Crowd Control with Vector Painting

Shin Jin Kang
School of Games, Hongik University, Korea
Email: directx@hongik.ac.kr

Soo-Kyun Kim
Department of Game Engineering, Paichai University
1 14 Yeon-Ja 1Gil Seo-gu, Daejeon, Korea 302-735
Email: kimsk@pcu.ac.kr

Producing crowd simulations is a time-consuming task. The amount of technical work required for controlling relevant crowd parameters is a hurdle for effective production. In this paper we propose a new authoring method that can easily create crowd simulations with intuitive painting interface. For this, Self Organized Distribution Map (SODM), Stroke-based Flow Field and Gradient Vector Flow (GVF) techniques were applied to automatically generate parameters from the image. A hybrid fluid model for crowd dynamics was then applied to create smooth crowd movement. Through this system, a user can quickly create the intended crowd simulation with various movements. This methodology would enable fast crowd simulation prototyping in an intuitive way.

Keywords: *Crowd Simulation, Gradient Vector Flow*

ACM Classification: *I.3.4 (Computer Graphics Utilities)*

1. Introduction

In recent years, crowd simulation has expanded from a research topic to the commercial production of animation and interactive games. To simulate the movement of a crowd realistically, many parameters need to be selected and controlled efficiently. But as various factors influence crowd simulation in complex ways, it requires great effort to do this well. Here, if the crowd parameters are not appropriately modelled or if the control method of reflecting this is not appropriate, the cost of crowd simulation will significantly increase.

Crowd control is the research area of controlling crowds with a few parameters with an intuitive interaction. Chenney (2004) suggested a stationary flow field for crowds by manually tiling small rectangular regions of velocity fields. Ulicny *et al* (2004) proposed a brush interface with which a user can specify parameters for individual agents in a crowd by painting them. Several groups (Lee *et al*, 2007); Lerner *et al*, 2007)) proposed methods for making a crowd animation based on movies of real crowds. Their agents' behaviour is based on the computed vector field from vision data that require sample movies of pedestrians. Kwon *et al* (2008) presented an approach to edit group motion as a whole while maintaining its neighbourhood formation and individual moving

* Corresponding author: Professor Soo-Kyun Kim

Manuscript received: 14 August 2012
Communicating Editor: Seok-Hun Kim

trajectories. Takahashi *et al* (2009) presented a spectral-based approach to smoothly transform a source group formation into a target formation while respecting the clusters. Recently, more intuitive methods of controlling crowds have been proposed. Jin *et al* (2008) proposed the vector-field-based authoring method, which is generated by locating anchor points. Oshita and Ogiwara (2009) presented a method that determines movement-related parameters based on the example paths given by the user.

Previous authoring methods have focused on controlling crowds in a specific system framework. In these systems, the user needs to be trained to have knowledge of the crowd and be accustomed to the specific interface that is normally based on 3D development environments. It would be technically burdensome for artists and designers, who typically have weak technical backgrounds but have to see the crowd movement quickly for their own interests.

This paper introduces a new crowd control technique by reducing data input production cost. The parameters necessary for our crowd simulation are generated by the system with reference to simple image data, and the system converts these data into a smooth crowd movement. Compared with previous research, our method gives more path control freedom to the user with image editing than Chenney's method, which is limited to predefined tiles. Moreover, more organized movements are possible than with Jin's and Oshita's methods by supporting the shape management of crowds. We believe our method may be useful for simulation in transportation and geographic information field (Predic *et al*, 2010; Hu and Ge, 2009).

Many types of crowd dynamics models have been proposed since Helbing and Molnar's work (1995). One of them, the continuous flow model, has emerged from fluid dynamics. In this approach, a crowd is viewed as a dense particle model of a continuous fluid. Hughes (2003) used fluid dynamics to describe crowd locomotion, producing large-scale effects. Treuille *et al* (2006) extended this work to simulations driven by dynamic potential fields. For smoothing crowd movements with vector field, we additionally adopted this fluid dynamics model for our simulation. We adopted the smoothed particle hydrodynamics (SPH) (Benz, 1990) framework to generate small-scale details, while large flows of crowds can be handled efficiently by a grid-based method that is combined with discretized parameters from the image data.

2. Image Marks with Vector Painting

The proposed system uses vector painting interface for crowd control. If user paints control marks on ordinary bitmap image files, the system converts it to vector field for crowd movement. The proposed system requires three control marks from the image: (1) a shape mark to recognize the shape of the crowd, (2) a stroke mark to recognize the direction of the crowd's movement, and (3) background mark to recognize obstacles. The image can be modified easily by the user without special drawing techniques. An intuitive drawing interface for data input can reduce the time cost for training the user for simulation production. Figure 1 shows the simple input bitmap image that is edited by the user and its simulation results. Figure 2 shows overall process of proposed system.

3.1 Shape Mark

The shape mark defines the shape of the crowd at a certain time. We regard shape as the collection of individuals' positions. If the user draws the shape of the crowd in sequence, the system discretizes it automatically and arranges individuals' positions within the corresponding area. These positions become waypoints in the simulation phase. If the positions are well distributed

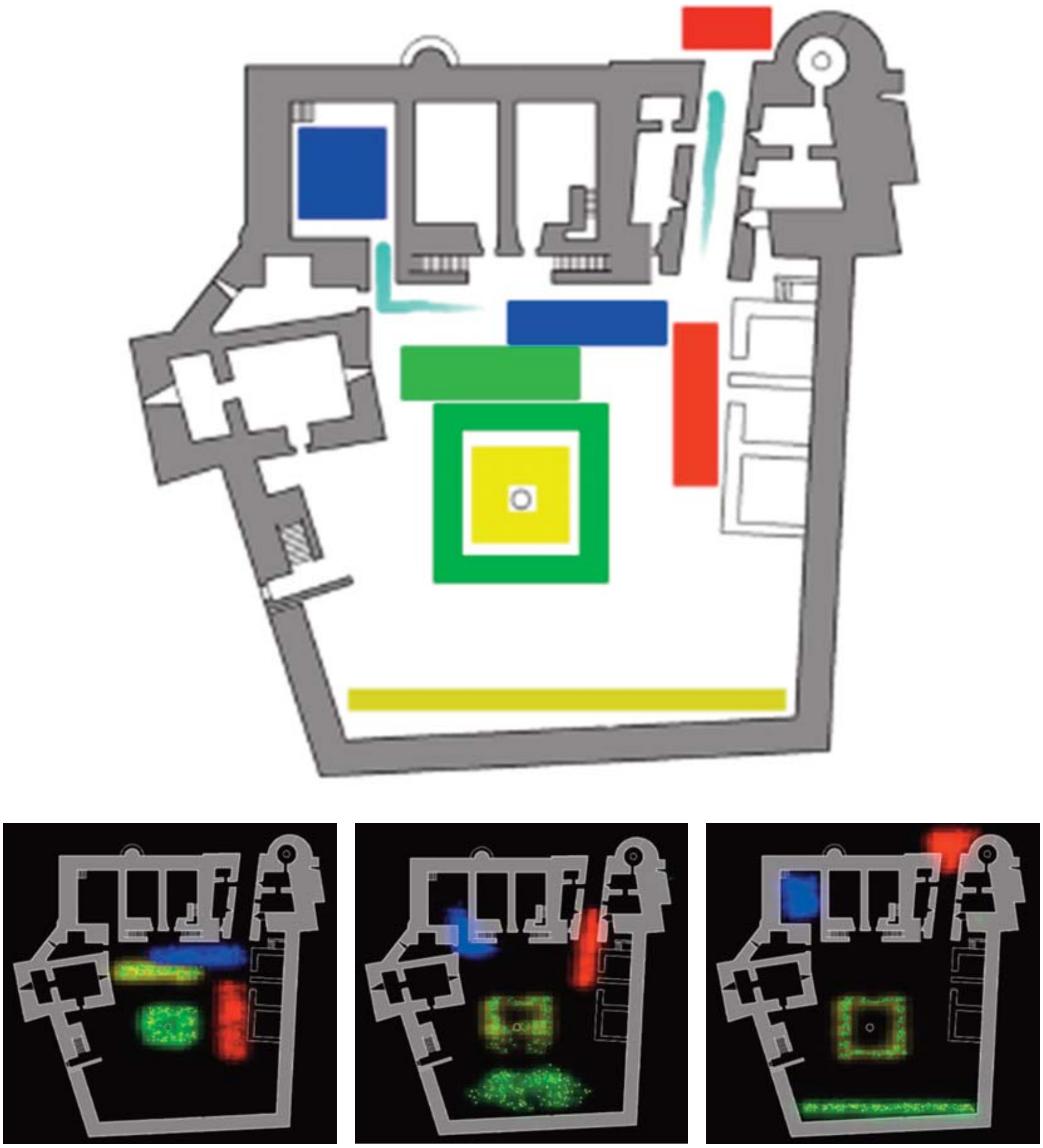
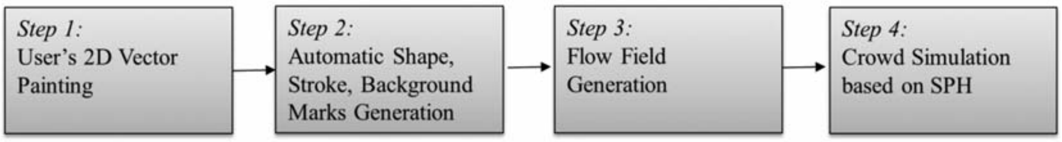


Figure 1: The input image with painting (above) and simulation steps (below)


Step 1:
User's 2D Vector Painting
Step 2:
Automatic Shape, Stroke, Background Marks Generation
Step 3:
Flow Field Generation
Step 4:
Crowd Simulation based on SPH
Figure 2: System Process

inside the shape area, the whole shape at a specific time can be expressed more correctly. To achieve this, we propose a new method for automatic position generation.

3.1.1 Position Generation

Our system supports automatic assignment of an individual's position within the defined shape. We generate these positions by satisfying the following requirements:

1. The location of the individuals is automatically assigned within the user-designated shape;
2. The adjacency among the allocated positions is maintained as much as possible;
3. The arranged position should have distance intervals.

If the first condition is not satisfied, much manual work will be needed to set the positions. If the second condition is not satisfied, there would be much path-crossing during the movement of crowds. The third condition reflects the instinct of the individuals that tend to secure a certain area. We created the Self-Organized Map with a Distance (SODM) method to satisfy all three conditions.

3.1.2 Self Organized Map (SOM)

Our SODM method for automatic positioning is an extension of the SOM method (Kohonen and Honkela, 2007). An SOM is an artificial neural network that is trained using unsupervised learning to produce a discretized representation of the input data on the training samples, which is called a *map*. A self organizing map is different from other artificial neural networks in the sense that it uses a neighbourhood function to preserve the topological properties of the input data. When a training example is fed to the network, its Euclidean distance to all the weight vectors is computed. The neuron with the weight vector that is most similar to the input is called the best matching unit (BMU). The weights of the BMU and the neurons that are close to it in the SOM lattice are adjusted towards the input vector.

3.1.3 Self-Organized Map with a Distance (SODM)

SOM has advantages in topology preservation but it does not guarantee the shape fitting and minimum distance maintenance of neurons. To satisfy the conditions in section 3.1.1, we redefined the updated formula of SOM. We added two external forces: *a shape fitting force*, $v_a(t)$, and a *distance maintenance force*, $v_d(t)$, defined as follows:

$$w(t+1) = w(t) + \theta(v,t)\eta(t)(D(t) - w(t)) + v_a(t) + v_d(t) \tag{1}$$

$$v_a(t) = \alpha(t)\theta_a(v,t)\eta_a(t)(N(t) - w(t)) \tag{2}$$

where $\alpha(t) = \begin{cases} 0 & \textit{if(w(t) is located in painted area)} \\ 1 & \textit{otherwise} \end{cases}$

Here $w(t)$ is the weight vector of a neuron at time t, $D(t)$ is the input vector, $\theta(v,t)$ is the neighborhood function, and $\eta(t)$ is the learning rate that decreases linearly with time. The neighbourhood function, $\theta(v,t)$, depends on the lattice distance between the BMU and the neuron v. Normally, Gaussian functions are used for this. We used radial basis function motivated by Zhu *et al*'s works (2009). The shape fitting force $v_a(t)$ forces the map to fit the shape and the distance maintenance force, $v_d(t)$ forces the map to maintain its internal distance. The shape fitting force

$v_a(t)$ moves the outside neurons towards the nearest point $N(t)$ inside of the shape object area. $\alpha(t)$ is the coefficient for being contained in the shape. $\theta_a(v,t)$ and $\eta_a(t)$ are the neighbourhood function and the learning rate for the area-fitting force equation, respectively. $\theta_a(v,t)$ increases linearly, unlike $\theta(v,t)$. Because of this, it has a stronger influence when the SOM update enters the local estimation phase after completing the global estimation phase. $v_a(t)$ is the distance maintenance force that guarantees the minimum area between trained neurons.

Figure 3 shows the comparisons with SODM results when 225 neurons were trained over 3,000 iterations. Figure 3 shows their progress steps. In these figures, we can see that the agent positions

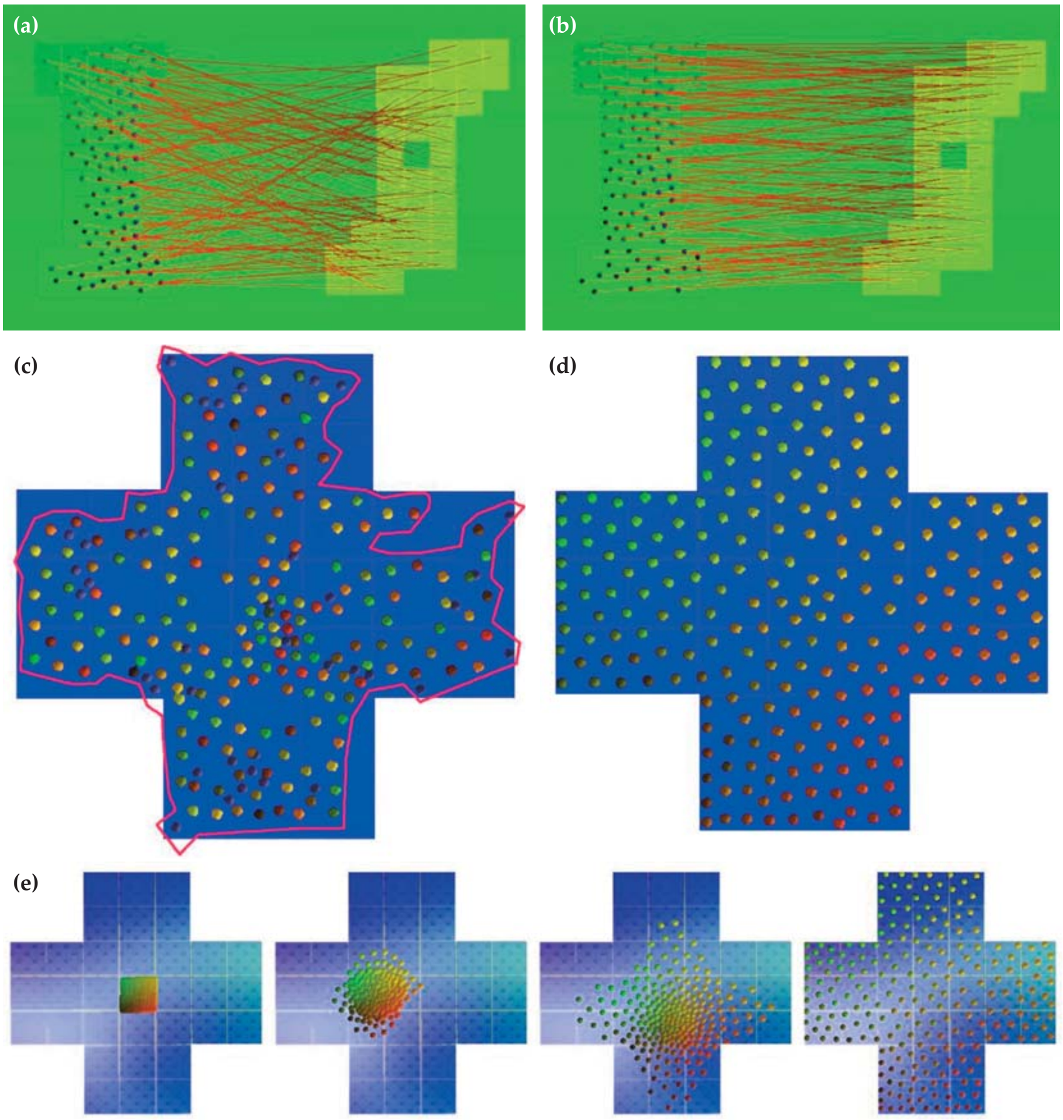


Figure 3: Comparison results. (a) Transition without SODM (b) Transition with SODM (c) Random distribution (d) SODM and (e) SODM progress steps

have colour adjacency, minimum separation and fully fit the defined shape. With the help of our automatic arrangement of individuals' positions, the user does not need to designate every position individually. This can reduce the effort of controlling the shape and positions of the crowd. It makes the system more scalable in terms of the number of individuals in the crowd being simulated.

3.2 Stroke Mark

To control crowd movement precisely, path information is necessary. The user may assign the path individually to each agent, but this will be burdensome for a large-scale crowd. To make the whole crowd move in a specific direction with minimum effort, we used the *stroke-based flow field*, which has direction and radius information. Our system generated this flow field by analyzing the stroke mark that is directly drawn by the user.

3.2.1 Stroke Direction

Stroke is a basic painting technique. With this, the user may easily express certain shapes and direction. It is an intuitive and natural interface for the user, but the system has difficulties in analyzing it. Because a bitmap consists of scalar values, directional and sequential information cannot be acquired directly. To solve this problem, we used a gradient in the brightness value of pixels. We consider that a pixel with high brightness is closer to the tail of the stroke. It is also closer to the starting point of the path. This can be naturally matched with the touch of the user's stroke, as it is identical with the direction of the stroke that is intuitively painted with the tablet or mouse.

With this idea, we tried to find out the best gradient pixels. We assumed that the central point of the brush is most identical to the direction that the user wants. So we extract these pixels by following three phases: (1) discretization, (2) binarization and (3) skeletonization. For the skeletonization algorithm, Zhang and Suen's algorithm (Zhang and Suen, 1998) was used. The points on the created central line were saved in a list data structure and sorted by brightness value. The system sampled the feature points and calculated the angles between adjacent points to generate the directional flow field around the brush.

3.2.2 Stroke Area

After obtaining the main direction vector, we need to determine how to interpret the rest of the brushstroke. We assume that the user wants to apply a flow field in proportion to the width of the brush. To approximate the width of the stroke, we processed the image in the following five steps: (1) extraction of the stroke's edge; (2) organization of the set of points R, of which the edge consists; (3) calculation of the minimum distance to the points R from the central line; (4) allocation of the minimum distance to the central line; (5) use of the minimum distance as the approximation width value of the stroke.

3.2.3 Generation of Flow Field

Equation 3 was used to generate the grid-based flow field. Here v_{xy} is the vector in every grid in the flow field, r_i is the set of sorted points in the central line by brightness, and the approximated width of stroke is h. The centre point of the grid in the flow field is G_{xy}. The scale of flow field is calculated by a radial basis function (RBF) ϕ. Figure 4 shows the process of generation of the flow field. And Figure 5 shows the example of flow field.

$$v_{xy} = \sum_{i=0}^{N-1}\left((r_{i+1} - r_i)\phi(\|G_{xy} - r_i\|, h)\right) \tag{3}$$

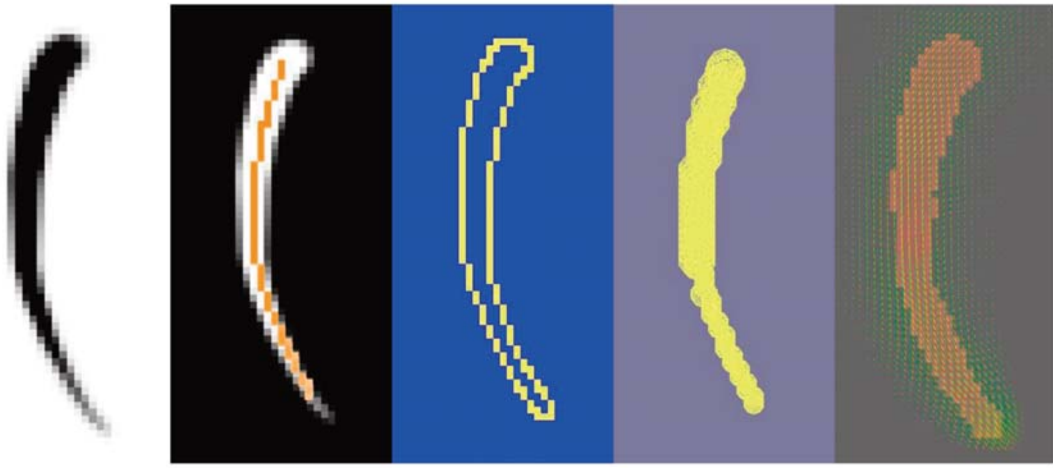


Figure 4: The process of flow field generation. (a) Stroke drawn by user (b) Skeletonization result (c) Outline detection (d) Width map (e) Generated flow field

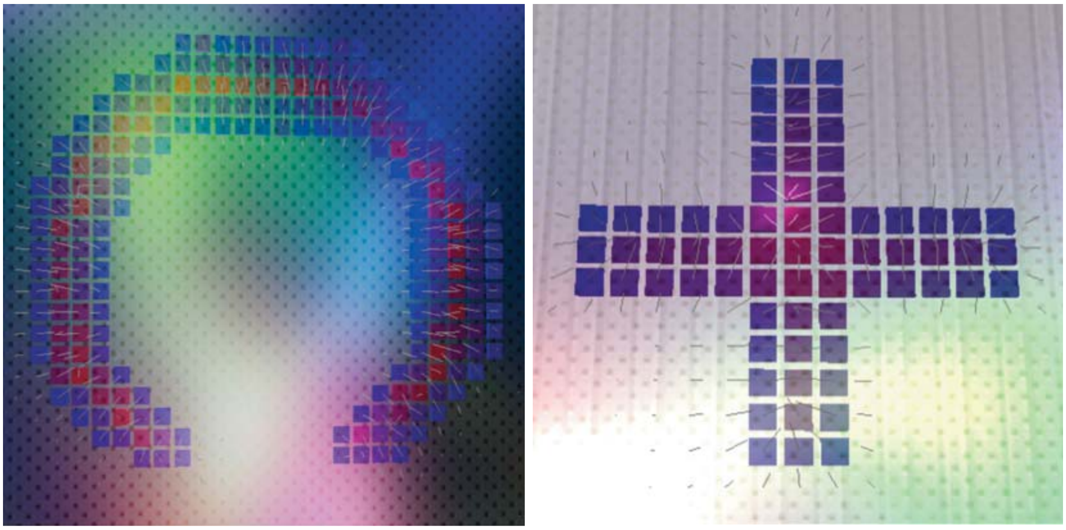


Figure 5: Generated flow field example

3.3 Background Mark

Crowd simulation can be applied in various research domains, such as transportation engineering, architectural engineering and emergency planning. Usually, in these kinds of domain, structural obstacle information is an important factor. These obstacles can be designated directly by the user. But if the obstacles are numerous or highly structural, much effort is required.

We attempt to use background information directly from image to reduce this workload. However, because of informational limitations of bitmap data, correct identification of objects is difficult. There should be some appropriate method for applying obstacle information to the image for input into the crowd system. We approximate background objects by adopting image processing methods and applying the *gradient vector flow* (GVF) technique to convert a scale value in the image to a vector field for simulation.

3.3.1 Gradient Vector Flow (GVF) Field

GVF is a contour detection technique based on the snake technique (Xu and Prince, 1997). GVF is computed as a spatial diffusion of the gradient of an edge map derived from the image. This computation causes diffuse forces to exist far from the object, and crisp force vectors near the edges. Combining these forces with the usual internal forces yields a powerful contour detection of the object.

In this paper, we used these internal and outer gradient force fields generated from GVF snake. Because the contour is optimized by GVF snake, the gradient is also turned into a contour. We assume that a background object is in grayscale in the input image. We applied a median filter to reduce the dust and converted this into an edge map for GVF snake iteration. After we have optimized the internal and outer gradient field generated by GVF snake, we negate the gradient field to have a repulsive vector from the contour and apply it around the contour area. This repulsive vector field makes the individual move away from every contour in the input image. We have experimented with this field on several image examples and found that it is useful on images such as section plan diagrams and topographical maps that have precise boundary information. Additionally, the repulsive vector field also can be converted to an attractive vector field by negating the vector value. In that case, it will make a gathering area for the crowd.

With a repulsive GVF field, the user can make the crowd free from collision with obstacles. This reduces the burden of designating the obstacle information for crowd simulation. With an attractive GVF field, the user can gather a crowd into a specific line. It can be a supplementary method for automatic positioning of the shape object. Figure 6 shows the input image, edge map and GVF fields that are generated by GVF snake.

3. Crowd Locomotion

After the system analyzes the input image, the parameters are prepared for crowd simulation. With these data, we attempt to produce the smooth crowd flow. Because our system has the possibility of generating a high-density large crowd via the automatic positioning in a specific shape, we need a crowd simulation system that is appropriate for dense circumstances. We try to achieve this goal by adopting a hybrid fluid dynamics model with the analogy between fluid and crowd movement (Hughes, 2003).

For large crowd flow control, an Eulerian grid-based model was implemented. The whole crowd moves under two static forces generated from the stroke-based vector field and GVF field. An additional dynamic advection force based on the Euler equation is applied inside the stroke-based vector field to create a stronger diffusive movement. For small detailed interactions in the grid, we used the Lagrangian method. We adopted smoothed particle hydrodynamics (SPH) based method (Kang and Kim, 2011) for the avoidance, repulsive and relational movement in detail. As a result, in the simulation phase, the individuals set their waypoints automatically, created through SODM on every shape object. Then, they are affected by three static forces generated

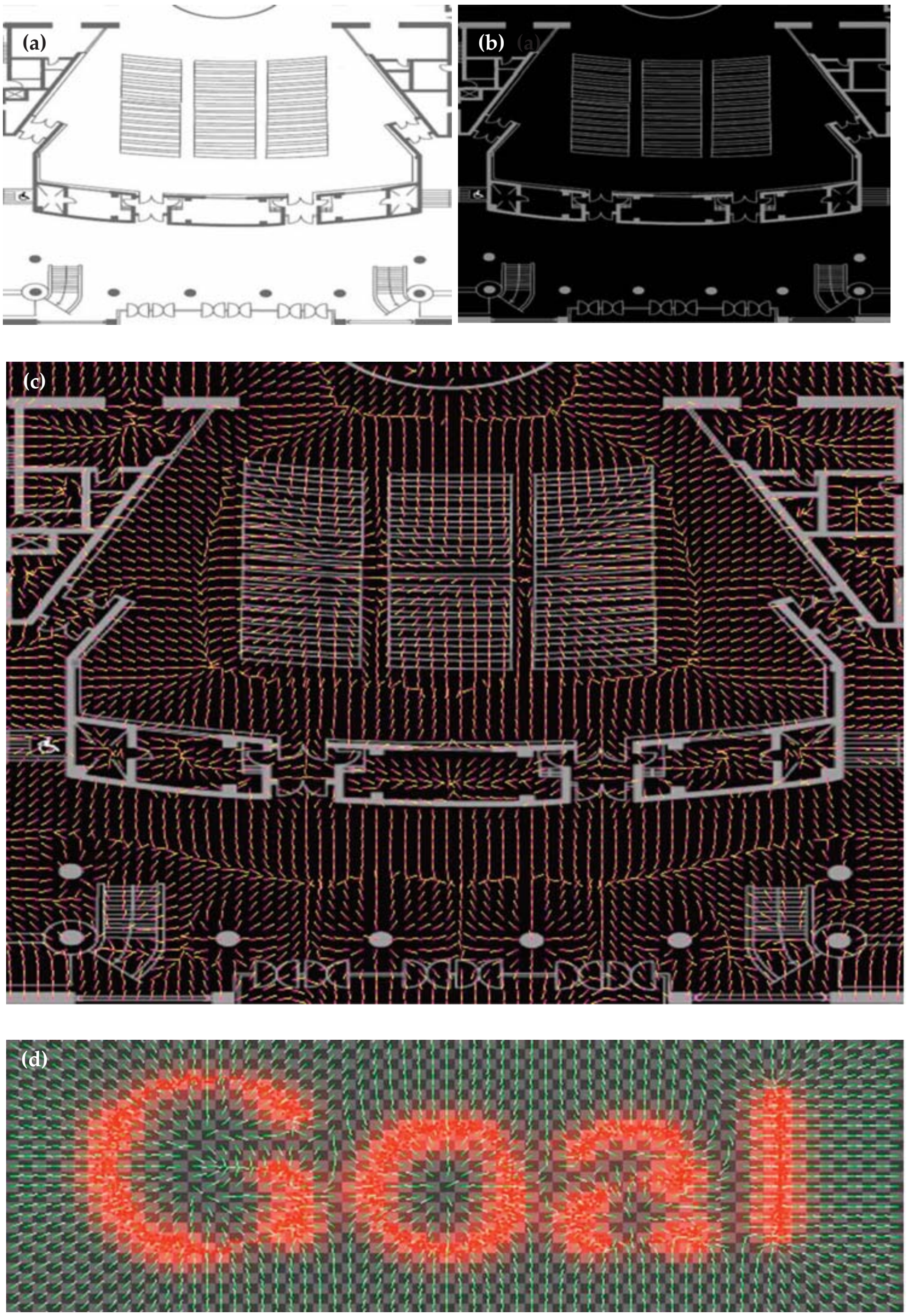


Figure 6: The input image and GVF fields (a) Original image (b) Preprocessed image (c) Repulsive GVF field (d) Attractive GVF field

from the image data for macroscopic movement and three dynamic forces for microscopic movement on the fly.

4.1 Advection Force

In addition to the basic crowd movements – avoidance, repulsive and relational movement, we added the crowd advection force based on the Euler equation in fluid dynamics. Let $u=u(x,t)$ denote the time-dependent vector field that specifies the velocity of location x on a cell at time t and its temporal derivative by u_t. The Euler equation that satisfies the incompressibility condition can then be written as Equation (4), where $p = p(x,t)$ is the hydrostatic pressure and f accounts for the external forces that affect fluid flow. This external force is used to control the fluid's flow. As for this force f, we used a driving force f^{drv}, which is obtained from the stroke-based flow field. This dynamic vector field u_i is applied to individuals and produces a more diffusive movement effect that matches the user's stroke. It produces an almost similar crowd shape with the stroke at specific time t.

$$u_t = -u \cdot \nabla u - \nabla p + f \tag{4}$$

5. Implementation

A number of simulations were performed on the proposed system, running on a 1.68GHz Intel Core2 Duo processor with an NVIDIA GeForce 8500 Mobile GT graphics card and 2GB memory on a notebook platform. This configuration can handle crowds with up to 1,000 animated polygon models, each constructed of 900 polygons, at 12 frames per second. The simulation was updated at each frame.

Figure 7 shows the sequential images of an exit simulation inside a theatre. Four hundred individuals separated into four groups and smoothly moved out through four different doors. They formed a jam around the door for a short time then passed out. This simulation could be produced with eight simple shape objects and four strokes on the theatre sectional diagram image. With the automatic generated repulsive GVF field, complicated manual work was unnecessary for the system to recognize the obstacles from the original sectional diagram. It took on average 4.5 minutes for single data preparation iteration. The simulation results show that two side doors make for more congestion compared with two front doors. Therefore, a user can easily suppose that human guide control might be necessary around that area.

Figure 8 shows the sequential images of a curve marching simulation with 1,000 individuals. This simulation could be produced with eight simple shape objects and four strokes on a blank bitmap image. It took on average 6 minutes for single data preparation iteration. We could easily control the path of crowds by editing strokes. With the combination of our stroke-based flow field with advection force, continuous smooth lane formation is maintained during the movement along the pavement.

Table 1 shows the summary of our simulation results. All simulations are tested with a 64×64 resolution image. Experimentally, the overall preproduction time rarely depends on the image editing time. There might be some individual differences in drawing skill, but the data input stage is inexpensive. The number of user drawn object shapes and strokes does not affect the overall preproduction time seriously because of the interface simplicity. In the automatic parameter generation stage, 90% is used for the SODM iteration. It took on average 10 seconds for distributions in the single shape object for 200 neurons for 1,000 iterations. The level of SODM

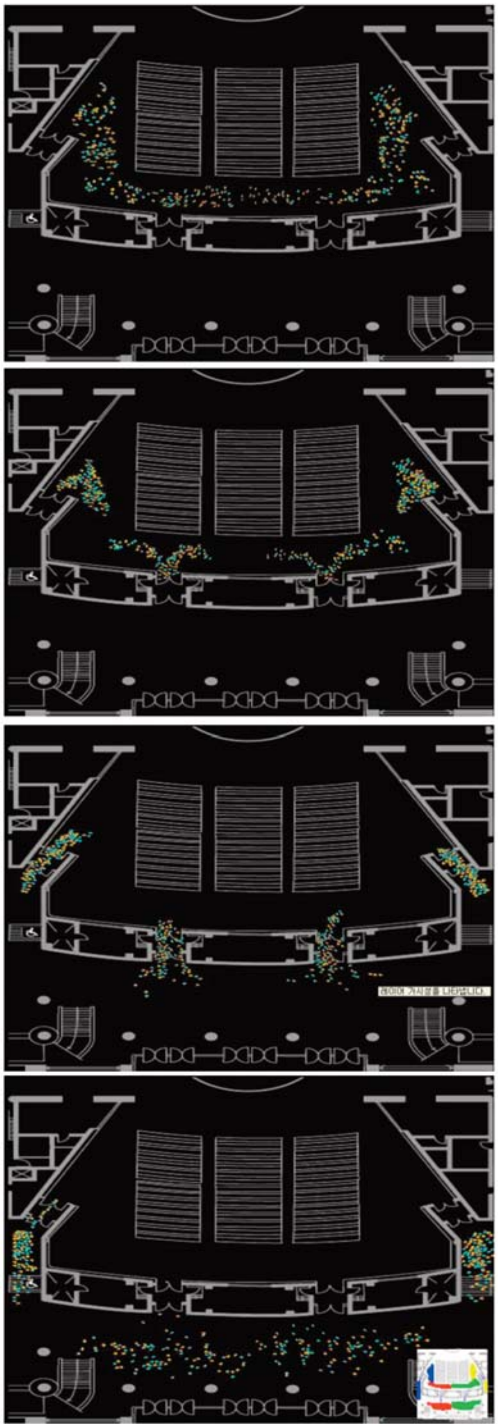


Figure 7: Exit simulation inside a theatre

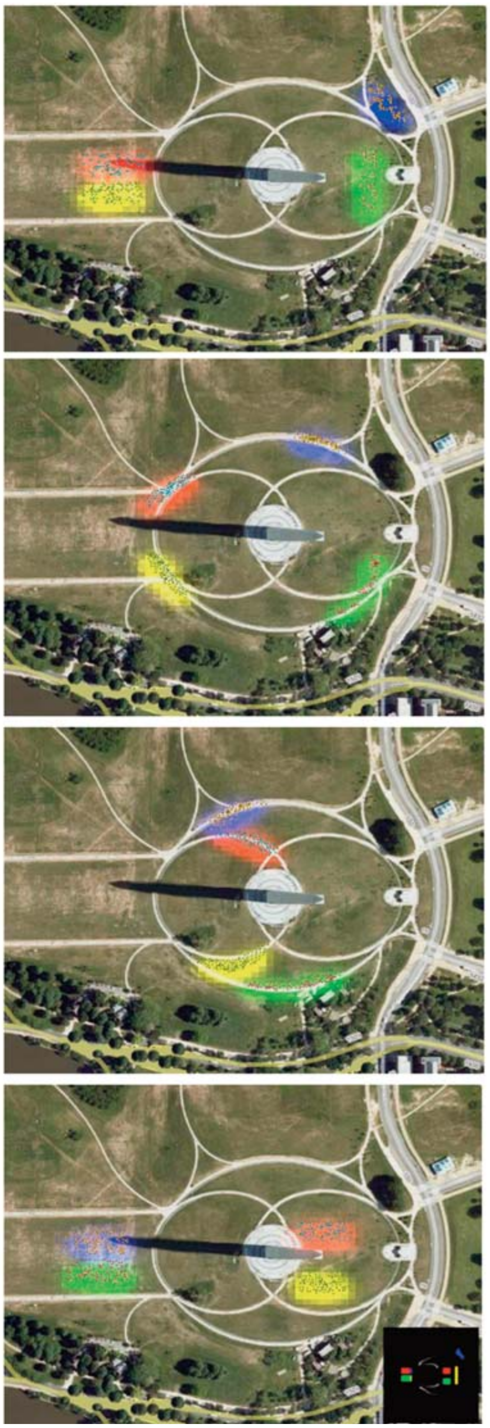


Figure 8: Curve marching simulation in the park

Example	# of Individuals	Figure	# of Object shapes	# of strokes	Image editing time(min)	Automatic parameter generation time (min)
Exit	400	7	8	4	1.0	3.5
Castle	600	2	8	4	1.5	5.0
Curve Marching	1,000	8	8	4	1.0	6.0

Table 1: Summary of simulation results

iteration can be adjusted depending on the requirement for topology preservation and shape fitting. The time for generating the GVF field and the stroke-based flow field from the image is relatively short and it can be negligible. In conclusion, in these examples, we could produce the smooth simulation results that involve the shape, path and obstacle factors in crowd simulations with simple image editing work. The number of individuals can be easily scaled without any additional manual work.

6. Conclusion

In this paper, we have shown that intuitive crowd simulation control is possible using an ordinary bitmap image that is drawn by the user. To do that, we suggested using SODM, stroke-based flow field generation, and GVF field to generate parameters for crowd simulation. Smooth crowd simulation was produced using a new hybrid crowd dynamics model based on fluid dynamics. Our system has the advantage in supporting the fast and easy interface for crowd control without losing the possibility to generate various crowd movements. Our sample indicates how our method can be applied to various domains of crowd simulations via few interactions from the users. It enables fast prototyping of crowd simulations that are related with image data such as a sectional diagram or a topological map. This method can be improved in various directions by combining image editing techniques.

References

BENZ, W. (1990): Smooth particle hydrodynamics: A review. *The Numerical Modeling of Nonlinear Stellar Pulsation*: 269–288.

CHENNEY, S. (2004): Flow tiles. *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 233–242.

HELBING, D. (2004): An improved fluid-dynamic model for vehicular traffic. *Physical Review* E.

HELBING, D. and MOLNAR, P. (1995): Social force model for pedestrian dynamics. *Physical Review* E 51: 4282–4286.

HU, Y.H. and GE, L. (2009): Learning ranking functions for geographic information retrieval using genetic programming. *Journal of Research and Practice in Information Technology*, 41(1): 39–52.

HUGHES, R.L. (2003): The flow of human crowds. *Annual Review of Fluid Mechanics* 35: 169–182.

JIN, X., XU, J., WANG, C.C.L., HUANG, S. and ZHANG, J. (2008): Interactive control of large crowd navigation in a virtual environment using a vector field. *IEEE Computer Graphics and Applications* 28: 37–46.

KANG, S.J. and KIM, S.K. (2011): Dense crowd simulation based on smoothed particle hydrodynamics. *Applied Mathematics & Information Sciences*, to appear.

KOHONEN, T. and HONKELA, T. (2007): Kohonen network. *Scholarpedia*.

KWON, T.S., LEE, K.H., LEE, J.H. and TAKAHASHI, S. (2008): Group motion editing. *Proceedings of the ACM Transactions on Graphics* 27(3): 80–90.

LEE, K.H., CHOI, M.G., HONG, O.Y., and LEE, J.H. (2007): Group behavior from video: A data-driven approach to crowd simulation. *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*: 109–118.

LERNER, A., CHRYSANTHOU, Y. and LISCHINSKI, D. (2007): Crowds by example. *Computer Graphics Forum* 26: 655–664.

MULLER, M., CHARYPAR, D. and GROSS, M. (2003): Particle-based fluid simulation for interactive applications. *Proceedings of the ACM Siggraph/Eurographics Symposium on Computer Animation*: 154–159.

OSHITA, M. and OGIWARA, Y. (2009): Sketch-based interface for crowd animation. *Lecture Notes in Computer Science* 5531: 253–262.

PREDIC, B., RANCIC, D. and MILOSAVLJEVIC, A. (2010): Impacts of applying automated vehicle location systems to public bus transport management. *Journal of Research and Practice in Information Technology*, 42(2): 79–98.

TAKAHASHI, S., YOSHIDA, K., KWON, T.S., LEE, K.H., and LEE, J.H. (2009): Spectral-based group formation control. *Computer Graphics Forum* 28(2): 135–141.

TREUILLE, A., COOPER, S. and POPOVIC, Z. (2006): Continuum crowds. *ACM Transactions on Graphics* 25.

ULICNY, B., CIECHOMSKI, P.H. and THALMANN, D. (2004): Crowdbrush: Interactive authoring of real-time crowd scenes. *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*: 243–252.

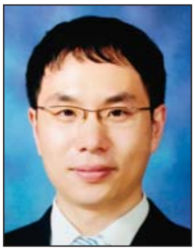
XU, C. and PRINCE, J.L. (1997): Gradient vector flow: A new external force for snakes. *IEEE Proceedings of the Conference on Computer Vision and Pattern Recognition*: 66–71.

ZHANG, T.Y. and SUEN, C.Y. (1998): A fast parallel algorithm for thinning digital patterns. *ACM Communications* 27(3): 236–239.

ZHU, X., ZHAO, J., DUANMU, C.J. and XU, H. (2009): Noisy motion-blurred images restoration based on RBFN. *Journal of Research and Practice in Information Technology*, 41(3): 195–208.

Biographical Notes

***Shin Jin Kang** received his PhD degree in computer science from Korea University in 2011. Since 2003, he has worked at Sony Computer Entertainment Korea and NCsoft as a lead game designer in various video games and MMORPGs including AION. He is now a professor at the School of Games at Hongik University. He is also the technical advisor of NCsoft.*


Shin Jin Kang

***Soo-Kyun Kim** received his PhD from the Computer Science & Engineering Department of Korea University, Seoul, Korea, in 2006. He worked at the Telecommunication R&D centre at Samsung Electronics Co., Ltd., from 2006 to 2008. He is now a professor in the Department of Game Engineering at Paichai University, Korea.*

Soo-Kyun Kim