TOWARDS STATIC AND DYNAMIC ANALYSIS OF ARCHITECTURAL ELEMENTS

MUHAMMAD USMAN

A THESIS SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE DEGREE OF

MASTER OF SCIENCE

GRADUATE PROGRAM IN DEPARTMENT OF ELECTRICAL ENGINEERING
AND COMPUTER SCIENCE
YORK UNIVERSITY
TORONTO, ONTARIO

JULY 2016

Abstract

This research investigates two analysis techniques, static and dynamic, to analyze the architectural elements in an environment design. The placement of an architectural element within a building design has great impact on various factors like pedestrian flow and visibility of a certain area. In this work we study computer-aided systems to help architects to improve their environment designs. As a first step, we perform a preliminary study to investigate the effects of pillar placements, choice of crowd simulators and flow-density relationships in crowd evacuation scenarios using crowd flow as an objective criterion. Next, we describe the development of two interactive computer-aided systems, `CODE` and μDOME. Both systems are capable of optimizing and analyzing different architectural elements like pillars, obstacles, door-openings and walls. `CODE` analyzes the architectural elements within an environment using dynamic crowd simulations, whereas μDOME analyzes the environments by computing spatial measures from space-syntax analysis which directly relate to human behavior. μDOME is integrated within a professional Autodesk Revit ® pipeline. Both systems are evaluated through user-studies.

Acknowledgements

I would like to express my humble and deepest appreciation to my supervisor, Professor Petros Faloutsos. His guidance, expertise and support has made my graduate experience a rewarding and transformative adventure. It was him who accepted me at first place and given me the opportunity to be a part of this exciting research zone as a graduate student in York University.

I would also like to thank Professor Melanie Baljko for being my supervisory committee member and also for accepting the position as my examination chair, as well as Professor Gunho Sohn for accepting the position as my internal examiner.

My sincerest appreciation goes to the members of the GaMaY Lab in the Department of Electrical Engineering and Computer Science at York University. In particular, I would like to thank my colleagues, Brandon Haworth and Glen Berseth. Their guidance, motivation, and support in pursuit of Crowds and Architectural Design Optimization research have been so helpful and valuable. They have provided direction, insight, and resources towards the successful completion of this thesis and the ongoing research projects.

In conclusion, I wish to recognize the various sources of financial support which made

this thesis possible: the Department of Electrical Engineering and Computer Science at York University and the NSERC Discovery Grant Program.

Table of Contents

List of Tables

List of Figures

Publications from the Thesis

All data collection and experiments presented in this thesis were conducted at GaMaY Lab, Department of Electrical Engineering and Computer Science, York University. This work is approved by York University's Human Research Participants Committee (Certificate No.: e2016-099 Computer-Assisted Design of Buildings).

Portions of the work presented in Chapter 3 have been published in CAVW 2015, MIG 2015 and VHCIE 2016 (Berseth et al. 2015, Haworth et al. 2015, 2016b). An overview of Chapter 4 has been published in CHI 2016 and VHCIE 2016 (Haworth et al. 2016a,b) and the complete work of Chapter 4 has been published in CASA 2016 and also get accepted for the journal (CAVW 2016). Work from Chapter 5 has been submitted to SIGGRAPH Asia 2016.

In Chapter 3 the formulation and development of optimization algorithm is contributed by Glen Berseth. Algorithm 1 in Chapter 4 is contributed by Brandon Haworth. In Chapter 5 optimization algorithm and Figure 5.2 is contributed by Glen Berseth and implementation of spatial metrics from space-syntax is contributed by Mahyar Khayatkhoei.

Chapter 1

Introduction

Architectural or building design is both an art and a science. The job of an architect is not just to create these designs but also to balance some important key factors like safety of the people who connect with the architectural elements, visibility of certain areas and the best space utilization. Architectural elements can be the pillars, obstacles or walls and the parameters of these elements can be their positions and sizes in an architectural environment. Such elements can significantly affect the navigation of individuals or crowds. For example, poorly placed pillars or obstacles can negatively affect visibility of a path. To achieve the ultimate objectives as smooth crowd navigations or high visibility, it is necessary to optimize and evaluate these architectural elements. Mostly the space of permissible building designs is extremely high-dimensional and continuous, and there are a multitude of diverse configurations which satisfy these constraints and criteria. However, it is impractical to thoroughly examine an entire architectural space. Computer-aided interactive systems can help to facilitate designers and architects in op-

timizing and analyzing architectural elements in their designs.

This research seeks to develop and evaluate the steps taken in: (a) analyzing the effects on the flow of crowd movements by architectural elements, (b) generating a preliminary system, CODE: Crowd Optimized Design of Environments, which affords the study and delivery of an improved optimization technique for configuring the architectural elements as well as presenting the dynamic analysis of architectural design with respect to these elements using crowd simulations, (c) and generating a professional tool, μDOME: User-in-the Loop Diversity Optimization of Environments, which affords to present static analysis of the architectural designs using spatial measures from space-syntax, integrated within an industry standard architectural design system, Autodesk Revit ®.

In this thesis, I will describe the development of such systems, their software architectures, and underlying computational processes. This can be categorized into three main tasks: The first task is to arrange the design of the systems in a top-down manner from the requirements perspective. Second task is to describe the steps to making such systems into working tools mapping from architecture to computational processes. The last task is to validate the systems which is done by conducting an A/B user study for each of the systems.

1.1 Contributions

Contributions from this work can be summarized as follow:

- Propose the use of dynamic crowd simulations to evaluate the placements of archi-

tectural elements in architectural environment designs.

- Present `CODE`, an interactive user-in-the-loop crowd-aware building and architectural design system that integrates dynamic crowd simulators, optimization and analysis techniques.
- Introduce backtrack functionality in Adaptive Mesh Refinement (AMR) optimization technique to avoid local minima.
- Present μDOME, an interactive user-in-the-loop building and architectural design system for static analysis of the architectural designs using spatial measures from space-syntax.
- Integrate μDOME within the professional Autodesk Revit ® pipeline.
- Present two user-studies to evaluate `CODE` and μDOME.

1.2 Thesis structure

The remainder of this thesis is structured as follow:

Chapter 2 presents the literature review and theoretical background in computer-aided design (CAD) methods, interactive design methods and architectural layout analysis.

Chapter 3 presents the preliminary studies with experiments for proof of concept showing that the optimized placements of obstacles or other architectural elements can help in improving crowd flow in the evacuation scenarios.

Chapter 4 describes the design and development of an interactive system for architectural or building designs. An improved optimization technique is used to optimize

architecural elements like pillars or obstacles. Architectural elements are analyzed using current generation dynamic crowd simulators. The system is evaluated with a user study.

Chapter 5 describes the design and development of an interactive system for optimizing and analyzing building or architectural designs. A multi-objective diversity optimization technique is used that uses spatial measures grounded in space-syntax analysis to quantify how humans interact with the architectural environments. The system is evaluated with a user study.

Chapter 6 concludes the thesis. Limitations and future work for the systems presented in this work are also discussed.

Chapter 2

Background and literature review

This chapter presents the literature review and theoretical background in dynamic crowd simulations, computer-aided architectural design methods, and human-factored architectural design analysis.

2.1 Dynamic Crowd Simulations

A variety of techniques have been developed and used for simulating crowds in different contexts. A particle-based approach, boids model (Reynolds 1987), considers local-level interactions to compute the speed, motion and relative position of each individual for developing large crowd behaviors. This work was further extended (Reynolds 1999) to focus more on steering behaviors and locomotions. Social forces based approaches (Helbing et al. 2000, Karamouzas et al. 2009) model the individual interactions as forces like repulsion and attraction. In (Ondřej et al. 2010), a vision-based approach is developed to detect and avoid the future collisions between interacting agents based on cognitive sci-

ence. (Singh et al. 2011) presents a rule-based hybrid framework to predict and avoid the future collisions. Work in (van den Berg et al. 2009) uses reciprocal velocity obstacles to compute the velocity space for collision avoidance. A continuum-based approach (Narain et al. 2009) introduces a concept of unilateral incompressibility for simulating large-scale crowds without collisions. An affordance based framework to compute space-time plan is proposed in (Kapadia et al. 2009). (Best et al. 2014, Narang et al. 2015) generate collision free crowd steering based on speed/density relationship through fundamental diagrams corresponding to large dense crowds.

2.1.1 Crowd Dynamics and Evaluation

Crowd dynamics and pedestrian motion have been studied for a long time. Pedestrian traffic flow shows distinct dynamics at different levels of densities which must be planned for (Fruin 1971). Evaluation of these approaches and the accuracy of simulations are beginning to attract lot of attention. A number of techniques have been proposed to address these issues. Fundamental diagrams were used as a measure of aggregate similarity in (Seyfried et al. 2008) to describe the dynamic of crowds by performing a large empirical study. (Singh et al. 2009a) proposes a framework to measure the performance of steering algorithms over a range of complex scenarios using different evaluation metrics and uses a scoring mechanism to rank these algorithms. This work is further extended in (Kapadia et al. 2011) with some new representative scenarios and evaluation metrics; and introduces a sound approach to measure the simulation quality and calculate the cov-

erage of steering algorithms. In some data-driven approaches, the performance of steering algorithms is measured using experimental data. (Lerner et al. 2009, 2010) evaluate the behaviors of individual crowds based on real-world data. A histogram approach is used to measure the flow characteristics of crowds in (Musse et al. 2012). An information-theoretic method is used in (Guy et al. 2012) to quantify the likeness of real-world data and simulations in complex crowd systems.

2.2 Computer-aided architectural design

A lot of research has been done exploring the design spaces of architectural or building environments to determine the optimal architectural designs. Such solutions produce new layout designs with respect to given objective criterion, while providing reasonable compromise between automation and author precision. (Merrell et al. 2010) presents a data-driven approach to automatically generate optimal design layouts from a trained model of prior real-world data. (Jiang et al. 2014, Rodriguez et al. 2013) studied the optimal placement of architectural elements like pillars and crowd positions for evacuation scenarios using a single steering algorithm. A framework which uses static path analysis is proposed in (Berseth et al. 2014) to generate optimized game layouts.

Since it is hard to evaluate the subjective criterion, mostly the tools select an optimization design to meet objective criterion then take a user-in-the-loop interactive approach to the subjective. (Felkner et al. 2013, Michalek and Papalambros 2002, Shi and Yang 2013, Turrin et al. 2011) are some of such tools with the user-in-the-loop automation

processes.

2.3 Human-factored architectural design analysis

Layout of a building or architectural design affect the behavior of its intended users. The significant connection between people and architectural layout should be accounted for and addressed as an important consideration during the design process. Primarily, there are two ways to analyze the architectural designs: (1) by simulating dynamic crowds through the architectural environments and (2) by computing spatial measures from space-syntax analysis which directly relate to human behavior.

Crowd simulation methods (Kapadia et al. 2015) are better representatives in measures of real human movement, but they are usually expensive to compute and thus far not readily integrated within interactive optimization and design systems. (Berseth et al. 2015, Feng et al. 2016) simulates the movement of people using a dynamic crowd simulator through an architectural design to approximate the crowd movement flow as an objective of the layout design. It is then used to analyze and automatically reconfigure the layout for maximizing the defined objective.

Dynamic crowd simulations can be expensive. A cheaper alternative to crowd simulation is static analysis. Space-syntax (Bafna 2003) is one such human-focused approach for static analysis of the architectural environments and proposes various measures such as visibility, organization and entropy to quantitatively evaluate architectural designs.

2.4 Our Work

As a first step, this work performs a preliminary study to investigate the effects of pillar placements, choice of crowd simulators and flow-density relationships in crowd evacuation scenarios using crowd flow as an objective criterion. Next, it describes the development of two interactive computer-aided systems, `CODE` and μDOME. Both systems are capable of optimizing and analyzing different architectural elements like pillars, obstacles, door-openings and walls. `CODE` is featured to analyze the architectural environments using dynamic crowd simulations. μDOME analyzes the environments by computing spatial measures from space-syntax analysis which directly relate to human behavior. Both systems are evaluated through user-studies.

Chapter 3

Effects of Pillar Placements in Crowd Evacuation

3.1 Overview

Architectural or building designs greatly affect the flow pattern of the people who connect with these designs. It has been well established that the proper placements of architectural elements like obstacles, pillars or doors can improve the pedestrian flow during evacuation scenarios. This chapter addresses the following key points:

1. To study the optimal placements of architectural elements (pillars) in environment designs and their impact on crowd flow during evacuation scenarios.
2. To investigate whether particular dynamic crowd simulators which are used to model crowd steering can significantly affect the flow results.
3. To determine the effects of different crowd densities on crowd flow.

Two sets of experiments are presented in this chapter. Experiment 1 (Section 3.4) addresses (1) and (2), whereas Experiment 2 (Section 3.5) addresses (3) from the list given above.

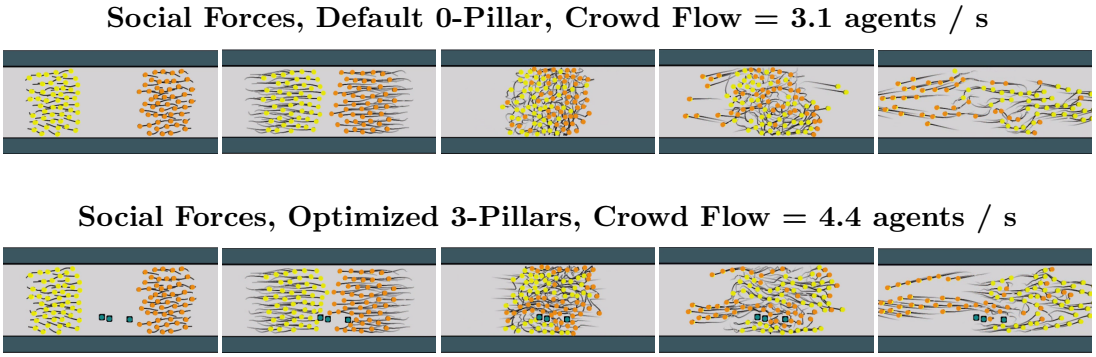


Figure 3.1: Simulation of social forces crowd simulator in a bi-directional hallway benchmark. *Blue* squares are pillars and the *Orange* and *Yellow* circles represent two different sets of crowd with opposite targets, and the line trailing behind them indicative of their most recent trajectories. Optimal pillar placements produce emergent lanes and significantly increase the crowd flow.

3.2 SteerSuite

SteerSuite (Singh et al. 2009b) is an open framework for crowd simulation and optimization techniques. It provides an easy yet powerful and comprehensive mechanism to develop your own AI, evaluate steering behaviors of dynamic crowd simulators, tune the parameters of steering algorithms, share the results with the community, analyze the architectural designs using space-syntax and much more. It also has a rich set of opti-

mization techniques implemented for linear as well as non-linear convex and non-convex optimization problems.

The dynamic crowd simulators in Section 3.3.2 and optimization formulation in Section 3.3.3 are also defined and integrated within this suite. In the remaining of this thesis, all the crowd steering algorithms, analysis and optimization techniques used will be referred through SteerSuite.

3.3 Methodology

This section describes the methodology for the experiments presented in Section 3.4 and Section 3.5. It includes the configuration of architectural environments, dynamic crowd simulators and the optimization formulation.

3.3.1 Architectural Environment Configuration

Architectural environment configuration is a specific setting of obstacles or pillars and the agents or crowd in an architectural environment design. It may also refer to an initial design setting of obstacles and crowds in the environment design during dynamic simulation. Formally, it is defined similarly to (Berseth et al. 2013), as $s = \langle \mathbf{O}, \mathbf{A} \rangle$, where $\mathbf{O}$ and $\mathbf{A}$ are the sets of static obstacles and agents in the scenario respectively. In our experimental setup, an obstacle $o \in \mathbf{O}$ is either a rectangular bounding box or a cylinder; whereas an agent $a \in \mathbf{A}$ is defined as $a = \langle \mathbf{x}, \mathbf{r}, \mathbf{g} \rangle$, where $\mathbf{x}$ is the current position of the agent, $\mathbf{r}$ is its collision radius, and $\mathbf{g}$ is the goal position of the agent.

Environment Subspace: By parameterizing an architectural environment configuration, we can define a configuration space of an architectural environment from which we can draw arbitrary samples, $\mathcal{S}_{sub}$.

3.3.2 Dynamic Crowd Simulators

The analysis and experiments presented in this chapter were performed using three well established dynamic crowd simulators selected for the diversity of their steering approaches. All of them are defined in SteerSuite:

- **ORCA**: An approach that uses reciprocal velocity obstacles to avoid collisions (van den Berg et al. 2009).
- **PPR**: A hybrid, rule based approach combining reactions, predictions, and planning (Singh et al. 2011).
- **SF**: A social forces based approach that uses repulsion and attraction forces for pedestrian dynamics (Helbing et al. 2000).

For each algorithm the default parameters, as suggested by the algorithm's developers, are used.

3.3.3 Optimization Formulation

Optimization techniques or algorithms are the heuristics which are used to find the satisfactory and optimal solutions to the given problem. They adopt different strategies to

compute the given objective either recursively or iteratively by comparing the different considerable solutions until one converges on or finds an optimum.

The optimization process use in the analysis is formulated as a minimization of crowd flow objective and a penalty function within the context of an architectural environment subspace. Given an environment subspace, $\mathcal{S}_{sub}$, a set of parameters, $\mathbf{p}$, and constraints on these parameters, $\mathcal{P}$, we can set up and solve a minimization problem to position the parameters to an objective as follows:

$$\mathbf{p}^* = \arg\min_{\mathbf{p}\in\mathcal{P}}(-f_r(\mathbf{p}) + g(\mathbf{p})). \tag{3.1}$$

$f_r(\mathbf{p})$ is the objective function to be evaluated, while $g(\mathbf{p})$ is a *penalty* term, which penalizes the violation of the constraints on the parameters. An example of it is to explicitly enforce non-overlapping constraints in the placement of obstacles.

The CMA-ES algorithm. For the scenarios presented in this analysis, the minimization formulation results in a non-convex problem which is solved with the Covariance Matrix Adaptation Evolutionary Strategy (Hansen and Ostermeier 1996)[1][2].

The CMA-ES algorithm is well suited to this domain: it is easy to implement, it can manage complex objective functions with noise and it is very competitive in converging to an optimal value in lesser iterations. Optimization formulation and algorithm to optimize the architectural elements (pillars) is used from SteerSuite.

[1]For more details see (Hansen and Ostermeier 1996), and http://en.wikipedia.org/wiki/CMA-ES

[2]The initial source code can be found at https://www.lri.fr/∼hansen/cmaesintro.html

In the next sections two sets of experiments are presented to address the three key points mentioned in Section 3.1 using this methodology.

3.4 Experiment 1

This experiment studies the optimal placements of architectural elements (pillars) in environment designs and their impact on crowd flow during evacuation scenarios. It also investigates the effects of using different dynamic crowd simulators defined in Section 3.3.2 on the crowd flow results.

In all the experiments, agents are represented by disks with a radius of 0.2286 meters. This radius value is used as it gives us more realistic results and has no negative effect on the simulation system. Medium density crowds are used in this experiment.

3.4.1 Objective Function

Crowd flow is considered an effective measure in evacuation scenarios. It has been defined in many different ways (Helbing et al. 2007, Johansson et al. 2008). For this experiment, crowd flow metric is defined as the ratio of total agents successfully evacuated $|A_c|$ over average completion time of all the agents t_{avg}:

$$f(\mathbf{p}) = \frac{|A_c|}{t_{avg}}, t_{avg} = \frac{\sum\limits_{a \in A} t_a}{|A|}, \tag{3.2}$$

where t_a is the simulated time that the agent a needed to complete the simulation and $|A|$ indicates the cardinality of set A.

An agent has completed a simulation if the agent reaches its goal location before the simulation terminates, A_c is the set of completed agents.

3.4.2 Benchmarks

Crowd flow patterns are studied using the dynamic crowd simulators mentioned in Section 3.3.2 on a variety of architectural scenarios with different configurations of corridors, pillars and exit doors.

Uni-directional Hallway. The configuration setup of this architectural environment scenario is shown in Figure 3.2. A hundred agents are randomly placed in a $12.5 \times 4\ m^2$ region (blue). Up to 4 architectural elements (pillars) are placed in the optimization region (grey). Each agent has a goal or target location (green) outside of the hallway. The distance between the closed boundaries of the optimization and the crowd regions is $3.5\ m$.

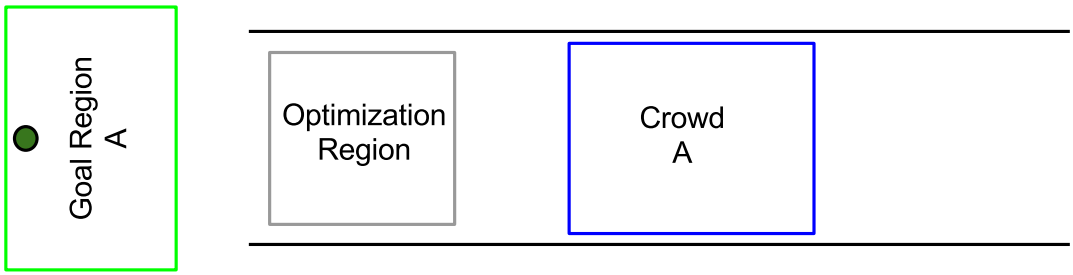


Figure 3.2: Uni-directional hallway scenario.

Bi-directional Hallway. This architectural environment scenario is an extension of the previous one with two sets of crowds, A and B, travelling in opposite directions in

the hallway, Figure 3.3. Each crowd contains 50 agents that are randomly placed in the corresponding blue region of size $6.25 \times 4\ m^2$. Up to 4 architectural elements (pillars) are placed in the $4 \times 4\ m^2$ optimization region (grey). Each group must cross the optimization region to reach its corresponding goal region (green).

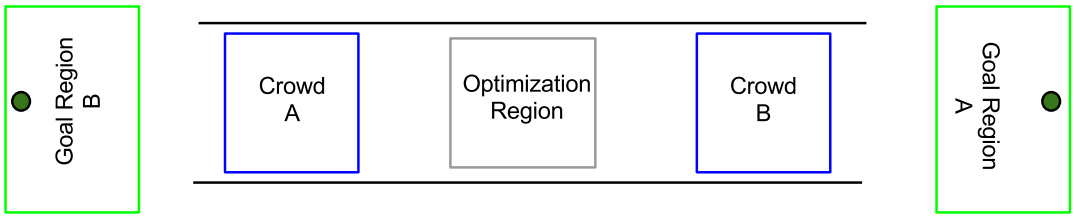


Figure 3.3: Bi-directional-hallway scenario.

Two-way Egress. In this architectural environment scenario, two sets of crowds are travelling from opposite directions in a hallway and must exit from the same door in the middle of the hallway, Figure 3.4. The arrangement of the agents, and the optimization region are identical to those of the previous scenarios. A door of size 1.3716 m is in the middle of the 34 m hallway. The size of the door is in accordance with local standard building codes.

Four-way Hallway. This is an extension of the previous bi-directional hallway architectural environment scenario in all cardinal directions, Figure 3.5. Four sets of crowds of 25 agents each travel from opposite directions in two hallways that share a $4 \times 4\ m^2$ optimization region (grey) in the center. The agents are randomly distributed in their corresponding region (blue) and must reach their goal region (green) across the corresponding hallway.

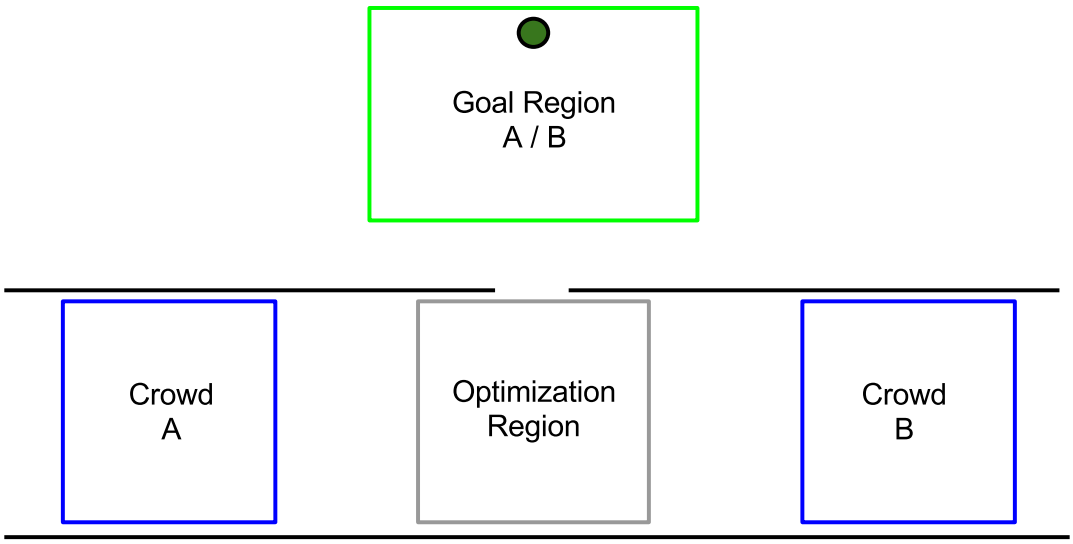


Figure 3.4: Two-way egress scenario.

These architectural scenarios are selected as they are challenging and quite common in evacuation cases.

3.4.3 Results

Figure 3.1 shows qualitative result for **SF** with 3 pillars. Selective placement of pillars significantly improves the crowd flow. Crowd flow values of all three crowd simulators are described in the following subsections:

Uni-directional Hallway. Table 3.1 shows the crowd flow as defined in Equation 3.2 for the optimal placement of one to four architectural elements (pillars) for all four architectural environment scenarios. Crowd flow with no architectural element (zero pillars) is also included for reference. It can be seen from the results that the flow value for the uni-directional hallway scenario improves with the placement of pillars. In particular,

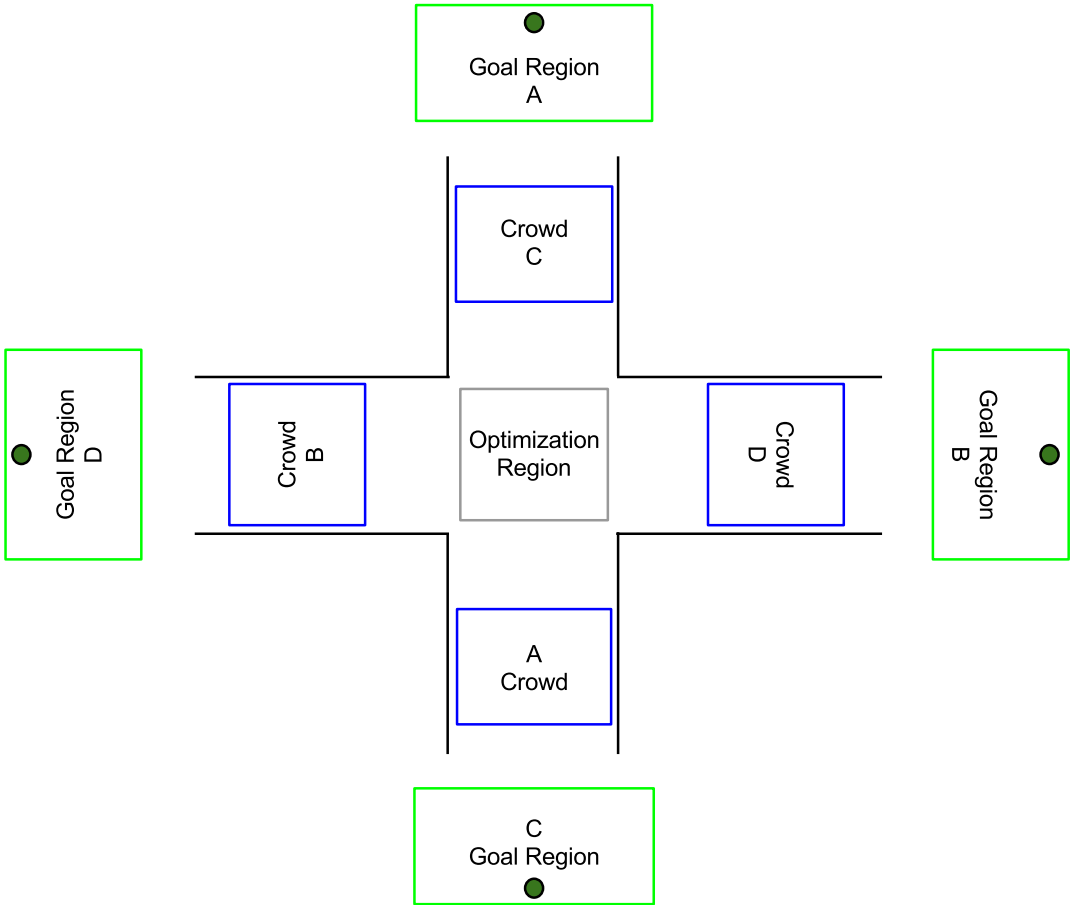


Figure 3.5: Four-way hallway scenario.

placement of three pillars produces the highest crowd flow for all three algorithms.

Bi-directional Hallway. Table 3.1 shows that crowd flow improves in most of the cases for bi-directional architectural scenarios. **SF** and **ORCA** show improved flow even with four architectural elements (pillars). **PPR** shows nearly 30% improvements with the optimal placement of two pillars.

Two-way Egress. Table 3.1 shows that only **ORCA** produces significant improvements in crowd flow with the placement of architectural elements (pillars) in the two-way

Uni-directional hallway					
Algorithm	Pillars				
	0	**1**	**2**	**3**	**4**
ORCA	6.116	6.61	6.601	6.627	6.619
PPR	1.916	2.178	2.169	2.19	2.151
SF	4.425	4.478	4.505	4.886	4.573

Two-way egress					
Algorithm	Pillars				
	0	**1**	**2**	**3**	**4**
ORCA	0.802	1.531	1.586	1.71	1.926
PPR	N/A				
SF	4.292	4.624	4.358	4.477	4.358

Bi-directional hallway					
Algorithm	Pillars				
	0	**1**	**2**	**3**	**4**
ORCA	2.843	3.639	3.698	3.539	3.626
PPR	1.686	2.164	2.221	2.088	2.111
SF	3.339	3.795	3.566	3.648	3.929

Four-way hallway					
Algorithm	Pillars				
	0	**1**	**2**	**3**	**4**
ORCA	3.013	3.789	3.64	3.509	3.626
PPR	N/A				
SF	3.475	3.76	3.842	3.932	3.761

Table 3.1: Crowd flow values, $f(\mathbf{p})$, for all four scenarios with **ORCA**, **SF** and **PPR**.

egress architectural environment scenario. **SF** also produces good crowd flow in all the cases but the improvement difference is not really significant. **PPR** had difficulties in completing this architectural environment scenario realistically and could not finish.

Four-way Hallway. Table 3.1 shows that **PPR** had a very difficult time with this architectural environment scenario. One reason is that the algorithm tends to make agents wait when they are unable to move forward. Because of this behavior, all the four crowds seem to reach a deadlock situation in the center. **SF** and **ORCA** both show improvements in crowd flow with the placement of pillars.

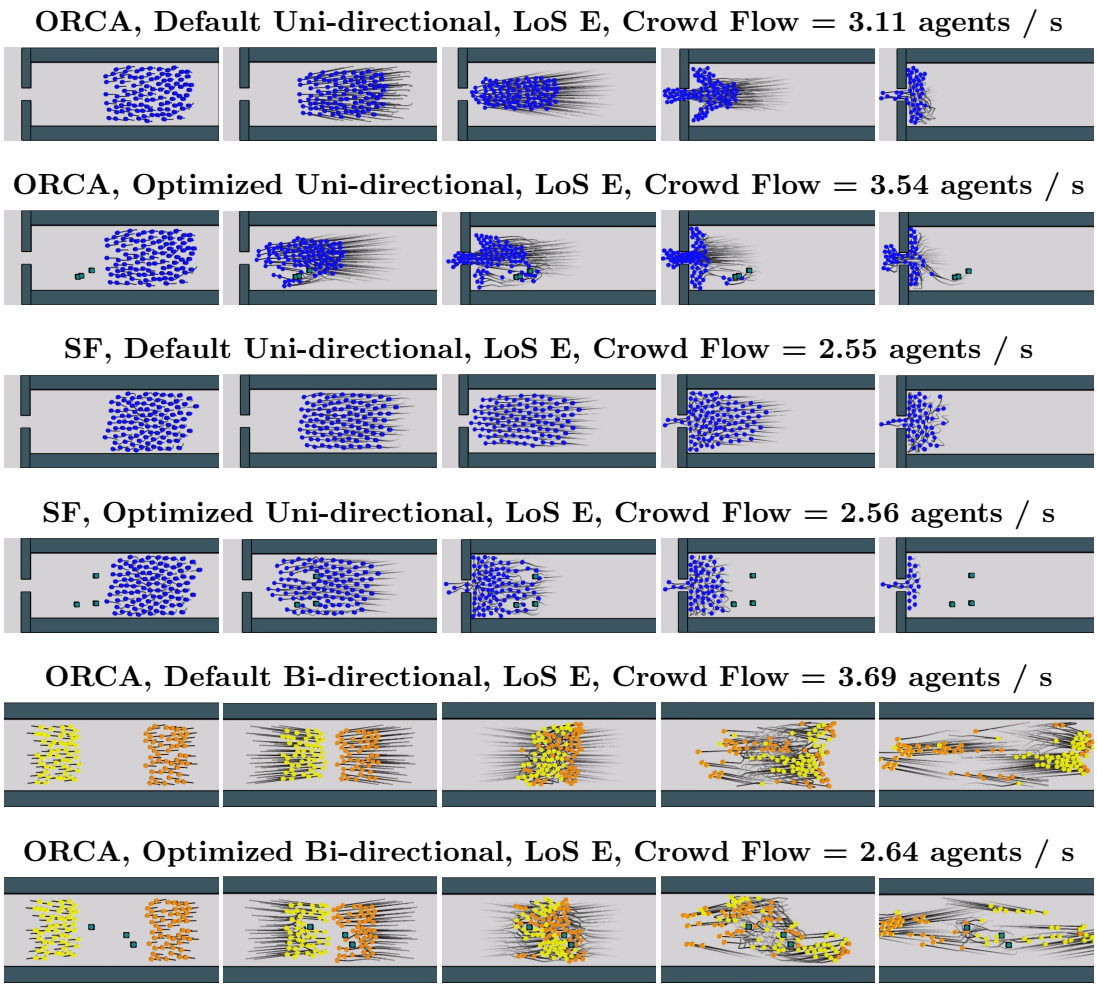


Figure 3.6: Simulation results of **SF** and **ORCA** for uni-directional and bi-directional hallway scenarios, using **LoS** E, for default and optimized versions of the environments. Optimal placement of architectural elements (pillars) help improve the critical density of steering algorithms, thereby increasing the effective **LoS** of the architectural environment. The optimal placement of architectural elements (pillars) is sensitive to the type of crowd simulator, architectural environment benchmark, and **LoS**.

3.5 Experiment 2

Results from Experiment 1 show that the placement of architectural elements (pillars) can greatly affect the crowd flow patterns. The results also reveal that choice of crowd simulation algorithm has significant impact on the flow results. The purpose of this experiment is to investigate flow-density relationships i.e. to determine the effects of different crowd densities on flow patterns during crowd evacuation scenarios.

3.5.1 Level of Service (LoS) of Pedestrian Crowds

Levels of Service (**LoS**) (Fruin 1971) provides a classification (A – E) of the objective **LoS** for pedestrian environments by assigning labels to different crowd densities. A reflects the least dense crowd, whereas E reflects the most dense crowd. These **LoS** classifications are provided for various types of pedestrian environments such as walkways, queues, and stairs. This study explores **LoS** in the walkable areas. Table 3.2 provides the original measures and descriptions used to classify **LoS** for pedestrian walkway environments.

Critical density, ρ_c, is an important concept and defined as a point at which increasing crowd density no longer increases the crowd flow. Crowd flow reaches the peak at critical density point and reduces afterwards.

This work studies the flow-density relationships for **ORCA**, **SF**, and **PPR** across **LoS** A – E using same optimization formulation that is being used in Experiment 1.

LoS	**Volume (ped/min)**	**Average Area (m^2/ped)**	**Density (ped/m^2)**	**Description**
A	16 or less	≥ 3.3	≤ 0.30	Threshold of free flow. Convenient passing, conflicts avoidable.
B	16 - 23	2.3 - 3.3	0.43 - 0.30	Minor conflicts, passing and speed restrictions.
C	23 - 33	1.4 - 2.3	0.71 - 0.43	Crowded but fluid movement, passing restricted, cross and reverse flows difficult.
D	33 - 43	0.9 - 1.4	1.11 - 0.71	Significant conflicts, passing and speed restrictions, intermittent shuffling.
E	43 - 56	0.5 - 0.9	2 - 1.11	Shuffling walk, reverse, passing, and cross flows very difficult; intermittent stopping.
F	*Variable up to max*	≤ 0.5	≥ 2	Critical density, flow sporadic, frequent stops, contacts with others.

Table 3.2: **LoS** classifications and their relevant data and descriptions, adapted from Fruin (1971).

3.5.2 Objective Function

This experiment also uses crowd flow measure as an objective criterion to evaluate architectural environments during evacuation scenarios. It is defined as the rate at which all the agents reach their target or goal positions:

$$f(\mathbf{p}) = \frac{|A_c|}{t_c}, t_c = t_l - t_0 \tag{3.3}$$

where t_0 and t_l are the completion times for the first and last agents to reach their goals respectively and $|A|$ indicates the cardinality of set A. A_c is the set of agents which have successfully reached to their targets within given simulation time.

3.5.3 Benchmarks

The following architectural scenarios are used in the experiment:

Uni-directional Hallway. The configuration setup of this architectural environment scenario is shown in Figure 3.7. A hundred agents are randomly placed in regions to accommodate crowd densities across all levels (A–E). These regions correspond to **LoS**: A ($6m \times 83.33m$), B ($6m \times 41.66m$), C ($6m \times 27.77m$), D ($6m \times 18.51m$) and E ($6m \times 8.33m$) as shown in Figure 3.7. Up to 4 architectural elements (pillars) are placed in the optimization region (grey). Each agent has a goal or target location (green) outside of the hallway.

Bi-directional Hallway. This architectural environment scenario is an extension of the previous one with two sets of crowds, A and B, travelling in opposite directions in the hallway, Figure 3.8. Each crowd contains 50 agents that are randomly placed in regions

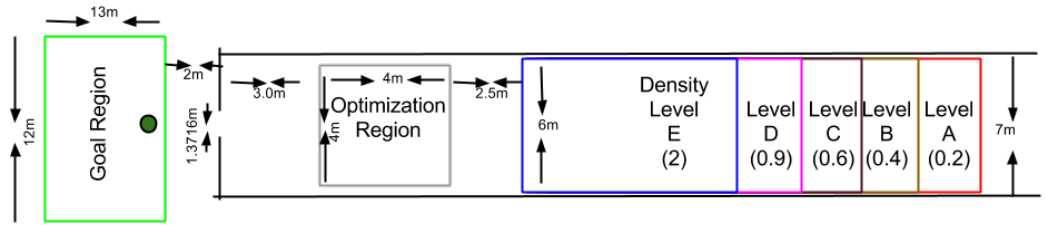


Figure 3.7: **LoS**: Uni-directional hallway scenario. A hundred agents are randomly placed in regions (A–E) to accommodate crowd densities across all levels. Grey is the optimization region where architectural elements (pillars) are placed. Green region is the goal or target area for each agent.

to accommodate crowd densities across our examined **LoS**. These regions correspond to **LoS**: A ($6m \times 41.66m$), B ($6m \times 20.83m$), C ($6m \times 13.88m$), D ($6m \times 9.25m$) and E ($6m \times 4.16m$). Up to 4 architectural elements (pillars) are placed in the 4×4 m^2 optimization region (grey). Each group must cross the optimization region to reach its corresponding goal region (green).

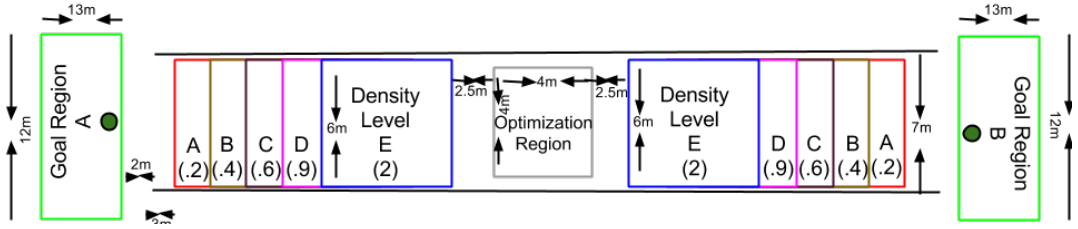


Figure 3.8: **LoS**: Bi-directional-hallway scenario. Fifty agents are randomly placed in regions (A–E) at each side of the hallway to accommodate crowd densities across all levels. Grey is the optimization region in the middle where architectural elements (pillars) are placed. Each agent has a goal or target area (Green) at the other side of the hallway.

3.5.4 Results

Figure 3.6 shows qualitative results for **SF** and **ORCA** with 3 pillars for **LoS** E. Crowd flow values of all three crowd simulators are described in the following subsections.

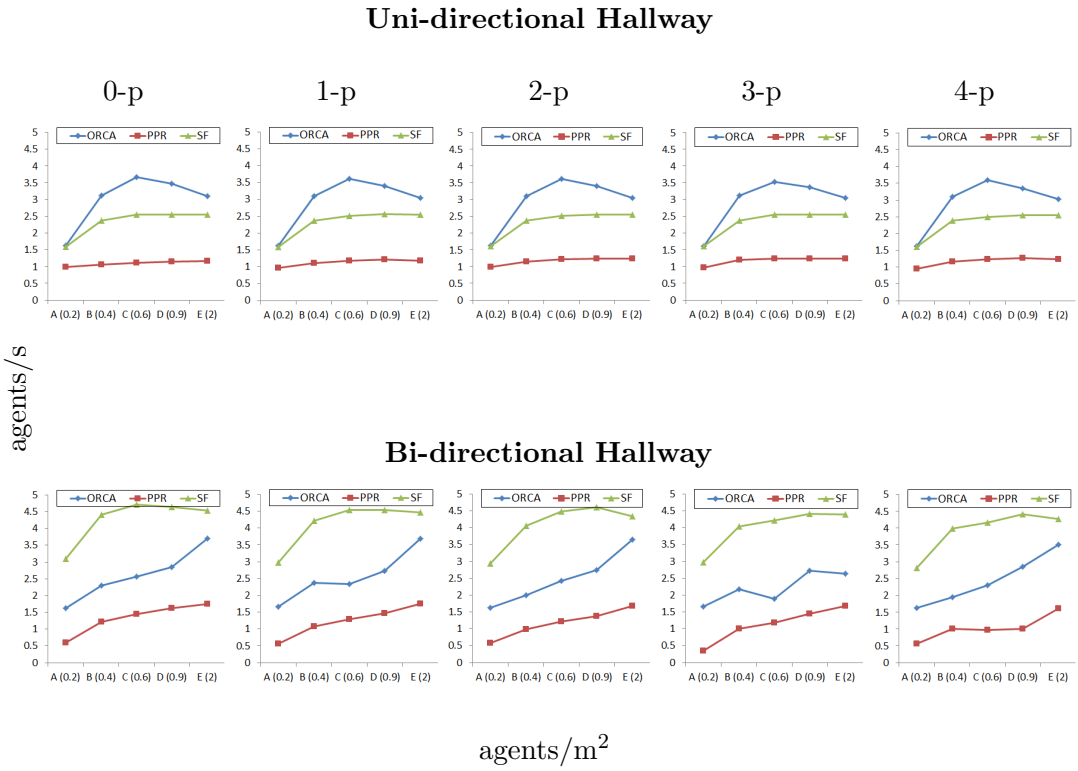


Figure 3.9: The optimal crowd flow values $f(\mathbf{p})$, across all density levels **LoS** (A–E), for **ORCA**, **SF**, and **PPR**, in the uni-directional and bi-directional hallways, where n-p means n number of pillars.

Uni-directional Hallway. Table 3.3 shows the crowd flow as defined in Equation 3.3 for the optimal placement of one to four architectural elements (pillars) for both of the architectural environment scenarios with **LoS** A and E. Crowd flow with no architectural

Uni-directional hallway (**LoS** A)					
Algorithm	Pillars				
	0	**1**	**2**	**3**	**4**
ORCA	1.61	1.62	1.62	1.61	1.63
PPR	0.98	0.98	0.98	0.98	0.96
SF	1.59	1.59	1.61	1.61	1.60

Bi-directional hallway (**LoS** A)					
Algorithm	Pillars				
	0	**1**	**2**	**3**	**4**
ORCA	1.62	1.67	1.63	1.67	1.66
PPR	N/A	0.56	0.55	0.56	0.48
SF	N/A	2.99	2.96	2.97	2.81

Uni-directional hallway (**LoS** B)					
Algorithm	Pillars				
	0	**1**	**2**	**3**	**4**
ORCA	3.11	3.11	3.11	3.11	3.10
PPR	1.05	1.12	1.14	1.21	1.17
SF	2.36	2.37	2.38	2.37	2.38

Bi-directional hallway (**LoS** B)					
Algorithm	Pillars				
	0	**1**	**2**	**3**	**4**
ORCA	2.30	2.37	2.01	2.15	2.02
PPR	N/A	0.99	0.97	1.01	0.78
SF	N/A	4.26	4.05	4.09	3.96

Uni-directional hallway (**LoS** C)					
Algorithm	Pillars				
	0	**1**	**2**	**3**	**4**
ORCA	3.67	3.61	3.44	3.52	3.60
PPR	1.10	1.19	1.22	1.24	1.24
SF	2.54	2.52	2.50	2.54	2.49

Bi-directional hallway (**LoS** C)					
Algorithm	Pillars				
	0	**1**	**2**	**3**	**4**
ORCA	2.56	2.25	1.89	2.19	2.33
PPR	N/A	0.96	1.18	1.21	1.29
SF	N/A	4.16	4.28	4.47	4.53

Uni-directional hallway (**LoS** D)					
Algorithm	Pillars				
	0	**1**	**2**	**3**	**4**
ORCA	3.47	3.40	3.72	3.36	3.33
PPR	1.14	1.28	1.23	1.27	1.28
SF	2.55	2.55	2.55	2.56	2.54

Bi-directional hallway (**LoS** D)					
Algorithm	Pillars				
	0	**1**	**2**	**3**	**4**
ORCA	2.84	2.73	2.75	2.73	2.85
PPR	N/A	1.53	1.48	1.49	1.50
SF	N/A	4.60	4.60	4.44	4.37

Uni-directional hallway (**LoS** E)					
Algorithm	Pillars				
	0	**1**	**2**	**3**	**4**
ORCA	3.11	3.05	3.08	3.05	3.54
PPR	1.17	1.18	1.24	1.23	1.24
SF	2.55	2.55	2.55	2.57	2.55

Bi-directional hallway (**LoS** E)					
Algorithm	Pillars				
	0	**1**	**2**	**3**	**4**
ORCA	3.69	4.47	3.64	2.63	3.50
PPR	N/A	1.75	1.77	1.63	1.83
SF	N/A	4.47	4.31	4.38	4.22

Table 3.3: Crowd flow values, $f(\mathbf{p})$, with all **LoS** using **ORCA**, **SF** and **PPR**.

element (zero pillars) is also included for reference. All three crowd simulators performed better at high density conditions (**LoS** E), as compared to low density conditions (**LoS** A). In general, **ORCA** outperforms both **SF** and **PPR** with 4 pillars and produces consistently higher flow.

In Figure 3.9, the top row shows the flow-density relationships for **ORCA**, **SF** and **PPR** for 0 – 4 architectural elements (pillars), in the uni-directional architectural environment scenario. **ORCA** consistently produces the greater flow – reaching the critical density value around **LoS** C with $\rho_c = 0.6$ agents/m^2. **SF** and **PPR** do not produce a prominent critical density but we can still see some increase near **LoS** C and the curve becomes smooth and gradually increases after this point.

Bi-directional Hallway. Table 3.3 shows that crowd flow improves in most of the cases for bi-directional architectural scenarios. All three crowd simulators performed better at high density conditions (**LoS** E), as compared to low density conditions (**LoS** A). In general, **SF** outperforms both **ORCA** and **PPR** with 2 pillars and produces consistently higher flow.

In Figure 3.9, the bottom row shows the flow-density relationships for **ORCA**, **SF** and **PPR** for 0 – 4 architectural elements (pillars), in the bi-directional architectural environment scenario. **SF** produces the highest flow rate for all 4 pillars, and also reaches critical density around **LoS** D. It is also interesting to observe that **ORCA** shows an unusual rise in flow at **LoS** D, before dropping again, for 3 pillars. In a sense, **PPR** also reached at a critical density between **LoS** B and C but the curve rapidly increased

afterwards.

3.6 Summary

This chapter presented a methodology to systematically study the impacts of configuration of an architectural environment on crowd flow. Results reveal many interesting insights, highlighting the sensitivity of optimal architectural environment configurations on the choice of crowd simulators. It is observed that in most of the scenarios, the optimal number of architectural elements (pillars) was found to be 3 and except uni-directional hallway benchmark, **SF** outperforms among all three crowd simulators.

Another observation is that the crowd density, **LoS**, afforded by a particular architectural environment is sensitive to how the crowd behaves, and how the architectural environment is configured. The critical density of crowd simulators effectively increases due to the optimal placement of 1–4 architectural elements (pillars) in the architectural environments, thereby increasing the crowd flow at greater densities (near **LoS** D and E). However, this behavior is not uniform across all crowd simulators and architectural environment scenarios.

Chapter 4

CODE: Crowd Optimized Design of Environments

This chapter describes the design and development of an interactive system for doing dynamic analysis of the architectural elements using crowd simulations. The system is also featured to optimize these architectural elements using an interactive optimization technique and then its results are provided as feedback in terms of aggregate statistics and heat maps to the users. A user study is also conducted to validate the performance of the system.

4.1 Introduction

In an arbitrary architectural environment design it is difficult to explore the movement or flow of people. It is practically impossible to exhaustively search the whole architectural environment configuration space while taking into account the subtle and aggregate effects

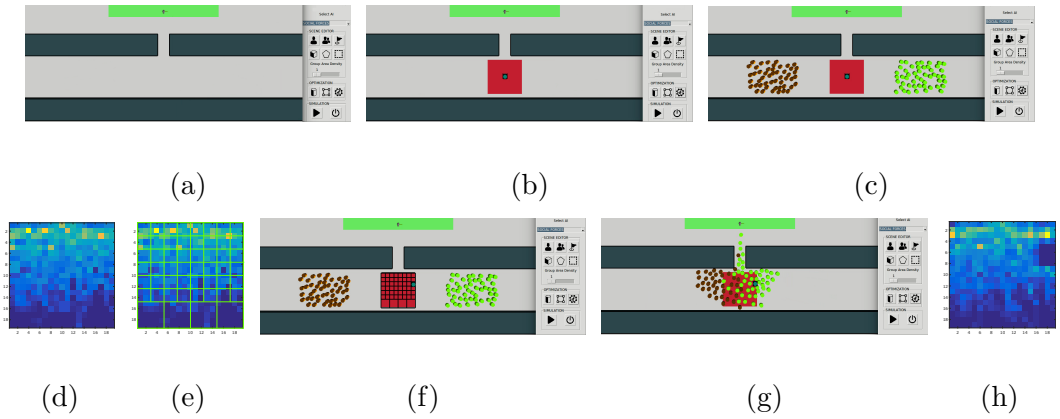


Figure 4.1: (a) The initial scenario with walls and target location. (b) Addition of the optimization region and initial best guess for the pillar location. (c) Addition of crowds to the scenario. (d) A heat map of the flow for the scenario in (c). (e) The computed `ISAB` optimal pillar location. (f) The `AMR` visualized in the scenario. (g) The new crowd simulation with increased flow and new flow as shown in (h).

on crowd flow. This also takes a lot of computation and human labor which makes it even more impractical.

This chapter presents `CODE`: a crowd-aware computational system for designing architectural environments (e.g., building floor plans). This system analyzes how the newly added architectural elements (e.g., pillars, obstacles or doorways) are going to impact the crowd flow. This is done by using the current-generation dynamic crowd simulators. The simulation is used to dynamically analyze the architectural designs and then its results are provided as feedback in terms of aggregate statistics and heat maps to the users. Moreover, this system may be used to automatically optimize the placement of

architectural elements with respect to maximizing the flow. These elements are placed in the architectural environment in addition to any constraints custom defined by the user. CODE provides an interactive architectural environment for architects and designers to iteratively refine their original design layout to accommodate the dynamic properties of crowd simulations quickly. CODE is designed with modularity and flexibility in mind, thus, it allows users to build architectural environments, select from different dynamic crowd simulators; and specify varying crowd configurations. Figure 4.1 shows an overview and the process within the CODE system.

4.2 Scenario Definition and Configuration

CODE allows architects and designers to develop scenarios with specific architectural layouts and varying crowd configurations. A scenario is defined as $\mathbf{S} = \langle \mathbf{E}, \mathbf{C} \rangle$ where $\mathbf{E}$ is the specification of the environment and $\mathbf{C}$ is the crowd. A crowd is more specifically defined as $\mathbf{C} = \langle \mathbf{A}, \mathbf{G}, \mathbf{P}' \rangle$ where $\mathbf{A}$ is the collection of agents, $\mathbf{G}$ are their desired goals, and $\mathbf{P}'$ are the parameters of the dynamic crowd simulator. Each architectural element $\mathbf{e} \in \mathbf{E}$ is defined as $\mathbf{e} = \langle \mathbf{p}', \mathbf{b}, s, \mathbf{r} \rangle$ where $\mathbf{p}'$ is its position, $\mathbf{b}$ is the bounds, and s is the element shape. Elements may be rectangular e.g., a wall, or cylindrical e.g., a pillar etc. Architects or designers may also specify a region $\mathbf{r}$ which indicates that the architectural elements $\mathbf{e}$ can be optimized by automatic placement anywhere within the region $\mathbf{r}$. For any immovable element, designer may choose not to define any region. To differentiate movable, $\mathbf{E}_p$, and immovable, $\mathbf{E}_f$, elements, the scenario definition can be redefined as

$\mathbf{S} = \langle \mathbf{E}_f \cup \mathbf{E}_p, \mathbf{C} \rangle$.

4.2.1 Scenario Design

Environments can be designed quickly using CODE by adding and deleting objects in the scenario and including constraint regions for optimizable architectural elements. Moreover, designers may indicate crowd configurations by adding agents in the scenario, mentioning their desired positions, selecting the crowd simulator algorithm, and may also define the behavior of the crowd.

4.2.2 Dynamic Crowd Simulator

At present the three current generation dynamic crowd simulators (as defined in Section 3.3.2) are supported in CODE. For each algorithm the default parameters, as suggested by the algorithm's developers, are used.

4.2.3 Optimization Formulation

Using a user designed scenario $\mathbf{S} = \langle \mathbf{E}_f \cup \mathbf{E}_p, \mathbf{C} \rangle$, the flow of the crowd is optimized by automatically generating the optimal configuration for each architectural element of the architectural environment $\mathbf{e} \in \mathbf{E}_p$. This is achieved by placing the architectural elements on a valid region to meet user constraints. So the optimization problem is formulated as:

$$\mathbf{E}_p^{opt} = \arg\max_{\mathbf{E}_p} \sum_{\mathbf{C}\in\mathbf{C}} f_{\mathbf{C}}$$
$$\text{s.t.} \quad \mathbf{p'_e} \in \mathbf{r_e} \;\; \forall \mathbf{e} \in \mathbf{E}_p. \tag{4.1}$$

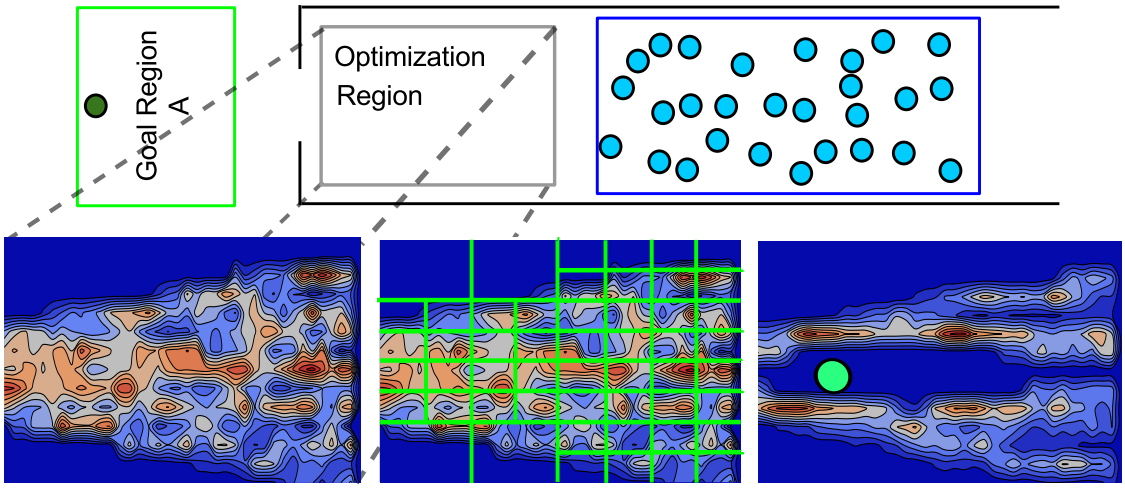
Crowd Flow. Simulating a crowd $\mathbf{C}$, where $\mathbf{A_c} \subseteq \mathbf{A}$ reach their destination $\mathbf{G}$, within some conservative maximum time threshold t_{sim}. Let the last agent t_a^f reaches its destination in time $\mathbf{A_c}$. So $\mathbf{A_{nc}} = \mathbf{A} \setminus \mathbf{A_c}$ are the agents who are unable to complete their goals. This gives the crowd flow, $f_{\mathbf{C}}$, as follows:

$$f_{\mathbf{C}} = \frac{|\mathbf{A_c}|}{t_a^f} \tag{4.2}$$

This crowd flow formulation works well for degenerate cases when $|\mathbf{A_{nc}}| > 0, t_a^f = t_{sim}$.

4.3 Interactive Optimization using Adaptive Mesh Refinement

An Adaptive Mesh Refinement (AMR) approach is used to discretize and adaptively sample the optimization search space for optimizing the building layouts which is typically a non-convex and continuous problem. Most commonly the turbulent hydrodynamics (Berger and Colella 1989) simulation uses the AMR technique which is used in `CODE` as an isomorphic analogy to dense crowd simulations in complex and realistic architectural environments. The core idea is to target some areas of interest defined by some heuristics and to discretize the sampling space of these areas at finer granularity. To include and/or enhance the information captured by the underlying data, the heuristic guided mesh adaptation is used at each iteration. This also allows automatic pruning



(a) (b) (c)

Figure 4.2: (a) The crowd flow density in the optimization region. (b) The AMR for (a). (c) The best pillar position according to the AMR sampling, and new crowd flow.

of unimportant areas. Crowd density as the granularity heuristic is used based on the results from previous studies (Haworth et al. 2015). To produce more accurate solutions, the subdivision level of the mesh is increased which produces finer granularity sampling but also results in high computation time and vice versa.

Figure 4.2 shows some results of the AMR algorithm. The pseudocode of this algorithm, Iterative Selection AMR (ISA), is provided in Algorithm 1 and its implementation is used from SteerSuite. To optimize the flow of a crowd, a set of optimizable architectural elements, $\mathbf{e} \in \mathbf{E_S}$, need to be placed on valid locations within the specified region $\mathbf{r}$. ISA works by placing one architectural element $\mathbf{E}_p$ per iteration. To understand how

Algorithm 1: `ISA`

input : $\mathbf{E}_p$ - *optimizable elements*

$\mathbf{E}_f$ - *fixed elements*

$\mathbf{S} \leftarrow \langle \mathbf{E}_f \cup \mathbf{E}_p, \mathbf{C} \rangle$- *scenario definition*

foreach $i \in |\mathbf{E}_p|$ **do**

  $\langle H, f^d_{\mathbf{C}} \rangle \leftarrow$ **ComputeHistogram(S)**

  $M \leftarrow$ **SubdivideMesh(**H**)**

  $\mathbf{F} \leftarrow -1$

  foreach *Node* $n \in M$ **do**

    $\mathbf{S}' \leftarrow$ **PlaceElement(**$\mathbf{S}$, $\mathbf{E}_p[i]$, n**)**

    $\mathbf{F}\ (n) \leftarrow$ **Simulate(**$\mathbf{S}'$**)**

  end

  $\mathbf{S} \leftarrow$ **PlaceElement(**$\mathbf{S}$, $\mathbf{E}_p[i]$, *index(***max(F)**,**F***)***)**

end

a crowd flows within current scenario set by the user we pre-compute a 2D histogram that describes the crowd density distribution over the area. It is generated by simulating crowd through the user-authored scenario without placing the current element. This histogram, H, and a default flow value, $f^d_{\mathbf{C}}$, is produced at each iteration based on scenario **S** and is done by calling **ComputeHistogram** function. Histogram is then used to form a space-partitioning structure, or mesh, M. The mesh M contains n nodes which are used as sample points by the **PlaceElement** function which, moves and reinserts the current element while invalidating intersecting placements. Then the simulation is executed and it stored the **F** (n), the flow value at the current node. For this current iteration the optimal selection is done with the maximum index value of **F**.

Backtracking. There is a chance that the `ISA` algorithm may get stuck in local minima after the initial bad placement of the current architectural element and there was no way of going back to select that element again. This is overcome by introducing backtracking in Algorithm 1. It allows the algorithm to go back to the previous iteration and selet the placement of architectural element again. Backtracking code triggers only when the crowd flow value of the current iteration is less than the previous one and if the **rand** returns a value which is less than backtracking probability, ϵ_p (default set to 0.2 and is tunable). The pseudocode of `ISA` with backtracking is provided in Algorithm 2 and is called Iterative Selection AMR with Backtracking (`ISAB`).

Algorithm 2: `ISAB`

input : $\mathbf{E}_p$ - *optimizable elements*

$\mathbf{E}_f$ - *fixed elements*

ϵ_p- *backtracking probability*

$\mathbf{S} \leftarrow \langle \mathbf{E}_f \cup \mathbf{E}_p, \mathbf{C} \rangle$- *scenario definition*

foreach $i \in |\mathbf{E}_p|$ **do**

  $\langle H, f^d_{\mathbf{C}} \rangle \leftarrow$ **ComputeHistogram(S)**

  $M \leftarrow$ **SubdivideMesh(**H**)**

  $\mathbf{F} \leftarrow -1$

  foreach *Node* $n \in M$ **do**

    $\mathbf{S}' \leftarrow$ **PlaceElement(**$\mathbf{S}$, $\mathbf{E}_p[i]$, n**)**

    $\mathbf{F}\ (n) \leftarrow$ **Simulate(**$\mathbf{S}'$**)**

  end

  if max(F) $< f^d_{\mathbf{C}}$ *&* **rand()** $< \epsilon_p$ **then**

    $i \leftarrow i - 1$

  else

    $\mathbf{S} \leftarrow$ **PlaceElement(**$\mathbf{S}$, $\mathbf{E}_p[i]$, *index(***max(F)**,$\mathbf{F}$*)***)**

  end

end

4.4 Experimental Setup

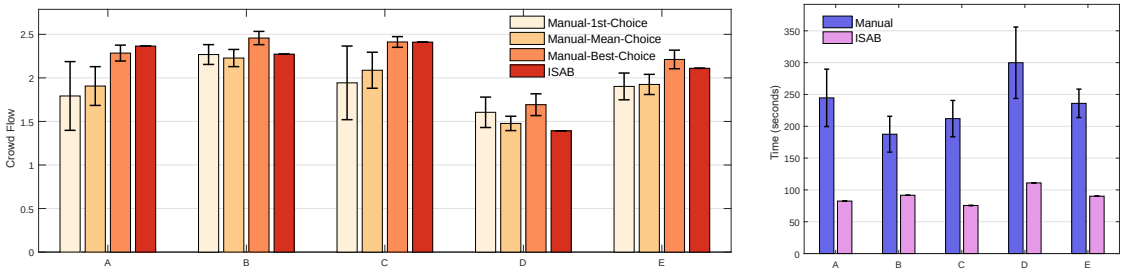
Experiments were conducted to validate the system and compare the crowd flow and pillar placement results with traditional human-created designs (see Section 4.5).
Four scenarios were designed, similar to Section 3.4.2, to place 1 and 2 architectural elements. All the scenarios were simulated 100 times with randomly chosen uniformly distributed crowd agents of radius 0.2286m. These scenarios are uni-directional hallway egress, bi-directional hallway, bi-directional hallway side egress and four-way hallway side egress. The crowd flow performance is derived from these simulations.

4.5 User-Study

Two user studies were conducted using placements involving 1 and 2 pillars. These studies are used to assess the influence of the automated `ISAB` technique on the authoring performance as well as to measure the usability of the overall system. Four optimal scenarios are designed by each participant. These scenarios are designed using two systems:

- `CODE` as a manual design system.
- `CODE` with the automated ISAB technique.

Based on selected dynamic crowd simulator, both the systems provide feedback on the placement of architectural elements. `CODE` is used to design the scenarios manually, so the participants can place the architectural elements and then run the crowd simulation. However, at this point `ISAB` technique is not allowed to be used. The scenarios are



(a) (b)

Figure 4.3: Average flow of Manual and `ISAB` assisted single pillar placement with 95% confidence intervals. The labels A to E represent scenarios. **A** is the uni-directional hallway scenario, **B** is the bi-directional hallway scenario, **C** is the bi-directional hallway side egress scenario, **D** is the four-way hallway and **E** is the average of all scenarios. (a) Comparison of the first, best, and mean manual choices with `ISAB` selections. (b) Time of completion of user average time and `ISAB`.

designed with predefined agents and goals. The participants were asked to perform the tasks using the same Desktop PC (Ubuntu 14.04, 12GB RAM, AMD A 10- 7800 Radeon R7, 12 Compute Cores, 3.5GHz). Participants were given training on how the system works, about different interface components and the feedback feature, pillars placement and crowd simulation.

4.5.1 Evaluation Metrics

Independent variables. The primary independent variables are the authoring methods. These are the manual pillar placement approach and the `ISAB` automated technique.

Dependent variables. `CODE` captures the following metrics:

- The final crowd flow rate given the user's choice of pillar placements for each scenario and method.
- The time in seconds t_a, needed to author a scenario, for each method.

4.5.2 1-Pillar Placement

For the four scenarios from Section 4.4, each participant is asked to choose an optimal placement for a single pillar with respect to the crowd flow.

Results. User study results with 1-pillar placement based on T-test are shown in Figure 4.3. The bars show the average results, and the error bars show the 95% confidence intervals. Three different cases are analyzed to compare the pillar placements of the participants with the `ISAB` assisted method. These three cases incorporate the first, average and best selection. It is worth noting that a 0.5 unit difference in average crowd flow is equal to approximately 30 more persons per minute, which results in a significant improvement for the evacuation tasks.

Figure 4.3a reveals that the results of the simulation with `ISAB` average flow is significantly better in almost all scenarios as compared to the participant's first selection. For the bi-directional hallway scenario, the performance of `ISAB` technique is as good as Manual and for the four-way hallway scenario the performance of the `ISAB` is slightly lower than Manual with statistically insignificant difference. Moreover, `ISAB` performs significantly better than the average performance of Manual placements, i.e. the average flow achieved

by the user over three trials. Moreover, the best performance of each user i.e., the maximum flow over the three trials in not much different from the ISAB in almost all the scenarios. On the other hand the average completion time for ISAB is found to be less than that of the manual pillar placements, shown in Figure 4.3b.

The results clearly states that the use of ISAB may improve average crowd flow in less time than humans and in some scenarios, Manual may perform better than ISAB but on average ISAB outperforms Manual performance. These results encouraged to continue the experiment with multiple pillars scenarios.

4.5.3 2-Pillars Placement

In a uni-directional scenario, each user is asked to optimally place two pillars with respect to crowd flow.

Results. Figure 4.4 shows the results of the experiment with two pillars. In this case, the participants have to struggle and find it more difficult to optimally place two pillars in the given scenario as compared to ISAB. With multiple placement decisions, participants face more difficulties in finding the optimal positions. Figure 4.5 illustrates the placement of 2 pillars by the user study participants and ISAB using **SF** (dynamic crowd simulator) in the uni-directional benchmark. Results show that ISAB clearly outperforms Manual human selection for the placement of two pillars.

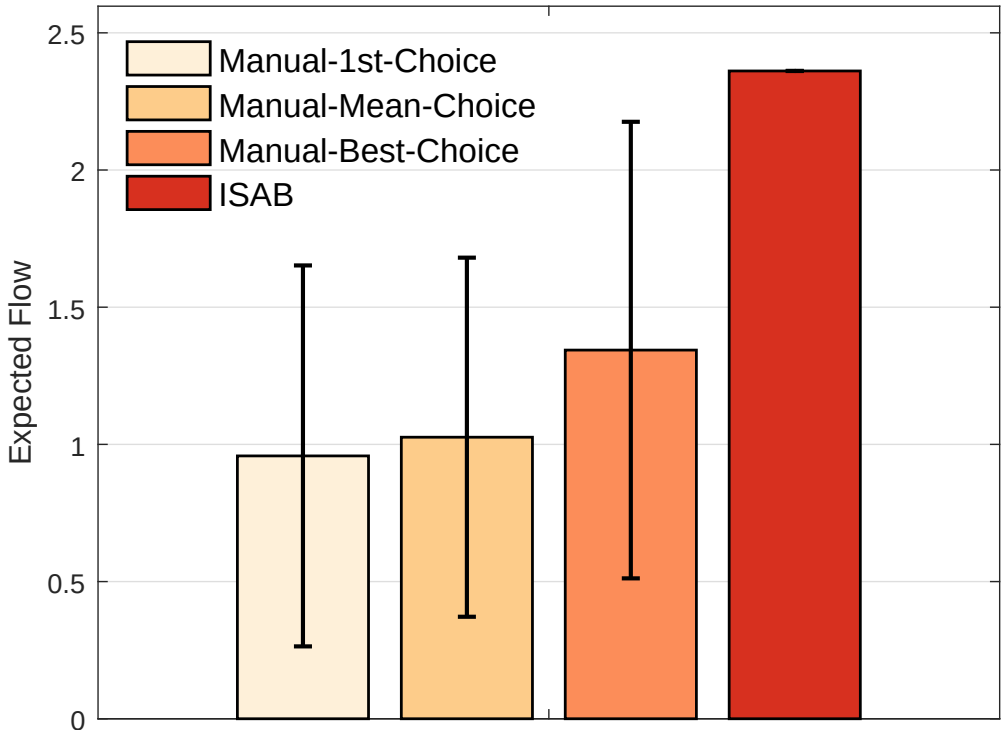


Figure 4.4: Average flow of Manual and `ISAB` assisted two pillar placements in uni-directional scenario, with 95% confidence intervals.

4.5.4 System Usability

Participants of the user-studies were provided with a survey on the usability and efficacy of `CODE`. The SUS score of `CODE` is 87.51, which indicates that even novice users were able to use this system with minimal training.

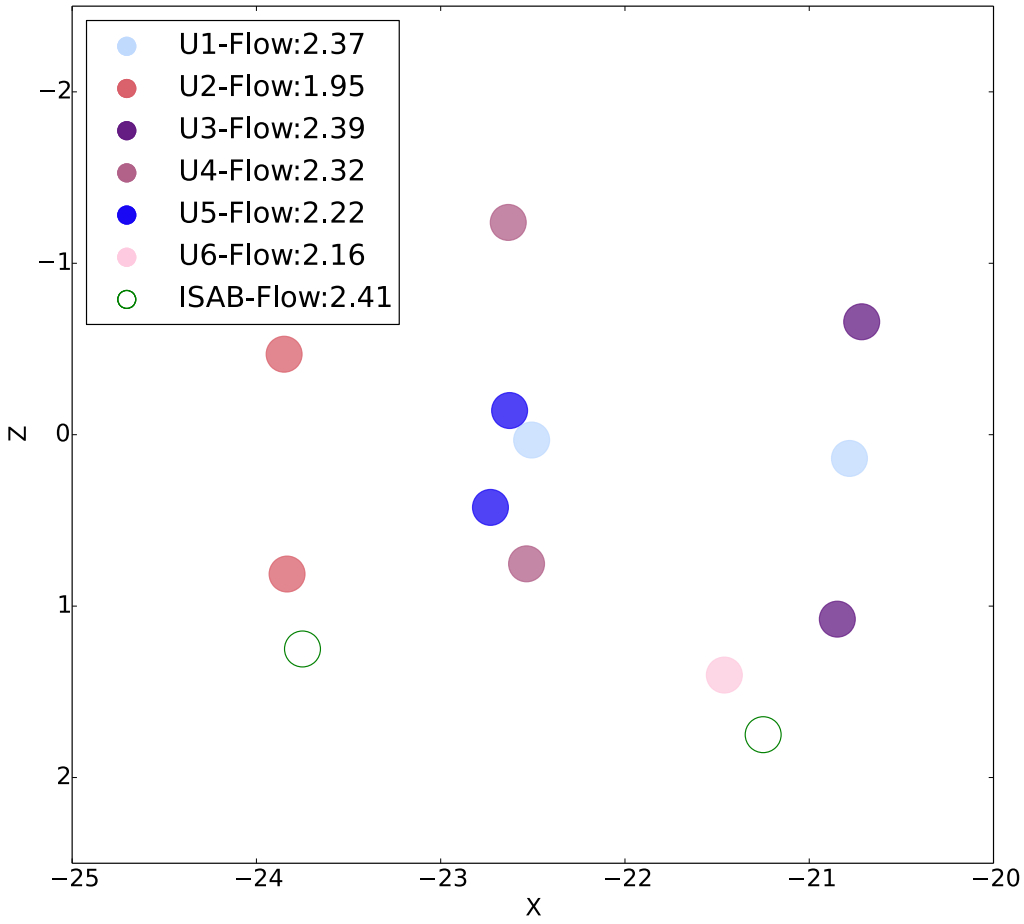


Figure 4.5: Manual and ISAB placements of 2 pillars for **SF** in the uni-directional hallway architectural environment. Crowd is moving from right-to-left.

4.6 Summary

This chapter presented CODE: a crowd-aware computational design system for designing architectural building layouts that also integrates dynamic analysis of the architectural environment designs using crowd simulation, and optimization of architectural elements (pillars). CODE is designed to be used as an aid to people involved with building designs

and layouts. These may be architects, fire marshals etc. The experiments were done using 1 and 2 pillar placements in the architectural environments. From the results of the experiment, it is clear that as building architectures become more complex, interactive automated systems like `CODE` can be of great assistance.

Chapter 5

μDOME: User-in-the Loop Diversity Optimization of Environments

This chapter describes the design and development of an interactive system for doing static analysis of the architectural elements in an environment design using space-syntax measures. User may also optimize these architectural elements using a multi-objective diversity optimization technique that uses spatial measures grounded in space-syntax analysis to quantify how humans interact with the architectural environments. It allows to explore much larger and complex architectural environments as compared to `CODE` which is presented in the previous chapter. The system is evaluated with a user study.

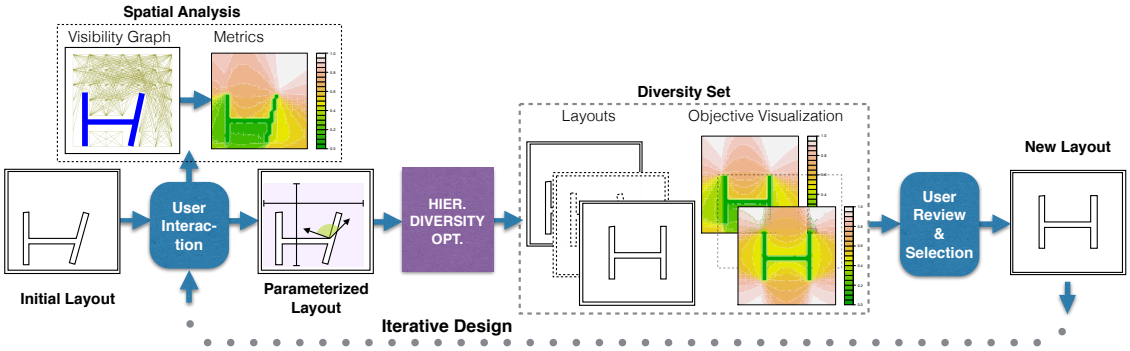


Figure 5.1: μDOME Framework Overview. Given an initial layout, the user selects the regions over which spatial analysis metrics can be computed and visualized, and the attributes (position, orientation) of elements in the design that can be adjusted in the next step. A multi-objective hierarchical diversity optimization produces a set of diverse near optimal solutions, from which the user may select one and repeat the process as desired.

5.1 Introduction

A variety of constraints need to be considered while designing a complex building architecture. A number of diverse designs can be used to address these constraints which may include but is not limited to aesthetic, safety, geometric, accessibility, clearance etc. A rich but highly constrained design space should be fully explored by the designer or architect to produce a comprehensive building design. Architects may use computer-aided design (CAD) tools to get useful analytical data which may be then used to achieve the optimal architectural designs.

To maintain the balance between automation and human control, this chapter pro-

poses a user-in-the-loop automated design exploration system, μDOME. This system aids the decision-making of architects by effectively exploring, analyzing and filtering the space of architectural environment layouts with the use of computer-aided design.

5.1.1 Framework Overview

Figure 5.1 shows the prototype framework overview of μDOME, for user-in-the loop diversity optimization and static analysis of the architectural environment designs. μDOME has an iterative process to refine upon the architectural designs:

Architectural Environment Parameterization and Constraint Definition. Architectural environment design begins with the architectural graph, G_A, which may also be considered as the environment's initial layout. A user then identifies any individual or group of architectural elements like junctions, pillars, hallways, walls etc. All the architectural elements are then grouped together. In the next step, the user decides on the element's attributes, their values and bounds. These attributes represent a user defined parameterization of the architecture graph, G_A, which together with the associated bounds, model the space of all admissible configurations of the architectural environment.

Spatial Analysis. G_V is the visibility graph of an architectural environment and is computed based on the discretization of a G_A, using a grid or an equivalent sampling structure. G_V makes it possible to compute the measures for quantitative analysis of an architectural environment including visibility, accessibility, clearance and order, using the space-syntax analysis (Bafna 2003). On some specific regions of interest or over the

whole environment, the user can compute and visualize these measures as well as the following stages of the process. Implementation of the spatial analysis metrics are primarily contributions of Mahyar Khayatkhoei and is included in the chapter for completeness.

Diversity Optimization. A multi-objective optimization problem for the architectural environment configuration can be formed using designer constraints, architectural environment parameters, and spatial analysis. A diversity measure is also added to the objective to get a set of solutions with near optimal and maximum diversity. Formulation and implementation of this optimization framework is primarily a contribution of Glen Berseth and is included in the chapter for completeness.

User-in-the Loop Iterative Design. The user can choose one or more of the provided diversity optimization results as the starting design for forthcoming iterations of the design process.

5.2 Architectural Environment Parameterization

5.2.1 Architectural Graph

An architectural graph $G_A = \langle N, E \rangle$ is use to represent a structure or building with a particular arrangement of architectural elements like walls and their interconnections. The $G_A = \langle N, E \rangle$ consists of $N = \{p_i\}$, a set of nodes and $E = \{e_i\}$, a set of edges. A specific location in space is represented by node $p \in R^2$. Each edge is a pair of nodes $e = \langle p_i, p_j \rangle$. Figure 5.2 shows an example of a graph abstracted from a building layout. In this graph, walls are represented by edges (e) and the junctions and end-points are

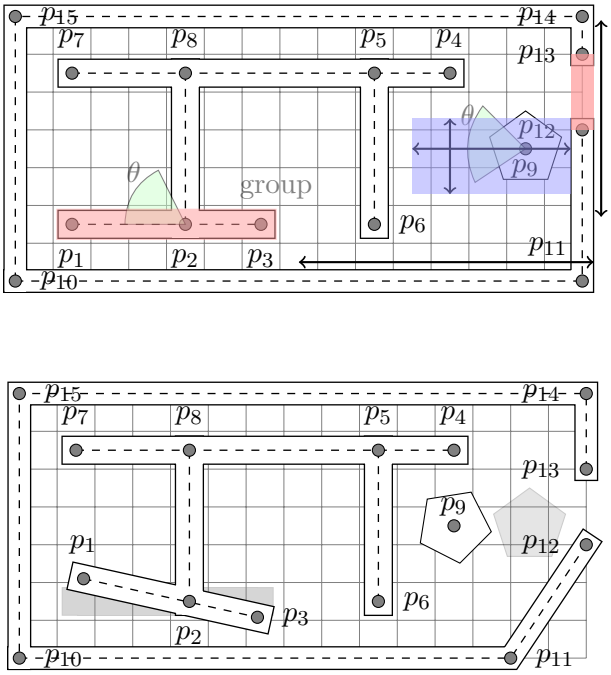


Figure 5.2: The layout of a floor plan, the corresponding graph parameterization of the walls and other architectural elements, and examples of parameter manipulation. The nodes p_i can be moved, grouped, rotated, and constrained within certain bounds. The top figure shows user defined constraints. For example the arcs, painted regions and arrows around p_9 are the user defined range within which that node can move or rotate. The bottom figure is a single instance of the initial configuration (top figure) to demonstrate the effect from user selection. For example the nodes p_1, p_2, p_3 were grouped and rotated around p_2. The grid cells provide a discrete representation of the open space that can be used for spatial static analysis.

represented by nodes (p). In addition to the inter-connected junctions and end-points the architectural graph may also have a single or more disconnected nodes which represent single objects like kiosks, ATMs, garbage bins etc. Each node or edge is associated with an architectural element in an architectural environment. The geometry of all the architectural elements are stored in a database for further reference.

5.2.2 Architectural Graph Parameterization

A variety of designs can be produced using a full parameter space (domain), $\mathcal{P}$, which represents all the nodes $\mathbf{p} = \langle p_i \in N \rangle$ with their associated bounds. μDOME allows the user to define parameters for the architectural elements which are considered a subset of constraints $C(\mathbf{p})$ for the overall environment design. The user starts working initially with a small subset of nodes. These nodes are then regarded as rigid bodies and a parameter vector is generated which contains the location of individual nodes p_i, and the rigid body parameters of each group of nodes, g_j, that the user has selected, $\mathbf{p} = (p_i, g_{j,pos}, g_{j,\theta})$.

Figure 5.2 illustrates two configurations of a floor plan having 16 nodes. An initial configuration is shown at the top while a new configuration is illustrated at the bottom. Around p_9, the arcs, painted regions and arrows are the user defined range within which that node can move or rotate. The group of nodes p_1, p_2, p_3 (in red) can rotate around p_2 within the specified range. The nodes p_{12} and p_{13} need to remain in a group by maintaining their initial distance but can move vertically along y-direction region. In the new configuration (bottom figure), the group of nodes p_1, p_2, p_3 has been rotated by

10 degrees, while node p_9 has been rotated around its geometric center by 45 degrees, and then moved to a new location. The example shows that the outer walls can also be rotated in a similar manner as that of the nodes p_{11}, p_{12} and p_{13}. It is important to note that the nodes in a group maintain their distances with each other.

5.3 Spatial Analysis

Human spatial cognition and architectural elements are the two main aspects which governed human behavior in an architectural environment (Hölscher et al. 2004). In particular, this work investigates the relationship between architectural elements and their impact on human behavior. For example, pillar placements can affect the crowd evacuation time of an architectural environment (Berseth et al. 2015) and introducing navigational signs in an architectural environment help people to navigate towards their goals (Hölscher et al. 2004). Similarly, some certain arrangements of walls can affect human attention directed towards important areas in an architectural environment.

Effects of architectural environments on their users can be quantified by using measures from spatial analysis which directly relate to human behavior. There are different approaches to compute these measures. One of the approaches is to run crowd simulations through the architectural environment design (Berseth et al. 2015). However, this approach is very sensitive to the dynamics of the simulation and can be computationally exhaustive depending on the selected simulation technique, and hence not very suitable for interactive systems.

A static approach, well established in space-syntax analysis (Hillier and Hanson 1984, Peponis et al. 1990, Turner and Penn 1999), is used for the spatial analysis of architectural designs. The main idea is to decompose the architectural environment space and represent it as a graph of architectural components. This graph is then spatially analyzed by computing features like integration and connectivity which reflect the behavior of people in that space.

5.3.1 Space Decomposition

First, the continuous space of an architectural environment is decomposed into a sparse and discrete graph-based representation. Visibility graphs (Desyllas and Duxbury 2001) are widely used to represent architectural environment space. Vertices of the graph represent the points in the architectural environment space and edges represent unobstructed lines of sight between the vertices, see Figure 5.1 (top, left). Visibility graphs are easy to compute and capture realistic and accurate spatial information (Desyllas and Duxbury 2001, Turner and Penn 1999). This work will refer to a visibility graph with the symbol G_V. Vertices and edges of the graph will be denoted by V_V and E_V respectively.

5.3.2 Visibility Graph

A visibility graph is constructed by sampling the architectural environment space with a fixed grid V. All the nodes in the grid are the vertices of the visibility graph. In some cases it is very effective to limit the visibility graph to important regions. For example,

the user may be more interested in the visibility of an architectural element from some important regions and not over the whole architectural environment space.

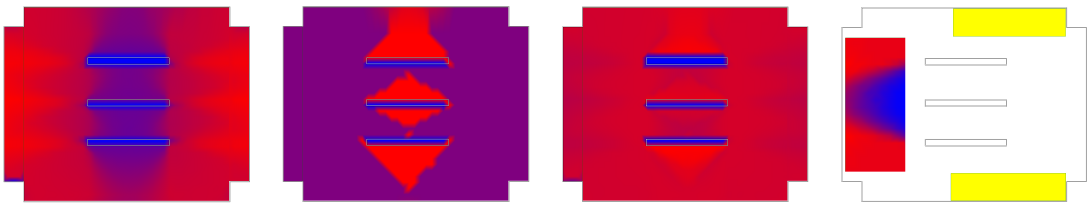
To address this need, two sets of regions of the grid nodes are defined, the *Query Region* with nodes $V_q \subset V$, and the *Reference Region* with nodes $V_r \subset V$. The two regions can overlap. A visibility graph is constructed from these two sets of nodes by computing the lines of sights between the nodes of each set. Points A and B in space are visible to each other if and only if there is a clear line of sight between them. Once a visibility graph is constructed for an architectural environment, several spatial graph features can be analyzed. Features can include connectivity, mean depth or shortest path between the nodes of the two regions. These features can then be considered static metrics for evaluating that architectural environment. The constructed visibility graph is a function of *Query Region* and *Reference Region*, provided by the user, in addition to the architectural graph and the sampling grid V:

$$G_V = \phi(V_r, V_q, G_A(\mathbf{p}), V) \tag{5.1}$$

The selection of *Query Region* and *Reference Region* is provided as optional interaction steps for architects and designers using μDOME for space analysis.

5.3.3 Spatial Metrics

Once the visibility graph is constructed, the next step is to compute the spatial metrics to characterize the relationship between architectural environments and human behav-



(**a**) (**b**) (**c**) (**d**)

Figure 5.3: Metrics values for a room in an art gallery. Heatmap color ranges from blue (low) to red (high). (a) Degree of visibility, where red areas show more integrated regions, and are good candidates for placement of fire exits, signs, main event, etc. (b) Tree depth, where blue areas have lower depth and are easier to access. (c) Entropy, where red areas have high entropy (order), resulting in better human environmental cognition and easier planning at those points. (d) Degree of visibility in *Query Region* with respect to *Reference Region* which is shown in yellow. The degree values (which are a function of *Reference Region*, are different in comparison to (a).

ior. μDOME makes use of spatial metrics from space-syntax analysis which have been proposed in the literature (Hillier et al. 1987, Jiang et al. 2000, Turner 2001). The result of the computation of these metrics are represented as heat maps, Figure 5.1 (top, left). Following metrics are considered in the spatial analysis:

Degree of Visibility. The number of edges $e_j \in E_V$ incident to the vertex $v_i \in V_V$ is considered as the degree of visibility of that vertex. Degree of visibility of a vertex is denoted as k_i, and the set of its neighbors are denoted as N_i. Thus, the degree of visibility is equal to the cardinality of this set:

$$k_i = |N_i|. \tag{5.2}$$

Regions with high degree of visibility can be considered as more connected and have better field of view in the architectural environment. For example, if a user wants to install safety signs, then positioning them at high visibility regions might be more appropriate. Figure 5.3(a) shows the computed visibility values in heat map formed over a sample architectural environment.

Tree Depth. A forest F_i is consist of all the trees whose root is $v_i \in V_V$, edges are a subset of E_V, and which span the whole connected graph G_V^i (a connected subgraph of G_V that includes v_i). The Tree Depth d_i is the rank of the graph tree T_i which has the minimum depth in a forest F_i.

$$d_i = \text{rank}(T_i). \tag{5.3}$$

The tree depth can be related to accessibility and distance notations. A node with high tree depth is connected to other regions of the architectural environment (visibility graph) through a longer sequence of nodes. Figure 5.3(b) shows the computed depth values in heat map formed over a sample architectural environment.

Entropy. For any given tree T_i with n_i^j vertices at each level j, a probability distribution $p_i(j)$ is defined for T_i with the support in $j \in [1, |V|]$. Then the entropy h_i at vertex i, is computed as:

$$p_i(j) = \frac{n_i^j}{\sum_{j' \in [1,|V|]} n_i^{j'}},$$
$$h_i = -\sum_{j \in [1,|V|]} p_i(j) \log_2 p_i(j). \tag{5.4}$$

Entropy measures the organization of an architectural environment. For example, how easy it is for crowd of pedestrians to plan and navigate through the architectural environment. If a node has a low entropy value then it means that the sequence of planning steps required to reach other regions of the architectural environment from that node is unbalanced. Figure 5.3(c) shows the computed entropy values as a heat map over a sample architectural environment.

Figure 5.3(d) shows the degree of visibility computed over the *Query Region* with respect to *Reference Region* which is shown in yellow. It is note worthy that changing the reference from the entire environment in part (a), to just the top and bottom hallways can greatly affect the values of the metrics, and hence our view of the architectural environment space.

For the complete visibility graph G_V with all the vertices V, these metrics are simply the averages of the corresponding per vertex measures:

$$\begin{aligned} K(G_V) &= \textstyle\sum_{i=1}^{|V|} \frac{k_i}{|V|}, \\ D(G_V) &= \textstyle\sum_{i=1}^{|V|} \frac{d_i}{|V|}, \\ H(G_V) &= \textstyle\sum_{i=1}^{|V|} \frac{h_i}{|V|}. \end{aligned} \tag{5.5}$$

5.4 Hierarchical Multi-Objective Diversity Optimization

A hierarchical multi-objective diversity optimization technique is integrated within μDOME that uses spatial measures (as defined in Section 5.3.3) grounded in space-syntax analysis to quantify how humans interact with architectural environments.

Using μDOME the user first needs to select the movable architectural elements with all of its associated structural constraints like transformation or rotation bounds etc. and multi-dimensional parameter space of permissible architectural environment configurations. Next, the user needs to define the optimization criteria by selecting one or more spatial metrics defined in Section 5.3.3. After defining the optimization criteria the user then needs to define the environment regions where the spatial metrics must be optimized specifically. For instance, if it is required to enhance the visibility of a painting in a room from the entrance(s), then this needs to be achieved while room layout is properly ordered and people have sufficient clearance between the walls for navigation. This leads to a multi-objective optimization problem which can be solved using existing approaches of optimal configuration (Hansen and Ostermeier 1996). A key novelty of this approach is that it provides various solutions which are all near optimal yet rich in diversity, instead of providing a single optimal configuration directly. This formulation introduces another term in the objective formulation, and requires the solver to focus the search to meet optimality criteria, while simultaneously broadening its exploration to maximize diversity of its candidate solutions. A novel hierarchical multi-objective algorithm for optimization is then formulated which ensures efficiency while maintaining balance between optimal-

ity and diversity. Using this optimization technique μDOME provides a unique service to the users by giving them the opportunity to select among the various architectural designs instead of just providing a single optimal solution. The proposed system serves to assist users by providing them a range of candidate solutions which meet the user specified constraints and search space. The user then continues the process by selecting one of the architectural designs from the set of diversified yet optimal designs. The user has the liberty to either use the same parameters or alter them, select the optimization objectives and may repeat this optimization process until a design is developed which satisfies the user needs. This leads to a healthy approach of brainstorming architectural design assistance, in order to achieve efficient and diverse optimal solutions.

Diversity measure. Diversity measure is introduced into the optimization framework to provide various solutions which are all near optimal yet rich in diversity, instead of providing a single optimal configuration of parameters directly. It is computed by calculating distances among the candidate solutions. Any standard distance metric can be used to measure diversity.

Figure 5.4 shows a detailed example to illustrate this feature. The columns in the figure correspond to a combination of metrics, the degree, the depth, and the entropy metric, respectively as defined in Section 5.3.3. All interior walls were chosen as moveable architectural elements in this example. The area in white is the *Reference Region*. The area with heat maps is the *Query Region*. The top row in the figure is the initial architectural environment. Subsequent rows are diverse, near-optimal results returned using

this optimization technique. Further details about optimization algorithm including the diversity measure can be found in the original paper (Berseth et al.).

5.5 Integration with Autodesk Revit ®

Autodesk provides powerful APIs (Application Programming Interface) that allow users to programmatically interact with the Autodesk software products and tailor them according to their requirements. These APIs are available in different programming languages. μDOME is integrated within the professional Autodesk Revit ® 2016 pipeline using the *.NET* API provided by AutoCAD. Figure 5.5 shows μDOME interface integrated within a professional CAD product.

5.6 User-Study

The usability and efficacy of μDOME is evaluated through user studies which are discussed below.

5.6.1 Usability

This study is done to evaluate the general usability of μDOME. μDOME is evaluated using a well established method – the *System Usability Scale* (Bangor et al. 2008, Brooke et al. 1996). The method is used as outlined by Brooke (Brooke et al. 1996). It's a reliable and widely used approach to roughly differentiate between usable and unusable systems. Each user in the study worked with μDOME for a maximum of 15 minutes.

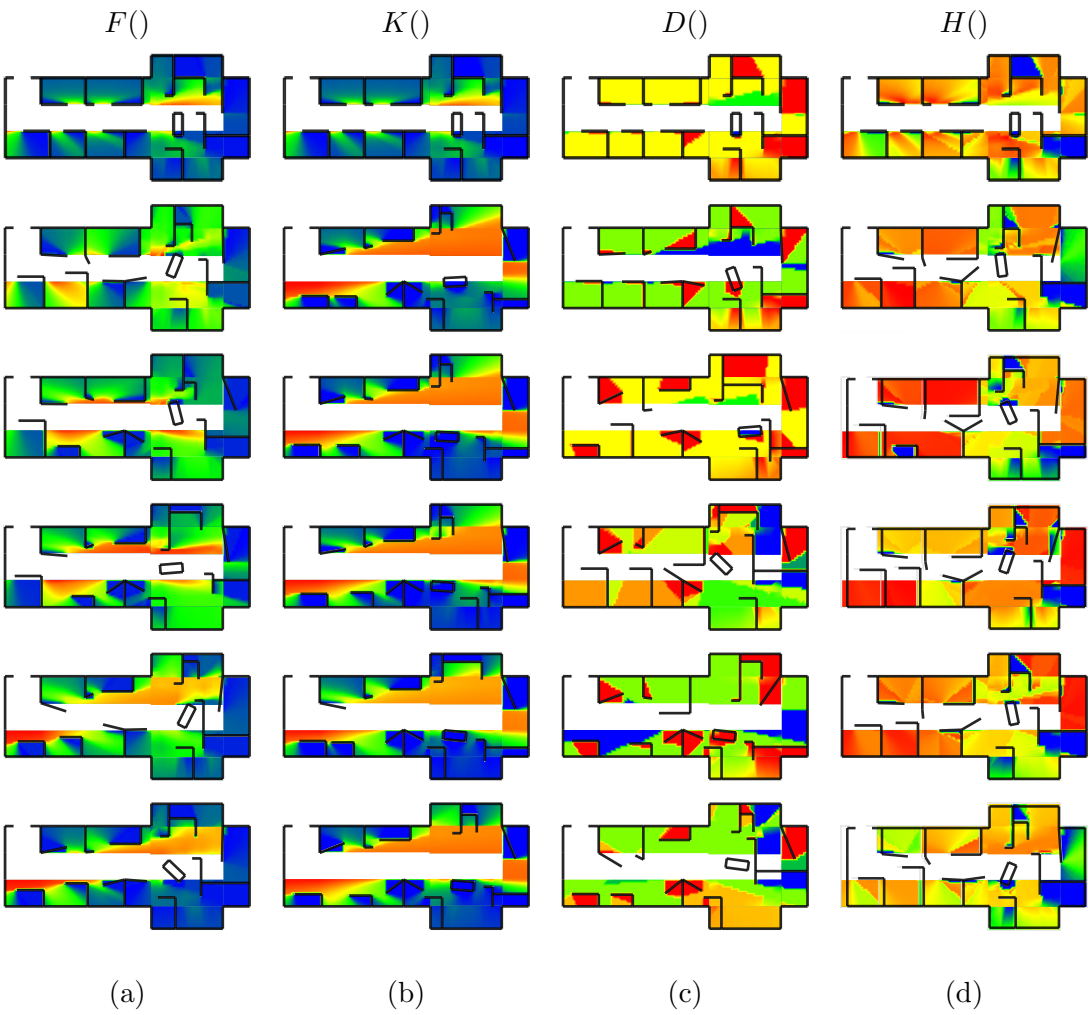


Figure 5.4: Diverse optimization results using our method. From left to right. Diverse, near-optimal candidate layouts for: (a) Weighted combination of all three metrics. (b) Degree of visibility. (c) Depth. (d) Entropy. All interior walls were chosen as moveable elements in this example. The area in white is the reference region. The area with heat maps is the query region. Top row is default configuration. Subsequent rows are diverse, near-optimal results returned using our method.

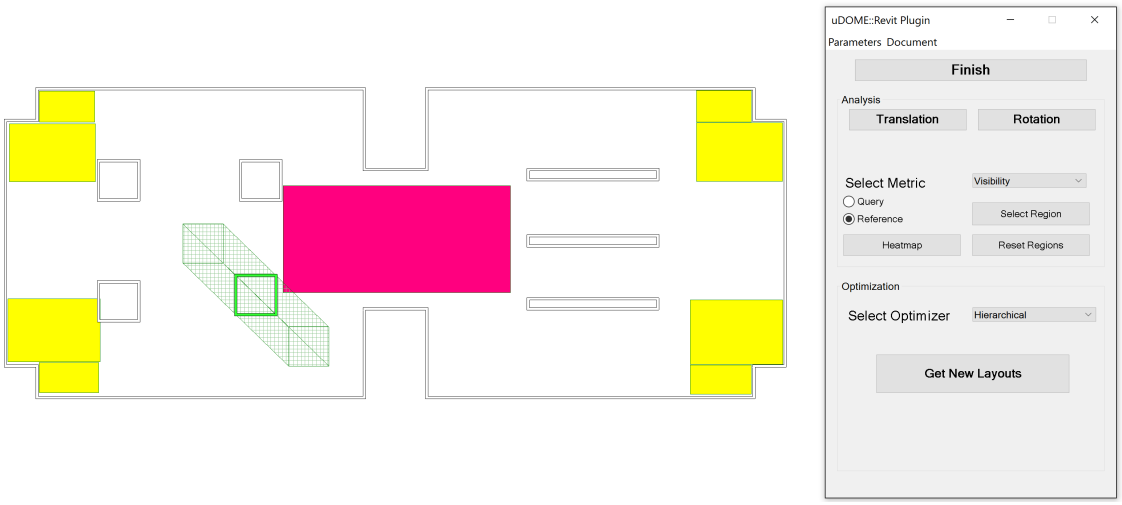


Figure 5.5: μDOME interface integrated within Autodesk Revit ® 2016. Solid green block represents user selected movable architectural elements i.e. set of walls, with translation constraints. Sketched green region shows the bounds for that translation constraint. Pink is the *Query Region*, whereas Yellow are the *Reference Region*.

Users were given brief training and instructions on how to use the μDOME interface. In total, 13 users aged between 23 to 30 participated in this study. All users were graduate level students in computer science or closely related fields of studies. All users were given the architectural environment scenario which is similar to Figure 5.4 in complexity.

The average, median and standard deviation of the SUS scores are shown in Table 5.1. A mean of 71.73 is considered "Good" according to (Bangor et al. 2009). The quartiles of the SUS scores are stated in Table 5.2. The usability was tested to make sure that the system is easy to use and all the features and components of the system are working smooth and efficient.

Table 5.1: SUS Results

Count	Average	Median	Standard Deviation
13	71.73	75	18.27

Table 5.2: SUS Quartiles

Quartile	Lower Bound	Upper Bound
1	22.5	67.5
2	67.5	75
3	75	82.5
4	82.5	87.5

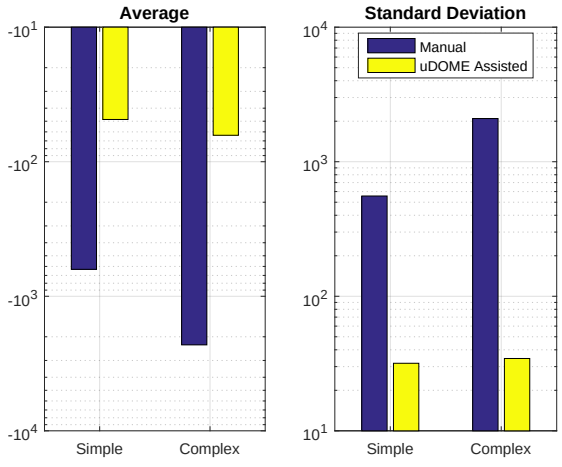


Figure 5.6: Comparison between manual group and users assisted by μDOME. μDOME users design layouts with significantly higher objective measures on average, and have lesser deviation, indicating greater consistency across users.

5.6.2 Efficacy

An experiment was conducted to find out how effective μDOME is when used by novice users. The effectiveness of the system was investigated based on how helpful the system is when novice users try to design optimal architectural designs. A group of 13 people are given basic training and instructions on how to use μDOME interface. Users are then divided into two groups. Group A manually design the optimal architectural layout, while group B uses μDOME to optimize the architectural layout. This experiment was run using two layouts, a simple room and a complex architectural environment. In Figure 5.6, the average combined metric values are shown at the top, the bottom of the figure shows its standard deviations. It is obvious from the figure that μDOME has positive effect on the metric value, i.e. higher on average metric. Also, good spatial analysis metric values are achieved by the users using μDOME. The standard deviation of the manual users shows that the optimization results can be very inconsistent when done manually while μDOME helps users and guides them with focused architectural design exploration. Moreover, it is worth mentioning even for these simple architectural design layouts, that the solutions designed by the users using μDOME were diverse, with different subjects finding new ways to maximize the objective.

5.7 Summary

This chapter presents μDOME, an interactive user-in-the-loop system for computer-aided architectural environment design that analyzes, and optimizes layouts of architectural

environments with respect to human behavior. It uses measure of spatial analysis, which are firmly grounded in space-syntax, and help quantify the effect of an architectural environment on its occupants behavior. Using a hierarchical multi-objective diversity optimization method from SteerSuite, μDOME produces a set of maximally diverse, yet near-optimal candidate solutions. At all the stages of the architectural design process, users are provided with several design candidates that they can then choose from (they may change parameter values as well if desired) thereby making them a crucial part of the overall design process. In other words, μDOME is not taking over the creative architecutural design process. Rather it enhances this creativity by aiding the users. User-studies show that by using μDOME, the design performance of even novice users (with basic training) improves without affecting the diversity of these designs.

Chapter 6

Conclusion

This thesis set out to describe the design and development of interactive computer-aided systems to do static and dynamic analysis of the architectural elements in an environment design. Both systems facilitate designers and architects to optimize and analyze the architectural elements like pillars, obstacles or walls in their environment designs. This chapter will summarize the work presented in the thesis. At the end, it will discuss the limitations and subsequent future directions of this work.

6.1 Results and Implications

First, a simulation framework and two preliminary studies are presented with experiments for the proof of concept showing that the optimal pillar placements can help in improving crowd flow in evacuation scenarios. The experimental results reveal many interesting insights, highlighting the sensitivity of optimal architectural configurations on the choice of dynamic crowd simulators. It is observed that in most of the scenarios, the optimal

number of architectural elements (pillars) was found to be 3. Another observation is that the level of service, **LoS**, afforded by a particular architectural environment is sensitive to how the crowd behaves, and how the architectural environment is configured. The critical density of crowd simulators effectively increases due to the optimal placement of 1–4 architectural elements (pillars) in the architectural designs, thereby increasing the crowd flow at greater densities (near **LoS** D and E). However, this behavior is not uniform across all crowd simulators and architectural environment scenarios.

Next, the development of an interactive computer-aided system is presented, `CODE`. The system uses dynamic crowd simulations to analyze the architectural elements in the layout designs. An enhanced automated technique is also presented for optimizing the placements of architectural elements (pillars). `CODE` is designed to be used as an aid to people involved with building designs and layouts. The user-study results reveal that as building architectures become more complex, interactive automated systems like `CODE` can be of great assistance. It has potential as an enabling technology for practitioners that are involved with the design and layout of buildings, such as architects, fire marshals, etc.

Now to study the static analysis, another computer-aided interactive system is presented, μDOME. It analyzes the environments using measures of spatial analysis (visibility, entropy and organization), which are firmly grounded in space-syntax, and help quantify the effect of an architectural environment on its occupants behavior. The results from a user-study show that by using μDOME, the design performance of even novice

users (with some basic training) improves without affecting the diversity of the design solutions. μDOME has been integrated within professional architectural design software system, Autodesk Revit ®.

6.2 Limitations and Future Work

The systems presented in this work are developed for proof-of-concept to test the static and dynamic techniques for analyzing the architectural elements in an environment design.

The first system presented is `CODE`. It will continue to be developed for complex crowd interaction scenarios by way of close collaboration with experts in these areas, along with plans of performing a usability study with the experts. Current work is limited with the use of specific type of space-partitioning structure and region-only constraints. Future work will address these limitations. It is the intention to increase the efficacy of `CODE` and compare its dynamic analysis results with the results of static analysis from μDOME. Integration of `CODE` with standard architectural design software is another future work.

μDOME only statically analyzes the architectural environment designs using space-syntax measures and shows feedback in term of heat maps visualizations. The next step is to dynamically analyze the complex architectural designs within Autodesk Revit ® using current generation dynamic crowd simulators. This analysis will be followed by a comparative study of the results from static and dynamic analysis on the set of complex architectural designs.

This thesis separately investigates the static and dynamic analysis of architectural elements. One of next steps is to compare the two techniques.

Bibliography

Sonit Bafna. Space Syntax: A Brief Introduction to Its Logic and Analytical Techniques. *Environment and Behavior*, 35(1):17–29, 2003.

Aaron Bangor, Philip T Kortum, and James T Miller. An empirical evaluation of the system usability scale. *International Journal of Human–Computer Interaction*, 24(6): 574–594, 2008.

Aaron Bangor, Philip Kortum, and James Miller. Determining what individual sus scores mean: Adding an adjective rating scale. *Journal of usability studies*, 4(3):114–123, 2009.

Marsha J Berger and Phillip Colella. Local adaptive mesh refinement for shock hydrodynamics. *Journal of computational Physics*, 82(1):64–84, 1989.

Glen Berseth, Mahyar Khayatkhoei, Muhammad Usman, Brandon Haworth, Mubbasir Kapadia, and Petros Faloutsos. μDOME: User-in-the loop diversity optimization of environments. To be published.

Glen Berseth, Mubbasir Kapadia, and Petros Faloutsos. Steerplex: Estimating scenario complexity for simulated crowds. In *Proceedings of Motion in Games*, pages 45:67–45:76. ACM, 2013.

Glen Berseth, M Brandon Haworth, Mubbasir Kapadia, and Petros Faloutsos. Characterizing and optimizing game level difficulty. In *Proceedings of the Seventh International Conference on Motion in Games*, pages 153–160. ACM, 2014.

Glen Berseth, Muhammad Usman, Brandon Haworth, Mubbasir Kapadia, and Petros Faloutsos. Environment optimization for crowd evacuation. *Computer Animation and Virtual Worlds*, 26(3–4):377–386, 2015.

A. Best, S. Narang, S. Curtis, and D. Manocha. Densesense: Interactive crowd simulation using density-dependent filters. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 97–102. Eurographics Association, 2014.

John Brooke et al. Sus - a quick and dirty usability scale. *Usability evaluation in industry*, 189(194):4–7, 1996.

Jake Desyllas and Elspeth Duxbury. Axial maps and visibility graph analysis: A comparison of their methodology and use in models of urban pedestrian movement. In *3rd International Space Syntax Symposium*, pages 27.1–27.13, 2001.

Juliana Felkner, Eleni Chatzi, and Toni Kotnik. Interactive particle swarm optimization for the architectural design of truss structures. In *Computational Intelligence for Engineering Solutions*, pages 15–22. IEEE, 2013.

Tian Feng, Lap-Fai Yu, Sai-Kit Yeung, KangKang Yin, and Kun Zhou. Crowd-driven mid-scale layout design. *Transactions on Graphics*, 35(4):132:1–132:14, 2016.

John J Fruin. Pedestrian planning and design. Technical report, 1971.

Stephen J. Guy, Jur van den Berg, Wenxi Liu, Rynson Lau, Ming C. Lin, and Dinesh Manocha. A statistical similarity measure for aggregate crowd dynamics. *Transactions on Graphics*, 31(6):190:1–190:11, 2012.

N. Hansen and A. Ostermeier. Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation. In *International Conference on Evolutionary Computation*, pages 312–317. IEEE, 1996.

Brandon Haworth, Muhammad Usman, Glen Berseth, Mubbasir Kapadia, and Petros Faloutsos. Evaluating and optimizing level of service for crowd evacuations. In *ACM SIGGRAPH Motion in Games*, pages 91–96. ACM, 2015.

Brandon Haworth, Muhammad Usman, Glen Berseth, Mahyar Khayatkhoei, Mubbasir Kapadia, and Petros Faloutsos. Towards computer assisted crowd aware architectural design. In *Proceedings of CHI Extended Abstracts on Human Factors in Computing Systems*, pages 2119–2125. ACM, 2016a.

Brandon Haworth, Muhammad Usman, Glen Berseth, Mahyar Khayatkhoei, Mubbasir Kapadia, and Petros Faloutsos. Using synthetic crowds to inform building pillar placements. In *Workshop on Virtual Humans and Crowds for Immersive Environments, Virtual Reality (VR)*. IEEE, 2016b.

D. Helbing, A. Johansson, and H. Zein Al-Abideen. Dynamics of crowd disasters: An empirical study. *Physical Review E*, 75(4):046109, 2007.

Dirk Helbing, Illés Farkas, and Tamas Vicsek. Simulating dynamical features of escape panic. *Nature*, 407(6803):487–490, 2000.

Bill Hillier and Julienne Hanson. The social logic of space. *Cambridge: Press syndicate of the University of Cambridge*, 1984.

WRG Hillier, Julienne Hanson, and John Peponis. Syntactic analysis of settlements. *Architecture and Behaviour*, 3(3):217–231, 1987.

Christoph Hölscher, Tobias Meilinger, Georg Vrachliotis, Martin Brösamle, and Markus Knauff. Finding the way inside: Linking architectural design analysis and cognitive processes. In *Spatial Cognition IV. Reasoning, Action, Interaction*, pages 1–23. Springer, 2004.

Bin Jiang, Christophe Claramunt, and Björn Klarqvist. Integration of space syntax into gis for modelling urban spaces. *International Journal of Applied Earth Observation and Geoinformation*, 2(3):161–171, 2000.

Li Jiang, Jingyu Li, Chao Shen, Sicong Yang, and Zhangang Han. Obstacle optimization for panic flow - reducing the tangential momentum increases the escape speed. *PLoS ONE*, 9(12):1–15, 2014.

A. Johansson, D. Helbing, H. Z A-Abideen, and S. Al-Bosta. From crowd dynamics to crowd safety: A video-based analysis. *Advances in Complex Systems*, 11(4), 2008.

Mubbasir Kapadia, Shawn Singh, William Hewlett, and Petros Faloutsos. Egocentric affordance fields in pedestrian steering. In *Symposium on Interactive 3D Graphics and Games*, pages 215–223. ACM, 2009.

Mubbasir Kapadia, Matt Wang, Shawn Singh, Glenn Reinman, and Petros Faloutsos. Scenario space: characterizing coverage, quality, and failure of steering algorithms. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 53–62. ACM, 2011.

Mubbasir Kapadia, Nuria Pelechano, Jan Allbeck, and Norm Badler. Virtual crowds: Steps toward behavioral realism. *Synthesis Lectures on Visual Computing*, 7(4):1–270, 2015.

Ioannis Karamouzas, Peter Heil, Pascal Beek, and Mark H. Overmars. A predictive collision avoidance model for pedestrian simulation. In *Proceedings of the 2nd International Workshop on Motion in Games*, pages 41–52. Springer-Verlag, 2009.

Alon Lerner, Yiorgos Chrysanthou, Ariel Shamir, and Daniel Cohen-Or. Data driven evaluation of crowds. In *Proceedings of Motion in Games*, pages 75–83. Springer, 2009.

Alon Lerner, Yiorgos Chrysanthou, Ariel Shamir, and Daniel Cohen-Or. Context-dependent crowd evaluation. In *Computer Graphics Forum*, volume 29, pages 2197–2206. Wiley Online Library, 2010.

Paul Merrell, Eric Schkufza, and Vladlen Koltun. Computer-generated residential building layouts. *Transactions on Graphics*, 29(6):181:1–181:12, 2010.

Jeremy Michalek and Panos Papalambros. Interactive design optimization of architectural layouts. *Engineering Optimization*, 34(5):485–501, 2002.

Soraia R Musse, Vinicius J Cassol, and Cláudio R Jung. Towards a quantitative approach for comparing crowds. *Computer Animation and Virtual Worlds*, 23(1):49–57, 2012.

Rahul Narain, Abhinav Golas, Sean Curtis, and Ming C. Lin. Aggregate dynamics for dense crowd simulation. *Transactions on Graphics*, 28(5):122:1–122:8, 2009.

Sahil Narang, Andrew Best, Sean Curtis, and Dinesh Manocha. Generating pedestrian trajectories consistent with the fundamental diagram based on physiological and psychological factors. *PLoS one*, 10(4):e0117856, 2015.

Jan Ondřej, Julien Pettré, Anne-Hélène Olivier, and Stéphane Donikian. A synthetic-vision based steering approach for crowd simulation. *Transactions on Graphics*, 29(4): 123:1–123:9, 2010.

John Peponis, Craig Zimring, and Yoon Kyung Choi. Finding the building in wayfinding. *Environment and behavior*, 22(5):555–590, 1990.

Craig W. Reynolds. Flocks, herds and schools: A distributed behavioral model. *SIGGRAPH Computer Graphics*, 21(4):25–34, 1987.

Craig W Reynolds. Steering behaviors for autonomous characters. *Game developers conference*, 1999:763–782, 1999.

S. Rodriguez, Y. Zhang, N. Gans, and N. M. Amato. Optimizing aspects of pedestrian traffic in building designs. In *RSJ International Conference on Intelligent Robots and Systems*, pages 1327–1334. IEEE, 2013.

Armin Seyfried, Maik Boltes, Jens Kähler, Wolfram Klingsch, Andrea Portz, Tobias Rupprecht, Andreas Schadschneider, Bernhard Steffen, and Andreas Winkens. Enhanced empirical data for the fundamental diagram and the flow through bottlenecks. In *Pedestrian and Evacuation Dynamics*, pages 145–156. Springer, 2008.

Xing Shi and Wenjie Yang. Performance-driven architectural design and optimization technique from a perspective of architects. *Automation in Construction*, 32:125–135, 2013.

Shawn Singh, Mubbasir Kapadia, Petros Faloutsos, and Glenn Reinman. Steerbench: a benchmark suite for evaluating steering behaviors. *Computer Animation and Virtual Worlds*, 20(5-6):533–548, 2009a.

Shawn Singh, Mubbasir Kapadia, Petros Faloutsos, and Glenn Reinman. An open framework for developing, evaluating, and sharing steering algorithms. In *Proceedings of Motion in Games*, pages 158–169. Springer-Verlag, 2009b.

Shawn Singh, Mubbasir Kapadia, Billy Hewlett, Glenn Reinman, and Petros Faloutsos. A modular framework for adaptive agent-based steering. In *Symposium on Interactive 3D Graphics and Games*, pages 141–150. ACM, 2011.

Alasdair Turner. A program to perform visibility graph analysis. In *Proceedings of the 3rd Space Syntax Symposium, Atlanta, University of Michigan*, pages 31–1, 2001.

Alasdair Turner and Alan Penn. Making isovists syntactic: isovist integration analysis. In *2nd International Symposium on Space Syntax, Brasilia*. Citeseer, 1999.

Michela Turrin, Peter von Buelow, and Rudi Stouffs. Design explorations of performance driven geometry in architectural design using parametric modeling and genetic algorithms. *Advanced Engineering Informatics*, 25(4):656–675, 2011.

Jur van den Berg, Stephen J. Guy, Ming C. Lin, and Dinesh Manocha. Reciprocal n-body collision avoidance. In *International Symposium on Robotic Research*, 2009.