OPTIMIZING CROWD SIMULATION BASED ON REAL VIDEO DATA

Zhixing Jin, Bir Bhanu

Center for Research in Intelligent Systems
University of California, Riverside
jinz@cs.ucr.edu, bhanu@cris.ucr.edu

ABSTRACT

Tracking of individuals and groups in video is an active topic of research in image processing and analyzing. This paper proposes an approach for the purpose of guiding a crowd simulation algorithm to mimic the trajectories of individuals in crowds as observed in real videos, which can be further used in image processing and computer vision research extensively. This is achieved by tuning the parameters used in the simulation automatically. It is required because the result of crowd simulation is very sensitive to the parameters. In our experiment, the simulation trajectories are generated by the RVO2 library and the real trajectories are extracted from the UCSD crowd video dataset. The Edit Distance on Real sequence (EDR) between the simulated and real trajectories are calculated. A genetic algorithm is applied to find the parameters that minimize the distances. The experimental results demonstrate that the trajectory distances between simulation and reality are significantly reduced after tuning the parameters of the simulator.

Index Terms— Crowd simulation, real video data, parameter optimization, genetic algorithm

1. INTRODUCTION

Crowd is one of the most common phenomena in the real life. It can be observed at every corner in the real world such as streets, schools, airports, train stations, shopping malls, *etc*. Therefore, there exist hundreds of topics related to crowd research, as they are useful in many real applications. For instance, crowd simulation is considered as an effective and efficient way in areas such as designing emergency evacuation exits, and traffic routes. Moreover, crowd simulation, especially real-time crowd simulation, has the ability to predict locations and simulate collision-free trajectories for individuals in crowd, which can provide valuable information to various image processing and computer vision research topics and applications. Therefore, it can be integrated to approaches such as tracking individuals in a complex environment, selecting the camera for surveillance in a video network, or even analyzing the crowd behavior itself, *etc*.

Nowadays, many *state-of-the-art* crowd simulation models have been proposed, *e.g.* social force [1], RVO2 [2], synthetic-vision based steering [3], and continuum dynamics [4]. However, in these papers, researchers usually did not compare the simulated trajectories to the real data because the same situations as in simulation cannot be found in the real world easily, or the data in reality cannot be captured completely. For example, the emergency evacuation is one of the most often simulated case [1], but in the reality, such situations do not happen frequently. Even when the situation happens, it is still difficult to get enough real data for further comparison. Several approaches use real data in normal situations [5, 6], but they have certain limitations such as unable to be collision-free [5] or very application-dependent [6].

In this paper, we utilize the real data of crowd in normal situations as the comparison to results from crowd simulation algorithms, as well as improve the simulation by optimize the parameters of the algorithm. To elaborate, we setup an environment that is the same as the reality and then execute the simulation algorithm to generate trajectories of individuals. The trajectories from the real data and the simulation are compared to further refine the simulator. The real data comes from the UCSD crowd video dataset, which is a continuous surveillance video on UCSD walkways for about one hour [7]. Parts of the dataset has been annotated which is helpful in this work. For the crowd simulation algorithm, we chose the RVO2 library [2]. This library is able to simulate a collision-free motion for each agent independently based on the optimal reciprocal collision avoidance (ORCA) formalism.

For parameter optimization, genetic algorithm is adopted based on the trajectory distances between reality and simulation. Genetic algorithm is a heuristic search which is known as one of the evolutionary computation methods [8]. To adopt genetic algorithm, a fitness function which is able to evaluate different parameter sets is necessary. The currently used evaluation metrics of simulation are mostly basic statistics such as maximal travel time, proportion of slower walkers [3], or have certain limitations such as specially designed for evacuation situations (*e.g.* the number of person evacuated in a certain time, the density distribution of agents over a set of polygonal regions [1, 9, 10]). According to our experimental settings,

these metrics are not the most appropriate. Therefore, the Edit Distance on Real sequence (EDR) [11] from the area of data mining is used in the evaluation process. The EDR is designed to evaluate the similarity between two trajectories, so we use it in our experiment to calculate the distance between real and simulated trajectories. The adoption of EDR can help provide more detailed information for refining the simulation algorithm while preventing the problem of over-fitting.

To the best of our knowledge, this paper is the first time to refine crowd simulation algorithms by optimizing their parameters based on EDR distances. The optimized crowd simulation algorithms can then be further utilized in other computer vision research, for example, provide predictions of pedestrian locations in multi-people tracking tasks. The result demonstrates that our approach significantly reduces the distance between the simulated trajectories of individuals and the trajectories extracted from real video. That is, the proposed approach is effective in parameter optimization for the simulation algorithms.

The rest of the paper is organized as follows. Section 2 describes the proposed approach in detail. The experimental results are then illustrated in Section 3. Finally, Section 4 contains the conclusion.

2. TECHNICAL APPROACH

The purpose of our approach is to find a set of parameters that minimizes the distances between the simulated and real trajectories extracted from the video, that is

$$\underset{P \in \mathbb{P}}{\operatorname{argmin}} ||T^s(P) - T^r|| \tag{1}$$

Where $\mathbb{P}$ is the parameter space and $||\cdot||$ is the distance measurement, $T^s(P)$ is the simulated trajectory based on parameter set P and T^r is the corresponding real trajectory.

At the beginning, we define the lower bound and the upper bound of the parameters (the range of the parameters), let's say P^{min} and P^{max}, where each of P^{min} and P^{max} is a set of parameters that will be used in the crowd simulation algorithm. Then we generate a population of parameters through randomly picking up a set of parameters in the range by the method of weighted average:

$$P = \sum_i W_i P_i \tag{2}$$

Here $i = 1, 2$, $P_1 = P^{min}$ and $P_2 = P^{max}$, W_i is a randomly generated weight vector and $\sum W_i = 1$.

After that, each individual in the population is visited to generate simulated trajectories and its fitness is calculated based on the distance between the simulated and real trajectories. When each individual has its fitness, we select the individuals with highest fitness values to generate a new population via crossover and mutation operations (these two operators will be discussed below in detail). This process terminates when one of the following criteria is satisfied: 1) the smallest trajectory distance $dist < T_d$; 2) the smallest average error rate of the simulated trajectories for different sets of parameters $e_{avg} < \varepsilon$ (the average error rate is defined in Section 3); 3) the maximal number of iterations is reached.

Algorithm 1 The pseudo code of the proposed approach.

Require: P_{min}, P_{max}
  $initialize(Pop)$
  while $!stop(Pop)$ **do**
    $D \leftarrow \emptyset$
    for all $P \in Pop$ **do**
      $d \leftarrow simulate(P)$
      $D \leftarrow D \cup \{d\}$
    end for
    $F \leftarrow fitness(D)$
    $Pop' \leftarrow select(Pop, F)$
    $crossover(Pop')$
    $mutation(Pop', P_{min}, P_{max})$
    $Pop \leftarrow Pop'$
  end while

The pseudo code of the algorithm is briefly described in Algorithm 1.

2.1. Crowd Simulation

The crowd simulation algorithm used in the current approach is RVO2 [2]. It is capable of efficiently generating motions of each individual without collisions based on a formalism called optimal reciprocal collision avoidance (ORCA). The name RVO stands for Reciprocal Velocity Obstacles, where velocity obstacle is defined as the the set of all relative velocities of A with respect to B that will lead to a collision in time τ. Given $D(p, r)$ as the circle centered at p with radius r, the velocity obstacle can be described as:

$$VO^{\tau}_{A|B} = \{v | \exists t \in [0, \tau] : v \cdot t \in D(p_B - p_A, r_A + r_B)\} \tag{3}$$

Here p_A and p_B are the current positions for A and B, and r_A and r_B are the radii of A and B. For the optimal reciprocal collision avoidance formalism, the purpose is to find the velocities v_A and v_B that are closest to v_A^{opt} and v_B^{opt}. The optimal result can be obtained by using RVO2 library. The parameters of this algorithm will be detailed described in Section 3.

2.2. Genetic Algorithm

In the genetic algorithm, each set of parameters is considered as an individual. The genetic algorithm keeps a population of different individuals and requires 1) a fitness function to evaluate the performance of each individual and 2) the crossover and mutation operators to generate new individuals (new population).

To calculate the fitness of each individual, we first use the Edit Distance on Real sequence (EDR) as the distance measurement between the real trajectory and its corresponding simulated trajectory. Assume the two trajectories T^1 and T^2 are $\{(t^1_{0,x}, t^1_{0,y}), (t^1_{1,x}, t^1_{1,y}), \ldots, (t^1_{m,x}, t^1_{m,y})\}$ and $\{(t^2_{0,x}, t^2_{0,y}), (t^2_{1,x}, t^2_{1,y}), \ldots, (t^2_{n,x}, t^2_{n,y})\}$, and the function $cost(t^1_i, t^2_j) = 0$ if and only if $|t^1_{i,x} - t^2_{j,x}| < \epsilon$ and $|t^1_{i,y} - t^2_{j,y}| < \epsilon$ (otherwise $cost = 1$), then the EDR is calculated using dynamic programming:

$$||T^1 - T^2||_{EDR} = \begin{cases} n & \text{if } m = 0 \\ m & \text{if } n = 0 \\ \min \begin{pmatrix} ||T^1_{Rest} - T^2_{Rest}||_{EDR} + cost(t^1_0, t^2_0), \\ ||T^1_{Rest} - T^2||_{EDR} + 1, \\ ||T^1 - T^2_{Rest}||_{EDR} + 1 \end{pmatrix} \\ \text{otherwise} \end{cases} \tag{4}$$

Here T^i_{Rest} denotes the rest part of trajectory i. Based on the EDR, the fitness is evaluated as below

$$f_i(T^s(P), T^r) = \frac{1}{\mathbb{Z}} \frac{1}{||T^s(P) - T^r||_{EDR}} \tag{5}$$

Here $f_i(\cdot)$ is the fitness for the i^{th} parameter set in the population, and $\mathbb{Z}$ is the normalization factor.

For the crossover operator, we still use the weighted average method, that is, given two sets of parameters P^i and P^j, the new set of parameters P is obtained by

$$P = \sum_k W_k P_k \tag{6}$$

Here $k = i, j$, and W_k is a randomly picked weight vector which states to $\sum W_k = 1$.

If one individual is selected to be mutated, then we randomly choose one of its parameter $p_i \in P$ and use the weighted average technique to randomly pick up its value in the range of p_i (as we did in the initialization step, but now only for one of the parameters).

3. EXPERIMENTAL RESULTS

In this section, we report the results of our parameter optimization algorithm on the UCSD crowd database (Figure 1 shows several frames from the dataset). The total number of the annotated individuals in the dataset is 189. However, not all these 189 individual data are used. When we optimize the parameter, we choose 1/6 of them (i.e. 31 out of 189) as training data and use the rest to test the optimization result. A perspective transformation is applied to map the trajectories from the image to the simulation plane.

In the RVO2 library, there are 10 parameters. Except the three parameters (the position, preferred velocity, and velocity) that need to be specified and calculated with respect to each agent, we try to optimize the rest of the seven parameters, including: 1) the time step of the simulation, 2) the maximal number of neighbors each agent can observe, 3) the maximal speed, 4) the maximal observation distance, 5) the radius of the agent, 6) the minimal amount of time for which the agent is safe with respect to other agents, and 7) the minimal amount of time for which the agent is safe with respect to obstacles. Although some of them are not guaranteed (such as 6) because the trajectories of other agents are fixed to real data, the parameters are still taken into account for their possible effect on the result of simulation.

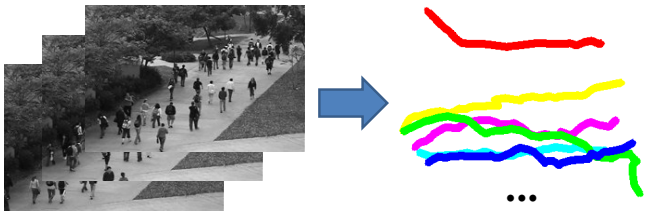

Fig. 1. Example frames and transformed trajectories. The trajectories are extracted from the dataset by annotation.

For each set of parameters, we initially tried to simulate all the individuals at the same time, but found that the generated trajectories are quite different from the trajectories extracted from real video. The reason is that when there are many individuals walking simultaneously, the difference between simulated and real trajectories will be eventually amplified, as known as the butterfly effect [12], and finally result in a totally different scenario. Therefore, we changed the simulation strategy to simulate each individual one by one while fixing the trajectories of all the other individuals to real data (ground truth). That is, the simulation is only done on a single person at each iteration. This strategy is applied for both training and testing. We then calculated the EDR for each individual and summed them up across individuals to obtain the total distance for a particular set of parameters.

Figure 2 compares the simulated trajectories of four individuals, as well as their corresponding data from real video. We can see that some of the simulated trajectories are quite similar to the real ones, but there are still some that differ largely. The most possible reason is the density of the crowd. The more individuals simultaneously exist in the scene (*i.e.* the higher the density of the crowd becomes), the larger possibility there is for the decisions made by crowd simulation algorithm depart from the real human behaviors. Additionally, different strategies between real human and simulation algorithm will take effect on the results (*e.g.* the trajectory (f) in Figure 2). The human behavior cannot be precisely predicted because we sometimes do not know the exact target position and we cannot take the accurate shortest path such as line when we walk. But in the simulation, the goal of each agent is preset and the machine will generate the shortest (or the least time-consuming) way to reach that goal.

The measurements we use to evaluate the set of parameters are based on the EDR values calculated. The first mea-

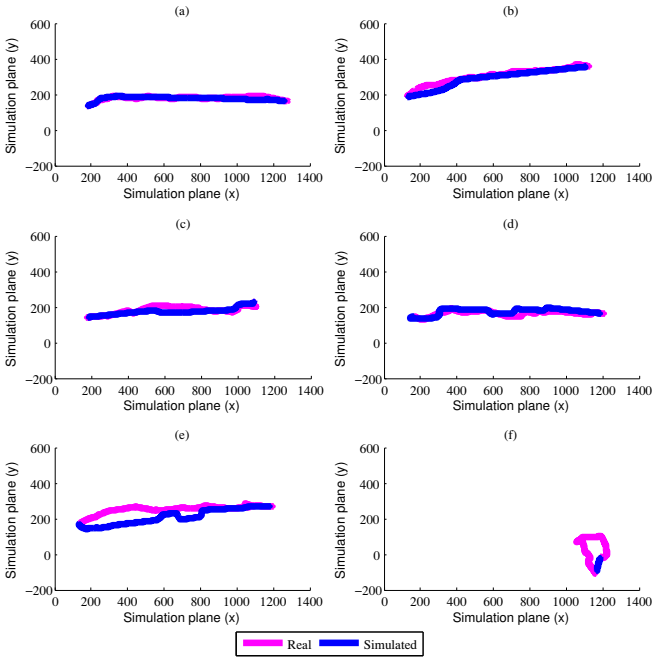

Fig. 2. Six sample trajectories. Both the simulated and the corresponding real trajectories are shown, with different distances between the trajectories.

surement is the average EDR value for each trajectory (individual), and the second measurement is the average error rate. The error rate here is defined as the number of wrong steps (EDR) over the total number of steps, and the average error rate is the mean of the error rates for all trajectories.

$$e_{avg} = \frac{1}{n} \sum_{\text{\# of traj.}} \frac{||T^s(P) - T^r||_{EDR}}{\text{number of total steps}} \qquad (7)$$

For the genetic algorithm in our experiment, the population size is set to 50, the crossover rate is set to 0.8, and the mutation rate is set to 0.1. The stopping threshold for the EDR value (the distance) is set to 1000, the stop threshold for the average error rate is set to 15%, and the maximal number of iterations is set to 100. The initial range of each parameter is estimated by using common knowledge. Although the simulation for one trajectory is at a real-time framerate, it takes about 2 hours to produce and evaluate one generation on a desktop with Intel Core Duo 2.93GHz CPU and 4GB memory.

After training, we then applied the optimized parameter set to the simulator and then run it on testing set. We also compared the results with the results from randomly picked parameter set on both training set and testing set. Figure 3 (a) is the comparison of EDR values on training set and testing set respectively. The values from two situations (optimized and randomly picked) are illustrated. It is shown that the EDR values after the optimization is significantly reduced, by an average of 304 on training set and 286 on testing set. Figure 3

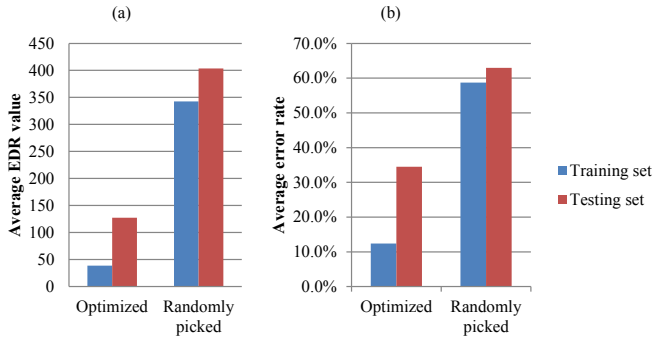

Fig. 3. The average EDR values and error rates on training and testing sets. For the optimized parameters, the average EDR values are 38.7 and 127.3 (on training and testing sets respectively, the same below), and the average error rates are 12.4% and 34.5%. For randomly picked parameters, the average EDR values are 342.5 and 403.4, and the average error rates are 58.7% and 62.9%.

(b) is the comparison between the error rates using optimized parameters and randomly picked parameters. The mean value is shown in the figure. We can also address that the optimized parameter set outperforms the randomly picked parameter set remarkably, which is revealed by the reduced error rate, about 47% on training set and 29% on testing set. The variances of these values are not considered since the inter-individual difference is one of the dominate factors on the result.

4. CONCLUSIONS

Because of the importance of parameters in crowd simulation and the lack of connection between simulated and real data, a new approach to optimize the parameters for crowd simulation algorithms is proposed in this paper. This is the first time that the data from real video is taken into account in parameter optimization for crowd simulation algorithms. Based on the UCSD crowd dataset, the results demonstrate that, after optimization, the error rate which measures the difference between simulated and real trajectories decreases by 47% on training set and 29% on testing set, which is quite significant. In the experiment, the efficiency of our approach is limited by the crowd simulation algorithm. In the future, we will expand the methods for speed-up of the simulation algorithm. Moreover, after the optimized parameters are obtained, the simulation is real-time at video rates and is quite helpful in predicting the location of each individual within crowds. Therefore it can be further used in many image processing and computer vision approaches such as tracking, camera selection in video networks.

Acknowledgement: We would like to thank the Statistical Visual Computing Lab at University of California, San Diego to provide the crowd video dataset. This work was supported in part by NSF grant 0905671. The contents and information do not reflect the position or policy of the U.S. Government.

5. REFERENCES

[1] D. Helbing, I. Farkas, and T. Vicsek, "Simulating dynamical features of escape panic," *Nature*, vol. 407, pp. 487, 2000.

[2] J. van den Berg, S.J. Guy, M.C. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *14th International Symposium on Robotics Research*, Sept. 2009.

[3] Jan Ondřej, Julien Pettré, Anne-Hélène Olivier, and Stéphane Donikian, "A synthetic-vision based steering approach for crowd simulation," in *ACM SIGGRAPH 2010 papers*, New York, NY, USA, 2010, SIGGRAPH '10, pp. 123:1–123:9, ACM.

[4] A. Treuille, S. Cooper, and Z. Popović, "Continuum crowds," in *ACM SIGGRAPH*, 2006, pp. 1160–1168.

[5] N. Courty and T. Corpetti, "Crowd motion capture," *Comput. Animat. Virtual Worlds*, vol. 18, no. 4-5, pp. 361–370, Sept. 2007.

[6] Soraia R. Musse, Cláudio R. Jung, Julio C. S. Jacques, Jr., and Adriana Braun, "Using computer vision to simulate the motion of virtual agents: Research articles," *Comput. Animat. Virtual Worlds*, vol. 18, no. 2, pp. 83–93, May 2007.

[7] A.B. Chan and N. Vasconcelos, "Modeling, clustering, and segmenting video with mixtures of dynamic textures," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 5, pp. 909 –926, May 2008.

[8] D.J. Zwickl, *Genetic algorithm approaches for the phylogenetic analysis of large biological sequence datasets under the maximum likelihood criterion*, Ph.D. thesis, The University of Texas at Austin, 2006.

[9] Yohei Murakami, Toru Ishida, Tomoyuki Kawasoe, and Reiko Hishiyama, "Scenario description for multi-agent simulation," in *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, New York, NY, USA, 2003, AAMAS '03, pp. 369–376, ACM.

[10] Bikramjit Banerjee and Landon Kraemer, "Validation of agent based crowd egress simulation," in *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1 - Volume 1*, Richland, SC, 2010, AAMAS '10, pp. 1551–1552, International Foundation for Autonomous Agents and Multiagent Systems.

[11] L. Chen, M. T. Özsu, and V. Oria, "Robust and fast similarity search for moving object trajectories," in *ACM SIGMOD International Conference on Management of Data*, 2005, pp. 491–502.

[12] J.S. Lee and A.K. Khitrin, "Quantum amplifier: Measurement with entangled spins," *Journal of Chemical Physics*, vol. 121, no. 9, pp. 3949–3951, Sept 2004.