Simulation of Pedestrian Flow on Campus using the Social Force Model

CX 4230 - Dr. Rich Vuduc

Group 1 - Ausaf Ahmed, Liam Frank, Mudit Gupta, Rahul Katre

Github: https://github.gatech.edu/cx4230-sp22-group1/campus-flow
- Code on main branch
- See README for details on where relevant code is

Website: https://github.gatech.edu/pages/cx4230-sp22-group1/campus-flow
- Static simulation of a school day (improvements coming later but not relevant to project)

Video: https://youtu.be/xwg09id7F54

Table of contents

Abstract

The system we developed is a tool to simulate the flow of students (pedestrians) across campus as they navigate between buildings throughout their day. The goals of the project are to identify how human crowd behavior emerges from pedestrian traffic rules and localized pedestrian decision making through the social force model. The social force model uses particles in a vector field, where the forces acting on the particle are determined by nearby obstacles which have a repulsive effect, nearby attractions which have an attractive effect, and nearby “strangers” (other particles) that have a repulsive effect. We will be running the simulation on a map that resembles the central part of campus, which will help us gain insights about choke points and areas of high traffic during the day, and how people take different routes to reach certain locations based on a schedule that they follow. To compare our implementation of the social force model with the one described in the original paper, we will also use maps that resemble the environments described in the original paper to see if our implementation results in the same behaviors of particles.

Project Description

Description of the system being studied

The system will consist of particles that represent the students that attend Georgia Tech. Each one of these people has things to do or places to be at a specific time, and this agenda will be intrinsic to each particle, which we call a particle’s **schedule**. Each schedule consists of a list of (location, time, duration) objects so that when a particle reaches its destination (location) at a specific time, it will remain there for a duration of time until it needs to go to the next location. People could start off their day at a certain building (dorm), and will visit locations in their schedule throughout the day, and end their day at their dorm. The destinations are the entrances of buildings, but the rest of these buildings also act as obstacles that particles need to avoid, since in the real world, you can only enter a building from defined points. Our simulation environment will begin at a smaller scale with 1-2 buildings and other small test scenarios, but we intend on expanding to a larger map of the main part of campus.

Goals

Our goals for the project were to implement the social force model and see how well it can mimic real pedestrian behavior like path planning and obstacle avoidance. We chose to specifically model pedestrian flow of students following a schedule between buildings on campus. Each simulated student would follow a schedule. The results of the simulation would help us identify areas where pedestrian traffic can build up and clog. Theoretically, an improved version of this simulation tool could be used by Georgia Tech for traffic simulation on campus.

Relevant aspects of real-world phenomenon

We implemented a model similar to the social force model described in the paper “Social force model for pedestrian dynamics” by Helbing and Molnar. The model defines several forces that act on each particle, which are attractive forces towards the destination (shortest path), repulsive forces from other particles and obstacles (personal space), and attractive forces from some particles and objects in the environment (friends, displays, etc.). The paper also describes directional weighting since things in the environment behind a person will not have as much of an affect on their behavior.

Since most pedestrians on campus are students, we want each simulated student to follow a schedule. We also tried to capture accurate pedestrian behavior. For example, we made walking paths in our simulated campus attractive regions since most people will choose to walk on a paved path rather than on the grass. However, since some students may cut across grassy areas to get to class on time, the resistance to walk on the grassy area was chosen to not be high enough to dissuade a determined student.

Literature review

For our topic, we want to simulate pedestrian traffic flow on campus at a microscopic scale, meaning that we would be simulating individual “students” moving between buildings. The students would be following a schedule, meaning that throughout the day they would travel between the buildings on campus, remaining at a building until the correct time. Using the simulation, one could identify points of congestion on campus, and how the students navigate around these points of congestion to reach their destinations. For microscopic modeling, we would use either cellular automata or particles in a vector field. With a generalized version of cellular automata, each cell would represent a student. The color of the cell would determine the destination building of the student. Once the student has reached the destination building, they would change to a different color to indicate that they are heading to a new destination.

However, there are limitations to typical cellular automata models. One paper by Sarmady, Haron, and Talib talks about some of them, as well as ideas to work around them. Typical models have difficulties with smoother and accurate movements due to coarse grain discretization of space, in addition to limited choice of speed. The Fine Grid Model utilizes small cells, with pedestrians having varying shapes and sizes (i.e. a pedestrian would take up multiple cells versus a single cell). Transition rules and next cell calculations only consider a center cell, with collision avoidance considering all cells of the pedestrian. Also, typical models only have pedestrians moving at a normal speed until reaching an obstacle or another pedestrian. This model uses perception of density, where an individual pedestrian assesses the local density around them and picks a specific speed correlating to that density from an empirical speed-density graph. In addition, typical models make pedestrians move the same speed in cardinal and ordinal directions. However, the displacement is different for each type of direction, which can skew the actual distance covered in time moving diagonally. This model has the pedestrian select cells which give optimal speed values that produce the least error from a desired movement speed, capping the movements of each pedestrian in each second of the simulation if the combination of the movements results in a displacement bigger than the free flow speed of the pedestrian. By implementing these ideas, the cellular automata model is improved to act more realistically regarding position, movement, direction, and speed.

In their paper on human mobility patterns, Serok N. and Blumenfeld-Lieberthal E. explain their spatio-temporal agent-based simulation techniques. Though this technique was not exactly a cellular automaton, it used a map divided into thousands of small cells, each with a certain classification (residential, employment, entertainment, public space, etc). On this map, there would be many agents with a unique set of parameters including age, marital status, employment status, etc. These parameters would determine where each agent wanted to be at different times of day.

These simulation techniques are quite analogous to our goals of simulating pedestrian flow on GT's campus. Different parts of the map will be either part of a walkway, class building, residence building, or something else, and there will be many different students with unique schedules that determine where they want to go and when.Another relevant detail from this paper is that different agents sometimes have non-routine destinations (going to an entertainment place at night), which may be analogous to GT students stopping somewhere to eat/study while waiting between/after classes.

With particles in a vector field, each particle would follow a vector field that directs them to their destination, but the presence of other nearby particles would affect their own field, meaning that areas of high congestion would naturally be avoided by newly arriving particles due to a large field surrounding the high congestion area pointing away from it. This is similar in concept to the social force model introduced by Helbing & Molnar [3] which describes how forces would model a person's motivations for certain movements, such as acceleration and velocity, keeping a distance from other pedestrians and borders, and being drawn towards other elements in the environment. In their paper, they found how a simulation following this model led to the creation of lanes based on walking direction, and people waiting for a doorway to be cleared by people going the opposite direction. Given that human behaviors can be observed using this model, it would be useful for modeling the movement patterns of students on campus without rigidly defining specific rules for movement.

Another model that could be used for our project is a queueing network, which was used by Løvås [4] to model pedestrian traffic in an evacuation simulation. In their paper, they simulated a building with multiple rooms and doorways, and a walkway network going through each doorway terminating at an exit node. Each room, represented by a node, has a capacity limitation, and each doorway has an "effective width" limiting the amount of people that can move through the door at a time step. The simulation tool they created, EVACSIM, models each person individually as a queueing network customer in order to incorporate things like reaction time and other human behavior.

If we used a queuing network for our project, then in theory a network could be made from different patches of the environment, forming a graph upon which paths would be routes between buildings on campus. The capacity of each node would represent the number of people that could fit in that node, just as how the room capacity works in the evacuation simulator. The width parameter of connections between nodes would represent attributes like path width, which would limit the amount of students that could move between nodes. Paths in our graph would have to be bidirectional because not all people in the simulation would have the same destination.

Simulation and model

Model summary

The model chosen to simulate pedestrian flow is the social force model proposed by Helbing et al. The social force model consists of the following forces that act on particles: an attraction force towards the destination for the particle, a repulsive force from other pedestrians, a repulsive force from the borders of obstacles, and an attraction force towards other people and objects (e.g. friends and window displays). The forces are calculated by taking the gradient of a potential function, to which the vector between the particle and the other object (particle, obstacle, attraction) is the input. Below are equations that the model defines for the forces. [4]

$$\vec{F}_\alpha^{\,0}(\vec{v}_\alpha, v_\alpha^0 \vec{e}_\alpha) := \frac{1}{\tau_\alpha}(v_\alpha^0 \vec{e}_\alpha - \vec{v}_\alpha).$$

Force towards goal

e_a is direction, v^0_a is desired velocity, v_a is actual velocity, t_a is relaxation time

$$\vec{f}_{\alpha\beta}(\vec{r}_{\alpha\beta}) := -\nabla_{\vec{r}_{\alpha\beta}} V_{\alpha\beta}[b(\vec{r}_{\alpha\beta})]\,.$$

Force away from other pedestrians

V_{ab} is monotonic decreasing potential function with elliptical equipotential lines

r_{ab} is the vector between particles a and b

$$\vec{F}_{\alpha B}(\vec{r}_{\alpha B}) := -\nabla_{\vec{r}_{\alpha B}} U_{\alpha B}(\|\vec{r}_{\alpha B}\|)$$

Force away from obstacles

U_{aB} is a monotonic decreasing potential function

r_{aB} is the vector between particle a and the part of border B closest to the particle

$$\vec{f}_{\alpha i}(\|\vec{r}_{\alpha i}\|, t) := -\nabla_{\vec{r}_{\alpha i}} W_{\alpha i}(\|\vec{r}_{\alpha i}\|, t)$$

Force towards attractions

W_{ai} is a potential function that increases with a shrinking r_{ai}, the vector between the object and the particle, and decreases with t since interest declines over time

The paper also describes a directional weighting mechanism that reduces the strength of forces that are not within the “vision cone” of the particle, or within some angle of the direction of motion for the particle. This is because of the fact that in real life, people do not have a full perception of things that exist or occur behind them. [4]

$$w(\vec{e}, \vec{f}) := \begin{cases} 1 \text{ if } \vec{e} \cdot \vec{f} \geq \|\vec{f}\| \cos\varphi \\ c \text{ otherwise.} \end{cases}$$

The forces acting on the particles are given by the following equations, the first one for repulsive effects, and the second one for attractive effects.

$$\vec{F}_{\alpha\beta}(\vec{e}_\alpha, \vec{r}_\alpha - \vec{r}_\beta) := w(\vec{e}_\alpha, -\vec{f}_{\alpha\beta})\vec{f}_{\alpha\beta}(\vec{r}_\alpha - \vec{r}_\beta)\,,$$

$$\vec{F}_{\alpha i}(\vec{e}_\alpha, \vec{r}_\alpha - \vec{r}_i, t) := w(\vec{e}_\alpha, \vec{f}_{\alpha i})\vec{f}_{\alpha i}(\vec{r}_\alpha - \vec{r}_i, t)\,.$$

These equations compute the force from each pair of particle, object, or attraction. The net force on a particle can be computed by summing all the forces between a particle and all other entities in the system.

$$\begin{aligned} \vec{F}_\alpha(t) := \vec{F}^0_\alpha(\vec{v}_\alpha, v^0_\alpha \vec{e}_\alpha) + \sum_\beta \vec{F}_{\alpha\beta}(\vec{e}_\alpha, \vec{r}_\alpha - \vec{r}_\beta) \\ + \sum_B \vec{F}_{\alpha B}(\vec{e}_\alpha, \vec{r}_\alpha - \vec{r}^\alpha_B) + \sum_i \vec{F}_{\alpha i}(\vec{e}_\alpha, \vec{r}_\alpha - \vec{r}_i, t)\,. \end{aligned}$$

Additional random fluctuations can be added to this force. The following equation gives the definition for the social force model, where w_a is the velocity vector for particle a.

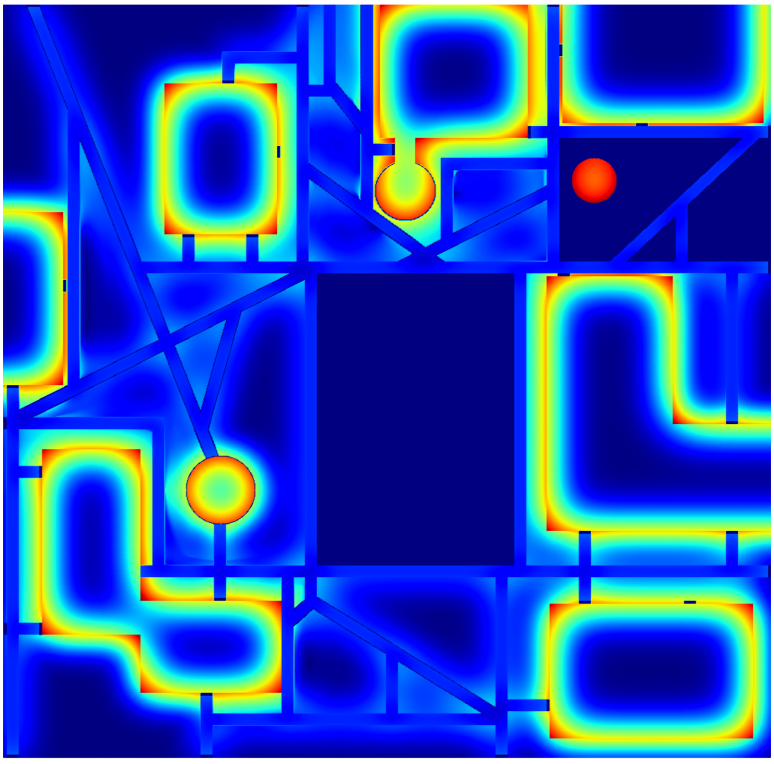
$$\frac{d\vec{w}_\alpha}{dt} := \vec{F}_\alpha(t) + \mathit{fluctuations}.$$

The paper also describes how a pedestrian’s maximal acceptable speed can be calculated, but in our implementation we simply limited the velocity with a constant. Furthermore, for attractive objects in our environment, we did not include the time dependency but rather modeled it as the opposite of an obstacle. This was mainly applied to walking paths, since we made the assumption that most pedestrians will choose to walk on a walking path by default.

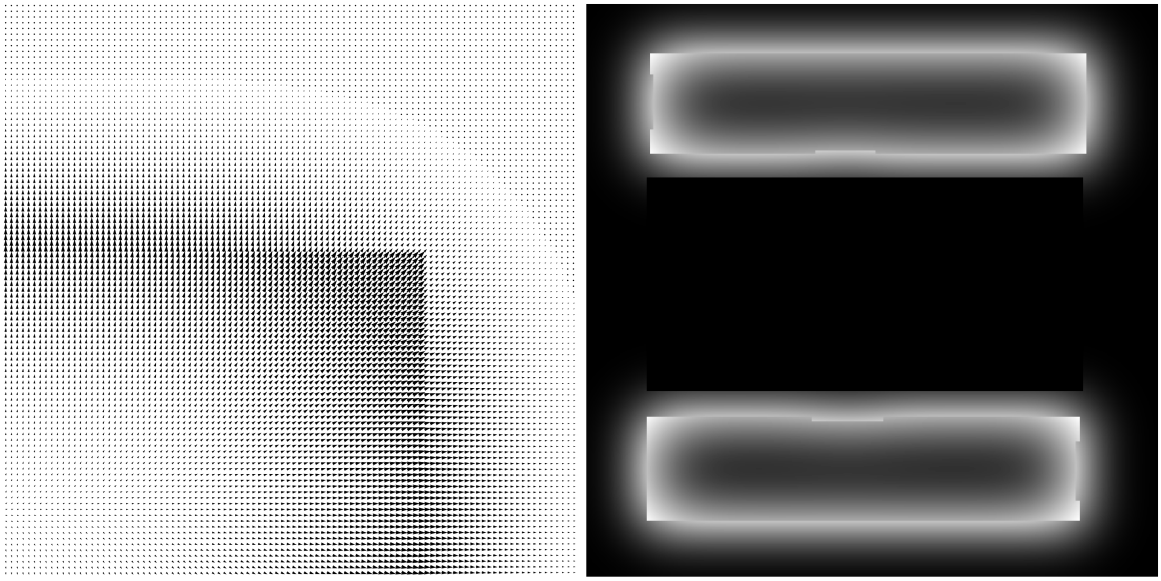
Our implementation

The main difference between the original model and our implementation of the model is in the way we calculate effects from the environment, meaning effects from all entities that are not particles. Rather than evaluating all of the stationary objects during each update of the simulation, we instead rely on a preprocessed vector field as a base, upon which the forces from other particles are added.

The map that the particles will move within is drawn in a photo tool such as Microsoft Paint, and the color of each pixel corresponds to some object in the environment. There will be different colors for obstacles, grass areas, and walking paths. In our implementation, obstacles have a strong repulsive vector field, grass areas have a weak repulsive vector field, and walking paths have a moderate attractive vector field. Our reasoning for these decisions is that walking paths are attractive to pedestrians, whereas walking on grass may not be. However, pedestrians who are in a hurry to get to their destination may choose to cut across the grassy area to reduce travel time. Using the map image, we create a vector field for the environment through a process of Gaussian blurring and then finding gradients for each color on the map, before combining the different fields into a single base layer. This means that the vector field of the environment is precomputed and only needs to be read by each particle when simulating. In theory, this leads to a significant reduction in the amount of computation needed, but also leads to a large space complexity since the entire vector field needs to be kept in memory. An generated vector field is shown below, where the color indicates the magnitude of the vector.



Here is a closer up image of a vector field near a building (obstacle), as well as the blur image from the entire field was generated. The vector field is intense closer to the building, but decreases in intensity as the distance increases. This is a result of the Gaussian blurring and dilation we applied to the mask of obstacles in the image. Effectively, we use the Gaussian distribution as a potential function for the field. Since only one side of the Gaussian is used, the potential function will be monotonic and decreasing as described in the paper. The sigma parameter of Gaussian blur gives us control over the spread of the field and the strength of the field close to the building. We also apply further weighting and masking to modify the field inside/outside regions, to make the field stronger, weaker, or point in the opposite direction.



Once the field is calculated, it is imported into the simulation. During simulation, a particle's direction of movement is based on a heading variable. Directional weighting is implemented as we noticed that particles would get stuck on obstacles without it. Every particle has some resistance to change their heading between frames. At each step, a particle updates its heading based on the field vector, the direction towards their goal, and the location of other nearby particles.

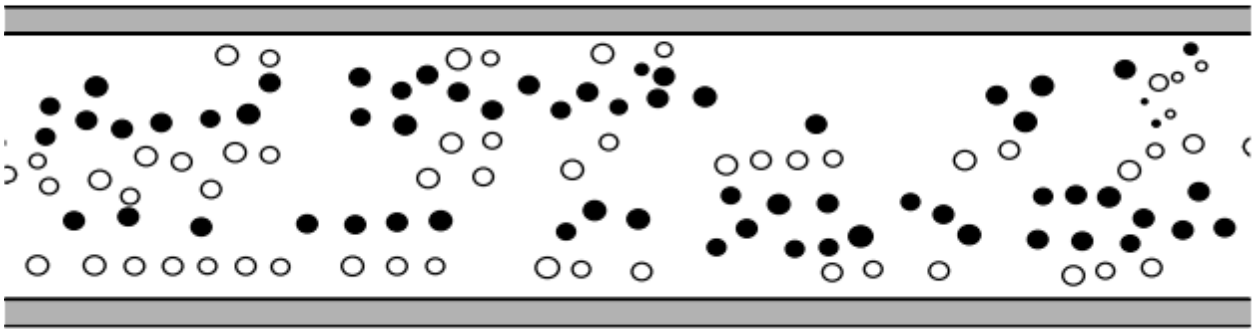
The particles of the simulation need schedules so that they know where to travel to and when. The schedule collection contains each particle's schedule, with a list of each event's location and departure time. For simplicity and to represent a fairly accurate real-life pattern, all particles have the same start and end location. The schedule generation consists of classifying each particle into different types, such as students and professors, determining the number of events a particle has on its schedule, and the location and departure time for each event. There are several parameters utilized to change the patterns of schedule generation, making the process much smoother rather than having to change or add large sections of code. These parameters also exist to allow the schedule generation to account for real-life patterns.

Platform of development

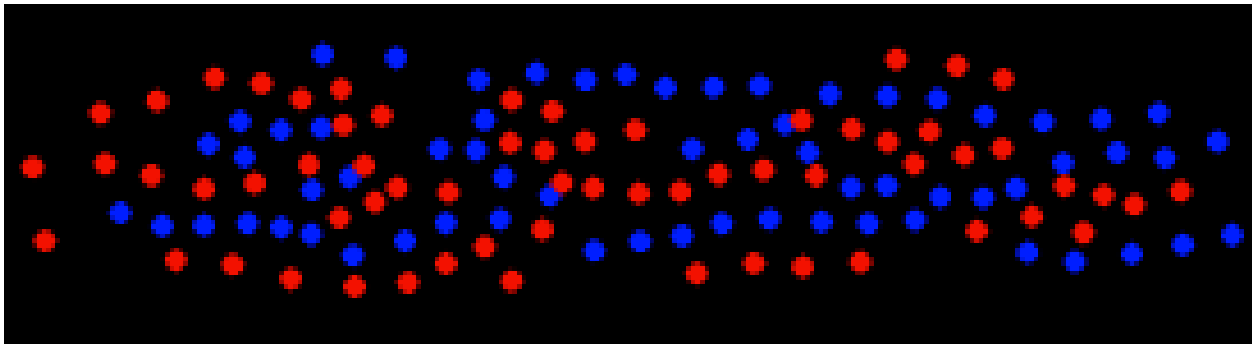
Due to the ease of visualizing moving particles using web technologies, we used a React website that displays the moving particles through a p5 sketch embedded in the page. The website will be hosted on GitHub pages. Much of the simulation code has been written using TypeScript, but for situations where more efficient code is needed to compute the different vector fields, we resorted to Python and NumPy. This is especially the case when doing precomputation tasks that only need to be run once to generate a base vector field that is a constant in the simulation. We developed a series of Python scripts, classes, and a command-line utility to process the image that is passed in as the map for the simulation, and output visualizations for debugging and fine-tuning the vector field. The code for the schedule generation is also written in Python, with the list of schedules housed in a JSON object within a TypeScript file. The output files from all of our preprocessing

Model verification

The conceptual model also had an implementation in the original paper, and showed 2 basic environments. To compare our implementation of the model, we also simulated the 2 basic environments and compared the behavior. The first basic environment is the long hallway with pedestrians walking either to the left or to the right.

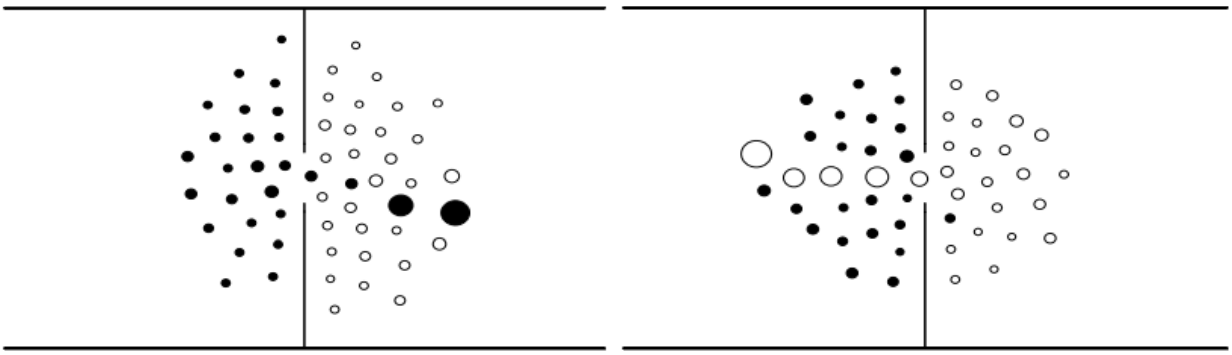


According to Helbing et. al. when the density of pedestrians moving in opposite directions is high enough, the social force model can cause pedestrians to form lanes, which is realistic behavior of pedestrians. In the figure above from their paper, the empty circles are moving in one direction while the filled circles move in the opposite direction; notice that there are 4-5 lanes formed in this “hallway”.



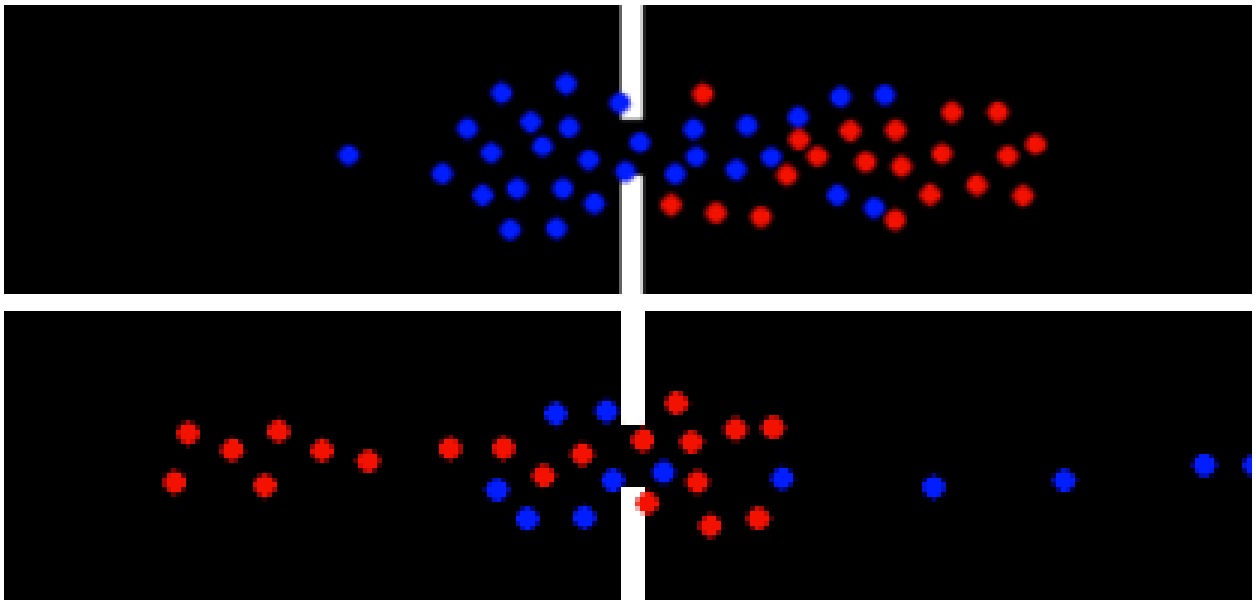
The image above is from our simulation. There is a similar lane formation between our implementation of the model and original social force model. The red particles are moving towards a destination on the left, and the blue particles are moving towards a destination on the right. While the lanes are not as “neat” as the ones in the original paper, this difference can be attributed to different force strengths that we chose for interactions between particles.

The second basic environment is the doorway, where there are two groups of pedestrians on opposite sides of a doorway, and both groups are trying to get to the other side.



In Helbing et al. the authors point out how one group dominates at the beginning and forces their way through the door, in which the leader creates a path for subsequent particles to follow. As time passes, the other group is able to take over and they are able to walk through the doorway in a similar manner. This “switching” motion continues as long as the conditions (groups on both sides of the doorway) are met. The above figures from their paper show this behavior in action. Particle sizes indicate velocity, and colors indicate direction.

The following images are results from our model. Although we had a bug where the particles would sometimes jump the obstacle, for the most part the switching behavior was present, since one group of particles would reach the door first and after a certain amount of particles crossed the doorway, the group that was waiting would be able to cross.



Both of these basic environments ultimately show that our implementation of the model is very similar to the one described in the original paper. The key difference is how obstacles are implemented, and since the original paper does not use an image based solution but rather a code-based solution for encoding the obstacles in the environment, this could explain how their particles understand not to move through the wall, but it also could just be a flaw in our vector field computation that causes the repulsive strength from the obstacle to not be large enough to resist a particle from passing through it.

For our actual simulation on the campus map, we compared by hand the movements of the particles with what we would do in real life at these places on campus. The particles mostly behave almost identically to humans in real life. Even though the simulation was implemented relatively simply, the particles still take shortcuts in a similar fashion to humans. They also tend to stay in groups and walk near the edges of buildings, similar to humans.

Experimental results and validation

Simulated experiments

1. Analyze travel time distributions for different start/end buildings

Based on the simulations, we kept track of the travel times for going between different buildings. For every pair of buildings, measurements were taken for the minimum, average, maximum, and standard deviation of the time it took to get between the two places. In most cases, the standard deviation was quite low since most particles would follow the same path between the same pair of buildings.

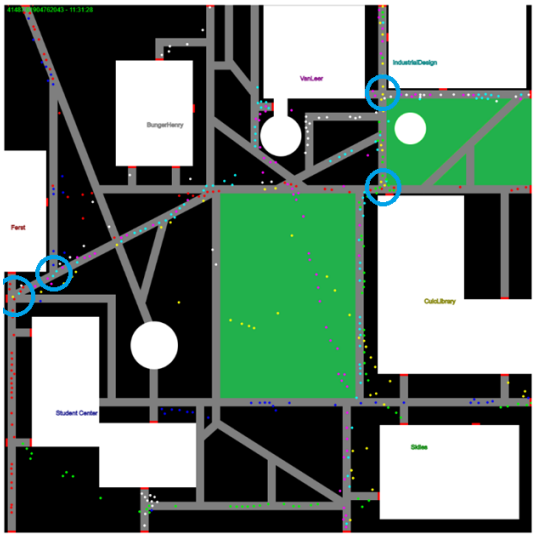
The buildings from which particles traveled most quickly were:

- Culc/Library to Skiles - ~30sec average but up to 1 minute
- Ferst to Student Center - ~40sec average but up to 1:15

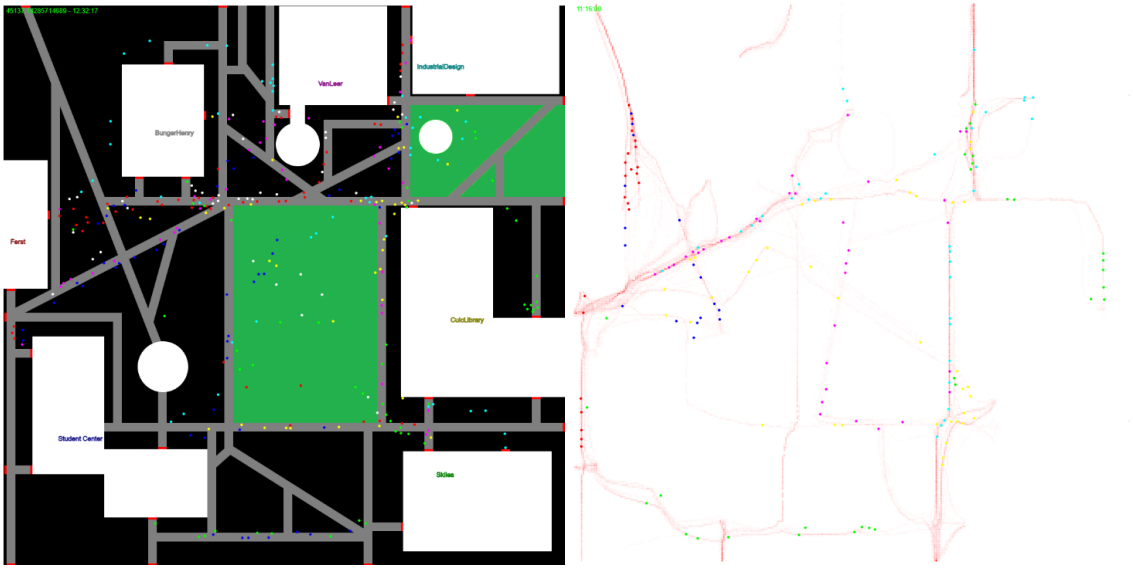
The longest travel times were up to 5 minutes and usually occurred between buildings on opposite ends of the map section being simulated.

Across the board, the longest trip time was 7.5 minutes, which implies that the 15 minute gap between classes at Georgia Tech is quite reasonable for most students taking classes near the center of campus.

2. Analyze which intersections get very busy

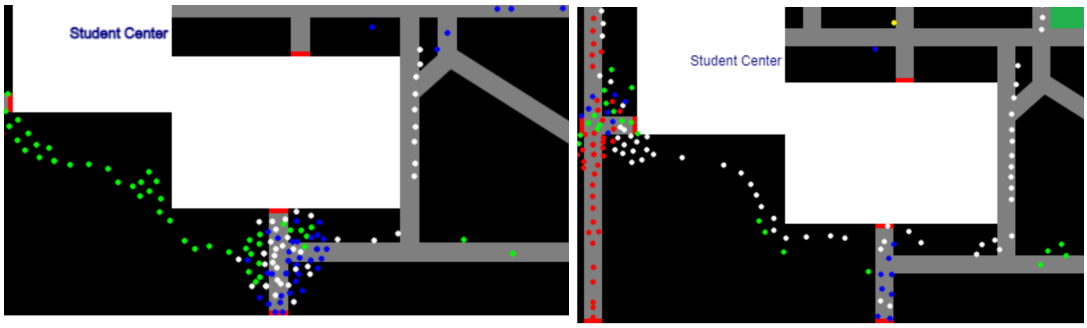


Above is a picture of the map during simulation when there is a moderate amount of particle traffic. In this instance, most particles are moving from borders (indicated by red lines on the edges of the map) to buildings. The intersections near the VanLeer, IndustrialDesign, and CulcLibrary buildings, as well as the ones south of the Ferst building, are the busiest. These intersections are circled in light blue. Note that other intersections, mainly those in the central parts of the map and near buildings, have higher amounts of particle volume too, but most of the particles move in the same direction, meaning that they do not have a lot of particle congestion and higher rates of particle flow.



Above, to the left, is another picture of the map during a different simulation when there is a moderate amount of particle traffic. In this instance, most particles are moving between buildings, with little to no interaction with borders. To the right is a heatmap of where particles

usually walk. The hotspots on the heatmap align with the visuals of heavy particle volume and congestion on the simulation maps.

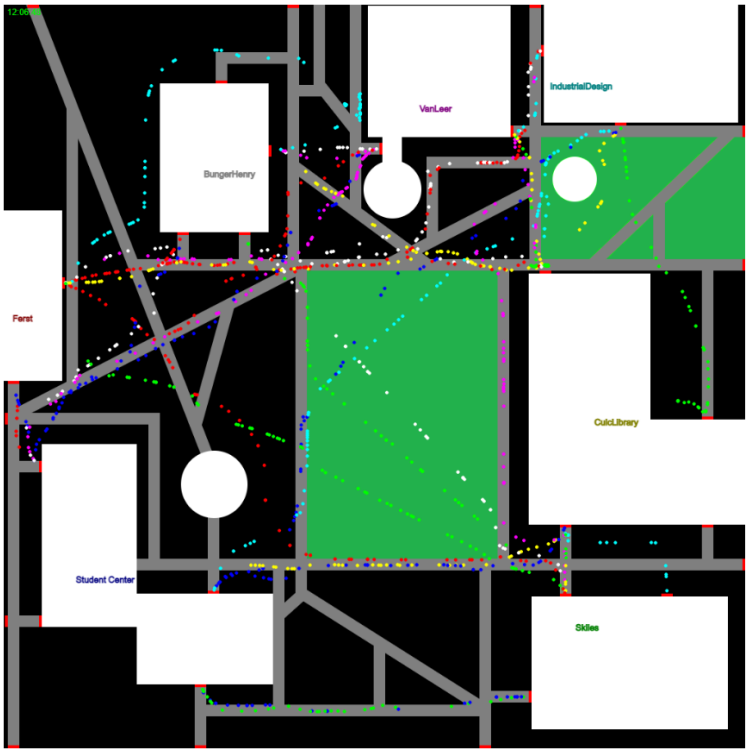


Above are zoomed in pictures of the map during simulation. The pathway intersection located just below the south Student Center door in the left picture highlights a pattern of particle movement during heavy congestion in this simulator. When there are too many particles clumped together and moving in separate directions, most particles are unable to make it past the cluster to reach their destination. Unfortunately, this fails to represent reality, as people would find a way to get through or around a large crowd. It is worth noting that in this particular case, the white particles had an error in their movement, causing them to get stuck and create a nearly impassable cluster. Specifically, the error occurred when the direction of the white particles' intended route was directly opposite to the direction of the potential field. So, code was implemented to have the particles curl around the building, either to the left or right. In the picture on the right, showcasing the map during simulation after this issue had been corrected, particles still clump together and cannot pass, except the location of the cluster is now at the intersection by the southwest door of the Student Center building and a western border of the map. The new path of the white particles cuts across the path of the blue particles to the southwest door of the Student Center building, which is what causes the issue. Also, a western border of the map and the southwest door of the Student Center building intersect with a different pathway at this spot of the map, which means that many different types of particles heading in different directions have to pass through a dense area of the map. This makes that spot of the map vulnerable to nearly impassable clusters forming.

3. Analyze common shortcuts

Particles often take "shortcuts" by going off the path if it means they can get to their destination faster. These shortcuts are taken by particles based on their preference for reaching their destination versus following the potential field / social force. When the field direction lines up relatively well with where the particles want to go, then they will align themselves more with the field. When the field is pushing them in a direction not towards their goal, particles may decide to just go off the paths and walk towards their goal.

In simulation, particles will often follow the paths for some time before making a cut through an open area in campus, or vice versa. This includes cutting across areas like Tech Green in situations like walking from Skiles to Van Leer. Below is a picture of the map during simulation when there is a heavy amount of particle traffic. Due to the efficiency of the simulator, this runthrough has particle collisions off. As a result, particles may overlap and not react to one another. Additionally, particles follow the same pathways when they have the same departure location and destination. From observing the picture, it can be seen that paths located near more buildings, especially near more doors to buildings, have a heavier volume of particles and therefore more particle congestion. Paths along the central part of the map, which involve more routes to and from buildings and map borders, also have greater particle density/congestion.



Based on the above picture, we can analyze which shortcuts the particles would take in the absence of resistance from other particles taking the same routes. With collision avoidance running, most of these shortcuts are still taken.

Below are some notable shortcuts which our particles usually took:

- Some shortcuts, such as the ones that the green, white, blue, and light blue particles take across the central green area (Tech Green), do mirror real-life tendencies. In real life, it is very common for people to cut across Tech Green when traveling in a diagonal direction.
- Other shortcuts, such as the light blue particles around the top and left of the BungerHenry building and the green, red, yellow, purple, white, and light blue particles outside the east door of the Ferst building, cut across unwalkable areas in real-life, represented by vast swaths of the black areas on the map.

Additionally, the shortcuts taken by particles affect their paths, which in turn impact where the particle volume and congestion are. So, the particle volume and congestion on the map may not be the same as a simulation where the particles perfectly follow the paths.

Experiment validation

We compared the trip times with google/apple maps to make sure the walking times were within reason to real life before making judgements about optimal/worse routes. An example route was the time to go from south of Skiles to the Industrial Design building. This trip takes about 6 minutes according to official maps, but the simulation said it took about 3 minutes. To account for this, we changed the time scale of the simulation to more closely reflect these times. We compared a number of different routes and chose the final time scale such that the average ratio between simulated trip time and official map trip time was closest to 1.
The time scale ended up at 35, which means that 35 milliseconds of simulation correspond to 1 second of real time.
The actual paths taken by most particles were also validated by cross referencing the results with possible routes based on a real map. Most of the shortcuts taken were valid based on other features of the map. A point of failure was that our map does not encode elevation, so there were some areas where particles took a technically impossible shortcut since there is a very steep incline in the way.

Discussion, conclusions, summary

Outcomes and learnings

We learned about the social force model and how it can be implemented to model pedestrian dynamics. We also found ways to improve the efficiency of existing social force simulations. One such example is precomputing the entire field for every pixel of the environment rather than computing social forces for each particle from each building. We also optimized the map creation process such that buildings don't need an explicit encoding since their field effects are simply extracted from the map image itself during precomputation of the field.

In regards to the system being modeled, we realized that students often travel near buildings since they go towards the goal until there is a building in their way. This showed up in simulation, and it is reflective of what we see in real life. From the viewpoint of the schedule generation, more particles leads to greater pedestrian volume, while more pathways and intersections, along with higher volumes of pedestrians, generally leads to greater pedestrian congestion. These are both basic principles of real-life pedestrian flow. Furthermore, despite the variety within particle schedules, particles followed the same paths to reach the same locations even when presented with multiple choices. This parallels real life, as people tend to walk the most efficient and well-known routes to reach a location.

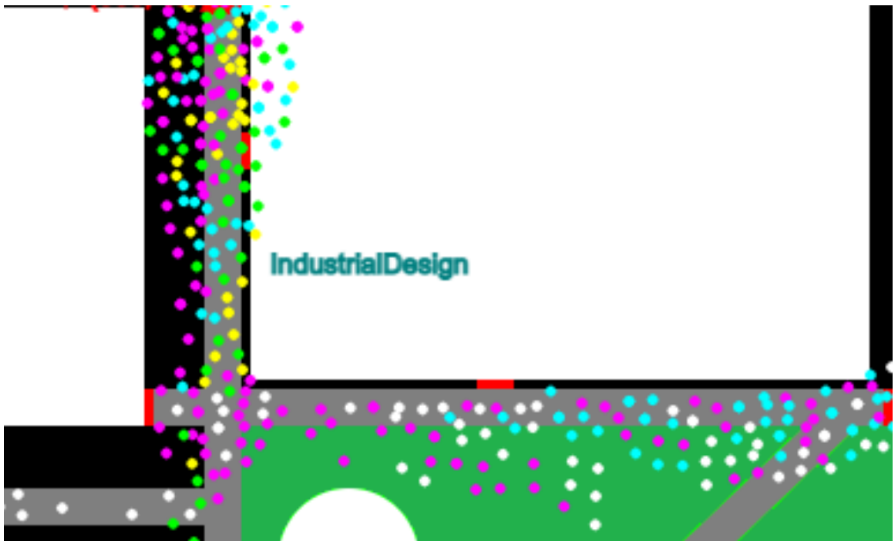
A long term outcome for this project is a potential use case for Georgia Tech's office for campus planning. Their current system for tracking traffic is to simply count people on different walkways on campus. A more fully fledged simulator could be used to provide GT with more meaningful and granular insights about the flow of students across campus.

Areas of improvement

Potential field/social force implementation

While running experiments with the simulator, it was apparent that some locations other than building entrances ended up being local minima within the potential field of the simulation. Because of this particles would sometimes get stuck and be unable to move since they simply follow the field towards their goals. A potential solution to this would be a curl field which forces particles to walk around the circumference of a building until the entrance is reached. This stops them from getting stuck on the exact opposite side of a building from its entrance.

We implemented curl fields, but they ended up not achieving the desired result, so we disabled them in simulation. The eventual solution was for particles to detect when they were in a local minima, and then, they would try to move perpendicular to the field which usually led them around the circumference of the building. This could be improved to ensure that particles know ahead of time where local minima might be such that they avoid those spots.



In the above picture, the simulation was run with an extremely high number of particles spawning per unit of time. Though walking off the paths is somewhat realistic for when there are large events on campus with hundreds of people, it is unrealistic for particles to start phasing through the walls of the buildings. This is a flaw within the implementation of the simulator that lets particles still enter buildings when there are enough other forces.

Improvements to simulator engine

First of all, instead of relying on a web application to visualize the simulation, it may be better to use plotting libraries in Python since they also have animation capabilities. Another implementation of the social force model we found online used Matplotlib to create animated scatter plots, where each point in the scatter plot was a particle. Furthermore, these libraries also allow for the use of NumPy matrices, which could be used instead of particle objects to improve efficiency of the simulation. For example, three Nx2 matrices could be used to describe the positions, velocities, and accelerations of all the particles, where N is the number of particles. The use of NumPy matrices would allow us to write vectorized functions for updating particle positions and compute the pairwise distance matrix much faster. The main downside of such an implementation would be that the user would no longer be able to interact with the simulation at runtime, since the particle matrices would be fixed in size.

Furthermore, interactions between particles could be optimized by also using an image to calculate the total vector field. This is similar to how our image preprocessing works to compute the vector field of the map ahead of time, but this new image would be recalculated at every timestep to update the particle interaction forces. Each particle would "stamp" itself on this image, and then blurring, filtering, and dilation would be applied to compute a resulting field around the particle. Each particle would look up the force at their current location in this image and use that to compute the next movement.

To analyze the potential improvement from this implementation, if we have N particles and our map has MxM pixels, this implementation would have $O(M^2)$ space complexity due to having an image the same size as the map, but only $O(N)$ time complexity as it only iterates over the N particles once. Our current solution has $O(N^2)$ space and time complexity since pairwise distance is evaluated for each pair of particles. On higher memory systems, this could be better especially when evaluating a larger number of particles (i.e. when $N >> M$) . The space complexity could also be improved by utilizing a hash table styled data structure. Since the keys of this hash table would still be numeric (integer coordinate pairs) it is possible that accessing this hash table would be relatively fast due to a simple hash function, and for sparse particle distributions the hash table would take up far less memory than a full MxM array. Many linear algebra libraries provide functionality for sparse matrices backed by hash tables.

Schedule Generation

Overall, it was possible to generate schedules for the particles to follow. Within the simulation, it would be unlikely for a more robust schedule generation to affect the particles' movements. However, when thinking about how to make realistic schedules, improvements exist. Most apparently, since the simulation time links to real-life time, acquiring accurate, real-world data about people's schedules and routines would really help the generator understand what a realistic series of events would look like, regarding both locations and times. The schedule generation could then implement these behaviors in order to create realistic schedules for particles to follow. Additionally, not all people travel from building to building. With more advanced maps and particle behavior, particles could interact with and travel to locations that aren't buildings, as well as meander, change course, and move at varying speeds.

References

1. A Simulation Model for Intra-Urban Movements. Serok, Nimrod & Blumenfeld-Lieberthal, Efrat. https://doi.org/10.1371/journal.pone.0132576
2. Simulation of Pedestrian Movements Using Fine Grid Cellular Automata Model. Sarmady, Siamak, Haron, Fazilah, & Talib, Abdullah Zawawi. https://doi.org/10.48550/arXiv.1406.3567
3. Modeling and simulation of pedestrian traffic flow. Løvås, Gunnar G. https://doi.org/10.1016/0191-2615(94)90013-2
4. Social force model for pedestrian dynamics. Helbing, Dirk & Molnar, Peter. https://doi.org/10.1103/PhysRevE.51.4282

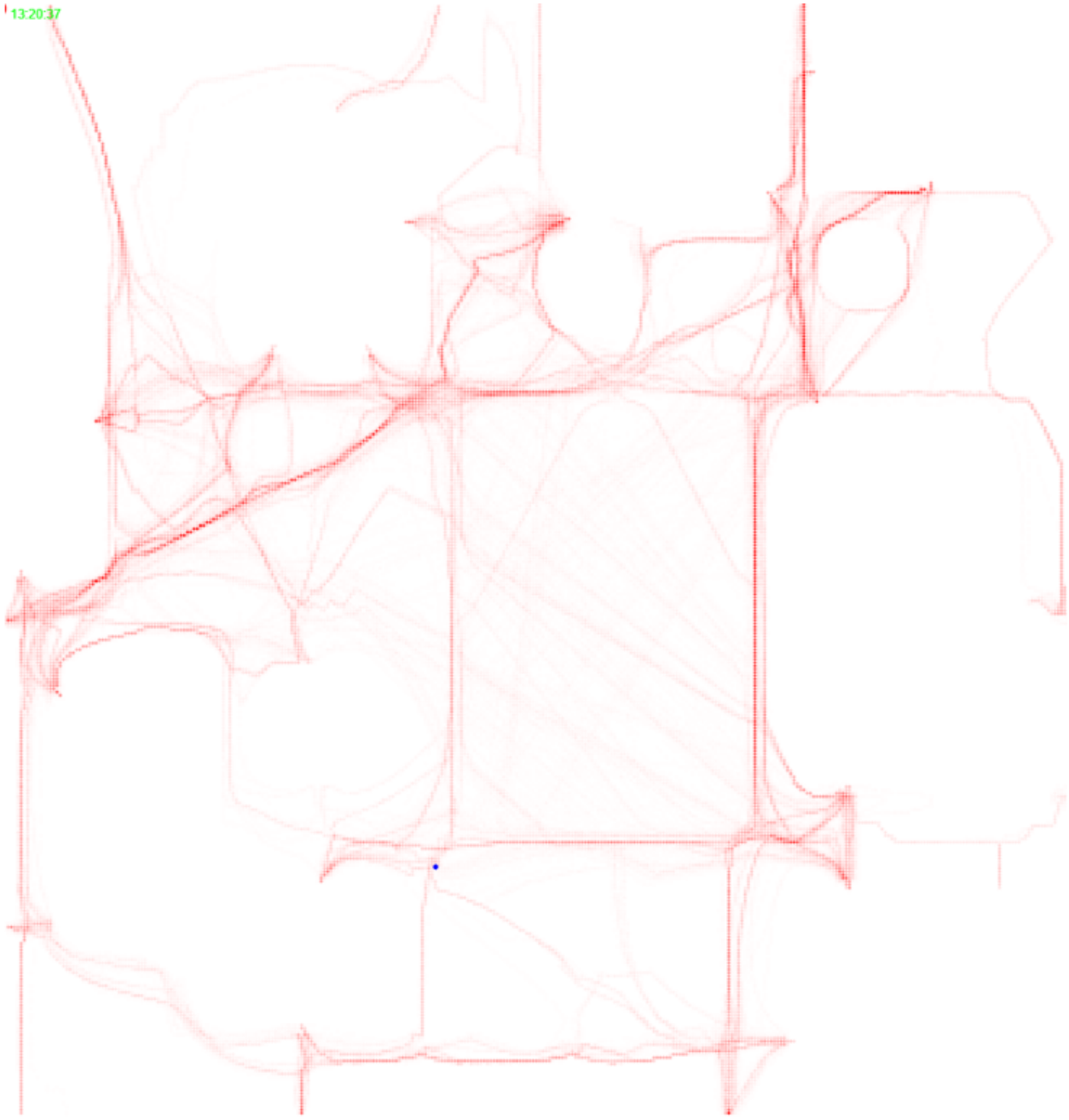
Appendix

Division of labor

Below is a list of divided-up tasks:

1. (Liam and Ausaf) Creating the environment for particles
    a. Determine color mapping for environment entities (path, grass, wall, etc.)
    b. Figure out what things in the environment affect the movement of pedestrians
    c. Create maps in paint ranging from simple (1-2 buildings) to complex (campus)
        i. Create a map just to test out preprocessing
        ii. Create a map to compare to model behavior in original paper
        iii. Create a map of campus by tracing Google Maps
2. (Mudit and Rahul) Work on model implementation
    a. (Rahul) Vector field preprocessing
        i. Separate out different colors to apply blurring per color
        ii. Use blurred image to define vector fields, combine different fields
        iii. Use Python for efficiency, export as JSON or something for JS to read in
        iv. Fine tune the weights and field generation
        v. Add dilation to image so that fields cover obstacles better
        vi. Add file exporting so that the JavaScript code can read in the field
    b. (Mudit) Implementing JavaScript simulator
        i. Create particle-particle interactions
        ii. Create update function to move particles around the view
3. (Liam and Mudit) Add schedules to simulation for particle spawning and routing
    a. (Liam) Generate schedules as a JSON object from a Python script
        i. Various schedule elements, consisting of locations and departure times
        ii. Parameters to change the schedule generation, like number of particles, number of events, location probability distributions, and time constraints
        iii. For selecting variable values, randomness is used to get better simulation runthrough and results, while weights are used to reflect real-life patterns more effectively
    b. (Mudit) Parse schedules into simulation spawning
        i. Load schedules as JSON into simulation frontend
        ii. Track particle status at each frame of simulation (is the particle ready to go to its next class, etc)
        iii. Destroy particles that have completed their schedules

More results/pictures



This is the final heat map generated from the particle’s movements throughout an entire day.