Optimized Monte Carlo Path Generation using Genetic Algorithms

Philip Dutré

Frank Suykens

Yves Willems

Optimized Monte Carlo Path Generation using Genetic Algorithms

Philip Dutré

Frank Suykens

Yves Willems

Department of Computer Science, K.U.Leuven

Abstract

In this technical report we present a new method for optimizing the generation of paths in Monte Carlo global illumination rendering algorithms. Ray tracing, particle tracing, and bidirectional ray tracing all use random walks to estimate various fluxes in the scene. The probability density functions necessary to generate these random walks are optimized using a genetic algorithm, such that a significant reduction of the variance, and thus more reliable flux estimates, are obtained.

Optimized Monte Carlo Path Generation using Genetic Algorithms

Ph. Dutré, F. Suykens, Y.D. Willems

Department of Computer Science
Katholieke Universiteit Leuven
Celestijnenlaan 200A, B-3001 Heverlee, Belgium
{philipd,franks,ydw}@cs.kuleuven.ac.be

Abstract: In this technical report we present a new method for optimizing the generation of paths in Monte Carlo global illumination rendering algorithms. Ray tracing, particle tracing, and bidirectional ray tracing all use random walks to estimate various fluxes in the scene. The probability density functions necessary to generate these random walks are optimized using a genetic algorithm, such that a significant reduction of the variance, and thus more reliable flux estimates, are obtained.

1 Introduction and motivation

Monte Carlo path integration is one possible approach of solving the global illumination problem in computer graphics. Various algorithms have been proposed, such as stochastic ray tracing [3], particle tracing [4], and bidirectional tracing [13, 17]. All of these algorithms can be described in a single mathematical framework [6, 16, 18], and require the generation of paths (random walks), starting from the light- or potential sources.

The drawback of all Monte Carlo rendering algorithms is the stochastic variance on the estimator of the value of the different flux integrals. Variance is visible as noise in the image. Apart from increasing the number of paths, variance can be reduced using a number of different strategies such as stratified sampling, importance sampling, control variates, antithetic variates etc.[9].

Various approaches have been presented to reduce the variance in Monte Carlo global illumination algorithms. Most of these approaches use partial knowledge about the illumination in the scene to be rendered, in order to generate subsequent paths more efficiently, based on the principle of importance sampling. This partial knowledge, which may be built up during a preprocessing phase or gradually as the algorithm develops, can take many forms. Several authors have described techniques such as the use of irradiance maps [20], photon maps [10], potential functions [5], 5D trees [14]. A disadvantage of these approaches is that they require a significant amount of information to be stored. A more recent algorithm is the so-called Metropolis Light Transport algorithm [19]. The paths which are used for generating the image are mutated in a probabilistic way, such that they can contribute in a more significant way to the image later on.

This report proposes a new approach. In order to obtain a good path sampling strategy, we will not store and adapt information in the form of illumination values, but we rather manipulate the sampling procedures themselves. We will try to compute *path generators* that produce optimal paths for the computation of a flux integral. Information about the scene to be rendered will therefore be stored implicitly in the path generator sampling functions. Since the optimization of path generators is a non-trivial and

very complex problem, the optimization of the path generators is carried out using a genetic algorithm.

Genetic algorithms (GAs) [8, 15], optimize a given problem by considering a set of candidate solutions. Each candidate solution is tested on its fitness towards the problem at hand. A new generation of candidate solutions is made, based on their fitness, and subject to several genetic operators such as crossovers and mutation. Intuitively, a GA bears resemblance to natural 'survival of the fittest'. After a number of generations, a candidate solution with a high fitness will hopefully emerge, which serves as a solution for the initial problem. GAs have only be scarcely used before in rendering. Rayvolution [1] describes an approach where single rays are individuals in a population, which is a somewhat different approach from ours. In [2], a system is described that makes use of a neural structure to optimally sample the environment.

2 Overview of the algorithm

For each pixel in an image, we need to generate paths, originating at the light sources, the potential source (the eye), or both. Bidirectional tracing, where two random walks are generated simultaneously, is the most general form of a path rendering algorithm. In order to generate a complete path, we need several sampling procedures, which together can be regarded as a path generator. A path generator can favour some directions of where to generate paths to. Well known examples are path generators based entirely on BRDF-sampling or path generators which explicitly sample some surface patches with a higher probability.

The goal of our algorithm is to generate optimal path generators for integrals expressing a flux, by making the variance as low as possible. Because a single path generator is usually limited in how well it samples its environment, we consider a set of path generators as individuals for our GA. The path generators in such a set are combined in a single, more complex path generator, which is more versatile. The algorithm is summarized below:

```
create a random population of sets of path generators

for i = 1 to GENERATIONS
    compute the fitness of each set in the population
    generate the next generation, using genetic operators

estimate the flux through the pixel with the best set of path
generators of the last generation
```

Note that in principle, the actual Monte Carlo integration of the flux is only carried out after the GA has selected the most optimal set of path generators.

3 Global illumination mathematical framework

The global illumination problem can be formulated by two dual sets of equations. The rendering transport equation [11, 7] is probably the best known, and expresses the transport of radiance between two surface points x and y:

$$L(x \to y) = L_e(x \to y) + \int_A dA_z L(z \to x) f_r(x, z \leftrightarrow y) G(x, z)$$

$$= L_e(x \to y) + TL(x \to y) \tag{1}$$

$$\text{where} \quad G(x, z) = \frac{\cos(n_x, \overrightarrow{xz}) \cos(n_z, \overrightarrow{zx}) V(x, z)}{r_{xz}^2}$$

$L_e(x \to y)$ is the initial radiance distribution at the light sources, A is the union of all possible surfaces in the scene, $f_r(x, z \leftrightarrow y)$ is the BRDF at x, with directions towards y and z, $V(x, y)$ equals 1 if x and y are mutually visible, 0 otherwise.

The dual transport equation, describing the propagation of potential, is given by:

$$W(x \leftarrow y) = W_e(x \leftarrow y) + \int_A dA_z W(y \leftarrow z) f_r(y, x \leftrightarrow z) G(y, z)$$

$$= W_e(x \leftarrow y) + T^* W(x \leftarrow y) \tag{2}$$

Radiance and potential are dual quantities. Both are defined w.r.t. a given source, a light-source in the case of radiance, or a potential-source in the case of potential. One can prove that T and T^* are adjoint to each other for the following inner product of two functions defined over AxA:

$$\langle F_1, F_2 \rangle = \int_A dA_z \int_A dA_t F_1(z \to t) F_2(z \leftarrow t) G(z, t) \tag{3}$$

The flux of a set S, consisting of surface points and directions around these points and which acts as the source of potential, can be written as an inner product in two different ways [6]:

$$\Phi(S) = \langle L^{\rightarrow}, W_e^{\leftarrow} \rangle = \int_A dA_x \int_A dA_y L(x \to y) W_e(x \leftarrow y) G(x, y)$$

$$\text{or} \tag{4}$$

$$\Phi(S) = \langle L_e^{\rightarrow}, W^{\leftarrow} \rangle = \int_A dA_x \int_A dA_y L_e(x \to y) W(x \leftarrow y) G(x, y)$$

The transport equations for radiance and potential can be recursively substituted in both flux equations. Using the adjointness of T and T^*, and recursively substituting the transport equations, we obtain the following expression for $\Phi(S)$:

$$\Phi(S) = \int_{A \times A} dA_x dA_y L_e(x \to y) G(x, y) W_e(x \leftarrow y)$$

$$+ \int_{A \times A \times A} dA_x dA_y dA_z L_e(x \to y) G(x, y) f_r(y, x \leftrightarrow z) W_e(y \leftarrow z) \tag{5}$$

$$+ \ldots$$

Each of these integrals can be considered as an integration over all possible paths of length 0, 1, 2 etc. We can rewrite this sum as a single integral, by integrating over the complete path space:

$$\Phi(S) = \int_{A^*} f(X) d\mu(X) \tag{6}$$

where X is a complete path $\langle x, y, z, \ldots \rangle$, $f(X)$ the contribution of the path to $\Phi(S)$ and $d\mu(X)$ a measure for a single path.

Monte Carlo path integration algorithms for computing the above $\Phi(S)$ are well-described in literature. The most general approach is given by bidirectional path tracing. For each flux estimate, a path is generated simultaneously starting from a light source and the potential source. By linking the end- and/or midpoints of both random walks, one is able to get an estimate for $\Phi(S)$. Stochastic ray tracing and particle tracing are special cases of bidirectional path tracing, where the length of one of the random walks is set to 1 or 0, respectively when next event estimation is used or not.

In order for our flux estimate to be non-biased, the sampling functions that generate the paths should sample the complete domain. In practice, this means that each time a new point is added to the path, the whole hemisphere around the current surface point is being sampled.

4 Genetic Algorithms

GAs were first proposed in the 1960s, and have been the subject of research since then. A GA is essentially an optimization algorithm that tries to maximize a given function, called the fitness function. The fitness function is constructed such that an individual with a high fitness is a good solution for the problem to be solved. The fitness function can take on a very complex form, and can even require the execution of programs. The GA searches for the highest value on the fitness function using operators such as crossover and mutation. This section will only give a brief explanation of the GA, as necessary for the understanding of this technical report. The more interested reader is referred to [15] or [8].

A simple GA works as follows:

1. Start with a random population of N candidate solutions (individuals). Each individual can be represented as a sequence of genes. In simple GAs, a gene is just a bit, but it can also be a more complex combination of parameters.

2. Calculate the fitness of each individual in the population.

3. Create a new population by generating N offspring. For each pair of offspring:

   - Select a pair of parent individuals. The probability for selecting an individual as a parent is an increasing function of its fitness.

   - With a probability p_{cross}, the parents are recombined in two new individuals, by crossing their sequence of genes at a random point.

   - Each gene of each offspring is mutated with a probability p_{mutate}.

4. Replace the current population with the new population, and go to step 2.

Each new population is called a generation. The GA stops after a number of preset generations. The individual with the highest fitness in the final generation is selected as a solution for the problem.

It is obvious that the efficiency of the GA depends on the setting of the various parameters, such as the number of generations and the probabilities for crossover and mutation to occur. Also, the representation of an individual as a sequence of genes is also important, since this defines the nature of the crossover and mutation operators.

5 Genetic Rendering

5.1 Representation of path generators

Suppose we want to compute the flux of a set S using a Monte Carlo path integration algorithm. We need to generate a number of paths, but a good distribution of paths is not known beforehand. An example is given in figure 1. In this scene, a light-source contributes light through two major bundles to the flux of the indicated patch.One can see that two different path generators might be used to compute the flux of set S: one generator would favour paths passing through the upper-left corner; the other would favour paths passing through the lower-right corner. This sampling scheme would be in accordance with the principle of importance sampling of the flux integral, which says that more paths should generated where the contribution is highest. Similar examples involving specular surfaces and secondary light sources are easily constructed. Note that the arrows only indicates the direction of light transport. The actual generated paths could be originating at the light source, the set S, or both.
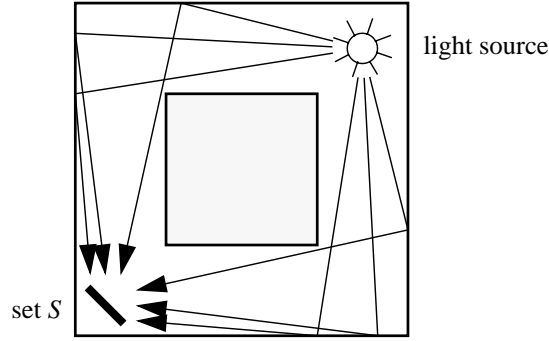

Fig. 1. Different light contributions to the flux of a set.

A path generator can be represented as a sequence of several samplers. Each sampler contains information about how the next point in the path should be generated. Classic examples of samplers are BRDF hemisphere sampling, uniform surface sampling, uniform hemisphere sampling or cosine hemisphere sampling, but in principle, any probability density function that samples the complete integration domain provides us with an unbiased estimator for the integral value. If we want to compute paths of variable length, an absorption coefficient associated with each sampler is also necessary. If no absorption occurs, each sampler adds a new point to the generated path so far.

5.2 Types of Samplers

As noted above, a path generator consists of a number of samplers. In our algorithm, we use 2 different samplers:

An *anchor point sampler* samples more outgoing directions in a solid angle directed towards an anchor point on a distant surface. Since a sampler has to cover the whole hemisphere in order to be unbiased, a uniform sampler over the entire hemisphere is added.

The anchor point sampler can be given a certain sampling behaviour by adjusting p_1 and p_2, the size of the preferred solid angle, and the position of the anchor point. Note that p_1 can also be smaller than p_2.
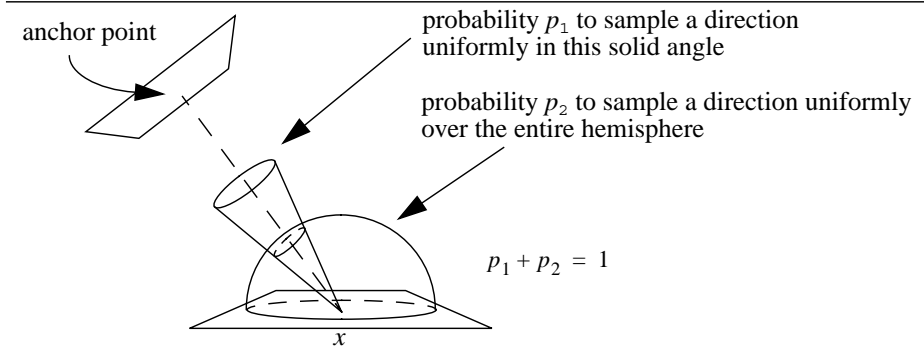
Fig. 2. Anchor point sampler

A *BRDF sampler* generates a new direction proportional to the value of the BRDF at the current surface point. This kind of sampler is useful when generating paths that meet highly specular surfaces.

It is possible to invent more samplers, to give more degrees of freedom to the path generators, e.g. a sampler that favours certain polygons or objects. It is the task of the GA to select the best samplers possible, not only their type, but also their various parameters.

An example of the behaviour of a path generator, is given in 3. The path generator consists of two anchor point samplers, with in both cases $p_1 > p_2$. The preferred solid angles are drawn in gray. Path 1 has a high probability of being sampled, because it lies within both gray angles. Path 2 on the other hand, has a lower probability. It was sampled outside the preferred region with sampler 1, but was then sent back to the 'correct' region due to anchor point 2. This kind of behaviour cannot be simulated using only BRDF sampling, and has the advantage that paths which stray of, are refocused some time later, thus ensuring that interesting paths are being generated.
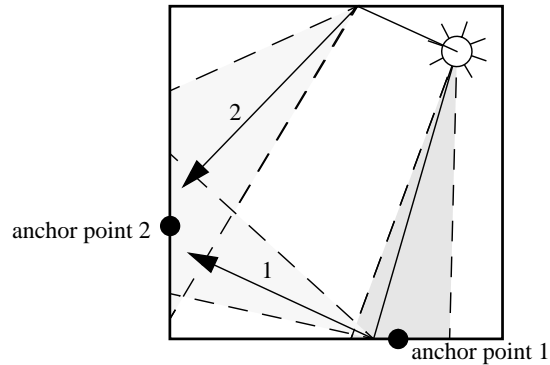

Fig. 3. Path generator with two anchor point samplers

5.3 Sets of path generators

A single path generator will probably not be flexible enough to sample accurately the flux integral. E.g. if a surface is illuminated by two distinct light sources, two anchor point samplers encompassing the light sources seem to be the best solution. We therefore consider a set of (weighted) path generators as an individual in the GA. Suppose

6

there are S path generators $p_i(X)$ in the set, each with an associated weight w_i. Each w_i indicates the probability that sampler i is chosen in order to generate a path. The total probability density function for generating a path X can then be written as:

$$pdf(X) = \sum_{i=1}^{S} w_i p_i(X) \qquad \sum_{i=1}^{S} w_i = 1 \qquad (7)$$

The estimator for the flux using a set of path generators, and generating N paths, is then given by:

$$\langle \Phi(S) \rangle = \frac{1}{N} \sum_{j=1}^{N} \frac{f(X_j)}{\sum_{i=1}^{S} w_i p_i(X)} \qquad (8)$$

Note that to combine the different path generators, the balance heuristic is used [18].

5.4 Fitness function

The fitness function is one of the more crucial aspects of a GA, since individuals are selected for creating offspring based on their fitness. The fitness function should therefore be a measure of how well a set of path generators can estimate $\Phi(S)$. Because we want the variance on $\langle \Phi(S) \rangle$ to be as small as possible, a natural starting point for designing a fitness function is the variance associated with the flux estimator. The variance on $\langle \Phi(S) \rangle$ is given by:

$$\sigma^2 = \int_D \frac{f(X)^2}{pdf(X)} d\mu(X) - \Phi(S)^2 \qquad (9)$$

All individuals in the population should be sorted according to their associated variance. Since $\Phi(S)^2$ is an (unknown) constant, the ordering does not change if we add it to both terms in the equation. The advantage is that we do not introduce an extra error by estimating $\Phi(S)^2$.

The remaining integral can be estimated by generating M paths with probability function $q(X)$:

$$\langle \sigma^2 + \Phi(S)^2 \rangle = \frac{1}{M} \sum_{i=1}^{M} \frac{f(X_i)^2}{pdf(X_i)q(X_i)} \qquad (10)$$

A lower value for this estimator indicates a better set of path generators. To compute the final fitness values, we scale these values and negate them, such that the lowest fitness equals 0; and the possible highest equals 1. These values can then be used for parent selection in the GA.

$$Fitness(ind) = 1 - \frac{\langle \sigma^2 + \Phi(S)^2 \rangle_{ind}}{\langle \sigma^2 + \Phi(S)^2 \rangle_{max}} \qquad (11)$$

The computation of the fitness for an individual requires M paths X_i to be sampled, and an evaluation of $f(X_i)$ for each path. This is very time-consuming, since $f(X_i)$ requires a fair number of visibility checks. However, if we use the same sets of M paths for all fitness calculations, we can precompute the associated $f(X_i)^2/q(X_i)$ values. We then use these stored $f(X_i)^2/q(X_i)$ values for the evaluation of the fitness of all individuals. The only thing left to do is divide by $pdf(X_i)$ and take the average. The advantage is that we do not have to generate M paths, and that the fitness value of an

individual remains the same over several generations. The drawback is that the pre-generated paths may be very bad in representing the function $f(X)$.

The overall result is that GA will produce a set of path generators that will sample the M paths efficiently, which is not exactly the same as an efficient sampling of $f(X)$. However, we are still assured of an unbiased estimator later on, when the actual Monte Carlo path generation is carried out.

5.5 Genetic Operators

The genetic operators are necessary in order for the GA to work. Since we consider a set of path generators as individuals for the GA, the crossover and mutation operators are chosen as follows:

Crossover operation: The crossover between two parents is schematically represented in figure 4. The first R path generators are switched between individuals to create the offspring. R is chosen randomly, such that at least one path generator is switched.
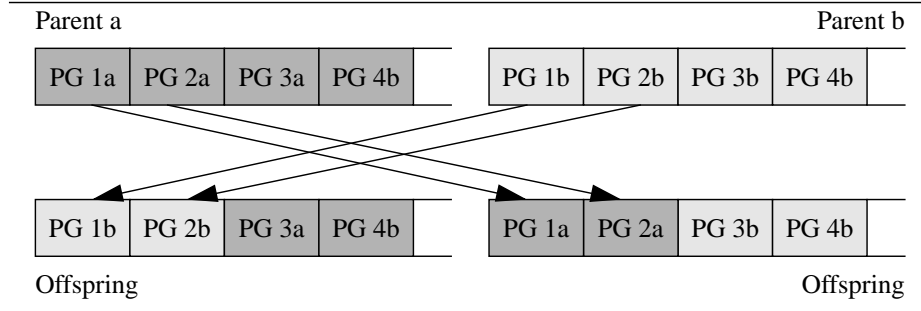
Parent a   Parent b

| PG 1a | PG 2a | PG 3a | PG 4b | | PG 1b | PG 2b | PG 3b | PG 4b |

| PG 1b | PG 2b | PG 3a | PG 4b | | PG 1a | PG 2a | PG 3b | PG 4b |

Offspring   Offspring

Fig. 4. Crossover of two sets of path generators

Mutation of a set of path generators: An offspring is subject to the following mutation procedure:

- All individual samplers, belonging to all generators in the set, are mutated with a probability $p_{sampler-mutate}$. A mutation of a sampler can be a simple operation such as shifting its anchor point, alter its absorption coefficient, or a complete replacement of one sampler by a new random sampler, which may be of a different type.

- With a probability p_{switch}, two path generators in the set switch positions within its internal representation. This is an event that allows the formation of a favourable order of path generators within a set, such that good combinations have a higher chance of being maintained during a crossover operation. This is a necessity because the order of path generators within a set is unimportant for the fitness, yet important for a good cross-over operator. If path generators that are a good combination are close to each other in the representation, the chance that they will break apart during a crossover is diminished [12].

- With a probability p_{weight}, the relative weights w_i of the path generators within the set are changed.

- With a probabilities p_{absorb}, the absorption coefficient can be changed. This actually controls the average length of the sampled path.

8

6 Experimental Results

We tested the above principles on a stochastic ray tracing algorithm, using next event estimation (a shadow ray) only at the end-point of a path. Thus, only path generators that start generate the path at the eye are part of the population. The test scene is shown in figure 5. A white diffuse surface is visible through the eye. which is indirectly illuminated through blue and red diffuse surfaces. We expect the GA to produce samplers that put anchor points on the blue and red surfaces, with a solid angle subtending these surfaces. The relative weight of these path generators depends on whether the pixel is situated on the left half or on the right half of the screen.
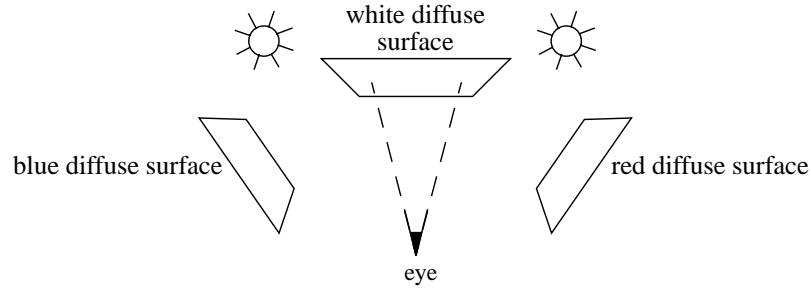

Fig. 5. Test scene

The position of these anchor points where indeed found by the GA. Also, the absorption coefficients were so that the path was absorbed with a high probability at the colored surfaces, such that a shadow feeler was sent to the light source. The exact sample point on the light surface was also optimized using a sampler. The resulting pictures are shown in figure 6. The picture on the left is computed with normal stochastic ray tracing, the picture on the right is achieved after the GA has been applied on a selected number of pixels. For pixels in between, the optimized path generators are combined. There is clearly less noise in the image generated with optimized path generators. The GA used 200 generations on a population of 150 individuals.
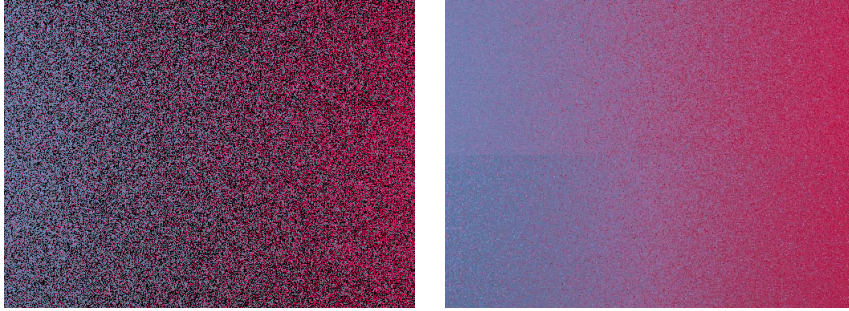

Fig. 6. Left: Stochastic ray tracing; Right: Stochastic ray tracing with optimized path generators

Some more runs were made, this time with varying size for the set of paths which is used in the fitness evaluations. The results are shown in figure 7, which shows the error with a reference image for both standard stochastic ray tracing, and with the use of our optimized path generators. The error made by our new algorithm is less than the error made by standard stochastic ray tracing.
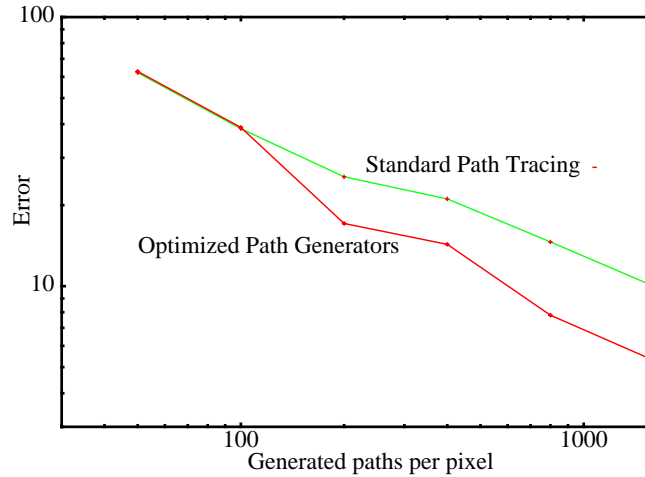
Fig. 7. Error vs number of paths per pixel

Although the algorithm was tried at a simple test scene, the current results are certainly encouraging.

7 Discussion

Ideally, our approach should be able to generate the best possible set of path generators for each flux. This means that the GA will decide where to generate paths, but also how they should be generated. E.g. if the camera looks at a mirror, that part of the path should be generated with a BRDF sampler, starting from the eye. A caustic could be generated by starting a path from the light source, etc. The same principle applies to more complex illumination conditions, such as secondary light sources, geometry constraints or visibility. By allowing a large degree of freedom in the set of path generators, the GA has the potential of locating the best sampling functions.

In practice, however, it is obvious that the behaviour and convergence rate of the GA is highly dependent on the different parameters of the GA. In the current implementation, parameters such as the number of generations, the crossover ratio, the chance for mutations etc. are set by hand and chosen by trial and error. By gaining more experience with genetic rendering, relationships between these parameters and the nature of the scene to be rendered could be derived. Literature about GA mentions lots of ideas on how to improve the convergence rate. A typical example is elitism. This means that the best parents are always duplicated in the next generation. The population is therefore less prone to sudden changes. We have used and implemented elitism in our algorithm. Due to the stochastic nature of our problem, this proved to be advantage.

Memory requirements for the algorithm is low, since we only need to store at most 2 populations at once, plus the test set of paths.

In order to generate a complete image, the flux for all pixels needs to be computed. In a straightforward approach, this means starting the GA from a random population for each pixel separately. One can see that this is not very economical. In principle, we want to use the same set of path generators for pixels which receive their illumination in a congruent manner. This does not necessarily mean that the surface visible through the pixel belongs to the same object, or has the same BRDF. An example is given in

figure 8. Although 2 different pixels see different objects in their aperture, with possible different BRDFs, the same sequence of 2 anchor point samplers is suitable for generating paths towards the light source. In the current implementation, the GA is run for a selected number of pixels. Intermediate pixels use the optimal set of path generators from the selected pixels, each weighted with the distance to the pixel under consideration. The coherence between pixels in finding the most optimal set of path generators is a topic worth further research.
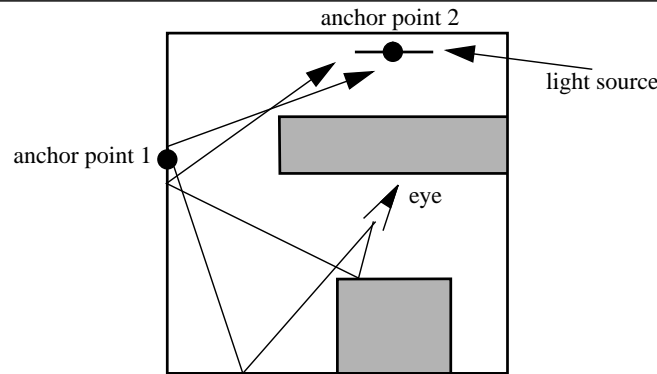

Fig. 8. Pixel coherence

8 Conclusion

We have presented a new way to determine the optimal path generators for use in a Monte Carlo rendering algorithm. Our approach uses a genetic algorithm to obtain an optimal set of path generators. This optimal set of generators is then used to compute the flux with Monte Carlo path integration. This still ensures us of an unbiased result.

The fitness values are based on an estimate for the variance and are computed with the same set paths for each set of path generators. This reduces the number of paths needed to run the algorithm.

Experimental results, currently applied to stochastic ray tracing only, indicate that less paths are needed to generate an image with the same error as standard sampling techniques. However, further testing is needed to really validate this technique for further use in rendering algorithms.

9 References

[1] M. Beyer and B. Lange. "Rayvolution: An evolutionary ray tracing algorithm". In *Proceedings of the Fifth Eurographics Workshop on Rendering*, pages 137-146, Darmstadt, Germany, June 1994. Also in *Photorealistic Rendering Techniques*, Springer-Verlag, 1995.

[2] E. Bustillo. "A neuro-evolutionary unbiased global illumination algorithm". In *Proceedings of the Eighth Eurographics Workshop on Rendering*, pages 263-274, St-Etienne, France, 1997. Also in *Rendering Techniques '97*, Springer-Verlag 1997.

[3] Robert L. Cook, Thomas Porter, and Loren Carpenter. "Distributed ray tracing.", *Computer Graphics (SIGGRAPH '84 Proceedings)*, volume 18, pages 137–145.

[4] Ph. Dutré and Y.D. Willems. "Importance-driven light tracing." In *Proceedings of the Fifth Eurographics Workshop on Rendering*, pages 185–194, Darmstadt, Germany, June 1994. Also in *Photorealistic Rendering Techniques*, Springer-Verlag, 1995.

[5] Ph. Dutré and Y.D. Willems. "Potential-driven Monte Carlo particle tracing in diffuse environments with adaptive probability density functions". In *Proceedings of the Sixth Eurographics Workshop on Rendering*, pages 306-315, Dublin, Ireland, June 1995. Also in *Rendering Techniques '95*, Springer-Verlag, 1995

[6] Ph. Dutré. *Mathematical Frameworks and Monte Carlo algorithms for Global Illumination in Computer Graphics*. Ph.D. Thesis, Katholieke Universiteit Leuven, September 1996.

[7] Andrew Glassner. *Principles of Digital Image Synthesis*. Morgan Kaufmann, 1995.

[8] D.E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989.

[9] J.M. Hammersly and D.C. Handscomb. *Monte Carlo Methods*. Chapman and Hall, London, 1964.

[10] Henrik Wann Jensen. "Global illumination using photon maps." In *Proceedings of the Seventh Eurographics Workshop on Rendering*, pages 22–31, June 1996.

[11] James T. Kajiya. "The rendering equation.", *Computer Graphics (SIGGRAPH '86 Proceedings)*, volume 20, pages 143–150.

[12] J.R. Koza. "Genetic Programming: A Paradigm for Genetically Breeding Populations of Computer Programs to Solve Problems", Technical Report CS-TR-90-1314, Computer Science Department, Stanford University, 1990

[13] E.P. Lafortune and Y.D. Willems. "Bi-directional path tracing." In *Proceedings of CompuGraphics*, pages 145–153, Alvor, Portugal, December 1993.

[14] E.P. Lafortune and Y.D. Willems, "A 5D Tree to Reduce the Variance of Monte Carlo Ray Tracing". In *Proceedings of the Sixth Eurographics Workshop on Rendering*, pages 11-20, Dublin, Ireland, June 1995. Also in *Rendering Techniques '95*, Springer-Verlag, 1995

[15] M. Mitchell. *An Introduction to Genetic Algorithms*. The MIT Press, 1996

[16] S. N. Pattanaik and S. P. Mudur. "The potential equation and importance in illumination computations." *Computer Graphics Forum*, 12(2):131–136, 1993.

[17] Eric Veach and Leonidas Guibas. "Bidirectional estimators for light transport." In *Proceedings of the Fifth Eurographics Workshop on Rendering*, pages 147–162, Darmstadt, Germany, June 1994. Also in *Photorealistic Rendering Techniques*, Springer-Verlag, 1995.

[18] Eric Veach and Leonidas J. Guibas. "Optimally combining sampling techniques for monte carlo rendering." In *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 419–428. ACM SIGGRAPH, 1995.

[19] Eric Veach and Leonidas J. Guibas. "Metropolis Light Transport", *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 65-76. ACM SIGGRAPH, 1997.

[20] G.J. Ward, F.M. Rubinstein and R.D. Clear. "A Ray Tracing Solution for Diffuse Interreflection". *SIGGRAPH 88 Conference Proceedings*, Annual Conference Series, pages 85-92. ACM SIGGRAPH, 1988.