2020_07_18__采样_Sobol采样.md

目录

Sobol采样

Sobol 采样是一个基于一系列矩阵的采样方式

这里是其矩阵的示意图

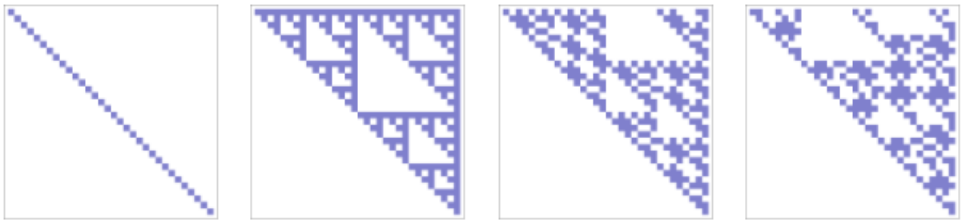
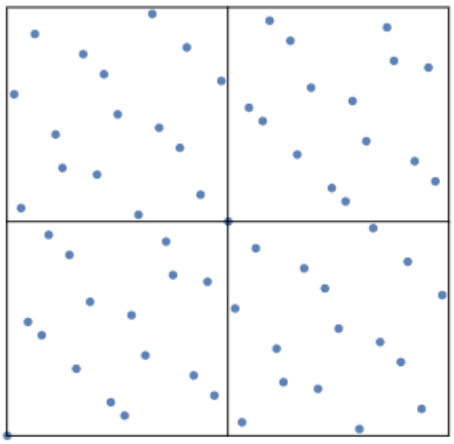


Figure 7.34: Generator matrices for the first four dimensions of the Sobol′ sequence. Note their regular structure.

1

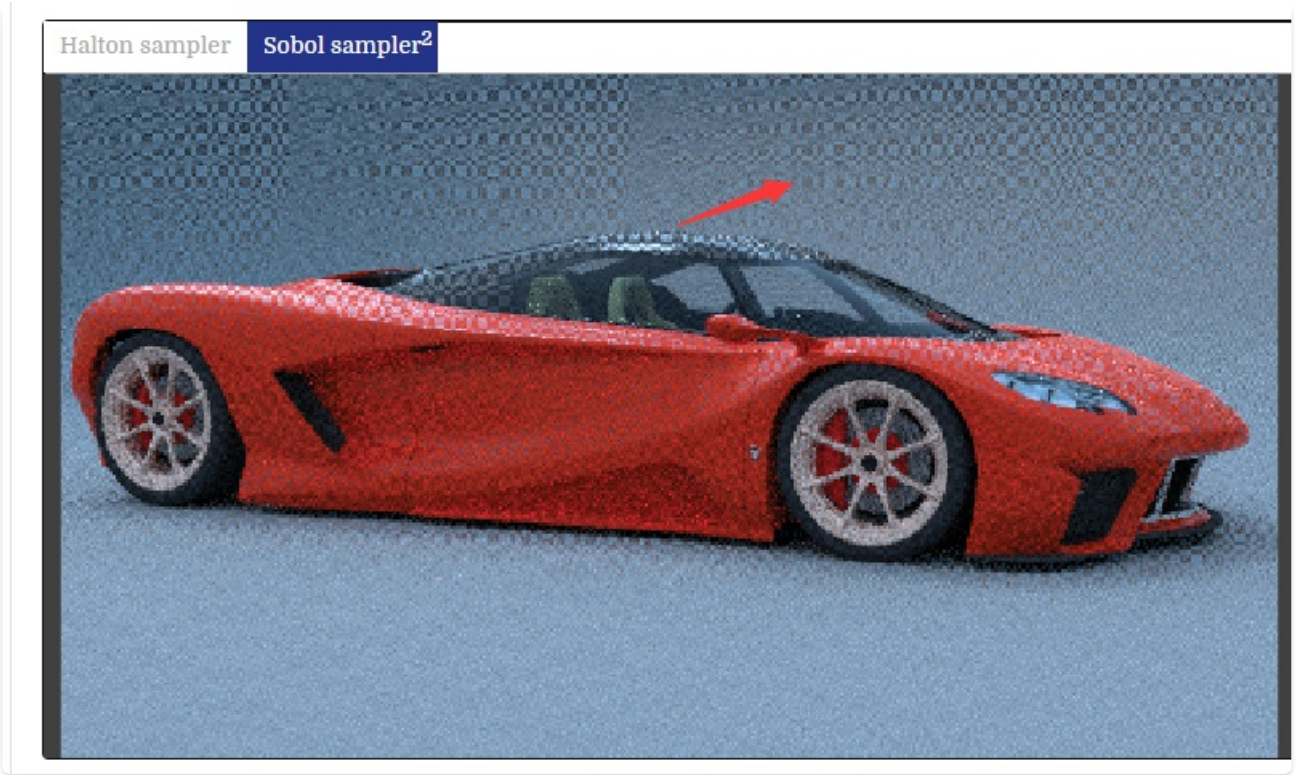
```
45
46  // SobolSampler Declarations
47  // Sobol 采样器
48  // 优点： 低差异分布
49  // 缺点： 容易出现结构化的网格伪像
50  class SobolSampler : public GlobalSampler {
```

2



3

其采样点的分布, 会非常相似



4

Sobol 的结构化伪影

```
54    SobolSampler(int64_t samplesPerPixel, const Bounds2i &sampleBounds)
55        : GlobalSampler(RoundUpPow2(samplesPerPixel)),
56          sampleBounds(sampleBounds) {
57        if (!IsPowerOf2(samplesPerPixel))
58            Warning("Non power-of-two sample count rounded up to %" PRId64
59                    " for SobolSampler.",
60                    this->samplesPerPixel);
61        // 采样的范围，选择是最小能包含 Sample大小 的一个 整数2次幂 的正方形
62        resolution = RoundUpPow2(
63            std::max(sampleBounds.Diagonal().x, sampleBounds.Diagonal().y));
64        log2Resolution = Log2Int(resolution);
65        if (resolution > 0) CHECK_EQ(1 << log2Resolution, resolution);
66    }
67    int64_t GetIndexForSample(int64_t sampleNum) const;
68    Float SampleDimension(int64_t index, int dimension) const;
69
70  private:
71    // SobolSampler Private Data
72    const Bounds2i sampleBounds;
73    int resolution, log2Resolution;
```

5

Sobol, 会用一个尽可能小的, 2的整数次幂为边长的正方形, 将一个采样区域囊括

```cpp
// sobol 的 获得第几个 sample 的 index
// m：幂次，包含了整个采样范围的，2次幂的 幂次（像素范围
// frame：第几个 sample
// p：相对于采样范围起点，我们的位移
// 对于 Sobol 采样，他就是把一个 [0,1) 的范围，放大到了 2^log2Resoultion 这样一个范围，去包含我们的采样范围
// TODO 整个逻辑没看明白
inline uint64_t SobolIntervalToIndex(const uint32_t m, uint64_t frame,
                                     const Point2i &p) {
    if (m == 0) return 0;

    const uint32_t m2 = m << 1;
    uint64_t index = uint64_t(frame) << m2;
    // 这里拿到的 index，在后 2*m 的位置上，都是 000000，这一点，保证了在 [2^m, 2^m] 这个映射范围内，前 2^(2*m) 个，是恰好映射的
    // 多余的 index 信息呢，记录在 64-2*m 的位置里

    // 输入是 frame，也就是 第几个样本
    // 使用的矩阵是 VdCSobolMatrices 这个矩阵，只会对 m 位做施加影响，不会对高次位 有影响
    uint64_t delta = 0;
    for (int c = 0; frame; frame >>= 1, ++c)
        if (frame & 1)  // Add flipped column m + c + 1.
            delta ^= VdCSobolMatrices[m - 1][c];

    // flipped b
    // (((uint64_t)((uint32_t)p.x) << m) | ((uint32_t)p.y)) 这里，前 32 位是 x 的信息，后 32 位是 y 的信息
    uint64_t b = (((uint64_t)((uint32_t)p.x) << m) | ((uint32_t)p.y)) ^ delta;

    // 对 b 做 2 的根逆
    for (int c = 0; b; b >>= 1, ++c)
        if (b & 1)  // Add column 2 * m - c.
            index ^= VdCSobolMatricesInv[m - 1][c];

    return index;
```

6

这一段逻辑没看懂, 但是其内核是一个 SampleNumber 到 Index 的一个映射

```
287  inline Float SobolSample(int64_t index, int dimension, uint64_t scramble = 0) {
288  #ifdef PBRT_FLOAT_AS_DOUBLE
289      return SobolSampleDouble(index, dimension, scramble);
290  #else
291      return SobolSampleFloat(index, dimension, (uint32_t)scramble);
292  #endif
293  }
294
295  inline float SobolSampleFloat(int64_t a, int dimension, uint32_t scramble) {
296      CHECK_LT(dimension, NumSobolDimensions) <<
297          "Integrator has consumed too many Sobol' dimensions; you "
298          "may want to use a Sampler without a dimension limit like "
299          "\"02sequence.\"";
300      // 这个就是一个 2进制版的 对运算做过简化的 一个求随机采样点的算法
301      uint32_t v = scramble;
302      for (int i = dimension * SobolMatrixSize; a != 0; a >>= 1, i++)
303          if (a & 1) v ^= SobolMatrices32[i];
304  #ifndef PBRT_HAVE_HEX_FP_CONSTANTS
305      return std::min(v * 2.3283064365386963e-10f /* 1/2^32 */,
306                      FloatOneMinusEpsilon);
307  #else
308      return std::min(v * 0x1p-32f /* 1/2^32 */,
309                      FloatOneMinusEpsilon);
310  #endif
311  }
```

7

2进制的矩阵, 快速计算的方法

```
48  Float SobolSampler::SampleDimension(int64_t index, int dim) const {
49      if (dim >= NumSobolDimensions)
50          LOG(FATAL) << StringPrintf("SobolSampler can only sample up to %d "
51                                     "dimensions! Exiting.",
52                                     NumSobolDimensions);
53      Float s = SobolSample(index, dim);
54      // Remap Sobol$'$ dimensions used for pixel samples
55      // xy 的取值范围
56      if (dim == 0 || dim == 1) {
57          s = s * resolution + sampleBounds.pMin[dim];
58          s = Clamp(s - currentPixel[dim], (Float)0, OneMinusEpsilon);
59      }
60      return s;
61  }
62
```

8

我们计算出的, 是一个 $[0, 1)$ 的范围, 要把它映射会, 相对于 CurrentPixel 的 xy 位移