中山大學 软件工程学院

SUN YAT-SEN UNIVERSITY SCHOOL OF SOFTWARE ENGINEERING

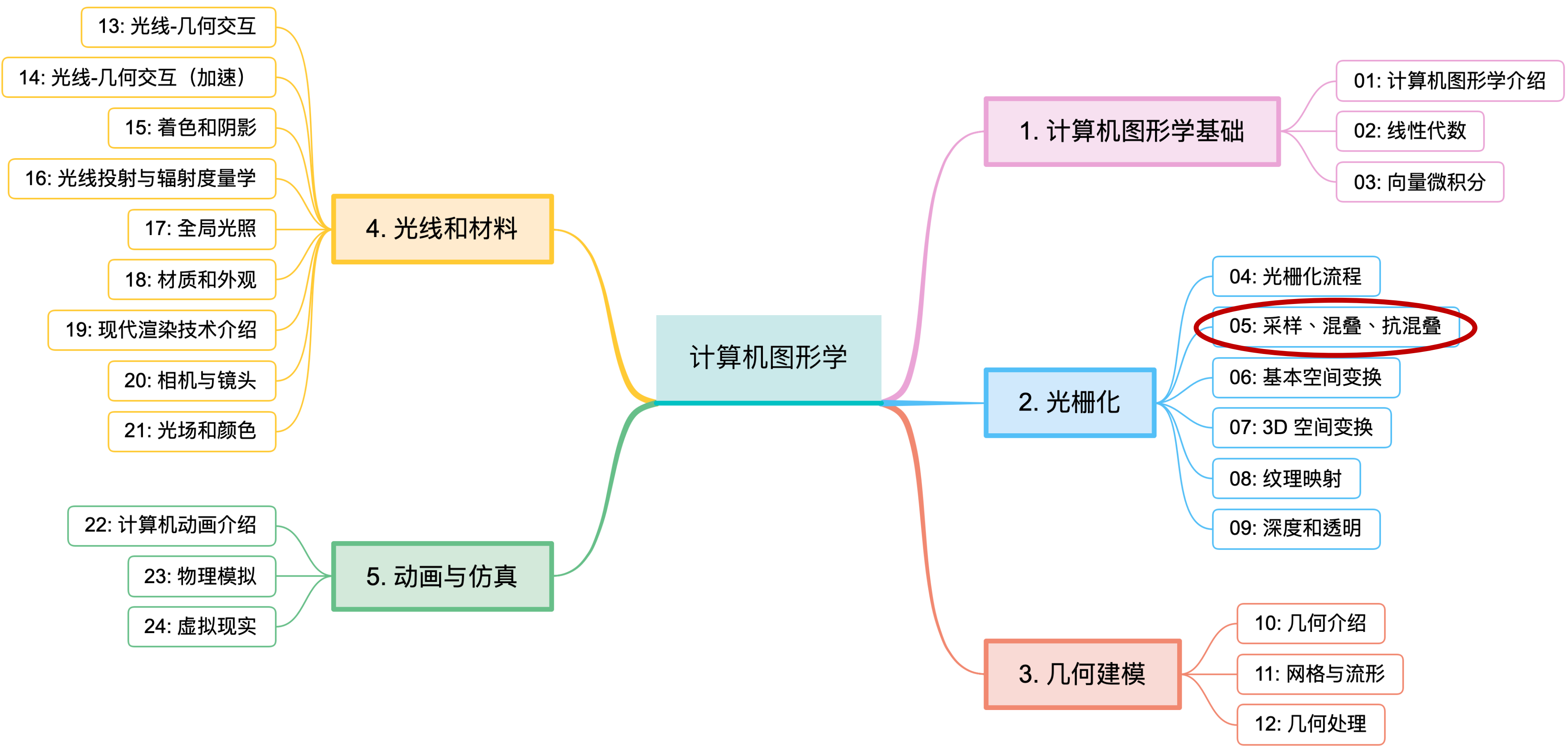
Lecture 05：采样与走样

SSE315：计算机图形学
Computer Graphics

陈壮彬

软件工程学院

chenzhb36@mail.sysu.edu.cn



计算机图形学
1. 计算机图形学基础
01: 计算机图形学介绍
02: 线性代数
03: 向量微积分
2. 光栅化
04: 光栅化流程
05: 采样、混叠、抗混叠
06: 基本空间变换
07: 3D 空间变换
08: 纹理映射
09: 深度和透明
3. 几何建模
10: 几何介绍
11: 网格与流形
12: 几何处理
4. 光线和材料
13: 光线-几何交互
14: 光线-几何交互（加速）
15: 着色和阴影
16: 光线投射与辐射度量学
17: 全局光照
18: 材质和外观
19: 现代渲染技术介绍
20: 相机与镜头
21: 光场和颜色
5. 动画与仿真
22: 计算机动画介绍
23: 物理模拟
24: 虚拟现实

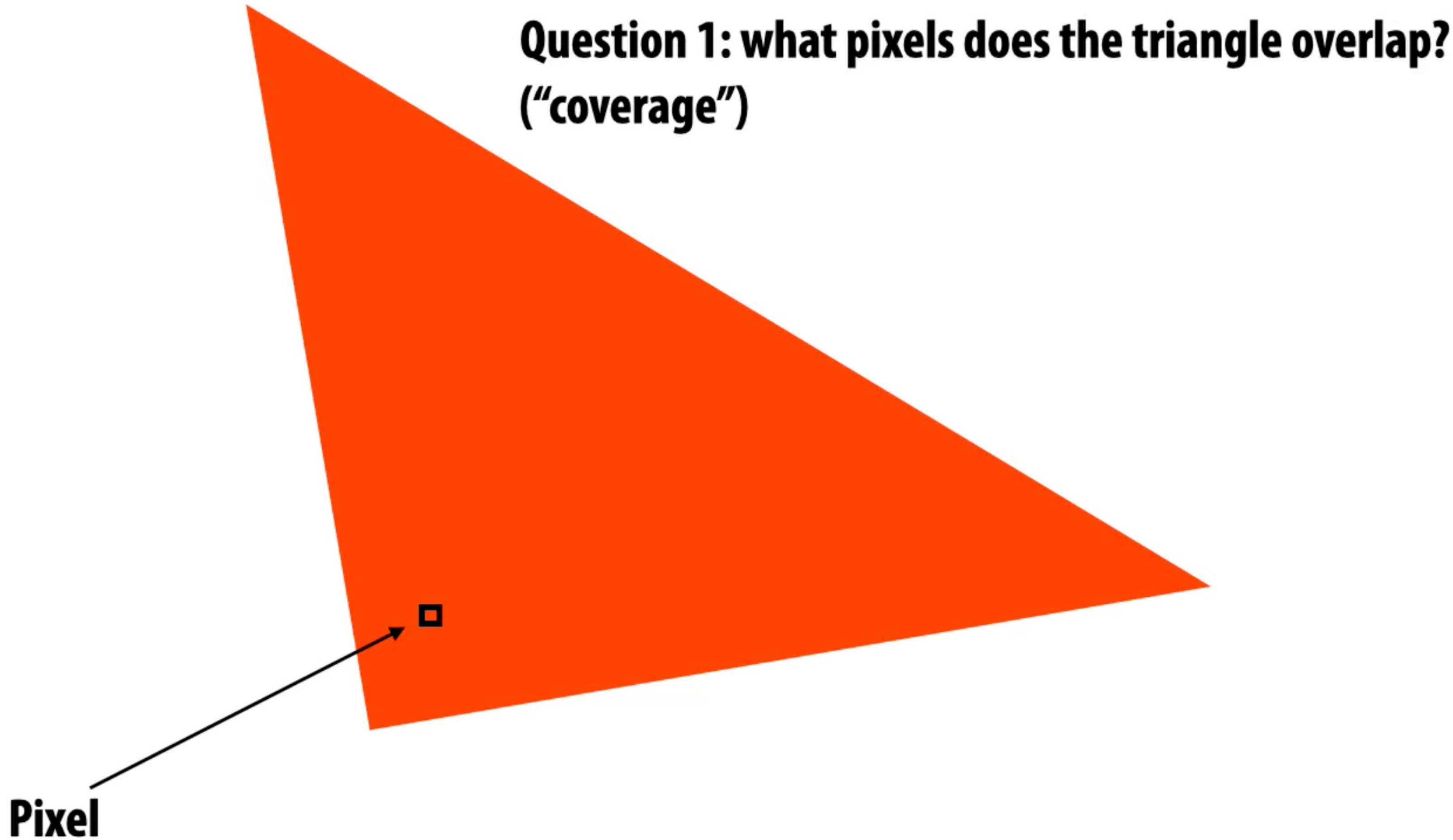
Today’s topics

□采样 Sampling

□走样 Aliasing

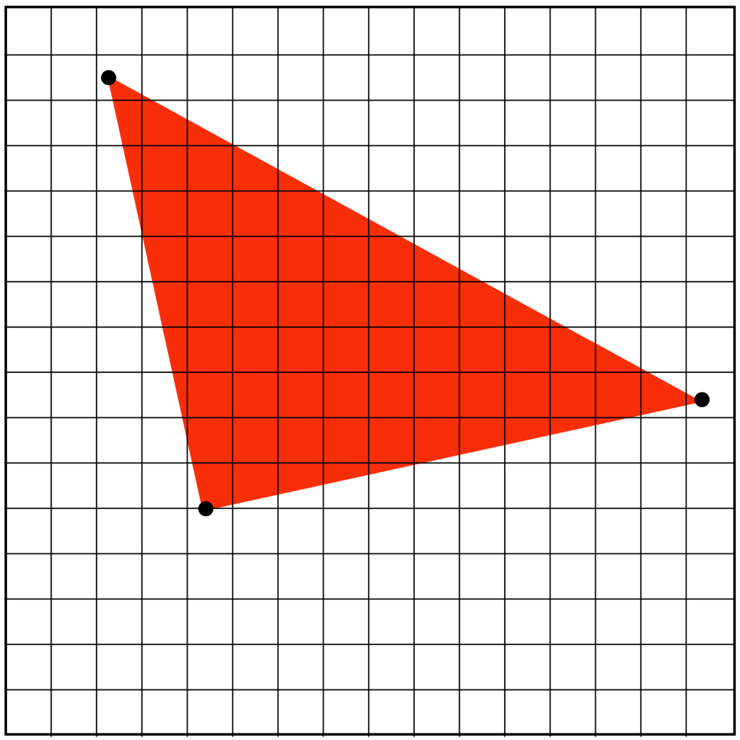
□抗走样 Antialiasing

基于采样的方式画三角形

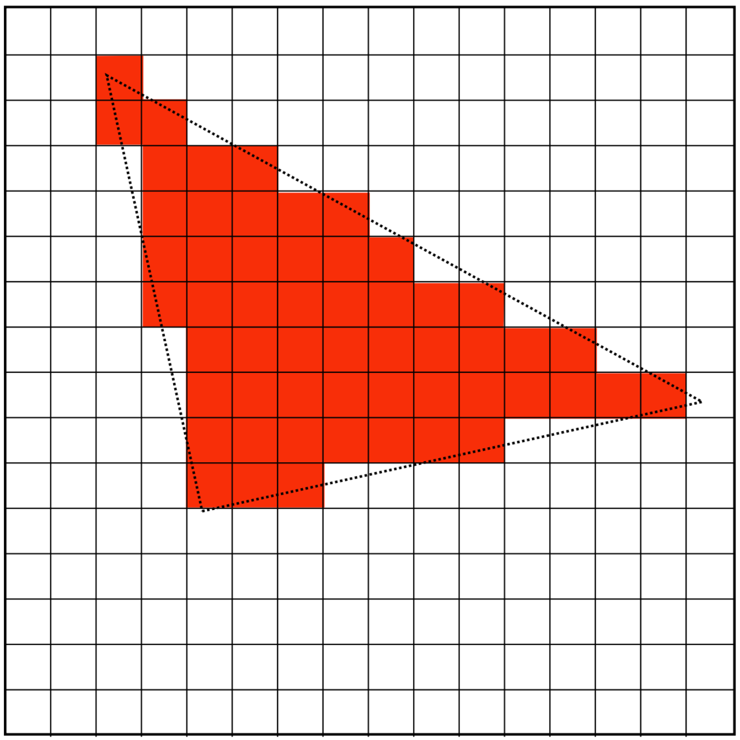


计算哪些像素被三角形覆盖

Input:
projected position of triangle vertices: P_0, P_1, P_2



Output:
set of pixels "covered" by the triangle

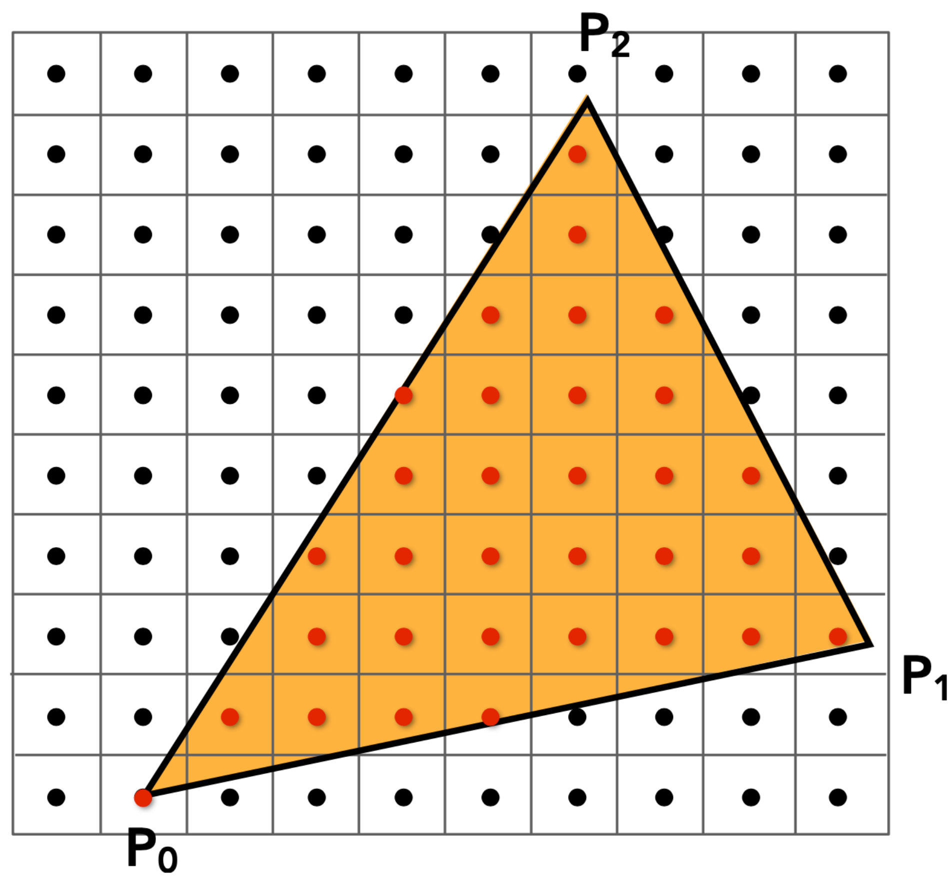


三角形内点测试：三线测试

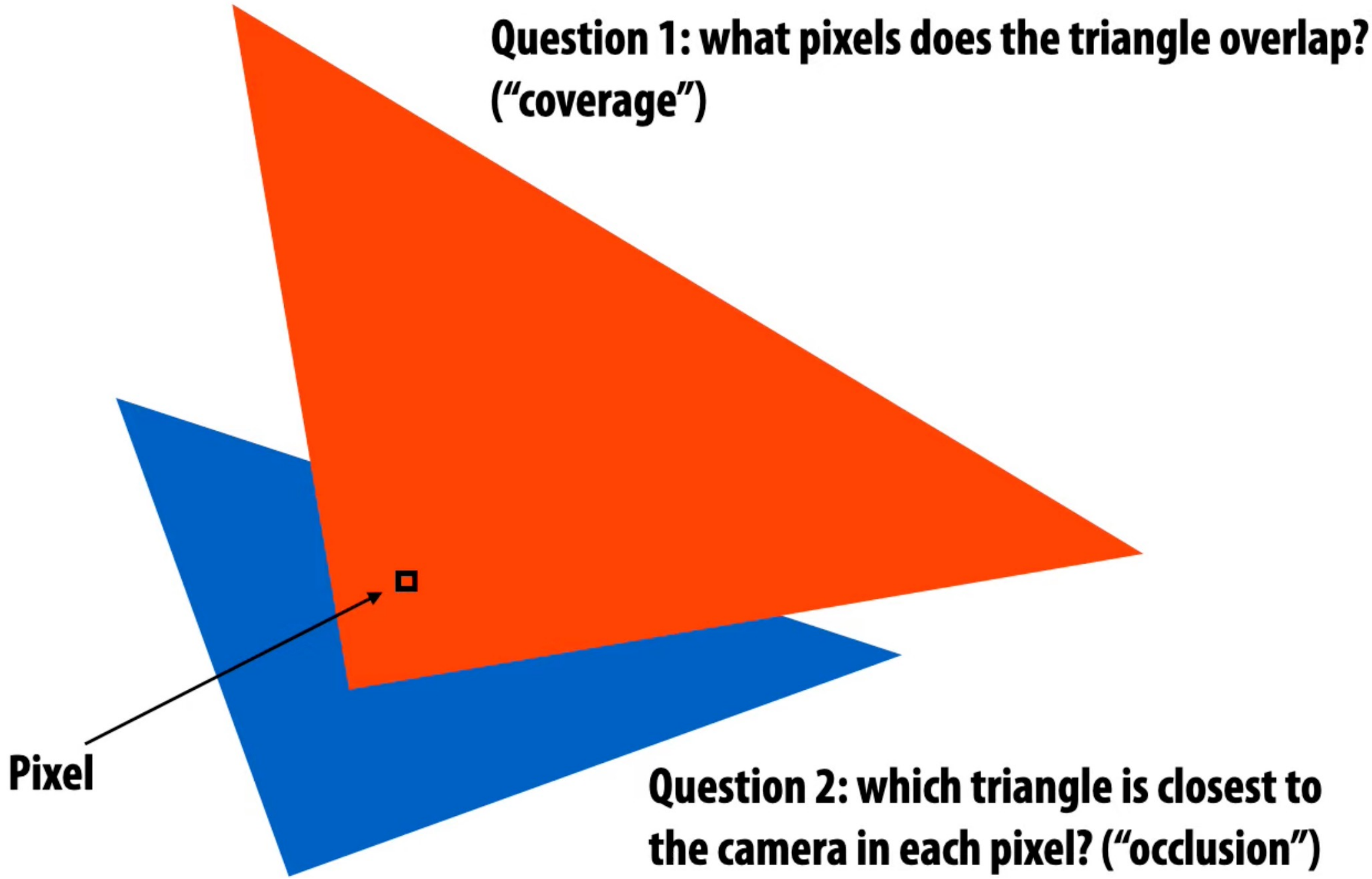
Sample point $s = (sx, sy)$ is inside the triangle if it is inside all three lines.

$$
\begin{aligned}
&inside(sx, sy) = \\
&\quad L_0(sx, sy) > 0 \;\&\& \\
&\quad L_1(sx, sy) > 0 \;\&\& \\
&\quad L_2(sx, sy) > 0;
\end{aligned}
$$

Note: actual implementation of $inside(sx,sy)$ involves ≤ checks based on edge rules

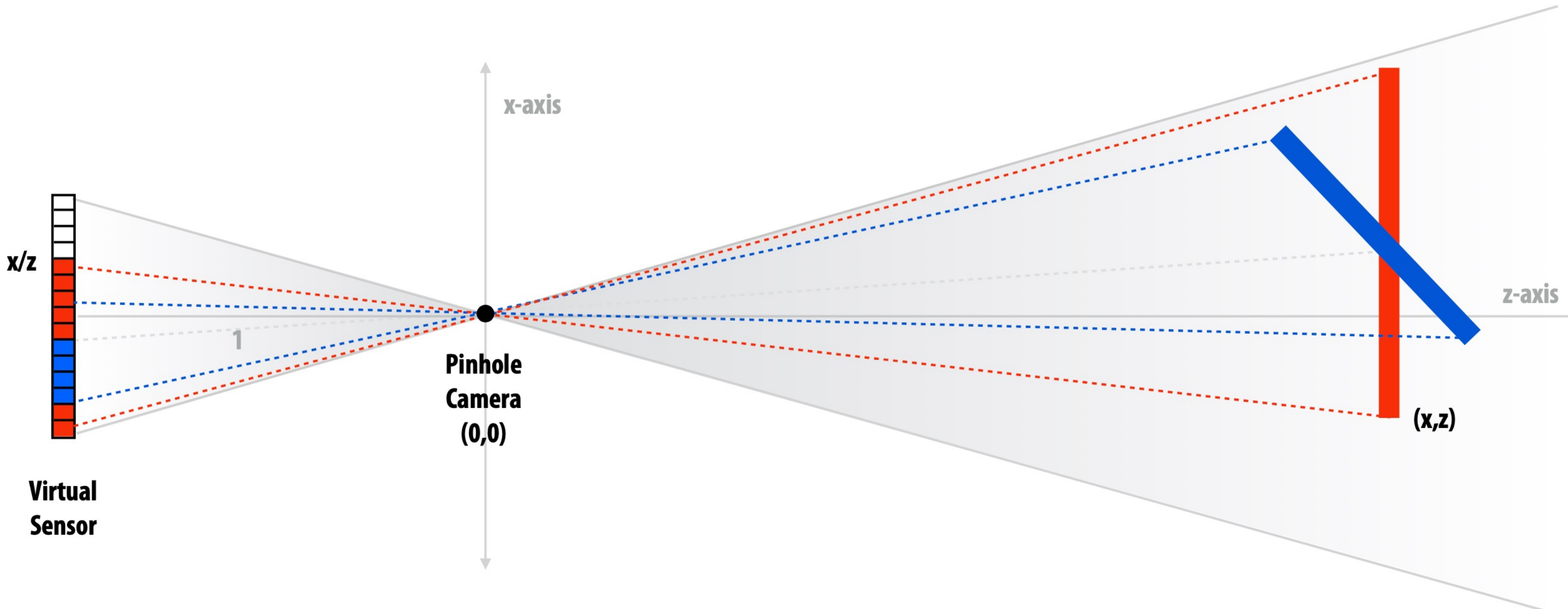


两个 (或多个) 三角形呢?


Question 1: what pixels does the triangle overlap?
("coverage")
Pixel
Question 2: which triangle is closest to
the camera in each pixel? ("occlusion")

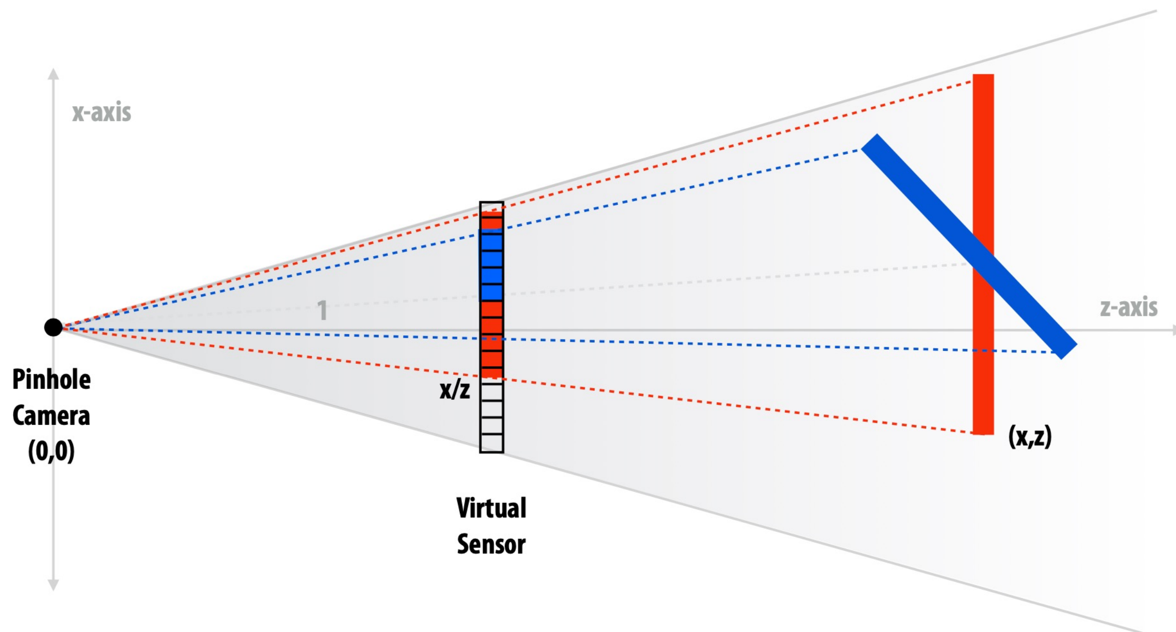
可见性 visibility 问题

□回顾针孔相机模型



可见性问题

□我们可以将其简化为一个虚拟传感器模型…

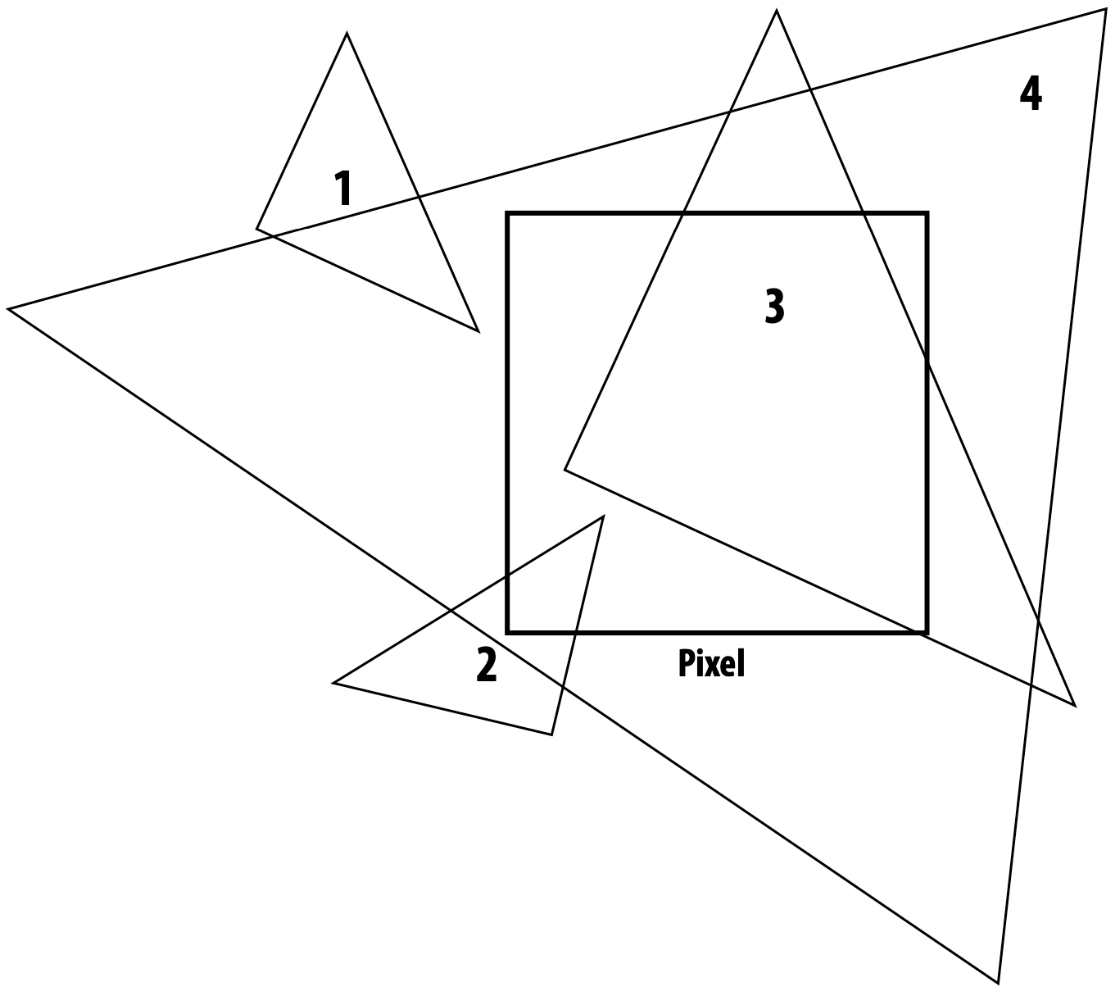


□光的可见性问题

- 覆盖 Coverage：哪个几何物体被透过针孔的光线打中？
- 遮挡 Occlusion：哪个物体最先被光打中？

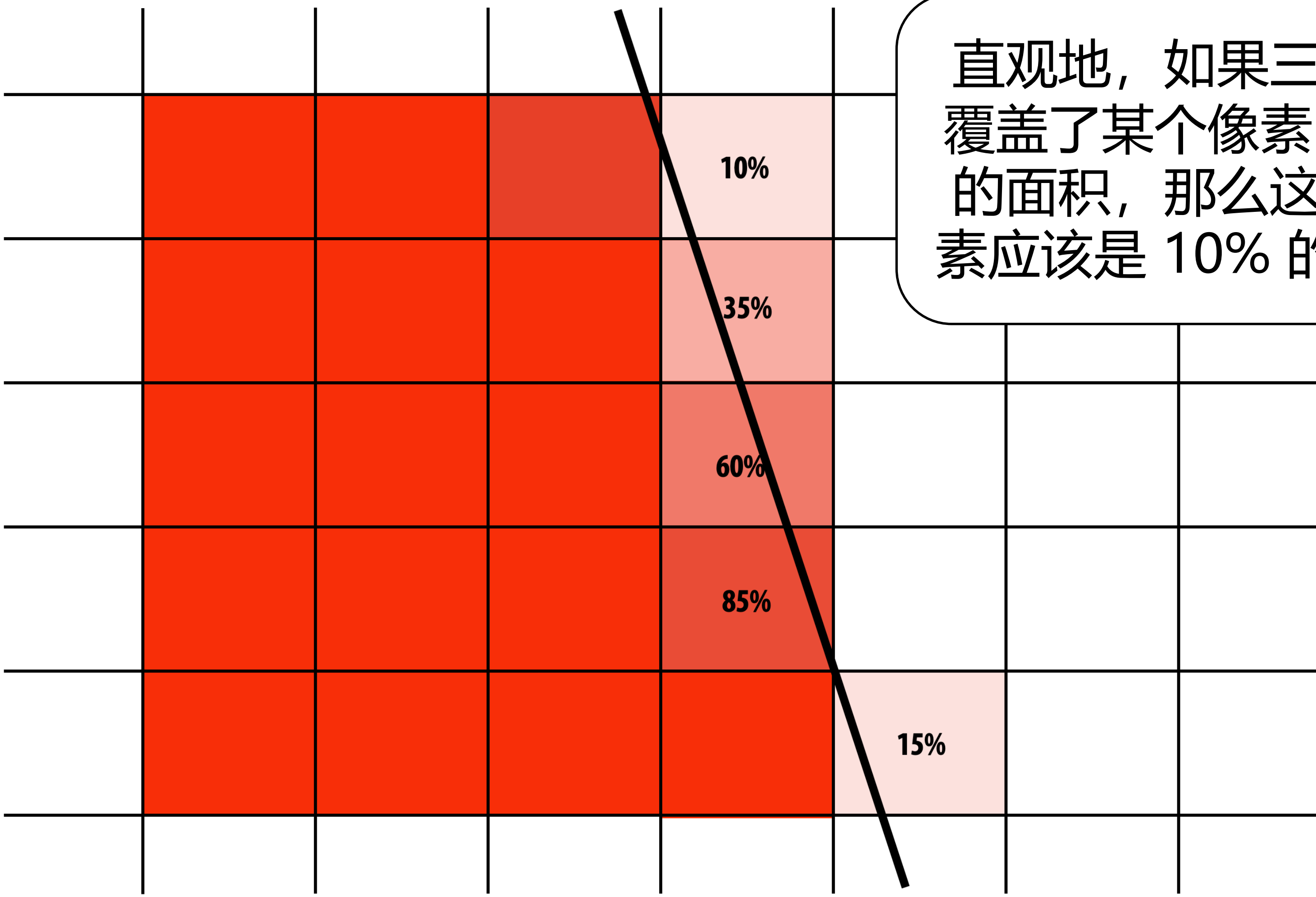
三角形覆盖问题

Q: Which triangles "cover" this pixel?



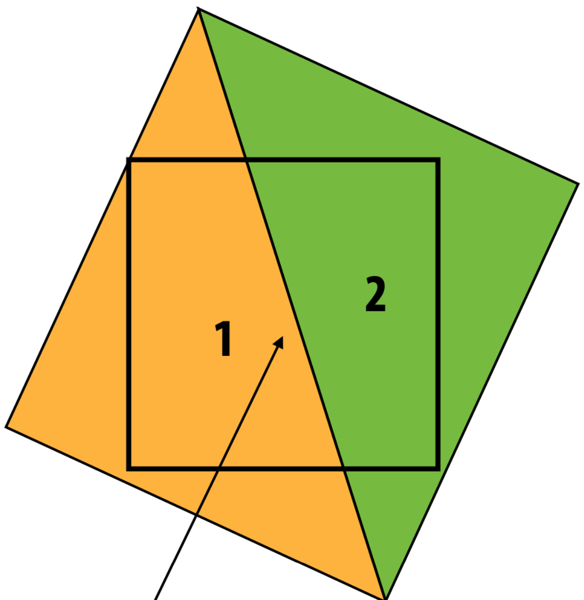
三角形覆盖问题

- 一种选择：计算三角形覆盖的像素面积的百分比，然后根据这个百分比对像素进行着色

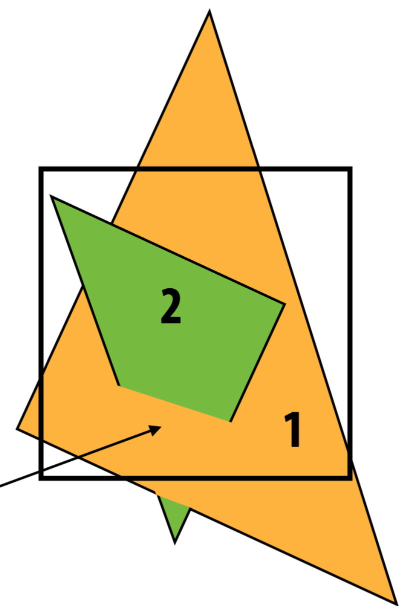


直观地，如果三角形覆盖了某个像素 10% 的面积，那么这个像素应该是 10% 的红色

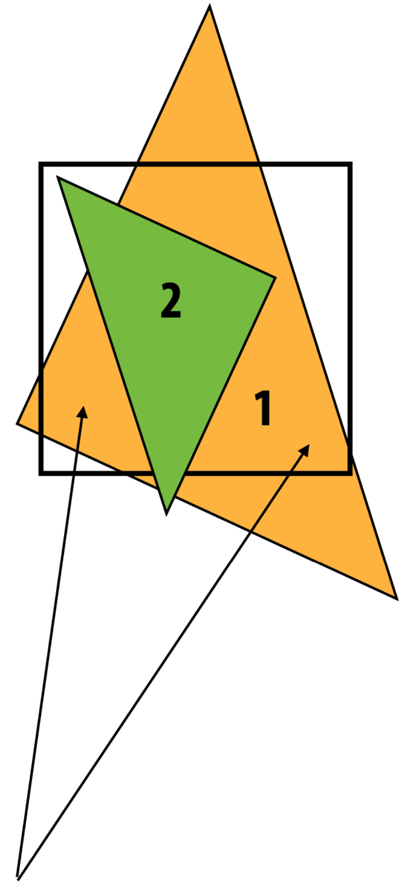
遮挡使覆盖变得更复杂



Pixel covered by triangle 1, other half covered by triangle 2



Interpenetration of triangles: even trickier

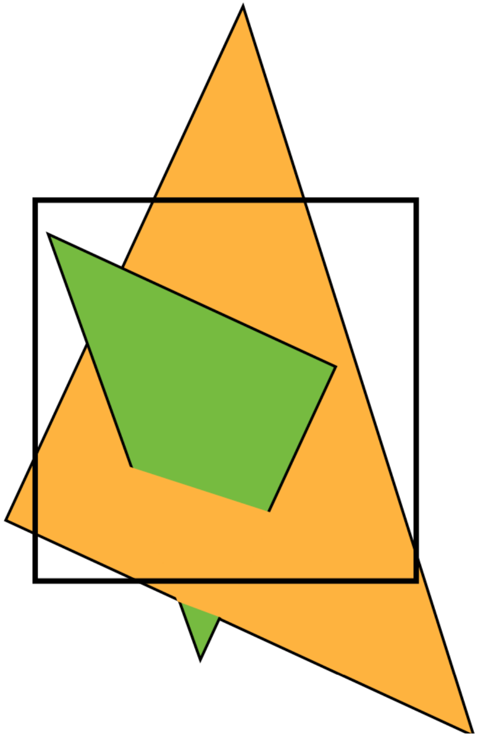


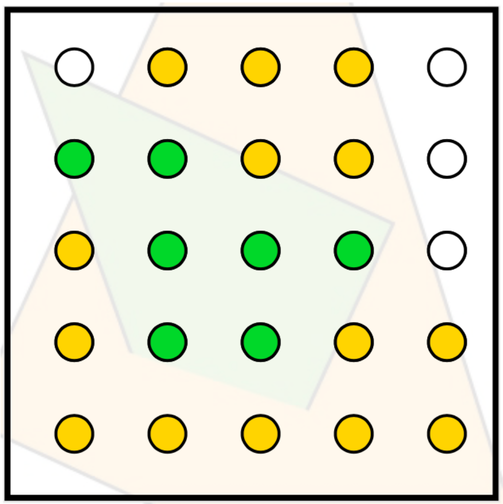
Two regions of triangle 1 contribute to pixel. One of these regions is not even convex.

如何估计不同三角形覆盖的百分比?

基于采样的覆盖

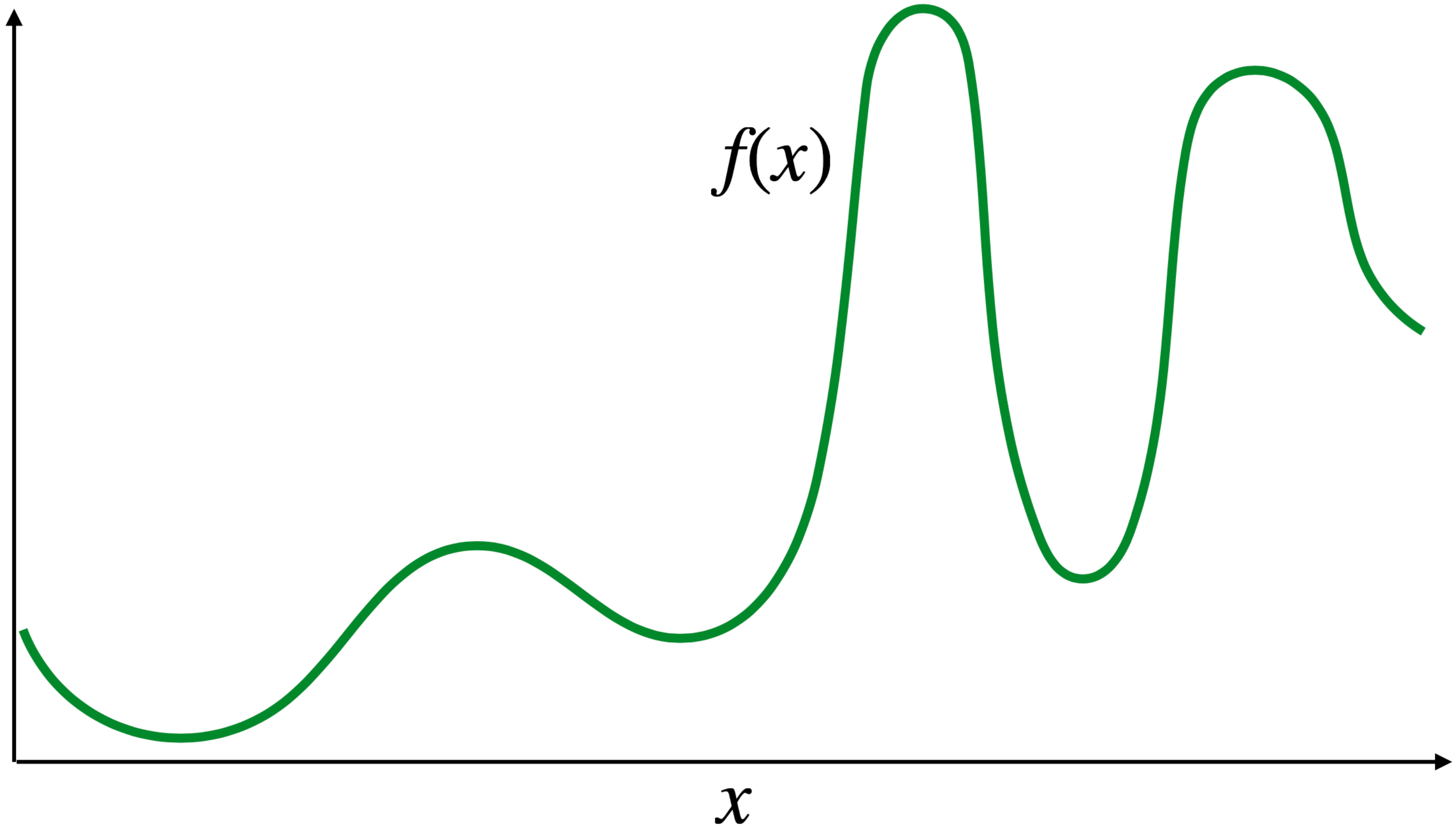
- 真实的场景很复杂！
  - 遮挡 occlusion 、透明度 transparency…
  - 以后将介绍更多
- 计算准确的覆盖率是不现实的
- 相反：将覆盖率视为采样问题
  - 不要计算 exact/analytical answer
  - 相反地，测试一组采样点
  - 有了足够的样本点及合适的位置选择，就可以开始得到一个很好的估计
- **首先，让我们先系统地介绍采样**





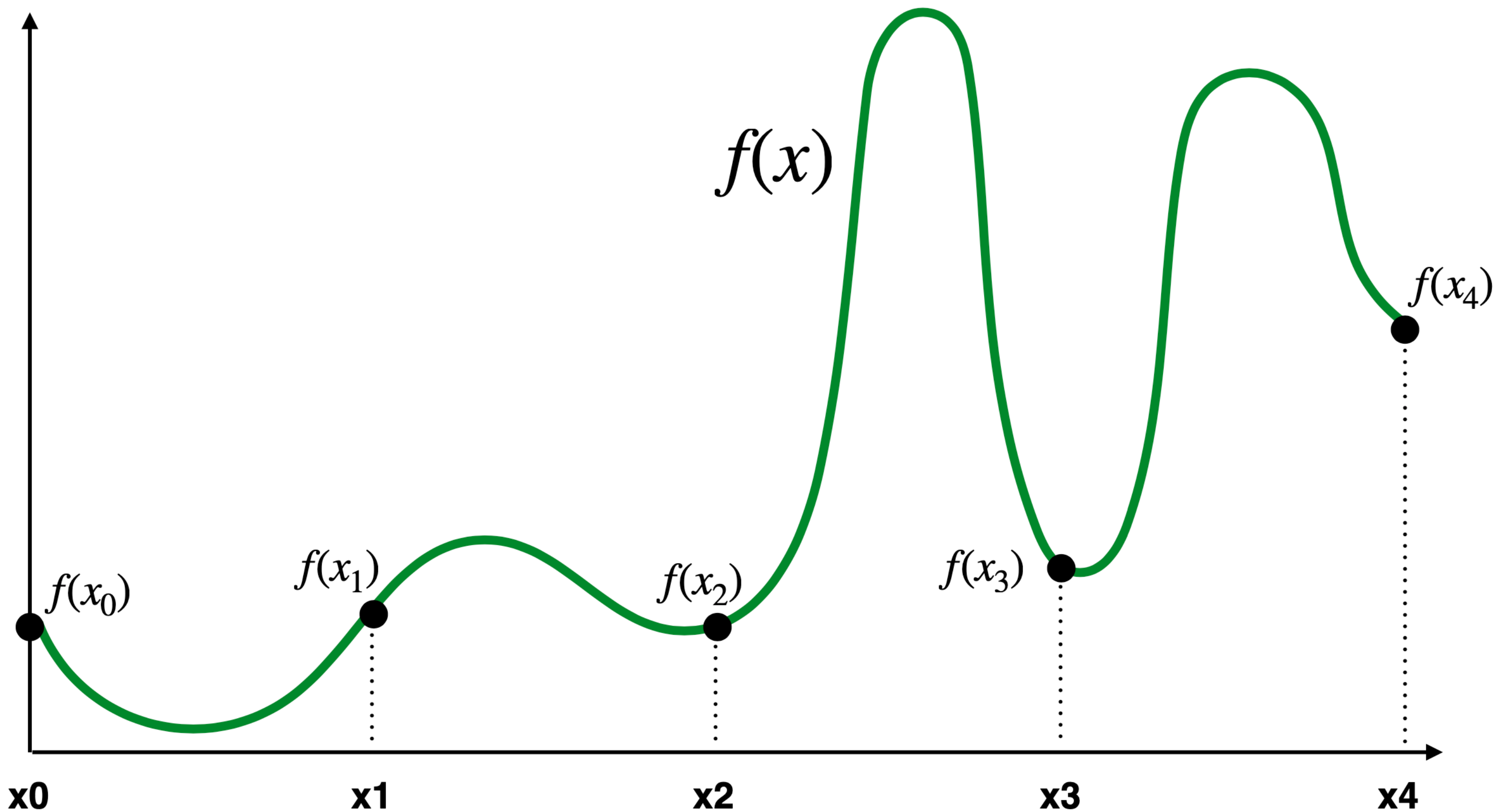
采样 101：采样 1D 的信号

□采样 Sampling

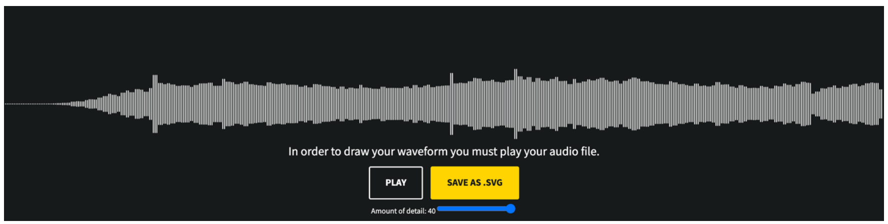


采样 = 对信号进行测量

- 以下为 $f(x)$ 的 5 个测量点（采样点）



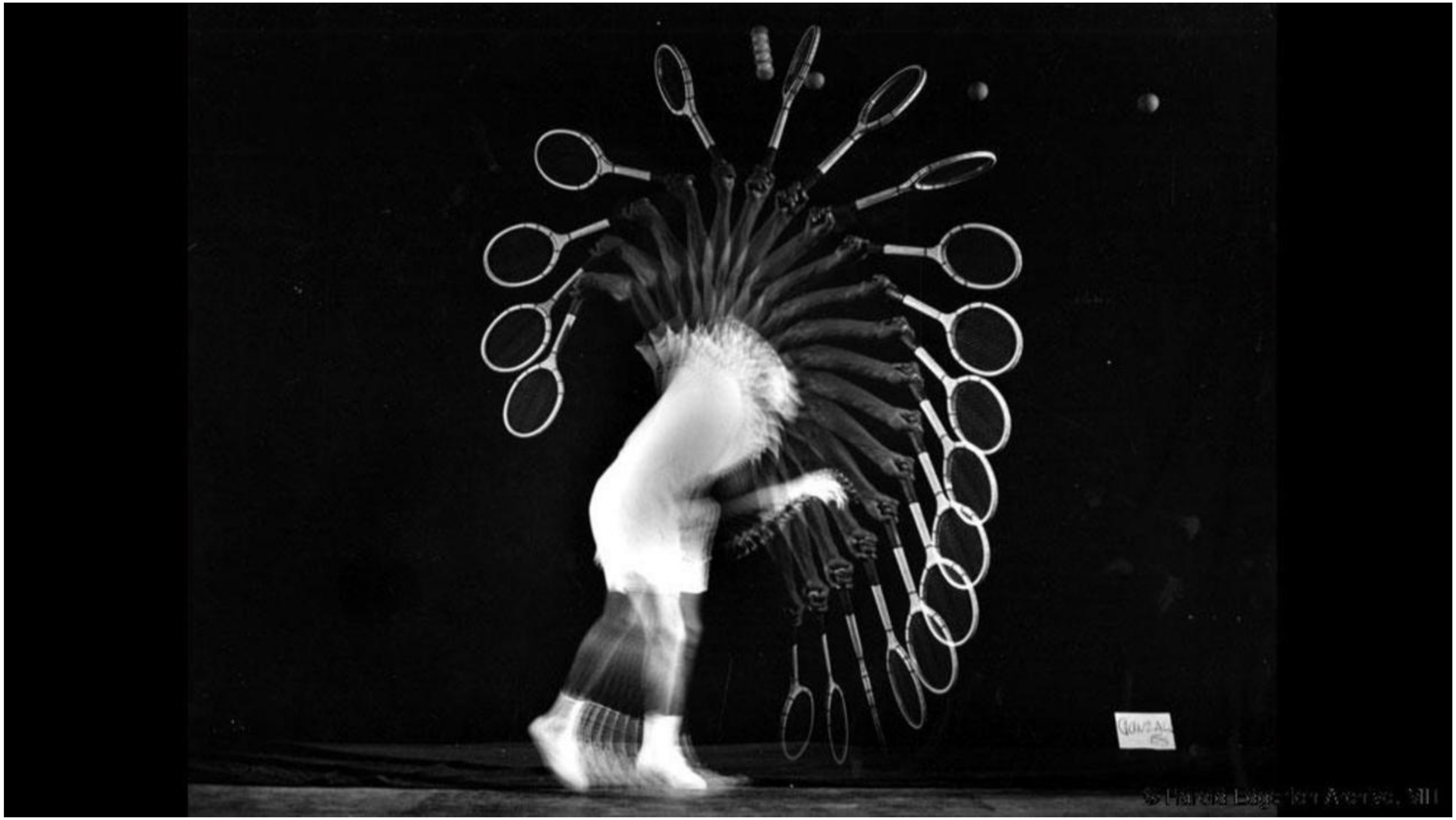
音频：存储 1D 声音信号



- 大多数面向消费者的音频每秒采样 44100 次，即 44.1KHz

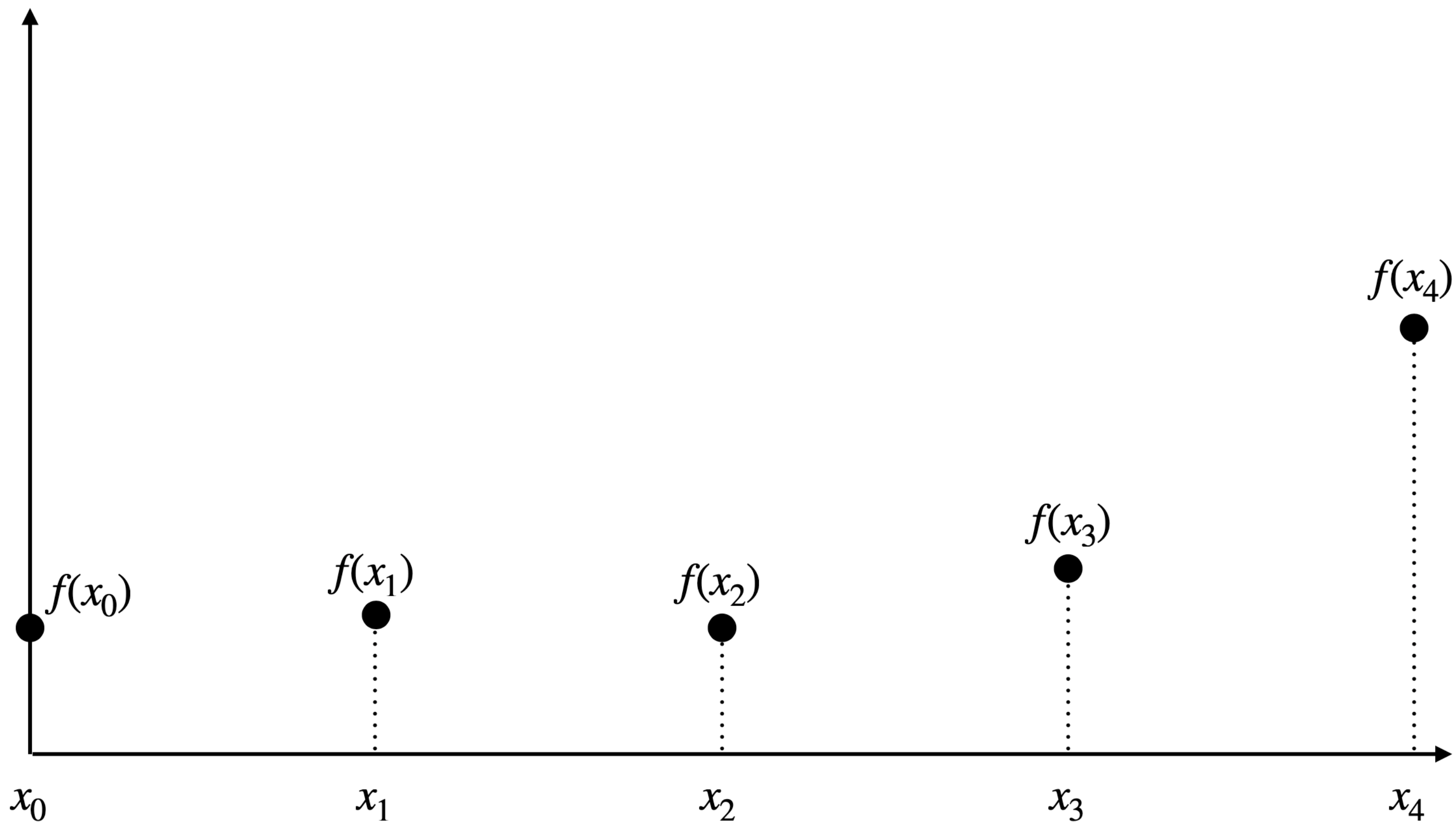
图像：存储 2D 彩色信号

视频：存储 3D 图像信号



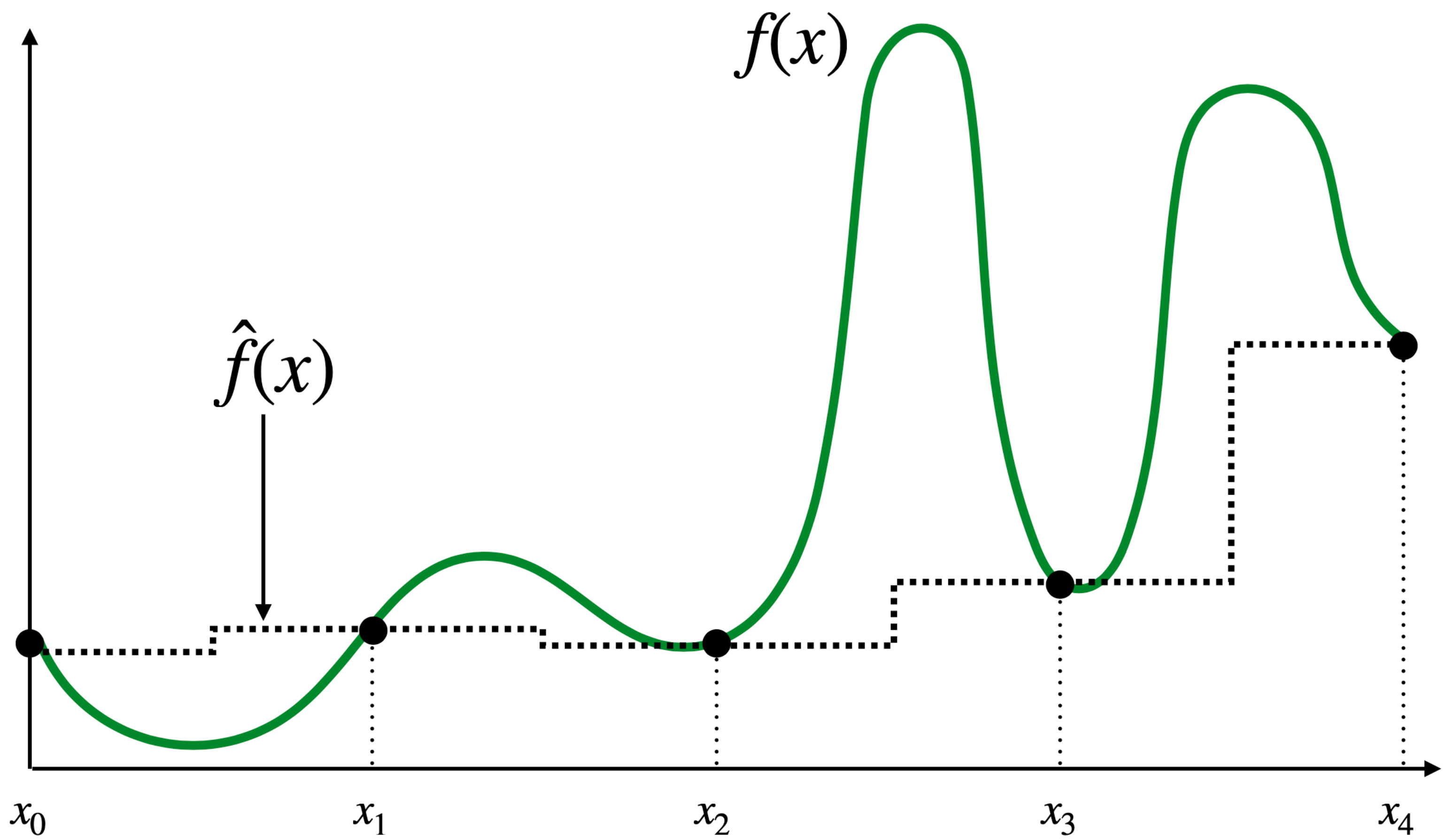
重建 Reconstruction

- 给定一组样本，如何重建原始信号?



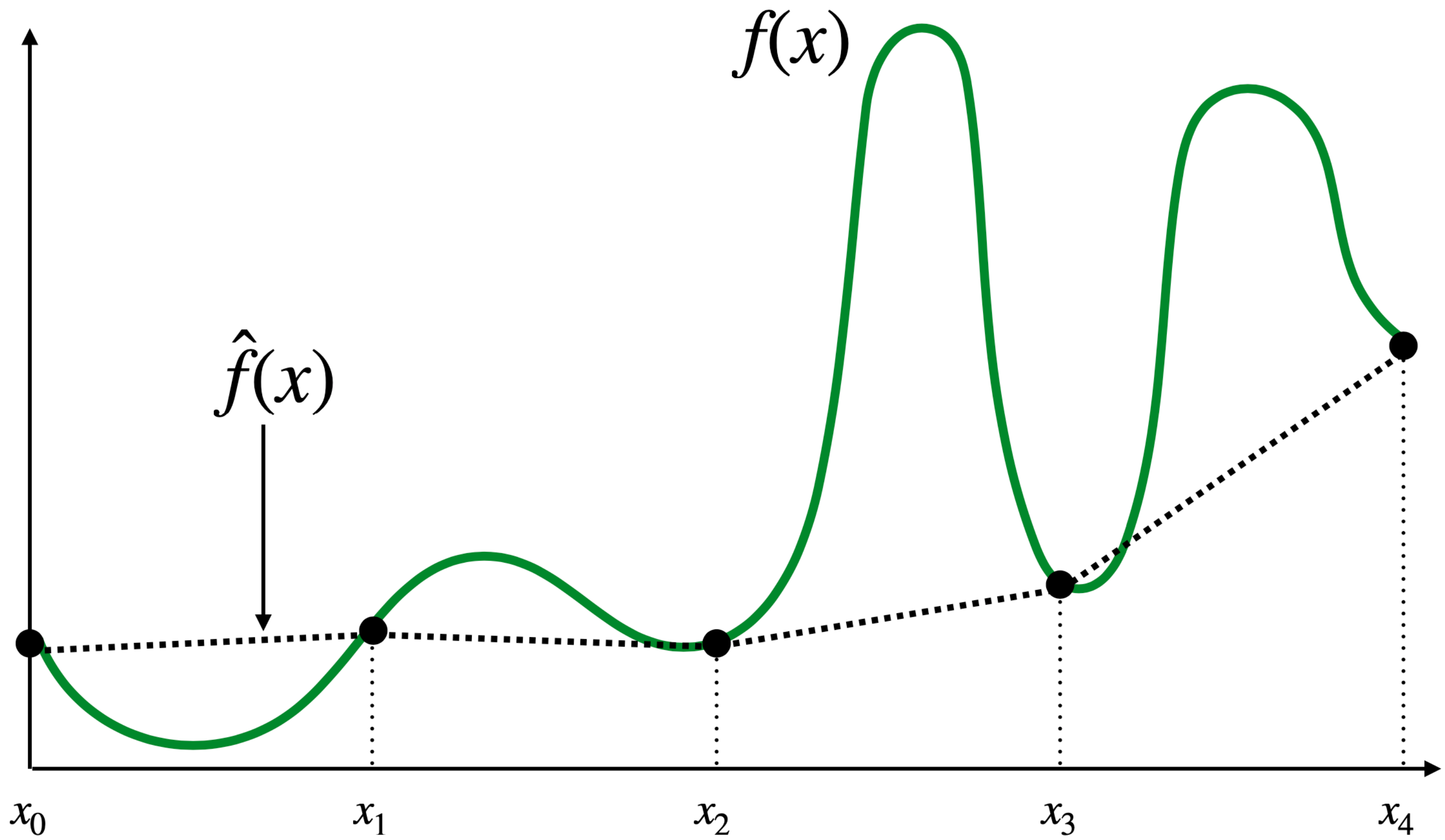
分段常数近似 Piecewise constant approximation

- $\hat{f}(x)$ = 最接近 x 的样本值



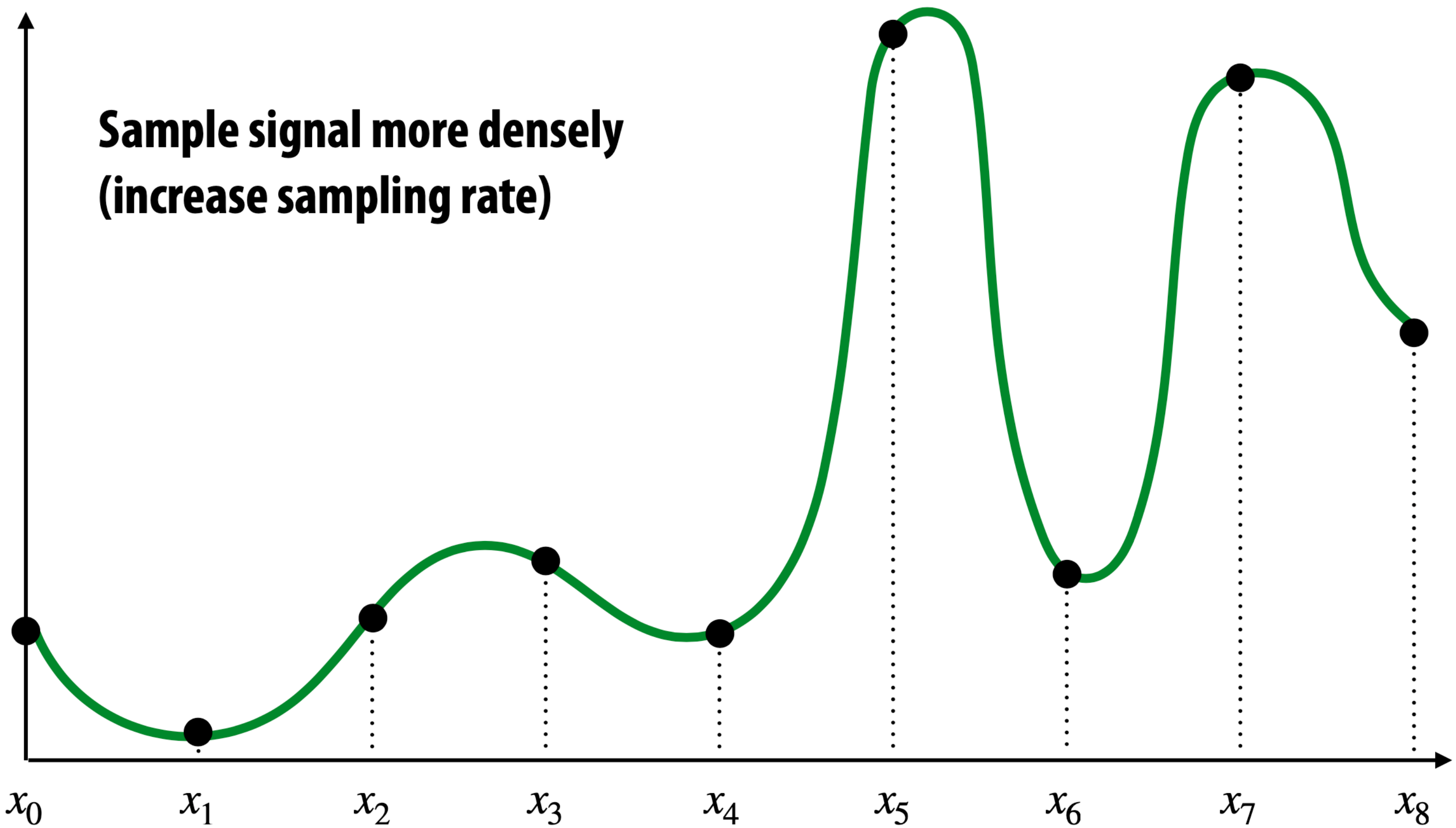
分段线性近似 Piecewise linear approximation

- $\hat{f}(x)$ = 两个连续样本值之间的线性插值



怎样更准确地表示信号?

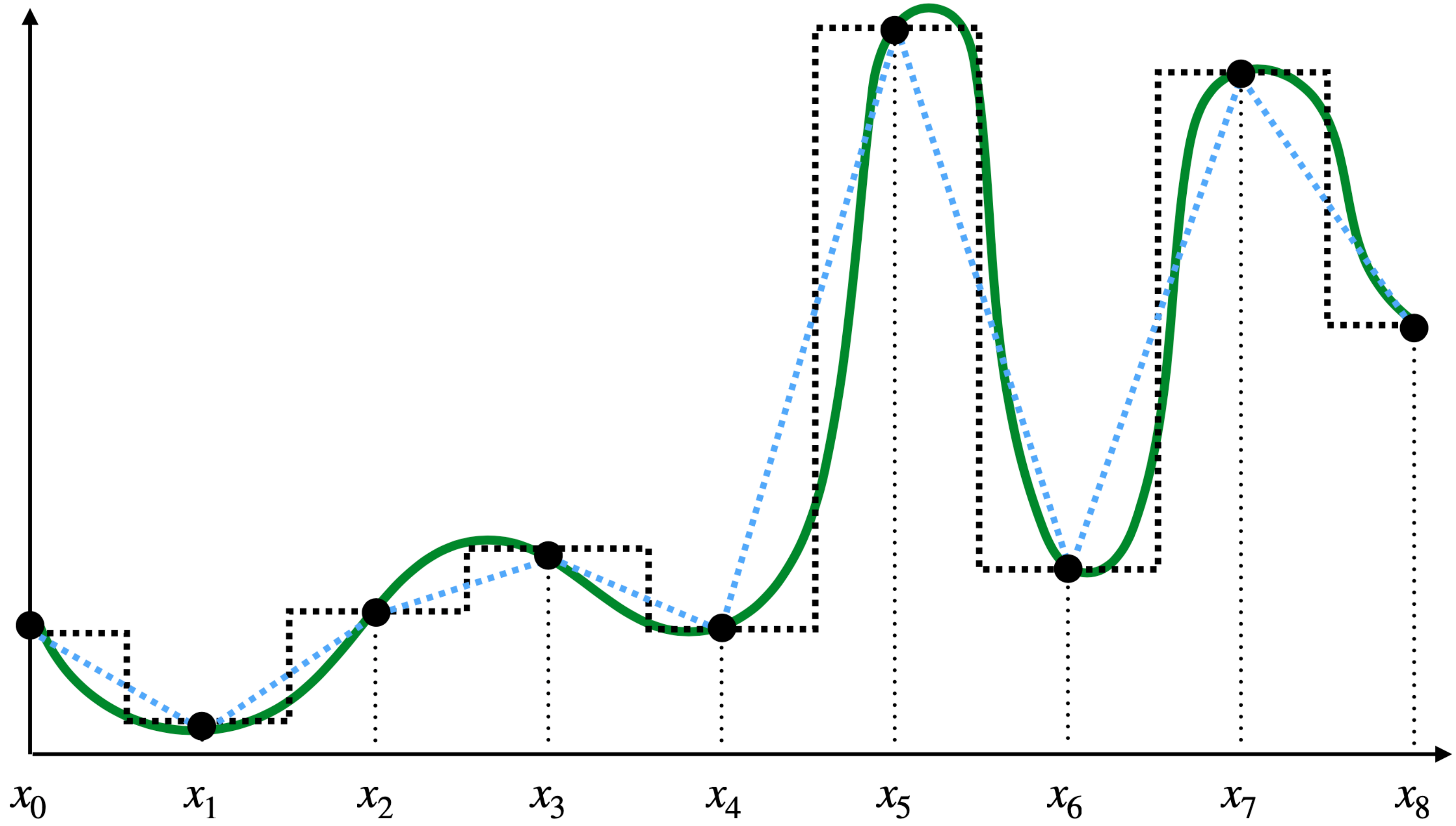
- 采样更多的点



基于更密集采样点的重建

= reconstruction via nearest

= reconstruction via linear interpolation



2D 采样和重建

- 2D 图片的情况类似
  - 采样值测量采样点处的图像（即信号）
  - 应用插值重构原来的图像

original

piecewise constant (“nearest neighbor”)

piecewise *bi*-linear

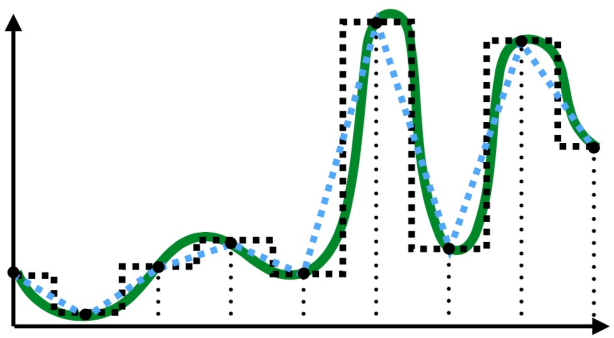
采样 101：总结

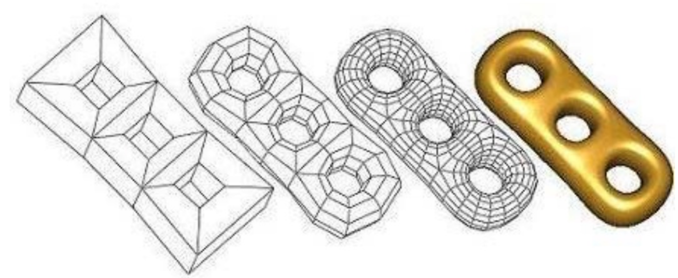
□采样 (sampling)=信号测量 (measurement of a signal)

- 将信号编码为离散样本集
- 每个样本表示特定点的值（尽管在现实中很难测量）

□重构 (reconstruction)=从离散样本集生成信号

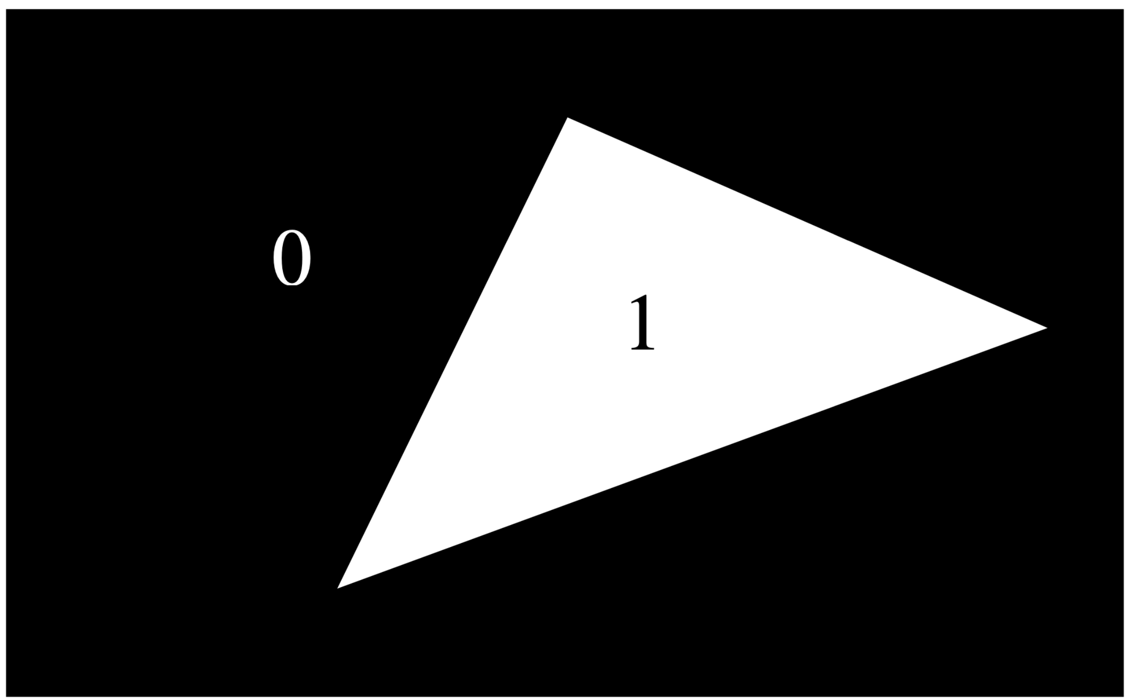
- 构造一个函数以插值或近似原来的函数值
- 例如，分段常数（即最近邻）或分段线性
- 适用于各种信号（音频、图像、几何图形...）



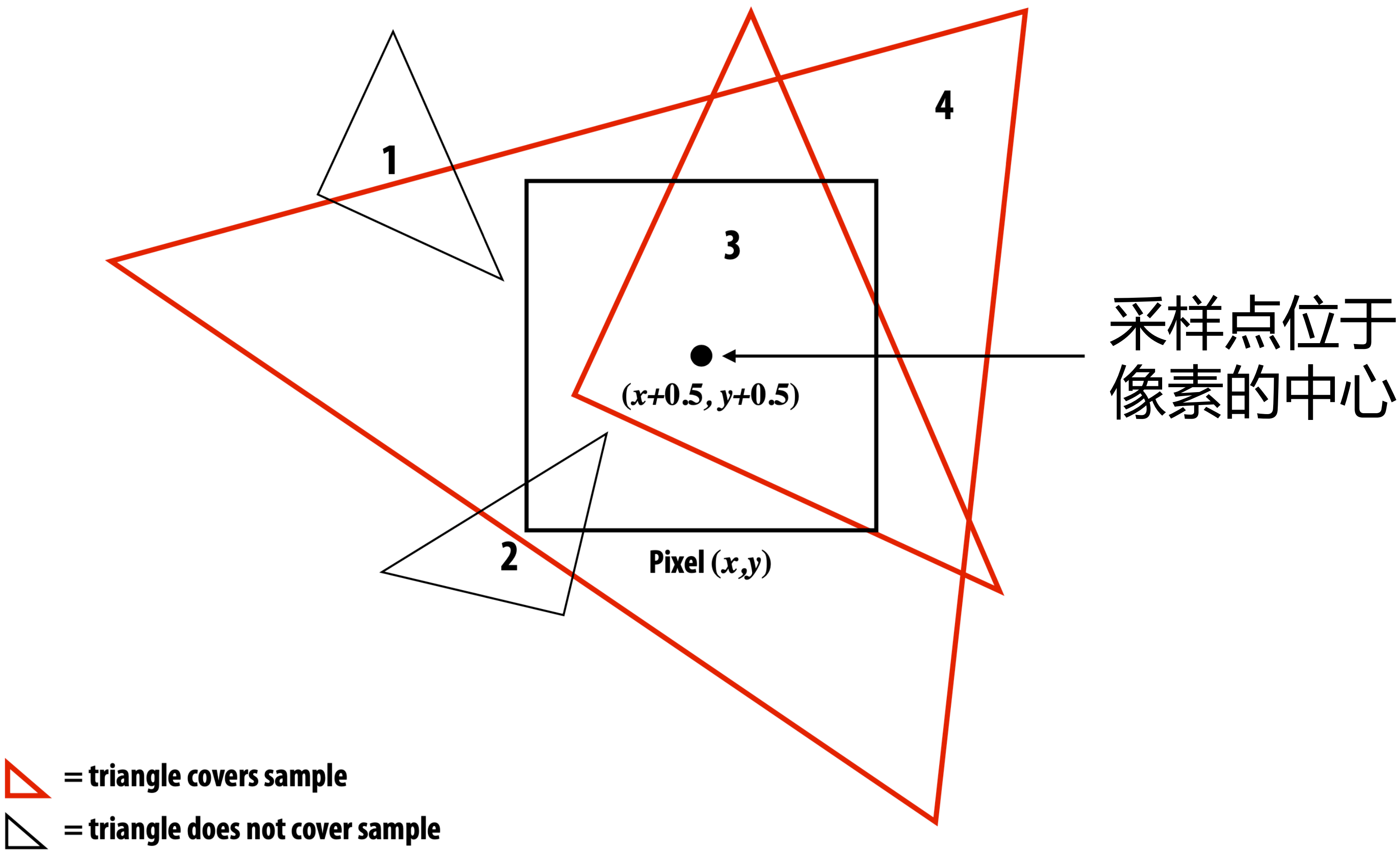


对于光栅化，我们是在采样哪个函数？

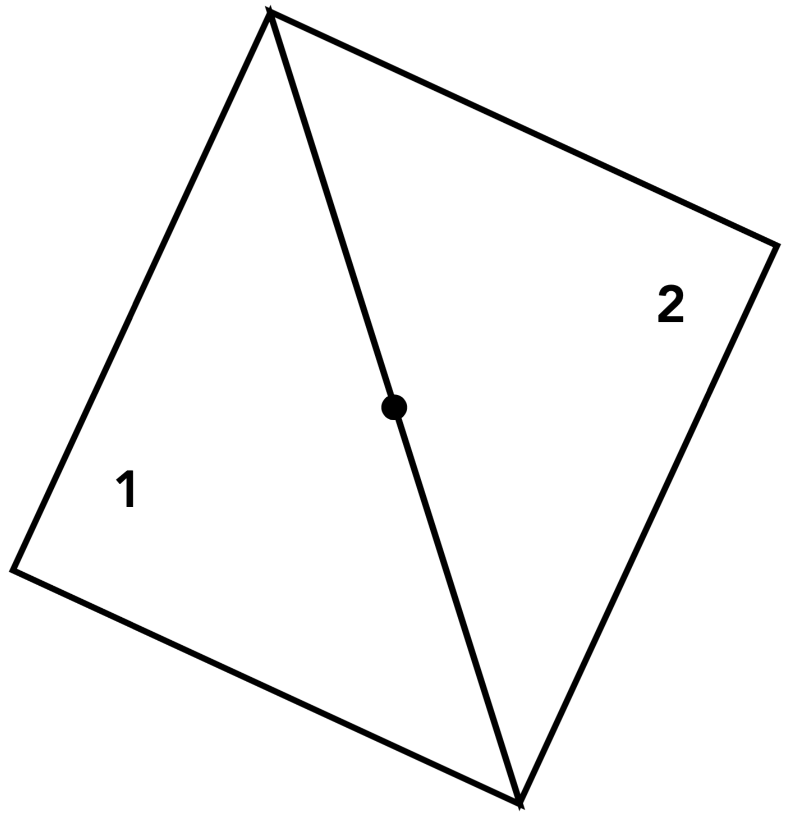
$$\text{coverage}(x, y) := \begin{cases} 1, & \text{triangle contains point } (x, y) \\ 0, & \text{otherwise} \end{cases}$$



对多个覆盖函数进行采样

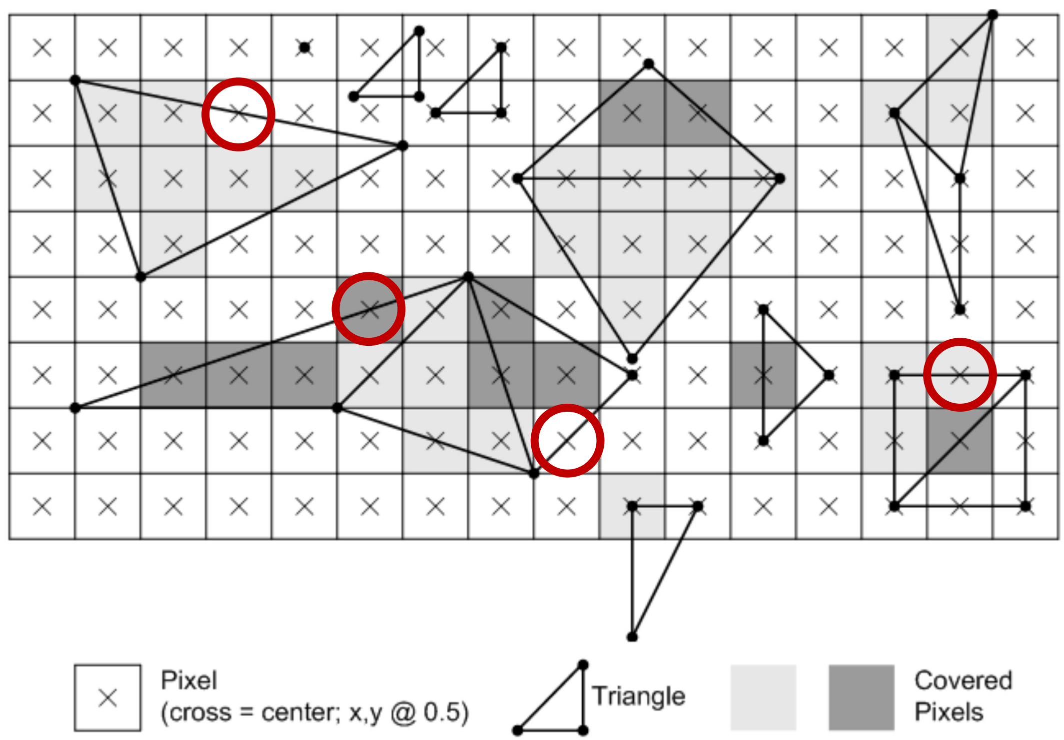


边上的采样点

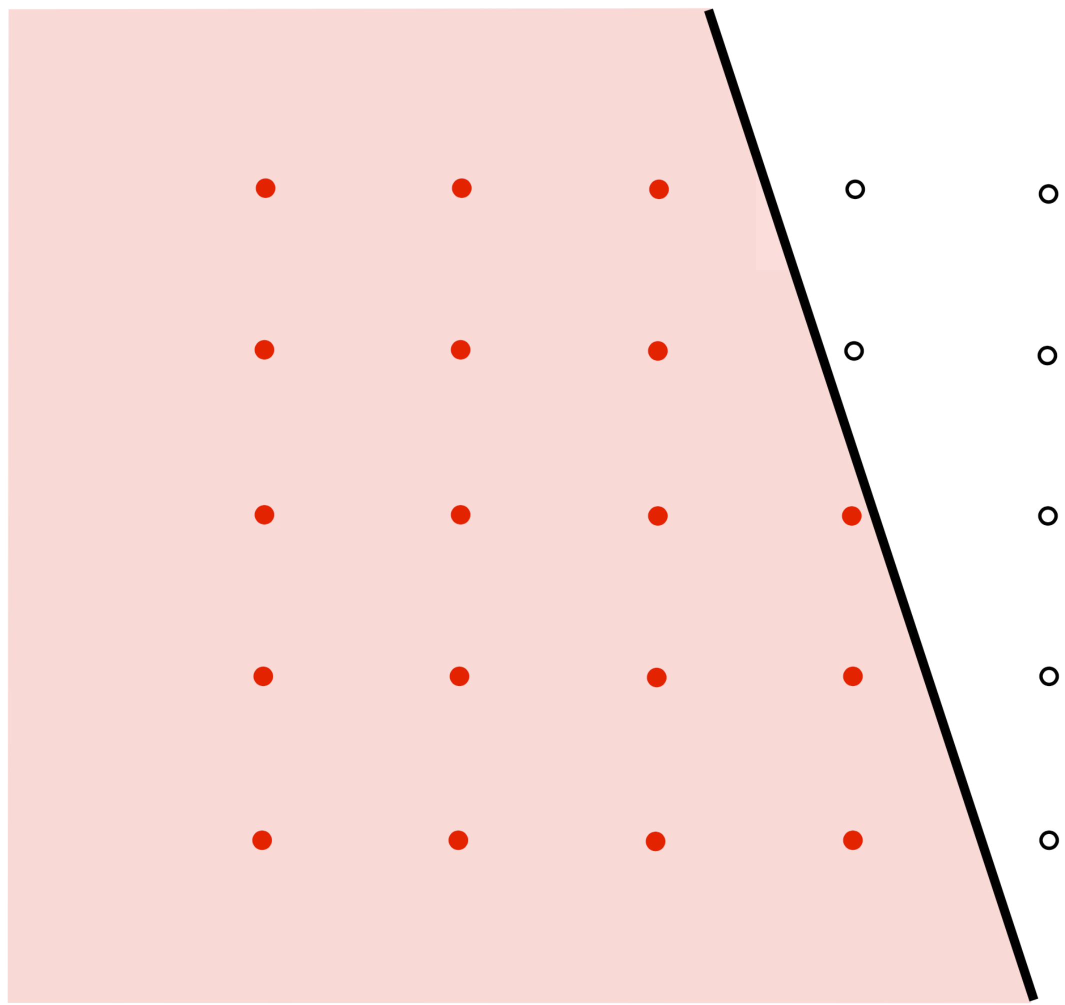


OpenGL/Direct3D 边缘规则

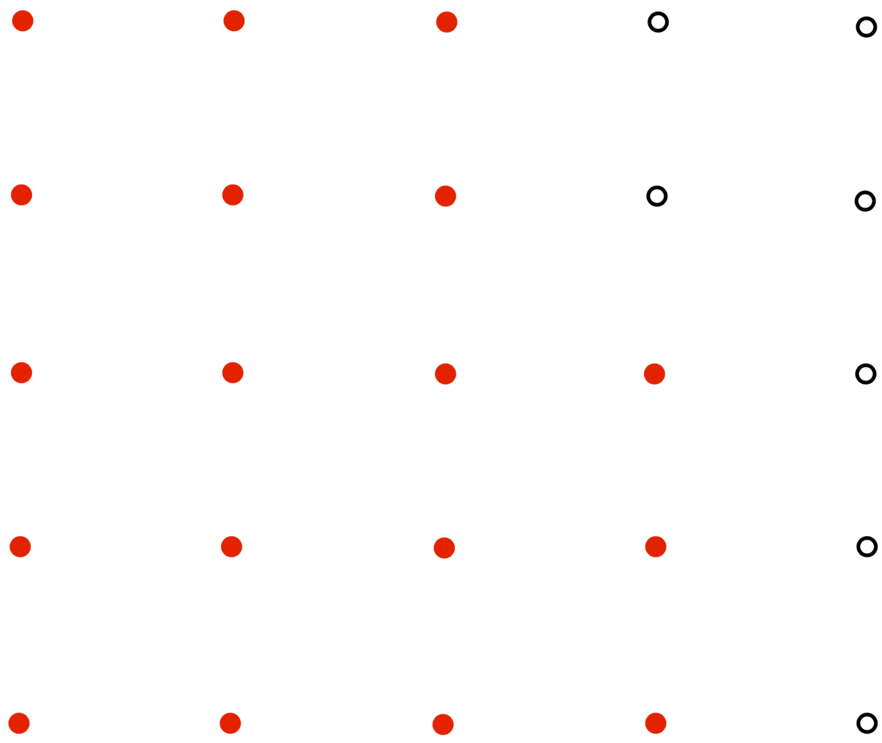
- 当采样点落在边上时，如果边是“顶边”或“左边”，则该采样被分类为三角形内
  - **顶边 (top edge)**: 高于所有其他边的水平边
  - **左边 (left edge)**：非水平且位于三角形左侧的边（三角形可以有一个或两个左边)



三角形覆盖函数的采样结果



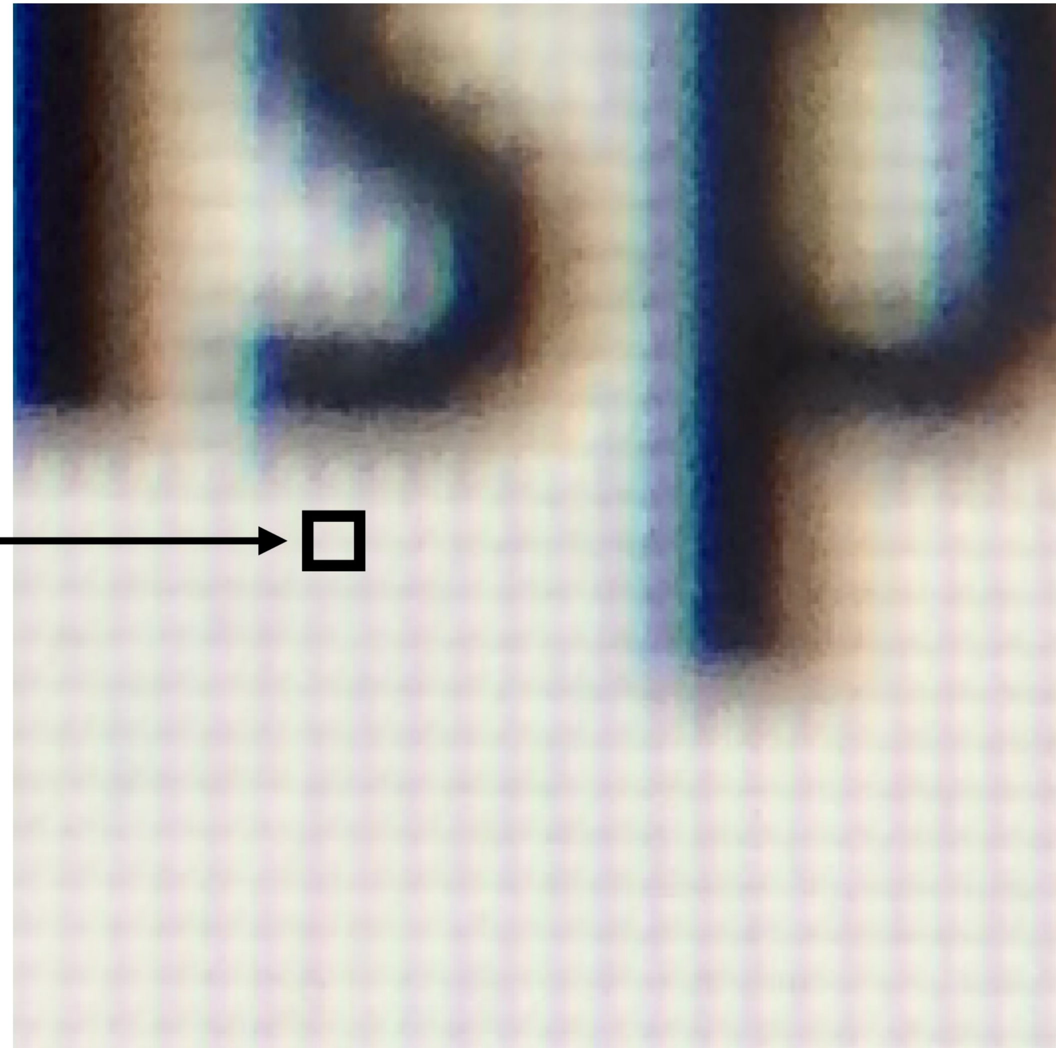
采样得到的数据



假设显示像素发射正方形光

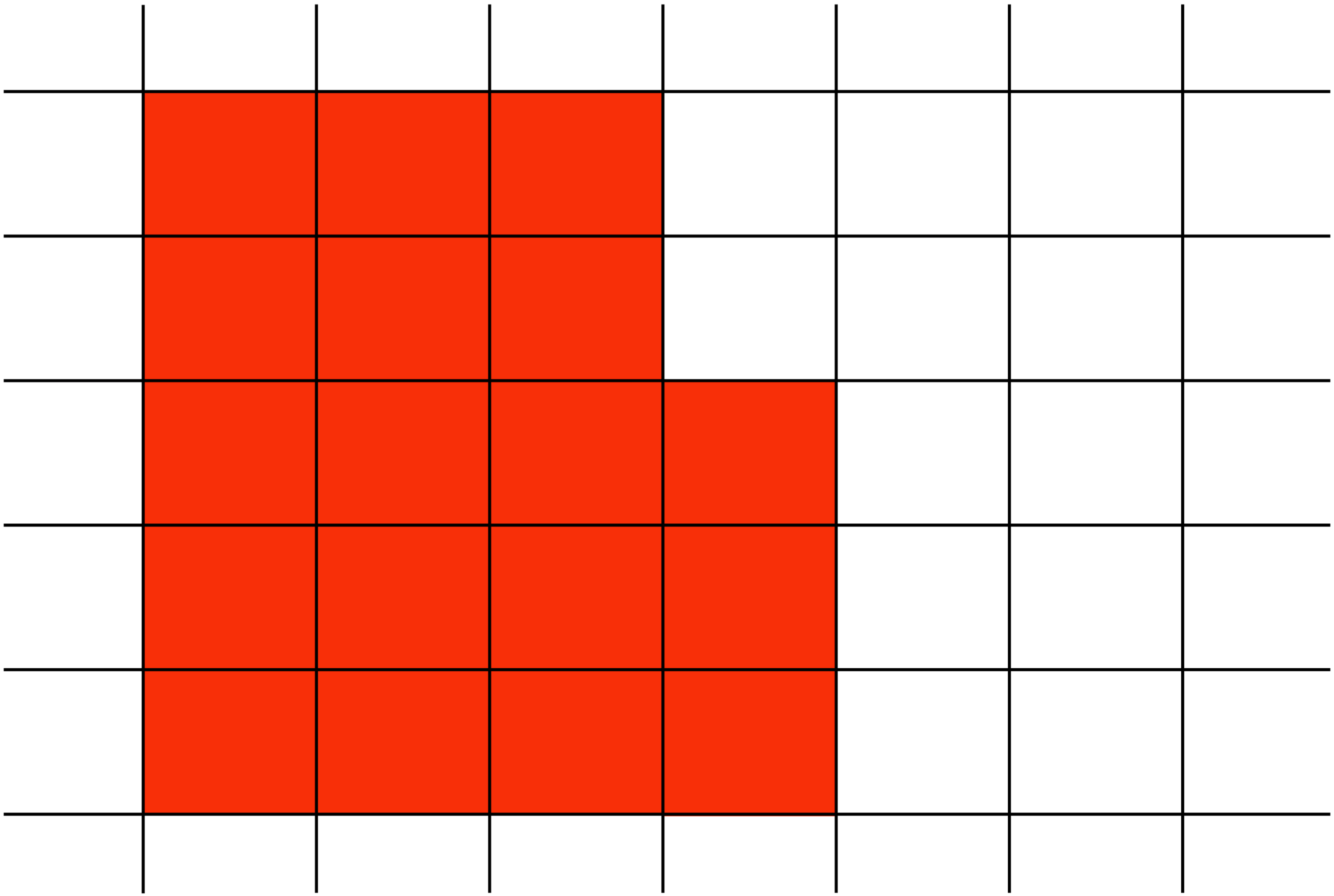
每个发送到显示器的图像样本都会被转换成相应颜色的小光方块

LCD pixel on laptop



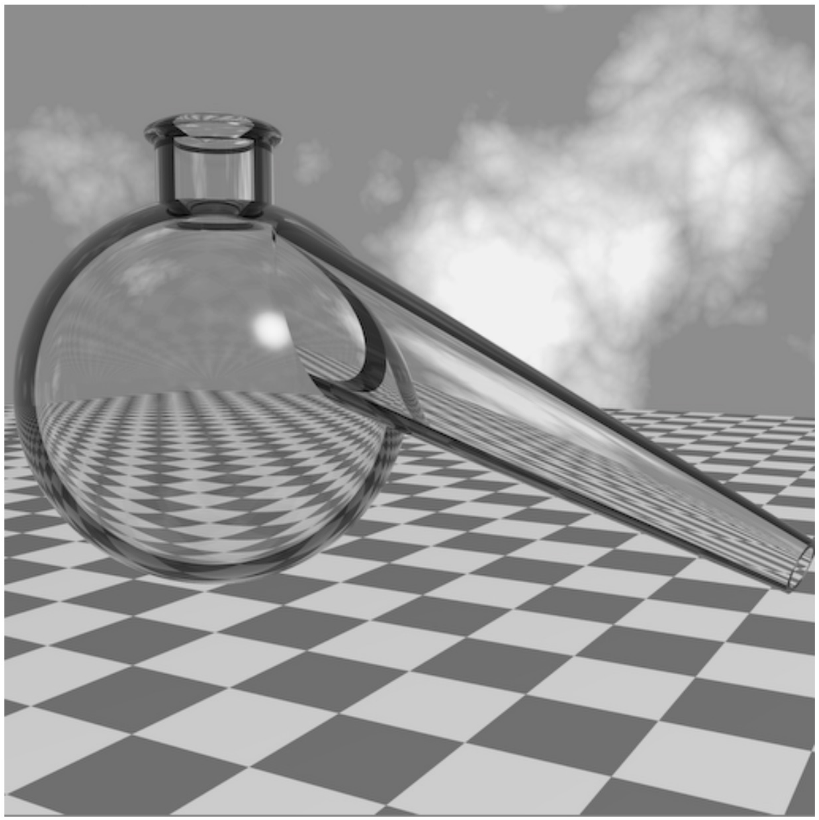
屏幕显示的结果

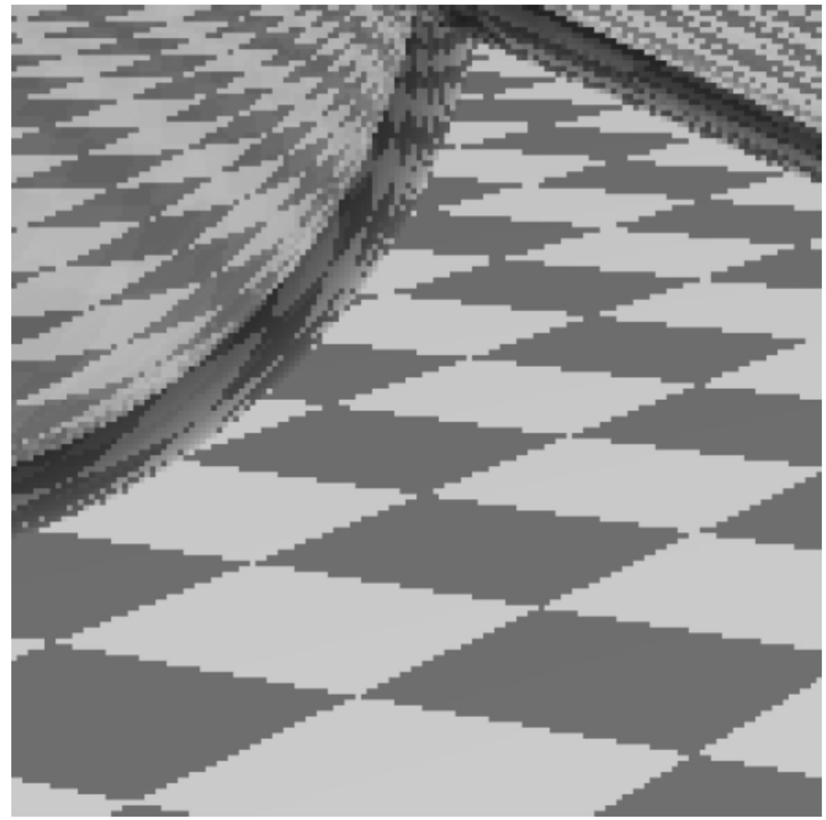
- 每个方格表示一个像素



但真实的覆盖信号是这样的

Jaggies（楼梯样式）

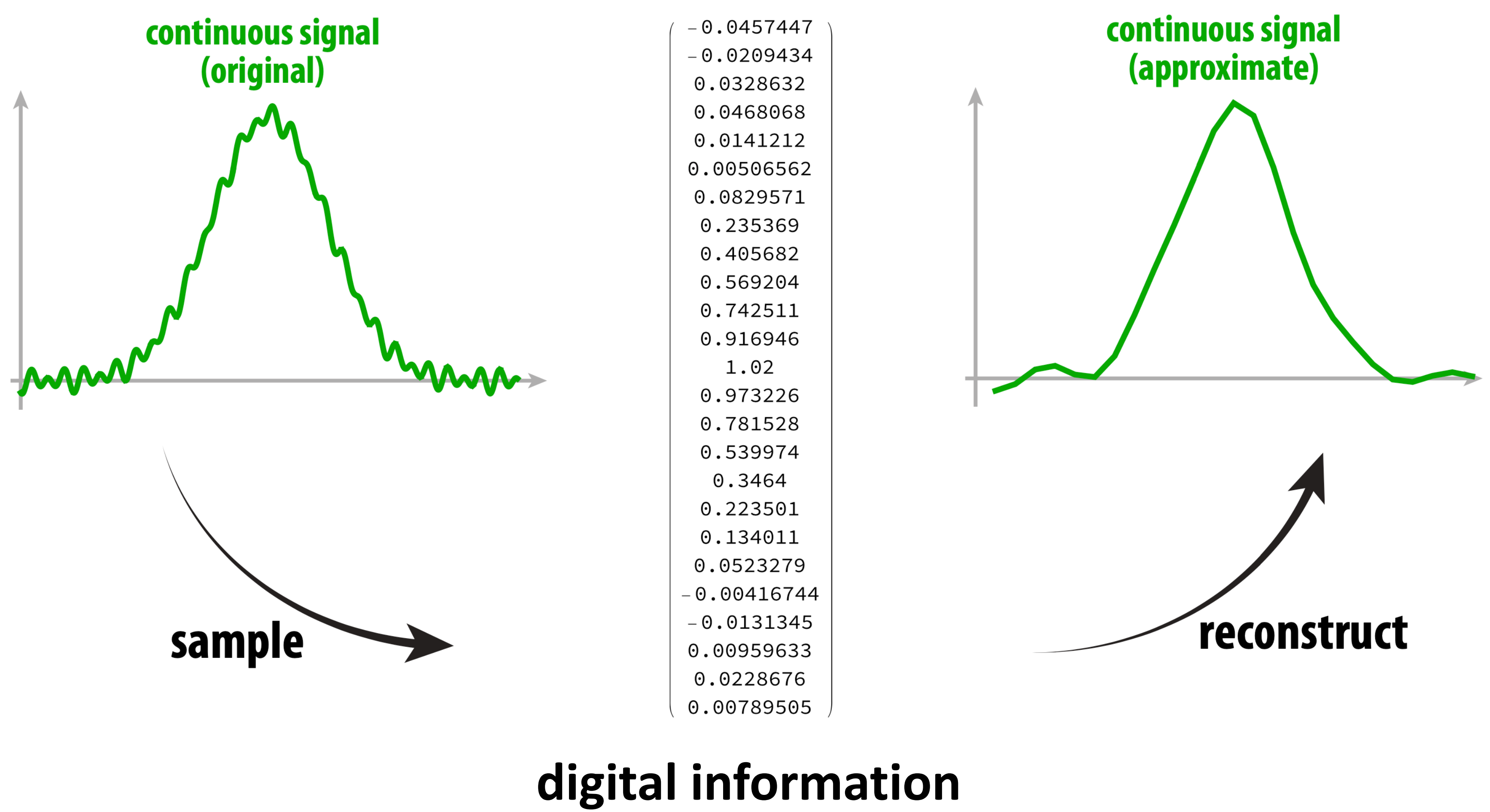




摩尔条纹 Moiré Patterns

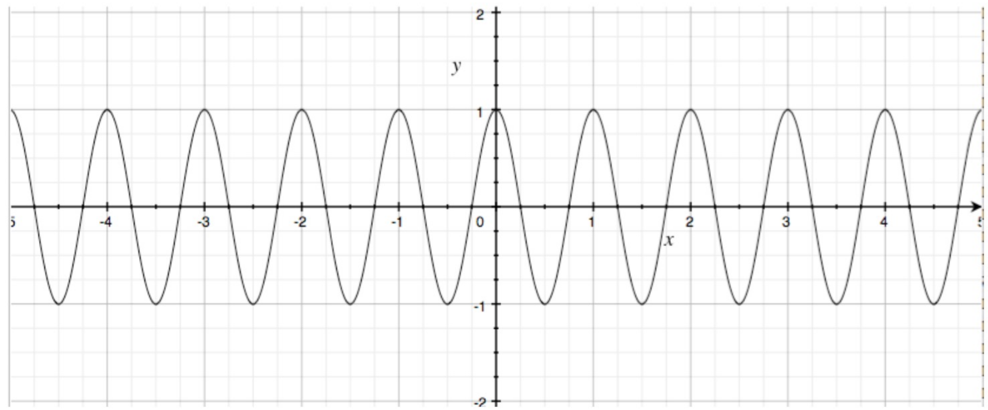
走样
Aliasing

采样和重构

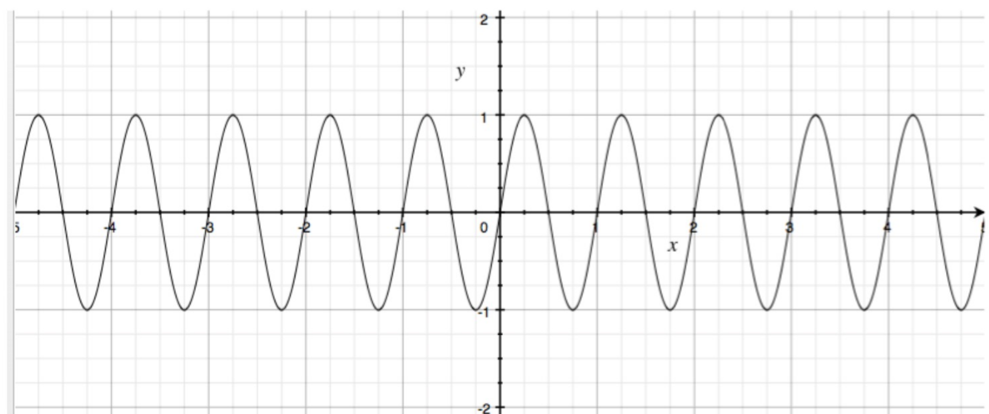


目标：尽可能地准确还原原来的信号

Sines and Cosines



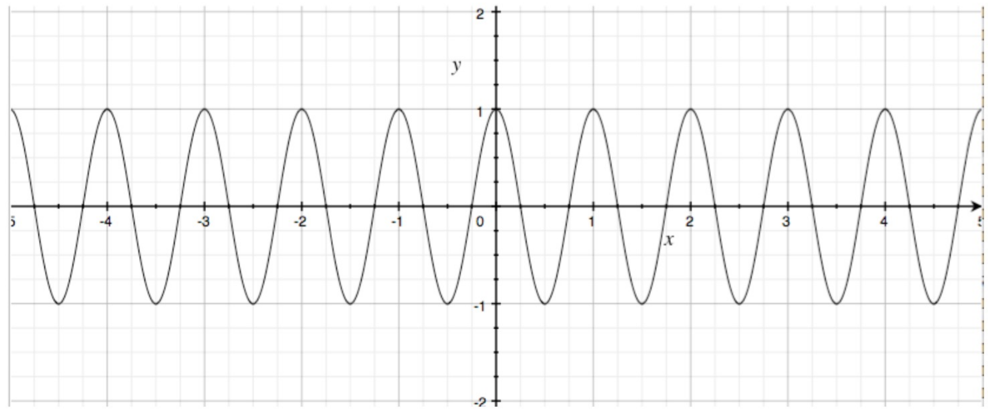
$\cos 2\pi x$



$\sin 2\pi x$

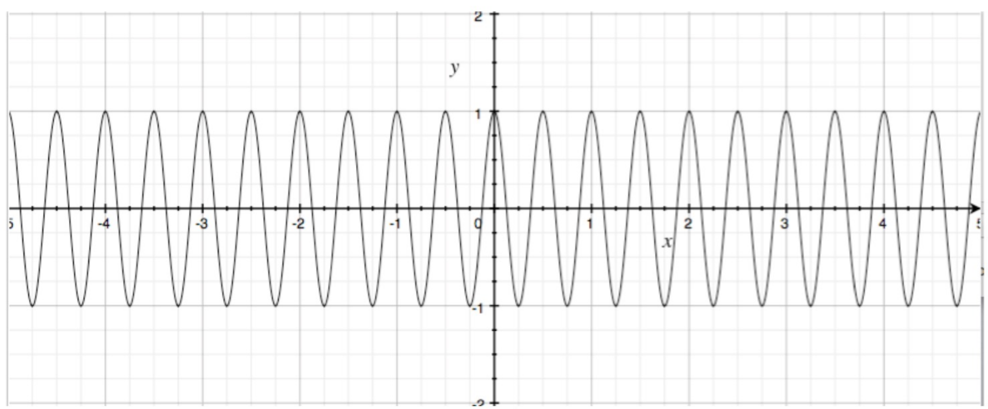
频率 cos2πfx

$$f = \frac{1}{T}$$



$$f = 1$$

$$\cos 2\pi x$$

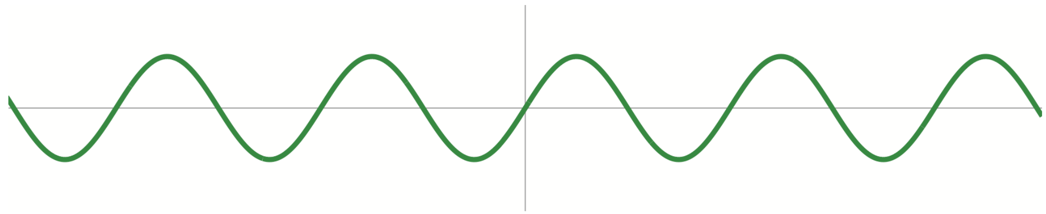


$$f = 2$$

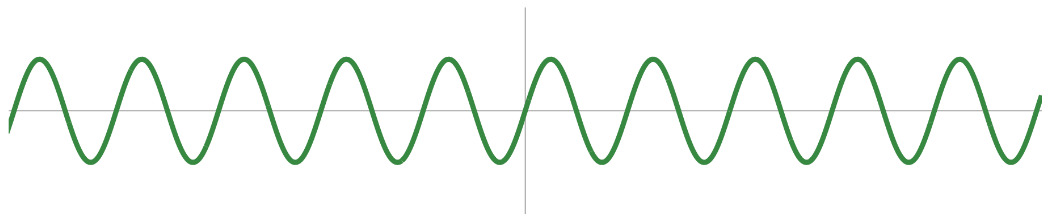
$$\cos 4\pi x$$

傅立叶变换 – 例子

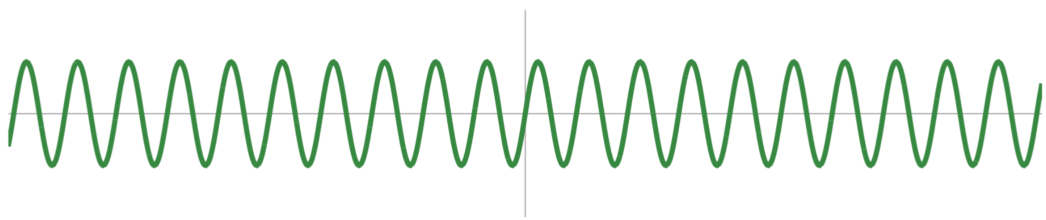
$f_1(x) = sin(\pi x)$



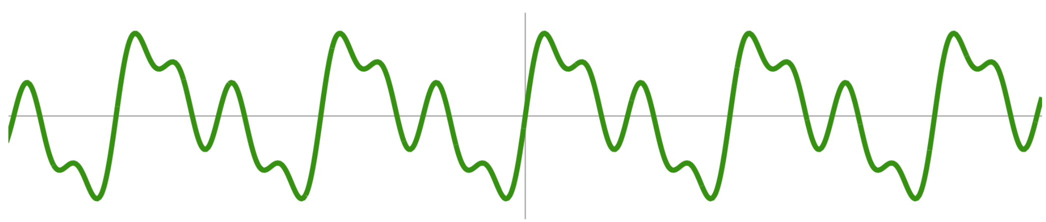
$f_2(x) = sin(2\pi x)$



$f_4(x) = sin(4\pi x)$



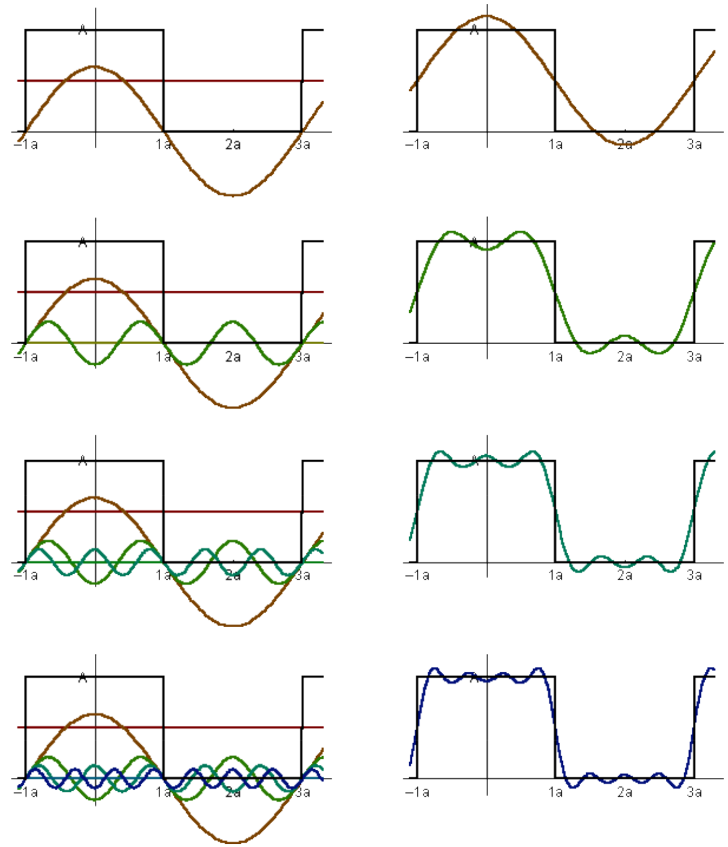
$f(x) = f_1(x) + 0.75\, f_2(x) + 0.5\, f_4(x)$



傅立叶变换

Represent a function as a weighted sum of sines and cosines

Joseph Fourier 1768 - 1830



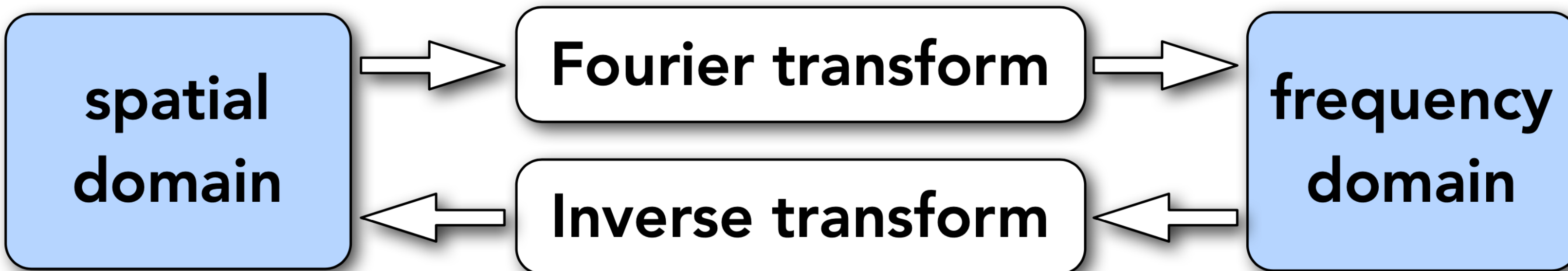
$$f(x) = \frac{A}{2} + \frac{2A\cos(t\omega)}{\pi} - \frac{2A\cos(3t\omega)}{3\pi} + \frac{2A\cos(5t\omega)}{5\pi} - \frac{2A\cos(7t\omega)}{7\pi} + \cdots$$

傅立叶变换将信号分解为频率

$f(x)$

$$F(\omega) = \int_{-\infty}^{\infty} f(x) e^{-2\pi i \omega x} dx$$

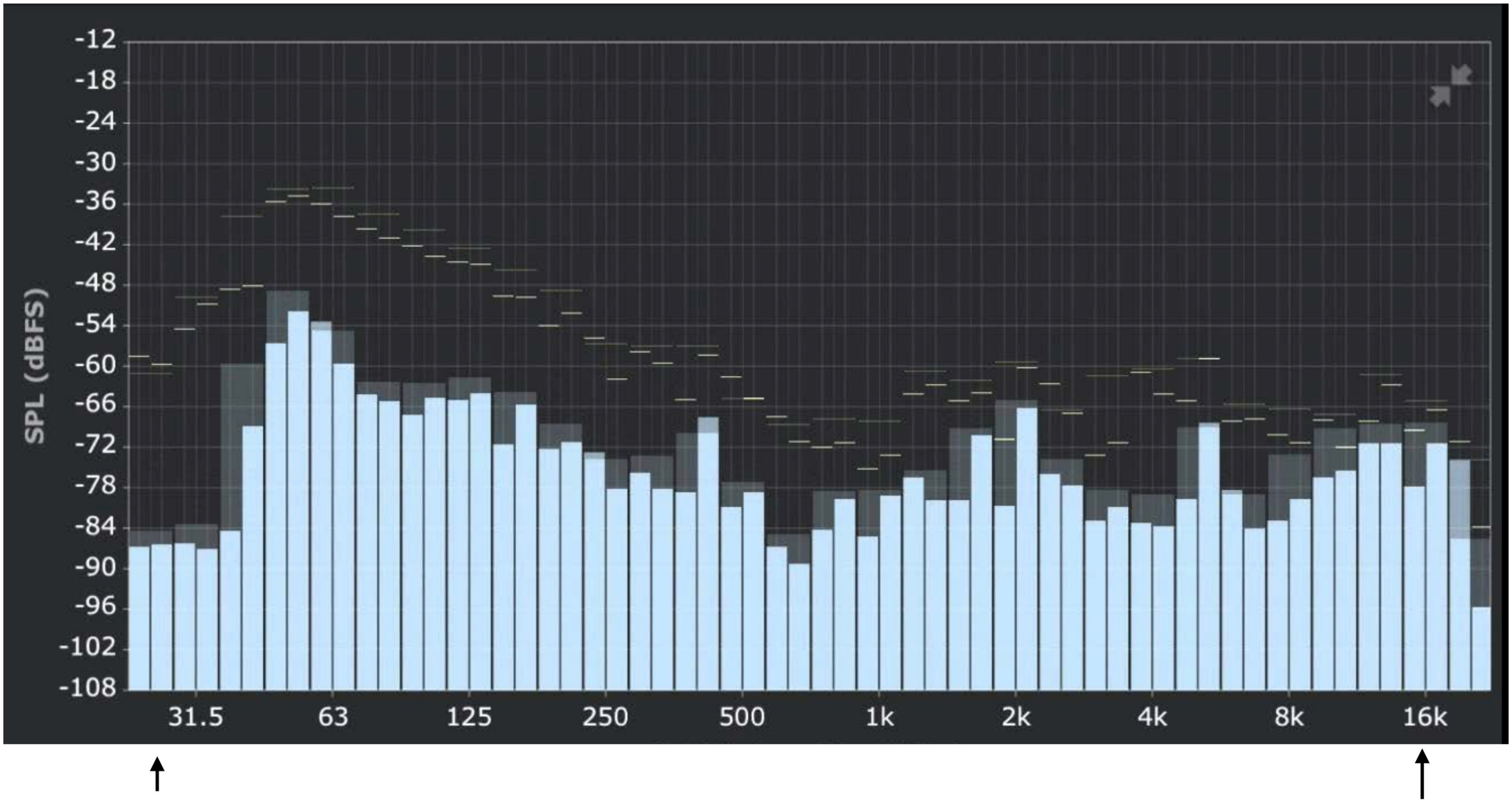
$F(\omega)$



$$f(x) = \int_{-\infty}^{\infty} F(\omega) e^{2\pi i \omega x} d\omega$$

Recall $e^{ix} = \cos x + i \sin x$

E.g., 音频频谱分析仪显示每个频率的幅度



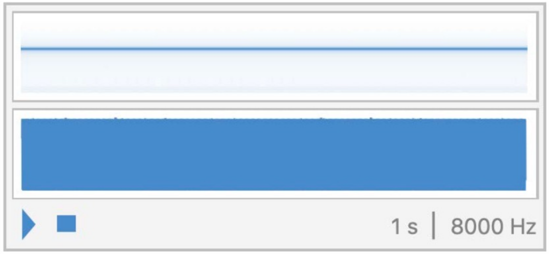
Intensity of low-frequencies (bass)

Intensity of high frequencies

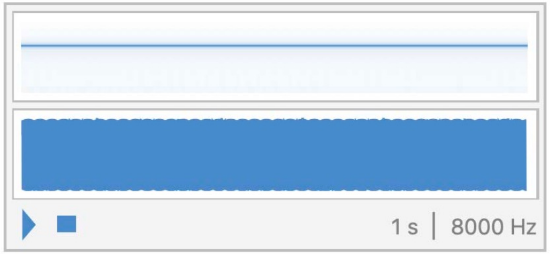
音频中的走样

□通过播放频率为 w 的正弦曲线来获得恒定的音调

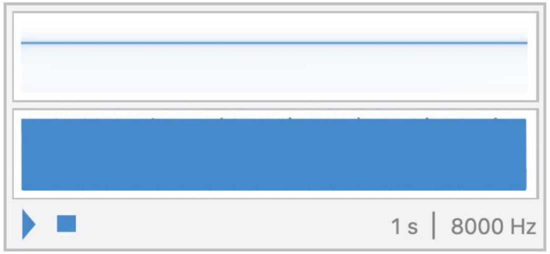
Play[sin[4000 t], {t, 0, 1}]



Play[sin[5000 t], {t, 0, 1}]



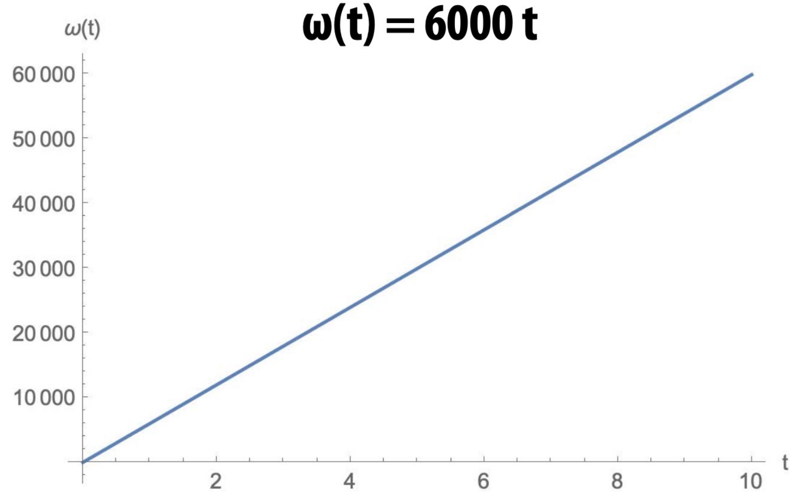
Play[sin[6000 t], {t, 0, 1}]



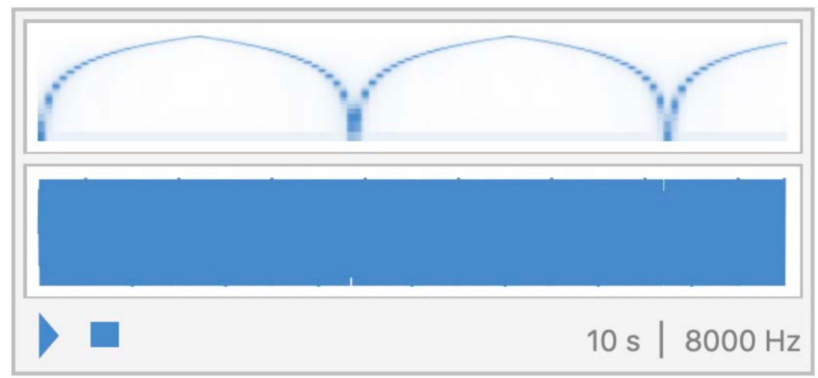
音频中的走样

- 如果我们随时间不停增加 w 会怎么样?

$\omega(t) = 6000\,t$

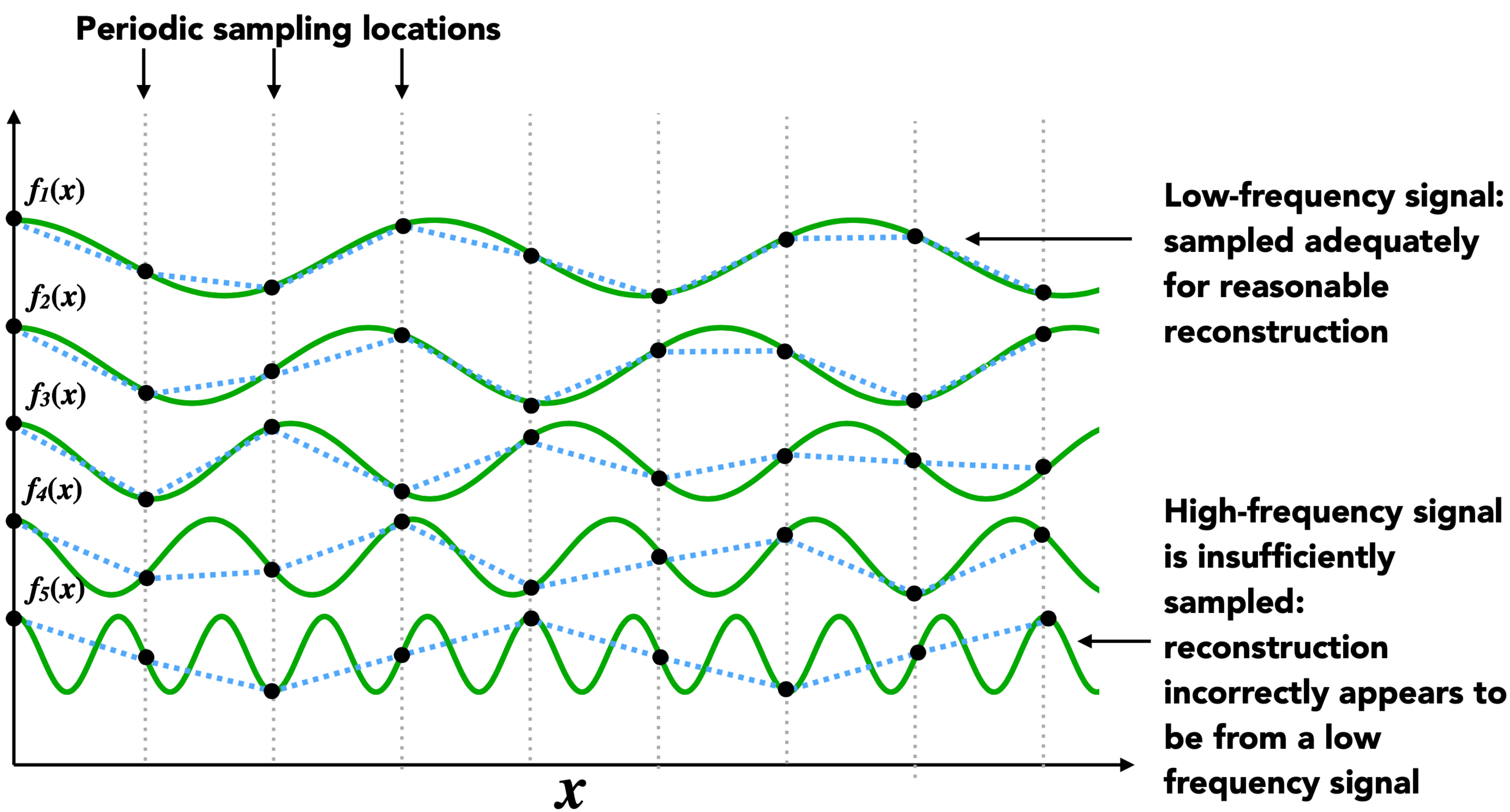



```
Play[Sin[ω t], {t, 0, 10}]
```

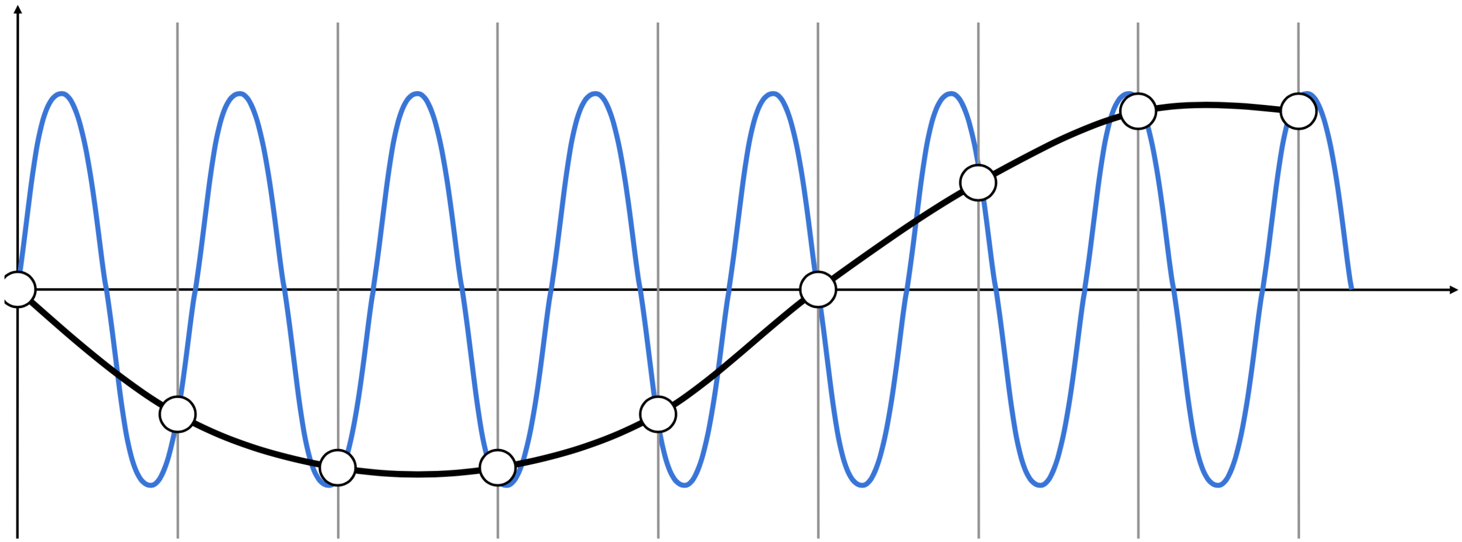


Why did that happen?

高频信号采样不足会导致走样



欠采样导致走样



- 高频信号采样不足：样本错误地看起来来自低频信号
- 在给定的采样率下无法区分的两个频率被称为

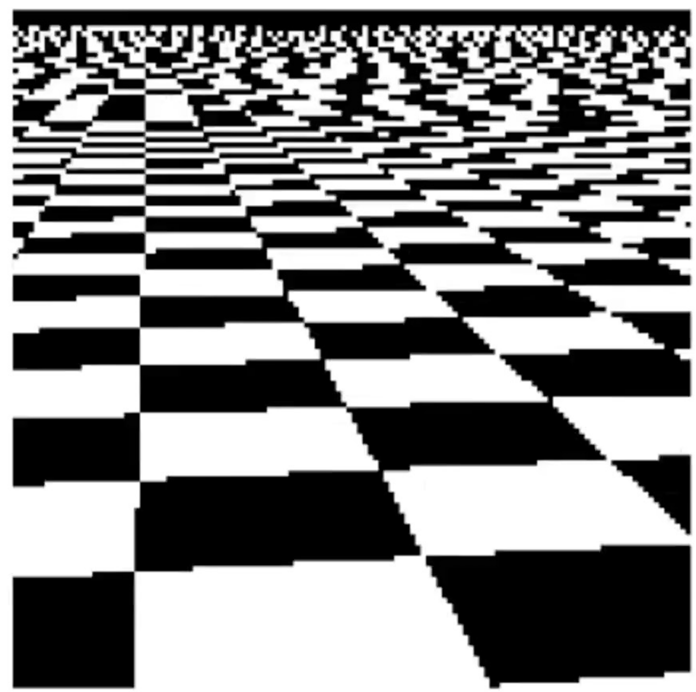
走样 aliasing

Alias = 虚假身份

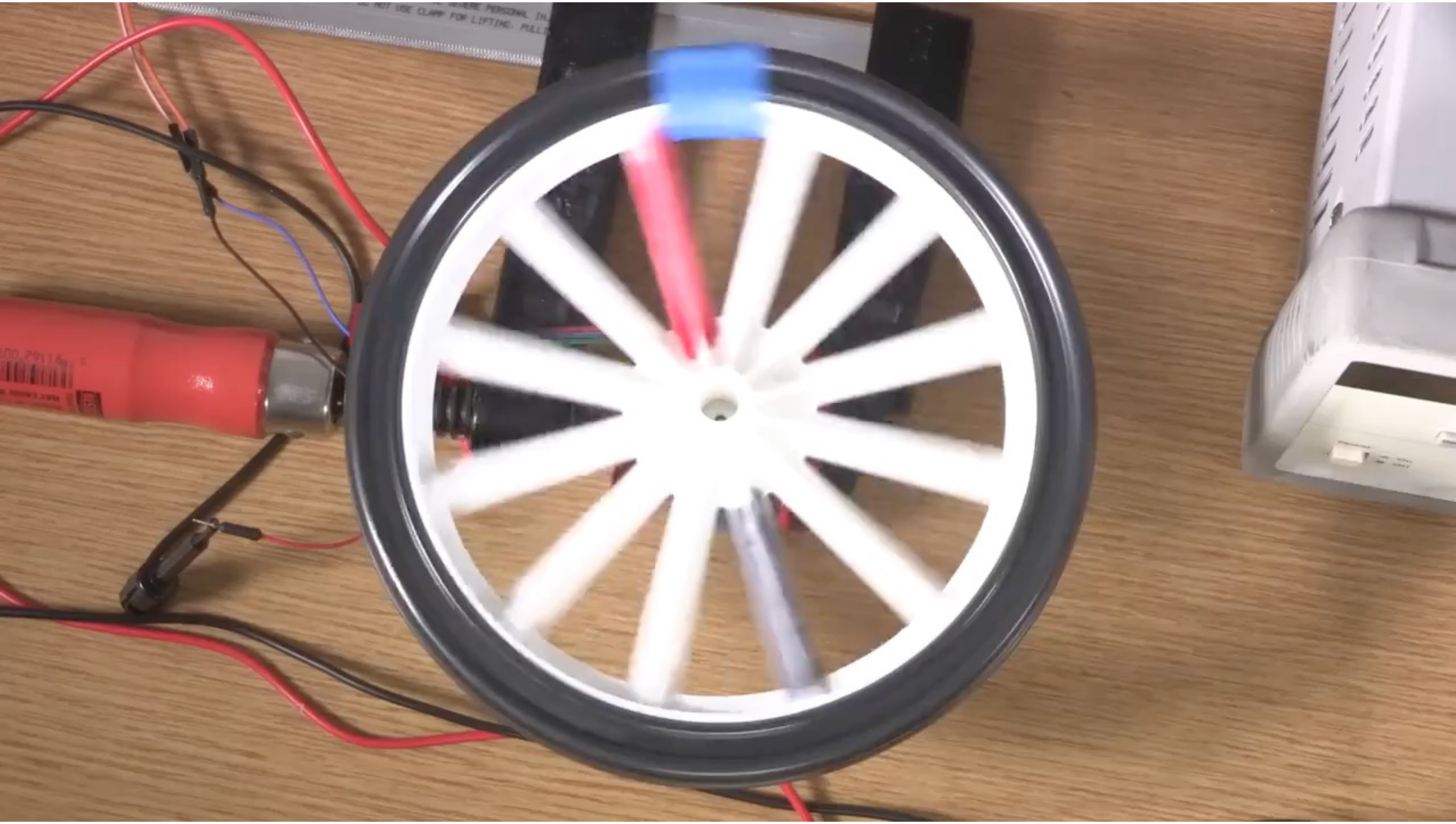
“Batman” = Bruce Wayne’s alias to hide his true identity

图像中的走样伪影

- 采样不完美 + 重建不完美导致图像伪影 (artifacts)
  - 静态图片中的锯齿 (jaggies)
  - 动画中的“缠绕 (roping)”或“闪烁 (shimmering)”
  - 图像高频区域的莫尔条纹 (Moiré patterns)

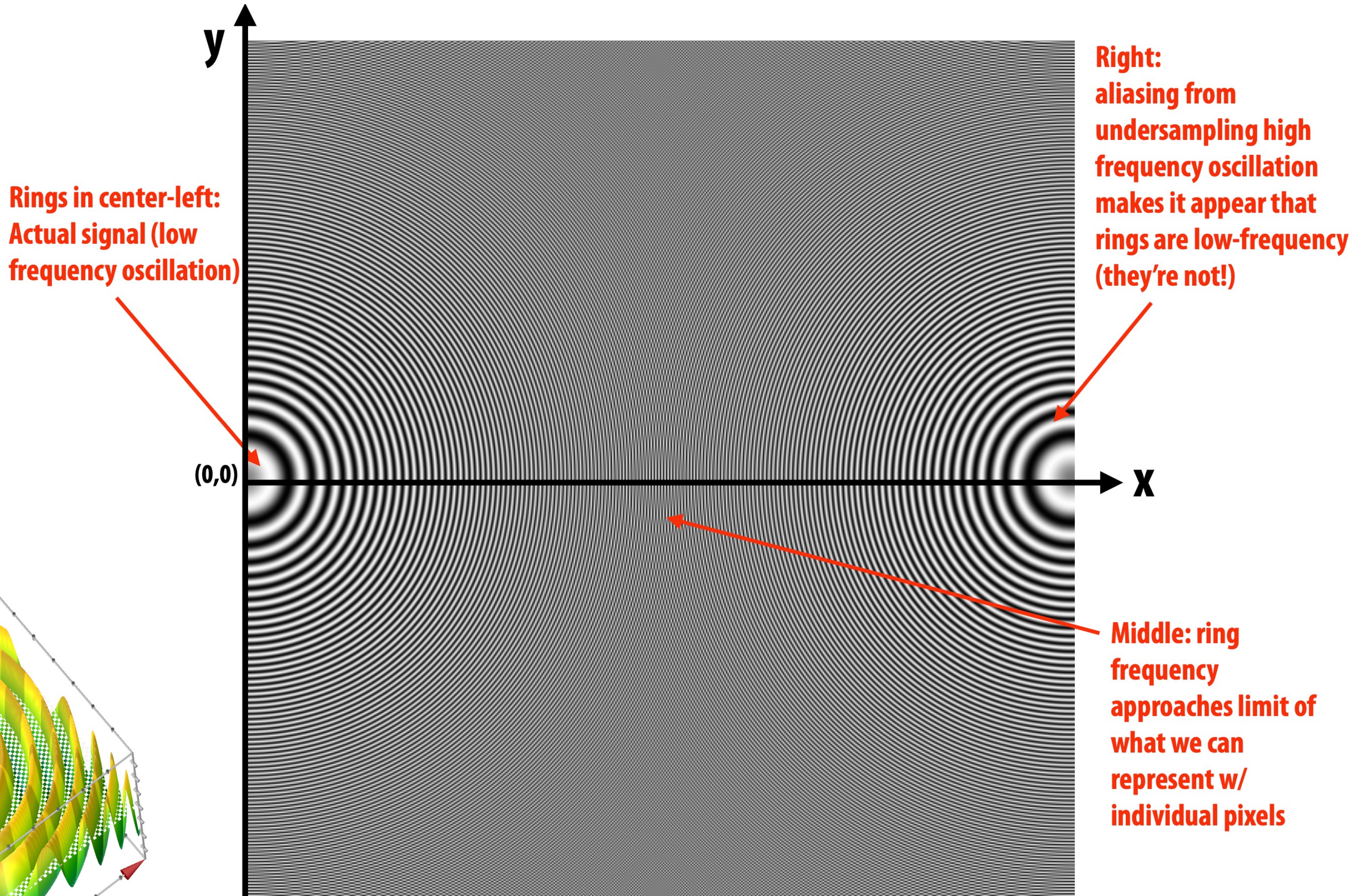


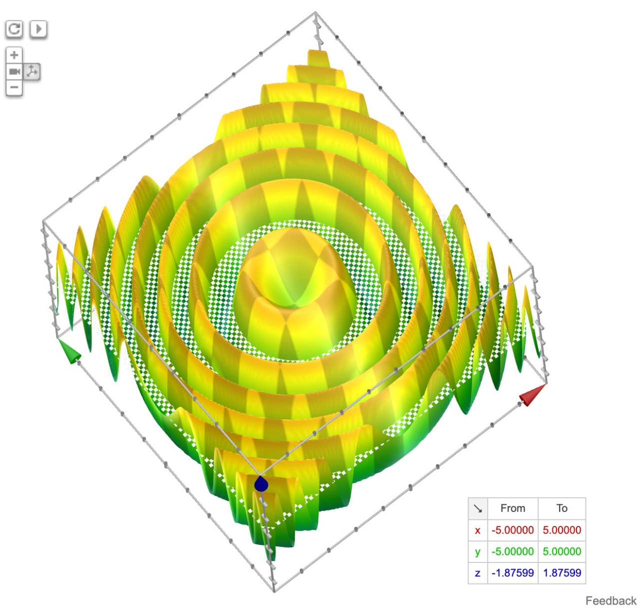
时间走样：车轮效果



对于快速旋转的车轮，相机的帧率（时间采样率）太低

空间走样：函数 $\sin(x^2+y^2)$

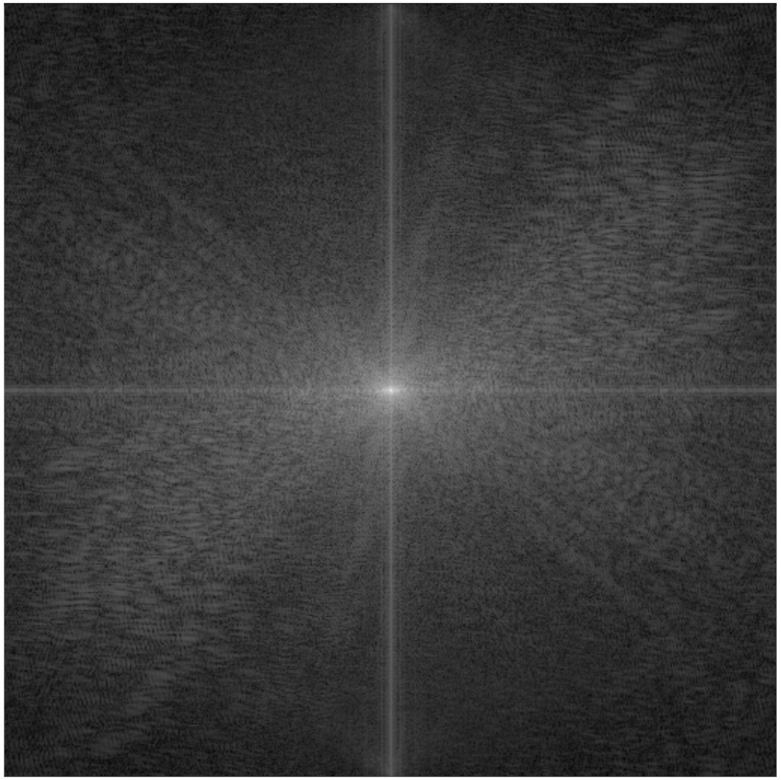




可视化频域空间

2D 频域空间

Spatial Domain

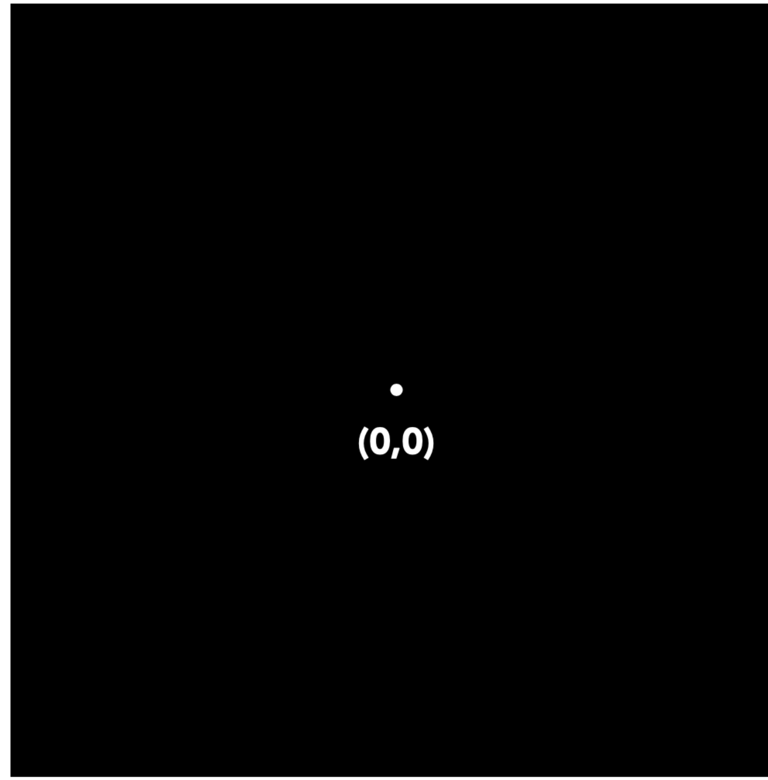


Frequency Domain

频域 (Frequency domain) 也被称为频域空间 (frequency space), 傅立叶域 (Fourier domain), 频谱 (spectrum)...

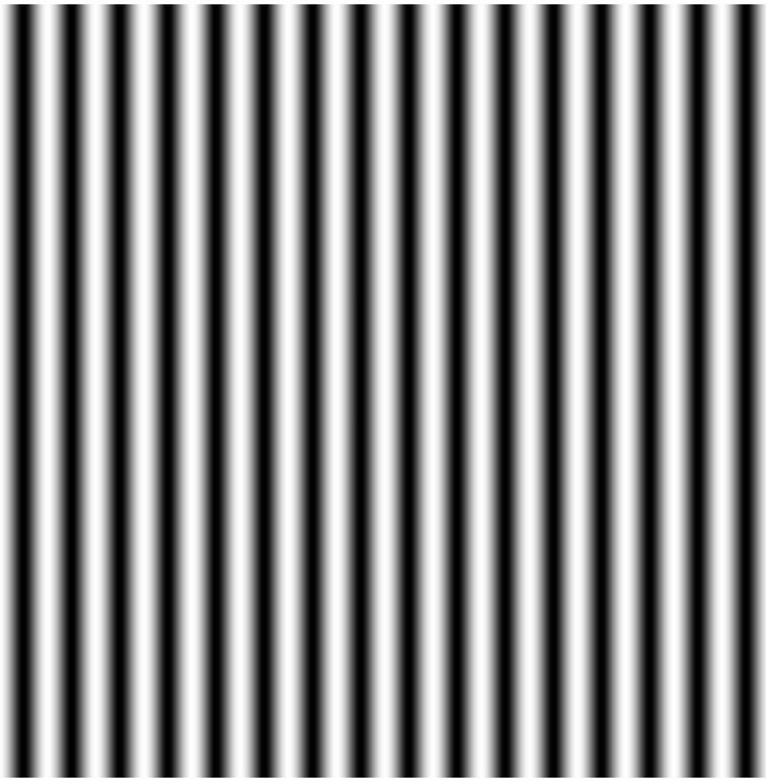
Constant

Spatial Domain

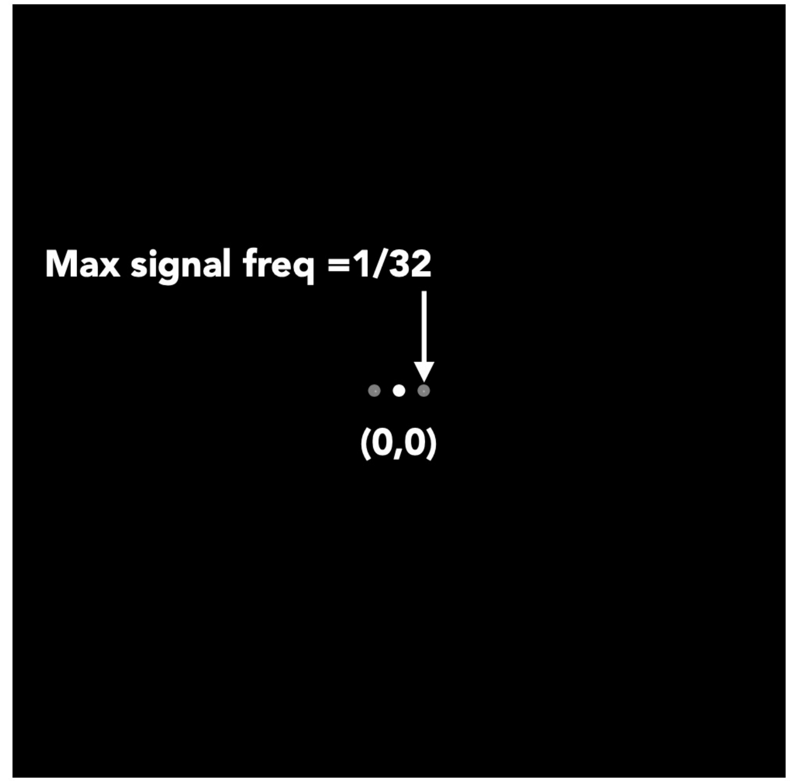


Frequency Domain

sin(2π/32)x – freq. 1/32; 32 pixels per cycle

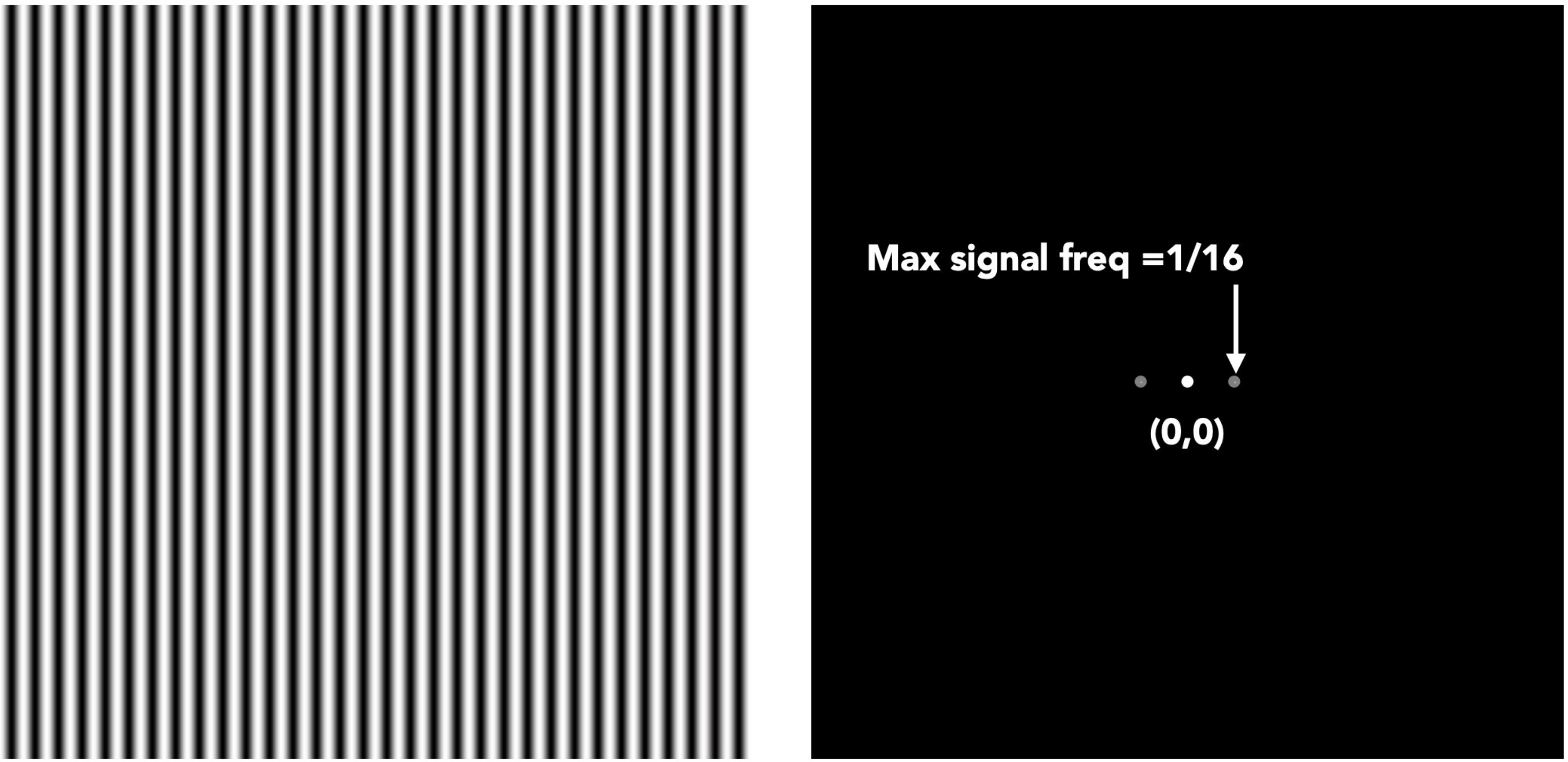


Spatial Domain



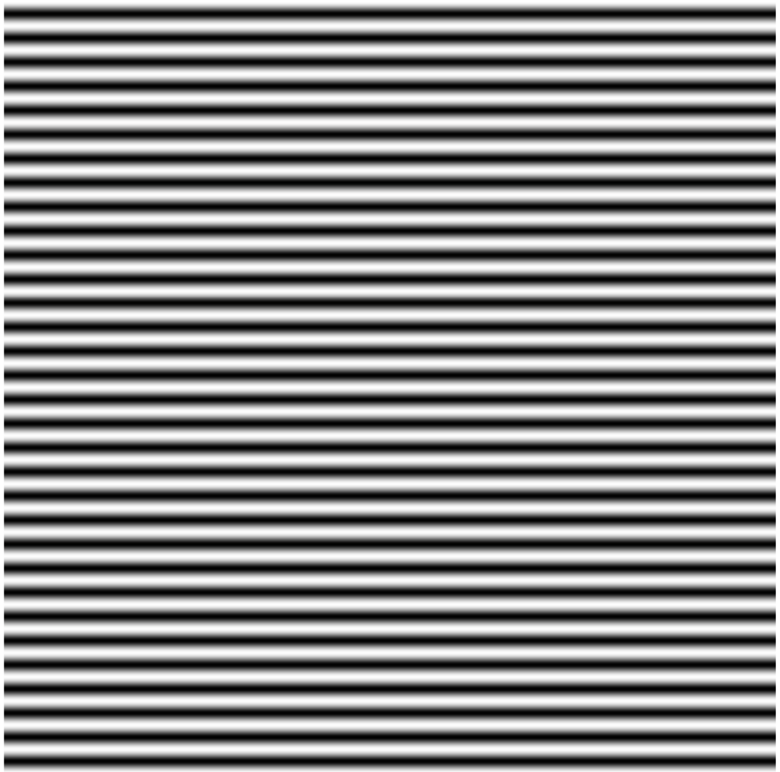
Frequency Domain

$\sin(2\pi/16)x$ – freq. 1/16; 16 pixels per cycle

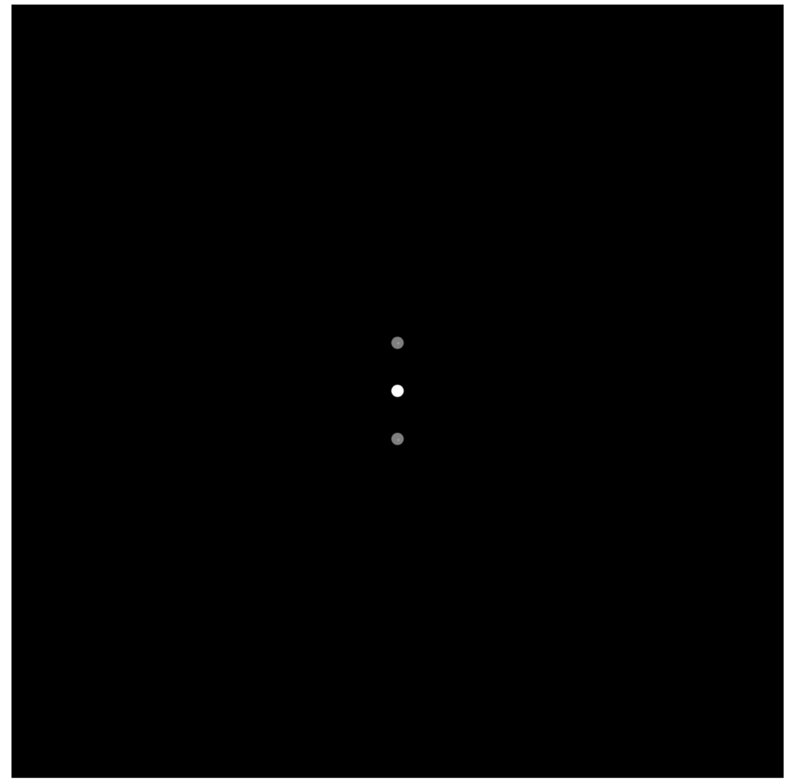


Spatial Domain Frequency Domain

$\sin(2\pi/16)y$

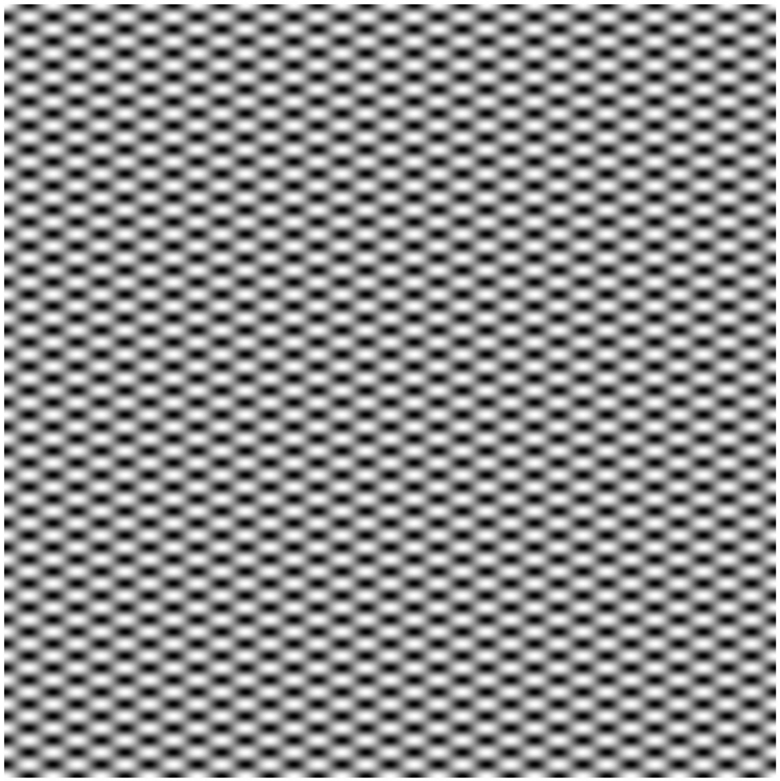


Spatial Domain

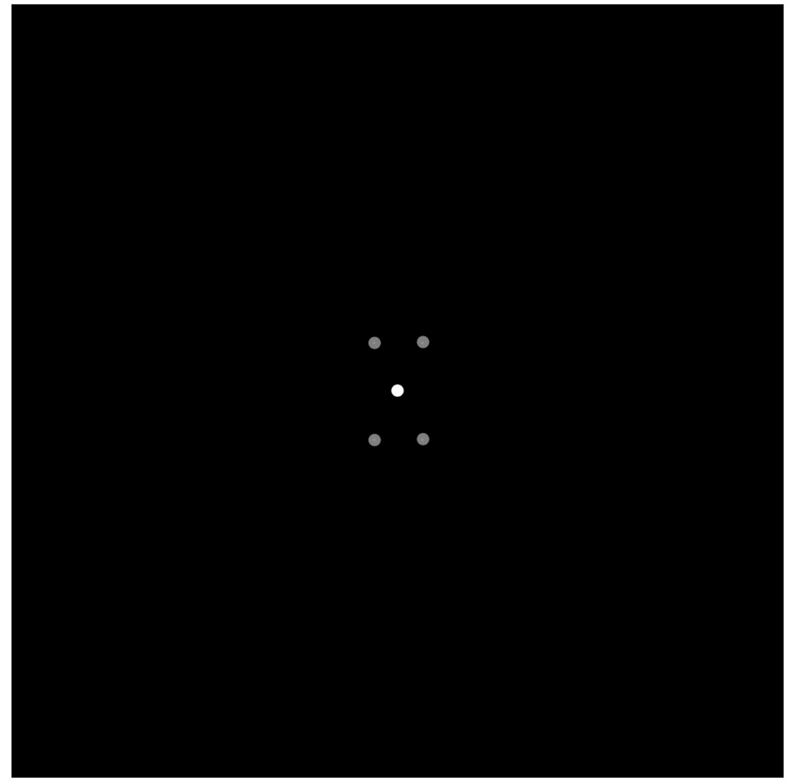


Frequency Domain

$\sin(2\pi/32)x \times \sin(2\pi/16)y$



Spatial Domain

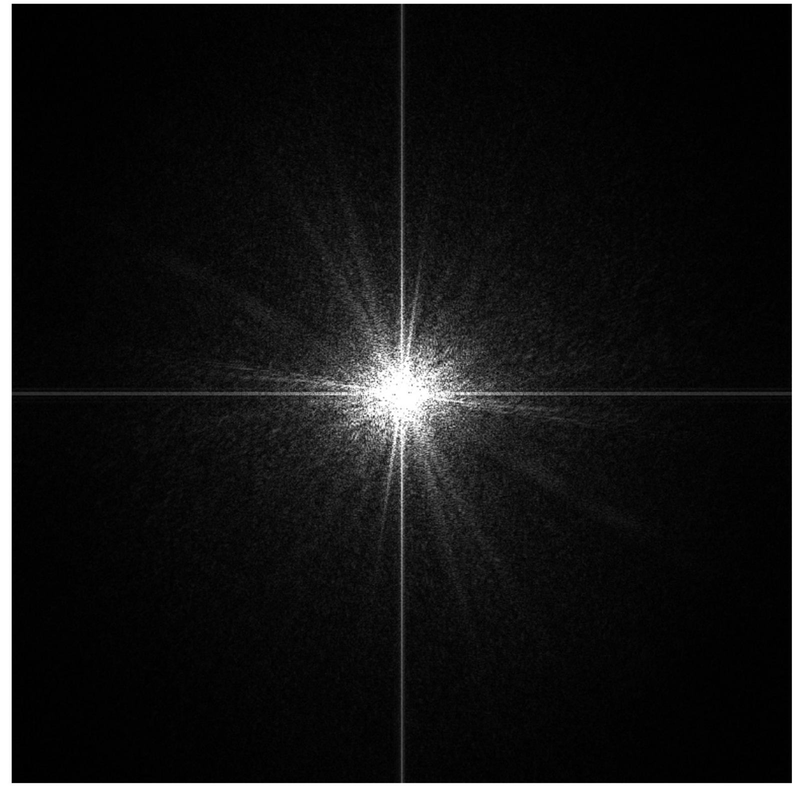


Frequency Domain

Filtering = Getting rid of certain frequency contents

可视化图像频率

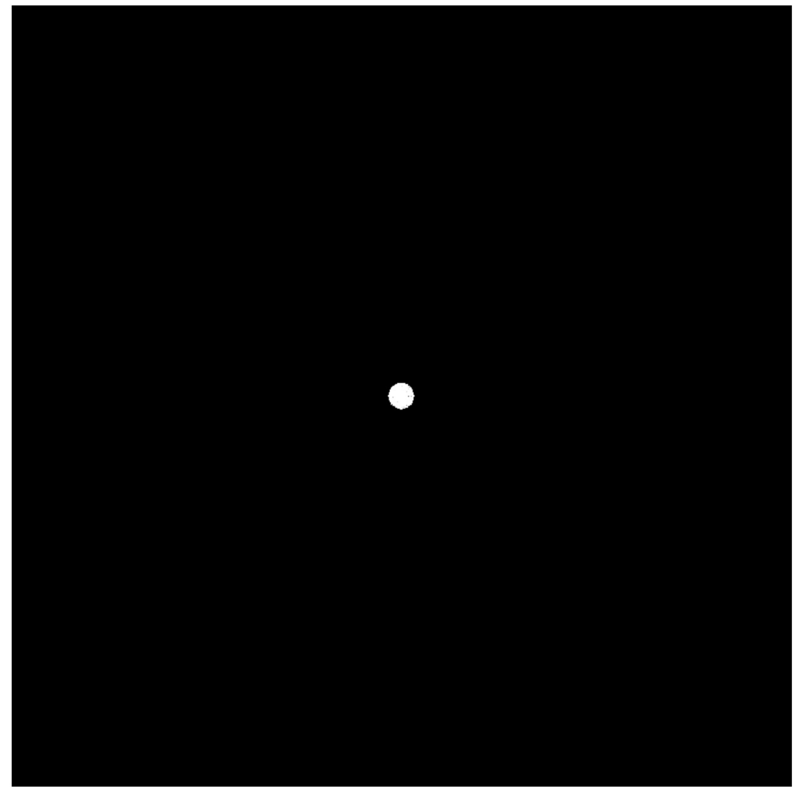
空域结果



频谱

过滤高频

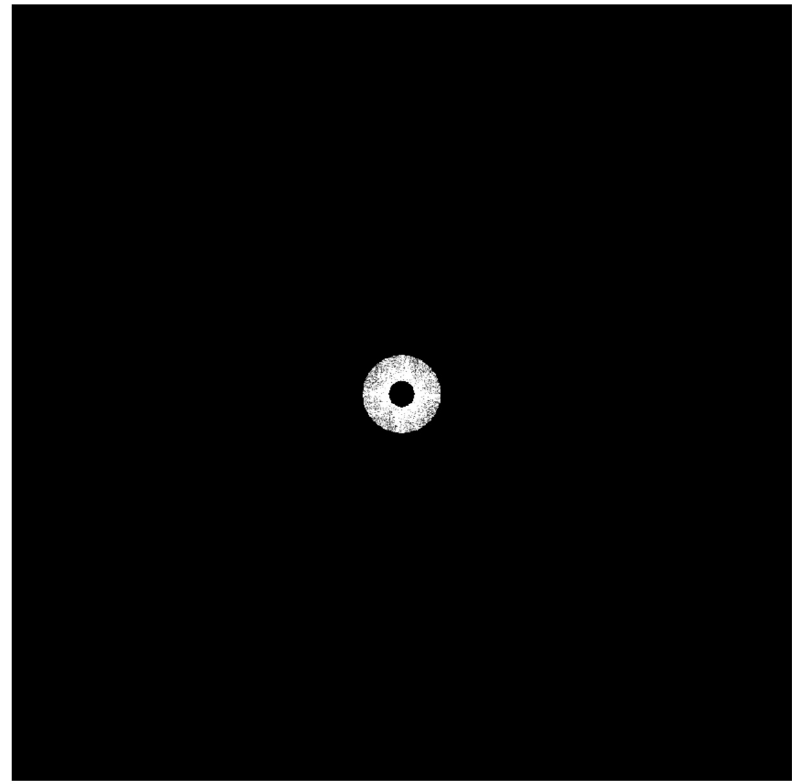
空域结果



频谱（低通滤波后）
高于阈值的频率的幅值为0

过滤低频和高频

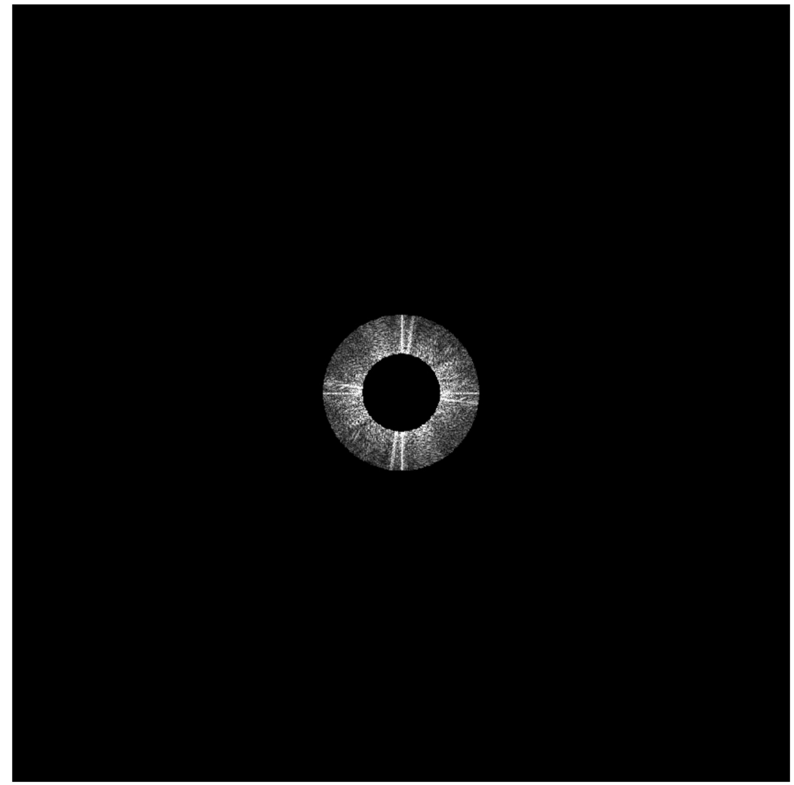
空域结果



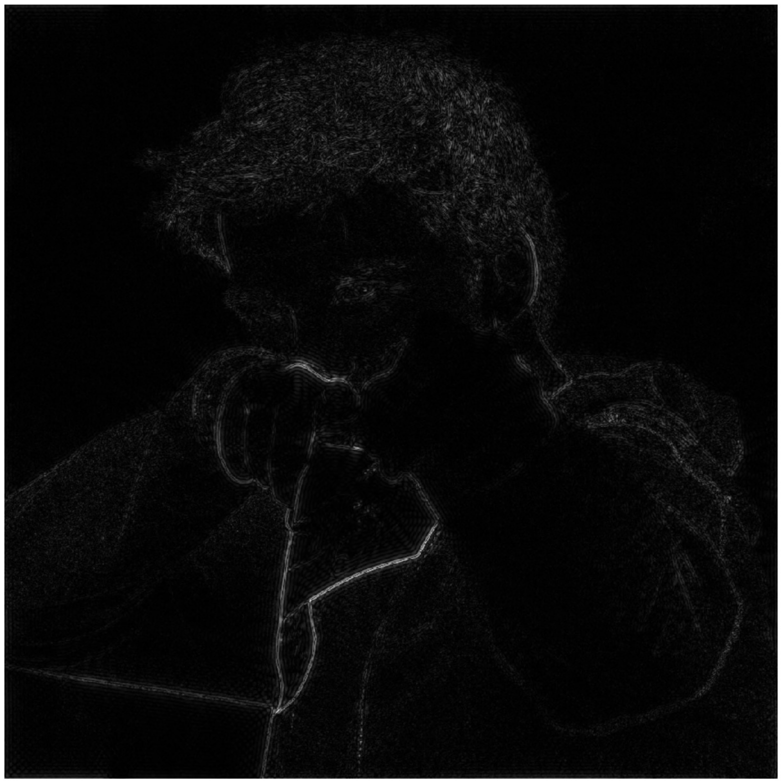
频谱（带通滤波后）

过滤低频和高频

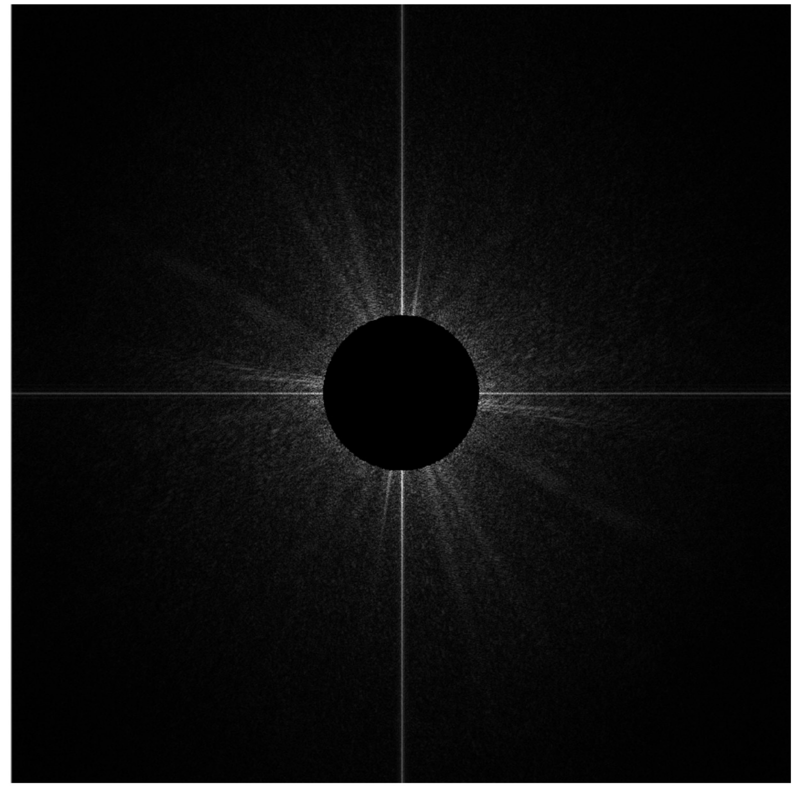

空域结果


频谱（带通滤波后）

过滤低频

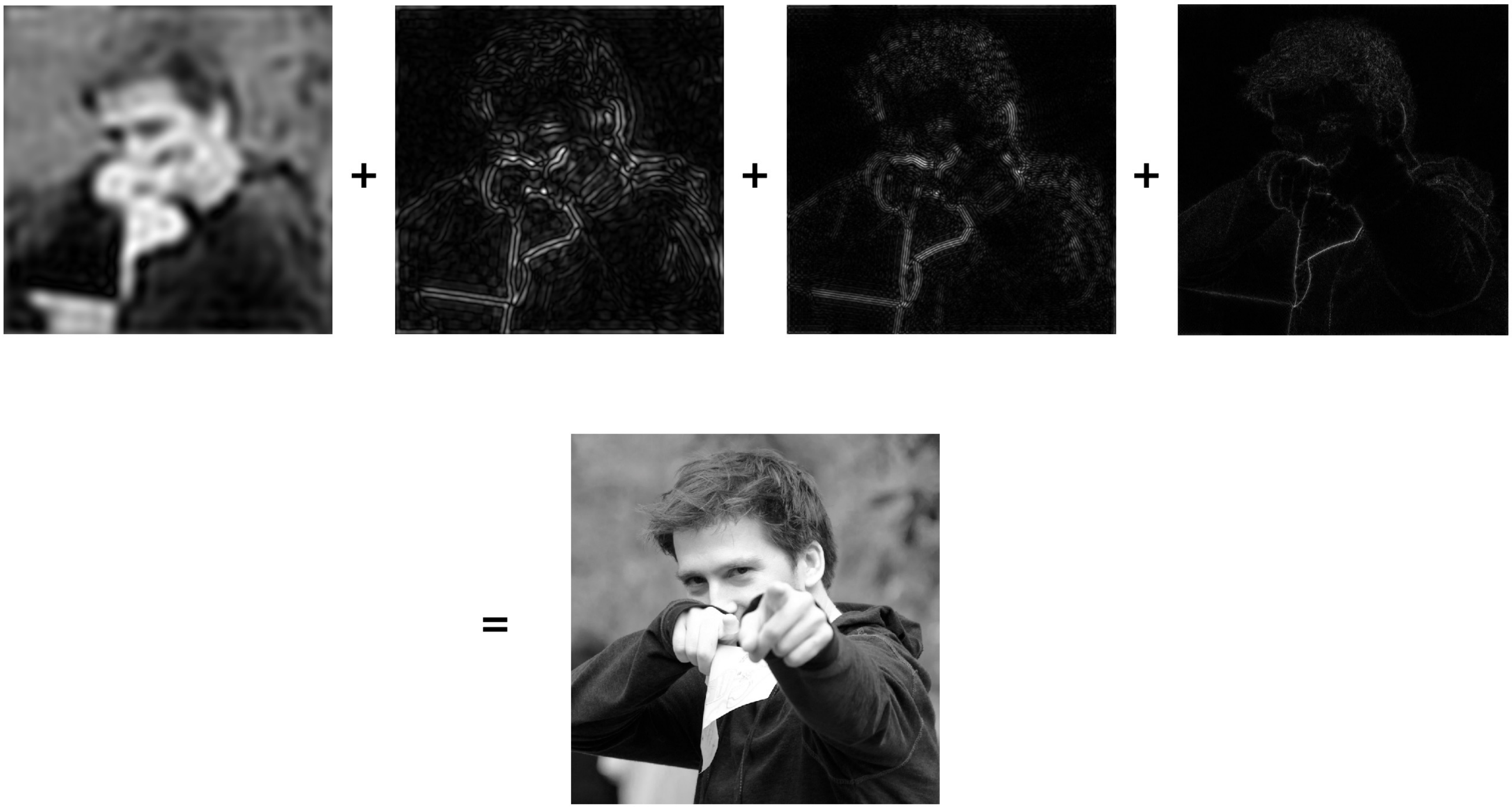


空域结果
(最锐利的边)



频谱（高通滤波后）
低于阈值的频率的幅值为0

图像分解为不同频率的部分



尼奎斯特定理 Nyquist Theorem

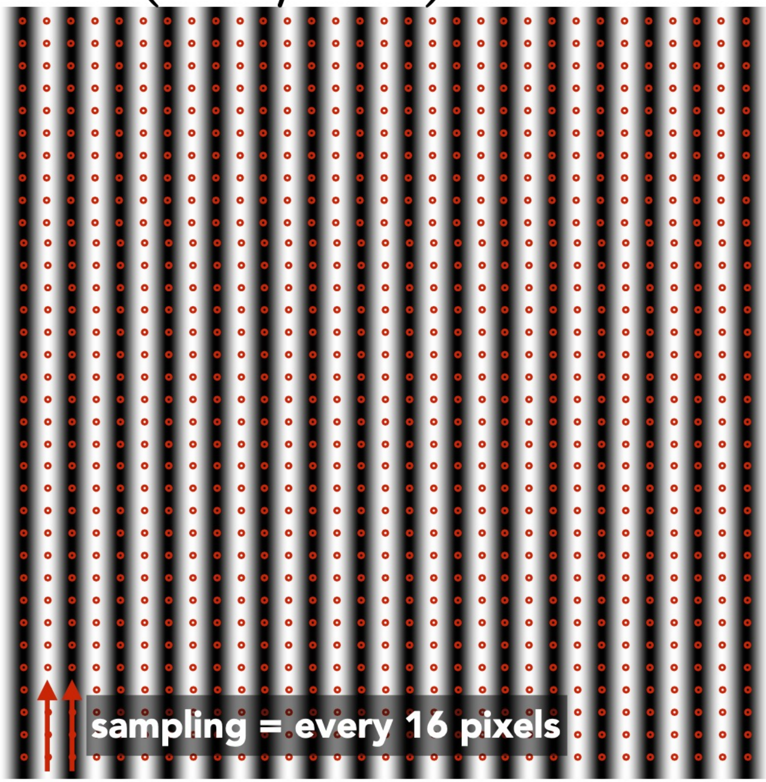
若信号的频率低于尼奎斯特频率（定义为采样频率的一半），则不会出现走样现象。

Theorem: We get no aliasing from frequencies in the signal that are less than the Nyquist frequency (which is defined as half the sampling frequency) *

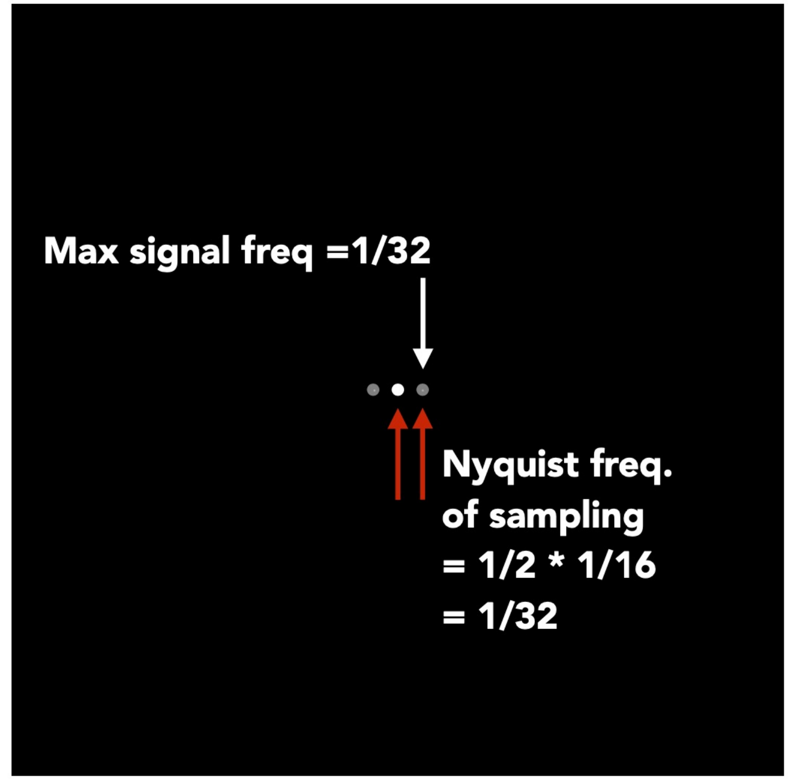
* See proof: https://en.wikipedia.org/wiki/Nyquist-Shannon_sampling_theorem

信号与尼奎斯特频率 – 例子

$\sin(2\pi/32)x$ — **frequency 1/32; 32 pixels per cycle**



Spatial Domain

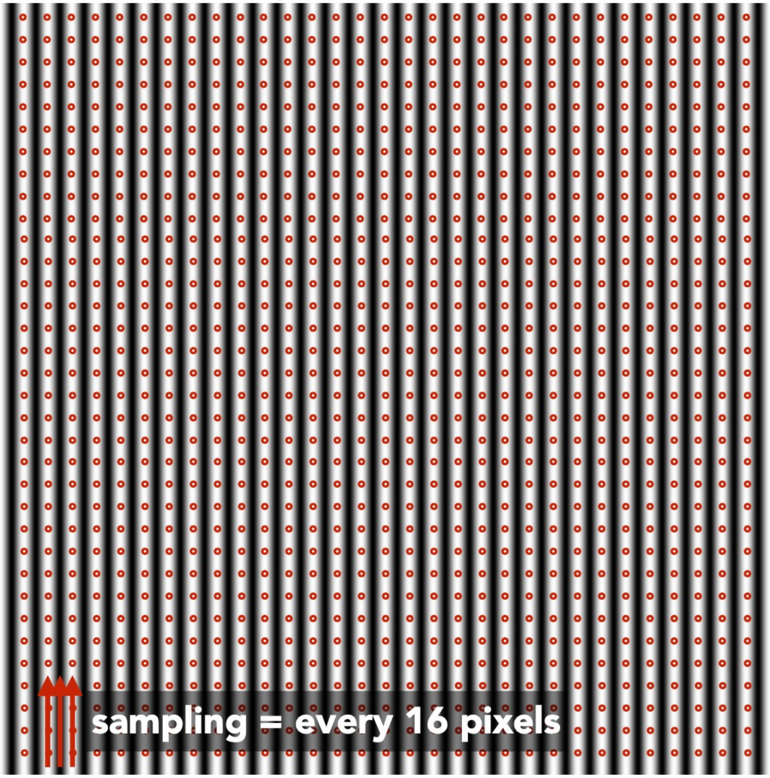


Frequency Domain

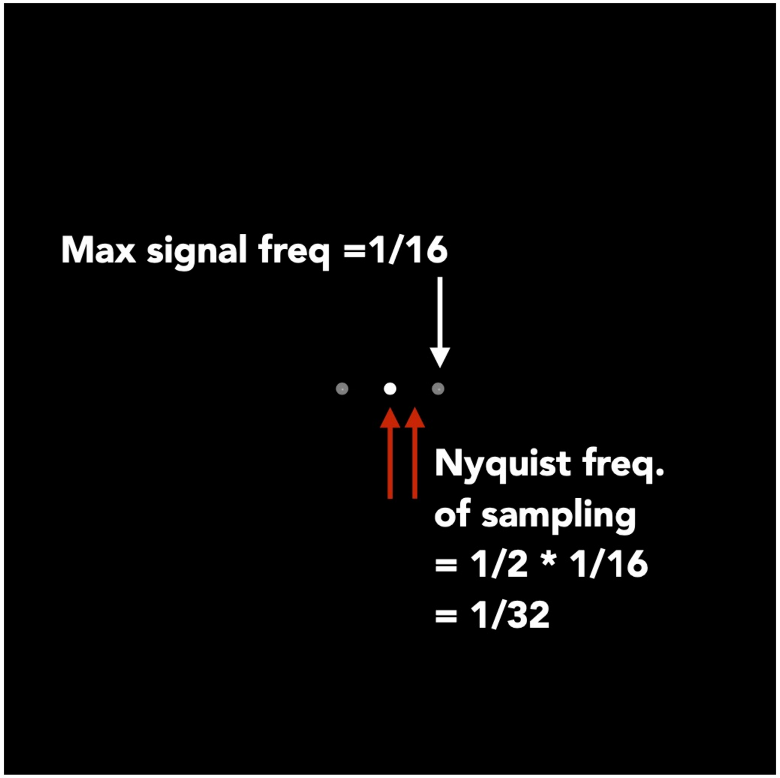
No aliasing!

信号与尼奎斯特频率 – 例子

$\sin(2\pi/16)x$ — **frequency 1/16; 16 pixels per cycle**



Spatial Domain



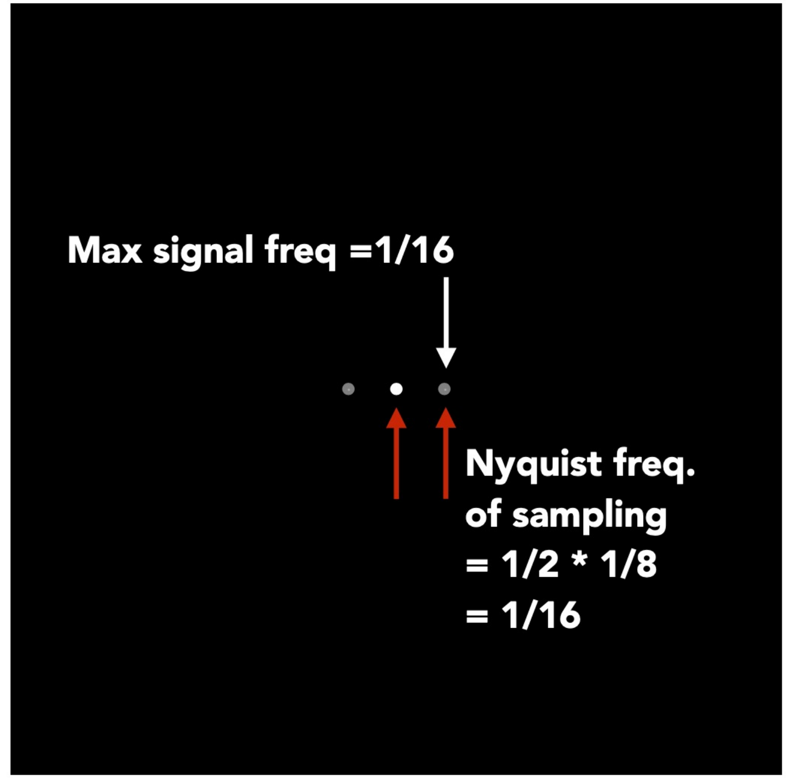
Frequency Domain

Aliasing!

信号与尼奎斯特频率 – 例子

$\sin(2\pi/16)x$ — **frequency 1/16; 16 pixels per cycle**

Spatial Domain



Frequency Domain

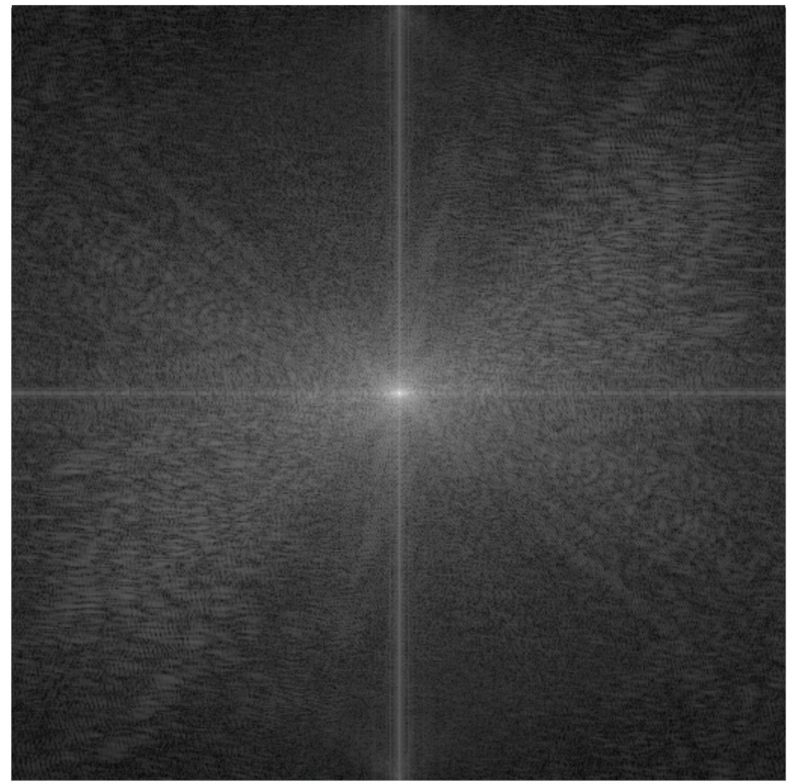
No aliasing!

图像频率：可视化例子

- 在接下来的一系列图片中
  - 图片的大小为 512 × 512 像素
  - 我们将逐渐模糊图像，看看频谱是如何缩小的，看看最大频率是多少

图像频率：可视化例子

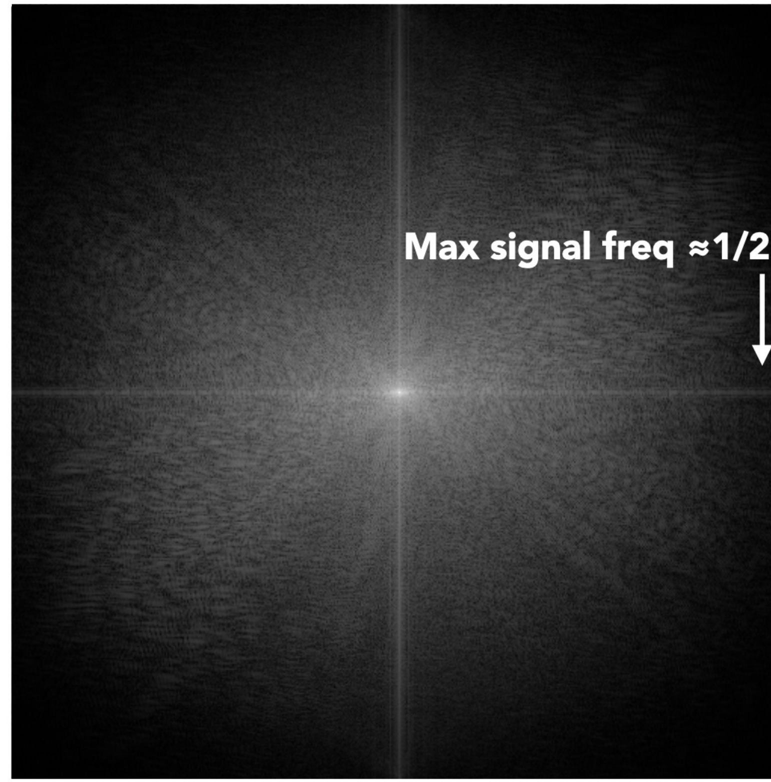
Spatial Domain



Frequency Domain

图像频率：可视化例子

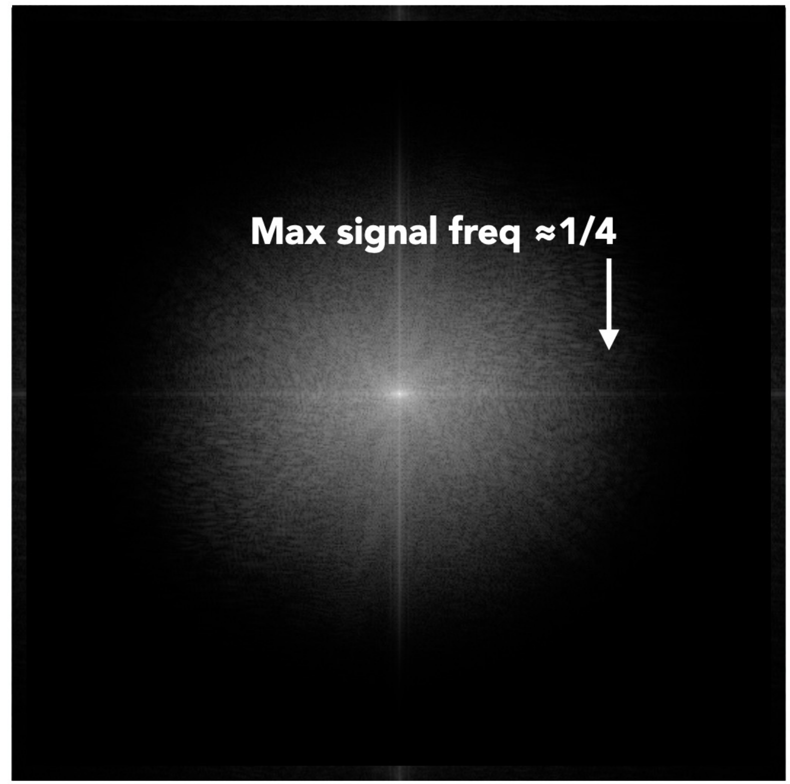
Spatial Domain



Frequency Domain

图像频率：可视化例子

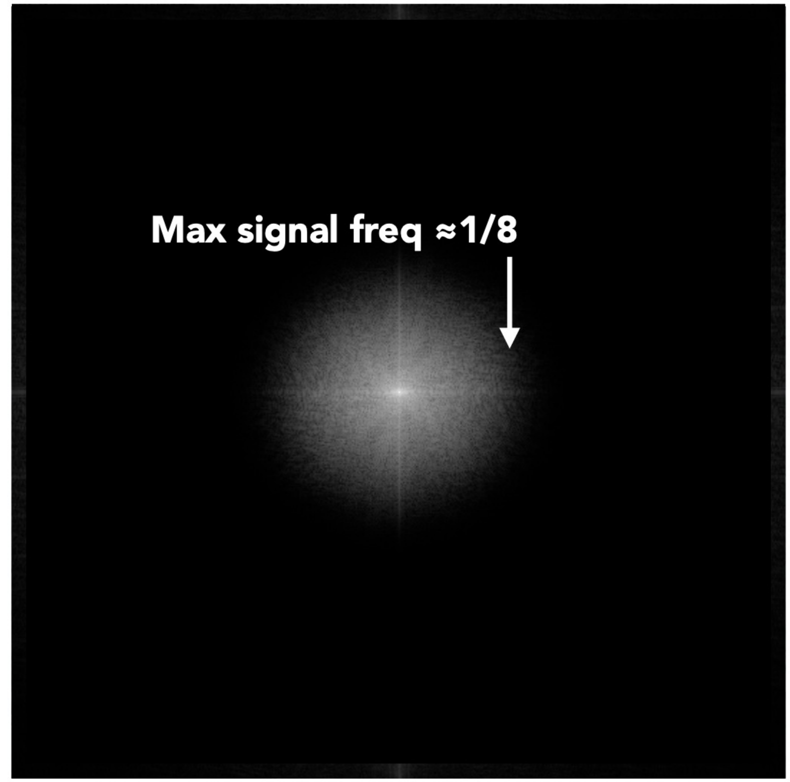
Spatial Domain



Frequency Domain

图像频率：可视化例子

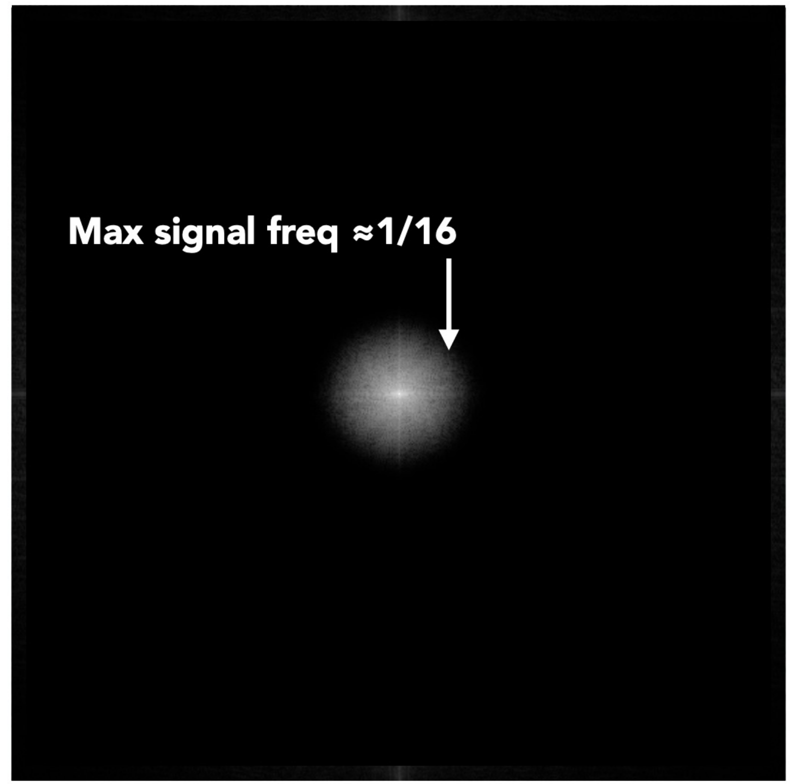
Spatial Domain



Frequency Domain

图像频率：可视化例子

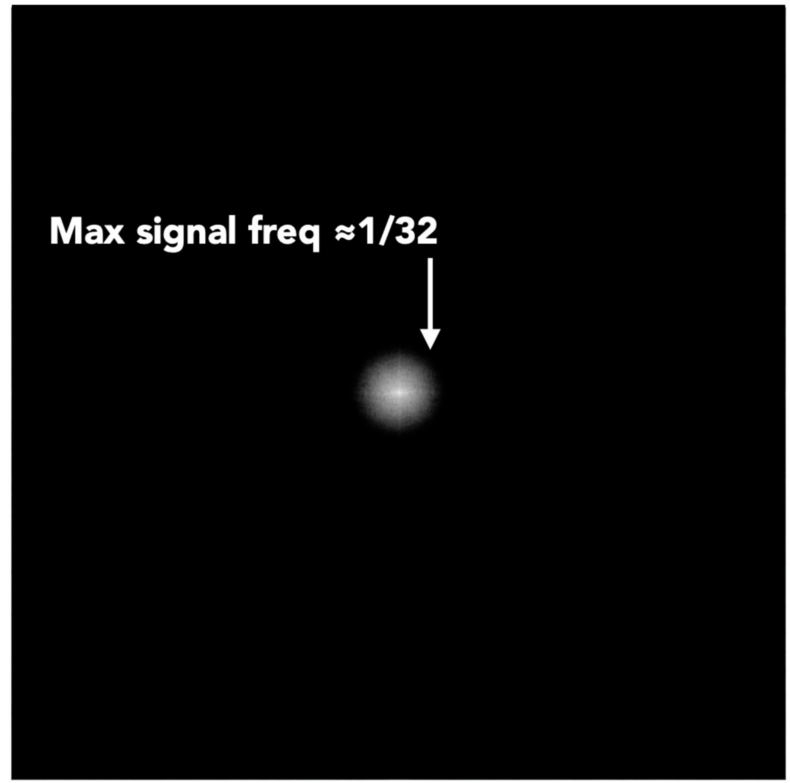
Spatial Domain



Frequency Domain

图像频率：可视化例子

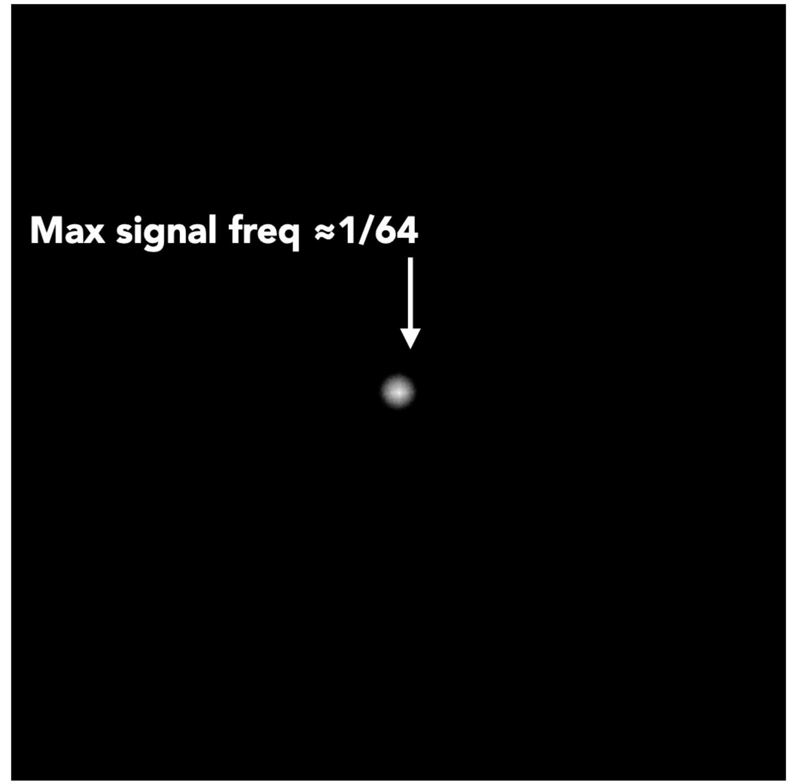
Spatial Domain



Frequency Domain

图像频率：可视化例子

Spatial Domain



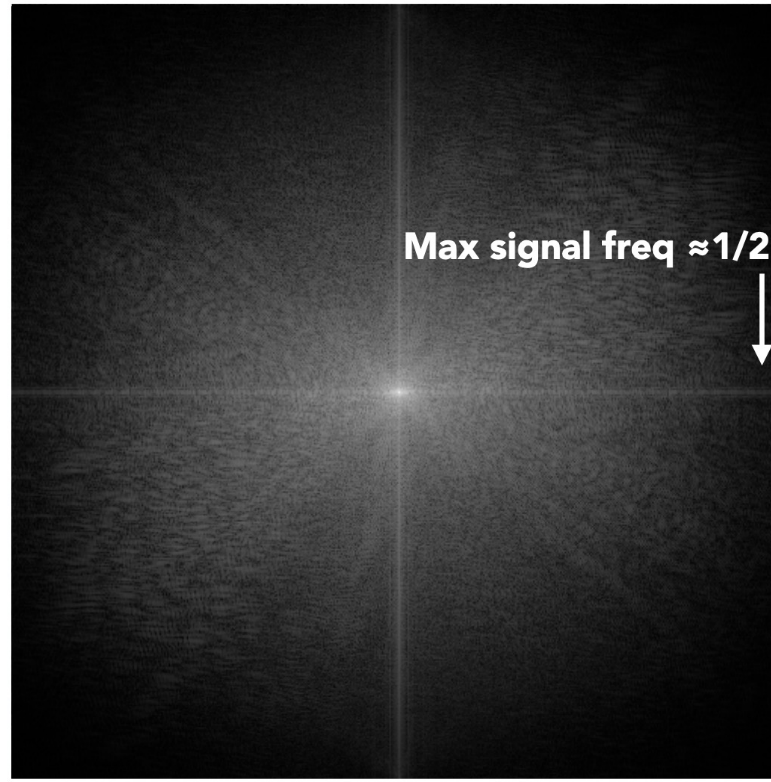
Frequency Domain

尼奎斯特频率：可视化例子

- 在接下来的一系列图片中
  - 每 16 个像素对图像进行可视化采样
  - 图片将逐渐模糊，直至其频率满足尼奎斯特频率 (因此无走样现象)

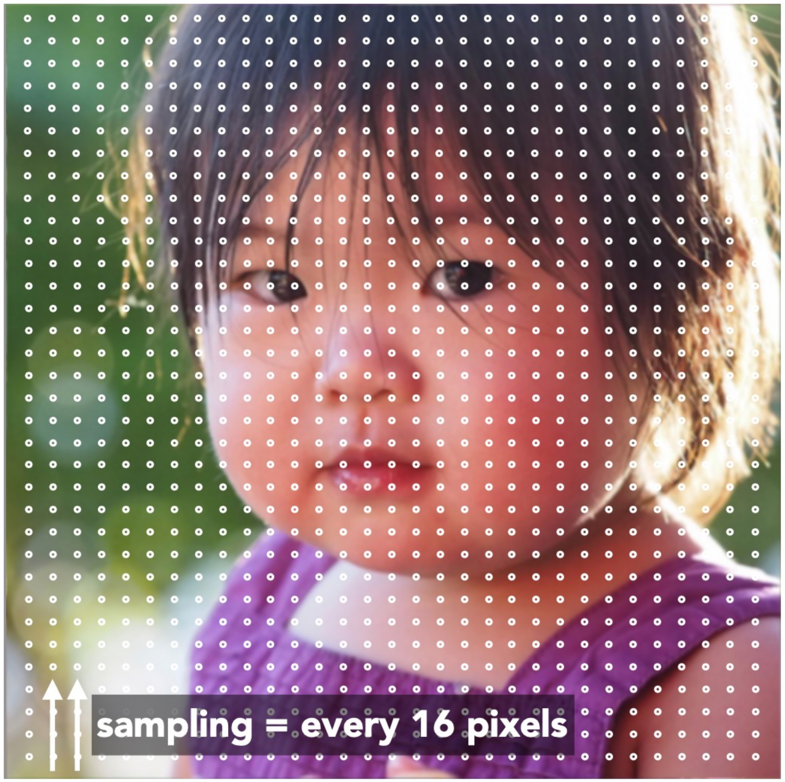
尼奎斯特频率：可视化例子

Spatial Domain

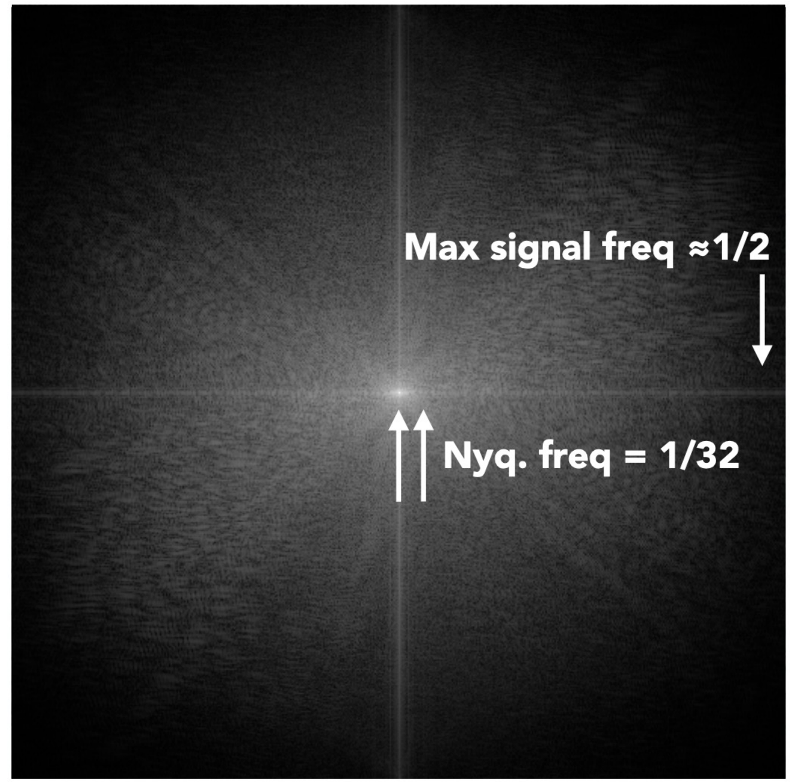


Frequency Domain

尼奎斯特频率：可视化例子

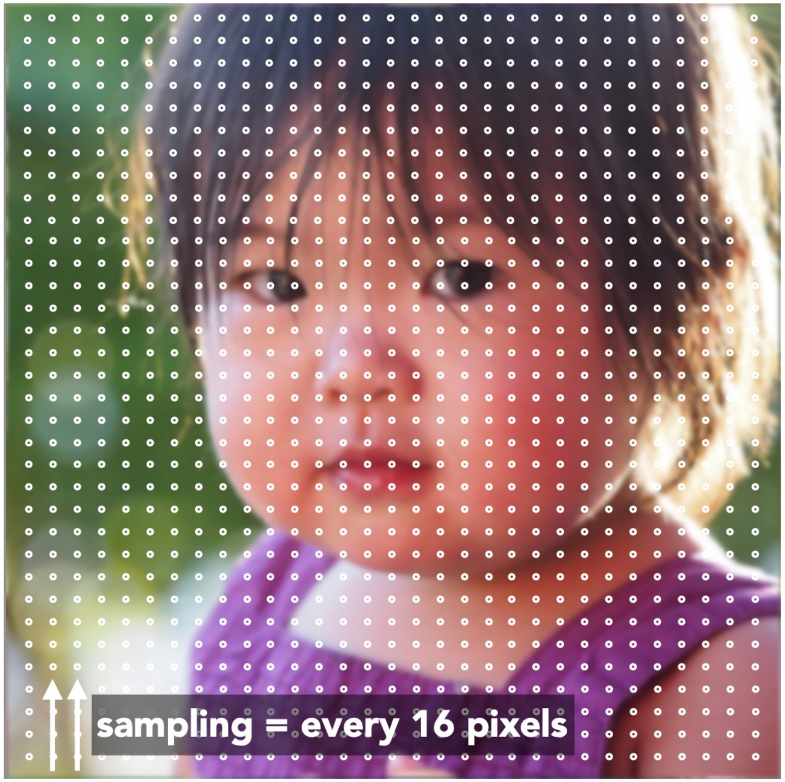


Spatial Domain

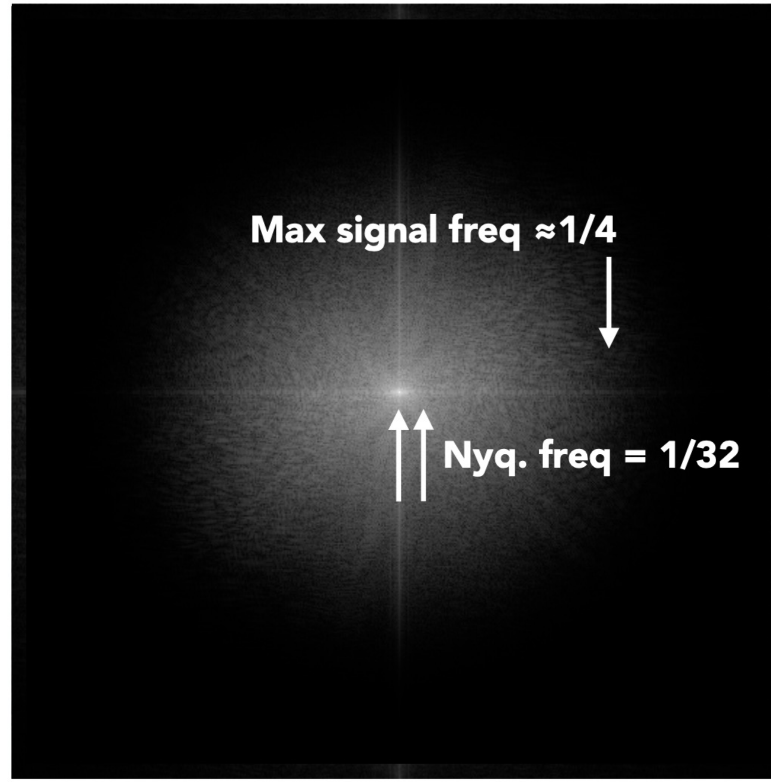


Frequency Domain

尼奎斯特频率：可视化例子

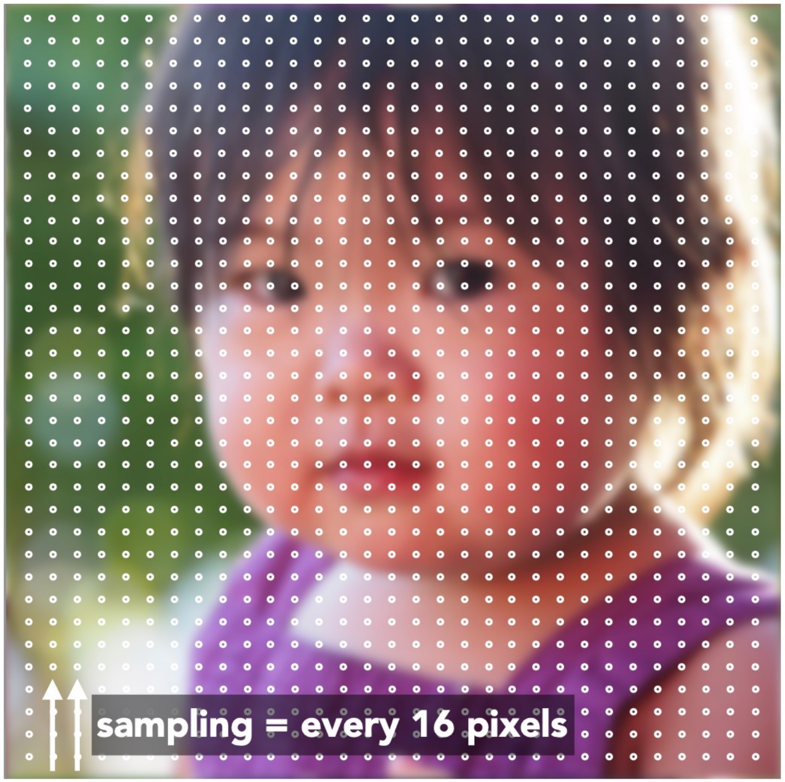


Spatial Domain

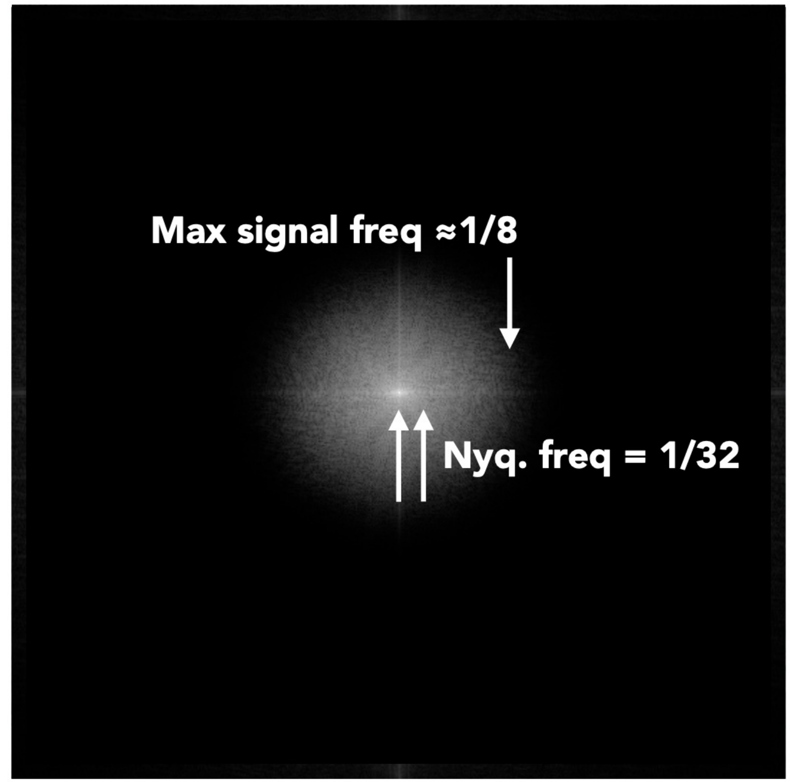


Frequency Domain

尼奎斯特频率：可视化例子

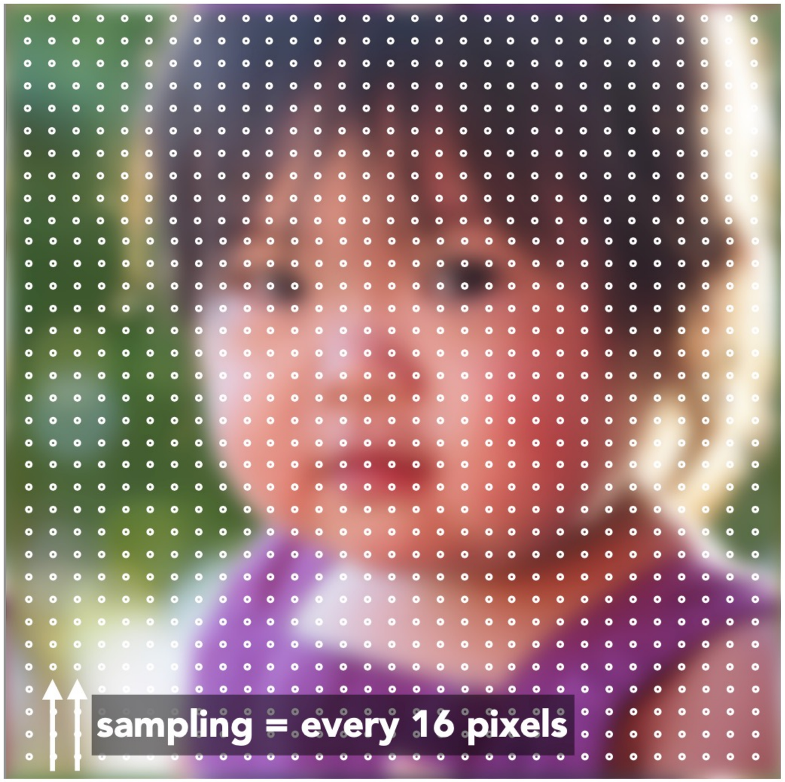


Spatial Domain

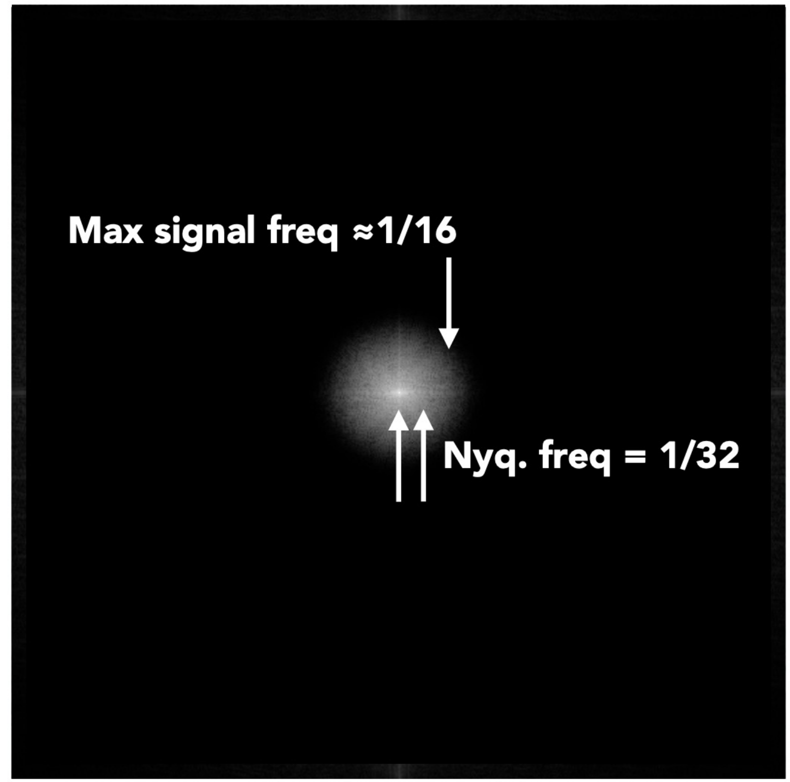


Frequency Domain

尼奎斯特频率：可视化例子

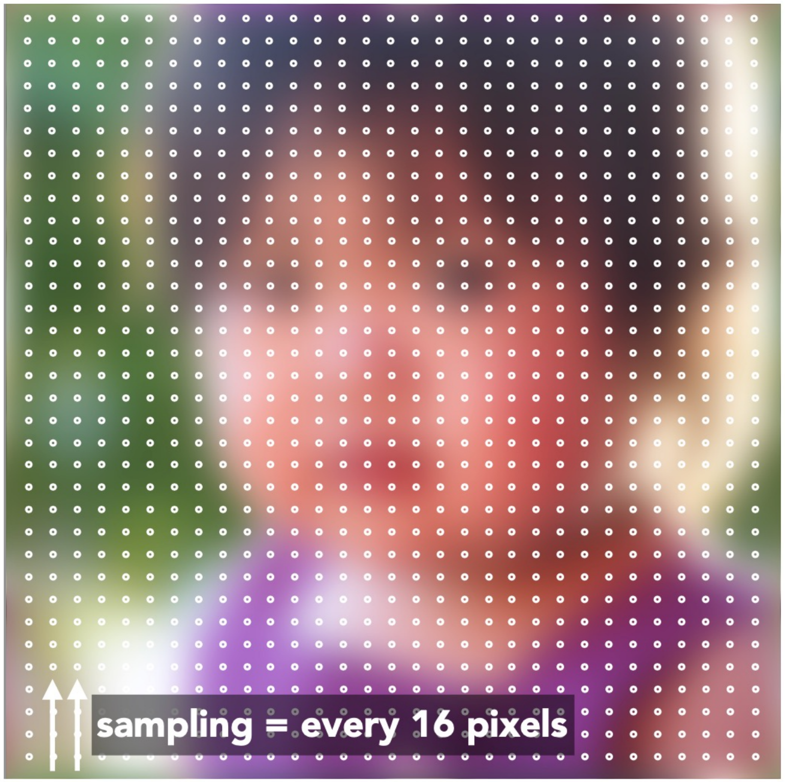


Spatial Domain

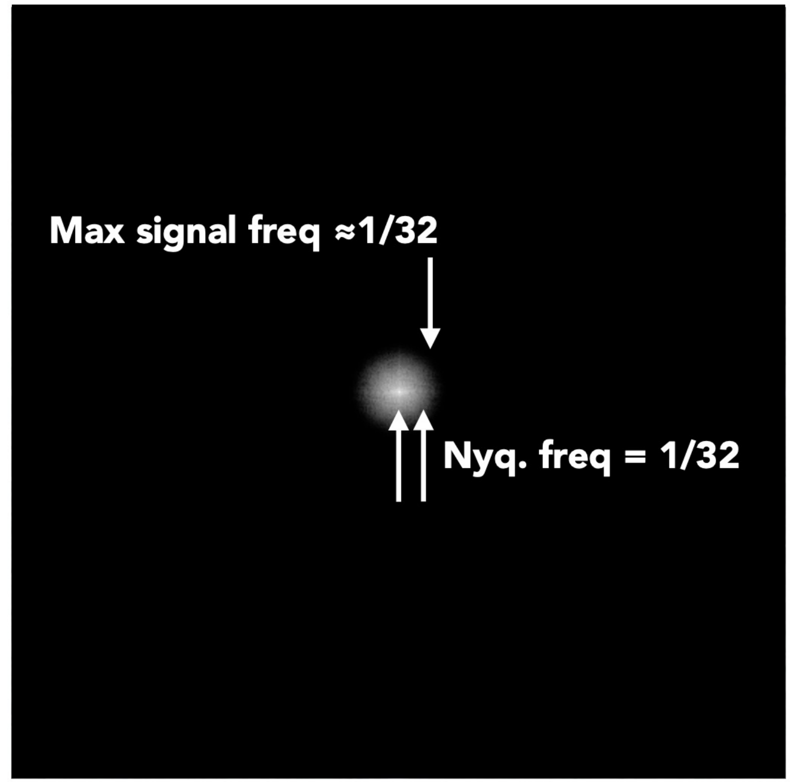


Frequency Domain

尼奎斯特频率：可视化例子

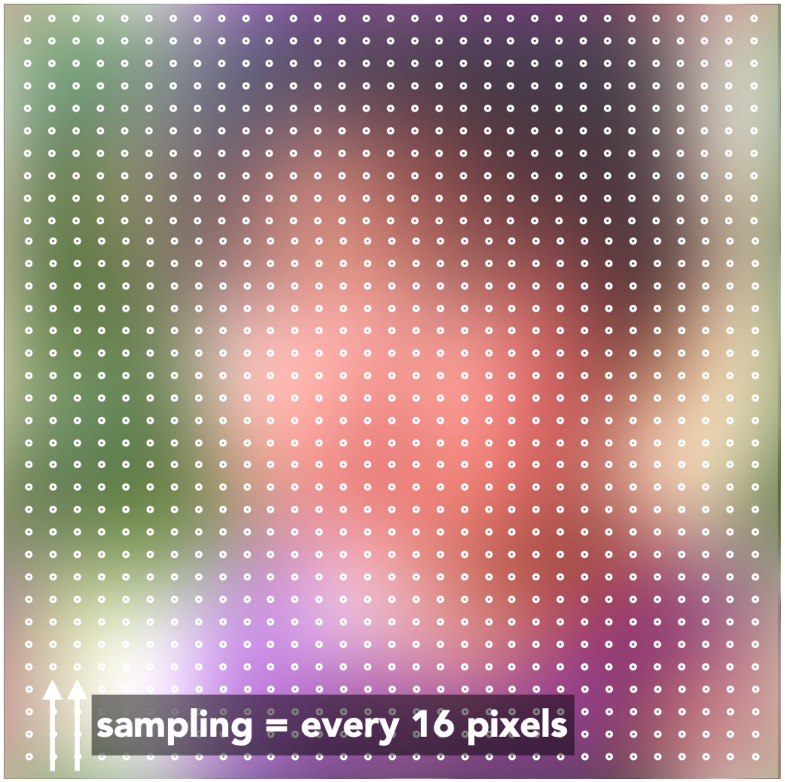


Spatial Domain

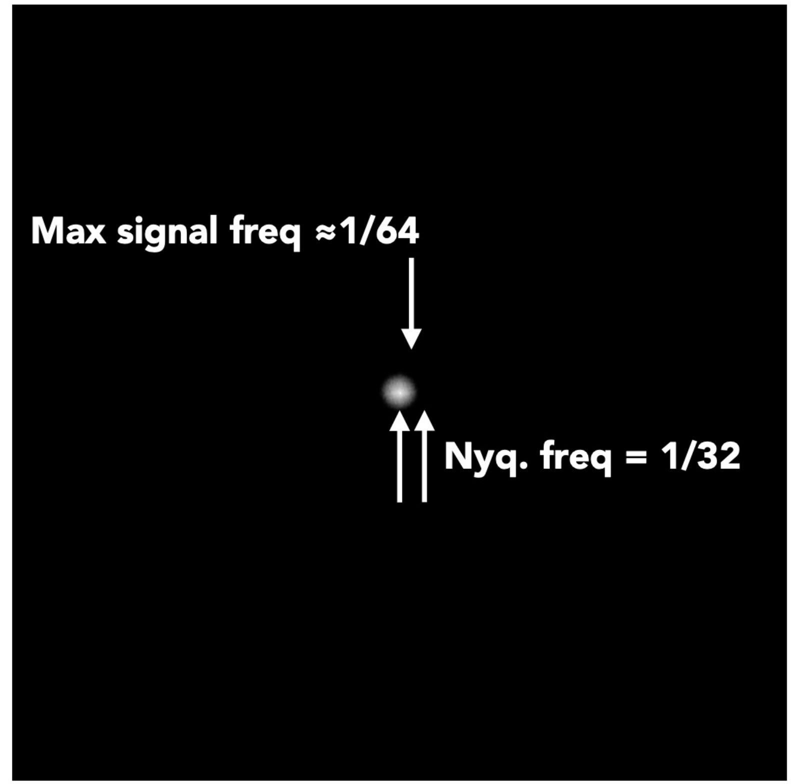


Frequency Domain

尼奎斯特频率：可视化例子



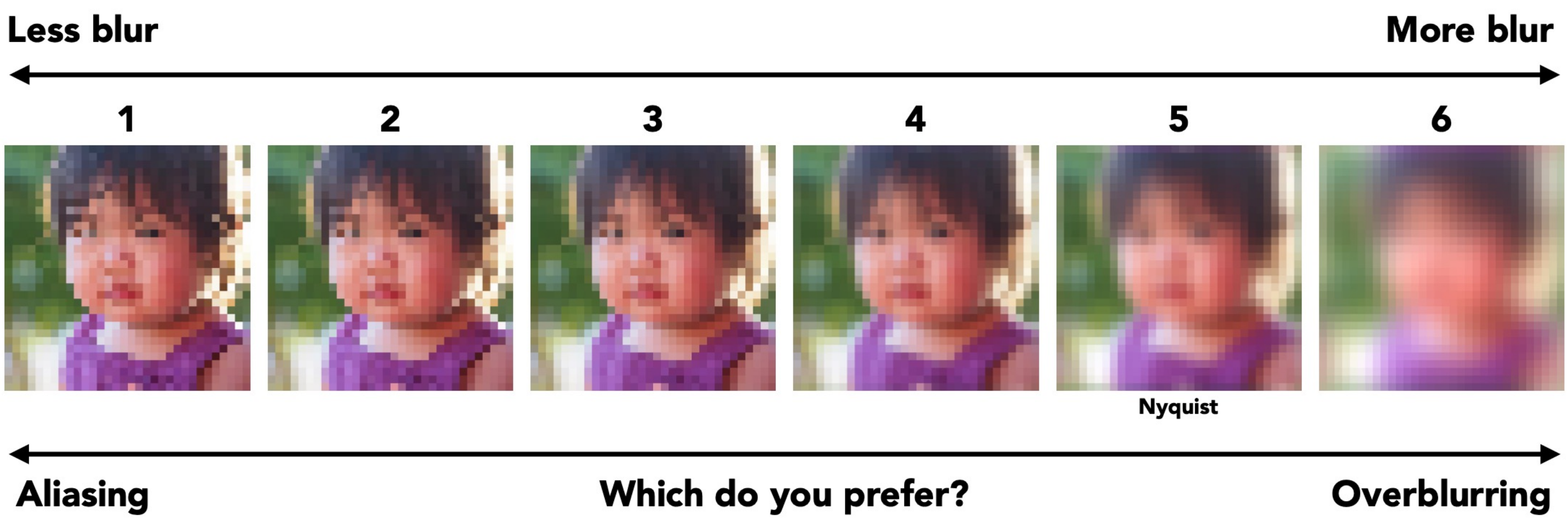
Spatial Domain



Frequency Domain

尼奎斯特频率：可视化例子

- 过滤（模糊化）降低原始图像的最大信号频率
- 通过每 16 个像素采样创建低分辨率图像
  - 采样频率为1/16，奈奎斯特频率为1/32



走样和过度模糊的判断可能很主观

Filtering = Convolution
(= Averaging)

Convolution

Signal	1	3	5	3	7	1	3	8	6	4
Filter	1/4	1/2	1/4							

Point-wise local averaging in a “sliding window”

Convolution

Signal

1	3	5	3	7	1	3	8	6	4

Filter

1/4	1/2	1/4

$$1 \times (1/4) + 3 \times (1/2) + 5 \times (1/4) = 3$$

Result

	3								

Convolution

Signal	1	3	5	3	7	1	3	8	6	4
Filter		1/4	1/2	1/4						

$$3 \times (1/4) + 5 \times (1/2) + 3 \times (1/4) = 4$$

Result		3	4							

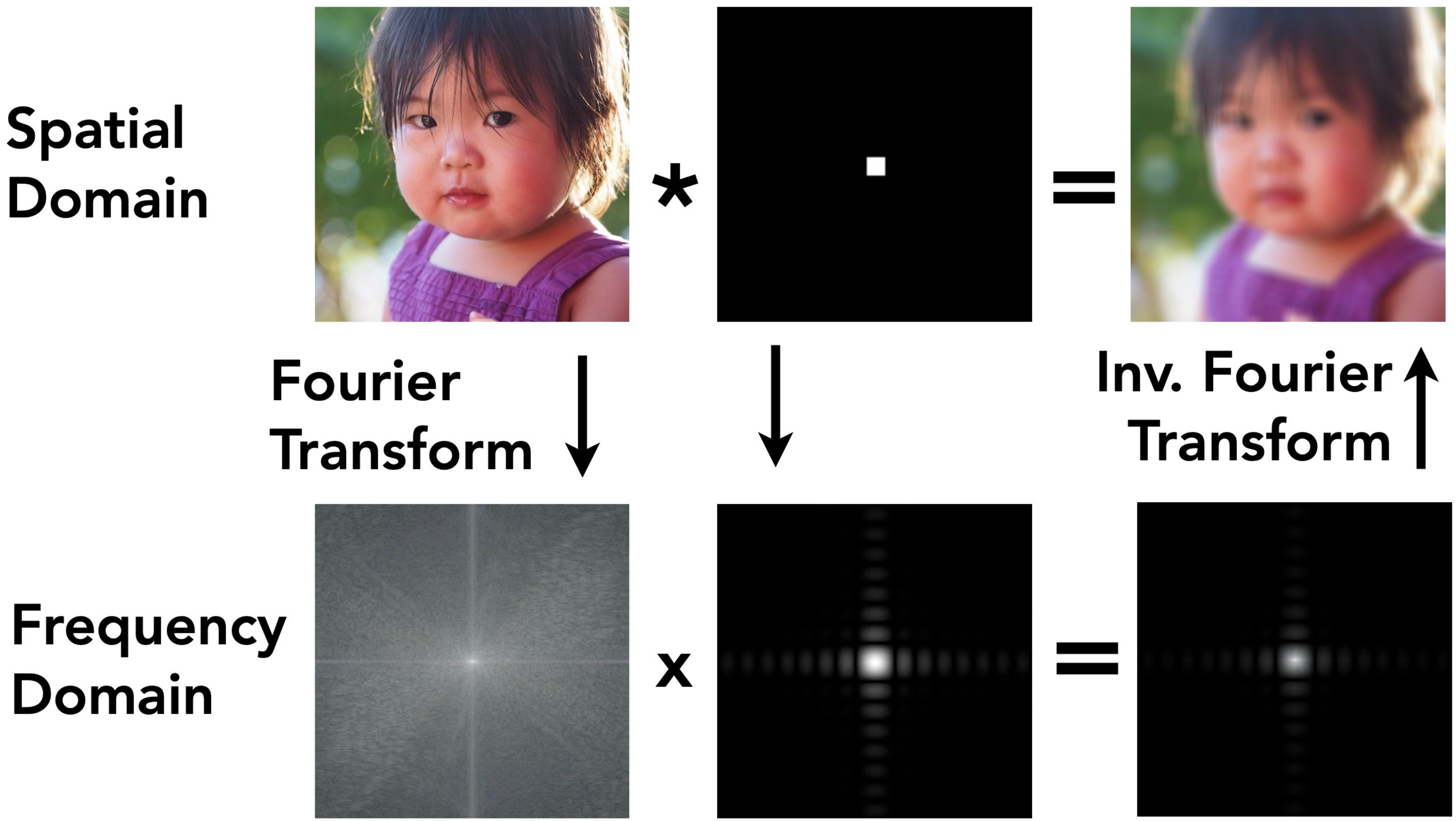
卷积定理 Convolution theorem

- 空间域中的卷积等于频率域中的乘法，反之亦然

- Option 1
  - 在空间域中用卷积滤波

- Option 2
  - 变换到频域（傅立叶变换）
  - 乘上卷积核的傅立叶变换
  - 变换回空域（傅立叶逆变换）

卷积定理

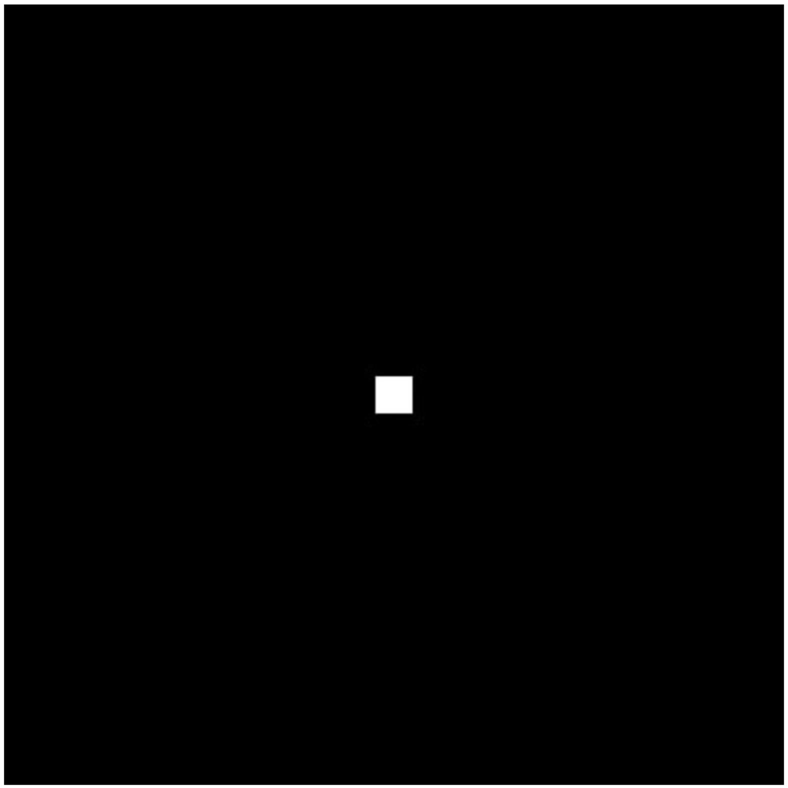

Spatial Domain
*
=
Fourier Transform
Inv. Fourier Transform
Frequency Domain
x
=

盒滤波器 Box filter

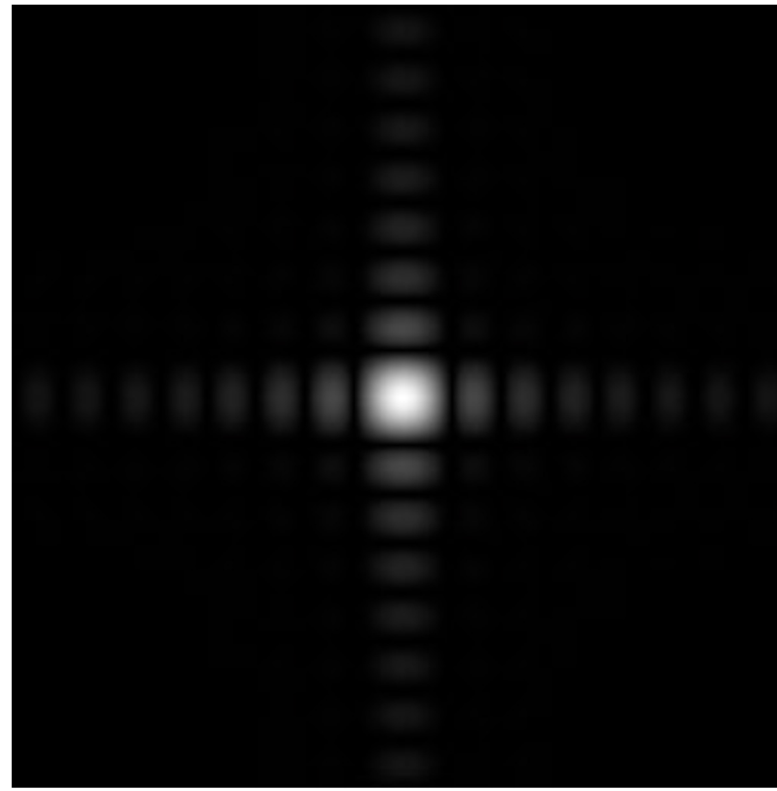
$$\frac{1}{9}\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

Example: 3x3 box filter

盒函数 = 低通滤波器

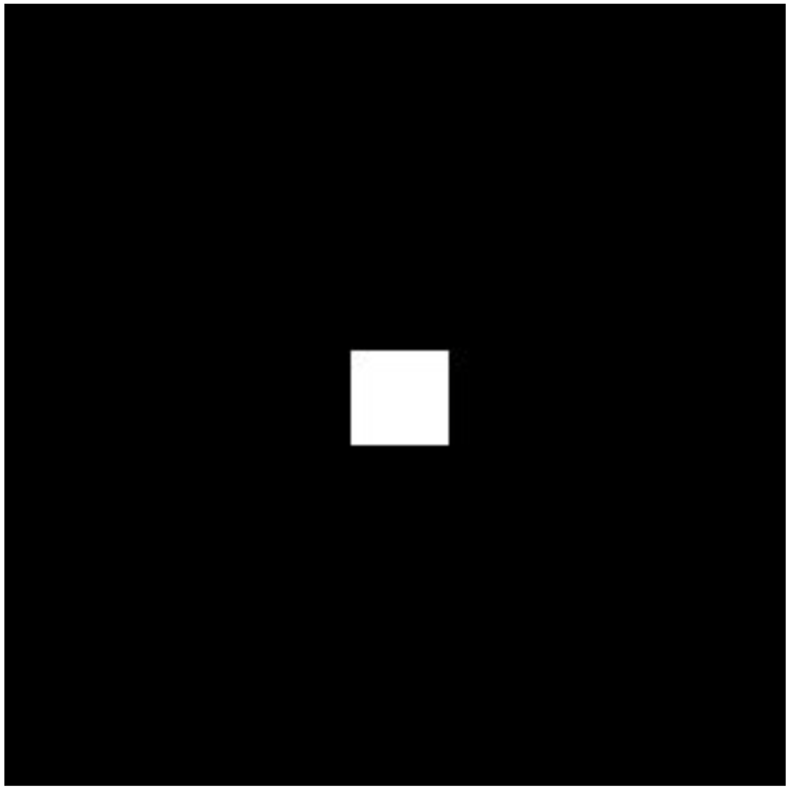


Spatial Domain

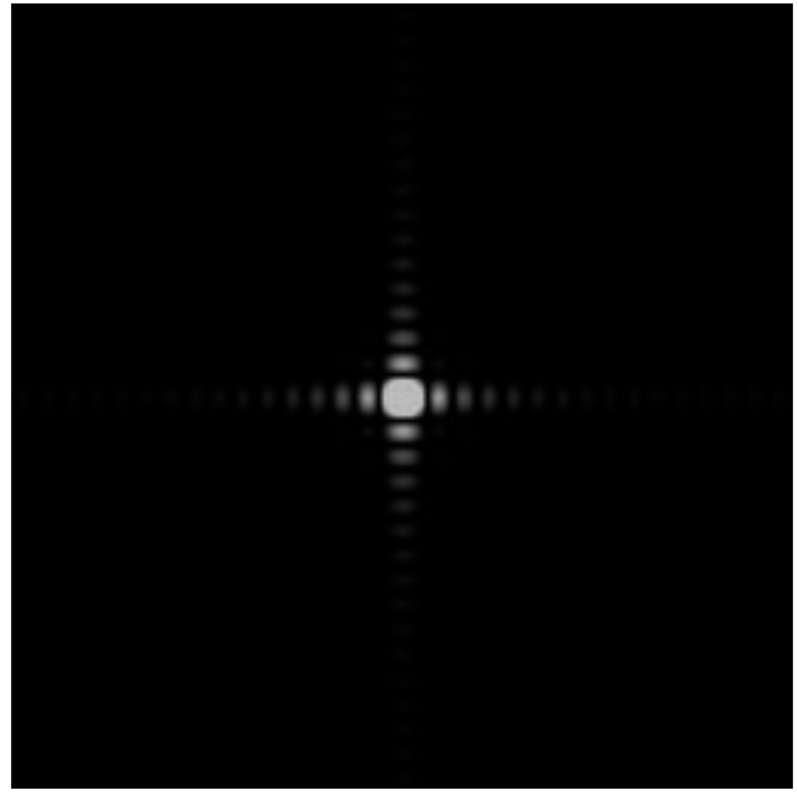


Frequency Domain

Wider Filter Kernel = Lower Frequencies



Spatial Domain



Frequency Domain

Antialiasing

抗走样

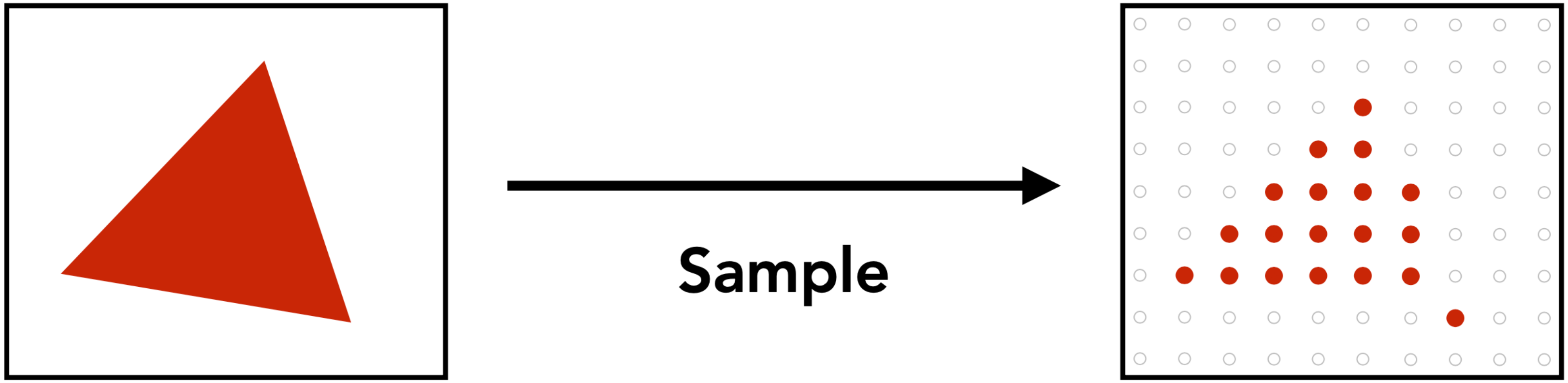
Theorem: We get no aliasing from frequencies in the signal that are less than the Nyquist frequency (which is defined as half the sampling frequency)

因此，以信号中最高频率的两倍
进行采样将消除走样现象

如何减少走样?

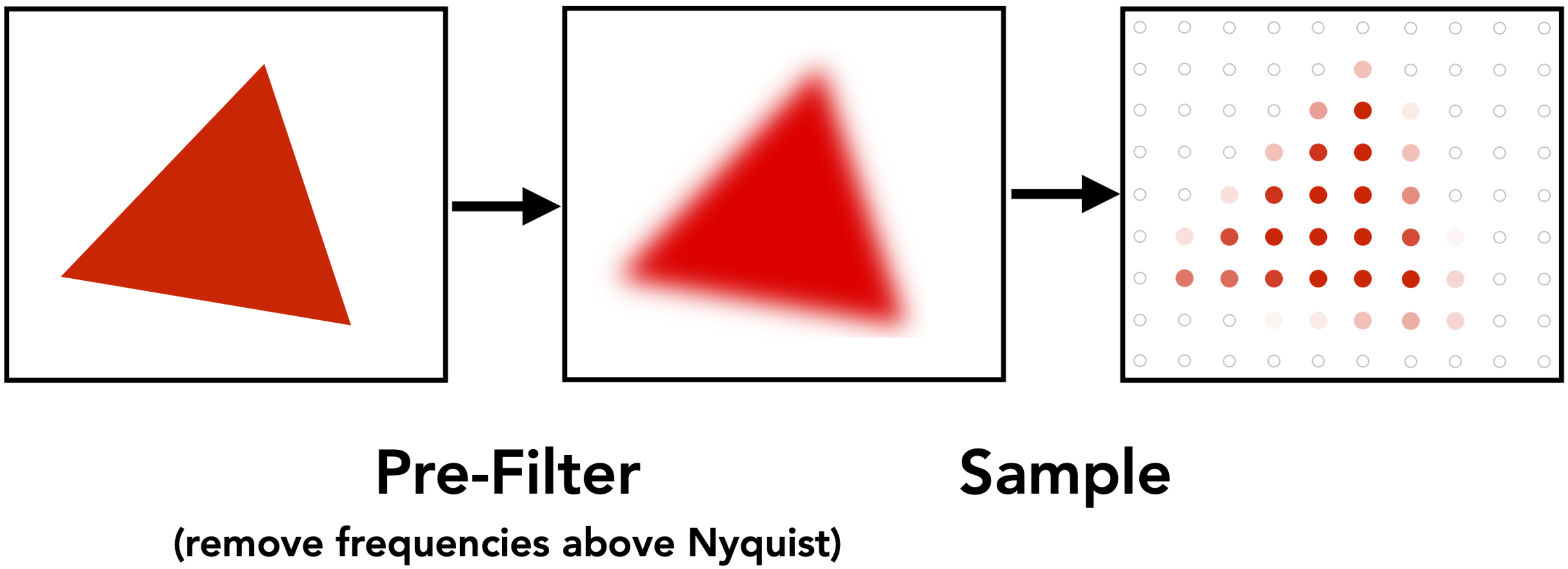
- 增加采样率（增加奈奎斯特频率）
  - 更高分辨率的显示器、传感器、帧缓冲区…
  - 但是：成本高昂，可能需要非常高的分辨率
- 抗走样
  - 在采样前去除（或降低）超过尼奎斯特频率的信号
  - 怎样在采样前滤除高频?

一般的采样操作



注意光栅化三角形中的锯齿为纯红色或白色

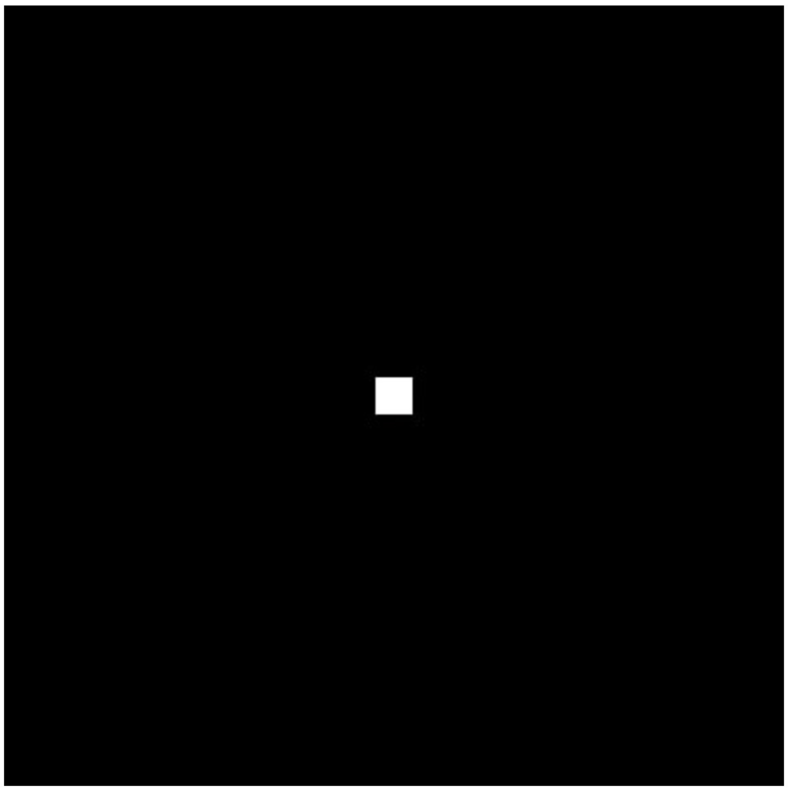
抗走样的采样



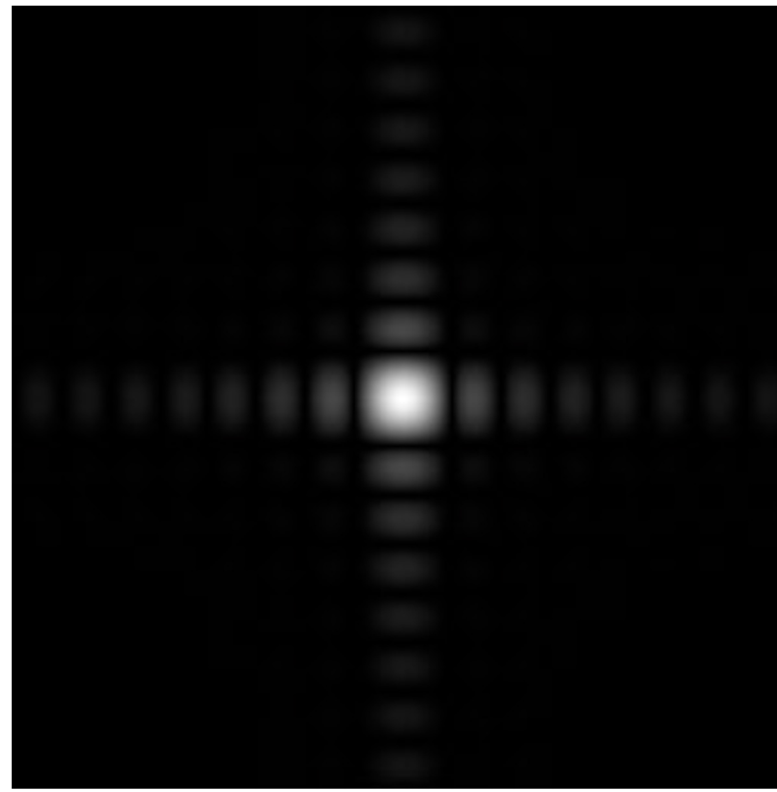
注意光栅化三角形中锯齿的值为红色与白色的中间值

一种实用的前置滤波器

- 一个 1 像素宽度的盒式滤波器 (box filter) 将过滤周期小于或等于 1 个像素宽度的频率



Spatial Domain

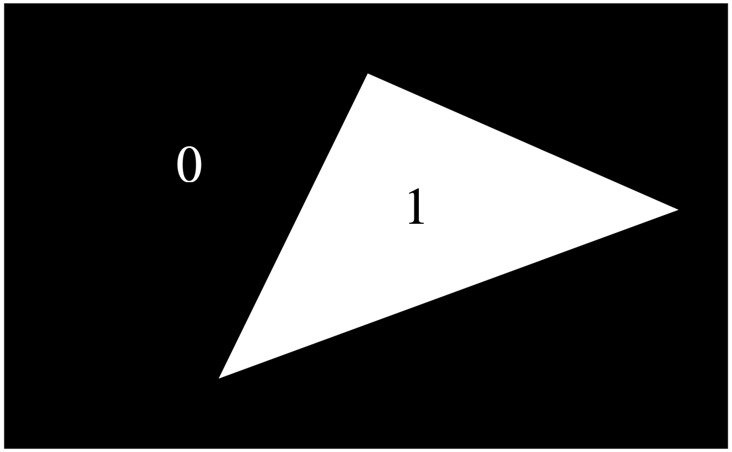


Frequency Domain

通过平均像素区域中的值消除走样

Solution:

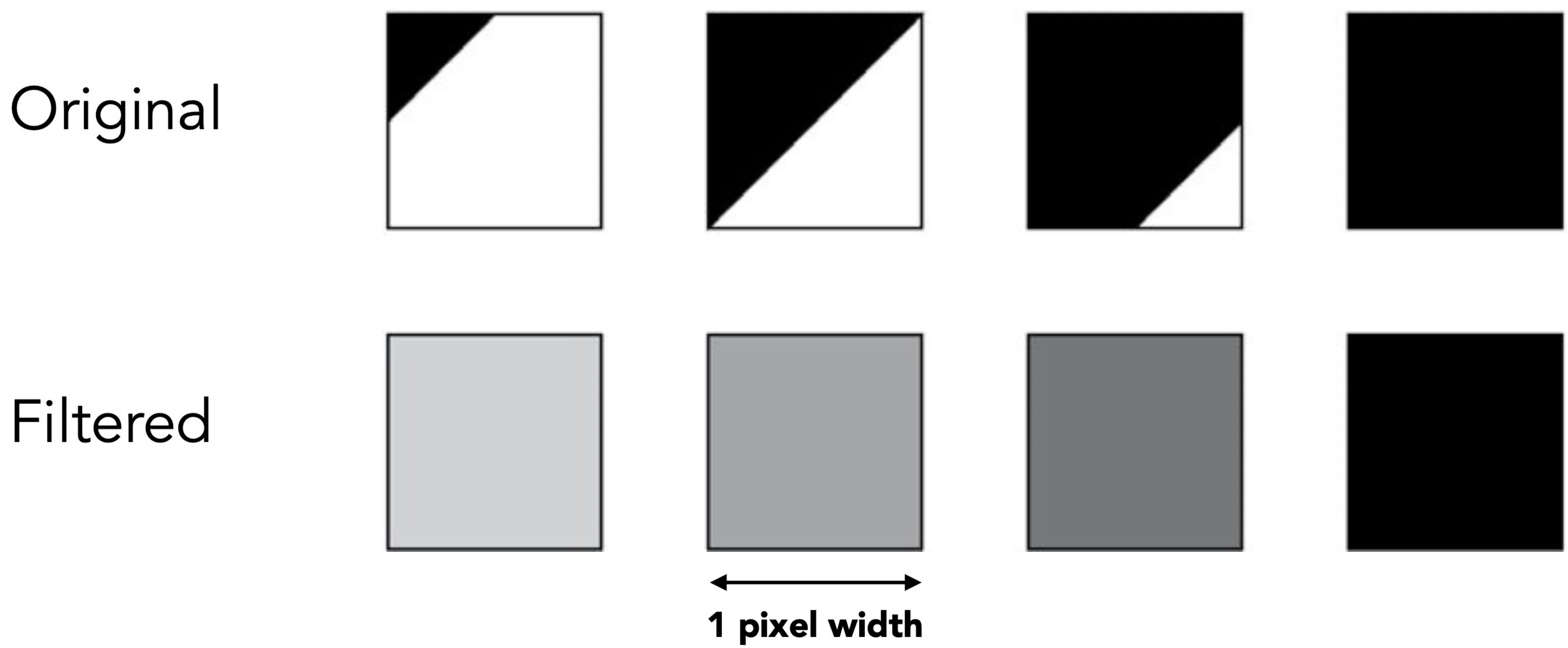
- **Convolve** f(x,y) by a 1-pixel box-blur
  - Recall: convolving = filtering = averaging
- **Then sample** at every pixel's center



f(x, y)

通过平均像素区域中的值消除走样

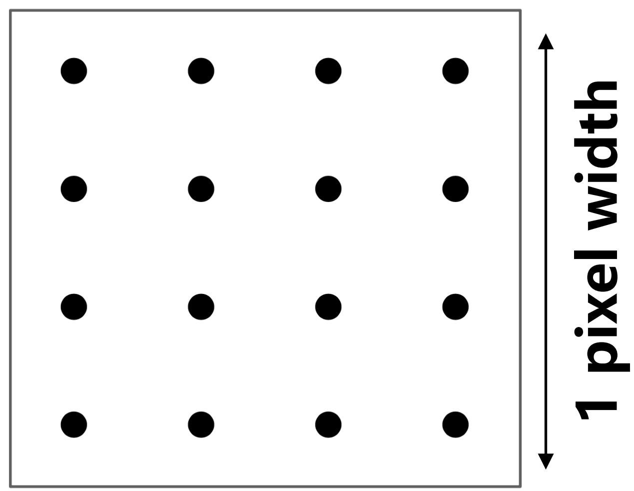
- 在光栅化一个三角形时，f(x, y) 的像素区域内的平均值等于三角形覆盖的像素的面积



Antialiasing by supersampling

超采样 Supersampling

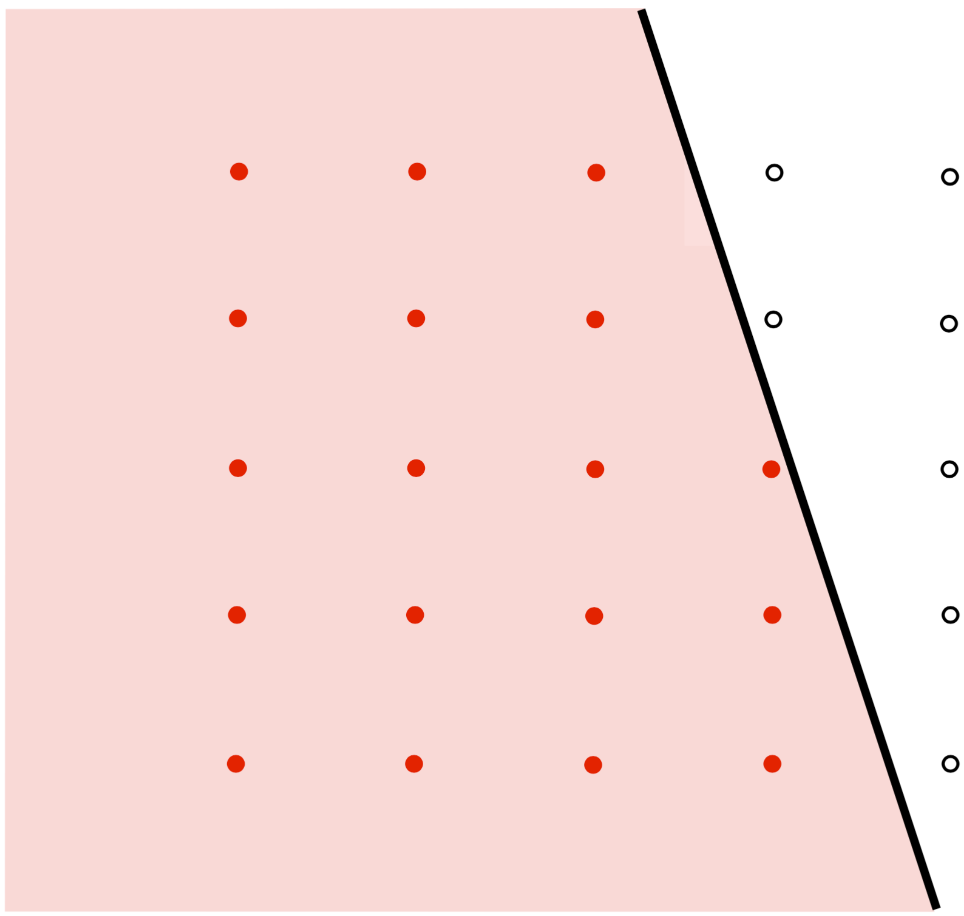
□通过对一个像素内的多个位置进行采样并取平均值来近似 1 像素盒滤波器的效果



4x4 supersampling

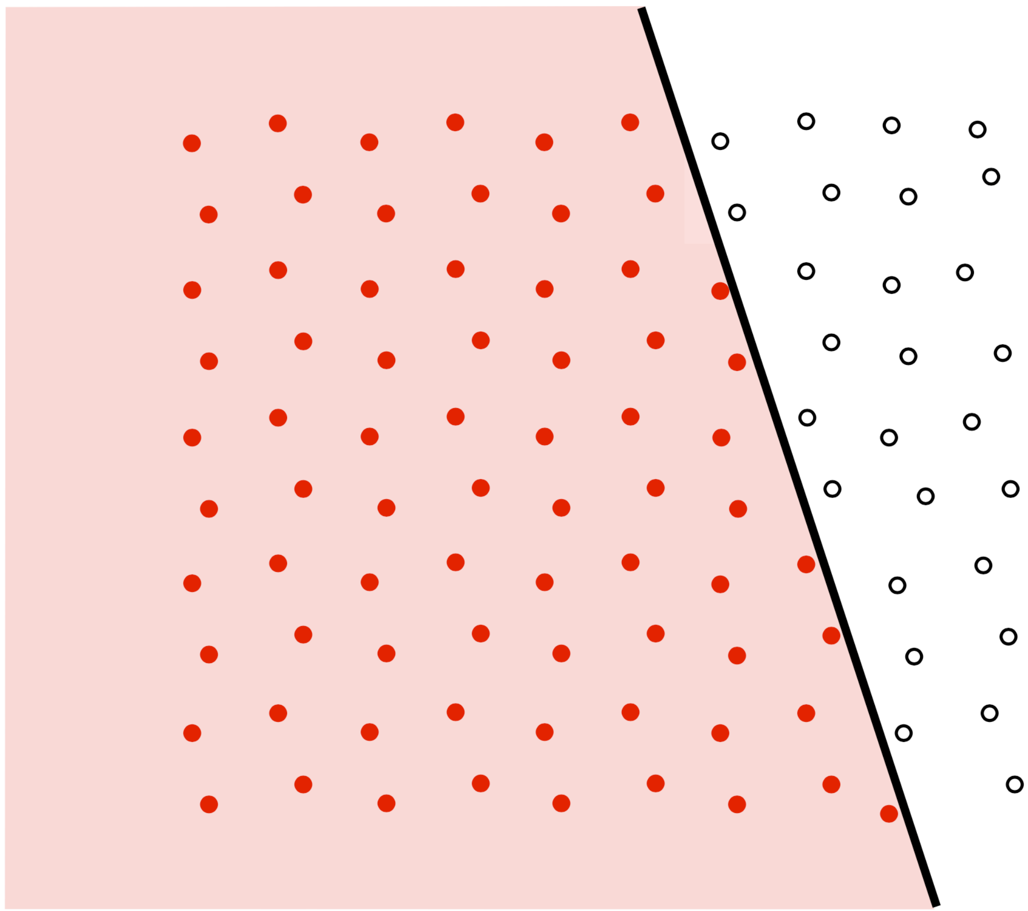
初始采样

- 每个像素采 1 个样本



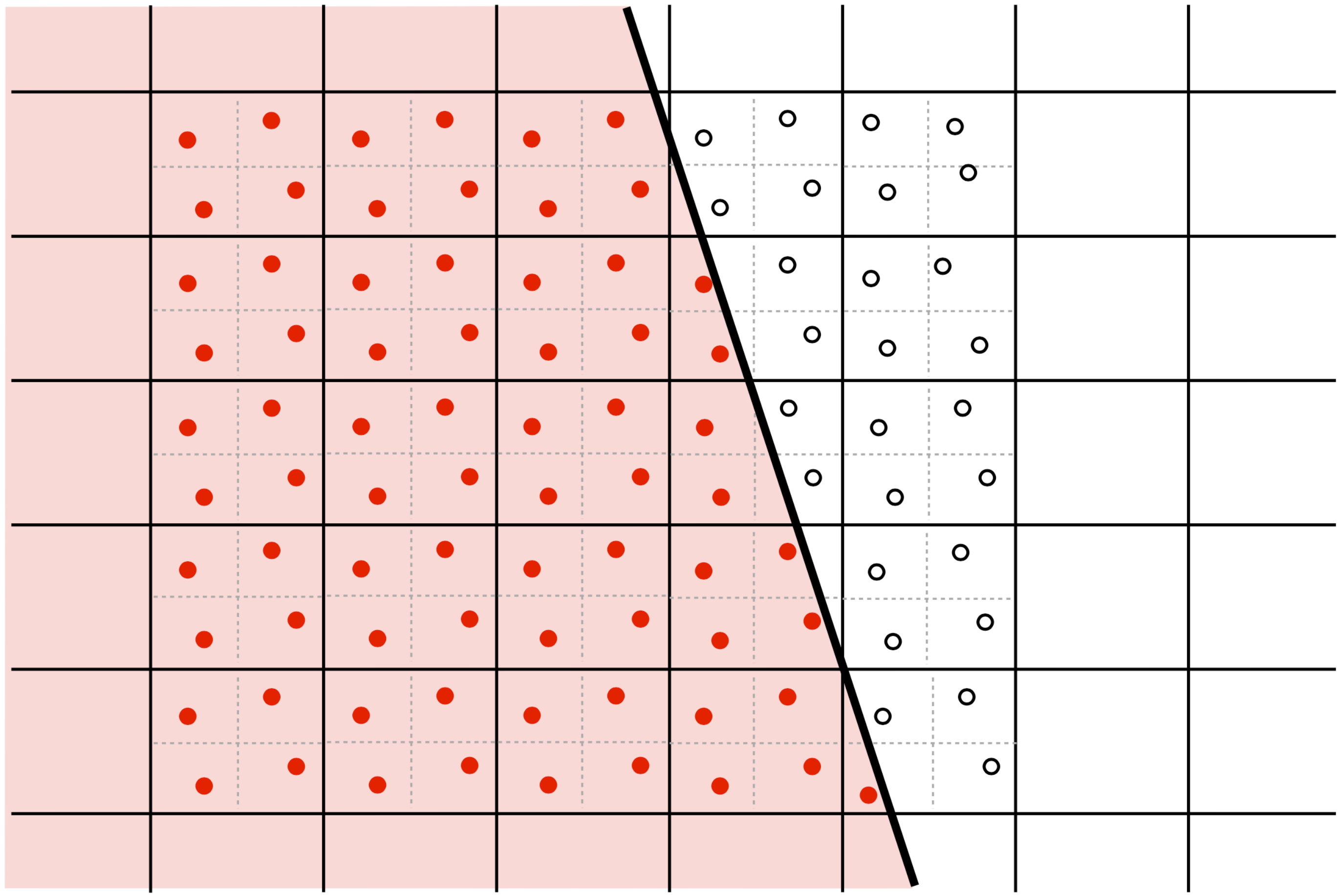
超采样

- 每个像素采 $n \times n$ 个样本 (4×4 的超采样)



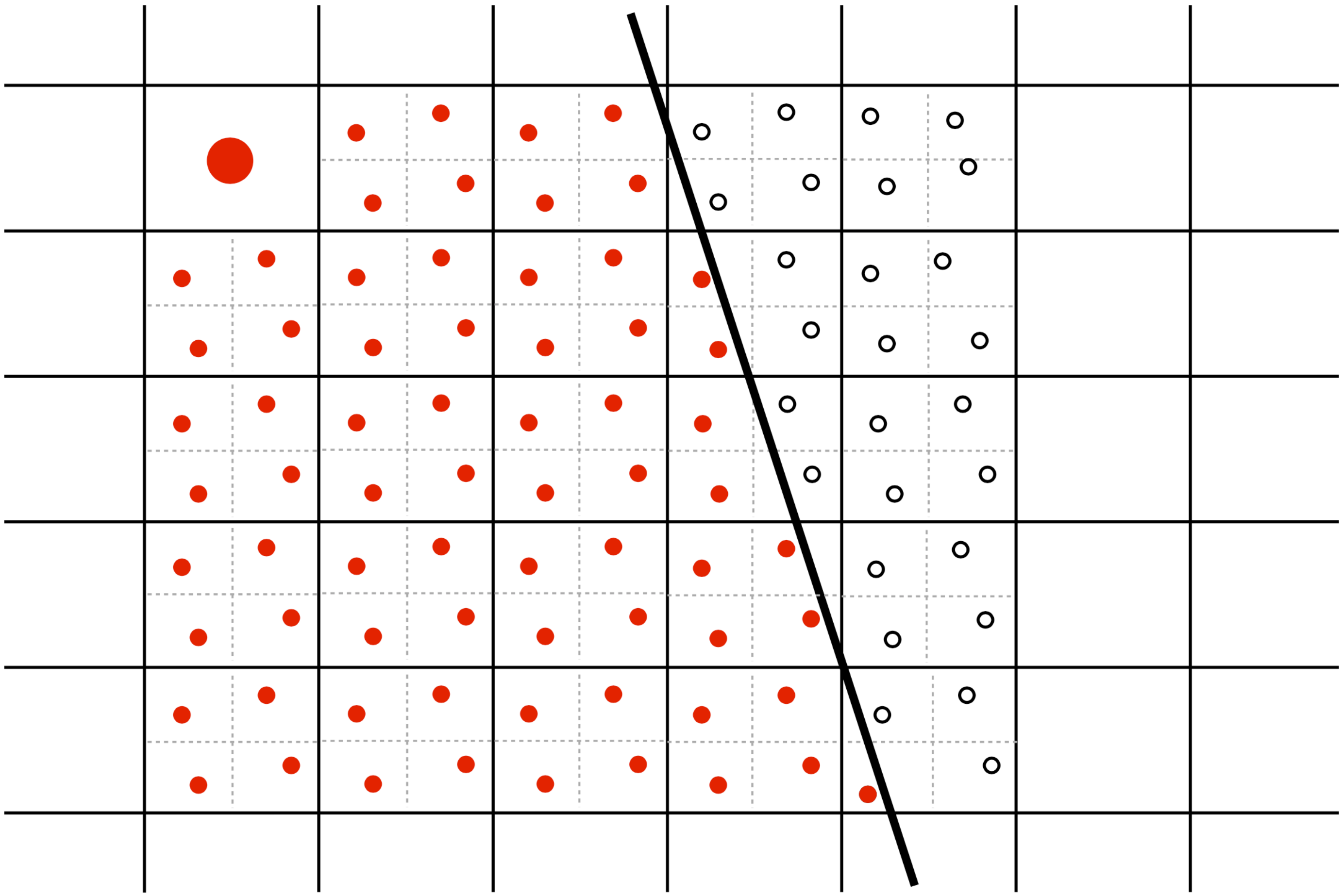
超采样

- 每个像素采 $n \times n$ 个样本 (4×4 的超采样)



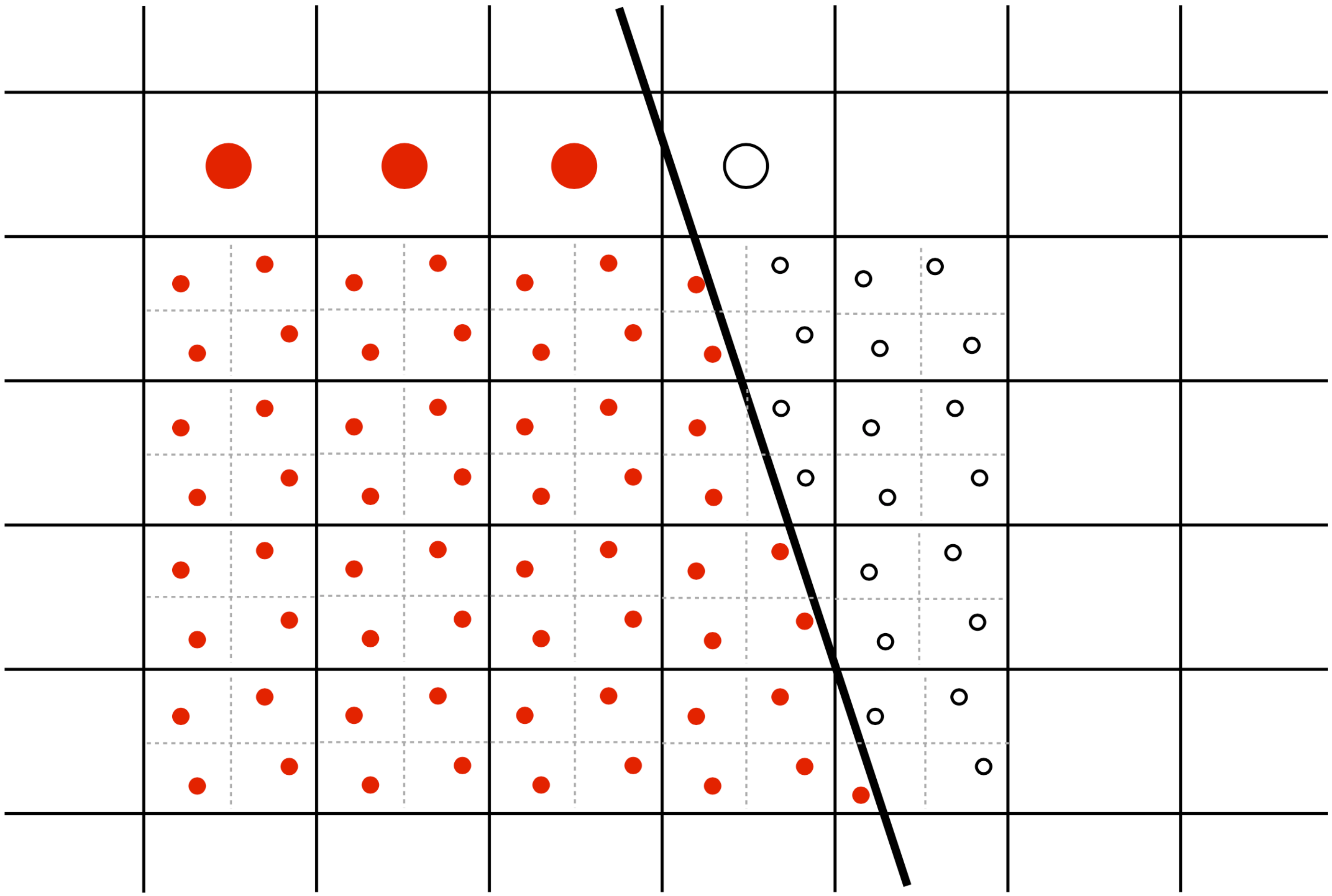
超采样

- 对每个像素里的 $n \times n$ 个样本取平均



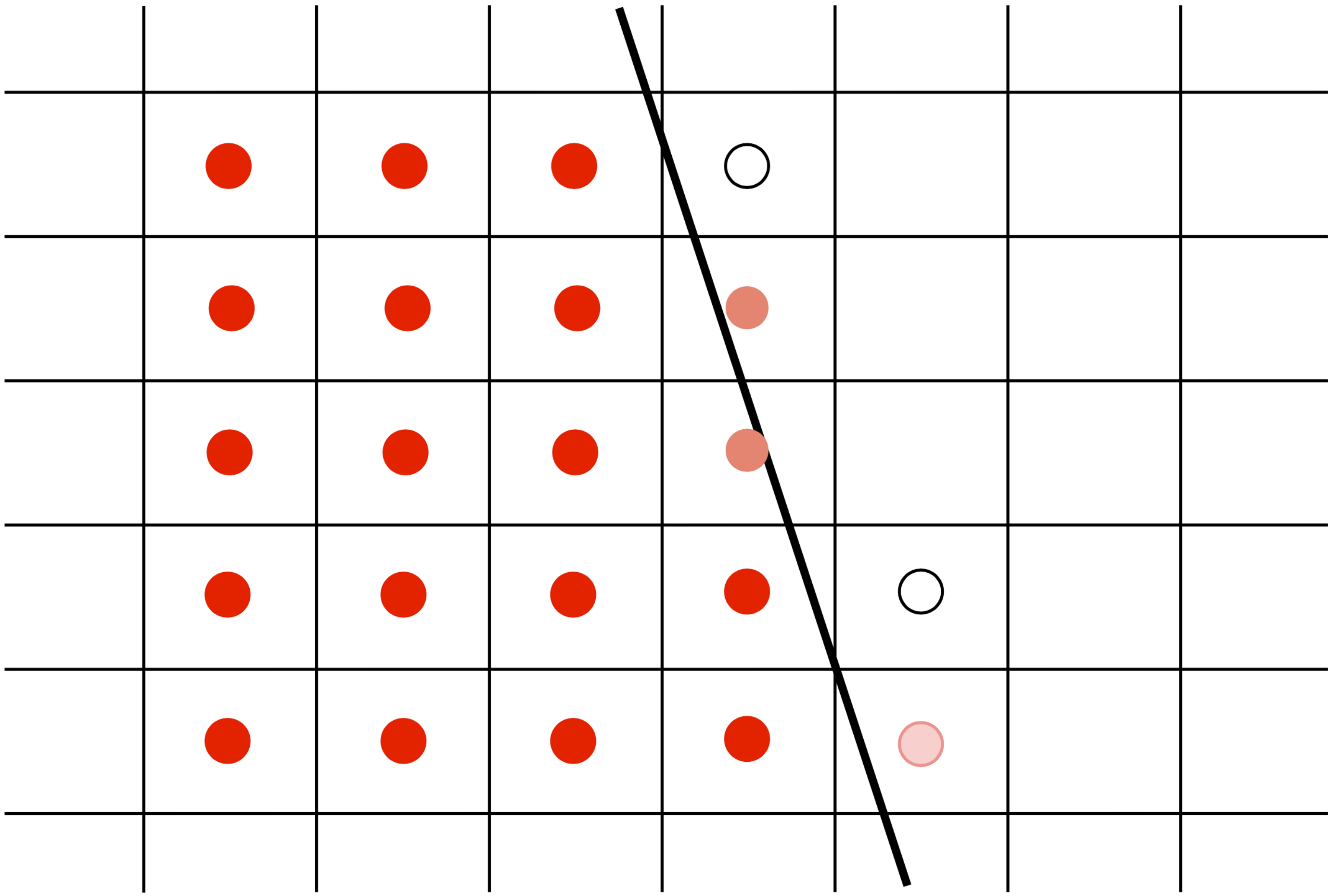
超采样

- 对每个像素里的 $n \times n$ 个样本取平均



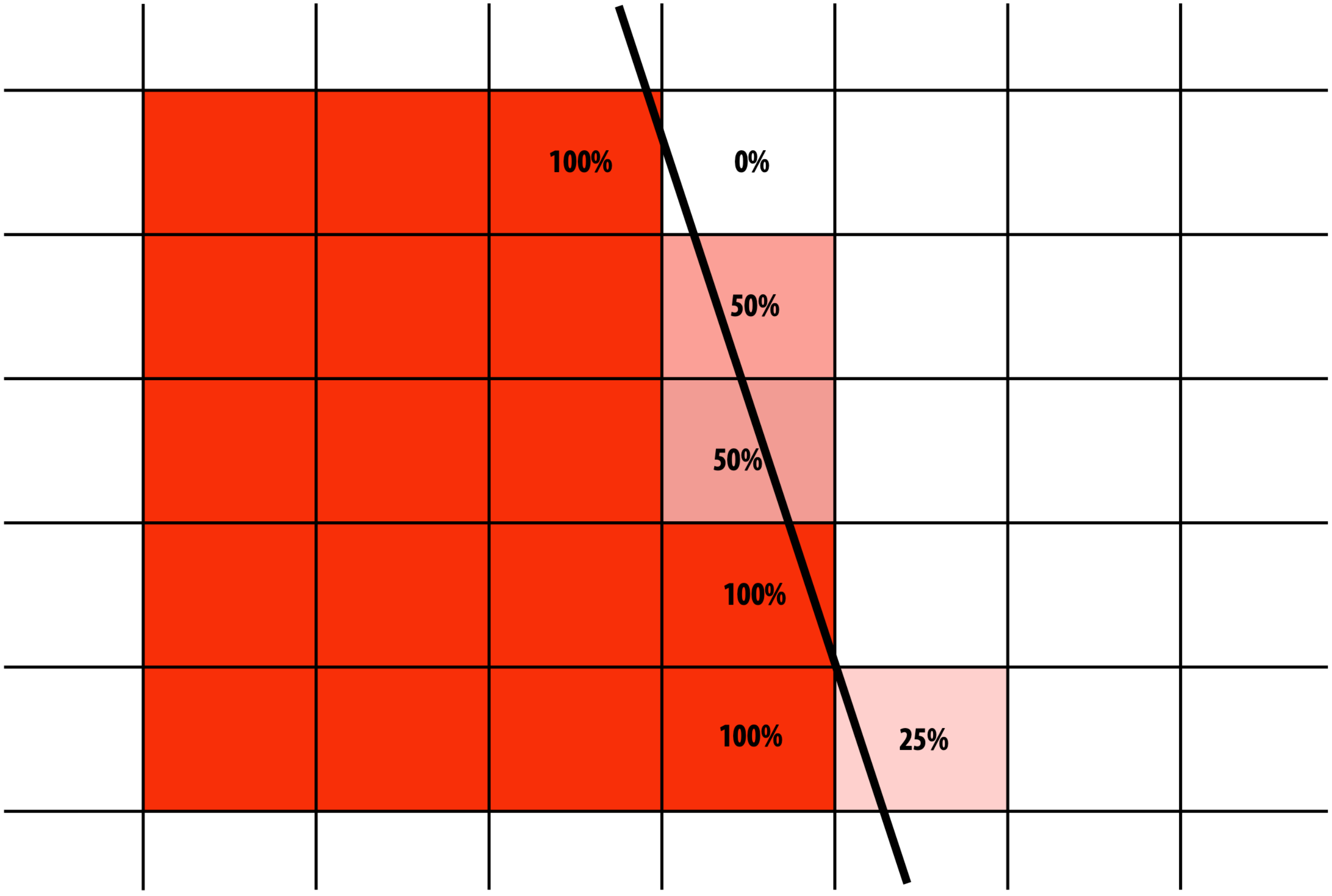
超采样

- 对每个像素里的 $n \times n$ 个样本取平均



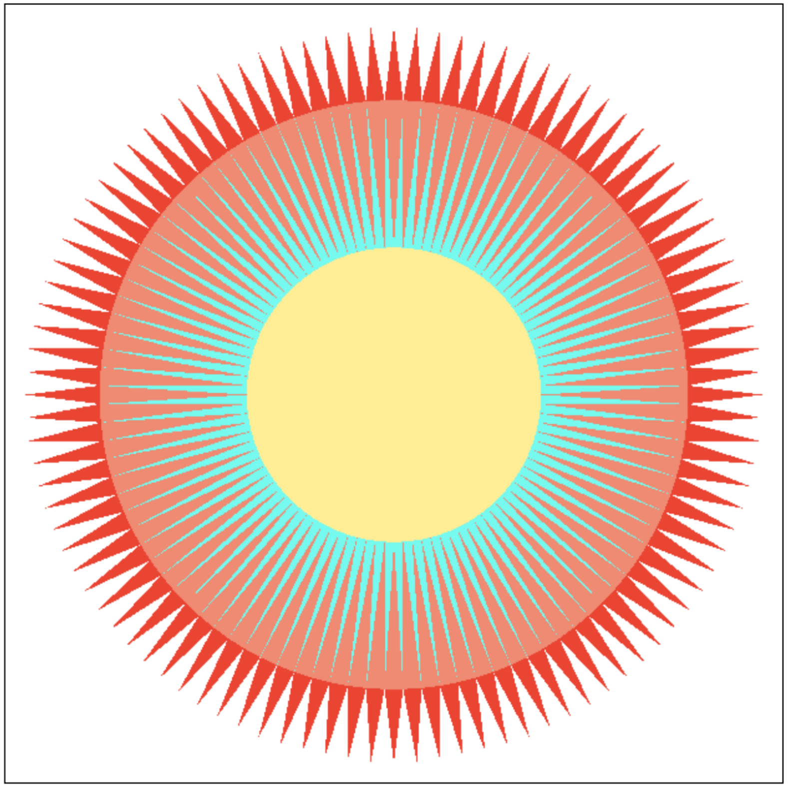
超采样的结果

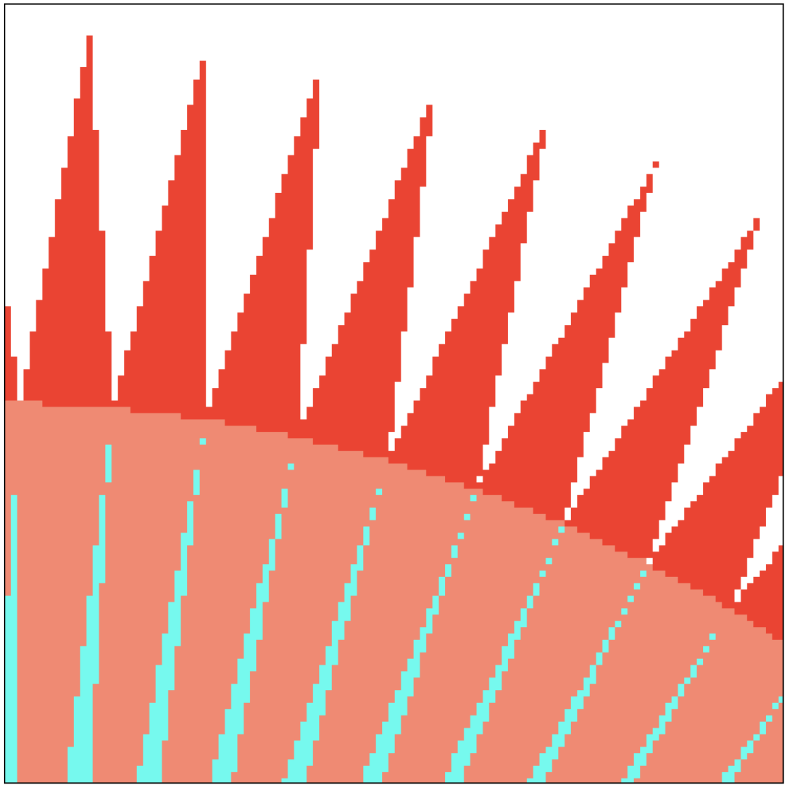
- 边进行了抗采样处理



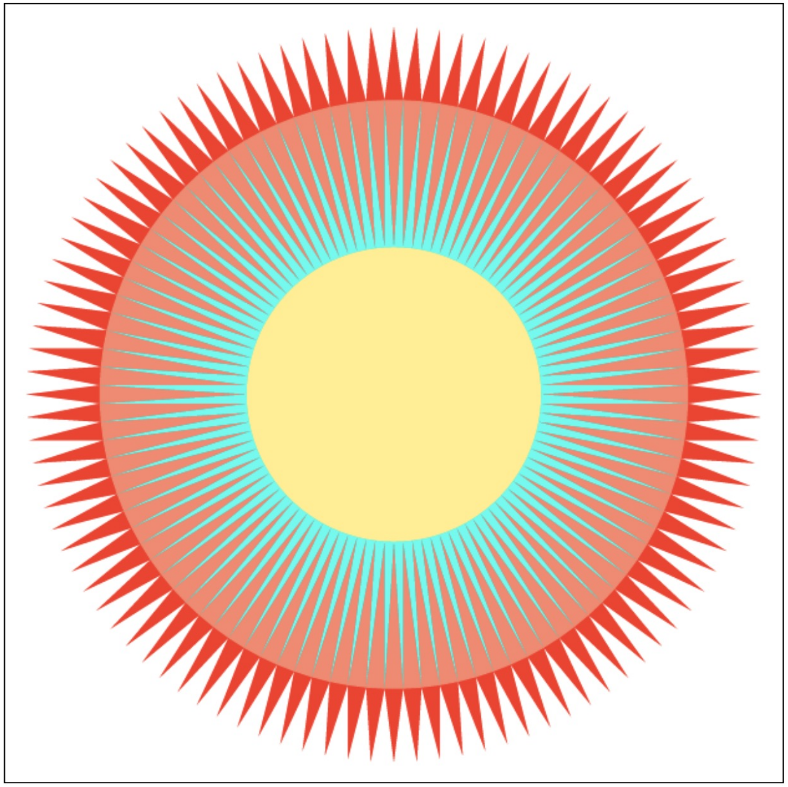
回顾一下目标 coverage signal

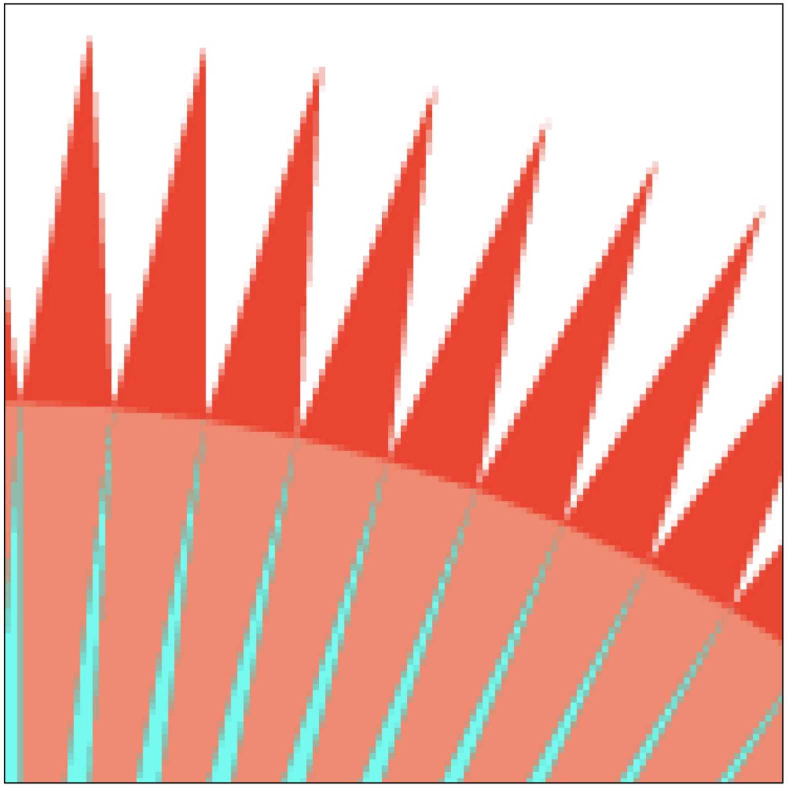
Point sampling



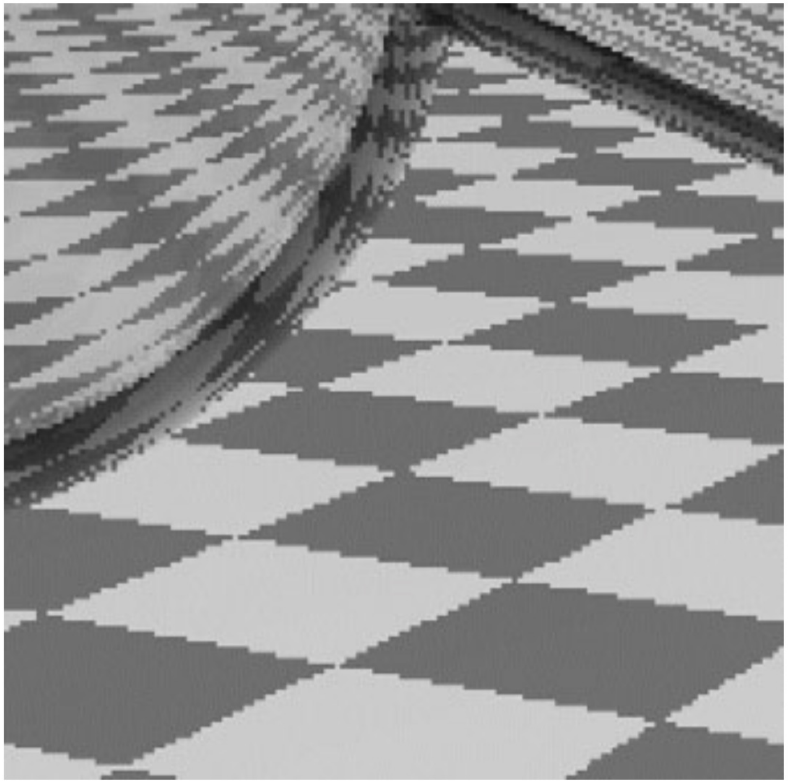


Antialiasing

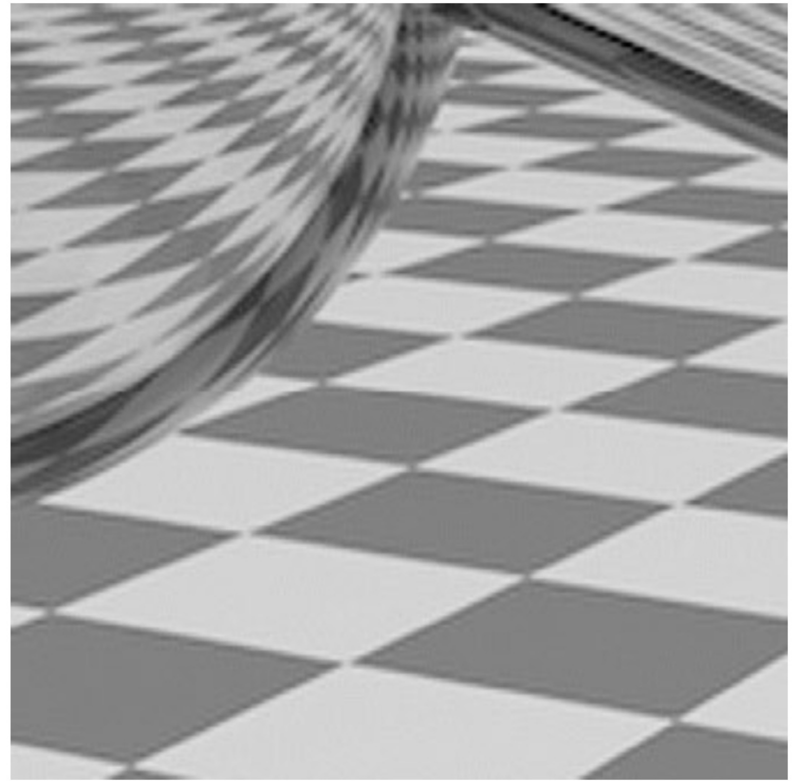




Point sampling vs. Antialiasing

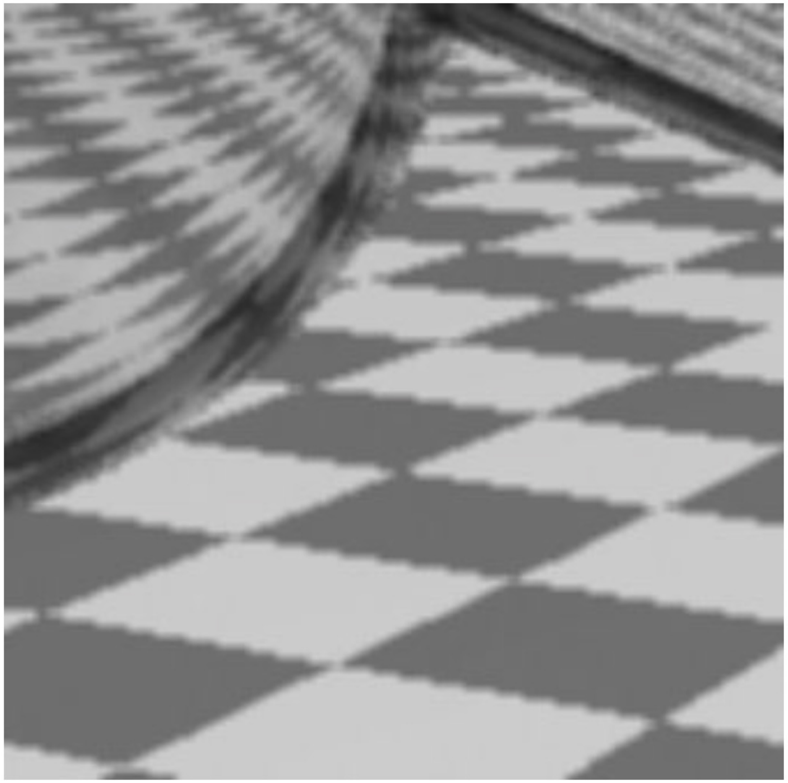


Jaggies

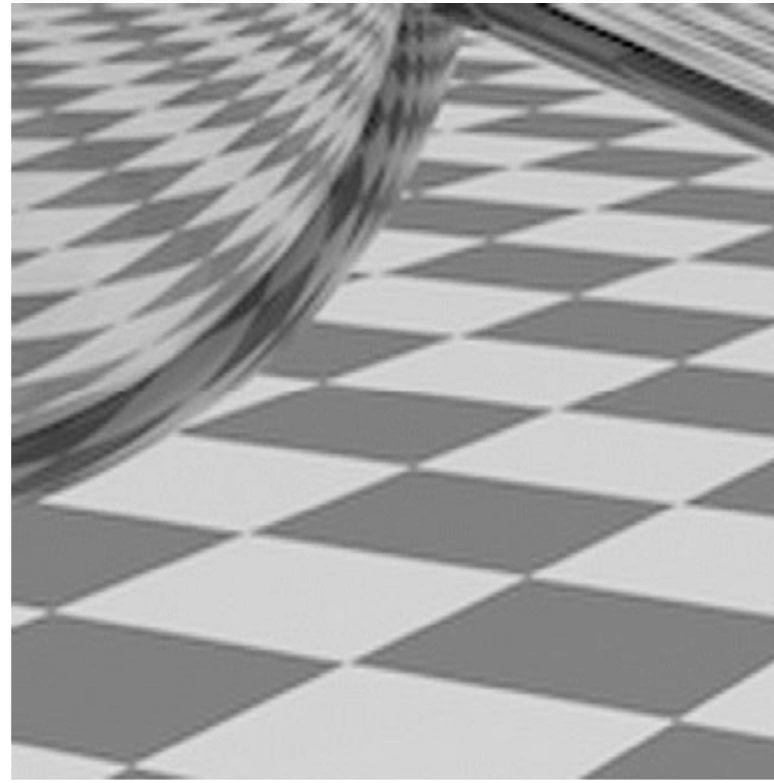


Pre-Filtered

Antialiasing vs Blurred Aliasing

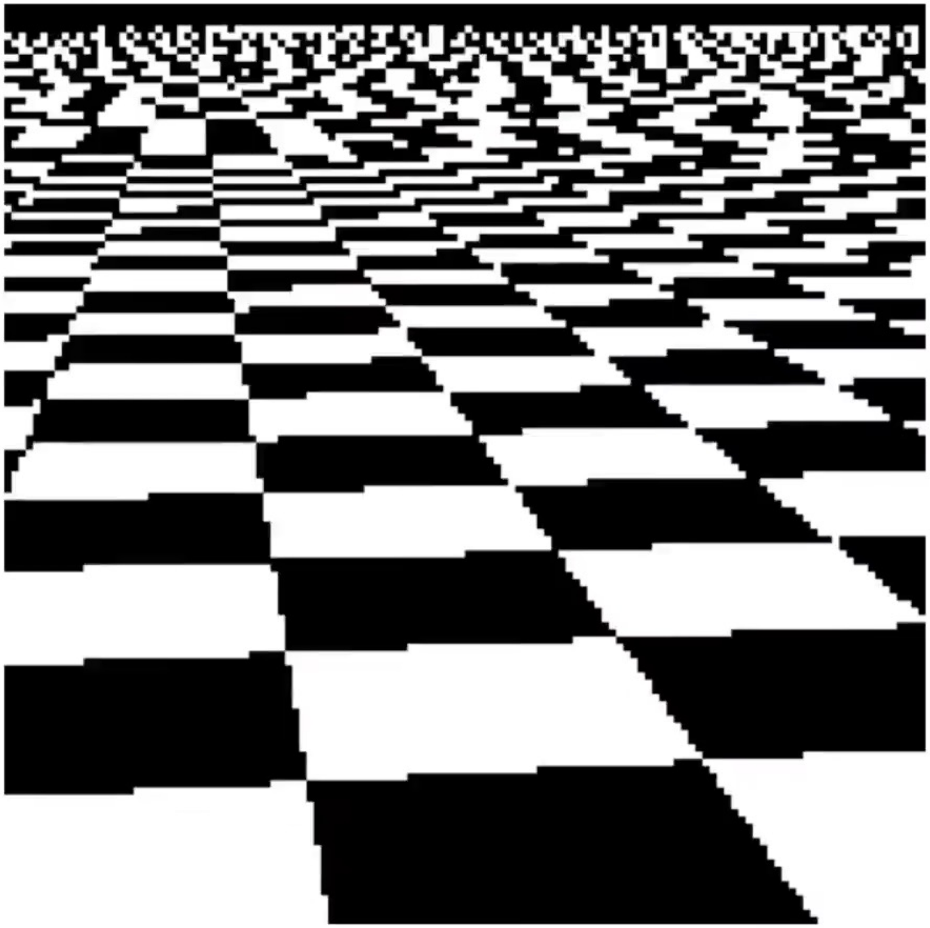


Blurred Jaggies
(Sample then filter)

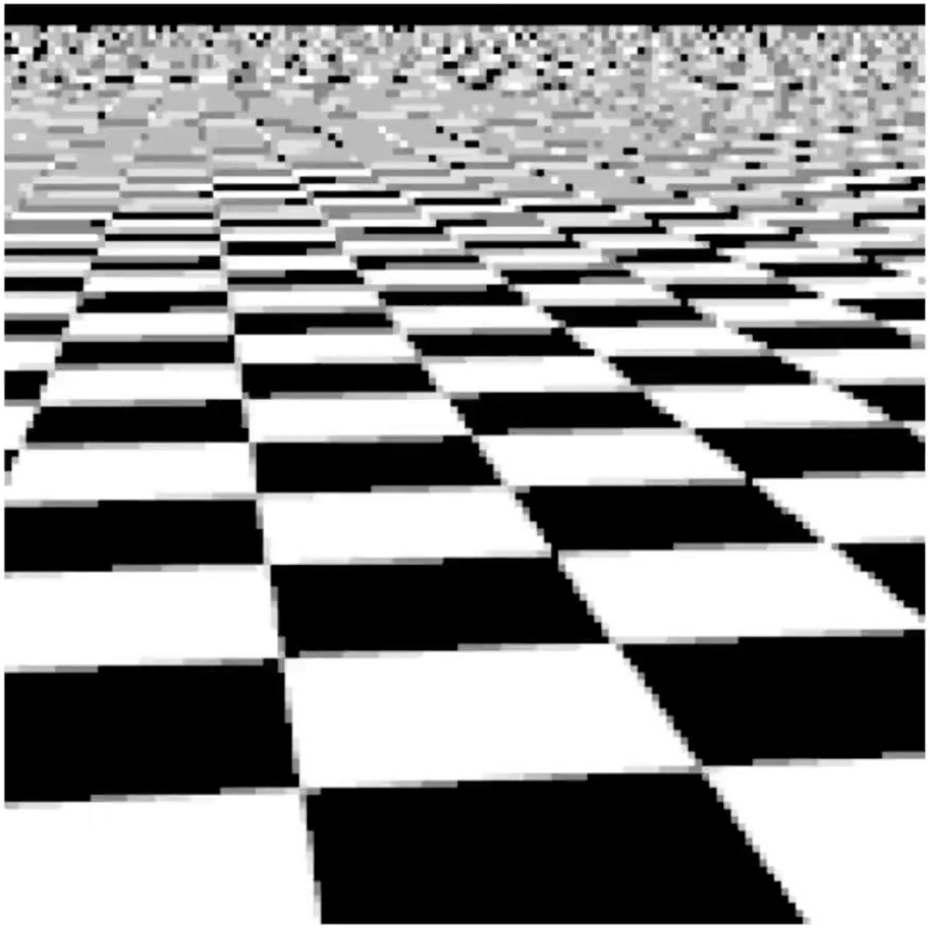


Pre-Filtered
(Filter then sample)

单采样和超采样

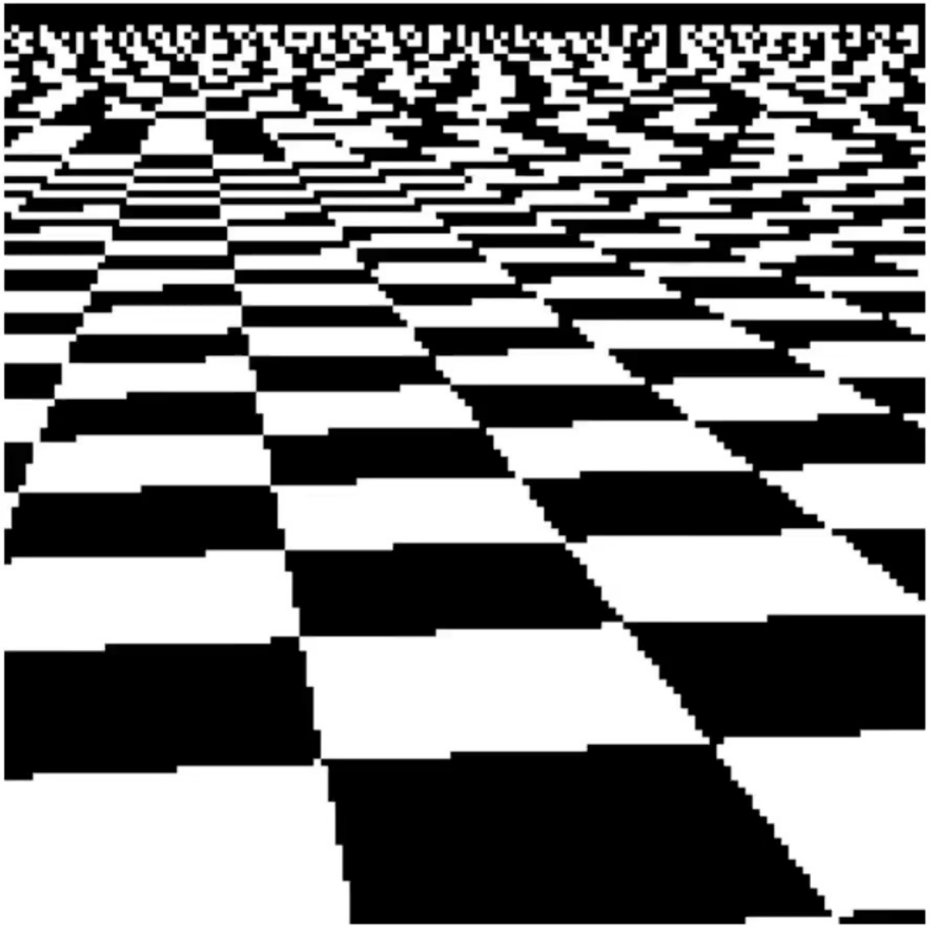


single sampling

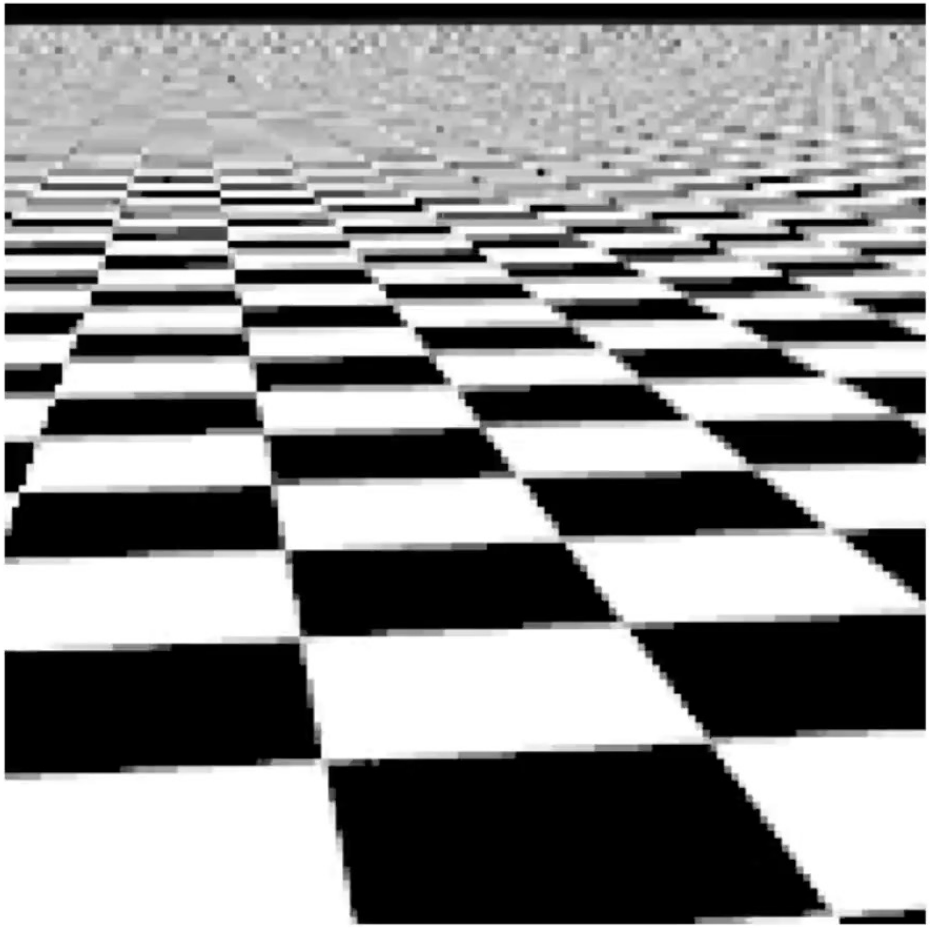


2x2 supersampling

单采样和超采样

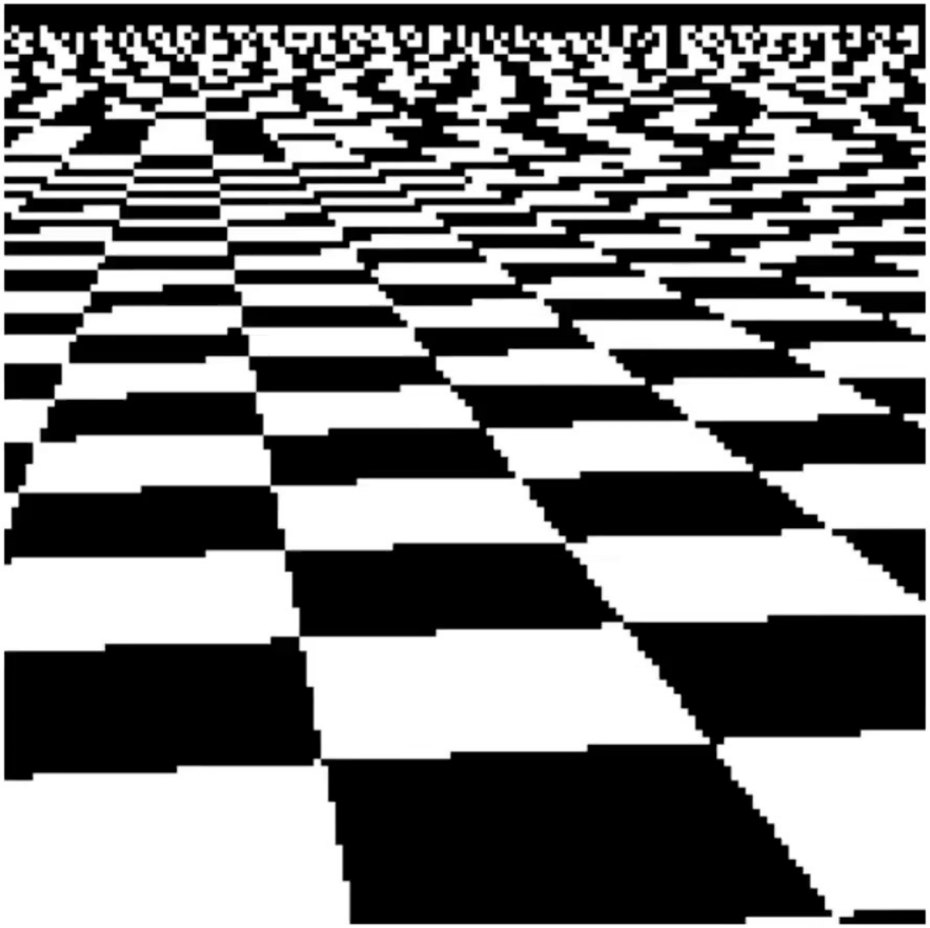


single sampling

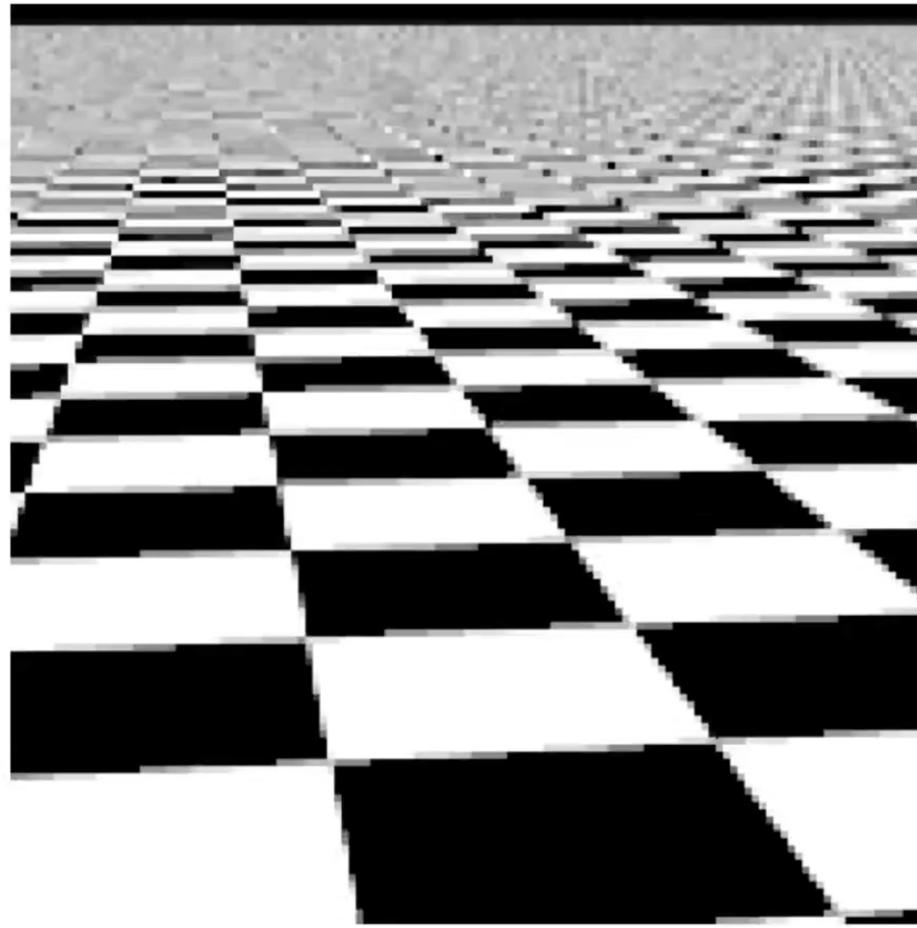


4x4 supersampling

单采样和超采样



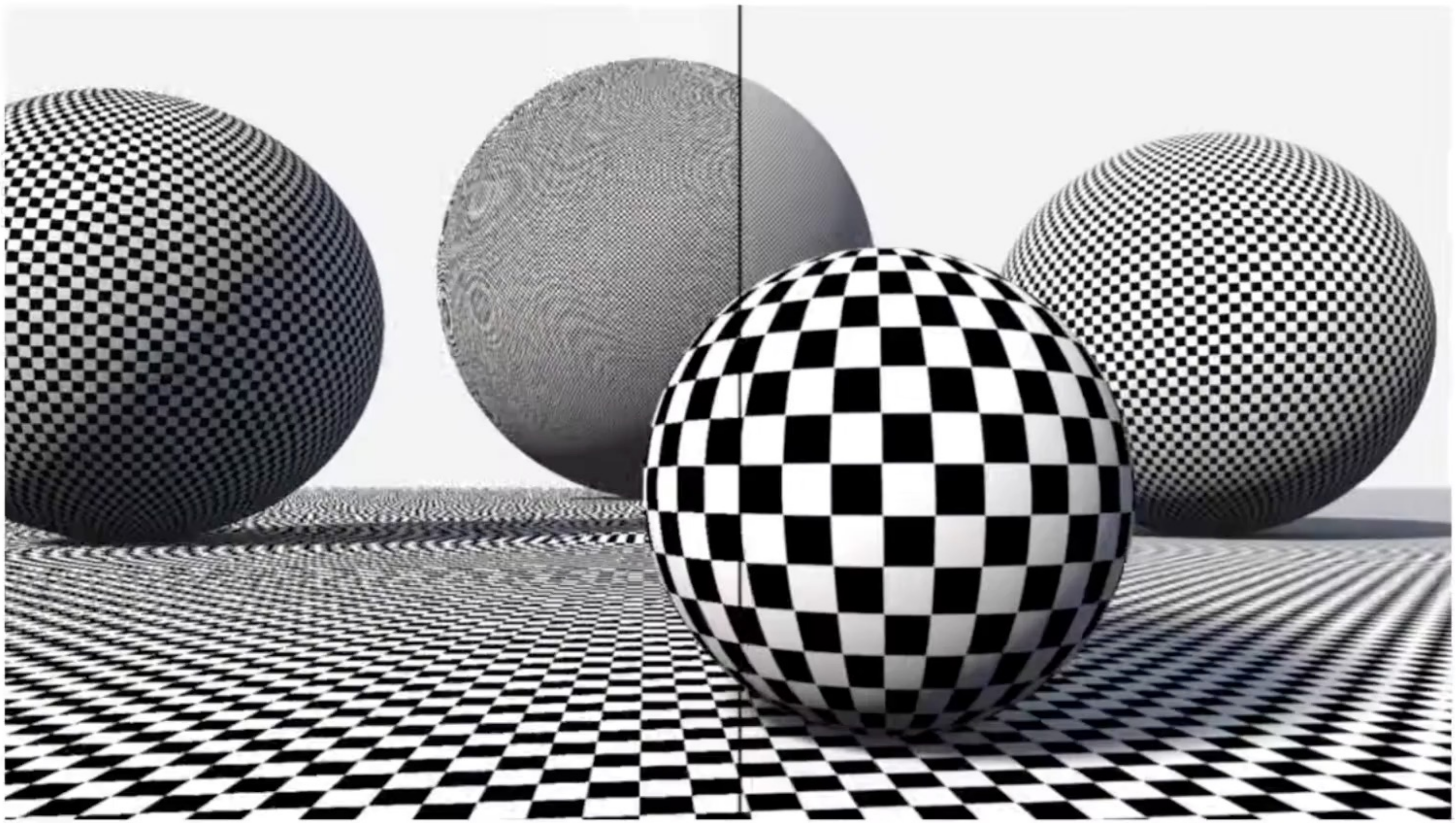
single sampling



32x32 supersampling

Checkerboard - Exact Solution

□在非常特殊的情况下，我们可以计算出确切的覆盖范围



Such cases are extremely rare – want
solutions that will work in the general case!

See Inigo Quilez, “Filtering the Checkerboard Pattern” & Apodaca et al, “Advanced Renderman” (p. 273)

谢谢

陈壮彬
软件工程学院
chenzhb36@mail.sysu.edu.cn