Interior Illumination Design Using Genetic Programming

Kelly Moylan and Brian J. Ross(✉)

Department of Computer Science, Brock University,
500 Glenridge Avenue, St. Catharines, ON L2S 3A1, Canada
{km07ok,bross}@brocku.ca
http://www.cosc.brocku.ca/

Abstract. Interior illumination is a complex problem involving numerous interacting factors. This research applies genetic programming towards problems in illumination design. The Radiance system is used for performing accurate illumination simulations. Radiance accounts for a number of important environmental factors, which we exploit during fitness evaluation. Illumination requirements include local illumination intensity from natural and artificial sources, colour, and uniformity. Evolved solutions incorporate design elements such as artificial lights, room materials, windows, and glass properties. A number of case studies are examined, including a many-objective problem involving 6 illumination requirements, the design of a decorative wall of lights, and the creation of a stained-glass window for a large public space. Our results show the technical and creative possibilities of applying genetic programming to illumination design.

Keywords: Illumination · Genetic programming · Radiance · Many-objective optimization

1 Introduction

The illumination design of interior spaces is a challenging task. Within any space, the efficient use of both natural and artificial light sources is required, as rooms are often occupied during both day and night. Light reflects off of walls, floor, furniture, and other objects, and the degree of reflection depends on their geometry and material composition. Although large windows can fill an environment with sunlight, designers must ensure that occupants are also protected from harsh brightness and glare. These same rooms can then be illuminated during the night, with artificial lighting designed to create completely different moods and settings. Although the aesthetic nature of this task is difficult to formally model, technical requirements such as illumination levels, colours, material effects, and others, are more easily quantified.

This paper explores evolutionary design problems in illumination using genetic programming (GP) [1].[1] We use the Radiance system for evaluating aspects

[1] See http://www.cosc.brocku.ca/~bross/IllumGP/ for more details about this research.

C. Johnson et al. (Eds.): EvoMUSART 2015, LNCS 9027, pp. 148–160, 2015.
DOI: 10.1007/978-3-319-16498-4_14

of illumination in an environment [2,3]. Radiance uses a sophisticated simulation of illumination, which lets it accurately account for a wide range of relevant factors. By using Radiance, the GP system can consider a variety of factors for artificial and natural (solar) illumination. Since multiple and often conflicting instances of lighting specifications can be used, the problem is multi-objective in nature. By using the "many objective" technique of sum of rank (or average rank), we consider problems using up to 6 factors. The GP language addresses a host of factors relevant to illumination design, including artificial lighting (location, intensity, colour), windows (size, locations, colour), and room materials (walls, floors, ceiling).

Another contribution made is the application of GP to some innovative problems in illumination design. First, we design a decorative "wall of lights", comprising a variable-sized grid of coloured lights. In another problem, an enormous stained-glass window is generated for a public space. In both these problems, we exploit GP's ability to evolved mathematical expressions for generating the colour of lights and windows, essentially evolving procedural textures to solve the problems. The evolution of procedural textures has an established history in evolutionary design [4,5].

The paper is organized as follows. Section 2 reviews some relevant concepts in illumination design, and discusses the Radiance illumination modeling system. We discuss the implementation of our system architecture in Sect. 3. Experiments and results are reported in Sect. 4, including a many-objective room illumination problem, a decorative wall of lights, and a stained glass wall. Concluding discussions are in Sect. 5.

2 Background

2.1 Illumination Design

Illumination is a topic of specialization within architecture and interior design [6]. The inverse illumination problem involves finding of potential lighting options for a pre-defined interior space [7]. It takes into account the desired lighting requirements in various locations in the space, and looks for optimal positions, kinds, and number of light sources. This allows a degree of freedom and creativity to solve illumination problems, given the many feasible solutions possible. Energy efficiency are often considered as well, since passive solar illumination can be used for both illumination requirements and energy saving (artificial light reduction, room heating).

The challenges of illumination design makes it an interesting problem for computer automation and computational intelligence. Fernandez [8] used a heuristic search to recreate an initial lighting scene. The program was tasked with finding placements of skylights to illuminate a pre-made building layout. Tena [7] used an interactive genetic algorithm to find solutions for inverse illumination problems. Castro [9] used a number of different heuristic and evolutionary based approaches, which searched for a desired illumination solution using optimally minimum emission power. Caldas [10] investigated energy efficiency concerns

involving illumination, using a multi-objective genetic algorithm. Another approach to the problem is to design a building model incorporating specific illumination characteristics – and especially in regards to solar illumination [11,12].

2.2 Radiance

Radiance is a well known open-source illumination simulation tool for designers and architects [2,3]. It implements a physically-based backward raytracer rendering algorithm that accounts for a number of important environmental factors, including light intensity, materials, geographic location, time, date, and others.

To use Radiance, one first defines a 3D model of the environment. For a room, this will be the obvious room walls, ceiling, floors, and windows. Material definitions specifying colours and reflectivity are also defined. The room can be filled with objects, such as furniture and decoration, and materials are similarly supplied. Artificial lighting is defined, and involves locations, colours, and intensities. The geographic location, time, and date are then supplied, in order to simulate natural sunlight accurately.

Once the environment is defined, Radiance performs an illumination simulation. A number of outputs can be obtained. Individual lighting measurements can be sampled at locations of interest. Lighting characteristics such as glare can be measured. An overall rendering of the scene can be generated. This can be a photorealistic image, or a colour-coded illuminosity map.

3 System Design

3.1 Genetic Programming Language

A strongly-typed GP language [13] is used (Table 1). Types designate specific design tasks within the environment. This defines GP trees similar to those used in grammar-guided GP [14]. For example, W is a type denoting the generation of windows and skylight for a room, and the *Windows* operator fulfills that design task. Numeric types include float (F), integer (I), and tree float (TF). Some of functions used are:

- *Materials(...)* assigns materials for the 4 walls, ceiling, and floor. If not used, pre-defined materials are assigned.
- *North_Wall_Center(I,a,b,c,d)* creates windows for the north wall. "*I*" is converted to a value between 0 and 30, and is the number of wall panels or sections to create for windows. The *a*, *b*, and *c* arguments use the fraction portion of the float value. They scale the windows, as shown in Fig. 1(a). All the windows on a wall's panels will have the same scale. The d argument determined whether a window is to be created on a panel. Its floating point expression is given the panel coordinates. If the value is positive, a window is defined on that panel. Otherwise no window is created.

Table 1. Language

Type	Function Name	Description
R	`Root(W,MM,LM,...)`	Creates the scene. Parameters vary according to design task
LM	`Top_Light(LB)`	Creates the artificial lighting
W	`Windows(C,NW,SW,EW,NW)`	Creates walls and windows
MM	`Materials(M,M,M,M,M,M)`	Creates materials
M	`(F,F,F,F,F)`	Material defn: R, G, B, reflection, roughness
C	`SkyLight(F,F,F,F)`	Skylight window on the ceiling. Args define coords for 2 opposite corners
NW	`North_Wall_Center(I,F,F,F,TF)`	Evenly patterns the wall of the
SW	`South_Wall_Center(I,F,F,F,TF)`	room based on percentage
EW	`East_Wall_Center(I,F,F,F,TF)`	measures for window size and
NW	`West_Wall_Center(I,F,F,F,TF)`	location on wall section
LM	`Top_Light(LB)`	Creates artificial lighting
LB	`Light_Filler(LB,LB)`	Branches the light creation tree
LB	`Basic_Light(F,F)`	Light source of fixed white intensity. Args define location
LW	`Light_Wall(I,I,TF,TF,TF)`	Grid of lights on a wall. TF expressions compute RGB of each light
SG	`Stained_Glass(I,I,TF,TF,TF)`	Grid of stained glass on wall
TF	Add(TF[2..4])	Add op for colour expressions. Between 2 to 4 arguments
TF	-, *, /, neg, sin, cos, log	Other math operators
TF	X, Y	Grid coordinates
TF	`ERCTF`	Ephemeral TF ($-1.0 \leq TF < 1.0$)
F	`ERCFloat`	Ephemeral float ($0.0 \leq F < 100.0$)
I	`ERCInt`	Ephemeral integer ($0 \leq I < 100$)

- *Basic_Light(...)*: Create a fixed intensity white light. A minimum distance of 0.5 metres between lights is enforced.
- *Light_Wall(I,J,R,G,B)* generates a K-by-L grid of lights. I and J are integers converted to values between $3 \leq K \leq 10$ and $3 \leq L \leq 36$. Each light's grid coordinate is accessible to the colour channel expressions.
- *Stained_Glass(I,J,R,G,B)* creates a grid of square stained glass windows on a wall. It works much like *Light_Wall(I,J,R,G,B)*. One difference is that we project each glass element's coordinate to the range $-1.0 \leq x, y \leq 1.0$. This range can be altered as desired.

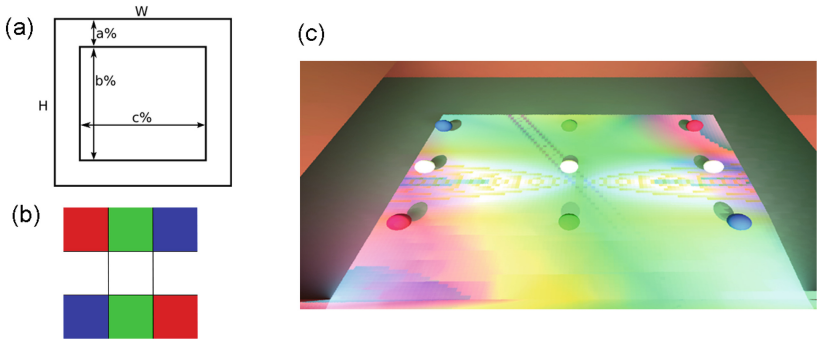


Fig. 1. (a) Window sizing parameters. (b) Colour map for wall of lights and stained glass. (c) Colour mapping in stained glass room. Spheres locations and colours show where matching done. Example uses solution in Fig. 6(a).

3.2 Fitness Evaluation

Fitness evaluation is performed on designated illumination readings of interest within the environment. These measurements are requested from Radiance to compute, at specific locations and directions. Illumination features used as fitness criteria are measured at desired locations in a room. They are as follows:

1. *Illumination intensity (or illumination)*: This is a measure of brightness. Measurements are performed in units of lux, a measurement of luminous flux per unit area. It is equal to one lumen per square metre. Lumens measure the total illumination power emitted from a light source. Lux readings are matched in the scene against corresponding target values, and the absolute difference between them is measured. Often multiple sample points are measured and averaged.
2. *Colour sampling*: Radiance's illumination model denotes light values using a high dynamic range (HDR) data type. This makes it difficult for specifying exact colours to match, as can be done with a more constrained RGB colour scheme. We therefore denote colour matching using ratios between colour channels:
$$Value1 = RedChannel \div GreenChannel$$
$$Value2 = RedChannel \div BlueChannel$$
$$Value3 = GreenChannel \div BlueChannel$$
This permits a range of colours having a similar hue, based on the relationship of channel values in a target colour. The sum of errors between the measured and target colour ratios is calculated. Low scores are preferred.
3. *Uniformity (evenness)*: Uniformity promotes gradual illumination changes. Typically some K number of illumination readings are sampled from an area. Then uniformity is:
$$Uniformity = L_{min}/L_{avg}$$

where L_{min} and L_{avg} are the minimum and average lux readings from the samples. We use the sum of 3 separate test areas for uniformity scores, giving a target value of 3.

Next, the absolute error differences between the measured values described above and the targets are placed in a feature vector. This vector denotes the multi-objective scores for the GP individual being assessed. Lower error values are better scores. Once all the population is assessed, the sum of ranks fitness score is determined for the population. This is a multi-objective scoring strategy for high-dimensional multi-objective problems [15]. Given a feature vector of size n, a population member k has a raw objective vector $(f_1^k, ..., f_n^k)$. Each objective f is separately ranked for the population, resulting in a rank vector $(r_1^k, ..., r_n^k)$. The sum of ranks score is calculated:

$$Fitness_k = \sum_{i=1}^{n} \frac{r_i^k}{max_i}$$

where max_j is the maximum rank value for objective j. The scaling by each objective's maximum rank is a normalization step that balances the contribution of each objective to the overall score.

3.3 GP Parameters

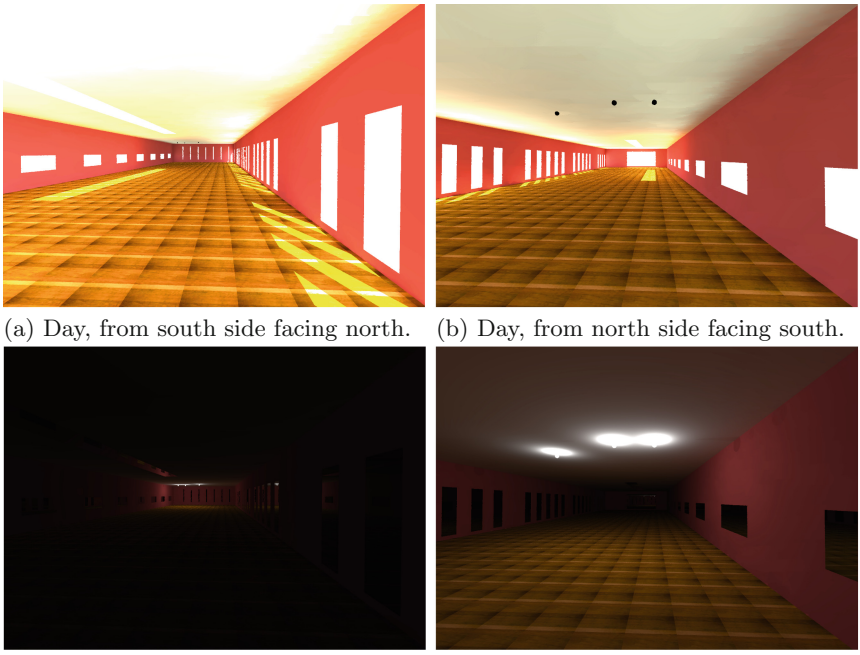
We use RobGP – a C++ based GP system [16]. It was chosen for its ease of integration with Radiance. Typical GP parameters used are shown in Table 2, and are common in the literature [1].

Table 2. GP Parameters

Parameter	Value
Runs/experiment	20
Generations	50
Population size	250
Initialization	Ramped half-and-half
Init. tree depth range	4–6
Max tree depth	11
Tournament size	3
Crossover rate	90 %
Mutation rate	10 %
Mut. grow tree depth range	2–4

Table 3. Room parameters

Parameter	Value
Room width	20 m
Room length	60 m
Ceiling height	6 m
Wall material	Plastic(0.309,0.051,0.051,0,0)
Ceiling material	Plastic(0.8,0.8,0.8,0,0)
Floor material	Radiance library wood floor
Glass definition	Glass(0.96,0.96,0.96)
Light size	0.125 m
Daylight date, time	Sept 23, 12:00EDT
Location	43.12 N, 79.25 W



(a) Day, from south side facing north. (b) Day, from north side facing south.

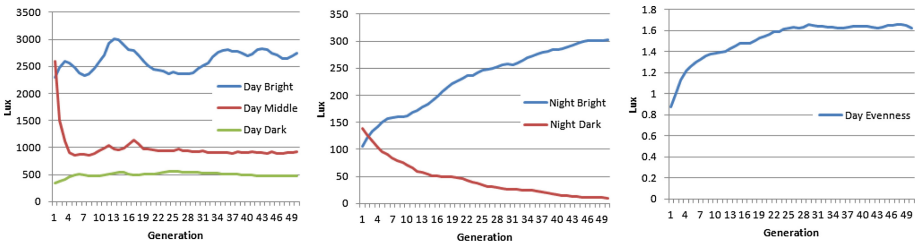
(c) Night, from south side facing north. (d) Night, from north side facing south.

Fig. 2. Day/night illumination best solution.

4 Results

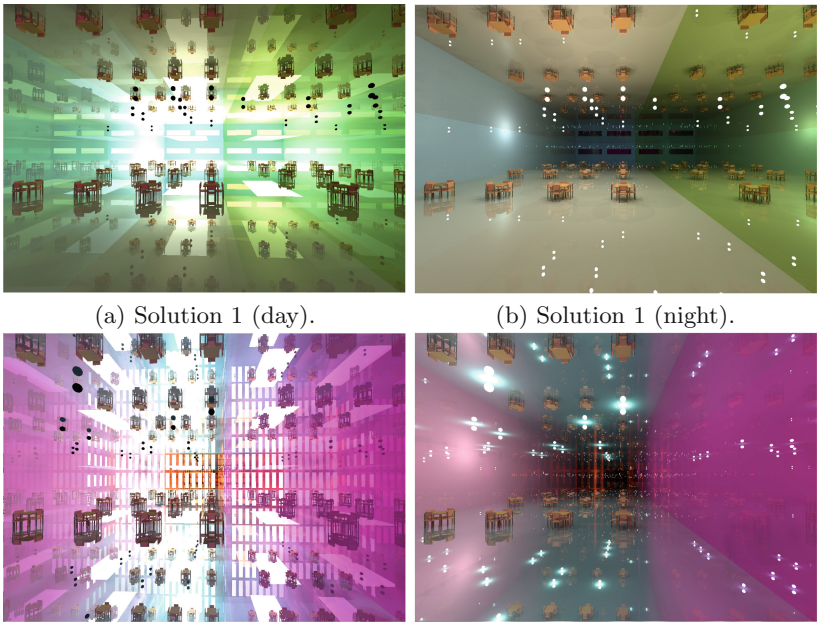
4.1 Illuminated Room: Day and Night

We consider the illumination of a rectangular room having a north-south orientation, with long walls facing east and west. Two variations of the problem are



(a) (day) #1, #3, #4. (b) (night) #2, #5. (c) (day) #6.

Fig. 3. Population fitness plots for day/night illumination (avg 20 runs). See text for target values for these objectives.



(a) Solution 1 (day). (b) Solution 1 (night).

(a) Solution 2 (day). (b) Solution 2 (night).

Fig. 4. Day/night illumination with materials evolution.

considered. In the first, we supply material definitions for the room (Table 3). The second experiment discards the predefined materials, and uses GP to evolve them. In both cases, we use GP to define artificial lights and a sky light on the ceiling, and optional windows on all 4 walls. The room is divided into 3 equal areas – north, middle, and south. Measurements are performed twice during the day – at noon, and during a moonless night. The objectives are:

#1 (day) South area having an illuminance of 4000 lux.
#2 (night) South area having an illuminance of 500 lux.

#3 (day) Middle area having an illuminance of 1000 lux.
#4, #5 (day, night) North having an illuminance of 0 lux.
#6 (day) Uniformity value of 3.

This represents a many-objective optimization problem. Illumination is sampled and averaged over 16 evenly distributed measurement locations in each section. Lights are turned off during the day.

Figure 2 shows the best scoring solution. The south sides (facing north) are made brighter by artificial lighting (day and night) and a skylight during the day. The middle and north sides are darker. The uniformity test encourages the even distributions of lights and windows. Figure 3 shows the population performance of the predefined material runs, averaged over 20 runs. Plots show the raw measurements of the factors. Most plots show a general convergence towards the desired target values. However, objective #4 (day dark) had difficulty reaching the target of 0.

Figure 4 shows two selected results from the second experiment. In both, side windows were ignored, and reflective walls were used to distribute light in the room. Whereas the predefined materials in Table 3 had no reflectivity, the evolved wall materials had reflection coefficients as high as 0.86 (a perfect mirror is 1.0). Analysis showed that material evolution resulted in statistically significant improvements in objectives #3 and 4, while the pre-defined material runs were superior in #5 and 6.[2]

4.2 Decorative Wall of Lights

The task is to define a wall of coloured lights at one side of the room. The *Light_Wall* function is used to do this, and treats light colouring as a procedural texture. The resolution of the light grid can be evolved to range between 3×3 to 10×36. We require a minimum distance of 0.5 m between neighbouring lights. A total of 9 lights in a 3×3 square pattern are selected from the whole light grid for colour sampling. The goal is to have these selected lights evolve colours that match those in the target colour map of Fig. 1(b). The colour score computes the sum of ratio errors of the 9 sampled lights and respective target colours (see Sect. 3.2). Therefore, this is a single-objective problem. Note that the lights not measured are more free to emit any colour (although overly bright lights may influence nearby colour measurements). This should result in interesting patterns of colours that are still somewhat constrained by the colour map. The integer grid coordinates of lights are used by the colour expressions.

Figure 5 shows 4 selected results from different runs. Figure 5(a) has the best score. The low resolution light grid produces different styles of images compared to high resolution bitmaps [5]. Also, direct colour matching is difficult for GP [17]. Therefore, typically 6 of the 9 colour targets might be satisfied at best.

[2] Two-tailed unpaired t-test with unequal variance, $p = 0.05\,\%$.

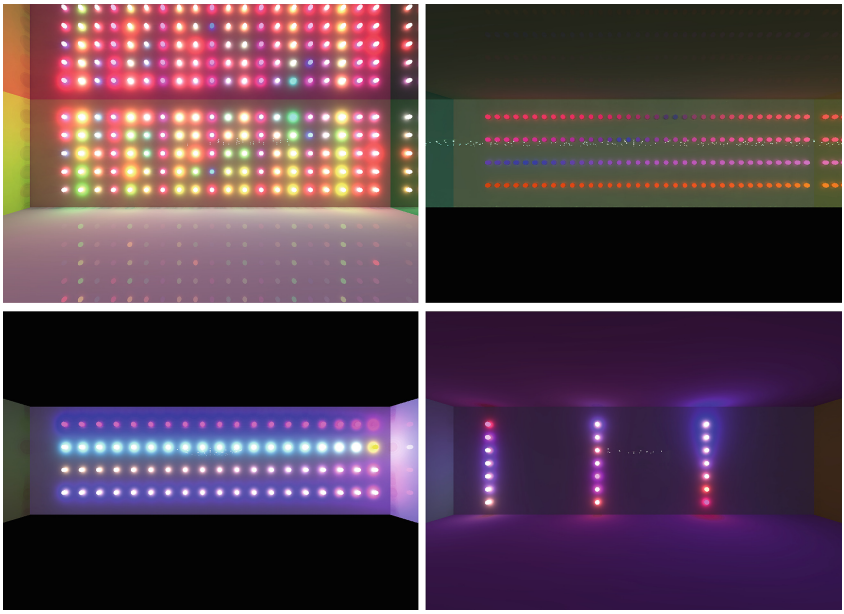


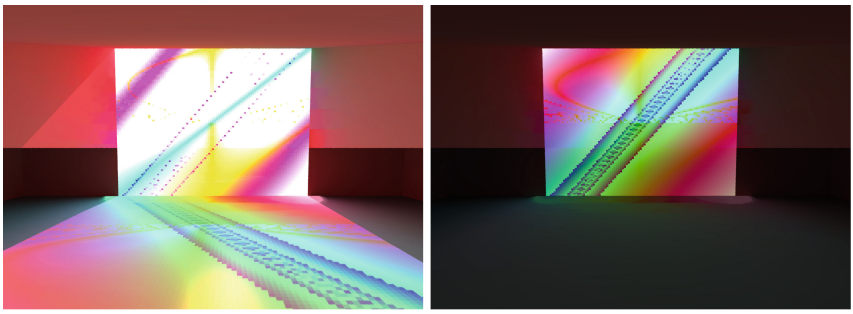
Fig. 5. Examples of light wall results.

4.3 Stained-Glass Windows

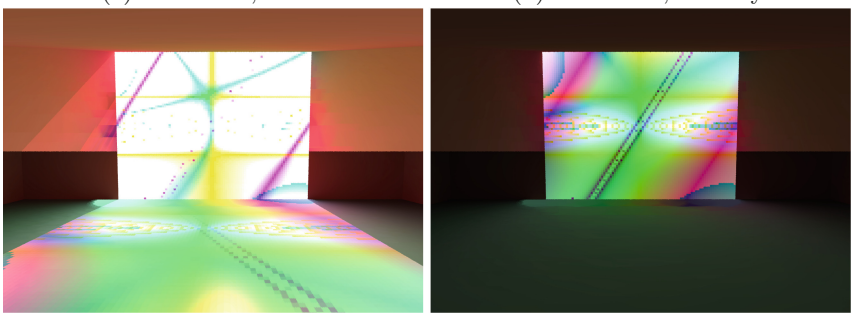
An enormous wall encompassing a stained glass window is to be created for a large public space. We use the *Stained_Glass* function to generate this windowed wall. The evolvable grid resolution resides between 50×50 to 100×100 (i.e. between 2500 and 10000 glass panels). We used a coordinate system of $-1.0 \leq x, y \leq 1.0$, with the origin (0,0) centered on the window.

To make this experiment different from the wall of lights problem, a new approach to colour analysis is used. Although somewhat contrived, the scheme is interesting and challenging, and must rely on Radiance's rendering abilities. Colour measurement is done at mid-day, when the southern Sun shining through the window maximally illuminates the interior floor. Although the window colours vary during the day as the Sun's direction changes, this mid-day illumination is used for fitness evaluation. Colour matching again uses the colour map in Fig. 1(b). We use 9 evenly-spaced sample points on the floor, from where projected window colours are sampled as they are seen by a viewer looking downward at the floor at a 45 degree direction. Thus colour is measured indirectly, and relies on the stained glass illuminating the sample positions on the floor that match the target colour map colour. The view from the window looking towards these locations is shown in Fig. 1(c). As before, a sum of colour ratio errors is used for colour scoring.

Figure 6 shows 2 interesting results. The window encompasses the entire far wall, and a vaulted ceiling slopes downwards on each side to the height indicated by the dark bands on each side of the window. Each image shows the window



(a) Solution 1, noon. (b) Solution 1, late day.



(c) Solution 2, noon. (d) Solution 2, late day.

Fig. 6. Example stained glass results.

illuminating the empty room at two times during the day. The evolved pattern is centered on the window, showing the effect of the projected texture coordinate space on the evolved colour expression [18]. Note how the window's colour changes substantially during the day.

5 Conclusion

This paper makes a number of contributions in evolutionary illumination design. The Radiance system allowed us to accurately consider a number of illumination factors. By treating illumination design as a many-objective problem, we considered up to 6 objectives. Elsewhere, we used up to 8 objectives, concluding factors such as glare. Although all fitness strategies introduce particular search biases, we feel that the sum of ranks is less prone to the immediate influences of poorly designed weighted sum formulae. It also handles higher-dimensional problems than Pareto ranking, without generating outlier solutions.

We also considered two innovative problems – a decorative wall of lights, and a stained-glass window. The success of these problems was due to GP's well known proficiency in evolving procedural textures. In both cases, rather than compute pixel colours on a bitmap, we defined the colour of lights and stained glass. Being able to apply procedural textures to these alternative problems was due to Radiance's illumination abilities.

There are many directions for future work. One obvious extension is to include energy-efficiency considerations, as done by Caldas [10], Marin *et al.* [12] and others. Although we considered the illumination of pre-existing structures, it is possible to integrate floorplan and 3D model design with illumination. We have only scratched the surface of using procedural textures for decorative illumination. By including texture operators that use noise and entropy [18], and employing aesthetic fitness evaluation techniques [19], many exciting results in illumination design are waiting to be discovered.

References

1. Koza, J.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge (1992)
2. Larson, G.W., Shakespeare, R., Ehrlich, C., Mardaljevic, J., Phillips, E., Apian-Bennewitz, P.: Rendering with Radiance: The Art and Science of Lighting Visualization. Morgan Kaufmann, San Francisco (1998)
3. Fuller, D., McNeil, A.: Radiance, http://www.radiance-online.org/
4. Sims, K.: Interactive evolution of equations for procedural models. Vis. Comput. **9**, 466–476 (1993)
5. Graf, J., Banzhaf, W.: Interactive Evolution of Images. In: Proceedings of International Conference on Evolutionary Programming, pp. 53–65 (1995)
6. Russell, S.: The Architecture of Light: Architectural Lighting Design Concepts and Techniques. Conceptnine, San Diego (2008)
7. Tena, J.E., Rudomin, I., Eugenio, A., Sada, G., Gordo, C.: An interactive system for solving inverse illumination problems using genetic algorithms. In: Proceedings of Computación Visual (1997)
8. Fernández, E., Besuievsky, G.: Inverse lighting design for interior buildings integrating natural and artificial sources. Comput. Graph. **36**, 1096–1108 (2012)
9. Castro, F., del Acebo, E., Sbert, M.: Energy-saving light positioning using heuristic search. Eng. Appl. Artif. Intell. **25**(3), 566–582 (2012)
10. Caldas, L.: Generation of energy-efficient architecture solutions applying gene_arch: an evolution-based generative design system. Adv. Eng. Inform. **22**(1), 59–70 (2008)
11. Watanabe, M.S.: Induction Design: A Method for Evolutionary Design. Birkhauser, Basel (2002)
12. Marin, P., Bignon, J.C., Lequay, H.: Generative exploration of architectural envelope responding to solar passive qualities. In: Design & Decision Support Systems in Architecture and Urban Planning. Eindhoven U of Tech (2008)
13. Montana, D.: Strongly typed genetic programming. Evol. Comput. **3**(2), 199–230 (1995)
14. McKay, R., Hoai, N., Whigham, P., Shan, Y., O'Neill, M.: Grammar-based genetic programming: a survey. GPEM **11**, 365–396 (2010)
15. Corne, D., Knowles, J.: Techniques for highly multiobjective optimisation: some nondominated points are better than others. In: Proceedings of GECCO 2007, pp. 773–780. ACM Press (2007)
16. Flack, R.: Robgp - robust object based genetic programming system, September 2009. http://sourceforge.net/projects/robgp/
17. Wiens, A., Ross, B.: Gentropy: evolutionary 2D texture generation. Comput. Graph. J. **26**(1), 75–88 (2002)

18. Ebert, D., Musgrave, F., Peachey, D., Perlin, K., Worley, S.: Texturing and Modeling: A Procedural Approach, 2nd edn. Academic Press, New York (1998)
19. den Heijer, E., Eiben, A.: Comparing aesthetic measures for evolutionary art. Applications of Evolutionary Computation. LNCS, vol. 6025, pp. 311–320. Springer, Heidelberg (2010)