



中国科学技术大学

University of Science and Technology of China

# 计算机图形学

## Computer Graphics

陈仁杰

[renjiec@ustc.edu.cn](mailto:renjiec@ustc.edu.cn)

<http://staff.ustc.edu.cn/~renjiec>

# GAMES online courses

计算机图形学与混合现实在线平台

<https://games-cn.org/>

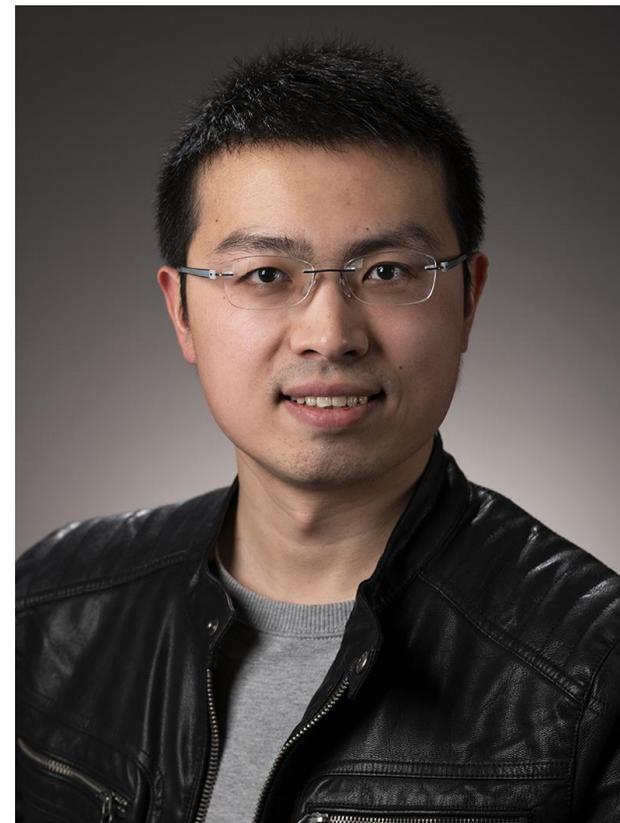
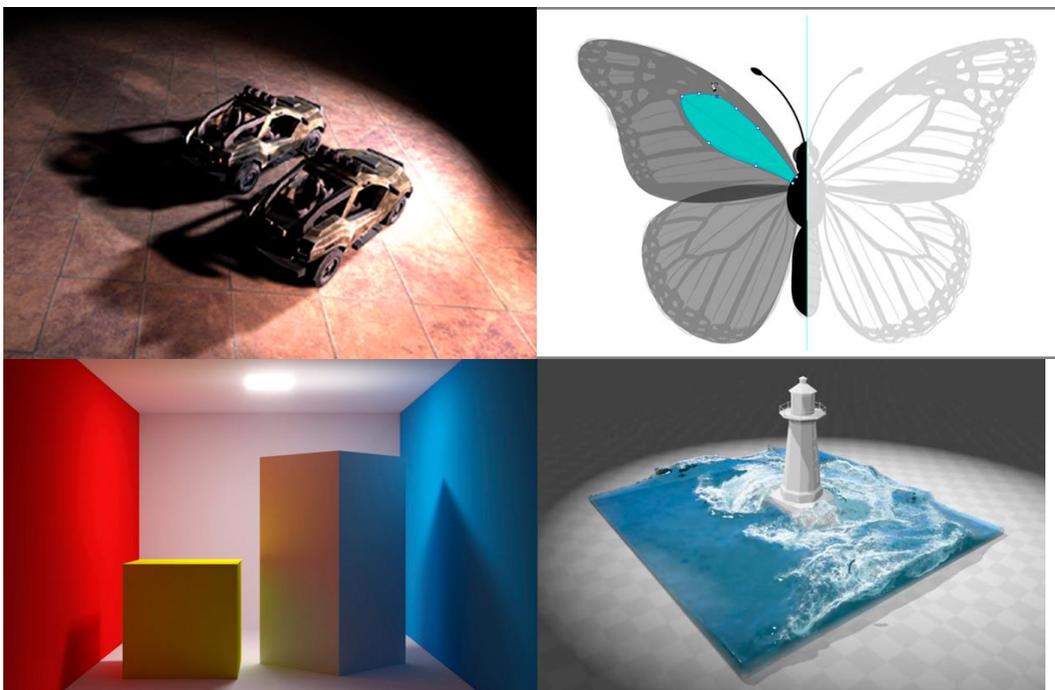
Graphics **A**nd **M**ixed **E**nvironment **S**ymposium

- GAMES101: 现代计算机图形学入门
- GAMES102: 几何建模与处理
- GAMES103: 基于物理的计算机动画入门
- GAMES104: 现代游戏引擎：从入门到实践
- GAMES105: 计算机角色动画基础

# GAMES 101: 现代计算机图形学入门

- 现代计算机图形学的四大组成部分

1. 光栅化成像
2. 几何表示
3. 光的传播理论
4. 动画与模拟



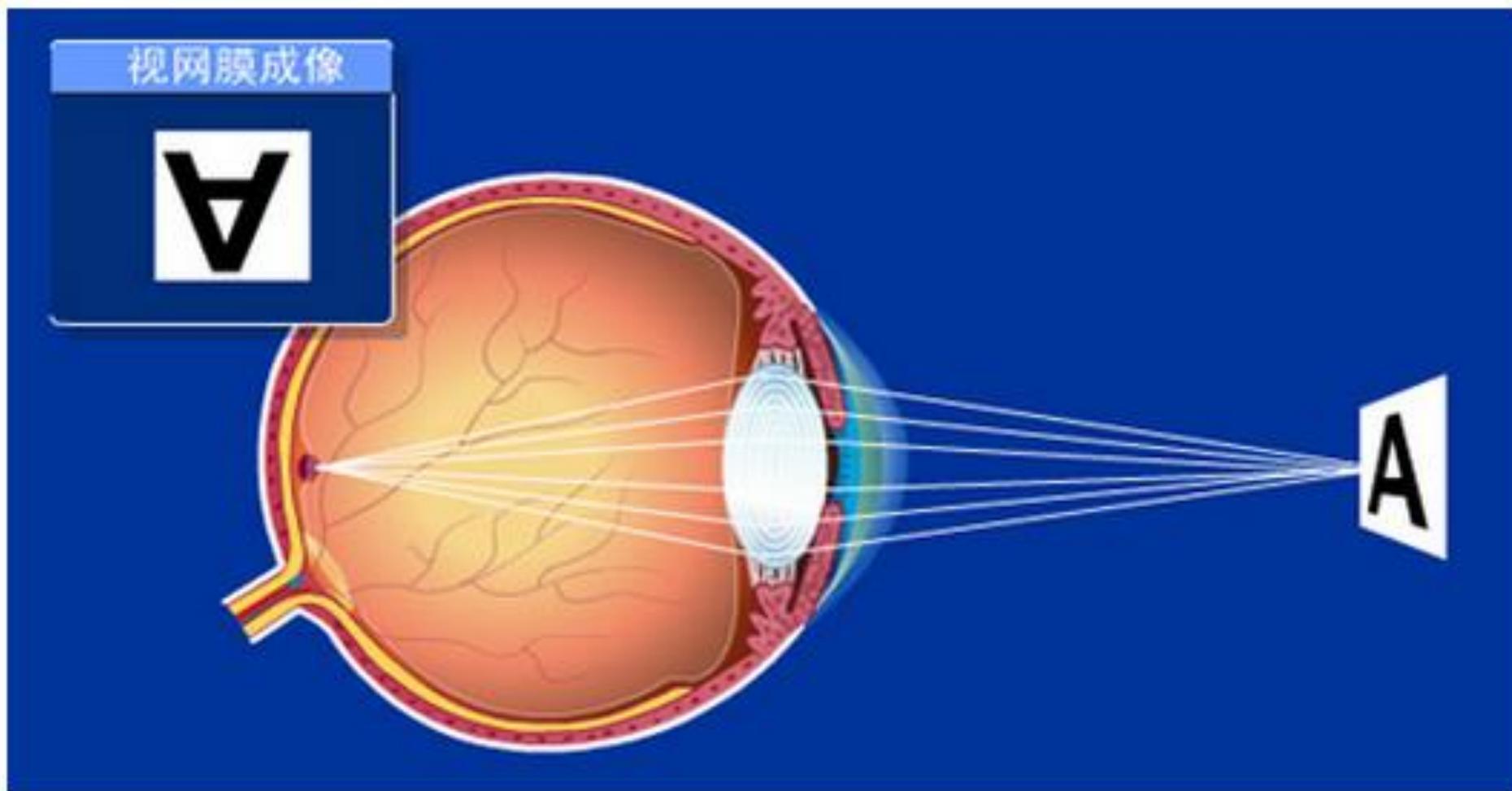
闫令琪, Assistant Professor  
University of California, Santa Barbara

# Coding

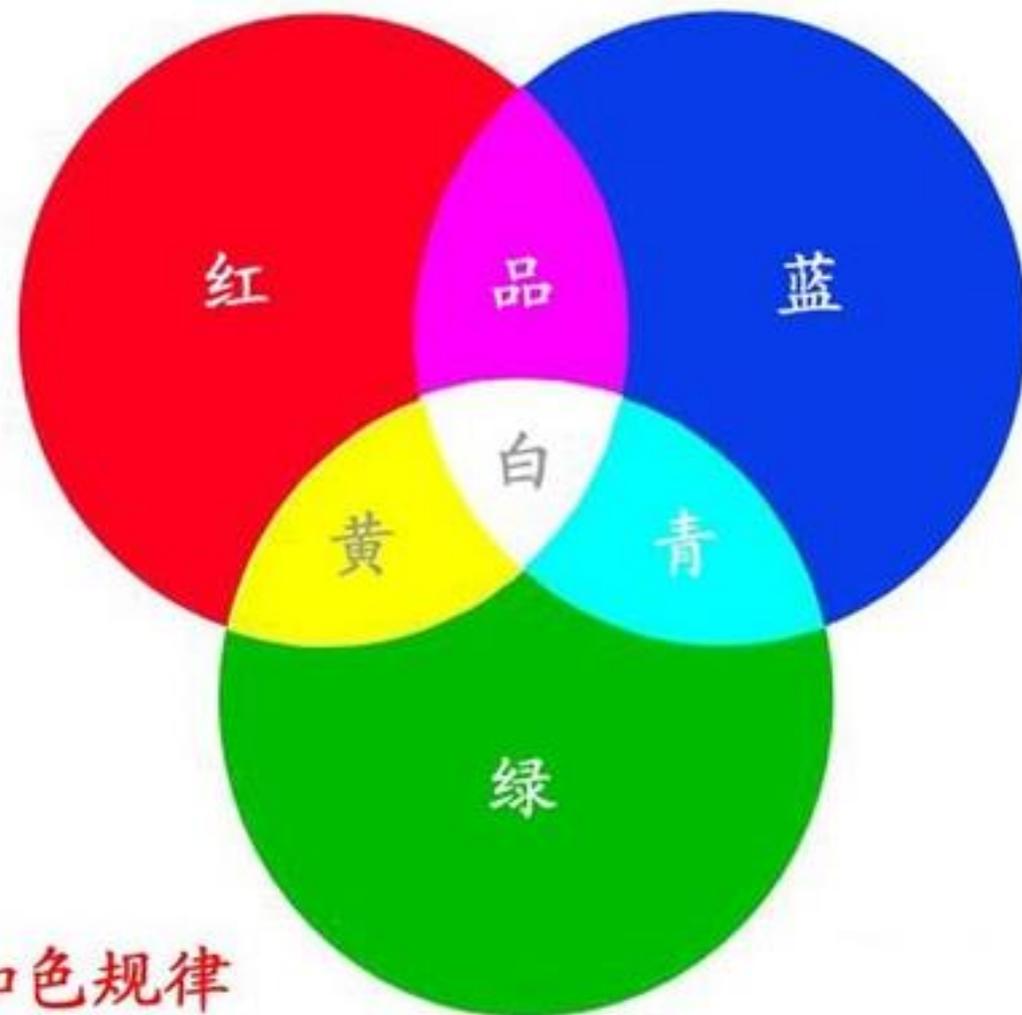
- Git, SourceTree
- VS
  - Release/Debug folder
  - Include, Library
  - Use **relative** path in include

# 数字图像简介

# 视网膜成像：上亿个感光细胞

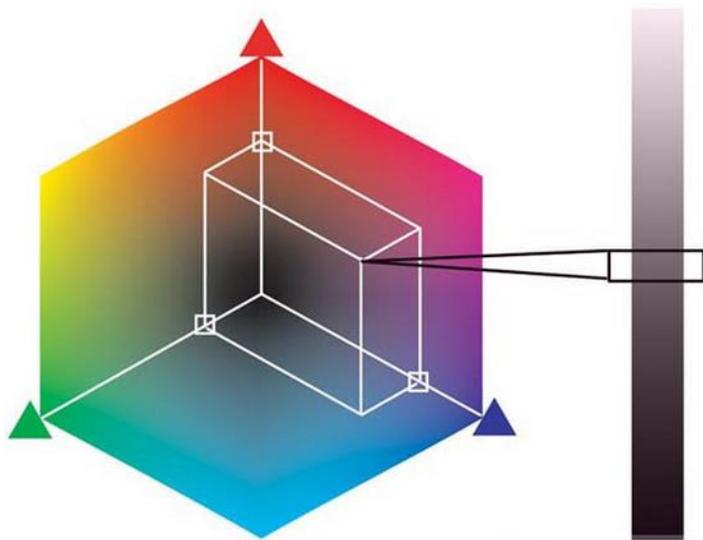


# 颜色三原色：红、绿、蓝



# RGB颜色空间

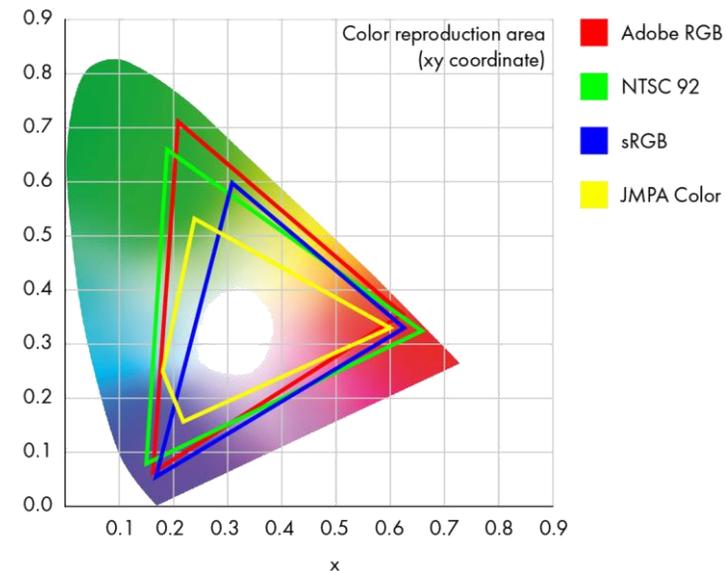
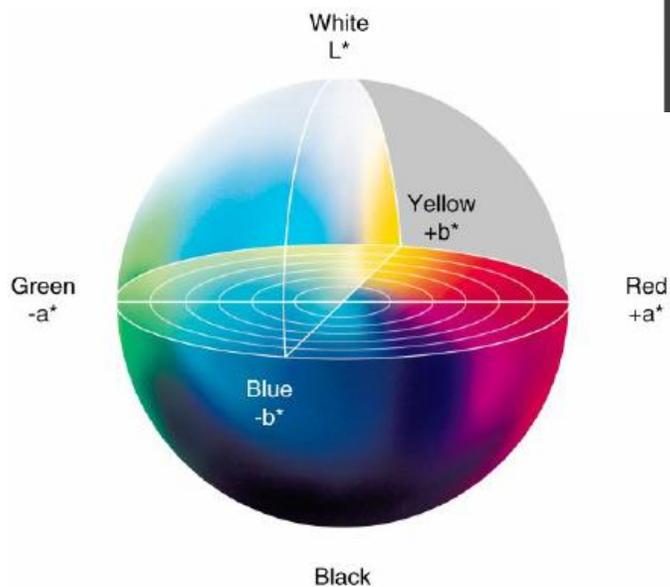
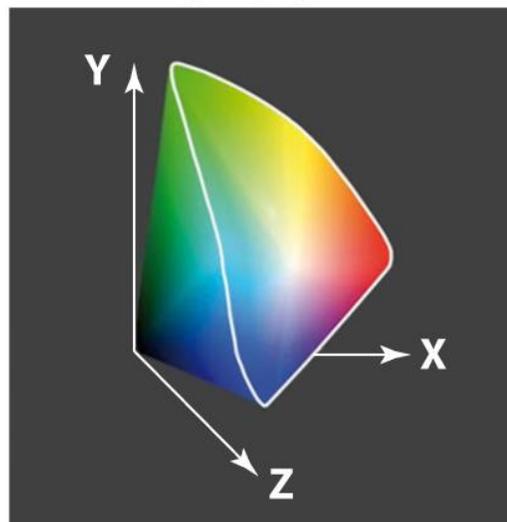
- Color = a **R** + b **G** + c **B**



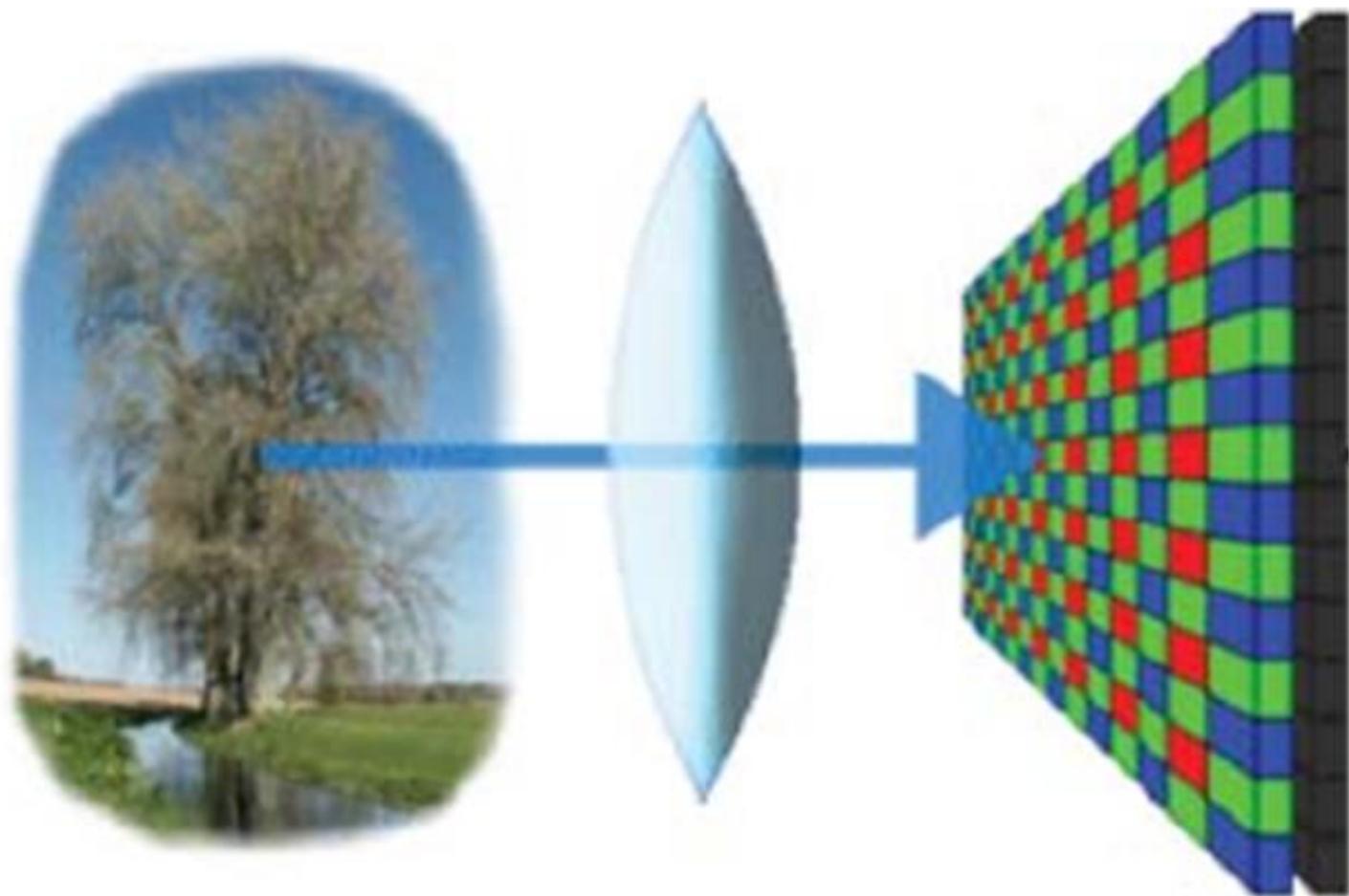
# 颜色空间：不同基函数

- RGB
- CMY (CMYK)
- CIE XYZ
- Lab

CIE XYZ  
The Human Gamut  
3D View

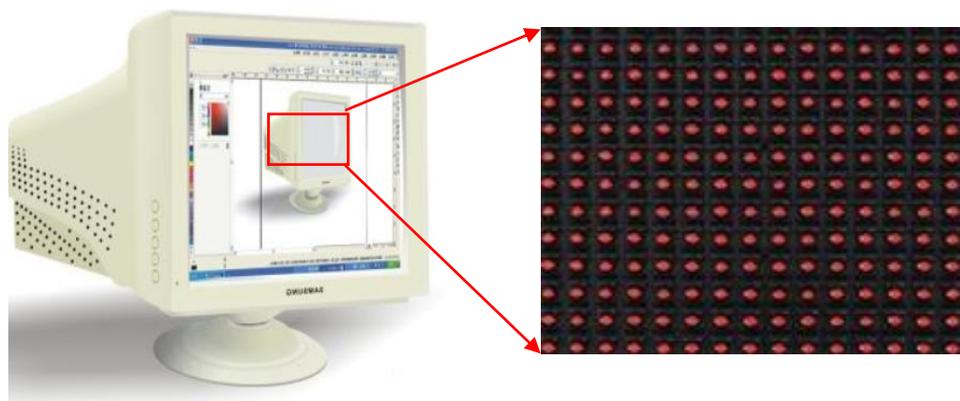
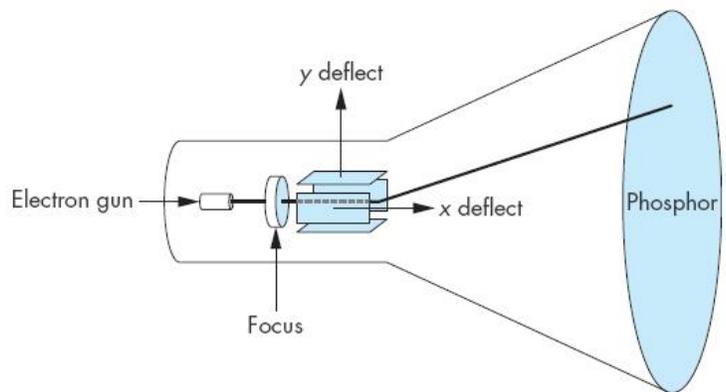


# 数字图像



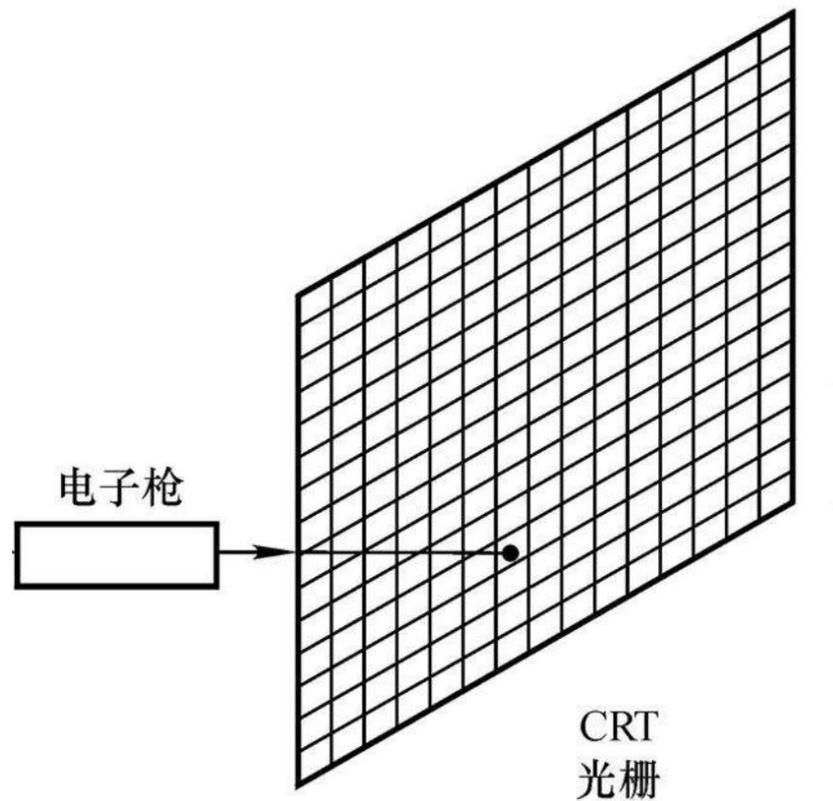
# 显示器的产生

- 1946年，美国宾夕法尼亚大学，世界上第一台电子计算机问世
- 1959年，MIT林肯实验室，第一台阴极射线显像管（CRT）显示器
  - 光栅显示器



# 光栅显示设备

- 逐个将“点”打在屏幕上的相应位置

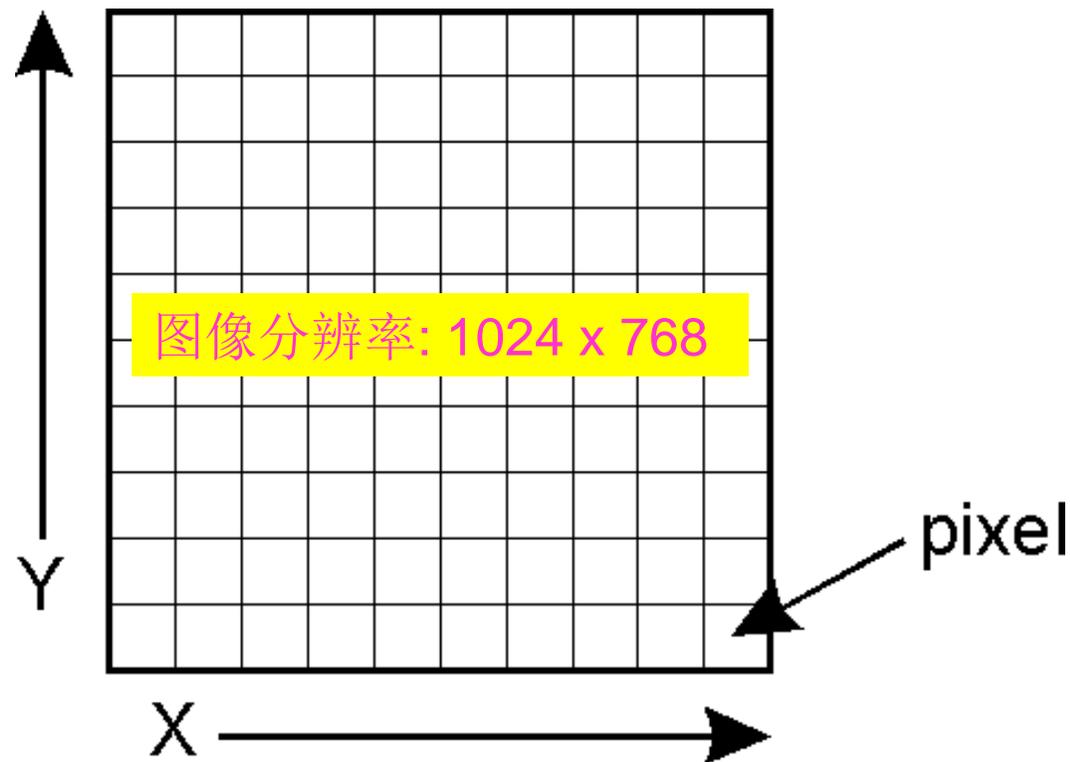


# 现有显示设备基本都是光栅显示器

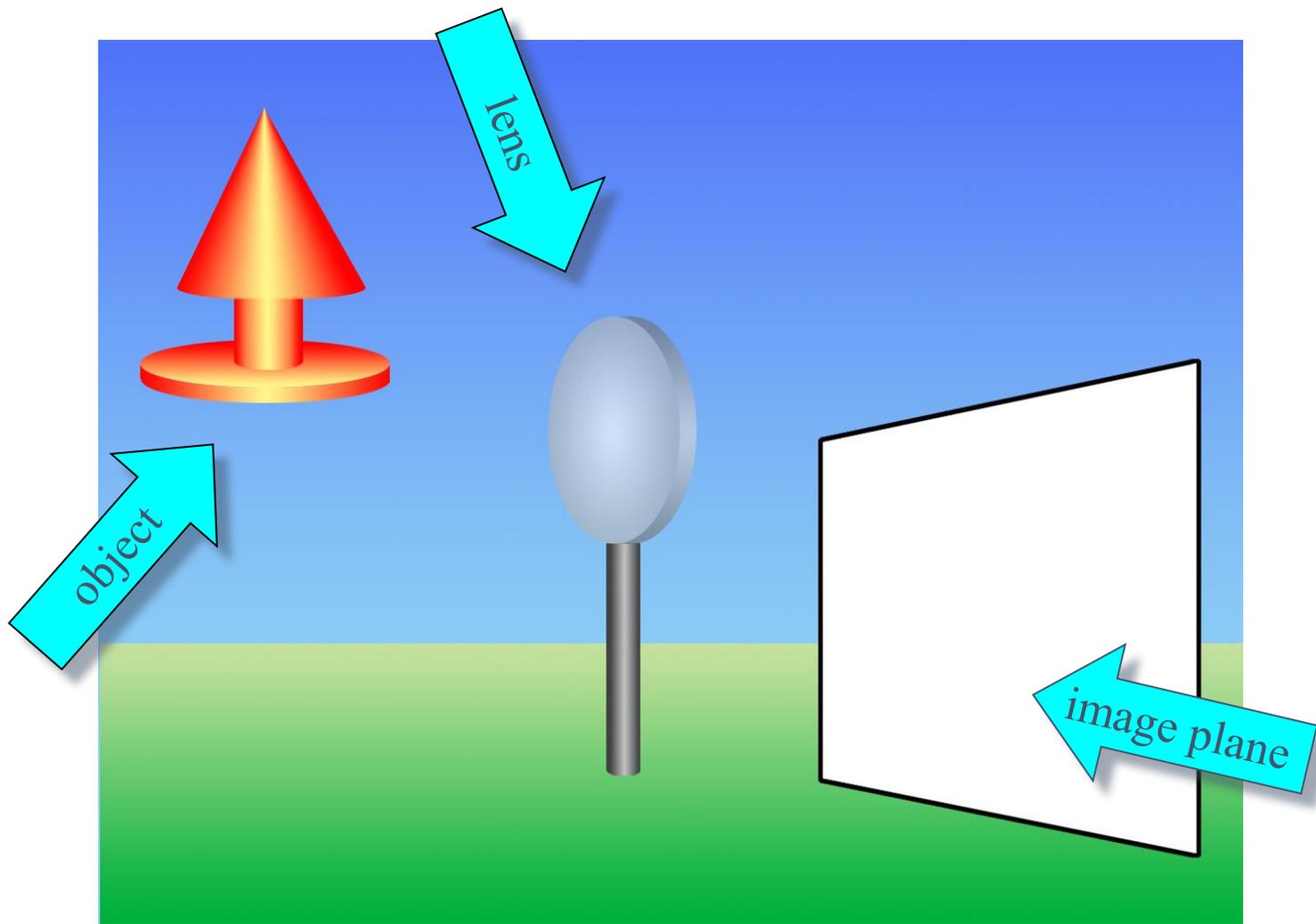


# 光栅显示器/数字图像的数学模型 (栅格图、点阵图)

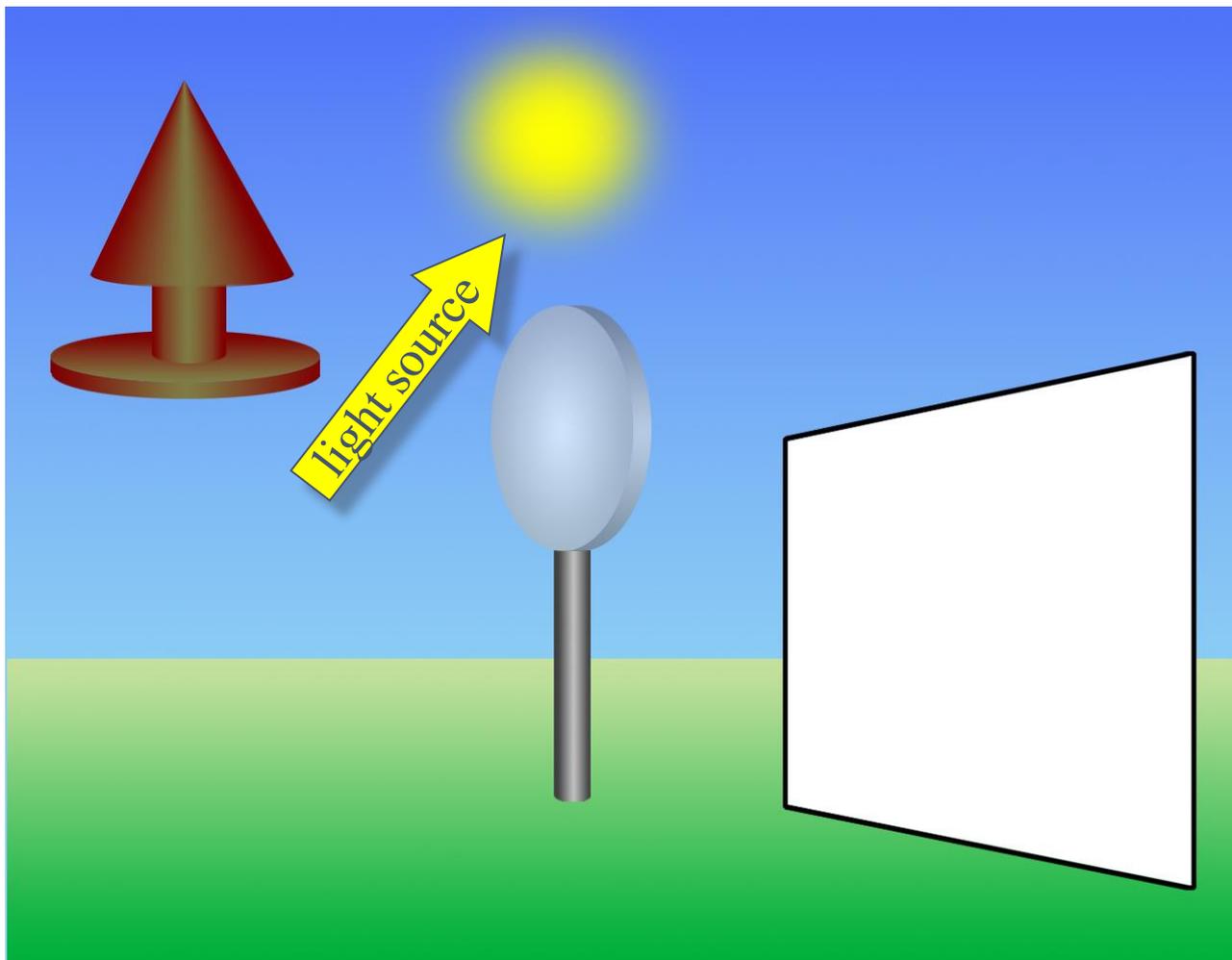
- 像素构成的**矩阵**：连续空间的离散**采样**



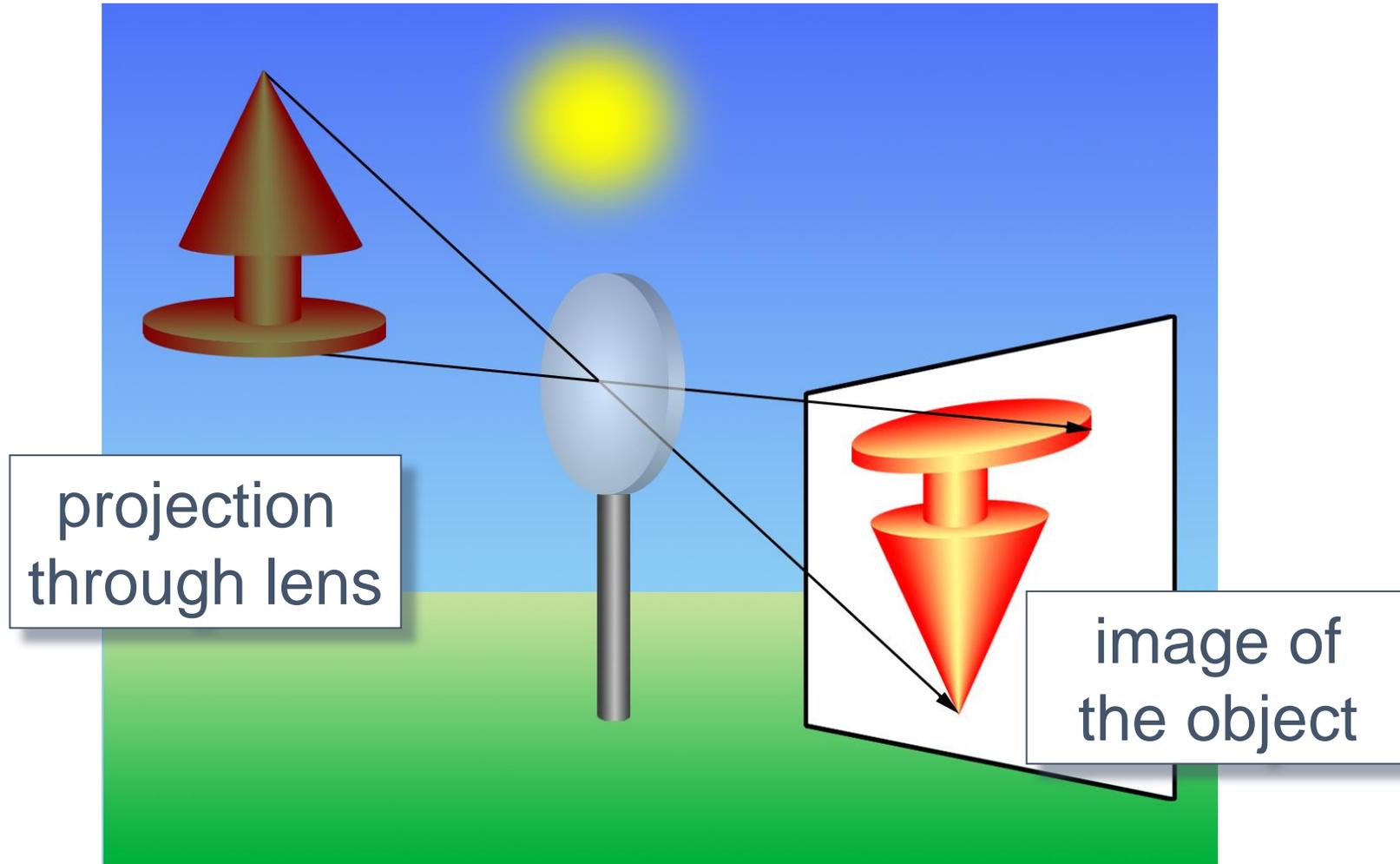
# 成像原理



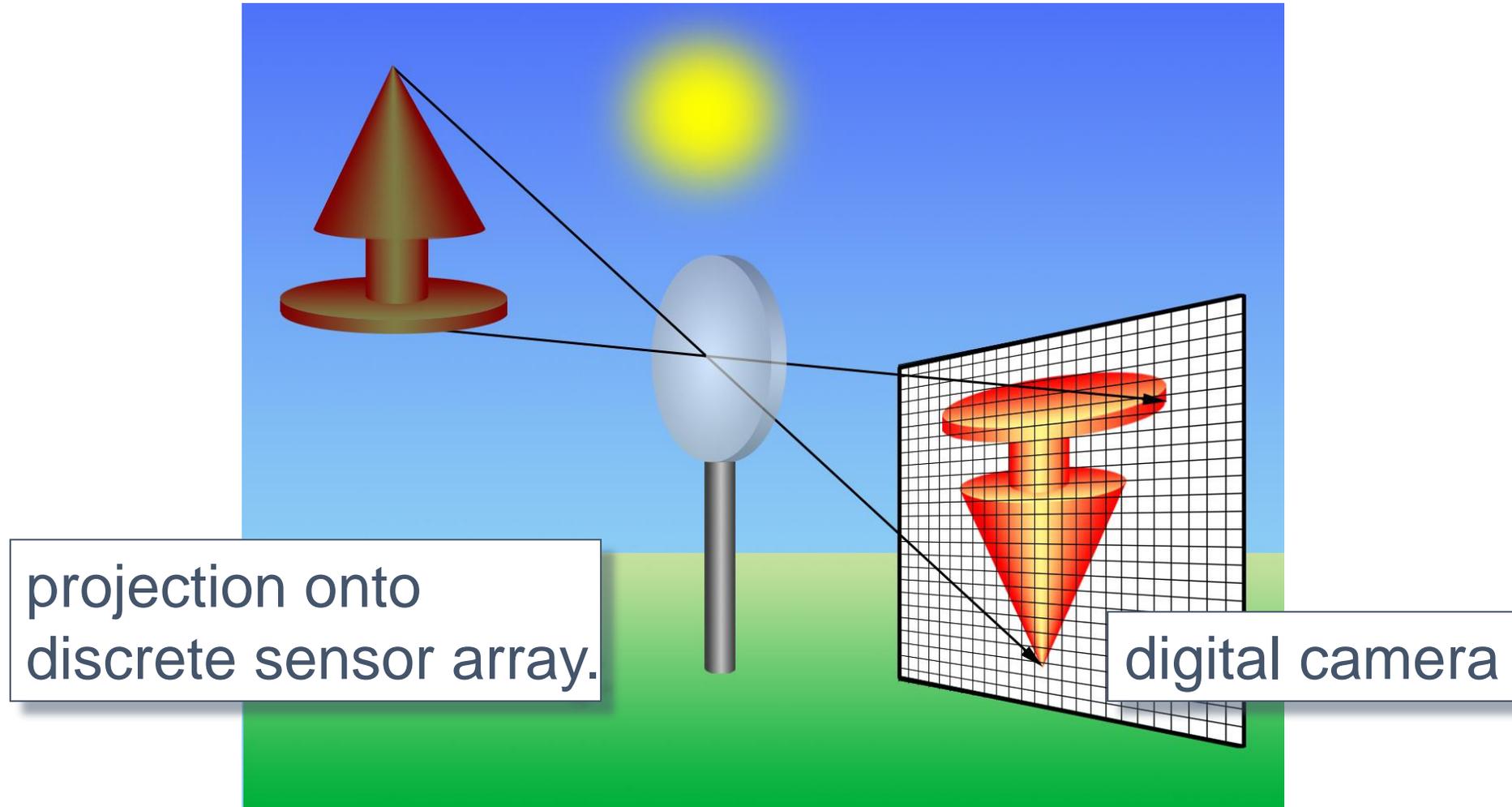
# 成像原理



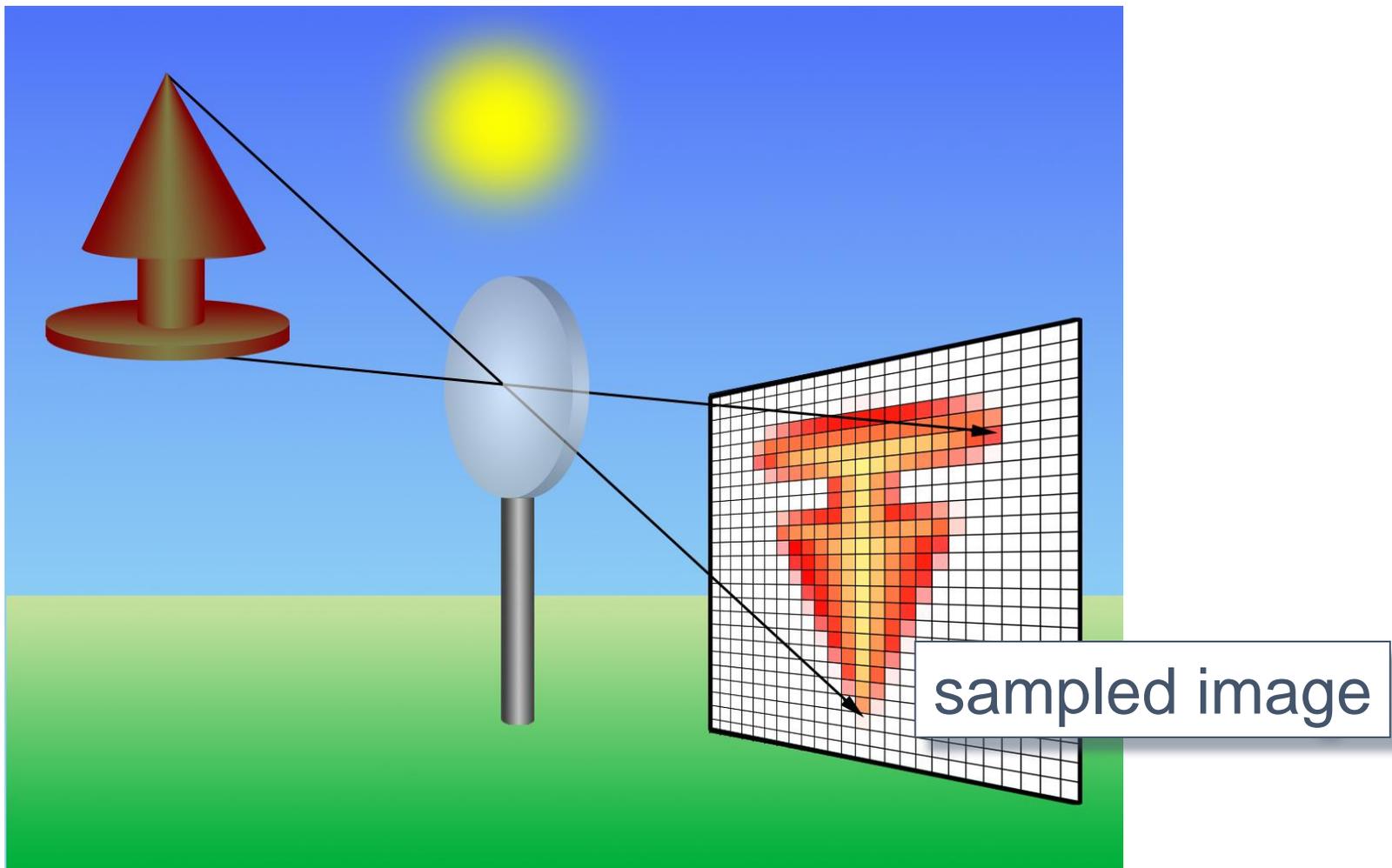
# 成像原理



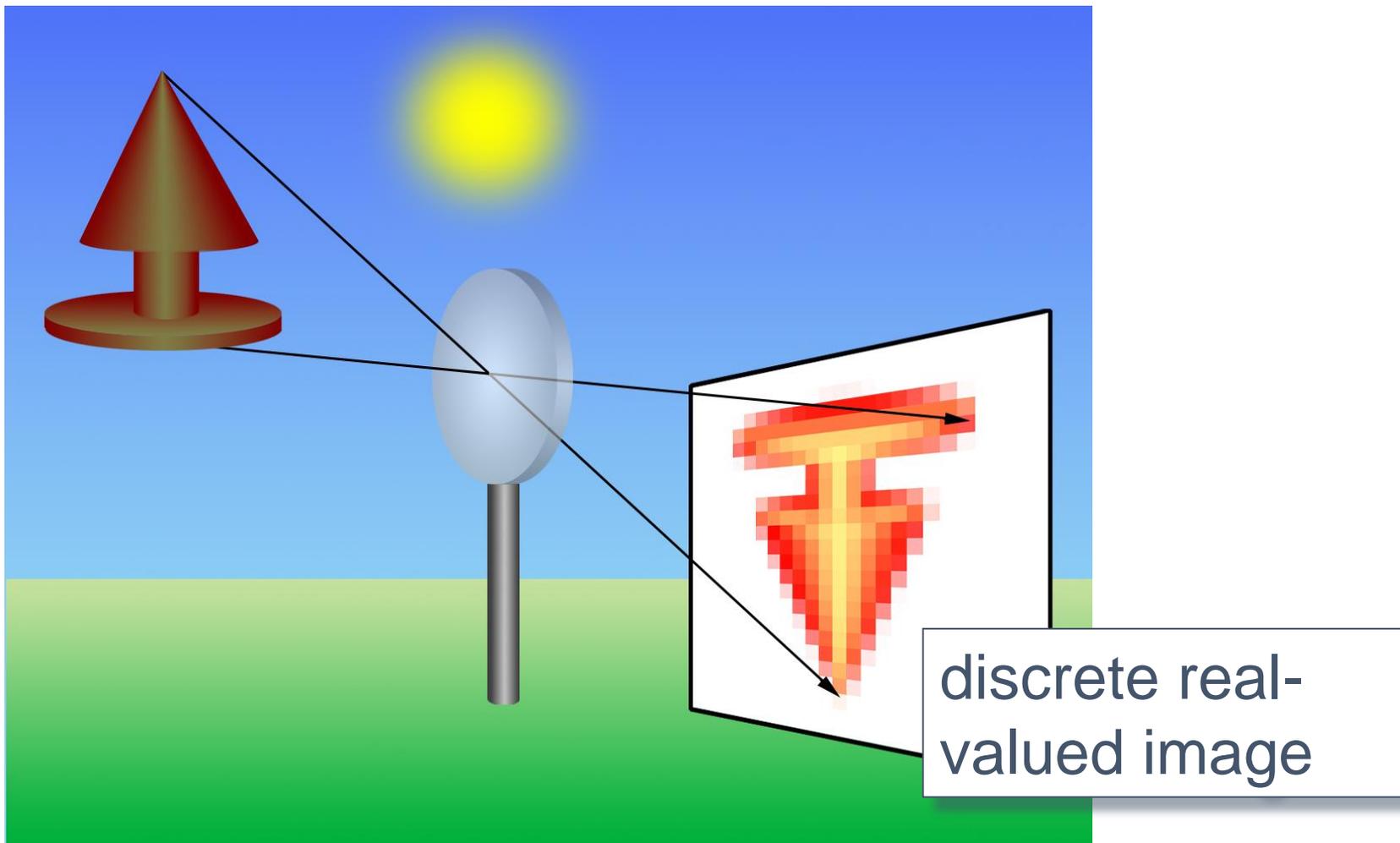
# 成像原理



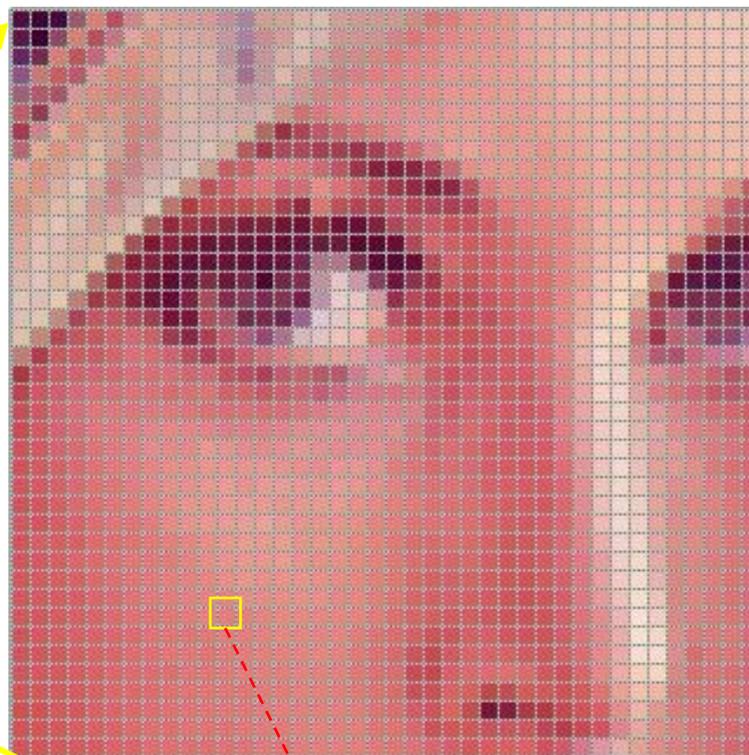
# 成像原理



# 成像原理



# 数字图像



$$0.6 R + 0.3 G + 0.1 B$$

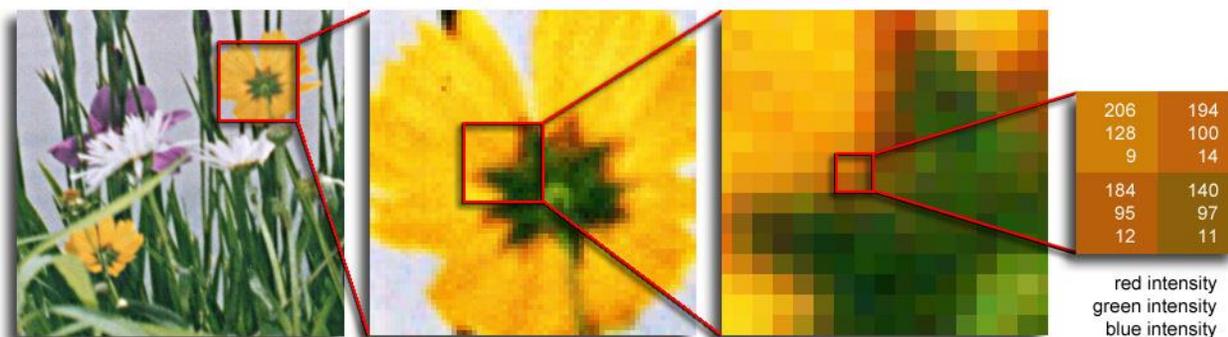


RGB 三原色

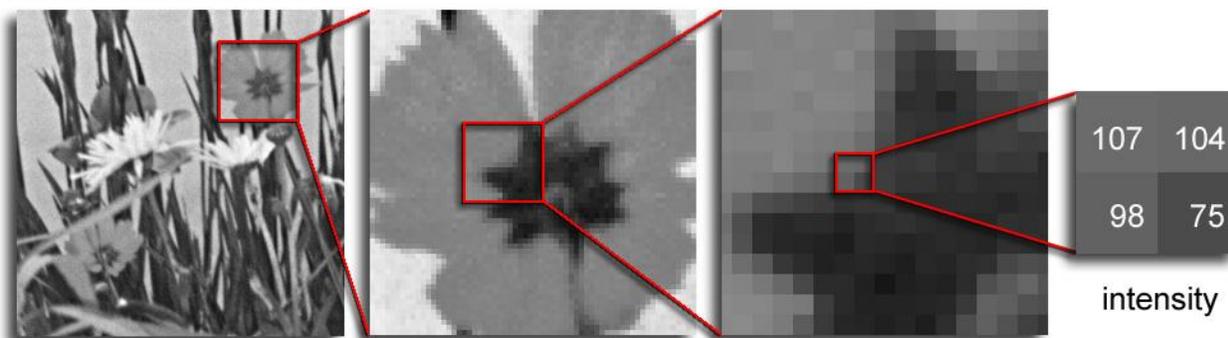
# 数字图像

- 像素的方阵列
- 每个元素称为像素， 存有一种颜色

彩色图像



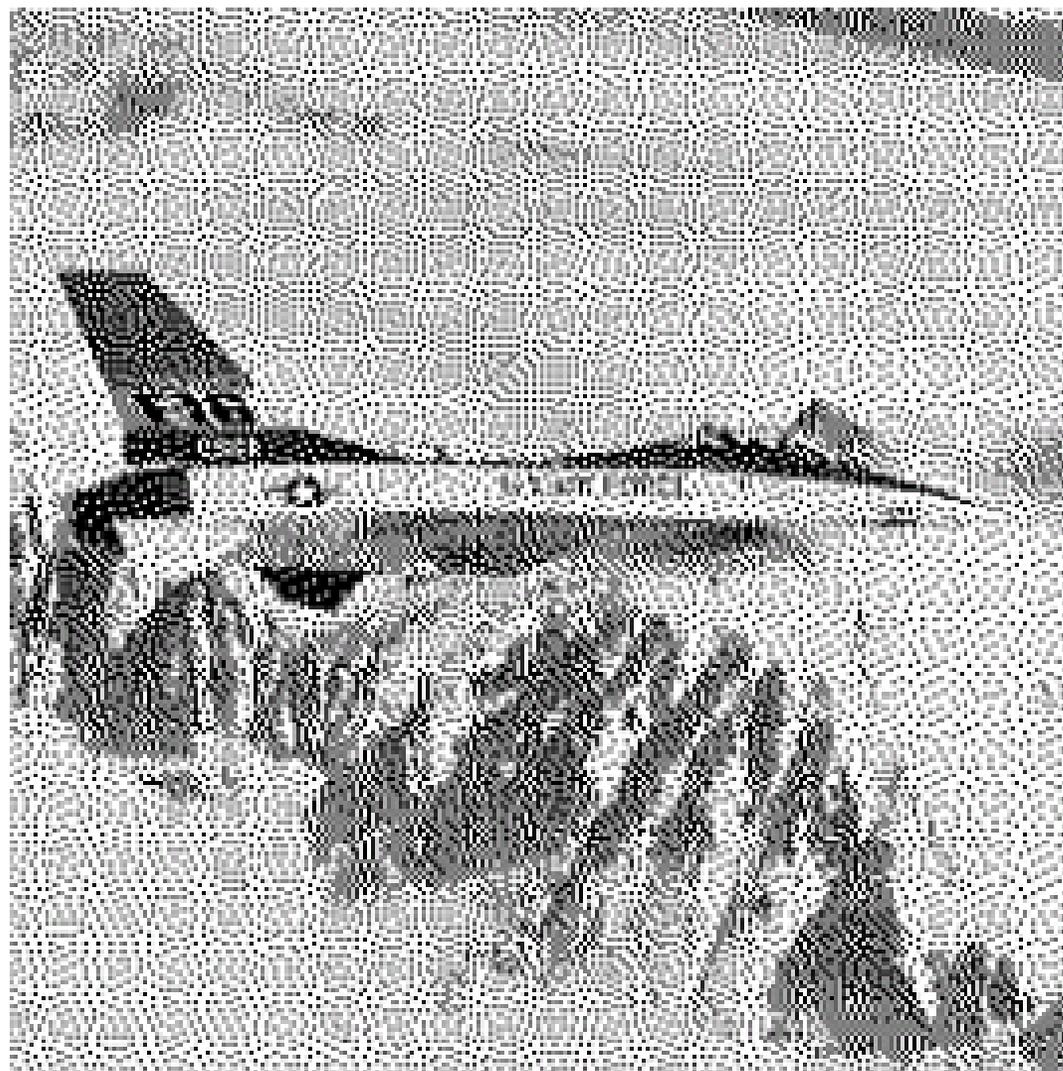
灰度图像



# 图像的种类

- 二值(Binary)图像 (0 or 1)
- 灰度(Gray)图像(0~255 or [0,1])
- 彩色(Color)图像
  - indexed color images
  - full color images (24 bits per pixel, 8-red, 8-green, 8-blue) )

# A Binary Image (二值图像)



# Gray Images (灰度图像)

- 8 bits per pixel



# Full Color Images (全彩图像)

- 每个像素24位 (bit)
  - R、G、B3个通道分别为灰度图像
  - 每个通道用8位(1个字节)编码
- 32位图像
  - (R, G, B, a)

# 颜色分量



# 图像文件格式

- BMP: bitmap
  - 无压缩, 文件大
- PNG
  - 压缩(zip, 无损), 可带alpha通道
- GIF
  - Lossy(有损压缩), 文件小, 可存储动画
- JPEG
  - 压缩率高
- JPEG 2000
  - 更高压缩率, 支持渐近(progressive)传输
- webp
- PSD...

量化与压缩

# 视频: 图像序列



# 图像的连续数学模型：平面区域上的向量值函数

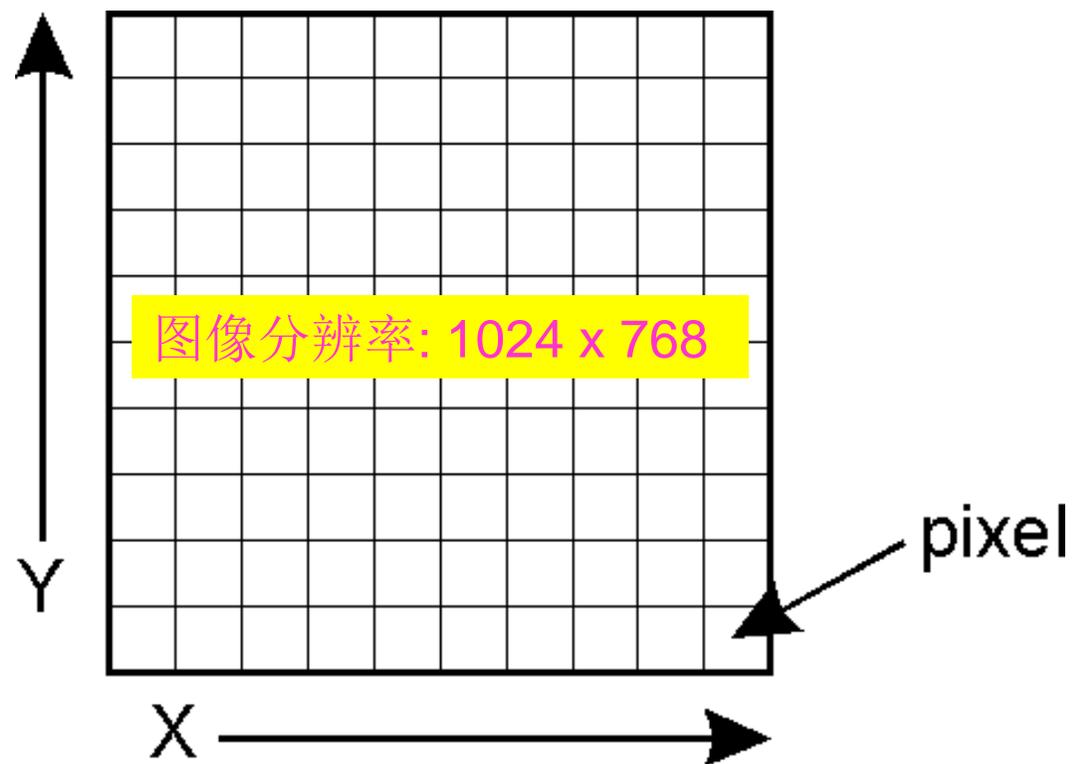
- 区域内每个点有个颜色值
  - 每种颜色值是红、绿、蓝三个成分的线性组合
  - 颜色空间是三维线性空间



$$0.6R + 0.3G + 0.1B$$

# 图像的离散数学模型：矩阵

- 分辨率



# 图像编程

# 面向对象编程

- 抽象
- 对象化
- 接口

# 图像的抽象

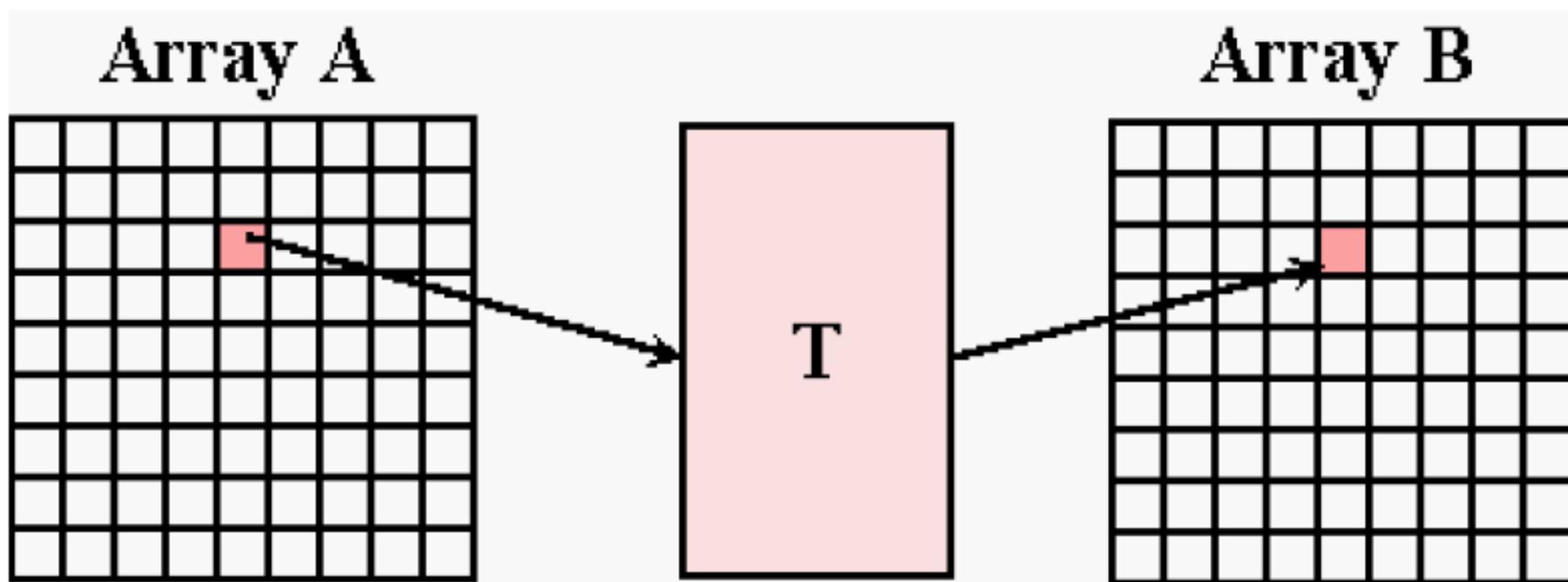
- 我们需要哪些信息可以决定一幅图像？
  - 宽、高
  - 每个元素的颜色
  
- 像素的矩阵!

# 图像对象的数据结构

```
class Image
{
public:
    int width() const;
    int height() const;
    color at(int x, int y) const;
    color& at(int x, int y);
    // void setPixel(int x, int y, color c);
private:
    ...
};
```

# 图像处理 (图像变换)

$$B[x, y] = T[A[x, y]]$$



# Examples

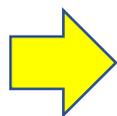
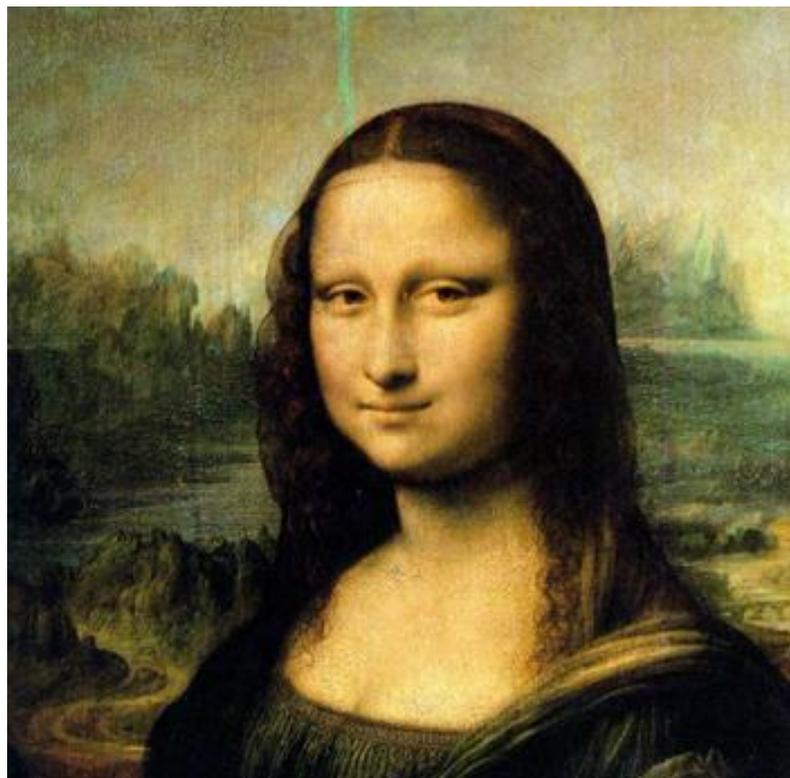
- Reverse color
  - $R = 255 - R$
- R, G, B channels
- Color2Gray
  - $\text{Gray} = a R + b G + c B$
  - $\text{Gray} = R*0.299 + G*0.587 + B*0.114$
- Add noise
- ...

如何实现各种图像处理算法？

# 例1：图像变形

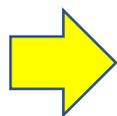
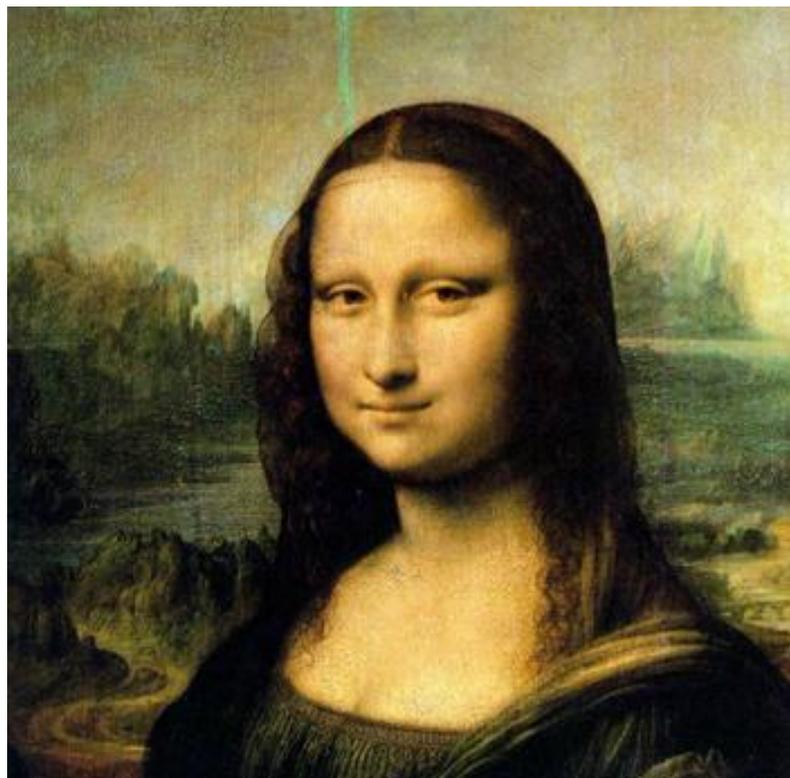
# 图像变形

- 特定效果的变形

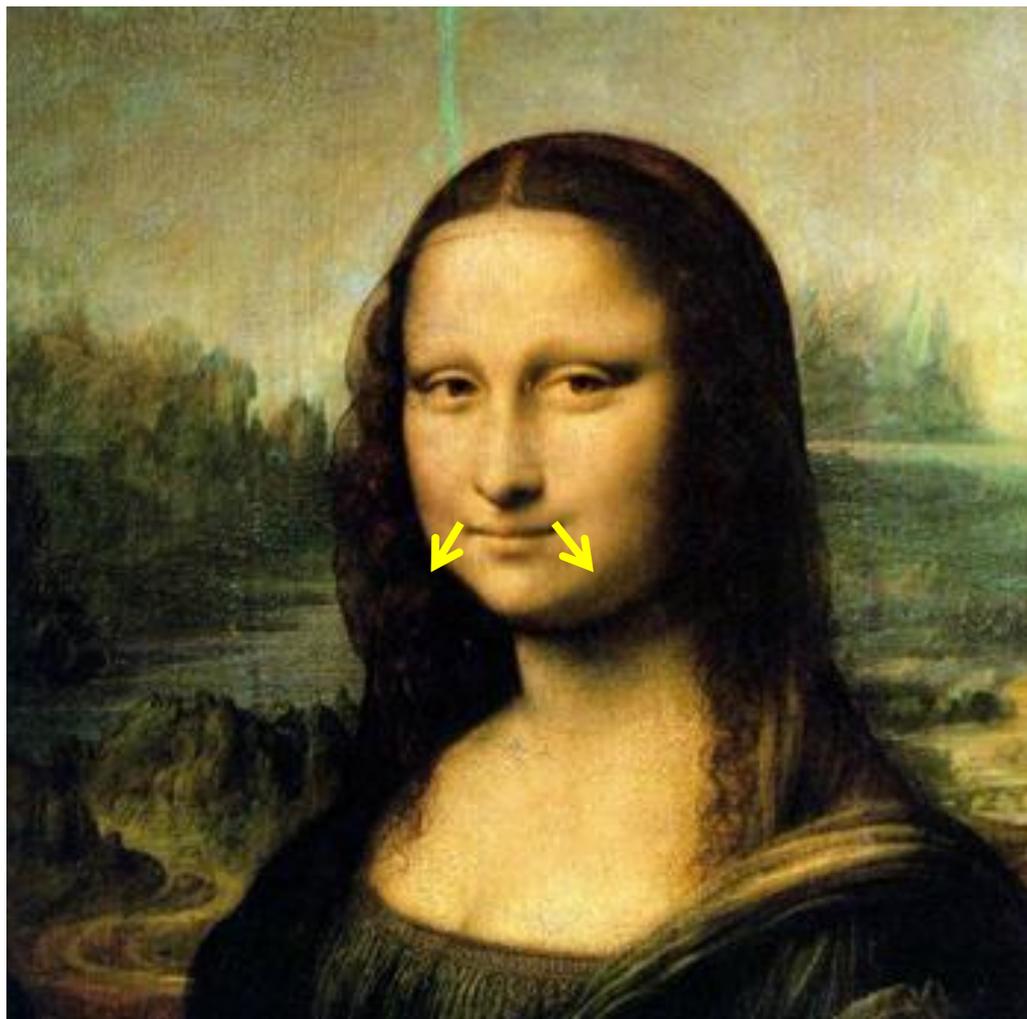


# 图像变形

- 特定效果的变形

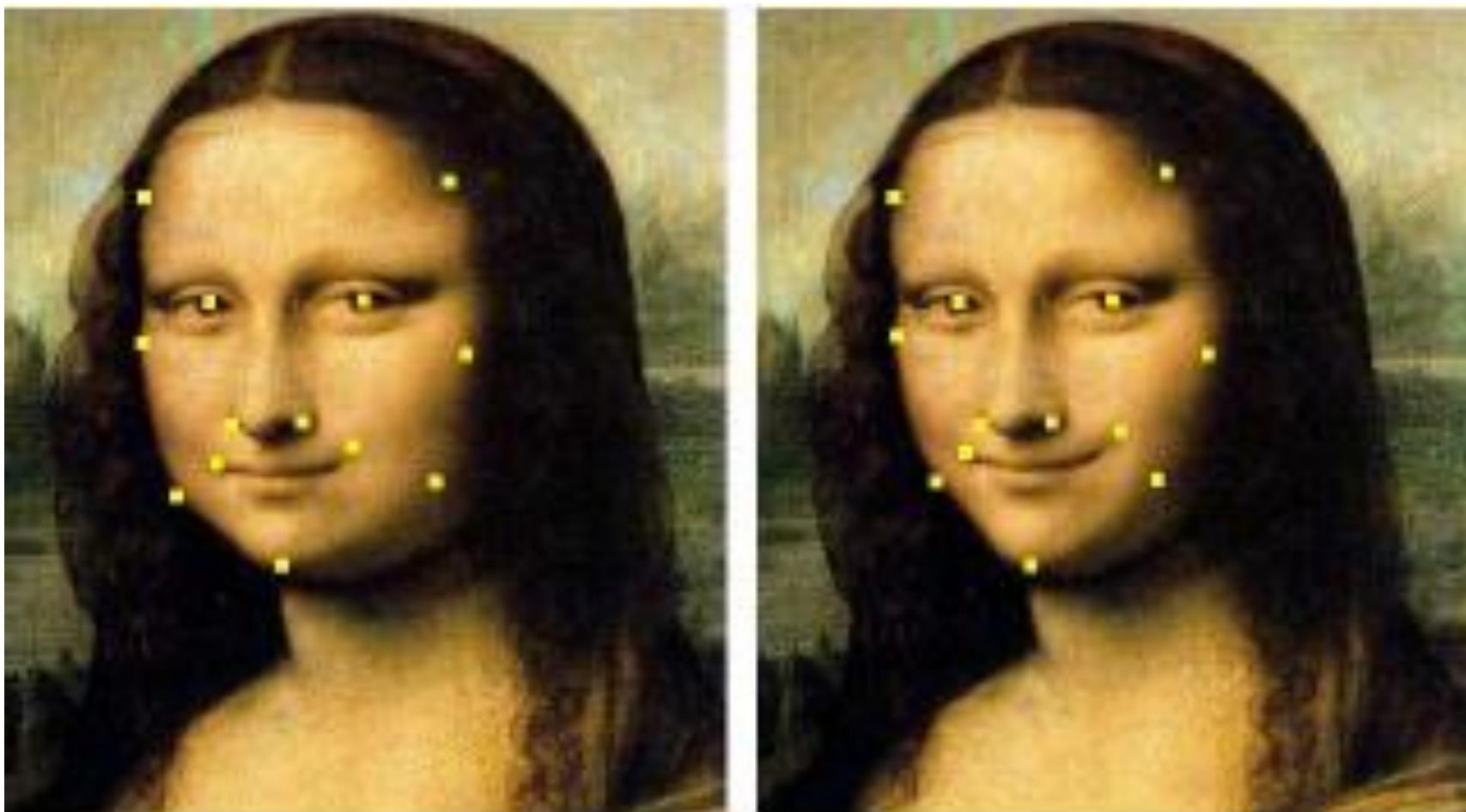


# 用户交互

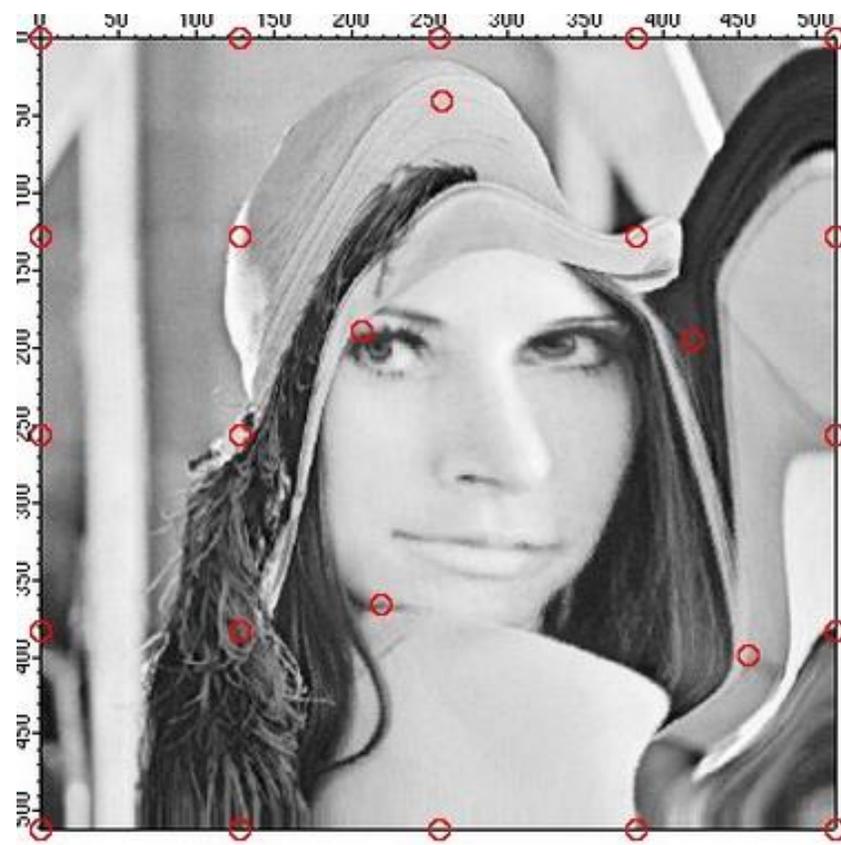
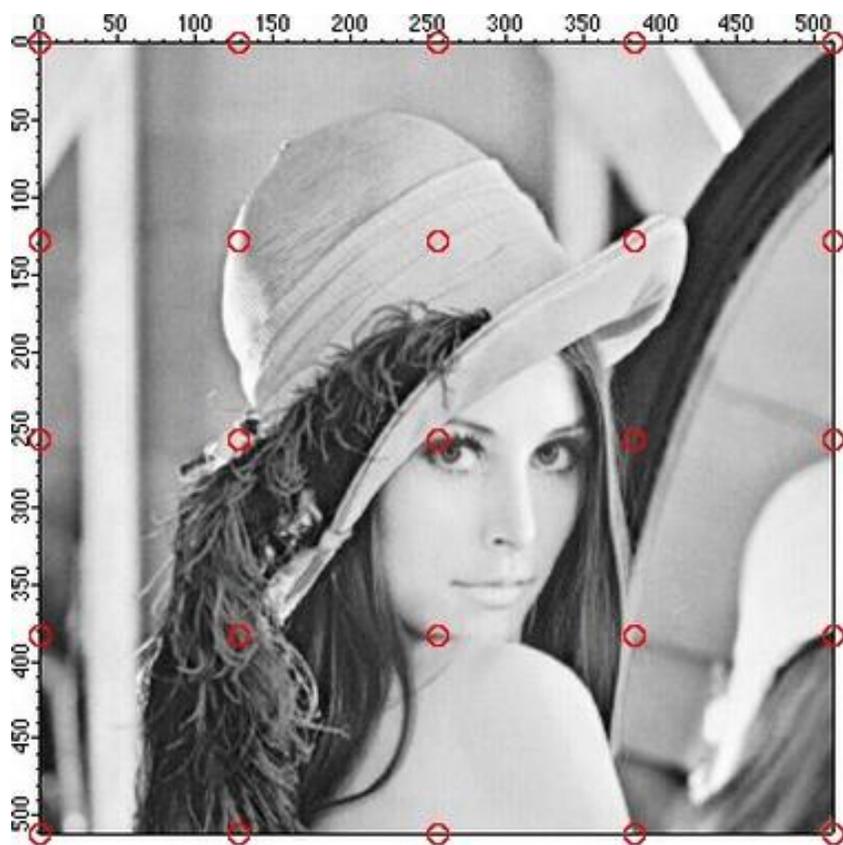


[Demo](#)

# 交互 (顶点偏移)



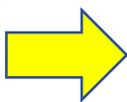
# 如何做到的？



## 例2：扣图

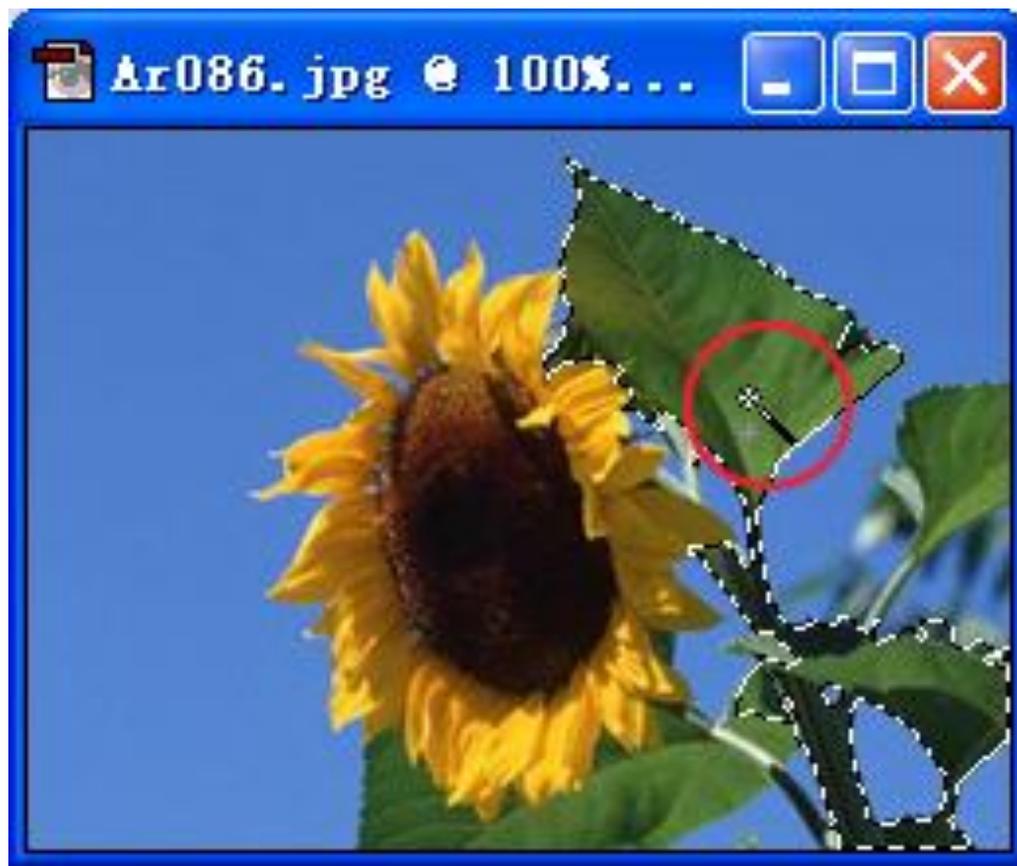
# 扣图

- 将物体从图像中分割出来



# 扣图

- Photoshop的工具： 魔术棒
  - 用户交互指定边界
  - 繁琐， 费时



# 扣图

- 智能做法：用户随意指定物体的大致内容



[Demo](#)

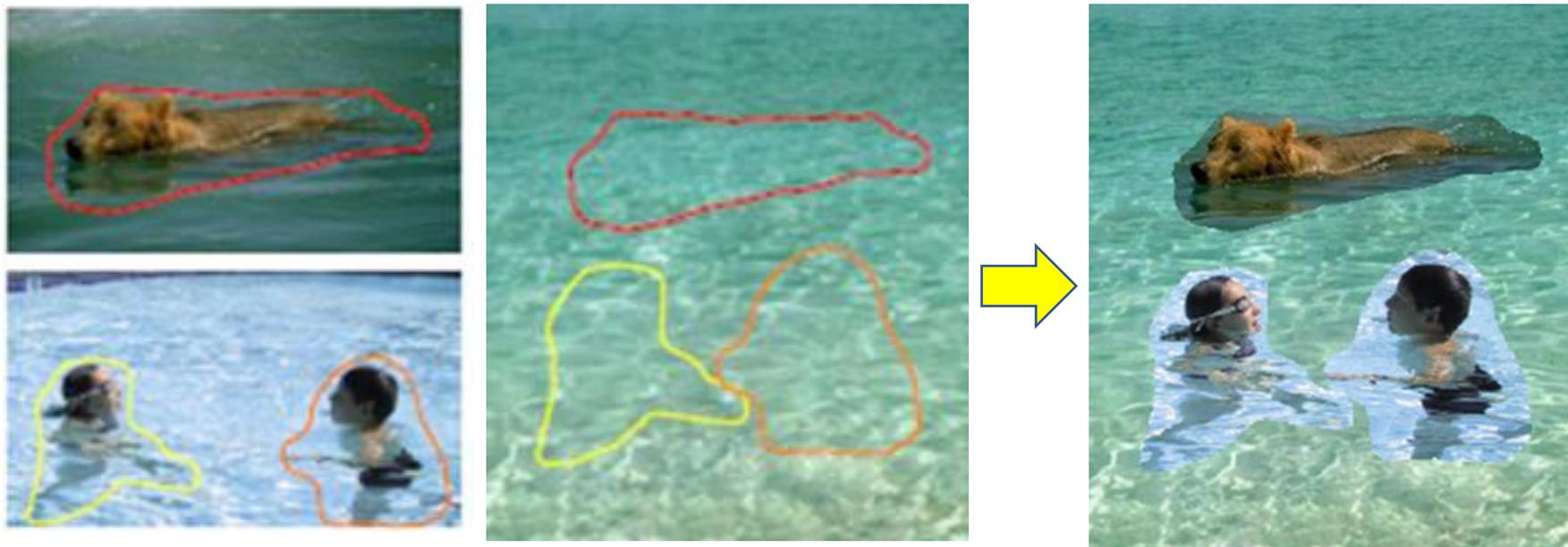
# 如何做到的？



# 例3：图像移植

# 图像移植

- 能否做到无缝移植?



Demo

# 如何做到的?



sources/destinations



cloning



seamless cloning

# 各种图像处理软件

- Photoshop (PS)
- CorelDraw
- Picasa
- 美图秀秀
- 光影魔术手
- ...

*Thank you!*

*Questions?*