Motion Synthesis for Sports using Unobtrusive Lightweight Body-worn and Environment Sensing

P. Kelly[1], C. Ó Conaire[1], N.E. O'Connor[1] and J. Hodgins[2]

[1]CLARITY: Centre for Sensor Web Technologies, Dublin City University, Dublin, Ireland
[2]Disney Research, Pittsburgh, PA, USA

Abstract

The ability to accurately achieve performance capture of athlete motion during competitive play in near real-time promises to revolutionise not only broadcast sports graphics visualisation and commentary, but also potentially performance analysis, sports medicine, fantasy sports and wagering. In this paper, we present a highly portable, non-intrusive approach for synthesising human athlete motion in competitive game-play with lightweight instrumentation of both the athlete and field of play. Our data-driven puppetry technique relies on a pre-captured database of short segments of motion capture data to construct a motion graph augmented with interpolated motions and speed variations. An athlete's performed motion is synthesised by finding a related action sequence through the motion graph using a sparse set of measurements from the performance, acquired from both worn inertial and global location sensors. We demonstrate the efficacy of our approach in a challenging application scenario, with a high-performance tennis athlete wearing one or more lightweight body-worn accelerometers and a single overhead camera providing the athlete's global position and orientation data. However, the approach is flexible in both the number and variety of input sensor data used. The technique can also be adopted for searching a motion graph efficiently in linear time in alternative applications.

Categories and Subject Descriptors (according to ACM CCS): I.3.0 [Computer Graphics]: General—

1. Introduction

The adoption of 3D graphical visualisation techniques has become increasingly prevalent in sports broadcasting, allowing expert analysts to highlight specific aspects of key game events and increase viewer insight, education and engagement. However, to date, in-game visualisation of gameplay has generally been limited to non-human objects, such as illustrations of offside lines or ball trajectories, as accurate performance capture of athlete motion during competitive play is impractical with traditional motion capture techniques. Optical motion capture systems, such as Vicon [Vic12] or Codamotion [Cod12], are typically considered the *gold standard* in human performance capture and the recent emergence of the Kinect [Kin12] has provided markerless full-body gaming in home environments. However, capturing sporting motion with any optical system can be challenging, even impractical, within large spatial volumes due to the cost and difficulty in densely and safely populating a capture area with enough cameras to ensure adequate coverage. This is especially problematic for Kinect systems, as the IR patterns from multiple devices can interfere with each other and significantly decrease tracking accuracy. Although large area captures are possible with traditional motion capture systems during practice sessions with heavy athlete instrumentation, multiple optical markers are unlikely to be worn, or allowed, during professional competitive matchplay. Additional complications can also arise during outdoor data capture sessions where there is little control over lighting conditions or occlusions, when motion from high-velocity movements is required and tracking (of markers or limbs) can easily be lost, or when strict constraints on reconstruction time leave little scope for manual intervention to correct captured motion artefacts.

An increasing demand for high-quality human motion in recent years has driven the computer animation community to research and develop techniques that simplify, expedite and reduce the cost of acquiring motion data or that increase the range of feasible capture environments. A variety of re-

cent approaches are shown in Table 1 along with a list of properties that are important for different applications. Not surprisingly, no one approach is perfect for all applications.

Our approach is aimed at applications that require capture outside of the tightly controlled laboratory or capture stage and subjects who can tolerate only a lightweight, unobtrusive and hidden form of instrumentation. We aim to capture high quality animations of *both* the root position (with respect to a global environment origin) and subject pose position for subsequent visualisation and animation. We do not target on-line applications, but instead aim to optimise animation quality within a few seconds or minutes of the performance of the full sequence of motion, so that in live sports broadcasts for example, an athlete's performance can be quickly and accurately synthesised for both visualisation and discussion during the break between periods. The trade-off for obtaining high quality capture from lightweight subject and environment instrumentation is that our approach requires a database of short pre-captured segments of motion that cover the basic behaviours expected from the subject during the capture. As such, truly novel action that are significantly distinct from those incorporated within the database can not be accurately synthesised using our method.

The approach can be seen as a data-driven puppetry technique, where a subject's performed motion is synthesised by finding related action sequences in the database using a sparse set of measurements from the performance. At the core of our approach is a memory efficient technique for searching a motion graph in *linear time*, regardless of the set of animation requirements or constraints. We demonstrate the approach using constraining input data acquired from one or more 3D accelerometers and a single overhead camera for global position and orientation of the subject. However, *any* other set of constraints or requirements could be used; including additional inertial sensors, other sources of global position data (differential GPS, ultra-wide-band tag tracking, or even simple human sketch to define a character's required path through a scene) or any manually defined animator constraints (such as times at which the character is required to perform specific actions). As such, the technique can be adopted as an efficient technique for any application that requires to search large motion graphs in linear time [KSG02, LCR*02, AF02, SH07].

There are three primary contributions of this paper. (1) We illustrate how multiple input data modalities (such as inertial, global position or orientation data and other manually defined constraints) can be efficiently incorporated into a motion graph search in *linear time*. Unlike previous approaches, the inclusion of global position data into a graph search does not require a large scale and computationally complex unrolling of the motion database. Alternatively, a linear 30% increase in computational cost (as opposed to an exponential increase in unrolling cost with respect to sequence length) is required. (2) Unlike previous work in the area, if inertial sensor input data is *solely* used to synthesise motions, our approach is guaranteed to obtain an *optimal* path through a motion graph (by globally minimising the squared error between real-world sensors and the local motion of the body). Although incorporating global information as input data invalidates this contribution (as an optimal path through the motion graph in that situation is not guaranteed), the technique can consistently provide synthesised motions that significantly reduce global positional and orientation error. This contribution is important for applications that do not need or require global positional data to be considered in a motion graph search. (3) In order to reduce memory requirements, previous approaches tend to reduce the size or complexity of its underlying motion graph before searching the graph for appropriate paths. This is especially true if long sequences of synthesised motion are required, or if the search requires global position constraints to be enforced. As the length of required sequences increase, the amount of compression required also grows (or the search tends to be inefficiently split into N multiple searches). In this work, we illustrate an alternative approach where *memory reduction* techniques are focused on the search algorithm, and not on the motion graph. This technique can be used to successfully synthesise motion sequences of arbitrary lengths on a standard desktop computer, and without a need in the reduction in complexity or size of the underlying motion graph.

This paper is organised as follows: Section 2 details previous work in the area. Section 3 provides a high level overview of the approach, while Sections 4 and 5 describe each individual component. Section 6 provides quantitative evaluation of our approach in a real world test scenario; in particular we focus on synthesising the rapid motions from a nationally ranked tennis player in competitive action. Tennis is a challenging test scenario for our work, given the speed, movement and explosive nature of the actions performed by high performance tennis players. In addition to tennis, for qualitative evaluation we also synthesise boxing motions, and extend the approach to a variety of other non-sports specific actions incorporated within the publicly available HDM05 [MRC*07] dataset. Finally, Section 7 provides conclusions and directions for future work.

2. Prior Work

Motion capture is a well-studied and broad research area, so in this brief account of related work we limit ourselves to a review of data-driven approaches for motion animation and inexpensive, portable human motion capture systems. We also briefly discuss the use of motion graphs and search algorithms, as we build on this work in our approach. A technology review for motion capture is provided in [WF02].

Changes in lighting, coverage or occlusion problems can introduce robustness issues for most optical systems. An alternative, less intrusive method, for motion inference that can overcome these drawbacks is to embed sensors in a sub-

Table 1: *Comparison of motion capture techniques; (N)one, (S)mall, (M)edium, (H)igh, (P)remium.*

Approach	Vicon	Kinect	[SPS*11]	[VAV*07]	[TZK*11]	This Work
Subject instrumentation	H	N	H	H	S	S
Environment instrumentation	H	S	S	S	N	S
Size of capture environment	M	S	H	H	H	H
Robust to outdoor lighting	✗	✗	✓	✓	✓	✓
Robust to occlusions	✗	✗	✗	✓	✓	✓
Reconstruct fast movements	✓	✗	✗	✓	✓	✓
Free-form motions captured	✓	✓	✓	✓	✗	✗
Database required	✗	✓	✗	✗	✓	✓
Reconstruction accuracy	P	M	M	H	M	M
Reconstruct global root motion	✓	✓	✓	✗	✗	✓
Performs activity recognition	✗	✓	✗	✗	✗	✓
Manual post-processing	✓	✗	✗	✗	✗	✗
Online time complexity	H	S	H	M	S	M
System Cost	H	S	M	M	S	S

ject's clothing. When held in a static position, the pitch and roll of a tri-axial accelerometer sensor can be inferred. Using this technique and the assumption of low acceleration motions, body limb positions can be directly inferred from wearable accelerometer data streams [LH01, TL06]. Combining an accelerometer with a magnetometer and a gyroscope in a single *inertial measurement unit* (IMU) device, allows sensor yaw to be determined and can help increase robustness against high acceleration motions [LWC*11]. IMUs have been incorporated into commercial products [XSE12] however are still prone to *drift* in accuracy over time, especially if fast movements with high accelerations are performed. Incorporating further sensors, such as ultrasonic time-of-flight devices, on the body can reduce the negative impact of this drift [VAV*07]. Shiratori et al. [SPS*11] takes an interesting alternative approach, instrumenting an actor with multiple outward-looking cameras and acquiring pose and global position of the subject using image processing techniques. From discussions with elite athletes and sports professionals, we believe that our target application requires minimal subject instrumentation. We therefore focus on the use of accelerometers alone, as they can be sewn into clothing and sporting equipment in a very unobtrusive way [SH08].

The most similar approaches to the one described in this paper are data-driven, using sensor data to index into a motion database and animate motion segments that exhibit similar motion data to those acquired by the sensors [LH01, RSH*05, SH08, Kum08, TZK*11, HBL11]. Ren et al. [RSH*05] search for upper-body motions in the database that have similar features to those extracted from silhouettes extracted by image processing techniques, Slyper and Hodgins [SH08] and Kumar [Kum08] describe a similar system but use features obtained from inertial sensors placed on the subject's arms to search the database, while Tautges et al. [TZK*11] and Ha et al. [HBL11] extend this approach of using inertial data to full-body motion synthesis and for significantly larger motion databases. These approaches adopt sliding window-based search algorithms, determining the actions to perform using only a small temporal sequence of accelerometer data. This technique, coupled with small window sizes, ensures that the latency between user movement and motion synthesis is short, allowing these techniques to be used for near real-time applications. However, the use of small independent windowing techniques can negatively affect animation quality, as pose transitions tend to be only loosely constrained, which can result in poor, unnatural motion animations. In addition, none of these approaches can reliably recover the global position, orientation (compass heading) and movement of the character over time.

Motion graph data structures [AF02, LCR*02, Kum08] can be used to encode a pre-captured motion database and limit unnatural motion transitions. Using this data structure, synthesised motions can be acquired by animating a sequence of poses associated with a path of connected nodes through the directed motion graph. Prior work in this area has included contributions by Safonova and Hodgins [SH07] who describe a number of additional transition rules, based on foot contact states, to minimise unnatural motions and broaden the variability of actions incorporated within the database by interpolating motions in the graph. Ren et al. [RSH*05] also extend the graph structure to allow original database motions to be performed at a variety of speeds. In order to acquire animations, many approaches [KSG02, LCR*02, AF02, SH07] use the motion graphs in conjunction with high-level animator requirements, such as simple human sketches defining a character's required path through a scene, or predefined times at which a character is expected to perform a high level action (such as a jump, wave or other such general movement) [AFO03]. Off-line variants of these techniques can produce high-quality animations that strictly adhere to the navigational constraints outlined by the animator. However, for realistic motion synthesis of human actions, the cost and time requirements of providing fine grained human limb and torso motion input requirements can be prohibitive, leading to searches being under-constrained in terms of local motion accuracy.

Many previous works [LCR*02, RP04, SH07, GSKJ03] navigate characters through a scene by *unrolling* or embedding the whole, or a portion, of the motion graph into the environment. For large graphs, this approach can be extremely expensive, both in terms of the space required to store the unrolled graph, and the time required to search for plausible solutions. As such, some approaches only unroll subsets of the graph at a time, or attempt to reduce its complexity and size [GSKJ03]. Even so, using *optimal* path search techniques, such as A* search [SH07] where the complexity is

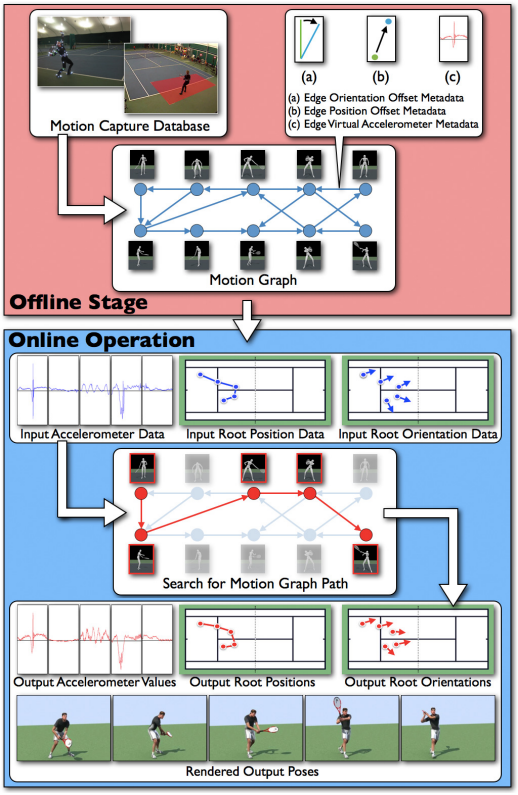


Figure 1: *Approach overview for a tennis test scenario with three input data modalities: root orientation data; root position information; and a single accelerometer stream.*

dependent on the heuristic used but can be polynomial with respect to time, can still be unrealistic. In this work, we use a variation of the Viterbi algorithm [For73] to search for motion graph paths that adhere to a weighted range of input data modalities. If the system is Markovian, Viterbi can *optimally* match the input readings to a suitable graph path with a complexity that is linear with respect to sequence length. In addition, if the memory reduction techniques of Abbasfar and Yoa [AY05] are incorporated, then very large data sequences and motion graphs can be searched in a single Viterbi pass. If the system is not Markovian (such as in our approach, when global root position and orientation data are incorporated into the search), the algorithm does not guarantee an *optimal* graph path solution, but the computational time and space requirements remain linear. To the authors' knowledge, no alternative search algorithm would guarantee an optimal solution in linear time.

3. Overview

Our approach has two stages, as outlined in Figure 1. In the **Offline Stage**, a motion graph is created that encodes a database of motions that we expect an actor to perform at capture time. These can be captured in short segments in a standard motion capture setup. The motion graph is then augmented to associate metadata such as virtual accelerometer traces with each graph edge. During the **Online Operation**, an actor's (i.e. athlete's) performed motion is synthesised by finding a sequences of connected nodes through the motion graph that have similar metadata values to those measured from the actor. The poses associated with each node on the determined graph path are then post-processed to acquire the final animated motion sequence. These two stages will now be discussed in detail.

4. Offline Stage

We use motion graph data structures to encode how clips from a captured motion database may be re-assembled to form new realistic motion sequences. We automatically create a motion graph using a point cloud driven approach, as described by Kovar [KGP02], using additional constraints that limit transitions to only occur during changes in foot contact state, and only between poses with the same contact state. Safonova and Hodgins [SH07] demonstrated that adopting these constraints ensures that motions acquired from the motion graph remain close to physically correct. The final stage for many approaches [KSG02, LCR*02] is to ensure that the created motion graph is strongly connected. This process ensures that a path exists between any two given nodes in the directed graph, thus making sure that there are no "dead ends" in the graph. We can skip this step because the Viterbi search automatically avoids "dead ends" unless they are a natural conclusion to the input data.

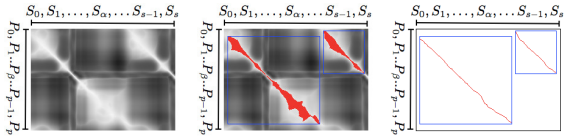
We extend the motion graph creation technique to incorporate two additional features. We modify the graph structure so that during performance capture, actions may be performed at speeds significantly different to those in the training data. Secondly, we *automatically* broaden the number of poses and increase graph connectivity by incorporating interpolated motion segments into the motion graph. Although these techniques increase both graph size and motion synthesis computation time, Section 6.1 illustrates how these techniques can increase motion synthesis accuracy.

4.1. Motion Speed Variations (MSV)

To allow variations in database motion playback speed, Kumar [Kum08] explicitly defines self-looping edges (i.e. starting and ending at the same node) within the motion graph. Traversing self-looping edges pauses the motion for a period of time, which can result in unnatural motion. We adopt an alternative approach, similar to Ren et al. [RSH*05], which allows more naturalistic variations in motion playback speed to be synthesised. Figure 2 illustrates the graph modification, where equidistant interpolated *slow-down nodes* are inserted between original nodes allowing the motion speed to be decreased by a factor of two, while

(a) (b)

Figure 2: *MSV; (a) Normal graph; (b) Additional slowdown nodes (grey dots) and speedup edges (dashed-lines).*



(a) (b) (c)

Figure 3: *Interpolation; (a) Similarity matrix; (b) Two groups of close similarity; and (c) Optimal paths through clustered groups.*

traversing a path containing *speedup edges* can double the speed of a performed action. Intermediary playback speeds between these two extremes can also be generated by alternating between normal and MSV edges. Poses from an acquired graph path are post-processed in the **Online Operation** stage to smooth out the resulting discontinuities in velocity. Further edges and nodes can also be added at alternative intervals if greater changes in motion speed are expected. It is important to note that additional edges should not be added when a subject has both feet off the ground, otherwise spurious motions could be generated where gravity will not be correctly enforced (its effect could be halved or doubled).

4.2. Interpolation

Given a limited amount of motion data, interpolation of existing clips can allow input constraints to be synthesised more accurately. In this work, we interpolate temporally similar database action segments and incorporate these new movements directly into the motion graph. Our approach is similar to that of Safonova and Hodgins [SH07], however we are more selective in the type of motions that we allow to be interpolated. Safonova and Hodgins create an interpolated sequence from two time-scaled graph paths that are compared for similarity using foot contact state patterns only. We extend this process, comparing the body positions, accelerations and movements of two segments before interpolation is allowed. Using this technique, only similar temporal *actions* are interpolated, while interpolation of dissimilar actions (e.g. a jumping-jack and a back-flip, for example) that can result in unnatural motions are more easily avoided, especially for sequences where little or no foot movement occurs. As with Safonova, our approach is automatic and does not require any human input or manual database labelling.

Given two input sequences from the motion database, S of length s poses and P of length p poses, a similarity matrix is created by directly comparing all poses S_α to P_β using the point cloud approach of Kovar [KSG02]. An illustrative visualisation of a similarity matrix is depicted in Figure 3(a), where the sequence poses of S and P are represented by the columns and rows of the matrix respectively, and the brightness of pixel (α,β) directly corresponds to the determined similarity between pose S_α and P_β. The similarity matrix is then filtered so that only pose pairs with the same foot contact states, and a point cloud difference below a threshold, t_i, remain – see the red pixels in Figure 3(b). Remaining neighbour pairs are then clustered into coherent groups using an iterative 8-neighbourhood connected components algorithm. Each group represents a continuous temporal sequence of motion frames (or actions) from S that can be mapped to a similar sequence in P. Two coherent groups, highlighted by blue bounding boxes, are obtained in Figure 3(b).

As a brute force algorithm for acquiring the *optimal* mapping between sequence pairs has exponential complexity, we apply a Dynamic Time Warping (DTW) [KR05] approach based on Dynamic Programming (DP), which has complexity of $O(\hat{s}\hat{p})$, where $\hat{s}$, $\hat{p}$ are the width and height of the clustered region. This algorithm can determine the optimum non-linear path through the clustered cells of the similarity matrix, so that each pose S_α is matched to a single pose P_β (and vice versa), using the point cloud similarity between poses as a cost function, and with the DTW alignment slope constrained between $1/2$ and 2 to prevent motion discontinuities. Once matched, two temporally aligned motion segments over 0.5 seconds in length are blended together using even weights to form new interpolated motions, using the approach described by Park and colleagues [PSKS04], and then added to the motion graph as previously described.

4.3. Metadata

Pre-calculated metadata is associated with each motion graph edge to allow a comparison of paths through the motion graph with the input data streams. As such, the required metadata depends on the input data to be provided. In the example scenario, three input data streams are expected at capture time (global position and orientation data, plus a single accelerometer stream – see the metadata features extracted in the top-right section of Figure 1), three metadata items are associated with each graph edge. The first metadata item describes the global orientation offset that is incurred when traversing the edge (i.e. moving from pose α to pose β). This offset is calculated as the local root compass heading offset (with respect to pose α) that occurs from unrolling node pose α to pose β (e.g. if the compass heading of α is 5^o and β is 7.5^o, then the metadata item is set to $+2.5^o$). Similarly, the global position offset metadata item is calculated as the local groundplane vector from unrolling α to β (e.g. if the position of α on the groundplane is $(5,2)$ and β is $(1,8)$, the offset vector $(-4,+6)$ is stored). The third meta-

data item in the example scenario models the inertial readings that are induced when moving between pose α and β. We model the induced inertial readings using tri-axial *virtual accelerometers* [SH08], with the virtual device matching the position and orientation of each real world sensor expected to be worn on the body during performance capture. If other input data streams are provided, global or inertial input metadata (such as jumping height, rotation from gyroscopes, etc) should be calculated and stored in a similar manner.

5. Online Operation

Given a series of T samples, where a sample corresponds to one set of synchronised readings from all input sensor readings (such as accelerometers, body root tracker, etc), the goal of the online stage is to find an appropriate path through the motion graph, such that the estimated motion is a good approximation of the actions performed. Once acquired, the database poses traversed on the graph path can be post-processed and animated. We use a variation of the Viterbi algorithm [For73] using the pre-calculated metadata in the traversed edges to find the motion graph path that best matches the measured input values in a least-squared sense. Viterbi's approach is a Dynamic Programming (DP) algorithm that finds the most likely sequence of unknown states (i.e. motion graph nodes) given a series of observations (i.e. T input data samples). More formally, the Viterbi algorithm may be viewed as a solution to the problem of maximum a posteriori probability (MAP) estimation of the state sequence of a finite-state discrete-time Markov process observed in memoryless noise [For73]. Algorithmically, it determines the optimal discrete-time sequence of states that minimises the sum of costs, $G(n,T)$:

$$G(n,T) = \sum_{t=1}^{T} C(n,t) \tag{1}$$

where t is a discrete-time period, n is the sequence node selected at time t and $C(n,t)$ is the local cost of selecting node n at time t. In our approach, the sequence of nodes in the Viterbi path is chosen so that a motion graph-edge exists between all nodes in the path. As such, we assign the Viterbi cost function to be

$$C(n,t) = \begin{cases} 1 & \text{if } t = 0 \\ L(e,t) & t > 0 \end{cases} \tag{2}$$

where $L(e,t)$, the local cost of selecting the edge, e, that transitions to node n at time t. This cost can be quantitatively obtained by directly comparing the input data samples at time t to the metadata associated with the edge e. A number of variants of $L(e,t)$, using differing input modalities, are described in Sections 5.2 and 5.4.

The Viterbi algorithm is an iterative process that constantly maintains N possible paths through the graph, where N is the number of nodes in the motion graph. At each time stage, the last node in every path is unique, i.e. for any given node n in the motion graph at time t, a path of length t ending in node n exists and is maintained by the algorithm. On each iteration, the length of each path is extended by a single node and a global minimum cost, $G(n,t)$, of reaching node n at time t is updated via

$$G(n,t) = arg \min_{e \in E_n} [L(e,t) + G(m,t-1)] \tag{3}$$

where $G(n,0) = 0$, E_n are the set of edges leading from node m to node n, and e an edge from that set. From Equation 3, it is clear that Viterbi assumes that temporal transitions from one state to another are Markovian, and therefore computing $G(n,T)$ requires only $2N$ memory elements, as only N states at time t and $t-1$ must be maintained. However, NT memory elements are required to store the DP lattice backtrace links, which stores the immediate predecessor to every N node at each time iteration.

At time T, the optimal path is determined to be the one from the N stored paths that has the lowest associated global cost, $G(n,T)$. Using the DP lattice, the temporal sequence of graph nodes from this Viterbi path can be acquired and animated. We use blending windows of size w at motion transitions in the graph path, where every pose within the blending window is made up of w weighted poses centred around the transition point. The blending of root position is achieved using Bézier curves, while the w weighted pose quaternions are blended using Bernstein polynomials as described by Park and colleagues [PSKS04]. In our experiments, w is typically set to 0.7 seconds. However, in order to avoid unnatural movement with respect to gravity, the root position is blended using $w = 0.1$ second when neither foot is in contact with the ground. Finally, footskate artefacts are removed using a technique similar to the inverse kinematics based approach proposed by Kovar [KSG02].

5.1. Complexity and Memory Reduction

The Viterbi algorithm (with a graph of N nodes, E edges and T input data samples) has a processing time of the order $O(ET)$ and a DP lattice memory footprint of order $O(NT)$ (Note: $E = NN$ if every node is connected to every other node in the graph). While the processing time has linear complexity with respect to the length of the input data, for long sequences and large graphs the memory requirements may be high. Fortunately, the NT lattice is highly redundant, and can be compressed during the online search to significantly reduce memory usage. The approach we use is similar to that of Abssafar and Yao [AY05], removing lattice points that can never be found in the optimal path. This reduction is achieved by periodically performing a backtrace from all nodes at time t and removing all unused DP lattice points between time $[0,t]$. Lattice nodes that are not included in any backtrace path are redundant and can be removed, as they will never form part of the optimal path.

5.2. Motion Synthesis using Wearable Sensor Data

An *optimal* Viterbi path through the motion graph can be acquired in the sense that the squared error between the entire sequence of measured real-world values and the virtual sensor values is guaranteed to be globally minimised. For wearable accelerometers, the local cost of transitioning between nodes using the graph edge e at time t is given by:

$$L(e,t) = C_A(e,t) = \frac{\sum_{s=1}^{S} ||E^s_{acc}(e) - M^s_{acc}(t)||^2}{S} \quad (4)$$

where S is the number of sensors worn by the subject, $E^s_{acc}(e)$ is the pre-computed virtual acceleration metadata assigned to edge e for sensor s (with the virtual device matching the position and orientation on the subject's body as the real world sensor), and $M^s_{acc}(t)$ is the acceleration for sensor s measured by the worn device at time t.

5.3. Incorporating Root Position and Orientation

The search technique of Section 5.2 allows the position and orientation of the actor to drift over time as the selected path through the motion graph contains slightly different amounts of position and orientation changes than that of the required motion sequence. In this section, we adapt the Viterbi algorithm to integrate additional global input data and significantly reduce this error. The algorithm is modified to store and maintain the global character positions and orientations for each of the N maintained paths through the graph. At $t = 0$, each of the N global positions, $P(n,t)$, and orientations, $O(n,t)$, are initialised to the first input data stream values. At time t, let v be the edge from the set, E_n, which leads from node m to node n and minimises Equation 3. The associated position, $P(n,t)$, for the path ending in pose n is subsequently updated via

$$P(n,t) = P_{pos}(v) + P(m,t-1) \quad (5)$$

where $P_{pos}(v)$ is the global position offset metadata associated with traversing edge v. $O(n,t)$ is similarly updated.

When calculating the local cost of transitioning a graph edge in Equation 3, we weight the costs from $C_A(e,t)$, plus position ($C_P(e,t)$) and orientation data ($C_O(e,t)$) using

$$L(e,t) = W_A C_A(e,t) + W_P C_P(e,t) + W_O C_O(e,t) \quad (6)$$

where W_A, W_P and W_O are weighting parameters. The position cost of transitioning to node n with edge e at time t is

$$C_P(e,t) = ||\hat{P}(n,t) - M_{pos}(t)|| \quad (7)$$

where $\hat{P}(n,t)$ is the global position of the character *if* it transitioned to node n at time t using edge e (as described in Equation 5). The local *orientation* cost, $C_O(e,t)$ is similarly calculated. To ensure that the magnitude of position, orientation and accelerometer data errors are roughly comparable perceptually, we use *metres*, *radians* and *g-force* as the measurement units in Equation 6.

This adaption significantly reduces the average global positional and orientation error of the synthesised motions. In addition, as the complexity of the solution remains linear at $O(ET)$, this adaptation can be achieved with a linear increase in computation cost, caused by the substitution of Equation 4 for Equation 6. However, it is important to note that this alteration also breaks the Markovian assumption inherent in the Viterbi algorithm, due to the position and orientation associated with each of the N stored paths. For a given time iteration t, both $P(n,t)$ and $O(n,t)$ are calculated using the first values provided by the input data streams plus the metadata offsets from each edge in the sequence of nodes in the path. As such, at time t (using Equations 3 and 6) future states depend not only upon the observed input data at point t and the present state at time $t-1$, but also on the full sequence of states that preceded it. As the adapted approach is non-Markovian, the optimal path through the motion graph with respect to the chosen cost function is *not* guaranteed.

5.4. Manually Defined Human Input Constraints

Additional human-defined constraints can also be incorporated into the search algorithm. Let the user input a sequence of T constraints, where $r(t)$ specifies the state requirement (such as required foot contact states or specific actions to be performed) at time t. When a single set of T constraints are provided, an additional set of N global costs are maintained

$$G_r(n,T) = \sum_{t=1}^{T} C_r(n,t) \quad (8)$$

where $C_r(n,t) = 0$ if the graph node n adheres to the required constraint $r(t)$ at time t, and $C_r(n,t) = 1$ otherwise. The optimal, animated Viterbi path, breaks the fewest number of constraints and, as such, will minimise this new cost. If two or more paths adhere equally to the set of user constraints, then the global cost of Equation 1 is used to select the optimal path. If γ different user constraint sets are provided, then γ additional global costs can be simultaneously maintained, with the optimal paths ranked using the costs of the most important constraints first.

6. Experimental Validation

In this section we quantitatively evaluate our approach in a real world test scenario by synthesising motions from a nationally ranked tennis player in competitive action. Tennis is a challenging test scenario for our work, given the speed, movement and explosive nature of the actions performed by high performance tennis players. In addition to the local motion of a player's limbs, the synthesised motion must also take into account the global movement of the athlete through the field of play. In order to acquire the motion database, we instrumented an indoor tennis court with a 12-camera Vicon optical motion capture system [Vic12] and captured 20 sequences from a 6.5×3.25 metre capture area that covered a number of variations of all the major tennis

Figure 5: *Synthesised serve motion using input data from six equally weighted accelerometers.*

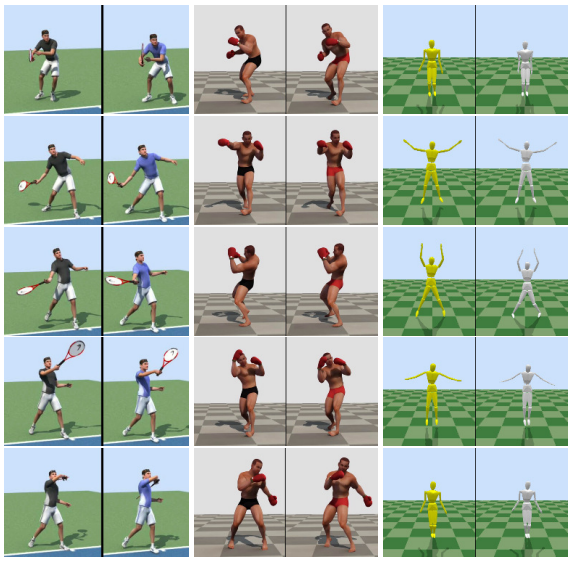


Figure 4: *Example synthesised (right figure) versus groundtruth motions (left); (Column 1) Tennis Forehand; (Column 2) Boxing Sequence; (Column 3) Jumping-jacks with HDM05 dataset.*

shots, enough data to allow the player to traverse the capture area efficiently plus a number of unscripted competitive game-plays. In total 15,304 frames at 30 Hz were captured (8.5 minutes of data). The camera placement and player instrumentation required to capture these isolated shots would not have been viable during competitive tournament play. For qualitative evaluation in a number of different scenarios, including tennis, boxing and motions from the publicly available HDM05 [MRC*07] dataset, readers are directed to Figures 4 and 5, and the supplementary videos associated with this paper.

In Section 6.2, a player's root position on the court is obtained using video processing from a single overhead camera [CKCO09], however any alternative modality, including human drawn sketches of temporal positions, could be used. The chosen source depends on the infrastructure available and accuracy required. For orientation, we simply make an assumption that the compass heading of the player is facing their direction of movement if the root position velocity is greater than a threshold (set empirically to 2 metres per second) otherwise we assume the player is facing the tennis net. Although simplistic, these assumptions provide good results.

6.1. Synthesis using Wearable Sensor Data

During the capture of the motion database, six ± 12 g triaxial wireless accelerometers (capable of transmitting synchronised data to a single base-station over 20 metres away) were placed on the body of the player (one on each forearm, each shin, one on the lower back and one on the chest). Using both the groundtruth Vicon and sensor data, we can quantitatively evaluate synthesis results via the approach of Section 5.2. This evaluation serves three purposes; firstly, we can investigate the benefits of incorporating Motion Speed Variations (MSVs) and interpolated motion sequences in a graph; secondly, we can compare the accuracy of *virtual sensors* and their real world counterparts; and thirdly, it provides a quantitative baseline for local-based motion reconstruction accuracy in subsequent experiments when both wearable and global information are provided as input data.

Table 2 provides quantitative evaluation of motion synthesis results from five different motion graphs. The first four graphs (named G_A to G_D respectively) were created from a 4.25 minute subset of the captured data, while the last graph, G_E, was created from all 8.5 minutes of data. It is important to note that although some, or all, of the test 20 tennis motion sequences are incorporated into the motion graphs used, when testing against a given groundtruth movement, the original Vicon motion is removed from the motion graph. As such, the synthesised motion can not come directly from the groundtruth-captured motion. Other specifics on the creation of each graph is provided by three columns in this table, which highlight whether a specific graph includes Motion Speed Variations (**MSV**) or Interpolation (**I**) sequences or not. In this work, interpolated sections were created using $t_i = 0.5$ metres, resulting in 17,567 potential new motion poses from 552 interpolated actions sequences with an average length of 1.1 seconds.

Using the smallest graph G_A, and synthesising motion from all 20 sequences (removing the groundtruth from the motion graph before applying our approach), we obtain an average joint position error of 12.7 cm plus a standard deviation of this error of 7.3 cm. By incorporating MSV or interpolated nodes, improvements in this error can be obtained. The error metric, as used by Tautges [TZK*11], is

Table 2: *Synthesis using 6 equally-weighted wearable accelerometer input data streams.*

Graph	MSV	I	Num. of Nodes	Num. of Edges	Average Joint Error (St.Dev.) In cm	Actions				Computational Increase
						TP	FP	TN	FN	
G_A	✗	✗	7,577	9,433	12.7 (7.3)	98	0	4	9	–
G_B	✓	✗	13,870	28,312	12.2 (7.0)	99	0	3	9	×2.00
G_C	✗	✓	25,054	95,318	11.9 (6.8)	104	0	3	4	×9.10
G_D	✓	✓	46,409	159,127	11.1 (6.1)	105	0	3	3	×15.87
G_E	✓	✓	100,686	317,600	10.8 (6.0)	105	0	3	3	×32.67
Synthesis without removing groundtruth from the motion graph.										
G_E	✓	✓	100,686	317,600	4.3 (5.0)	111	0	0	0	×32.67

calculated by directly comparing the positions from all bone joints (from pelvis, lowerback, upperback, neck, head, femur, tibia, foot, toes, clavicle, humerus, radius, hand) in each synthesised motion pose to the groundtruth joint positions, and acquiring the average position error over the entire 20 sequences. A potential issue with this metric is that it evaluates each body pose as a single connected object. As such, an error in one bone can negatively impact the accuracy result of other connected bones. For example, an error in the shoulder joint will subsequently cause errors in the connected arm bones (even if the local arm joint angles may be perfect), and in essence the error can be counted twice (or more). However, as this metric has been used in prior literature we adopt it for fairness of comparison. From the first four rows of Table 2, precision increases are obtained from including MSVs and interpolation into a graph.

We also evaluate our results on a higher semantic level, determining the number of correctly synthesised tennis stroke *actions* in each sequence. Within the 20 test sequences, 111 tennis strokes were performed – 20 serves, 43 forehands, 40 backhands (13 single-handed, 27 double-handed) and 8 overhead volleys. In Table 2, True Positive (**TP**) represents the number of correctly performed strokes in the synthesised motion (where a correct shot is one that is performed *in full* and the peak of the shot, i.e. the time when the ball is hit, is no further than 0.25 seconds from that of the groundtruth motion), False Positive (**FP**) represents strokes made at times when no such action occurred in the groundtruth motion, True Negative (**TN**) represents the number of strokes that occur in the groundtruth motion but no stroke (at all) was performed in the synthesised motion, and False Negative (**FN**) represents the number of incorrect strokes made by the reconstruction (e.g. the algorithm synthesises a forehand instead of a backhand).

From Table 2, graph G_A correctly performs $98/111$ strokes, never performs a shot when it should not, misses four stokes completely and misclassifies nine shots. Of the four strokes it missed, three are relatively unique strokes that appear in the database which were not selected to be part of the test graph, as such it is not surprising that they are synthesised incorrectly. The fourth missed stroke is correctly synthesised if MSVs (G_B) or interpolation (G_C) is incorporated. All but one of the nine misclassified strokes were overhead volleys incorrectly synthesised as forehands. By adding interpolated sequences, the graphs become more connected and more examples of overhead volleys are introduced, resulting in a reduction in the number of misclassifications to three or four. Graph G_E illustrates that doubling the number of nodes and edges in the graph by using all 8.5 minutes of data captured in the database leads to a slight decrease in joint error, but does not alter the accuracy of actions performed. This higher joint error accuracy is offset by an increase in the computational time needed to acquire a solution. Using the baseline approach (graph G_A), an optimal search solution for an example 24 second sequence can be obtained in only 5 seconds (on a standard Intel Core2 Duo 3.0 GHz processor, with 4 GB of RAM). This rises to 163 seconds when using graph G_E, or 32.6 times the computational cost of the baseline approach. These timings do not include the offline metadata calculation, processed on average at $2,567$ edges per second. The offline motion graph creation process (which is only implemented once to create the graph) is processed on average at 51 Vicon frames per second on a standard desktop computer.

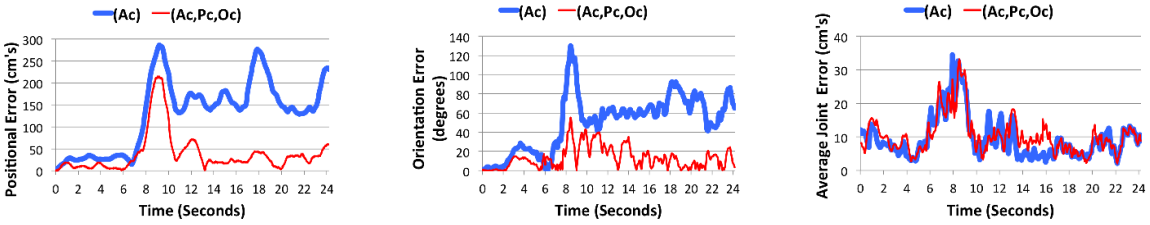
By examining the last row of the table, which uses graph G_E again but incorporates all the groundtruth frames in the search, an evaluation can be made on how well the virtual accelerometers model the readings of the real devices. Ideally, the average joint error should approach zero, as the groundtruth poses should be selected in each synthesised frame. However, the 4.3 cm error indicates a discrepancy between the real and idealised sensor models. This error is caused by a number of reasons including; tracking inaccuracies in the motion database, not incorporating all physical forces such as centrifugal acceleration and vibrations in the model, and inaccurate synchronisation and calibration.

6.2. Synthesis using Global Data

Although the results of Section 6.1 tend to produce good accuracy in terms of average joint error and stroke selection,

Table 3: *Synthesis with graph G_D for different input modality configurations.*

	Number of Sensors	Equation 6 Weights W_A	W_P	W_O	Average Joint Error (St.D.) in cm	Avg. Position Error (St.D.) in cm	Avg. Orientation Error (St.D.) in Degrees	Actions TP	FP	TN	FN
Three synthesis results using various accelerometer streams only.											
a	6	1	0	0	11.1 (6.1)	142.8 (114.3)	41.1 (27.3)	105	0	3	3
b	2	1	0	0	11.8 (6.3)	154.6 (137.6)	39.1 (35.8)	105	0	3	3
c	1	1	0	0	12.5 (6.8)	173.5 (163.8)	56.4 (44.2)	98	0	2	11
Synthesis using position and position-orientation input data.											
d	0	0	1	0	20.9 (10.4)	27.4 (21.5)	50.3 (48.6)	15	32	79	17
e	0	0	1	1	17.7 (8.4)	27.7 (22.6)	17.3 (12.4)	25	10	68	18
Three synthesis results using multiple input modalities.											
f	6	0.5	1	1	11.6 (7.0)	44.7 (50.4)	17.6 (13.2)	100	1	3	8
g	2	0.5	1	1	11.4 (6.4)	51.4 (47.0)	22.4 (15.1)	105	0	2	4
h	1	0.5	1	1	12.0 (6.7)	49.9 (52.2)	22.1 (14.7)	98	0	2	11
Synthesis with added human input constraints.											
i	1	0.5	1	1	11.9 (6.7)	58.5 (61.3)	21.6 (15.0)	110	0	0	1



(a) Global positional error over time (b) Global orientation error over time (c) Average joint error over time

Figure 6: *Temporal errors from a specific example test sequence using graph G_D and 6 inertial sensors.*

due to the vigorous and rapid movements in our test corpus, the approach performs less well with respect to global position and orientation (compass heading) errors as seen in row **a** of Table 3, where average errors of 1.4 metres and 41.1 degrees are obtained respectively. Conversely when using the approach of Section 5.3 with no acceleration data good position and orientation data can be obtained, at the expense of joint accuracy – see rows **d** and **e** of Table 3. Note that the position error in this table is calculated using the groundtruth Vicon data as a baseline. However, if the error is is calculated using the input data from the overhead camera as a baseline, the error value drops by roughly 13.7 cm on average (i.e. the calibration of the Vicon system with respect to the overhead camera is on average 13.7 cm off). Other experiments using position only information can be even lower, for example the 10 minute sequence of motion obtained from the publicly available HDM05 [MRC*07] dataset (as seen in the supplementary video associated with this paper) obtains an average position error of just 2.9 cm with a standard deviation of this error of 1.1 cm. This error is significantly lower as the movements around the scene are far less fast and aggressive than those in our test scenario.

By weighting the six accelerometer streams, at $W_A = 0.5$, with global input data using the approach of Section 5.3, significant improvements in global positional and orientation error values can be obtained, for a minor increase in the local joint synthesis error – see row **f** of Table 3. The benefits of using this approach can also be seen in Figure 6, which depicts how errors evolve temporally in an illustrative example sequence where a **TN** (i.e. missed shot) occurs at the 9 second mark. At this point regardless of the input data modalities, motion is synthesised poorly. This introduces significant global position and orientation errors into the animation. However, unlike the accelerometer-only approach **(Ac)**, when synthesising motion using both local and global data modalities **(Ac,Pc,Oc)** the animated motion is able to quickly recover, removing the error in position without significantly compromising local joint accuracy.

Using the results of our quantitative experiments, we determined that the optimal results using accelerometer-only synthesis were achieved in our tennis scenario when all six accelerometers were used (i.e. row **a** of Table 3). This result is expected as the data streams simultaneously reduce the ambiguity of motion at six independent body locations.

When only two (one on each forearm) or one sensor (on the right forearm) was employed, a subsequent increase in the average joint error was observed (see rows **b** and **c** of Table 3 respectively). However this was *not* the case when the accelerometer data was augmented with global position and orientation data (compare the joint error of rows **g** and **h** to row **f**), where the error from the one and six sensor setups were roughly on a par, and the two sensor setup performed significantly better, both in terms of joint error and the number of **TP** shots correctly synthesised. For a fully automatic approach in our tennis scenario, the best results were obtained by discarding the use of all but the two arm sensors *and* incorporating global data. In this setup, the two arm sensors provide adequate information to correctly synthesise 95% of all shots correctly, while the weighted global data reduced motion ambiguity between shots in a more efficient manner than the combination of leg, chest and hip acceleration information.

6.3. Synthesis using Manually Defined Constraints

For configuration **i** of Table 3 a semi-automatic approach was adopted, incorporating manually defined human input constraints into the setup. In this scenario, the motion dataset is manually annotated so that each pose is labelled with an action type (forehand, backhand, serve, volley, or none); the input constraints, $r(t)$ of Equation 8, are used to define the temporal location of shots to be performed in a sequence (defined as "don't care" at all times apart for ± 10 frames around the time when a ball is hit). This minimal human input ensures that if the connectivity of the motion graph is strong enough (in one case it was not), a correct shot is always obtained using a single right-arm sensor (that can perhaps be embedded into a tennis racket in a real-world scenario), while simultaneously minimising the costs of Equation 6.

6.4. Complexity, Memory Reduction and Windowing

Using the memory reduction technique of Section 5.1, an almost linear compression ratio with the number of samples added to the DP lattice was achieved as illustrated in Figure 9(a), achieving an average compression of over $120:1$ with 600 input data samples added. Using any reasonably well-connected motion graph should result in similar high-compression ratios. With regards to the added complexity of synthesising motion using global input data, the time taken to process each edge increases by an average of 28.4%, see Figure 9(b) which graphs a timing comparison between the two techniques using a number of different example sequences and graph sizes.

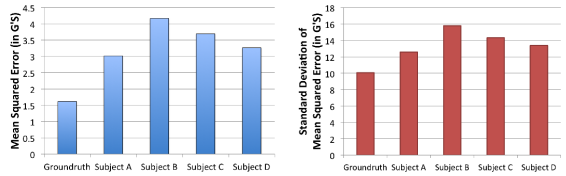
In Figure 10, we adapt the approach to operate with a lower latency, using a sliding window-based approach and evaluate the resultant depreciation in synthesis accuracy with two different motion graphs. The baseline results are obtained by synthesising motion using all input data streams **(Ac,Pc,Oc)** without windowing. In either figure, regardless of the input modality, roughly a 5 second window is required to reach joint error levels similar to that of the baseline approach. However, to achieve similar *global* positional errors, a 10 second window is required when using graph G_A (the most appropriate motion graph for a real-time scenario), or a 40 second window with the larger graph G_D. For graph G_A a sliding window size of less than 1 second results in the global input data having little, or no, impact on the synthesised motion. This rises to 3 seconds when using G_D.

6.5. Synthesis of Different Subjects

In this final experimental results section, we perform an investigation into how accurately the motion graph of one player (using motion graph G_D) and be used to drive the synthesised motion of three different player subjects. Unfortunately, due to budgetary constraints and the significant cost in cleaning up motion data captured in sub-optimal outdoor lighting conditions, the groundtruth motion capture data for each of the three additional subjects could not be obtained. As such, an alternative, more rudimentary form of quantitative results are provided than those previously presented in this paper, and are solely intended to act as an indication of its potential performance. In order to obtain quantitative results, we synthesised motion from each of the subjects using a single accelerometer (placed on the right hand of the subject). The virtual accelerations obtained from these synthesised motions were then directly compared to the real, input, accelerometer data. The closer the match in the real and virtual accelerometer readings, the higher the probability that a close match between real and synthesised motions have been acquired. In Figure 7 we present these results, and for qualitative evaluation readers are also directed to the supplementary videos associated with this paper.

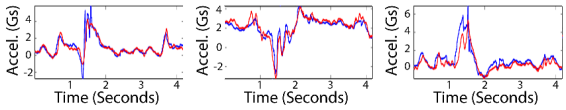
To establish a base error rate, we firstly compare the virtual and real accelerations from *groundtruth* Vicon motions. Theoretically, this error should be zero, however as seen in Figure 7 an average mean squared error of rate 1.6 g-force units (or g, were $1\text{g} \approx 9.81\text{ms}^2$), with a standard deviation of 10.0g, was obtained for the *Groundtruth* results. The discrepancy that is inherent in the two signals can be clearly seen in Figure 8, where time-slice of real and virtual accelerometer data are overlaid. The variance between groundtruth readings has five main causes: (1) inaccuracies in the motion database, caused by inaccurate marker tracking, calibration and data clean-up; (2) the virtual model is rigidly attached to the limb of the motion capture skeleton, while a real sensor can slide, move and vibrate against a limb, especially during high speed motions; (3) the virtual sensor is not modelled to incorporate all physical forces, such as centrifugal acceleration; (4) inaccurate calibration of the exact sensor locations and orientations; and (5) inaccurate synchronisation between the real-world sensors.

From Figure 7 it can be seen that when the captured player



(a) (b)

Figure 7: *Comparison of errors between virtual and real accelerometer data for different players; (a) Average mean squared error; (b) Standard Deviation of the error.*



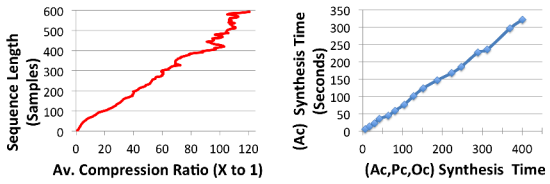
(a) X-Axis (b) Y-Axis (c) Z-Axis

Figure 8: *Comparison of real (blue) and virtual (red) groundtruth accelerometer data.*

and motion graph are of the same actor (i.e. *Subject A*) the error rate rises to 3.0g (or a 87% increase in the base groundtruth error rate). As shown in Table 3 row **c**, this error rate allows still allows accurate synthesised motions to be obtained. When motion of different subjects is synthesised, an increase in the error is expected and seen. The smallest increase in error occurs with *Subject D*, where the error rises to 3.26g (or an 8% increase with respect to that of *Subject A*'s error). The largest increase in error occurs with *Subject B*, with a 4.16g error (a 38% increase in *Subject A*'s error). It should be noted that these error increases are relatively minor when compared to the initial 87% increase from the base groundtruth error rate experienced by *Subject A*. However, as can be seen in the supplementary videos, the increase in error tends to expedite drift with respect to a subject's global position and orientation, and as such highlights the importance of incorporating global data into the search algorithm for these specific capture scenarios.

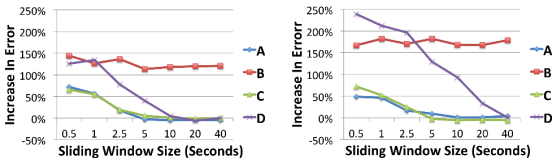
7. Discussion

In this work we describe a data-driven approach that can incorporate multiple and diverse input data modalities to synthesise accurate high-quality motion animations. For inertial sensor data, an optimal path through a motion graph with respect to the defined cost function is guaranteed. In addition, global position and/or orientation data can be efficiently incorporated into the algorithmic process, which can significantly reduce global errors, while the computational complexity remains linear with respect to time. The approach can be used with large motion graphs and input data sequences and can be adopted to allow human performance synthesis with light instrumentation, or to create animations that adhere to manually defined animator requirements.



(a) (b)

Figure 9: *(a) Average memory compression ratio over time; (b) Timing comparison synthesis of accelerometers-only **(Ac)** Vs accelerometers, position and orientation data **(Ac,Pc,Oc)**.*



(a) Results using graph G_A. (b) Results using graph G_D.

Figure 10: *Percentage increase in average joint error and average position error from a baseline approach when using various sliding windowing sizes; **A** Average Joint Error (Accelerometer Input Only); **B** Average Position Error (Accelerometers Only); **C** Average Joint Error (All input data streams); and **D** Average Position Error (All input streams).*

As with many data-driven techniques, an inherent limitation of the approach is the inability to reconstruct free-form motion that is significantly different from the movements stored in the motion database. Although we mitigate the issue by broadening the type of movements allowed using MSVs and interpolation, and ensuring that we begin with a relatively complete database of motions expected to be performed at capture time, this remains an inherent limitation of the approach. Although a relatively complete database of movements can be captured quickly for many activities, especially sporting scenarios that have a limited number of well-defined motions that occur frequently, acquiring appropriate motion databases for some scenarios (such as skiing, driving, and biking) in a lab-setting would be difficult. However, in Table 1, only with the approach of Shiratori et al. [SPS*11] can these specific motions be captured with global root positioning.

A further limitation lies in the simplified virtual sensor model employed, as discussed in Section 6.1. In our test scenario, we attempt to mitigate a number of the causes of virtual sensor error by using tightly fitting sensor straps to maintain a fixed position on the person's body, performing predefined motions to acquire accurate sensor locations, as well as performing motion clean-up to ensure a good motion database. In future work, we would likely expand the model to incorporate more sophisticated modelling. However, am-

biguity from a one-to-many mapping from the (small number of) accelerometer readings to body joint angles is an inherent issue and is more difficult to overcome. For alternative application scenarios, the use of alternative or additional inertial sensors, such as gyroscopes or IMUs, could be investigated as they can be easily incorporated into the approach.

Finally, an equal importance weighting is currently given to each worn sensor device. However, as seen in Section 6.2, some sensors provide more distinct and valuable information, whereas the additional information provided by others can be negligible. Quantitative experiments can tell us the optimal sensor locations for specific application scenarios. In future work, we plan to analyse the motion graph as a whole and use this information to associate higher weightings to the metadata of specific graph edges that have the most distinctive readings and therefore the best chance of leading to a good match. In our tennis scenario, for example, the weighting given to the arm sensors would be automatically set to significantly higher levels than the other data streams because their readings tend to be more distinct with respect to the database motions.

Acknowledgements

We are grateful to Moshe Mahler, Justin Macey, Iain Matthews and the Tennis Ireland athletes for their resources, discussions and suggestions. This work is supported by Science Foundation Ireland under grant 07/CE/I1147, the Tyndall National Institute under NAP Grant 209, and the EU FP7 project REVERIE, ICT − 287723.

References

[AF02] ARIKAN O., FORSYTH. D. A.: Synthesizing constrained motions from examples. *ACM Transactions on Graphics, Special Issue: ACM SIGGRAPH, 21*, 3 (2002), 483–490. 2, 3

[AFO03] ARIKAN O., FORSYTH D. A., O'BRIEN J. F.: Motion synthesis from annotations. *ACM Transactions on Graphics, Special Issue: ACM SIGGRAPH, 22*, 3 (2003), 402–408. 3

[AY05] ABBASFAR A., YAO K.: Survivor memory reduction in the viterbi algorithm. *IEEE Communications Letter 9* (2005), 352–354. 4, 6

[CKCO09] CONAIRE C. O., KELLY P., CONNAGHAN D., OÕCONNOR N. E.: Tennissense: A platform for extracting semantic information from multi-camera tennis data. In *International Conference on Digital Signal Processing* (2009), pp. 1–8. 8

[Cod12] Codamotion. http://www.codamotion.com, 2012. 1

[For73] FORNEY G.: The viterbi algorithm. *Proceedings of the IEEE 61*, 3 (1973), 268–278. 4, 6

[GSKJ03] GLEICHER M., SHIN H. J., KOVAR L., JEPSEN A.: Snap-together motion: assembling run-time animations. In *Symposium on Interactive 3D graphics* (2003), pp. 181–188. 3

[HBL11] HA S., BAI Y., LIU C. K.: Human motion reconstruction from force sensors. *ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2011), 129–138. 3

[KGP02] KOVAR L., GLEICHER M., PIGHIN F.: Motion graphs. *ACM Transactions on Graphics, Special Issue: ACM SIGGRAPH, 21*, 3 (jul 2002), 473–482. 4

[Kin12] Kinect. http://www.xbox.com/kinect, 2012. 1

[KR05] KEOGH E., RATANAMAHATANA C. A.: Exact indexing of dynamic time warping. *Knowledge and Information Systems 7*, 3 (2005), 358–386. 5

[KSG02] KOVAR L., SCHREINER J., GLEICHER M.: Footskate cleanup for motion capture editing. In *ACM SIGGRAPH/Eurographics Symp. on Computer Animation* (2002), pp. 97–104. 2, 3, 4, 5, 6

[Kum08] KUMAR M.: *A Motion Graph Approach for Interactive 3D Animation using Low-cost Sensors*. Master's thesis, Virginia Polytechnic Institute and State University, 2008. 3, 4

[LCR*02] LEE J., CHAI J., REITSMA P., HODGINS J. K., POLLARD N. S.: Interactive control of avatars animated with human motion data. *ACM Trans. on Graphics 21*, 3 (2002). 2, 3, 4

[LH01] LEE J., HA I.: Real-time motion capture for a human body using accelerometers. *Robotica 19*, 6 (2001), 601–610. 3

[LWC*11] LIU H., WEI X., CHAI J., HA I., RHEE T.: Realtime human motion control with a small number of inertial sensors. In *ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (2011), pp. 133–140. 3

[MRC*07] MÜLLER M., RÖDER T., CLAUSEN M., EBERHARDT B., KRÜGER B., WEBER A.: *Documentation Mocap Database HDM05*. Tech. Rep. CG-2007-2, Universität Bonn, June 2007. 2, 8, 10

[PSKS04] PARK S. I., SHIN H. J., KIM T. H., SHIN S. Y.: On-line motion blending for real-time locomotion generation: Research articles. *Computer Animation and Virtual Worlds 15*, 3-4 (2004), 125–138. 5, 6

[RP04] REITSMA P., POLLARD N.: Evaluating motion graphs for character navigation. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (August 2004), pp. 89–98. 3

[RSH*05] REN L., SHAKHNAROVICH G., HODGINS J. K., PFISTER H., VIOLA P.: Learning silhouette features for control of human motion. *ACM Trans. on Graphics 24*, 4 (2005), 1303–1331. 3, 4

[SH07] SAFONOVA A., HODGINS J. K.: Construction and optimal search of interpolated motion graphs. *ACM Transactions on Graphics, Special Issue: ACM SIGGRAPH, 26*, 3 (2007), 402–408. 2, 3, 4, 5

[SH08] SLYPER R., HODGINS J.: Action capture with accelerometers. In *2008 ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (July 2008), pp. 193–199. 3, 6

[SPS*11] SHIRATORI T., PARK H. S., SIGAL L., SHEIKH Y., HODGINS J. K.: Motion capture from body-mounted cameras. No. 31, pp. 31:1–31:10. 3, 12

[TL06] TIESEL J., LOVISCACH J.: A mobile low-cost motion capture system based on accelerometers. In *International Symposium on Visual Computing* (2006), pp. II: 437–446. 3

[TZK*11] TAUTGES J., ZINKE A., KRÜGER B., BAUMANN J., WEBER A., HELTEN T., MÜLLER M., SEIDEL H.-P., EBERHARDT B.: Motion reconstruction using sparse accelerometer data. *ACM Trans. on Graphics 30* (2011), 18:1–18:12. 3, 8

[VAV*07] VLASIC D., ADELSBERGER R., VANNUCCI G., BARNWELL J., GROSS M., MATUSIK W., POPOVIĆ J.: Practical motion capture in everyday surroundings. *ACM Transactions on Graphics 26*, 3 (2007), 35. 3

[Vic12] Vicon. http://www.vicon.com, 2012. 1, 7

[WF02] WELCH G., FOXLIN E.: Motion tracking: No silver bullet, but a respectable arsenal. *IEEE Computer Graphics and Applications 22*, 6 (2002), 24–38. 2

[XSE12] Xsens. http://www.xsens.com, 2012. 3