EUROGRAPHICS '98 / N. Ferreira and M. Göbel
(Guest Editors)



# Motion Reparametrization

Fernando Wagner da Silva[1,2] Luiz Velho[1] Jonas Gomes[1]

[1]IMPA–Instituto de Matemática Pura e Aplicada
Estrada Dona Castorina, 110, 22460-320 Rio de Janeiro, RJ, Brazil
{nando,lvelho,jonas}@visgraf.impa.br

[2]LCG - Laboratório de Computação Gráfica, COPPE - Sistemas / UFRJ
21945-970, Rio de Janeiro, RJ, Brazil, Caixa Postal 68511

**Abstract**

*This paper presents an algorithm for reparametrization of 1D signals in the discrete domain, according to a user-defined velocity function. The application outlined in this work is the reparametrization of joint curves obtained from human motion captured data. Using the method described here we were able to create effects such as slow-motion and accelerated-time.*

***keywords:*** *motion capture, computer animation, motion processing, digital signal processing.*

## 1. Introduction

In recent years, the use of reparametrization techniques for signal and curve reshaping has become very common. In sound processing, these techniques are used to modify the "pitch" of the sound in certain time intervals, by expanding and/or compressing the signal according to some resampling heuristics. In curve design, arc length based techniques are used to model general parametric curves [1].

In time-dependent applications, such as animation, sound and video processing, reparametrization techniques are even more important. Slow-motion and accelerated-time effects can be achieved by altering specific time-dependent parameters of the animated objects. In keyframe-based animation systems, parameters such as the velocity and acceleration of a moving object can be altered by reparametrizing its corresponding curves [2]. In professional video editing systems, the effect of slow-motion or accelerated-time can be achieved by changing the playback speed of a video stream according to some velocity function. Changing the playback speed in audio does not work, because the frequency content changes and drastically affects the audio perception. In fact, sound is not a temporal signal but a frequency$\times$time signal. A reparametrization of audio, without changing the frequency content, has been described in [3].

Another time-dependent application is motion capture. In this animation technique, the movement of a real object is reconstructed by recording its position and orientation at specific instants of time using special hardware. In the case of human motion capture, the global position and orientation of several joints of an actor are recorded, generating a set of 1D signals which are the sampled values of the motion of actors' joints. These sampled motion curves are processed and then mapped onto a skeleton hierarchy which will drive a virtual actor in the computer [4].

It is straightforward to think about the application of reparametrization techniques to motion capture data. However, most of current techniques require a continuous (parametric) representation of the motion curve. From the discrete point of view, there is no information about the continuous description of the motion curve: it is represented by its set of samples. Consequently, a possible solution for the reparametrization problem can be achieved by resampling: we reconstruct the motion curve from the samples, reparametrize the constructed curve, and sample it again. Spline-based interpolation techniques could be used to reconstruct the motion curve. Unfortunately, this would be very time and memory consuming, since a typical motion curve for one single joint consists of over 1000 samples.

In this work, we present an algorithm for motion reparametrization which works mostly on the discrete domain of the captured motion curves, performing a local resampling where necessary. According to a user-defined ve-

locity function – traced over the temporal description of the original signal – the algorithm computes a discrete integral which is used to locate those regions that need to be expanded or compressed in the original signal. The practical application of this method is the generation of slow-motion and accelerated-time effects in motion capture data, without the need to change the playback speed during the visualization process. Moreover, this algorithm can be easily adapted to work with different types of signals.

## 2. The Algorithm

A continuous curve $F(t)$ may be reparametrized using a function $p(t)$ resulting in $\widetilde{F} = F(p(t))$ (see Figure 1).

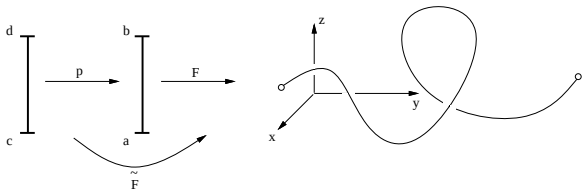


**Figure 1:** *Parametrization of a continuous curve.*

Most of existing applications were designed to receive as input signals that were sampled at uniform rates, and during the visualization process these applications usually assume a constant playback speed. This happens because most of the reconstruction devices need uniform sampling. Therefore, to generate a velocity-changing effect we need to resample the signal in some way so that this effect is noticeable when the signal is played at uniform speed.

This is illustrated in Figure 2. The parametrization on the left makes no change on the sampling pattern. On the other hand, the parametrization on the right transforms the uniform sampling of the interval $[c,d]$ into a non-uniform pattern of the interval $[a,b]$.

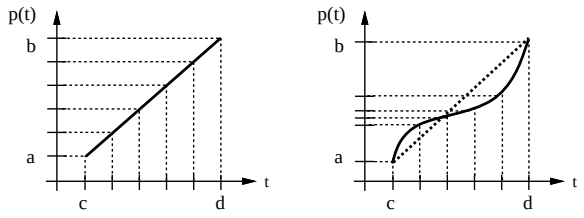


**Figure 2:** *Two different parametrization functions.*

Note that a parametrization does not affect the shape of the curve, but merely changes the curve speed as indicated by the equation bellow

$$\widetilde{F} = p'(t).F'(p(t)) \qquad (1)$$

As a result, all time-dependent parameters of the curve are affected by this transformation. Equation (1) shows that a more intuitive interface should allow the user to change the derivative function $p'(t)$ of the reparametrization function $p(t)$, instead of $p(t)$ itself. From the knowledge of $p'(t)$, we obtain $p(t)$ using integration.

The approach used in our algorithm computes the integral along with the local resampling, using a velocity×time function to set the new sampling pattern of the signal. This function is defined by the user and is used to compute the discrete integral over the original signal. This process is exemplified in Figure 3. In this figure, a polygonal curve was used to define the velocity behavior of the new signal over the temporal description of the original signal.

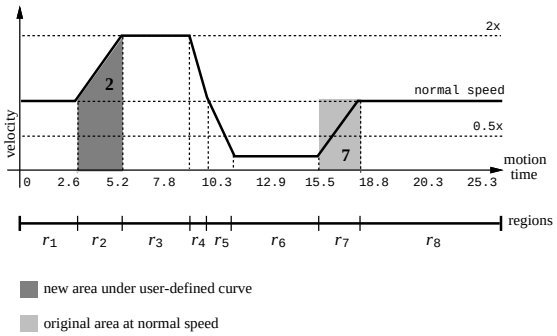


**Figure 3:** *A velocity×time function.*

In the first step of the method, several regions $r_i$ of the user-defined velocity curve are identified by the algorithm. These regions will define segments of monotonic increase or decrease in the velocity curve. In Figure 3 two of these regions, marked as 2 and 7, are outlined. Each one of these regions comprises two areas: $\Delta_c$, defined by the original (constant) normal-speed function; and $\Delta_n$, defined by the new velocity function. The ratio $\Delta_r = \Delta_c/\Delta_n$ between these areas determines, for each region, how the new signal must be expanded ($\Delta_r > 1$) or compressed ($\Delta_r < 1$) with respect to the original signal. The case $\Delta_r = 1$ means that an identity is assumed between the original signal and the new signal (i.e., no compression or expansion).

The expansion/compression behavior of the method can be better understood by using the example of Figure 3. In region 2, the trapezoid defined by the velocity function indicates that the new signal starts at uniform (constant) speed and then accelerates uniformly (constant acceleration) until reaches the double speed. In this case, the corresponding region in the new signal will comprise less samples than the original one (i.e., a compression, or accelerated-time effect). A similar approach is used in region 7, but now there will be more samples in the new signal (i.e., an expansion, or slow-motion effect).

The second step of the algorithm uses, for each region $r_i$,

the ratio $\Delta_{r_i}$ between the calculated areas to compute the new number of samples in that region. The sum of these computed samples for each region will give the total number of samples of the new signal. Finally, for each region, the calculated new number of samples is used to generate an index relating the positions of the new samples with respect to the old signal. When one or more positions of this new index fall between two samples of the original signal, interpolation is used. On the other hand, when one or more samples of the original signal are between two positions of the new index, their values are weighted to generate a sample in the new signal.

In our current implementation, the velocity function is defined by a piecewise linear curve. This was done to facilitate the user-interface interaction and the implementation of the algorithm itself. However, this algorithm can be extended to work with velocity functions defined by curves of higher degrees. Further details of the algorithm can be found in [5].

## 3. An Example

The example presented in this section shows the application of our method to a joint motion curve obtained from motion captured data. The algorithm was implemented using the graphical interface facilities of *MC Animator* [6], a motion capture based animation system which implements several techniques for motion processing, analysis and reuse.

Figure 4 shows the original joint motion curve. Using the temporal description of this curve, a velocity function was defined by the user in a special interface object (Figure 5). This object allows an interactive construction of the curve and imposes some constraints to ensure temporal coherence. The velocity function described in this example is the same that was used in Figure 3.

Figure 6 shows the result of the reparametrization using the algorithm described in this work. Note the compression and expansion of specific parts of the reparametrized signal with respect to the original one (the corresponding regions $r_i$ were labeled to facilitate this comparison). Also, note that the length of the new signal, which has more samples than the original signal, has changed, but its overall characteristics remain the same.

## Acknowledgments

The authors would like to thank Viewpoint Datalabs, Inc. and Biovision, Inc. for access to motion capture data, and to the Brazilian Council for Scientific and Technological development (CNPq) for the financial support. This research has been developed in the laboratory of VISGRAF project at IMPA and at LCG/UFRJ, as part of the Master programme of the first author [7]. The VISGRAF project is sponsored by CNPq, FAPERJ, FINEP and IBM Brasil.

## References

1. Farin, G., *Curves and Surfaces for CAGD*. Academic Press, 1989.
2. Doris H. U. Kochanek and Richard H. Bartels, “Interpolating Splines with Local Tension, Continuity, and Bias Control”. In *Computer Graphics (SIGGRAPH ’84 Proceedings)*, 18, pp. 33-41, july 1984.
3. Goldenstein, S. and Gomes, J., “Time Warping of Audio Signals”. Preprint, IMPA, 1998.
4. Silva, F., Velho, L., Cavalcanti, P. and Gomes, J., “An Architecture for Motion Capture Based Animation”. In *Proceedings of SIBGRAPI’97, X Brazilian Symposium of Computer Graphics and Image Processing*, pp. 49-56, october 1997.
5. Motion Capture - Research website, VISGRAF Lab., *http://www.visgraf.impa.br/Projects/mcapture*
6. Silva, F., Velho, L., Cavalcanti, P. and Gomes, J., “A New Interface Paradigm for Motion Capture Based Animation Systems”. In *Proceedings of the 8th EUROGRAPHICS Workshop on Computer Animation and Simulation*, pp. 19-36, september 1997.
7. Silva, F., *Um Sistema de Animação Baseado em Movimento Capturado*. Master Thesis, Federal University of Rio de Janeiro - COPPE/UFRJ, march 1998.

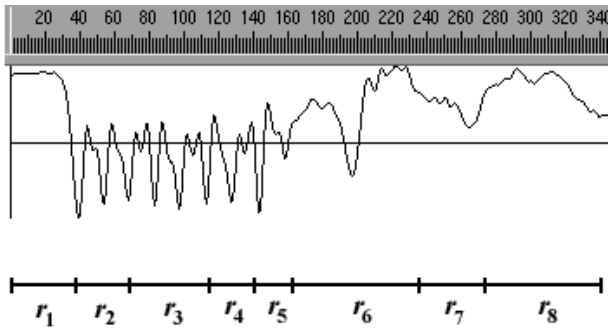


**Figure 4:** *Original motion curve. The regions $r_i$ defined by the velocity curve are outlined at the bottom of the image.*

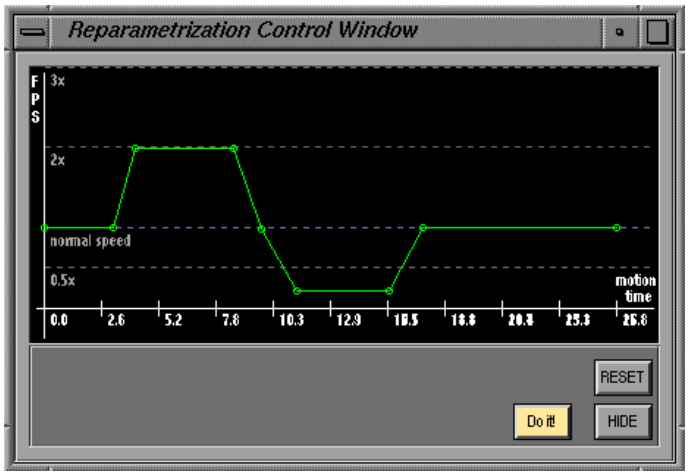


**Figure 5:** *User-defined velocity function.*

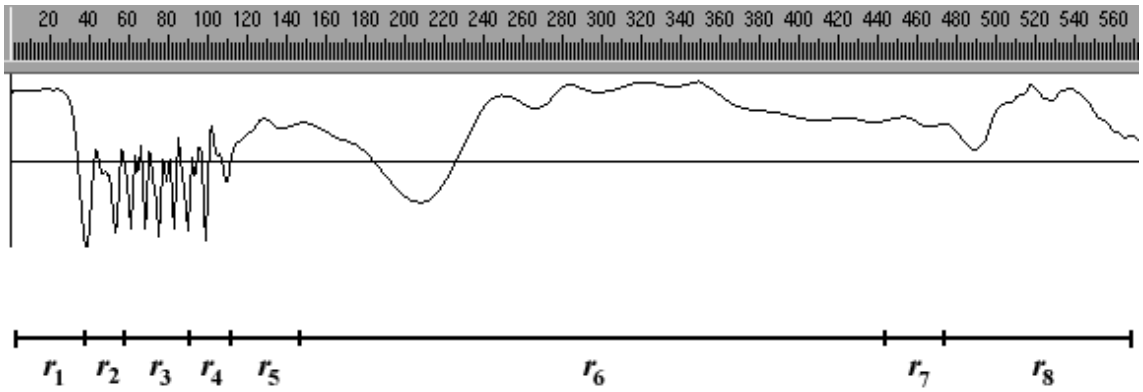


**Figure 6:** *Reparametrization of the motion curve using the algorithm.*