Movement Awareness for a Sentient Environment

Robert Headon
Laboratory for Communication Engineering
Department of Engineering, University of Cambridge, England
rph25@cam.ac.uk

***Abstract*— This paper describes a system that can observe, recognise and analyse human movements, to provide this awareness to context-aware applications. The movement recognition and characterisation components of this *sentient system* are described in detail. The system uses the ground reaction force to classify and analyse movements in an non-clinical environment. The signal is classified using statistical pattern recognition. Equipped with knowledge of the movement, characterisation is the process of analysing the ground reaction force to extract parameters of the movement. The components of the movement awareness system operate in a distributed computing environment.**

I. Introduction

Active environments are populated with devices that can sense, compute and communicate to provide information about their surroundings to applications. A *sentient system* uses sensing technologies in the environment, data processing and supporting middleware to maintain a representation of physical space in a world model, allowing shared perception between computers and people [1]. Information from the model is used by context-aware applications to change their behaviour according to the physical space. Location awareness has long been explored and motivated the development of many successful location sensing technologies and applications. This paper presents a new modality of awareness which allows applications to include perception of human movement in their operation.

In this paper movement is considered to be whole body, such as a step, crouch, jump, rising to stand, and not transition such as "moving backward". Localised movement, such as waving an arm, is considered to be a gesture. Movements can be characterised for a more accurate *description*, e.g. the height of a jump or distinguishing between left versus right foot during a step, etc.

During movement the forces exerted on the environment change. The force can be observed using a suitable sensing system and used to classify movement patterns. This research uses pressure sensors underneath a floor to observe the vertical component of the ground reaction force (GRF). The GRF is processed into feature vectors and Hidden Markov Models (HMMs) are used to model the movement signals. These models are used to classify movements, following which it is possible to characterise the movement by analysing the position and force information contained in the GRF vectors.

The processes that comprise the movement awareness system operate in a distributed computing environment. The sentient computing paradigm dictates that applications should have ubiquitous access to awareness of the environment. Distributed systems also allow flexibility in the location of processing components, such as recognition. In this research movements are represented as distributed CORBA objects. There may be a large number of movement objects and default servants are used to access the objects, providing a scalable solution. Movement objects are persistently stored in a repository.

Section II reports related work, considering approaches to observing and recognising movement. Section III presents an overview of the movement awareness system describing the interactions between the processes involved. Section IV exemplifies how movement affects the GRF and details the development of the movement classifier. Section V describes the repository and characterisation processes, including implementation, and considers jumping as a case-study. Section VI concludes this paper.

II. Related Work

Movement characterisation is an important technique in clinical analysis. It is primarily a supervised process, using the ground reaction force and manual recording techniques to assess a movement, usually to diagnose an abnormality or to appraise the rehabilitation of a patient. Clinical analysis has different requirements, applications and operating environment to a sentient system. Force plates are used to observe the force exerted by the user to the floor. Each corner of the force place sits on a load-cell, which are used to measure the GRF in one, two or three dimensions. Commercial force plates, such as Kistler [2], are expensive and unsuitable for wide-spread deployment. A number of weight-sensitive floors have been developed for deployment in active environments, with different requirements. The Olivetti and Oracle Research Laboratory (ORL) developed the Active Floor [3] to recognise people using their GRF trace during gait. The system used a custom array of nine force-plates, each 500 $\times$ 500 mm. This work was later replicated at Georgia Tech using a different feature set and a single force plate to model the GRF signal during gait [4]. A large force plate (2400 $\times$ 1800 mm) has been deployed to track a single moving person in an active environment [5]. These systems use load-cells, which measure the absolute force acting on the floor. VTT Information Technology developed a floor sensor using capacitive sensors that respond to changes in the pressure rather than absolute force. The floor is used to control immersive games [6] by sensing

the location of an individual player, or average location of a group when there is more than one player.

Movement tracking has been an active area of computer vision research for several years, and should be distinguished from movement *recognition*. There are several approaches to motion capture using vision; marker-based, edge detection and motion-energy images. The latter technique is an interesting untagged system for movement recognition. Motion-energy and motion-history images are used to recognise movements using temporal templates [7]. This approach represents motion as changes between sequences of images and uses the patterns of these changes to model a movement. Vision systems suffer when occlusion or background movement is present, operate in a limited workspace and tend to be resource intensive.

Accelerometers can be used to determine the angle between limbs for biomechanical analysis. This requires careful placement of the sensors and is a tagged, often tethered solution. Wearable devices fitted with an accelerometer have been demonstrated to decipher the user activity *states* of walking, running, sitting or standing [8], [9].

A representation of physical space is maintained in a world model, which allows shared perception of the physical world between computers and people. The world model contains information such as the the location and orientation of people and objects, environmental conditions, device capability and state, and perhaps a plan of the building. The SPIRIT [10] project maintains a comprehensive model of the environment, focusing on the location of objects and resource information. The location and resource data are represented as a set of persistent CORBA objects. These objects make up the world model that is seen by applications. The Microsoft Easy Living [11] system maintains a world model for an expected home environment. The model focuses on geometry and location of people and objects. They use pressure sensors to determine if a person is sitting or standing as the occlusion of the chair deteriorates the performance of their vision tracking system and movement awareness may be a useful input to the system. The objective of these systems is to maintain a model of the physical world, allowing applications to use the physical space as the interface with computers. Movement awareness is a new modality for these world models.

III. The Movement Awareness System

This section describes the distributed components of a sentient system used to recognise and characterise movements. Figure 1 shows the component interactions. This middleware layer enables all networked applications to access the awareness of movements in the environment. All components are implemented as CORBA objects. Communication is via method invocations over the ORB, and a notification service is used to communicate asynchronous events.

The system uses the Active Floor, which looks and feels like a regular floor, to sense the GRF. The floor consists of an array of load-cells beneath the floor to emulate an array of force plates. The load-cells are connected to a data acquisition device (DAQ) that samples the sensors at 1kHz and provides the signal conditioning and A/D conversion required for strain measurement, and is connected to a server. As the DAQ is tightly coupled to the sensors of the Active Floor, which is in turn distributed throughout the environment, the DAQ is represented as a CORBA object and the sensor data can be streamed over the LAN to other distributed applications that process the data. The Ground Reaction Force Extraction component takes the strain observations, and knowing the arrangement of the load-cells, determines the magnitude and location of the forces acting on the floor. These GRF vectors can be used to track the location of an object on the floor and for movement recognition and characterisation.

The Movement Recognition component processes streams of GRFs. The first operation is to detect whether movement is present, termed static detection. If a movement is present, the magnitude of the GRF is processed into observation vectors that are used in statistical classification of the signal. This implementation uses HMMs and the HMM Toolkit (HTK) [12] to implement the classification system. Observation vectors are sent to the recogniser (HTKRecog), which returns a list of results, i.e. the movements in the corresponding GRF stream. The recognition component requests the Movement Repository to create a movement object by invoking the appropriate method and passing the recognition result and GRF vector. The movement object is created and in doing so is *intra*-characterised. A reference to the object (IOR) is returned. The movement IOR and recognition result are presented as a structured event and the OMG Notification Service [13] is used to disseminate this event to clients.

The Movement Characterisation component seeks to relate movements for *inter*-characterisation. This service listens to movement events on the notification channel and compares them to a list of previous movements to investigate if they are related. If a relation is determined, the movements are inter-characterised and attributes of the movement are updated. The movement event is then re-sent to the notification channel to notify clients that the movement has updated parameters.

The information generated by the system enables applications to respond to movement conditions and events in the environment. One destination for the information could be an existing world model, which acts as the mediating layer between applications and the awareness of physical space. An application could also use the information directly by receiving movement events from the notification channel, accessing the created movement objects through the default servants, and observing the GRF if necessary. An example would be the controller of a 3D first-person game which uses actions in the real world and maps them to actions in the game [14]. When a movement event, such as a jump, is received, the controller causes the on-screen character to perform the same action.

IV. Movement Recognition

This section describes the Recognition component of the movement awareness system. An example is presented to relate the effects of movement on the GRF. The section continues to specify the signal model and features used for classification.

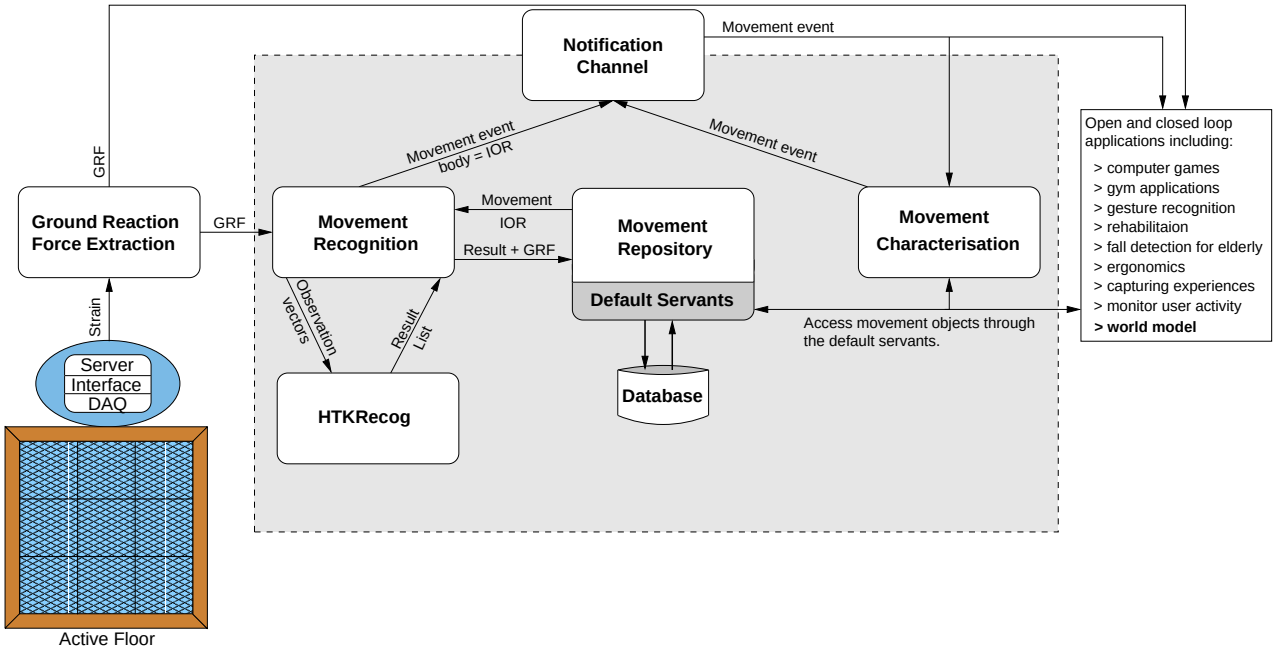

Fig. 1. Components of the movement awareness system.

The model parameters are specified and performance of the classifier evaluated.

A. The GRF and Movement

As body mass is short term fixed, the force experienced by the floor is dependent on the acceleration of the body acting upon it (treating the trunk as centre of mass). If the GRF is less than body weight, then the weight of the body is not being supported by the floor and this signifies acceleration downwards. For example, when you crouch there is a downward acceleration of the body and so the GRF will be reduced. And conversely, when thrusting the body upwards (as in a jump) additional force is required to propel upwards, and this is experienced by the floor. The GRF for the crouch, jump and drop-land movements is presented in Fig. 6.

B. Signal Model and Feature Selection

The *statistical approach* to pattern recognition represents a pattern by d-features, viewed as a point in d-dimensional space. Training patterns, defined from a large set of sampled movements and captured under a range of conditions, are used to determine the decision boundaries. The Hidden Markov Model [15] is one type of statistical model and is chosen for classifying the movements in this research. HMMs are ideal as they implicitly model all of the many sources of variability inherent in real movements, and are shown to work well in practice. The duration of the movement is inherently incorporated by the choice of HMM type (left-right model), which has the property that as time increases, the state index increases or remains the same. The duration is modelled by the number of states and the state transition probabilities.

The choice of good features for time-series discrimination is a fundamental step in statistical pattern recognition and is a highly problem dependent task. Ideally features will not be based on a person's individual characteristics. The addition of a new person would necessitate the training of new models for each movement, and the addition of a new movement would necessitate the training of a new model for all the people. Futhermore the recognition process would have many more models to evaluate which would degrade performance. The underlying biomechanical process is similar between individuals, in the normal case, and can be modelled to classify movement. The major individual difference is body weight, which scales the magnitude of the forces exerted during movement. To eliminate this the GRF time-series is normalised by body weight, determined during a period of static. This results in an acceleration time-series which is used in feature extraction.

The acceleration time-series is segmented into windows of 20 ms duration, i.e. 20 samples at 1kHz sample rate. Features are extracted and represent this segment of the waveform. The three primary features are the mean ($\bar{x}$), the standard deviation (s), and the slope (m). The *mean* provides information on the intensity of the signal and is a simple description of the underlying trend. The *standard deviation* is a measure of the variation of the signal. The *slope*, calculated from the acceleration time-series, captures the rate of change of acceleration. The slope is that of a least-squares straight line fit of the windowed data. These features ($\bar{x}$, s, m) represent a single window. Delta and acceleration (i.e. first and second order regression) coefficients are appended to the feature vector to add time derivatives between the windows. The delta coefficients are computed from (1) for each of the basic features ($\bar{x}$, s, m), with c_t representing the basic feature to which the regression is applied. Applying (1) to the

computed delta coefficients yields the acceleration coefficients. The feature vector is presented in Fig. 2, where x_i represent the normalised GRF data points of the window and y_1 and y_n are the end-point values of the straight line fit in the window.

$$d_t = \frac{\sum_{\theta=1}^{2} \theta(c_{t+\theta} - c_{t-\theta})}{2\sum_{\theta=1}^{2} \theta^2} \quad (1)$$

C. Model Parameters and Performance

The movement classifier uses continuous density HMMs in a Bakis-1 model configuration. This is a left-right arrangement in which it is only possible to remain in the current state or progress to the next. This form is chosen as it inherently models the duration of the movement. It is necessary to determine a suitable window size for feature extraction and number of states for each model. An iterative search was performed to determine these parameters, with the window size evaluated in the range of 10-30 ms, and the number of states ranging from 1-20. The models were trained using embedded re-estimation. The window size was selected at 20 ms, and the number of states for the movement models are specified in Table I.

Movement	# States	# Samples	# Correct	# Insertion
Crouch	11	102	102	1
Sit	6	34	34	0
Jump	8	102	102	0
Step	8	27	27	0
Rise to Stand	8	34	34	0
Drop-land	7	102	102	0
Static	1-2 (ergodic)	315	312	0

TABLE I
EVALUATION OF RECOGNITION SYSTEM.

Data not previously used for training the models is used to show the performance of the classifier. Several minutes of new GRF data were recorded and the movements were also videotaped. Using the video recording the sequence of movements was manually labelled. The GRF was put through the classifier which generated a list of recognised movements. This list is compared to the manual observation. Table I shows the number of test sequences for each movement, together with the number of correct identifications and insertion errors. An insertion error occurs when a movement is incorrectly recognised as being present when it was not physically performed. This experiment is somewhat limited in the number of movement samples and the performance should be considered an upper-bound.

The results show that the classifier can correctly recognise a sequence of movements and provides no information regarding the alignment of the movement boundaries. Accurate boundary marking is crucial for characterisation. Static detection marks the start and end times of a movement period and consecutive movements may occur (e.g. crouch into jump) in this segment. The start and end times of these movements are determined by the classifier. To show the alignment accuracy a new GRF time-series composed of several movements was classified. Figure 3 shows this GRF signal on which the movement boundaries are indicated and the movements labelled. It should be noted that the movements were segmented by the HMMs alone and static detection was not used.

V. MOVEMENT CHARACTERISATION AND STORAGE

The result of classifying a GRF stream is a sequence of movements. The GRF can then be analysed, with knowledge of the movement type, to express features of the movement. Additional parameters can be determined when considering the *relationship* between movements and characterisation may be performed several times for a movement. This section describes the Movement Repository and Characterisation components. The Movement Repository is both a factory for creating movement objects and offers a persistent store for movements. The Characterisation component implements a technique for identifying the relationship between movements, necessary for inter-characterisation. A case-study is presented to illustrate the concept of movement characterisation.

A. Movement Repository

Processes in the distributed computing environment require access to the attributes or characteristics of movements, which are represented as CORBA objects defined in IDL (interface definition language). IDL is platform neutral and has mappings to all major programming languages. Movements are represented as objects, in preference to structures, as they contain both data and methods, and inheritance in IDL is only possible for interfaces, not structures. An IDL interface is equivalent to a class declaration. There are common attributes for every movement and it is desirable to use a movement base class from which all subsequent movements are derived. This approach also allows flexibility in defining new movement types which may subclass other movements, for example a horizontal leap or vertical jump may inherit from the jump class. The IDL interface of the movement base class is presented in Fig. 4. Within the body of an interface fields are declared as attributes, with a type and name. Defined types, i.e. types declared by the programmer, and basic types are supported as attributes.

```
interface Movement
{
  readonly attribute Timestamp time;
  readonly attribute unsigned long duration;
  readonly attribute Stabilogram stblgrm;
  readonly attribute Location loc;
  readonly attribute HTKRecog::Result recogResult;
  void setNextMovement (in Movement m);
  void setPreviousMovement (in Movement m);
};

interface Step : Movement
{
  readonly attribute Phase heelContact;
  // ...
};
```

Fig. 4. IDL definition for Movement and Step (partial) interfaces showing inheritance.

Mean	Standard Deviation	Slope	1^{st} Diff.	2^{nd} Diff.
$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$	$s = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \overline{x})^2}{n}}$	$m = \frac{y_n - y_1}{n}$	$delta \times 3$	$acceleration \times 3$

Fig. 2. Feature vector.

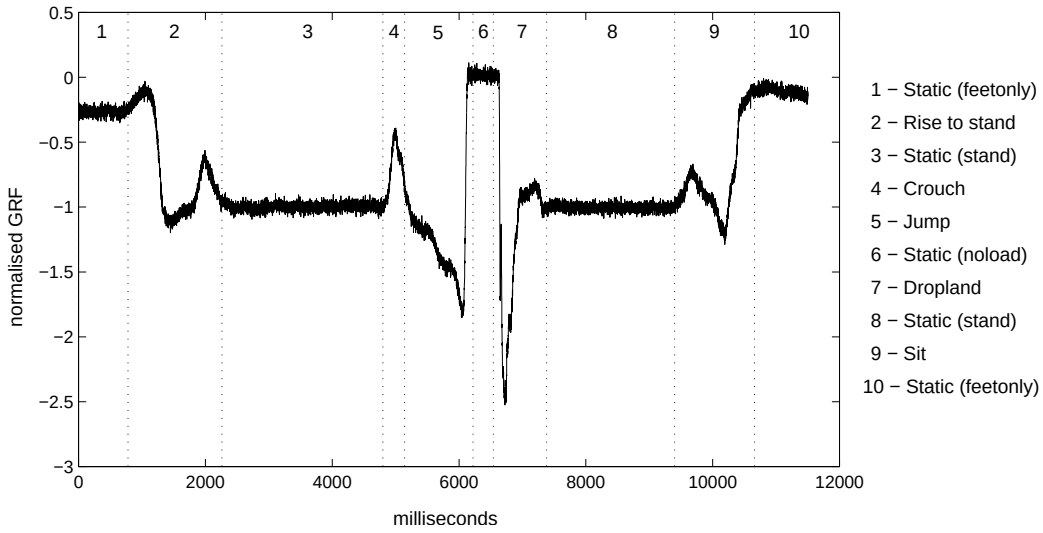

Fig. 3. Labelled recognition showing movement boundaries.

The Movement Repository provides an interface to allow distributed processes to create a movement object. An object is created by invoking createMovement(...) on the repository, passing the recognition result and GRF time-series, which returns the object reference (IOR) for the newly created movement. The method examines the recognition result and instantiates a movement object of the correct type. The GRF is used to intra-characterise the movement in the constructor. The Movement Repository also provides persistent storage of movement objects, and it is necessary to uniquely identify each object using the time and location of the movement. This URI functions as the primary key for the tables in the movement database, and also forms the ObjectID part of the IOR to identify the object in the portable object adaptor (POA). This is necessary for implementing the default servants.

In this implementation movement objects do not have their own servants and instead default servants provide access to the objects. Default servants can be used where objects are not required to have their own servants. The scalability aspects of this cannot be overemphasised as they provide the ability to store a very large number of objects in a fixed about of memory. There is one default servant for each class of movement, and one POA for each default servant. The state of the object, e.g. attributes, are stored in a database, using the URI as the primary key. The default servants use the PortableServer::Current interface to extract the URI for the movement from the ObjectID, thus identifying it. The servant can then identify the required record in the database and access the attributes of the movement. For example a client wants to retrieve information on the heel-contact phase of a step and requests the attribute from an object reference. This request is in fact processed by the default servant that represents and services requests for all step objects. The default servant extracts the URI for the movement from the ObjectID. It then uses a wrapper class to generate the SQL command required to retrieve the record from the database and also to convert the data stored in the field to the appropriate CORBA type, including defined types. The result is returned to the caller. This operation is transparent to the client which simply invokes operations on an object reference.

B. Implementation of the Movement Repository

The Movement Repository is implemented in C++ using omniORB4 and a mySQL database for the storage of objects. omniORB is a high performance ORB and version 4 has support for the PortableServer::Current interface, necessary for implementing default servants. omniORB also provides a mechanism for serialising defined types allowing them to be stored in binary fields of a database. mySQL was chosen as the database primarily as there was an existing installation at the laboratory, and it is freely available, although any database capable of storing binary data would suffice.

Each movement is specified as an interface in an IDL module. This specification is parsed to generate the code (SQL and C++) required for the specification of tables, creation and updating of table records, and implementation of the default servant attribute accessor methods.

A database table definition is required for each interface. The columns of the table correspond to the attributes of the interface, and the rows correspond to the movement objects. The columns have suitable mySQL types to store the attribute

types specified in the interface. IDL basic types are mapped to mySQL basic types, and defined types, i.e. those declared and defined in that or a related module, are serialised and mapped to BLOB (binary large object) types in mySQL. The BLOB type can hold a variable amount of binary data. The defined types are serialised using cdrMemoryStream, proprietary to omniORB [1]. Each table also has a URI column which is used to identify the movement and is the primary key. Inheritance is also supported and is described using an example. If Step inherits from Movement, then for each row in Step table, there is a corresponding row in Movement table, with the same value in the URI field. The IDL module is parsed to generate the table definitions in SQL.

The IDL module is again parsed to generate the appropriate SQL statements to insert, retrieve and update records in the database. Helper functions are used to convert defined types to and from serialised form, and to apply database specific escape characters for binary data storage. These helper functions also convert from mySQL types to the appropriate CORBA types.

A default servant is specified for each movement interface. In this system the primary role of the default servants is to provide access to the attributes of the movements stored in the database. Again the IDL module is parsed to implement the accessor functions for each attribute in the interface. The functions use a helper class that allows them to indirectly access records of the database and have the helper function query the database and convert from the mySQL type to the appropriate CORBA type. The default servant simply provides a pointer to the database, table name (from the IDL interface name), URI for the movement, and the attribute name. Figure 5 shows a piece of C++ code used to retrieve the heel-contact phase from a step object. This code is automatically generated by parsing the Step interface defined in IDL.

C. Inter-movement Characterisation

Analysing a movement in isolation provides some characteristics, but there is a wealth of information contained *between* movements. The role of the Movement Characterisation component is to identify relationships between movements to support inter-movement characterisation. Movement objects can be informed of their next and previous movement relationships and use this information to update their attributes accordingly.

It is non-trivial to identify movement relationships when the system operates in stand-alone mode. The identity of the person that generated the movement is unknown. Some systems can identify people using the GRF during a step [3], [4], [16], but this is restricted to one movement, under certain circumstances and requires many training patterns. It is not believed that all movements can be used to identify a person, and the system would require a significant number of training patterns for each movement for each person. A high-resolution location sensing technology such as the Active Bat [17] could be used to identify who generated the force by considering the respective location of the person and the force. The system could use this information when available, but a solution to the problem of identifying related movements in stand-alone mode is sought. The Movement Characterisation process uses a set of predictions, generated for each movement, to identify a relationship between movements.

The Movement Characterisation service consumes movement events. These events consist of the recognition result, presented in structured fields, and an object reference to the movement. The service stores object references in MovementContainer objects that hold a list of related movements and the prediction set. These predictions are the set of feasible proceeding movements in terms of time, type, position, direction and movement specific attributes. When a new movement event arrives it is matched to the prediction set in each container to see if a feasible relationship exists. If the new movement is related then it is added to the corresponding container and the movements already in the container are subsequently re-characterised. An event is sent to notify that the movement object has been updated and a new set of predictions are formed. These predictions can also provide feedback to the movement recognition and GRF extraction components. The containers self-destroy when no more feasible movements are possible, e.g. when the maximum time to the next feasible movement is exceeded.

D. Case-study: Jumping

A case-study is presented to illustrate the GRF for a set of related movements and to indicate some of the characteristics that can be determined during the jump movement. The interested reader is referred to [18] for further information on characterising the jump and other movements.

The three movements of crouch, jump and drop-land are combined in this description as they have a logical association when considering a jumping movement. Notably, a person must lower their body by hip and knee flexion to generate enough potential to leave the ground with sufficient velocity to reach a height (greater than ground height), despite the acceleration due to gravity. A drop-landing is common after a jump.

The GRF for a jumping movement was recorded and Fig. 6 presents the magnitude and position traces of this recording. The sequence of movements includes crouch, jump and drop-land. Temporal parameters are computed from the magnitude of the GRF and the position component of the GRF yields spatial parameters. The magnitude is used to determine the phases of the jump movement, i.e. thrust, take-off and flight. The first two phases can be fully specified during intra-characterisation, with the duration of the flight phase remaining unspecified or estimated until the corresponding drop-land has been observed and related. The height of the jump can be determined by the duration of the flight phase, or by the amount of force exerted during the thrust phase. The distance of the jump can be computed as the distance between the start position of the jump and the end position of the drop-land. A line of progression is defined between these two points and indicates the direction of the movement. There are two classes of jump: vertical jump

[1] c.f. MemBufStream in previous versions of omniORB.

```
inline virtual CHAPS::Phase heelContact()
{
  return getUserField<CHAPS::Phase> (db, ''Step'', getMovementURI(), ''heelContact'');
}
```

Fig. 5. Snippet of auto-generated C++ code taken from the Step default servant showing the call to retrieve an attribute from the database.

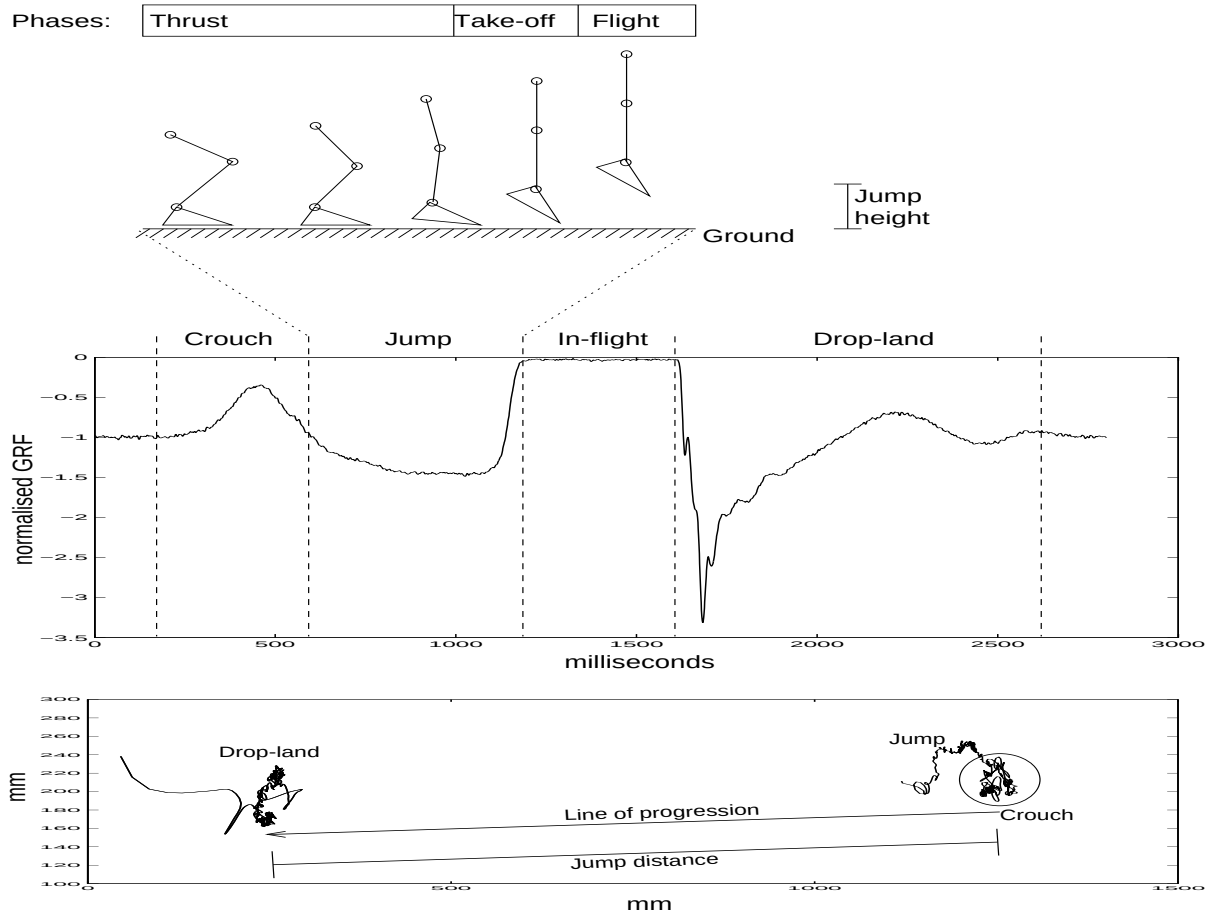

Fig. 6. Jump characterisation: the jump phases and magnitude and position of the forces during the crouch, jump and drop-land movements.

or horizontal leap. This jump movement is determined to be a horizontal leap by considering the jump distance.

VI. Conclusion

Movement awareness is a new modality to the sentient computing model and enables the development of context-aware applications that respond in real-time to human movements. This paper presents an overview of the distributed components and their interactions that comprises the movement awareness system. The concept of movement recognition and characterisation is introduced and their operation and implementation is described. The movement classifier is shown to correctly identify a set of whole body human movements. The development of the characterisation process resulted in a system of default servants and persistent storage of CORBA objects. This approach provides a scalable solution to manage large numbers of movement objects. The majority of the default servant and repository code is auto-generated from IDL definitions and the same process can be applied to other objects with specified IDL interfaces. It is advantageous to expand the set of movements that can be recognised and characterised. Specifically the recognition and characterisation of the complex related movements involved in the lifting task presents a useful measure of assessing the technique and risk of lifting in manual materials handling, the cause of the majority of work related back injuries.

Acknowledgement

The author thanks Andy Hopper, Rupert Curwen, AT&T Laboratories Cambridge, the Cambridge European Trust and Selwyn College.

References

[1] A. Hopper, "The Clifford Paterson Lecture, 1999 Sentient Computing," *Phil. Trans. R. Soc. Lond.*, vol. A (2000), no. 358, pp. 2349–2358, 2000.

[2] "Kistler force plate," http://www.kistler.com/.

[3] M. Addlesee *et al.*, "The ORL Active Floor," *IEEE Personal Communications*, vol. 4, no. 5, pp. 35–41, October 1997.

[4] R. J. Orr and G. D. Abowd, "The Smart Floor: A Mechanism for Natural User Identification and Tracking," in *Proceedings of the 2000 Conference on Human Factors in Computing Systems*, The Hague, Netherlands, 1-6 April 2000.

[5] A. Schmidt *et al.*, "Context Acquisition Based on Load Sensing," in *UbiComp 2002*, G. Borriello and L. E. Holmquist, Eds., vol. LNCS 2498. Springer-Verlag, 2002, pp. 333–350.

[6] "Lumetila," http://www.vtt.fi/tte/projects/lumetila.

[7] G. Bradski and J. Davis, "Motion Segmentation and Pose Recognition with Motion History Gradients," in *WACV*, 2000, pp. 238–244.

[8] C. Randall and H. Muller, "Context awareness by analysing accelerometer data," in *The Fourth International Symposium on Wearable Computers*, B. MacIntyre and B. Iannucci, Eds. IEEE Computer Society, October 2000, pp. 175–176.

[9] R. DeVaul and S. Dunn, "Real-Time Motion Classification for Wearable Computing Applications," December 2001, MIT Technical Report.

[10] "SPIRIT," http://www.uk.research.att.com/spirit/.

[11] B. Brummit *et al.*, "EasyLiving: Technologies for Intelligent Environments," in *Handheld and Ubiquitous Computing*, September 2000.

[12] S. Young *et al.*, *The HTK Book (for HTK Version 3.0)*. Microsoft Corporation, 2000, http://htk.eng.cam.ac.uk/.

[13] O. Management Group, "Notification Service Specification," June 2000.

[14] R. Headon and R. Curwen, "Movement awareness for ubiquitous game control," *Personal Ub Comp*, vol. 6, no. 5-6, pp. 407–415, 2002.

[15] L. R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, February 1989.

[16] P. C. Cattin, D. Zlatnik, and R. Borer, "Biometric System using Human Gait," in *Mechatronics and Machine Vision in Practice (M2VIP)*, Hong-Kong, August 2001.

[17] A. Ward, A. Jones, and A. Hopper, "A New Location Technique for the Active Office," *IEEE Personal Communications*, pp. 42–47, October 1997.

[18] http://www-lce.eng.cam.ac.uk/˜rph25/MovementCharacterisation/.