Firefly: Virtual Illumination Drones for Interactive Visualization

Sergej Stoppel, Magnus Paulson Erga, and Stefan Bruckner, *Member, IEEE Computer Society*

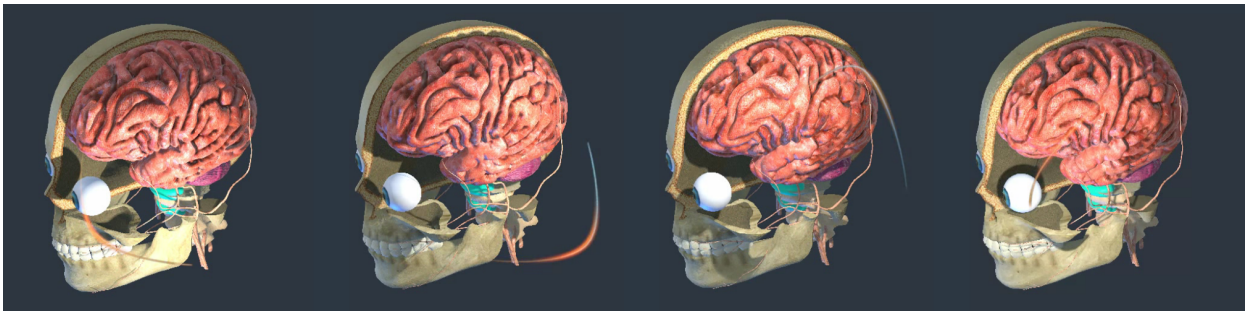

Fig. 1. Lighting designs with static light sources aim to emphasize properties of the illuminated object such as the average surface variation or curvature. Sophisticated as those approaches can be, they can never account for all local properties of the illuminated object. We propose animated lights, or Fireflies, to solve this challenge by moving the light on a path that emphasizes the local properties over the time. In this image we show four consecutive positions of a Firefly designed to emphasize the brain.

Abstract—Light specification in three dimensional scenes is a complex problem and several approaches have been presented that aim to automate this process. However, there are many scenarios where a static light setup is insufficient, as the scene content and camera position may change. Simultaneous manual control over the camera and light position imposes a high cognitive load on the user. To address this challenge, we introduce a novel approach for automatic scene illumination with Fireflies. Fireflies are intelligent virtual light drones that illuminate the scene by traveling on a closed path. The Firefly path automatically adapts to changes in the scene based on an outcome-oriented energy function. To achieve interactive performance, we employ a parallel rendering pipeline for the light path evaluations. We provide a catalog of energy functions for various application scenarios and discuss the applicability of our method on several examples.

Index Terms—Dynamic lighting design, lighting drones

1 Introduction

Illumination has a crucial impact on the appearance of 3D objects and shape perception in computer-generated scenes. Once the geometry, textures, and material properties of the scene have been defined, its appearance is greatly affected by the illumination setup. Shape perception, for example, is highly dependent on light placement. Uncommon positioning of light sources can distort the perceived geometry as it is known from crater or dome illusions. The traditional approach for lighting specification is an iterative process of trial and error, where the user continuously adjusts the light parameters and evaluates the rendered image. This makes lighting design a challenging task even for static lights.

Moreover, in interactive scenarios static lights alone may not be sufficient. When asked to visually assess the geometry of objects, observers most commonly rotate them back and forth. Studies show that the motion of an object relative to the light source helps to evaluate its geometry and material properties [10, 40]. When the object of interest cannot be moved easily, observers tend to obtain geometry and material cues by moving the light source. It has been shown that participants have a clear intuition of how the light source distance and position affect the shading patterns for a variety of different surfaces [35]. This indicates that animated lights could be used to enhance shape perception in 3D scenes.

Modern animation approaches use moving light sources to emphasize temporal changes in the scene or to adapt the light conditions to a dynamic setup. While static lighting specification is already a challenging task with some existing solutions for automated support, the simultaneous control of the camera and moving light sources represents an even greater hurdle for most users. Furthermore, when considering the rapid advances in immersive virtual reality technology, which limit the available degrees of freedom for interacting with parameters like light source configurations, the requirement for "intelligent" light sources that automatically adjust to the movement of the camera becomes more and more pressing. To address this challenge, we propose a novel automated approach for scene illumination with dynamic light sources that offers additional perceptual cues compared to static lights. In addition to the static light sources in the scene, we generate a Firefly – a moving light drone that illuminates the scene while flying on an automatically generated path. Our system continuously optimizes the Firefly path based on a flexible energy function, tailored for various visualization tasks. We propose several energy function constructions for different visualization scenarios, that are designed to enhance different aspects of the rendered objects, while following established lighting design rules from photography and art.

2 Related Work

Automation and optimization techniques have a long tradition in the field of visualization. A common problem is the choice of suitable parameters for a particular visualization technique and/or dataset. In volume rendering, for instance, the specification of transfer functions is a challenging task. While presets can provide some additional support to the user, the presence of additional variables such as differences in

- *Sergej Stoppel is with the University of Bergen, Norway, E-mail: sergejsto@gmail.com*
- *Magnus Paulson Erga is with the University of Bergen, Norway, E-mail: magnus.p.erga@gmail.com*
- *Stefan Bruckner is with the University of Bergen, Norway, E-mail: stefan.bruckner@uib.no*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

the data acquisition necessitates more advanced approaches. Examples include the work of Ruiz et al. [34], who presented a framework to define transfer functions based on a target distribution provided by the user. Similarly, Borga et al. [6] proposed an optimization based algorithm that shifts preset transfer functions, to account for general deviations and local variations in the data. To deal with occlusion in flow visualization, Günther et al. [16] translate the occlusion problem into a view-dependent global optimization problem that is solved with the least squares method. Modern visualization solutions for 3D scenes often employ large and complex scene geometries. Early on it was found that the increased scene complexity hindered the orientation of the user and impaired the ability to navigate effectively. Freitag et al. [13] automatically adjusted the camera speed based on viewpoint quality to reduce the cognitive effort for camera control in indoor scenes. Xie et al. [45] proposed an automatic camera path planing method, that aims to improve the user's sense of direction in VR setups. The roles of static, animated, and interactive presentations of 3D scenes have been investigated by Froes et al. [14]. Coffey et al. [7] extended the study to virtual reality. Our approach can be seen as a natural development of expanding animations to light setups.

As our approach aims to generate virtual drones to automatically support the user in their task, we draw inspiration from physical autonomous vehicles. Already in 1969, Keiser and Peebles [21] discussed a concept for automatic drone control. Nikolos et al. [29] used an evolutionary approach to design a Bezier path for unmanned aerial vehicles. While their setup significantly differs from ours, Srikanth et al. [38] used physical light drones for rim illumination of dynamic objects in indoor photography. Joubert et al. [19] presented an interactive path planing tool for drone cinematography, particularly focusing on the importance of the trajectory smoothness and the spatial awareness of the user.

Illumination has a significant impact on the perception of a scene, but can necessitate a tedious trial and error process in order to arrive at a desired result. Cost et al. [8] discussed an automated approach for lighting design that employed optimization strategies based on object geometry, material properties, and design goals. To support non-experts, Shacked et al. [36] developed a method for fully automatic lighting design based on a perceptual quality metric. Gumhold [15] used entropy to place a light source that maximizes the information added by illumination. Halle et al. [17] presented LightKit, a lighting system for 3D scenes inspired by light designs of artists and photographers. Lee et al. [26] introduced Light Collages, an illumination system that enhances local features using a globally inconsistent lighting setup. Wang et al. [43] proposed a lighting system that enhances visual cues for local and global features. Zhang and Ma [47] extended automatic three-point lighting setup to volume rendering employing global illumination. Recent work by Wambecke et al. [42] introduced a lighting design approach based on photographic rules, taking into account the shapes and materials of the objects. A coherent lighting setup is especially important in augmented reality, as the rendered object must be integrated into an already illuminated scene. Haller et al. [18] used real-time shadow maps to add realism to the augmented scene. Okumura et al. [30] and Klein et al. [25] incorporated blurring filters on the rendered image to match the depth of field of the captured video stream. Aittala [5] used real-world observations from a diffuse sphere to adjust virtual lighting parameters for augmented reality setups.

To produce an effective lighting setup, it is important to account for various aspects of human perception. Studies by Ramachandran [31], Kleffner and Ramachandran [24], and Mamassian et al. [27] investigated the assumptions made by the human visual system and how they are affected by the light position. Doerschner et al. [10] identified three motion cues the visual system relies on to distinguish between matte and shiny surfaces. Kersten et al. [23] performed a study on information provided by cast shadow motion. Their research suggests that the visual system assumes a stationary light source even if a moving light source is present. However, this may be due to the fact that no visual cues for the light position where presented to the participants. A recent study by Schütt et al. [35], where the participants could control the light position to some degree, suggests that participants do have

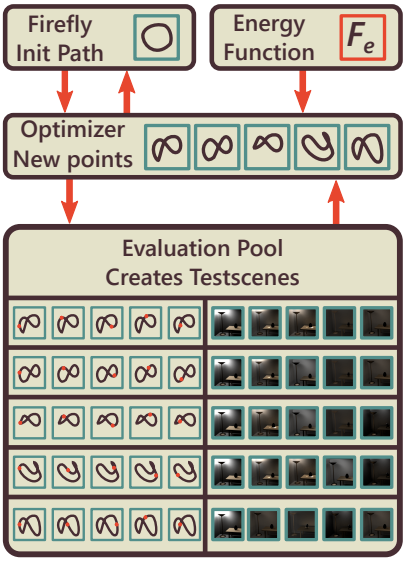

Fig. 2. A conceptual overview of the Firefly system. The system operates in a parallel multi-threaded manner. The only input required by the user is a selection of the energy function from a provided library. The system iteratively updates the Firefly path by creating a set of new paths and evaluating them in the background (Evaluation Pool). The Firefly transitions to a better path when one is found.

a clear intuition of how light positions affect the appearance of the illuminated object. Guided by these findings, we inform the user of the Firefly position through visual cues and avoid sudden and unexpected trajectory changes by imposing constraints on the path shape.

In a cinematic context, lighting is often used to convey emotions, and the effects of lighting on the perceived scene atmosphere have been investigated in several studies. An overview of different lighting setups can be found in the book Advanced RenderMan by Apodaca and Gritz [4]. De Melo et al. [9] focused on the emotions induced by different lighting setups and proposes a model for the expression of emotions in virtual humans with a composition of lights, shadows, and chromatic filters. Wisessing et al. [44] investigated how animated characters are perceived when viewed under different lighting conditions. Nasr et al. [11] presented a lighting system that automatically adjusts to accommodate variations of the dramatic scene characteristics. We use the findings of these studies as guidelines in the construction of the Firefly paths.

3 Firefly

The goal of our approach is to support users by providing an animated light source – a Firefly – that moves along an adaptive path continuously adjusting its trajectory if necessary. The Firefly complements additional static light sources and in particular aims to enhance dynamic aspects of the exploration process. As the control and planning of a moving light source is even more complex than the design of a static lighting environment, automatic generation and adaptation of the Firefly path is a key component of our system.

Whereas previous approaches for the placement of static lights could partially rely on precomputation, our aim is to provide a fully dynamic solution that adapts interactively to changes in the camera and scene setup, and does not impose any constraints on the content of the scene. A key aspect of our approach is that the FireFly acts in a view-dependent manner, i.e., it aims to adjust its trajectory according to what the user sees. As such, our approach is an online optimization process. We designed Firefly as an independent component that can be easily integrated into existing systems. To achieve a high degree of flexibility for versatile illumination tasks, we created the Firefly system as a plugin-based detached optimization process. We explain the general Firefly system in the remainder of this section, discussing the individual components in detail in the following subsections.

A general overview of our approach can be seen in Figure 2. The

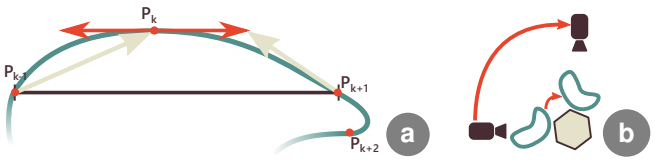

Fig. 3. (a) To reduce the complexity during the optimization process we compute the tangents at point P_k automatically from the neighboring points P_{k-1} and P_{k+1}. (b) Assuming that small changes of the camera position introduce only small changes to the scene, we transform the Firefly path with the camera around the object of interest to provide an initial path.

Firefly generation and optimization works as a parallel process to the rendering pipeline. The only input required from the user is the selection of an energy function from a catalog. When a Firefly is triggered during the interaction process, an initial Firefly path is constructed and placed in similar positions as lights in common photography setups. Next, the optimizer suggests a new set of light paths. The light paths are sent to an evaluation pool, which handles every test path in a separate thread. In each thread a new scene with the Firefly path is generated and the path is evaluated according to the energy function. After all paths have been processed, the evaluation pool returns a fitness value for each path to the optimizer. When the optimizer has found a better path than the one that is currently displayed, the Firefly transitions to the new best light path, without interrupting the optimization process. If the user moves the camera during the optimization, the existing Firefly path is transformed accordingly. To provide the user with visual ques on the position change of the Firefly, we display a tail behind the moving light source. The user is able to remove or adjust the tail through a simple slider. During the development of the Firefly, we found that users preferred a Firefly with a tail as reference.

3.1 Path Generation and Adaptation

Before discussing the optimization process of the Firefly path, we briefly describe the generation of the initial Firefly path and the adaptation to the camera movement. The Firefly path is defined by a set of control points as a smooth, closed, three dimensional curve. While our system is not limited to a specific path formulation, we implement the Firefly path as collection of cubic Bezier curves, with a C^1 transition at the endpoints. To reduce the complexity during the optimization, we automatically compute the tangent vectors for the Bezier curve and use the segment endpoints as optimization parameters.

Because we designed the Firefly system with the goal of high flexibility, we generate the initial Firefly path as a simple shape that is able to adapt very fast, i.e., as a circular path between the camera and the object of interest. The simple circular shape has several advantages, as it has the lowest curvature of all possible shapes in the same space, thus avoiding sudden trajectory changes at the initialization. Furthermore, the circular shape naturally provides well distributed initial control points for the optimization process, as the control points are evenly spaced and cover a relatively large portion of the design space. Thus, the circular shape provides a fast convergence rate at the beginning of the optimization process. A common approximation for a circle with n Bezier curves is to construct the tangents at a point p_k with the length l as:

$$l = \frac{4}{3} \cdot \tan\left(\frac{\pi}{2n}\right), \tag{1}$$

and the same direction as the circle tangent. To keep the Firefly path consistent, we keep the initial ratio of the tangent length and the distance between neighboring points fixed. Having saved the initial tangent length l and the initial distance d between two points p_{k-1} and p_{k+1}, we can compute the new tangent t_k at the point p_k as:

$$t_k = (p_{k+1} - p_{k-1}) \cdot \frac{l}{d} \tag{2}$$

We illustrate the automatic tangent computation in Figure 3(a). The initial placement of the path is inspired by photography rules discussed in Section 3.5 as an elevated key light.

As the control points are updated during the optimization, the Firefly transitions from the old to the updated path on a linearly interpolated trajectory. The intermediate position of the Firefly is then computed as:

$$P_{trans} = P_{new} \cdot \frac{t}{w} + P_{old} \cdot \frac{w-t}{w}, \tag{3}$$

where w is a time window for the interpolation and t is the time that has passed since the interpolation start. In our implementation, we use a window size w of three seconds. As the sum of two smooth functions is smooth as well, the transition occurs without undesired jumps of the Firefly. When the user changes the camera position, the Firefly path needs to adapt to the changes in the visible scene. A reasonable assumption is that small changes to the camera transformation result in small changes to the visible scene and thus the path energy. Therefore, to create a well suited initial condition for the path, we rotate the Firefly path together with the camera position around the object of interest. We illustrate this transformation in Figure 3(b).

3.2 Energy Function

We aim for a flexible and easily-adjustable lighting system that automatically creates a well-suited light path for various application scenarios. To achieve this goal, we draw inspiration from approaches as active contour models [20], originally developed in the context of image segmentation, where the outline of an object is modeled as an energy-minimizing deformable curve using a combination of energy terms for the contour shape and image properties. This provides a high degree of flexibility in modeling specific application requirements using different variations of the individual energy terms. Inspired by this concept, we define the Firefly path through a plugin-based energy function. The optimization of the Firefly path can be formulated as a minimization problem of the used energy function. As such, the choice of the energy function directly determines the shape and the evolution of the Firefly path. In this section, we discuss the components of an energy function for light path optimization on a conceptual level, while we introduce detailed formulations of energy functions for specific application scenarios in Section 5.

Previous approaches for lighting design are guided by rules taken from photography or image properties such as entropy. In addition, our approach needs to take the path shape into consideration as well. For instance, drastic and unexpected changes of the light trajectory can be confusing for the user [23], as they might be misinterpreted as object movement in the scene. Hence, a general energy function consists of two components, one component for the rendered scene and one component for the path properties. A general energy function for light path optimization can be constructed as follows:

$$E = \alpha \cdot E_I + (1-\alpha) \cdot E_P \tag{4}$$

where E_I is the image energy, E_P the path energy, and α is a weight to control the influence of the energy components. The weight α essentially controls the degree of directional change of the Firefly path. A value of $\alpha = 1$ discards the path shape completely and can produce very sharp trajectory changes. On the other hand a value of $\alpha = 0$ results in a closed curve with the least trajectory change, i.e., a large circle. Because we ensure C^1 continuity of the path, we can choose a relatively high α without causing excessively sharp trajectory changes. While this weight can differ for various application scenarios, we used $\alpha = 0.95$ for all examples in this paper. In Figure 4, we illustrate the light trajectories resulting from different values for the weight α. In the remainder of this section, we discuss the image and path energy in detail.

Image Energy: The image energy is the most versatile and important component of the energy function. With image energy, we denote any energy formulation that can be derived directly from the rendered scene. The formulation of this component indirectly determines how

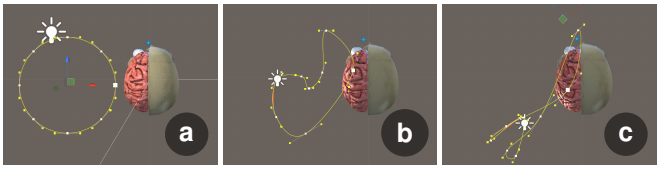

Fig. 4. The influence of α on the light trajectory shape. The weight is set to 0, 0.95 and 0.999 in (a), (b), and (c), respectively. Using $\alpha = 0$ results in a circular trajectory (a), whereas neglecting the path shape too much can result in sharp trajectory turns (c). An α of 0.95 (b) results in a light path that is able to accommodate the image energy without excessively sharp turns.

the light path will illuminate the scene. As there are countless possibilities to define a meaningful image energy, we illustrate only a handful formulations and their effects on the resulting light path.

As we are interested in how the light path affects the scene as seen from the current camera position, we only consider illumination contributions that are visible from the current camera's point of view. As mentioned before, we formulate the path computation as a minimization problem of the energy function, and therefore the light path must be aggregated into a single energy value. To achieve this, the illumination evaluation must be aggregated over the rendered image as well as over the path. Therefore, we define the image energy as two nested functions. Using the notation I for image and P for the path domain, we define the energy function as:

$$E_I = F_P\left(F_I(sc)\right) \text{ or:} \tag{5}$$
$$E_I = F_I\left(F_P(sc)\right) \tag{6}$$

As the respective functions are not necessarily commutative, the order of the evaluation can have critical impact on the resulting energy function and should be chosen according to the application scenario.

The purpose of a light is the illumination of the scene. Therefore the image energy must account for the scene illumination explicitly or implicitly. A straight forward measure of the scene illumination is the measurement of the brightness of the rendered object. A useful target for a Firefly would be the illumination of the scene with a desired intensity. We can formulate such behavior with the following energy function:

$$E_I = max_P\left(avg_I(br(sc) - \gamma)\right) \tag{7}$$

where γ is the target brightness. Such a formulation forces all light positions on the path to be close to the desired brightness on average. However, the energy does not penalize an image with too dark and too bright regions, as long it does not change the average brightness of the scene. Changing the order of the functions yields the following energy function:

$$E_I = avg_I\left(max_P(br_{x,y} - \gamma)\right). \tag{8}$$

This energy function first creates a new image with the maximal difference along the path for each pixel and then computes the average of the image. This means that we first aggregate over time and evaluate the influence of the light path locally. Therefore, this formulation penalizes paths that produce surface illuminations differing strongly from the desired brightness anywhere on the illuminated object, hence enforcing a uniformly illuminated object for the whole Firefly path.

In Figure 5, we illustrate the effects of these two energy functions on the Firefly path shape and the illuminated scene. For this example we have chosen γ as 0.3. The first row of Figure 5 shows the paths of the Firefly. In Figure 5(a) we show the initial Firefly path, in (b) and (c) we depict a Firefly path after 30 iterations with the image energy defined in Equation 7 and Equation 8, respectively. The second row shows a captured scene that illustrates the differences in the energy functions. The initial path comes close to the object, creating an overly bright appearance in Figure 5(d). Using the $max_P(avg_I)$ energy function

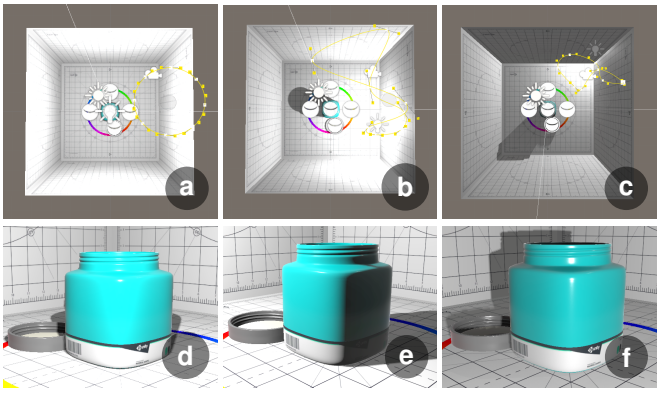

Fig. 5. An illustration of path shapes with varying energy functions. The first row shows the paths seen from above while the second row presents a scene state typical for the energy function. (a) and (d) show the path and a scene for the initial path. The object is too bright for some positions of the Firefly. (b) and (e) show the resulting path for the energy function described in Equation 7. On this path the Firefly can take positions that create strong shadows in the scene. In (c) and (f) the path was changed according to the energy function in Equation 8. The scene is much more evenly illuminated compared to (e).

Fig. 6. We allow the user to select objects of interest. Here, the user selected the paint jar in (a) with a object mask shown in (b). If the user wants to further specify a local region of interest, they can use an input mask. The combined object and input mask are shown in (c).

enforces a greater distance between the object and the light, but it can create strong shadows, as shown in Figure 5(e). Using $avg_I(max_P)$ results in a scene as in Figure 5(f). Clearly, the surface is much more evenly illuminated compared to the one in Figure 5(e). We want to point out that we have chosen these energy functions to demonstrate the importance of the function order and not necessarily as best suited for certain illumination scenarios. We introduce energy functions tailored to specific application scenarios in Section 5.

Lighting setups are commonly centered around an object of interest. However, the user may be interested in multiple objects in the scene or might want to change the object of interest. To address this, we allow the user to select objects of interest in the scene though a simple click on the object. When an object is selected, the system generates an object mask M_{obj} to evaluate only the pixels covered by the mask. We illustrate such a mask in Figure 6(b), where the ink container in Figure 6(a) is selected as the object of interest resulting in a mask as shown in Figure 6(b), which discards the background completely. If the user is only interested in parts of the image, they can add an additional input mask M_{in} to emphasize these regions. The input mask is defined as a smooth function between 1 and 0, that decreases with the distance to the mouse position. A combination of the object mask and input mask can be seen in Figure 6(c).

Path Energy: The visual system is very sensitive to changes in the lighting conditions. Sudden and unexpected changes of the light trajectory can lead to misinterpretation of the scene dynamics. As indicated by Kersten et al. [23], users can misinterpret unexpected changes of light conditions as movements in the scene. The likelihood of such misinterpretations can be reduced using two strategies: by employing a smooth light trajectory without sharp turns and by giving the user explicit feedback on the light positions. The path energy ensures the former. The degree of how drastically a path is changing

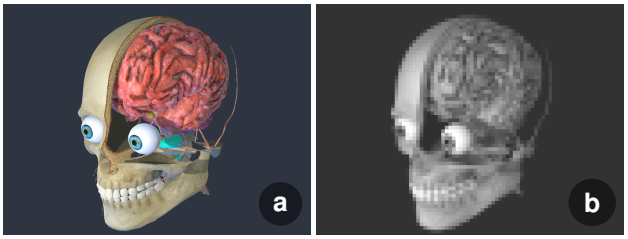

Fig. 7. The brightness values of the rendered image in (a) are partitioned into 64×64 segments resulting in a down-sampling shown in (b). This partitioning provides a good trade-off between the preserved level of detail and data reduction.

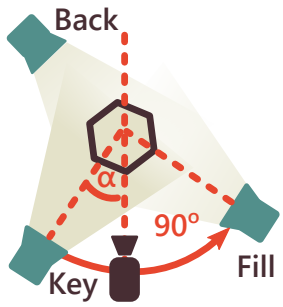

Fig. 8. Firefly can be easily integrated into existing lighting setups. In this Figure, a Firefly is integrated into a three-point lighting setup as the key light at an angle α of 30°. A fill light is positioned at 90° to the key light. The back light is placed behind the illuminated object with respect to the camera position.

can be measured through the curvature of the path. To avoid sharp turns, the path energy is defined as the maximal curvature $\kappa(s)$ over the path P:

$$E_P = \max(\kappa(s)), \quad s \in P. \tag{9}$$

This simple restriction in addition to the C^1 continuous path formulation ensures a smooth trajectory of the Firefly without excessively sharp turns.

3.3 Optimization

Having specified the desired energy function, a well-suited path for the Firefly is one that minimizes the total energy. Choosing a suitable optimization approach is crucial for the quality of the Firefly path. As mentioned in Section 3.1, the Firefly path is defined by a set of control points. Thus, the dimensionality of the parameter space is equivalent to the number of control points. Hence, there is a trade-off between the problem complexity and the granularity of the direct path shape specification. Because the Firefly is created as a supportive tool during user interaction, the optimization must provide adequate results on the fly. In our examples, we construct the Firefly path with eight control points, which provides enough control for the Firefly path definition, while constraining the optimization to a reasonable degree of complexity that allows us to maintain interactivity.

As the energy functions are dependent on the scene composition, they can be highly non-convex, resulting in many local minima. This imposes a challenge for many gradient descent algorithms. Even modern methods such as Adagrad, AdaDelta, RMSprop, and NADAM [33] can get stuck in local minima. To overcome local minima, we implemented an adapted version of the Simulated Annealing (SA) algorithm [41]. Traditional SA considers one neighboring state $\tilde{S}$ of the current state S, and decides whether to update the system state to $\tilde{S}$, based on a temperature-dependent probability function. In each iteration, the system temperature is decreased, stabilizing the energy state, until the computational budget has been used up. In our approach, we always evaluate an ensemble of neighbors at once, thus increasing the convergence speed of the algorithm.

A straight-forward adoption of SA to ensemble sampling is a Monte Carlo sampling of the ensemble over the parameter subspace. However, several publications [22, 37, 46] show an improved convergence rate of Latin hypercube sampling over Monte Carlo, due to its improved space filling properties. Therefore, in each iteration, we perform a Latin hypercube sampling, computed using the method of Stein [39]. We assume a normal parameter distribution and no parameter correlation for the sampling process. In each iteration, we use the best state of the ensemble for the update decision.

3.4 Sample Evaluation

As mentioned previously, computation speed is crucial for interactive scenarios. A common bottleneck for light design approaches is the evaluation of the scene. This is even more true for our approach, as the scene needs to be evaluated not just for different light positions but for different light paths. To approach this challenge, we can essentially employ two strategies: increase the computational performance of our approach and reduce the computational complexity of the problem.

To increase the performance of our approach, sampling and evaluation of the scene are performed in a detached parallel rendering pipeline. We employ a worker thread pool to generate the scenes and collect their information for each a test path. Optimization techniques require an aggregation of the fitness to a numeric value. For the Firefly path, the fitness is computed through the energy function. As mentioned in Section 3.2, the order of the energy function components affects the order of the aggregation over the image domain and the path domain. If the aggregation is performed over the image domain first, then the worker can return a single numerical value for each scene. However, if the aggregation is performed over the path domain first, then the worker would need to store the scene information for every light sample on the path before performing the aggregation.

Clearly, this creates a substantial overhead for the worker performance. To address this challenge, we need to find a suitable trade-off between computational efficiency and the preservation of features of the energy function. Usually, light affects the scene over neighborhoods instead of isolated points. Therefore it is reasonable to assume that neighboring pixels will have similar energy characteristics. We use this fact to abstract the localized energy states in the scene through partitions that store the average energy of their pixels. We illustrate the partition of the illumination energy in Figure 7. In our current implementation, we divide the scene image into 64 by 64 partitions, which still captures enough details. For each path, the workers evaluate the partitions first over the path domain and then over the image domain according to the energy function, and then report the resulting image energy as a single value.

3.5 Lighting Design

The three-point lighting setup is a common lighting method used in photography, cinematography, and computer-generated imagery. It is a relatively simple but versatile approach which forms the basis for most lighting setups. In the following, we briefly discuss the integration of Firefly into a basic version of three-point lighting as described in several photography text books [12, 32]. As the name suggests, three-point lighting uses three light sources, a *key light*, a *fill light*, and a *back light*, as shown in Figure 8.

The key light is the main and usually the strongest light of the setup. The goal of the key light is to produce tonal variations in the image. Therefore, it is usually placed to one side of the illuminated object in order to produce shadows visible to the camera, as illustrated in Figure 8. A strong key light can create very strong shadows and thus hide geometric details. To overcome this problem, a secondary light is used to "fill" out the shadows with a soft light. The fill light is often placed at 90° to the key light (see Figure 8). The fill light is usually less bright and softer than the key light, playing only a secondary role for the illumination. The back light is placed behind the illuminated object. Instead of providing direct illumination of the object, the back light

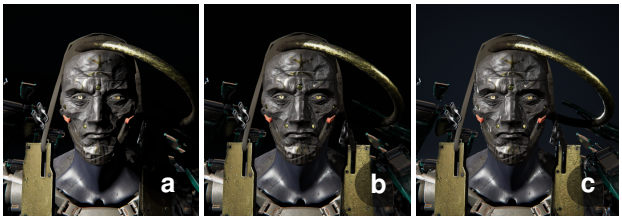

Fig. 9. A comparison between (a) a single key light, (b) a combination of key and fill light, and (c) a three-point lighting setup.

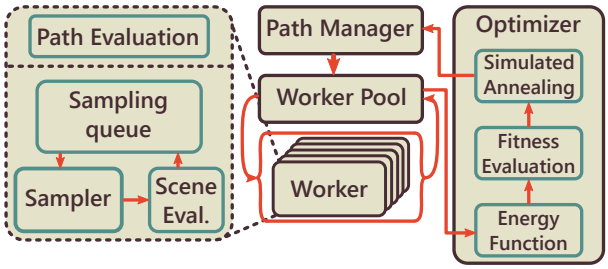

Fig. 10. The Firefly system is implemented as a background process running parallel to the main thread. To achieve on-the-fly adaptation of the Firefly path, we evaluate 40 paths at once in a worker pool. When the paths are evaluated, the information is sent to the optimizer that computes the path energy and requests further samples.

emphasizes and provides subtle highlights of the object's silhouette. This helps to separate the object from the background and provides additional depth cues.

We integrate our Firefly into a three-point light setup by considering it as the key light. A common placement of the key light is at an angle α of 30° to 45° at the side of the camera and the illuminated object, as shown in Figure 8. Furthermore, the key light is commonly placed slightly above the camera. We initialize the Firefly path position at an α value of 30° and a relative elevation above the camera equal to half of the path radius. The fill light moves with the key light and is placed at an angle of 90° to the key light rotated on a plane defined by the right and front camera vectors. The back light is placed behind the object in the view space. However, this setup should be considered as guideline rather than a rule, as the Firefly path converges quickly even when poorly initialized.

In Figure 9, we illustrate the effect of the number of lights on a model from the animated short film Adam [1]. In Figure 9(a), only the key light is used and in (b) a fill light was added to the scene creating a more even illumination. In Figure 9(c), the addition of a back light provides additional subtle highlights of the geometry.

4 Implementation

We implemented the Firefly system as a camera component in Unity [3]. Unity is a popular multi-purpose engine that supports 2D and 3D graphics. The drag-and-drop functionality of Unity and the C# scripting interface provide fast prototyping possibilities. Furthermore, the efficient plugin system allows for easy sharing of the results across multiple platforms. To use the system, the user simply adds Firefly as a camera component to the scene setup. For selecting the object of interest, Firefly only requires that the 3D objects have a collision geometry and a unique name.

Firefly is implemented as a background process, running in parallel to the main rendering thread, as illustrated in Figure 10. When an object of interest is selected, a new Firefly path is initialized and the optimizer creates a list with control point offsets that is sent to the path manager. The path manager creates the light paths that are evaluated in a worker thread pool. The workers create a sampling queue of scenes that are rendered and evaluated in a process that is fully transparent to the user. The primary benefit of a worker thread pool over creating a new thread for each path evaluation is that the thread creation and destruction overhead is restricted to the pool creation. The size of the worker thread pool depends on the used hardware. We found that on an Intel Core i7 3.00 GHz CPU, a worker thread pool of size five delivered the best performance.

Each worker evaluates one path at a time by computing its image and path energy. The image energy is computed on the GPU by rendering the scene for uniformly-sampled Firefly positions along the path. Thus, the number of scene evaluations directly corresponds to the evaluation time for the image energy. During the development of the Firefly system, we found that evaluating 14 samples for each path provides a robust estimate of the path quality while still allowing for a fast computation. When a worker has finished the evaluation of a path, the information of this path is stored in the worker pool. When all paths have been processed, the worker pool sends the results of all paths to the optimizer.

In the optimizer, the collected information is used to compute the final energy function. Next, the optimizer evaluates the fitness of the paths and possibly updates the Firefly path to a better one. The last step of the iteration is the computation of a set of new sampling parameters that are sent to the path manager. On an Nvidia GeForce GTX 780 GPU and an Intel Core i7 3.00 GHz CPU and a screen resolution of 1920×1200, one such iteration requires on average 973 ms for the evaluation of 40 paths with 14 rendered images for each path. The main thread is virtually unaffected by this computation and we did not detect a noticeable drop of the frame rate for the rendered scene.

5 Results

In the following, we demonstrate the use of Firefly for four different scenarios. To show the versatility of our approach, we selected examples covering scientific visualization applications as well as scenarios inspired by applications in the entertainment industry. All Firefly paths described in this paper are initialized according to the basic three-point lighting setup. In each iteration, the sampling is performed with 40 test paths and 14 samples for each path. As it is difficult to fully capture the dynamic behavior of our approach in text and still images, we encourage the reader to also refer to our supplemental video.

5.1 Molecular Structures

The exploration and analysis of molecular data is a prominent topic in scientific visualization. The purpose of molecular visualization is to provide an understanding of the rich and highly complex world of atomic structures, by mapping molecular structures, their functionality, properties, and interactions to visual characteristics. Molecular data is commonly very crowded, and the visualization of molecules features high visual complexity. The illumination of such complex geometry is a challenging task. In this example we use a 3D model of a tRNA structure 1GAX, consisting of 17210 atoms. This molecule has several deep cavities and tunnels, that pose a challenge to an illumination setup. A static setup will often result in deeply shadowed cavities and tunnels, hindering the geometry assessment. We can address this challenge by guiding the Firefly to the regions we want to illuminate using an input mask. We define a simple energy function that illuminates the surface with a desired brightness value of $\gamma = 0.3$. As strong shadows can be beneficial for shape perception, we use the energy function given in Equation 7. In Figure 11, we show two images of the same scene illuminated with two different Firefly paths. In Figure 11(a) ,the mask was set to cover the left side of the molecule, while in (b) the right part was the focus. The Firefly path automatically deforms to create a better energy for the masked part only. One can clearly see that tunnels on the left side of the molecule are much better illuminated in the top image.

5.2 Human Anatomy

Advances in computer technology have profoundly affected the domain of medical education. It has been shown that using 3D computer models as a teaching medium of human anatomy significantly improves the recollection of anatomical structures [28]. Often such models are slightly exaggerated to emphasize structures and textures of the anatomical objects. The visualization of such structures highly benefits from a lighting setup that highlights the variations of the model surfaces.

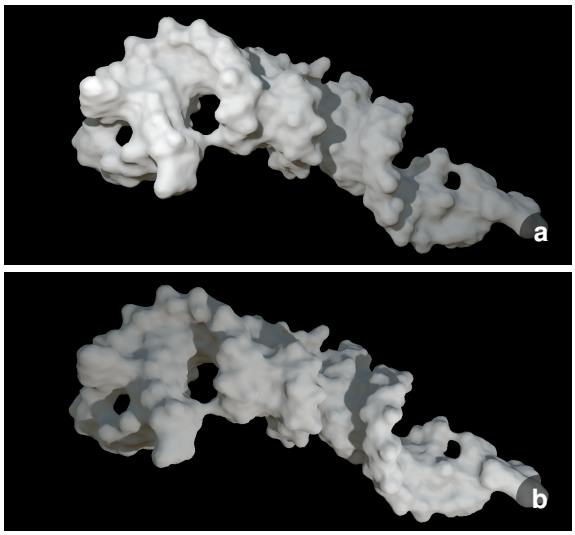

Fig. 11. Input mask guided illumination. In (a) the mask was held over the left side of the molecule. In (b) the right molecule side was masked. The Firefly path adapts automatically to the input mask. This way the user is able to adjust the illumination indirectly by moving the input mask.

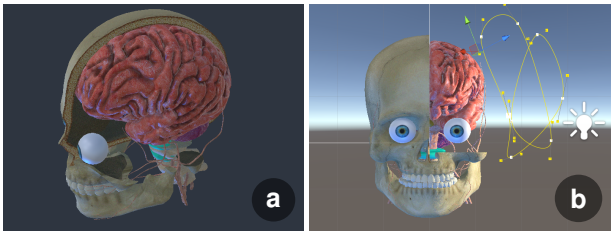

Fig. 12. (a) A model of the human head with a static light setup. (b) A firefly path after 34 iterations.

However, as the models tend do be highly complex, it is difficult to find a static light source that provides a good illumination of all relevant structures. We illustrate such a scenario on a model of the human head [2]. We show the model illuminated with the preset static lights in Figure 12(a). The complex spatial relationships between the individual structures can significantly benefit from a moving light source. Intuitively, a well-suited light path should not illuminate the object near to the camera position as this would decrease perceived shape variation [27]. Instead, the light should illuminate the object from several sides to account for all aspects of the model surface.

The shape of the brain gyri and sulci is most prominent with a high contrast between the valleys and ridges of the surface. We can indirectly measure the local contrast of the model through a local variance measure. However, using the variance alone does not account for the brightness of the image. Therefore, the image brightness must be included in the energy as well. Because the image brightness value is much higher than the variance, we multiply the variance with an importance factor ξ. To allow for stronger shadows, we measure the average instead of the maximum brightness for this example. Incorporating the difference in brightness and variance results in the following energy function:

$$E_I = \sum_{x,y} \xi \cdot \left(min_P |(var(sc,s) - \theta|) + avg_P(|br_{x,y} - \gamma|) \right) \qquad (10)$$

where θ is the desired variance. Here, we used $\theta = 0.02$ and $\xi = 5$. Using this energy function favors paths that emphasize high variance in the scene while maintaining uniform illumination. In Figure 12(b), we show the resulting Firefly path after 34 iterations. One can see that the Firefly path resembles two loops with a relatively uniform distance to the model. We show four consecutive snapshots for this Firefly path

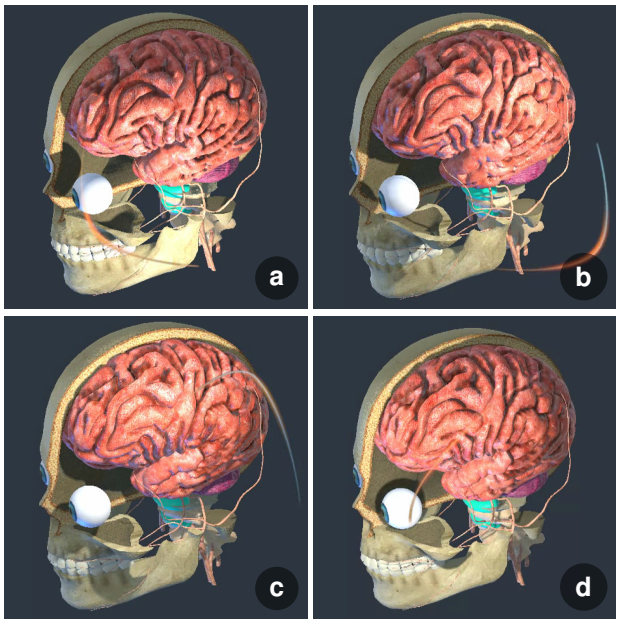

Fig. 13. Four consecutive positions of the Firefly. The Firefly travels on a loop illuminating the head from (a) below, (b) right, (c) above and (d) left.

in Figure 13. One can see that the Firefly indeed travels on a path that illuminates the scene from different directions (up, down, left, right). Interestingly, the Firefly path forms a loop, as this light trajectory is better suited to emphasize the features at the edge of the brain as well as the ones in the center of the model.

5.3 Animation

Lighting setup is a crucial part of cinematic animations. In addition to static lights, many animation techniques employ dynamic lights as they can be used to convey the passing of time, emphasize a dramatic change of a character, or change the focus of attention. When an illuminated character moves in the scene, the dynamic light needs to adjust to this motion, which poses a challenge on the animator to carefully construct a lighting path that still creates the desired result while in motion. With our approach this challenge can be addressed through the automatic adjustment of the Firefly path. As the Firefly path is defined relative to the object of interest, it moves with the object automatically adapting to the changing conditions. In this example, we use a character from the animated short film Adam [1]. We illuminate the same scene with three different illumination setups that create different effects for the user. The first setup creates a dramatic atmosphere, the second a threatening one, and the third setup aims to produce a calm atmosphere.

To create a dramatic effect, cinematography often employs rim lights. Rim lights are used to create dramatic scenes with a Chiaroscuro lighting. This artistic technique, developed in the Renaissance, uses strong lighting on one side of the object to create distinct one-sided shadows. We can formulate this by maximizing the variance and requiring a certain brightness in the image. To restrict the effect to one side of the object, we compute the dot product between the normalized vector from the Firefly to the object l_o and the right vector r in camera space. Minimizing the dot product favors light positions on the right side of the object.

To create a threatening scene effect, we follow the guidelines of digital cinematography [4]. Disregarding the light color, a threatening effect can be generated by using a low key light, that has a light intensity ratio of 8:1 between the key and fill light, and a strong back light. Furthermore, the light should illuminate the object from underneath creating a high variance in the scene.

In contrast, a calming atmosphere can be achieved with a high key light, i.e., a similar intensity of key and fill light. This setup is reduces the overall variance in the image. In addition, the light elevation should

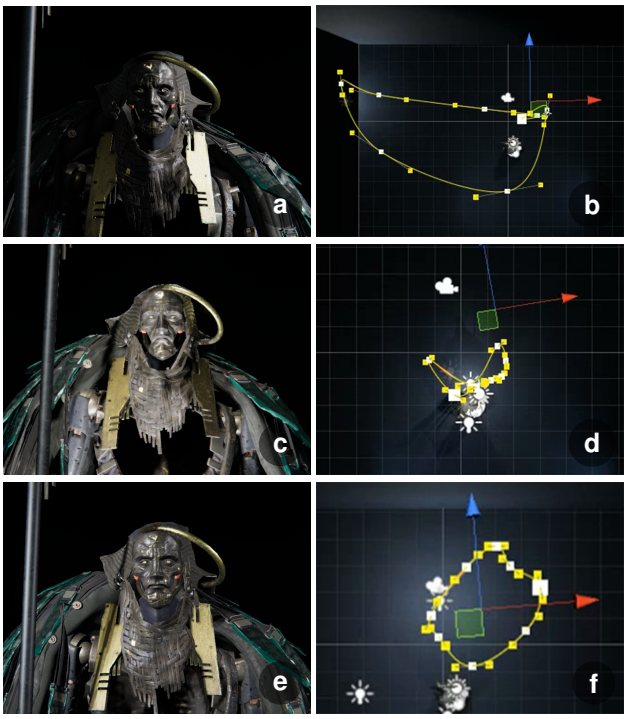

Fig. 14. Difference between scenes with different moods encoded through the energy function. (a) shows a dramatic scene, (b) a threatening scene and (e) a calm scene. Images (b),(d) and (f) show the corresponding paths, respectively.

be close to the elevation of the camera. Following these guidelines we can define the three energy functions as:

$$\text{dramatic:} \quad E_I = \sum_P -var_I(s) + |br_s - \gamma| + (l_0 \cdot r) \tag{11}$$

$$\text{threatening:} \quad E_I = \sum_P -var_I(s) + |br_s - \gamma| + |el + \frac{\pi}{3}| ds \tag{12}$$

$$\text{calm:} \quad E_I = \sum_P var_I(s) + |br_s - \gamma| + |el| ds. \tag{13}$$

Where el is the elevation angle in radians. In Figure 14, we show a representative result for the energy functions and the corresponding paths. One can clearly see the different moods present in the images. Figure 14(a) shows a dramatic scene, (c) a threatening scene, and (e) shows a rather calm atmosphere. The corresponding paths are shown in (b),(d), and (f).

5.4 Still Life

The assumption of light position is stronger for scenes with a familiar setup. Illuminating such a scene with a Firefly constructed with the previously described energy functions might create an odd scene not matching the user's expectations. Therefore, when designing an animated light for a realistic scene, for example a still life, we need to closely follow photographic lighting setup rules.

In this example, we illuminate a still life and formulate the energy function guided by the approach of Wambecke et al. [42]. This method, based on photographic rules, optimizes the azimuth and elevation of the light to emphasize the surface variation of the object while constraining the light position to the upper front hemisphere. Illuminating the scene with this method results in a rendering as shown in Figure 15(a).

A Firefly constructed in accordance with this method should be arranged above the camera and the object, with an azimuth range that accounts for the majority of the surface variation. Following the approach of Wambecke et al., we break down the light position into the light azimuth and elevation. The azimuth is computed using the structure tensor of the geometry measuring the direction of surface

Fig. 15. Still life rendered with the method of Wambecke et al. (a). The same model illuminated with the Firefly method (b,c,d). The lighting in (a) and (b) is almost identical, but (c) and (d) reveal the surface variation of the remaining model parts.

variance, and the elevation is used to produce a grazing light. For more details, we refer to the paper of Wambecke et al. [42].

For the azimuth, we first obtain the gradient of the surface depth ∇d for each pixel, and compute the local structure tensor $S_{x,y}$ for each image partition $P_{x,y}$:

$$\nabla d = \left(\frac{-n_x}{n_z}, \frac{-n_y}{n_z} \right) \tag{14}$$

$$S_{x,y} = \sum_{p \in P_{x,y}} \nabla d \nabla d^T. \tag{15}$$

The eigenvector e_1 of S associated to the largest eigenvalue λ_1 represents the direction in which most surface variations appear for each partition. If e_1 is pointing downwards, then the vector is simply flipped. Wambecke et al. define the light azimuth at an angle of e_1 in view space. In our approach, we measure the fitness of the Firefly azimuth by computing the dot product between e_1 and the normalized vector from the Firefly to the center of gravity of the illuminated object l_o.

To favor light positions close to e_1 for the whole path, we define the azimuth energy as:

$$E_{az} = max_s \left(1 - (e_1 \cdot l_o)\right). \tag{16}$$

The elevation of the light is chosen such that l_o is orthogonal to hidden surface normals n_0 defined as:

$$n_0 = \begin{cases} n & \text{if } l_{e_0,o} \cdot n > 0 \\ \vec{0} & \text{if } l_{e_0,o} \cdot n \leq 0 \end{cases} \tag{17}$$

where $l_{e_0,o}$ is the vector between the light position projected on an elevation of 0 and the object of interest. Therefore, the elevation energy is defined as:

$$E_{el} = max_s (l_{e_0,o} \cdot n_0). \tag{18}$$

Incorporating the information of the optimal light direction can lead to light positions too far away or too near to the object, resulting in too weak or to strong illumination. A straightforward solution would be to define a minimum distance between the object and the Firefly, but such a condition could produce rather unnatural illumination. Using the energy function, we can instead easily control the distance between the

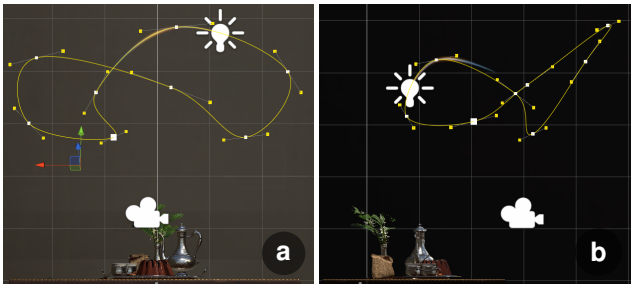

Fig. 16. The Firefly path for the still life scene seen from the front (a) and from the side (b). In (b), the Firefly moves away from the object briefly to generate illumination from the front that does not produce shadows on the object.

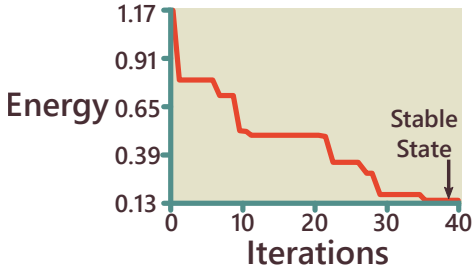

Fig. 17. A representative convergence curve for an energy function described in Equation 8. The Firefly path significantly improves already after a single iteration. After 10 iterations (approximately 9 seconds) the energy was more than halved. After 30 iterations, the energy state was almost optimal. The stable state was reached after 36 iterations requiring just under 33 seconds.

object and the Firefly indirectly by minimizing the maximal brightness difference for each image partition.

Putting all parts of the energy function together, the image energy is evaluated as a sum of the individual energy states over the image:

$$E_I = \sum_{x,y} \left(E_{az} + E_{el} + max_P(|br_{x,y} - \gamma|) \right). \quad (19)$$

Where x and y are the partition indices. The desired brightness γ was set to 0.3. Using this energy function results in a scene illumination shown in Figure 15(b), (c), and (d). We show the static lighting setup computed with the method of Wambecke in Figure 15(a). Naturally, the Firelfy produces very similar results as the static state-of-the-art method – in fact the light positions in images (a) and (b) are almost identical. However, for the static setup we had to adjust the distance between the light source and the object to achieve the desired brightness. Our method adjusted the distance automatically based on the desired brightness value. Furthermore, our method emphasizes all of the geometry throughout the path. In Figure 15(c), the Firefly creates shadows that emphasize the shape of the cake and in Figure 15(d) the shape of the coffee bag is much more prominent. We show the corresponding Firefly path in Figure 16. As expected, the path is organized above and in front of the object, forming an arc on the azimuth plane. In Figure 16(b), one can see that the path moves away from the object over a short path section in front of the object. This movement of the light away from the object generates a light position that produces almost no shadows on the object, thus creating an overall stronger illumination of the scene. To account for this, the light moves farther away in order to achieve an illumination close to the desired values.

6 Discussion

In our experiments we found that the animated light of the Firefly provides a powerful and versatile addition to a static lighting setup. We deliberately chose relatively simple energy functions for most of our examples, but nonetheless showed that our approach is capable of incorporating state-of-the-art solutions for static lighting design in a dynamic setup. The Firefly approach can be easily integrated into other renderers, but we believe that our implementation as a Unity plugin makes it already accessible to a large number of applications. Our method is independent of the scene content and rendering method, and hence can be used with a wide variety of different types of data.

While the evaluation of the test paths requires around 500 milliseconds on average, the fact that this process is executed in the background makes it transparent to the user. After the initialization, the optimization requires 35 to 40 iterations to reach a stable state. We show a representative convergence rate in Figure 17. The whole optimization process takes around 34 seconds for full convergence, but since the Firefly continuously updates to the best available solution, a good result is typically already achieved after about 9 seconds. When the user changes the view, the optimization needs 3 to 6 iterations to update the Firefly path to a new stable state, requiring 2.5 to 5 seconds. Because these background computations do not affect the performance of the main rendering thread and the Firefly transitions smoothly into a new path, we did not experience any negative effects on the overall usability. The initialization of the worker pool requires around 9 seconds during the program start, since all the workers need to be initialized with the corresponding Unity scene and settings.

When the user employs a selection mask, regions outside the selection are not accounted for in the illumination optimization. This can cause unwanted effects outside the selection mask, such as strong shadows or change of light intensity. However, we believe that these results are tolerable as the user deliberately shifts the attention to selected substructures. While dynamic light can emphasize and enhance certain aspects of the scene, it can also be misleading in some situations. Dynamic lights might be unsuitable for scenarios with highly dynamic scenes or dynamic textures as the change of light position might be interpreted as a change in the scene. Furthermore, dynamic lights might deflect the user's attention and thus may not be not suitable for applications where high concentration is needed to carefully study the data.

In this paper, we define a limited number of energy functions. To allow for an easy energy function formulation, we construct the energy function as an assembly of building blocks, measuring different properties in the scene and the rendered image. New energy functions can be constructed by using different combinations of these building blocks. However, the current version of the Firefly requires the user to define the energy function by writing a short code segment combining the building blocks. We are working on a visual editor that will allow the user to easily construct custom energy functions by combining the building blocks in a drag-and-drop manner. Finally, the effects of animated lights on the perception need to be formally evaluated, and we plan to conduct a user study investigating their perceptual consequences.

7 Conclusion and Future Challanges

We presented a novel approach for the automated generation of dynamic illumination paths in interactive scenes. Our approach shows that animated light provides a powerful and versatile addition to static lighting setups. We designed the Firefly tool as a flexible plugin that can be easily added to various visualization scenarios. The applicability of Firefly was demonstrated on various examples ranging from scientific visualization to applications for the entertainment industry. At present, the Firefly travels with a constant speed over the curve. In the future, we would like to investigate how the speed of the Firefly can be adapted for a better illumination without appearing unnatural to the user. While in this paper we only used point lights, Firefly is not restricted to a specific light type. In the future, we will investigate how Firefly can be integrated into complex lighting setups with box lights, strip lights, ring lights, and reflector probes. Moreover, we will investigate how a Firefly with dynamic chromatic light could emphasize changing moods in the scene.

Acknowldegments

The research presented in this paper was supported by the MetaVis project (#250133) funded by the Research Council of Norway.

REFERENCES

[1] Adam. https://unity3d.com/pages/adam. Accessed: 2018-03-17.

[2] U-anatomy; ufulio anatomy realistic. https://ufulio.wixsite.com/ufulioanatomy. Accessed: 2018-03-23.

[3] Unity game engine. https://unity3d.com. Accessed: 2018-03-23.

[4] A. A. Apodaca and L. Gritz. *Advanced RenderMan: Creating CGI for Motion Pictures*. Morgan Kaufmann, 1999.

[5] M. Aittala. Inverse lighting and photorealistic rendering for augmented reality. *The Visual Computer*, 26(6):669–678, 2010.

[6] M. Borga, A. Persson, R. Lenz, S. Lindholm, and G. Lathen. Automatic tuning of spatially varying transfer functions for blood vessel visualization. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2345–2354, 2012.

[7] D. Coffey, F. Korsakov, H. Hagh-Shenas, L. Thorson, A. Ellingson, D. Nuckley, and D. F. Keefe. Visualizing motion data in virtual reality: Understanding the roles of animation, interaction, and static presentation. *Computer Graphics Forum*, 31(3pt3):1215–1224, 2012.

[8] A. C. Costa, A. A. de Sousa, and F. N. Ferreira. Lighting design: A goal based approach using optimisation. In *Proc. Eurographics Workshop on Rendering*, pp. 317–328, 1999.

[9] C. de Melo and A. Paiva. Expression of emotions in virtual humans using lights, shadows, composition and filters. In *Proc. Affective Computing and Intelligent Interaction*, pp. 546–557, 2007.

[10] K. Doerschner, R. W. Fleming, O. Yilmaz, P. R. Schrater, B. Hartung, and D. Kersten. Visual motion and the perception of surface material. *Current Biology*, 21(23):2010–2016, 2011.

[11] M. S. El-Nasr and I. Horswill. Real-time lighting design for interactive narrative. In *Proc. Virtual Storytelling. Using Virtual RealityTechnologies for Storytelling*, pp. 12–20, 2003.

[12] F. Hunter, Steven Biver, and P. Fuqua. *Light Science & Magic: An Introduction to Photographic Lighting*. Focal Press, 2015.

[13] S. Freitag, B. Weyers, and T. W. Kuhlen. Automatic speed adjustment for travel through immersive virtual environments based on viewpoint quality. In *Proc. 3DUI*, pp. 67–70, 2016.

[14] M. E. Froese, M. Tory, G. W. Evans, and K. Shrikhande. Evaluation of static and dynamic visualization training approaches for users with different spatial abilities. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2810–2817, 2013.

[15] S. Gumhold. Maximum entropy light source placement. In *Proc. IEEE Visualization*, pp. 275–282, 2002.

[16] T. Günther, H. Theisel, and M. Gross. Decoupled opacity optimization for points, lines and surfaces. *Computer Graphics Forum*, 36(2):153–162, 2017.

[17] M. Halle and J. Meng. Lightkit: A lighting system for effective visualization. In *Proc. IEEE Visualization*, pp. 48–57, 2003.

[18] M. Haller, S. Drab, and W. Hartmann. A real-time shadow approach for an augmented reality application using shadow volumes. In *Proc. ACM Symposium on Virtual Reality Software and Technology*, pp. 56–65, 2003.

[19] N. Joubert, M. Roberts, A. Truong, F. Berthouzoz, and P. Hanrahan. An interactive tool for designing quadrotor camera shots. *ACM Transactions on Graphics*, 34(6):238:1–238:11, 2015.

[20] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1988.

[21] B. E. Keiser and P. Z. Peebles. An automatic system for the control of multiple drone aircraft. *IEEE Transactions on Aerospace and Electronic Systems*, AES-5(3):515–524, 1969.

[22] M. Keramat and R. Kielbasa. Latin hypercube sampling monte carlo estimation of average quality index for integrated circuits. *Analog Integrated Circuits and Signal Processing*, 14(1):131–142, 1997.

[23] D. Kersten, P. Mamassian, and D. C. Knill. Moving cast shadows induce apparent motion in depth. *Perception*, 26(2):171–192, 1997.

[24] D. A. Kleffner and V. S. Ramachandran. On the perception of shape from shading. *Perception & Psychophysics*, 52(1):18–36, 1992.

[25] G. Klein and D. Murray. Compositing for small cameras. In *Proc. IEEE/ACM International Symposium on Mixed and Augmented Reality*, pp. 57–60, 2008.

[26] C. H. Lee, X. Hao, and A. Varshney. Light collages: lighting design for effective visualization. In *Proc. IEEE Visualization*, pp. 281–288, 2004.

[27] P. Mamassian and R. Goutcher. Prior knowledge on the illumination position. *Cognition*, 81(1):1–9, 2001.

[28] D. T. Nicholson, C. Chalk, W. R. J. Funnell, and S. J. Daniel. Can virtual reality improve anatomy education? a randomised controlled study of a computer-generated three-dimensional anatomical ear model. *Medical Education*, 40(11):1081–1087, 2006.

[29] I. K. Nikolos, K. P. Valavanis, N. C. Tsourveloudis, and A. N. Kostaras. Evolutionary algorithm based offline/online path planner for UAV navigation. *IEEE Transactions on Systems, Man, and Cybernetics*, 33(6):898–912, 2003.

[30] B. Okumura, M. Kanbara, and N. Yokoya. Augmented reality based on estimation of defocusing and motion blurring from captured images. In *Proc. IEEE/ACM International Symposium on Mixed and Augmented Reality*, pp. 219–225, 2006.

[31] V. S. Ramachandran. Perception of shape from shading. *Nature*, 331(6152):163–166, 1988.

[32] A. Richards. *How to Set Up Photography Lighting for a Home Studio*. CreateSpace, 2014.

[33] S. Ruder. An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747, 2016.

[34] M. Ruiz, A. Bardera, I. Boada, I. Viola, M. Feixas, and M. Sbert. Automatic transfer functions based on informational divergence. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):1932–1941, 2011.

[35] H. H. Schütt, F. Baier, and R. W. Fleming. Perception of light source distance from shading patterns. *Journal of Vision*, 16(3):9, 2016.

[36] R. Shacked and D. Lischinski. Automatic Lighting Design using a Perceptual Quality Metric. *Computer Graphics Forum*, 20(3):215–227, 2001.

[37] M. D. Shields and J. Zhang. The generalization of latin hypercube sampling. *Reliability Engineering and System Safety*, 148(1):96–108, 2016.

[38] M. Srikanth, K. Bala, and F. Durand. Computational rim illumination with aerial robots. In *Proc. Computational Aesthetics*, pp. 57–66, 2014.

[39] M. Stein. Large sample properties of simulations using latin hypercube sampling. *Technometrics*, 29(2):143–151, 1987.

[40] Y. Tani, K. Araki, T. Nagai, K. Koida, S. Nakauchi, and M. Kitazaki. Enhancement of glossiness perception by retinal-image motion: Additional effect of head-yoked motion parallax. *PLOS ONE*, 8(1):1–8, 2013.

[41] P. J. M. van Laarhoven and E. H. L. Aarts. *Simulated annealing*. Springer Netherlands, 1987.

[42] J. Wambecke, R. Vergne, G.-P. Bonneau, and J. Thollot. Automatic lighting design from photographic rules. In *Proc. Eurographics Workshop on Intelligent Cinematography and Editing*, pp. 1–8, 2016.

[43] L. Wang and A. E. Kaufman. Lighting system for visual perception enhancement in volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(19):67–80, 2013.

[44] P. Wisessing, J. Dingliana, and R. McDonnell. Perception of lighting and shading for animated virtual characters. In *Proc. ACM Symposium on Applied Perception*, pp. 25–29, 2016.

[45] J. Xie, Y. Zhou, W. Wu, and Z. Zhou. Automatic path planning for augmented virtual environment. In *Proc. International Conference on Virtual Reality and Visualization*, pp. 372–379, 2016.

[46] M. Yang, Z. Liu, and W. Li. A fast general extension algorithm of latin hypercube sampling. *Journal of Statistical Computation and Simulation*, 87(17):3398–3411, 2017.

[47] Y. Zhang and K.-L. Ma. Lighting design for globally illuminated volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2946–2955, 2013.