Rendering (Signed) Distance Function

Author: [Jacks0n]

Link: [https://zhuanlan.zhihu.com/p/345227118]

(Signed) Distance Function/(有向)距离场可用于隐式表面表达，有粒度小、附带额外几何信息等优点，游戏中常用于体积云、碰撞检测等。本文主要介绍图形学中用于渲染的 SDF 采样方法。

1. Preliminary

图形学中 SDF 常被定义为: $|f(x)| = \inf_{y\in\Omega} \|x-y\|,\ x \in \mathbb{R}^{\mathrm{n}}$，$\Omega$ 为构成所有曲面的点集，当 x 在表面内部时取负，在外部时取正。由于考虑内外，常假设曲面是有向光滑闭曲面。

直观地理解，SDF 描述了以 x 为中心，半径 $|f(x)|$ 的超球体 $S_{|f(x)|}(x)$ 内没有任何曲面的点集。

下面的数学是随便写的，不想看可以跳过。

可见 f 在定义域有: $|f(x)-f(x_0)| \leq \|x-x_0\|$，否则与范数所满足的三角不等式矛盾。 由此有 f 满足 1-Lipschitz 条件，从而是一致连续的。

对于任意 x_0，若有 $y_0 \in \Omega$ 使得 $f(x_0) = \|x_0 - y_0\|$，则有 $v = \partial_{\vec{d}} f|_{x_0+} = \partial_{t+} f(x_0 + t\vec{d}) = -1$ ，其中 $\vec{d} = \frac{y_0 - x_0}{\|y_0 - x_0\|}$。 证明：因 $S_{|f(x)|-t}(x_0 + t\vec{d}) \subset S_{|f(x)|}(x)$，从而 $f(x_0 + t\vec{d}) = \|x_0 + t\vec{d} - y_0\|,\ t \to 0^+$，$|v| = \|\vec{d}\| = 1$，取负。

可见若 f 在 x_0 可微，应有唯一的 $y_0 \in \Omega$ 使得 $f(x_0) = \|x_0 - y_0\|$，且有 $\|\nabla f(x_0)\| = 1$。

实际上 SDF 是程函方程的特例，下面不加边界条件地给出程函方程：

$$|\nabla u(x)| = \frac{1}{g(x)},\ x \in \Omega$$

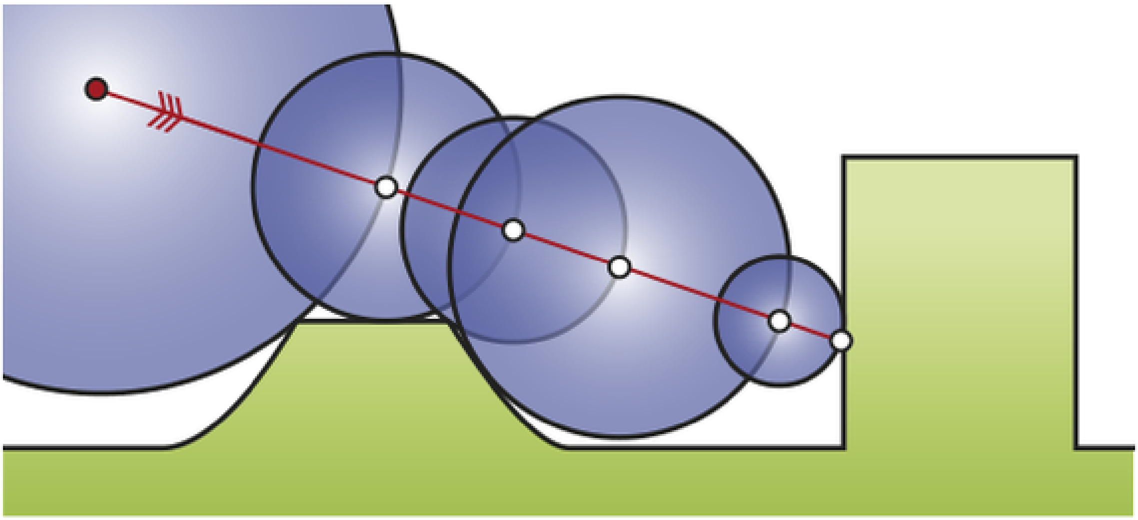
令 $g(x) \equiv 1$ 即可，有大量的数学理论支撑着 SDF 的应用 [^Eikonal]

2. Sampling Techiniques

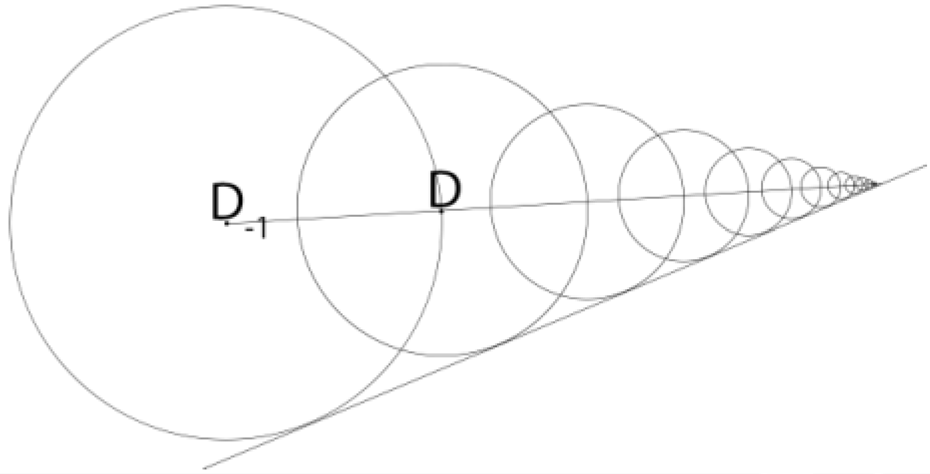
采样方法关注如何高效求解光线与 SDF 的交点，或者: 设光线为 $\delta(t) = o + t\vec{d},\ o, \vec{d} \in \mathbb{R}^{\mathrm{n}}$，即求解 $f(\delta(t)) = 0$

2.1 Sphere Tracing

2.1.1 Naive Techinique



众所周知的采样方法如图中所示，由于球体内不可能有相交点，每次步进 $|f(x_0)|$ 即可，直到命中表面。需要注意由于 $f(\delta)$ 通常无法在有限步收敛至精确解，浮点精度通常采用阈值 ϵ 判定命中，最简单的情况是 ϵ 取 0.001 等固定值。



[^Seb18]

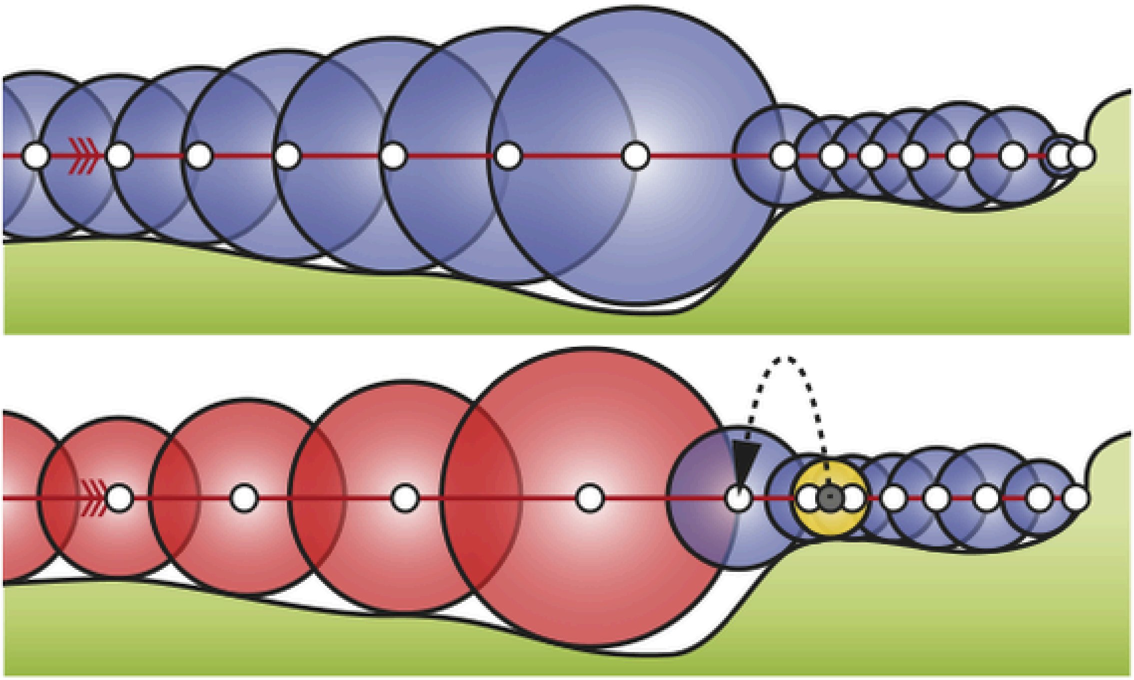
另外部分实现会根据最后两步作修正，即假设最后两步均相接于同一平面，步进构成等比数列，则有修正步进 c:

$$
\frac{c+f(x_n)+f(x_{n-1})}{f(x_{n-1})}=\frac{c+f(x_n)}{f(x_n)}
$$
$$
\Rightarrow\ c=\frac{f(x_n)}{1-(f(x_n)-f(x_{n-1}))}
$$

第一项等式由过最后两圆心向平面作垂线得到的三角形相似得出。

2.1.2 Enhanced Sphere Tracing

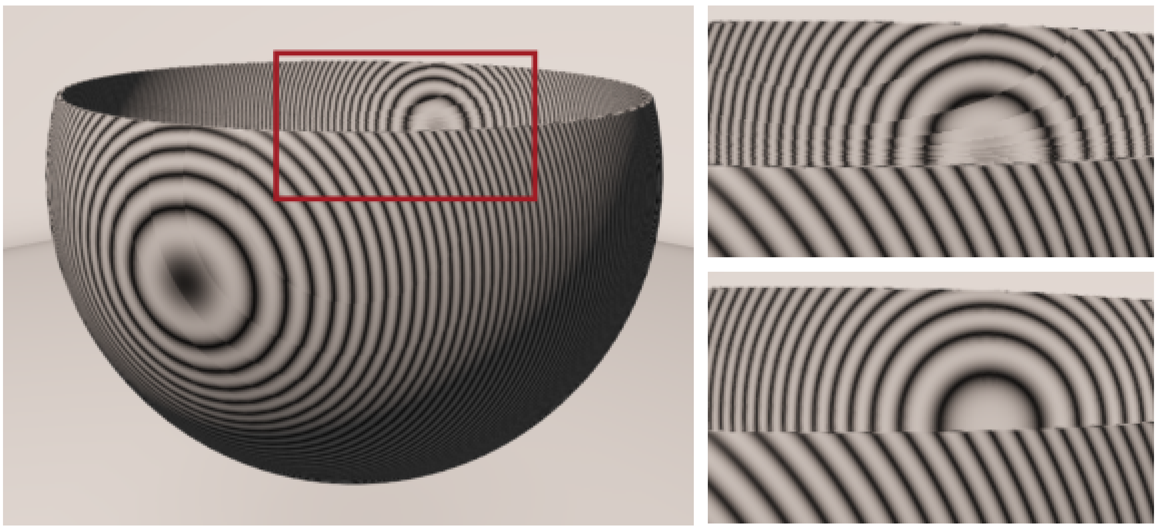
[^KS14] 提出，由于只要求步进时采样的球覆盖光线至交点即可，可以尝试步进 $\omega|f(x_n)|$，其中 $\omega \geq 1$，并判断 $f(x_n)$ 与 $f(x_{n+1})$ 处的球有无交集来判断步进的合理性。由于球心位于光线上，若球体的交集非空必定包含了光线。



[^KS14] 回退图示

判断则是 $|f(x_n)|+|f(x_{n+1})| \geq \omega |f(x_n)|$，理想情况只需一次额外的加法和比较。作者给出通常 $\omega \approx 1.2$ 较好。

这篇文章同时给出了数值连续性的改进。上文中提到的 ϵ 取固定值的方法在采样不同频率的信号下表现较差，在渲染结果上表现为视觉上有断层，一种改进方法是按照一块像素在世界空间的投影大小调整带宽，具体做法是用对采样到的交点（深度）作不动点迭代后处理。



[^KS14] 100 倍放大

右上未作后处理，右下为计算 3 次迭代的结果。

考虑 $f(x)=err(\|x-o\|)$，$x \in \mathbb{R}^n$，$err: \mathbb{R} \to \mathbb{R}$ 应在一定范围足够小，即充当 ϵ，则该等式的解之一就是与正确交点误差为 $err(\|x-o\|)$ 的点。 那么有迭代：
$p_{i+1}=p_i+\vec{d}(f(p_i)-err(\|p_i-o\|))$ 也可以写为：

$$\begin{aligned} &\delta(t_{i+1}) &&= \delta(t_i) + \vec{d}(f(\delta(t_i)) - err(t_i)) \\ \Leftrightarrow\ &t_{i+1} &&= t_i + f(\delta(t_i)) - err(t_i) \end{aligned}$$

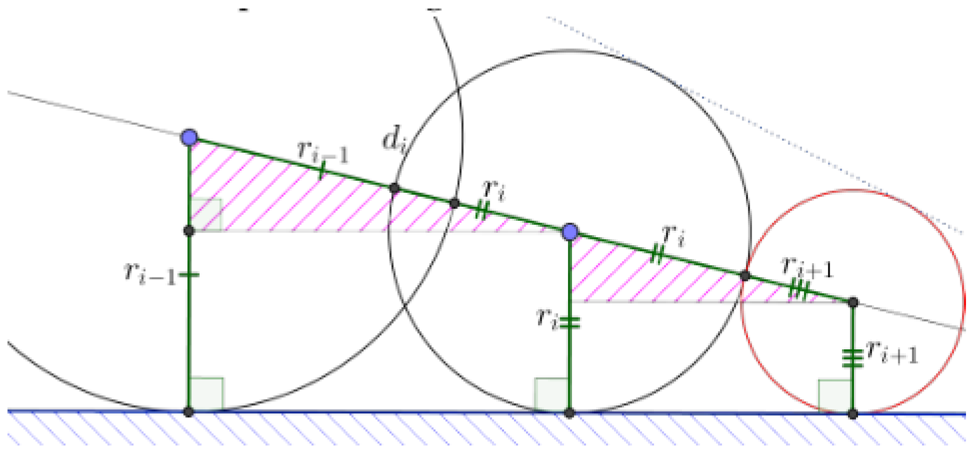
对右边求导，有 $1 + \nabla f(\delta(t)) \cdot \delta'(t) - err'(t)$，$\nabla f(\delta(t))$ 近似为交点的法线 $\vec{n}$，则有 $1 + \vec{n} \cdot \vec{d} - err'(t)$，$\vec{n}$ 与 $\vec{d}$ 反向，从而该迭代在 $|err'(t)|$ 足够小时线性收敛。

对于针孔相机模型，为了覆盖一块像素，可以取 $err(t) = \tan(\frac{\text{FOV}}{2}) \cdot \text{texel} \cdot t$，其中 texel 为纹素大小，即 $\frac{1}{\text{Resolution}}$（假设相机的 Aspect Ratio 与 Render Target 的长宽比相同）。

事实上，直接用迭代公式计算 t 也可以，适用于渐进式渲染（TAA 就完事了），但游戏中不太需要如此的数值稳定性，在分形渲染中较为有用。

2.1.3 More Tracing Acceleration

[^BV18] 提出，可以假设连续的两个步进球体相切于同一平面，下一步尝试步进相切的球，失败则回退至保守步进。



[^BV18]

从图中由三角形相似 $\frac{d_i}{r_i + r_{i+1}} = \frac{r_{i-1} - r_i}{r_i - r_{i+1}}$，从而：$d_{i+1} = r_{i+1} + r_i = r_i \cdot \frac{2d_i}{d_i + r_{i-1} - r_i}$，其中 d_i 是步进长度。 虽然涉及除法，但比 [^KS14] 回退次数少了很多，总的来讲还是有加速。

补充：在 GPU 实现中，cache miss 是更易导致性能问题，因此不常用此类"过步进"算法。

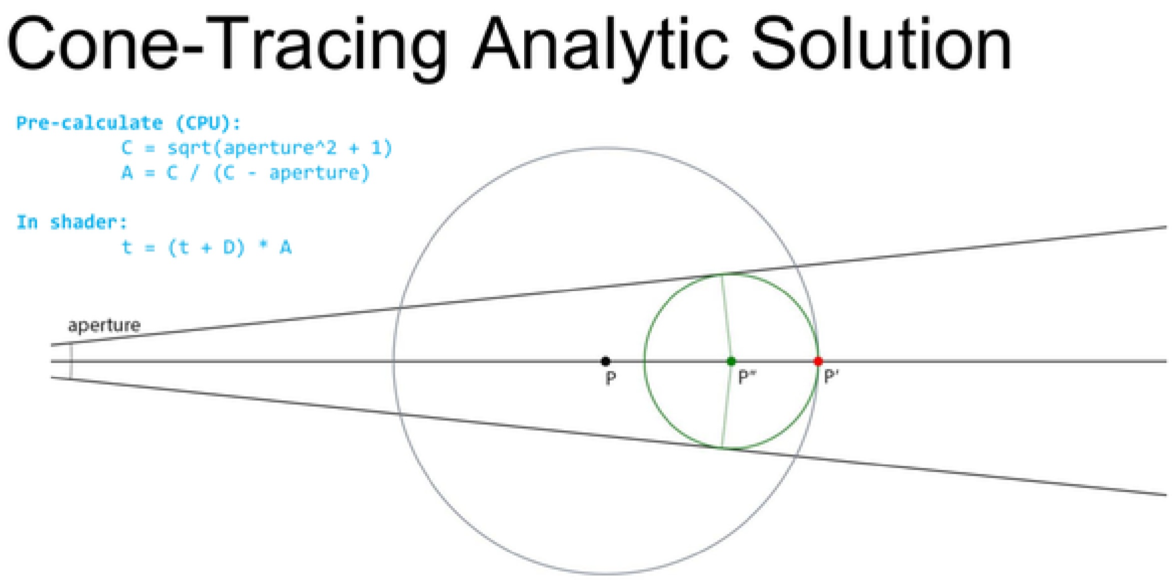
2.1.4 Hierarchical Acceleration

无论用哪种步进，在步进次数较少时加速不理想，在游戏等实时渲染中步数受限时用 naive 就足够了，不过可以对其做出改进。

第一种是先粗略地 Per Tile 采样再 Per Pixel 采样。对于大多数 Per Tile 策略，Tile 不宜过大或过小，通常选择 8x8。 注：在 GPU 实现时也应考虑目标平台的 Compute Pipeline 特性。

[^Seb18] 提到了圆锥采样：比起光线的半直线，一次性步进投影能够覆盖 Tile 的球体（圆锥），Per Pixel 时采样对应 Tile 计算出起点继续步进。

接下来要算出给定 $f(x_n)$ 时球体可以向前步进的距离。考虑下图，有 $t_{P'} = t_P + f(\delta(t_P))$，现在要计算 $t_{P''}$；由于以 P'' 为圆心的圆与圆锥和以 P 为圆心的圆内接，可以推到得到 $t_{P'}/t_{P''}$ 是由给定圆锥确定的常数，可以在 CPU 提前计算。总得来讲比光线步进每次只多一次乘法。



最后 Per Pixel 计算时若精确计算起点涉及超越函数计算，可以为 Tile 预计算或快速地近似，只要不超出球体就好。（个人猜测预计算三角函数性能更好也更简洁）

更正：Per Pixel 计算时计算光线和球体交点即可。

注：描述一个球（假设为 3 维）需要位置 4 个数，但光线（这里是圆锥的轴）与贴图空间坐标有双射关系，只需要存储深度（光线长度 t）和半径即可。

第二种是 mipmap。这里偏下题，在实际应用中，受限于显存，有时用 16bit 或 8bit 存储，并限制步进上界，这导致就算有一大片空区域，也只能按照上界一点点步进，因此需要按不同上界构建 mipmap。

mipmap 的构建不能像纹理卷积得到，而是要在 mipmap 的纹素中心扩大区域上界重新计算，当然可以离线计算，但 [^Seb18] 提到可以通过数值求解程函方程得到满意的结果。在已知 mip n-1 的情况，不断在 mip n 的体素格点计算即可。详情见 [^Eikonal] 和 Aaltonen Sebastian 的 **https://zhuanlan.zhihu.com/p/345227118/www.shadertoy.com/view/MtK3zD**。

2.2 Binary Search

由于 SDF 的符号在内外相反，可以二分查找 $f(\delta(t))$ 的零点，由于不常用不讨论。

2.3 Segment Tracing

这种方法通过导数获取额外信息，已经有聚聚介绍过了，http://zhuanlan.zhihu.com/p/266747723，本文再解释一次。

考虑 SDF 满足的：$|f(x)-f(x_0)| \leq \|x-x_0\|$，这只是在定义域上满足的，局部可以有更强的约束：$|f(x)-f(x_0)| \leq \lambda\|x-x_0\|,\ 0 \leq \lambda \leq 1$

又有：$|f(\delta(t))-f(\delta(t_0))| \leq \lambda|t-t_0| = \lambda\Delta t,\ 0 \leq \lambda \leq 1$

而问题是求解 $f(\delta(t^\star))=0$，若能知道 $t \in [t_0, t^\star]$ 时使不等式成立的 $\lambda_{\max}$ 就可以一次性步进 $\Delta t = \frac{|f(\delta(t_0))-f(\delta(t^\star))|}{\lambda_{\max}} = \frac{|f(\delta(t_0))|}{\lambda_{\max}}$ 而保证不错过零点，由于 $\lambda \leq 1$，步进距离至少和 sphere tracing 一致，[^GG20] 全文都在讨论各种 SDF 的 Lipschitz 常数的估计方法。

Digression

既然用到了导数，可以联想到牛顿法等方法，毕竟要解决的是求零点问题。

另外论文中讨论的是 $g(f(\delta(t)))$，多出来的 $g(x):\mathbb{R}\to\mathbb{R}$ 被作者称为 fall-off function，这里简单解释下它的意义：[^WMW86] 最早提出用一种场函数 $f(x)$ 描述软体，具体做法是在场景设置几个点 $\{p_n\}$，每个点有其影响范围 $\{r_n\}$，则 $f(x)=g(\frac{\|x-p_i\|}{r_n})$，其中 r_i，p_i 取最近的点，所以相当于渲染 $f(x)$ 的某个等值面。这里取与最近点的的距离相当于 SDF，从而有 $g(f(\delta(t)))$，可见经过映射后不一定满足先前的 SDF 定义，但 Segment Tracing 只考虑 Lipschitz 条件，从而可以应用在这种情况。由于略去影响不大下文中将不考虑 $g(x)$。

Back to Topic

首先有 $\lambda\leq\sup_{t\in[t_0,t_0+\Delta t]}(f(\delta(t)))'\leq\sup_{t\in[t_0,t^*]}(f(\delta(t)))'$，可以用该上界估计替代 λ，注意这里 λ 是相对于一段区间而言。而 $(f(\delta(t)))'=\nabla f(\delta(t))\cdot\delta'(t)=\nabla f(\delta(t))\cdot\vec{d}$，如何估计这项导数的上界就是重点。

对有简单解析形式的 SDF，如球体、多面体等，和有简单解析形式的 SDF "变形"函数（也就是算子），可以直接计算梯度与光线方向的内积，或者估计一个球体内的雅可比行列式，或者估计一个球体内的雅可比矩阵的谱半径，推导很繁琐这里不给出具体例子。

由于对 λ 的估计大多保守，步进距离 $\frac{f(t_0)}{\lambda}$ 可以乘上系数 κ 适当多步进一些，作者给出 $\kappa\approx 2$ 时较好。

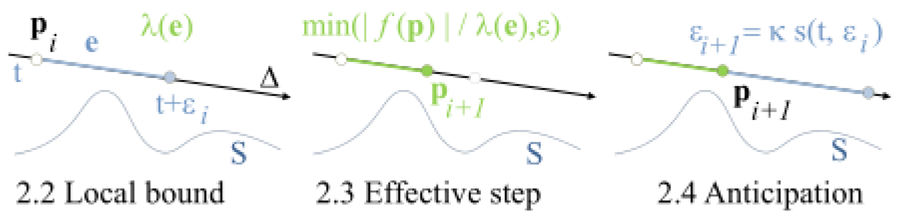


Figure 3: *Overview of the main steps of the algorithm.*

估计上界；计算步进；Anticipation（预测步进）

对于体素中存储的离散化 SDF，估计 $\sup_{t\in[t_0,t_0+\Delta t]}(f(\delta(t)))'$ 是不现实的，那么要额外存储局部的 Lipschitz 常数上界，或者计算局部梯度作为估计等，具体做法仍需商讨。

这种方法也可以应用在 Per Tile 的步进，原文没有提及，推导以后再说。

2.4 Gradient Sampling

怎么采样梯度（法线）呢，见 [^Iq]。

3. 结语

这里介绍的只是 SDF 的一部分，没有提到具体解决方案、SDF 离散计算、加速结构、碰撞检测等，有心情再写。

如果你在清华大学，欢迎来未来动漫游戏社团找我交流图形学和游戏开发 :/

4. 引用

[^Eikonal]: http://en.wikipedia.org/wiki/Eikonal_equation
[^BV18]: Csaba B., Gábor V., Accelerating Sphere Tracing.
[^GG20]: Eric Ga., Eric Gu., Segment Tracing Using Local Lipschitz Bounds.
[^Iq]: http://www.iquilezles.org/www/index.htm
[^KS14]: Benjamin K., Henry S., Enhanced Sphere Tracing.
[^Seb18]: Aaltonen S., GPU-based clay simulation and ray-tracing tech in Claybook.
[^WMW86]: Wyvill G., McPheeters C., Wyvill B.: Data structure for soft objects.