INTERACTIVE EVOLUTIONARY DESIGN OF MOTION VARIANTS

Jonathan Eisenmann
Computer Science & Engineering, The Ohio State University, 2015 Neil Ave, Columbus, OH, U.S.A.

Matthew Lewis
Advanced Computing Center for the Arts & Design, The Ohio State University, 1224 Kinnear Rd, Columbus, OH, U.S.A.

Bryan Cline
Computer Science & Engineering, The Ohio State University, 2015 Neil Ave, Columbus, OH, U.S.A.

Keywords: Animation, Evolutionary Graphics, Interaction Techniques.

Abstract: This paper presents an intuitive method for novice users to interactively design custom populations of stylized, heterogeneous motion, from one input motion clip, thus allowing the user to amplify an existing database of motions. We allow the user to set up lattice deformers which are used by a genetic algorithm to manipulate the animation channels of the input motion and create new motion variations. Our interactive evolutionary design environment allows the user to traverse the available space of possible motions, presents the user with populations of motion, and gradually converges to a satisfactory set of solutions. Each generated motion sequence can undergo a motion filtering process subject to user-specified, high-level metrics to produce a result crafted to fit the designer's interest.

1 INTRODUCTION

The human visual system possesses acute pattern recognition abilities which can expose unnatural qualities in synthetic crowd animation if insufficient motion variation is present. Often times, crowd animations will also need to display particular meaningful expressive qualities in order to set the mood for a scene. The motions of crowd agents must not only be diverse and expressive, but they also often need to appear visually coherent. Therefore a certain amount of similarity between some of the individuals in a crowd is desirable.

A common method for achieving variation within a crowd consists of providing agents with a broad library of motions to choose from, using behavioral rules and blending between the motions as the agents transition from one action to another. Behavioral selection of motion clips from a library should achieve good results if the library is large enough to provide satisfactory variation. However, this method is not always feasible for a small studio without access to an extensive library of motion clips or for a crowd animation with novel motion that cannot be motion captured. Furthermore, it does not provide an easy way to tune the crowd motion to fit a particular style.

We have developed an interactive evolutionary design system that will help designers create a diverse set of crowd motions that belong to a few families of expressive motion and can be tailored to fit the designer's preferences. Our system generates a range of motion variants using a single input motion. The input can be in the form of keyframe animation sequences and motion capture data should work as well. Therefore our method can not only be easily used to amplify a small database of motion clips for crowd animation but also as an aid in the ideation process in motion development for a character.

The system alters the input animation via user-defined free form deformations which reshape the animation channels specifying the motion. Attributes that control deformations act as genes within our design environment's genetic algorithm. The resulting population of motions can then be filtered using several techniques described in this paper that enforce a set of constraints. Physical constraints can be utilized to define the physical properties of each motion. The designer views these filtered motions at each generation to interactively determine the fitness of the crowd members. Through continued interaction, the

Eisenmann J., Lewis M. and Cline B. (2009).
INTERACTIVE EVOLUTIONARY DESIGN OF MOTION VARIANTS.
In *Proceedings of the International Joint Conference on Computational Intelligence*, pages 127-134
DOI: 10.5220/0002320301270134

designer can guide the population to one or more areas of the space of solutions which portray an interesting and desirable set of emotions or expressions. Interesting motions can be saved in a library for use later in the design process or as part of the final set of motions.

Our system also has the benefit of providing the designer with not just static samples from a space of possibilities, but also a neighborhood of similarity around each motion that can be used to tweak a single motion if desired. This flexibility is an improvement over a static library of motion capture clips in the same way that images produced by a procedural shader are more flexible than a set of scanned images. Motions generated by the system can be used to inspire animators with new movement ideas or stored in a database for generating crowd animation.

2 RELATED WORK

Our approach draws from previous work in areas ranging from genetic algorithms to motion variation techniques and crowd design methods.

Interactive evolutionary design, unlike many genetic algorithm applications, lets the user interactively determine the fitness of the evolved solutions. Sims applied this technique to a wide array of application areas, including procedural models and textures (Sims, 1993). In the application area of character animation Lim and Thalmann presented an intuitive interface for searching through a design space by selecting one from a pair of options in "tournament" style (Lim and Thalmann, 2000). Ventrella introduced a physically-based animation and creature design system that allows users to interactively determine fitness of individuals explicitly by altering fitness functions as well as implicitly by selecting individuals of high fitness. His motion model consisted of single DOF joints activated by parameterized sinusoidal functions for the sake of fast interactivity (Ventrella, 1995). We extend his novel work in this area by introducing a flexible framework for generating motion which allows for a wide range of expressive motion.

There are a wide variety of ways to generate motion variation. Sung proposes a method for synthesizing motion clips for crowd animation using motion graphs (Sung, 2007). Amaya et al. use signal processing techniques to embed varying emotions into a neutral motion clip (Amaya et al., 1996). Chi et al. introduce a system for modifying human motion using a Laban inspired effort and shape parameterization (Chi et al., 2000). Neff and Fiume introduce a method that uses both low-level and composite properties to edit character motion iteratively and interactively (Neff and Fiume, 2005). Wang et al. present a motion signal filter method for making a motion more animated or cartoon-like (Wang et al., 2006). Gleicher has provided an extensive survey of constraint-based motion editing techniques with particular attention paid to "per-frame inverse kinematics plus filtering" techniques (Gleicher, 2001).

Crowd design has typically centered on the problems of navigation and behavioral patterns. For example, Kwon et al. present a graph-based approach for intelligently deforming group motion trajectories (Kwon et al., 2008). In addition, Treuille et al. introduce a particle-based solution for crowd navigation without the use of agent-based dynamics (Treuille et al., 2006). Li and Wang have used interactive evolutionary design to tune the parameters in a virtual force based system (Li and Wang, 2007). Musse and Thalmann created a real-time interactive system with three methods for editing crowd behaviors (Musse and Thalmann, 2001). Sung et al. presented an efficient statistics-based scalable model for goal-directed crowd behavior that can satisfy duration, orientation, position, and pose constraints for individuals within the crowd (Sung et al., 2005). Similarly, the crowd simulation middleware in use today focuses mainly on navigation and behavioral rules while relying on extensive motion libraries to provide variation of motion (Massive Software, 2009). Time warping and blending between actions selected by behavior models are a popular means of generating motion variation in crowds.

Our method seeks to complement prior work in this area by addressing the need that crowd designers may have to craft the expressive motion of crowd members. We choose to focus exclusively on generating the variety of expressive motion portrayed by the crowd. We encourage use of our system in conjunction with the existing methods for navigation and behavior.

3 INTERACTIVE EVOLUTIONARY DESIGN

There are often two types of designers involved in the evolutionary design paradigm: the meta-designer and the designer. The meta-designer defines the search space by creating the original prototype to be altered by the genetic algorithm. In our case this is a motion clip with parameterized deformers applied to it. The meta-designer also defines which attributes are adjustable and gives ranges to these attribute values.

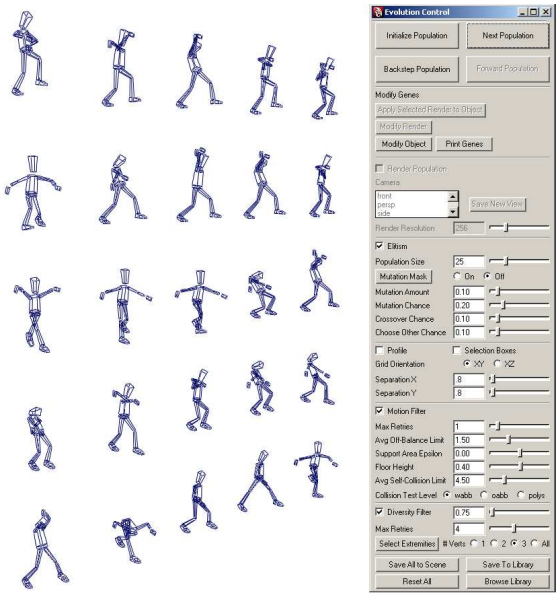

Figure 1: Interactive evolutionary design interface - the phenotypes are arranged in a grid for easy viewing and selection with the mouse.

There must be enough meaningful attributes to provide a sufficiently rich search space for the algorithm to traverse. The designer who directs the evolution process will be able to explore solutions that exist in the space defined by the meta-designer.

The technique described in this paper assumes that a parametric model with a rich search space has already been designed. Problems in the field of parametric modeling are both challenging and interesting, but they are beyond the scope of this paper. Furthermore, the solutions to parametric modelling problems are typically domain specific. Bezirtzis, et al. have given attention to some of the general concerns of parametric search space design in their industrial design case study (Bezirtzis et al., 2007) with emphasis on the fact that the design of a parametric space is an iterative process that can only be verified empirically. In general, the areas of high fitness in a parametric search space for interactive evolutionary design should be much larger than those in typical non-interactive evolutionary design applications. This is due to the small number of individuals that can be viewed in one population by the human visual system and the low number of generations required for convergence in a reasonable interactive environment. Also, discontinuities or sharp features in the search space should be avoided as much as possible. Otherwise, the designer will find that slight changes in gene values can result in dramatically different phenotypic expression.

In our system the list of attributes belonging to the parametric model corresponds to a genotype which is represented by a fixed-length array of floating point numbers (gene values). The phenotype in our system is the motion produced when these gene values are mapped to the model's parameters. The designer chooses a set of parents that will participate in the reproduction process for the next generation. Our genetic algorithm chooses two distinct, random parents from this set each time it produces a new offspring. It copies the genes from one parent and then switches to copying the other parent's genes with a user-defined crossover probability. The genes are then mutated given a user-defined probability by adding a random value between -1 and 1 scaled by a user-defined mutation amount.

There are many variants on reproduction algorithms for genetic search, and we are not married to this particular algorithm. However, the ability to let the user choose more than just two parents for the next generation provides a very nice property for the evolutionary design of crowd motions: The designer is able to simultaneously engineer distinct subspaces of motion for an entire crowd. Alternately, the designer can save interesting motions from exploration of one area of the search space into a motion library and can reintroduce these motions later, when exploring other areas of the search space. In addition, we encourage designers to explore different areas of the space by using a functionality in our interface that allows designers to backstep to previous generations and explore characteristics found in previously ignored parts of the population. It is important to note that our system cannot provide enough variation from one input motion to create all the types of action that might be required in a crowd scene. For example, it cannot turn a walking motion into a sitting motion or vice versa. Instead it provides varieties of walking motions that can be placed in a motion clip library along with varieties of sitting motions that were generated from a separate input motion.

A designer cannot design just any particular preconceived motion using our system. It will only be possible to generate motions that exist in the available search space. Furthermore, if a designer has a specific visual expressive quality in mind, it can often be modelled more quickly using conventional methods. This system is more useful for discovering novel motions and expressive qualities during exploration of the search space. Interactive evolutionary design has typically been aimed at aiding designers in the ideation phase of development of a character or idea, and the same applies here. Our system can be used to craft a novel motion for an individual character or a set of characters, but it may also serve as inspira-

tion in the planning stages of new character motion development.

4 MOTION GENERATION

Our method requires interaction from the designer at each iteration of the algorithm (see figure 2). We begin by preparing the input motion for modification by the genetic algorithm which, at the discretion of the designer, sends the resulting motion through a number of constraint-based filters and a dispersion algorithm.

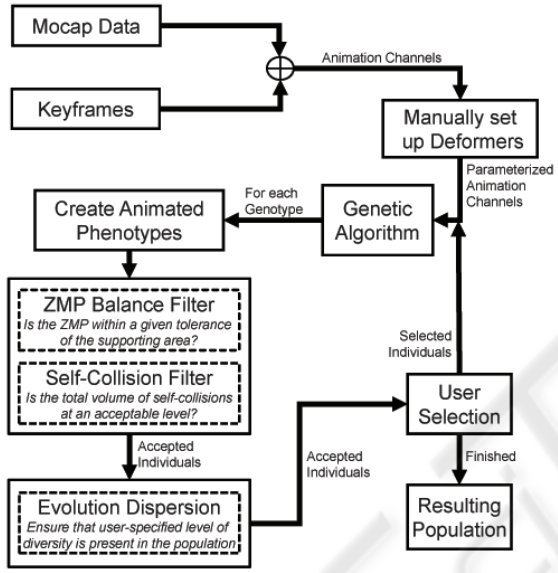

Figure 2: Pipeline Overview.

4.1 Input

The underlying representation of our motion evolution system consists of the set of animation channels taken from the input motion clip. We do not use a simplified motion model based on sinusoidal signals or any other similar technique. Therefore our system is not limited to cyclical motions, but can be used to generate variants on any type of motion that can be keyframed. Each animation channel is converted to piecewise Bezier spline geometry based on the data frames and their tangents. The meta-designer sets up deformers (Sederberg and Parry, 1986) to alter these splines and also defines which attributes of the deformers can be changed by gene values from the genetic algorithm. The splines are then altered by these deformers during the evolutionary design process. Once the spline geometry has been changed, the information is then transferred back to the animation channels of the individual being evolved. The resulting motion is displayed to the designer. We set the system up this way so that any 2D deformation that does not violate the one-to-one correspondence of the function represented by the animation channels may be applied, giving the meta-designer complete flexibility in creating and controlling the space of possible motions. See figure 3 for an example of how lattice deformers can be applied to an animation channel.

Ideally, the parametric models produced via the application of lattice deformers will provide a wide range of variation with few problems. However, some parameters may fight each other and occasionally lead to unrealistic or undesirable results. We address this by adding a layer of filters to the pipeline that automatically detect problematic results above a tolerance threshold and generate individuals to replace these unacceptable phenotypes. Though it would be ideal to correct these phenotypes, we choose to replace them in the interest of faster interactivity. This will be discussed further below. These filters, though general in some regards, should specifically apply to the type of motion being generated. In our case, any number of high-level qualities of locomotive animation can be addressed here, but we choose to focus on two particular properties of motion where we have observed the abuses of large variation: balance and self-collision.

4.2 Balance

In order to measure the level of balance in individual phenotypes and decide if they are acceptable we have adopted the zero moment point (ZMP) algorithm of Tak et al. ZMP in dynamic motion analysis is similar to the center of gravity in the static case. It is defined as the point on the ground plane under a character where there is zero moment. In other words, if this point were modeled as a joint between the character and the ground, there would be no actuation at this joint. As a character moves, the ZMP will create a trajectory over the ground plane.

Our balance filter ensures that the ZMP is always within the character's support area. The support area is the convex hull of the contact area between the feet and the floor. This definition encapsulates both the single and double stance phases of bipedal motion and allows for seamless calculation between the two phases. In some cases, the designer may want the motion of individuals in the crowd to appear exaggerated or cartoon-like. In a case such as this, a precise balance constraint might cause more harm than good. Therefore we have implemented the filter in such a way that the designer has control over the allowable balance error. We define balance error as the

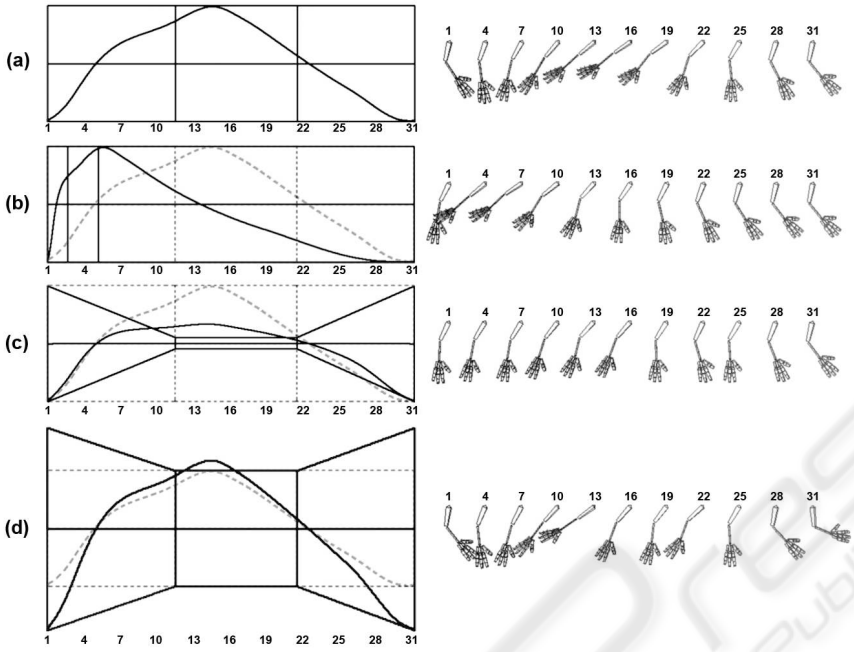

Figure 3: A variety of motions achieved using free form deformations on the rotation channels for the arm. Only the elbow joint values are represented by the graphs, but the same deformations are being applied to the shoulder in the animation sequence to the right. The X-axis of each lattice controls the passage of time (labeled in frames), and the Y-axis values control the rotation angle of the joint at any point in time. The first curve and accompanying animation sketch represent the original motion. Notice the lattice is in its initial, undeformed state.

distance from the ZMP to the closest point on the support area, and we sum this error over the entire motion sequence.

$$\sum_{i=1}^{F} \|ZMP_i - closestPoint(ZMP_i, K \cdot SA_i)\| \quad (1)$$

F is the number of frames in the animation, ZMP contains the zero moment point trajectory over all the frames, and SA contains the support area hull over all the frames. We also allow the user to enforce a lesser or greater degree of balance on the entire sequence by adding a custom support area scale factor K. The value of K can be adjusted through the system's interface thus allowing the user to shrink the support area to constrain the ZMP error more towards the center of support if more stable motion is required.

4.3 Self-Collision

Since the variation of arbitrary animation channel attributes may introduce self-collisions, we have developed a set of tests to filter out these self-colliding motions. We employ a simple bounding box method, testing for overlap of world-aligned bounding boxes and then calculate the volume of object aligned bounding boxes if two links are found to be intersecting. We then sum the volumes of all the intersections.

$$\sum_{j=1}^{L} \sum_{k=j+1}^{L} \{volume(O_j \cap O_k) \mid A_{j,k} = 0,\ W_j \cap W_k \neq \emptyset\} \quad (2)$$

L is the number of links or bones in the character, O contains the object-aligned bounding boxes of the links, W contains the world-aligned bounding boxes, and A is an adjacency matrix which describes the spatial adjacency of the links. The above summation is accumulated over every frame of the motion sequence. By summing the total volume we avoid over-penalizing quick, grazing collisions while still penalizing quick, high-volume collisions as well as slow, grazing collisions. We ignore collisions that are design artifacts of the given model's geometry as well as collisions between adjacent links. Since we do not test at the polygon-polygon level for intersection in order to reduce running time, this filter only serves as a heuristic and not an exact measure of self-collision. Nevertheless, in practice, it gives a good indication of the level of self-collision for a given phenotype, especially since we want to keep computation to a minimum in the interest of higher levels of interaction.

4.4 Replacement Method

It would be preferable to fix unacceptable phenotypes and bring them back into the allowable space of motion defined by our balance and self-collision constraints. In fact, the method of Tak et al. optimally transforms unbalanced animation sequences into balanced motions (Tak et al., 2002). There are also methods for correcting self-colliding animation via inverse kinematics optimization (Müller, 2004). However, we do not attempt to fix unacceptable motions because this would require an optimization process. Optimization would introduce longer wait times between generations, reducing the interactivity of our evolutionary design environment. Instead, we generate a new individual motion to replace the old one and resubmit it to the filters for evaluation. We are able to proceed in this way because our space of possible motions was created from a balanced original motion that was free from self-collisions and so the rich search space of variants will provide a viable, balanced replacement motion within a few iterations. Furthermore, the reproduction process uses the user-selected motions as input for any newly generated replacement motion so the offspring will most likely have similar balance and self-collision error to the parents' errors. Even so, generating replacement individuals with our reproduction algorithm may not always result in an acceptable individual, especially if the user's constraints are too restrictive or if the search space is not rich enough. If this occurs, we let the replacement process run for a user-defined maximum number of iterations and then force it to move on, replacing the unbalanced individual with the most balanced option found so far.

4.5 Diversity

Although duplicate motions in a crowd are harder to spot than duplicate appearances, they can be spotted just as easily whether or not each character has a different appearance. Moreover, varied motion between two visually equivalent individuals can help to obscure the fact that they have a similar appearance (Mcdonnell et al., 2008). It follows that diversity of motion provides desirable qualities for crowd animation. Genetic algorithms by their very nature seek convergence to a specific area of the search space. Because we want to design coherent sets of motion and because proximity in the search space corresponds to phenotypic similarity in our system, this convergence is beneficial. However, it can also be problematic since we want to find a diverse group of motions. We do not want the algorithm to converge so far that the resulting population becomes homogenous. The designer in the interactive evolutionary design paradigm exercises ultimate control over how far the population converges. As mentioned earlier, if the designer chooses a variety of individuals as parents in one generation, the chances of diversity in the next generation are greater.

In order to encourage diversity in the population and ensure sufficient sampling of the local search space surrounding the designer's regions of interest, we run the evolution dispersion algorithm introduced by Marks et al. (Marks et al., 1997) at each generation. This algorithm alters the genes of individuals in a way that will discourage similarity between phenotypes. We measure the phenotypic difference between two individuals by comparing their point clouds. Our method is somewhat similar to the technique used by Kovar, et al. in their paper on motion graphs (Kovar et al., 2002). The point cloud for a character consists of the positions of a subset of the character's vertices over time. We consider the distance between point clouds to be the sum of the root mean squared error between all the corresponding vertices of two point clouds. We allow the designer to specify which parts of the model to choose vertices from when making this comparison. The designer can also specify the level of dispersion required at each generation.

4.6 Using the Filters

Because the motion filters require extra computation, using them will inevitably result in longer wait times between generations. The balance and self-collision filters are most useful during the first couple generations of the design process when the algorithm is sampling a wide area of the search space. These filters make the designer's job easier by avoiding the areas of the search space where two or more parameters fight each other and produce unwanted motion artifacts. As the design process progresses and the algorithm begins to converge, these filters should be turned off to speed up the interactivity of the system. In contrast, the diversity filter should only be used near the end of the design process when the algorithm is sampling a smaller area of the search space. Using the above filters in this way will maximize the fitness of the options presented to the designer throughout the process in a way that is customizable to fit the designer's needs.

5 RESULTS

We implemented our interactive evolutionary design motion generation method in a layer of MEL and

Python code over Autodesk's Maya environment. We chose to use Maya because it is flexible enough to apply evolutionary design not only to animation, but also to modeling, texturing, and special effects domains. We also chose this software environment because most of the designers at our research facility are familiar with its interface, and we would like them to be able to create and evolve designs from their own parametric models.

Table 1: Average time (sec) and average number of rejected phenotypes while producing a generation of size 25.

Filter	None	Balance	Collide	Disperse
Time	10.0	76.0	47.8	28.0
Reject	-	1.5	3.5	2

Our hardware consists of an Intel Xeon 2.66 GHz cpu with 4 GB of RAM and an Nvidia Quadro FX 5600 graphics card. Performance metrics of our current implementation under varying conditions can be seen in Table 1. Each filter was set up with a maximum of 2 replacements per individual except for the dispersion filter which had a maximum of two replacements per population.The number of replacements required at each generation depends on the richness of the search space as well as on the designer's selections and the constraint thresholds set by the designer. The time required to replace an individual that does not meet the constraints is approximately equal to the time required to generate the original individual. The running time is dependent on the size of the population so smaller populations will take less time per individual.

For these tests, we created a parametric walk cycle model with lattice deformers on 14 different animation channels. Note, however, that any type of deformer available in Maya may be utilized. The deformers' transformations were controlled by 14 floating point numbers from the fixed-length array that formed the genotype of each individual. Although the size of the population is adjustable by the user, we generally use populations of size 25 because larger populations are harder for the human visual system to fully and easily comprehend. We are currently working to improve the running times for the various filters in the interest of better interactivity.

Figure 4 shows a sampling of the variety of motions that can be produced from a single parameterized walk cycle. These motions are shown in an animation sketchbook style where every fourth frame is drawn to show the change in form over time. We are displaying the motions in animation sketchbook style only for the purposes of the paper. In our evolutionary design interface, the motions are actually displayed as time-varying animation.

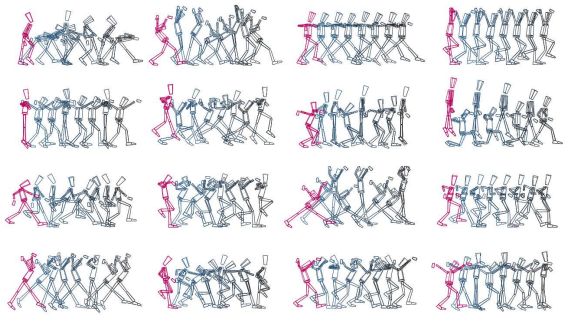

Figure 4: A variety of sixteen motions evolved from a single walk cycle motion clip.

We have conducted a user study to determine if our software truly provides an intuitive way for designers to create varieties of motion. In this study we asked three graduate students from the college of design who were already familiar with the Maya software environment to participate. After a brief 10-15 minute tutorial on how to use the system, each designer explored the space of options provided by the deformers in our parametric model of a walk cycle motion and decided on a style of motion to interactively develop. The entire process, including the tutorial, search space exploration, and motion design took anywhere from an hour and a half to two hours for each designer. The designers all felt that the system was easy to learn and enjoyed experiencing the interactive evolutionary design process. They agreed that the motions produced would make for interesting, varied background crowd character motions or as a fertile starting point for the development of a character motion with an individualized style. However, they said that the motion produced would need to be refined with a high-level of control over specific keyframes if it were to be used for foreground or hero characters. The populations they created (See the animations on the project web site at http://accad.osu.edu/Projects/Evo/) including a set of energetic dancers, a group of feminine characters, and a mob of zombies demonstrate the wide variety of motions that are acheivable from just one input motion.

6 FUTURE WORK

There are many more ways to filter the individual motions that could prove to be helpful to the crowd designer. A few such filters that could be useful include: joint torque analysis based on limits from a comfort model (Ko and Badler, 1996), various meth-

ods of psychological analysis (aggressiveness, energy level, coordination, etc.), Laban movement analysis (effort and shape), as well as gender or age analysis. We are investigating methods for implementing the balance and self-collision filtering process in parallel on subsets of the population using our quad-core processors which should speed up computation times and enhance the interactivity of the system.

7 CONCLUSIONS

Our method presents a novel approach to evolving families of expressive motion, making it easier for a crowd designer to quickly and intuitively find a satisfying combination of motion variations for a specific application. This method could prove especially useful to those who do not have access to motion capture facilities or cannot afford to spend time capturing a wide range of motion clips. Our interaction model allows the user to view and make decisions about entire generations at once, and our reproduction algorithm allows for evolution of multiple (even mutually exclusive) styles of motion simultaneously. Our use of user-defined constraints plus the designer's selections as the determination of fitness exemplifies a hybrid system that seeks to maximize the designer's time and attention in the evaluation of populations by filtering out the individuals who do not meet the given criteria.

REFERENCES

Amaya, K., Bruderlin, A., and Calvert, T. (1996). Emotion from motion. In *Graphics Interface '96*, pages 222–229.

Bezirtzis, B. G., Lewis, M., and Christeson, C. (2007). Interactive evolution for industrial design. In *C&C '07: Proceedings of the 6th ACM SIGCHI Conference on Creativity & Cognition*, pages 183–192, New York, NY, USA. ACM.

Chi, D., Costa, M., Zhao, L., and Badler, N. (2000). The emote model for effort and shape. In *SIGGRAPH '00 Proceedings*, pages 173–182, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.

Gleicher, M. (2001). Comparing constraint-based motion editing methods. *Graphical Models*, 63(2):107–134.

Ko, H. and Badler, N. I. (1996). Animating human locomotion with inverse dynamics. *Computer Graphics and Applications, IEEE*, 16(2):50–59.

Kovar, L., Gleicher, M., and Pighin, F. (2002). Motion graphs. In *SIGGRAPH '02 Proceedings*, volume 21, pages 473–482, New York, NY, USA. ACM Press.

Kwon, T., Lee, K. H., Lee, J., and Takahashi, S. (2008). Group motion editing. In *SIGGRAPH '08 Proceedings*, pages 1–8, New York, NY, USA. ACM.

Li, T.-Y. and Wang, C.-C. (2007). An evolutionary approach to crowd simulation. In *Autonomous Robots and Agents*, pages 119–126.

Lim, I. S. and Thalmann, D. (2000). Tournament selection for browsing temporal signals. In *Symposium on Applied Computing '00 Proceedings*, pages 570–573, New York, NY, USA. ACM.

Marks, J. et al. (1997). Design galleries: a general approach to setting parameters for computer graphics and animation. In *SIGGRAPH '97 Proceedings*, pages 389–400, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.

Massive Software (2009). Massive prime. Accessed online www.massivesoftware.com/prime/.

Mcdonnell, R., Larkin, M., Dobbyn, S., Collins, S., and O'Sullivan, C. (2008). Clone attack! perception of crowd variety. In *SIGGRAPH '08 Proceedings*, volume 27, pages 1–8, New York, NY, USA. ACM.

Müller, A. (2004). Collision avoiding continuation method for the inverse kinematics of redundant manipulators. In *Robotics and Automation '04 Proceedings*, volume 2, pages 1593–1598 Vol.2.

Musse, S. R. and Thalmann, D. (2001). Hierarchical model for real time simulation of virtual human crowds. *Visualization and Computer Graphics, IEEE Transactions*, 7(2):152–164.

Neff, M. and Fiume, E. (2005). Aer: aesthetic exploration and refinement for expressive character animation. In *SCA '05 Proceedings*, pages 161–170, New York, NY, USA. ACM Press.

Sederberg, T. W. and Parry, S. R. (1986). Free-form deformation of solid geometric models. *SIGGRAPH '86 Proceedings*, 20(4):151–160.

Sims, K. (1993). Interactive evolution of equations for procedural models. *The Visual Computer*, 9(8):466–476.

Sung, M. (2007). Continuous motion graph for crowd simulation. In *Technologies for E-Learning and Digital Entertainment*, volume 4469, pages 202–213. Springer Berlin / Heidelberg.

Sung, M., Kovar, L., and Gleicher, M. (2005). Fast and accurate goal-directed motion synthesis for crowds. In *Symposium on Computer Animation '05 Proceedings*, pages 291–300, New York, NY, USA. ACM Press.

Tak, S., Song, O.-Y., and Ko, H.-S. (2002). Spacetime sweeping: An interactive dynamic constraints solver. In *Computer Animation '02 Proceedings*, pages 261–271, Washington, DC, USA. IEEE Computer Society.

Treuille, A., Cooper, S., and Popovic, Z. (2006). Continuum crowds. *ACM Transactions on Graphics*, 25(3):1160–1168.

Ventrella, J. (1995). Disney meets darwin-the evolution of funny animated figures. *Computer Animation*, 00.

Wang, J., Drucker, S. M., Agrawala, M., and Cohen, M. F. (2006). The cartoon animation filter. *ACM Transactions on Graphics*, 25(3):1169–1173.