2013 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS)
November 3-7, 2013. Tokyo, Japan

Optimizing Aspects of Pedestrian Traffic in Building Designs

Samuel Rodriguez, Yinghua Zhang, Nicholas Gans and Nancy M. Amato

Abstract— **In this work, we investigate aspects of building design that can be optimized. Architectural features that we explore include pillar placement in simple corridors, doorway placement in buildings, and agent placement for information dispersement in an evacuation. The metrics utilized are tuned to the specific scenarios we study, which include continuous flow pedestrian movement and building evacuation. We use Multi-dimensional Direct Search (MDS) optimization with an extreme barrier criteria to find optimal placements while enforcing building constraints.**

I. Introduction

In this paper, we investigate the use of tools in robotics and control to improve the design of buildings. We use methods of optimization and roadmap-based motion planning to determine how placement of agents and common design features, such as pillars and doors, can affect the flow of human traffic through a building. Specifically, we seek to determine whether it is possible to place design features to optimize the rate or time it takes for humans to move through a region or floor plan. Additionally, we attempt to factor in environmental features and optimization of the environment that should be of great interest in robotics. A robot team could be placed in a flow of pedestrians to influence movement, guide, or provide information. The robots could self-optimize their configuration given the perceived motion of the human traffic. In emergency scenarios, a team of robots could be deployed in order to organize human traffic and inform them of important information. In the future, the design of spaces will become increasingly important for service robots and humans interacting in unstructured environments such as homes and offices.

We consider three scenarios: maximizing flow in a continuous traffic problem, minimizing total time in a building evacuation, and maximizing the number of encounters between two sets of agents. In these cases, we use methods of roadmap-based motion planning to simulate the behavior for a large number of pedestrians. Each pedestrian is represented by an agent, and is given partial knowledge of the environment and given a goal position. Each agent will plan a path towards its destination, based on its knowledge, and adapt its path based on new knowledge or environmental factors, including the presence of other agents.

In the continuous pedestrian traffic scenario, agents are initialized in one region of the environment and given a destination in another region. Once agents reach the destination, they are re-initialized to the initial region and pass through the area again. This way, agents continuously pass through the environment under consideration. We attempt to maximize the flow (i.e., person per minutes) through predefined regions of the environment by placing obstacles such as pillars in order to influence flow. An example of this scenario is shown in Figure 2.

In the evacuation scenario, agents are initially placed throughout an environment. As the simulation progresses, agents plan evacuation routes given each agent's environmental knowledge and mapping. The scenario is concluded when a predefined percentage of the population has evacuated a specified area. We attempt to minimize the time to evacuate the area through placement of doors. We also attempt to maximize the number of encounters that directing agents have with agents evacuating the building. An encounter is defined by passing within a pre-defined distance.

Consider a general optimization problem in the form

$$\begin{cases} \underset{\mathbf{x}\in\Omega}{maximize} & f(\mathbf{x}) \\ subject\ to & \mathbf{c}(\mathbf{x}) \preceq 0 \end{cases} \tag{1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is the target function to be optimized, $\mathbf{x} \in \mathbb{R}^n$ is a set of variables, $\Omega \subset \mathbb{R}^n$ is the space on which $f(\mathbf{x})$ is defined, and $\mathbf{c} : \mathbb{R}^n \to \mathbb{R}^m$ are constraints on $\mathbf{x}$. In our building design tasks the optimization problem, $f(\mathbf{x})$ can represent the evacuation time, pedestrian flow, or number of encounters. The variables $\mathbf{x}$ are the coordinates of obstacles, openings, or agents, and Ω is the floor plan of the building. The dimension of the optimization variable vector is determined by the number of objects to place. The feasible range is determined by both the set $\Omega \in \mathbb{R}^n$, where $f(\mathbf{x})$ is defined, and the constraints $\mathbf{c} \preceq 0$.

Often there is no explicit expression of f as a function $\mathbf{x}$, rather f can be evaluated for any particular $\mathbf{x} \in \Omega$, and acts something like a black box. Without an explicit expression for f, the analytical search direction (such as the gradient direction or Newton direction) is unavailable, limiting optimization algorithms. In such a case, Multi-dimensional Direct Search (MDS) is a good option [1], [2]. In MDS, multiple points in the search space are sampled. This establishes a section of the search space that seems closer to the optimum, and directs future sampling points.

When constraints are taken into account, many candidate techniques can be considered. The augmented Lagrangian

S. Rodriguez and N. Amato are with the Parasol Lab., Dept. of Computer Science and Engineering, Texas A&M Univ., College Station, Texas, 77843, USA {sor8786, amato}@tamu.edu

Y. Zhang and N. Gans are with the SeRViCE Lab, Department of Electrical Engineering, University of Texas at Dallas, Richardson, TX 75080, USA {yxz102220, ngans}@utdallas.edu

*The work of Rodriguez and Amato is supported in part by NSF awards CRI-0551685, CCF-0833199, CCF-0830753, IIS-096053, IIS-0917266 by THECB NHARP award 000512-0097-2009, by Chevron, IBM, Intel, Oracle/Sun and by Award KUS-C1-016-04, made by King Abdullah University of Science and Technology (KAUST).

978-1-4673-6357-0/13/$31.00 ©2013 European Union

method is always an option [3], by turning the constraints to punishment terms and adding them to the target function. Barrier methods are also used to reject the unfeasible points by giving them unacceptable function values, including extreme barrier [4] and progressive barrier [5]. If viewing the problem as a multi-object problem, namely, trying to increase the target function and decrease a constraint violation function at the same time, so called filtering methods [6] have been applied to MDS [7]. In our case, the feasible range is mainly determined by the building map, i.e., the set Ω, and we cannot place any obstacles or openings outside the building. Thus, our constraints are usually unrelaxable [5]. Therefore, we employ an *Extreme Barrier* method rather than the methods that temporarily allow infeasible points, that is $f(\mathbf{x}) = -\infty,\ \forall \mathbf{x} \notin \Omega$. We refer to our method as MDS-EB.

This paper is organized as follows. Section II briefly details related research, Section III describes important features of our roadmap-based multi-agent system, in Section IV our optimization search strategy is described, our approach to applying this optimization to virtual environments is described in Section V, and experimental results are show in Section VI.

II. Related Work

A. Multi-Agent Systems and Environmental Factors

Surveys of the approaches that have been used to study crowd simulation and evacuation have been presented in [8], [9]. Four main approaches are used including flow-based, cellular automata, activity-based and agent-based models. In flow-based modeling, the environment is represented by nodes which represent physical structures such as rooms, stairs, lobbies, and hallways. Using travel time through nodes and capacity limits while traversing areas, a macroscopic, global view of how movement and evacuation may take place can be obtained. Cellular automata based approaches discretize the environment and agents are placed in the underlying grid. This grid often represents valid movements for the agents but can be extended to represent hazards such as toxic spills or fires. Activity-based models are concerned with social factors including a fairly complete set of social and psychological attributes. Agent-based modeling allows for the representation of a number of factors, for example physical motions, gestures, proximity to other agents, influence of the environment, age and other social factors that may be included [9]. It is at this level of detail that we analyze in our simulations. In [10], the need for agent-based models to accurately represent real-world scenarios is described.

In [11] agent panic was simulated when evacuating simple environments. They also investigated the optimal strategy in order to escape from a smoke filled room. An approach to find the optimal evacuation time in simple 2D environments is described in [12] where the occupants have n possible exits and use an evacuation function to select routes. The idea of different levels of agent knowledge and planning ability is considered in [13]. This is in part due to psychology studies that show building occupants usually decide to use familiar exits, such as where they entered the building. Our research could be used to help redirect people to the best exit in a given situation. In [14], a system is developed for simulating the local motion and global way finding behaviors of crowds moving in a natural manner. They are able to simulate patient and impatient agents and pushing between agents. Improvements on the social forces model were made by considering factors that reduce shaking and vibration caused by applying social forces in densely crowded areas.

Reactive force flocking behaviors were presented in [15] where basic local rules are applied to agents including cohesion, alignment, and avoidance to create realistic looking flocking motion. In [16], steering behaviors are described for autonomous characters that integrate the ability to follow paths using local path information. These flocking techniques have been shown to add interesting and complex movements to a group of agents utilizing only those basic behaviors.

A great deal of work has been done studying real buildings and factors that influence agent movement. The movement of people in buildings and design considerations that should be made is presented in [17]. Some interesting aspects to consider include normal building use, types of populations and the need to consider evacuation policies. These factors as well as step geometries and types of handrails can impact safety and movement of people moving in a building. There has even been work on studying environmental factors that can improve the likelihood of people walking which promotes a healthy lifestyle [18].

Natural movement in realistic environments is important in studying fine grain agent motion. In [19], an agent-based system is used to simulate human movement that can generate aggregate motion similar to what is found in real buildings. It incorporates factors such as destination selection, field of view, and periodically updating the decision. Many different approaches have been proposed to handle specialized environments including pedestrians moving through Penn Station [20], underground malls [21] and a passenger ship [22].

We have used our roadmap-based with multi-agent systems to study egress and evacuation behaviors with directing agents [23], [24], [25]. The roadmap-based approach allows us to encode complex structures and handle motion and path extraction in the same way we have basic environments. We have integrated evacuation with directing agents to study evacuation strategies [23], [24]. We have also studied environmental features that effect egress for pedestrian and vehicle agents [25].

B. Optimization in Building Design

Various optimization algorithms have been used to provide systematic solutions to building design tasks. Typically, the focus has been on minimizing the cost of construction or upkeep of the building. A gradient algorithm is used to minimize the heating and cooling cost of a building based on an explicit system model [26]. In [27], building behavior is simulated by an Artificial Neural Network, and a multi-objective Genetic Algorithm is used to minimize the energy consumption while simultaneously improving the thermal

comfort for occupants. In [28], a two-stage solution by Genetic Algorithm and Ordinal Optimization is applied to minimize the annual gross energy cost of a manufacturing plant, taking random variables into account. Sequential Linear Programming is used to minimize the construction cost of dividing a building into multiple rooms [29].

III. Overview: Our Multi-Agent System

In this section, we describe the main aspects of our multi-agent system that impact the overall motion of agents in a scenario. This includes a description of the agents, their motion model, the environmental model and metrics we use to evaluate a scenario. For more information on the roadmap-based multi-agent system and applications we have studied previously, please see our prior work [23], [24], [25].

A. Pedestrian Agent Model

In this work we consider scenarios consisting of a set of N agents, $A = a_1, a_2, ..., a_N$. An agent a_i is represented by positional, velocity, and acceleration values: $a_i = \{\mathbf{p}, \mathbf{v}, \mathbf{a}\}$. These values dictate the agent's motion state in the environment. Agents are equipped with a behavior rule responsible for creating a plan for the agent given its goals and knowledge of the environment. In the scenarios we consider here, the behavior results in a route through the environment.

B. Environmental Representation

Our environmental model allows us to study both basic and complex environments. The environment is composed of surfaces that represent the valid space. The agents use each surface for generating valid roadmaps and determining if their current state is in the valid space. A surface is composed of polygons, which when projected down to a plane, do not overlap (i.e., the barycentric coordinates for each point in the polygon represent a unique location). The 3D-world coordinates also provide the height component for agents and nodes on that surface. Valid transitions between surfaces are allowed if a pre-defined height difference is met.

We utilize our roadmap-based approach to represent valid motion through potentially complex environments. The roadmap consists of a set of nodes sampled in the free/valid space of the environment. The nodes are connected using simple local planning techniques with a valid edge added to the roadmap between two nodes if the intermediate nodes lie in the valid space. An agent that needs a valid path through the environment can then query the roadmap for a start and goal configuration by connecting the configurations to the nearest node in the roadmap in the same connected component and using basic graph search techniques, a valid path can be returned.

C. Motion Model

An agent is equipped with force rules, $F = \{F_1, F_2, ..., F_M\}$ which determine the force applied to an agent at each time step given aspects of the environment perceived at that time. The cumulative forces are used to update the acceleration, velocity, and position components. For the scenarios presented here, the agents are equipped with a goal-based force rule, wall avoidance, nearest neighbor avoidance, and, if perceived, an external object avoidance force rule. The goal-based force rule guides the agent along the planned route. Each force is weighted so that a user can tune the amount of influence each component has on the agent's motion.

D. Heterogeneous Agents

Our agents also have the ability to have heterogeneous values for each component to prevent the appearance of each agent behaving exactly the same. Examples of heterogeneous values our agents have include knowledge of the environment (areas and mapping), maximum velocities and accelerations, and maximums for each force rule. Without this ability, agents would appear to move and react in almost the exact manner which prevents the simulations from looking realistic. Equipping agents with heterogeneous values would also be useful in representing an actual population where each person may move at different speeds, with differing levels of avoidance preferred between neighbors or objects in the environment.

IV. Optimization System

A. Multi-dimensional Direct Search

MDS on an n dimensional target function $f(\mathbf{x})$ iteratively performs a series of operations on a set of n+1 evolving vertices [1], which defines a simplex. We use $\mathbf{v}_j^k$ to represent each vertex, and the set $\mathbf{v}^k = \{\mathbf{v}_0^k, \mathbf{v}_1^k ... \mathbf{v}_n^k\}$ to represent the simplex, where the subscript $j \in \{0, 1, 2...n\}$ denotes the vertex index, and superscript k denotes iteration number. For each iteration k, vertices are sorted such that $f(\mathbf{v}_0^k) \geq f(\mathbf{v}_j^k)$ (for maximization problems). We call $\mathbf{v}_0^k$ the *incumbent point* and $f(\mathbf{v}_0^k)$ the *incumbent value*.

Based on the simplex, MDS generates three different types of *try points*. The first type is the *reflection try point*. As $\mathbf{v}_0^k$ is the current maximum in the simplex, it is reasonable to assume this vertex lies in a better region of the search space than the others. Therefore, we perform a reflection to generate the set of trial points $\mathbf{r}^k = \{\mathbf{r}_1^k, \mathbf{r}_2^k ... \mathbf{r}_n^k\}$ where

$$\mathbf{r}_j^k = \mathbf{v}_0^k - (\mathbf{v}_j^k - \mathbf{v}_0^k). \tag{2}$$

If there is a $\mathbf{r}_{j_r}^k$ in $\mathbf{r}^k$, such that $f(\mathbf{r}_{j_r}^k) > f(\mathbf{v}_0^k)$, where $j_r \in \{1, ...n\}$, it is possible that better points could be found further along this direction. So we perform an *extension* step, generating a second type of *extension try point* $\mathbf{e}^k = \{\mathbf{e}_1^k, \mathbf{e}_2^k ... \mathbf{e}_n^k\}$ where

$$\mathbf{e}_j^k = \mathbf{v}_0^k - \lambda(\mathbf{v}_j^k - \mathbf{v}_0^k) \tag{3}$$

in which $\lambda > 1$ is the extension parameter. If there is an $\mathbf{e}_{j_e}^k$ in $\mathbf{e}^k$, such that $f(\mathbf{e}_{j_e}^k) > f(\mathbf{r}_{j_r}^k) > f(\mathbf{v}_0^k)$, where $j_e \in \{1, ...n\}$, we accept $\mathbf{e}^k$ to update the vertices, i.e. $\mathbf{v}_j^{k+1} = \mathbf{e}_j^k$ for $j = 1, 2...n$, otherwise we accept $\mathbf{r}^k$, i.e. $\mathbf{v}_j^{k+1} = \mathbf{r}_j^k$.

If there is no $\mathbf{r}_{j_r}^k$, such that $f(\mathbf{r}_{j_r}^k) > f(\mathbf{v}_0^k)$, we perform a *contraction* step, generating a third type of *contraction try points* $\mathbf{c}^k = \{\mathbf{c}_1^k, \mathbf{c}_2^k ... \mathbf{c}_n^k\}$ where

$$\mathbf{c}_j^k = \mathbf{v}_0^k + \theta(\mathbf{v}_j^k - \mathbf{v}_0^k) \tag{4}$$

in which $\theta \in (0,1)$ is the contraction parameter. In this case, we accept $\mathbf{c}^k$ to update the vertices, i.e. $\mathbf{v}_j^{k+1} = \mathbf{c}_j^k$ for $j = 1, 2...n$. Trial points prediction and simplex update are made iteratively, until a termination condition is met.

For a maximization problem, pseudocode of the MDS algorithm is in the Appendix. For minimization problems, the trial points sequences are also obtained by (2) -(4), but the criteria for simplex update become swapping all "greater than" inequalities for "less than".

Besides the three types of try points, we optionally add more try points to each iteration to improve the performance of MDS. They are called *poll points* [4], and are given by

$$\mathbf{p}_j^k = \mathbf{v}_0^k + \Delta^k D \mathbf{u}_j^k \tag{5}$$

where Δ^k is the step length for the kth iteration, $D \in \mathbb{R}^{n \times m}$ is a generating matrix, and $\mathbf{u}_j^k \in \mathbb{N}^m$ is a selection vector, where j is a poll point index. The poll point direction set given by $D\mathbf{u}_j^k$ can be designed to be dense in the entire space with probability 1, so the poll points have higher probability to over-perform $\mathbf{v}_0^k$ than the reflection, extension and contraction points, see the LTMADS algorithm in [4].

B. Optimization Applied to Building Features

Suppose we need to place l features in a $2D$ plane, e.g., placing objects on the building floor in our scenario, the coordinates of the l objects can be represented by a vector with dimension $2l$. According to the MDS algorithm, we have an optimization problem with $\mathbf{x} \in \mathbb{R}^{2l}$, and we need $2l+1$ try points to build the simplex. Each try point is given by a vector

$$\mathbf{x} = \begin{bmatrix} \alpha_1 & \beta_1 & ... & \alpha_l & \beta_l \end{bmatrix}^T \tag{6}$$

where $\begin{bmatrix} \alpha_i & \beta_i \end{bmatrix}$, $i \in \begin{bmatrix} 1 & 2 & ... & l \end{bmatrix}$ is the coordinates of the object-i. The scenario (e.g., evacuation) is simulated for each of the $2l+1$ vectors in the simplex, and the corresponding target function values are obtained.

The optimization algorithm proceeds in Fig. 11. When a try point is unfeasible, i.e., $x \notin \Omega$, we assign a large magnitude negative value to $f(\mathbf{x})$ for a maximization problem, and a large magnitude positive value for a minimization problem. The augmented algorithm that avoids infeasible points is given by Fig. 1.

```
1:  procedure MDS-EB
2:      Obtain f(v_j^0) for j = 1, 2...2n+1 by simulations
3:      j = 1; dim = 2n + 1;
4:      tryType = reflection;
5:      J = f(v_{2n+1}^0)
6:      while terminating condition NOT met do
7:          P = MDS(J)
8:          if P feasible then
9:              Implement simulation with input P
10:             J = the resulting metric
11:         else
12:             J = { -∞  for maximization
                    {  ∞  for minimization
13:         end if
14:     end while
15: end procedure
```

Fig. 1: MDS-EB algorithm

V. Application To Environmental Design

In this section we describe our application of optimization techniques to environmental design. We specifically apply the optimization procedure to placement of agents (pillars or information providing agents) and door placement in a building environment. These components can significantly impact actual usage of a building especially in certain scenarios.

A. Optimizing the Environment

Pillars represent an architectural component often used for beautification of an area but can also have an impact on the motion and flow of agents moving near these components. There has been work on finding obstacle placements to improve pedestrian flow depending on the type of agent motion [30].

Door placement is another designed feature that can impact overall motion. Door placement is traditionally been done by architects through standards that have been developed over the years and for symmetrical purposes. While these may result in good placements, it would be beneficial to know what the optimal placement of such features are.

While this type of optimization can be applied to static objects designed into an environment, we believe this approach could also be useful for mobile robotic units to place themselves in an environment so as to influence the flow of motion or to deliver information to agents as they pass the robots. In this way, robot teams could determine how to best place themselves in order to assist in a given scenario.

B. Scenarios

We consider two different scenarios of agent motion in this work. The first is continuous agent motion for a set of agents. In this scenario, agents move from one area to another in the environment. Once the second area is reached, they are reset to a location in the first area with their next goal set to another location in the second area. This process continues for a pre-defined number of time steps.

The second scenario is an evacuation of an area. In this scenario, agents are equipped with a set of exits of a building and destination areas. The agents extract routes through the environment from their location through an exit and to a final destination.

C. Metrics

We utilize metrics specific to each scenario. In the continuous flow scenario, we compute the average speed for all agents on a specified surface in the direction of the flow. This surface is also the surface where potential pillar locations are being evaluated during the optimization process. At each time step, the metric averages the speed in the direction of the flow for all agents on the surface at that time. These values are accumulated over the duration of the simulation and the

average speed is returned. In this way, we can determine the effect the pillar placement has on the speed of agents in a specific direction.

For the evacuation scenario, we utilize two metrics to analyze motion. The evacuation time metric reports the time for a pre-defined percentage of the population to reach the final destination area. This allows us to capture the overall effect of the door placement without factoring in the effect of slow agents. The information dispersement metric returns the number of evacuating agents that encounter an information providing agent. An encounter is defined as passing within a pre-defined distance of the agent.

VI. Experiments

We show optimization results for two simulated environments: a simple corridor and a building environment. These environments consist of multiple surfaces with our roadmap-based approach encoding valid transitions for the agents. The placement of the objects through the optimization process influences the motion of the agents.

A. Environment and Agent Setup

The simple corridor environment is shown in Fig. 2. In this scenario, continuous flow is taking place through the corridor. Agents are initialized at one end of the environment and select a route that takes them to the other end. The corridor is composed of three surfaces. The first contains the initial placements of the agents and one part of the corridor. The second surface contains the area where pillars will be placed and where the average speed metric will be calculated. The final surface contains the destination region. The goal is to find a pillar placement to maximize the flow rate of agents. The optimal pillar locations found are shown in Fig. 2 for one, two, three and four pillars.

In this example, 100 agents are moving through the environment with a maximum speed ranging from 2.0 to 3.5 m/s. The continuous flow simulation takes place for 5000 time steps before returning the computed average speed for the agents. The force rules the agents are equipped with include a goal-based force (F_g), a nearest neighbor avoidance force rule (F_{nn}), a pillar avoid force rule (F_p), and wall avoid force (F_w). The range for these forces varying based on the heterogeneous settings to allow for agents moving at different speeds. Overall, the settings of these forces are approximately $|F_g| = 4|F_{nn}| = 1.5|F_p| = 3|F_w|$ which sets the goal-based force with the highest priority, followed by pillar avoidance, wall, and then nearest neighbor.

The building environment with the optimal door placements is shown in Fig. 3. In this environment, we use the evacuation metric (in time steps). This environment is based on the first two levels of a building that houses the Department of Computer Science and Engineering at Texas A&M University. The building has been modified to have all the agents evacuate through a single exit of the building, and we are optimizing door placement to produce the minimal evacuation time. In this example, 200 agents are randomly placed on the two floors of the building with maximum velocities ranging from 2.85–3 m/s. On average the force

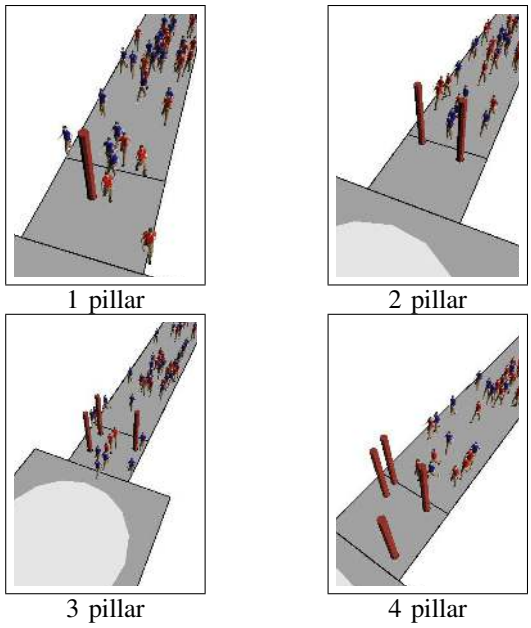


Fig. 2: A corridor environment.

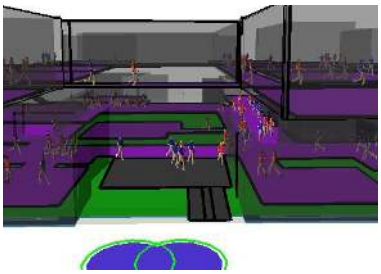


Fig. 3: A two-level building environment with two doors leading out.

rules are set so that $|F_g| = 2|F_{nn}|$. In the agent encounter scenario, only the first floor of the building in Fig. 3 is used with four exits available for agents to use. The agents are initialized in a similar way as in the door placement case.

B. Optimizing Pedestrian Flow Rate Via Pillar Placement

In the corridor scenario, we take flow rate as the optimization function $J = f(\mathbf{x})$, where $\mathbf{x}$ is a vector of dimension $2n$, and n is the number of pillars. The feasible range Ω is a subset of $\mathbb{R}^{2n}$, specified as [45:64, -11:11]. The pillars have a radius of 0.85 m and the agents have a geometric radius of 1.55 m. The ESC-EB algorithm is used to search the maximal flow rate through Ω.

For the one-pillar case, the optimization process is shown in Fig. 4. The solid line in the upper subplot shows the flow rate changing for every try point. The dashed line shows the flow rate of the current best try point, which we call the *incumbent point*. The incumbent point is always kept in the simplex for the next iteration and represents the current best configuration. We see the incumbent flow rate is strictly increasing, and the try point flow rate converges to it. The pillar coordinates of each try point are shown in the lower

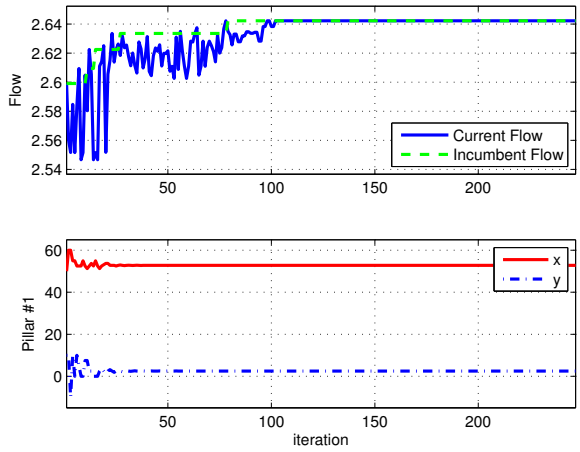


Fig. 4: Flow Optimization for 1 Pillar

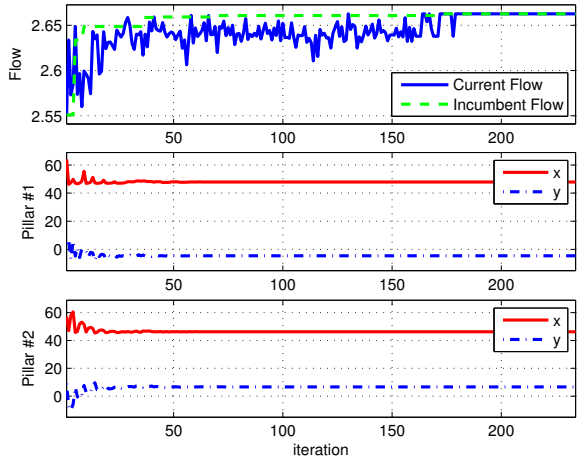


Fig. 5: Flow Optimization for 2 Pillars

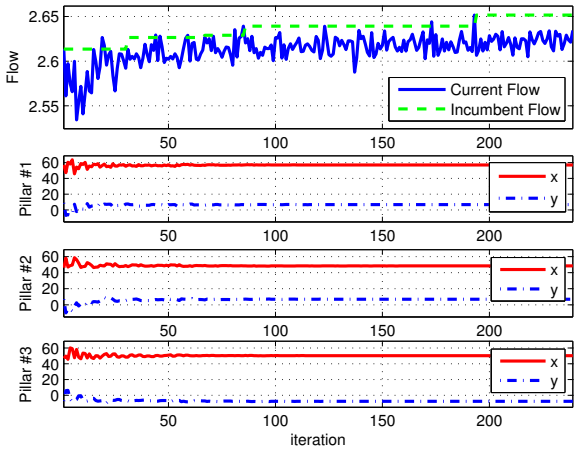


Fig. 6: Flow Optimization for 3 Pillars

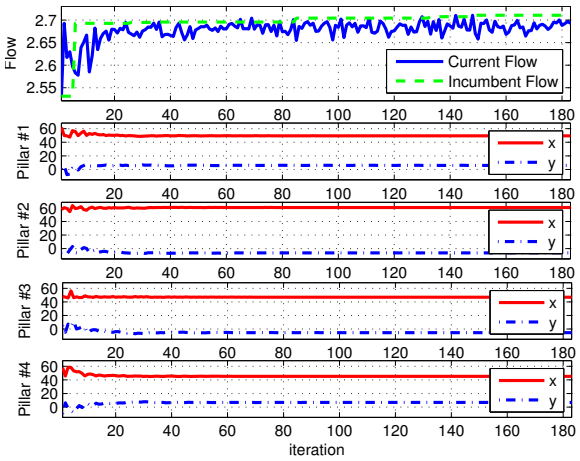


Fig. 7: Flow Optimization for 4 Pillars

subplot. It appears that the coordinates converge faster than flow rate.

Fig. 5 shows results for the two-pillar case. The flow rate of the try points converges to the incumbent rate as well, but a little bit slower than the one-pillar case. It is very interesting to note that the optimized flow rate for two pillars is faster than for one pillar. It seems counter-intuitive that adding obstacles can improve flow rate, but this type of phenomena has been observed in high-density opposite flow directions where obstacles can be used to maintain flow in each direction [30].

Optimizations for the three and four-pillar cases are shown in Fig. 6 and Fig. 7, respectively. We see the pillar positions converge well, but the convergence of flow rate for all try points to the flow rate of the incumbent point becomes slower. This indicates that, in a high dimensional Ω, $f(\mathbf{x})$ has a very bumpy neighborhood around the optimal point. Three pillars show a better optimized flow rate than for one pillar, but worse than the optimized flow rate for two pillars. The optimized flow rate for four pillars appears to be the best out of the four scenarios. Investigation of optimizing the number of pillars and position of pillars is an avenue of future work. Fig. 8 shows the coordinates of all try points during the four pillar scenario and illustrates how the four pillar coordinates converge to their optimal values. For the one, two and three-pillar cases, there are similar processes.

There are some interesting observations about the optimal placement, shown in Fig. 2. The two pillar case seems to be a subset of the three and four pillar case. The structure of the three pillar case is also similar to that of the four pillar result. In the three and four pillar case, there seems to be a predisposition for pillar coordinates that align with each other in the direction of the traffic flow.

C. Optimizing Evacuation Time Via Door Placement

In the evacuation scenario, we take evacuation time (in simulation time steps) as the optimization function $J = f(\mathbf{x})$, where $\mathbf{x}$ is a vector of dimension n, and n is the number of doors. The feasible range Ω is in $\mathbb{R}^n$ and determined by the building environment. The width of the corridor leading to the exit doors is 37 m with each door having a width of 6 m.

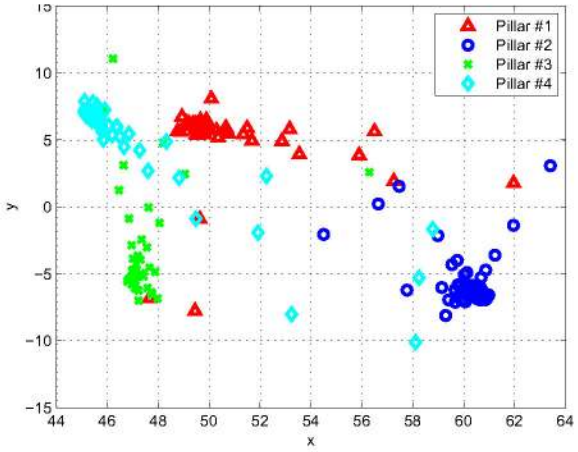


Fig. 8: 4 Pillars Evolving During the Optimization

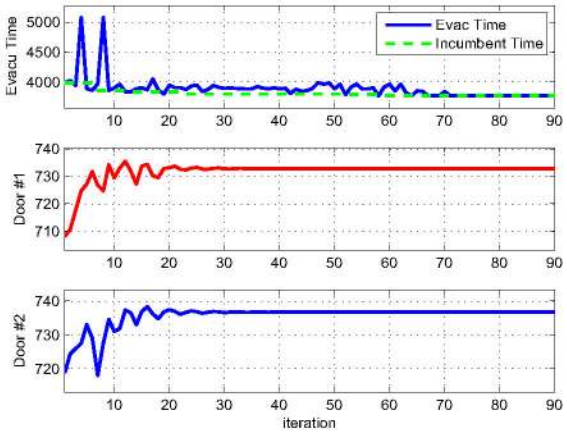
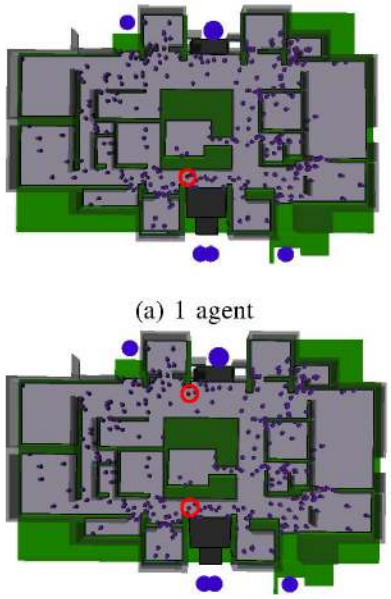


Fig. 9: Flow Optimization for 2 Doors

With these defined, the ESC-EB algorithm is used to search the minimal evacuation time in Ω. For the two-door case, the optimization process is shown in Fig. 9. The solid line in the upper subplot shows the evacuation time changing with the evolving try points. The dashed line shows the evacuation time changing with the evolving incumbent point. We see the incumbent evacuation time keeps decreasing, and the try point evacuation time converge to it. A slight improvement is achieved in evacuation time from the initial placement. The large spikes in evacuation time indicate that there are some very poor placement options, and this information could be valuable to architects as well. The evolving positions of the two doors are shown in the two lower subplots. The resulting door placement is shown in Fig. 3 showing overlapping doors (or one large door) leading out with an effective width of 10 m. It is interesting to note that the optimal placement does not occur in the center of the foyer wall.

D. Optimizing Agent Encounters

In optimizing agent encounters we use an evacuation scenario and measure the number of encounters between the evacuating agents and directing agents. The distance at which an encounter is considered valid is set to 8 m, which is large enough to span most corridors in the building. The optimized placement of the information providing agents is shown in Fig. 10 for one and two agents. These placements correspond to placements near the main exits allowing for agents to pass within the pre-defined distance before exiting the building.



(a) 1 agent

(b) 2 agents

Fig. 10: First floor of the building environment.

VII. Conclusion

In this paper we applied optimization for placements of objects in buildings such as pillars, agents, and door placement. The placement of these features could impact agent movement and could provide insight to architects and designers about how to best place these features. It would also show potentially bad placement of these features through analyzing the output of the metrics from the optimization process. In this preliminary work, we have studied two types of scenarios, continuous flow and evacuation. In the future, it would be interesting to study optimal placement of these features over a number of scenarios. Also, since each try point is run through the simulation only once, it might be interesting to see if the same optimal values are returned for many runs through the simulation per try point and for varying population types.

References

[1] V. Torczon, "On the convergence of the multidirectional search algorithm," *SIAM Journal on Optimization*, vol. 1, no. 1, pp. 123–145, 1991.

[2] V. Torczon, "On the convergence of pattern search algorithms," *SIAM Journal on Optimization*, vol. 7, no. 1, pp. 1–25, 1997.

[3] R. M. Lewis and V. Torczon, "A direct search approach to nonlinear programming problems using an augmented lagrangian method with explicit treatment of the linear constraints," *Technical Report of the College of William and Mary*, pp. 1–25, 2010.

[4] C. Audet and J. E. D. Jr., "Mesh adaptive direct search algorithms for constrained optimization," *SIAM Journal on optimization*, vol. 17, no. 1, pp. 188 – 217, 2006.
[5] C. Audet and J. E. D. Jr., "A progressive barrier for derivative-free nonlinear programming," *SIAM Journal on optimization*, vol. 20, no. 1, pp. 445 – 472, 2009.
[6] R. Fletcher and S. Leyffer, "Nonlinear programming without a penalty function," *SIAM Journal on optimization*, vol. 91, pp. 445 – 472, 2009.
[7] C. Audet and J. E. D. Jr., "Ma pattern search filter method for nonlinear programming without derivatives," *SIAM Journal on optimization*, vol. 17, no. 1, pp. 188 – 217, 2006.
[8] G. Santos and B. E. Aguirre, "A critical review of emergency evacuation simulation models," in *Workshop on Building Occupant Movement During Fire Emergencies*, pp. 27–52, 2005.
[9] N. Pelechano, J. Allbeck, and N. Badler, *Virtual Crowds: Methods, Simulation, and Control*. Synthesis Lectures on Computer Graphics and Animation, Morgan & Claypool, 2008.
[10] P. M. Torrens, "Moving Agent Pedestrians Through Space and Time," *Annals of The Association of American Geographers*, vol. 102, pp. 35–66, 2012.
[11] D. Helbing, I. Farkas, and T. Vicsek, "Simulating dynamical features of escape panic," in *NATURE*, pp. 487–490, 2000.
[12] S. C. Pursals and F. G. Garzon, "Optimal building evacuation time considering evacuation routes," *European Journal of Operational Research*, vol. 192, pp. 692–699, 2009.
[13] N. Pelechano and N. Badler, "Modeling crowd and trained leader behavior during building evacuation," *IEEE Computer Graphics and Applications*, vol. 26, no. 6, pp. 80–86, 2006.
[14] N. Pelechano, J. Allbeck, and N. Badler, "Controlling individual agents in high-density crowd simulation," in *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2007.
[15] C. W. Reynolds, "Flocks, herds, and schools: A distributed behaviroal model," in *Computer Graphics*, pp. 25–34, 1987.
[16] C. W. Reynolds, "Steering behaviors for autonomous characters," in *Game Developers Conference*, 1999.
[17] J. Pauls, "The movement of people in buildings and design solutions for means of egress," *Fire Technology*, vol. 20, pp. 27–47, 1984.
[18] N. Owen, N. Humpel, E. Leslie, A. Bauman, and J. Sallis, "Understanding environmental influences on walking: Review and research agenda," *American Journal of Preventive Medicine*, vol. 27, no. 1, pp. 67 – 76, 2004.
[19] A. Penn and A. Turner, "Encoding natural movement as an agent-based system: an investigation into human pedestrian behaviour in the built environment," *Environment and Planning B: Planning and Design*, vol. 29, pp. 473–490, 2002.
[20] W. Shao and D. Terzopoulos, "Autonomous pedestrians," in *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, (New York, NY, USA), pp. 19–28, ACM Press, 2005.
[21] H. Furuta and M. Yasui, "Evacuation simulation in underground mall by artificial life technology," in *Proceedings of the Fourth International Symposium on Uncertainty Modeling and Analysis*, 2003.
[22] C. W. Johnson and L. Nilsen-Nygaard, "Extending the use of evacuation simulators to support counter terrorism," in *International Systems Safety Conference*, 2008.
[23] S. Rodriguez and N. M. Amato, "Behavior-based evacuation planning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pp. 350–355, 2010.
[24] S. Rodriguez and N. M. Amato, "Utilizing roadmaps in evacuation planning," *24th Intern. Conf. on Computer Animation and Social Agents (CASA), 2011, in Intern. Journal of Virtual Reality*, pp. 67–73, 2011.
[25] S. Rodriguez, A. Giese, N. M. Amato, S. Zarrinmehr, F. Al-Douri, and M. Clayton, "Environmental effect on egress simulation," in *Proc. of the 5th Intern. Conf. on Motion in Games, 2012, Lecture Notes in Computer Science (LNCS)*, pp. 7–18, 2012.
[26] S. A. Jurovics, "Optimization applied to the design of an energy-efficient building," *IBM Journal of Research and Development*, vol. 22, pp. 378 – 385, July 1978.
[27] L. Magnier and F. Haghighat, "Multiobjective optimization of building design using trnsys simulations, genetic algorithm, and artificial neural network," *Building and Environment*, vol. 45, p. 739 746, 2010.
[28] H. Liu, Q. Zhao, N. Huang, and X. Zhao, "A simulation-based tool for energy efficient building design for a class of manufacturing plants," *IEEE Transactions on Automation Science and Engineering*, vol. 10, pp. 117 – 123, Jan. 2013.
[29] *Building Design Optimization Using Sequential Linear Programming*, Mar. 2007.
[30] D. Helbing, L. Buzna, A. Johansson, and T. Werner, "Self-organized pedestrian crowd dynamics: Experiments, simulations, and design solutions," in *TRANSPORTATION SCIENCE*, pp. 1–24, 2005.

APPENDIX

procedure MDS(f(**x**))
  switch $tryType$
    case reflection
      save $f(\mathbf{x})$ as $f(\mathbf{r}^k_{j-1})$;
      if $j > dim$ **then**
        if $\exists \mathbf{r}^k_{j_r} \in \mathbf{r}^k : f(\mathbf{r}^k_{j_r}) > f(\mathbf{v}^k_0)$ **then**
          $tryType$ = extension; j = 1;
          return $\mathbf{e}^k_j = \mathbf{v}^k_0 - \lambda(\mathbf{v}^k_j - \mathbf{v}^k_0)$
        else
          $tryType$ = contraction; j = 1;
          return $\mathbf{c}^k_j = \mathbf{v}^k_0 + \theta(\mathbf{v}^k_j - \mathbf{v}^k_0)$
        end if
      else
        return $\mathbf{r}^k_j = \mathbf{v}^k_0 - (\mathbf{v}^k_j - \mathbf{v}^k_0)$
      end if
    case extension
      save $f(\mathbf{x})$ as $f(\mathbf{e}^k_{j-1})$;
      if $j > dim$ **then**
        if $\exists \mathbf{e}^k_{j_e} \in \mathbf{e}^k : f(\mathbf{e}^k_{j_e}) > f(\mathbf{r}^k_{j_r})$ **then**
          $\mathbf{v}^k_j = \mathbf{e}^k_j$ for $j = 1, 2...n$
        else
          $\mathbf{v}^k_j = \mathbf{r}^k_j$ for $j = 1, 2...n$
        end if
        $tryType$ = reflection; j = 1;
        return $\mathbf{r}^k_j = \mathbf{v}^k_0 - (\mathbf{v}^k_j - \mathbf{v}^k_0)$
      else
        return $\mathbf{e}^k_j = \mathbf{v}^k_0 - \lambda(\mathbf{v}^k_j - \mathbf{v}^k_0)$
      end if
    case contraction
      save $f(\mathbf{x})$ as $f(\mathbf{c}^k_{j-1})$;
      if $j > dim$ **then**
        $\mathbf{v}^k_j = \mathbf{c}^k_j$ for $j = 1, 2...n$
        $tryType$ = reflection; j = 1;
        return $\mathbf{r}^k_j = \mathbf{v}^k_0 - (\mathbf{v}^k_j - \mathbf{v}^k_0)$
      else
        return $\mathbf{c}^k_j = \mathbf{v}^k_0 + \theta(\mathbf{v}^k_j - \mathbf{v}^k_0)$
      end if
    $j = j + 1$
end procedure

Fig. 11: MDS algorithm