Interactive Formation Control in Complex Environments

Joseph Henry, Hubert P. H. Shum and Taku Komura

Abstract—The degrees of freedom of a crowd is much higher than that provided by a standard user input device. Typically, crowd control systems require multiple passes to design crowd movements by specifying waypoints, and then defining character trajectories and crowd formation. Such multi-pass control would spoil the responsiveness and excitement of real-time control systems. In this paper, we propose a single-pass algorithm to control a crowd in complex environments. We observe that low level details in crowd movement are related to interactions between characters and the environment, such as diverging/merging at cross points, or climbing over obstacles. Therefore, we simplify the problem by representing the crowd with a deformable mesh, and allow the user, via multi-touch input, to specify high level movements and formations that are important for context delivery. To help prevent congestion, our system dynamically reassigns characters in the formation by employing a mass transport solver to minimise their overall movement. The solver uses a cost function to evaluate the impact from the environment, including obstacles and areas affecting movement speed. Experimental results show realistic crowd movement created with minimal high-level user inputs. Our algorithm is particularly useful for real-time applications including strategy games and interactive animation creation.

Index Terms—Three-Dimensional Graphics and Realism, Animation, Input devices and strategies, Gaming

1 Introduction

CROWD control research has become increasingly popular due to its potential applications in computer games and animations. 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. 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 [1]. Such kinds of multi-pass control are time consuming and inefficient, thus degrading the user's experience.

Recent research eases the pain of formation control in crowds by utilizing algorithms such as space-time optimization [2], spectral analysis [3] and hierarchical group control [4]. 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 with fine details, but we also want a user to be able to specify these details interactively with a simple control scheme.

- *J. Henry is with the Institute of Perception, Action and Behaviour, University of Edinburgh, UK. E-mail: j.henry@ed.ac.uk*
- *H.P.H. Shum is with the Faculty of Engineering and Environment, Northumbria University, UK. Email: hubert.shum@northumbria.ac.uk*
- *T. Komura is with the Institute of Perception, Action and Behaviour, University of Edinburgh, UK. Email: tkomura@inf.ed.ac.uk*

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 approach [2]. However, we observe that most of the detail of the character movements is affected by the interactions between the crowd and the environment. In the above situation, the crowd is split so that each character walks into the street that is the closest and least congested, while avoiding crossing the path of the other agents. In other situations, a character may actively climb over an obstacle, rather than pass through a congested pathway, as it is a better option to reach their goal. We believe that these kind of interactions can be computed automatically without a significant loss in simulation quality.

In this paper, we propose a new method to reduce the dimensionality of the crowd control problem by a mesh-based control scheme, and use a multi-touch device to capture multiple control points simultaneously. The subtle control signals from the fingers are used to deform the mesh and alter the way it interacts with obstacles and the environment. Being a single-pass approach, the user can design the high level movement and formation of a crowd intuitively using our system. The low level details, such as the optimal trajectories of individual characters to achieve the formation, as well as the various interactions between the characters and the environment, are automatically controlled by the system in real-time.

The quality of the resultant animation depends heavily on the intelligence of the characters for achieving the movement and formation criteria. Unlike previous mesh-based methods that rigidly constrain the characters to the mesh [2], our characters possess the intelligence to decide their optimal target positions to form the desired formation using a mass transport solver [5]. This essentially minimizes the overall movement of all characters and reduces the chance of potential blocking among them. This is particularly useful for interactive applications, where users may wish to change the motion of the crowd suddenly, in which case, flexibility in the characters' motion will help prevent degradation in simulation quality. We integrate the metric proposed in [6] into our mass transport solver such that the characters evaluate the optimal paths by minimizing a cost function. This allows them to navigate realistically in complex terrains involving obstacles and areas that penalize movement speed.

Experimental results show that our system can produce realistic scenes of a crowd controlled through minimal high-level input signals from the user. We create scenes in which the crowd has to pass through complex environments such as a street with multiple moving cars, constrained environments such as narrow pathways, and areas with several route choices that contain different obstacles. The characters interact with different environment objects in particular ways, such as climbing over walls and ducking while walking under trees.

Compared to the approach in our previous work in [7], we have enhanced our system such that the complexity of the environment is considered when assigning final positions in the crowd formation. In particular, our system now takes into account areas of the environment that require characters to conduct special actions such as crawling, jumping, climbing and swimming. This is achieved by applying a modified version of the distance metric based on the Eikonal function [6], instead of using a simple Euclidean distance metric when solving for the characters' goal positions. By including information on the impact of the environment on an agent's path the system can produce realistic and efficient formation control even under very crowded and complex environments, which could have easily resulted in congestion in our previous system. We also present an in-depth user study to analyze our user interface compared to traditional mouse-based controllers.

Our system is best applied to real-time crowd control applications involving formation changes and environment interactions such as strategic games. It can also be used for interactive animation creation to generate scenes such as city-scale crowd flow.

2 Related Work

Crowd simulation has largely been focused on synthesizing realistic pedestrian movements based on agent models [8], [9], [10], fluid models [11], optimization [6], [12] and data-driven models [13], [14]. 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 commands, 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. [2] applies Laplacian mesh editing [15] to deform and concatenate existing crowd formations to synthesize larger scale animations. Takahashi et al. use spectral analysis to automatically interpolate two given formations [3]. Each formation is represented by a Delaunay triangulated mesh in these methods. Gu and Deng [4] propose a representation called formation coordinates, which is a local coordinate system that is similar to polar coordinates. In [16] behaviours of pedestrian crowds are used to determine the formation of groups in response to the local environment. These behaviours however are not applicable to controlling large groups involving more than a few characters and only define a subset of shapes that a group can form. 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. Although Gu and Deng provide an extension to their work to include specification of formation interpolation, [17], this still requires additional stages of input from the user to achieve nice transitions between formations. 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 [5]. This results in the characters tracking the desired formation effectively whilst not requiring additional user input to do so.

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 [4], [18] or subgroups [19], [20] of the crowd, or trajectories of some vertices that represent the crowd [2]. 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 [21]. However, there can be cases that the flow is opposite to the direction that the characters are supposed to move. [22] define an interface for manipulating continuum-based fields to allow a user to refine the flow of the crowd. This approach does not allow direct control over the characters in the simulation, instead relying on the user to specify the general interaction of a crowd with the environment. This generally requires the user to author overall crowd flow by augmenting an existing simulation using multiple input strokes. We prefer to use a more interactive process allowing the users to easily intervene and adjust the trajectories of the characters 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 Incorporating Motion Data

Previous crowd simulation largely focuses on the use of simple running and avoiding motion to guide a set of characters through a virtual environment. Gu and Deng looked at enhancing the diversity of agent motion in crowds, [23]. However, motions that involve a character interacting with the environment around them were not considered. Since many interesting scenarios involve a richer set of actions involving direct character-environment interactions, we wish to consider the effect of such motions on the character's path planning. We suggest the use of patch-based approaches in order to achieve this. Previous work on embedding motion data in virtual environments involves either placement of specific patches in a regular tiled grid, [24], [25], or stitching of irregular shaped patches in a highly constrained fashion, [26], [27]. While these methods are effective for producing large scale crowd scenes the relatively inflexible nature of the patches makes them difficult to apply directly to interactive control of a crowd. We wish for the characters to be able to enter and exit a given patch based largely on the directions given by the user. We therefore propose a simplified version of motion data patches that permits flexibility in the simulation whilst still conveying relevant information on character-environment interactions.

2.4 Contributions

There are two major contributions in this paper:

1) We propose a new single-pass scheme to manipulate a crowd in a constrained environment. We model the crowd as a deformable mesh, and allow the user to give high level instructions on the crowd movement and formations. The system then controls the detailed movements of individual characters, considering the passive and active dynamics of interactions with respect to the environment.
2) We propose a new method to compute the optimal trajectories of the characters in the crowd. The characters are not constrained to specific locations in the deformable mesh, but cooperate with each other to fill the mesh using a mass transport solver. We integrate an augmented version of the cost function proposed in [6] into the solver, such that the system considers the impact from the environment to individual characters and computes their optimal trajectories to achieve the required formation.

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

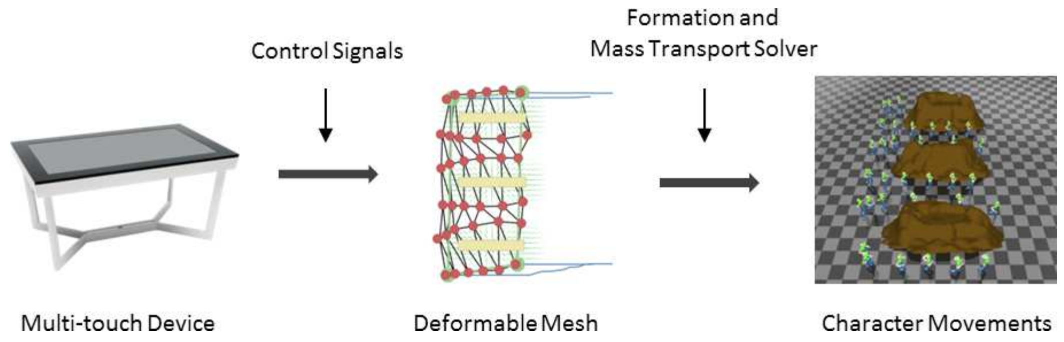


Fig. 1. The overview of the proposed system.

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 [28], 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 continuous spatio-temporal trajectories that specify 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.

We found in some cases a vertex or agent can take 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. In these cases, the agents do not track the formation as well as desired. To handle this we apply a small opposing force to the movement of the user's constraints so as to slow the movement of the mesh and allow the agents to catch up. This force is calculated proportional to the average distance of each agent from their respective vertex on the control mesh. To prevent the constraints from ever moving backwards, the magnitude of the opposing force is limited to that of the constraint velocity.

Although the agent's goal points have been discussed in terms of the vertices of $\mathbf{V_f}$ they are not limited to these points on the control shape, particularly when there is a discrepancy between the number of agents and the number of vertices. In this situation, the goal positions of the agents can be determined using barycentric coordinates across the triangles of the mesh. With a triangle ID and appropriate barycentric coordinates an arbitrary number of goal points can be produced at arbitrary positions across the mesh. Furthermore, the number of goal points can be easily altered with the number of agents, without affecting the mesh shape by adding or removing vertices.

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 vertices on a mesh. Using the traditional point-based control system [28], 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 neighbouring vertices should follow such a control point. Igarashi et al. [28] 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 vertices 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 [29] 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 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.

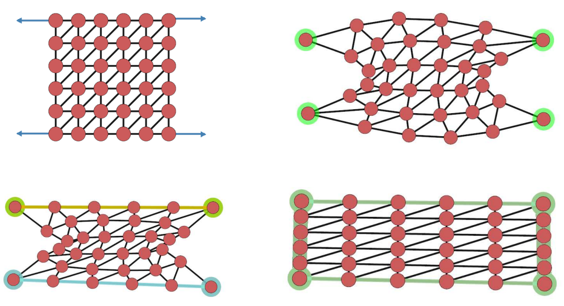


Fig. 2. Controlling (Top Left) a rectangular mesh with (Top Right) point-based control, (Bottom Left) line-based control, and (Bottom Right) area-based control on the four corners.

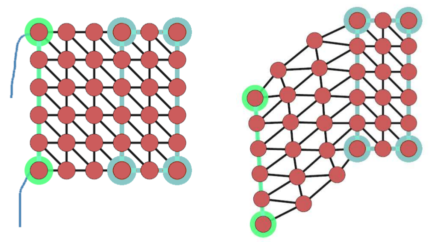


Fig. 3. (Left) The user applies a line control and an area control onto the same mesh. (Right) The resultant deformation.

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 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 **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.

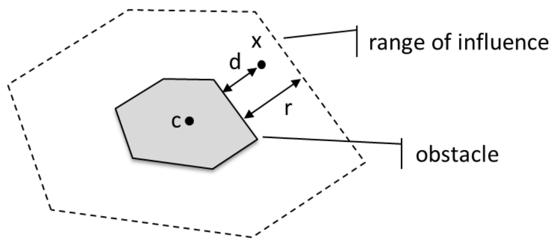


Fig. 4. The field produced by the obstacle is dependent on the distance of the point from the obstacle relative to the range parameter, as well as its relative direction from the obstacle centre.

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, and the minimum distance between two obstacles to at least that of an agent's diameter. Concave obstacles can still be modelled with a few smaller convex obstacles. In this case, the potential fields produced by each of the convex obstacles do not prevent a user from directing a crowd into the respective concave area. For complex environments, such as several concave obstacles packed tightly together, the agents or the mesh vertices may get stuck and become unable to follow a user's commands. Therefore, it is preferable in such an interactive application to limit the complexity of the environment. this is because we wish for the user to decide the overall direction of the mesh movement as much as possible, and for all agents to be able to track the user's instructions well.

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. In this section we first describe how we determine the goal position of each agent by employing a solution to the mass transportation problem (Section 6.1). We then discuss how we use a potential field construction to incorporate an environment-aware metric in our mass transport solver (Section 6.2). This metric not only accounts for obstacles and other agents in the environment (Section 6.2.1) but also considers the motion data that will be used in the final render of the scene (Section 6.2.2-6.2.3). By doing this we are able to encode motion data information implicitly in an agent's planning. This results in better goal assignment and subsequently more efficient movement of agents to satisfy a user's input formation.

6.1 Assigning Agent's Goal Positions

In order to assign each agent's goal position in the user-defined formation we use a formulation for solving the transportation problem, which is used to compute the Earth Mover's Distance [5]. 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. The solution to the problem consists of finding the amounts of the objects to be transported across all routes, henceforth referred to as "flows' ' ($f_{i,j}$), that minimises the overall cost of transportation between the two point sets. A set of flows can be evaluated using:

$$\sum_{i \in I} \sum_{j \in J} c_{i,j} f_{i,j}, \tag{2}$$

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. The optimal set of flows therefore minimises Equation (2). Readers are referred to [5] 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. 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 act as the supply and demand signals from the source and goal points respectively. The flows that minimise Equation (2) satisfy these signals. For example, for the purposes of mass transport, high supply weight from a point $i \in I$ and low demand weights on several points in J can 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. This provides a set of point-to-point correspondences between the agents and the target formation, which is recalculated at every time step.

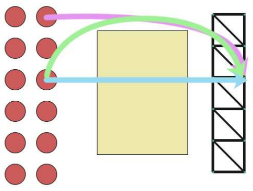


Fig. 5. Assigning formation goal points based on Euclidean distance (blue arrow) fails to consider the true length of the agent's path in the presence of obstacles (green arrow). It is more efficient to assign this goal point to an agent whose true distance to travel is smaller (pink arrow). This can be achieved using a cost metric based on geodesic distance or equivalent in the mass transport solver.

6.2 Environment-Aware Metric for Goal Assignment

Appropriate assignment of agents to the vertices of the formation mesh, $\mathbf{V_f}$, is achieved by employing a suitable cost metric, $c_{i,j}$, in the mass transport solver. In previous work, [7], a Euclidean distance metric was used to solve the transportation problem for an agent's target location. However, this 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 5. 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 5, pink arrow). By using the Euclidean distance metric (Figure 5, blue arrow) the cost provided to the mass transport solver does not reflect the route the agent must take to reach the formation (Figure 5, green arrow). The true shortest distance that takes into account the obstacles must be used to obtain the best assignment of agent goal positions in the formation. In the next section we describe how we incorporate this information into the mass transport solver.

6.2.1 Evaluating Cost to the Goal

In continuum-based crowd simulation [6] the cost for an agent to travel to its goal is given by an approximation of the Eikonal equation. This approximation uses a cost metric that accounts for the environment as well as other agents. We construct a potential field to determine the cost to travel to the vertices of the formation mesh, $\mathbf{V_f}$, for a given point in the environment. In [6] the overall cost for an agent to travel to its destination is provided by a combination of the length of the path to the goal, the time taken, and a discomfort field based on obstacles and other agents in the environment:

$$\underbrace{\alpha\int_P 1ds}_{\text{Path Length}}+\underbrace{\beta\int_P 1dt}_{\text{Time}}+\underbrace{\gamma\int_P gdt}_{\text{Discomfort}}, \tag{3}$$

where α, β, and γ are weights; g is the value of discomfort at a given point in the environment; and dt and ds mean the integral is taken with respect to time or path length respectively. Readers are referred to [6] for more details on how the individual values for path length and discomfort in Equation (3) are calculated. Using the equality $ds = fdt$ where f is speed, Equation (3) can be rewritten and simplified to

$$\int_P Cds,\ \text{where } C \equiv \frac{\alpha f+\beta+\gamma g}{f}. \tag{4}$$

By applying Equation (4) to a 2-Dimensional grid of the environment we can compute a *unit cost field* for a given scene. To produce the final potential field, $\mathbf{\Phi}$, we employ the same approach as [6] of using the fast marching method [30] to approximate the *Eikonal equation*:

$$\|\nabla\phi\left(x\right)\| = C, \tag{5}$$

where $\phi\left(x\right)$ is the value of the field at a given point x in the environment. A potential field $\mathbf{\Phi_i}$ is constructed for each vertex v_i in the user-defined control mesh $\mathbf{V_f}$. In each case, $\mathbf{\Phi_i} = 0$ in the cell containing v_i and everywhere else $\mathbf{\Phi_i}$ satisfies Equation (5). An example potential field for a single vertex in $\mathbf{V_f}$ can be seen in Figure 6. Given an agent's position in the environment, A_{pos}, we can retrieve the cost (C_{A_{pos},v_i}) for the agent to travel to each vertex, v_i, in $\mathbf{V_f}$ by taking the value at that location in the appropriate potential field:

$$C_{A_{pos},v_i} = \mathbf{\Phi_i}\left(A_{pos}\right). \tag{6}$$

Due to the discrete nature of the 2-dimensional grid we use bilinear interpolation to achieve a more accurate reading from $\mathbf{\Phi_i}$. These values can be passed into our mass transport solver in order to assign agents an appropriate goal position in the final formation. Once a point in the mesh is assigned to an agent their route to the location is computed using gradient descent on the field. The characters then move to their corresponding target locations with a simple PD controller. A maximum speed is defined to avoid unnatural fast movement.

6.2.2 Using Environment Information for an Improved Cost Metric

In the majority of previous crowd simulation research, environments consist solely of traversable "free" space or impassable obstacles. However, in a number of real-life environments there are certain objects that, while traversable, will affect a person's speed of travel across them. Examples of these include low areas through which a person must crawl or

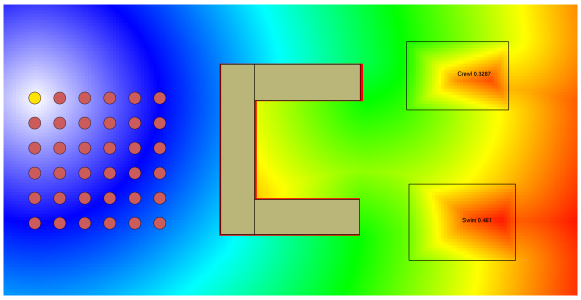


Fig. 6. An example of a potential field produced for a single vertex in the user-defined control mesh in an environment containing obstacles and motion data patches. Red indicates a high cost to travel and white indicates a low cost. The field shown is with respect to the formation position of the bright yellow agent.

fences/walls over which people must jump or climb. In this section, we describe how we apply such an idea to agent planning to produce appropriate motion in various environments.

The fast marching method [30] is an approach that can be used to track a moving boundary expanding outwards from a source point. The concept of a *speed field* is used to define the rate at which the front of the moving boundary propagates. In this way, low values in the speed field can be used to represent obstacles, whilst high values can signify open areas in a scene. We can therefore think of this value as representing the speed at which an agent can travel through a given point, x, in the environment. For impassable areas the travel speed can be considered to be zero (creating infinitely high cost values) whilst for open areas, in which an agent can move freely, the travel speed can be considered to have a value of one, allowing an agent to move at their desired speed. Areas that do not permit an agent to move freely but are still traversable can be assigned an intermediate value for travel speed. It is also possible that certain areas of an environment, such as moving walkways, allow an agent to travel faster than their normal running speed. We incorporate this idea into the current framework by rewriting C in Equation (4) to be:

$$C \equiv \frac{\alpha f + \beta + \gamma g}{f \times TravelSpeed\,(x)}, \tag{7}$$

where $TravelSpeed\,(x)$ represents the speed at which an agent can travel through a given point, x, in the environment.

6.2.3 Embedding Motion Data in the Environment

By considering environment travel speed in the cost function, we can easily incorporate a variety of different motions into agent planning by extracting velocity information from motion data. Our method only relies on information from the 2D trajectory of a given motion, so it works equally well with both video tracking or full-body motion capture data. Given a trajectory we take the average velocity and use this as the motion's travel speed value. These values are then adjusted according to their relative speed with respect to the running motion data. This gives a value of one to the motion used for the character moving in open space and a value relative to this motion for all other data. Table 1 shows the values extracted from the motion data used in our experiments. Note that, with this method, at no point is a value of zero assigned to a piece of motion as this is used to represent impassable obstacles.

Motion Type	Average Travel Speed
Run	1.0
Crawl	0.33
Duck	0.30
Swim	0.46
Jump	1.24
Climb Up	0.50
Climb Down	0.48

TABLE 1
The average speeds extracted from motion data used in the experiments

This information can be applied to the scene by creating "patches' '. These patches are represented as 2D polygons in the scene and contain the motion data file name, ID, and average speed of the motion. Entrance to a patch is detected when the agent's current motion ID switches from the ID for the running data, used in the open environment, to the ID given by the patch. When an agent enters a patch, the motion data ID determines the motion for the agent to carry out and the average travel speed determines the speed of the agent as it passes through the patch. The average travel speed also determines the speed field value in any grid cells that contain the patch, so that it can be used in Equation (7) during the planning phase. Examples of these patches, and how they affect the overall cost field, can be seen as the two transparent blocks on the right side of Figure 6. For cyclic motions such as crawling, ducking, and swimming, patches can be of arbitrary shape and size however, for single-step motions such as jumping and climbing, the length of the patch is constrained by the length of the motion data's trajectory. This prevents these actions from inappropriately executing multiple times in the final render. Additionally, for the crawling, ducking, and swimming motions, an appropriate transition motion is applied when it is detected that the agent enters or leaves the patch.

7 Experimental Results

We have produced scenes showing a group of characters passing through different static environments including corridors, woodlands, an obstacle course and a dynamic environment where cars are moving around. Each scene contains open space and obstacles as well as various objects that the characters can traverse in order to reach their goal. The formation of the characters is manipulated in some of the examples such that they can pass through narrow pathways, interact directly with certain parts of the environment, 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 obstacle course example, which consists of 100 characters.

7.1 Handling Motion Data Patches

Here we show how characters in the crowd are affected by motion data patches in different environments. Experiments show how the crowd can handle an arbitrary number of static or dynamic patches. Our first example involves a crowd moving through dense woodland, incorporating several trees modelled by small obstacles. The branches of the trees overlap with one another and as a result, characters must duck to pass through (see Figure 7(a)). This motion is represented in the scene by applying appropriately sized motion data patches around the base of each tree. These patches contain the average travel speed from the ducking motion data used in the final render of the scene. The characters are able to pass through the small gaps between each tree while exhibiting a slower travel speed that resembles the required ducking motion.

In the next example, the characters are controlled to pass through an environment in which there are multiple dynamic obstacles (cars) and dynamic patches (mice). 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 7(b)). We can tune the strength of the potential field to adjust the distance at which a character starts to avoid the obstacles. Furthermore, in the accompanying video the characters can be seen to react to the mice with a jump motion when they pass through the crowd. This reaction is produced by the movement of the dynamic patches and the effect on the character's speed when they interact.

We also produce another example where a larger crowd passes through an obstacle course with narrow corridors, walls to climb/jump over, netting to crawl under, and a pool to swim through (see Figure 7(c)). This scene contains a variety of different motion data patches that allow the characters to interact appropriately with the environment. Even in such a complex condition, the characters have no difficulty moving through the area.

7.2 Defining Crowd Trajectories

In these examples we show how a user is able to control the overall motion of the crowd while the low-level interactions of the crowd are handled by the system. We first show an example in which characters pass through an environment containing three separate pathways. Each pathway contains a different kind of terrain, from top-to-bottom: a net to crawl under, open space to run through, and a pool to swim through. A snapshot can be seen in Figure 8(a). The accompanying video shows the user specifying the subsets of the crowd that pass through each individual pathway using a basic multi-touch gesture. 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 pathways automatically. In each pathway the characters move appropriately according to the type of terrain they are passing through.

In addition to this, we show a similar example in which characters have multiple paths through which they can travel (see Figure 8(b)). This time however the pathways are narrower than before. The central pathway contains open space and the two pathways either side have objects that the characters must climb over. In the accompanying video the user performs a gesture to move the entire crowd through the central pathway. When the central pathway becomes too congested the characters choose the slower side pathways rather than waiting for the central pathway to clear. The side pathways have become a better choice for following the formation specified by the user. This example shows the ability for the system to adapt the character's trajectories to best follow the user's commands.

7.3 Formation Manipulation

Here we show examples of the different levels of formation manipulation in our system that can be used to produce different visual effects. We first show the system's ability to interpolate between different high-level crowd formations quickly and accurately. We then show how a user can control various aspects of these formations to carry out certain tasks.

A set of mesh shapes is registered to define formations that the crowd can switch between. Some examples of these formations can be seen in Figure 9. The user is able to specify the trajectory of the crowd using the multi-touch device and the crowd's formation can be switched depending on the environment. In the current demo, the formation is switched when the crowd pass over pre-defined checkpoints embedded in the environment however, it would be reasonable to allow the user to handle this switching either through a basic button interface or a set of multi-touch gestures. Figure 9(a) shows the crowd in an arrow formation which is effective for passing

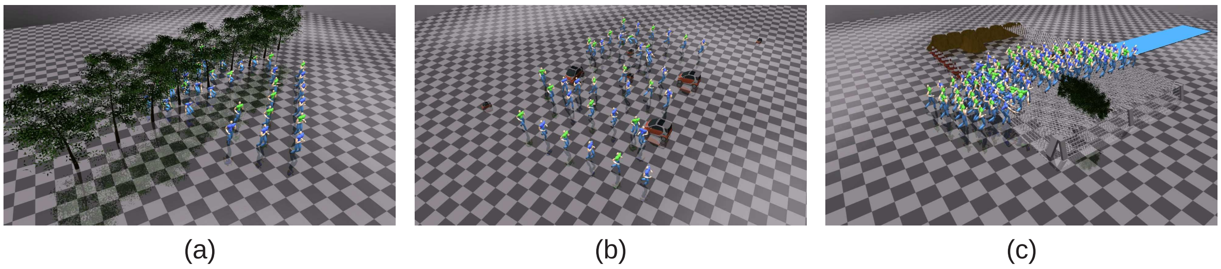


Fig. 7. A crowd (a) moving through dense woodland, (b) avoiding large and small moving cars, and (c) passing through a complex obstacle course area. The crowd is able to interact with many arbitrarily placed objects in the environment by responding to motion data patches in the scene.

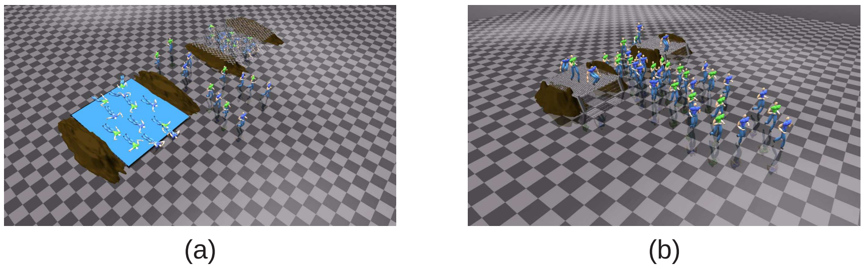


Fig. 8. A crowd (a) moving through several pathways containing various terrain, and (b) choosing slower pathways in response to congestion. The crowd plans an optimal trajectory to follow the instructions of the user whilst also considering the environment.

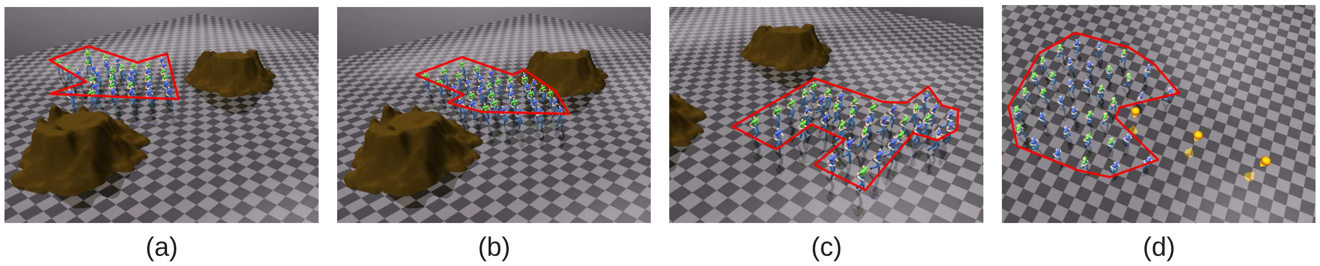


Fig. 9. A crowd adapting it's formation based on user signals and the environment: (a) An arrow formation is used to pass easily through the corridor, (b) The crowd formation is affected by the surrounding environment, (c) The crowd can easily transition to a variety of formations. (d) The current crowd formation can be easily manipulated to achieve certain tasks.

through a narrow corridor. This formation can be directly switched to from the original square formation. The sides of the arrow deform temporarily when the formation is inside the corridor to ensure that the characters pass through without trouble (Figure 9(b)). Once through the corridor the crowd can switch back to their original formation or to another formation entirely, depending on the user's requirements (Figure 9(c)). By using the mass transport solver to assign character's goal positions, transitions between different formations occur quickly and with minimal congestion.

Not only does our system provide easy, high-level transition between different crowd formations but we also allow the user to manipulate individual formations directly. We show an example where the crowd formation resembles a ″pacman″ character. A set of simple gestures can provide interactive control over multiple aspects of the formation. In this case the user is able to translate the formation whilst simultaneously manipulating the ″pacman″ character's mouth. This helps to create visual effects as well as perform certain tasks, such as collecting items in the environment.

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 supposed to pass around an obstacle and merge again (Figure 10 Top Left). Because some of the characters are prevented from moving by the surrounding characters and the obstacle for a while

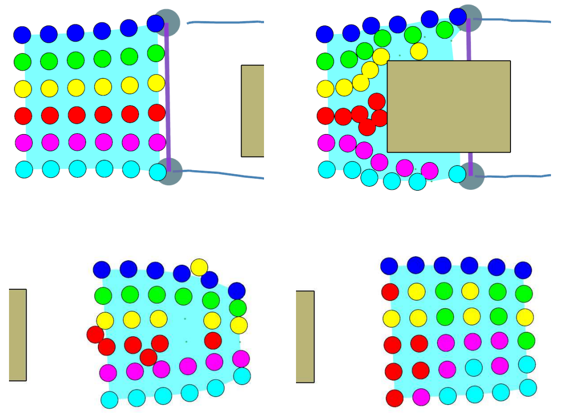


Fig. 10. Effect of using the mass transport solver: (Top Left) initial condition, (Top Right) characters caught in the middle, (Bottom Left) the final state when the locations for the characters are fixed in the formation, and (Bottom Right) final state when using the mass transport solver to compute the optimal final locations.

(see Figure 10 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 10 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 10 Bottom Right). Notice that the mapping of the characters to the mesh vertices in the final formation is different from the initial formation.

7.5 User Study

To evaluate the effectiveness of the current system for interactively moving and defining crowd formations we carried out a user study. In the study we had a total of 15 participants, consisting largely of postgraduate students all aged between 20 and 35. To compare our system to other user-control approaches we implemented a mouse controller based on those found in current real-time strategy games. This controller included a basic mouse control interface (Figure 12) and the movement of the characters was determined using the approach in [6]. Participants were given some practice time to get comfortable with using the mouse scheme and our proposed multitouch control scheme. In general, participants spent 1-2 minutes practicing with each control scheme. Once happy with each scheme, participants were asked to carry out a number of tasks to test them. In each task users were presented with one of 4 different environments (Figure 11). Each of the environments contained a set of obstacles as well as a number of collectible items, and users were instructed to guide a set of characters to collect the items in the environment in as little time as possible. The position of the collectible items in the scene was randomised at the start of each task to prevent any experimenter bias from their placement. Each environment was presented twice to the user: once for each control scheme, and the order in which users tested the control schemes was switched to prevent any bias from task experience.

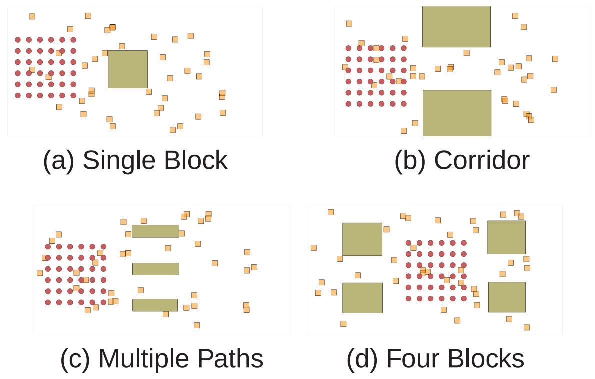


Fig. 11. The initial setup for environments used in the user study. The positions of the items to collect (small orange blocks) were randomised for each trial.

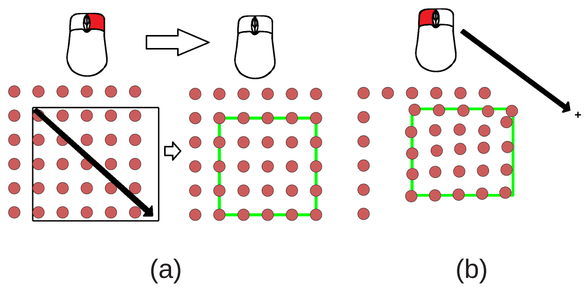


Fig. 12. The mouse control interface used in the user study. (a) The user could select the characters by right-clicking and dragging across them in the scene. (b) The characters could then be given a goal point by left-clicking in any open space.

In most scenarios, the multitouch controller enables more efficient collection of items by allowing a user to move the crowd and manipulate its shape simultaneously. A comparison of the times taken to complete each task with the different controllers is shown in Figure 13. It can be seen that the time for task completion is reduced in three of the four scenes when using our multitouch controller (Figure 11(a)-(c)). In fact, for the ”Single Block”, ”Corridor”, and ”Multiple Paths” environments the multitouch controller shows a 35%, 16%, and 20% decrease in the median completion time respectively, compared to the mouse controller. The greater amount of open space in the ”Single Block” environment allowed users to take advantage of the multitouch controller's simultaneous movement and shape control capability. In a number of cases the users were able to expand and contract the group formation whilst guiding the agents around the scene resulting in a large improvement in task completion time. In the ”Multiple Paths” scene, participants utilised the mesh interaction with the environment to divide the crowd into several pathways at once. This allowed the user to cover much of the environment with minimal gestures and complete the task more efficiently when compared to using the mouse.

In the ”Four Blocks” scene (Figure 11(d)) the mouse controller gives a lower average time for task completion. This highlights a limitation of the current approach: the crowd must remain as a cohesive whole. With the central placement of the crowd in the ”Four Blocks” scene the best strategy to collect items is to split up the crowd and send them to different corners of the environment simultaneously, something the mouse controller is able to do more effectively than our multitouch controller.

In addition to completing the above tasks, participants were also asked a set of questions with regards to their experience of each control mechanism. Figure 14 shows the average scores given by participants for each of the questions outlined in Figure 15. In all cases both the mouse and the multitouch control scheme averaged a score of between 3 and 4, with the multitouch control showing a better score in the question concerning participants’ overall view of the control scheme. The slightly better score for the mouse control scheme in questions 1-3 may in part be due to the familiarity of the participants with using a mouse device. A number of the participants commented that their experience of using a mouse device in real-time strategy as well as other games may have meant that they favoured this device implicitly, through what may be referred to as a “mouse prior”. The multitouch device, despite being comparably less common than the mouse, still showed strong scores in response to the questions and this suggests that such devices are appealing as a method for interactive crowd control.

We observe that other formation control interfaces, e.g. sketching-based [4], [17], utilise a control scheme similar to the mouse controller. Such controllers have limited responsiveness due to requiring multiple passes to direct a crowd. This suggests that the multitouch controller would produce lower task completion times compared to such interfaces, particularly given the advantages of the multitouch’s single-pass control shown in the current study. This would be interesting to perform as a follow-up to this study in future work.

7.6 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 multitouch input we used a G4 multitouch overlay from PQ labs, attached to a 24” Acer S240HL LCD monitor. 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 ∼32 frames per second. We found that with

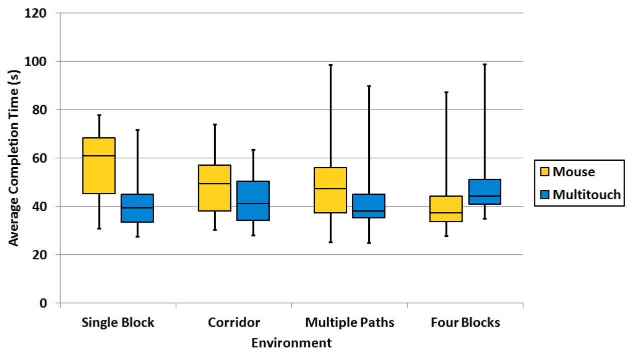


Fig. 13. Box-whisker plot of task completion time for each controller in four different environments

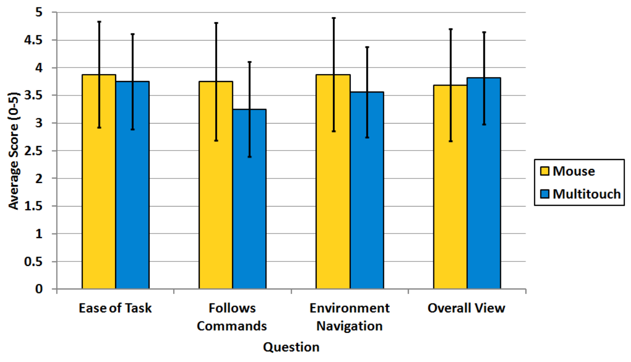


Fig. 14. Average scores and their standard deviations of both controllers given in response to user study questions (see Figure 15 for the full questions)

Question 1: *Please rate how you found it to complete the tasks using this control mechanism. (0 = Very Hard, 5 = Very Easy)*
Question 2: *Please rate how well you felt the characters followed your commands. (0 = Not At All, 5 = Very Much So)*
Question 3: *Please rate how you found it to navigate your characters in the various environments using this control mechanism. (0 = Very Hard, 5 = Very Easy)*
Question 4: *Please rate your overall experience of this control mechanism. (0 = Very Poor, 5 = Very Good)*

Fig. 15. The questions presented to each user after completion of the tasks using either the mouse or the multitouch controller.

the current unoptimised implementation, framerates reduced to around 8-10fps at a crowd size of >160 making interactive control of the crowd quite difficult.

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 [31]. 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. We also enhanced the motion of characters in the simulation by embedding motion data into the environment in the form of patches. This motion data is incorporated into the character's path planning by including information about the motion in the cost metric. Our method provides a more enriching user experience in real-time applications such as games. A comprehensive user study shows the advantage of using our method for navigating a crowd through a given environment and suggests that a multitouch device is a promising medium through which to provide user control of virtual crowds. 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.

In this work we have presented actions that either passively or actively interact with the environment (such as running and avoiding, or crawling and climbing respectively). We would like to extend this approach by considering motion data involving character-character interactions. A modified version of the patch-based approach could enable us to simulate scenes involving dynamic interactions between characters, such as two armies fighting [27].

Although agents consider the future motion of dynamic obstacles in the simulation through use of the method in [6] we currently do not account for dynamic obstacles when planning the mesh movement. This choice was made to give greater control of the mesh to the user. Having the mesh follow an optimal path rather than that specified by the user may make the user feel less in control of the crowd and thus degrade their experience. That being said, an RVO-like obstacle avoidance mechanism [32] would provide greater intelligence to the mesh motion, for example preventing it from passing in front of moving cars, and consequently produce smoother motion.

With the current framework, the computational cost of the MTS imposes a bottleneck on the size of the crowd that can be controlled interactively. Assigning the position of groups of agents as opposed to individuals in the formation would help to alleviate this cost for much larger crowds. A hierarchical system could be used to assign varying sized groups to their appropriate positions in the formation.

Finally, as indicated by the user study, the implicit group cohesion in our method limits the crowd's ability to multi-task. A future development of this work would consider approaches for splitting and merging of the crowd and user specification of subgroups. Alternative methods for shape manipulation through the multitouch device may provide fruitful avenues for such research.

References

[1] P. Kanyuk, "Brain springs: fast physics for large crowds in WALL-E," *IEEE Comput. Graph. Appl.*, vol. 29, no. 4, pp. 19–25, Jul. 2009.

[2] T. Kwon, K. H. Lee, J. Lee, and S. Takahashi, "Group motion editing," in *SIGGRAPH '08: ACM SIGGRAPH 2008 papers*. New York, NY, USA: ACM, 2008, pp. 1–8.

[3] S. Takahashi, K. Yoshida, T. Kwon, K. H. Lee, J. Lee, and S. Y. Shin, "Spectral-based group formation control." *Comput. Graph. Forum*, vol. 28, no. 2, pp. 639–648, 2009.

[4] Q. Gu and Z. Deng, "Formation sketching: an approach to stylize groups in crowd simulation," in *Proceedings of Graphics Interface 2011*, ser. GI '11. School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada: Canadian Human-Computer Communications Society, 2011, pp. 1–8.

[5] Y. Rubner, C. Tomasi, and L. J. Guibas, "A metric for distributions with applications to image databases," in *Proceedings of the Sixth International Conference on Computer Vision*, ser. ICCV '98. Washington, DC, USA: IEEE Computer Society, 1998, pp. 59–.

[6] A. Treuille, S. Cooper, and Z. Popović, "Continuum crowds," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 1160–1168, 2006.

[7] J. Henry, H. P. H. Shum, and T. Komura, "Environment-aware real-time crowd control," in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ser. SCA '12. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2012, pp. 193–200. [Online]. Available: http://dl.acm.org/citation.cfm?id=2422356.2422384

[8] D. Helbing, I. Farkas, and T. Vicsek, "Simulating dynamical features of escape panic." *Nature*, vol. 407, no. 6803, pp. 487–490, September 2000.

[9] H. Yeh, S. Curtis, S. Patil, J. van den Berg, D. Manocha, and M. Lin, "Composite agents," in *SCA '08: Proceedings of the 2008 ACM SIGGRAPH/Eurographics symposium on Computer animation*. ACM, 2008.

[10] J. Ondrej, J. Pettr, A.-H. Olivier, and S. Donikian, "A synthetic-vision-based steering approach for crowd simulation," *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2010)*, vol. 29, no. 4, 2010.

[11] R. Narain, A. Golas, S. Curtis, and M. C. Lin, "Aggregate dynamics for dense crowd simulation," *ACM Trans. Graph.*, vol. 28, no. 5, pp. 1–8, 2009.

[12] M. Sung, M. Gleicher, and S. Chenney, "Scalable behaviors for crowd simulation," pp. 519–528, September 2004.

[13] A. Lerner, E. Fitusi, Y. Chrysanthou, and D. Cohen-Or, "Fitting behaviors to pedestrian simulations," in *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ser. SCA '09. New York, NY, USA: ACM, 2009, pp. 199–208.

[14] K. H. Lee, M. G. Choi, Q. Hong, and J. Lee, "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: Eurographics Association, 2007, pp. 109–118.

[15] O. Sorkine, Y. Lipman, D. Cohen-Or, M. Alexa, C. Rössl, and H.-P. Seidel, "Laplacian surface editing," in *Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, 2004, pp. 179–188.

[16] I. Karamouzas and M. Overmars, "Simulating and evaluating the local behavior of small pedestrian groups," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 18, no. 3, pp. 394–406, 2012.

[17] Q. Gu and Z. Deng, "Generating freestyle group formations in agent-based crowd simulations," *Computer Graphics and Applications, IEEE*, vol. 33, no. 1, pp. 20–31, 2013.

[18] M. J. Park, "Guiding flows for controlling crowds," *Vis. Comput.*, vol. 26, no. 11, pp. 1383–1391, Nov. 2010.

[19] M. Oshita and Y. Ogiwara, "Sketch-based interface for crowd animation," in *Proceedings of the 10th International Symposium on Smart Graphics*, ser. SG '09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 253–262.

[20] M. Kim, K. L. Hyun, J. Kim, and J. Lee, "Synchronized multi-character motion editing," *ACM Trans. Graph.*, 2009.

[21] J. Kato, D. Sakamoto, M. Inami, and T. Igarashi, "Multi-touch interface for controlling multiple mobile robots," in *Proceedings of the 27th international conference extended abstracts on Human factors in computing systems*, ser. CHI EA '09. New York, NY, USA: ACM, 2009, pp. 3443–3448.

[22] S. Patil, J. van den Berg, S. Curtis, M. C. Lin, and D. Manocha, "Directing crowd simulations using navigation fields," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 2, pp. 244–254, Feb. 2011. [Online]. Available: http://dx.doi.org/10.1109/TVCG.2010.33

[23] Q. Gu and Z. Deng, "Context-aware motion diversification for crowd simulation," *Computer Graphics and Applications, IEEE*, vol. 31, no. 5, pp. 54–65, 2011.

[24] B. Yersin, J. Maïm, J. Pettré, and D. Thalmann, "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: ACM, 2009, pp. 207–214.

[25] K. H. Lee, M. G. Choi, and J. Lee, "Motion patches: building blocks for virtual environments annotated with motion data," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 898–906, 2006.

[26] M. Kim, Y. Hwang, K. Hyun, and J. Lee, "Tiling motion patches," in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ser. SCA '12. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2012, pp. 117–126. [Online]. Available: http://dl.acm.org/citation.cfm?id=2422356.2422375

[27] H. P. H. Shum, T. Komura, M. Shiraishi, and S. Yamazaki, "Interaction patches for multi-character animation," *ACM Trans. Graph.*, vol. 27, no. 5, pp. 1–8, 2008.

[28] T. Igarashi, T. Moscovich, and J. F. Hughes, "As-rigid-as-possible shape manipulation," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 1134–1141, 2005.

[29] M. S. Floater, "Mean value coordinates," *Computer Aided Geometric Design*, vol. 20, 2003.

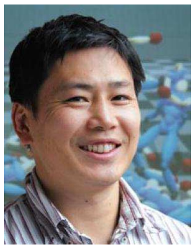
[30] J. Tsitsiklis, "Efficient algorithms for globally optimal trajectories," *Automatic Control, IEEE Transactions on*, vol. 40, no. 9, pp. 1528–1538, 1995.

[31] M. Lau and J. J. Kuffner, "Behavior planning for character animation," in *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*. New York, NY, USA: ACM, 2005, pp. 271–280.

[32] J. van den Berg, S. Patil, J. Sewall, D. Manocha, and M. Lin, "Interactive navigation of individual agents in crowded environments," in *I3D '08: Proceedings of the 2008 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*. ACM, 2008.

Joseph Henry received the MSc degree in Artificial Intelligence in 2010 from the University of Edinburgh and a First class BSc degree in Neuroscience in 2009 from Cardiff University. He is currently working toward a PhD degree in the School of Informatics at the University of Edinburgh. His research interests include artificial intelligence, crowd and character simulation, and robotics.

Hubert P. H. Shum is a Senior Lecturer (Assistant Professor) at Northumbria University. Before joining the university, he worked as a Lecturer in the University of Worcester, a post-doctoral researcher in RIKEN Japan, as well as a research assistant in the City University of Hong Kong. He received his PhD degree from the School of Informatics in the University of Edinburgh. His research interests include character animation, machine learning, human computer interaction and physical simulations.



Taku Komura is a Reader (Associate Professor) at the Institute of Perception, Action and Behaviour, School of Informatics , University of Edinburgh. As head of the Computer Animation and Robotics group his research has focused on data-driven character animation, physically-based character animation, crowd simulation, cloth animation, anatomy-based modelling and robotics. Recently, his main research interests have been in indexing and animating complex close interactions, which includes character-character interactions and character-object interactions.