Density-controlled crowds

Kevin Jordao, Julien Pettré, Marie-Paule Cani

Density-controlled crowds

K. Jordao[1], J. Pettré[1] and M.-P. Cani[2]

[1]Inria-Rennes, France
[2]Laboratoire Jean Kuntzmann, University of Grenoble & Inria, France

Abstract

We present a novel algorithm for populating and animating an environment of a given density. Our algorithm considers two kind of densities: static density (when people doesn't move) and dynamic density (when people move). The resulting flows of characters are computed according to a density map provided by a user. We discuss the advantages and drawbacks of the proposed solution, as well as future work.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism —Animation

1. Introduction

Designers need tools to populate virtual environment in an easy and intuitive manner. In a crowd simulation, density results from agents goals and traffic conditions: density is not explicitly controlled and it can be difficult to obtain a desired result. Density is a key factor for the general aspect of a crowd. However, there is still a lack of techniques allowing to populate virtual environments by controlling the density.

In this work, we suggest using crowd patches [YMPT09] to enable populating a virtual environment with controlled density. Crowd patches are small elements of precomputed moving crowd animations. Animated characters in patches can be either exogenous (navigate from patch to patch) or endogenous (always remain inside one given patch). Patch animations are periodic in time so that they can be played endlessly. Boundary conditions of patches are controlled, so that patches can be connected together to compose seamless large animations. Boundary conditions are controlled by patterns which stand at the interface of two connected patches, and which register all the spatio-temporal waypoints for exogenous characters.

Based on the crowd patches technique, our objective is to populate a given environment with controlled density. As an input, the user provides a map of desired density for moving and static people. As an output, we compute the crowd partches to get the final animation.

Our solution is an iterative process. The solution starts with a set of empty patches that cover the user's map. Iteratively, we fill patterns with the required number of spatio-temporal waypoints. Then, we complete crowd patches content with endogenous characters. From all these constraints (spatio temporal way-points, and presence of endogenous characters), a final step is to compute collision free trajectories for exogenous characters.

2. Computation of density-controlled crowd patches

Our algorithm input is a density map. This map is a regular grid adapted to the environment. The user defines a density value for each cell, as illustrated in Figure 2. Thz output of our algorithm is a set of crowd patches, one for each cell. The animation of each patch matches the user-defined density. The resulting crowd is composed by endogenous and exogenous characters animated by the patches .

The local density of the user's map will define the flow through patches. This flow is directly controlled by the number of input/output points at the interface of patches, called patterns. The goal of our algorithm is to find how to dispatch these inputs/outputs on each pattern, then to compute the trajectories between them.

The first step of our algorithm is to create the patterns of each crowd patch, i.e., to define a set of input/output points for characters in patches, according to the density map. Knowing that patches should have periodic animations as defined in [YMPT09], all the patterns of patches should have the same number of input and output points.

The process for computing the patterns is the following :

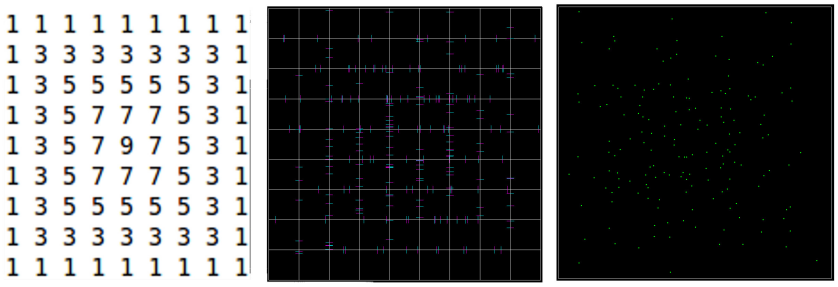

Figure 1: *The pipeline of our algorithm. A density map as input (left). Patterns are deduced from the map (middle). Crowd patches are computed from the patterns to create the desired crowd (right).*

- We take the patch with lowest density for computing its patterns;
- We add a input/ouput on a side of the patch. The side is determined by the densities of the neighbors patches. Highest the density of a neighbor patch, the more likely this shared side will be choosen. We repeat this step until no more input/output points can be inserted;
- Then some patches can still present an unsatisfying density. So, we add randomly static people inside to reach the desired density.

Once we have computed patterns and locations of static people for each patch, we need to compute the inner trajectories. For that we:

- Connect each input point to an output point. We formulate and solve this step as a stable marriage problem. .
- Deform trajectories to solve residual collisions.

3. Preliminary results

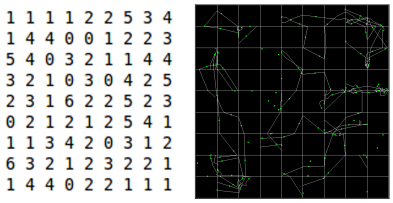

Figure 2: *Result get by a random density map.*

We implemented our solution and obtained some early results. We tested our solution on the example presented in 2. Here the map is generated randomly, to test our algorithm. We can observe more people in dense areas than in others areas.

4. Future work and conclusion

Our technique allows for an easy design of crowd with various densities. However in its current state, our technique shows some limitations.

A mobility parameter Because a crowd can have different proportions of static/dynamic density, we want to add to our algorithm a mobility parameter. This parameter will ensure the percentage of static and moving people in the scene. For instance it will enable to easly create a concert crowd by setting this parameter to zero, or a train station by setting this parameter to 0.5 or a public place with the parameter set to 1.

Accurate location of static people For now, our technique is fully random for locating static people. We can imagine ing a static map to guide our algorithm for locating people in the scene. It will avoid obtaining people waitting in undesired place, and add people waiting in some points of interest, like information panels or windows shop .

Intuitive interface We use a script for the density map information. This mecanism is not as intuitive as a graphic interface. A way to improve this system is to create an interface where user could paint the density of the crowd and see directly the result, to adjust in real time densities parameters.

Evolution of the crowd density in time The density in a crowd evolve with time (daily cycles). We want to improve our algorithm to be able, given a set of timed density maps, to compute the evolution of densities during time. It will enable to build environments with cyclic days.

Our method is a first step to a density crowd control. It allows to build a crowd with a given denstity, but we still need to work on improving our algorithm and our interface.

References

[YMPT09] YERSIN B., MAÏM J., PETTRÉ J., THALMANN D.: Crowd patches: populating large-scale virtual environments for real-time applications. In *Proc. of the 2009 symp. on Interactive 3D graphics and games* (2009), I3D '09, ACM, pp. 207–214. 1