CONFIDENCE

UBIQUITOUS CARE SYSTEM TO SUPPORT INDEPENDENT LIVING

Small or medium-scale focused research project

DELIVERABLE **D3.1-2**

Report on Indoor Reconstruction and Interpretation Techniques

Contract number:	214986
Project acronym:	CONFIDENCE
Project title:	UBIQUITOUS CARE SYSTEM TO SUPPORT INDEPENDENT LIVING

Deliverable number:	**D3.1**
Nature:	Report
Dissemination level:	PU – Public
Report date:	2008-07-31

Author(s):	Mitja Luštrek, Bojan Nemec, Boštjan Kaluža, Rok Piltaver, Matjaž Gams, Igone Vélez, Kristina Larsson, Narciso González
Partners contributed:	JSI, CEIT, UMP, JYU
Contact:	Jožef Stefan Institute, Jamova cesta 39, 1000 Ljubljana, Slovenia +386 1 4773380, mitja.lustrek@ijs.si

The CONFIDENCE project was funded by the European Commission under the 7th Framework Programme (FP7) –theme 3 "Information & Communication Technologies".

ICT 1-7.1 ICT & Ageing

CONFIDENCE

UBIQUITOUS CARE SYSTEM TO SUPPORT INDEPENDENT LIVING

Small or medium-scale focused research project

Reviewed by:	**Approved by:**	
Andrés García Alonso External Reviewer CEIT	Igone Vélez CONFIDENCE Project Coordinator CEIT	Pablo de Lecea Head of CONFIDENCE Project Management Office CEIT

INDEX

1 INTRODUCTION

The main objective of Confidence is the **development and integration of innovative technologies** to develop a care system for the **detection of abnormal events** (such as falls) or **unexpected behaviors** that may be related to a health problem in **elderly people**. As explained in the DoW, the Confidence system has been divided into four subsystems:

- *Localization subsystem*: This subsystem relies on radio technology and performs two tasks: identification and localization of the tags. It provides the position of each tag with an accuracy of a few centimeters. Two localization subsystems will be built; the first one will work indoors and will be placed in the base-station. The second one will work outdoors and will be located in the portable device.
- *Reconstruction subsystem*: This second subsystem receives the estimates of the positions of the tags and generates a model of the user and the environment. Two reconstruction subsystems will be developed; the first one will be placed in the base-station and will work indoors. The second will be located in the portable device and will function outdoors.
- *Interpretation subsystem*: The third subsystem interprets this data to make a decision about the situation. This subsystem will be provided with “intelligence”, so that it can learn from the user’s habits and help to detect early symptoms of illness, such as Parkinson, spine cancer etc. An indoor interpretation subsystem will be placed in the base-station and an outdoor interpretation subsystem will be located in the portable device.
- *System interface subsystem*: The fourth subsystem is the system interface that is responsible for the user interface, system setup and alarm handling. The system interface of both the base-station and the portable device will be developed. The system interface subsystem of the portable device will be able to work indoors and outdoors. However, the system interface subsystem of the base-station will only work indoors.

This report will explain the different techniques that can be used for the indoor reconstruction and interpretation subsystem. Additionally, this document will propose the techniques for the indoor reconstruction and interpretation subsystem and will explain the reasons why these techniques are proposed.

2 RELATED WORK

Fall detection is an area where quite a lot of research has been done. The most common approach is to use tri-axial accelerometers with threshold-based algorithms, examples being **Bourke et al.** [5] and **Kangas et al.** [13]. Such algorithms detect a fall simply when the measured accelerations reach a threshold value. There are even several sensors with build-in hardware fall detection [1][8][21].

Willis [26] used pressure transducers besides accelerometers. He developed a more complex fall detection algorithm based on dynamic belief network models, which can be used to model and produce conclusions about the state of complex temporal environments.

Zhang et al. [30][31] embedded the accelerometer in a cell phone. They used machine learning to detect falls. The more obvious falls were recognized with Support Vector Machines. For the dubious cases, they used Kernel Fisher Discriminant to reduce the dimensionality of the problem and *k*-Nearest Neighbor algorithm to classify them as falls or non-falls.

Researchers using accelerometers give a lot of attention to the optimal position of the sensor on the body [5][13]. A head-worn accelerometer provides excellent impact detection sensitivity, but it is problematic regarding usability and acceptance. A better option is waist-worn accelerometer. The wrist does not appear to be an optimal position for fall detection. Some researchers made a step further and used accelerometers for trying to recognize the impact and the posture after the fall [14].

In Confidence we might not be able to derive accelerations from the velocities computed from the changes in tag locations. The success of this approach is questionable due to the relatively low expected localization precision and sampling rate (low at least for the computation of the second derivatives of locations). But if the accuracy of the computed accelerations proves sufficient, we can easily add threshold-based rules. We can also include accelerations among the attributes for machine learning, which we are going to use anyway. We do not plan to add dedicated algorithms to analyze accelerations, since the goal of Confidence is to investigate an alternative to accelerometer-based fall detection.

Bourke and Lyons [4] used a bi-axial gyroscope sensor mounted on the torso to measure the pitch and roll angular velocities. They measured the peaks in angular velocity, angular acceleration and torso angle change. They introduced a threshold-based algorithm to distinguish between falls and activities of daily living, which proved 100 % successful in the test setting. This method – like the accelerometer-based methods ones – may be used in Confidence if the localization accuracy is sufficient.

Wu [28] studied horizontal and vertical velocities of different parts of the torso during falls and normal activities (walking, sitting down and standing up from a chair, descending stairs, picking up an object from the floor, transferring in and out of a tub and lying down on the bed). He found that both velocities increase dramatically during the descending phase of the fall. His study provides the velocity parameters for fall detection, which could be useful for Confidence – again assuming that the sensor precision and sampling rate prove adequate for sufficiently fine velocity measurements.

Fu et al. [9] detected falls with an asynchronous temporal contrast vision sensor, which extracts changing pixels in the image from the background and reports temporal contrast. They computed centroid events as the average of the motion events reported by the sensor and measured their vertical velocity to detect falls. They were able to distinguish between falls and normal behavior (walking, crouching down and sitting down). This technique may be suitable for Confidence, since it likely does not require very high sensor precision. However, we will probably achieve a similar effect by including various tag speeds (averaged over different groups of tags) among the attributes for machine learning.

Many researchers investigated fall detection using video surveillance. They extracted features from the image signal and used one of the abovementioned methods. Since such feature extraction is not applicable to Confidence, we do not discuss it here.

3 ARCHITECTURE OF THE INDOOR RECONSTRUCTION AND INTERPRETATION SUBSYSTEM

Figure 1 shows the architecture of the indoor reconstruction and interpretation subsystem. This subsystem will be placed in the base station and will have the interfaces described in D1.2 System Model. The figure identifies the different software modules to be developed in WP3.

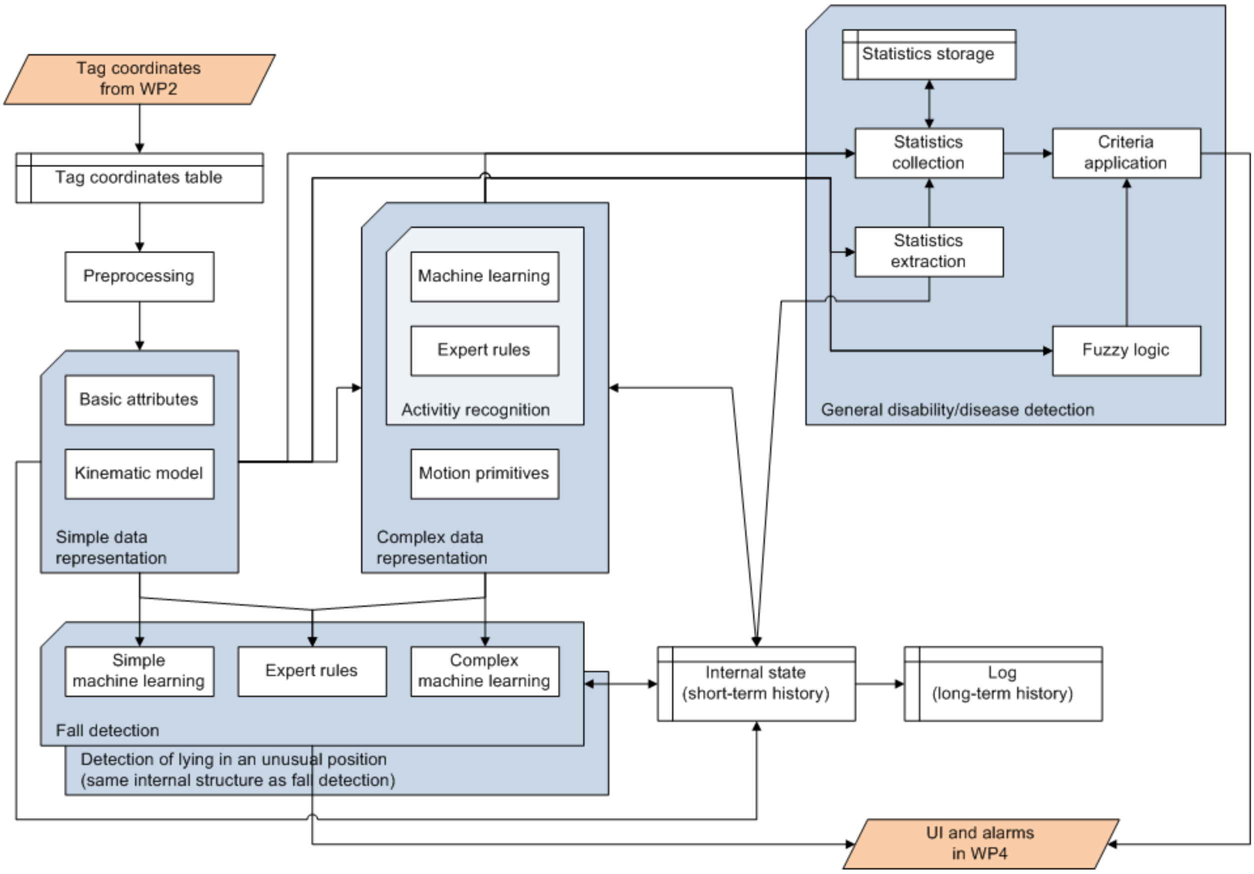

Figure 1. Reconstruction and interpretation subsystem architecture.

Tag coordinates table stores the tag coordinates as they come from WP2. When a whole snapshot of the body tags is available (i.e., the current coordinates of all the body tags), it is first **preprocessed**, which mainly means that the noise is filtered. Then the raw data is represented in various ways useful for further analysis. **Simple data representations** describe each snapshot, whereas **complex data representations** describe actions spanning multiple snapshots.

Fall detection. Simple machine learning classifies simple data representations into those describing falls and non-falls. Complex machine learning does the same with complex data representations. Expert rules use all data representations. The final decision on whether a fall has occurred and whether an alarm needs to be raised combines the outputs of all three approaches and is forwarded to WP4.

Detection of lying in an unusual position works the same way as fall detection.

General disability/disease detection is based on statistics collected over different periods of time. Some of them are derived directly from the data representations and some need dedicated extraction. A special statistics-gathering module is Fuzzy logic. Finally criteria are applied to the statistics to determine whether their values warrant raising a warning. This decision is forwarded to WP4.

All the data regarding the current state of the user and his/her environment are stored in the **internal state**, from which they can be retrieved by all methods that need them. After a time they are moved to the **log**.

4 PROPOSED METHODS FOR THE INDOOR RECONSTRUCTION AND INTERPRETATION SUBSYSTEM

4.1 PREPROCESSING

The location information from the Confidence hardware will be noisy. Since it is expected to be sampled with the frequency of 10 Hz, the noise of a tag location can be reduced by taking into account the previous locations of the same tag, unless the movement is very fast. This is particularly important for the computation of velocities and accelerations via numerical differentiation.

Low-pass filtering can be used for smoothing, but it introduces time delays in the filtered signals. Since the expected sampling frequency is not very high, low pass filtering significantly corrupts the measured signals and limits the results of the motion estimation algorithms. An appropriate way to solve this problem is to use Kalman filtering, which can optimally estimate the non-measurable states of a dynamic system [17].

Kalman filter addresses the problem of the optimal estimation of the state x of a discrete-time process, described by the linear stochastic difference equation

$$x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1} \tag{1}$$

with a measurement

$$z_k = Hx_k + v_k \tag{2}$$

Here w and v represents the process and the measurement noise with the covariances Q and R respectively. Then the optimal estimation $\hat{x}$ of the process state x can be obtained using the Kalman filter in the form

$$\hat{x}_k = \hat{x}_k^- + K(z_k - H\hat{x}_k^-) \tag{3}$$

where $\hat{x}_k^-$ is the predicted value of the state calculated using Equation (1) and K is the Kalman gain governed by the equation

$$K_k = \frac{P_k^- H^T}{HP_k^- H^T + R} \tag{4}$$

Kalman filtering is based on a-priori and a-posteriori estimation of the error covariance P_k^- and P_k , which are computed sequentially using the equations

$$\begin{aligned} P_k^- &= AP_{k-1}A^T + Q \\ P_k &= (I - K_k H)P_k^- \end{aligned} \tag{5}$$

In our case we have a process that describes the movement of tags in 3D space described by its position and velocity, but only the position is measurable. We want to estimate also the velocity of the tags. Therefore our state can be described by

$$x = \begin{bmatrix} p_x \\ p_y \\ p_z \\ v_x \\ v_y \\ v_z \end{bmatrix} \tag{6}$$

where p and v denote the position and velocity respectively and the indices x, y, z denote the Cartesian coordinates. The process matrix A and the output matrix H, which describe the movement of a particle in 3D space, can be described in the state space as

$$A = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \; H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{7}$$

Noise covariance matrices are defined experimentally. Since Confidence hardware is not available yet, we have so far only experimented with Smart infrared motion capture equipment (more on this in Section 5). For Smart system, the best results are obtained with the selections of $Q = 0.01I$ and $R = I$, where I is the square identity matrix of dimension 6. We have verified the selections with the difference between the measured and the smoothed trajectory, which we presume should be smaller than the standard error in each point and each coordinate. We will proceed the same way for the Confidence hardware, once the hardware or at least an estimation of its noise is available.

We will most likely also need the estimation of the acceleration of the tags. Reasonable estimates may be obtained using the numerical differentiation of the estimated velocities. In testing the best results were obtained with Rauch-Tung-Striebel (RTS) algorithm [24], which performs fixed-interval offline smoothing of the estimated signals. Note that discrete state space model presumes normalized time, therefore the velocity and the acceleration signals have to be multiplied by $t_{sampling}^{-1}$ and $t_{sampling}^{-2}$ respectively.

The initial Kalman filter implementation uses the Kalman filter toolbox for Matlab [19].

4.2 SIMPLE DATA REPRESENTATION

4.2.1 Basic attributes

These attributes are the coordinates of the body tags and the quantities derived directly from them. *B* is the number of body tags. *N* is the number of sets of coordinates in the time interval under observation. Three coordinate systems are considered:

- Reference coordinate system is fixed relative to the environment. It captures properties such as the location in the apartment, whether the user is lying on the floor or on the bed etc.
- Body coordinate system is fixed relative to the body of the user. Its purpose is to unify the instances of recurring movements that only differ in the location and direction of the user.

- Body coordinate system belonging to the first set of coordinates in the interval under observation. Its purpose is also to unify the instances of recurring movements that only differ by the location and direction of the user. However, unlike in the previous coordinate system, the changes in position in the interval are also captured (relative to the location and direction at the beginning of the interval).

The coordinate systems are described in more detail at the end of the section.

Velocity can be expressed with its x, y, and z components or with its absolute value and the two angles describing its direction. The representations are equivalent and both of them are listed, but we lean towards the second one, so the first one is written in gray. This notation is used everywhere in the deliverable: the options we are unlikely to choose, even though they are legitimate, are gray.

Attributes in the reference coordinate system:

- ($x^t_{i\mathrm{R}}$, $y^t_{i\mathrm{R}}$, $z^t_{i\mathrm{R}}$) … coordinates of the body tag i at time t; i = 1 ... B, t = 1 ... N
- ($v^t_{\mathrm{x}i\mathrm{R}}$, $v^t_{\mathrm{y}i\mathrm{R}}$, $v^t_{\mathrm{z}i\mathrm{R}}$) … x, y and z velocity of a body tag
- $v^t_{i\mathrm{R}}$ … absolute velocity of a body tag
- ($\varphi^t_{i\mathrm{R}}$, $\theta^t_{i\mathrm{R}}$) … angles of movement of a body tag with respect to the z and x axes
- (x^t_{OR}, y^t_{OR}, z^t_{OR}) … coordinates of the origin of the body coordinate system
- (Φ^t_{OR}, Θ^t_{OR}) … orientation of the x axis of the body coordinate system with respect to the z and x axes
- (v^t_{xOR}, v^t_{yOR}, v^t_{zOR}) … x, y and z velocity of the origin of the body coordinate system
- v^t_{OR} … absolute velocity of the origin of the body coordinate system
- (φ^t_{OR}, θ^t_{OR}) … angles of movement the origin of the body coordinate system with respect to the z and x axes
- (x_{OfR}, y_{OfR}, z_{OfR}) … coordinates of the origin of the body coordinate system belonging to the first set of coordinates in the interval
- (Φ_{OfR}, Θ_{OfR}) … orientation of the x axis of the body coordinate system with respect to the z and x axes belonging to the first set of coordinates in the interval.

Attributes in the body coordinate system:

- ($x^t_{i\mathrm{B}}$, $y^t_{i\mathrm{B}}$, $z^t_{i\mathrm{B}}$) … coordinates of a body tag
- ($v^t_{\mathrm{x}i\mathrm{B}}$, $v^t_{\mathrm{y}i\mathrm{B}}$, $v^t_{\mathrm{z}i\mathrm{B}}$) … x, y and z velocity of a body tag
- $v^t_{i\mathrm{B}}$ … absolute velocity of a body tag
- ($\varphi^t_{i\mathrm{B}}$, $\theta^t_{i\mathrm{B}}$) … angles of movement of a body tag with respect to the z and x axes.

Attributes in the body coordinate system belonging to the first set of coordinates in the interval:

- ($x^t_{i\mathrm{Bf}}$, $y^t_{i\mathrm{Bf}}$, $z^t_{i\mathrm{Bf}}$) … coordinates of a body tag
- ($v^t_{\mathrm{x}i\mathrm{Bf}}$, $v^t_{\mathrm{y}i\mathrm{Bf}}$, $v^t_{\mathrm{z}i\mathrm{Bf}}$) … x, y and z velocity of a body tag
- $v^t_{i\mathrm{Bf}} = v^t_{i\mathrm{R}}$ … absolute velocity of a body tag
- ($\varphi^t_{i\mathrm{Bf}}$, $\theta^t_{i\mathrm{Bf}}$) … angles of movement of a body tag with respect to the z and x axes
- (x^t_{OBf}, y^t_{OBf}, z^t_{OBf}) … coordinates of the origin of the body coordinate system
- (Φ^t_{OBf}, Θ^t_{OBf}) … orientation of the x axis of the body coordinate system with respect to the z and x axes
- (v^t_{xOBf}, v^t_{yOBf}, v^t_{zOBf}) … x, y and z velocity of the origin of the body coordinate system
- $v^t_{\mathrm{OBf}} = v^t_{\mathrm{OR}}$ … absolute velocity of the origin of the body coordinate system
- (φ^t_{OBf}, θ^t_{OBf}) … angles of movement of the origin of the body coordinate system with respect to the z and x axes.

Accelerations can also be added to the list of attributes once we have an estimate of the noise of the Confidence hardware and can judge whether we can compute them with a reasonable accuracy.

Reference coordinate system

This is the coordinate system used by the localization subsystem and is fixed relative to the environment. It is a right-handed coordinate system with the z axis pointing upwards.

Body coordinate system

This is the coordinate system fixed relative to the body (specifically the torso) shown in Figure 2. Its origin is located at the mid-point (*O*) of the line connecting both hip tags (H_L and H_R for the left and right hip respectively). This line also defines the y axis, which points towards the left hip. The z axis is perpendicular to the y axis, touches the line connecting both shoulder tags (S_L and S_R for the left and right shoulder respectively) at point S_z and points upwards. The x axis is perpendicular to the y and z axes and points forwards.

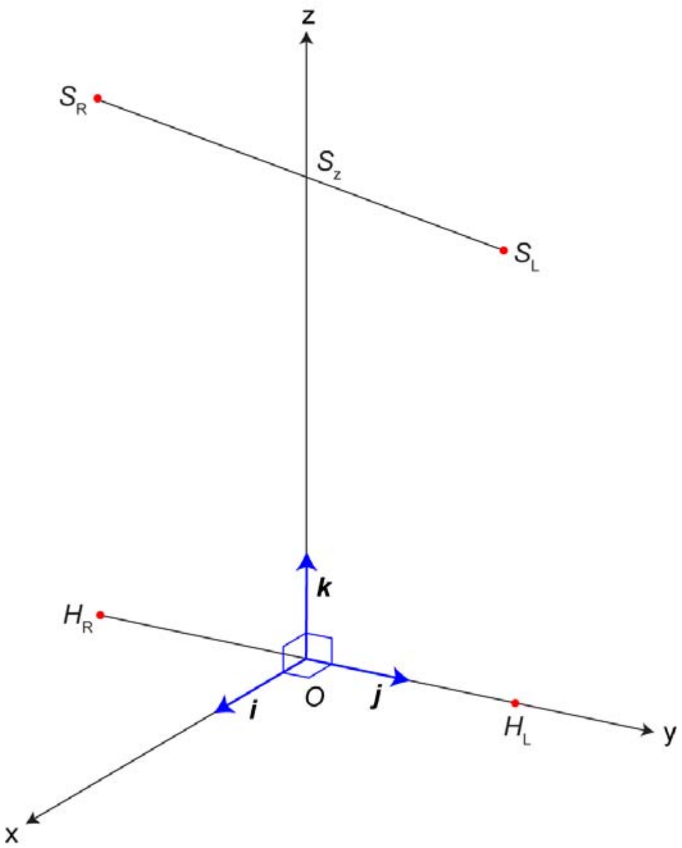

Figure 2. Body coordinate system.

In order to work with the body coordinate system, we need to calculate its origin O and basis (***i***, ***j***, ***k***) in the reference coordinate system. Note that bold type is used to denote vectors and ***x*** denotes a vector from the origin to the point *X*. The equation for the origin is straightforward:

$$\boldsymbol{o} = \frac{\boldsymbol{h}_L + \boldsymbol{h}_R}{2} \tag{8}$$

So is the equation for ***j***:

$$\boldsymbol{j} = \frac{\boldsymbol{h}_L - \boldsymbol{o}}{|\boldsymbol{h}_L - \boldsymbol{o}|} \tag{9}$$

To obtain ***k***, $\boldsymbol{s}_z$ must be calculated first:

$$\begin{aligned} \boldsymbol{s}_z &= \boldsymbol{s}_R + a(\boldsymbol{s}_L - \boldsymbol{s}_R) \\ (\boldsymbol{s}_z - \boldsymbol{o})(\boldsymbol{h}_L - \boldsymbol{h}_R) &= 0 \end{aligned} \tag{10}$$

$$a = \frac{(\boldsymbol{s}_{\mathrm{R}} - \boldsymbol{o})(\boldsymbol{h}_{\mathrm{L}} - \boldsymbol{h}_{\mathrm{R}})}{(\boldsymbol{s}_{\mathrm{L}} - \boldsymbol{s}_{\mathrm{R}})(\boldsymbol{h}_{\mathrm{L}} - \boldsymbol{h}_{\mathrm{R}})}$$

Once we have $\boldsymbol{s}_{\mathrm{z}}$, we calculate $\boldsymbol{k}$ as follows:

$$\boldsymbol{k} = \frac{\boldsymbol{s}_{\mathrm{z}} - \boldsymbol{o}}{|\boldsymbol{s}_{\mathrm{z}} - \boldsymbol{o}|} \tag{11}$$

Finally, we obtain $\boldsymbol{i}$ as the cross product of $\boldsymbol{j}$ and $\boldsymbol{k}$:

$$\boldsymbol{i} = \boldsymbol{j} \times \boldsymbol{k} \tag{12}$$

We use transformation matrix $\boldsymbol{T}_{\mathrm{R}\to\mathrm{B}} = \boldsymbol{T}^{-1}_{\mathrm{B}\to\mathrm{R}}$ to transform coordinates in the reference coordinate system into the body coordinate system [2]. Vector $\boldsymbol{i}_{(\mathrm{B})}$ is the basis vector $\boldsymbol{i}$ belonging to the body coordinate system. Notation $\boldsymbol{i}_{(\mathrm{B})\mathrm{R}}$ indicates that the vector is expressed in the reference coordinate system. Notation for the other vectors is analogous.

$$\boldsymbol{T}_{\mathrm{R}\to\mathrm{B}} = \begin{bmatrix} x_{i(\mathrm{B})\mathrm{R}} & y_{i(\mathrm{B})\mathrm{R}} & z_{i(\mathrm{B})\mathrm{R}} & -\boldsymbol{o}_{(\mathrm{B})\mathrm{R}}\boldsymbol{i}_{(\mathrm{B})\mathrm{R}} \\ x_{j(\mathrm{B})\mathrm{R}} & y_{j(\mathrm{B})\mathrm{R}} & z_{j(\mathrm{B})\mathrm{R}} & -\boldsymbol{o}_{(\mathrm{B})\mathrm{R}}\boldsymbol{j}_{(\mathrm{B})\mathrm{R}} \\ x_{k(\mathrm{B})\mathrm{R}} & y_{k(\mathrm{B})\mathrm{R}} & z_{k(\mathrm{B})\mathrm{R}} & -\boldsymbol{o}_{(\mathrm{B})\mathrm{R}}\boldsymbol{k}_{(\mathrm{B})\mathrm{R}} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{13}$$

If the vector $\boldsymbol{p}_{\mathrm{R}} = (x, y, z, 1)$ represents a point in the reference coordinate system, the corresponding vector $\boldsymbol{p}_{\mathrm{B}}$ in the body coordinate system is calculated as follows:

$$\boldsymbol{p}_{\mathrm{B}} = \boldsymbol{T}_{\mathrm{R}\to\mathrm{B}}\boldsymbol{p}_{\mathrm{R}}^{\mathrm{T}} \tag{14}$$

Alternative body coordinate system

The alternative coordinate system shown in Figure 3 is also fixed relative to the body (specifically the hips), but it is using the vertical axis of the reference coordinate system. Its origin is located at the mid-point (O) of the line connecting both hip tags (H_{L} and H_{R} for the left and right hip respectively). The z axis is the vertical axis of the reference coordinate system. The y axis is perpendicular to the z axis, lies on the plane defined by the hip tags and a point on the z axis, and points towards the left hip. The x axis is perpendicular to the y and z axes and points forwards when the person is upright (in general it points in the direction of the cross product of vectors lying on the y and z axes).

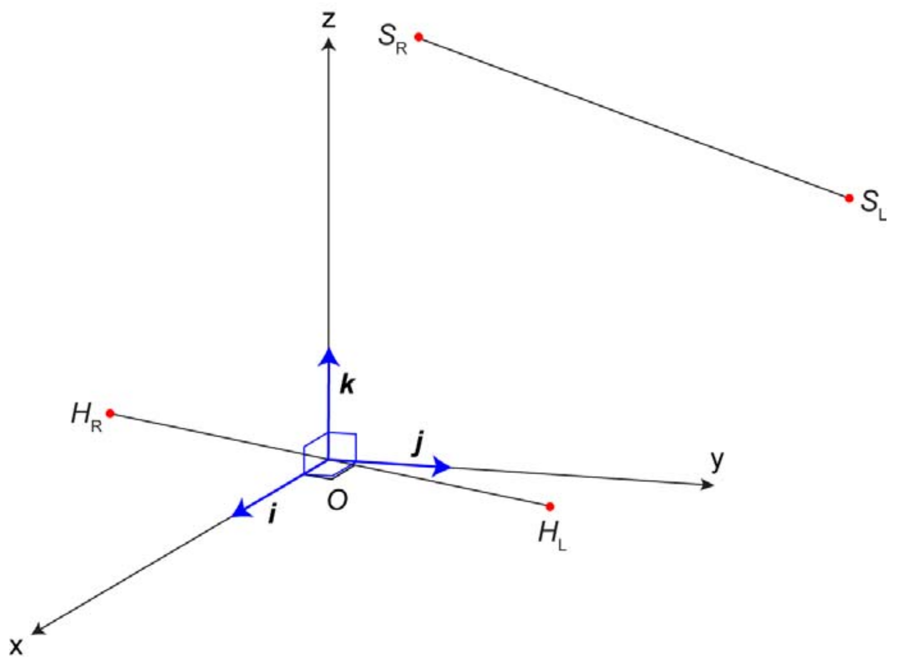

Figure 3. Alternative body coordinate system.

The origin O is calculated the same way as for the first body coordinate system:

$$o = \frac{h_L + h_R}{2} \qquad (15)$$

The basis vector $\boldsymbol{k}$ is the same as in the reference coordinate system: $\boldsymbol{k}$ = (0, 0, 1).

The basis vector $\boldsymbol{i}$ is perpendicular to $\boldsymbol{k}$ and to the vector from O to H_L:

$$i = \frac{k \times (h_L - o)}{|k \times (h_L - o)|} \qquad (16)$$

Finally, we obtain $\boldsymbol{j}$ as the cross product of $\boldsymbol{k}$ and $\boldsymbol{i}$:

$$j = k \times i \qquad (17)$$

The transformation into the alternative body coordinate system is done the same way as for the first one.

4.2.2 Kinematic model

The motion of the human body is detected using 12 tags as shown in Figure 4. The tags [1, 2, 3], [4, 5, 6], [7, 8, 9] and [10, 11, 12] are assigned to right arm, left arm, right leg and left leg respectively. The reference coordinate system is placed as shown.

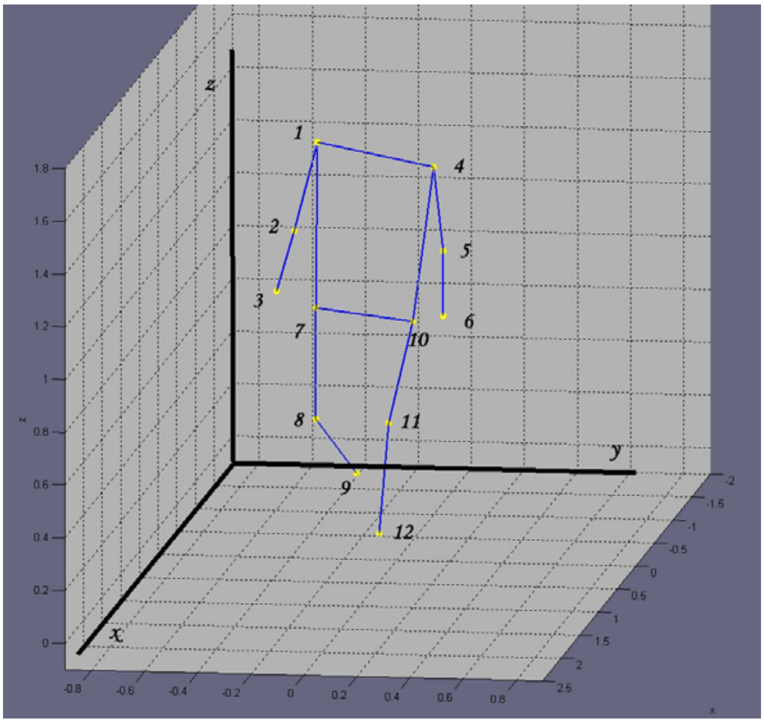

Figure 4. Kinematic model of the human body.

First, we have to estimate the orientation of the torso, which is determined with the markers [7, 10, 4, 1]. The line between the markers 7 and 10 is aligned with the y axis of the torso coordinate system. Assuming that the neck position can be described as the mid-point between the markers [4, 1], the torso coordinate system can be calculated using the following equations:

$$b = \frac{m_{10} - m_7}{|m_{10} - m_7|} \qquad (18)$$

$$c_1 = \frac{m_{4,1} - m_{10}}{\left|m_{4,1} - m_{10}\right|}$$

$$a = \frac{b \times c_1}{\left|b \times c_1\right|}$$

$$c = \frac{a \times b}{\left|a \times b\right|}$$

$$R = \begin{bmatrix} a & b & c \end{bmatrix}$$

The resulting matrix R is a 3 × 3 rotation matrix describing the local rotation in the reference coordinate system. We proceed with the estimation of the torso angles of arms and legs, respectively. The local coordinate systems were placed in the left and the right shoulder for the arms and the left and the right hip for the legs, respectively. For the right arm, we use the following equations to estimate the local coordinate system. In this case the z axis is aligned with the line [1, 2].

$$c = \frac{m_2 - m_1}{\left|m_2 - m_1\right|}$$

$$a_1 = \frac{m_3 - m_1}{\left|m_3 - m_1\right|}$$

$$b = \frac{c \times a_1 \eta}{\left|c \times a_1\right|} \tag{19}$$

$$a = \frac{b \times c}{\left|b \times c\right|}$$

$$R = \begin{bmatrix} a & b & c \end{bmatrix}$$

The scalar η has the value of 1 if the angle between the vectors [1, 2] and [2, 3] is smaller than 180° and –1 otherwise. The angle between the vectors [1, 2] and [2, 3] can be calculated using the well known formula

$$ab = |a||b|\cos(\varphi) \tag{20}$$

The torso, shoulder and hip rotation will be represented by quaternions. Unit quaternions provide a convenient mathematical notation for representing orientations and rotations of objects in three dimensions. Compared to Euler angles they are simpler to compose and avoid the problem of gimbal lock. Compared to rotation matrices they are more efficient and more numerically stable. Quaternions have found their way into applications in computer graphics, robotics, navigation, and orbital mechanics of satellites. One of the most used ways to represent the unit quaternion q is with the scalar-vector pair as

$$q = (w, v), \quad w = \cos(\frac{\theta}{2}), \quad v = u \sin(\frac{\theta}{2}) = (x, y, z) \tag{21}$$

where u is a unit vector and θ is the rotation around the vector u.

Conversion between a unit quaternion and a rotation matrix can be specified as

$$R = \begin{bmatrix} 1 - 2y^2 - 2z^2 & 2xy + 2wz & 2xy - 2wy \\ 2xy - 2wz & 1 - 2x^2 - 2z^2 & 2yz - 2wx \\ 2xz + 2wy & 2yz - 2wx & 1 - 2x^2 - 2y^2 \end{bmatrix} \quad (22)$$

In summary, the kinematic model gives us the following attributes:

- the angles of the left and right elbow at time t α^t_{EL} and α^t_{ER}
- the angles of the left and right knee α^t_{KL} and α^t_{KR}
- the angles of the left and right shoulder (represented by quaternions) q^t_{SL} and q^t_{SR}
- the angles of the left and right hip (also represented by quaternions) q^t_{HL} and q^t_{HR}.

We plan to extend the kinematic model to be able to measure the rotation of the shoulders with respect to the hips around the vertical and horizontal axes.

4.3 COMPLEX DATA REPRESENTATION

4.3.1 Activity recognition

Activities are defined as the periods of time the user has a common characteristic posture or a set of postures occuring in a certain order. The activities we consider are:

- Walking (standing, running)
- Sitting
- Lying (conscious and moving, sleeping, unconscious, dead)
- On all fours
- Kneeling
- Squatting
- Other (intermediate postures between those in the list and postures not considered).

Intermediate activities between each pair of activities in the list may also be considered: the transition between standing and sitting, between standing and lying etc. We may go even further and separate the transitions into normal ones and falls (where applicable), which will enable us to detect falls at this stage already. To increase the reliability of fall detection, a dedicated module described in Section 4.4 will still be added: it should be easier to separate falls from non-falls than to separate activities into all possible types.

Each activity has a number of parameters. Some of them were selected in cooperation with medical experts and other project partners [15][16]. Further expert consultation with the project partners representing end users will be needed before the list is finalized. The proposed parameters (not all are applicable to all activities) are:

- Duration
- Speed and angle of movement (of the whole body, of single limbs and torso, of single tags; absolute, in each direction)
- Smoothness of movement (of different parts) – measured as the speed and frequency of the changes in the direction of movement
- Flexibility (of different parts) – measured as the angles of joints
- Symmetry, straightness of posture and movement
- The height of lifting the feet (when walking)
- General movement and posture signature – it may be represented by a centroid or a ‘cloud’ of the user’s past movements and postures, possibly in the form of curves.

Machine learning

There are at least two options to form the attribute vector:

- A set of simple attributes derived from a snapshot of the body tags (described in Section 4.2) is captured. These attributes form the attribute vector.
- Simple attributes derived from consecutive snapshots of body tags are captured in the time interval of T seconds. N sets of attributes are captured, one after every update or less often ($N = T \cdot$ update frequency or $N = T \cdot$ update frequency / 2 ...). The concatenation of these N sets forms the attribute vector. A new attribute vector is obtained after every update (thus overlapping with the previous one) or less frequently.

The first option is likely suitable for the more static primary activities. The second option is needed for the intermediate activities. We may use either of the options or both.

Since some of the attributes from Section 4.2 are redundant, a reasonable subset should be selected. These are some of the options:

- $(x^t_{i\mathrm{R}}, y^t_{i\mathrm{R}}, z^t_{i\mathrm{R}})$, $v^t_{i\mathrm{R}}$, $(\varphi^t_{i\mathrm{R}}, \theta^t_{i\mathrm{R}})$; α^t_{EL}, α^t_{ER}, α^t_{KL}, α^t_{KR}, q^t_{SL}, q^t_{SR}, q^t_{HL}, q^t_{HR}
- $(x^t_{\mathrm{OR}}, y^t_{\mathrm{OR}}, z^t_{\mathrm{OR}})$, $(\Phi^t_{\mathrm{OR}}, \Theta^t_{\mathrm{OR}})$, v^t_{OR}, $(\varphi^t_{\mathrm{OR}}, \theta^t_{\mathrm{OR}})$; $(x^t_{i\mathrm{B}}, y^t_{i\mathrm{B}}, z^t_{i\mathrm{B}})$, $v^t_{i\mathrm{B}}$, $(\varphi^t_{i\mathrm{B}}, \theta^t_{i\mathrm{B}})$; α^t_{EL}, α^t_{ER}, α^t_{KL}, α^t_{KR}, q^t_{SL}, q^t_{SR}, q^t_{HL}, q^t_{HR}
- $(x_{\mathrm{OfR}}, y_{\mathrm{OfR}}, z_{\mathrm{OfR}})$, $(\Phi_{\mathrm{OfR}}, \Theta_{\mathrm{OfR}})$; $(x^t_{i\mathrm{Bf}}, y^t_{i\mathrm{Bf}}, z^t_{i\mathrm{Bf}})$, $v^t_{i\mathrm{Bf}} = v^t_{i\mathrm{R}}$, $(\varphi^t_{i\mathrm{Bf}}, \theta^t_{i\mathrm{Bf}})$; α^t_{EL}, α^t_{ER}, α^t_{KL}, α^t_{KR}, q^t_{SL}, q^t_{SR}, q^t_{HL}, q^t_{HR}
- $(x_{\mathrm{OfR}}, y_{\mathrm{OfR}}, z_{\mathrm{OfR}})$, $(\Phi_{\mathrm{OfR}}, \Theta_{\mathrm{OfR}})$; $(x^t_{\mathrm{OBf}}, y^t_{\mathrm{OBf}}, z^t_{\mathrm{OBf}})$, $(\Phi^t_{\mathrm{OBf}}, \Theta^t_{\mathrm{OBf}})$, $v^t_{\mathrm{OBf}} = v^t_{\mathrm{OR}}$, $(\varphi^t_{\mathrm{OR}}, \theta^t_{\mathrm{OR}})$; $(x^t_{i\mathrm{B}}, y^t_{i\mathrm{B}}, z^t_{i\mathrm{B}})$, $v^t_{i\mathrm{B}}$, $(\varphi^t_{i\mathrm{B}}, \theta^t_{i\mathrm{B}})$; α^t_{EL}, α^t_{ER}, α^t_{KL}, α^t_{KR}, q^t_{SL}, q^t_{SR}, q^t_{HL}, q^t_{HR}.

The attributes described in Section 4.2 are general. Task-specific attributes will likely be added. The class of the attribute vector is in both cases the activity described by the vector. In the second case the vector may contain multiple activities, which can be handled by setting the class to the longest activity or by excluding such vectors from the training data.

Activity recognition may be enhanced by appending A previous activities as recognized by the classifier to the attribute vector. This is particularly relevant in the first case where only a single set of attributes is used. The danger of appending previous activities is that the machine learning algorithm may learn that the current activity is always the same as the previous one, since this will often be the case. The problem may be circumvented by having two classifiers C_A and C_0. C_A's attribute vector contains A previous activities as recognized by C_0. C_0's attribute vector does not contain any previous activities. This way even if C_A gives a lot of weight to previous activities, the previous activities as recognized by C_0 will change, as C_0 is not burdened with C_A's inertia.

Many machine learning algorithms from publicly available toolkits [7][27] and those developed by the project partners will be tested. The primary selection criterion will be the performance (in terms of accuracy, precision, recall) on the recordings of activities (described in Section 5). We expect algorithms that cope well with many attributes (such as Support Vector Machines [6]) to perform particularly well, as well as ensemble methods [22] (such as boosting and bagging). Various attribute selection techniques will also be tried. However, we will consider the explicability of the results, too, in which decision trees and rules excel. These algorithms – even if they are not used in the final Confidence software – will help us understand which attributes are important. Such understanding will be useful to design new attributes for machine learning and expert rules.

Expert rules

These are hand-crafted rules. They will be designed after the observation of the recordings of activities and the consideration of machine-learned classifiers. Two examples:

An activity is Lying if

- the difference between the z coordinates of the left hip and left shoulder tag is at most 0.3 m

- the difference between the z coordinates of the right hip and right shoulder tag is at most 0.3 m.

An activity is Sitting if

- the z coordinates of the ankle tags are below the z coordinates of the knee and hip tags by at least 0.2 m
- the z coordinates of the knee and hip tags are within 0.3 m of each other
- the z coordinates of the shoulder tags are at least 0.3 m above the z coordinates of the hip tags.

The rules may use all simple data representations described in Section 4.2. Whether they may also use the complex data representations is yet to be decided. The reason the use of complex data representations may be inadvisable is that they are derived from simple data representations, which is a process that introduces errors. The rules also cannot be perfectly reliable, so the errors may be compounded.

Final decision

There are at least three options to reach the final decision regarding the activity:

- The machine-learned classifier and the expert rules may output a degree of certainty in their decision and the final decision may be to select the one that is more certain.
- Another machine-learned classifier may be used. Its attributes are the outputs of the activity classifier and the rules and the attributes they use to reach their decision. Its class may be either the activity or the choice of the classifier or the rules.
- Expert rules may be designed to choose between the classifier and the rules depending on circumstances.

One of these options will be selected once the activity classifier and the rules are designed and their performance is analyzed.

4.3.2 Movement primitives

Several methods to represent movement with simple features called movement primitives have been developed. These methods were inspired by the movement of living organisms, which is thought to consist of primitive 'building blocks' as well. They are typically used in robotics to model the movement and to control robots. For the purpose of Confidence, only the modeling part is interesting.

Movement is typically represented by a mixture of nonlinear differential equations [10][11][23][25]. Each particular type of movement is characterized by its set of weights. The weights are learned from an observed movement by some form of regression. These weights are the data representation we are seeking.

Other approaches to modeling movement with movement primitives are possible: they may be selected manually [18][20] or derived automatically with dimensionality reduction techniques [12].

4.4 FALL DETECTION

4.4.1 Simple machine learning

This is machine learning on simple data representations described in Section 4.2. The procedure is the same as in activity recognition described in Section 4.3.1. Simple attributes derived from consecutive snapshots of body tags are captured in the time interval of T seconds. N sets of attributes are captured, one after every update or less often ($N = T \cdot$ update frequency or $N = T \cdot$ update frequency / 2 ...). The concatenation of these N sets forms the attribute vector.

A new attribute vector is obtained after every update (thus overlapping with the previous one) or less frequently.

Like in activity recognition, a subset of attributes is selected. Task-specific attributes will likely be added to the general attributes from Section 4.2. The class of the attribute vector is True if the interval described by the vector contains a fall and False otherwise. If a vector contains only a part of the fall, this can be handled by setting the class to True if it contains more than half of the fall or by excluding such vectors from the training data.

The machine learning algorithm will be selected the same way as for activity recognition.

4.4.2 Complex machine learning

This is machine learning on complex data representations described in Section 4.3. The attribute vector consists of activities and their parameters. There are at least two ways to select the time interval from which the attribute vector is formed:

- The last two or more activities are used if only the activities in the list in Section 4.3.1 are considered. The last three or more activities are used if intermediate activities between those in the list are considered as well. Very short activities can be ignored to reduce the effect of erroneous activity recognition.
- All the activities in the time interval of T seconds are considered.

The class of the attribute vector is True if the interval described by the vector contains a fall and False otherwise.

The machine learning algorithm will be selected the same way as for activity recognition described in Section 4.3.1. Different algorithms may prove suitable, though, particularly because Complex machine learning will probably work with fewer attributes than the simple one.

4.4.3 Expert rules

These are hand-crafted rules. They will be designed after the observation of the recordings of activities and the consideration of machine-learned classifiers. An example:

A fall has occurred if:

- the z coordinates of shoulder and hip tags are lower than 0.5 m at time t
- the average downward speed of these tags between the times t and t – 1.5 s is higher than 1 m/s
- the z coordinates of the tags on the arms are higher than 0.5 m at time t – 1.5 s.

The rules may use all simple (Section 4.2) and complex (Section 4.3) data representations.

Rules regarding velocity and acceleration, such as those used by the accelerometer- and gyroscope-based fall detection methods described in Section 1, can be added here.

4.4.4 Final decision

The final decision whether a fall has occurred is made the same way as in activity recognition described in Section 4.3.1. Since we have three detection methods here, we may also use voting, possibly weighted by the methods' certainty.

4.5 DETECTION OF LYING IN AN UNUSUAL POSITION

4.5.1 Simple machine learning

This module very similar to Simple machine learning used for fall detection. Two time intervals can be used:

- A short one can analyze fast motion (exercising, crawling on the floor due to an injury …). If the user is still, however, it cannot tell whether he/she is sleeping or unconscious/dead.

- A long one with the coordinates also sampled at long intervals can differentiate sleeping from unconsciousness/death.

An even simpler variant of machine learning uses a single snapshot of the body tags, so that only posture is taken into account, not movement. The attribute indicating whether the location of the user is marked for lying is added to the attribute vector in all cases.

4.5.2 Complex machine learning

This module very similar to Complex machine learning used for fall detection, except that the current activity must be lying. The attribute indicating whether the location of the user is marked for lying is added to the attribute vector.

4.5.3 Expert rules

Expert rules are also used. A simple rule covers most of the cases:

The user is lying in an unusual position if:
- his/her current activity is lying
- his/her current location is not marked for lying
- he/she does not move much (which is a parameter of the activity).

4.5.4 Final decision

The final decision whether the user lies in an unusual position is made the same way as in fall detection described in Section 4.4.4.

4.6 General disability/disease detection

Whether the user has developed some sort of disability, has fallen ill is otherwise unwell will be detected from a number of statistics of daily life activities. These statistics will be selected in cooperation with the project partners representing end users, who will consult medical experts [15]. In addition to the parameters of the activities listed in Section 4.3.1, the proposed statistics are:
- The time needed to pick up the portable device, which is supposed to indicate rough manual dexterity
- The time needed to press a button on the portable device after the hand is near the device, which is supposed to indicate fine manual dexterity
- The speed of standing up from a sitting position
- The speed of sitting down
- The speed of standing up from a lying position
- The speed of lying down
- Reflexes
- Balance
- The number of times the user gets up at night
- The time spent sitting or lying still
- The time spent in motion
- The time spent during recognizable activities
- The time spent in typical relations to other tags and places, e.g. on the bed, in each chair, in the kitchen, etc.

Some of the statistics are straightforward to measure, whereas others require dedicated methods, some of them quite complex. These methods will be designed after the list of statistics if finalized and confirmed by medical experts.

For the listed statistics, the following aggregate values are computed (where applicable):

- the average value over some period of time (e.g., the average speed of sitting down in a day)
- the maximum value (e.g., the maximum speed of sitting down in a day)
- the standard deviation of the values (e.g., the standard deviation of the speed of sitting down in a day)
- the number of occurrences (e.g., the number of time the user has sat down in a day).

These values are aggregated over multiple time periods (to be selected by the project partners representing end users in cooperation with medical experts). Multiple time periods are needed to detect both rapid changes (hours to detect the onset of a disease and go to the doctor) and long-terms disabilities (months to detect weakening of leg muscles and start using a cane).

Finally criteria are applied to the statistics to determine whether their values warrant raising a warning. The criteria may be absolute (e.g., raise a warning if the user walks with less than *x* m/s for *y* hours) or relative (e.g., raise a warning if the user's walking speed decreases by more than *x* % in *y* hours). They will also be selected by the project partners representing end users in cooperation with medical experts. A way to determine absolute criteria is to observe normal persons (or persons in normal conditions) and persons who are unwell and measure the relevant statistics. Machine learning algorithms can then be used to learn a classifier to distinguish between the two.

4.6.1 Fuzzy logic

Fuzzy logic [29] is a module that is able to recognize unusual behavior in general by learning about the usual behavior. This is useful because general unwellness, which we are trying to detect, is reflected in equally general abnormalities. The basic idea of Fuzzy logic module is that things that happen often are usual and things that happen rarely are unusual. So if we remember what happened in the past and how often it happened, we can judge how usual or unusual an observed behavior is based on the number of times this behavior or behavior similar to it has happened before. The behavior this module deals with is specific to each user and the module needs some time to learn it before it starts working properly.

Let us look at a simple example. We divide an observed room into a mesh of equal-sized rectangles. We observe the user moving in the room. When we detect that he/she is present in a certain rectangle, we increment a counter that counts the number of occurrences in that rectangle. After some learning we know where the user usually moves. Now suppose that he/she steps on the table. Because something like that (stepping on the table) has never happened before, the counter for the rectangle where the table is placed is zero or at least relatively low compared to the other counters. That means that we have just observed an unusual behavior.

Fuzzy logic module is basically just an enhancement of the procedure described in the previous paragraph. The state of the user in terms of Fuzzy logic module is not characterized only by his/her position, but also by other attributes of his/her movement. The module takes into account the following attributes:

- the position in space (in which rectangle)
- the height of the current position (e.g., low on the ground, sitting height, standing height, high)
- the velocity in the (x, y) plane (e.g., standing still, walking slowly, walking normally, walking fast, running)
- the velocity in the z direction (e.g., moving up or down quickly or remaining at the same height)
- the direction of movement (e.g., north, east, south, west).

As the examples show, the real values of the attributes are discretized (e.g., the height of 0.632 m is described as sitting height, the velocity of 1.32 m/s is described as normal walking speed). The module uses fuzzy discretization [3] – that means that an attribute can belong to more than one discrete class with a certain probability (e.g., the height of 0.713 m could be discretized into sitting height with 62 % probability and standing height with 38% probability). Fuzzy discretization is a compromise between low memory consumption and the need for few learning examples on one hand and high accuracy of representation on the other hand.

Knowledge can be added to Fuzzy logic module by remembering from which previous states the user usually moves to the current state and how long he/she usually stays in a certain state.

The module also makes it possible to explain why a certain behavior is unusual. For instance if the user steps on a chair to reach an object on top of a closet, Fuzzy logic module will recognize that as unusual behavior and explain it like this: »it is unusual because: the user has never been that high in the current position (he/she usually walks on the ground near the closet); the user has been in this unusual position for quite a long time (while trying to reach the object on top of the closet); the user has never moved like he/she moved before ending up in this unusual position (moving up – stepping on the chair)".

The knowledge about the usual behavior is gained with time and can also be automatically (slowly) adapted to new behavior patterns by forgetting old data and adding new data acquired during the normal use to the representation of the usual behavior. The usual behavior can be represented either as the sum of all the occurrences of a certain state through all of the observed time or just through a certain period of time after a certain action. For instance, Fuzzy logic module can separately remember what usually happens after the user wakes up. Let us say that user usually goes to the toilet a minute after waking up, but one day does not do so. Fuzzy logic module that remembers what usually happens after waking up will know that the user did not go to the toilet and that this is unusual behavior.

5 EXPERIMENTS

To be able to detect specific types of behavior, examples of such behavior have to be recorded for analysis and training of machine learned classifiers. So far we know that recordings of falls are necessary. We also need recordings of normal behavior to test whether we can distinguish it from falls. We will likely require recordings of behaviors related to the general disability/disease detection as well, but we must first finalize the list of the statistics used for this task.

In order to detect all falls, we will record multiple types of falls. The list of falls and their descriptions was prepared in cooperation with medical experts and is pending approval of the project partners representing the end users [15]. The descriptions of normal behaviors were prepared in cooperation with a physician.

Since Confidence hardware is not available yet, we will record the behaviors of interest with Smart infrared motion capture equipment, which we have at hand. The Smart system consists of six infrared cameras, which record markers placed on the body. These markers are the substitutes for Confidence tags. The accuracy of the Smart system is ~1 mm and the sampling rate 60 Hz. Smart's characteristics thus significantly exceed the expected characteristics of the Confidence hardware, so noise will have to be added to the recordings (more on this in Section 5.2).

The recordings will be carried out by healthy young volunteers, as it would be too dangerous for the elderly to fall. They will be supervised by a physician, who will ensure the authenticity of their behavior. The experiments will be carried out according to the guidelines in D6.1 Ethics Manual.

5.1 BEHAVIORS TO RECORD

5.1.1 Falls

The list of falls is not exhaustive because some types of falls are difficult to measure (e.g., lateral falls because the markers are placed laterally and would be obscured or broken and may even injure the test subject) and some are too dangerous (e.g., falls on the stairs).

Fainting. When falling from a standing position, the person falls slowly, with lesser acceleration towards the ground compared to accidental falls. The knees bend and the body descends vertically. No attempt to break the fall is made. He/she lands flat on the ground.

- While standing still [forward]
- While standing still [backward]
- While walking [forward]
- While sitting (eating), ending up leaning to one side
- While sitting (eating), ending up with the head on the table

Tripping over an object [forward]. The person falls quickly, with greater acceleration towards the ground compared to fainting. He/she attempts to regain balance or break the fall.

- Landing flat on the ground
- Landing on all fours

Slipping [backward]. Similar to tripping, but in the opposite direction.

- Landing flat on the ground
- Landing sitting

Falling from a chair [forward]

- Landing flat on the ground (after trying to stand up)
- Landing on all fours (after trying to stand up)

- Landing sitting (after sliding down)

Being assaulted. The person falls backwards and lands lying flat on the ground. The fall is preceded by defensive motion, i.e. raising the arms in front of the face. This motion is in itself interesting to detect.

5.1.2 Normal behavior

- Walking – slow, the arms swing less than in a young person
- Lying down on a (high) bed on the back – the movement is slow, the slippers are removed first
- Lying down on a (high) bed on the stomach
- Lying down on a (low) mattress on the back
- Lying down on a (low) mattress on the stomach
- Sitting down on a chair and standing up – several unnecessary moves are made, the person grips the chair many times, he/she leans on the knees with the hands when standing up

5.2 ADDING NOISE

As explained, the accuracy of the Smart system is ~1 mm, which significantly exceed the expected characteristics of the Confidence hardware. Thus, noise will have to be added to the recordings. This idea is illustrated in Figure 5, where $c(n)$ is the positioning data obtained with the Smart system and $\hat{c}(n)$ is the positioning data with accuracy more similar to the one of the Confidence system. $w(n)$ will be a random variable with a statistical distribution to be determined by WP2.

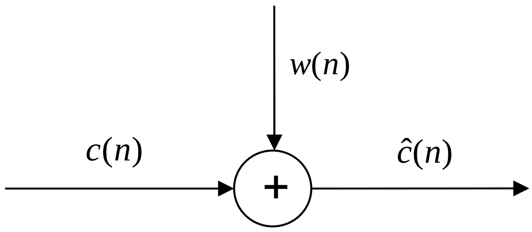

Figure 5. Adding noise to the data acquired with the Smart system.

6 EVALUATION OF THE PROPOSED METHODS

Smart system will give us very precise recordings of behaviors of interest (as described in Section 5) and we will also have a method for adding varying degrees of noise to them. We will take advantage of this to investigate the sensor precision required to detect certain types of behavior. Since it is easy to ignore one or more tags in a recording, we will also investigate the number and the placement of tags required to detect each type of behavior. As a result, we will be able to tell:

- the sensor precision and the number of tags necessary to detect a given behavior with a given accuracy
- the placement of tags best suited to a given task.

7 CONCLUSION

The first major goal of Confidence is fall detection, where quite a lot of research has been done already. However, the existing solutions are based on accelerometers, gyroscopes and computer vision, whereas Confidence uses radio tag localization. Some of the principles used by the existing solutions can be applied to Confidence, but due to the novel hardware platform we will focus on fall detection methods specifically tailored to it. The reason for our decision is twofold: we expect such methods to perform better, since they will fully utilize the inputs available, and they are scientifically novel. Since we will ensure the robustness of the reconstruction and interpretation subsystem by running several methods for the same task in parallel and combining their outputs, we can easily add methods from the existing solutions if desired.

We expect that given the rich data available from the localization subsystem, fall detection will be quite reliable if the full complement of tags is used. Training an accurate classifier using machine learning algorithms appears feasible and the expert rules will probably cover a large portion of the falls as well, so a combination of these two approaches is expected to be reasonably robust. Whether sufficient reliability can be achieved with fewer tags and if so, which tags give the best results, is an open question. The answer to it may also tell us how practical and marketable a Confidence-like solution can be.

The second major goal of Confidence is general disability/disease detection. We decided not to focus on any particular disability or disease because it is probably easier to assess whether something is wrong with the user in general than to tell what specifically ails him/her. In addition, the specifics of the problem are not important – what matters is whether one is present or not. We are not aware of any previous attempts at this task, so if we solve it successfully, this will certainly be an important contribution. Since the project partners representing end users are still investigating disability/disease indicators, we could not plan our methods in detail yet. We only decided on a framework for managing statistics related to disabilities and diseases. We will devise the software modules for computing specific statistics when we know what exactly these statistics are. The only method we have planned in detail so far is the one based on fuzzy logic described in Section 4.6.1.

REFERENCES

[1] AlertOne Services, Inc. iLife Fall Detection Sensor. http://www.falldetection.com/ [2008-07-18].

[2] Bajd, T. (2001). Osnove robotike. Založba FE in FRI, Ljubljana, Slovenia.

[3] Bezdek, J. C (1993). Fuzzy Models – What are they, and why?. IEEE Transactions on Fuzzy Systems, 1 (1), 1–6, 1993.

[4] Bourke, A. K., and Lyons, G. M. (2006). A threshold-based fall-detection algorithm using a bi-axial gyroscope sensor. Medical Engineering & Physics 30 (1), 84–90.

[5] Bourke, A. K., Scanaill, C. N., Culhane, K. M., O'Brien, J. V., and Lyons, G. M. (2006). An optimum accelerometer configuration and simple algorithm for accurately detecting falls. In proceedings of 24th IASTED International Conference on Biomedical Engineering, 156-160.

[6] Cristianini, N., and Shawe-Taylor, J. (2000). An introduction to Support Vector Machines and other kernel-based learning methods. Cambridge University Press.

[7] Demšar, J., Zupan, B., Leban, G. (2004). Orange: From experimental machine learning to interactive data mining. White Paper (www.ailab.si/orange), University of Ljubljana, Faculty of Computer and Information Science.

[8] Doughty, K., Lewis, R., and McIntosh, A (2000). The design of a practical and reliable fall detector for community and institutional telecare. Journal of Telemedicine and Telecare 6, 150–154.

[9] Fu, Z., Culurciello, E., Lichtsteiner, P., and Delbruck, T. (2008). Fall detection using an address-event temporal contrast vision sensor. Proceedings of IEEE International Symposium on Circuits and Systems, 424–427.

[10] Ijspeert, A. J., Nakanishi, J., and Schaal, S. (2001). Trajectory formation for imitation with nonlinear dynamical systems. Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems.

[11] Ijspeert, A. J., Nakanishi, J., and Schaal, S. (2002). Movement imitation with nonlinear dynamical systems in humanoid robots. Proceedings of International Conference on Robotics and Automation.

[12] Jenkins, O. C., and Matarić, M. J. (2002). Deriving action and behavior primitives from human motion data. In Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems.

[13] Kangas, M., Konttila, A., Lindgren, P., Winblad, I., and Jamsa, T. (2008). Comparison of low-complexity fall detection algorithms for body attached accelerometers. Gait & Posture 28 (2), 285–291.

[14] Kangas, M., Konttila, A., Winblad, I., and Jamsa, T. (2007). Determination of simple thresholds for accelerometry-based parameters for fall detection. In proceedings of 29th Annual International Conference of the IEEE, Engineering in Medicine and Biology Society, 1367–1370.

[15] Luštrek, M., and Gams, M. (2008). R3.4-3 Scenario Details. Project report, Confidence.

[16] Luštrek, M., Gams, M., and González, N. (2008). R3.4-2 Scenario Details. Project report, Confidence.

[17] Maybeck, P. S. (1979). Stochastic models, estimation, and control. Mathematics in Science and Engineering 141.

[18] Moeslund, T. B., Fihl, P., Holte, M. B., and Moeslund, T. B. (2006). Action recognition using motion primitives. Proceedings of 15th Danish Conference on Pattern Recognition and Image Analysis.

[19] Murphy, K. Kalman filter toolbox for Matlab. http://www.cs.ubc.ca/~murphyk/Software/Kalman/kalman.html

[20] Nakaoka, S., Nakazawa, A., Yokoi, K., and Ikeuchi, K. (2004). Leg motion primitives for a humanoid robot to imitate human dances. Journal of Three Dimensional Images 3 (1), 73–78.

[21] Noury, N., Barralon, P., Virone, G., Boissy, P., Hamel, M., and Rumeau, P. (2003). A smart sensor based on rules and its evaluation in daily routines. In proceedings of 25th Annual International Conference of the IEEE, Engineering in Medicine and Biology Society, volume 4, 3286–3289.

[22] Okun, O., and Valentini, G. (2008). Supervised and unsupervised ensemble methods. Springer.

[23] Perk, B. E., and Slotine, J. J. E. (2006). Motion primitives for robotic flight control.

[24] Sarkka, S. (2008). Unscented Rauch-Tung-Striebel smoother. IEEE Transactions on Automatic Control 53 (3)

[25] Schaal, S. (2003). Dynamic movement primitives – a framework for motor control in humans and humanoid robotics. Proceedings of The International Symposium on Adaptive Motion of Animals and Machines

[26] Willis, D. J. (2000). Ambulation monitoring and fall detection system using Dynamic Belief Networks. PhD thesis, Monash University, School of Computer Science and Software Engineering.

[27] Witten, I. H., and Frank, E. (2005). Data Mining: Practical machine learning tools and techniques, 2nd Edition. Morgan Kaufmann, San Francisco, USA.

[28] Wu, G. (2000). Distinguishing fall activities from normal activities by velocity characteristics. Journal of Biomechanics 33 (11), 1497–1500.

[29] Zadeh, L. (1965). Fuzzy Sets. Information and Control 8, 338–353.

[30] Zhang, T., Wang, J., Liu, P., and Hou, J. (2006). Fall detection by embedding an accelerometer in cellphone and using KFD algorithm. IJCSNS International Journal of Computer Science and Network Security 6 (10).

[31] Zhang, T., Wang, J., Xu, L., and Liu, P. (2006). Fall detection by wearable sensor and one-class SVM algorithm. Lecture Notes in Control and Information Science 345, 858–863.