Appear in Proceedings of IEEE International Conference on Cybernetics and Intelligent Systems, 2008.

Simulating Virtual Crowd with Fuzzy Logics and Motion Planning for Shape Template

Jen-Yao Chang
Computer Science Department
National Chengchi University
Taipei, Taiwan
g9314@cs.nccu.edu.tw

Tsai-Yen Li
Computer Science Department
National Chengchi University
Taipei, Taiwan
li@cs.nccu.edu.tw

***Abstract*—Simulation of crowd motions has great potential in many applications in robotics, games and animations. However, it is also a great challenge to be able to control the motion of a virtual crowd according to the intents of its designer such as making the crowd conform to a specific shape while avoiding collisions with other agents in the crowd or with obstacles. In this paper, we propose a simulation mechanism that works in two steps: global motion planner for a shape template and fuzzy controller for shape constraints. The system first uses a motion planer to generate a rough path for a desired shape that may be partially in collision with the environment. Then a fuzzy controller based on several criteria is used to move the agents in a group to conform to the desired shape. We will demonstrate the implemented system with several simulation examples that show the path of the shape template and how the crowd effectively conforms to the template.**

***Keywords*—Virtual Crowd Simulation, Motion Planning, Fuzzy Control for Crowd Motion, Compuer Animation**

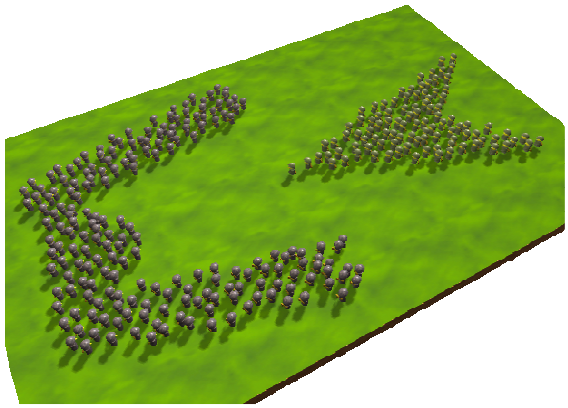

Figure 1. Example of shape formation for crowd simulation in a battle game

I. INTRODUCTION

Group formation is a problem that has been studied in robotics for long. In recent years, the problem of simulating the motion of a large crowd consisting of many agents has attracted much attention in the applications in the entertainment industry such as games and films. The characteristics of the problem defined for this type of applications differ from the robotic applications in several ways: the physical model of the agents can be simplified, the simulation can be centrally controlled, and the visual effect usually is the primary objective. Much previous work has successfully demonstrated the power of using virtual forces or similar techniques to create realistic emergent group behavior for a large crowd with consistent group behaviors. Some commercial simulation packages can also offer this function to animation designers. However, due to the nature of the simulation-based approaches, which usually lacks a global view and fine-grain control, it is still a great challenge to create crowd motions that can conform to a user-specified shape along a reasonable path for a crowd in an automatic fashion.

In many applications, the shape of a crowd may bear special meanings in the application context that the designer would like to convey. For example, forming an arrow shape for a group of soldiers, as shown in Fig. 1, could be useful in defeating their enemy. In addition, different shapes of crowds for congregated animals are often created in cartoon animations to create special visual effects. For example, a crowd of bees can form the shape of an elephant to express powerfulness visually. However, with the traditional tools for creating animations, authoring such a scene would be very tedious and time-consuming because the complex motion of the crowd needs to be specified by hand. Most current crowd simulation tools cannot easily enforce such a shape constraint either.

In this paper, we propose a novel approach to the problem of creating crowd motions with shape preference automatically according to a user-defined goal. The approach consists of two steps: first generating a rough path for the desired shape template and then control the crowd to move with fuzzy rules. Given the specification of goal configuration and how strictly the shape should be respected, the planner attempt to generate a path for the shape template that may not be all collision free. The template along the path will be used to guide the motion of the crowd with fuzzy rules of several types. The resulting crowd motion would be one that moves along a reasonable route that can keep the desired shape as much as possible.

In the next section, we will discuss the previous work pertaining to crowd motion simulation. In Section III, we will describe a modified version of Best-First planning algorithm that is used to generate the motion for the template. In Section IV,

978-1-4244-1674-5/08 /$25.00 ©2008 IEEE

we will present the classes of fuzzy rules that we have used to control the motion of the crowd such that they can perform desirable movement behaviors. In Section V, we will use simulation examples to illustrate the effectiveness of the planner and control system. Finally, we will conclude the paper with future directions.

II. Related Work

The topic of crowd simulation has attracted much attention in the field of computer animation in the past two decades. In his early work on simulating flocking behaviors for boids, Reynolds [12][13] proposed a virtual force model consisting of three types of steering forces: separation, cohesion, and alignment, which are used to drive the motion of each individual agent to create emergent behaviors. Despite the advantage of easy implementation, the resulting crowd motion is difficult to control by simply adjusting the weights of these virtual forces. Anderson used a similar force model and some shape constraints for the crowd to make the agents conform to the given shape in an incremental fashion [1]. However, the shape of the crowd needs to be determined before hand, and the time that the system takes to adjust the shape makes it difficult for online applications.

Computing paths for a single robot or a group of robots are classical motion planning problems in robotics [3]. However, in most traditional motion planner, the robots are considered as a rigid body or a system consisting of several rigid body parts that cannot be deformed at run time. Bayazit added some global information in the roadmap constructed for a given environment to facilitate more sophisticated group behaviors exploiting the knowledge of the environment [2]. Kamphuis proposed to generate crowd motions by planning the motion of some given shapes with the Probabilistic Roadmap Method (PRM)[7]. This method can overcome the problem of group shape control that is difficult to achieve by virtual forces. However, the bottleneck of this method is that limited shapes are allowed in the planning; thus, it limits the feasible paths that can be generated for more flexible shapes.

The methods of using fuzzy logics have been widely used in many applications including robot motion controls. It has the advantage of being able to incorporate uncertainties seamlessly into traditional control methods. Adaptive fuzzy logic controllers (FLC) were designed especially for controlling robots to tackle various kinds of uncertainties through linguistic presentation. In [14], Tunstel proposed an FLC for autonomous robots with three basic types of behaviors: goal-seek, route-follow, and localize. Some of these behaviors are also considered in this paper. Berman has also used a hierarchical behavior model with fuzzy rules to control many agents moving in a scene while avoiding collisions with each other [3]. Michaud proposed using fuzzy selection to select appropriate actions according to the environment the robot is situated in [10].

III. Motion Planning for Shape Template

Most early work on the problem of motion planning for flexible objects focused on existing physical objects with flexibility to some degree such as a plate or a string [11][7][4]. However, no much work has been done for objects that can be deformed to an arbitrary shape probably due to the difficulty of parameterizing this kind of objects. In our previous work [5], we have attempted to design a motion planner for a reshapeable object that can deform to pass narrow passages in the environment to reach a goal while keeping the shape of circle as much as possible. However, due to the complexity of maintaining the configuration of a flexible shape, the performance of the planner starts to degrade when the dimension of orientation is introduced for an asymmetric shape.

Algorithm: Template_BFP(T_{init} , T_{goal})

Input: Initial and goal template configurations T_{init}, T_{goal}, obstacle bitmap *OB*.

Output: a feasible path *P*.

1. Initialize a priority queue *Q* sorted according to the objective function *F*, and a configuration space C.
2. *SUCCESS* = **false**;
3. Insert T_{init} into *Q*
4. **while** *Q* is not Empty and *SUCCESS* = **false**
5. **begin**
6. T_c = Dequeue(*Q*)
7. **for** all neighbor *T'* of T_c
8. **if** *T'* is collision-free in *OB* and not visited in *C* **then**
9. Insert *T'* into *Q*
10. Mark *T'* as visited in *C*
11. **if** *T'* = T_{goal} **then**
12. *SUCCESS* = **true**
13. **end**
14. **if** *SUCCESS* = **true then**
15. **return** the constructed path by tracing the template configuration from T_{goal} back to T_{init}
16. **else return failure**

Figure 2. The Template_BFP algorithm

Instead of coping with detail shape changes in global motion planning, in this paper we have adopted a two-level planning approach: generating a path for the desired shape template with minimal collisions and then deforming the shape according to the configurations of the template along the path in a postprocessing step. We will describe these two steps in the following subsections.

A. Planning for Shape Template

We assume that, as in a traditional path planning problem, we are given a geometric description of the 2D shape that we prefer the crowd to form and its initial and goal configurations, which are in a 3D configuration space (C-space, (x, y, θ)). The objective of the planner in this stage is to generate a continuous path in the template C-space such that the center of the shape, defined as part of the geometry, remains collision-free along this path. In addition to this hard constraint, some soft constraints, acting as preferences, can be specified and used as the search criteria. For example, the length of the path and the degree of overlapping with the obstacles in the environment are two criteria that we have used in our planner. The degree of overlapping is defined as the areas (or number of cells) in the

Algorithm: Deform_Shape (T, C)

Input: Template configuration T, parent shape configuration C, Obstacle Bitmap OB

Output: New configuration C'

Define: G is a cell in C, and Dist(G) is the distance to Template at T

1. Clone C' from C
2. Q_{src} and Q_{dst} are queues sorted according Dist(G).
3. **for each** G_i of $C_{Boundary}$ in C'
4. Insert G_i into Q_{src}
5. **for each** G_i' neighbor of G_i
6. **if** G_i' is collision-free in OB **then**
7. Insert G_i' into Q_{dst}
10. nb_moves = 0
11. **while** Q_{src} is not **nil** and Q_{dst} is not **nil**
12. **begin**
13. S_c = Dequeue(Q_{src})
14. D_c = Dequeue(Q_{dst})
15. **if** Dist(D_c) > Dist(S_c) **then break**
16. remove S_c in C'
17. add D_c in C'
18. nb_moves ++;
19. **end**
20. **if** nb_moves = 0 **then return nil**
21. **return** C'

Figure 3. The DeformShape procedure used to deform the shape according to a template configuration

workspace where the template and the obstacles in the environment overlap.

The path planning algorithm is a modified Best-First Planning algorithm that is commonly used in solving the motion planning problems in lower dimensional C-spaces [3]. We assume that the workspace and the C-space are all represented as discrete space of uniform grids. The algorithm, called Template_BFP, is outlined in Fig. 2. The main differences from traditional BFP planners are in line 6 and line 8. In line 8, a configuration is considered legal if the center of the template is not inside an obstacle. In line 6, the priority in a queue is defined according to a function F, which is composed of two components: *shape* cost and *length* cost. The shape cost for a template configuration is calculated based on the number of cells overlapping with the obstacle regions. The length cost is computed based on the Euclidean distance of the center along the path in the workspace. Since the two costs are computed based on different metrics, we have to normalize them before they are linearly composed according to user-specified weights. The path generated by the algorithm is then smoothed according to the same criteria to obtain a less jerky and more natural movement for the shape template.

B. Deforming Shape According to Environment Constraints

Once the path for the shape template is generated, we will use it to compute the real shape, possibly deformed, in every step of the path for the crowd according to the constraints of the environment. Since the agents in a crowd cannot move instantaneously, the shape of a crowd cannot be changed without a limit. Without losing generality, we further assume that the agents can move one cell at most for a given time step. Consequently, the shape deformation can be realized by focusing on the boundary cells only.

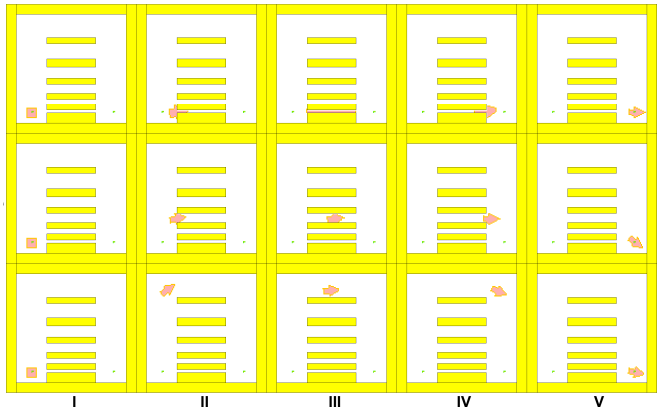

Figure 4. Snapshots of the deformed arrow shape in the generated paths with different search criteria

The algorithm for shape deformation is shown in Fig. 3. The procedure takes the shape configuration from the previous step, the current template configuration, and the obstacle bitmap as inputs. A shape configuration consists of a list of cells constituting the deformed shape in the free workspace. The output of this procedure is the shape configuration for the current step. What the procedure does is moving the cells in the boundary region of the shape forward to the unoccupied region of the template at the new configuration. We keep two priority queues for the source and destination sets of cells for the movement and attempt to move cells one by one from the source to the destination region until one of the queues becomes empty. The order of movement priority is computed based on the distance to desired template. That is, the cells that are farther away will have higher priority to be relocated to the empty region in the new template configuration. For each template configuration along the path, we may call the procedure multiple times until the shape cannot be further deformed to yield a better score on the distances to the template. In other words, the path of the template is parameterized by time according to the speed limit of the cells in the shape to allow large deformation to occur.

C. Example

We have implemented the planner with the Java language. Snapshots of the deformed shape along the paths generated with three different criteria are shown in Fig. 4. The initial configuration for the template is at the left corner while the goal is at right corner. The initial shape is a rectangle, and the desired shape is an arrow. Along the path in each case, the arrow shape is formed gradually and kept as long as the environment is allowed. In this example, we have designed an environment consisting of several passages with various widths. In the three cases shown in Fig. 4, we have used different weights on the costs of shape and path length. The weights (w_{shape}, w_{length}) for the two costs in the three examples are (0.0, 1.0), (0.5, 0.5), and

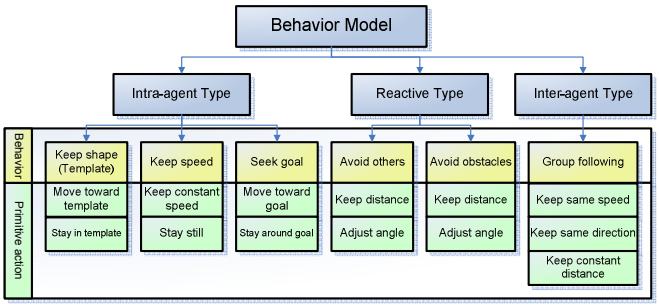

Figure 5. Hierarchy of behavior model used in designing fuzzy rules

(1.0, 0.0), respectively. For example, in case (a), the shape is deformed to go through the lowest narrow passage to yield the shortest path. The planning times for generating the template paths are 516, 703, and 1812ms, respectively, and the post-processing times for deforming the shape are 422, 1312, and 19,125ms, respectively, on a P4 1.6GHz machine. The case in (c) takes the longest time since a longest path is generated as a result of searching a larger portion of the configuration space.

IV. Using Fuzzy Logics to Control Crowd Motions

Although the path for the desired shape deformed according to the environment has been generated as described in the previous section, the final crowd motion may not exactly match the deformed shape. Instead, the deformed shape is used as a reference to guide the simulation of the crowd motion. In this section, we will describe how we use the mechanism of fuzzy control to design appropriate fuzzy rules to control the motions of the agents in a crowd to form a desired shape.

A. *Fuzzy Behavior Model*

According to our observation of the interaction between people in our real life, we have modeled the behaviors of the agents in a crowd and the primitive actions associated with these behaviors. We classify the behaviors into three types: *Intra-agent*, *reactive*, and *inter-agent.* Each category of behaviors contains several fuzzy behaviors, and each behavior model contains primitive actions, as depicted in Fig. 5. In the Intra-agent category, we consider the behaviors that are based on an individual agent's intents regardless of other agents and the environment. These behaviors include keeping a constant speed, seeking the goal, and attempting to move within a given shape template. The second category concerns reactive behaviors such as avoiding collisions with other agents or the obstacles in the environment. The third inter-agent category is about how to maintain collective motions as a group with emergent behaviors, which is similar to the virtual force model proposed by Reynolds[12].

B. *Design of Fuzzy System*

We have used the systems described in [3][15] as references to design our fuzzy control system. We use linguistic variables and fuzzy rules to describe the relation between sensation and actions. The linguistic variables and number of linguistic terms associated with each primitive action are listed in Table I. Examples of linguistic terms for speed and orientation related linguistic variables are shown in Fig. 6. Each primitive action is represented by a distinct control policy governed by the fuzzy inference of General Modus Ponens (GMP). We have

TABLE I. Lanaguage variables and number of Linguistic items for Each Fuzzy Behaviors

Fuzzy Behavior	Primitive Action	Linguistic Variables	Linguistic Terms
Keep speed	Keep constant speed & Stay still	Current_Speed	7
		Expected_Acceleration	9
Seek goal	Move toward goal	FacingAngle_Difference	9
		Expected _TurningAngle	9
	Stay around goal	Distance_To_Goal	3
		Expected_Acceleration	9
Keep shape	Move toward template & Stay in template	FacingAngle_Difference	9
		Expected_TurningAngle	9
Avoid others	Keep distance	Distance_To_Agent	3
		Expected_Acceleration	9
	Adjust angle	Relative_Angle	9
		Avoid_Agent_Angle	9
Avoid obstacles	Keep distance	Distance_To_Obstacle	3
		Obstacle_Force	3
	Adjust angle	Left_Detection	3
		Front_Detection	3
		Right_Detection	3
		Obstacle_TurningAngle	9
Group forming	Keep same speed	Relative_Speed	5
		Expected_Acceleration	9
	Keep same direction	Relative_Angle	9
		Expected_TurningAngle	9
	Keep constant distance	Relative_Distance	3

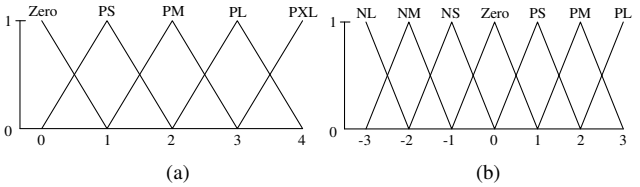

Figure 6. Examples of linguistic terms for (a) speed related variables and (b) orientation related vartiables in the membership functions

designed a fuzzy knowledge base (FKB) to store the fuzzy rules and used a fuzzy logic controller (FLC) to control the motions of the agents. In each control loop, the system fuzzifies the position, speed, and orientation of the agents and environmental information such as goal position, surrounding agents, and obstacle configurations for the processing of FKB. Then it defuzzifies the results to obtain the control parameters that drive the motions of the agents. In this work, we have used the method of center of gravity to defuzzify the result. Before sending the result for execution, all of the generated control commands will be filtered through a physical module to ensure that the physical constraints such as maximal acceleration and velocity are not violated.

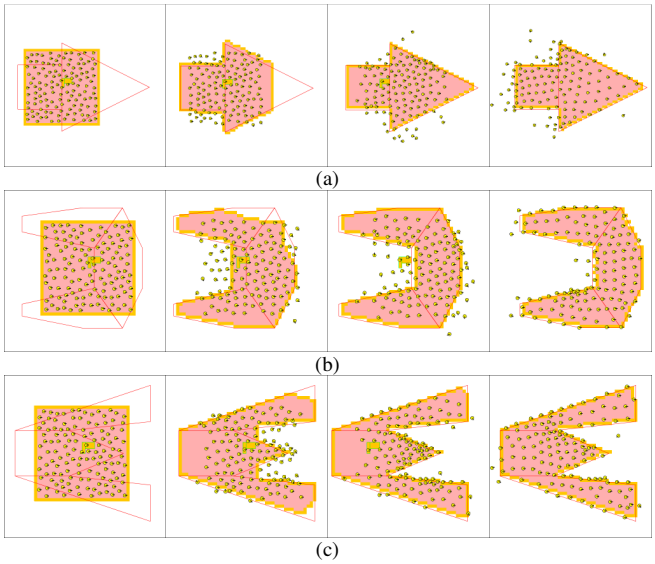

Figure 7. Examples of crowd motions conforming to a given shape

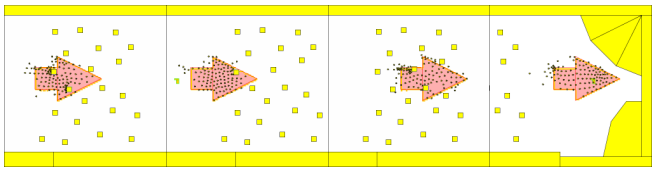

Figure 8. Example of crowd motion moving through scattered obstacles while maintaining an arrow shape

Figure 9. Snapshots of crowd simulation in 3D for the example in Fig. 8

V. Experimental Results

A. Implementation

We have implemented the crowd simulation system described in the previous sections with the Java language. The system contains a graphical user interface allowing the user to design the environment, specify the desired shape, and position the initial and goal configurations for the shape template. After the problem is specified, the system will first generate a path for the shape template to reach the goal and then deform the shape along the path according to the obstacles in the environment. This transforming shape will then be used in the keep-shape behavior (see Section IV) to constrain the motion of the crowd.

B. Simulation Examples

We have used several template shapes to test the effectiveness of the fuzzy control system as shown in Fig. 7. The snapshots in each sub-figures are taken in the formation of the desired shapes (arrow-shape, u-shape, w-shape) from the initial square shape. Note that the red hollow shapes are the desired shape templates specified by the user while the pink solid regions are the collections of cells deforming to fit the shape template. Since there are no obstacles in this case, the regions will fit the template eventually. The dots in each figure are the agents composing the crowd (128 dots in this case). Although not all of them can retain in the desired shapes during the deformation process, they all intend to move into the designated shapes under the keep-shape fuzzy rules described in the previous section.

We have also used an environment scattered with many small obstacles, as shown in Fig. 8, to test the robustness of the fuzzy control system on environment variables. Since the obstacles are relatively small, the template penetrates them directly to reach the goal on the right of the workspace. The arrow shape also did not deform much due to the small size of the obstacles. However, the formation of a specific shape may not be kept all the time due to the obstruction of the scattered obstacles. Nevertheless, the agents in the crowd can reform and resume the desired shape very quickly along the way of penetrating the obstructive region and thereafter. The resulting simulation can be exported to commercial 3D animation packages such as Maya for postprocessing of better rendering quality as shown in Fig. 9.

VI. Conclusions And Future Work

Crowd simulation is a popular topic with great application potential in recent years. The generation of plausible crowd motions has also been demonstrated in much previous work. However, designing a crowd simulation system that can generate controllable results remains a great challenge. In this paper, we have described a motion planning and fuzzy control system that can generate crowd motions that can conform to a user-specified shape while complying to environmental constraints. Despite the high complexity of the path planning system, we have designed a practical planner that can generate paths for a deformable shape in an on-line manner. In addition, we have designed a collection of fuzzy rules to control the agents in the crowd to move in a collective, collision-free, and conforming manner toward the goal location. We have used several simulation examples to demonstrate the effectiveness and robustness of the implemented system.

Although the fuzzy control system has been built and successfully demonstrated, not all desired crowd motion behaviors have been fully covered. For example, in the current system, we sometimes find that although the agents can remain in the desired shape, it takes some time for them to spread evenly inside the shape. In addition, all the agents in the crowd in our current system use the same set of rules. This makes the creation of emergent group motion easier but it also limits the behavior diversity that can be created when the behavior model of each agent is individualized. We are currently working on these directions to design a more controllable virtual crowd.

ACKNOWLEDGMENT

This research was founded in part by the National Science Council (NSC) of Taiwan under contract no. NSC 93-2213-E-004-001.

REFERENCES

[1] M. Anderson, E. McDaniel, S. Chenney, "Constrained animation of flocks," in *Proc. of ACM Eurographics/SIGGRAPH Symp. on Computer Animation 2003*, 2003.

[2] O.B. Bayazit, J.M. Lien, N.M. Amato, "Better flocking behaviors in complex environments using global roadmaps," in *Proc. of the 2002 Artificial Life (ALIFE)*, pp.528-534 , 2002.

[3] S. Berman, M.A. Oliveira, Y. Edan and M. Jamshidi, "Hierarchical fuzzy behavior-based control of a multi-agent robotic system," in *Proc. of the 7th IEEE Mediterranean Conf. on Control and Automation*. June. 1999.

[4] J. Brown, J.C. Latombe, and K. Montgomery, "Real-time knot tying simulation," *The Visual Computer J*., 20(2/3):165-179, May 2004.

[5] J.Y. Chang and T.Y. Li, "Simulating crowd motion with shape preference and fuzzy rules," in *Proc. of Intl. Symp on Artificial Life and Robotics (AROB2007)*, Oita, Japan, 2007.

[6] G. Chen and T.T. Pham, *Introduction to fuzzy systems*, Chapman & Hall/CRC, 2005.

[7] K. Gupta, "Motion planning for flexible shapes (systems with many degrees of freedom): a survey," *The Visual Computer*, 14(5/6):288-302, 1998.

[8] A. Kamphuis and M.H. Overmars, "Finding path for coherent groups using clearance," in *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp.193-202, 2004.

[9] J.C. Latombe, *Robot motion planning*, Kluwer, Boston, MA, 1999.

[10] F. Michaud, G. Lachiver, and C. T. Le Dinh, "Fuzzy Selection and Blending of Behaviors for Situated Autonomous Agent," in *Porc. of IEEE Intl. Conf. on Fuzzy Systems*, pp. 258-264, Sep. 1996.

[11] M. Moll, L. E. Kavraki, "Path Planning for Deformable Linear Objects," *IEEE Trans. on Robotics*, 22(4):625-636, 2006.

[12] C.W. Reynolds, "Flocks, herds, and schools: A distributed behavioral model," *Computer Graphics: ACM SIGGRAPH*, pp.25-34, 1987.

[13] C.W. Reynolds, "Steering behaviors for autonomous characters," in *Porc. of 1999 Game Developers Conference*, pp.763-782, 1999.

[14] E. Tunstel, H. Danny, T. Lippincott, M. Jamshidi, "Autonomous navigation using an adaptive hierarchy of multiple fuzzy-behaviors," in *Proc. of Intl Symp on Computational Intelligence in Robotics and Automation*, pp. 276-281, 1997.

[15] J. Yen and R. Langari, *Fuzzy logic: intelligence, control, and information*, Prentice Hall, 1998.