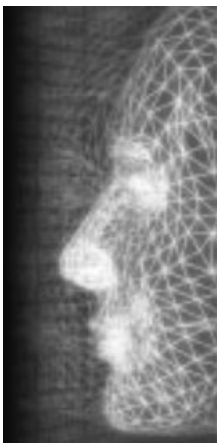
COMPUTER ANIMATION AND VIRTUAL WORLDS
Comp. Anim. Virtual Worlds 2008; **19**: 319–330
Published online 25 July 2008 in Wiley InterScience
(www.interscience.wiley.com) DOI: 10.1002/cav.231

Shape-constrained flock animation



By Jiayi Xu, Xiaogang Jin, Yizhou Yu, Tian Shen and Mingdong Zhou*

We propose a novel shape-constrained flock animation system for interactively controlling flock navigation in virtual environments. This system is capable of making the spatial distribution of a flock meet static or deforming shape constraints while performing flock simulation. Such a capability can find many applications in the entertainment industry. Given a 3D constraining shape, our system first draws a set of uniform sample points through a 3D surface mosaicing process or a stratified point sampling strategy. Once correspondences between flock members and sample points have been established, points on the target shape are used as homing destinations to guide flock migration. Under a global path control scheme, an effective fuzzy control logic, which dynamically adjusts steering forces and control forces, has been developed to create visually pleasing shape-constrained flock animations. Copyright © 2008 John Wiley & Sons, Ltd.

Received: 23 June 2008; Accepted: 23 June 2008

KEY WORDS: flock animation; shape constraint; fuzzy control; Kalman filter; global path control

Introduction

A flock of birds or a school of fish can exhibit beautifully orchestrated group behaviors. We are often amazed at their elegance as well as their level of coordination and synchronization. This is especially true when a flock assumes the approximate shape of a real object. It is indeed an exhilarating event because of its rareness. We would like to develop techniques for digitally reproducing such effects. This type of techniques have many applications in the entertainment industry especially in advertising and film making because group animation techniques have been widely applied there recently to either reduce the shooting cost or create scenarios not existing in the real world. Our goal in this paper is to introduce methods that produce physically plausible flock animation, which at the same time, assume a recognizable static or deforming shape. Although every member of a flock follows the overall behavior of a leader, the actual velocity, and motion trajectory of the member may have local fluctuations. Therefore, "plausible" means such local visual characteristics of flocking should be retained.

A flock in general consists of discrete creatures, each with a set of parameterized rules governing its behavior. The control rules can be inferred from user-defined scripts, local forces, and so on. Pioneering work of flock animation can be traced back to the seminal work of Reynolds,[1] which is popularly known as local control agents of flock animation. This model demonstrated that flocking behavior could be generated from simple local rules.

Controlling the movement of flocks has often been a desired goal during flock simulation. Many previous works only allowed users to set initial conditions at the very beginning of such simulation. In the literature, only a few methods were devoted to the interactive control of flocks. Crowdbrush, proposed by Ulicny *et al.*,[2] is featured with convenient graphical interface elements for controlling crowds. But the control operations are still limited to individuals by specifying the property or response rule of agents, which becomes tedious work for large crowds.

Vector fields have proved to be useful in guiding the animation of particle systems. It was first proposed by Reynolds,[3] where a vector field can be used as a velocity field to drive the flow direction of a flock. However, an effective and efficient method to calculate such a field was not described. Other papers motivated by this idea use the concept of fields, for example, velocity field, force field, potential field, or road-map to represent the velocity or moving direction of an entire flock. Nevertheless, controlling flock behavior in a global setting without any

*Correspondence to: X. Jin, State Key Laboratory of CAD & CG, Zhejiang University, Hangzhou 310027, P.R. China.
E-mail: jin@cad.zju.edu.cn

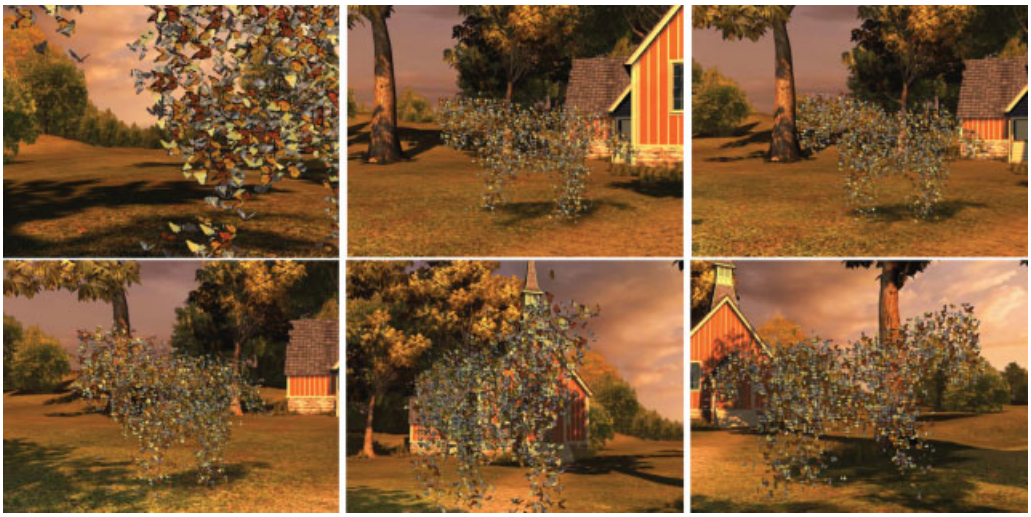


Figure 1. A flock of butterflies flies in the virtual environment and tracks a running horse.

autonomous behavior among individuals would not be adequate. In our simulation system, besides the naturally inherited flock steering property, we introduce a vector field simulating external environmental forces in a scene.

In this paper, we propose an agent-based flock simulation system that is capable of satisfying user-specified static or dynamic shape constraints. The global flocking behavior emerges from the action of the individuals, producing large-scale flock animations in real time that is otherwise hard to achieve. As an example, in Figure 1, we have created a simulation to show a flock of butterflies that covers the surface of a horse flying under user-specified route in the virtual environment while keeping visual characteristics of flocking.

Our approach has the following contributions:

- During simulation, we can make an entire flock meet shape constraints with arbitrary topology. Desirable visual effects have been achieved using a surface mosaic algorithm or stratified point sampling method.
- We have employed the Kalman filter[4] to compute local control forces necessary for tracking dynamic shape constraints. Tracking can be successfully performed even when the target shape deforms rapidly.
- By carefully integrating the effects of basic steering forces, local control forces and external vector fields using a fuzzy logic, individual flocking behavior can be naturally generated on the fly.
- Using a global path control scheme, we can impose hard constraints on agents' positions at specific times while retaining visual characteristics of flocking.

The rest of the paper is organized as follows. Section "Related Works" reviews related works. Section "Overview" gives an overview of our framework. Section "Shape Constraints" addresses issues related to the initialization and setup of shape constraints. Section "Local Force Model" gives a detailed description of our local control model. Section "Global Path Control" discusses our global path control scheme. Experimental results will be presented in Section "Results," and we come to the conclusions in Section "Conclusion and Discussion."

Related Works

A natural way to simulate large crowds is based on agents who have individualized behavior. This approach is attractive for several reasons. First, such models are able to mimic complex global behavior with a few simple local control rules. In addition, such models can capture each agent's personality, which is helpful in yielding complex heterogeneous behavior.

Reynolds proposed a distributed behavioral model that described the behavior of large groups of birds, herds, and fish with perceptual skills existing in the real world.[1] This pioneering work can generate complex flocking animation from a few predefined rules, mainly separation, cohesion, and alignment rules. Since this work, flock animation has attracted widespread attention in crowd simulation. Each individual of a flock acts according to its local perception of the dynamic environment. Extensions of this approach integrate locally

DOI: 10.1002/cav

controlled agents with other models. Erra *et al*.[5] mapped steering behaviors model onto the GPU. The testing shows that simulating 4000 boids runs at 30 frames per second. Tu and Terzopoulos[6] created behavioral animation for artificial fishes, where virtual agents are endowed with synthetic vision and perception of the environment. Bayazit *et al*.[7,8] studied four kinds of group behaviors, namely, homing, exploring, passing through narrow areas, and shepherding. Global information in the form of a roadmap enables these flocking behaviors. Furthermore, the global knowledge of the environment is updated by communication between individuals. Lien *et al*.[9,10] addressed the problem of shepherding flocks in their system. The work concentrated on the implementation of how a group of shepherds work cooperatively without communication to efficiently control the flock. Once again, the simulated behaviors do not accurately resemble those in real life. Saber[11] proposed in his theoretical framework three distributed flocking algorithms. Two for free flocking and one for constrained flocking with the presence of multiple obstacles. Among them, cost functions are employed to model the migration of flocks. However, no animation result is provided. Hartman and Benesš[12] simulated bird-like creatures based on the work of Reynolds. They introduced a complementary force to define the chance of the boid to become a leader and try to escape. The simulation runs at 30 frames per second with hundreds of boids. Skrba *et al*.[13] described in their paper how to animate and render large herds of furry animals. In their system, a leading sheepdog is in charge of the cohesion and separation behaviors of sheep. The rendering can be real-time using impostors. However, only a few basic behaviors have been explored. Sometimes, visual artifacts may appear.

Previous works had addressed interactive control of flock simulation. But complex static or dynamic shape constraints have not been thoroughly investigated. Anderson *et al*.[14] used an iterative sampling method to generate user-constrained group animation. The constraints are agent based. However, this method is computationally intensive. Lai *et al*.[15] produced controlled flock behaviors with significantly reduced computational cost by building group motion graphs. Their model relies on clustering group configurations from sample animations. Wojtan *et al*.[16] approached this problem from a nonlinear constraint minimization point of view. The actual minimization is performed using the adjoint method. However, the type of control that can be applied is quite limited, typically in the form of a straight path or simple planar shapes.

Vector fields have been explored in the animation of particle systems. Xiao and Hubbold[17] introduced artificial force fields to guide navigation in virtual environments. The flow tiles model developed by Chenney[18] ensures the smooth flow of agents without resorting to explicit collision detection. Sakuma *et al*.[19] used a vector field for navigation in their psychological model. They designed a vector field that covers the scene using an attractive and repulsive force model. Pedestrians determine their moving direction by referring to the vector field at the present location. Relying on global vector fields but no autonomous movement component, realism has been sacrificed to some extent in crowd simulations produced in most previous works. Treuille *et al*.[20] integrated global navigation with a resolution-dependent dynamic potential field, solving the motion of crowd. However, shape constraints were not considered in their approach.

Overview

Figure 2 gives an illustration of our framework. In this paper, we focus on interactively controlling the behavior of huge flocks. The members of a flock are constrained to a given static or deforming shape during the animation. First of all, we perform a uniform surface sampling operation on the input target shape. We designate each agent in a flock a destination on the 3D shape. Driven by local fuzzy control and global path control, individual agents move toward their destinations along user-defined trajectories.

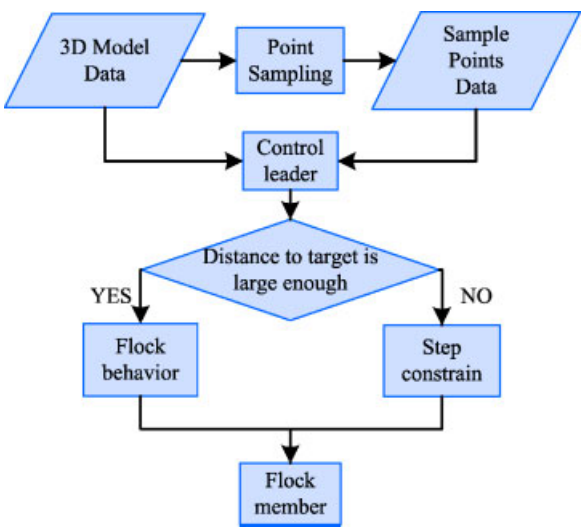


Figure 2. The framework of our flock model.

Comp. Anim. Virtual Worlds 2008; **19**: 319–330
DOI: 10.1002/cav

To process a 3D object, we first cover the surface with a set of evenly distributed sample points based on the surface mosaicing technique in Reference [21] or the stratified point sampling method as described in Reference [22]. Secondly, in accordance with the relative location of these sample points, we establish correspondences between agents in the flock and sample points on the surface.

To smoothly transform the present spatial distribution of agents to the distribution of samples on the target surface, each individual agent moves itself from the present source location to its corresponding destination obeying local control rules. This destination of an agent controls its homing behavior, global route, and other additional adjustments. When an agent remains far from its destination, its behavior primarily complies with that of Reynolds's boid model,[1] which consists of separation, cohesion, and alignment rules. When an agent becomes close to its destination, additional local control forces, including homing and damping forces, are applied in order to smoothly reach the destination. In the most complex situation when target itself is moving, Kalman filter is incorporated; therefore, the agent will keep pace with its target object. The overall performance is emerged from the behaviors of all flock members.

A variety of creatures and objects can be the agents of a flock. They can be represented using static meshes or animation models.

Following the aforementioned approach, we can make a flock smoothly transforming among different static shapes as well as tracking a deforming shape.

Shape Constraints

As discussed, we impose shape constraints on an entire flock. Such constraints are realized by establishing correspondences between flock members and sample points on the shape. In this paper, we choose to have shape constraint on the model surface instead of inside a 3D model. The reason is simply as for small creatures like butterflies, bees, ants, etc., prevalence in the physical world; they tend to cover a certain real object, a trunk for example, instead of filling a volume. Since the constraining shapes are represented as triangle meshes, we aim to spread these sample points evenly on the mesh surfaces. In the experiments shown in this paper, we always apply one of the following two sampling methods: surface mosaic sampling and stratified point sampling.

For better performance, the sample positions on the target surface are obtained in the preprocessing step. They are retrieved later during every time-step of flock simulations. If the shape constraint is a deforming object, we only perform this preprocessing for the object shape in the first frame of the deformation sequence assuming there exist vertex correspondences throughout the entire sequence.

Surface Mosaic Sampling

In order to capture the shape of an object, we present an optimized sampling technique based on 3D surface mosaicing methods[21,23] with necessary revisions.

Once the 3D model and number of sample points have been determined, we assign the initial position of sample points using random sampling. After the random initialization, the final position of sampling points is defined by an energy minimizing iterative process. We assume each flock member contains spring-like energy. The energy between any two gives rise to a repulsive force, which helps define the final stable position of the flock members. In practice, we only consider repulsive forces between neighbors whose Manhattan distance is less than A, where A is related to the number of elements N and the total surface area M as follows:

$$A = k\sqrt{\frac{M}{N}} \tag{1}$$

where k is 2.0 in our current implementation. Thus, the spring-like energy between points i and its neighbor j can be defined as

$$E_{ij} = e^{\frac{-\mathrm{dis}_{ij}^2}{2\sigma^2}} \tag{2}$$

where σ is the interaction radius. It is used to control the fall-off of potential energy, and does not affect the final pairwise distances in equilibrium. σ can be set as follows:

$$\sigma = w\frac{M}{N} \tag{3}$$

In this version, we choose w as 0.9. Decrease the value of w will increase the influence of j to i, which will eventually cause the convergence of sampling process slowly and differently. We can choose dis_{ij} as Manhattan distance or Euclidean distance. Manhattan distance tends to produce quadrangular distributions and Euclidean distance tends to produce hexagonal ones.

In principle, the gradient of this energy introduces repulsive force. In addition, we will make sure that the repulsive force be restricted to the tangent plane.

DOI: 10.1002/cav

1: Compute the value of M, σ and A
2: **for** each step **do**
3: **for** each neighborhood j of i **do**
4: Calculate the distance between i and j
$\text{dis}_{ij} = |p_i(x) - p_j(x)| + |p_i(y) - p_j(y)| + |p_i(z) - p_j(z)|$
5: Compute repulsive force $F_i = \sum_{j \in U} \text{norm}(p_i - p_j) \times E_{ij}$
6: Compute new position of i $p_i' = p_i + \sigma t \times F_i$
7: **end for**
8: Projection p_i' onto the mesh
9: **end for**

Table 1. Algorithm 1: overall sampling process

However, the computation is quite involved and does not ensure a simple closed-form solution. By using a simplified version introduced by Lai *et al.*,[21] we define the repulsive force as

$$F_i = \sum_{j \in U} \text{norm}(p_i - p_j) \times E_{ij} \tag{4}$$

where U represents the neighborhood of point i, and i does not belong to U. Function norm$(\cdot)$ returns the normalized input vector.

Finally, we arrive at the new position of point i after each iteration:

$$p_i' = p_i + \delta t \times \sum_{j \in U} \text{norm}(p_i - p_j) \times E_{ij} \tag{5}$$

Here, δt is the size of the time step. However, the new position of point i may not lie on the model surface, an orthogonal projection back onto the mesh is necessary.

The overall sampling process of a triangular mesh model is listed in Table 1.

Stratified Point Sampling

The surface mosaic sampling process is computationally expensive for complex objects. If the uniform point distribution is not required, we choose the stratified point sampling strategy proposed by Nehab and Shilane.[22]

This algorithm involves three steps. First of all, the model is voxelized using octree. After that, one triangle is chosen from each voxel and from this triangle a new sample is produced. Instead of choosing the surface point closest to the center of each voxel, we pick a vertex according to a probability distribution that favors points close to the center, while allowing for a user controllable variation. In this sampling procedure, we use an exponential distribution function $\lambda e^{-\lambda d}$ with the distance d between the sample point and the center of the original voxel. Each triangle in a voxel is subdivided until the probability density function can be considered constant throughout its area. The function value at the centroid of a terminal subtriangle is multiplied by its area and is defined as the priority. The subtriangle for sampling is selected according to this priority. In the last step, we eliminate those sample points that are too close to each other. Our solution is to enforce minimum distance between any two samples.

Point Correspondence Establishment

When establishing correspondences between flock agents and sample points on the constraining shape, a random assignment suffices in most scenarios. Nevertheless, when there is a sequence of constraining shapes and subsequent constraining shapes have a large overlap, random assignment may generate chaotic results. To achieve better temporal coherence between two overlapping constraining shapes, we obtain initial correspondences by spherically projecting source positions on the first shape onto the surface of the second shape, and let these projected points subject to energy optimization. Because of the initial correspondences, during the interactions, the movement of the projected points on the second shape is in a relatively small scale. As a result, the generated flock animation becomes fluent.

Local Force Model

In this paper, we focus on simulating large groups of creatures acting in concert using flocking models. In this section, we propose a fuzzy local control model for simulating shape-constrained flock animation. We treat each

DOI: 10.1002/cav

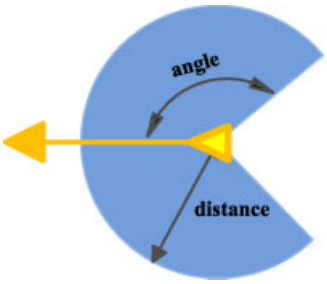


Figure 3. The definition of perceptual neighborhood of a boid.

agent in the flock as a particle. Its actual behavior is determined by a series of steering forces and control forces, which we will introduce in the following sections.

Basic Steering Behaviors

Basic flocking behaviors used here comply with Reynolds's boid model.[1] Each flock member observes three steering rules: separation, cohesion, and alignment. The definitions of these steering rules rely on the neighbors of the flock member. The neighbors of one flock member include its surrounding partners who are sufficiently close and also within the angle of perception (Figure 3).

In our technique, every agent maintains its own attribute list, including maximum speed, maximum acceleration, viewable neighborhood radius, and angle. The overall behavior of each agent is mainly determined by the mixture of these different steering forces. Thus, we assign every steering force a different weight to express individuality. More specifically, each agent has the following parameters:

- *MaxForce*, *MaxSpeed*;
- *SeparationRadius*, *SeparationAngle*, *SeparationWeight*;
- *AlignmentRadius*, *AlignmentAngle*, *AlignmentWeight*;
- *CohesionRadius*, *CohesionAngle*, *CohesionWeight*.

where *MaxForce* and *MaxSpeed* are used for constraining flock behavior within a reasonable range. *SeparationRadius* and *SeparationAngle* are, respectively, the perceptual radius and angle in the separation rule. The meaning of other parameters can be inferred similarly. Figure 4 illustrates the definitions of separation, cohesion, and alignment steering rules. Each rule defines its own perceptual radius and angle. Usually, the separation radius is the largest one while the alignment radius the least one. The separation angle and the cohesion angle are usually larger than the alignment angle. In summary, the radius of each steering rule is related to the number of members in the flock N:

$$r = \frac{J}{\sqrt[2]{N}} \tag{6}$$

where J is a random floating point number which is different for these three kind of radius. In our present implementation, 60.0 to 120.0 for separation radius, 30.0 to 120.0 for cohesion radius, 30.0 to 60.0 for alignment radius are appropriate.

In most crowd control methods, collisions are prevented in advance locally by adjusting direction and speed when other agents or obstacles approach. In our flock model, because of the separation force, agents move away from each other automatically when they become too close. Therefore, we do not explicitly handle collision avoidance.

External Vector Fields

Since a flock typically consists of lightweight creatures or objects and they are immersed in the air or water, their motion is inevitably influenced by the surrounding fluid medium. Thus, considering the influence from the surrounding medium makes it possible to produce more natural flocking animations. On the other hand, the fluid medium has its own dynamics. But we choose to only

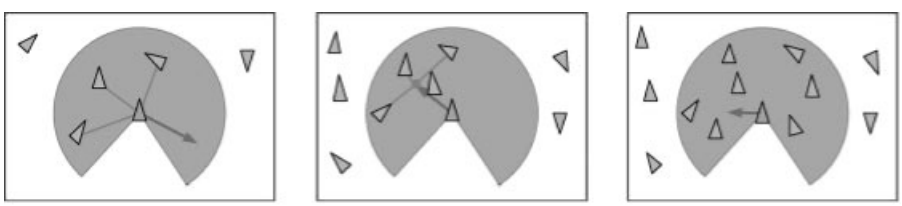


Figure 4. Separation, cohesion, and alignment rules. (a) Separation: maintains a certain separation distance from other nearby flockmates. (b) Cohesion: coheres with the average position of other nearby flockmates. (c) Alignment: aligns itself with other nearby flockmates.

DOI: 10.1002/cav

focus on simulating the motion of the agents themselves because they are directly visible and do not strongly influence the large-scale dynamics of the fluid. The external force field, such as a wind force field, the flock receives from the surrounding medium is modeled as a vector field directly specified by the user. This vector field is smooth and can be time varying during the simulation.

For example, to simulate a simple rotational vortex field, the vector field at any position can be computed as

$$p = \text{Pos} - \text{Pos}_{\text{center}} \tag{7}$$

$$F = d \times \text{Vec3}(p.z, 0, -p.x) \tag{8}$$

where Pos is the position of an agent, $\text{Pos}_{\text{center}}$ is the center of the vortex. Vec3 represents a vector, d is a constant that can be used to scale the effect of this vector field.

Homing Behavior

As stated by Reynold, the original boid model can only model flock wandering behavior. Sometimes, people are more interested in pointing flock with time varying designation. In this section, we present an enhanced version by introducing a new steering behavior: *homing*. This technique is especially useful when dealing with dynamic tracking problem which is hard to implement using existing methods.

In our algorithm, all the simulated agents belong to one flock. Each agent is designated a virtual destination. This homing behavior always drives an agent toward its own destination. As a result, if the destination is defined to a point on the constraining shape, the agent will move gradually to the destination point while maintaining the dynamic characteristics of a boid. Note that in the event that the constraining shape is moving or deforming, the agents need to approach their dynamically changing destination points. This is actually a dynamic tracking problem, and our solution is based on the Kalman filter[4] which was originally designed for similar purposes. See Figure 5 for an explanation of Kalman filter. Its specific equations for time and measurement updates are listed in the following equation array:

Kalman filter time update equations:

Predicted state: $\widehat{x}_{k|k-1} = A\widehat{x}_{k-1|k-1} + Bu_k$;

Predicted estimate covariance: $P_{k|k-1} = AP_{k-1|k-1}A^T + Q$;

Kalman filter measurement update equations:

Optimal Kalman gain: $K_k = P_{k|k-1}H^T(HP_{k|k-1}H^T + R)^{-1}$;

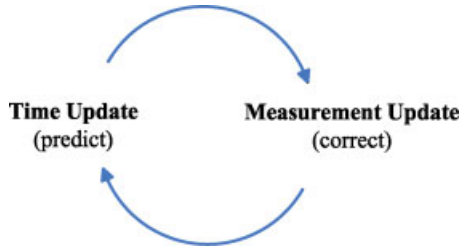


Figure 5. A Kalman filter cycle. The time update projects the current state estimate ahead in time. The measurement update adjusts the projected estimate by an actual measurement at that time.

Updated state estimate: $\widehat{x}_{k|k} = \widehat{x}_{k-1|k-1} + K_k(z_k - H)\widehat{x}_{k-1|k-1}$;

Updated estimate covariance: $P_{k|k} = (I - K_kH)P_{k-1|k-1}$.

In our solution, the state of each agent in a flock is evaluated using a series of linear stochastic difference equations provided that local steering and external forces, current position, and velocity are known. For each agent, we record two positions, its current position and the position of its virtual destination. We treat the agent's state from time step $k-1$ to k as the predicted value $\widehat{x}_{k|k-1}$, $\widehat{x}_{k|k-1} = \begin{bmatrix} \text{Pos}_{k-1} \\ \text{Vel}_{k-1} \end{bmatrix}$, where Pos and Vel are the position and velocity of the agent. Thus, we obtain $A = \begin{bmatrix} I & \delta t I \\ 0 & I \end{bmatrix}$, $B = \begin{bmatrix} 0 & 0 \\ 0 & \delta t \end{bmatrix}$, $u_k = a$, where a is the acceleration. On the other hand, we set the virtual leader's state as the measurement z_k and use it to generate *a posteriori* state estimate $\widehat{x}_{k|k}$, $\widehat{x}_{k|k} = \begin{bmatrix} \text{Pos}_k \\ \text{Vel}_k \end{bmatrix}$ which represents the actual state the agent will reach at time step k. Covariance Q and R are defined by users. With the computed posteriori state estimate, the *homing* force can be formulated as

$$F_h = \frac{2 \times m \times (\text{Pos}_k - \text{Pos}_{k-1} - \text{Vel}_{k-1} \times \delta t)}{\delta t^2} \tag{9}$$

where m represents the mass of the agent.

Note that when covariance Q is fixed, covariance R can be dynamically adjusted to control how closely the agents follow the virtual destinations. When R becomes zero, $\widehat{x}_{k|k}$ coincides with the state of the virtual leaders. In principle, R should always be bounded away from zero to allow local fluctuations and maintain visual characteristics of flocking.

DOI: 10.1002/cav

Fuzzy Control Logic

In our flock simulation, all agents eventually need to stay close to their destinations on the constraining shape. When this happens, the density of agents around the constraining shape will increase significantly and the magnitude of their interaction forces will also increase significantly. Strong interaction forces will give rise to a chaotic situation where agents hover back and forth around their destinations but cannot reach a relatively steady state.

In summary, we propose a fuzzy control method to tackle this problem by associating an adjustable weight with each type of forces. External vector field forces is always added to reflect the natural fluctuation. When an agent is far away from its destination, its dynamics are basically governed by basic steering forces. Thus, the weights for these types of forces should be large while the weights for other forces should be small. When the agent moves closer to its destination, different strategies are used depending on whether we are performing transformation between a series of static model objects or dealing with dynamic tracking problem. In the former case, we increase the weight of an additional damping force to direct the movement. This damping force plays the role of slowing down the agent nearby its destination and prevents excessive overshooting. While for the later one, the weights for basic steering forces and external vector field forces decrease, and the weight for the homing force increases.

Actually when the target positions are rapidly changing, a portion of the agents may not catch up with them, reflecting the different personality of the agents. When an agent almost reaches its destination, we use a stepped steering method to perfect the result. In other words, by setting up a local coordinate frame for every one, we rotate and move it in a right pace to navigate it smoothly toward the target position. When the destination is finally reached, the agent can either be completely bound to the destination or float around it.

Figure 6 shows the weight curves for basic steering forces, homing force, damping force, and external vector field force, respectively. They are piecewise continuous functions.

Figure 7 shows the comparison between two examples. The left picture of Figure 7 indicates a distribution of flock agents with relatively large basic steering forces among them. While in the right one, the flock has a smaller weight for the basic steering forces, therefore, its spatial distribution appears more concentrated.

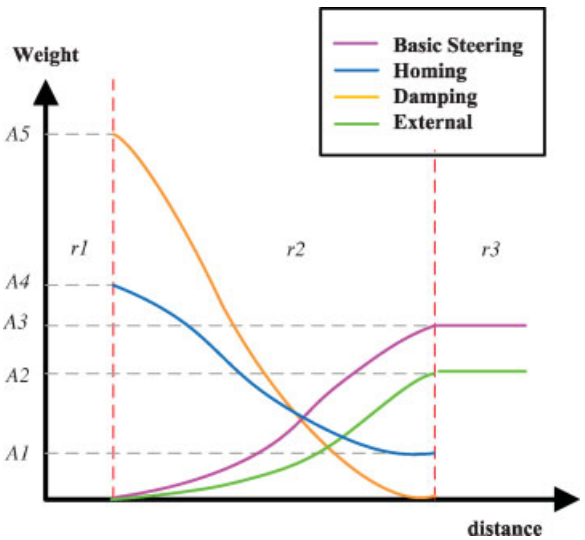


Figure 6. Weighting scheme for basic steering forces, homing force, external force, and damping force.

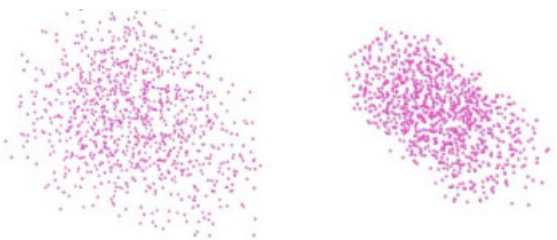


Figure 7. Comparison of a crowd of flock with a non-flock one.

Global Path Control

In order to drive the motion of an entire flock in a more coherent manner when it migrates from one constraining shape to another, users can interactively specify a global path between the centers of the two shapes. We use a cubic B-spline curve $R(u)$ interpolating end points for representing the global path. A similar path between every pair of corresponding sample points on the two shapes can be obtained automatically. The agents in the flock will follow these paths during migration. As the B-spline path curve for each agent is smooth and seldom interleaving due to the good quality of point correspondence, the simulation runs fluent.

Let O_1 and O_2 represent the centers of the source mesh and the target mesh, respectively. We set O_1 as the start point, O_2 as the end point of the B-spline curve. Between O_1 and O_2, users can insert other control points. We re-parameterized the path using its arc length in order to

Comp. Anim. Virtual Worlds 2008; **19**: 319–330
DOI: 10.1002/cav

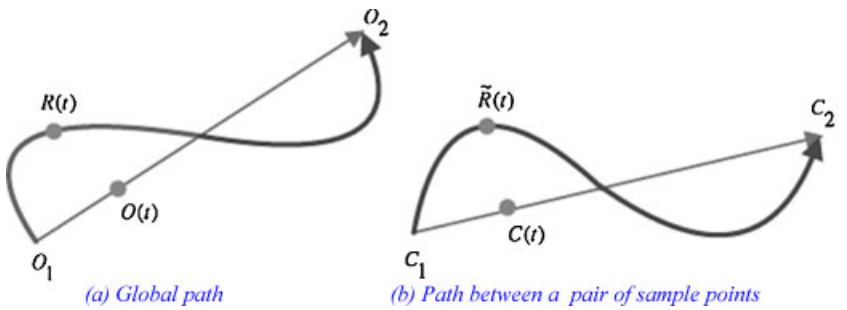


Figure 8. User-designed global path (a) and the computed path for one pair of sample points (b).

better control the speed.[24] Similarly, assuming C_1 and C_2 are two corresponding points on the two constraining shapes, we get the expression $C(t)$, $t \in [0, 1]$.

$$O(t) = (1-t)O_1 + tO_2 \tag{10}$$

$$C(t) = (1-t)C_1 + tC_2 \tag{11}$$

Let α be the ratio between $\|C_1C_2\|$ and $\|O_1O_2\|$, that is, $\alpha = \frac{\|C_1C_2\|}{\|O_1O_2\|}$. We use α as a scale parameter to modulate the global path $R(t)$. The path between C_1 and C_2 can be constructed as

$$\widetilde{R}(t) = \alpha R(t) - \alpha O(t) + C(t) \tag{12}$$

It is easy to verify that $\widetilde{R}(0) = C_1$ and $\widetilde{R}(1) = C_2$. Figure 8(a) shows an example of a global path. The automatically calculated path between a pair of sample points is shown in Figure 8(b).

One major advantage of this path control scheme is that we only need to parameterize the path once for all sample points. When following its path to a new destination, an agent translates and rotates to make its velocity vector align with the tangential direction of the path.

Results

We have developed a shape-constrained flock simulation system and run a series of flock simulations consisting of thousands of agents. All experiments are performed on an Intel Core2 Duo 2GHZ CPU with 2GB memory and an NVIDIA GeForce 8800 GTS graphics card. With this system, we can simulate shape-constrained flock animation in real time. For static shape constraints, simulations can achieve nearly 100 frames per second for thousands of agents. When simulating a flock of 1500 agents tracking dynamic objects, including rendering, the update rate can still reach 30 frames per second. We provide a clip of screen captured video as can be seen from the accompanied material.

We have run several simulations of flock transferring between two different shapes. In Figure 9, we have created a smooth transition of a shape-constrained flock from a cylinder to a sphere while keeping visual characteristics of flocking. Figure 10 is another example where a dog-shaped flock gradually forms into a whale.

In Figure 11, our approach is applied to a deforming waterfowl model. The target point for each individual agent changes its position at each frame. This example illustrates different flock behaviors under different moving speeds of the target shape. If the bird shakes his head slowly, flock members will follow their target points

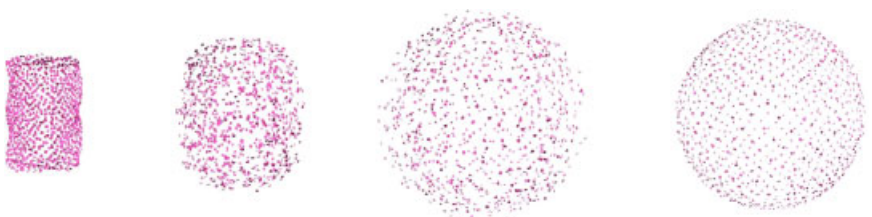


Figure 9. By one-to-one correspondence between sample points on two given meshes, our approach creates a smooth shape-constrained flock animation from a cylinder to a sphere.

DOI: 10.1002/cav

Figure 10. By adopting different constraints at source and destination, a dog-shaped flock changes into a whale.

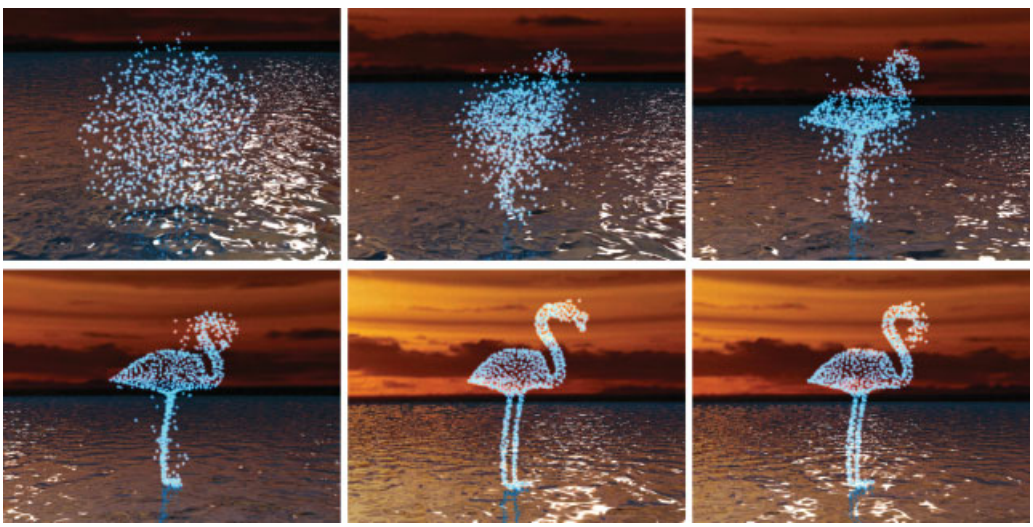


Figure 11. A flock transforms according to a dynamically changing waterfowl shape constraint.

strictly. If the virtual leaders move fast, the agents will disperse but still flock to the new positions. We can adjust the tracking parameters in Kalman filter to reflect how closely agents follow the virtual leaders. Figure 1 gives another illustration of dynamic tracking problem. The whole flock tries to follow the running horse while performing the flocking behaviors.

The pictures shown in Figures 1, 9–11 are rendered as a post-process. More specifically, we output the control data into scripts and read in by 3DS MAX for final rendering.

Conclusion and Discussions

In this paper, we propose a straightforward and practical method for simulating shape-constrained flock animation based on fuzzy control rules. Flock navigation is governed by arbitrary shape constraints while following a user-controlled global path. The animations created are capable of satisfying user-specified static or deforming shape constraints while keeping the characteristics of flock. Such effects are desirable in producing motion pictures.

DOI: 10.1002/cav

When the number of agents included in flock is less than 5000, the entire flock can be updated and rendered in real time in our implementation. However, the computation of flock behaviors increases rapidly when the number of agents goes up. We will seek to simplify the steering force computation. How to incorporate the powerful parallel calculation ability of GPU into our system is one of our future research topics.

ACKNOWLEDGEMENTS

This project is supported by the National Natural Science Foundation of China (Grant: 60573153, 60533080), the China 863 program (Grant: 2006AA01Z314), Key Technology R&D Program (Grant: 2007BAH 11B03), and the Program for New Century Excellent Talents in University (Grant: NCET-05-0519).

References

1. Reynolds C. Flocks, herds, and schools: a distributed behavior model. *Computer Graphics* 1987; **4**: 25–34.
2. Ulicny B, Ciechomski P, Thalmann D. Crowdbrush: interactive authoring of real-time crowd scenes. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2004; 243–252.
3. Reynolds C. Steering behaviors for autonomous characters. In *Proceedings of Game Developers Conference 1999*; 763–782.
4. Kalman R. A new approach to linear filtering and prediction problem. *Transactions of the ASME-Journal of Basic Engineering* 1960; **82**: 35–45.
5. Erra U, Chiara RD, Scarano V, Tatafiore M. Massive simulation using GPU of a distributed behavioral model of a flock with obstacle avoidance. In *Proceedings of the Vision, Modeling, and Visualization Conference*, 2004; 233–240.
6. Tu X, Terzopoulos D. Artificial fishes: physics, locomotion, perception, behavior. In *Proceedings of ACM SIGGRAPH'94*, 1994; 43–50.
7. Bayazit OB, Lien JM, Amato NM. Roadmap-based flocking for complex environments. In *10th Pacific Conference on Computer Graphics and Applications*, 2002; 104–115.
8. Bayazit OB, Lien JM, Amato NM. Swarming behavior using probabilistic roadmap techniques. *Lecture Notes in Computer Science* 2005; **3342**: 112–125.
9. Lien JM, Bayazit OB, Sowell R, Rodriguez S, Amato NM. Shepherding behaviors. In *Proceedings IEEE International Conference on Robotics and Automation*, 2004; 4159–4164.
10. Lien JM, Rodriguez S, Malric J, Amato NM. Shepherding behaviors with multiple shepherds. In *Proceedings IEEE International Conference on Robotics and Automation*, 2005; 3402–3407.
11. Saber RO. Flocking for multi-agent dynamic systems: algorithms and theory. *IEEE Transactions on Automatic Control* 2006; **51**: 401–420.
12. Hartman C, Beneš B. Autonomous boids. *Computer Animation and Virtual Worlds* 2006; **17**: 199–206.
13. Skrba L, Dobbyn S, McDonnell R, O'Sullivan C. Animating dolly: real-time herding and rendering of sheep. In *Eurographics Ireland Workshop*, 2006; 7–12.
14. Anderson M, McDaniel E, Chenney S. Constrained animation of flocks. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2003; 286–297.
15. Lai Y, Chenney S, Fan S. Group motion graphs. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2005; 281–290.
16. Wojtan C, Mucha P, Turk G. Keyframe control of complex particle systems using the adjoint method. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2006; 15–23.
17. Xiao D, Hubbold R. Navigation guided by artificial force fields. In *Proceedings of the SIGCHI Conference on Human Factors in Computing System*, 1998; 179–186.
18. Chenney S. Flow tiles. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2004; 233–242.
19. Sakuma T, Mukai T, Kuriyama S. Psychological model for animating crowded pedestrians: virtual humans and social agents. *Computer Animation and Virtual Worlds* 2005; **16**: 343–351.
20. Treuille A, Cooper S, Popović Z. Continuum crowds. *ACM Transactions on Graphics* 2006; **25**: 1160–1168.
21. Lai Y, Hu S, Martin R. Surface mosaics. *The Visual Computer* 2006; **22**: 604–661.
22. Nehab D, Shilane P. Stratified point sampling of 3D models. In *Proceedings of the 1st Eurographics Symposium on Point-Based Graphics*, Zurich, Switzerland, 2004; 49–56.
23. Turk G. Re-tiling polygonal surfaces. *Computer Graphics* 1992; **26**: 55–64.
24. Watt A, Watt M. In *Advanced Animation and Rendering Techniques*. ACM Press: New York, 1992.

Authors' biographies:

Jiayi Xu is a PhD student of the State Key Lab of CAD&CG, Zhejiang University, P. R. China. Her research interests include texture design, crowd and group animation, and general-purpose GPU computing. She received her BSc degree in computer science and technology from Zhejiang University.

DOI: 10.1002/cav

Xiaogang Jin is a professor of the State Key Lab of CAD&CG, Zhejiang University, P. R. China. He received his BSc degree in Computer Science in 1989, MSc and PhD degrees in Applied Mathematics in 1992 and 1995, all from Zhejiang University. His research interests include crowd and group animation, cloth animation, video abstraction, facial animation, implicit surface modeling, digital geometry processing, and texture synthesis.

Yizhou Yu received the PhD in computer science from University of California at Berkeley in 2000, and the MS degree in applied mathematics and the BS degree in computer science from Zhejiang University, China, in 1994 and 1992, respectively. He is currently an associate professor in the Department of Computer Science at University of Illinois at Urbana-Champaign. He has developed techniques in the areas of computer graphics and computer vision, including mesh editing based on differential coordinates, inverse global illumination, shape from shadow, feature-based texture synthesis, and controllable fluid simulation. He has authored or co-authored more than 50 research papers, and is on the editorial board of the Visual Computer. He is a recipient of the best paper award at 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, 2002 National Science Foundation CAREER Award and 1998 Microsoft Graduate Fellowship.

Tian Shen received her MSc degree and BSc degree in Computer Science in 2007 and 2005 from Zhejiang University. She then joined S3 Graphics Co., Ltd. as an engineer of the Verification Group in the Shanghai Architecture Design Department. Her research interests include flock animation and crowd animation.

Mingdong Zhou is an undergraduate student of the State Key Lab of CAD&CG, Zhejiang University, P. R. China. His research interests include computer animation, geometry modeling and mesh fusion.

DOI: 10.1002/cav