计算机视觉

边缘检测

中国传媒大学

COMMUNICATION UNIVERSITY OF CHINA

本节主题：

模板匹配

本节主题：

模板匹配

方向可调滤波器

本节主题：

模板匹配

方向可调滤波器

接缝裁剪

本节主题：

模板匹配
方向可调滤波器
接缝裁剪
边缘检测

回顾

互相关 $H = G \otimes F$

$$H[x,y] = \sum_{u=-K}^{K} \sum_{v=-K}^{K} G[u,v]F[x+u, y+v]$$

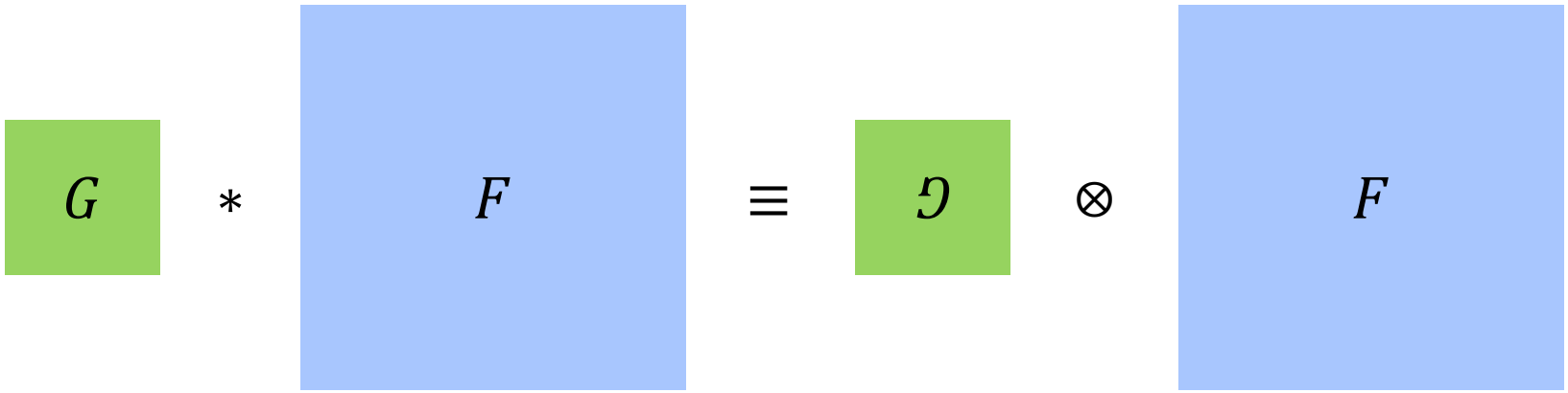
卷积 $H = G * F$

$$H[x,y] = \sum_{u=-K}^{K} \sum_{v=-K}^{K} G[u,v]F[x-u, y-v]$$

回顾

互相关 $H = G \otimes F$

$$H[x,y] = \sum_{u=-K}^{K} \sum_{v=-K}^{K} G[u,v]F[x+u,y+v]$$



卷积 $H = G * F$

$$H[x,y] = \sum_{u=-K}^{K} \sum_{v=-K}^{K} G[u,v]F[x-u,y-v]$$

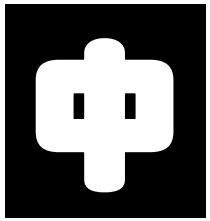
模板匹配

输入图像

中国传媒大学

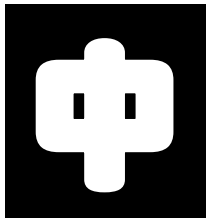
如何找到“中”字？

输入图像



模板

输入图像



模板

如何在图像中找到与模板匹配的部分？

输入图像

回顾：线性代数

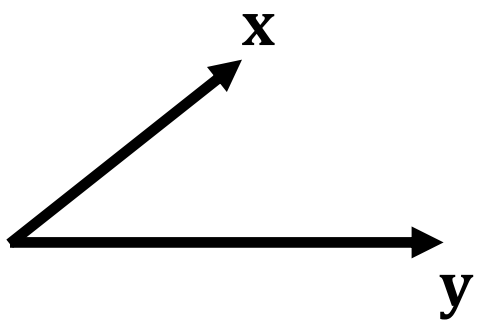
定义：

点积，也称作内积或标量积，两个向量之间的点积被定义为

定义：

点积，也称作内积或标量积，两个向量之间的点积被定义为

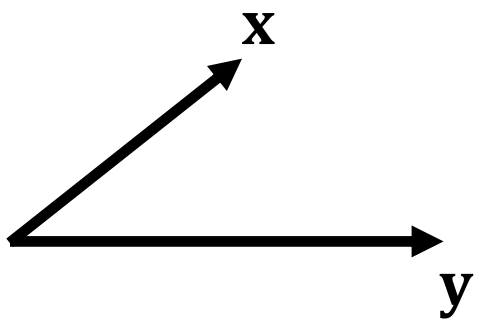
$$\mathbf{x} \cdot \mathbf{y} =$$



定义：

点积，也称作内积或标量积，两个向量之间的点积被定义为

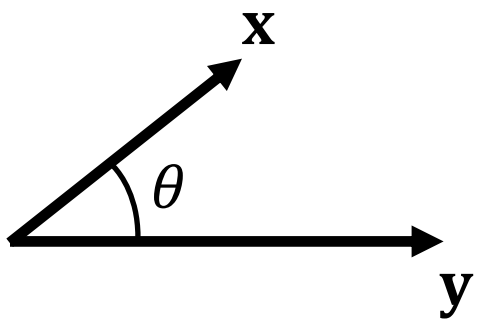
$$\mathbf{x} \cdot \mathbf{y} = \sum_{i=1}^{N} x_i y_i$$



定义：

点积，也称作内积或标量积，两个向量之间的点积被定义为

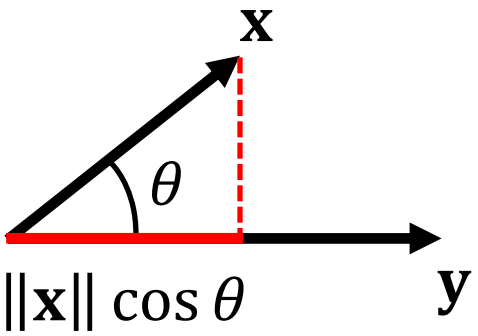
$$\mathbf{x} \cdot \mathbf{y} = \sum_{i=1}^{N} x_i y_i = \|\mathbf{x}\| \|\mathbf{y}\| \cos\theta$$



定义：

点积，也称作内积或标量积，两个向量之间的点积被定义为

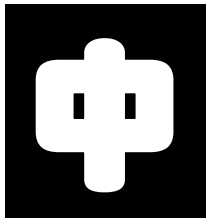
$$\mathbf{x} \cdot \mathbf{y} = \sum_{i=1}^{N} x_i y_i = \|\mathbf{x}\| \|\mathbf{y}\| \cos\theta$$



回顾：线性代数

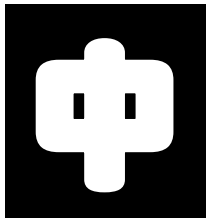
模板匹配

直觉启发



模板

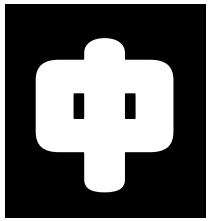
输入图像



⊗

模板

输入图像

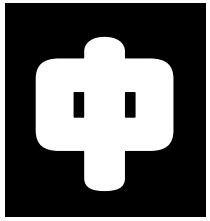


模板

⊗

子图像

输入图像



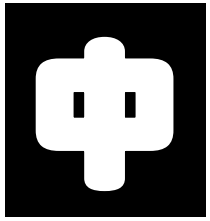
模板

$\otimes$

子图像

输入图像

相关性评分 $= \mathbf{t} \cdot \mathbf{s}$



模板

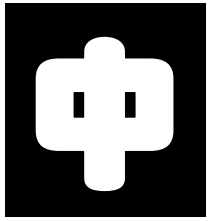
⊗

子图像

输入图像

相关性评分 $= t \cdot s$

看起来眼熟吗？



模板

⊗

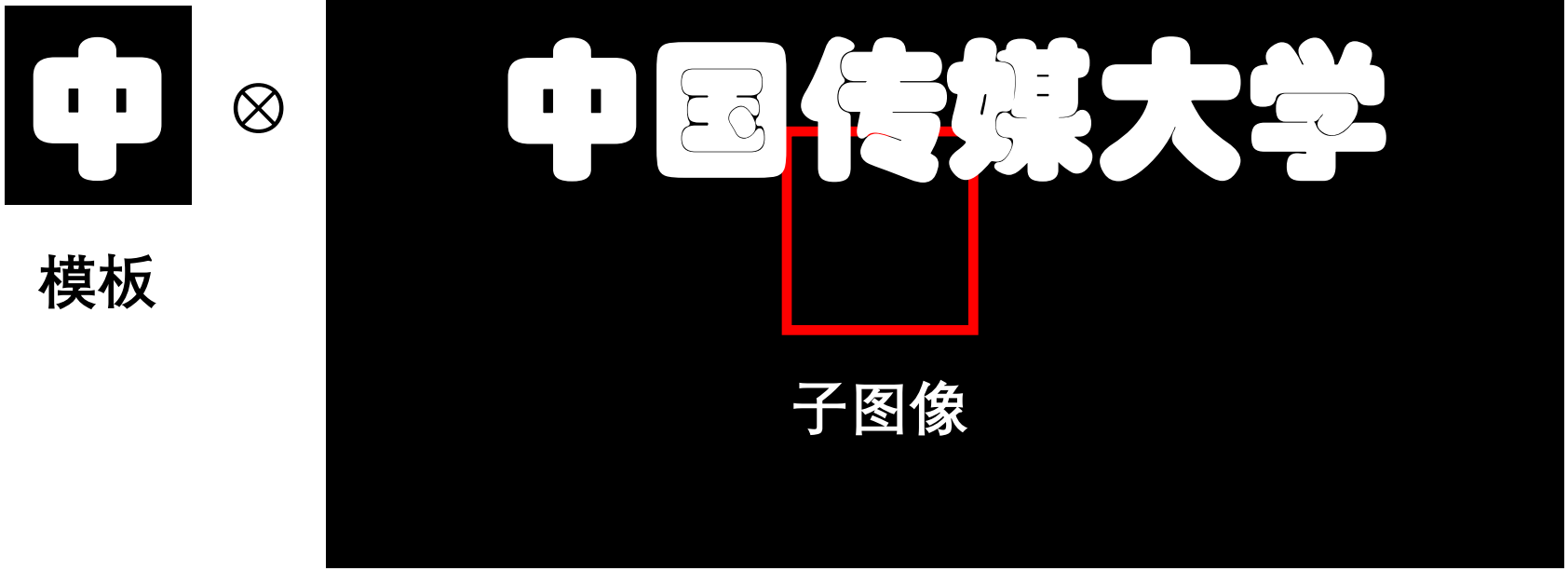
子图像

输入图像

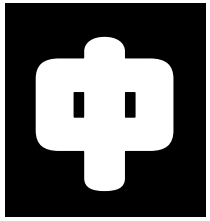
相关性评分 $= t \cdot s$

看起来眼熟吗？

点积



$$\text{相关性评分} = \mathbf{t} \cdot \mathbf{s} = \|\mathbf{t}\| \|\mathbf{s}\| \cos\theta$$

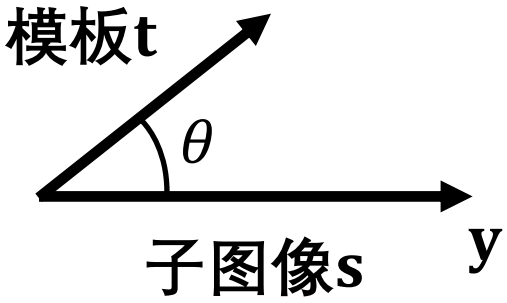


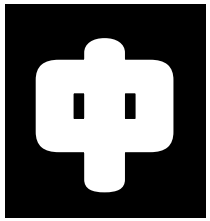
模板

⊗

输入图像

$$\text{相关性评分} = \mathbf{t} \cdot \mathbf{s} = \|\mathbf{t}\| \|\mathbf{s}\| \cos\theta$$





⊗

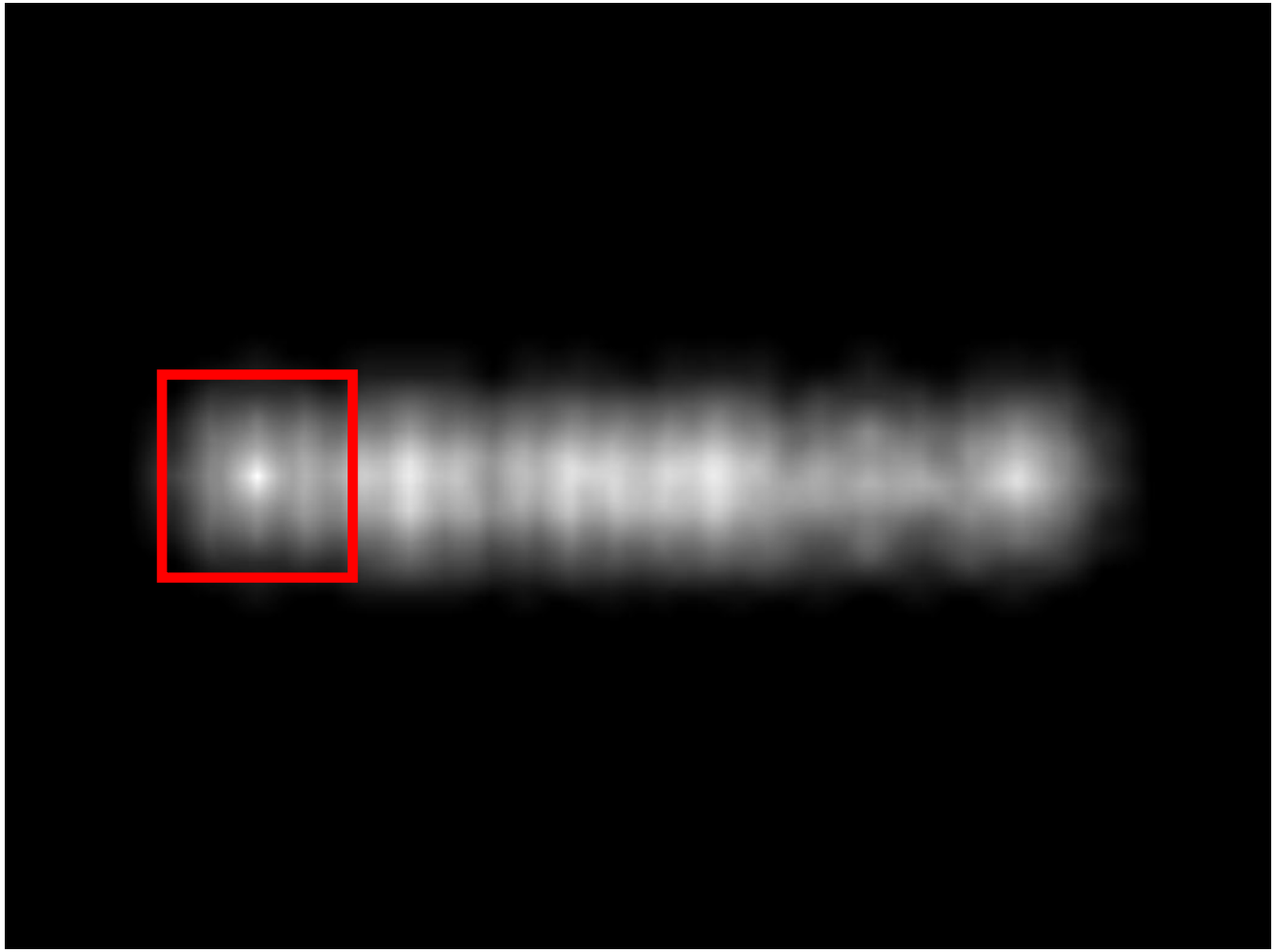
模板

输入图像

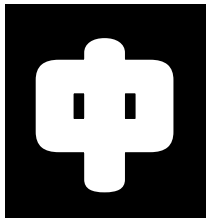
=

输入图像

=



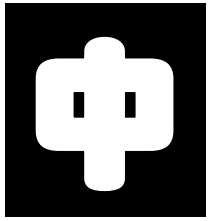
输入图像



模板

⊗

输入图像



模板

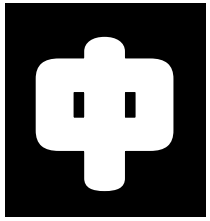
⊗

子图像

输入图像

$$\text{相关性评分} = \mathbf{t} \cdot \mathbf{s} = \|\mathbf{t}\|\|\mathbf{s}\| \cos\theta$$

相关性评分对乘性亮度变化敏感



模板

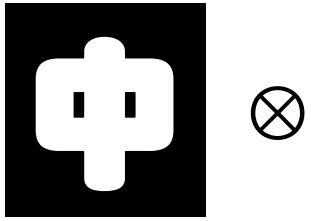
⊗

子图像

输入图像

$$\text{相关性评分} = \mathbf{t} \cdot \mathbf{s} = \|\mathbf{t}\|\|\mathbf{s}\| \cos\theta$$

如何让相关性评分不受亮度变化的影响呢？



模板

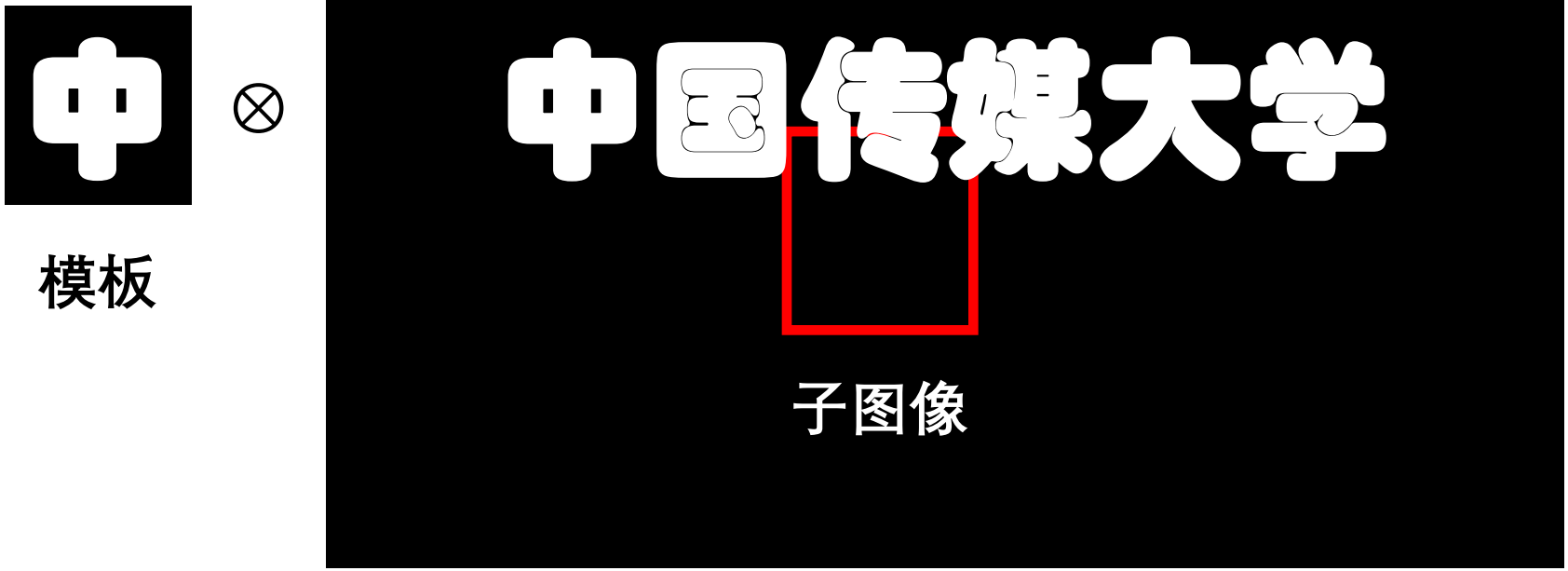
⊗

子图像

输入图像

$$\text{相关性评分} = \mathbf{t} \cdot \mathbf{s} = \|\mathbf{t}\|\|\mathbf{s}\| \cos\theta$$

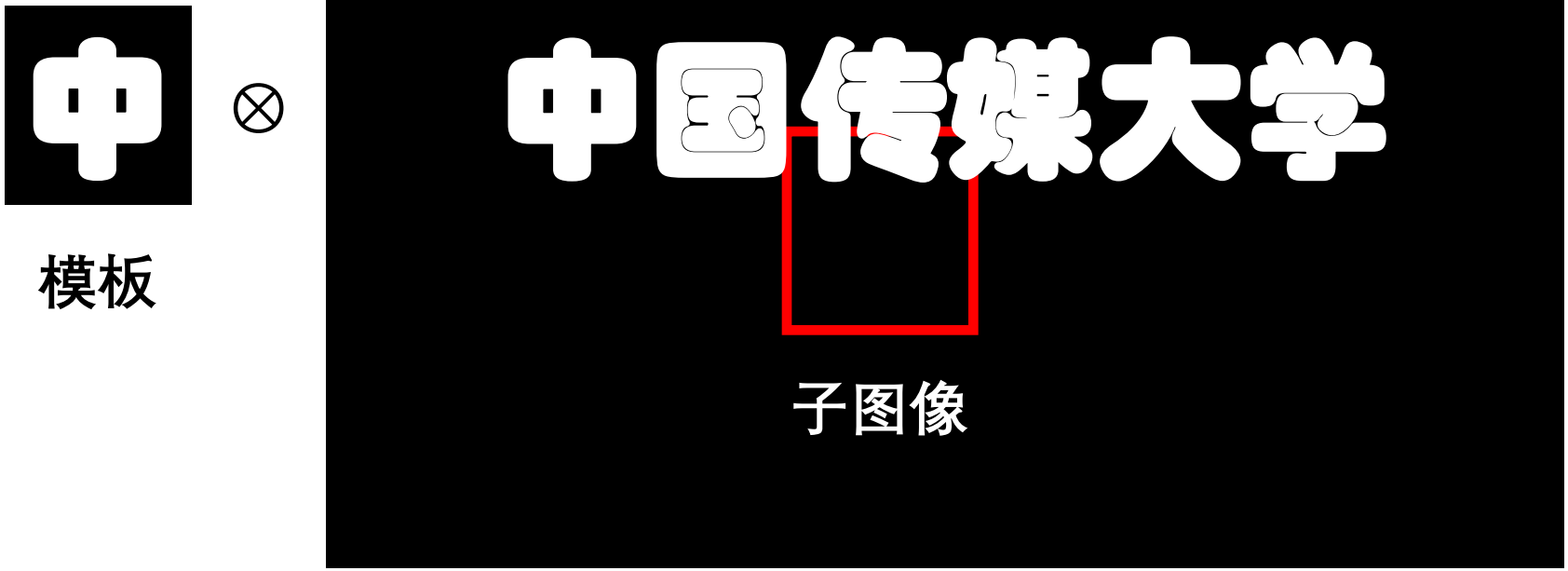
$$\text{归一化评分} = \frac{\mathbf{t}}{\|\mathbf{t}\|} \cdot \frac{\mathbf{s}}{\|\mathbf{s}\|} = \cos\theta$$



$$\text{相关性评分} = \mathbf{t} \cdot \mathbf{s} = \|\mathbf{t}\|\|\mathbf{s}\| \cos\theta$$

$$\text{归一化评分} = \frac{\mathbf{t}}{\|\mathbf{t}\|} \cdot \frac{\mathbf{s}}{\|\mathbf{s}\|} = \cos\theta$$

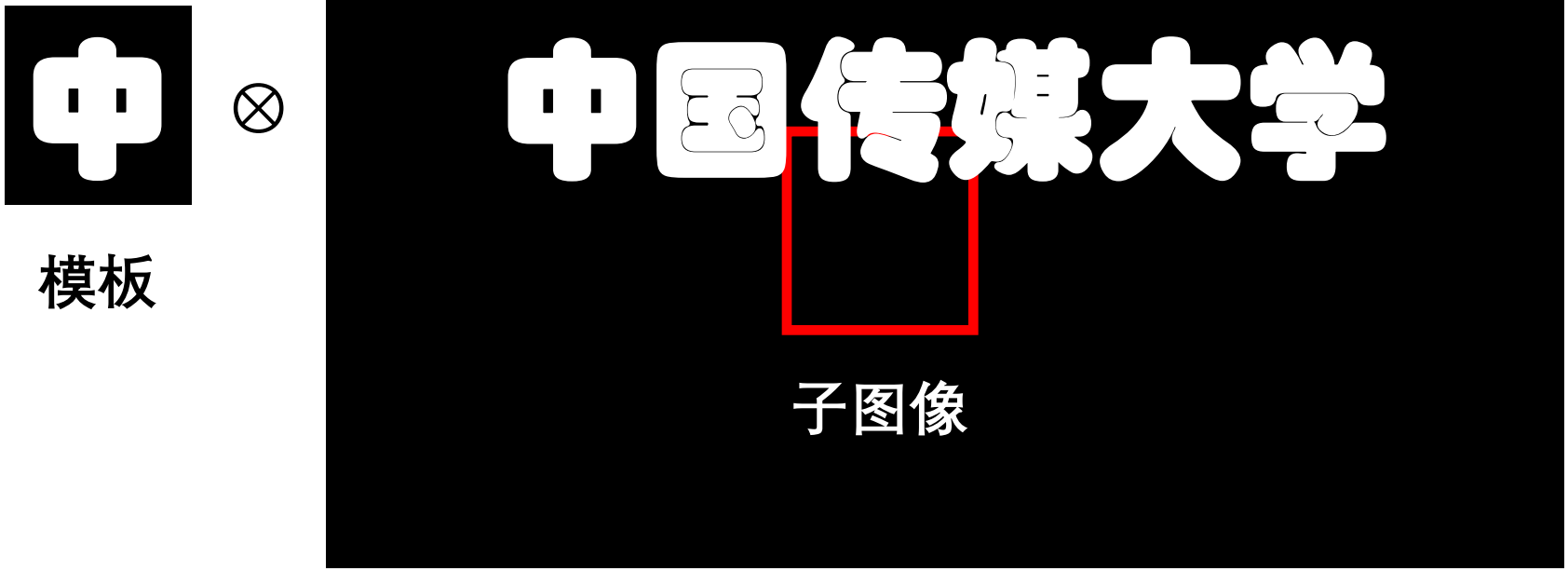
如何让相关性评分既不受乘性也不受加性亮度变化的影响呢?



$$\text{相关性评分} = \mathbf{t} \cdot \mathbf{s} = \|\mathbf{t}\|\|\mathbf{s}\| \cos\theta$$

$$\text{归一化评分} = \frac{\mathbf{t}}{\|\mathbf{t}\|} \cdot \frac{\mathbf{s}}{\|\mathbf{s}\|} = \cos\theta$$

$$\text{归一化互相关} = \frac{\mathbf{t} - \bar{\mathbf{t}}}{\|\mathbf{t} - \bar{\mathbf{t}}\|} \cdot \frac{\mathbf{s} - \bar{\mathbf{s}}}{\|\mathbf{s} - \bar{\mathbf{s}}\|}$$

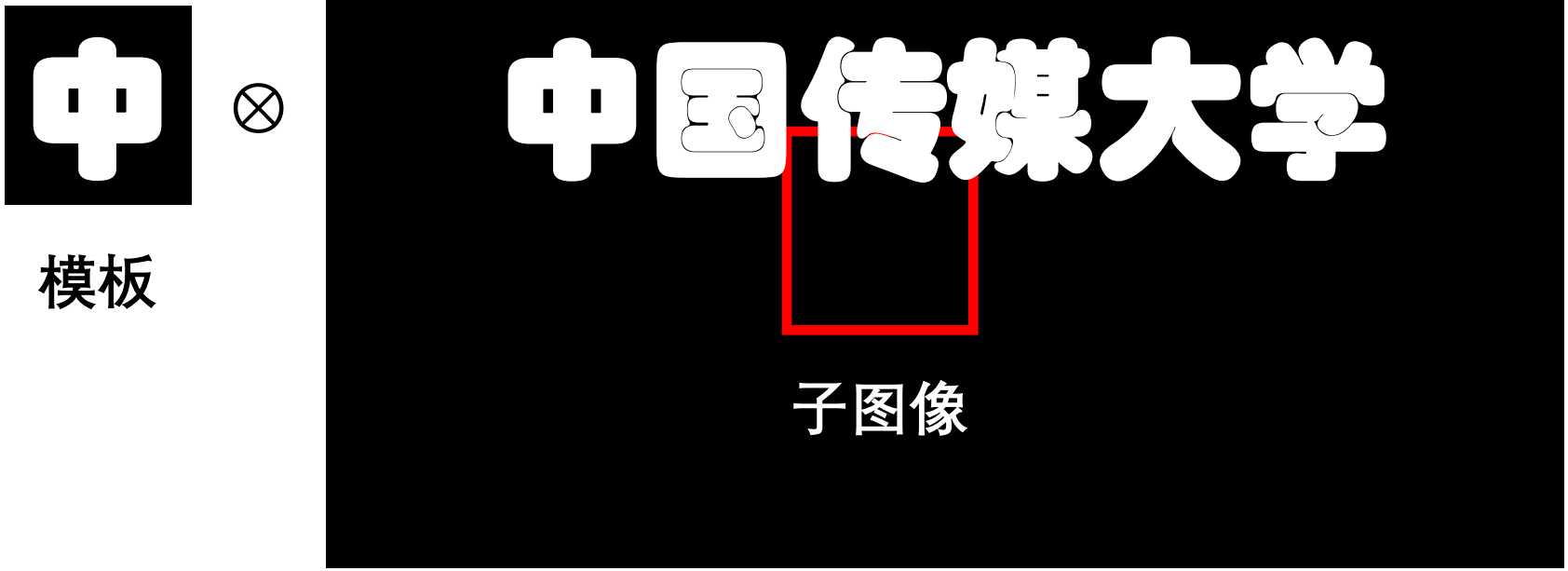


$$\text{相关性评分} = \mathbf{t} \cdot \mathbf{s} = \|\mathbf{t}\|\|\mathbf{s}\| \cos\theta$$

$$\text{归一化评分} = \frac{\mathbf{t}}{\|\mathbf{t}\|} \cdot \frac{\mathbf{s}}{\|\mathbf{s}\|} = \cos\theta$$

$$\text{归一化互相关} = \frac{\mathbf{t} - \bar{\mathbf{t}}}{\|\mathbf{t} -} \quad \frac{\mathbf{s} - \bar{\mathbf{s}}}{}$$

平均亮度



$$\text{相关性评分} = \mathbf{t} \cdot \mathbf{s} = \|\mathbf{t}\|\|\mathbf{s}\| \cos\theta$$

$$\text{归一化评分} = \frac{\mathbf{t}}{\|\mathbf{t}\|} \cdot \frac{\mathbf{s}}{\|\mathbf{s}\|} = \cos\theta$$

$$\text{归一化互相关} = \frac{\mathbf{t} - \bar{\mathbf{t}}}{\|\mathbf{t} - \bar{\mathbf{t}}\|} \cdot \frac{\mathbf{s} - \bar{\mathbf{s}}}{\|\mathbf{s} - \bar{\mathbf{s}}\|}$$

Where's Waldo?

归一化互相关滑动窗口匹配

归一化互相关结果

归一化互相关结果


检测结果


检测结果

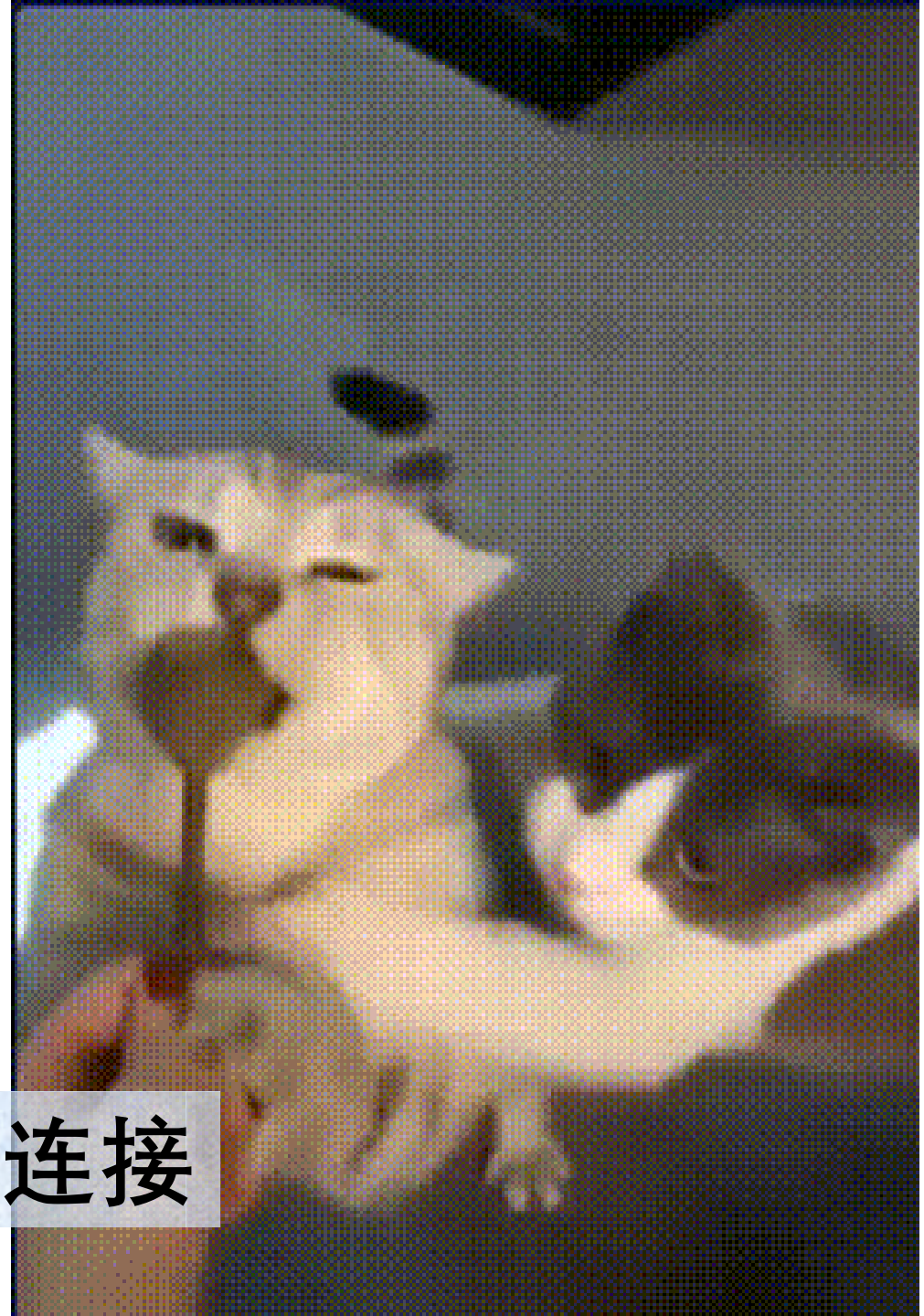

如果模板和子图像不
完全相同怎么办?

视角的差异

不同的大小

不同的实例


遮挡



高度关节连接

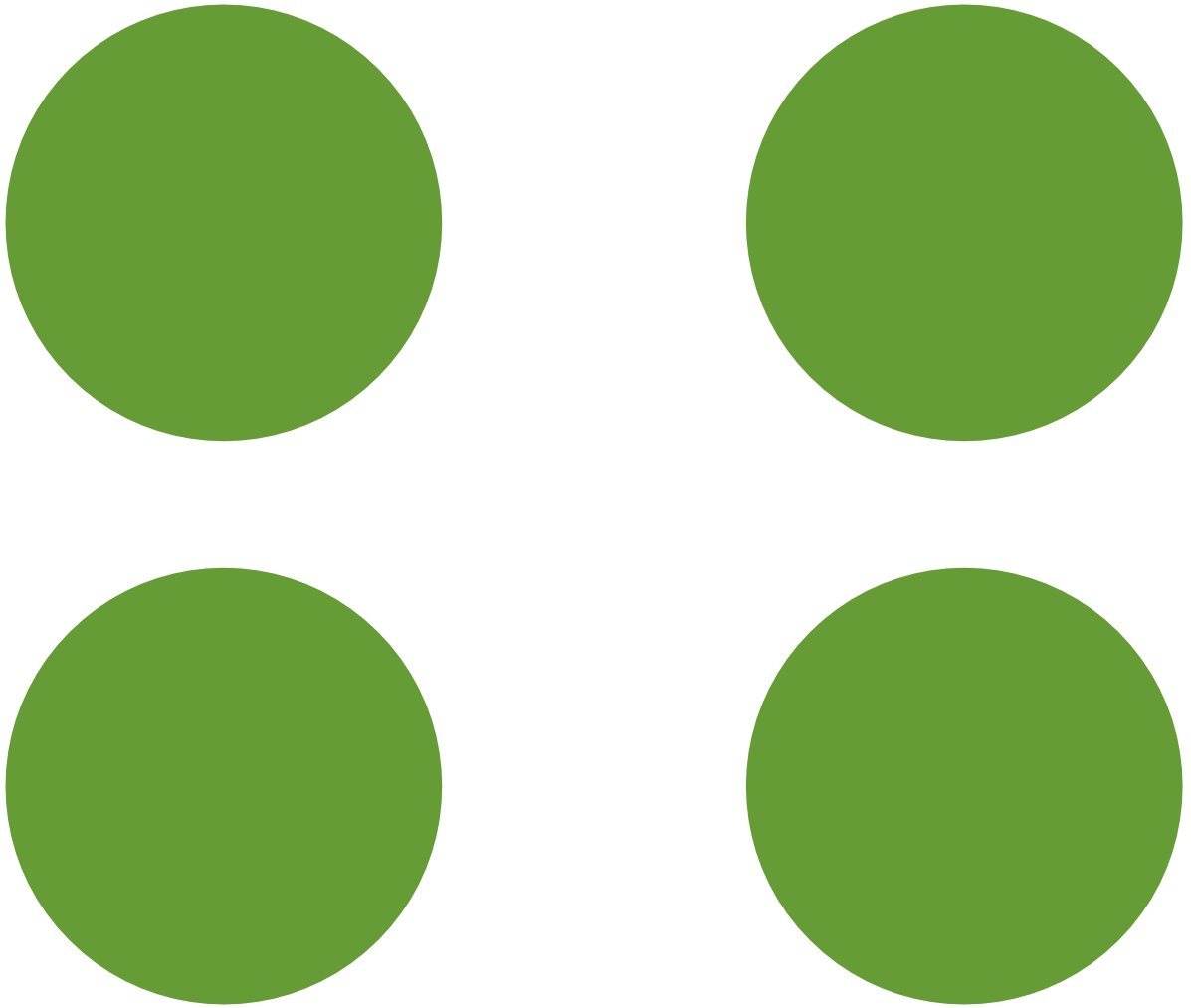
对比

Contrast

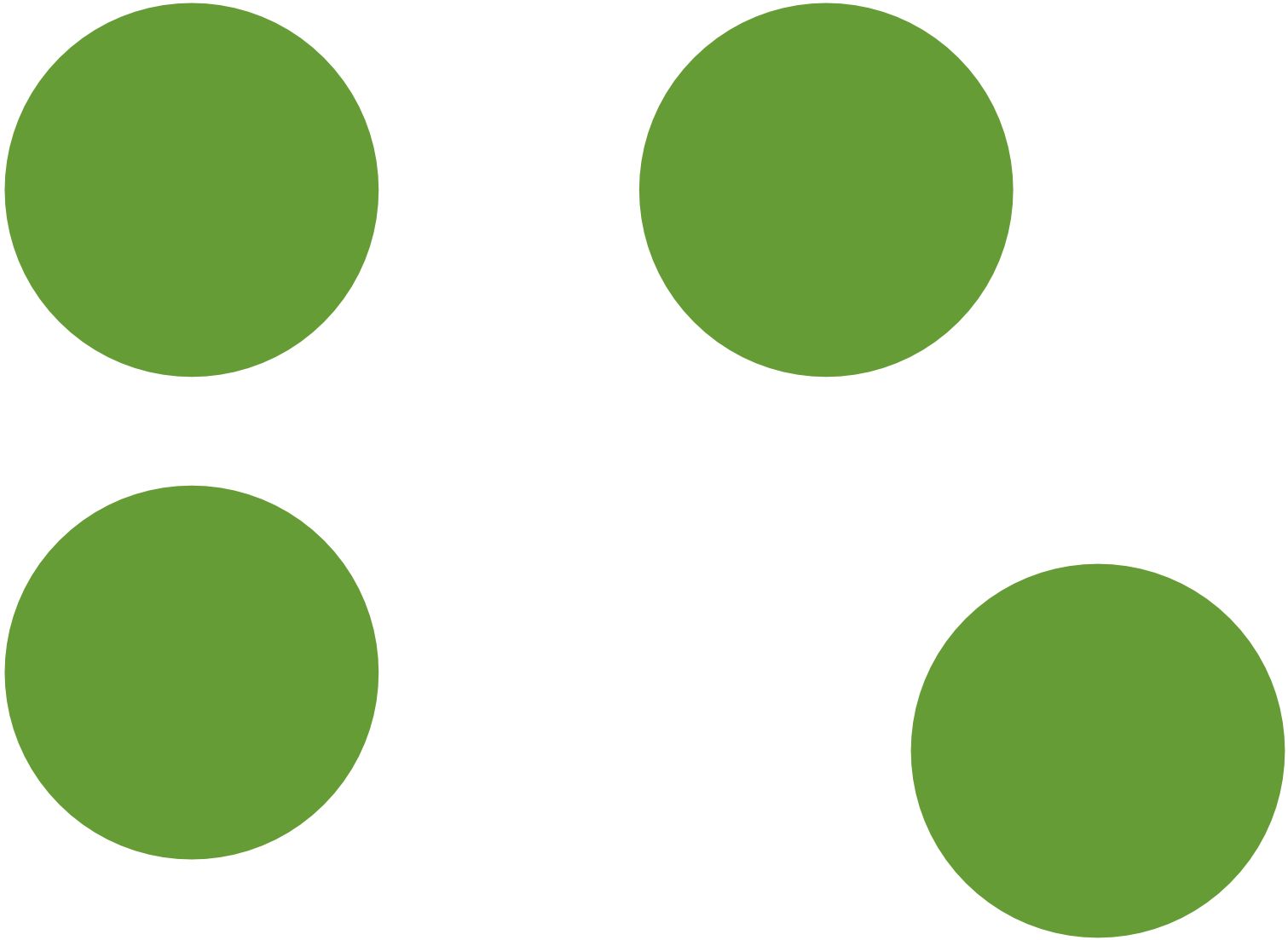
“Without **contrast**

you’re dead.”

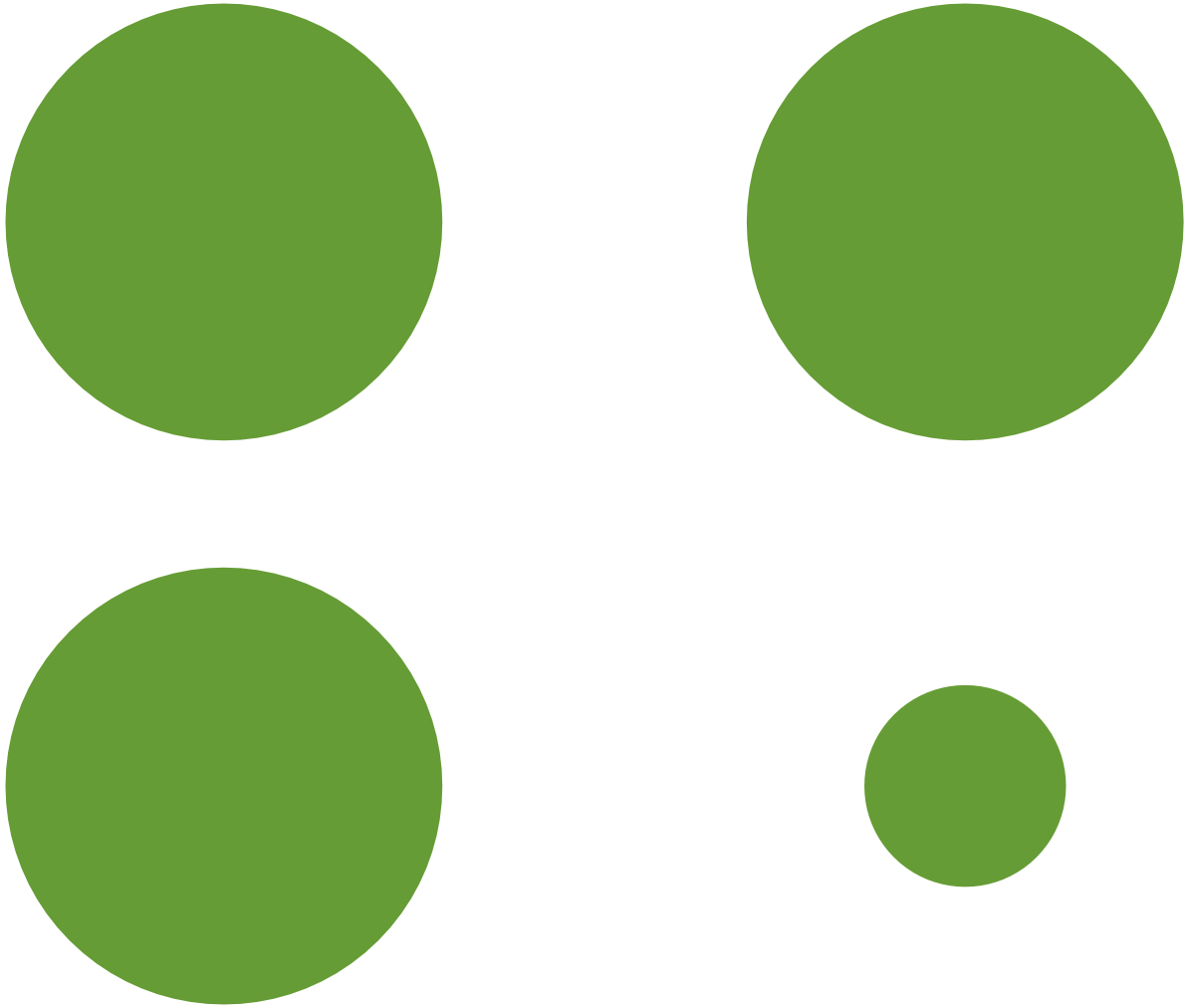
—Paul Rand



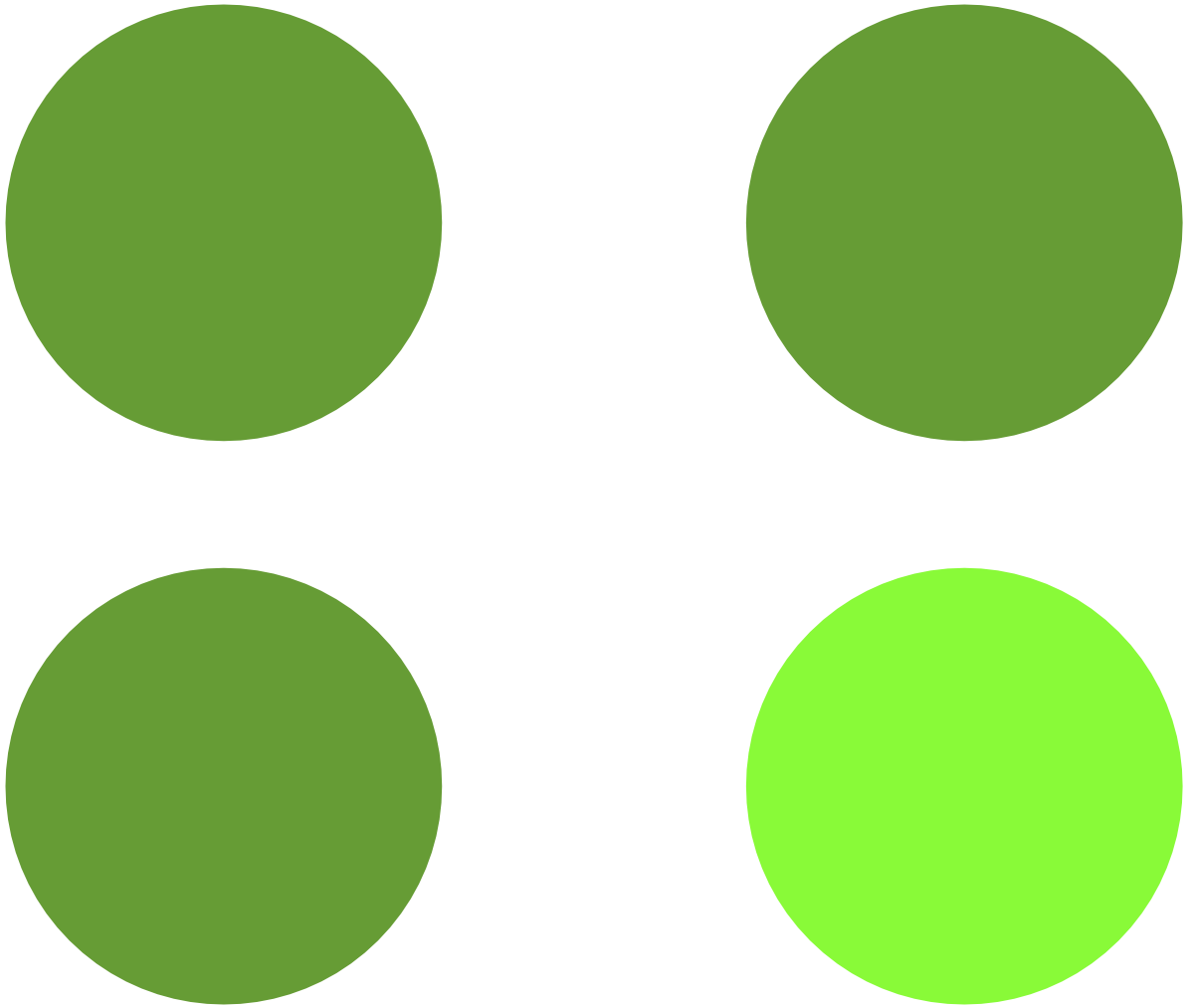
位置的对比



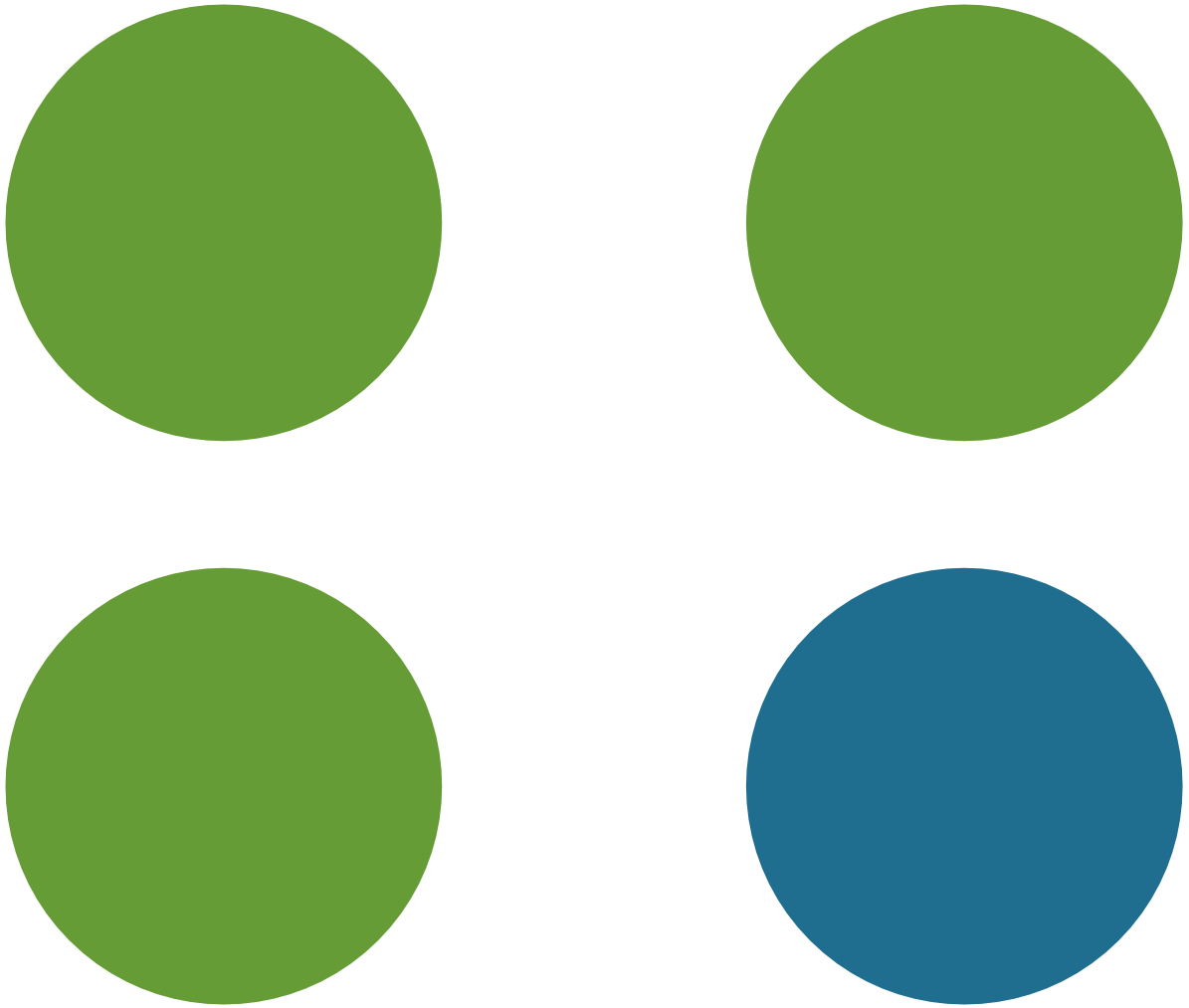
大小的对比



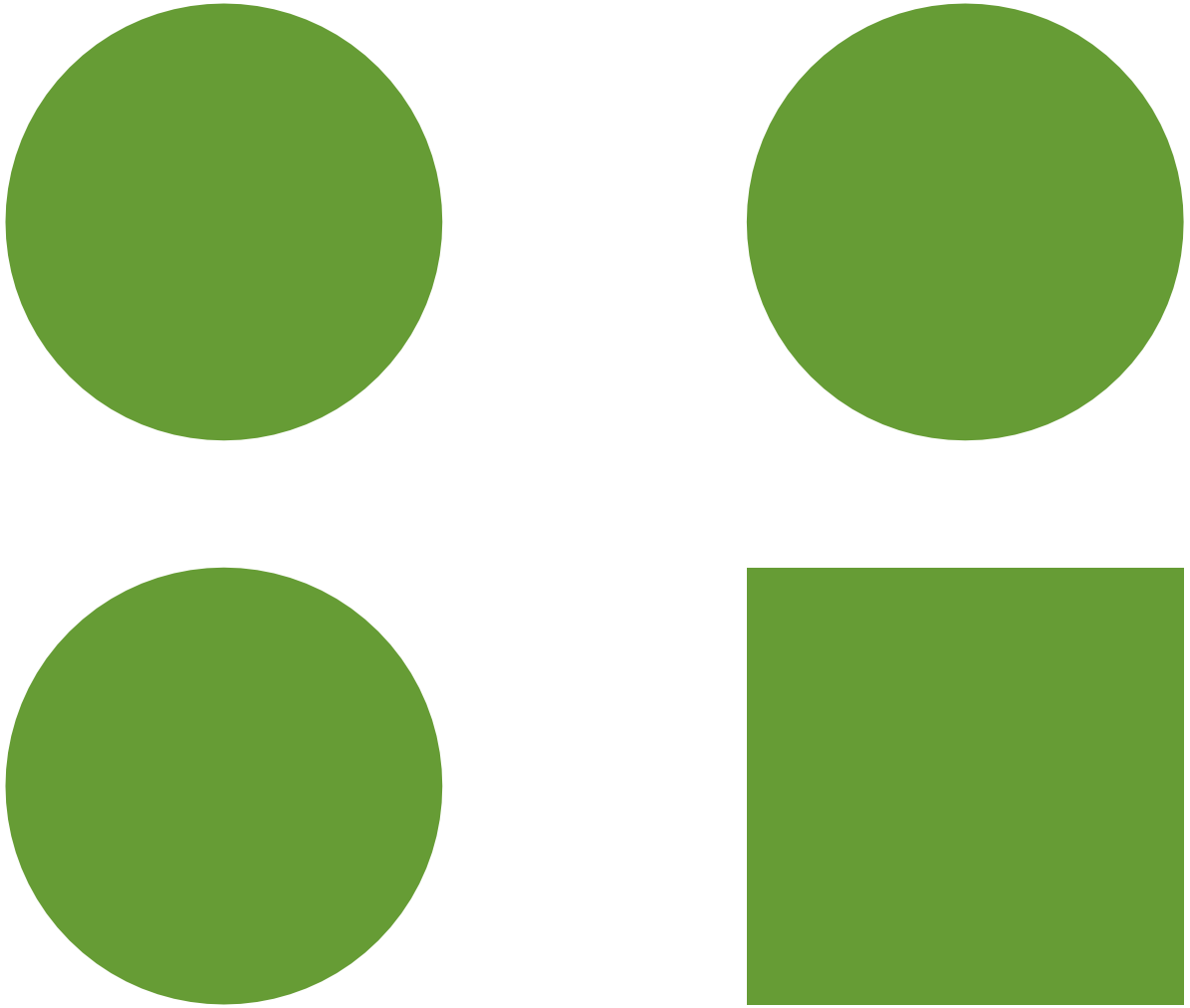
亮度的对比



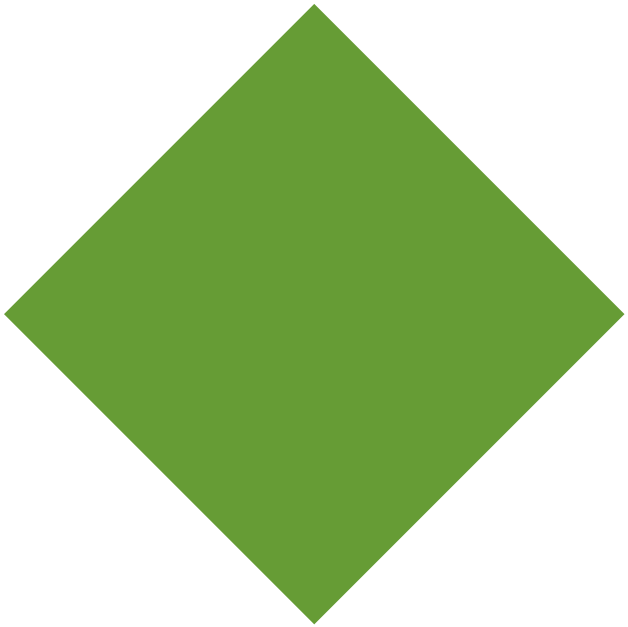
色彩的对比



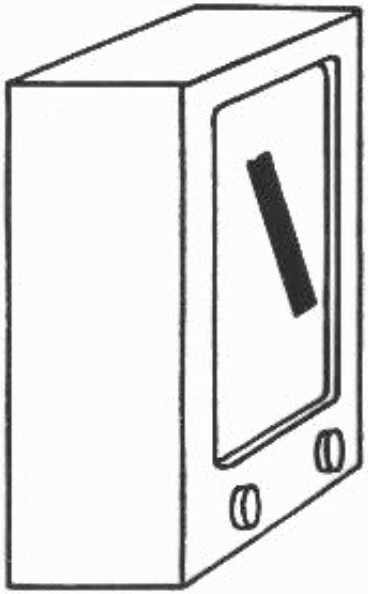
形状的对比

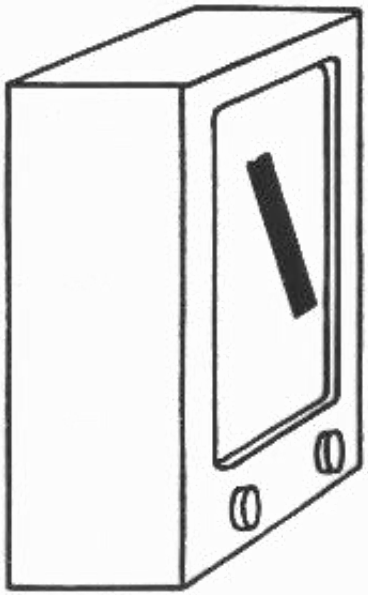


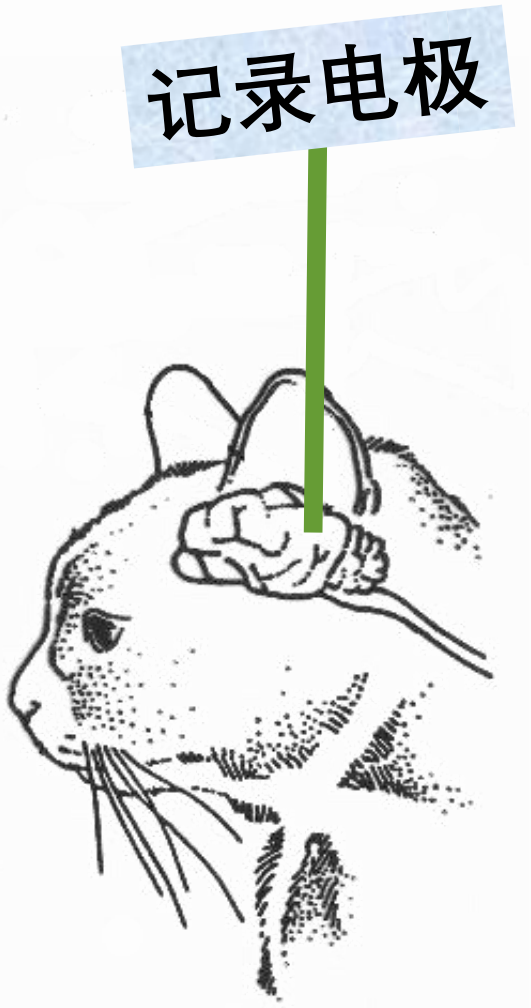
朝向的对比

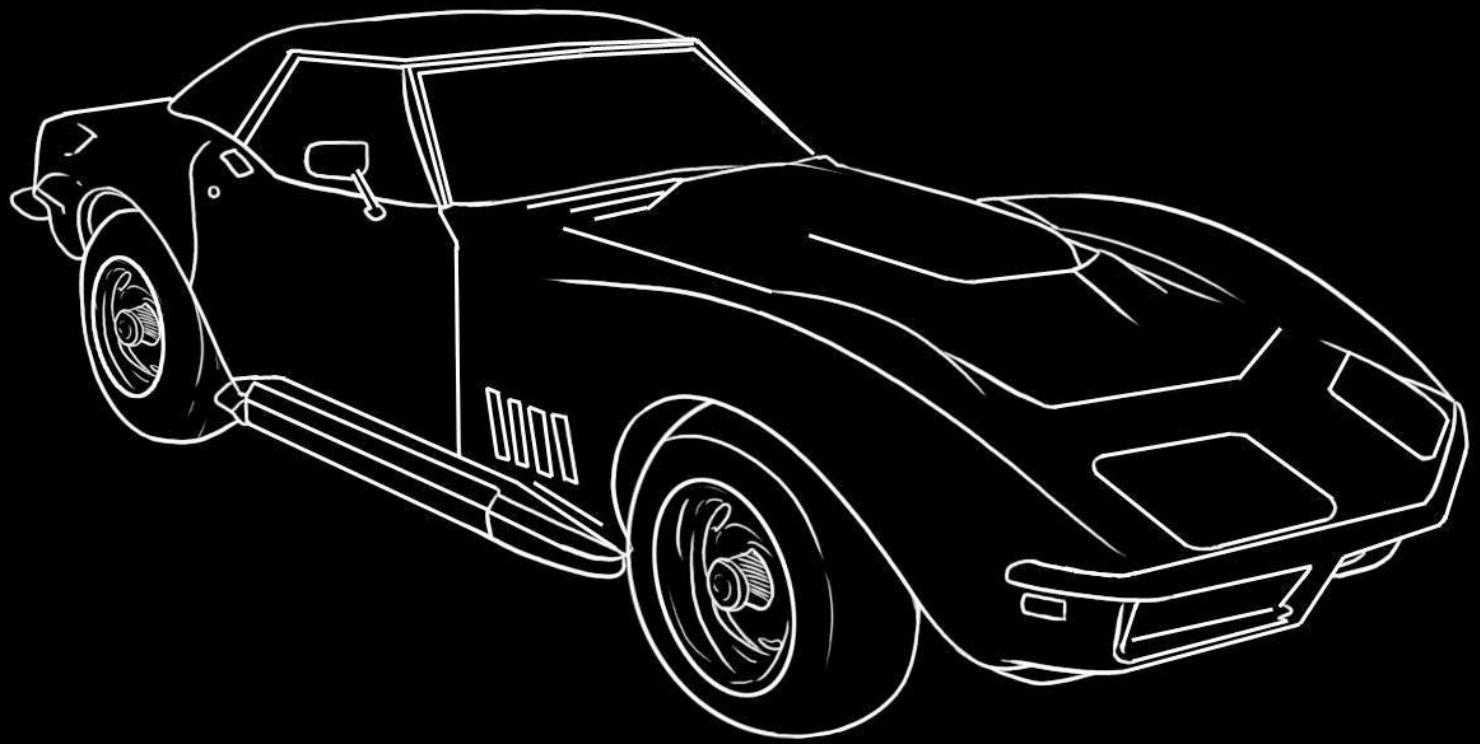


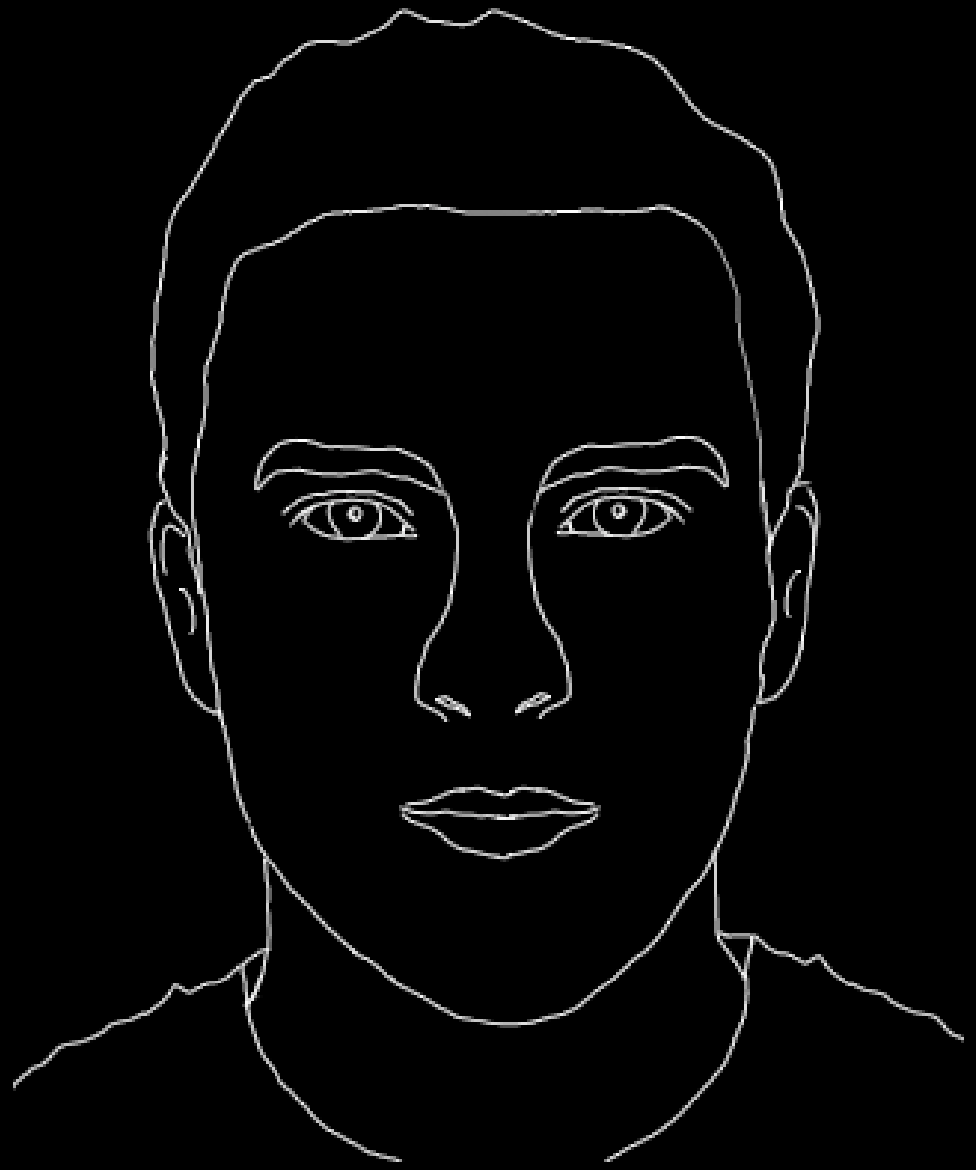
大卫·休伯和图斯坦·威瑟

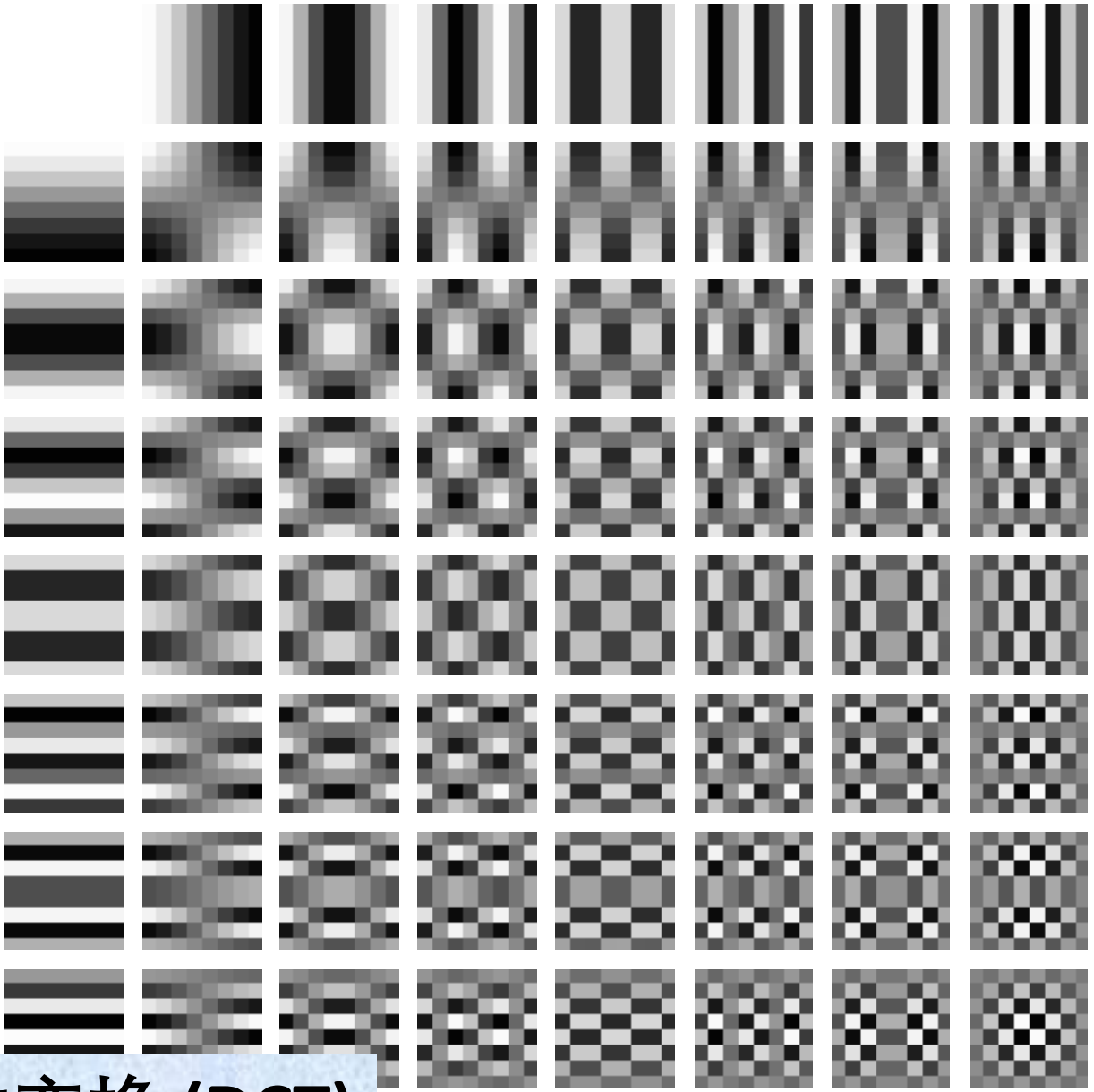




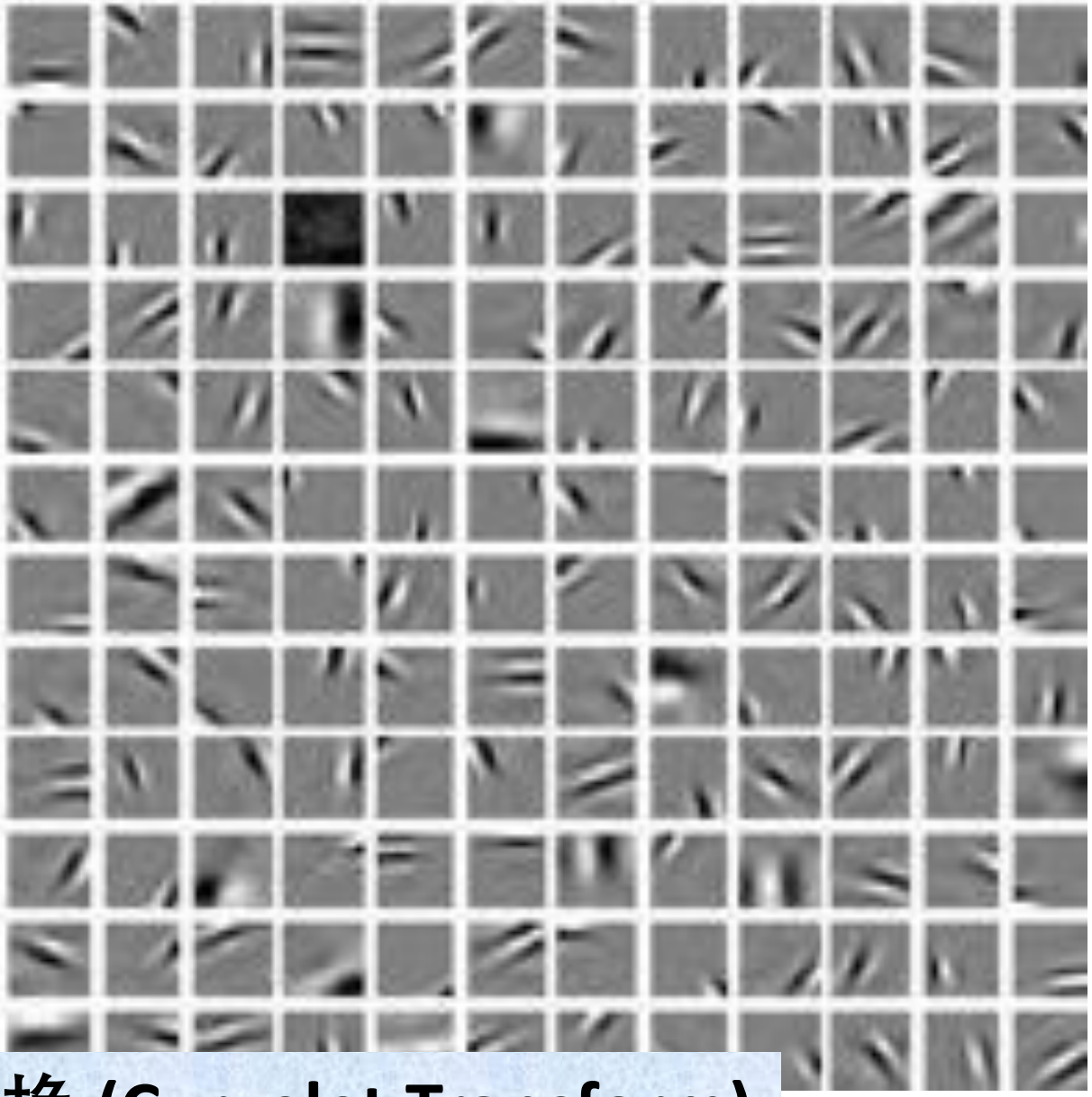

记录电极



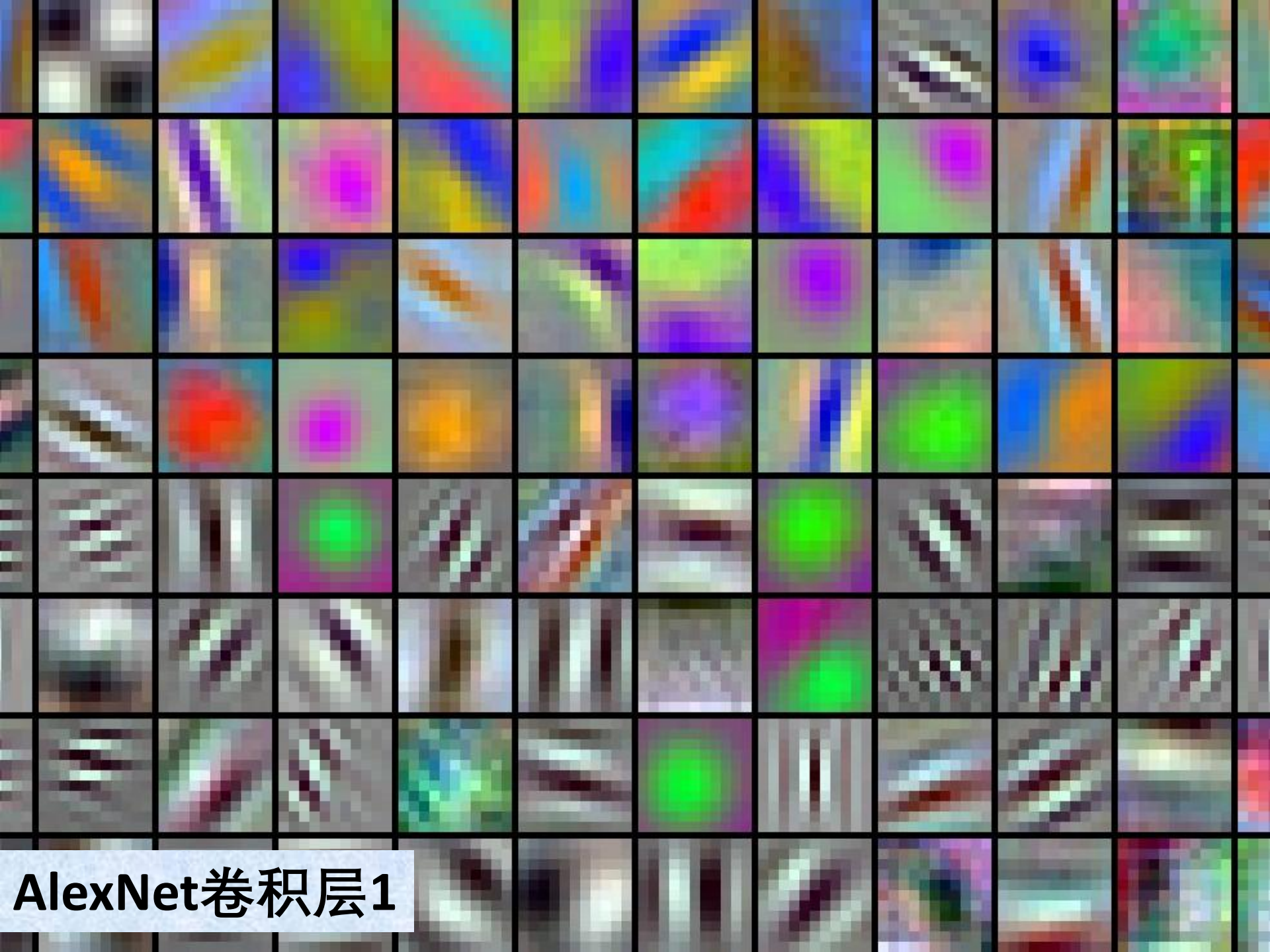




离散余弦变换 (DCT)



曲线波变换 (Curvelet Transform)


AlexNet卷积层1


STATE
LIBRARY


反射率变换


深度不连续

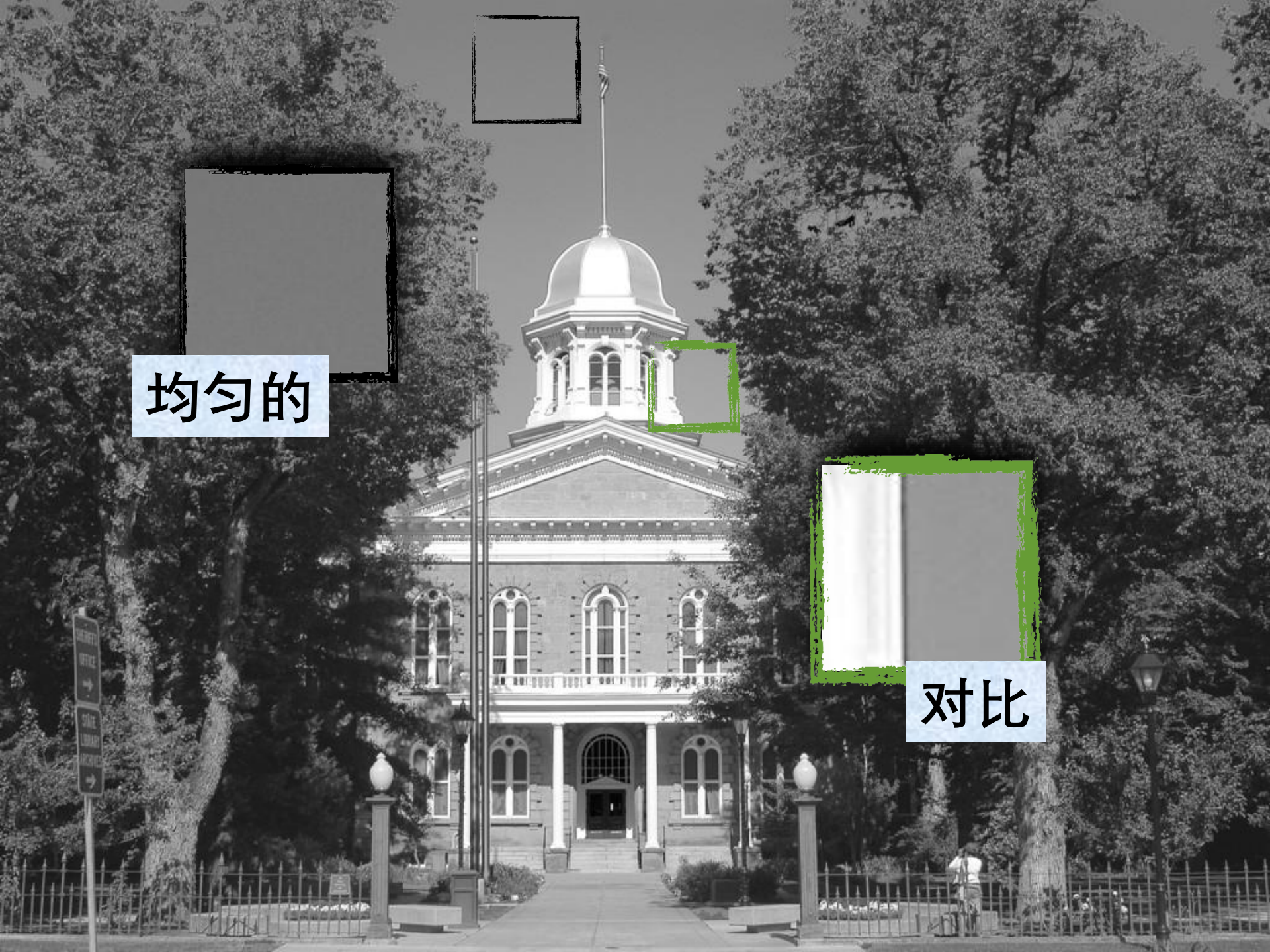

投影

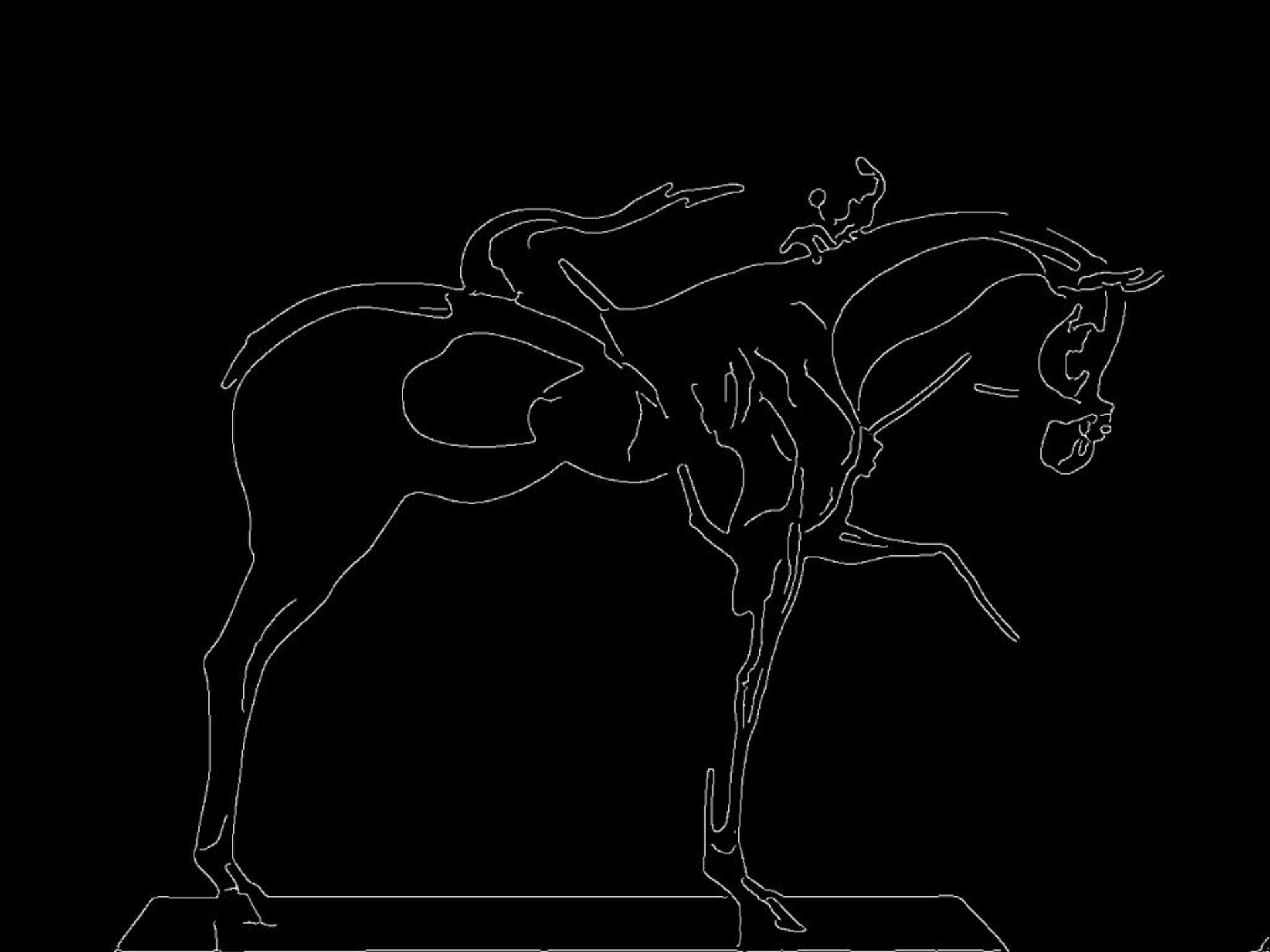

表面朝向变化

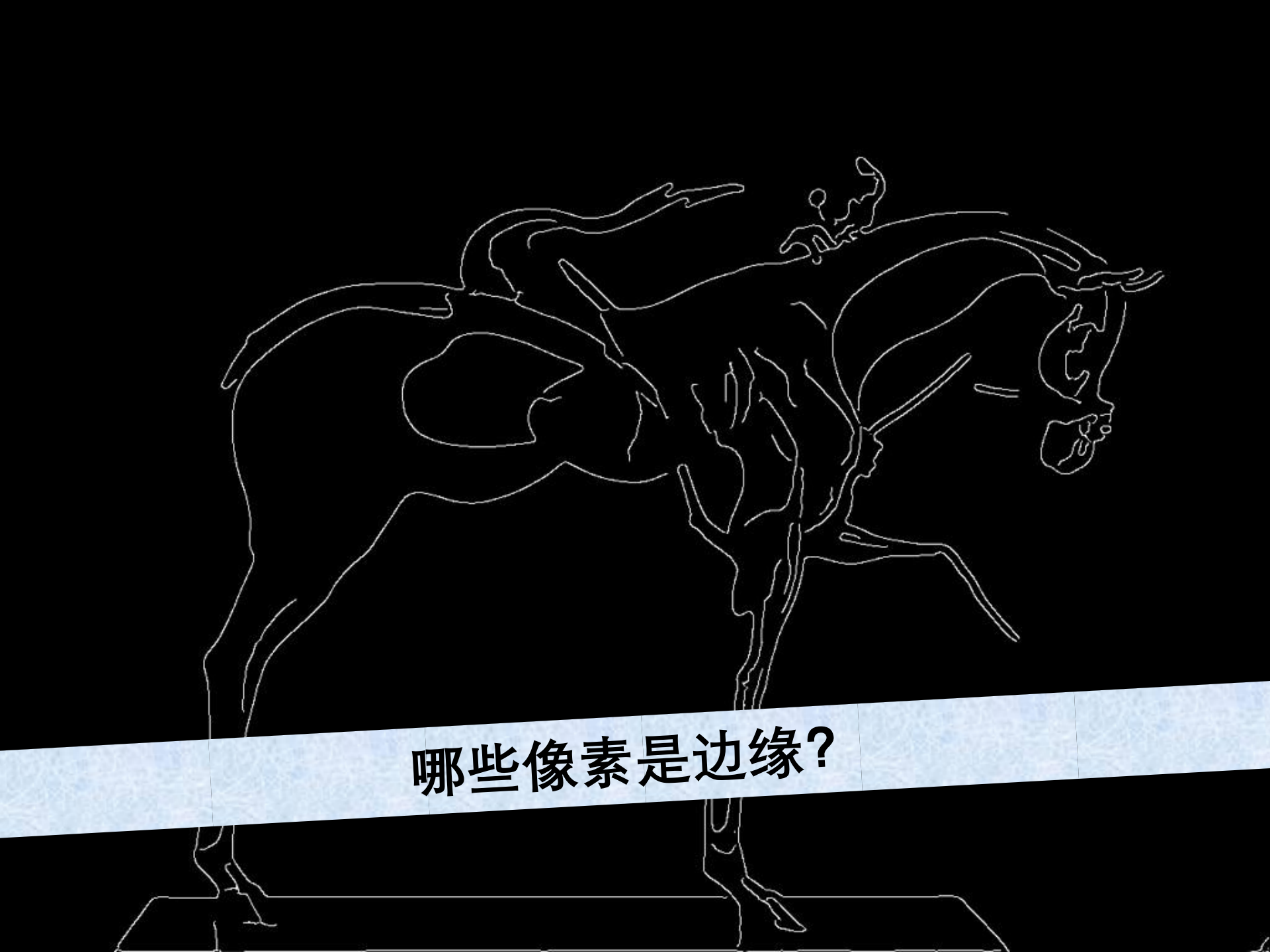

STATE
LIBRARY

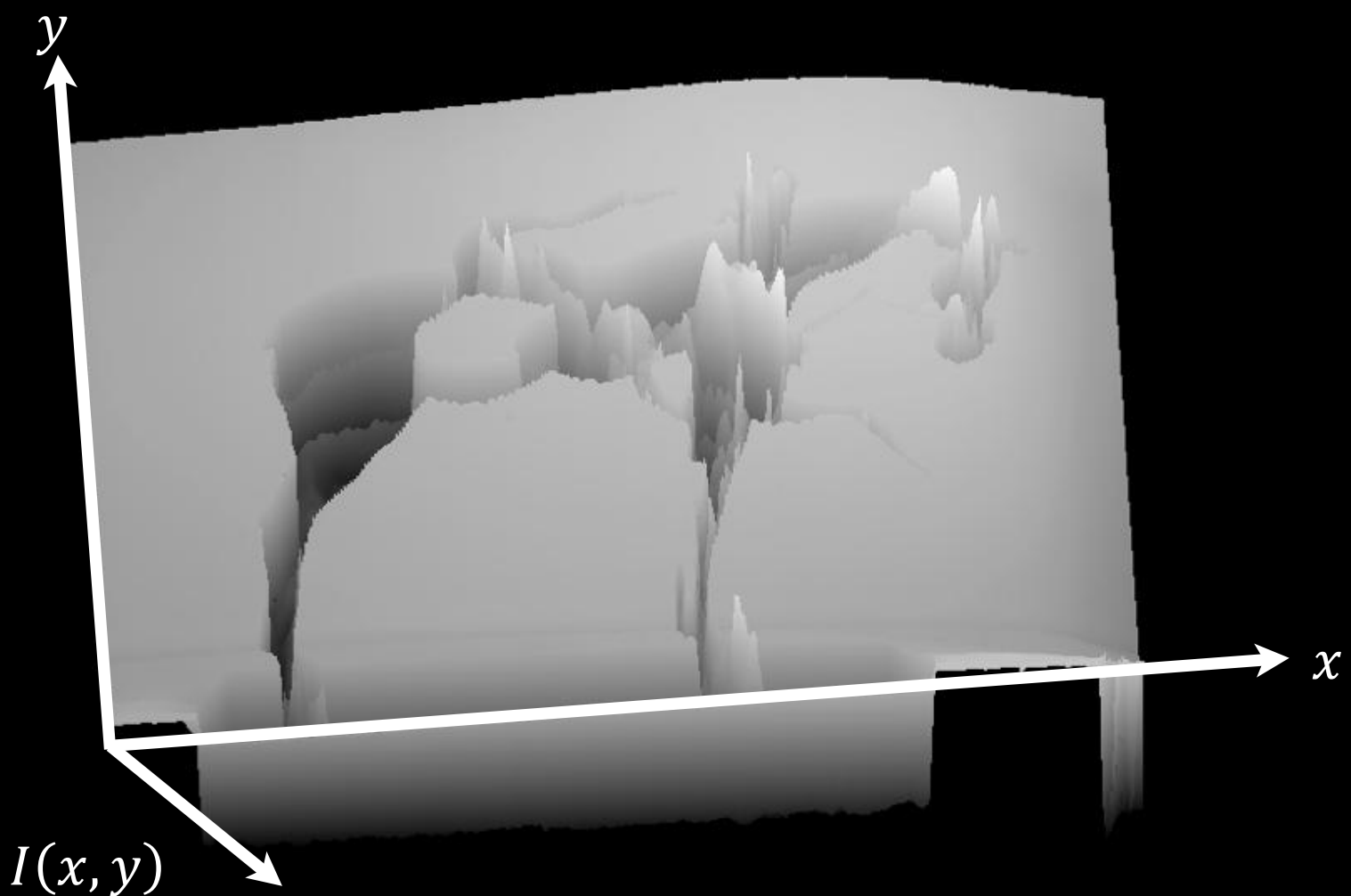

均匀的

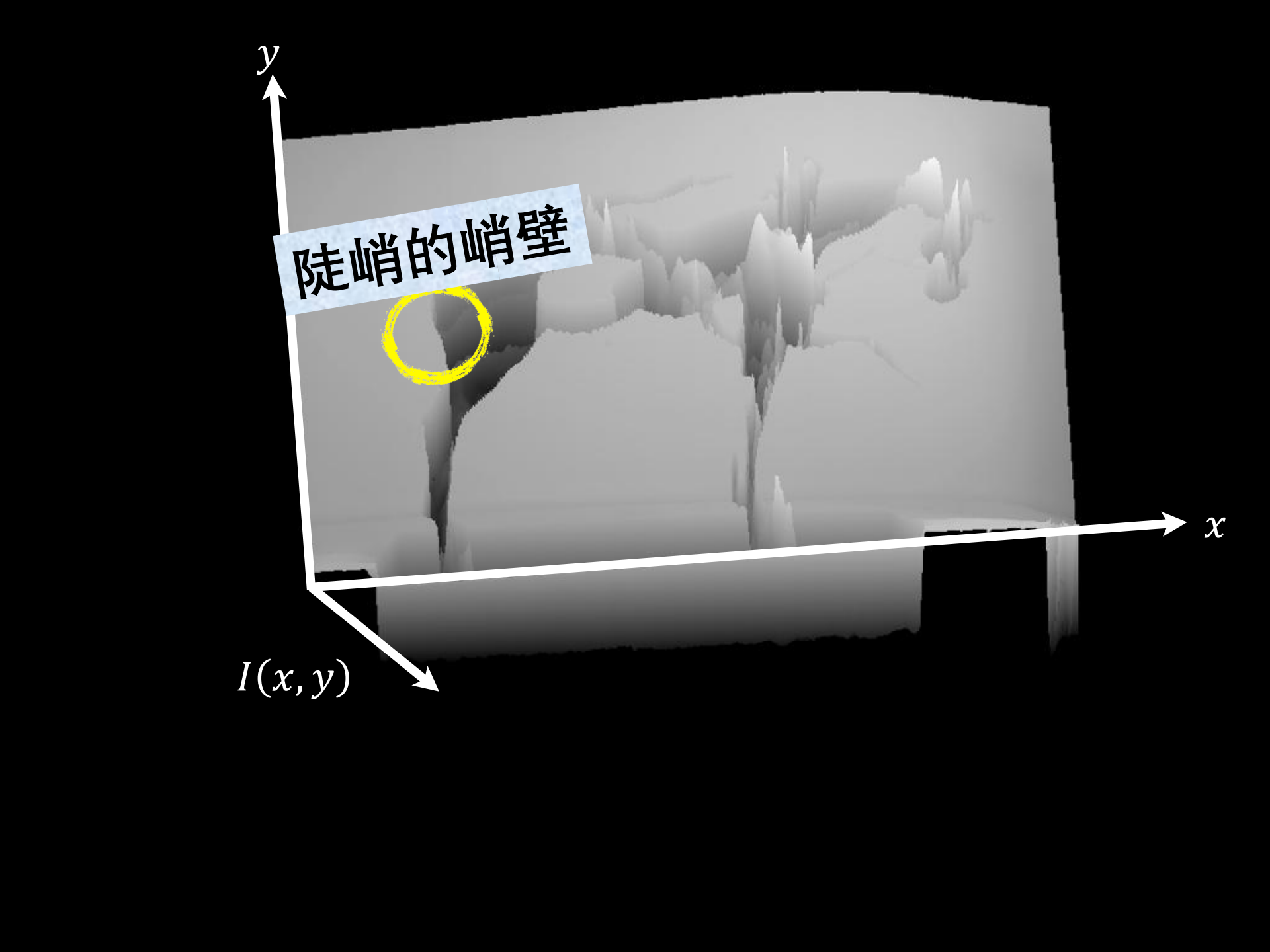

均匀的

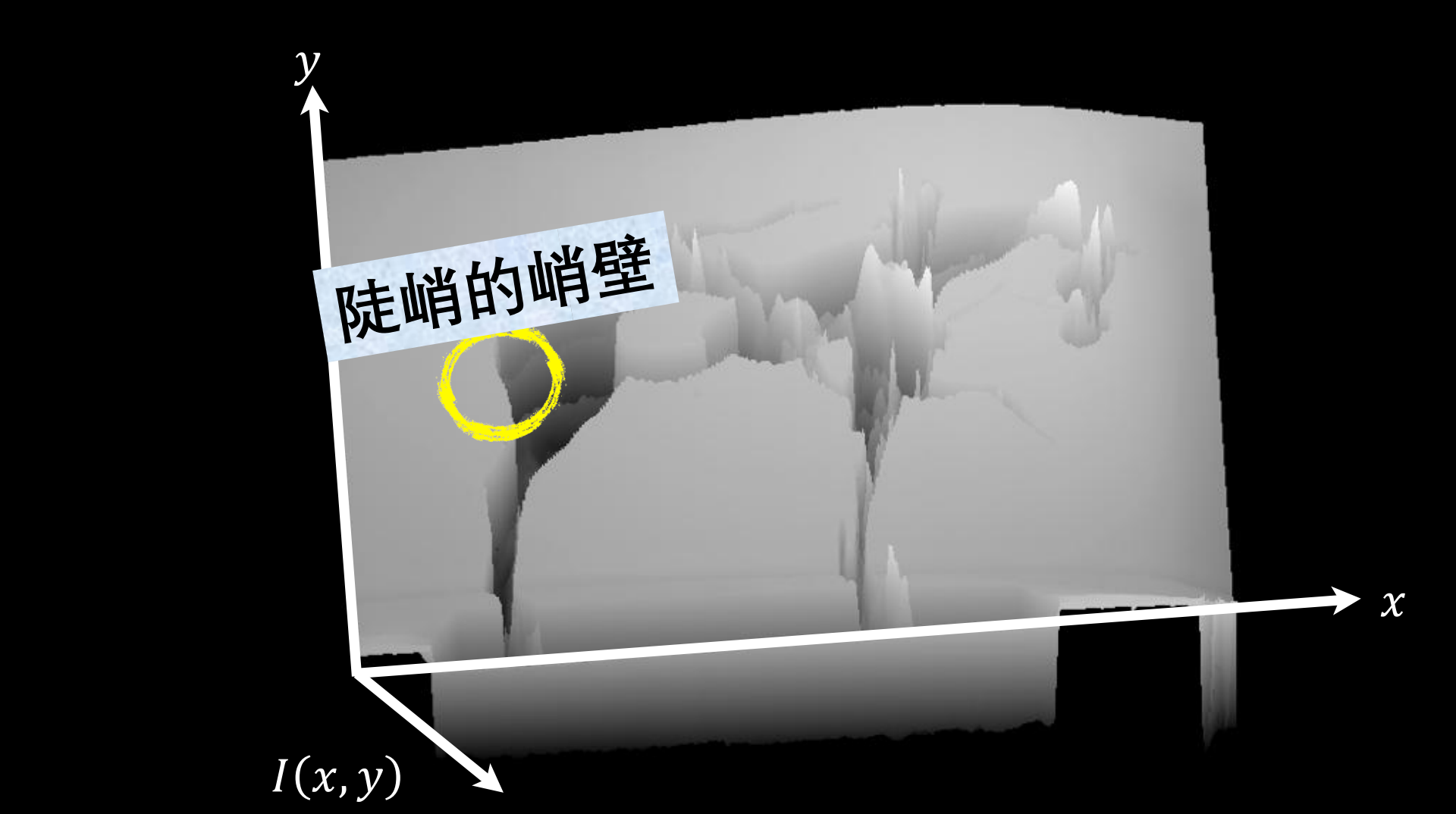

均匀的
对比




哪些像素是边缘?

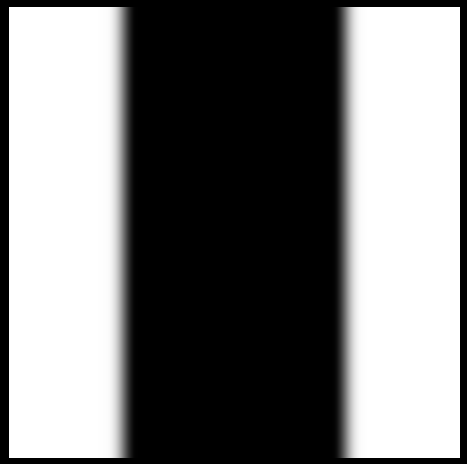

y
x
I(x, y)


y
陡峭的峭壁
x
I(x, y)



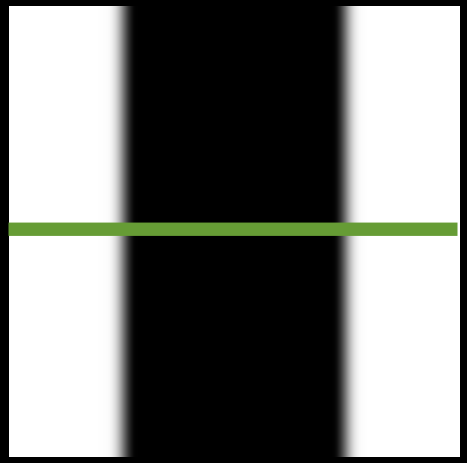
目标： 识别图像中的突然局部变化

边缘检测



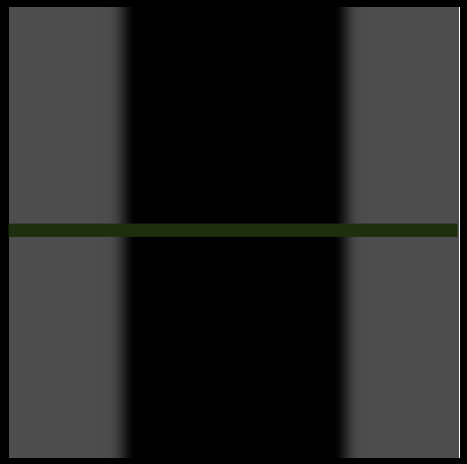
图像

边缘检测

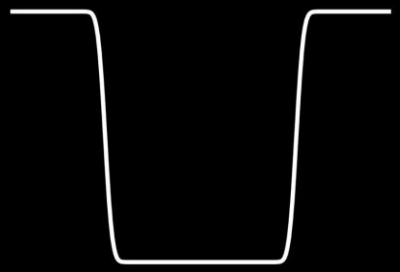


图像

边缘检测

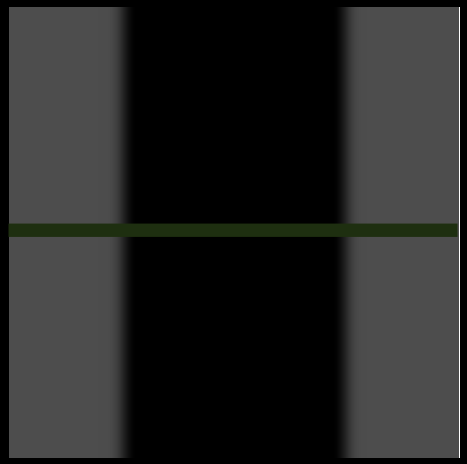


图像

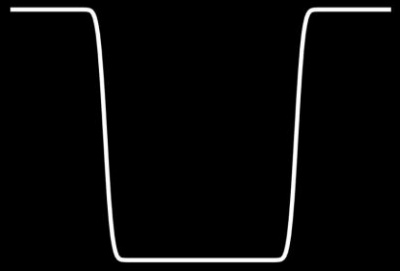


强度函数
（切片）

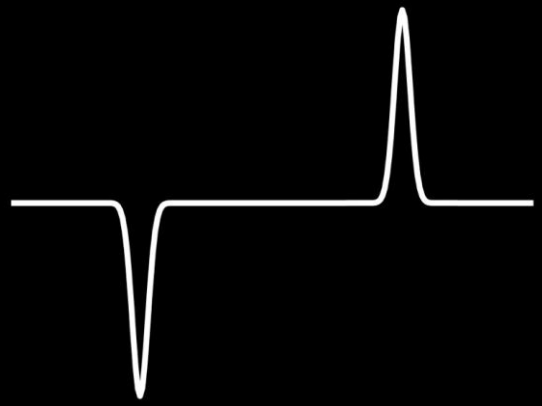
边缘检测



图像

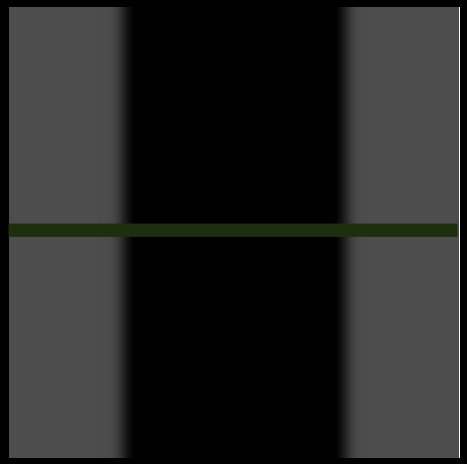


强度函数
（切片）

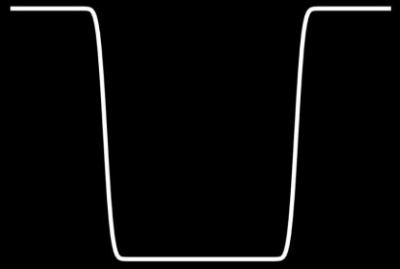


一阶导数

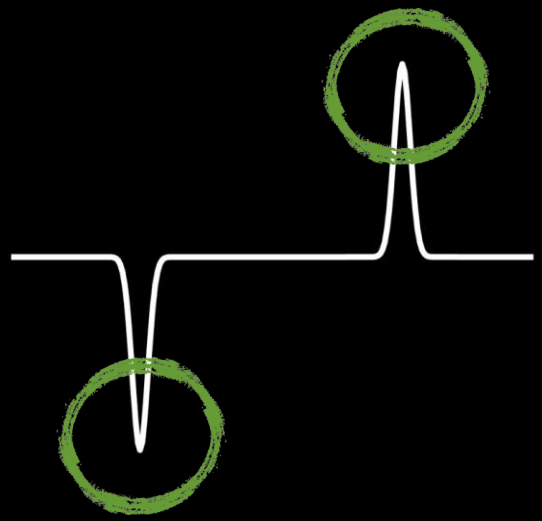
边缘检测



图像

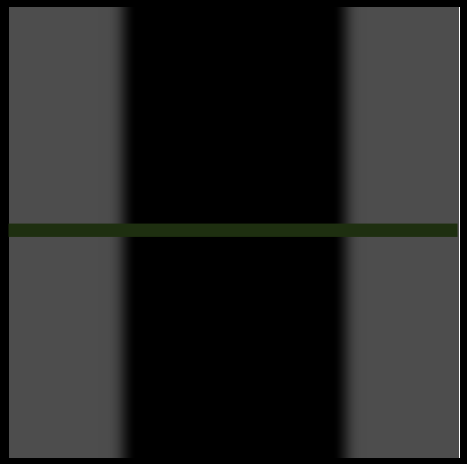


强度函数
（切片）

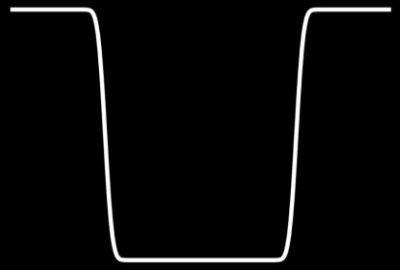


一阶导数

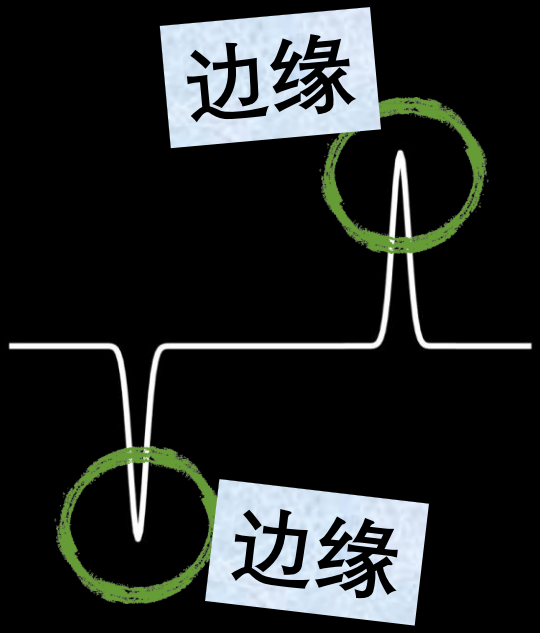
边缘检测



图像



强度函数
（切片）

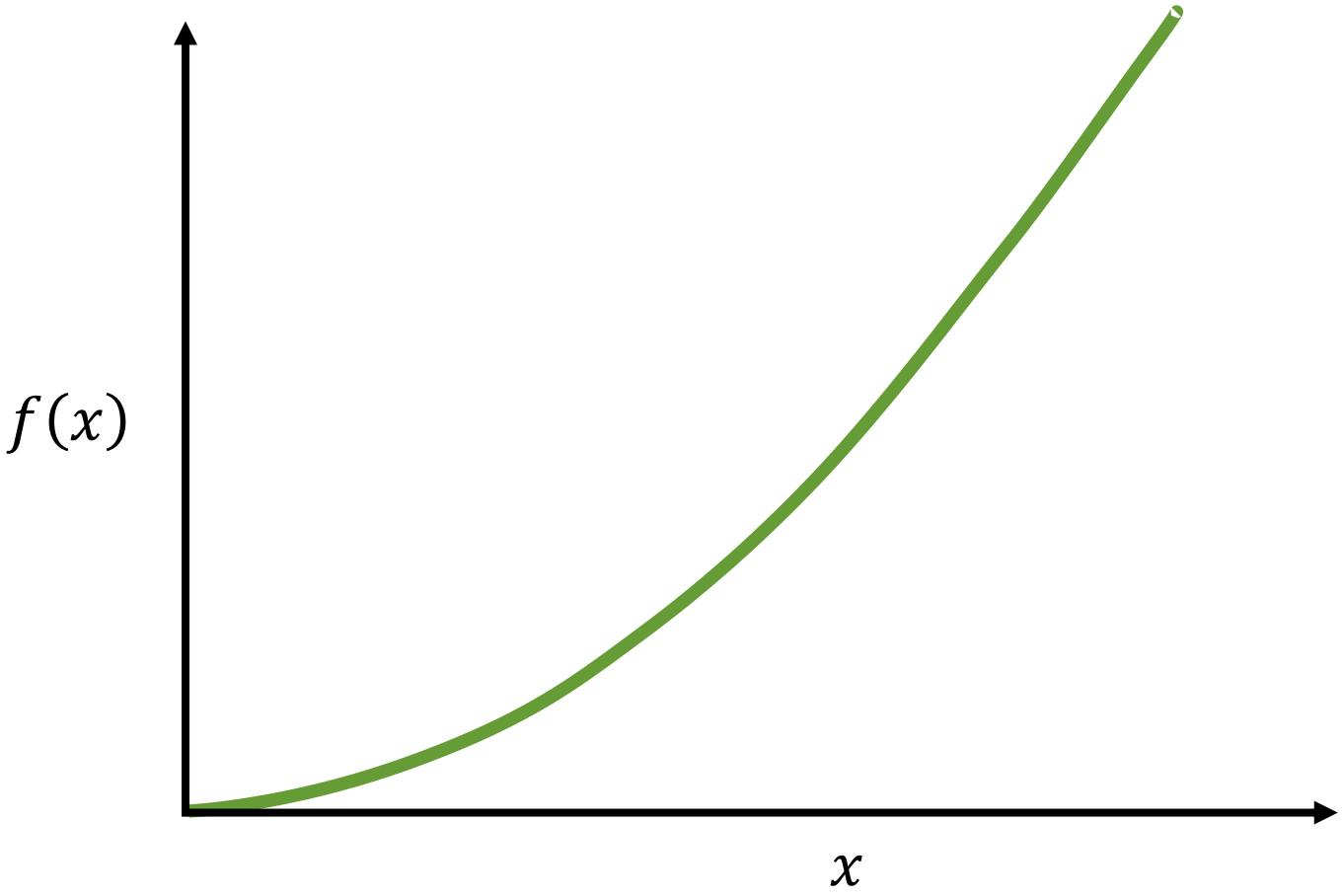


一阶导数

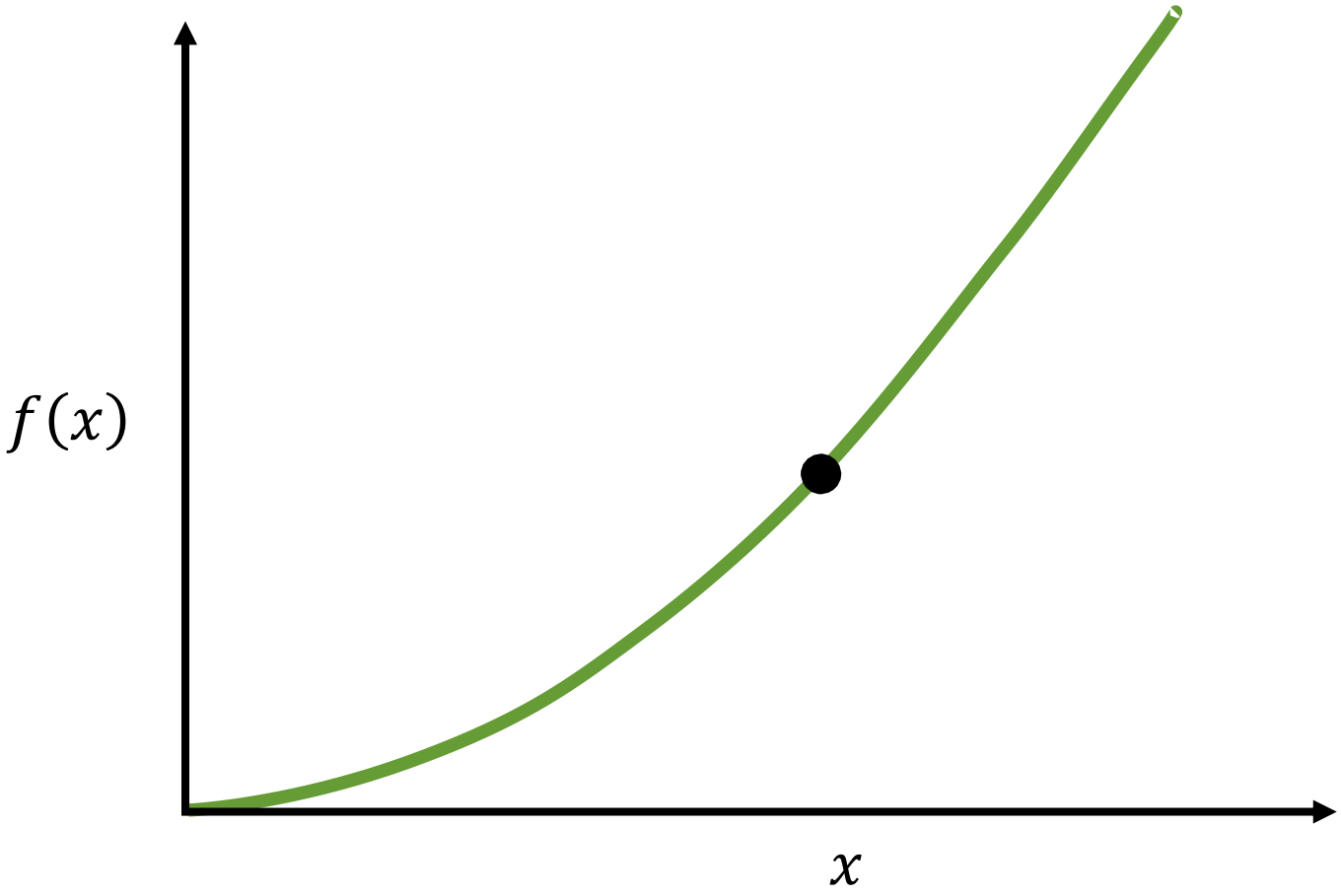
导数

$$\frac{df(x)}{dx} = \lim_{\varepsilon \to 0} \frac{f(x+\varepsilon) - f(x)}{\varepsilon}$$

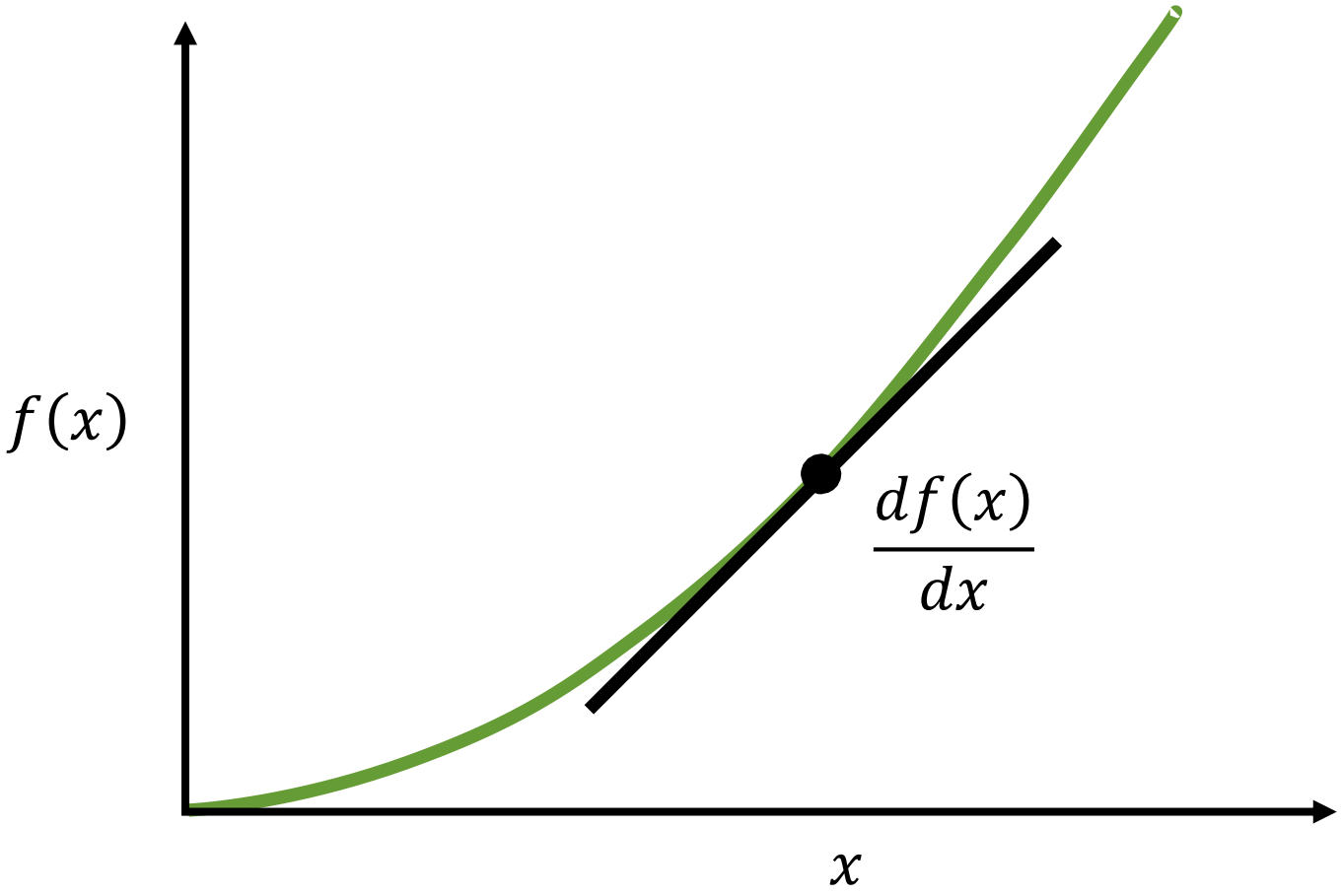
$$\frac{df(x)}{dx} = \lim_{\varepsilon \to 0} \frac{f(x+\varepsilon) - f(x)}{\varepsilon}$$



$$\frac{df(x)}{dx} = \lim_{\varepsilon \to 0} \frac{f(x+\varepsilon) - f(x)}{\varepsilon}$$



$$\frac{df(x)}{dx} = \lim_{\varepsilon \to 0} \frac{f(x+\varepsilon) - f(x)}{\varepsilon}$$



偏导数

偏导数

对于一个2D函数，$f(x,y)$，它的偏导数为:

偏导数

对于一个2D函数，$f(x,y)$，它的偏导数为:

$$\frac{\partial f(x,y)}{\partial x} = \lim_{\varepsilon \to 0} \frac{f(x+\varepsilon, y) - f(x,y)}{\varepsilon}$$

偏导数

对于一个2D函数，$f(x,y)$，它的偏导数为:

$$\frac{\partial f(x,y)}{\partial x} = \lim_{\varepsilon \to 0} \frac{f(x+\varepsilon, y) - f(x,y)}{\varepsilon}$$

$$\frac{\partial f(x,y)}{\partial y} = \lim_{\varepsilon \to 0} \frac{f(x, y+\varepsilon) - f(x,y)}{\varepsilon}$$

对于一个2D函数，$f(x,y)$，它的偏导数为:

$$\frac{\partial f(x,y)}{\partial x} = \lim_{\varepsilon \to 0} \frac{f(x+\varepsilon, y) - f(x,y)}{\varepsilon}$$

$$\frac{\partial f(x,y)}{\partial y} = \lim_{\varepsilon \to 0} \frac{f(x, y+\varepsilon) - f(x,y)}{\varepsilon}$$

对于离散数据:

对于一个2D函数，$f(x,y)$，它的偏导数为:

$$\frac{\partial f(x,y)}{\partial x} = \lim_{\varepsilon \to 0} \frac{f(x+\varepsilon, y) - f(x,y)}{\varepsilon}$$

$$\frac{\partial f(x,y)}{\partial y} = \lim_{\varepsilon \to 0} \frac{f(x, y+\varepsilon) - f(x,y)}{\varepsilon}$$

对于离散数据:

$$\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x+1, y) - f(x,y)}{1}$$

对于一个2D函数，$f(x,y)$，它的偏导数为:

$$\frac{\partial f(x,y)}{\partial x} = \lim_{\varepsilon \to 0} \frac{f(x+\varepsilon, y) - f(x,y)}{\varepsilon}$$

$$\frac{\partial f(x,y)}{\partial y} = \lim_{\varepsilon \to 0} \frac{f(x, y+\varepsilon) - f(x,y)}{\varepsilon}$$

对于离散数据:

$$\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x+1,y) - f(x,y)}{1}$$

$$= f(x+1,y) - f(x,y)$$

有限微分

$$\frac{\partial f(x,y)}{\partial x} \approx f(x+1,y) - f(x,y)$$

$$\frac{\partial f(x,y)}{\partial y} \approx f(x,y+1) - f(x,y)$$

有限微分

$$\frac{\partial f(x,y)}{\partial x} \approx f(x+1,y) - f(x,y)$$

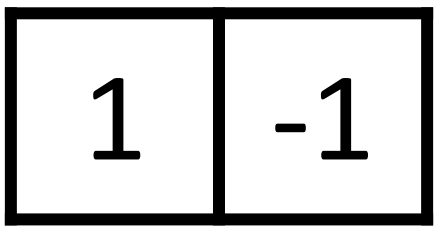
$$\frac{\partial f(x,y)}{\partial y} \approx f(x,y+1) - f(x,y)$$

对应的卷积滤波器是什么？

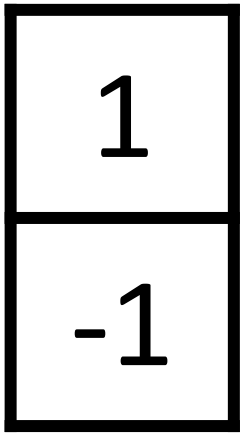
有限微分

$$\frac{\partial f(x,y)}{\partial x} \approx f(x+1,y) - f(x,y)$$

$$\frac{\partial f(x,y)}{\partial y} \approx f(x,y+1) - f(x,y)$$



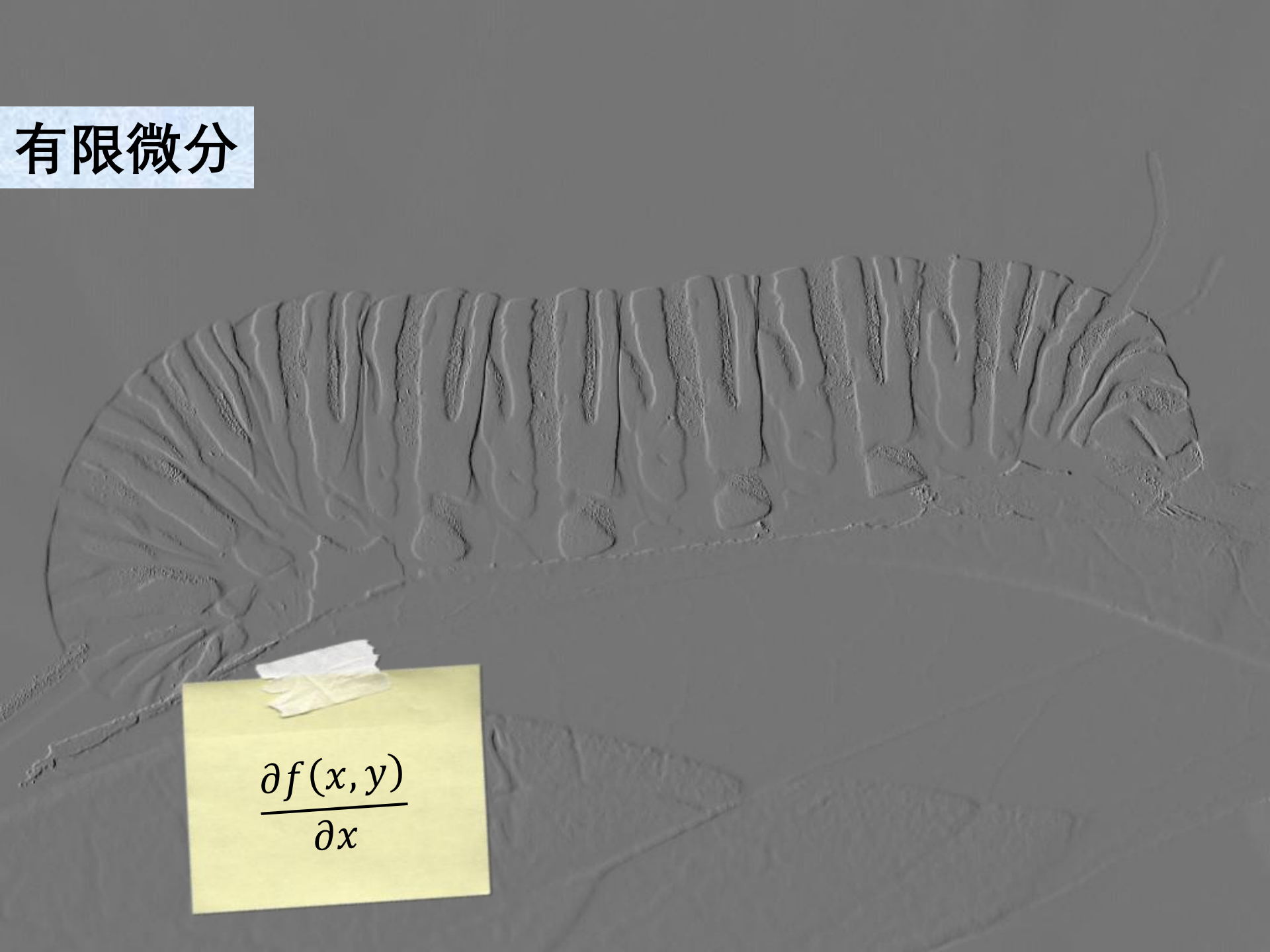
$$\frac{\partial f(x,y)}{\partial x}$$



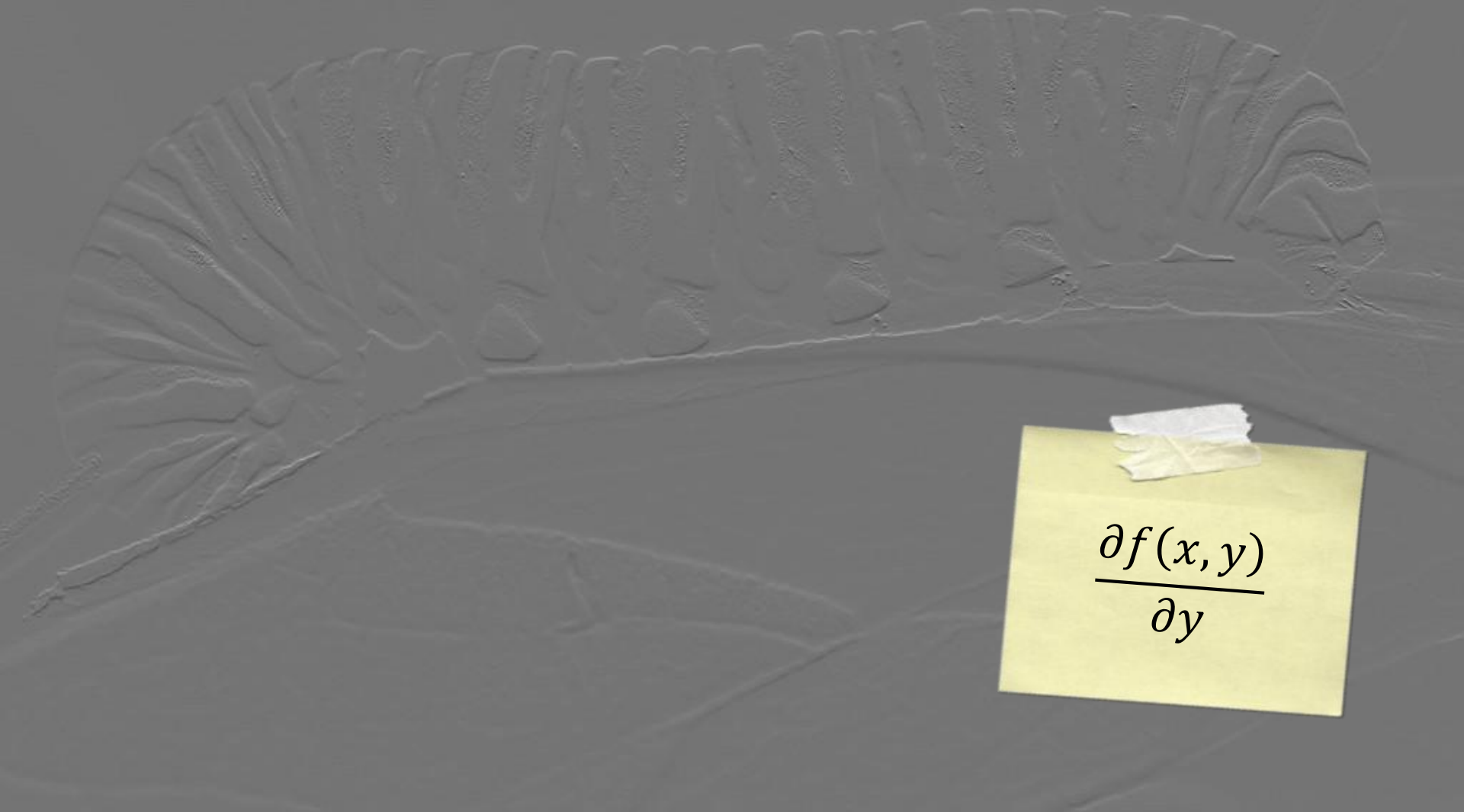
$$\frac{\partial f(x,y)}{\partial y}$$


输入图像

有限微分



有限微分



$$\frac{\partial f(x,y)}{\partial y}$$

其他微分滤波器

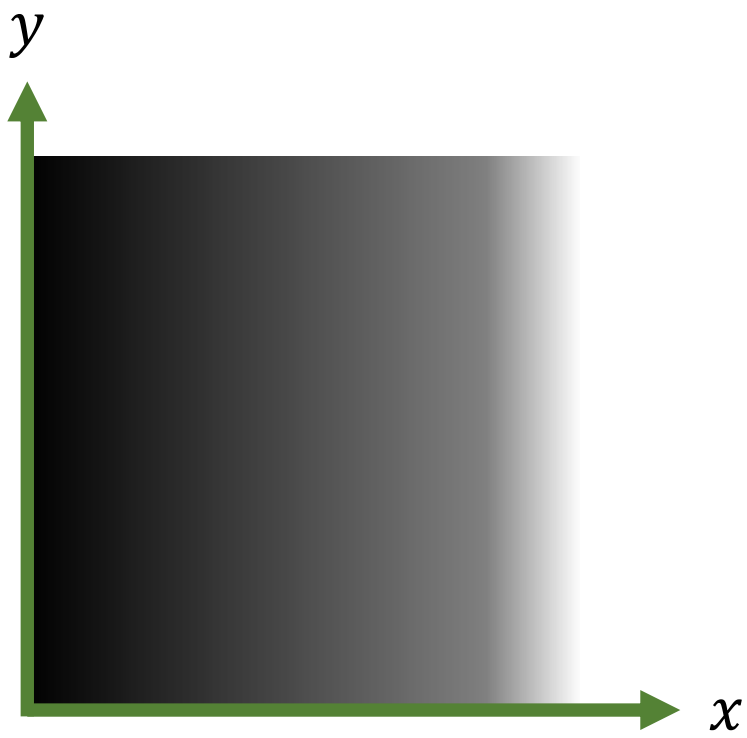
	$\frac{\partial}{\partial x}$	$\frac{\partial}{\partial y}$
Sobel	$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$	$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$
Prewitt	$\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$	$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$
Roberts	$\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$	$\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$

梯度

指向函数最大增长率方向的向量，其大小为增长率

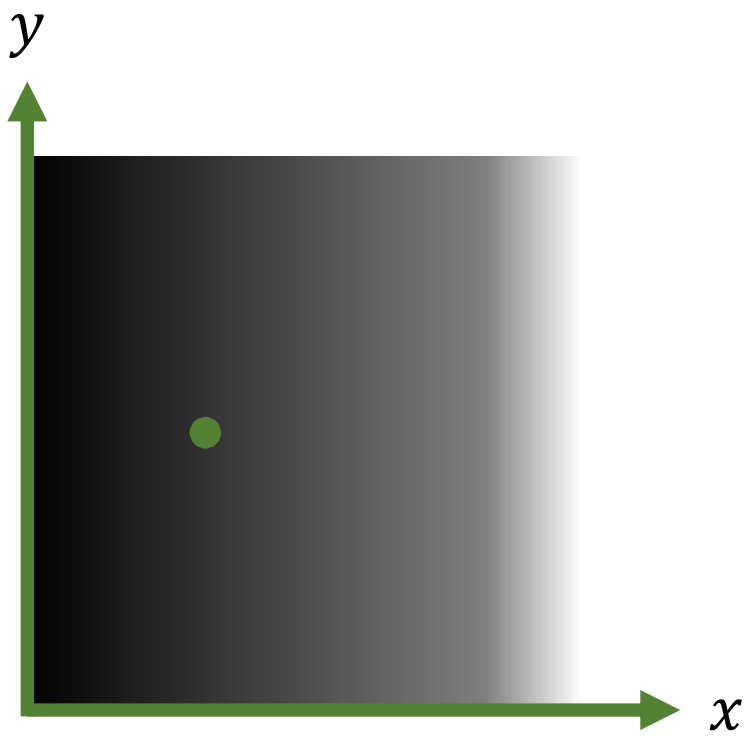
梯度

指向函数最大增长率方向的向量，其大小为增长率



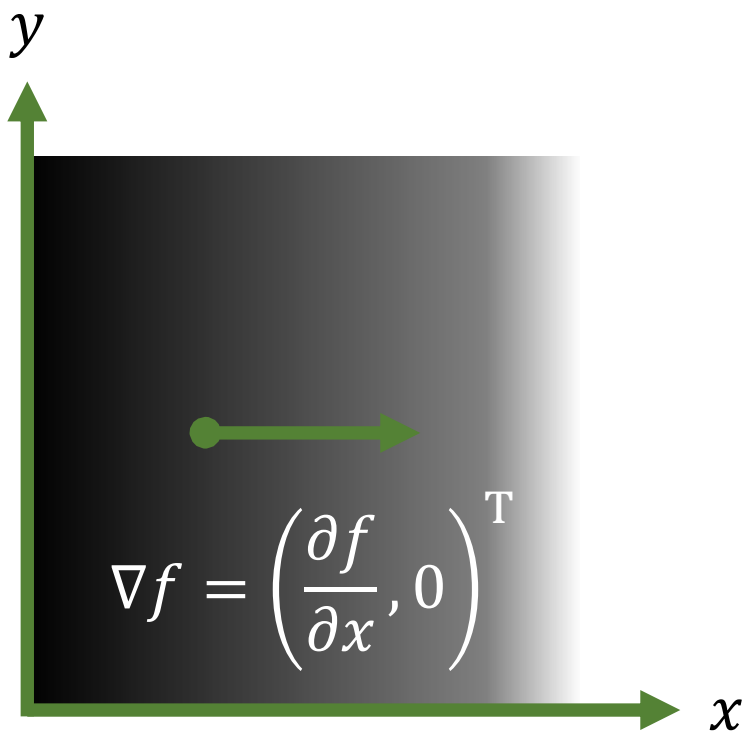
梯度

指向函数最大增长率方向的向量，其大小为增长率



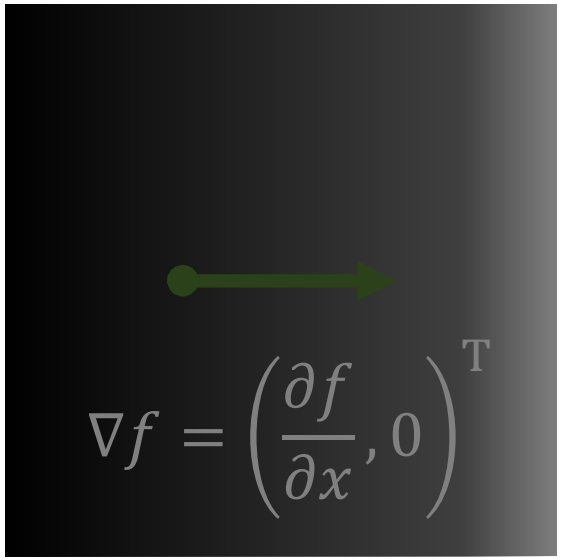
梯度

指向函数最大增长率方向的向量，其大小为增长率



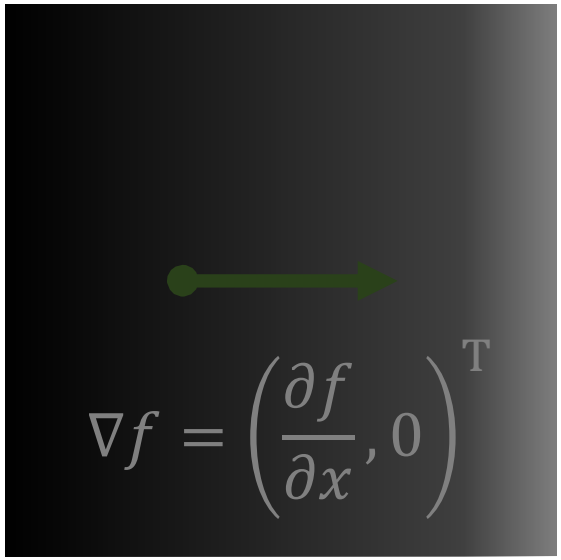
梯度

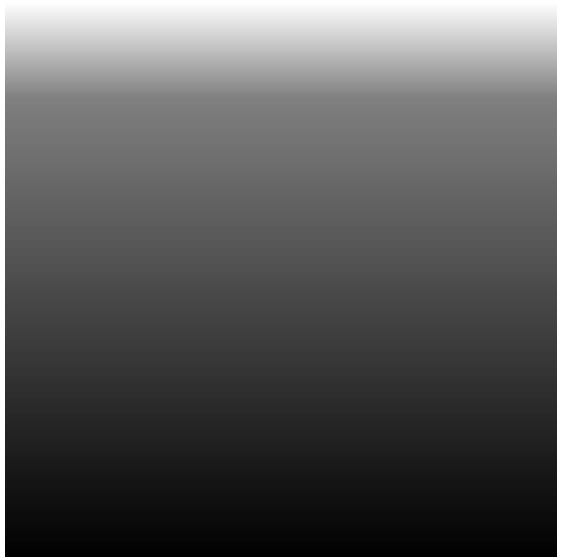
指向函数最大增长率方向的向量，其大小为增长率



梯度

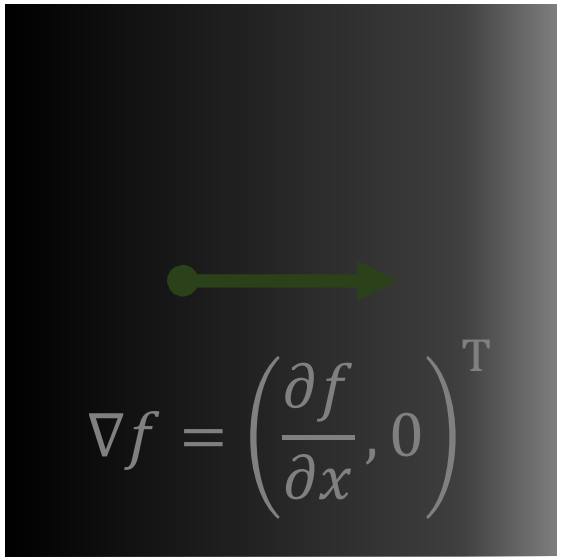
指向函数最大增长率方向的向量，其大小为增长率

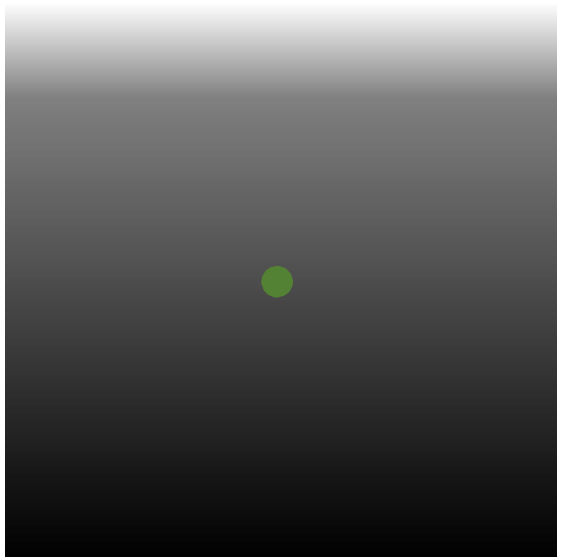




梯度

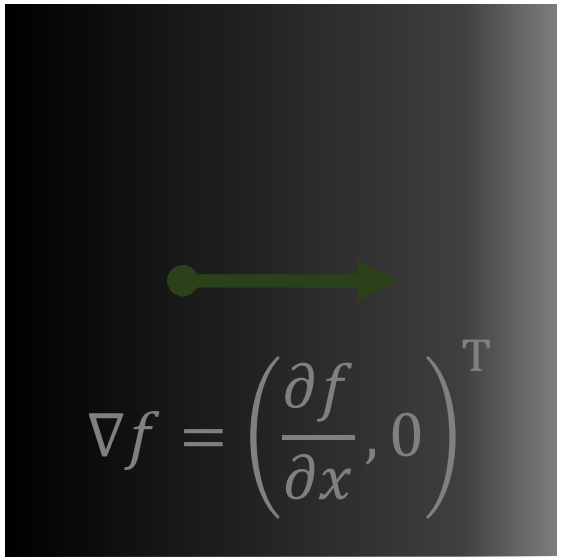
指向函数最大增长率方向的向量，其大小为增长率

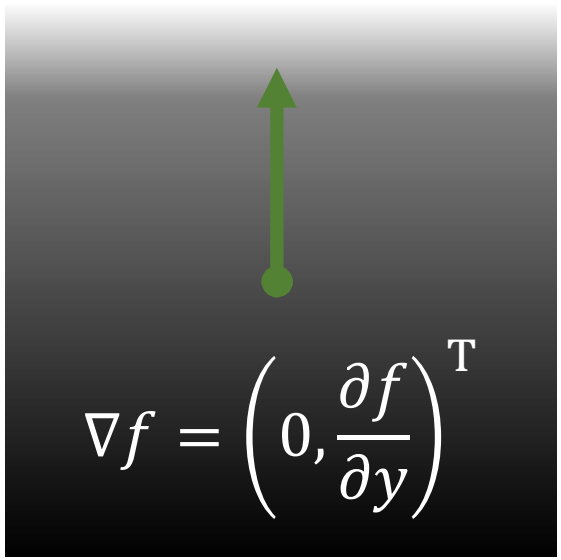




梯度

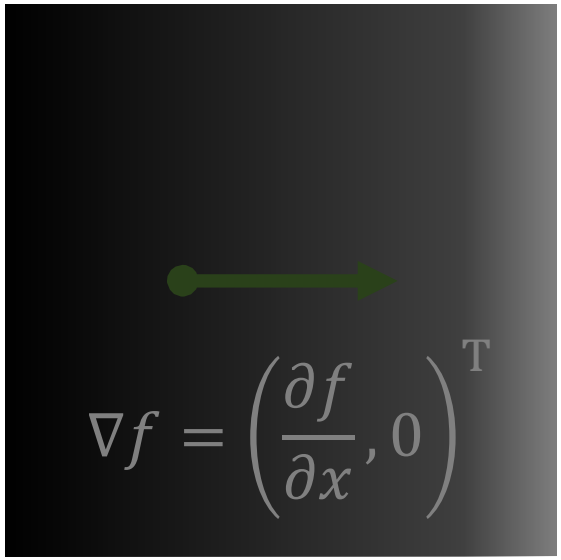
指向函数最大增长率方向的向量，其大小为增长率

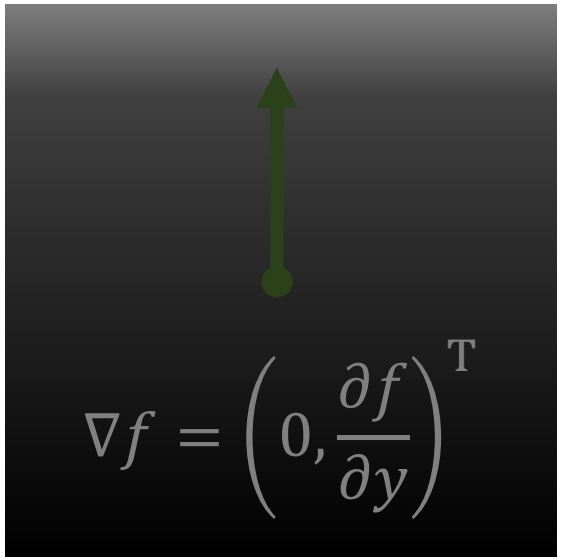


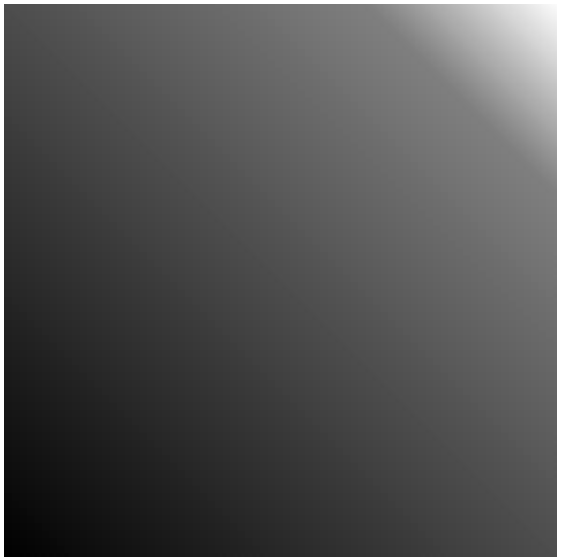


梯度

指向函数最大增长率方向的向量，其大小为增长率

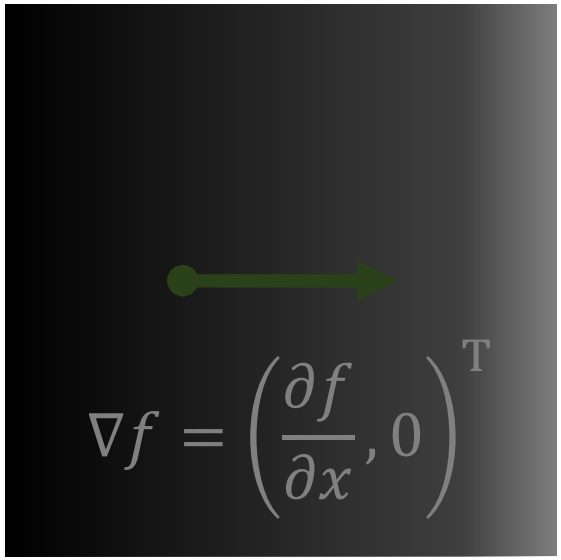


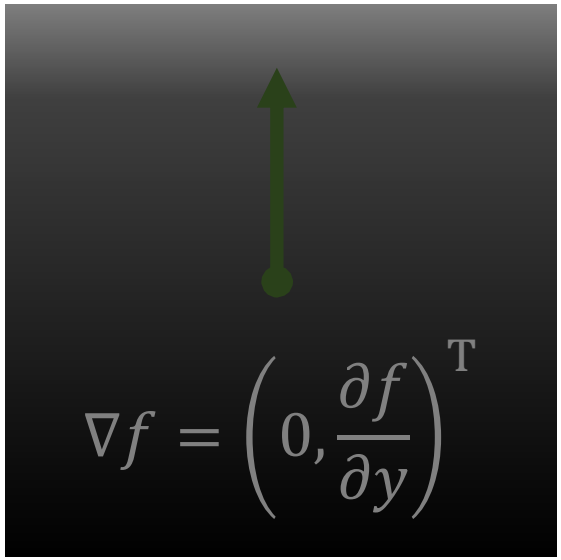


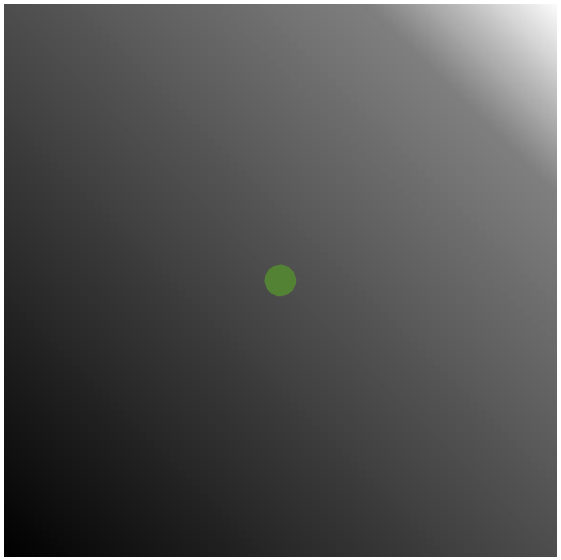


梯度

指向函数最大增长率方向的向量，其大小为增长率

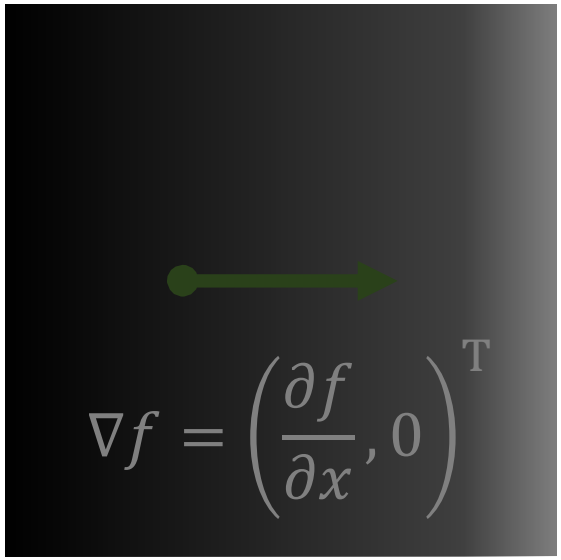


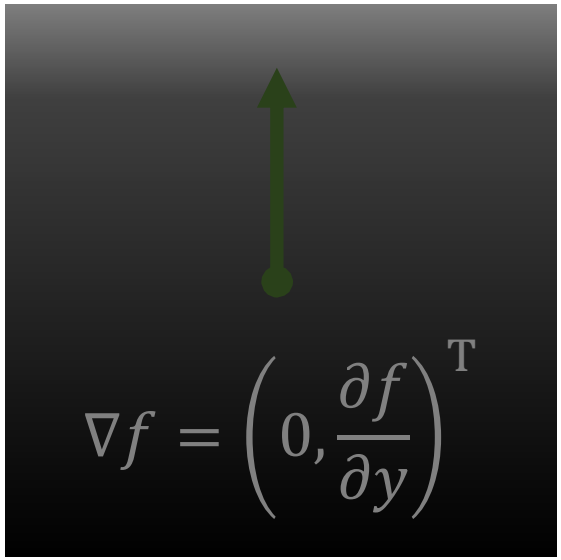


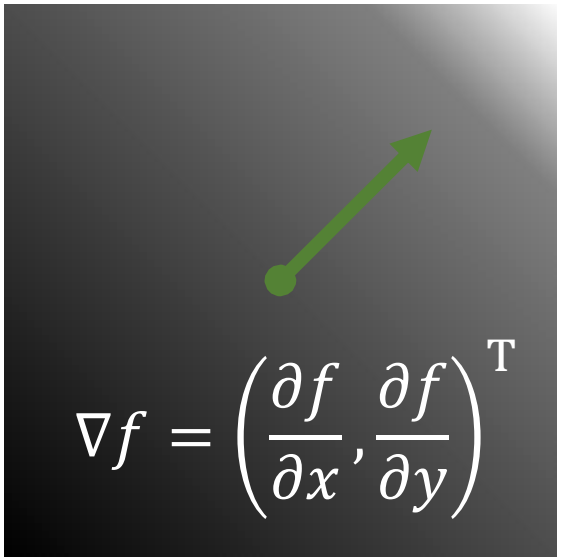


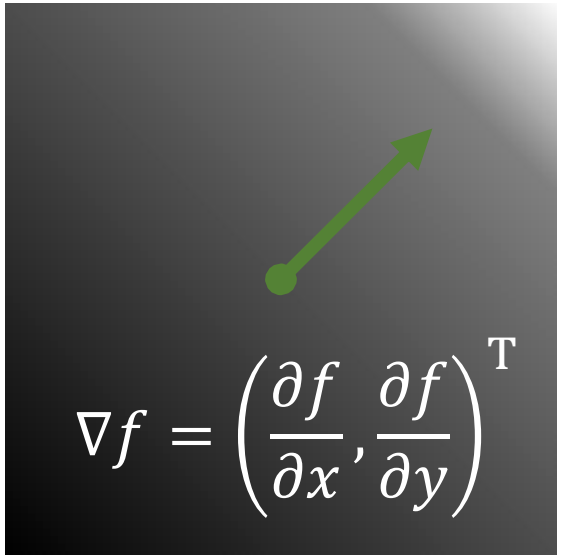
梯度

指向函数最大增长率方向的向量，其大小为增长率

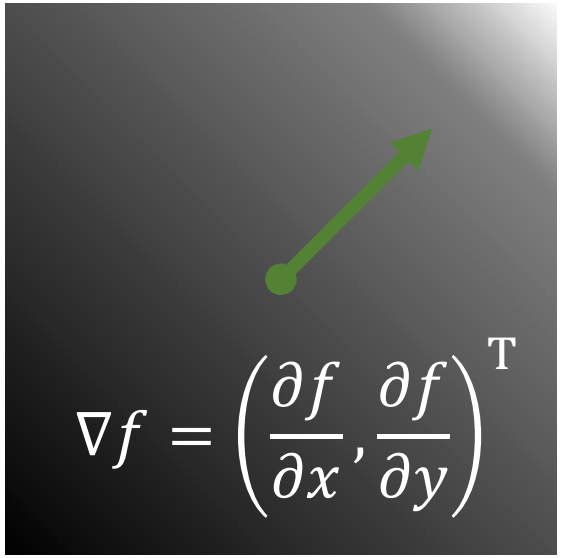




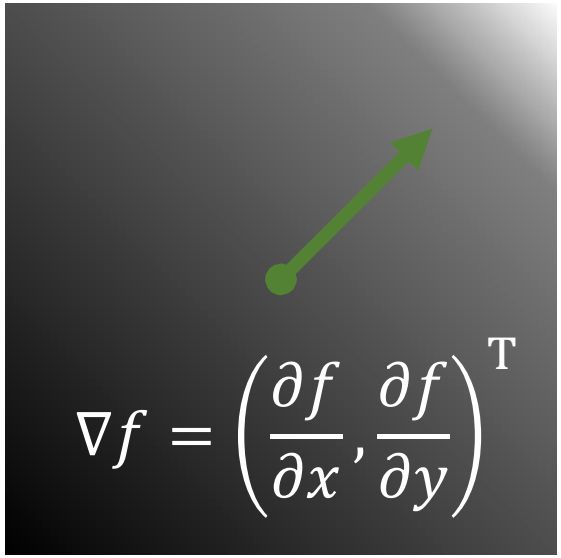



$\nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right)^{\mathrm{T}}$

梯度幅度

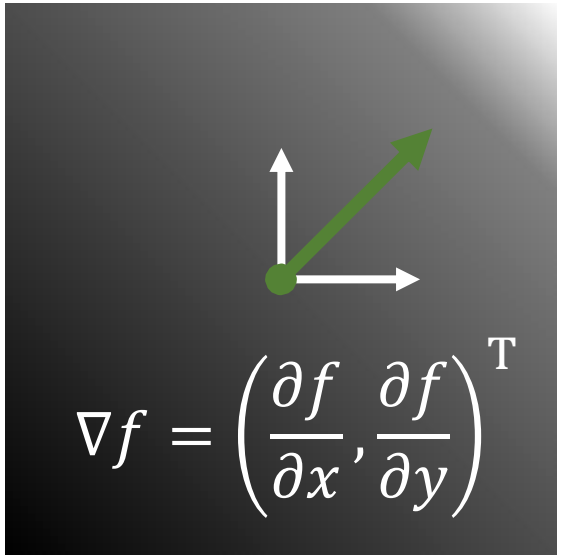


梯度幅度 $\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$



梯度幅度 $\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$

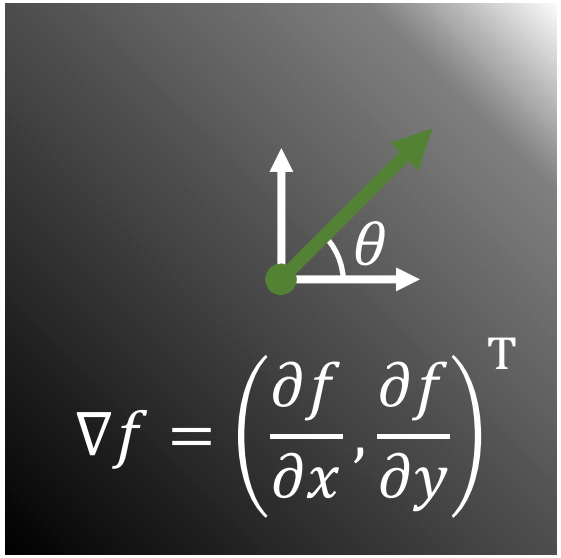
梯度方向



$$\nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right)^{\mathrm{T}}$$

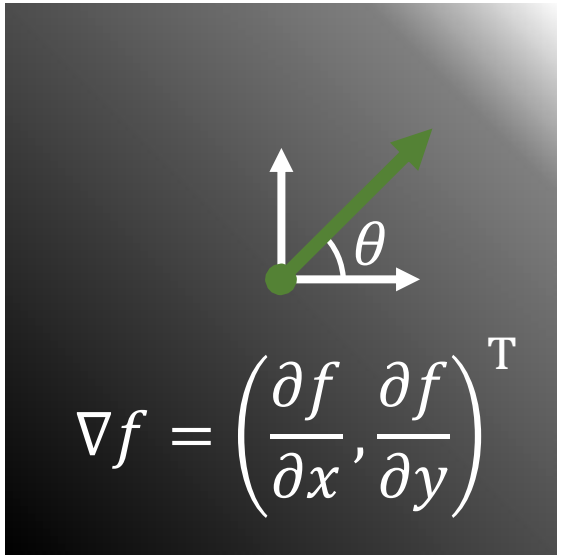
梯度幅度 $\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$

梯度方向



梯度幅度 $\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$

梯度方向

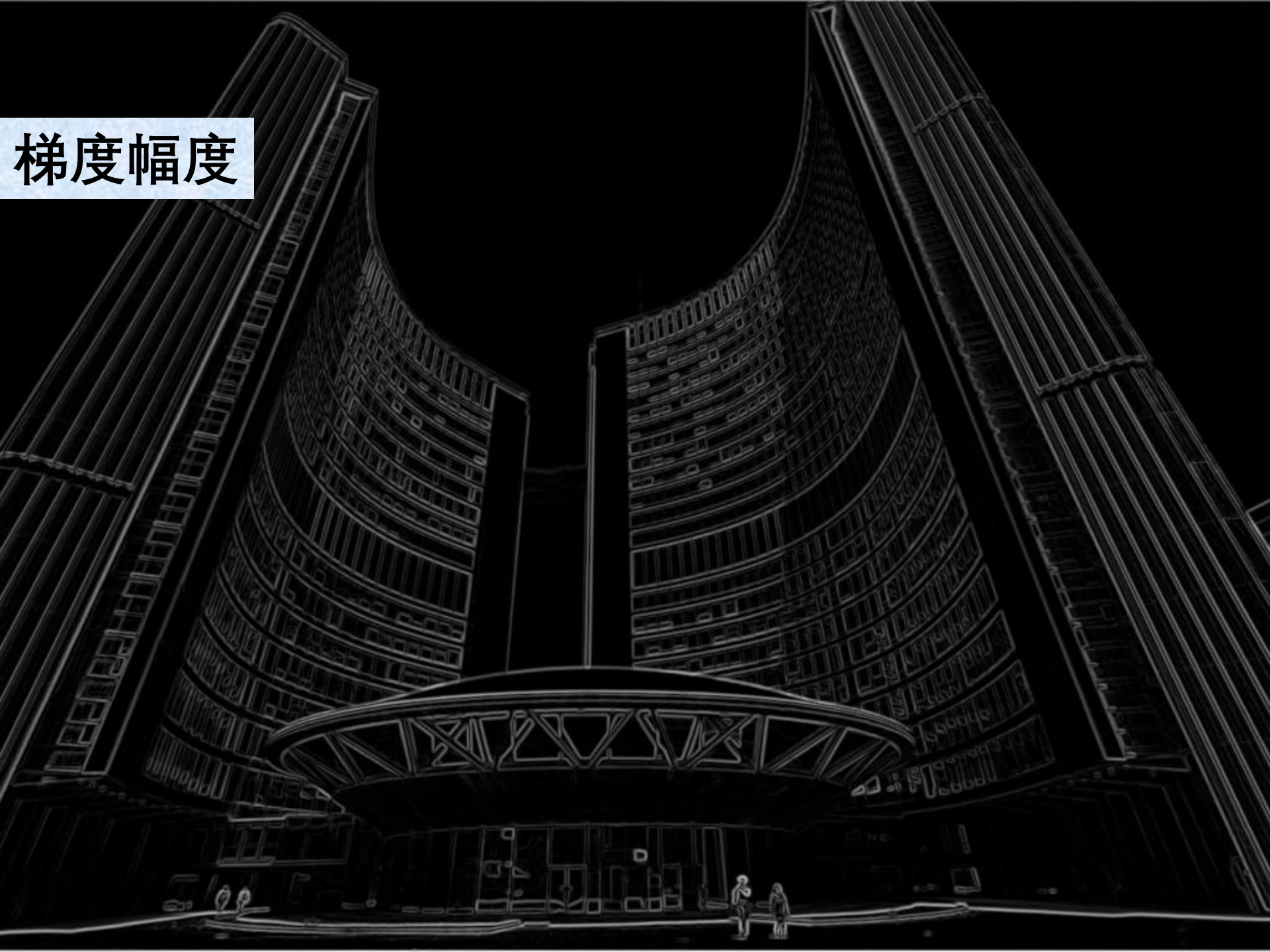


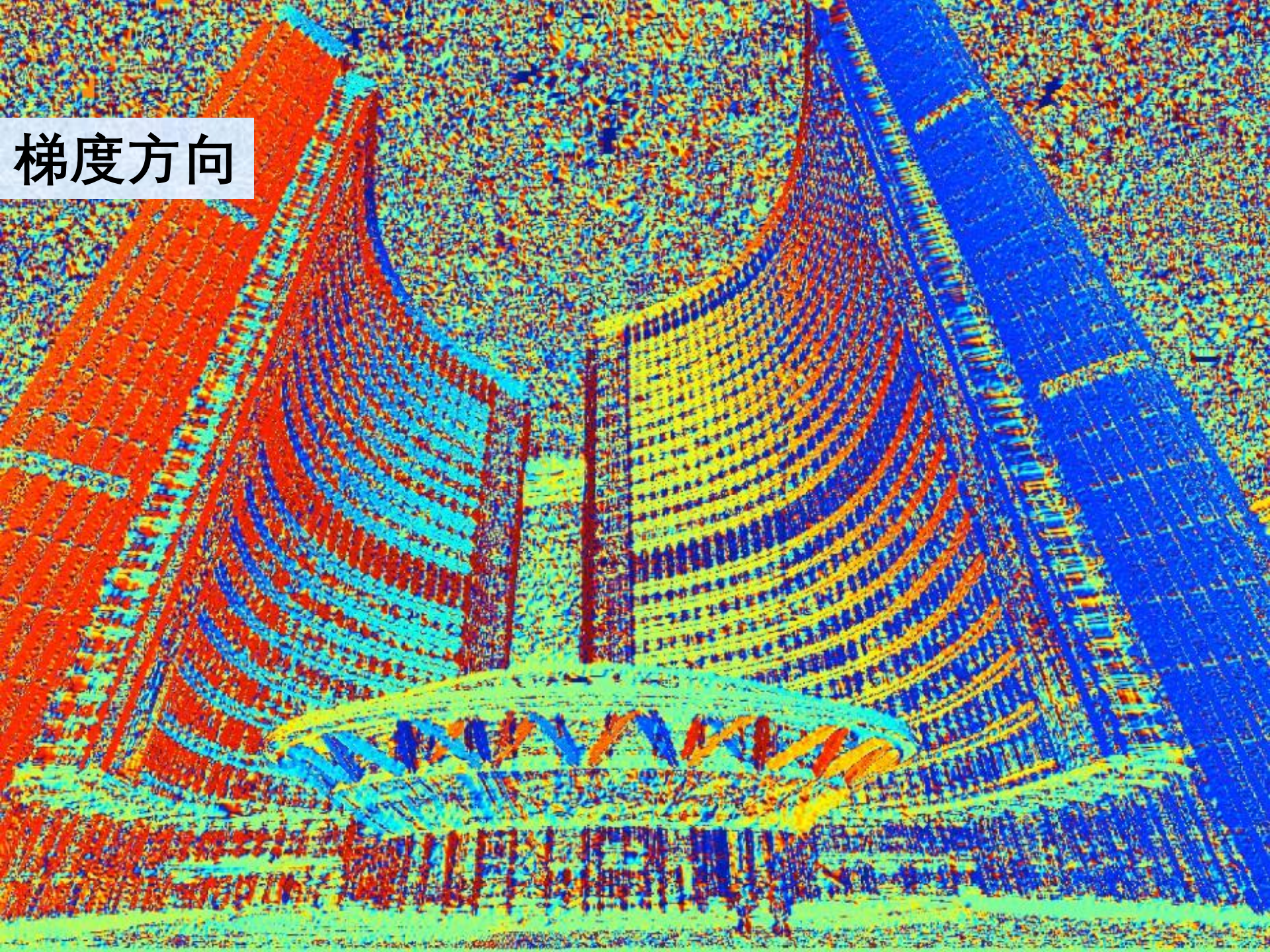
梯度幅度 $\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$

梯度方向 $\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} \Big/ \frac{\partial f}{\partial x}\right)$


原图

梯度幅度




梯度方向

Python时间

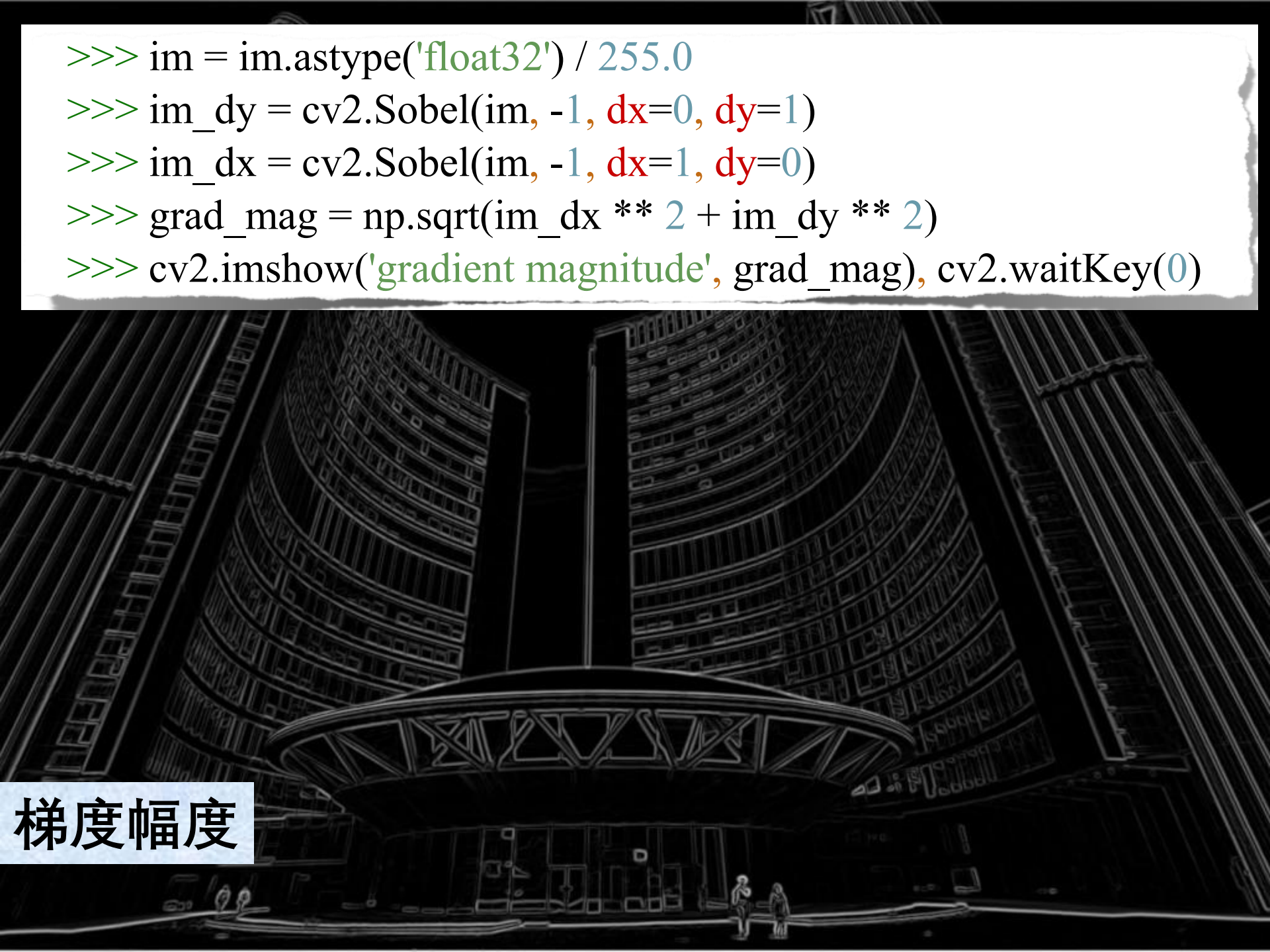
```
>>> im_dy = cv2.Sobel(im, -1, dx=0, dy=1)
>>> cv2.imshow('Sobel', im_dy)
>>> cv2.waitKey(0)
```

```
>>> im_dy = cv2.Sobel(im, -1, dx=0, dy=1)
>>> cv2.imshow('Sobel', im_dy)
>>> cv2.waitKey(0)
```

```
>>> im_dy = cv2.Sobel(im, -1, dx=0, dy=1)
>>> cv2.imshow('Sobel', im_dy)
>>> cv2.waitKey(0)
```

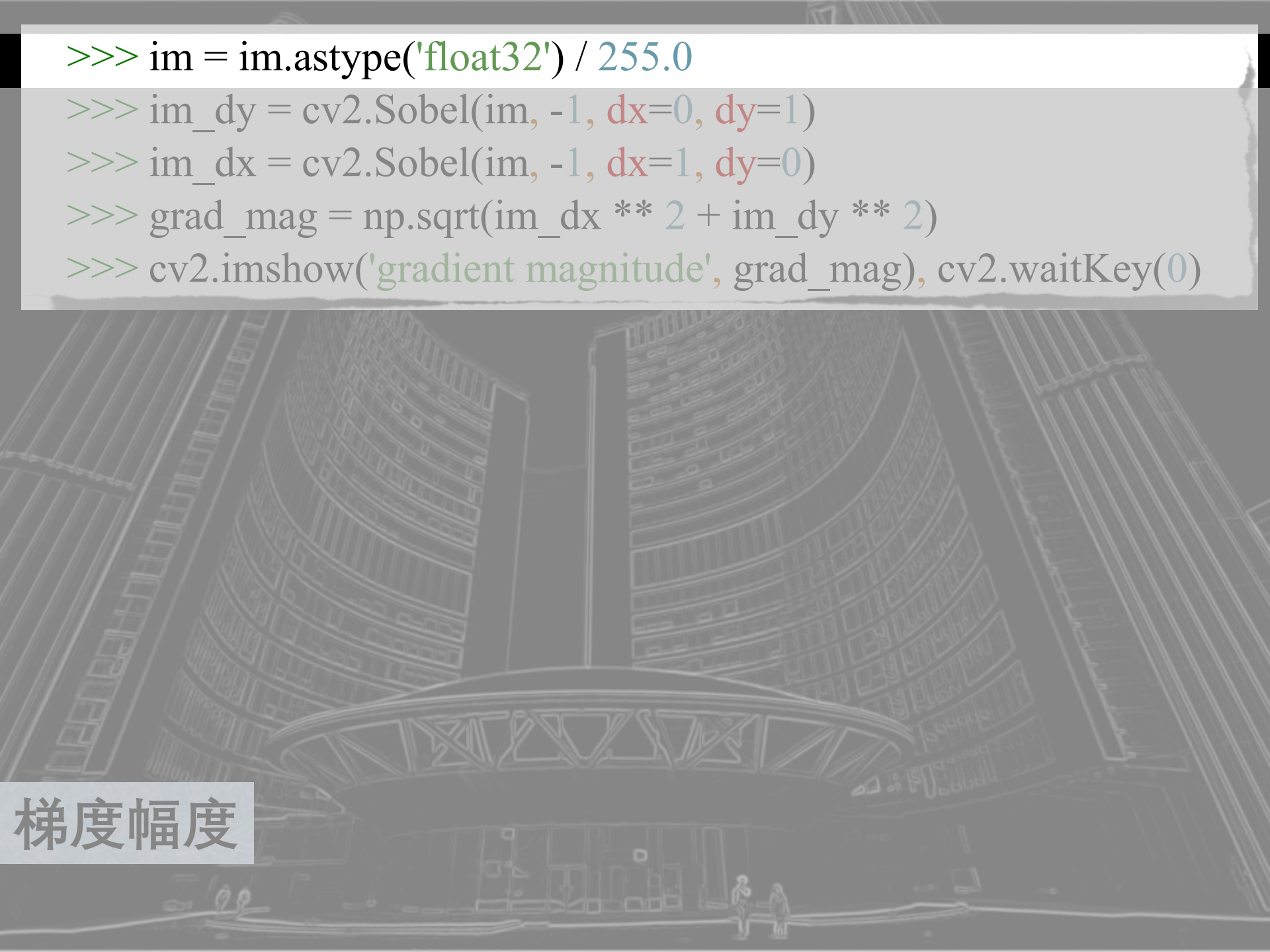
```
>>> im_dy = cv2.Sobel(im, -1, dx=0, dy=1)
>>> cv2.imshow('Sobel', im_dy)
>>> cv2.waitKey(0)
```

```
>>> im = im.astype('float32') / 255.0
>>> im_dy = cv2.Sobel(im, -1, dx=0, dy=1)
>>> im_dx = cv2.Sobel(im, -1, dx=1, dy=0)
>>> grad_mag = np.sqrt(im_dx ** 2 + im_dy ** 2)
>>> cv2.imshow('gradient magnitude', grad_mag), cv2.waitKey(0)
```



梯度幅度

```
>>> im = im.astype('float32') / 255.0
>>> im_dy = cv2.Sobel(im, -1, dx=0, dy=1)
>>> im_dx = cv2.Sobel(im, -1, dx=1, dy=0)
>>> grad_mag = np.sqrt(im_dx ** 2 + im_dy ** 2)
>>> cv2.imshow('gradient magnitude', grad_mag), cv2.waitKey(0)
```

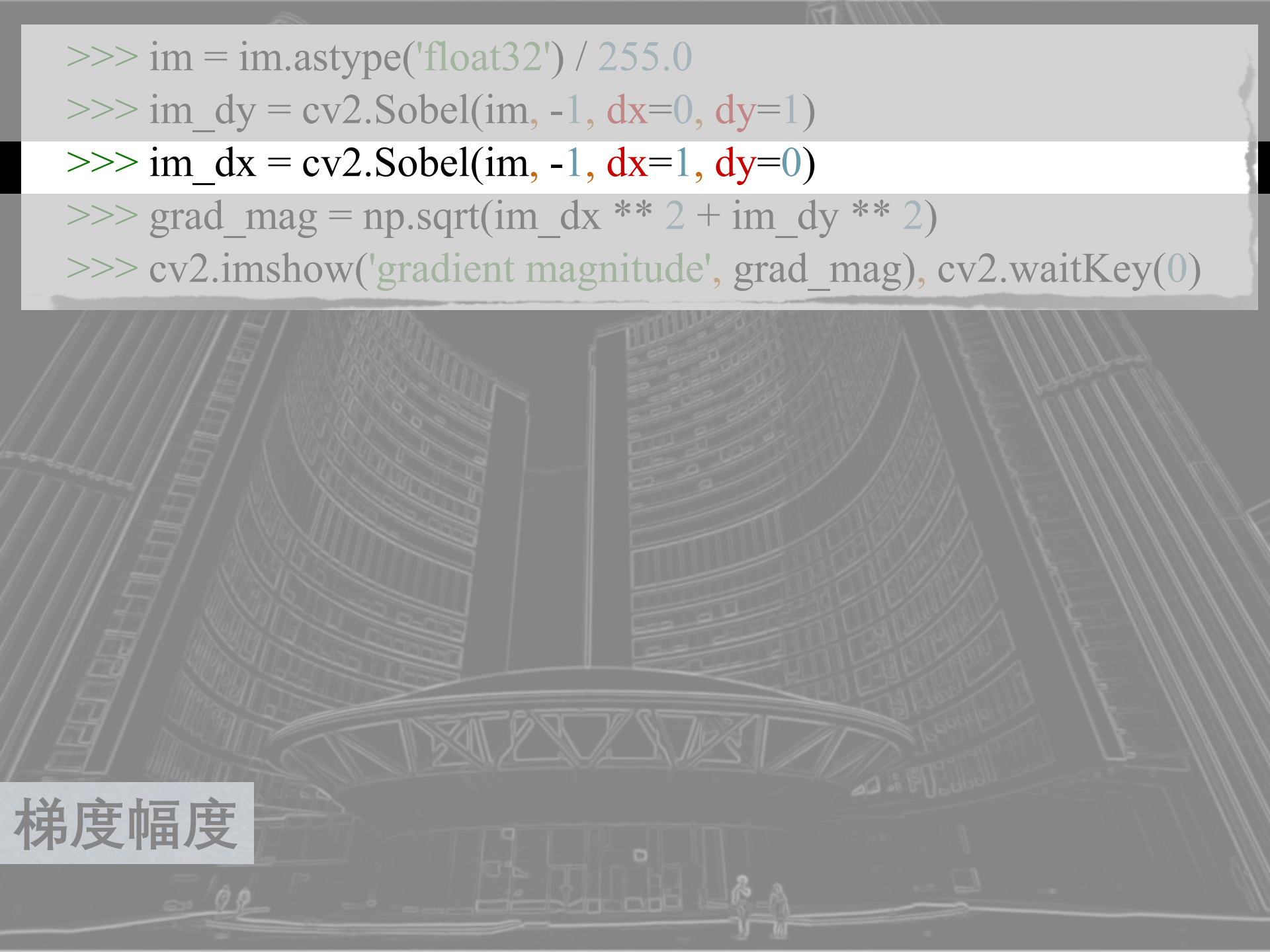


梯度幅度

```
>>> im = im.astype('float32') / 255.0
>>> im_dy = cv2.Sobel(im, -1, dx=0, dy=1)
>>> im_dx = cv2.Sobel(im, -1, dx=1, dy=0)
>>> grad_mag = np.sqrt(im_dx ** 2 + im_dy ** 2)
>>> cv2.imshow('gradient magnitude', grad_mag), cv2.waitKey(0)
```

梯度幅度

```
>>> im = im.astype('float32') / 255.0
>>> im_dy = cv2.Sobel(im, -1, dx=0, dy=1)
>>> im_dx = cv2.Sobel(im, -1, dx=1, dy=0)
>>> grad_mag = np.sqrt(im_dx ** 2 + im_dy ** 2)
>>> cv2.imshow('gradient magnitude', grad_mag), cv2.waitKey(0)
```



梯度幅度

```
>>> im = im.astype('float32') / 255.0
>>> im_dy = cv2.Sobel(im, -1, dx=0, dy=1)
>>> im_dx = cv2.Sobel(im, -1, dx=1, dy=0)
>>> grad_mag = np.sqrt(im_dx ** 2 + im_dy ** 2)
>>> cv2.imshow('gradient magnitude', grad_mag), cv2.waitKey(0)
```

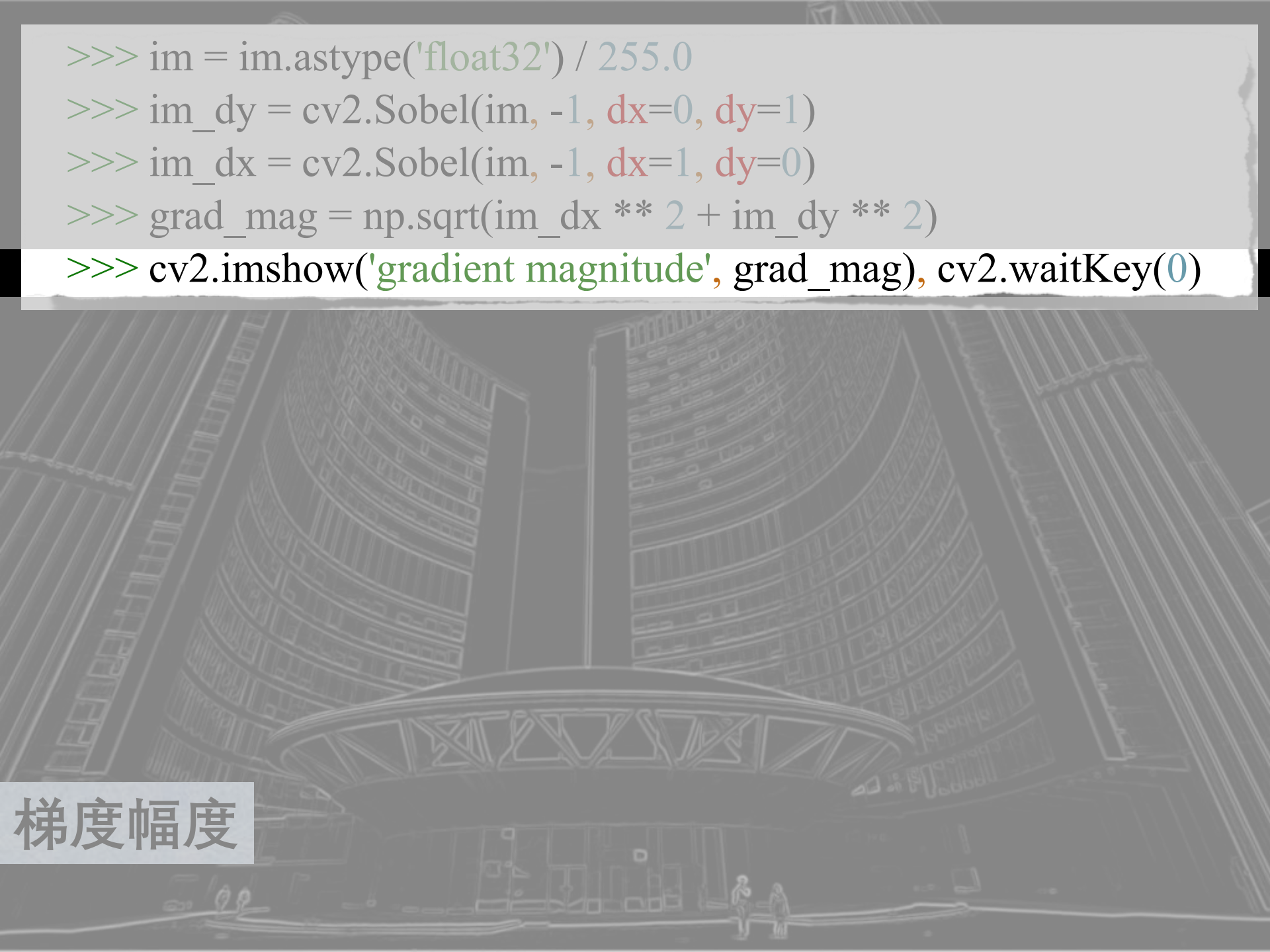
梯度幅度

```
>>> im = im.astype('float32') / 255.0
>>> im_dy = cv2.Sobel(im, -1, dx=0, dy=1)
>>> im_dx = cv2.Sobel(im, -1, dx=1, dy=0)
>>> grad_mag = np.sqrt(im_dx ** 2 + im_dy ** 2)
>>> cv2.imshow('gradient magnitude', grad_mag), cv2.waitKey(0)
```

幂运算

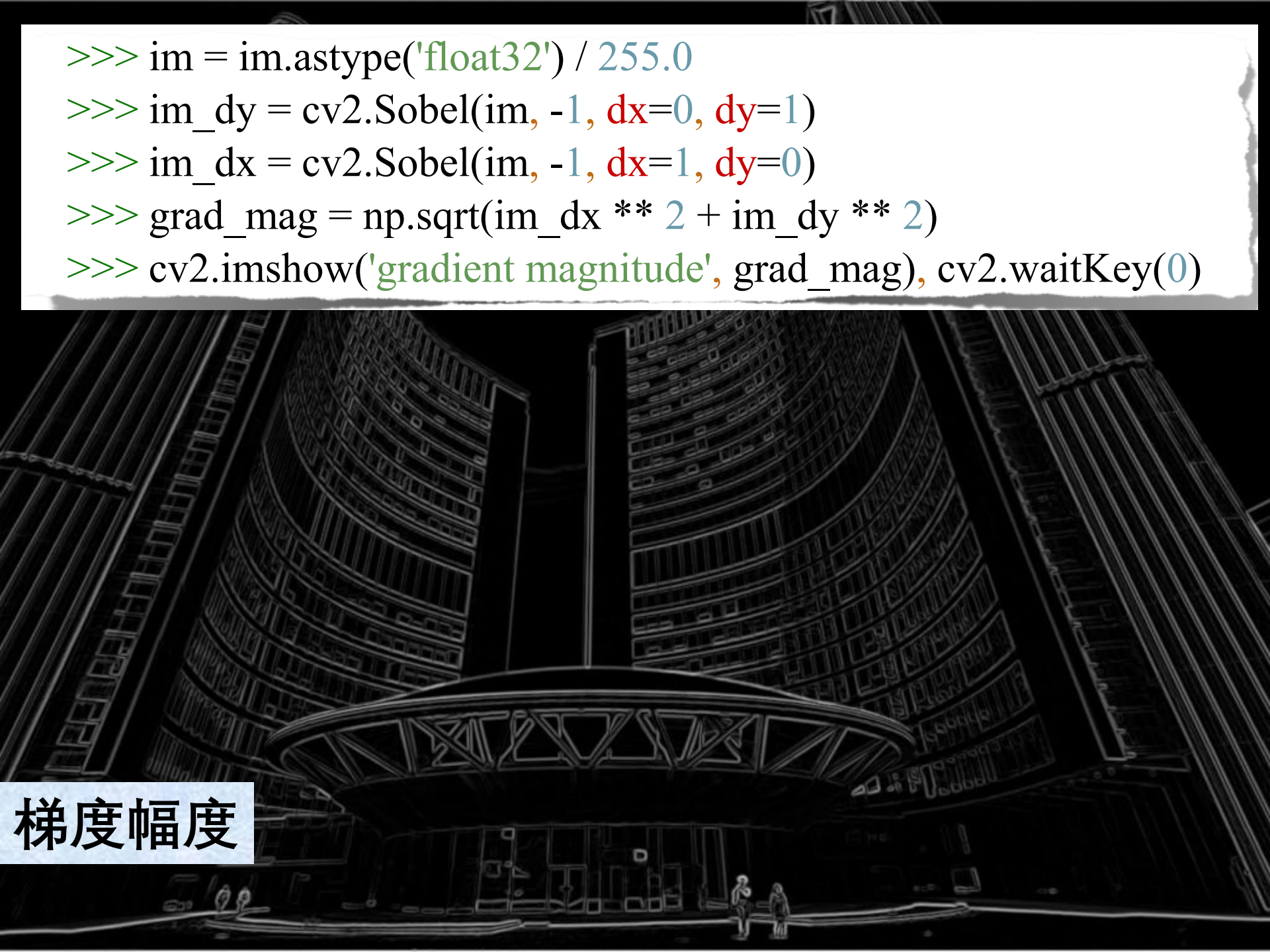
梯度幅度

```
>>> im = im.astype('float32') / 255.0
>>> im_dy = cv2.Sobel(im, -1, dx=0, dy=1)
>>> im_dx = cv2.Sobel(im, -1, dx=1, dy=0)
>>> grad_mag = np.sqrt(im_dx ** 2 + im_dy ** 2)
>>> cv2.imshow('gradient magnitude', grad_mag), cv2.waitKey(0)
```



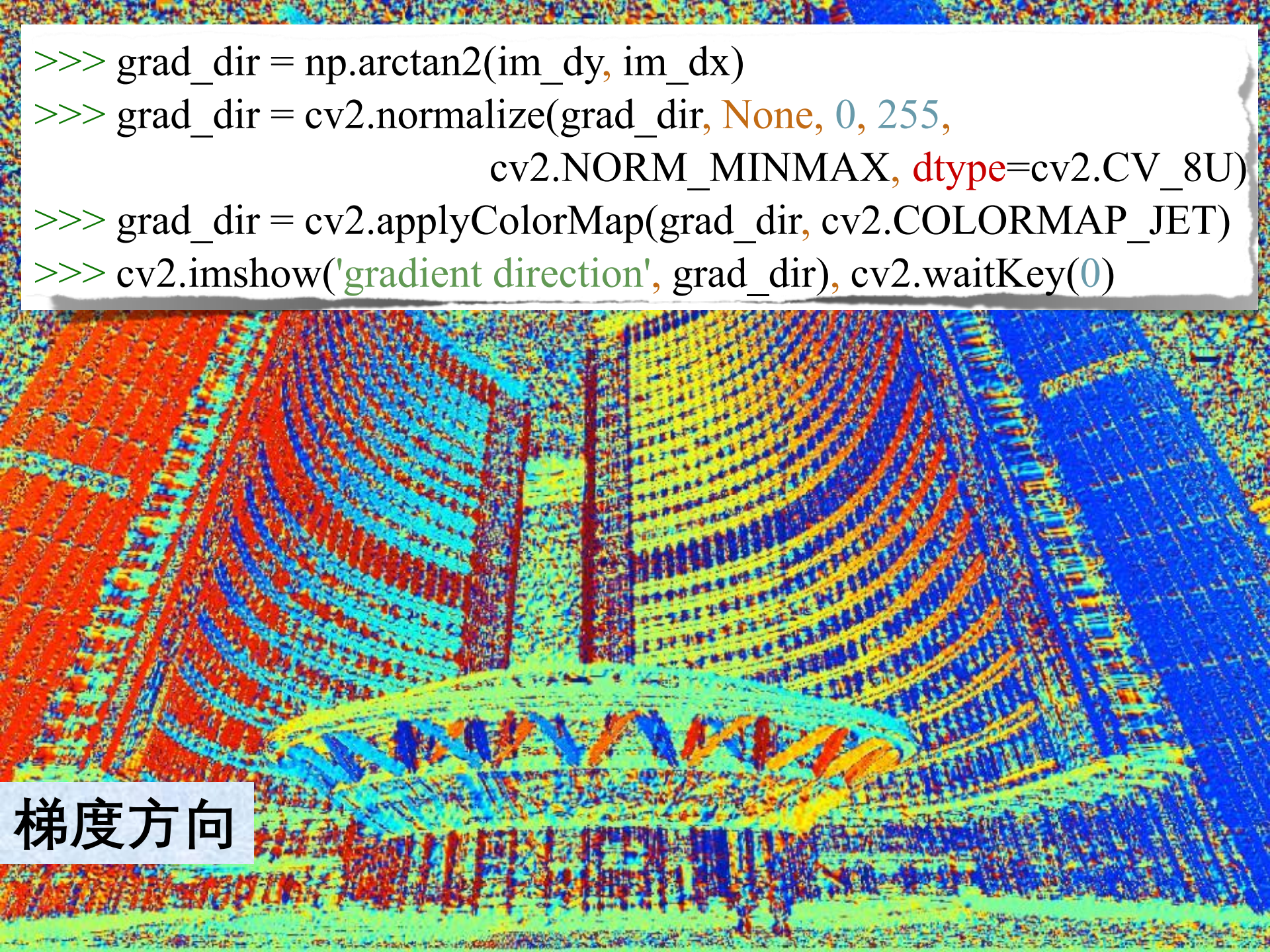
梯度幅度

```
>>> im = im.astype('float32') / 255.0
>>> im_dy = cv2.Sobel(im, -1, dx=0, dy=1)
>>> im_dx = cv2.Sobel(im, -1, dx=1, dy=0)
>>> grad_mag = np.sqrt(im_dx ** 2 + im_dy ** 2)
>>> cv2.imshow('gradient magnitude', grad_mag), cv2.waitKey(0)
```

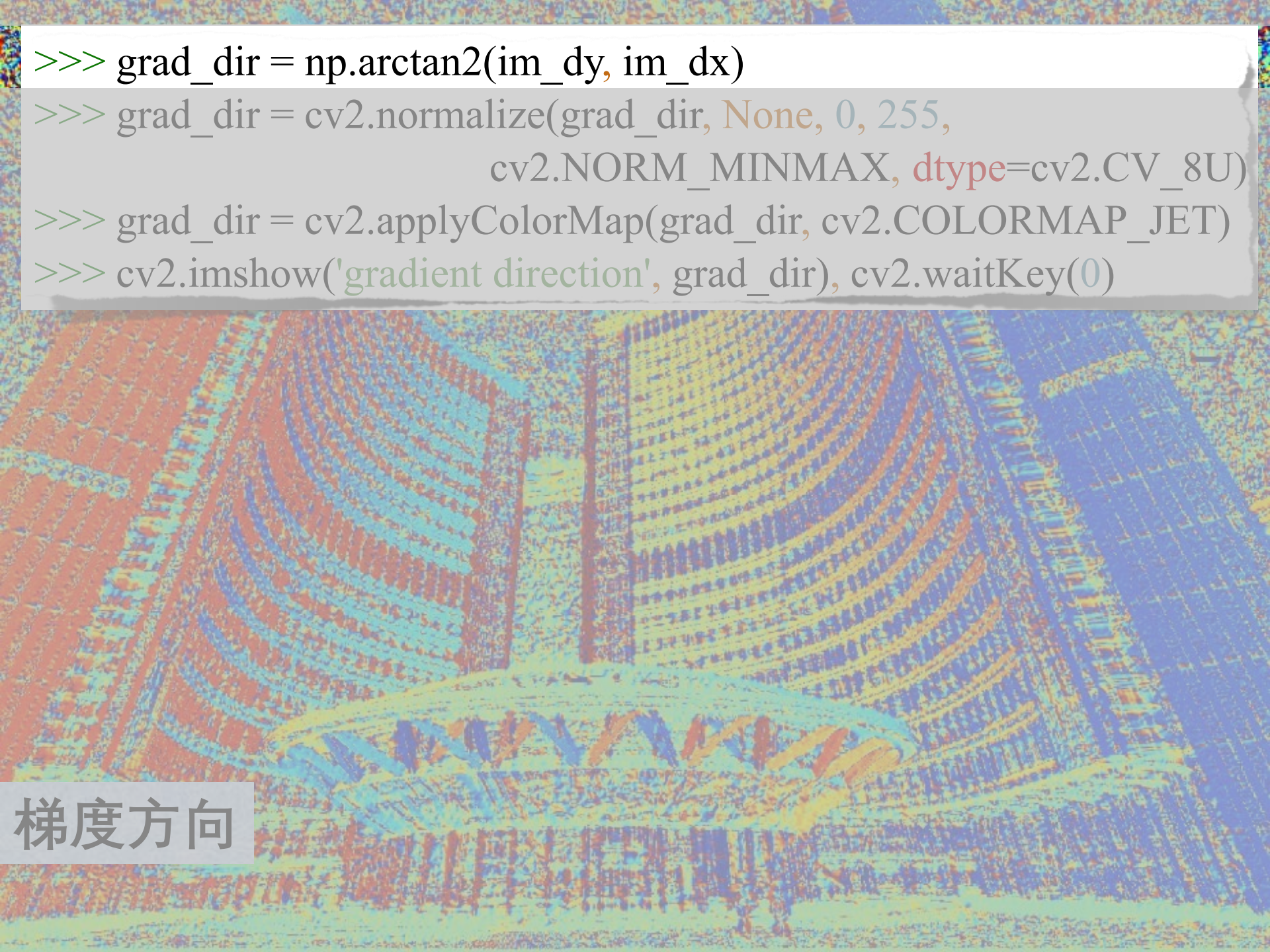


梯度幅度

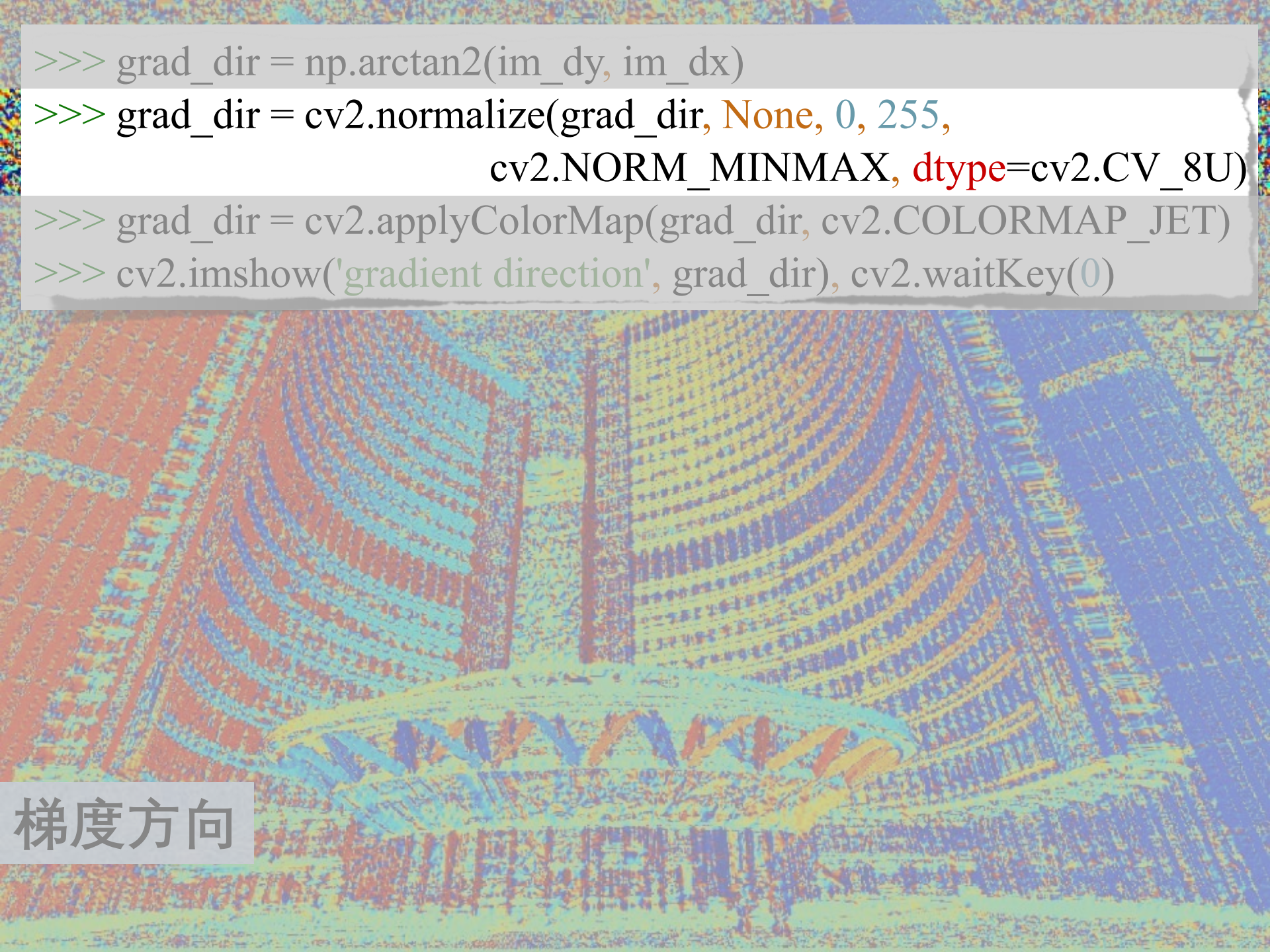
```
>>> grad_dir = np.arctan2(im_dy, im_dx)
>>> grad_dir = cv2.normalize(grad_dir, None, 0, 255,
                             cv2.NORM_MINMAX, dtype=cv2.CV_8U)
>>> grad_dir = cv2.applyColorMap(grad_dir, cv2.COLORMAP_JET)
>>> cv2.imshow('gradient direction', grad_dir), cv2.waitKey(0)
```



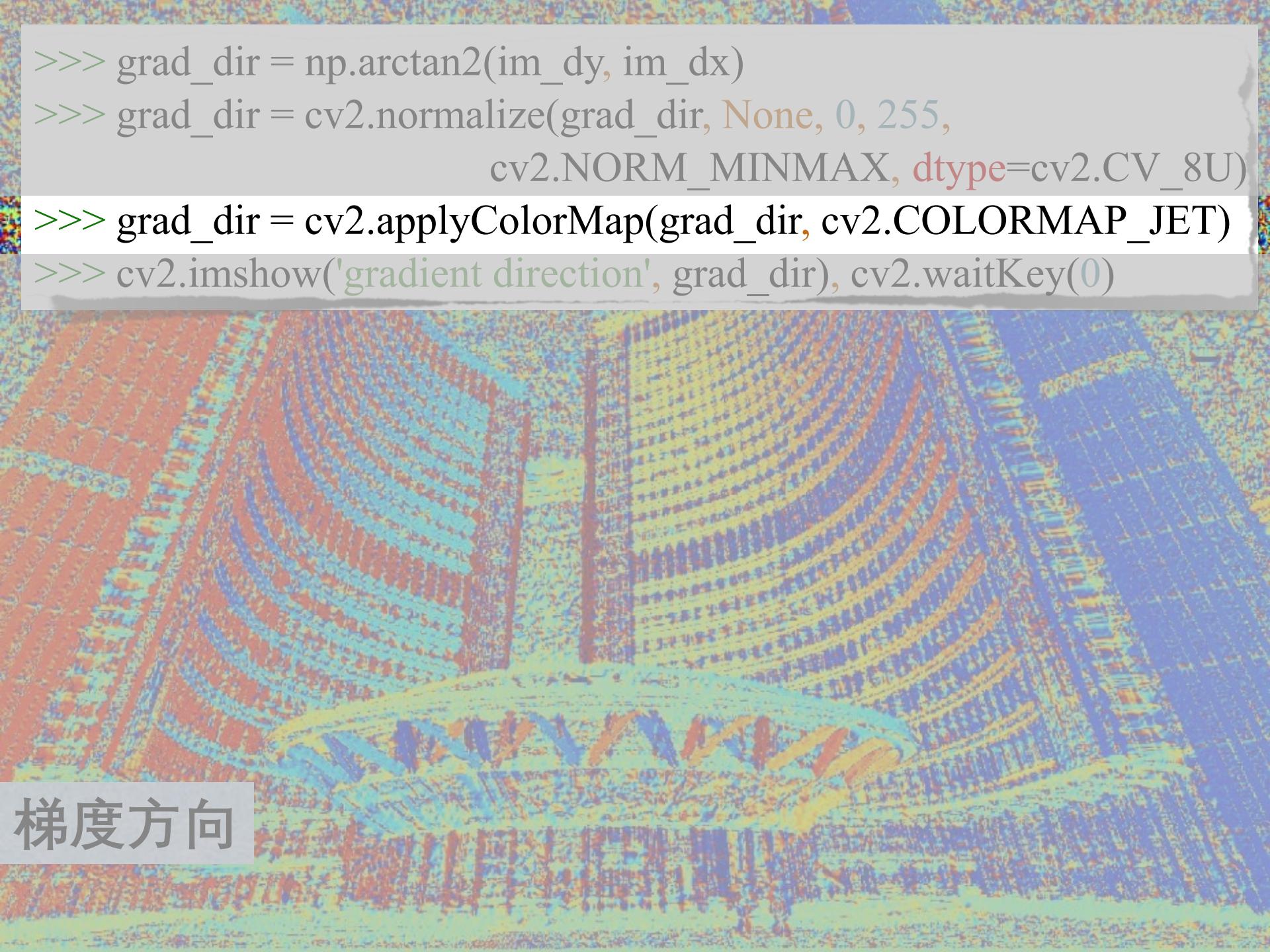
```
>>> grad_dir = np.arctan2(im_dy, im_dx)
>>> grad_dir = cv2.normalize(grad_dir, None, 0, 255,
                             cv2.NORM_MINMAX, dtype=cv2.CV_8U)
>>> grad_dir = cv2.applyColorMap(grad_dir, cv2.COLORMAP_JET)
>>> cv2.imshow('gradient direction', grad_dir), cv2.waitKey(0)
```



```
>>> grad_dir = np.arctan2(im_dy, im_dx)
>>> grad_dir = cv2.normalize(grad_dir, None, 0, 255,
                             cv2.NORM_MINMAX, dtype=cv2.CV_8U)
>>> grad_dir = cv2.applyColorMap(grad_dir, cv2.COLORMAP_JET)
>>> cv2.imshow('gradient direction', grad_dir), cv2.waitKey(0)
```

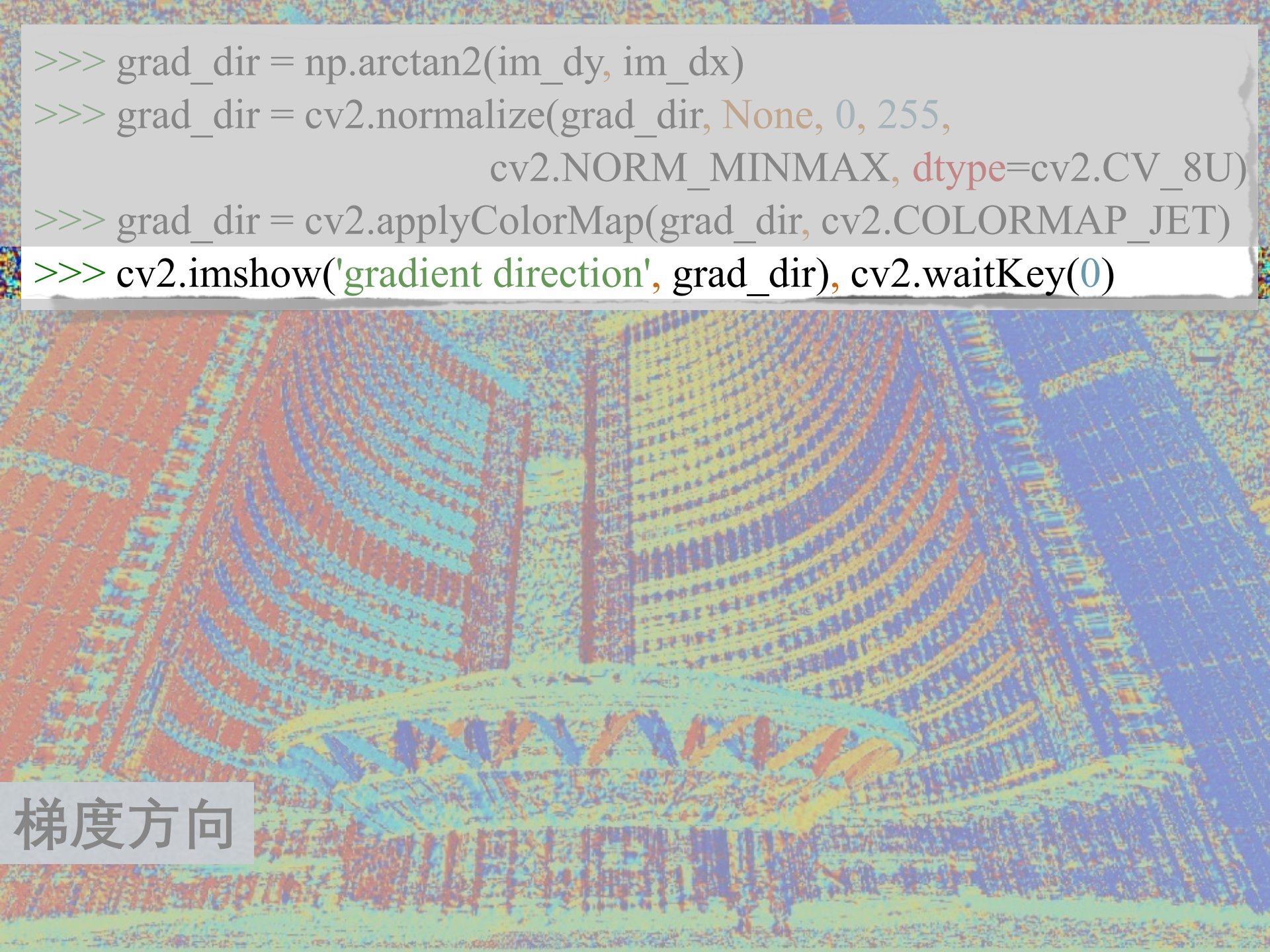


```
>>> grad_dir = np.arctan2(im_dy, im_dx)
>>> grad_dir = cv2.normalize(grad_dir, None, 0, 255,
                             cv2.NORM_MINMAX, dtype=cv2.CV_8U)
>>> grad_dir = cv2.applyColorMap(grad_dir, cv2.COLORMAP_JET)
>>> cv2.imshow('gradient direction', grad_dir), cv2.waitKey(0)
```



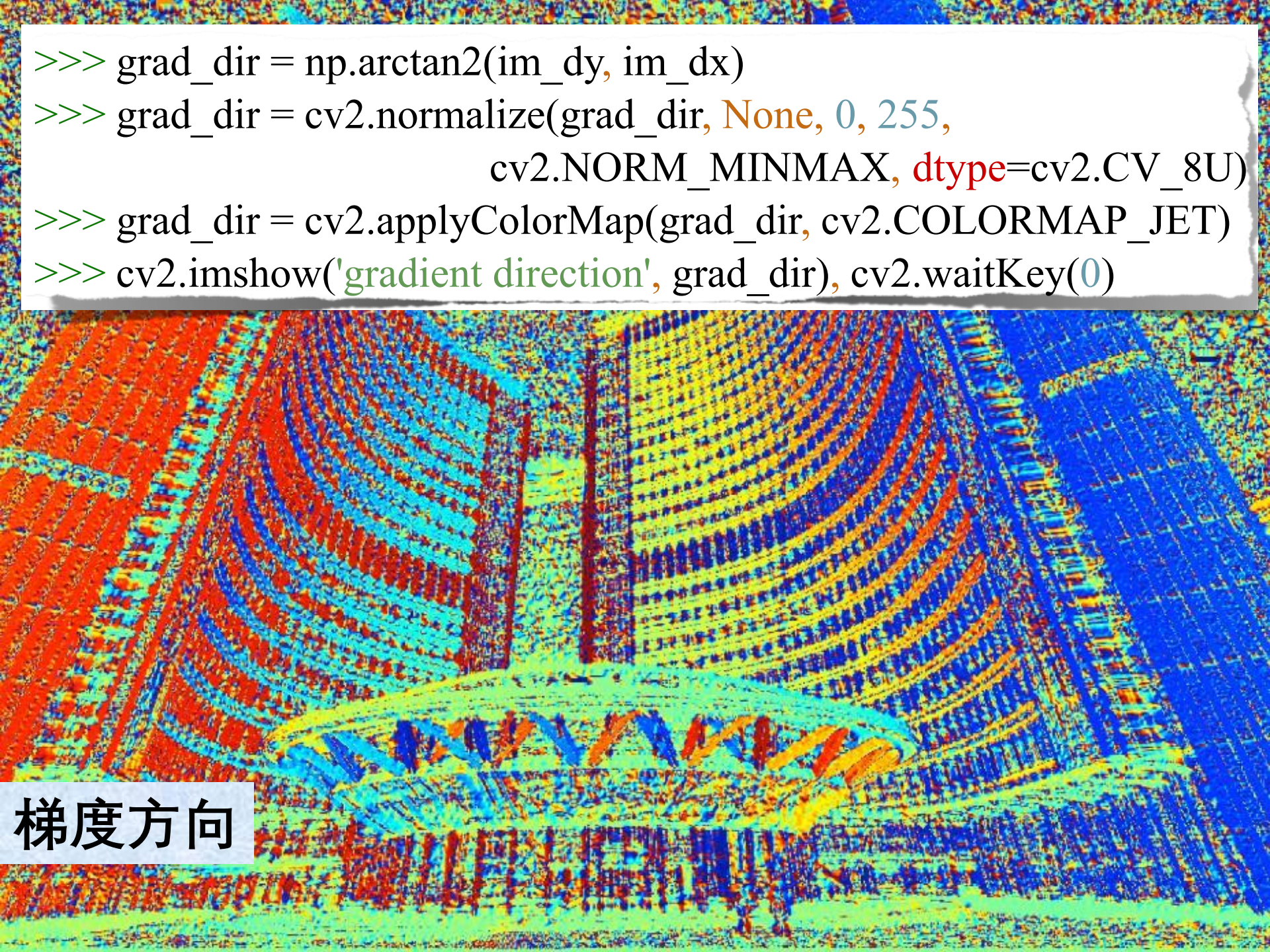
梯度方向

```
>>> grad_dir = np.arctan2(im_dy, im_dx)
>>> grad_dir = cv2.normalize(grad_dir, None, 0, 255,
                             cv2.NORM_MINMAX, dtype=cv2.CV_8U)
>>> grad_dir = cv2.applyColorMap(grad_dir, cv2.COLORMAP_JET)
>>> cv2.imshow('gradient direction', grad_dir), cv2.waitKey(0)
```

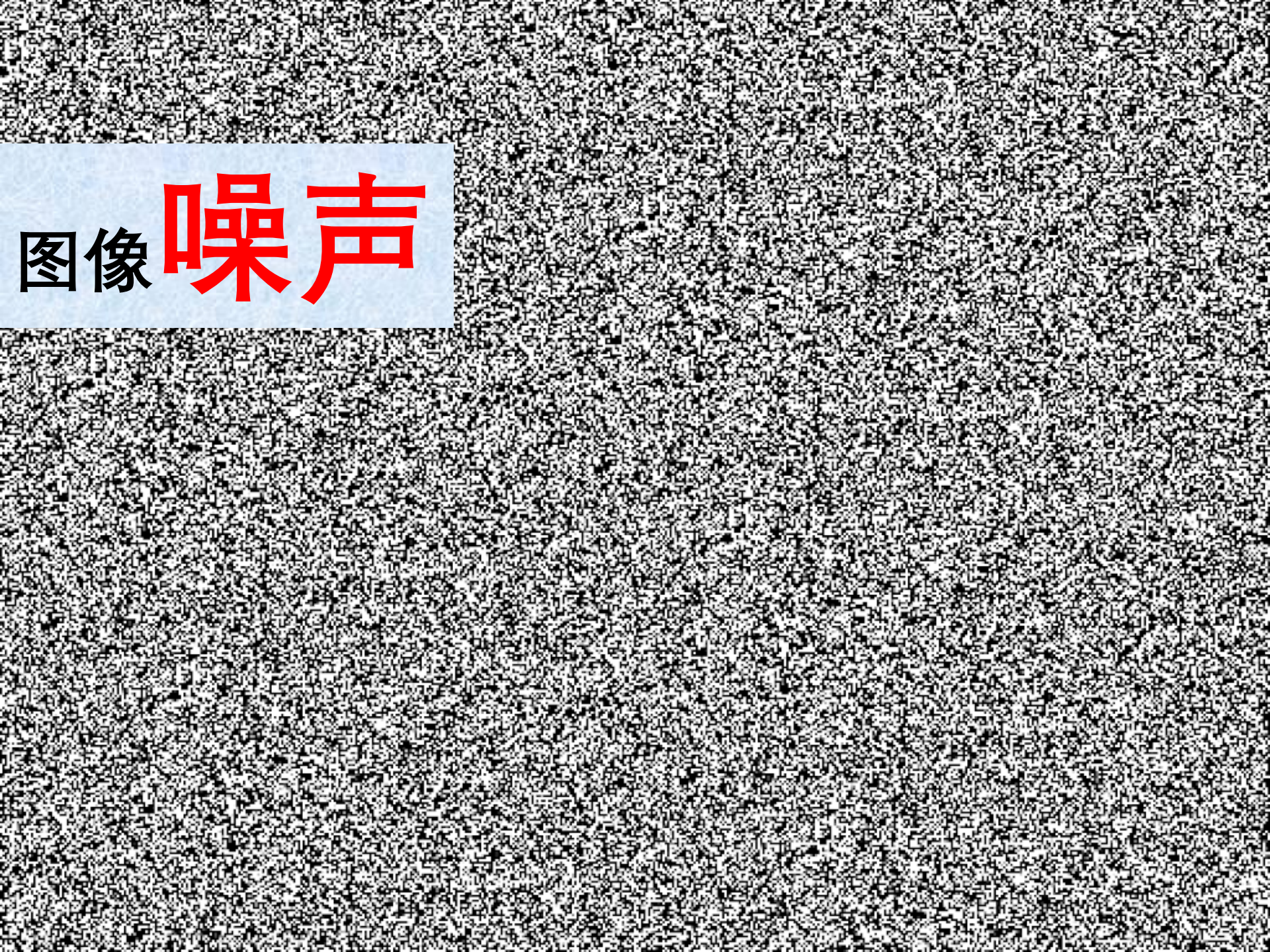


梯度方向

```
>>> grad_dir = np.arctan2(im_dy, im_dx)
>>> grad_dir = cv2.normalize(grad_dir, None, 0, 255,
                             cv2.NORM_MINMAX, dtype=cv2.CV_8U)
>>> grad_dir = cv2.applyColorMap(grad_dir, cv2.COLORMAP_JET)
>>> cv2.imshow('gradient direction', grad_dir), cv2.waitKey(0)
```

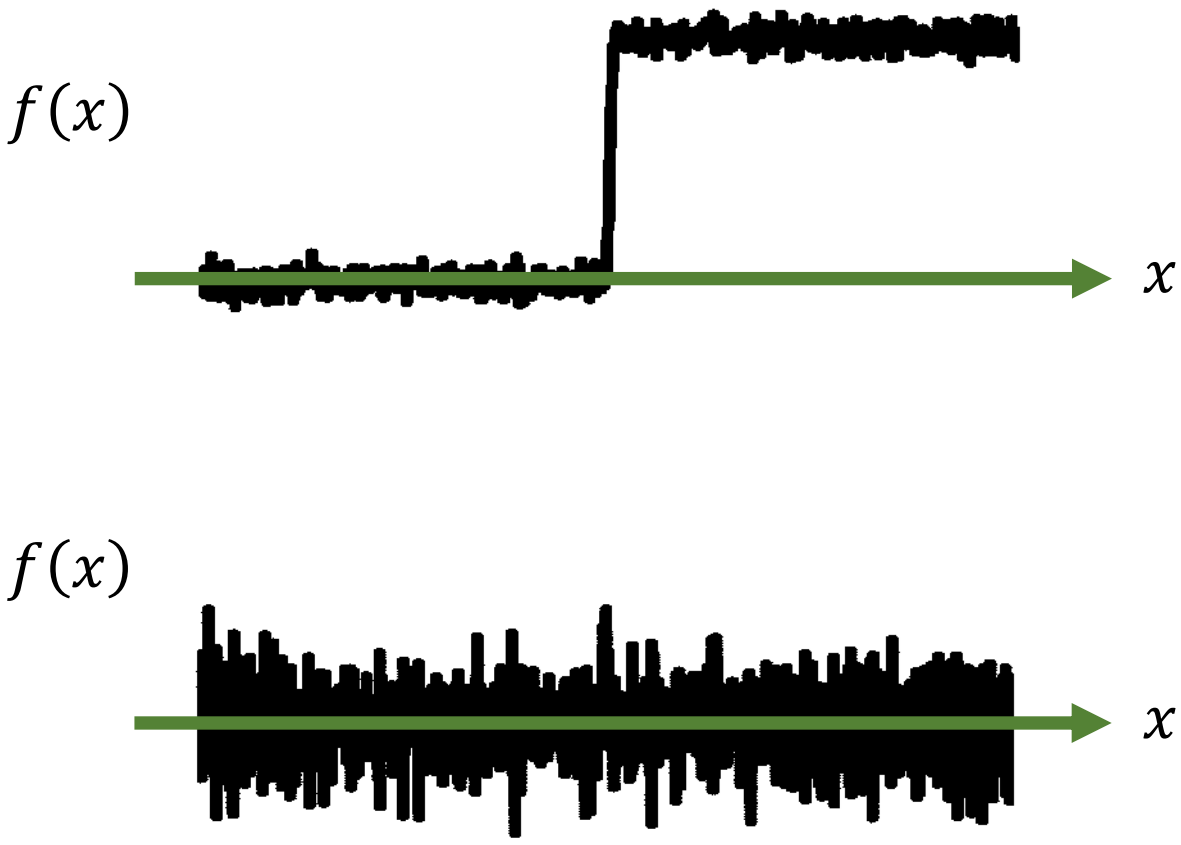


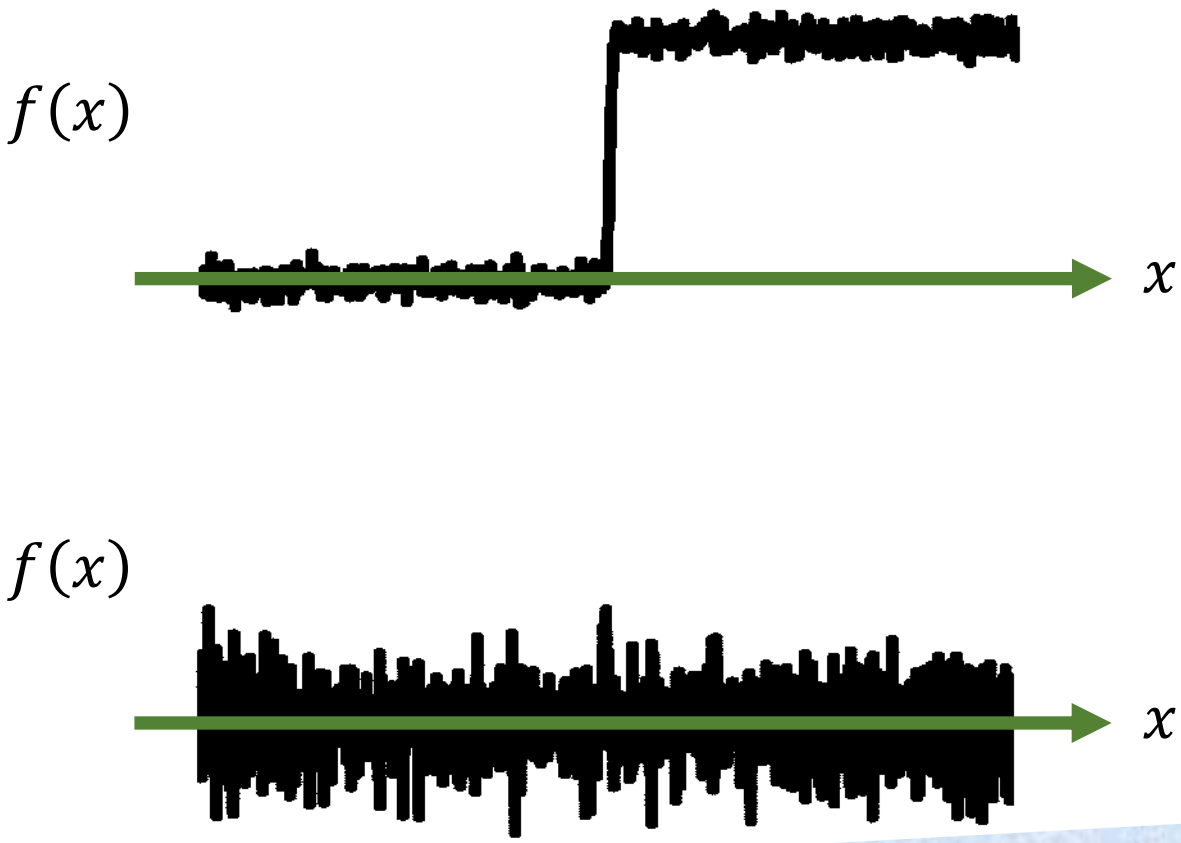
Python时间


图像噪声


$f(x)$
x

$$\frac{d}{dx} f(x)$$


$f(x)$
x
$\frac{d}{dx}f(x)$
x



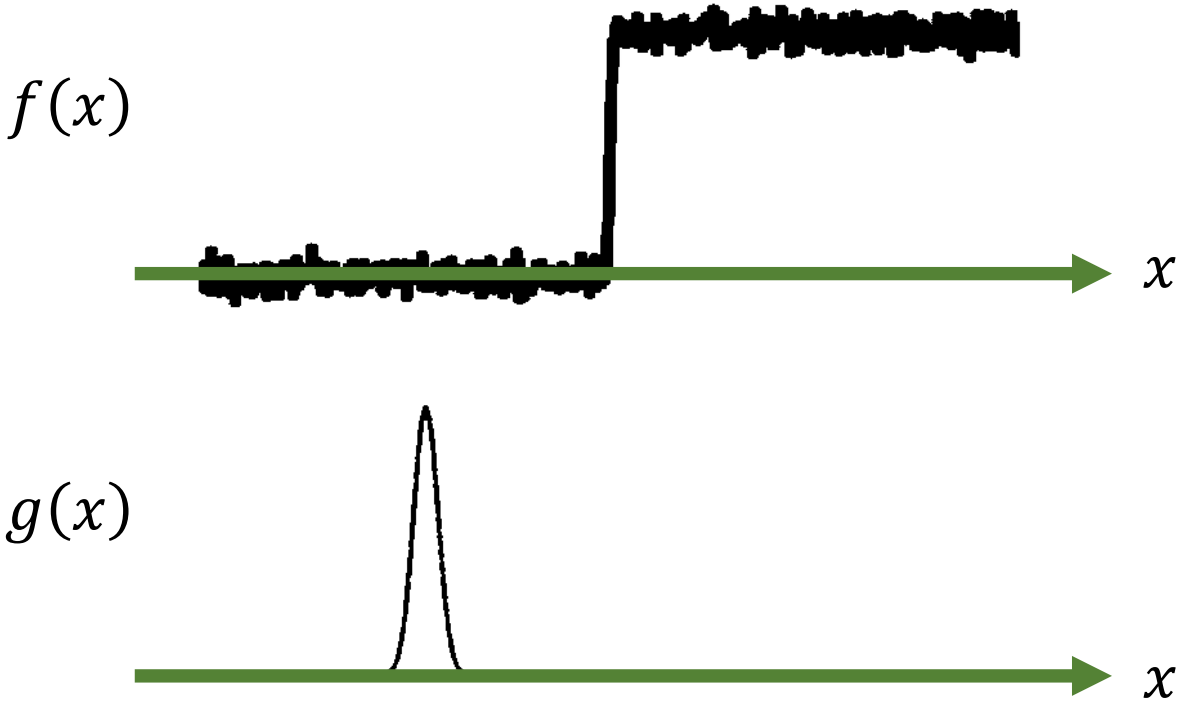
噪声被放大了！

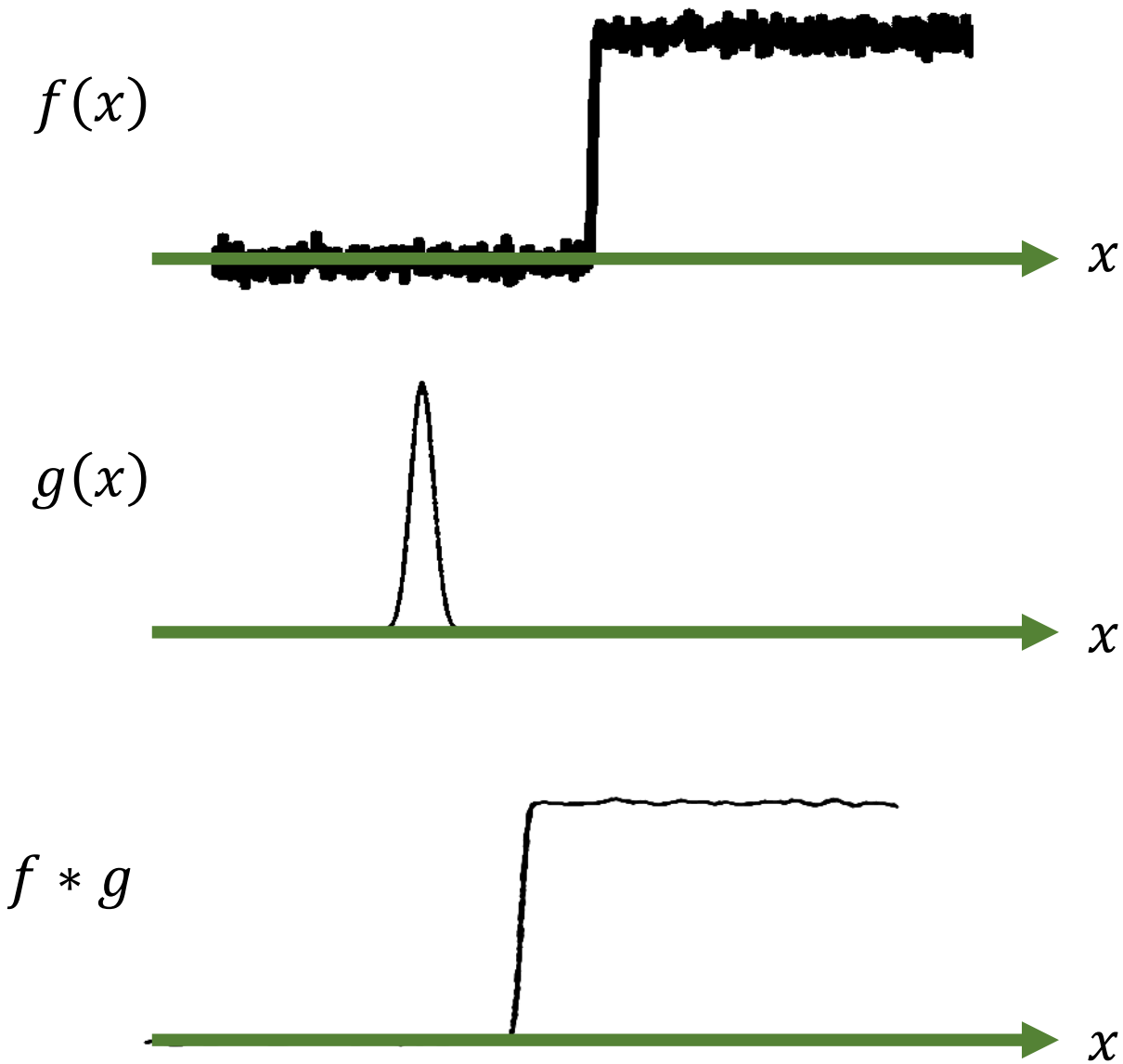

$f(x)$
x

如何避免噪声带来的问题?

如何避免噪声带来的问题？

滤波降噪


$f(x)$
x
$g(x)$
x


$f(x)$
x
$g(x)$
x
$f * g$
x

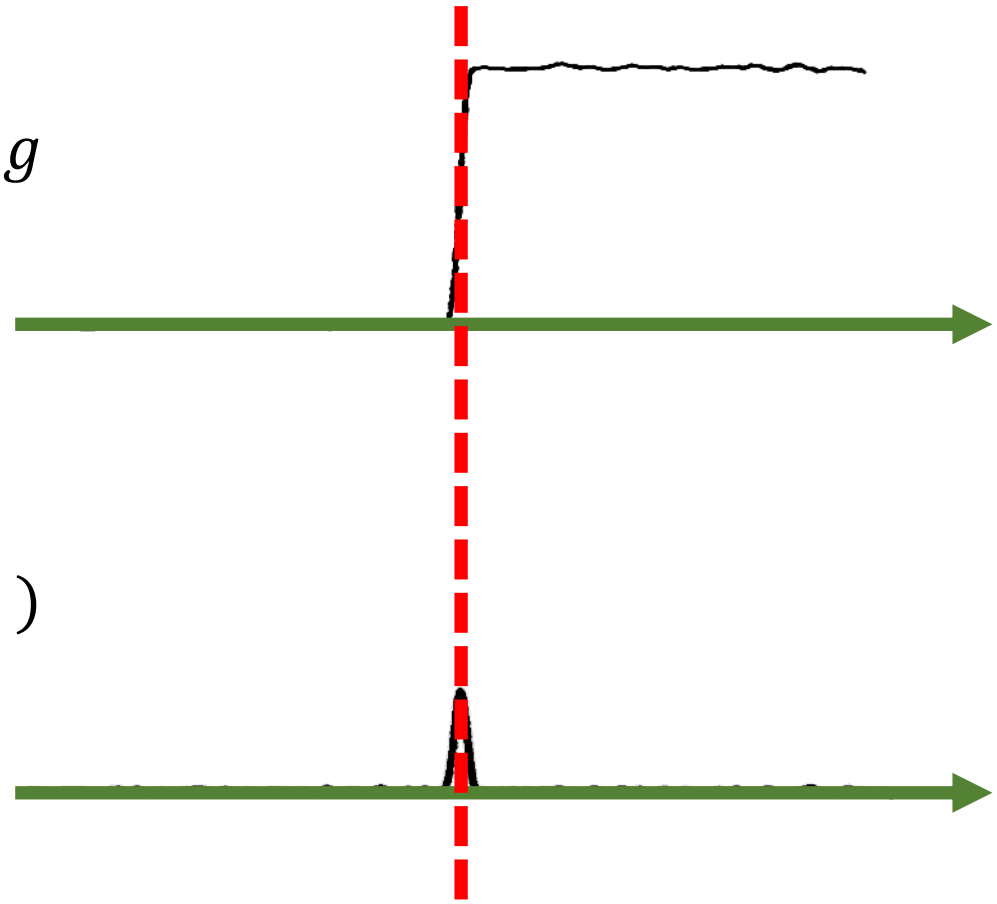
$f * g$

$f * g$

$\frac{d}{dx}(f * g)$

$f * g$

$\frac{d}{dx}(f * g)$



2D怎么办?

$$\frac{\partial}{\partial x}(g * I)$$

$$\frac{\partial}{\partial x}(g * I)$$

高斯滤波器

图像

$$\frac{\partial}{\partial x}(g * I)$$

$$\frac{\partial}{\partial x}(g * I) \equiv \frac{\partial g}{\partial x} * I$$

$$\frac{\partial}{\partial x}(g * I) \equiv \frac{\partial g}{\partial x} * I$$

卷积微分定理

高斯一阶导数
(Derivative of Gaussian, DoG)

$$\frac{\partial}{\partial x}(g * I) \equiv \frac{\partial g}{\partial x} * I$$

$$\frac{\partial}{\partial x}(g * I) \equiv \frac{\partial g}{\partial x} * I$$

如何进行离散滤波?

$$\frac{\partial}{\partial x}(g * I) \equiv \frac{\partial g}{\partial x} * I$$

方案1

0.0113	0.0838	0.0113
0.0838	0.6193	0.0838
0.0113	0.0838	0.0113

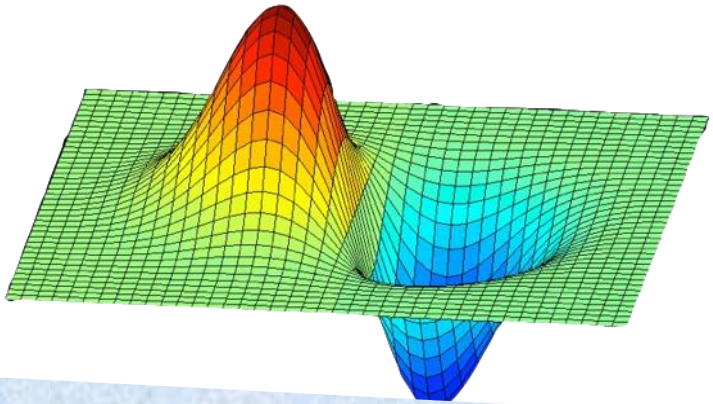
$*$

1	-1

$$\frac{\partial}{\partial x}(g * I) \equiv \frac{\partial g}{\partial x} * I$$

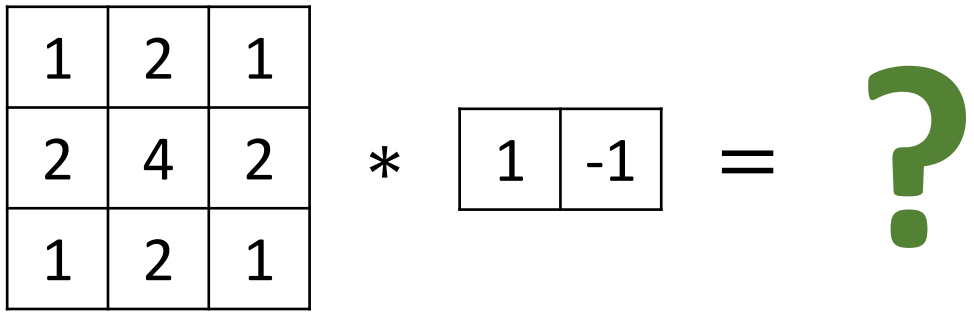
方案2

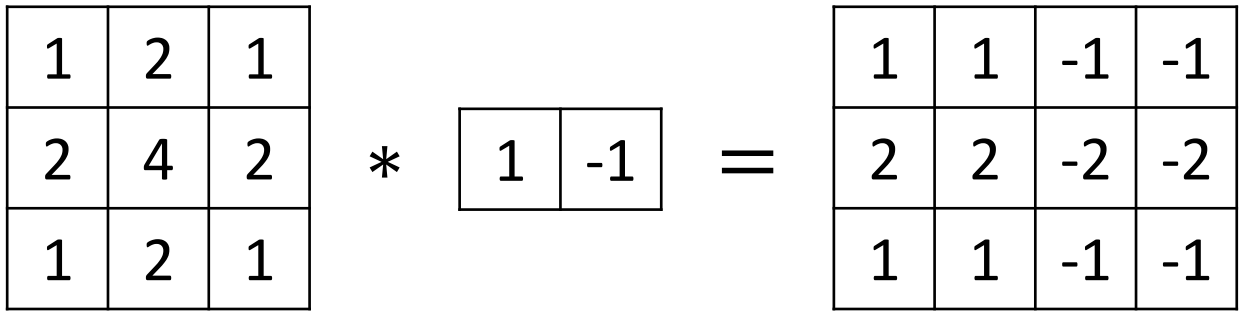
$$\frac{\partial}{\partial x} G(x, y; \sigma) = -\frac{x}{2\pi\sigma^4} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$$

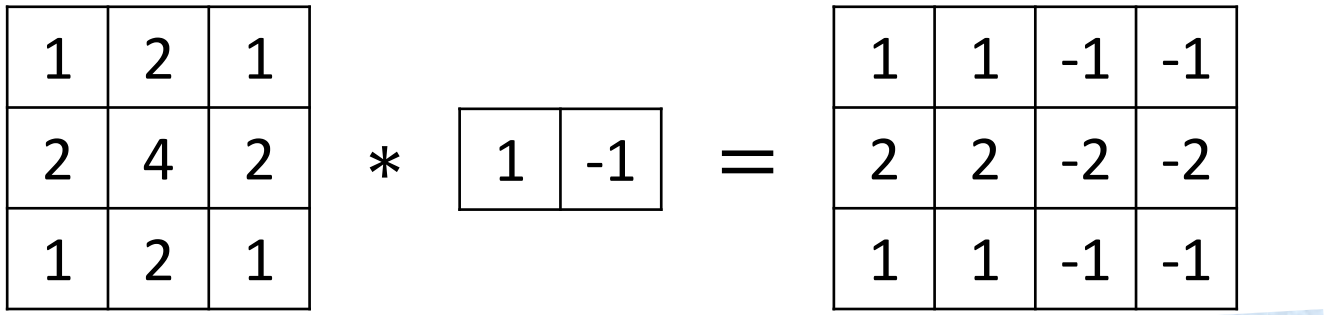


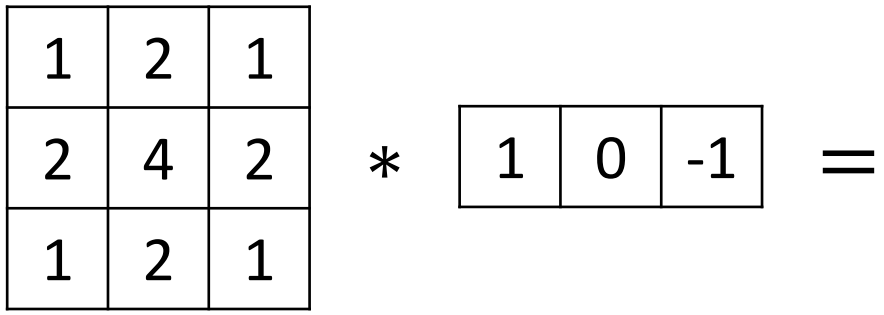
采样高斯一阶导数

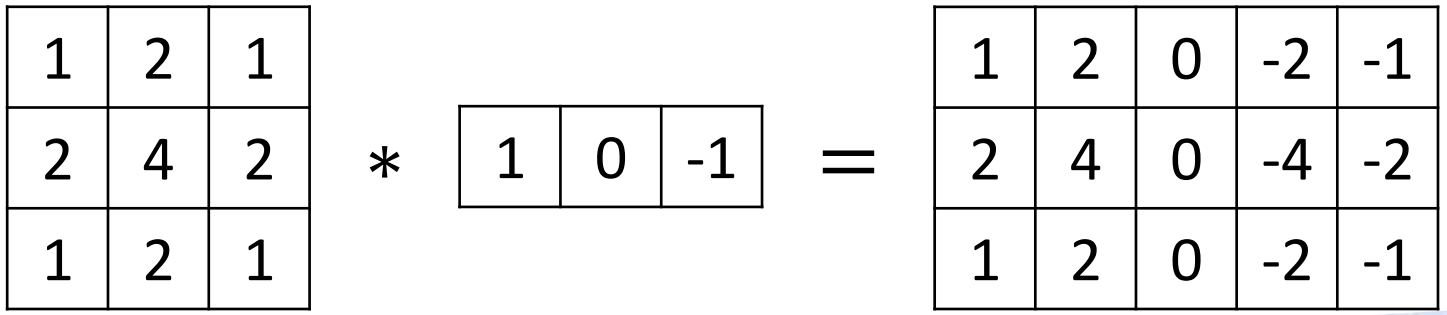
哪种方法更好?


1 2 1
2 4 2
1 2 1
*
1 -1
=
?


1 2 1
2 4 2
1 2 1
*
1 -1
=
1 1 -1 -1
2 2 -2 -2
1 1 -1 -1


1 2 1
2 4 2
1 2 1
*
1 -1
=
1 1 -1 -1
2 2 -2 -2
1 1 -1 -1
没有中心


1 2 1
2 4 2
1 2 1
*
1 0 -1
=


1 2 1
2 4 2
1 2 1
*
1 0 -1
=
1 2 0 -2 -1
2 4 0 -4 -2
1 2 0 -2 -1
眼熟吗

1	2	1
2	4	2
1	2	1

$*$

1	0	-1

$=$

1	2	0	-2	-1
2	4	0	-4	-2
1	2	0	-2	-1

Sobel

1	0	-1
2	0	-2
1	0	-1

方向可调

滤波器

The Design and Use of Steerable Filters

William T. Freeman and Edward H. Adelson

Abstract— **Oriented filters are useful in many early vision and image processing tasks. One often needs to apply the same filter, rotated to different angles under adaptive control, or wishes to calculate the filter response at various orientations. We present an efficient architecture to synthesize filters of arbitrary orientations from linear combinations of basis filters, allowing one to adaptively "steer" a filter to any orientation, and to determine analytically the filter output as a function of orientation. Steerable**

a filter of arbitrary orientation without explicitly applying that filter.

We use the term "steerable filter" to describe a class of filters in which a filter of arbitrary orientation is synthesized as a linear combination of a set of "basis filters." We will show that both two- and three-dimensional functions are steerable as well as how many basis filters are needed to steer a given filter.

TPAMI, 1991

方向可调滤波器

一类方向选择滤波器，其中任意方向的滤波器可以通过一组“基”滤波器的线性组合来表示

定义：

方向导数，$\nabla_{\mathbf{u}} f(x_0)$，是函数$f(x_0)$在$x_0$点沿$\hat{\mathbf{u}}$方向的变换率。它可以被定义为：

定义：

方向导数，$\nabla_{\mathbf{u}} f(x_0)$，是函数$f(x_0)$在$x_0$点沿$\hat{\mathbf{u}}$方向的变换率。它可以被定义为：

$$\nabla_{\mathbf{u}} f(x_0) = \lim_{h \to 0} \frac{f(x_0 + h\hat{\mathbf{u}}) - f(x_0)}{h}$$

定义：

方向导数，$\nabla_{\mathbf{u}} f(x_0)$，是函数$f(x_0)$在$x_0$点沿$\hat{\mathbf{u}}$方向的变换率。它可以被定义为：

$$\nabla_{\mathbf{u}} f(x_0) = \lim_{h \to 0} \frac{f(x_0 + h\hat{\mathbf{u}}) - f(x_0)}{h}$$

$$= \lim_{h \to 0} \frac{f(x_0 + h\hat{\mathbf{u}}) - f(x_0)}{h\hat{\mathbf{u}}} \cdot \hat{\mathbf{u}}$$

定义：

方向导数，$\nabla_{\mathbf{u}} f(x_0)$，是函数$f(x_0)$在$x_0$点沿$\hat{\mathbf{u}}$方向的变换率。它可以被定义为：

$$\nabla_{\mathbf{u}} f(x_0) = \lim_{h \to 0} \frac{f(x_0 + h\hat{\mathbf{u}}) - f(x_0)}{h}$$

$$= \lim_{h \to 0} \frac{f(x_0 + h\hat{\mathbf{u}}) - f(x_0)}{h\hat{\mathbf{u}}} \cdot \hat{\mathbf{u}}$$

$$= \nabla f(x_0) \cdot \hat{\mathbf{u}}$$

定义：

方向导数，$\nabla_{\mathbf{u}} f(x_0)$，是函数$f(x_0)$在$x_0$点沿$\hat{\mathbf{u}}$方向的变换率。它可以被定义为：

$$\nabla_{\mathbf{u}} f(x_0) = \lim_{h \to 0} \frac{f(x_0 + h\hat{\mathbf{u}}) - f(x_0)}{h}$$

$$= \lim_{h \to 0} \frac{f(x_0 + h\hat{\mathbf{u}}) - f(x_0)}{h\hat{\mathbf{u}}} \cdot \hat{\mathbf{u}}$$

$$= \nabla f(x_0) \cdot \hat{\mathbf{u}}$$

点积，梯度在$\hat{\mathbf{u}}$方向的投影

$$\nabla_{\mathbf{u}} f(x_0) = \nabla f(x_0) \cdot \hat{\mathbf{u}}$$

$$\nabla_{\mathbf{u}} f(x_0) = \nabla f(x_0) \cdot \hat{\mathbf{u}}$$

令$\hat{\mathbf{u}} = (\cos\theta\,, \sin\theta)$

$$\nabla_{\mathbf{u}} f(x_0) = \nabla f(x_0) \cdot \hat{\mathbf{u}}$$

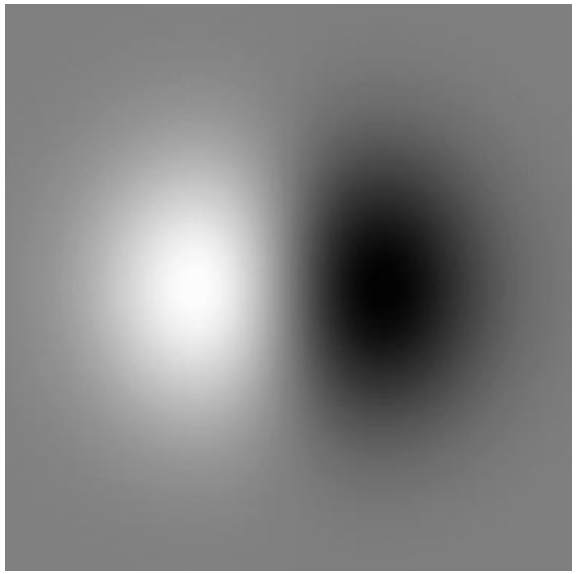
令$\hat{\mathbf{u}} = (\cos\theta\,, \sin\theta)$

$$\cos\theta \cdot \frac{\partial G(x,y)}{\partial x} + \sin\theta \cdot \frac{\partial G(x,y)}{\partial y} = \frac{\partial G(x,y)}{\partial \hat{\mathbf{u}}}$$

$$\nabla_{\mathbf{u}} f(x_0) = \nabla f(x_0) \cdot \hat{\mathbf{u}}$$

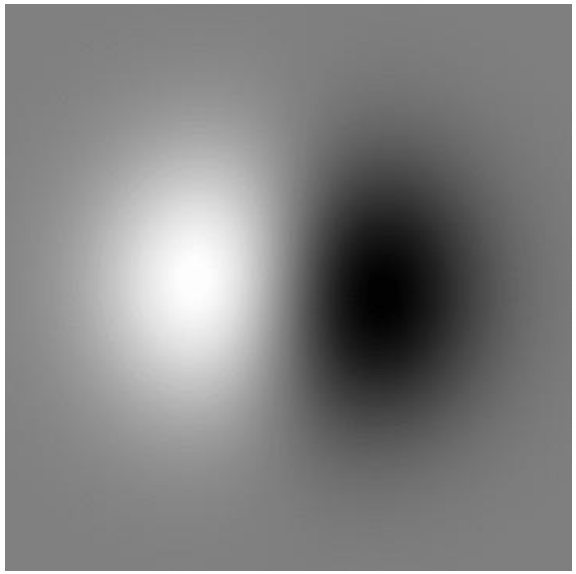
令$\hat{\mathbf{u}} = (\cos\theta\,, \sin\theta)$

$$\cos\theta \cdot \frac{\partial G(x,y)}{\partial x} + \sin\theta \cdot \frac{\partial G(x,y)}{\partial y} = \frac{\partial G(x,y)}{\partial \hat{\mathbf{u}}} := \frac{\partial G(x,y)}{\partial \theta}$$



$$+$$

$$=$$



$$\cos\theta \cdot \frac{\partial G(x,y)}{\partial x} \qquad \sin\theta \cdot \frac{\partial G(x,y)}{\partial y} \qquad \frac{\partial G(x,y)}{\partial \theta}$$

$$I_\theta(x,y) = \left[\cos\theta \cdot \frac{\partial G(x,y)}{\partial x} + \sin\theta \cdot \frac{\partial G(x,y)}{\partial y}\right] * I(x,y)$$

$$I_\theta(x,y) = \left[\cos\theta \cdot \frac{\partial G(x,y)}{\partial x} + \sin\theta \cdot \frac{\partial G(x,y)}{\partial y}\right] * I(x,y)$$

能更有效地计算吗？

$$I_\theta(x,y) = \left[\cos\theta \cdot \frac{\partial G(x,y)}{\partial x} + \sin\theta \cdot \frac{\partial G(x,y)}{\partial y}\right] * I(x,y)$$

能更有效地计算吗？

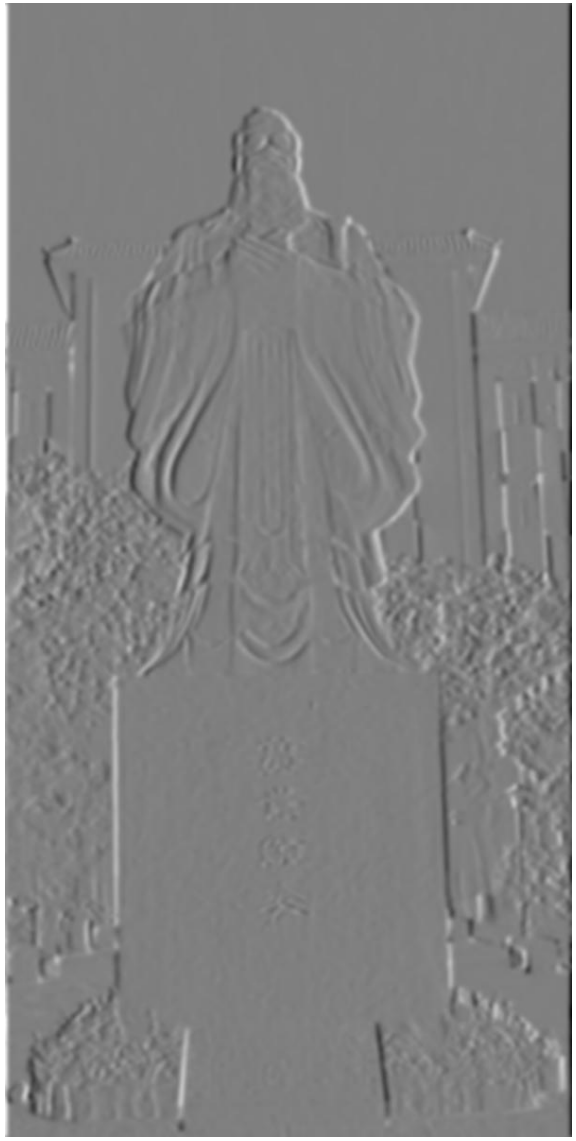
$$= \cos\theta \left[\frac{\partial G(x,y)}{\partial x} * I(x,y)\right] + \sin\theta \left[\frac{\partial G(x,y)}{\partial y} * I(x,y)\right]$$

$$I_\theta(x,y) = \left[\cos\theta \cdot \frac{\partial G(x,y)}{\partial x} + \sin\theta \cdot \frac{\partial G(x,y)}{\partial y}\right] * I(x,y)$$

能更有效地计算吗？

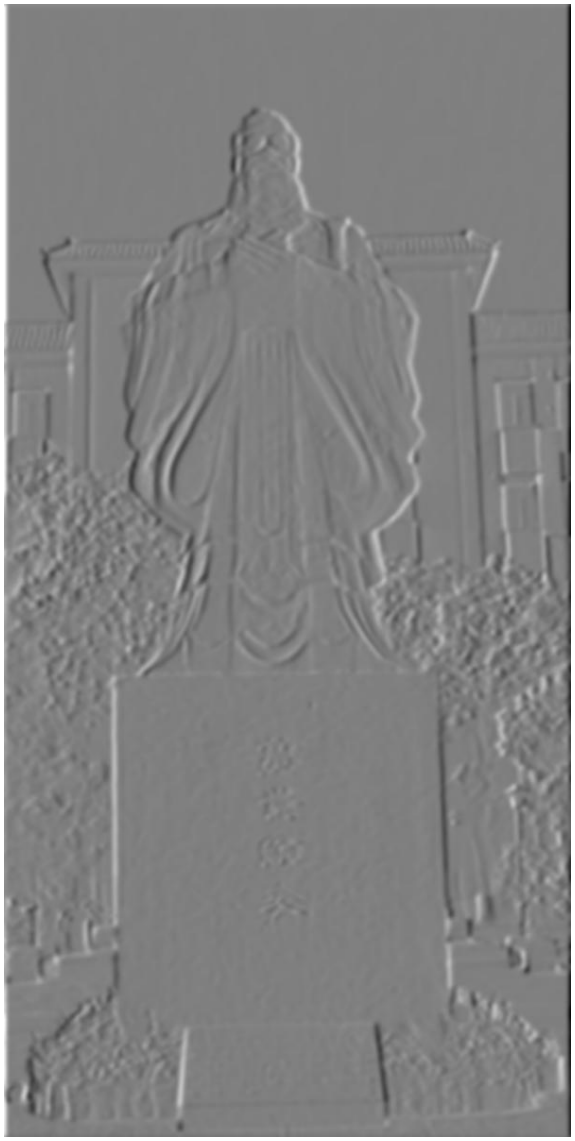
$$= \cos\theta \left[\frac{\partial G(x,y)}{\partial x} * I(x,y)\right] + \sin\theta \left[\frac{\partial G(x,y)}{\partial y} * I(x,y)\right]$$

利用分配率高效地实现



+

=



$$\cos\theta\left[\frac{\partial G}{\partial x} * I\right] \qquad \sin\theta\left[\frac{\partial G}{\partial \mathrm{y}} * I\right] \qquad \frac{\partial[G * I]}{\partial\theta}$$

接缝裁剪

Seam Carving for Content-Aware Image Resizing

Shai Avidan
Mitsubishi Electric Research Labs

Ariel Shamir
The Interdisciplinary Center & MERL

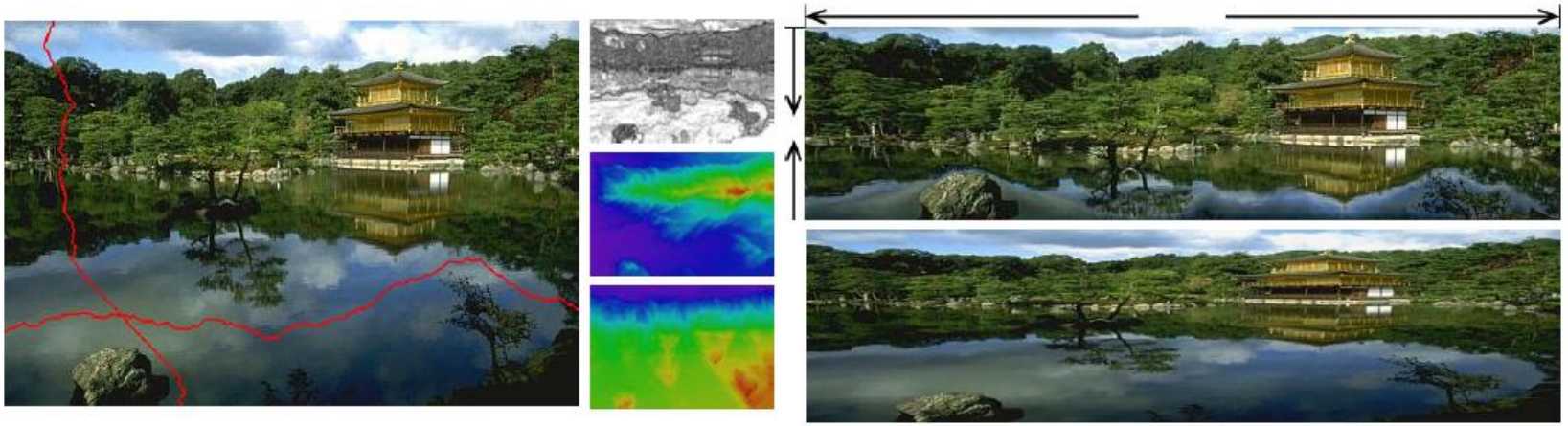


Figure 1: A seam is a connected path of low energy pixels in an image. On the left is the original image with

SIGGRAPH, 2007


输入图像
ICUC
ICUC
ICUC

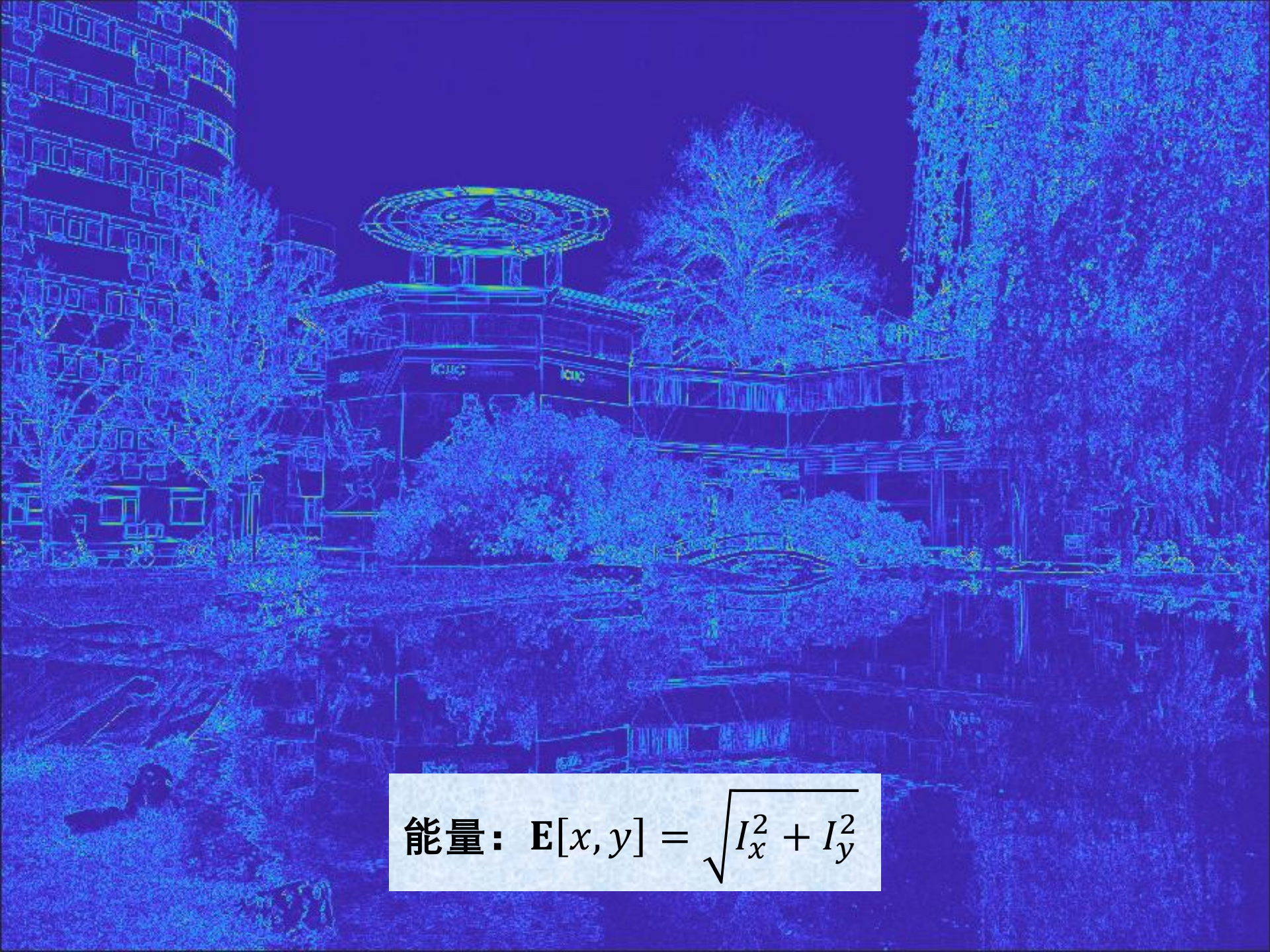
裁剪图像

调整图像大小

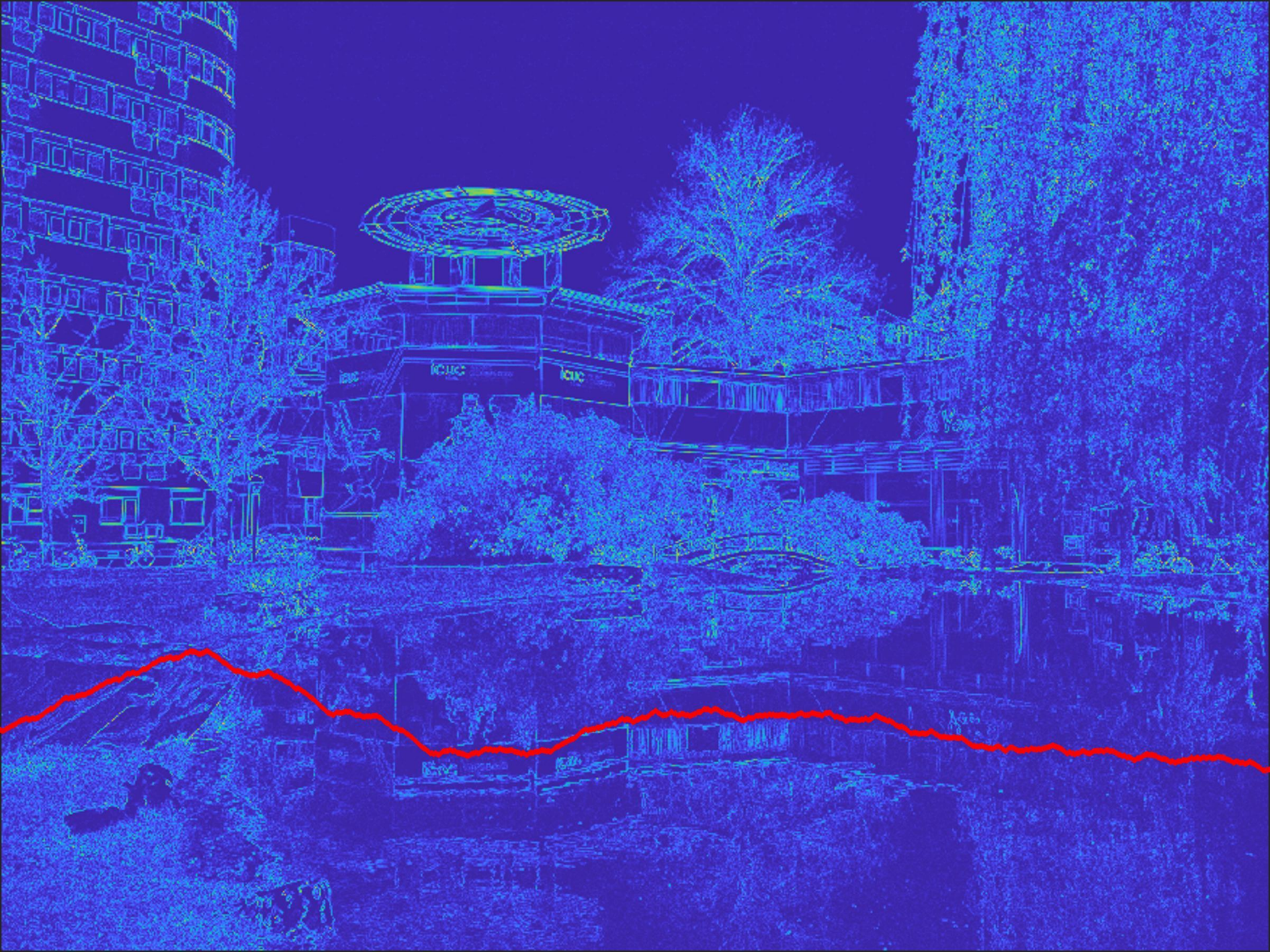

输入图像
ICUC
ICUC
ICUC


接缝裁剪结果
ICUC
ICUC
ICUC

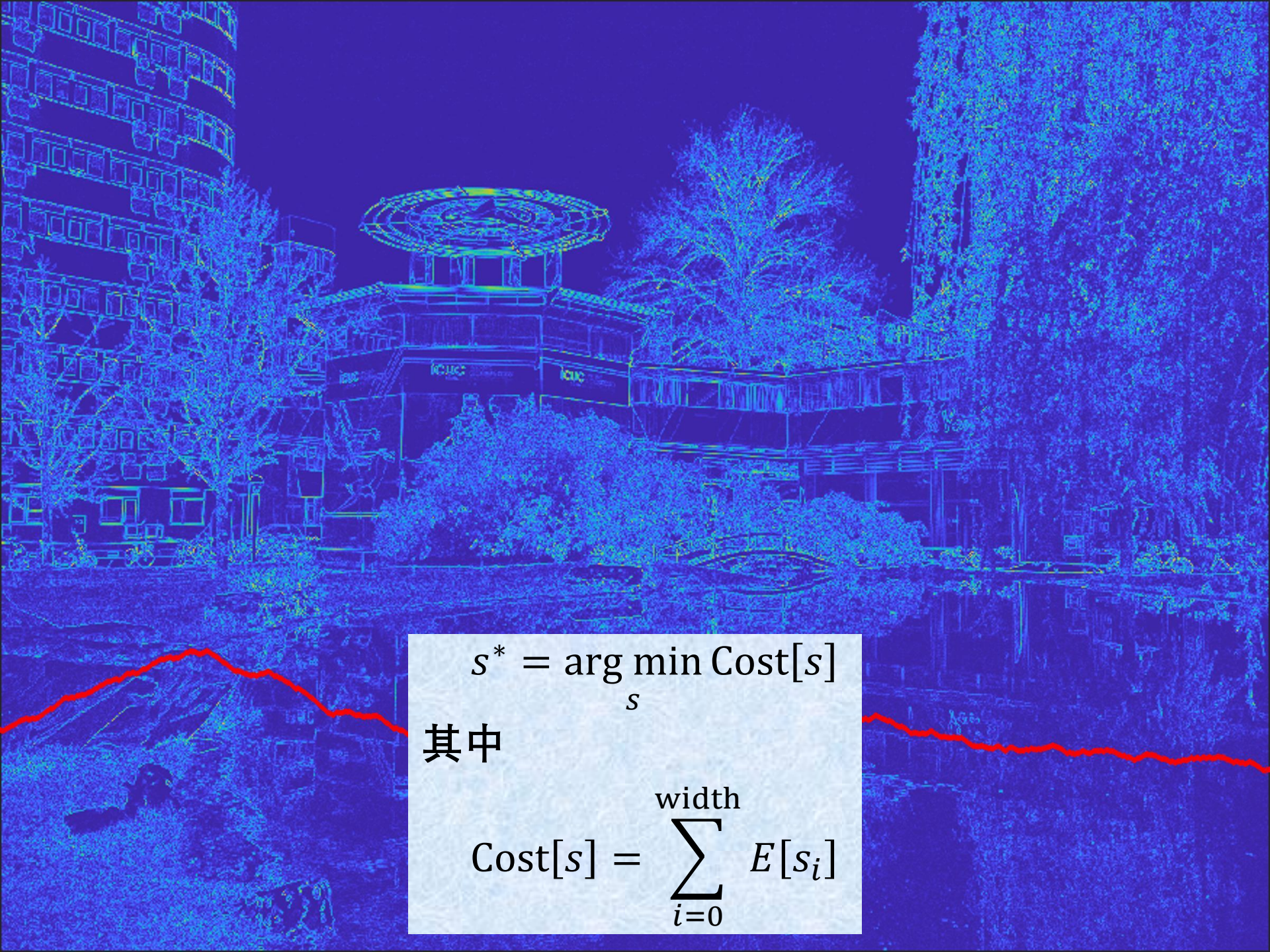

接缝裁剪结果
通过移除不显著接缝减小图像尺寸


能量：$\mathbf{E}[x,y] = \sqrt{I_x^2 + I_y^2}$

接缝是由相邻像素组成的路径



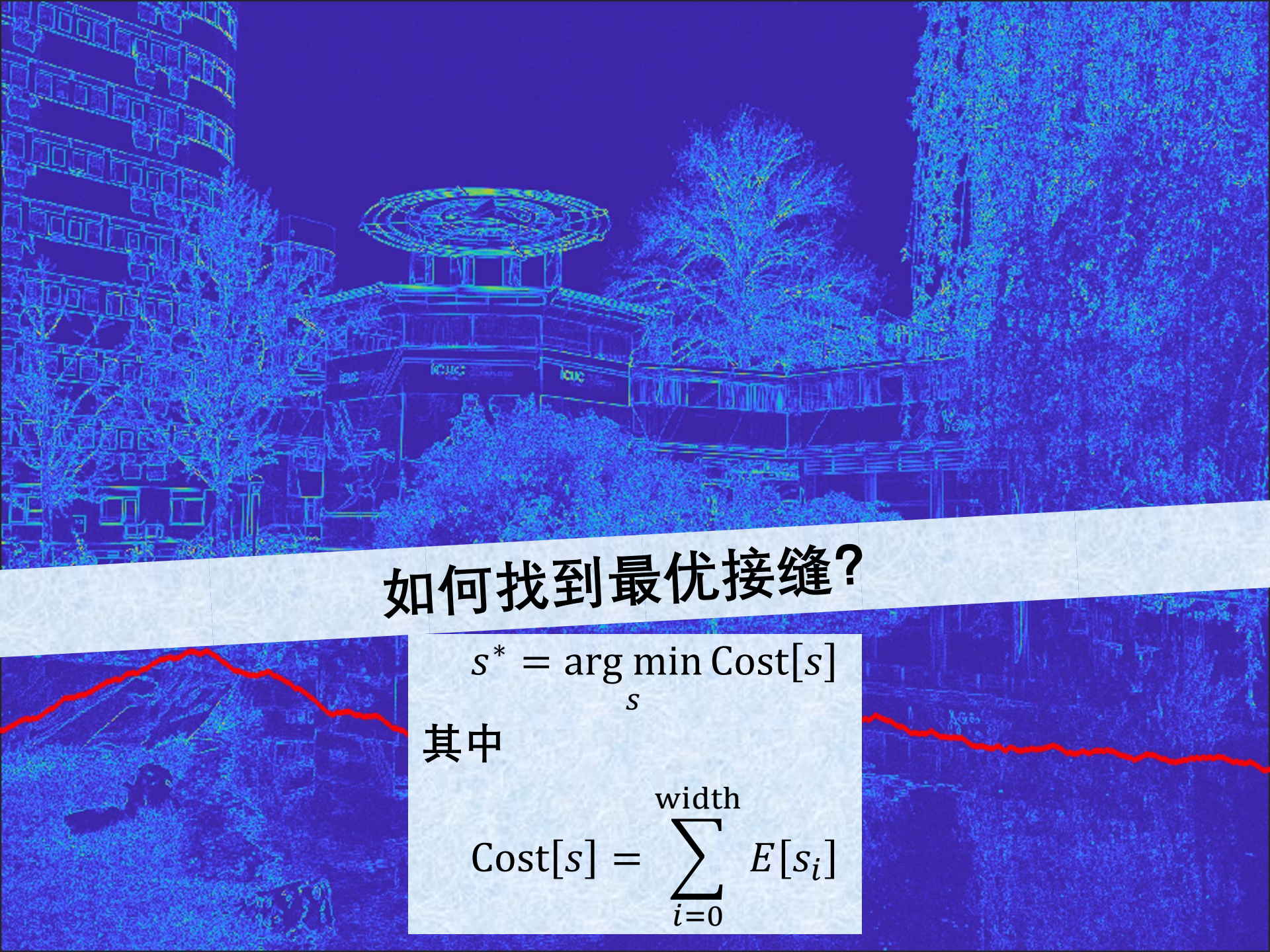

目标：找到累积能量最小的接缝



$$s^* = \arg\min_{s} \text{Cost}[s]$$

其中

$$\text{Cost}[s] = \sum_{i=0}^{\text{width}} E[s_i]$$



如何找到最优接缝?

$$s^* = \arg\min_s \text{Cost}[s]$$

其中

$$\text{Cost}[s] = \sum_{i=0}^{\text{width}} E[s_i]$$

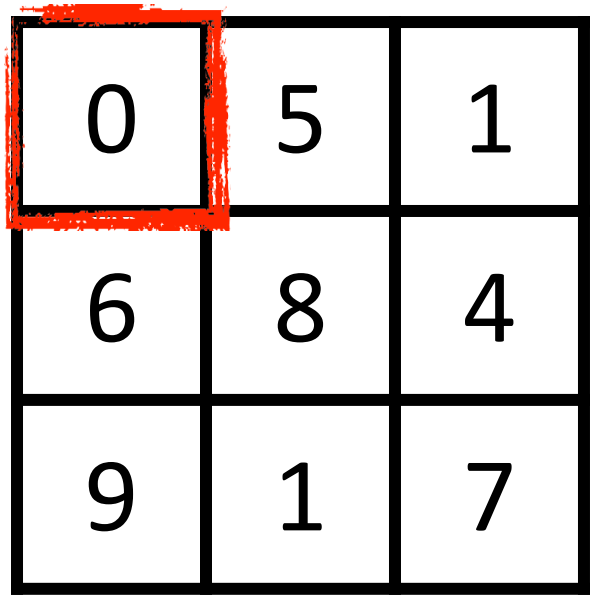
穷举 (暴力) 法


贪婪
算法

$\mathbf{E}[i, j]$

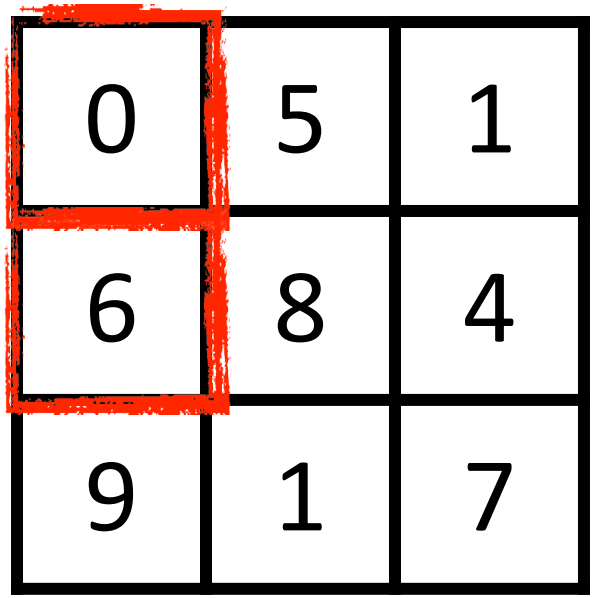
0	5	1
6	8	4
9	1	7

$\mathbf{E}[i,j]$



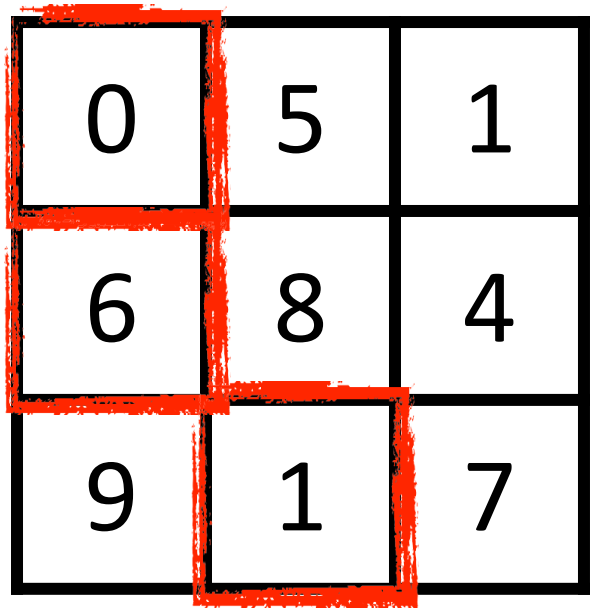
贪婪算法

$\mathbf{E}[i,j]$



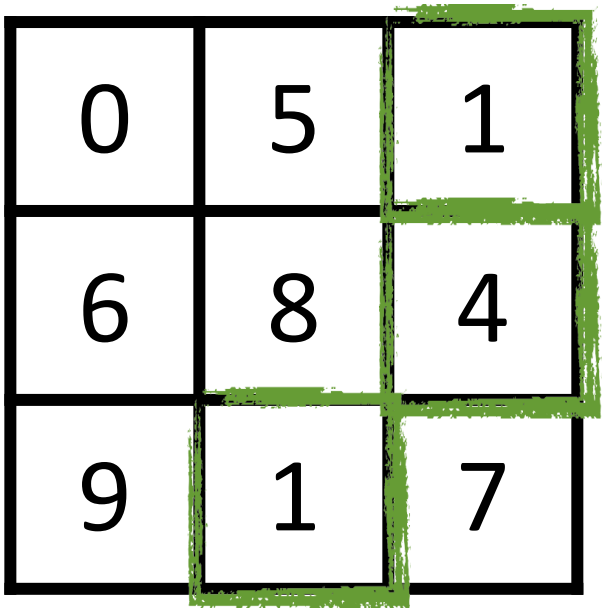
贪婪算法

$\mathbf{E}[i,j]$



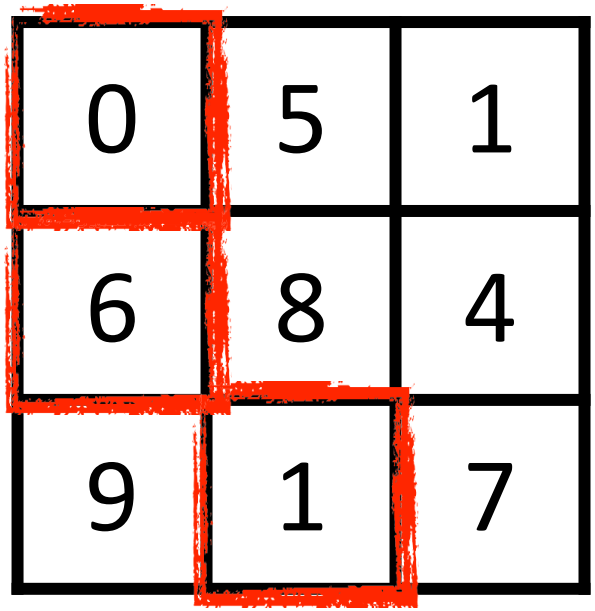
贪婪算法

$\mathbf{E}[i,j]$

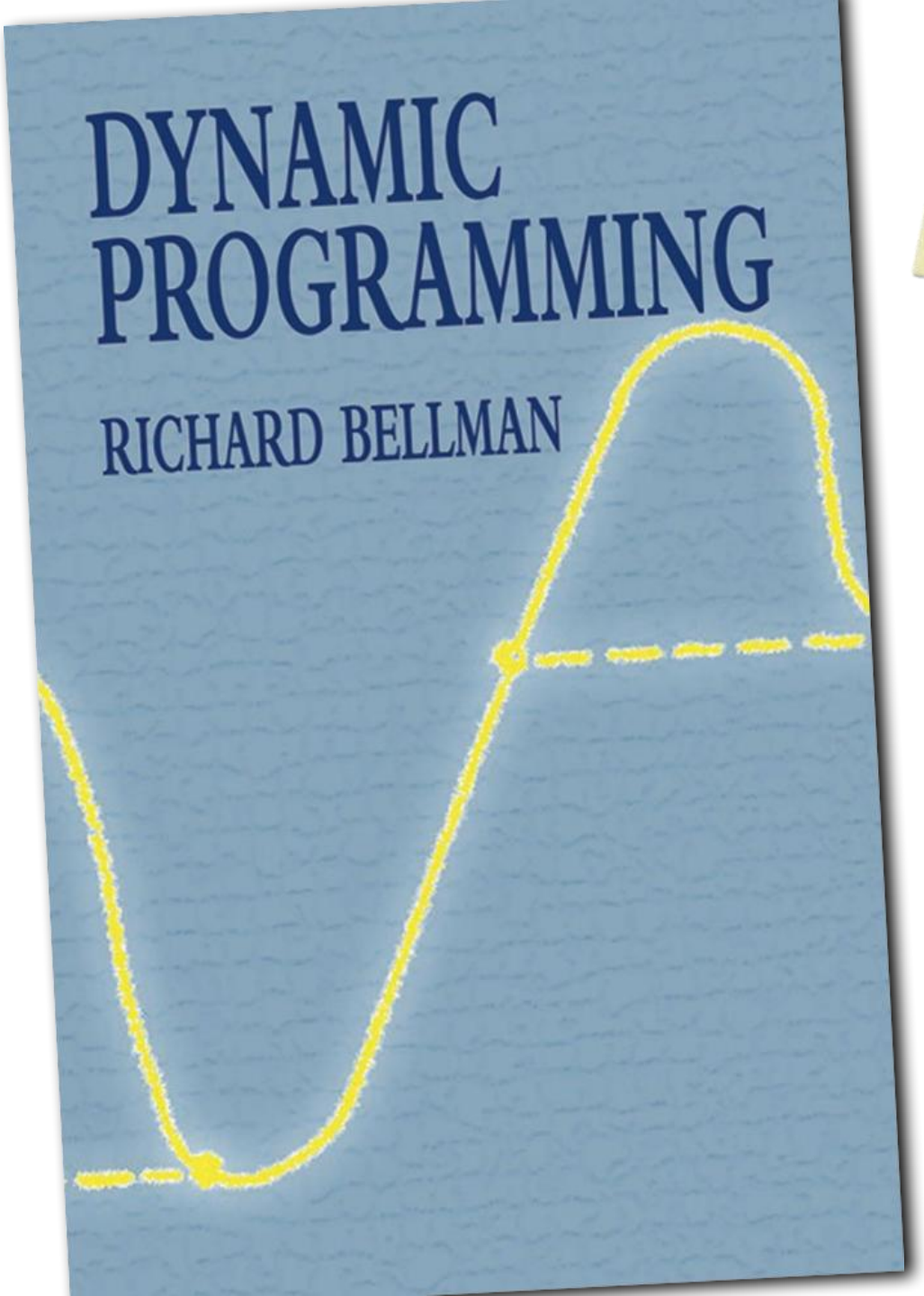


最优路径

$\mathbf{E}[i,j]$



贪婪算法



动态规划

$$\mathbf{M}[i,j] =$$
$$\mathbf{E}[i,j] +$$
$$\min(\mathbf{M}[i-1,j-1], \mathbf{M}[i,j-1], \mathbf{M}[i+1,j-1])$$

动态规划

$$\mathbf{M}[i,j] =$$
$$\mathbf{E}[i,j] +$$
$$\min(\mathbf{M}[i-1,j-1], \mathbf{M}[i,j-1], \mathbf{M}[i+1,j-1])$$

以像素$[i,j]$结尾的最小累积能量

$$\mathbf{M}[i,j] =$$
$$\mathbf{E}[i,j] +$$
$$\min(\mathbf{M}[i-1,j-1], \mathbf{M}[i,j-1], \mathbf{M}[i+1,j-1])$$

$\mathbf{E}[i,j]$

0	5	1
6	8	4
9	1	7

最优路径

$\mathbf{M}[i,j]$

局部最优解

$$\mathbf{M}[i,j] =$$
$$\mathbf{E}[i,j] +$$
$$\min(\mathbf{M}[i-1,j-1], \mathbf{M}[i,j-1], \mathbf{M}[i+1,j-1])$$

$\mathbf{E}[i,j]$

0	5	1
6	8	4
9	1	7

最优路径

$\mathbf{M}[i,j]$

0	5	1

局部最优解

$$\mathbf{M}[i,j] =$$
$$\mathbf{E}[i,j] +$$
$$\min(\mathbf{M}[i-1,j-1], \mathbf{M}[i,j-1], \mathbf{M}[i+1,j-1])$$

$\mathbf{E}[i,j]$

0	5	1
6	8	4
9	1	7

最优路径

$\mathbf{M}[i,j]$

0	5	1
6		

局部最优解

$$\mathbf{M}[i,j] =$$
$$\mathbf{E}[i,j] +$$
$$\min(\mathbf{M}[i-1,j-1], \mathbf{M}[i,j-1], \mathbf{M}[i+1,j-1])$$

$\mathbf{E}[i,j]$

0	5	1
6	8	4
9	1	7

最优路径

$\mathbf{M}[i,j]$

0	5	1
6	8	

局部最优解

$$\mathbf{M}[i,j] =$$
$$\mathbf{E}[i,j] +$$
$$\min(\mathbf{M}[i-1,j-1], \mathbf{M}[i,j-1], \mathbf{M}[i+1,j-1])$$

$\mathbf{E}[i,j]$

0	5	1
6	8	4
9	1	7

最优路径

$\mathbf{M}[i,j]$

0	5	1
6	8	5

局部最优解

$$\mathbf{M}[i,j] =$$
$$\mathbf{E}[i,j] +$$
$$\min(\mathbf{M}[i-1,j-1], \mathbf{M}[i,j-1], \mathbf{M}[i+1,j-1])$$

$\mathbf{E}[i,j]$

0	5	1
6	8	4
9	1	7

最优路径

$\mathbf{M}[i,j]$

0	5	1
6	8	5
15		

局部最优解

$$\mathbf{M}[i,j] =$$
$$\mathbf{E}[i,j] +$$
$$\min(\mathbf{M}[i-1,j-1], \mathbf{M}[i,j-1], \mathbf{M}[i+1,j-1])$$

$\mathbf{E}[i,j]$

0	5	1
6	8	4
9	1	7

最优路径

$\mathbf{M}[i,j]$

0	5	1
6	8	5
15	6	

局部最优解

$$\mathbf{M}[i,j] =$$
$$\mathbf{E}[i,j] +$$
$$\min(\mathbf{M}[i-1,j-1], \mathbf{M}[i,j-1], \mathbf{M}[i+1,j-1])$$

$\mathbf{E}[i,j]$

0	5	1
6	8	4
9	1	7

最优路径

$\mathbf{M}[i,j]$

0	5	1
6	8	5
15	6	12

局部最优解

$$\mathbf{M}[i,j] = \mathbf{E}[i,j] + \min(\mathbf{M}[i-1,j-1], \mathbf{M}[i,j-1], \mathbf{M}[i+1,j-1])$$

$\mathbf{E}[i,j]$

0	5	1
6	8	4
9	1	7

最优路径

$\mathbf{M}[i,j]$

0	5	1
6	8	5
15	6	12

局部最优解

为了找到最佳接缝，从最小值开始回溯

$$\mathbf{M}[i,j] = \mathbf{E}[i,j] + \min(\mathbf{M}[i-1,j-1], \mathbf{M}[i,j-1], \mathbf{M}[i+1,j-1])$$

$\mathbf{E}[i,j]$

0	5	1
6	8	4
9	1	7

最优路径

$\mathbf{M}[i,j]$

0	5	1
6	8	5
15	6	12

局部最优解

为了找到最佳接缝，从最小值开始回溯

$$\mathbf{M}[i,j] = \mathbf{E}[i,j] + \min(\mathbf{M}[i-1,j-1], \mathbf{M}[i,j-1], \mathbf{M}[i+1,j-1])$$

$\mathbf{E}[i,j]$

0	5	1
6	8	4
9	1	7

最优路径

$\mathbf{M}[i,j]$

0	5	1
6	8	5
15	6	12

局部最优解

为了找到最佳接缝，从最小值开始回溯

$$\mathbf{M}[i,j] = \mathbf{E}[i,j] + \min(\mathbf{M}[i-1,j-1], \mathbf{M}[i,j-1], \mathbf{M}[i+1,j-1])$$

$\mathbf{E}[i,j]$

0	5	1
6	8	4
9	1	7

最优路径

$\mathbf{M}[i,j]$

0	5	1
6	8	5
15	6	12

局部最优解

为了找到最佳接缝，从最小值开始回溯


输入图像

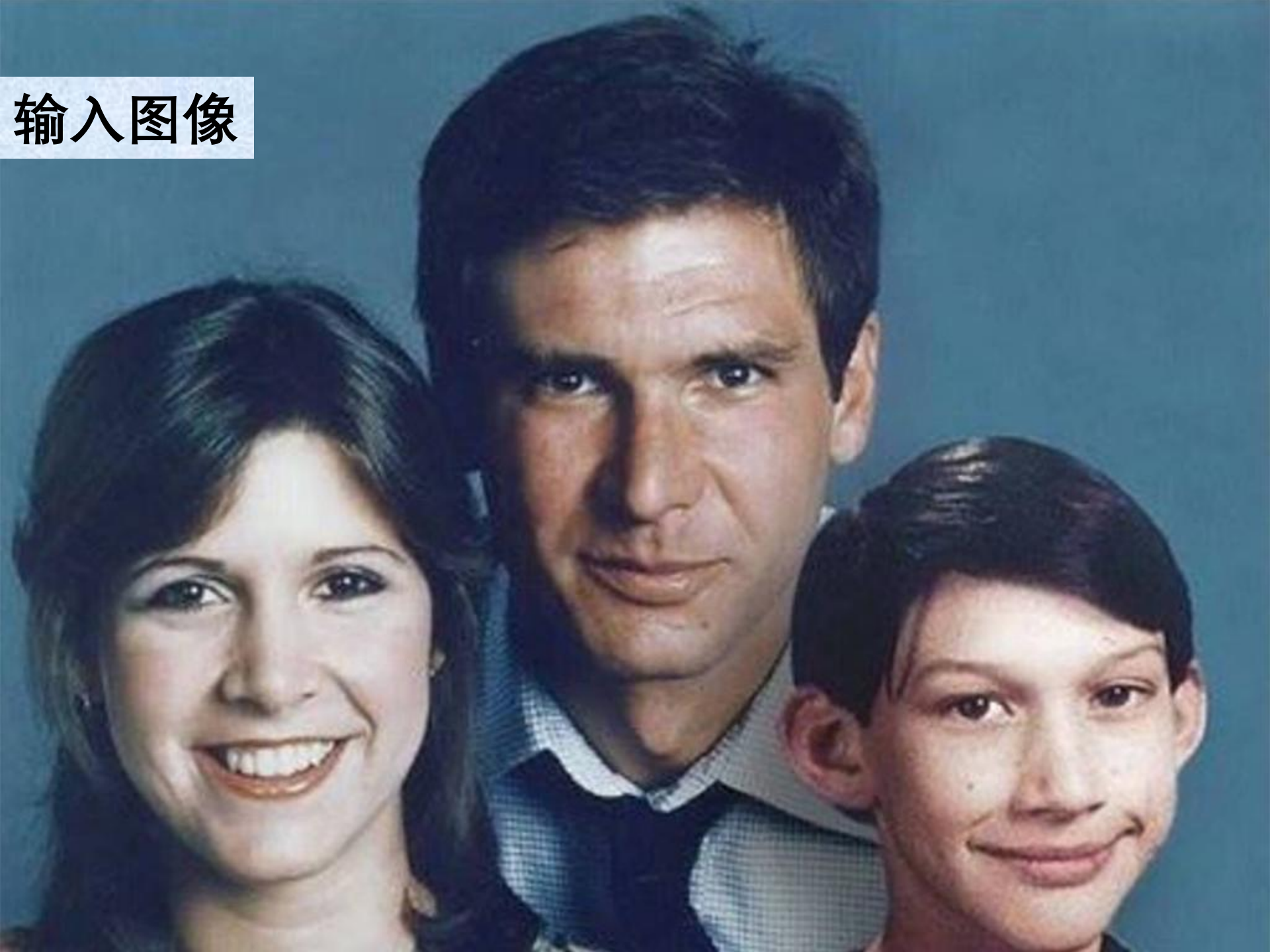
接缝裁剪结果

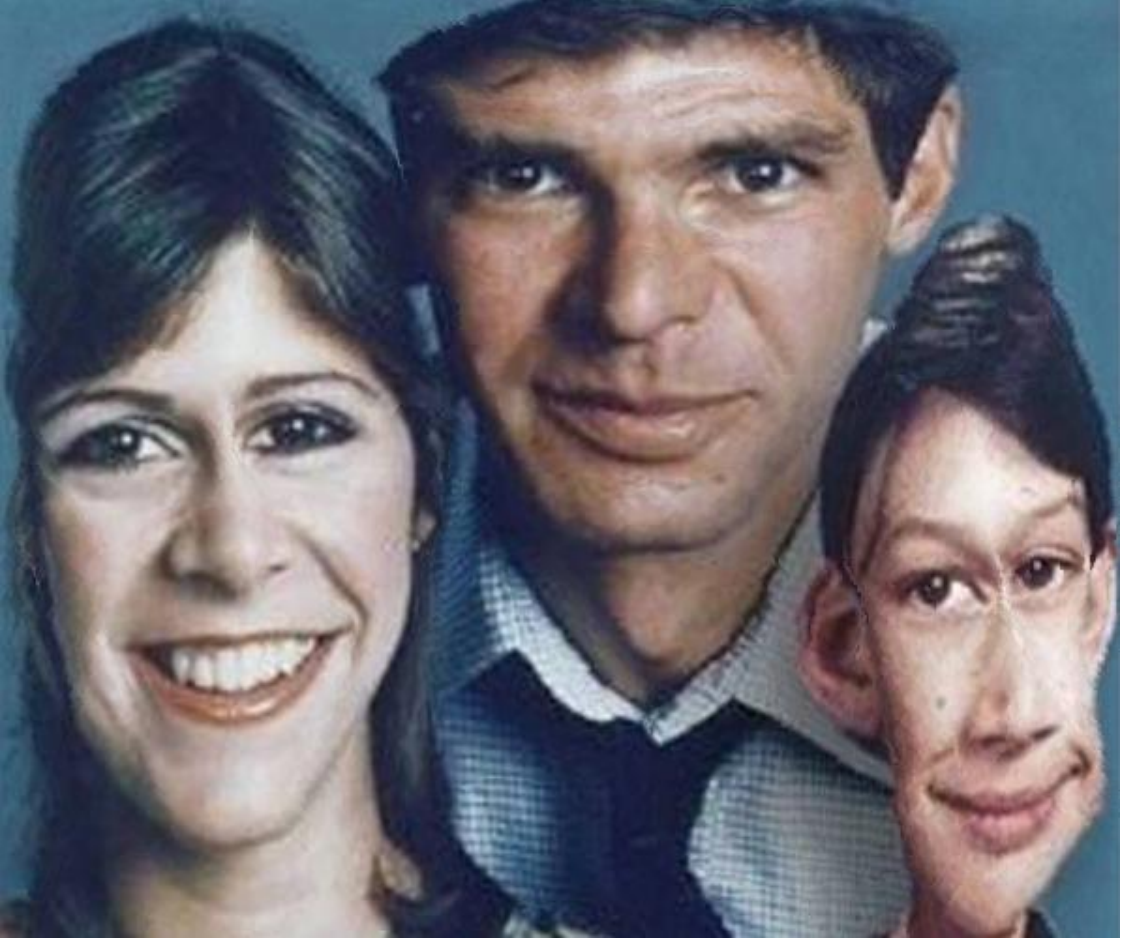

输入图像

接缝裁剪结果


输入图像
LEUTY LIFEGUARD STATION

接缝裁剪结果


输入图像



接缝裁剪结果

Canny算子

IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. PAMI-8, NO. 6, NOVEMBER 1986

A Computational Approach to Edge Detection

JOHN CANNY, MEMBER, IEEE

Abstract—**This paper describes a computational approach to edge detection. The success of the approach depends on the definition of a *comprehensive* set of goals for the computation of edge points. These goals must be precise enough to delimit the desired behavior of the detector while making minimal assumptions about the form of the solution. We define detection and localization criteria for a class of edges, and present mathematical forms for these criteria as functionals on the operator impulse response. A third criterion is then added to ensure**

detector as input to a program which could isolate simple geometric solids. More recently the model-based vision system ACRONYM [3] used an edge detector as the front end to a sophisticated recognition program. Shape from motion [29], [13] can be used to infer the structure of three-dimensional objects from the motion of edge con-

TPAMI, 1986

A Computational Approach to Edge Detection

JOHN CANNY, MEMBER, IEEE

Abstract—**This paper describes a computational approach to edge detection. The success of the approach depends on the definition of a *comprehensive* set of goals for the computation of edge points. These goals must be precise enough to delimit the desired behavior of the detector while making minimal assumptions about the form of the solution. We define detection and localization criteria for a class of edges, and present mathematical forms for these criteria as functionals on the operator impulse response. A third criterion is then added to ensure**

detector as input to a program which could isolate simple geometric solids. More recently the model-based vision system ACRONYM [3] used an edge detector as the front end to a sophisticated recognition program. Shape from motion [29], [13] can be used to infer the structure of three-dimensional objects from the motion of edge con-

引用超过4万次！

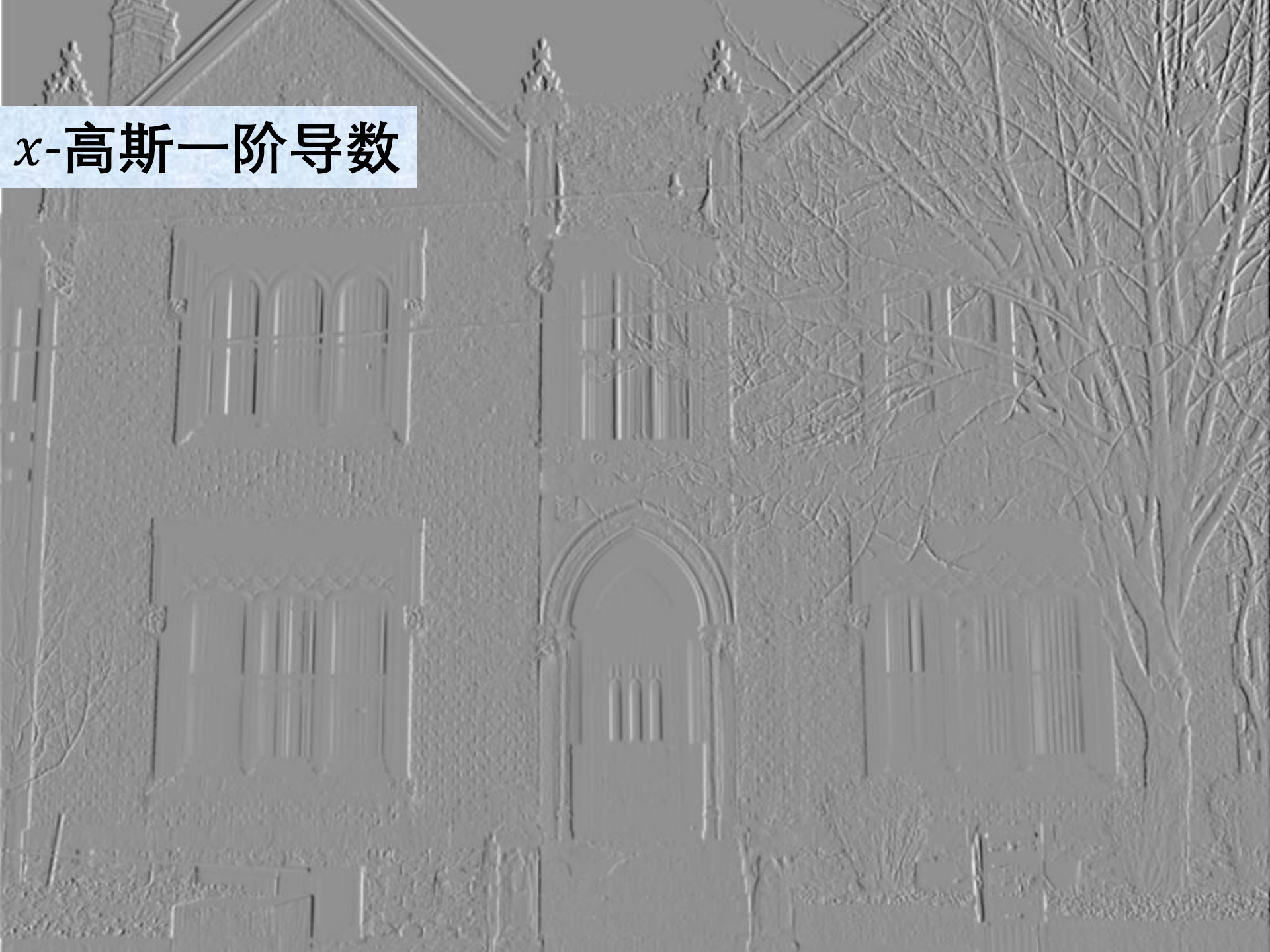
4

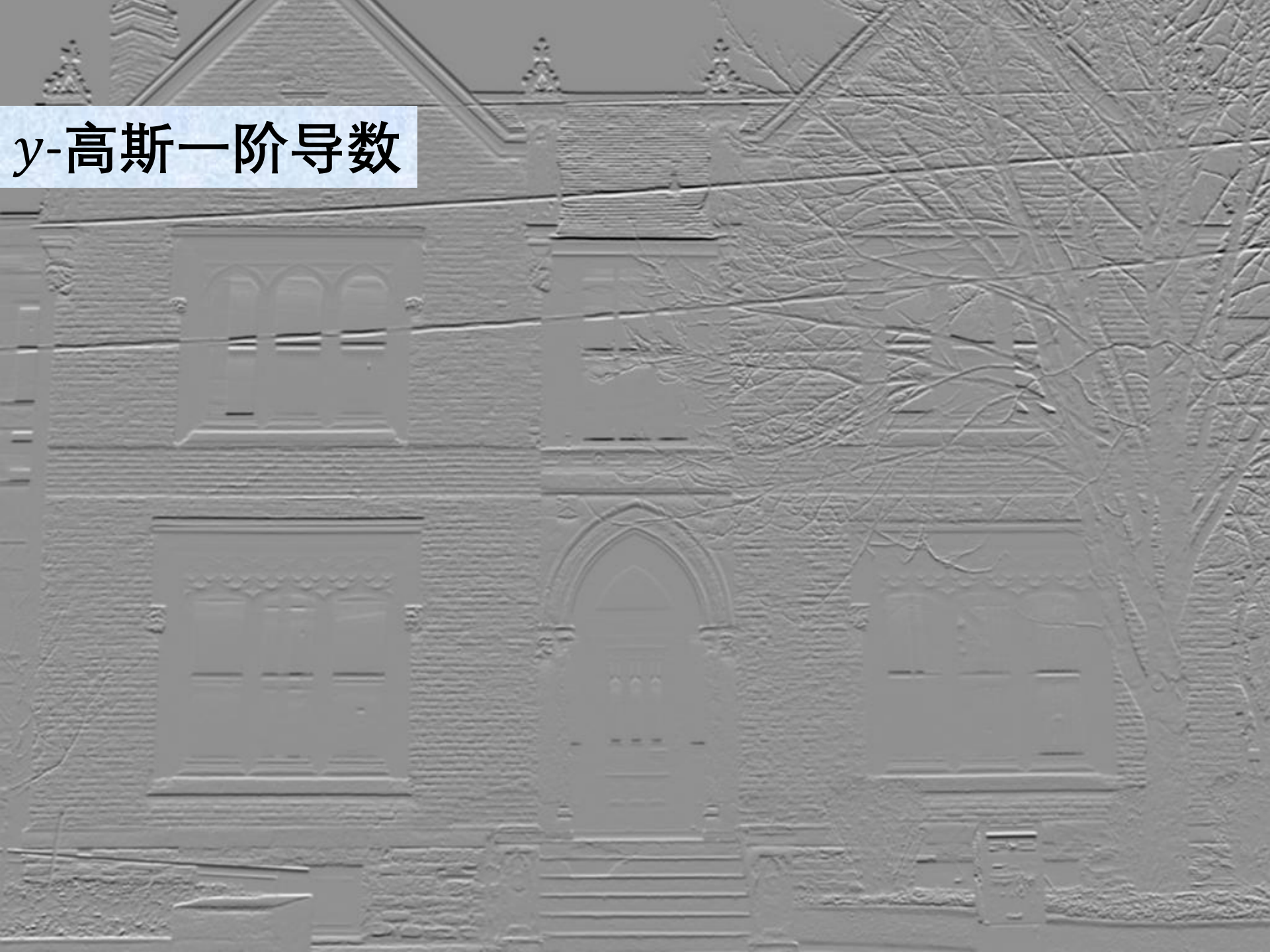
主要步骤

步骤1

使用x、y高斯一阶导数对图像滤波


输入图像


x-高斯一阶导数


y-高斯一阶导数

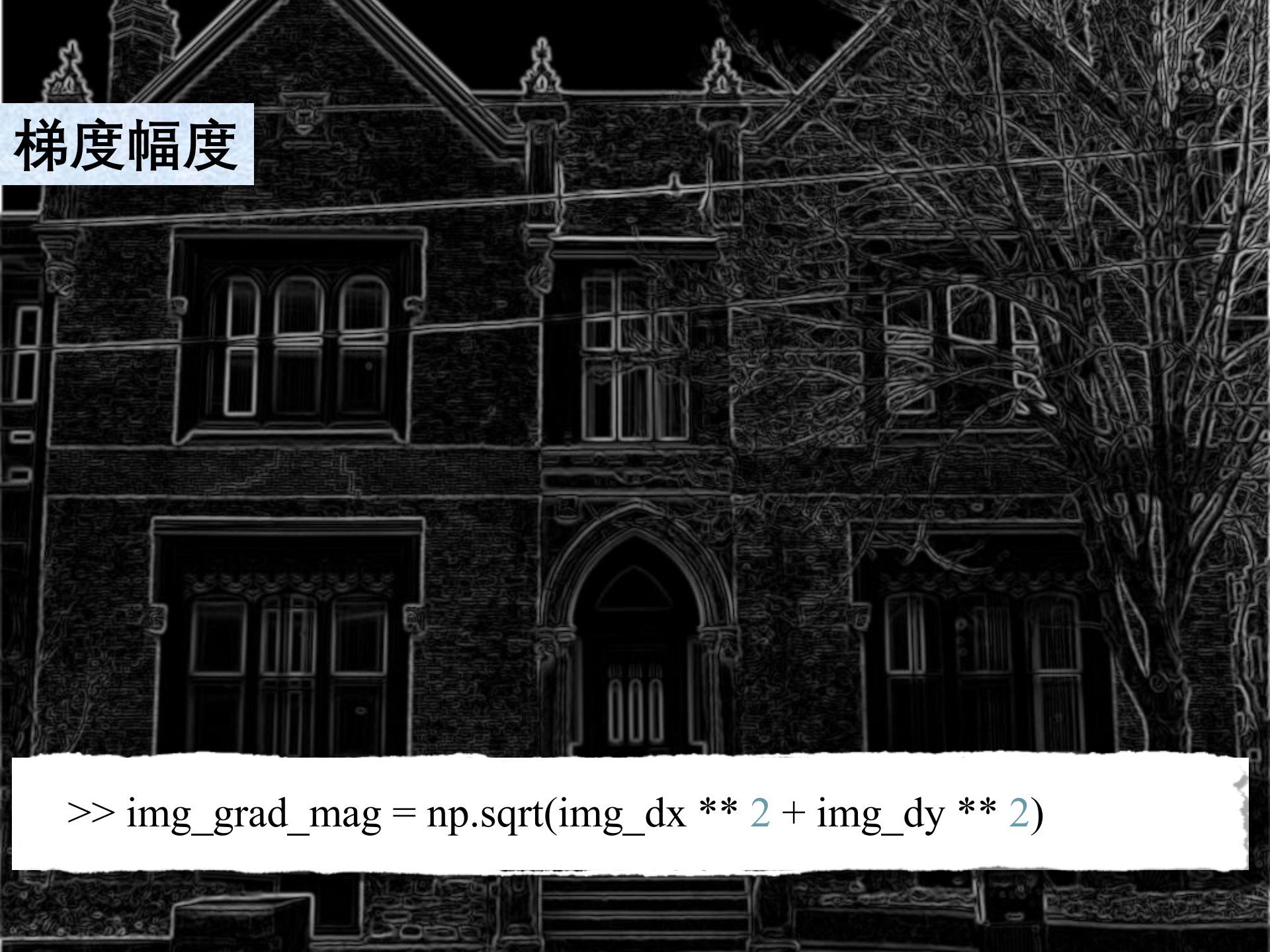
步骤2

求梯度的幅度和方向

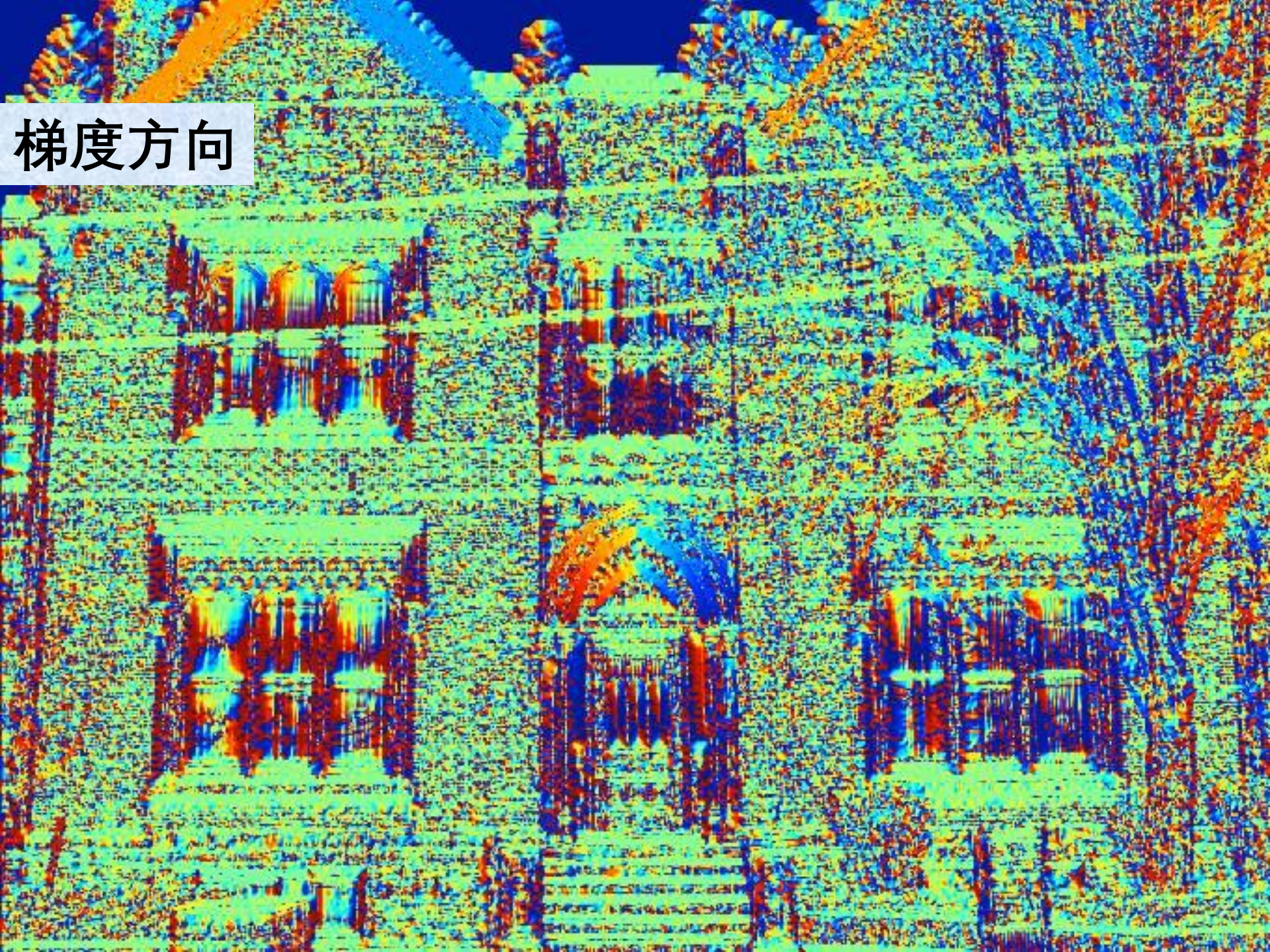

输入图像

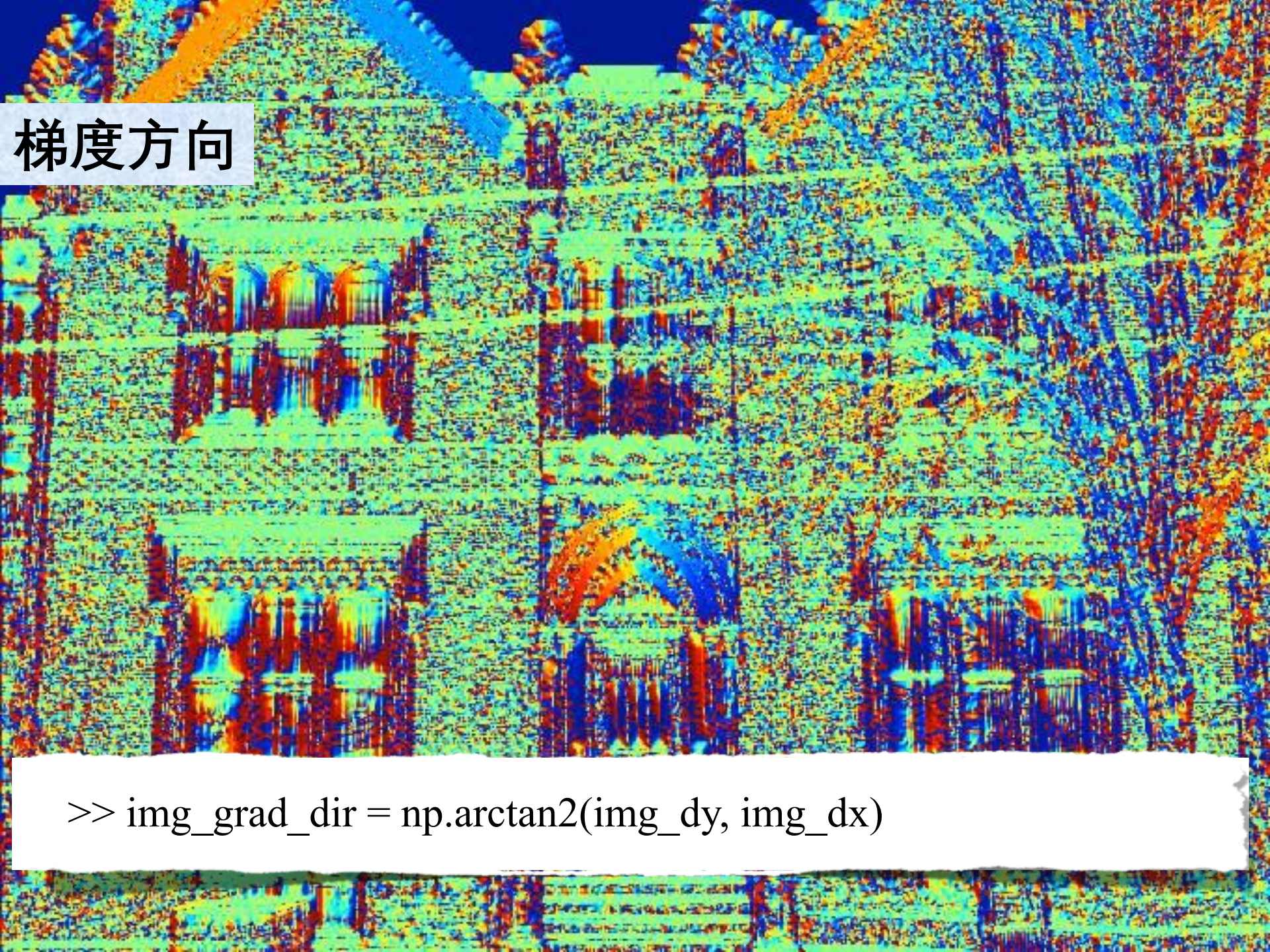

梯度幅度

梯度幅度



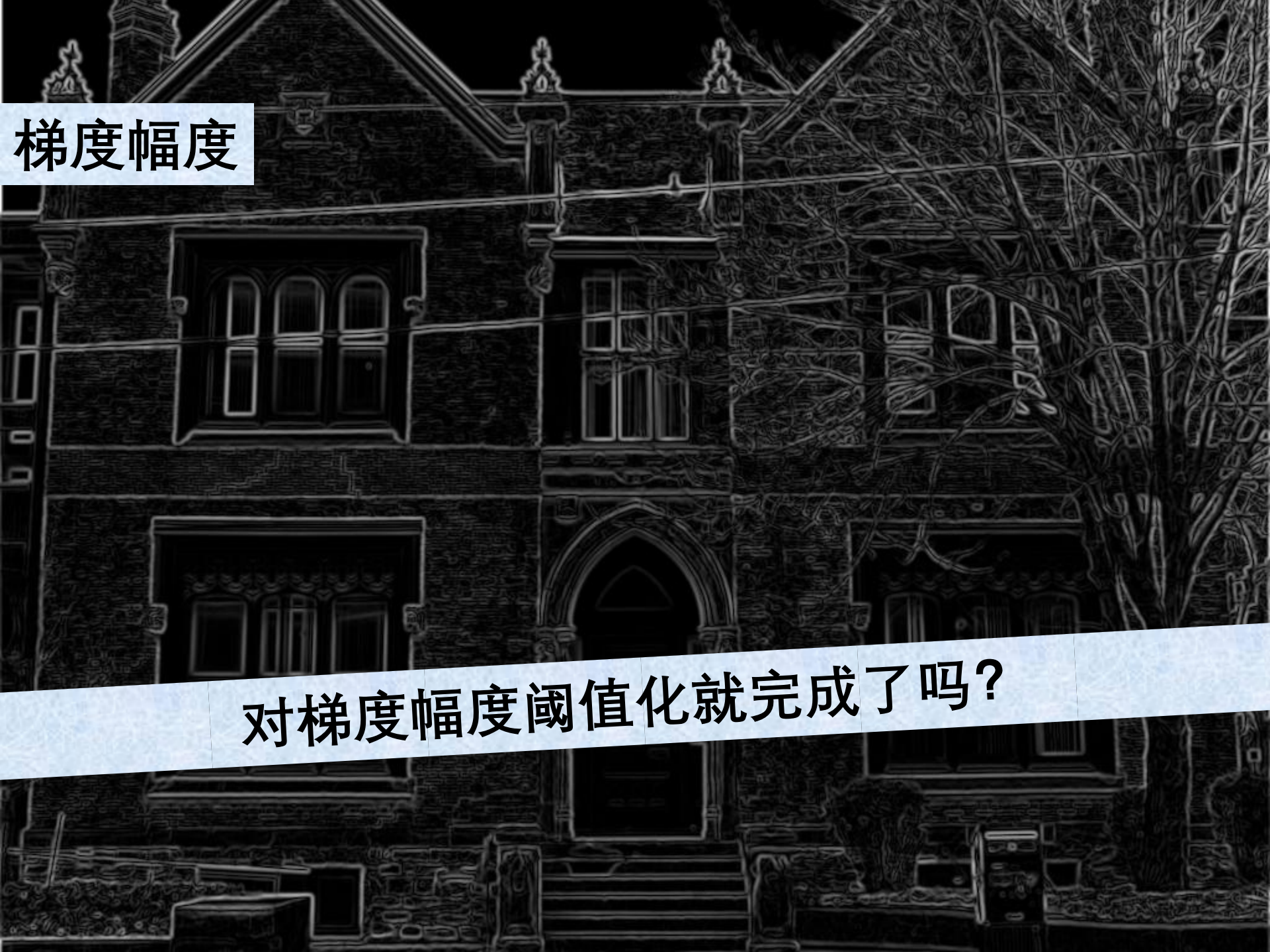
```
>> img_grad_mag = np.sqrt(img_dx ** 2 + img_dy ** 2)
```


梯度方向




```
>> img_grad_dir = np.arctan2(img_dy, img_dx)
```


梯度幅度


梯度幅度
对梯度幅度阈值化就完成了吗？

梯度幅度

```
>> img_thresh = im_grad_mag > 15
```


梯度幅度阈值

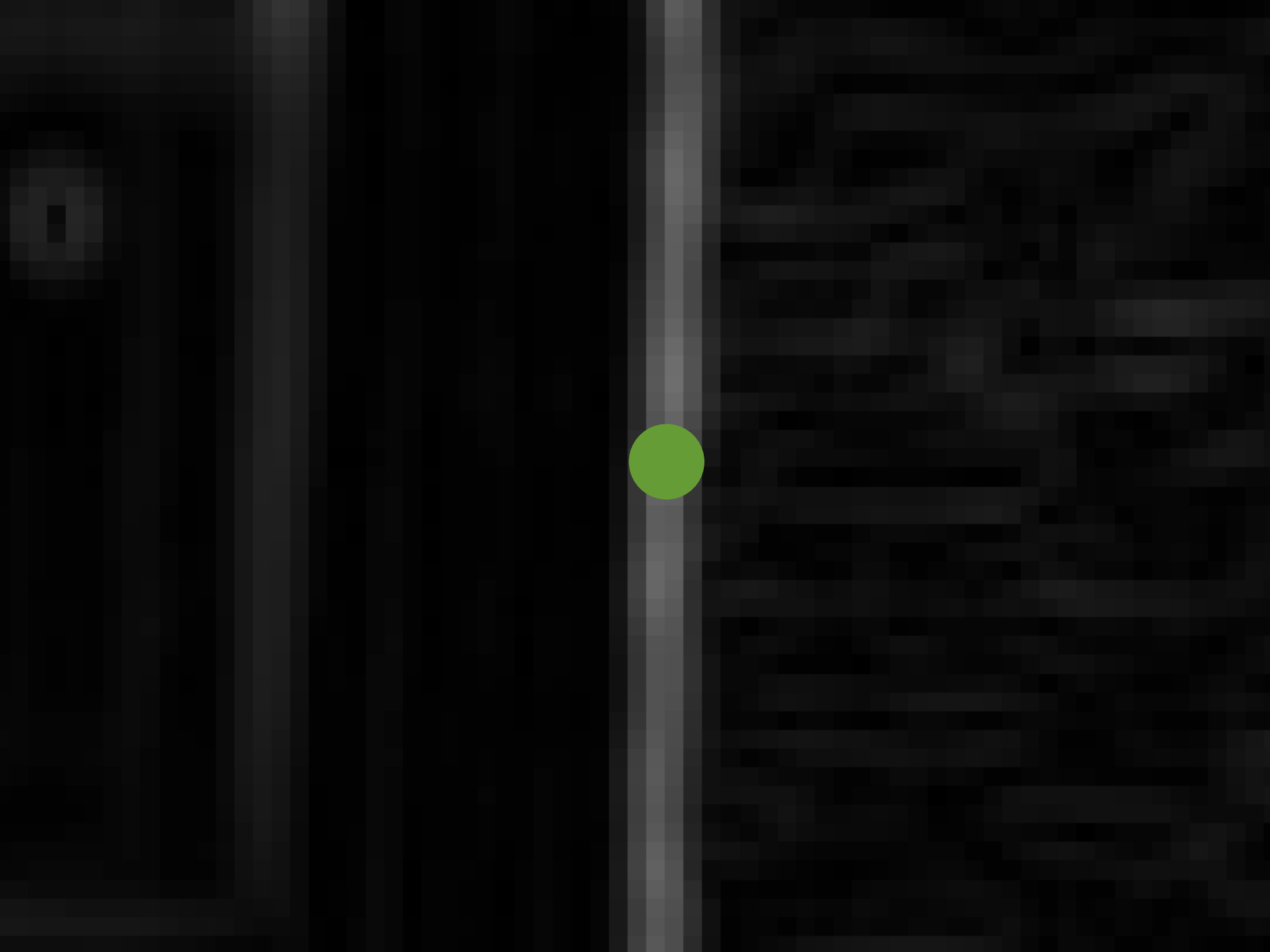

梯度幅度阈值
问题：导致边缘变粗

步骤3

执行非极大值抑制


梯度幅度


梯度幅度



梯度幅度是否大于沿梯度方向的相邻值？

梯度幅度是否大于沿梯度方向的相邻值？

梯度幅度是否大于沿梯度方向的相邻值？

梯度幅度是否大于沿梯度方向的相邻值？

如果是，保留，否则抑制


梯度幅度


非极大值抑制

步骤4

阈值化并连接

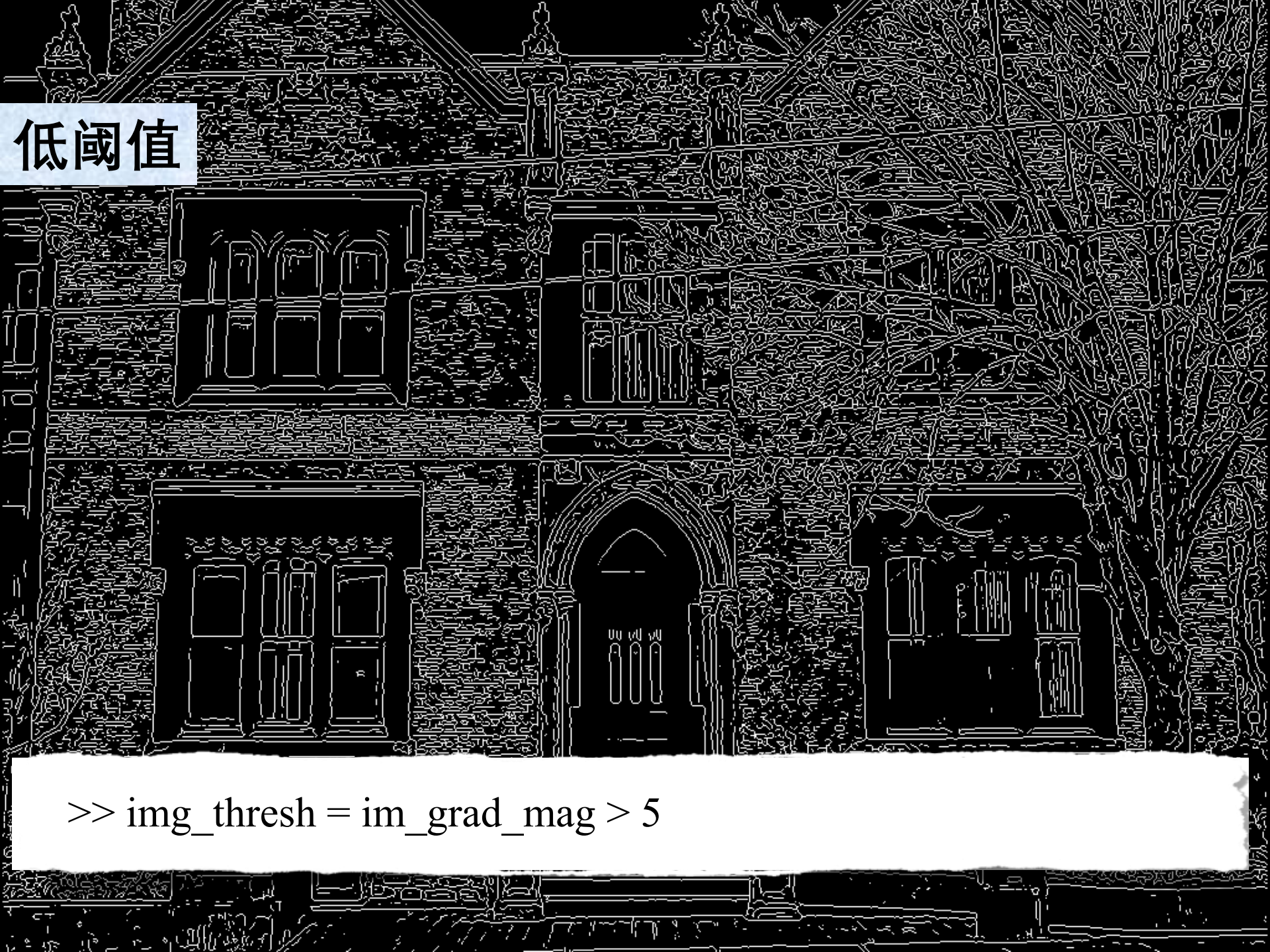
步骤4

阈值化并连接

滞后阈值法

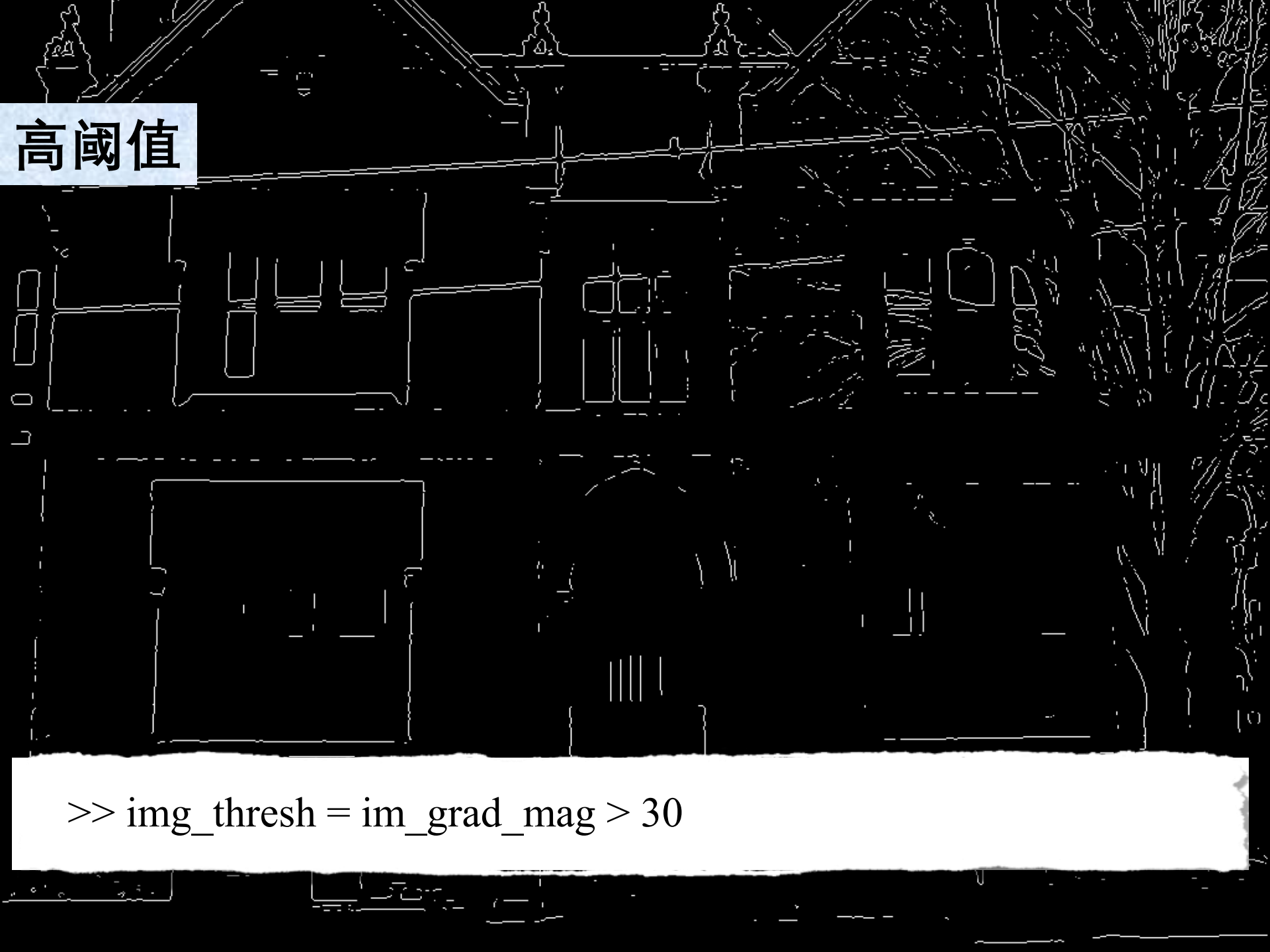

非极大值抑制


低阈值




```
>> img_thresh = im_grad_mag > 5
```


高阈值



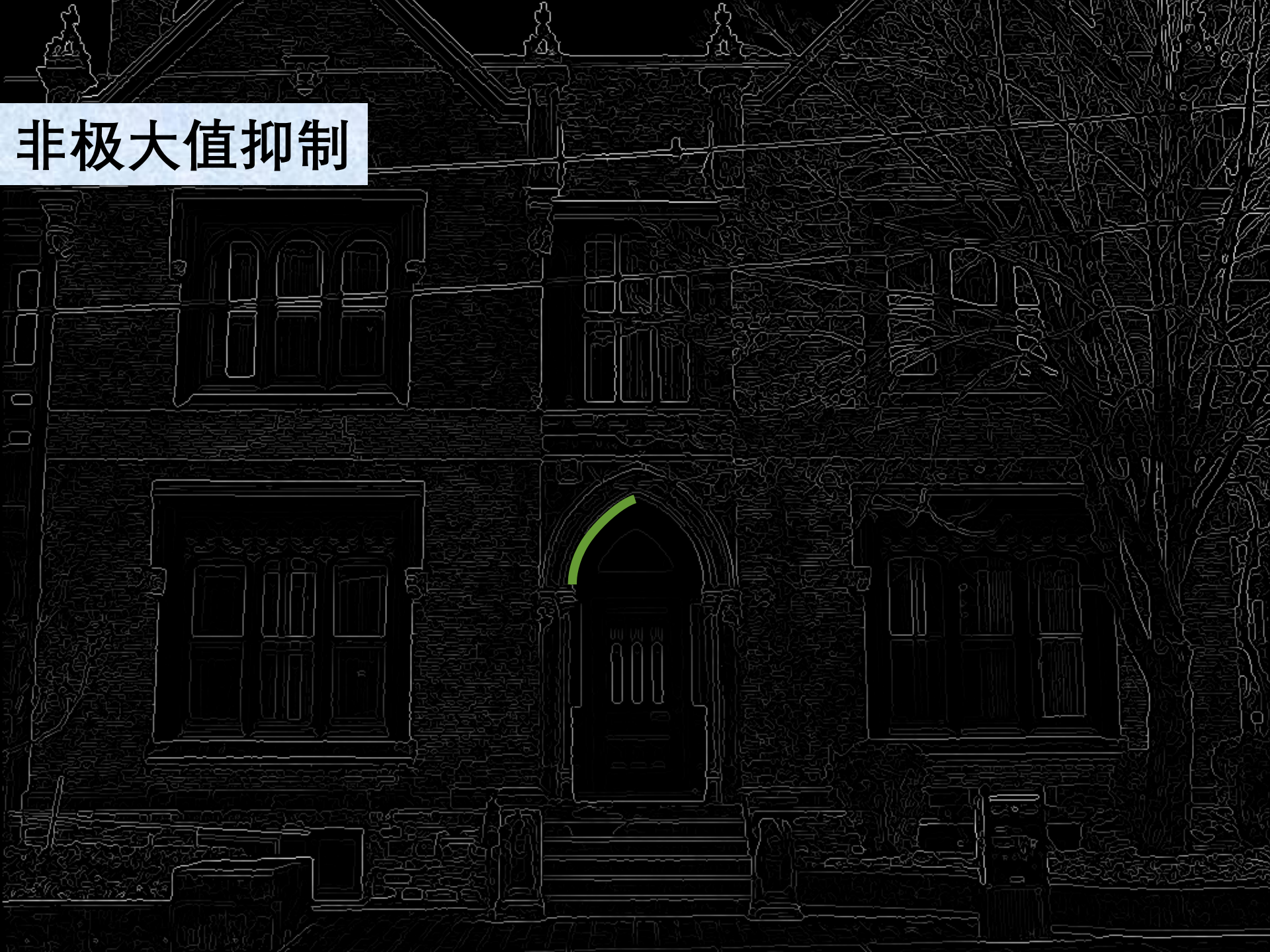

```
>> img_thresh = im_grad_mag > 30
```

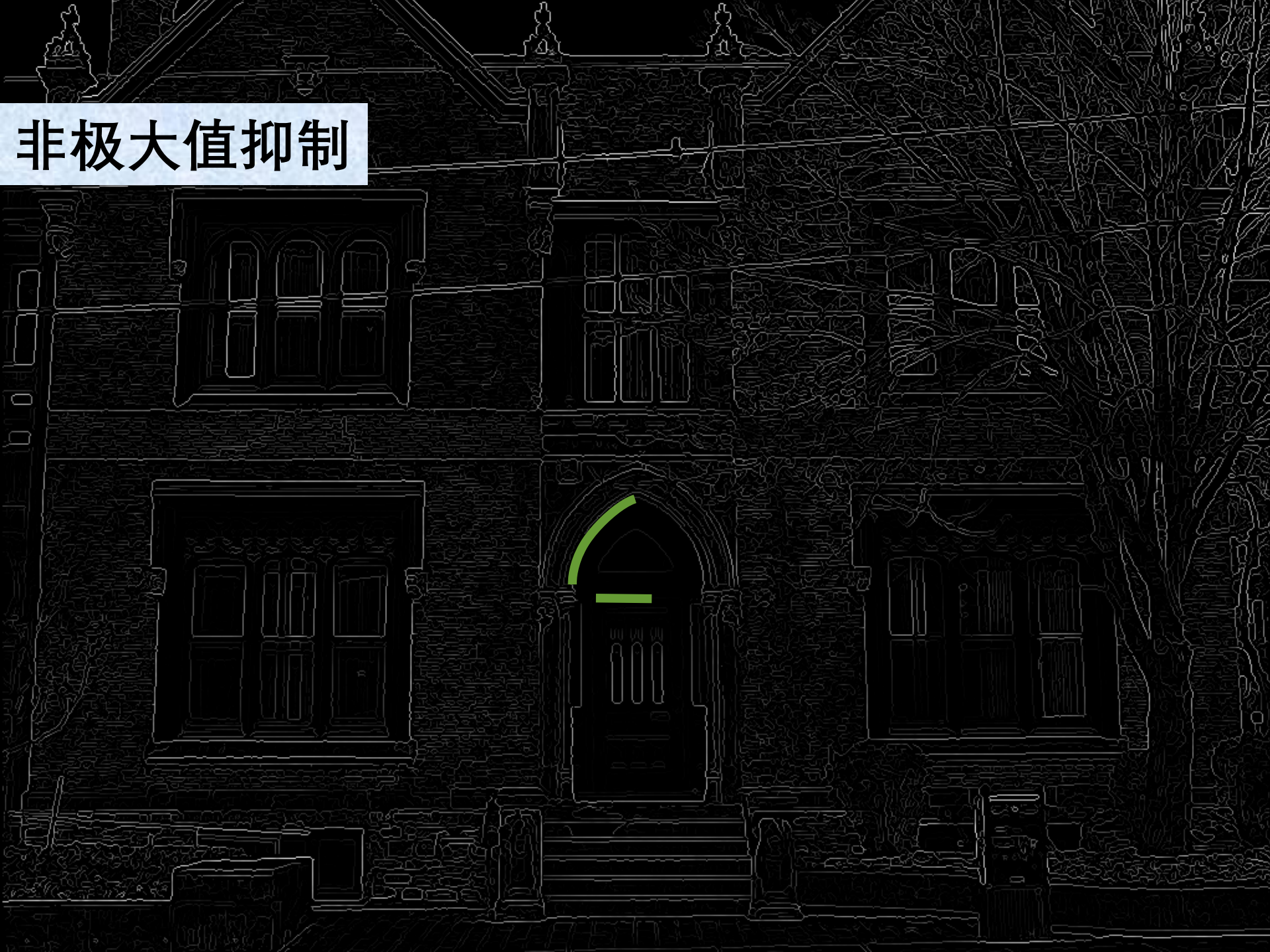
高阈值

低阈值

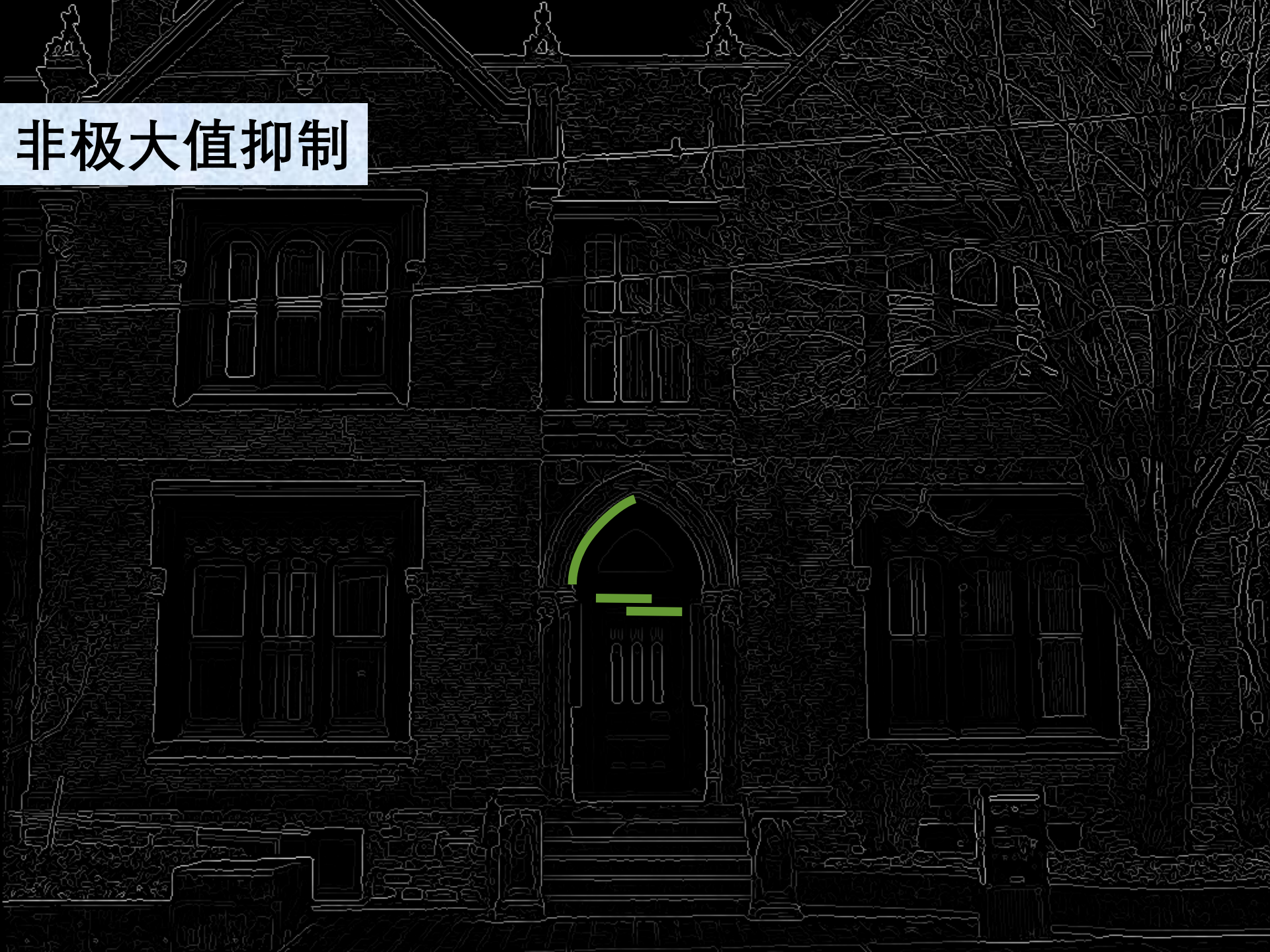

非极大值抑制

非极大值抑制




非极大值抑制

非极大值抑制



定义两个阈值，一个低一个高

定义两个阈值，一个低一个高

从通过高阈值的像素开始，只保留边缘路径(**垂直梯度方向**)上通过低阈值的像素

从通过高阈值的像素开始，只保留边缘路径(**垂直梯度方向**)上通过低阈值的像素

从通过高阈值的像素开始，只保留边缘路径(**垂直梯度方向**)上通过低阈值的像素

从通过高阈值的像素开始，只保留边缘路径(**垂直梯度方向**)上通过低阈值的像素


非极大值抑制


滞后阈值法

太长不看?

Canny算子

1. 使用x、 y高斯一阶导数对图像滤波

Canny算子

1. 使用x、 y高斯一阶导数对图像滤波

2. 求梯度的幅度和方向

Canny算子

1. 使用x、 y高斯一阶导数对图像滤波

2. 求梯度的幅度和方向

3. 执行非极大值抑制

Canny算子

1. 使用x、 y高斯一阶导数对图像滤波

2. 求梯度的幅度和方向

3. 执行非极大值抑制

4. 阈值化并连接

Python时间

```
>>> im_blur = cv2.GaussianBlur(im, (5, 5), cv2.BORDER_DEFAULT)
>>> img_edge = cv2.Canny(im_blur, threshold1=20, threshold2=45)
>>> cv2.imshow('Canny edge', img_edge), cv2.waitKey(0)
```

```
>>> im_blur = cv2.GaussianBlur(im, (5, 5), cv2.BORDER_DEFAULT)
>>> img_edge = cv2.Canny(im_blur, threshold1=20, threshold2=45)
>>> cv2.imshow('Canny edge', img_edge), cv2.waitKey(0)
```

```
>>> im_blur = cv2.GaussianBlur(im, (5, 5), cv2.BORDER_DEFAULT)
>>> img_edge = cv2.Canny(im_blur, threshold1=20, threshold2=45)
>>> cv2.imshow('Canny edge', img_edge), cv2.waitKey(0)
```

```
>>> im_blur = cv2.GaussianBlur(im, (5, 5), cv2.BORDER_DEFAULT)
>>> img_edge = cv2.Canny(im_blur, threshold1=20, threshold2=45)
>>> cv2.imshow('Canny edge', img_edge), cv2.waitKey(0)
```

```
>>> im_blur = cv2.GaussianBlur(im, (5, 5), cv2.BORDER_DEFAULT)
>>> img_edge = cv2.Canny(im_blur, threshold1=20, threshold2=45)
>>> cv2.imshow('Canny edge', img_edge), cv2.waitKey(0)
```

Python时间

如何确定正确的

尺度?


尺度?

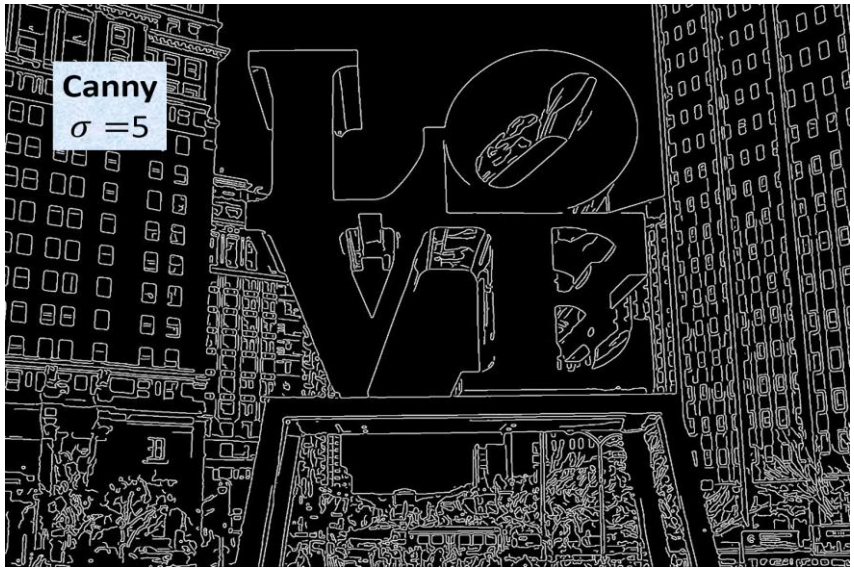

输入

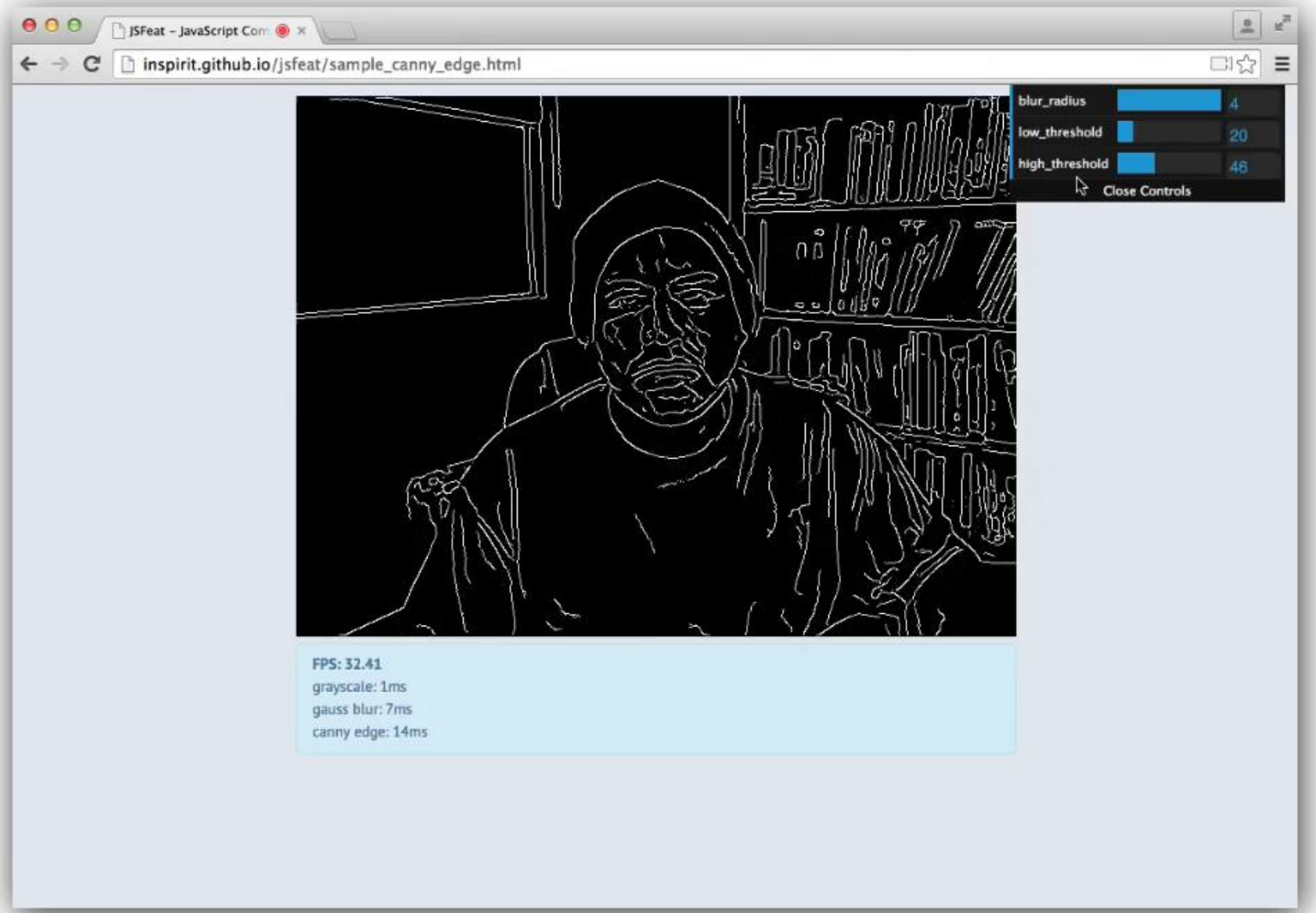

Canny
$\sigma = 1$


Canny
σ =5

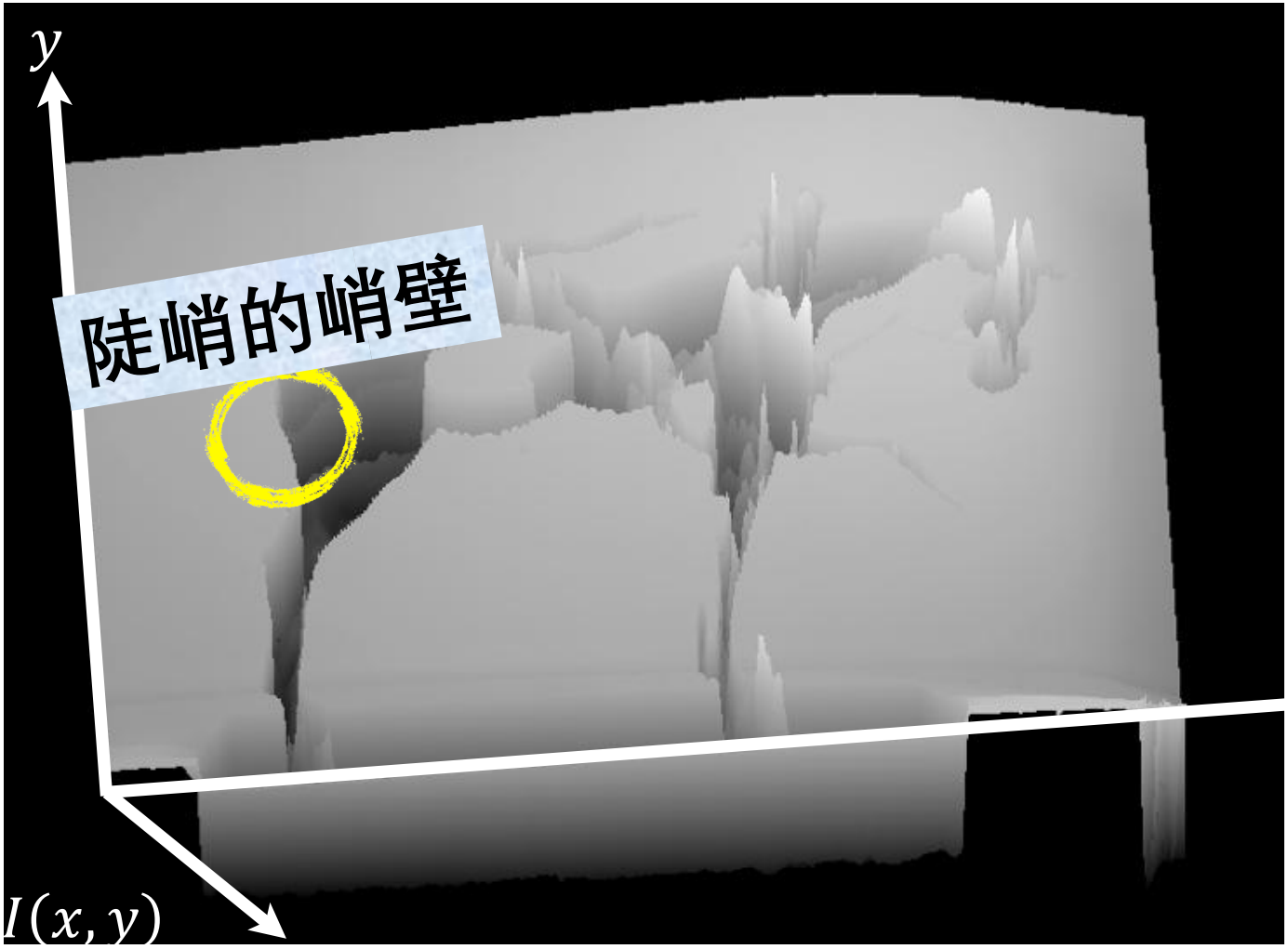

输入


Canny
σ =1


Canny
σ =5


JSFeat - JavaScript Com
inspirit.github.io/jsfeat/sample_canny_edge.html
blur_radius
4
low_threshold
20
high_threshold
46
Close Controls
FPS: 32.41
grayscale: 1ms
gauss blur: 7ms
canny edge: 14ms

还有一件事…



目标： 识别图像中的突然局部变化


检测颜色变换?

检测纹理变换？

如何结合各种线索来检测物体轮廓?

机器学习

有监督的学习

有监督的学习

$\mathbf{x}$

特征

有监督的学习


x
特征
f
y
类别标签
给定训练数据，估计函数f

给定训练数据，估计函数f

$$\{(\mathbf{x}_i, y_i)\}$$

给定训练数据，估计函数f

$$\{(\mathbf{x}_i, y_i)\}$$

Learning to Detect Natural Image Boundaries Using Local Brightness, Color, and Texture Cues

David R. Martin, *Member*, *IEEE*, Charless C. Fowlkes, and Jitendra Malik, *Member*, *IEEE*

Abstract—The goal of this work is to accurately detect and localize boundaries in natural scenes using local image measurements. We formulate features that respond to characteristic changes in brightness, color, and texture associated with natural boundaries. In

Structured Forests for Fast Edge Detection

Piotr Dollár
Microsoft Research
pdollar@microsoft.com

C. Lawrence Zitnick
Microsoft Research
larryz@microsoft.com

Abstract

Edge detection is a critical component of many vision systems, including object detectors and image segmentation algorithms. Patches of edges exhibit well-known forms of local structure, such as straight lines or T-junctions. In this paper we take advantage of the structure present in local

手工打造的算法

深度

学习


边界
检测器

PUSHING THE BOUNDARIES OF BOUNDARY DETECTION USING DEEP LEARNING

Iasonas Kokkinos
Center for Visual Computing
CentraleSupélec and INRIA
Chatenay-Malabry, 92095, France
{iasonas.kokkinos}@ecp.fr

ABSTRACT

In this work we show that adapting Deep Convolutional Neural Network training to the task of boundary detection can result in substantial improvements over the current state-of-the-art in boundary detection

PUSHING THE BOUNDARIES OF BOUNDARY DETECTION USING DEEP LEARNING

Iasonas Kokkinos
Center for Visual Computing
CentraleSupélec and INRIA
Chatenay-Malabry, 92095, France
{iasonas.kokkinos}@ecp.fr

ABSTRACT

In this work we show that adapting Deep Convolutional Neural Network training
[illegible] can result in substantial improvements over the

在挑战性基准上超越了人类的准确性！