Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2012)
P. Kry and J. Lee (Editors)

Environment-aware Real-Time Crowd Control

Joseph Henry[1] Hubert P. H. Shum[2] and Taku Komura[1]

[1]University of Edinburgh, United Kingdom
[2]University of Worcester, United Kingdom

Abstract

Real-time crowd control has become an important research topic due to the recent advancement in console game quality and hardware processing capability. The degrees of freedom of a crowd is much higher than that provided by a standard user input device. As a result most crowd control systems require the user to design the crowd movements through multiple passes, such as first specifying the crowd's start and goal points, then providing the agent trajectories with streamlines. Such a multi-pass control would spoil the responsiveness and excitement of real-time games. In this paper, we propose a new, single-pass algorithm to control crowds using a deformable mesh. When controlling crowds, we observe that most of the low level details are related to passive interactions between the crowd and the environment, such as obstacle avoidance and diverging/merging at cross points. Therefore, we simplify the crowd control problem by representing the crowd with a deformable mesh that passively reacts to the environment. As a result, the user can focus on high level control that is more important for context delivery. Our algorithm provides an efficient crowd control framework while maintaining the quality of the simulation, which is useful for real-time applications such as strategy games.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation; I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction Techniques

1. Introduction

Crowd control research has become increasingly popular due to its potential applications in computer games and animation. In real-time strategy games such as *StarCraft 2* and *Age of Empires Online*, controlling military units to attack the opponents is a key criterion for success in the game. Players have to control the units using mouse gestures, which consist of multiple clicks and drags, to define the movement and formation of the units. Similarly, crowd simulation software like *Massive* requires the animators to carefully design the behaviour of the characters, such as programming their synthetic sensors and effectors, in order to control the formation of a crowd [Kan09]. Such kinds of control are time consuming and inefficient, thus degrading the user's experience.

Recent research eases the pain of crowd control by utilizing algorithms such as space-time optimization [KLLT08], spectral analysis [TYK*09] and hierarchical group control [GD11]. These algorithms require multiple steps for designing the crowd movements, including insertion of intermediate keyframes and specification of the trajectories of some characters, especially when obstacles and environments are involved. This hugely limits the usability of the algorithm particularly in real-time applications. Here, we see a dilemma: we wish the crowd to be realistic and contain fine details, but we want a simple control scheme that users can handle in real-time.

The major difficulty of crowd control lies in its high degree of freedom. Each character in the crowd is an entity and should be able to move independently under different circumstances. For example, when a crowd walks along a pathway that diverts into multiple smaller roads, the user needs to specify how the crowd should split into smaller groups and pass through each of the available routes. This requires a lot of user input and is generally achieved using a multi-pass algorithm approach. We observe that most of the movement in such a situation is affected by passive interactions between the crowd and the environment. In the above situation, the crowd will be split so that each person walks into the street that is closest to him/her, while avoiding crossing the path of the other agents. We believe that these kind of passive inter-

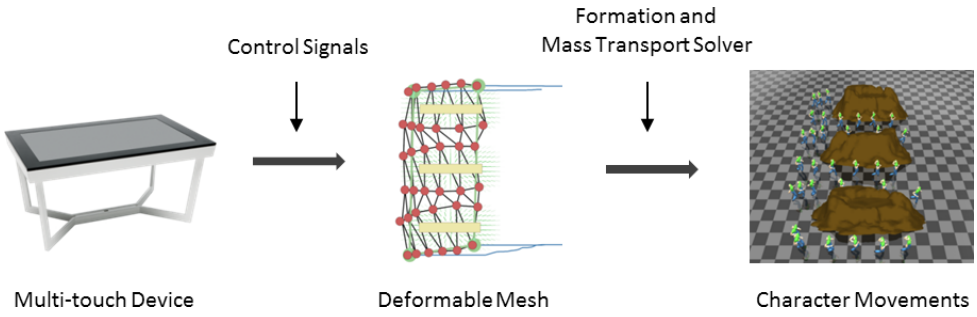


Figure 1: *The overview of the proposed system.*

actions can be computed automatically without a significant loss in simulation quality.

In this paper, we propose a new method to reduce the dimensionality of crowd control by making use of the passive dynamics between the individual agents, as well as those between the agents and the environment. In order to achieve this goal, we propose a new mesh-based crowd control scheme that makes use of a multi-touch device's simultaneous input capability. The subtle control signals from the fingers are used to deform the mesh and alter the way it interacts with obstacles and the environment. Further control signals from the mesh are used to determine character movement within the scene. Our method is a single-pass approach in which the user can manipulate the crowd and see the updates to its formation in real-time. Unlike previous mesh-based methods that rigidly constrain the characters to the mesh [KLLT08], we allow characters to switch target positions between frames by performing a Mass Transport solver similar to that widely used to compute the Earth Mover's Distance [RTG98]. This essentially minimizes the overall movement of all characters and reduces the chance of potential blocking among characters during formation transitions, hence producing more realistic animations.

Experimental results show that our system can produce realistic scenes of a crowd controlled through minimal, intuitive and high-level input signals from the user. We create scenes in which the crowd has to pass through complex environments such as a town with several diverging/converging routes, a street where multiple cars are driving, and constrained environments such as narrow pathways.

Our system is best applied to real-time crowd control applications such as strategic games. It can also be used efficiently to create scenes such as city-scale crowd flow for computer animations.

2. Related Works

Crowd simulation has largely been focused on synthesizing realistic pedestrian movements based on agent models [HFV00, YCP*08, OPOD10], fluid models [NGCL09], optimization [TCP06, SGC04] and data-driven models [LFCCO09, LCHL07]. In this research, we are more interested in crowd and formation control. The main focus is to control a group of characters according to user demand for animation synthesis and real-time applications such as computer games. We propose to control crowd formations using a multi-touch device, and discuss techniques which could possibly be applied for such a purpose here.

2.1. Representation for Crowd Formation Control

In crowd formation control, agents are directed in such a way that they move in a similar direction to the other agents in the crowd while maintaining an overall formation. This is essentially a high dimensional problem with a single objective as compared to ordinary agent-based crowd control systems, in which the control signal is distributed to the individual agents.

One well known solution to such a problem is using a deformable mesh to represent the crowd. Kwon et al. [KLLT08] applies Laplacian mesh editing [SLCO*04] to deform and concatenate existing crowd formations to synthesize larger scale animations. Takahashi et al. use spectral analysis to automatically interpolate two given formations [TYK*09]. Each formation is represented by a Delaunay triangulated mesh in these methods. Gu and Deng [GD11] propose a representation called formation coordinates, which is a local coordinate system that is similar to polar coordinates. For our multi-touch interface, we adopt the deformable mesh representation as this allows complex shapes to be manipulated by low dimensional control signals.

One problem with previous methods is that the characters are strongly bound to the target location in the formation: once their target locations are defined in the goal formation, the characters are required to reach their respective positions even if other characters might be blocking them at the formation border. In real situations, people in the border will simply shift toward the centre of the formation to produce space for the people arriving late. We found this problem can be solved by providing more degrees of freedom to the

agents when interpolating formations by employing a mass transport solver [RTG98].

2.2. Crowd Control Interface

In this research, we propose to use a multi-touch device to manipulate the formation of a crowd during gait in real-time. Here, we review how previous methods can be applied for such a purpose.

In most crowd control methods, strokes are used as control lines to specify the movements of the crowd. These are applied to the whole [GD11, Par10] or subgroups [OO09, KHKL09] of the crowd, or trajectories of some vertices that represent the crowd [KLLT08]. Such trajectories can be replaced by the trajectories of the user's fingers on the multi-touch device for real-time control.

One problem that arises when directly applying previous methods for real-time crowd control is the difficulty in specifying low level details when the crowd interacts with the environment. For example, it is difficult to move different groups of characters in a crowd through narrow corridors unless a multi-pass scheme is used, in which the user stops the animation and draws multiple strokes offline to specify the individual paths for different groups. A possible solution is to define a vector field and move each subgroup along its gradient [KSII09]. However, there can be cases that the flow is opposite to the direction that the characters are supposed to move. We prefer to use a more interactive process allowing the users to easily intervene and adjust the trajectories on-the-fly.

In this paper, we solve these problems by making use of the passive dynamics of the interactions between the characters, as well as those between the characters and the environment. Our mesh deforms automatically based on the influence of the environment, while keeping the overall formation.

2.3. Contributions

In this paper, we propose an efficient crowd control framework based on a deformable mesh. There are two major contributions:

1. We propose a new real-time scheme to manipulate crowd formations by combining the deformable mesh representation and a mass transport solver. The characters are not constrained to specific locations inside the deformable mesh, but cooperate together to fill it in based on the solution to the mass transport problem.
2. We propose a new scheme to increase the manipulability of a crowd's formation in constrained environments by making use of the passive dynamics of the interactions between the characters and the environment. This scheme allows the user to focus on the design of high level movements, while leaving the fine details to the system.

3. Method Overview

Our system is a three-layer system that consists of the user-input layer, the intermediate mesh representation layer, and the agent layer. Figure 1 shows the overview of the proposed system. The control signals consist of the high level user inputs from the multi-touch device that specifies the overall movement and formation of the crowd (see Section 4). The intermediate deformable mesh changes its shape according to the user input and its interaction with the environment (see Section 5). The mesh configuration is used to convert the high level signals into lower level control signals for the agents. Finally, the individual agents are guided to the area specified by the deformable mesh using the solution of the mass transport problem (see Section 6).

4. Movement and Formation Control

In this section, we explain how we create a mesh to represent a crowd, and control the crowd with the user control signals from the multi-touch device. We first describe the mesh representation and its deformation model. Then, we explain our deformation scheme based on the input from the multi-touch device.

4.1. Crowd Representation

We use a deformable mesh whose shape is computed by the as-rigid-as-possible deformation scheme [IMH05] to represent the formation of the agents. In our experiments, we use a rectangular shape composed of a uniform triangle strip, although this can be easily enhanced to arbitrary shapes by applying uniform sampling and Delaunay triangulation.

The user interacts with the mesh using a multi-touch device. When the user touches the mesh, the nearest vertex is selected as a control point. Let us define a control point as $c_i \in \mathbf{C}$, where i is the index of the control point, and $\mathbf{C}$ denotes the set of all control points. The user then drags the control points on the screen to define a continuous spatio-temporal trajectory that specifies where the control points must pass in the future frames. We represent each trajectory as a set of two-dimensional check points by dividing the trajectory into segments of a pre-defined length. For each frame, the target location of each control point $c_i(p_i)$ is defined based on the next check point in the corresponding user drawn trajectory p_i. We pass the current location of each control point $c_i(p_i)$ and the set of vertices of the current mesh $\mathbf{V_c}$ into the as-rigid-as-possible transformation solver (described in the next subsection) to generate the deformed mesh, which is called the user mesh, $\mathbf{V_u}$. $\mathbf{V_u}$ is subjected to the deformation based on the environment in a later stage.

4.2. Point, Line and Area Controls

Here, we propose different control schemes to overcome the limitation of multi-touch systems and produce a wider variety of control signals for manipulating the deformable mesh.

While a high resolution mesh provides flexibility for defining the crowd formation, the number of touch points a user can manipulate with a multi-touch device is limited. As a result, the user can only directly control a subset of vertexes on a mesh. Using the traditional point-based control system [IMH05], it is difficult for the user to control the rigidity when deforming the mesh. That is, when dragging a control point on a mesh, the system does not know how much the neighbour vertexes should follow such a control point. Igarashi et al. [IMH05] solve the problem by allowing the user to predefine the rigidity of the mesh manually however, this is not a plausible option for use in real-time control. We therefore present a set of controls, namely line and area controls, in addition to the point-based control scheme. These controls provide the user with the ability to manipulate the mesh with varying levels of rigidity.

The line-based control system constrains the vertices of the control mesh that are between two points specified by the user. When two control points c_0 and c_1 are defined, the vertexes between them are sampled as supplementary control points. In our as-rigid-as-possible solver, they are applied as soft constraints as they are allowed to be affected by the environment in a similar way to the rest of the uncontrolled vertices on the mesh (see section 5). When c_0 and c_1 are moved, the target location of the supplementary control points are computed by linearly interpolating the updated positions of c_0 and c_1. Defining multiple line constraints on the mesh allows the user to manipulate different sections of the mesh in different ways simultaneously.

We also propose an area-based control that provides rigidity to a two-dimensional portion of the mesh. The section on which this control is applied is determined by the convex hull of three or more user-defined control points. When the area control is created, supplementary control points are sampled along the edges and inside its convex hull. They act as soft constraints on the mesh in the same way as in the line control. The mean value coordinates [Flo03] of these supplementary control points are computed and are subsequently used to update their positions throughout the lifetime of the area control. In this way, the user can use a few control points to manipulate varying proportions of the mesh.

The effects of using the three different control schemes; point, line, and area, can be seen in Figure 2. Given the same control signal (Figure 2 Top Left) but different control schemes, the final formation is different (Figure 2 Top Right, Bottom Left and Bottom Right). The different forms of control confer varying levels of rigidity to the mesh, giving the user greater flexibility in the kinds of formations they can create with a small number of user inputs.

In order to identify the type of control that a user wants to apply, the timing that the fingers are placed on the multi-touch screen is examined. A single touch input gives basic point-based control, two simultaneous touch points indicates a line control, whilst three or more simultaneous touch points creates an area control. This scheme allows for different kinds of control to be applied simultaneously to different parts of the mesh. In Figure 3, left, we show an example where the user applies a line control at the left of a square and an area control on its right. The result when the user drags these areas is shown in Figure 3, right. It can be observed that the left half of the shape is deformed while the right part is kept rigid thanks to the two types of control used.

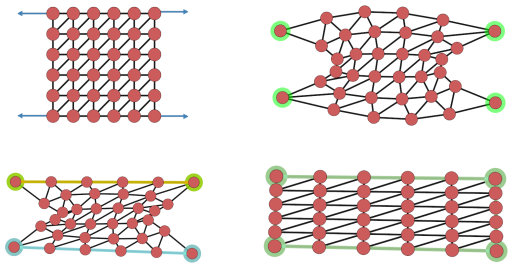


Figure 2: *Controlling (Top Left) a rectangular mesh with (Top Right) point-based control, (Bottom Left) line-based control, and (Bottom Right) area-based control on the four corners.*

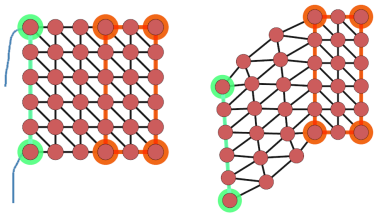


Figure 3: *(Left) The user applies a line control and an area control onto the same mesh. (Right) The resultant deformation.*

5. Environment-guided Mesh Deformation

In this section, we explain how we deform the control mesh of the crowd according to its interaction with the environment. This scheme is especially important for achieving effective obstacle avoidance in scenes where there are multiple obstacles, such as city scenes with several streets that are diverging and merging. The deformation of the control mesh is guided by a potential field generated by the environment. By letting these low-level interactions be controlled by our system, we allow the user to concentrate on the higher level control of the crowd.

The environment is modelled with a set of objects. Each object generates a potential field that will affect near-by vertices of the control mesh. Referring to Figure 4, the potential field at point $\mathbf{x}$ is computed based on the distance (d) from

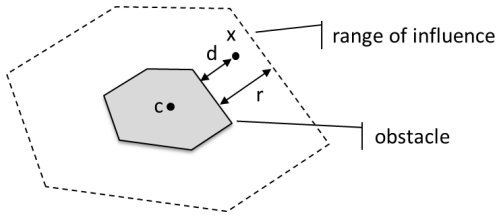


Figure 4: *The field produced by the obstacle is dependent on the distance of the point from the obstacle relative to the range parameter, as well as its relative direction from the obstacle centre.*

the object, the predefined range (r) and the direction vector from the centre of the object ($\mathbf{o}$). The amplitude of the potential field is computed based on the distance between the obstacle surface and the sample point:

$$f(\mathbf{x}) = \begin{cases} 1 - \frac{d}{r} \ (0 < d < r) \\ 0 \ (r \leq d). \end{cases} \quad (1)$$

We divide the floor into grid cells of equal size, and compute the potential field for each cell. The direction of the field is set to $\frac{\mathbf{x}-\mathbf{c}}{\|\mathbf{x}-\mathbf{c}\|}$ where $\mathbf{x}$ is the position at the centre of the cell and $\mathbf{c}$ is the centre of the obstacle. We use this approach because simply using the distance field can cause vertices to move slowly when there are long edges on the obstacle. The direction vector towards the obstacle centre increases the tangent element of the vector field in such cases.

Given a user mesh from the previous stage $\mathbf{V_u}$, we examine the position of each vertex of the control mesh, and sum the potential field produced by all the obstacles at that position. We also monitor the collisions between the vertices and the obstacles, and push them out to the nearest point on the surface if they penetrate through the obstacle. The edges of the control mesh are allowed to pass through the environment. The vertex positions of $\mathbf{V_u}$ are updated based on the field and the final mesh $\mathbf{V_f}$ is then computed. In order to prevent the control mesh getting stuck in the environment, we limit the obstacles shapes to convex hulls, as well as the minimum distance between two obstacles.

6. Character Mapping

Once the configuration $\mathbf{V_f}$ for the control mesh is decided we next have to determine which point on the mesh each agent will move to. To minimise obstructions between agents in the crowd during transition to the new formation it is necessary to assign an agent a target position based on their current configuration. For this we use a formulation for solving the transportation problem, which is used to compute the Earth Mover's Distance [RTG98]. The transportation problem is solved by minimising the amount of work to move objects from a set of source locations I to a set of target locations J. A set of point-to-point flows $f_{i,j}$ that minimises the overall energy can be computed as:

$$\sum_{i \in I} \sum_{j \in J} c_{i,j} f_{i,j}, \quad (2)$$

where $c_{i,j}$ is the cost of travelling from point $i \in I$ to point $j \in J$. This cost is assessed for the full connectivity of the two point-sets. Readers are referred to [RTG98] for further details. In this work, the source points correspond to the locations of the agents and the target points are the vertices of the mesh $\mathbf{V_f}$ at the current simulation step. A Euclidean distance metric is used for the cost of travel, and the correspondence between points is recalculated at every time step. Each point $i \in I$ and $j \in J$ can be weighted to allow user-defined partial/full matching between the two point-sets. These weights affect the flows that are provided as the solution to the transportation problem. For the purposes of mass transport, irregular weights produce solutions with one source point feeding to multiple goal points. In the current work it is desirable for each agent to be assigned to only one goal point and vice-versa. To achieve this we assign a weight of 1 to all source and target points to allow full mapping from current agent positions to candidate locations in the mesh.

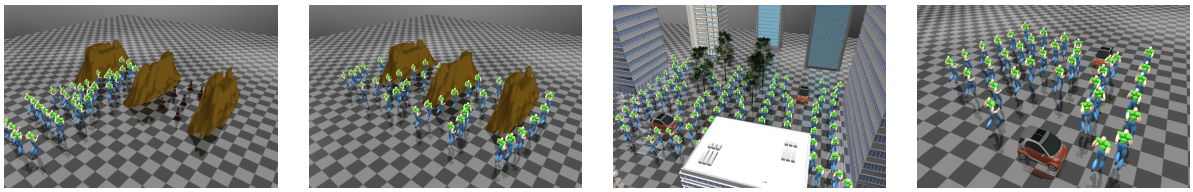
Once a point in the mesh is assigned to an agent their route to the location is computed using grid-based A$*$ search on a binary occupancy grid with similar resolution to the potential field grid. Each character considers other characters as obstacles and computes the optimal path. The characters then move to their corresponding target locations with a simple PD controller. A maximum speed is defined to avoid unnatural fast movement.

7. Experimental Results

We have produced scenes showing a group of characters passing through different static environments including a gateway, corridors, woodlands, a city area, and a dynamic environment where cars are moving around. The formation of the characters is manipulated in some of the examples such that they can pass through narrow pathways or produce visual effects. We also show an experiment that presents the advantage of using the mass transport solver for mapping each character to a vertex of the control mesh. The readers are referred to the supplementary video for further details. All the examples were produced starting from a uniform rectangular formation with 36 characters, except the city example, which consists of 100 characters.

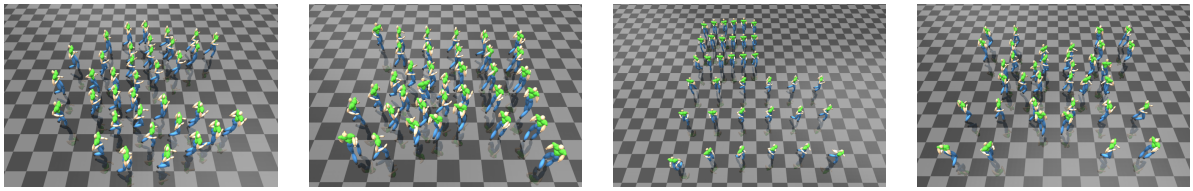
7.1. Environments with Static Obstacles

We first show an example in which the characters pass through an environment produced by three long, rectangular obstacles. Three different ways to pass the crowd through the environment are produced: (1) fully expanding them such



(a) (b) (c) (d)

Figure 5: *A crowd moving through (a) a corridor by squeezing into one row, (b) a corridor by spreading out, (c) a city area, and (d) a crowd avoiding cars.*



(a) (b) (c) (d)

Figure 6: *A crowd deformed into the shape of (a) "Pacman", (b) an arrow, (c) a letter "L" and (d) a star.*

that they pass between the obstacles as well as outside them, (2) unevenly split them to make two groups pass through the two corridors produced by the obstacles, and (3) squeezing them into a single corridor by stretching the formation parallel to the obstacles. Some of the snapshots are shown in Figure 5(a) and (b). Notice that the movement of individual characters is not defined explicitly by the user. Instead, based on the formation defined, the characters fit into the pathway automatically. We also produce another example where a larger scale of crowd passes through an urban area with many buildings, cars and trees (see Figure 5(c)). Even in such a complex condition, the characters have no difficulty moving through the area.

7.2. Environments with Dynamic Obstacles

In this example, the characters are controlled to pass through an environment in which there are multiple dynamic obstacles (cars). When the cars approach the characters, the characters automatically avoid them according to the user control and the potential field produced by the cars. (see Figure 5(d)). We can tune the strength of the potential field to adjust the distance at which a character starts to avoid the obstacles.

7.3. Formation Manipulation

In some experiments, we control the formation of the crowd to create different effects. We synthesized formations including shapes of "Pacman", an arrow, a letter "L" and a star (Figure 6). Line control is used to bend the rectangular formation into the mouth of the "Pacman" and the "L" shape. Area control is used to expand and shear the formation, as well as maintaining the rigidity of the arrow head while manipulating the tail. Point control is used to create the four points of the star formation.

7.4. Mass Transport Solver

In the last experiment, we show examples that clarify the advantage of using the mass transport solver for guiding the characters. The characters in a square formation are sup-

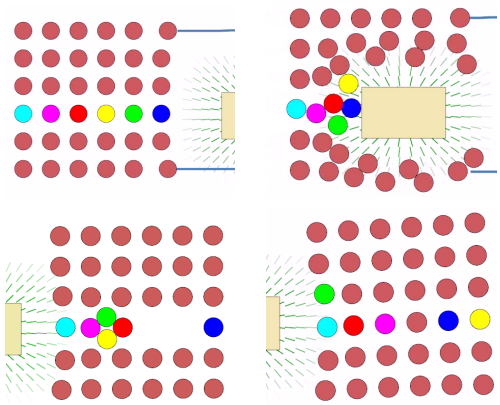


Figure 7: *Effect of using the mass transport solver: (Top Left) initial condition, (Top Right) characters caught in the middle, (Bottom Left) the final state when the locations for the characters are fixed in the formation, and (Bottom Right) final state when using the mass transport solver to compute the optimal final locations.*

posed to pass around an obstacle and merge again (Figure 7 Top Left). Because some of the characters are prevented from moving by the surrounding characters and the obstacle for a while (see Figure 7 Top Right), they are late to arrive to the group. In the case where the characters are required to return to their original position in the formation, they are blocked by the characters that filled in the row in advance (Figure 7 Bottom Left). This problem is particularly challenging in dense crowds where there is not enough space for the characters to pass through. In contrast, with our interpolation scheme based on the mass transport solver, the blocking character simply shifts into the formation to make room for the late arriver (see Figure 7 Bottom Right). Notice that the mapping of the characters to the mesh vertex in the final formation is different from the initial formation.

7.5. Computational Costs and 3D Rendering

The experiments are run on one core of a Core i7 2.67GHz CPU with 1GB of memory. For the multi-touch input we used an MTMini device [San10] along with the associated open-source Community Core Vision software [San08] for tracking touch points. The computation of the 2D trajectories that includes the deforming of the control mesh, reshaping it through its interaction with the environment, computing of the character destination by the mass transport solver, and updating their positions are all done in real-time at a rate of 60 frames per second.

The final 3D scene involves computing the movements of each character. We created a simple locomotion database with running motions. Based on the planned movement trajectory, the characters select the optimal motions with a precomputed search tree [LK05]. We allow minor adjustments in the original motion to better fit the movement trajectory, and apply inverse kinematics to fix the supporting foot on the floor. The motion planning process is in real-time, but the rendering process is done offline due to the large number of characters and the lack of rendering optimization such as level-of-detail.

8. Conclusion and Discussion

In this paper we present a novel method for effective user-guided control of crowd formation and motion in virtual environments. Currently, formation controls in computer games are rather basic. In most cases, a group of characters is moved from one location to another by simple mouse control. As the dimensionality of the user control is limited, the only solution is to let low level character-character or character-environment interactions be handled by the system. The current work utilises this idea to allow more refined control over a crowd's formation whilst still keeping the necessary control signals relatively simple. We have shown that the user can control the characters in various ways to move through the environment by subtly changing the way they control the formation via a multi-touch device. Our method provides a more enriching user experience in real-time applications such as games. The method is particularly well suited to applications where group cohesiveness is important e.g. real-time strategy games or social group motion in crowds.

Even though we have only presented actions such as running and avoiding in our examples, it is possible to add more complex interactions such as characters crawling under obstacles, side stepping along a narrow space, jumping over ditches and climbing up ladders. Such effects will be easier to produce by embedding the specific movements in the environment using patch-based approaches [YMPT09, LCL06]. The patch-based approach will also enable us to simulate scenes of character-character interactions, such as two armies fighting [SKSY08].

Using a Euclidean distance metric to solve the transportation problem for an agent's target location is not optimal, particularly in environments with large obstacles. Consider a situation where there is an obstacle between the crowd and their target formation as in Figure 8. The best solution would have the agents on the outside of the crowd move to locations in the middle of the target formation as they travel a shorter distance and reach these points earlier (Figure 8, pink arrow). With the Euclidean distance metric (Figure 8, blue arrow) the route the agent must take to reach the formation (Figure 8, green arrow) is not considered by the mass transport solver. The true shortest distance that takes into account the obstacles must be used for obtaining such a solution. We use the Euclidean distance to keep the computational cost low. We did not find this to be a problem since in the majority of cases the agents tracked the mesh closely and situations such as that in Figure 8 were very rare.

One improvement to the current approach would be providing a feedback signal to the movement of the control mesh. This signal would be based on how well the agents are tracking the mesh or even the state of the mesh vertices, that is, whether they are interacting with an obstacle

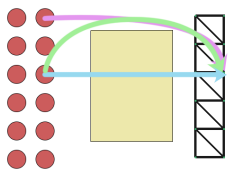


Figure 8: *Assigning formation goal points based on Euclidean distance (blue arrow) fails to consider the true length of the agent's path in the presence of obstacles (green arrow). It is more efficient to assign this goal point to an agent whose true distance to travel is smaller (pink arrow). This can be achieved using a cost metric based on geodesic distance or equivalent in the mass transport solver.*

or not. This would help when a vertex or agent takes particularly long to negotiate an obstacle or large deformation of the mesh occurs as a result of the user forcing the mesh to collide with large obstacles. The system could handle these situations by adjusting the movement of the mesh accordingly. Additionally, the current method does not account for the future motion of dynamic obstacles when planning the mesh movement. An RVO-like obstacle avoidance mechanism [vdBPS*08] would provide greater intelligence to the mesh motion, for example preventing it from passing in front of moving cars.

These enhancements to the current system provide interesting avenues for future work. It will also be necessary to prove the validity of this control scheme with a comprehensive user study.

Acknowledgement

We thank the anonymous reviewers for their constructive comments. This work is partially supported by grants from EPSRC (EP/H012338/1) and EU FP7 TOMSY.

References

[Flo03] FLOATER M. S.: Mean value coordinates. *Computer Aided Geometric Design 20* (2003). 4

[GD11] GU Q., DENG Z.: Formation sketching: an approach to stylize groups in crowd simulation. In *Proceedings of Graphics Interface 2011* (School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 2011), GI '11, Canadian Human-Computer Communications Society, pp. 1–8. 1, 2, 3

[HFV00] HELBING D., FARKAS I., VICSEK T.: Simulating dynamical features of escape panic. *Nature 407*, 6803 (September 2000), 487–490. 2

[IMH05] IGARASHI T., MOSCOVICH T., HUGHES J. F.: As-rigid-as-possible shape manipulation. *ACM Trans. Graph. 24*, 3 (2005), 1134–1141. 3, 4

[Kan09] KANYUK P.: Brain springs: fast physics for large crowds in wallċe. *IEEE Comput. Graph. Appl. 29*, 4 (July 2009), 19–25. 1

[KHKL09] KIM M., HYUN K. L., KIM J., LEE J.: Synchronized multi-character motion editing. *ACM Trans. Graph.* (2009). 3

[KLLT08] KWON T., LEE K. H., LEE J., TAKAHASHI S.: Group motion editing. In *SIGGRAPH '08: ACM SIGGRAPH 2008 papers* (New York, NY, USA, 2008), ACM, pp. 1–8. 1, 2, 3

[KSII09] KATO J., SAKAMOTO D., INAMI M., IGARASHI T.: Multi-touch interface for controlling multiple mobile robots. In *Proceedings of the 27th international conference extended abstracts on Human factors in computing systems* (New York, NY, USA, 2009), CHI EA '09, ACM, pp. 3443–3448. 3

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

[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), 898–906. 7

[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

[LK05] LAU M., KUFFNER J. J.: Behavior planning for character animation. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation* (New York, NY, USA, 2005), ACM, pp. 271–280. 7

[NGCL09] NARAIN R., GOLAS A., CURTIS S., LIN M. C.: Aggregate dynamics for dense crowd simulation. *ACM Trans. Graph. 28*, 5 (2009), 1–8. 2

[OO09] OSHITA M., OGIWARA Y.: Sketch-based interface for crowd animation. In *Proceedings of the 10th International Symposium on Smart Graphics* (Berlin, Heidelberg, 2009), SG '09, Springer-Verlag, pp. 253–262. 3

[OPOD10] ONDREJ J., PETTRÏ£¡ J., OLIVIER A.-H., DONIKIAN S.: A synthetic-vision-based steering approach for crowd simulation. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2010) 29*, 4 (2010). 2

[Par10] PARK M. J.: Guiding flows for controlling crowds. *Vis. Comput. 26*, 11 (Nov. 2010), 1383–1391. 3

[RTG98] RUBNER Y., TOMASI C., GUIBAS L. J.: A metric for distributions with applications to image databases. In *Proceedings of the Sixth International Conference on Computer Vision* (Washington, DC, USA, 1998), ICCV '98, IEEE Computer Society, pp. 59–. 2, 3, 5

[San08] SANDLER S.: Community core vision software. http://ccv.nuigroup.com/, 2008. 7

[San10] SANDLER S.: Multitouch mini. http://sethsandler.com/multitouch/mtmini/, 2010. 7

[SGC04] SUNG M., GLEICHER M., CHENNEY S.: Scalable behaviors for crowd simulation, September 2004. 2

[SKSY08] SHUM H. P. H., KOMURA T., SHIRAISHI M., YAMAZAKI S.: Interaction patches for multi-character animation. *ACM Trans. Graph. 27*, 5 (2008), 1–8. 7

[SLCO*04] SORKINE O., LIPMAN Y., COHEN-OR D., ALEXA M., RÖSSL C., SEIDEL H.-P.: Laplacian surface editing. In *Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Geometry Processing* (2004), pp. 179–188. 2

[TCP06] TREUILLE A., COOPER S., POPOVIĆ Z.: Continuum crowds. *ACM Trans. Graph. 25*, 3 (2006), 1160–1168. 2

[TYK*09] TAKAHASHI S., YOSHIDA K., KWON T., LEE K. H., LEE J., SHIN S. Y.: Spectral-based group formation control. *Comput. Graph. Forum 28*, 2 (2009), 639–648. 1, 2

[vdBPS*08] VAN DEN BERG J., PATIL S., SEWALL J., MANOCHA D., LIN M.: Interactive navigation of individual agents in crowded environments. In *I3D '08: Proceedings of the 2008 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games* (2008), ACM. 8

[YCP*08] YEH H., CURTIS S., PATIL S., VAN DEN BERG J., MANOCHA D., LIN M.: Composite agents. In *SCA '08: Proceedings of the 2008 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2008), ACM. 2

[YMPT09] YERSIN B., MAÏM J., PETTRÉ J., THALMANN D.: Crowd patches: populating large-scale virtual environments for real-time applications. In *I3D '09: Proceedings of the 2009 symposium on Interactive 3D graphics and games* (New York, NY, USA, 2009), ACM, pp. 207–214. 7