

# 机器视觉技术与应用

## 0. 导论

李竹

杭州电子科技大学  
电子信息学院



“机器视觉”，即采用机器代替人眼来做测量和判断。

Input	Output	Digital image	Knowledge
Digital image	Digital Image Processing		Computer Vision
Knowledge	Computer Graphics		Artificial Intelligence

Input	Output	Digital image	Knowledge
Digital image	Digital Image Processing	Computer Vision	
Knowledge	Computer Graphics	Artificial Intelligence	

## 数字图像处理

图像增强

图像分割

图像压缩

图像变换

## 计算机视觉

图像识别

目标检测

图像检索

图像理解

## 计算机图形学

工业设计

特效动画

电子游戏

虚拟现实

## 人 工 智 能

信息挖掘

控制决策

语言翻译

语言理解

Input	Output	Digital image	Knowledge
Digital image	Digital Image Processing		Computer Vision
Knowledge	Computer Graphics		Artificial Intelligence

## 数字图像处理

- 图像增强
- 图像分割
- 图像压缩
- 图像变换

## 计算机视觉

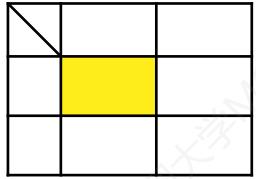
- 图像识别
- 目标检测
- 图像检索
- 图像理解

## 计算机图形学

- 工业设计
- 特效动画
- 电子游戏
- 虚拟现实

## 人 工 智 能

- 信息挖掘
- 控制决策
- 语言翻译
- 语言理解



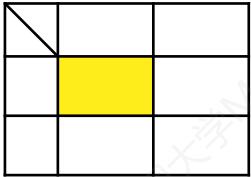
## 数字图像处理

图像增强

图像分割

图像压缩

图像变换



定义：强调图像的整体或局部特性，加强判读和识别效果，满足特殊分析需要。

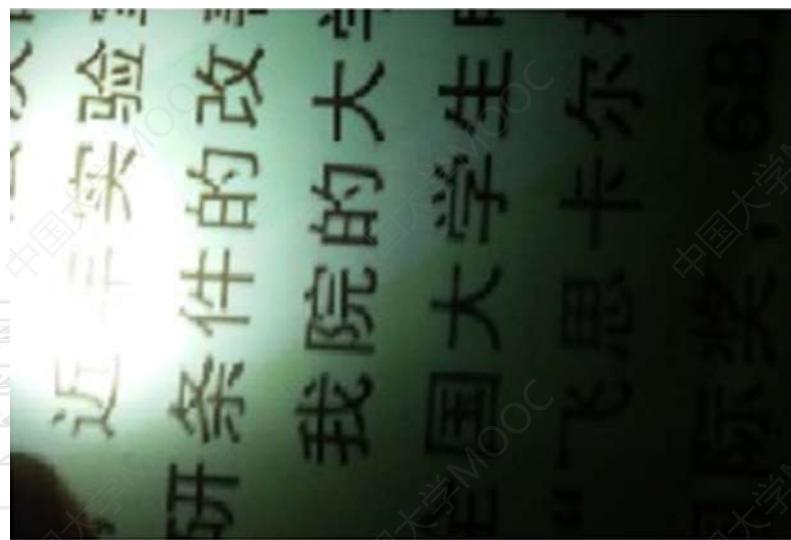
数字图像处理

图像增强

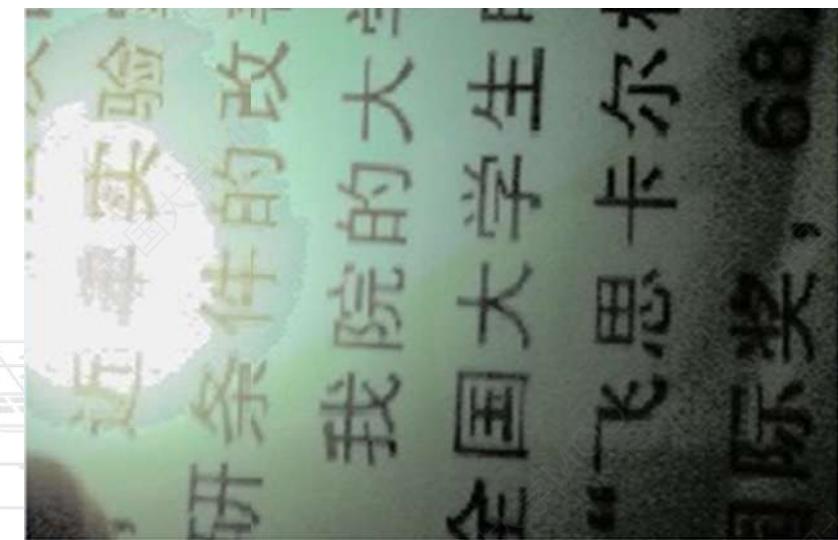
图像分割

图像压缩

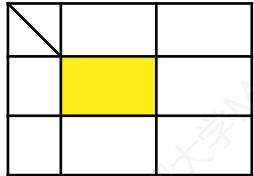
图像变换



原图



增强后



定义：强调图像的整体或局部特性，加强判读和识别效果，满足特殊分析需要。

## 数字图像处理

图像增强

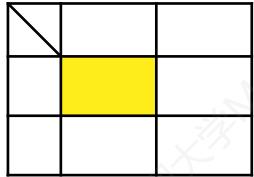
图像分割

图像压缩

图像变换



He K, Jian S, Fellow, et al. Single Image Haze Removal Using Dark Channel Prior[J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2011, 33(12):2341-2353.



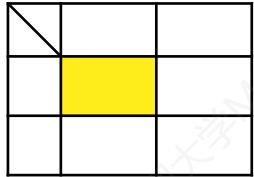
## 数字图像处理

图像增强

图像分割

图像压缩

图像变换



数字图像处理

图像增强

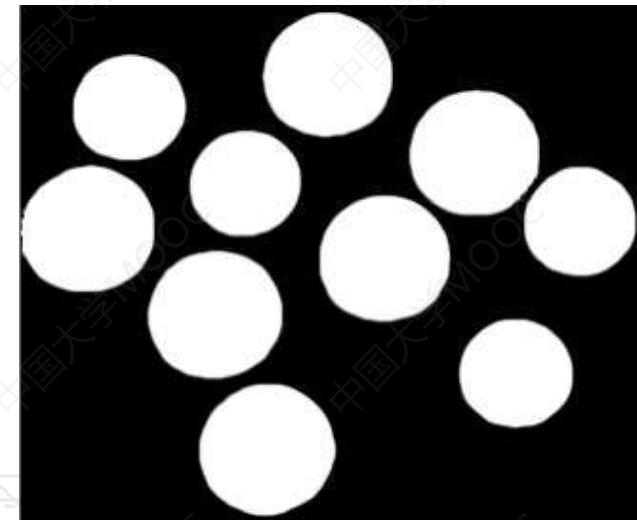
图像分割

图像压缩

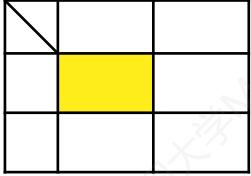
图像变换



原图



二值图像



数字图像处理

图像增强

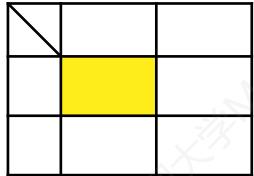
图像分割

图像压缩

图像变换

## 语意分割 Demo





数字图像处理

图像增强

图像分割

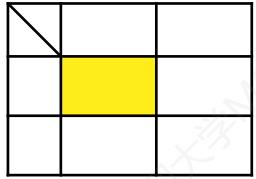
图像压缩

图像变换



## 智能驾驶中的语意分割

Zhao H , Qi X , Shen X , et al. ICNet for Real-Time Semantic Segmentation on High-Resolution Images[C]// 2017.



数字图像处理

图像增强

图像分割

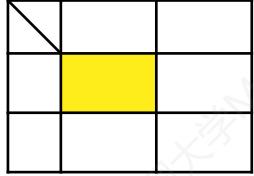
图像压缩

图像变换



256 K

21 K



数字图像处理

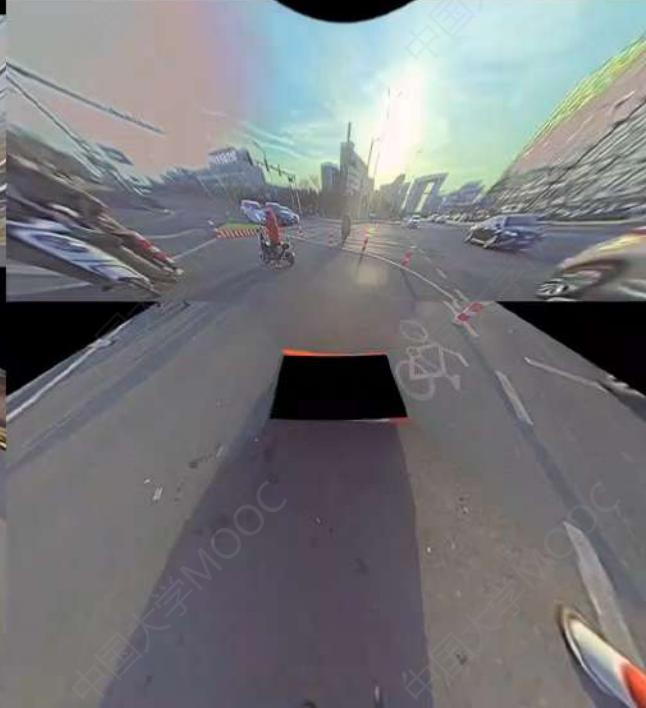
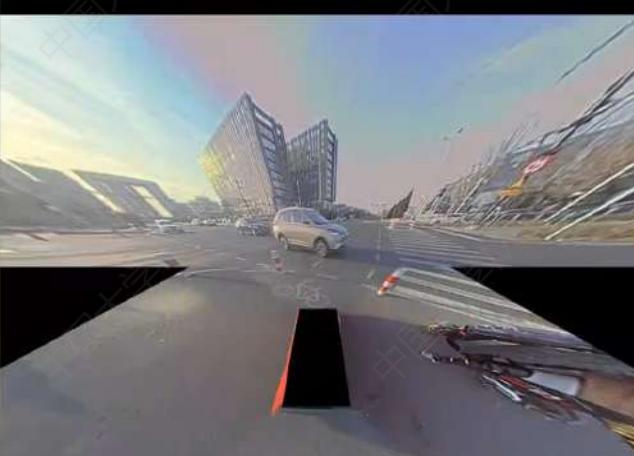
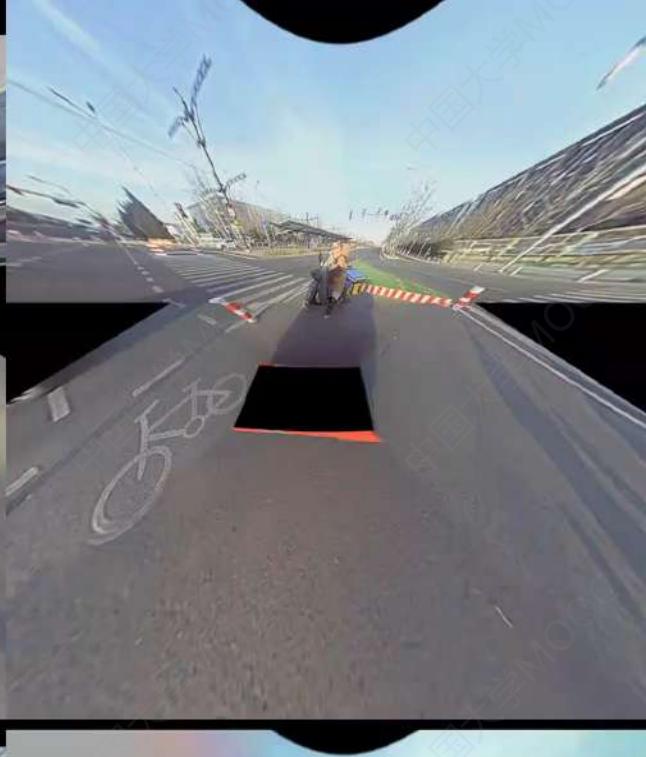
图像增强

图像分割

图像压缩

图像变换

## 几何变换 Demo

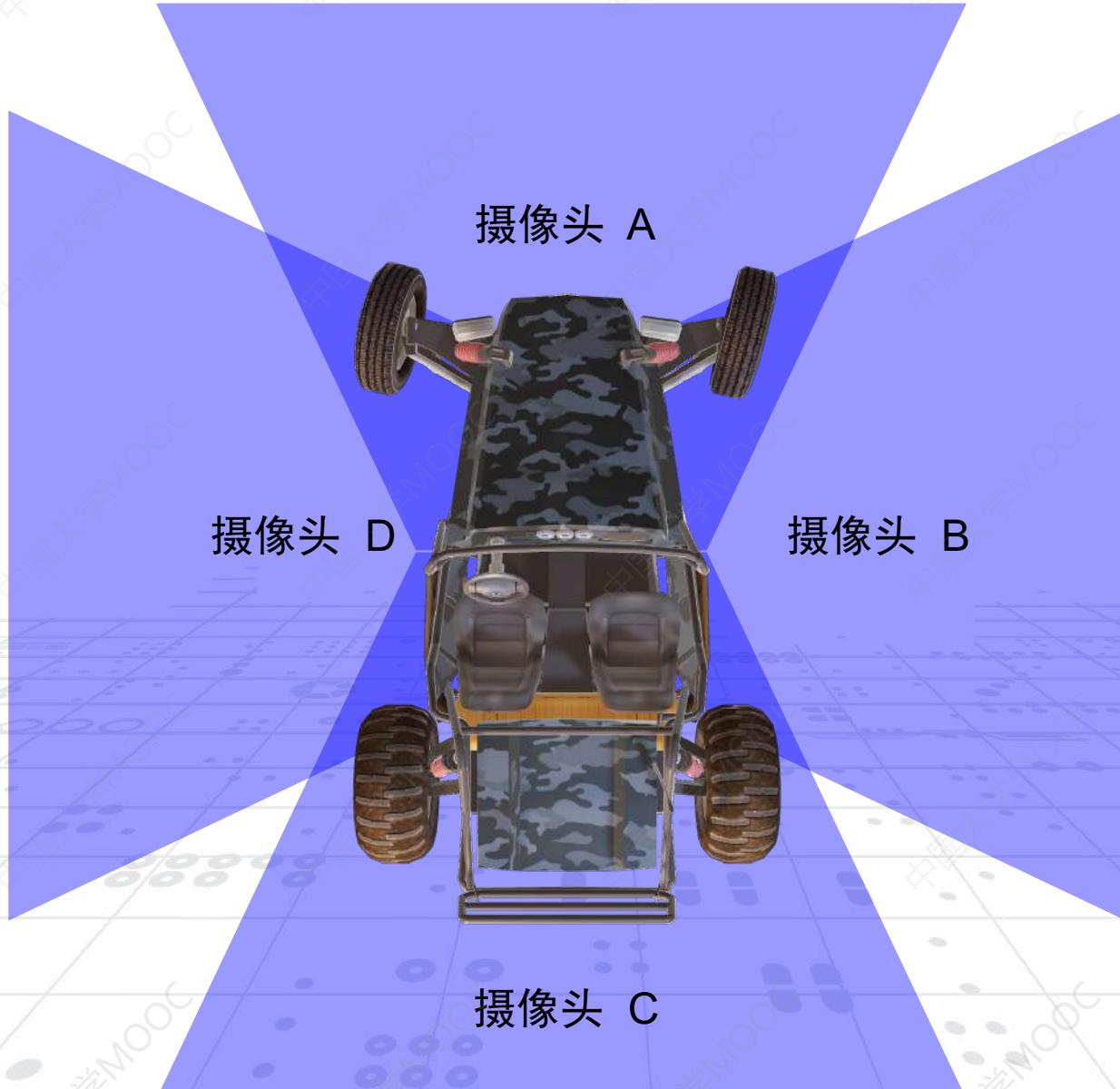


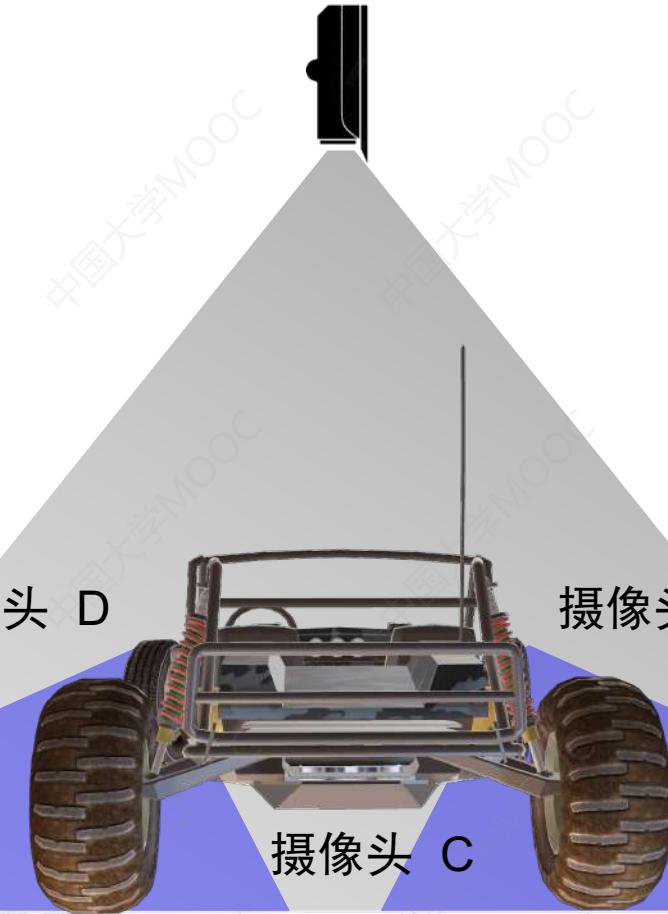
鸟瞰视角

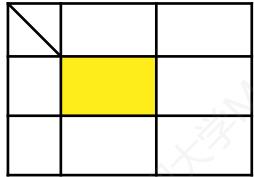




摄像头位置？？







## 数字图像处理

图像增强

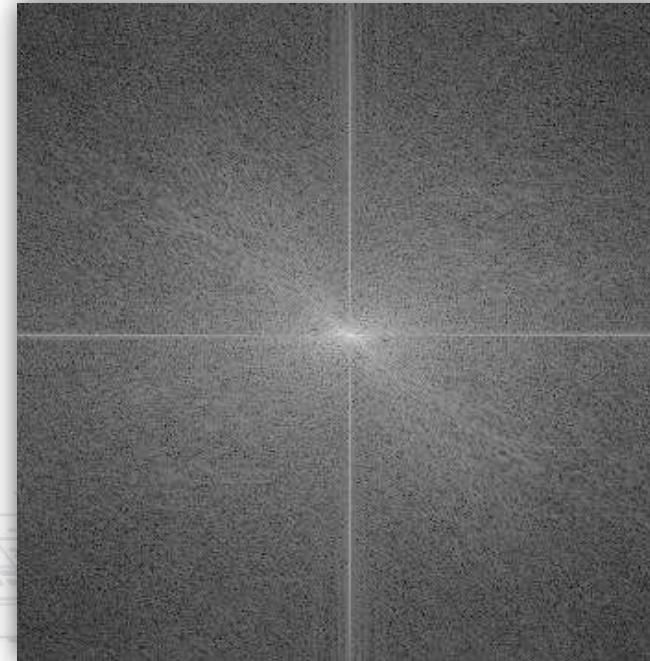
图像分割

图像压缩

图像变换



原图



频谱图

Input	Output	Digital image	Knowledge
Digital image	Digital Image Processing	Computer Vision	
Knowledge	Computer Graphics	Artificial Intelligence	

## 数字图像处理

图像增强  
图像分割  
图像压缩  
图像变换

## 计算机视觉

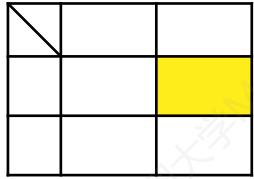
图像识别  
目标检测  
图像检索  
图像理解

## 计算机图形学

工业设计  
特效动画  
电子游戏  
虚拟现实

## 人 工 智 能

信息挖掘  
控制决策  
语言翻译  
语言理解



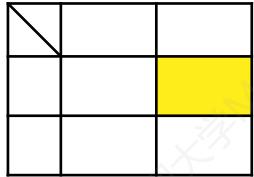
## 计算机视觉

图像识别

目标检测

图像检索

图像理解



## 计算机视觉

图像识别

目标检测

图像检索

图像理解

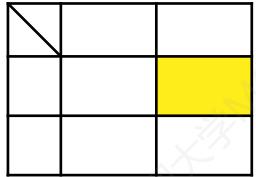


Input



西瓜

Output



## 计算机视觉

图像识别

目标检测

图像检索

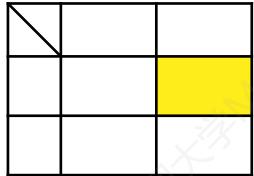
图像理解



Input



Output



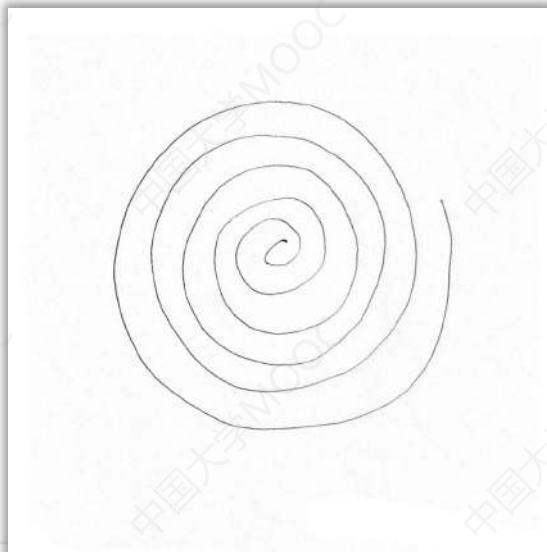
## 计算机视觉

图像识别

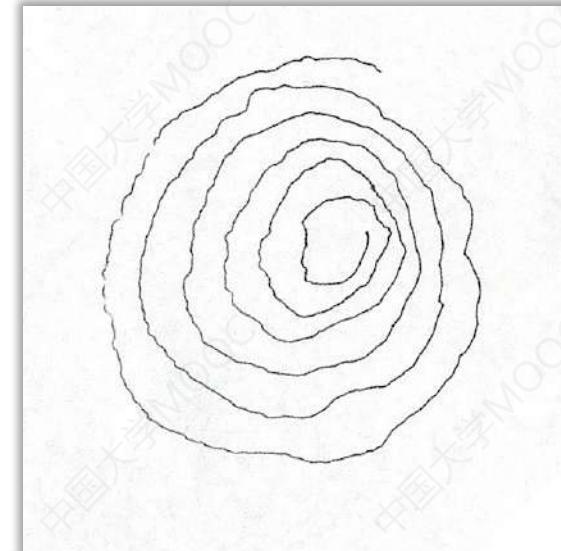
目标检测

图像检索

图像理解



普通人



PD患者

Zhu Li, Jiayu Yang, Yanwen Wang\*, Miao Cai, Xiaoli Liu, Kang Lu; Early diagnosis of Parkinson's disease using Continuous Convolution Network: Handwriting recognition based on off-line hand drawing without template. Journal of Biomedical Informatics. 2022 April 29.

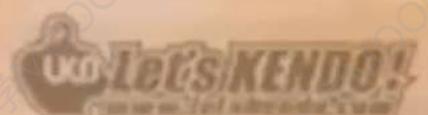


前田康喜 × 田中晃司

大阪

先鋒

神奈川





C: \ Users \ YHzui \ Desktop \ pythonProject \ control.py

main ▾

draw.py control.py

```
1 import cv2
2 import mediapipe as mp
3 import cmath
4 import pyautogui as pg
5 import numpy as np
6
7 flag = 0
8 dra = 0
9
10 mp_drawing = mp.solutions.drawing_utils
11 mp_hands = mp.solutions.hands
12
13 hands = mp_hands.Hands(
14     static_image_mode=False,
15     max_num_hands=1,
16     min_detection_confidence=0.5,
17     min_tracking_confidence=0.5)
18
19 cap = cv2.VideoCapture(0)
20 x = []
21 y = []
22 fl = []
```

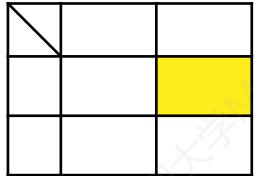
终端 Local Local +

```
Frames per second using video.get(cv2.CAP_PROP_FPS) : 36.0
480 640 3
Frames per second using video.get(cv2.CAP_PROP_FPS) : 36.0
480 640 3
Frames per second using video.get(cv2.CAP_PROP_FPS) : 36.0
[ WARN:0@35.551] global D:\a\opencv-python\opencv-python\modules\videoio\src\cap_msmf.cpp (539) `anonymous-namespace'::SourceReaderCB::~SourceReaderCB terminating async callback
PS C:\Users\YHzui\PycharmProjects\pythonProject> python draw.py
```

Version Control TODO 问题 Python Packages Python 控制台 终端

20:167 CRLF UTF-8 列 4个空格 Python 3.9 (pythonProject)

15:49  
2022/6/29



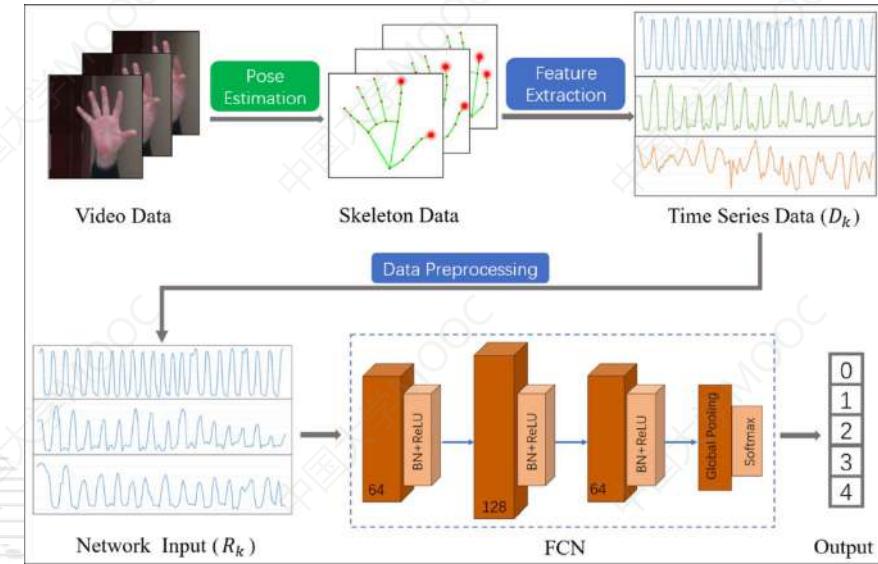
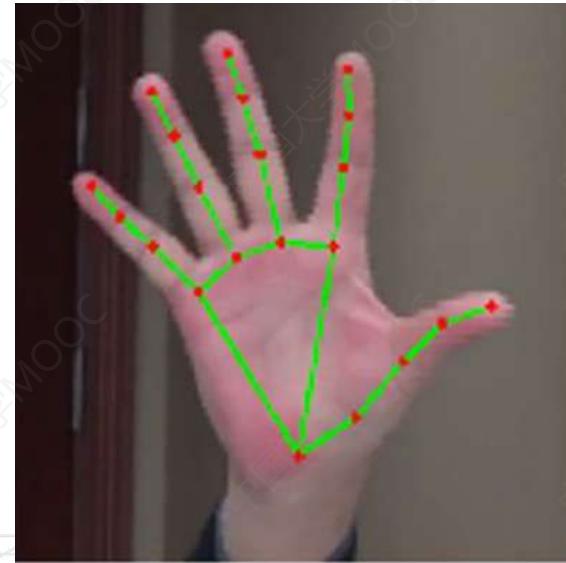
## 计算机视觉

图像识别

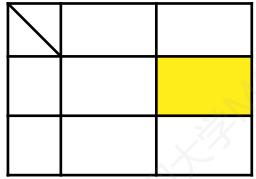
目标检测

图像检索

图像理解



Zhu Li, Kang Lu, Miao Cai, Xiaoli Liu, Yanwen Wang, Jiayu Yang; An Automatic Evaluation Method for Parkinson's Dyskinesia Using Finger Tapping Video for Small Samples. Journal of Medical and Biological Engineering. 2022 May 14.



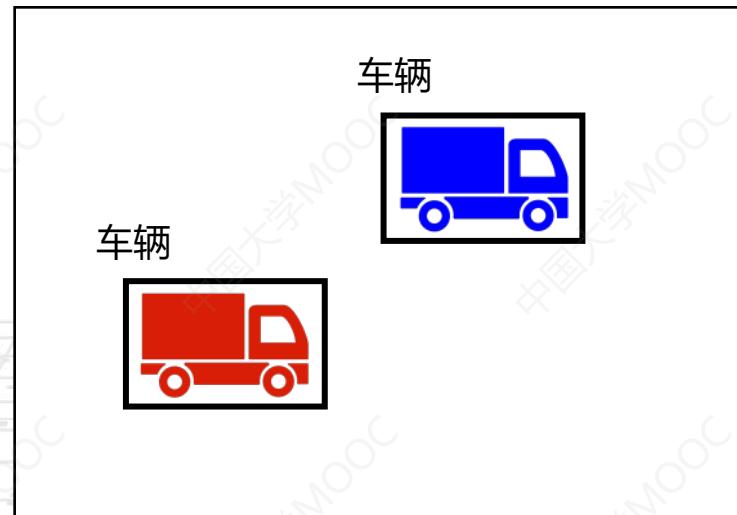
## 计算机视觉

图像识别

目标检测

图像检索

图像理解



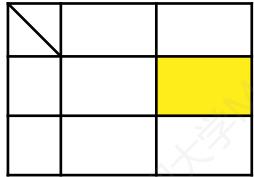
车辆



车辆



T 时刻



## 计算机视觉

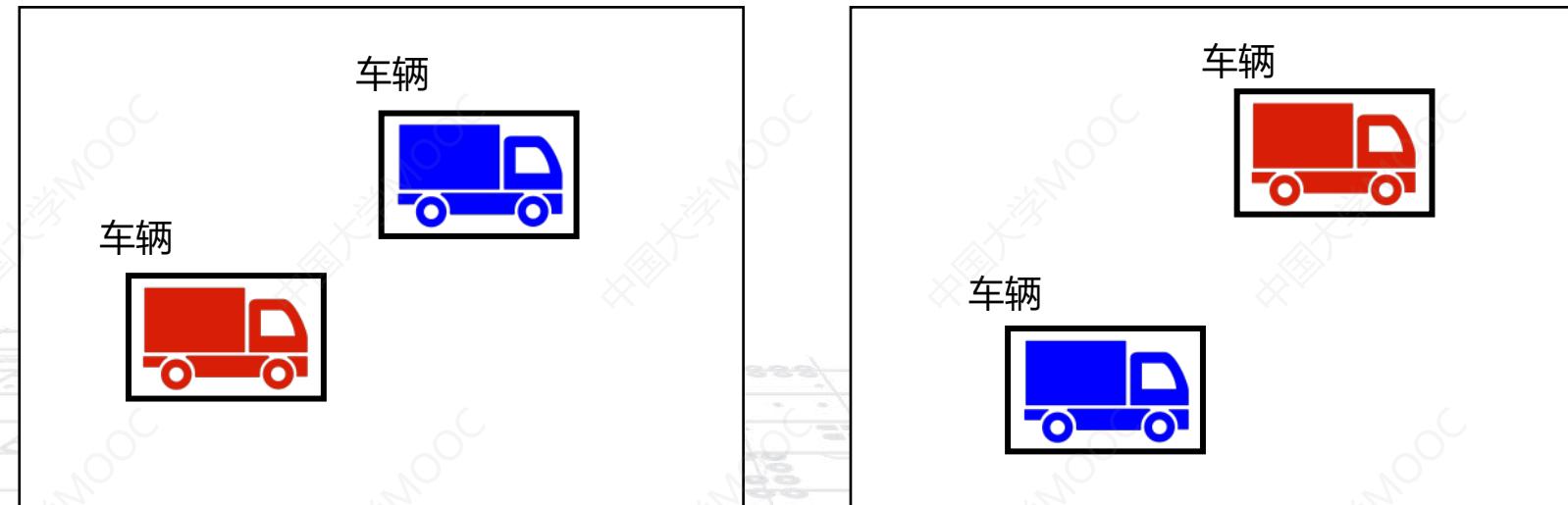
图像识别

目标检测

图像检索

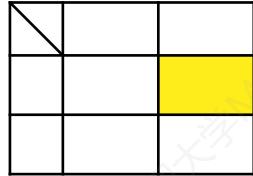
图像理解

# 检测 (Detection)



T 时刻

T+1 时刻



# 追踪 (Tracking)

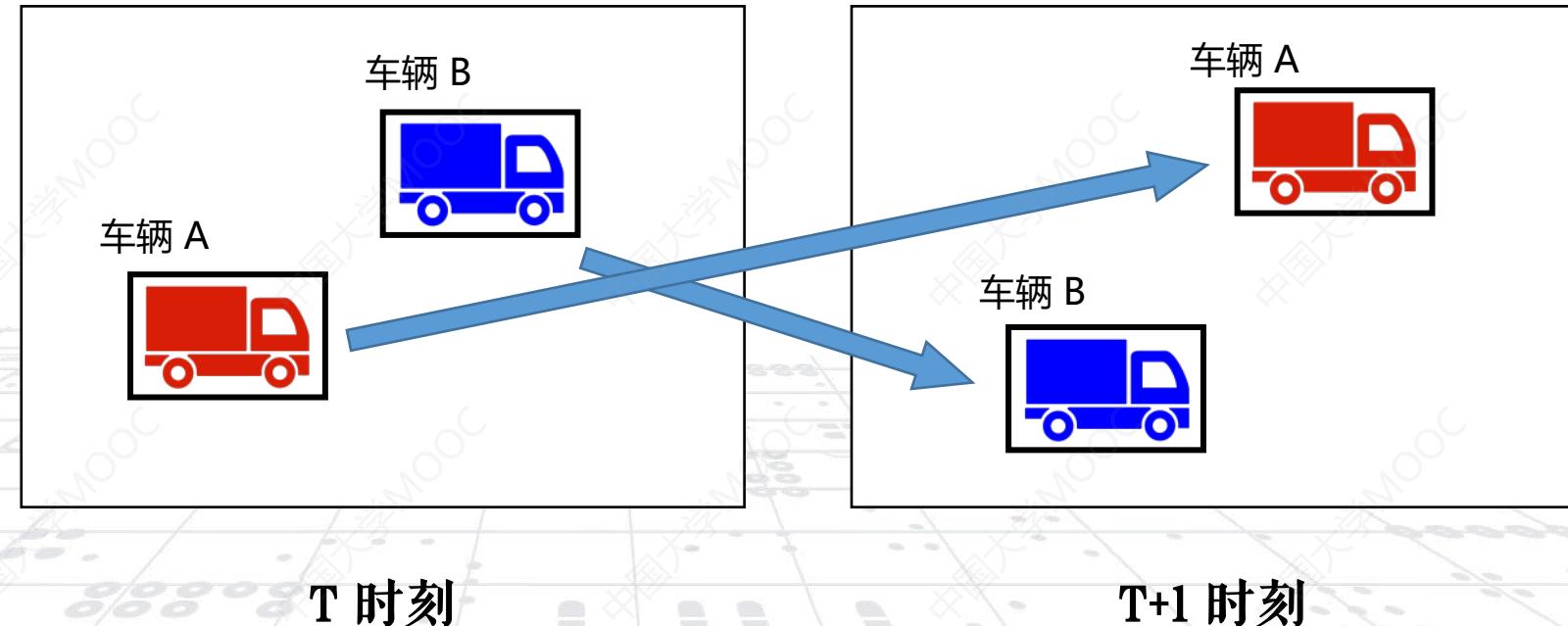
计算机视觉

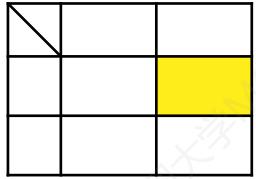
图像识别

目标检测

图像检索

图像理解





## 计算机视觉

图像识别

目标检测

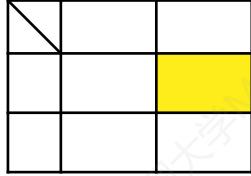
图像检索

图像理解

Yolov5  
Demo



豪门盛宴



## 计算机视觉

图像识别

目标检测

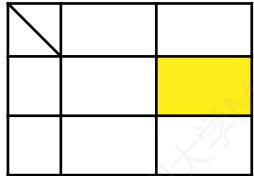
图像检索

图像理解

 淘 淘宝，以图搜产品；



百度，以图搜图。



## 计算机视觉

图像识别

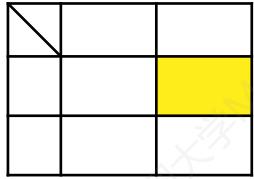
目标检测

图像检索

图像理解



Karpathy A , Fei-Fei L . Deep Visual-Semantic Alignments for Generating Image Descriptions[J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2016:664-676.



## 计算机视觉

图像识别

目标检测

图像检索

图像理解

Input	Output	Digital image	Knowledge
Digital image	Digital Image Processing	Computer Vision	
Knowledge	Computer Graphics	Artificial Intelligence	

## 数字图像处理

图像增强

图像分割

图像压缩

图像变换

## 计算机视觉

图像识别

目标检测

图像检索

图像理解

## 计算机图形学

工业设计

特效动画

电子游戏

虚拟现实

## 人 工 智 能

信息挖掘

控制决策

语言翻译

语言理解

Input	Output	Digital image	Knowledge
Digital image	Digital Image Processing		Computer Vision
Knowledge	Computer Graphics		Artificial Intelligence

## 数字图像处理

图像增强  
图像分割  
图像压缩  
图像变换

## 计算机视觉

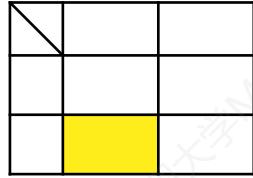
图像识别  
目标检测  
图像检索  
图像理解

## 计算机图形学

工业设计  
特效动画  
电子游戏  
虚拟现实

## 人 工 智 能

信息挖掘  
控制决策  
语言翻译  
语言理解



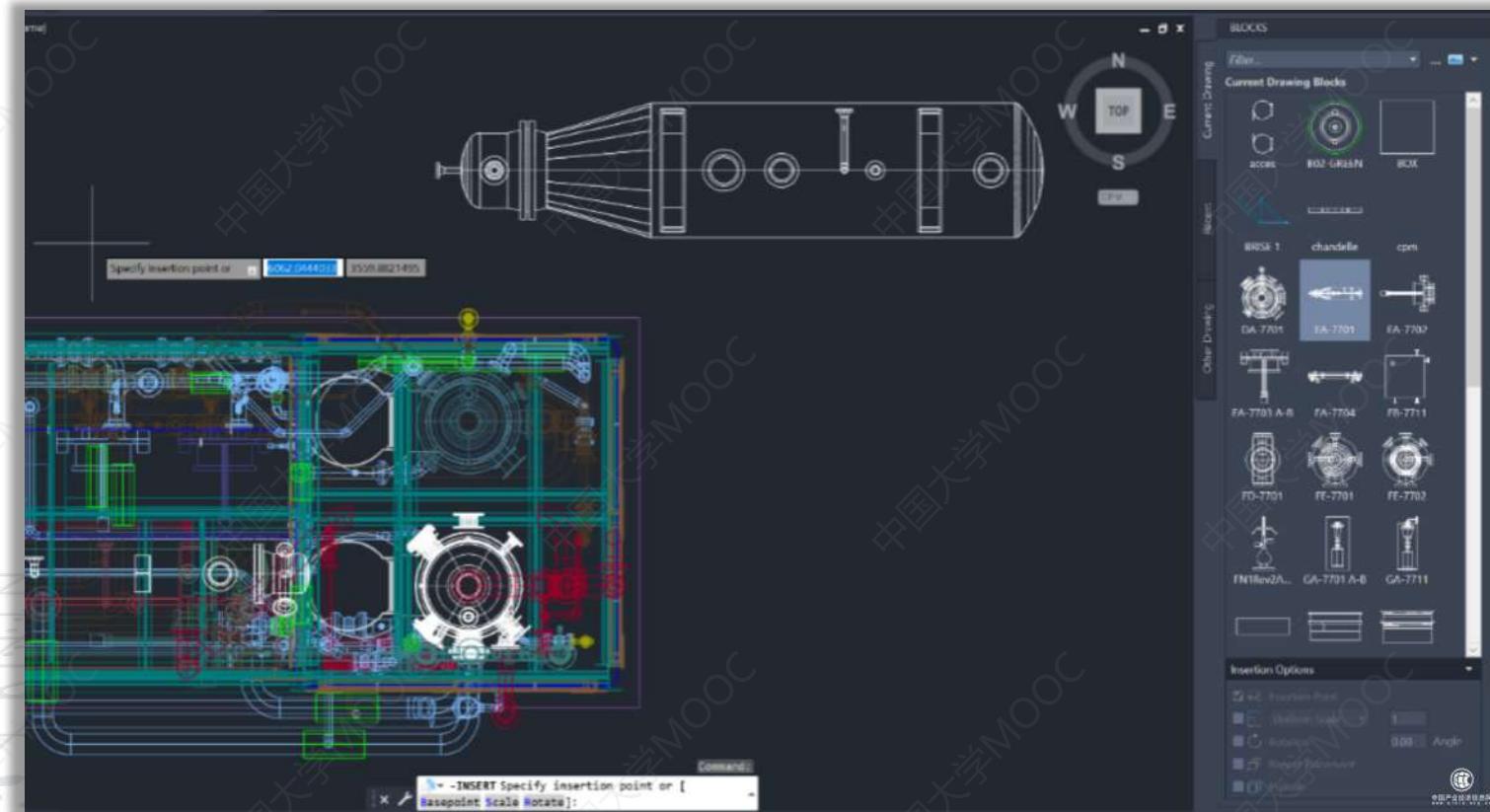
## 计算机图形学

工业设计

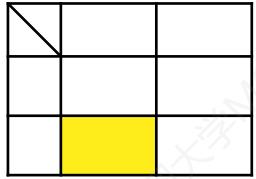
特效动画

电子游戏

虚拟现实



AutoCAD



## 计算机图形学

工业设计

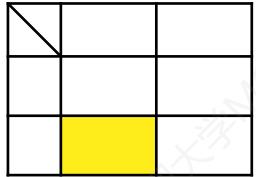
特效动画

电子游戏

虚拟现实



2007年，变形金刚



## 计算机图形学

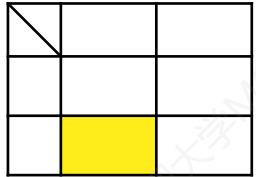
工业设计

特效动画

电子游戏

虚拟现实





## 计算机图形学

工业设计

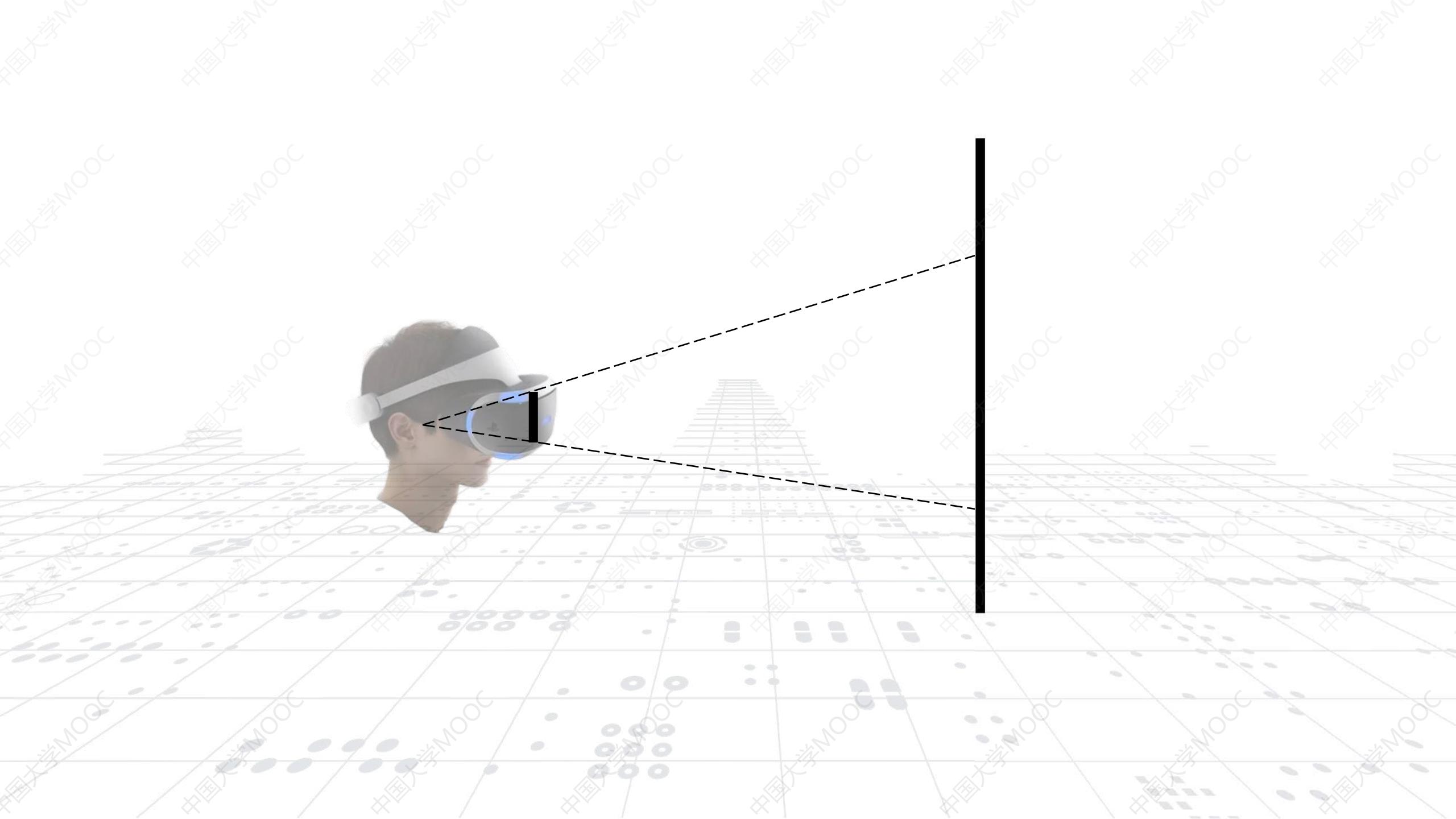
特效动画

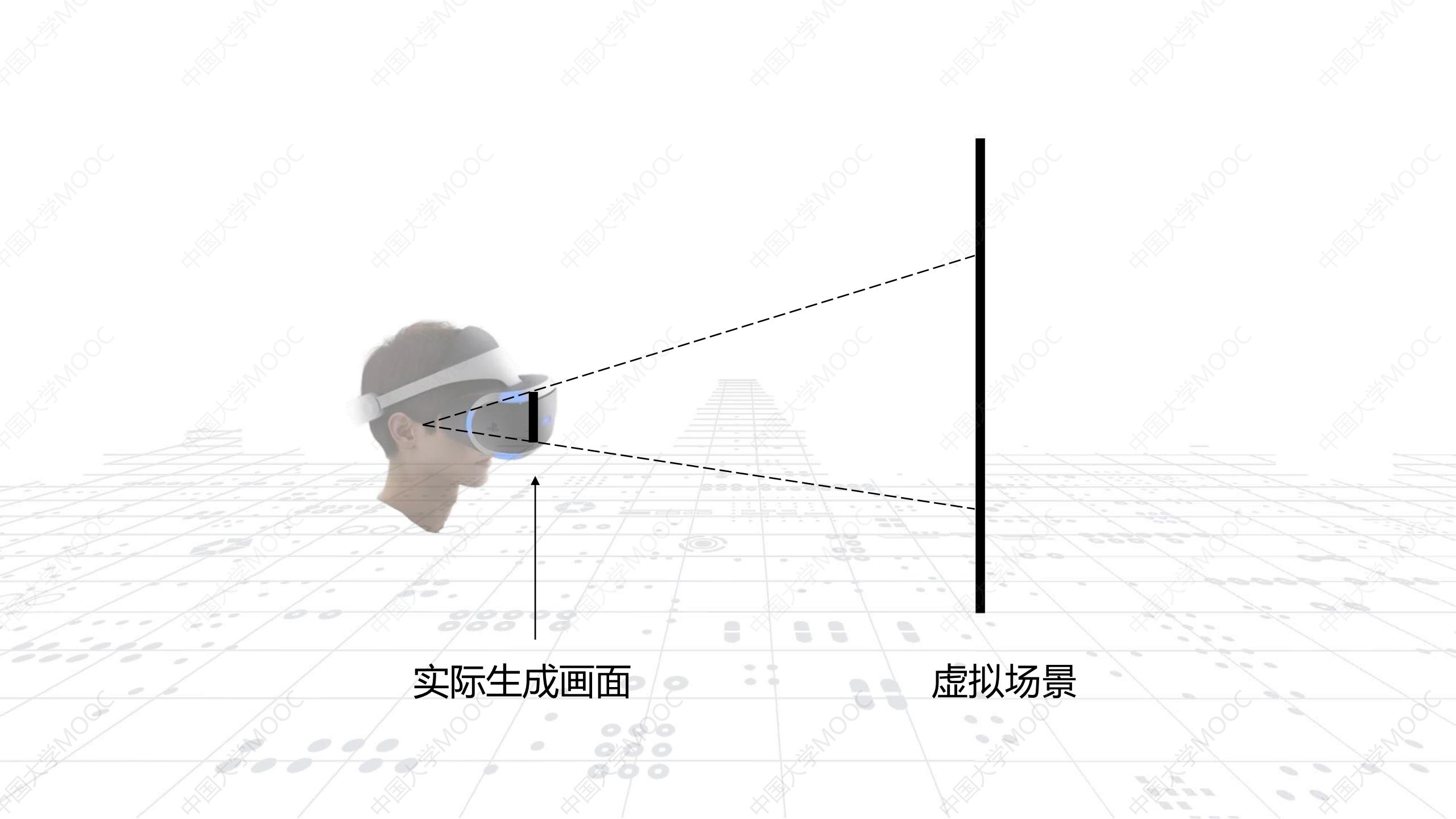
电子游戏

虚拟现实



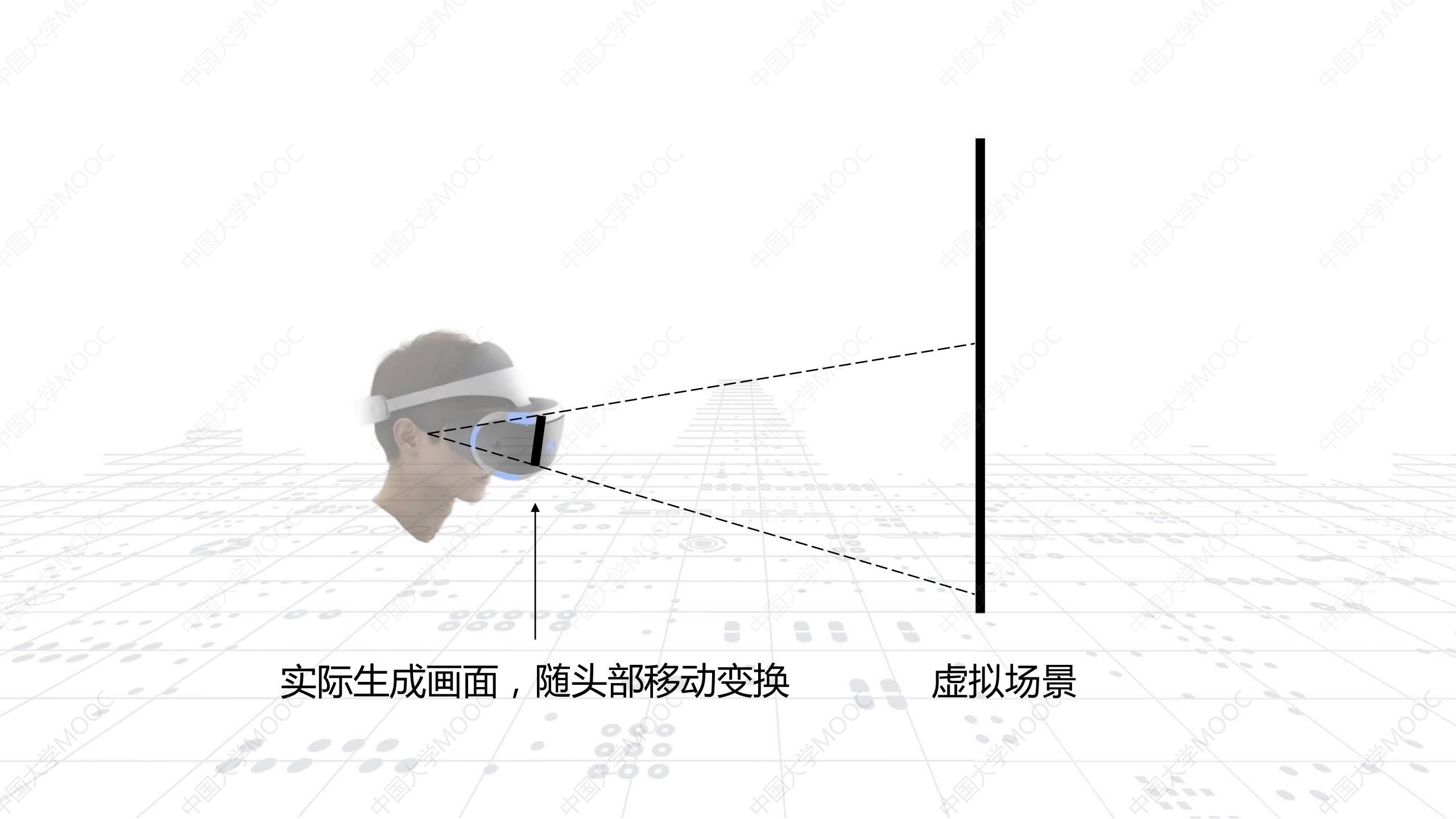
PS4 VR眼镜





实际生成画面

虚拟场景



实际生成画面，随头部移动变换

虚拟场景

Input	Output	Digital image	Knowledge
Digital image	Digital Image Processing		Computer Vision
Knowledge	Computer Graphics		Artificial Intelligence

## 数字图像处理

图像增强  
图像分割  
图像压缩  
图像变换

## 计算机视觉

图像识别  
目标检测  
图像检索  
图像理解

## 计算机图形学

工业设计  
特效动画  
电子游戏  
虚拟现实

## 人 工 智 能

信息挖掘  
控制决策  
语言翻译  
语言理解

Input	Output	Digital image	Knowledge
Digital image	Digital Image Processing	Computer Vision	
Knowledge	Computer Graphics	Artificial Intelligence	

## 数字图像处理

图像增强  
图像分割  
图像压缩  
图像变换

## 计算机视觉

图像识别  
目标检测  
图像检索  
图像理解

## 计算机图形学

工业设计  
特效动画  
电子游戏  
虚拟现实

## 人 工 智 能

信息挖掘  
控制决策  
语言翻译  
语言理解



## 人 工 智 能

信息挖掘

控制决策

语言翻译

语言理解

从大量的数据中通过算法搜索隐藏于其中信息的过程。



## 人 工 智 能

信息挖掘

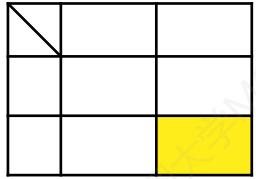
控制决策

语言翻译

语言理解



自动  
驾驶



## 人 工 智 能

信息挖掘

控制决策

语言翻译

语言理解





## 人 工 智 能

信息挖掘

控制决策

语言翻译

语言理解

自然语言处理( NLP, Natural Language Processing )

Input	Output	Digital image	Knowledge
Digital image	Digital Image Processing	Computer Vision	
Knowledge	Computer Graphics	Artificial Intelligence	

## 数字图像处理

图像增强

图像分割

图像压缩

图像变换

## 计算机视觉

图像识别

目标检测

图像检索

图像理解

## 计算机图形学

工业设计

特效动画

电子游戏

虚拟现实

## 人 工 智 能

信息挖掘

控制决策

语言翻译

语言理解

Input	Output	Digital image	Knowledge
Digital image	Digital Image Processing	Computer Vision	
Knowledge	Computer Graphics	Artificial Intelligence	

## 数字图像处理

图像增强

图像分割

图像压缩

图像变换

## 计算机视觉

图像识别

目标检测

图像检索

图像理解

## 计算机图形学

工业设计

特效动画

电子游戏

虚拟现实

## 人 工 智 能

信息挖掘

控制决策

语言翻译

语言理解

Input	Output	Digital image	Knowledge
Digital image	Digital Image Processing	Computer Vision	
Knowledge	Computer Graphics	Artificial Intelligence	

## 数字图像处理

图像增强

图像分割

图像压缩

图像变换

## 计算机视觉

图像识别

目标检测

图像检索

图像理解

## 计算机图形学

工业设计

特效动画

电子游戏

虚拟现实

## 人 工 智 能

信息挖掘

控制决策

语言翻译

语言理解

# 机器视觉技术

## 数字图像处理

图像增强  
图像分割  
图像压缩  
图像变换

## 计算机视觉

图像识别  
目标检测  
图像检索  
图像理解

## 人 工 智 能

信息挖掘  
控制决策  
语言翻译  
语言理解

# 这门课主要学什么？



目标检测



目标计数



缺陷检测



图像增强



图像变换



图像压缩

# 涉及的知识



高等数学 / 概率论 / 线性代数 / 数字信号处理 /  
矩阵理论 / 几何学 / 机器学习 / 最优化理论



C++、python、matlab



OpenCV、Halcon

# OpenCV库介绍

李竹

杭州电子科技大学

电子信息学院



# 本章概要

1. OpenCV简介
2. 编译过程简介
3. OpenCV的基本数据格式
4. OpenCV的基本操作

# Opencv简介

OpenCV于1999年由Intel建立。OpenCV是一个开源的跨平台计算机视觉库，可以运行在Linux、Windows和Mac OS操作系统上。它轻量级而且高效——由一系列C函数和少量C++类构成，同时提供了Python、Ruby、MATLAB等语言的接口，实现了图像处理和计算机视觉方面的很多通用算法。目前的最新版本是opencv4.0。

Windows平台下，opencv提供了已经编译好的文件，可以直接在微软的VS2015和VS2017中调用。同时也提供源代码，如果使用其他编译器，可以自己编译源码。

Opencv主页：<https://opencv.org/>

# Opencv简介

OpenCV在VS中的配置，请参考以下两个文档。在配置的时候希望大家不要只是依葫芦画瓢地操作，应该在理解程序编译运行的原理的基础上进行操作。

1.VS2015下的OpenCV的配置(C++).pdf

2.编译更加强大的opencv(opencv\_contrib编译方法).pdf

# 编译原理简介

## C/C++代码编译过程

C语言的编译链接过程要把我们编写的一个c程序（源代码）转换成可以在硬件上运行的程序（可执行代码）。

分为两个部分，编译和链接。

编译就是把文本形式源代码翻译为机器语言形式的目标文件的过程。

链接是把目标文件、操作系统的启动代码和用到的库文件进行组织，形成最终生成可执行代码的过程。

# 编译原理简介

## Step1 编译

编译过程又可以分成两个阶段：编译和汇编。

在编译阶段，编译器会首先进行预处理，预处理主要对下面的内容进行处理：宏定义指令。2) 条件编译指令。3) 头文件包含指令。4) 特殊符号。

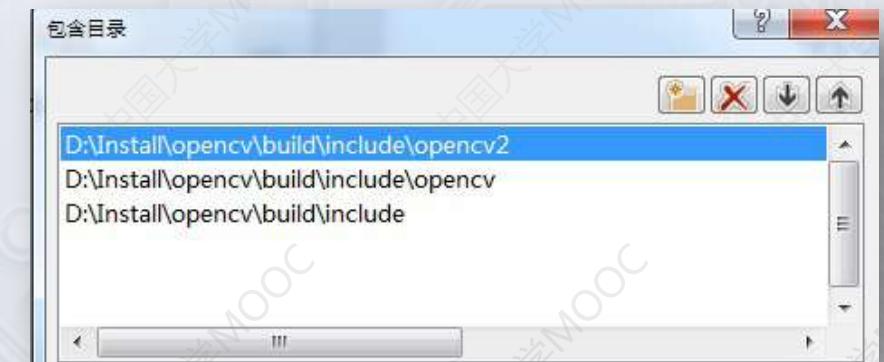
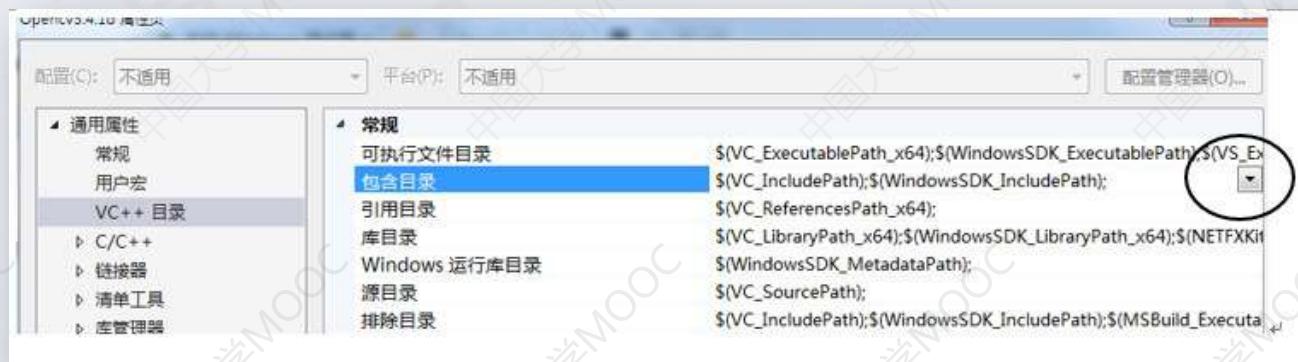
编译阶段，编译器还会进行一些优化，检查语法规则。然后会把代码生成为汇编码。

汇编阶段会把汇编码生成成为目标机器所能够读取的机器码。

# 编译原理简介

## Step1 编译

我们在使用opencv的库时候，就需要通过#include <opencv.hpp>的语言去读取opencv的头文件。通过头文件中的函数定义，调用opencv 的函数。因此我们需要告诉编译器，这些头文件的。在VS中设置包含目录就是这个目的（详细设置请参考文档）。



# 编译原理简介

## Step2 链接

编译阶段生成的机器码实际上还不能马上运行。因为opencv头文件中只有函数的定义，没有函数的内容。函数的内容通常是封装在dll动态库（Linux下为.so后缀文件）或者lib静态库（Linux下为.a后缀文件）文件中。那么在链接阶段，就需要告诉编译器，去哪里找这些库文件，以及库文件的名称。

# 编译原理简介

## Step2 链接

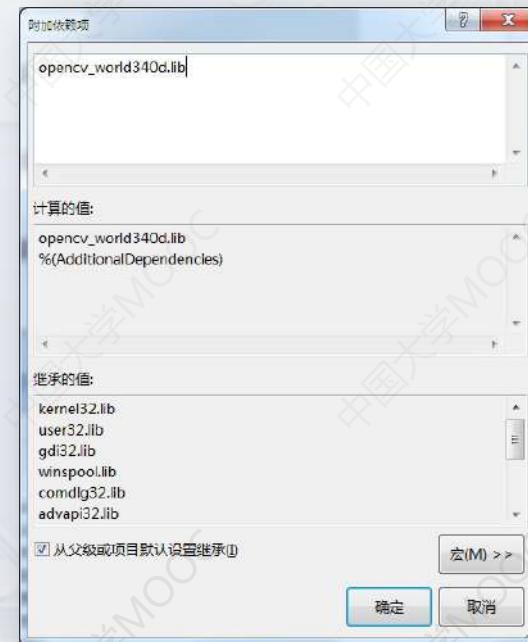
在windows下使用dll静态库文件会按照以下顺序寻找(程序当前目录  
>System32目录>环境变量Path所设置路径)。所以我们需要在系统属性的  
环境变量Path中添加opencv库存放的路径



# 编译原理简介

## Step2 链接

同时还需要在IDE ( VS ) 中指定库目录已经链接器链接文件的名称，即依赖项（详细参考文档）。



# Opencv的基本数据格式

OpenCV早起版本中，采用IplImage格式（intel另一个开源库的格式）来保存图像。到2.0以后的版本，opencv库引入面向对象的思想，采用C++重写了大量代码。并引入了Mat类作为图像容器。Mat（matrix缩写）类本身是一个矩阵格式，也可以用来保存图像。所以opencv中的矩阵运算也可以使用Mat型。

读取图片并显示，请参照文档《VS2015下的OpenCV的配置(C++).pdf》的最后部分。

以下介绍一些Mat的基本操作，这些基本操作对于图像Mat还是单纯的数据矩阵都是通用的。

# Mat的基本操作

1. 创建: opencv提供了很多方法创建mat , 这里介绍两种。

( 1 ) 使用Mat()构造函数

```
cv::Mat M1(2,2,CV_8UC3,Scalar(0,0,255))
```

这个函数的意思是 , 创建了一个名为M1的Mat , 该Mat的尺寸为2,2 , 类型为CV\_8UC3 , 即8位uchar类型 , 该Mat通道数为3。这个mat的每一个元素包含了3个通道或者说3个数值。然后用0,0,255为每一个元素赋值。

这里8位uchar型的取值为0~255 , 实际上如果一个Mat是用来表示RGB图像的时候应该声明为CV\_8UC3型。

# Mat的基本操作

Mat型可以定义各种类型，可按F12快捷键查看头文件中的定义。定义的方式如下：

CV\_(位数) + (数据类型) + (通道数)

如CV\_32FC1表示32位float型数据，有如下类型

CV\_8UC1 CV\_8UC2 CV\_8UC3 CV\_8UC4 CV\_8SC1 CV\_8SC2  
CV\_8SC3 CV\_8SC4 CV\_16UC1 CV\_16UC2 CV\_16UC3 CV\_16UC4  
CV\_16SC1 CV\_16SC2 CV\_16SC3 CV\_16SC4 CV\_32SC1 CV\_32SC2  
CV\_32SC3 CV\_32SC4 CV\_32FC1 CV\_32FC2 CV\_32FC3 CV\_32FC4  
CV\_64FC1 CV\_64FC2 CV\_64FC3 CV\_64FC4

# Mat的基本操作

## 1. 创建:

( 2 ) 使用create()函数,代码如下

```
cv::Mat M3;
```

```
M3.create(3,4,CV_8UC3);
```

表示首先声明一个mat型，名叫M3，其尺寸为3行，4列。

# Mat的基本操作

## 2. 复制:

### (1) 浅复制

```
cv::Mat srcM(2,2,CV_8UC3,Scalar(0,0,255))
```

```
cv::Mat dstM;
```

```
dstM=srcM;
```

表示首先声明一个mat名为 srcM，并初始化。然后声明一个mat名为dstM  
通过 “=” 把srcM复制给dstM。

这样生成的矩阵，只是新生成一个矩阵头，dstM的数据依然指向矩阵srcM  
的数据，类似C++中的浅拷贝。

# Mat的基本操作

## 2.复制:

### ( 2 ) 深复制

```
cv::Mat srcM(2,2,CV_8UC3,Scalar(0,0,255))
```

```
cv::Mat dstM;
```

```
srcM.copyTo(dstM);
```

通过copyTo函数，可以实现深复制。也就是dstM是一个全新的矩阵，他在内存中的地址和srcM是不一样的。

另外copyTo函数还可以加上掩模参数

# Mat的基本操作

3.遍历Mat：当我们想读取或者修改Mat的任意内容时候，可以用以下方式访问Mat。实际上遍历Mat有十数种方式，这里仅做简要介绍。

## ( 1 ) 利用指针.ptr

```
int height= image.rows; //行数  
Int width = image.cols * image.channels(); //每行元素的总元素数量  
for (int j=0; j<height; j++)  
{  
    //定义指针data , 其值为image的第j行的头地址  
    uchar* data= image.ptr<uchar>(j);  
    for (int i=0; i<width; i++)  
    {  
        //----开始处理每个像素,每个元素的值减少到1/2----  
        data[i]= data[i]/2;  
        //-----结束像素处理-----  
    } //单行处理结束  
}
```

# Mat的基本操作

3.遍历Mat：当我们想读取或者修改Mat的任意内容时候，可以用以下方式访问Mat。实际上遍历Mat有十数种方式，这里仅做简要介绍。

## ( 1 ) 利用指针.ptr

```
int height= image.rows; //行数  
Int width = image.cols * image.channels(); //每行元素的总元素数量  
for (int j=0; j<height; j++)  
{  
    //定义指针data , 其值为image的第j行的头地址  
    uchar* data= image.ptr<uchar>(j);  
    for (int i=0; i<width; i++)  
    {  
        //----开始处理每个像素,每个元素的值减少到1/2----  
        data[i]= data[i]/2;  
        //-----结束像素处理-----  
    } //单行处理结束  
}
```

# Mat的基本操作

## 3.遍历Mat：

( 2 ) 利用.at,如果是3通道的RGB图像

```
int height= image.rows; //行数  
Int width = image.cols ; //每行元素的总元素数量  
for (int j=0; j<height; j++)  
{  
    for (int i=0; i<width; i++)  
    {  
        //----开始处理每个像素,每个元素的值减少到1/2----  
        image.at<Vec3b>(j,i)[0]= image.at<Vec3b>(j,i)[0]/div*div + div/2;  
        image.at<Vec3b>(j,i)[1]= image.at<Vec3b>(j,i)[1]/div*div + div/2;  
        image.at<Vec3b>(j,i)[2]= image.at<Vec3b>(j,i)[2]/div*div + div/2;  
        //-----结束像素处理-----  
    } //单行处理结束  
}
```

# Mat的基本操作

## 3.遍历Mat：

( 2 ) 利用.at , 如果是单通道的灰度图

```
int height= image.rows; //行数  
Int width = image.cols * image.channels(); //每行元素的总元素数量  
for (int j=0; j<height; j++)  
{  
    for (int i=0; i<width; i++)  
    {  
        //----开始处理每个像素,每个元素的值减少到1/2----  
        image.at<uchar>(j,i)= image.at<uchar>(j,i)/2;  
        //-----结束像素处理-----  
    } //单行处理结束  
}
```

# 谢谢！

# 机器视觉技术与应用

## 2. 基本概念

李竹

杭州电子科技大学  
电子信息学院



# 本章概要

硬件介绍

1

基本概念

2

图像种类

3

色彩模型

4

直方图

5

# 本章概要

硬件介绍

1

基本概念

2

图像种类

3

色彩模型

4

直方图

5

工业相机



民用相机



VS

# 工业相机

# 民用相机

长

工作时间

短

快

快门速度

慢

逐行

扫描方式

隔行

高

拍摄帧率

低

Raw data

输出数据

压缩

高

产品价格

低



VS



# 工业相机

---

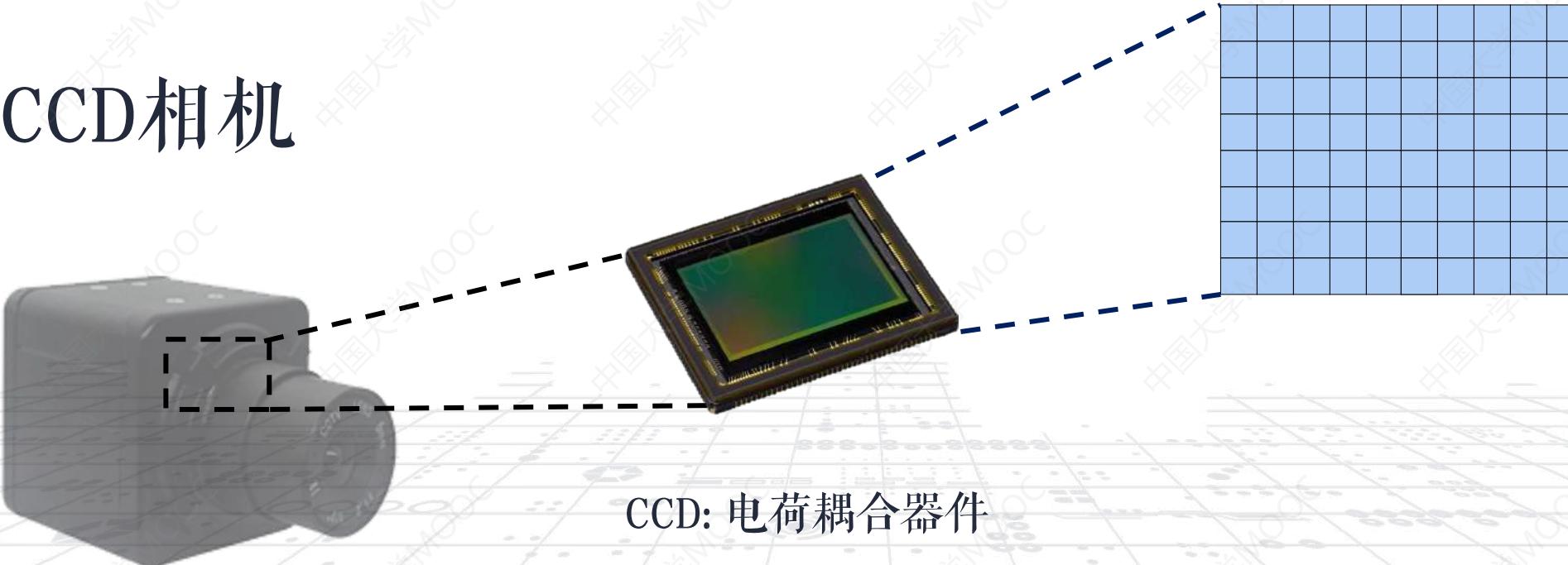
CCD相机

CMOS相机

# CCD相机

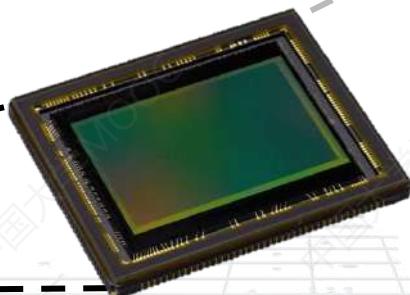
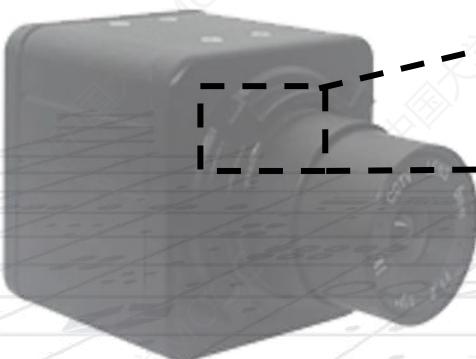


# CCD相机

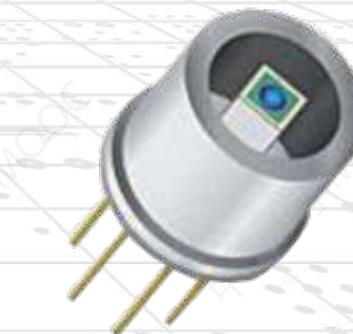


CCD: 电荷耦合器件  
( Charge-Coupled Device )

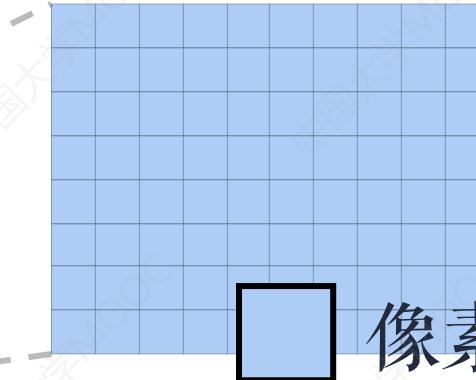
# CCD相机



CCD: 电荷耦合器件  
( Charge-Coupled Device )



光电二极管



像素



选型中心

产品中心

解决方案

下载中心

服务支持

新闻中心

关于我们



热线电话

400 999 7595

## 面阵相机



// GigE Vision 接口 // 分辨率最高可达 5.1MP // 全局曝光、全局复位和行曝光传感器可选 // 全分辨率下帧率高达 286fps // 近红外和偏振光版本可选

首页 > 产品中心 > 面阵相机 > Allied Vision > Mako系列

# Mako系列相机

## 灵巧机身、非凡性能

Mako G-032C

在线咨询

销售

技术

# 规格参数

型号	Mako G-032C
品牌	Allied Vision
分辨率	658 × 492
帧率(fps)	102.3
传感器厂商	Sony
传感器	1/3" ICX424 Global shutter CCD
像元尺寸	7.4µm
像素深度	8 bit, 12 bit
数据接口	GigE
镜头接口	C, CS, M12
光谱	彩色
图像数据格式	BayerRG8/12/12p, BGR8p, RGB8p, YUV411p/422p/444p, Mono8
动态范围	61.4 dB
供电要求	12 ~ 24 VDC 或 PoE
额定功率	2.4 W @ 12 VDC; 2.8 W @ PoE
工作温度	+5°C ~ +45°C, 封装温度
机械尺寸(W×H×L)	29.2 mm × 29.2 mm × 60.5 mm (包含转接器)

## 规格参数

	Mako G-032C
型号	Allied Vision
品牌	
分辨率	658 × 492
帧率(fps)	102.3
传感器厂商	Sony
传感器	1/3" ICX424 Global shutter CCD
像元尺寸	7.4µm
像素深度	8 bit, 12 bit
数据接口	GigE
镜头接口	C, CS, M12
光谱	彩色
图像数据格式	BayerRG8/12/12p, BGR8p, RGB8p, YUV411p/422p/444p, Mono8
动态范围	61.4 dB
供电要求	12 ~ 24 VDC 或 PoE
额定功率	2.4 W @ 12 VDC; 2.8 W @ PoE
工作温度	+5°C ~ +45°C, 封装温度
机械尺寸(W×H×L)	29.2 mm × 29.2 mm × 60.5 mm (包含转接器)

## 规格参数

型号	Mako G-032C
品牌	Allied Vision
分辨率	658 × 492
帧率(fps)	102.3
传感器厂商	Sony
传感器	1/3" ICX424 Global shutter CCD
像元尺寸	7.4µm
像素深度	8 bit, 12 bit
数据接口	GigE
镜头接口	C, CS, M12
光谱	彩色
图像数据格式	BayerRG8/12/12p, BGR8p, RGB8p, YUV411p/422p/444p, Mono8
动态范围	61.4 dB
供电要求	12 ~ 24 VDC 或 PoE
额定功率	2.4 W @ 12 VDC; 2.8 W @ PoE
工作温度	+5°C ~ +45°C, 封装温度
机械尺寸(W×H×L)	29.2 mm × 29.2 mm × 60.5 mm (包含转接器)

## 规格参数

型号	Mako G-032C
品牌	Allied Vision
分辨率	658 × 492
帧率(fps)	102.3
传感器厂商	Sony
传感器	1/3" ICX424 Global shutter CCD
像元尺寸	7.4μm
像素深度	8 bit, 12 bit
数据接口	
镜头接口	
光谱	
图像数据格式	BayerRG8/12/12p /422p/444p, Mono8
动态范围	
供电要求	12 ~ 24 VDC 或 PoE
额定功率	2.4 W @ 12V @ PoE
工作温度	+5°C ~ +45°C, 封装温度
机械尺寸(W×H×L)	29.2 mm × 29.2 mm × 60.5 mm (包含转接器)

型号	宽高比	光导摄像管 直径 (mm)	靶面区域			
			对角线 (mm)	宽 (mm)	高 (mm)	转换系数* (Crop Factor)
1/6"	4:3	4.233	3.000	2.400	1.800	14.422
1/4"		6.350	4.000	3.200	2.400	10.817
1/3.6"		7.056	4.500	3.600	2.700	9.615
1/3.2"		7.938	5.000	4.000	3.000	8.653
1/3"		8.467	5.678	4.536	3.416	7.620
1/2.7"		9.407	6.000	4.800	3.600	7.211
1/2.5"		9.407	6.592	5.270	3.960	6.564
1/2.3"		9.407	6.718	5.371	4.035	6.441
1/2"		10.160	7.182	5.760	4.290	6.024
1/1.8"		11.044	7.700	6.160	4.620	5.619
1/1.7"		12.700	8.000	6.400	4.800	5.408
1/1.6"		14.111	8.933	7.176	5.319	4.843
2/3"		14.941	9.500	7.600	5.700	4.554
1"		15.875	10.070	8.080	6.010	4.297
2/3"		16.933	11.000	8.800	6.600	3.933
1"		25.400	16.000	12.800	9.600	2.704

# 面阵相机



# 线阵相机

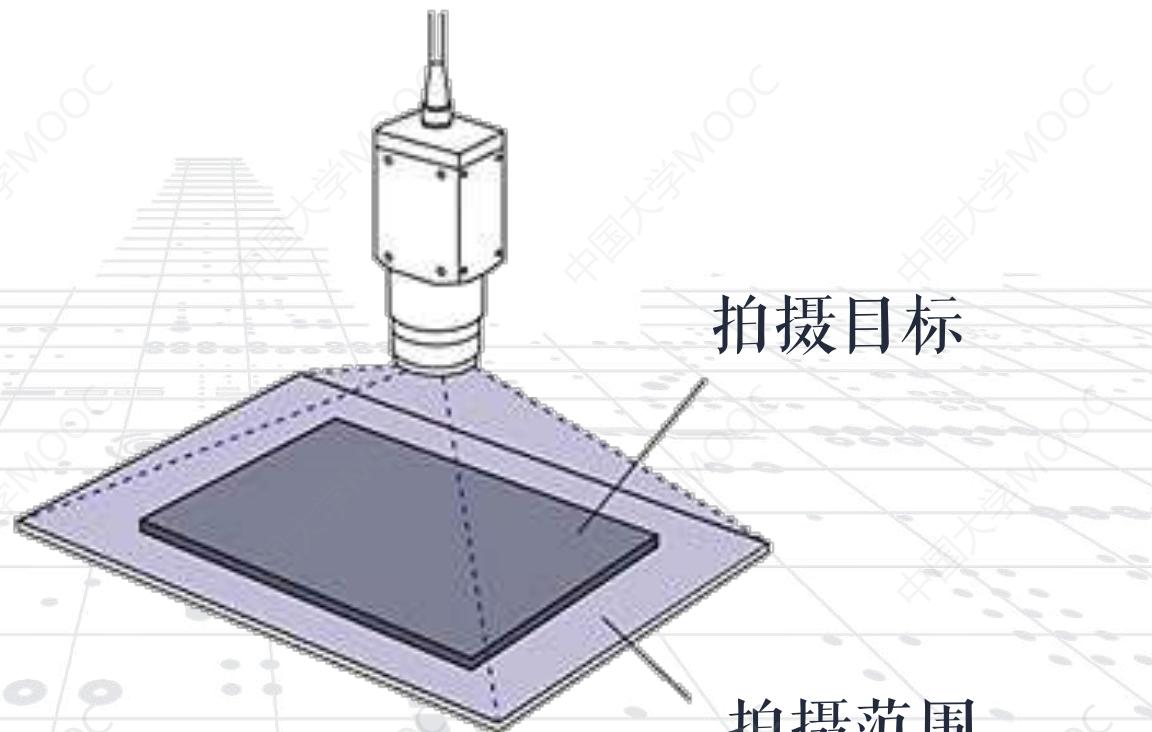


VS

# 面阵相机

一次拍摄一个区  
域，较为常用。

# 线阵相机



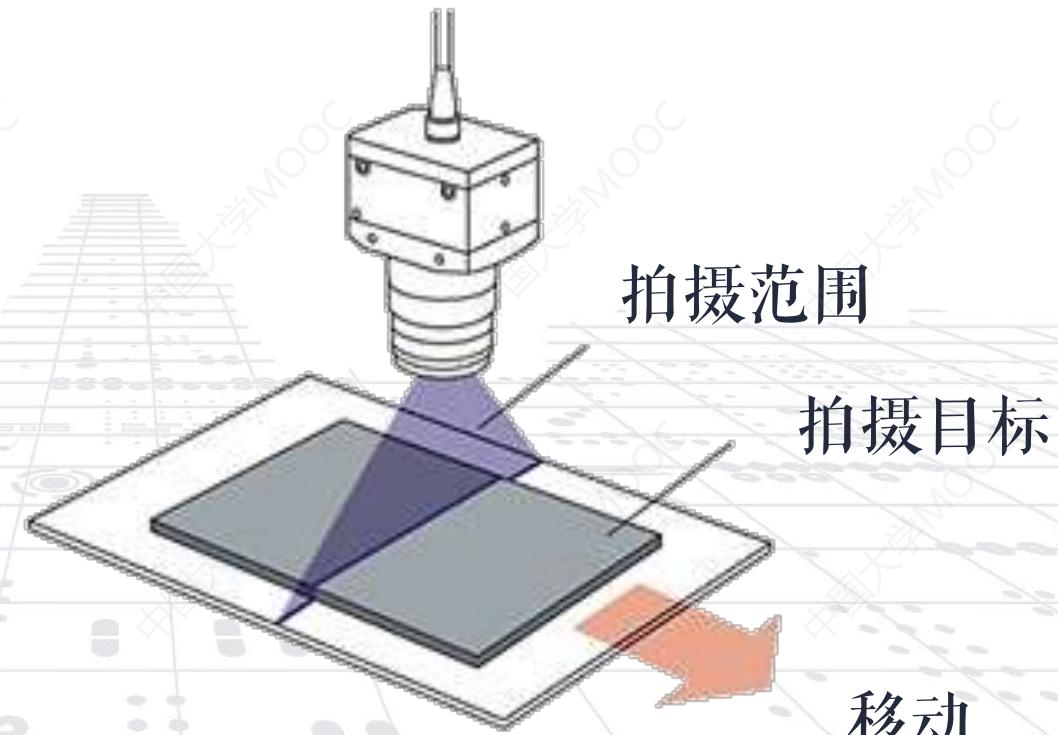
拍摄范围

拍摄目标

# 面阵相机

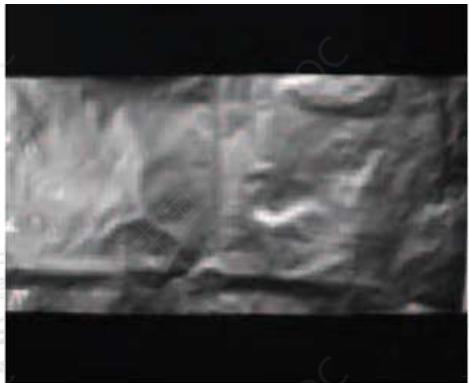
一次拍摄一行像素，  
通过移动以及拼接来  
获取图像。

# 线阵相机



移动

# 面阵相机



价格便宜  
较为常用

# 线阵相机

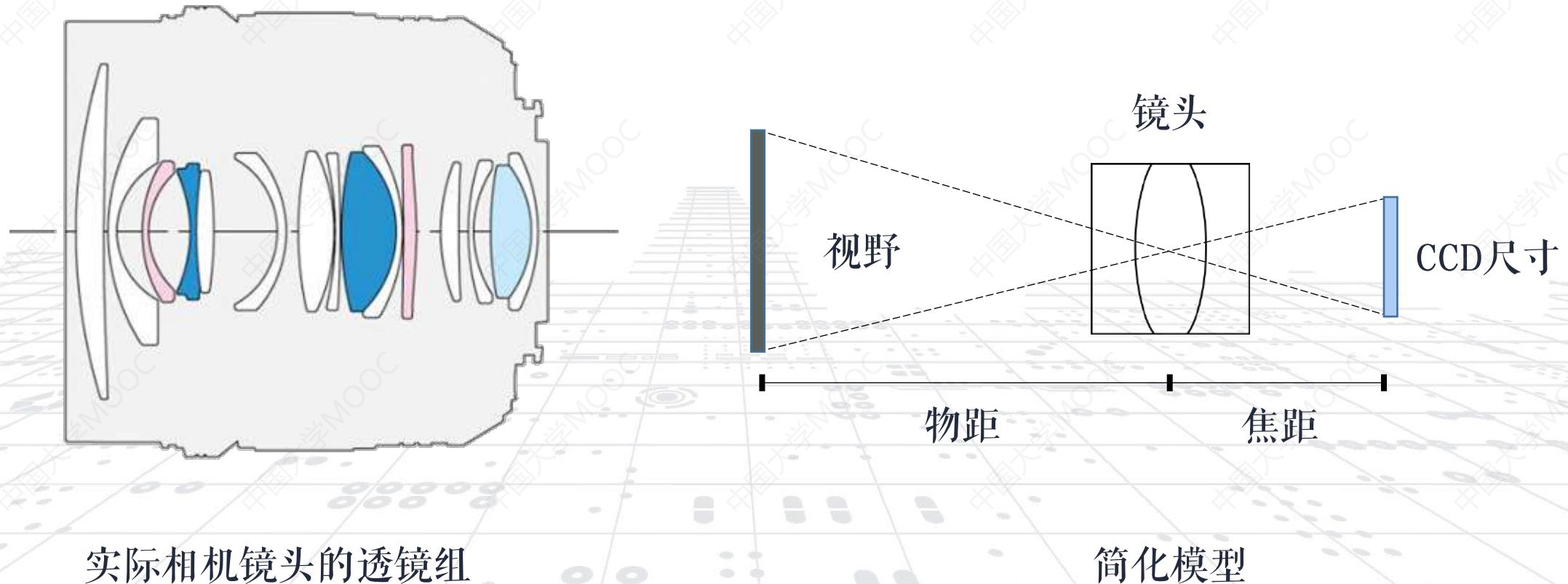
VS



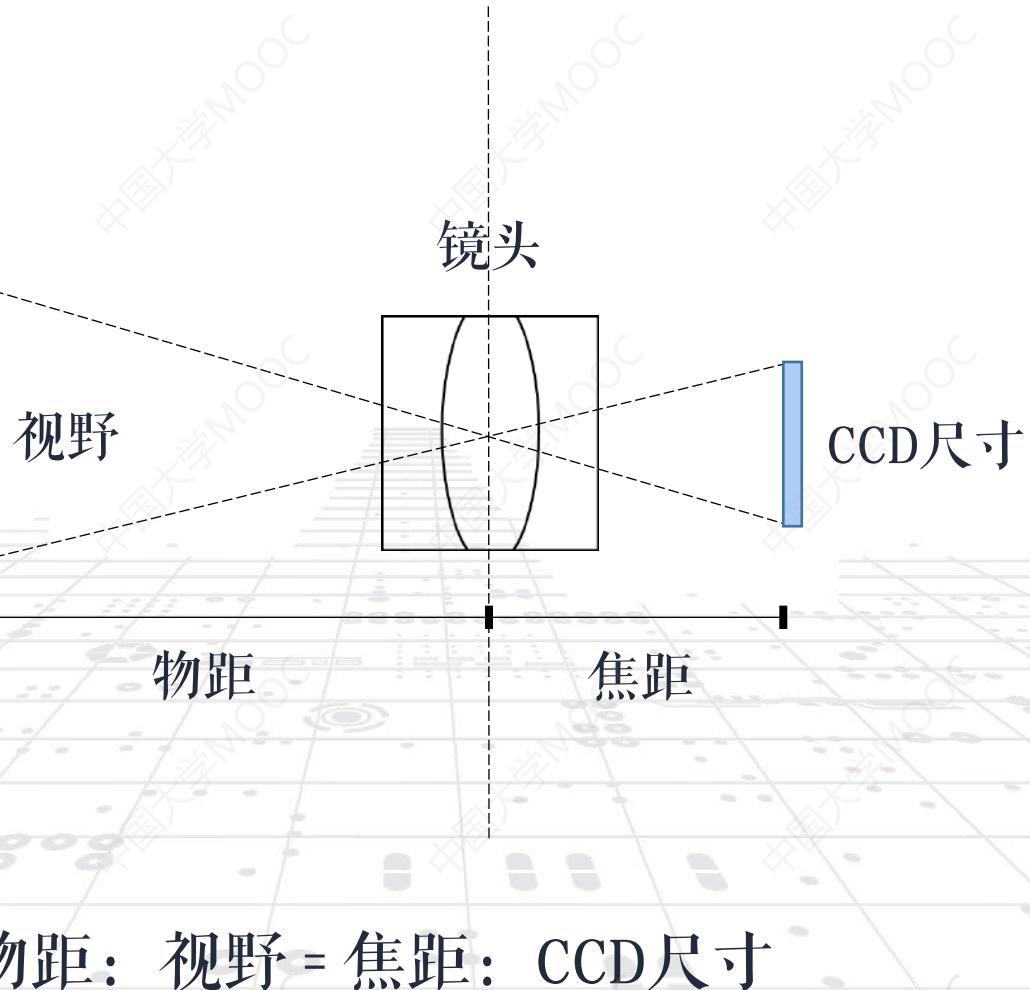
补光效果好  
成像速度快  
价格较昂贵



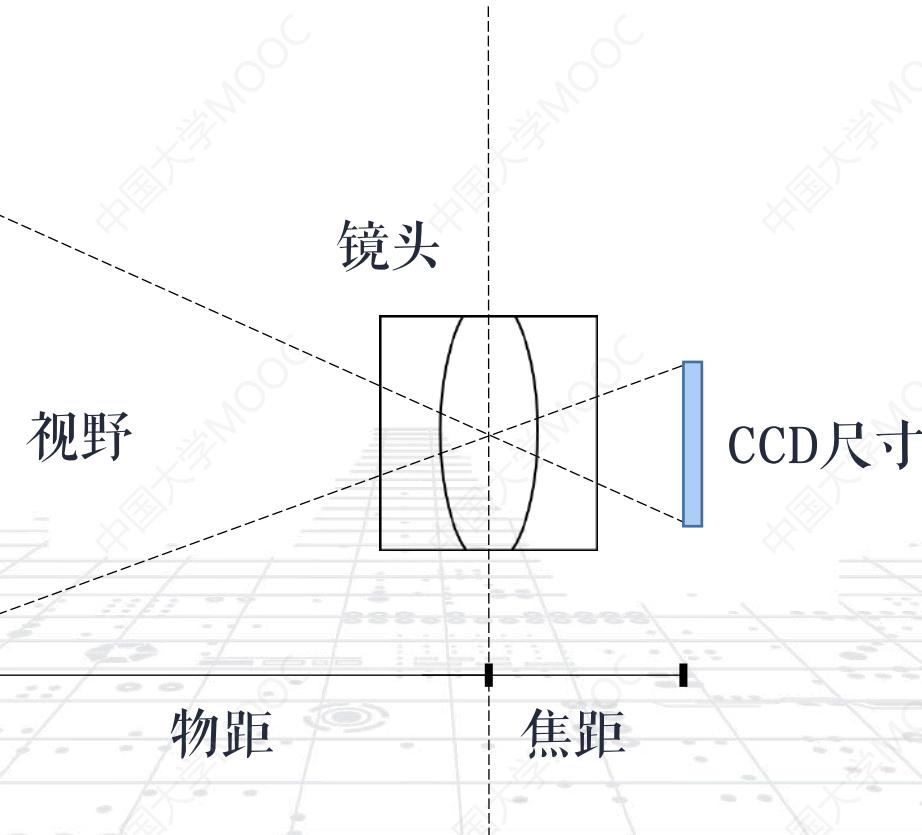
# 成像模型



# 成像模型



# 成像模型

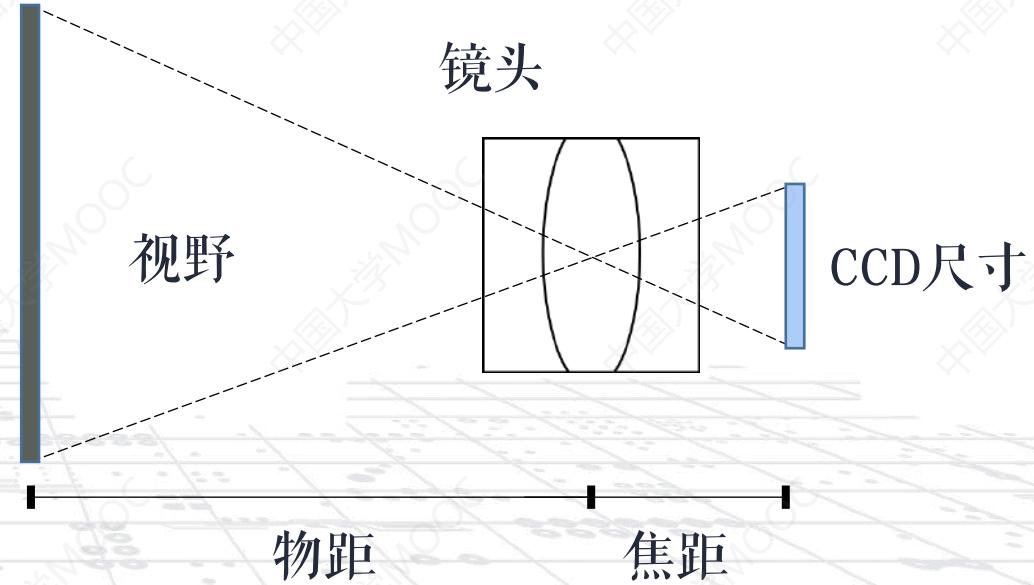


$$\text{物距} : \text{视野} = \text{焦距} : \text{CCD尺寸}$$

# 成像模型

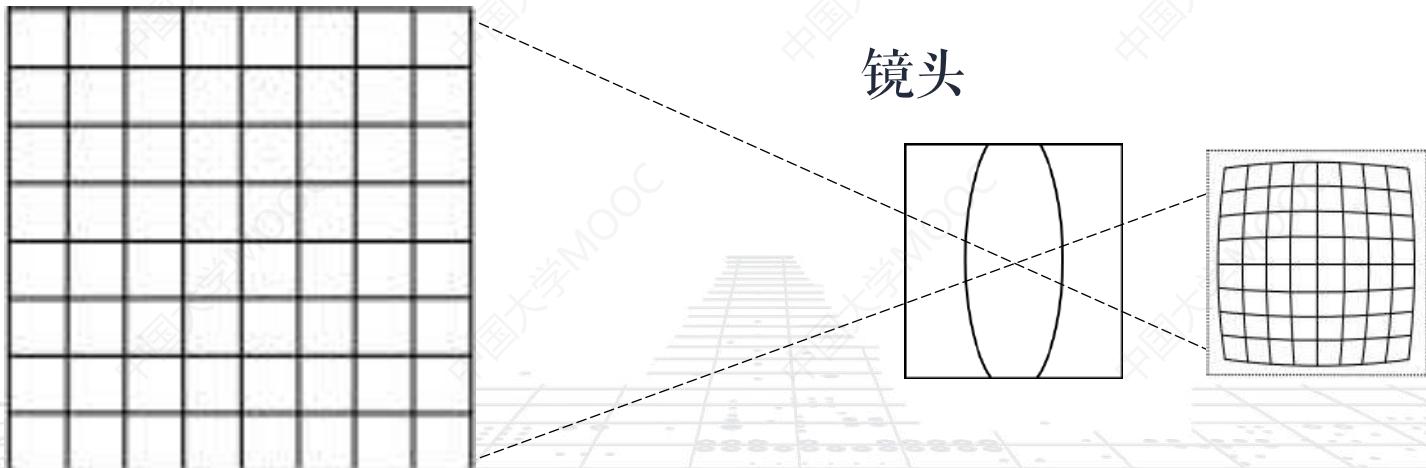
根据实际情况进行合理选择相机参数。

例：镜头焦距 16mm、CCD 尺寸 3.6mm  
时，为了得到 45mm 的视野，物距应  
为 200mm



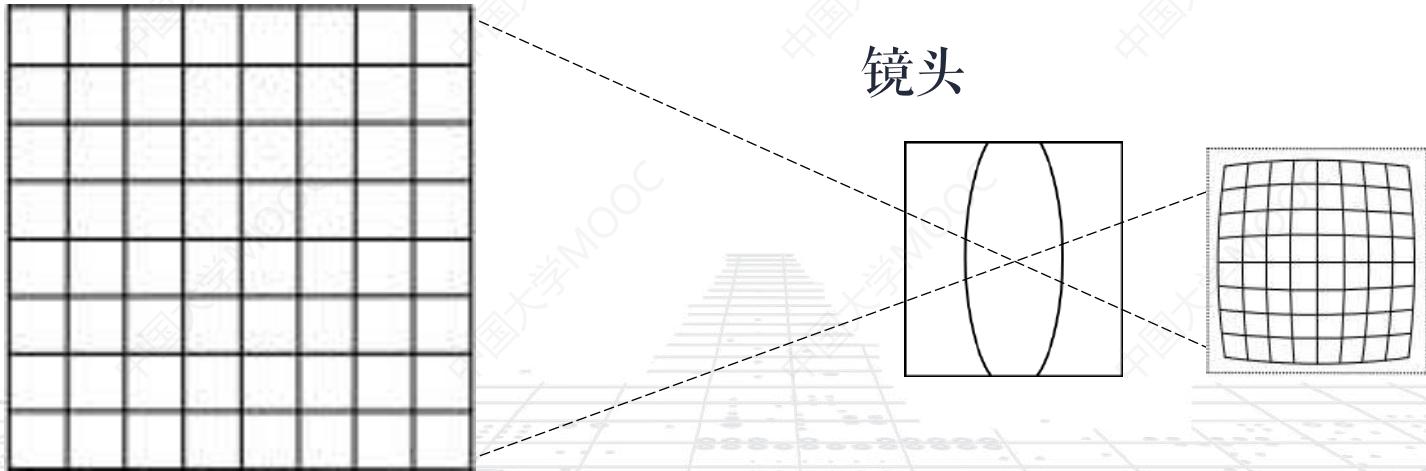
物距 : 视野角 = 焦距 : CCD尺寸

# 成像模型



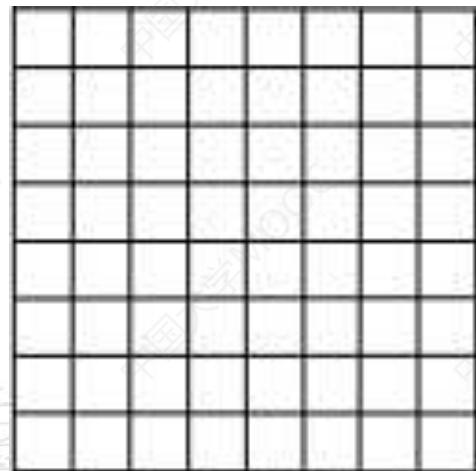
**畸变**: 镜头中间部分的放大率和周围的放大率不同而产生的变形。

# 成像模型

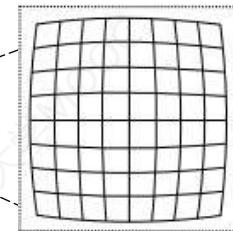
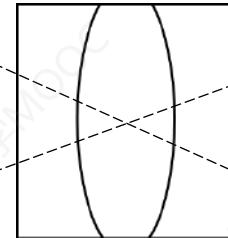


**畸变：**镜头中间部分的放大率和  
周围的放大率不同而产生的变形。

# 成像模型



镜头



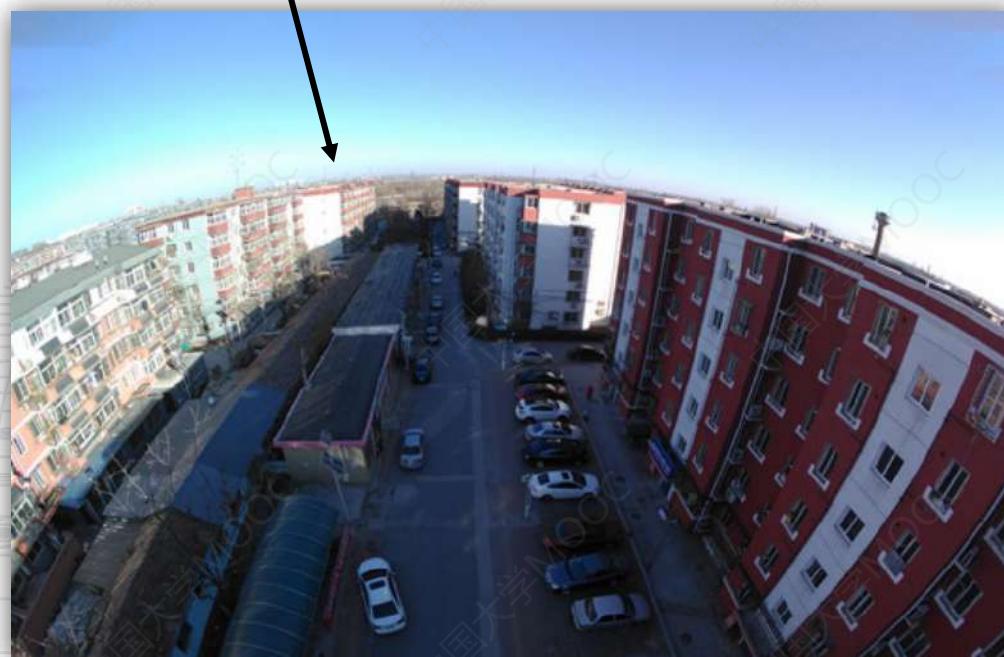
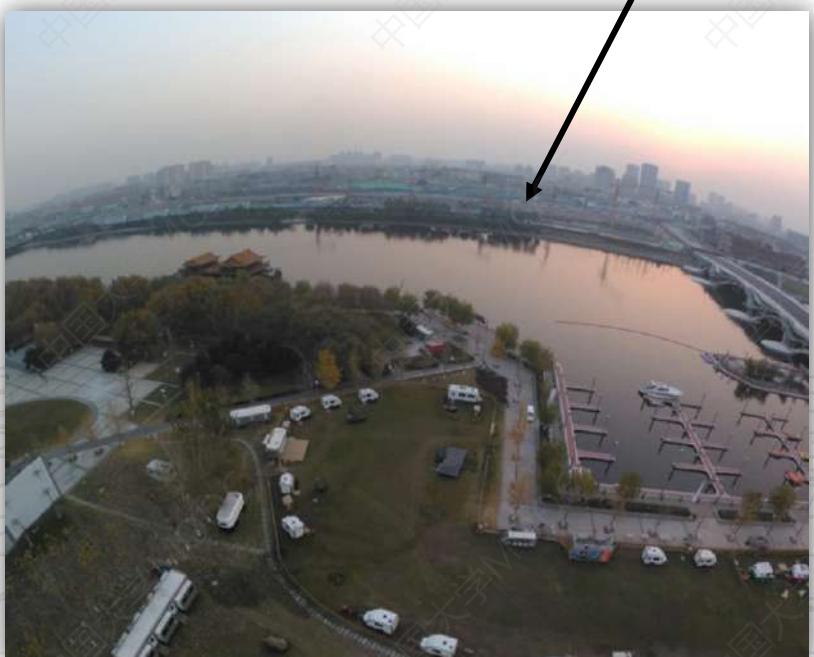
**畸变：**镜头中间部分的放大率和周围的放大率不同而产生的变形。

一般来说，焦距越短失真程度会约大。

在一些如测量尺寸的视觉任务中会产生误差。

# 畸变

畸变引起直线发生扭曲



广角相机拍摄的图片

# 远心镜头

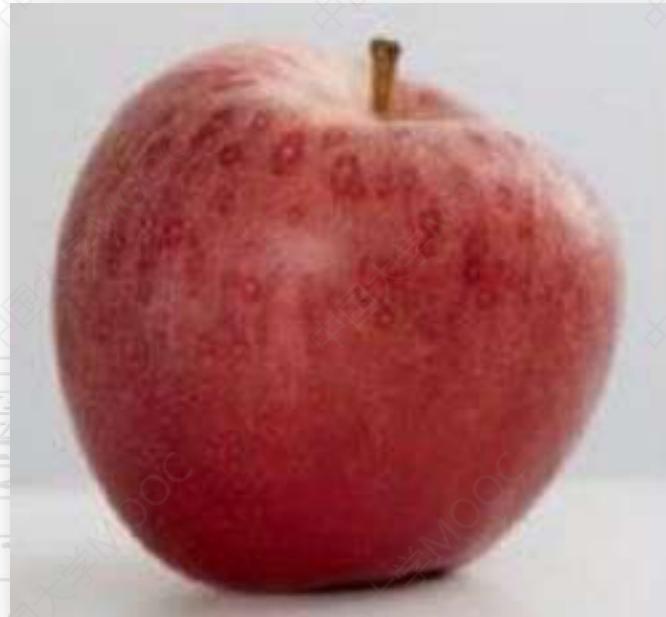


远心镜头，几乎没有畸变

# 红外相机



BC-C2900短波红外相机



普通相机

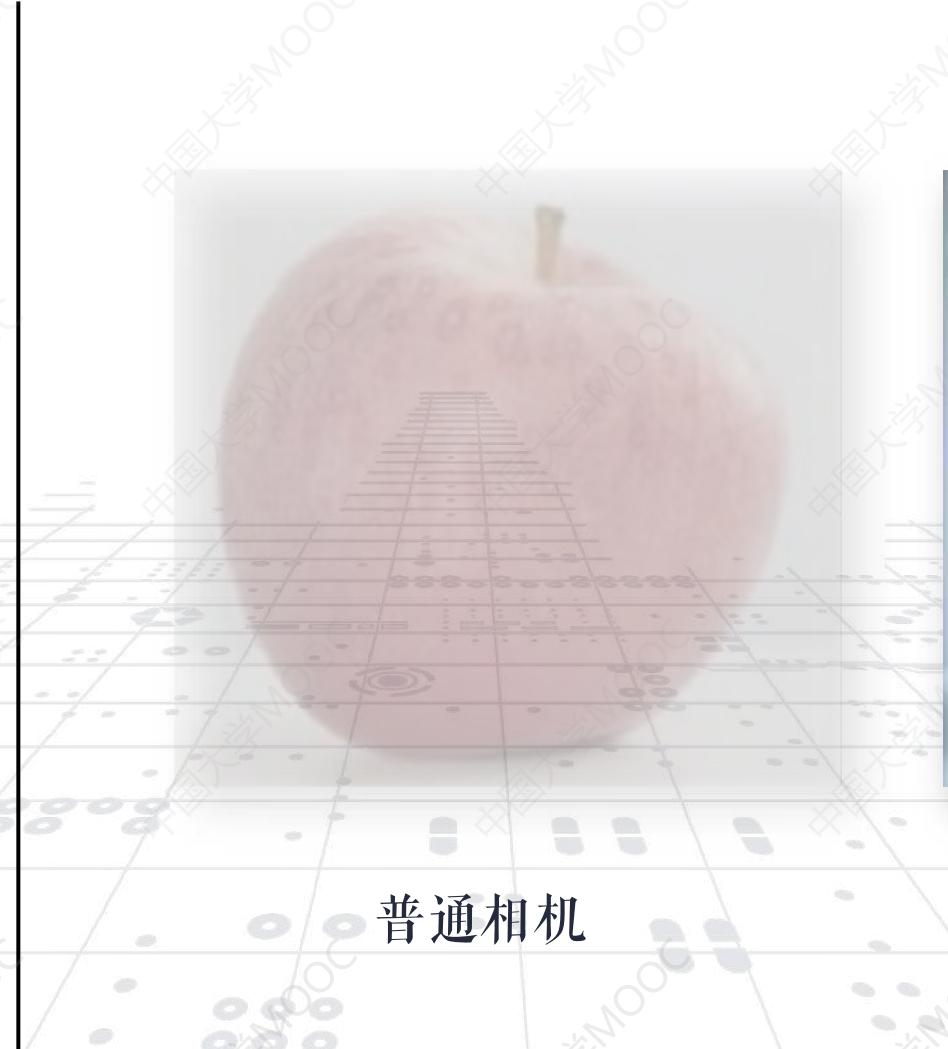


短波红外

# 红外相机



BC-C2900短波红外相机



普通相机

短波红外



液体吸收率高，可以拍摄  
到皮下液态的腐烂部分

# 双目相机



Intel realsense d455

## 双目测距原理

# 双目相机



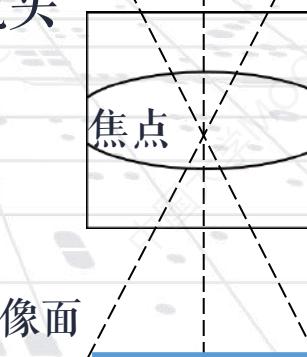
Intel realsense d455

物理面

镜头

焦点

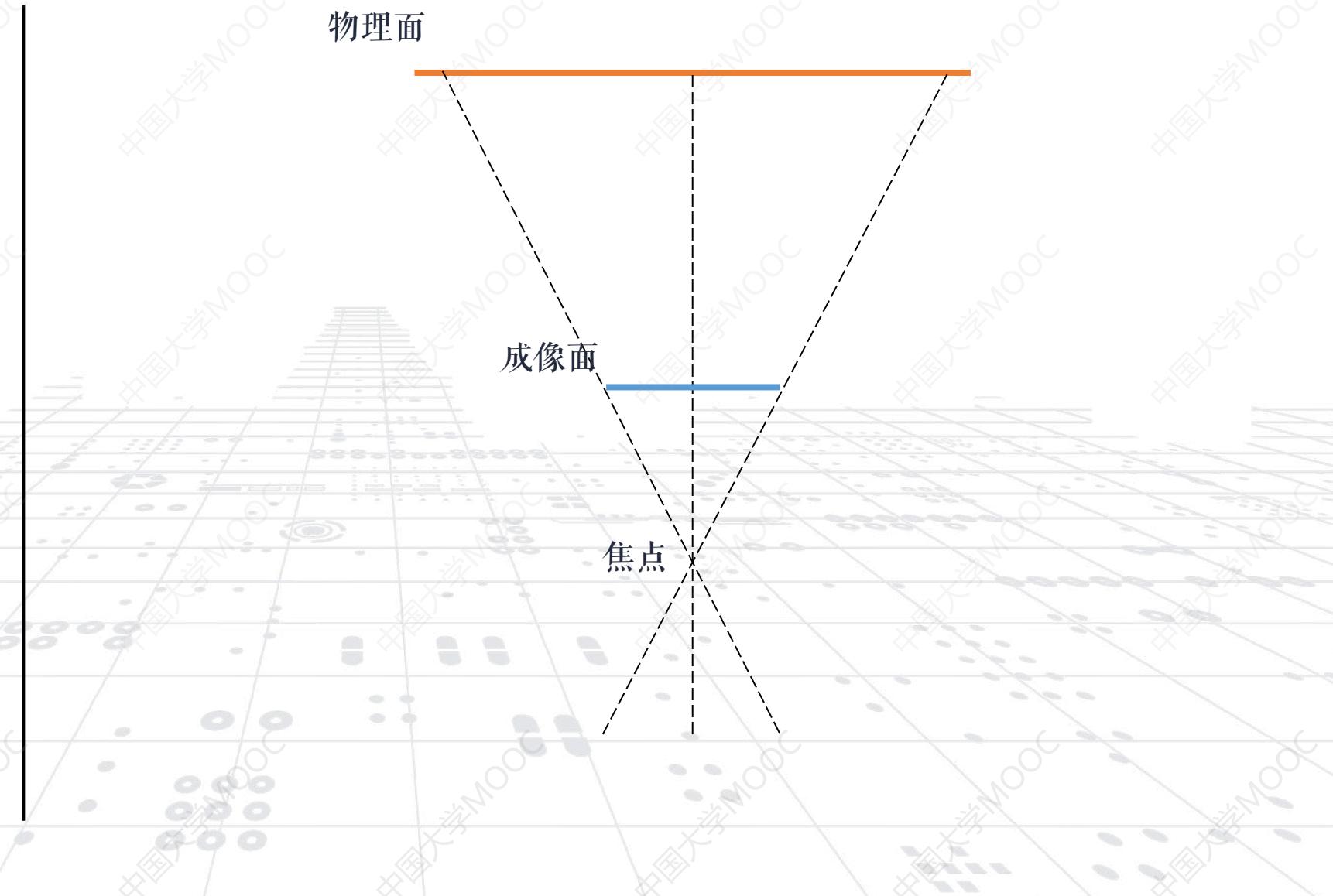
成像面



# 双目相机



Intel realsense d455



# 双目相机

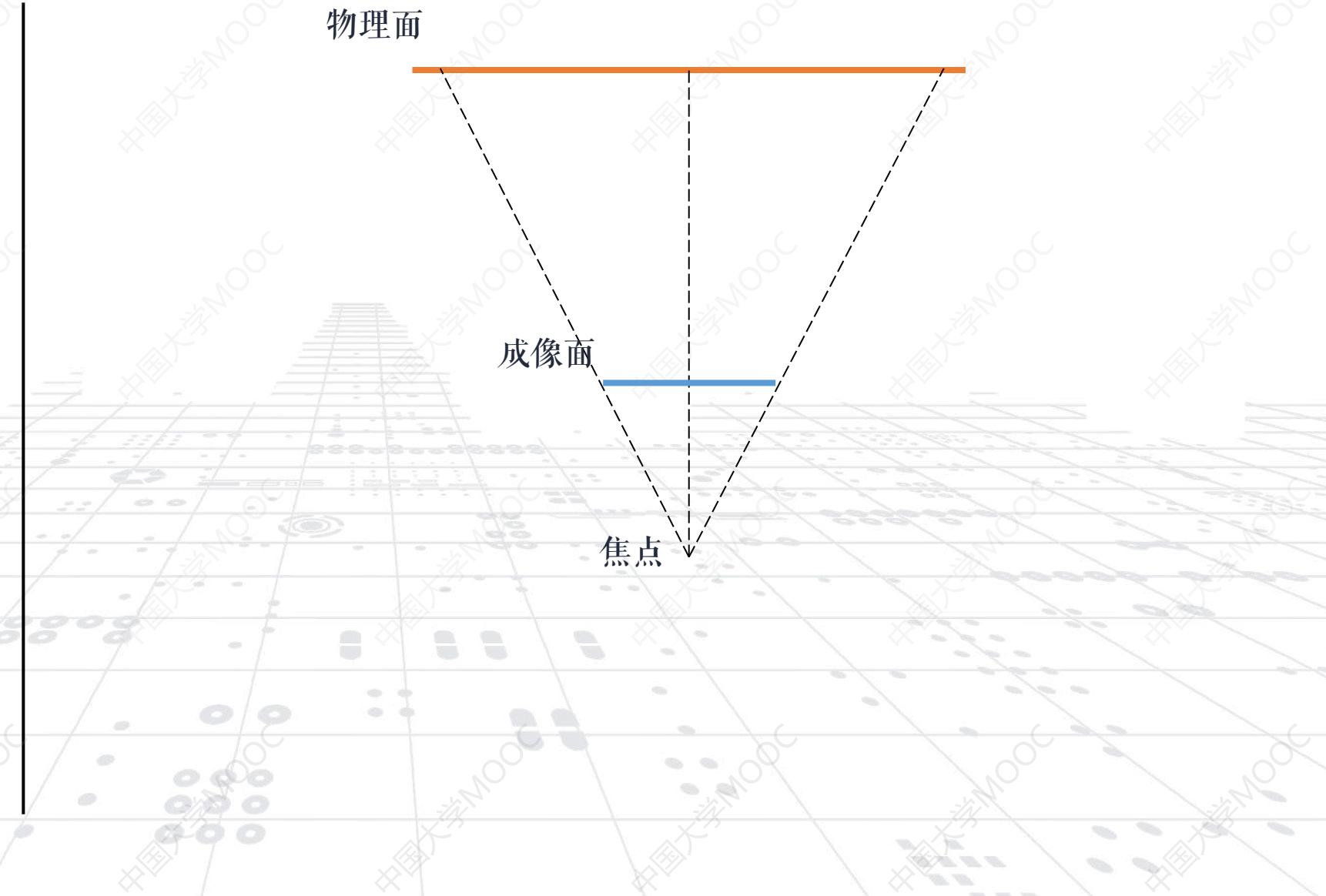


Intel realsense d455

物理面

成像面

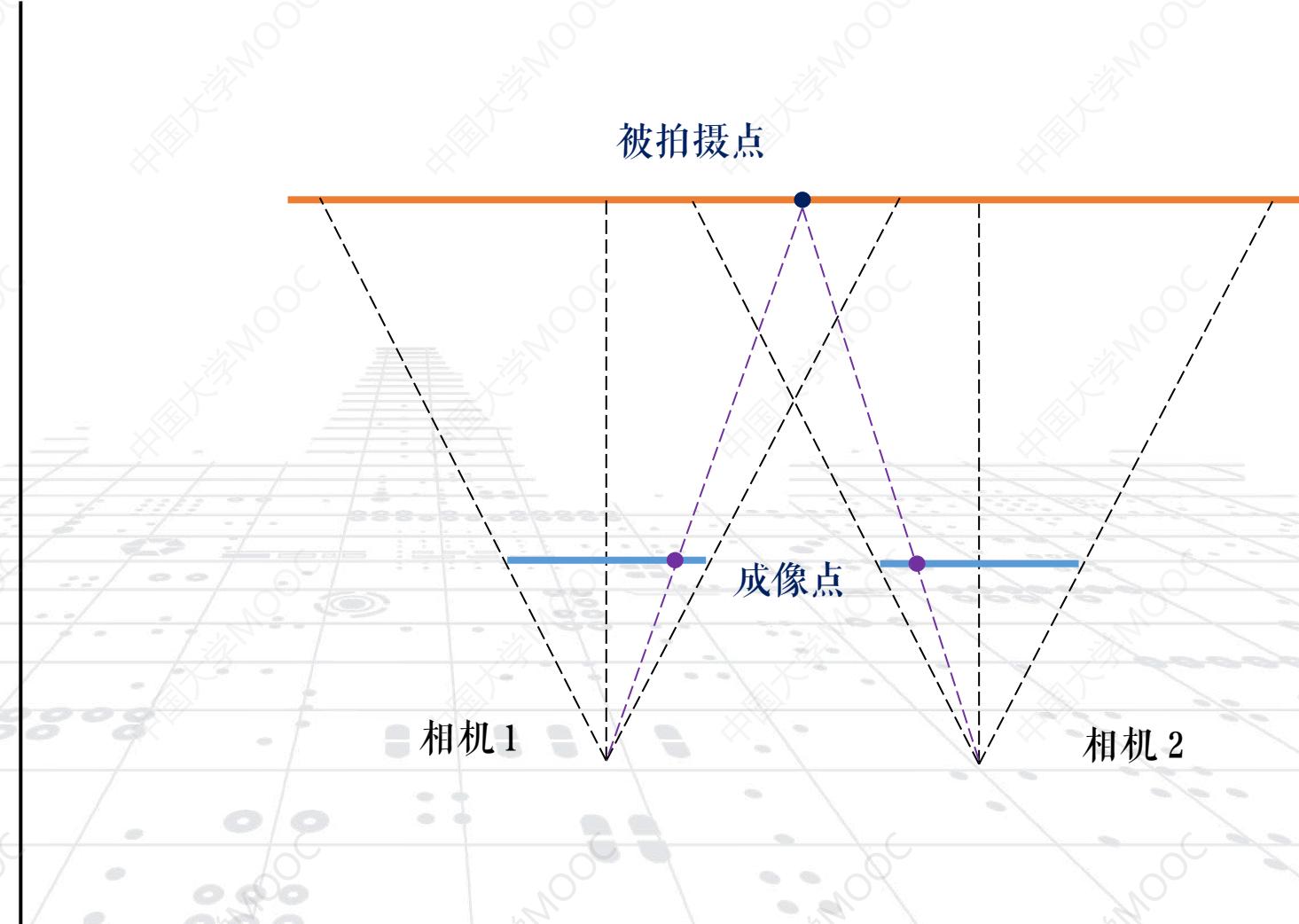
焦点



# 双目相机



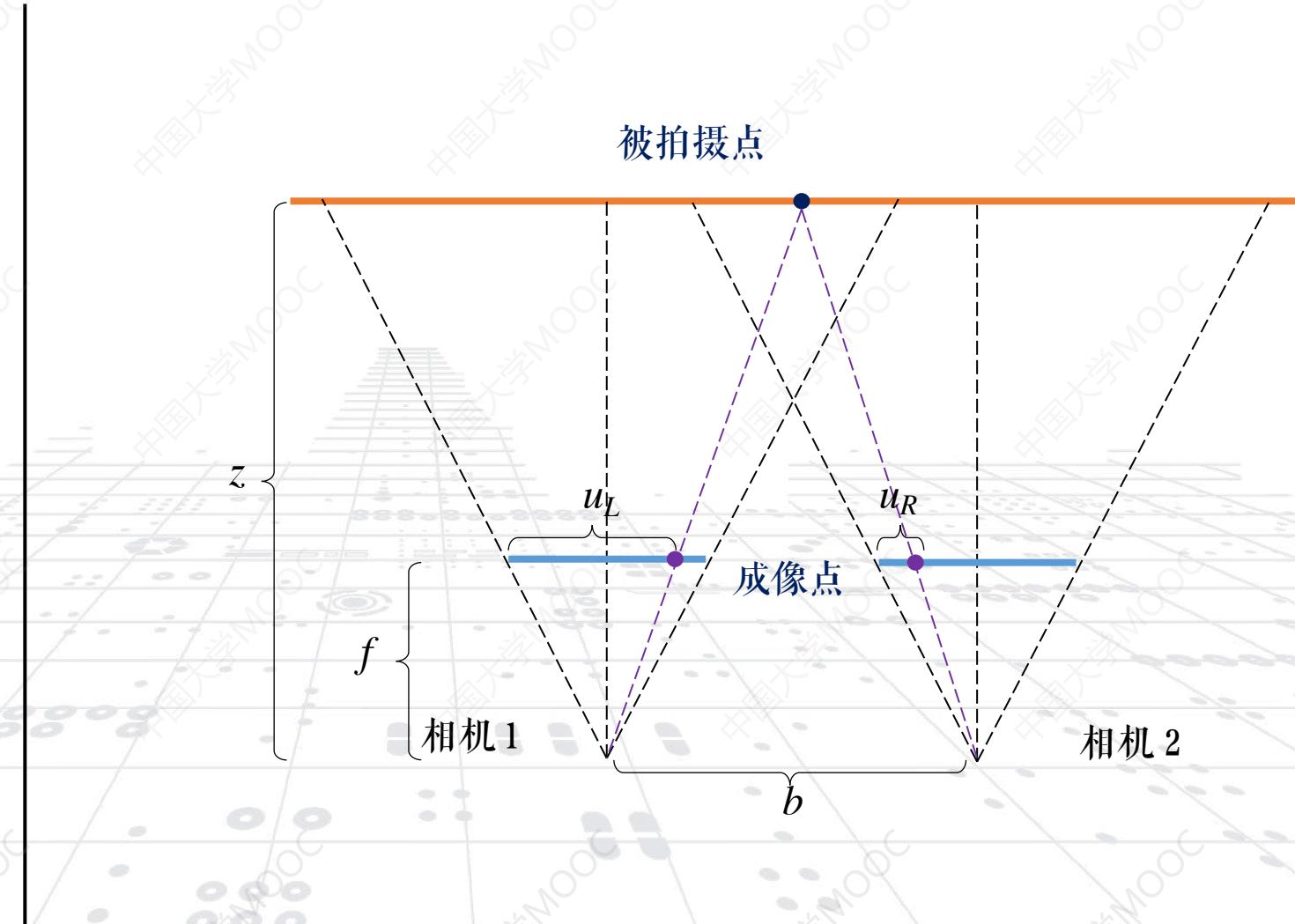
Intel realsense d455



# 双目相机



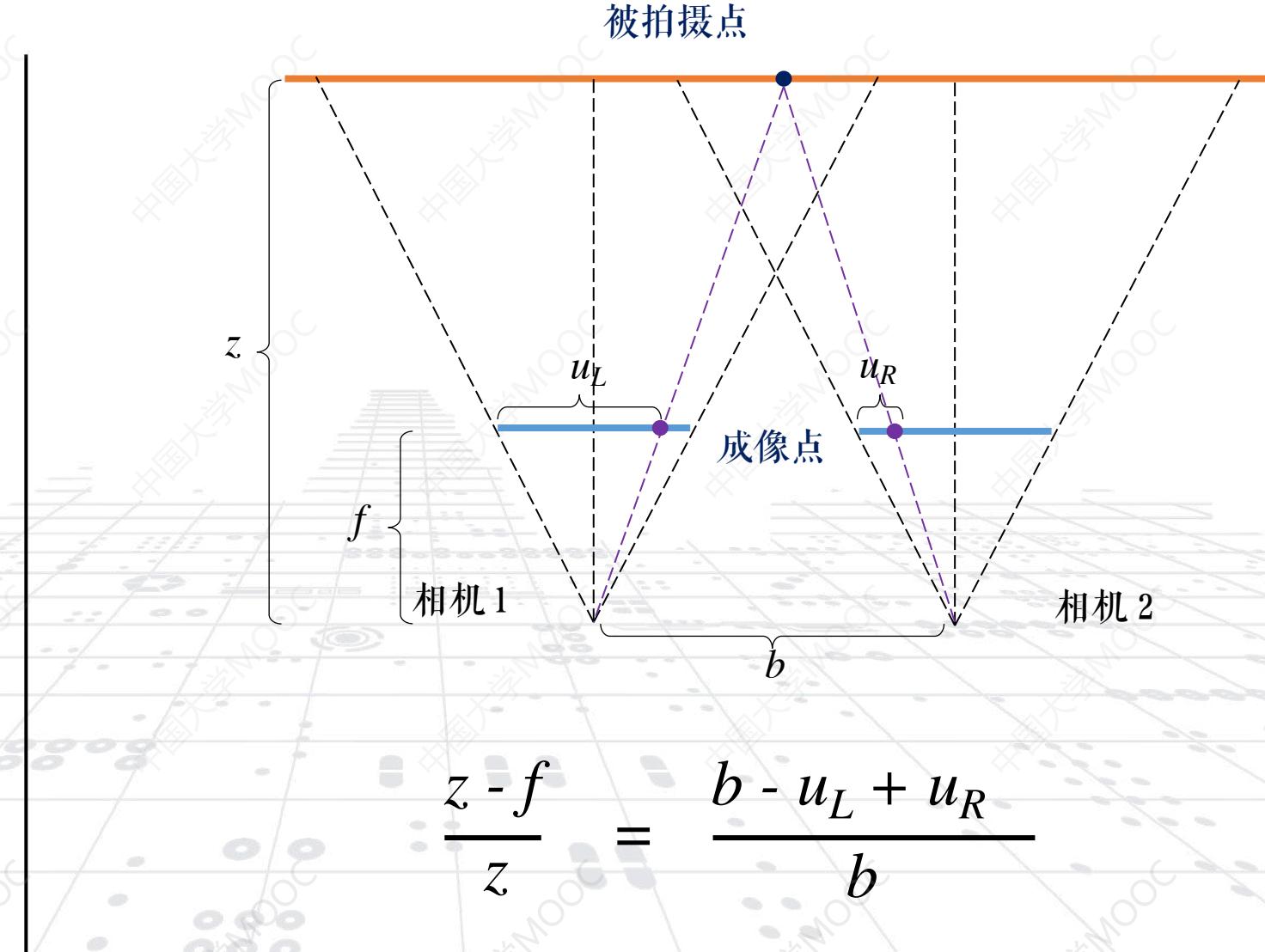
Intel realsense d455



# 双目相机



Intel realsense d455



# 双目相机



Intel realsense d455

## 常见应用



## 2.5 相机

Demo ( Halcon )

# 本章概要

硬件介绍

1

基本概念

2

图像种类

3

色彩模型

4

直方图

5

# 本章概要

硬件介绍

1

基本概念

2

图像种类

3

色彩模型

4

直方图

5

# 图像分辨率

图像分辨率:  $160 \times 128$



水平像素: 160

垂直像素: 128

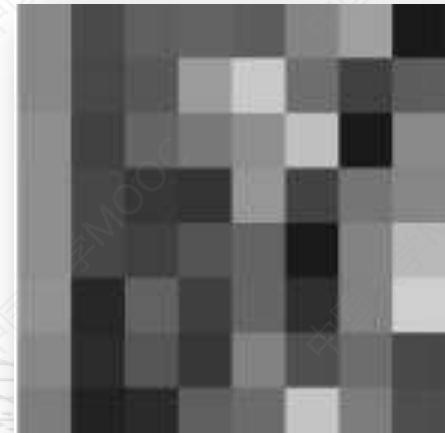
视频分辨率: 1080P



画面:  $1920 \times 1080$

P: 逐行扫描

\* 美国电影电视工程师协会  
( SMPTE ) 标准



分辨率 ↑

细节 ↑

# 图像灰度级

8位图像



灰度级:  $2^8$ , 256级

像素值: 0-255



灰度级 ↑

细节 ↑

# 图像坐标系

[ 0, 0 ]

[  $M-1$ , 0 ]



[  $M-1$ , 0 ]

[  $M-1$ ,  $N-1$  ]

# 图像坐标系

[ 0, 0 ]



[ M-1, 0 ]

矩阵表示

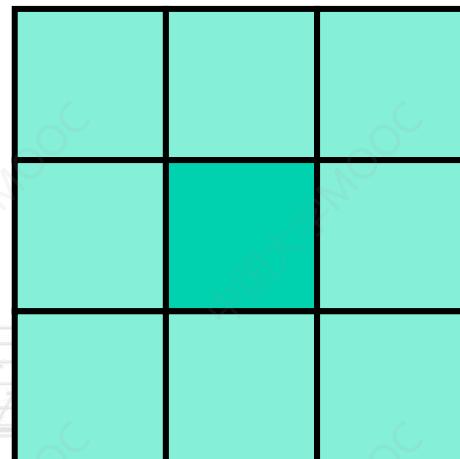
$$\begin{bmatrix} f(0,0) & \cdots & f(0, N - 1) \\ \vdots & \ddots & \vdots \\ f(M - 1, 0) & \cdots & f(M - 1, N - 1) \end{bmatrix}$$

[ M-1, 0 ]

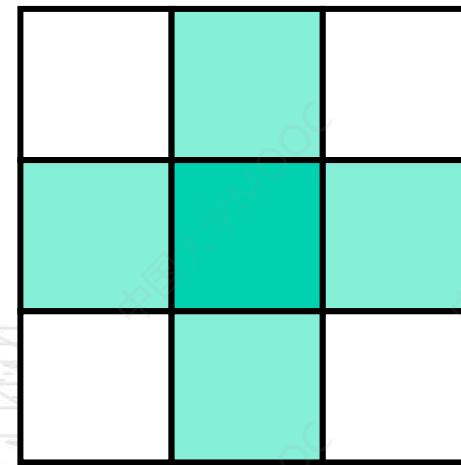
[ M-1, N-1 ]

# 像素空间关系

中心像素  
邻接像素  
非邻接像素



八邻接关系



四邻接关系

# 本章概要

硬件介绍

1

基本概念

2

图像种类

3

色彩模型

4

直方图

5

# 本章概要

硬件介绍

1

基本概念

2

图像种类

3

色彩模型

4

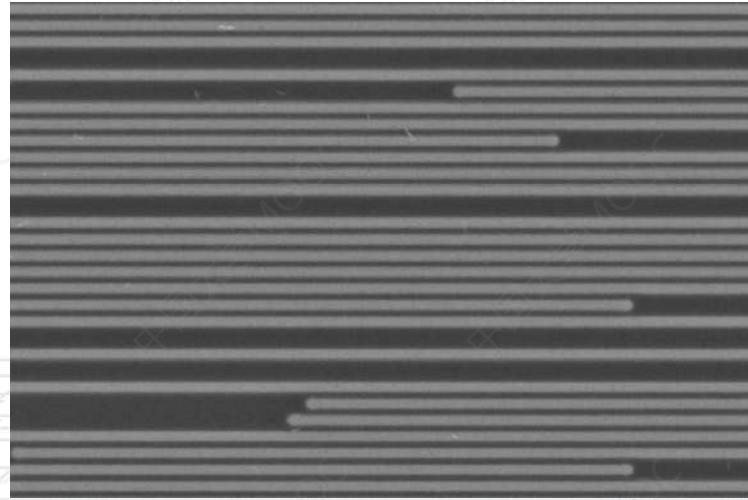
直方图

5

# 二值图像

像素值为0和1  
或者0和255。

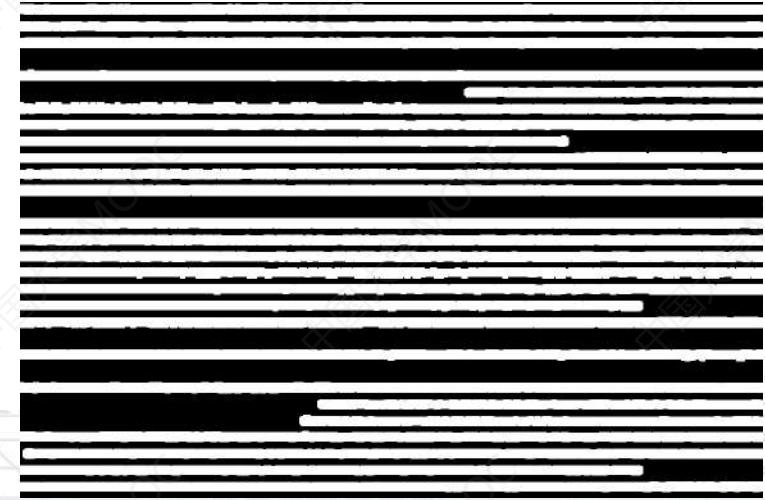
通常作为像  
素标签使用



原图（芯片内部）



背景像素



二值图



前景像素

# 灰度图像

像素值：单通道

取值范围 [0, 255],

0: 纯黑

255: 纯白

※一些图像算法中  
需要使用灰度图进  
行运算。



灰度图像



彩色图像

# 彩色图像



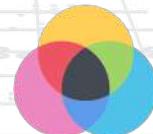
RGB：色彩显示



HSV：色彩描述



YCbCr：视频数据



CMYK：印刷行业

# 本章概要

硬件介绍

1

基本概念

2

图像种类

3

色彩模型

4

直方图

5

# 本章概要

硬件介绍

1

基本概念

2

图像种类

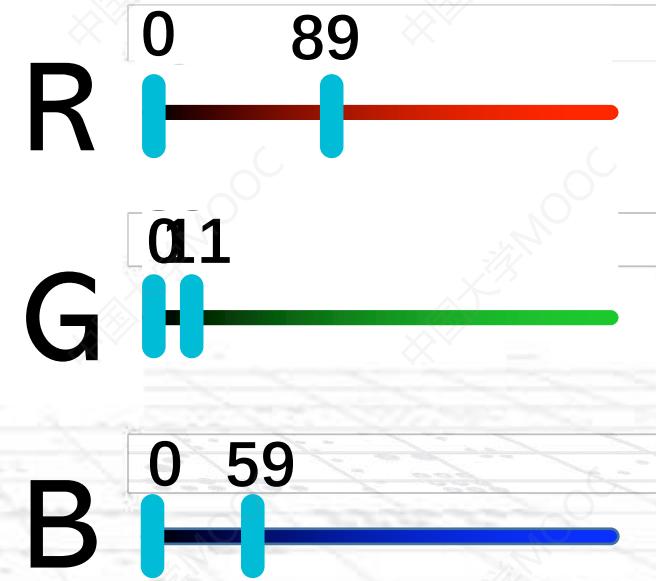
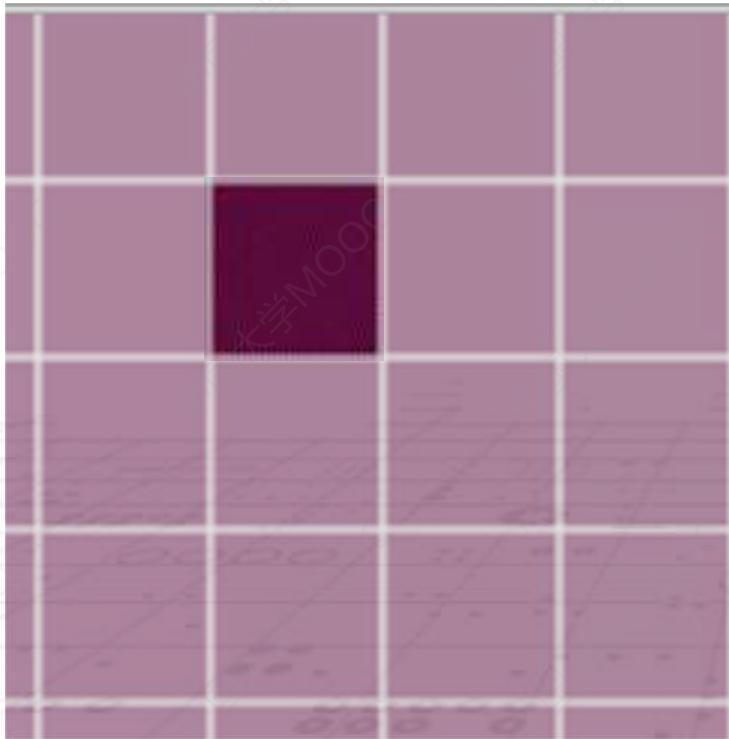
3

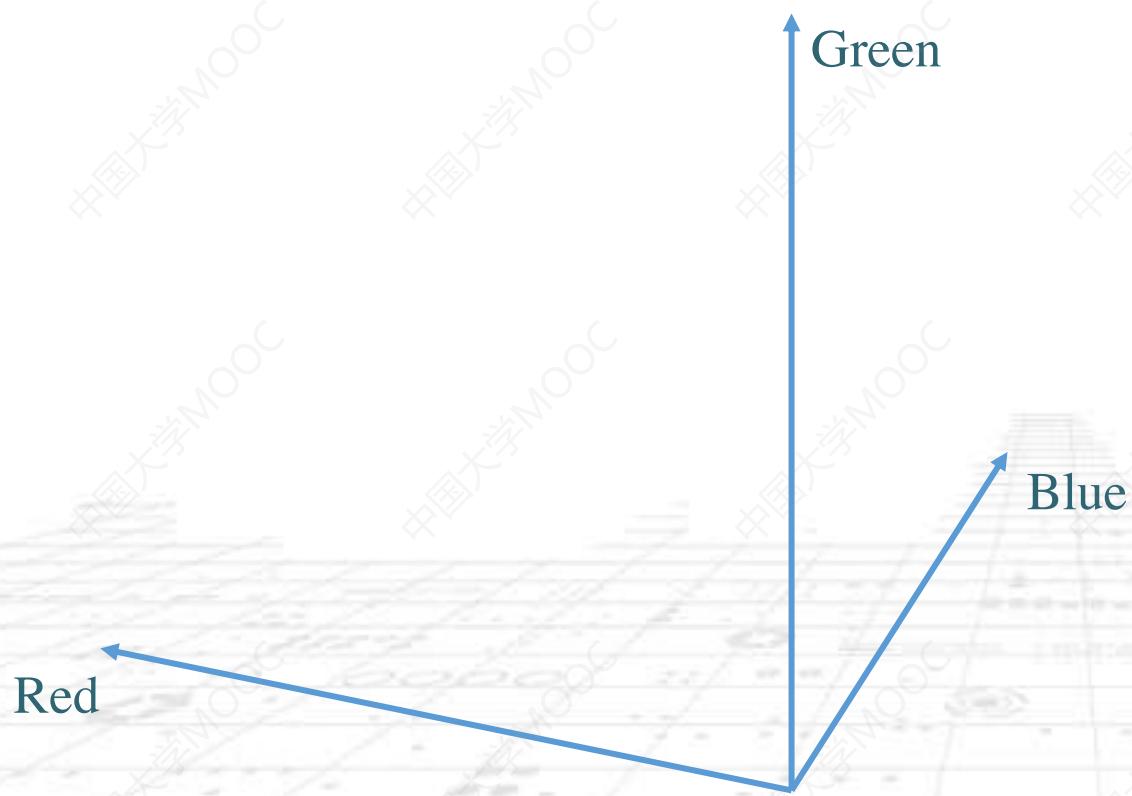
色彩模型

4

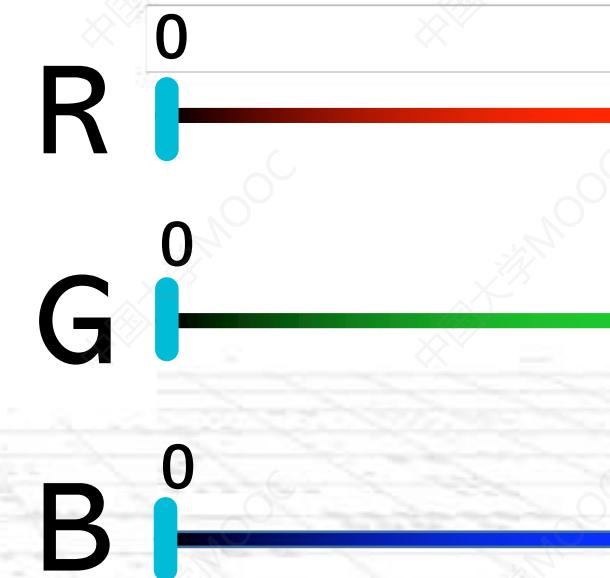
直方图

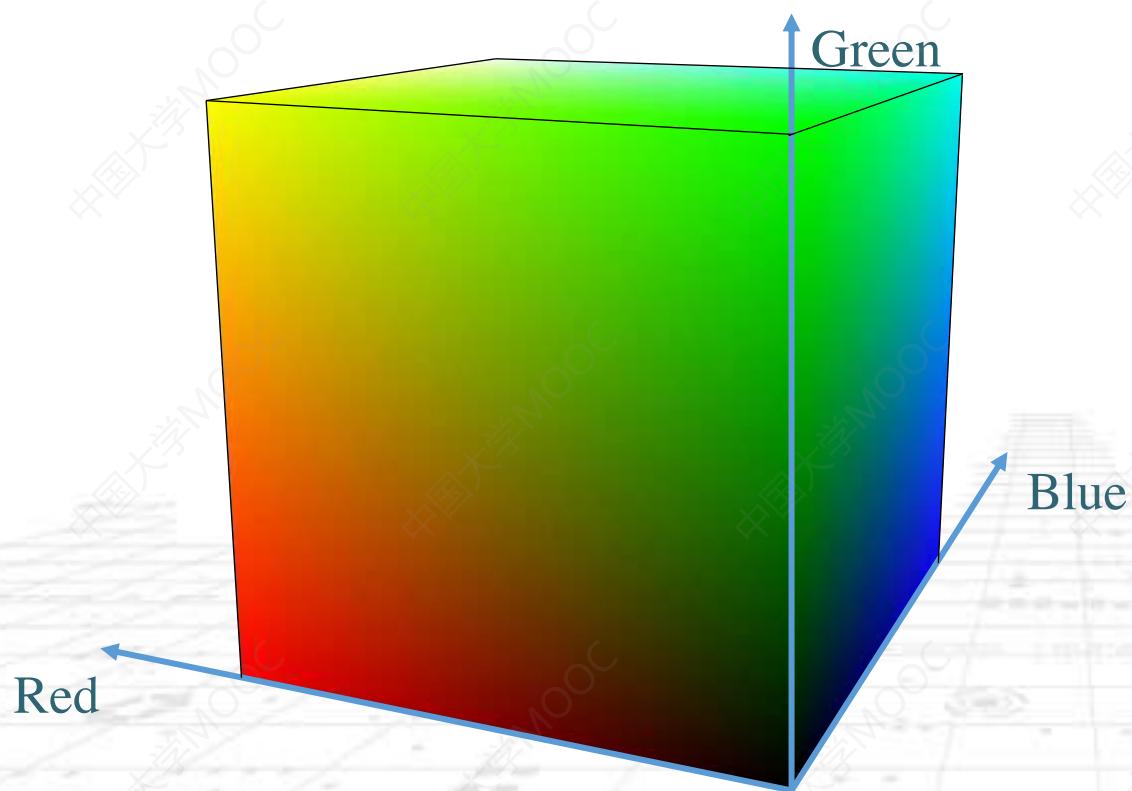
5



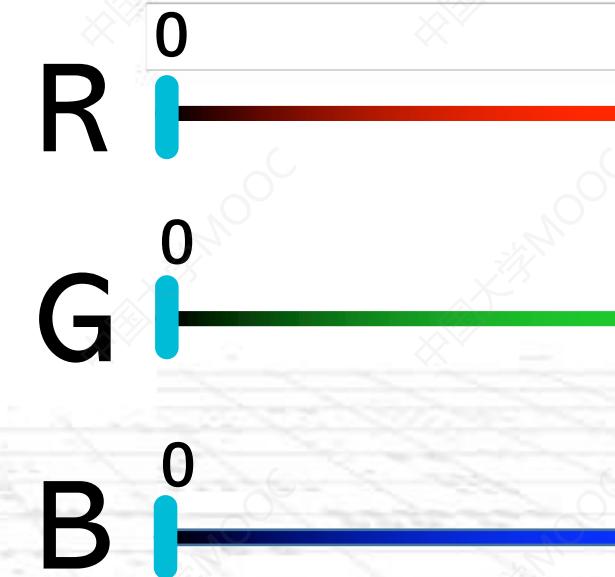


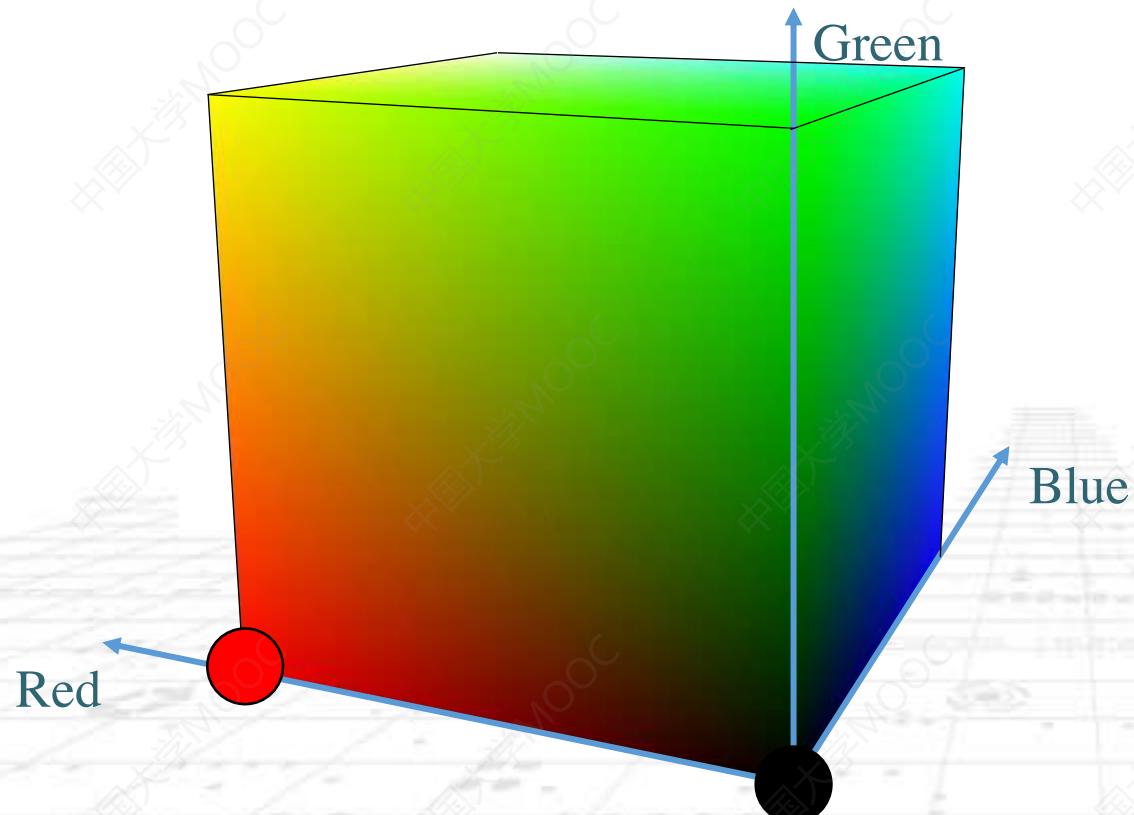
RGB色彩空间



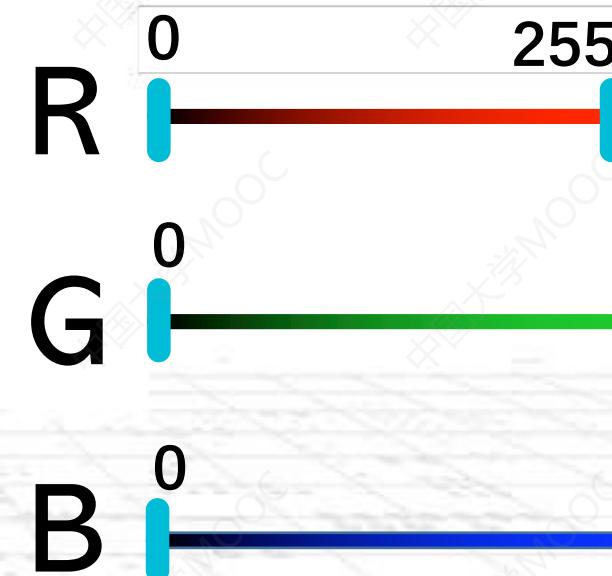


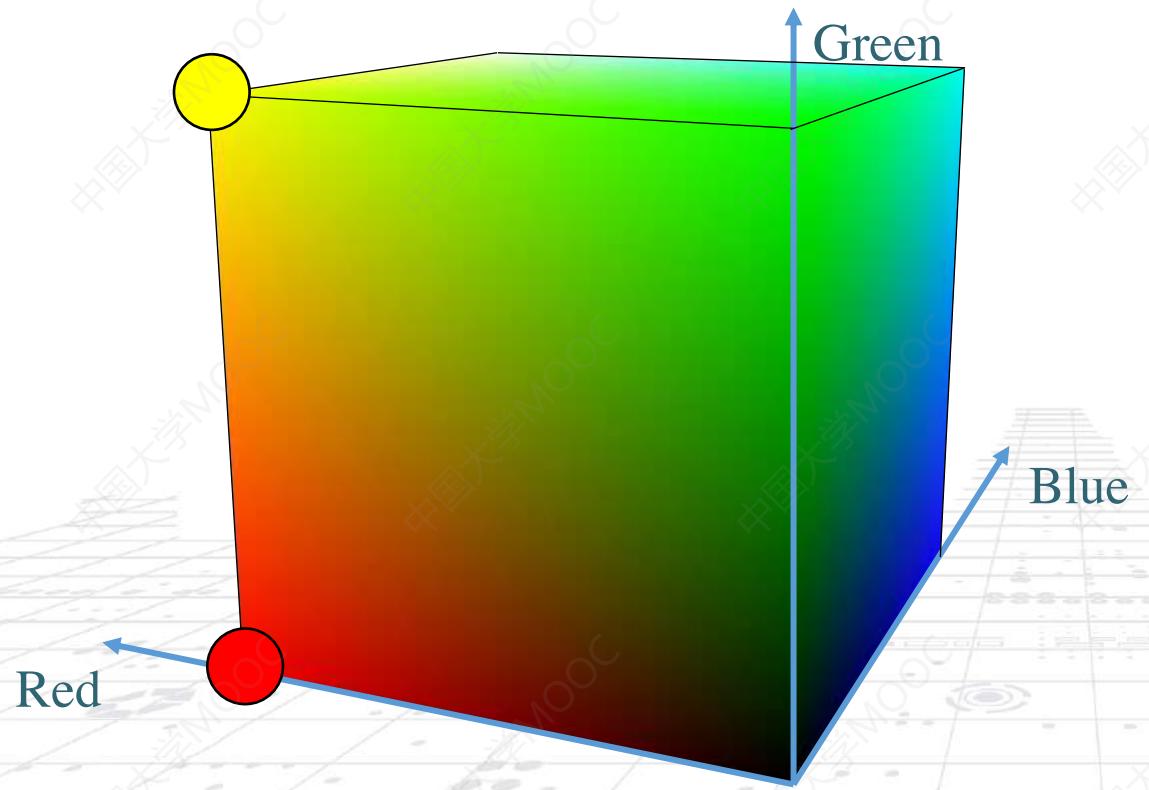
RGB色彩空间



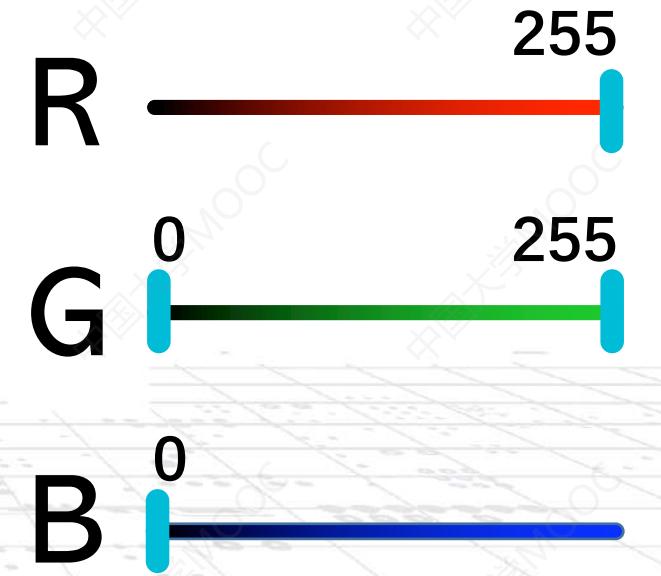


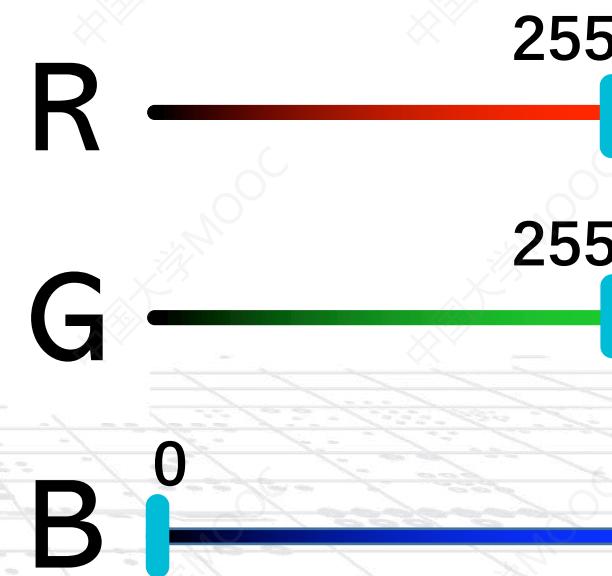
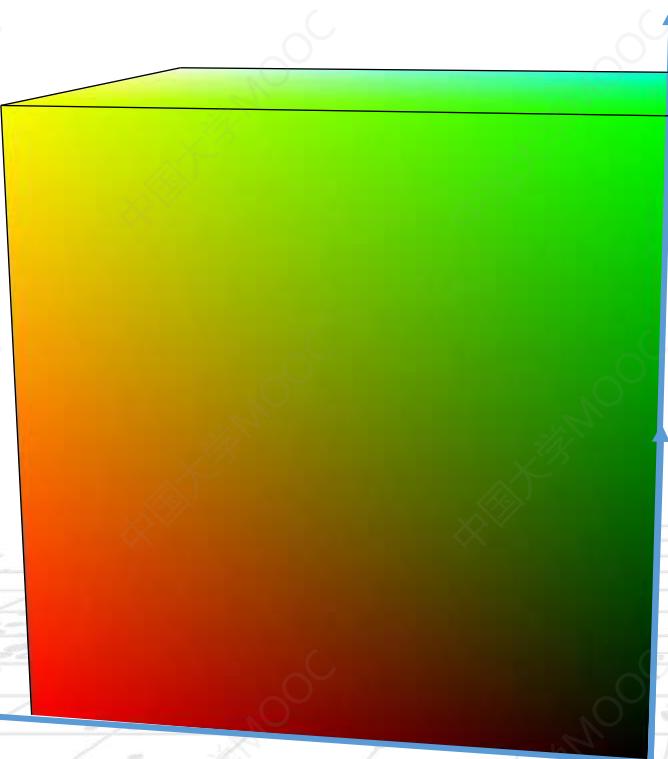
RGB色彩空间

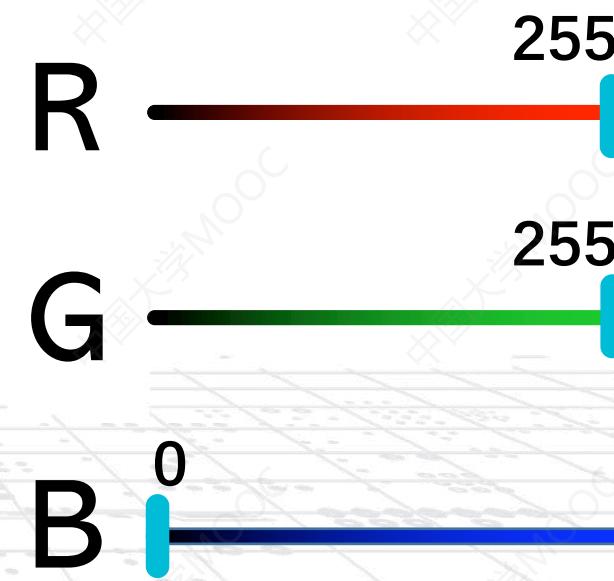
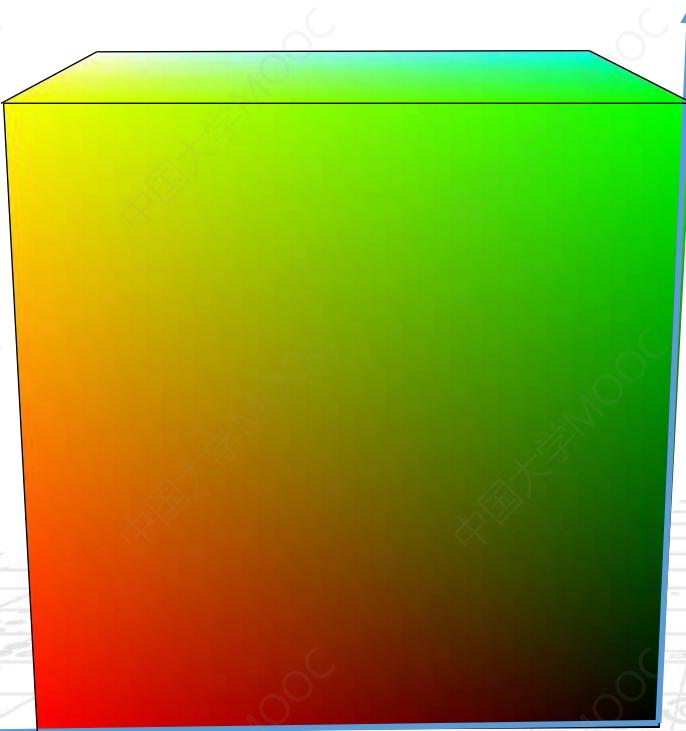


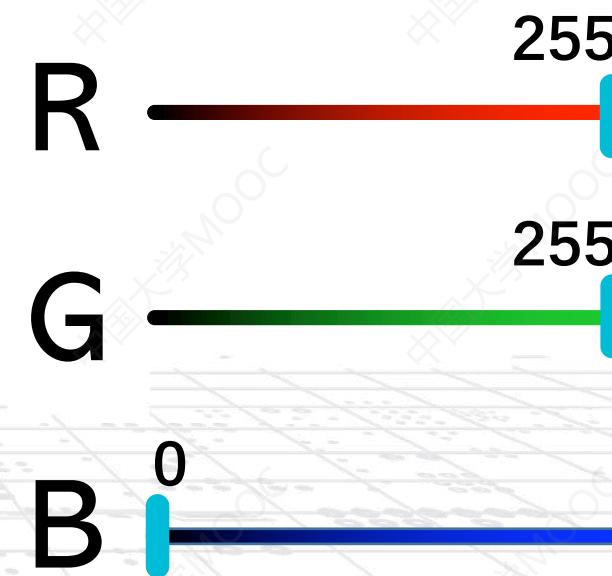
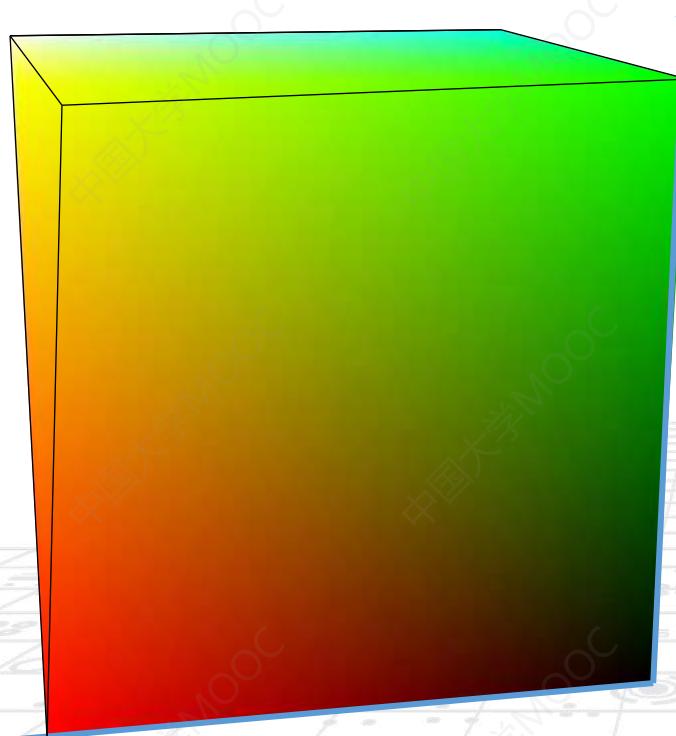


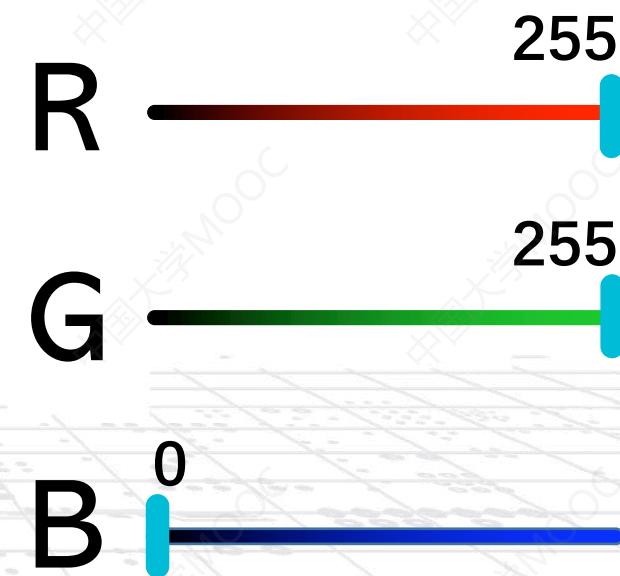
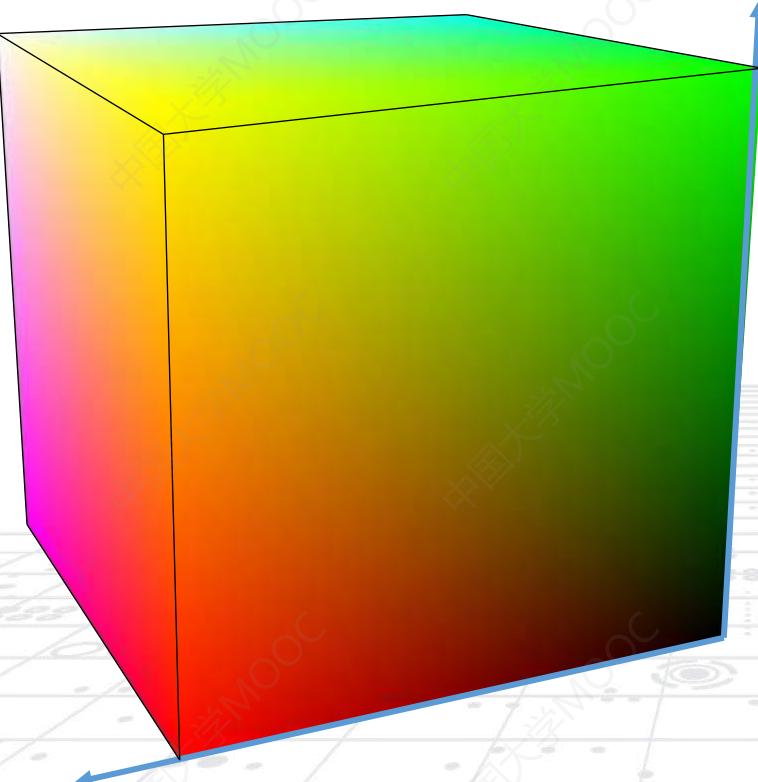
RGB色彩空间

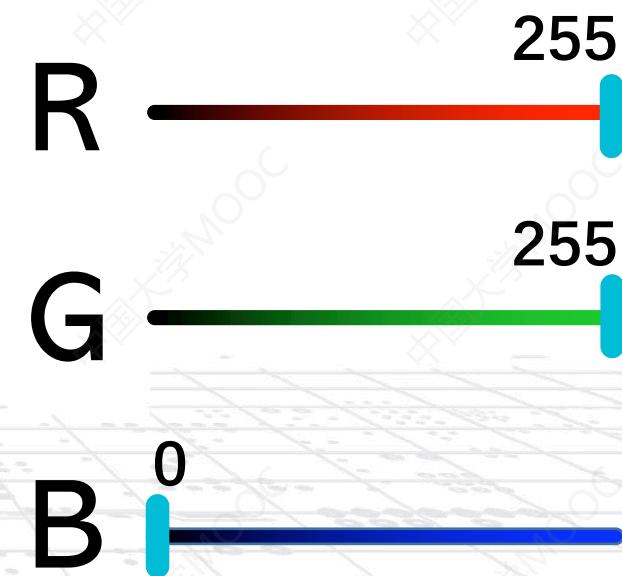
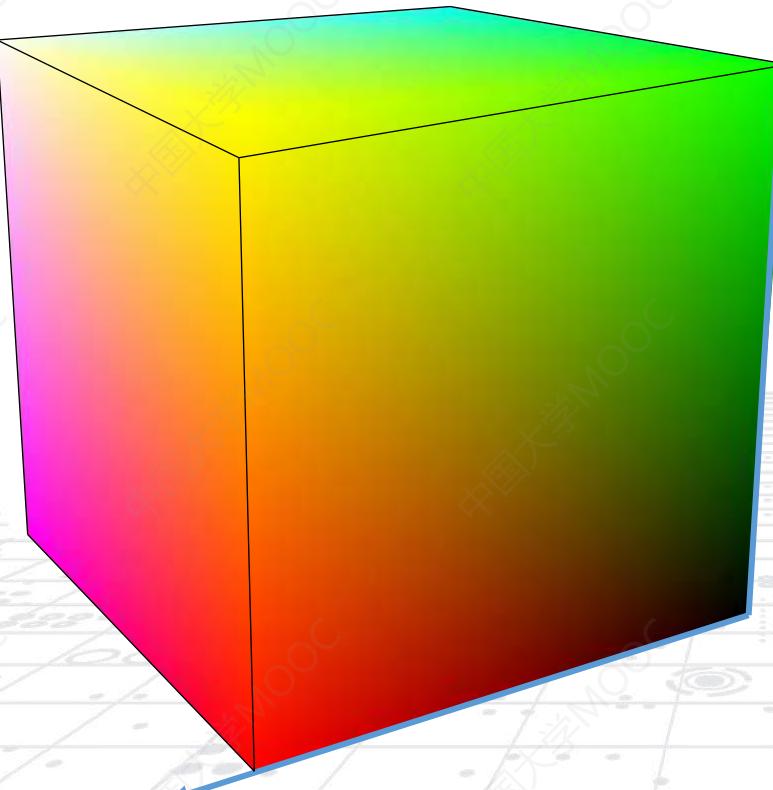


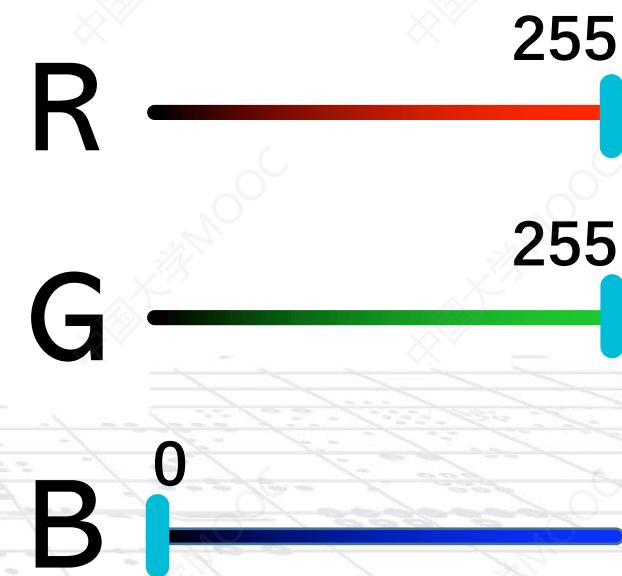
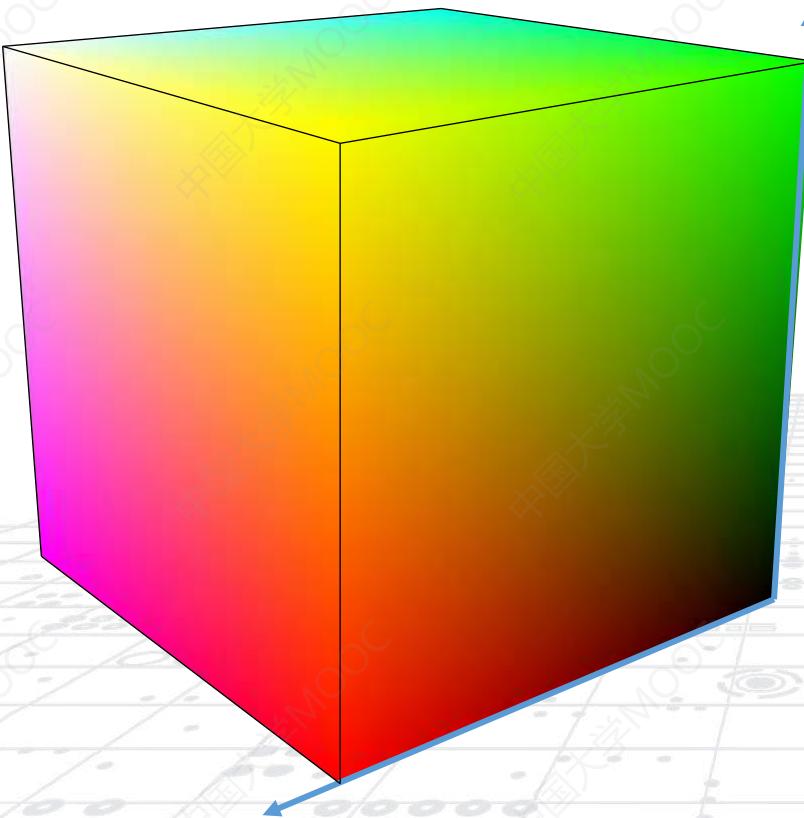


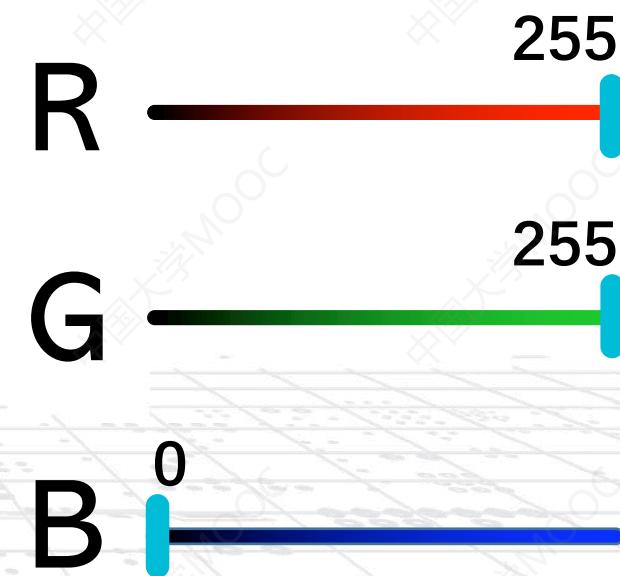
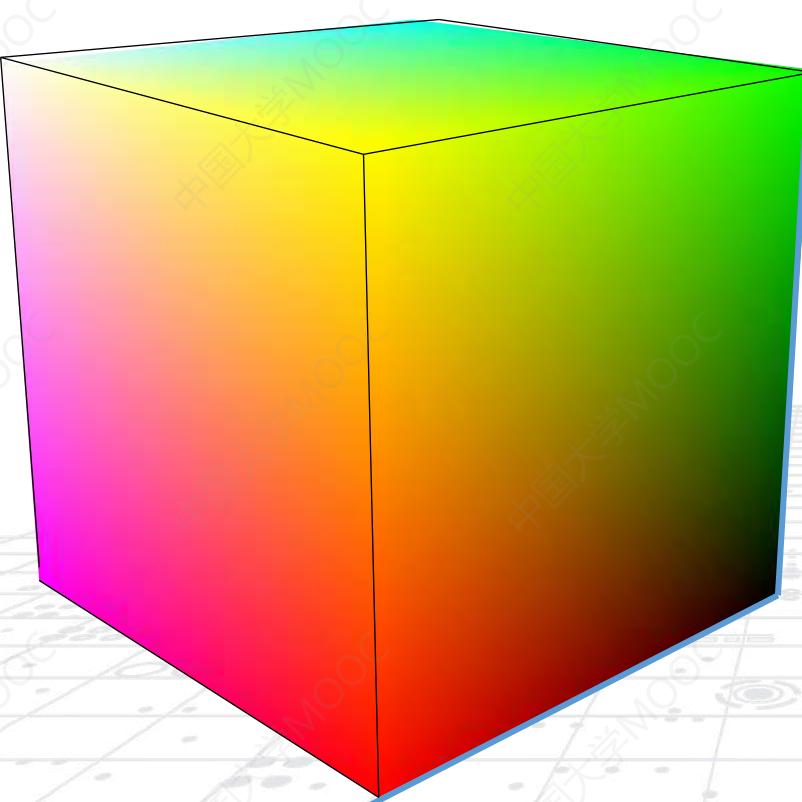


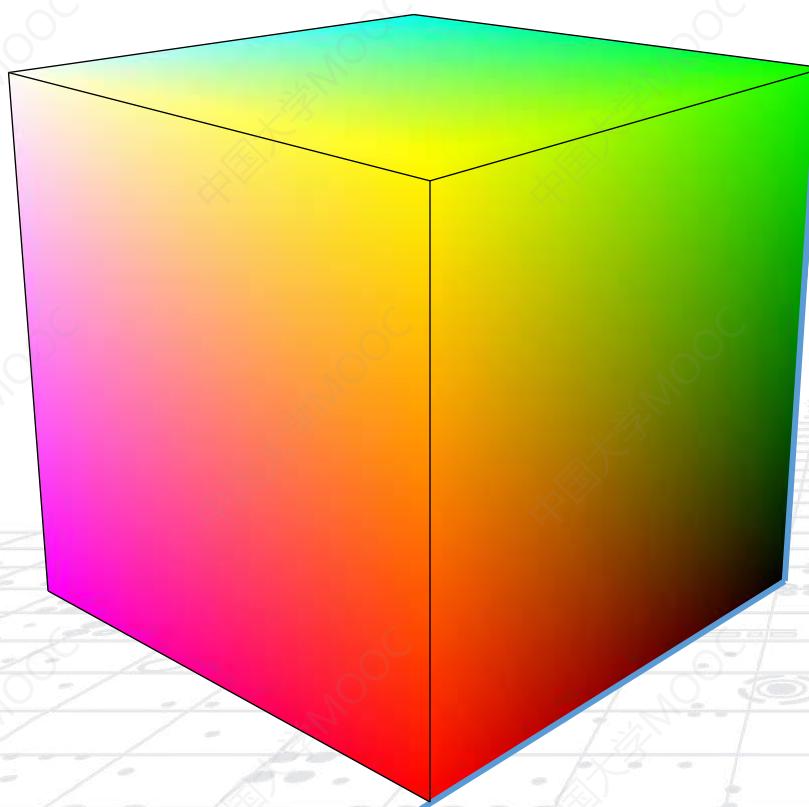




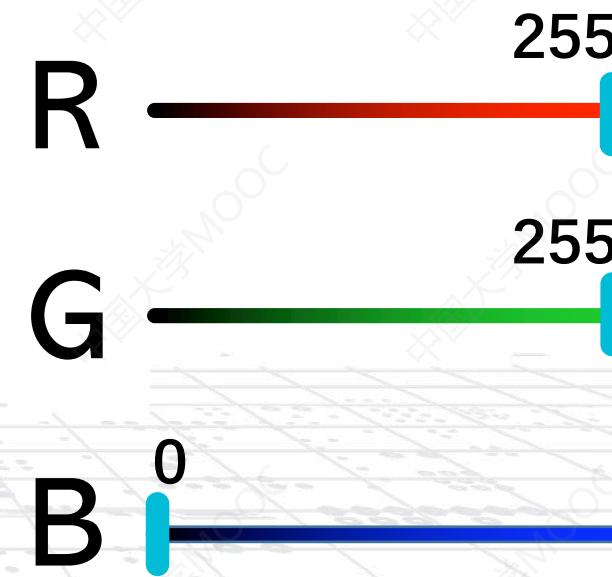


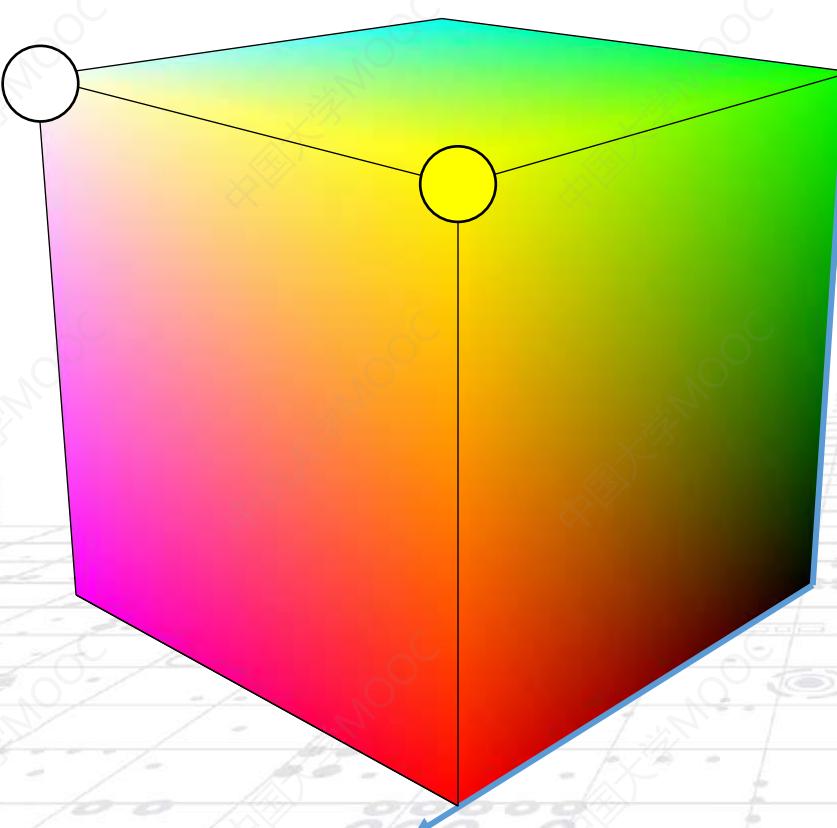




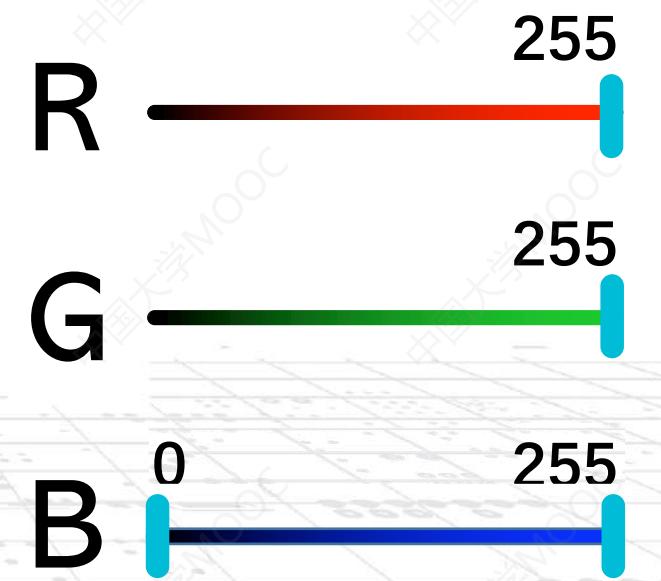


RGB色彩空间



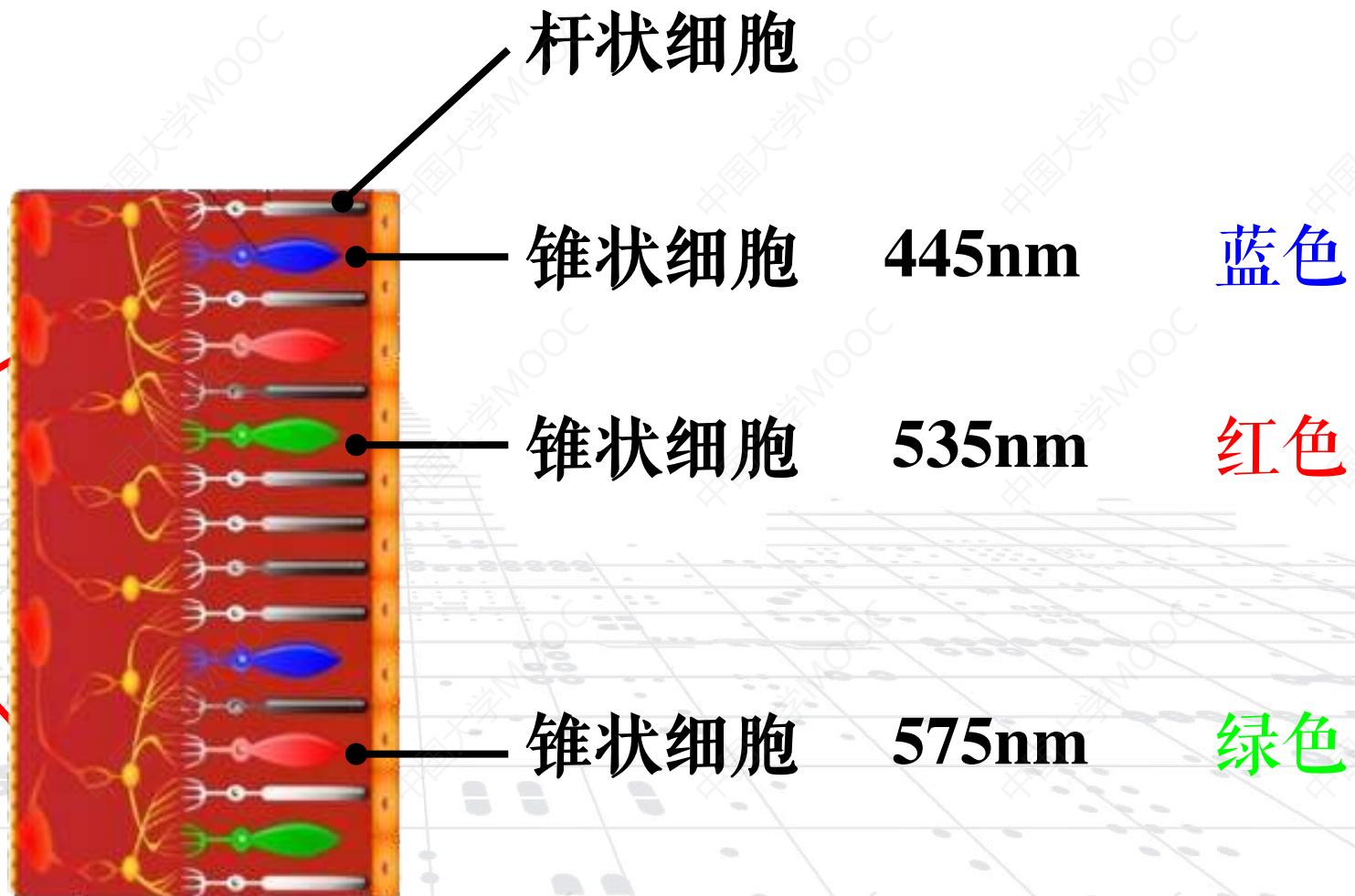
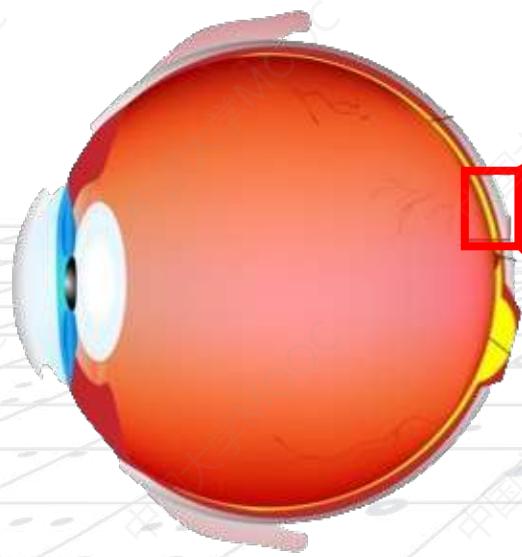


RGB色彩空间



灰度图像  $\longleftrightarrow$  RGB图像

$$\text{Gray} = R * 0.299 + G * 0.587 + B * 0.114$$





针脚	说明	针脚	说明
1	红色	6	红色回路（接地）
2	绿色	7	绿色回路（接地）
3	蓝色	8	蓝色回路（接地）
4	未连接	9	+5V（熔断电流为 250 mA）
5	接地（模拟）	10	接地（同步回路）

1987年，IBM  
VGA(Video Graphics Array)视频图形阵列

# 应用1: 色彩校正

建设进展

大咖解读

太空生活

高清美图

视频记录

幕后故事

# 神舟十二号3名航天员顺利进驻天和核心舱



3名航天员向全国人民敬礼致意

## 回放：神舟十二号与天和核心舱对接

神舟十二号发射圆满成功，在太空飞行数小时后，神舟十二号载人飞船将与天和核心舱交会对接，3名航天员依次进入空间站组合体。

- 太空WiFi怎么用？揭开中国空间站背后的“奥秘”
- 神舟十二号与天和核心舱完成自主快速交会对接
- 回放：神舟十二号载人飞船发射任务
- 1分钟模拟动画带你了解神舟十二号任务全流程
- 神舟十二号发射成功背后“小”装置发挥“大”作用

航天员太空生活“剧透”

神舟十二号飞行任务四大特点

航天员见面会视频

天和升空

天舟二号揭秘





# 标准色板





标准色板



标准色板



# 自动颜色校正算法



标准色板



# 自动颜色校正算法

## 自动颜色定位算法



标准色板

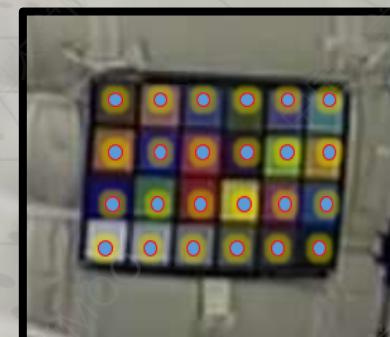


# 自动颜色校正算法

自动定位  
▼  
读取色值



标准色板



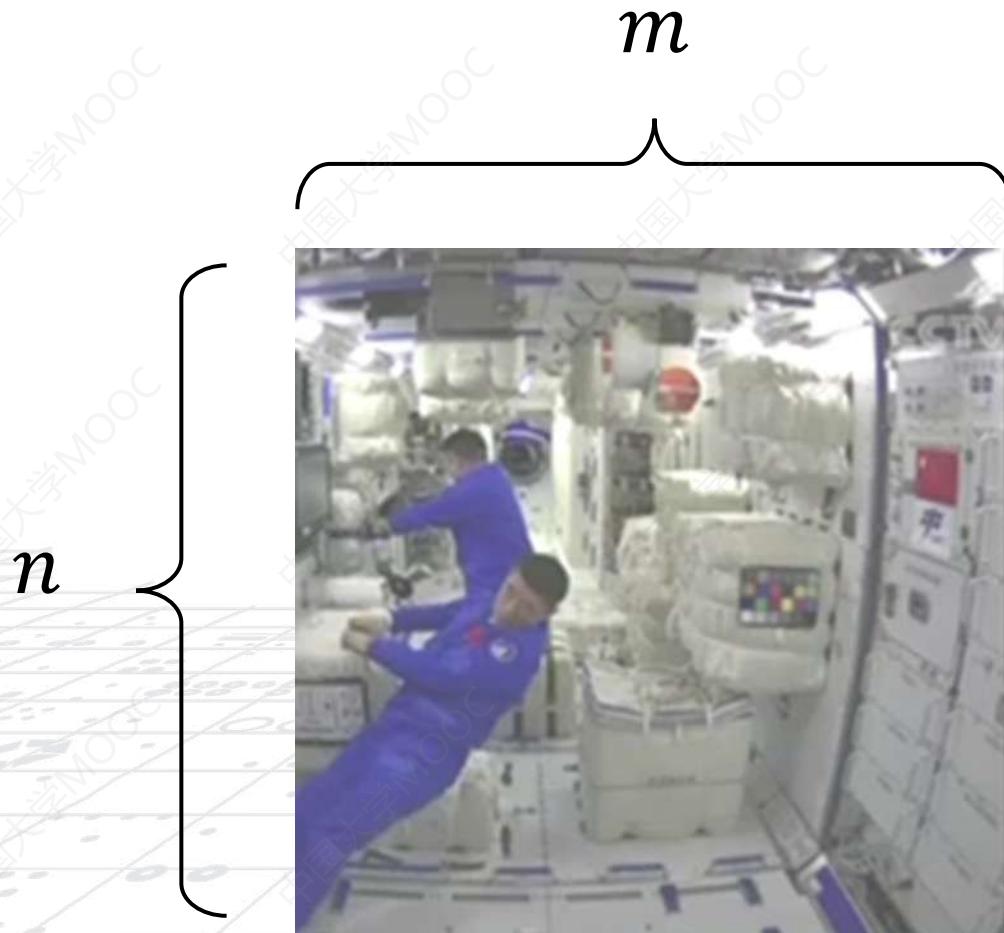
# 自动颜色校正算法

自动定位  
▼  
读取色值  
▼  
颜色校正



标准色板





$$k = m \times n$$

$$\begin{bmatrix} r_1 & g_1 & b_1 \\ r_2 & g_2 & b_2 \\ \vdots & \vdots & \vdots \\ r_k & g_k & b_k \end{bmatrix}$$

拍摄得到



$$\begin{bmatrix} r_1 & g_1 & b_1 \\ r_2 & g_2 & b_2 \\ \vdots & \vdots & \vdots \\ r_k & g_k & b_k \end{bmatrix}$$

拍摄得到

$$\begin{bmatrix} r_1 & g_1 & b_1 \\ r_2 & g_2 & b_2 \\ \vdots & \vdots & \vdots \\ r_k & g_k & b_k \end{bmatrix}$$

拍摄得到



$$\begin{bmatrix} R_1 & G_1 & B_1 \\ R_2 & G_2 & B_2 \\ \vdots & \vdots & \vdots \\ R_k & G_k & B_k \end{bmatrix} = \begin{bmatrix} r_1 & g_1 & b_1 \\ r_2 & g_2 & b_2 \\ \vdots & \vdots & \vdots \\ r_k & g_k & b_k \end{bmatrix}$$

校正结果



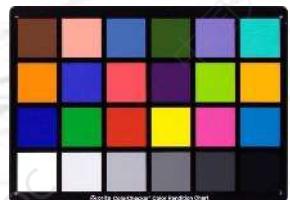
$$\begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix}$$

拍摄得到

Color Correction Matrix  
(CCM)



$$\begin{bmatrix} R'_1 & G'_1 & B'_1 \\ R'_2 & G'_2 & B'_2 \\ \vdots & \vdots & \vdots \\ R'_{24} & G'_{24} & B'_{24} \end{bmatrix}$$



校色板真值

$$- \begin{bmatrix} r'_1 & g'_1 & b'_1 \\ r'_2 & g'_2 & b'_2 \\ \vdots & \vdots & \vdots \\ r'_{24} & g'_{24} & b'_{24} \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix}$$



校色板  
拍摄值

Color Correction Matrix  
(CCM)

## 通过拉格朗日乘数法求解

$$\begin{bmatrix} R'_1 \Delta R_1 & G'_1 & B'_1 \\ \Delta R_2 & \Delta G_2 & \Delta B_2 \\ \vdots & \vdots & \vdots \\ \Delta R_{24} & \Delta G_{24} & \Delta B_{24} \\ R'_{24} & G'_{24} & B'_{24} \end{bmatrix} = \begin{bmatrix} R'_1 & G'_1 r'_1 B'_1 & g'_1 \\ R'_2 & G'_2 r'_2 B'_2 & g'_2 \\ \vdots & \vdots r'_2 \vdots & \vdots g'_2 \\ G'_{24} & B'_{24} & \vdots \\ r'_{24} & g'_{24} & b'_{24} \end{bmatrix} - \begin{bmatrix} b'_1 r'_1 \\ b'_2 r'_2 \\ \vdots \\ b'_{24} r'_{24} \end{bmatrix} \begin{bmatrix} g'_1 & b'_1 \\ A_{11} & b' A_1 \\ \vdots & \vdots A_{21} \\ A_{31} & b' A_{31} \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ 2A_{21} & A_{13} & A_{23} \\ A_{31} & A_{23} & A_{33} \end{bmatrix}$$

最小化均方差:  $\varepsilon = \sum_{i=1}^n (\Delta R_i^2 + \Delta G_i^2 + \Delta B_i^2)$

白平衡约束:

$$\begin{aligned} & A_{11} + A_{12} + A_{13} \\ & = A_{21} + A_{22} + A_{23} \\ & = A_{31} + A_{32} + A_{33} \end{aligned}$$

## 应用2: 颜色分离

Demo

1978年， Ray Smith

## HSV色彩模型

H (Hue): 色 调

S (Saturation): 饱和度

V (Value): 亮 度

The screenshot shows a scholar profile for Alvy Ray Smith. At the top is a circular portrait of a smiling man with white hair and a beard. To the right is a blue circular icon with a white envelope symbol. Below the portrait, the name "Alvy Ray Smith" is displayed, followed by the title "independent scholar". A note states "No verified email - [Homepage](#)". Below this, three research interests are listed: "computer graphics", "cellular automata", and "computer history". The main section is titled "ARTICLES" and contains a list of publications. A red dashed box highlights the first two entries:

ARTICLE	CITED BY
Color gamut transform pairs AR Smith ACM Siggraph Computer Graphics 12 (3), 12-19, 1978	1584
Plants, fractals, and formal languages AR Smith ACM SIGGRAPH Computer Graphics 18 (3), 1-10, 1984	692
Blue screen matting	611

# RGB → HSV

$$R' = R/255$$

$$G' = G/255$$

$$B' = B/255$$

$$C_{max} = \max(R', G', B')$$

$$C_{min} = \min(R', G', B')$$

$$\Delta = C_{max} - C_{min}$$

$$H = \begin{cases} 0^\circ & , \Delta = 0 \\ 60^\circ \times \left( \frac{G' - B'}{\Delta} \bmod 6 \right) & , C_{max} = R' \\ 60^\circ \times \left( \frac{B' - G'}{\Delta} + 2 \right) & , C_{max} = G' \\ 60^\circ \times \left( \frac{B' - G'}{\Delta} + 4 \right) & , C_{max} = B' \end{cases}$$

$$S = \begin{cases} 0 & , C_{max} = 0 \\ \frac{\Delta}{C_{max}} & , C_{max} \neq 0 \end{cases}$$

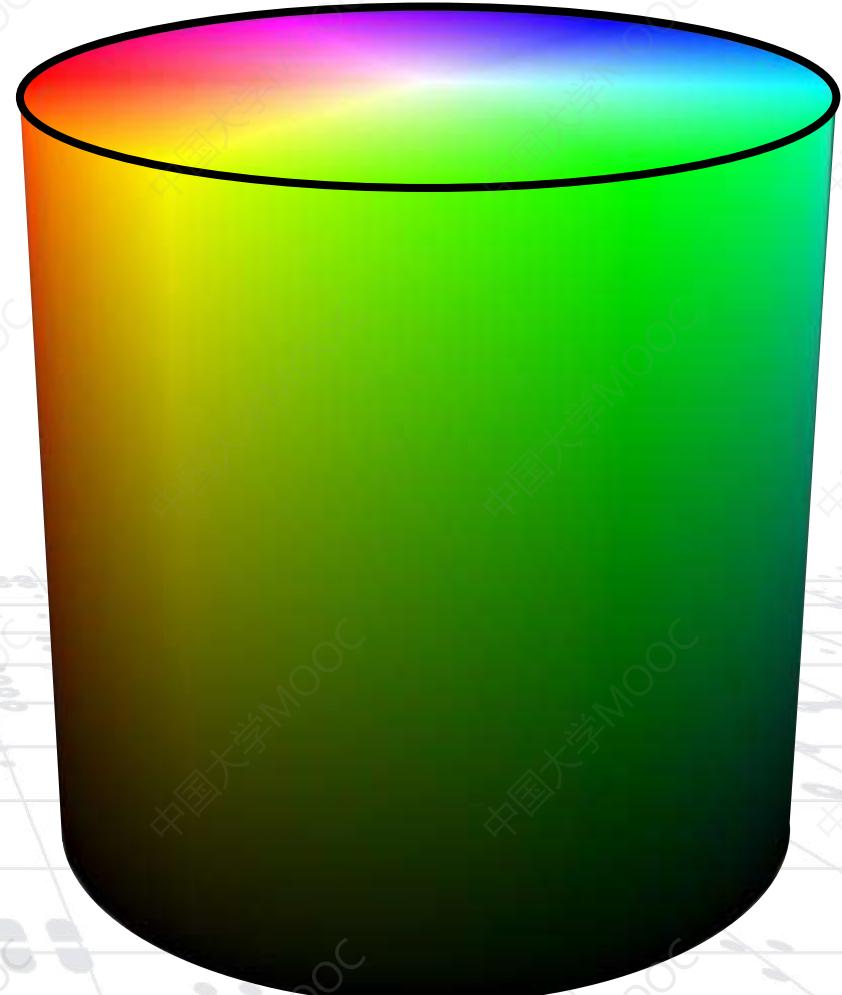
$$V = C_{max}$$

# HSV色彩模型

**H (Hue):**

**S (Saturation):**

**V (Value):**

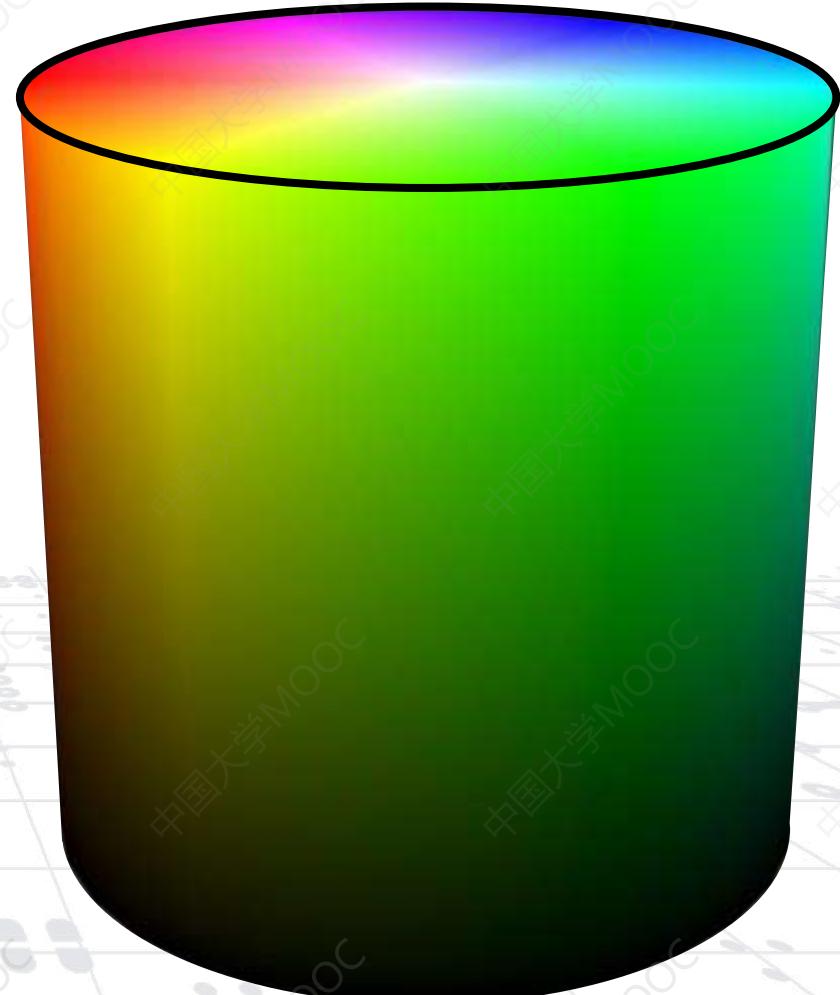


# HSV色彩模型

**H (Hue):**

**S (Saturation):**

**V (Value):**

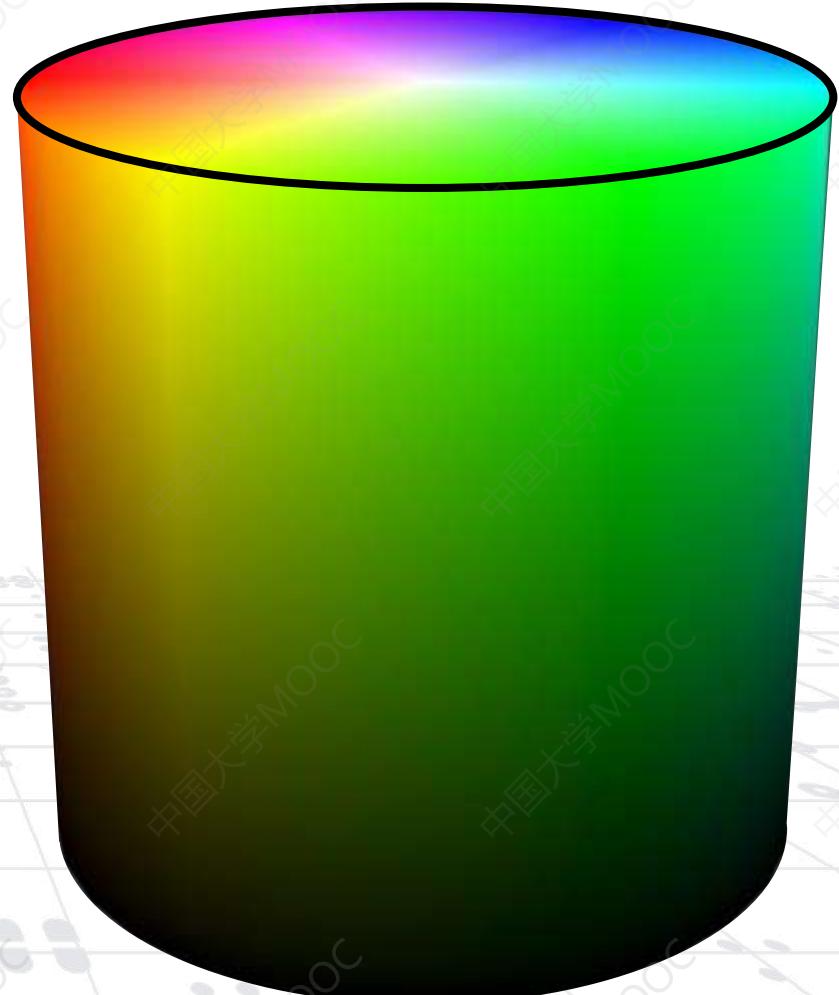


# HSV色彩模型

**H (Hue):**

**S (Saturation):**

**V (Value):**

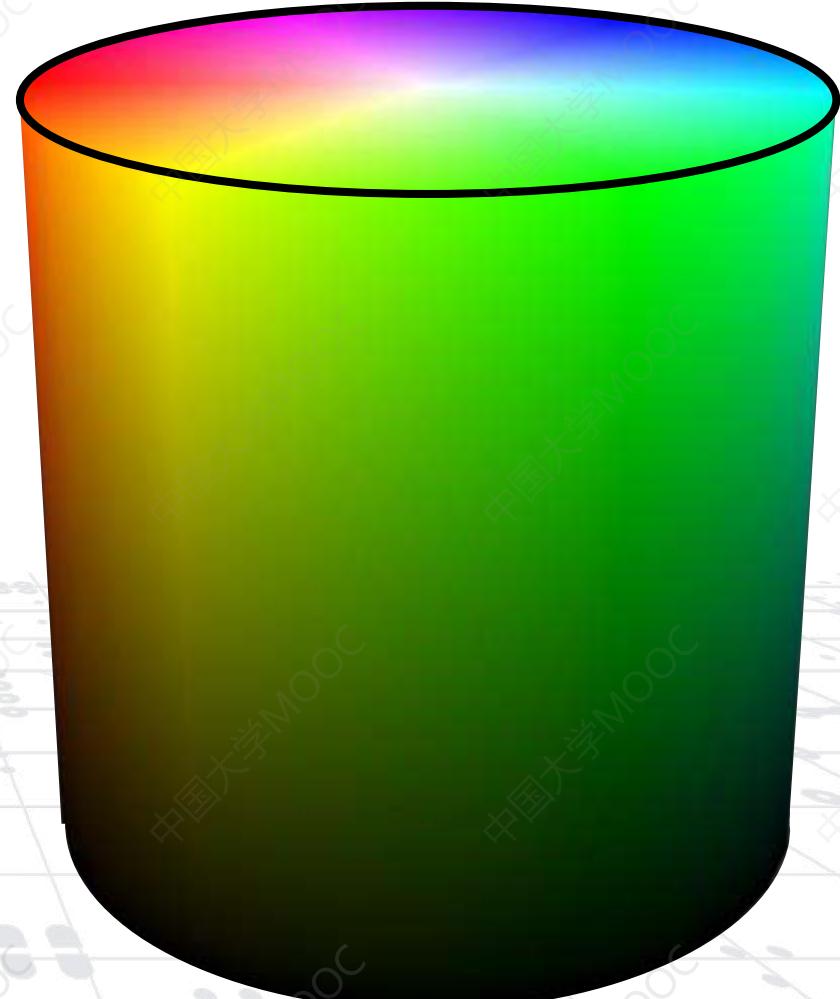


# HSV色彩模型

**H (Hue):**

**S (Saturation):**

**V (Value):**

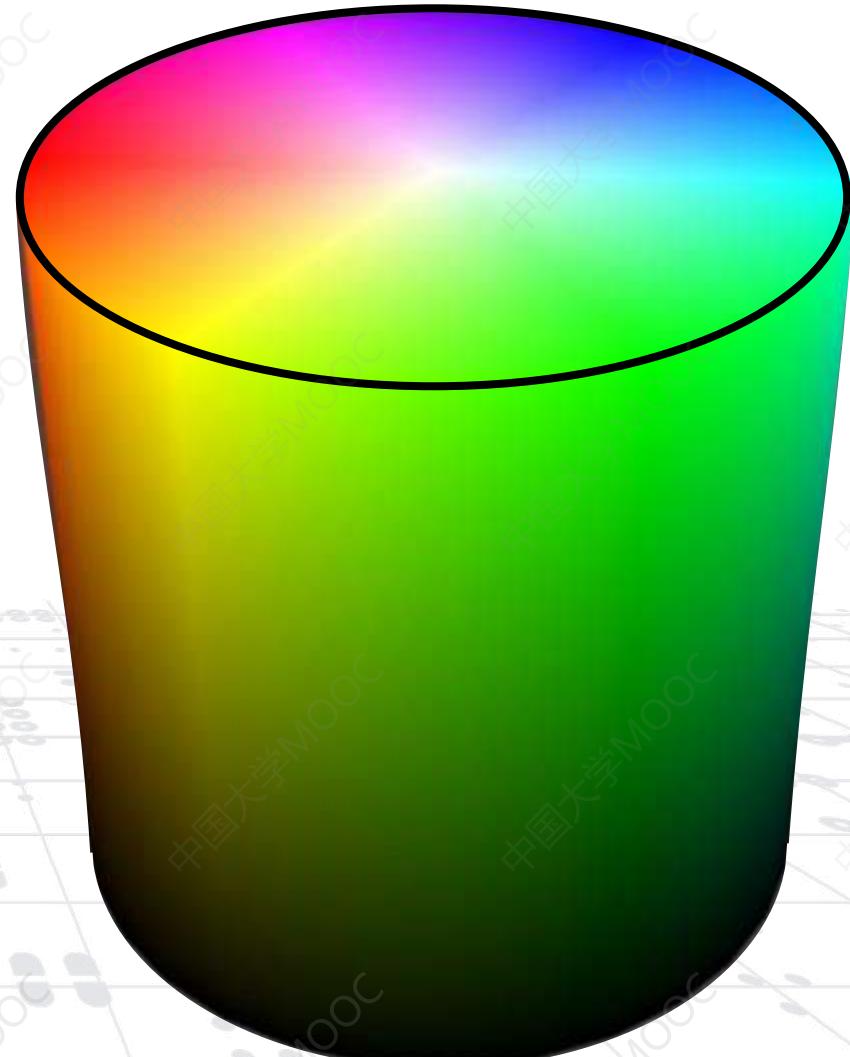


# HSV色彩模型

**H (Hue):**

**S (Saturation):**

**V (Value):**

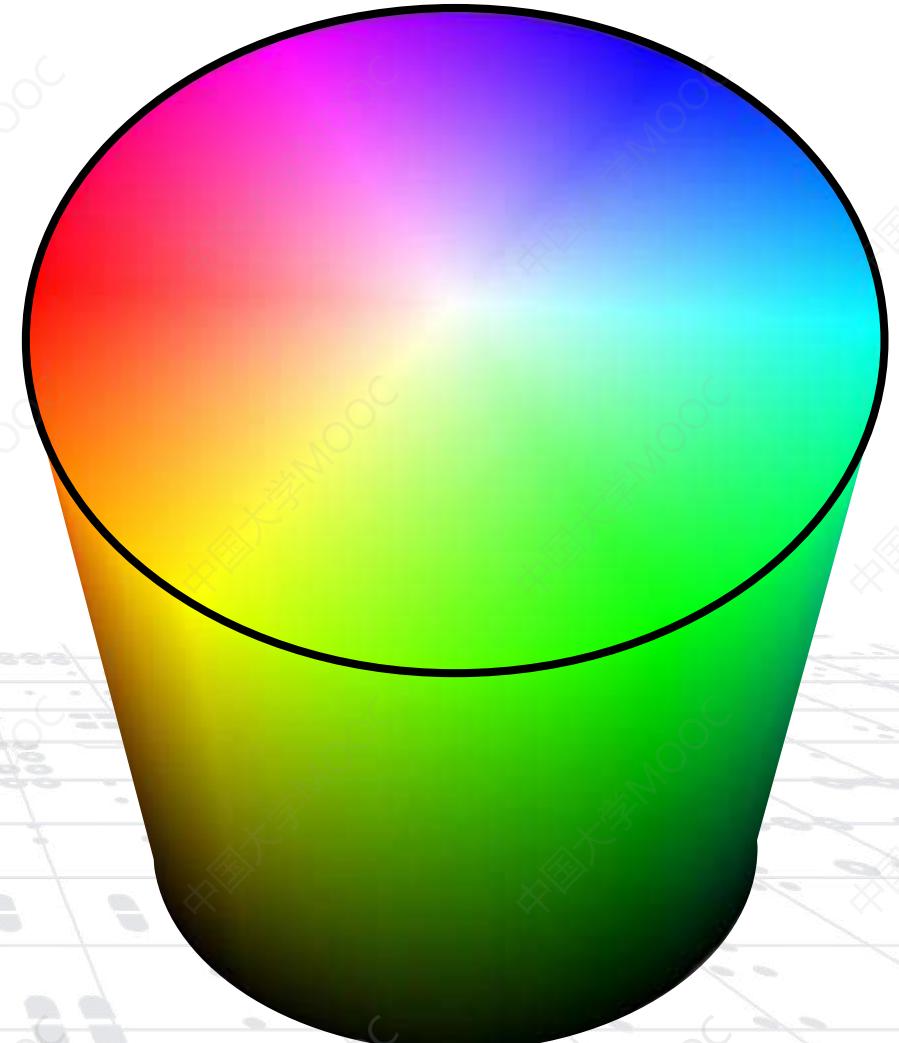


# HSV色彩模型

**H (Hue):**

**S (Saturation):**

**V (Value):**

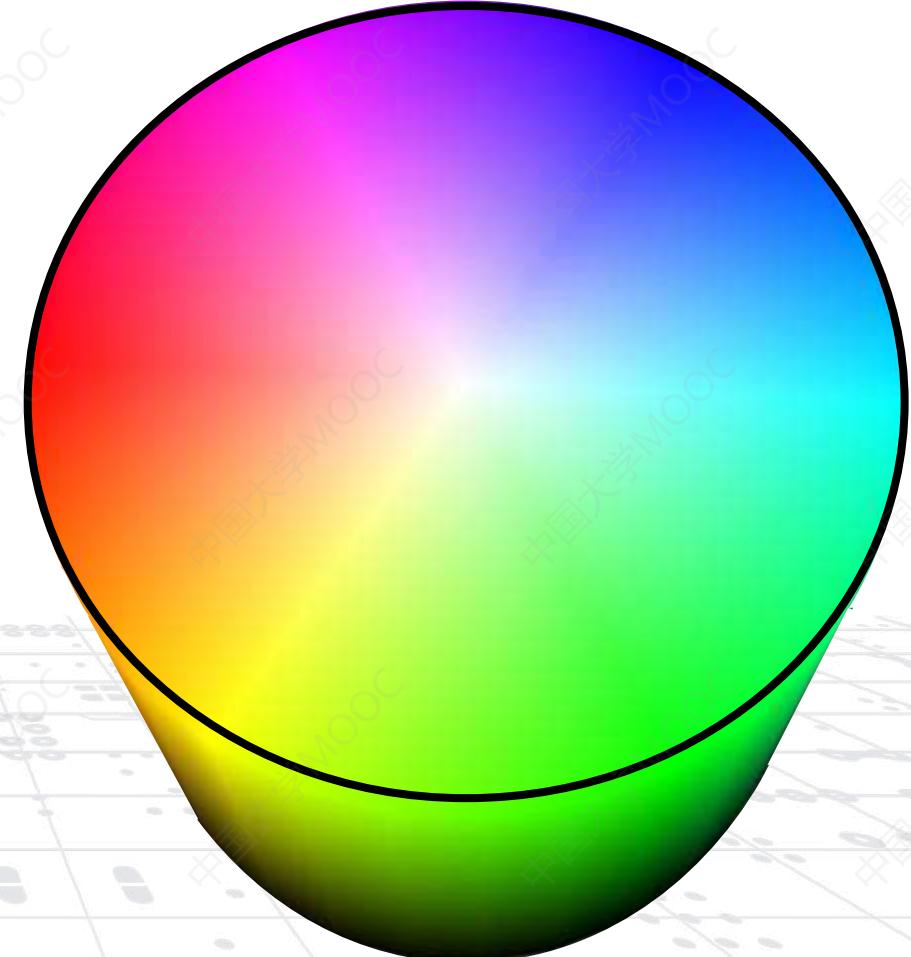


# HSV色彩模型

**H (Hue):**

**S (Saturation):**

**V (Value):**

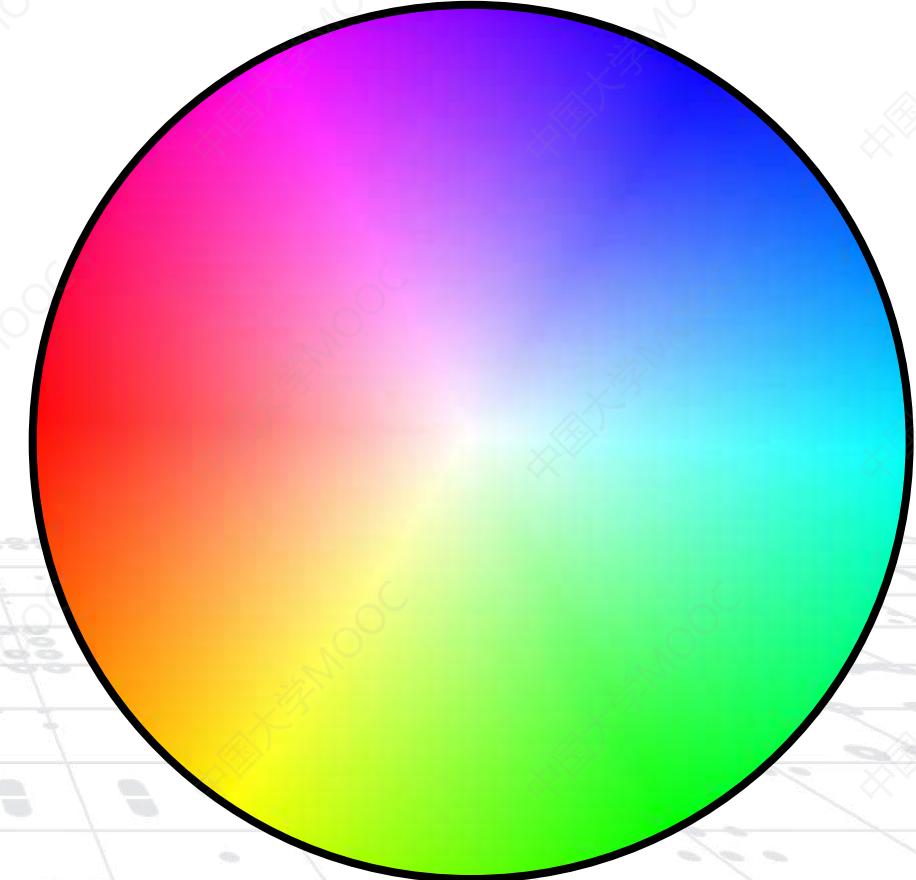


# HSV色彩模型

**H (Hue):**

**S (Saturation):**

**V (Value):**



# HSV色彩模型

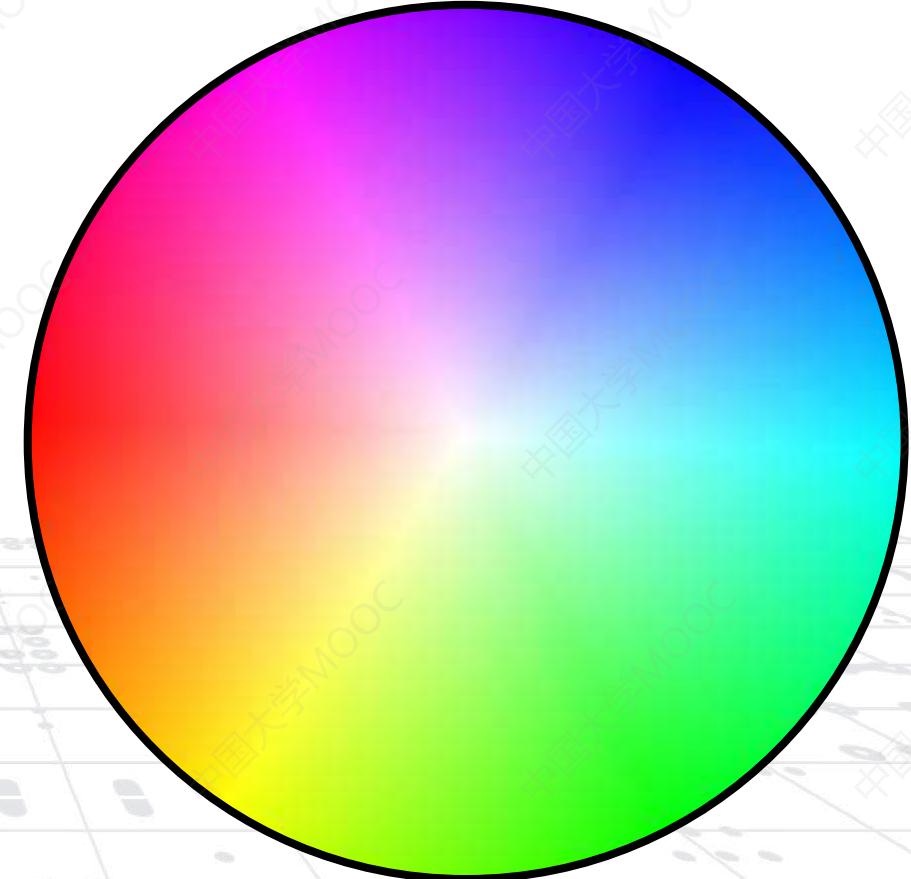
**H (Hue):**



**S (Saturation):**



**V (Value):**

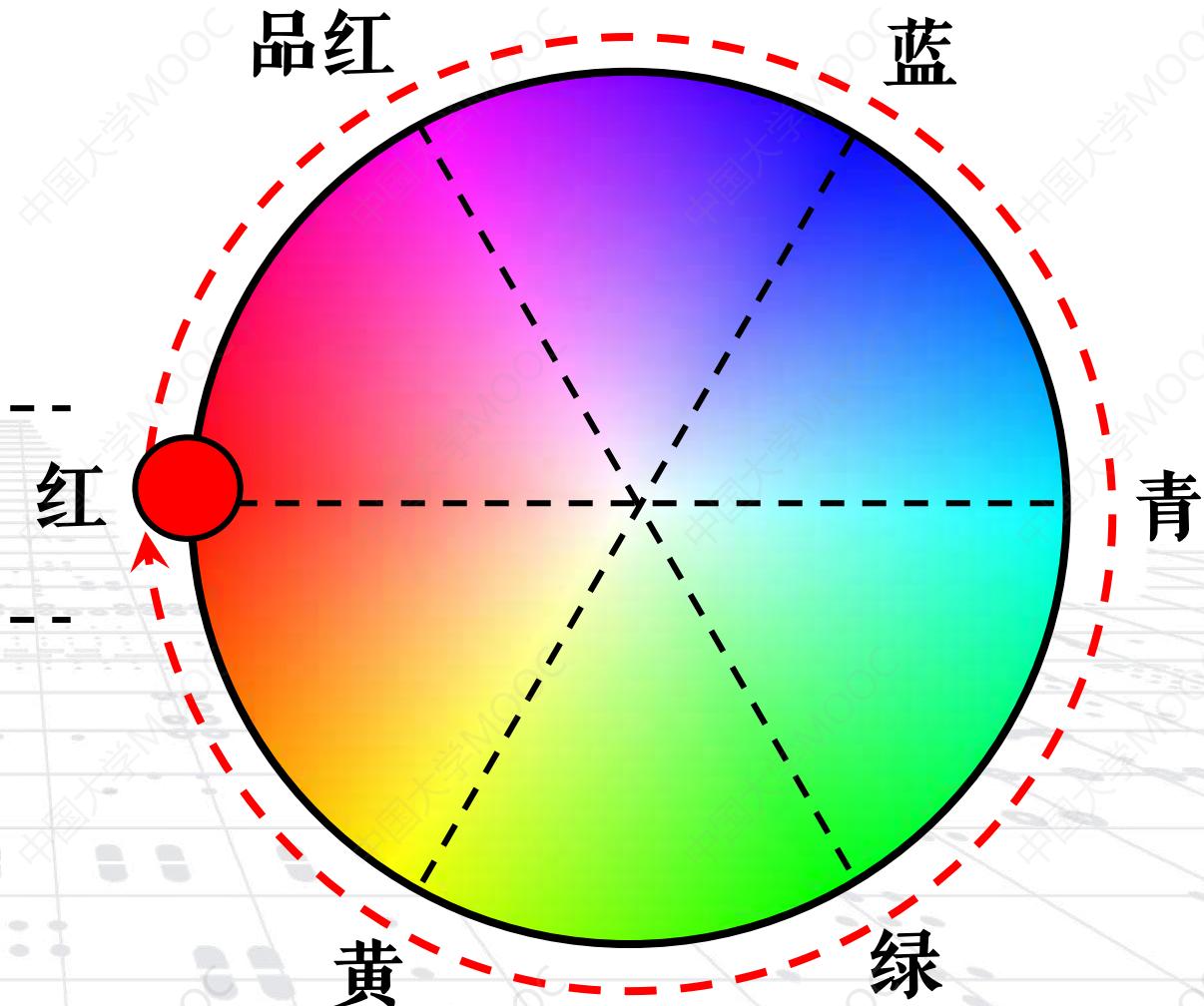


# HSV色彩模型

H (Hue): 色调,  $0 \sim 360^\circ$

S (Saturation):

V (Value):

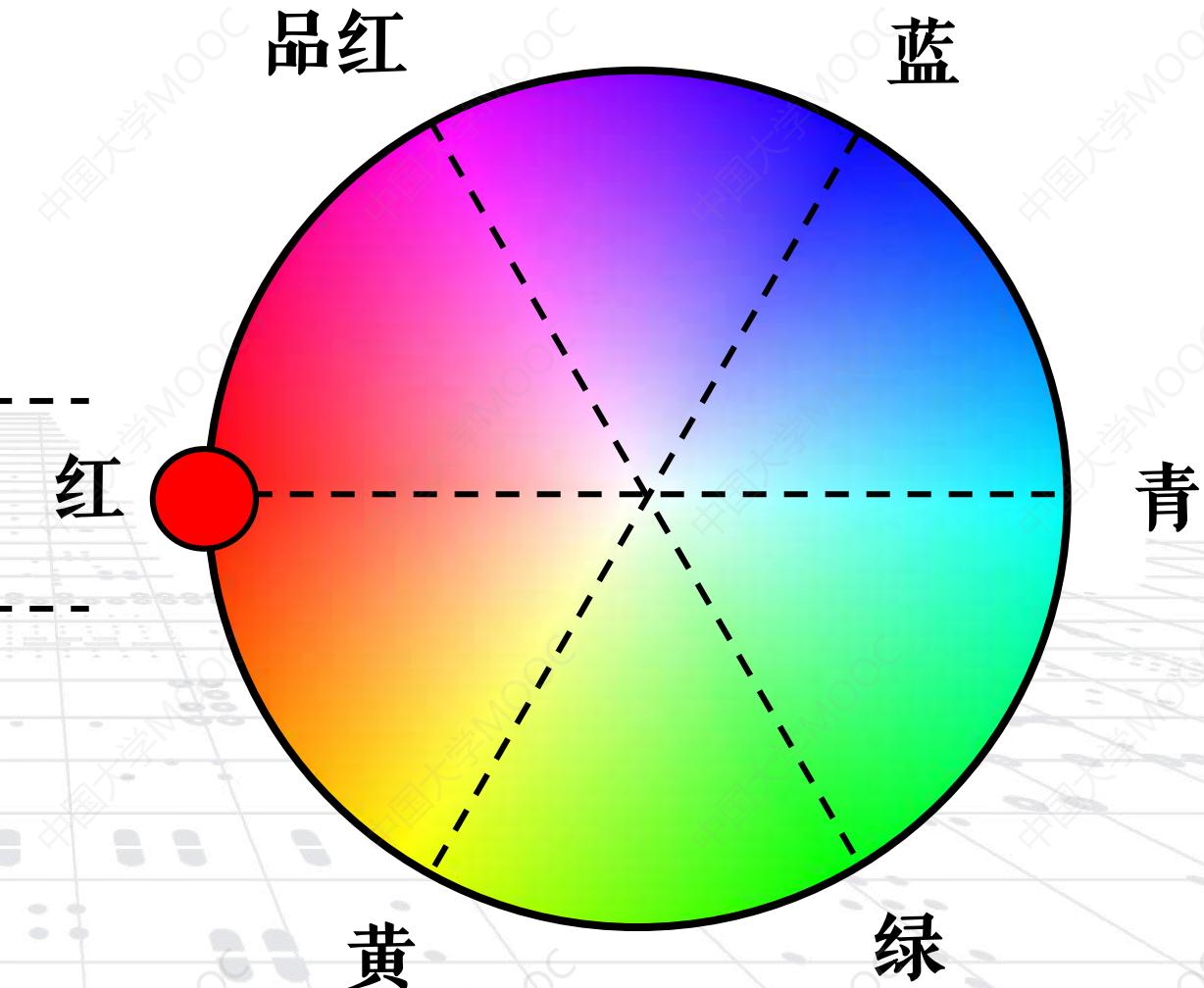


# HSV色彩模型

H (Hue): 色调,  $0 \sim 360^\circ$

S (Saturation): 饱和度,  $0 \sim 1$

V (Value):

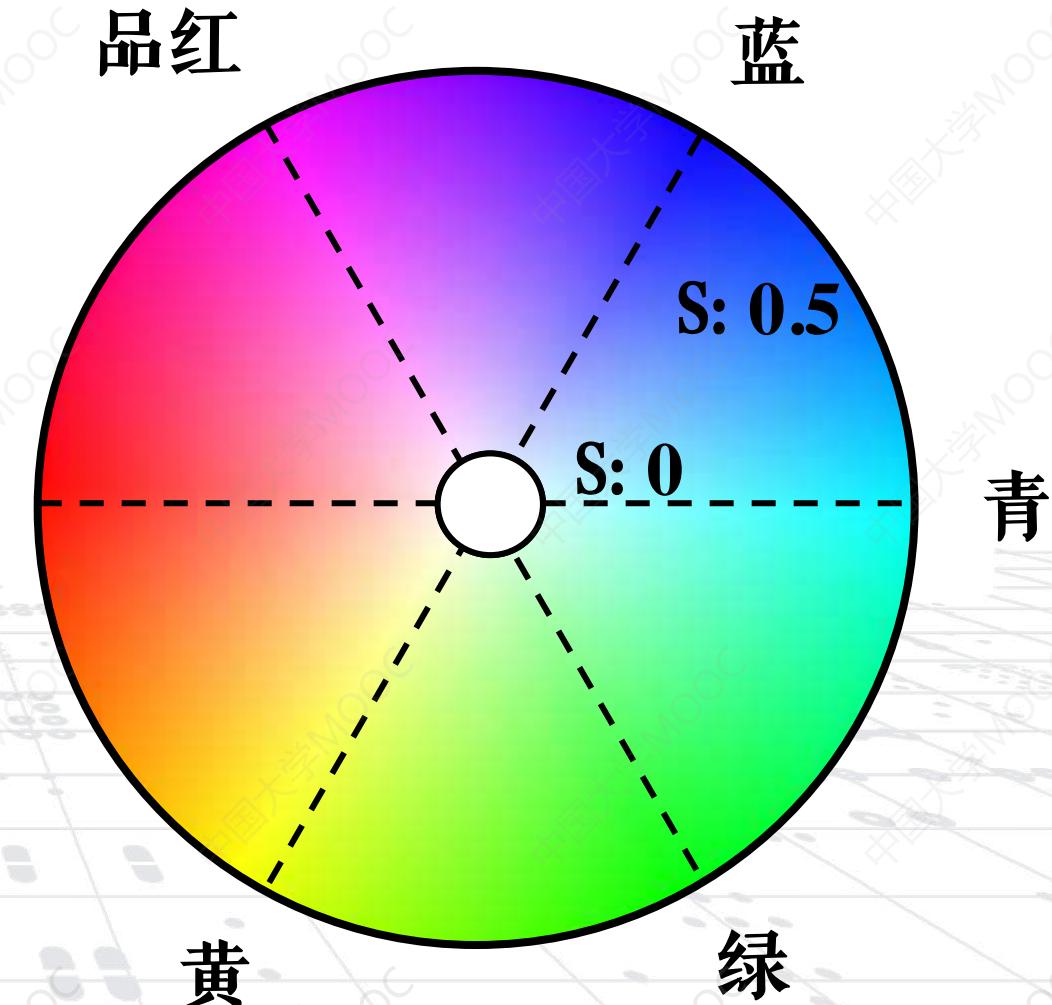


# HSV色彩模型

H (Hue): 色调,  $0 \sim 360^\circ$

S (Saturation): 饱和度,  $0 \sim 1$

V (Value):

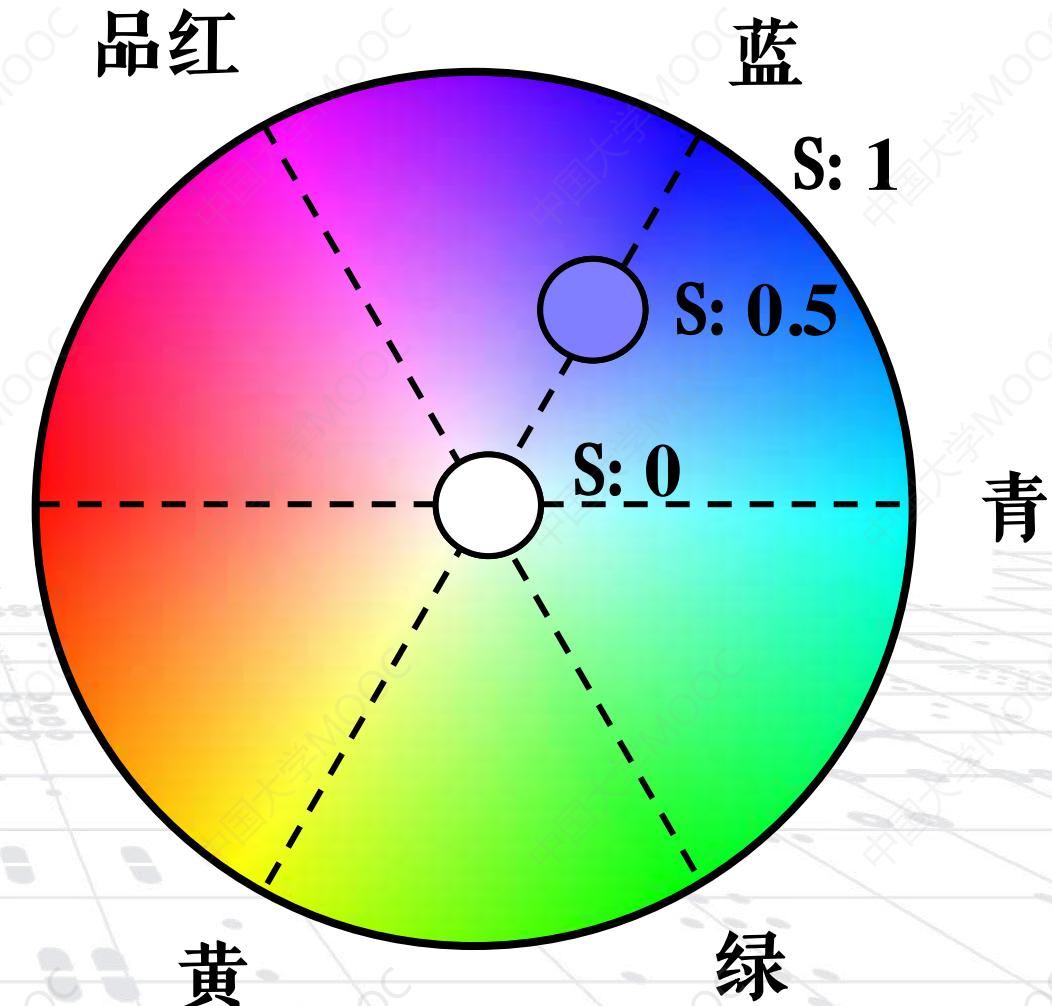


# HSV色彩模型

H (Hue): 色调,  $0 \sim 360^\circ$

S (Saturation): 饱和度,  $0 \sim 1$

V (Value):

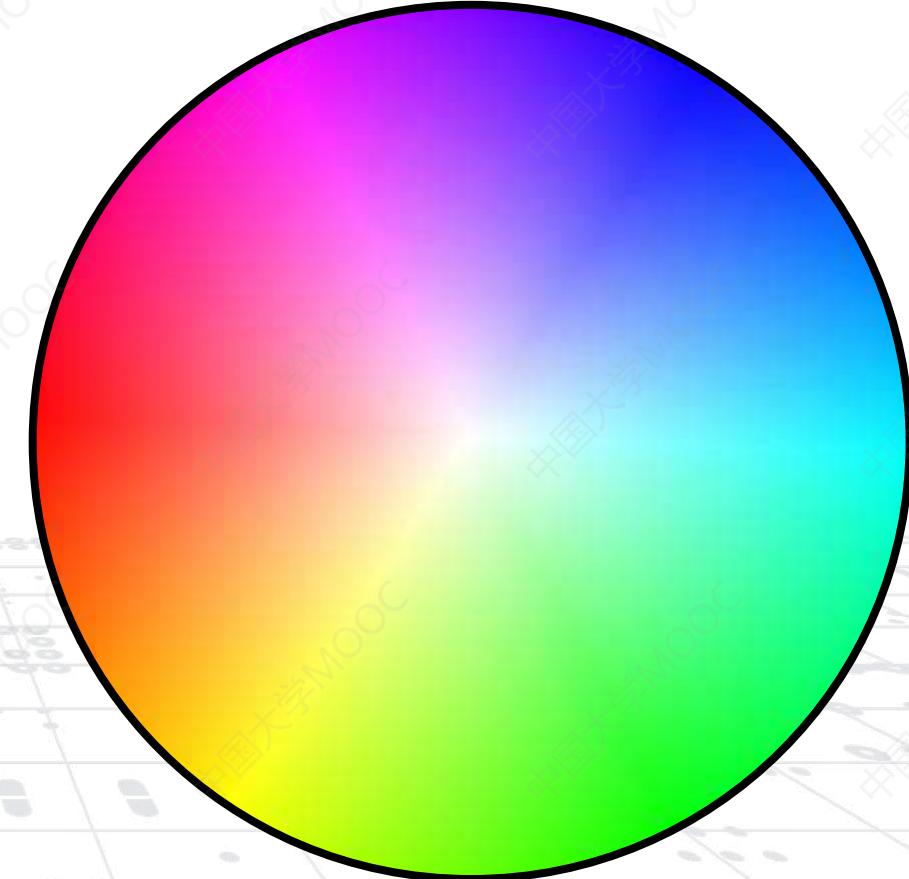


# HSV色彩模型

H (Hue): 色调,  $0 \sim 360^\circ$

S (Saturation): 饱和度,  $0 \sim 1$

V (Value):

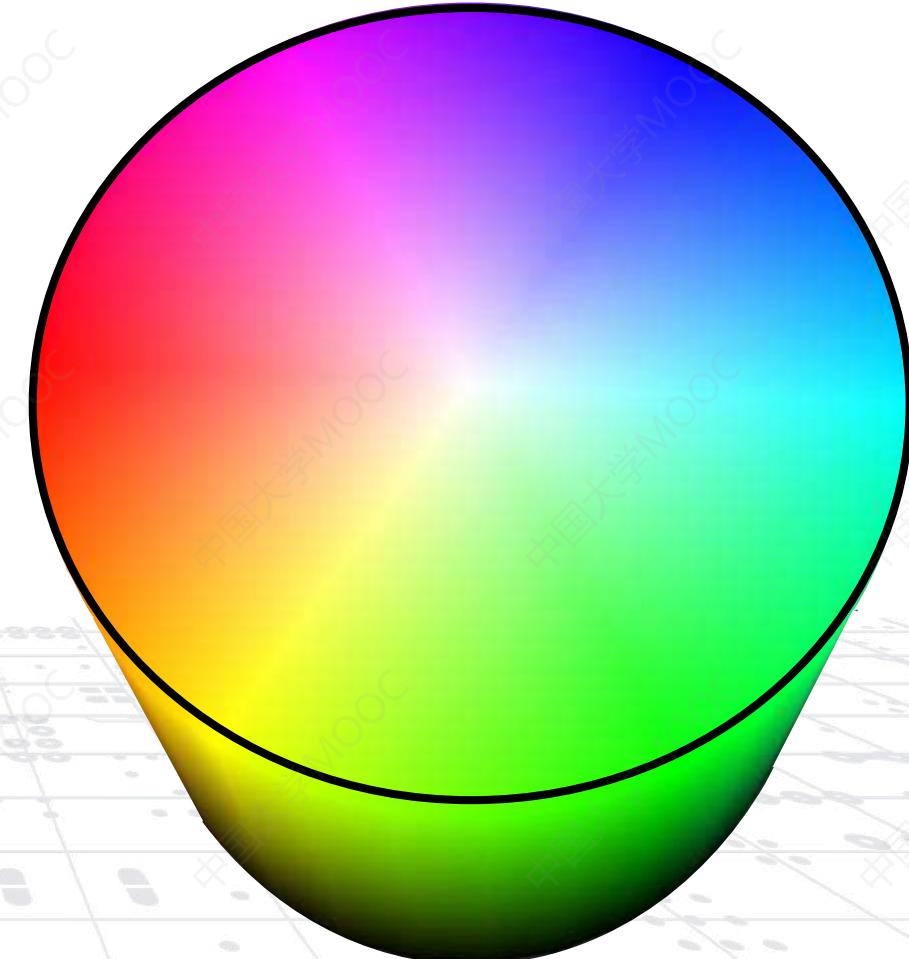


# HSV色彩模型

H (Hue): 色调,  $0 \sim 360^\circ$

S (Saturation): 饱和度,  $0 \sim 1$

V (Value):

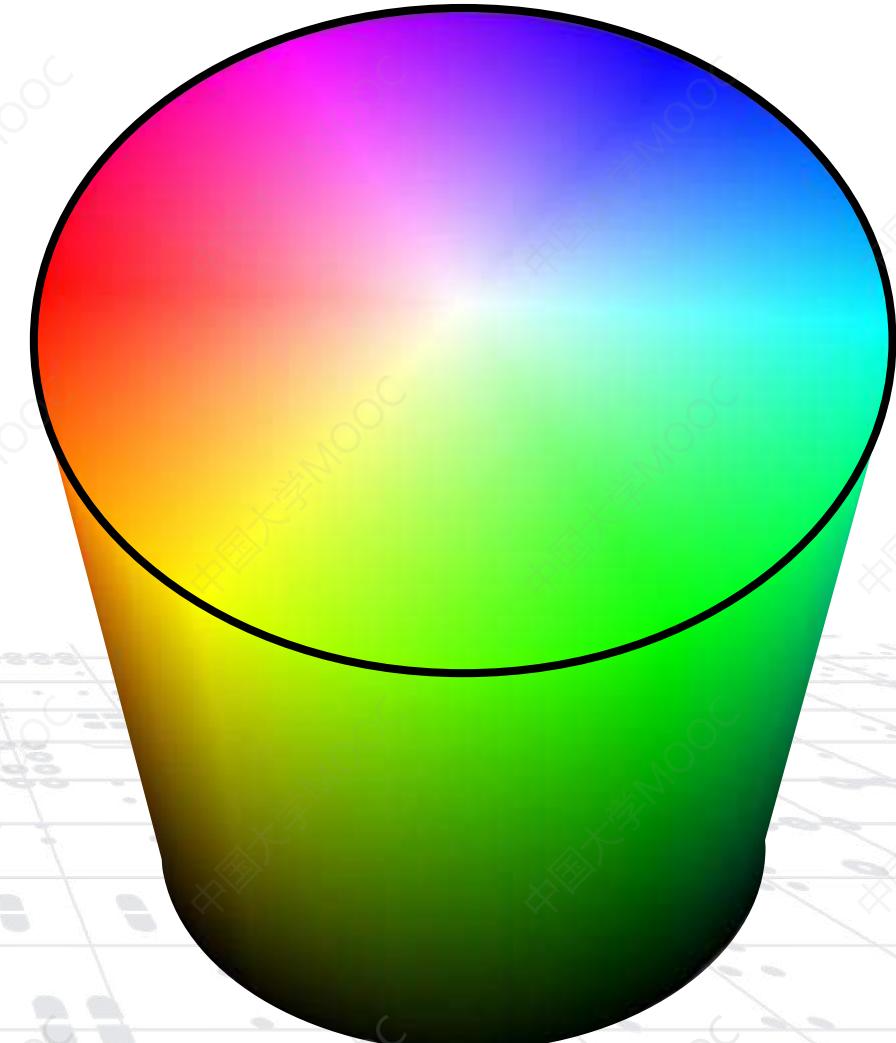


# HSV色彩模型

H (Hue): 色调,  $0 \sim 360^\circ$

S (Saturation): 饱和度,  $0 \sim 1$

V (Value):

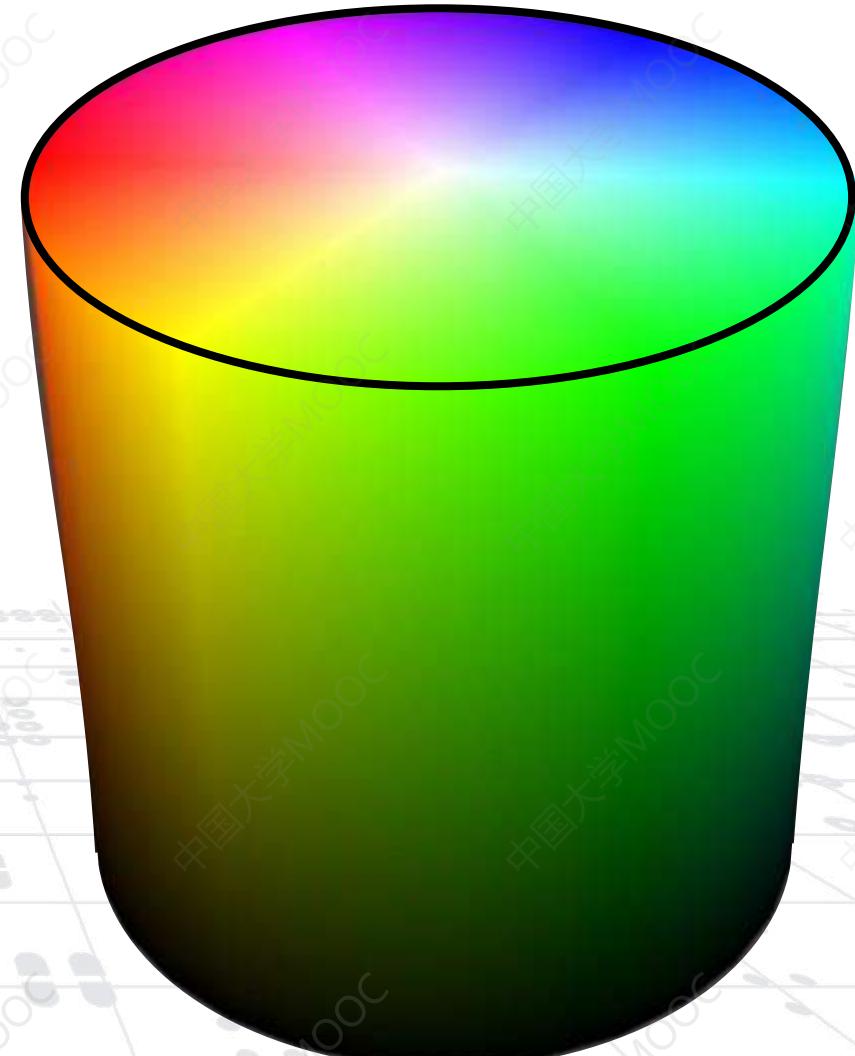


# HSV色彩模型

H (Hue): 色调,  $0 \sim 360^\circ$

S (Saturation): 饱和度,  $0 \sim 1$

V (Value):

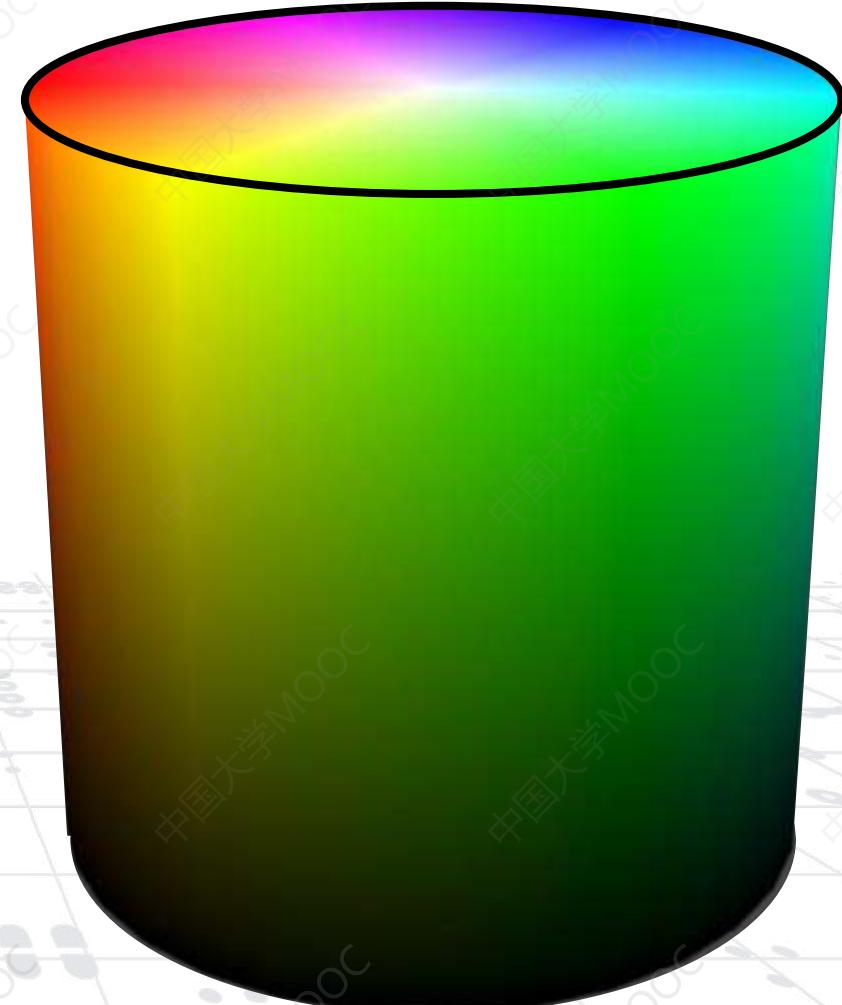


# HSV色彩模型

H (Hue): 色调,  $0 \sim 360^\circ$

S (Saturation): 饱和度,  $0 \sim 1$

V (Value):

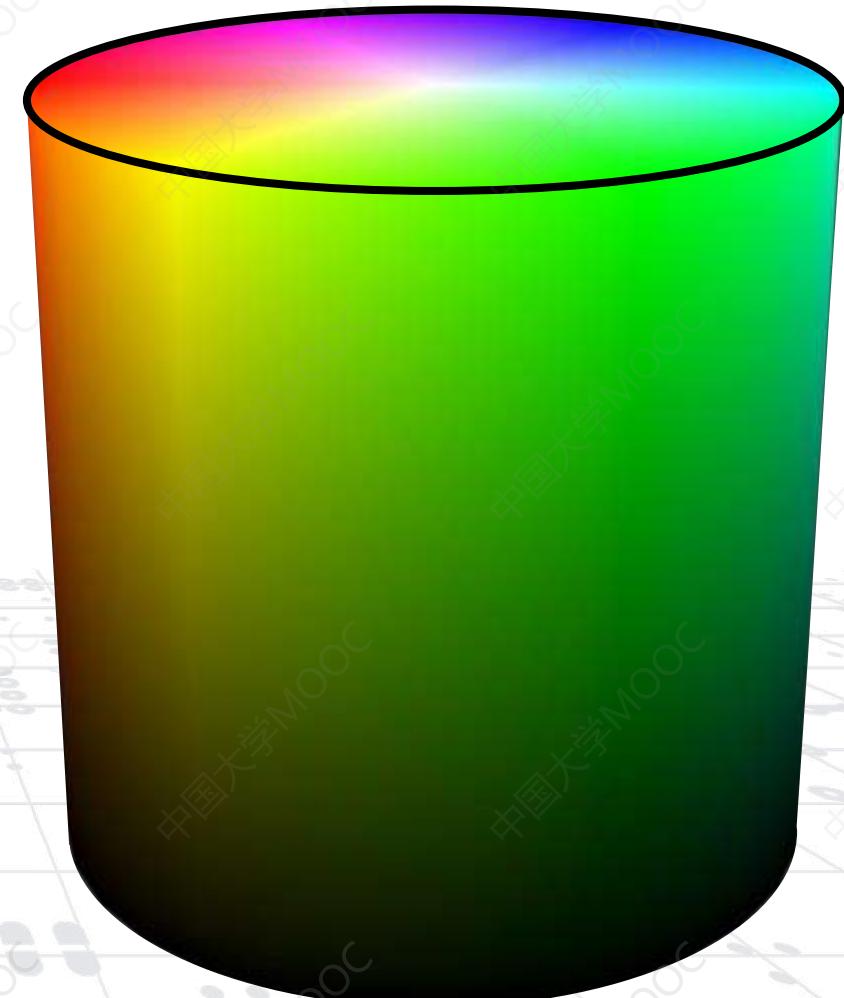


# HSV色彩模型

H (Hue): 色调,  $0 \sim 360^\circ$

S (Saturation): 饱和度,  $0 \sim 1$

V (Value):

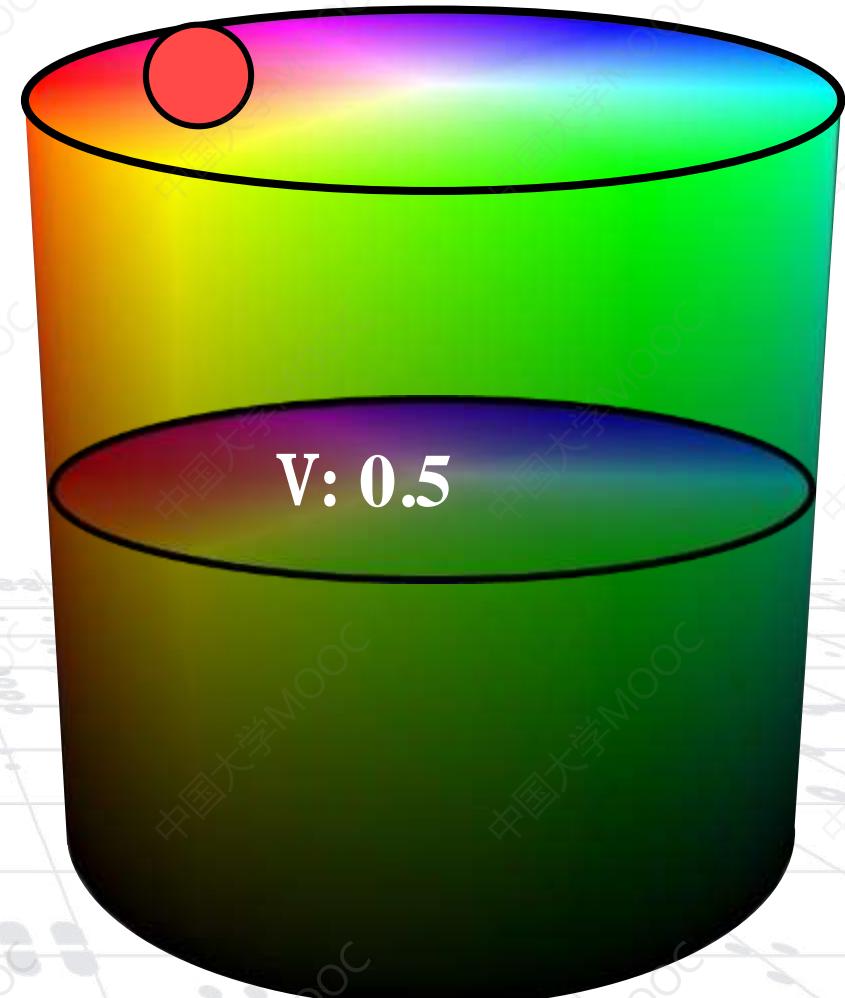


# HSV色彩模型

H (Hue): 色调,  $0 \sim 360^\circ$

S (Saturation): 饱和度, 0 ~ 1

V (Value):

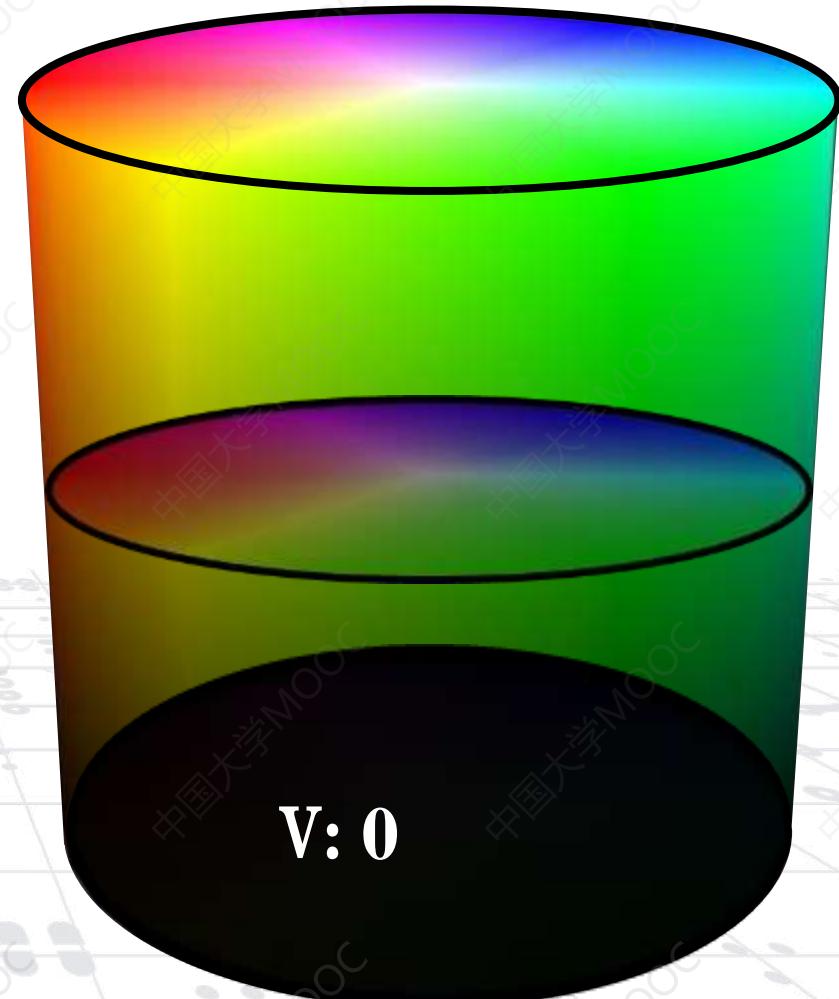


# HSV色彩模型

H (Hue): 色调,  $0 \sim 360^\circ$

S (Saturation): 饱和度,  $0 \sim 1$

V (Value): 亮度,  $0 \sim 1$

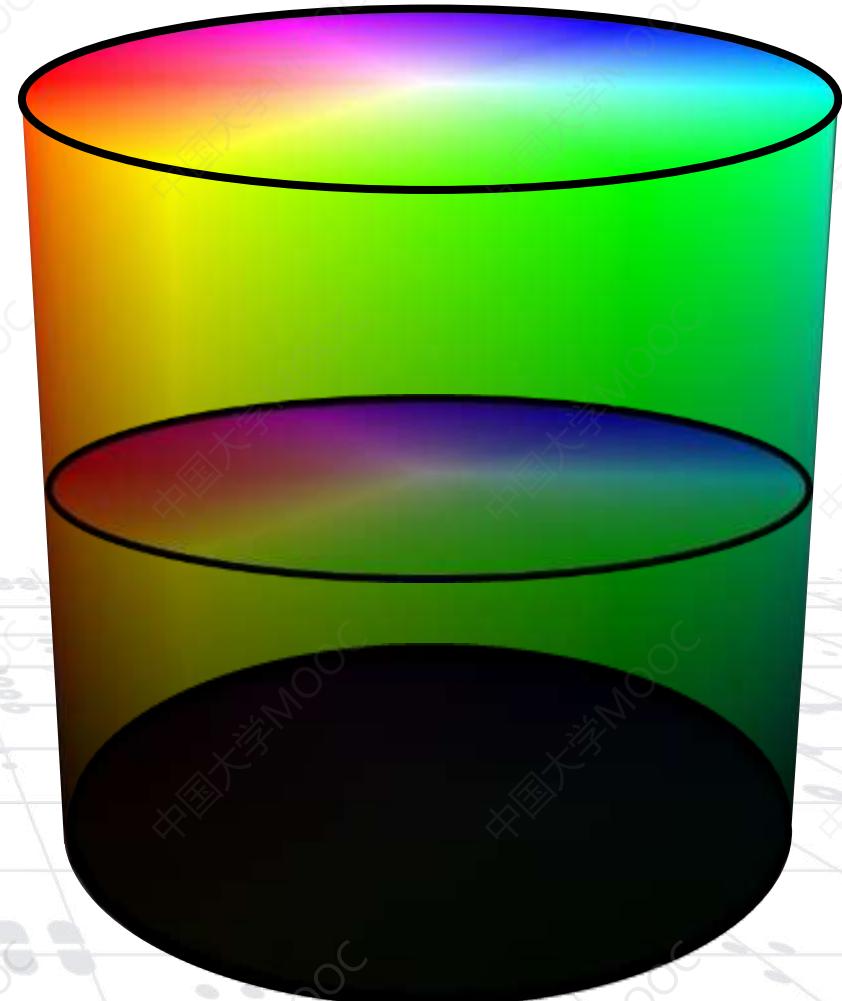


# HSV色彩模型

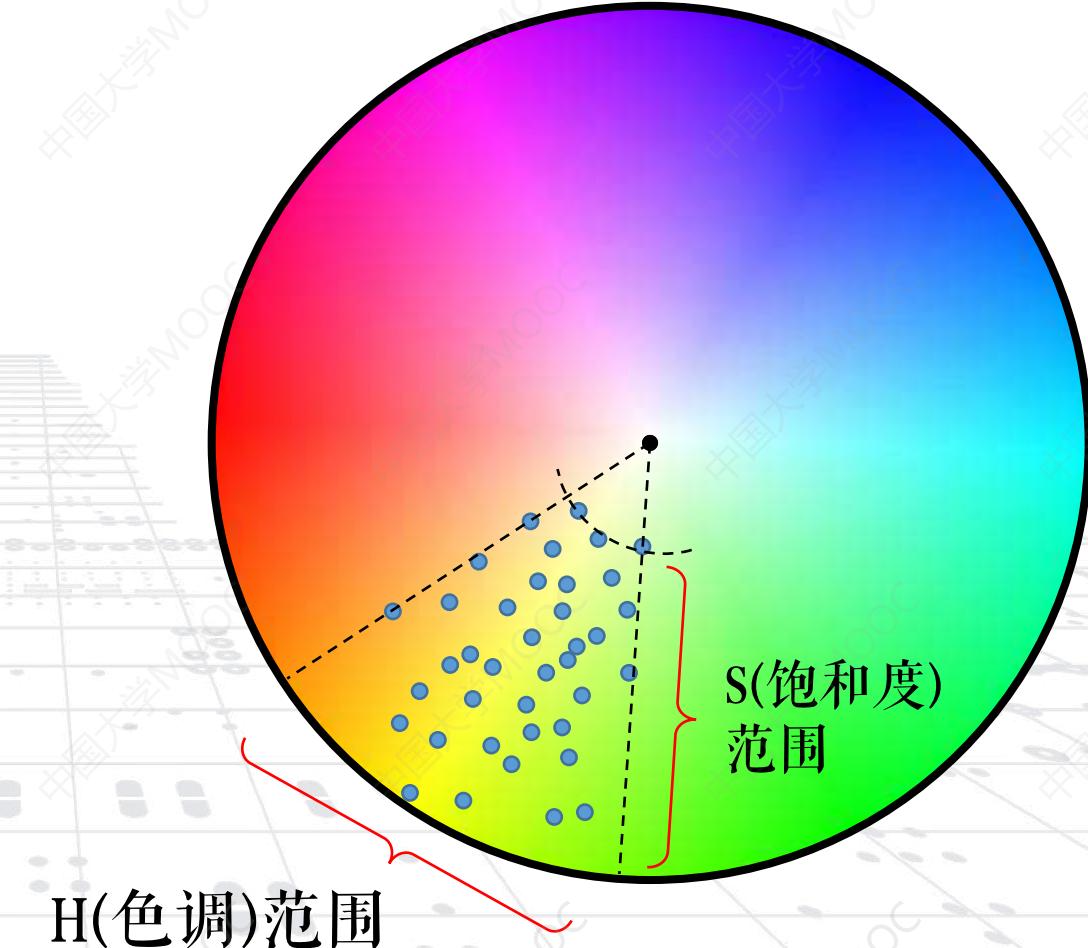
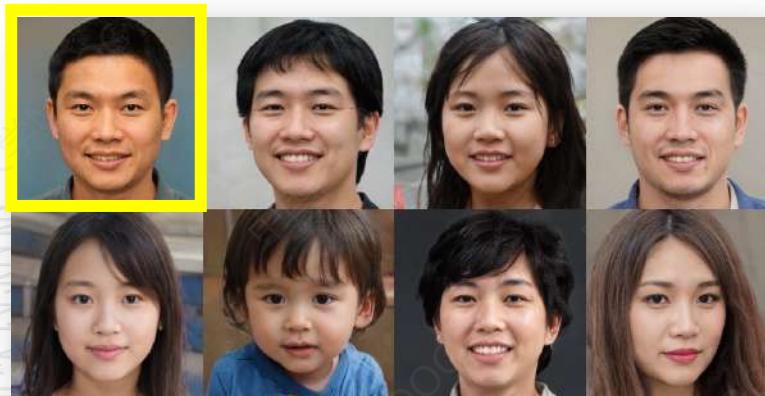
H (Hue): 色调,  $0 \sim 360^\circ$

S (Saturation): 饱和度,  $0 \sim 1$

V (Value): 亮度 ,  $0 \sim 1$



# 开源数据集





# 设定范围

H



[274, 327]

S



[0.16, 1]

V



[0.21, 1]

[1] Vezhnevets V , Sazonov V , Andreeva A . A  
Survey on Pixel-Based Skin Color Detection  
Techniques[J]. Graphicon, 2003:85--92.

# 肤色检测算法

- ① 读取RGB值
- ② 转换HSV值
- ③ 值域判断
- ④ 标记结果



## 设定范围

H



## 设定范围

H



[274, 327]



[0.16, 1]

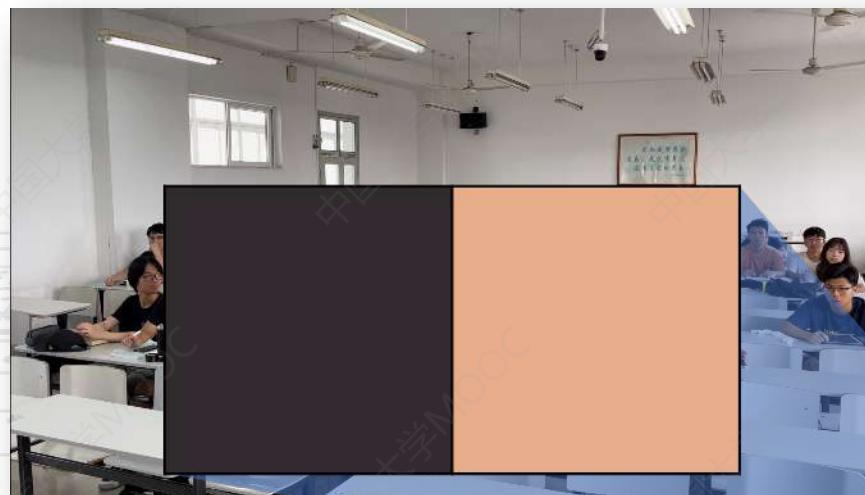


[0.21, 1]

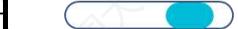
0.21, 1

# 肤色检测算法

- ① 读取RGB值
- ② 转换HSV值
- ③ 值域判断
- ④ 标记结果



设定范围

H  [274, 327]

S  [0.16, 1]

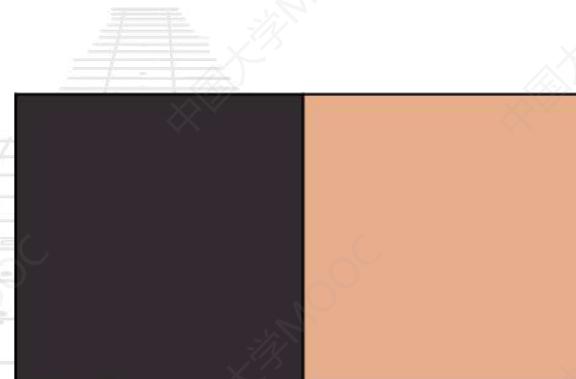
V  [0.21, 1]



# 肤色检测算法

- ① 读取RGB值
- ② 转换HSV值
- ③ 值域判断
- ④ 标记结果

49	232	H	[274, 327]
42	174	S	[0.16, 1]
46	140	V	[0.21, 1]



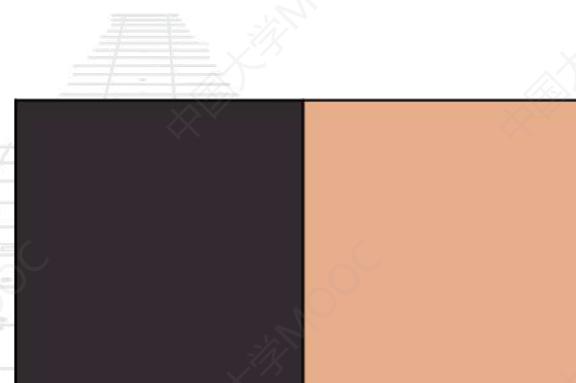
# 肤色检测算法

- ① 读取RGB值
- ② 转换HSV值
- ③ 值域判断
- ④ 标记结果

152      282  
0.18      0.4  
0.19      0.9

设定范围

H	<input type="range"/>	[274, 327]
S	<input type="range"/>	[0.16, 1]
V	<input type="range"/>	[0.21, 1]



# 肤色检测算法



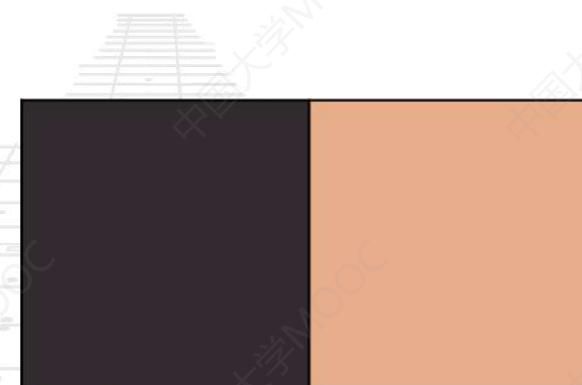
✗	152	✓	282
✓	0.18	✓	0.4
✗	0.19	✓	0.9

设定范围

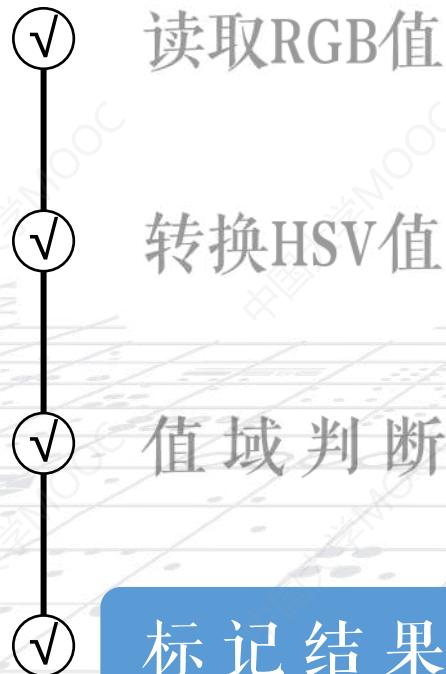
H [274, 327]

S [0.16, 1]

V [0.21, 1]



# 肤色检测算法



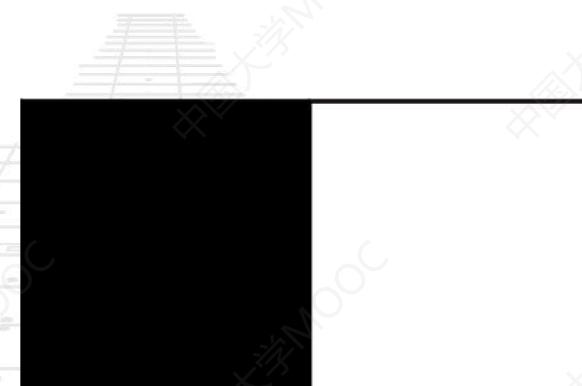
✗	152	✓	282
✓	0.18	✓	0.4
✗	0.19	✓	0.9

设定范围

H [274, 327]

S [0.16, 1]

V [0.21, 1]



# 肤色检测demo

# 本章概要

硬件介绍

1

基本概念

2

图像种类

3

色彩模型

4

直方图

5

# 本章概要

硬件介绍

1

基本概念

2

图像种类

3

色彩模型

4

直方图

5

## 简化目标图像

150	0	0	255
150	100	100	150
255	100	100	150
255	100	100	255
255	150	150	255
255	150	150	255

0 50 100 150 200 255

## 简化目标图像

	0	0	5
150	100	100	150
255	100	100	150
255	100	100	255
255	150	150	255
255	150	150	255

0      50      100      150      200      255



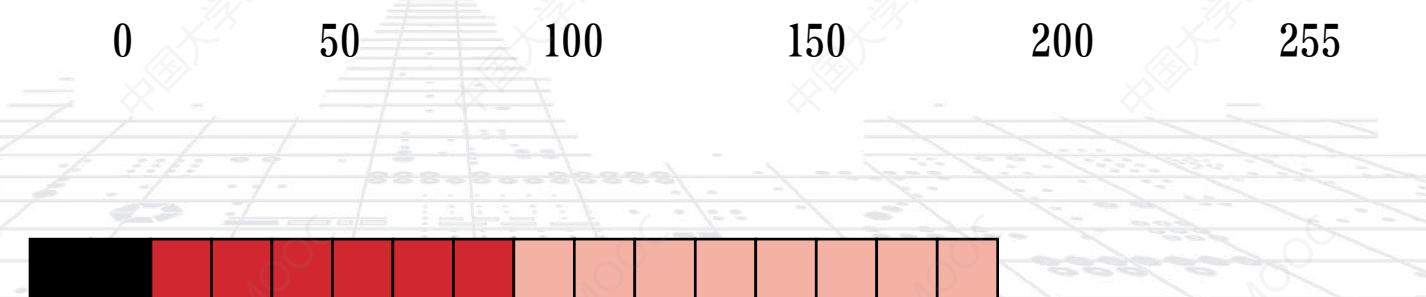
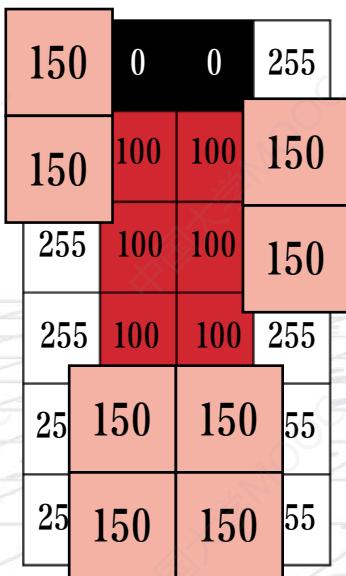
## 简化目标图像

150	0	0	255
15	100	100	50
25	100	100	50
25	100	100	255
255	150	150	255
255	150	150	255

0      50      100      150      200      255

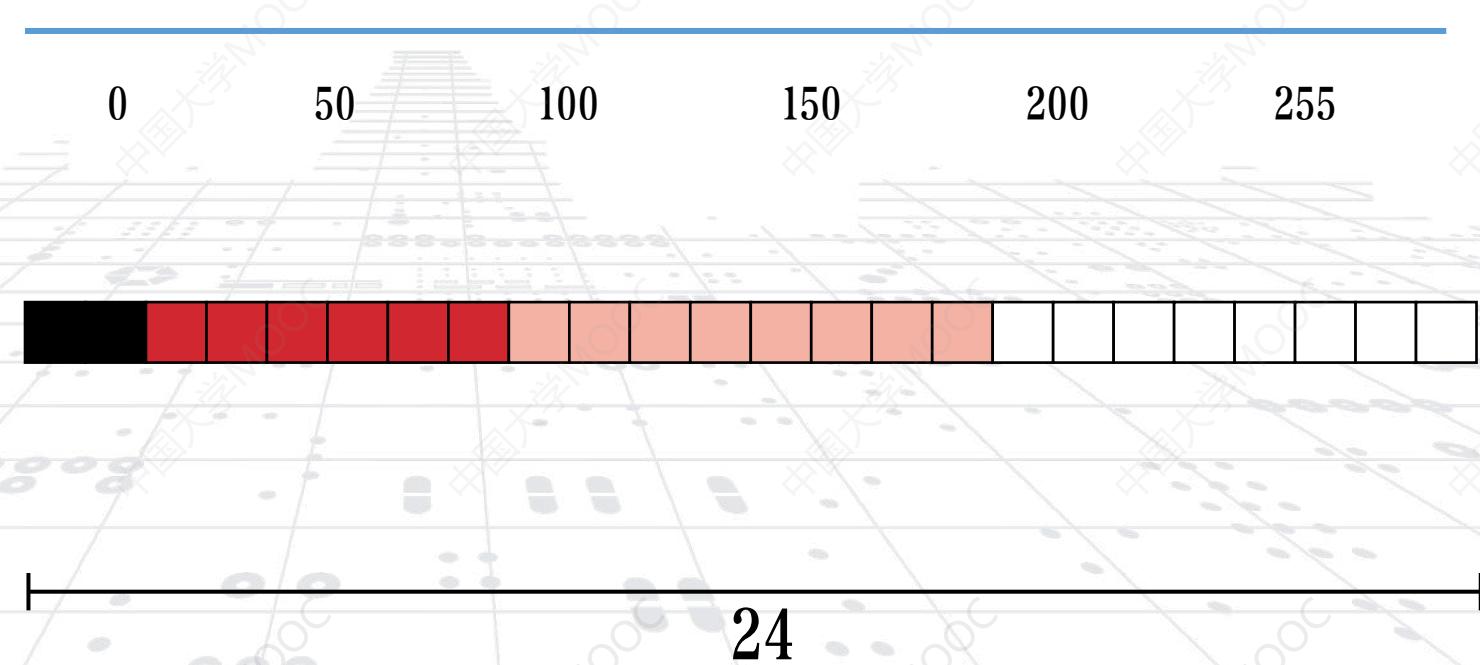


## 简化目标图像



## 简化目标图像

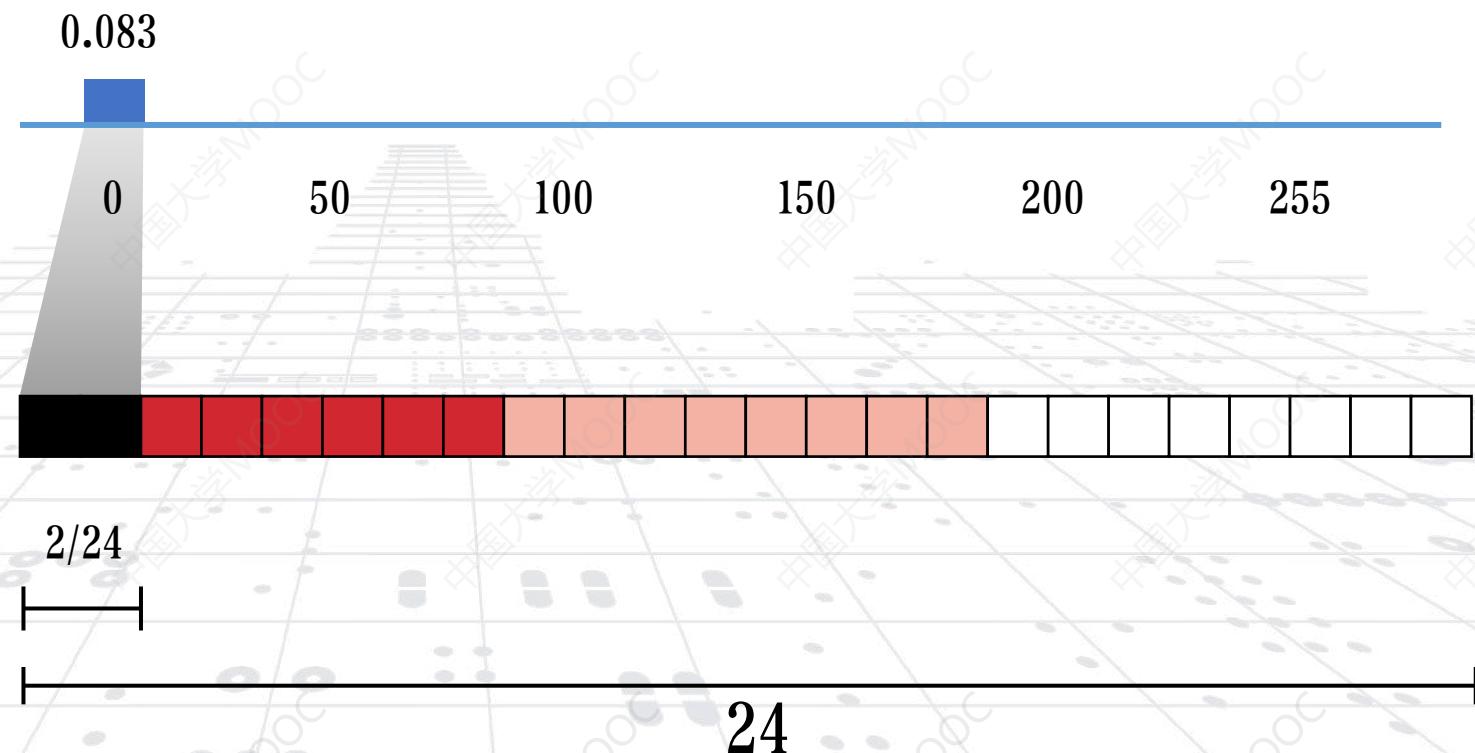
150	0	0	255
150	100	100	150
255	100	100	150
255	100	100	255
255	150	150	255
255	150	150	255



24

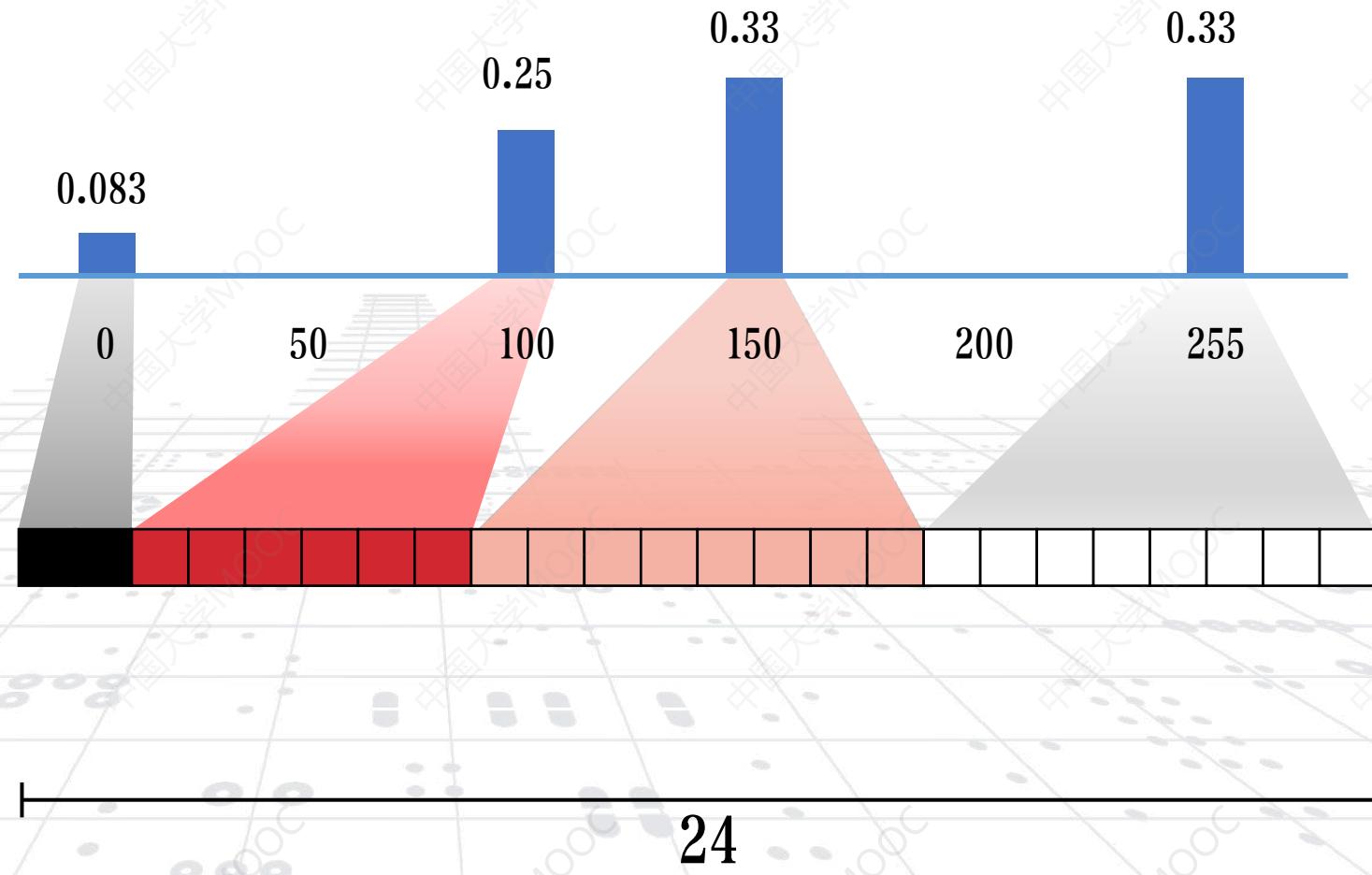
## 简化目标图像

150	0	0	255
150	100	100	150
255	100	100	150
255	100	100	255
255	150	150	255
255	150	150	255



## 简化目标图像

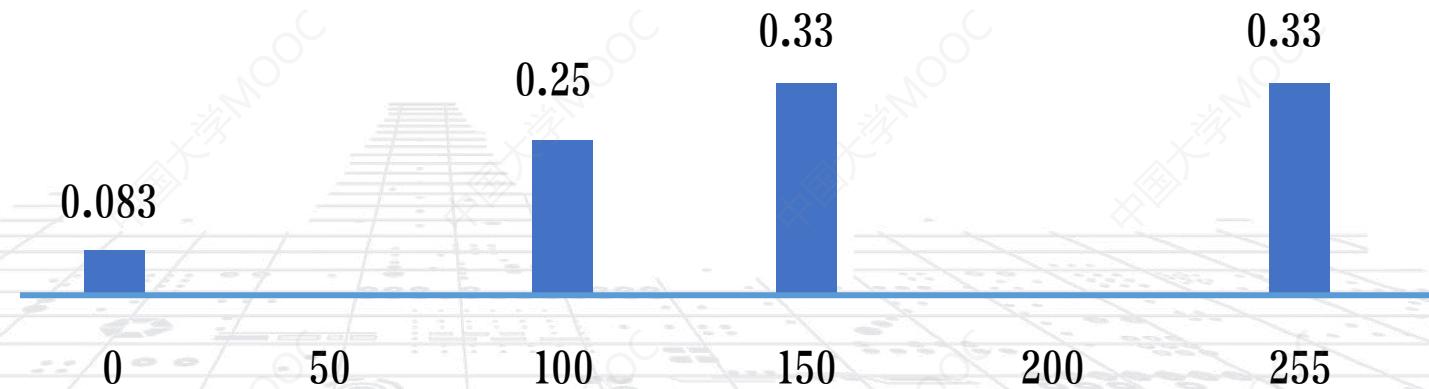
150	0	0	255
150	100	100	150
255	100	100	150
255	100	100	255
255	150	150	255
255	150	150	255



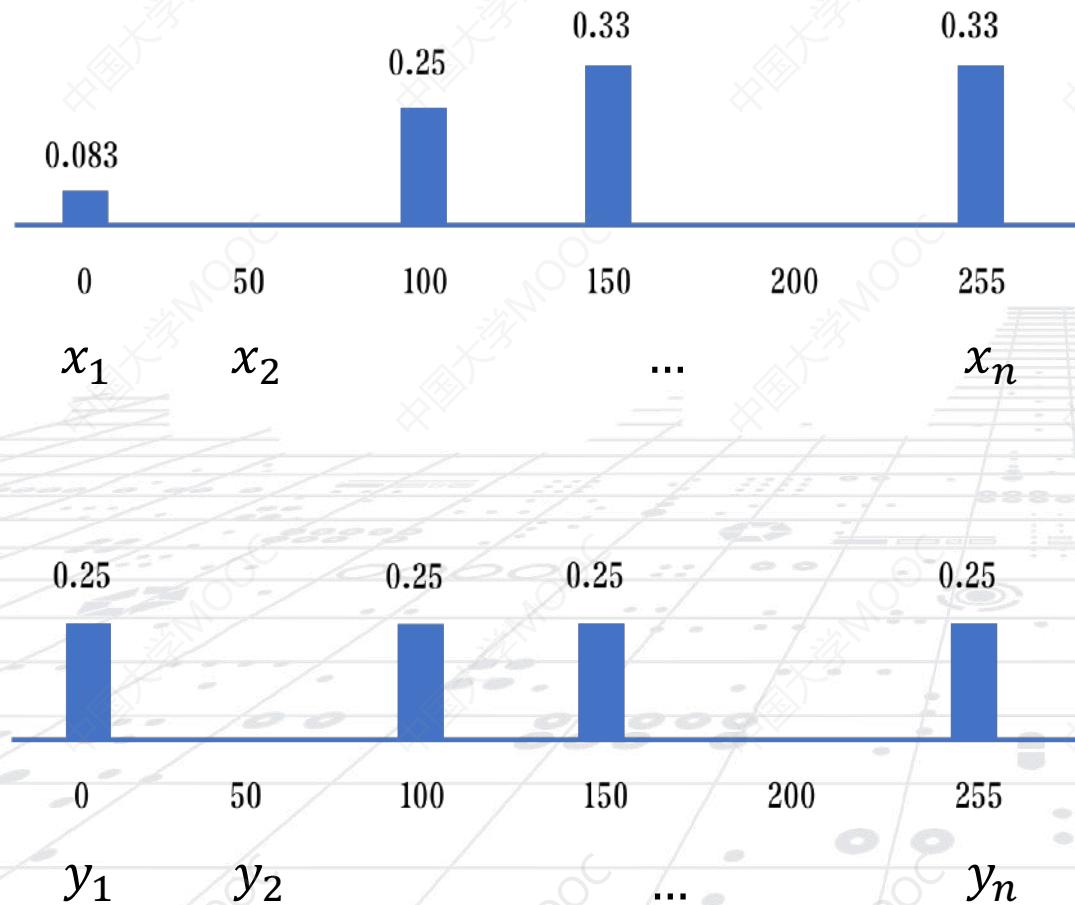
## 简化目标图像

150	0	0	255
150	100	100	150
255	100	100	150
255	100	100	255
255	150	150	255
255	150	150	255

## 灰度直方图



# 图像匹配



欧几里得距离

$$d(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$
$$= \sqrt{(x_1 - y_1)^2 + \dots + (x_n - y_n)^2}$$

# 画质判断

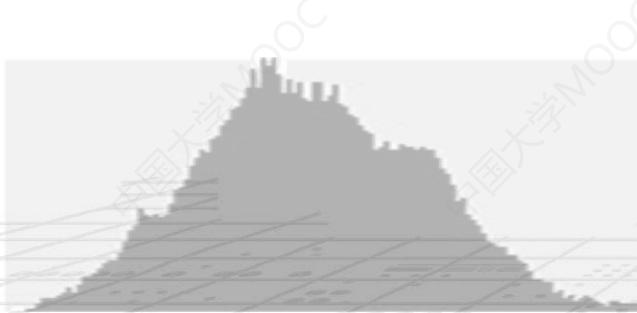
直方图：



曝光正常

# 画质判断

直方图：



曲线偏左，  
图像较暗，  
曝光不足

# 画质判断

直方图：



曲线严重偏左，  
图像严重偏暗，  
严重曝光不足。

# 画质判断

直方图：



曲线严重偏右  
图像严重偏亮  
严重曝光过度

# 画质判断

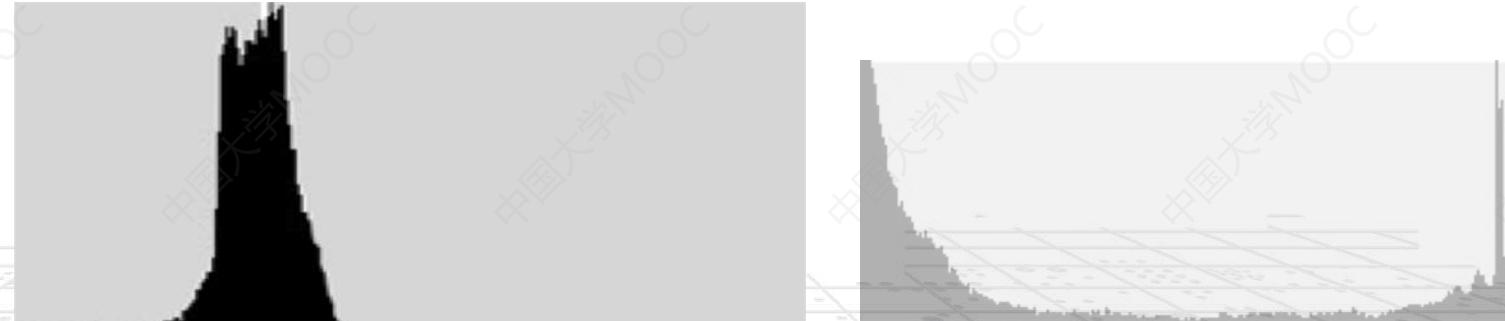
直方图：



曲线偏右  
图像较亮  
曝光过度

# 画质判断

直方图：



动态范围过小  
图像细节不足

# 画质判断

直方图：



曲线向两端溢出  
图像明暗反差过大

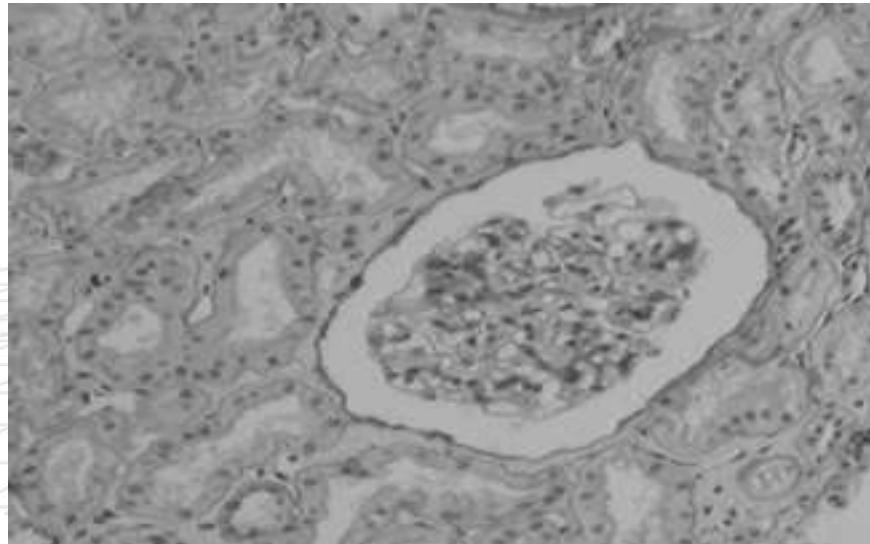
# 画质判断

直方图：



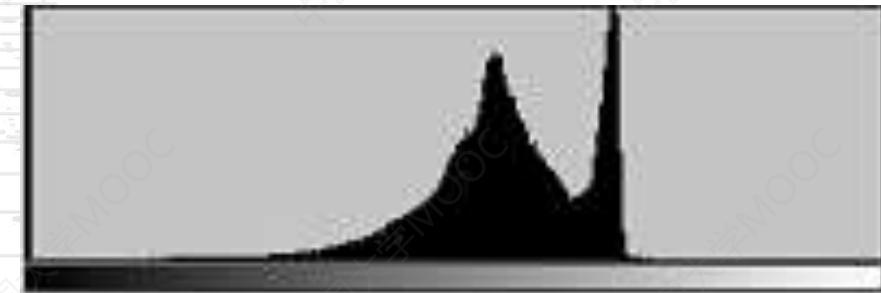
曲线向两端溢出  
图像明暗反差过大

# 二值化阈值



灰度图

分离细胞和背景，如何取阈值？



直方图

# 谢谢！

# 机器视觉技术与应用

## 3. 图像二值化

李竹

杭州电子科技大学

电子信息学院



# 本章概要

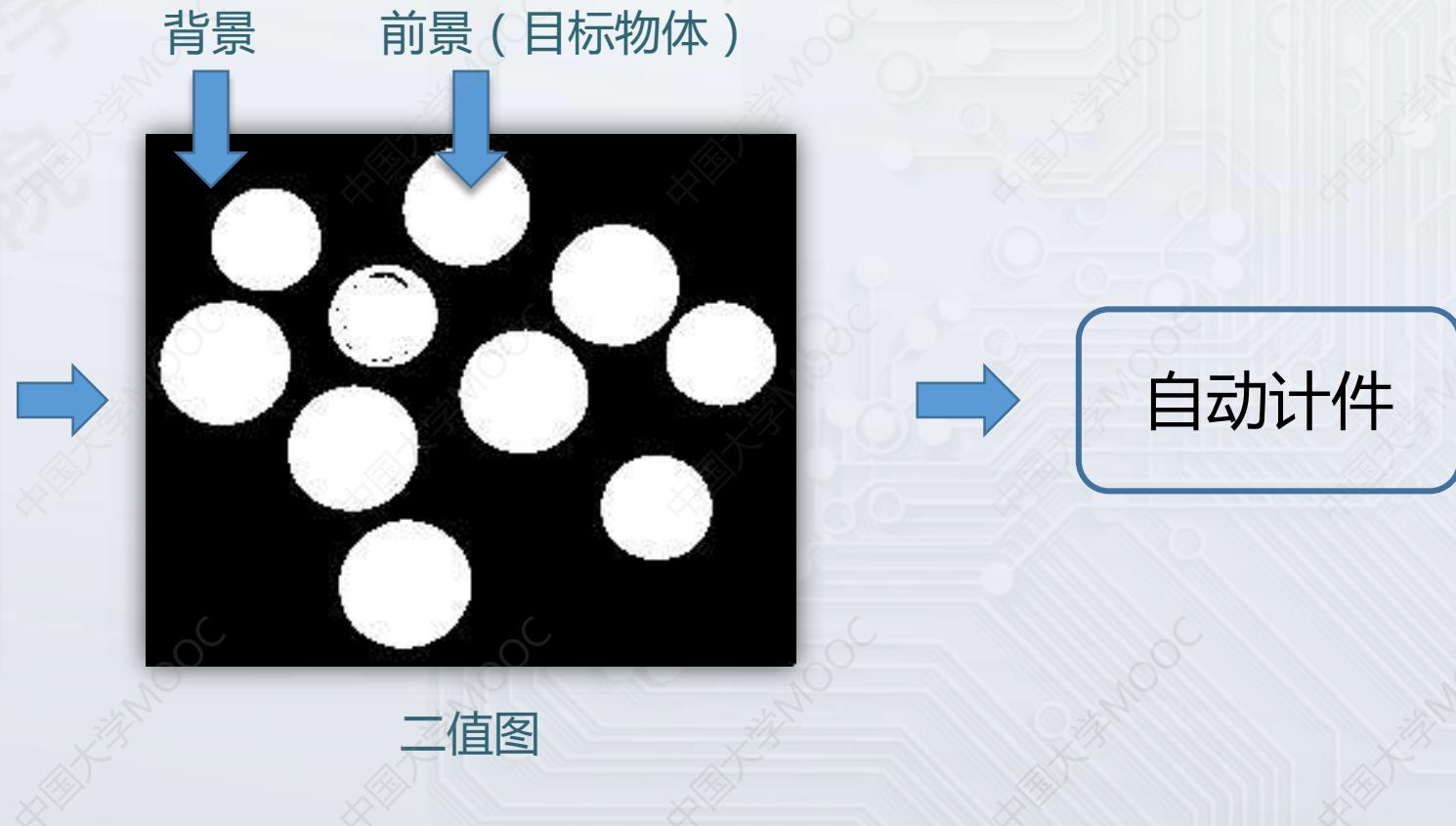
1. 图像二值化的目的
2. 几种全局二值化方法
3. 局部自适应二值化
4. 连通域分析

# 图像二值化

指将256阶的灰度图通过合适的阈值，转换为黑白二值图。即，像素只有0,1两种取值（或0和255）。其目的通常为将图像的前后景进行分割，以便进行进一步的处理。



原图



# 图像二值化

指将256阶的灰度图通过合适的阈值，转换为黑白二值图。即，像素只有0,1两种取值（或0和255）。其目的通常为将图像的前后景进行分割，以便进行进一步的处理。



原图



# 图像二值化

二值化的数学表达，对于图像中的像素 $[x,y]$ ，其灰度值为 $f(x,y)$ , 设置门限值(阈值)为 $TH$ ，则：

$$g(x, y) = \begin{cases} 255 & \text{If } f(x, y) \geq TH \\ 0 & \text{If } f(x, y) < TH \end{cases}$$

# 图像二值化

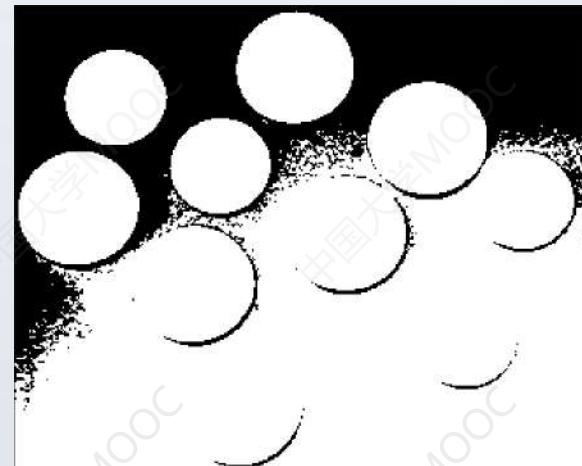
二值化的关键在于阈值的选择。合理的阈值应该尽可能的分离前景和背景。



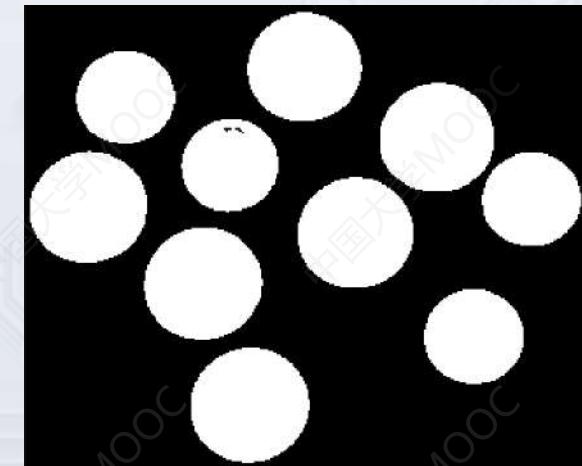
原图



阈值 : 200



阈值 : 100

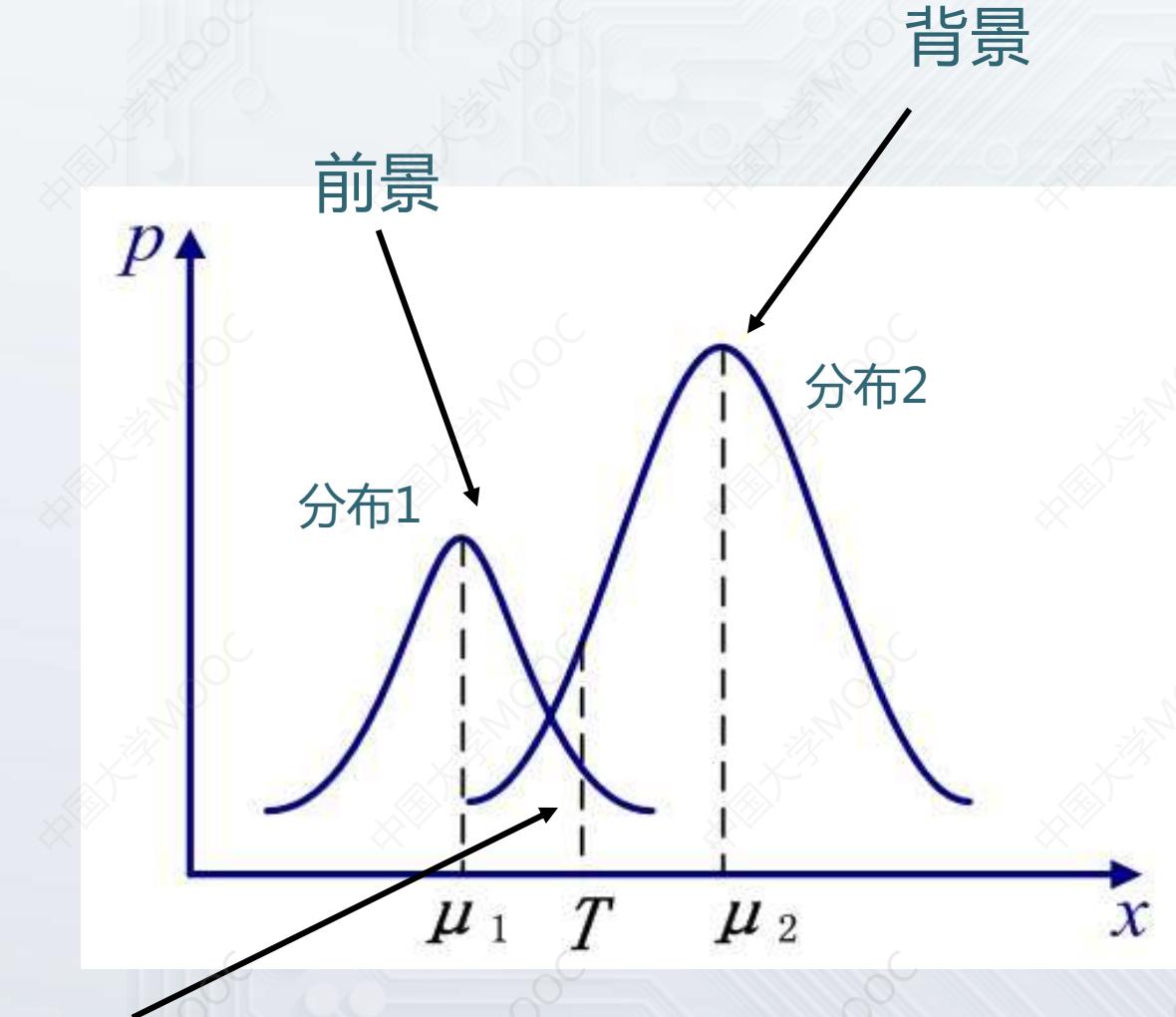


阈值 : 129

# 全局二值化

全局二值化即使用单一阈值对画面中的所有像素进行分割。通常可以分割直方图具有双峰性的图像。

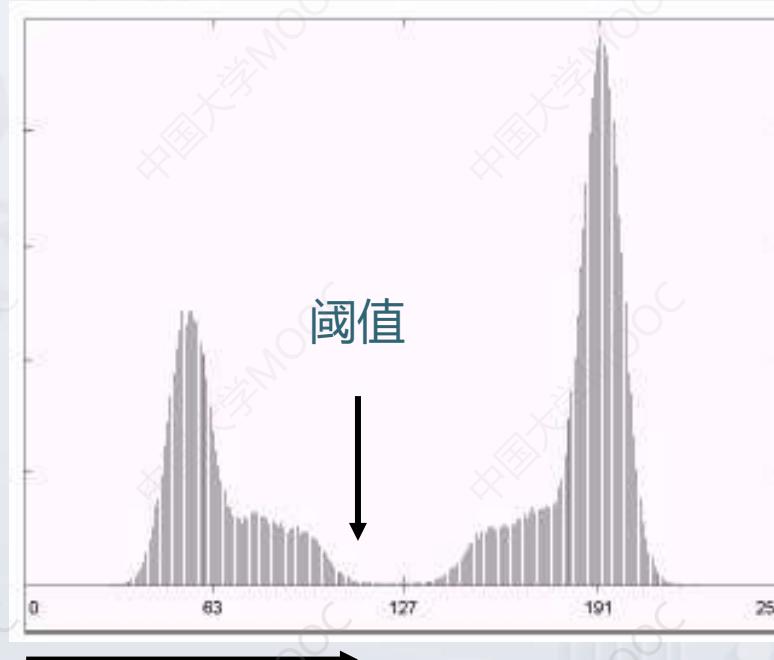
前景和背景的分布有交错部分，表示部分前景和背景像素颜色相同。由于有交错部分存在，通过阈值将前后景完全分离是不可能的，只能寻找最优解



前景和背景交错部分

# P-tile法

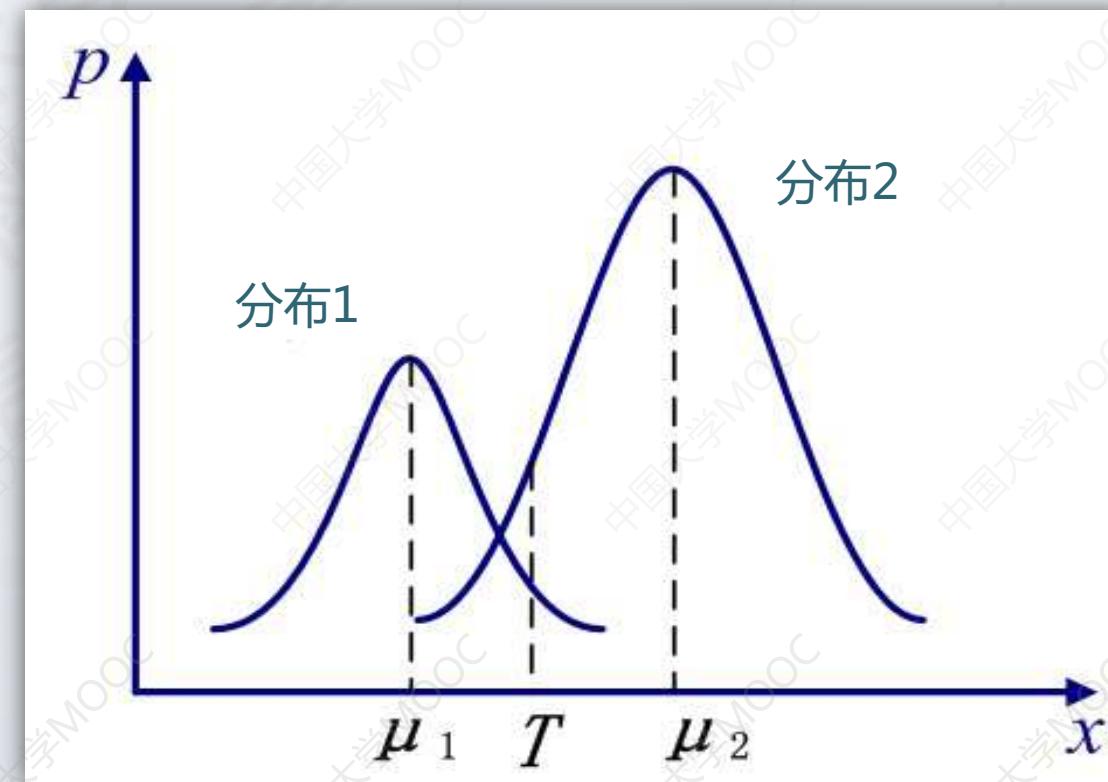
需要预先获得图像中前景占完整画面的比值 $P\%$ ，依次累积灰度直方图，直到该累积值大于或等于前景图像（目标）所占面积，此时的灰度级即为所求的阈值。



累加像素数量，当百分比达到预设的  
 $P\%$ 时，灰度级设为阈值

# 最小误判概率法

设前景像素点灰度概率密度函数为 $p(x)$ ，背景像素点灰度概率密度函数为 $q(x)$ ，分布函数如图。前景像素个数占图像总像素数的百分比为 $\theta_1$ ，背景为 $\theta_2 = 1 - \theta_1$



# 最小误判概率法

设分割阈值为 $T$ ，前景像素被错分为背景的概率为

$$E_1(T) = \theta_1 \int_T^{\infty} p(x)dx$$

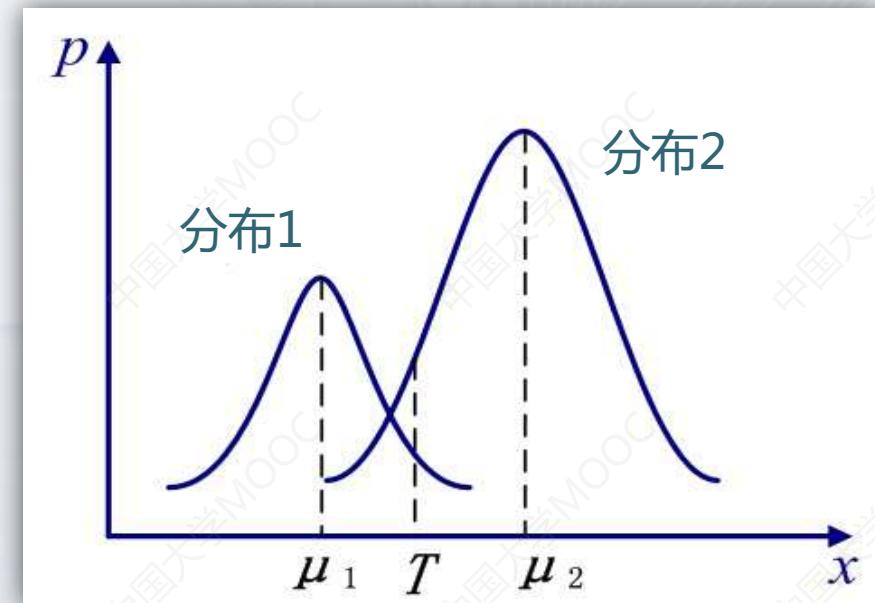
背景像素被错分为前景的概率为：

$$E_2(T) = \theta_2 \int_{-\infty}^T q(x)dx$$

阈值 $T$ 造成的错误分割概率为：

$$E(T) = \theta_1 \int_T^{\infty} p(x)dx + \theta_2 \int_{-\infty}^T q(x)dx$$

最小误判概率法就是求一个使 $E(T)$ 的数值最小的阈值 $T$



# 最小误判概率法

$E(T)$ 取得最小值时，其导数为0。

$$E(T) = \theta_1 \int_T^\infty p(x)dx + \theta_2 \int_{-\infty}^T q(x)dx$$

$$\frac{\partial E}{\partial T} = \theta_1 p(T) - \theta_2 q(T) = 0$$

$$\theta_1 p(T) = \theta_2 q(T)$$

假设图像中前景和背景像素灰度都呈正态分布,均值和方差分别为  $\mu_1, \sigma_1^2, \mu_2, \sigma_2^2$

所以有

$$\theta_2 e^{\frac{(\mu_2-T)^2}{2\sigma_2^2}} = \theta_1 e^{\frac{(\mu_1-T)^2}{2\sigma_1^2}}$$

# 最小误判概率法

为了便于计算，假设

$$\sigma_1^2 = \sigma_2^2 = \sigma^2$$

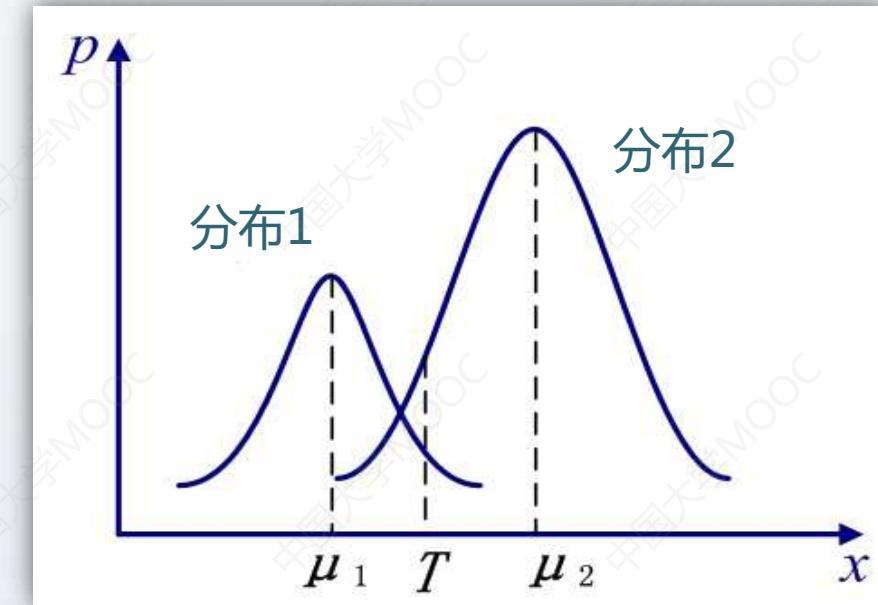
$$\theta_1 = \theta_2 = \frac{1}{2}$$

最佳阈值公式

$$\theta_2 e^{\frac{(\mu_2-T)^2}{2\sigma^2}} = \theta_1 e^{\frac{(\mu_1-T)^2}{2\sigma^2}}$$

$$\frac{(\mu_2-T)^2}{2\sigma^2} = \frac{(\mu_1-T)^2}{2\sigma^2}$$

$$T = \frac{\mu_1 + \mu_2}{2}$$



# 最小误判概率法

- 迭代法实现

1. 选择阈值 $T$ 的初始估计值。（求图像的最大灰度值A，最小灰度值B，令 $(A+B)/2$ 为初始值。）
2. 用 $T$ 分割图像，得到两组像素，即分割结果。 $G1$ 由所有灰度值大于 $T$ 的像素组成， $G2$ 由所有灰度值 $\leq T$ 的像素组成。
3. 对区域 $G1$ 和 $G2$ 中的所有像素计算平均灰度值 $\mu_1$ 和 $\mu_2$ 。
4. 计算新的阈值：
$$T = \frac{\mu_1 + \mu_2}{2}$$
5. 重复步骤2到4，两次迭代所得的 $T$ 值之差小于预设值，或者直接定义迭代次数。

# 大津(OTSU)法

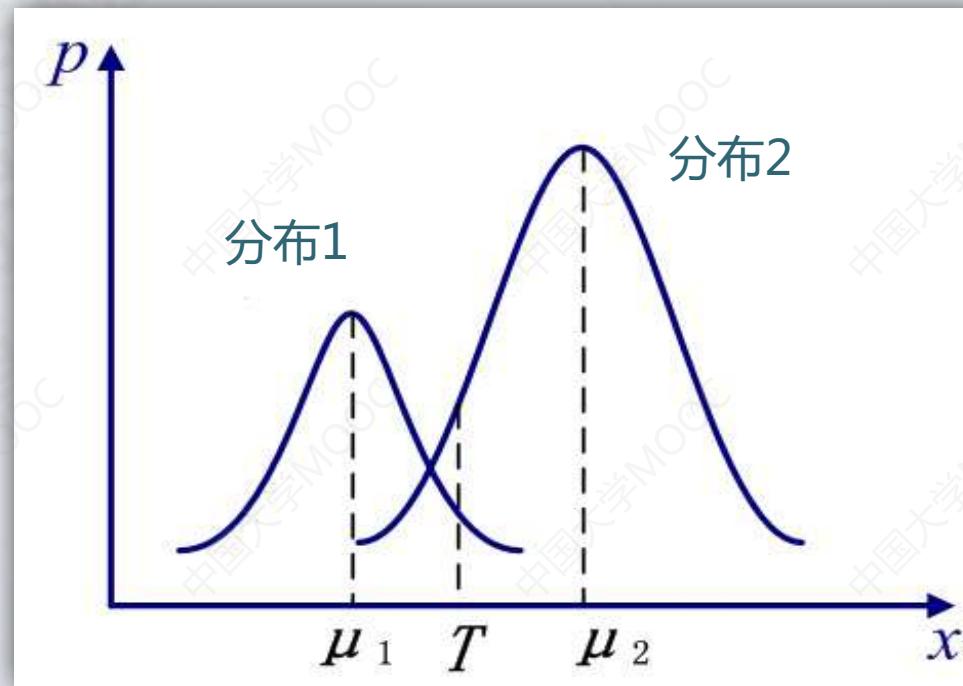
大津法考虑最佳的阈值应该使，类内方差尽可能小，类间方差尽可能大。

分布1：

像素数量： $w_1$

均值： $\mu_1$

$$\text{方差} : \sigma_1^2 = \frac{\sum_{i=1}^{w_1} (X_i - \mu_1)^2}{w_1}$$



分布2：

像素数量： $w_2$

均值： $\mu_2$

$$\text{方差} : \sigma_2^2 = \frac{\sum_{i=1}^{w_2} (X_i - \mu_2)^2}{w_2}$$

图像总像素数： $w_t$

均值： $\mu_t$

方差： $\sigma_t^2$

# 大津(OTSU)法

类内方差：
$$\sigma_w^2 = \frac{w_1\sigma_1^2 + w_2\sigma_2^2}{w_1 + w_2} = \frac{\sum_{i=1}^{w_1}(X_i - \mu_1)^2 + \sum_{j=1}^{w_2}(X_j - \mu_2)^2}{w_1 + w_2}$$

代入

$$\mu_t = \frac{w_1 \times \mu_1 + w_2 \times \mu_2}{w_1 + w_2}$$

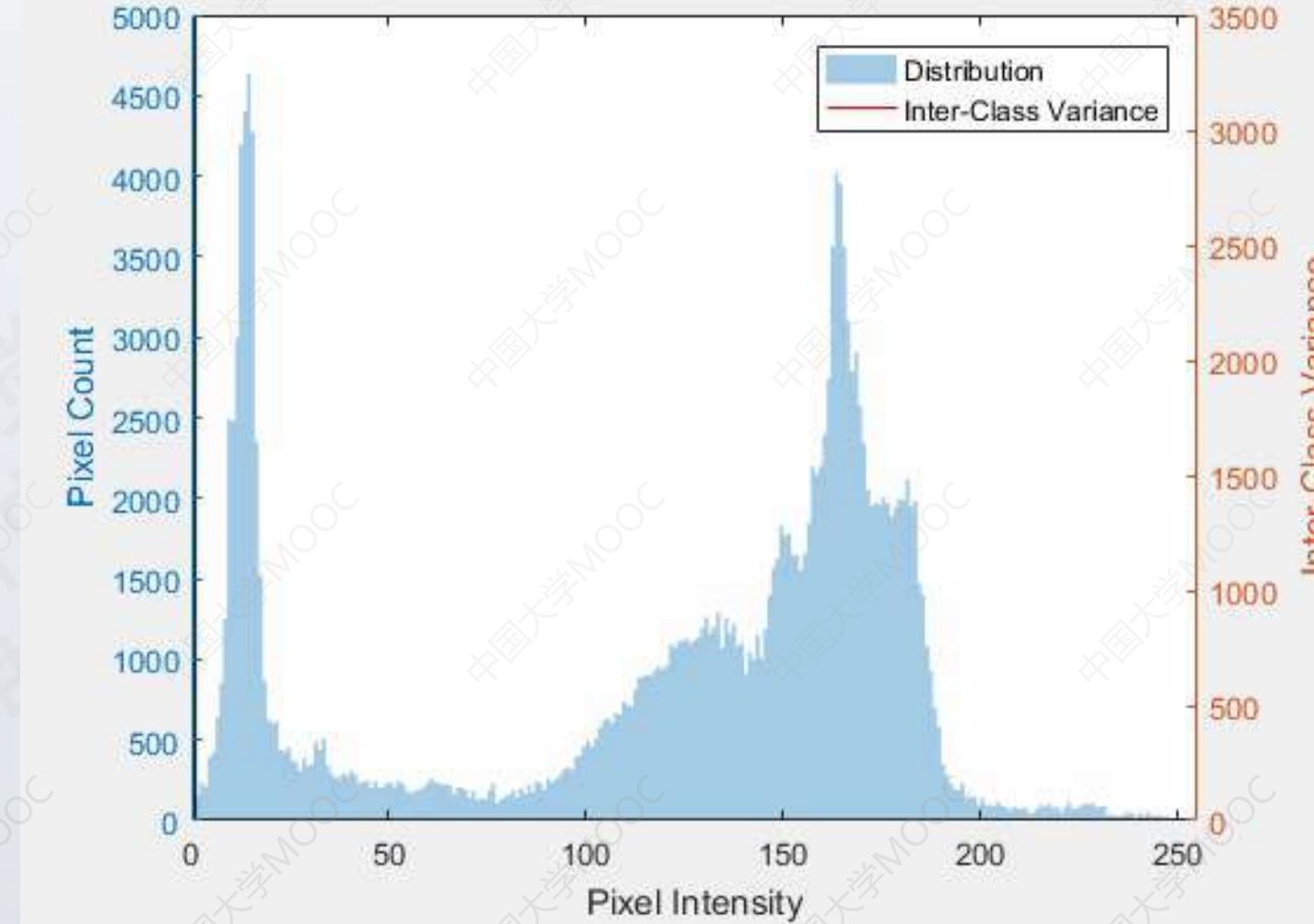
类间方差：
$$\sigma_b^2 = \frac{w_1(\mu_1 - \mu_t)^2 + w_2(\mu_2 - \mu_t)^2}{w_1 + w_2} = \frac{w_1 w_2 (\mu_1 - \mu_2)^2}{(w_1 + w_2)^2}$$

分离度：
$$\frac{\sigma_b^2}{\sigma_w^2}$$
      由于  $\sigma_t^2 = \sigma_w^2 + \sigma_b^2$       则      
$$\frac{\sigma_b^2}{\sigma_w^2} = \frac{\sigma_b^2}{\sigma_t^2 - \sigma_b^2}$$

求分离度最大时的阈值，即求  $\sigma_b^2$  最大，而类间方差的分母为固定值，因此即求分子最大时的阈值

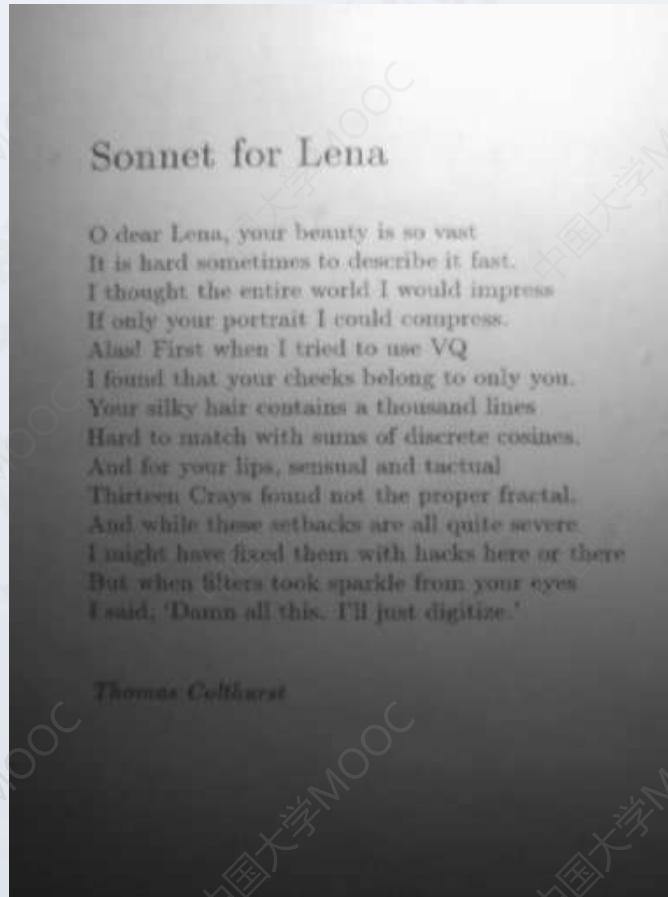
# 大津(OTSU)法

Demo By Lucas(CA) - Own work, CC BY-SA 4.0,  
<https://commons.wikimedia.org/w/index.php?curid=67144384>



# 局部自适应二值化

有一些场合，如左图，单一的阈值是不可能将前景的字母和背景分离开，因为一些字符的颜色和背景颜色是相同的。颜色分布上不呈双峰性。



# 局部自适应二值化

## Chow and Kaneko algorithm

- a. 将图像分成多个子区域
- b. 对每个区域求解阈值
- c. 通过插值法计算每个像素的阈值

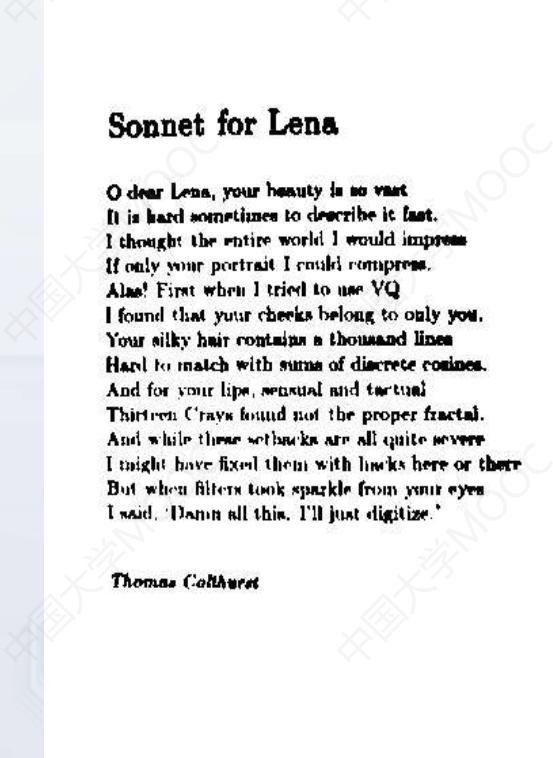
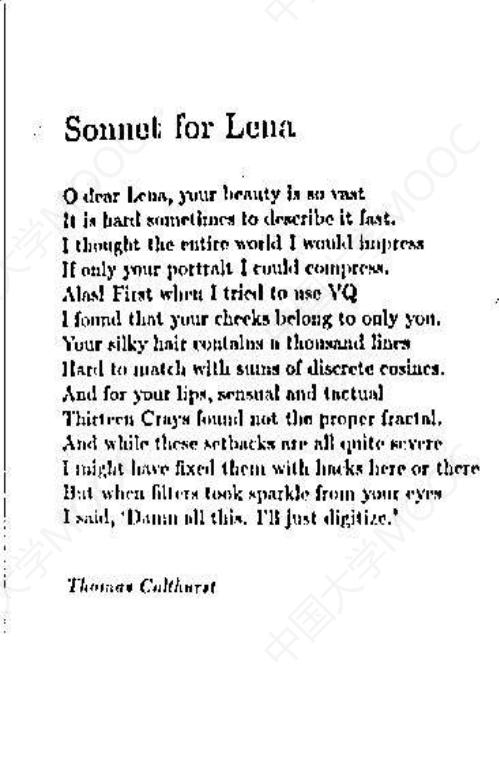
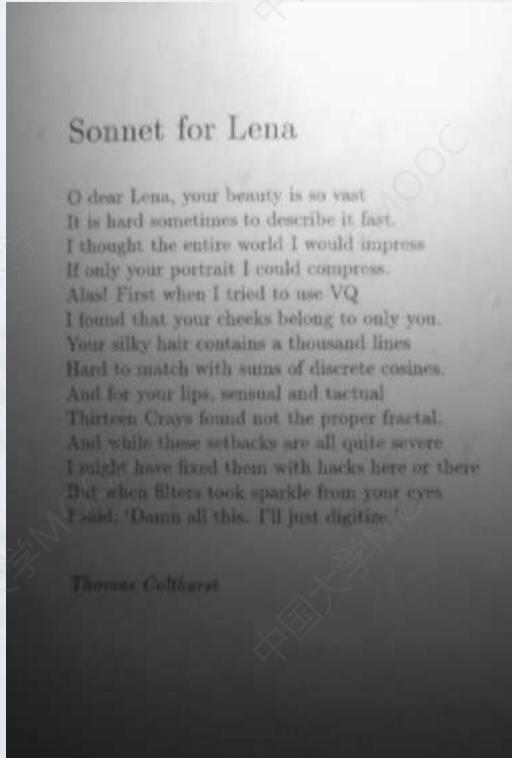
分成多个子区域，每个局部区  
域内的像素满足双峰性。



	Sonnet for Lena			
O dear Lena, your beauty is so vast It is hard sometimes to describe it fast. I thought the entire world I would impress If only your portrait I could compress.				
Alas! First when I tried to use VQ I found that your cheeks belong to only you. Your silky hair contains a thousand lines. Hard to match with sums of discrete cosines.				
And for your lips sensual and tactful Thirteen Crays found not the proper fractal. And while these setbacks are all quite severe I might have fixed them with hacks here or there.				
But when filters took sparkle Faded, 'Dance all this. I'll just digitize.'				
Thomas Colthurst				

# 局部自适应二值化

取不同尺寸的子区域的二值化效果。

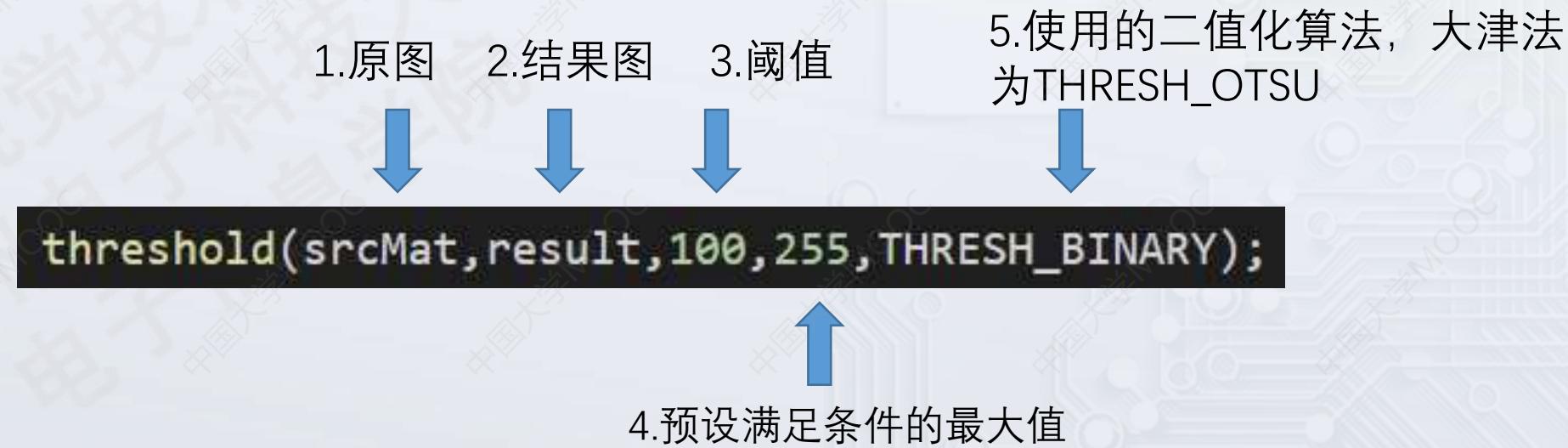


7x7

75x75

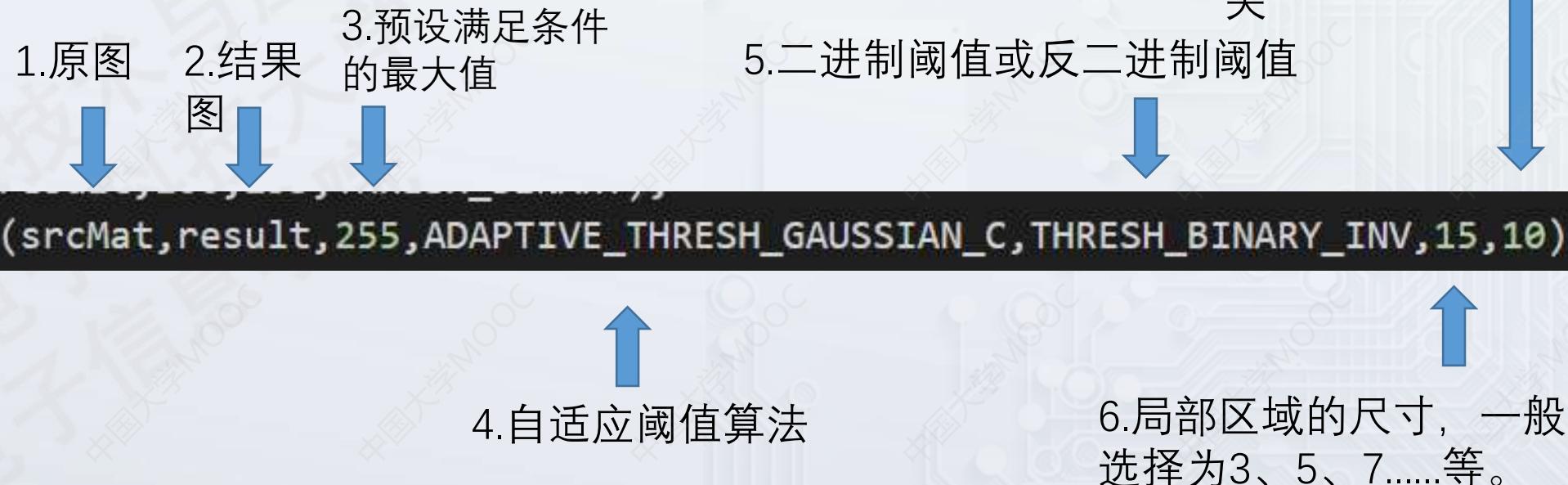
# Opencv中的二值化函数

Threshold的最后一个参数如果是THRESH\_BINARY，可以直接对彩色图像进行处理。如果使用大津法，输入必须为灰度图。可以先调用cvtColor进行格式转换处理。



# Opencv中的二值化函数

## 区域自适应二值化。



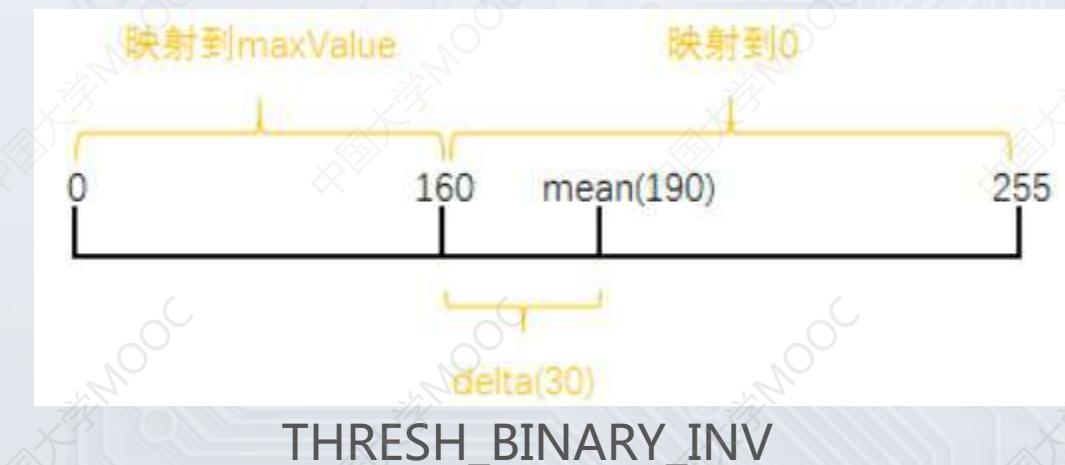
# Opencv中的二值化函数

## 关于参数4和参数5

参数4中：

ADAPTIVE\_THRESH\_MEAN\_C，为局部邻域块的平均值。该算法是先求出块中的均值，再减去常数C。

ADAPTIVE\_THRESH\_GAUSSIAN\_C，为局部邻域块的高斯加权和。该算法是在区域中(x, y)周围的像素根据高斯函数按照他们离中心点的距离进行加权计算，再减去常数C。

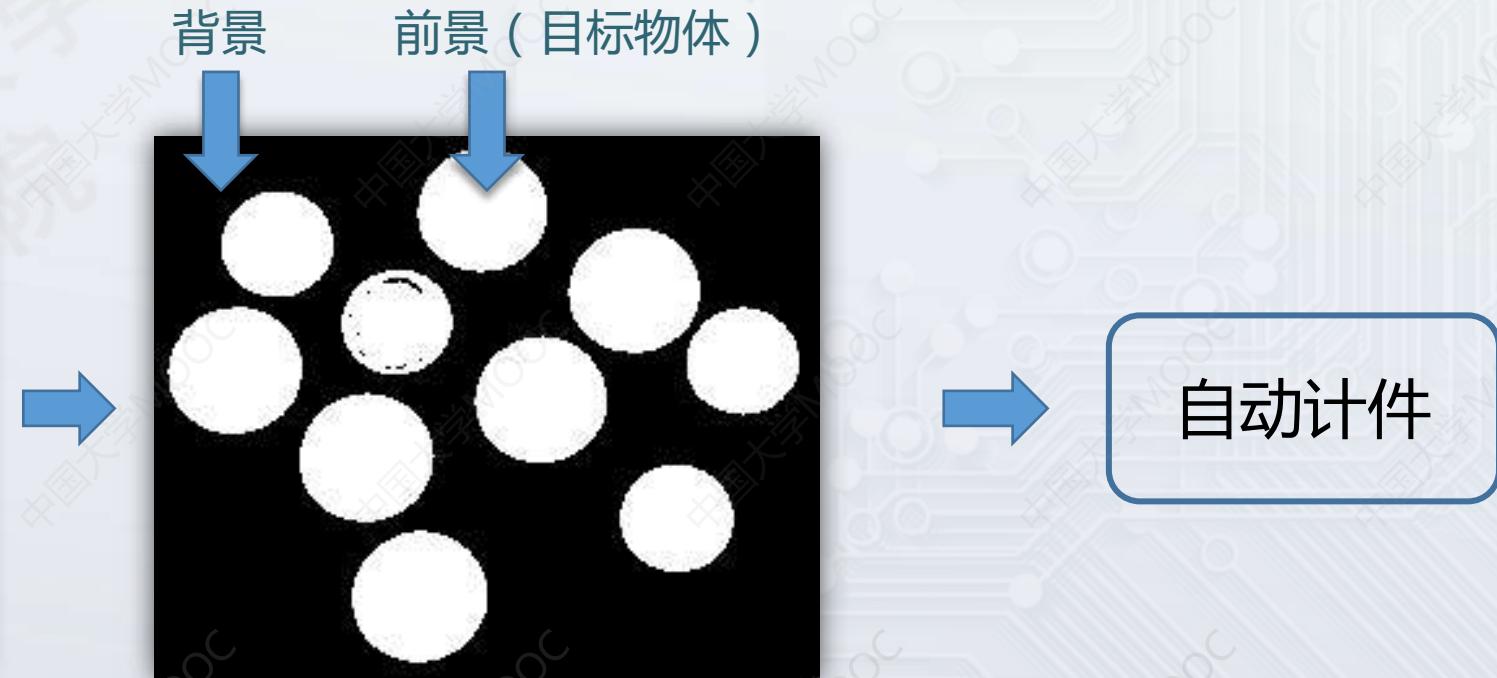


# 连通域标记

考虑：完成二值化后是否可以实现自动计件



原图



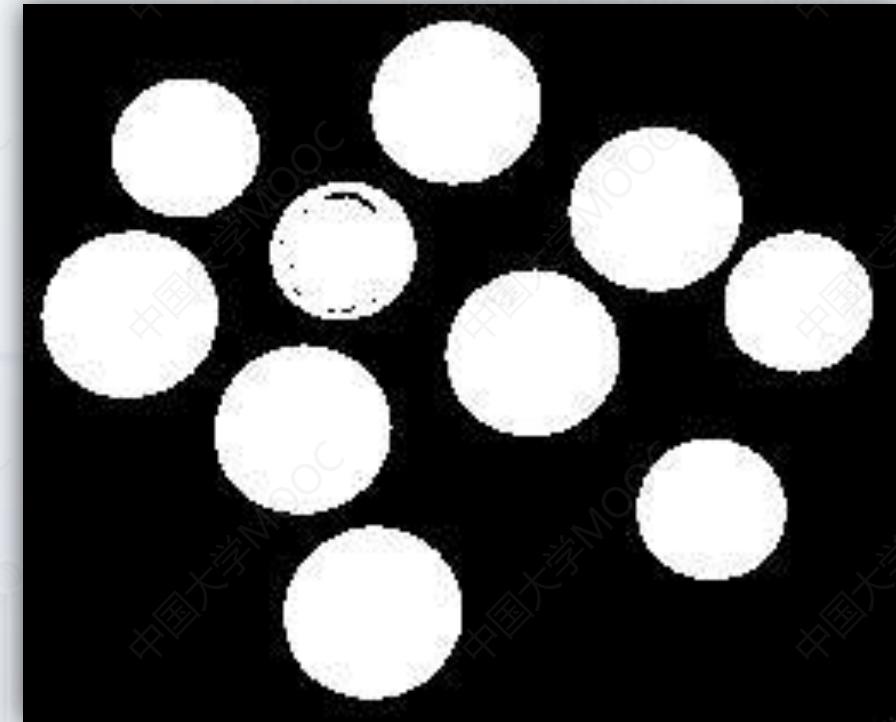
二值图

自动计件

# 连通域标记

为讨论方便，将二值化结果简化

				1				
	1	1	1	1				
	1	1	1	1				
			1			1		
				1	1			
						1		



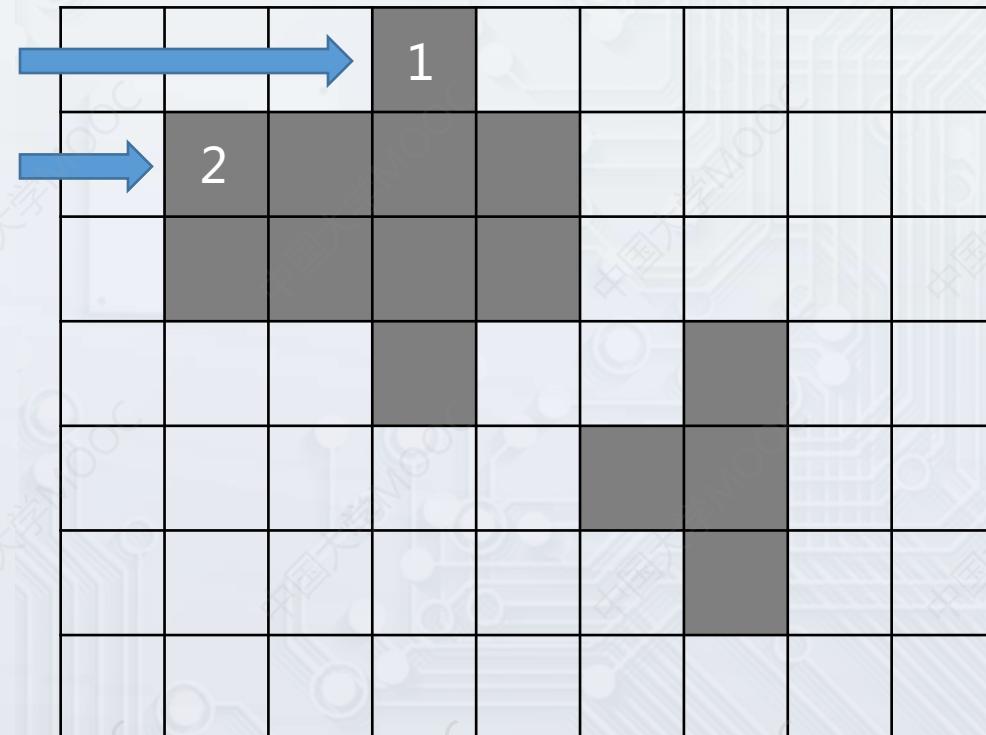


# 连通域标记

## Two-pass (等价表法)

## 1.第一次遍历。

a.当像素 $[x,y]$ 的领域（4-邻域）的所有像素都没有被赋标签值，则给 $[x,y]$ 一个新的标签，并且标签累加1。



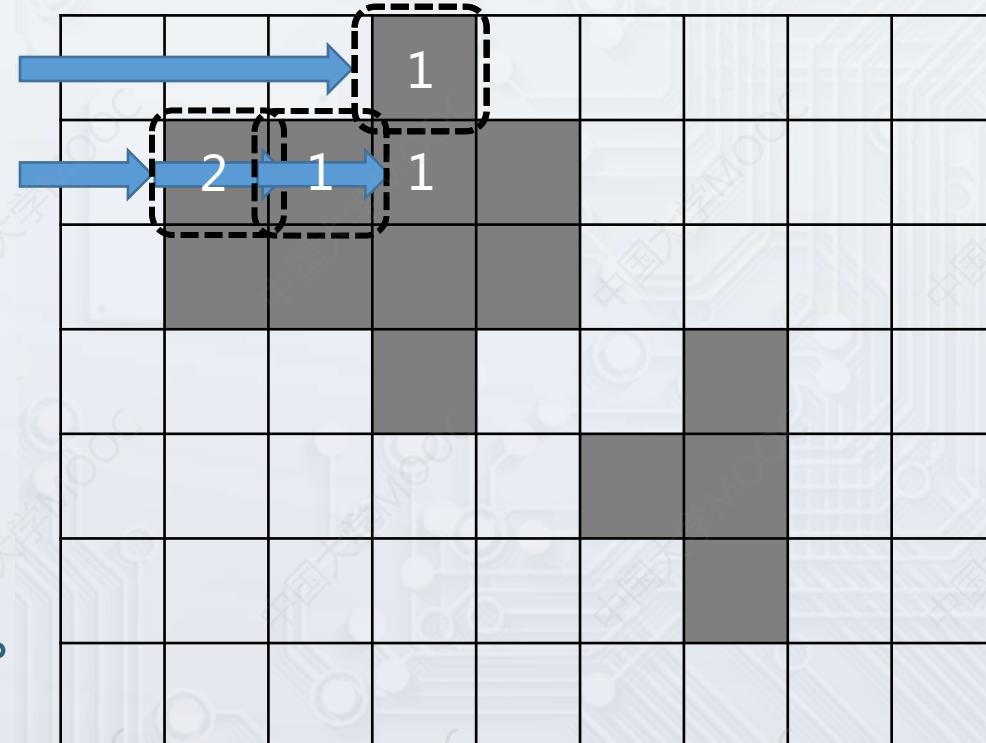
# 连通域标记

等价表 : { 1 2 }

## Two-pass ( 等价表法 )

### 1. 第一次遍历。

- a. 当像素 $[x,y]$ 的邻域（4-邻域）的所有像素都没有被赋标签值，则给 $[x,y]$ 一个新的标签，并且标签累加1。
- b. 当像素 $[x,y]$ 的邻域（4-邻域）中存在标签种类大于1时，则给 $[x,y]$ 一个邻域中值最小的标签。
- C. 当像素 $[x,y]$ 的邻域（4-邻域）中存在标签种类等于1时，则给 $[x,y]$ 一个相同的标签。



# 连通域标记

## Two-pass (等价表法)

## 1.第一次遍历。

- a. 当像素 $[x,y]$ 的邻域（4-邻域）的所有像素都没有被赋标签值，则给 $[x,y]$ 一个新的标签，并且标签累加1。
  - b. 当像素 $[x,y]$ 的邻域（4-邻域）中存在标签种类大于1时，则给 $[x,y]$ 一个邻域中值最小的标签。
  - c. 当像素 $[x,y]$ 的邻域（4-邻域）中存在标签种类等于1时，则给 $[x,y]$ 一个相同的标签。

等价表 :

$$\{1\ 2\} \quad \{3\ 4\}$$

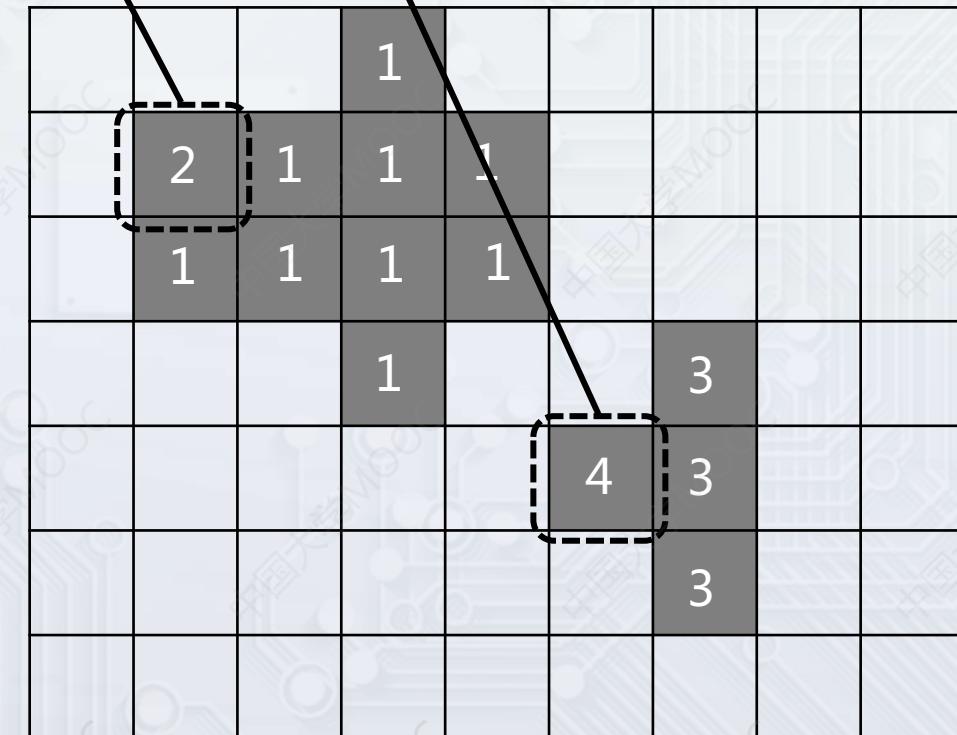
# 连通域标记

## Two-pass ( 等价表法 )

2.第二次遍历。  
给所有的像素重新赋值，值为等价表中的最小值。

等价表 :

{ 1 2 } { 3 4 }



# 连通域标记

## Two-pass ( 等价表法 )

## 2.第二次遍历。

给所有的像素重新赋值，值为等价表中的最小值。

## 等价表

$$\{1\ 2\} \quad \{3\ 4\}$$

			1				
	1	1	1	1			
	1	1	1	1			
			1			3	
					3	3	
						3	

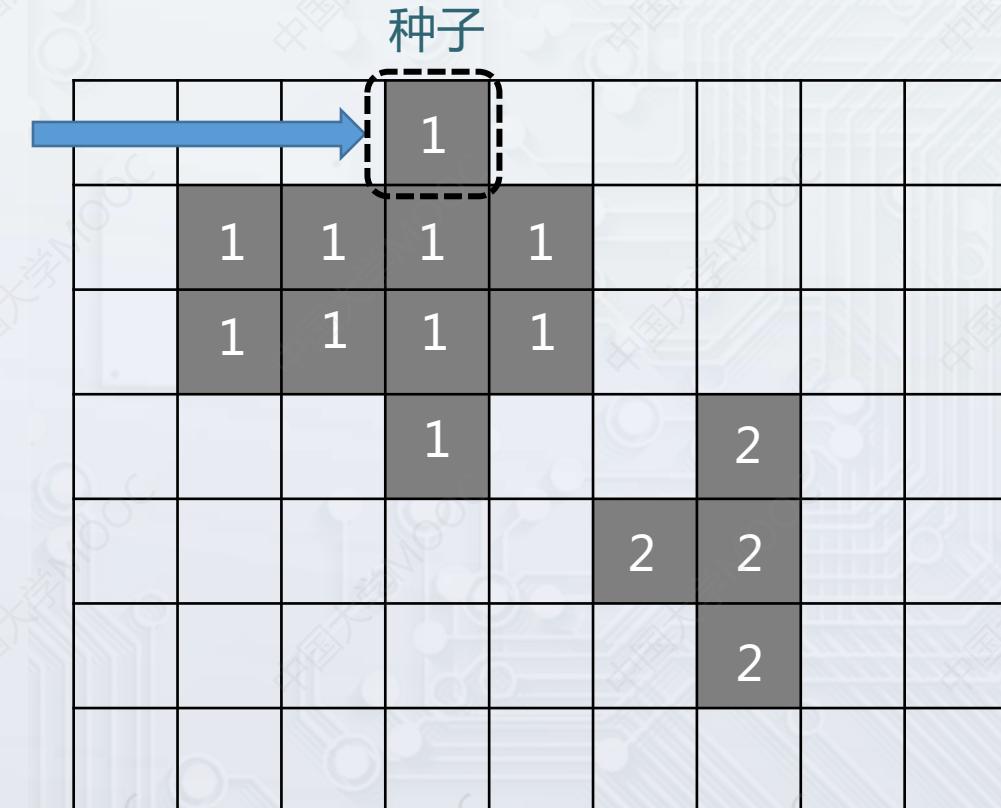
# 连通域标记

## Seed Filling ( 种子填充法 )

### 1. 遍历。

当像素 $[x,y]$ 为前景，则作为种子。给与一个标签，将后将与之相邻的前景像素全部给与相同标签。重复该步骤。

重复以上步骤。只到遍历结束



# Opencv中的连通域标记函数

connectedComponentsWithSatas函数，可以对黑白二值图进行连通域标记，同时返回连通域的状态和中心坐标。

函数原型：



# Opencv中的连通域标记函数

## 3.状态矩阵

包括背景共有11连通域



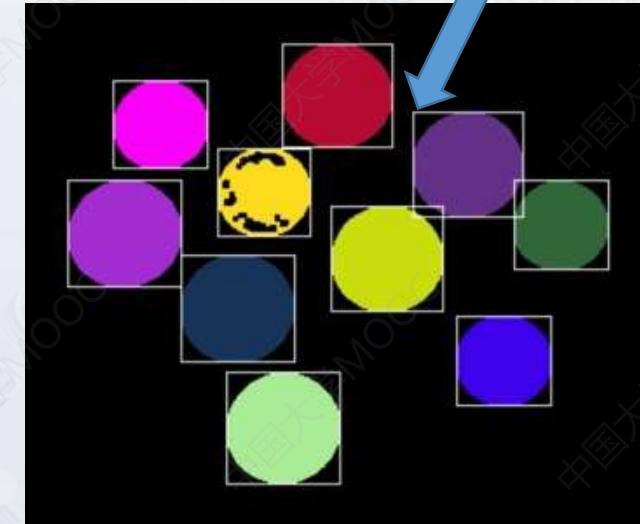
```
CV_EXPORTS_W int connectedComponentsWithStats(InputArray image, OutputArray labels,  
OutputArray stats, OutputArray centroids,  
int connectivity = 8, int ltype = CV_32S);
```

状态矩阵size=11×5

[0, 0, 368, 309, 86075;  
150, 27, 64, 60, 3034;  
53, 48, 55, 51, 2232;  
225, 66, 64, 61, 3080;  
113, 87, 54, 51, 1843;  
27, 105, 66, 62, 3202;  
283, 105, 55, 52, 2263;  
178, 120, 65, 61, 3133;  
92, 148, 66, 62, 3231;  
250, 183, 55, 52, 2289;  
118, 215, 66, 65, 3330]

第一行为背景

bounding box



每行5个参数分别为该连通域最小外接四边形  
( bounding box ) 的 x, y, width, height 和面  
积 ( 像素数量 )。

# Opencv中的连通域标记函数

## 3.连通域中心的矩阵

包括背景共有11连通域



```
CV_EXPORTS_W int connectedComponentsWithStats(InputArray image, OutputArray labels,  
OutputArray stats, OutputArray centroids,  
int connectivity = 8, int ltype = CV_32S);
```

中心坐标矩阵size=11×2

[185.9378797560267, 157.9491606157421;  
181.318391562294, 56.63941990771259;  
80.24148745519713, 73.30913978494624;  
256.1298701298701, 96.31331168831169;  
140.192078133478, 111.6370048833424;  
59.49531542785759, 135.6611492816989;  
309.5810870525851, 131.0565620857269;  
210.0705394190871, 150.1184168528567;  
124.0959455277004, 178.6146703806871;  
276.9467016164264, 208.4713848842289;  
150.4366366366366, 247.0654654654655]

第一行为背景

每一行为一个连通域的中心坐标 [x, y]。

# 谢谢！

# 机器视觉技术与应用

## 4. 图像形态学

李竹

杭州电子科技大学

电子信息学院



# 本章概要

1. 图像形态学的基本运算
2. 开运算和闭运算
3. 灰度图的形态学运算

# 图像形态学

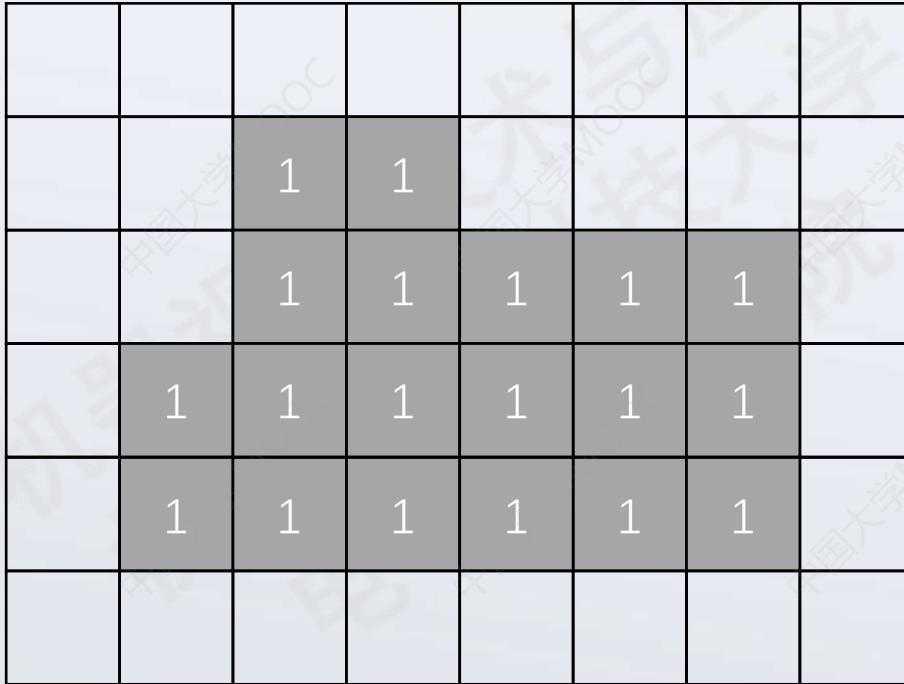
图像形态学的理论基础为**集合论**。

图像中的集合代表二值图像或者灰度图像的形状。如二值图像的前景像素集合。

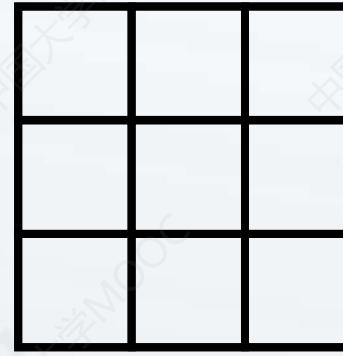
图像形态学的作用是简化图像数据，保持基本形状特性，去除不相干的结构等。

基本运算包括，膨胀、腐蚀、开运算、闭运算、顶帽运算和底帽运算等。

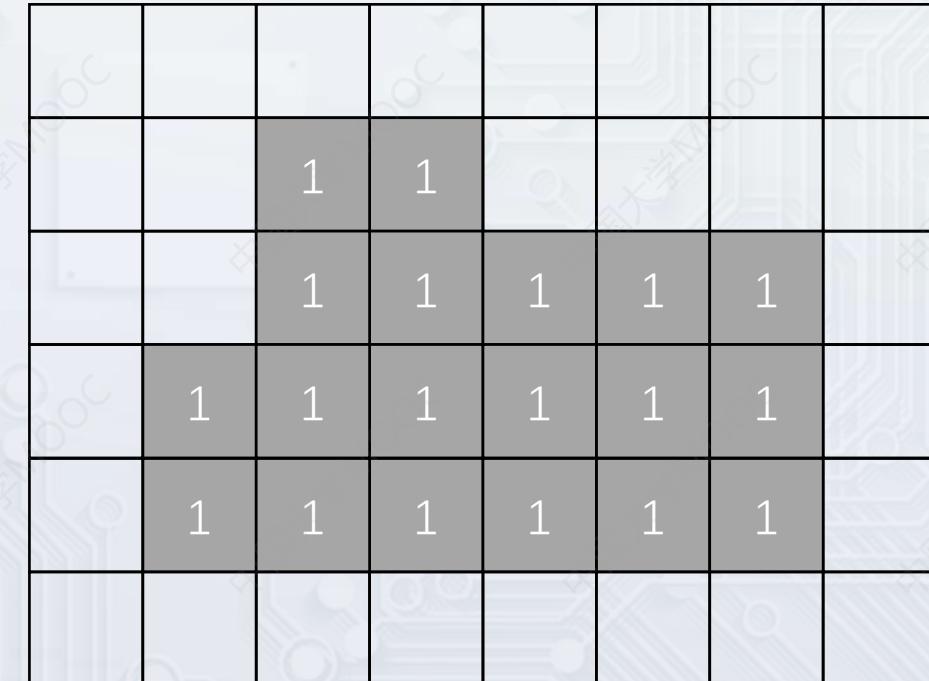
# 膨胀运算



输入二值图像图像，值为1的像素为待处理前景像素集合



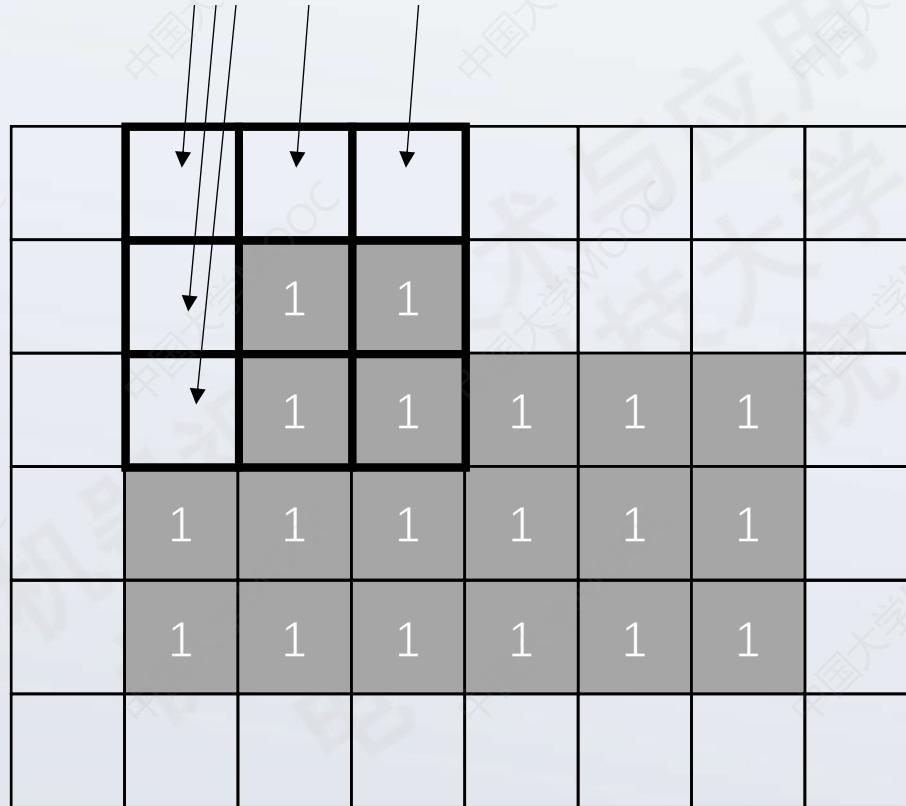
3×3结构元素，中心位置为锚点，使用结构元素遍历所有待处理像素。



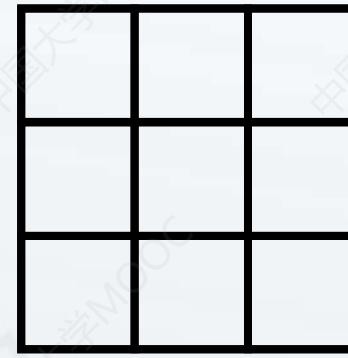
膨胀运算处理结果

# 膨胀运算

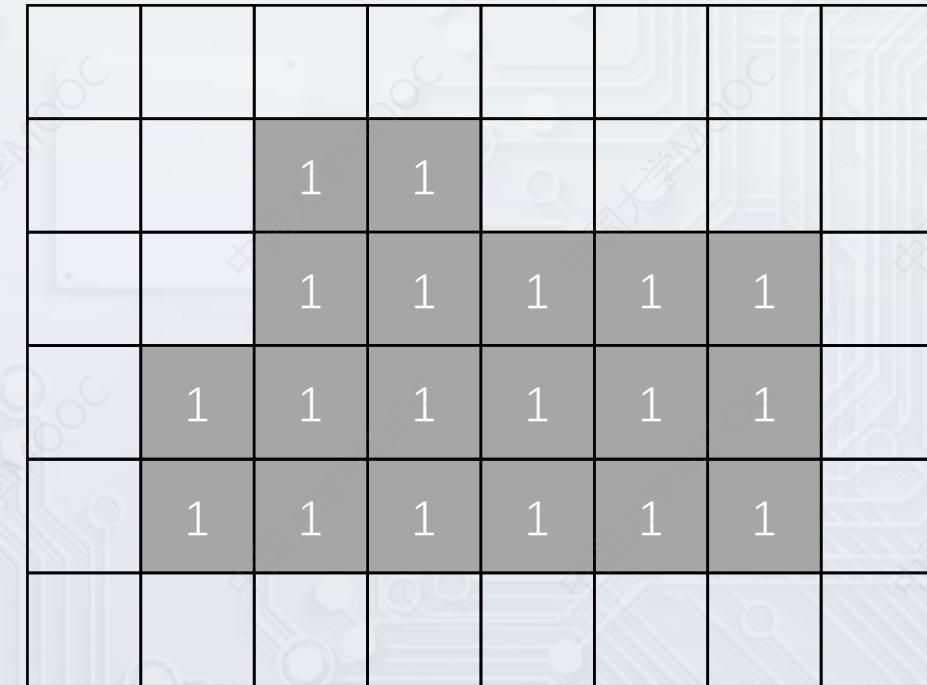
结构元素覆盖的所有点置1



输入二值图像图像，值为1的像素为待处理前景像素集合



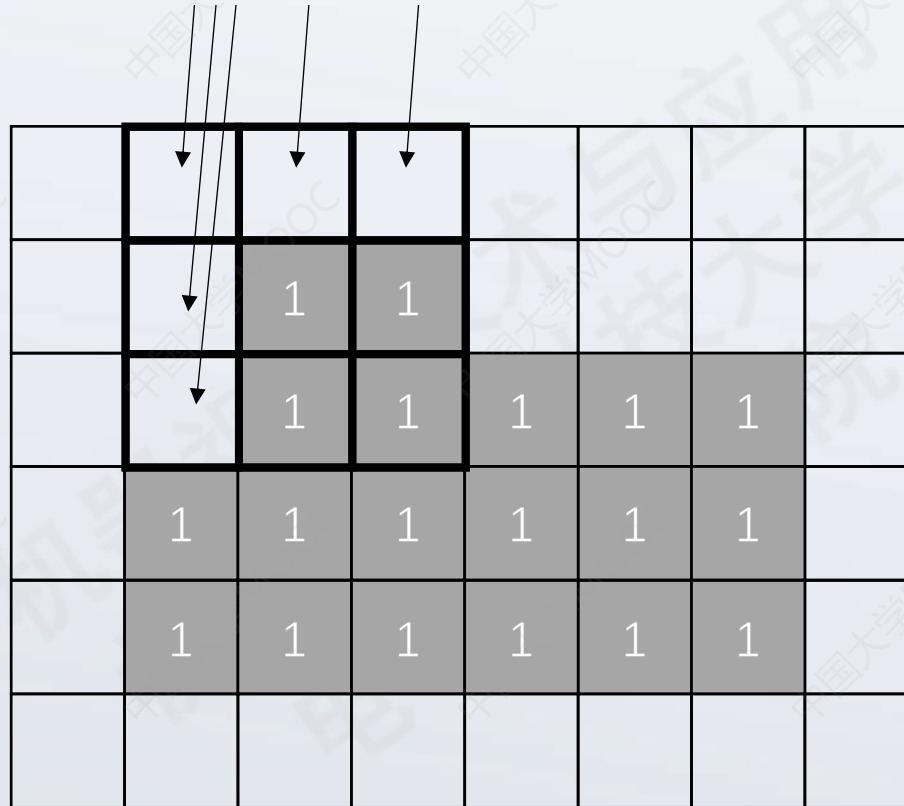
3×3结构元素，中心位置为锚点，使用结构元素遍历所有待处理像素，遍历时，锚点对齐待处理像素。



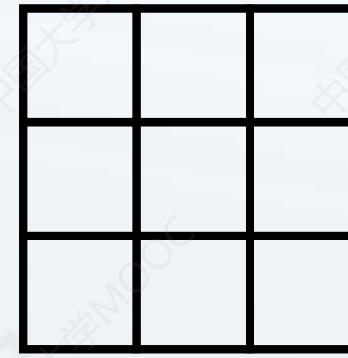
膨胀运算处理结果

# 膨胀运算

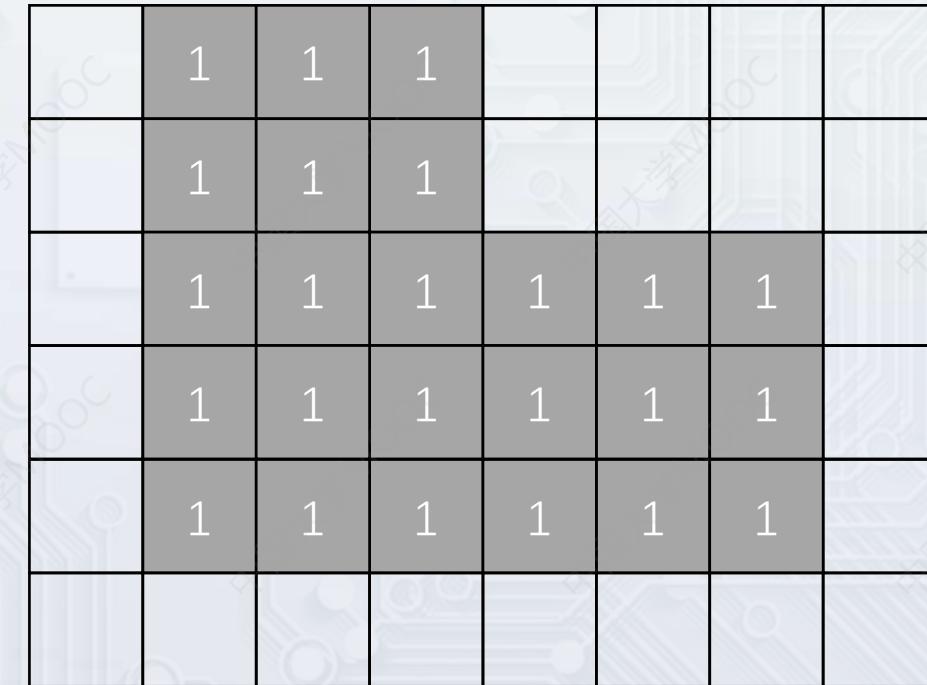
结构元素覆盖的所有点置1



输入二值图像图像，值为1的像素为待处理前景像素集合

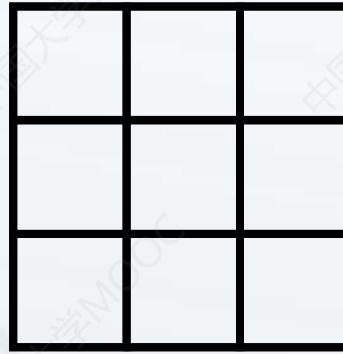


3×3结构元素，中心位置为锚点，使用结构元素遍历所有待处理像素，遍历时，锚点对齐待处理像素。

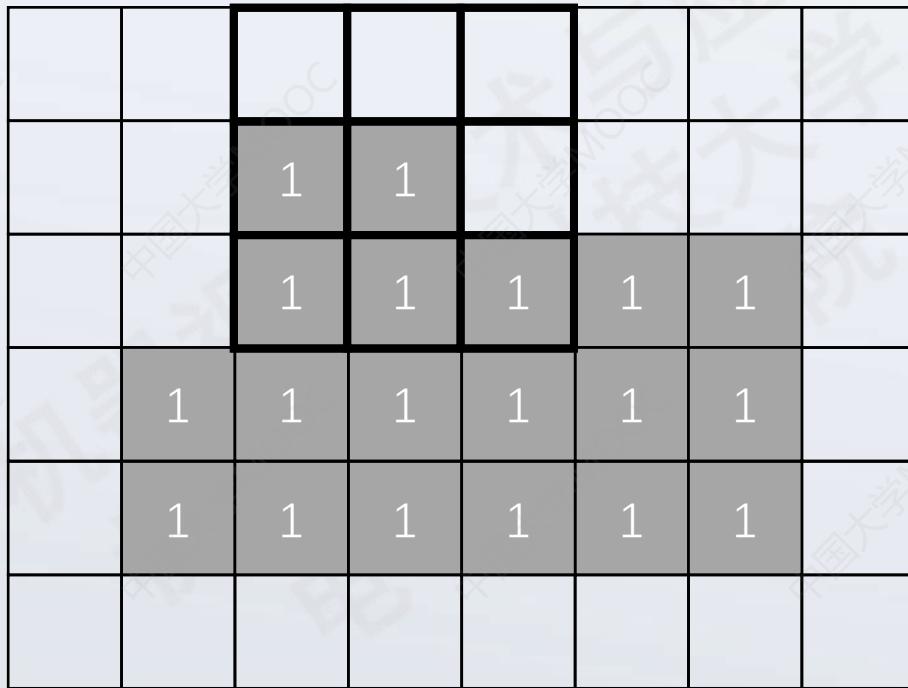


膨胀运算处理结果

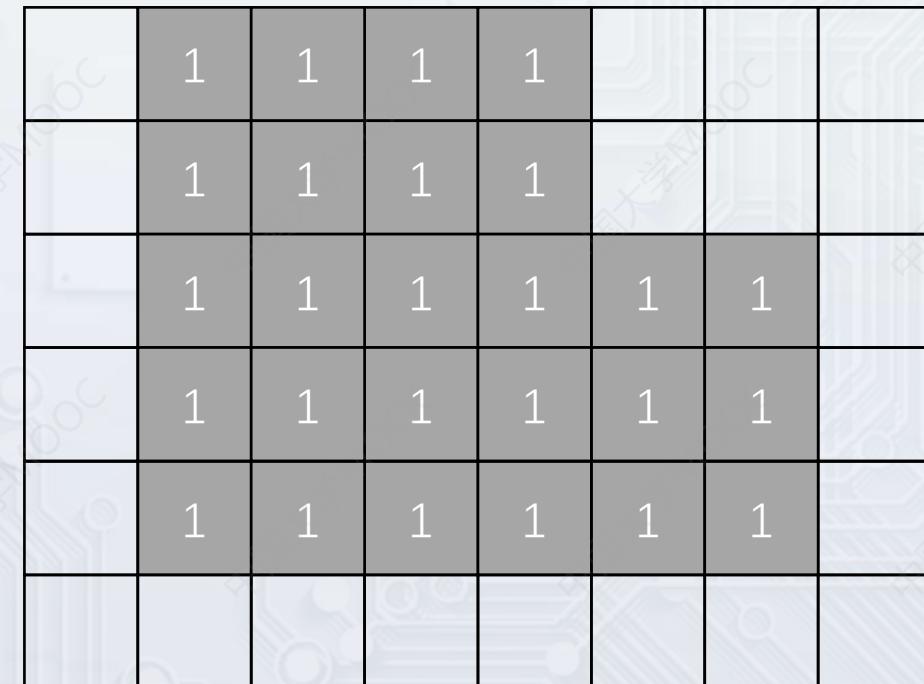
# 膨胀运算



3×3结构元素，中心位置为锚点，使用结构元素遍历所有待处理像素，遍历时，锚点对齐待处理像素。



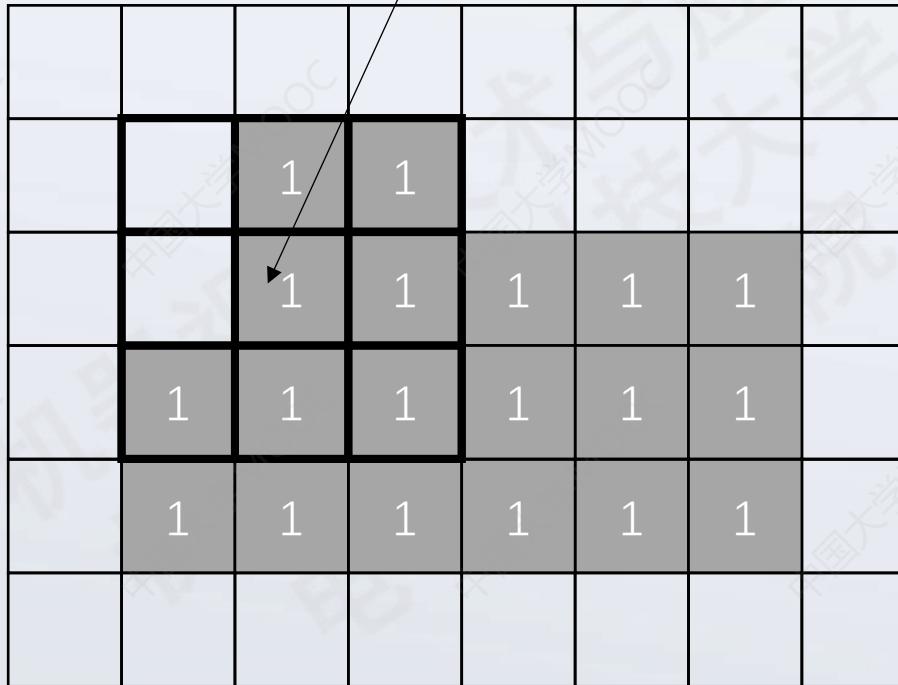
输入二值图像图像，值为1的像素为待处理前景像素集合



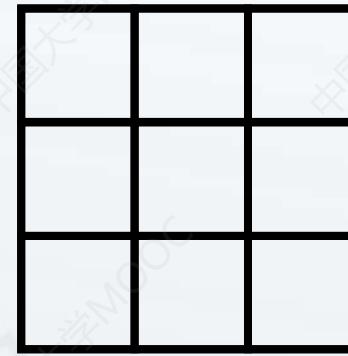
膨胀运算处理结果

# 膨胀运算

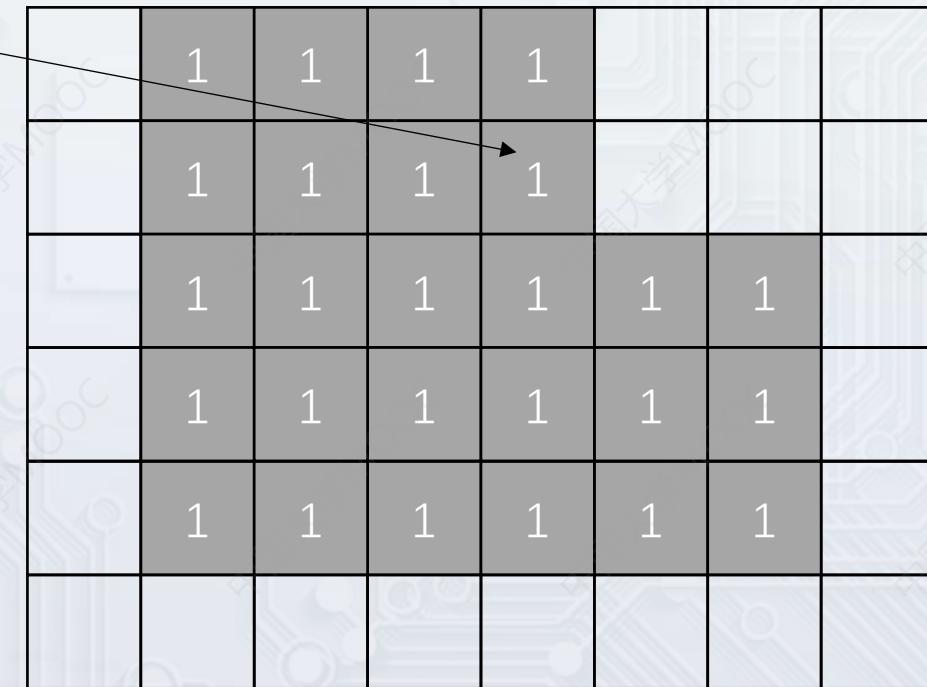
注意：这里是遍历输入图像，而不是遍历膨胀图像，故第三步对齐的像素位为，而非



输入二值图像图像，值为1的像素为待处理前景像素集合

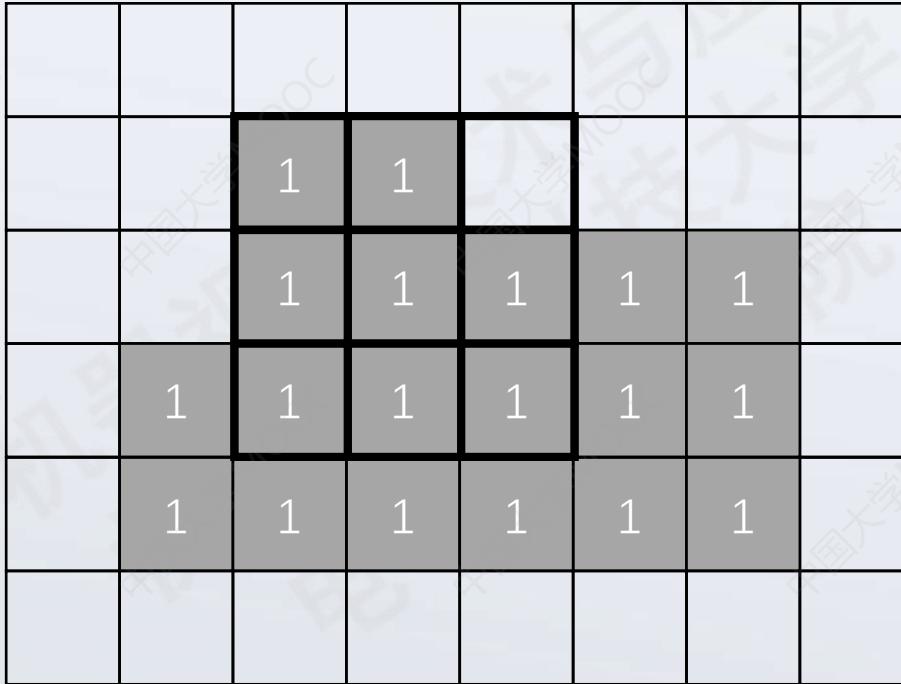


3×3结构元素，中心位置为锚点，使用结构元素遍历所有待处理像素，遍历时，锚点对齐待处理像素。

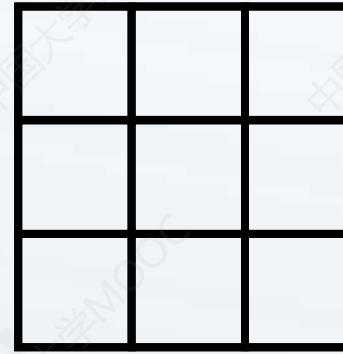


膨胀运算处理结果

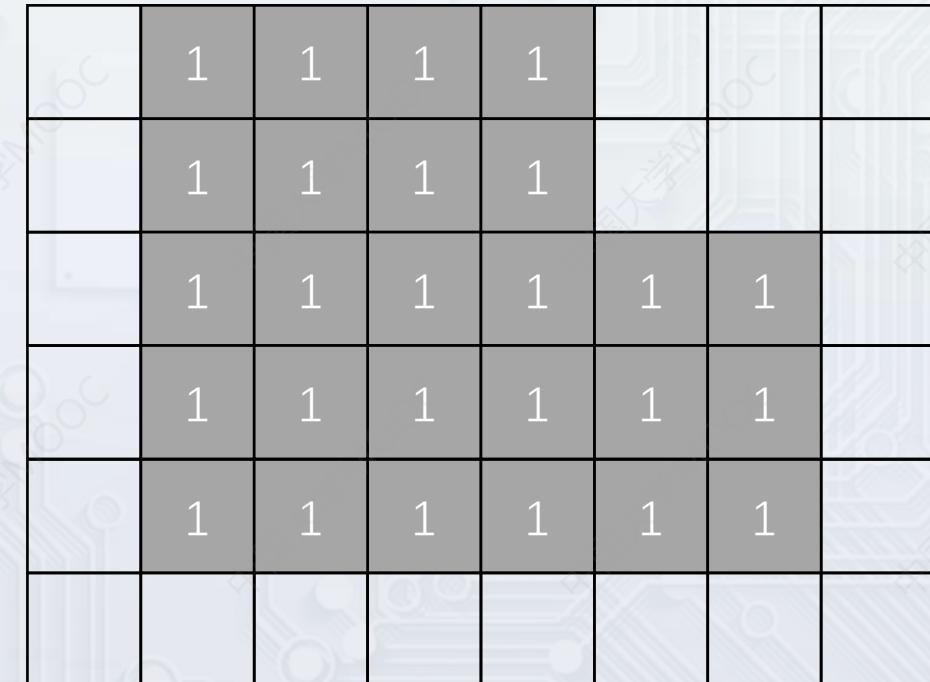
# 膨胀运算



输入二值图像图像，值为1的像素为待处理前景像素集合

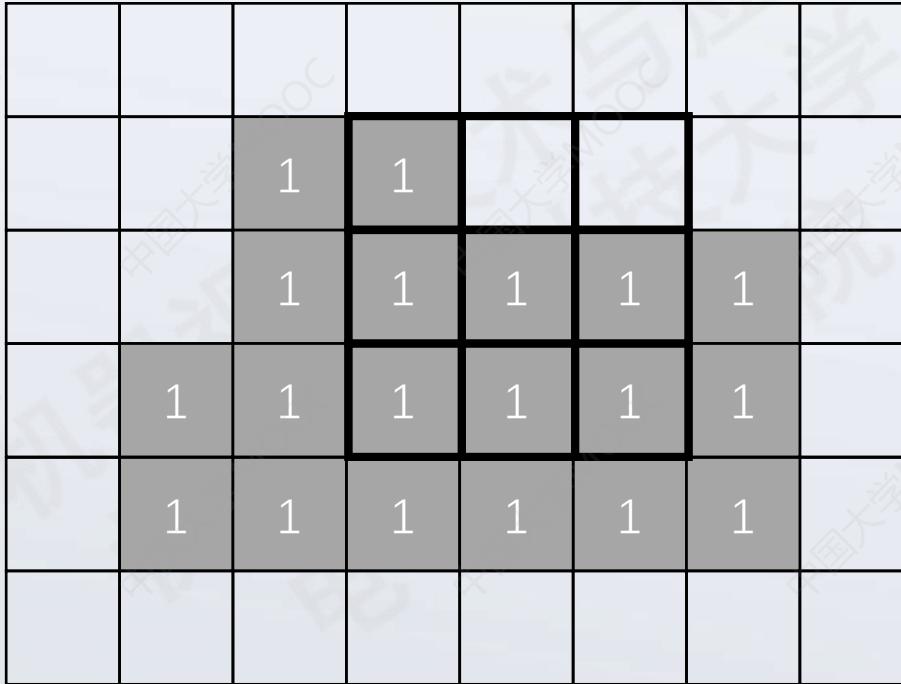


3×3结构元素，中心位置为锚点，使用结构元素遍历所有待处理像素，遍历时，锚点对齐待处理像素。

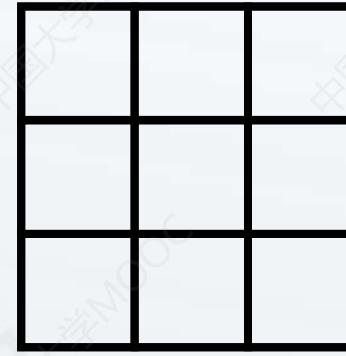


膨胀运算处理结果

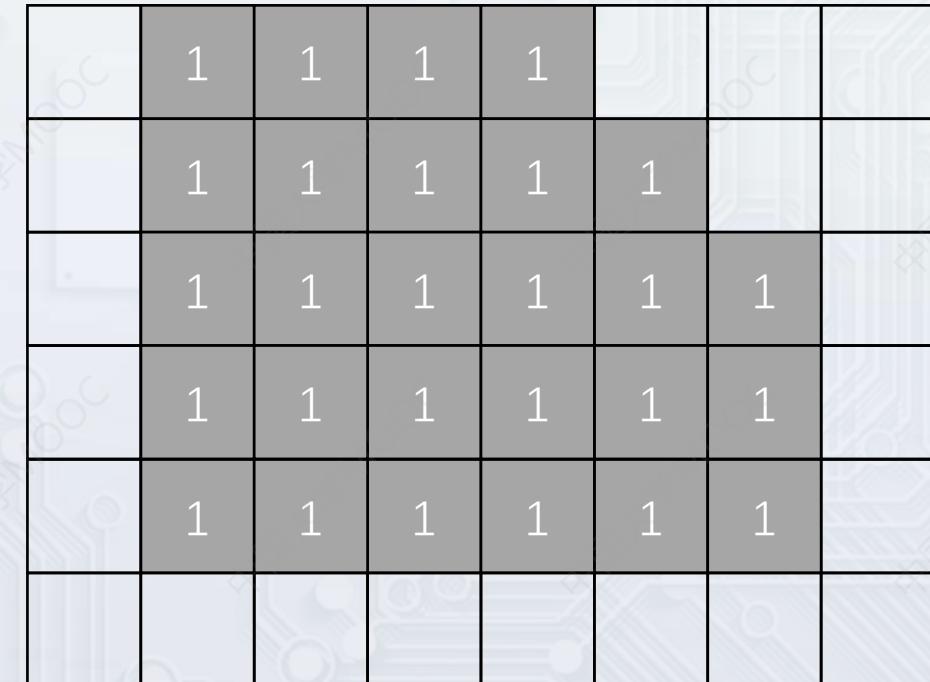
# 膨胀运算



输入二值图像图像，值为1的像素为待处理前景像素集合

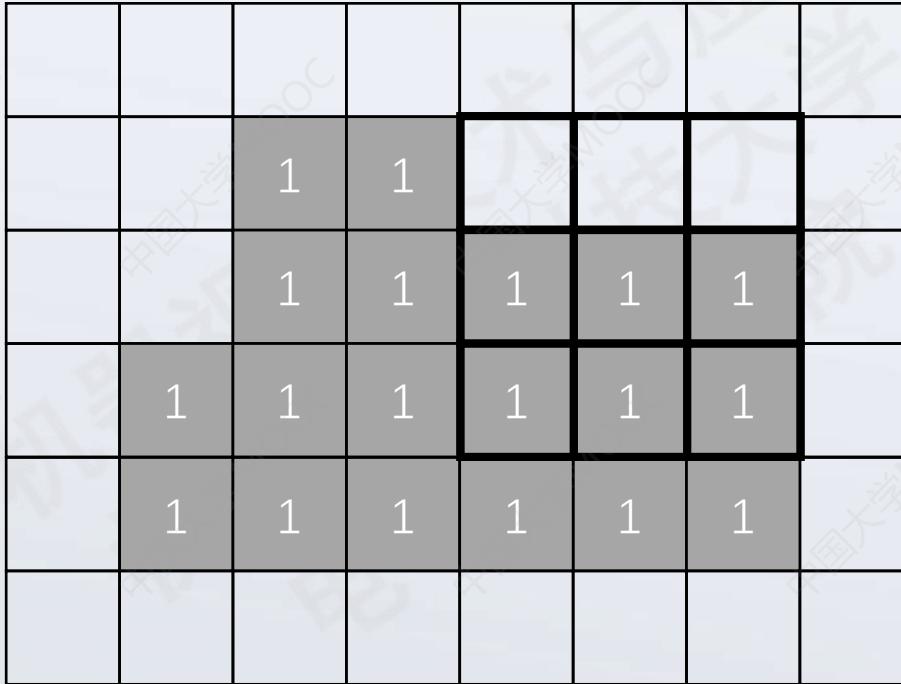


3×3结构元素，中心位置为锚点，使用结构元素遍历所有待处理像素，遍历时，锚点对齐待处理像素。

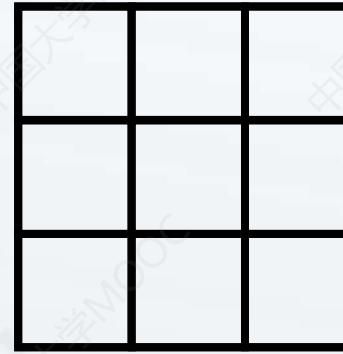


膨胀运算处理结果

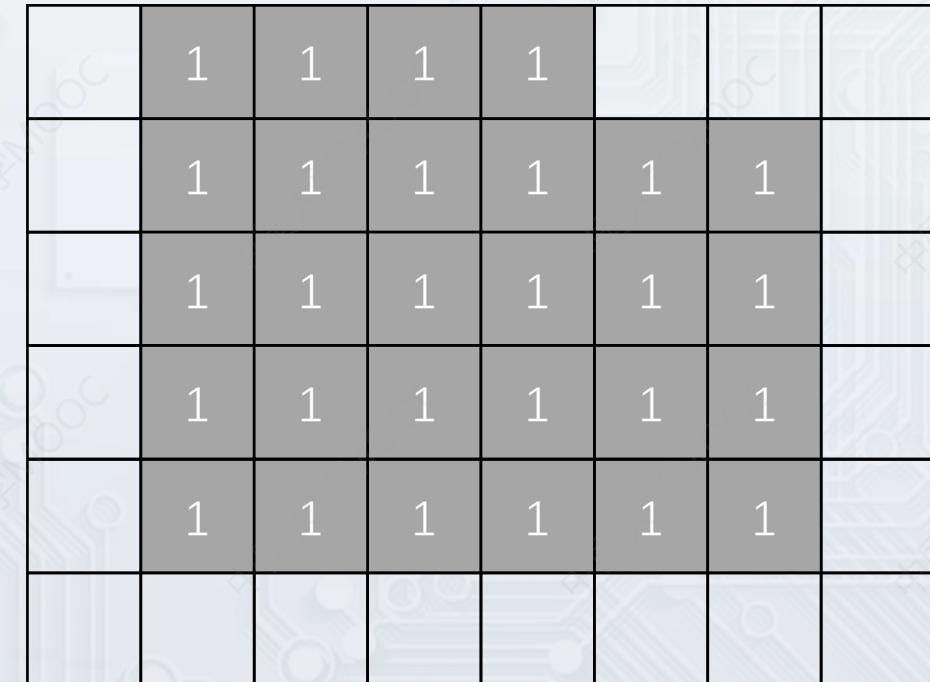
# 膨胀运算



输入二值图像图像，值为1的像素为待处理前景像素集合

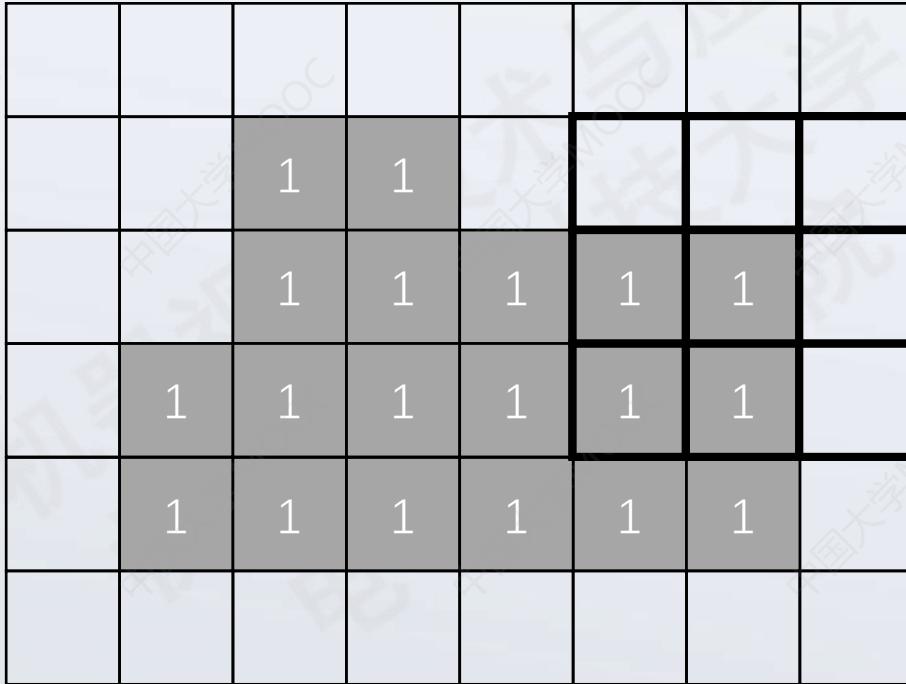


3×3结构元素，中心位置为锚点，使用结构元素遍历所有待处理像素，遍历时，锚点对齐待处理像素。

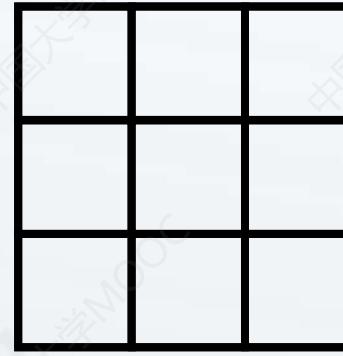


膨胀运算处理结果

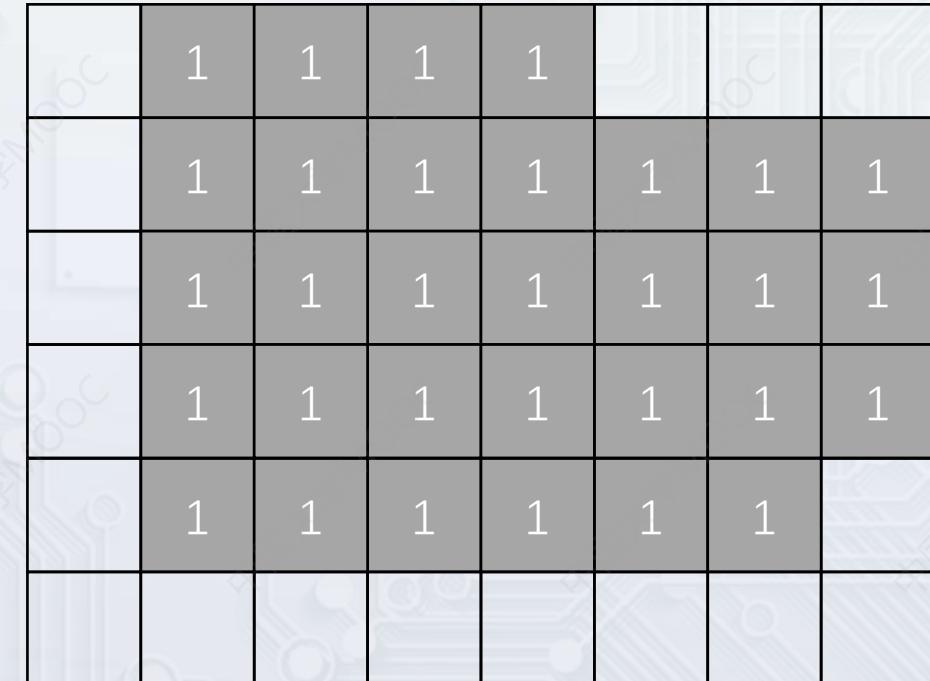
# 膨胀运算



输入二值图像图像，值为1的像素为待处理前景像素集合

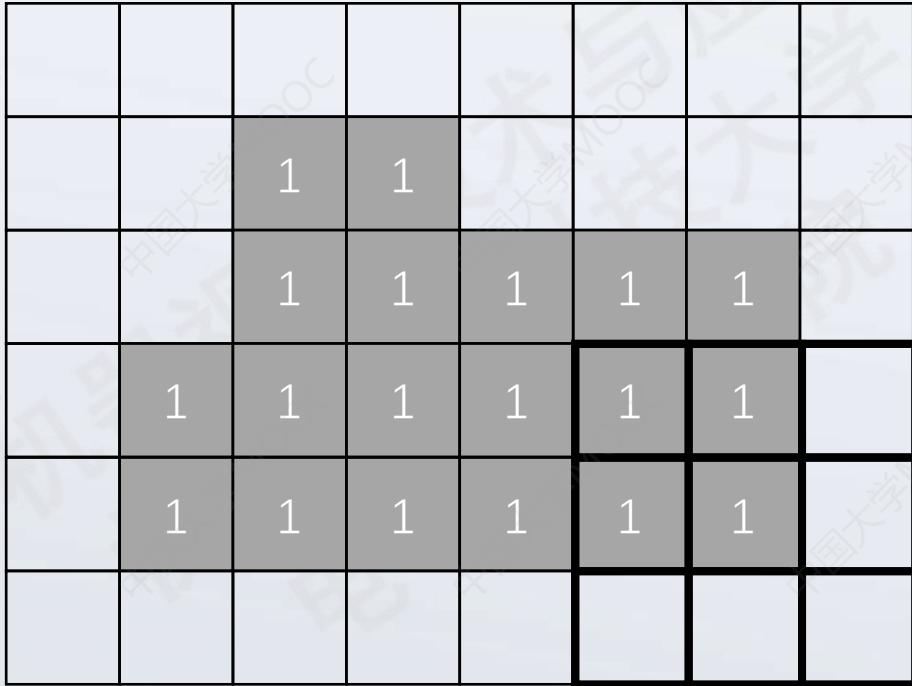


3×3结构元素，中心位置为锚点，使用结构元素遍历所有待处理像素，遍历时，锚点对齐待处理像素。

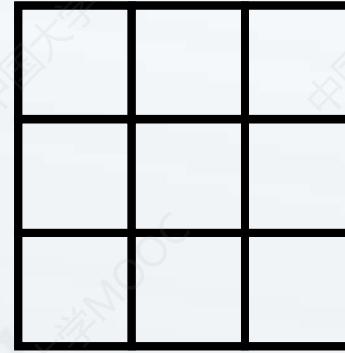


膨胀运算处理结果

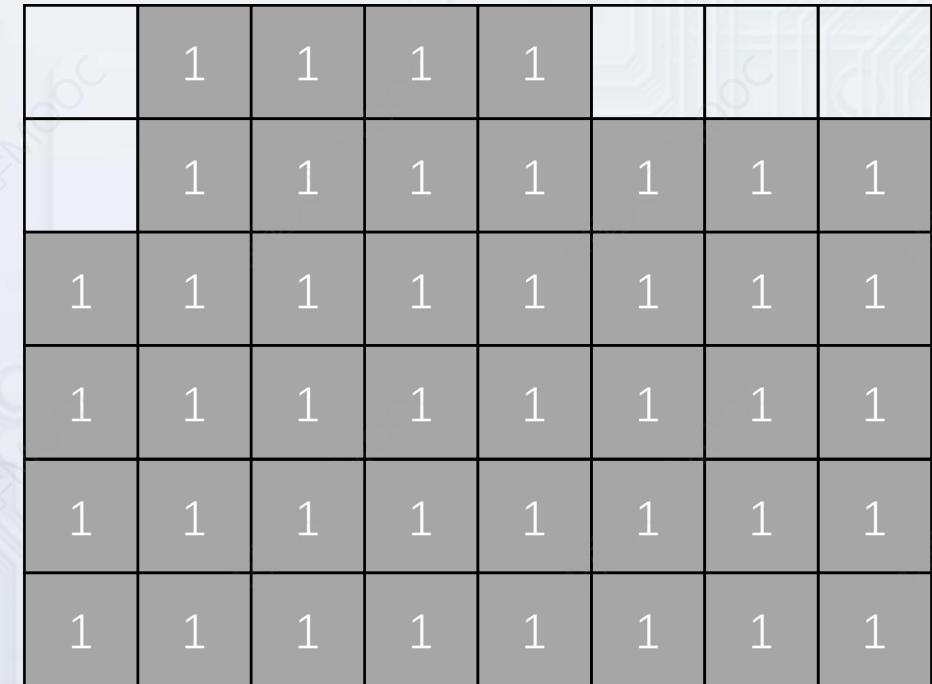
# 膨胀运算



输入二值图像图像，值为1的像素为待处理前景像素集合

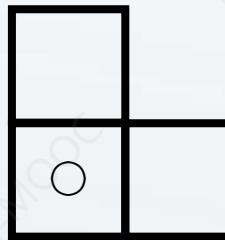


3×3结构元素，中心位置为锚点，使用结构元素遍历所有待处理像素，遍历时，锚点对齐待处理像素。



膨胀运算处理结果

# 膨胀运算



结构元素可以为任意形状，试用左图的结构算对输入图像做膨胀运算。标有○的位置为锚点。

			1	1			
			1	1	1	1	1
	1	1	1	1	1	1	
1	1	1	1	1	1	1	

输入二值图像图像，值为1的像素为待处理前景像素集合

# 膨胀运算

作用：如图，由于无法实现理想的二值化，使得原本连通的像素集合被分成不同的连通域，从而影响目标物的提取。可通过膨胀运算使其恢复连通性。

Historically, certain computer programs were written using only two digits rather than four to define the applicable year. Accordingly, the company's software may recognize a date using "00" as 1900 rather than the year 2000.



Historically, certain computer programs were written using only two digits rather than four to define the applicable year. Accordingly, the company's software may recognize a date using "00" as 1900 rather than the year 2000.



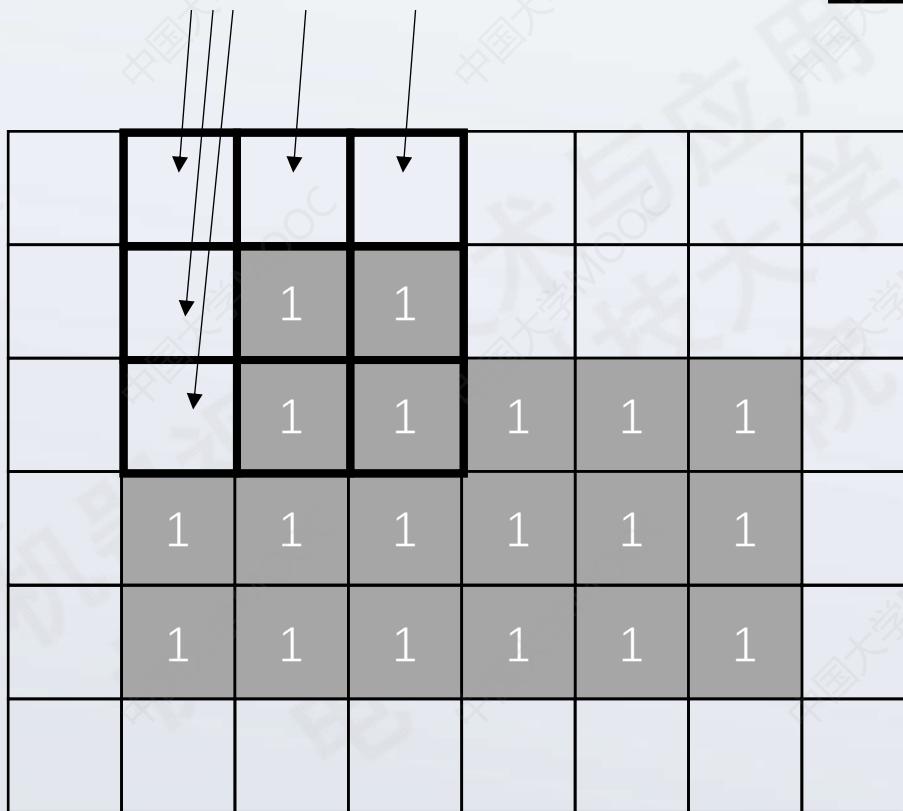
# 膨胀运算

作用：如图，由于无法实现理想的二值化，使得原本连通的像素集合被分成不同的连通域，从而影响目标物的提取。可通过膨胀运算使其恢复连通性。

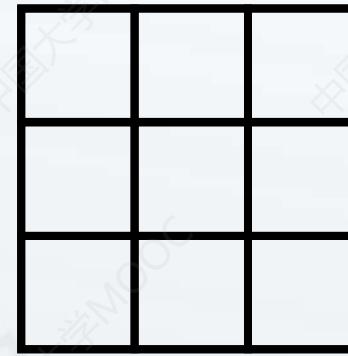


# 腐蚀运算

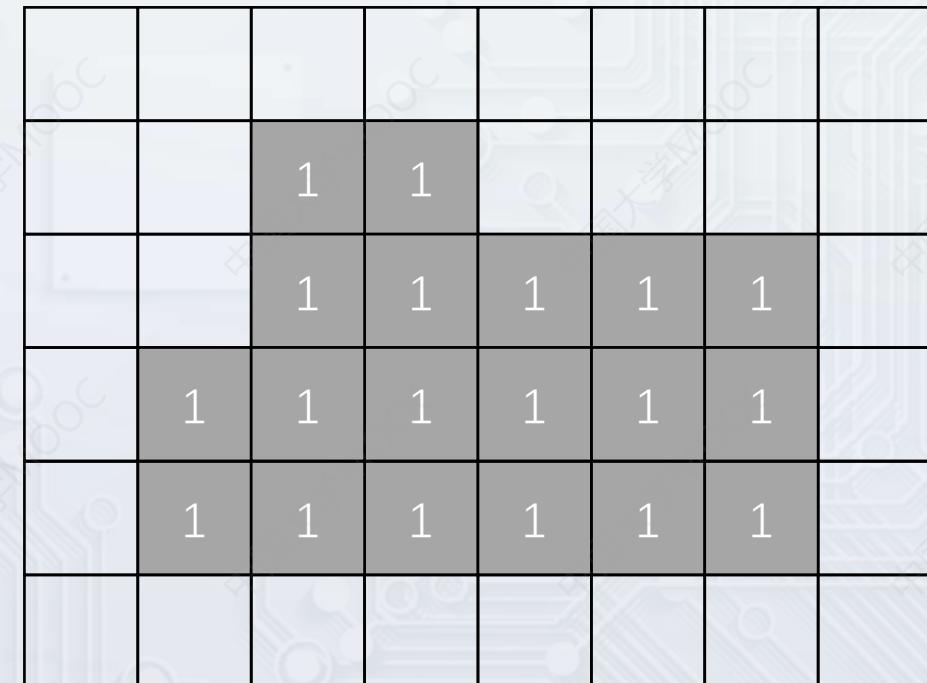
结构元素覆盖的范围如果出现0，则被处理的像素置0。



输入二值图像图像，值为1的像素为待处理前景像素集合



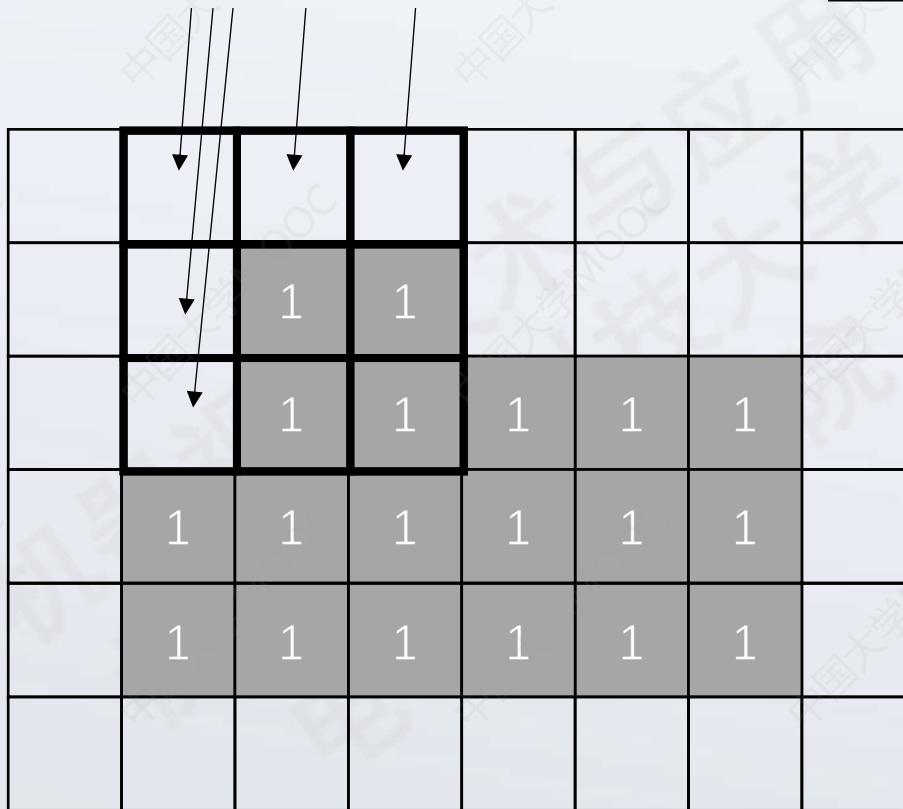
3×3结构元素，中心位置为锚点，使用结构元素遍历所有待处理像素。



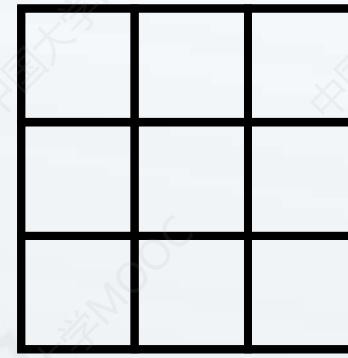
腐蚀运算处理结果

# 腐蚀运算

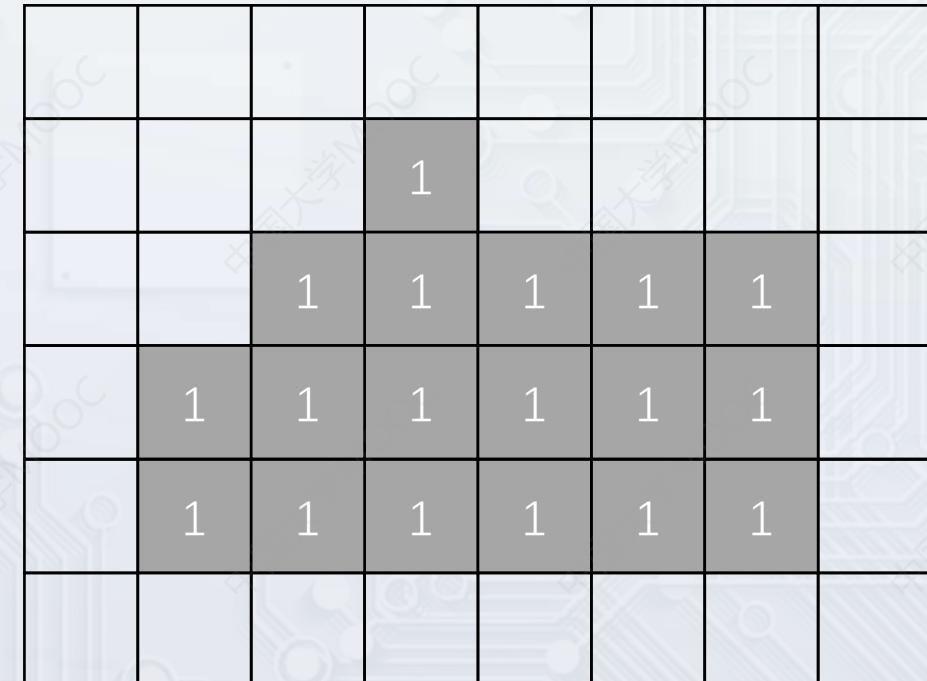
结构元素覆盖的范围如果出现0，则被处理的像素置0。



输入二值图像图像，值为1的像素为待处理前景像素集合

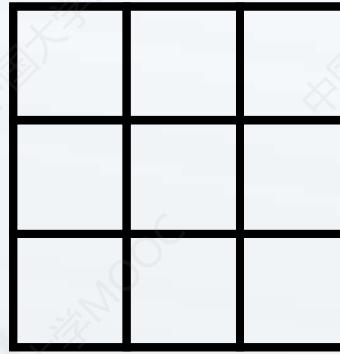


3×3结构元素，中心位置为锚点，使用结构元素遍历所有待处理像素。

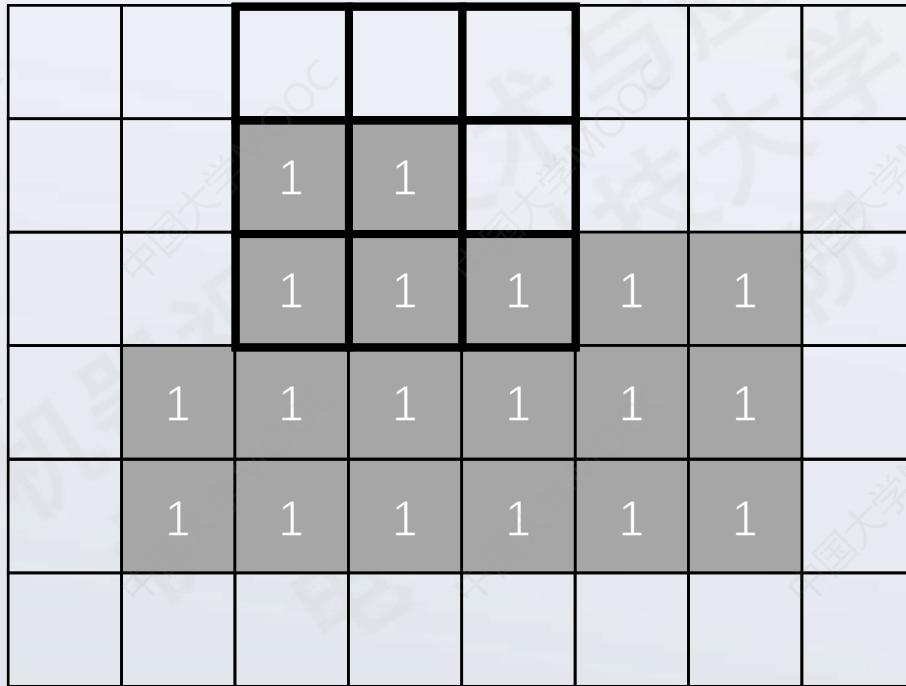


腐蚀运算处理结果

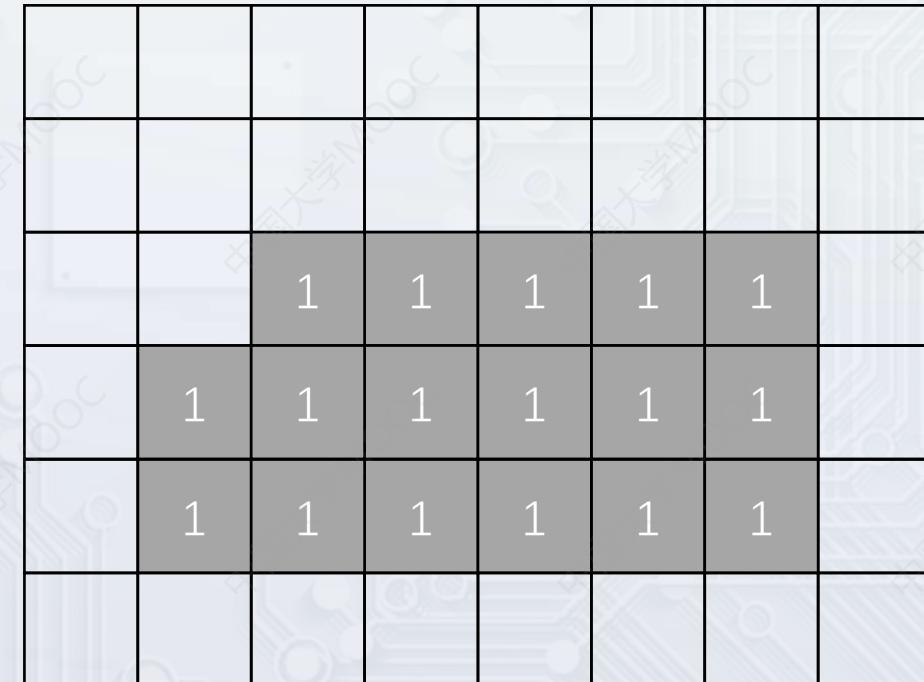
# 腐蚀运算



3×3结构元素，中心位置为锚点，使用结构元素遍历所有待处理像素。

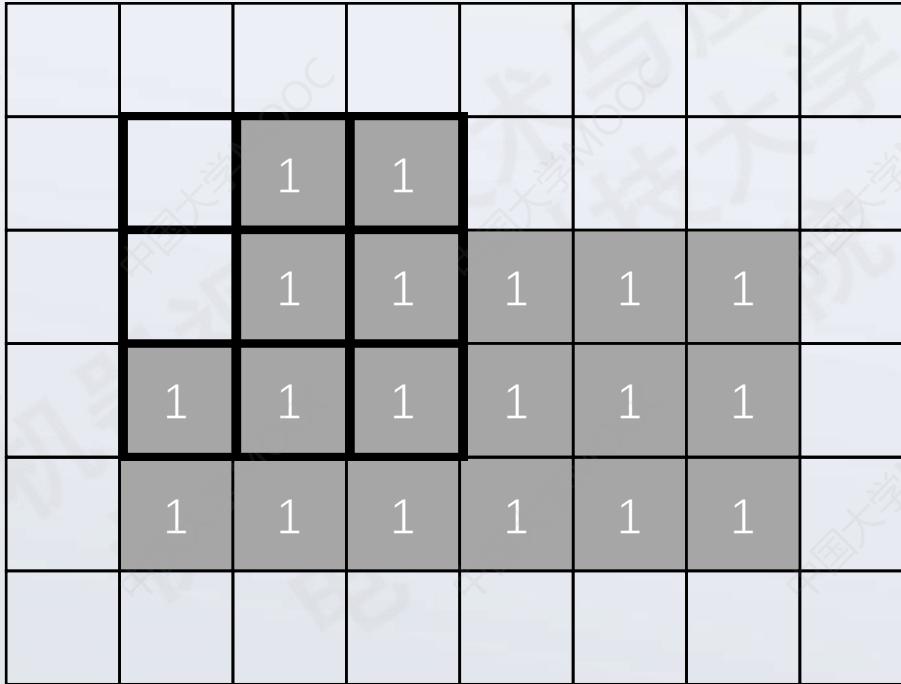


输入二值图像图像，值为1的像素为待处理前景像素集合

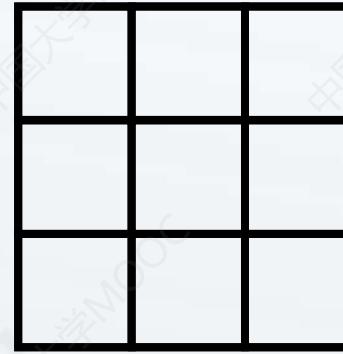


腐蚀运算处理结果

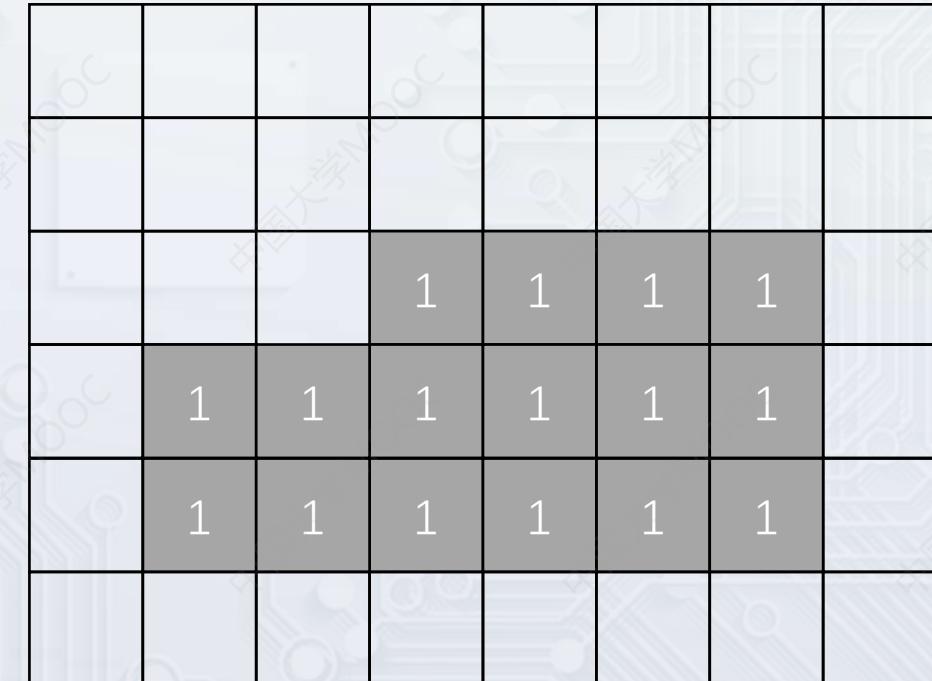
# 腐蚀运算



输入二值图像图像，值为1的像素为待处理前景像素集合

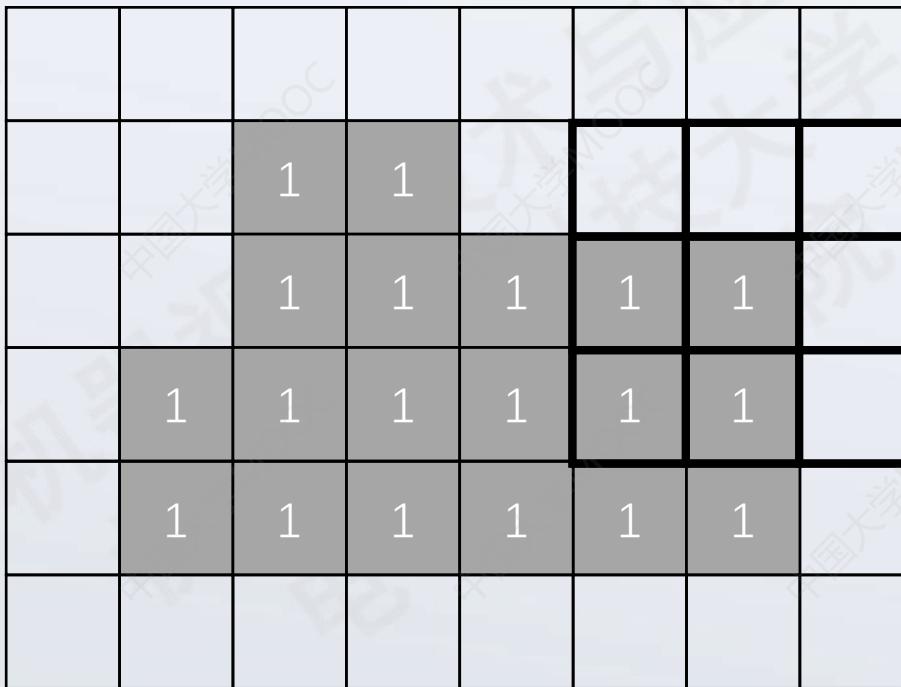


3×3结构元素，中心位置为锚点，使用结构元素遍历所有待处理像素。

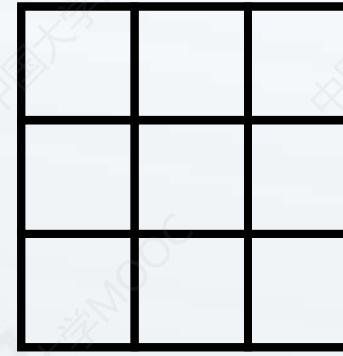


腐蚀运算处理结果

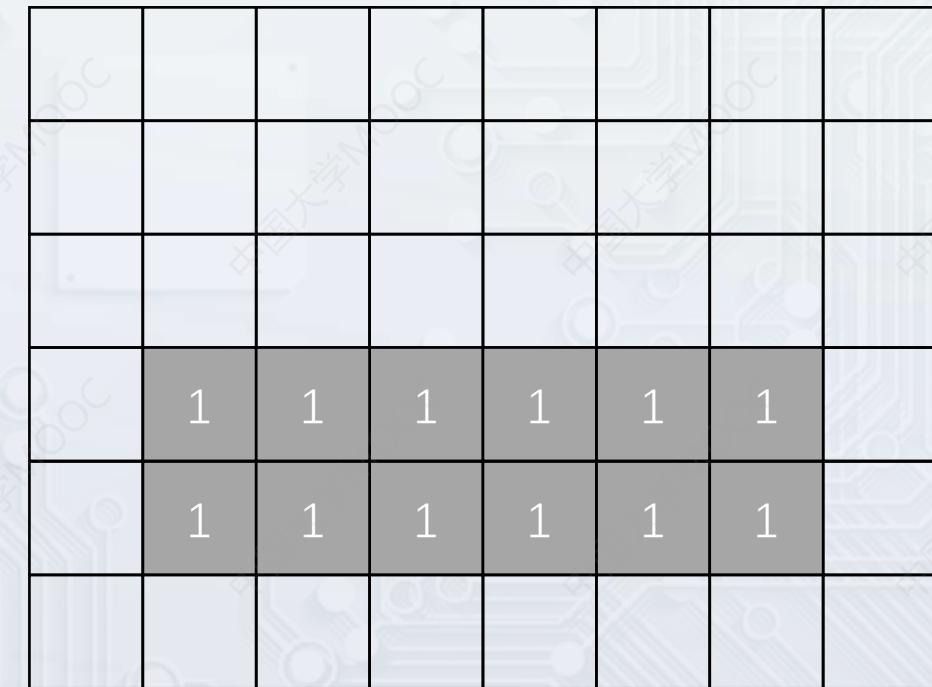
# 腐蚀运算



输入二值图像图像，值为1的像素为待处理前景像素集合

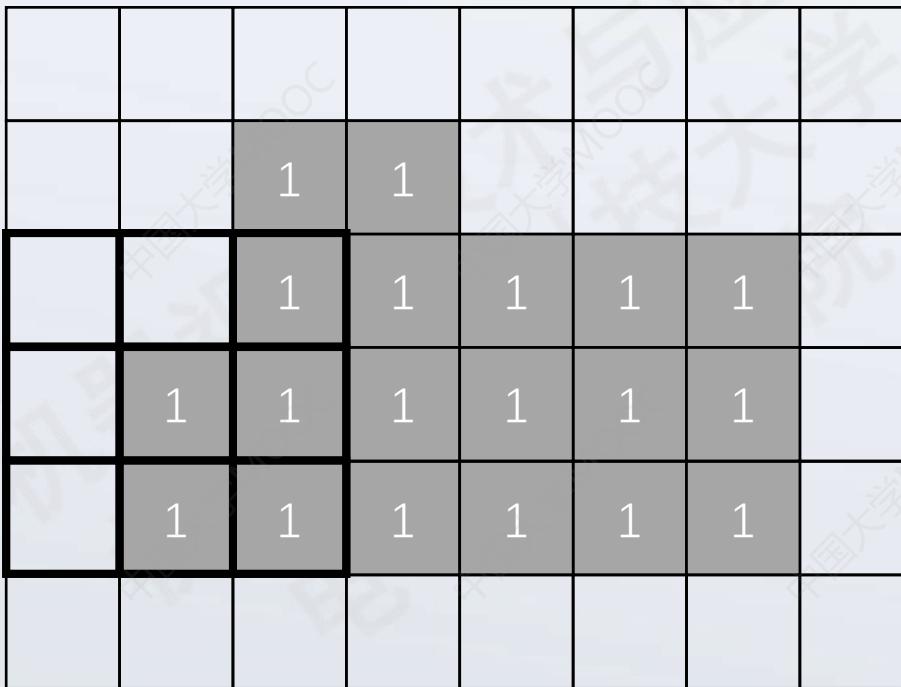


3×3结构元素，中心位置为锚点，使用结构元素遍历所有待处理像素。

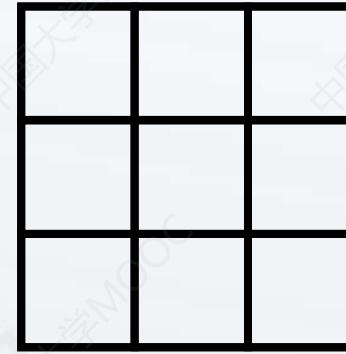


腐蚀运算处理结果

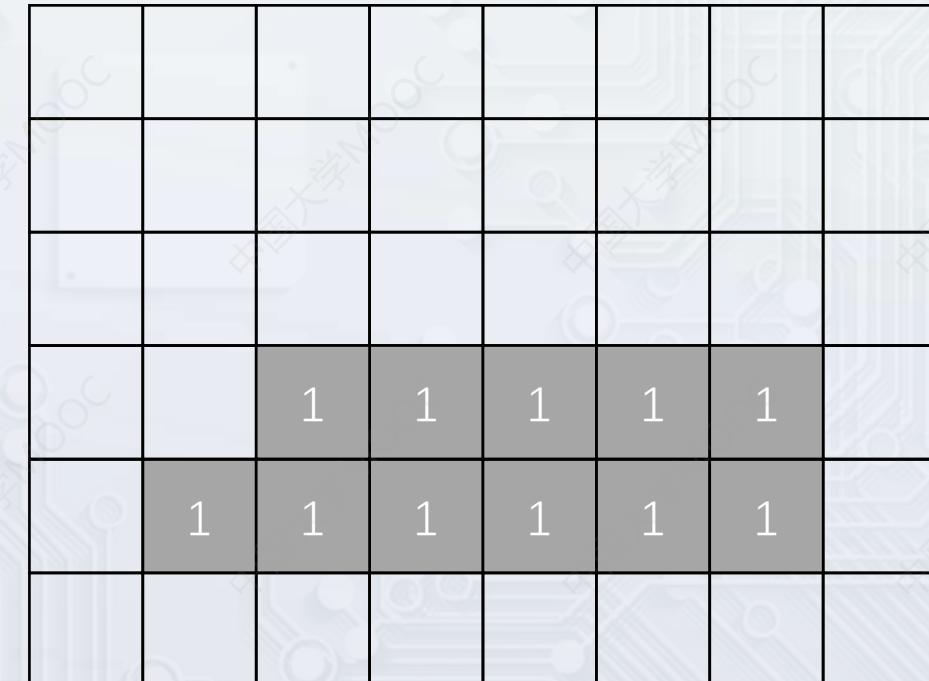
# 腐蚀运算



输入二值图像图像，值为1的像素为待处理前景像素集合



3×3结构元素，中心位置为锚点，使用结构元素遍历所有待处理像素。

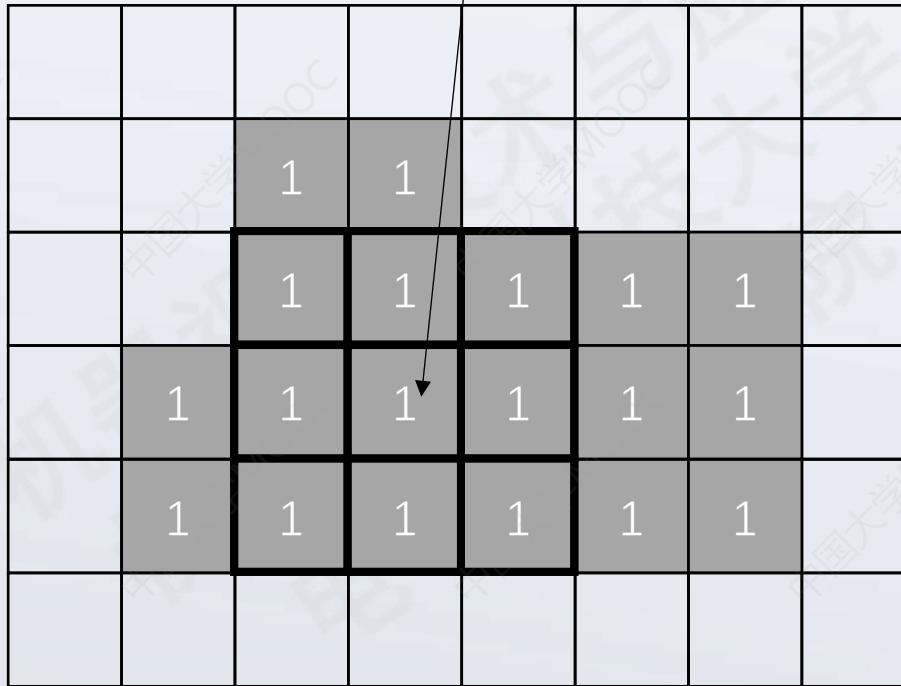


腐蚀运算处理结果

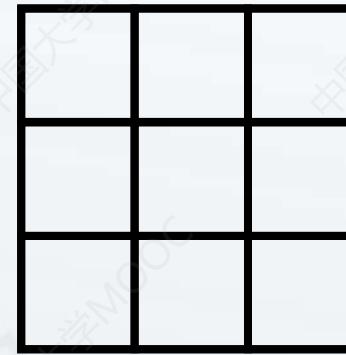
# 腐蚀运算

结构元素覆盖的范围没有0出现，则被处理的像素置保留。

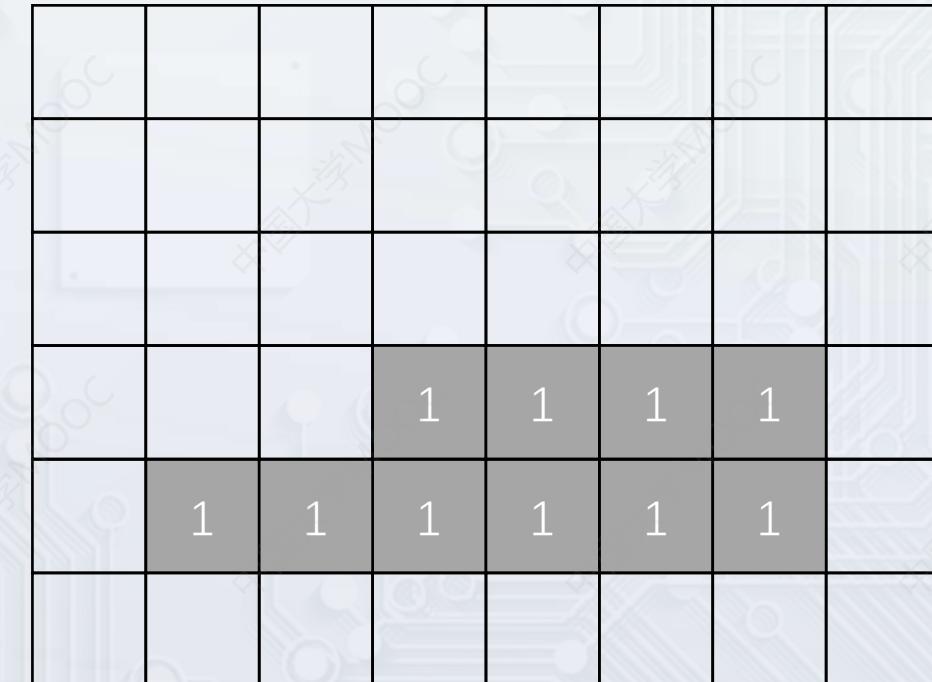
注意：是和原图比较而非腐蚀过的结果比较。



输入二值图像图像，值为1的像素为待处理前景像素集合



3×3结构元素，中心位置为锚点，使用结构元素遍历所有待处理像素。

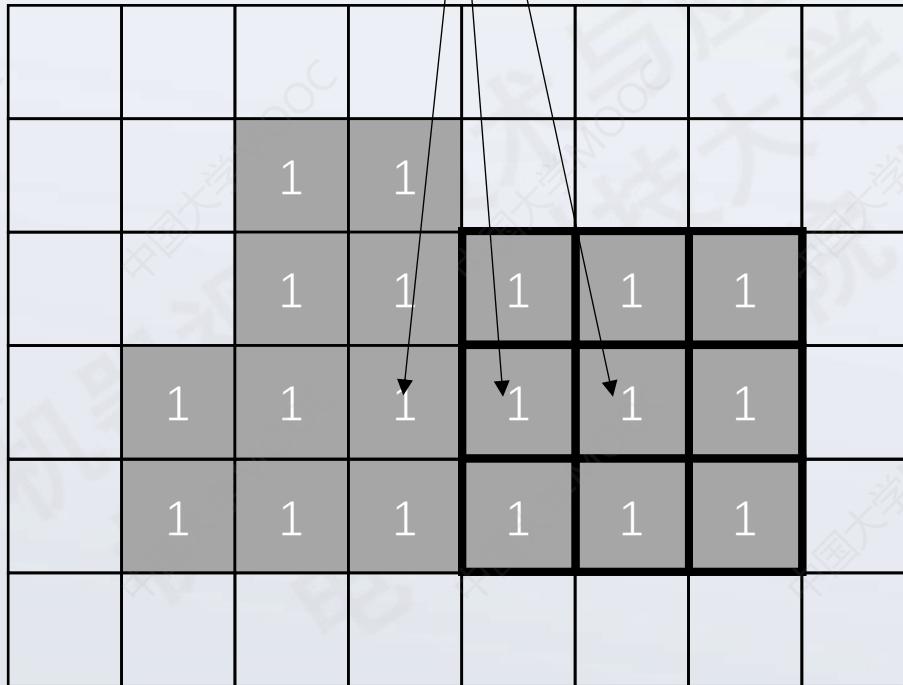


腐蚀运算处理结果

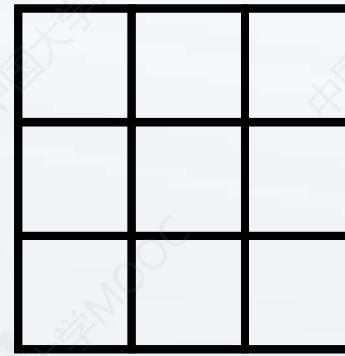
# 腐蚀运算

结构元素覆盖的范围没有0出现，则被处理的像素置保留。

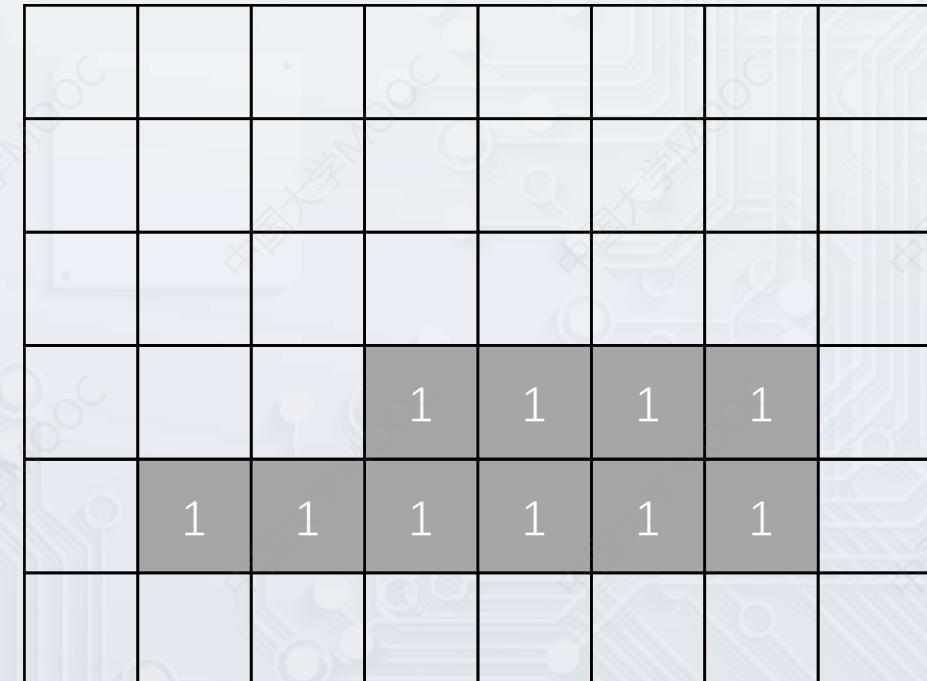
注意：是和原图比较而非腐蚀过的结果比较。



输入二值图像图像，值为1的像素为待处理前景像素集合

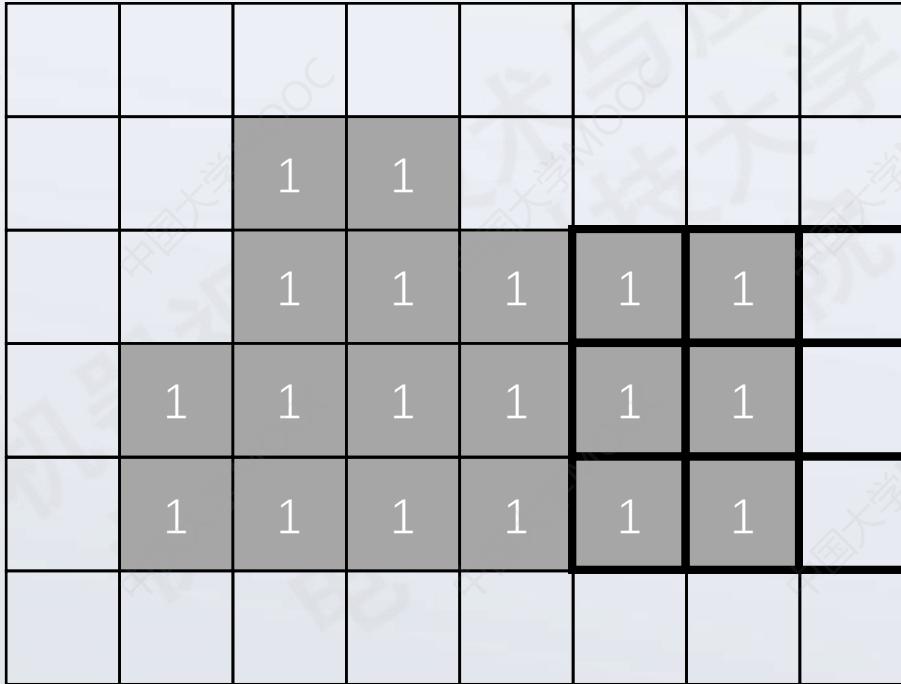


3×3结构元素，中心位置为锚点，使用结构元素遍历所有待处理像素。

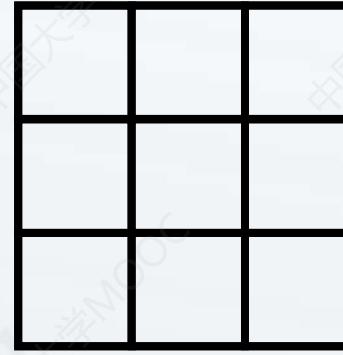


腐蚀运算处理结果

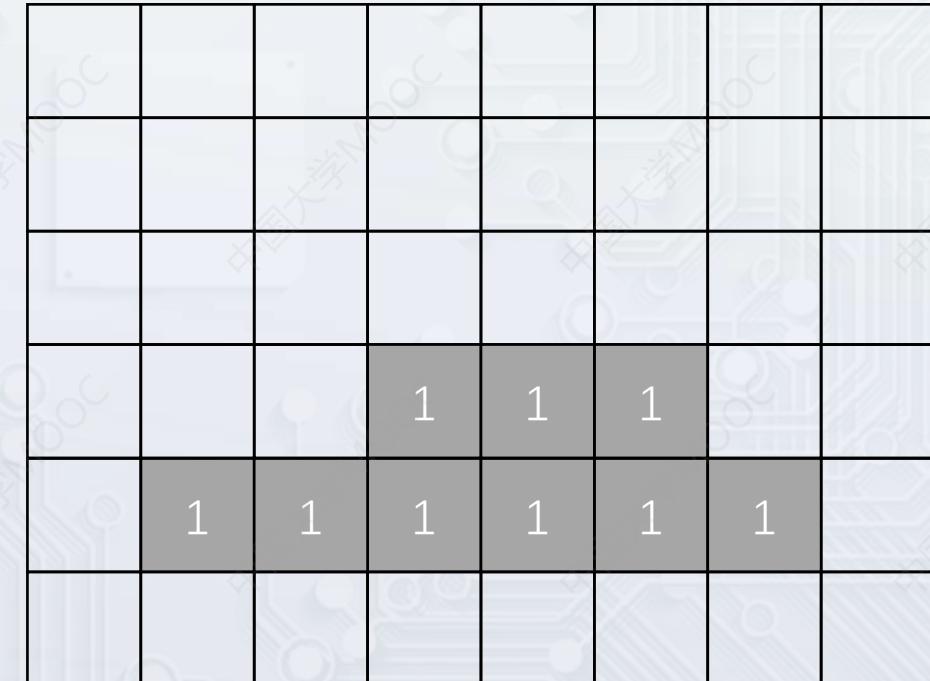
# 腐蚀运算



输入二值图像图像，值为1的像素为待处理前景像素集合

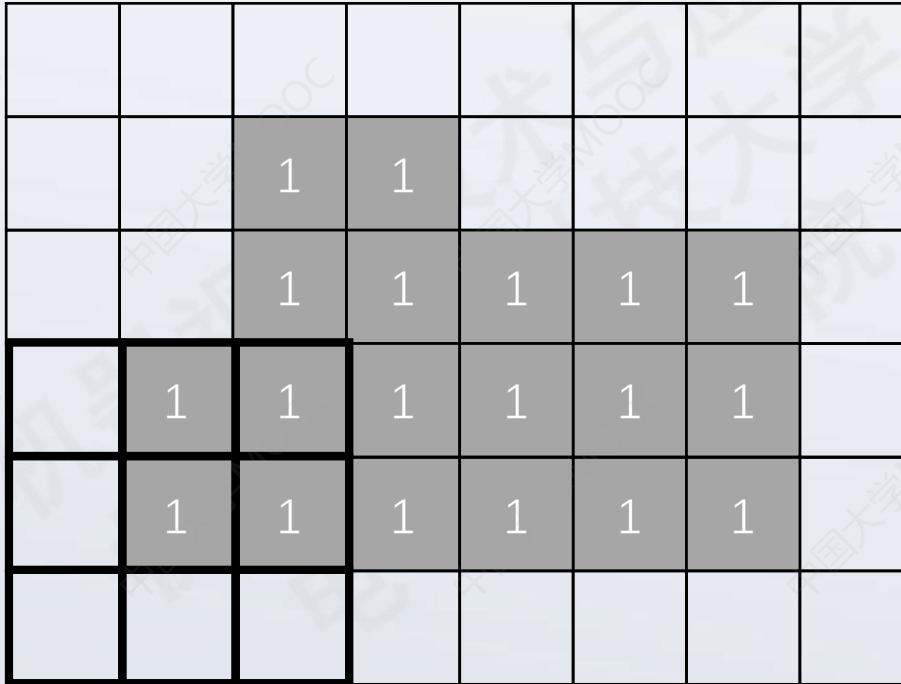


3×3结构元素，中心位置为锚点，使用结构元素遍历所有待处理像素。

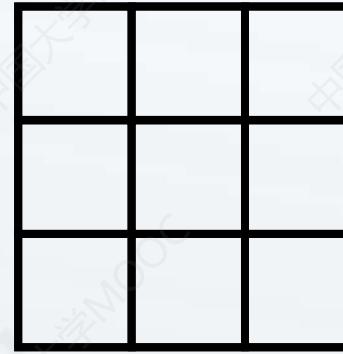


腐蚀运算处理结果

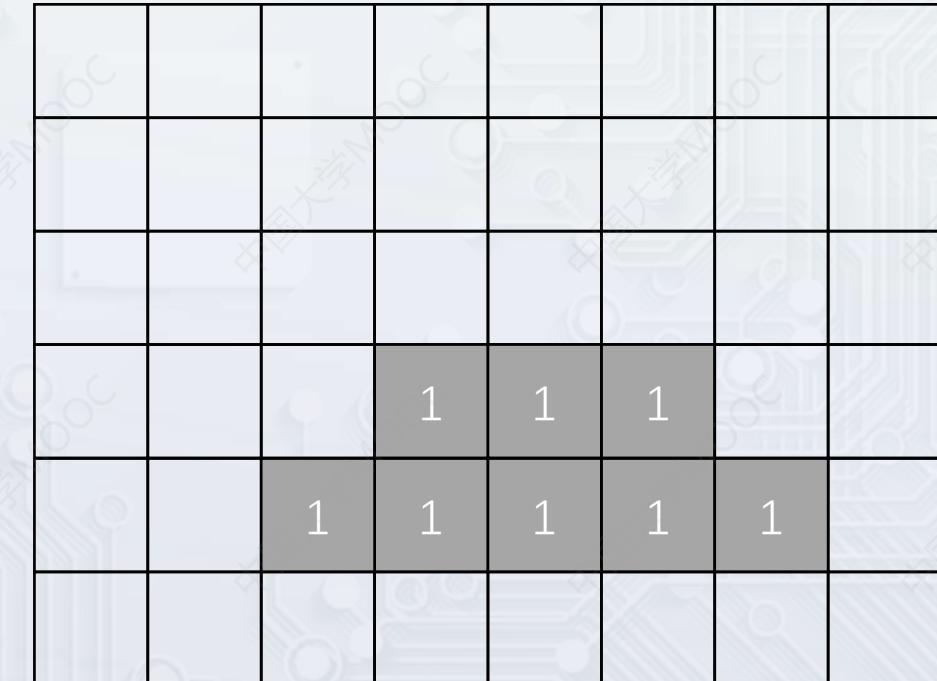
# 腐蚀运算



输入二值图像图像，值为1的像素为待处理前景像素集合

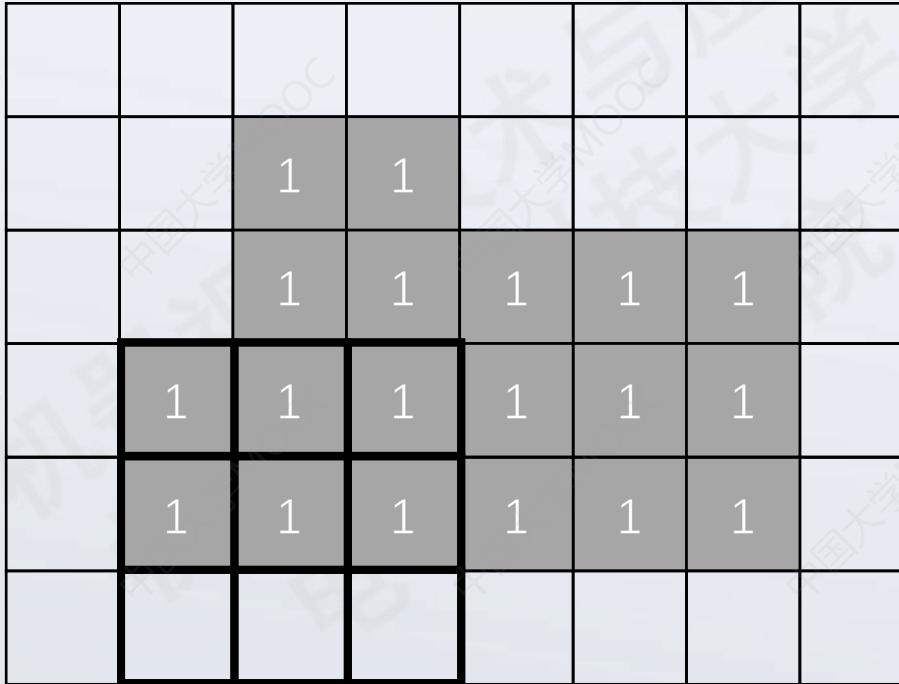


3×3结构元素，中心位置为锚点，使用结构元素遍历所有待处理像素。

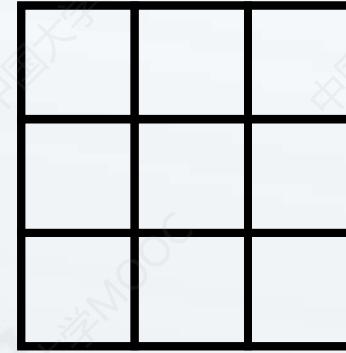


腐蚀运算处理结果

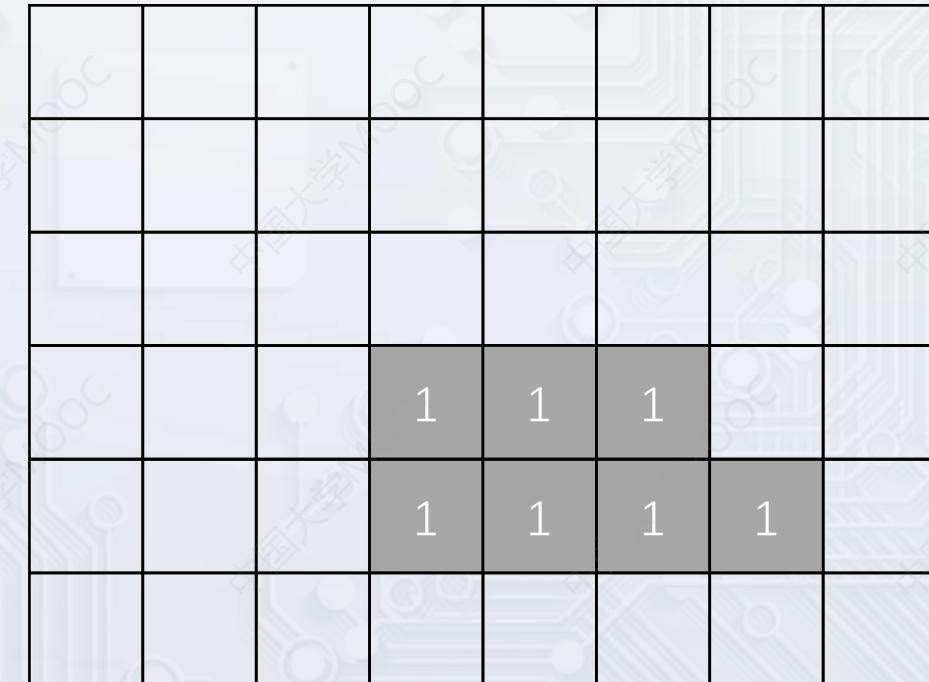
# 腐蚀运算



输入二值图像图像，值为1的像素为待处理前景像素集合

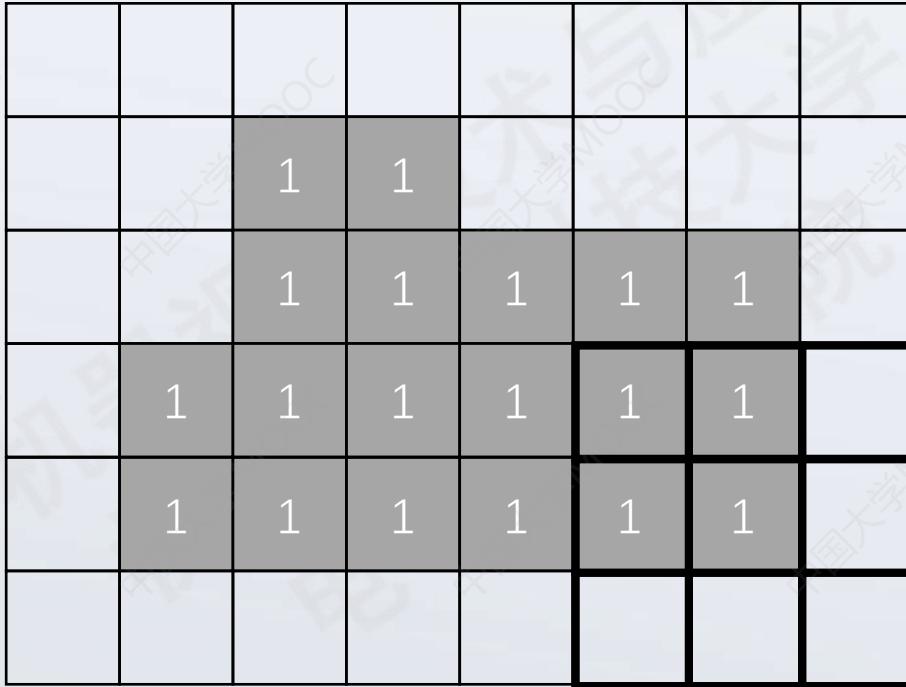


3×3结构元素，中心位置为锚点，使用结构元素遍历所有待处理像素。

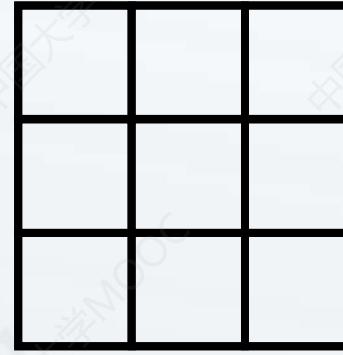


腐蚀运算处理结果

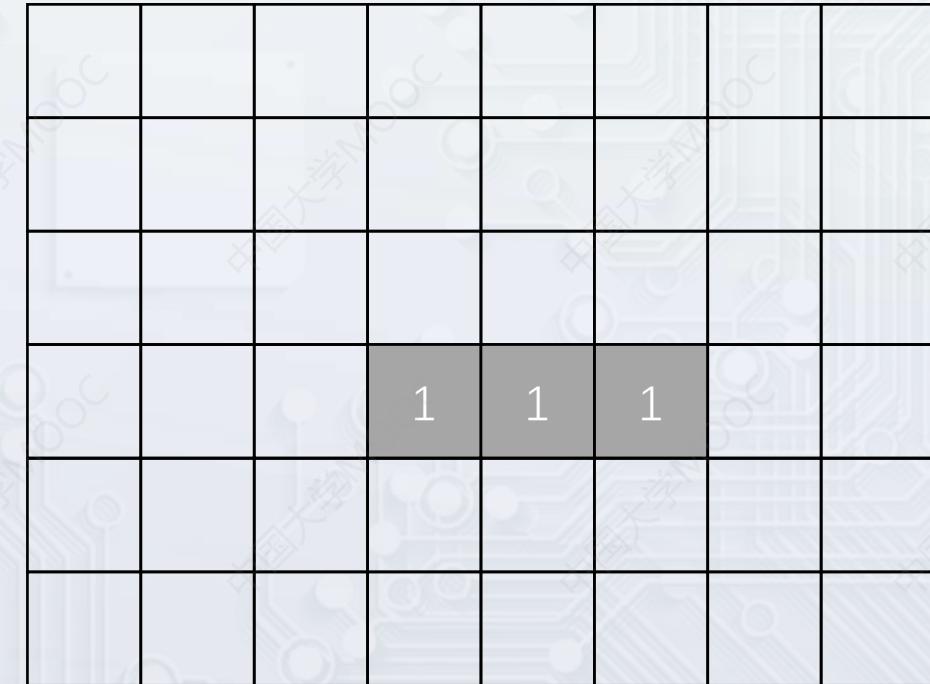
# 腐蚀运算



输入二值图像图像，值为1的像素为待处理前景像素集合

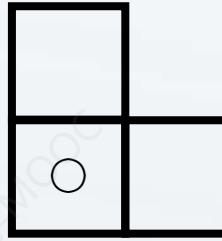


3×3结构元素，中心位置为锚点，使用结构元素遍历所有待处理像素。



腐蚀运算处理结果

# 腐蚀运算



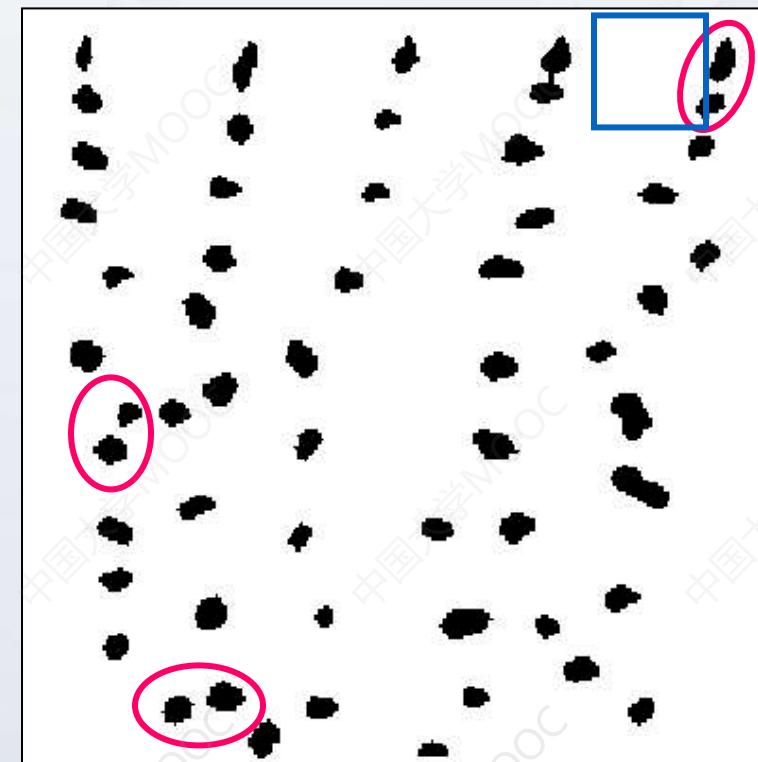
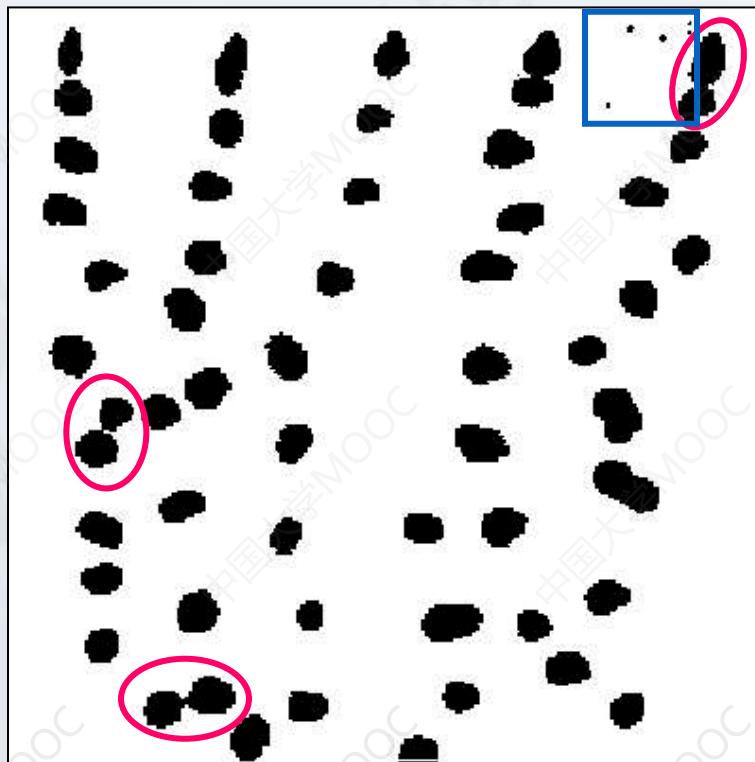
结构元素可以为任意形状，试用左图的结构算对输入图像做腐蚀运算。标有○的位置为锚点。

		1	1				
		1	1	1	1	1	
	1	1	1	1	1	1	
	1	1	1	1	1	1	

输入二值图像图像，值为1的像素为待处理前景像素集合

# 腐蚀运算

作用：如图，去除一些黏连像素。以及去除噪声。



# 膨胀和腐蚀运算的问题？？

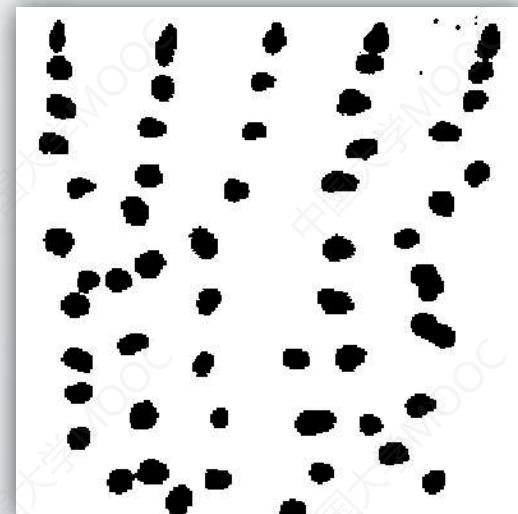
原图



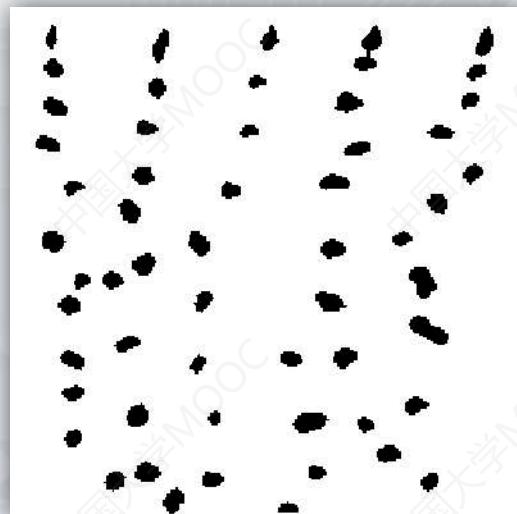
膨胀处理



原图



腐蚀处理



# 膨胀和腐蚀运算的问题？？

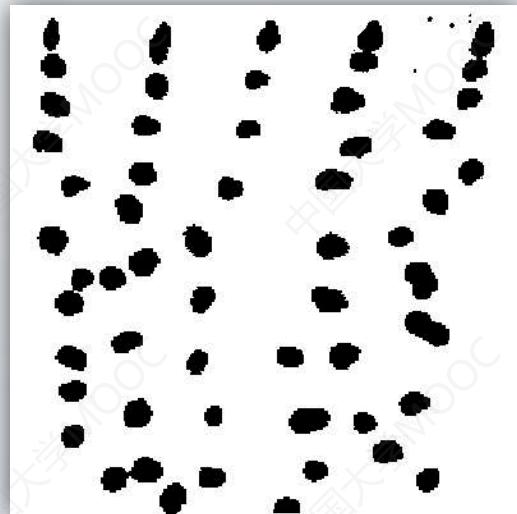
原图



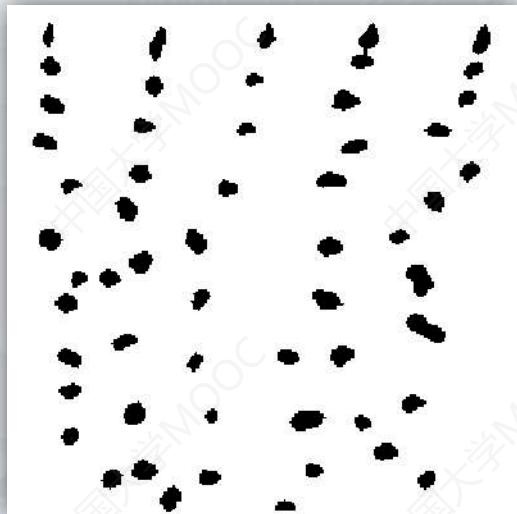
膨胀处理



原图



腐蚀处理



目标物体变形，对识别时的特征提取会造成影响

# 开运算

开运算：对图像先进行 $3 \times 3$ 腐蚀，  
再对腐蚀结果进行 $3 \times 3$ 膨胀。

	1	1	1					
	1	1	1		1	1	1	
	1	1	1		1	1	1	
	1	1	1	1	1	1	1	

原图

	1	1	1					
	1	1	1		1	1	1	
	1	1	1		1	1	1	
	1	1	1	1	1	1	1	

腐蚀结果

# 开运算

开运算：对图像先进行 $3 \times 3$ 腐蚀，  
再对腐蚀结果进行 $3 \times 3$ 膨胀。

	1	1	1					
	1	1	1		1	1	1	
	1	1	1		1	1	1	
	1	1	1	1	1	1	1	

原图



				1				
				1				1

腐蚀结果

# 开运算

开运算：对图像先进行 $3 \times 3$ 腐蚀，再对腐蚀结果进行 $3 \times 3$ 膨胀。

先腐蚀再膨胀的结果并不是恢复原状，而是会消除黏连部分，同时不影响其他部分的形状

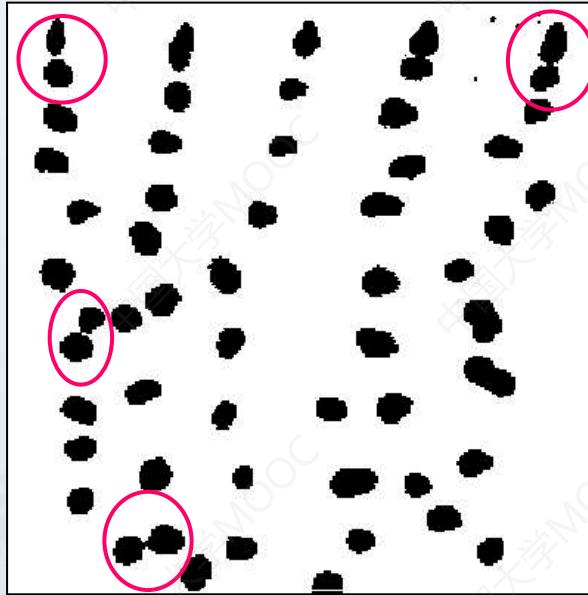
	1	1	1					
	1	1	1			1	1	1
	1	1	1		1	1	1	
	1	1	1		1	1	1	

腐蚀结果

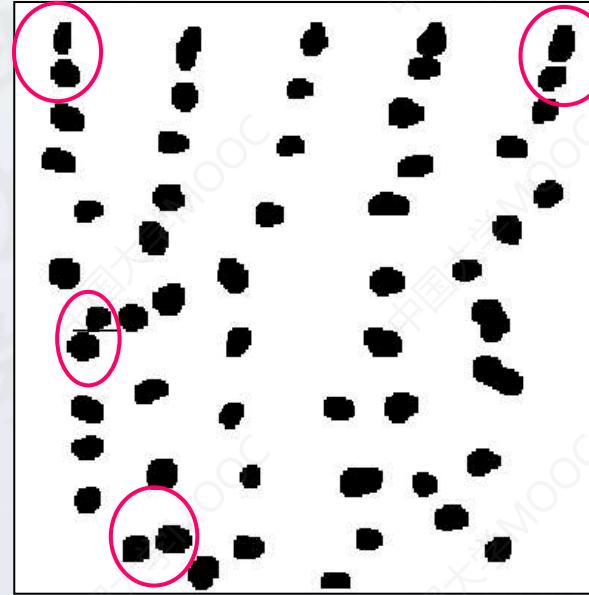
			1					
			1					
			1					

对腐蚀结果进行膨胀的结果

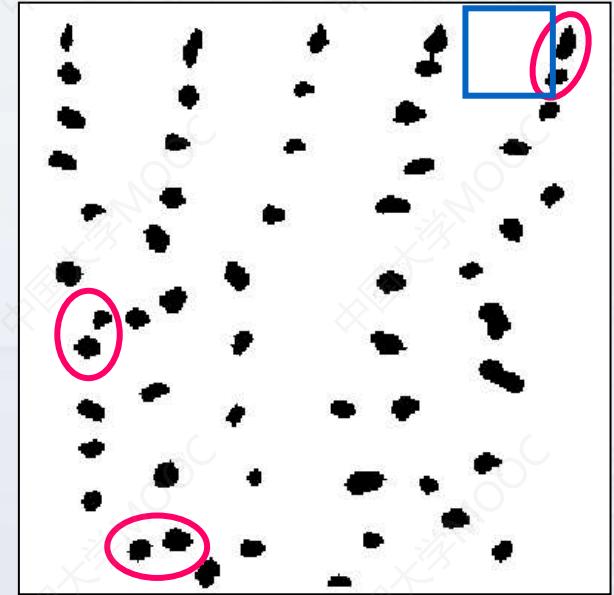
# 开运算



原图



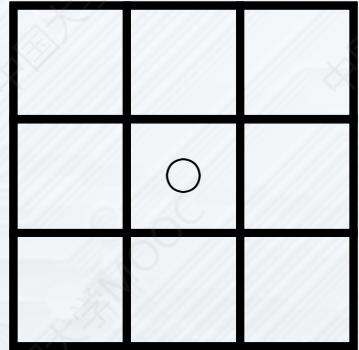
开运算



腐蚀

# 闭运算

闭运算：对图像先进行膨胀，再对膨胀结果进行腐蚀。



3×3结构元素

1	1	1						
1	1	1		1	1			
1	1	1		1	1			
1	1	1	1	1	1			

原图

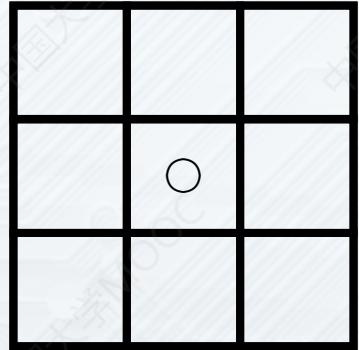


1	1	1	1	1				
1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1

膨胀结果

# 闭运算

闭运算：对图像先进行膨胀，再对腐蚀结果进行腐蚀。



3×3结构元素

1	1	1						
1	1	1	1	1	1			
1	1	1	1	1	1			
1	1	1	1	1	1			

腐蚀结果



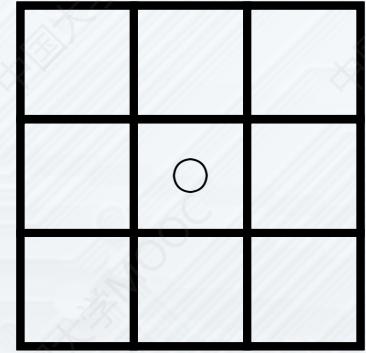
1	1	1	1	1				
1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1

膨胀结果

# 闭运算

闭运算：对图像先进行膨胀，再对腐蚀

结果进行腐蚀。



3×3结构元素

小的裂缝，小孔等被填充，并不影响原来的形状

1	1	1						
1	1	1		1	1	1		
1	1	1		1	1	1		
1	1	1	1	1	1	1		

原图

1	1	1	1					
1	1	1	1	1	1	1		
1	1	1	1	1	1	1		
1	1	1	1	1	1	1		

对膨胀结果进行腐蚀处理的结果

# 闭运算



原图



闭运算



dilation

# 灰度图的形态学处理

3×3结构元素

最小值

209	125	191	9	168	246	158	14	
232	205	101	113	42	141	122	136	
33	37	168	98	31	36	91	200	
234	108	44	196	128	39	213	240	
162	235	181	204	246	66	150	34	
25	203	9	48	88	216	141	146	
72	246	71	126	150	66	235	121	

灰度图的腐蚀运算：遍历像素结构元素锚点和待处理像素对齐，该像素处理后的值等于结构元素范围内的**最小值**。

125	101	9	9	9	42	14	14
33	33	9	9	9	31	14	14
33	33	37	31	31	31	36	91
33	33	37	31	31	31	34	34
25	9	9	9	39	39	34	34
25	9	9	9	48	66	34	34
25	9	9	9	48	66	66	121

# 灰度图的形态学处理

3×3结构元素

最大值

3×3结构元素			最大值					
			209	125	191	9	168	246
	232	205	101	113	42	141	122	136
33	37	168	98	31	36	91	200	
234	108	44	196	128	39	213	240	
162	235	181	204	246	66	150	34	
25	203	9	48	88	216	141	146	
72	246	71	126	150	66	235	121	

灰度图的膨胀运算：遍历像素结构元素锚点和待处理像素对齐，该像素处理后的值等于结构元素范围内的

最大值。

The diagram illustrates the dilation operation. On the left, a 3x3 kernel is shown with its center element highlighted. An arrow points from this kernel to a specific pixel in the 3x9 input matrix below. The input matrix contains various gray values. The dilation process involves applying the kernel's maximum value (246) to the target pixel, resulting in a value of 246 in the output matrix. The output matrix is shown on the right, where the target pixel has been updated.

232	232	205	191	246	246	246	158
232	232	205	191	246	246	246	200
234	234	235	246	246	246	240	240
235	235	235	246	246	246	240	240
235	235	235	246	246	246	240	240
246	246	246	246	246	246	235	235
246	246	246	150	216	235	235	235

# 顶帽和底帽变换

顶帽变换：原图 – 灰度开运算结果（灰度腐蚀+灰度膨胀）

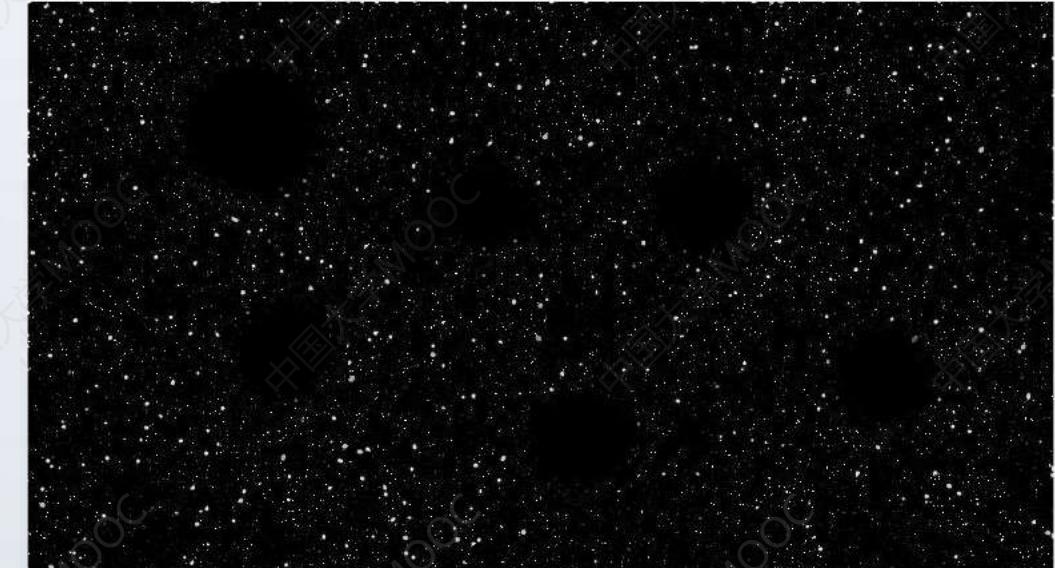
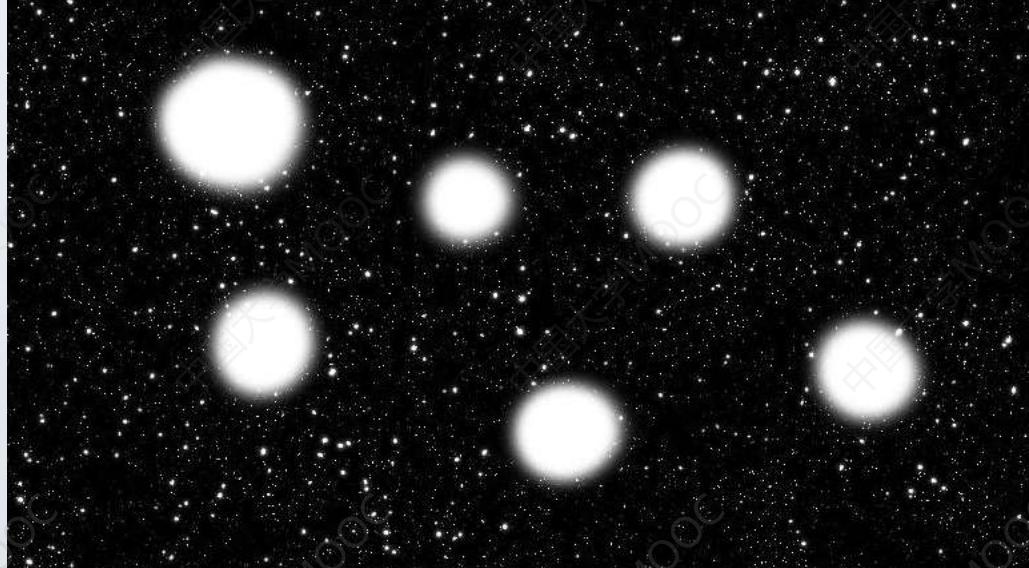
- 保留比结构元素小的部分
- 保留比周围环境亮的像素

底帽变换：灰度闭运算结果（灰度膨胀+灰度腐蚀）– 原图

- 保留比结构元素小的部分
- 保留比周围环境暗的像素

# 顶帽和底帽变换

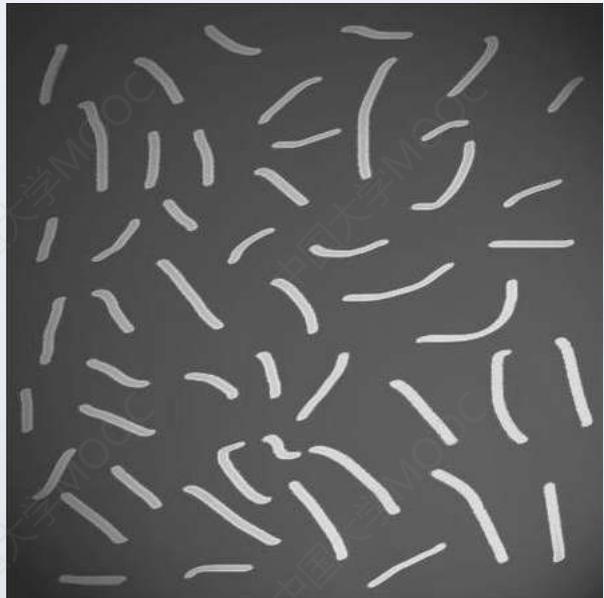
如下图，假设我们希望提取图中那些小的白色像素，如使用腐蚀处理将不可能在保留小的图像的同时去除大的图像。我们可以选择面积大小介于6个最大圆和小的白色颗粒之间的结构算子对其进行顶帽运算。



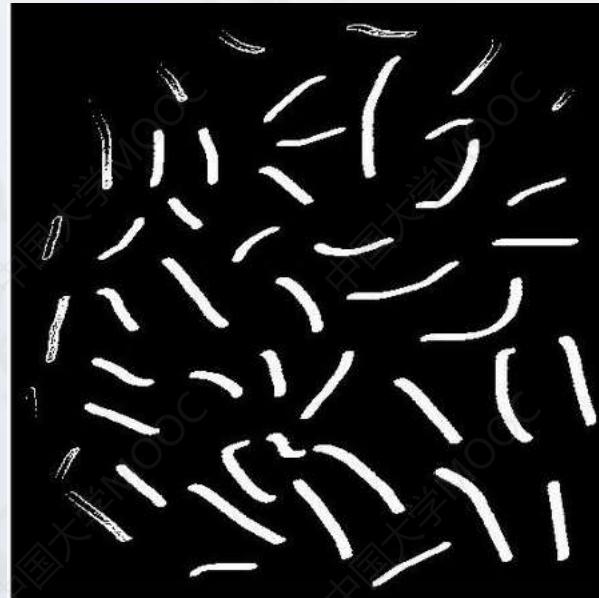
图来源：[https://en.wikipedia.org/wiki/Top-hat\\_transform](https://en.wikipedia.org/wiki/Top-hat_transform)

# 顶帽和底帽变换

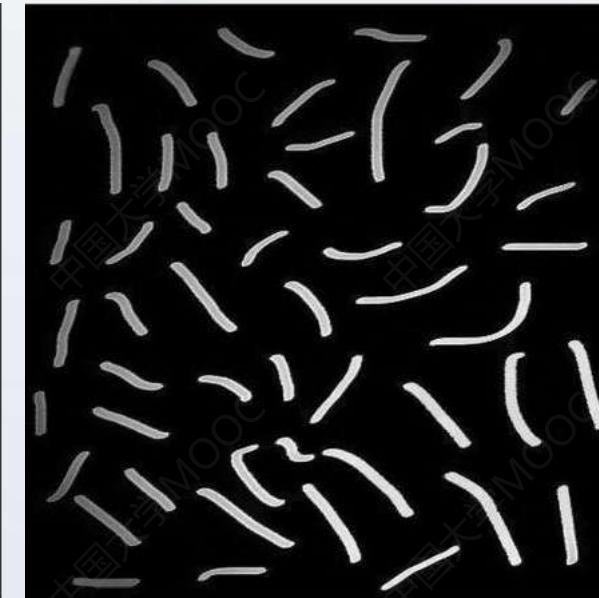
顶帽处理还可以消除背景光照不均匀的现象，从而改善在二值化时的效果。同样结构元素的尺寸大小要根据目标物体的大小进行选择。



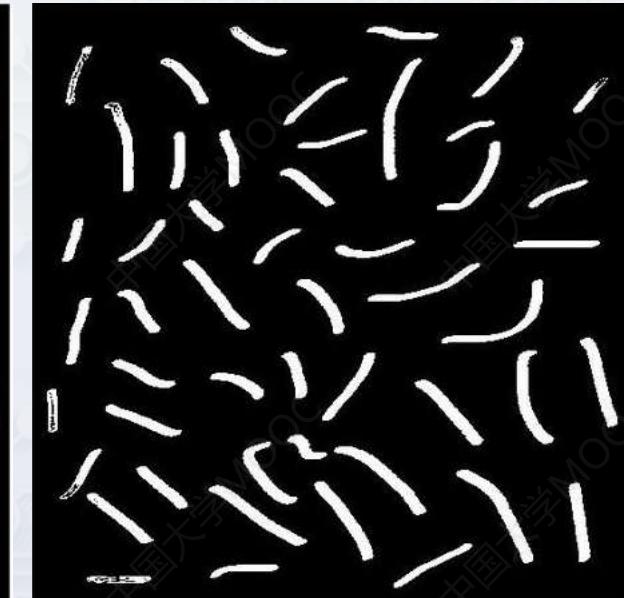
原图



原图二值化结果



顶帽处理结果



顶帽二值化结果

图来源：[https://en.wikipedia.org/wiki/Top-hat\\_transform](https://en.wikipedia.org/wiki/Top-hat_transform)

# 谢谢！

# 机器视觉技术与应用

## 5. 空间滤波

李竹

杭州电子科技大学

电子信息学院

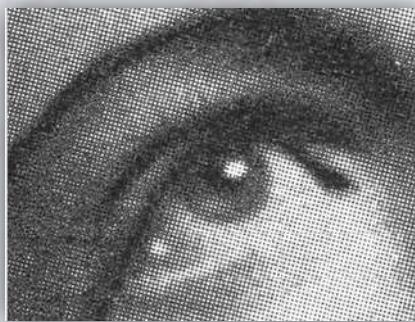


# 本章概要

1. 图像卷积的基本概念
2. 平滑滤波
3. 中值滤波
4. 边缘提取

# 卷积的基本概念

空间滤波是一种采用滤波处理的影像增强方法。其理论基础是空间卷积和空间相关。目的是改善影像质量，包括去除高频噪声与干扰，及影像边缘增强、线性增强以及去模糊等。分为低通滤波（平滑化）、高通滤波（锐化）和带通滤波。



降噪

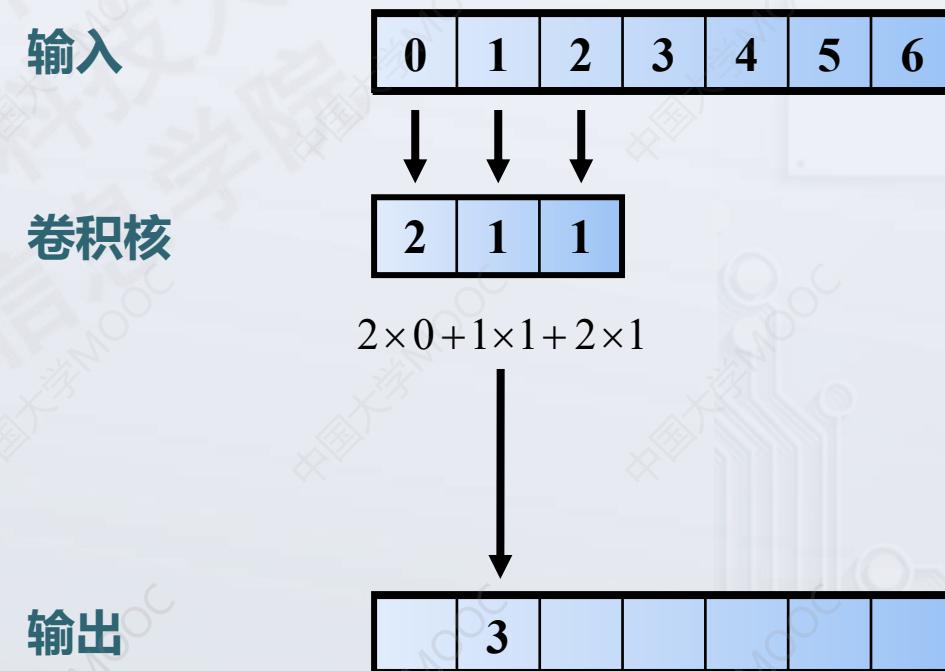


边缘提取



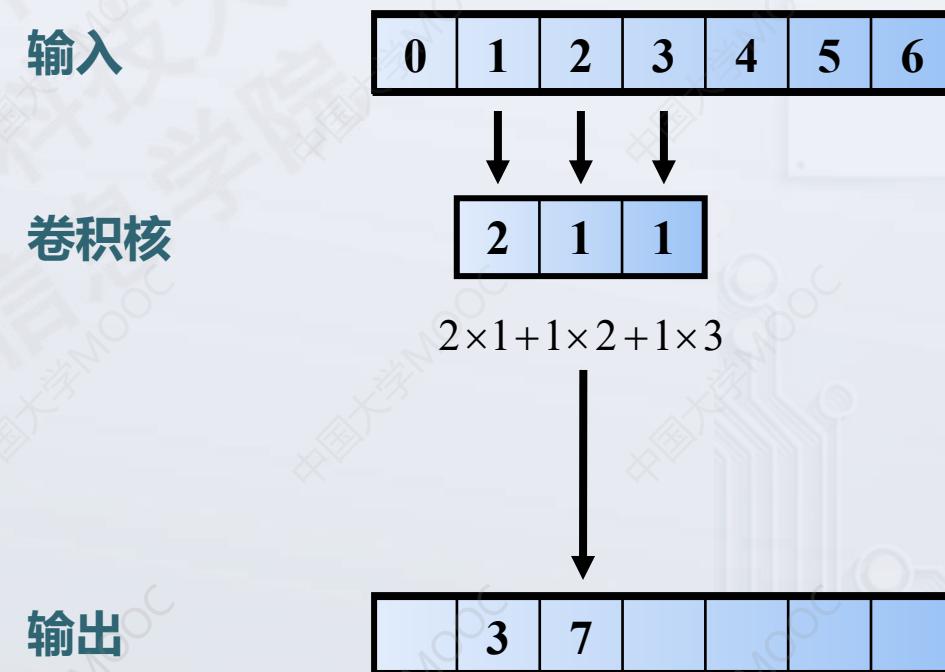
# 一维卷积实例

对数字图像做卷积操作其实就是利用卷积核在图像上滑动，将图像点上的像素灰度值与对应的卷积核上的数值相乘，然后将所有相乘后的值相加作为卷积核中间像素对应的图像上像素的灰度值，并最终滑动完所有图像的过程。



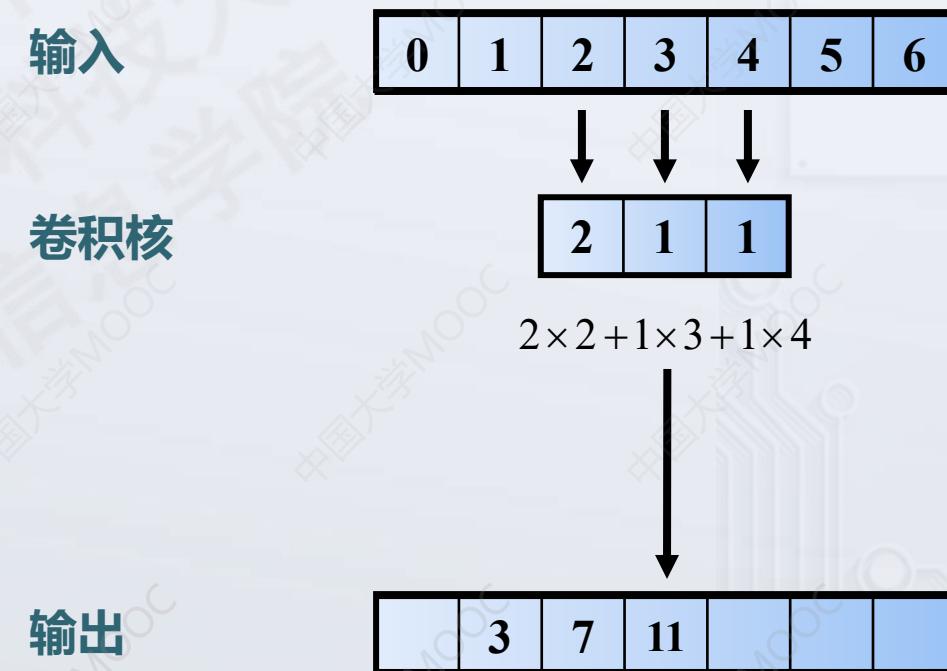
# 一维卷积实例

对数字图像做卷积操作其实就是利用卷积核在图像上滑动，将图像点上的像素灰度值与对应的卷积核上的数值相乘，然后将所有相乘后的值相加作为卷积核中间像素对应的图像上像素的灰度值，并最终滑动完所有图像的过程。



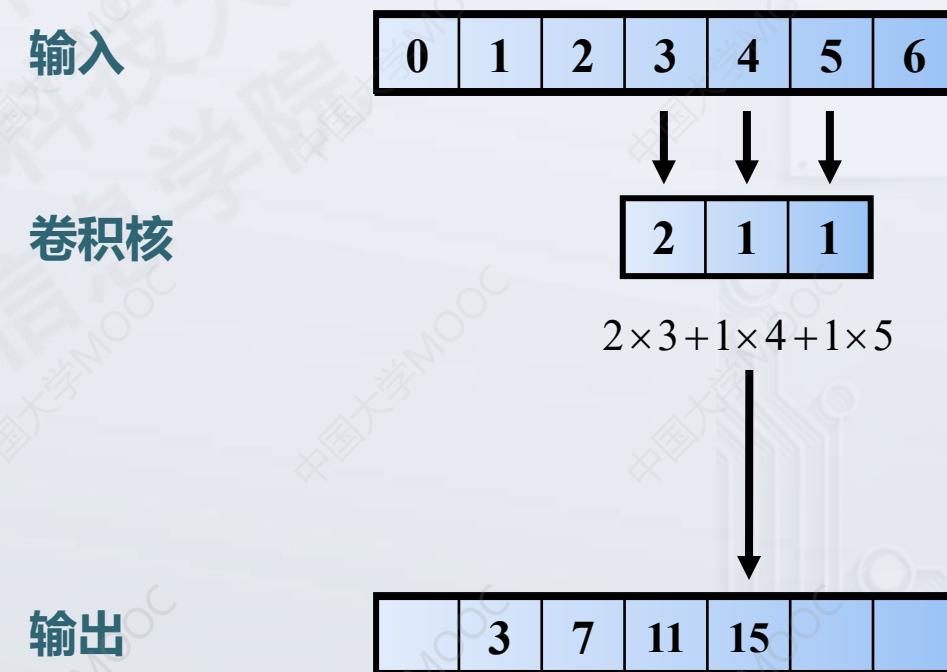
# 一维卷积实例

对数字图像做卷积操作其实就是利用卷积核在图像上滑动，将图像点上的像素灰度值与对应的卷积核上的数值相乘，然后将所有相乘后的值相加作为卷积核中间像素对应的图像上像素的灰度值，并最终滑动完所有图像的过程。



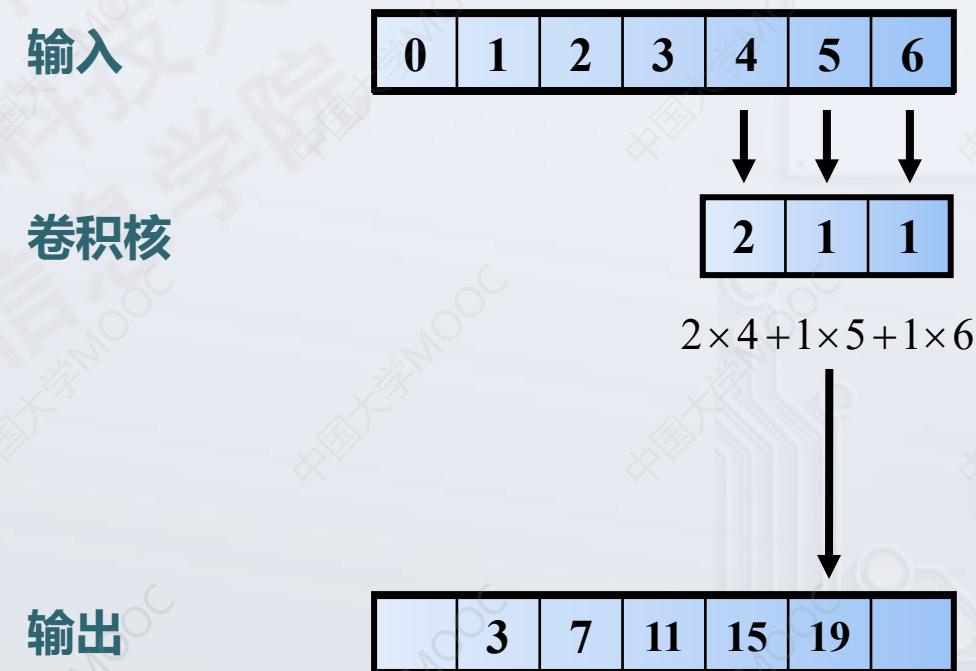
# 一维卷积实例

对数字图像做卷积操作其实就是利用卷积核在图像上滑动，将图像点上的像素灰度值与对应的卷积核上的数值相乘，然后将所有相乘后的值相加作为卷积核中间像素对应的图像上像素的灰度值，并最终滑动完所有图像的过程。



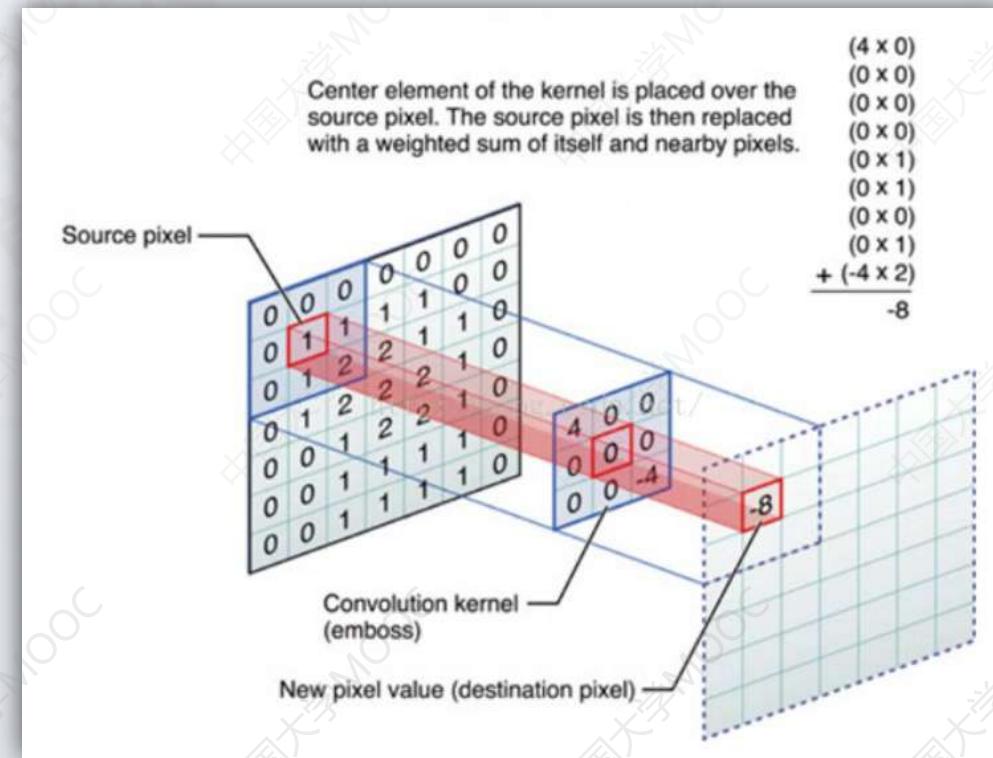
# 一维卷积实例

对数字图像做卷积操作其实就是利用卷积核在图像上滑动，将图像点上的像素灰度值与对应的卷积核上的数值相乘，然后将所有相乘后的值相加作为卷积核中间像素对应的图像上像素的灰度值，并最终滑动完所有图像的过程。



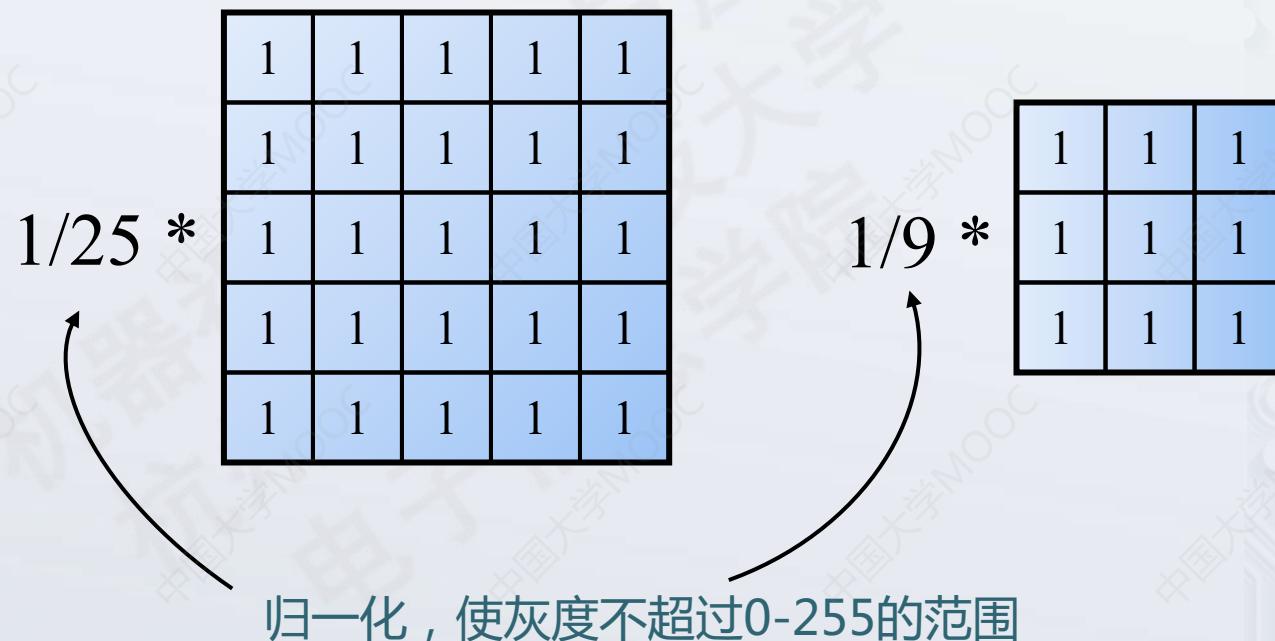
# 二维卷积实例

对数字图像做卷积操作其实就是利用卷积核在图像上滑动，将图像点上的像素灰度值与对应的卷积核上的数值相乘，然后将所有相乘后的值相加作为卷积核中间像素对应的图像上像素的灰度值，并最终滑动完所有图像的过程。



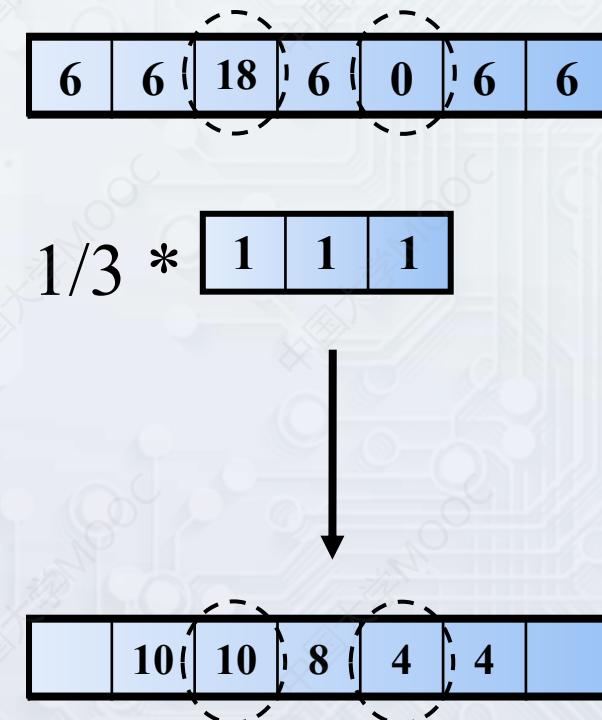
# 卷积的应用：均值滤波

两种基本的平滑卷积



卷积的物理意义：在 $5 \times 5$ 或 $3 \times 3$ 范围内取平均值。

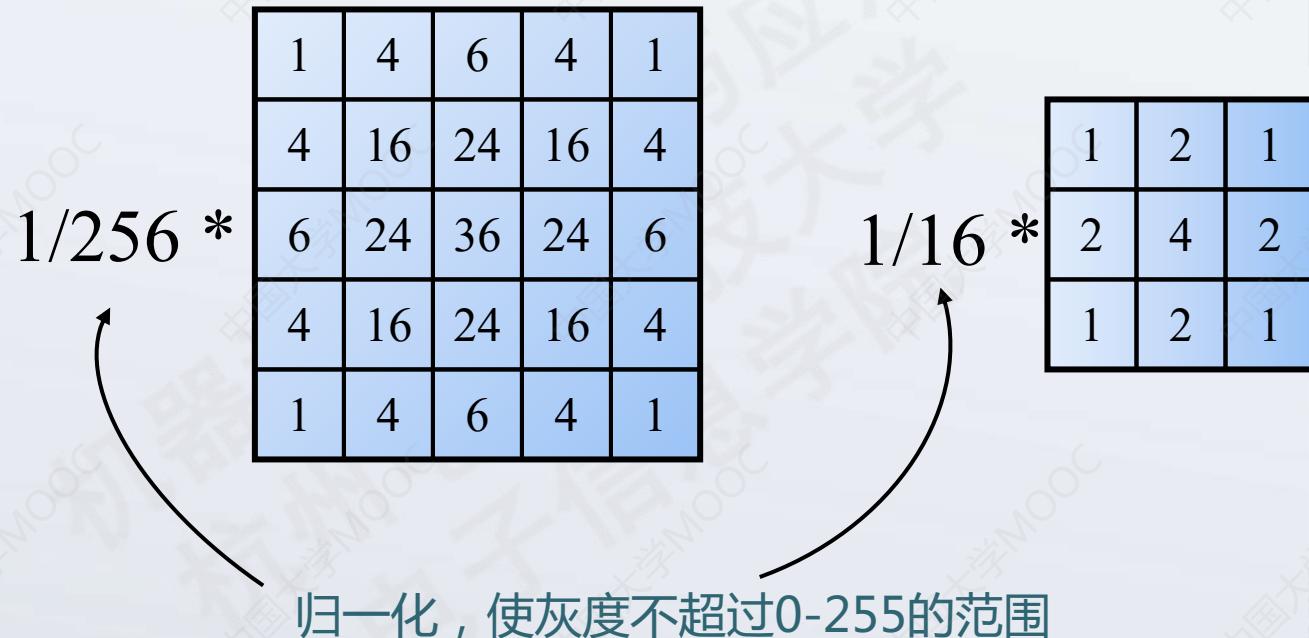
为方便讨论，以1D为例



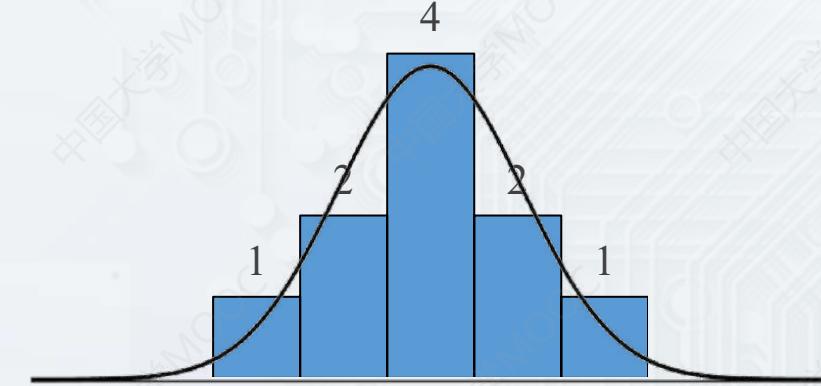
与周围差距较大的值趋向于与周围相似，整体值趋向于平均化

# 卷积的应用：平滑滤波

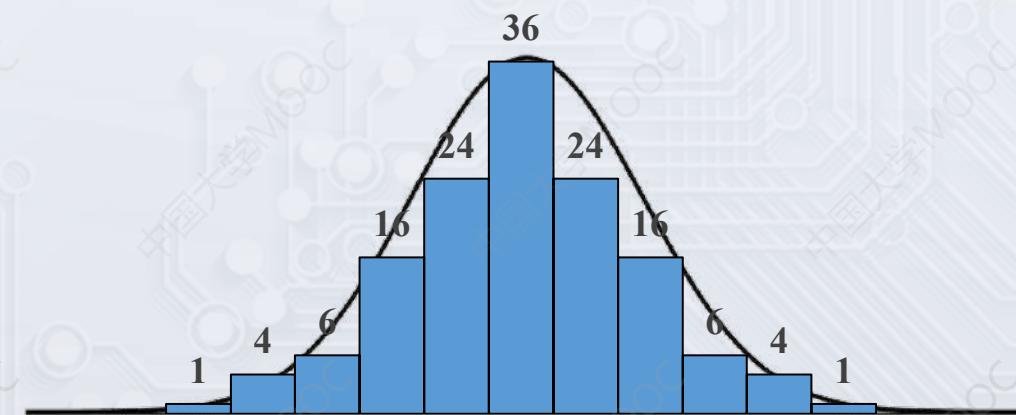
通过高斯分布加权的高斯平滑卷积



卷积的物理意义：在 $5 \times 5$ 或 $3 \times 3$ 范围内取平均值。同时距离待处理像素越近的像素权重更大，即对输出影响越大。

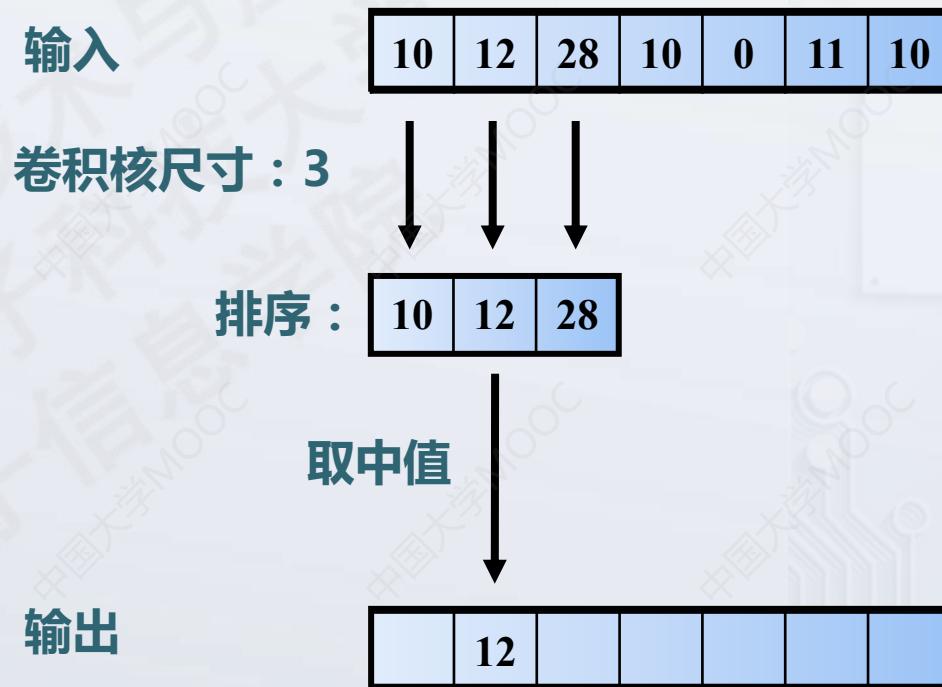


权重的取值为高斯分布的近似值



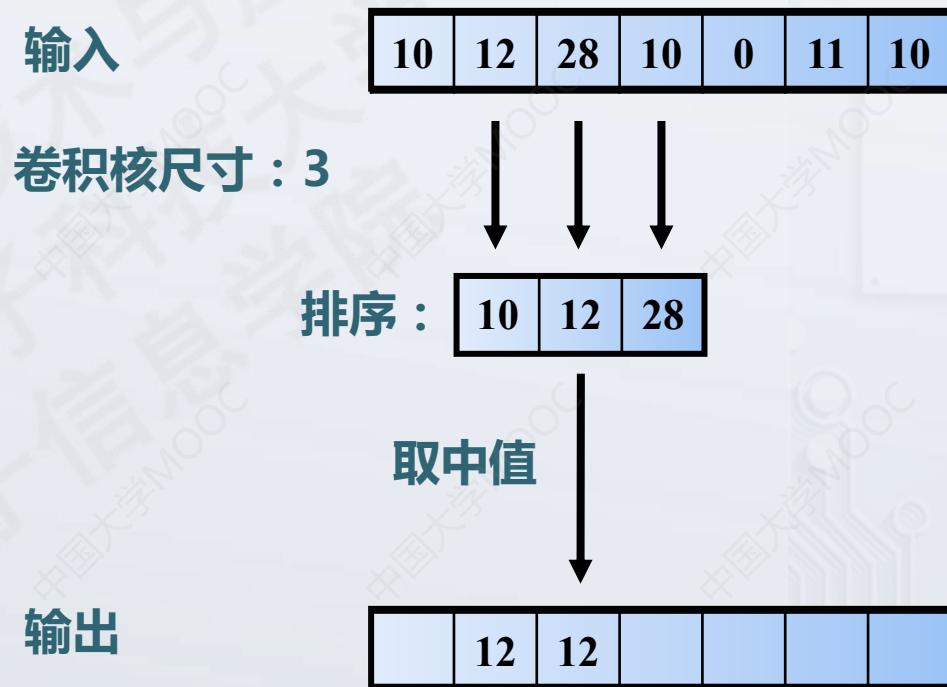
# 中值滤波

中值是一种非线性滤波，不需要制定卷积核，只需要制定滤波器尺寸



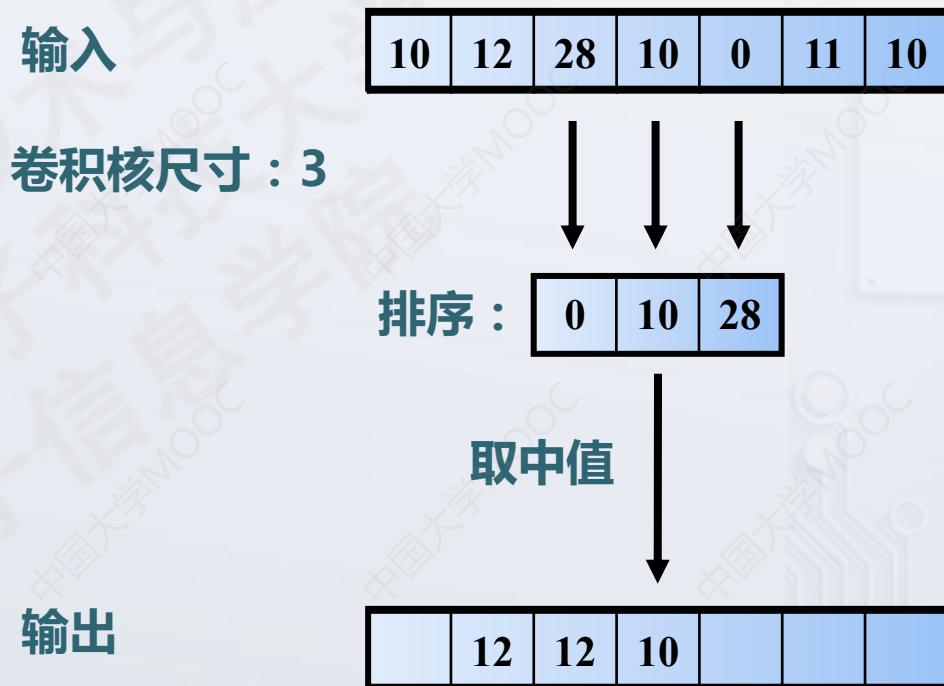
# 中值滤波

中值是一种非线性滤波，不需要制定卷积核，只需要制定滤波器尺寸



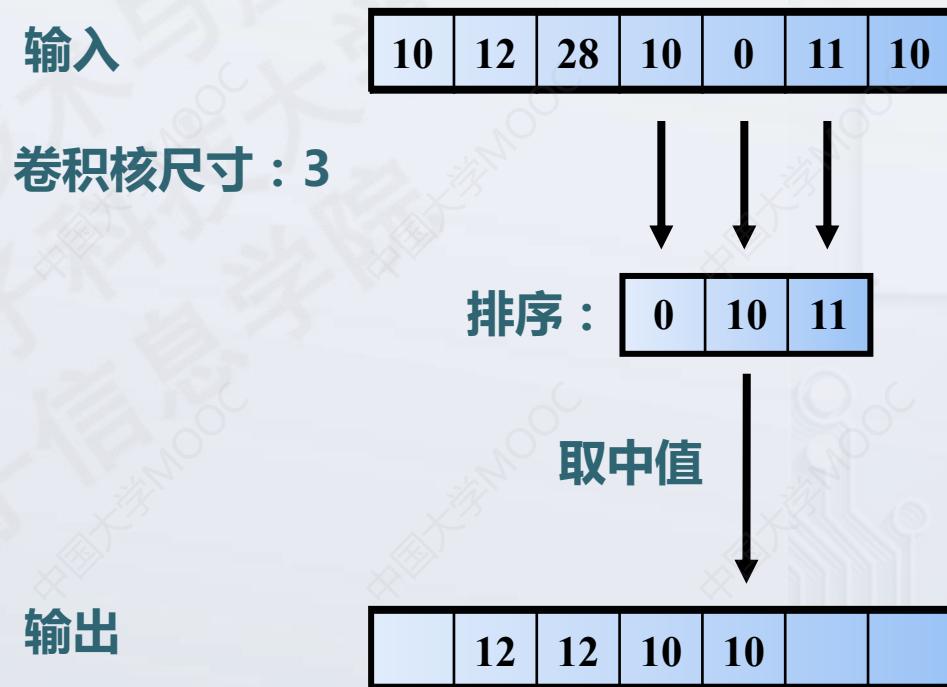
# 中值滤波

中值是一种非线性滤波，不需要制定卷积核，只需要制定滤波器尺寸



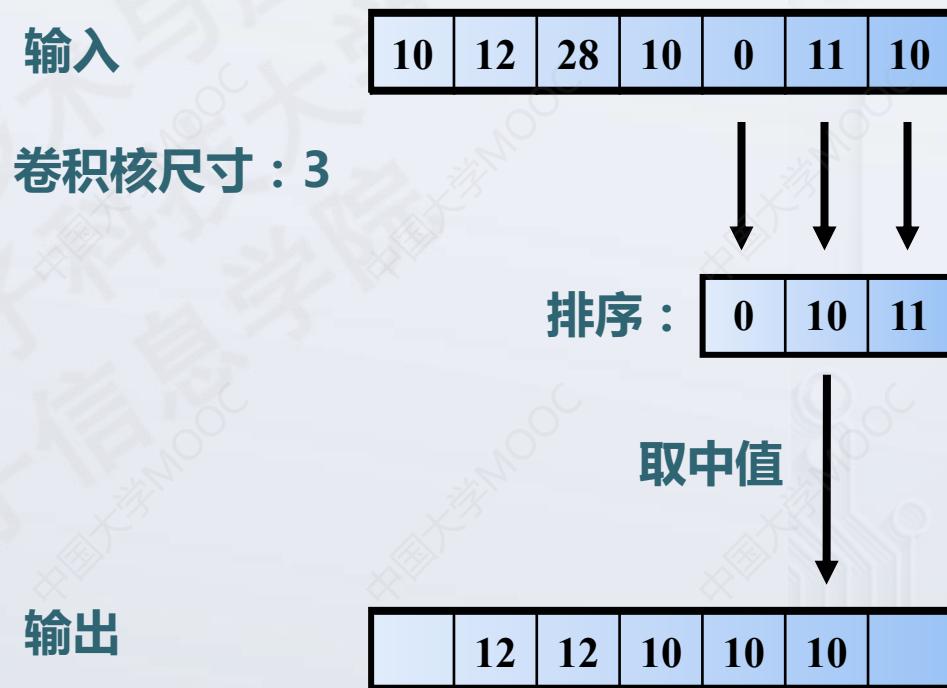
# 中值滤波

中值是一种非线性滤波，不需要制定卷积核，只需要制定滤波器尺寸



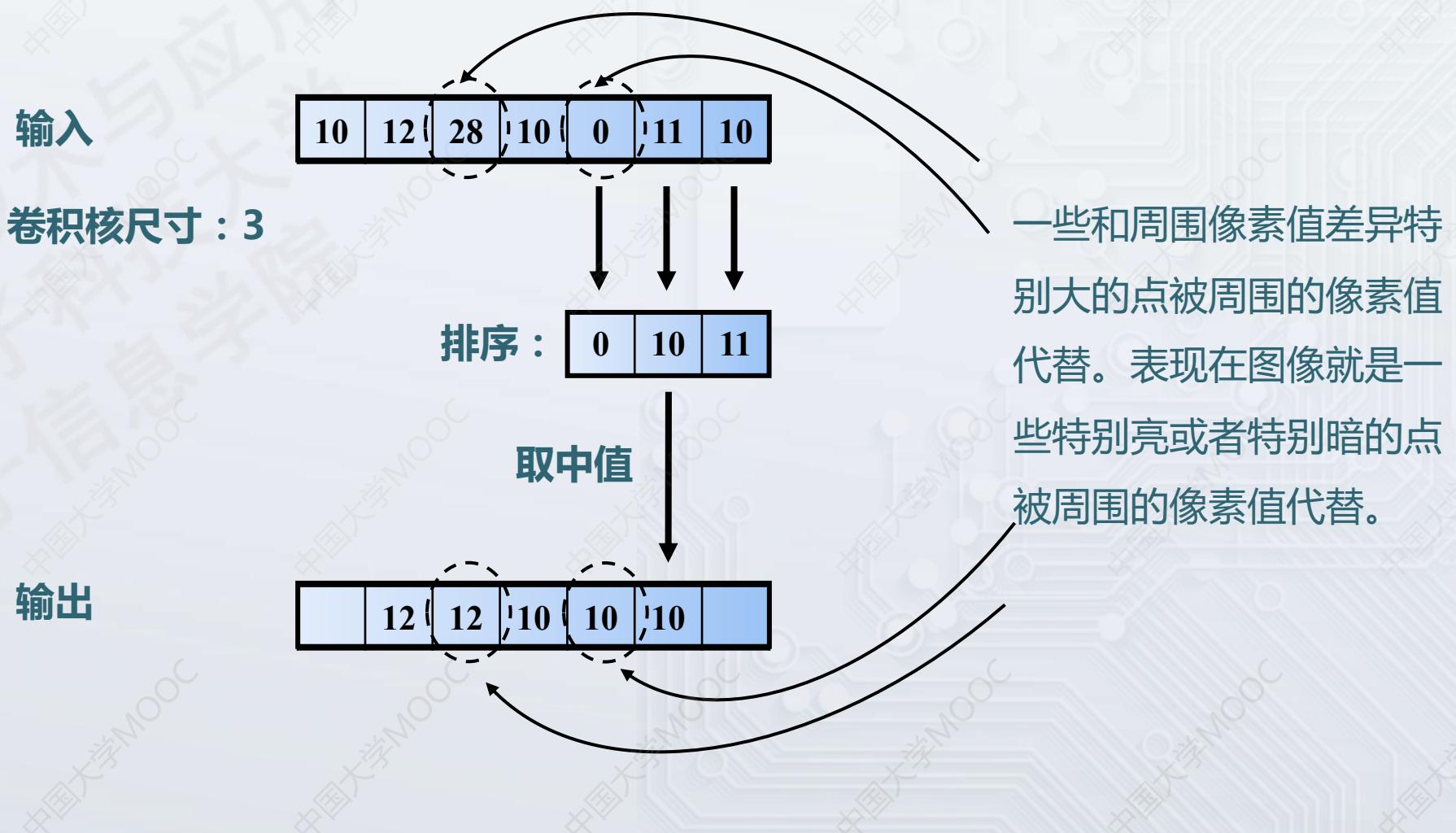
# 中值滤波

中值是一种非线性滤波，不需要制定卷积核，只需要制定滤波器尺寸



# 中值滤波

中值是一种非线性滤波，不需要制定卷积核，只需要制定滤波器尺寸



# 图像中两种常见的噪声

**椒盐噪声**：它是一种随机出现的白点或者黑点，即亮的区域有黑色像素或是在暗的区域有白色像素（或是两者皆有）。椒盐噪声的成因是图像信号受到突如其来的强烈干扰而产生。椒盐噪声通常使用**中值滤波**降噪

---



**高斯噪声**：主要来源是在采集过程中产生的，例如由照明不良和/或高温引起的传感器噪声。其概率分布上符合正太分布。高斯噪声通常使用**平滑滤波**进行降噪。



# 图像的边缘检测

边缘检测是图像处理中的重要问题，是很多图像处理算法的基本步骤之一。边缘是图像中的重要特征信息，如深度学习中的卷积神经网络，其本质就是通过卷积抓取基本的边缘特征，再不断向上构建更高层次的特征。

边缘在图像上表现为亮度变化剧烈的像素点。



# 图像的边缘检测

图像



像素值



一阶微分



二阶微分



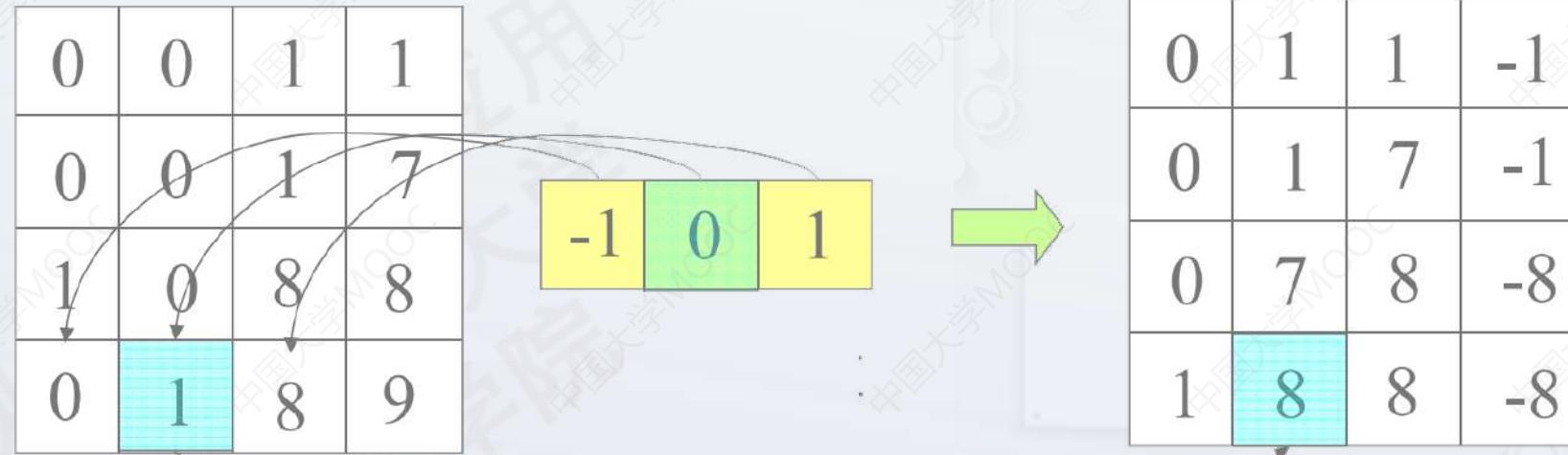
# 图像的边缘检测

离散信号的微分计算 = 减法运算

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

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

# 图像的边缘检测



$[-1, 0, 1]$ 卷积核表示，对每个像素使用他右边的像素值减去左边的像素值。即求水平上的微分值。如果使用右边的卷积核呢？

-1
0
1

# Prewitt 算子

其实质是把降噪的平滑卷积核  
边缘提取的卷积相结合



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



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



# Sobel 算子

如果使用近似高斯分布的平滑  
卷积，则：



$$\begin{matrix} * & \begin{matrix} -1 & & 1 \\ -2 & & 2 \\ -1 & & 1 \end{matrix} & = & \begin{matrix} \text{blurred image} \end{matrix} \end{matrix}$$



$$\begin{matrix} * & \begin{matrix} -1 & -2 & -1 \\ & & \\ 1 & 2 & 1 \end{matrix} & = & \begin{matrix} edge map \end{matrix} \end{matrix}$$



# 拉普拉斯（Laplacian）算子

拉普拉斯算子通过求图像的二阶微分，获得边缘信息

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

水平方向的一阶微分

-1	1	

-1	1	

水平方向  
二阶微分

	1	
1	-4	1
	1	

垂直方向的一阶微分

1		
-1		

1		
-1		

垂直方向  
二阶微分

1	-2	1

	1	
1	-2	1
	1	

	1	
1	-4	1
	1	

# canny算子

最常用的边缘检测算子，很好的解决了伪边缘的问题，主要步骤如下：

- 高斯卷积降噪
- 计算图像的一阶微分，方向和幅值
- 非极大值抑制
- 双阈值处理



原图



sobel



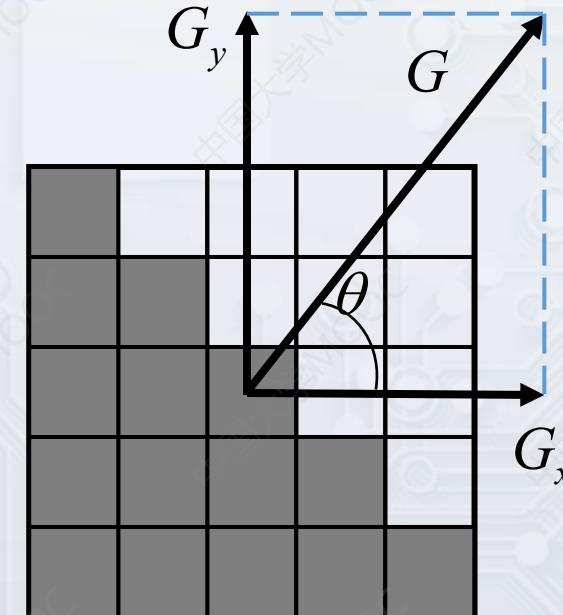
canny

# canny算子

边缘的方向和幅值，一阶微分通常只求水平及垂直方向的梯度，在此基础上 canny算子进一步求出边缘的实际方向和该方向上的幅值。

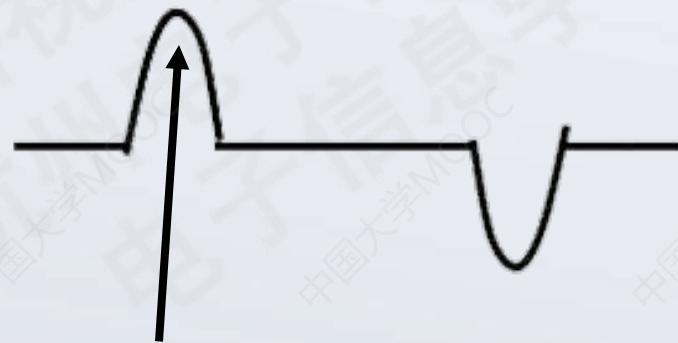
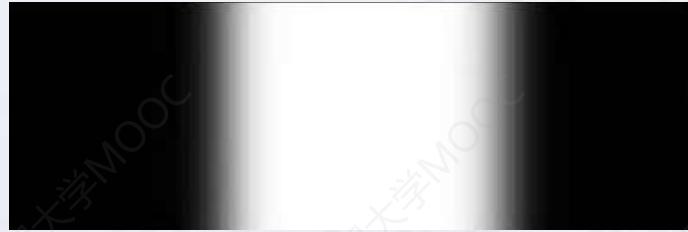
幅值：  $G = \sqrt{G_x^2 + G_y^2}$

方向：  $\theta = \tan^{-1}\left(\frac{G_y}{G_x}\right)$

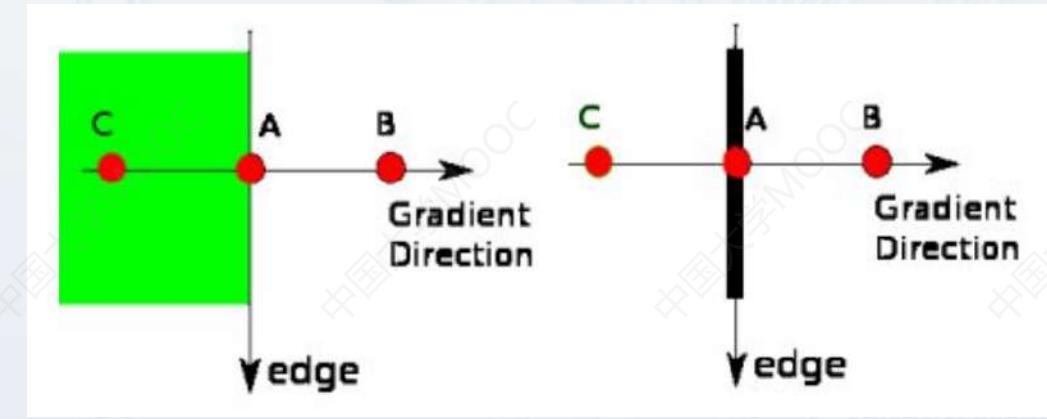


# canny算子

通过非极大值抑制，找到梯度方向上的局部最大值。



并非梯度响应值大的像素就是边缘像素，而是局部最大值为边缘像素。



判断点A是否为边缘点，需要把和其边缘方向上的B点和C点进行比较，如果是最值的话则进入下一步，否则置0（抑制）。

# canny算子

双阈值筛选并连接边缘像素，设置最大阈值HT和最小阈值LT。

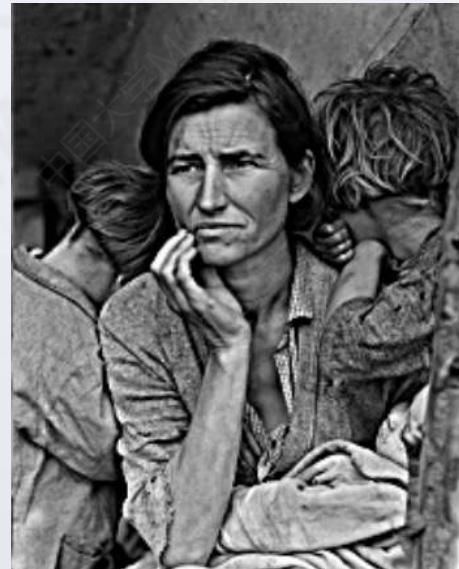
像素	评价
$\text{梯度值} > \text{HT}$	边缘像素
$\text{HT} > \text{梯度值} > \text{LT}$	如果该像素周围有边缘像素，则该像素位边缘像素，否则为非边缘像素。
$\text{梯度值} < \text{LT}$	非边缘像素

# 锐化算子

图像锐化可以实现边缘增强，凸显细节。可通过以下算子实现，实质是拉普拉斯算子。



原图



锐化结果

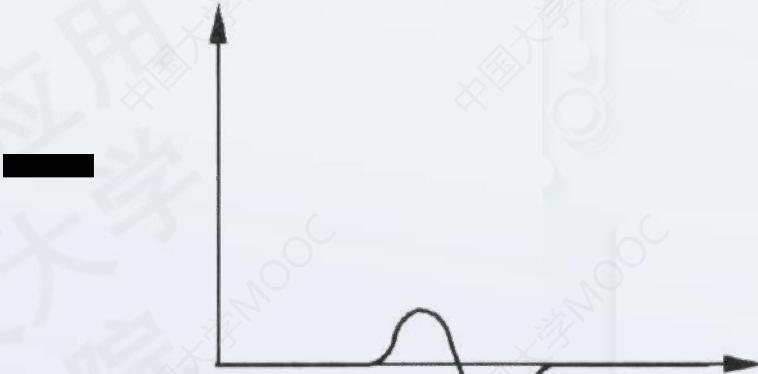
	-1	
-1	<b>5</b>	-1
	-1	

卷积核

# 锐化算子



原图



二阶微分



锐化（边缘增强）

A 3x3 kernel matrix where all elements are zero except for the center element, which is 1.

0	0	0
0	1	0
0	0	0

A 3x3 kernel matrix for edge detection. It uses a central value of 1 and surrounding values of -4 and 1 to calculate the second derivative.

0	0	0
1	-4	1
0	0	0

A 3x3 kernel matrix for edge detection. It uses a central value of 5 and surrounding values of -1 and -1 to calculate the second derivative.

0	0	0
-1	5	-1
0	-1	0

# 谢谢！

# 机器视觉技术与应用

## 6. 几何变换

李竹

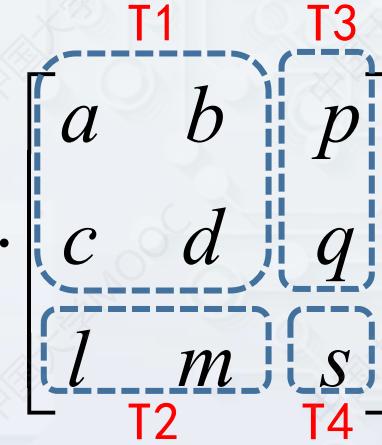
杭州电子科技大学

电子信息学院



# 2维度几何运算矩阵

$$\begin{bmatrix} x' & y' & 1 \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} \cdot T_{2D} = \begin{bmatrix} x & y & 1 \end{bmatrix} \cdot$$



T1 : 比例、旋转、对称、错切

T2 : 平移

T3 : 投影

T4 : 整体缩放

# 齐次坐标

$$\begin{bmatrix} x' & y' & 1 \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} \cdot T_{2D} = \begin{bmatrix} x & y & 1 \end{bmatrix} \cdot \begin{bmatrix} a & b & p \\ c & d & q \\ l & m & s \end{bmatrix}$$

齐次坐标：对一个在2维平面上的点  $(x, y)$ ，对任意非零实数  $Z$ ，  
三元组  $(xZ, yZ, Z)$  即称之为该点的齐次坐标。

# 齐次坐标

使用 $n+1$ 维，来表示 $n$ 维的坐标。

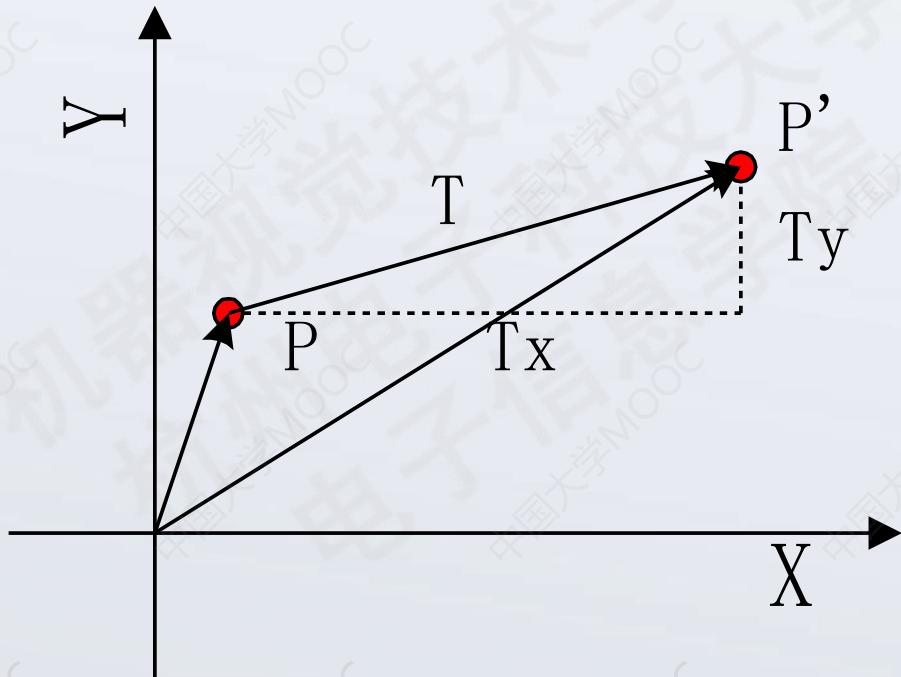
- 统一坐标的加法运算和乘法运算，  
运算时提高效率。
- 表示无穷远的点。

$$(x, y, z) \rightarrow (x/z, y/z)$$

- 如， $(2,2,1)$ ， $(4,4,2)$ 表示同样的点。
- 当 $z=0$ 的时候，表示无穷远的点。

# 平移变换

- 原坐标 $(x, y)$ , 平移后坐标 $(x', y')$



$$\begin{cases} x' = x + Tx \\ y' = y + Ty \end{cases}$$

矩阵相乘形式

$$\begin{bmatrix} x' & y' & 1 \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ T_x & T_y & 1 \end{bmatrix}$$

平移是一种不产生形变而移动物体的刚体变换,  $T_x$ 和 $T_y$  称为平移向量

# 比例缩放

图像的比例缩放是指将给定的图像在x轴方向按比例缩放 $a$ 倍，在y轴方向按比例缩放 $b$ 倍，从而获得一幅新的图像。如果 $a=b$ ，称这样的比例缩放为图像的全比例缩放。如果 $a$ 不等于 $b$ ，图像的比例缩放会改变原始图像的像素间的相对位置，产生几何畸变。

- 原坐标 $(x, y)$ ，缩放后坐标 $(x', y')$

$$\begin{cases} x' = x s_x \\ y' = y s_y \end{cases}$$

# 比例缩放

$$\begin{cases} x' = S_x \times x \\ y' = S_y \times y \end{cases}$$

矩阵相乘形式



$$\begin{bmatrix} x' & y' & 1 \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} \cdot \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

# 比例缩放

$$\begin{cases} x' = S_x \times x \\ y' = S_y \times y \end{cases}$$

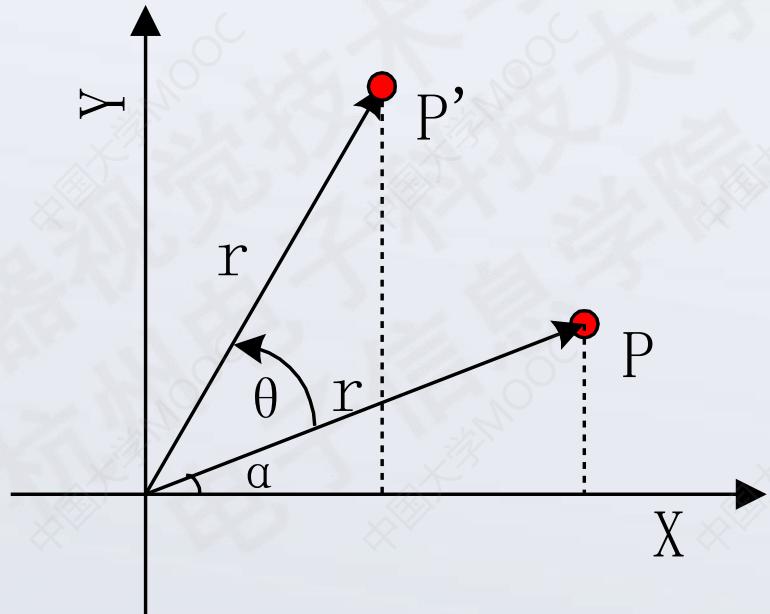
矩阵相乘形式



$$\begin{bmatrix} x' & y' & 1 \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} \cdot \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

# 旋转

- 原坐标 $(x, y)$ , 旋转后坐标 $(x', y')$



逆时针旋转  $\theta$  角

$$\begin{aligned}x' &= r \cos(a + \theta) \\&= r \cos a \cos \theta - r \sin a \sin \theta \\&= x \cos \theta - y \sin \theta\end{aligned}$$

$$\begin{aligned}y' &= r \sin(a + \theta) \\&= r \cos a \sin \theta + r \sin a \cos \theta \\&= x \sin \theta + y \cos \theta\end{aligned}$$

# 旋转

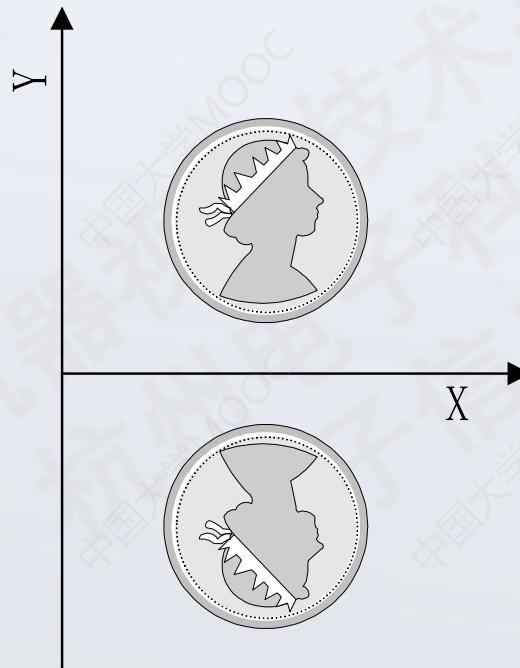
$$\begin{bmatrix} x' & y' & 1 \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} \cdot \begin{bmatrix} a & b & p \\ c & d & q \\ l & m & s \end{bmatrix}$$

$$\begin{aligned} x' &= r \cos(a + \theta) \\ &= r \cos a \cos \theta - r \sin a \sin \theta \\ &= x \cos \theta - y \sin \theta \end{aligned}$$

$$\begin{bmatrix} \cos \theta & \sin \theta & 0 \\ -\sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

# 对称变换

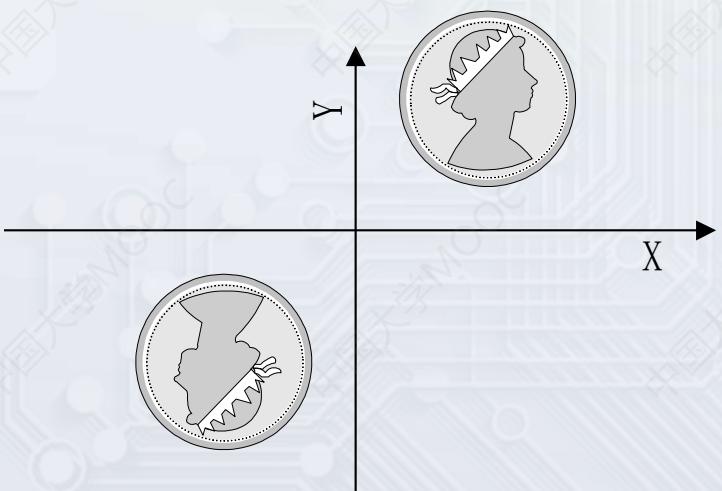
对称变换后的图形是原图形关于某一轴线或原点的镜像。



X轴对称



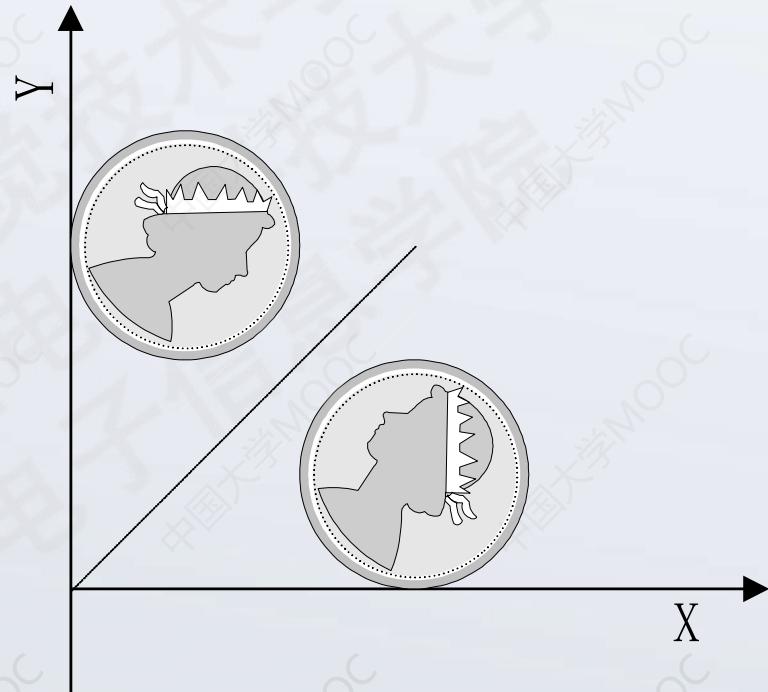
y轴对称



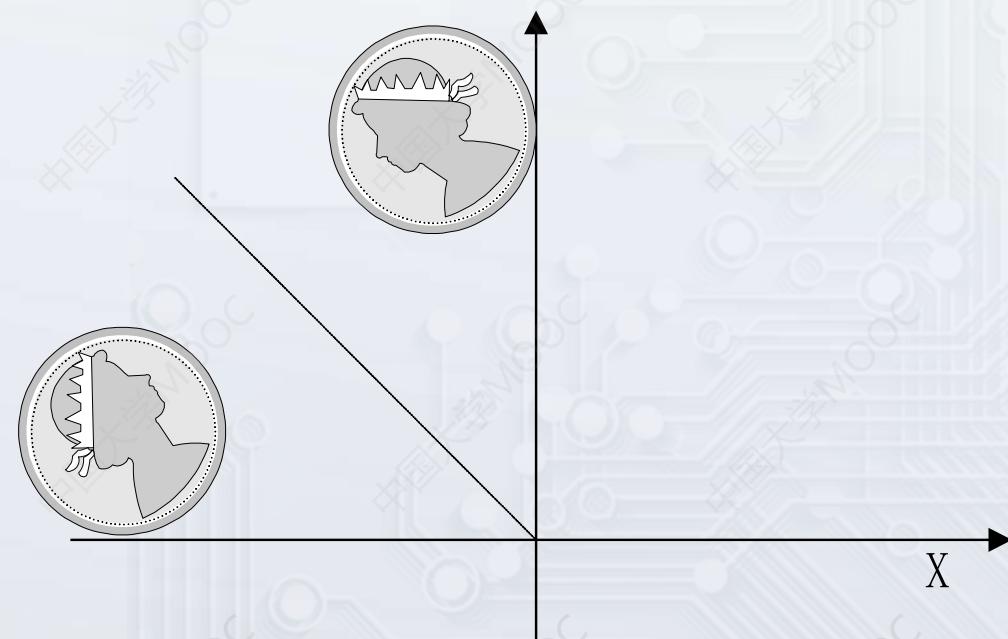
原点对称

# 对称变换

对称变换后的图形是原图形关于某一轴线或原点的镜像。



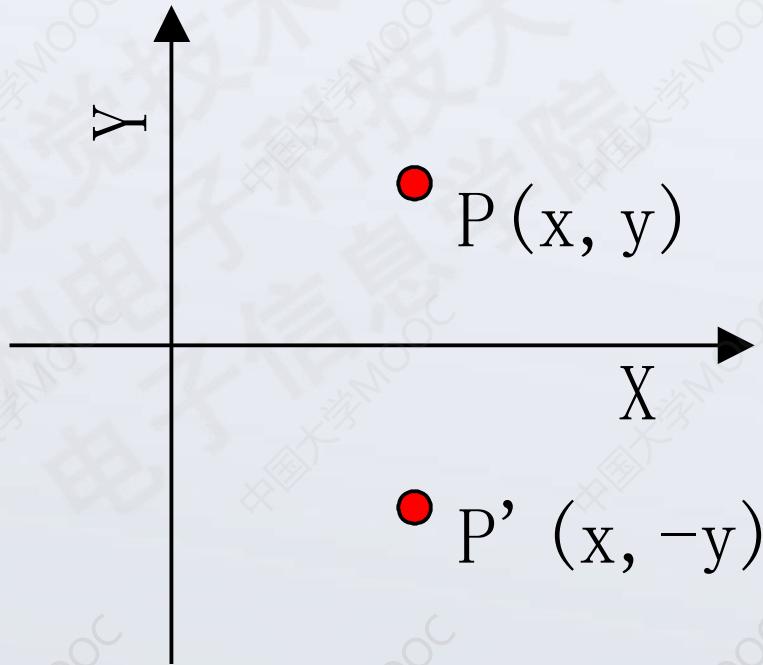
关于 $x=y$ 轴对称



关于 $x=-y$ 轴对称

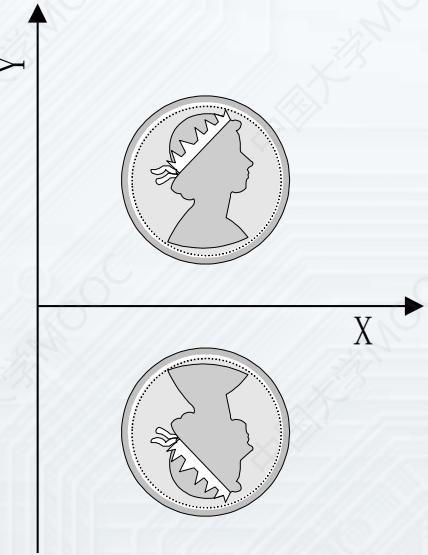
# 对称变换

关于x轴对称



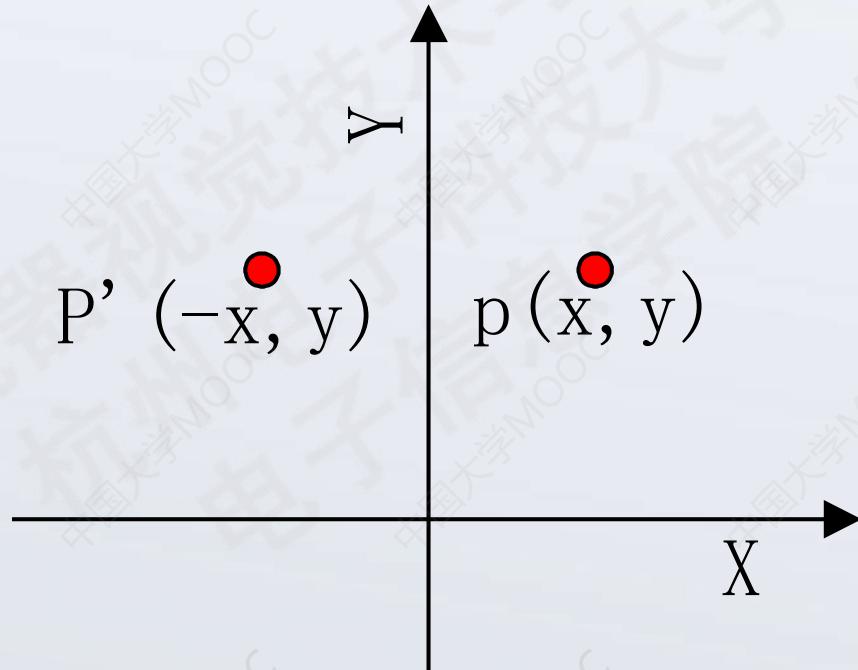
变换矩阵

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



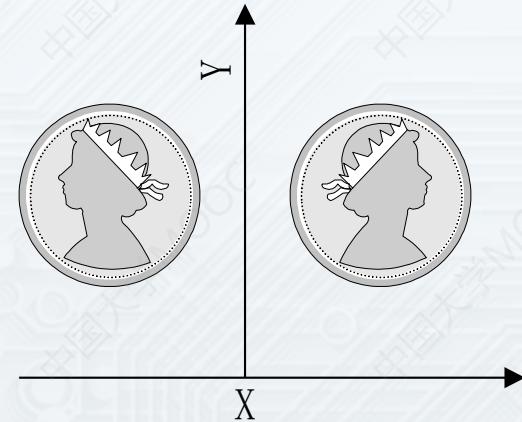
# 对称变换

关于y轴对称



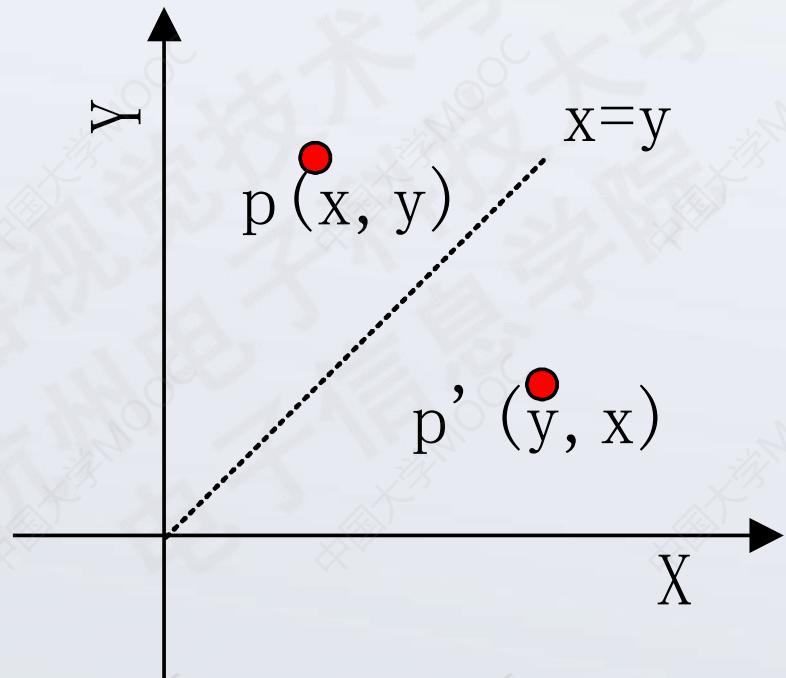
变换矩阵

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



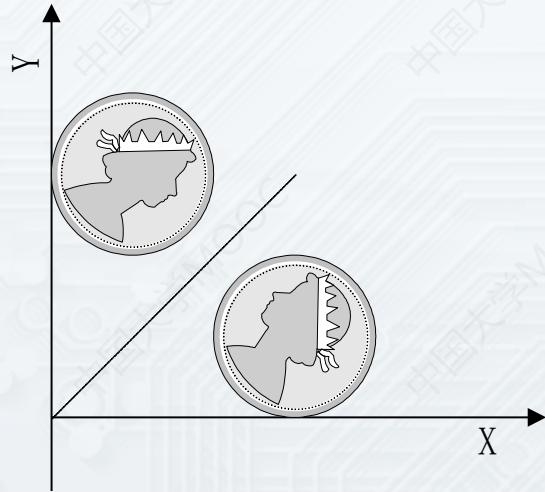
# 对称变换

关于 $x=y$ 轴对称



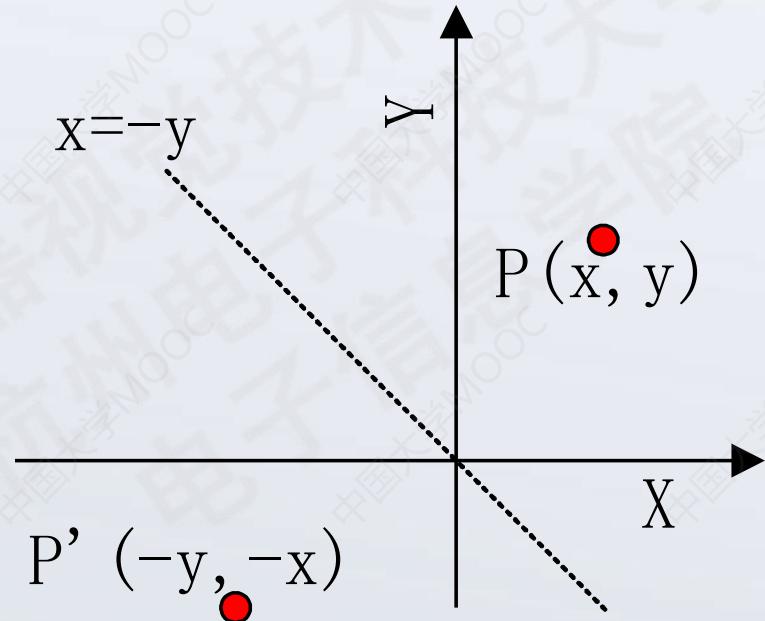
变换矩阵

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



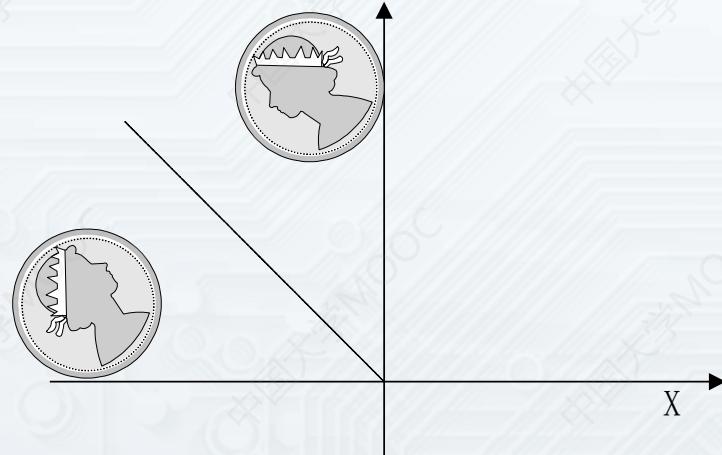
# 对称变换

关于 $x=-y$ 轴对称



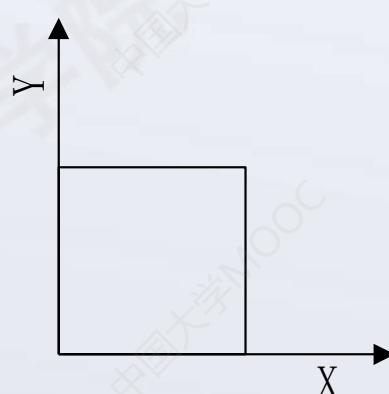
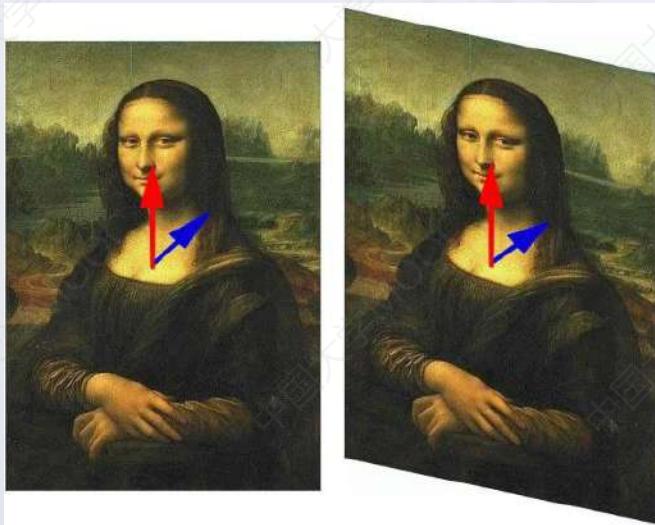
变换矩阵

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

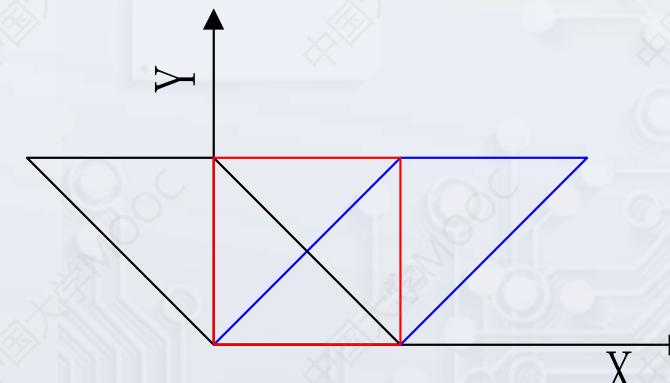


# 错切变换

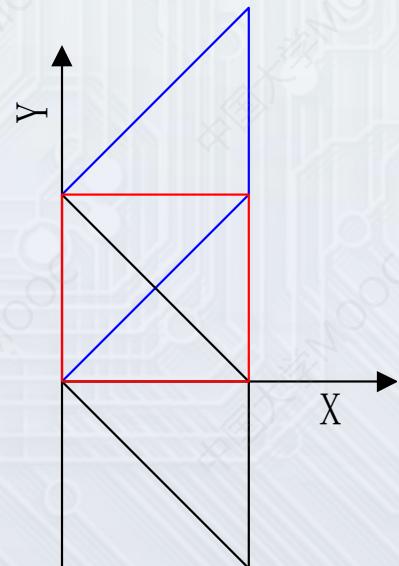
错切变换，也称为剪切、错位变换，用于产生弹性物体的变形处理。



原图



X方向错切



y方向错切

# 错切变换

错切变换，也称为剪切、错位变换，用于产生弹性物体的变形处理。

$$\begin{bmatrix} 1 & d & 0 \\ b & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- (1) 改变d，x方向错切。
- (2) 改变b，y方向错切。
- (3) 改变2个参数，两个方向同时错切。

# 复合变换

复合变换是指：

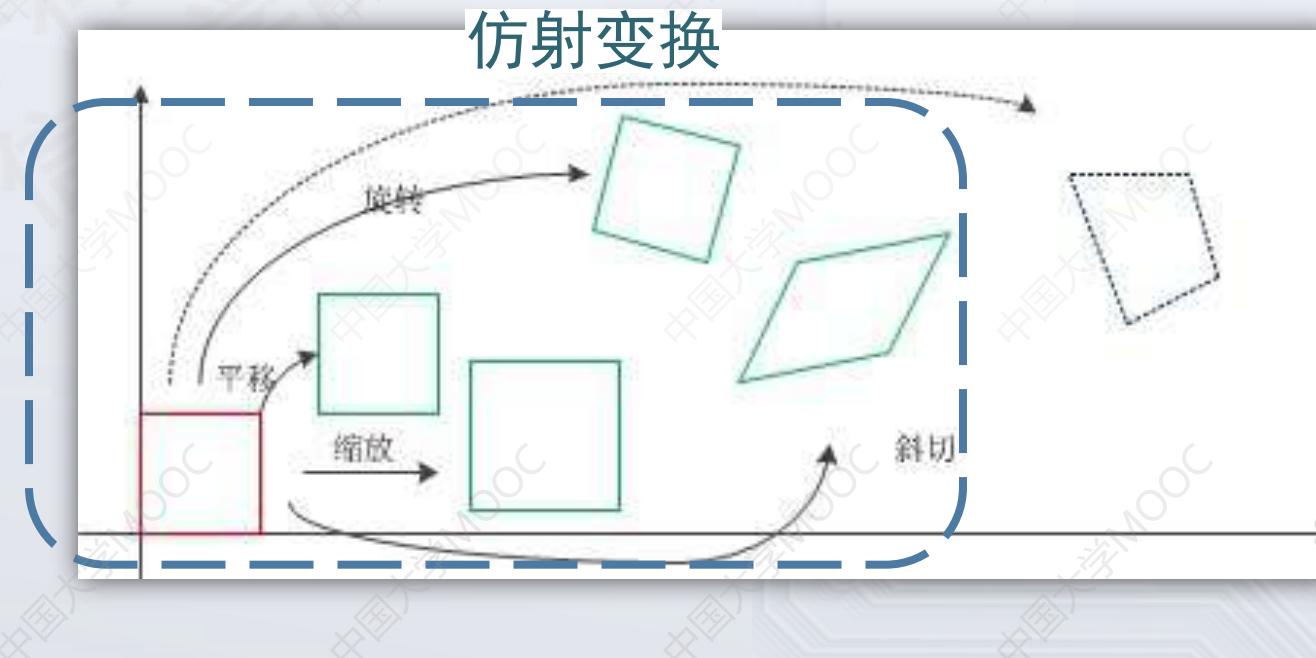
- 图形作一次以上的几何变换，变换结果是每次的变换矩阵相乘。
- 任何一复杂的几何变换都可以看作基本几何变换的组合形式。

复合变换具有形式：

$$\begin{aligned} P' &= P \cdot T = P \cdot (T_1 \cdot T_2 \cdot T_3 \cdots T_n) \\ &= P \cdot T_1 \cdot T_2 \cdot T_3 \cdots T_n \quad (n > 1) \end{aligned}$$

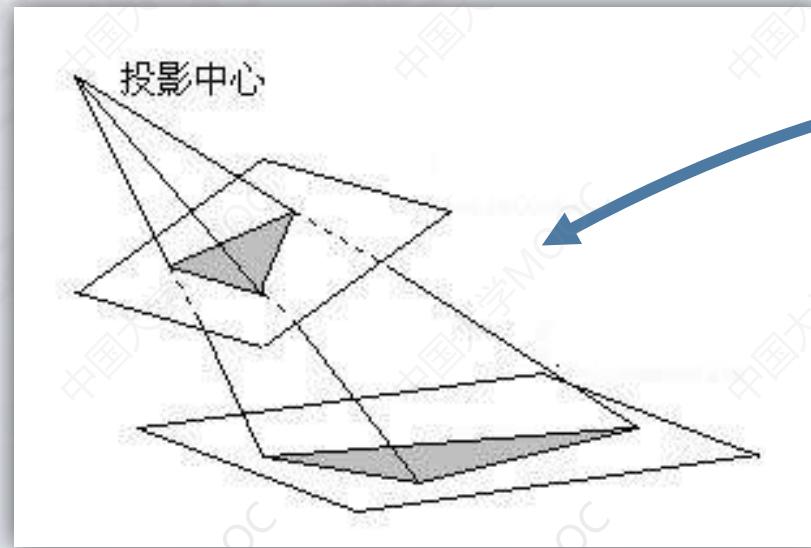
# 仿射变换和投影变换

仿射允许图像任意移动，任意倾斜，在x和y方向上任意伸缩。仿射变换中，点共线特性，及平行线平行特性不变。但线段长度，及直线夹角不一定保持不变。



# 仿射变换和投影变换

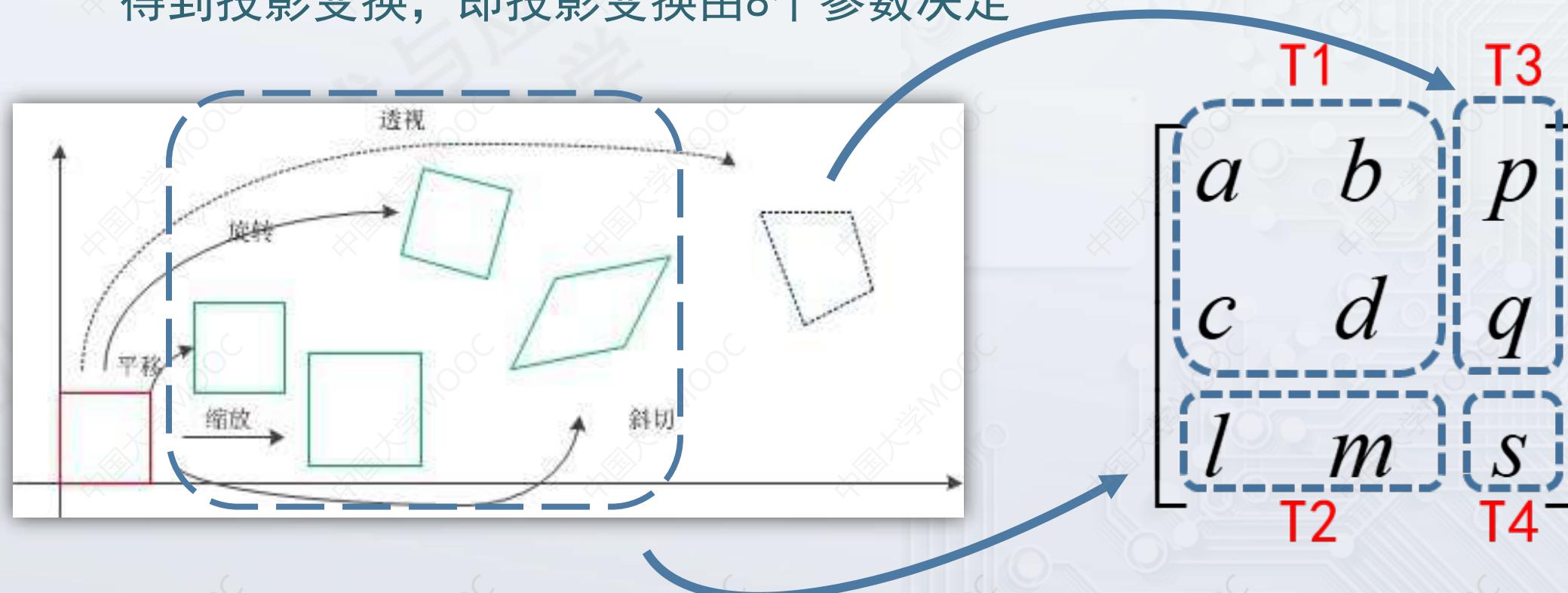
经过投影变换的图像，可以看做物体在投影面上形成的投影图像。其性质是点共线特性仍旧保持，平行线有可能不再平行。



该图像的变形即为投影变换

# 仿射变换和投影变换

如果在仿射变换的基础上，继续改变T3部分的2个参数，可以得到投影变换，即投影变换由8个参数决定



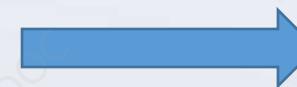
S取1，所有的仿射变换均可以通过改变6个参数决定。

# 投影变换的应用

我们在扫描二维码的时候，往往做不到使成像面完全平行于实际画面，可通过投影变换进行矫正。



实际拍摄的图像



投影变换



矫正后的图像

# 仿射矩阵的获得

将仿射变换写成方程形式，可以看出，如果要计算出仿射变换后的新图像，需要求解6个未知数，则需要6个方程式。

$$\begin{bmatrix} x' & y' & 1 \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} \cdot \begin{bmatrix} a & b & 0 \\ c & d & 0 \\ l & m & 1 \end{bmatrix}$$

$$x' = ax + cy + l$$

$$y' = bx + dy + m$$

如果知道某个原图中的点在变换后的图中的位置，则可以得到2个方程式。故仿射变换至少要知道3对点的对应关系。



# 投影矩阵的获得

投影变换中有8个未知数，需要8个方程式。即需要4组对应点信息。

$$\begin{bmatrix} x' & y' & 1 \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} \cdot \begin{bmatrix} a & b & p \\ c & d & q \\ l & m & 1 \end{bmatrix}$$

$$x' = \frac{ax + cy + l}{px + qy + 1} \quad y' = \frac{bx + dy + m}{px + qy + 1}$$

如果知道某个原图中的点在变换后的图中的位置，则可以得到2个方程式。故投影变换至少要知道4对点的对应关系。



# 投影矩阵的获得

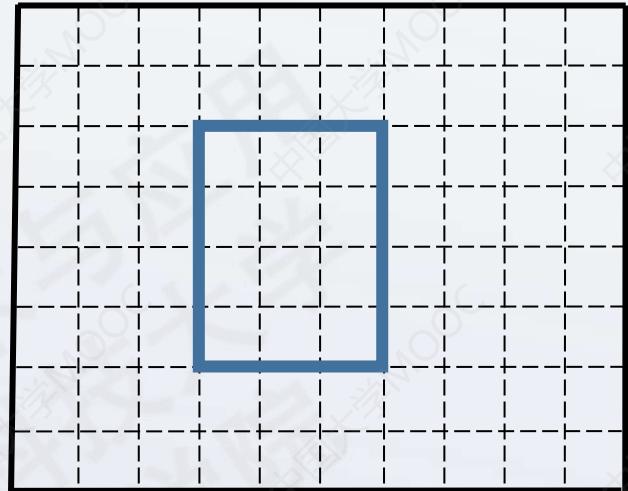
对应点的匹配关系通常可以通过特征点匹配实现，将在后续章节介绍。



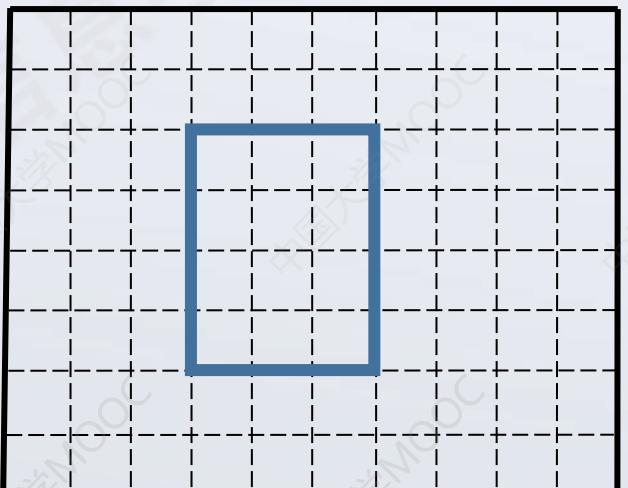
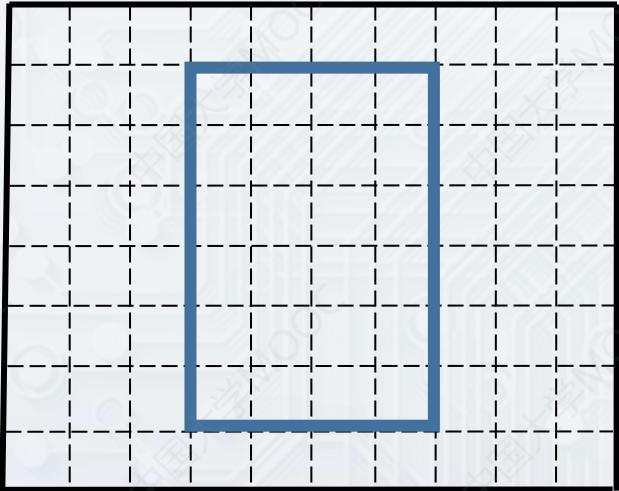
# 插值算法

图像发生几何变化时，像素数量发生变化。

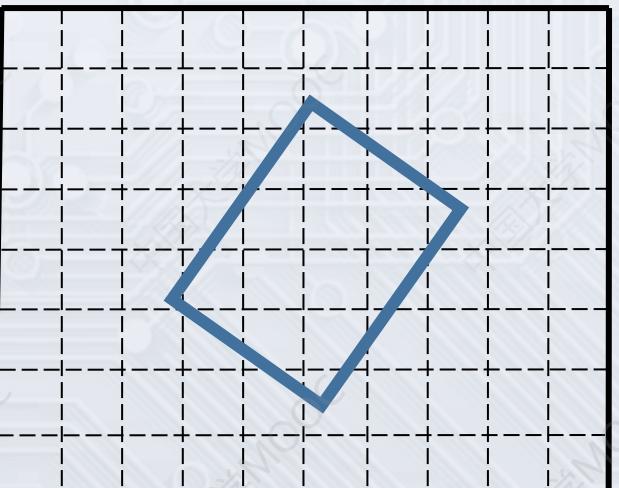
需要使用插值算法估算新像素的值。



缩放  
→

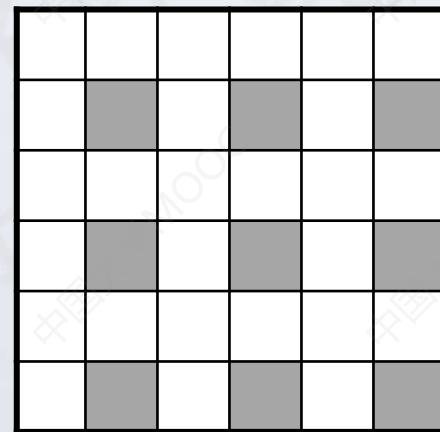


旋转  
→

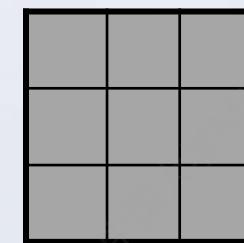
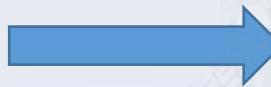


# 插值算法

当图像缩小时，新图像的像素值可以采用等间距采样的方式获得新图像的像素值。



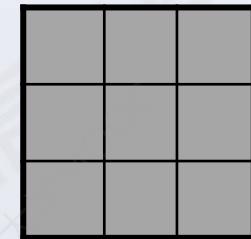
原图



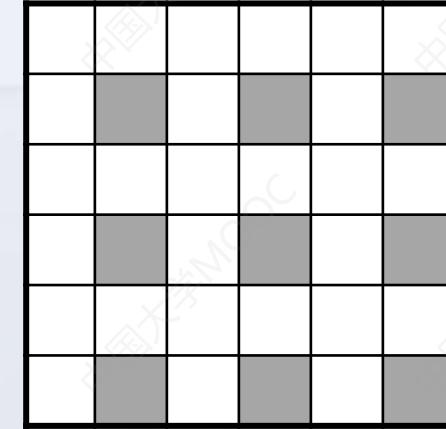
缩小后

# 插值算法

当图像放大时，则需要对新产生的像素的像素值进行推算，即对图像进行插值。



原图



放大后

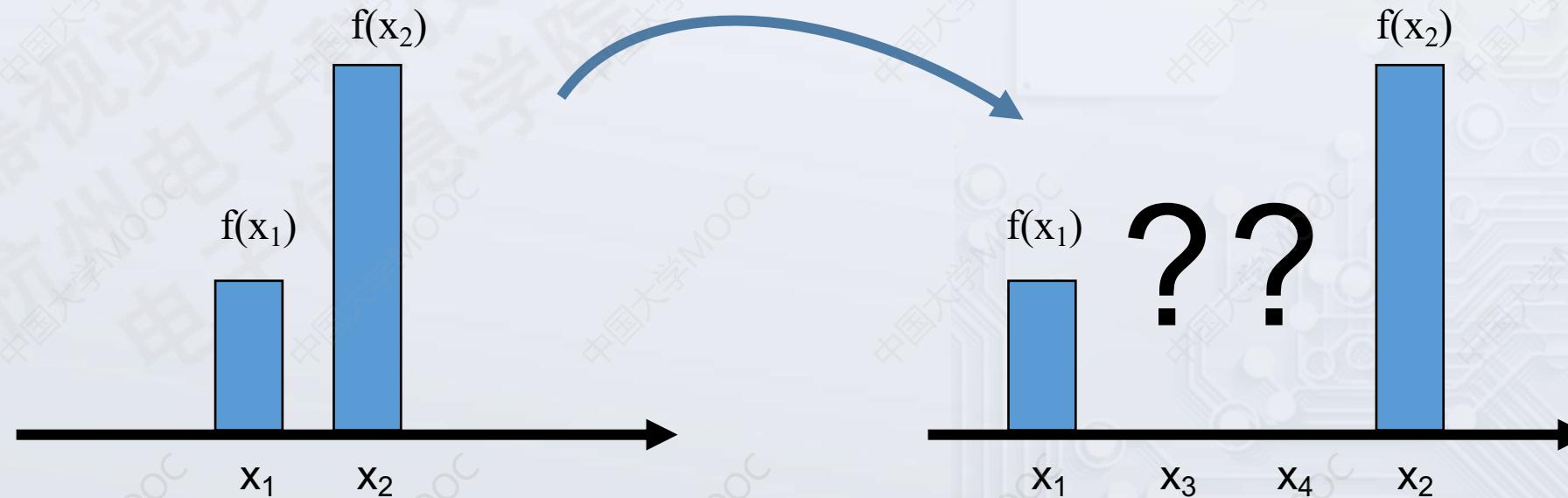
# 经典插值算法

1. 最近邻插值法(nearest neighbor interpolation)
2. 双线性插值法(bilinear interpolation)
3. 双三次插值法(bicubic interpolation)

# 最近邻插值法

最近邻插值法，不需要计算，是最快的插值算法。

对待定的像素值，直接取其最近的像素的像素值。

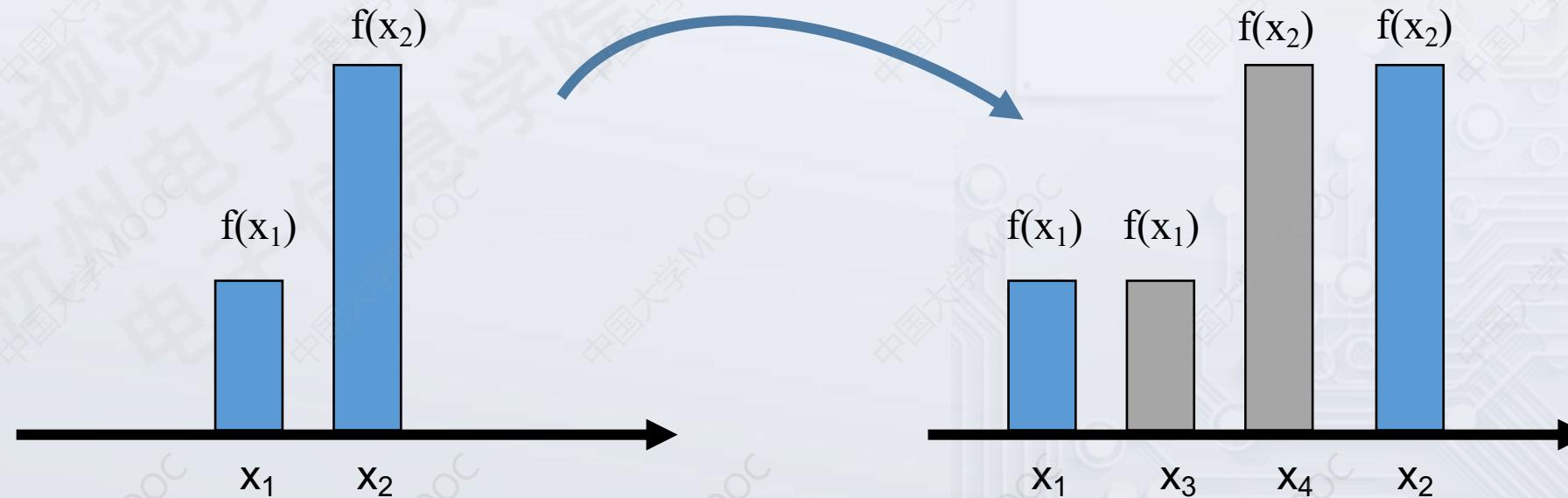


1D的例子

# 最近邻插值法

最近邻插值法，不需要计算，是最快的插值算法。

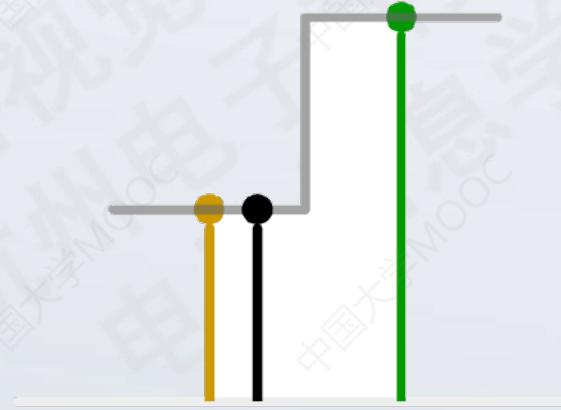
对待定的像素值，直接取其最近的像素的像素值。



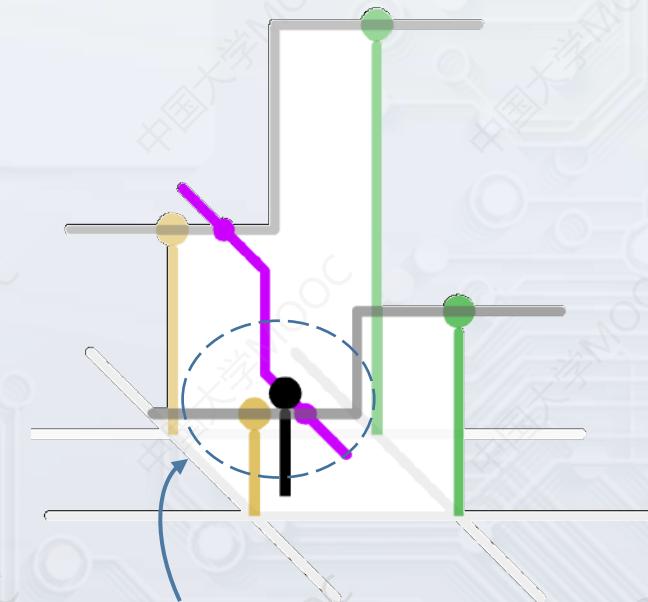
1D的例子

# 最近邻插值法

最近邻插值法的缺点是插值生成的图像灰度上的不连续，在灰度变化的地方可能出现明显的锯齿状。



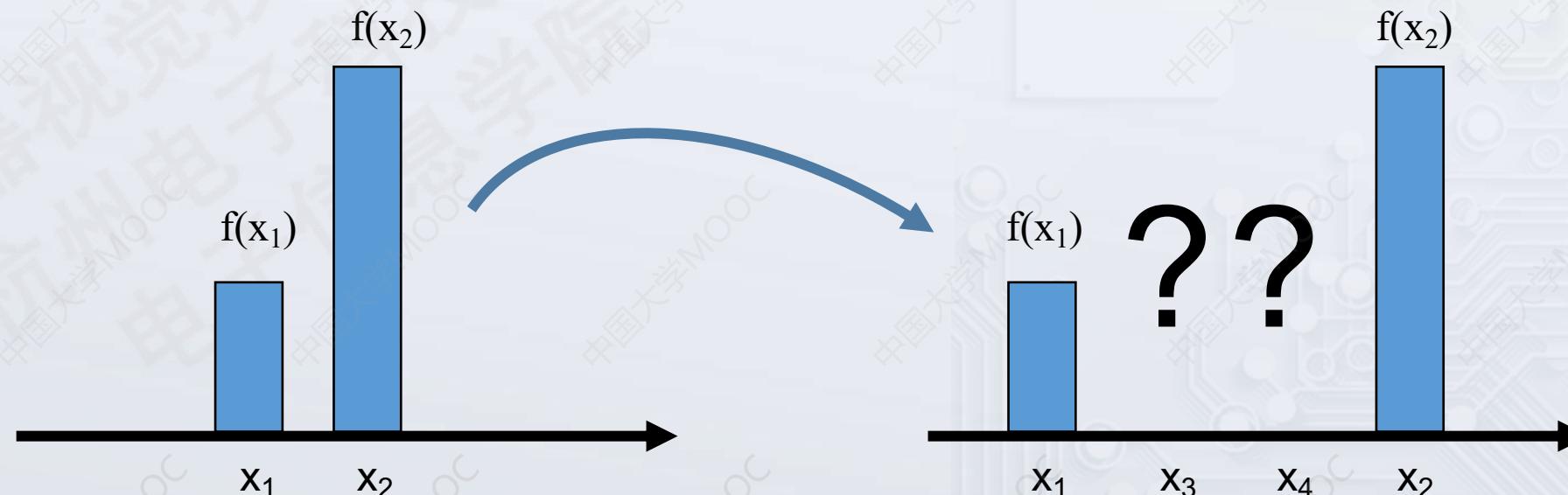
2D的例子



2维平面中取最近像素值

# 线性插值法

线性插值用过两插值节点的直线近似原函数，推算新的像素值。

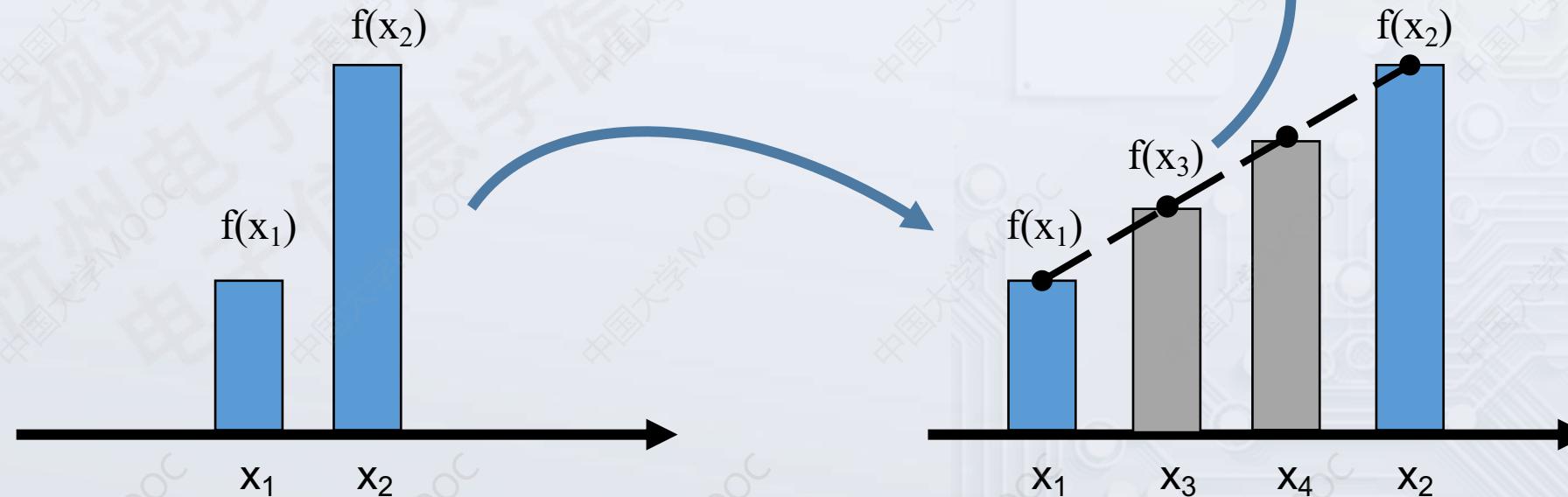


1D的例子

# 线性插值法

线性插值用过两插值节点的直线近似原函数，推算新的像素值。

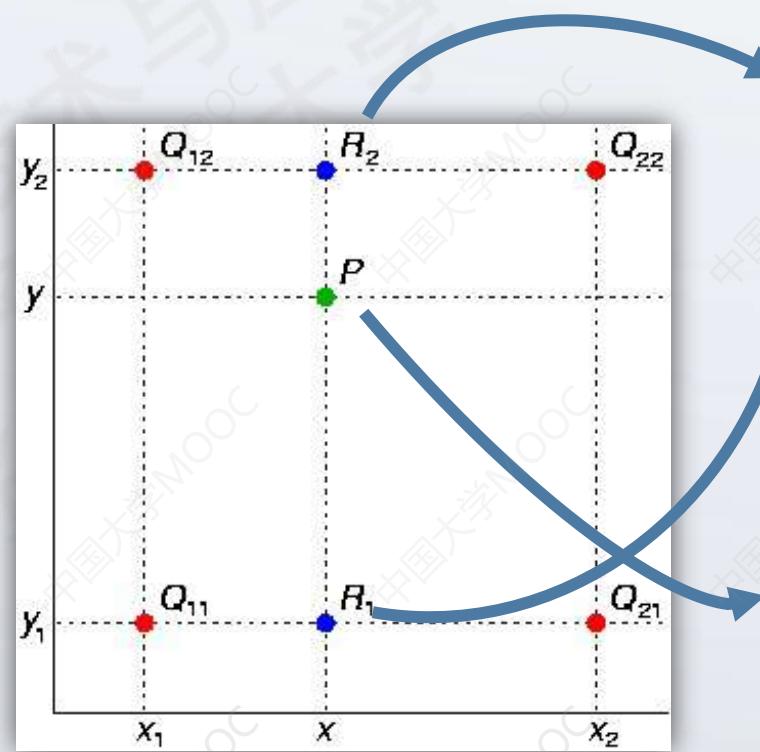
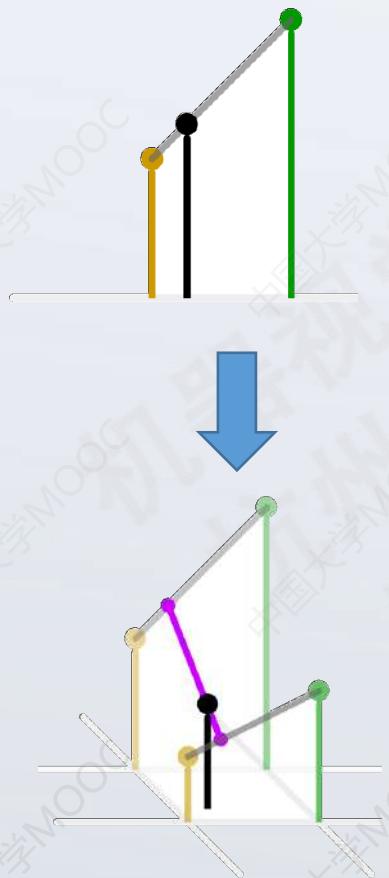
$$f(x_3) \approx \frac{x_2 - x_3}{x_2 - x_1} f(x_2) + \frac{x_3 - x_1}{x_2 - x_1} f(x_1)$$



1D的例子

# 双线性插值法

1维信号中的插值法扩展到2维，即为双线性插值法。



$$f(R_2) = f(x, y_2) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22})$$

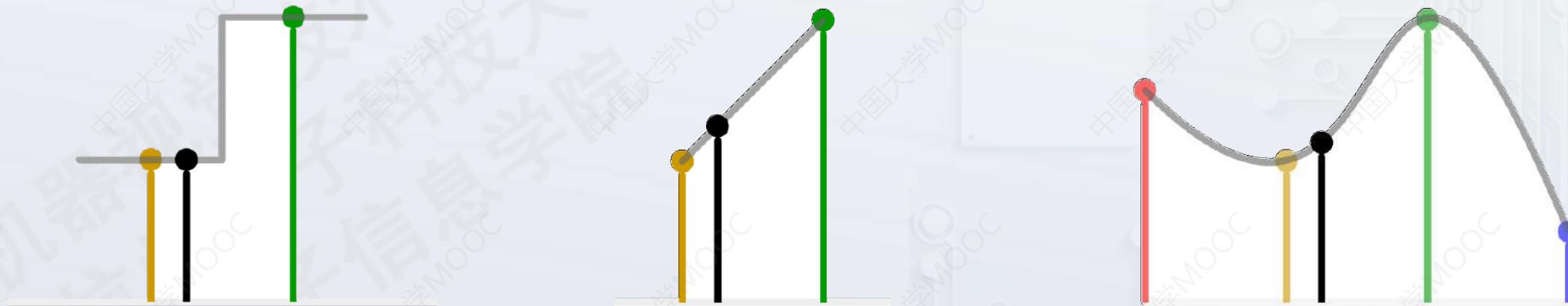
$$f(R_1) = f(x, y_2) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21})$$

代入

$$f(P) = f(x, y_1) \approx \frac{y_2 - y}{y_2 - y_1} f(x, y_2) + \frac{y - y_1}{y_2 - y_1} f(x, y_1)$$

# 3次插值法

3次插值法，通过3次多项式曲线逼近原函数。



只考虑最近的一点



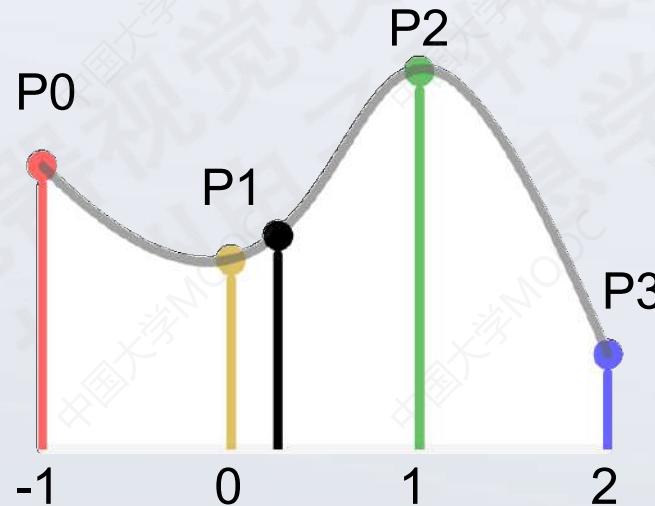
考虑最近的两点，  
直线拟合原函数。



考虑最近的四点，  
曲线拟合原函数。

# 3次插值法

假设，我们要对0和1之间的一点进行插值运算。我们通过3项式去逼近原函数。



3项式曲线方程，及其导数：

$$f(x) = ax^3 + bx^2 + cx + d$$

$$f'(x) = 3ax^2 + 2bx + c$$



$$f(0) = d$$

$$f(1) = a + b + c + d$$

$$f'(0) = c$$

$$f'(1) = 3a + 2b + c$$



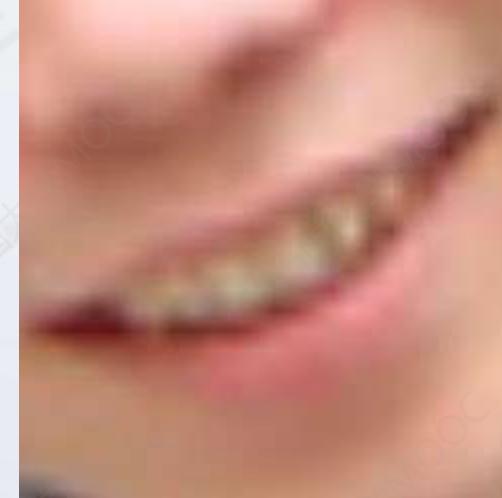
$$f(0) = p_1 \quad f(1) = p_2$$

$$f'(0) = \frac{p_2 - p_0}{2} \quad f'(1) = \frac{p_3 - p_1}{2}$$

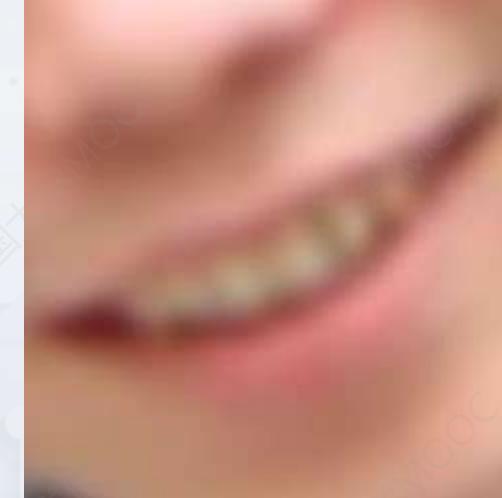
# 3次插值法



nearest



bilinear



bicubic

# 谢谢！

# 机器视觉技术与应用

## 8. 图像特征

李竹

杭州电子科技大学

电子信息学院



# 本章概要

1. 颜色特征
2. 形状特征
3. 纹理特征
4. 角点
5. SIFT、HOG

# 图像特征

图像特征描述，即通过数学方法来描述图像的属性或特点。是图像识别，图像分析等任务中的一个基本问题。

- 传统方法，通过人工设计数学方法来描述图像的特征。
- 深度学习，通过训练数据自动提取特征。

但是，人工设计的特征量在很多视觉任务中仍然起到很重要的作用。

图像特征的选择：

- 可区别性（能够反映图像的本质特点）
- 鲁棒性（旋转，光照，尺寸等）
- 维度（可区别性和鲁棒性的平衡）

# 特征选择



如何区分3种水果？

颜色

形状

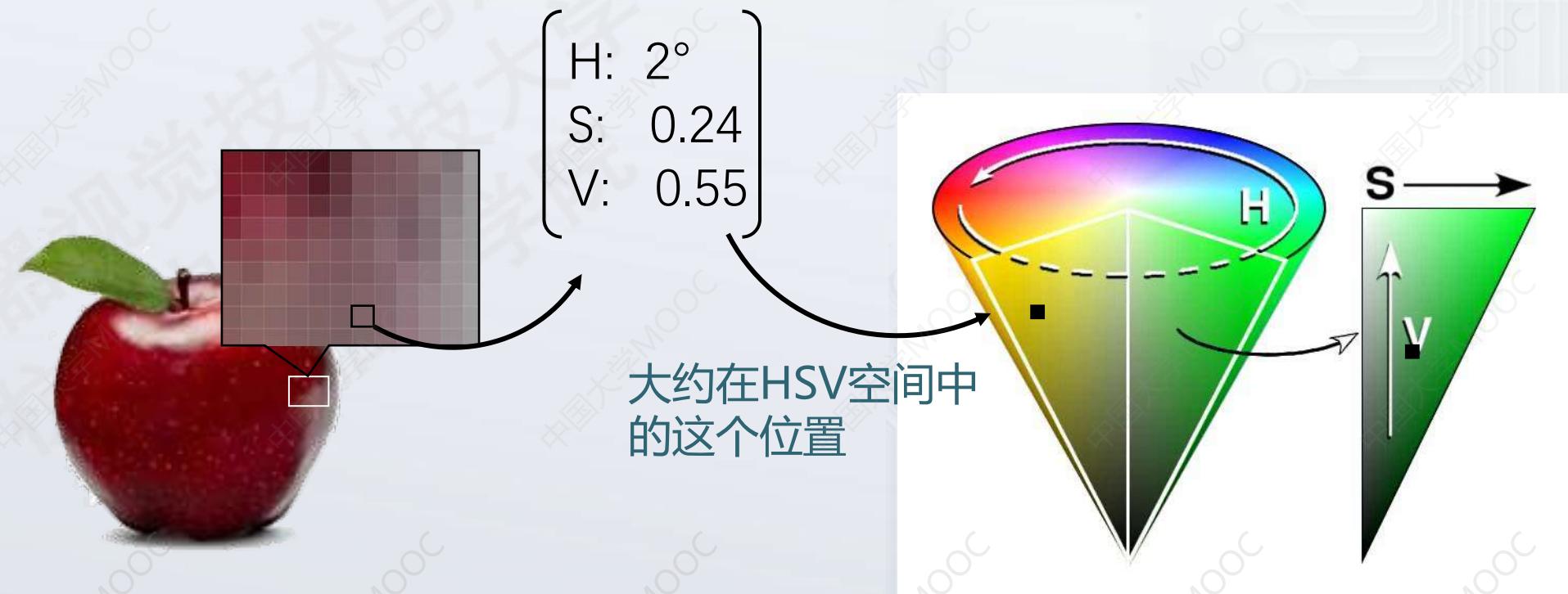
重量

具有可区分性  
需要对其量化

基本不具有可  
区分性

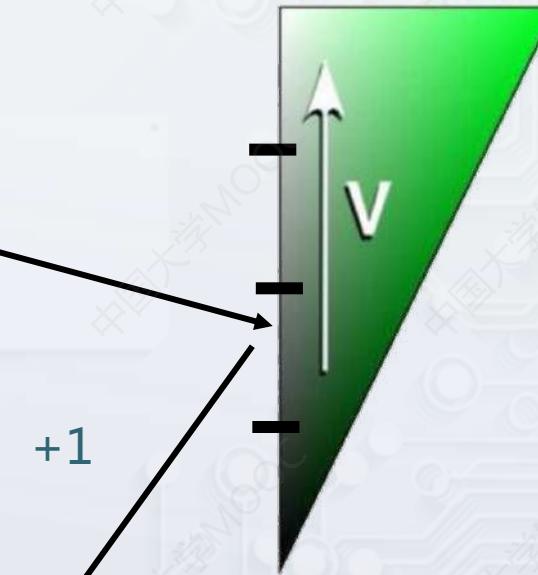
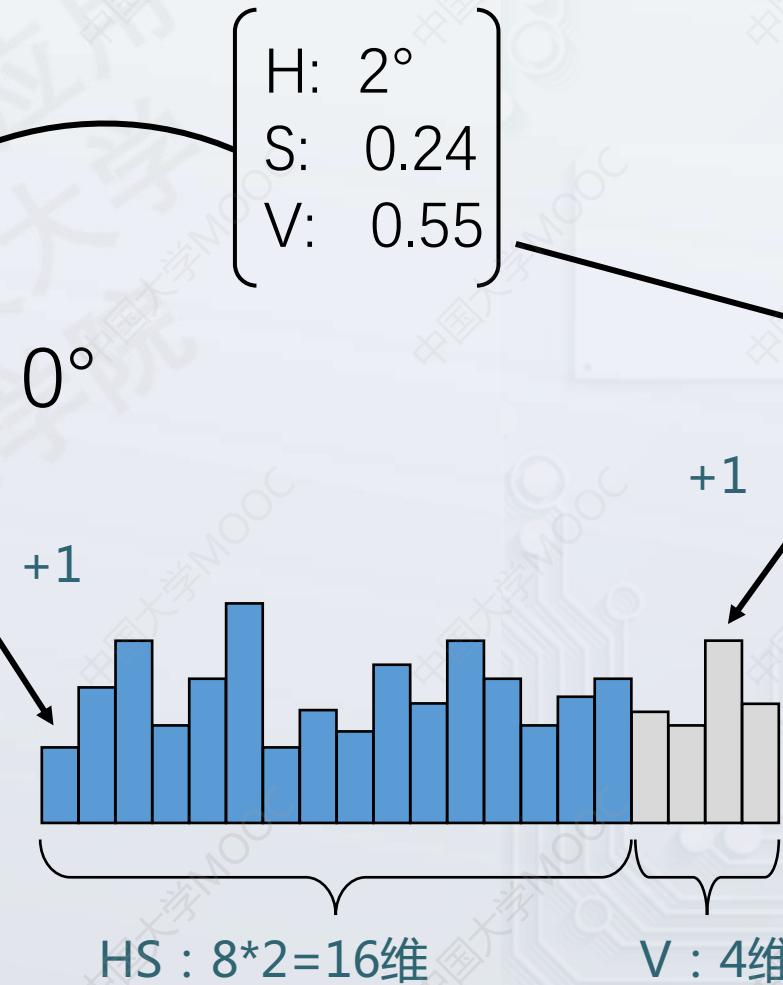
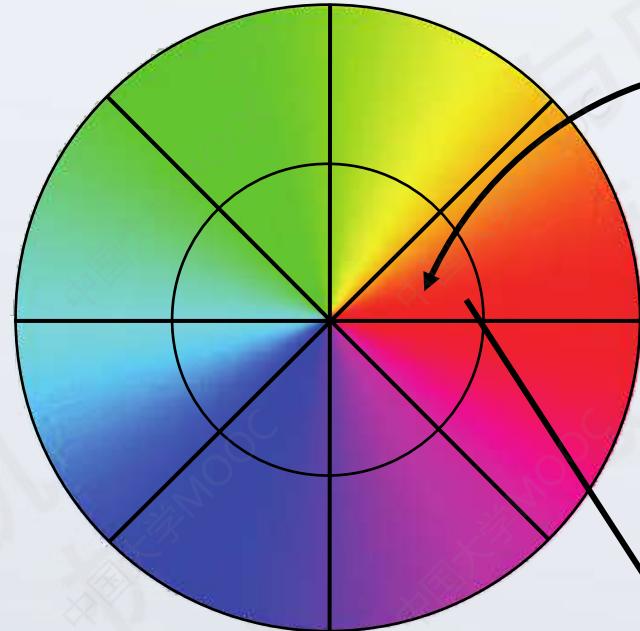
# HSV直方图

HSV特征，将图像所有像素的RGB值转化为HSV格式，并将HSV值量化后一维展开。



# HSV直方图

色调 ( H ) 8等分量化，饱和度 ( S ) 2等分量化，亮度 ( V ) 4等分量化。



# 形状特征

区域特征：计算连通域形状特征。

1. 面积：连通域像素总数
2. 周长：边界像素的总数，或相邻边界像素的距离总和。
3. 圆形度：如果F是区域的面积，max是从中心到所有轮廓像素的最大距离，则圆形度定义为：

$$C = \frac{F}{(\pi * \text{max}^2)}$$



1.00



0.81



0.63



0.44



0.26



0.07

# 形状特征

4. 紧凑度(compactness)：如果F是连通域的面积，L是连通域的周长，则紧凑度定义为：

$$C' = \frac{L^2}{4F\pi} \quad C = \max(1, C')$$

圆的形状系数C为1.如果该区域较长或具有孔，则C大于1。



# 形状特征

## 5. 矩形度：最小外接矩形和连通域面积的比值



1.00



0.84



0.69



0.53



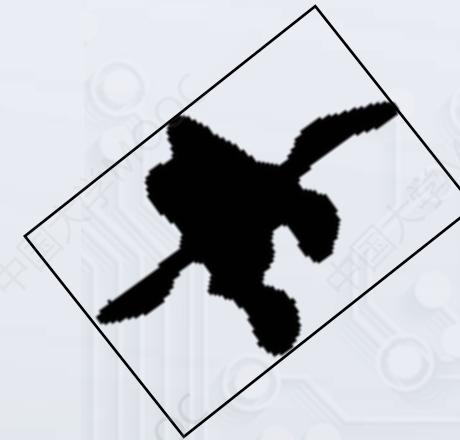
0.38



0.22



外接矩形



最小外接矩形

# 形状特征

## 6. 宽长比：长轴与短轴的比值



6.61



5.41



4.36



3.23



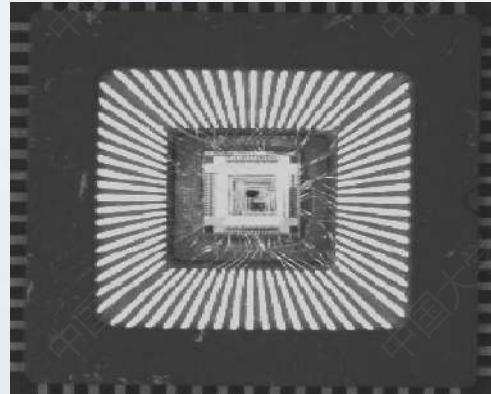
2.12



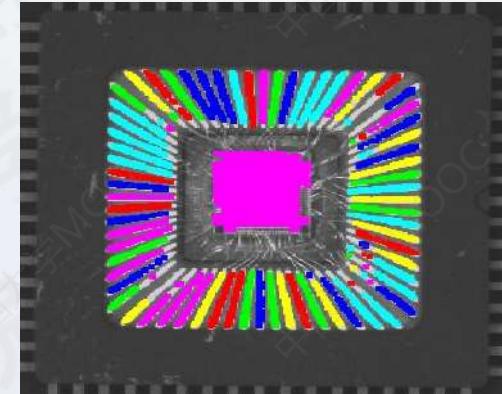
1.00

# 形状特征

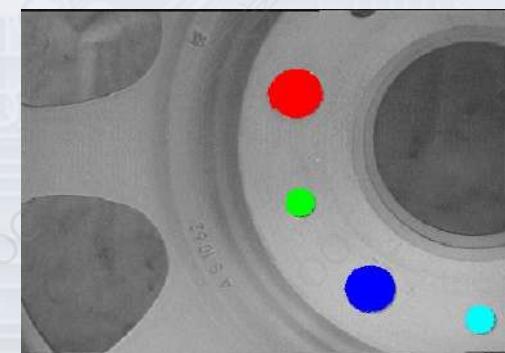
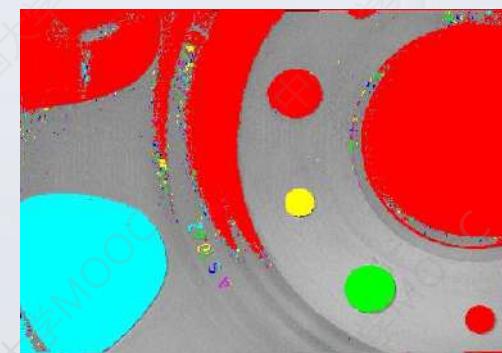
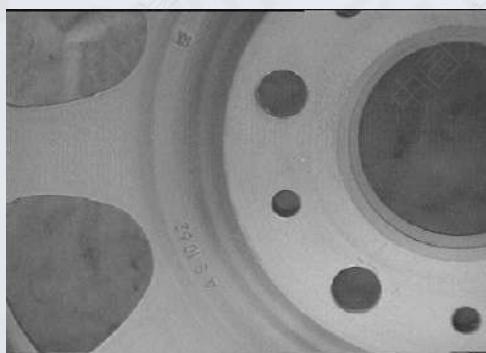
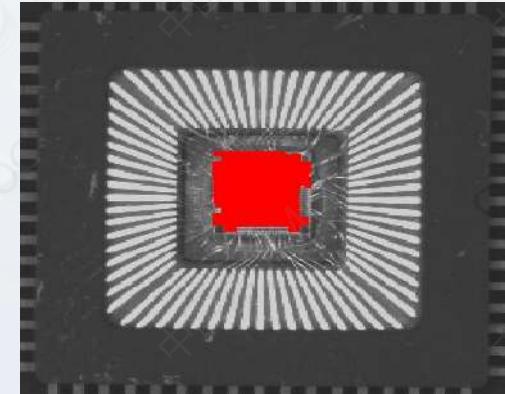
原图



二值化即连通域标记

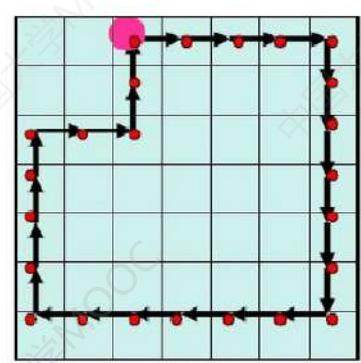
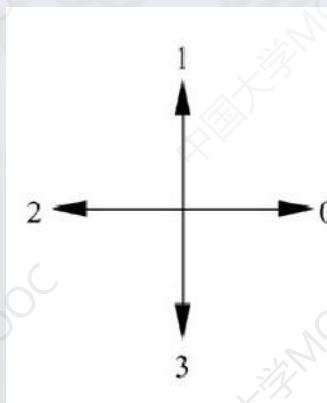
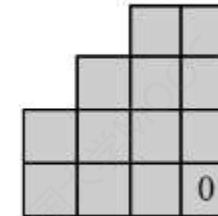
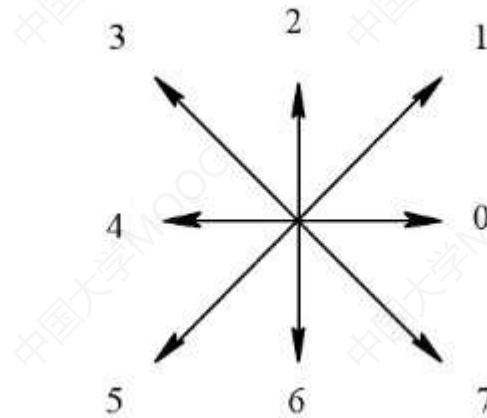
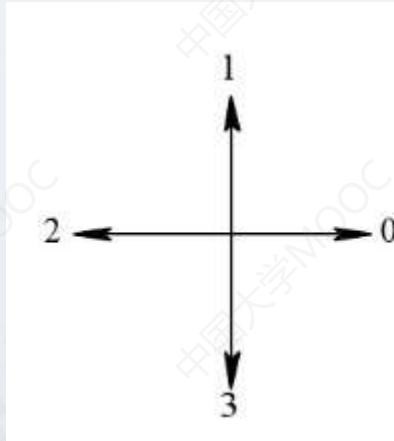


特征选择



# 形状特征

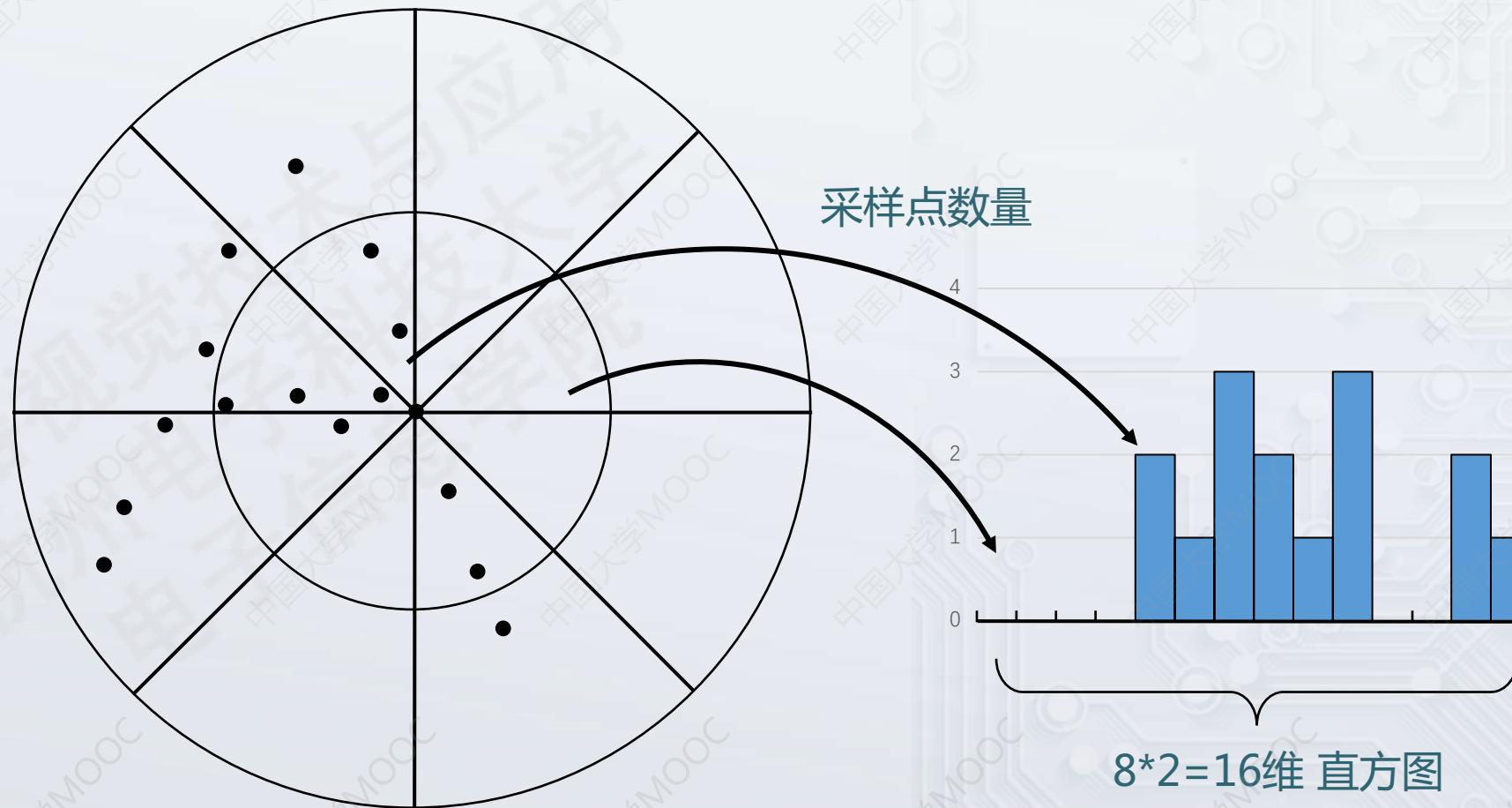
## 链表：描述外接轮廓的形状



000033333322222211110011

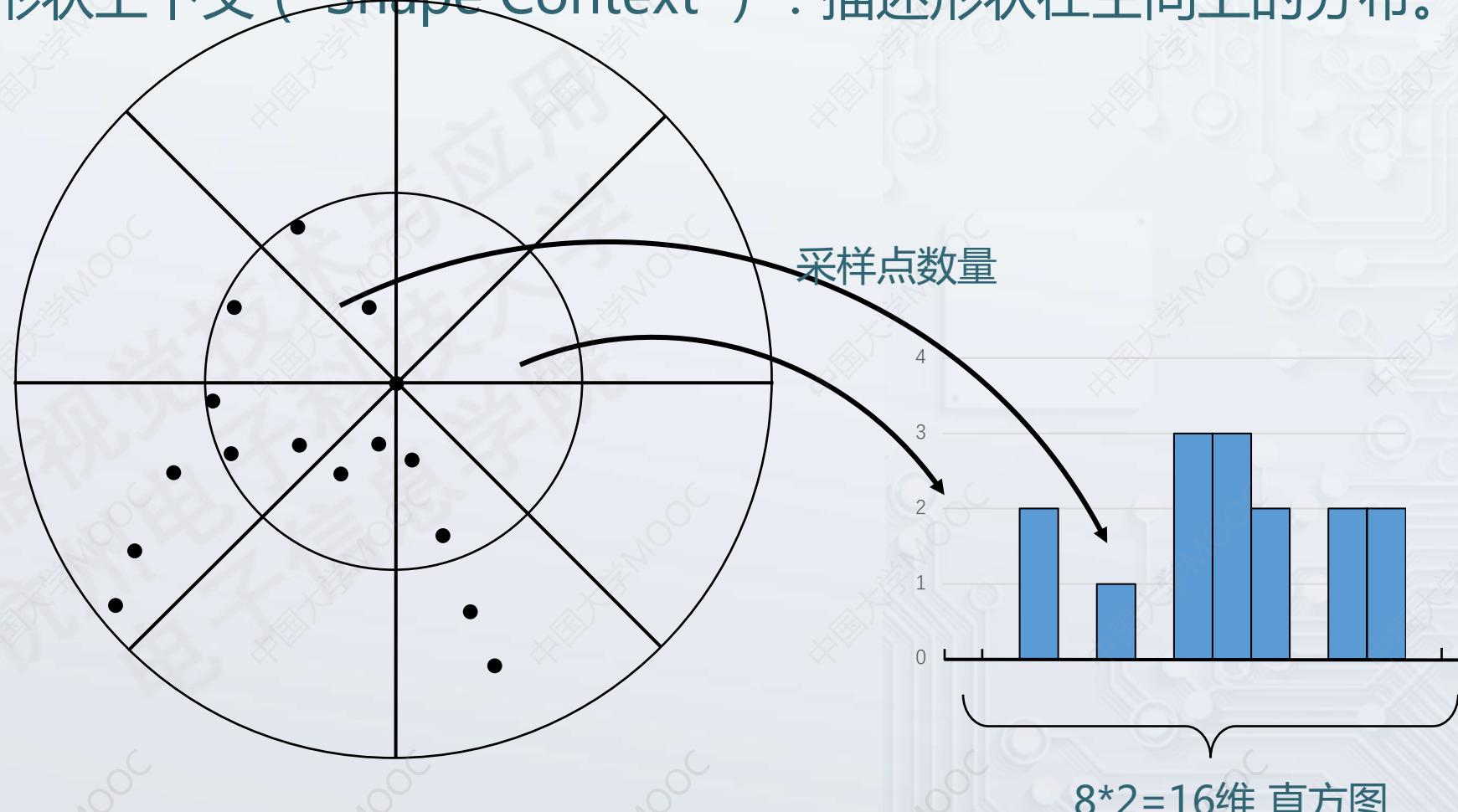
# 形状特征

形状上下文（ Shape Context ）：描述形状在空间上的分布。



# 形状特征

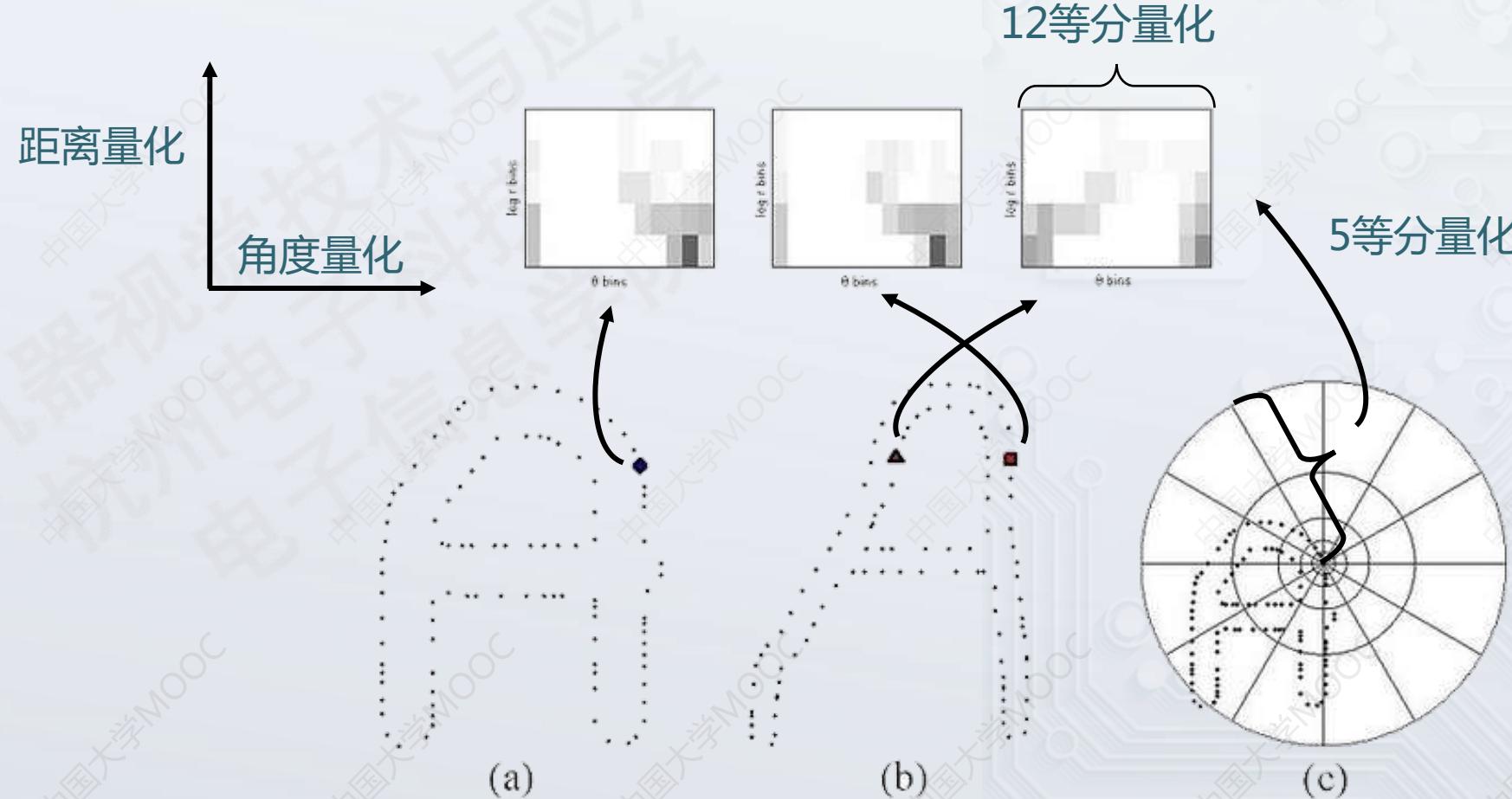
形状上下文（ Shape Context ）：描述形状在空间上的分布。



# 形状特征

S. Belongie; J. Malik & J. Puzicha (April 2002). "Shape Matching and Object Recognition Using Shape Contexts" (PDF). IEEE Transactions on Pattern Analysis and Machine Intelligence. 24 (24): 509–521.  
doi:10.1109/34.993558

原文使用了2维直方图。颜色深浅表示值的大小。从直方图中可以看到，相对位置接近的采样点的直方图，分布也相似。

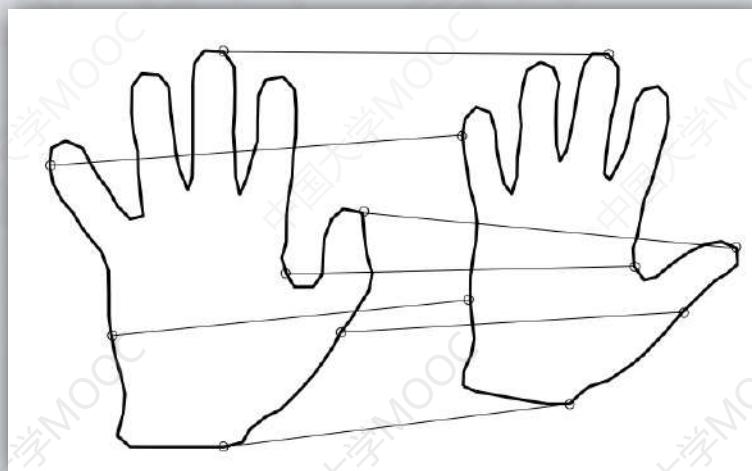


# 形状特征

对两个形状，假设分别有N个采样点。构成 $N \times N$ 的cost matrix。

Cost matrix的每个成分即为点与点的shape context直方图之间的相似度。转化一次指派问题，对点进行匹配。匹配的总cost即为形状的总相似度。一次指派问题可通过匈牙利算法求解。

## 形状A



形状B

## 形状B

形状A

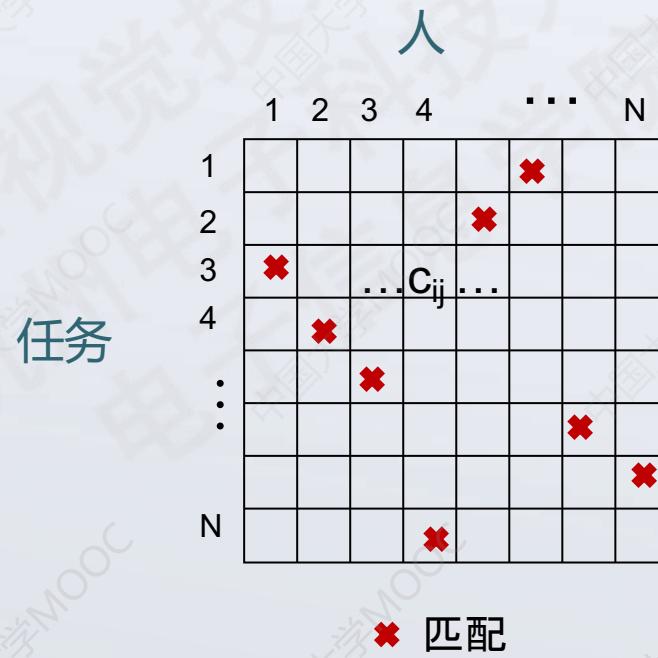
	1	2	3	4	...	N
1						$\times$
2					$\times$	
3	$\times$			$C_{ij}$		
4		$\times$				
⋮	⋮		$\times$			$\times$
N				$\times$		$\times$

## ✖ 匹配

# 形状特征

一次指派问题 ( Assignment problem )。指派N个人去做N件事。

已知第*i*人做第*j*件事的费用为 $c_{ij}$  ( $i,j=1,2,\dots,N$ )。则如何指派任务，总消费最小。



Minimize

$$z = \sum_{i=1}^N \sum_{j=1}^N c_{ij} x_{ij},$$

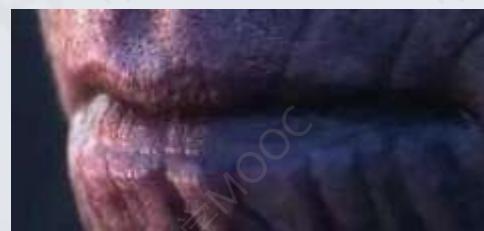
subject to

$$\begin{cases} \sum_{j=1}^N x_{ij} = 1, i = \{1, 2, \dots, N\}, \\ \sum_{j=1}^N x_{ij} = 1, j = \{1, 2, \dots, N\} \end{cases}$$

$$x_{ij} = \{0, 1\}$$

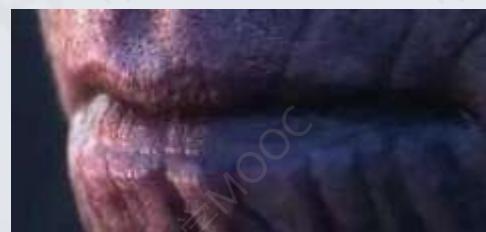
# 图像的局部特征

很多应用场景下，不需要对图像的全局进行描述就可以进行图像匹配。  
对图像局部进行特征描述的描述算子成为局部特征。



# 图像的局部特征

局部特征对遮挡具有一定的鲁棒性。



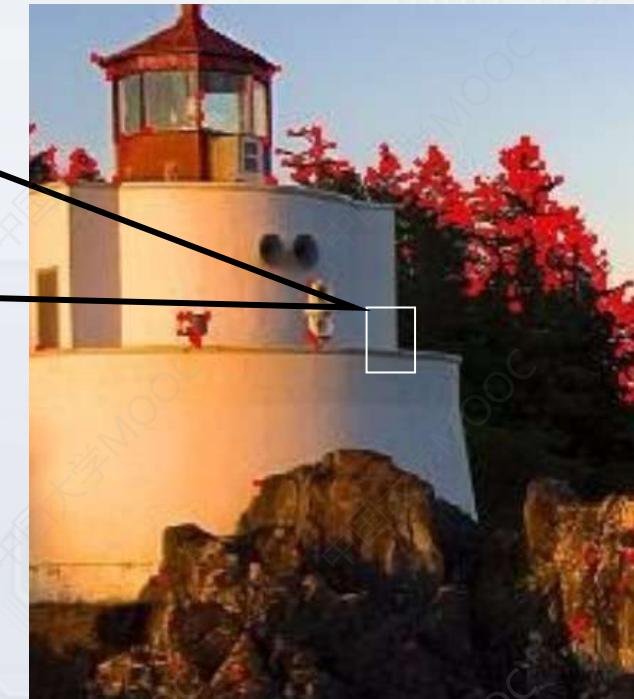
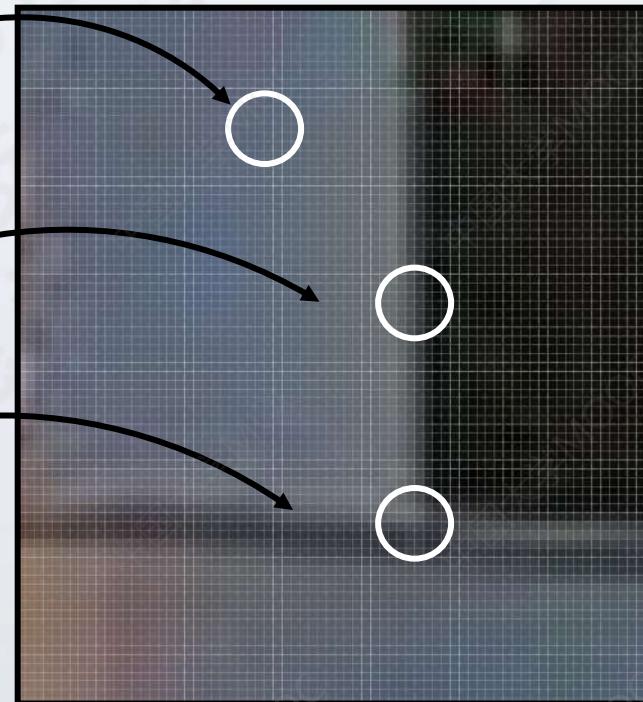
# 角点检测

图像中，哪些位置的像素具有较强的特征？？

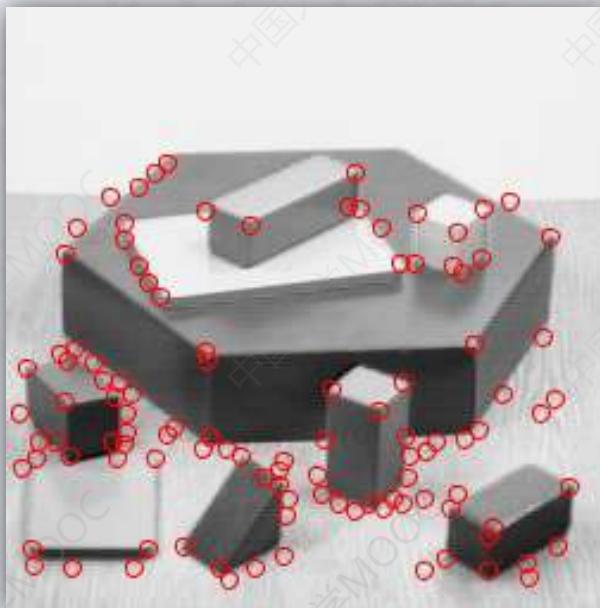
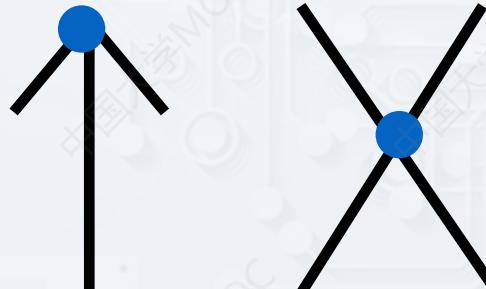
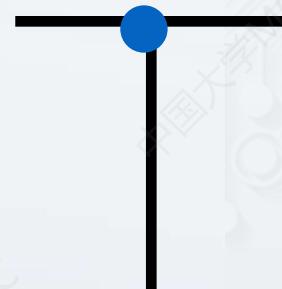
亮度平滑变化或  
近似恒定区域的  
像素

边缘上的像素

角落的像素



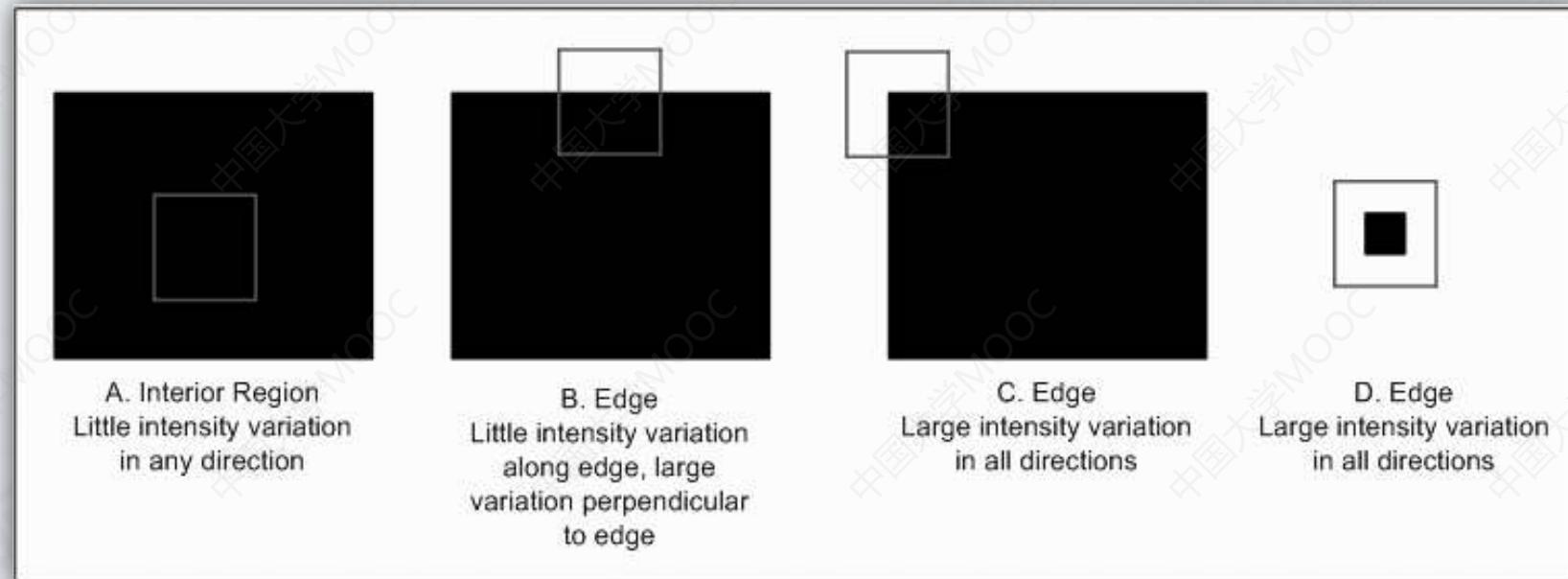
# 角点检测



角点通常是目标轮廓上曲率的局部极大值，对掌握目标的轮廓特征具有决定作用。

通常认为角点是二维图像亮度变化剧烈的点，或两条线的交叉处

# 角点检测

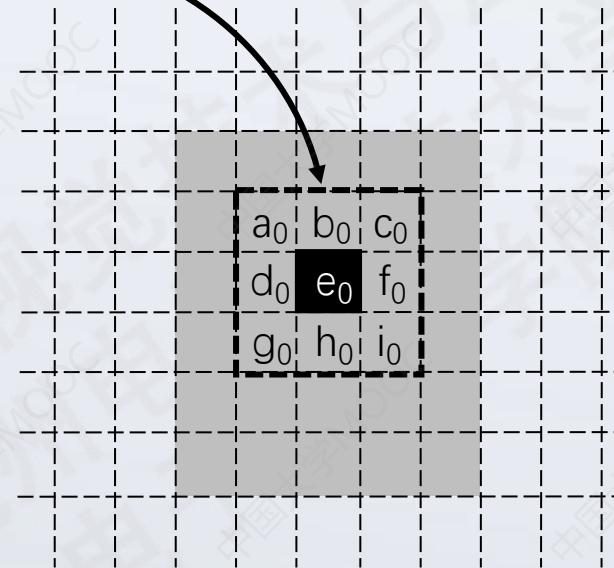


角点检测算法基本思想是使用一个固定窗口（取某个像素的一个邻域窗口）在图像上进行任意方向上的滑动，比较滑动前与滑动后两种情况，窗口中的像素灰度变化程度，如果存在任意方向上的滑动，都有着较大灰度变化，那么我们可以认为该窗口中存在角点。

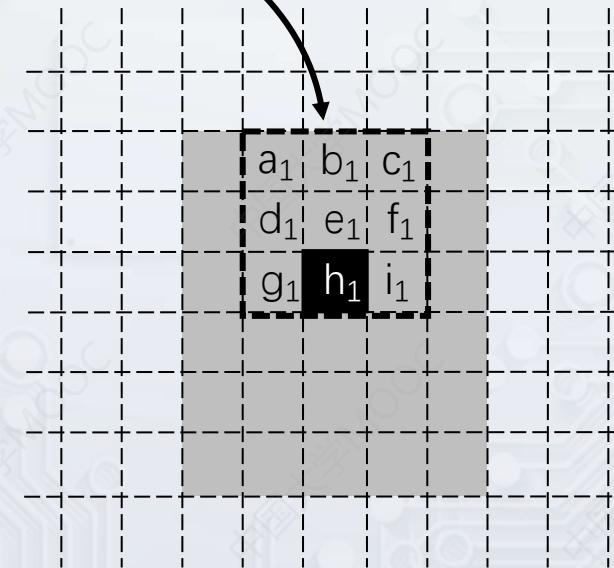
# 角点检测

考察该像素是否为角点，在该像素周围设置 $3\times 3$ 滑动窗口，计算SSD(sum of squared differences)。

滑动窗口



滑动窗口

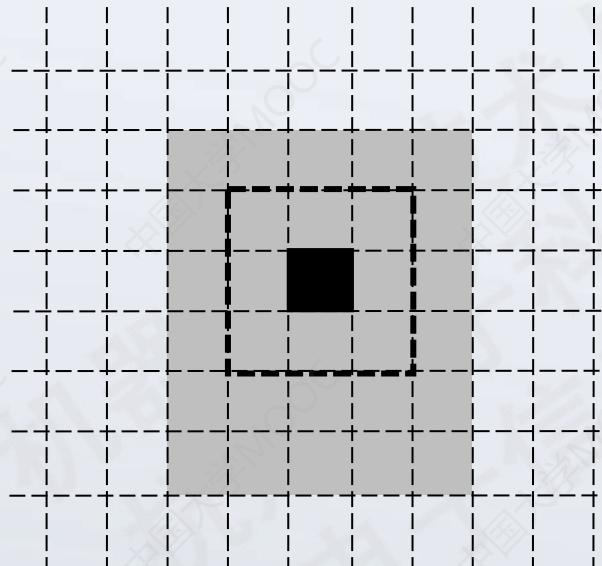


$$F = (a_0 - a_1)^2 + (b_0 - b_1)^2 + (c_0 - c_1)^2 + (d_0 - d_1)^2 + \\ (e_0 - e_1)^2 + (f_0 - f_1)^2 + (g_0 - g_1)^2 + (h_0 - h_1)^2 + (i_0 - i_1)^2$$

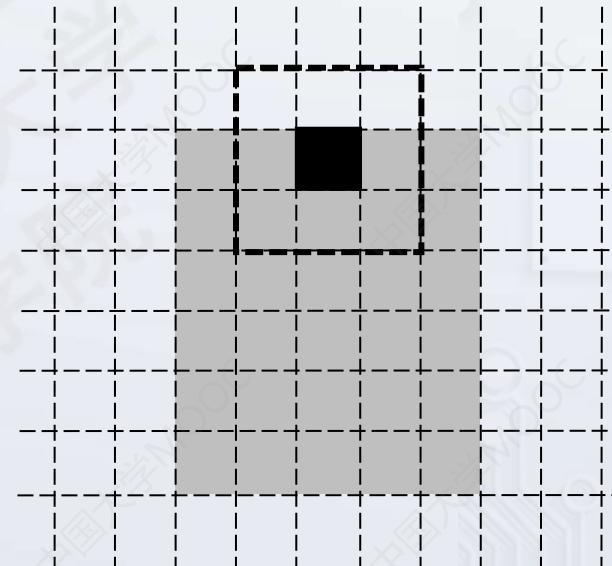
分别计算其在水平、垂直、左对角线和右对角线方向上的灰度变化值平方和，并把其中**最小值**的称为该象素点的灰度变化特征值。

# 角点检测

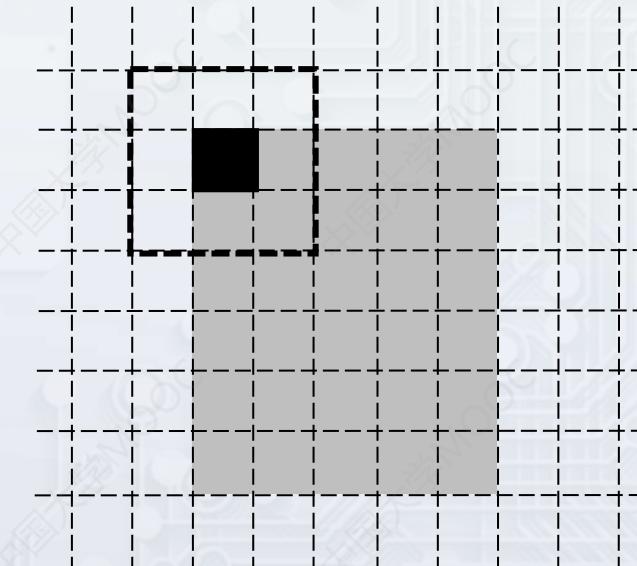
灰度变化特征值最大的像素即为角点。只检测了窗口函数在8个基本方向上移动的强度变化，不能准确提取出全部角点



任何方向移动滑窗，  
灰度变化值都为0。



左右移动滑窗，灰  
度变化值都为0。



左右移动滑窗，灰  
度变化值都较大。

# Harris角点

设图像函数为 $I$ ，像素坐标为 $(x,y)$ ，滑动窗口的滑动变量为 $(u,v)$ 。加权的SSD，即灰度变化量的数学表达式为：

$$S(u, v) = \sum_u \sum_v \omega(x, y) (I(x + u, y + v) - I(x, y))^2$$

↑                      ↑                      ↑  
加权函数或窗口    像素 $(x,y)$ 周围    像素 $(x,y)$ 的亮度值  
函数                  像素的亮度值

$$\sum_u \sum_v \omega(x, y) =$$



# Harris角点

$$S(u, v) = \sum_u \sum_v \omega(x, y) \left[ \frac{I(x+u, y+v) - I(x, y)}{I(x, y)} \right]^2$$

泰勒分解

$$\begin{aligned} I(x+u, y+v) &= I(x, y) + I_x(x, y) \cdot u + I_y(x, y) \cdot v + \dots [R_n(x, y)] \\ &\approx [I(x, y) + I_x(x, y) \cdot u + I_y(x, y) \cdot v] \end{aligned}$$

高次项，省略

代入原式

$$\begin{aligned} S(u, v) &= \sum_u \sum_v \omega(x, y) \left[ \frac{(I(x, y) + I_x(x, y) \cdot u + I_y(x, y) \cdot v) - I(x, y)}{I(x, y)} \right]^2 \\ &= \sum_u \sum_v \omega(x, y) (I_x(x, y) \cdot u + I_y(x, y) \cdot v)^2 \end{aligned}$$

# Harris角点

$$\begin{aligned} S(u, v) &\approx \sum_u \sum_v \omega(x, y) (I_x(x, y) \cdot u + I_y(x, y) \cdot v)^2 \\ &= \sum_u \sum_v \omega(x, y) \left[ (I_x(x, y) \cdot u)^2 + 2 \cdot I_x(x, y) \cdot u \cdot I_y(x, y) \cdot v + (I_y(x, y) \cdot v)^2 \right] \end{aligned}$$

矩阵形式



$$S(u, v) \approx [u, v] \sum_u \sum_v \omega(u, v) \begin{bmatrix} I_x(x, y)^2 & I_x(x, y)I_y(x, y) \\ I_x(x, y)I_y(x, y) & I_y(x, y)^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

# Harris角点

$$\begin{aligned} S(u, v) &\approx \sum_u \sum_v \omega(x, y) (I_x(x, y) \cdot u + I_y(x, y) \cdot v)^2 \\ &= \sum_u \sum_v \omega(x, y) [(I_x(x, y) \cdot u)^2 + 2 \cdot I_x(x, y) \cdot u \cdot I_y(x, y) \cdot v + (I_y(x, y) \cdot v)^2] \end{aligned}$$

矩阵形式



$$S(u, v) \approx [u, v] \sum_u \sum_v \omega(u, v) \begin{bmatrix} I_x(x, y)^2 & I_x(x, y)I_y(x, y) \\ I_x(x, y)I_y(x, y) & I_y(x, y)^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

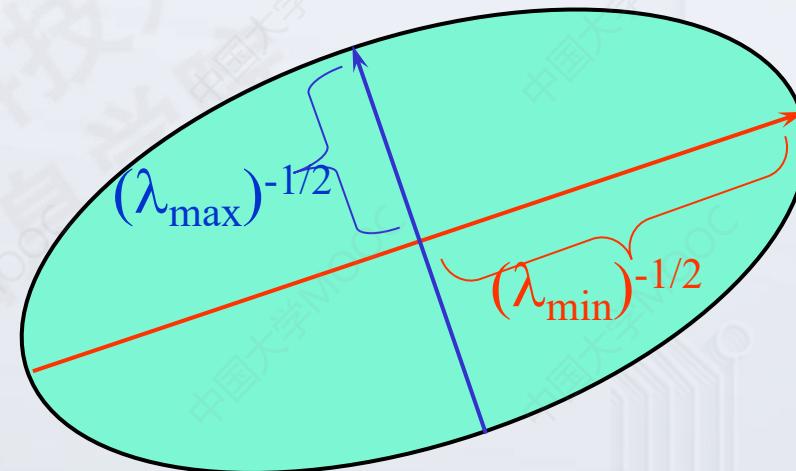
海森矩阵

$$M = \sum_u \sum_v \omega(u, v) \begin{bmatrix} I_x(x, y)^2 & I_x(x, y)I_y(x, y) \\ I_x(x, y)I_y(x, y) & I_y(x, y)^2 \end{bmatrix} = \begin{bmatrix} A & C \\ C & B \end{bmatrix}$$

# Harris角点

$$S(u, v) = \sum_u \sum_v \omega(x, y)(I(x+u, y+v) - I(x, y))^2 \approx Au^2 + 2Cuv + Bv^2$$

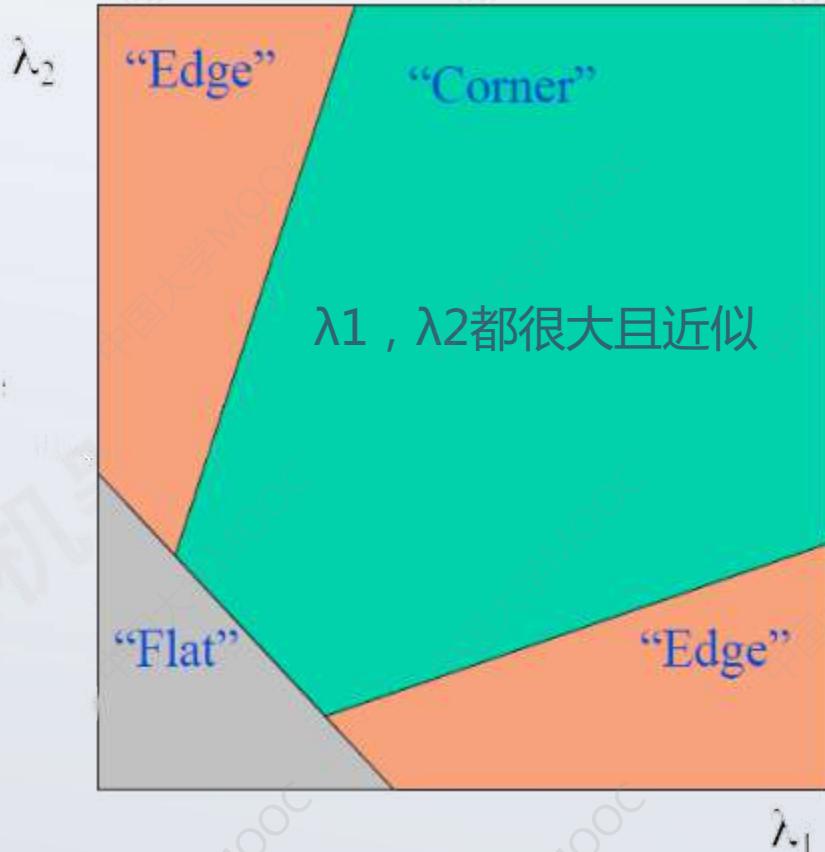
可以看做椭圆公式.  $\lambda_{max}$  和  $\lambda_{min}$  为海森矩阵的特征值.对于二维图像的某点的hessian矩阵，其最大特征值和其对应的特征向量对应其邻域二维曲线最大曲率的强度和方向，即山坡陡的那面；最小特征值对应的特征向量对应与其垂直的方向，即平缓的方向。



Hessian matrix  $M = \sum_u \sum_v \omega(u, v) \begin{bmatrix} I_x(x, y)^2 & I_x(x, y)I_y(x, y) \\ I_x(x, y)I_y(x, y) & I_y(x, y)^2 \end{bmatrix} = \begin{bmatrix} A & C \\ C & B \end{bmatrix}$

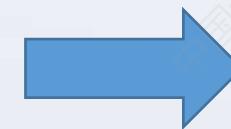
# Harris角点

$\lambda_2$ 远大于 $\lambda_1$



$\lambda_1, \lambda_2$ 都很小且近似

$\lambda_1$ 远大于 $\lambda_2$



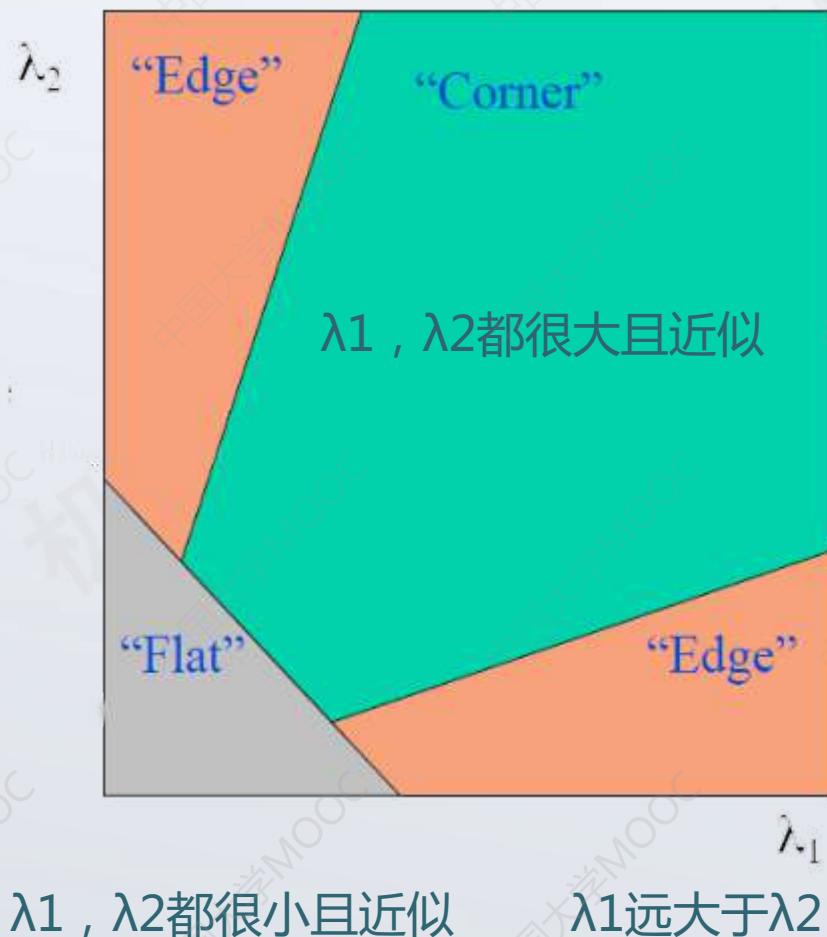
问题

$$M = \sum_u \sum_v \omega(u, v) \begin{bmatrix} I_x(x, y)^2 & I_x(x, y)I_y(x, y) \\ I_x(x, y)I_y(x, y) & I_y(x, y)^2 \end{bmatrix} = \begin{bmatrix} A & C \\ C & B \end{bmatrix}$$

计算量大，对每个像素都需要  
计算海森矩阵的特征值。

# Harris角点

$\lambda_2$ 远大于 $\lambda_1$



$$M = \sum_u \sum_v \omega(u, v) \begin{bmatrix} I_x(x, y)^2 & I_x(x, y)I_y(x, y) \\ I_x(x, y)I_y(x, y) & I_y(x, y)^2 \end{bmatrix} = \begin{bmatrix} A & C \\ C & B \end{bmatrix}$$



进一步定义  
响应函数

$$R = \det M - k(\text{trace}M)^2$$

$$\det M = \lambda_1 \lambda_2 = AC - B$$

$$\text{trace}M = \lambda_1 + \lambda_2 = A + C$$

只需要计算水平和垂直方向的差分即可得到响应函数。

# Harris角点

$$M = \sum_u \sum_v \omega(u, v) \begin{bmatrix} I_x(x, y)^2 & I_x(x, y)I_y(x, y) \\ I_x(x, y)I_y(x, y) & I_y(x, y)^2 \end{bmatrix} = \begin{bmatrix} A & C \\ C & B \end{bmatrix}$$

$$\det M = \lambda_1 \lambda_2 = AC - B \quad \text{trace} M = \lambda_1 + \lambda_2 = A + C$$

$$R = \det M - k(\text{trace} M)^2 = \lambda_1 \lambda_2 - k(\lambda_1 + \lambda_2)^2$$

对于边缘来说，假设  $\lambda_1 \gg \lambda_2$ ，且  $k\lambda_1 > \lambda_2$ ，则

$$R = \lambda_1 \lambda_2 - k(\lambda_1 + \lambda_2)^2 \approx \lambda_1 \lambda_2 - k\lambda_1^2 < k\lambda_1 \lambda_1 - k\lambda_1^2 = 0$$

该像素位边缘的条件：  $R < 0$

# Harris角点

$$M = \sum_u \sum_v \omega(u, v) \begin{bmatrix} I_x(x, y)^2 & I_x(x, y)I_y(x, y) \\ I_x(x, y)I_y(x, y) & I_y(x, y)^2 \end{bmatrix} = \begin{bmatrix} A & C \\ C & B \end{bmatrix}$$

$$\det M = \lambda_1 \lambda_2 = AC - B \quad \text{trace} M = \lambda_1 + \lambda_2 = A + C$$

$$R = \det M - k(\text{trace} M)^2 = \lambda_1 \lambda_2 - k(\lambda_1 + \lambda_2)^2$$

对于非边缘像素来说，假设  $\lambda_1$  与  $\lambda_2$  近似相等，则

$$R = \lambda_1 \lambda_2 - k(\lambda_1 + \lambda_2)^2 \approx \lambda_1 \lambda_2 - k4\lambda_1 \lambda_2 = (1 - 4k)\lambda_1 \lambda_2 \approx \lambda_1 \lambda_2$$

如果 R 很大，则为角点，如果 R 很小，则为平缓区域像素。

# Harris角点

$$M = \sum_u \sum_v \omega(u, v) \begin{bmatrix} I_x(x, y)^2 & I_x(x, y)I_y(x, y) \\ I_x(x, y)I_y(x, y) & I_y(x, y)^2 \end{bmatrix} = \begin{bmatrix} A & C \\ C & B \end{bmatrix}$$

$$R = \det M - k(\text{trace}M)^2$$

$$\det M = \lambda_1 \lambda_2 = AC - B$$

$$\text{trace}M = \lambda_1 + \lambda_2 = A + C$$

R is big : corner

R<0 : edge

R is small : flat

优点：避免了较为复杂的特征值计算，通过一个数值即可判断像素是否为角点。

# SIFT

SIFT，即尺度不变特征变换（ Scale-invariant feature transform , SIFT ），一种特征描述方法。具有

尺度鲁棒性

旋转鲁棒性

光照鲁棒性

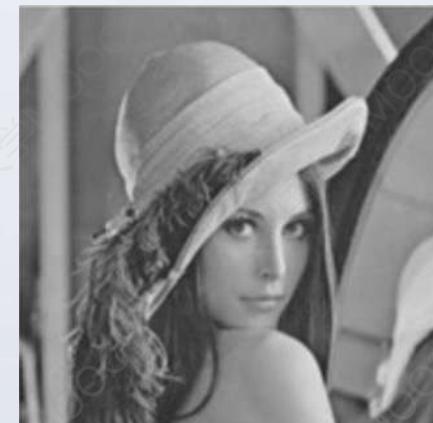
SIFT本身包括了特征点筛选及特征点描述的步骤。

# DoG特征点检出

**尺度空间理论(Scale-space theory,Lindeberg,1962)**：高斯核是唯一可以产生多尺度空间的核。人眼去看一张照片时，随着观测距离的增加，图像会逐渐变得模糊。这里的“尺度”就是二维高斯函数当中的 $\sigma$ 值，一张照片与不同 $\sigma$ 值的二维高斯函数卷积后得到很多张清晰度不同的高斯图像，这就好比你用人眼从不同距离去观测那张照片。



原图



$\sigma=0.6$



$\sigma=2$

# DoG特征点检出

**SIFT的特征点筛选目的**：寻找在不同尺度空间下的极值点，保证这些特征点在放大或者缩小的条件下均存在。

**筛选方法**：通过DoG近似LoG

LoG(Laplacian of Gaussian)：使用不同 $\sigma$ 获得不同的尺度空间图像，然后通过Laplacian算子获得极值。这一运算过程可以用DoG(Difference-of-Gaussian)近似。

# DoG特征点检出

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$$

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$$

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma)$$



原图



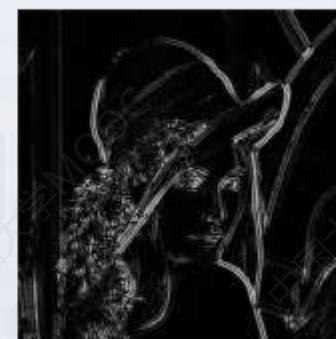
高斯模糊后的图像

$L(x, y, \sigma)$ : 高斯模糊后图像

$I(x, y)$ : 原图

$G(x, y, \sigma)$ : 高斯核函数

$D(x, y, \sigma)$ : DoG图像

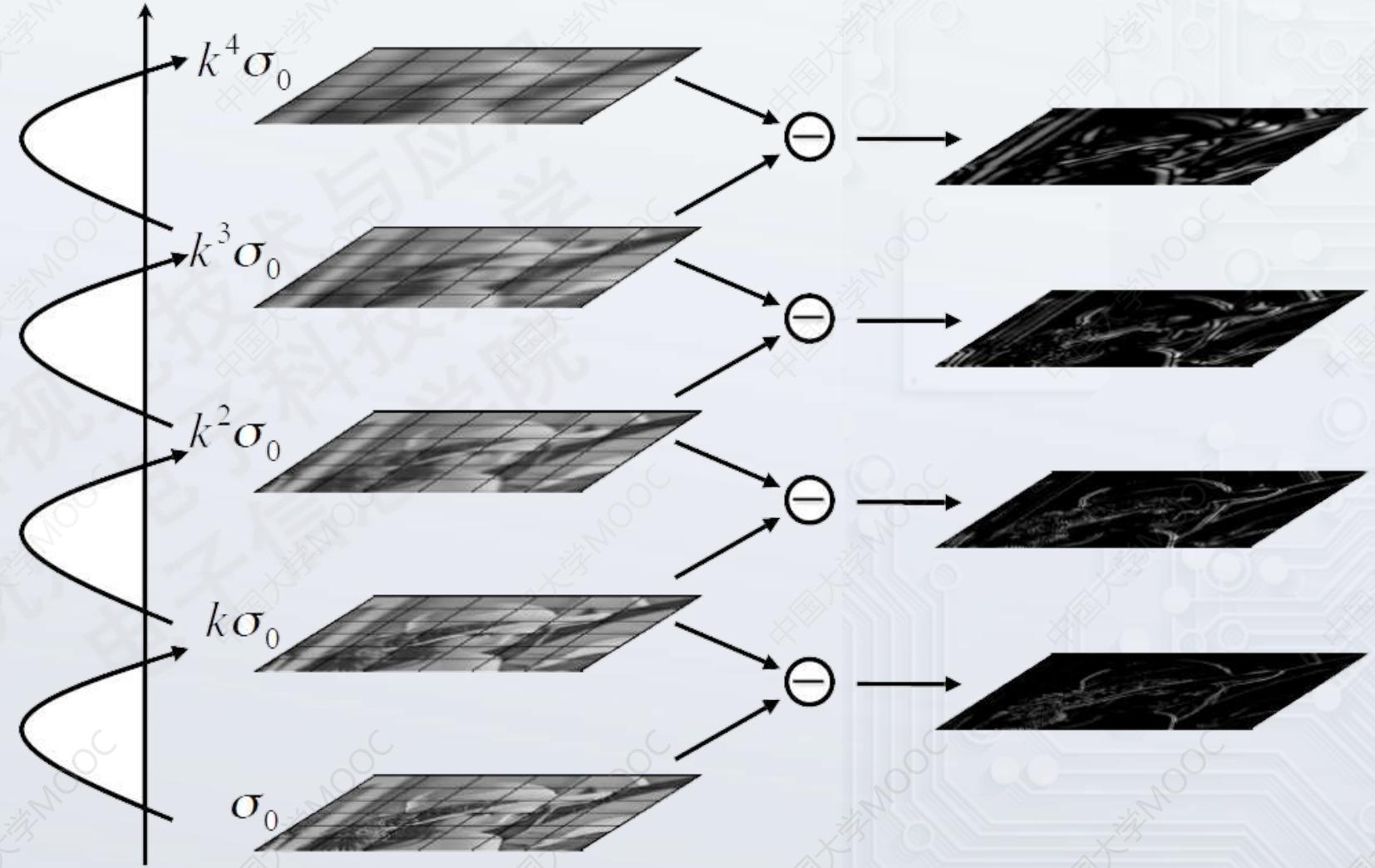


DoG



LoG

# DoG特征点检出



# DoG特征点检出

$\sigma$ 值的增加



需要用更大的高斯窗口进行平滑模糊处理。

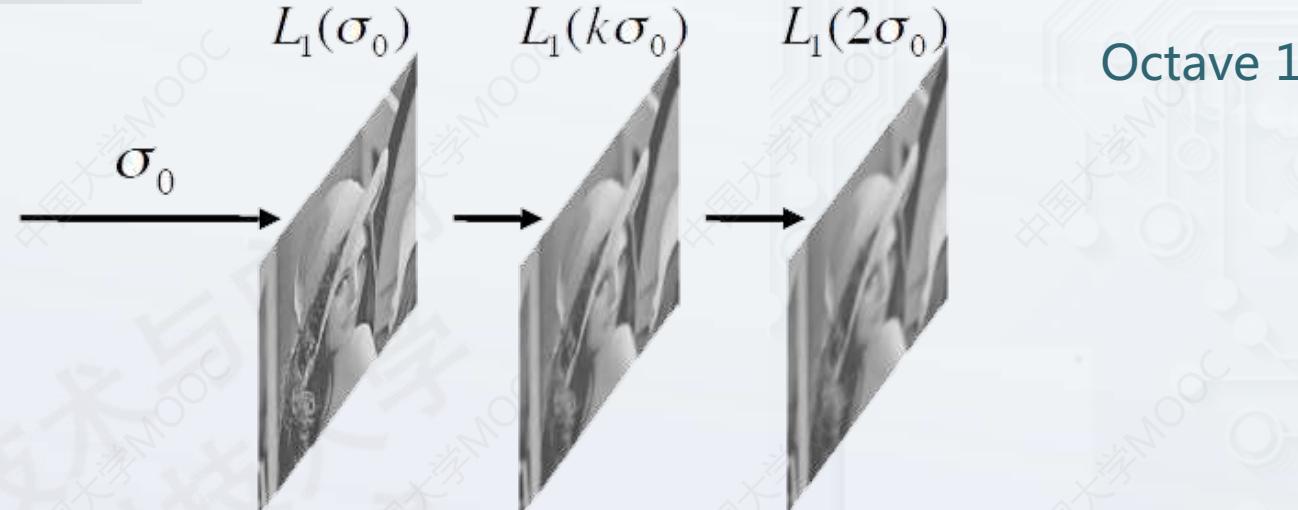
会使得无法处理的区域增加，同时增加计算量。

解决方法：图像金字塔

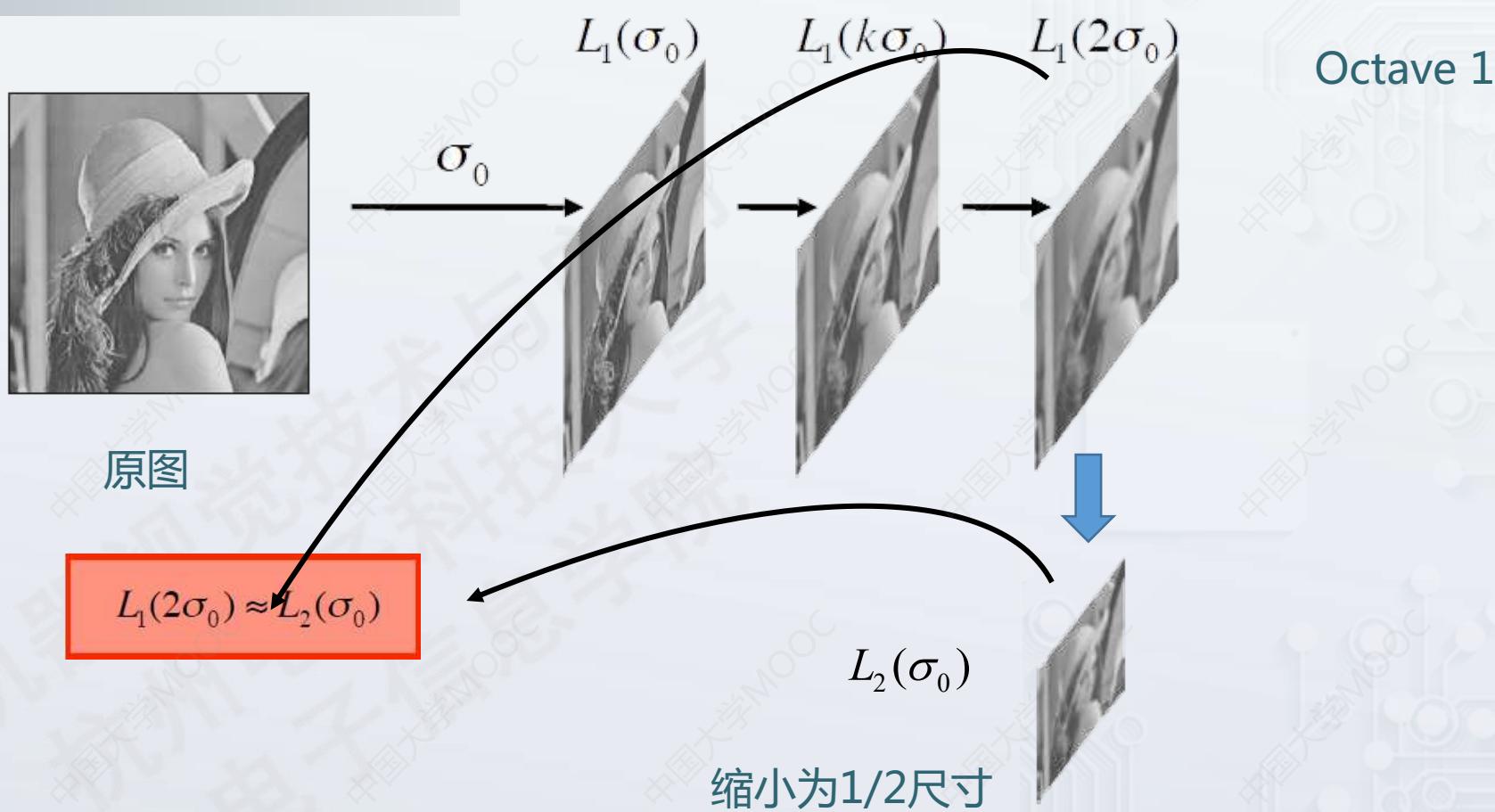
# 图像金字塔



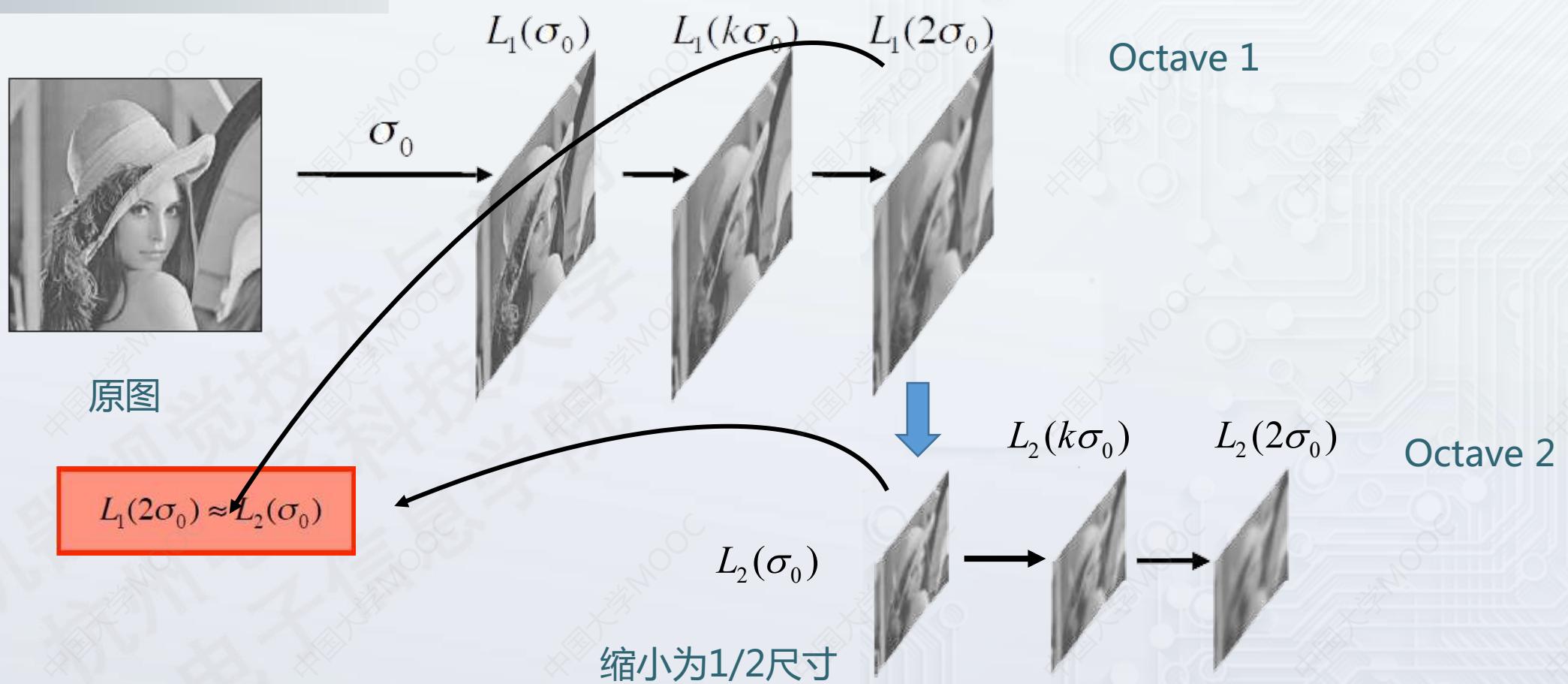
原图



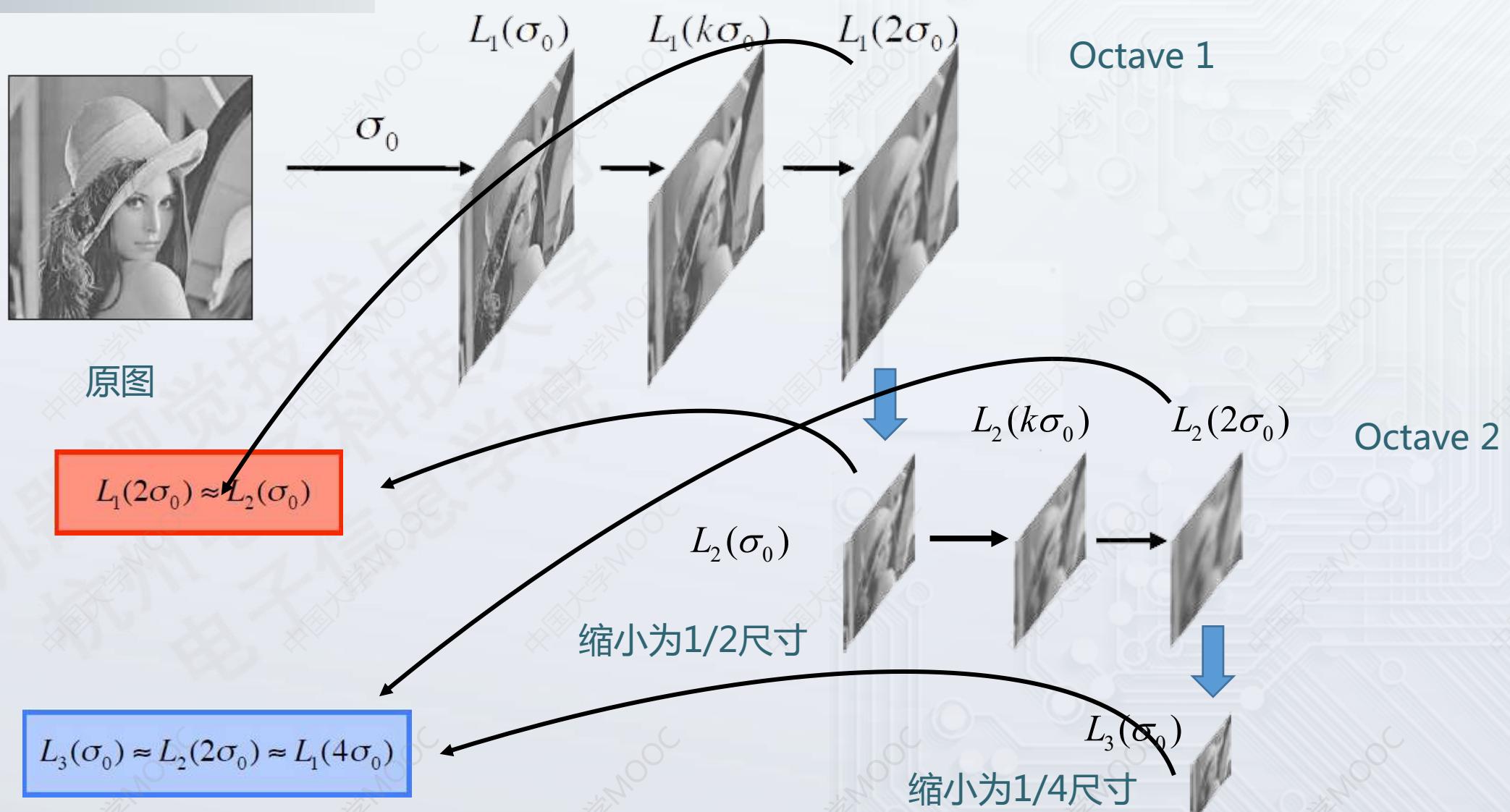
# 图像金字塔



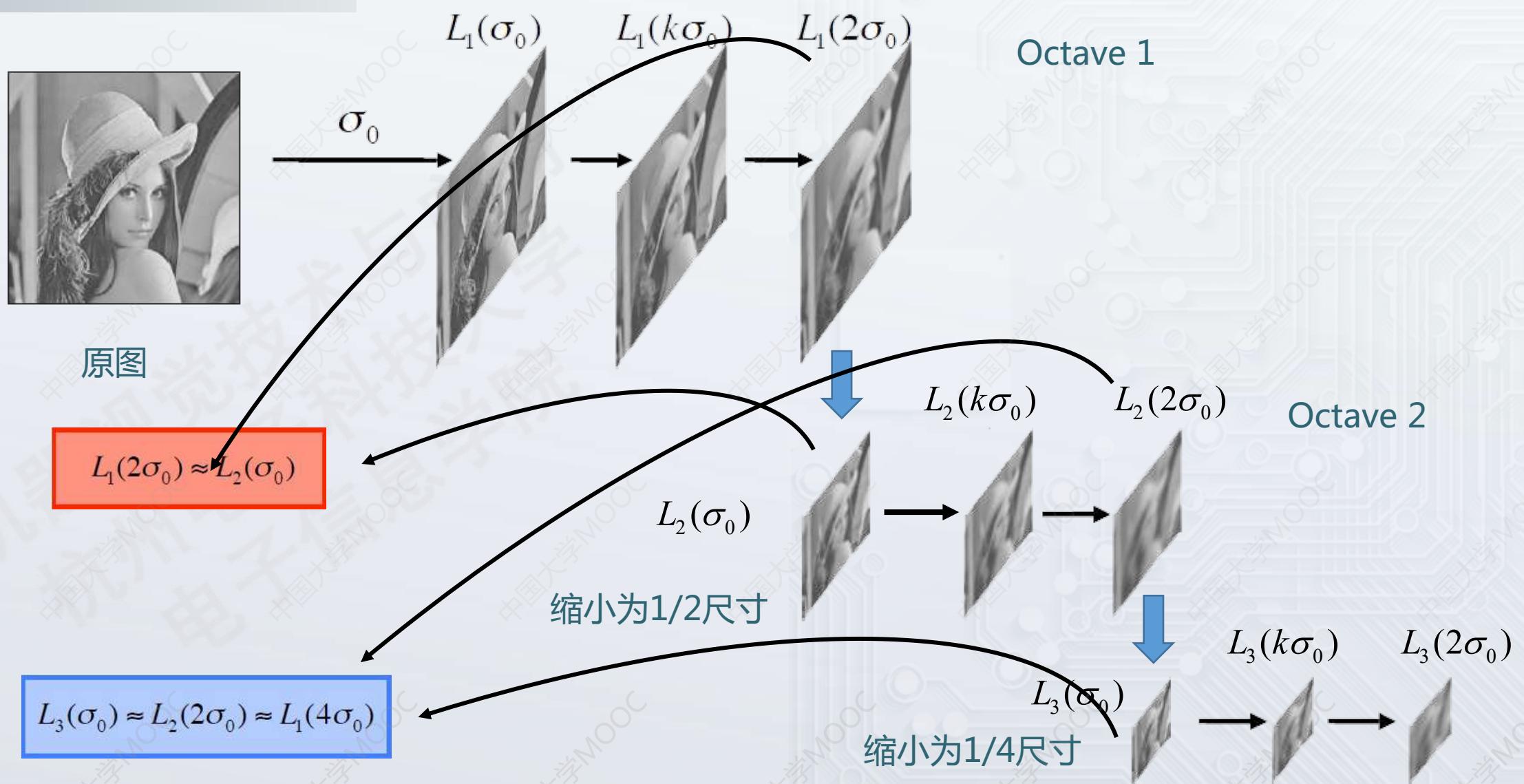
# 图像金字塔



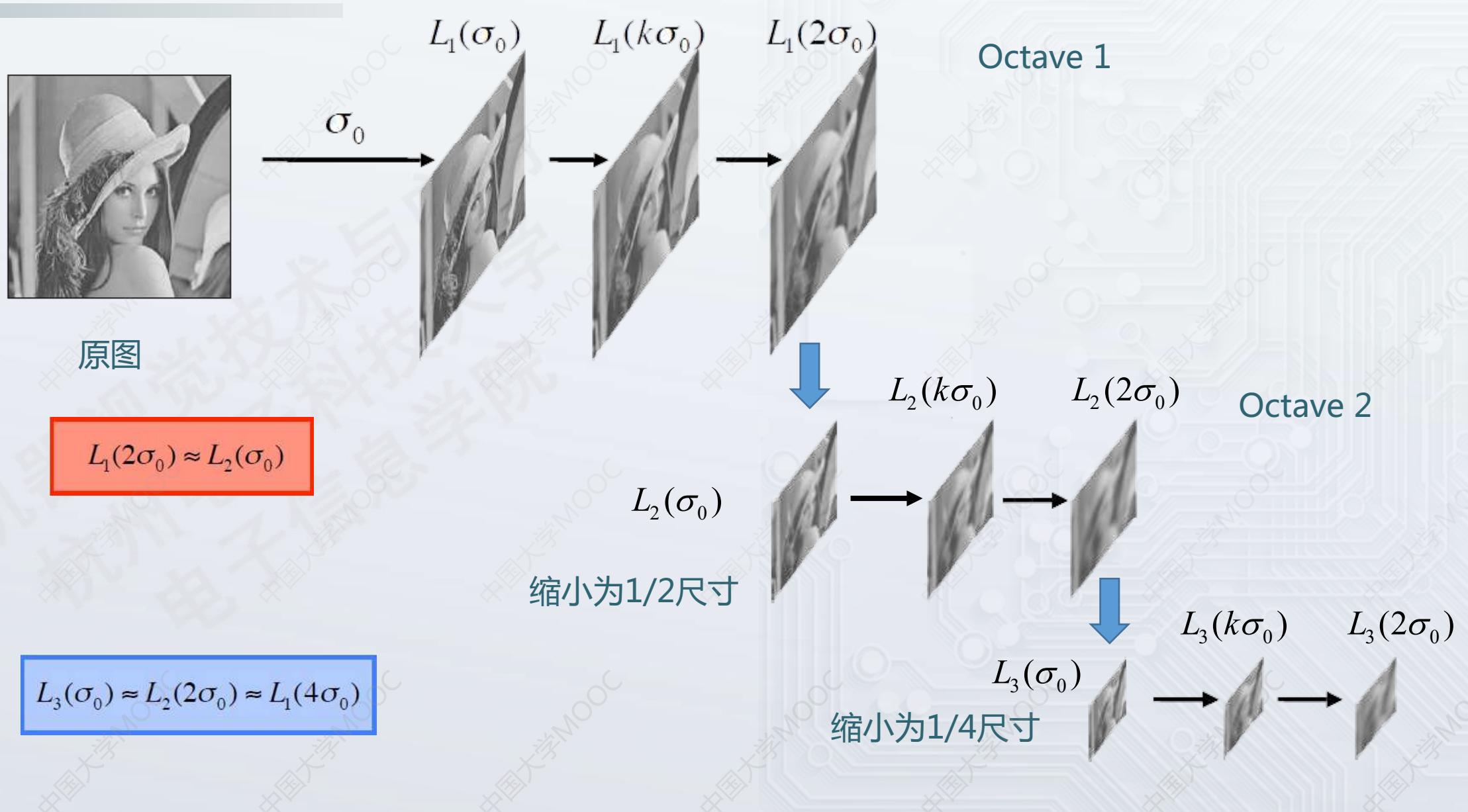
# 图像金字塔



# 图像金字塔

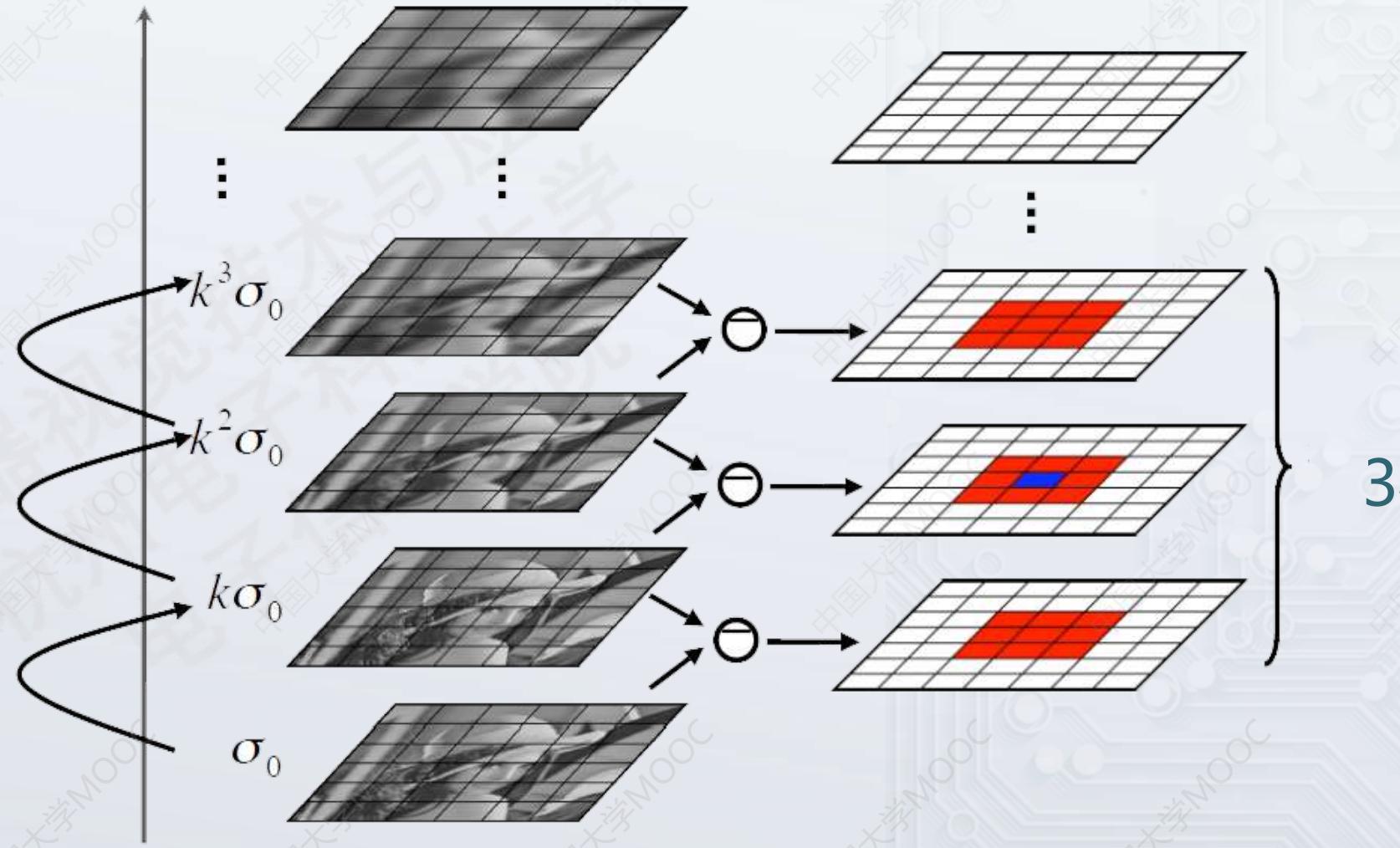


# 图像金字塔



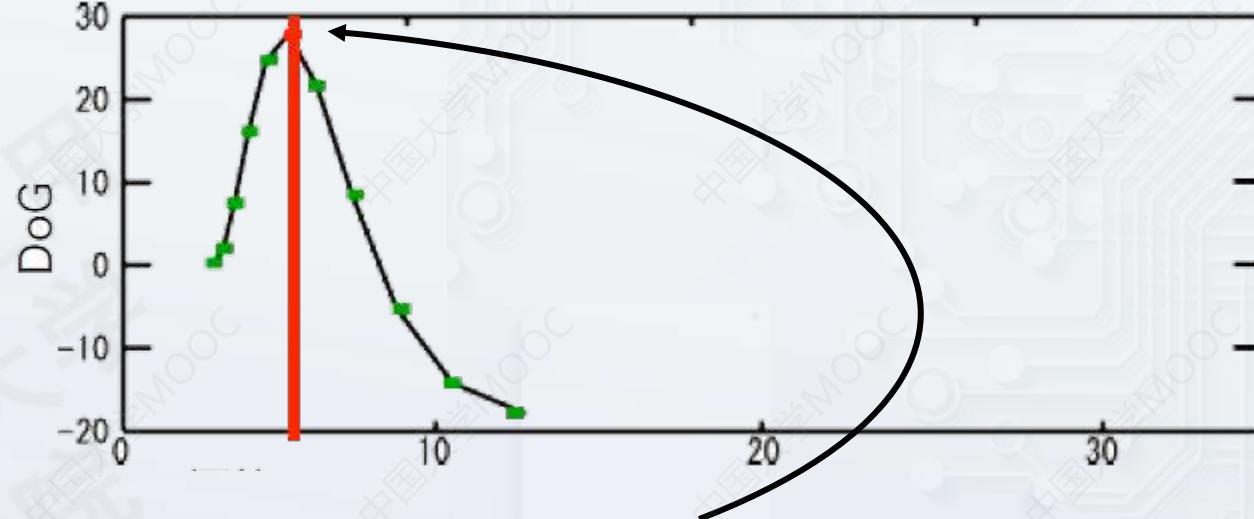
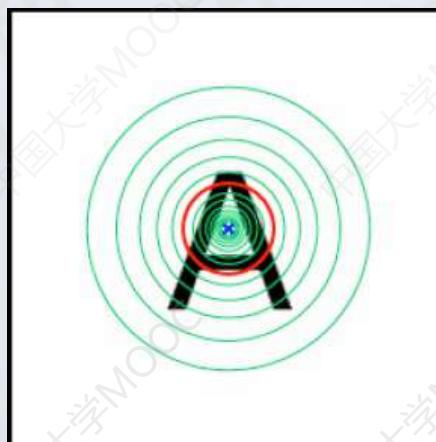
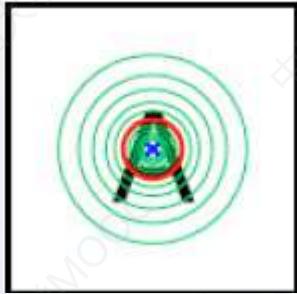
# DoG极值检出

将某像素和其周围的26个像素进行比较



3枚1组

# 极值的性质



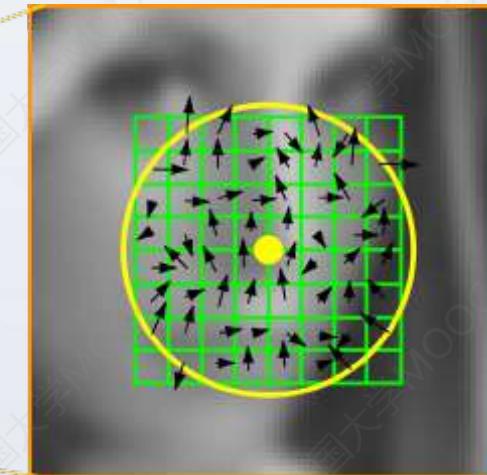
对尺度变换具有鲁棒性的点



# 特征点描述

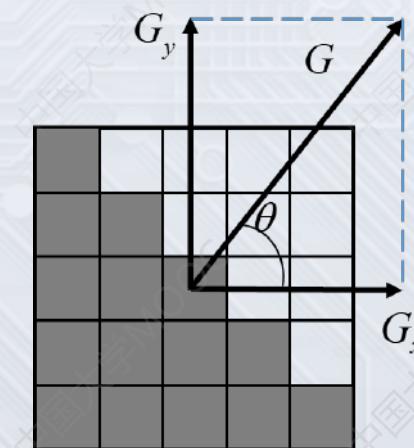
使用梯度值对特征点进行表述，计算像素的梯度值和梯度方向。

相对像素的亮度值，梯度对光照具有更好的鲁棒性。



$$\text{value} : G = \sqrt{G_x^2 + G_y^2}$$

$$\text{direction} : \theta = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

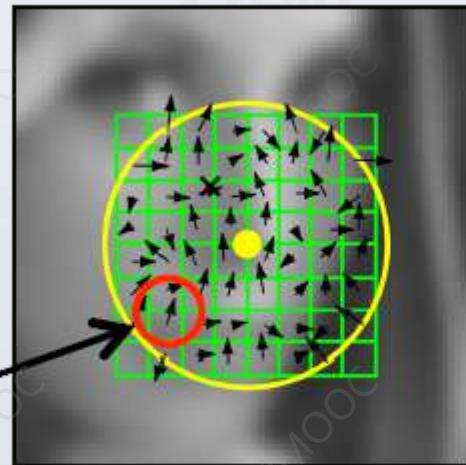


# 特征点描述

使用计算得到的梯度信息建立梯度直方图。

- 角度，36等分离散化
- 在特征点对应的scale层，计算梯度
- 高斯核及梯度强度计算权重

梯度方向



高斯核函数  
梯度强度  $\times$  = 权重

$$G = \sqrt{G_x^2 + G_y^2}$$

$$\text{value} : G = \sqrt{G_x^2 + G_y^2}$$

$$\text{direction} : \theta = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$



# 特征点描述

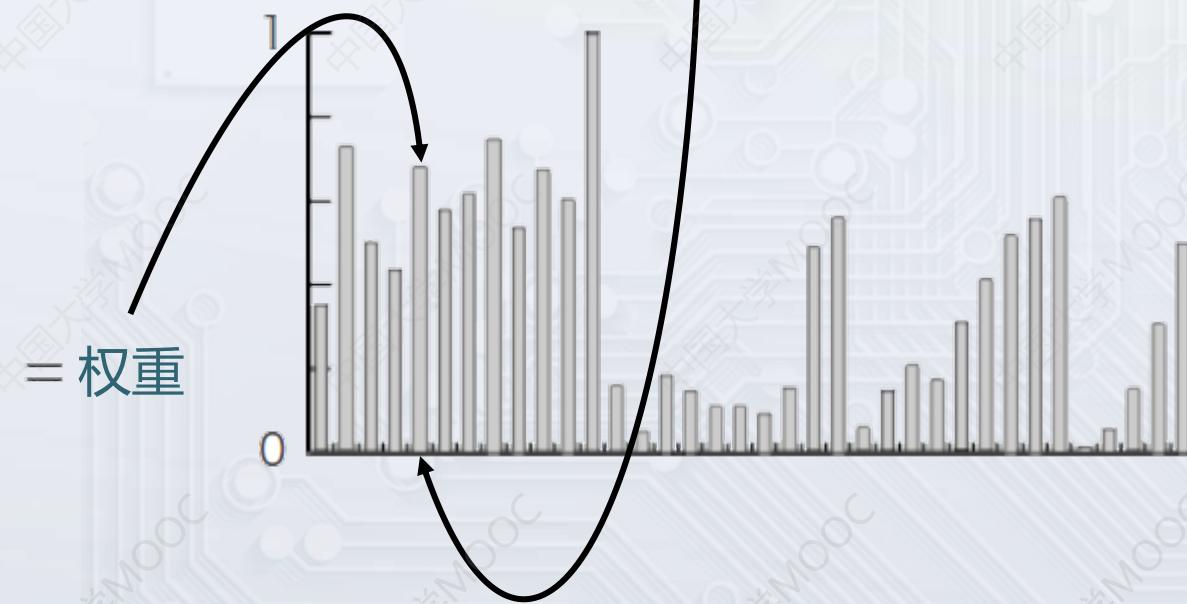
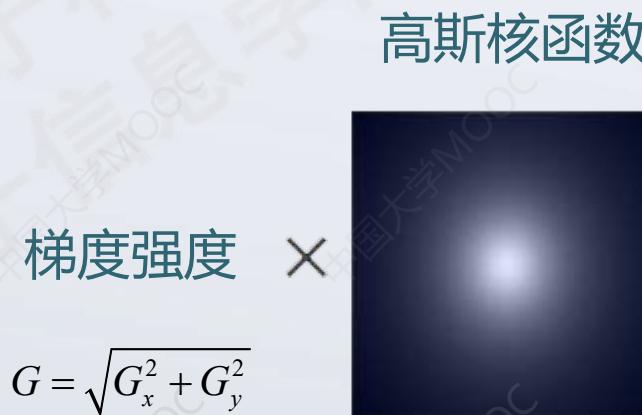
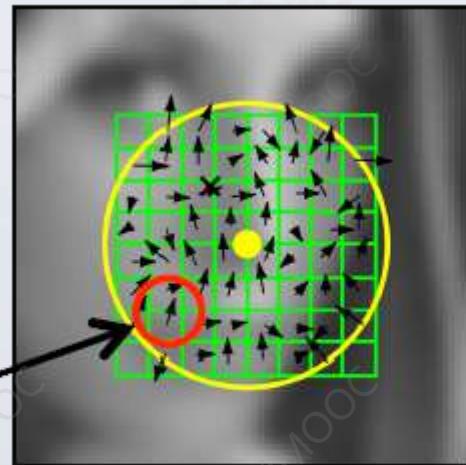
使用计算得到的梯度信息建立梯度直方图。

- 角度，36等分离散化
- 在特征点对应的scale层，计算梯度
- 高斯核及梯度强度计算权重

$$\text{value} : G = \sqrt{G_x^2 + G_y^2}$$

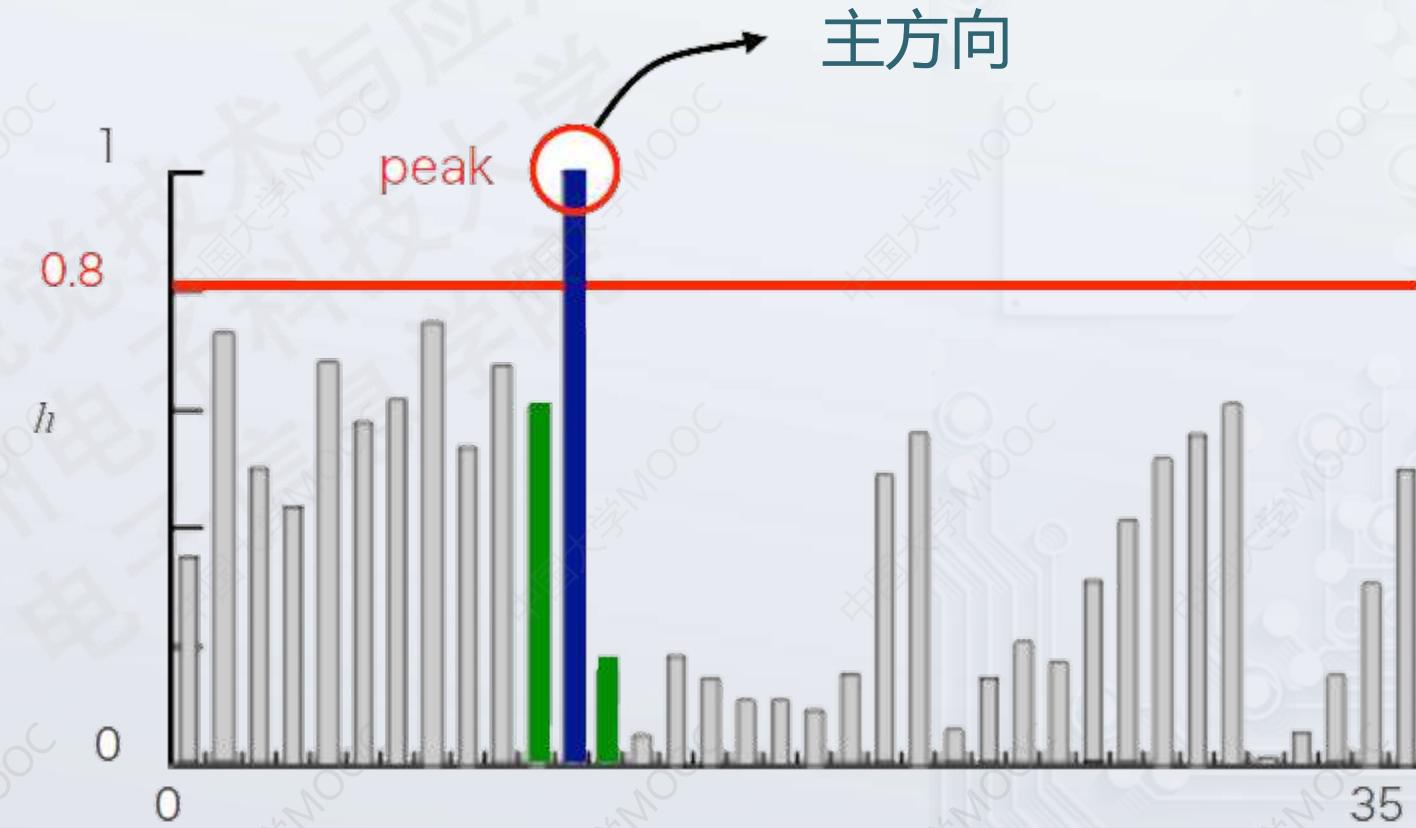
$$\text{direction} : \theta = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

梯度方向



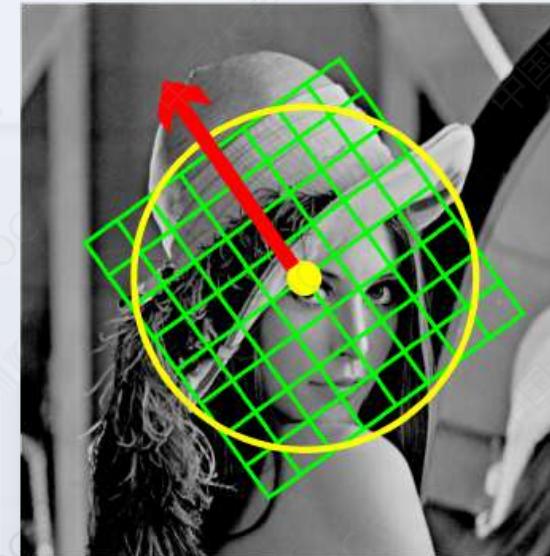
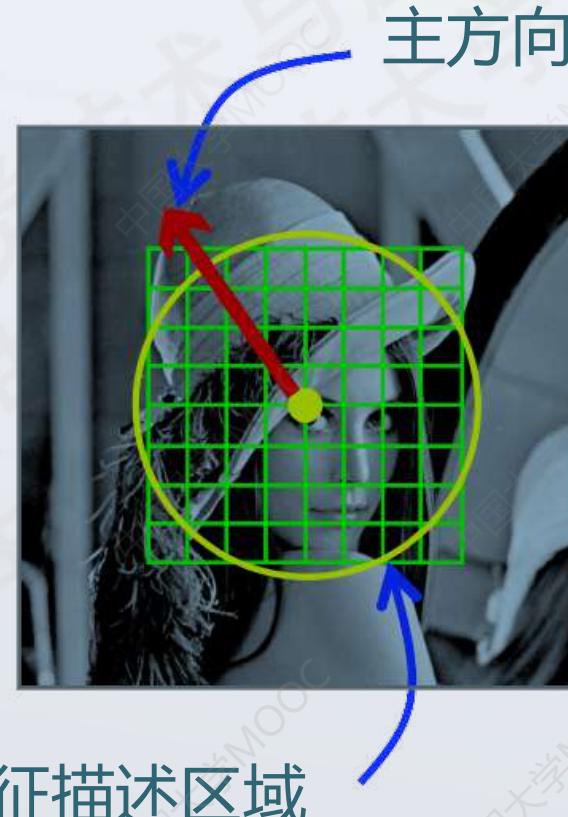
# 特征点描述

获得具有代表性的梯度方向，作为主方向。



# 特征点描述

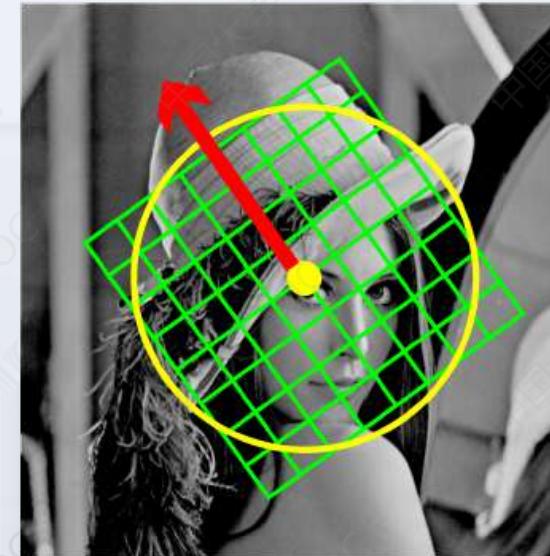
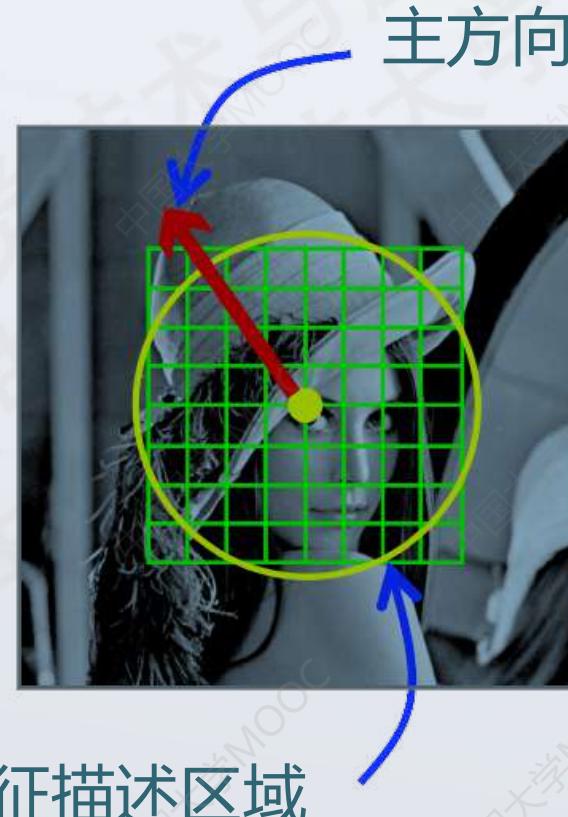
获得具有代表性的梯度方向，作为主方向。以主方向作为参考，计算各个梯度的相对角度。



获得旋转鲁棒性

# 特征点描述

获得具有代表性的梯度方向，作为主方向。以主方向作为参考，计算各个梯度的相对角度。



获得旋转鲁棒性

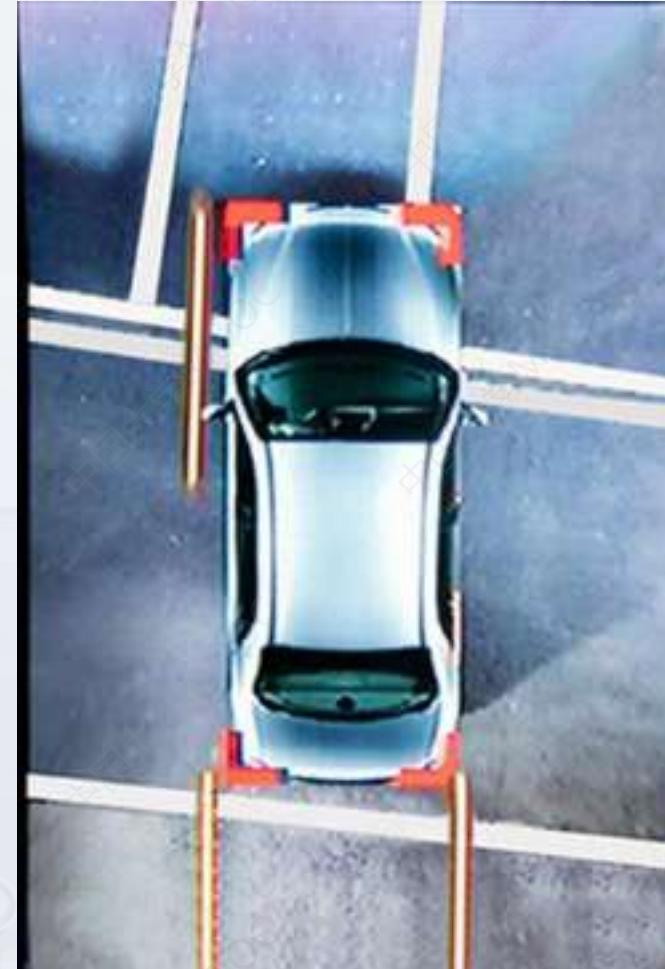
# 特征点的应用

- 图像拼接
- 运动跟踪
- 物体识别

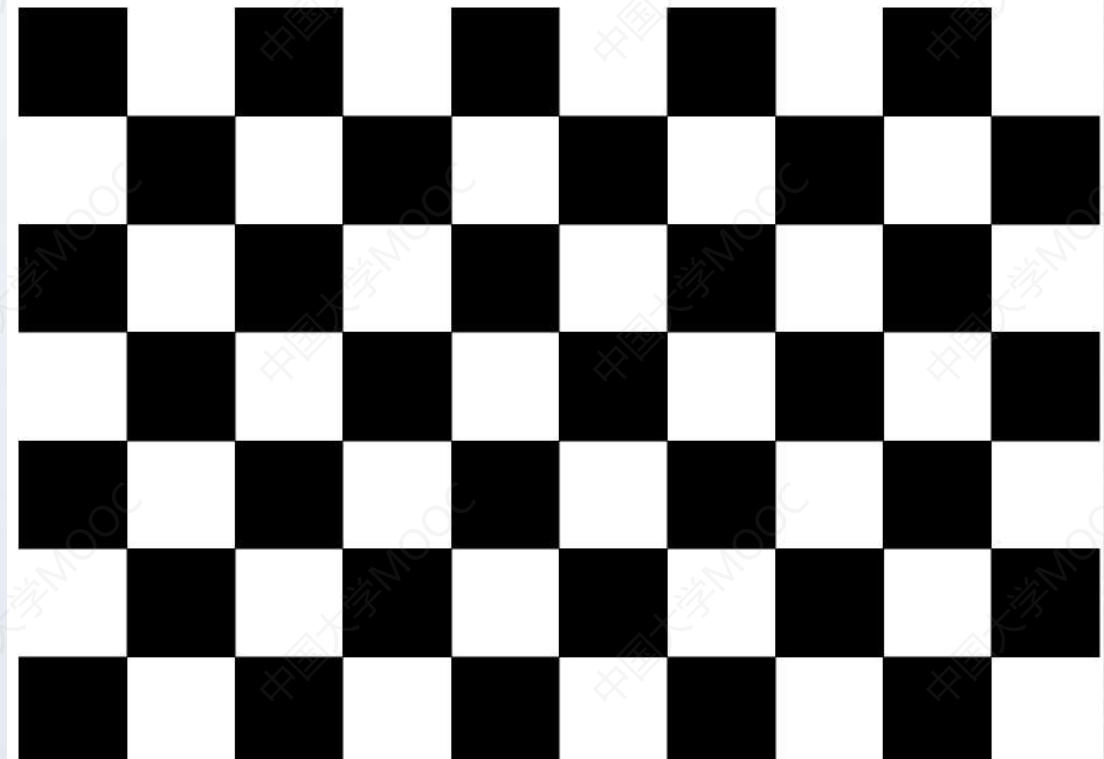
# 全景泊车



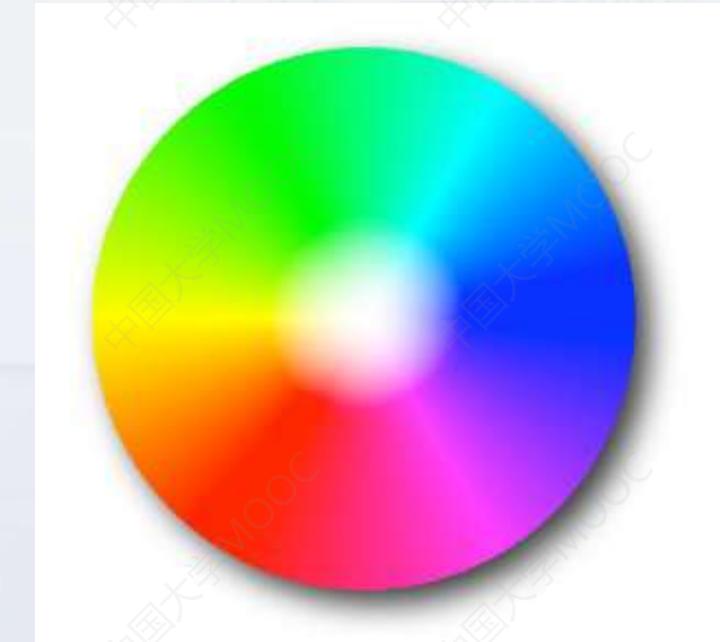
# 全景泊车



# 全景泊车



# 移动物体追踪



Color coding

# 移动物体追踪



箭头表示运动向量

当前位置



前一帧位置

# HOG

方向梯度直方图 ( Histogram of Oriented Gradient, HOG ) 特征是一种在计算机视觉和图像处理中用来进行物体检测的特征描述子。它通过计算和统计图像局部区域的梯度方向直方图来构成特征。Hog特征结合SVM分类器已经被广泛应用于图像识别中。

# HOG

SIFT : 对特征点的描述方法

HOG : 对一定区域的特征量的描述方法

- 可以表现较大的形状
- 非常适合行人及车辆检测



# HOG

## HOG的主要步骤

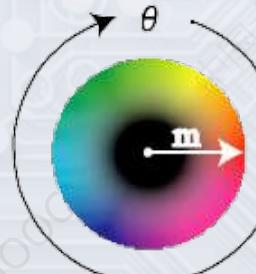
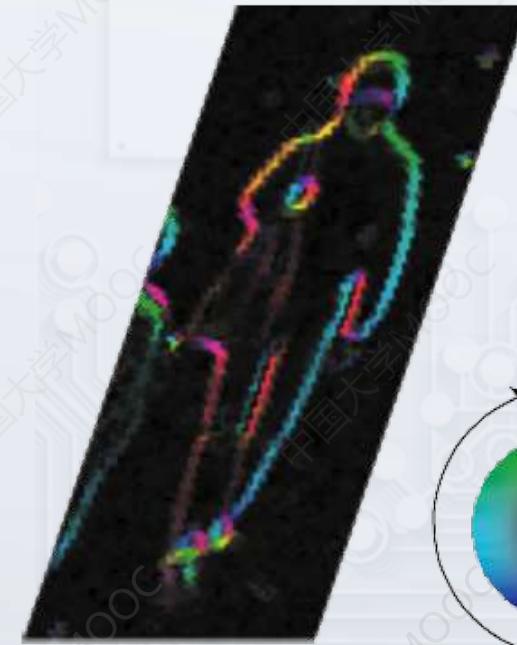
- Gamma矫正
- 计算梯度信息
- 以cell为单位计算梯度直方图
- 以block为单位，对特征量进行归一化

# HOG

计算各个像素的梯度强度和方向。

$$\text{value} : G = \sqrt{G_x^2 + G_y^2}$$

$$\text{direction} : \theta = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

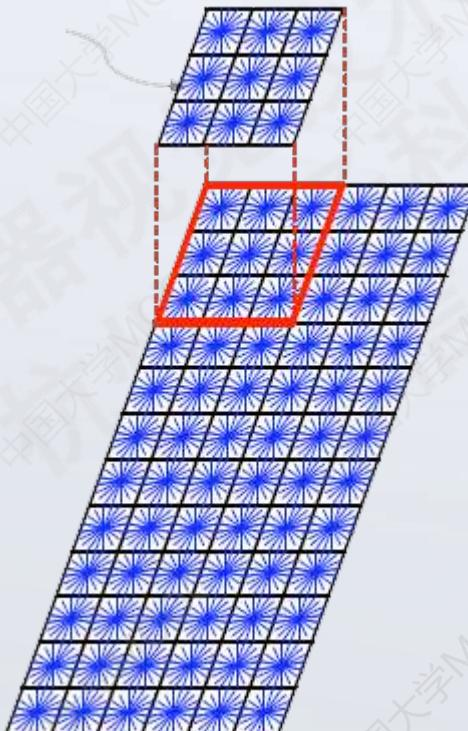


# HOG

每3x3个cell组成一个block，在每个block中进行归一化。

- Block的特征向量

$$\mathbf{V}_k = [\mathbf{F}_{ij}, \mathbf{F}_{i+1j}, \mathbf{F}_{i+2j}, \mathbf{F}_{ij+1}, \mathbf{F}_{i+1j+1}, \mathbf{F}_{i+2j+1}, \mathbf{F}_{ij+2}, \mathbf{F}_{i+1j+2}, \mathbf{F}_{i+2j+2}]$$



使用向量 $\mathbf{V}$ 进行归一化

$$v = \frac{f}{\sqrt{\|\mathbf{V}\|^2 + \epsilon^2}} \quad (\epsilon = 1)$$

Block数 : 40

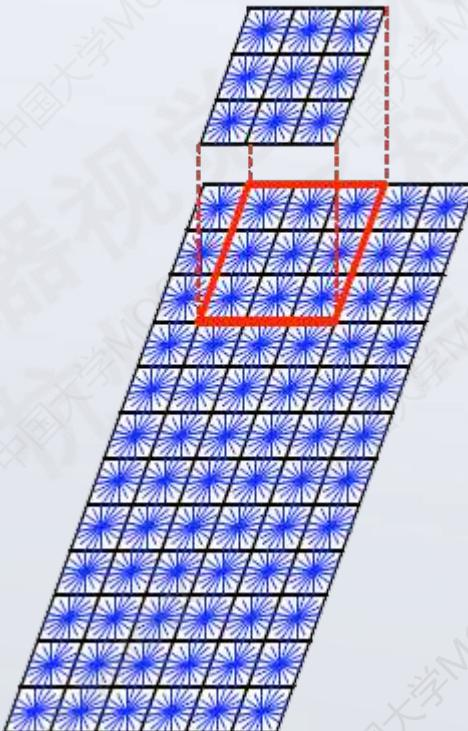
维度 :  $40 \times 81 = 3240$

# HOG

每3x3个cell组成一个block，在每个block中进行归一化。

- Block的特征向量

$$\mathbf{V}_k = [\mathbf{F}_{ij}, \mathbf{F}_{i+1j}, \mathbf{F}_{i+2j}, \mathbf{F}_{ij+1}, \mathbf{F}_{i+1j+1}, \mathbf{F}_{i+2j+1}, \mathbf{F}_{ij+2}, \mathbf{F}_{i+1j+2}, \mathbf{F}_{i+2j+2}]$$



使用向量 $\mathbf{V}$ 进行归一化

$$v = \frac{f}{\sqrt{\|\mathbf{V}\|^2 + \epsilon^2}} \quad (\epsilon = 1)$$

Block数 : 40

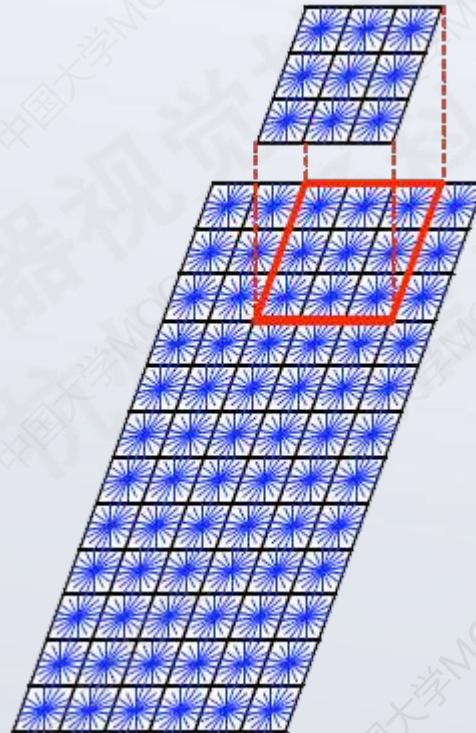
维度 :  $40 \times 81 = 3240$

# HOG

每3x3个cell组成一个block，在每个block中进行归一化。

- Block的特征向量

$$\mathbf{V}_k = [\mathbf{F}_{ij}, \mathbf{F}_{i+1j}, \mathbf{F}_{i+2j}, \mathbf{F}_{ij+1}, \mathbf{F}_{i+1j+1}, \mathbf{F}_{i+2j+1}, \mathbf{F}_{ij+2}, \mathbf{F}_{i+1j+2}, \mathbf{F}_{i+2j+2}]$$



使用向量 $\mathbf{V}$ 进行归一化

$$v = \frac{f}{\sqrt{\|\mathbf{V}\|^2 + \epsilon^2}} \quad (\epsilon = 1)$$

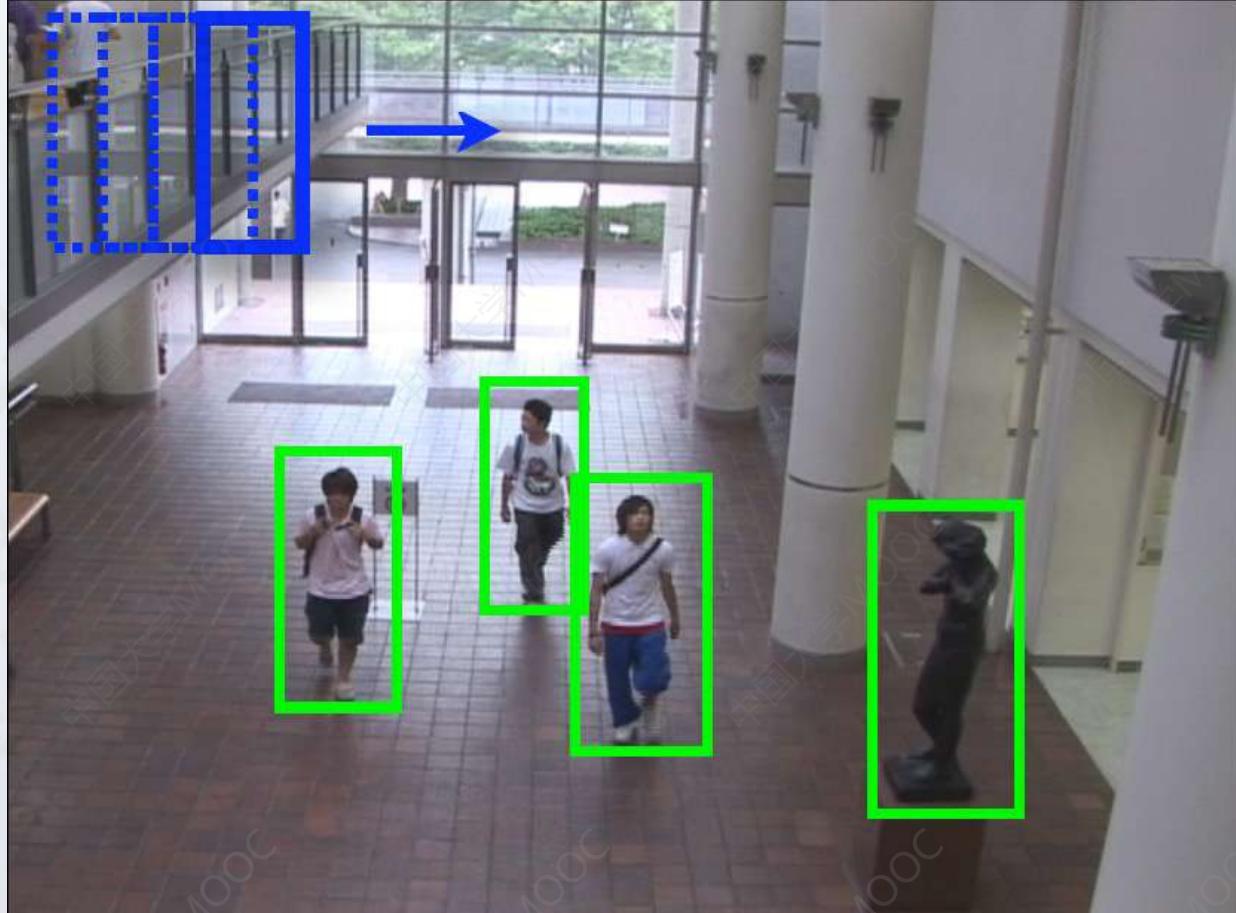
Block数 : 40

维度 :  $40 \times 81 = 3240$

# HOG



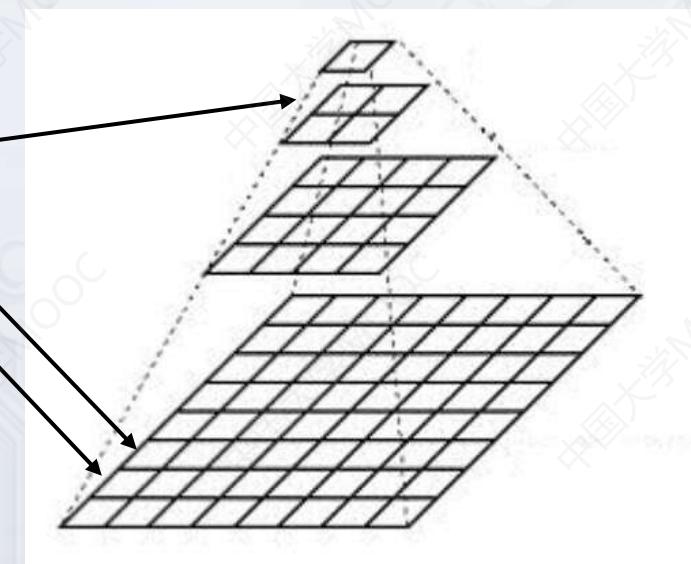
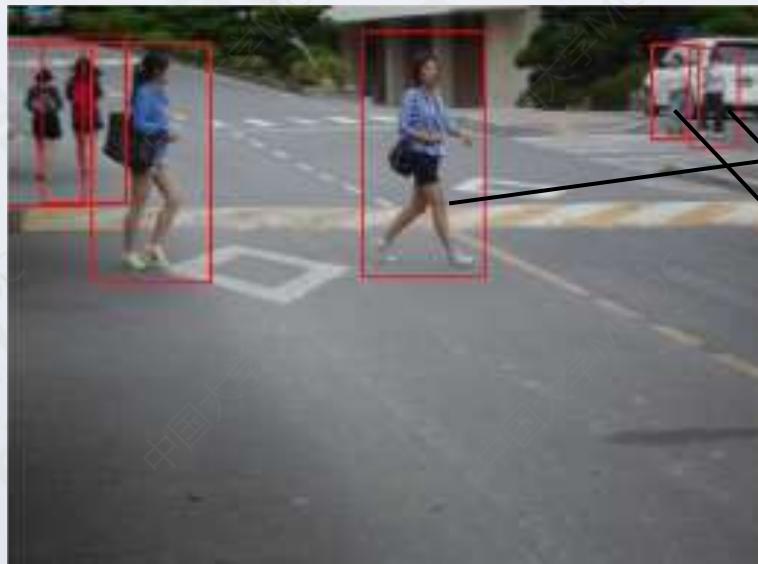
# HOG



# HOG

## 使用图像金字塔对应尺寸变化

pedestrian with different size are detected from different layer



# 谢谢！

# 机器视觉技术与应用

## 9. 频率域处理

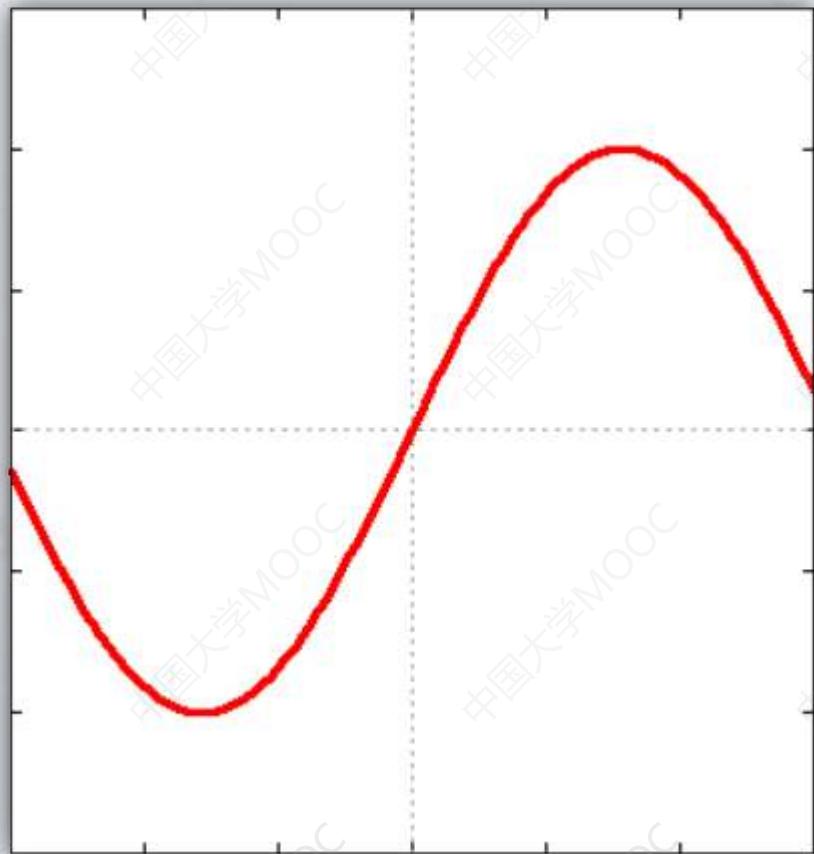
李竹

杭州电子科技大学

电子信息学院



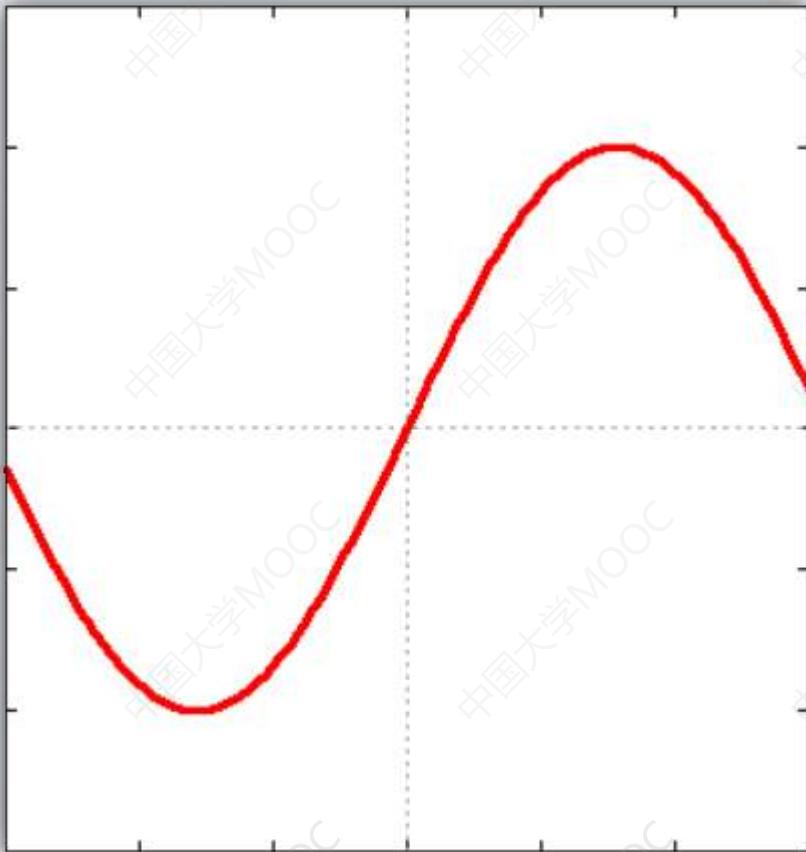
# 傅里叶变换



某函数表达式如： $y = \sum_{k=1}^{\infty} \frac{2}{k} (-1)^{k+1} \sin(kx)$

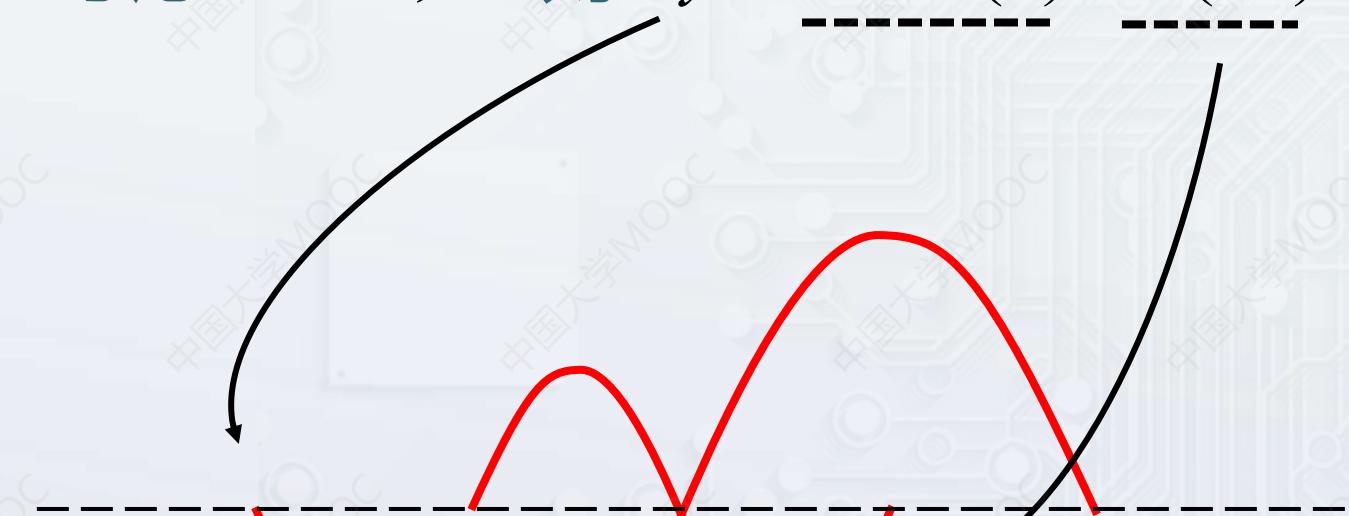
$$k = 1 \quad \text{则} \quad y = 2 \sin x(x)$$

# 傅里叶变换

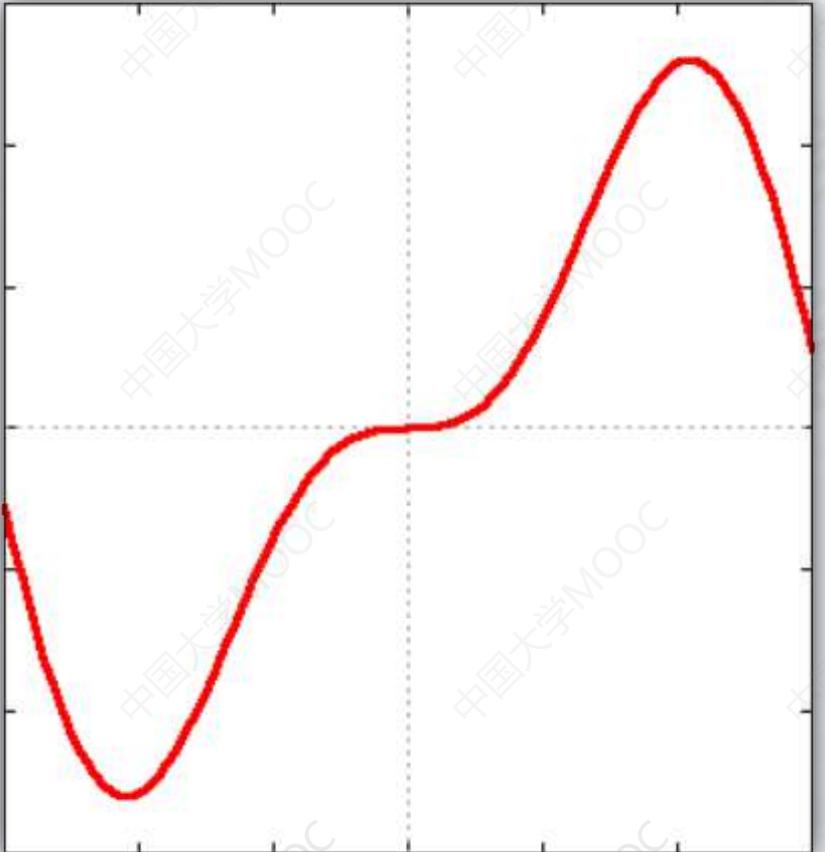


某函数表达式如： $y = \sum_{k=1}^{\infty} \frac{2}{k} (-1)^{k+1} \sin(kx)$

考虑  $k = 1, 2$  则  $y = 2\sin(x) - \sin(2x)$



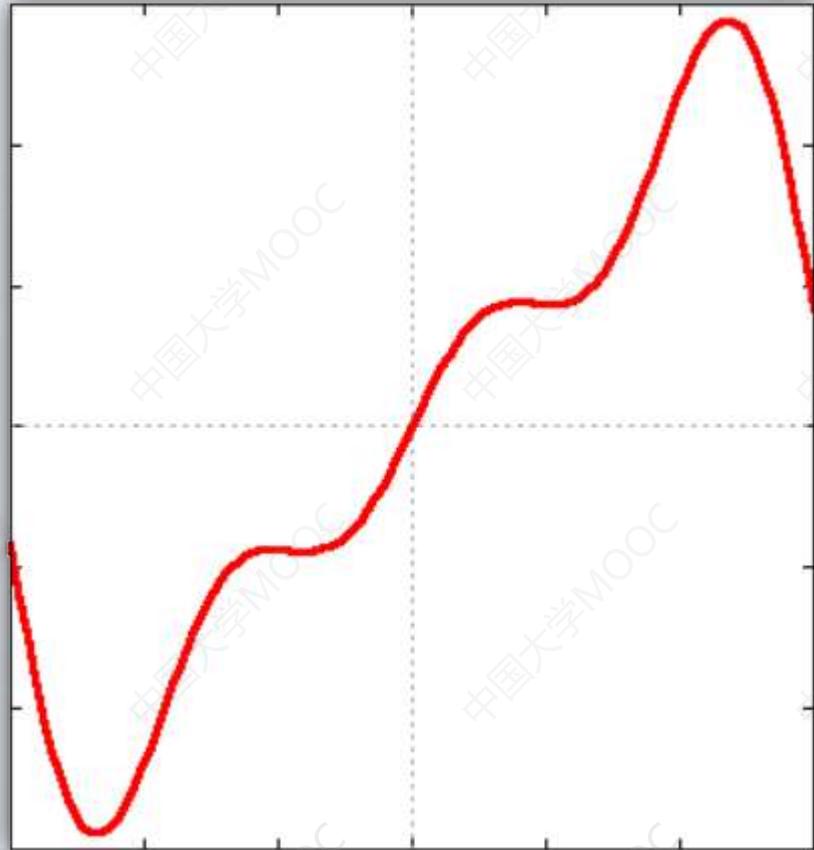
# 傅里叶变换



某函数表达式如： $y = \sum_{k=1}^{\infty} \frac{2}{k} (-1)^{k+1} \sin(kx)$

考虑  $k = 1, 2$  则  $y = 2\sin(x) - \sin(2x)$

# 傅里叶变换

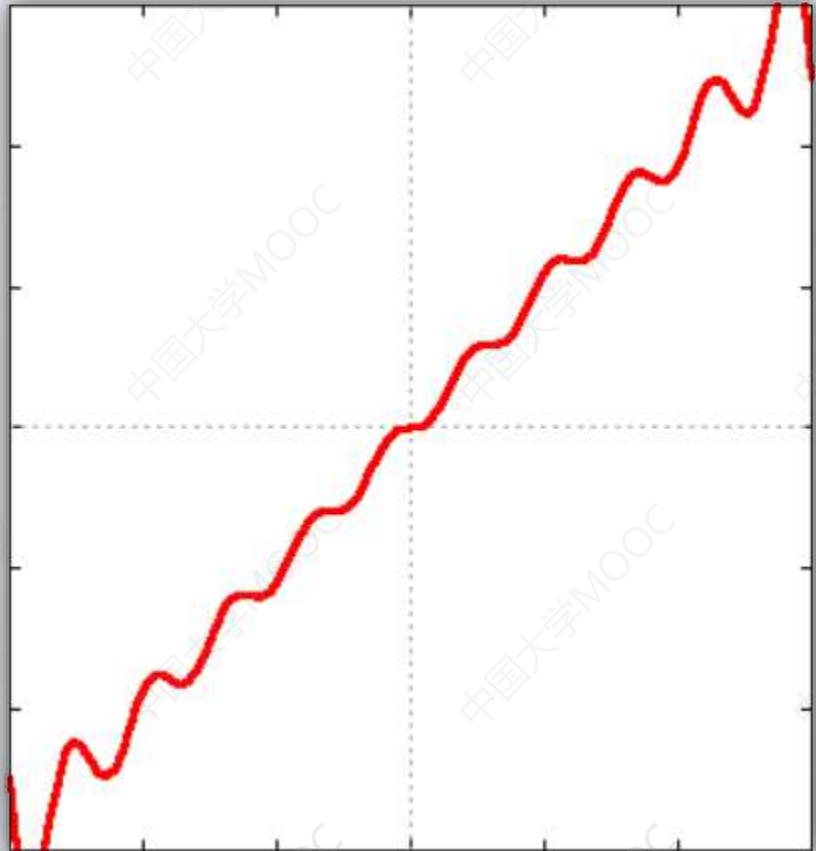


某函数表达式如： $y = \sum_{k=1}^{\infty} \frac{2}{k} (-1)^{k+1} \sin(kx)$

考虑  $k = 1, 2, 3$  则

$$y = 2\sin(x) - \sin(2x) + \frac{2}{3}\sin(3x)$$

# 傅里叶变换

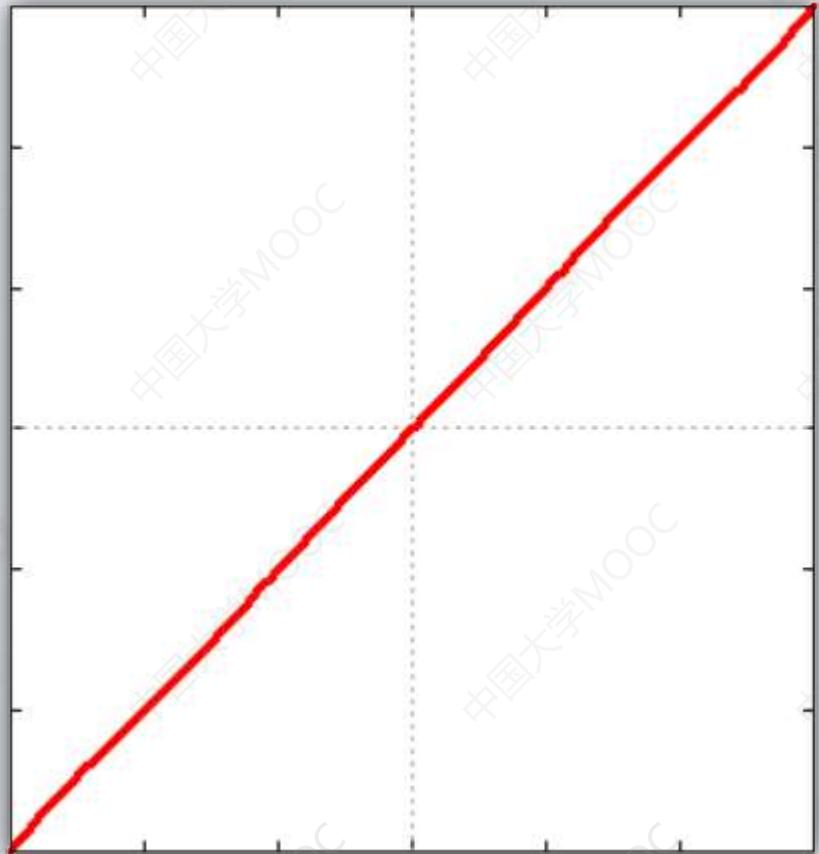


某函数表达式如： $y = \sum_{k=1}^{\infty} \frac{2}{k} (-1)^{k+1} \sin(kx)$

考虑  $k = 1, 2, 3, \dots, 10$

$$y = 2\sin(x) - \sin(2x) + \frac{2}{3}\sin(3x) - \frac{2}{5}\sin(5x) + \frac{2}{7}\sin(7x) - \frac{2}{9}\sin(9x) + \frac{2}{11}\sin(11x)$$

# 傅里叶变换



某函数表达式如： $y = \sum_{k=1}^{\infty} \frac{2}{k} (-1)^{k+1} \sin(kx)$

考虑  $k = 1, 2, 3, \dots, 100$

# 傅里叶变换

周期为 $2\pi$ 的周期函数，可以通过以下形式近似。

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{\infty} (a_k \cos(kx) + b_k \sin(kx))$$

$$a_k = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \cos(kx) dx, \quad k = 0, 1, 2, \dots$$

$$b_k = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \sin(kx) dx, \quad k = 0, 1, 2, \dots$$

# 傅里叶变换

周期为 $2\pi$ 的周期函数，可以通过以下形式近似。

常数项

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{\infty} (a_k \cos(kx) + b_k \sin(kx))$$

各种不同频率的 $\sin$ ,  $\cos$ 函数的组合

$$a_k = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \cos(kx) dx, \quad k = 0, 1, 2, \dots$$

$$b_k = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \sin(kx) dx, \quad k = 0, 1, 2, \dots$$

# 傅里叶变换

周期为 $2\pi$ 的周期函数，可以通过以下形式近似。

常数项

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{\infty} (a_k \cos(kx) + b_k \sin(kx))$$

各种不同频率的 $\sin$ ,  $\cos$ 函数的组合

$$a_k = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \cos(kx) dx, \quad k = 0, 1, 2, \dots$$

$$b_k = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \sin(kx) dx, \quad k = 0, 1, 2, \dots$$

$a_k, b_k$ 称为傅里叶系数

# 傅里叶变换

例 :  $f(x) = x^2$

$$a_k = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \cos(kx) dx = \frac{1}{\pi} \int_{-\pi}^{\pi} x^2 \cos(kx) dx \left\{ \begin{array}{l} \frac{\pi^2}{3}, k = 0 \\ (-1)^k \frac{4}{k^2}, k = 1, 2, \dots \end{array} \right.$$

$$b_k = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \sin(kx) dx = \frac{1}{\pi} \int_{-\pi}^{\pi} x^2 \sin(kx) dx = 0$$

代入 

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{\infty} (a_k \cos(kx) + b_k \sin(kx)) = \frac{\pi^2}{3} + \sum_{k=1}^{\infty} (-1)^k \frac{4}{k^2} \cos(kx)$$

# 傅里叶变换

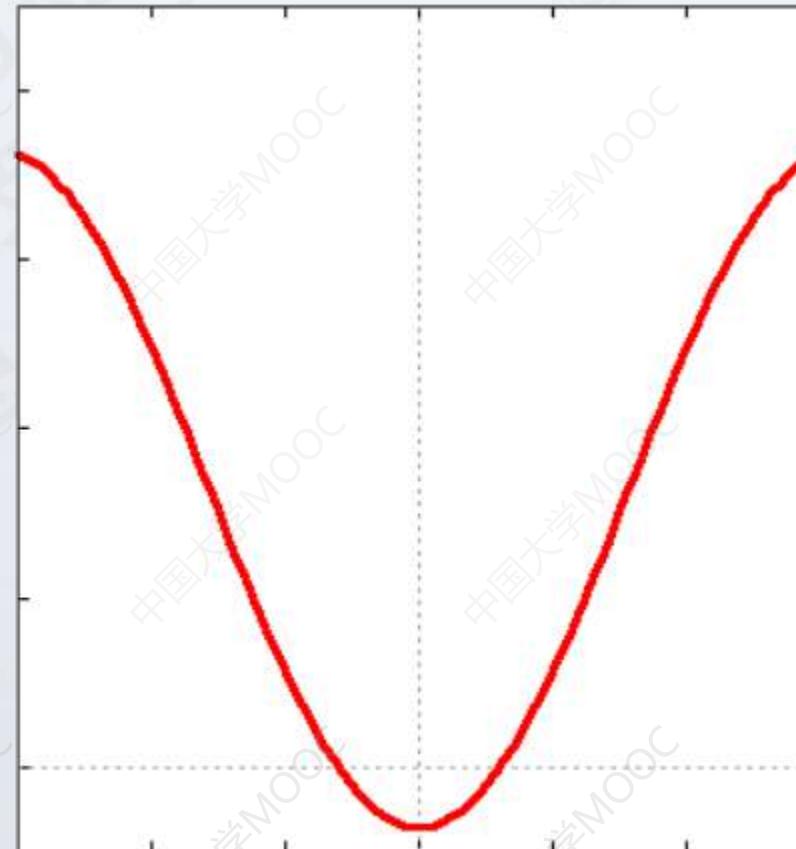
例 :  $f(x) = x^2 = \frac{\pi^2}{3} + \sum_{k=1}^{\infty} (-1)^k \frac{4}{k^2} \cos(kx)$



$k=0$

# 傅里叶变换

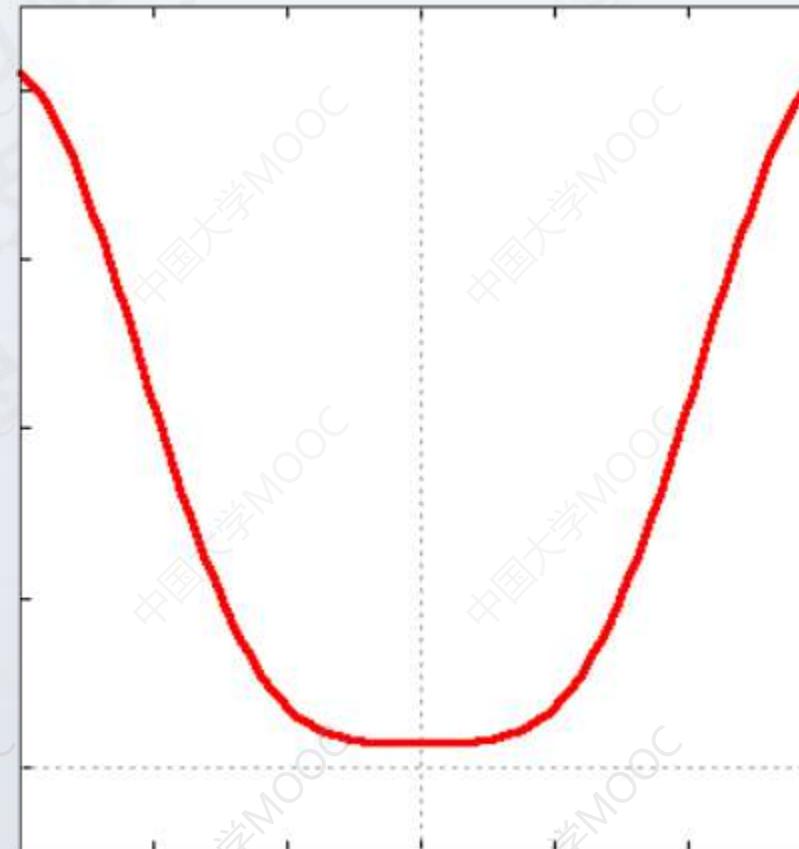
例 :  $f(x) = x^2 = \frac{\pi^2}{3} + \sum_{k=1}^{\infty} (-1)^k \frac{4}{k^2} \cos(kx)$



$k=1$

# 傅里叶变换

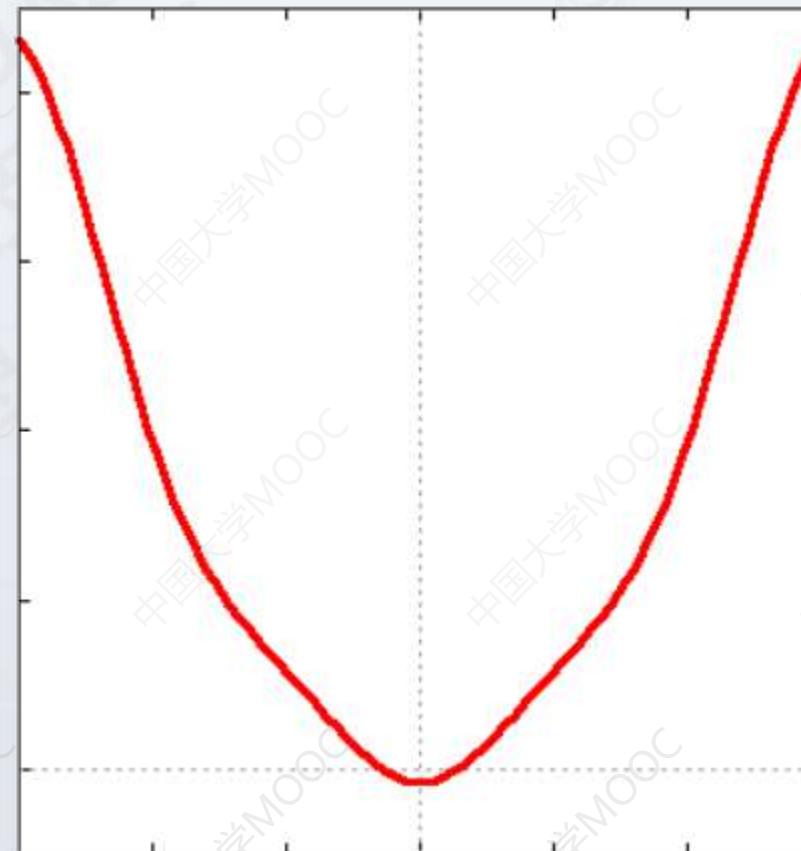
例 :  $f(x) = x^2 = \frac{\pi^2}{3} + \sum_{k=1}^{\infty} (-1)^k \frac{4}{k^2} \cos(kx)$



$k=2$

# 傅里叶变换

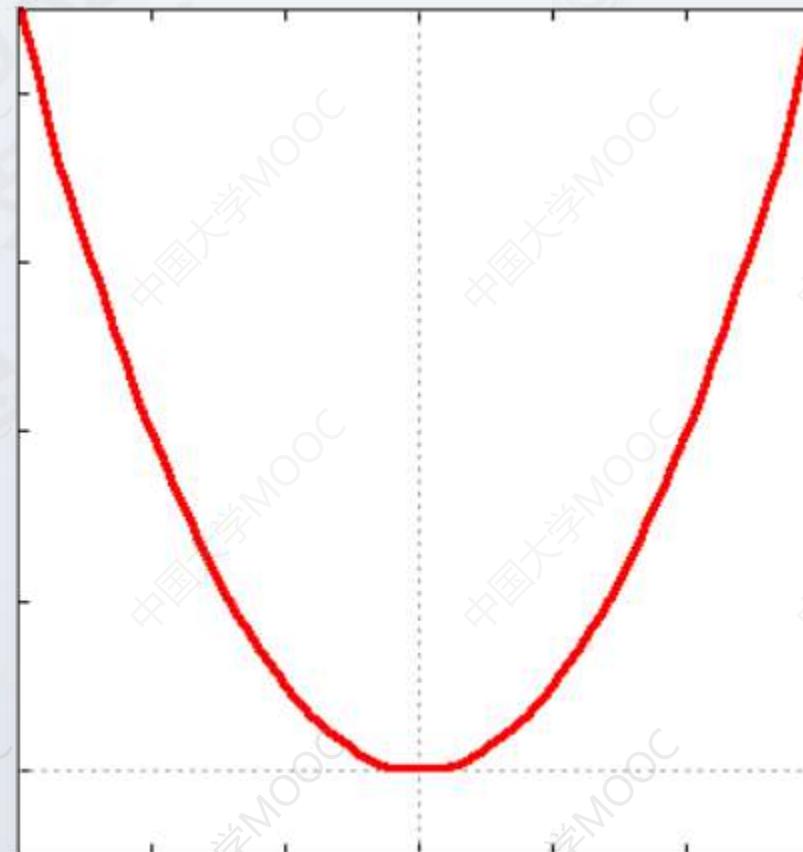
例 :  $f(x) = x^2 = \frac{\pi^2}{3} + \sum_{k=1}^{\infty} (-1)^k \frac{4}{k^2} \cos(kx)$



k=3

# 傅里叶变换

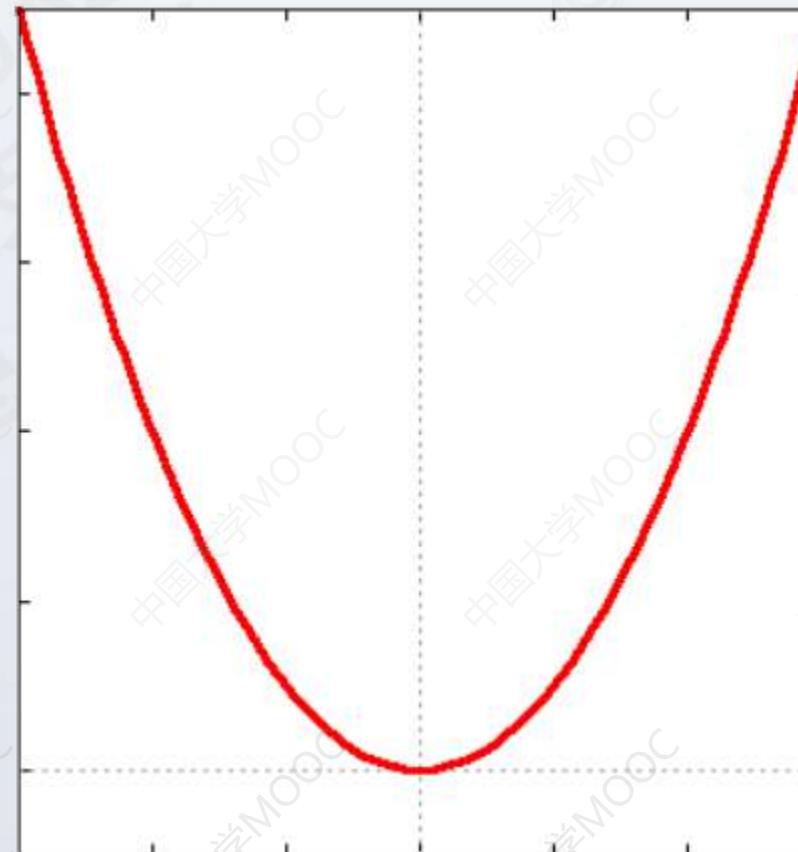
例 :  $f(x) = x^2 = \frac{\pi^2}{3} + \sum_{k=1}^{\infty} (-1)^k \frac{4}{k^2} \cos(kx)$



k=10

# 傅里叶变换

例 :  $f(x) = x^2 = \frac{\pi^2}{3} + \sum_{k=1}^{\infty} (-1)^k \frac{4}{k^2} \cos(kx)$



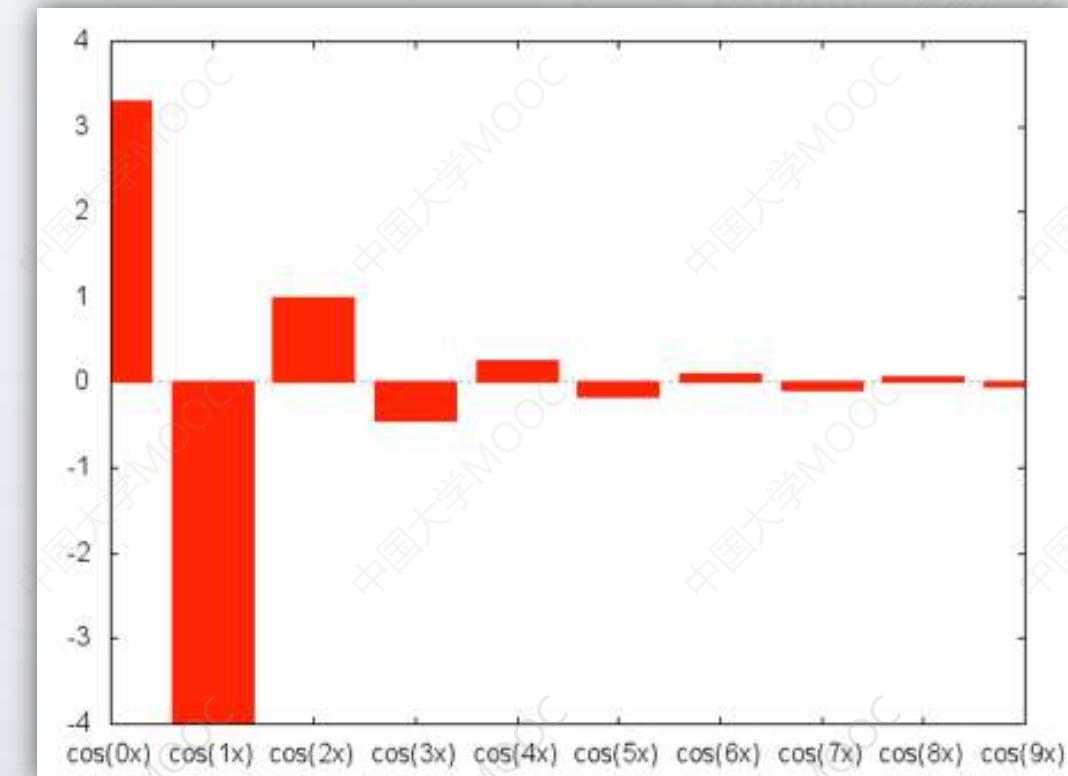
$k=100$

# 傅里叶系数和频率

例 :  $f(x) = x^2 = \frac{\pi^2}{3} + \sum_{k=1}^{\infty} (-1)^k \frac{4}{k^2} \cos(kx)$

频率	系数
常数	$\pi^2/3$
$\cos(x)$	-4
$\cos(2x)$	1
$\cos(3x)$	-4/9
...	...

通过求解傅里叶系数，可以知道一个函数是由什么样频率的函数构成的



# 傅里叶变换

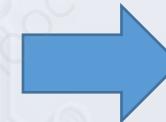
傅里叶展开的复数形式：

欧拉公式： $e^{i\theta} = \cos \theta + i \sin \theta$  代入

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{\infty} (a_k \cos(kx) + b_k \sin(kx))$$

$$a_k = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \cos(kx) dx, \quad k = 0, 1, 2, \dots$$

$$b_k = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \sin(kx) dx, \quad k = 0, 1, 2, \dots$$



$$\begin{aligned} f(x) &= \sum_{k=0, \pm 1}^{\pm \infty} c_k e^{ikx}, \\ c_k &= \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) e^{-ikx} dx \end{aligned}$$

# 傅里叶变换

周期从 $2\pi$ 改为 $T$ ：

$$f(x) = \sum_{k=0,\pm 1}^{\pm\infty} c_k e^{ikx}, \quad c_k = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) e^{-ikx} dx$$



$$f(x) = \sum_{k=0,\pm 1}^{\pm\infty} c_k e^{ikx}, \quad c_k = \frac{1}{T} \int_{-T/2}^{T/2} f(x) e^{-i\frac{2\pi}{T} kx} dx$$



$$f(x) = \sum_{k=0,\pm 1}^{\pm\infty} \left( \frac{1}{T} \int_{-T/2}^{T/2} f(t) e^{-i\frac{2\pi}{T} kt} dt \right) e^{ikx}$$

# 傅里叶变换

扩展至无穷大周期  $T \rightarrow \infty$ , 连加改为积分, 积分变量  $2\pi/T \rightarrow \omega$

$$f(x) = \sum_{k=0, \pm 1}^{\pm \infty} \left( \frac{1}{T} \int_{-T/2}^{T/2} f(t) e^{-i \frac{2\pi}{T} kt} dt \right) e^{ikx}$$



傅里叶变换

$$f(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{i\omega x} \left( \int_{-\infty}^{\infty} f(t) e^{-i\omega t} dt \right) d\omega$$

傅里叶变换 :  $F(\omega) = \int_{-\infty}^{\infty} f(t) e^{-i\omega t} dt$



$$X_k = \sum_{n=0}^{N-1} x_n e^{-j \frac{2\pi k n}{N}}, k = 0, 1, 2, \dots, N-1$$

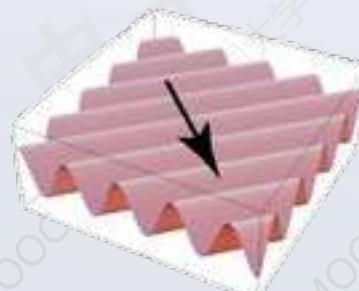
傅里叶逆变换 :  $f(x) = \int_{-\infty}^{\infty} F(\omega) e^{i\omega t} d\omega$

$N$ 个离散点  $x_n (n=0, 1, \dots, N-1)$

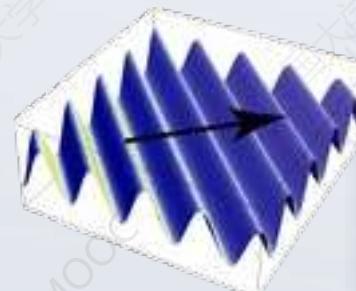
# 二维离散傅里叶变换

$$X_{u,v} = \sum_{h=0}^{H-1} \sum_{w=0}^{W-1} x_{h,w} e^{-j\pi(\frac{uh}{H} + \frac{vx}{W})}$$

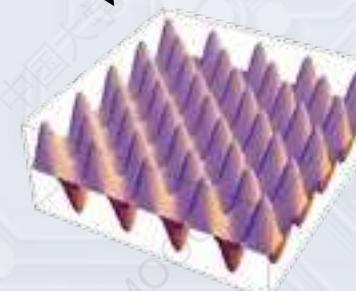
输入为二维数据，输出为包含的二维三角函数



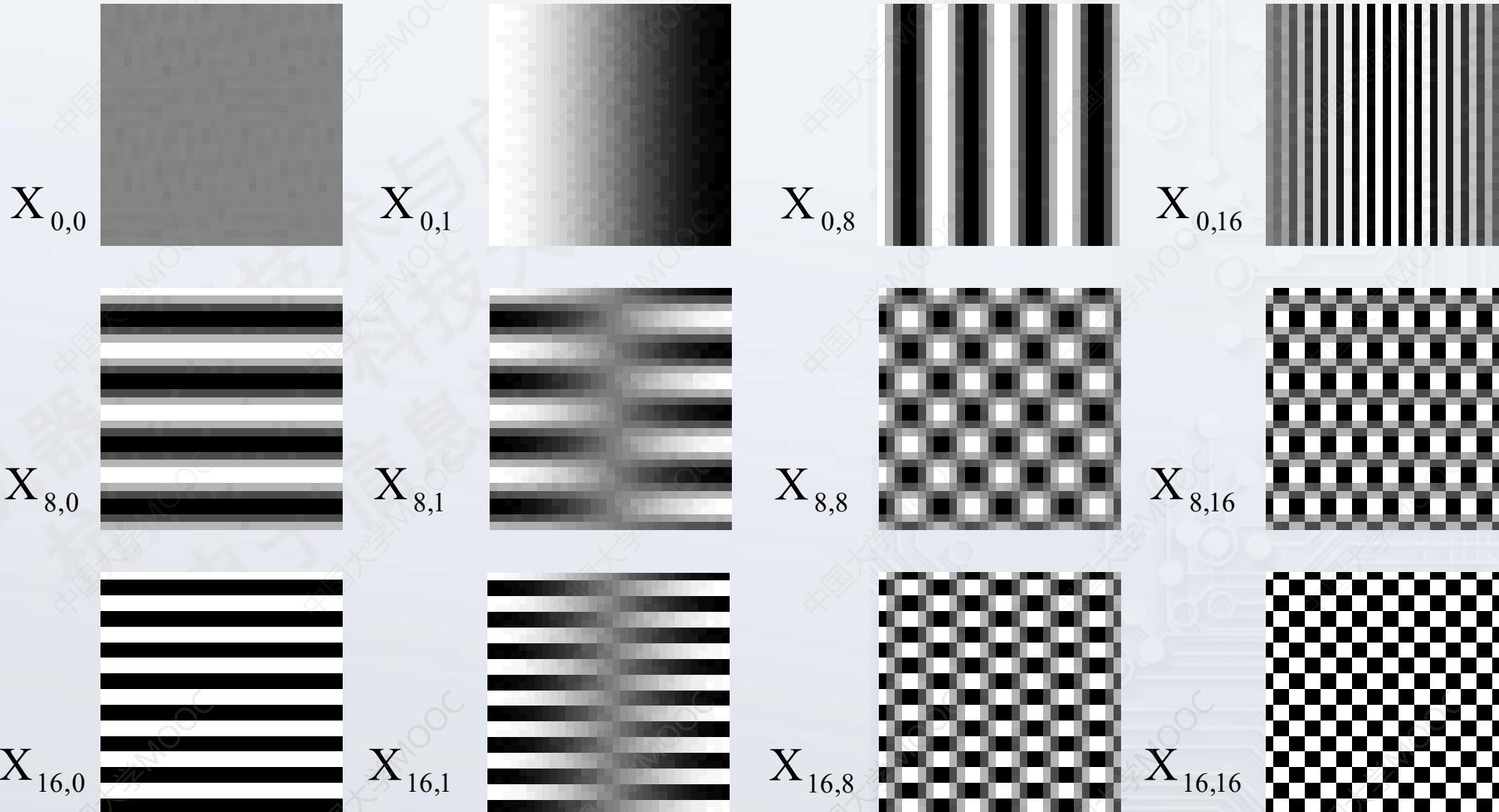
+



=



# 二维离散傅里叶变换



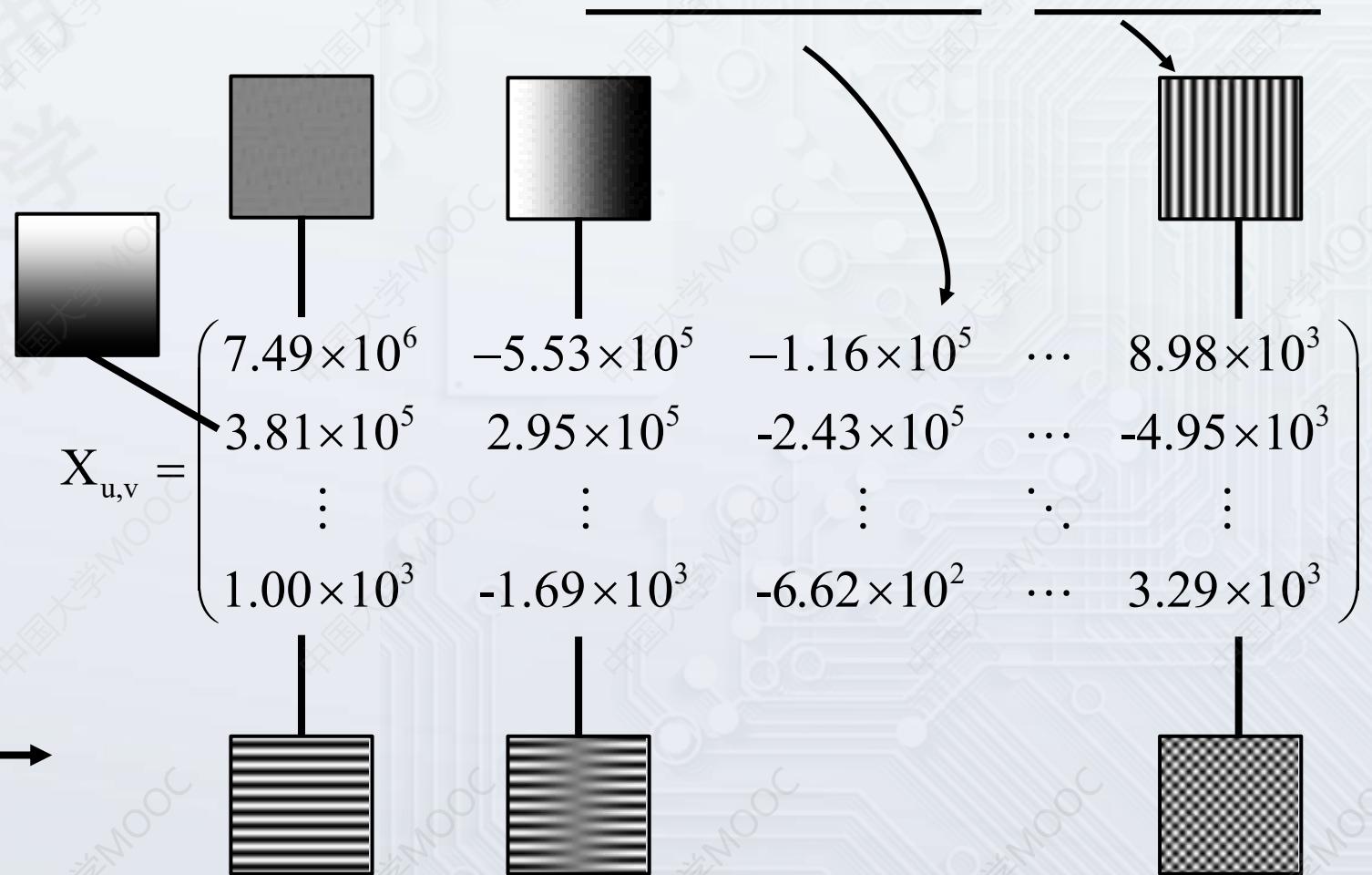
# 二维离散傅里叶变换

二维图像，进行傅里叶变换后，可以得到



# 二维离散傅里叶变换

二维图像，进行傅里叶变换后，可以得到傅里叶变换的值和对应的波形

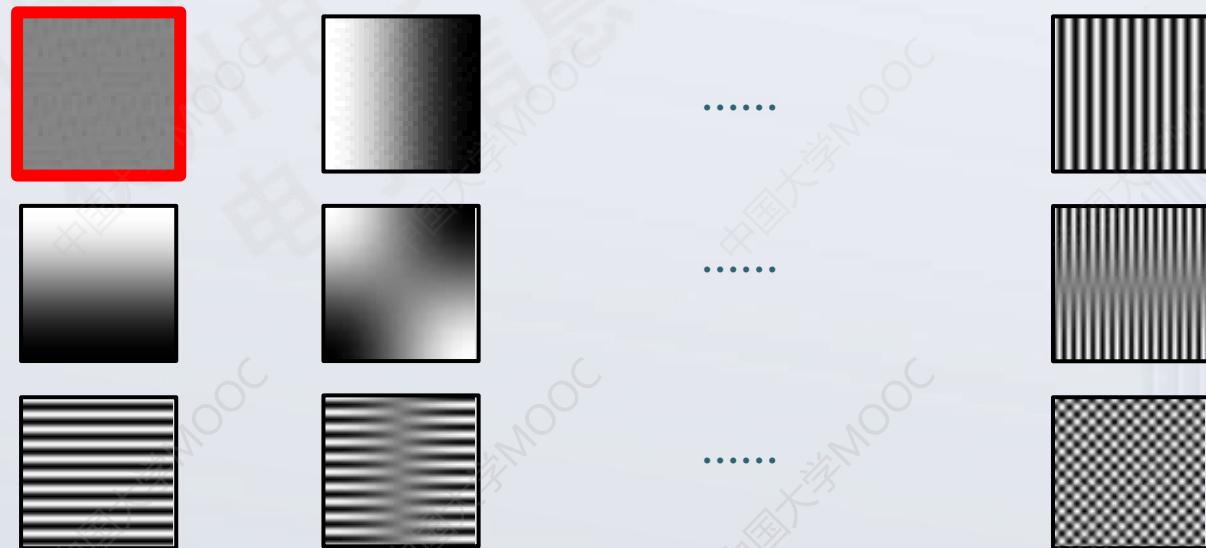


# 图像重建

傅里叶变换的值与对应的波形相乘后相加，则

$$X_{u,v} = \begin{pmatrix} 7.49 \times 10^6 & -5.53 \times 10^5 & -1.16 \times 10^5 & \dots & 8.98 \times 10^3 \\ 3.81 \times 10^5 & 2.95 \times 10^5 & -2.43 \times 10^5 & \dots & -4.95 \times 10^3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1.00 \times 10^3 & -1.69 \times 10^3 & -6.62 \times 10^2 & \dots & 3.29 \times 10^3 \end{pmatrix}$$

重建结果

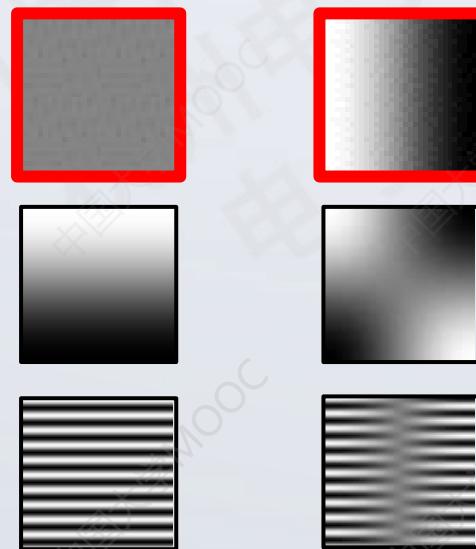


# 图像重建

傅里叶变换的值与对应的波形相乘后相加，则

$$X_{u,v} = \begin{pmatrix} 7.49 \times 10^6 & -5.53 \times 10^5 & -1.16 \times 10^5 & \dots & 8.98 \times 10^3 \\ 3.81 \times 10^5 & 2.95 \times 10^5 & -2.43 \times 10^5 & \dots & -4.95 \times 10^3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1.00 \times 10^3 & -1.69 \times 10^3 & -6.62 \times 10^2 & \dots & 3.29 \times 10^3 \end{pmatrix}$$

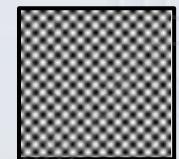
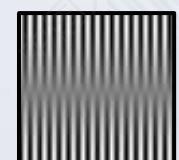
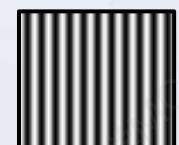
重建结果



.....

.....

.....

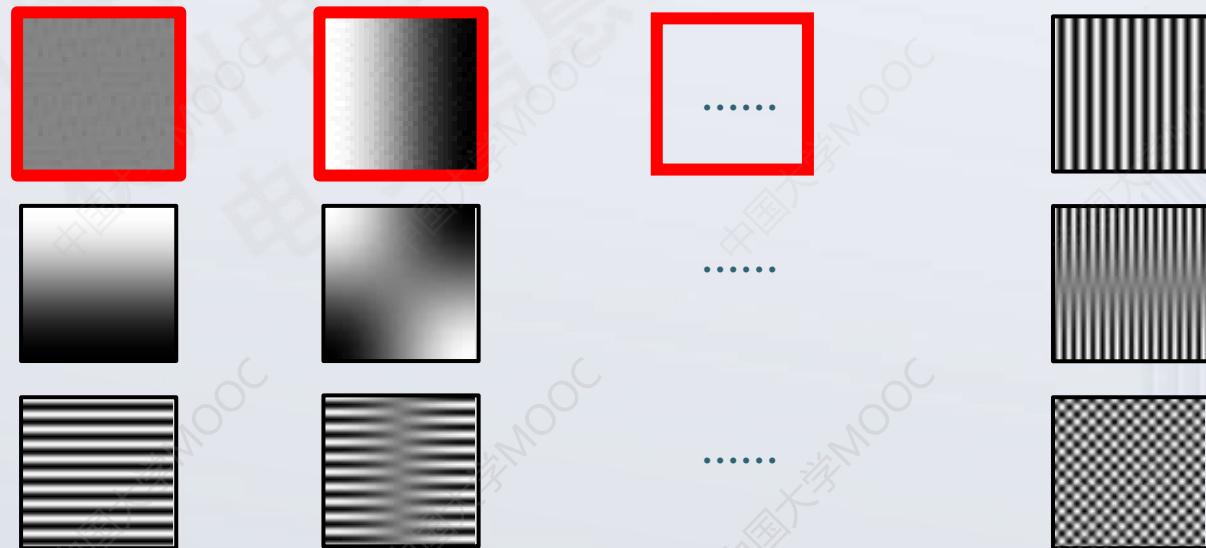


# 图像重建

傅里叶变换的值与对应的波形相乘后相加，则

$$X_{u,v} = \begin{pmatrix} 7.49 \times 10^6 & -5.53 \times 10^5 & -1.16 \times 10^5 & \dots & 8.98 \times 10^3 \\ 3.81 \times 10^5 & 2.95 \times 10^5 & -2.43 \times 10^5 & \dots & -4.95 \times 10^3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1.00 \times 10^3 & -1.69 \times 10^3 & -6.62 \times 10^2 & \dots & 3.29 \times 10^3 \end{pmatrix}$$

重建结果

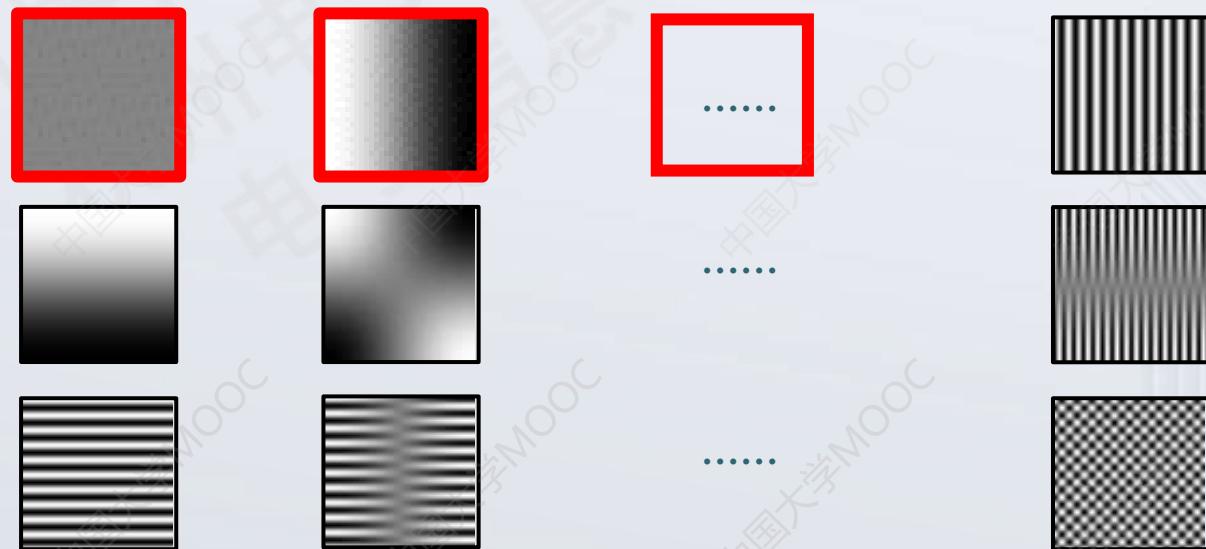
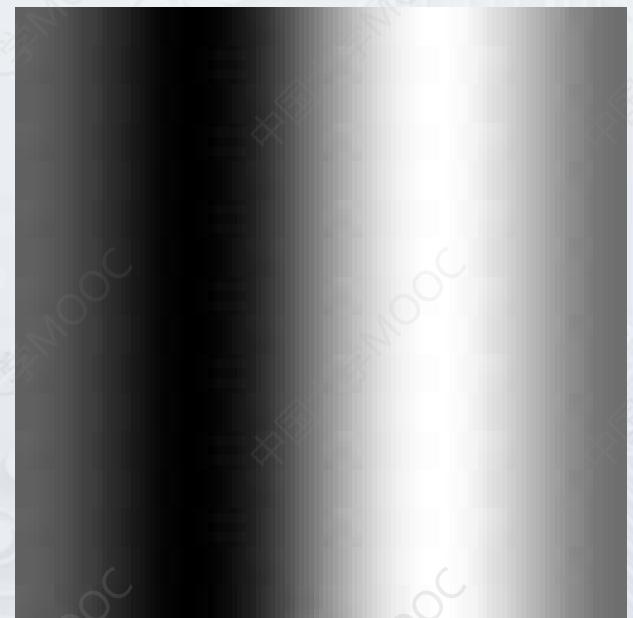


# 图像重建

傅里叶变换的值与对应的波形相乘后相加，则

$$X_{u,v} = \begin{pmatrix} 7.49 \times 10^6 & -5.53 \times 10^5 & -1.16 \times 10^5 & \dots & 8.98 \times 10^3 \\ 3.81 \times 10^5 & 2.95 \times 10^5 & -2.43 \times 10^5 & \dots & -4.95 \times 10^3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1.00 \times 10^3 & -1.69 \times 10^3 & -6.62 \times 10^2 & \dots & 3.29 \times 10^3 \end{pmatrix}$$

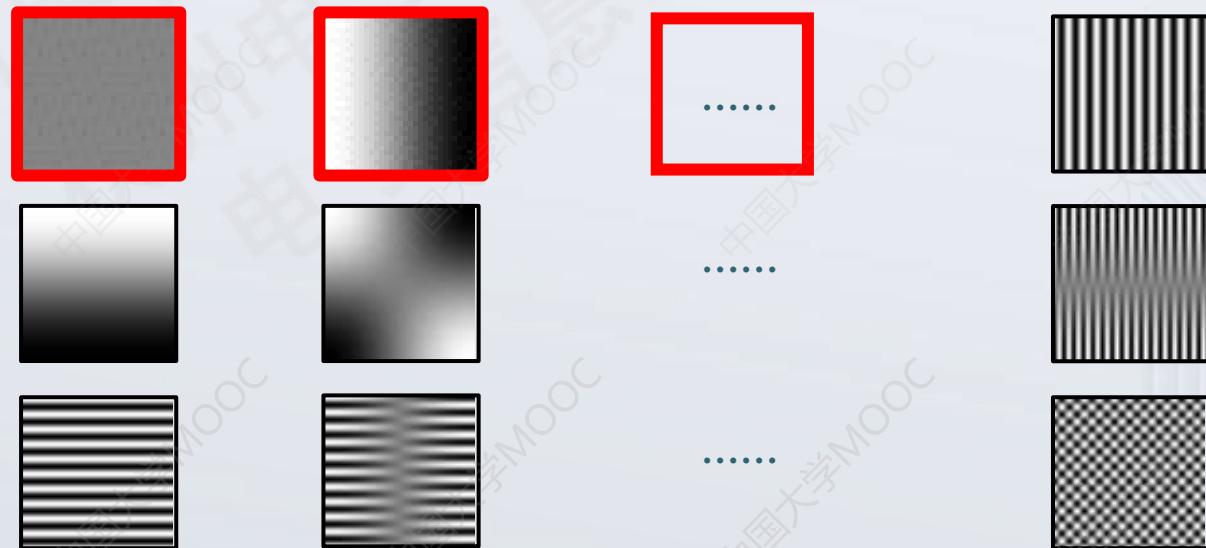
重建结果



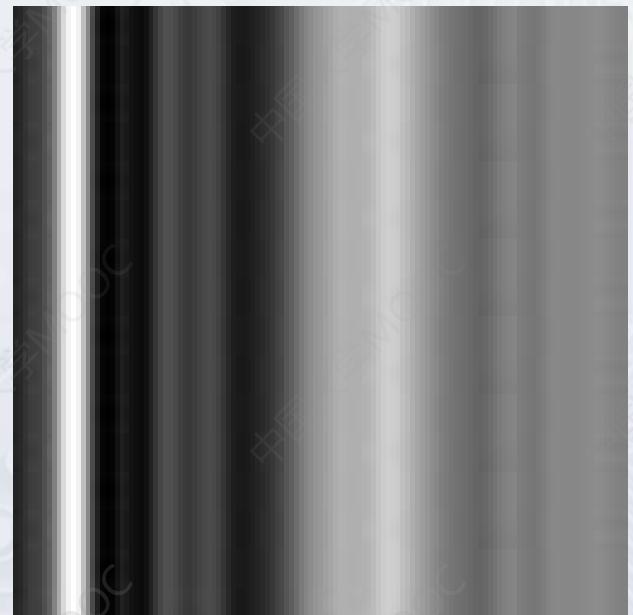
# 图像重建

傅里叶变换的值与对应的波形相乘后相加，则

$$X_{u,v} = \begin{pmatrix} 7.49 \times 10^6 & -5.53 \times 10^5 & -1.16 \times 10^5 & \dots & 8.98 \times 10^3 \\ 3.81 \times 10^5 & 2.95 \times 10^5 & -2.43 \times 10^5 & \dots & -4.95 \times 10^3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1.00 \times 10^3 & -1.69 \times 10^3 & -6.62 \times 10^2 & \dots & 3.29 \times 10^3 \end{pmatrix}$$



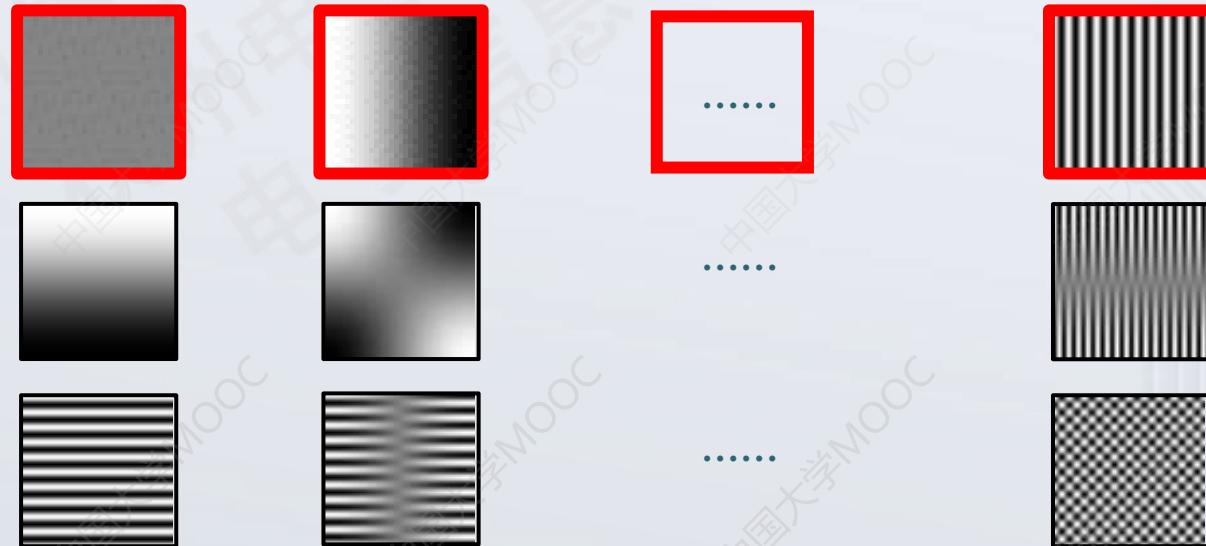
重建结果



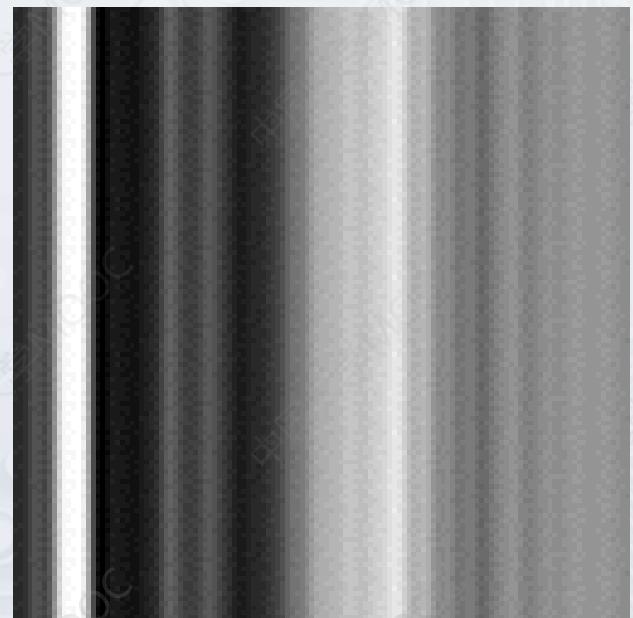
# 图像重建

傅里叶变换的值与对应的波形相乘后相加，则

$$X_{u,v} = \begin{pmatrix} 7.49 \times 10^6 & -5.53 \times 10^5 & -1.16 \times 10^5 & \dots & 8.98 \times 10^3 \\ 3.81 \times 10^5 & 2.95 \times 10^5 & -2.43 \times 10^5 & \dots & -4.95 \times 10^3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1.00 \times 10^3 & -1.69 \times 10^3 & -6.62 \times 10^2 & \dots & 3.29 \times 10^3 \end{pmatrix}$$



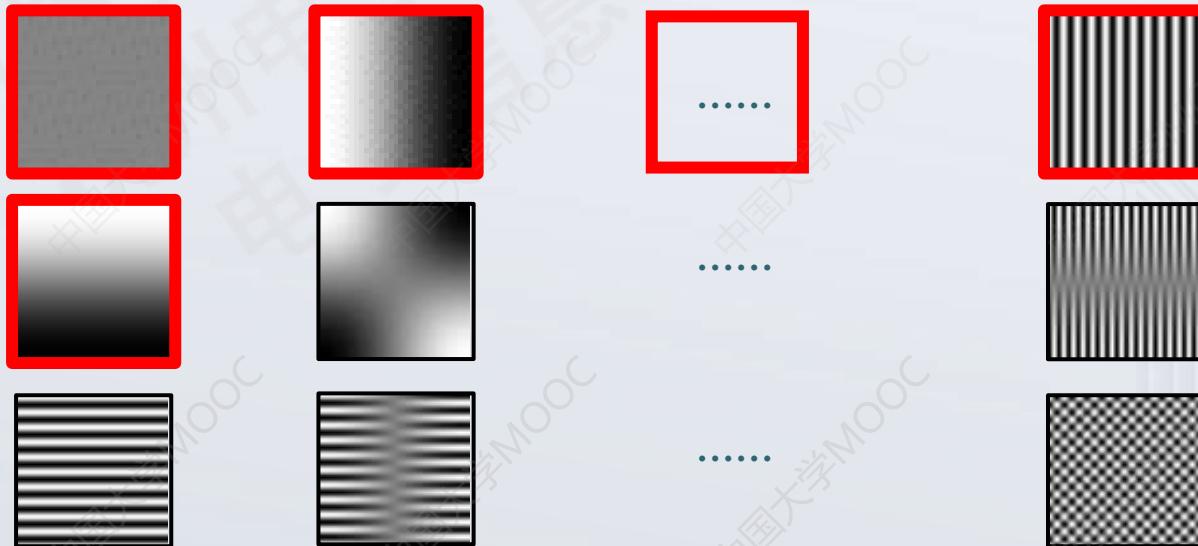
重建结果



# 图像重建

傅里叶变换的值与对应的波形相乘后相加，则

$$X_{u,v} = \begin{pmatrix} 7.49 \times 10^6 & -5.53 \times 10^5 & -1.16 \times 10^5 & \cdots & 8.98 \times 10^3 \\ 3.81 \times 10^5 & 2.95 \times 10^5 & -2.43 \times 10^5 & \cdots & -4.95 \times 10^3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1.00 \times 10^3 & -1.69 \times 10^3 & -6.62 \times 10^2 & \cdots & 3.29 \times 10^3 \end{pmatrix}$$



重建结果

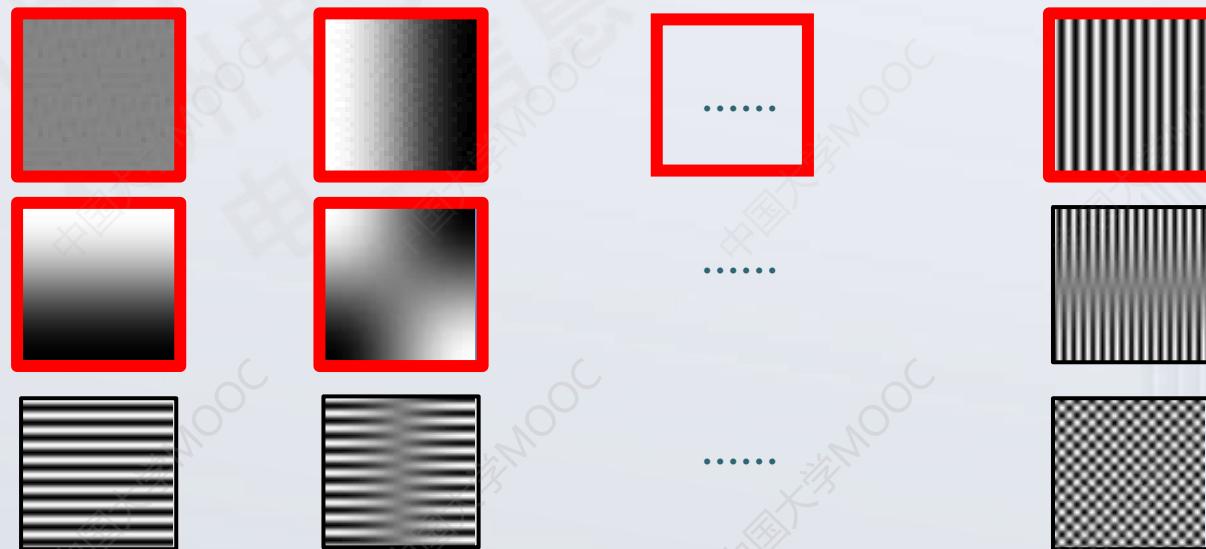
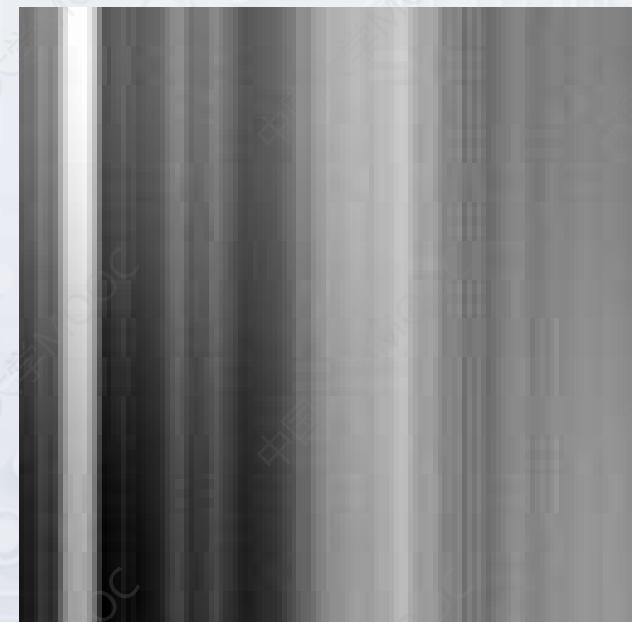


# 图像重建

傅里叶变换的值与对应的波形相乘后相加，则

$$X_{u,v} = \begin{pmatrix} 7.49 \times 10^6 & -5.53 \times 10^5 & -1.16 \times 10^5 & \cdots & 8.98 \times 10^3 \\ 3.81 \times 10^5 & 2.95 \times 10^5 & -2.43 \times 10^5 & \cdots & -4.95 \times 10^3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1.00 \times 10^3 & -1.69 \times 10^3 & -6.62 \times 10^2 & \cdots & 3.29 \times 10^3 \end{pmatrix}$$

重建结果

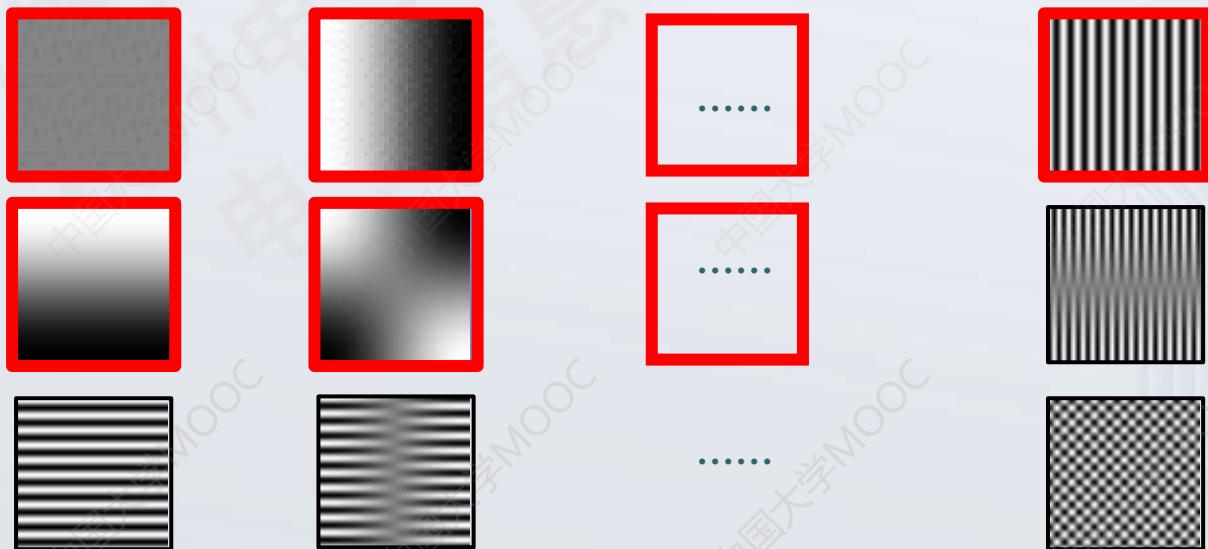
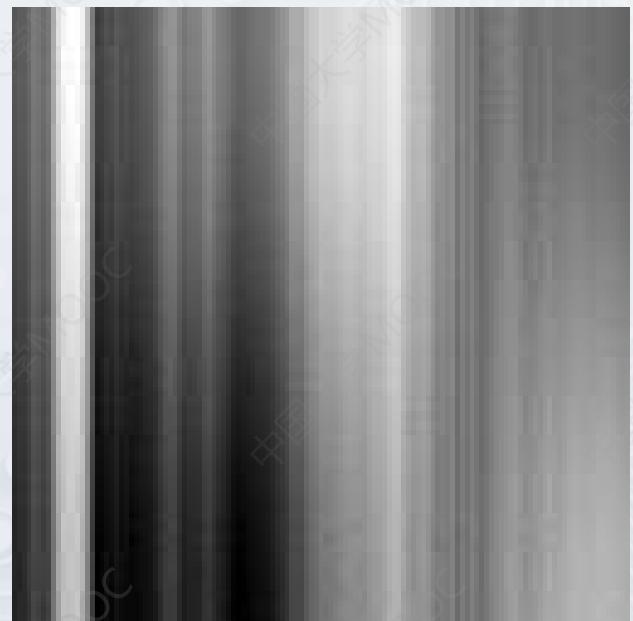


# 图像重建

傅里叶变换的值与对应的波形相乘后相加，则

$$X_{u,v} = \begin{pmatrix} 7.49 \times 10^6 & -5.53 \times 10^5 & -1.16 \times 10^5 & \dots & 8.98 \times 10^3 \\ 3.81 \times 10^5 & 2.95 \times 10^5 & -2.43 \times 10^5 & \dots & -4.95 \times 10^3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1.00 \times 10^3 & -1.69 \times 10^3 & -6.62 \times 10^2 & \dots & 3.29 \times 10^3 \end{pmatrix}$$

重建结果

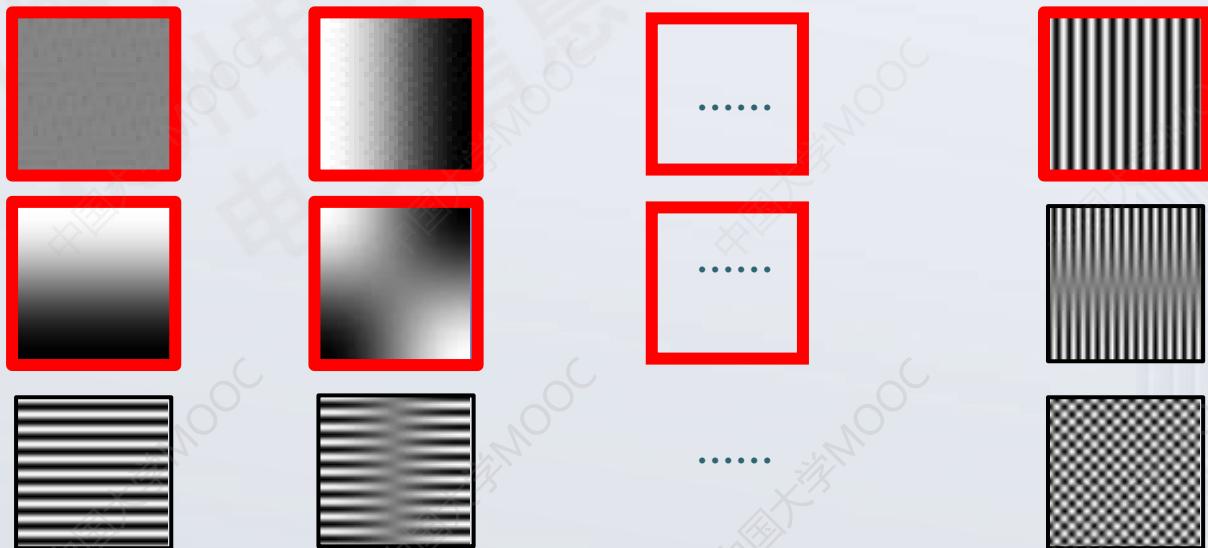
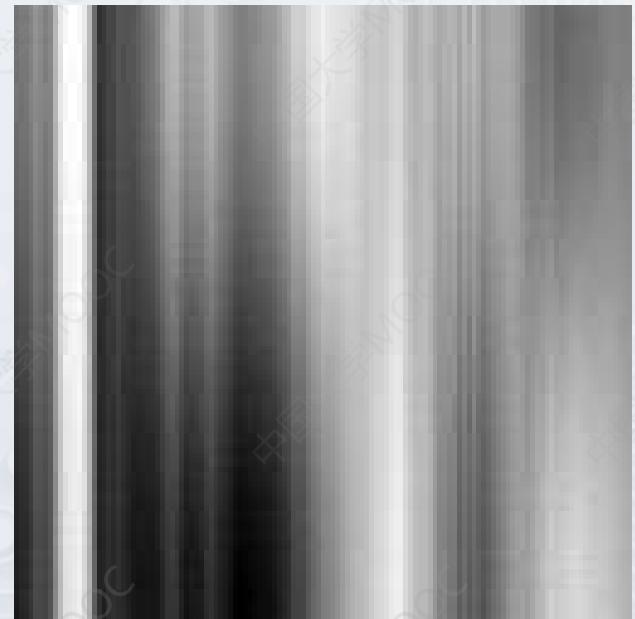


# 图像重建

傅里叶变换的值与对应的波形相乘后相加，则

$$X_{u,v} = \begin{pmatrix} 7.49 \times 10^6 & -5.53 \times 10^5 & -1.16 \times 10^5 & \dots & 8.98 \times 10^3 \\ 3.81 \times 10^5 & 2.95 \times 10^5 & -2.43 \times 10^5 & \dots & -4.95 \times 10^3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1.00 \times 10^3 & -1.69 \times 10^3 & -6.62 \times 10^2 & \dots & 3.29 \times 10^3 \end{pmatrix}$$

重建结果

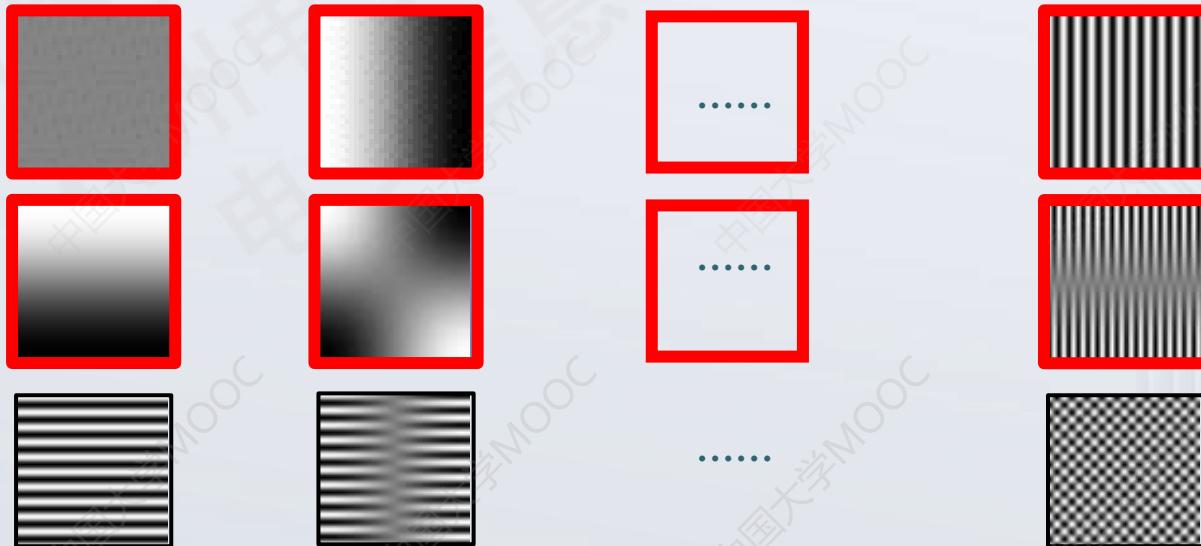
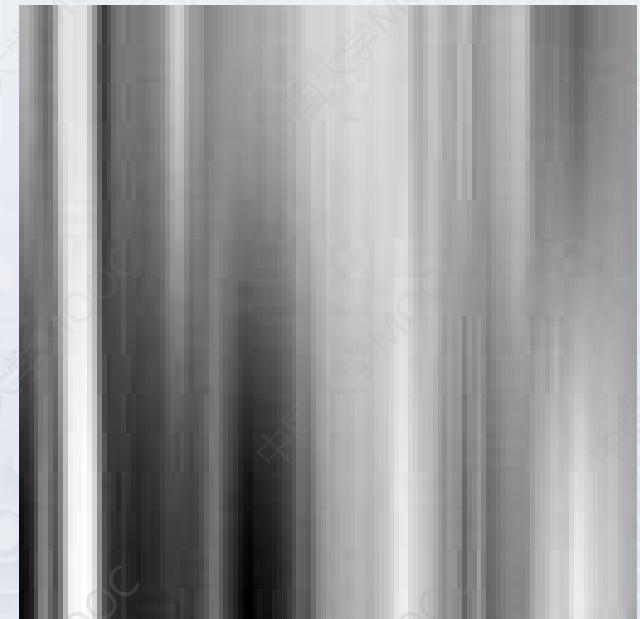


# 图像重建

傅里叶变换的值与对应的波形相乘后相加，则

$$X_{u,v} = \begin{pmatrix} 7.49 \times 10^6 & -5.53 \times 10^5 & -1.16 \times 10^5 & \cdots & 8.98 \times 10^3 \\ 3.81 \times 10^5 & 2.95 \times 10^5 & -2.43 \times 10^5 & \cdots & -4.95 \times 10^3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1.00 \times 10^3 & -1.69 \times 10^3 & -6.62 \times 10^2 & \cdots & 3.29 \times 10^3 \end{pmatrix}$$

重建结果

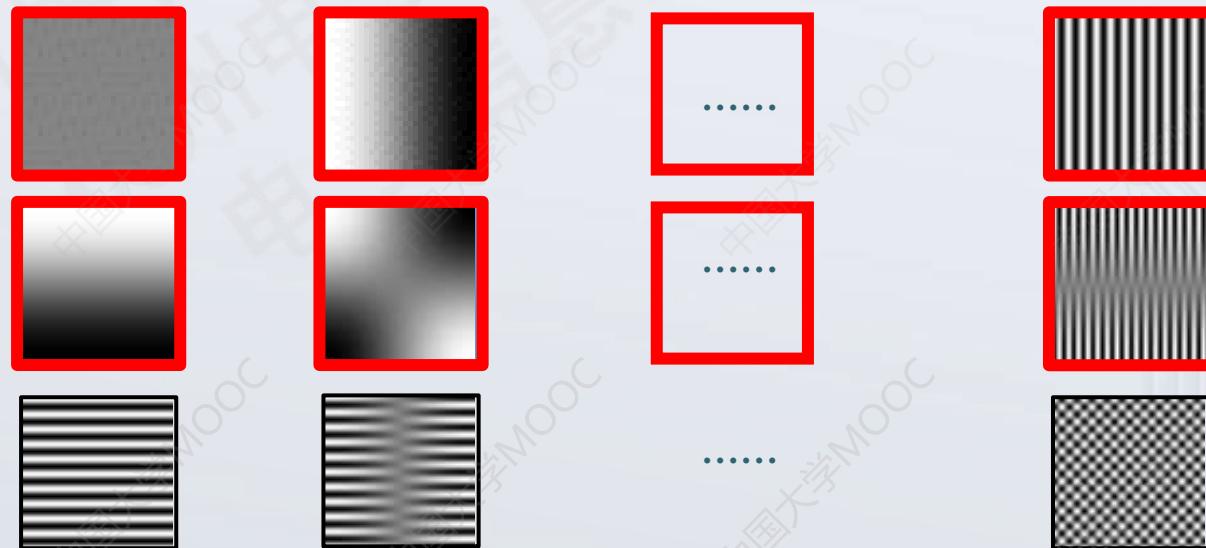
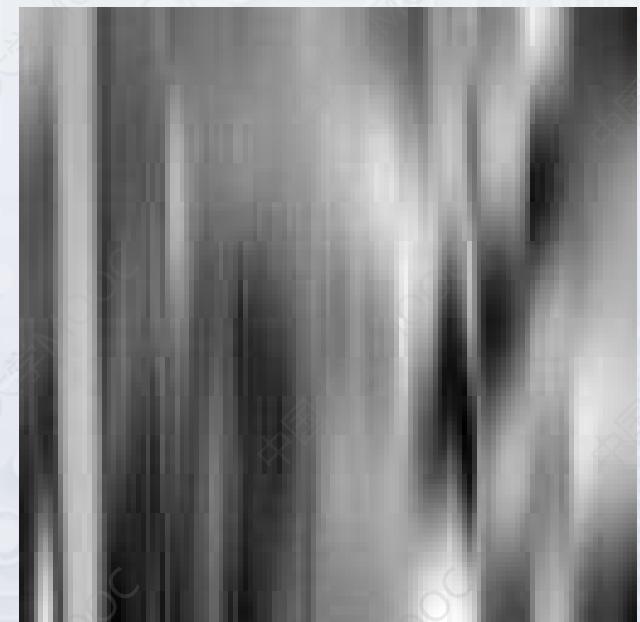


# 图像重建

傅里叶变换的值与对应的波形相乘后相加，则

$$X_{u,v} = \begin{pmatrix} 7.49 \times 10^6 & -5.53 \times 10^5 & -1.16 \times 10^5 & \cdots & 8.98 \times 10^3 \\ 3.81 \times 10^5 & 2.95 \times 10^5 & -2.43 \times 10^5 & \cdots & -4.95 \times 10^3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1.00 \times 10^3 & -1.69 \times 10^3 & -6.62 \times 10^2 & \cdots & 3.29 \times 10^3 \end{pmatrix}$$

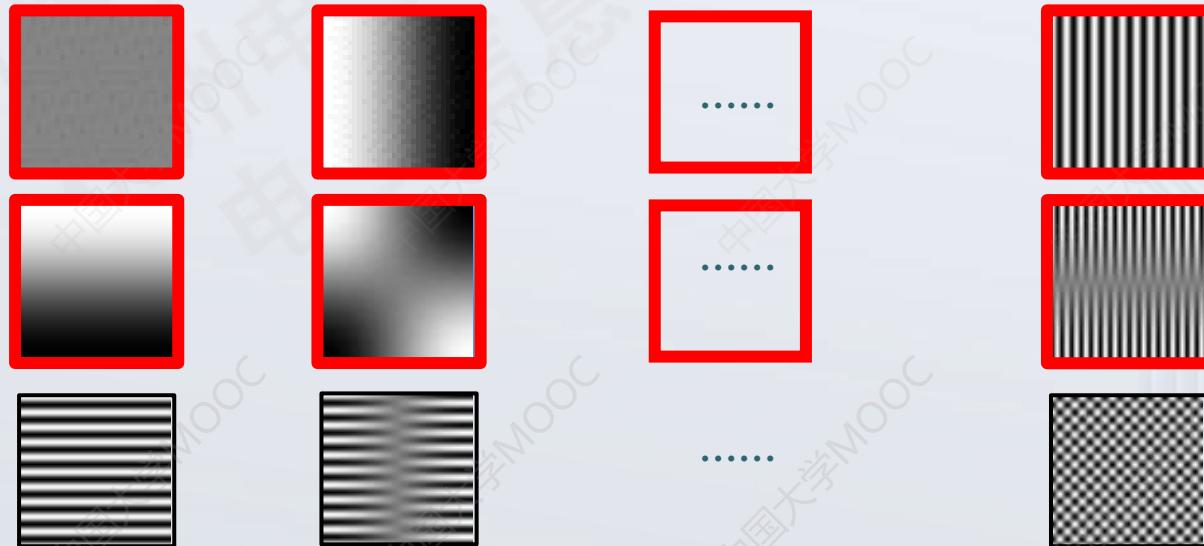
重建结果



# 图像重建

傅里叶变换的值与对应的波形相乘后相加，则

$$X_{u,v} = \begin{pmatrix} 7.49 \times 10^6 & -5.53 \times 10^5 & -1.16 \times 10^5 & \cdots & 8.98 \times 10^3 \\ 3.81 \times 10^5 & 2.95 \times 10^5 & -2.43 \times 10^5 & \cdots & -4.95 \times 10^3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1.00 \times 10^3 & -1.69 \times 10^3 & -6.62 \times 10^2 & \cdots & 3.29 \times 10^3 \end{pmatrix}$$



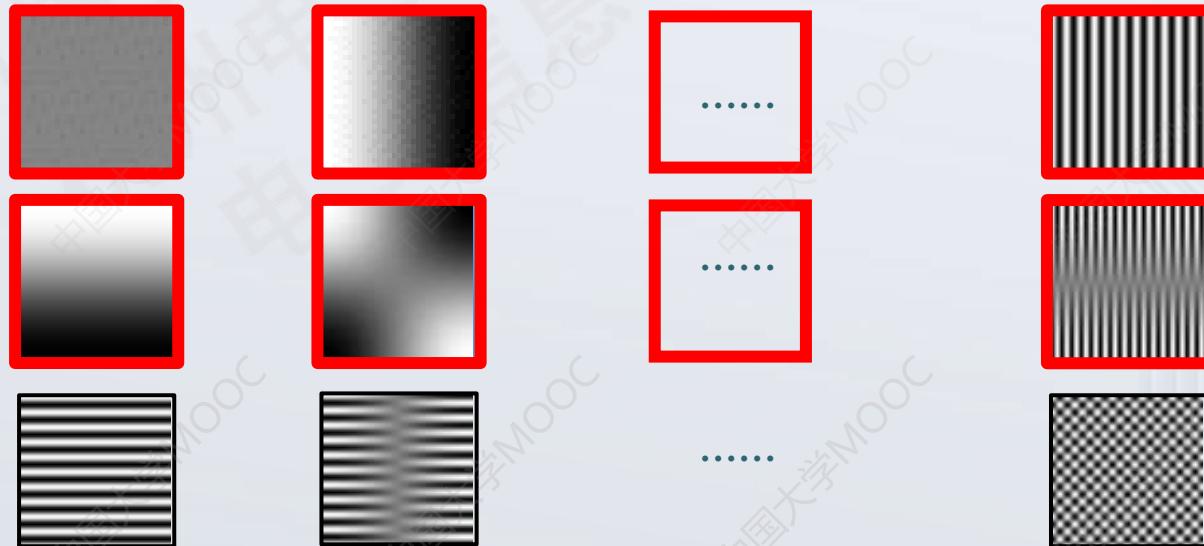
重建结果



# 图像重建

傅里叶变换的值与对应的波形相乘后相加，则

$$X_{u,v} = \begin{pmatrix} 7.49 \times 10^6 & -5.53 \times 10^5 & -1.16 \times 10^5 & \dots & 8.98 \times 10^3 \\ 3.81 \times 10^5 & 2.95 \times 10^5 & -2.43 \times 10^5 & \dots & -4.95 \times 10^3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1.00 \times 10^3 & -1.69 \times 10^3 & -6.62 \times 10^2 & \dots & 3.29 \times 10^3 \end{pmatrix}$$



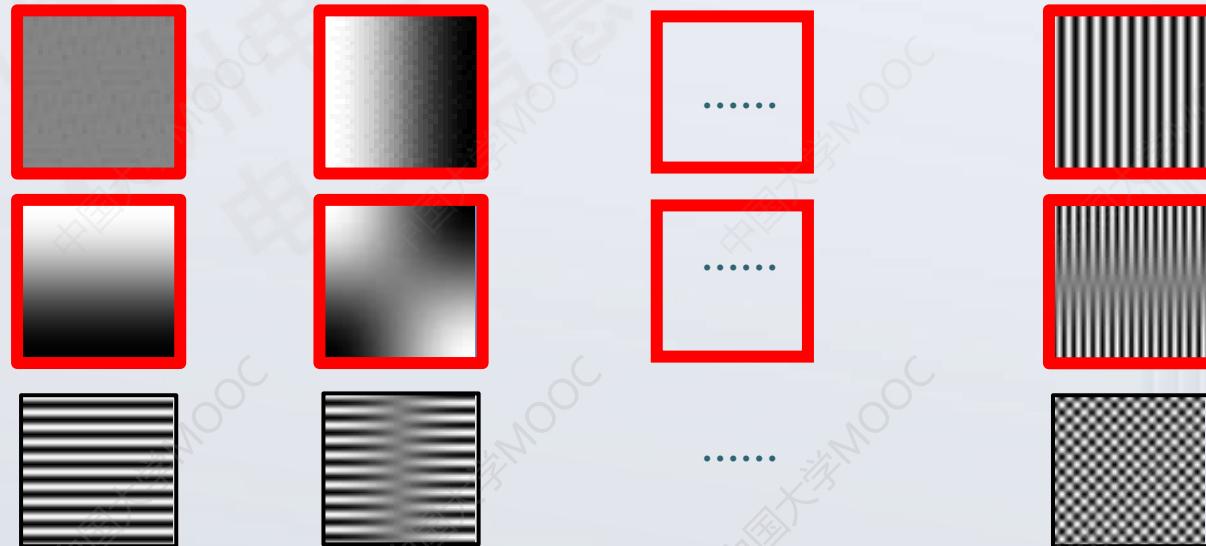
重建结果



# 图像重建

傅里叶变换的值与对应的波形相乘后相加，则

$$X_{u,v} = \begin{pmatrix} 7.49 \times 10^6 & -5.53 \times 10^5 & -1.16 \times 10^5 & \cdots & 8.98 \times 10^3 \\ 3.81 \times 10^5 & 2.95 \times 10^5 & -2.43 \times 10^5 & \cdots & -4.95 \times 10^3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1.00 \times 10^3 & -1.69 \times 10^3 & -6.62 \times 10^2 & \cdots & 3.29 \times 10^3 \end{pmatrix}$$



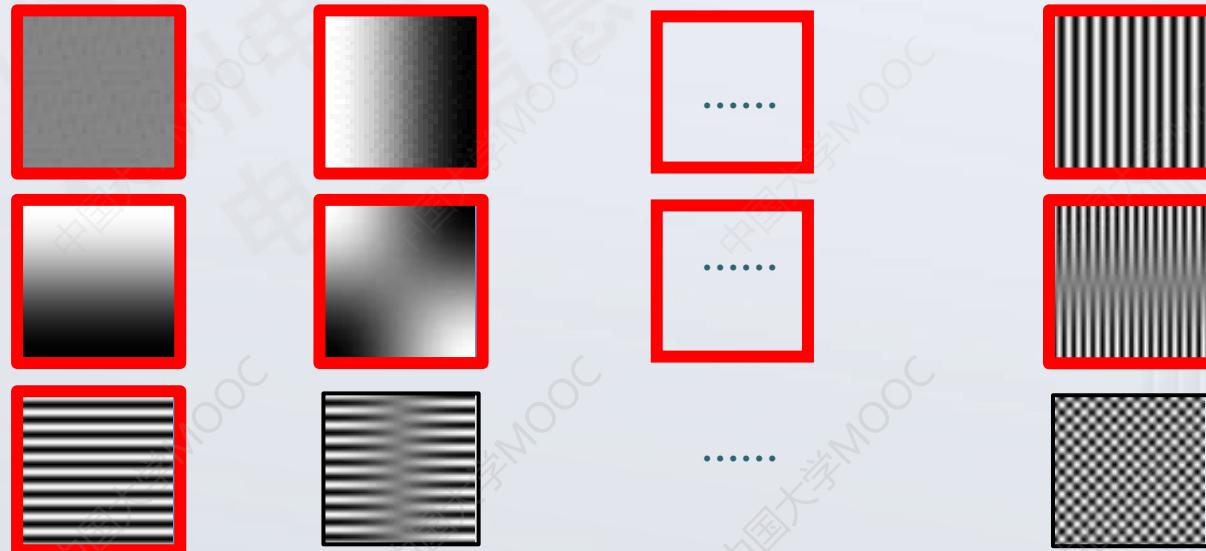
重建结果



# 图像重建

傅里叶变换的值与对应的波形相乘后相加，则

$$X_{u,v} = \begin{pmatrix} 7.49 \times 10^6 & -5.53 \times 10^5 & -1.16 \times 10^5 & \cdots & 8.98 \times 10^3 \\ 3.81 \times 10^5 & 2.95 \times 10^5 & -2.43 \times 10^5 & \cdots & -4.95 \times 10^3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1.00 \times 10^3 & -1.69 \times 10^3 & -6.62 \times 10^2 & \cdots & 3.29 \times 10^3 \end{pmatrix}$$



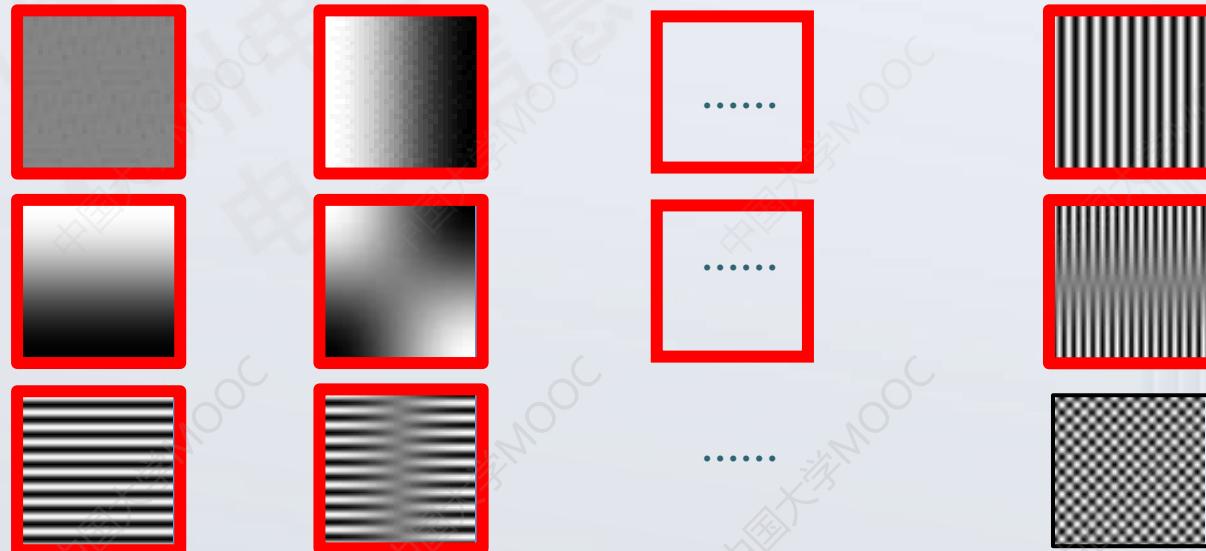
重建结果



# 图像重建

傅里叶变换的值与对应的波形相乘后相加，则

$$X_{u,v} = \begin{pmatrix} 7.49 \times 10^6 & -5.53 \times 10^5 & -1.16 \times 10^5 & \cdots & 8.98 \times 10^3 \\ 3.81 \times 10^5 & 2.95 \times 10^5 & -2.43 \times 10^5 & \cdots & -4.95 \times 10^3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1.00 \times 10^3 & -1.69 \times 10^3 & -6.62 \times 10^2 & \cdots & 3.29 \times 10^3 \end{pmatrix}$$



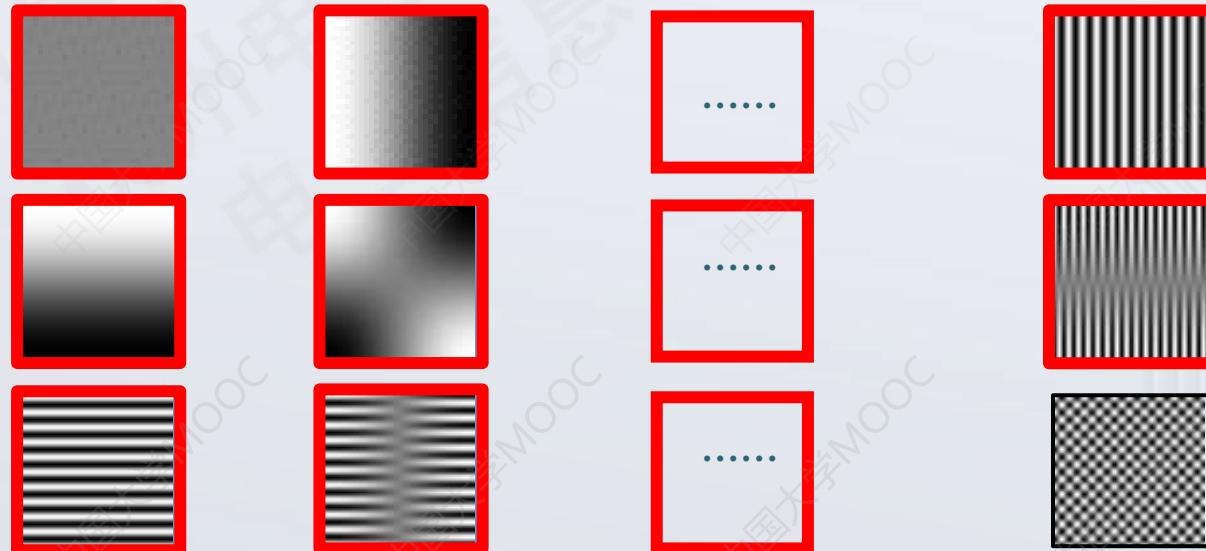
重建结果



# 图像重建

傅里叶变换的值与对应的波形相乘后相加，则

$$X_{u,v} = \begin{pmatrix} 7.49 \times 10^6 & -5.53 \times 10^5 & -1.16 \times 10^5 & \cdots & 8.98 \times 10^3 \\ 3.81 \times 10^5 & 2.95 \times 10^5 & -2.43 \times 10^5 & \cdots & -4.95 \times 10^3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1.00 \times 10^3 & -1.69 \times 10^3 & -6.62 \times 10^2 & \cdots & 3.29 \times 10^3 \end{pmatrix}$$



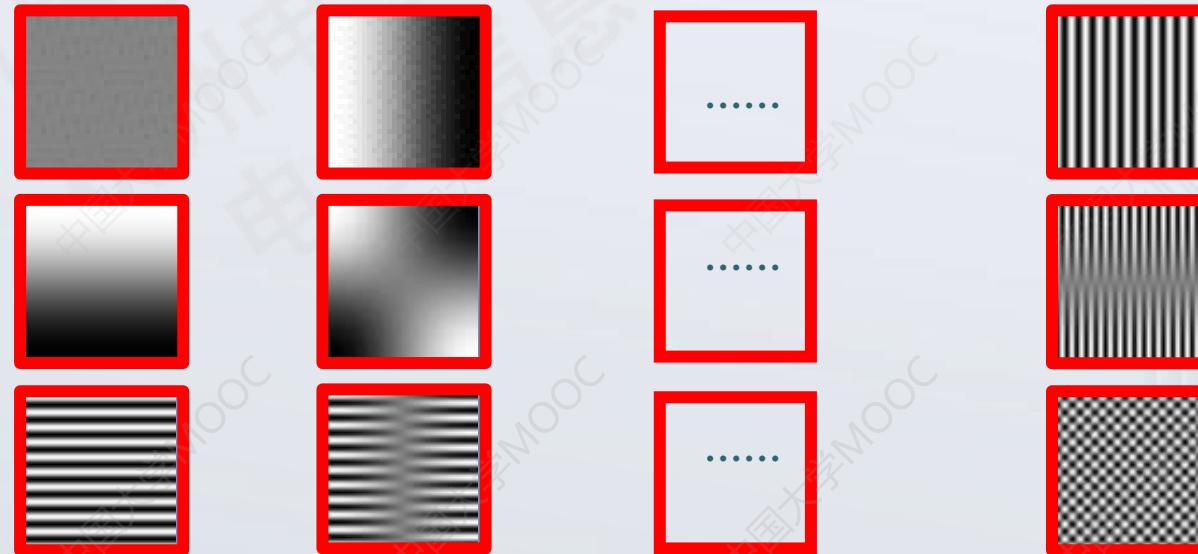
重建结果



# 图像重建

傅里叶变换的值与对应的波形相乘后相加，则

$$X_{u,v} = \begin{pmatrix} 7.49 \times 10^6 & -5.53 \times 10^5 & -1.16 \times 10^5 & \cdots & 8.98 \times 10^3 \\ 3.81 \times 10^5 & 2.95 \times 10^5 & -2.43 \times 10^5 & \cdots & -4.95 \times 10^3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1.00 \times 10^3 & -1.69 \times 10^3 & -6.62 \times 10^2 & \cdots & 3.29 \times 10^3 \end{pmatrix}$$



重建结果



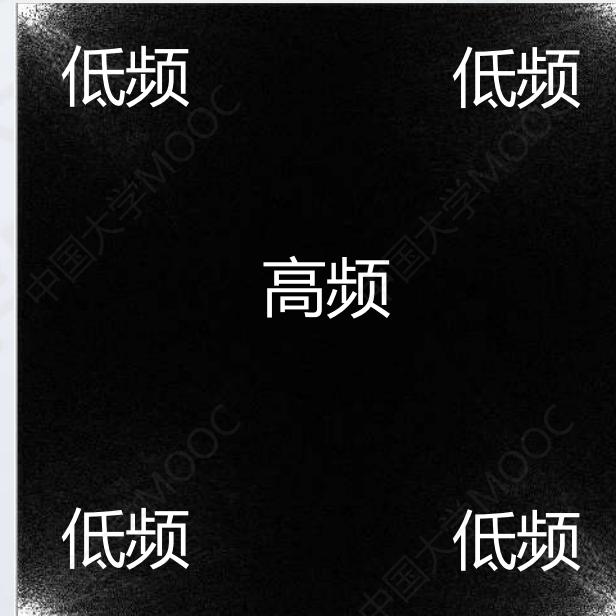
重建完成！

# 图像的频谱图

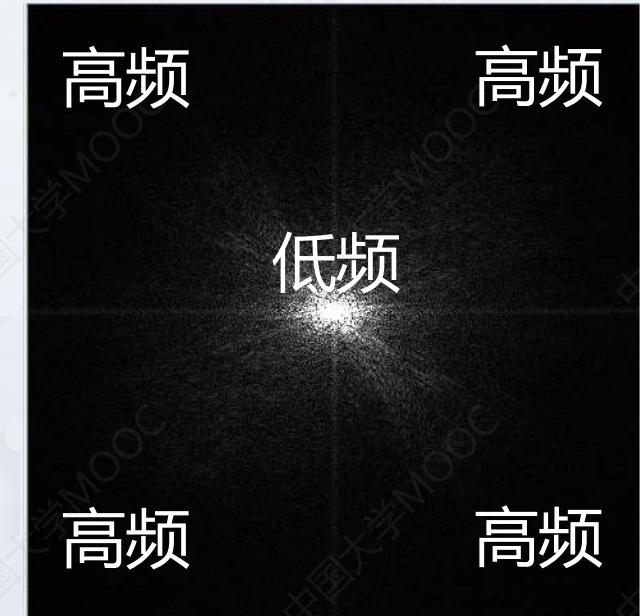
原图



频谱图



平移后的频谱图

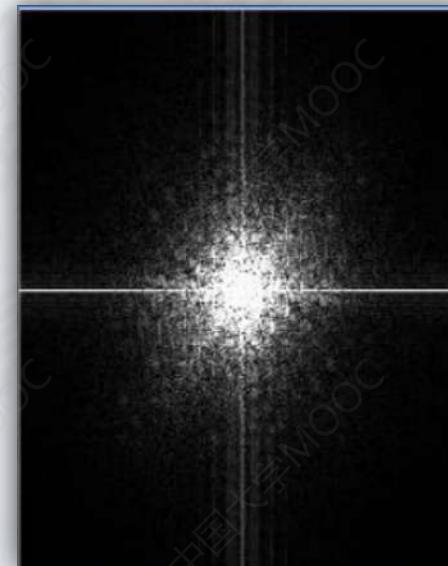


# 图像的频谱图

原图



频谱图



轮廓的基本信息

边缘，线条，纹理

区域

噪声

# 图像的频谱图

低对比度图像

频谱图



高对比度图像

频谱图



# 图像的频谱图

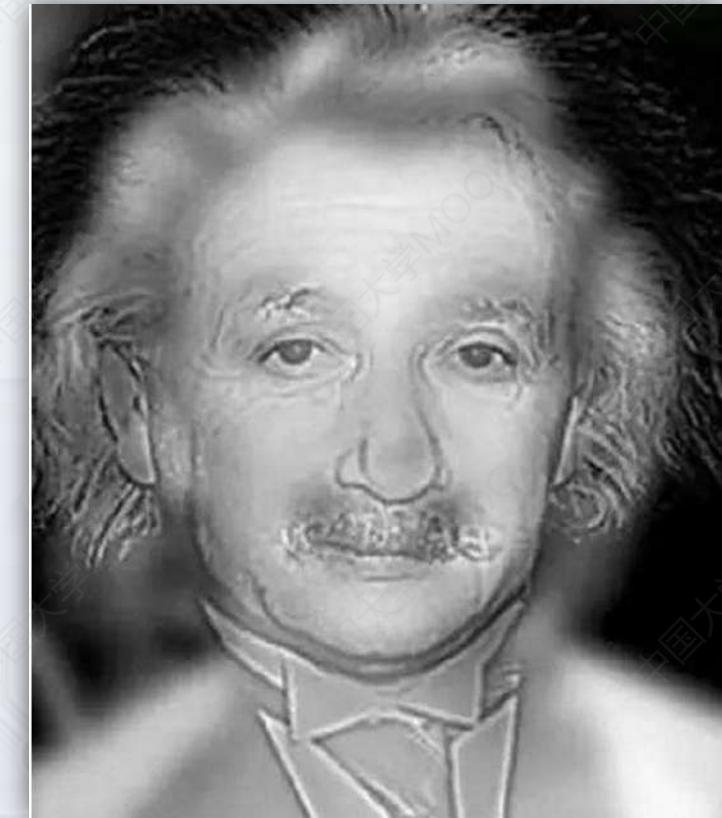
图像及频谱图



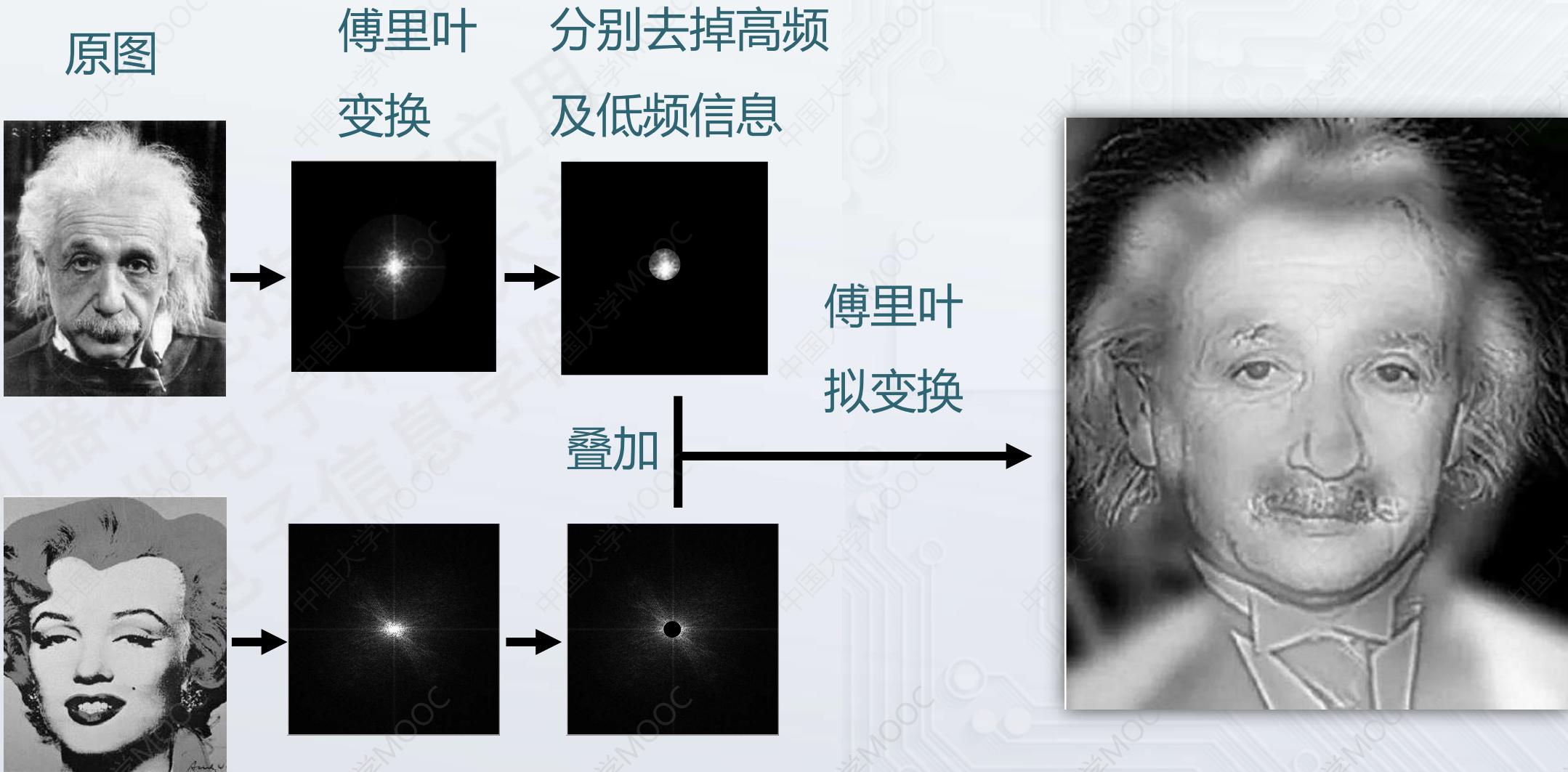
去掉一些频率信息及对应的重建图像



# 频率域处理的应用



# 频率域处理的应用



# 缺陷检测

零件，标签，板材，布料等，如果要检测上面存在的缺陷？

标签

缺陷



# 缺陷检测

零件，标签，板材，布料等，如果要检测上面存在的缺陷？

标签

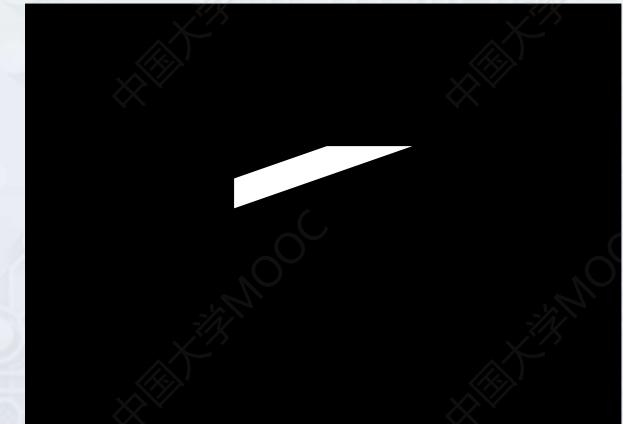


缺陷

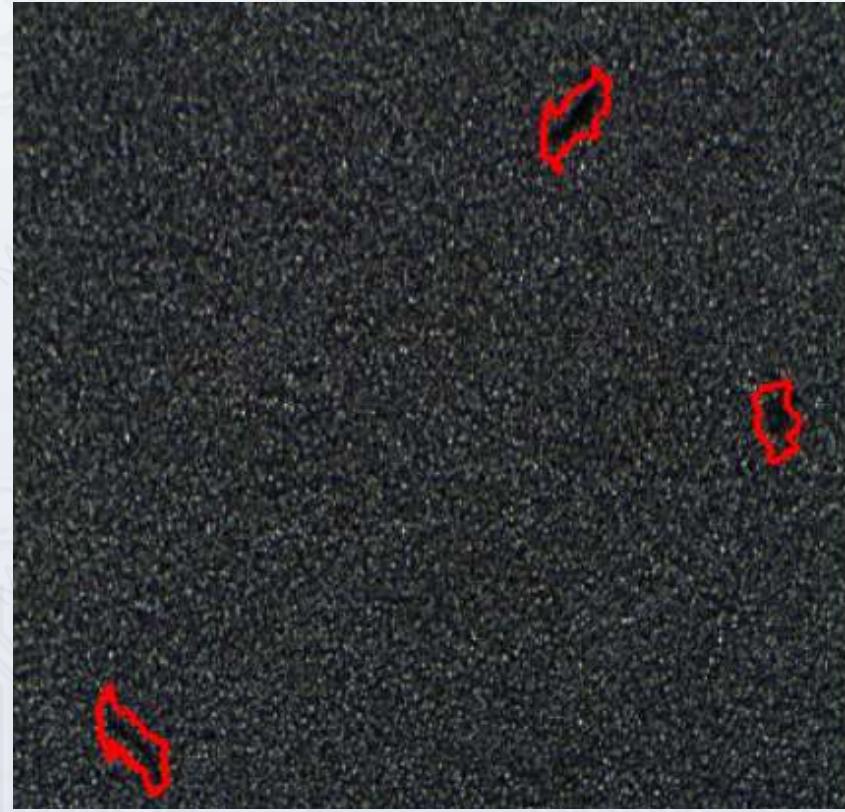
模板



缺陷



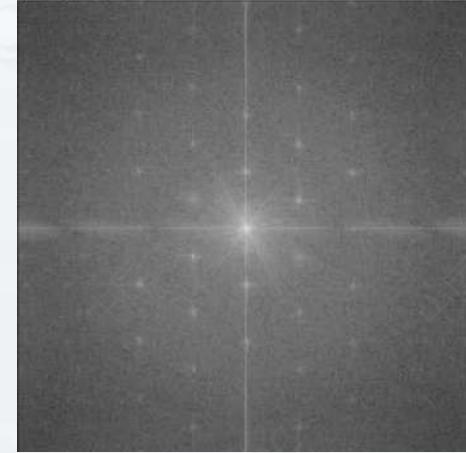
# 缺陷检测



# 图像复原

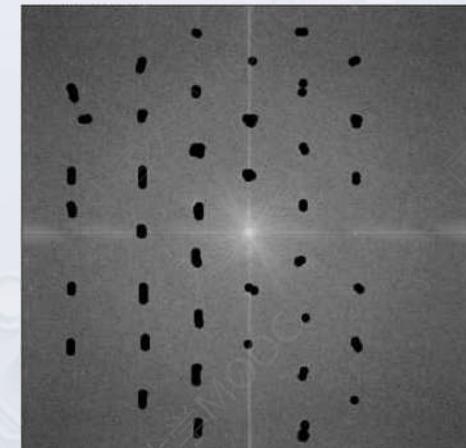


DFT  
→



Filters some  
frequency

←  
Inverse  
DFT



# 谢谢！

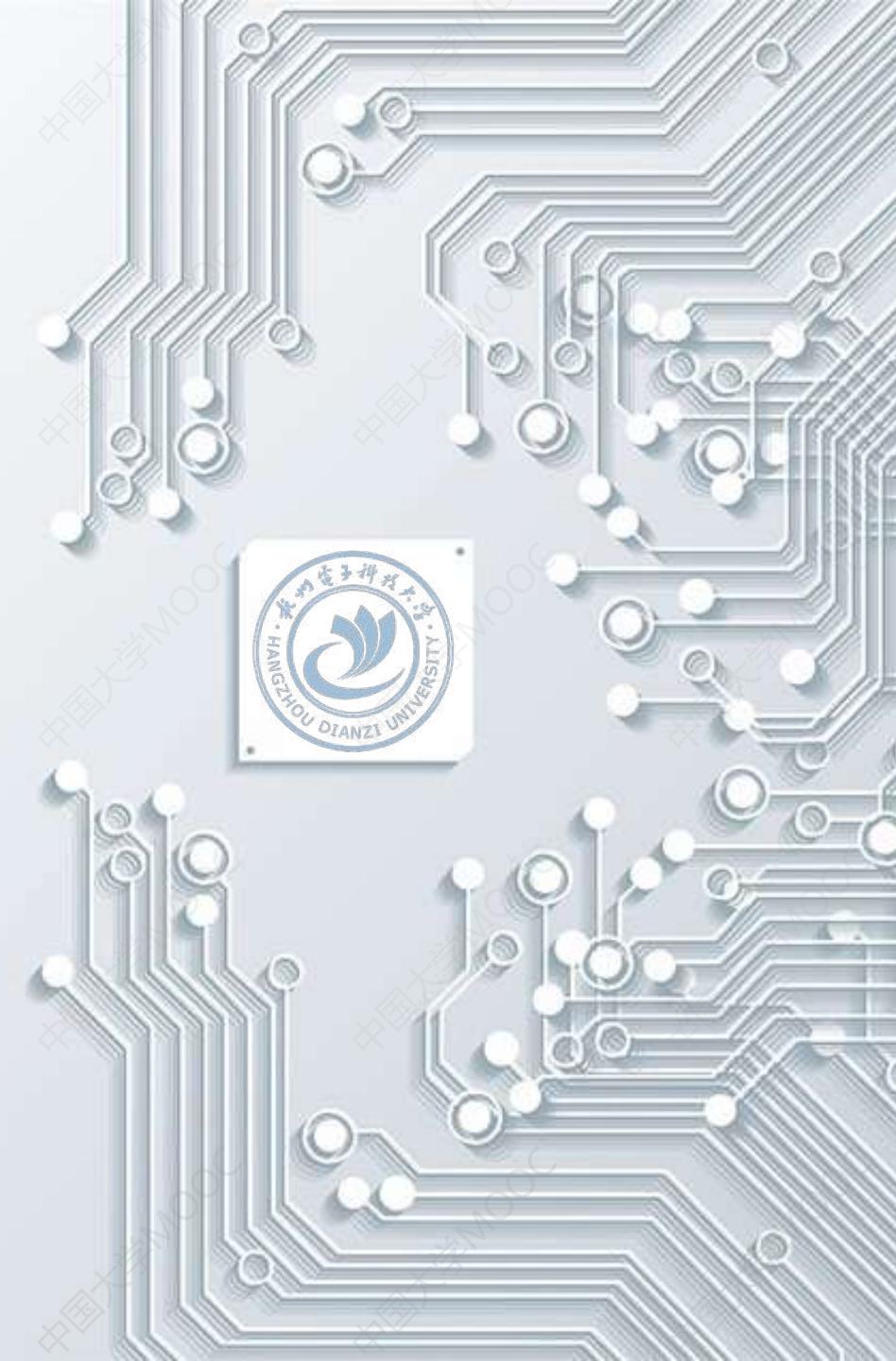
# 机器视觉技术与应用

## 10. 图像压缩

李竹

杭州电子科技大学

电子信息学院



# 本章概要

1. 图像压缩的目的
2. 图像压缩的原理
3. JPEG的压缩原理
4. 霍夫曼编码

# 图像压缩的必要性

计算机中的数字图像，通常用8bit来进行量化。假设，有一副 $512\times 512$ 的分辨率的图像，则数据量为：

$$512\times 512\times 8\text{bit}=2048\text{Kbit}=2\text{Mbit}$$

医学图图像量化有时候使用12bit的图像，如果图像分辨率为 $1024\times 1024$ ，则数据量为：

$$1024\times 1024\times 12\text{bit}=12\text{Mbit}$$

如果是视频文件，假设是一段90分钟的 $512\times 512$ 分辨率的彩色电影，24frame/s，则数据量为：

$$90\times 60\times 24\times 3\times 512\times 512\times 8\text{bit}=97,200\text{M}$$

# 图像压缩原理

图像中存在冗余信息。

图像中有些数据代表了无用信息，有些信息是重复的。这就为图像压缩带来了可能性。减少冗余信息即可达到图像压缩的目的。

三种基本的数据冗余。

- 编码冗余
- 像素间冗余
- 心理冗余

# 编码冗余

如果一个图像中的灰度级编码，使用了多于实际需要的编码符号，就称为图像中包含了编码冗余。

如左图，如果使用8位图像来表示，我们就说该图像中存在着编码冗余。

因为该图像中只存在2个灰度级。



# 像素间冗余

反映图像中相互关系的信息，任何给定像素的值可以根据这个像素相邻的像素进行预测。

原始图像数据为： 234 223 231 238 235

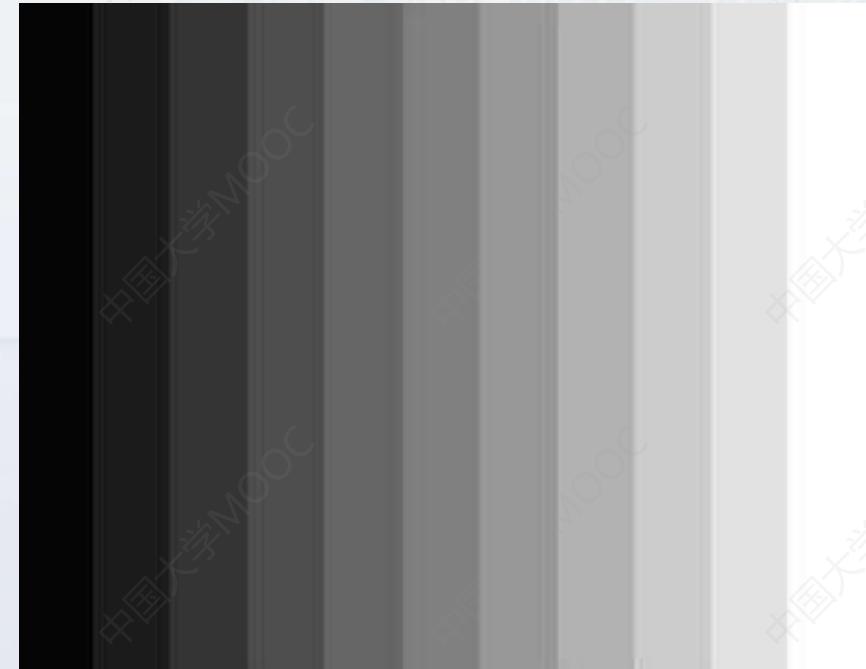
压缩后的数据为： 234 -11 8 7 -3



# 心理视觉冗余

人的眼睛感觉到的图像，不仅仅取决于该区域的反射光。还和人的大脑有关。

- 人类对所有的视觉信息感受的敏感度不同。不同信息对视觉的重要程度不同
- 重要性较低的信息就被认为是心理冗余



马赫带效应

# JPEG的压缩原理

YCbCr和RGB图像的转换方式

在RGB三种颜色中，人类对绿色及红色的敏感程度较大。蓝色的敏感程度较小。假设红色和蓝色对亮度的贡献程度分别为 $K_R$ 和 $K_B$

$$Y = K_R \cdot R + (1 - K_R - K_B) \cdot G + K_B \cdot B$$

通常， $K_R = 0.299$ ， $K_B = 0.114$

$$Y = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B$$

蓝色和红色的色差：

$$C_b = \frac{1}{2} \cdot \frac{B - Y}{1 - K_B} \quad C_r = \frac{1}{2} \cdot \frac{R - Y}{1 - K_R}$$

# JPEG的压缩原理

网膜上有两种感光细胞，能够感知亮度变化的视杆细胞，以及能够感知颜色的视锥细胞，由于视杆细胞在数量上远大于视锥细胞，所以我们更容易感知到明暗细节。



# JPEG的压缩原理

YCbCr和RGB图像的转换方式

在RGB三种颜色中，人类对绿色及红色的敏感程度较大。蓝色的敏感程度较小。假设红色和蓝色对亮度的贡献程度分别为 $K_R$ 和 $K_B$

$$Y = K_R \cdot R + (1 - K_R - K_B) \cdot G + K_B \cdot B$$

通常， $K_R = 0.299$ ， $K_B = 0.114$

$$Y = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B$$

蓝色和红色的色差：

$$C_b = \frac{1}{2} \cdot \frac{B - Y}{1 - K_B} \quad C_r = \frac{1}{2} \cdot \frac{R - Y}{1 - K_R}$$

# JPEG的压缩原理

图像分块：jpeg格式压缩的时候，是将图像分割成 $8\times 8$ 的block，然后对各个block进行单独压缩。这是由于在较小区域内，像素和像素具有更好的相关性。



# JPEG的压缩原理

离散余弦变换(DCT)：离散傅里叶变换的一种形式。立叶级数展开式中，如果被展开的函数是实偶函数，那么其傅立叶级数中只包含余弦项。

# JPEG的压缩原理

离散余弦变换(DCT)：离散傅里叶变换的一种形式。在离散傅立叶级数展开式中，如果被展开的函数是实偶函数，那么其傅立叶级数中只包含余弦项。



35	38	55	90	65	50	72	163
40	42	68	112	77	56	66	157
66	66	90	108	74	53	87	177
84	91	83	72	57	66	126	197
90	80	76	55	65	113	173	207
60	57	64	77	107	160	198	208
65	75	88	127	158	188	202	203
102	116	137	163	186	197	198	202

低频

1136.0	-292.3	83.3	-237.2	195.2	-32.0	23.7	-52.9
1236.0	-242.4	2.5	-247.6	217.8	-20.0	35.7	-67.7
1442.0	-198.3	120.4	-286.7	182.4	-11.2	28.2	-25.1
1552.0	-255.1	332.9	-224.3	62.2	-25.3	-9.3	-6.4
1718.0	-429.2	376.0	-74.6	-35.4	21.4	17.2	15.8
1862.0	-643.2	178.9	30.5	-38.2	24.8	7.0	-1.9
2212.0	-605.1	-30.6	50.7	0.0	5.2	-14.9	-18.2
2602.0	-408.2	-98.5	9.0	7.1	-11.9	2.5	-2.6

DCT结果

高频

# JPEG的压缩原理

1136.0	-292.3	83.3	-237.2	195.2	-32.0	23.7	-52.9
1236.0	-242.4	2.5	-247.6	217.8	-20.0	35.7	-67.7
1442.0	-198.3	120.4	-286.7	182.4	-11.2	28.2	-25.1
1552.0	-255.1	332.9	-224.3	62.2	-25.3	-9.3	-6.4
1718.0	-429.2	376.0	-74.6	-35.4	21.4	17.2	15.8
1862.0	-643.2	178.9	30.5	-38.2	24.8	7.0	-1.9
2212.0	-605.1	-30.6	50.7	0.0	5.2	-14.9	-18.2
2602.0	-408.2	-98.5	9.0	7.1	-11.9	2.5	-2.6

将离散余弦变换的结果，  
从浮点数据量子化为整数

量子化结果 = $\text{round}( X_{u,v} / p_{u,v} )$

$X_{u,v}$  : DCT变换后的值

$p_{u,v}$  : 量化系数

$\text{round}$  : 取整

# JPEG的压缩原理

Jpeg预设了2个量化系数表。一个用于亮度量化，一个用于色差量化

16	11	10	16	24	50	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

亮度量化表

17	18	24	46	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

色差量化表

# JPEG的压缩原理

Jpeg预设了2个量化系数表。

16	11	10	16	24	50	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

亮度量化表

左上角量化系数较小，  
更多地保存信息。

一个用于亮度量化

17	18	24	46	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

色差量化表

右下角量化系数较大，  
更少地保存信息。

-415.38	-30.19	-61.20	27.24	56.12	-20.10	-2.39	0.46
4.47	-21.86	-60.76	10.25	13.15	-7.09	-8.54	4.88
-46.83	7.37	77.13	-24.56	-28.91	9.93	5.42	-5.56
-48.53	12.07	34.10	-14.76	-10.24	6.30	1.83	1.95
12.12	-6.55	-13.20	-3.95	-1.87	1.75	-2.79	3.14
-7.73	2.91	2.38	-5.94	-2.38	0.94	4.30	1.85
-1.03	0.18	0.42	-2.24	-0.88	-3.02	4.12	-0.66
-0.17	-0.14	-1.07	-4.19	-1.17	-0.10	0.50	1.68

DCT结果

-26	-3	-6	2	2	-1	0	0	0
0	-2	-4	1	1	0	0	0	0
-3	1	5	-1	-1	0	0	0	0
-3	1	2	-1	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

量化结果

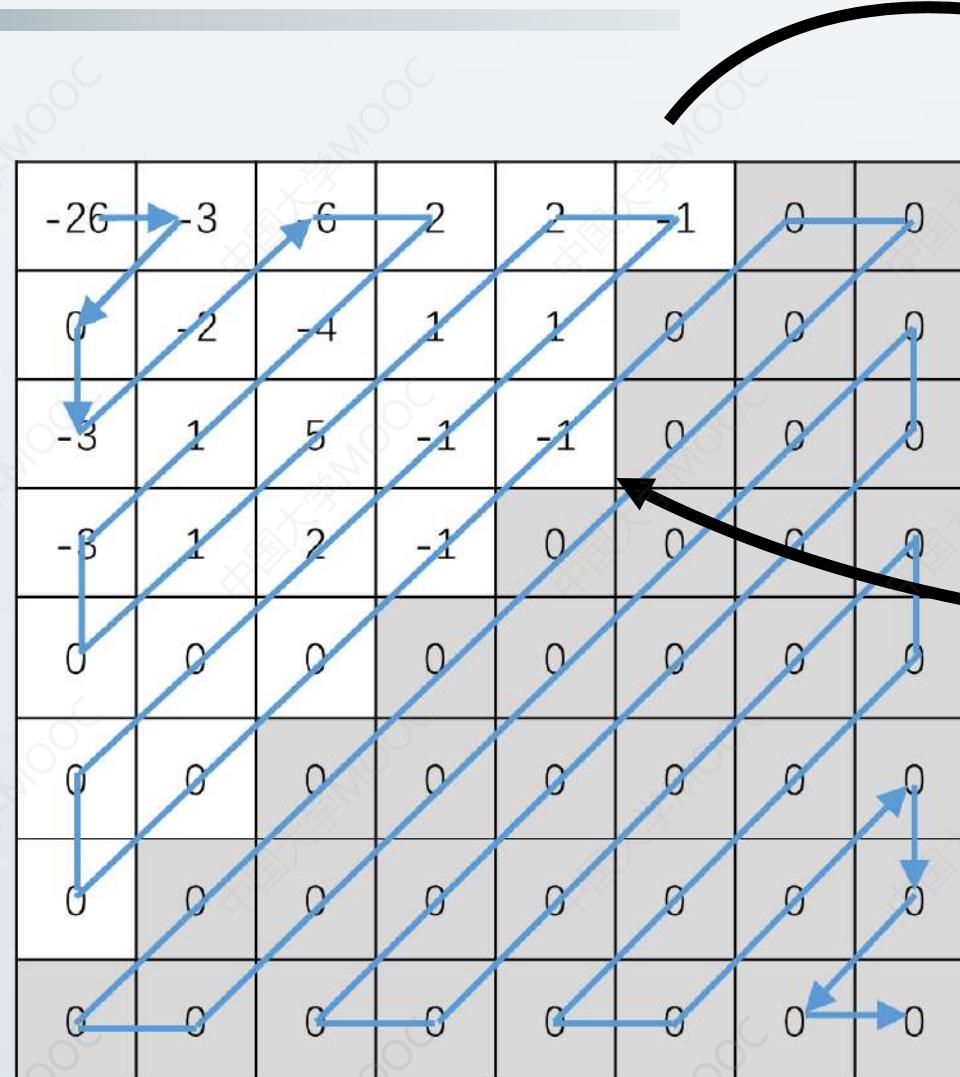
-415.38	-30.19	-61.20	27.24	56.12	-20.10	-2.29	0.46		-26	-3	-6	2	2	-1	0	0
4.47	-21.86	-60.76						4	4.88	0	-2	-4	1	1	0	0
-46.83	7.37	77.13	-24.56	-28.91	9.93	5.42	-5.56			1	5	-1	-1	0	0	0
-48.53	12.07	34.10	-14.76		16	11	10	16	24	50	51	61				
					12	12	14	19	26	58	60	55		1	2	-1
12.12	-6.55	-13.20	-3.95		14	13	16	24	40	57	69	56		0	0	0
					14	17	22	29	51	87	80	62				
-7.73	2.91	2.38	-5.94		18	22	37	56	68	109	103	77		0	0	0
					24	35	55	64	81	104	113	92				
-1.03	0.18	0.42	-2.24		49	64	78	87	103	121	120	101		0	0	0
					72	92	95	98	112	100	103	99		0	0	0

DCT结果

量化系数表

量化结果

# JPEG的压缩原理



Zigzag 扫描

一维展开

[-26, -3, 0, -3, -2, -6, 2, -4, 1, -3, 0, 1, 5, 1, 2, -1, 1, -1, 2, 0, 0, 0, 0, 0, -1, -1]

截断

使用zigzag顺序扫描的原因，0  
主要集中在右下角的高频段。以  
这样的顺序可以尽可能使0集中。

# JPEG的压缩原理



35	38	55	90	65	50	72	163
40	42	68	112	77	56	66	157
66	66	90	108	74	53	87	177
84	91	83	72	57	66	126	197
90	80	76	55	65	113	173	207
60	57	64	77	107	160	198	208
65	75	88	127	158	188	202	203
102	116	137	163	186	197	198	202

至此， $8 \times 8$ 的像素block已转换为如下的一维数组，最后一步是对该数组进行编码，以进一步压缩数据。



$[-26, -3, 0, -3, -2, -6, 2, -4, 1, -3, 0, 1, 5, 1, 2, -1, 1, -1, 2, 0, 0, 0, 0, -1, -1]$

# 游程编码

A A A A A B B B B C C C

# 游程编码

A A A A A B B B B C C C



5A 4B 3C

- 游程编码方法简单，但是只适用于有大量重复内容的信息。如AAA占用3个字节，编码为(3,A)占用两个字节，能够节省一个字节的空间。
- 可以看出连续出现的次数越多压缩效果越好，节省的空间越大。
- 如连续出现次数总是为1，这个时候会越压越大，比如A占用一个字节，编码为(1,A)占用两个字节，比原来多了一个字节。

# 信息熵

- 信息压缩的极限，可以根据香农的信息论相关理论得到。
- 在均匀分布的情况下，假定一个字符（或字符串）在文件中出现的概率是 $p$ ，那么在这个位置上最多可能出现 $1/p$ 种情况。需要 $\log_2(1/p)$ 个二进制位表示替代符号。
- 假定文件有 $n$ 个部分组成，每个部分的内容在文件中的出现概率分别为 $p_1$ 、 $p_2$ 、... $p_n$ 。那么，替代符号占据的二进制最少为：

$$\log_2(1/p_1) + \log_2(1/p_2) + \dots + \log_2(1/p_n) = \sum \log_2(1/p_i)$$

# 霍夫曼编码 (Huffman coding)

一种不等长，非前缀编码方式。1951年，MIT的哈夫曼提出。

等长式编码

A	B	C	D	E
000	001	010	011	100

D A E B C B A C B B C  
011 000 100 001 010 001 000 010 001 001 001 010

编码后数据量： $12 \times 3\text{bit} = 36\text{bit}$

# 霍夫曼编码 (Huffman coding)

## 非等长式编码

A	B	C	D	E
110	0	10	1110	1111

D A E B C B A C B B B C  
1110 110 1111 0 10 0 110 10 0 0 0 10  
A : 2      B : 5      C : 3      D : 1      E : 1

编码后数据量： $3 \times 2 + 1 \times 5 + 2 \times 3 + 4 \times 1 + 4 \times 1 = 25\text{bit}$

如何确定什么字符使用较长的编码，什么字符使用较短的编码？

# 霍夫曼编码 (Huffman coding)

非等长式编码

A	B	C	D	E
110	0	10	1110	1111



如果对调BC

D A E B C B A C B B B C  
A : 2      B : 5      C : 3      D : 1      E : 1

编码后数据量 :  $3 \times 2 + 2 \times 5 + 1 \times 3 + 4 \times 1 + 4 \times 1 = 27\text{bit}$

# 霍夫曼编码 (Huffman coding)

非等长式编码

A	B	C	D	E
110	0	10	1110	1111



如果对调BC

D A E B C B A C B B B C  
A : 2      B : 5      C : 3      D : 1      E : 1

$$\text{编码后数据量} : 3 \times 2 + 2 \times 5 + 1 \times 3 + 4 \times 1 + 4 \times 1 = 27 \text{bit}$$

出现概率越大的数据使用越短的编码，可以得到最短编码

# 霍夫曼编码 (Huffman coding)

## 非前缀编码

A	B	C	D	E
110	0	10	1110	1111

任何一个数据的编码都不与其他数据的编码的前缀重复

如果B的编码为1，则和其他编码的前缀重复。又比如，C为11，也和其他编码的前缀重复。



# 霍夫曼编码 (Huffman coding)

非前缀编码

A	B	C	D	E
110	0	10	1110	1111

已知码表，对编码后的信息进行解码，不需要知道断位信息，即可解码

1110110111101001101000010

# 霍夫曼编码 (Huffman coding)

非前缀编码

A	B	C	D	E
110	0	10	1110	1111

已知码表，对编码后的信息进行解码，不需要知道断位信息，即可解码

×

[1]110110111101001101000010

# 霍夫曼编码 (Huffman coding)

非前缀编码

A	B	C	D	E
110	0	10	1110	1111

已知码表，对编码后的信息进行解码，不需要知道断位信息，即可解码

×

[11]10110111101001101000010

# 霍夫曼编码 (Huffman coding)

非前缀编码

A	B	C	D	E
110	0	10	1110	1111

已知码表，对编码后的信息进行解码，不需要知道断位信息，即可解码

×

1110110111101001101000010

# 霍夫曼编码 (Huffman coding)

非前缀编码

A	B	C	D	E
110	0	10	1110	1111

已知码表，对编码后的信息进行解码，不需要知道断位信息，即可解码

√ 1110|110111101001101000010

D

# 霍夫曼编码 (Huffman coding)

非前缀编码

A	B	C	D	E
110	0	10	1110	1111

已知码表，对编码后的信息进行解码，不需要知道断位信息，即可解码

×

1110[1]10111101001101000010

D

# 霍夫曼编码 (Huffman coding)

非前缀编码

A	B	C	D	E
110	0	10	1110	1111

已知码表，对编码后的信息进行解码，不需要知道断位信息，即可解码

×

1110[11]0111101001101000010

D

# 霍夫曼编码 (Huffman coding)

非前缀编码

A	B	C	D	E
110	0	10	1110	1111

已知码表，对编码后的信息进行解码，不需要知道断位信息，即可解码

√ 1110<sup>---</sup><sub>110</sub>11101001101000010

D A

# 霍夫曼编码 (Huffman coding)

非前缀编码

A	B	C	D	E
110	0	10	1110	1111

已知码表，对编码后的信息进行解码，不需要知道断位信息，即可解码

×

11101101|11101001101000010

D A

# 霍夫曼编码 (Huffman coding)

非前缀编码

A	B	C	D	E
110	0	10	1110	1111

已知码表，对编码后的信息进行解码，不需要知道断位信息，即可解码

×

1110110111101001101000010

D A

# 霍夫曼编码 (Huffman coding)

非前缀编码

A	B	C	D	E
110	0	10	1110	1111

已知码表，对编码后的信息进行解码，不需要知道断位信息，即可解码

×

1110110~~111~~<sup>111</sup>101001101000010

D A

# 霍夫曼编码 (Huffman coding)

非前缀编码

A	B	C	D	E
110	0	10	1110	1111

已知码表，对编码后的信息进行解码，不需要知道断位信息，即可解码

√ 1110110111101001101000010

D A E

# 霍夫曼编码 (Huffman coding)

非前缀编码

A	B	C	D	E
110	0	10	1110	1111

已知码表，对编码后的信息进行解码，不需要知道断位信息，即可解码

1110110111101001101000010

D A E B C B A C BBB C

# 霍夫曼编码 (Huffman coding)

非前缀编码

A	B	C	D	E
110	0	10	1110	1111

已知码表，对编码后的信息进行解码，不需要知道断位信息，即可解码

1110110111101001101000010

D A E B C B A C BBB C

# 霍夫曼编码 (Huffman coding)

## 前缀编码

A	B	C	D	E
110	1	11	1110	1111

如果B的编码为1，则和其他编码的前缀重复。又比如，C为11，也和其他编码的前缀重复。

A : (1)10      B : 1

A : (11)0      C : 11

# 霍夫曼编码 (Huffman coding)

非前缀编码

A	B	C	D	E
110	1	11	1110	1111

1110110111101001101000010  
↓  
[1] [11] or [1][1][1] or [11][1]

# 霍夫曼编码 (Huffman coding)

霍夫曼编码通过二叉树获得编码结果

Input	P
$S_1$	0.4
$S_2$	0.3
$S_3$	0.1
$S_4$	0.1
$S_5$	0.06
$S_6$	0.04

# 霍夫曼编码 (Huffman coding)

霍夫曼编码通过二叉树获得编码结果

Input	P	Step1
$S_1$	0.4	0.4
$S_2$	0.3	0.3
$S_3$	0.1	0.1
$S_4$	0.1	0.1
$S_5$	0.06	0.1
$S_6$	0.04	0.1

# 霍夫曼编码 (Huffman coding)

霍夫曼编码通过二叉树获得编码结果

Input	P	Step1	Step2
$S_1$	0.4	0.4	0.4
$S_2$	0.3	0.3	0.3
$S_3$	0.1	0.1	0.2
$S_4$	0.1	0.1	0.1
$S_5$	0.06	0.1	
$S_6$	0.04		

# 霍夫曼编码 (Huffman coding)

霍夫曼编码通过二叉树获得编码结果

Input	P	Step1	Step2	Step3
$S_1$	0.4	0.4	0.4	0.4
$S_2$	0.3	0.3	0.3	0.3
$S_3$	0.1	0.1	0.2	0.3
$S_4$	0.1	0.1	0.1	
$S_5$	0.06	0.1		
$S_6$	0.04			

# 霍夫曼编码 (Huffman coding)

霍夫曼编码通过二叉树获得编码结果

Input	P	Step1	Step2	Step3	Step4
$S_1$	0.4	0.4	0.4	0.4	0.6
$S_2$	0.3	0.3	0.3	0.3	0.4
$S_3$	0.1	0.1	0.2	0.3	
$S_4$	0.1	0.1	0.1		
$S_5$	0.06	0.1			
$S_6$	0.04				

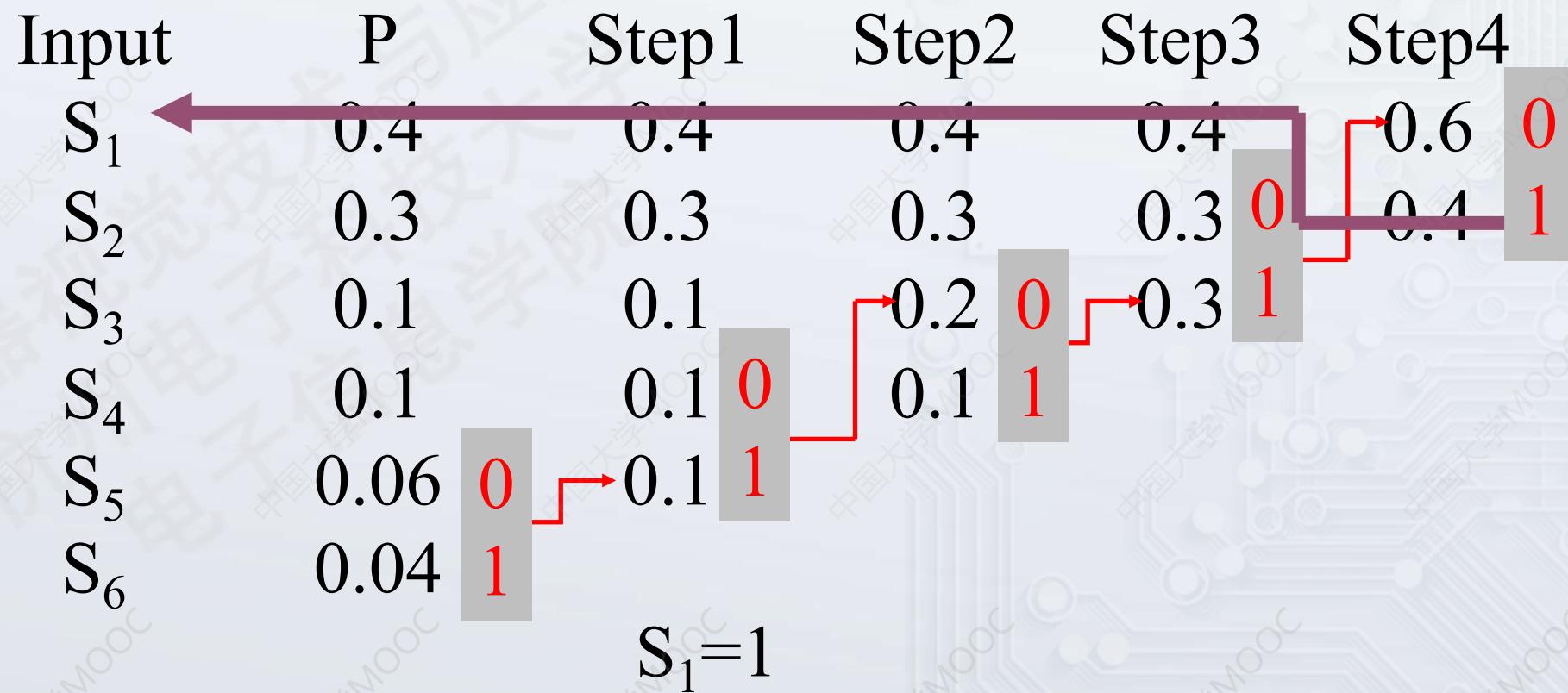
# 霍夫曼编码 (Huffman coding)

霍夫曼编码通过二叉树获得编码结果

Input	P	Step1	Step2	Step3	Step4	
$S_1$	0.4	0.4	0.4	0.4	0.6	0
$S_2$	0.3	0.3	0.3	0.3	0.4	1
$S_3$	0.1	0.1	0.2	0.3		
$S_4$	0.1	0.1	0.1	0.3		
$S_5$	0.06	0	0.1	0.1		
$S_6$	0.04	1				

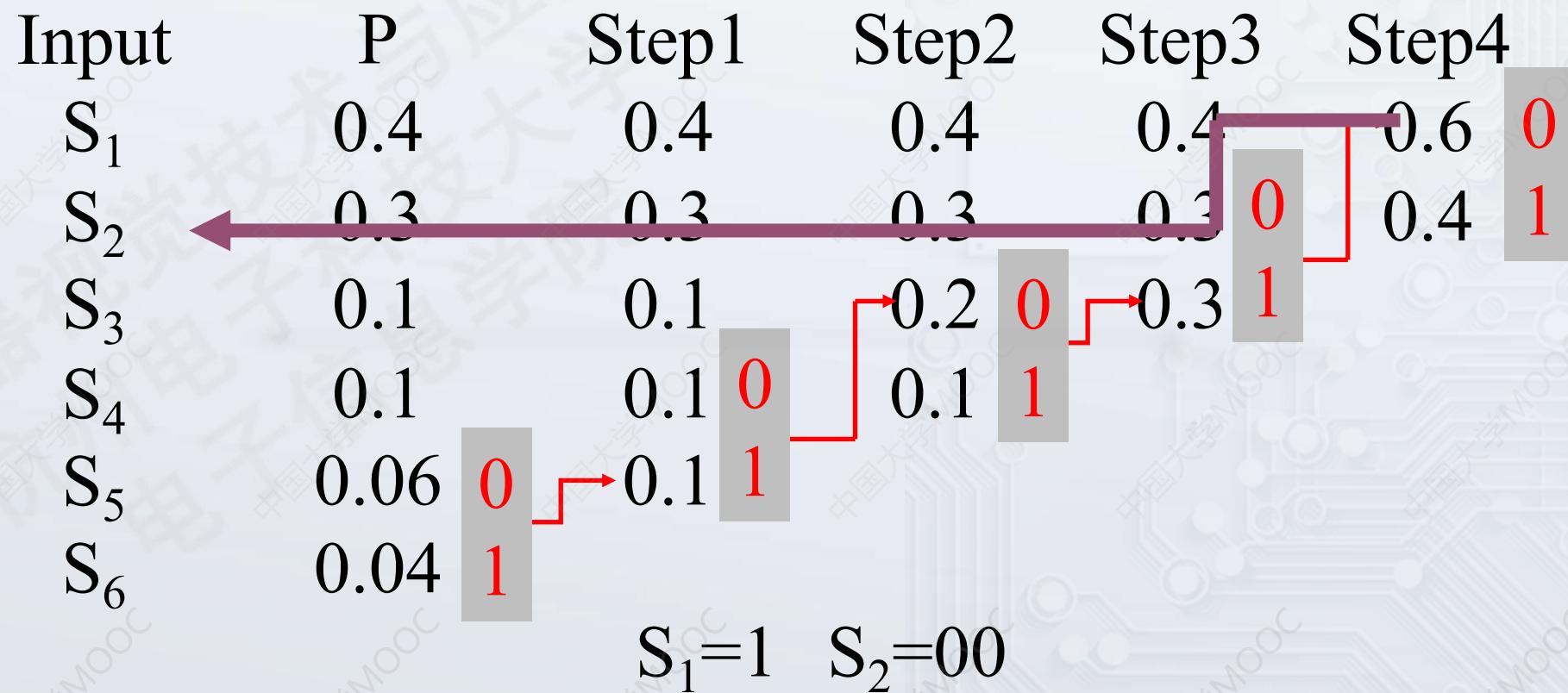
# 霍夫曼编码 (Huffman coding)

霍夫曼编码通过二叉树获得编码结果



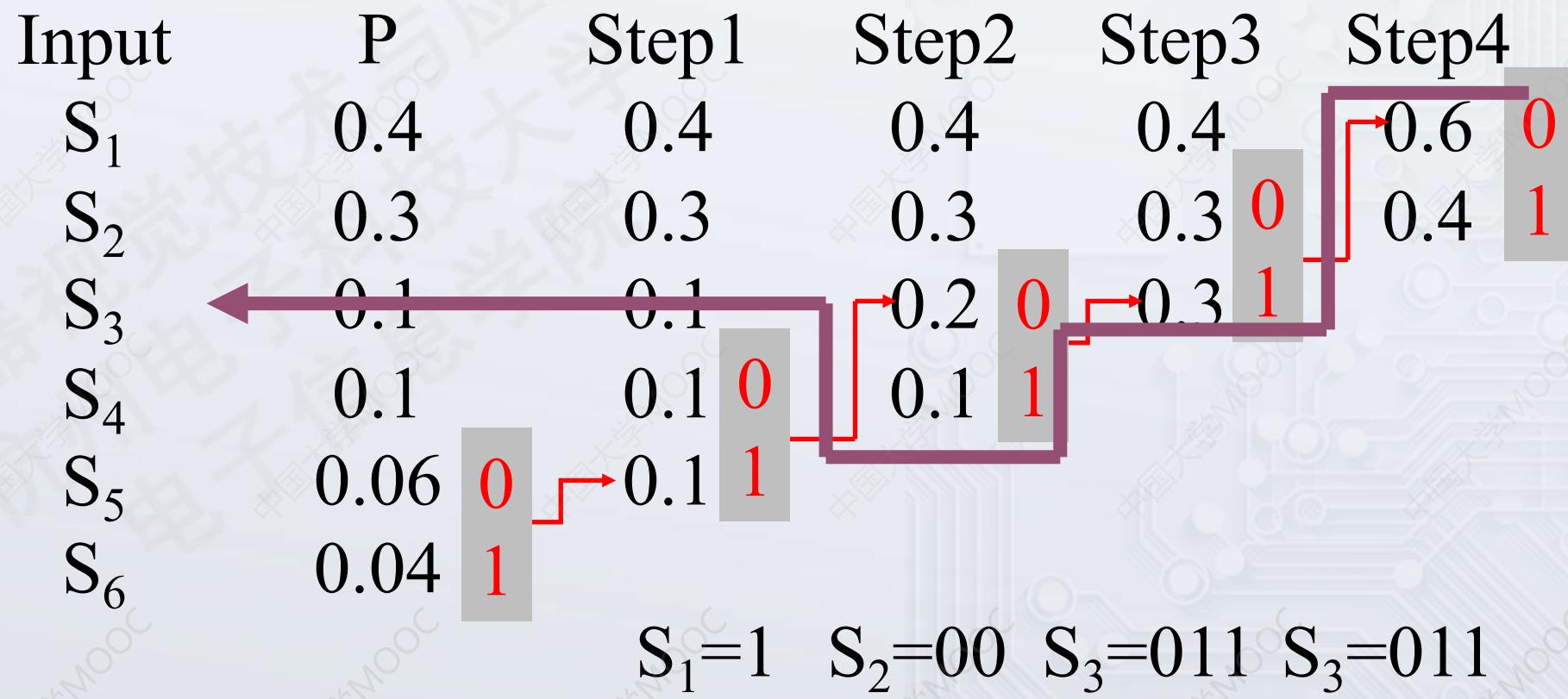
# 霍夫曼编码 (Huffman coding)

霍夫曼编码通过二叉树获得编码结果



# 霍夫曼编码 (Huffman coding)

霍夫曼编码通过二叉树获得编码结果



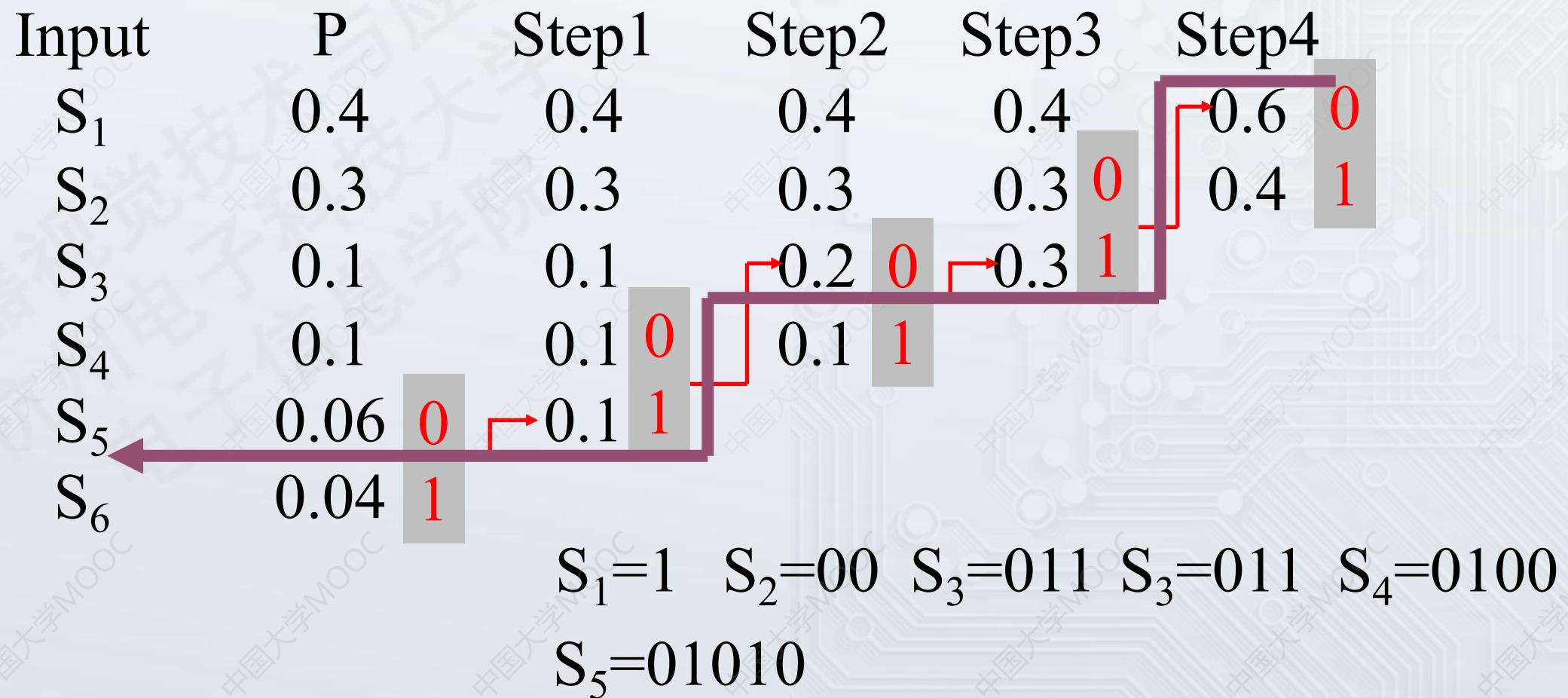
# 霍夫曼编码 (Huffman coding)

霍夫曼编码通过二叉树获得编码结果



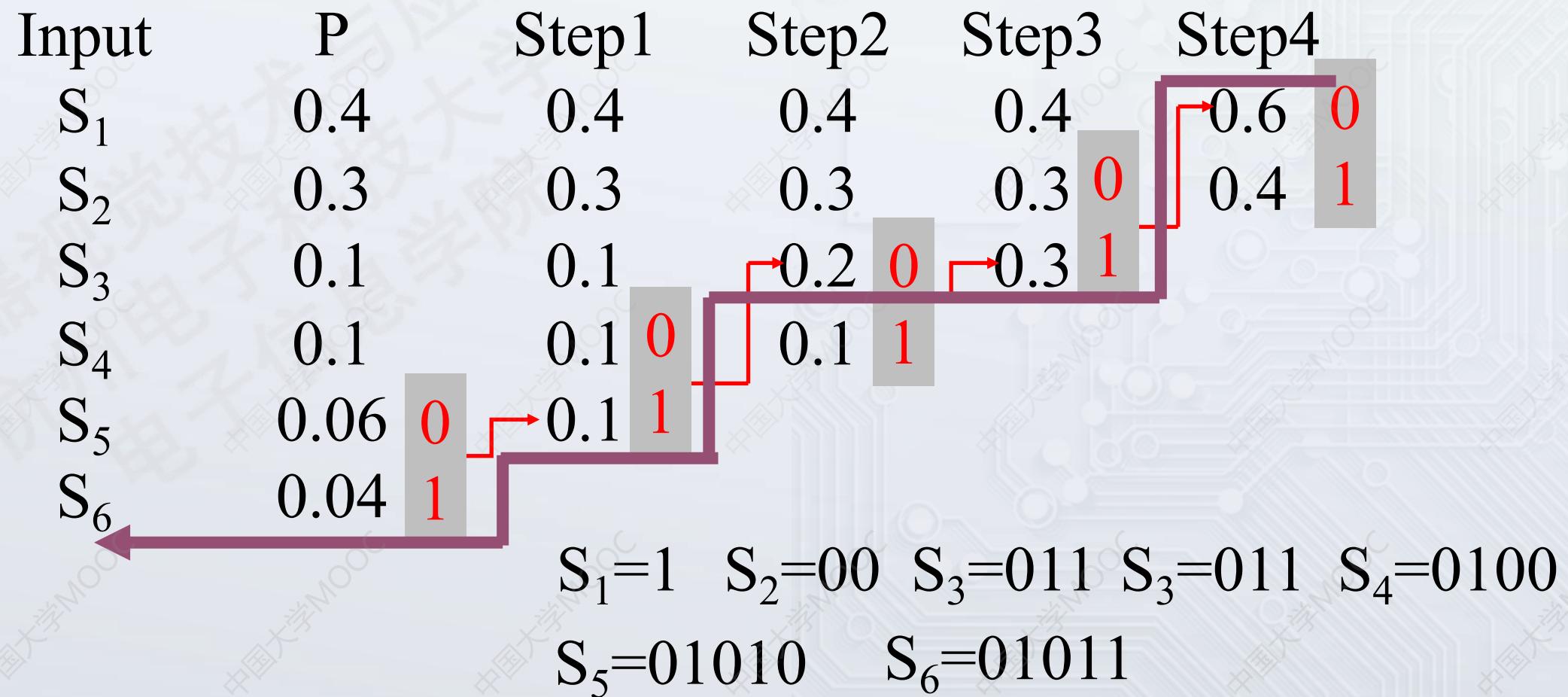
# 霍夫曼编码 (Huffman coding)

霍夫曼编码通过二叉树获得编码结果



# 霍夫曼编码 (Huffman coding)

霍夫曼编码通过二叉树获得编码结果

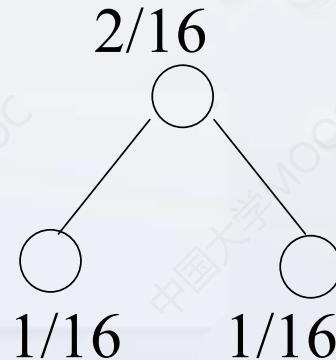


# 霍夫曼编码 (Huffman coding)

霍夫曼编码通过二叉树获得编码结果

30	10	20	40
20	40	0	20
20	20	30	30
20	40	40	20

值 : 0    10    20    30    40  
概率 :  $1/16$   $1/16$   $7/16$   $3/16$   $4/16$



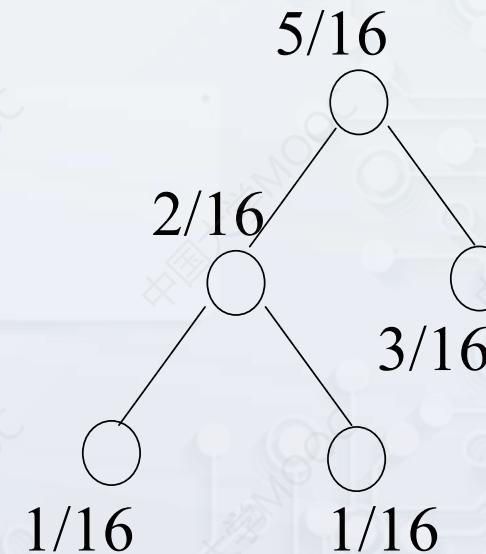
# 霍夫曼编码 (Huffman coding)

霍夫曼编码通过二叉树获得编码结果

30	10	20	40
20	40	0	20
20	20	30	30
20	40	40	20

值 : 0 10 20 30 40

概率 :  $1/16$   $1/16$   $7/16$   $3/16$   $4/16$



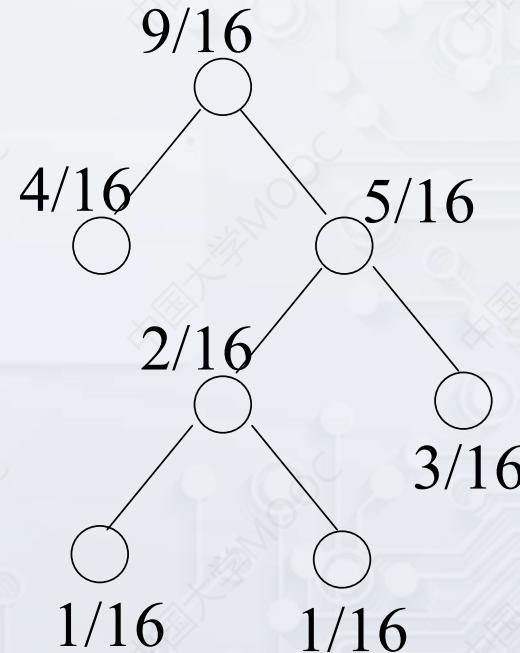
# 霍夫曼编码 (Huffman coding)

霍夫曼编码通过二叉树获得编码结果

30	10	20	40
20	40	0	20
20	20	30	30
20	40	40	20

值 : 0 10 20 30 40

概率 :  $1/16$   $1/16$   $7/16$   $3/16$   $4/16$

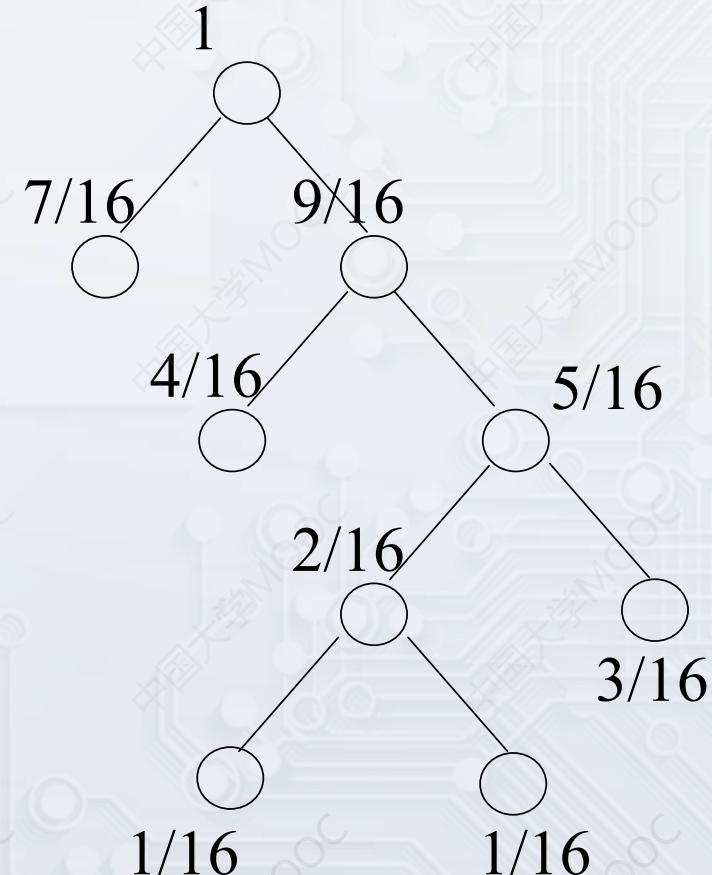


# 霍夫曼编码 (Huffman coding)

霍夫曼编码通过二叉树获得编码结果

30	10	20	40
20	40	0	20
20	20	30	30
20	40	40	20

值 : 0    10    20    30    40  
概率 :  $1/16$   $1/16$   $7/16$   $3/16$   $4/16$

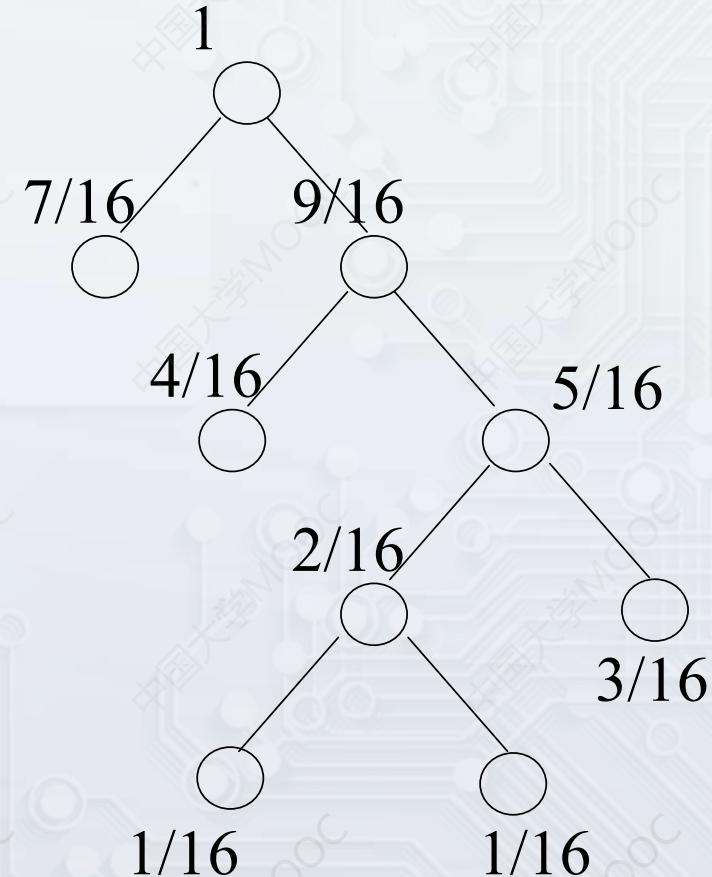


# 霍夫曼编码 (Huffman coding)

霍夫曼编码通过二叉树获得编码结果

30	10	20	40
20	40	0	20
20	20	30	30
20	40	40	20

值 : 0    10    20    30    40  
概率 :  $1/16$   $1/16$   $7/16$   $3/16$   $4/16$

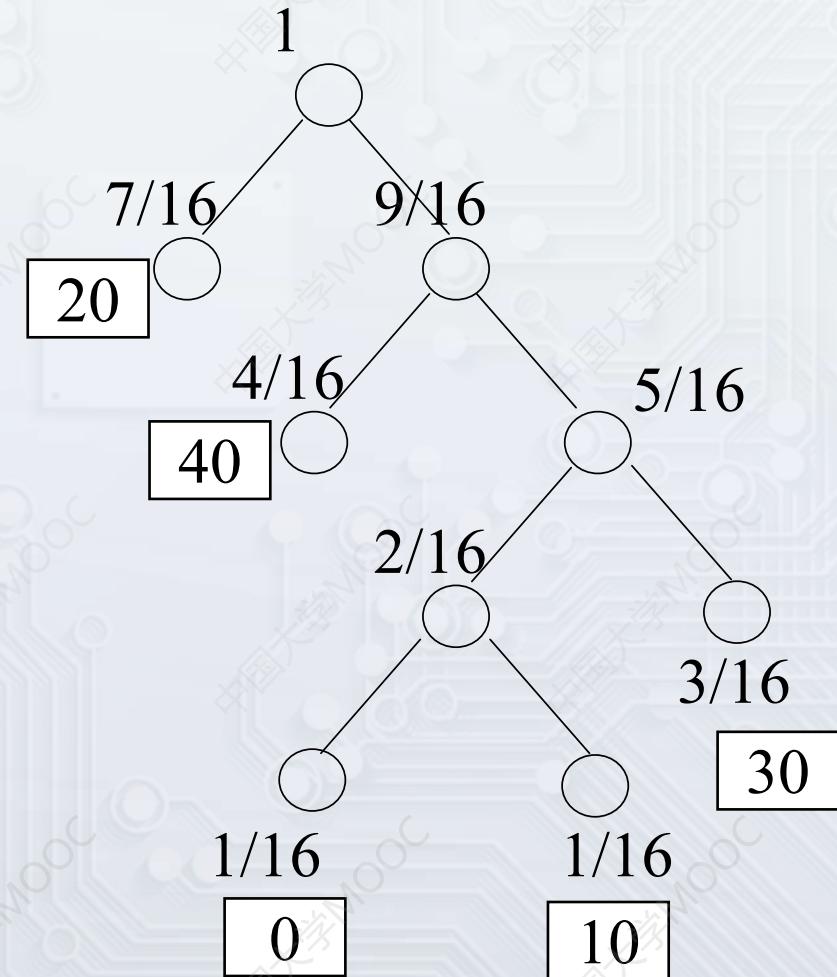


# 霍夫曼编码 (Huffman coding)

霍夫曼编码通过二叉树获得编码结果

30	10	20	40
20	40	0	20
20	20	30	30
20	40	40	20

值 : 0    10    20    30    40  
概率 :  $1/16$   $1/16$   $7/16$   $3/16$   $4/16$

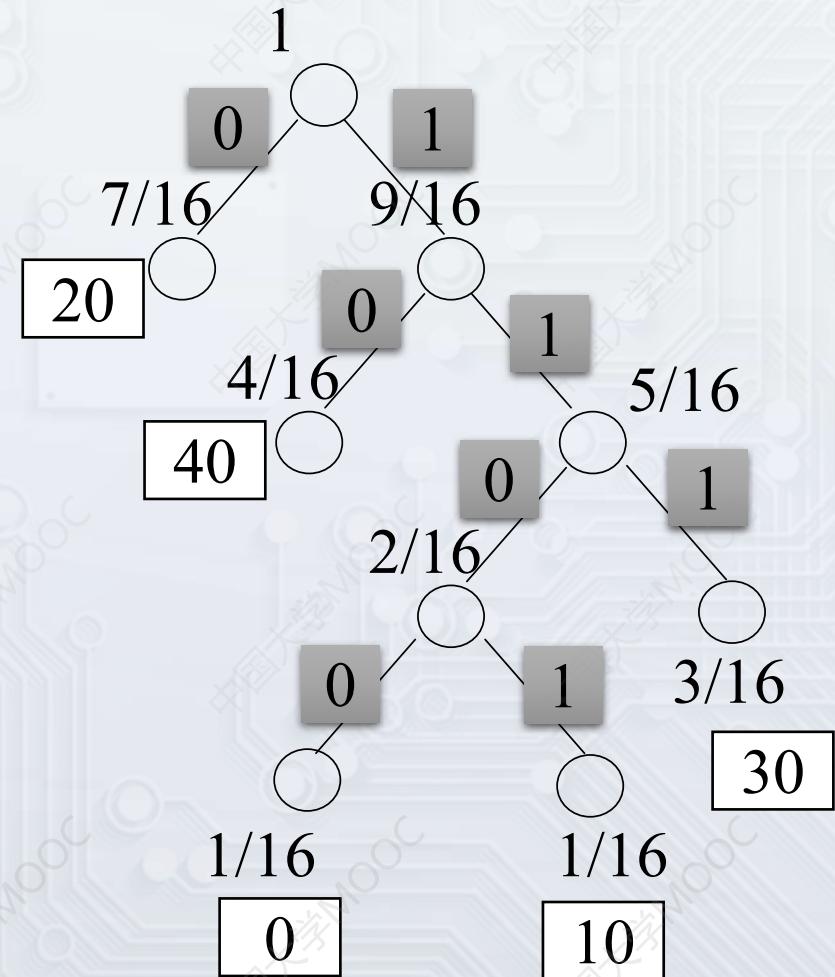


# 霍夫曼编码 (Huffman coding)

霍夫曼编码通过二叉树获得编码结果

30	10	20	40
20	40	0	20
20	20	30	30
20	40	40	20

值 : 0    10    20    30    40  
概率 :  $1/16$   $1/16$   $7/16$   $3/16$   $4/16$

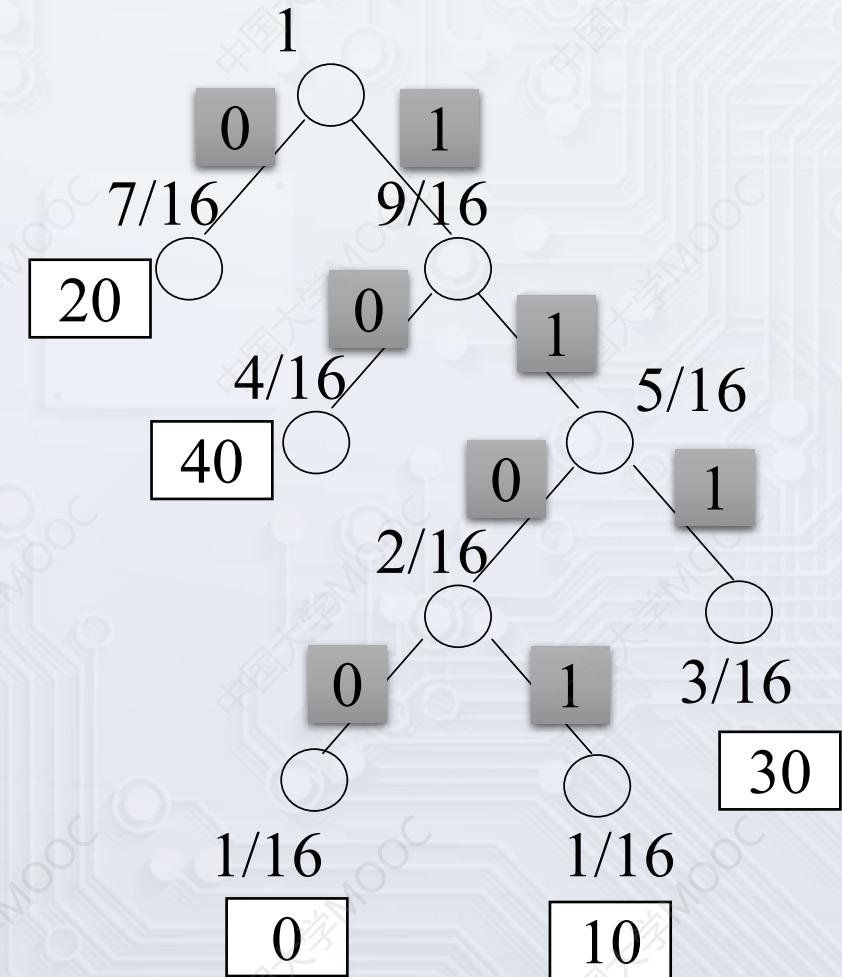


# 霍夫曼编码 (Huffman coding)

霍夫曼编码通过二叉树获得编码结果

30	10	20	40
20	40	0	20
20	20	30	30
20	40	40	20

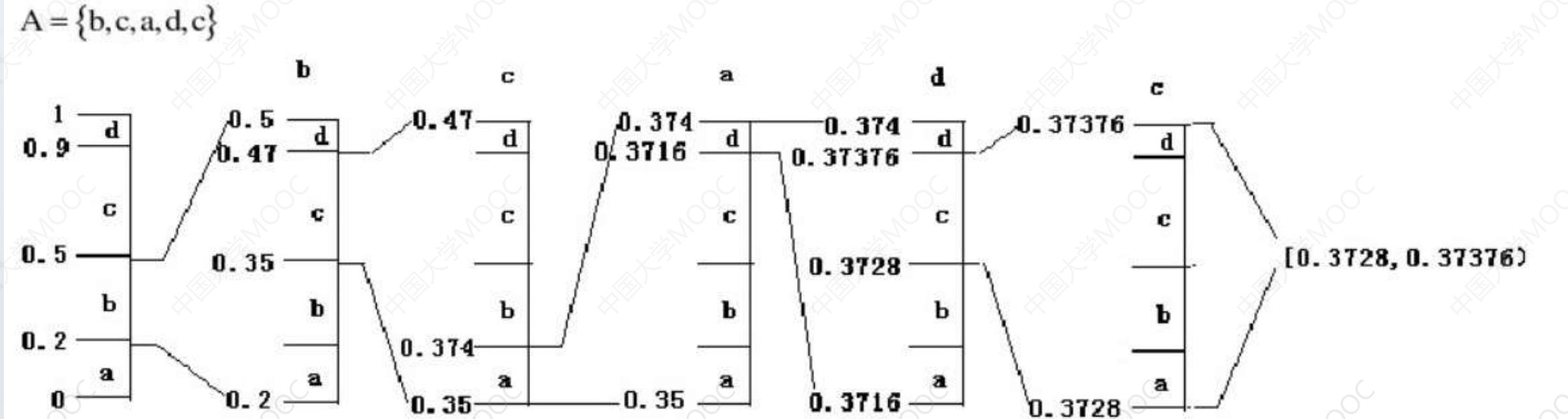
值 : 0    10    20    30    40  
概率 :  $1/16$      $1/16$      $7/16$      $3/16$      $4/16$   
编码 : 1100    101    0    111    10



# 数学编码

$$A = \{b, c, a, d, c\}$$

$$p(a) = 0.2, p(b) = 0.3, p(c) = 0.4, p(d) = 0.1$$



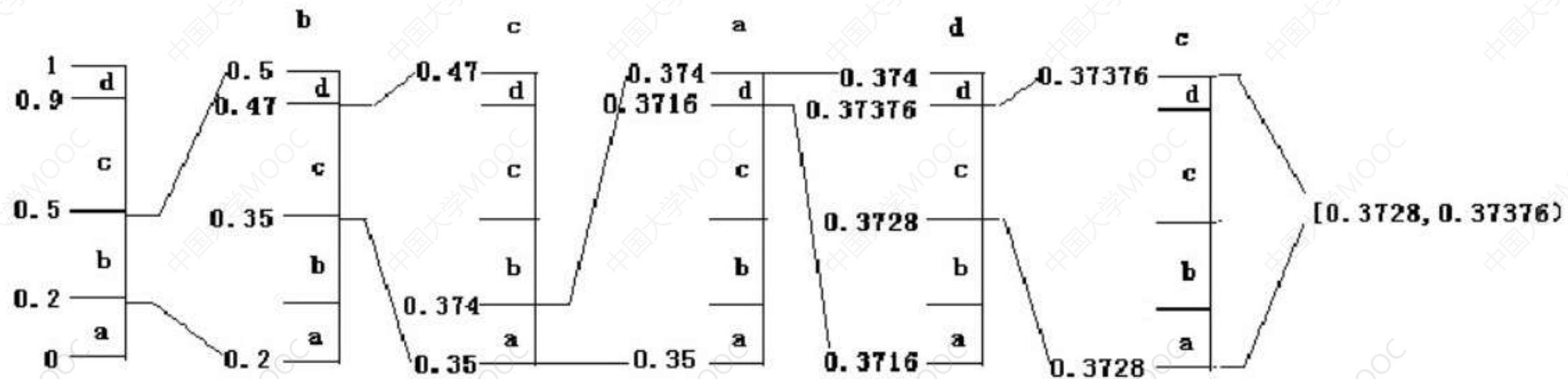
# 数学编码

[0. 3728, 0. 37376)

[0. 0101111011, 0. 0101111101)

0. 010111111

$$A = \{b, c, a, d, c\}$$



# 谢谢！

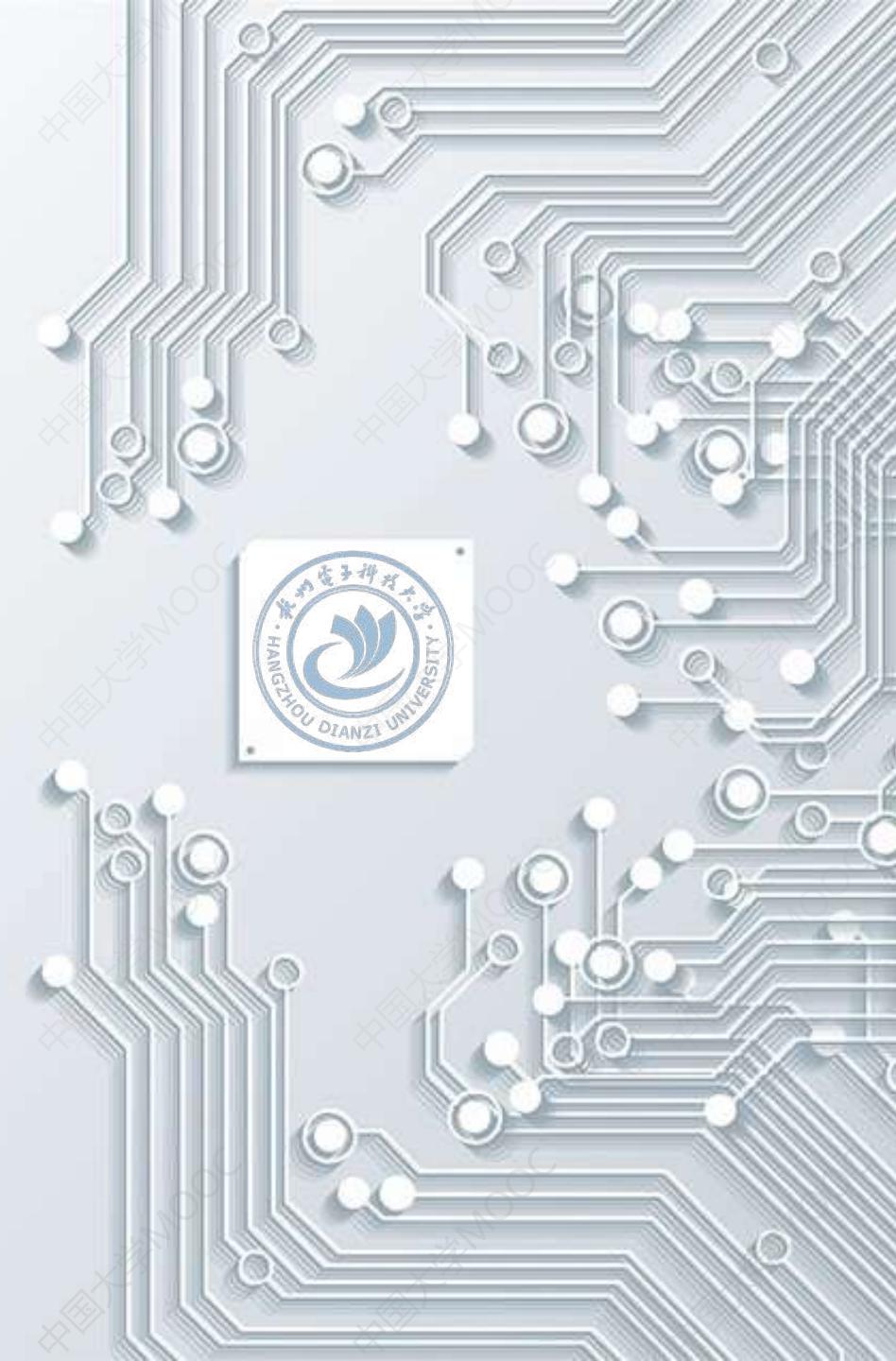
# 机器视觉技术与应用

## 11. 直方图增强

李竹

杭州电子科技大学

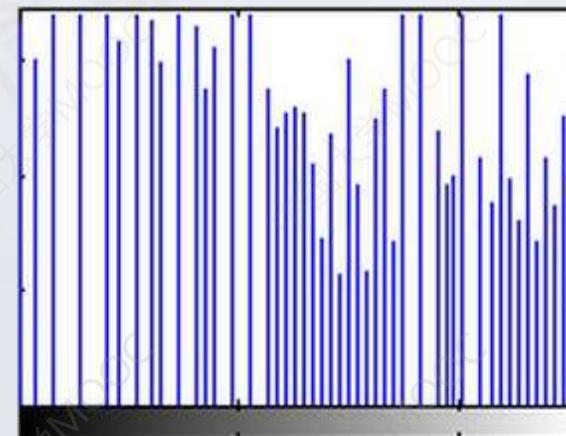
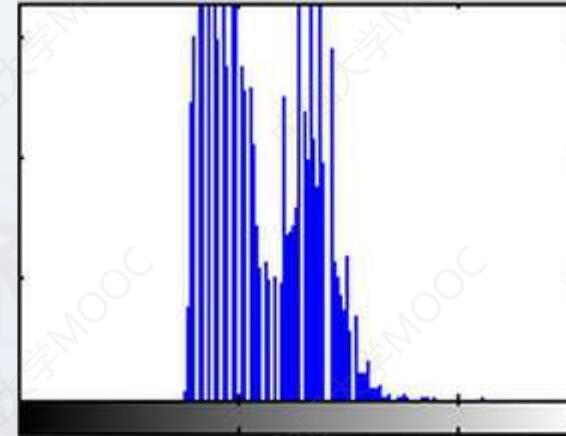
电子信息学院



# 本章概要

1. 直方图线性变换
2. Gamma矫正
3. 直方图均衡

# 直方图的线性变换

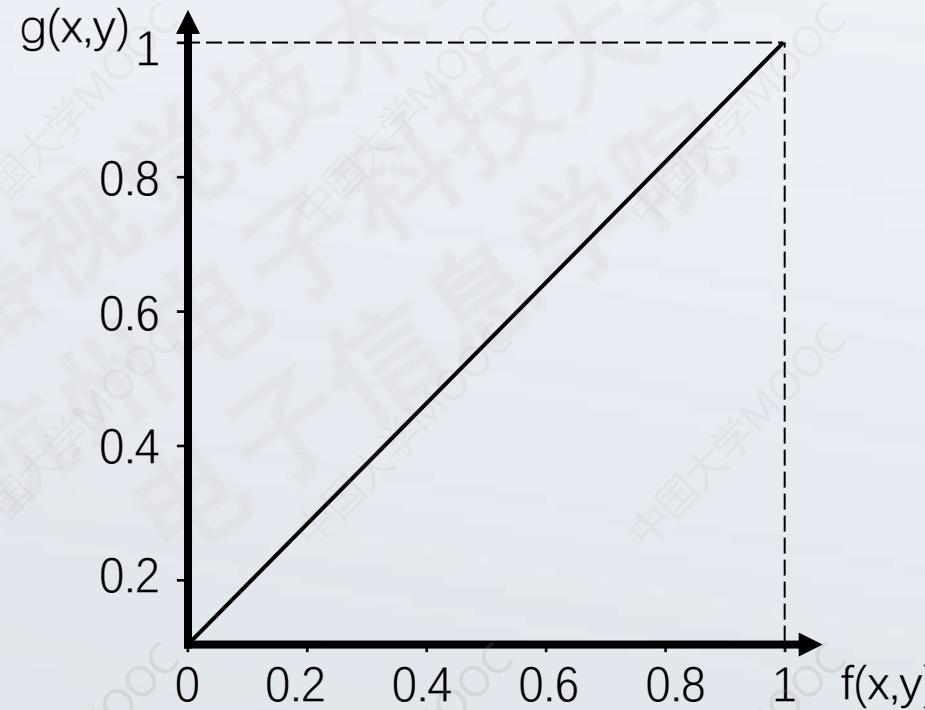


动态范围(Dynamic Range)是物理学中常见的概念，表示某一物理量最大值与最小值的比率，通常以对数表示，单位为dB。

对于图像，它指场景中最明亮处与最黑暗处的亮度之比。

# 直方图的线性变换

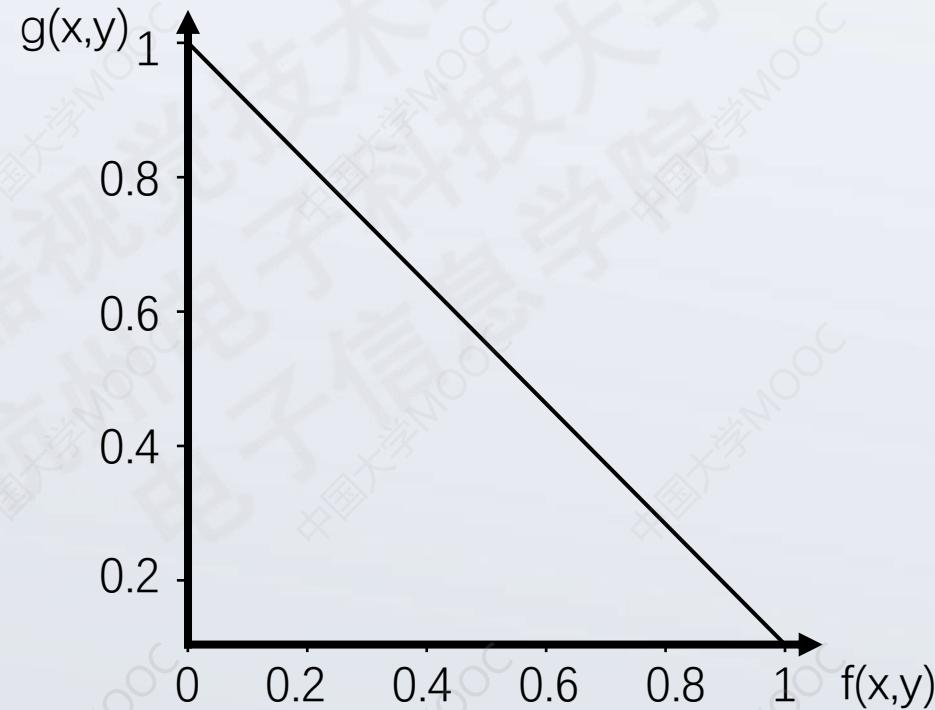
假设 $[x,y]$ 位置的像素值为 $f(x,y)$ ，作为线性变换的输入。线性变换的输出为 $g(x,y)$ 。  
则输入和输出的关系可用如下曲线表示。即  $g(x,y) = f(x,y)$ 。灰度归一化至 $[0,1]$ 。



当输入和输出的关系曲线为一条 $45^{\circ}$ 的直线时，则表示输入输出完全相同。

# 直方图的线性变换

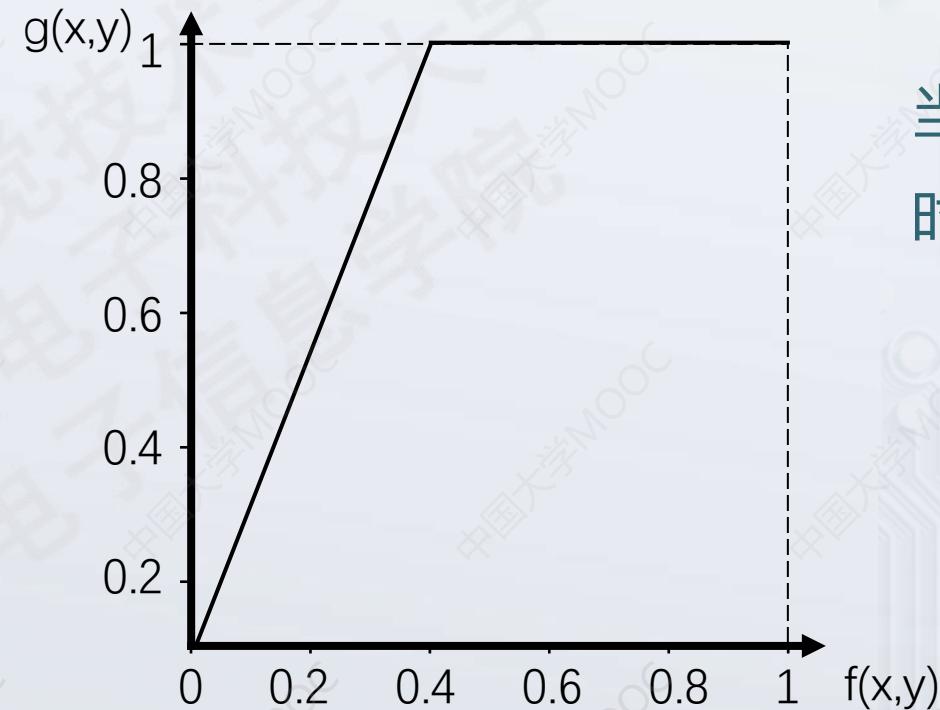
假设 $[x,y]$ 位置的像素值为 $f(x,y)$ ，作为线性变换的输入。线性变换的输出为 $g(x,y)$ 。  
则输入和输出的关系可用如下曲线表示。即  $g(x,y) = f(x,y)$ 。灰度归一化至 $[0,1]$ 。



当输入和输出的关系曲线为一条 $-45^{\circ}$ 的直线时，则表示图像反色。

# 直方图的线性变换

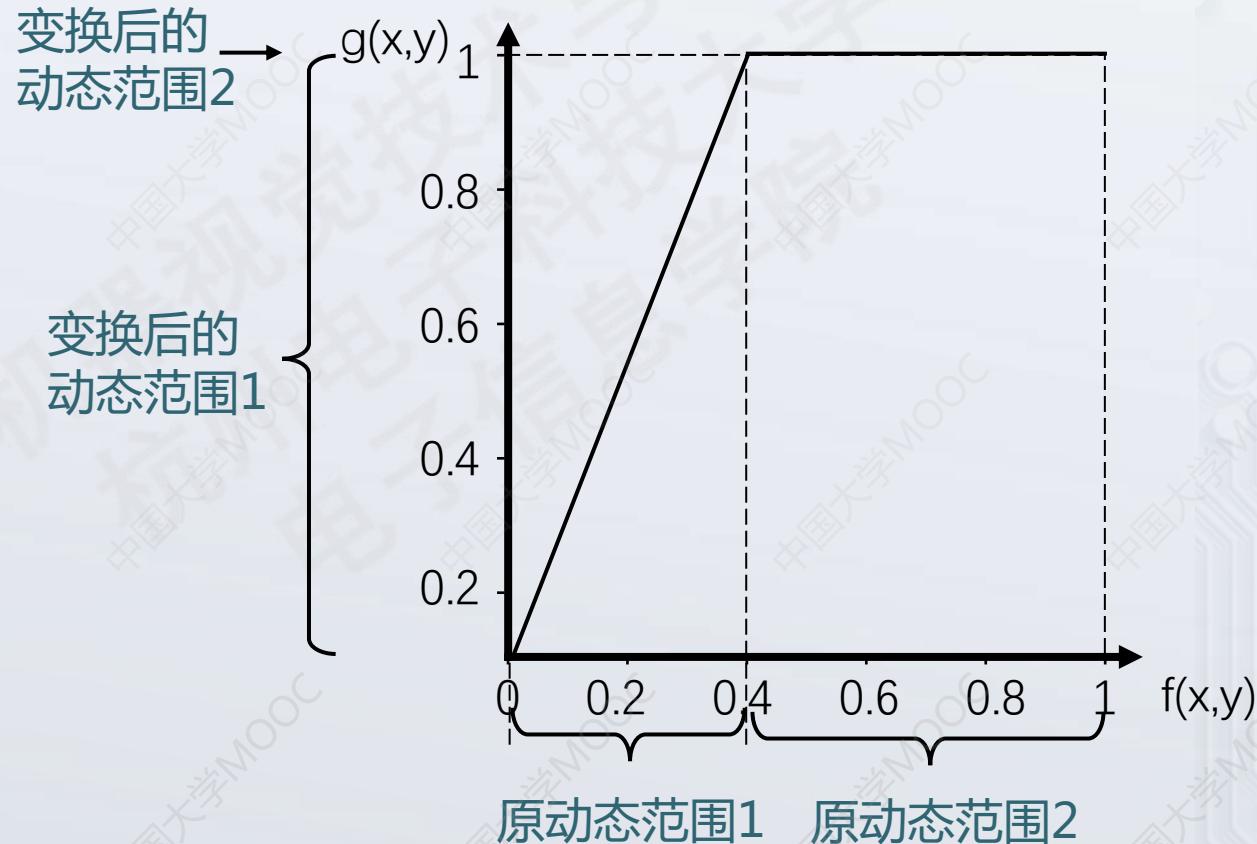
假设 $[x,y]$ 位置的像素值为 $f(x,y)$ ，作为线性变换的输入。线性变换的输出为 $g(x,y)$ 。  
则输入和输出的关系可用如下曲线表示。即  $g(x,y) = f(x,y)$ 。灰度归一化至 $[0,1]$ 。



当输入和输出的关系曲线为左图  
时，则表示输入和输出的关系？

# 直方图的线性变换

假设 $[x,y]$ 位置的像素值为 $f(x,y)$ ，作为线性变换的输入。线性变换的输出为 $g(x,y)$ 。  
则输入和输出的关系可用如下曲线表示。即  $g(x,y) = f(x,y)$ 。灰度归一化至 $[0,1]$ 。

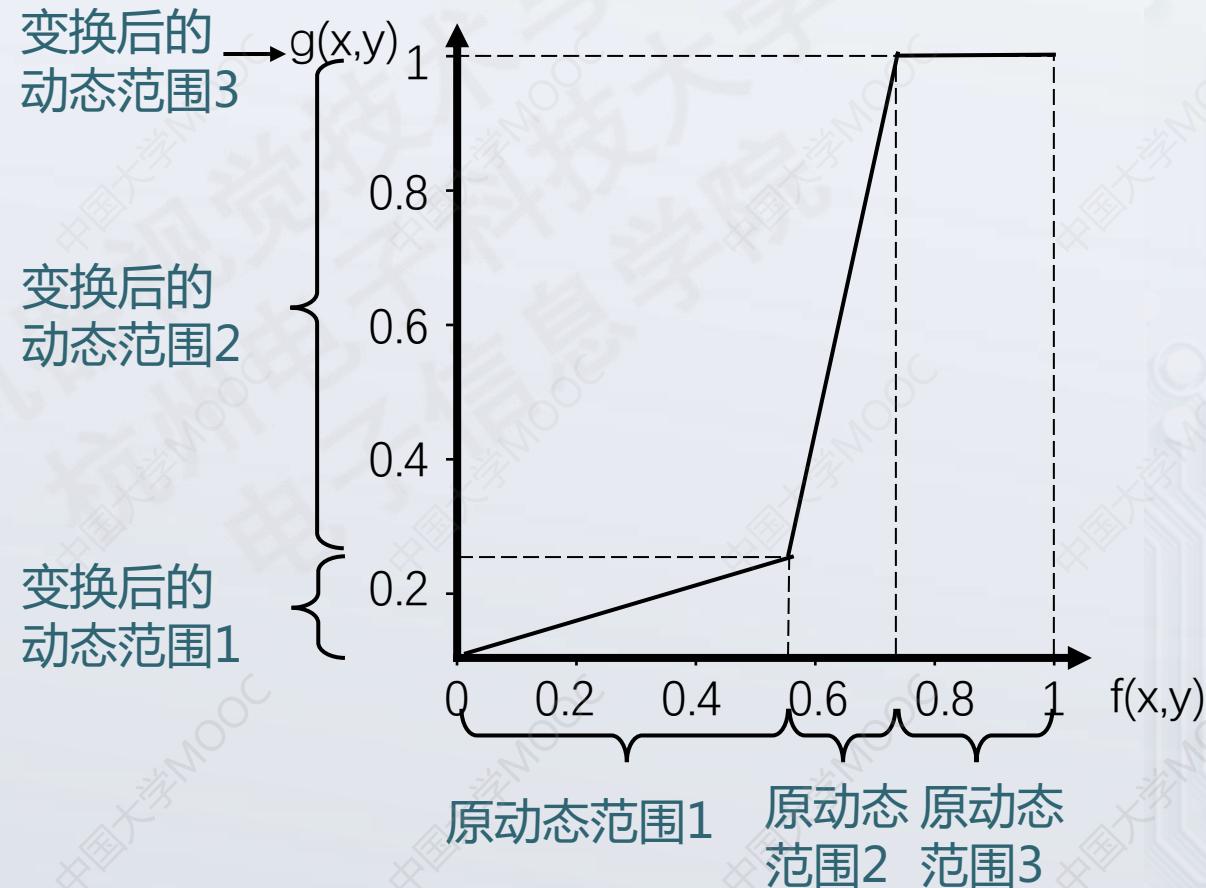


当输入和输出的关系曲线为左图时，则表示输入和输出的关系？

动态范围1被扩展。  
动态范围2被压缩。

# 直方图的线性变换

假设 $[x,y]$ 位置的像素值为 $f(x,y)$ ，作为线性变换的输入。线性变换的输出为 $g(x,y)$ 。  
则输入和输出的关系可用如下曲线表示。即  $g(x,y) = f(x,y)$ 。灰度归一化至 $[0,1]$ 。

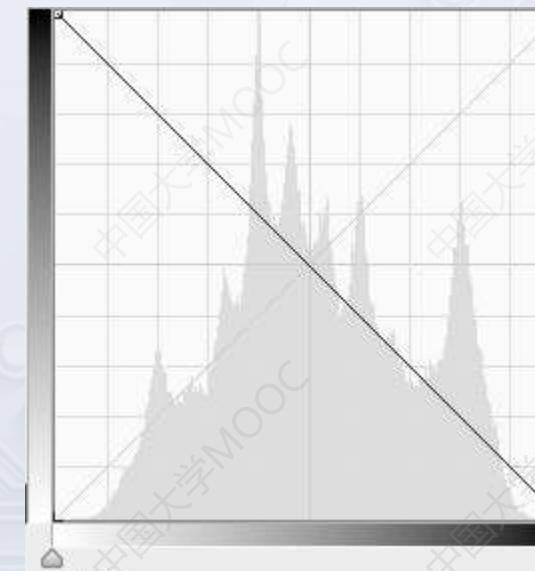
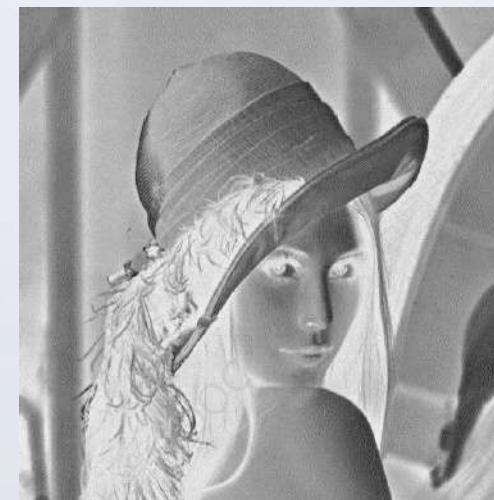
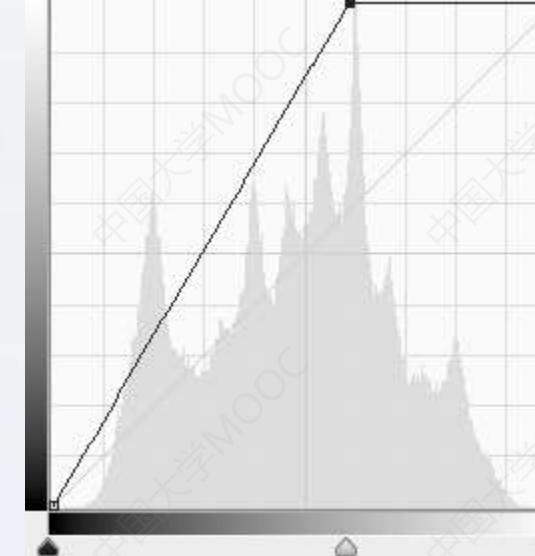


结论：

当直线大于 $45^\circ$ 时，动态范围被扩展，即细节被增强。

当直线小于 $45^\circ$ 时，动态范围被压缩，即细节被压缩。

# 直方图的线性变换



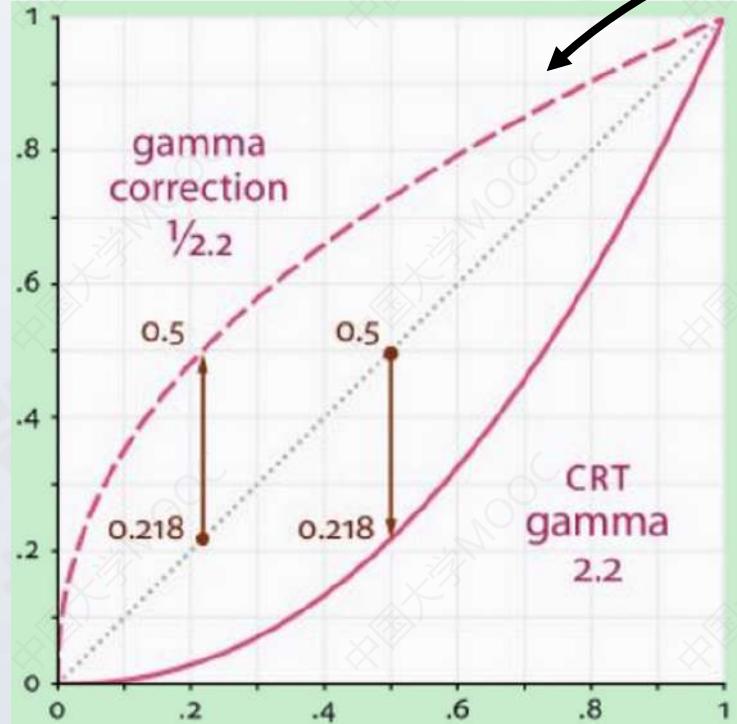
# Gamma矫正

最早用于显示器的校验，由于人眼的亮度感知到的亮度的变化并非线性均匀分布的，为了获得正确的颜色感受，需要进行非线性矫正。其像素亮度转换公式如下：

$$f(I) = I^\gamma$$

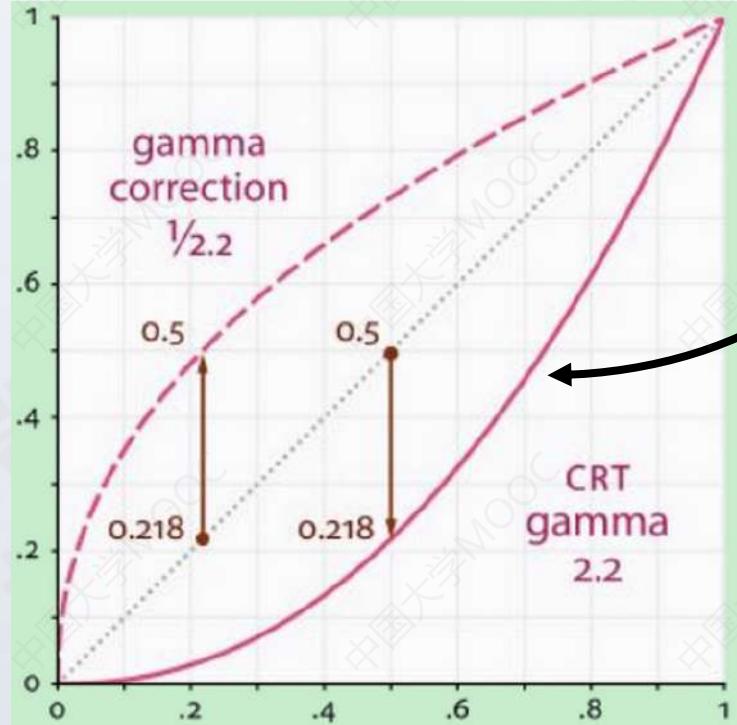
其中， $f$ 为转换函数， $I$ 为输入像素的亮度。

# Gamma矫正



1.  $\gamma < 1$ : 如虚线所示，灰度较低的区域内，动态范围变大。即较暗的区域，细节被增强。如， $\gamma$  设为  $1/2.2$  是，动态范围  $[0,0.2]$  被扩大到了  $[0,0.5]$ ，灰度较高的的范围，动态范围变小，即细节丢失。

# Gamma矫正



2.  $\gamma > 1$ : 如实线所示，  
灰度较低的区域内，动态范围变小。即较  
暗的区域，细节被减弱。  
灰度较高的的范围，动态范围变大，即细  
节增强。

# Gamma矫正



原图

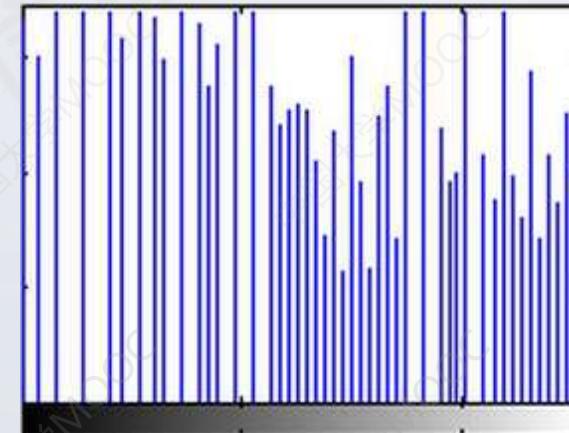
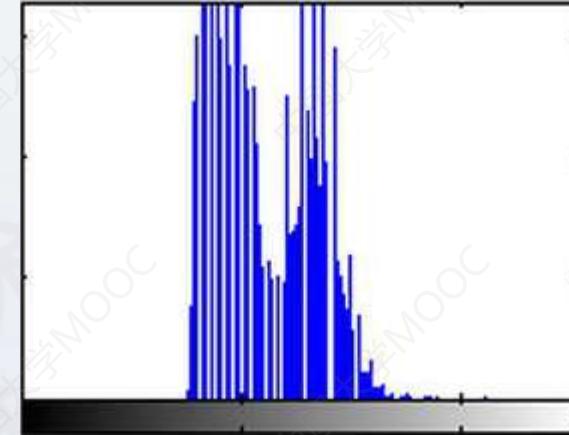


Gamma=1/2.2



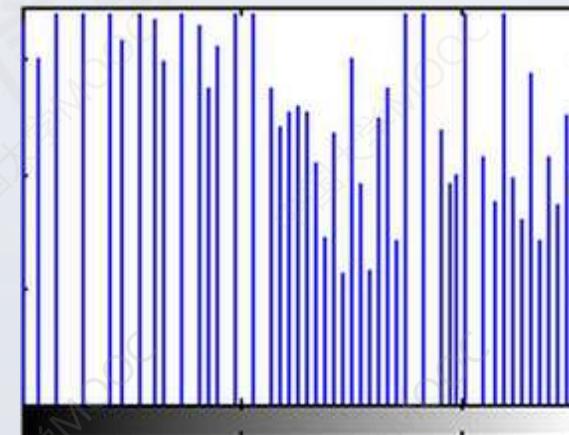
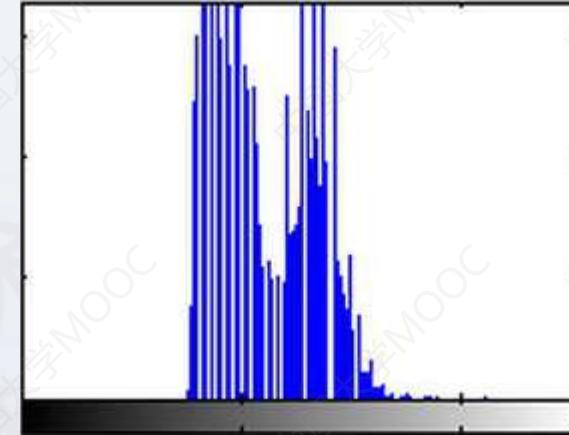
Gamma=2.2

# 直方图均衡



通过映射函数，对原图进行变化，使原本集中的分布变为均匀分布。

# 直方图均衡



映射必须满足的条件：

①保证原来的大小关系不变，较亮的区域，依旧是较亮的，较暗依旧暗，只是对比度增大，

②如果是八位图像，那么像素映射函数的值域应在0和255之间的，不能越界。

实际使用累积函数作为映射函数。

# 直方图均衡

假设输入图像为 $f(x,y)$ ，输出图像为 $g(x,y)$ ， $k$ 为灰度值， $k=0,1,2,\dots,L-1$ 。8位图像中 $L$ 为256。灰度级 $k$ 在输入输出图像中的概率分别为 $p_f(k)$ 和 $p_g(k)$ ，

输入图像中灰度级0到n的概率  $S_f(n) = \int_0^n p_f(k)dk$

输出图像中灰度级0到n的概率  $S_g(n) = \int_0^n p_g(k)dk$

如果满足条件1，对某灰度级r则必有  $S_f(r) = S_g(s)$

# 直方图均衡

$$S_f(r) = S_g(s)$$

微分得

$$p_f(r)dr = p_g(s)ds$$

假设变换函数为  $T(r) = L \times S_f(r)$ ，则：

$$s = T(r) = L \cdot S_f(r) = L \cdot \int_0^r p_f(k)dk$$

$$\frac{ds}{dr} = L \cdot p_f(r)$$

则：  $p_g(s) = \frac{1}{L-1}$

即概率平均分布

# 直方图均衡

由上可得，映射函数为

$$s = T(r) = L \cdot S_f(r) = L \cdot \int_0^r p_f(k) dk$$

离散形式为：

$$s = T(r) = L \cdot S_f(r) = L \cdot \sum_0^r \frac{n_r}{n}$$

# 直方图均衡

由上可得，映射函数为

$$s = T(r) = L \cdot S_f(r) = L \cdot \int_0^r p_f(k) dk$$

离散形式为：

$$s = T(r) = L \cdot S_f(r) = L \cdot \sum_0^r \frac{n_r}{n}$$

# 直方图均衡

255	128	200	50
50	200	255	50
255	200	128	128
200	200	255	50



255	112	191	64
64	191	255	64
255	191	112	112
191	191	255	64

gray value	Number of pixels	Probability	Cumulative probability	Gray value after mapping	Rounding
50	4	0.25	0.25	$0.25 * (255 - 0) = 63.75$	64
128	3	0.1875	0.4375	111.5625	112
200	5	0.3125	0.75	191.25	191
255	4	0.25	1	255	255

# 谢谢！

# 机器视觉技术与应用

## 11. 背景差分法

李竹

杭州电子科技大学

电子信息学院



# 本章概要

1. 背景差分法
2. 高斯背景建模

# 移动物体检测

下图为固定监控录像所拍摄到的画面，当有移动物体(如行人)出现在视野内时，使用什么方法可以检测到。

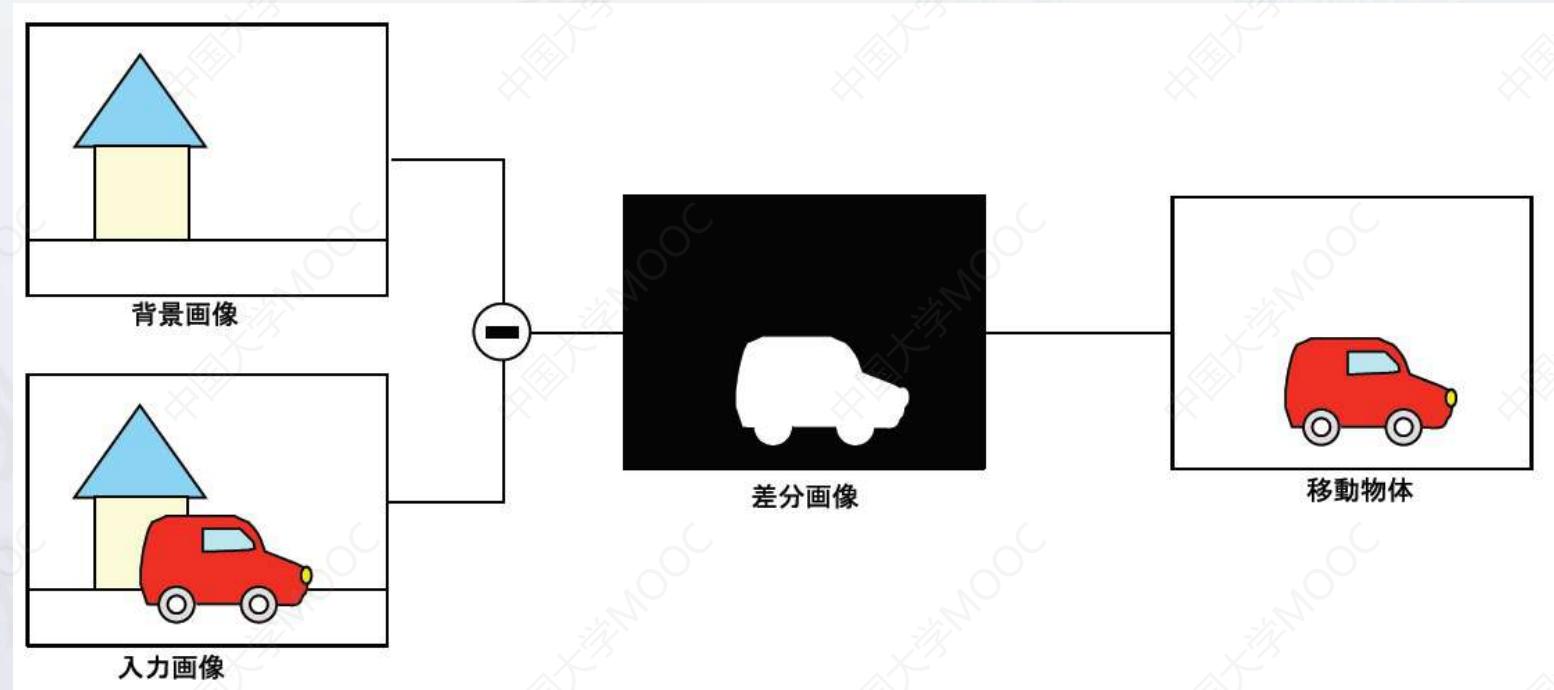
监控区域



一个行人进入画面



# 背景差分法



[1] C. Stauffer and W. Grimson, “Adaptive background mixture models for real-time tracking,” IEEE International Conference on Computer Vision and Pattern Recognition, 1999.

# 背景差分法

建立一个背景图像，计算当前输入图像和背景图像的差分，数学表达如下

$$\Delta_t = |I_t - B_t|$$

$I_t$  : frame  $t$     $B_t$  : image of background

设置一个阈值，判断像素属于背景还是移动物体

$$M_t(x, y) = \begin{cases} 1 & , \Delta t(x, y) > Th \\ 0 & , \Delta t(x, y) \leq Th \end{cases}$$

# 背景差分法

建立一个背景图像，计算当前输入图像和背景图像的差分，数学表达如下

$$\Delta_t = |I_t - B_t|$$

$I_t$  : frame  $t$     $B_t$  : image of background

设置一个阈值，判断像素属于背景还是移动物体

$$M_t(x, y) = \begin{cases} 1 & , \Delta t(x, y) > Th \\ 0 & , \Delta t(x, y) \leq Th \end{cases}$$

# 背景差分法



## 一些问题

— 噪声

— 阳光的闪烁

— 摆动的树枝，旗帜

— 阴影

— 环境光照的变换(如白天到夜晚)

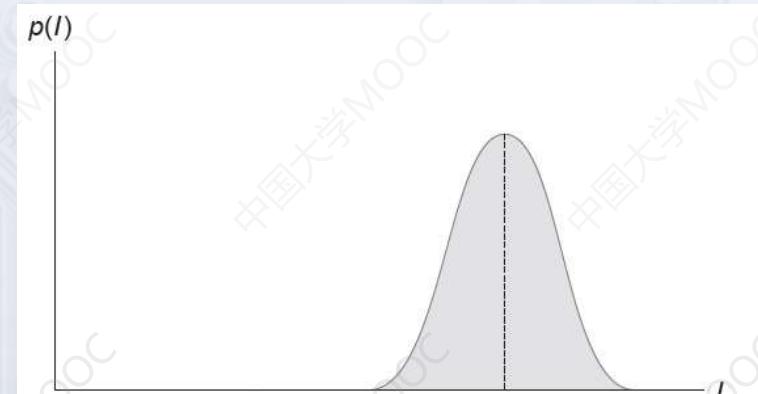
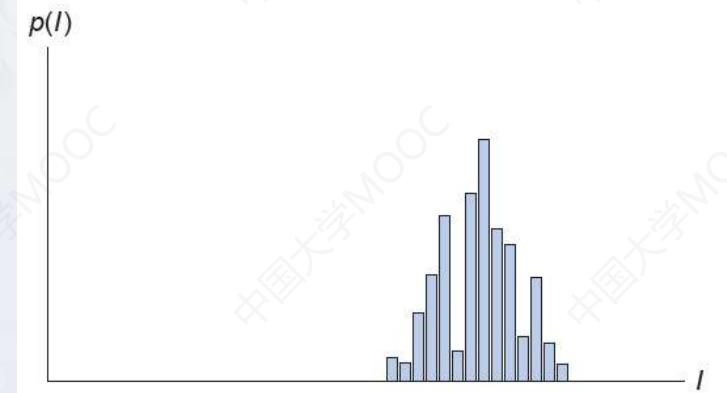
# 高斯背景建模



如果我们统计某个像素在某段时间内的灰度概率。

## 一些问题

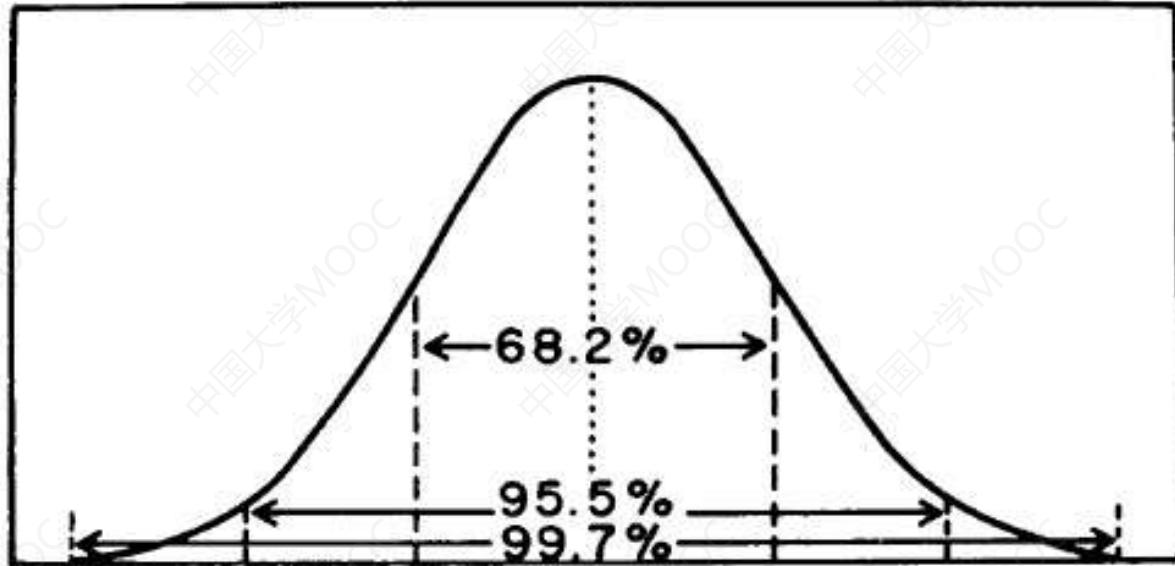
- 噪声
- 阳光的闪烁



# 高斯背景建模

## 一些问题

- 噪声
- 阳光的闪烁



- 计算每个像素的  $L$  和  $\sigma$ 。
- 根据  $L$  和  $\sigma$  为每个像素设置不同的阈值

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-L)^2}{2\sigma^2}}$$

$L$ : mean

$\sigma$ : variance

$$M_t(x, y) = \begin{cases} 1 & , \Delta t(x, y) > Th \\ 0 & , \Delta t(x, y) \leq Th \end{cases}$$



$$M_t(x, y) = \begin{cases} 1 & , |I_t(x, y) - L(x, y)| > Th \times \sigma \\ 0 & , |I_t(x, y) - L(x, y)| \leq Th \times \sigma \end{cases}$$

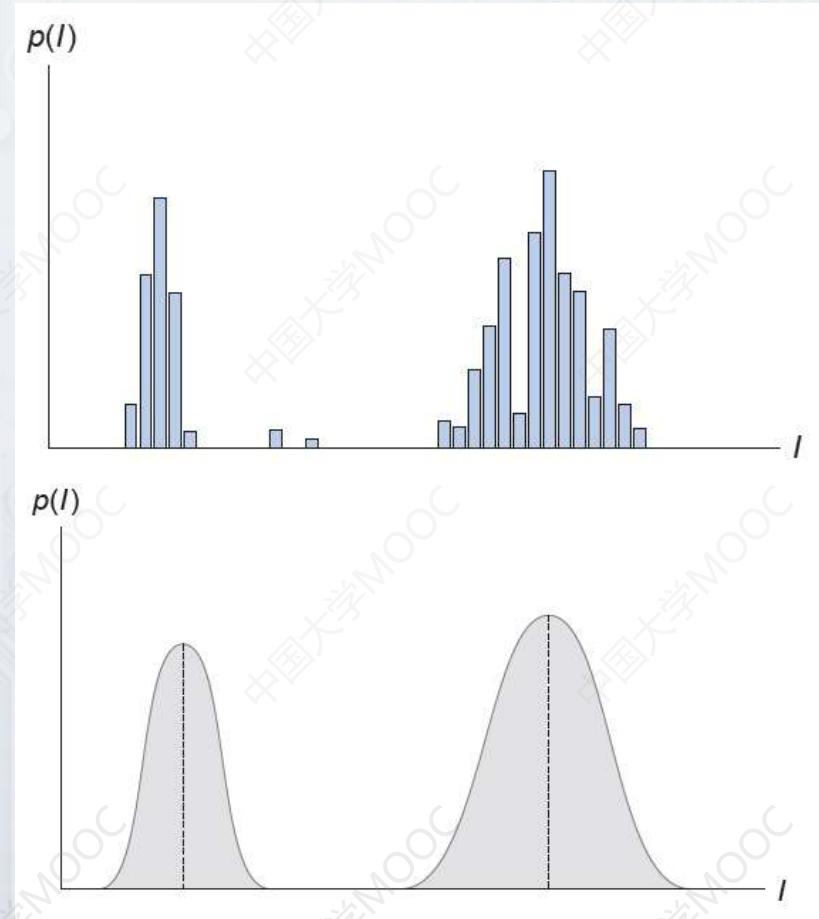
# 混合高斯建模

## 一些问题



如果我们统计某个像素在某段时间内的灰度概率。

### —摇动的树枝，旗帜



# 阴影处理

## 一些问题



背景



Frame t

— 阴影



Frame t+i

阴影



背景



Frame t

阴影

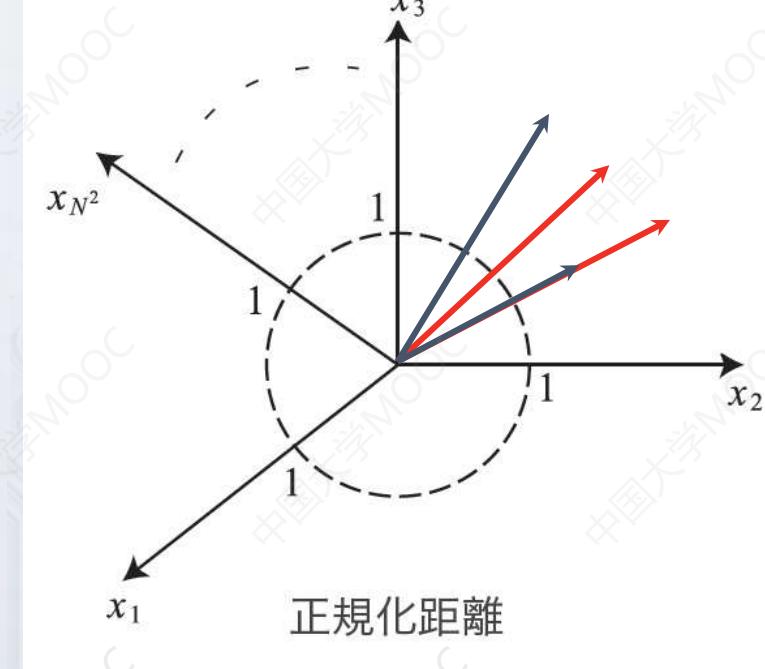


Frame t+i

# 阴影处理

## 一些问题

### 一阴影



# 背景更新

## 一些问题

### —环境光变化



# 背景更新

## 一些问题

— 环境光变化

— 环境光变化

— IIR filters : 使用当前帧对背景图像进行更新

$$B(t) = \alpha \cdot I(t) + (1 - \alpha)B(t - 1)$$
$$(0 < \alpha < 1)$$

$I(t)$ : frame t

$B(t-1)$ : bg image on time t-1

$\alpha$ : a weight factor

# 谢谢！

# 机器视觉技术与应用

## 13. 非固定背景目标检测

李竹

杭州电子科技大学

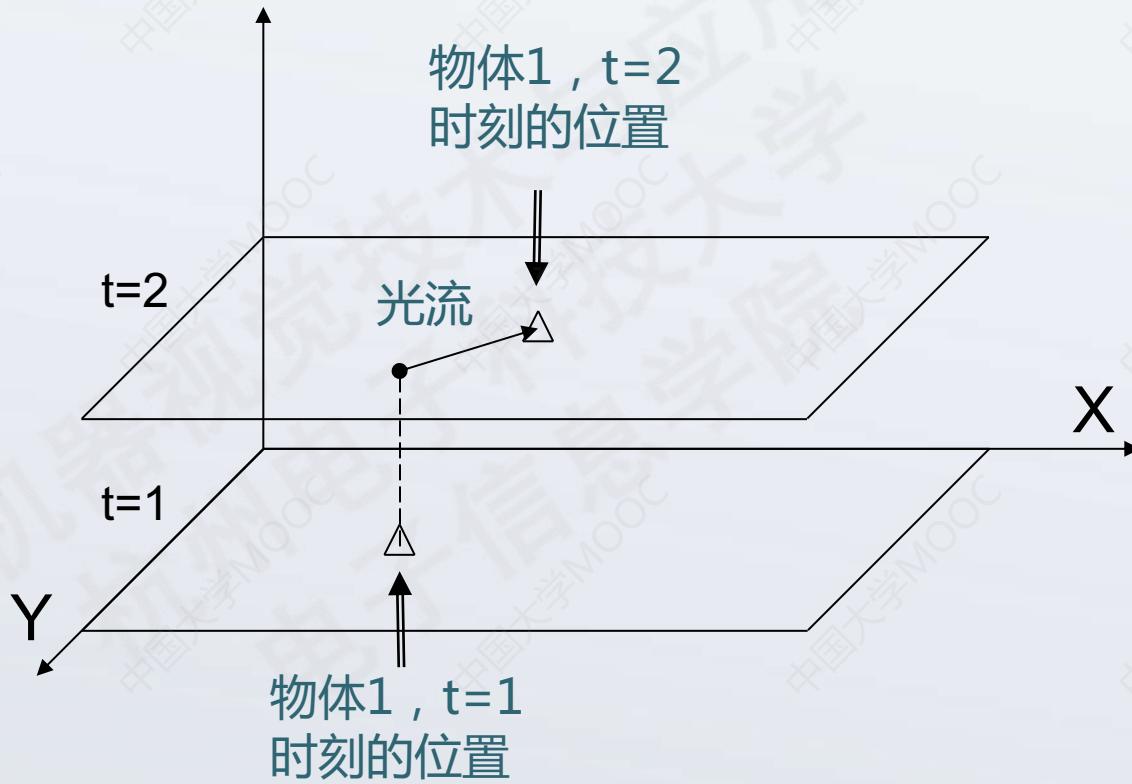
电子信息学院



# 本章概要

1. 光流法
2. 特定物体目标检测

# 光流法

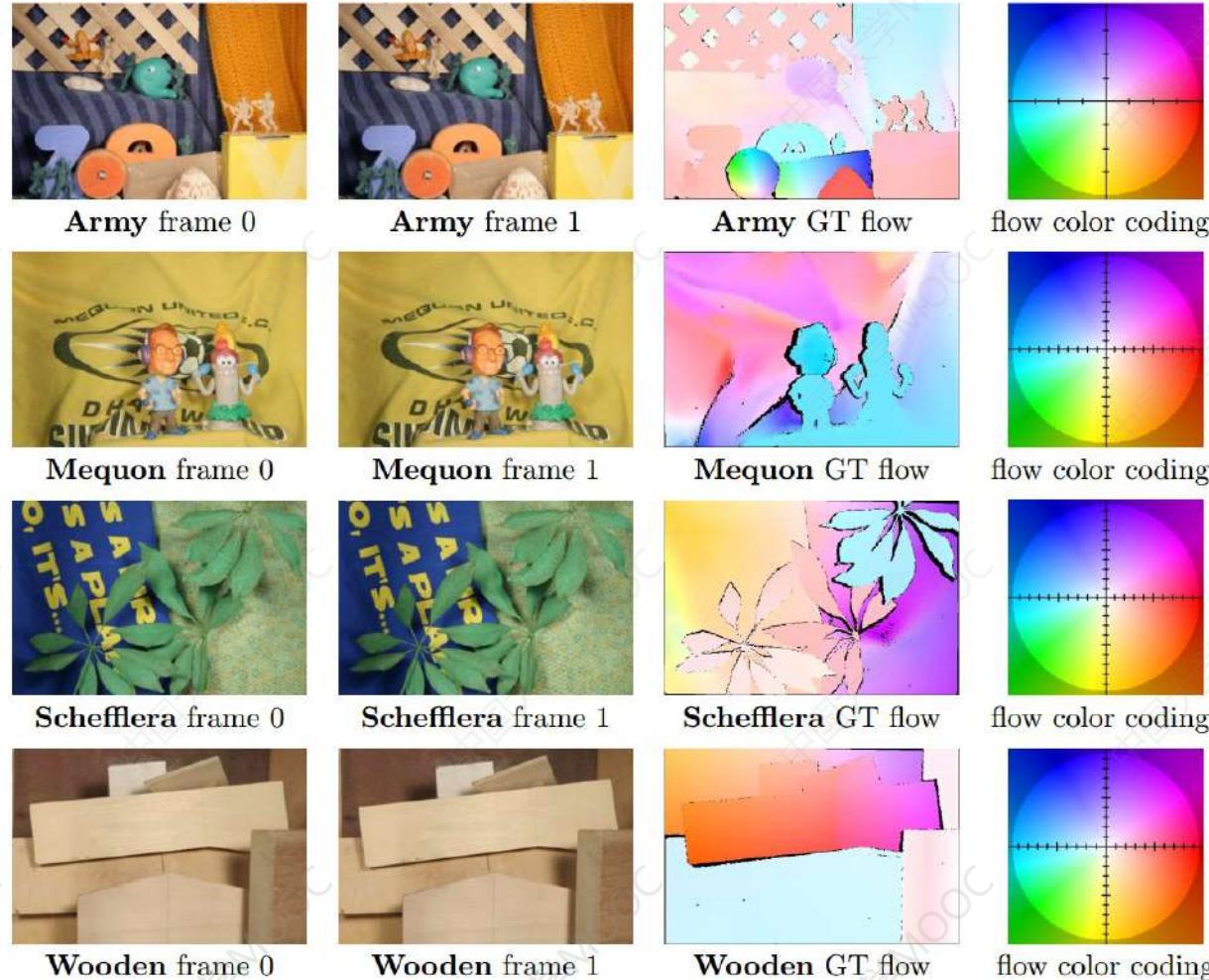


光流是空间运动物体在观测成像面上的像素运动的瞬时速度。

光流场是指图像中所有像素点构成的一种二维(2D)瞬时速度场。

1981年，Horn & Schunck创造性地将二维速度场与亮度变化相结合，引入基本光流约束方程及整体平滑约束条件，建立了光流计算的基本模型。

# 光流法



## 光流法的标准测试数据集

<http://vision.middlebury.edu/flow/>

Baker, S., et al. "A database and evaluation methodology for optical flow." In Proceedings of the IEEE International Conference on Computer Vision 2007:1-8.

# 光流法

假设1:连续的2帧图像之间，同一像素的像素值不变

$$E(x, y, t) = E(x + \Delta x, y + \Delta y, t + \Delta t)$$

$E(x, y, t)$  : t时刻坐标(x,y)的像素值



泰勒分解

$$E(x, y, t) = E(x, y, t) + \Delta x \frac{\partial E}{\partial x} + \Delta y \frac{\partial E}{\partial y} + \Delta t \frac{\partial E}{\partial t}$$

# 光流法

假设1:连续的2帧图像之间，同一像素的像素值不变

$$E(x, y, t) = E(x, y, t) + \Delta x \frac{\partial E}{\partial x} + \Delta y \frac{\partial E}{\partial y} + \Delta t \frac{\partial E}{\partial t}$$

两边除以 $\Delta t$

$$\frac{\Delta x}{\Delta t} \frac{\partial E}{\partial x} + \frac{\Delta y}{\Delta t} \frac{\partial E}{\partial y} + \frac{\partial E}{\partial t} = 0$$

假设2:  $\Delta t$ 无限接近于0

$$\frac{\partial x}{\partial t} = u \quad \frac{\partial y}{\partial t} = v$$

像素值空间梯度  $E_x, E_y$

像素值时间梯度  $E_t$

$$\frac{\partial x}{\partial t} \frac{\partial E}{\partial x} + \frac{\partial y}{\partial t} \frac{\partial E}{\partial y} + \frac{\partial E}{\partial t} = 0$$

$$E_x u + E_y v + E_t = 0$$

# 光流法

假设1:连续的2帧图像之间，同一像素的像素值不变

-已知量  $E_x, E_y, E_t$

$$E_x u + E_y v + E_t = 0$$

-未知数  $u, v$

-假设平滑运动  
-平滑运动约束

$$e_s = \iint ((u_x^2 + u_y^2) + (v_x^2 + v_y^2)) dx dy$$

-光流基本约束式为:

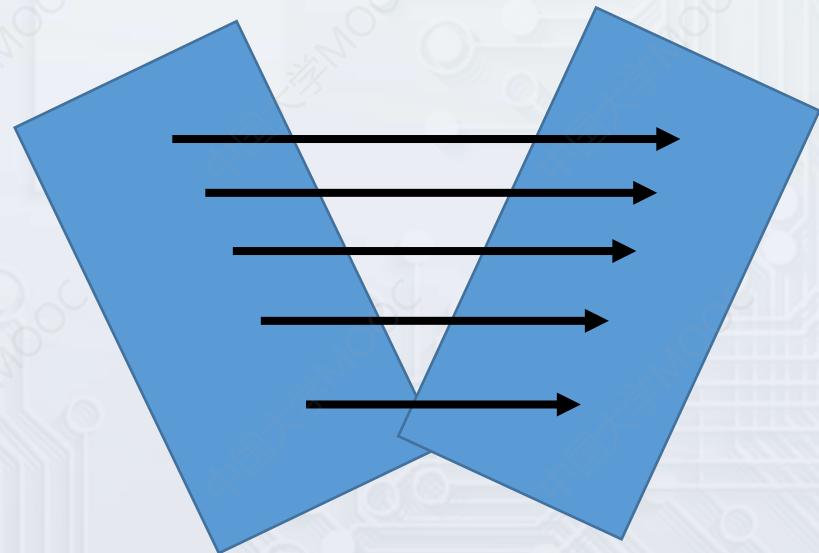
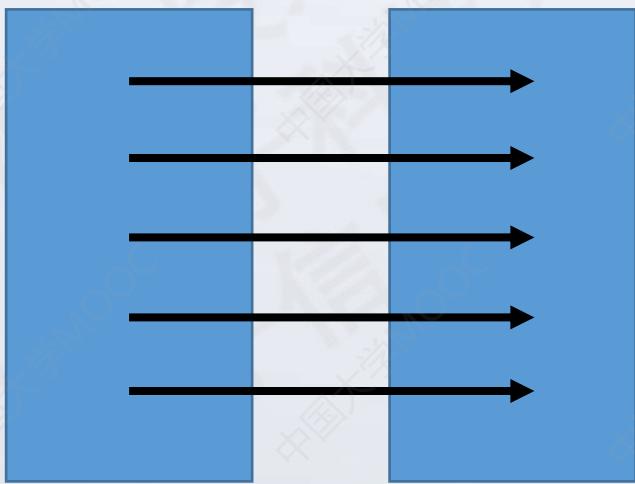
$$e_c = \iint (E_x u + E_y v + E_t)^2 dx dy$$



$$\text{Minimize } e_c + \alpha e_c$$

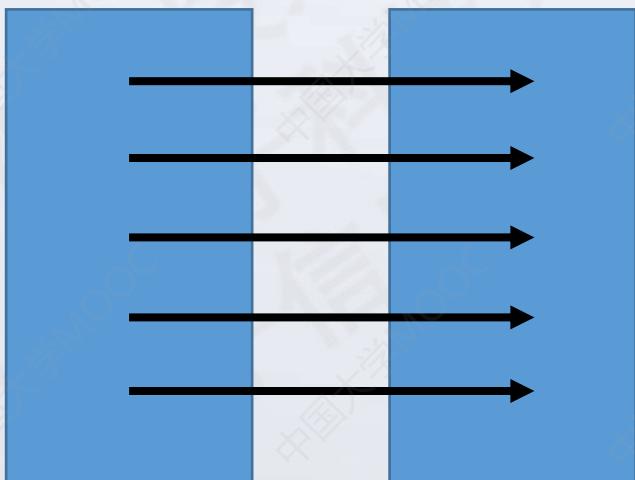
# 光流法

## 平滑运动约束



# 光流法

假设相邻的像素具有相同的运动向量



$$\begin{bmatrix} I_x(p_1) & I_y(p_1) \\ I_x(p_2) & I_y(p_2) \\ \vdots & \vdots \\ I_x(p_{25}) & I_y(p_{25}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(p_1) \\ I_t(p_2) \\ \vdots \\ I_t(p_{25}) \end{bmatrix}$$

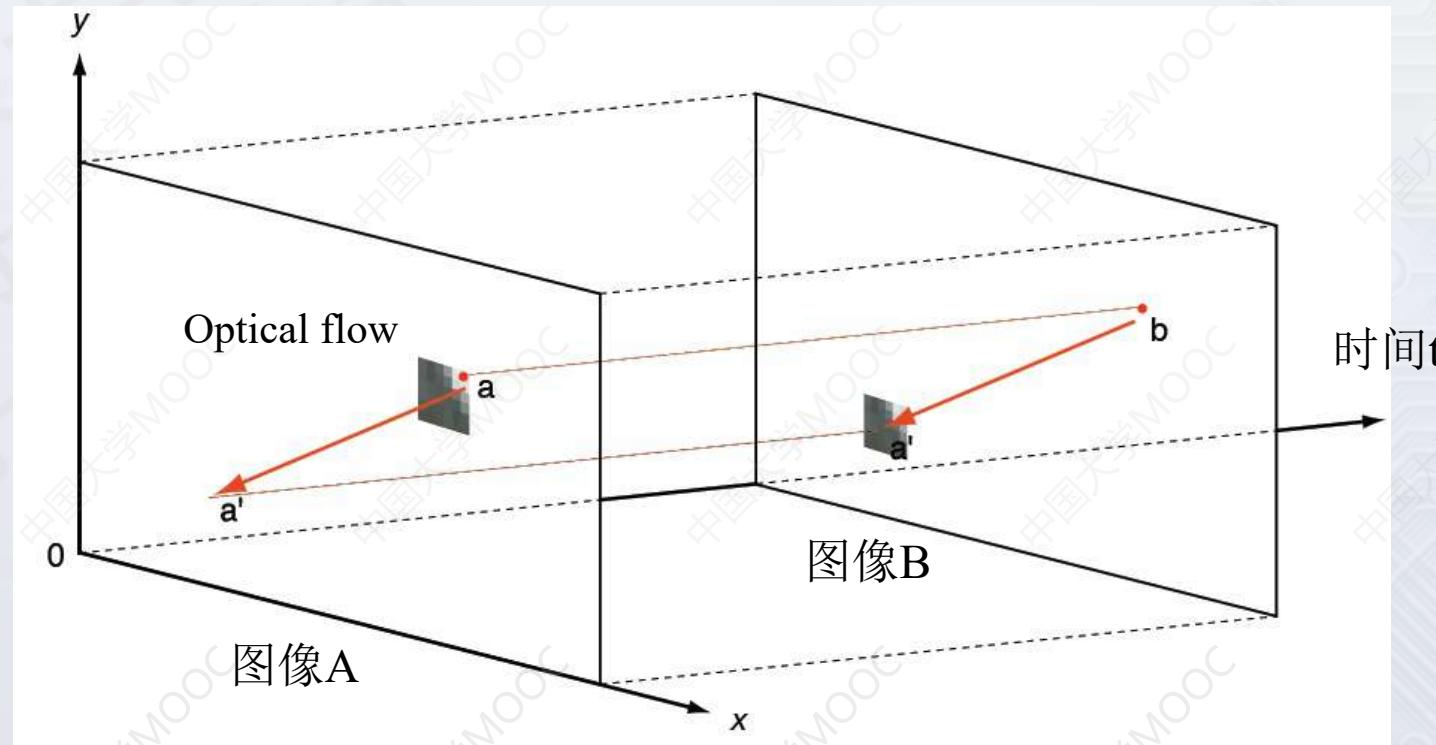
$A$   
 $25 \times 2$

$d$   
 $2 \times 1$

$b$   
 $25 \times 1$

# 光流法

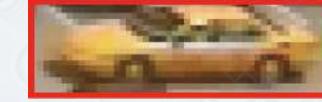
Block matching法：计算图像A中的某一像素的光流，将该像素为中心的一个区域作为模板，在下一帧图像中进行模板匹配。



# 特定物体追踪



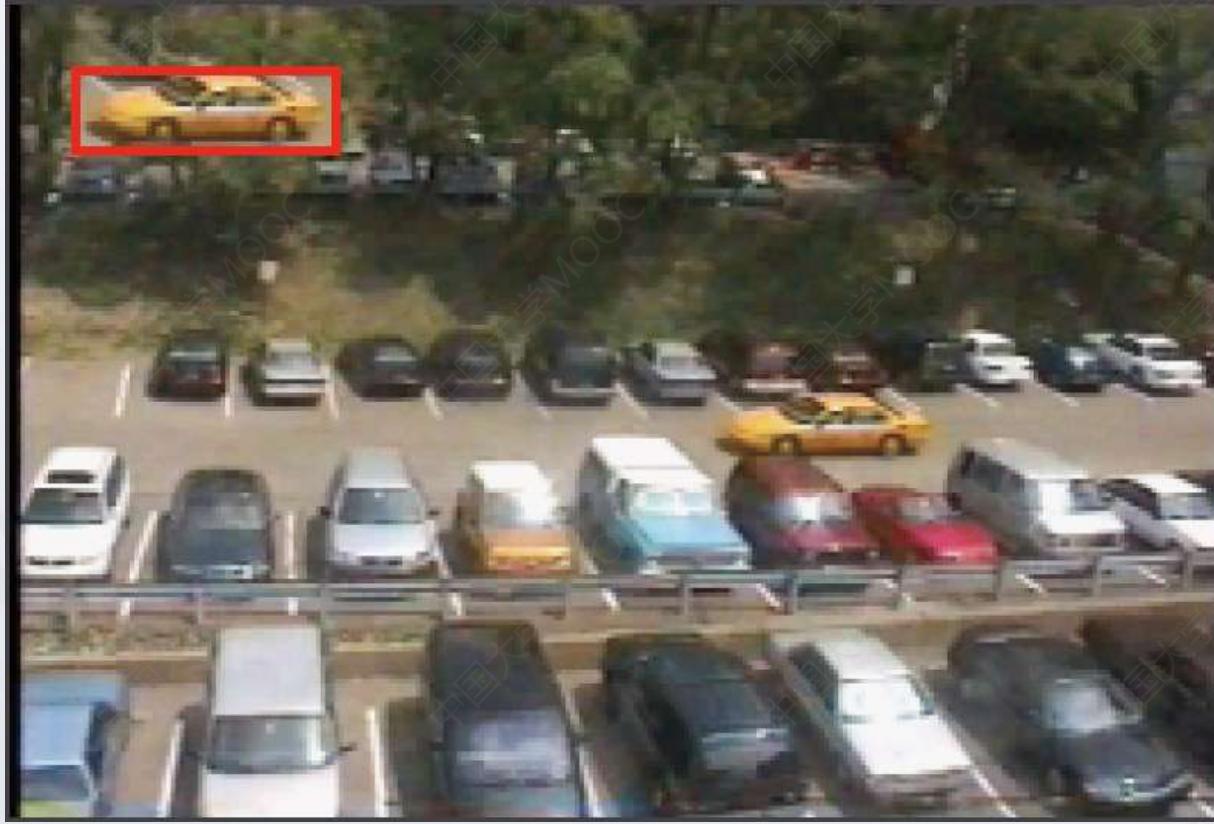
输入图像



模板

任务：在输入图像中找到特殊目标物体，即模板图像的位置

# 特定物体追踪



输入图像



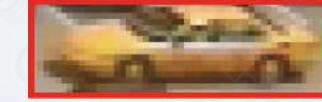
模板

任务：在输入图像中找到特殊目标物体，即模板图像的位置

# 特定物体追踪



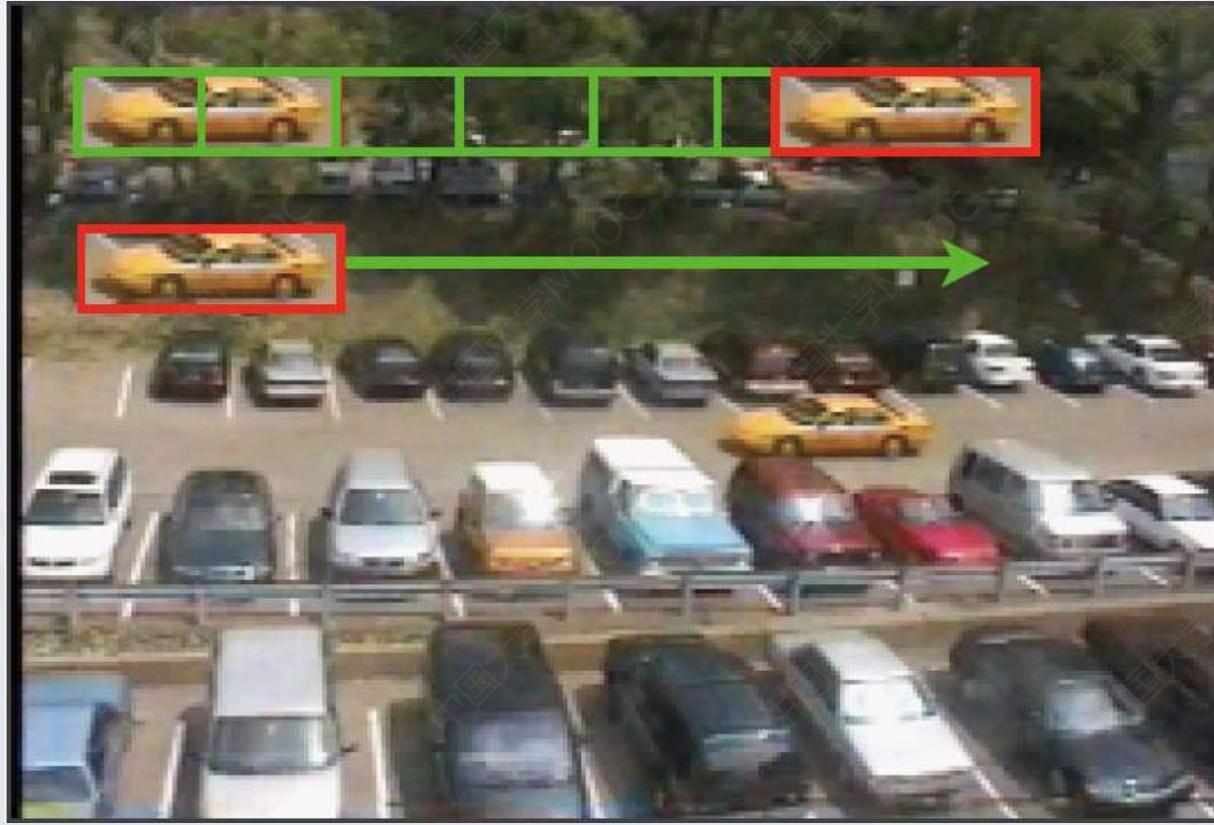
输入图像



模板

任务：在输入图像中找到特殊目标物体，即模板图像的位置

# 特定物体追踪



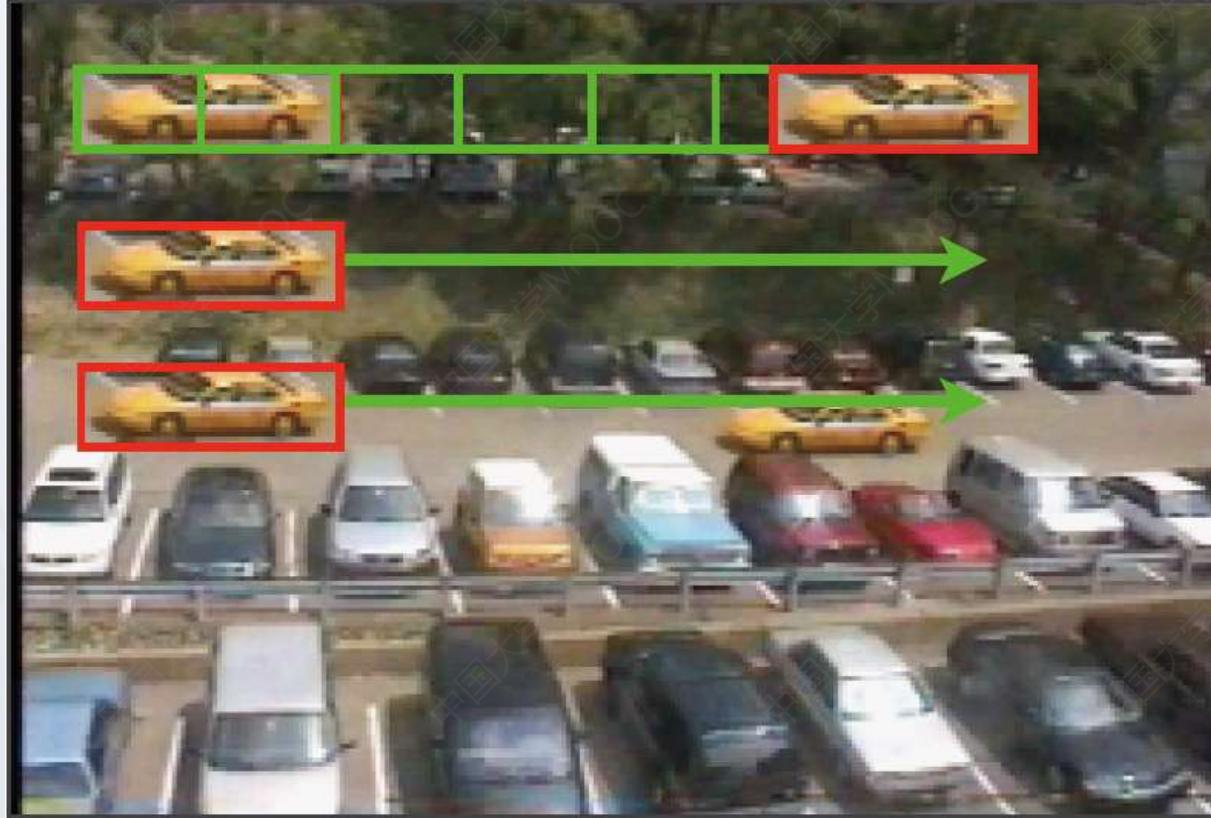
输入图像



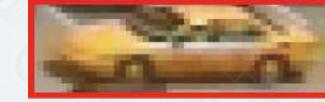
模板

任务：在输入图像中找到特殊目标物体，即模板图像的位置

# 特定物体追踪



输入图像



模板

任务：在输入图像中找到特殊目标物体，即模板图像的位置

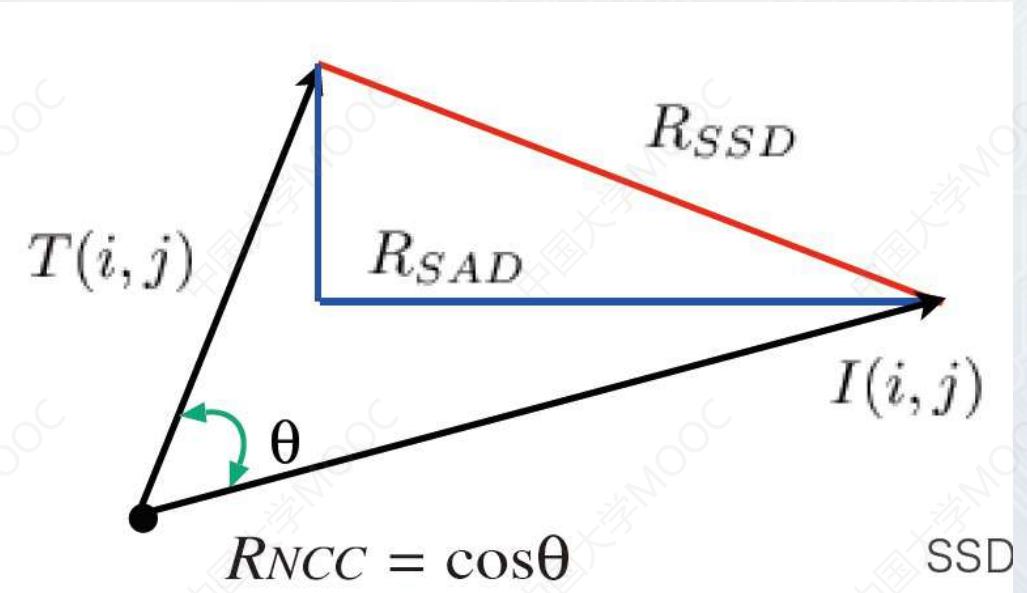
# 特定物体追踪

相似度的计算 : SSD , SAD ,

$$R_{SSD} = \sum_{j=0}^{N-1} \sum_{i=0}^{M-1} (I(i, j) - T(i, j))^2$$

$$R_{SAD} = \sum_{j=0}^{N-1} \sum_{i=0}^{M-1} |I(i, j) - T(i, j)|$$

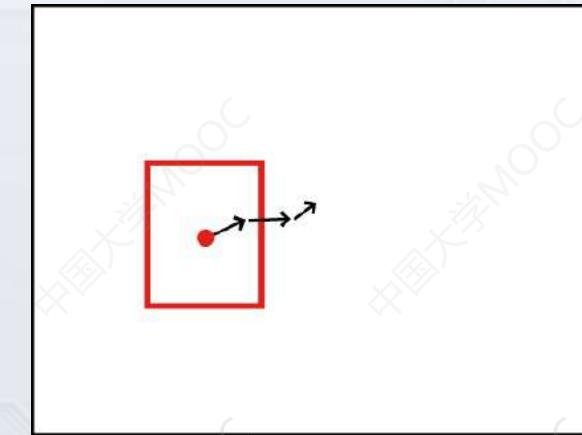
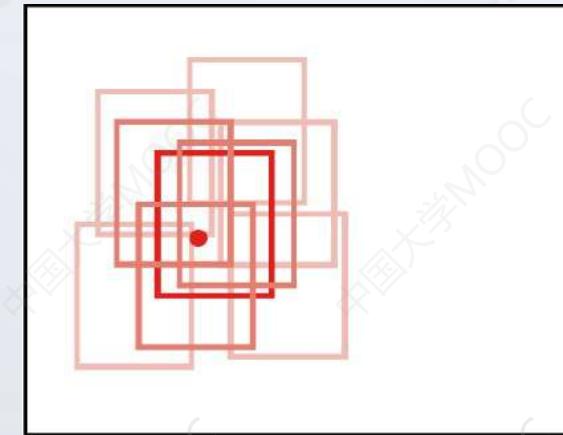
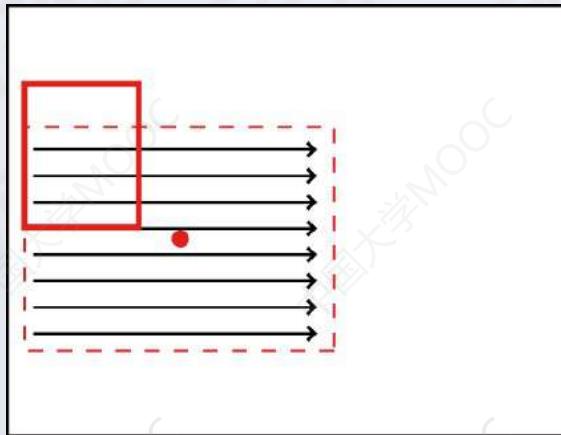
$$R_{NCC} = \frac{\sum_{j=0}^{N-1} \sum_{i=0}^{M-1} (I(i, j)T(i, j))}{\sqrt{\sum_{j=0}^{N-1} \sum_{i=0}^{M-1} I(i, j)^2 \times \sum_{j=0}^{N-1} \sum_{i=0}^{M-1} T(i, j)^2}}$$



# 特定物体追踪

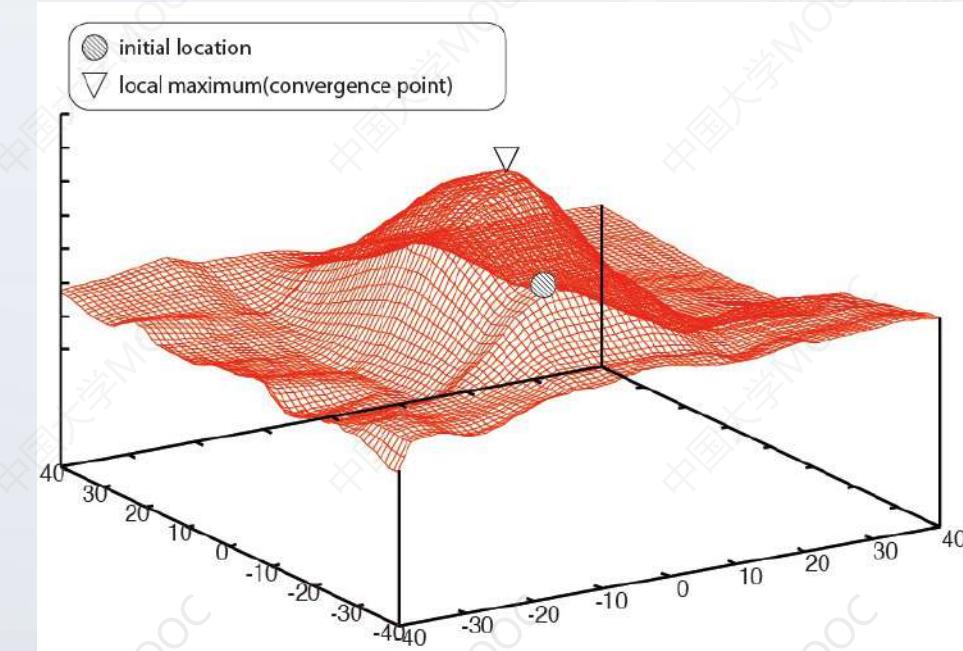
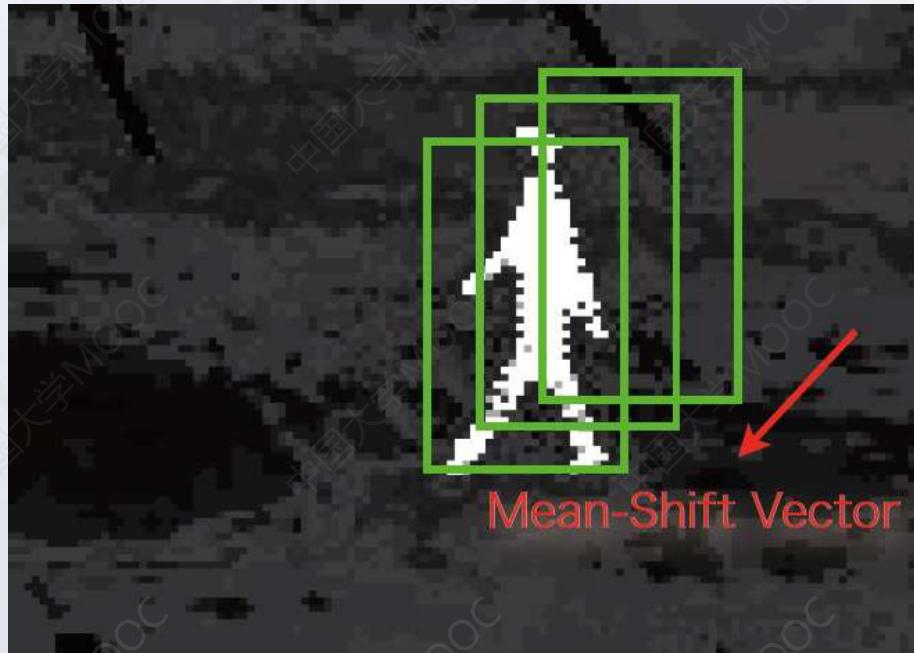
改进：

- 相似度：通过直方图计算
- 追踪方法：Mean-Shift , Particle Filter

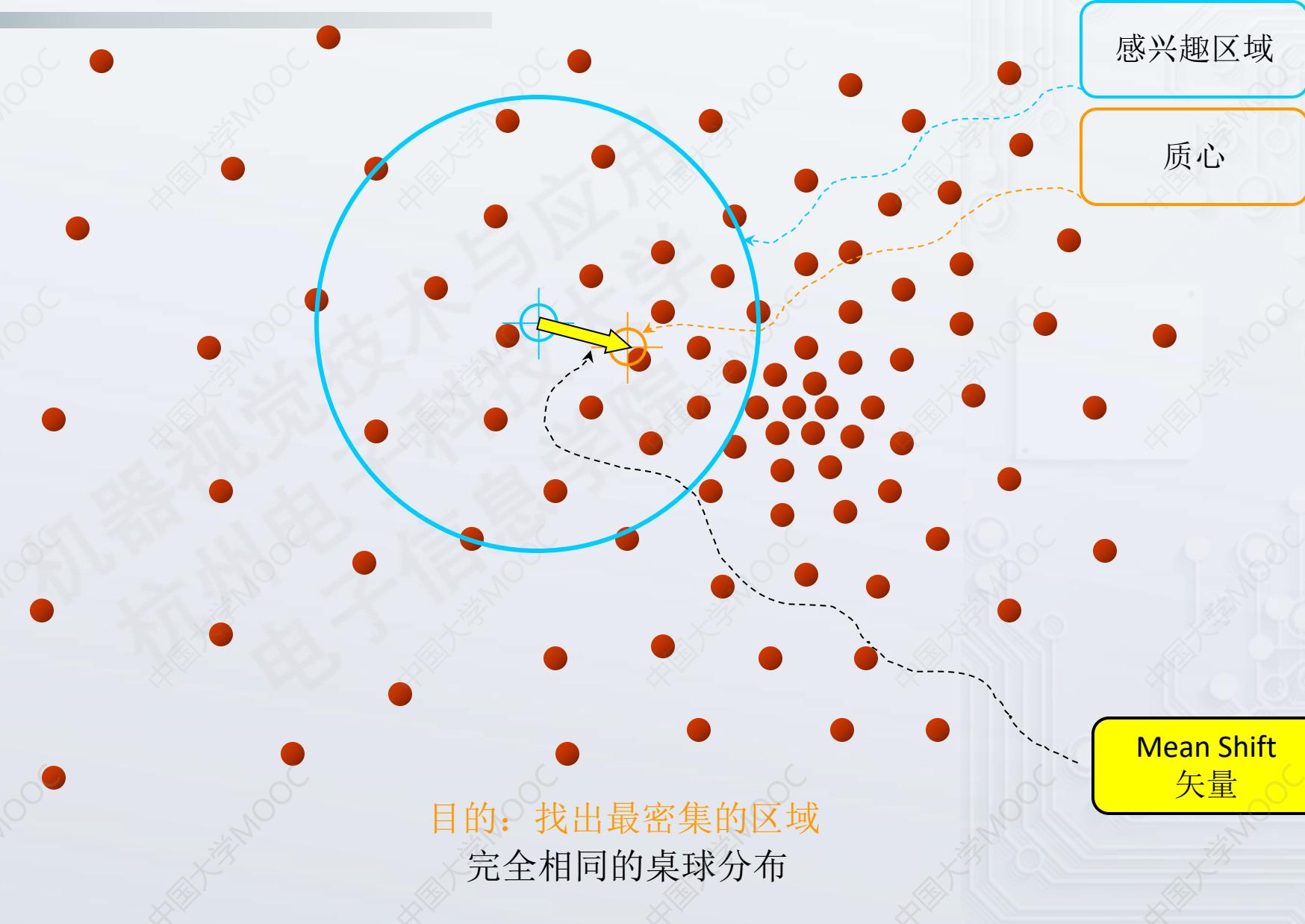


# Mean-Shift

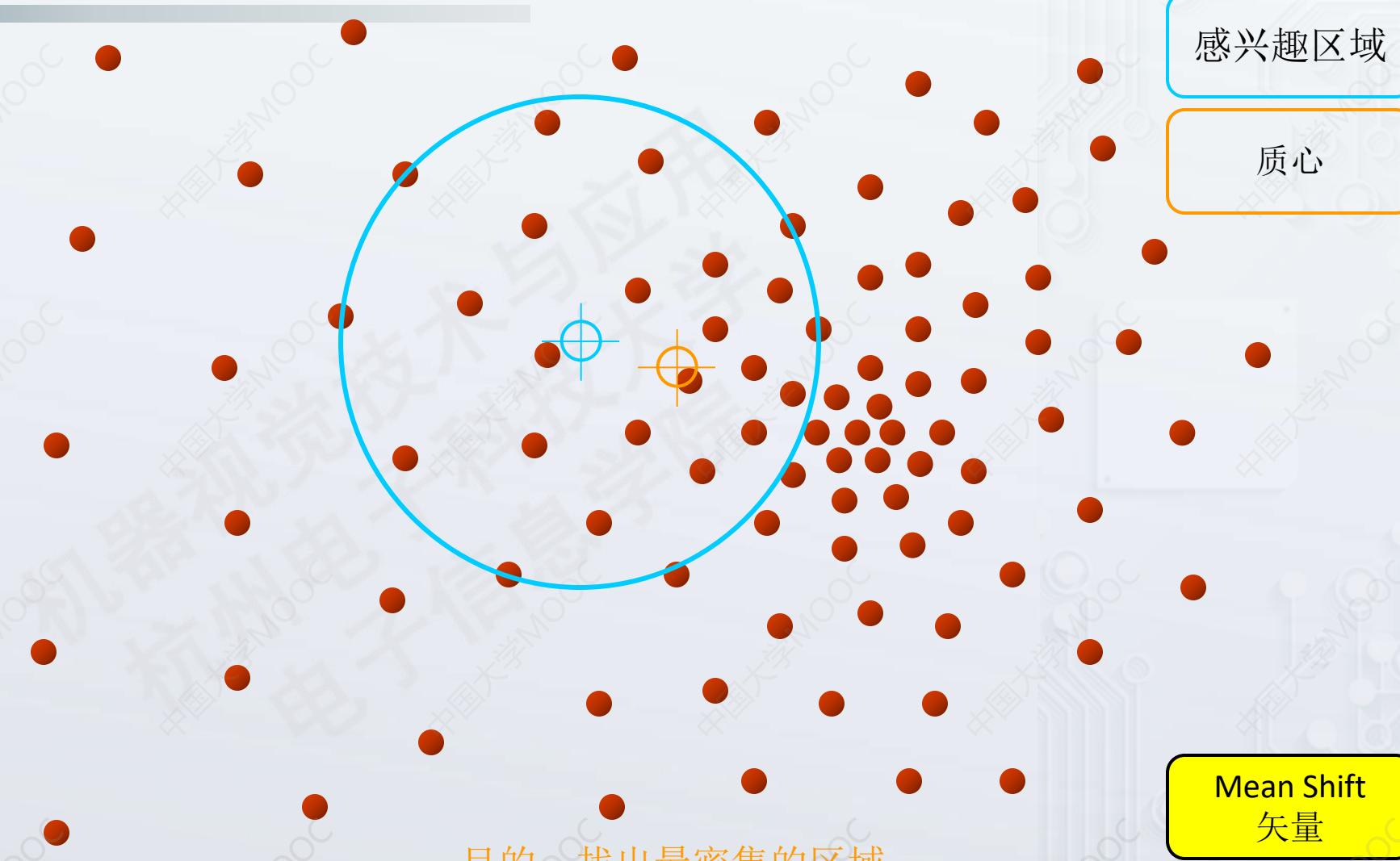
Mean Shift算法,通常是指一个迭代的步骤,即先算出当前点的偏移均值,移动该点到其偏移均值,然后以此为新的起始点,继续移动,直到满足一定的条件结束.



# Mean-Shift

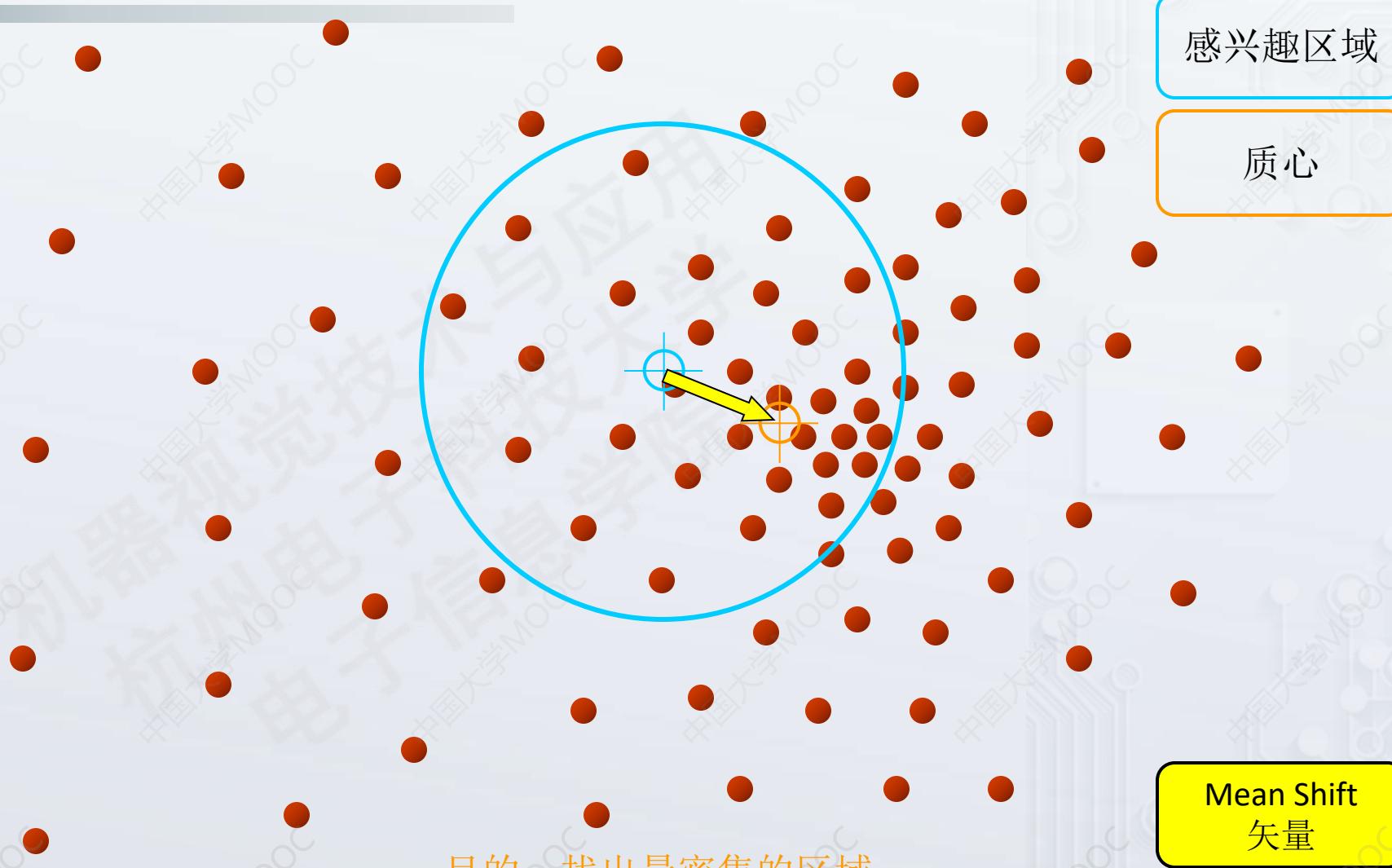


# Mean-Shift



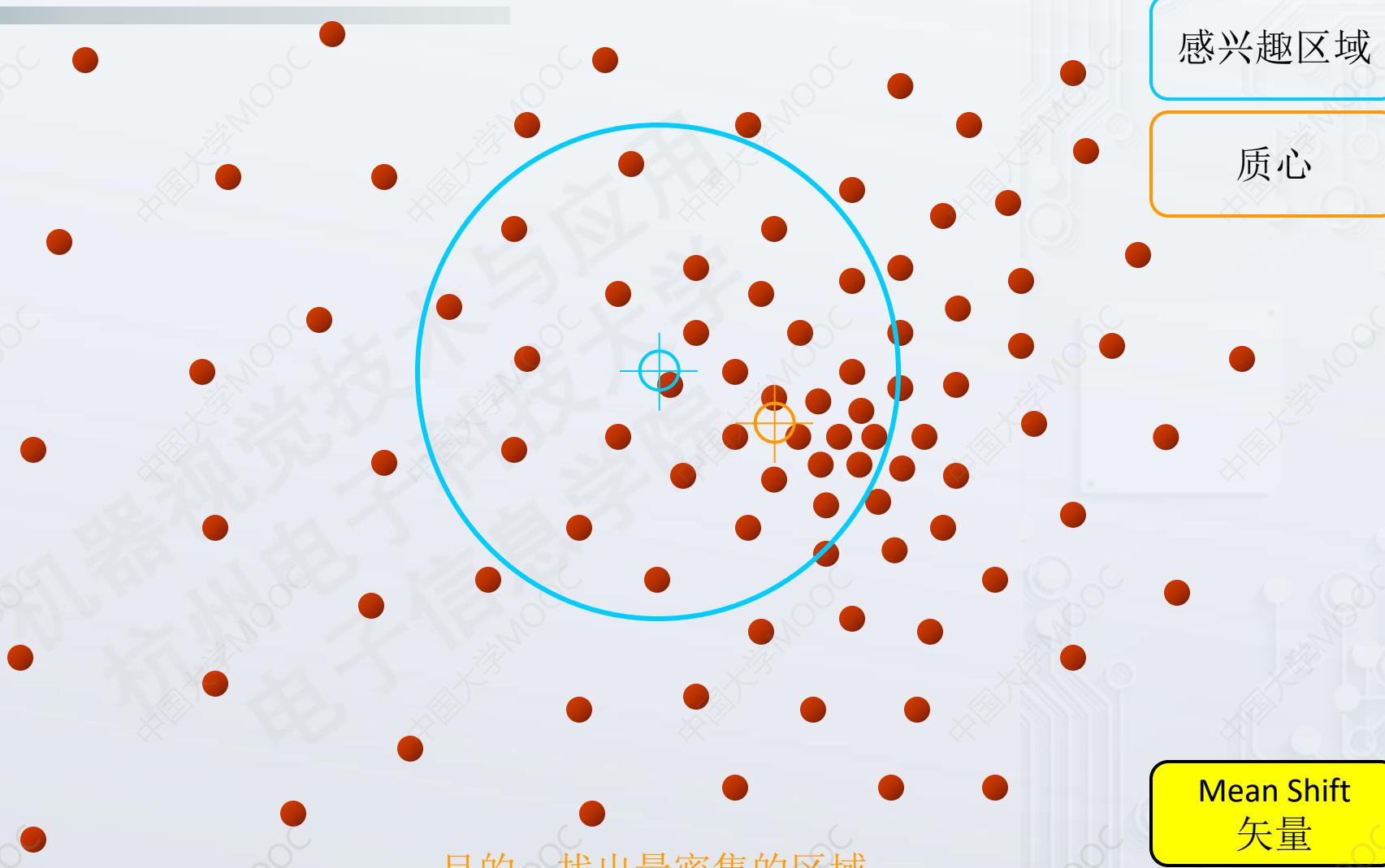
目的：找出最密集的区域  
完全相同的桌球分布

# Mean-Shift

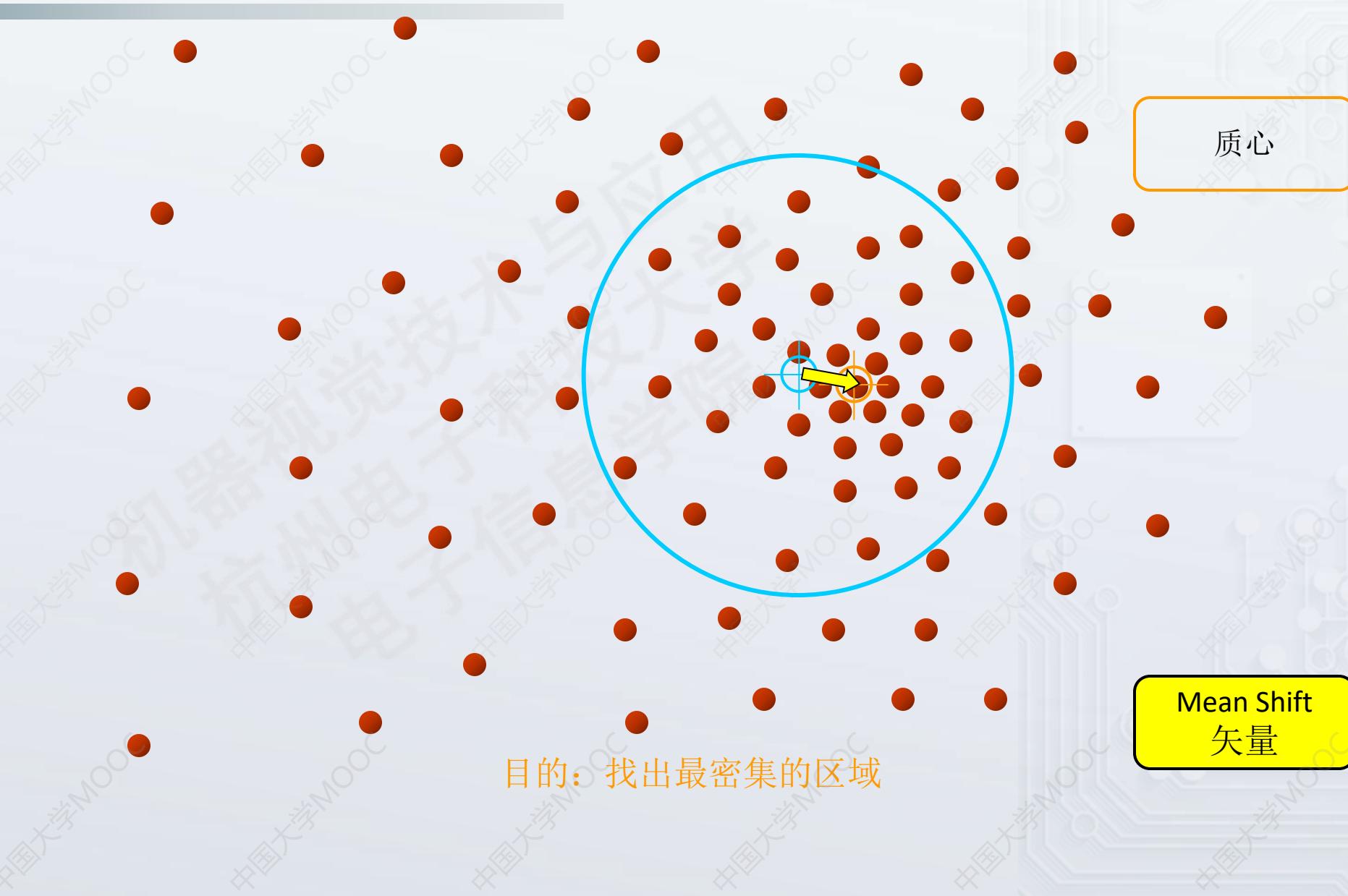


目的：找出最密集的区域  
完全相同的桌球分布

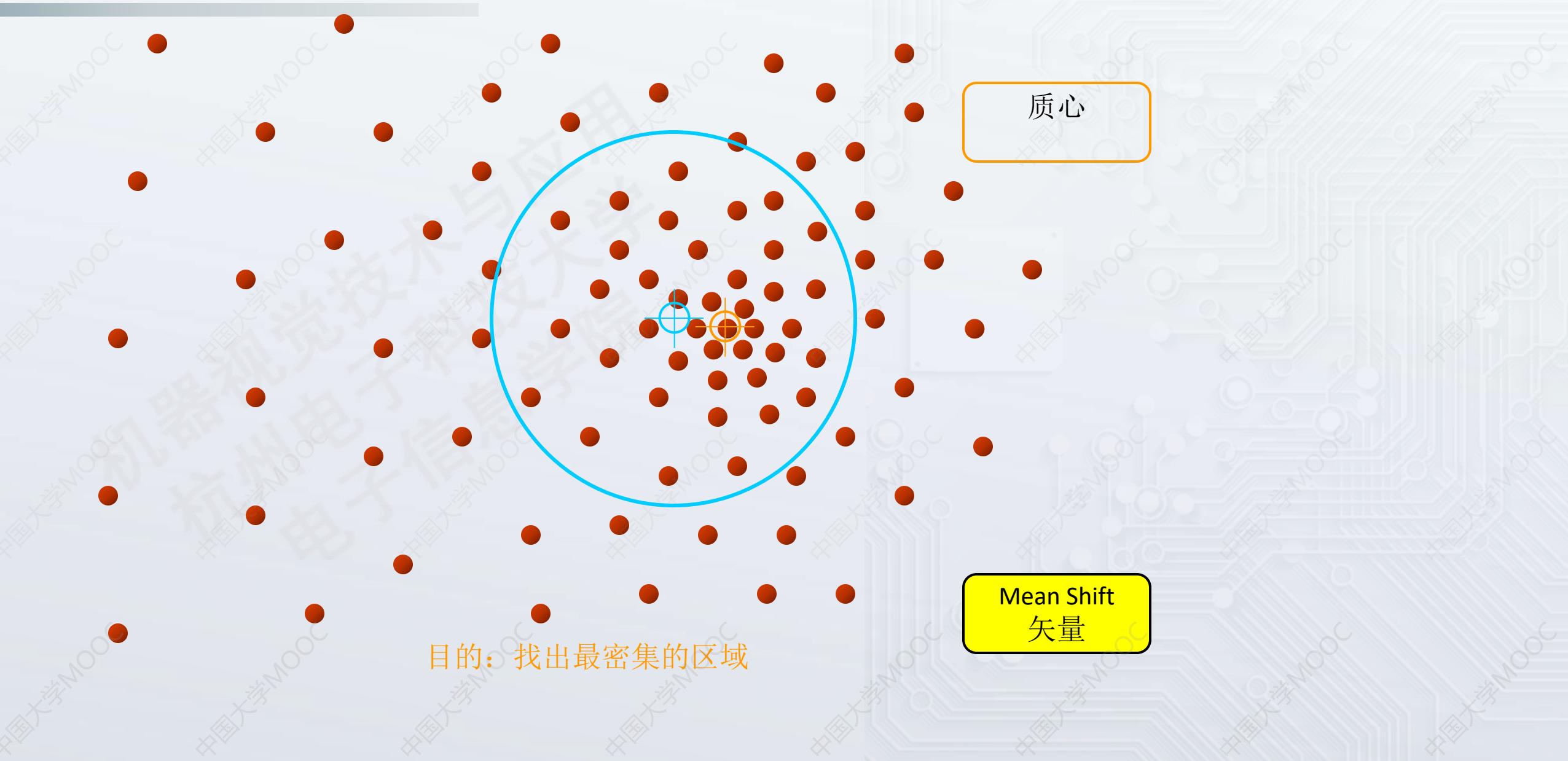
# Mean-Shift



# Mean-Shift

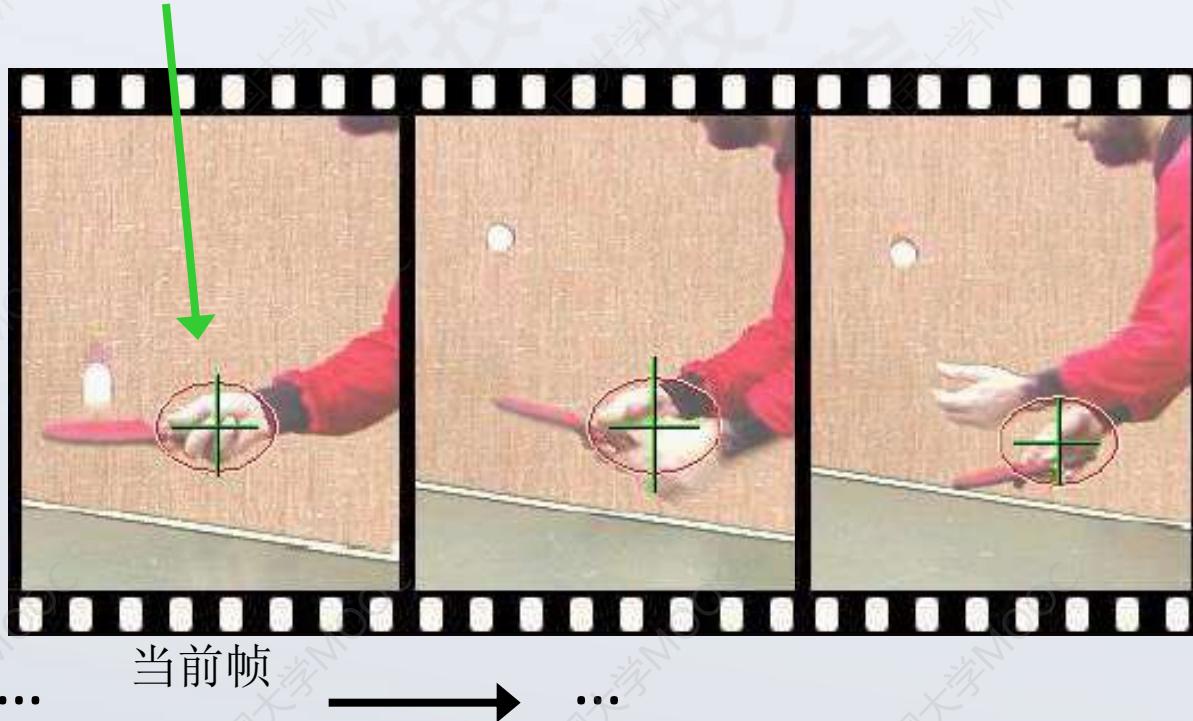


# Mean-Shift

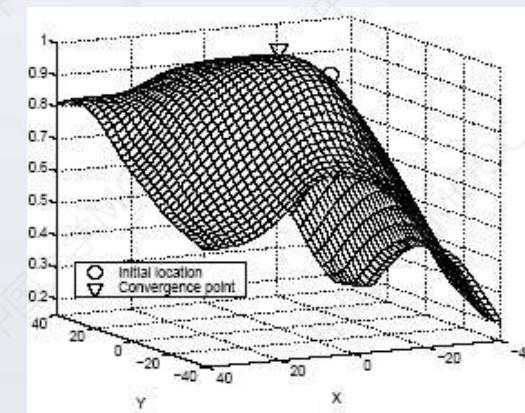
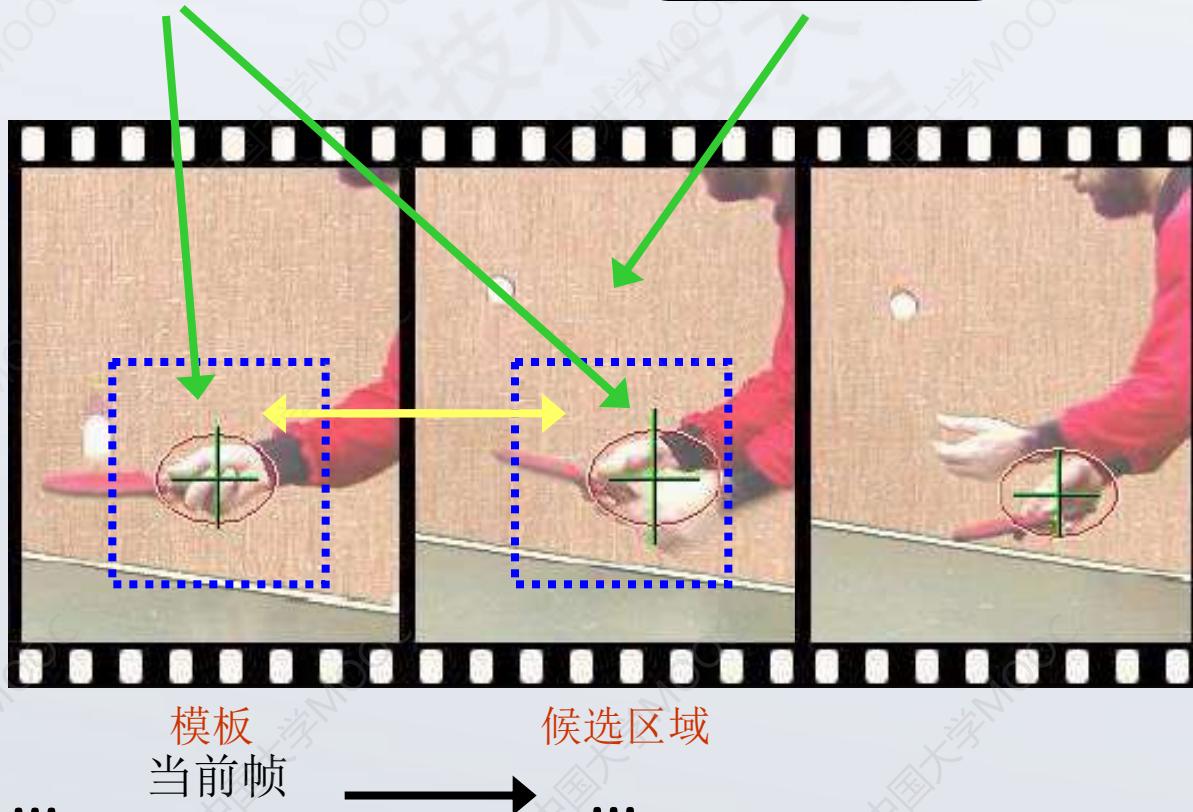
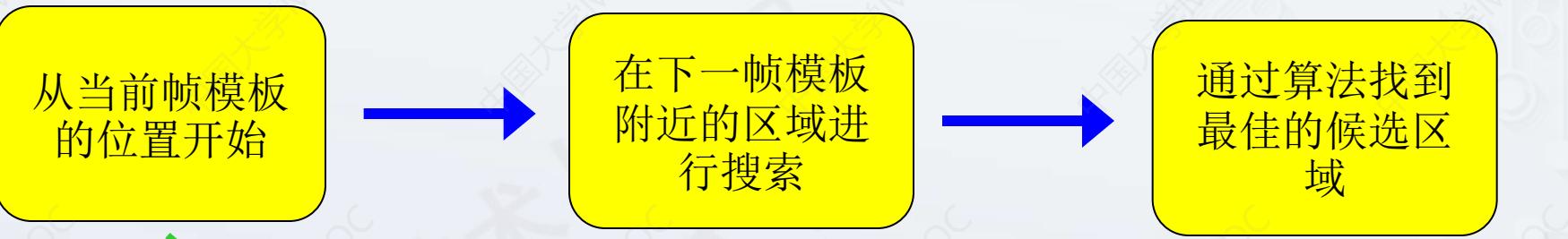


# Mean-Shift

- 在当前帧选取一个合适的模板
- 选择一个特征空间
- 在选择的特征空间中表示选取的模板



# Mean-Shift



# Particle filter

- 初始化，生成粒子集
- 预测每个粒子
- 计算直方图距离（相似度）
- 根据直方图距离计算每个粒子
- 选择目标的位置作为具有最小直方图距离的粒子
- 对粒子进行采样以进行下一次迭代。

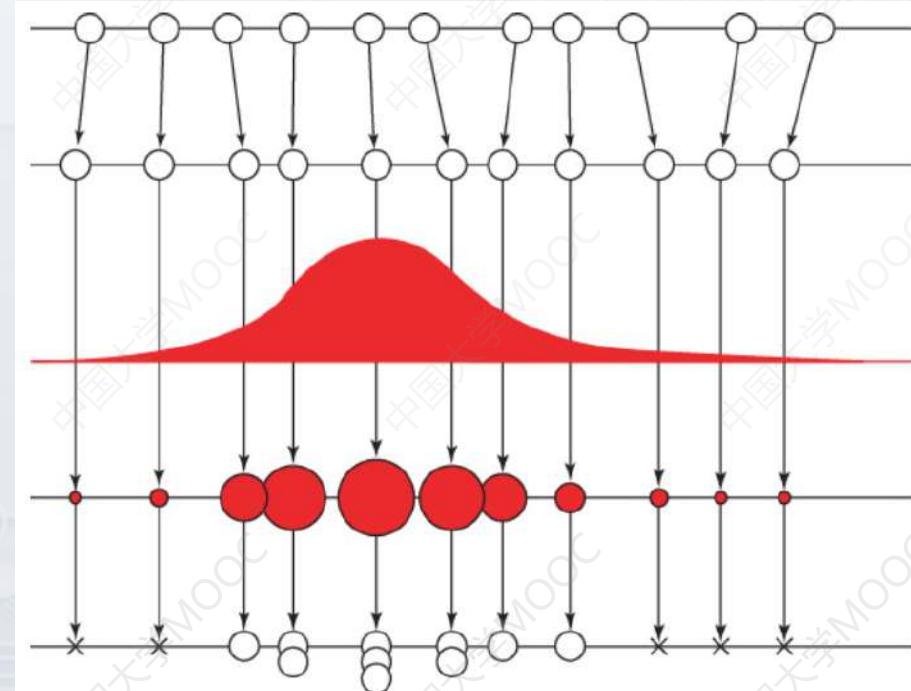
在帧t上生成粒子集

帧t + 1的预测

观察

估计

生成新的粒子集



# Particle filter

- 初始化，生成粒子集
- 预测每个粒子
- 计算直方图距离（相似度）
- 根据直方图距离计算每个粒子
- 选择目标的位置作为具有最小直方图距离的粒子
- 对粒子进行采样以进行下一次迭代。

在帧t上生成粒子集

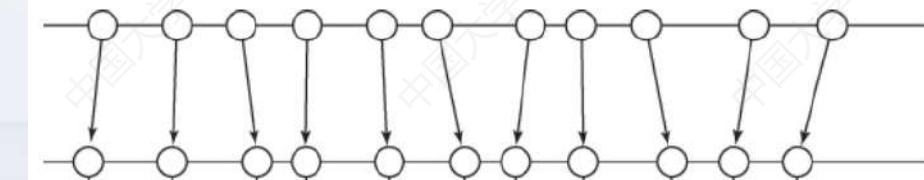


# Particle filter

- 初始化，生成粒子集
- 预测每个粒子
- 计算直方图距离（相似度）
- 根据直方图距离计算每个粒子
- 选择目标的位置作为具有最小直方图距离的粒子
- 对粒子进行采样以进行下一次迭代。

在帧t上生成粒子集

帧t + 1的预测



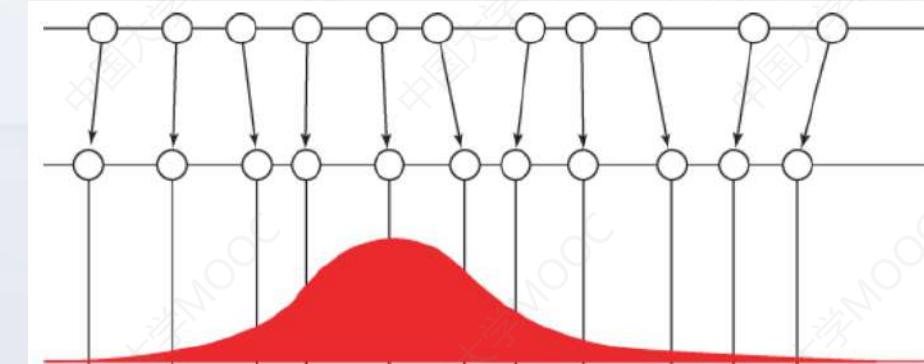
# Particle filter

- 初始化，生成粒子集
- 预测每个粒子
- 计算直方图距离（相似度）
- 根据直方图距离计算每个粒子
- 选择目标的位置作为具有最小直方图距离的粒子
- 对粒子进行采样以进行下一次迭代。

在帧t上生成粒子集

帧t + 1的预测

观察



# Particle filter

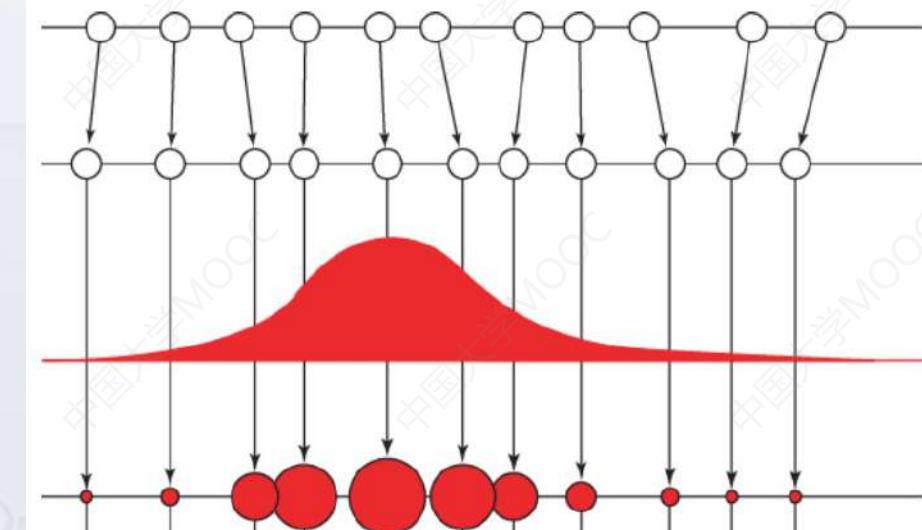
- 初始化，生成粒子集
- 预测每个粒子
- 计算直方图距离（相似度）
- 根据直方图距离计算每个粒子
- 选择目标的位置作为具有最小直方图距离的粒子
- 对粒子进行采样以进行下一次迭代。

在帧t上生成粒子集

帧t + 1的预测

观察

估计



# Particle filter

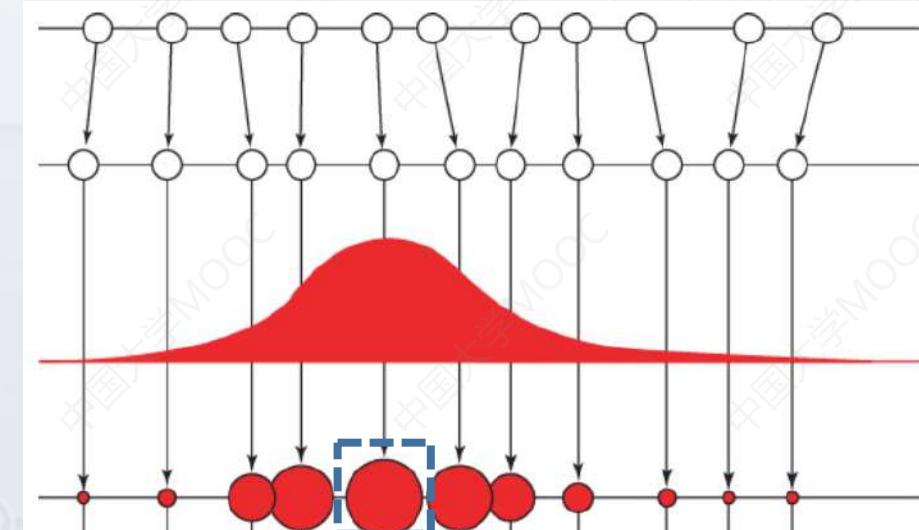
- 初始化，生成粒子集
- 预测每个粒子
- 计算直方图距离（相似度）
- 根据直方图距离计算每个粒子
- 选择目标的位置作为具有最小直方图距离的粒子
- 对粒子进行采样以进行下一次迭代。

在帧t上生成粒子集

帧t + 1的预测

观察

估计



# Particle filter

- 初始化，生成粒子集
- 预测每个粒子
- 计算直方图距离（相似度）
- 根据直方图距离计算每个粒子
- 选择目标的位置作为具有最小直方图距离的粒子
- 对粒子进行采样以进行下一次迭代。

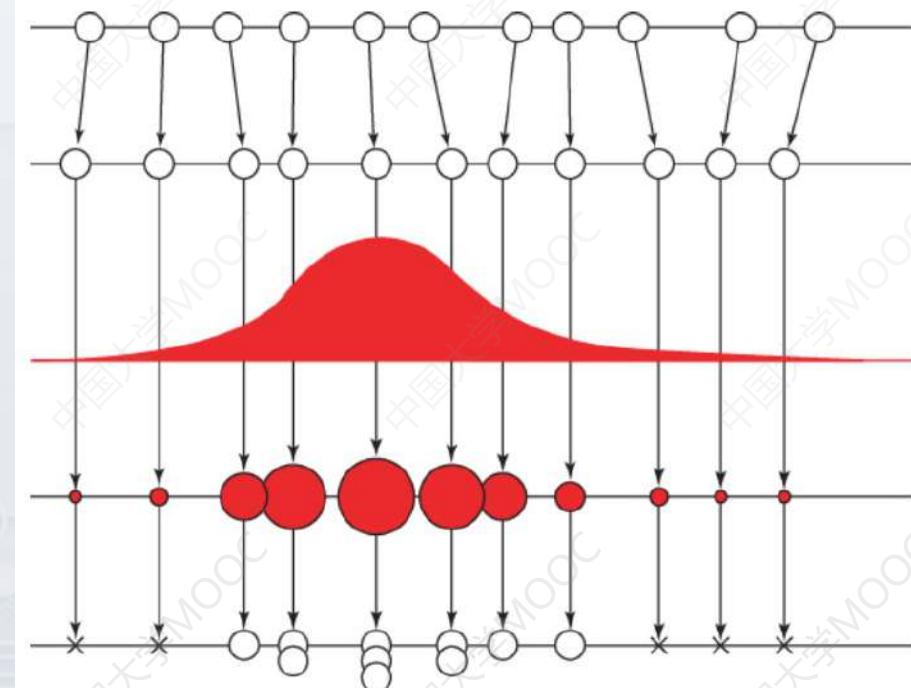
在帧t上生成粒子集

帧t + 1的预测

观察

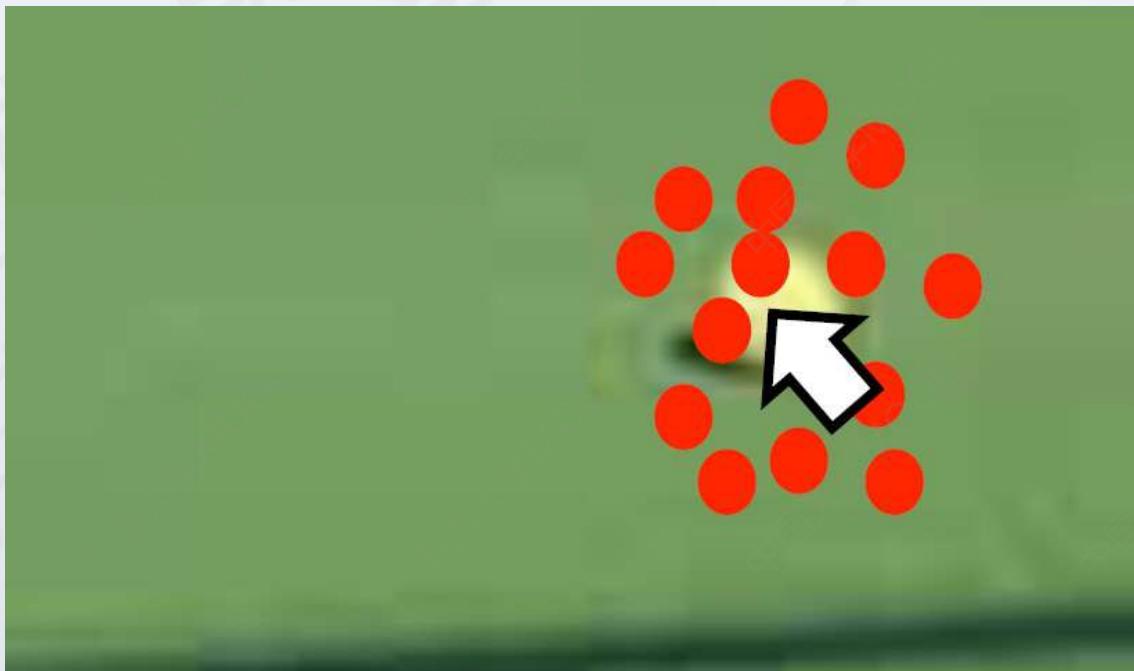
估计

生成新的粒子集



# Particle filter

在帧t上生成粒子集



# Particle filter

使用某些转换模型来计算新位置

例如均匀线性运动模型

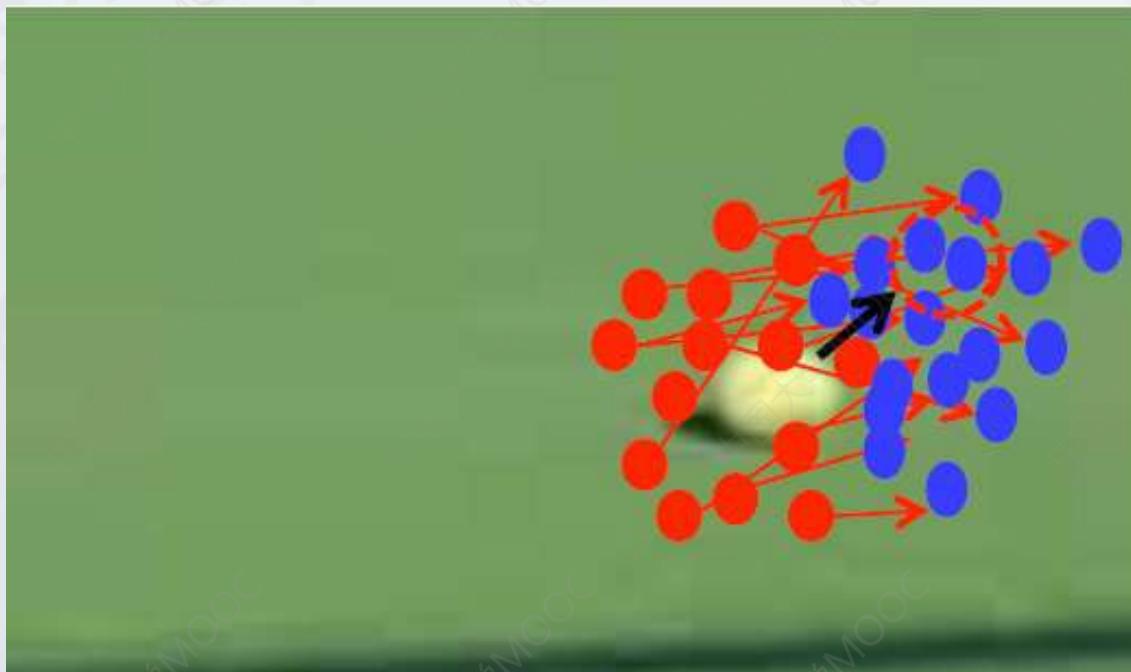
$$x_{t+1} = x_t + u_t \Delta t + w_x$$

x,y: 位置

$$y_{t+1} = y_t + v_t \Delta t + w_y$$

u,v: 速度

w<sub>x</sub>,w<sub>y</sub>: 噪声



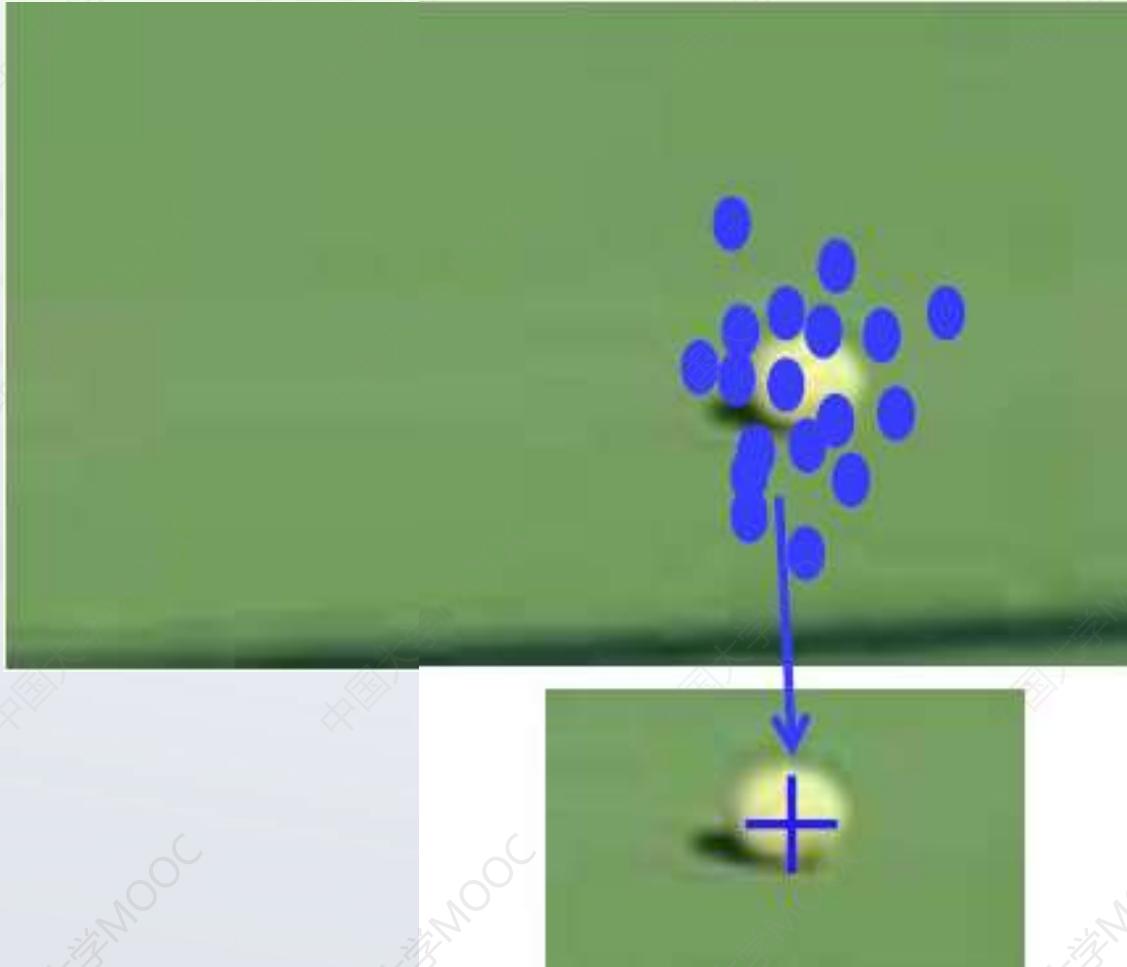
# Particle filter

在帧t上生成粒子集



# Particle filter

在帧t上生成粒子集



# 谢谢！

# 数字图像处理

## 14. 图像识别

李竹

杭州电子科技大学

电子信息学院



# 本章概要

1. SVM体系
2. 神经网络体系

# 分类

1. 监督学习
2. 无监督学习
3. 半监督学习
4. 强化学习

# 监督学习

从给定的训练数据集中学习一个函数(模型)，当新的数据被输入时，可以根据函数(模型)预测结果。

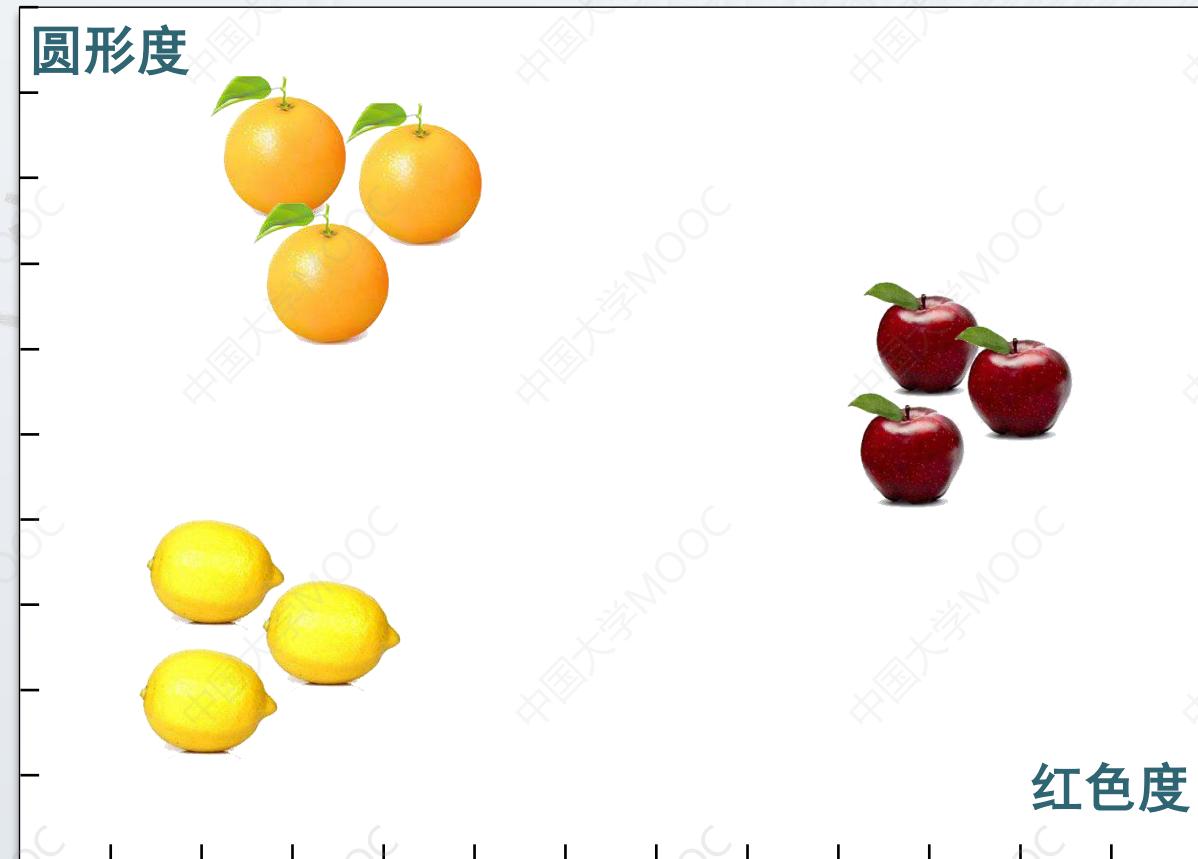
用以训练函数(模型)的数据，称为“训练数据”，在监督学习中，每个训练数据都同时有一个明确的标识，称为标注。

例：以颜色和形状识别，区分水果



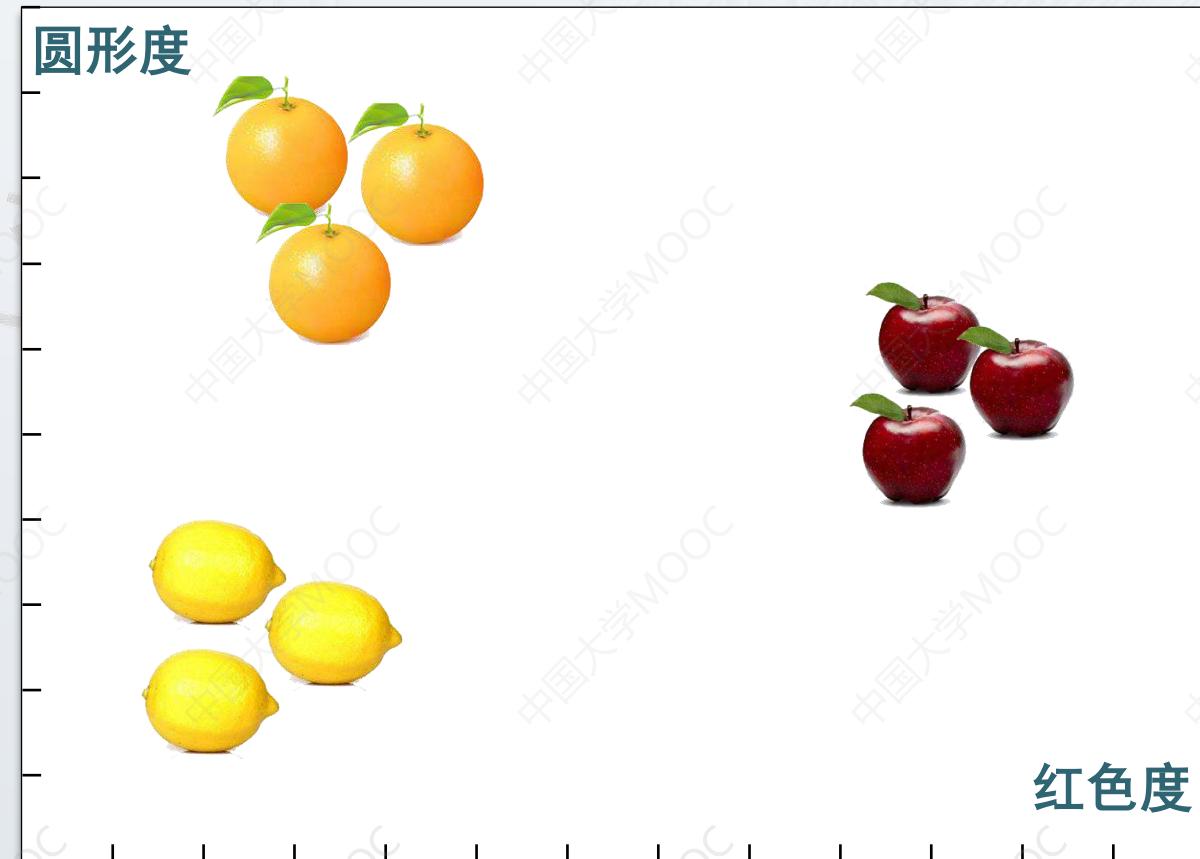
# 监督学习

将两个特征量量化  
可以构建一个2维的特征空  
间，这样每个训练样本可以  
映射到特征空间中。



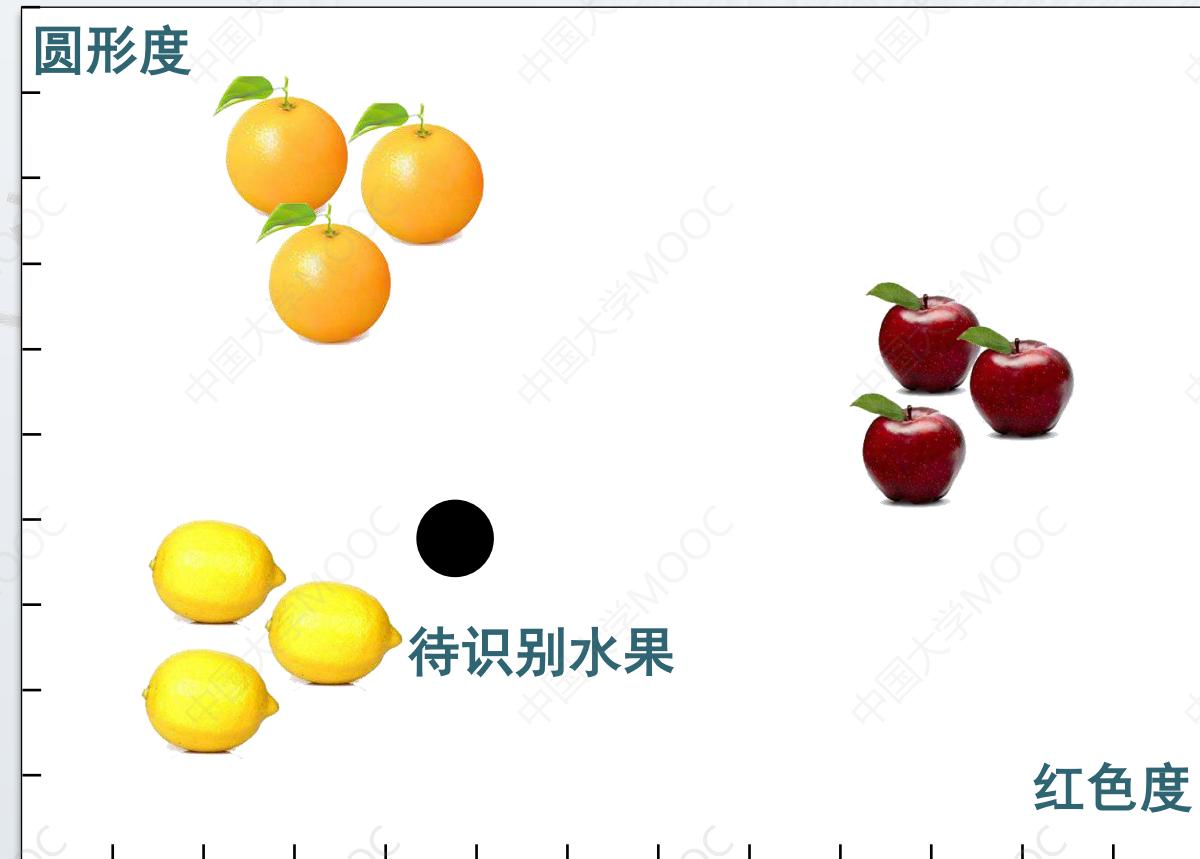
# 监督学习

当一个待识别的样本输入后，同样将圆形度及红色度两个特征量计算得到，并映射到特征空间。



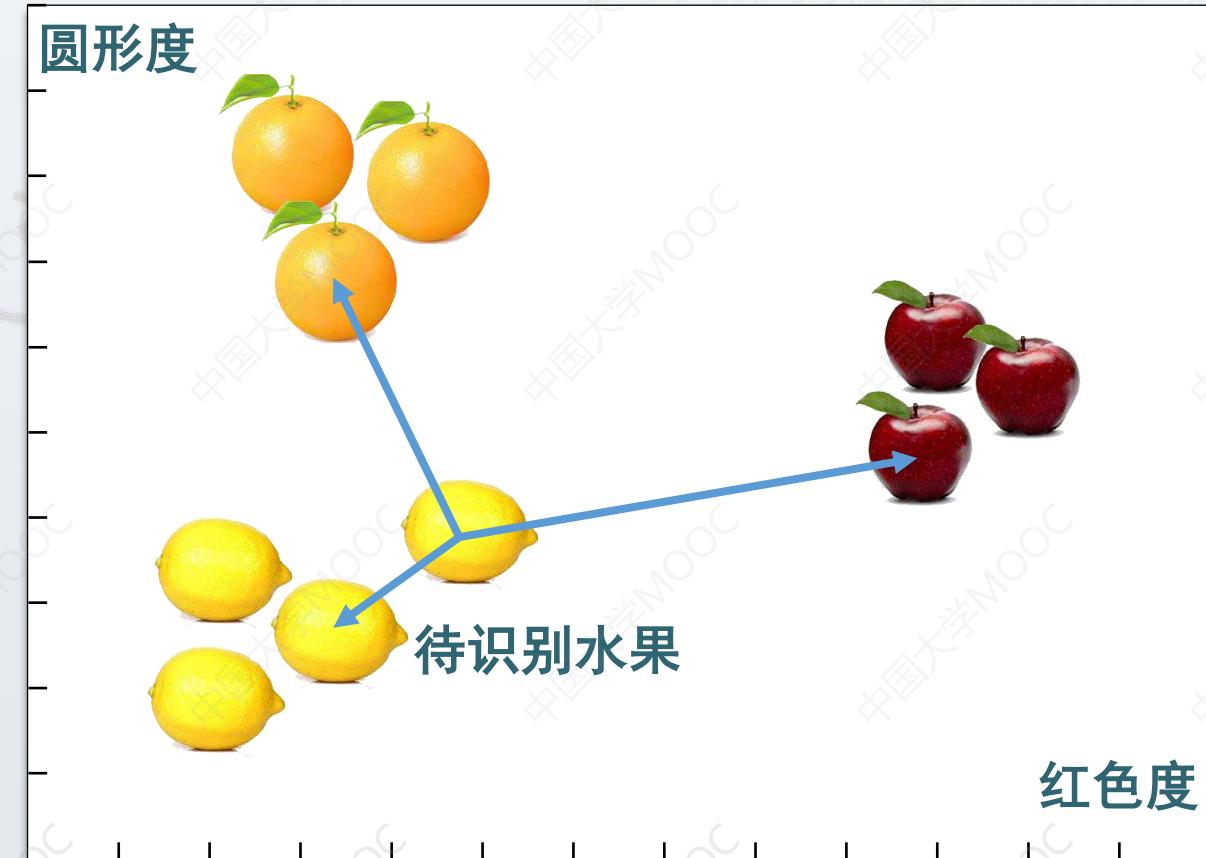
# 监督学习

当一个待识别的样本输入后，同样将圆形度及红色度两个特征量计算得到，并映射到特征空间。



# 最近邻算法

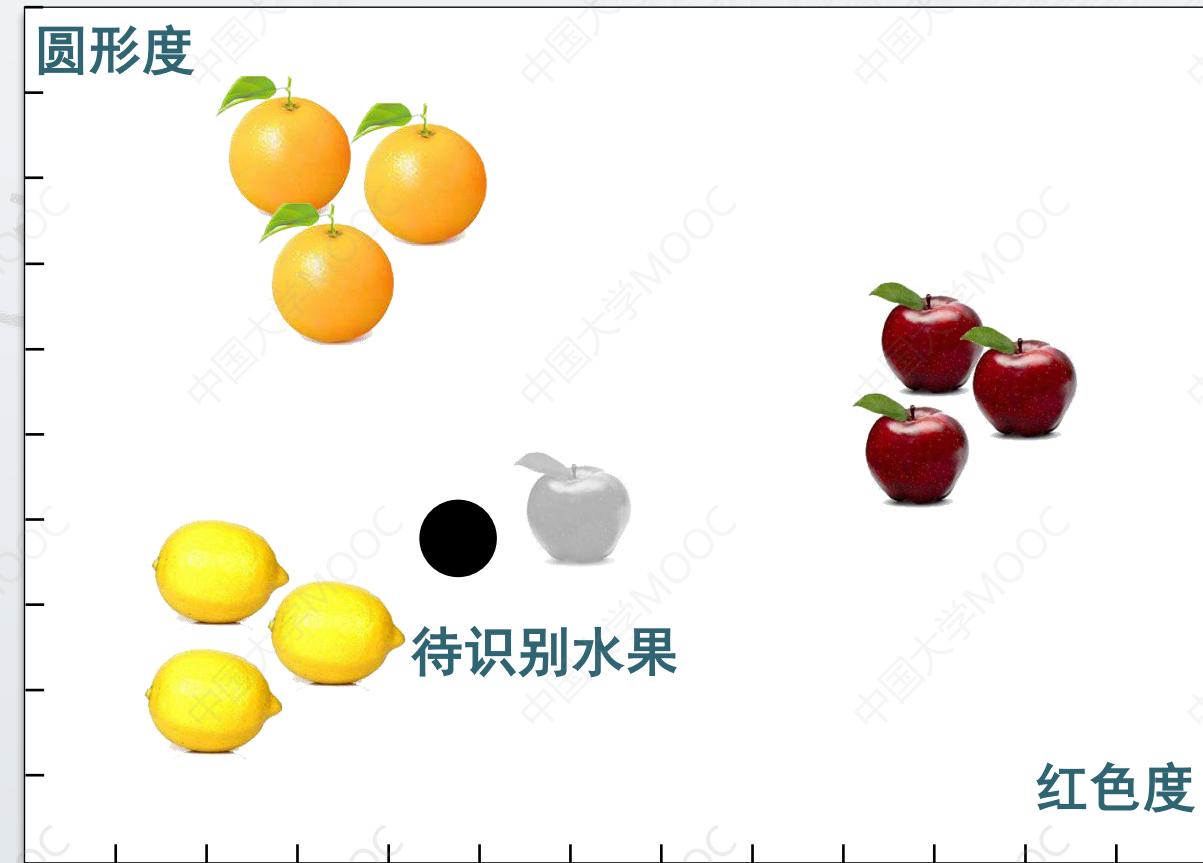
最近邻算法 (Nearest Neighbor, NN) : 特征空间中与待测样本最近的训练样本决定了待测样本的类别。



# 最近邻算法

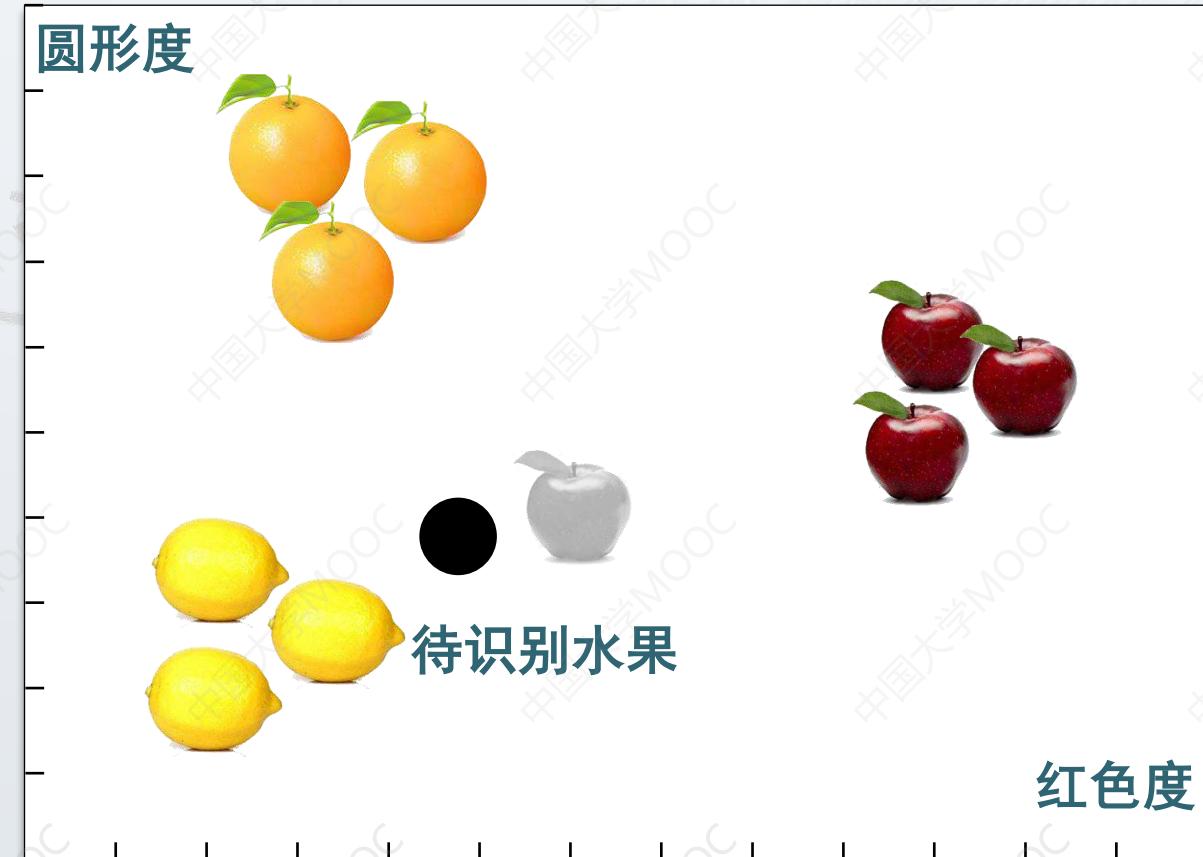
最近邻算法 (Nearest Neighbor, NN) : 特征空间中与待测样本最近的训练样本决定了待测样本的类别。

问题：噪声



# K-最近邻算法

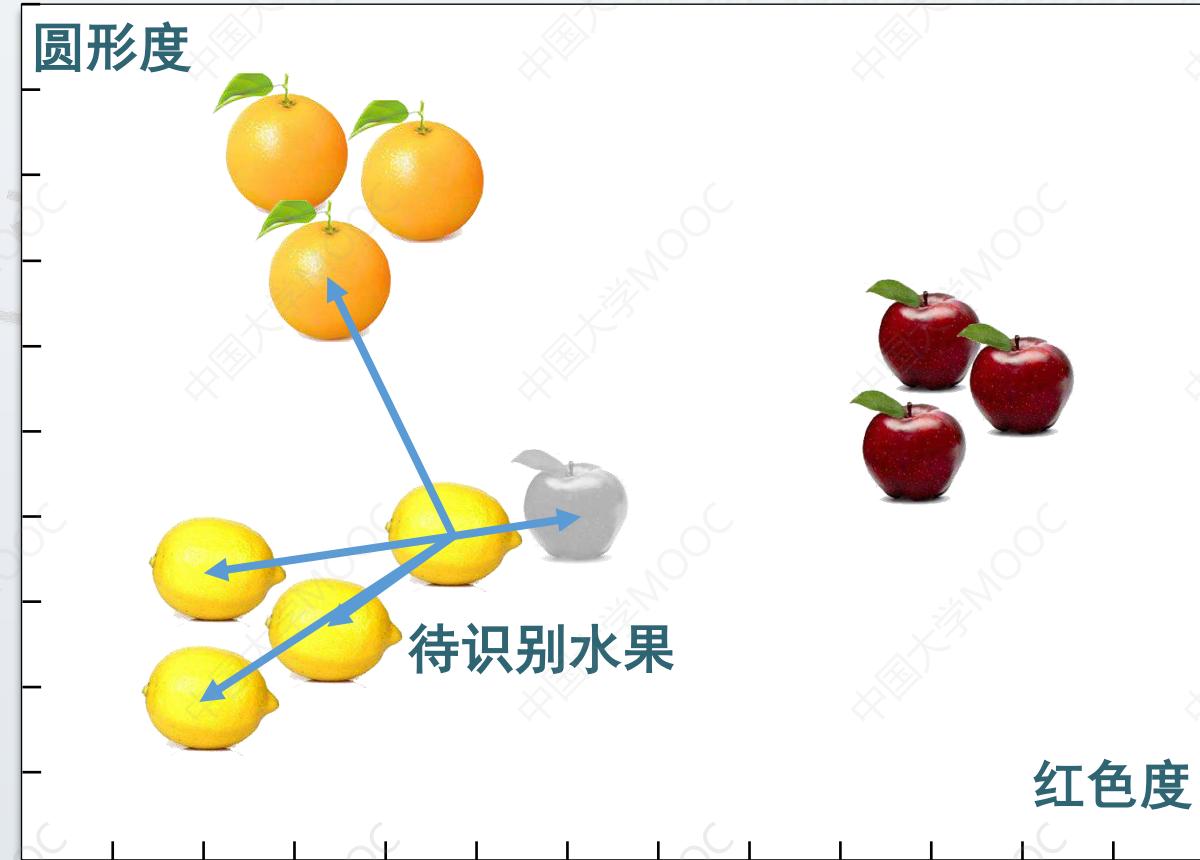
K最近邻(k-NearestNeighbor, kNN): 在特征空间中的k个最相邻的样本中的大多数属于某一个类别，则该样本也属于这个类别。



# K-最近邻算法

K最近邻(k-NearestNeighbor, kNN): 在特征空间中的k个最相邻的样本中的大多数属于某一个类别，则该样本也属于这个类别。

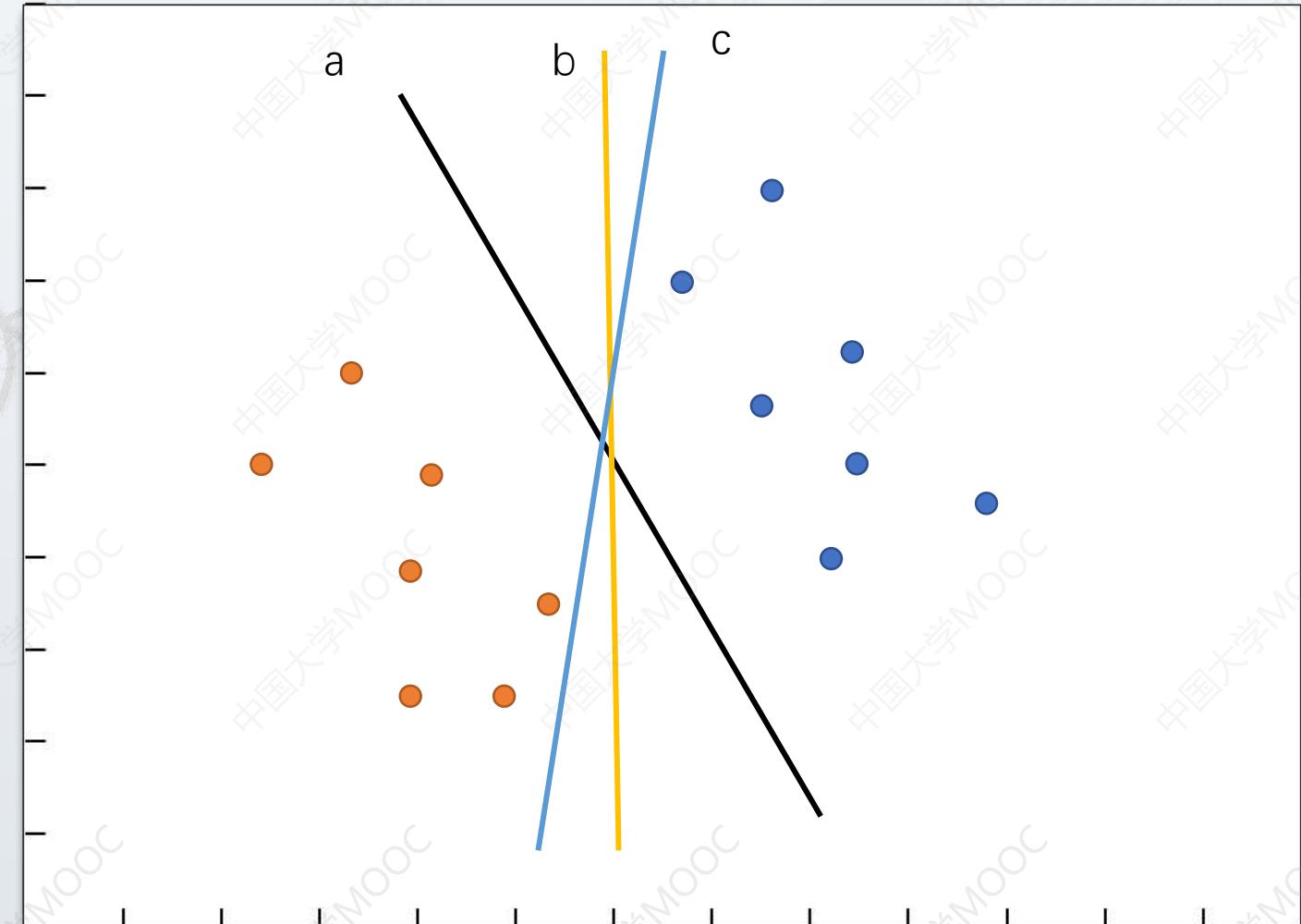
如k=5时



# 支持向量机

支持向量机 (Support Vector Machine, SVM)：监督学习的一种，是一个典型的二分类模型。

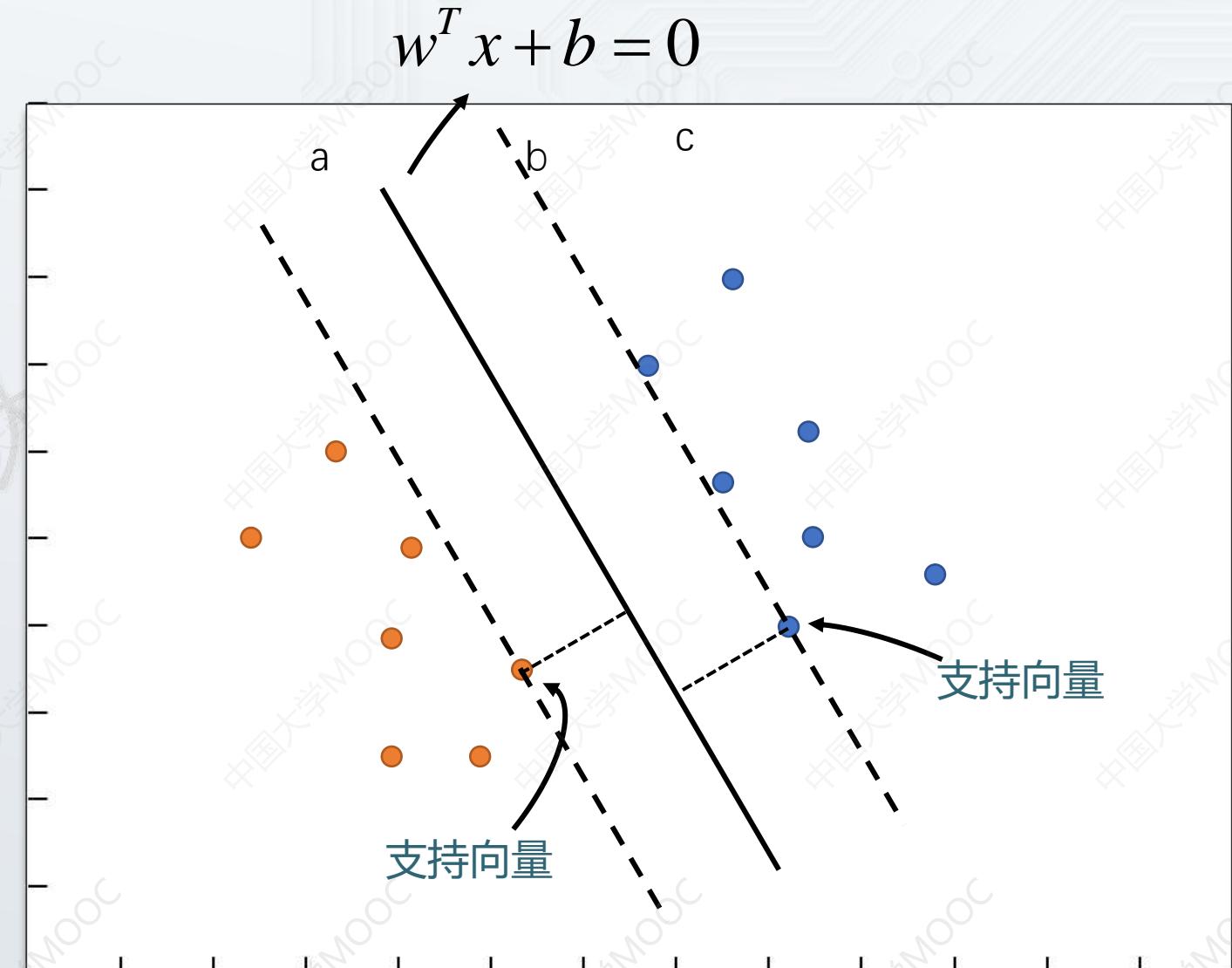
假设我们有正负两种样本，通过一个超平面将特征空间分开，如何选择超平面？



# 支持向量机

距离超平面最小的样本，我们称之为支持向量，我们所要找的最优超平面，就是使支持向量到超平面距离最大，我们认为，这样的超平面，就是最优的

SVM的核心问题就是找到这一超平面



# 支持向量机

空间样本中的任意一点 $x$ , 到超平面 $(w,b)$ 的距离可表示为:

$$\frac{|w^T x + b|}{\|w\|} = 0$$

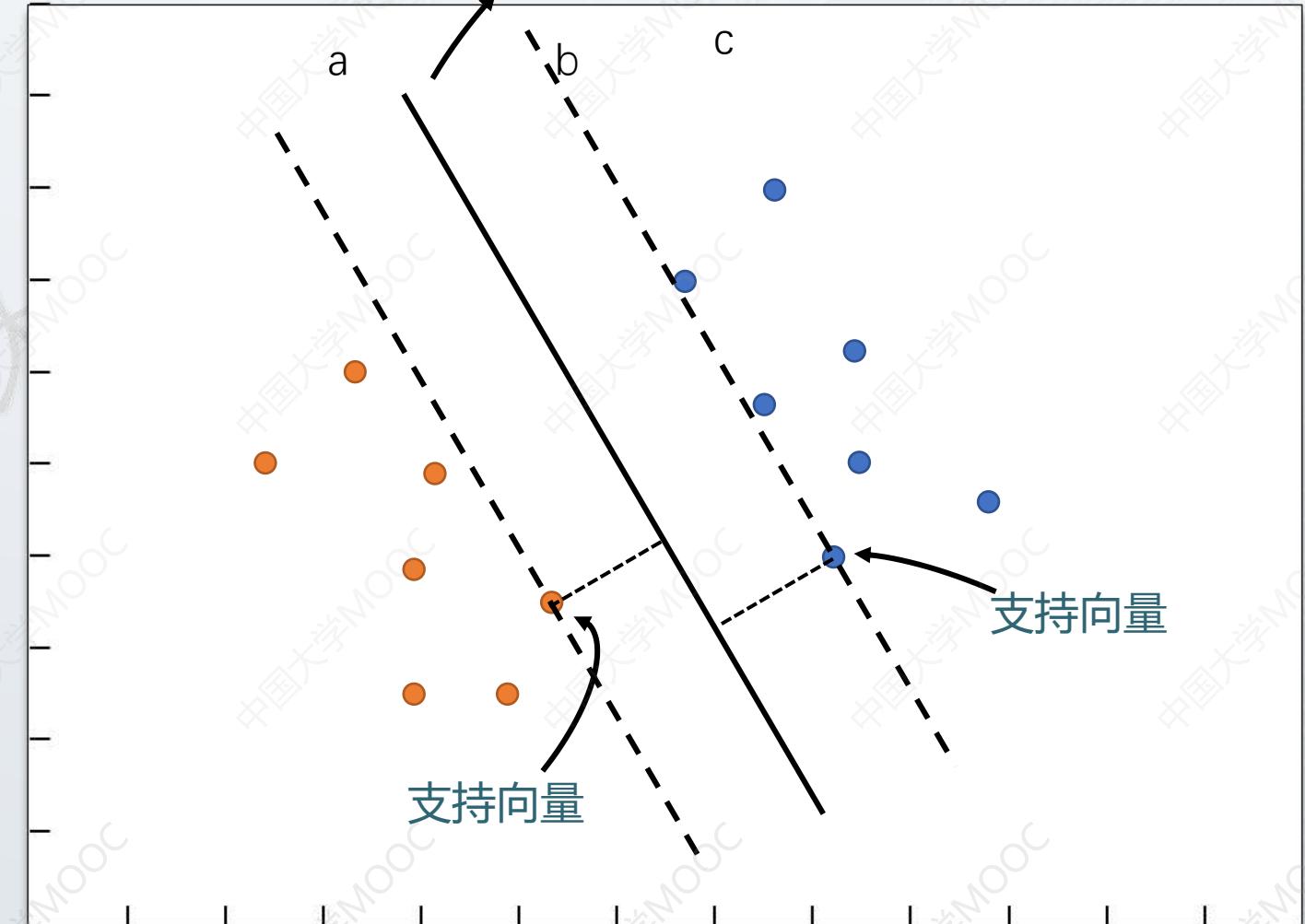
目标函数:

$$\arg \max_{w,b} \left\{ \min_n \left( \text{label} \cdot \frac{|w^T x + b|}{\|w\|} \right) \right\} = 0$$

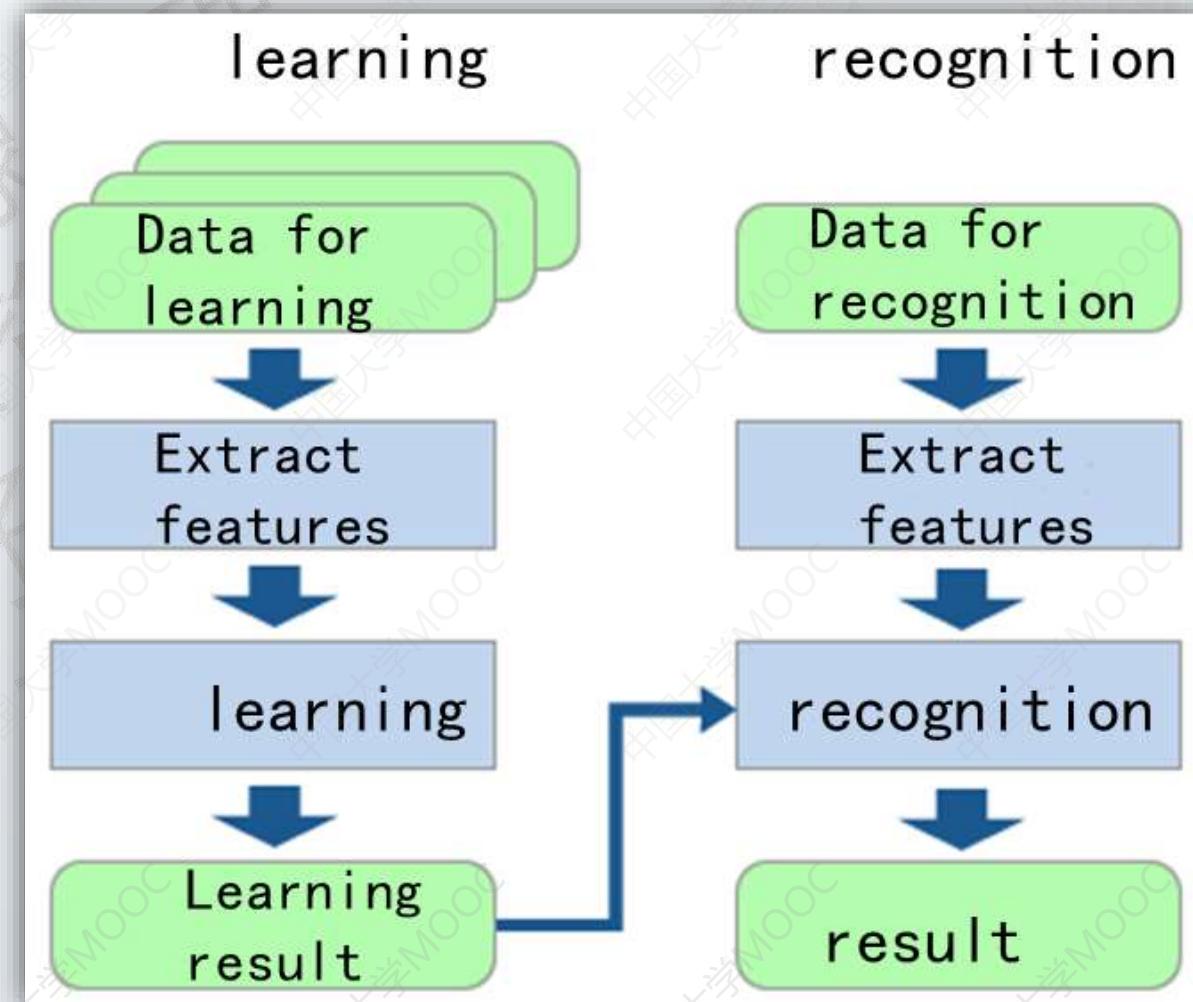
min: n个样本中距离超平面最近的点。

距离支持向量最远

$$w^T x + b = 0$$



# 识别过程



# 无监督学习

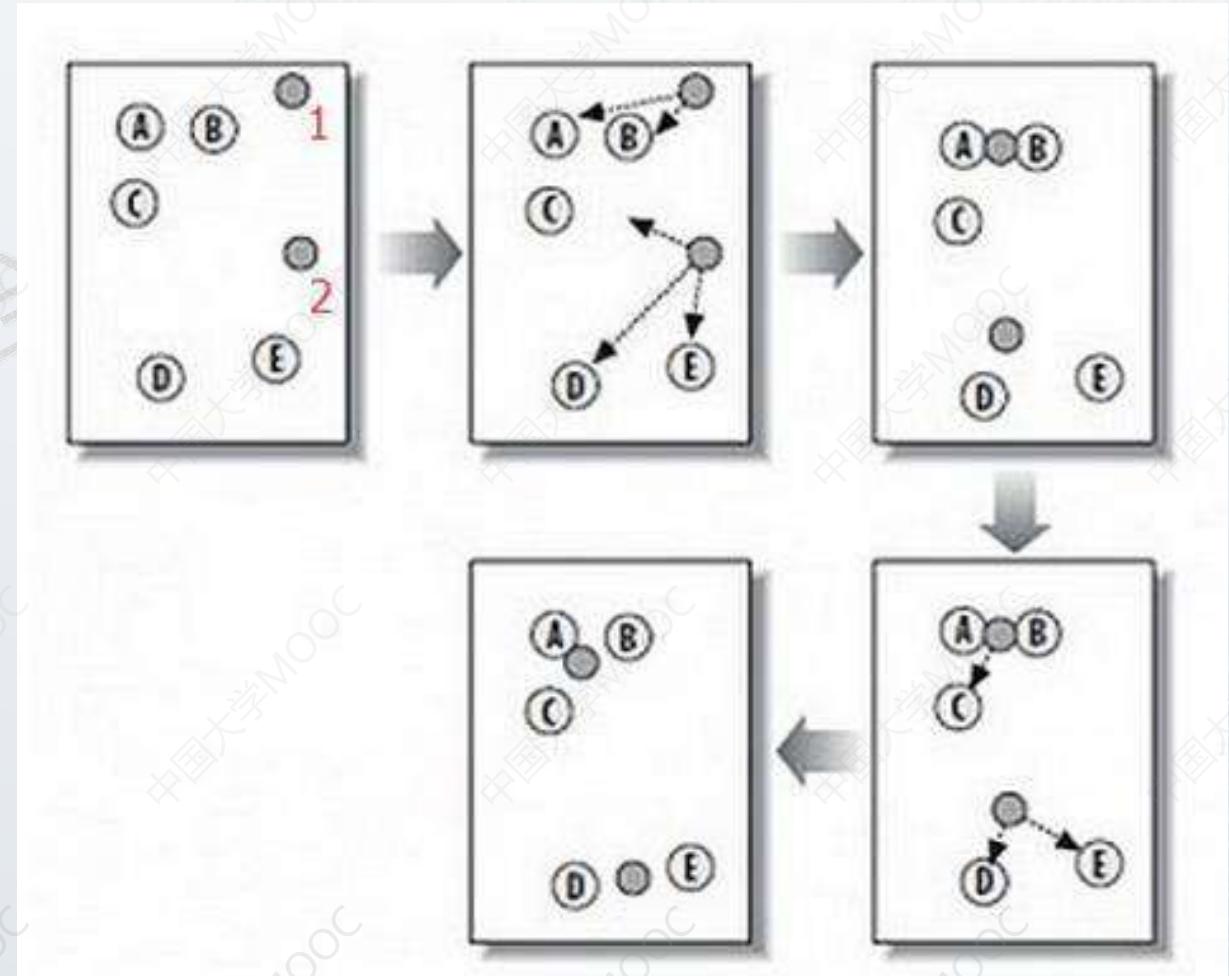
在无监督学习中，数据并不被特别标注，学习模型是为了推断出数据的一些内在结构。

聚类算法为典型的无监督学习。

# K-means

k均值聚类算法 (k-means clustering algorithm) 是一种迭代求解的聚类分析算法。

1. 随机选取k个初始质心；
2. 对每个样本点，计算得到距其最近的质心，将其类别标为该质心所对应的类；
3. 重新计算k个cluster对应的质心；
4. 重复step 2, 3直到质心不再发生变化

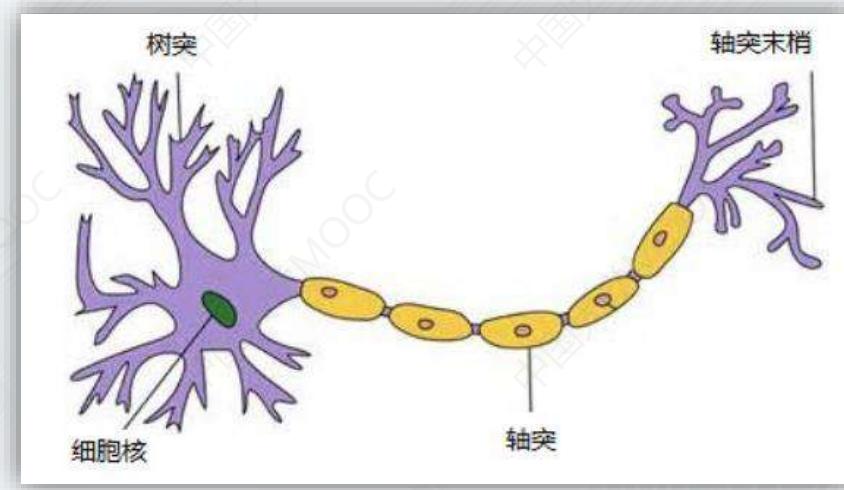


# 神经网络

一个神经元通常具有多个树突，主要用来接受传入信息；一条轴突，轴突尾端有许多轴突末梢可以给其他多个神经元传递信息。轴突末梢跟其他神经元的树突产生连接，从而传递信号；

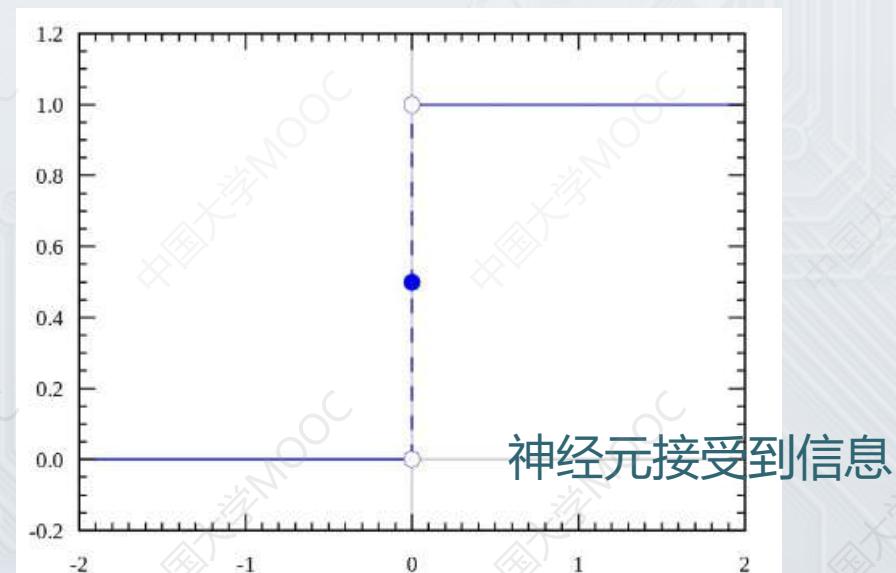
当一个神经元细胞接收到的生物的电的综合超过一个阈值时，细胞就会输出结果，做出判断。

低于阈值，输出0

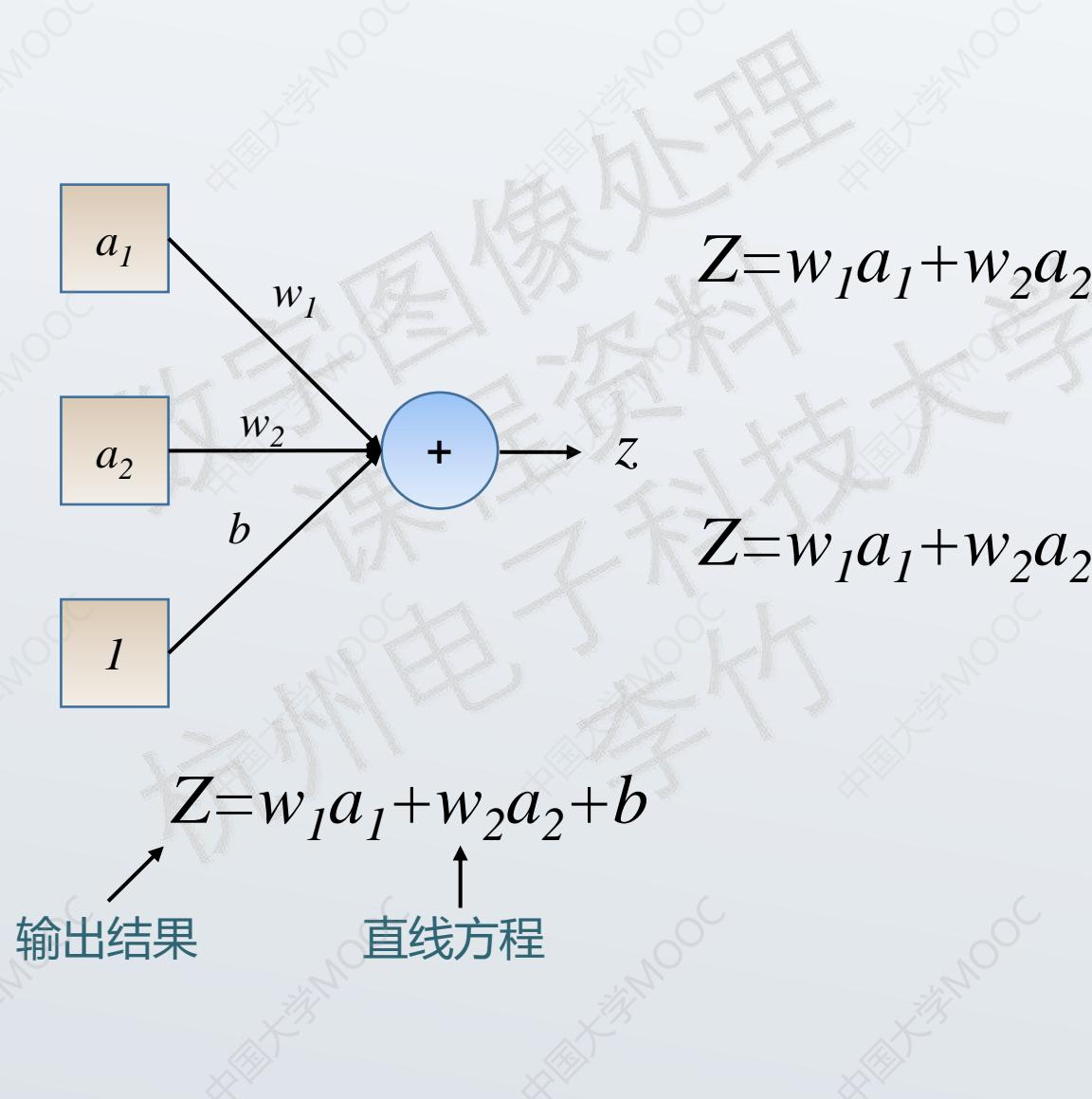


神经元

超过阈值，输出1

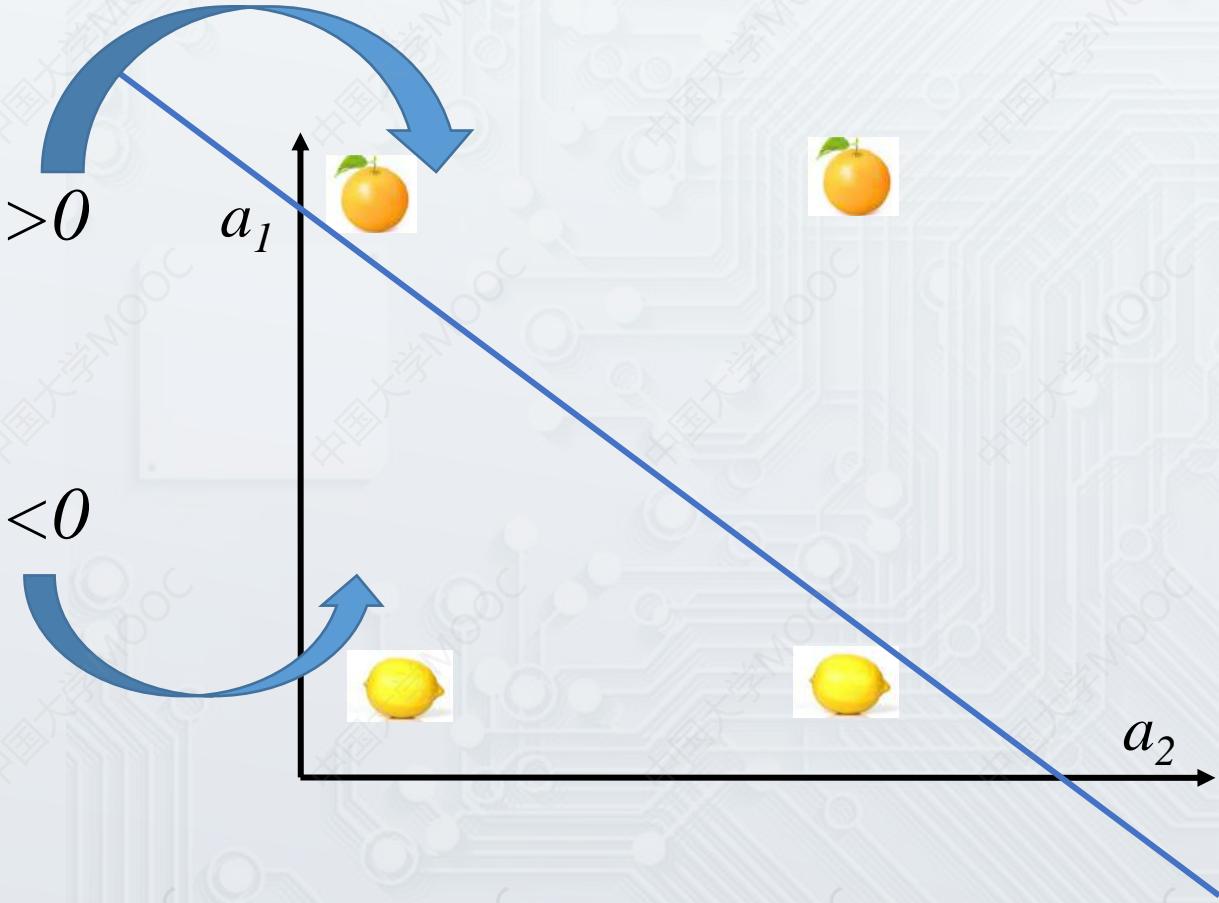


# 感知器



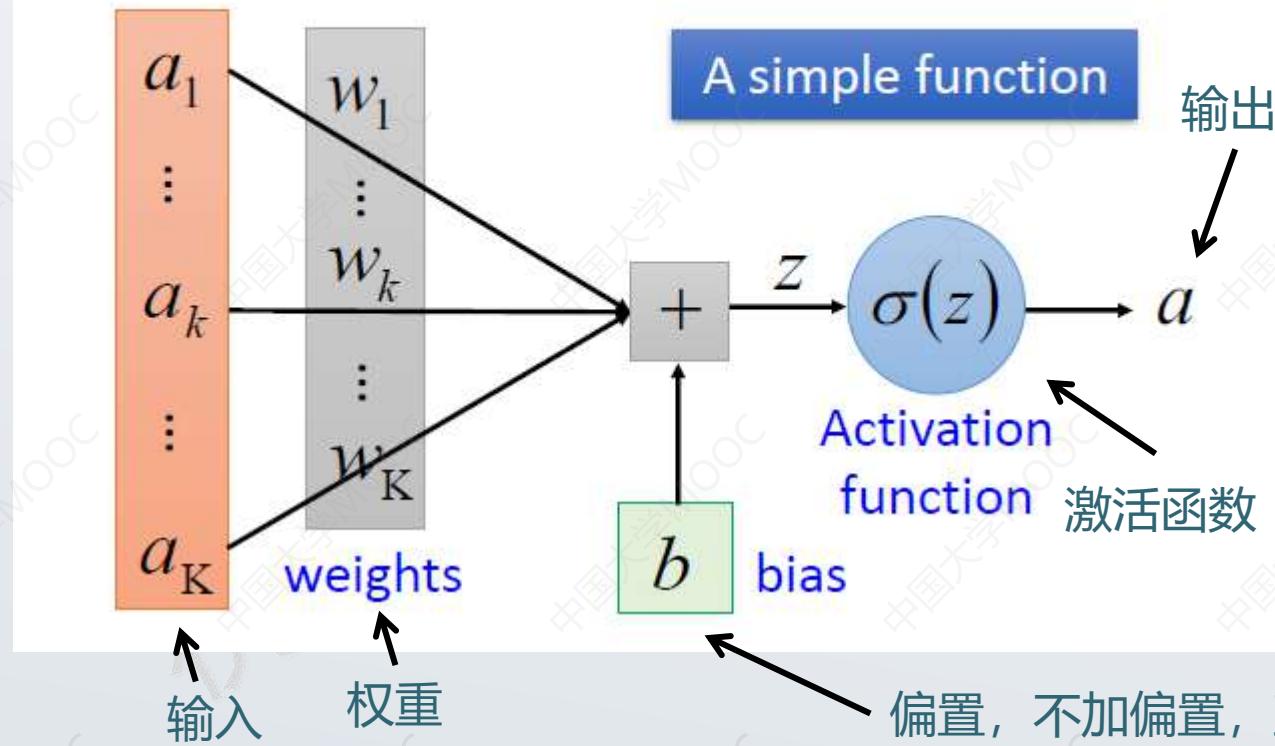
$$Z = w_1a_1 + w_2a_2 + b > 0$$

$$Z = w_1a_1 + w_2a_2 + b < 0$$



# 感知器

$$z = a_1 w_1 + \cdots + a_k w_k + \cdots + a_K w_K + b$$

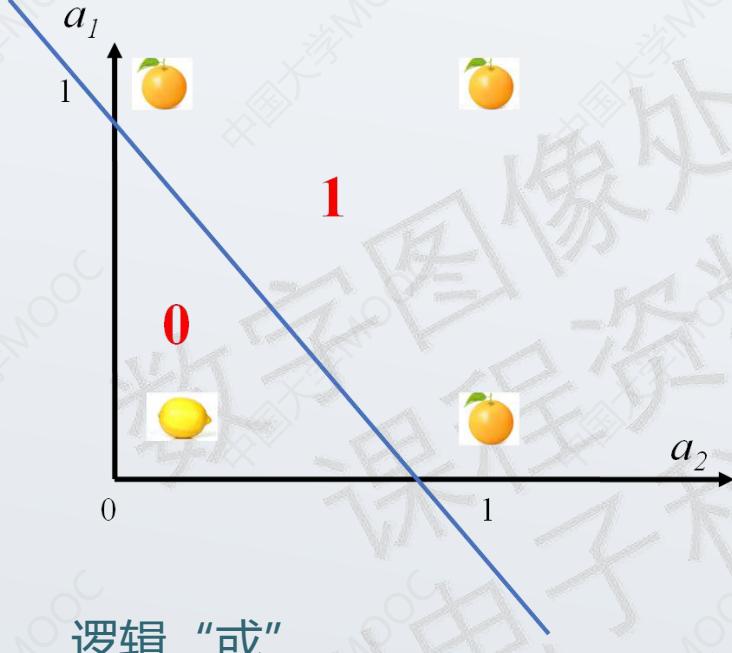


每个输入乘以权重相加，然后加上偏置b，乘以激活函数，得到输出。

Z的表达式为超平面方程

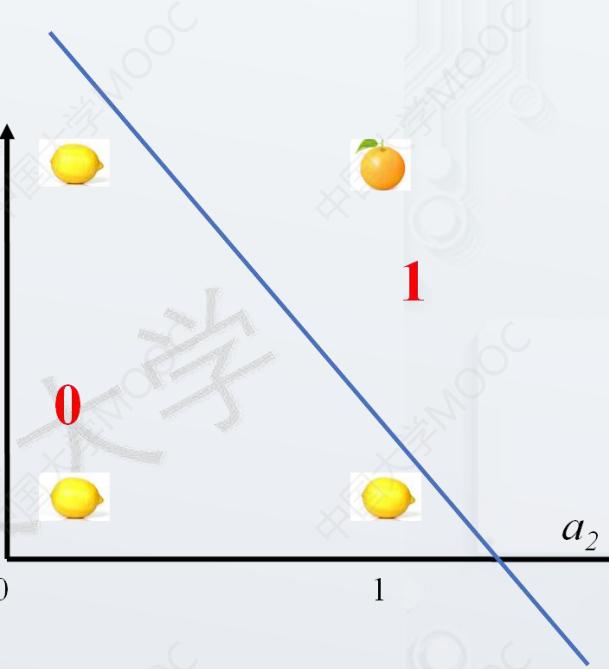
$$z = a_1 w_1 + \cdots + a_k w_k + \cdots + a_K w_K + b$$

# 感知器



逻辑“或”

$a_1$	$a_2$	$z$
0	0	0
0	1	1
1	0	1
1	1	1



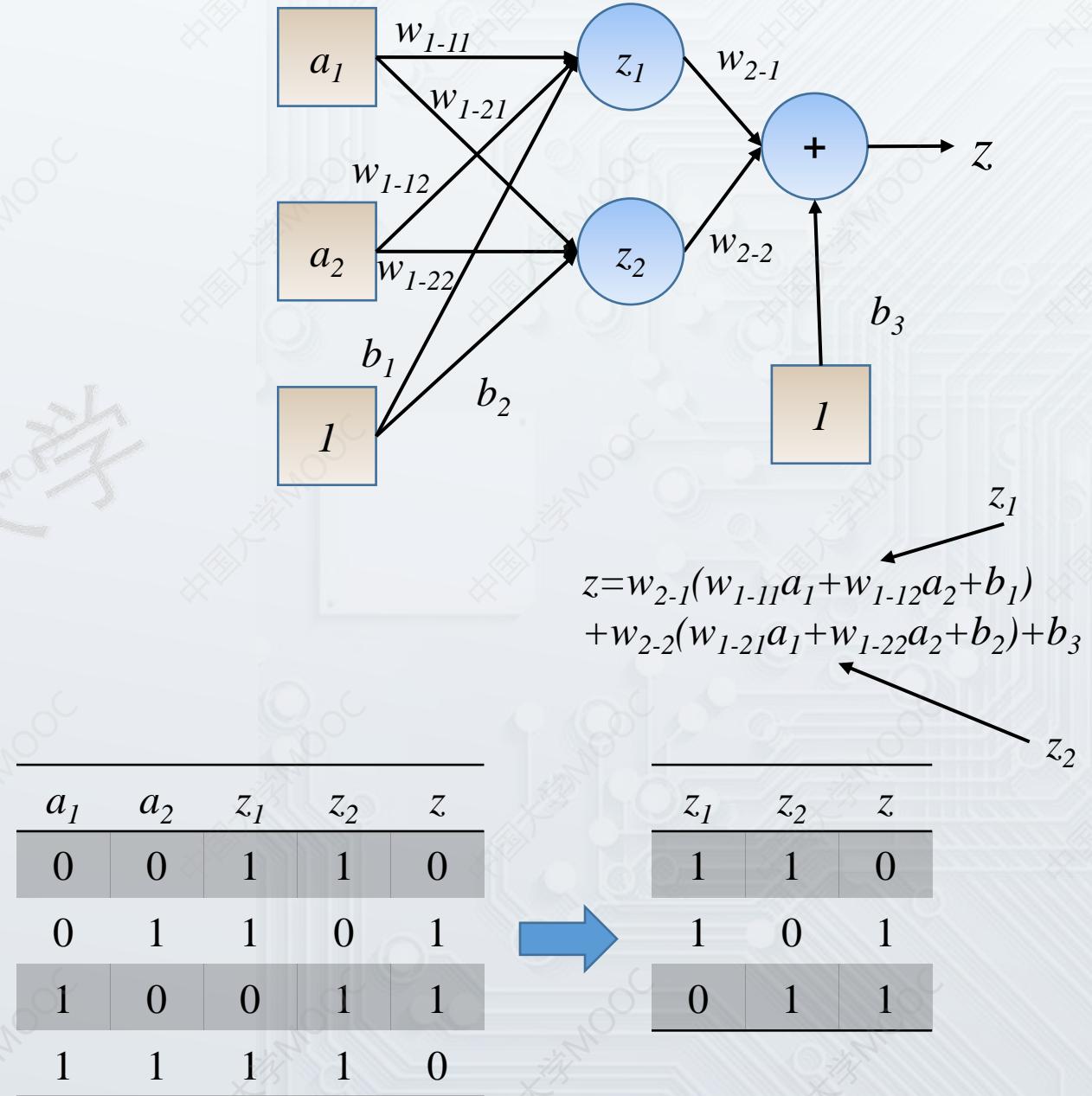
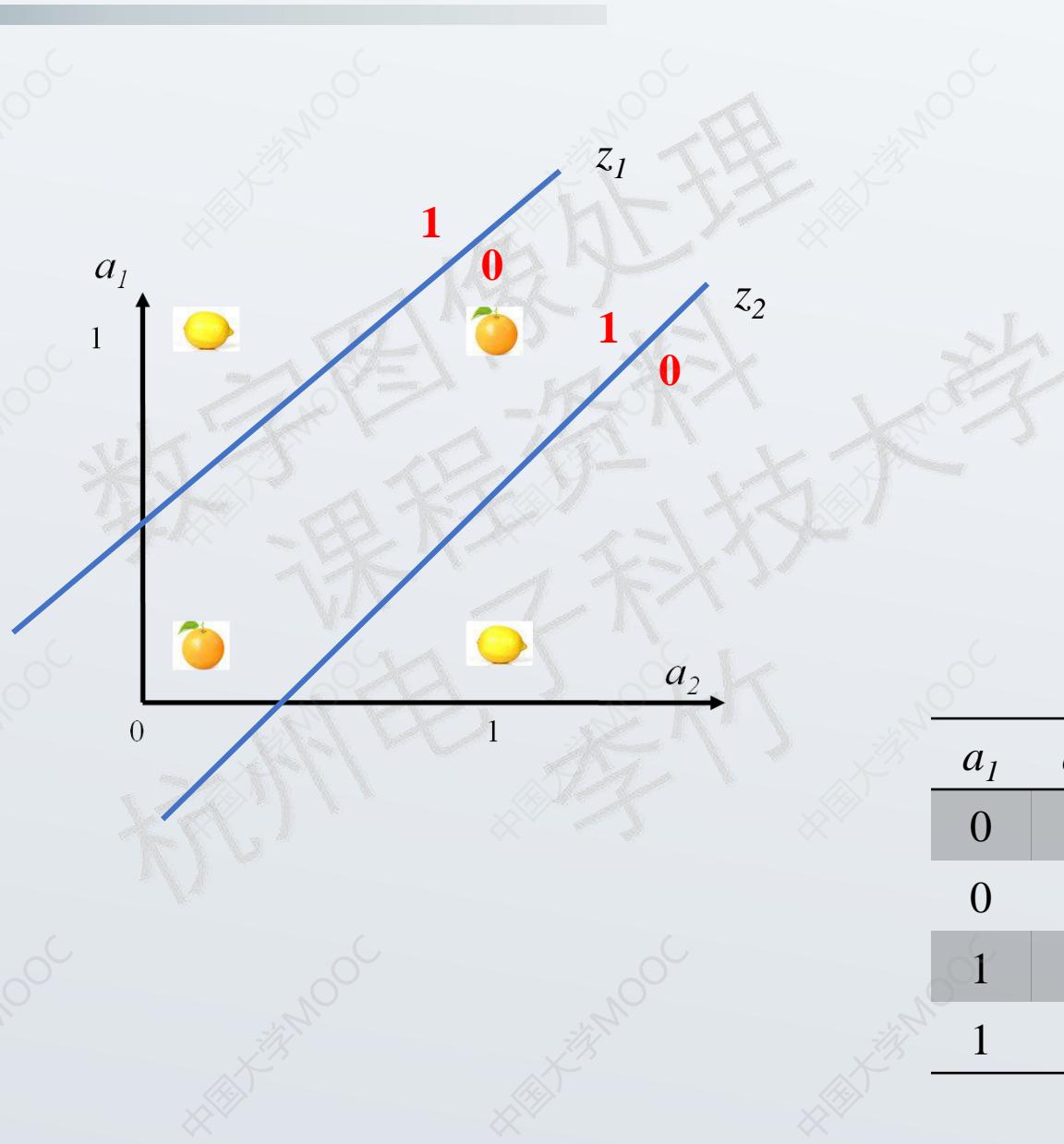
逻辑“与”

$a_1$	$a_2$	$z$
0	0	0
0	1	0
1	0	0
1	1	1



逻辑“异或”，线性不可分。

# 逻辑形式

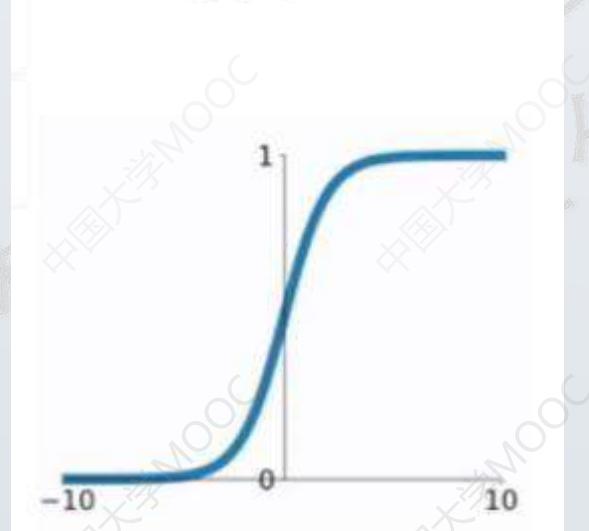


# 激活函数

激活函数给神经元引入了非线性因素，使得神经网络可以任意逼近任何非线性函数，这样神经网络就可以应用到众多的非线性模型中。

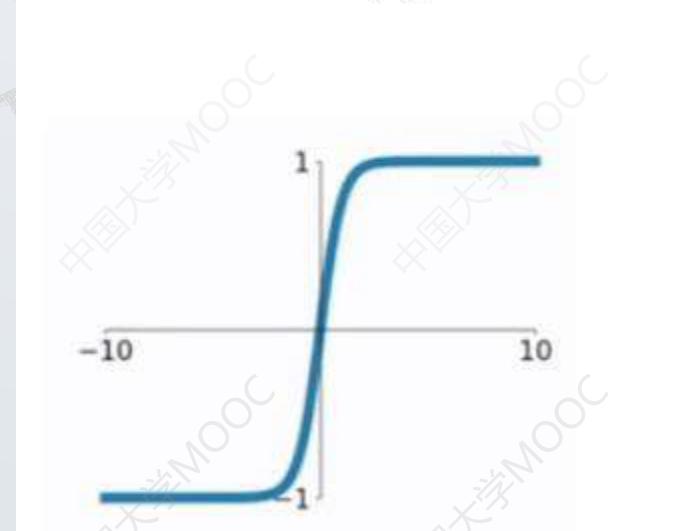
sigmoid函数

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$



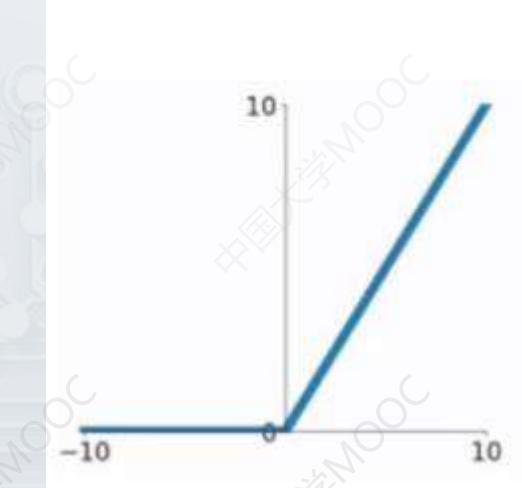
tanh函数

$$\tanh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

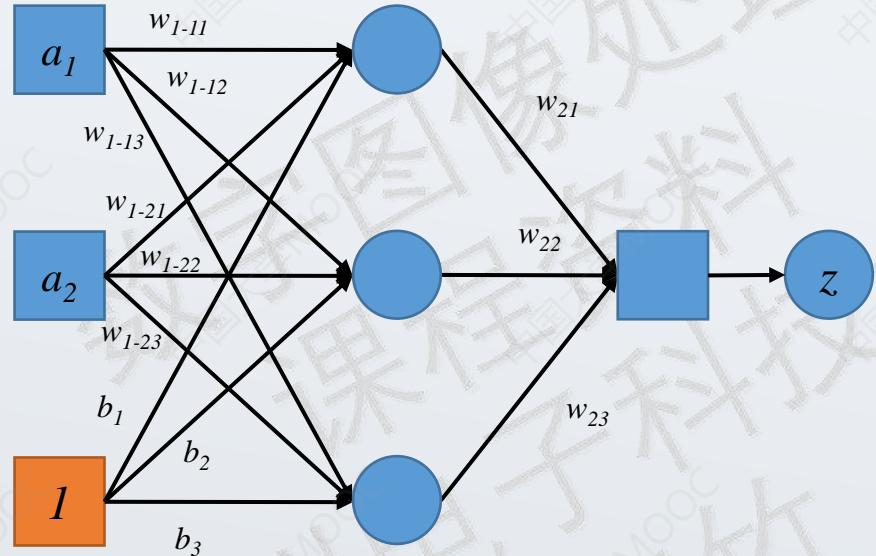


Relu函数

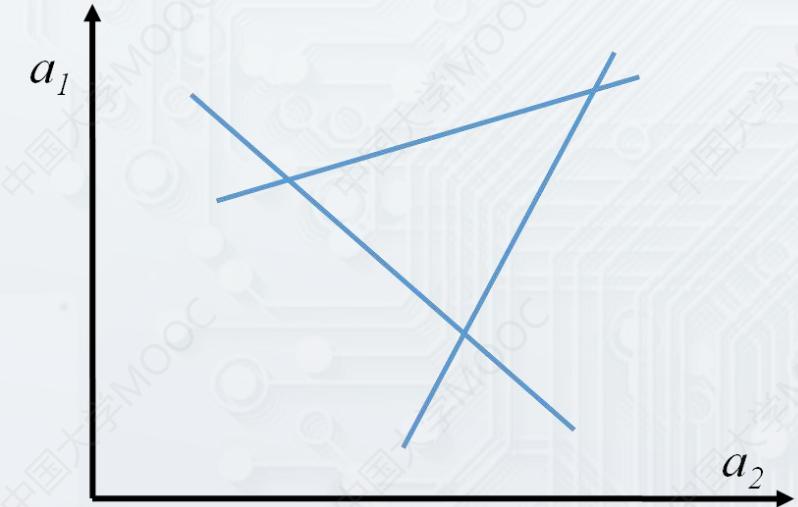
$$f(x) = \max(0, x)$$



# 激活函数的作用



无激活函数，线性  
叠加得到的分类器  
始终为线性的

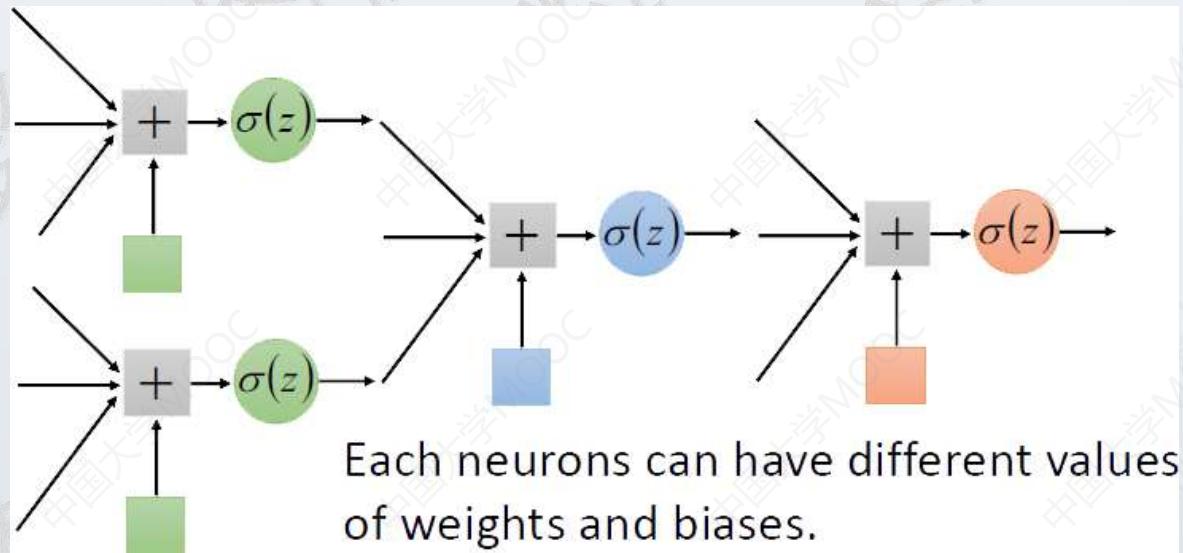


添加激活函数，  
可以得到非线性  
分类器

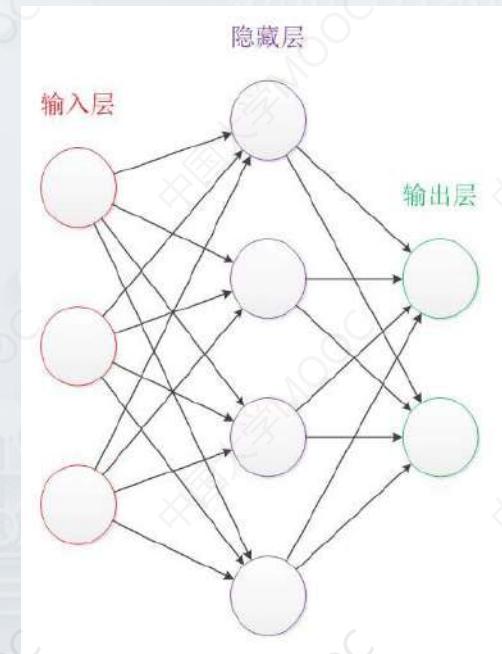


# 神经网络

如果我们增加神经元构成网络，即构成了神经网络，每个神经元可以有不同的权重值和偏置值

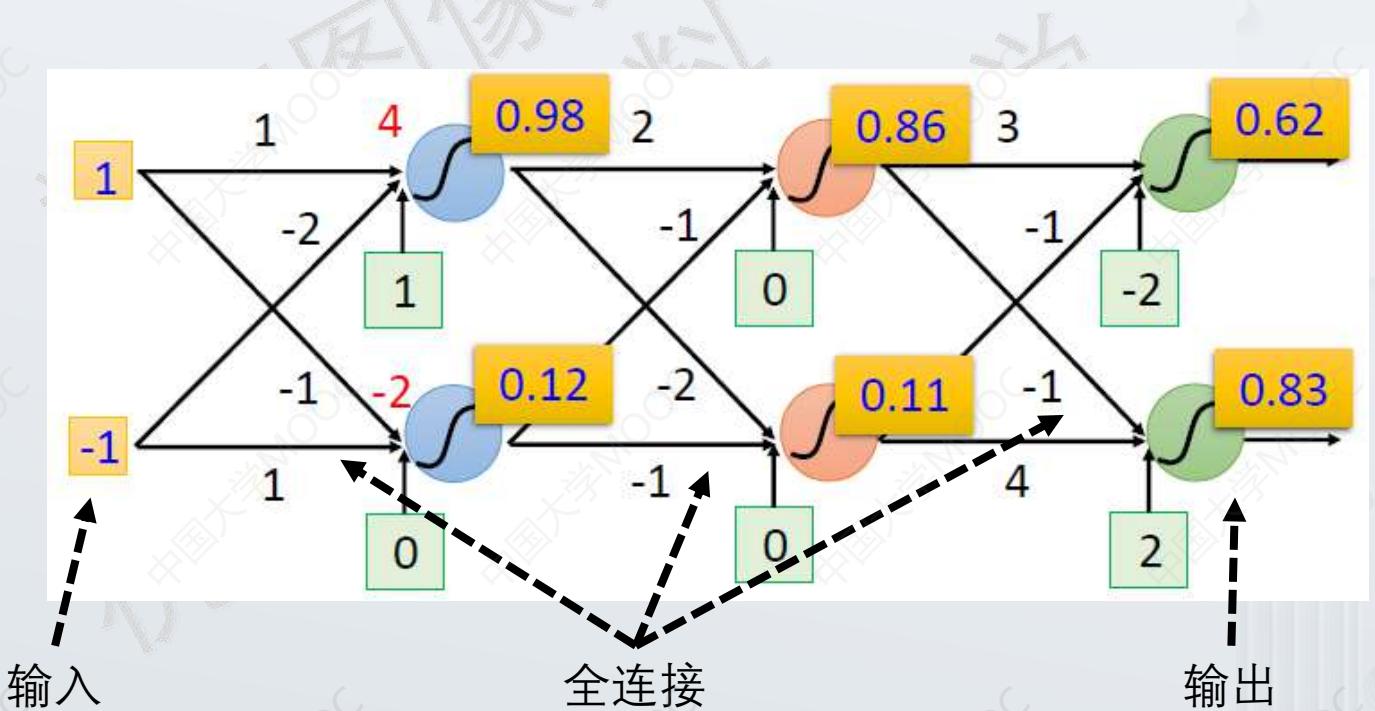


增加隐藏层的层数，即“深度”神经网络



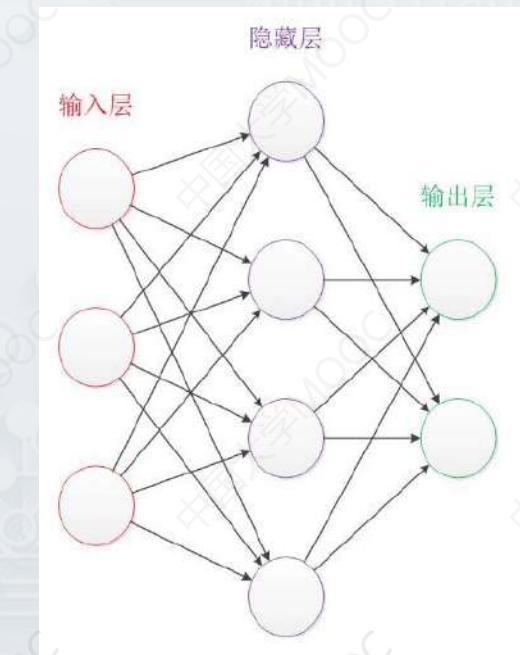
# 神经网络

如果我们增加神经元构成网络，即构成了神经网络，每个神经元可以有不同的权重值和偏置值

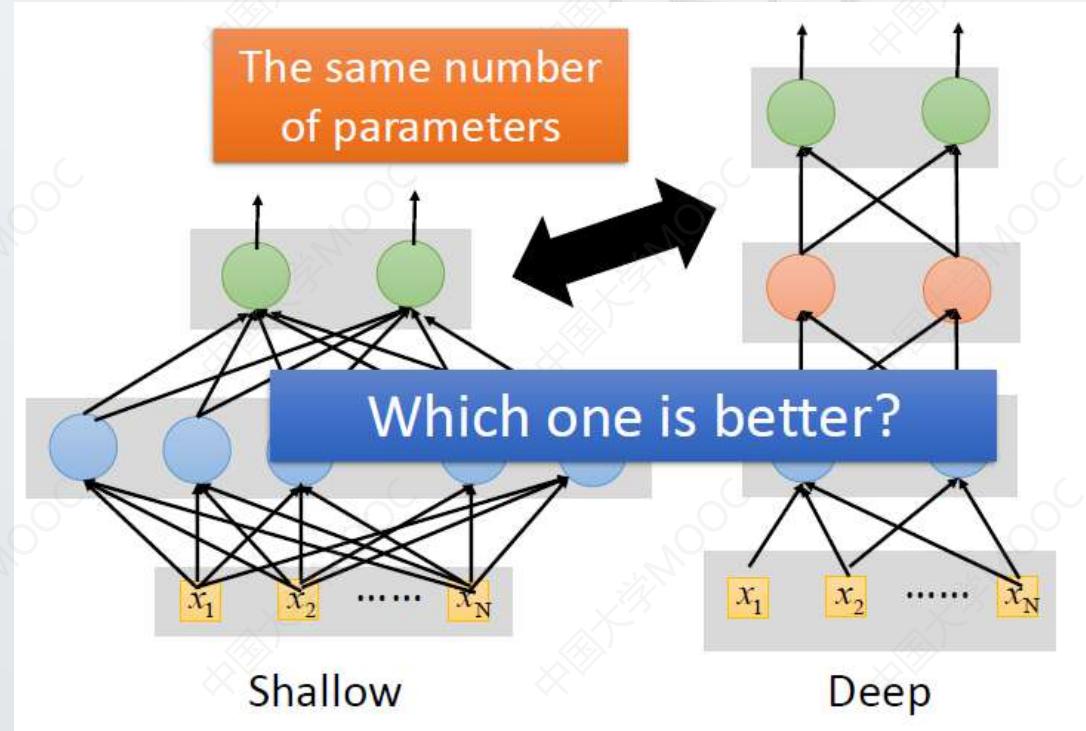


$$f \left( \begin{bmatrix} 1 \\ -1 \end{bmatrix} \right) = \begin{bmatrix} 0.62 \\ 0.83 \end{bmatrix}$$

增加隐藏层的层数，即“深度”神经网络



# 神经网络



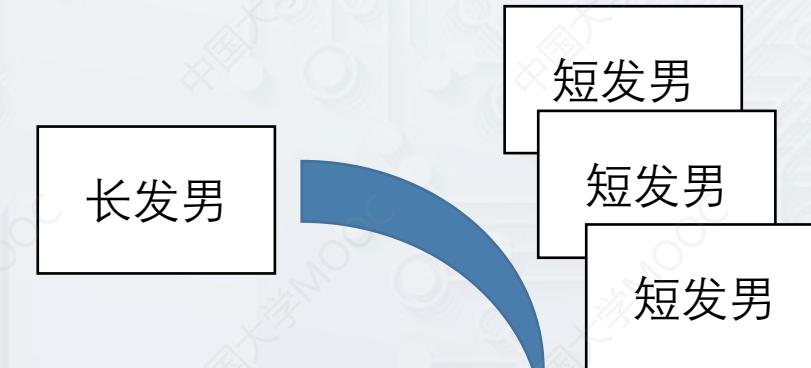
实际上，只要添加足够多的神经元，只需要一层隐藏层即可拟合出任何函数。本质上和前述的boosting类似。

但是深层网络比扁平网络更有优势，实质上进行了进步的特征映射，详细后述。

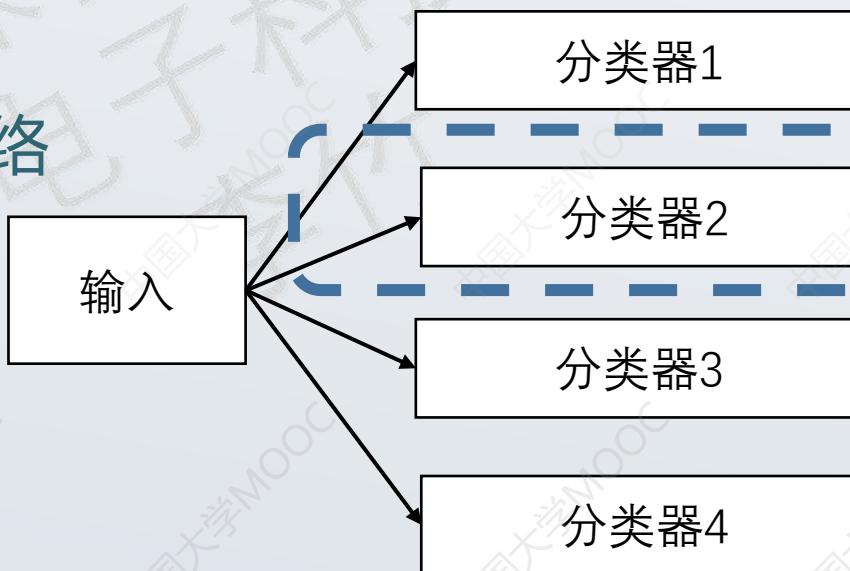
K Hornik, M Stinchcombe, H White, Multilayer feedforward networks are universal approximators, 《Neural Networks》, 1989, 2 (5):359-366

# 神经网络

- 一个4分类的数据集



- 扁平化网络

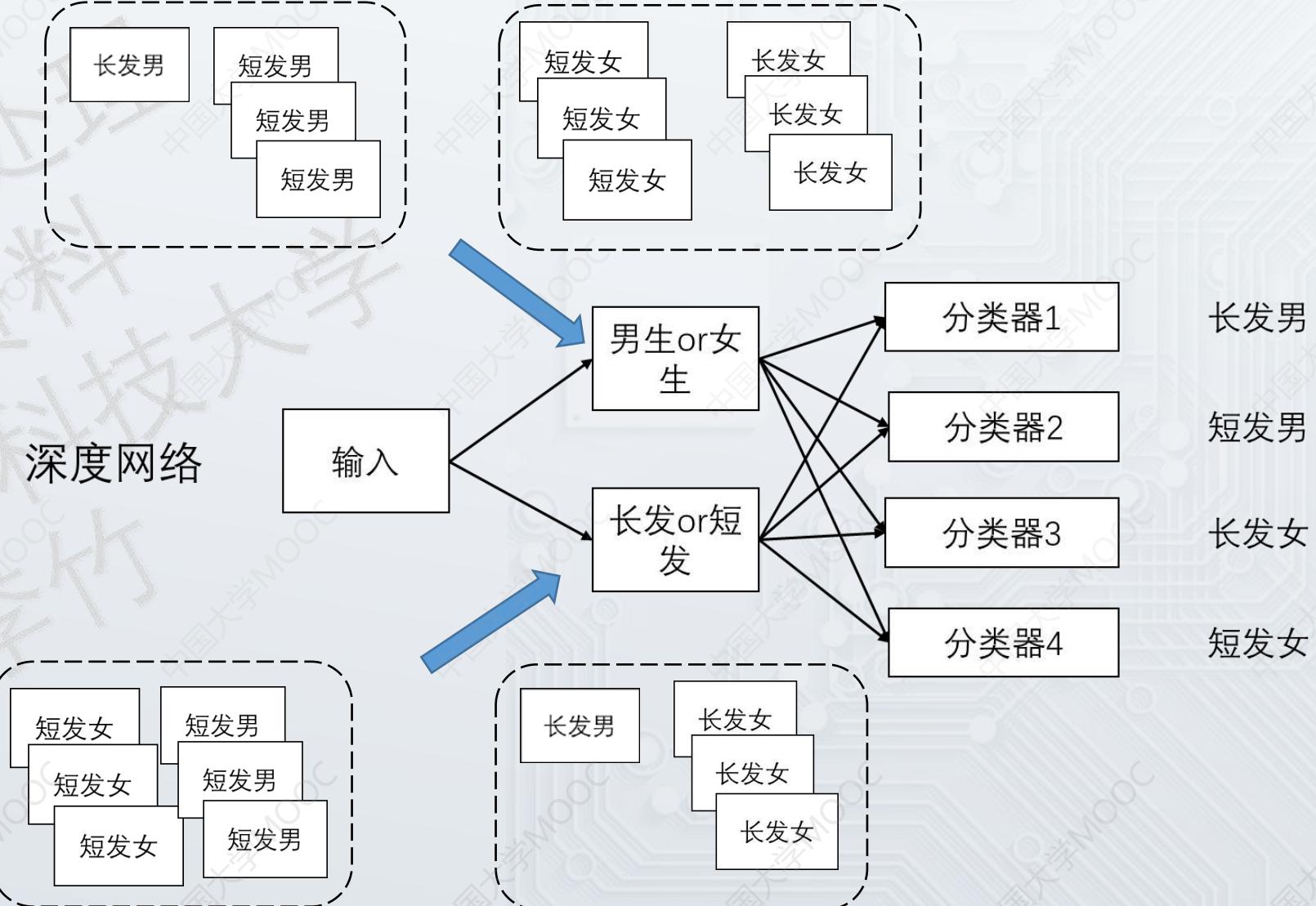


长发男  
短发男  
长发女  
短发女

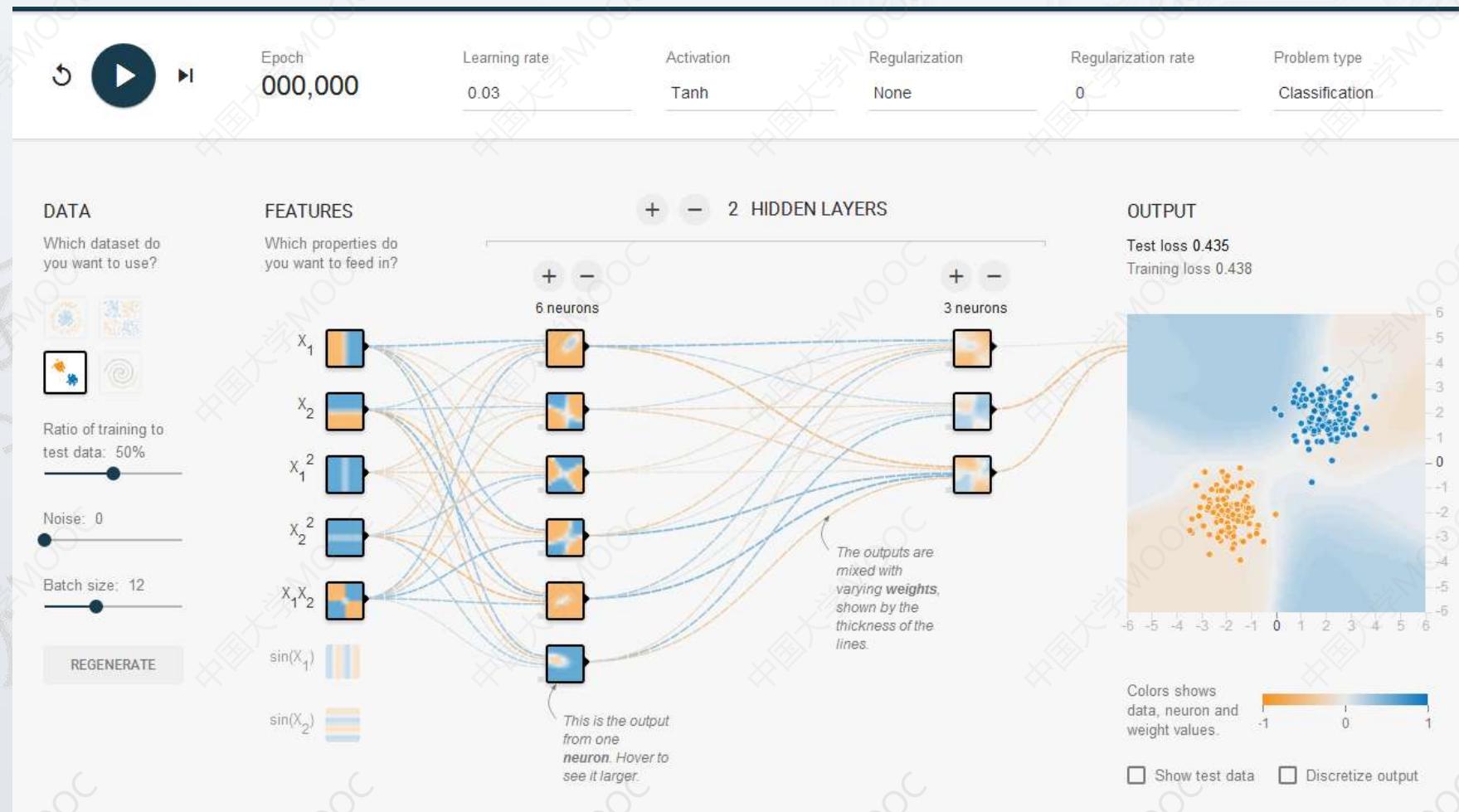
- 数据量较少，识别能力很弱

# 神经网络

- 相当于用低等特征组合出高等特征



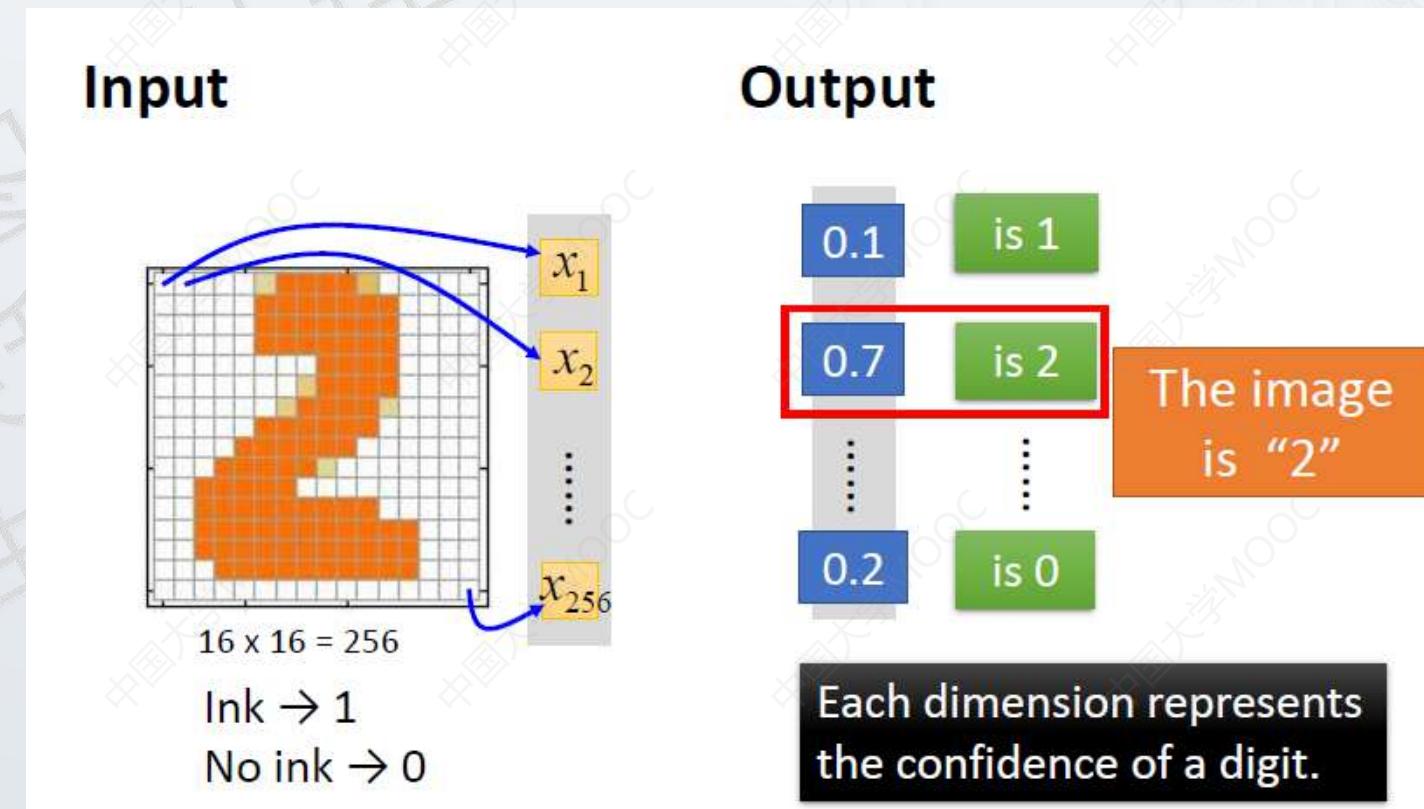
# 神经网络



<http://playground.tensorflow.org>

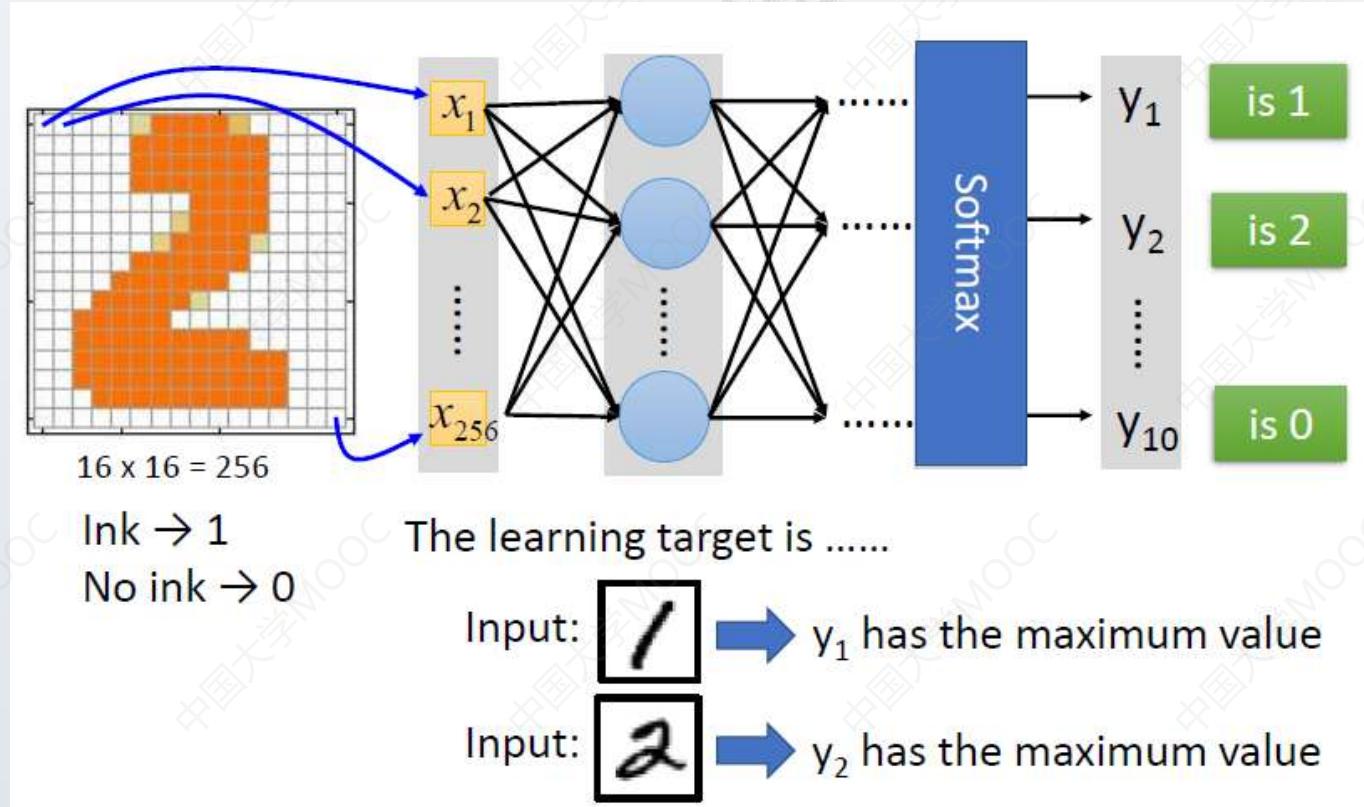
# 实例：数字识别

- 输出层，因为共有10个数字，所以输出层有10个节点，输出一个10维的向量。



- 输入的图像的size为 $16 \times 16$ ，即输入一个256维的向量，输入层有256个节点，

# 实例：数字识别



网络结构及内部的参数，权重，偏置，引入卷积处理后，还有卷积的系数

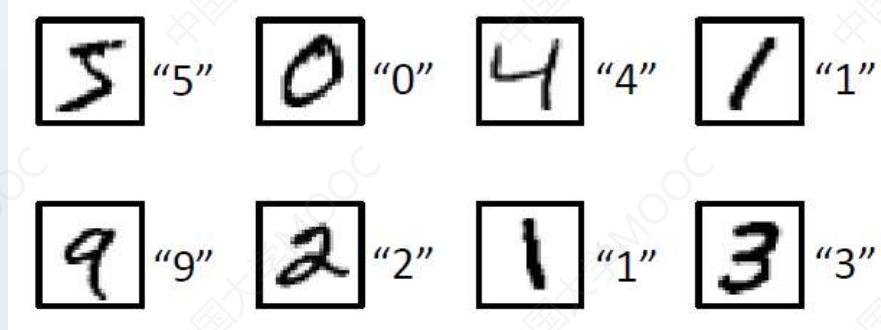
这个网络可以使：

输入任意形态的1的图像，在输出层, $y_1$ 的值最大。

输入任意形态的2的图像，在输出层, $y_2$ 的值最大。

# 实例：数字识别

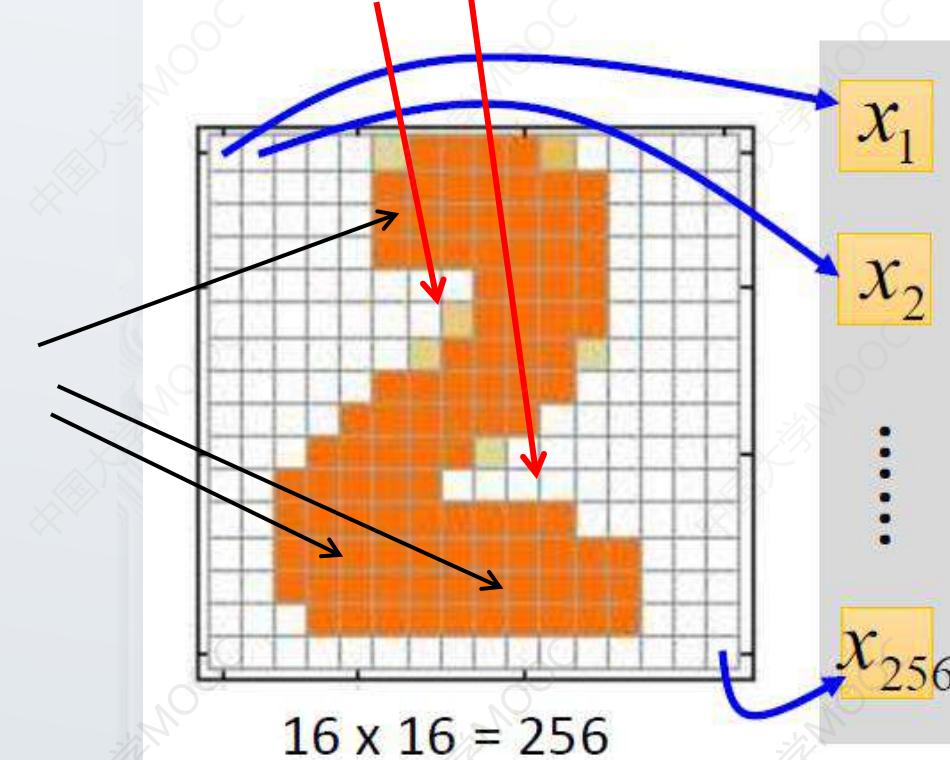
首先准备训练数据，并给训练数据加标签



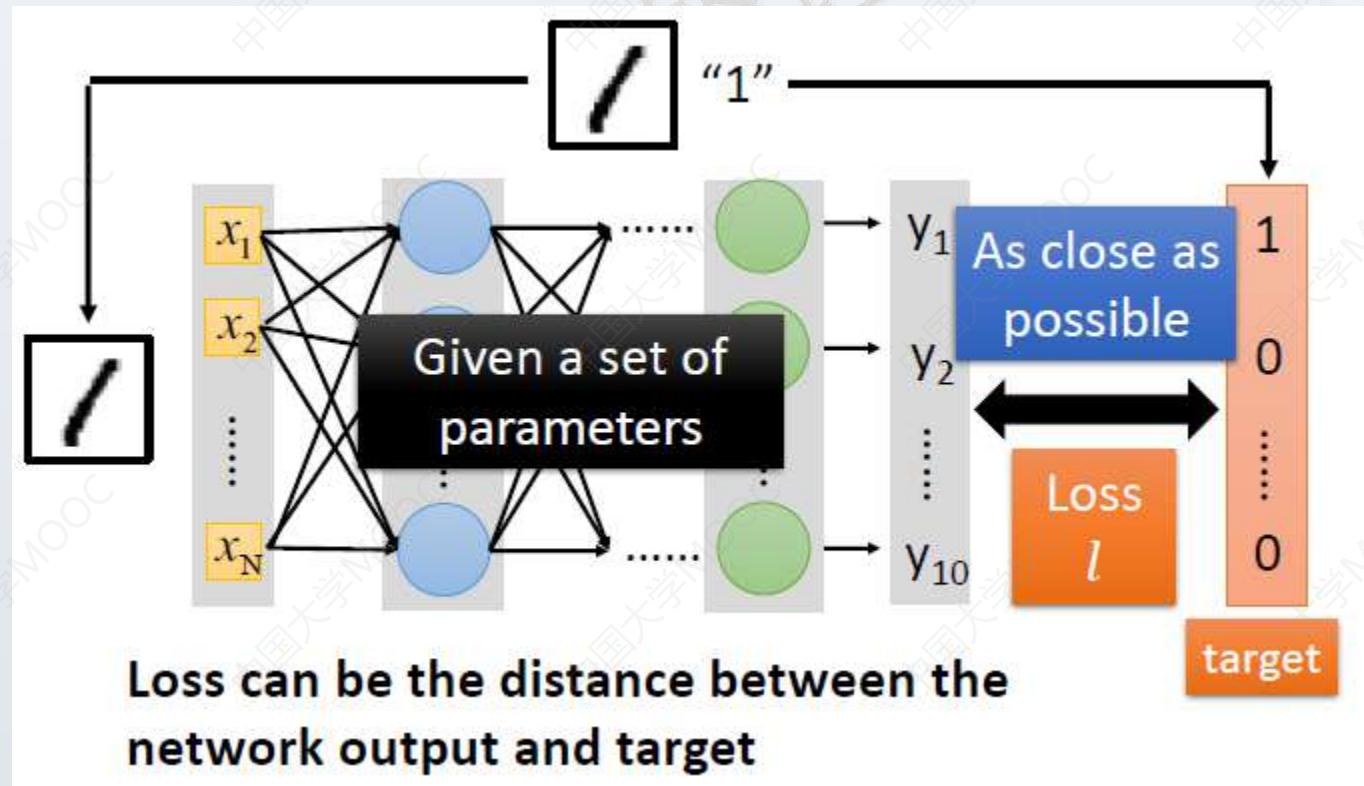
当图像为2时，这一类位置的像素通常为黑色，训练的结果会使这些像素对应的节点的权重增大。即这些像素为黑色时输出2的可能性增大

经过大样本的训练，一些节点的权重会显著变化

这些位置的像素为白色时，输出2的可能性较大，因此大量训练会使这里的权重负增加。



# 实例：数字识别



网络输出和真值的差距越小，则我们的参数越合适。这个差距即深度学习中所说的 Loss，损失。我们通过一个函数来计算 Loss 的大小，该函数即损失函数。

既然输出的是一个十维的向量，最简单的方法就是通过距离计算损失：

Square Error

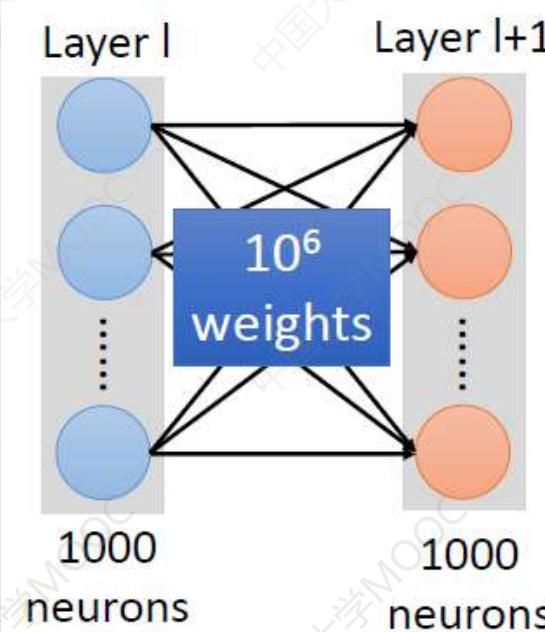
$$\sum_{i=1}^{10} (y_i - \hat{y}_i)^2 = 0$$

# 实例：数字识别

所以我们训练的目的是找到一组网络的参数配置，这组参数可以使得在所有的训练数据上，损失函数的值最小。

那么我们是否直接把所有的配置列出来，然后计算损失，再排序取最小值？？

需要海量的计算，无法实现

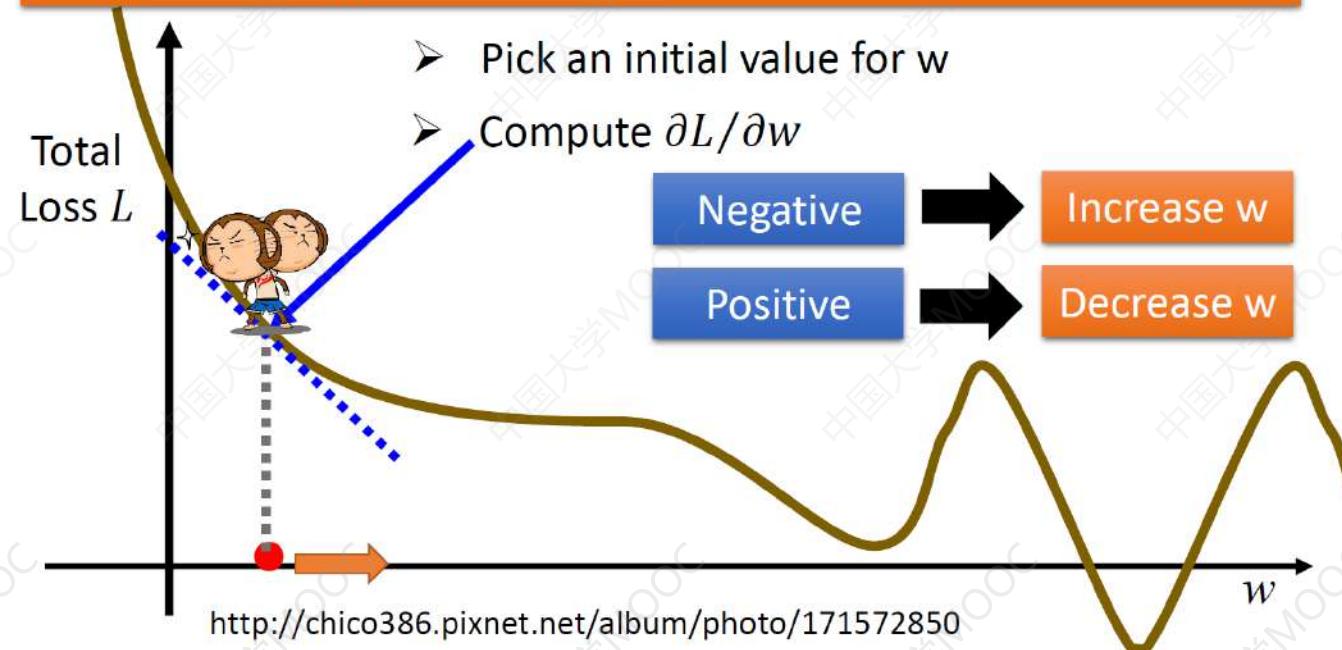


# 实例：数字识别

## Gradient Descent

Network parameters  $\theta = \{w_1, w_2, \dots, b_1, b_2, \dots\}$

Find ***network parameters  $\theta^*$***  that minimize total loss  $L$

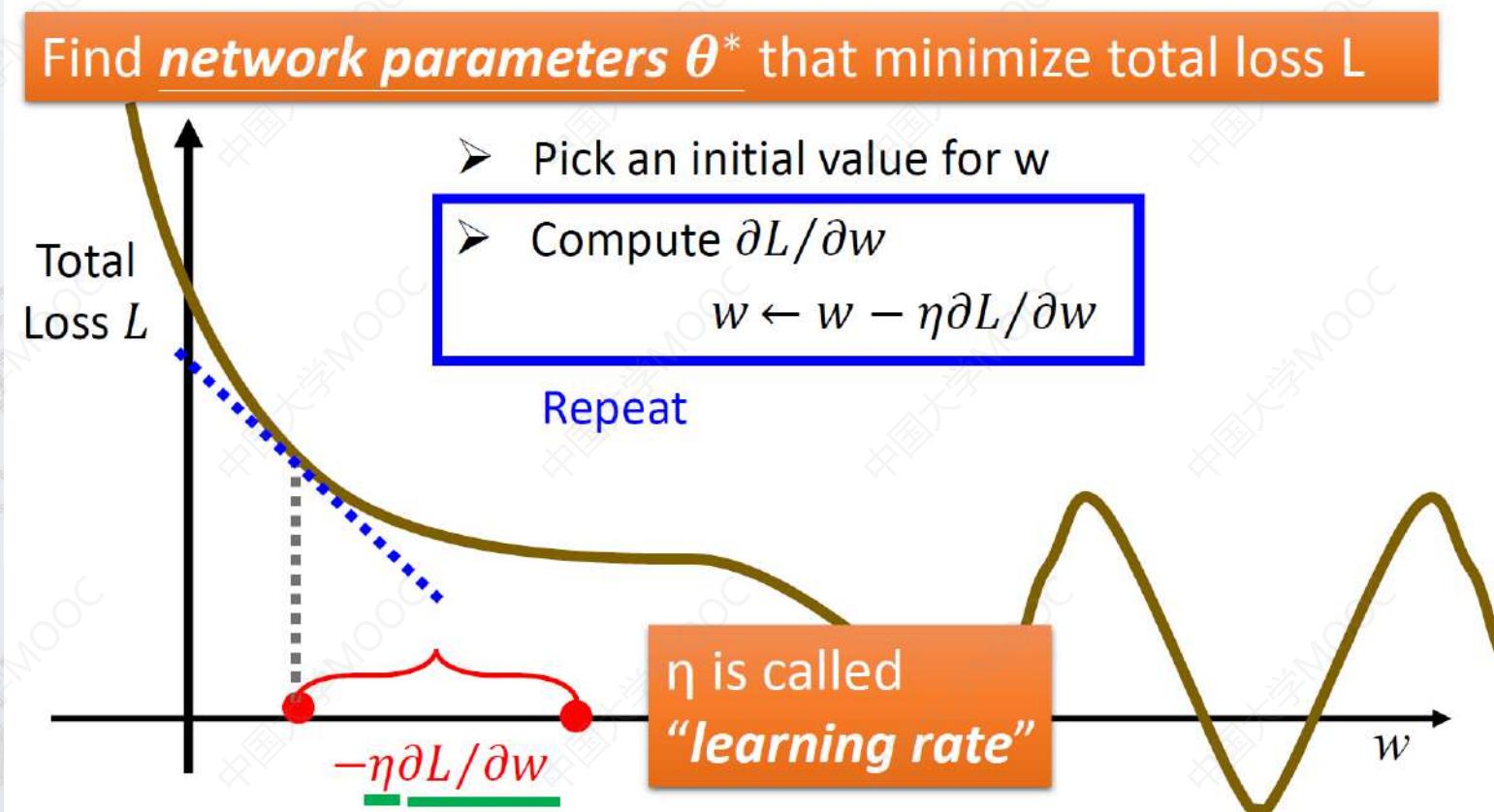


选取一个初始值 $w$ ，计算偏导 $\partial L / \partial w$   
(因为有多个参数)， $L$ 为损失函数。

偏导为正，减小 $w$

偏导为负，增大 $w$

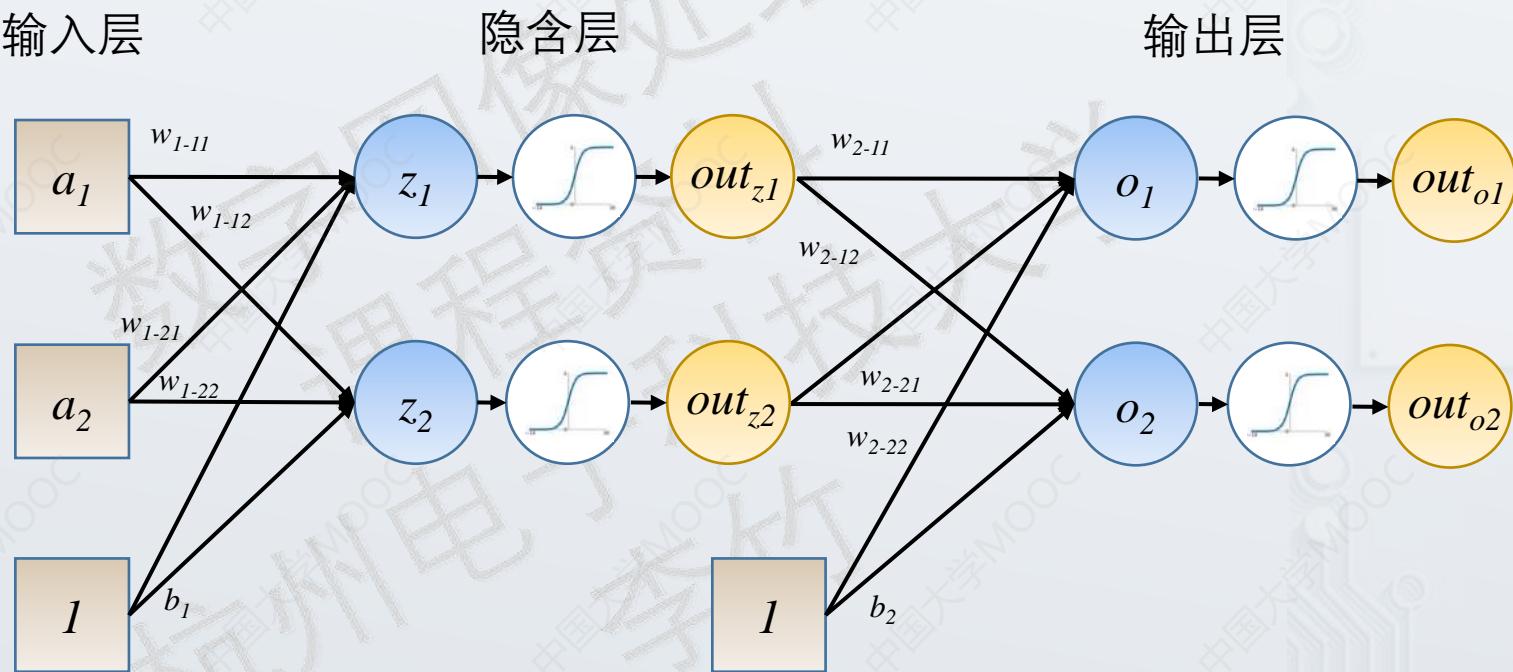
# 实例：数字识别



如此处偏导为负，将参数  $w$  增加为  $w - \eta \partial L / \partial w$  (因为有多个参数) ,  
 $\eta$  称为学习率，学习率越大，收敛的步进就越大。

# 正向传输

输入层



输入:  $a_1=0.05, a_2=0.10$

权重:

$w_{1-11}=0.05, w_{1-12}=0.20, w_{1-21}=0.25,$

$w_{1-22}=0.30$

$w_{2-11}=0.40, w_{2-12}=0.45, w_{2-21}=0.50,$

$w_{2-22}=0.55$

$b_1=0.35, b_2=0.60$

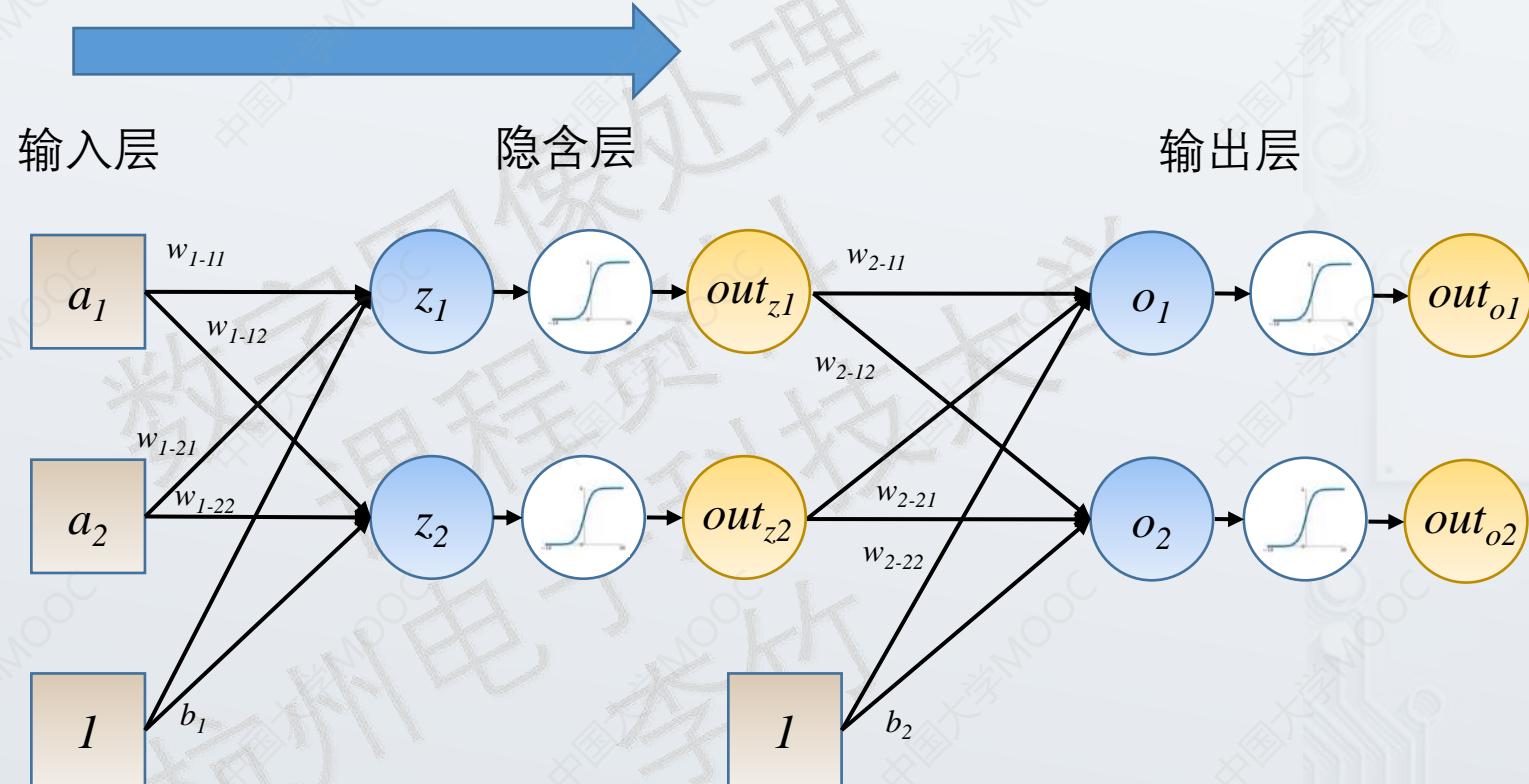
激活函数: sigmoid

优化目标: 使输出为

$out_{o1}=0.01$

$out_{o2}=0.99$

# 正向传输



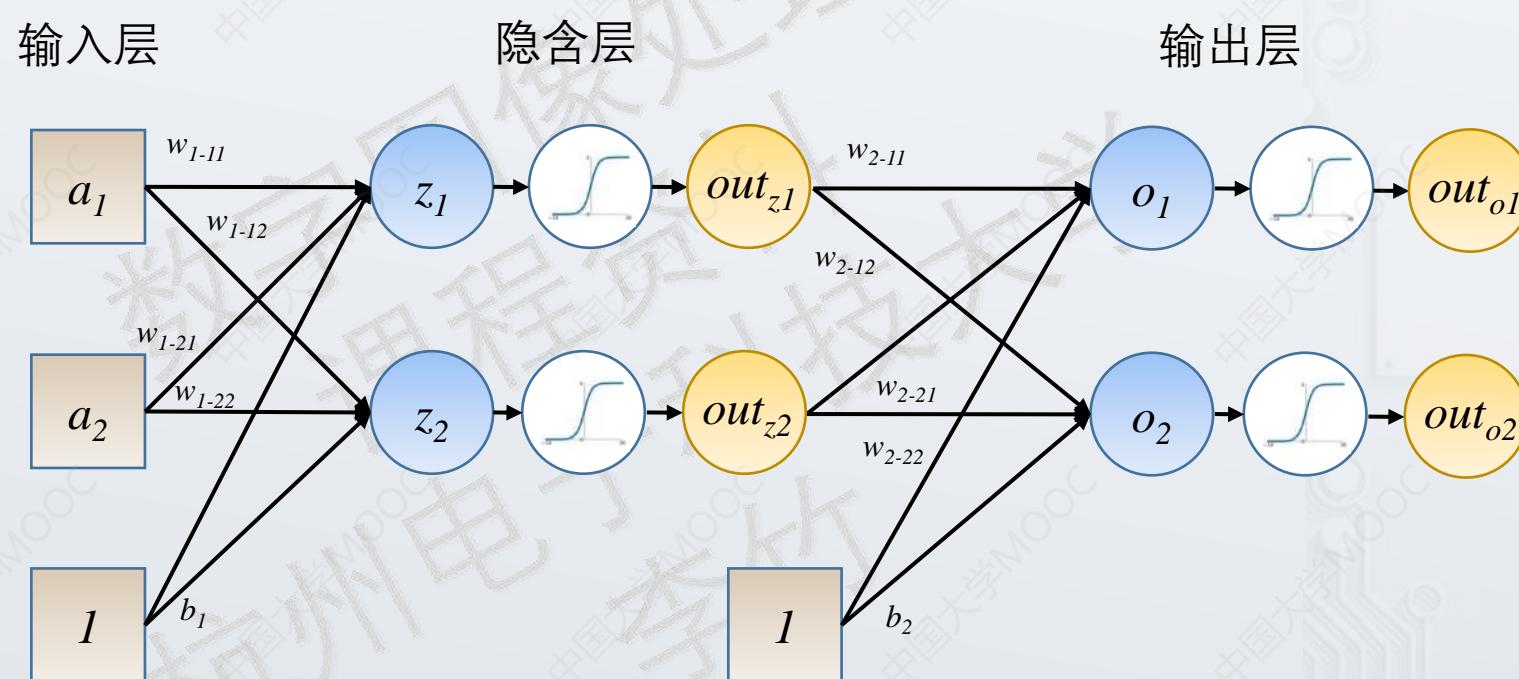
$$\begin{aligned}z_1 &= a_1 \times w_{1-11} + a_2 \times w_{1-21} + b_1 \times 1 \\&= 0.05 \times 0.05 + 0.10 \times 0.25 + 0.35 \times 1 \\&= 0.3775\end{aligned}$$

经过激活函数作用

$$out_{z_1} = \frac{1}{1 + e^{-0.3775}} = 0.593269992$$

同理可得  $out_{z_2} = 0.596884378$

# 正向传输



$$\begin{aligned} o_1 &= out_{z1} \times w_{2-11} + out_{z2} \times w_{2-21} + b_2 \times 1 \\ &= 0.4 \times 0.593269992 + 0.45 \times 0.596884378 + 0.6 \times 1 \\ &= 1.105905967 \end{aligned}$$

↓  
经过激活函数作用 (设为  $out_{o1}$ )

$$out_{o1} = \frac{1}{1 + e^{-1.105905967}} = 0.75136507$$

同理可得  $out_{o2} = 0.772928465$

前向传输结束, 得到输出:

$$out_{o1} = 0.75136507$$

$$out_{o2} = 0.772928465$$

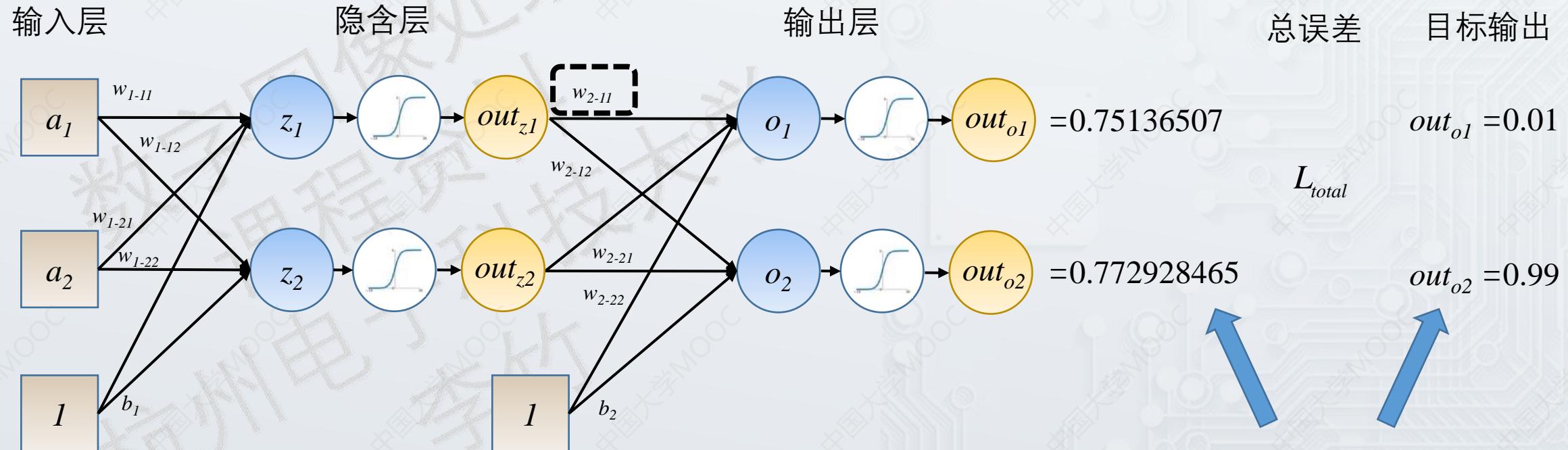
下一步, 反向传输, 调整参数, 使输出接近

$$out_{o1} = 0.01$$

$$out_{o2} = 0.99$$

# 反向传输

我们首先通过偏导数计算 $w_{2-11}$ 对整体误差的影响，并调节 $w_{2-11}$ 减小Loss

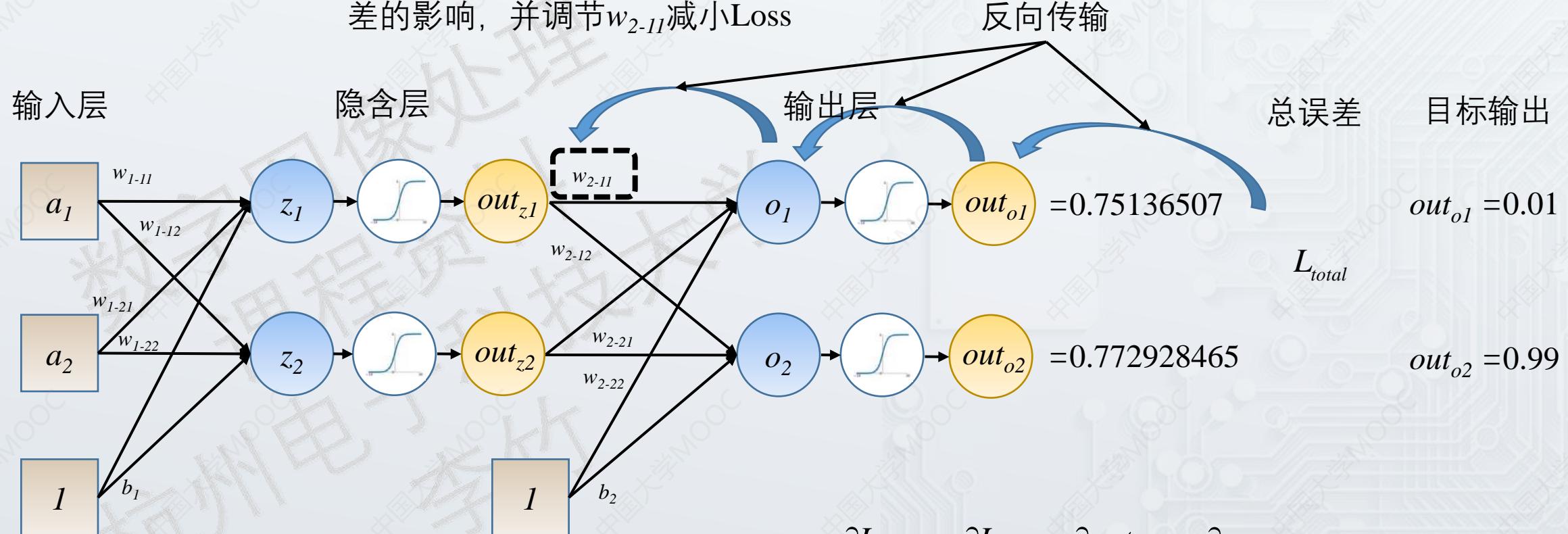


使用square error计算总损失 $L_{total}$

$$L_{total} = \frac{1}{2}(0.01 - 0.75136507)^2 + \frac{1}{2}(0.99 - 0.772928465)^2 \\ = 0.298371109$$

# 反向传输

我们首先通过偏导数计算 $w_{2-11}$ 对整体误差的影响，并调节 $w_{2-11}$ 减小Loss



复合函数求导的链式法则：

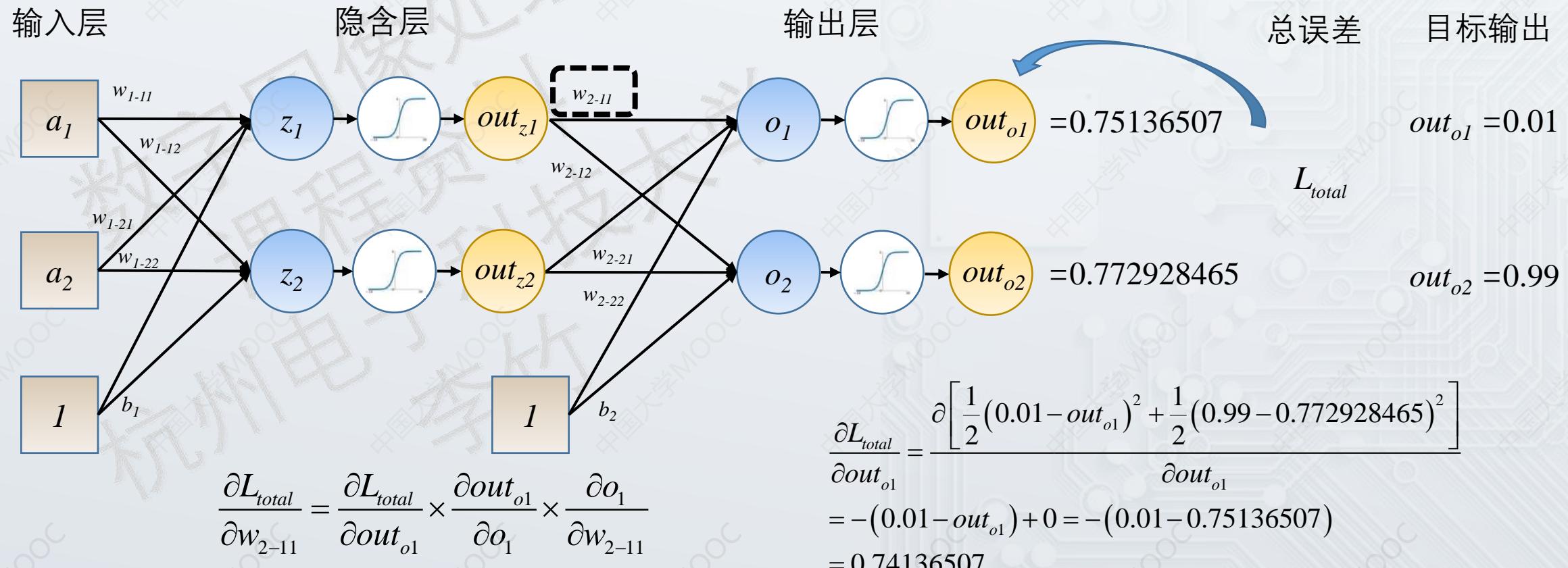
$$\frac{\partial u}{\partial x} = \frac{\partial u}{\partial y} \times \frac{\partial y}{\partial z} \times \frac{\partial z}{\partial x}$$

$$\frac{\partial L_{total}}{\partial w_{2-11}} = \underbrace{\frac{\partial L_{total}}{\partial out_{o1}}}_{\text{分别求出}} \times \underbrace{\frac{\partial out_{o1}}{\partial o_1}}_{\text{分别求出}} \times \underbrace{\frac{\partial o_1}{\partial w_{2-11}}}_{\text{分别求出}}$$

分别求出

# 反向传输

我们首先通过偏导数计算 $w_{2-11}$ 对整体误差的影响，并调节 $w_{2-11}$ 减小Loss



# 反向传输

我们首先通过偏导数计算 $w_{2-11}$ 对整体误差的影响，并调节 $w_{2-11}$ 减小Loss

输入层

$$a_1$$

$$w_{1-II}$$

$$w_{1-I2}$$

$$a_2$$

$$w_{1-21}$$

$$w_{1-22}$$

$$I$$

$$b_1$$

隐含层

$$z_1$$

$$w_{2-11}$$

$$w_{2-12}$$

$$z_2$$

$$w_{2-21}$$

$$w_{2-22}$$

$$I$$

$$b_2$$

输出层

$$o_1$$

$$w_{2-11}$$

$$o_2$$

$$w_{2-21}$$

$$w_{2-22}$$

$$out_{o1}$$

$$out_{o2}$$

$$out_{o1} = 0.75136507$$

$$out_{o2} = 0.772928465$$

总误差

$$L_{total}$$

目标输出

$$out_{o1} = 0.01$$

$$out_{o2} = 0.99$$

$$\frac{\partial L_{total}}{\partial w_{2-11}} = \frac{\partial L_{total}}{\partial out_{o1}} \times \frac{\partial out_{o1}}{\partial o_1} \times \frac{\partial o_1}{\partial w_{2-11}}$$

$$\begin{aligned}\frac{\partial out_{o1}}{\partial o_1} &= \frac{\partial}{\partial o_1} \left[ \frac{1}{1+e^{-o_1}} \right] \\ &= out_1(1-out_1) = 0.75136507(1-0.75136507) \\ &= 0.186815602\end{aligned}$$

# 反向传输

我们首先通过偏导数计算 $w_{2-11}$ 对整体误差的影响，并调节 $w_{2-11}$ 减小Loss

输入层

$$a_1$$

$$w_{1-11}$$

$$w_{1-12}$$

$$a_2$$

$$w_{1-21}$$

$$w_{1-22}$$

$$I$$

$$b_1$$

隐含层

$$z_1$$

$$z_2$$

$$out_{z1}$$

$$out_{z2}$$

输出层

$$o_1$$

$$o_2$$

$$out_{o1}$$

$$out_{o2}$$

总误差

$$L_{total}$$

目标输出

$$out_{o1} = 0.01$$

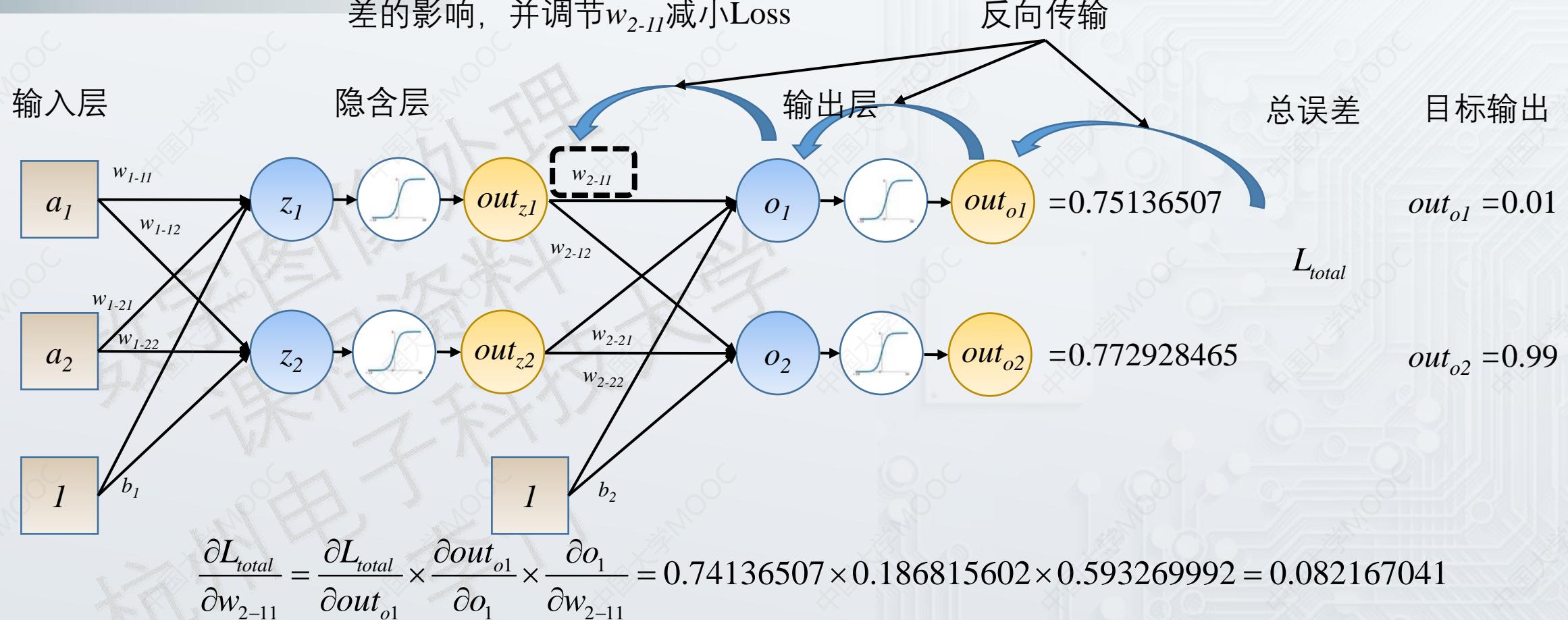
$$out_{o2} = 0.99$$

$$\frac{\partial L_{total}}{\partial w_{2-11}} = \frac{\partial L_{total}}{\partial out_{o1}} \times \frac{\partial out_{o1}}{\partial o_1} \times \frac{\partial o_1}{\partial w_{2-11}}$$

$$\begin{aligned}\frac{\partial o_1}{\partial w_{2-11}} &= \frac{\partial [out_{z1} \times w_{2-11} + out_{z2} \times w_{2-21} + b_2 \times 1]}{\partial w_{2-11}} \\ &= \frac{\partial [out_{z1} \times w_{2-11} + 0 + 0]}{\partial w_{2-11}} = out_{z1} = 0.593269992\end{aligned}$$

# 反向传输

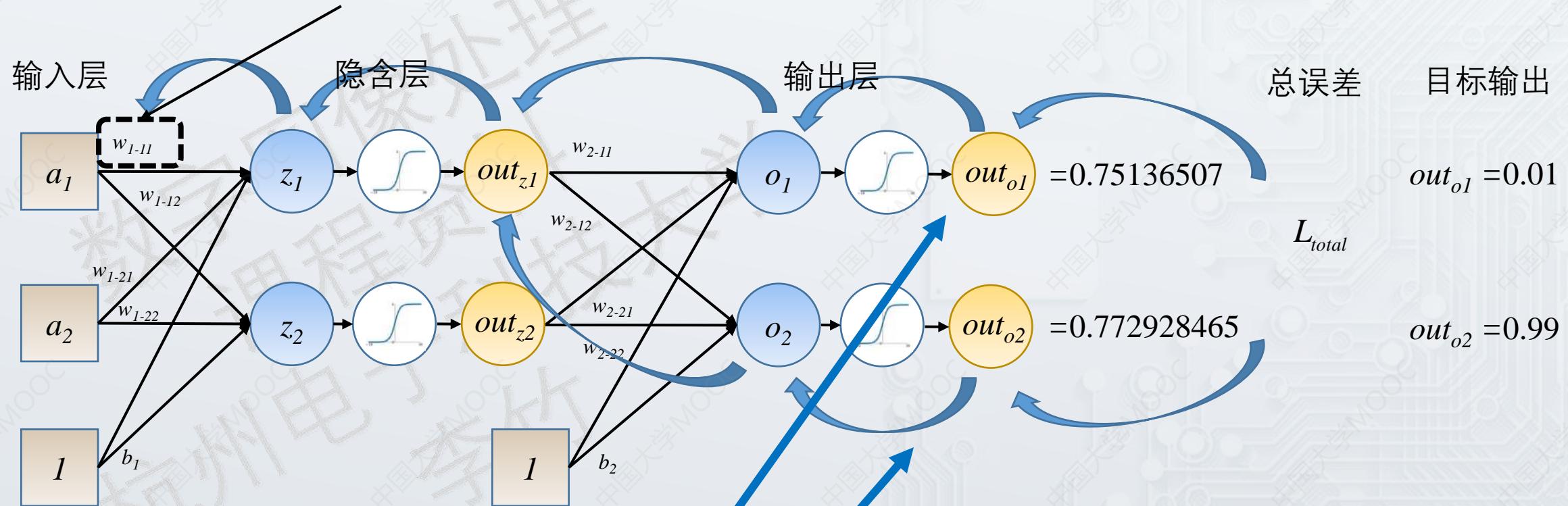
我们首先通过偏导数计算 $w_{2-11}$ 对整体误差的影响，并调节 $w_{2-11}$ 减小Loss



根据之前的权重更新公式（学习率取0.5），  
更新后的权重为  $w_{2-11}' = w_{2-11} - \eta \times \frac{\partial L_{total}}{\partial w_{2-11}} = 0.4 - 0.5 \times 0.082167041 = 0.35891648$

# 反向传输

如果要计算 $w_{1-11}$ , 会接收到 $o_1$ 和 $o_2$ 的误差

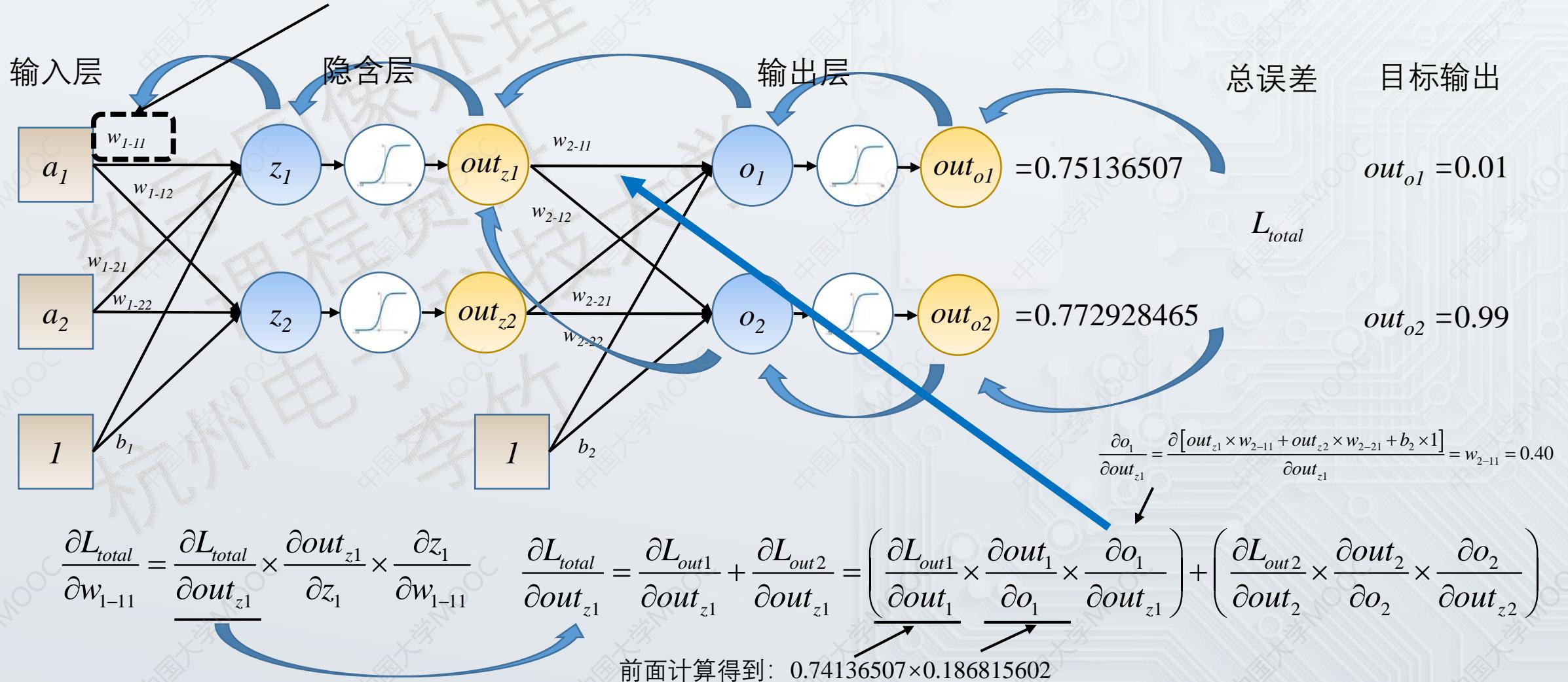


$$\frac{\partial L_{total}}{\partial w_{1-11}} = \frac{\partial L_{total}}{\partial out_{z1}} \times \frac{\partial out_{z1}}{\partial z_1} \times \frac{\partial z_1}{\partial w_{1-11}}$$

$$\frac{\partial L_{total}}{\partial out_{z1}} = \frac{\partial L_{out1}}{\partial out_{z1}} + \frac{\partial L_{out2}}{\partial out_{z1}} = \left( \frac{\partial L_{out1}}{\partial out_1} \times \frac{\partial out_1}{\partial o_1} \times \frac{\partial o_1}{\partial out_{z1}} \right) + \left( \frac{\partial L_{out2}}{\partial out_2} \times \frac{\partial out_2}{\partial o_2} \times \frac{\partial o_2}{\partial out_{z1}} \right)$$

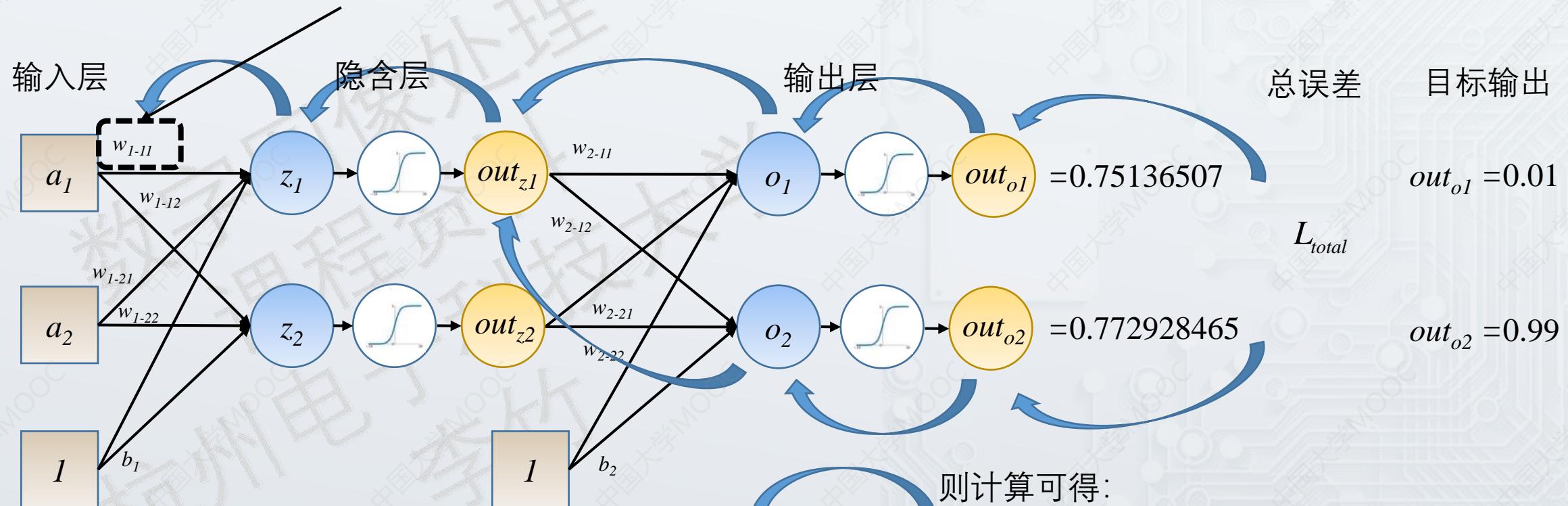
# 反向传输

如果要计算 $w_{1-11}$ , 会接收到 $o_1$ 和 $o_2$ 的误差



# 反向传输

如果要计算 $w_{1-11}$ , 会接收到 $o_1$ 和 $o_2$ 的误差



$$\frac{\partial L_{total}}{\partial w_{1-11}} = \frac{\partial L_{total}}{\partial out_{z1}} \times \frac{\partial out_{z1}}{\partial z_1} \times \frac{\partial z_1}{\partial w_{1-11}}$$

$$\frac{\partial L_{total}}{\partial out_{z1}} = \frac{\partial L_{out1}}{\partial out_{z1}} + \frac{\partial L_{out2}}{\partial out_{z1}}$$

则计算可得:

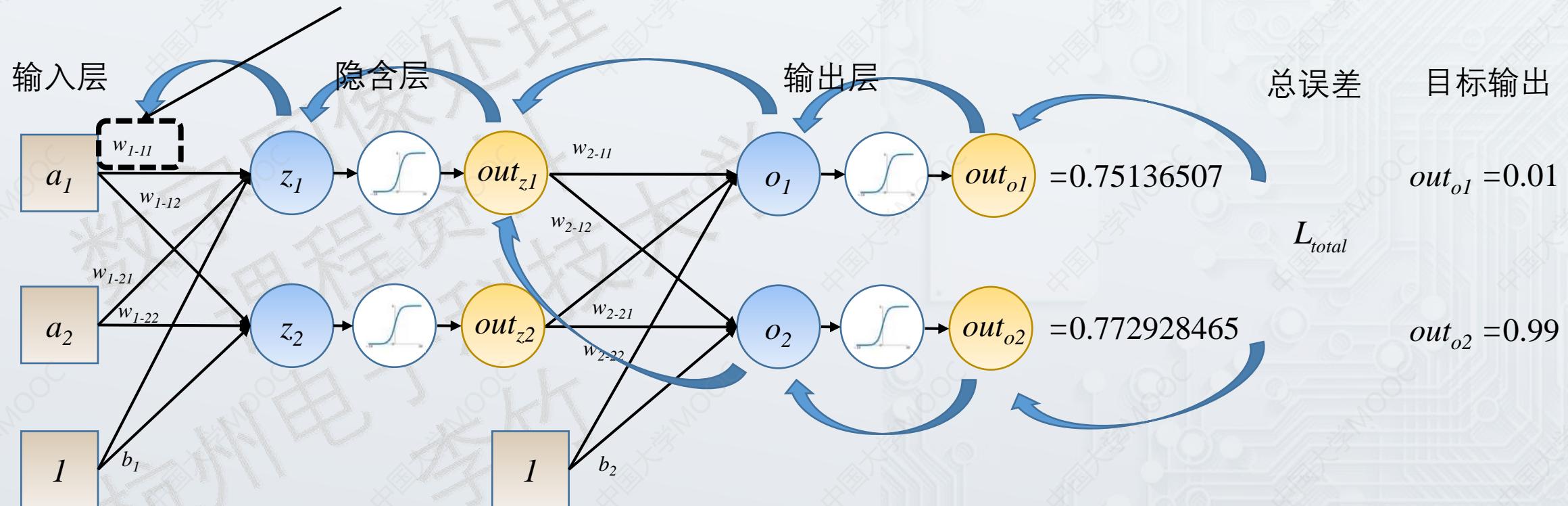
$$0.74136507 \times 0.186815602 \times 0.40 = 0.055399425$$

同理可得:

$$-0.019049119$$

# 反向传输

如果要计算 $w_{1-11}$ , 会接收到 $o_1$ 和 $o_2$ 的误差

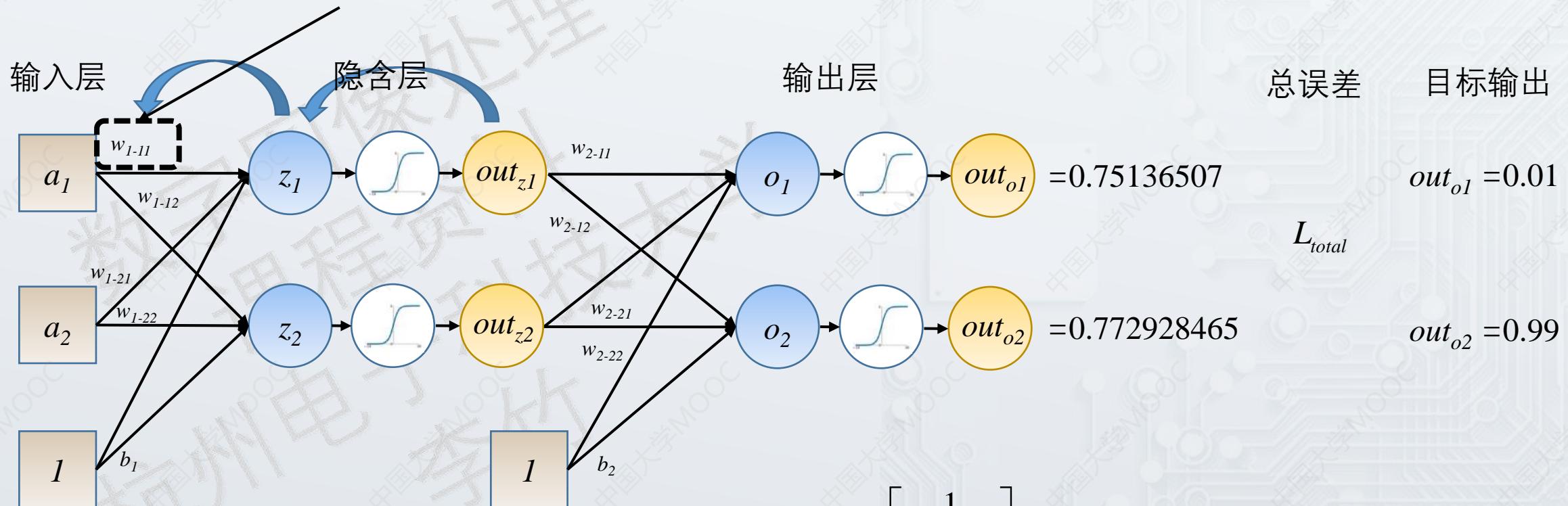


$$\frac{\partial L_{total}}{\partial w_{1-11}} = \frac{\partial L_{total}}{\partial out_{z1}} \times \frac{\partial out_{z1}}{\partial z_1} \times \frac{\partial z_1}{\partial w_{1-11}}$$

$$\frac{\partial L_{total}}{\partial out_{z1}} = \frac{\partial L_{out1}}{\partial out_{z1}} + \frac{\partial L_{out2}}{\partial out_{z1}} = 0.055399425 + (-0.019049119) = 0.036350306$$

# 反向传输

如果要计算 $w_{1-11}$ , 会接收到 $o_1$ 和 $o_2$ 的误差



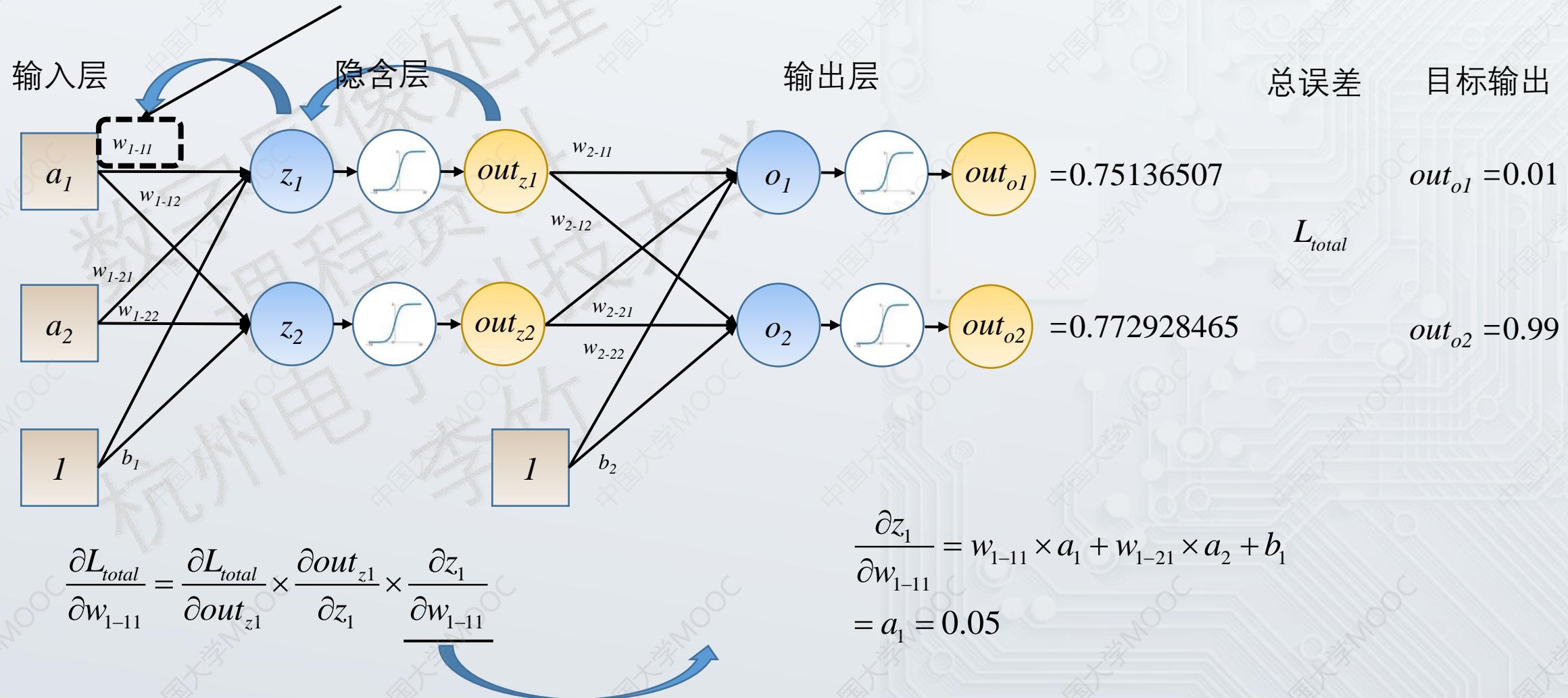
$$\frac{\partial L_{total}}{\partial w_{1-11}} = \frac{\partial L_{total}}{\partial out_{z1}} \times \frac{\partial out_{z1}}{\partial z_1} \times \frac{\partial z_1}{\partial w_{1-11}}$$

$$\begin{aligned}\frac{\partial out_{z1}}{\partial z_1} &= \frac{\partial \left[ \frac{1}{1+e^{-z_1}} \right]}{\partial z_1} = out_{z1}(1 - out_{z1}) \\ &= 0.59326999 \times (1 - 0.59326999) = 0.241300709\end{aligned}$$

$$\frac{\partial L_{total}}{\partial out_{z1}} = 0.055399425 + (-0.019049119) = 0.036350306$$

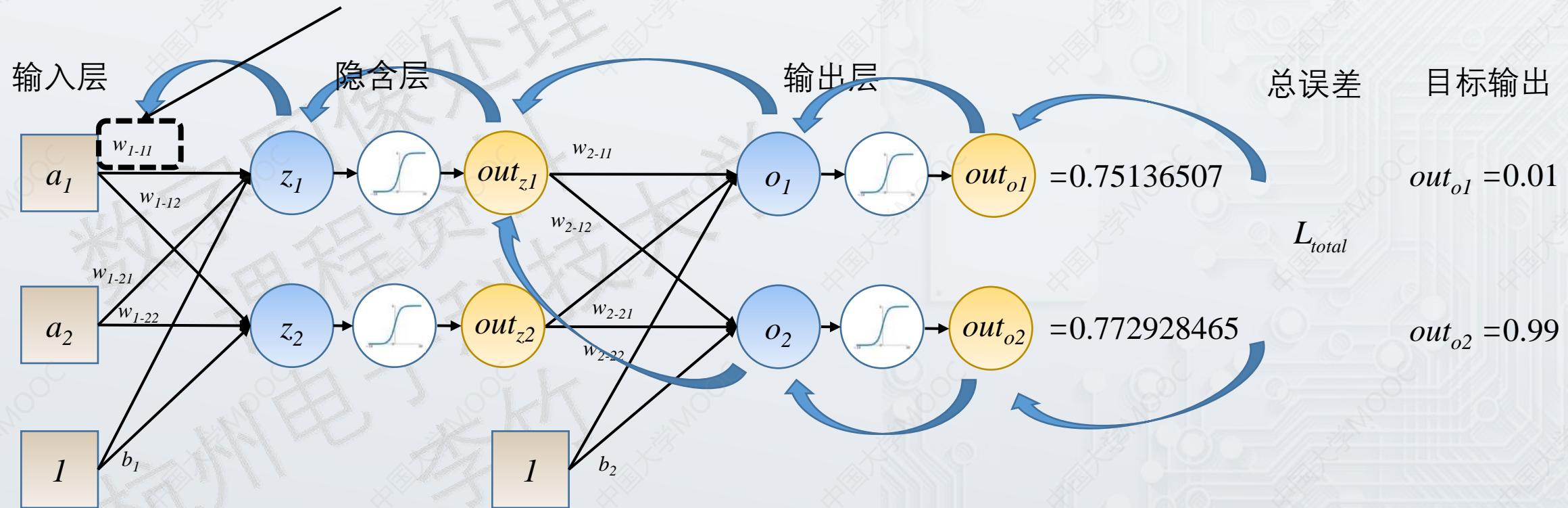
# 反向传输

如果要计算 $w_{1-11}$ , 会接收到 $o_1$ 和 $o_2$ 的误差



# 反向传输

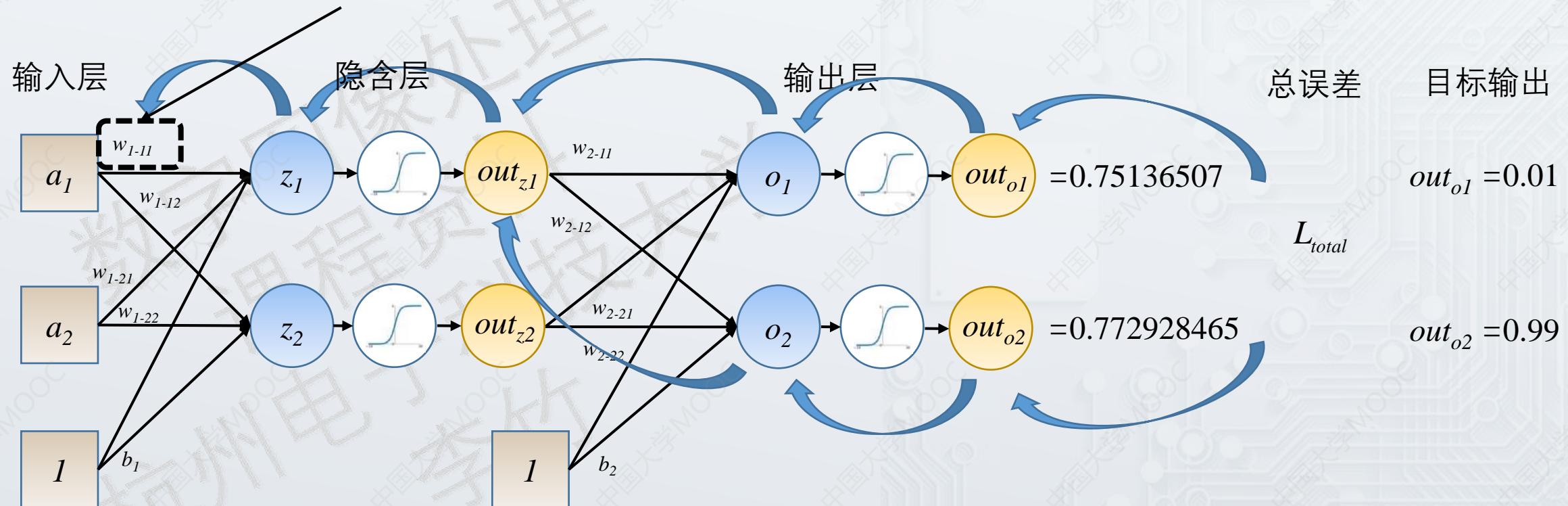
如果要计算 $w_{1-11}$ , 会接收到 $o_1$ 和 $o_2$ 的误差



$$\frac{\partial L_{total}}{\partial w_{1-11}} = \frac{\partial L_{total}}{\partial out_{z1}} \times \frac{\partial out_{z1}}{\partial z_1} \times \frac{\partial z_1}{\partial w_{1-11}} = 0.036350306 \times 0.241300709 \times 0.05 = 0.000438568$$

# 反向传输

如果要计算 $w_{1-11}$ , 会接收到 $o_1$ 和 $o_2$ 的误差



最后更新权重:  $w_{1-11}' = w_{1-11} - \eta \times \frac{\partial L_{total}}{\partial w_{1-11}} = 0.15 - 0.5 \times 0.000438568 = 0.149780716$

# SoftMax

我们在标记训练数据的时候，实质上是

5	“5”
0	“0”
4	“4”
1	“1”
9	“9”
2	“2”
1	“1”
3	“3”

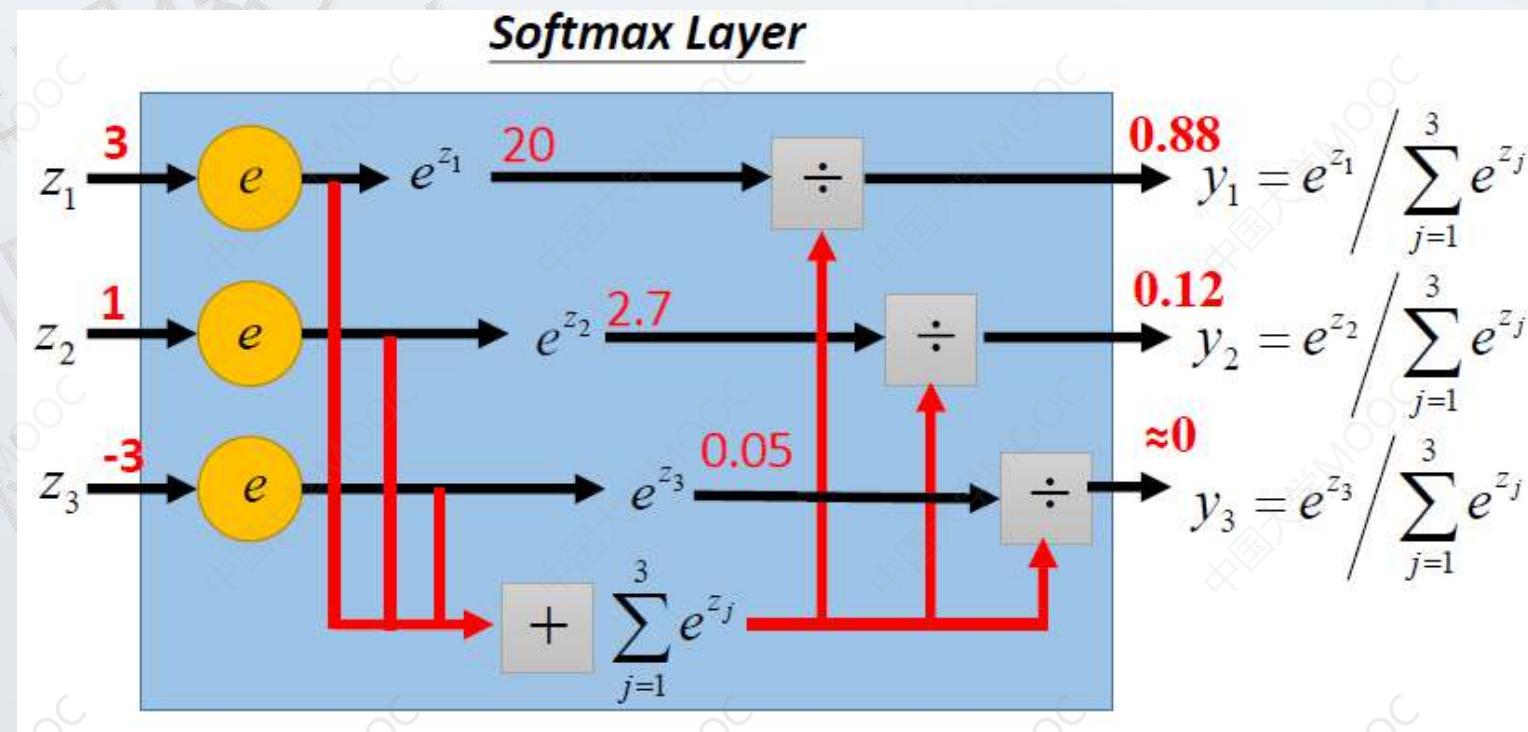
类别	概率
0	0
1	0
2	0
3	0
4	0
5	1
6	0
7	0
8	0
9	0

因为需要和概率比较loss，所以我们需要把输出从不确定的任意值转换为概率。  
通过softmax。

one-hot标签

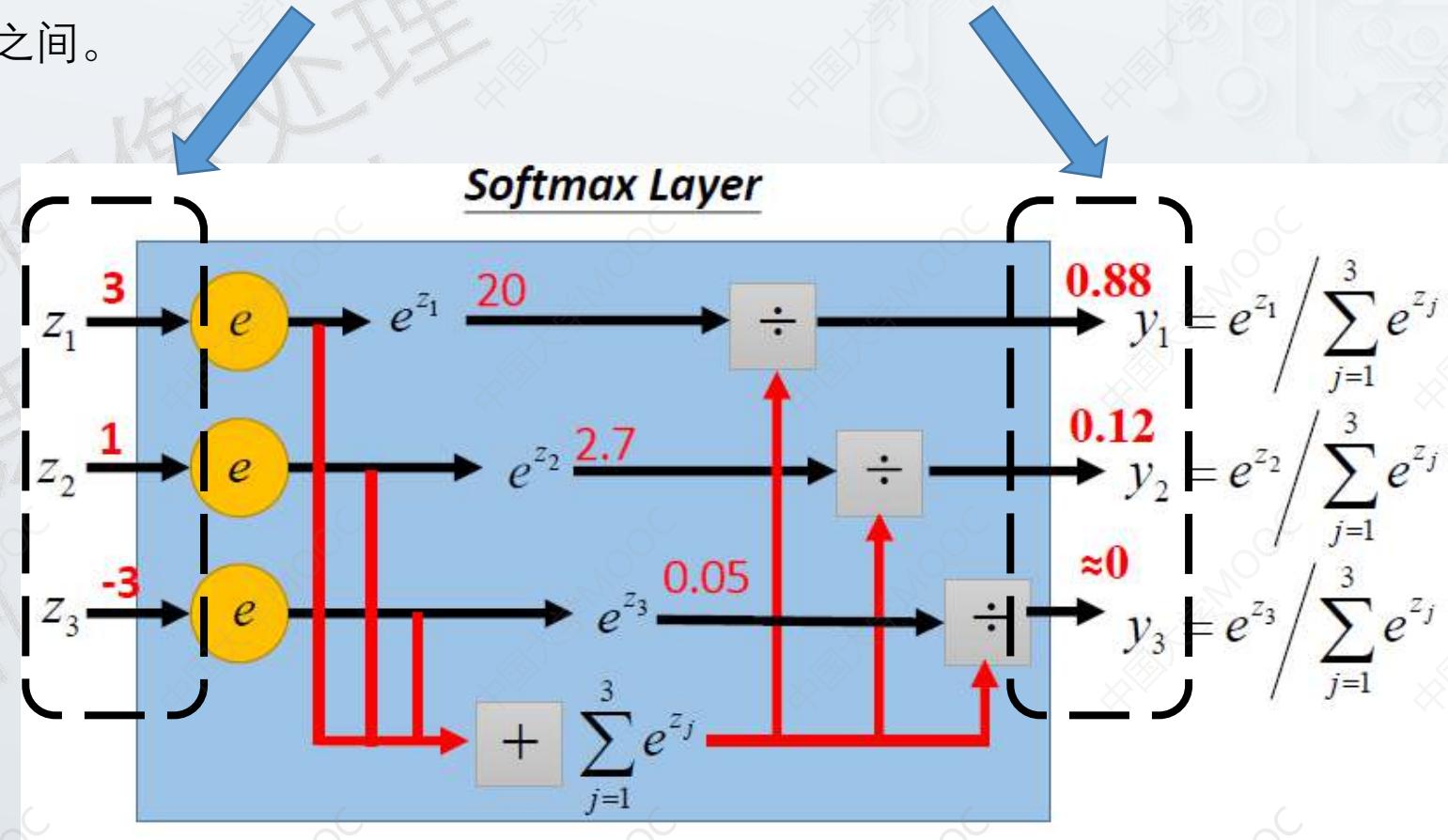
# 反向传输

如输出值为 $z_1=3, z_2=1, z_3=-3$ ,计算 $e^{z_1}, e^{z_2}, e^{z_3}$ , 及总和 $e^{z_1} + e^{z_2} + e^{z_3}$ , 则第一项的输出为 $e^{z_1}/(e^{z_1} + e^{z_2} + e^{z_3})$ , 以此类推。



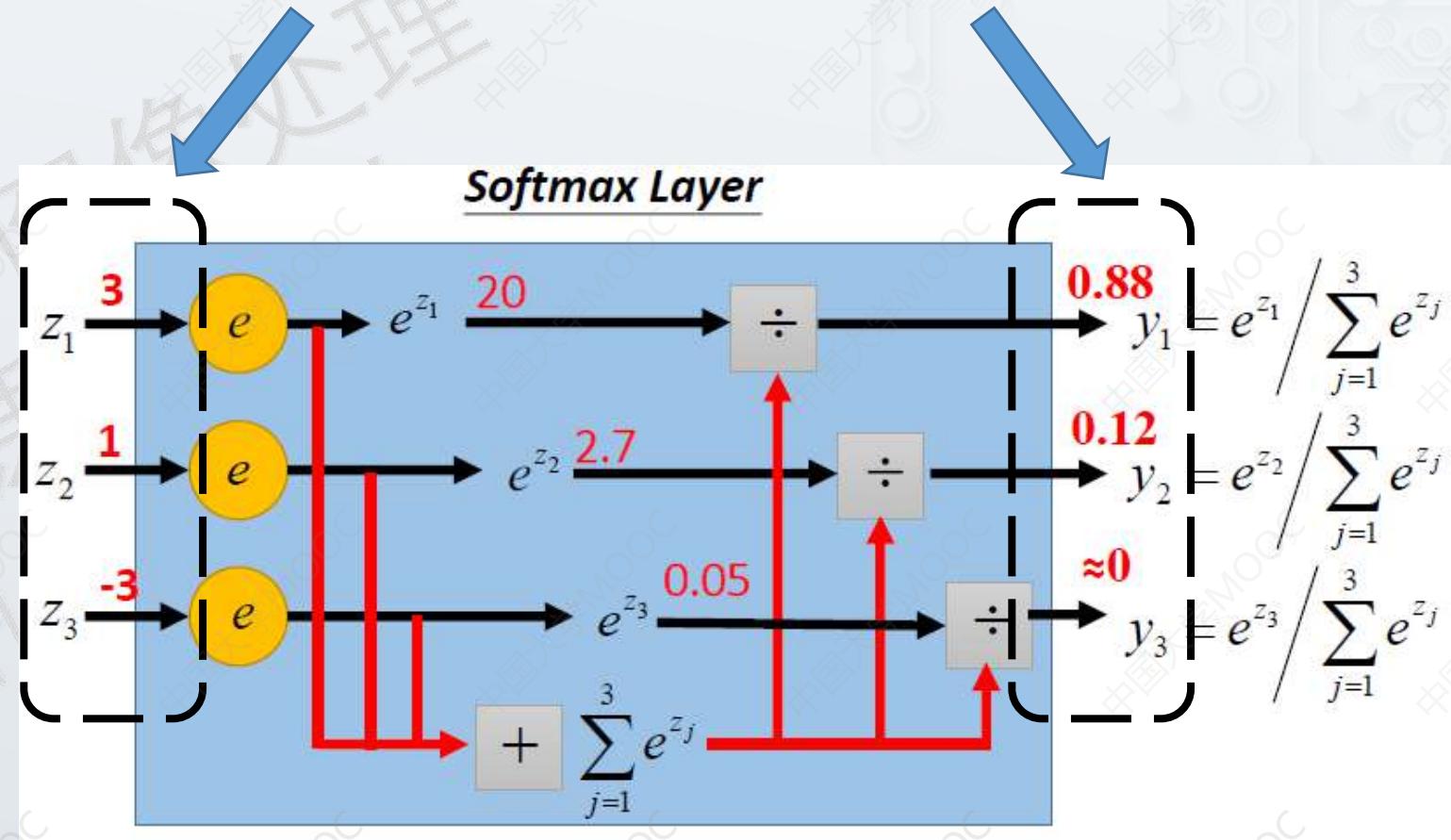
# 反向传输

- 将输出大小，正负不确定的值，转化为概率形式，总和为1，每一项在0到1之间。



# 反向传输

2. 指数函数的特点，大的越大，小的越小（接近于0）。



# 反向传输

方差

Square Error

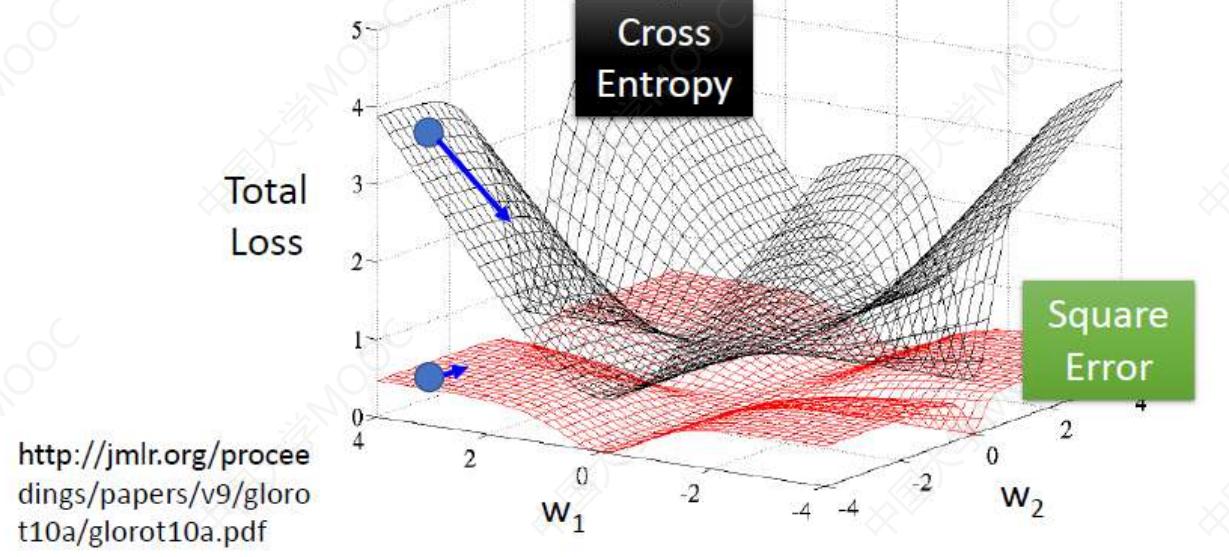
$$\sum_{i=1}^{10} (y_i - \hat{y}_i)^2 = 0$$

交叉熵

Cross Entropy

$$-\sum_{i=1}^{10} \hat{y}_i \ln y_i = 0$$

When using softmax output layer,  
choose cross entropy



**谢谢！**