFlow. The study consisted of (1) an introduction to MotionFlow, (2) four open-ended tasks lasting one hour in total, and (3) a post-questionnaire with an informal interview. The introduction phase lasted about 25 minutes, including a 10-minute demonstration by a proctor and 15 minutes of self-exploration. In each task, the participants were instructed to think aloud and, once finished, were given a post-questionnaire. We also recorded the participants' screens and comments for each session. Once all tasks were completed, the participants completed a final post-questionnaire including an informal interview about their experience with MotionFlow.

5.1.1 Tasks and Questionnaires

The tasks given in our study were (T1) generating a set of representative pose states as per the perception of the participants; identifying (T2) the most common and (T3) the unique motion pattern; and (T4) organizing unlabeled motion data into a meaningful set of motion patterns. The last task was based on free exploration, and intended to evaluate system usability and study expert strategies in solving a practical analytics task. Since there is no ground truth answer for each task, we did not quantitatively measure the analysis results users generated. We were more interested in studying how the experts connect each visual component of the system. We also designed tasks to expose participants to all components of MotionFlow in order to better understand the sensemaking process supported by visual components.

At the end of each task, the experts were given a set of in-study questionnaire (Q2–Q6: T1, Q7–Q10: T2 & T3, Q11–Q13: T4, Q1: end of all tasks), and it consists of thirteen 7-point Likert scale (1 = strongly agree, 7 = strongly disagree) questions:

Q1 I was able to learn how to use MotionFlow;
Q2 I was able to categorize poses as per my perception by manipulating the structure of the clusters;
Q3 I was able to recognize similar pose clusters from the data;
Q4 I was able to understand variations in pose clusters from the data;
Q5 I was able to identify frequent transitions among pose clusters;
Q6 I was able to able to generate a pose tree following guidelines;
Q7 I was able to identify the trends in motions from the pose tree;
Q8 I was able to compare difference between motion sequences;
Q9 I was able to identify frequent motion patterns;
Q10 I was able to identify distinct/unique motion patterns;
Q11 I was able to categorize motion sequences as per my perception by creating the motion pattern tabs;
Q12 I feel confident about my pose organization; and
Q13 I feel confident about replicating my pose organization.

We also asked participants seven open-ended questions on their procedure in completing each task in order to elicit feedback regarding the visualizations and interactions provided by MotionFlow.

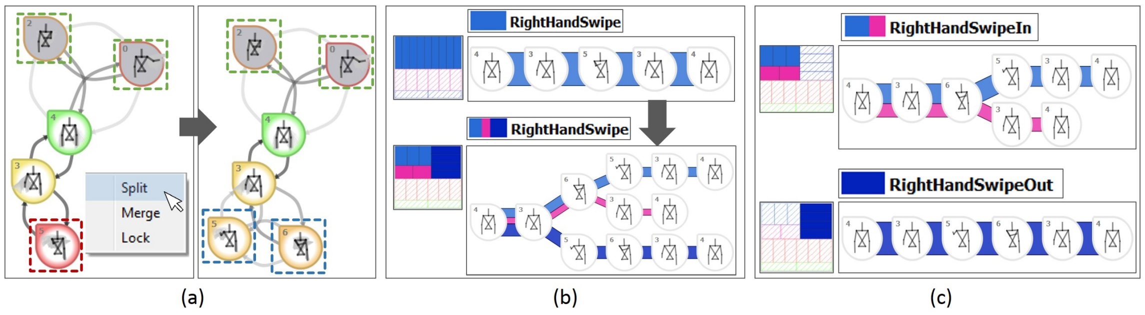


Fig. 8. Workflow of exploring and organizing motion patterns. After selecting an interesting sub-pattern in Figure 7, users can further analyze it for specifying pattern models. (a) By manipulating pose state clustering in the motion graph, (b) the structure of motion pattern is transformed showing hidden sub-patterns. (c) Through multi-tab interactions, the users can further organize the motion pattern into two sub-pattern groups.

Table 1. Collection of motion data used for the analysis tasks. The data is obtained from gesture elicitation studies on mid-air hand gestures [18].

Gesture Dataset	Gesture Styles
Changing volume of music player (T1–3)	(1) moving both hands down, (2) moving both hands up, (3) clapping hands, (4) moving right hand high-up and left hand mid-up, (5) moving right hand down, (6) swiping right hand.
Starting/stopping music player (T4)	(1) pulling one hand close, (2) pushing one hand away, (3) drawing a triangle, (4) drawing a rectangle, (5) moving right hand up, (6) swiping right hand, (7) moving both hands up, (8) moving both hand away, (9) crossing two hands, (10) moving both hands down, (11) both hands swiping, (12) touching shoulders.

5.1.2 Dataset

In Tasks 1–3, we used 34 clips of motion data with a total of 3,080 human poses and an average of 91 frames (min:66, max:133). In Task 4, we used 68 clips of data with a total of 5,657 poses and an average of 83 frames (min:62, max:128). Using MotionFlow, we organized the dataset into a set of gesture styles (Table 1). Our categorization was based on personal experience, thus it does not imply a ground-truth for gesture data aggregation. We provide such aggregation to show the complexity of tasks in terms of gesture style variations in the dataset.

5.2 Results

Figure 9 summarizes that experts indicated that MotionFlow is easy to learn, and effectively supported their pattern analysis tasks. Here, we summarize our findings from the expert review.

5.2.1 Analysis Process and Strategy

T1: Generating Representative Pose States. Most experts first tried to find the optimal number of partitions, so that the pose tree has a reasonable number of states and transitions. Then, they manipulated pose clusters while inspecting the structure of the pose tree. One expert (E5) tried to iteratively modify the clusters without linking the motion graph and the pose tree views.

T2 & 3: Identifying Common/Unique Patterns. All experts first explored the pose tree to identify and select candidate patterns. Then, they formed a hypothesis that the current selection is reasonable to their perception and interaction context. For confirmation, four experts (E1, E2, E4, E6) manipulated pose states in the motion graph and then observed whether the manipulation results affect the structure of the selected sub-tree patterns. If changes were not noticed, they ran the gesture animation to check for irregular motion. The other experts (E3, E5) inclined to directly run the gesture animation to observe detailed motion rather than using the visual overview.

T4: Organizing Motions into Patterns. Task 4 was comprehensive of previous tasks, so the experts adopted their strategies for Task 2 and 3. Additional interactions included creating and modifying tabbed windows for storing and organizing identified gesture patterns.

5.2.2 Efficiency

The experts were impressed that they could explore, understand, and organize an unstructured motion database in less than half an hour. E1 commented, *"Saving time. No need for tagging the video. The frequency of sequences can be easily determined."* E6 also mentioned, *"Video annotation is usually done by the highly paid researchers, [...] but simply going over the whole dataset. Using this tool, I think we can easily find out different set of unique gestures."*

5.2.3 Effectiveness

Interactive Pose Clustering. The responses for Q2–Q6 indicate that participants agreed that MotionFlow is effective in generating representative human poses based on their perception. All experts preferred local manipulations (split/merge) than global control of cluster numbers (adjusting slider-bar) in the clustering process. E6 particularly appreciated the split/merge manipulation: *"Split/Merge is especially useful when I want to find the uniqueness of the gesture."* The expert also appreciated the adjusting the view of poselets. Also, E2 mentioned that *"rotating view of poses inside of poselet is extremely helpful in understanding the context of poses in 3D space."*

Pose Tree with Flow Visualization. As shown in the rating results for Q7–Q10, experts were able to understand complete motion trends (i.e., start and end of motion sequence) and transition frequency from the flow visualization in the pose tree. All experts mentioned that the flow visualization in the pose tree gives a clear understanding of frequent and infrequent motion patterns. E6 mentioned that flow visualization required only low cognitive load when identifying aggregation of similar gesture patterns, and it does not require iteratively playing gesture animation. He could intuitively recognize which gestures to focus on for detailed analysis. Similarly, E1 said, *"the pose tree gives me full access into specific intermediary poses, that would not be available in typical video-based observational data."* The experts echoed that a complete view of motion sequences cannot be observed in the motion graph view, where only the connectivity between two successive pose states is accessible. Also they agreed that the pose tree is *"good for comparing multiple sequences"*.

Organizing Motion Patterns. Responses on questions Q11–Q12 indicate that all experts, except for E5, were able to confidently create a meaningful set of patterns. E1 mentioned that organizing patterns by grouping them into the tabbed windows helped avoid gesture clustering errors, which are common to automated methods. E4 commented, *"[...] with human perception, we can overcome the errors which come from computational methods."* E1, E2, and E4 all noted that the search

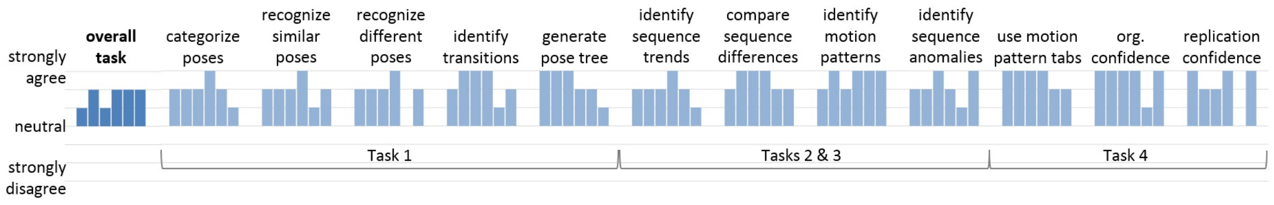


Fig. 9. Results from 7-point Likert scale questionnaire. Individual column groups correspond to an in-study question. The task description is provided above each group. Within a group, each of the 6 responses represents each of the 6 experts, ordered E1 through E6, from left to right.

and retrieval of similar patterns by querying interesting pattern is helpful in generating a set of motion patterns distributed in the pose tree.

5.2.4 Applicability

E4 and E5 noted that MotionFlow can be useful in organizing their database. E4 noted that *"in my case, hand motion data should be captured and clustered based on its similarity. However, there exists no such tool to do this kind of work. ... [I] hope this system is applied to my project to generate optimal pose clusters from huge amount of data soon."* Similarly, E5 commented that MotionFlow is *"valuable for separating noisy poses."* E1 mentioned that MotionFlow could be useful for improving gesture recognition: *"[The] pose tree [could] help us identify frequent poses that can cause misclassification of a gesture. This information can be used to subsequently improve the system for identifying erroneous or ambiguous gesture cues."*

6 Discussion

Our expert review revealed that MotionFlow is effective in analyzing and aggregating motion sequences into a set of patterns. Specifically, our pose tree and motion graph visualizations were effective in specifying the underlying structure and relationship of the motion dataset while leveraging human perception and gesture context.

The experts used the interactive pose clustering not only for generating distinct pose states, but also for understanding the context of pose states by exploring the transition patterns in the pose tree. One expert noted that *"the combination of motion graph and pose tree was very helpful during the initial organization of the data into reasonable chunks to start with."* However, one participant (E5) lacked such insight and found it difficult to organize the gesture database. He could not easily relate the pose tree and motion graph in understanding the role of pose states in the pose tree. Rather, he tried to obtain a well-organized pose tree by only manipulating pose clusters in the motion graph. In the informal interview, he reported that *"I fixated on defining pose states, and could not relate the visual components of MotionFlow."* From observation, we noticed that (a) relating the pose tree and motion graph is critical for understanding and organizing the database, and (b) the sensemaking process is supported well by guiding experts to understand the linking between the two visual components.

In Tasks 2–4, the experts used the animation view to validate their hypothesis that the selected candidate motion pattern is coincident with their background knowledge. They explored motion patterns by looking at details, such as thick edges in the pose tree and the large screen area in the treemap. To refine the flow visualization, they manipulated the pose states in the motion graph. However, when a group of motions being explored consisted of a smaller number of instances, they preferred the animation view. This implies that the motion tracking data itself is a crucial component for our understanding of human motion as it directly represents parts of the human body. However, when analyzing large amounts of motion data, such organization with a low level data representation is difficult. Our interactive visualizations provide us visual summaries that enable us to efficiently reach an acceptable understanding of the data. A key insight is that the animation cannot be replaced by pose tree or motion graph alone, but all components should be integrated to support the sensemaking process.

Limitations and Future Work. Even if our initial results are promising in analyzing and organizing motion patterns, there are limitations in MotionFlow's current form. While gesture elicitation studies normally start from a common neutral pose, some applications may start from different poses, leading to many root nodes branching out to multiple tree diagrams. This will limit the pose tree in providing a concise overview of patterns having multiple starting poses. In such cases, MotionFlow provides a global overview as a forest of trees. Then users can explore motion patterns through searching and navigating the multi-tree structure, and progressively organize the data into pattern tabs. Another potential solution is to develop a visual aggregation method that combines intermediate motion sequences into a single structure (e.g., single tree and graph) while preserving sequential completeness and conciseness. However, this is a very challenging problem, involving a NP-complete problem of subgraph isomorphism [35].

In broader contexts, general motion data (i.e., beyond the gesture elicitation studies that MotionFlow was designed for) can involve hours of activity recording rather than single gesture trials. Our current approach does not support the analysis of long motion sequences. Although analyzing long motion sequences is not the main concern in our application scenario, this could be an interesting direction to investigate in the future. Even despite scale issues, human motion is intrinsically compositional, i.e., longer motions are composed of sequences of shorter motions. This means that we can even use MotionFlow's shorter sequences to manage longer periods of motion data. For example, for motion data consisting of multiple human actions such as walking, crawling, or drinking, we may use the individual action as states, and represent the motion data as a transition between them. Then, we can apply our visual encoding approach to simplify and aggregate motion patterns using the action state transitions.

The scalability issue can also occur in the color coding applied to the pose tree, treemap, and multi-tab windows. When presenting a large number of sequential patterns, the color boxes can be cluttered and hard to discern. Specifically, the colored box in the tabbed windows was not actively used by the experts compared with other views. As one expert participant suggested, we will explore this visual space to provide a more effective overview of the categorization in the future.

7 Conclusion

We presented MotionFlow, a visual analytics system that supports pattern analysis of human motion targeting applications in gesture pattern studies in HCI. MotionFlow provides interactive clustering with local and global manipulations that enables users to generate representative pose states based on their perception. The pose tree with flow visualizations provides an overview of complete motion trends. MotionFlow also provides a motion pattern tab interface allowing users to explore and organize motion data into a set of meaningful patterns. Results from an *expert review* showed that MotionFlow is easy to learn, and effectively supports experts in performing gesture pattern analysis.

Acknowledgments

This work was partially supported by the NSF Award No.1235232 from CMMI and 1329979 from CPS, as well as the Donald W. Feddersen Chaired Professorship from Purdue School of Mechanical Engineering. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors.

References

[1] Z. Ahmed, P. Yost, A. McGovern, and C. Weaver. Steerable clustering for visual analysis of ecosystems. In *Proceedings of the EuroVis Workshop on Visual Analytics*, 2011.

[2] D. S. Alexiadis, P. Kelly, P. Daras, N. E. O'Connor, T. Boubekeur, and M. B. Moussa. Evaluating a dancer's performance using kinect-based skeleton tracking. In *Proceedings of the ACM Conference on Multimedia*, pages 659–662, 2011.

[3] M.-F. Balcan and A. Blum. Clustering with interactive feedback. In *Algorithmic Learning Theory*, pages 316–328. Springer, 2008.

[4] P. Berkhin. A survey of clustering data mining techniques. In *Grouping multidimensional data*, pages 25–71. Springer, 2006.

[5] J. Bernard, N. Wilhelm, B. Kruger, T. May, T. Schreck, and J. Kohlhammer. MotionExplorer: Exploratory search in human motion capture data based on hierarchical aggregation. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2257–2266, 2013.

[6] J. Blaas, C. P. Botha, E. Grundy, M. Jones, R. S. Laramee, and F. H. Post. Smooth graphs for visual exploration of higher-order state transitions. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):969–976, 2009.

[7] P. Blikstein. Unraveling students' interaction around a tangible interface using gesture recognition. In *Educational Data Mining 2014*, 2014.

[8] T. L. Booth. *Sequential machines and automata theory*, volume 3. Wiley New York, 1967.

[9] L. Bourdev and J. Malik. Poselets: Body part detectors trained using 3D human pose annotations. In *Proceedings of the IEEE Conference on Computer Vision*, pages 1365–1372, 2009.

[10] M. Burch, F. Beck, and S. Diehl. Timeline trees: visualizing sequences of transactions in information hierarchies. In *Proceedings of the ACM Conference on Advanced Visual Interfaces*, pages 75–82, 2008.

[11] W. Cui, S. Liu, L. Tan, C. Shi, Y. Song, Z. Gao, H. Qu, and X. Tong. TextFlow: Towards better understanding of evolving topics in text. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2412–2421, 2011.

[12] I. Davidson, S. Ravi, and M. Ester. Efficient incremental constrained clustering. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining*, pages 240–249, 2007.

[13] E. Fredkin. Trie memory. *Communications of the ACM*, 3(9):490–499, 1960.

[14] D. Gotz and H. Stavropoulos. DecisionFlow: Visual analytics for high-dimensional temporal event sequence data. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):1783–1792, 2014.

[15] D. Guo, D. Peuquet, and M. Gahegan. Opening the black box: interactive hierarchical clustering for multivariate spatial patterns. In *Proceedings of the ACM Symposium on Advances in Geographic Information Systems*, pages 131–136, 2002.

[16] M. Harrower and C. A. Brewer. Colorbrewer. org: an online tool for selecting colour schemes for maps. *The Cartographic Journal*, 40(1):27–37, 2003.

[17] M. S. Hossain, P. K. R. Ojili, C. Grimm, R. Muller, L. T. Watson, and N. Ramakrishnan. Scatter/gather clustering: Flexibly incorporating user feedback to steer clustering results. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2829–2838, 2012.

[18] S. Jang, N. Elmqvist, and K. Ramani. GestureAnalyzer: visual analytics for pattern analysis of mid-air hand gestures. In *Proceedings of the ACM Symposium on Spatial User Interaction*, pages 30–39, 2014.

[19] T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Information Processing Letters*, 31(1):7–15, 1989.

[20] L. Kovar, M. Gleicher, and F. Pighin. Motion graphs. *ACM Transactions on Graphics*, 21(3):473–482, 2002.

[21] G. Kurtenbach and E. A. Hulteen. Gestures in human-computer communication. *The Art of Human-Computer Interface Design*, pages 309–317, 1990.

[22] S. Liu, Y. Wu, E. Wei, M. Liu, and Y. Liu. StoryFlow: Tracking the evolution of stories. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2436–2445, 2013.

[23] M. Monroe, R. Lan, H. Lee, C. Plaisant, and B. Shneiderman. Temporal event sequence simplification. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2227–2236, 2013.

[24] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M.-C. Hsu. PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *Proceedings of the IEEE Conference on Data Engineering*, pages 215–224, 2001.

[25] C. A. Pickering, K. J. Burnham, and M. J. Richardson. A research study of hand gesture recognition technologies and applications for human vehicle interaction. In *Proceedings of the IEEE Conference on Automotive Electronics*, pages 1–15, 2007.

[26] A. J. Pretorius and J. J. Van Wijk. Visual analysis of multivariate state transition graphs. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):685–692, 2006.

[27] M. Rasmussen and G. Karypis. gcluto: An interactive clustering, visualization, and analysis system. *UMN-CS TR-04*, 21, 2004.

[28] P. Riehmann, M. Hanfler, and B. Froehlich. Interactive Sankey diagrams. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 233–240, 2005.

[29] J. Ruiz, Y. Li, and E. Lank. User-defined motion gestures for mobile interaction. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 197–206, 2011.

[30] T. Schultz and G. L. Kindlmann. Open-box spectral clustering: applications to medical image analysis. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2100–2108, 2013.

[31] J. Seo and B. Shneiderman. Interactively exploring hierarchical clustering results [gene identification]. *IEEE Computer*, 35(7):80–86, 2002.

[32] B. Shneiderman. Tree visualization with tree-maps: 2-D space-filling approach. *ACM Transactions on Graphics*, 11(1):92–99, 1992.

[33] M. Tory and T. Möller. Evaluating visualizations: Do expert reviews work? *IEEE Computer Graphics and Applications*, 25(5):8–11, 2005.

[34] H. Y. Tsang, M. Tory, and C. Swindells. eSeeTrack—visualizing sequential fixation patterns. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):953–962, 2010.

[35] J. R. Ullmann. An algorithm for subgraph isomorphism. *Journal of the ACM*, 23(1):31–42, 1976.

[36] F. Van Ham, H. Van de Wetering, and J. J. van Wijk. Interactive visualization of state transition systems. *IEEE Transactions on Visualization and Computer Graphics*, 8(4):319–329, 2002.

[37] Vinayak, S. Murugappan, H. Liu, and K. Ramani. Shape-It-Up: Hand gesture based creative expression of 3D shapes using intelligent generalized cylinders. *Computer-Aided Design*, 45(2):277–287, 2013.

[38] K. Wagstaff, C. Cardie, S. Rogers, S. Schrödl, et al. Constrained k-means clustering with background knowledge. In *Proceedings of the International Conference on Machine Learning*, pages 577–584, 2001.

[39] X. Wang and I. Davidson. Flexible constrained spectral clustering. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining*, pages 563–572, 2010.

[40] M. Wattenberg and F. B. Viégas. The Word Tree, an interactive visual concordance. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1221–1228, 2008.

[41] A. Wexelblat. Research challenges in gesture: Open issues and unsolved problems. In *Gesture and Sign Language in Human-Computer Interaction*, volume 1371 of *Lecture Notes in Computer Science*, pages 1–11. Springer, 1998.

[42] J. O. Wobbrock, H. H. Aung, B. Rothrock, and B. A. Myers. Maximizing the guessability of symbolic input. In *Extended Abstracts of the ACM Conference on Human Factors in Computing Systems*, pages 1869–1872, 2005.

[43] K. Wongsuphasawat and D. Gotz. Exploring flow, factors, and outcomes of temporal event sequences with the Outflow visualization. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2659–2668, 2012.

[44] K. Wongsuphasawat, J. A. Guerra Gómez, C. Plaisant, T. D. Wang, M. Taieb-Maimon, and B. Shneiderman. LifeFlow: visualizing an overview of event sequences. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 1747–1756, 2011.

[45] Y. Wu, S. Liu, K. Yan, M. Liu, and F. Wu. OpinionFlow: Visual analysis of opinion diffusion on social media. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):1763–1772, 2014.

[46] H. Zhou and H. Hu. Human motion tracking for rehabilitation—a survey. *Biomedical Signal Processing and Control*, 3(1):1–18, 2008.