Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2012)
R. Boulic and T. Komura (Editors)

Cloning Crowd Motions

Yi Li[†1] Marc Christie[1] Orianne Siret[1] Richard Kulpa[2] and Julien Pettré[1]

[1]Team MimeTIC, IRISA / INRIA-Rennes, France
[2]M2S, UEB, Université de Rennes 2, France

Abstract

This paper introduces a method to clone crowd motion data. Our goal is to efficiently animate large crowds from existing examples of groups of characters motions by applying an enhanced copy and paste technique on them. Specifically, we address spatial and temporal continuity problems to enable animation of significantly larger crowds than our initial data. We animate many characters from the few examples with no limitation on duration. Moreover, our animation technique answers the needs of real-time applications through a technique of linear complexity. Therefore, it is significantly more efficient than any existing crowd simulation-based technique, and in addition, we ensure a predictable level of realism for animations. We provide virtual population designers and animators with a powerful framework which (i) enables them to clone crowd motion examples while preserving the complexity and the aspect of group motion and (ii) is able to animate large-scale crowds in real-time. Our contribution is the formulation of the cloning problem as a double search problem. Firstly, we search for almost periodic portions of crowd motion data through the available examples. Secondly, we search for almost symmetries between the conditions at the limits of these portions in order to interconnect them. The result of our searches is a set of crowd patches that contain portions of example data that can be used to compose large and endless animations. Through several examples prepared from real crowd motion data, we demonstrate the advantageous properties of our approach as well as identify its potential for future developments.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

1. Introduction

A challenging problem within computer graphics is how to combine the realism of animation with the complexity of crowds. Moreover, the difficulty increases with the constant growth of size of virtual environments. The most natural answer to this problem is crowd simulation, but it can not satisfy all needs of virtual population designers, because the design work requires not only substantial computational resources, but also tedious manual work of setting up simulations. An interesting alternative to crowd simulation is example-driven approaches, but its resulting animations are limited by the size and duration of initial data.

In this paper, we propose to create and animate large populations by *cloning* example motion data. The aim of our cloning method is to address the motion continuity problems in order to enable large crowd animations without limit of duration. The key idea is to cut sets of trajectories from real data which displays some expected effects, and then paste them into virtual environments as repeatable textures. Consequently, it is possible to create infinite animations in both time and space. Our technique combines the concept of crowd patches [YMPT09] with trajectory editing tools [KLLT08, KHKL09]. However, unlike Yersin *et al.*, who create trajectories inside periodic patches to satisfy connectivity, we propose to extract periodic patches by searching through real data sets and establish connectivity between patches. This process raises two issues: first, how to extract periodic motions inside real data so that both spatial and temporal continuities can be guaranteed; second, how to adapt real data in order to extract pieces of motions from different data sets and patch them together.

To address these issues, we formulate the problem as a two-stage search: first, a search for *almost periodic* portions

† The authors are very grateful to Steve Tonneau who rendered valuable help in proofreading.

of motion data; and second, a search for *almost symmetry* between the conditions at the limits of portions. The portions exhibiting such properties can then be transformed into a set of interconnectable crowd patches. We further extend the crowd patches with global deformation techniques as a means to adapt initial data to diverse environment geometries.

The key contributions of this paper are:

- a technique to automatically extract almost periodic portions of crowd motions from real data sets and apply local transformations to make these portions periodic;
- a technique to automatically find almost symmetries and apply local transformations on periodic portions to enable smooth interconnections;
- and finally a technique to map patches of crowds extracted from real data into new virtual environments.

These techniques enable infinitely large environments to be populated with only the cost of replaying existing data. The paper is organized as follows: after comparing and contrasting our approach with state-of-the-art contributions in the field and recalling the concept of crowd patches on which our work is built in Section 2, we detail our core contribution by introducing the notions of periodicity and connectivity in Section 3. We then illustrate our contribution on an example with 1D data for the sake of clarity in Section 4 and present main results in Section 5. Finally, we discuss the limitations and conclude our paper in Sections 6 and 7, respectively.

2. Related Work

Previous contributions to the research area of populating large-scale virtual environments can be split up in two categories: approaches with realistic simulation models, and approaches which extract and adapt real data.

2.1. Crowd Simulation

Crowd Simulation is a natural solution to the problem of populating interactive environments. Various types of solutions and simulation models have emerged recently: the seminal rule-based Reynolds' model [Rey87] (later extended in [Rey99]), the force-based models [HM95,PAB07,JHS07], the continuum dynamics based model [TCP06], the local field-based model [KSHF09, KSH*12], the velocity-based models [PPD07, POO*09, GCC*10, KO10], and the vision based model [OPOD10].

These models enable the creation of impressive crowd motions from simple individual behaviours, actions, and interactions. However, crowd simulation techniques do not answer some specific needs in the task of populating large interactive environments. Firstly, it is difficult to match a simulation setup with a desired content because of the chaotic nature of crowd simulation, despite some user-friendly tools such as the CrowdBrush [UCT04]. Secondly, although many solutions have been proposed to improve performance such as the level-of-details methods for rendering and simulation [DHOO05, PCM*06, KOOP11], simulations of virtual crowds are still computationally expensive in terms of time and memory usage and hence the sizes of the virtual environments have to be limited. Thirdly, recent efforts in detecting and classifying simulation artifacts [KSA*09] demonstrate that it is almost impossible to perform a simulation without these artifacts. We have to point out that any kind of strange behaviour, when it is detected by spectators, will break the believability of a populated scene.

Unlike crowd simulation, our solution is based on example of crowd motions: We reproduce the motion itself instead of modelling how humans move and then reproduce their behaviours in simulation. It is more intuitive in that the designer selects the aspects of the motion he/she desires rather than selecting and specifying the behaviours that the virtual characters should simulate. In addition, it enables richer content creation: One only needs to parse the illustrated examples and hence there is no need to model all expected behaviours.

2.2. Example-based and Data-driven Crowds

Example-based or data-driven methods [MH03, LCL07, KLLT08, SKSY08, LFCCO09, HLLO10, JCP*10, TLG11] have been applied in recent years to crowd simulation with the aim of reproducing in virtual worlds a number of characteristics extracted from recorded data sets.

In [LCL07], after decomposing trajectories of pedestrians from real data in order to isolate interactions between 2 or 3 pedestrians, Lerner *et al.* employ at run-time an agent-based simulation technique: Each time an agent faces an interaction with one or multiple characters, the database is queried for similar character states. Our approach differs in that we do not rely on the concept of agent: Every animation is directly taken from the example data sets and hence there is no search process. Lerner *et al.* target the creation of an example database which captures solutions for any interaction between individual characters, while we aim at capturing specific situations that we want to faithfully reproduce. Furthermore, since our technique is not restricted to simple interactions between a few pedestrians, it can capture and reproduce complex and visually rich interactions.

Other approaches such as the ones in [LCHL07, JCP*10] identify the nature of relations between characters in a number of different situations. The underlying geometric relations are analyzed using a statistical approach, modelled, and further simulated in virtual environments. In an implicit way, these techniques capture a large number of social and cultural rules which guide human relations in groups. In comparison to these techniques, our approach does not attempt to identify such relations, but only reproduces pieces of trajectories in situations that are potentially very complex.

Consequently, our approach can capture complex situations that can not be explained through the previous statistical approaches that relies on models with only a limited set of parameters.

As shown in [KLLT08], one can edit trajectories of animations — limited in both space and time — for complex groups and then adapt them to a given virtual environment while preserving the original structure of the motions. Even though the editing process can successfully integrate new constraints from the environment (e.g., obstacles), it is not scalable to very large environments unless a very large data set is available as the input. Our approach differs in that we can generate an unlimited animation in both space and time from a few examples of limited sizes, because the source trajectories are simply the motifs that can be copied and then pasted into the environment.

Among the previous work, our approach is most similar to the method proposed by Shum *et al.* in [SKSY08] to create large-scale animations where characters have close interactions (e.g., a duel). Just like our approach, Shum *et al.* store the interactions between the characters in patches before concatenating spatio-temporally the patches to compose large-scale scenes. However, since the method in [SKSY08] is designed for close interactions, it can not be applied to generate scenes where multiple characters continuously interact. On the contrary, our approach is designed with characters who continuously interact in mind. Furthermore, all actions inside the action-level motion graph created by Shum *et al.* in [SKSY08] have to undergo a manual annotation process, which is potentially a tedious thing to do, epseically when the motion capture data contain many characters.

2.3. Crowd Patches

Our approach, which is strongly pertained to the notion of motif, is built upon *crowd patches* [YMPT09] in where the authors draw a parallel between challenges in object texturing and challenges in populating virtual environments.

The objective of [YMPT09] is to populate virtual environments with convex polygon-shaped patches, where each patch Φ contains cyclic animations over a constant period τ. If a dynamic character never leaves the patch, it is a so-called *endogenous* object and its trajectory Γ must meet the patch periodicity condition: $\Gamma(0) = \Gamma(\tau)$. When a dynamic character crosses from one patch to a neighbouring patch (i.e., it is *exogenous*), the presence of another exogenous character is required to fulfil the periodicity condition. Furthermore, trajectory of an exogenous character must meet the common limit conditions which are captured by the concept of *patterns* Ψ as defined in [YMPT09]: In order to connect two patches at two edges, one from each patch, the edges' patterns must be mirror images of each other (i.e., the edges are of equal length; the patterns have the same period; and for each entry/exit point of one pattern, there must be an exit/entry point with the same time and the same position in the other pattern). For more details about crowd patches, we refer the reader to [YMPT09].

2.4. Comparison to Previous Work

In this paper, we extend the concept of crowd patch in order to copy and paste elements of existing motion data, because crowd patches enable the on-the-fly generation of crowd animations without any limitations on neither size nor duration. However, the challenges we tackle here differ strongly from those identified by Yersin *et al.* in [YMPT09], where the problem of generating trajectories is solved inside a patch, assuming that the conditions defined by the patterns (i.e., positions and time of entries/exits of the patch) are given. The problem we target is the inverse one: How can one extract *periodic* patches that can *connect* to other patches through patterns, given a set of motion tracked trajectories. Therefore, we search for spatio-temporal portions of trajectories which satisfy the periodicity conditions of patches, and then exploring the possible connections between the different portions we have extracted in order to build a global animation through simple composition. To find periodic portions of trajectories in real data, we propose an *almost periodic* criterion which has to be satisfied by portions of trajectories and then locally deform these partial trajectories as done in [KLLT08, JCP*10].

3. Cloning Crowd Motion Data

In this section, we describe in detail how we clone crowd motions by copying trajectories inside crowd patches and then pasting these trajectories to empty virtual environments in order to populate them.

3.1. Crowd Motion Data

In our problem setting, we are given motion capture data of a set of N pedestrians, $\mathcal{R} = \{r_1, ..., r_N\}$, and the total duration of the motion capture is T sec. For pedestrian r_i, its 2D trajectory Γ_i is composed of the pedestrian's physical state $\mathbf{q}_i(t)$ at time t, where $\mathbf{q}_i(t) = [x_i(t), y_i(t)]^\mathsf{T}$ and T is the symbol for vector transpose. However, our method is not limited to this case and it can be generalized to higher dimensional physical states such as $\mathbf{q}_i(t) = [\mathbf{x}_i(t), \dot{\mathbf{x}}_i(t)]^T$, where $\mathbf{x}_i(t)$ is the degrees of freedom (DOFs) of pedestrian r_i and $\dot{\mathbf{x}}_i(t)$ is the first-order derivatives of the DOFs with respect to time.

3.2. Copying Motions: from Data to Patches

In the initial step of our method, we search for portions of captured trajectories that can be transformed into patches. More specifically, we check whether the content inside every portion is almost periodic, where the notion of almost periodic is defined through a similarity metric between initial and final states of the patch.

A portion of data is delimited in space and time by a polygon P and an interval $[t_0, t_0+\tau]$, where t_0 is the start time and $t_0+\tau \leq T$, respectively. In order to obtain the delimited portion of data, the user has first to choose an appropriate polygon P, where the size of P depends on the environment he/she wants to populate. Furthermore, the user should take note of that, even though the shape of P is not limited by the crowd patches, square-shaped polygons are often preferred because they greatly simplify the implementation. Secondly, the user needs to choose appropriate values for both t_0 and τ. Finally, the delimited portion of data consists of the segments of the trajectories that intersect the surface of P during the time period $[t_0, t_0+\tau]$.

In order to verify whether a portion of data that is delimited in space and time by a polygon P and an interval $[t_0, t_0+\tau]$, respectively, is part of a patch Φ, we extract the initial state of all pedestrians inside the portion $\mathbf{Q}(t_0) = [\mathbf{q}_0(t_0), ..., \mathbf{q}_m(t_0)]$, and their final state $\mathbf{Q}(t_0+\tau) = [\mathbf{q}_0(t_0+\tau), ..., \mathbf{q}_n(t_0+\tau)]$, where m and n are the numbers of pedestrians inside P at time t_0 and $t_0+\tau$, respectively. If $m = n$, we sort the items in the vector $\mathbf{Q}(t_0+\tau)$ so that $\sum_{k=0}^{n} \mathbf{d}_q[k]$ is minimized, where $\mathbf{d}_q[k] = \|\mathbf{q}_k(t_0) - \mathbf{q}_k(t_0+\tau)\|$. The trajectories inside the extracted portion is considered to be almost periodic if $\mathbf{d}_q[k] < \varepsilon \ \forall k \in \{0, 1, ..., n\}$, where ε is a small positive number, and the portion is part of the patch Φ. On the contrary, when $m \neq n$, the extracted portion can be rejected (i.e., it is not part of a patch) right away, because the almost periodicity condition implies that there are an equal number of pedestrians at time t_0 and $t_0+\tau$.

To quantify how periodic the patch Φ is (i.e., how similar $\mathbf{Q}(t_0)$ and $\mathbf{Q}(t_0+\tau)$ are), we define the periodicity cost function $f_p(\Phi) = \max \mathbf{d}_q$.

$$f_p(\Phi) = \max \mathbf{d}_q. \tag{1}$$

Finally, the trajectories inside the patch Φ are deformed using the Laplacian motion edition method presented in [KHKL09] such at $\mathbf{q}_k(t_0)$ and $\mathbf{q}_k(t_0+\tau)$ are connected at $\mathbf{q}_k(t_0) + (\mathbf{q}_k(t_0+\tau) - \mathbf{q}_k(t_0))/2$ after the deformations. A final verification step ensures the deformations do not generate any collision. If a collision does occur, the patch is rejected.

3.3. Pasting Motions: Patch Connection

From the set of patches we have obtained in the previous step, we now search for possible connections (more specifically the existence of *mirror patterns*) between different patches with the aim of composing large scale crowd animations. Once again, it is highly unlikely to find two perfectly matching mirror patterns, and hence we search for *almost symmetric* patterns instead. Between two almost symmetric patterns, we define a connectivity cost function in order to identify patterns that can be connected with minimum deformations.

Given a patch Φ, the pattern Ψ_i for side i of the polygon P that defines Φ geometrically contains two sets of data (i.e., I_i and O_i) that specify the inputs and ouputs at the limit of Ψ_i: I_i is the set of intersections between Ψ_i and exogenous trajectories entering Ψ_i, while O_i is the set of intersections between Ψ_i and exogenous trajectories exiting Ψ_i.

Before the almost symmetry condition between patterns Ψ_i and Ψ_j (taken from patches Φ_a and Φ_b, respectively) can be established, we have to verify that Ψ_i and Ψ_j (more specifically the sides of Φ_a and Φ_b where Ψ_i and Ψ_j are defined) have the same length and the patterns (i.e., Φ_a and Φ_b) share the same period of time τ. If these conditions are fulfilled, Ψ_i and Ψ_j are considered to be almost symmetric if $|I_i| = |O_j|$ and $|O_i| = |I_j|$, where $|I_i|$ is the cardinality of the set I_i.

To quantify how symmetric patterns Ψ_i and Ψ_j are, we define a connectivity cost function f_c as

$$f_c(\Psi_i, \Psi_j) = \max(f_c^{IO}, f_c^{OI}), \tag{2}$$

where f_c^{IO} quantify how symmetric I_i and O_j are, while f_c^{OI} quantify how symmetric O_i and I_j are. In the following paragraph, we detail how f_c^{IO} is computed and f_c^{OI} can be obtained in a similar way. The sets I_i and O_j can be represented by two vectors of state and time tuples: $\mathbf{QT}_i^I = [(\mathbf{q}_{i0}^I, t_{i0}^I), ..., (\mathbf{q}_{im}^I, t_{im}^I)]$ and $\mathbf{QT}_j^O = [(\mathbf{q}_{j0}^O, t_{j0}^O), ..., (\mathbf{q}_{jm}^O, t_{jm}^O)]$, where the superscripts I and O stand for input and output, respectively, and m is the cardinality of the sets. Next, we sort the items in $\mathbf{QT}_j^O$ so that $\sum_{k=0}^{m} \mathbf{d}_{qt}[k]$ is minimized, where

$$\mathbf{d}_{qt}[k] = \mathbf{d}_q[k]\mathbf{d}_t[k] = \|\mathbf{q}_{ik}^I - \mathbf{q}_{jk}^O\| |t_{ik}^I - t_{jk}^O|. \tag{3}$$

Finally, f_c^{IO} is defind as $f_c^{IO} = \max \mathbf{d}_{qt}$.

When patches Φ_a and Φ_b present a good degree of connectivity through patterns Ψ_i and Ψ_j, we deform the trajectories entering Φ_a at I_i and the ones exiting Φ_b at O_j in the way as described in Section 3.2. Note that here one-dimensional time warp has to be performed in addition to spatial path editing using the same technique (i.e., the Laplacian motion edition method presented in [KHKL09]). The trajectories exiting Φ_a at O_i and the ones entering Φ_b at I_j are connected in a similar way.

3.4. Composing Animations

In the previous two steps, we have not only extracted portions of motion capture data that are transformable into patches, but also identified possible connections between

these patches through symmetric patterns. In addition, we have assigned costs to the patches (i.e., periodicity cost) and to their connections with other patches (i.e., connectivity cost), and these costs are proportional to the amount of deformations that have to be done to the trajectories in order to make the final animations both spatially and temporally continuous. If we create an undirected graph where each node of the graph represents a patch, the cost assigned to the edge between two nodes is simply the connectivity cost we computed in Section 3.3 for the two neighbouring patterns (one from each patch). Using any standard graph traversal algorithm, the minimum-cost path between two patches can be easily obtained. Note that the periodicity costs of the patches are not stored in the graph. If a patch should be ignored because its periodicity cost is too high, we simply remove the node that represents the patch from the graph.

Finally, before a patch is inserted into a given virtual environment, it can be deformed using the boundary element method based technique proposed in [GSN04, GN08] in order to adapt the patch to the exact geometries of the environment. The benefits of such deformations are illustrated in Sections 4.4 and 5.2.

4. An Illustrative Example

4.1. Crowd Motion Data

In this section, we illustrate the method presented in Section 3 by animating a large number of pedestrians queueing using the motion capture data of 28 pedestrians following each other in a circle (see Figure 1). This example is chosen because it allows us to display the data in a two-dimensional plot (more specifically in polar coordinate system). From the angle-time plot shown in Figure 1, it is clear that the trajectories last 90*sec*. The oscillatory phenomenon in the figure corresponds to the propagation of a counter-flowing stop-and-go wave.

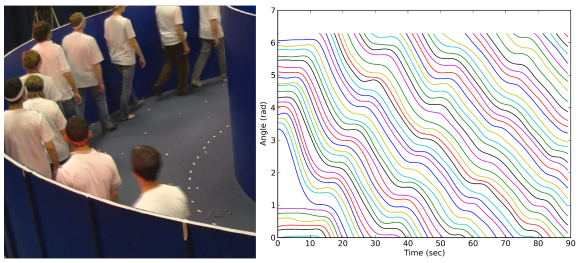

Figure 1: *An illustrative example of pedestrians following each other.* ***Left****: A snapshot of the motion capture session.* ***Right****: Plot of the pedestrians' polar angles with respect to time.*

4.2. The Patch Search

Following the method described in Section 3.2, we search for portions of the motion capture data that can be transformed into patches. Since the given data is one-dimensional, the boundary of a potential patch forms a rectangle and the area of the rectangle is bounded by the horizontal lines for time t_0 and time $t_0+\tau$ and by the vertical lines for angle a_0 and angle a_1 (see Figure 2). Essentially, the vertical dimension of the rectangle represents the spatial coverage, while its horizontal dimension represents the temporal coverage.

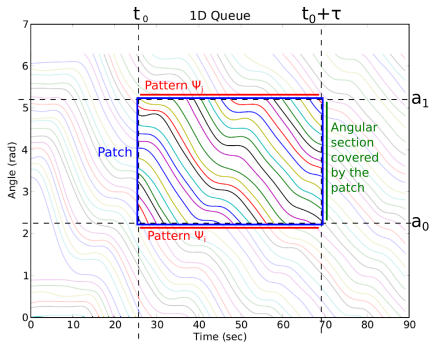

Figure 2: *An illustration of a rectangular portion extracted from the one-dimensional data of 28 pedestrians following each other in a circle.*

In order to accelerate the patch searches, we first fix the period τ and discretize the search space. Since the period of the propagation of stop-and-go waves we want to capture is about 20*sec*, τ is set to the same value (i.e., $\tau = 20sec$). Secondly, we set the polar angle step size to $\pi/10rad$, while the time step size is set to $1sec$ to obtain the discretized search space. Thirdly, we construct a pattern for each point (a_0,t_0) in the discretized search space under condition of $t_0+\tau \leq 90sec$, because the motion capture data lasts 90*sec*. Fourthly, by associating a pattern pair (Ψ_i, Ψ_j), a patch Φ can be constructed if the portion bounded by Ψ_i and Ψ_j is a rectangle and the trajectories inside the rectangle are almost periodic. Finally, if the almost periodicity condition is met, the periodicity function $f_p(\Phi)$ is evaluated. For example, as shown in Figure 2, the rectangle bounded by patterns Ψ_i and Ψ_j forms a patch, because the trajectories inside the rectangle are almost periodic: There are an equal number of pedestrians inside the patch at time t_0 and time $t_0+\tau$. Consequently, the periodicity function $f_p(\Phi)$ can be evaluated as illustrated in Figure 3.

The results of the patch search can be stored inside a periodicity matrix as shown in Figure 4, where column i and row j correspond to patterns Ψ_i and Ψ_j, respectively. Each element of the matrix corresponds to a portion bounded by Ψ_i and Ψ_j. If the portion is rectangular and the trajectories inside are almost periodic, the portion can be transformed to a patch Φ and the value of the element is set to the periodicity function $f_p(\Phi)$. Otherwise, we don't give any value to the element. Note that the periodicity matrix is pseudo-diagonal because we sort the patterns by the starting time t_i, and only two patterns with the same starting time can be transformed

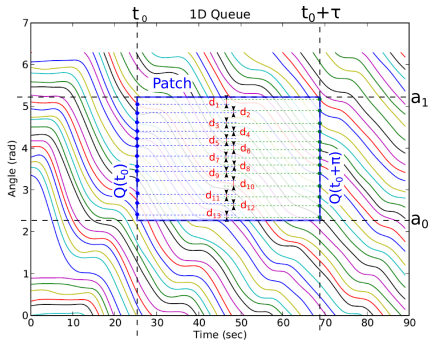

Figure 3: *An example of a patch: The trajectories are almost periodic and hence the periodicity function $f_p(\Phi)$ can be evaluated.*

into a patch. Finally, although the motion capture data lasts only 90*sec*, Figure 4 shows that we are able to extract a large number of patches, and among them the ones with the lowest periodicity values can be used to animate crowds with a high level of realism.

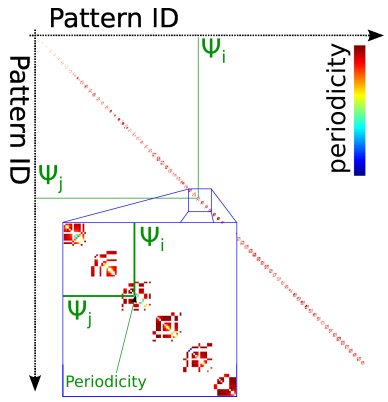

Figure 4: *The periodicity matrix.*

4.3. Patch Connectivity

We now search for possible interconnections between the set of patches constructed in the previous step. Following the steps described in Section 3.3, we consider each possible pair of patterns (Ψ_i, Ψ_j) and then evaluate $f_c(\Psi_i, \Psi_j)$ as illustrated in Figure 5 if the patterns are almost symmetric. Note that, in Figure 5, pattern Ψ_i has as many outputs as pattern Ψ_j has entries.

The results from the search can be stored in a connectivity matrix as shown in Figure 6. Note that the value of a matrix element is set to $f_c(\Psi_i, \Psi_j)$ if and only if patterns (Ψ_i, Ψ_j) are almost symmetric. If they are not, the matrix element is left empty. Since the patterns are sorted by the starting time t_i, the bottom-right part of the matrix is denser.

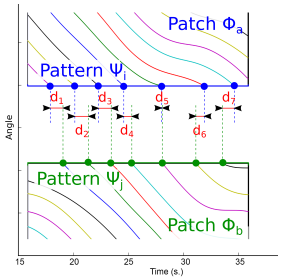

Figure 5: *Connecting two patches by matching almost symmetric patterns.*

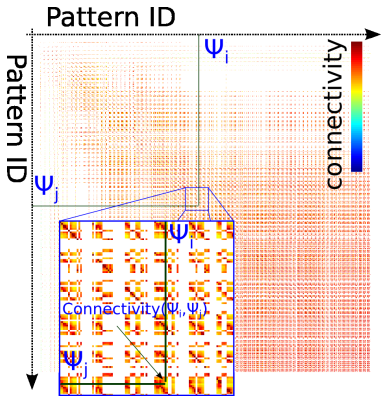

Figure 6: *The connectivity matrix.*

4.4. Animation Composition

The process of animating a long queue of pedestrians by creating a sequence of interconnected patches using the data stored in the periodicity and connectivity matrices is shown in Figure 7. Once a sequence of interconnected patches with low connectivity cost between every two adjacent patches is found, trajectories inside these patches are edited twice: They are first deformed to make ensure the periodicity in time, and the second pass of deformations ensure that transitions between adjacent patches are continuous in both time and space. If necessary, a global deformation can be done to a patch so that it fits the shape of the target patch inside the virtual environment. This enables us to impose the queue to follow a specific path (e.g., in Figure 7, the patches are deformed so that the long queue follows a huge circular path).

In this illustrative example, we created and animated motions of a hug crowd. Since all trajectories have been prerecorded and processed, creating the animation does not require addition computation. Our method is not only efficient, the resulting animation is also realistic both at the individual level and at the global scale. Furthermore, we managed to capture successfully the stop-and-go waves. In the next sec-

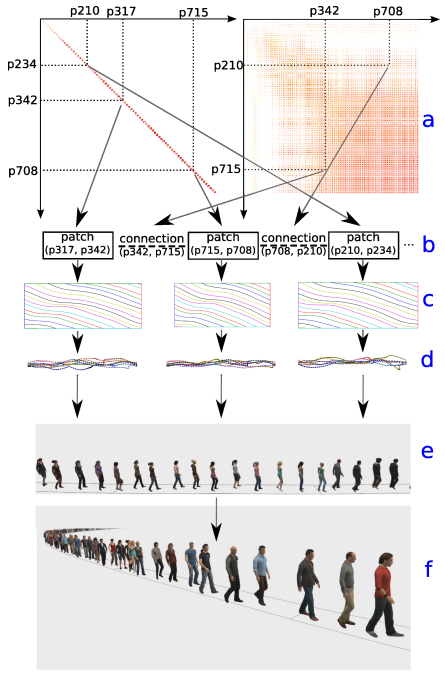

Figure 7: *Composing animation of a long queue of pedestrians: a) The connectivity and periodicity matrices are looked up to extract patches and connections between them. b) A sequence of connected patches. c) The patches displayed in the angle-time plots. d) The 2D spatial representation of the trajectories inside the patches. e) The trajectories are used to animate a long queue of pedestrians. f) Patches are globally deformed so that the pedestrians follow a circular path. Note that we can compose very long queues because our solution is only limited by the processing capacity of GPU.*

tion, we demonstrate that our approach remains expressive and efficient in more complex situations.

5. Results

5.1. 3-Lane Crossing

For this example, we use the motion capture data of people moving along three intersecting lanes as shown in Figure 8. Even though the trajectories are quite complex as the pedestrians have to avoid head to head collisions in addition to keep suitable distance with the pedestrian ahead, we still succeed in extracting almost periodic motions. After choosing square as the patch polygon, multiple patches were extracted from the motion capture data, for instance, one of them is shown at the top-right corner of Figure 8. These patches can then be combined to generate an animation where virtual pedestrians walk along a virtual wire netting consisting of parallel lines (at angle 0, 120, or 240 degrees to the horizontal axis). This example, although not the most natural one, demonstrates that a large-scale animation can be constructed from some snips of trajectories of captured motions and the interactions between the virtual pedestrians faithfully reproduce real world interactions. Furthermore, more complex scenarios can be built by combining various sources. For example, the companion video contains one example where three-lane crossing patches are combined with 1D queueing data.

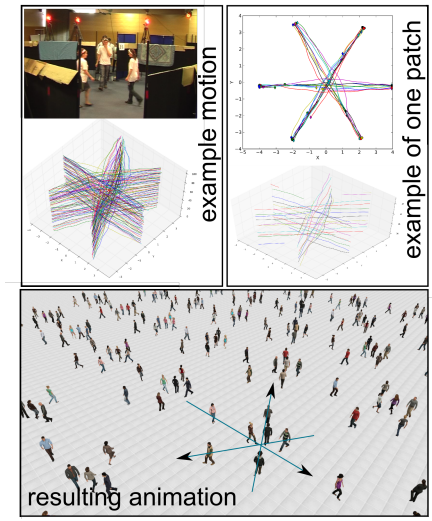

Figure 8: *Complex crossing data. Top-left: picture from a motion capture session, where pedestrians start from three different positions and then exit the area on the opposite side. Top-right: a patch extracted from these motion capture data. Bottom: snapshot of a large environment populated with the computed patches.*

5.2. Interactions of 24 People

In this example, we consider data from three motion capture sessions where 24 pedestrians standing in circle were asked to move to points which are diametrically opposite to their initial positions as shown in Figure 9. Our method maintain the very complex interactions between the pedestrians and reuse them to animate virtual characters. The large virtual environment shown in Figure 9 is populated with three different patches computed from the three distinct examples of interactions between the 24 pedestrians. Once again, this example is not designed to synthesize a realistic global behaviour, even though the cloned motions are as natural as the ones in the motion capture data. The virtual crowd inherits complex interactions that could be hard to obtain with crowd simulation techniques, while the computational cost is again the lowest possible. Furthermore, the animations are not limited in neither space nor time as patches can be added whenever they become visible from the camera.

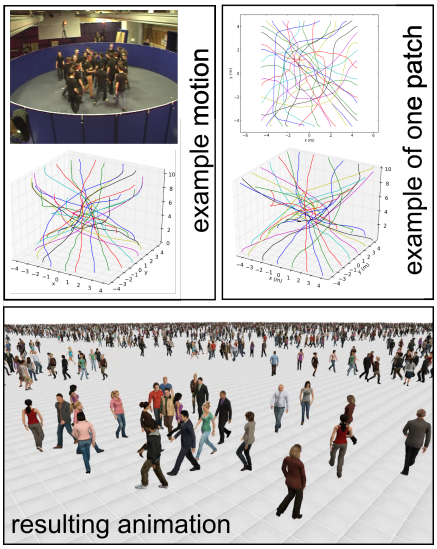

Figure 9: *Cloning complex crossing data. Top-left: an initial set of crowd motion data, where the trajectories of the 24 pedestrians cross each other. Top-right: an extracted patch. Bottom: snapshot of a large environment populated with the computed patches.*

6. Discussions

6.1. Decomposition of Motion Data

Our method requires that the user chooses the data decomposition method as well as the discretization parameters of the search space, because the search problem is intractable without this simplification. Since the size and duration of patches depend on the phenomenon in data the user wants to clone and it is much easier in practice to only use basic shapes (e.g., squares, equilateral triangles, etc.), it is quite intuitive for the user to provide decomposition parameters, even if a minimum expertise is required. Instead of asking the user to provide the decomposition parameters, a more flexible solution consists in letting the user sketch some approximate properties for patches (e.g., shape, size, and duration) and then searching in the neighbourhood of the sketched properties. This additional degree of freedom in the search process would greatly extend the solution space, and improve data periodicity and connectivity.

6.2. Improving Periodicity and Connectivity

When data hardly exhibits almost periodic portions or almost connectivity between patterns, additional degrees of freedom should be explored. Instead of simply playing on local deformations as described in the previous section, periodicity and connectivity can be improved by removing supernumerary trajectories. For example, when attempting to connect two patterns and the numbers of inputs and outputs are not equal, instead of rejecting the connection, the supernumerary trajectories that lead to the rejection can be removed in order to improve the connectivity. Similarly, additional trajectories (e.g., synthetic ones obtained by simulation) can be added to compensate for the differences. Lastly, the provision of a metric is necessary to quantify the impact of the removal/addition.

6.3. Motion Continuity

Our method ensures C^0 continuity; nevertheless, it can be easily improved in two ways to ensure C^1 continuity of trajectories. The first solution is to take into account C^1 continuity during the search stages by extending f_c and f_p functions with the derivatives $\dot{q}_i$ of character states with the aim of giving preference to patches and connections with the best C^1 continuities. A second solution is to ensure C^1 continuity at the stage of trajectory editing (i.e., the stage when trajectories are made periodic / interconnected). Furthermore, these two solutions can be combined. However, these additional constraints may prevent the method from providing a wide variety of solutions. Despite the fact that C^1 continuity may be desired for low density motions when pedestrians may move relatively fast, we argue that C^0 continuity is enough in the case of high density motions such as our demos.

6.4. Combining Cloned Patches with Synthetic Crowd Patches

The patches we created in this paper were extracted from the example data, in contrast to the crowd patch paper [YMPT09], where the patches were generated from patterns. It is straightforward to combine data-based patches with synthetic patches in a single environment: Data-based patches can be spread over the environment and the gaps can be filled in by synthetic patches. Furthermore, as mentioned above, one could also mix data-based trajectories with synthetic trajectories in a single patch.

6.5. Analogies with Texture Synthesis and Motion Graph Techniques

Even though our technique has similarities with both texture synthesis and motion graph techniques, there are some fundamental differences between the techniques. Compared to texture synthesis techniques, we are working with punctual continuity constraints and our material is human motion (with many specificities about the way they should be edited and a great sensitivity of spectators to artifacts). Instead of preserving a single character motion as done by motion graphs techniques, we preserve interactions between multiple characters. This difference has a profound impact on the nature of similarity between states and therefore the exploration of the solution space. For these reasons, we do not directly refer to those two problems in our paper.

7. Conclusions

We have presented a new method that can create large-scale crowd animations with unlimited duration by extracting patches from existing data that is limited in both space and time.

Our method has several advantages compared to previous methods that consider and duplicate individual and local relations. Firstly, it can populate large-scale virtual environments on the fly with artifact-free motions, because our method relies on the technique of crowd patches [YMPT09] and hence it replaces computationally expensive animations with data replays. This is of critical importance because large-scale virtual environments are becoming increasingly important for architecture design, urban planning, entertainment etc. Secondly, our method allows for easy usage by novices because it follows the familiar copy-and-paste metaphor. Thirdly, macroscopic group structures can be reproduced through the selection of specific crowd phenomenons in example data, and hence the anticipated results can be easily achieved without the burden of crowd modelling and simulation tuning steps, which are always difficult and chaotic by nature. Finally, the method intrinsically generates realistic animations when real motion data is used as the input.

We have demonstrated the potential of our method through various examples. Unfortunately, crowd motion data is still a rare and expensive resource. To overcome this lack of data problem, we intend to combine the original crowd patches approach with our crowd motion cloning method in order to handle the large variety of challenges originated from designers' intentions, data-sets, and environments. Furthermore, since we only use a limited set of pedestrian models at moment, each model has to be used many times to clone virtual characters when simulating large crowds. We should add accessories, texture variation, and decals to disguise clones [MLD*08].

References

[DHOO05] DOBBYN S., HAMILL J., O'CONOR K., O'SULLIVAN C.: Geopostors: A Real-Time Geometry/Impostor Crowd Rendering System. *ACM Transactions on Graphics 24*, 3 (2005). 2

[GCC*10] GUY S. J., CHHUGANI J., CURTIS S., DUBEY P., LIN M., MANOCHA D.: Pledestrians: a least-effort approach to crowd simulation. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, Switzerland, 2010), SCA '10, Eurographics Association, pp. 119–128. 2

[GN08] GREMINGER M. A., NELSON B. J.: A deformable object tracking algorithm based on the boundary element method that is robust to occlusions and spurious edges. *Int. J. Comput. Vision 78*, 1 (jun 2008), 29–45. 5

[GSN04] GREMINGER M., SUN Y., NELSON B.: Boundary element deformable object tracking with equilibrium constraints. In *Proceedings of the 2004 IEEE International Conference on Robotics and Automation* (may 2004), vol. 4, pp. 3896–3901. 5

[HLLO10] HUERRE S., LEE J., LIN M., O'SULLIVAN C.: Simulating believable crowd and group behaviors. In *ACM SIGGRAPH ASIA 2010 Courses* (New York, NY, USA, 2010), SA '10, ACM, pp. 13:1–13:92. 2

[HM95] HELBING D., MOLNÁR P.: Social force model for pedestrian dynamics. *Phys. Rev. E 51* (may 1995), 4282–4286. 2

[JCP*10] JU E., CHOI M. G., PARK M., LEE J., LEE K. H., TAKAHASHI S.: Morphable crowds. *ACM Trans. Graph. 29*, 6 (dec 2010), 140:1–140:10. 2, 3

[JHS07] JOHANSSON A., HELBING D., SHUKLA P.: Specification of the social force pedestrian model by evolutionary adjustment to video tracking data. *ADVANCES IN COMPLEX SYSTEMS 10* (2007), 271. 2

[KHKL09] KIM M., HYUN K., KIM J., LEE J.: Synchronized multi-character motion editing. *ACM Trans. Graph. 28*, 3 (jul 2009), 79:1–79:9. 1, 4

[KLLT08] KWON T., LEE K. H., LEE J., TAKAHASHI S.: Group motion editing. *ACM Trans. Graph. 27*, 3 (aug 2008), 80:1–80:8. 1, 2, 3

[KO10] KARAMOUZAS I., OVERMARS M.: Simulating the local behaviour of small pedestrian groups. In *Proceedings of the 17th ACM Symposium on Virtual Reality Software and Technology* (New York, NY, USA, 2010), VRST '10, ACM, pp. 183–190. 2

[KOOP11] KULPA R., OLIVIERXS A.-H., ONDŘEJ J., PETTRÉ J.: Imperceptible relaxation of collision avoidance constraints in virtual crowds. *ACM Trans. Graph. 30*, 6 (dec 2011), 138:1–138:10. 2

[KSA*09] KAPADIA M., SINGH S., ALLEN B., REINMAN G., FALOUTSOS P.: Steerbug: an interactive framework for specifying and detecting steering behaviors. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2009), SCA '09, ACM, pp. 209–216. 2

[KSH*12] KAPADIA M., SINGH S., HEWLETT W., REINMAN G., FALOUTSOS P.: Parallelized egocentric fields for autonomous navigation. *The Visual Computer* (jan 2012), 1–19. 10.1007/s00371-011-0669-5. 2

[KSHF09] KAPADIA M., SINGH S., HEWLETT W., FALOUTSOS P.: Egocentric affordance fields in pedestrian steering. In *Proceedings of the 2009 symposium on Interactive 3D graphics and games* (New York, NY, USA, 2009), I3D '09, ACM, pp. 215–223. 2

[LCHL07] LEE K. H., CHOI M. G., HONG Q., LEE J.: Group behavior from video: a data-driven approach to crowd simulation. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, Switzerland, 2007), SCA '07, Eurographics Association, pp. 109–118. 2

[LCL07] LERNER A., CHRYSANTHOU Y., LISCHINSKI D.: Crowds by example. *Computer Graphics Forum (Proceedings of Eurographics) 26*, 3 (2007). 2

[LFCCO09] LERNER A., FITUSI E., CHRYSANTHOU Y., COHEN-OR D.: Fitting behaviors to pedestrian simulations. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2009), SCA '09, ACM, pp. 199–208. 2

[MH03] METOYER R. A., HODGINS J. K.: Reactive pedestrian path following from examples. In *Proceedings of the 16th International Conference on Computer Animation and Social Agents* (may 2003), pp. 149–156. 2

[MLD*08] McDonnell R., Larkin M., Dobbyn S., Collins S., O'Sullivan C.: Clone attack! perception of crowd variety. *ACM Trans. Graph. 27*, 3 (aug 2008), 26:1–26:8. 9

[OPOD10] Ondřej J., Pettré J., Olivier A.-H., Donikian S.: A synthetic-vision based steering approach for crowd simulation. *ACM Trans. Graph. 29*, 4 (jul 2010), 123:1–123:9. 2

[PAB07] Pelechano N., Allbeck J. M., Badler N. I.: Controlling individual agents in high-density crowd simulation. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, Switzerland, 2007), SCA '07, Eurographics Association, pp. 99–108. 2

[PCM*06] Pettré J., Ciechomski P. d. H., Maïm J., Yersin B., Laumond J.-P., Thalmann D.: Real-time navigating crowds: scalable simulation and rendering: Research articles. *Comput. Animat. Virtual Worlds 17*, 3-4 (jul 2006), 445–455. 2

[POO*09] Pettré J., Ondřej J., Olivier A.-H., Cretual A., Donikian S.: Experiment-based modeling, simulation and validation of interactions between virtual walkers. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2009), SCA '09, ACM, pp. 189–198. 2

[PPD07] Paris S., Pettré J., Donikian S.: Pedestrian reactive navigation for crowd simulation: a predictive approach. *Comput. Graph. Forum* (2007), 665–674. 2

[Rey87] Reynolds C. W.: Flocks, herds and schools: A distributed behavioral model. *SIGGRAPH Comput. Graph. 21*, 4 (aug 1987), 25–34. 2

[Rey99] Reynolds C. W.: Steering behaviors for autonomous characters. In *Proceedings of Game Developers Conference* (1999). 2

[SKSY08] Shum H. P. H., Komura T., Shiraishi M., Yamazaki S.: Interaction patches for multi-character animation. *ACM Trans. Graph. 27*, 5 (dec 2008), 114:1–114:8. 2, 3

[TCP06] Treuille A., Cooper S., Popović Z.: Continuum crowds. *ACM Trans. Graph. 25*, 3 (jul 2006), 1160–1168. 2

[TLG11] Torrens P., Li X., Griffin W. A.: Building agent-based walking models by machine-learning on diverse databases of space-time trajectory samples. *Transactions in GIS 15* (jul 2011), 67–94. 2

[UCT04] Ulicny B., Ciechomski P. d. H., Thalmann D.: Crowdbrush: interactive authoring of real-time crowd scenes. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, Switzerland, 2004), SCA '04, Eurographics Association, pp. 243–252. 2

[YMPT09] Yersin B., Maïm J., Pettré J., Thalmann D.: Crowd patches: populating large-scale virtual environments for real-time applications. In *Proceedings of the 2009 symposium on Interactive 3D graphics and games* (New York, NY, USA, 2009), I3D '09, ACM, pp. 207–214. 1, 3, 8, 9