Towards Configurable Motion Capture with Prioritized Inverse Kinematics

Manuel Peinado [1] , Bruno Herbelin [2] , Marcelo Wanderley[3], Benoit Le Callennec[2], Ronan Boulic[2], Daniel Thalmann[2], Daniel Méziat[1]

(1) Escuela Politécnica

University of Alcalá, Spain

(2) VRLAB

EPFL, Lausanne, CH-1015, Switzerland

(3) Mc Gill University,

Montreal, Canada

E-mail: manupg@aut.uah.es, Bruno.Herbelin@epfl.ch, marcelo.wanderley@mcgill.ca, benoit.lecallennec@epfl.ch, Ronan.Boulic@epfl.ch, Daniel.Thalmann@epfl.ch, meziat@aut.uah.es

Abstract

This paper explores the potential of prioritized Inverse Kinematics for full body motion capture. We are especially interested in compensating for the use of a minimal set of sensors by proposing pertinent constraints associated with a priority level. We evaluate the believability issues and the performance requirements arising through a case study exploiting recorded motions of professional clarinet performers. We conclude by presenting a framework in which end users can easily configure the IK engine for 3D real-time interaction.

Keywords

Motion Capture, 3D interaction, gesture.

1. Introduction

Numerous classes of applications, from CAD-CAM to entertainment, need to control the 3D posture of complex articulated bodies like digital mannequins, animated characters, partly-autonomous virtual human, etc.. In such context, traditional interaction devices like the mouse often find their usability limit as they provide for only a small number of independent degrees of freedom. While they fit perfectly for defining the few high level parameters driving an integrated motion engine (e.g. walking, running), they failed to define simultaneously the numerous parameters of an arbitrary posture. This has been acknowledged very early by Badler who investigated the use of four magnetic sensors (waist, head and both hands) for driving the posture of a human model with Inverse Kinematics [1]. The goal is to recreate human postures while minimally encumbering the end user with sensor attachments. However, the uncontrolled degrees of freedom, like the swivel angle of the arms, can lead over time to important differences between the end user and the virtual human model as shown in [2].
An alternate approach introducing springs and a discomfort criteria was proposed by Hirose [3]. It required some tuning of the spring stiffness and some higher computing time to stabilize the posture. An analytic method was generalized by Molet [4] to robustly distribute the sensor data over multiple joints. Other approaches, identifying in addition the skeleton structure and segment lengths, were proposed by Bodenheimer [5] and O'Brien [6]. A special analysis of the measurements was proposed by Shin to cope with animated characters with very different proportions compared to the person wearing the sensors [7]. Recently Zordan proposed an

improvement to physically-based method applied for optical sensor data [8]. However, all the recent approaches reduce the ambiguity of the posture determination by using a high number of magnetic or optical sensors.

In this paper, we explore the use of a small number of sensors together with prioritized Inverse Kinematics to reduce the size of the solution space. Prioritized Inverse Kinematics allows constraints to be associated with a priority level so that important properties are enforced first (e.g. balance) while less important adjustments are made in the remaining solution space [9]. The next section briefly recalls how the strict priority levels can be enforced owing to special projection operators. We then evaluate the believability issues and the performance requirements arising through a case study exploiting recorded motions of professional clarinet performers. We conclude by presenting a framework in which end users can easily configure the IK engine for 3D real-time interaction.

2. Motion Capture as the Solution of a Prioritized IK Problem

2.1. Conflicting Constraints and Synergistic Solutions owing to Priority levels

Human postures can be characterized by at least three potentially conflicting requirements (the corresponding constrained body part and goal are indicated into parenthesis):

- Maintaining the balance (the center of mass has to project in the support area),
- Viewing (eye gaze direction has to go through a target area),
- Reaching (hands have to reach a target location, carry or manipulate an object).

A common approach is to enforce those constraints by associating them with independent set of joints; for example the root node is responsible for maintaining the balance, each arm is responsible for controlling the corresponding hand, and the neck chain is responsible for controlling the eye gaze direction. Fast analytic solutions are available for solving such simplified context [1][7]. However, such an approach also reduces the solution space in a way that prevent finding some pertinent solutions. This limitation is most likely to occur when all the constraints cannot be enforced simultaneously. We advocate for a more ***synergistic*** approach where all the constraints may take advantage of every joints to achieve their goal [9]. Conflicts among individual constraint solutions now become a central issue to solve as they may share common joint subsets. We solve it by providing the possibility to associate a priority level to an individual constraint, or to a constraint set. The final solution is built owing to projection operators (briefly outlined in the next section) that guarantee the strict enforcement of high priority constraints first; low priority constraints being enforced in the remaining solution space. The architecture fully developed in [9] can enforce an arbitrary number of priority levels; however one should keep in mind that the posture solution space has a finite dimension that cannot exceed the sum of all the dimensions of the constraint spaces.

For example, in a standing reach task, the balance has to be enforced with the highest priority. Similarly, the viewing constraint has a higher priority than the reach because the subject has to look at the target location (Fig. 1). In case a low priority constraint cannot be fully satisfied, the resulting error is also minimized as can be noticed for the right hand reach constraint in Fig. 1.

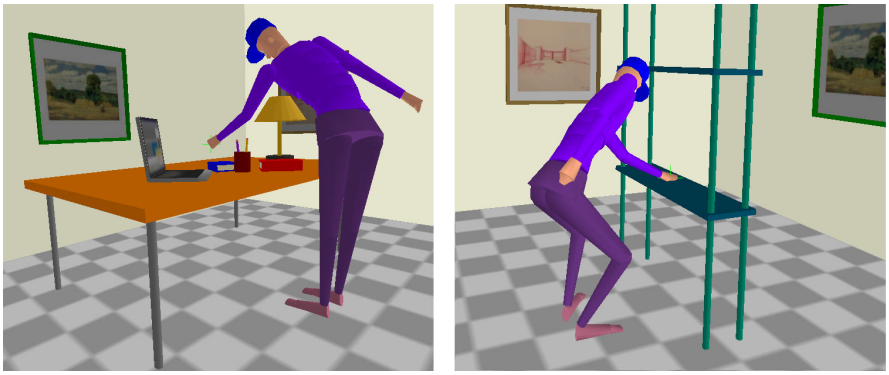
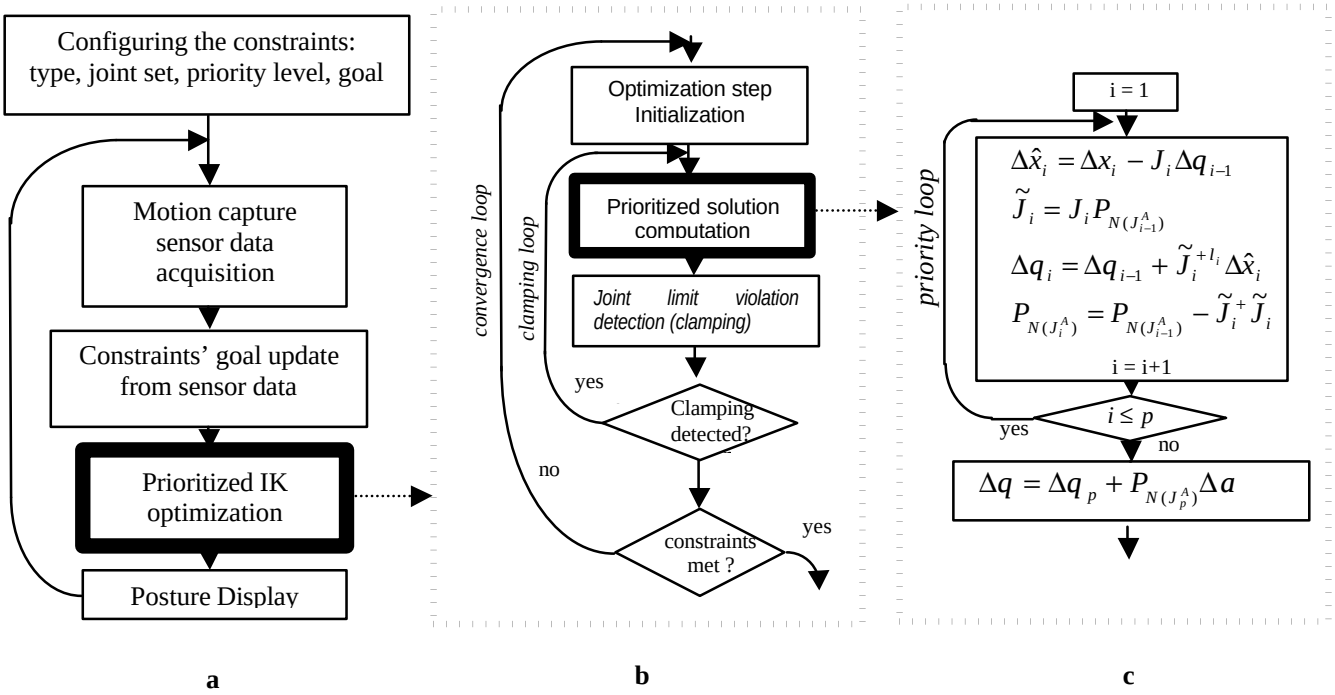


Fig. 1: Postures obtained by assigning priorities to three types of constraints (balance, gaze, reach).

2.2. Principle of the Motion Capture with Prioritized IK

Figure 2 provides an overview of motion capture with prioritized IK (2a left), together with a more detailed view of the IK optimization loop (2b middle) and a detailed description of the construction of the prioritized solution (2c right). We now briefly describe each of these drawings.



a **b** **c**

Fig. 2: Outline of the motion capture process (a) exploiting the Prioritized Inverse Kinematics convergence loop (b); details of the prioritized solution construction (c).

2.2.1 General Overview

The general structure of the motion capture appears on Fig. 2a; first an initialization phase determines the set of constraints that will be active during the motion capture session. Each constraint can be characterized by:

- its type: balance, position, orientation
- its controlled effector (i.e. the body part to which the constraint is applied)
- how the goal of the effector is computed from which sensor data
- its recruited joint set (i.e. the joints that will contribute to achieve the constraint)
- its priority level
- its activity: a constraint can be activated/deactivated based on some triggering event

An additional optimization criteria can be specified at that initialization stage too; a typical optimization criterion is to be close to a desirable posture. This is also application dependent and can be tailored to meet end-users needs.

Then we enter the main motion capture loop, where the sensor data are acquired and expressed in the same reference frame and units as the virtual human, prior to build the current state of the constraints' goals. The next stage is to run the prioritized IK convergence loop to obtain a posture that meets the specified constraints. When used for 3D real-time interactions the convergence loop is limited to one step to ensure a sufficient frame rate (performance data are provided in section 3.2.7). For off-line processing there is no such limitation; the optimization exits the loop when all constraints are met, or when the errors become stable, or when a maximum number of iteration has been reached. The final stage is to display the resulting posture and/or to record it.

2.2.2 The Prioritized Inverse Kinematics Convergence Loop

Our general architecture relies on an efficient computation of projection operators enforcing the constraint satisfaction at individual priority levels [9]. This property being valid only within the neighborhood of the current state, the norm of any desired constraint variation, noted Δx, is limited to a maximum value and we iterate the computation of the prioritized solution, noted $\Delta\theta$, until the constraints are met or until the sum of the error reaches a constant value (Fig 2b).

Figure 2b also highlights how we handle the inequality constraints associated to the joint limits. Basically we check whether the computed prioritized solution $\Delta\theta$ leads to violate one or more joint limits. If it is the case, additional equality constraints are inserted to clamp the flagged joints on their limit, and a new prioritized solution is searched as long as no additional inequality constraint is violated. This mechanism is necessary to guarantee the constraints' error minimization.

2.2.3 Building the Prioritized IK Solution

Figure 2c provides the technical details of the prioritized solution construction. The full description can be found in [9]. Let us denote $\boldsymbol{J}$ the Jacobian matrix expressing the partial derivatives of a constraint $\boldsymbol{x}$ with respect to the degrees of freedom q. We use its pseudo-inverse, noted $\boldsymbol{J}^+$, to build the projection operators $P_{N(J)}$ on the kernel of $\boldsymbol{J}$, noted *N(J).* The sub-space *N(J)* can be interpreted as the *remaining solution space for lower priority constraints*, once the solution for the constraint $\boldsymbol{x}$ is determined with $\boldsymbol{J}^+$.

For the sake of clarity, we assume that each constraint has a distinct priority level indexed by i, with 1 being the priority of highest rank, and p being the total number of priority levels.

The priority management needs to introduce an additional Jacobian matrix, called the *Augmented Jacobian* and noted J_i^A. It simply piles up *all* the individual constraint Jacobians J_i from level one to level i into one matrix.

The initialization stage consists in:

- computing the individual Jacobian matrices,
- initializing the partial solution vector $\Delta\theta_0$ to zero
- initializing the projection operator P (on the remaining solution space) to Identity.

At that stage, all the joints are assumed to be within their limits.

Then, the computation of the prioritized solution $\Delta\theta$ starts from the highest level contribution (for i =1). The priority loop adds the contribution of one priority level at a time. The contribution of each priority level i is decomposed as follow :

- First the compensated constraint $\Delta\hat{x}_i$ removes the influence of all higher priority levels, from *1* to *i-1*.
- Then we exploit an intermediate Jacobian $\tilde{J}_i$ which is the restricted Jacobian J_i to the remaining solution space for level i (this space is noted $N(J_{i-1}^A)$).
- The cumulated solution up to level i is updated into $\Delta\theta_i$.
- Afterwards, the projection operator $P_{N(J_i^A)}$ is updated for the next priority level *i+1*.

After the loop, we can add an optional criterion optimization term expressed in the joint variation space, noted $\Delta\alpha$. The sections 3.2.4 and 3.2.5 exploit this possibility by attracting the current posture towards a desired posture. In such a context the vector is simply the error vector between the desired posture and the current posture.

The next section explores a case study in detail.

3. IK-Based Motion Capture with Minimal Sensor Set

The present section exploits a database of captured motions in the field of music performance studies. For each motion we have a set of optical markers' position and the video of the performer. The next section briefly recalls the purpose of this research. Then we present various tests highlighting the interest of a prioritized IK technique to recover a believable motion from a partial set of sensor data.

3.1. The Musician Performance Case Study

Research into measuring performer movements derived from the third author's interest in analyzing the correlation between physical and musical gestures. We specifically focused on ancillary body movements (also called accompanist/non-obvious gestures or expressive movements), movements that do not have a direct link to the generation of sound, but are an integral part of the performance.

Although it is known that musicians perform various movements that are not primarily related to sound production [10] [11], there is little evidence on how and why they do it.

Questions that we are analyzing in this research concern the production and reproducibility of performer ancillary body movements: a) are ancillary movements essential for performance (i.e. can a musician play without moving her body and instrument)? b) is the same musician playing the same piece several times going to produce similar movements (i.e. how are these movements linked to the structure of the piece being played)? and c) are different musicians going to perform similar movements while playing the same piece or are these movements idiosyncratic?

Results so far [12][13] suggested the importance of several factors, highlighting at least three levels of influence: *material/physiological* (the instrument characteristics and the constraints of the human body), *structural* (the characteristics of the piece being played, e.g. rhythm, articulation, tempo, etc.), and *interpretative* (the momentarily choice of movement strategies by the performer). The two first levels are common to several musicians; the third one is mostly idiosyncratic.

We are currently studying the musical implications of this research in terms of audience perception [14] and relationship to the score being performed [15]. Expected results, in addition to a better understanding of music performance and the role of ancillary movements, include the design of new enactive interfaces and digital musical instrument that take expressive movements into account.

With the IK motion capture tests presented in the following sections, we want to explore how much we can recover from a given performance when knowing only the trajectory of a small set of optical sensors. It is clear that the material/physiological influence dominates in the IK motion capture process, but we expect to be able to recover part or most of the structural and interpretative influences from this very limited information.

3.2. Recovering the Motion with IK

3.2.1 Sensor Configuration and Problem Statement

We face the following challenge: a professional performer plays the clarinet. Her motion is captured using six positional sensors: two for the ends of the clarinet, and the remaining four for the head, shoulder, hip and knee of the player. All the sensors are located in the right side of the body.

The amount of data provided by this set of sensors is insufficient for a traditional motion recovery method, such as [4]. An alternate approach must be used in order to reconstruct the original motion, at least to the extent that its main features are preserved. For this purpose, and inspired by previous works such as [1], we propose to use Inverse Kinematics (IK), a technique that is naturally suited for such problems since its main goal is the computation of a body configuration that satisfies a set of geometric constraints.

The application of IK to our clarinet playing scenario seems straightforward: one could just create four end effectors (head, shoulder, hip and knee) that followed the corresponding the sensors, and run IK at each frame in order to find a pose that satisfies all constraints. Naïve as this approach may seem, it works fine with some modifications that we describe in the rest of this section.

3.2.2 Animating the Head

Three constraints are created to achieve a realistic motion of the head. The first one is trivial: a positional constraint that has the head (skullbase joint) follow the head sensor.

If we only use this constraint, an important problem arises: the player's mouth is not, in general, located at the clarinet barrel. This leads of course, to unrealistic results that can be easily avoided by creating an additional constraint forcing the head to follow the clarinet barrel sensor.

Although it may seem that these two constraints are sufficient to ensure a satisfying head motion, there is yet another concern related to the orientation of the head. Indeed, using only two positional constraints leads to the head adopting some visually unpleasant orientation at certain frames (see Figure 3a)

To solve this, we make use of the following observation: during their performances, clarinettists maintain a constant angle between the instrument and their head, in the sagittal plane. This angle has been found to be of about 160º, as shown in Figure 3b. Thus, we can achieve a realistic motion by attaching an orientation constraint to the head of the figure, and update it on every frame so that it keeps an angle of 160º with the clarinet, whose orientation is also recomputed on a per frame basis.

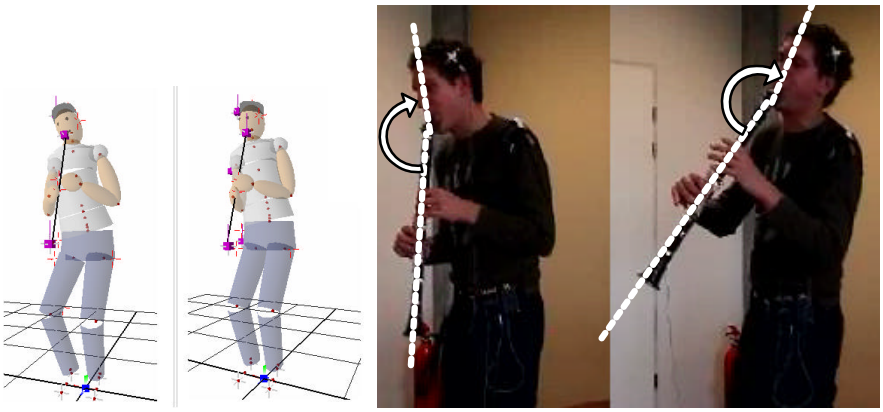


Fig. 3 (a) With only two constraints, the orientation of the head looks unrealistic (left) unless an orientation constraint is added (right). (b) A clarinettist typically keeps a constant angle between his head and instrument

3.2.3 Keeping Balance and the Feet on the Ground

Our observation of the motion of the professional clarinettists analysed, reveals that these particular performers keep their feet on the ground all the time, without the slightest lifting of the heels. This is easy to implement as a first approximation: we just have to add a pair of positional constraints to the ankles to prevent them from moving. An additional center of mass constraint is used to control the balance of the figure. The center of mass is forced to project onto the support polygon defined by the position of the feet on the floor.

See section 3.2.6, "Enforcing Priority Levels", for more information on how these constraints are actually set up so that they remain unaffected by the presence of other less critical constraints.

3.2.4 Flexing the Legs

When using IK to animate humans, obtaining leg configurations that look well is one of the most difficult part. This is mainly due to the fact that, in the standard rest posture, the legs are stretched in a near singular configuration.

If we observe a real musician playing the clarinet, we realize that leg flexion plays an important role in the overall expressiveness of the performance (Figure 4a). Thus, our virtual player will not look realistic unless we find a good way to recover the motion of the legs from the information provided by the position of the available sensors.

The easiest way to go may seem to attach a constraint to the knee of the figure so that it follows the corresponding sensor. Unfortunately, this approach does not work properly. The IK engine simply fails to produce realistic leg configurations as we have data on only one side of the body. A workaround consists in

making use of a well known feature of the IK method, known as the optimization secondary task. This task allows us to satisfy an optimization criterion, in addition to the primary tasks (constraints). This criterion can serve a lot of different purposes, but its most common use is to "attract" the body configuration towards a goal posture.

In our current case, optimization can help us achieve a satisfactory motion of the legs. All we need to do is select a goal posture in which the legs are moderately flexed, as the one in Figure 4b.

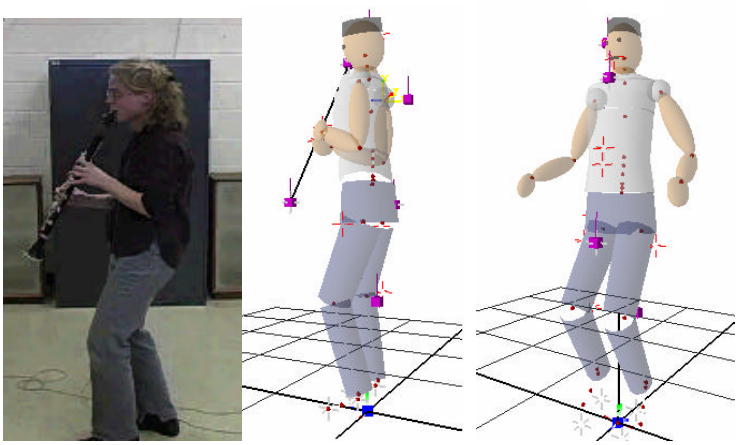


Fig 4. (a) A leg flexion optimization task helps us to obtain believable leg motion
(b) Attracting the body configuration towards this posture prevents self-collision

The key here is that the optimization has the lowest priority, so that it is only satisfied when all other tasks are achieved (see section 2.2.3). For us, this means that the leg flexion never alters the achievement of the other higher priority constraints driven by the sensor data. In other words, if the player is standing upright in the original animation, then the sensors located in her head or on her hip are high, and no flexion occurs since it would prevent the head or hip constraints from reaching their goal. If, on the other hand, the player is slightly crouched (i.e., the legs flexed), then the sensors lay on a lower position, and flexing the knees helps reaching them, so the optimization succeeds.

3.2.5 Computing Collision-Free Arm Postures

As stated earlier, the only sensors attached to the body of the player during her performance where located on the head, the shoulder, the hip and the knee. No information is directly available regarding the position of the arms. However, we can make it up by leveraging our knowledge of the position of the clarinet ends.

The key point to note is that, when a clarinettist performs, as in Figure 4, the position of his hands is fixed around the middle of the instrument, with only the fingers moving to produce the actual notes. With this in mind, the approach we use is:

- Compute the desired position for the hands, with the left hand in the upper part and the right hand in the lower part of the instrument, in order to imitate the real pose adopted by the player. Note that these points must be recomputed at each frame.
- Create an IK constraint for each hand. These constraints are set up so that their goals are the points computed for the left and right hand, respectively.

Self collision avoidance. Although the joint limits for the shoulder are carefully designed in order to avoid unrealistic or unfeasible configurations, the fact is that some motions remain for which self collisions between the player's arms and torso become problematic.

Our first approach to the solution of this problem was to employ a pair of additional positional constraints attached to the elbows of the figure. These constraints were activated when the elbows were colliding with the upper body, and their effect was to repel the arms in a direction orthogonal to the estimated point of collision. This approach does indeed work, but imposes a substantial performance penalty since (a) the relative position of the arms and the torso must be constantly tracked in order to detect collisions and (b) in the event of a self collision, two additional IK constraints need to be created.

A better solution is to extend the optimization task introduced in section 3.2.5. All we have to do is modify the goal posture so that it looks like the one shown in Figure 4b. In this posture, the arms are posed in such a way

that the attraction towards this posture repels them from the torso, thus preventing potential collisions from taking place. Note that this solution to the self collision problem does not penalize the performances, as we already exploit it to facilitate knee flexion.

3.2.6 Enforcing Priority Levels

In the previous sections we have presented a method for recovering clarinet-playing motion from the position of few sensors. This method makes use of as much as nine IK constraints (also called "tasks"). Using a classical IK solver, many of these tasks would conflict at some stage or another, resulting in unsatisfying compromise behaviour.

In our case, we know that the achievement of certain tasks is critical. For instance, the constraints that fix the feet on the ground are among the most important: if they are not fully satisfied, the result is a visually unpleasant flickering of the feet, which would make the whole body motion unrealistic.

To solve such task conflicts, and to ensure that the most important constraints are satisfied all the time, we make use of the priority mechanism provided by our IK engine. This mechanism allows us to assign a different priority to each task. When two or more tasks conflict, those with high priorities are fulfilled first while the lower priority tasks are achieved in the remaining solution space. By assigning priority levels to constraints, we can establish an ordering on the list of tasks, so that those tasks whose completion has a bigger visual impact get a higher priority, and are thus more likely to be achieved.

After experimenting with different schemes, the following task ranking seems to give the best results:

Task	**Priority Rank**
Keep the feet fixed on the ground	1
Place the hands over the clarinet, control head orientation	2
Center of Mass	3
Others	4

Recall that the aforementioned optimization task is always given the lowest possible priority (i.e. rank > 4).

3.2.7 Performance Issues

In an attempt to suit the needs of different users, we provide two modes of operation: online and offline. The former provides real-time performance and can thus be used in interactive applications. The key to achieving real-time performance is that only one IK iteration is performed for each animation frame. This is efficient, but has the drawback that the convergence is not always perfect, some visual artifacts showing up in parts where the original motion was brisk. These artifacts are minimized thanks to the priority scheme sketched in the previous section.

The frame rate achieved on a 2.4 GHz Pentium IV machine with 512 MB RAM is 9-10 frames per second (including a display of the resulting posture). This is not enough to track the sensors at their original sampling frequency (100 Hz), but suffices to provide a relatively smooth motion.

The second operating mode is the offline mode. In this mode, several IK iterations (the actual number is customizable, though fairly good results are obtained with values of 10-20) are performed for each frame. This allows for a better convergence of the IK iteration, which results in a more realistic motion at the cost of a reduced performance. Our benchmarks show that a 90-second segment of clarinetist motion requires about 2 minutes of computation, with 20 convergence iterations per frame.

4. A general-purpose Motion Capture Development Environment

The use of IK solution for motion capture requires precise and complex tuning of the multiple constraints. To optimize this task, we propose a flexible software architecture enabling the prototyping of motion capture solution.

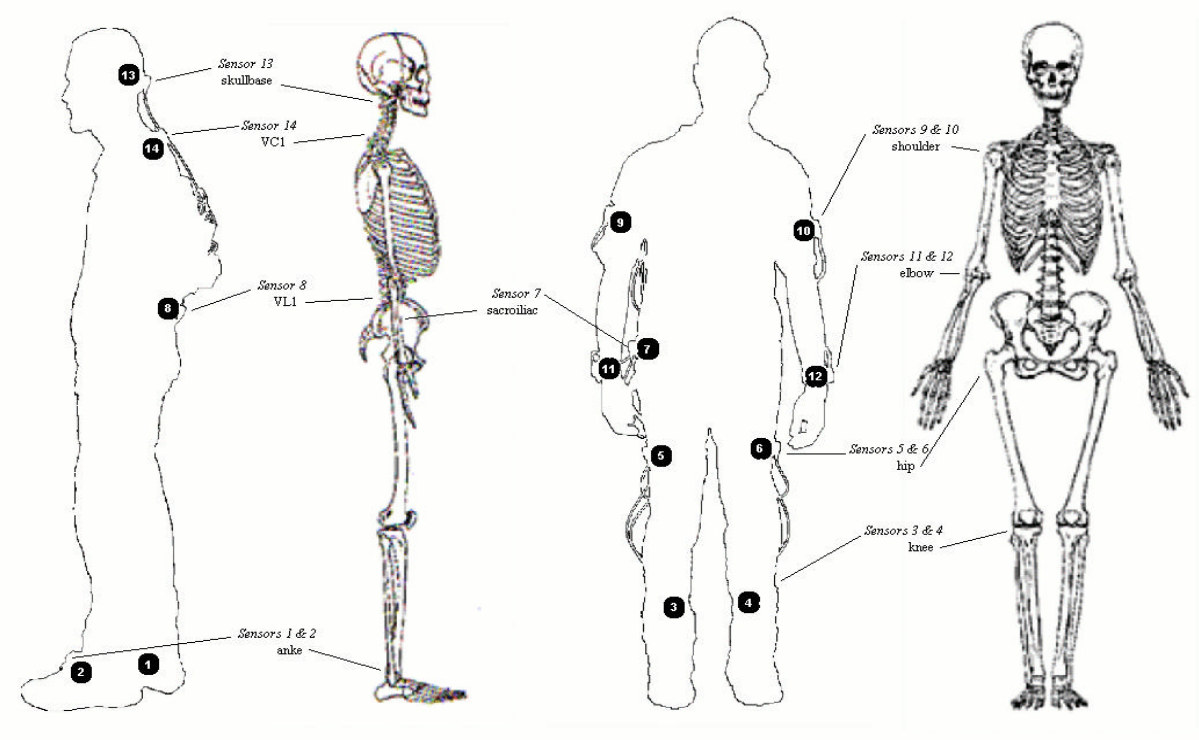


Fig 5. Sensors configuration for full body motion capture and their link to the anatomical joints

4.1. Working with Motion Capture Data

In our general settings for capturing the human body movement, we place up to 14 magnetic sensors from Ascension Technology to measure the orientation of the main body segments (as shown in Fig. 5). We try to fix them as close as possible to the bones (e.g. on the shin bone), but we need to find a compromise for most of them (e.g. sensor 7 on the outside side of the thigh). Once equipped with the sensors, the subject can move freely in the magnetic field area (Motion Star Wireless™ with dual emitter provides an optimal 2.5x2.5m area).

However, setting up the motion capture is very constraining (quite impossible for a single user) and does not allow working repetitively on the same data (which is common need for development). In order to overcome these problems, we developed an emulator of motion capture device on top of our proprietary input device drivers. The sensor emulator program fills a shared-memory bloc with sensors data, and the motion capture client can use them in real time, without knowing the data acquisition technology (Motion Star™, Flock of Birds™, or files). Following the same structure, another program was developed with the aim to provide the video of the recorded gesture together with the Motion Star data at the same time; the MocapSim software already contains a database of more than 50 sequences and can play the reference video at the same time as the motion-capture file. Moreover, the sensors are displayed in the 3D scene next to the animated virtual human. This allows us to notice artifacts in the data recording (e.g. a sensor is not tracked correctly when too close from the magnetic emitter; this way, we can visualize this and make sure the data are correct before using them). The full motion capture test bed is shown in Figure 6.

4.2. Animation Framework

Our animation engine can perform many different actions (from simple look-at up to full procedural walk) and can associate dynamically any action to a Virtual Human (see [16] for details). The resulting animation depends on the importance of the actions (the high importance action is applied exclusively), the scope of the actions (what parts of the body is animated) and the blending between them (during transitions).

In order to perform basic motion-capture animations, a specific “Mocap” action was developed based on the human motion capture algorithm from Molet, Boulic and Thalmann [4]. This purely analytical animation is based on the transfer of sensors orientation to the joints rotations; the Figure 5 shows the correspondence between the sensors and the associated joint. This technique is very efficient but may introduce some distortions (mainly in extreme cases).

The Prioritized Inverse Kinematics is seen as one action. It was integrated in this framework together with a constraint management system. This IK action can work on the full body or a reduced body scope (e.g. arm, leg, back) and let one create/delete/modify the constraints applied to the joints (see section 2.2.1). One or multiple IK update can be performed for each frame to ensure the better convergence of the IK algorithm as already mentioned.

Moreover, thanks to the flexibility of our development platform (presented in details in [17]), we have multiple access points to the animation framework. The mostly used is the embedded Python interpreter, allowing for scripting and fully configuring animations. The computation overhead when using Python is very limited as the commands are only interfaces to the internal and optimized C/C++ implementations.

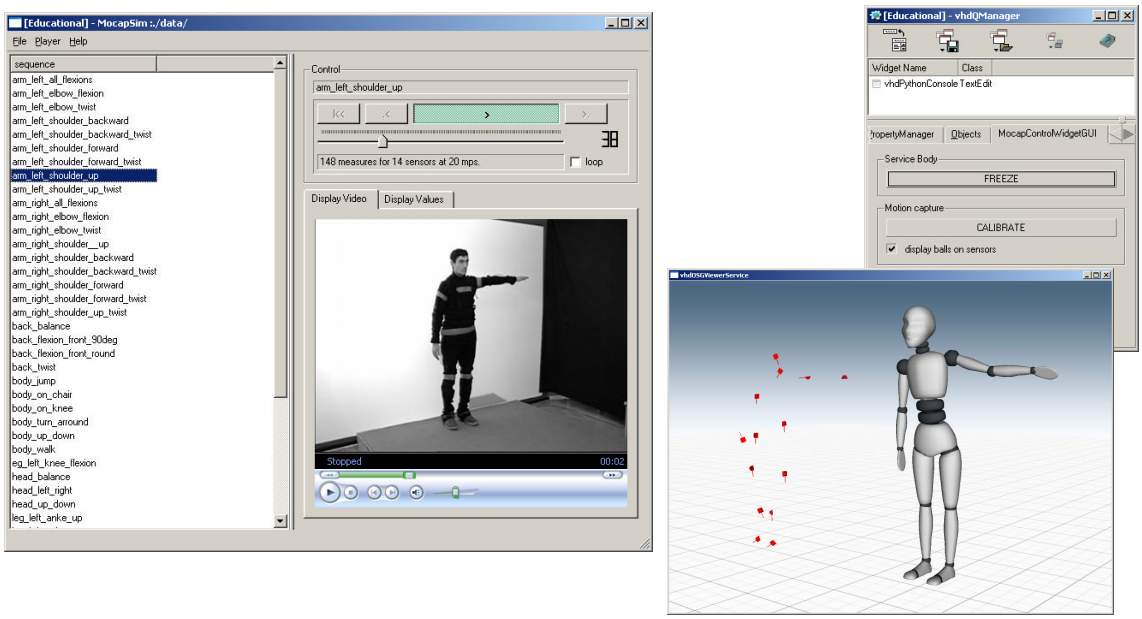


Fig 6. : On the left, MocapSim application: emulation of Ascension Motion-Star™ with its database of motions and videos. On the right, the motion capture client: the 3D rendering (in the window down, the sensors and the resulting motion-captured virtual-human are moving in real time), and its system control GUI (top-right window).

4.3 Integrating an Inverse Kinematics Action

The inverse kinematics action was integrated in the animation framework together with a constraint management system. The *hink* action can work on the full hierarchy or a reduced body scope (e.g. arm, leg, back) and let you create/delete/modify the constraints applied to the joints (see section 2.2.1). One or multiple IK update can be performed each frame to ensure the better convergence of the IK algorithm.

The declaration and the configuration of an inverse kinematics motion capture can easily be set up in our environment using simple Python script. The dynamic effectors constraints are directly taken from the displayed sensors position (and/or orientation). Figure 7 illustrates the configuration of an IK puppetry animation loop applied to a virtual human named Carlo and exploiting the data from one sensor.

5. Conclusions and Future Work

We have presented a system that provides a powerful and flexible way to create IK-based animations of humans. This system has been succesfully employed to reproduce the motion captured performances of several clarinetists.

We have been able to circunvent the problem of having a limited set of sensors, located for additional difficulty in the same side of the body. Thanks to the flexible Python interface, we have succeeded in quickly figuring out the optimal configuration of IK constraints in order to obtain motions that preserve as much as possible the features of the original performances.

Two modes of operation have been implemented and tested. A real-time mode, in which accuracy is sacrified in order to obtain interactive frame rates, and an off-line mode where several IK iterations are performed at each frame so that the results are fully accurate.

A number of issues remain as future work. The main one is the overall performance of the system. As stated earlier, the off-line mode offers more accurate results at the expense of non-interactive frame rates. Our goal is to increase the performance so that the off-line mode becomes on-line on current hardware.

We also plan to investigate how well our system adapts to alternate sources of motion capture data, such as magnetic sensors. In such a case the orientation of each sensor is available together with its position, and we believe that this additional information could be taken into account to obtain more realistic results with a smaller number of IK constraints.

```
agentname = "carlo"
hagentService.hinkCreate( agentName, "ikmocap", "carlo_skeleton", "all")

tPos = hagentService.getPosition("sensor")
constraint_1 = hagentService.hinkCreatePosition( agentname, "ikmocap", "L_WRIST", tPos)
hagentService.hinkSetRecruitingLevel(id,2)

constraint_2 = hagentService.hinkCreateFixPosition( agentname, "ikmocap", "L_ANKLE")
hagentService.hinkSetPriority(id,3)

constraint_3 = hagentService.hinkCreateFixPosition( agentname, "ikmocap", "R_ANKLE")
hagentService.hinkSetPriority(id,3)

hagentService.activateAction(agentname, "ikmocap")
while (TRUE):
        tPos = hagentService.getPosition("sensor")
        hagentService.hinkSetPosition(constraint _1, tPos)
```

Fig 7. Example of one-arm inverse kinematics motion capture configuration. This Python script creates a full-body IK action with one dynamic position constraint on the left hand and glues the feet on the ground (*hagentService* is the animation engine).

Acknowledgements

This work results from discussions initiated at the first ENACTIVE days, organized by Annie Luciani for the 6th EU Framework Program Network of Excellence ENACTIVE . It was partly supported by the Swiss National Science Foundation under the grant 200020-101464. We are grateful to Sébastien Shertenleib and Michal Ponder for maintaining the VHD++ middleware environment. We wish to thank Nicolas Elsig for his 3D model, Sébastien Schertenleib for his participation in motion-capture recordings, and the clarinet performers Marie-Julie Chagnon and Lars Wouters.

References

1. Badler, N. I., Hollick, M. J., Granieri, and J. P., **Real-Time Control of a Virtual Human Using Minimal Sensors,** In Presence Teleoperators and Virtual Environments, Volume 2(1), pp. 82-86, 1993.
2. Molet T., Boulic R. and Thalmann D., **A Real Time Anatomical Converter For Human Motion Capture**, in Boulic R. and Hegron G. (eds) Eurographics workshop on Computer Animation and Simulation 96, ISBN 3-211-828-850, Springer- Verlag Wien, pp.79-94.
3. Hirose M., Deffaux G., Nakagaki Y., **Development of an Effective Motion Capture System Based on Data Fusion and Minimal Use of Sensors**, VRST'96, ACM-SIGGRAPH and ACM-SIGCHI, pp 117-123, July 1996.
4. Molet T., Boulic R., Thalmann, D. (1999) **Human Motion Capture Driven by Orientation Measurements**, Presence Vol.8 (2), MIT Press, pp 187-203
5. Bodenheimer R., Rose, C., Rosenthal, S., Pella, J., **The Process of Motion Capture: Dealing with the Data,** In proceedings of Computer Animation and Simulation, pp. 3-18, September 1997.
6. O'Brien, J., BodenHeimer, R.E., Brostow, G.J., Hodgins, J.K., **Automatic Joint Parameter Estimation from Magnetic Motion Capture Data,** In proceedings of Graphics Interface, pp. 53-60, May 2000.
7. Shin, H. J., Lee, J., Shin, S. Y., Gleicher, M., **Computer puppetry: An importance-based approach,** In ACM Transactions on Graphics, Volume 20, pp. 67-94, 2001.
8. Zordan, V., Van Der Horst, N., **Mapping Optical Motion Capture Data to Skeletal Motion Using a Physical Model,** In proceedings of ACM SIGGRAPH/EUROGRAPHICS Symposium on Computer Animation, pp. 245-250, 2003.
9. Baerlocher, P., Boulic, R., **An Inverse Kinematic Architecture Enforcing an Arbitrary Number of Strict Priority Levels**, The Visual Computer, Springer Verlag, 20(6), 2004
10. Delalande, F, **La gestique de Gould**. In Glenn Gould Pluriel. Louise Courteau, editrice, inc., pp. 85-111, 1988.
11. Davidson, J. **Visual Perception of Performance Manner in the Movements of Solo Musicians**. In Psychology of Music, vol. 21, pp. 103-113, 1993.
12. Wanderley, M. M. **Non-Obvious Performer Gestures in Instrumental Music**. In A. Braffort, R. Gherbi, S. Gibet, J. Richardson, and D. Teil (eds.) Gesture-Based Communication in Human-Computer Interaction. Springer-Verlag, pp. 37 – 48, 1999.
13. Wanderley, M. M. **Quantitative Analysis of Non-obvious Performer Gestures**. In I. Wachsmuth and T. Sowa (eds.) Gesture and Sign Language in Human-Computer Interaction. Springer Verlag, pp. 241-253, 2002.
14. Vines, B., Wanderley, M. M., Krumhansl, C., Nuzzo, R., and Levitin, D. **Performance Gestures of Musicians: What Structural and Emotional Information do they Convey?** In A. Camurri and G. Volpe (eds.) Gesture-Based Communication in Human-Computer Interaction - 5th International Gesture Workshop, GW 2003, Genova, Italy. Springer-Verlag, pp. 468 – 478, 2004.
15. Wanderley, M. M., Vines, B., Middleton, N., McKay, C. and Hatch, W. **Expressive Movements of Clarinetists: Quantification and Musical Considerations.** Submitted for publication (2004).
16. Emering, L., Boulic, R., Molet T., Thalmann, D.,**Versatile Tuning of Humanoid Agent Activity,** Computer Graphics Forum, Vol 19(4), 2000, pp 231-242
17. Ponder, M., Molet, T., Papagiannakis, G., Magnenat-Thalmann, N., Thalmann, D., **VHD++ Development Framework: Towards Extendible, Component Based VR/AR Simulation Engine Featuring Advanced Virtual Character Technologies**, proc. of Computer Graphics International 2003