# Topology-based Catalogue Exploration Framework for Identifying View-Enhanced Tower Designs

Harish Doraiswamy
New York University

Nivan Ferreira
New York University

Marcos Lage
Universidade Federal Fluminense

Huy Vo
New York University

Luc Wilson    Heidi Werner    Muchan Park    Cláudio Silva
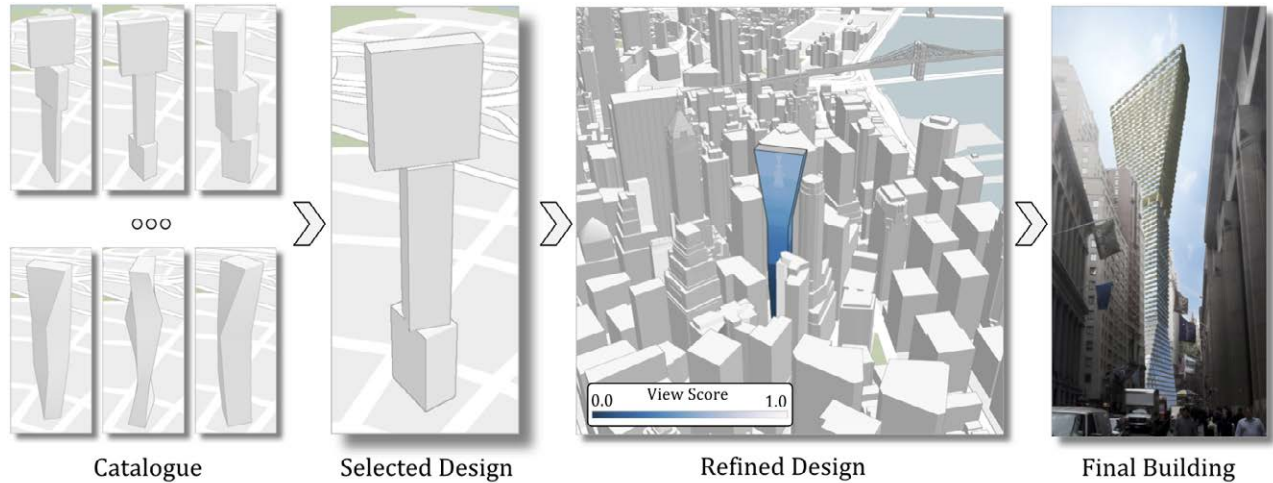
Kohn Pedersen Fox Associates PC      New York University

**Figure 1:** *The topology-based catalogue framework identifies building designs that have good performance with respect to outward views. In their conceptual design phase, architects use the catalogue to identify a few high performing designs, which are then further refined to satisfy client objectives and government regulations into a final building design.*

## Abstract

There is a growing expectation for high performance design in architecture which negotiates between the requirements of the client and the physical constraints of a building site. Clients for building projects often challenge architects to maximize view quality since it can significantly increase real estate value. To pursue this challenge, architects typically move through several design revision cycles to identify a set of design options which satisfy these view quality expectations in coordination with other goals of the project. However, reviewing a large quantity of design options within the practical time constraints is challenging due to the limitations of existing tools for view performance evaluation. These challenges include flexibility in the definition of view quality and the ability to handle the expensive computation involved in assessing both the view quality and the exploration of a large number of possible design options. To address these challenges, we propose a catalogue-based framework that enables the interactive exploration of conceptual building design options based on adjustable view preferences. We achieve this by integrating a flexible mechanism to combine different view measures with an indexing scheme for view compu-

tation that achieves high performance and precision. Furthermore, the combined view measures are then used to model the building design space as a high dimensional scalar function. The topological features of this function are then used as candidate building designs. Finally, we propose an interactive design catalogue for the exploration of potential building designs based on the given view preferences. We demonstrate the effectiveness of our approach through two use case scenarios to assess view potential and explore conceptual building designs on sites with high development likelihood in Manhattan, New York City.

**CR Categories:** I.3.8 [Computer Graphics]: Applications;

**Keywords:** computational topology, visualization, architectural design, performance based design, urban computing

## 1 Introduction

There is a growing necessity for high performance tower design in architecture which negotiates between the requirements of the client and the physical constraints of a building site. In dense, geographically constrained cities like New York City (NYC) and Hong Kong, towers are an important building type for two primary reasons. First, the average size of developable sites has shrunk as easy to develop, larger sites are developed first. And, second, these cities are zoned for high densities, making towers the only option for maximizing all allowable building area. Architects are interested in exploring and identifying building design variations from a design parameter space that satisfy their needs at an early stage in their design process. This helps them mitigate the cost and impact of backtracking within the process that is caused due to suboptimal choices [Kimpian et al. 2009]. These choices are typically

made based on multiple performance measures. Fast revision cycles during this stage can help maximize the coverage of the set of design choices or design space, while balancing the trade-off between physical constraints and performance measures.

One such performance measure is the *quality of outward views* (or views) from the dwelling units of a building. Desirable views are known to have value in the real estate market as they can significantly impact sale and rental prices [Benson et al. 1998]. However, in spite of their importance, considering view quality in building design is challenging current architectural workflows. This is especially true in dense urban environments where views are increasingly limited by the surrounding context and challenging to detect. In fact, there is little consensus on how to objectively measure views since rating their quality is inherently subject to viewer preference and contextually specific conditions [Tsigkari et al. 2013]. Therefore, many possible view quality measures should be considered during the design process.

Measuring view quality not only involves the integration of different datasets but is also expensive to compute. The typical process in the architect's workflow involves the use of a suite of off-the-shelf software, such as McNeel Rhinoceros or Autodesk Revit, that does not effectively support the study of a design parameter space or the associated views. These tools are limited in both their computational power and their capability to help users explore a high quantity of possible design choices [Tsigkari et al. 2013]. The addition of practical limitations such as project deadlines forces the computation of view quality to be simplified to coarser non-realistic versions and also limits the number of such computations. In fact, in practice, only a handful of designs are typically considered at any given point in the conceptual design phase.

The goal of this work is to better equip architects in their early design phases to make more informed decisions. Any such useful system should be able to realistically evaluate the view measures for a large quantity (potentially millions) of possible building designs. It should also be able to identify "good" designs that, in addition to having good performance, also cover distant regions of the parameter space, thus providing geometrically distinct options to the architects. An equally important goal is to have an effective interface that helps in exploring and comparing the potentially large number of design variations.

## 1.1 Contributions

In this paper, we present a catalogue-based framework that was designed in collaboration with architects. The framework allows architects to interactively explore conceptual building design options based on their view preferences. In particular, our contributions are as follows:

1. We define four different view analysis types, each representing a different aspect of the view. Users can weight the importance of each view type based on specific preferences.

2. To enable fast evaluation of views for different design variations we create a simple yet accurate indexing scheme using the view scores from a subset of the view space. Computing the view scores at a given location along a specified direction is transformed into a quick look up, taking $O(1)$ time. We also show experimental results demonstrating the accuracy of the proposed indexing technique.

3. An efficient topology-based technique to identify distinct high performing building designs from the design parameter space.

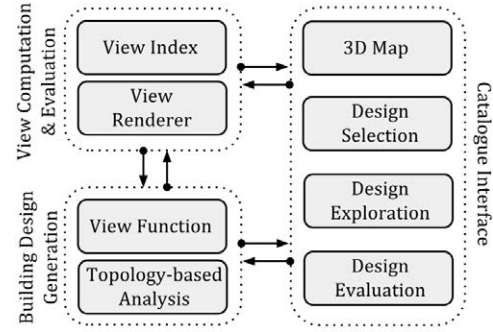4. A novel visual interface that allows the interactive exploration



**Figure 2:** *Framework Overview.*

and evaluation of varying building designs across different categories.

We show the effectiveness of our framework through two use case scenarios set in Manhattan, New York City. The first case study employs our framework to compare the potential views available between two building sites: one in the Financial District and the other in Midtown. This will demonstrate how site specific view types can be identified and prioritized as part of an architectural pre-design phase. These obtained insights are then used in the second case study to inform the schematic architectural design of a hypothetical mixed use tower on the Financial District site.

Note that this work uses 3D context models which closely represent the existing physical conditions of the built environment so that any generated results can be integrated into the design process of actual building projects.

## 1.2 Framework Overview

Our framework, illustrated in Figure 2, is primarily divided into three components – view computation and evaluation; building design generation; and a catalogue interface.

The first component, described in Section 3, handles the task of computing views and creating an indexing scheme that supports the efficient evaluation of views.

The building generation component, described in Section 4, uses techniques from computational topology to produce high performing design variations for a given design category. Here, the building designs are classified into a set of categories based on their geometric properties.

The final component, described in Section 5, consists of a catalogue interface that makes use of multiple linked visualization widgets. This interface allows users to interactively explore optimally performing design variations based on their view scoring preferences. The catalogue visual interface supports the following features:

- Interactively change both the design categories, as well as view preferences to generate the required design variations.

- A shopping cart based interface to allow users to compare designs across multiple categories and view preferences.

- A parallel coordinate based filtering interface to help users filter design variations.

- Ability to explore properties of a given building design, such as view scores distribution and efficiency of building design over the different parts of the building.

- Ability to explicitly verify the views from different parts of the building, and explore the view extent over the city.
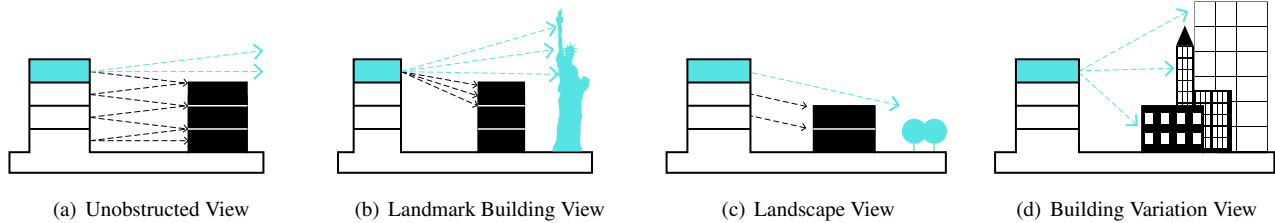
(a) Unobstructed View     (b) Landmark Building View     (c) Landscape View     (d) Building Variation View

**Figure 3:** *Illustration of the different view types that are used to evaluate the views from a building.*

## 2 Related Work

Performance driven design has received a lot of attention recently in the field of architecture. It consists of applying a different set of (usually conflicting) metrics, such as daylight and solar gain [Chronis et al. 2012], to evaluate and select the best architectural designs for a given project [Zheng et al. 2014]. A recent trend is the consideration of these criteria in early stages of design [Kimpian et al. 2009; Keough and Benjamin 2010] to help better inform important design decisions and avoid expensive backtracking further along the design process.

Orthogonal to our work, there has been a lot of research on procedural modeling for architecture [Wonka et al. 2003; Müller et al. 2006; Wonka et al. 2011; Demir et al. 2014; Schwarz and Müller 2015] in which the goal was to ease the generation of varied building designs without any particular focus on performance metrics. In addition to views, important relationships driving the procedural modeling of towers include building core, zoning regulations and building use. While these relationships can be geometrically modeled [Gane and Haymaker 2007], we use rules of thumbs provided by the architects suitable for early design exploration.

Multi-objective optimization techniques have been used as a way to explore parametric design spaces, but it has only recently been adopted in the architecture community as a way to guide initial phases of design [Keough and Benjamin 2010]. Vanegas et al. [2012] used a Monte Carlo Markov Chain (MCMC) based approach to identify procedural models to create urban environments based on user preferences. The selection of multiple designs is achieved by running a number of Markov Chains from different initial positions and collecting the best designs found. A generalization of MCMC, called reversible jump MCMC, is also commonly used used in procedural modeling [Ripperda and Brenner 2006; Talton et al. 2011]. Genetic algorithms have also been used for the optimization process during building design [Caldas and Norford 2002; Chronis et al. 2012].

Visualization techniques are commonly used in the general context of exploring complex high dimensional parameter spaces. Berger et al. [2011] coupled the parameter space with multiple objective functions based on multidimensional projection and predictive analysis. Visual interfaces have also been proposed specific to applications domains, for example to browse through parameter spaces representing shape collections [Averkiou et al. 2014; Talton et al. 2009; Kleiman et al. 2013]. Different from these techniques, the approach used in our work couples visualization and computational topology techniques which enable efficient detection and interactive exploration of interesting building designs based on multiple view criteria.

Topology-based techniques have been frequently used in mesh analyses and processing including topology-based shape matching [Hilaga et al. 2001; Dey et al. 2010], topological simplification and cleaning [Chiang and Lu 2003; Wood et al. 2004; Pascucci et al. 2007], surface segmentation and parametrization [Zhang et al.

2005; Dey et al. 2013]. They are also common in the field of scientific visualization (see e.g. [Weber et al. 2007a; Zhou and Takatsuka 2009; Pascucci et al. 2010]). More recently, topology-based methods have also been used to explore and analyze high dimensional data [Weber et al. 2007b; Harvey and Wang 2010; Gerber et al. 2010; Oesterling et al. 2011]. By providing a succinct representation of the data (or functions), these techniques help not only in the visualization of the data, but also in their efficient processing. We decided to adapt topology-based techniques for the problem at hand due to two main reasons – (i) the set of high performing buildings are naturally represented as the set of maxima in a high-dimensional space; and (ii) these topological features can be computed efficiently.

## 3 View Analysis Framework

As mentioned in Section 1, the quality of views is a critical component in the design of a building especially in a dense urban environment. In order to effectively measure view quality, we first identify four different perspectives, or *view types* (Section 3.1) that capture different qualitative aspects of the view from a given location and look at direction. We then use a rasterization approach to compute the view metrics (associated with each view type) from the visible scene (Section 3.2). This allows calibration and adjustment of the view analysis through subjective evaluation of actual views. As described later, these metrics can be combined to derive view scores that capture multiple qualitative view preferences. Finally, to support the efficient computation of view metrics for a large number of location-direction pairs, we propose a simple yet effective indexing scheme (Section 3.3).

### 3.1 View Types

We have identified the following four view types for our analysis:

**Unobstructed View** captures the notion of *how far* a viewer can see from a given location and is illustrated in Figure 3(a). The view metric associated with this view type is the average distance from a viewpoint to obstructions within a human field of view.

**Landmark View** quantifies how much of select landmarks are visible. For example, the highlighted floor level of the building icon in Figure 3(b) is able to view the top portion of the Statue of Liberty.

**Landscape View** similar to Landmark View, quantifies how much of certain landscape elements are visible. This is illustrated in Figure 3(c). Since preference for landscape type can vary based on location, such as open space or water bodies, the Landscape View has been optionally subdivided into these two view subcategories.

**Building Variation View** quantifies the amount of variation in aesthetic building character within a view. The intuition here is the following: achieving a broad diversity of buildings in a single view is more visually interesting, and potentially more desirable (as illustrated in Figure 3(d)). In order to define such variations, buildings

are grouped into five age categories based on their year of construction. This is done since buildings constructed within the same time periods were typically constructed in a common architectural language with similar materials and massing forms. The associated metric is then defined as the entropy of the quantity of building ages that is present in that view. The more varied the age of buildings in a single view, the higher its value.

We would like to note that, contrary to what is normally expected, not all of above defined views are positively impacted by height. While height certainly improves unobstructed view, it can decrease the quality of other view types. In fact, both landmark and diversity views can be negatively impacted by height. Consider, for example, landmark views. Since views from very high floors literally provide a bird's eye view, it is difficult to differentiate between buildings thus reducing the significance of smaller landmarks (which form a very small fraction of the view). This makes the problem of designing towers to optimize views both interesting and non-trivial.

### 3.2 View Computation

We compute the different view metrics using a rasterization approach. Given a view point and a look at direction, the camera is placed at that point and positioned along the required direction. The scene is then rendered to a texture in which each color channel is used to encode different properties of the scene – distance from camera, material type (water, park, building), building types (landmark, landscape, or regular), and building age. The pixels in the resulting texture are then appropriately aggregated to compute the view metrics.

To compute the view metrics of a building, the building is first divided into a set of fixed sized windows. Here, the height of the window is equal to the proposed floor height of the building. The view metrics are then computed for each window at its center along the direction of the window's outward-pointing normal. The view metrics of the building is then defined as the average value of the metrics over all windows.

### 3.3 Indexing Strategy for View Computation

The number of view metric computations required for a given building is equal to the number of windows in that building. A typical building has several thousands of windows, making it expensive to compute its view metrics. Thus, the process of computing view metrics for a large number of buildings is computationally expensive. This would in turn limit the ability of identifying high performing building designs, since this process would require the view computation from millions of windows (see Section 4).

In order to speedup this operation, we propose a simple, yet effective indexing scheme (illustrated in Figure 4) that allows for the efficient computation of a good approximation of the view metrics.

**Building the Index.** Given a building site, we first create a hexahedral tower which is the extrusion of the bounding box of the development site. The height of this tower is equal to the maximum permissible height (for NYC, this values is equal to 2,000 feet as set by the Federal Aviation Administration). Such a tower has a property that it would envelope all the candidate buildings for that site. We are interested in computing views inside this volume, which we refer as the *envelope*. A regular volumetric grid is created from this tower, where the nodes of this grid correspond to the different *view points*. The view metrics are computed for each of these points along a discrete set of view directions surrounding that point. Figure 4 shows the top view of a horizontal slice of this volume. The view points are colored gray, and the view directions from which
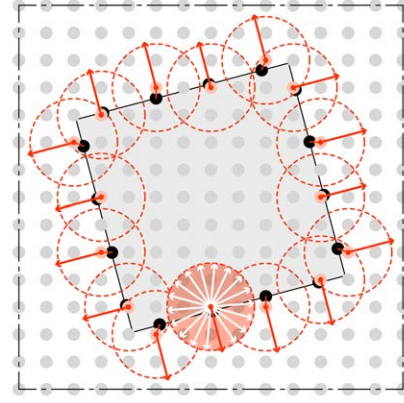


**Figure 4:** *Illustration of our indexing scheme for efficient computation of view metrics. One slice of the envelope is shown together with a set of grid points (gray) and a set of sample directions (orange circles). Given a pair of a point location and a look at direction, the view metrics are approximated by retrieving the values associated with the closest point-direction pair in the index. This process is illustrated for the different windows of the building (black points) along the direction normal to each window.*

the view scores are computed are represented as circular disks in the figure.

Each view point together with the view direction can be encoded using 4 variables, namely the coordinates $(x, y, z)$, and the view direction angle $\theta$. Note that we assume that the view direction is horizontal (i.e., parallel to the ground). Computing the view metrics of all the view points along the different directions results in a multi-variate function defined on 4D space. Based on the accuracy preference of the user, the resolution of the envelope grid and the number of view directions can be optionally varied.

**Evaluating view metrics using the index.** Given a point $p$ contained in the envelope and a view direction $d$, the corresponding view metrics are computed as follows. The 4D point defined by $p$ and $d$ is used as the key to look-up in the index. This look-up process, when using the nearest neighbor to approximate the given point, is highlighted in Figure 4. Note that this look-up for a given 4D point is a constant time operation. We later show in Section 6, that this indexing scheme is efficient and also results in a very good approximation of the actual view metrics.

## 4 Topology-based Design Classification

The key idea in our framework is the use of concepts from computational topology to efficiently identify the parameters of different design variations that maximize the view scores. We now describe the view function defined on the design parameter space and explain how the topology of this function is used to identify possible design variations for the catalogue.

### 4.1 View Function

In this paper, we focus on buildings that are towers formed by connecting multiple profile curves at different heights. Each of these profile curves can be appropriately transformed via different operations such as scaling, rotation, etc. to obtain different building designs. The view scores of such buildings are dependent on the extent of each of these $d$ operations. Assuming the magnitude of each of these operations is in $\mathbb{M} = [0, 1]$, the set of possible building designs is represented by the $d$-dimensional design parameter space $\mathbb{M}^d$. The *view function* is then defined as $f : \mathbb{M}^d \rightarrow \mathbb{R}$, a real valued
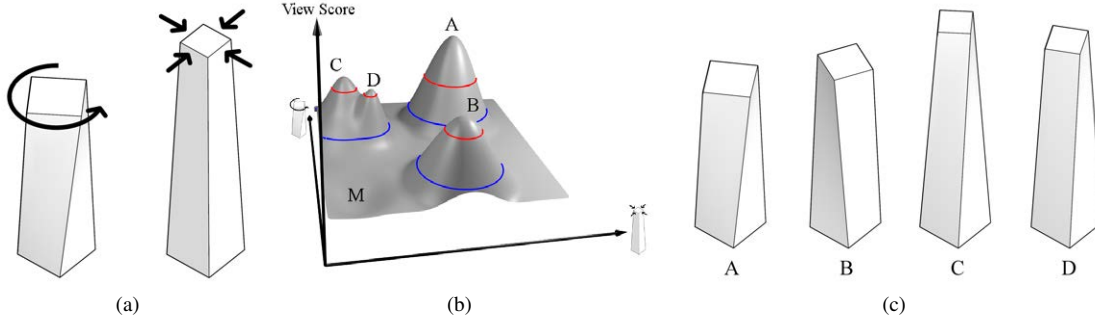
**Figure 5:** *An Example view function.* *(a) A 2-dimensional parameter space is defined using the rotate and scale operations on a hexahedron. (b) The view function is defined on the parameter space and is represented as a terrain, where the height indicates the view score. (c) The design variations corresponding to the four maxima of the view function.*
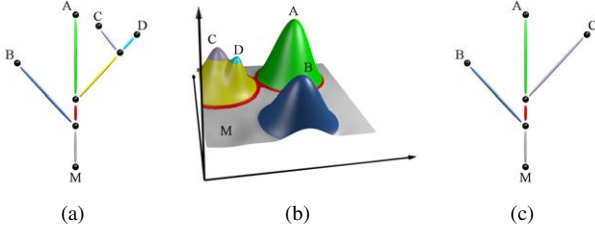


**Figure 6:** *The contour tree is used to identify the design variations as the topological features of the view function. (a) The contour tree of the 2-dimensional view function in Figure 5(b). (b) The influence regions of the extrema are highlighted using the same color as the corresponding leaf edges of the contour tree. (c) The simplified contour tree after removing the maximum D.*

function that maps each point of the $d$-dimensional parameter space to the view score of the corresponding building.

For example, consider a building design that is a hexahedron which is constructed by connecting two squares. For ease of exposition, consider the following two operations – scale and rotation of the top square of the hexahedron as illustrated in Figure 5(a). These two operations form a 2-dimensional parameter space, $\mathbb{M}^2$, of building designs. A sample view function defined on this space is illustrated as a terrain in Figure 5(b). The height of a point in this terrain represents the view score of the corresponding building. The set of operations that constitute the different dimensions of this space are described in detail in Section 5.

### 4.2 Topology Background

Consider the 2-dimensional view function in Figure 5(b). The different "peaks" in this figure correspond to parameters that locally maximize the view function. For example, the building designs for the four peaks in Figure 5(b) is shown in Figure 5(c).

Generalizing this to higher dimensions, our goal is to use the set of *significant maxima* of the view function to classify the possible building design variations that can be used by architects. In order to efficiently compute the candidate set of maxima, we propose to use a topological abstraction of the function called the contour tree.

**Level set topology and Contour trees.** Given a smooth real-valued function $f$, the *critical points* of $f$ are exactly where the gradient becomes zero. *Regular points* are those which are not critical. The preimage $f^{-1}(a)$ of a real value $a$ is called a *level set*. It is the set of all points on the domain having function value equal to $a$. Two level sets, in blue and red, of the 2-dimensional view function are highlighted in Figure 5(b).

The contour tree of a function $f$ defined on a simply-connected domain tracks the evolution of the topology of its level sets with changing function values [Carr et al. 2003]. Topological changes occur at critical points, whereas topology of the level set is preserved across regular points [Matsumoto 2002]. Formally, the contour tree is defined as the quotient space under an equivalence relation that identifies all points within a connected component of a level set.

Figure 6(a) shows the contour tree of the view function shown in Figure 5(b). The nodes of the contour tree correspond to critical points of the function. In particular, leaves of the contour tree correspond to the set of maxima and minima of $f$. The advantage of this structure is that it can be used to efficiently identify the set of significant maxima, where the significance is defined using *topological persistence*.

**Topological Persistence.** Consider a sweep of the function $f$ in increasing order of function value, as mentioned earlier, topology of the level set changes at critical points during this sweep. In particular, at a critical point, either a new level set component is created or an existing level set component is destroyed. A critical point is a creator if new topology appears and a destroyer otherwise. It turns out that one can pair up each creator $c_1$ uniquely with a destroyer $d_1$ which destroys the topology created at $c_1$. The *topological persistence* [Edelsbrunner et al. 2002] of $c_1$ and $d_1$ is defined as $f(d_1) - f(c_1)$, which intuitively indicates the lifetime of the feature created at $c_1$, and thus the importance of $c_1$ and $d_1$. In the 2-dimensional case, the persistence of a maximum (minimum) is equal to the height (depth) of the peak (valley).

### 4.3 Design Classification

In order to support efficient representation of the parameter space, we model the high dimensional space $\mathbb{M}^d$ as a $k$-nearest-neighbor graph $\mathbb{G}_d$ of a set of points sampled uniformly from $\mathbb{M}^d$. The view function is then defined as a piece-wise linear (PL) function $f_v : \mathbb{G}_d \rightarrow \mathbb{R}$, where the function is defined on the vertices of $\mathbb{G}_d$ and linearly interpolated within each edge. The contour tree of $f_v$ is then computed using the sweep algorithm [Carr et al. 2003]. This algorithm takes $O(n \log n + m\alpha(m))$ time, where $n$ and $m$ are the number of vertices and edges, respectively, in $\mathbb{G}_d$.

The contour tree is then used to simplify the function to remove low persistent (noisy) critical points. This is accomplished by removing low-persistent leaves from the contour tree [Carr et al. 2004]. This amounts to smoothing the input function in the region corresponding to the removed edges. For example, the peak corresponding to maximum $D$ is small (and thus has low persistence). It is removed (Figure 6(c)) to generate the de-noised function which smoothes out

the region corresponding to *D* (colored cyan in Figure 6(b)). The persistence based simplification using the contour tree can be performed efficiently in $O(n \log n)$ time [Carr et al. 2004]. When two maxima have similar function values but are close to each other in the input domain, one of them typically has a low persistence value. Such maxima are therefore removed during the simplification process. Intuitively, this filtering is equivalent to retaining a single design variation from a set of designs having similar properties (by being close to each other in the parameter space).

Once the function is simplified to remove noise, the resulting set of maxima forms the candidate building designs of the catalogue for a given category (see Section 5.2). For the view function in Figure 5(b), the building designs corresponding to the three remaining maxima, A, B and C, form the three design variations for the catalogue. The simplification threshold can optionally be specified by the user interactively. Further, given a minimum view score, the remaining maxima can also be filtered to retain only those having view score greater than the minimum score. We use a default threshold of 0.01, which removes all critical points having persistence less than 1% of the highest function value.

# 5 Design Catalogue

For a given project, architects select a building type for the context and program (such as office or residential), and then come up with a handful of design categories to develop. The tower designs considered during the early design stage are typically constructed through a set of simple operations, and are later refined relative to context and program. We select a range of design categories and operations representative of early tower design exploration, which we first describe in this section. The design variations, also referred to as *massings*, resulting from a combination of these operations are then used to generate the parameter space for the topological analysis. Finally, we describe the catalogue interface using which architects can explore design variations across different categories.

## 5.1 Design Categories

In this work, we are interested in towers which are generated through a series of profile polygons. The profile polygons are either connected through lofting one profile to the next, or extruding each profile vertically up to the height of the profile above (see Figure 7). We assume that all the profile polygons for a given design have the same number of sides. The profile polygons could optionally have *filleted* (or rounded) corners allowing for a spherical facade. We assume that a uniform fillet is applied on all corners of the profile polygons. While we assume convexity, the profile polygon need not be regular. We use the term *block* to indicate the part of a tower between two consecutive profile polygons.

The above properties are then used to classify possible design variations into a set of design categories. Each design category is uniquely defined using a 5-tuple, whose elements are: (1) number of sides in the profile polygons; (2) number of blocks; (3) profile polygon connectivity – loft or extrude; (4) building facade – fillet or no fillet; and (5) polygon shape – regular or not.

## 5.2 Design Operations and Parameter Space

Given a design category, we support a set of geometric operations that can be applied on the profile polygons to obtain design variations. The different supported operations are summarized below:

**Scale:** In case the given category should have regular shape, a profile polygon is uniformly scaled. Otherwise, the profile polygon is scaled non-uniformly along the *x* and *y* axes.
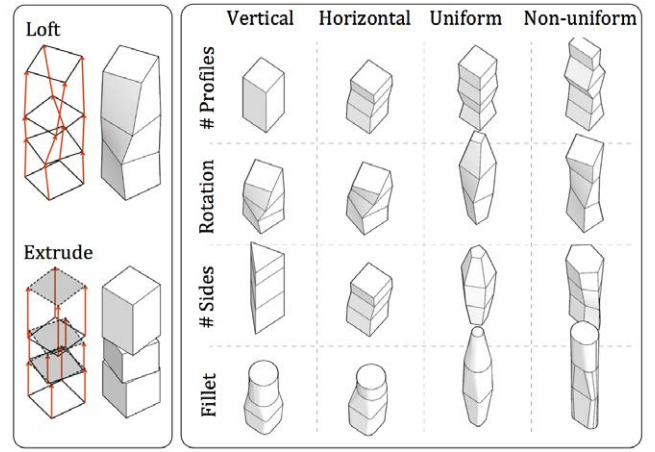


**Figure 7:** *Design variations for a proposed tower building are obtained though a set of operations that are performed on profile polygons. It is typical for architects to start the building design process with variations on primitive shapes. A high level design category is first chosen based on whether the tower is to be constructed using the **loft** operation, or the **extrude** operation. A subset of the possible design variations when the tower is constructed using the loft operation is shown on the right. Here, the design variations were obtained using combinations of two operations.*

**Rotate:** A given polygon can be rotated within a permissible range with respect to the polygon below it. This is to ensure that the "twist" that occurs always results in valid structures.

**Horizontal movement:** The profile polygon can be translated horizontally. The maximum possible translation is determined by the intersection to adjacent profile polygons, and is ensured to be within the development site.

**Vertical movement:** The designed tower has to satisfy zonal constraints such as the floor-area ratio (FAR), which is the total area allowed as a factor of the lot area. Thus, having profile polygons with large areas will result in shorter towers and vice versa. Therefore, the vertical position of a profile polygon is implicitly determined by the number of floors in the corresponding block, and hence, the vertical movement can be controlled by the total floor areas that is assigned to each block. To avoid arbitrarily high towers, zoning regulations usually dictate the maximum allowed height.

Figure 7 illustrates a subset of design operations together with the possible design variations when using a tower that is generated by lofting one profile to the next.

The permissible set of operations for each of the categories determines the dimension *d* of the corresponding design parameter space $\mathbb{M}^d$. Each of these operations contribute to either one or two dimensions of $\mathbb{M}^d$. For example, the scaling operation contributes to one dimension per profile polygon for a category with regular polygon shape, while it contributes to two dimensions per profile polygon when non-uniform scaling is used. On the other hand, the rotation operation always contributes one dimension to $\mathbb{M}^d$.

A building variation corresponding to a *d*-dimensional point is generated by performing the geometric operations defined by the coordinates of that point. We always start this generation with a base set of profile polygons that are regular having the given number of sides. In this paper, we fix the number of sides of the profile polygons to be four.
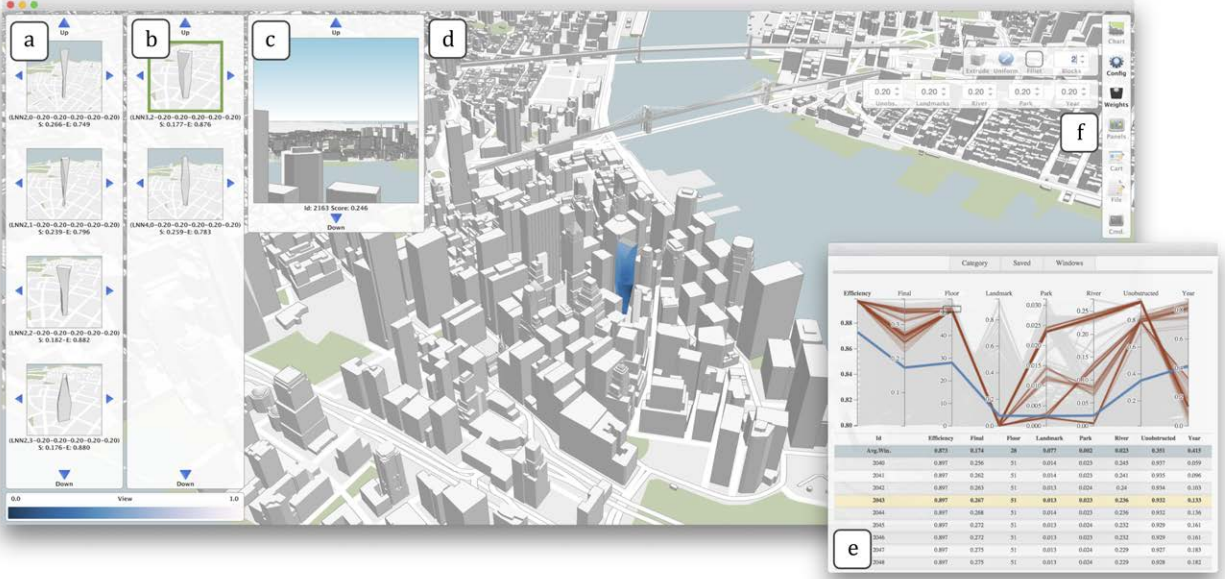
**Figure 8:** *The different components of the catalogue exploration interface. **(a)** Summary panel. **(b)** Building cart. **(c)** View panel. **(d)** 3D map. **(e)** Design exploration chart. **(f)** Design selection controls.*

## 5.3 Catalogue Exploration Interface

We develop a visual exploration interface that allows users to explore and evaluate the design variation across various design categories and view preferences. The interface, shown in Figure 8 and the accompanying video, is composed of three main components:

**Design Selection.** This component allows users to select the design category and set their view preferences. A design category is selected by changing the set of design properties. The view preference is specified by providing weights for the different view metrics. The view scores of the different buildings in the parameter space is then computed as the weighted sum of the corresponding view metrics to obtain the view function.

**Design Exploration.** This component allows the user to explore the identified design variations. It consists of three linked visualization widgets – a *summary panel*, a *building cart* and a *3D map*. Once the set of significant maxima of the view function are computed, the summary panel visualizes the corresponding building geometries using a color map. The building cart acts as a shopping cart, allowing users to save designs of interest. The designs in the cart can span across different categories and view preferences.

Users can view a building design in the context of its neighborhood on the 3D map by selecting the design of interest from either the summary panel or the building cart. The context provided by the 3D map allows users to understand the possible view opportunities of different designs and help in the evaluation process. This can help inform manual refinement of the designs.

**Design Evaluation.** This component allows users to evaluate and compare the performance of the design variations present in the summary view as well as the building cart. It is composed of two linked visualization widgets – a *design exploration chart* and a *view pane*. The design exploration chart uses parallel coordinates to allow the user to compare and filter building design variations. Each axis of the chart represents one view metric.

The design exploration chart can be used in two modes – *building mode* and *window mode*. In the building mode, the set of high dimensional points (lines of the chart) represents the building designs currently loaded in the summary panel or in the buildings cart. In this mode, there are two additional axes in the parallel coordinates representing the building efficiency and the floor overlap area of the corresponding design. The *building efficiency* is defined as the ratio of the area of the building core to the total built area. The building core is primarily comprised of the elevator shafts, stairs, and mechanical equipment. The *floor overlap area* is the area of the intersection between all floors of the building. A feasible design should have overlap area greater than the area of the building core. In the window mode, high dimensional points represents the windows of the selected building. This mode has an additional axis representing the floor corresponding to a window, and helps users select windows of interest. The view, as seen from the windows that are selected using the chart, can be inspected using the view pane. Additionally, the user can also compute the *reach* with respect to a selected set of windows, which essentially highlights all the buildings that can be seen from these windows.

**Other features.** In order for our framework to be included in the actual architectural design workflow, the interface also supports exporting building design variations together with the associated view performance information. Importing this data into common 3D modeling software allows architects to calibrate the design options relative to other criteria such as zoning regulations (not part of this work), while still maintaining high view performance. It also supports importing building designs so that they can be evaluated and compared with those generated using our technique.

## 6 Implementation and Experiments

The design catalogue was implemented using C++, Qt 5.4, and OpenGL 4.3. We now briefly describe the choices made during the implementation, and discuss results from our experiments evaluating the performance of the index.

### 6.1 Implementation

**Preprocessing.** As discussed in Section 3.2, we first pre-compute the view index for the building site. Computation of the view metrics is essentially the aggregation of values encoded in the color
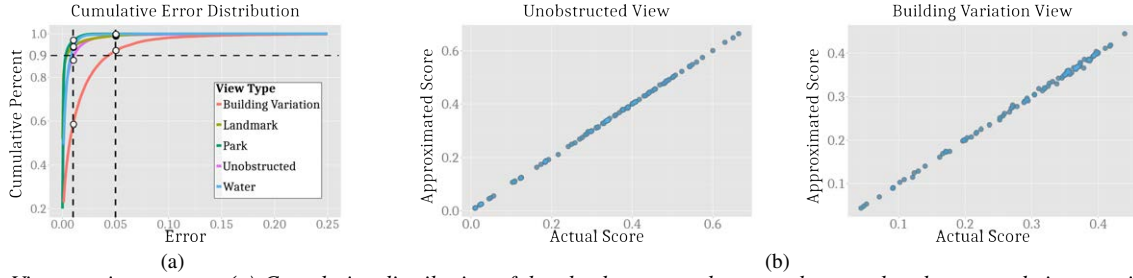
**Figure 9:** *View metric accuracy. **(a)** Cumulative distribution of the absolute errors between the actual and computed view metrics. Note that the error is less than 0.05 for over 90% of the points for all metrics. **(b)** Comparison of view metrics at a building scale.*

channels of the pixels of the view, which can be easily parallelized using the GPU. We make use of *compute shaders* for this computation which is part of the OpenGL pipeline. This also helps avoid the costly memory transfers between the GPU and CPU.

**Design parameter space** $\mathbb{G}_d$**.** We use Latin hypercube sampling [McKay et al. 2000] to generate sample points in $\mathbb{M}^d$. We then construct the discrete domain $\mathbb{G}_d$ as the $k$-nearest neighbor graph of these points. In our experiments we use $k = 20$. We find that this value provides a reasonable approximation of the connectivity between points in the space.

**Computing the view function.** Even though computing the view score of a single window from the index is extremely efficient (see next section), computing the various view functions typically requires the view score computation of several tens of millions of windows. However, the number of possible design categories are finite for a given number of blocks and number of sides of the profile polygon. Therefore, when preprocessing a given development site, we also compute and store the view metrics for the different possible parameter spaces corresponding to the different design categories. Thus, when users update their view preference, the view function will only have to be computed from the view metrics, which will in turn be used to compute the topological features. Given that this is an embarrassingly parallel operation, we utilize the multiple CPU cores available to further speedup this computation. Alternatively, the GPU could also be used for this purpose.

**Context 3D map and buildings.** Our map visualization makes use of geospatial data extracted from OpenStreetMap [Wiki 2015] to render the context map layer in real-time. OpenStreetMap (OSM) is a community-built database that aims to provide a complete mapping of data across the world with many rendering frameworks available, notably the rasterized, tile-based "slippy map". However, they are only limited to 2D rendering (with some being 2.5D, e.g. showing a fixed angle view) with pre-rendered images. In particular, it would not be possible to produce arbitrary 3D visualizations as illustrated in Figure 1, let alone interactive addition of customized building designs and annotations. Thus, we chose a direct, vector-based rendering approach to provide both flexibility and scalability for our mapping framework.

For the purpose of our application, we are only interested in the road networks, land use and buildings tags of the OSM database. While high resolution CAD models are available for popular buildings in New York City [SketchUp 2015] such as the Empire State Building or the Freedom Tower, the repository is far from complete. On the other hand, buildings in OSM are updated daily by the community by either tracing aerial imagery (e.g. Bing Maps) or through street-level observations and measurements, thus, providing reliably accurate information. Our system supports both CAD models and OSM buildings to allow high-quality rendering of landmark structures, and to efficiently manage a complete view of the context buildings, respectively. Since building shape in OSM is defined implicitly by a set of outlines, extruding heights, and roof

types, we render them using tessellation shaders to minimize the data storage and transfer bandwidth.

## 6.2 Experiments

The experiments were performed on a workstation with a Intel Xeon E5-2650 CPU, 32 GB RAM, 3 TB of disk storage, and an Nvidia GTX 680 graphics card having 3 GB of GPU RAM. The experiments reported next were performed using the two potential development sites that were selected by the architects (see Section 7).

**Performance analysis.** As mentioned in Section 3.2, we encode the different view properties in the color channel while rendering the scene in order to compute the view metrics. We get on an average around 130 fps to render the views at a resolution of $256 \times 256$. The resolution of the grid for the view index was 10 ft × 10 ft along the *xy*-plane, and every 12 ft above the ground (which is equal to the floor-to-floor height). This ensures that the sampled points are aligned with the different floors of any building on that site. Also, at each sampled point, the view was computed along every 6°, resulting in 60 directions for each point. The computation of the index therefore required the computation of 6.4 million views and 7.48 million views for the two sites respectively. This took a total of 860 minutes and 936 minutes respectively. This time includes the time used by the compute shader. For a single image, the time spent on the compute shader was significantly less than a millisecond.

The approximate view metric computation, which is essentially a constant time index look-up, was possible on an average of 33.3K computations per second on a single CPU, over two orders of magnitude faster than the exact computation.

**Index accuracy.** The approximate view metrics computed using the index, in addition to being efficient, should also be reasonably accurate in order to be beneficial. To test the accuracy of our approximation, we compared the view metrics computed using our index against the actual view metrics. The experiment was performed on a set of 20000 random points, 10000 from each site along random directions. Figure 9(a) plots the cumulative distribution of the absolute error for the different view metrics. Note that the error is less than 0.05 for over 90% of the points for all view metrics. In fact, for metrics other than the building variation view, the error is less than 0.01 for close to 90% of the time. This is because, the building variation view is highly dependent on the number of different buildings that are visible, and even a slight deviation would have greater impact on the score when compared to other metrics.

The view function is computed on a building scale, where the view score of the building is the average of the view scores over its windows. Thus, the impact of the above errors is even less when considering them at a building scale. For example, the scatter plots in Figure 9(b) assess the accuracy of the index at a building scale for the unobstructed and building variation views respectively, which had the highest error margin in Figure 9(a). This experiment was
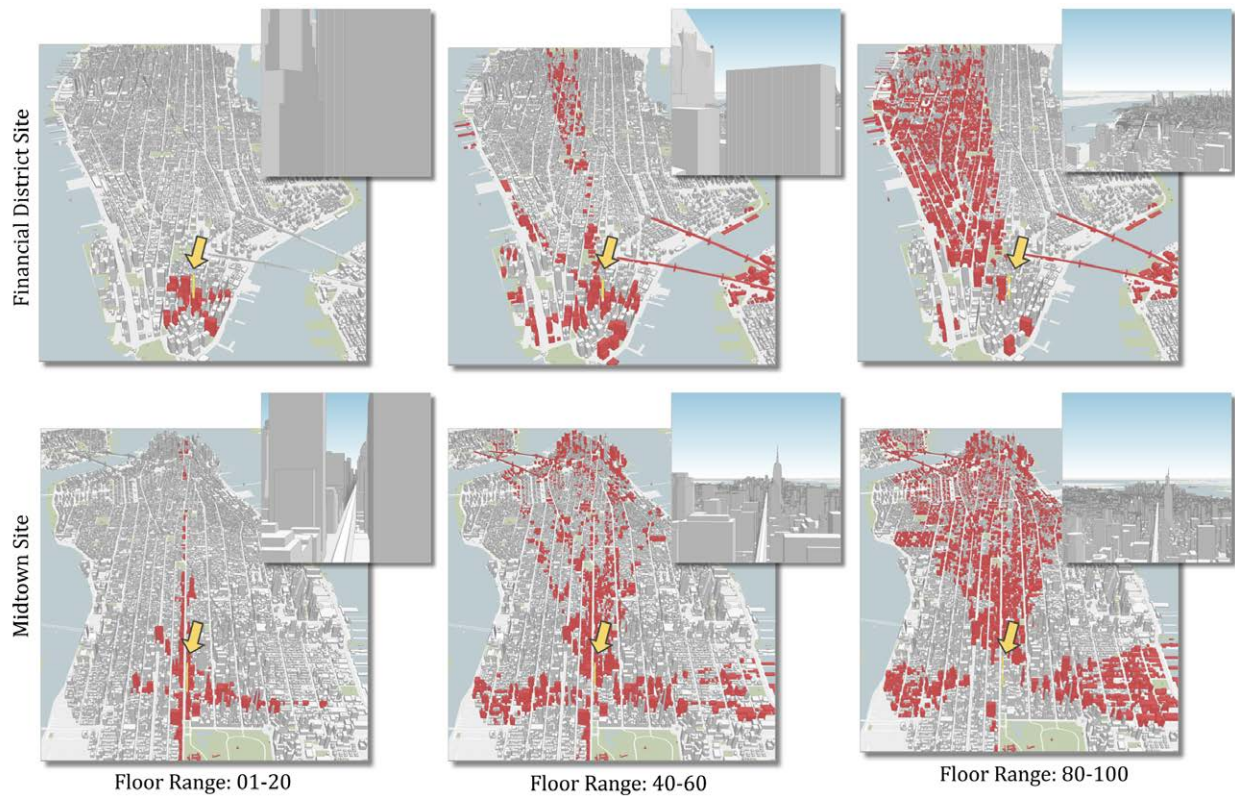
**Figure 10:** *The views obtainable from different height zones (20 floors each) from the two sites in Financial district and Midtown. The buildings highlighted in red indicate the reach from these zones. The inset shows the typical view from a representative window.*

performed for a random set of 100 building designs, and the *maximum* absolute error for these two view metrics was $7 \times 10^{-3}$ and $10^{-2}$ respectively.

While increasing the grid resolution of the index would improve the precision of the approximation, it increases the index creation time. However, the lookup time, being an $O(1)$ operation, does not change. In this paper, we chose the resolution that provided a good trade-off between precision and index creation time.

## 7 Case Studies

Architects begin with a program (office or residential) and a site, both provided by the client. The first part of the architectural design and development workflow is an analysis of the site by test fitting the program to understand site limitation and opportunities relative to the required program. Based on the results of the analysis the architect will generate three to five simple, but distinct design schemes. These schemes are then refined relative to client feedback, zoning regulations, structural requirements and environmental goals. Finally, a single scheme is selected to develop further, eventually being constructed.

To demonstrate the capacity of our framework we present two use cases which illustrate its usefulness within the above workflow. First, in Section 7.1, we use our framework to compare the possible views that can be obtained on two different sites, one in Financial District and the other in Midtown. This will help developers to identify view types that have to be prioritized for a given site. Next, in Section 7.2, using the insights gained from the above analyses, we focus on the Financial District site and use our framework in the schematic architectural design of a hypothetical mixed use tower.

### 7.1 Use Case 1: Site Comparison

Multiple sites within the same urban environment can often have radically different view potentials depending on the built environment. Both of the sites selected for this comparative analysis are surrounded by tall, high density development. The Financial District site (Site 1) is a full block in an organic grid configuration with four narrow street frontages. In contrast, the Midtown site (Site 2) is a double corner lot at the east end of a typical Manhattan block with frontage along one wide avenue and two narrow cross-streets. Figure 10 highlights the location of these sites. To evaluate the set of views possible, we use a simple one block building with the extrude option and a no-fillet facade. We then use the maxima identified from the topological analysis to understand the kind of views possible from each site.

Both sites have views with close distance obstructions at lower levels. Assuming a typical 12 ft floor to floor height, the unobstructed views start improving at around the 20th floor for Site 1, and the 30th floor for Site 2. This is predominantly due to existing building streetwalls opposite the narrow cross-streets bordering each site. Exceptional views, achieving an average view depth of 400 ft or greater, begin for Site 1 at approximately the 40th floor facing east, and for all orientations above the 70th floor. This can be slightly improved when the tower faces are rotated towards the street intersections rather than aligning with the lot lines. Interestingly, the tallest tower in the immediate context of Site 1 which negatively impacts unobstructed views is the Trump Building, a designated New York City landmark. Views of this building are achievable at all floors from the northwest corner of the site across the street intersection. In contrast, landmark views for Site 2 to the Empire State Building or Rockefeller Center begin as narrow, partial views at the 20th floor and significantly improve above the height of the surrounding context at the 40th floor.
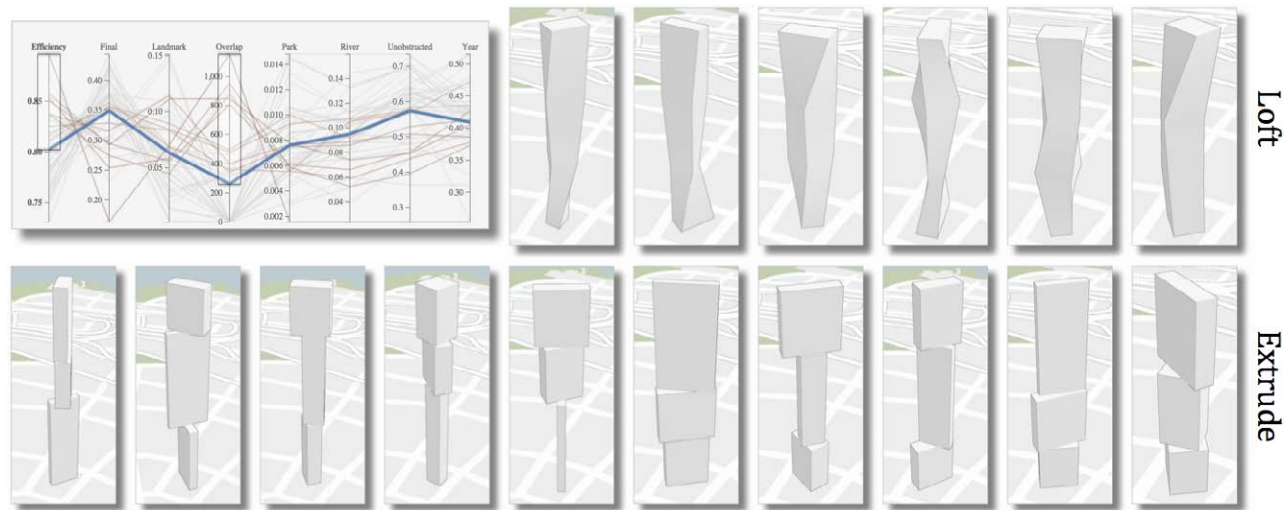
**Figure 11:** *Filtering building designs to retain only those having above average values for efficiency and overlap resulted in a total of 16 designs across the loft and extrude categories. The average performance is indicated using the blue line in the parallel coordinate chart.*

This is further corroborated using the *reach* functionality of the catalogue interface. The facade surfaces within three, 20-floor zones were evaluated for their view reach shown in Figure 10. The maxima with the highest unobstructed view was chosen for this analysis. The inset in the figure shows a representative perspective view from one window each on floors 20, 60 and 100 respectively.

Battery Park is located a quarter mile south of Site 1. However, evaluating for visibility to open green space reveals only low scoring views, through narrow gaps between buildings, could be achieved starting at the 45th floor. This is in part due to the obstruction caused by the surrounding tall development configured within an organic city grid. On the other hand, this site is located near the southern tip of Manhattan where the East River and Hudson River converge. River visibility is found to be a more contextually appropriate Landscape View evaluation method for this site since the surrounding body of water is both continuous and greater in surface area than Battery Park. High quality river views begin on this site facing southeast at the 40th floor.

Site 2 is also located a quarter mile from an open green space: Central Park. Multiple tall buildings are located between the park and the site with the potential to impede these views, however their configuration on the more regular Manhattan city-grid was found to accommodate better viewing opportunities. The Landscape View evaluation for Site 2 identified a slight northwest orientation, out of alignment with the city grid, which achieves the largest amount of high scoring park views. This specific rotation opens up view opportunities, starting at the 25th floor, through a diagonal mid-block view corridor. While Site 2 is located a greater distance from either river than Site 1, high quality river views are still achievable around the 50th floor.

A similar analysis process evaluating for views with Building Variation revealed that both sites have multiple opportunities for views with a diverse composition of buildings.

In summary, Site 1 was found to provide high quality views at lower floor levels than Site 2 for the Unobstructed View, Landmark View and Landscape River View types. It was also revealed that only low quality park views are available on Site 1 while Site 2 has the potential to achieve high quality park views at floor levels well below the height of the tall building obstructions in the immediate context.

## 7.2 Use Case 2: Building Design

We now focus on the Financial District site to demonstrate the capacity of our tool in the schematic design of a hypothetical mixed use tower with high view performance. As the designers for this project, we use the insight on view trends gained from the first use case to prioritize certain view types and design categories while exploring the possible design variations.

**Design Configuration and View Preference.** For this project, our goal is to have a tower with three multilevel programmatic zones one on top of the other: hotel in the lowest zone, service apartments in the middle, and high-end condominiums at the top. To contain the three zones within a single building form we will explore design categories with three blocks generated via both the extrude and loft operations.

We prioritize unobstructed view, landscape river views, and building variation views based on the assessed view potential for the site and hypothetical designer and client preferences. Accordingly, we weight the view metrics to consider the following three view options: **(1)** Unobstructed 0.4, River 0.4, Building Variation 0.2; **(2)** Unobstructed 0.4, River 0.3, Building Variation 0.3; and **(3)** Unobstructed 0.3, River 0.4, Building Variation 0.3.

From among the building variations for each category and view preference combination, we only select buildings having both efficiency and overlap value greater than the corresponding averages to find commercially viable buildings. This process for Option 1 is illustrated in Figure 11, which resulted in 6 lofted and 10 extruded buildings. Repeating this process for the other view options resulted in a total of 48 buildings which were then saved to the building cart.

**Identifying Good Designs.** Determining which building option to develop further requires the comparison of the view performance and testing the feasibility of planning a commercially viable building using the design. We first filter the saved set of designs to retain only buildings having view metrics for river, building variation, and unobstructed view above their respective averages. Figure 12 shows the different designs from this set that have the best performance with respect to the view types of interest.

By looking at designs that have the best view for the different view types (from among the filtered designs), we found that designs having higher building efficiency typically have a lower view scores. Our focus is to find an option which is balanced, that is, has good
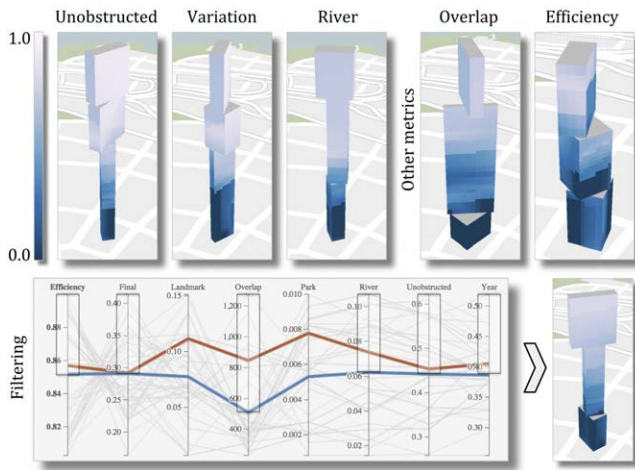
**Figure 12:** *High performing designs across view types are considered, and a design that maintains good balance between the view metrics and building efficiency is selected for further refinement. The red line in the parallel coordinate chart indicates the performance of the selected design while the blue line indicates the average performance of the designs being considered.*



**Figure 13:** *The selected design is appropriately calibrated to ensure that it conforms to the program.*

efficiency as well as consistently higher values over all view metrics. We accordingly select the building design that maintains a balance between building efficiency and the view scores, highlighted in the parallel coordinate chart in Figure 12.

**Design Calibration.** Further geometric refinement is performed on the selected building design to ensure that the other criteria, such as structure or zoning, are also maintained. We use Grasshopper and Rhino for this process. In particular, we calibrate the overall design for the location of the building core, usability of individual floors, and view opportunities (obtained from our earlier analysis). For example, while fitting the building core into the design, the blocks might need some minor alignment. Parts of the massing are further adjusted based on their respective use (e.g., hotel or residential) and mechanical floors are introduced. Mechanical floors are dedicated floors containing mechanical and electrical devices used in the operation of the entire building system. This process for one such refinement is illustrated in Figure 13.

In addition to the view metrics from individual windows of a building, the exported data also contains information about the best view direction for each window. We use this information to design a facade system where individual window panels are tilted to take advantage of specific view opportunities (Figure 14(a)). Figures 14(b) and 14(c) show two refined building design options that are generated from the design selected in Figure 12. When compared to the original design, the performance of refined the designs improves with respect to the river and unobstructed views (see Figure 14(d)).

Finally, the design in Figure 14(b) (shown along with its context in Figure 1) is chosen to be developed into a building. Figures 1 and 14(e) show the final building that is proposed to the client.

# 8 Discussion and Conclusions

We now briefly discuss the limitations of our technique and outline directions for future work.

**Assumptions during view computation.** We assume that the views are not occluded from the same building for which they are computed. However, this is not always true especially for extruded buildings. For example, a large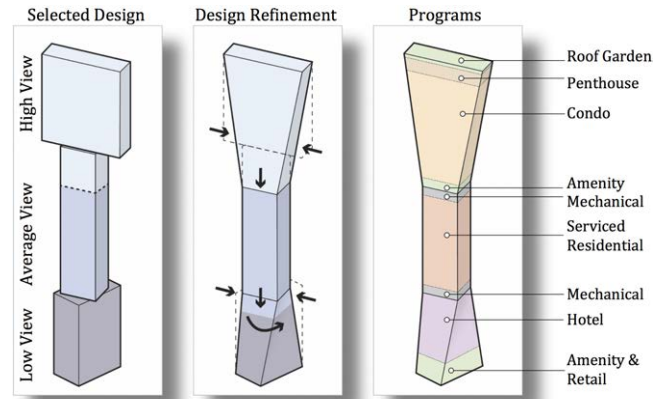r block could occlude views from a smaller block below it, especially on floors close to it. This can be overcome by using a two pronged approach which first identifies view points having such issues, followed by explicitly computing the views for these points.

Another assumption we made is with respect to view direction. We assume that the direction is always normal to the view window. This could miss the possible views that could be obtained by looking downward from a window. However, adjusting the view angle to be slightly larger than a human field of view could account for moderate head tilting. Alternatively, different aspects of the views could also be captured by using a weighted field of view.

**Improving view computation.** Currently, when calculating view metrics, we cannot obtain the actual views; just the values. Thus, when the user requests for views from a particular point, we have to recompute the views from that point. This could be overcome by using light fields [Levoy 2006] to store pixel values for all possible directions and using it for both view generation as well as view metric computation. The main challenge in using this approach is the space required for storing the light field. In addition to the regular pixel color, we also need to encode other details required for view metric computation for each ray. Given the size of the building site and the resolution at which we require the views, we estimate several tens of giga-bytes of space required to store the light field for a single site. It will be interesting to explore encoding which can efficiently store and retrieve such a light field.

**Customizing view functions.** Our current implementation assigns the same view preference (weights) to all floors of a building. It would however be interesting for different floors to have different view preferences, allowing architects to customize different floors of the buildings to particular types of views. This can be accomplished by weighing the view scores based on the floor when computing the building's score, and can be easily incorporated into our framework when generating the scalar function for topological analyses.

**Improving building design quality.** Due to the sampling of the parameter space that is performed, the identified maxima need not be an actual maxima in the continuous parameter space but would be close to it. This is however acceptable since architects are interested in identifying diverse design options with good performance, not necessarily the best options. An additional step of optimization performed using an identified maximum as seed point and traversing the parameter space towards the actual local maximum could produce the best design options as well.
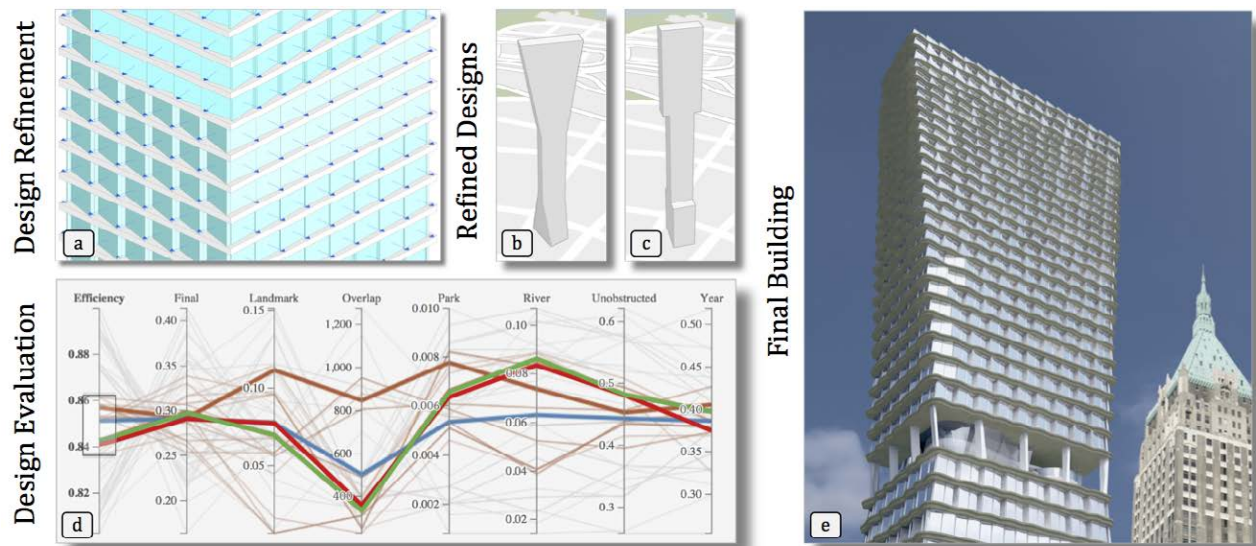
**Figure 14:** *After calibrating the selected design, it is further refined to fine tune the views form different windows (a) to obtain refined designs (b) and (c). The refined design performs better than the original design with respect to the river and unobstructed views (d). Here, the red and green lines corresponds to the refined designs while the brown line corresponds to the originally selected design. Finally, one of the designs is selected to be developed for the client (e).*

**Factors affecting tower designs.** In the design of a tower there are many interdependent relationships that determine the size, shape and design. For example, the floor plate size is determined by the use and the core size. In NYC the distance from the core to the exterior wall in a residential tower is typically 25–35 feet, but is 45 feet for office. The number of elevators in the core also varies depending on the program. The depth of the floor plate then drives the structural systems, which then drives the floor to floor height as a longer span results in deeper structure and hence a higher floor to floor. Similarly, as mentioned in Section 5.2, floor plate size together with FAR influences the height of the building. While we consider FAR in the building generation process, we do not procedurally model the building core. Instead, the building efficiency metric allows architects, through the interface, to filter for viable designs based on the program. The actual fitting of the core into the design is done later during the design calibration process.

**Handling the dynamic nature of cities.** As cities evolve with time, some views are more likely to be affected by new constructions than others. One interesting direction for future work is to use predictive models and simulate the consequences of future developments.

**Conclusions.** In this paper we proposed a topology-based technique to identify high performing building design options with respect to view quality. To efficiently accomplish this, we first defined ways to quantify different perspective views and transform the problem of computing the corresponding metrics to an efficient texture lookup operation, which was then used to compute the view function. The topological features of the view function was then used to identify a diverse set of high performing building designs. We also developed an interactive visual interface to explore the design space based on a user's view preferences. The effectiveness of the proposed framework was demonstrated through use case scenarios involving real world building sites, and it is currently being used by our practicing architect collaborators.

## Acknowledgements

## References

AVERKIOU, M., KIM, V. G., ZHENG, Y., AND MITRA, N. J. 2014. ShapeSynth: Parameterizing Model Collections for Coupled Shape Exploration and Synthesis. *Computer Graphics Forum 33*, 2, 125–134.

BENSON, E., HANSEN, J., SCHWARTZ, ARTHURL., J., AND SMERSH, G. 1998. Pricing Residential Amenities: The Value of a View. *J. Real Estate Finance Econ. 16*, 1, 55–73.

BERGER, W., PIRINGER, H., FILZMOSER, P., AND GRÖLLER, M. E. 2011. Uncertainty-Aware Exploration of Continuous Parameter Spaces Using Multivariate Prediction. *Computer Graphics Forum 30*, 3 (June), 911–920.

CALDAS, L. G., AND NORFORD, L. K. 2002. A Design Optimization Tool Based on a Genetic Algorithm. *Automation in construction 11*, 2, 173–184.

CARR, H., SNOEYINK, J., AND AXEN, U. 2003. Computing Contour Trees in All Dimensions. *Comput. Geom. Theory Appl. 24*, 2, 75–94.

CARR, H., SNOEYINK, J., AND VAN DE PANNE, M. 2004. Simplifying Flexible Isosurfaces Using Local Geometric Measures. In *Proceedings of IEEE Visualization*, 497–504.

CHIANG, Y.-J., AND LU, X. 2003. Progressive Simplification of Tetrahedral Meshes Preserving All Isosurface Topologies. *Computer Graphics Forum 22*, 3, 493–504.

CHRONIS, A., TSIGKARI, M., GIOUVANOS, E., AISH, F., AND ZAKI, A. A. 2012. Performance Driven Design and Simulation Interfaces: A Multi-objective Parametric Optimization Process. In *Proceedings of the 2012 Symposium on Simulation for Architecture and Urban Design*, Society for Computer Simulation International, 14.

DEMIR, I., ALIAGA, D., AND BENES, B. 2014. Proceduralization of Buildings at City Scale. In *Proceedings of 2nd International Conference on 3D Vision (3DV)*, vol. 1, 456–463.

DEY, T., LI, K., LUO, C., RANJAN, P., SAFA, I., AND WANG, Y. 2010. Persistent Heat Signature for Pose-oblivious Matching of Incomplete Models. *Computer Graphics Forum 29*, 5, 1545–1554.

DEY, T. K., FAN, F., AND WANG, Y. 2013. An Efficient Computation of Handle and Tunnel Loops via Reeb Graphs. *ACM Trans. Graph. 32*, 4 (July), 32:1–32:10.

EDELSBRUNNER, H., LETSCHER, D., AND ZOMORODIAN., A. 2002. Topological Persistence and Simplification. *Disc. Comput. Geom. 28*, 4, 511–533.

GANE, V., AND HAYMAKER, J. 2007. Conceptual Design of Highrises with Parametric Methods. In *Proceedings of 25th eCAADe Conference Proceedings on Predicting the Future*, 293–302.

GERBER, S., BREMER, P., PASCUCCI, V., AND WHITAKER, R. 2010. Visual Exploration of High Dimensional Scalar Functions. *IEEE Trans. Comp. Graph. 16*, 6, 1271–1280.

HARVEY, W., AND WANG, Y. 2010. Topological Landscape Ensembles for Visualization of Scalar-Valued Functions. *Computer Graphics Forum 29*, 993–1002.

HILAGA, M., SHINAGAWA, Y., KOHMURA, T., AND KUNII, T. L. 2001. Topology Matching for Fully Automatic Similarity Estimation of 3D Shapes. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, ACM, New York, NY, USA, SIGGRAPH '01, 203–212.

KEOUGH, I., AND BENJAMIN, D. 2010. Multi-objective Optimization in Architectural Design. In *Proceedings of the 2010 Spring Simulation Multiconference*, Society for Computer Simulation International, SpringSim '10, 191:1–191:8.

KIMPIAN, J., MASON, J., COENDERS, J., JESTICO, D., AND WATTS, S. 2009. Sustainably Tall: Investment, Energy, Life Cycle. ACADIA, 130–143.

KLEIMAN, Y., FISH, N., LANIR, J., AND COHEN-OR, D. 2013. Dynamic Maps for Exploring and Browsing Shapes. *Computer Graphics Forum 32*, 5, 187–196.

LEVOY, M. 2006. Light Fields and Computational Imaging. *Computer 39*, 8 (Aug.), 46–55.

MATSUMOTO, Y. 2002. *An Introduction to Morse Theory*, vol. 208. Amer. Math. Soc.

MCKAY, M. D., BECKMAN, R. J., AND CONOVER, W. J. 2000. A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code. *Technometrics 42*, 1 (Feb.), 55–61.

MÜLLER, P., WONKA, P., HAEGLER, S., ULMER, A., AND VAN GOOL, L. 2006. Procedural Modeling of Buildings. *ACM Trans. Graph. 25*, 3 (July), 614–623.

OESTERLING, P., HEINE, C., JÄNICKE, H., SCHEUERMANN, G., AND HEYER, G. 2011. Visualization of High Dimensional Point Clouds Using their Density Distribution's Topology. *IEEE Trans. Comp. Graph. 17*, 11, 1547–1559.

PASCUCCI, V., SCORZELLI, G., BREMER, P.-T., AND MASCARENHAS, A. 2007. Robust On-line Computation of Reeb Graphs: Simplicity and Speed. *ACM Trans. Graph. 26*, 3, 58.

PASCUCCI, V., TRICOCHE, X., HAGEN, H., AND TIERNY, J., Eds. 2010. *Topological Methods in Data Analysis and Visualization*. Springer.

RIPPERDA, N., AND BRENNER, C. 2006. Reconstruction of Façade Structures Using a Formal Grammar and RjMCMC. In *Pattern Recognition*, vol. 4174 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 750–759.

SCHWARZ, M., AND MÜLLER, P. 2015. Advanced Procedural Modeling of Architecture. *ACM Trans. Graph. 34*, 4 (July), 107:1–107:12.

SKETCHUP, 2015. 3D Warehouse. https://3dwarehouse.sketchup.com/.

TALTON, J. O., GIBSON, D., YANG, L., HANRAHAN, P., AND KOLTUN, V. 2009. Exploratory Modeling with Collaborative Design Spaces. *ACM Trans. Graph. 28*, 5 (Dec.), 167:1–167:10.

TALTON, J. O., LOU, Y., LESSER, S., DUKE, J., MĚCH, R., AND KOLTUN, V. 2011. Metropolis Procedural Modeling. *ACM Trans. Graph. 30*, 2 (Apr.), 11:1–11:14.

TSIGKARI, M., CHRONIS, A., JOYCE, S. C., DAVIS, A., FENG, S., AND AISH, F. 2013. Integrated Design in the Simulation Process. In *Proceedings of the Symposium on Simulation for Architecture & Urban Design (SimAUD '13)*, Society for Computer Simulation International, 28:1–28:8.

VANEGAS, C. A., GARCIA-DORADO, I., ALIAGA, D. G., BENES, B., AND WADDELL, P. 2012. Inverse Design of Urban Procedural Models. *ACM Trans. Graph. 31*, 6 (Nov.), 168.

WEBER, G. H., DILLARD, S. E., CARR, H., PASCUCCI, V., AND HAMANN, B. 2007. Topology-controlled Volume Rendering. *IEEE Trans. Comp. Graph. 13*, 2, 330–341.

WEBER, G., BREMER, P.-T., AND PASCUCCI, V. 2007. Topological Landscapes: A Terrain Metaphor for Scientific Data. *IEEE Trans. Comp. Graph. 13*, 6 (Nov.), 1416–1423.

WIKI, O., 2015. Using OpenStreetMap — OpenStreetMap Wiki. [Online; accessed 31-March-2015].

WONKA, P., WIMMER, M., SILLION, F., AND RIBARSKY, W. 2003. Instant Architecture. *ACM Trans. Graph. 22*, 3 (July), 669–677.

WONKA, P., ALIAGA, D., MÜLLER, P., AND VANEGAS, C. 2011. Modeling 3D Urban Spaces Using Procedural and Simulation-based Techniques. In *ACM SIGGRAPH 2011 Courses*, 9:1–9:261.

WOOD, Z., HOPPE, H., DESBRUN, M., AND SCHRÖDER, P. 2004. Removing Excess Topology from Isosurfaces. *ACM Trans. Graph. 23*, 2, 190–208.

ZHANG, E., MISCHAIKOW, K., AND TURK, G. 2005. Feature-based Surface Parameterization and Texture Mapping. *ACM Trans. Graph. 24*, 1, 1–27.

ZHENG, Y., CAPRA, L., WOLFSON, O., AND YANG, H. 2014. Urban Computing: Concepts, Methodologies, and Applications. *ACM Trans. Intell. Syst. Technol. 5*, 3 (Sept.), 38:1–38:55.

ZHOU, J., AND TAKATSUKA, M. 2009. Automatic Transfer Function Generation Using Contour Tree Controlled Residue Flow Model and Color Harmonics. *IEEE Trans. Comp. Graph. 15*, 6, 1481–1488.