EUROGRAPHICS 2013 / I. Navazo, P. Poulin
(Guest Editors)

Volume 32 (2013), Number 2

Supplemental Material: Path Space Regularization for Holistic and Robust Light Transport

Anton S. Kaplanyan and Carsten Dachsbacher

Karlsruhe Institute of Technology, Germany

1. Minimalistic Example: Path Tracing with Regularization

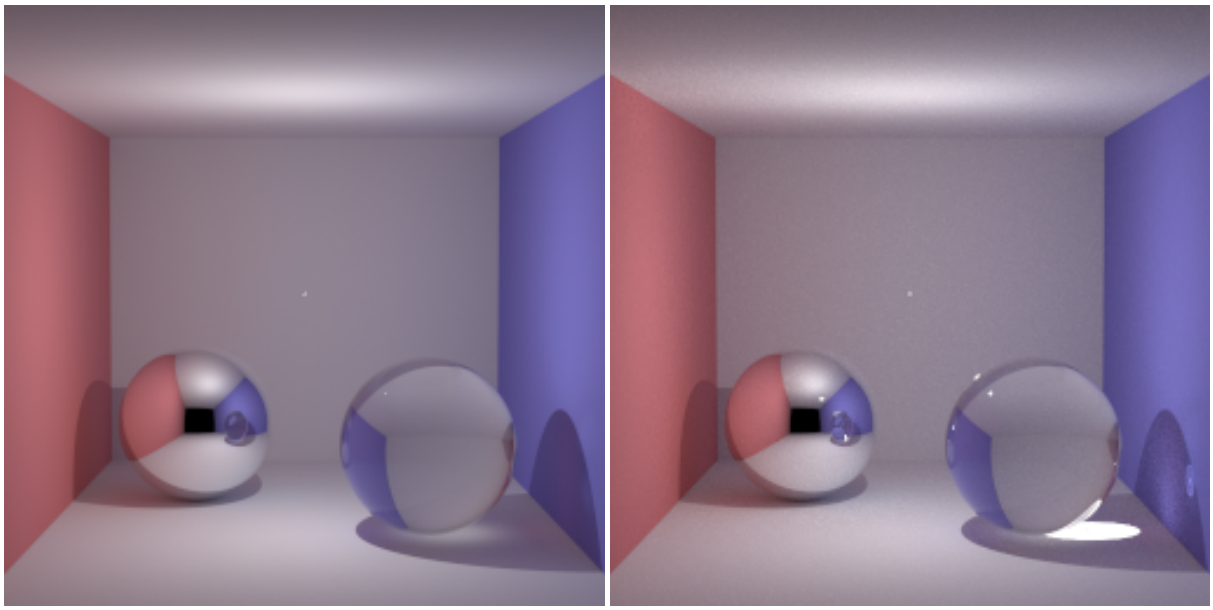
Here we show the little modifications required to an existing renderer for the example of a conventional unidirectional path tracer. This code is self-contained and based on the implementation "SmallPT" by Kevin Beason (which can be found at http://www.kevinbeason.com/smallpt/explicit.cpp). The changes and additions in the code required for our regularization method are highlighted in red.

```
#include <math.h>   // smallpt, a Path Tracer by Kevin Beason, 2008
#include <stdlib.h> // Make : g++ -O3 -fopenmp smallpt.cpp -o smallpt
#include <stdio.h>
typedef struct Vec {        // Usage: time ./explicit 16 && xv image.ppm
  double x, y, z;                  // position, also color (r,g,b)
  Vec(double x_=0, double y_=0, double z_=0){ x=x_; y=y_; z=z_; }
  Vec operator+(const Vec &b) const { return Vec(x+b.x,y+b.y,z+b.z); }
  Vec operator-(const Vec &b) const { return Vec(x-b.x,y-b.y,z-b.z); }
  Vec operator*(double b) const { return Vec(x*b,y*b,z*b); }
  Vec mult(const Vec &b) const { return Vec(x*b.x,y*b.y,z*b.z); }
  Vec& norm(){ return *this = *this * (1/sqrt(x*x+y*y+z*z)); }
  double dot(const Vec &b) const { return x*b.x+y*b.y+z*b.z; } // cross:
  Vec operator%(Vec&b){return Vec(y*b.z-z*b.y,z*b.x-x*b.z,x*b.y-y*b.x);}
} const cVec;
struct Ray { Vec o, d; Ray(Vec o_, Vec d_) : o(o_), d(d_) {} };
enum Refl_t { DIFF, SPEC, REFR };  // material types, used in radiance()
struct Sphere {
  double rad;       // radius
  Vec p, e, c;      // position, emission, color
  Refl_t refl;      // reflection type (DIFFuse, SPECular, REFRactive)
  Sphere(double rad_, Vec p_, Vec e_, Vec c_, Refl_t refl_):
  rad(rad_), p(p_), e(e_), c(c_), refl(refl_) {}
  double intersect(const Ray &r) const { // returns distance, 0 if nohit
    Vec op = p-r.o; // Solve t^2*d.d + 2*t*(o-p).d + (o-p).(o-p)-R^2 = 0
    double t, eps=1e-4, b=op.dot(r.d), det=b*b-op.dot(op)+rad*rad;
    if (det<0) return 0; else det=sqrt(det);
    return (t=b-det)>eps ? t : ((t=b+det)>eps ? t : 0);
  }
};
Sphere spheres[] = {//Scene: radius, position, emission, color, material
  Sphere(1e5, Vec( 1e5+1,40.8,81.6), Vec(),Vec(.75,.25,.25),DIFF),//Left
  Sphere(1e5, Vec(-1e5+99,40.8,81.6),Vec(),Vec(.25,.25,.75),DIFF),//Rght
  Sphere(1e5, Vec(50,40.8, 1e5),     Vec(),Vec(.75,.75,.75),DIFF),//Back
  Sphere(1e5, Vec(50,40.8,-1e5+170), Vec(),Vec(),           DIFF),//Frnt
  Sphere(1e5, Vec(50, 1e5, 81.6),    Vec(),Vec(.75,.75,.75),DIFF),//Botm
  Sphere(1e5, Vec(50,-1e5+81.6,81.6),Vec(),Vec(.75,.75,.75),DIFF),//Top
  Sphere(16.5,Vec(27,16.5,47),       Vec(),Vec(1,1,1)*.999, SPEC),//Mirr
  Sphere(16.5,Vec(73,16.5,78),       Vec(),Vec(1,1,1)*.999, REFR),//Glas
  Sphere(5e-3,Vec(50,81.6-36.5,81.6),Vec(4,4,4)*1e7, Vec(), DIFF),//Lite
};
double molif_r = 1.;  // Global mollification radius, shrinks per sample
double mollify(Vec& l, cVec& rd, Vec& n, Vec& nl, double dist, int type){
  double cos_max=1./sqrt(1.+(molif_r/dist)*(molif_r/dist));// Cone angle
  double solid_angle=2.*M_PI*(1.-cos_max);    // Solid angle of the cone
  Vec out = rd -n*2*n.dot(rd);                    // Reflection vector
  if(type == REFR){                          // Compute refraction vector
    double nc=1,nt=1.5,nnt=n.dot(nl)>0?nc/nt:nt/nc,ddn=rd.dot(nl),cos2t;
    if((cos2t=1-nnt*nnt*(1-ddn*ddn))>0)             // Refraction vector
      out =(rd*nnt-n*((n.dot(nl)>0?1:-1)*(ddn*nnt+sqrt(cos2t)))).norm();
  }
  return l.dot(out)>=cos_max?(1./solid_angle)/l.dot(out):0.; // Mollify
}
int numSpheres = sizeof(spheres)/sizeof(Sphere);
inline double clamp(double x){ return x<0 ? 0 : x>1 ? 1 : x; }
```

```
inline int toInt(double x){ return int(pow(clamp(x),1/2.2)*255+.5); }
inline bool intersect(const Ray &r, double &t, int &id){
  double n=sizeof(spheres)/sizeof(Sphere), d, inf=t=1e20;
  for(int i=int(n);i--;) if((d=spheres[i].intersect(r))&&d<t){t=d;id=i;}
  return t<inf;
}
Vec radiance(const Ray &r, int depth, unsigned short *Xi,int E=1){
  double t;                               // distance to intersection
  int id=0;                               // id of intersected object
  if (!intersect(r, t, id)||depth>10) return Vec();// if miss / too deep
  const Sphere &obj = spheres[id];        // the hit object
  Vec x=r.o+r.d*t, n=(x-obj.p).norm(), nl=n.dot(r.d)<0?n:n*-1,f=obj.c,e;
  double p = f.x>f.y && f.x>f.z ? f.x : f.y>f.z ? f.y : f.z; // max refl
  if (++depth>5||!p) if (erand48(Xi)<p) f=f*(1/p); else return obj.e*E;
  for (int i=0; i<numSpheres; i++){ // Explicit connections
    const Sphere &s = spheres[i];
    if (s.e.x<=0 && s.e.y<=0 && s.e.z<=0) continue; // skip non-lights
    Vec sw=s.p-x,su=((fabs(sw.x)>.1?Vec(0,1):Vec(1))%sw).norm(),sv=sw%su;
    double cos_a_max = sqrt(1-s.rad*s.rad/(x-s.p).dot(x-s.p));
    double eps1 = erand48(Xi), eps2 = erand48(Xi);
    double cos_a = 1-eps1+eps1*cos_a_max;
    double sin_a = sqrt(1-cos_a*cos_a), phi = 2*M_PI*eps2;
    Vec l = su*cos(phi)*sin_a + sv*sin(phi)*sin_a + sw*cos_a;
    if (intersect(Ray(x,l.norm()), t, id) && id==i)  // shadow ray
      e = e + f.mult(s.e*2*M_PI*(1-cos_a_max)*((obj.refl==DIFF) ?
        (l.dot(nl)*M_1_PI) : mollify(l,r.d,n,nl,t,obj.refl)));// BRDF
  }
  if (obj.refl == DIFF){                  // Ideal DIFFUSE reflection
    double r1=2*M_PI*erand48(Xi), r2=erand48(Xi), r2s=sqrt(r2);
    Vec w=nl, u=((fabs(w.x)>.1?Vec(0,1):Vec(1))%w).norm(), v=w%u;
    Vec d = (u*cos(r1)*r2s + v*sin(r1)*r2s + w*sqrt(1-r2)).norm();
    return obj.e*E+e+f.mult(radiance(Ray(x,d),depth,Xi,0));
  } else if (obj.refl == SPEC)            // Ideal SPECULAR reflection
    return obj.e+e+f.mult(radiance(Ray(x,r.d-n*2*n.dot(r.d)),depth,Xi));
  Ray reflRay(x, r.d-n*2*n.dot(r.d));     // Ideal dielectric REFRACTION
  bool into = n.dot(nl)>0;                // Ray from outside going in?
  double nc=1, nt=1.5, nnt=into?nc/nt:nt/nc, ddn=r.d.dot(nl), cos2t;
  if ((cos2t=1-nnt*nnt*(1-ddn*ddn))<0)    // Total internal reflection
    return obj.e+e+f.mult(radiance(reflRay,depth,Xi));
  Vec tdir = (r.d*nnt - n*((into?1:-1)*(ddn*nnt+sqrt(cos2t)))).norm();
  double a=nt-nc, b=nt+nc, R0=a*a/(b*b), c = 1-(into?-ddn:tdir.dot(n));
  double Re=R0+(1-R0)*c*c*c*c*c,Tr=1-Re,P=.25+.5*Re,RP=Re/P,TP=Tr/(1-P);
  return obj.e+e+f.mult(depth>2 ? (erand48(Xi)<P ?   // Russian roulette
    radiance(reflRay,depth,Xi)*RP:radiance(Ray(x,tdir),depth,Xi)*TP) :
  radiance(reflRay,depth,Xi)*Re+radiance(Ray(x,tdir),depth,Xi)*Tr);
}
int main(int argc, char *argv[]){
  int w=256,h=256,samps = argc==2 ? atoi(argv[1])/4 : 512; // # samples
  Ray cam(Vec(50,52,295.6), Vec(0,-0.042612,-1).norm()); // cam pos, dir
  Vec cx=Vec(w*.5135/h), cy=(cx%cam.d).norm()*.5135, r, *c=new Vec[w*h];
  for(int y=0; y<h;y++){unsigned short Xi[3]={0,0,y*y*y};// Looping rows
    fprintf(stderr,"\rRendering (%d spp) %5.2f%%",samps*4,100.*y/(h-1));
    for (unsigned short x=0; x<w; x++)   // Loop cols
      for (int sy=0, i=(h-y-1)*w+x; sy<2; sy++)     // 2x2 subpixel rows
        for (int sx=0; sx<2; sx++, r=Vec()){      // 2x2 subpixel cols
          for (int s=0; s<samps; s++){
            molif_r = 1.*pow(1.+s,-1./6); // Mollification shrinkage
            double r1=2*erand48(Xi), dx=r1<1 ? sqrt(r1)-1: 1-sqrt(2-r1);
            double r2=2*erand48(Xi), dy=r2<1 ? sqrt(r2)-1: 1-sqrt(2-r2);
            Vec d = cx*( ( (sx+.5 + dx)/2 + x)/w - .5) +
              cy*( ( (sy+.5 + dy)/2 + y)/h - .5) + cam.d;
            r = r + radiance(Ray(cam.o+d*140,d.norm()),0,Xi)*(1./samps);
          } // Camera rays are pushed ^^^^^ forward to start in interior
          c[i] = c[i] + Vec(clamp(r.x),clamp(r.y),clamp(r.z))*.25;
        }
  }
  FILE *f = fopen("image.ppm", "w");         // Write image to PPM file.
  fprintf(f, "P3\n%d %d\n%d\n", w, h, 255);
  for (int i=0; i<w*h; i++)
    fprintf(f,"%d %d %d ", toInt(c[i].x), toInt(c[i].y), toInt(c[i].z));
}
```

Note: For Windows systems it is necessary to define a few functions and constants. As a simple solution one could add two following lines in the very beginning of the file:

```
#define _USE_MATH_DEFINES
#define erand48(dummy) (double(rand()) / RAND_MAX)
```



(a) Standard path tracer (b) Path tracer with regularization

Figure 1: *Rendering with equal number of samples per pixel (65536 samples) using listing 1 with a very small light source (a) without regularization and (b) with regularization. Note that with regularization enabled the path tracer can find all transport paths, even pure specular chains (e.g. glass-mirror-light of which the caustic of the glass sphere seen in the reflection on the mirror sphere consists). Our algorithm resorts to a consistent estimation only for the paths that it cannot sample in an unbiased way.*

2. Regularization with Monte-Carlo Methods

First, we prove that it is possible to construct a consistent Monte-Carlo estimator with the canonical case of a d-dimensional delta distribution $\delta(\cdot)$ located at the origin of the d-dimensional space Ω.

Lemma 1. *Given a d-dimensional integrand of a form $f(x) = \delta(x)g(x)$, let $f_r(x) = \varphi_r(x)g(x)$ be a mollification of $f(x)$, where $g(x)$ is Lebesgue-integrable on Ω; and $\varphi_r(x)$ is a mollifier function with support on a ball $\mathbb{B}_r \subset \Omega$, such that $||\mathbb{B}_r|| \propto r^d$. Then, given that the random variable X is i.i.d., $X \sim \pi(\Omega)$, where $\pi(\cdot)$ is some importance sampling distribution of $g(x)$, such that $\forall X_n \in \Omega \mid g(X_n) > 0 : p(X_n) > 0$, then the expectation of the Monte-Carlo estimate $\hat{F}_N = \mathrm{E}\left[\frac{1}{N}\sum_{n=1}^{N}\frac{f_{r_n}(X_n)}{p(X_n)}\right]$ is*

$$\operatorname*{plim}_{N\to\infty}\left[\hat{F}_N\right] = \int_\Omega f(x)\mathrm{d}x = g(0) = F \tag{1}$$

for bandwidth sequences $\{r_n\}$ decreasing within the bounds $\mathrm{O}\left(n^{-1/d}\right) < r_n < \mathrm{O}(1)$.

Proof. In order to prove Eq. 1, we show that the mean squared error (MSE) of the left-side estimate vanishes to zero. Our proof is based on the results of Tornberg [Tor02]. We split the MSE into an analytical part $\mathrm{B}[\hat{F}]^2$ and a stochastic part $\mathrm{Var}[\hat{F}]$:

$$MSE[\hat{F}] = \mathrm{E}[\hat{F} - F]^2 = \mathrm{B}[\hat{F}]^2 + \mathrm{Var}[\hat{F}]. \tag{2}$$

Analytical Error. This mollification error (bias) $\mathrm{B}[\hat{F}]$ can be written as follows for a given constant bandwidth r

$$\mathrm{B}[\hat{F}_r] = \int_\Omega \varphi_r(x)g(x)\mathrm{d}x - \int_\Omega \delta(x)g(x)\mathrm{d}x = \int_\Omega \varphi_r(x)g(x) - g(0). \tag{3}$$

Denote the k-th moment of the mollifier function φ_r as $\mathcal{M}_k(\varphi_r) = \int_\Omega x^k\varphi_r(x)\mathrm{d}x$. Next, by expanding $g(x)$ into a d-dimensional Taylor series around zero in the first term, we obtain:

$$\begin{aligned}\mathrm{B}[\hat{F}_r] &= \sum_{k=0}^{\infty}\left(g^{(k)}(0)\int_\Omega \frac{x^k}{k!}\varphi_r(x)\mathrm{d}x\right) - g(0)\\ &= g(0)\mathcal{M}_0(\varphi_r) + g'(0)\mathcal{M}_1(\varphi_r) + \frac{1}{2}g''(0)\mathcal{M}_2(\varphi_r) + \mathrm{O}\left(\mathcal{M}_3(\varphi_r)\right) - g(0)\\ &= \frac{1}{2}g''(0)\mathcal{M}_2(\varphi_r) + \mathrm{O}\left(\mathcal{M}_4(\varphi_r)\right),\end{aligned} \tag{4}$$

where the multidimensional derivative $g^{(k)}$ is expressed as $g^{(k)}(x) = \nabla^k g(x)$. The last equality is due to the vanishing odd moments of φ_r and the normalization property of the mollifier.

Then we insert the mollifier of a form $\varphi(x) = r^{-d}\varphi_0(x/r)$:

$$\mathrm{B}[\hat{F}_r] = \frac{1}{2}g''(0)\int_{\mathbb{B}_r} x^2 r^{-d}\varphi_0(x/r)\mathrm{d}x + \mathrm{O}\left(\int_{\mathbb{B}_r} x^4\varphi_r(x)\mathrm{d}x\right) = \frac{1}{2}g''(0)r^2\mathcal{M}_2(\varphi_0) + \mathrm{O}\left(r^4\right), \tag{5}$$

where the constant $\mathcal{M}_2(\varphi_0) = \int_\Omega x^2\varphi_0(x)\mathrm{d}x$ is the second moment of the initial mollifier function φ_0.

Now we consider the upper bound of the collected average error B_N during Monte-Carlo simulation at step N, where r_n is varying at each step:

$$\mathrm{B}[\hat{F}]_N = \frac{1}{N}\sum_{n=1}^{N}\mathrm{B}[\hat{F}_{r_n}] = \frac{1}{N}\sum_{n=1}^{N}\left(\frac{1}{2}g''(0)r_n^2\mathcal{M}_2(\varphi_0) + \mathrm{O}\left(r_n^4\right)\right). \tag{6}$$

We need to enforce $\mathrm{B}[\hat{F}]_N \to 0$ as $N \to \infty$ for a consistent estimation. Thus we have the first boundary condition for the bandwidth shrinkage rate:

$$r_n < \mathrm{O}\left(n^0\right) = \mathrm{O}(1). \tag{7}$$

Stochastic Error. Here we make sure that the stochastic term of the MSE – the variance of the Monte-Carlo integration – also vanishes. The variance of the estimator $\hat{F}$ for a single sample with mollification bandwidth r is

$$\begin{aligned}\mathrm{Var}[\hat{F}_r] &= \mathrm{E}[\hat{F}_r^2] - \mathrm{E}[\hat{F}_r]^2 = \int_\Omega r^{-2d}\varphi_0(x/r)^2 g(x)\mathrm{d}x - \left(g(0)+\mathrm{B}[\hat{F}_r]\right)^2 \\ &= \int_\Omega r^{-2d}\varphi_0(x/r)^2 \sum_{k=0}^{\infty}\frac{x^k}{k!}g^{(k)}(0)\mathrm{d}x - \left(g(0)+\mathrm{B}[\hat{F}_r]\right)^2 \\ &= r^{-d}\left(\int_\Omega \varphi_0(y)^2\mathrm{d}y\right) g(0)^2 + \mathrm{O}(1). \end{aligned} \tag{8}$$

The equation in the last line is obtained after the change of variable $y = x/r$. Also according to Eq. 7 the bandwidth r is asymptotically $r \ll 1$. We then express the average variance of the Monte-Carlo integration as

$$\mathrm{Var}[\hat{F}]_N = \frac{1}{N^2}\sum_{n=1}^{N}\mathrm{Var}[\hat{F}_{r_n}] = g(0)\left(\int_\Omega \varphi_0(x)^2\mathrm{d}x\right)\frac{1}{N^2}\sum_{n=1}^{N} r_n^{-d} + \frac{1}{N^2}\sum_{n=1}^{N}\mathrm{O}(1). \tag{9}$$

The first equality is due to the assumption that all samples are i.i.d. Again, we enforce $\mathrm{Var}[\hat{F}]_N \to 0$ as $N \to \infty$ in order to eventually achieve consistency. This yields the second boundary condition for the bandwidth shrinkage rate

$$r_n > \mathrm{O}\left(n^{1/d}\right). \tag{10}$$

□

Theorem 1. *Given the unified space of all light transport paths Ω, the Monte-Carlo estimation $\hat{I}_N = \mathrm{E}\left[\frac{1}{N}\sum_{n=1}^{N}\frac{f_{r_n}(\bar{x}_n)}{p(\bar{x}_n)}\right]$ of the path integral as in Eq. 1 in the paper with a measurement function $f(\bar{x})$ as an integrand, converges consistently under the conditions of Lemma 1 with selective regularization of $f_r(\bar{x})$, i.e.*

$$\operatorname*{plim}_{N\to\infty}\left[\hat{I}_N\right] = I \tag{11}$$

only if the sequence of mollification bandwidths $\{r_n\}$ decreases within the bounds $\mathrm{O}\left(n^{-1/d}\right) < r_n < \mathrm{O}(1)$.

Proof. The integrand $f(\bar{x})$ can be uniquely separated into two integrands: an integrand which consists solely of samplable (regular) paths $g(\bar{x})$ and an integrand consisting only of non-samplable (irregular) paths $h(\bar{x})$ (due to the determinism of the path classification). Then the path integral can also be decomposed into a regular and an irregular part $I = I^0 + I^1 = \int_\Omega g(\bar{x})\mathrm{d}\mu(\bar{x}) + \int_\Omega h(\bar{x})\mathrm{d}\mu(\bar{x})$, where f_r is a selectively regularized measurement function. The conventional Monte-Carlo proof passes for the regular integral $\int_\Omega g(\bar{x})\mathrm{d}\mu(\bar{x})$. Thus we need to prove that the Monte-Carlo estimate of the last integral converges to I^1.

Given an i.i.d. random variable on Ω, and the fact that the space Ω is translation invariant, we apply Lemma 1 to the Monte-Carlo estimator of the regularized integral $\hat{I}_r^1 = \int_\Omega h_r(\bar{x})\mathrm{d}\mu(\bar{x})$, yielding

$$\operatorname*{plim}_{N\to\infty}\left[\hat{I}_N^1\right] = I^1. \tag{12}$$

This result implies the bounds for the mollification bandwidth shrinkage rate from Lemma 1. □

3. Regularization with Markov Chain Monte-Carlo Methods

In the following we provide a proof for Theorem 3 in the paper.

3.1. Preliminaries

The transition (Markov) kernel $K(\cdot,\cdot)$ of a Markov chain is a mapping $\Omega \mapsto \Omega$ with source $x \in \Omega$ and target $y \in \Omega$ which is defined as

$$K(x,y) = P(y|x), \tag{13}$$

where $P(y|x)$ is the conditional probability of accepting the new state y given that the current state of the Markov chain is x.

The transition kernel is also defined for a set $Y \subset \Omega$

$$K(x,Y) = \int_Y K(x,y)\mathrm{d}y, \tag{14}$$

denoting the probability of moving to the set Y from the current state x.

A transition kernel $K(\cdot,\cdot)$ is called *Harris-recurrent* if, given sets A and B from Ω along with a positive number ε and a probability measure ρ, the following holds

1. If $\tau_A = \inf\{n \geq 0 : x_n \in A\}$, then $P_z(\tau_A < \infty) > 0$ for all z.
2. If $x \in A$ and $C \subset B$, then $K(x,C) \geq \varepsilon\rho(C)$.

In essence, this technical definition can be rephrased as follows: given two states x_1 and x_2 in A, then there is at least an ε chance that they can be moved together to the same point at the next time step.

A transition kernel $K^n(\cdot,\cdot)$ is called an *n-step transition kernel* and is defined recursively as

$$K^n(\cdot,\cdot) = \int_\Omega K(y,\cdot)K^{n-1}(\cdot,y)\mathrm{d}y, \tag{15}$$

where $K^1 = K(\cdot,\cdot)$. The term $K^n(x_0,x_n)$ denotes the probability of landing into the state x_n after n moves (mutations) of the Markov chain initialized with the state x_0.

Note that generally the kernel $K = K_n$ depends on the move n. In this case the Markov chain is called a *non-homogeneous Markov chain*. The central limit theorems, which provide the ergodicity conditions for non-homogeneous Markov chains, were developed by Dobrushin [Dob56]. Before introducing the criterion of weak ergodicity, we first define the *Dobrushin's ergodic coefficient* $\delta(K)$ as

$$\delta(K^{(m,k)}) = \frac{1}{2}\sup_{x,y\in\Omega}\int_\Omega |K^m(x,z) - K^{m+k}(y,z)|\mathrm{d}z, \tag{16}$$

which is bounded in the range $0 \leq \delta(K) \leq 1$ and "assesses" the changes in the kernel $K(\cdot,\cdot)$ from the step m till the step $m+k$.

Now we provide the necessary and sufficient criterion for the *weak ergodicity* of a non-homogeneous Markov chain [Dob56]

$$\forall m \geq 0 : \lim_{k\to\infty} \delta(K^{(m,k)}) \to 0, \tag{17}$$

which intuitively means that the changes in the transition kernel at every move (iteration) should be small enough to provide a weak mixing of the Markov chain. Note that $\delta(K^{(m,k)}) \leq \prod_{n=m}^{k} \delta(K^{(n,n+1)})$ for all m and k. Hereafter we will denote the ergodic coefficient for a single step, with a slight and innocuous ambiguity, as $\delta(K^{(n,n+1)}) = \delta(K^n)$ for brevity. Alternatively, Eq. 17 can be written in an equivalent way as

$$\prod_{n=0}^{\infty} \delta(K^n) \to 0. \tag{18}$$

We refer the interested reader to the corresponding literature on Markov chains for the further reading, e.g. the book of Pierre Brémaud [Bre99]. We will use the aforementioned results during the proof of the following theorem.

3.2. Main result

Theorem 2. *Given the unified space of all light transport paths Ω as a state space, and a Harris-recurrent transition kernel $K(\cdot,\cdot)$ on it, the Markov chain Monte-Carlo estimation of an integral*

$$I = \int_\Omega f(\bar{x}) \mathrm{d}\mu(\bar{x}) \tag{19}$$

with selective regularization of the integrand $f_r(\bar{x})$ (as in Sect. 4 in the paper) converges almost surely if the mollification bandwidth shrinkage rate is $\mathrm{O}(\gamma^n) \leq r_n < \mathrm{O}(1)$ for some $\gamma \in (0;1)$.

Proof. Similarly to the proof of Theorem 1, we consider a decomposition of Ω into a subspace of regular paths Ω_{reg} and a subspace of irregular (non-samplable) paths Ω_{irreg}.

The outline of our proof is as follows: we first use the lower bound of the original Harris-recurrent kernel $K(\cdot,\cdot)$ (initially defined for regular paths only) and then show that by adding the regularization of the ill-posed paths, the Markov chain, which runs on both regular and (regularized) irregular paths, becomes weakly ergodic.

Since the original kernel K is Harris-recurrent, it passes the original Metropolis proof [MRR*53] and there exists a lower bound $0 < \kappa < 1$ for it, such that for all regular paths on Ω_{reg} the following holds [Bre99]:

$$\delta(K^n) \leq \mathcal{L}\kappa^n, \tag{20}$$

for some positive finite constant $0 < \mathcal{L} < \infty$.

The Markov chain running on regularized paths is similar to the simulated annealing process: the target distribution is slowly "cooled down" throughout the integration. Thus, following Brémaud [Bre99], we first find a lower bound of the regularized kernel K'_{r_n} for the step n. For that, we first assume that the mollification happens in $0 < d < \infty$ dimensions. By adding the mollification, we make the transition kernel also non-zero on the target subspace of all irregular paths Ω_{irreg} and its regularized neighbourhood. It is important to notice that the set of regularized paths, which represent one irregular subpath, should have a smoothly vanishing mass due to the gradual reduction of the mollification bandwidth (and correspondingly the support of the regularized set).

Then we recall the property of a d-dimensional mollifier: there is a constant $0 < \mathcal{M}_\varphi < \infty$ (usually equal one) that $\varphi_0 < \mathcal{M}_\varphi$ everywhere, and thus $\varphi_n < \mathcal{M}_\varphi r_n^{-d}$. The transition probability for escaping such set of (regularized) irregular paths is proportional to contribution of regularized paths, which grows as the lower bound of the mollifier. Using this property, along with the lower bound κ for the non-mollified part of the original transition kernel (we use the regular mutations from the original kernel K for non-mollified dimensions), we obtain a lower bound for the Dobrushin's ergodicity coefficient for the transition kernel K'_{r_n} at move n with bandwidth r_n (see e.g. [Bre99, Chapter 6] for a more detailed proof of the simulated annealing):

$$\delta({K'_{r_n}}^n) \leq \delta(K^n)\mathcal{M}_\varphi \mathcal{L} r_n^{-d} \leq \mathcal{M}_\varphi \mathcal{L}\kappa^n r_n^{-d}. \tag{21}$$

See also Mitra et al. [MRSV86] for similar bounds on simulated annealing with slowly mixing non-homogeneous Markov chains.

Now we can use Eq. 18 (Dobrushin's inequality for non-homogeneous Markov chains)

$$\prod_{n=1}^{\infty} \delta({K'_{r_n}}^n) \to 0, \tag{22}$$

which guarantees the weak ergodicity of the constructed Markov chain.

By inserting the lower bound from Eq. 21 into this equation

$$\prod_{n=1}^{\infty} \delta({K'_{r_n}}^n) \leq \prod_{n=1}^{\infty} \left(\mathcal{M}_\varphi \mathcal{L}\kappa^n r_n^{-d} \right) \to 0, \tag{23}$$

we obtain the final bounds on the shrinkage rate of the mollification bandwidth r_n:

$$\mathrm{O}\left(\gamma^n\right) \leq r_n < \mathrm{O}(1), \tag{24}$$

where $\kappa \leq \gamma < 1$ should be greater or equal to the spectral gap κ of the original transition kernel $K(\cdot,\cdot)$. □

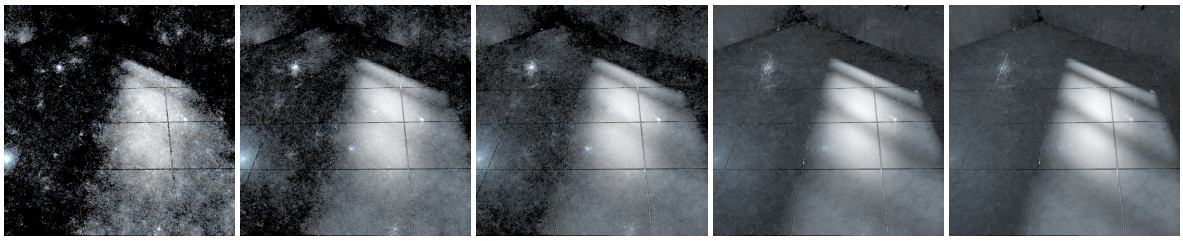


Figure 2: *An illustration of the convergence of the regularized Metropolis light transport method. The light is coming through a window from outside. The caustics on the floor are regularized using angular mollification of the specular BRDF on the window glass for demonstration purposes. Left to right: 1/5/10/20/50 mutations per pixel. Note how the silhouettes of the caustics become sharper during the integration process, demonstrating the behaviour of the MCMC estimator with regularization in practice.*

3.3. Discussion on the Convergence of the Regularized Markov Chain Monte-Carlo Methods

Selecting the regularization parameters. One practical sequence of bandwidths would be $r_n = r_0 \gamma^n$. There are no theoretical restrictions on the selection of the initial bandwidth r_0. However, from practical considerations, we recommend to select it such that the used set of mutation strategies can easily sample the subset of the path space which is covered by the smallest regularized feature in the image. According to our experiments, this parameter can be determined using the same heuristic as the one used for computing the initial global radius in progressive photon mapping (PPM) [HOJ08].

Note that the parameter $\gamma \in (0;1)$ has a different meaning comparing to the Monte Carlo integration. This parameter ideally should be selected close enough to the kernel spectral gap (the distance between two largest eigenvalues of the transition kernel) which denotes the base of the geometric convergence rate of the probability measure $K^n(x_0,\cdot)$ with any valid initial state x_0 to the target distribution in total variation [Bre99]. It has to be chosen close enough to one to guarantee practical convergence. Intuitively that means that depending on the mutation strategies used, the parameter γ should be selected such that the regular part of the integral (the part of the original MLT method) converges faster than the singularities from ill-posed paths start to appear in the integrand. In case if this value is selected wrongly (smaller than the spectral gap), the practical mixing of the chain is dominated by the regularized features and eventually lead to a wrong image where regularized features would be oversampled while the rest of the image would remain undersampled.

Requirements to the MLT mutation strategies. The proof of Th. 2 is based on the assumption that the set of mutation strategies used in some particular implementation of Metropolis light transport meets the practical conditions for Harris recurrence of the transition kernel. In case of Metropolis-Hastings sampling used in MLT this means that there should be a non-zero probability (bounded with some positive constant from below) that the chain will return into any current state it can ever land into.

For the original MLT [Vea98] this is practically achieved by using the bidirectional mutation strategy, which can regenerate the path from scratch and thus guarantees that the visited path can be sampled again.

In case of Kelemen mutations in the hypercube of random numbers [KSKAC02], this condition is satisfied by the so-called "large step" mutation, which also regenerates a complete path from scratch, providing the positive probability of returning into any state.

References

[Bre99] BREMAUD P.: *Markov Chains: Gibbs Fields, Monte Carlo Simulation, and Queues*. Springer, 1999. 6, 7, 8

[Dob56] DOBRUSHIN R. L.: Central limit theorems for non-stationary Markov chains II (Russian). *Theory of probability and its applications 1*, 4 (1956), 356–425. 6

[HOJ08] HACHISUKA T., OGAKI S., JENSEN H. W.: Progressive photon mapping. *ACM Transactions on Graphics 27*, 5 (2008), 130:1–130:8. 8

[KSKAC02] KELEMEN C., SZIRMAY-KALOS L., ANTAL G., CSONKA F.: A simple and robust mutation strategy for the metropolis light transport algorithm. *Computer Graphics Forum 21*, 3 (2002), 531–540. 8

[MRR*53] METROPOLIS N., ROSENBLUTH A. W., ROSENBLUTH M. N., TELLER A. H., TELLER E.: Equation of state calculations by fast computing machines. *The Journal of Chemical Physics 21*, 6 (1953), 1087–1092. 7

[MRSV86] MITRA D., ROMEO F., SANGIOVANNI-VINCENTELLI A.: Convergence and finite-time behavior of simulated annealing. *Advances in Applied Probability 18*, 3 (1986), 747–771. 7

[Tor02] TORNBERG A.-K.: Multi-dimensional quadrature of singular and discontinuous functions. *BIT Numerical Mathematics 42*, 3 (2002), 644–669. 4

[Vea98] VEACH E.: *Robust monte carlo methods for light transport simulation*. PhD thesis, Stanford University, 1998. AAI9837162. 8