Real-time Motion Editing for Reaching Tasks Using Multiple Internal Graphs

Christos Mousas
dept. of Informatics
University of Sussex
Brighton, England
C.Mousas@sussex.ac.uk

Paul Newbury
dept. of Informatics
University of Sussex
Brighton, England
P.Newbury@sussex.ac.uk

Abstract— **In this paper we present a motion editing technique for designing animation based on a database of reference postures and the construction of multiple internal graphs in real-time. These internal graphs are generated for each body part that we wish to define in the animation. These graphs drive the body parts that are pre-defined by the user in order to design realistic movements during the animation process. The proposed solution is developed for reaching postures where the locomotion of the character isn't updated.**

Keywords-motion editing; internal graphs; reaching;

I. INTRODUCTION

Real-time character animation is an important aspect for various areas of computer animation. Motion editing, which can be described as the technique where different motions can be rearranged or can be blended for producing a new sequence of motions, is applicable in various applications from video games and films to sports and medicine. This paper proposes a solution for reaching tasks where a character can reach an object with his/her hand. These reaching tasks are quite common in various areas, from video games and virtual worlds to medicine and sports. The real-time aspect of this work is important as there are a lot of applications that require the characters to have the ability to perform tasks in real time without the need for the designers to pre-compute all possible situations. Finally, even though this proposed solution for designing internal graphs focuses on a single aspect, it is proposed that this method can be implemented in various different approaches for designing realistic animated scenes.

The methodology presented here is based on the ability to construct internal graphs for the desired body parts of the character to be animated. The internal graphs are graphs that tend to drive the desired body parts, designing transition paths, in order to generate realistic animations. The construction of these graphs is based in the ability to extract reference postures from the database, which are in-between the start and final position of the character. The transitions between the extracted postures are neither linear nor smooth, due to the limitation of the motion capture database to contain all the desired postures. Additionally, because observed a discontinuity between the extracted postures, this discontinuity assumed as error. Thus, this error positions should be removed in order to generate realistic transitions between the poses. For this reason this research works with signal processing techniques in order to reconstruct the internal graphs, designing smooth transitions from the start to the final posture. The signal processing techniques tries first to remove the extreme values normalizing the 3D graph and then to smooth the resulted values in order to design a smooth graph for the transition. These techniques, which are quite common to image processing, work well at interactive frame rates in this approach.

Finally, after having designed the internal graphs, in order to generate a realistic animation it is necessary to construct a time wrapping function. The time wrapping function is necessary for avoiding transitions between the postures that are linear in time and for that reason a simple method for designing the timings for an animated procedure is proposed. The proposed solution is a dynamic function where each time step for every body part transition is calculated separately based on the analysis of the animation examples in the database. Thus, independent timings for each body part are produced.

The structure of this paper is organized as follows. In Section 2 related work is discussed. Section 3 covers the methodology for the extraction and real time reconstruction of the internal motion graphs. Section 4 presents the results and the limitations of our approach. Conclusions are drawn in Section 5 with discussion of the proposed solution and the potential future work.

II. BACKGROUND

During the past years a significant number of motion editing techniques and algorithms have been developed by the research community in order to be able to design new animated scenes. From the literature [1], [2], [3] it is apparent that the use and editing of motion capture data is the best approach in order to achieve realistic animated scenes. Motion editing gives the ability to construct new motion meeting specific constraints, defined by the user, in order to be reused in new situations and animated scenes. For that reason the research community has focused on different methods for keeping the original quality of the motion data while meeting new constraints.

978-1-4673-1121-2/12/$31.00 ©2012 IEEE

The extraction and generation of animated scenes from a set of motions that are predefined in a motion capture database is not something new. Proposed solutions such as [4] tried to extract and synthesize novel motion sequences from a database of motion capture examples while a statistical model is learning from the captured data enabling realistic synthesis of new motions. In this approach even if different motions can be extracted and synthesized in order to be produced new sequences of animation, by adding the first and the last keyframe of the motion, this solution lack of the ability to specify constraints or complex goals that should be followed by the character.

As [4] and other researchers [3], [5] have focused in statistical analysis in order to extract and compose new animated sequences. In [3] Min et al. applying a statistical analysis technique constructed a deformable motion model that controls the motion's geometric variation. In this approach even if the user can specify the constraints of the character for a specific body part, the synthesized animation cannot generate new motions that may not be represented by the deformable model. Additional, in [5] Lee et al. show an example of preprocessing a motion capture database in order to collect and extend unlabeled sequence of motion data. Even if shows that such a motion database can be flexible the avatar's motion lacks context-dependent details of the recorded motion. Furthermore, another methodology for producing in real-time natural human motion proposed by Safonova et al. [6] where space-time optimization is used giving the ability to edit the motion capture data keeping the physical properties at the same time. This approach of motion generation even if can provide realistic result, doesn't gives the ability to the user to control the character.

Another interesting approach in motion editing, which influences the proposed solution in this research, is the motion graphs [1] proposed by Kovar et al. Motion graphs give the ability to encapsulate connections among the database, but even though motion graphs can generate automatic transitions between original motion data, they cannot match very well specified constraints by the user, because the motions cannot be modified. A relative approach to motion graphs is proposed by Arika et al. [7] where the framework that is designed generates animated scenes by cutting and pasting motion capture data.

Although different approaches have been proposed based on the biomechanics and kinematics, Huang et al. [8] propose a solution where motor controllers with biomechanical limits are applied and Lv et al. [9] proposed a solution that utilizes reaching strategies from biomechanics in order to guarantee the naturalness of the reaching motion while expanding and enriching the flexibility of human behaviour. Other methods, basically those who are relative to kinematics solutions, even if they can produce a desirable result sometimes lack the computational cost or the final result is not the desirable one.

Finally, another solution for character animation that seems to share similar characteristics with the approach proposed in this paper is that by Wang et al. [10] and is called layered interpolation. In this approach the motion capture data is divided in three layers and final natural posture is further achieved by compensating the motion correlation between limbs.

III. METHODOLOGY

This section presents the methodology used in order to reconstruct the animation. The methodology is separated in the following steps:

- **Specify user constraint**: The user has the ability to specify the position of the reaching object (final posture) that the character should reach. This is based on the ability to place an object in the 3D space.
- **Register reference postures**: The registration process refers to the ability to extract the relevant postures from the database.
- **Reconstruct graphs**: The process where the extracted values are processed in order to design the internal graphs.
- **Generate animation**: The generation process of the time-wrapping function that animates the character with natural looking movements.

The implementation of this approach is presented in the figure below.

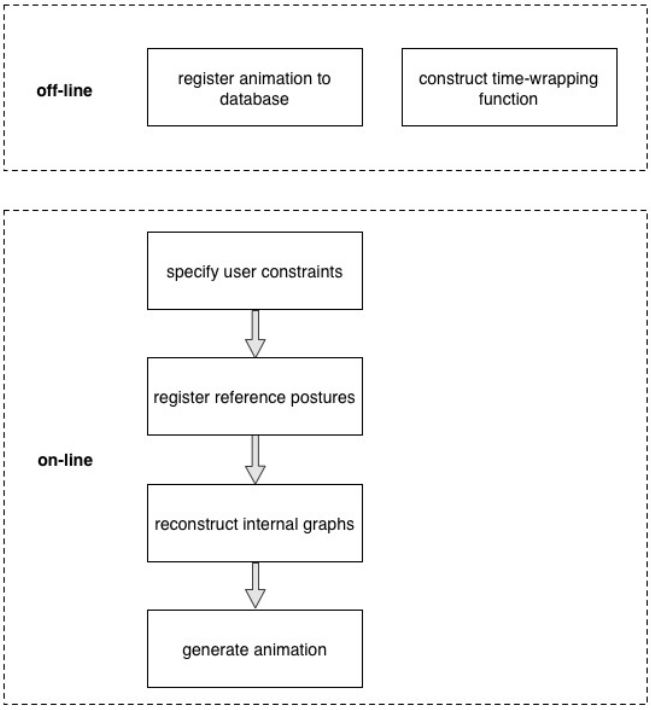

Figure 1. The system architecture consists of off-line and on-line computations. The registration of motion capture data and the time-wrapping function are implemented off-line. The user constraints, the internal graphs reconstruction and the final animations are generated on-line.

A. Postures Extraction

In this approach, the first step in order to design an animated scene is the ability to extract reference postures based on the examples that are stored in the database. The

motion capture database [11] that used is provided by the Carnegie Mellon University and consist of 2.500 posture examples. This motion capture database was chosen because is widely used by the research community. Thus, in order to extract the most relevant examples based on the constraints that the user has specified (the position of the reaching task in the 3D space) the Euclidian distance between the straight line that joins the start and the final position of the character's right hand with the most relevant for each distance step is used. From the literature [2], [3] the most common way to extract and handle information is the reduction of dimensionality using the Principal Component Analysis (PCA) method that can then reduce the retrieval time of motion capture data. In this research reduction of dimensionality was not required as the extracted 3D data can be edited fast enough in real time.

In the registration process, because it is intented to have the complete control of the 3D character, the related postures, positions and orientations, for each joint of the character (except feet) are registered. The technique does not register the position and orientation of feet because of observed foot sliding in the transition process at the in-between postures. This ensures that feet will stay static keeping their initial position and orientation. Thus, we conclude to have the registered data as $\Psi_n^j = [P_1^j, R_1^j, \dots P_n^j, R_n^j]$. The registered data Ψ_n^j is presented as an array of positions and rotations for each joint j with n reference postures length. Thus Ψ_n^j is the character posture for joint j in the n-th example, P_n^j is the position of joint j in the n-th example frame and R_n^j is the rotation of joint j in the n-th example. The position for each body part is a Vector3 position and the rotation is in Euler angles. All relevant postures are stored in our database in order based on the distance step that appeared. Furthermore, because the start and final position of the hand do not exist in the database, as they are specified by the user, they were added respectively as first and last elements by the user constraint. Finally, it is necessary to be mentioned that the character consists of 15 joints.

B. Internal Graphs Reconstruction

The basic part of this approach is based in the ability to reconstruct in real time the internal graphs. After having registered the related positions of each body part, as described in the previous section, signal processing techniques are used in order to reconstruct and normalise the graph.

In order to reconstruct the motion graph an iterative method using a smooth algorithm is designed. This approach, even though it seems to give a satisfactory result, reduces dramatically the frame rate of the application due to the requirement for multiple iterations. For this reason an edge detection algorithm is initially used, the Laplacian of Gausian (LoG) detector, in order to detect the changes of the graph. This detector is quite common in image processing and is used in order to detect the light intensity of an image. As this involves 3D signal processing, the kernel for the 3D Gaussian operator can be defined as:

$$G_{(x,y,z)=}\frac{1}{(\sigma\sqrt{2\pi})^3}e^{-\frac{x^2+y^2+z^2}{2\sigma^2}} \quad (1)$$

Thus, the kernel for the 3D LoG operator is described as:

$$LoG_{(x,y,z)} = \nabla^2\left(I_{(x,y,z)} * G_{(x,y,z)}\right) \quad (2)$$

where $I_{(x,y,z)}$ is the 3D graph and $\nabla^2 = \frac{\theta^2}{\theta x^2} + \frac{\theta^2}{\theta y^2} + \frac{\theta^2}{\theta z^2}$.

The LoG edge detector is responsible for giving those areas in the graph that have discontinuity. Thus, having detected the discontinuity of the graph those areas can be normalized by using thresholding segmentation. It was chosen to work with the thresholding technique as it removes the noisy edges (Figure 3) and can be implemented in real-time without reducing dramatically the frame rate of the application.

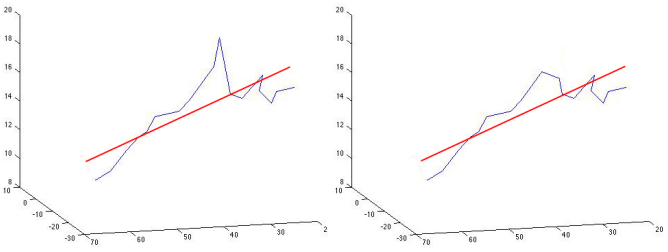

Figure 2. Graph before (left) and after (right) thresholding the extreme values

The thresholded values are calculated by the division between the mean value and the $LoG_{(P_k,R_k)}$ edge detection for the specific P_k and R_k position and orientation. Thus the threshold segmentation for the P_k and R_k edge position or orientations is formulated as:

$$(P_k, R_k) = (P_k, R_k) * \frac{\frac{1}{n}\sum_{i=1}^{n}(P_i, R_i)}{LoG_{(P_k,R_k)}} \quad (3)$$

Even if the thresholding segmentation can normalise the graphs, it is observed that the result isn't adequately continuous and smooth. Finally, because the requirement is to have as smooth as possible animation transition between extracted postures the moving average algorithm is applied. This algorithm can ensure a normalized continuity of the reconstructed graph.

$$(P_k, R_k)_s = \sum_{i=-n}^{i=n}\frac{(P_{k+1}, R_{k+1})}{2n+1} \quad (4)$$

Where $(P_k, R_k)_s$ are the smoothed points of positions and orientation for each body part of the average of an odd number of consecutive $2n+1$ (n=1, 2, 3,...) points of the raw data: $P_{k-n}, P_{k-n+1}, \dots, P_{k-1}, P_k, P_{k+1}, \dots, P_{k+n-1}, P_{k+n}$. It was chosen to use both a noise reduction algorithm and a smoothing algorithm because even if discontinuity of the input data is removed this doesn't ensure that the output data will be normalized and smooth. Thus, while applying the smooth filter a clear and continuous graph is constructed as figure 3 shows.

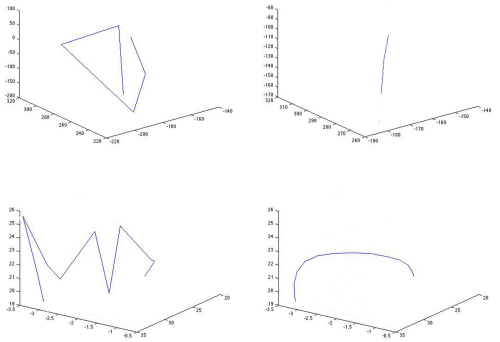

Figure 3. (upper row) The registered rotations of shoulder (left) and the reconstructed approach (right). (lower row) The registered postures of right hand (left) and the reconstructed approach (right).

Finally, in order to ensure that the thresholding and moving average algorithm will not reduce the size of the registered data, and will not remove or edit the start and final position of the desired animation, the original position and orientation of the first and last position for each body part were added at the array of postured respectively. Thus, can be ensured that the animated character will follow the constraints specified by the user.

C. Generating Animation

Once the internal graphs have been registered and reconstructed the generation of the animation is the next step. The user has the ability to specify the constraint, which is the reaching position of the character's hand. Thus, the generation of the animation can be created. At this point a smooth and linear in time animation would look unrealistic, thus a time wrapping function is constructed as described in the next sub-section.

1) Modeling Time Wrapping Function

In order to design a natural looking animation it is necessary to design a time-wrapping function. Min et al. [3] propose a solution that transforms the constraint time wrapping function to unconstrained function applying statistical analysis in order to construct a deformable timing model. Even if this approach can give a natural in timing animated sequence it is not as dynamic as it would be desired. For that reason, as the user has the ability to place the number of frames that are involved in the animation, a dynamic function is computed where each time step is calculated separately. Therefore, a time wrapping function is calculated off-line while analysing the existing animations in the database. Thus, the existing animated scenes for each body part were decomposed first. This gives the ability to analyse and construct a dynamic model of the movement of each body part. In order to understand how the body parts move, first the keyframes of each animated part were mapped linearly in order to have the same duration. The average of time that each body part moves against the distance from the starting position is then calculated. In this way, time-wrapping graphs that contain the distance between starting pose and time frames were constructed. The following figure 5 shows an example of right hand position and abdomen rotation against time.

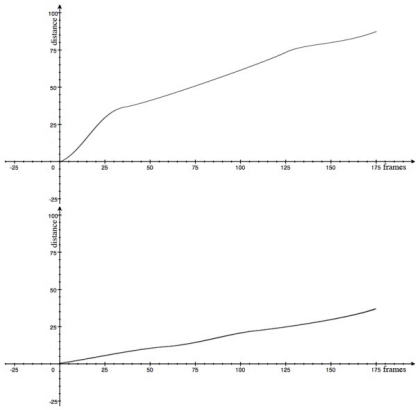

Figure 4. Time-wrapping construction model. The distance of position for the right hand (upper), and the distance of rotation of the abdomen (lower).

The distances of positions and rotations calculated as $d_{P^J} = \left|P_1^j - P_n^j\right|$ and $d_{R^j} = \left|R_1^j - R_n^j\right|$ where P_1^j, R_1^j is the starting position and rotation of joint j in the first frame of an the animation sequence and P_n^j, R_n^j is the position and rotation of joint j in the n-th frame.

In order to recompose this time wrapping function to a new sequence of animation with n number of frames, defined by the user, the number of frames of the new sequence to the existing time function for each body part as well as the distance of the starting position and rotation to the final position and rotations were mapped linearly first. Thus, for each body part the following equation needs to be solved:

$$T_n^j = c * \frac{t_{nextFrame}^j}{t_{total}} \tag{5}$$

where $c = \frac{d_{reg}}{d_{ui}}$ is the mapping constant that results from the division between the d_{reg} which is the registered distance between the start and the current frame position and the d_{ui} is the distance of user input between the first and last position. At the second part of equation (5) t_{total} is the total number of frames that the user has define and $t_{nextFrame}^j$ is the number of frames from current to the next pose transition.

D. Algorithm

Having analysed the whole procedure of how to design realistic animated characters based in related reference postures while reconstructing multiple internal graphs, we conclude in the implementation of this algorithm. The proposed algorithm can run in interactive frame-rates and its implementation seems to be a good approach for designing realistic animations.

Algorithm 1 Design Internal Graphs

```
1: for each j joint in Joints
2:     register reference postures
3:     if edge is detected using (2)
4:         normalize edge using (3)
5:     #end if
6:     smooth reference P_k and R_k using (4)
7: #end for
8: return transition points P_k and R_k
```

IV. EVALUATION

In order to evaluate this approach an application has been developed in the Unity3D game engine. In this application the character tries to reach with the right hand a specific area in the 3D space while the locomotion is not updated. Even if the starting position of the user is changed, the character position was chosen to keep standing with hands stretched downwards as a starting position. The evaluation process tests were performed on a system with an Intel i7 processor at 2.2 GHz with 8GB Ram. In the Unity3D game engine maximum performance was chosen, which gives as a maximum frame rate 250fps, of the system in order to be tested. Thus, with a 3D mesh of 2000 polygons this implementation needs 50ms for the registration process and the real-time implementation of the algorithm for the animation process doesn't exceeds the 140 fps.

In order to ensure that this approach will not only work in the situation described earlier, where the user is trying to reach an object in the 3D space, various other animations are generated. Examples for reaching tasks where more than one constraint is specified, like a reaching task where the user should first stoop in order to grab the object and then to place it in a position raising his hand (figure 6). Having designed more complex tasks such as this, it can be concluded that the proposed solution can work very well in complex situations with more than one constraint specified by the user.

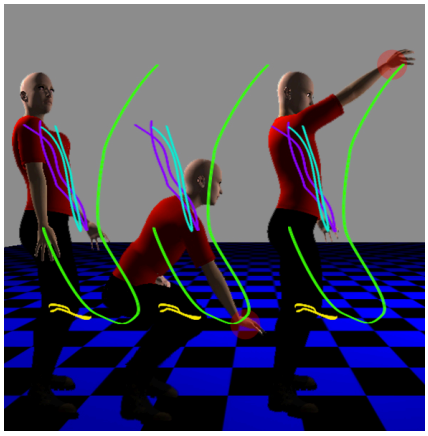

Figure 5. Generated graphs for two user specified constraints for reaching. Right hand graph (green line), shoulder graph (purple line), knee graph (yellow) and abdomen graph (cyan).

A. Limitations

One basic limitation of the proposed system is the ability to reconstruct accurate graphs while the extracted reference postures are extreme values. This is because at the database there are not enough postures and the extracted values are far away from the target. For that reason other techniques that are more suitable with the normalization process should be implemented. Another limitation that the current implementation has, is the ability to design new animated scenes where the reference postures are not the most suitable, such as a reaching task where the object is behind the character. Thus, even if it is possible to extract reference postures and reconstruct the graph the final result is not as realistic as wished due to the lack of appropriate reference postures in the database.

V. CONCLUSION AND FUTURE WORK

In this paper a method that generates character animation for reaching tasks while the locomotion is not updated is proposed. An application is developed in the Unity3D game engine in order to ensure that this approach can be implemented and run at interactive frame rates. In this approach each body part moves separately from another, and this research proposes a simple method to compose a time-wrapping function in order to enhance the realistic representation of the generated animation. Finally, problems that observed in literature such as the foot sliding does not appeared, due to with multiple internal graphs the animated body parts should be defined and registered.

A. Future Work

In future work we would like to generalize and extend this approach by producing examples of animated scenes not only for reaching tasks but also for various different tasks such as walking jumps and golf swing motions to different situations and environments. Finally, due to this approach handling each body part separately in order to design realistic animations, we would like to apply this to more complex scenarios where more that one user specified constrain could take place. For example a complex character movement where the character can reach with the hand an object in the 3D space while walking and jumping in order to avoid an obstacle.

REFERENCES

[1] L. Kovar, M. Gleicher, and F. Pighin, "Motion graphs", in Proceedings of SIGGRAPH 2002.

[2] S. R. Carvalho, R. Boulic, and D. Thalmann, "Motion Pattern Encapsulation for Data-Driven Constraint-Based Motion Editing", Motion in games, Second International Workshop, MIG 2009, Zeist, The Netherlands, November, 2009.

[3] J. Min, Y.-L. Chen, and J. Chai, "Interactive generation of human animation with deformable motion models" ACM Trans. Graph. 29, 1, Article 9, December 2009.

[4] K.M. Tanco, and A. Hilton, "Realistic synthesis of novel human movements from a database of motion capture examples", in Proceedings of the IEEE Workshop of Human Motion HUMO, Austin, Texas, USA, December 2000.

[5] J. Lee, J. Chai, P. Reitsma, J. K. Hodgins, and N. Pollard, "Interactive control of avatars animated with human motion data", ACM Transactions on Graphics (SIGGRAPH 2002) 21, 1 (July).

[6] A. Safonova, J. K. Hodgins, and N. S. Pollard, "Synthesizing Physically Realistic Human Motion in Low Dimensional, Behaviour- Specific Spaces", in Proceeding of SIGGRAPH 2004

[7] O. Arikan, and D. A. Forsyth, "Interactive Motion Generation from Examples" in SIGGRAPH '02 Proceedings of the 29th annual conference on Computer graphics and interactive techniques, San Antonio, Texas, USA, July 2002

[8] W. Huang, M. Kapadia, and D. Terzopoulos, "Fullbody hybrid motor control for reaching", in Proceedings of the Third international conference on Motion in games, MIG 2010, Utrecht, The Netherlands, November 2010

[9] P. Lv, M. Zhang, M. Xu, H. Li, P. Zhu, and Z. Pan, "Biomechanics-based reaching optimization", Vis. Comput. 27 (June, 2011), 613–621.

[10] J. Wang and S. Xia, "Layered interpolation for interactive avatar control" in Proceedings of the 10th International Conference on Virtual Reality Continuum and Its Applications in Industry, VRCAI 2011, Hong Kong, China, December 2011

[11] Carnegie Mellon University (CMU) Graphics Lab "Motion Capture Database", http://mocap.cs.cmu.edu/