ABSTRACT
Because of advances in micro-electro-mechanical sensor technologies, animation studios are adopting body sensor networks to reconstruct human movements. This paper introduces a novel software environment designed to create and customize body sensor networks by interfacing with heterogeneous sensor devices. Customizability is achieved through a new file format consisting of sensor definitions, configuration and connectivity semantics in conjunction with traditional motion data. Body sensor networks require ample transfer speeds and synchronization protocols to ensure that angular readings are computed accurately. Those concepts are highlighted through two contrasting case studies of software and hardware-centric motion processing. In the first study, several wireless sensor modules are integrated using Bluetooth. In the second study, a prototype motion capture suit, featuring a motion processing multiplexer, is integrated to form a comparison.

KEY WORDS
Animation, Motion Capture, Inertial Body Sensor Networks, Hardware-Centric, Software-Centric.

1. Introduction
With advances in computer graphics, virtual space has become an immersive experience for audiences. In the context of filmmaking and video games, realism is achieved by reconstructing organic and inorganic objects as animated 3D topologies. Character animation is particularly problematic because human motion is complex in nature. Even the most detailed character rigs are minimalistic in comparison with the musculoskeletal intricacy of the human body. With advances in motion capture, traditional frame-by-frame techniques are becoming optional. The motion of a human can be captured quickly and accurately using optical cameras or inertial sensors. Motion capture is led by the optical medium, but because of advances in the miniaturization of microelectronics, inertial body sensor networks are becoming a viable alternative.

Typical full-body motion capture suits feature between fifteen and twenty inertial measurement units (IMU) for a full-body configuration. We can classify an IMU as being either software-centric (a device that outputs raw unprocessed data) or hardware-centric (a device that outputs processed motion from its on-board microcontrollers). This paper presents two corresponding studies of integration while providing an overview of motion processing. Focus is placed on the challenges faced in integrating two dissimilar motion capture systems rather than proving or disproving which approach is better in terms of performance.

In the first study, several software-centric Shimmer 2R devices [1] [2] [3] are connected to the proposed software environment over Bluetooth to construct an upper body sensor network. In an idealistic interference-free environment, the Bluetooth communications specification supports only fourteen channels. Consequently, full body motion capture was not possible due to Bluetooth limitations. Each Shimmer 2R device contains an InvenSense 500 series gyroscope and a Freescale accelerometer. This study also discusses the development of a driver for retrieving, synchronizing and processing angular rates at the software level from the Shimmer devices.

In the second study, a new prototype motion capture suit is discussed to highlight a hardware-centric approach for creating body sensor networks whereby the hardware microcontrollers are tasked with motion processing. The hardware consists of a multiplexer and several inertial measurement units containing InvenSense IMU3000 gyroscopes and Freescale MMA8 series accelerometers. The multiplexer’s microcontroller can accommodate twenty connections but, for the purpose of this experiment, only seven were used to achieve upper body motion tracking. The system’s constituent nodes are daisy chained using a twisted pair cable that facilitates adequate data transfer speeds. The main focus of this prototype has been to establish a functional body sensor network (BSN) in terms of hardware and firmware. Future studies will be focused on evaluating the system’s performance against more established systems such as the Animazoo IGS systems and Xsens MVN.

Both studies are achieved using a novel software environment (see section 3) designed specifically for hardware integration, customization and data analysis. It provides the functionality for monitoring and modifying individual stages of motion processing: sensor fusion, dead reckoning and computer-hardware
intercommunications. Motion data from each IMU device can be visualized in its raw, intermediate or processed form. To interface with new hardware, Skeletrix, our software environment provides a modular driver development kit (DDK). Using Skeletrix, heterogeneous devices can function in parallel and display data on a shared skeletal rig. All hardware specific configuration information is logged through our extended version of the popular Biovision Hierarchy format.

2. Body Sensor Networks

BSNs are constellations of sensory devices that measure key properties in the human body [4] [5] [6]. In the context of motion capture, BSNs employ IMUs typically containing gyroscopes, accelerometers and magnetometers. Data originating from each component is gathered and fused to produce the appropriate angular rotations required for animation.

Abstractly, the human body can be seen as a list of articulated segments and represented as a kinematic skeleton. IMUs are placed strategically on each major body part (e.g. torso, limbs, neck, head etc.) to measure motion. Larger IMUs may be strapped on top of clothing, while smaller devices can be sown directly into fabric. While performing gesticulations, even the tightest straps will slide in relation to the skin, and the skin will move in relation to the fat and muscle tissues causing artefacts in the motion data. This problem is referred to as sensor distance noise [7] and outlines a major problem that is specific to inertial motion capture.

Although inertial suits are a modern concept, the idea behind BSNs has been exploited in many application areas outside animation. In medical science, hospitals use telemedical networks [8] to monitor patient vital signs, measure rehabilitation progress and conclude treatment solutions. For example, measureable factors may include pulse, heart rate, blood pressure, blood oxygenation, etc. Measuring subtle movements can provide important information about a patient’s health. For example, Mercury [9] is an inertial BSN used to supervise patients with epilepsy or Parkinson’s disease. This system consists of eight sensor modules placed on each limb and connected wirelessly to a laptop. Wireless interconnectivity is used to avoid the inaccuracies caused by cables interfering with the patient’s natural motor functions.

Aside from practicality, what is the difference between wired and wireless BSNs and more specifically, software and hardware-centric motion capture hardware? The answer to this question can be found by looking closely at the inner workings of inertial suits. A typical animation contains up to 100 frames per second of motion. However, IMU sensors output angular rates at frequencies exceeding 1,000 Hz. Sensor frequencies may also be referred to as sampling rate or output data rate (ODR). High ODRs will produce large data sets that require compressing. Sensor fusion and motion processing algorithms are used to reduce that data in length while improving accuracy. Having a software-centric approach requires all data to be sent to an external computer for processing. This approach puts stress on the computer-BSN intercommunications and can lead to inaccuracies if data is lost. Wireless data transfers are achieved in bursts using large buffers to ensure that no readings are truncated. Each independent device may function at slightly different speeds depending on operating temperature and other factors. Consequently, IMUs in BSNs require synchronization.

Wired BSNs can contain more advanced microelectronics because there are fewer power-usage constraints. For example, the ETH Zurich Sensor Hardware (48 accelerometers) and the Lancaster Multi-Accelerometer Platform (30 accelerometers) are wired BSNs [7]. The Lancaster multi-accelerometer platform is embedded in a pair of trousers and a lab coat while ETH Zurich uses straps. Wired hardware-centric BSNs require multiplexer devices to form an intermediate gateway between the computer and sensors. Each node’s microcontroller undertakes sensor specific computations (i.e. signal conversions, sensor fusion etc.) while the multiplexer undertakes general computations (i.e. synchronization, handshaking, data packaging, etc.).

These two approaches have clear benefits and drawbacks. Wireless systems are more practical while wired systems are more powerful. This paper introduces the concept of heterogeneous systems and how they can function concurrently. Figure 1 illustrates the integration of both software and hardware-centric IMUs into a body BSN used experimentally in this paper.

Figure 1. Upper-body sensor networks combined: MTDS prototype on the left and Shimmer 2R on the right.
3. Skeletrix BSN Software Environment

Entitled Skeletrix, this software environment’s goal is to facilitate inertial motion capture development. It provides a suitable experimentation environment and architectural scaffolding for researchers interested in studying, modifying or engineering BSNs. To differentiate itself from other animation software, Skeletrix contains no functionalities for animation editing. Instead, it provides tools for: hardware integration, calibration, person dead reckoning, software-centric sensor fusion, etc. Skeletrix relies heavily on external Biovision Hierarchy Extended (BVHE) motion files as it stores no information natively (see section 3.2).

The Skeletrix architecture uses object-oriented C-based programming. Its five components can be categorised as mathematics, kinematics, animation, system and visualization libraries. First, the mathematics library uses quaternion (for performance reasons and to avoid gimbal lock [10]) and vector algebra. Second, the kinematics library is used to import or create skeletal rigs. Each skeletal bone can be assigned physical properties such as weight and centre of mass. Third, the animation library feeds motion capture data into the kinematic model as local-space rotations, world-space rotations or positions (for optical systems). Fourth, the system library is used to create virtual sensor objects that interface with hardware and apply motion data back to the kinematic model. It provides the capability for importing drivers and motion processing modules. A motion-processing module is an algorithm or data filter for tasks such as dead reckoning (calculating horizontal displacement during gait) and sensor fusion. Lastly, the visualization libraries are used to generate the UI and render OpenGL-based graphics. Multithreading becomes central when working with multiple libraries at the same time. Figure 2 illustrates the principal libraries and their constituent objects.

Qt was chosen for the graphical user interface (GUI) layer because of its OpenGL capabilities and its tools for developing highly customizable visuals. Qt is an open-source project that is widely used for developing software applications that require sophisticated frontends (e.g. Autodesk Maya). Figure 3 illustrates the main Skeletrix interfaces. The Skeletrix interface layer consists of four main UIs. The main UI (a), which prompts the user on start-up, is responsible for kinematic visualisations of character animation. The kinematics viewer (b) allows users to access and visualise more detailed kinematic information such as positional vectors, rotational transformations and hierarchy links. The system viewer (c) allows users to analyze the incoming stream of data as obtained from the hardware and processed by Skeletrix. This interface resembles that of the kinematics viewer in dealing with virtual sensor objects (as opposed to kinematic bones). Lastly, the animation viewer (d) provides access and the ability to export finalized motion data in a tabular format.

Figure 2. Simplified architectural diagram of Skeletrix.
3.1 Driver Development Kit

The driver development kit (DDK) provides researchers with a bridging architecture for connecting to inertial motion capture devices. In this context, drivers are self-contained dynamic link libraries (DLL). Although working examples are provided, drivers must be built in accordance to a strict set of guidelines. The bridging procedure has two important steps. The first step is validation through which a connection is established between the software environment and driver module. The system manager interrogates the driver and validates its attributes to ensure a trouble-free motion recording session. The second step focuses on hardware handshaking and is specific to each system. If the bridging is successful, data will begin to stream from the hardware. As shown in Figure 4, several drivers can be bridged simultaneously to stream data from multiple heterogeneous systems in real-time.

![Figure 4. System connectivity diagram combining s1 (wired BSN) with s2 (WBAN).]

3.2 Biovision Hierarchy Extended

Biovision Hierarchy Extended (BVHE) lies at the core of the Skeletrix architecture. This new file format serves the purpose of logging motion data while simplifying the overall user experience. As the title would imply, BVHE is an extended version of the popular Biovision Hierarchy (BVH) format. BVHE aims to tighten the relationship between software and inertial motion capture hardware. Users are free to utilize arbitrary numbers of inertial measurement units configured specifically to a recording session. To achieve that functionality, BVHE contains system configuration extensions in conjunction with traditional motion data.

The traditional BVH format encompasses a hierarchy definition in the file header and motion data in the file body. BVHE’s main addition is a system section containing the syntax for one or several inertial devices. In succession, each system is given: connectivity ports, driver links, a rotational space, a calibration method, a versioning parameter, channel definitions and one or several IMU definitions. In turn, each IMU definition contains a unique sensor name, a kinematic bone linkage, a rotational offset and a scale percentage. The percentage attribute is used mainly by exoskeleton systems to calibrate potentiometers. The following code snippet (Figure 5) illustrates the system section of a BVHE file.

```plaintext
SYSTEM s1 {
   PORT COMM03
   DLL def intensely
   SPACE world
   CALIBRATION dynamic
   VERSIONING v1.0
   CHANNELS xgyro xgyro xgyro
   xacc xacc xacc
   SENSOR imu3000_mma8450qr1_1 {
      BONE Chest
      OFFSET 0 0 0
      SCALE 100
   }
   SENSOR imu3000_mma8450qr1_2 {
      BONE Shoulder
      OFFSET 0 0 0
      SCALE 100
   }
   SENSOR imu3000_mma8450qr1_3 {
      BONE Head
      OFFSET 0 0 0
      SCALE 100
   }
}
```

![Figure 5. BVHE system code snippet for system s1.]

Actor file systems are used in inertial motion capture to match the kinematic rig to the user’s bodily proportions. Instead of making a bespoke actor file system, the BVH hierarchy definition was extended to accommodate actor data such as rotational, positional and length offsets. Additionally, weight properties are added to that hierarchy to aid person dead reckoning and perform other operations involving basic Newtonian physics. Representing all this extra information within an editable file allows users to customize the way in which they apply inertial motion capture technologies by simply using text editors.

4. Creating a Wireless Body Area Network

We chose the Shimmer Development kit as it provides a well-documented set of open source software and firmware to help development. Firmware can be flashed onto the Texas Instruments MSP430 microcontroller to customize the way in which the device operates. As previously mentioned, each IMU device contains an InvenSense 500 Series gyroscope and Freescale accelerometer. To create a wireless body area network (WBAN), data from several Shimmer 2R devices must be retrieved and unified at the software level. A driver was developed in the Skeletrix DDK to access data from a single device. Several driver instances can be used concurrently to form a Shimmer 2R network.

4.1 Interfacing with Hardware

The Shimmer 2R devices can be interfaced through Bluetooth or 802.15.4 radio. For this experimentation we used the Bluetooth communication method provided. Each device is paired using a predefined password and given a virtual communication port (VCP) service. VCPs
allow these devices to emulate serial communication port connectivity. A serial communication port monitor can then be used to observe the I/O messages between the computer and hardware. Next, a communication port was opened using the appropriate settings and a connection was established with the Shimmer IMU. Confusingly, Shimmer has no on/off switch and there is no clear indication of the sensor being activated. Therefore, to check the device is functioning and connected, a single-byte test instruction can be sent to the device to toggle its LED. Although the communication port connection was verified, there was no incoming data from the device. This is because most inertial measurement units require handshaking instructions to activate data streaming. The Shimmer 2R handshaking instructions were identified by monitoring how the Shimmer 2R communicates with its native software. This handshaking procedure was emulated in the driver module and the device started streaming data back to the computer. Notably, each instruction sent must be followed by a pause to ensure that the device has time to respond. All incoming data was stored in the driver’s buffer. The structure of the response must be understood in order to decode and identify sensor data. Based on our previous experience with similar sensor chips, we knew what data to anticipate for the accelerometer and gyroscope. The incoming list of bytes must be divided into smaller, single frame, readings. Each frame of motion begins and finishes with one reoccurring delimiter byte. The maximum value that can be stored in a byte is 256 while the gyroscope’s zero rate output (ZRO) is 1350mV. Therefore, two bytes are required per axial reading. Six readings are expected per message (three for the gyroscope and three for the accelerometer) to represent a motion reading. Shimmer devices also produce timestamps that are useful for motion processing and network synchronizations. In summary, a typical IMU’s motion reading, containing gyroscope and accelerometer data, can be represented by fifteen bytes: one delimiter, two to represent the timestamp and twelve for motion.

4.2 Results

The gyroscope readings can be converted into angles by first subtracting the zero-rate output (ZRO) and then dividing the result by the sensitivity as shown in the following formula. Sensitivity, ZRO and other attributes can be found in the gyroscope’s datasheet.

\[
\frac{\text{Gyroscope Reading} - \text{ZRO}}{\text{Sensitivity}}
\]

The result is a list of angular rates that must be converted into world-space rotations before being applied within the kinematic model. Three Shimmer 2R devices were strapped to the right arm segments on the hand, forearm and upper arm. The purpose of this test was to measure the transition from a sitting posture to a standing posture. The motion data presented has a frame rate of 50fps and contains 300 frames of motion. Figure 6 shows a relaxed behaviour up to frame 85. Its rotations can be visualised on the graph as a list of high points and depressions. The transition is finalized and the spikes flatten at frame 220.

5. Motion Tracking Detection System

Our prototype inertial motion tracking detection system (MTDS), shown in Figure 1 and 7, is a hardware-centric BSN interconnecting homogenous IMUs. Its first goal is to provide modularity whereby sensor nodes can be added or removed in accordance with a recording session’s requirements. The suit can be used in its full-body configuration, upper-body (as for this experiment) or as an arbitrary array of sensors. Its second goal is to achieve plug-and-play simplicity of computer peripherals in the otherwise complicated medium of inertial motion capture. Upon start-up, the hardware is designed to perform all handshakes, calibrations and drift compensations autonomously. To achieve this functionality, the development was focused more on firmware and hardware development rather than software (which was kept relatively minimal). The hardware was developed in two stages: the multiplexer and the IMU device.

The multiplexer was designed as a belt-worn module containing four AA batteries, a Bluetooth emitter for wireless connectivity, a serial port connector for wired connectivity and a power switch. The harness uses twisted pair cables to avoid signal interference (a common problem with lengthy ribbon cables). The multiplexer is tasked with powering the constellation of IMUs. Instructions are sent to each sensor to either update
firmware (using a boot loader) or commence data streaming. Each node’s microcontroller must be interrogated at perfect time intervals while taking into account processor loop delays. Data is then gathered, validated, packaged and communicated to the computer by the multiplexer.

As previously mentioned, each MTDS IMU device contains an InvenSense IMU3000 gyroscope and a MMA8 series accelerometer. For simplicity, in this prototype we left out the magnetometer. In comparison with the Shimmer hardware, the microcontroller is a less powerful Atmel AVR 8-bit chip. Efficient resource allocation was key in achieving the desired functionality with this hardware configuration. The IMU’s microcontroller is first tasked with basic conversions to generate quaternion rotations from the gyroscope output. Although sensor fusion is not fully implemented, the gyroscope and accelerometer outputs can be combined or used separately depending on the experiment.

The suit needs to compute two types of angular compensations at both software and firmware levels: resting output and calibration. The resting output, also referred to as zero-rate output is the voltage reading (or ADC reading) of the sensors when the device is motionless. Each IMU was given a rotational threshold to identify whether the hardware is in a resting position. If an IMU is resting, its output is summed and averaged over a period of ten or more seconds. The result is subtracted from each rotation to reduce the resting output to zero. Usability-wise, the MTDS hardware must be left motionless on a flat surface to self-compensate rotations before being worn.

Calibration, whereby the kinematic model is adjusted to match the performer’s pose, is achieved at the software level. The performer is asked to mimic the onscreen pose while the software computes the rotational difference between the motion data and the kinematic model. Most commercially available systems, such as the Animazoo IGS systems, perform calibration by pre-setting the kinematic skeleton to assume a T-pose and instructing the motion performer to stand perfectly straight with their arms extended laterally away from the torso.

5.1 Results

Preliminary testing involved comparing real motion against its virtual reconstruction using a video stream (as shown in Figure 7). Despite some noticeable drift, the skeletal rig mirrors the performer’s movements for several minutes before drift is observed as significant. We found drift to increase as the motion performer moves more rapidly whereby the rotational increments are larger.
6. Conclusion

This paper demonstrates a more flexible approach for interconnecting heterogeneous IMUs as one BSN. The proposed software environment, Skeletrix, establishes a unified methodology (for gathering, processing and outputting usable motion data) that can be applied to a wide range of systems. Inertial hardware that varies in performance and cost can display data in Skeletrix as one system while relying on our extended file format (BVHE) for configuration. We tested the experimental DDK using two studies and successfully gathered motion data from Shimmer 2R devices and our in-house built motion tracking detection system (MTDS). In doing so, the paper presents a contrasting overview of our experience with software and hardware-centric motion processing. In the context of these two experiments, we found that wireless BSNs put more stress on the computer-hardware interconnectivity and computer resources while wired BSNs can function more autonomously.

6.1 Future Work

For MTDS to reach its potential, it requires full sensor fusion to further reduce drift, including the integration of magnetometers. Future iterations of this system will be more autonomous and more accurate. The goal is to create a motion capture suit that can function independently of a computer and outside the comfort of an animation-recording studio. Hardware upgrades will involve the implementation of 32-bit ARM processors, better MEMS technologies (e.g. the MPU9150 9-axis motion tracking device from InvenSense) and lithium ion battery/charger modules. Future studies will be focused on benchmarking the system against industry acclaimed systems such as the Animazoo IGS and Xsens MVN systems. Additionally, the implications of combining dissimilar motion capture devices alongside MTDS, concurrently, will also be investigated. Skeletrix will be released to provide researchers with a suitable experimentation environment for developing, testing and evaluating inertial motion capture technologies and more specifically, body sensor networks.

References