EG UK Theory and Practice of Computer Graphics (2008)
Ik Soo Lim, Wen Tang (Editors)

Efficient Path Matching Motion Generation Algorithm for Multi Agent Environment

Newman Lau, Chapmann Chow, Pouro Lee, Bartholomew Iu

Multimedia Innovation Centre, School of Design, The Hong Kong Polytechnic University
newman.lau@polyu.edu.hk, sdchap@polyu.edu.hk, sdpouro@polyu.edu.hk, sdbart@polyu.edu.hk

Abstract

This paper presents an efficient method to generate a large amount of continuous motion data from motion captured data. Given user defined way point lists for each agent, the algorithm can automatically generate collision free walking paths for them. The walking path data is then analyzed and transformed into a sequence of agent states such as walking or standing state. Based on the randomized depth first algorithm, the agent states are matched with a sequence of corresponding motion clips. The final motion is obtained by blending the motion clips to fit with the speed of the agents' walking paths. From our experiments, our algorithm generates natural looking motions.

Categories and Subject Descriptors: I.6 Simulation and Modeling

1. Introduction

This paper presents an efficient method to generate a large amount of continuous motion data from existing motion captured data library. The method is developed for a large crowd simulation project which simulates motions of a crowd of agents in a virtual environment. The method can generate natural looking motion by retaining the details of original captured motion.

Our method starts by capturing sufficient quantity of motion data. The motion data is then divided into small pieces called motion clips and these motion clips are used to construct the *motion graph* for motion planning of the agents. Based on the *crowd behavior algorithm*, the agents avoid collision while they follow the user defined way point list to generate walking paths. The agents' walking paths are then analyzed to create a list of agent states. By applying a randomized depth first search algorithm, we find out a sequence of motion clips that matches with the agent state. Finally, the motion clip sequence is blended together to best fit with the speed of the path of the agents. Since the motion clip has a speed difference from the path, foot skating problem occurs. We choose a blending algorithm that can handle the foot skating problem.

2. Related Works

Motion synthesis in crowd simulation has been studied by many previous researchers for many years. Several algorithms of procedural motion synthesis [BMTT90, BUT04, BC89] were designed to generate controllable walking motions. These algorithms can generate motions matching exactly with arbitrary walking paths. However, these algorithms cannot utilize the motion captured data and generate realistic motions. Also, these algorithms can generate walking motions but lack the flexibility to adapt to different categories of motions.

Another approach is graph based motion synthesis. Graph based algorithms generates motions by connecting motion clips from the motion graph. Different algorithms have been proposed [AFO03, HGP04]. One similar algorithm to our algorithm is [SKG05]. This algorithm uses a fast path planner based on probabilistic roadmaps to navigate through the environment, and the algorithm produces motion clips that are approximately satisfy the pose, position, orientation and time constraints. Then, by adjusting the motion clips, it synthesizes motions that exactly satisfy the constraints while avoiding collision with the environment.

3. Motion data pre-processing

Our implementation utilizes motion data from the motion capture system. In our current implementation, we only focus on two main categories of motion data: standing motions and walking motions. Transition categories, such as standing-to-walking and walking-to-standing motions, are also captured as the connection edges in the motion graph which will be presented in section 3.3. Sufficient transition motions must be captured to ensure the completeness of the motion graph. In general, our method would be extended for handling more motion categories, such as sitting or jumping motions. In section 6, we will discuss the extension in more details.

3.1 Motion clips preparation

The captured motion data is first divided into small units called "motion clips". The aim of preparation of motion clips is to build a motion graph for searching suitable motions to the agents. These motion clips form the basic building blocks of the motion graph. Each motion clip has four attributes: the starting frame, ending frame, starting category and ending category. For walking motion clips, both starting and ending categories are "Walk". For standing motion clips, both are defined "Stand". For standing-to-walking transition motion clips, the starting category is "Stand" and the ending category is "Walk". The walking-to-standing transition motions are also similarly defined.

The starting frame and ending frame is the frame number of the original captured motion data. A problem needed to be solved would be how to choose the starting and ending frame of each motion clip. We use the error function

$$D = \min_{\theta, x_0, z_0} \sum_i w_i \left\| \mathbf{p}_i - \mathbf{T}_{\theta, x_0, z_0} \mathbf{p}'_i \right\|^2 \qquad (1)$$

to calculate the similarity between two postures. High error value means the postures are not similar and they should not be connected. To solve the problem, we define *resting postures* which are postures common in motion clips. The starting and ending frame are chosen at resting postures. If the end of a clip and the start of another clip share the same resting posture, the error value will be small. The connectivity between the two motion clips can thus be guaranteed. Examples of resting postures are putting two arms down or putting two arms in the front.

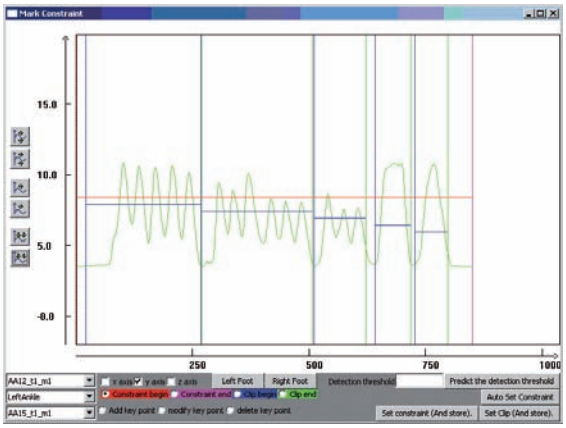

Figure 1: *Preparation of motion clips and setting of constraint*

3.2 Setting foot constraints

One problem of motion synthesis is to match the motion with the speed realistically. If the speed of the agent does not match with the speed of the motion data, the feet of the agent will look they are skating on the ground. In our implementation, we set constraints on motion clips at the frames which the feet must stick to the ground [IAF06]. During the motion blending stage in section 4.4, the foot position will be fixed to the ground even if the speed of motion clip does not match with the path speed. The foot skating problem will be discussed in more detail in section 4.4.

3.3 Building motion graph

The last pre-processing stage is to generate a motion graph [KGP02] using the motion clips. In our implementation, we have put additional conditions upon motion clip connectivity. To connect two motion clips in the motion graph, these motion clips must satisfy two conditions. The first condition is the posture similarity condition. We use the error function (1) as a measure of similarity. The error value between the ending posture of one motion clip and the starting posture of the other motion clip must be less than or equal to a predefined threshold value. The second condition is the category condition. The ending category of one motion clip must be the same as the starting category of the other motion clip.

The motion graph building process continues until all combination of starting and ending frame of motion clips are compared. Upon completion of the process, a directed graph would be created. If a starting or ending frame is chosen such that the motion clip cannot connect to other clips, the motion graph would generate dead end node, which means no connection to other nodes. To ensure all motion clips being matched to an agent can be followed by another motion, Tarjan's algorithm [Tar72] is applied to remove all dead end nodes. The resulting motion graph is a strongly connected graph connecting all categories of motions.

One problem of building motion graph is that the motion graph building process may generate fragmented motion graphs. Another problem is the dead end removal process may eliminate useful motion clips from the motion graph. In order to avoid these problems, sufficient motion clips are prepared for ensuring the connectivity of the motion graph.

4. Implementation

The automatic motion synthesis method can be divided into 4 stages:

1. Generation of walking path
2. Analysis of walking path
3. Search of motions from the motion graph
4. Blending and resolve foot skating problem

The following sub sections describe these stages in more details.

4.1 Generation of walking path

In our simulation system, the user can define a way point list to direct the agents to follow. The user can also define the time at which the agents must reach a specific way point. During the simulation, the agents try to move along the way point list. Based on the crowd behavior algorithms proposed in [Rey99], the agents can avoid static obstacles (such as walls) or dynamic obstacles (such as neighboring agents) while they move. The simulation system will record the locomotion data of the agents for every frame of the simulation. The recorded data forms the final walking path.

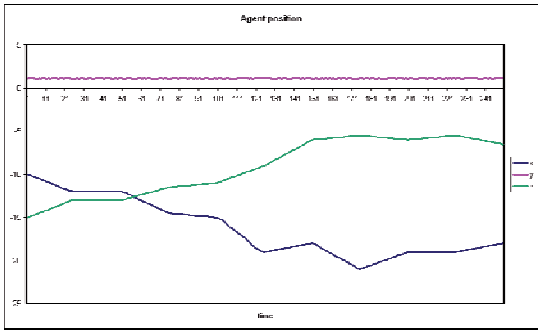

Figure 2: *Position of Agent*

In order to synthesize realistic motion, the physical attributes of the agents must be carefully configured. The maximum speed and maximum turning angle are limited. Otherwise, the agent may move or turn too fast that the motion data cannot match with.

4.2 Analysis of walking path

The locomotion data of the actual walking path includes the position and orientation of the agents at each frame. Let the position at frame time t be p_t and the frame rate per second be fps, the speed v_t is then calculated from the position data using equation (2).

$$v_t = \begin{cases} (p_t - p_{t-1}) * fps & for \quad t > 0 \\ 0 & for \quad t = 0 \end{cases} \qquad (2)$$

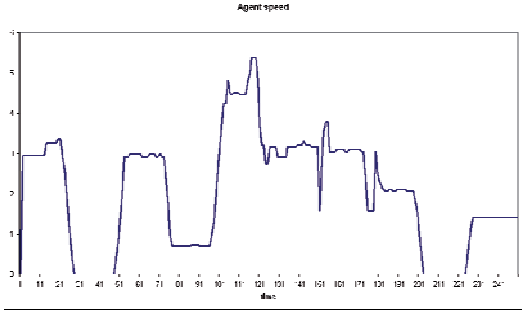

Figure 3: *Speed of agent*

Let agent state at frame time be a_t. The speed data is then transformed into agent state data by the following criteria:

if $v_t > 0$, $a_t = WALK$
if $v_t = 0$, $a_t = STAND$

In crowd environment, agents may collide with each other. As a result, the agents may stand for short intervals in walking paths. Walking motions should still be applied in these short intervals. To distinguish these short stop intervals from normal stop, if the stop interval is shorter than a user defined threshold, the stand state would be converted to walking state. A reasonable length of threshold could be the minimum time required by two transition motions, from walk to stand and from stand to walk. After conversion, several broken walking states could be combined into one longer walking state. Short intervals of walking states are converted using the similar method. Finally, an agent state sequence can be generated.

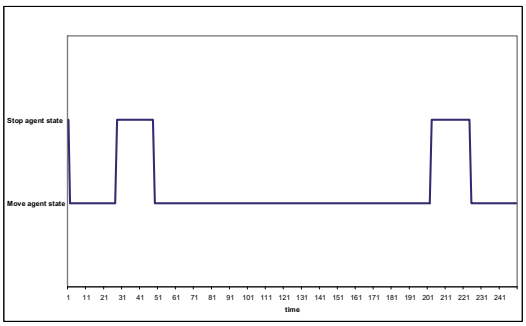

Figure 4: *Agent state*

4.3 Search of motion from the motion graph

When the sequence of the agent state segments is prepared, the next task is to search motion clips for the segments. The motion searching process is an iteration process starting from the first segment to the last segment. For each segment, a randomized depth first searching algorithm is applied to search motion clips from the motion graph. Randomization is necessary because we do not want to produce agents having synchronized motion sequence. A group of synchronized agents behave like robot army more than human beings. The randomized depth first searching algorithm is illustrated in figure 5.

The objective of this searching algorithm is to return a motion clip sequence containing n motion clips. Among these motion clips, the first *n - 1* are *homogeneous motion clips* and the last one is a transition motion clip. Homogeneous motion clips means all of the motion clips have the same category with the current agent state segment. The homogeneous motion clips must not be longer than the segment to avoid extension of incorrect motion into next segment. However, the transition motion clip must overlap with the boundary between the current segment and next segment. It can assure the smoothness of transition between two agent states.

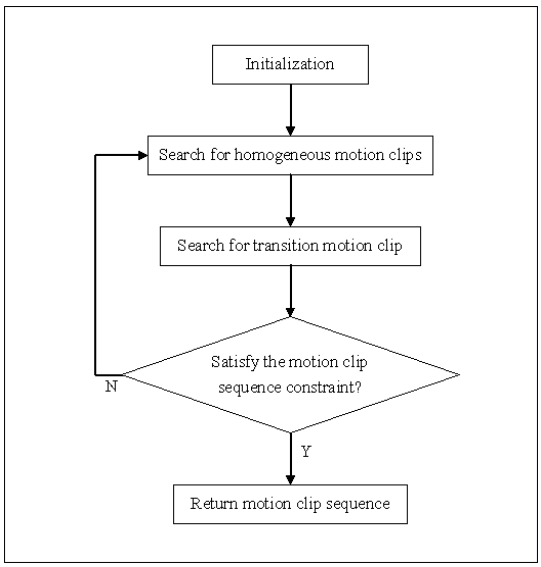

Figure 5: *Searching algorithm*

The following subsections discuss the steps of the searching algorithm. For purpose of explanation, we define the length of agent state segment i to be t_i and the length of motion clip j to be t_j.

4.3.1 Initialization

In the initialization step, we create a searching tree to store the potential motion clips. Initially, the tree is empty and the current node points to the root of the tree.

4.3.2 Search for a homogeneous motion clips

In the next step, we search for a sequence of homogeneous motion clips. First, we query from motion graph a list of motion clips which connects to the last queried motion clip. If the last queried motion clip does not exist, the search space is the whole motion graph. These motion clips must satisfy the following constraints.

- *starting category = agent state of current segment*
- *ending category = agent state of current segment*

The query result is then added to the searching tree as child nodes. We randomly select a child node from the tree one at a time. If the child node satisfies condition (3), set the child node as the current node and recursively query next motion clip.

$$\sum_{j}^{M} t_j < t_i \quad (3)$$

, where M = total number of motion clips

Otherwise, remove the child node and go to next step to search for transition motion clip.

The aim of this step is to generate a longest homogeneous motion clip sequence that is still shorter than the agent state segment. As stated in the condition (3), if the total length of the motion clips is shorter than the current segment, this step continues to append motion clips. Once a motion clip is added to make the homogeneous motion clip sequence longer than the segment, the sequence excluding this clip is the longest sequence we aim to generate.

4.3.3 Search for transition motion clip

After the generation of the homogeneous motion clip sequence, we search for the transition motion clip. The transition motion clip will be placed at last in the motion clip sequence. In order to overlap with the boundary between the current segment and the next segment, the transition motion must start in the current segment and end in the next segment. First, we query a list of motion clips from the subset of motion graph which connects to last homogeneous motion clip. These motion clips must satisfy the following constraints.

- *starting category = agent state of current segment*
- *ending category = agent state of next segment*

We then randomly select a motion clip from the list one at a time. If the condition (4) is satisfied, stop searching and return the final motion clip sequence.

$$\sum_{j}^{M} t_j + t_{transition} > t_i \quad (4)$$

, where M = total number of motion clips

If none of the transition motion clip satisfies the condition (4), continue the search for homogeneous motion clip sequence from the searching tree.

4.4 Blending and resolve foot skating problem

Although the motion clip matches the agent state of each segment, the speed of the matched motion clips may not exactly be equal to the speed of the agent. The feet of the agents may seem to slide on the ground. This problem can be solved by adjusting the foot position in the motion blending stage.

We use the blending algorithm mentioned in [LICL08]. This algorithm is a single pass, seven steps algorithm. The input to the algorithm is the walking path and the motion list prepared in previous sections, and the output is the blended motion without foot skating problems and having the speed of the generated motion matched with the path. We choose this algorithm because it has several advantages, they are:

- The blending algorithm can be seamlessly integrated with our implementation;
- The blending algorithm is an on-line algorithm, this can reduce the time needed for the whole crowd simulation;
- The blending algorithm works with a motion graph with relatively small set of motion clips.

The blending algorithm works as follows. First, the algorithm calculates a *skeleton center* and matches this center with the speed of the path. The author claims that using this center is better than using the skeleton root or the skeleton center of mass as the skeleton center can keep the fine movement of the skeleton hip. From the skeleton center, the real skeleton root is calculated. Then the rotational data of the upper body is blended by a cubic interpolation equation, while the rotational data of the lower body is blended with a walking motion clip scaling step. The next step is to perform a center pull in order to ensure the generated motion is on the path. The final step is to snap the foot on the group. The author uses a simple measure to determine when the foot should be snapped: when the ankle touches the ground, then the ankle should be snapped to the group until the ankle is pulled upward. We find that a more sophisticated measure mentioned in [KSG02], which considers the heel and/or ball in the calculation, can be used to replace the final step. We, however, use the approach in [LICL08] because a simple measure is enough in our implementation.

5. Result

We have applied the algorithm in our crowd simulation system. We have captured over 130 motion data using motion capture system. From these data, we have generated a motion graph containing more than 300 motion clips. The length of motion clips ranges from about 10 frames to more than 500 frames.

	Number of Agents	Average speed	Weighted average speed deviation	Average number of segments
1	10	2.98m/s	0.31m/s	1.00
2	10	0.00m/s	0.00m/s	1.00
3	10	2.15m/s	0.24m/s	4.00
4	100	2.76m/s	0.55m/s	1.23
5	100	1.77m/s	0.74m/s	4.56
6	500	1.14m/s	1.37m/s	4.81

Table 1: *Experimental result (part 1)*

	Extended homogeneous motion percentage	Overlapped transition motion percentage
1	0%	100%
2	0%	100%
3	0%	100%
4	0%	100%
5	0%	100%
6	0%	100%

Table 2: *Experimental result (part 2)*

Table 1 and Table 2 show the result of several experiments. The duration is 60 seconds in all experiments. The average speed column is the average speed of movement of all agents. The Weighted average speed deviation column is the weighted average deviation between the actual speed of movement and the speed of searched motion clip of all agents. It is calculated from the equation (5)

$$\frac{1}{n}\sum_{j}^{n}\left(\frac{1}{\sum_i w_i}\sum_i w_i \left\| vm_i - vc_i \right\|\right) \quad (5)$$

,where w_i = length of the i^{th} motion clip, vm_i = actual speed of movement, vc_i = speed of the i^{th} motion clip and n = number of agents

The average number of segments column is the average of number of agent state segments of all agents. The extended homogeneous motion percentage column is the percentage of homogeneous motion clips extended to other agent state segments. The overlapped transition motion percentage column is the percentage of transition motion overlapped with the boundary between two segments.

In experiment 1 and 4, the agents are directed to move for the whole simulation. In experiment 1, since the environment is not congested, the agents can move with a higher speed without collision with each others. As shown in table 1, the agents have only one agent state segment, that is, the walking state, during the simulation. In experiment 4, the environment is much more congested compared with experiment 1. The agents may collide with each others, causing them to stop and wait for several frames. As a result, some agents may generate standing state segments in their walking path. As shown in table 1, the average number of agent state segment is 1.23. The speed deviations in both experiments are relatively low because the agents move with normal speed and walking motion clips of normal speed are matched with them.

In experiment 2, the agents are directed to stand for the whole simulation. The only agent state segment is the standing state and the standing motion clips are matched. The speed deviation = 0 because the speed of both the agents and the matched motion clips are 0.

In experiment 3, 5 and 6, the agents are directed to move in a walk-stand-walk-stand fashion. As the environment becomes more congested, the agents need to stop and wait for neighbors to pass. This can be reflected in the columns of average speed of agents and average number of segments. The average speed decreases and the average number of segments increases as the number of agents increases. The speed deviation is relatively high, but is still acceptable for generating animation.

As shown in table 2, our algorithm can successfully generate transition motions which overlap with the boundary between two segments for all experiment. The runtime result demonstrates that the transition motions are animated smoothly. Also, our algorithm can successfully generate homogeneous motion clips which do not extend to

other segments. The runtime result demonstrates that the motion clips are correctly matched with the agent states. Our algorithm reduces the foot skating effect by adjusting the foot position in the blending stage. The resulting animations of all experiments look realistic and natural.

6. Discussion

As stated in the beginning of the paper, our current implementation only focuses on the walking motions and the standing motions. As an extension, we can improve this method to handle other types of motion categories, such as jumping and sitting motions. In the stage of generation of walking path, we can allow the user to define periods of jumping or sitting motions. In the stage of analysis of walking path, prioritized agent state mechanism can be applied during the analysis to generate specific agent state segments such as jumping or sitting segments. The remaining steps of our algorithm are not required to change, since they are ready for such extension.

Another possible extension of the method is to add other behavior constraints in the motion library. For example, we can add emotion as an additional constraint in the motion library. When we query for a motion from the motion graph, in addition to the starting category and the ending category constraints, we can add an emotion constraint. For example, we can query for a "walking happily" motion or a "standing angrily" motion from the motion graph.

References

[AFO03] ARIKAN O., FORSYTH D. A., O'BRIEN J. F.: Motion synthesis from annotations. *ACM Transactions on Graphics 22*, 3 (2003), 402–408.

[BMTT90] BOULIC R., MAGNENAT-THALMANN N., THALMANN D.: A global walking model with real-time kinematic personification. *Visual Computer 6*, 6 (1990), 344–358.

[BUT04] BOULIC R., ULICNY B., THALMANN D.: Versatile walk engine. *Journal of Game Development 1*, 1 (2004).

[BC89] BRUDERLIN A., CALVERT T.: Goal-directed, dynamic animation of human walking. In *Proc. of ACM SIGGRAPH 1989* (July 1989), Annual Conference Series, ACM SIGGRAPH, pp. 233–242.

[GLE01] GLEICHER M.: Motion path editing. In *I3D '01: Proc. of the 2001 symposium on Interactive 3D graphics*, (2001), pp. 195–202.

[GR96] GUO S., ROBERGE J.: A high-level control mechanism for human locomotion based on parametric frame space interpolation. In *Proc. of Eurographics Workshop on Computer Animation and Simulation '96* (Aug. 1996), pp. 95–107.

[HGP04] HSU E., GENTRY S., POPOVIĆ J.: Example-based control of human motion. In *Proc. of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2004), pp. 69–77.

[IAF06] IKEMOTO L., ARIKAN O., FORSYTH D.: Knowing when to put your foot down. In *I3D '06: Proc. of the 2006 symposium on Interactive 3D graphics and games*, (2006), pp. 49–53.

[KGP02] KOVAR L., GLEICHER M., PIGHIN F.: Motion graphs. ACM Transactions on Graphics 21, 3 (2002), 473–482.

[KSG02] KOVAR L., SCHREINER J., GLEICHER M.: Footskate cleanup for motion capture editing. In *Proc. of the ACM SIGGRAPH symposium on Computer animation*, pp. 97–104. ACM Press, 2002.

[LK05] LAU M., KUFFNER J. J.: Behavior planning for character animation. In *SCA '05: Proc. of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 271–280, 2005.

[LICL08] LAU N., IU B., CHOW C., Lee P.: Motion synthesis with adaptation and path fitting. In *Computer Graphics and Artificial Intelligence, 2008*. To appear.

[LCL06] LEE K. H., CHOI M. G., LEE J.: Motion patches: building blocks for virtual environments annotated with motion data. *ACM Trans. Graph 25*, 3 (2006), pp.898–906.

[LHP05] LIU C. K., HERTZMANN A., POPOVIC Z.: Learning physics-based motion style with nonlinear inverse optimization. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, pages 1071–1081, 2005.

[PSS04] PARK S. I., SHIN H. J., SHIN S. Y.: Online motion blending for real-time locomotion generation. In *Computer Animation and Virtual Worlds 15* (2004), pp. 125–138.

[PLS03] PETTRÉ J., LAUMOND J.-P., SIMÉON T.: A 2-stages locomotion planner for digital actors. In *SCA '03: Proc. of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2003), pp. 258–264.

[Rey99] REYNOLDS C. W.: Steering behaviors for autonomous characters. In *Game Developers Conference*, 1999, pp 763–782.

[SHP04] SAFONOVA A., HODGINS J. K., POLLARD N. K.: Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, (2004) pp. 514–521.

[SKG05] SUNG M., KOVAR L, GLEICHER M.: Fast and accurate goal-directed motion synthesis for crowds. In *SCA '05: Proc. of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 291–300, 2005.

[Tar72] TARJAN R.: Depth-first search and linear graph algorithms. *SIAM Journal on Computing 1*, 2 (1972), pp. 146–160.