Procedural Design of Exterior Lighting for Buildings with Complex Constraints

MICHAEL SCHWARZ
Cornell University and Arizona State University
and
PETER WONKA
Arizona State University and KAUST

We present a system for the lighting design of procedurally modeled buildings. The design is procedurally specified as part of the ordinary modeling workflow by defining goals for the illumination that should be attained and locations where luminaires may be installed to realize these goals. Additionally, constraints can be modeled that make the arrangement of the installed luminaires respect certain aesthetic and structural considerations. From this specification, the system automatically generates a lighting solution for any concrete model instance. The underlying, intricate joint optimization and constraint satisfaction problem is approached with a stochastic scheme that operates directly in the complex subspace where all constraints are observed. To navigate this subspace efficaciously, the actual lighting situation is taken into account. We demonstrate our system on multiple examples spanning a variety of architectural structures and lighting designs.

Categories and Subject Descriptors: I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling; I.3.6 [**Computer Graphics**]: Methodology and Techniques; I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism; I.3.8 [**Computer Graphics**]: Applications; G.1.6 [**Mathematics of Computing**]: Optimization—*Constrained optimization*

General Terms: Algorithms

Additional Key Words and Phrases: Procedural modeling, exterior lighting design

ACM Reference Format:

Michael Schwarz and Peter Wonka. 2014. Procedural design of exterior lighting for buildings with complex constraints. ACM Trans. Graph. 33, 5, Article 166 (August 2014), 16 pages.
DOI: http://dx.doi.org/10.1145/2629573

This research was partially funded by the National Science Foundation. M. Schwarz was supported in part by a DAAD postdoctoral fellowship.
Authors' addresses: M. Schwarz (corresponding author), Esri R&D Center Zurich, Technoparkstrasse 1, 8005 Zurich, Switzerland; email: michschw@acm.org; P. Wonka, KAUST (King Abdullah University of Science and Technology), Thuwal, Saudi Arabia.
 0730-0301/2014/08-ART166 $15.00
DOI: http://dx.doi.org/10.1145/2629573

1. INTRODUCTION

The continuously increasing amount, scale, and complexity of virtual worlds, accompanied with growing users' expectations on realism and level of detail, pose high challenges on content creation. To cope with them, procedural modeling methods have proven very useful, especially if larger amounts of similar, but varied, detailed objects have to be created. While such approaches have been developed for a wide range of objects, like plants, cities, buildings, and rooms, they focus primarily on geometry. The important aspect of lighting, however, has been largely ignored, although it substantially influences the final visual result.

Often, light sources are part of the scene, and placing them such that a reasonable lighting result is obtained can be difficult and tedious. This design task involves deciding what illumination should be achieved as well as finding the right number, type, and location of light sources and adjusting their parameters. Determining such an appropriate light source setup is generally far from straightforward, not the least because the cumulative effect of all sources as well as occlusion and geometric constraints need to be considered (see Figure 2). Hence, even well-trained specialists usually require many iterations to achieve the desired illumination for one concrete variation of an object. The challenges are exacerbated in a procedural context, as the modeler would have to reason about spatial relationships (including occlusion) and how multiple lights play together in a way that holds for all possible variations. Even ignoring the fundamental problem of how to express such deductions in a procedural description, directly adding lighting to an object as part of this object's procedural generation is consequently rarely feasible, necessitating a separate, manual modeling step instead. Further, this task needs to be executed for each generated variation of an object, intensifying the demand for a more automated and integrated process.

In this article, we address these challenges and present a system that extends the procedural modeling workflow with lighting design capabilities. The system utilizes a shape grammar for procedural specification and augments it with new modeling operations that allow the modeler to describe goals concerning the illumination that should be achieved. Furthermore, operations are introduced for specifying potential locations where luminaires (i.e., light sources and their supporting bodies) can be installed and what kind of luminaires may be employed there. Importantly, it is also possible to impose constraints on a group of these installation sites, like enforcing to use the same luminaire at all group sites. This enables the modeler to constrain the placement of luminaires such that the luminaire-augmented object conforms to a desired aesthetic convention. The complicated task of finding an actual luminaire configuration that satisfies the goals as well as possible is instead left to the system. Thus, the user can concentrate on the general design of the lighting

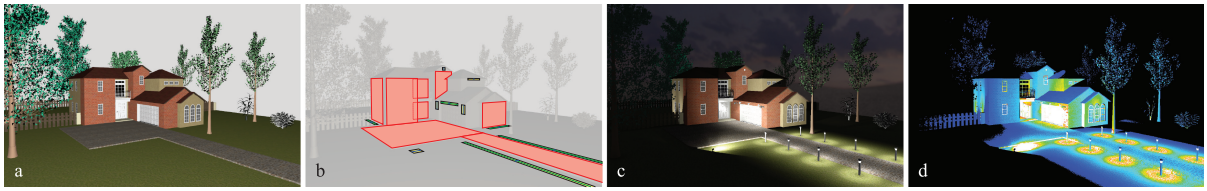

Fig. 1. Our system addresses lighting design for buildings in a procedural modeling context. A procedural building model (a) is augmented by a procedural lighting design specification that expresses the desired lighting in terms of lighting goals, luminaire installation sites, and constraints (b). For a concrete instance of this model, an according lighting solution ((c); (d): luminance visualization with a black / blue–green–red / white color map) is then automatically determined by our system.

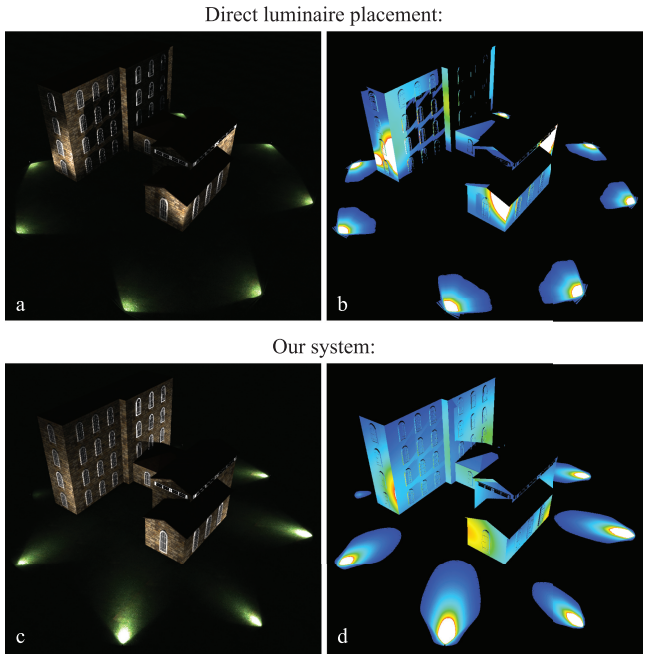

Fig. 2. Simple example where eight luminaires surrounding a building are supposed to light all façades. (a), (b) An effort to directly model a good solution struggles and would require numerous manual iterations to achieve a satisfying result; (c), (d) by contrast, our system takes a lighting design specification as input and successfully determines a solution that meets the desired goal. See Section 7.5 for details.

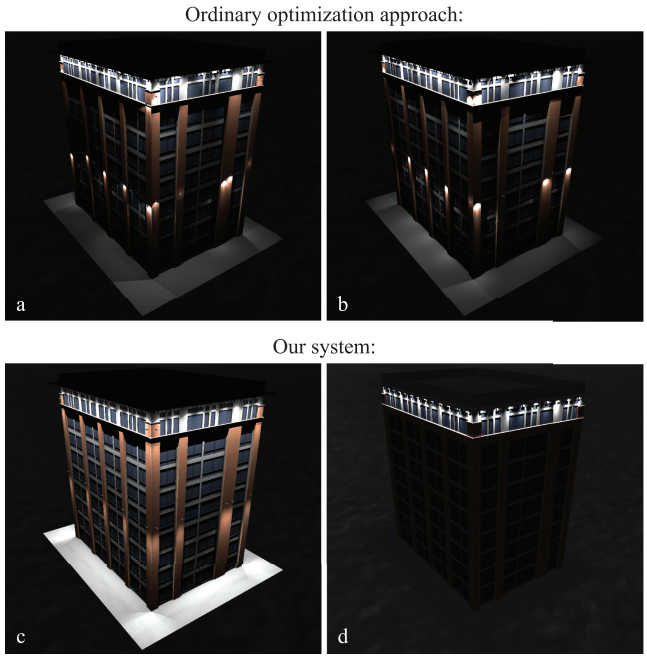

Fig. 3. Example design for an office building, where the luminaires at the brick columns are requested to be identical, use the same flux, and be mounted at consistent heights. Common optimization approaches can account for such constraints by treating them as soft. However, as demonstrated by the results after (a) 2,000; and (b) 200,000 iterations that both fail to meet the constraints and constitute unsatisfying lighting solutions, such approaches are not successful in coping with the complex space of valid luminaire configurations; (c) by contrast, our system yields a good solution after 2,000 iterations, which respects all constraints. See Section 7.6 for details. (d) Example of a different lighting design.

and need not bother with its concrete implementation. When generating a new object with a grammar, our system creates the object as usual, but also an according concrete specification of goals and luminaire installation sites. Subsequently, it derives an appropriate luminaire configuration and incorporates this result into the object. While this approach to procedural lighting design is applicable to many domains, our focus in this article is on architectural buildings and their exterior lighting (see Figure 1 for an example).

Determining a luminaire configuration that realizes a specific lighting design is generally a hard problem. In particular, such a configuration has to satisfy all modeled constraints. These introduce complex dependencies among the variables at multiple sites, rendering the set of constraint-observing configurations an incoherently and complexly shaped subspace of the domain of all configurations. Moreover, the illumination resulting from a configuration is an intricate function of its parameters that additionally is expensive to compute. As standard methods like simulated annealing are not well suited to deal with such a challenging setup (see Figure 3), we approach this constrained optimization problem with a stochastic scheme that operates directly in the complex subspace of constraint-satisfying configurations. It further accounts for the actual lighting situation to guide the exploration of this subspace.

To the best of our knowledge, this work is the first to address the (nonmanual) lighting of procedurally generated content. Unlike previous work on lighting design, our system supports complex constraints, which play a substantial role in enabling sophisticated designs. Our main contributions are thus twofold: First, we introduce an approach to specify the lighting design procedurally as part

of the ordinary procedural modeling process. Second, an integrated optimization and constraint satisfaction approach is presented that determines a luminaire configuration implementing a given, potentially complex, and in our case procedural specification.

2. RELATED WORK

Lighting design. Generally, lighting design is concerned with determining a configuration of lights whose resulting illumination satisfies some objective. In computer graphics, most related work focuses on lighting a single object or a spatially confined aggregation of objects with a few light sources. One objective targeted by several methods [Shacked and Lischinski 2001; Gumhold 2002; Lee et al. 2004] is to convey as much information as possible about an object, often taking perceptual considerations into account.

A major line of research, however, concentrates on (interactive) systems where a concrete lighting result is specified as goal and suitable light-source parameters are derived to achieve this goal. A quintessential part of most of these systems is an intuitive interface to input the desired lighting results. Some rely on direct specification and manipulation of lighting-induced features like shadows or highlights [Poulin and Fournier 1992; Pellacini et al. 2002], while others adopt a painting paradigm where the intended illumination result is painted by the user [Schoeneman et al. 1993; Anrys and Dutré 2004; Okabe et al. 2007; Shesh and Chen 2007]. Advanced examples are the system for cinematic lighting by Pellacini et al. [2007] and the recent work by Lin et al. [2013]. In our system, the desired result is specified procedurally in terms of photometric quantities.

A few approaches are concerned primarily with interior architectural lighting. For instance, Zmugg et al. [2010] take a user-painted illumination goal as input and derive the positions of (identical) light sources, assuming surfaces of identical white diffuse material. A more complex, general system was devised by Costa et al. [1999] that, given a lighting goal specified via fictitious lights, a cost function expressed via a custom scripting language and a number of desired lights, optimizes the parameters of these lights. The radiosity-based radioptimization method [Kawai et al. 1993] allows specifying goals, like radiosity value ranges for scene elements, and accordingly optimizes user-selected variables, like parameters of light sources (of fixed position) or surface reflectivities.

These systems all make substantial simplifications. They consider only abstract lights (points or fixed surface patches) that are described by a few independent parameters; this often renders the addressed problem amenable to efficient optimization techniques, though. By contrast, our system deals with models of real luminaires, comprising both a (parametric) luminaire body and a light source, whose position and orientation are influenced by the body's configuration. Moreover, existing systems basically treat the individual lights in isolation, whereas we support relating multiple luminaires in complex ways. Although Costa et al. [1999] in principle allow for specifying complex constraints, their optimization method treats them as a black box, simply rejecting all solution proposals that violate a constraint. This becomes impractical when faced with more than a few simple constraints. By contrast, our approach explicitly accounts for all constraints and hence readily copes with complex, highly constrained lighting specifications.

Procedural modeling. In procedural modeling, content is generated according to some procedure and this is often expressed using a formal grammar. A well-known example is L-systems, which are frequently employed to describe growth processes, like for modeling plants [Prusinkiewicz and Lindenmayer 1990] or deriving street networks [Parish and Müller 2001]. Shape grammars

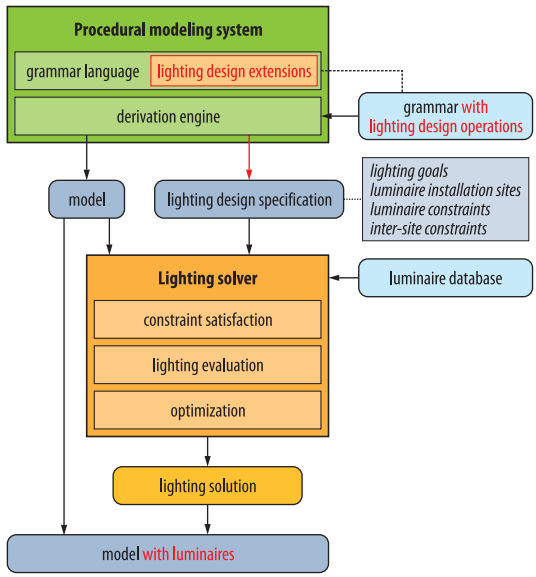

Fig. 4. Overview of our procedural lighting design system.

[Stiny and Gips 1972; Stiny 2006] are the basis for several architectural modeling approaches; they have been adapted for façades [Wonka et al. 2003] and entire buildings. The most prominent example is CGA shape [Müller et al. 2006], for which also extensions [Krecklau and Kobbelt 2011; Whiting et al. 2009] and generalizations, like enhanced language expressiveness [Krecklau et al. 2010], were suggested. Motivated by its wide use, we chose the recent variant of CGA shape found within the commercial software CityEngine [Esri 2012] as basis for our system's grammar language.

To facilitate the often tedious modeling process, user interfaces were devised that allow interactive visual grammar editing [Lipp et al. 2008], graph-based rule editing and modeling [Patow 2012], or direct profile modeling [Kelly and Wonka 2011]. Another option is to incorporate domain knowledge, enabling a direct higher-level specification of the result. Recent examples include floorplan generation from given requirements [Merrell et al. 2010] and optimizing furniture layout based on arrangement criteria [Merrell et al. 2011; Yu et al. 2011]. Similarly, Talton et al. [2011] search for specific instances of a probabilistic L-system that satisfy certain prescribed objectives, and Vanegas et al. [2012] explore parameter changes to a given simple procedural model that make it meet specified target values for predefined indicators. The encountered domain to optimize over has a sufficiently simple structure for these systems to be able to resort to standard stochastic methods, executing arbitrary random changes to a current solution in an effort to improve on it. By contrast, we are dealing with lighting specifications that comprise complex constraints, imposing intricate dependencies onto the subspace of valid solutions. Consequently, sampling from and moving within this subspace involve solving a challenging constraint satisfaction problem. With computing lighting being expensive, the optimization's efficiency also becomes significantly more important. Among others, we address this issue by choosing the changes considered for finding a better solution wisely and not just arbitrarily.

3. OVERVIEW

Figure 4 presents an overview of our system. The desired lighting design is input by specifying goals for the illumination that should be

attained and identifying locations where luminaires can be installed to realize these goals (Section 4). Additionally, constraints on the luminaires and on groups of locations can be imposed, which is often crucial to implement aesthetic and structural desiderata.

Based on this design specification, our system determines a lighting solution that achieves the targeted goals well while fully adhering to all constraints defined (Section 5). The underlying constrained optimization problem is tackled with an integrated stochastic approach that operates directly in the complex space of luminaire configurations satisfying all constraints. Because illumination is expensive to compute, our approach further pays special attention to efficacy.

This lighting design module is integrated into a procedural modeling system. Extending a standard shape grammar language such that a lighting design can be specified procedurally (Section 6), our system directly embeds lighting design into the procedural modeling workflow. For each generated model instance, an according lighting design specification instance is produced and a corresponding lighting solution is then determined and incorporated into the model.

4. LIGHTING DESIGN SPECIFICATION

In real-world lighting design, one common approach is to first identify what kind of illumination should be attained where. This process is typically guided by functional considerations, like accentuating a certain structure or providing enough light to allow for facial recognition or to identify obstacles when walking. Moreover, aesthetic factors may heavily influence the decision. In a subsequent step, appropriate luminaires are selected to realize these lighting goals and, finally, a concrete configuration is determined. In practice, several iterations of this general procedure may be necessary to establish a satisfactory lighting solution.

Similarly, a lighting design is specified in our system by defining lighting goals and identifying locations at which luminaires may be installed, describing them via installation sites. Additionally, luminaire and inter-site constraints may be established to restrict the set of candidate luminaires and enforce consistency across installation sites, respectively. The design specification is complemented by a database of luminaires. Our system can then automatically determine a configuration at the installation sites that realizes the specification, well achieving the goals.

4.1 Lighting Goals

A lighting goal is specified via a set of objectives concerning the illumination of one or more surfaces, referred to as *goal sites*. Such a site is not necessarily an actual scene surface, but may be a virtual one. For the goal and for each of the objectives, the relative importance is given by a weight.

The illumination is described in terms of either one of the photometric measures illuminance and luminance. *Illuminance* E, measured in lx ($= \mathrm{lm/m^2}$), quantifies the light that reaches a surface, whereas *luminance* L ($\mathrm{cd/m^2}$) additionally takes the scattering on the surface into account, thus corresponding to the reflected light that is perceived by the observer. When computing luminance for evaluating goals, we ignore the specular component of the surface's material and account only for its diffuse part (assuming Lambertian behavior), as otherwise the luminance depends on the viewer's position that is unknown to the system and may be arbitrary.

Supported objectives include the desired *minimum* and *maximum* values for the photometric measure adopted for a goal. A target value can also be prescribed for its *mean* over all surfaces, and for its *variance* an upper bound may be imposed. Moreover, a desired minimum uniformity of the illumination across all surfaces can be specified. This is commonly expressed as the ratio of minimum to mean illumination, referred to as average or *mean uniformity*. Alternatively, sometimes the ratio of minimum to maximum illumination is employed, termed general or *extreme uniformity*. In addition to these photometric ones, further objectives are of course conceivable, like high energy efficiency or color rendition quality, but we did not explore this in the current system.

4.2 Luminaires

A luminaire (also called light fixture) is a device that houses one lamp (or sometimes multiple), the actual source of light, and often comprises reflectors and shielding and diffusion components that influence the overall light pattern.

Light source. Each lamp is characterized by quantities like the emitted luminous flux Φ (measured in lm), its color temperature, and the color rendition performance. Its most distinctive feature, though, is typically the pattern of emitted light. At a sufficient distance, the lamp can be reasonably approximated as a point light, allowing the pattern to be compactly represented as a directional luminous intensity distribution. Such far-field photometric data is routinely provided by lighting manufacturers.

In our system, light sources are hence modeled as point lights with an associated directional distribution. This distribution typically represents a real-world luminaire, but it could easily encode a simpler computer-graphics omnidirectional or spotlight. Moreover, we assume a falloff of intensity that is quadratic in distance as in the real world, and gradually fade it out once it drops below a threshold to limit the spatial support of each light.

Luminaire body. The luminaire's body is represented by a geometric model in our system. This model may be composed of several elements forming a chain, where each element can be scaled in one direction or rotated around an axis. The amount of scaling or rotation, respectively, is a free parameter of the model with a specific range of values. This way, varying pole heights and tilt angles, for instance, are readily supported.

For each model, we additionally record the corresponding insertion point, the installation normal direction, and, if applicable, the direction of forward orientation (see Figure 6). When the luminaire gets placed in the scene, these are aligned with the installation point, the normal direction of the installation site, and, if specified by the modeler, the site's orientation direction, respectively. Furthermore, information about the location of the associated point light source and the frame for its luminous intensity distribution, that both may depend on the model's parameters, is kept.

Luminaire model. In our system, a luminaire hence consists of a luminous intensity distribution and a luminaire body model, as well as a range for the lamp's luminous flux. By default, this is simply the value from the manufacturer-provided photometric data, but in all our examples, we generally specify a larger range to allow for light dimming and thus more flexibility. To support an efficient selection of a set of luminaires, we assign a name and a set of permissible *installation location types* (wall, ceiling, and floor) to each luminaire and further classify it using one or more general *luminaire types*, which include floodlight, spotlight, bollard, street light, and in-ground light, for instance. Figure 5 shows examples of luminaires.

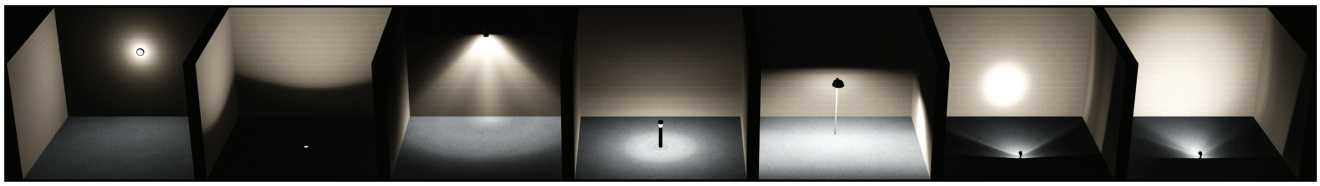

Fig. 5. Some examples of luminaires as used in our system. The two luminaires on the right use the same body and luminous flux but feature different luminous intensity distributions.

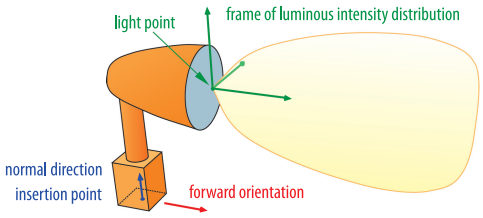

Fig. 6. To enable correctly installing a luminaire, several geometric quantities are maintained.

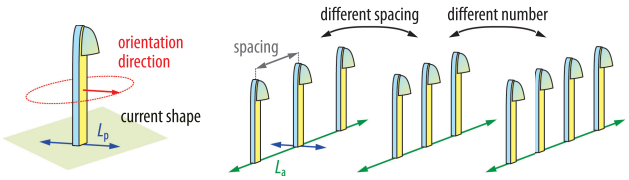

Fig. 7. A luminaire installation site's specification may define translational freedom (along L_p), prescribe an orientation direction, and allow for accommodating more than one luminaire.

4.3 Luminaire Installation Sites

Potential locations where luminaires can be installed are specified by means of installation sites. In its most restricted form, such a site prescribes an exact position on a surface, with the site's normal direction being derived from this surface. Alternatively, a line segment L_p may be specified on which the installation point has to lie (see Figure 7). This degree of freedom allows modeling, for example, that a luminaire is horizontally centered between two windows while the exact installation height is left flexible. Moreover, an orientation direction for the luminaire can be specified, for instance, to enforce that a wall-mounted luminaire points straight downwards.

An installation site is not restricted to accommodate only one (or none) luminaire but may also support multiple identical, linearly arranged, and uniformly spaced luminaires. The site's specification hence further includes a range for the number of luminaires to install, as well as a line segment L_a along which to arrange them. The luminaires are placed such that, as a group, they are centered in this segment, with their spacing being flexible within the constraints imposed by the segment's length and the minimum spacing implied by the concretely employed luminaire (and its orientation and configuration). This covers common cases like bollards placed along a pathway where the exact spacing of the luminaires and hence their number is not known a priori. The distance of the bollards to the pathway can also be left flexible by additionally specifying a (nondegenerate) line segment L_p for the position.

Luminaire constraints. The specific set of luminaires that may be used at an installation site is established by luminaire constraints. These allow putting restrictions on the installation location type, the luminaire type, and the name of permissible luminaires. Additionally, photometric constraints can be imposed, like prescribing a range of permissible luminous flux values or of peak luminous intensity values (that depend on the flux and the luminaire's intensity distribution). One may also define constraints on further attributes of luminaires, like manufacturer, product family, or energy efficiency, but we currently don't maintain these in our luminaire database.

Inter-site constraints. In practice, it is often desirable that, at a certain group of installation sites, the luminaires eventually placed share some characteristics. For instance, assume multiple installation sites of common purpose exist along a wall, that their shared luminaire constraints leave multiple options, and that they all adopt identical vertical line segments for positioning. It is then easily possible that a different luminaire is installed at each site, that the luminous flux at two different sites varies, and that luminaires are installed at inconsistent heights. Even though such a diverse configuration may indeed best achieve the specified lighting goals, one instead typically wants that the same luminaire and the same flux are used at all these installation sites and that the luminaires are all installed at the same vertical position, even if this means sacrificing completely achieving the lighting goals.

Such important design objectives can be enforced by inter-site constraints. For a group of sites, our system currently supports (among others) requiring:

—that the same luminaire is used,

—that the used luminaires share at least one luminaire type,

—that all luminaires use the same luminous flux value,

—that, if a certain luminaire is installed at multiple sites, all instances use the same luminous flux values,

—that, if a certain luminaire has a parametric body model and is installed at multiple sites, all instances use the same parameter values (implying that pole heights or tilt angles are consistent),

—that, if a line segment for the final position is specified, the positions at all sites are consistent,

—that, if more than one luminaire is installed at a site, the spacing is consistent across all sites, or

—that the same number of luminaires is installed.

5. DETERMINING LIGHTING SOLUTIONS

Implementing a lighting design based on its specification requires determining a suitable luminaire configuration at all installation sites. Such a configuration must adhere to all constraints defined in the specification to be valid (Section 5.1). Ensuring this requires solving an intricate constraint satisfaction problem [Dechter 2003],

where we are faced with a mixture of discrete and continuous variables that are partially coupled, causing the domain of some variables to depend on the values of others. In particular, inter-site constraints often entail complicated dependencies.

At the same time, the configuration should yield a lighting that meets the specified lighting goals. Because generally no solution exists that fulfills all goals completely, we seek a configuration that attains the goals as well as possible while respecting all constraints. Assessing a certain configuration Λ (Section 5.5) involves computing the illumination induced by Λ (Section 5.6) and is hence not only a complex and highly nonlinear function of Λ but also expensive to evaluate.

We tackle this joint optimization and constraint satisfaction problem with an integrated stochastic approach (Section 5.4) that operates directly in the highly constrained, complexly shaped space of valid configurations. It also harnesses knowledge about the induced lighting situation to generate more efficacious proposals for improving an intermediate solution (Section 5.7).

In the following exposition, we first focus on the aspect of constraint satisfaction. After characterizing luminaire configurations, discussing the involved variables as well as the dependencies among them established by the specified constraints, and how these restrict the variables' domains (Section 5.1), we present an approach to generate a valid (i.e., constraint-satisfying) configuration (Section 5.2). Subsequently, we show how a variable of an existing valid configuration can be changed such that the resulting configuration is still valid (Section 5.3), before turning to and elaborating on our stochastic optimization approach that these techniques are directly used by and integrated into (Section 5.4). The cost function employed for assessing configurations is detailed (Section 5.5), and the involved evaluation of the lighting induced by a configuration is described (Section 5.6). Finally, we discuss generating efficacious proposals for changing an existing configuration (Section 5.7).

5.1 Valid Luminaire Configurations

A luminaire configuration $\Lambda = (\lambda_1, \lambda_2, \ldots, \lambda_N)$ consists of the configurations λ_k at all N installation sites; each is described by a tuple of variables

$$\lambda_k = (M_k, \Phi_k, \Theta_k, R_k, n_k, \mathbf{x}_k, \Delta\mathbf{x}_k),$$

where $M_k \in \mathcal{M}$ identifies the installed luminaire model, with $\mathcal{M}$ denoting the set of all known luminaire models, $\Phi_k \in \mathbf{R}$ is the luminous flux, $\Theta_k \in \cup_{d\in\mathbf{N}_0}\mathbf{R}^d$ are the concrete parameter values for the luminaire's body model, $R_k \in \mathrm{SO}(3)$ determines the orientation of the installed luminaire instances, $n_k \in \mathbf{N}_0$ is the number of luminaires installed at the site, $\mathbf{x}_k \in \mathbf{R}^3$ specifies the position of the first luminaire instance's installation point, and $\Delta\mathbf{x}_k \in \mathbf{R}^3$ is the spacing between successive luminaire instances (the i-th luminaire instance is installed at $\mathbf{x}_k + i\,\Delta\mathbf{x}_k$).

The resulting configuration domain

$$\mathcal{D} = \bigtimes_{k=1}^{N} \mathcal{D}_k = \left(\mathcal{M}, \mathbf{R}, \cup_{d\in\mathbf{N}_0}\mathbf{R}^d, \mathrm{SO}(3), \mathbf{N}_0, \mathbf{R}^3, \mathbf{R}^3\right)^N$$

is a high-dimensional space comprising both discrete and continuous dimensions. The lighting specification imposes several constraints on $\mathcal{D}$, defining the subspace $\mathcal{V} \subset \mathcal{D}$ of valid configurations. $\mathcal{V}$ is no longer a simple product space but features a complex structure with intricate relationships between various dimensions (see Figure 8).

Concretely, the specification for an installation site k restricts the number of luminaires n_k to a range $n_k^{\min}, \ldots, n_k^{\max}$. The spacing

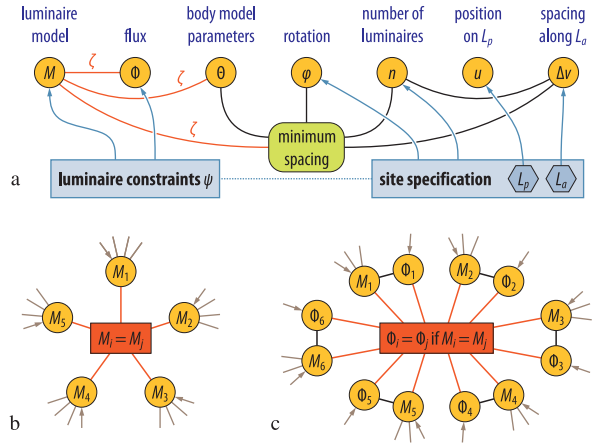

Fig. 8. Example fragments of the complex constraint network resulting from a lighting design specification. (a) The variables describing the configuration at an installation site (shown in orange) are constrained by the site's specification and the referenced luminaire constraints ψ. Moreover, the domain of a variable may depend on the concrete values of other variables. For instance, the luminaire model M restricts the permissible flux range for Φ according to the model specification $\zeta(M)$. Reciprocally, the flux value Φ restricts the set of permissible luminaire models for M. Most site variables are also mutually dependent due to the implied minimum spacing (a derived quantity shown in green). (b) Dependencies among variables from different sites are established by inter-site constraints. For instance, a use-same-luminaire constraint for sites $k \in \mathcal{S} = \{1, \ldots, 5\}$ makes all M_k interdependent. As these dependencies hold in addition to the existing ones (indicated by the incident taupe-colored arcs), all other variables dependent on any M_k become transitively interdependent. (c) Complex inter-site constraints may involve multiple variables at all associated sites (here: $\mathcal{S} = \{1, \ldots, 6\}$).

$\Delta\mathbf{x}_k$ is confined to a distance $\Delta v_k \in [0, \|L_{\mathrm{a},k}\|]$ along the site's line segment $L_{\mathrm{a},k}$, and the 3D position $\mathbf{x}_k$ becomes a function of the 1D position $u_k \in [0, \|L_{\mathrm{p},k}\|]$ on the site's line segment $L_{\mathrm{p},k}$, Δv_k, and n_k. The site's normal fixes one axis of the luminaire's local space and hence two degrees of freedom of the orientation R_k. If an orientation direction is prescribed, R_k's third degree of freedom, the rotation angle φ_k around the normal, also becomes fixed. Overall, this allows us to describe each configuration λ_k slightly more concisely as

$$\tilde{\lambda}_k = (M_k, \Phi_k, \Theta_k, \varphi_k, n_k, u_k, \Delta v_k).$$

The luminaire constraints $\psi = \psi(k)$ specified for the installation site define a set of luminaires $\mathcal{M}_\psi \subseteq \mathcal{M}$, requiring $M_k \in \mathcal{M}_\psi$. The constraints may also restrict the flux Φ_k (either directly or by prescribing a peak intensity) for each permissible luminaire M_k to a range $\mathbf{\Phi}_\psi(M_k)$.

Additionally, the model specification $\zeta = \zeta(M_k)$ for a luminaire M_k defines the flux range supported by the luminaire, the number d_k of body model parameters $\theta_{k,i}$, and their permissible values, thus constraining Φ_k and $\Theta_k = (\theta_{k,1}, \ldots, \theta_{k,d_k})$ for a certain M_k. The specification ζ also yields a minimum spacing distance that may depend on the installed luminaires' orientation φ_k and body parameter values Θ_k, providing a lower bound for Δv_k. It further imposes an upper bound on the number n_k of luminaires that may be installed at the site. This is one example where multiple variables of a site's configuration become coupled in a complex way.

By contrast, inter-site constraints ξ entail dependencies between multiple installation sites. Some constraints enforce that a certain

(set-valued) function $f(\tau_k)$ of a configuration variable τ_k yields (at least) one common result at all sites $k \in \mathcal{S}(\xi)$, where $\mathcal{S}(\xi)$ denotes the set of sites associated with the constraint

$$\bigcap\nolimits_{k\in\mathcal{S}(\xi)} f(\tau_k) \neq \emptyset$$

This couples related variables across multiple installation sites and, if the function is injective, the dimensionality of the configuration space is effectively reduced by $|\mathcal{S}(\xi)| - 1$ dimensions. One such example is the use-same-luminaire constraint, which uses $\tau_k \equiv M_k$ and $f(M) = \{M\}$. By contrast, a use-same-luminaire-type constraint employs the noninjective function $f(M) = type(M)$, where $type(M)$ yields the set of luminaire types to which a luminaire M belongs.

Other inter-site constraints result in even more complicated dependencies. For instance, the lighting specification may dictate that, if a certain luminaire is installed at multiple sites, all instances use the same flux values (see Figure 8(c)), thus requiring the configuration to satisfy

$$\bigwedge\nolimits_{M\in\{M_k|k\in\mathcal{S}(\xi)\}} \left|\left\{\Phi_\ell \mid \ell \in \mathcal{S}(\xi) \wedge M_\ell = M\right\}\right| = 1.$$

To capture the interplay of all constraints that result from the lighting specification, we define an indicator function $\chi(\Lambda)$ that is zero if and only if a configuration Λ satisfies all these constraints. A configuration Λ with $\chi(\Lambda) = 0$ is called *valid* and the space of valid configurations is thus given by

$$\mathcal{V} = \left\{\Lambda \mid \Lambda \in \mathcal{D} \wedge \chi(\Lambda) = 0\right\} \subset \mathcal{D}.$$

5.2 Generating a Valid Configuration

In order to create a random valid configuration $\Lambda \in \mathcal{V}$, we pursue a constructive approach that can yield any possible configuration and is guaranteed to find one if any exists. Conceptually, we start with the unconstrained domain $\mathcal{D}$ and successively apply all constraints to narrow it down to the space $\mathcal{V}$, subsequently selecting one random configuration out of $\mathcal{V}$. However, because the domains of many configuration variables are dependent on the values of other variables, we cannot simply represent each variable's domain in isolation and hence representing the whole space $\mathcal{V}$ explicitly is generally not practical with respect to both storage and computational effort.

On the other hand, a greedy approach that visits all variables one after another, each time first determining a local domain by applying all constraints implied by the values chosen for the already visited variables and then selecting a random value from that domain, is not very efficient. Even if the order in which the variables are visited is chosen carefully, the purely local view makes it unavoidable that sometimes values are selected that cause the domains of some variables visited later to become empty. This only becomes known, though, when the first variable with an empty domain is visited and necessitates backtracking and selecting a different value for a variable visited earlier.

For example (see Figure 9), when a same-flux constraint ξ applies to multiple sites $k \in \mathcal{S}(\xi)$, the flux at all these sites is restricted to

$$\mathbf{\Phi}_\xi = \bigcap\nolimits_{k\in\mathcal{S}(\xi)} \bigcup\nolimits_{M\in\mathcal{M}_{\psi(k)}} \mathbf{\Phi}_{\psi(k)}(M), \tag{1}$$

and the permissible luminaires at each site k are limited to those with a compatible flux range (i.e., $M_k \in \{M \mid M \in \mathcal{M}_{\psi(k)} \wedge \mathbf{\Phi}_{\psi(k)}(M) \cap \mathbf{\Phi}_\xi \neq \emptyset\}$). However, when greedily processing one variable a time, these limitations only become fully known after multiple variables have been visited, at which point a wrong value may already have been selected. If one first chooses a flux value, it may, for example, be compatible with all but the last site. Consequently, only when the luminaire of this last site is to be chosen and the derived domain is empty does it become apparent that the value of some variable visited earlier causes problems. Visiting the variables in a different order, that is, first choosing the luminaire at one or multiple sites and then determining the flux, results in a similar situation.

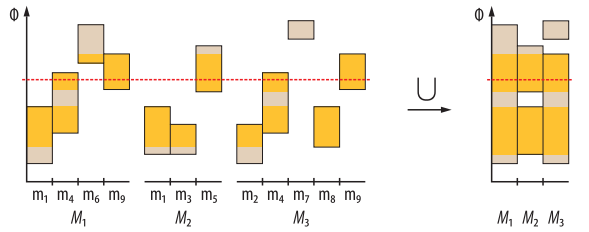

Fig. 9. Example of the joint (M_k, Φ_k) domain for a group of three sites related by a same-flux constraint ξ. The flux range $\mathbf{\Phi}_{\psi(k)}(M)$ for each permissible luminaire M_k (left) establishes the overall flux range supported at each site (right). The same-flux constraint restricts the flux values to the orange subset $\mathbf{\Phi}_\xi$, and each concrete flux value further constrains the choice of luminaires. For instance, for a flux corresponding to the red line, $M_1 \in \{m_4, m_9\}$, $M_2 \in \{m_5\}$, and $M_3 \in \{m_4, m_9\}$ holds. Note that the permissible flux range for a certain luminaire M may differ across sites due to different luminaire constraints $\psi(k)$; for example, $\mathbf{\Phi}_{\psi(1)}(m_1) \supset \mathbf{\Phi}_{\psi(2)}(m_1)$.

Approach. Avoiding the major shortcomings of the two extremes of full representation and greedy, purely local exploration, our approach offers an efficient middle ground. We first partition the configuration space's variables into independent groups such that exactly all those site configurations λ_k are binned together that are (transitively) connected via at least one constraint. For each partition, we incrementally and lazily build an explicit representation of the relevant subset Γ of $\mathcal{V}$'s dimensions as implied by the constraints that apply. This enables efficiently determining random values for the partition's configurations λ_k respecting all constraints.

More precisely, we impose an order $\gamma_1, \ldots, \gamma_{|\Gamma|}$ on the variables contained in Γ and employ a sequence of according sets Ω_j to represent (incrementally tightened) supersets of these variables' domains. In a first phase, these sets are successively determined (Figure 10 illustrates). Ω_i is initialized to all values permissible for γ_i under all constraints that rely only on the variables $\gamma_1, \ldots, \gamma_i$. This involves that, for each preceding variable $\gamma_{i'}$, $i' < i$, that is related to γ_i via a constraint, the set $\Omega_{i'}$ is forward enforced onto Ω_i, excluding values from Ω_i for which no constraint-satisfying value in $\Omega_{i'}$ exists. If conversely this enforcement reveals that any such related preceding set (potentially) features a value that has no matching γ_i value in Ω_i, then Ω_i is backward enforced onto these related preceding sets $\Omega_{i'}$, narrowing them further down by removing those values from $\Omega_{i'}$ with which no value in Ω_i satisfies the constraint among the corresponding variables. In case any value gets removed from $\Omega_{i'}$, we propagate this refinement by backward enforcing $\Omega_{i'}$ (which may trigger further enforcements) and subsequently forward enforcing it. While this constraint inference takes permissible values for other variables into account, it does not consider concrete variable assignments. Consequently, each resulting set Ω_i may not always constitute the exact domain of γ_i but a superset of it.

In a second phase, concrete values are successively chosen for all variables. Beginning with $i = 1$, a random value for γ_i is selected from Ω_i. This assignment is forward enforced onto all sets $\Omega_{i'}$, $i' > i$, corresponding to directly dependent variables. Whenever

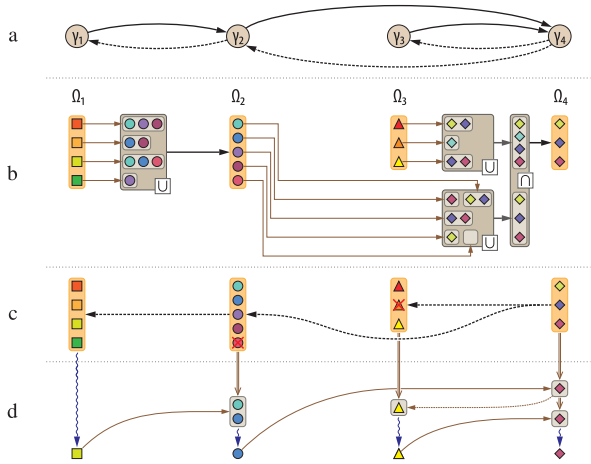

Fig. 10. Finding a valid configuration, demonstrated on a simple example involving four variables γ_i. (This corresponds, for instance, to a situation with two sites where $\gamma_1 \equiv M_1$, $\gamma_2 \equiv \Phi_1$, $\gamma_3 \equiv M_2$, $\gamma_4 \equiv \Phi_2$, and a same-flux constraint holds.) (a) Constraint network, where each mutual dependency is shown as a forward- (solid line) and a backward- (dashed line) directed arc; (b) in a first phase, the sets Ω_i, representing permissible values for γ_i, are determined. Ω_1 is initialized to the unconstrained domain of γ_1. Because of the dependency $\gamma_1 \to \gamma_2$, it is then forward enforced onto Ω_2: for each $\omega \in \Omega_1$, the corresponding set of values for γ_2 respecting the constraint among γ_1 and γ_2 is determined, and the union of these sets constitutes Ω_2. As γ_3 does not (directly) depend on any γ_i, $i < 3$, Ω_3 is set to the unconstrained domain of γ_3. Subsequently, Ω_4 is determined by forward enforcing Ω_2 and Ω_3: each of them yields a set of values for γ_4 satisfying the respective constraint, and the intersection of these two sets identifies those values meeting both constraints in which γ_4 is engaged; (c) because the intersection discarded an element (◇) and also because at least one element of the forward-enforced sets yielded an empty set (●), Ω_4 is backward enforced onto Ω_2 and Ω_3. Since for ▲ $\in \Omega_3$ no $\omega \in \Omega_4$ exists that satisfies the constraint among γ_3 and γ_4, it is removed from Ω_3. Analogously, ● is removed from Ω_2, which is then backward enforced onto Ω_1 (where it does not inflict any change); (d) utilizing the sets Ω_i, concrete values for the variables γ_i are determined in a second phase. First, a random value (■) from Ω_1 is chosen for γ_1. Due to the dependency $\gamma_1 \to \gamma_2$, this assignment is forward enforced onto Ω_2, yielding a refined set $\Omega_2|_{\gamma_1}$, from which a value (●) for γ_2 is subsequently selected (entailing forward enforcement onto Ω_4, which is then backward enforced onto Ω_3). Note that if $\Omega_2|_{\gamma_1}$ was empty, backtracking would be performed, causing a different value for γ_1 to be chosen. Values for the remaining variables γ_3 and γ_4 are determined analogously.

such inference causes a set $\Omega_{i'}$ to become smaller, $\Omega_{i'}$ is forward enforced itself, followed by backward enforcement to all sets $\Omega_{j'}$, $i < j' < i'$, where $\gamma_{j'}$ participates in a constraint with $\gamma_{i'}$. This process is repeated for all other variables γ_i, $i = 2, \ldots, |\Gamma|$, eventually yielding a valid configuration $\Lambda|_\Gamma$ for the sites covered by the partition, if one exists.

As each set Ω_i may initially be a superset of γ_i's domain, it is possible that a value is selected for γ_i that later turns out to be actually outside the domain. Such cases result in at least one set $\Omega_{i'}$, $i' > i$, becoming empty during constraint enforcement and are hence detected early on. We then perform backtracking, which involves removing the offending value from Ω_i, resetting all transitively related $\Omega_{i'}$, $i' > i$, and continuing with another value from Ω_i. To facilitate resets, we maintain a set $\bar{\Omega}_j$ of discarded values for each Ω_j, $j > 1$.

Efficiency-related details. While the ordering chosen for the variables contained in Γ has no impact on the scheme's correct performance, it can affect the number of constraint enforcements and how often backtracking is required. Aiming for keeping such overhead low, we assign each constraint a preferred enforcement direction based on functional considerations and heuristics (like making a discrete variable enforce its assignment onto a continuous one's domain) and utilize the resulting directed graph's topological sort to derive the ordering. (This approach was not used for the toy example in Figure 10.)

For selected pairs of variables γ_a and γ_b, $a < b$ (like luminaire model M and flux Φ), we also refrain from using two sets $\Omega_{a'}$ and $\Omega_{b'}$ in favor of a single 2D set Ω_j, where for each contained value of γ_a the corresponding set of permissible γ_b values is stored. This enables a more fine-grained constraint inference and better pruning, and thus helps in limiting the number of cases where backprojection eventually becomes necessary. Similarly, if multiple variables γ_i are constrained to have identical values, we employ a single shared set $\Omega_{j'}$ for them.

Examples. Concretely, if a partition comprises only one site k (i.e., the site is not affected by any inter-site constraints), we first determine the set of luminaires $\mathcal{M}_{\psi(k)}$ allowed by the according luminaire constraints $\psi(k)$ and, if $n_k^{\min} > 1$, subsequently prune it by applying the minimum spacing constraint implied by each luminaire's specifications $\zeta(M)$. After selecting a luminaire M_k, we successively choose values for the remaining variables of $\tilde{\lambda}_k$. Each time a variable has been assigned a value, the domains of the outstanding ones are accordingly refined. By processing the variables in a carefully chosen order $(M_k, \Phi_k, \Theta_k, \varphi_k, n_k, u_k, \Delta v_k)$, backtracking never becomes necessary.

If sites are related via inter-site constraints, some sets Ω_j usually cover variables common to multiple sites explicitly. For example, if a same-flux constraint ξ applies, we maintain a single set $\mathbf{\Phi}_\xi$ (Eq. (1)) for all Φ_k, thus avoiding selecting a flux at one site that finally is incompatible with another site. As a consequence, backtracking may generally only become necessary in case a complex interplay of multiple constraints or an intricate constraint apply. An example for the latter case is a use-same-flux-if-same-luminaire constraint ξ, where for each site $k \in \mathcal{S}(\xi)$ we maintain the domain of (M_k, Φ_k) as a 2D set. This is rather inexpensive to represent (for each M, a set of intervals for Φ is stored) and helps both detecting and avoiding incompatible choices early on. We first select the luminaires M_k for all sites, which may involve backtracking, and subsequently directly choose compatible flux values Φ_k.

Alternative approaches. Unlike our approach, the common procedure of starting with an arbitrary configuration $\Lambda^0 \in \mathcal{D}$ and then projecting it into $\mathcal{V}$ is not viable due to $\mathcal{V}$'s complex shape; please refer to the supplemental material for a more detailed discussion.

5.3 Moving in the Subspace of Valid Configurations

Given a valid configuration $\Lambda \in \mathcal{V}$, a new configuration Λ' can be derived by modifying the value of one variable τ, like increasing the flux Φ_k at some site k. However, the dependencies among the variables imposed by the various constraints may necessitate further changes to other variables for Λ' to not leave the space $\mathcal{V}$ and remain valid.

Therefore, we pursue a constructive approach similar to the one described in Section 5.2, which also ensures that, if a modification of τ is possible, an according value is found. First all variables that are ultimately affected by τ's value via constraints are determined,

then an explicit representation of the according subspace of $\mathcal{V}$ is incrementally and lazily constructed. In the set Ω_τ representing τ's domain, we optionally flag all those values that entail a change to other variables, using separate flags for whether any of these variables belong to the same site as τ or to another site. A new value can then be chosen directly from Ω_τ, potentially taking into account whether a change to other variables of the modified site or at other sites is acceptable or not. If changes to further variables are required, new values for them are subsequently determined from the according sets Ω_j.

For example, if a change of luminaire M_k at site k is considered and a same-flux constraint ξ applies, we initially determine all luminaires $M \in \mathcal{M}_{\psi(k)} \setminus \{M_k\}$ other than the current one together with the respective flux intervals $\mathbf{\Phi}_{\psi(k)}(M)$ that are allowed according to the luminaire constraints $\psi(k)$, storing them in a 2D set Ω. These flux intervals $\mathbf{\Phi}_{\psi(k)}(M)$ are subsequently intersected with the overall flux ranges $\cup_{M\in\mathcal{M}_{\psi(k')}} \mathbf{\Phi}_{\psi(k')}(M)$ that are permissible at the other sites k' affected by the same-flux constraint, flagging those subintervals that would require a change of luminaire at at least one of these other sites. Finally, if the pruned candidate set $\Omega|_M$ is not empty, a new luminaire from the set is selected and further changes to other triggered variables are carried out, yielding a new valid configuration.

5.4 Optimization Approach

Finding a valid configuration $\Lambda^* \in \mathcal{V}$ that satisfies the lighting goals to the best extent possible eludes a direct, exact solution due to the complexity of the problem. We thus explore the space of valid configurations $\mathcal{V}$ using a Markov chain Monte Carlo approach [Robert and Casella 2004].

A new chain is started by constructing a valid configuration as described in Section 5.2. In each successive iteration, a configuration proposal $\Lambda' \in \mathcal{V}$ is generated based on the chain's current state Λ and accepted with probability [Metropolis et al. 1953]

$$\alpha(\Lambda'|\Lambda) = \min\{1, \exp\big((C(\Lambda) - C(\Lambda'))/T\big)\},$$

where the employed Boltzmann-like energy distribution function favors good lighting solutions. The cost function $C(\hat{\Lambda})$ assesses how well the lighting goals are satisfied (Section 5.5), which involves computing the lighting induced by $\hat{\Lambda}$ (Section 5.6). If the cost increases by Λ', the proposal still has a chance of being accepted, which helps to avoid getting stuck in a local minimum. As the number of accepted configuration proposals increases, we gradually lower the temperature T, reducing the acceptance probability for cost-increasing configurations. Please refer to the supplemental material for details on the annealing schedule as well as other implementation-specific choices.

Elementary mutations. Typically, the proposal Λ' is derived from Λ by changing the value of one random variable τ. This may entail changes to further variables. We employ the constructive approach from Section 5.3 and determine the new value τ' by applying a relative change $\Delta\tau$ to the current value for noncategorical variables and randomly choosing a new value otherwise. $\Delta\tau$ is sampled from a truncated normal distribution [Robert 1995] whose standard deviation is a function of T, thus increasingly favoring small changes over time.

Another available mutation of Λ is the variant of only changing the lamp at a site, that is, changing the luminaire to a different one that employs the same body model but has a different luminous intensity distribution and flux range.

Performance enhancements. Striving to not just finding a good solution but also keeping the number of needed iterations low, we diversify the navigation of the search space by running multiple independent chains and generating multiple random initial configurations. Additionally, the elementary mutations are complemented with composite mutations; these come in two flavors: higher-level mutations (e.g., focus mutation), corresponding to orchestrated sequences of elementary mutations, and random sequences of multiple mutations (see what follows). Moreover, information about the induced lighting situation is harnessed to make efficacious change proposals (Section 5.7).

Interleaved exploration of chains. If the progress in lowering the solution cost $C(\Lambda)$ has stalled for some time, we randomly suspend the current chain and start a new one. For its initial state, we either generate a new valid configuration, thus considering a potentially largely different configuration, or pick the configuration Λ^* that yielded the lowest cost so far, thus resuming trying to further improve this best solution. This way, ultimately multiple initial solutions are explored in an interleaved fashion.

Instead of creating just one new initial configuration, we actually generate multiple initial configurations in successive iterations (20 for all results in the article). The first one is always accepted, while for the following ones the acceptance probability $\alpha(\Lambda'|\Lambda)$ applies. This strategy aims at avoiding spending many iterations on exploring a bad initial solution and has proven successful in practice. Effectively, it results in switching between coarsely exploring the solution space and refining (or exploiting) a certain configuration.

Focus mutation. Going beyond direct changes to a single variable, a higher-level focus operation is supported and its application to a random site in Λ augments the arsenal of mutation procedures. For a given installation site, the focus operation randomly selects a goal site (weighted by proximity) that potentially may be illuminated, and then directs the luminaire(s) at the installation site towards this goal site. Both the orientation direction and a luminaire body model's tilt parameter may be changed by this focusing, which is an effective means to assign a luminaire to a different goal site. In particular, achieving the same modification by repeatedly applying random variable changes can easily require an impractical number of iterations. When generating an initial configuration, we apply the focus operation to all installation sites.

Multiple mutations per iteration. Sometimes, only the interplay of several elementary (and focus) mutations may yield an improvement in cost. However, realizing them over successive iterations is unlikely if any single of these mutations applied individually would temporarily substantially increase the cost. To alleviate such situations, we perform a random number m of mutations when generating a new proposal. On the other hand, multiple random mutations may not necessarily act together in a synergistic way and can even inhibit each other. Therefore, care should be taken that it cannot happen too often that two unrelated changes are made and accepted, with one improving and one worsening the situation, as dealing with them in separate iterations would be preferable in those cases. We address this by choosing m's distribution to strongly favor a single mutation.

5.5 Cost Function

The cost for a luminaire configuration Λ is defined as

$$C(\Lambda) = \kappa_G C_G(\Lambda) + C_E(\Lambda)$$

and assesses both how well the specified objectives are met (C_G) and how effectively the luminaires' flux is employed for lighting the goal sites (C_E).

Goal satisfaction. The total cost for missing objectives is

$$C_\mathrm{G}(\Lambda) = \sum_i \frac{g_i}{\sum_{o\in\mathcal{O}_i} w_{o,i}} \sum_{o\in\mathcal{O}_i} w_{o,i}\, C_o(\Lambda, i),$$

where g_i is the relative importance weight of goal i, $\mathcal{O}_i$ denotes the set of objectives defined for goal i, and $w_{o,i}$ is the weight for the objective o of goal i. The individual cost functions C_o penalize a deviation of the lighting result from the objective's target.

For minimum and maximum objectives, the costs

$$C_\mathrm{min}(\Lambda, i) = \frac{1}{A_i}\int_{\mathcal{P}_i} \max\{min_i - v_\Lambda(\mathbf{p}), 0\}\, dA_\mathbf{p} \quad \text{and}$$
$$C_\mathrm{max}(\Lambda, i) = \frac{1}{A_i}\int_{\mathcal{P}_i} \max\{v_\Lambda(\mathbf{p}) - max_i, 0\}\, dA_\mathbf{p}$$

consider how much the result values $v_\Lambda(\mathbf{p})$ are below or above the target value of min_i or max_i, respectively, on average, where $\mathcal{P}_i$ comprises the goal sites of goal i and $A_i = \int_{\mathcal{P}_i} dA_\mathbf{p}$. The cost for a mean objective with a target value of μ_i is

$$C_\mathrm{mean}(\Lambda, i) = \left|\mu_i - \mu_{\Lambda,i}\right|,$$

where $\mu_{\Lambda,i} = \frac{1}{A_i}\int_{\mathcal{P}_i} v_\Lambda(\mathbf{p})\, dA_\mathbf{p}$, while for a variance objective, the cost

$$C_\mathrm{var}(\Lambda, i) = \max\left\{\sqrt{var_{\Lambda,i}} - \sqrt{var_i}, 0\right\},$$

with $var_{\Lambda,i} = \frac{1}{A_i}\int_{\mathcal{P}_i}(v_\Lambda(\mathbf{p}))^2\, dA_\mathbf{p} - (\mu_{\Lambda,i})^2$, penalizes exceeding the specified upper bound var_i. Finally, extreme uniformity and mean uniformity objectives with target lower bounds of eu_i and mu_i, respectively, have costs

$$C_\mathrm{eu}(\Lambda, i) = 1 - eu_{\Lambda,i}/eu_i \quad \text{and} \quad C_\mathrm{mu}(\Lambda, i) = 1 - mu_{\Lambda,i}/mu_i,$$

where

$$eu_{\Lambda,i} = \frac{\min_{\mathcal{P}_i} v_\Lambda(\mathbf{p})}{\max_{\mathcal{P}_i} v_\Lambda(\mathbf{p})} \quad \text{and} \quad mu_{\Lambda,i} = \frac{\min_{\mathcal{P}_i} v_\Lambda(\mathbf{p})}{\mu_{\Lambda,i}}$$

if $\mu_{\Lambda,i} > 0$ and $eu_{\Lambda,i} = mu_{\Lambda,i} = 0$ otherwise.

Luminaire effectiveness. To encourage that installed luminaires are actually and mainly contributing to the illumination of goal sites, we penalize other uses with a cost of

$$C_\mathrm{E}(\Lambda) = \kappa_\mathrm{ng}\frac{1}{|\mathcal{I}^{>0}(\Lambda)|}\sum_{k\in\mathcal{I}^{>0}(\Lambda)} \bar{\Phi}^\mathrm{ng}_{\Lambda,k} + \kappa_\mathrm{w}\frac{1}{|\mathcal{I}^{>0}(\Lambda)|}\sum_{k\in\mathcal{I}^{>0}(\Lambda)} \bar{\Phi}^\mathrm{w}_{\Lambda,k},$$

where $\mathcal{I}^{>0}(\Lambda)$ is the set of installation sites with at least one luminaire. $\bar{\Phi}^\mathrm{ng}_{\Lambda,k}$ is the fraction of the flux averaged over all luminaires installed at site k that ends up lighting nongoal surfaces, whereas $\bar{\Phi}^\mathrm{w}_{\Lambda,k}$ corresponds to wasted illumination, like shining into the sky.

For the weighting constants, we use $\kappa_\mathrm{G} = 10$, $\kappa_\mathrm{ng} = 1$ and $\kappa_\mathrm{w} = 3$.

5.6 Lighting Evaluation

To evaluate how well the specified goals have been reached, the illumination at the goal sites needs to be determined and aggregate quantities that allow checking the objectives must be derived.

Calculation points. We sample the goal sites at a number of points and compute the illumination for these *calculation points*, which is also a common approach in evaluating a real-world lighting design [IESNA 2000]. Specifically, during initialization, we impose a uniform grid on each goal site of a goal i. For each grid cell that overlaps the site's surface, we create a sample point $\mathbf{p}_{i,j}$ as close to the cell center as possible and assign it a weight $a_{i,j}$ according to the surface area overlapping the grid cell. When evaluating the cost function C, integrals $\int f(v_\Lambda(\mathbf{p}))\, dA_\mathbf{p}$ are approximated by weighted sums $\sum_j a_{i,j} f(v_{\Lambda,i,j})$, where $v_{\Lambda,i,j} = v_\Lambda(\mathbf{p}_{i,j})$.

Lighting computation. Given a configuration Λ, the illumination $v_{\Lambda,i,j}$ at each considered point $\mathbf{p}_{i,j}$ is determined on the GPU in a pixel shader, where a separate pass is performed for each luminaire. The according luminous intensity distribution is provided as a texture and visibility is resolved using a cube shadow map. To make a shadow map update only necessary if the light position has changed, we opted to not include luminaires in the shadow map, thus ignoring the (typically negligible) occlusion they cause.

Result aggregation. The illumination values are then aggregated per goal site with a segmented parallel reduction using compute shaders. The determined quantities are stored for better situation analysis (Section 5.7) and subsequently further aggregated per goal to derive statistics that enable assessing how well the goals' objectives are met.

Luminaire effectiveness. Additionally, we compute for each luminaire how much of its flux actually hits goal sites, hits other surfaces, and is wasted by not hitting anything. To this end, first all goal sites are rendered into a cube depth map centered at the luminaire. Subsequently, for each direction of a discretization of the spherical domain, we query both this goal site map and the shadow map to determine whether a goal site, any other surface, or nothing is hit, sample the luminous intensity distribution, and then assign the intensity to the determined hit class. Finally, a parallel reduction over all directions is performed, yielding the flux fractions for all three hit classes after normalization.

5.7 Efficacious Change Proposals

Starting many chains over time and applying random (elementary and focus) mutations as outlined in Section 5.4 will eventually explore the whole configuration space $\mathcal{V}$. However, such an uninformed approach is generally not efficient for finding good solutions, as the proposal generation is (largely) agnostic to the lighting situation induced by the current state Λ (focus mutations account for some spatial relationships). Therefore, we instead sample the proposal Λ' from a distribution that favors changes to the current state Λ that affect the cost C, ideally reducing it.

Concretely, we randomly decide whether to apply a random mutation or to take the actual lighting situation arising from Λ into account. In the latter case, we analyze the situation and try to come up with a modification of Λ that improves on it and hence leads to a lower cost. Our strategy is to address a specific cost factor, determine the set of relevant changes that can potentially reduce its cost, and pick one of them. To enable and support this strategy, we maintain information about which goal sites are and which may potentially be affected by an installation site and also keep installation-site- and goal-site-specific statistics from the lighting evaluation.

First, we randomly select an unsatisfied goal or a subeffective installation site according to their contribution to the overall cost C. In the case of a goal, we then randomly choose an objective according to the associated cost and randomly pick a goal site where the

objective is missed. Subsequently, we randomly select an installation site that either contributes to the illumination of the goal site or that may contribute but currently does not. We then determine all changes that may help to directly improve satisfying the objective and randomly pick one of them. In addition to elementary mutations to single variables, we also consider composite actions, like changing the luminaire, focusing it onto the goal site, and increasing the flux such that the goal site is within the light's cutoff distance.

If a subeffective installation site was selected, we aim to improve the effectiveness of its luminaire(s) by randomly applying either the focus operation, optionally in combination with a change of luminaire, or another random elementary mutation to this site.

6. PROCEDURAL LIGHTING DESIGN

In our system, a lighting design can be specified procedurally as part of an extended procedural modeling workflow.

Procedural modeling. Our procedural modeling system is inspired by CGA shape [Müller et al. 2006; Esri 2012] and extends it in several ways to enable effective lighting design. It builds on the concept of a shape, which comprises a symbol, a set of attributes, an oriented bounding box called scope, and geometry inside this scope. Each shape with symbol V is eventually refined by a production rule of the form $V \rightarrow A_1 \dots A_n$, unless no such rule was defined, thus making V a terminal symbol. If a rule is applied to a shape, the actions A_i are sequentially executed. An action is either simply a symbol W, which creates an instance of the current shape and assigns the symbol W to it, or an operation that modifies or splits the current shape, where a split operation decomposes the current shape into a set of shapes and performs actions on them according to the operation's arguments.

Starting from an initial shape that represents a lot, a whole building can be created by iteratively applying rules. This derivation process implicitly defines a shape tree, where the initial shape forms the root and the shapes produced by a rule's actions become the children of the shape for which the rule is applied. The final building model is then defined by the set of leaf nodes that correspond to all shapes with terminal symbols.

For procedural lighting design, we introduce several new operations and augment the shape tree to a *model tree*, where additional types of nodes other than shape nodes may occur. Analogous to before, the final model is then given by the subset of leaf nodes that are shape nodes.

Lighting goals. As detailed in Section 4.1, a lighting goal comprises both objectives and the goal sites for which they apply. Accordingly, a goal is specified in two steps in our modeling system. First, the objectives are defined with the new operation

lightingGoal(i, *quantity*, $spec_1, \dots$),

where i is an identifying (but not necessarily unique) number, *quantity* determines whether to consider illuminance or luminance, and the arguments $spec_k$ specify the concrete objectives; mean(ℓ, w), for instance, imposes a target mean illumination level of ℓ. Providing an objective's importance weight w is optional (defaults to 1), as is the case with the goal's weight, which can be specified via an argument weight(g).

The objectives are recorded in the model tree, with the operation creating a goal node and inserting it as the child of the currently active node, where the initial active node on executing a rule is the shape node for which the rule is applied. The generated goal node also becomes the new active node, and thus all nodes created by subsequent actions in the rule are successors of this goal node.

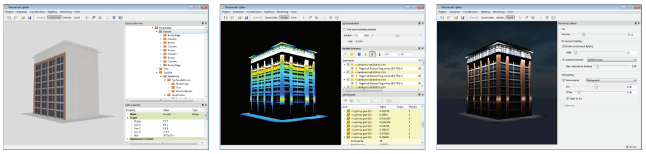

Fig. 11. Screenshots from our procedural lighting design system.

In a second step, the goal sites the objectives relate to are specified with the operation *lightingGoalSite*(*goal*). It takes the current shape, whose geometry is required to be a polygon in our prototype implementation, and creates a new goal site node for the corresponding surface. This node is added as a child to the current active node and, hence, is a leaf node in the model tree. Note that, by modifying the current shape and creating only a goal site but no new shape node for this shape state, virtual surfaces, like a plane above the floor, can easily be realized. The goal site is added to the site set of the goal that corresponds to the closest goal node ancestor of the goal site's node whose identifying number matches *goal*.

Luminaire installation sites. Candidate installation sites for placing luminaires are specified with the *lumSite*() operation that creates a new leaf installation-site node as child of the current active node. The line segments L_p and L_a defining the installation point are specified relative to the current shape using a local parameterization of its geometry, which must be a polygon.

Luminaire constraints. The *lumConstraints*() operation creates a new luminaire-constraints node and makes it the active node. The constraints are assigned to succeeding installation sites via an identifying number, analogous to how lighting goals are referenced by goal sites. This also makes it possible to initialize the new constraints with those of an ancestor luminaire-constraints node, facilitating a hierarchical constraint management.

Inter-site constraints. Analogously, inter-site constraints can be defined with the *interLumSiteConstraints*() operation. It generates a corresponding constraints node that becomes the active node. Again, the association with installation sites is based on an identifying number, where the mutual constraints apply to the group of all installation sites that reference the constraints node.

Further extensions. In addition to the discussed new operations that are directly concerned with lighting design, we incorporated several further extensions over CGA shape [Müller et al. 2006] and its CityEngine realization [Esri 2012] into our system that benefit the modeling capabilities and are useful for lighting design. Examples include a new split operation that yields the unoccluded parts of a shape, the notion of explicit construction stages that may be referenced by occlusion-related operations, an extended type system, and roofs with overhangs of nonzero thickness. More details are provided in Appendix A.

7. RESULTS AND DISCUSSION

Our system was implemented as part of a procedural modeling application (see Figure 11). After a grammar describing both the building and its lighting specification has been loaded, concrete instances of this procedural description can be created and inspected. To generate a lighting solution, the user selects a luminaire database and the system then verifies whether a configuration exists at all that satisfies the constraints. If this is not the case, an according message is shown to the user. Otherwise, the optimization procedure can be run and, after it has stopped, either because a configuration

Table I. Statistics for Selected Examples

Scene	Goals	Goal sites	Calculation points	Luminaire installation sites	Permissible luminaires[1]	Groups of constrained sites[2]	Inter-site constraints	Installed luminaires	Time[3]
Figure 1	10	10	16,325	12	19	3	10	26	10.66 s
Figure 2 c	18	18	35,374	8	19	1	1	8	13.58 s
Figure 3 c	32	32	73,704	72	7	2	8	72	121.23 s
Figure 3 d	32	32	73,704	44	3	1	4	44	72.37 s
Figure 13 a	4	4	36,000	8	19	1	1	8	8.20 s
Figure 13 d	4	4	36,000	4	19	1	2	4	5.46 s
Figure 14 a	7	7	14,217	12	14	4	14	28	17.36 s
Figure 14 b	11	810	42,852	20	6	4	12	22	18.66 s
Figure 15 a	1	2	88,676	14	8	2	6	14	48.08 s
Figure 15 b	1	3	195,764	24	12	2	4	24	77.67 s
Figure 15 c	1	12	204,303	32	5	2	8	32	95.19 s
Figure 16 a	1	12	214,368	12	16	6	18	50	22.65 s
Figure 16 d	3	8	10,842	7	17	1	1	12	6.96 s

The optimization was run for 2,000 iterations. [1]Number of different luminaires permitted by the design's luminaire constraints. [2]Number of groups of installation sites mutually constrained by inter-site constraints. [3]Average over five runs.

that satisfies all goals was found, a maximum number of iterations was reached, or the user aborted it, the lighting solution can be viewed and exported. The application also provides functionality to explore the evolution of the solution, the installed luminaires, and the illumination and goal satisfaction at the goals and their sites.

For rendering, the system uses Direct3D 11 and pursues a deferred lighting approach, employing cube shadow maps and texture-based luminous intensity distributions for the point lights (Section 5.6). Moreover, irradiance environment map lighting [Ramamoorthi and Hanrahan 2001] is performed and modulated by an approximate ambient occlusion solution. All results presented in this article are direct captures from our application, using the global photographic operator [Reinhard et al. 2002] for tonemapping. On our GeForce GTX 680 card, they can be rendered in real time with triple-digit frame rates at full-HD resolution.

7.1 Examples

The system has been successfully employed for modeling and designing the lighting of various architectural structures, ranging from residential buildings (Figures 1, 14, and 16(b)–(d)) over larger structures, like office buildings (Figures 2, 3), skyscrapers (Figure 16(a)), and warehouses (Figure 15), to monuments (Figure 13). Most of these structures and their lighting designs were motivated by real-world examples. For all results, we employed a common luminaire database populated with 3D models and photometric data made publicly available by lighting manufacturers; it comprises 19 luminaires, using 11 different luminaire models that span a wide range of luminaire types (see Figure 5 for some examples).

Table I provides a statistic overview of the modeled goals, installation sites, and constraints, as well as the shown results. The optimization was run for 2,000 iterations, generally yielding good solutions that satisfy the specified goals reasonably well.

The required optimization time, measured on a system with an Intel Xeon X5675 and a GeForce GTX 680, depends heavily on the concrete scene and its design specification. Rendering affected shadow maps (and deriving luminaire effectivenesses) usually dominates the time consumed by a single iteration in the optimization; consequently, this time is influenced by both the geometric complexity of the scene and how many light positions and directions were changed. Note that an installation site with potentially many installed luminaires, and especially an inter-site constraint comprising many installation sites, can cause a change to a single installed luminaire to trigger changes to many others, entailing a high iteration time.

7.2 Modeling Example

To provide an idea of what a lighting design specification may comprise, we consider the example in Figure 1. Overall, ten goals were defined for different façade parts, the court, and the driveway. Each goal aims for the illumination to fall into a certain luminance range and features a single goal site. In Figure 1(b), the (visible) goal sites are shown in red, while the yellow surfaces correspond to installation sites. At each installation site, the design allows for translation along the blue line segment and, if a green line is shown, the site may additionally accommodate more than one luminaire arranged along this line (Section 4.3). For instance, for each of the two sites on either side of the driveway, we permit between two and six luminaires. These two sites are further subjected to an inter-site constraint to ensure that both employ the same number of luminaires, the same spacing, the same luminaires, and the same flux per luminaire. Another inter-site constraint requires the two sites left and right of the garage door to use the same luminaire with the same flux and the same model parameter values (i.e., the same tilt angles) and to position them at the same height.

Respecting all constraints, the obtained result in Figure 1(c) satisfies the goals well, as also evidenced by the visualization of the scene luminance in Figure 1(d).

7.3 Optimization Progress

For the final result in Figure 3(c), the graph in Figure 12 shows how the cost C of the accepted solutions evolved during the optimization. Moreover, three of these intermediate solutions are shown above the graph. Please consult the supplemental video to view a full sequence of the accepted solutions.

The large cost increases in the graph correspond to starts of new chains performed to avoid getting trapped in a local minimum of the solution space. In these situations, the optimizer sometimes resumed further refining the best solution known at that point. The graph also illustrates that, due to the stochastic nature of the optimization, it can never be known whether a significantly better solution will be found soon. We observe, however, that after coarsely exploring the

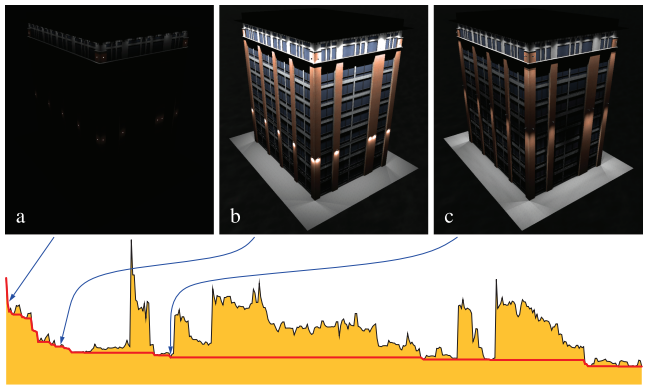

Fig. 12. Progression of the optimization for the design in Figure 3(c). The graph shows the cost of the accepted candidate solutions, with the red line indicating the cost of the best encountered solution.

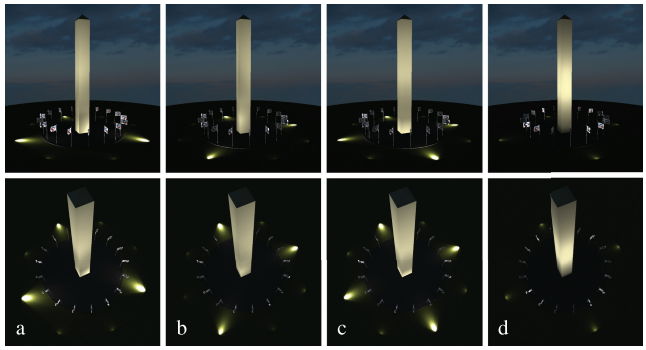

Fig. 13. Variants of lighting a monument using different installation-site specifications and constraints.

solution space with several generated initial configurations, generally a configuration is found that can be refined to a good result and that the optimization manages to perform such a refinement.

In practice, this means that good results can usually be achieved by running only a few thousand iterations, as demonstrated by the shown examples, which all relied on 2,000 iterations. Indeed, even the intermediate solutions (b) and (c) in Figure 12 already constitute reasonable results after less than 500 iterations. Note that, given the complexity of the problem, the many degrees of freedom, and the constraints they are subjected to, a few thousand iterations seems a rather low number, even for producing merely reasonable results, thus hinting at the efficiency of our optimization approach.

7.4 Variations

Luminaire installation sites and constraints. The monument example in Figure 13 showcases different luminaire-installation-site designs, all aiming at reaching a target illuminance value at all four sides of the obelisk. In variant (a), eight installation sites are specified that are constrained to use the same luminaire; variant (b) additionally fixes the orientation towards the center. Variant (c) partitions the sites into two interleaved groups, further requiring each to use the same model parameter value and the same flux. Only four installation sites are modeled in variant (d), constraining them to use the same luminaire and the same flux.

The example demonstrates the usefulness of constraints to implement a certain aesthetic taste. In general, the optimization may come

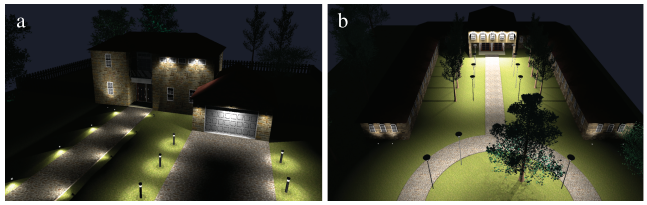

Fig. 14. Two examples of residential houses lit with our system.

up with solutions that are unexpected and potentially unwanted by the modeler because they are visually unpleasing or structurally objectionable to him, despite actually satisfying the goals rather well. This is highly subjective, though, and basically means that the modeler omitted some constraints from the design specification that establish his taste, which can be easily remedied by adding them.

Lighting goals. Figure 3 demonstrates varying lighting goals for a model. In Figure 3(d), the top floor is to be illuminated while the brick columns are supposed to remain dark, which is specified via a maximum target value of zero. By contrast, variant (c) aims at illuminating the building's façades and provides additional luminaire installation sites at the brick columns.

Building model. Applying a lighting design to varying instantiations of a procedural building model is shown in Figure 16(b)–(d). The model offers several degrees of variation, like the number of stories or whether to include certain installation sites (e.g., the one behind the hedge). The lighting goals always aim at illuminating the driveway, the entrance, and the front façade with the ground-level windows.

A more complex example yielding widely different instances is depicted in Figure 15. In all variants, the goal is to illuminate the parking space surrounding the warehouse.

7.5 Direct Modeling of Lighting Solutions

Even in seemingly simple examples, usually many complex decisions have to be made to arrive at a luminaire configuration that well satisfies the goals, like choosing the correct luminaire, directing it appropriately, and determining the right flux. Obtaining a good solution by directly modeling luminaires as part of a building (rather than using our system) is hence extremely hard, especially when procedural variations are in place where this is basically impractical. In particular, the necessary direct reasoning about the cumulative effect of all lights and the geometric relationships induced by the existing procedural rules is generally not feasible, let alone the entailed challenge of formulating it in the modeling system's grammar language. Note that such procedural languages describe successive local refinements of shapes and (currently) lack specific means for global coordination (beyond limited occlusion queries) and referencing other shapes. Any nonlocal decision affecting more than one shape has hence to be expressed as a local decision no later than when their common parent shape gets refined (at which point these shapes consequently do not even exist yet).

Figure 2 shows a specific example where direct modeling appears comparatively simple. To reach the goal of illuminating all façades, we selected a promising luminaire and directed all luminaire instances inwards. While this indeed manages to illuminate the façades as shown in Figure 2(a), for considerable parts of them the illumination is very weak, as can be clearly seen in the illuminance visualization shown in Figure 2(b). Of course, we could incrementally try different luminaires or adapt the flux, rotation,

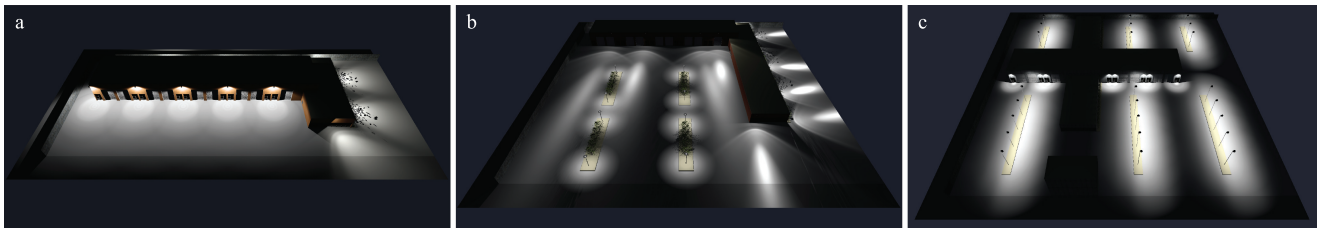

Fig. 15. Three example instances of a procedural warehouse model, where the lighting design aims at illuminating the parking space.

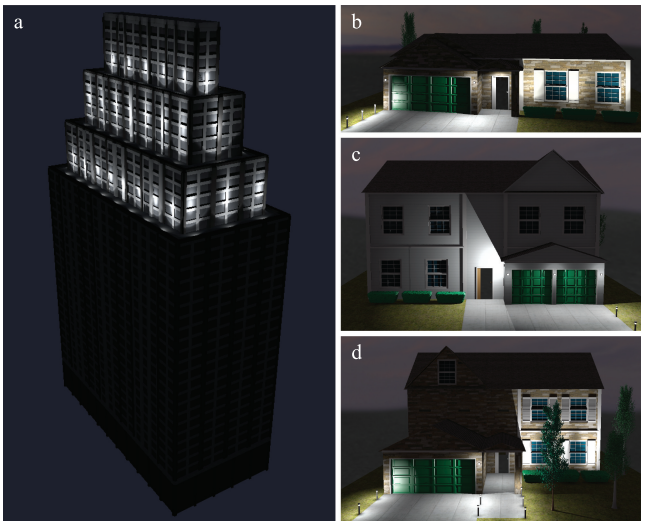

Fig. 16. (a) A skyscraper with its top being illuminated; and (b)–(d) three variants of a procedural residential house model.

and tilt angle to improve the solution, but this requires many manual iterations and only holds for a single variation. By contrast, the result obtained by specifying goals and running the optimization directly provides good results (Figure 2(c) and (d)) and specifying the lighting design is comparatively easy.

7.6 Importance of Hard Constraints

A key characteristic of our optimization approach is that the modeled constraints are always satisfied. To test the importance of this feature for the progress and success of the optimization, we treat all inter-site constraints as soft constraints. A penalty is determined for all violated constraints, scaled by a weighting constant, and added to the overall solution cost C. Alternatively, the soft constraints could be encoded as factors [Yeh et al. 2012] (a test did not indicate any improvement). We also don't explicitly account for inter-site constraints when generating the initial solution and perform only random (elementary and focus) mutations. Altogether, this brings the optimization procedure closer to a "standard" simulated annealing approach.

Soft constraints obviously don't guarantee that the constraints are eventually satisfied. Moreover, their penalty cost may have undesirable effects, like trading goal fulfillment for reduction in penalty. It is hence not surprising that such an approach performs badly, especially for heavily constrained models. As an example, Figure 3(a) and (b) show results after 2,000 and 200,000 iterations. Even in the latter case, important constraints are violated, yielding a visually unsatisfying luminaire pattern. On top of that, the lighting solution leaves room for improvement (e.g., the façades end up too dark). It is particularly revealing to compare these results with the intermediate solutions in Figure 12(b) and (c), obtained with our approach after less than 500 iterations.

7.7 Scalability to Many Buildings

In principle, our system could be directly applied to larger environments comprising many buildings, where the lighting of one building accounts for the influence of the neighboring buildings' lighting. However, if an optimization across multiple, independent building groups were indeed desired, it would be better to first optimize per building group (e.g., all buildings on a lot) and then run the optimization on all buildings for fine-tuning the illumination, which benefits from the influence of individual luminaires being rather localized.

We conjecture that, in practice, especially in residential areas, the lighting design for most building is usually performed in isolation, not the least because one can generally hardly rely on the lighting from surrounding lots. Consequently, we focus on lot-level scenarios and leave exploring large-scale scenes, which also entails extensions to the modeling system like street network support, for future work.

7.8 Limitations and Future Work

For a first-time user, our system requires some time to learn how to effectively specify a lighting design and, if no prior experience with procedural modeling exists, the initial learning curve can be clearly steep. For instance, the examples in Figure 16(b)–(d) were created by a computer-science undergraduate student with no background in procedural modeling and lighting design. After receiving our application and an 11-page manual, it took him about 20 hours to come up with the first reasonable models with some variations. A considerable amount of the time was spent getting familiar with procedural modeling in general.

While we currently account only for direct illumination during lighting evaluation, it would be interesting to incorporate a fast global illumination algorithm. However, despite the wide variety of proposed interactive approaches [Ritschel et al. 2012], none of them appears to be particularly well suited for dealing with many primary lights with complex emission profiles. Moreover, any method can be expected to entail substantially increased iteration times.

Owing to the procedural modeling context considered, the lighting design specification is currently explicitly defined. An interesting direction is to further automate this design task. For instance, one might try to learn goals, installation sites, and constraints from example models. Alternatively, semantic annotations and attributes of scene elements may be harnessed to derive a specification from recommendations in lighting standards. Another avenue is that of

offering an interactive design component that allows the user to edit specification parameters, to modify and add constraints, and to trigger changes like choosing a different or selecting a specific luminaire.

8. CONCLUSION

We have presented a system for lighting design where the design is specified in terms of lighting goals, luminaire installation sites, and constraints, and an according lighting solution that comprises luminaires placed in the scene is automatically determined. The specification is expressed procedurally using extensions to a grammar language for the procedural modeling of buildings. This allows both for a natural integration into the modeling workflow and for the direct application of a design to varying instances of a building model. The intricate problem of determining a luminaire configuration that respects all constraints of the specification and satisfies the goals as well as possible is addressed by an advanced stochastic approach that directly navigates in the complex subspace of valid configurations. We applied our system to various examples, demonstrating its effectiveness.

APPENDIX

A. GENERAL EXTENSIONS TO CGA SHAPE

In order to better address some modeling challenges we encountered, our procedural modeling system provides several extensions (in addition to those related to lighting design) over CGA shape [Müller et al. 2006] and its subsequent commercial realizations [Esri 2012]. In the following, we discuss some of them.

Unoccluded parts. During modeling, often simple base masses are intentionally placed such that intersections and thus surface occlusions occur to form a more complex mass. While the result of an occlusion query may then frequently be sufficient to decide how to further refine a shape, like whether to make it a (partially hidden) wall or develop it into a (fully visible) window, it clearly is not enough once simply keeping the occluded part is not an option, like when the visible part of a wall should be uniformly split and then refined. In particular, for specifying a goal site, it is mandatory to avoid any occluded parts as these can never be illuminated, precluding meeting constraints like minimum and biasing the solutions for others like mean. To address this, we introduced a new split operation *unoccludedParts* that determines the unoccluded parts of the current shape and executes the actions specified in the operation's argument list on them. Optionally, the occluder tested against can be enlarged in the direction normal to the currently considered face to improve robustness.

Construction stages. For correct results, it is necessary that, when an occlusion test is executed, all relevant occluding shapes have already been generated. However, sometimes neither CGA shape's original approach of statically assigning priorities to rules and then selecting the next shape to refine based on these priorities nor CityEngine's new solution of constructing the model twice, with the second pass using the first pass' result for occlusion testing, provide enough control.

We address this problem by allowing the whole model generation to be structured into multiple *construction stages*. A crucial role is played by the new operation stage(k) that indicates that, in the current branch of refinement, the construction stage k has been reached. This is reflected in the model tree by adding a new stage node that then becomes the new active node. If k is higher than the globally achieved construction stage K, which is initially zero, then the execution of the current rule is suspended and the processing continues with a shape that has not been refined yet. If no more such shapes exist, K is increased to the lowest stage reached by all suspended rule executions, and the according rule executions are resumed one after another. This process is run until no more pending executions exist.

To ensure that a certain stage k' has been reached before the action A is executed, it thus suffices to precede A with stage(k'). Moreover, we augmented some operations to optionally take stage information into account. Occlusion computations, for instance, may be restricted to consider only those shapes that were available at a certain stage. If this stage has not yet been reached globally, the rule execution is suspended until K has accordingly advanced.

It is interesting to note that stage(k) allows guiding the derivation process into a growth process with discrete growth steps (as k can be any expression, the number of steps is basically unbound). This operation also naturally serves as a synchronization point and hence offers the possibility for a controlled communication between different branches of construction; augmenting the expressiveness of the grammar language accordingly remains for future work.

Materials. We further extended the type system to treat entities like assets and materials as first-class objects; these can be defined globally (with a unique identifier) but also constructed on the fly. This enables operations like replacing one specific material used in an asset with a user-defined one and restricting the texture coordinate projection operation to faces with selected materials, which we use to adjust parts of an imported asset to match the rest of the model, like making all wall parts consistently textured.

Roofs. Our system supports common roof types, like hipped and gabled, where all roof overhangs produced are of nonzero thickness and thus feature an eave or rake-board face and separate faces for the rooftop and soffit parts. To facilitate selecting a specific part of a roof in a component split, we further introduced semantic tags that can be associated with parts of a shape.

REFERENCES

F. Anrys and P. Dutré. 2004. Lighting design by simulated annealing. Tech. rep. CW393, Katholieke Universiteit Leuven.

A. C. Costa, A. A. Sousa, and F. N. Ferreira. 1999. Lighting design: A goal based approach using optimisation. In *Proceedings of the Eurographics Workshop on Rendering.* 317–328.

R. Dechter. 2003. *Constraint Processing*. Morgan Kaufmann, San Fransisco.

Esri. 2012. Esri CityEngine 2012.1.

S. Gumhold. 2002. Maximum entropy light source placement. In *Proceedings of the IEEE Conference on Visualization*. 275–282.

IESNA. 2000. *The IESNA Lighting Handbook* 9th Ed. Illuminating Engineering Society of North America, New York.

J. K. Kawai, J. S. Painter, and M. F. Cohen. 1993. Radioptimization – Goal based rendering. In *Proceedings of SIGGRAPH'93*. 147–154.

T. Kelly and P. Wonka. 2011. Interactive architectural modeling with procedural extrusions. *ACM Trans. Graph.* 30, 2, 14:1–14:15.

L. Krecklau and L. Kobbelt. 2011. Procedural modeling of interconnected structures. *Comput. Graph. Forum* 30, 2, 335–344.

L. Krecklau, D. Pavic, and L. Kobbelt. 2010. Generalized use of non-terminal symbols for procedural modeling. *Comput. Graph. Forum* 29, 8, 2291–2303.

C. H. Lee, X. Hao, and A. Varshney. 2004. Light collages: Lighting design for effective visualization. In *Proceedings of the IEEE Visualization Conference*. 281–288.

W.-C. Lin, T.-S. Huang, T.-C. Ho, Y.-T. Chen, and J.-H. Chuang. 2013. Interactive lighting design with hierarchical light representation. *Comput. Graph. Forum* 32, 4, 133–142.

M. Lipp, P. Wonka, and M. Wimmer. 2008. Interactive visual editing of grammars for procedural architecture. *ACM Trans. Graph.* 27, 3, 102:1–102:10.

P. Merrell, E. Schkufza, and V. Koltun. 2010. Computer-generated residential building layouts. *ACM Trans. Graph.* 29, 6, 181:1–181:12.

P. Merrell, E. Schkufza, Z. Li, M. Agrawala, and V. Koltun. 2011. Interactive furniture layout using interior design guidelines. *ACM Trans. Graph.* 30, 4, 87:1–87:9.

N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. 1953. Equation of state calculations by fast computing machines. *J. Chem. Phys.* 21, 6, 1087–1092.

P. Müller, P. Wonka, S. Haegler, A. Ulmer, and L. Van Gool. 2006. Procedural modeling of buildings. *ACM Trans. Graph.* 25, 3, 614–623.

M. Okabe, Y. Matsushita, L. Shen, and T. Igarashi. 2007. Illumination brush: Interactive design of all-frequency lighting. In *Proceedings of the Pacific Graphics Conference.* 171–180.

Y. I. H. Parish and P. Müller. 2001. Procedural modeling of cities. In *Proceedings of SIGGRAPH 2001.* 301–308.

G. Patow. 2012. User-friendly graph editing for procedural modeling of buildings. *IEEE Comput. Graph. Appl.* 32, 2, 66–75.

F. Pellacini, F. Battaglia, R. K. Morley, and A. Finkelstein. 2007. Lighting with paint. *ACM Trans. Graph.* 26, 2, 9:1–9:14.

F. Pellacini, P. Tole, and D. P. Greenberg. 2002. A user interface for interactive cinematic shadow design. *ACM Trans. Graph.* 21, 3, 563–566.

P. Poulin and A. Fournier. 1992. Lights from highlights and shadows. In *Proceedings of the ACM Symposium on Interactive 3D Graphics*. 31–38.

P. Prusinkiewicz and A. Lindenmayer. 1990. *The Algorithmic Beauty of Plants*. Springer, New York.

R. Ramamoorthi and P. Hanrahan. 2001. An efficient representation for irradiance environment maps. In *Proceedings of SIGGRAPH 2001*. 497–500.

E. Reinhard, M. Stark, P. Shirley, and J. Ferwerda. 2002. Photographic tone reproduction for digital images. *ACM Trans. Graph.* 21, 3, 267–276.

T. Ritschel, C. Dachsbacher, T. Grosch, and J. Kautz. 2012. The state of the art in interactive global illumination. *Comput. Graph. Forum* 31, 1, 160–188.

C. P. Robert. 1995. Simulation of truncated normal variables. *Statist. Comput.* 5, 2, 121–125.

C. P. Robert and G. Casella. 2004. *Monte Carlo Statistical Methods* 2nd Ed. Springer, Verlag.

C. Schoeneman, J. Dorsey, B. Smits, J. Arvo, and D. Greenberg. 1993. Painting with light. In *Proceedings of SIGGRAPH'93*. 143–146.

R. Shacked and D. Lischinski. 2001. Automatic lighting design using a perceptual quality metric. *Comput. Graph. Forum* 20, 3, 215–226.

A. Shesh and B. Chen. 2007. Crayon lighting: Sketch-guided illumination of models. In *Proceedings of Graphite'07.* 95–102.

G. Stiny. 2006. *Shape: Talking about Seeing and Doing*. MIT Press, Cambridge, MA.

G. Stiny and J. Gips. 1972. Shape grammars and the generative specification of painting and sculpture. In *Information Processing 71: Proceedings of IFIP Congress 1971.* 1460–1465.

J. O. Talton, Y. Lou, S. Lesser, J. Duke, R. Měch, and V. Koltun. 2011. Metropolis procedural modeling. *ACM Trans. Graph.* 30, 2, 11:1–11:14.

C. A. Vanegas, J. Garcia-Dorado, D. G. Aliaga, B. Benes, and P. Waddell. 2012. Inverse design of urban procedural models. *ACM Trans. Graph.* 31, 6, 168:1–168:11.

E. Whiting, J. Ochsendorf, and F. Durand. 2009. Procedural modeling of structurally-sound masonry buildings. *ACM Trans. Graph.* 28, 5, 112:1–112:9.

P. Wonka, M. Wimmer, F. X. Sillion, and W. Ribarsky. 2003. Instant architecture. *ACM Trans. Graph.* 22, 3, 669–677.

Y.-T. Yeh, L. Yang, M. Watson, N. D. Goodman, and P. Hanrahan. 2012. Synthesizing open worlds with constraints using locally annealed reversible jump MCMC. *ACM Trans. Graph.* 31, 4, 56:1–56:11.

L.-F. Yu, S.-K. Yeung, C.-K. Tang, D. Terzopoulos, T. F. Chan, and S. Osher. 2011. Make it home: Automatic optimization of furniture arrangement. *ACM Trans. Graph.* 30, 4, 86:1–86:11.

R. Zmugg, S. Havemann, and D. W. Fellner. 2010. Towards a voting scheme for calculating light source positions from a given target illumination. In *Proceedings of the Eurographics Italian Chapter Conference*. 41–48.

Received November 2013; revised February 2014; accepted March 2014