中国科学技术大学
University of Science and Technology of China

数学建模
Mathematical Modeling

陈仁杰
中国科学技术大学

2. 抽象

方法论

- 抽象
  - 简化非本质的因素
  - 透过现象看本质
- 迭代
  - 将问题转化为已有问题
  - 组合、递归
- 执行方法
  - 分治法Divide-and-conquer
  - Coarse-to-fine

第一性原理

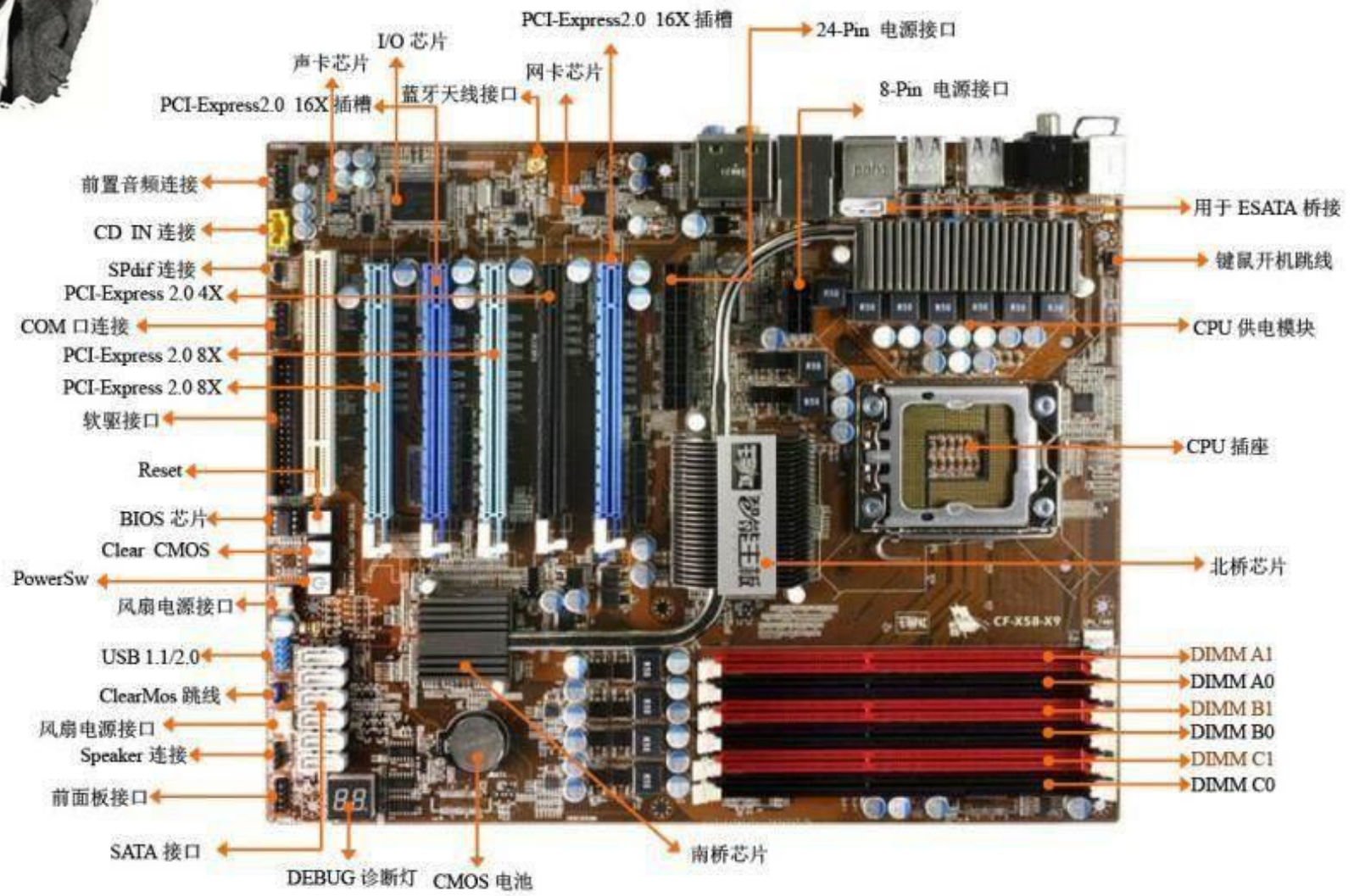
- 第一性原理：就是看透事物本质的根本方法
- 最早由古希腊哲学家亚里士多提出：“在每个系统探索中都存在第一性原理。第一性原理是基本的命题和假设，不能被省略和删除，也不能被违反”
- 大道至简：本质、道、底层逻辑
- 举例
  - 几何学的第一性原理
  - 物理学的第一性原理
  - 人际交往的第一性原理
  - ...

(1) 计算机系统

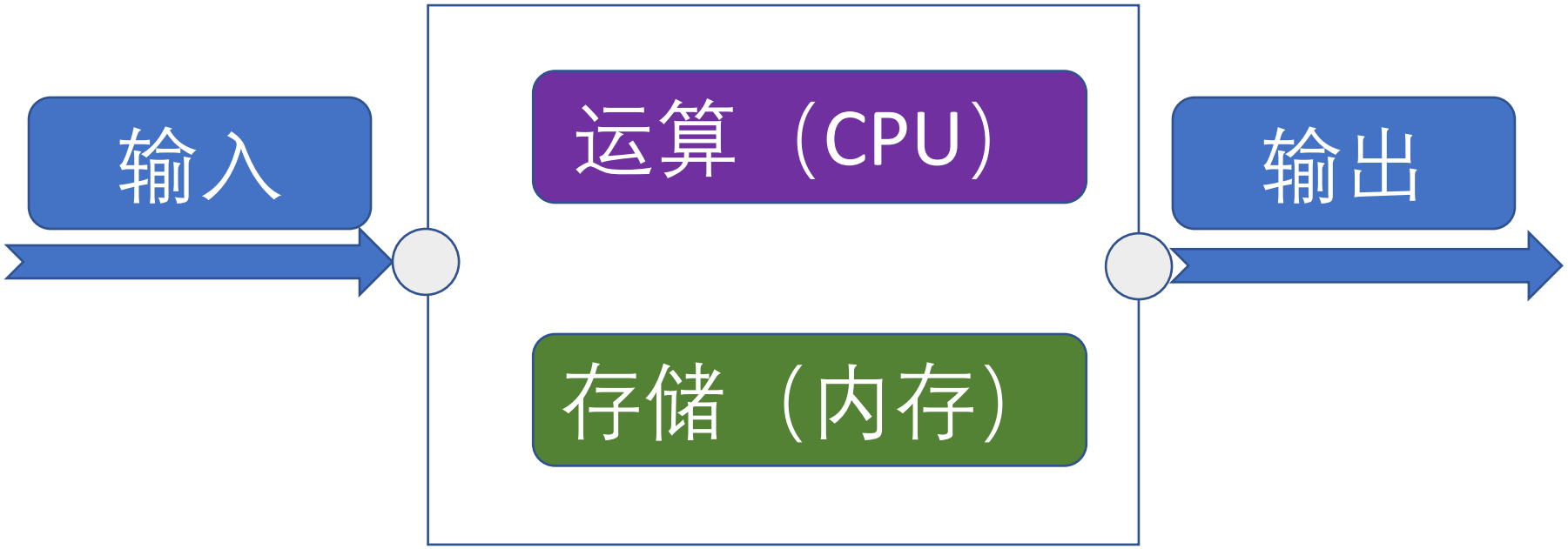
什么是计算机?

- 功能需求：解决一切可以用“计算”来解决的问题

冯.诺依曼计算机的体系结构

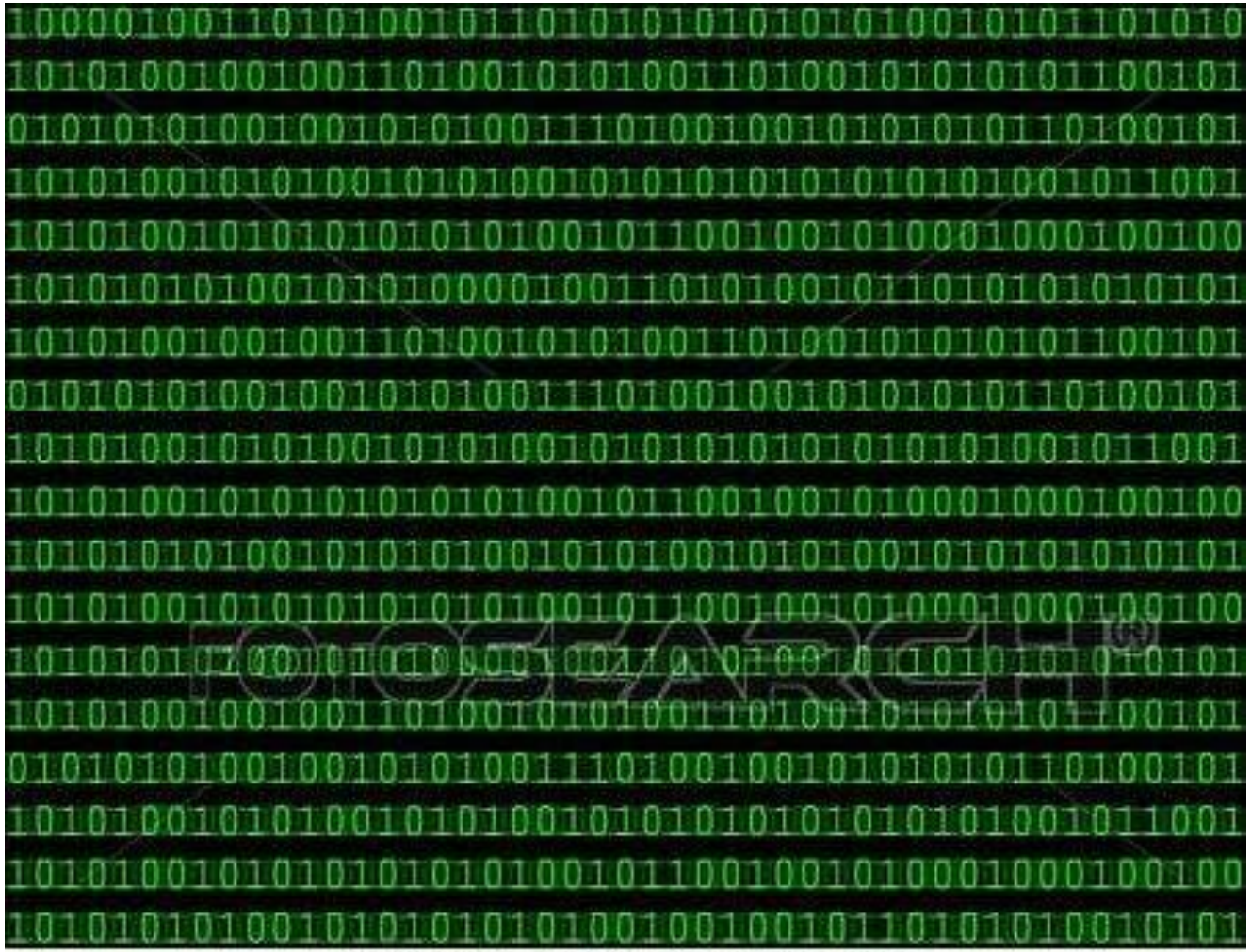


计算机结构的抽象



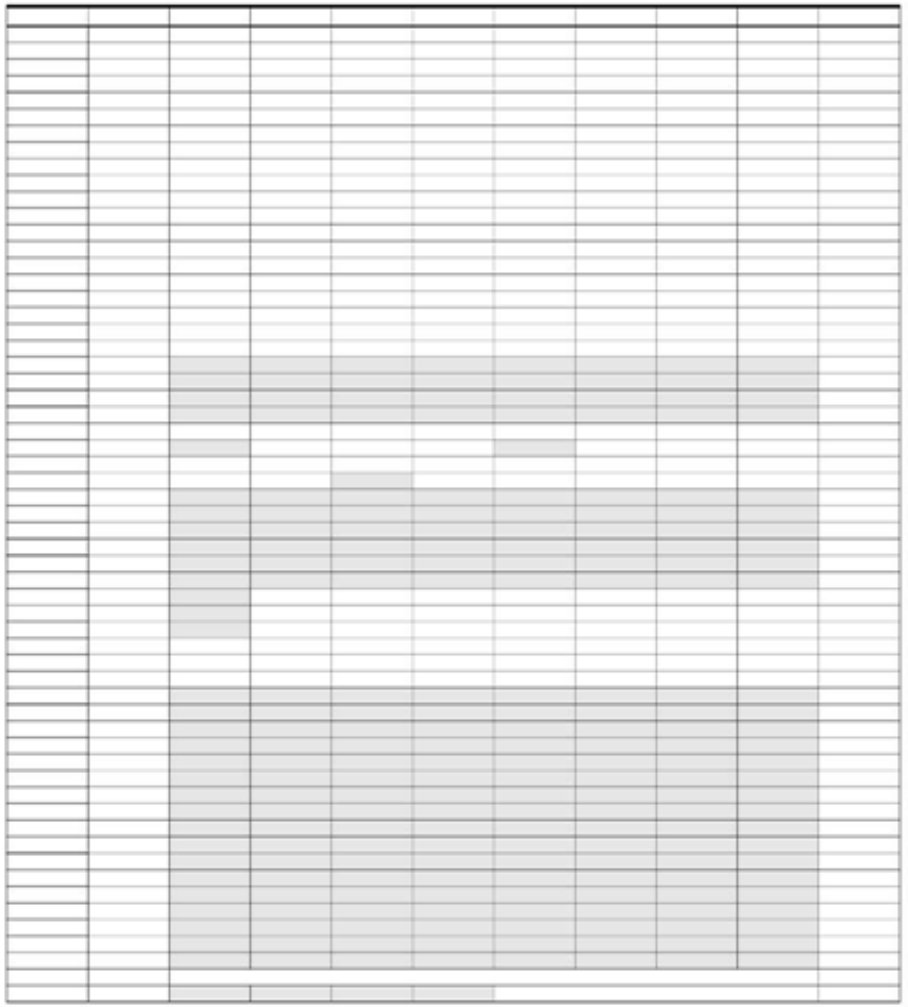
存储器：内存

- bit（位）：表示两个状态0/1（开/关）



存储器：内存

- byte（字节）：8 bits，表示256个状态



```
          0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00000000h: CA FE BA BE 00 00 00 33 00 22 0A 00 06 00 14 09 ; Êþº¾...3."......
00000010h: 00 15 00 16 08 00 17 0A 00 18 00 19 07 00 1A 07 ; ................
00000020h: 00 1B 01 00 06 3C 69 6E 69 74 3E 01 00 03 28 29 ; .....<init>...()
00000030h: 56 01 00 04 43 6F 64 65 01 00 0F 4C 69 6E 65 4E ; V...Code...LineN
00000040h: 75 6D 62 65 72 54 61 62 6C 65 01 00 12 4C 6F 63 ; umberTable...Loc
00000050h: 61 6C 56 61 72 69 61 62 6C 65 54 61 62 6C 65 01 ; alVariableTable.
00000060h: 00 04 74 68 69 73 01 00 1F 4C 63 6F 6D 2F 70 61 ; ..this...Lcom/pa
00000070h: 64 64 78 2F 74 65 73 74 2F 61 73 6D 2F 48 65 6C ; ddx/test/asm/Hel
00000080h: 6C 6F 57 6F 72 6C 64 3B 01 00 04 6D 61 69 6E 01 ; loWorld;...main.
00000090h: 00 16 28 5B 4C 6A 61 76 61 2F 6C 61 6E 67 2F 53 ; ..([Ljava/lang/S
000000a0h: 74 72 69 6E 67 3B 29 56 01 00 04 61 72 67 73 01 ; tring;)V...args.
000000b0h: 00 13 5B 4C 6A 61 76 61 2F 6C 61 6E 67 2F 53 74 ; ..[Ljava/lang/St
000000c0h: 72 69 6E 67 3B 01 00 0A 53 6F 75 72 63 65 46 69 ; ring;...SourceFi
000000d0h: 6C 65 01 00 0F 48 65 6C 6C 6F 57 6F 72 6C 64 2E ; le...HelloWorld.
000000e0h: 6A 61 76 61 0C 00 07 00 08 07 00 1C 0C 00 1D 00 ; java............
000000f0h: 1E 01 00 0C 48 65 6C 6C 6F 2C 57 6F 72 6C 64 21 ; ....Hello,World!
00000100h: 07 00 1F 0C 00 20 00 21 01 00 1D 63 6F 6D 2F 70 ; ..... .!...com/p
00000110h: 61 64 64 78 2F 74 65 73 74 2F 61 73 6D 2F 48 65 ; addx/test/asm/He
00000120h: 6C 6C 6F 57 6F 72 6C 64 01 00 10 6A 61 76 61 2F ; lloWorld...java/
00000130h: 6C 61 6E 67 2F 4F 62 6A 65 63 74 01 00 10 6A 61 ; lang/Object...ja
00000140h: 76 61 2F 6C 61 6E 67 2F 53 79 73 74 65 6D 01 00 ; va/lang/System..
00000150h: 03 6F 75 74 01 00 15 4C 6A 61 76 61 2F 69 6F 2F ; .out...Ljava/io/
00000160h: 50 72 69 6E 74 53 74 72 65 61 6D 3B 01 00 13 6A ; PrintStream;...j
00000170h: 61 76 61 2F 69 6F 2F 50 72 69 6E 74 53 74 72 65 ; ava/io/PrintStre
00000180h: 61 6D 01 00 07 70 72 69 6E 74 6C 6E 01 00 15 28 ; am...println...(
00000190h: 4C 6A 61 76 61 2F 6C 61 6E 67 2F 53 74 72 69 6E ; Ljava/lang/Strin
000001a0h: 67 3B 29 56 00 21 00 05 00 06 00 00 00 00 00 02 ; g;)V.!..........
000001b0h: 00 01 00 07 00 08 00 01 00 09 00 00 00 2F 00 01 ; ............./..
000001c0h: 00 01 00 00 00 05 2A B7 00 01 B1 00 00 00 02 00 ; ......*·..±.....
000001d0h: 0A 00 00 00 06 00 01 00 00 00 03 00 0B 00 00 00 ; ................
000001e0h: 0C 00 01 00 00 00 05 00 0C 00 0D 00 00 00 09 00 ; ................
000001f0h: 0E 00 0F 00 01 00 09 00 00 00 37 00 02 00 01 00 ; ..........7.....
00000200h: 00 00 09 B2 00 02 12 03 B6 00 04 B1 00 00 00 02 ; ...²....¶..±....
00000210h: 00 0A 00 00 00 0A 00 02 00 00 00 05 00 08 00 06 ; ................
00000220h: 00 0B 00 00 00 0C 00 01 00 00 00 09 00 10 00 11 ; ................
00000230h: 00 00 00 01 00 12 00 00 00 02 00 13             ; ............
```

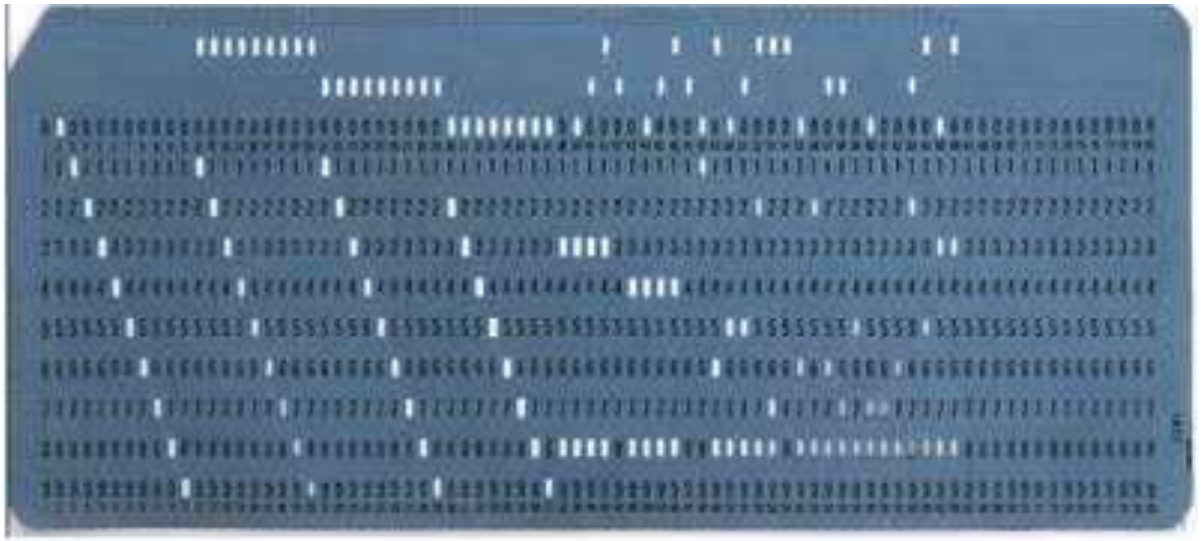
CPU（处理器）

- 负责**程序（指令序列）的执行**
  - 指令序列也是存放在存储里面
  - 计算机加电启动后，**CPU从一个固定的存储地址开始执行**
- CPU指令集
  - 计算类：各类基本数学运算，如加减乘除、sin/cos 等等
  - I/O 类：从输入输出设备（存储）读数据、写数据
  - 指令跳转类：在满足特定条件下跳转到新的当前程序执行位置

机器语言：用二进制和编码方式提供的指令系统编程程序

计算7+10并存储的程序

```
100001 10
00000111
100010 10
00001010
100101 11
00000110
111101 00
```

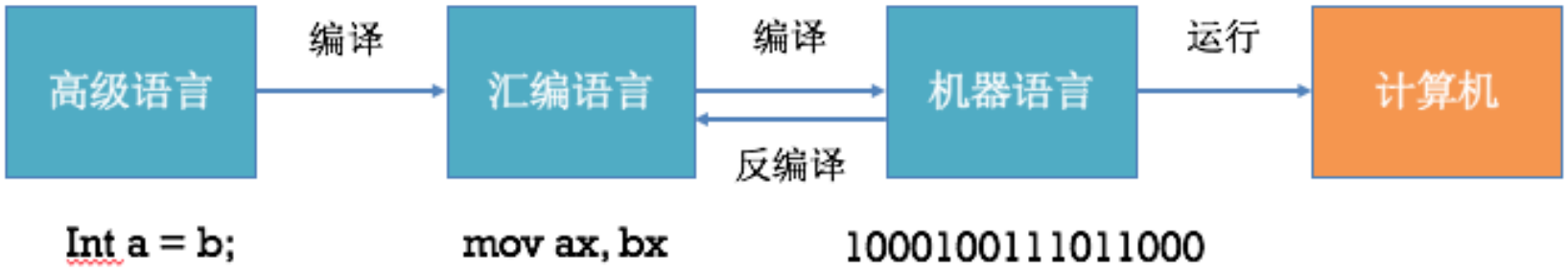


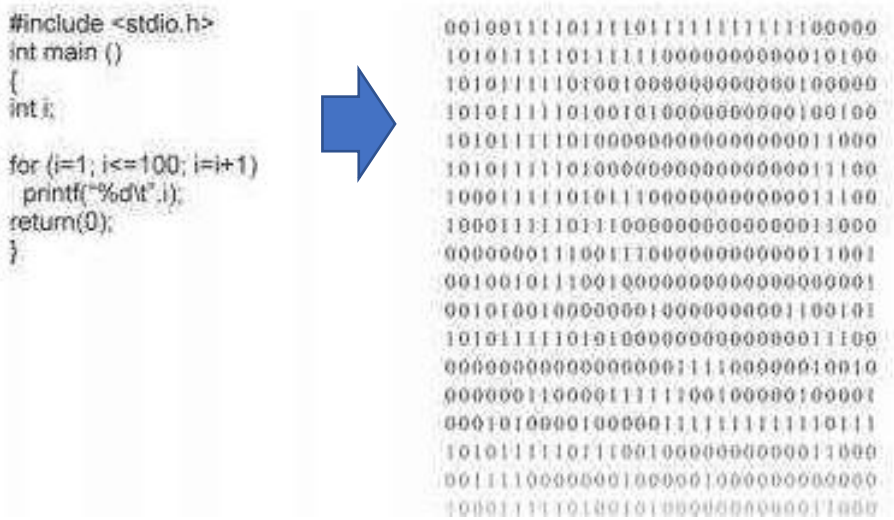
打孔卡

编程语言+编译器


机器语言
•由二进制代码构成
•计算机硬件可以识别
•可以表示简单的操作
•例如：加法、减法、数据移动等等
汇编语言
•将机器指令映射为一些助记符。如ADD、SUB、mov等；
•抽象层次低，需要考虑机器细节。
高级语言
•关键字、语句容易理解；
•有含义的数据命名和算式；
•抽象层次较高；例如，算式：a+b+c/d
•屏蔽了机器的细节；例如，这样显示计算结果：
cout<<a+b+c/d

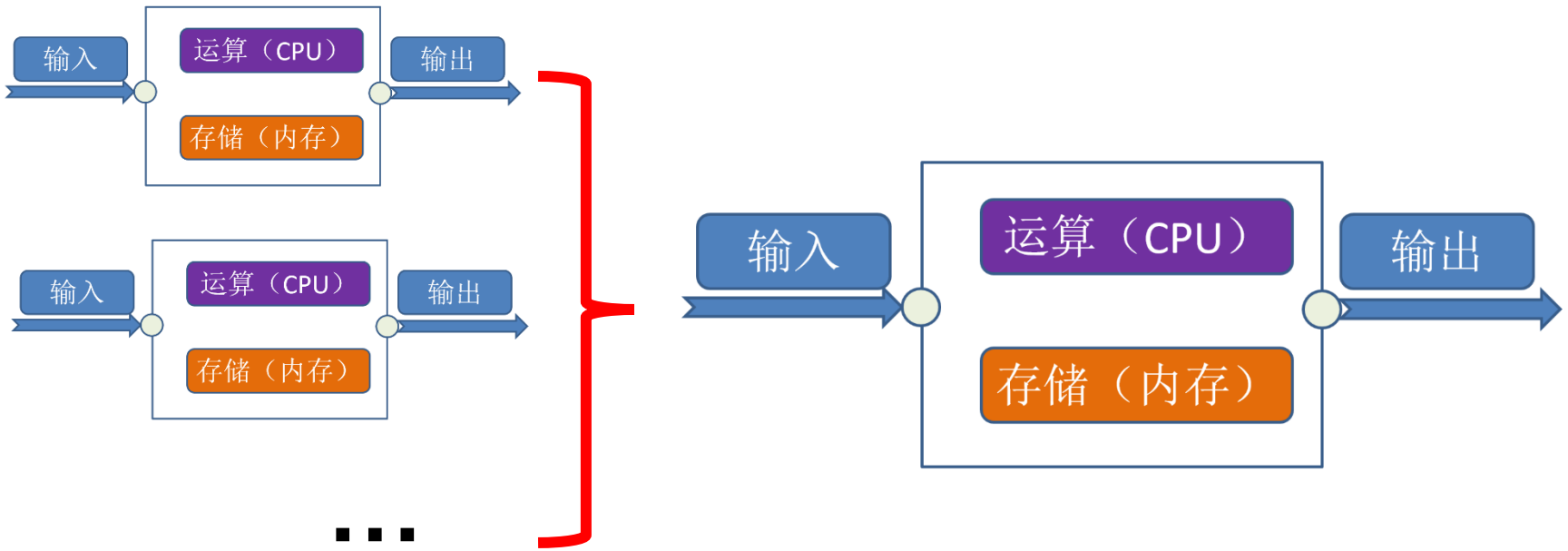
编程语言+编译器


高级语言
编译
汇编语言
编译
反编译
机器语言
运行
计算机
Int a = b;
mov ax, bx
1000100111011000


#include <stdio.h>
int main ()
{
int i;
for (i=1; i<=100; i=i+1)
printf("%d\t",i);
return(0);
}

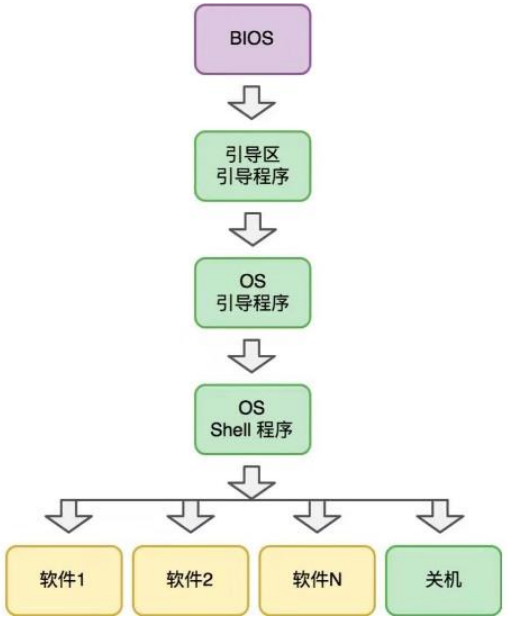
递归原则

- 1：中央处理器 + 存储：可以支持任意复杂的“计算” -- y=f(x)
- 2：输入输出设备：是电脑无限可能的扩展能力

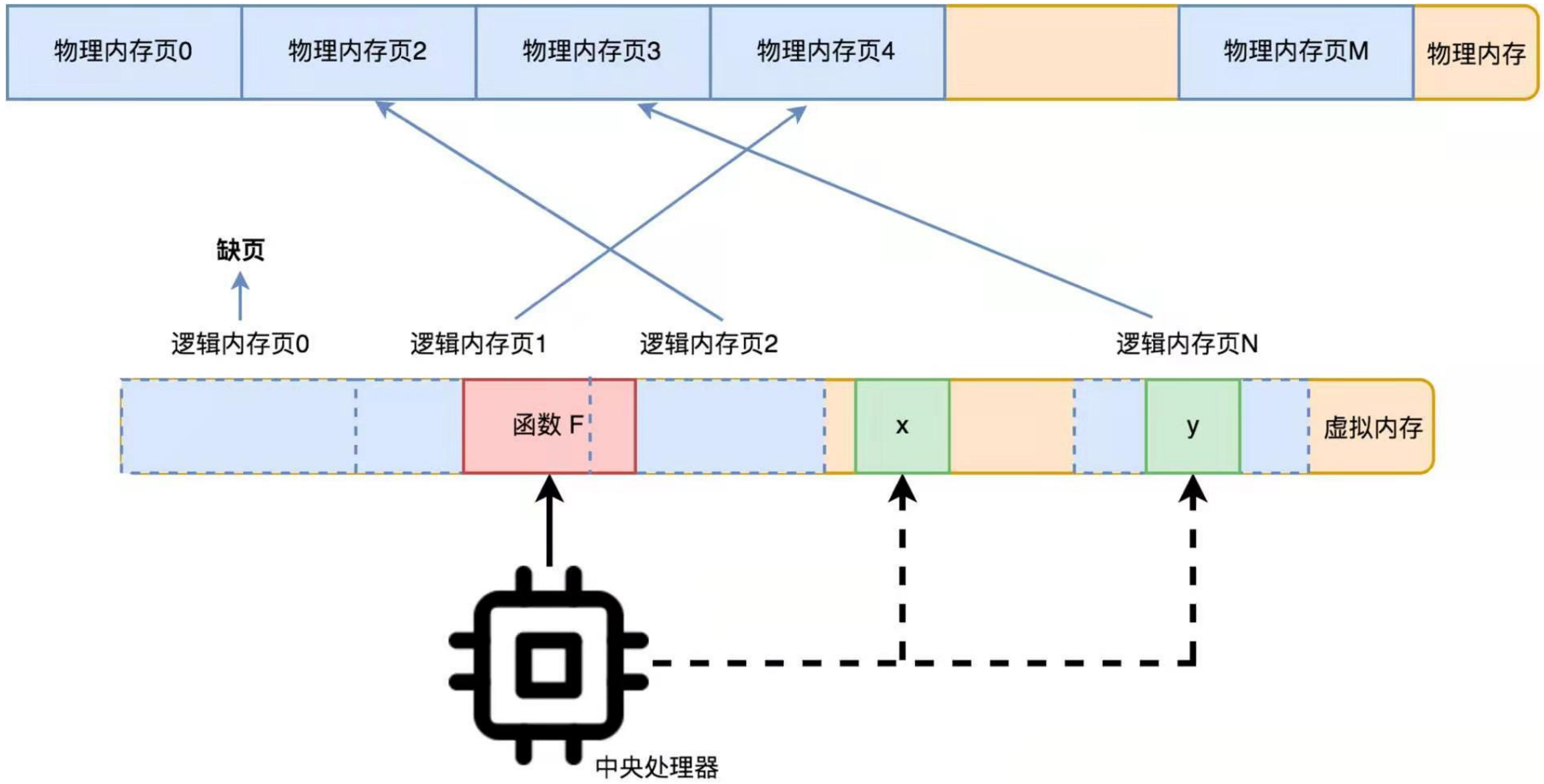


操作系统

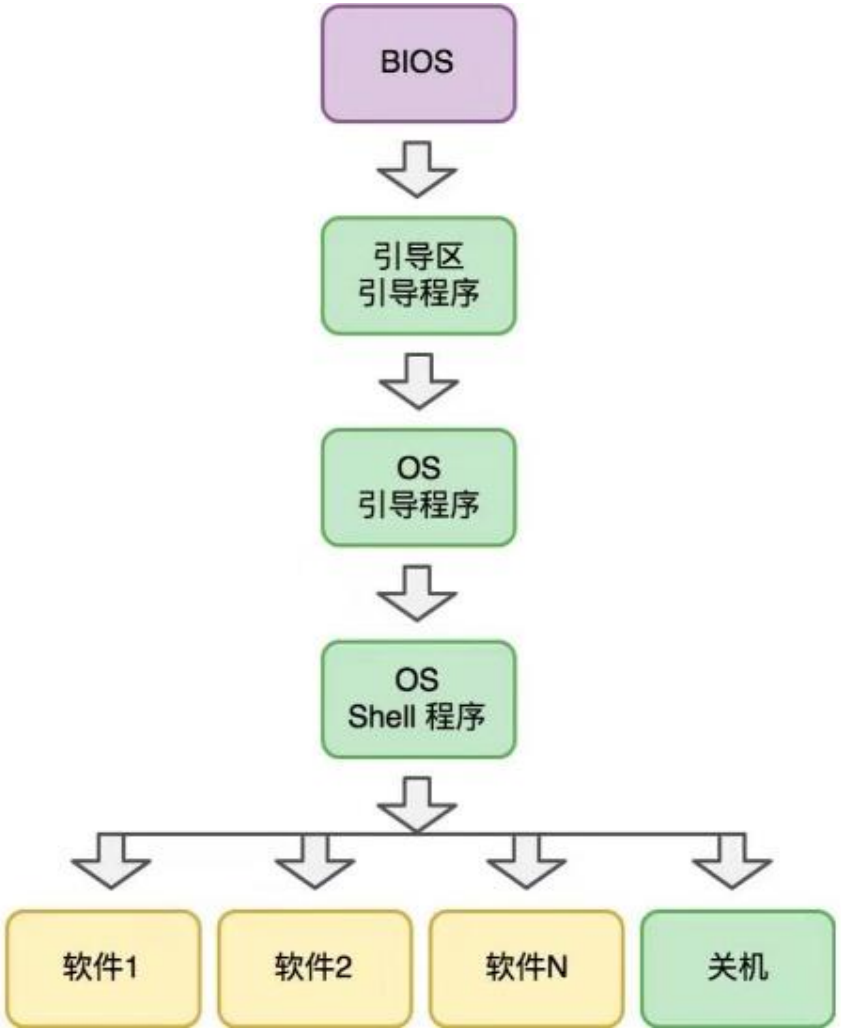
- 解决软件治理的问题
  - 进程与处理机管理、作业管理、存储管理、设备管理、文件管理
- 提供基础编程接口 (Windows API)
  - GDI: 文本、图像、画图
  - ...

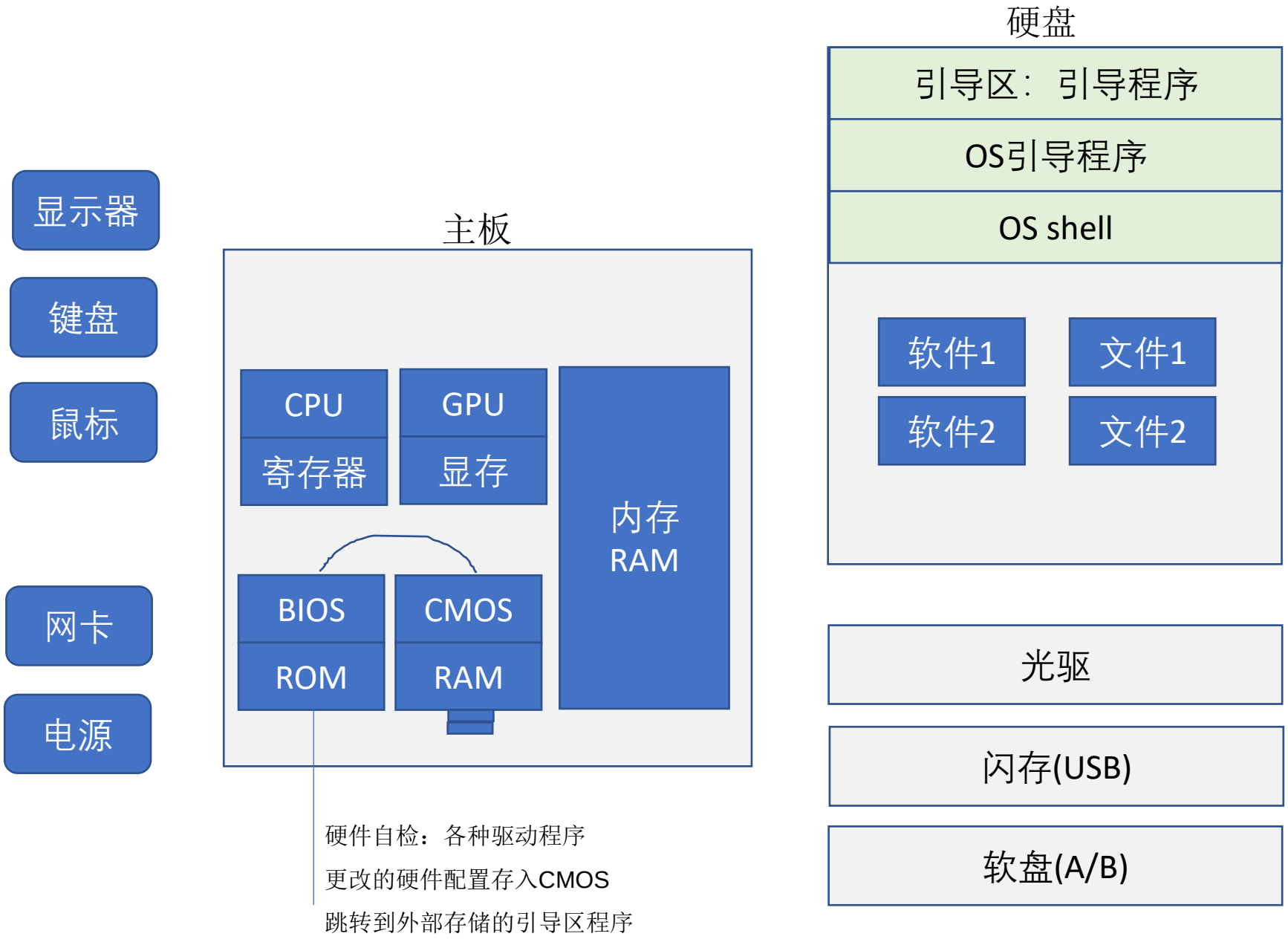


物理内存与虚拟内存

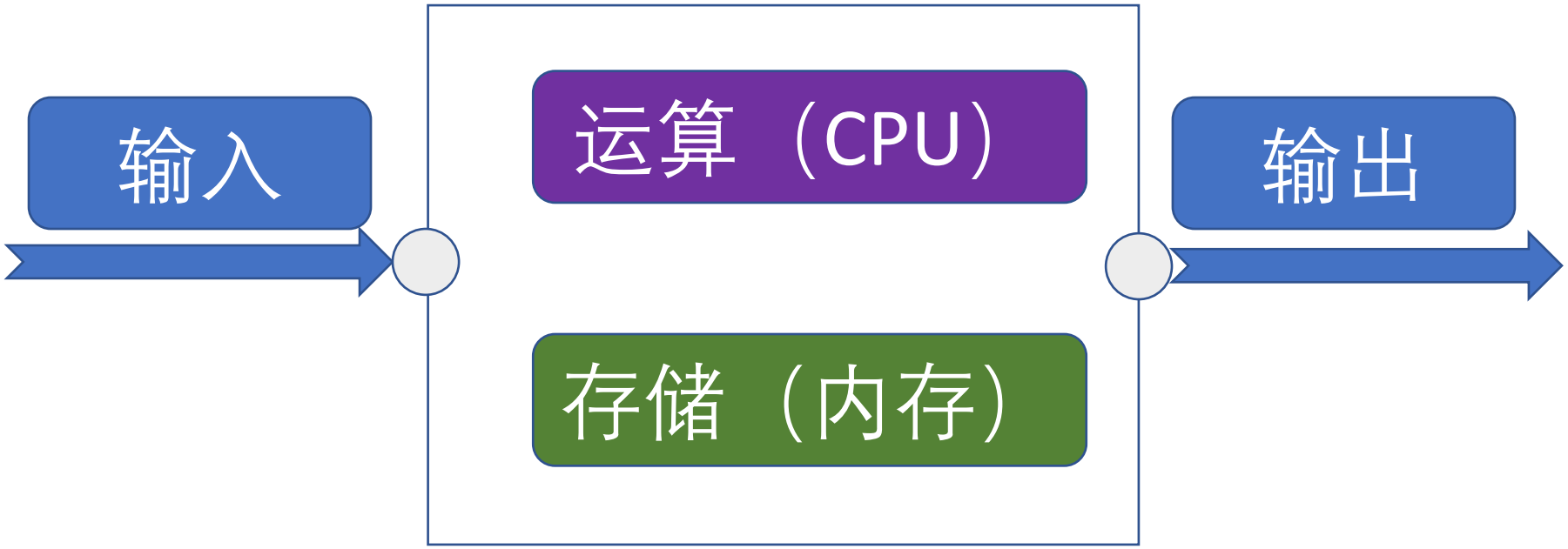

物理内存页0
物理内存页2
物理内存页3
物理内存页4
物理内存页M
物理内存
缺页
逻辑内存页0
逻辑内存页1
逻辑内存页2
逻辑内存页N
函数 F
x
y
虚拟内存
中央处理器

计算机运行全过程



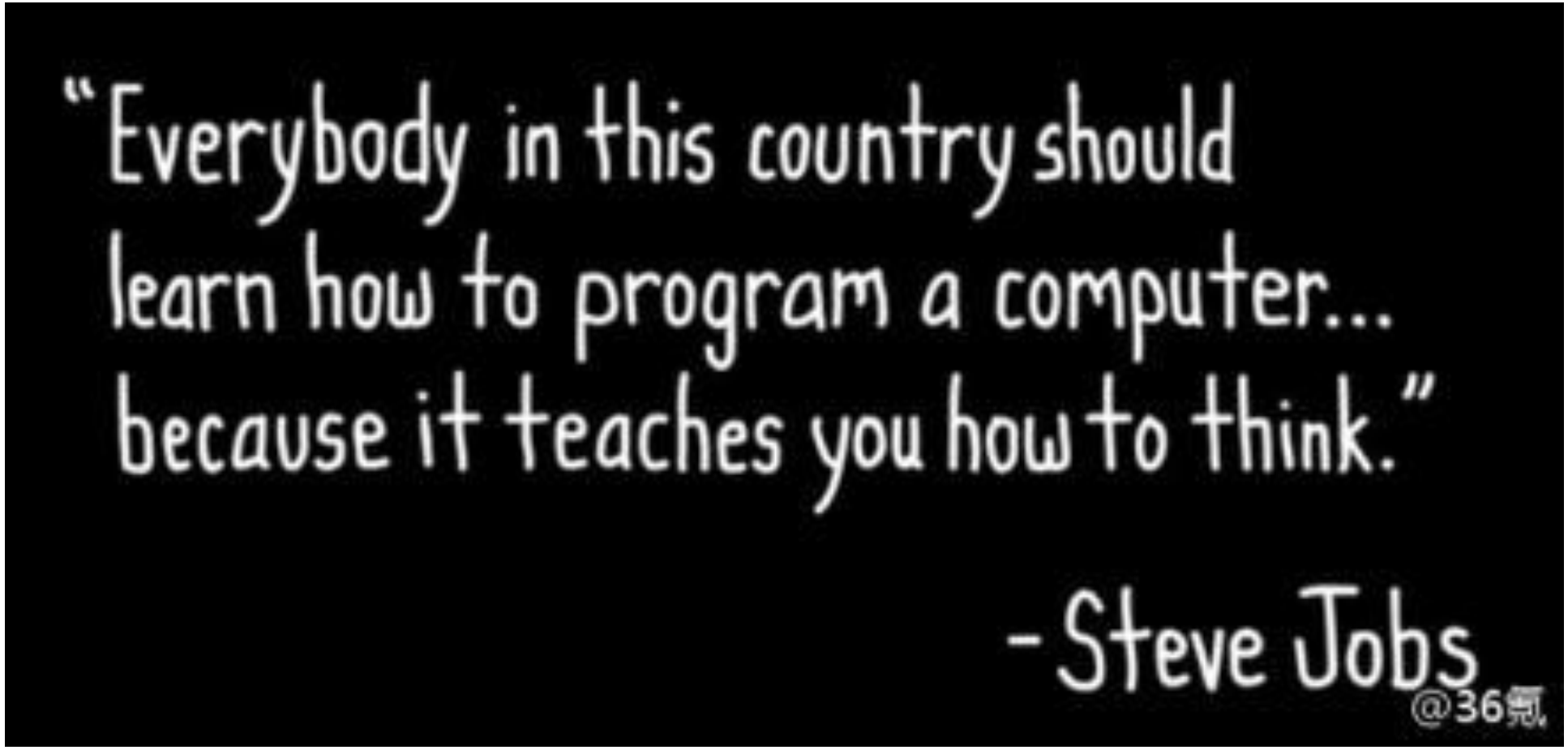

显示器
键盘
鼠标
网卡
电源
主板
CPU
寄存器
GPU
显存
内存
RAM
BIOS
ROM
CMOS
RAM
硬盘
引导区：引导程序
OS引导程序
OS shell
软件1
文件1
软件2
文件2
光驱
闪存(USB)
软盘(A/B)
硬件自检：各种驱动程序
更改的硬件配置存入CMOS
跳转到外部存储的引导区程序

“响应体”对象的抽象



例子：人、电视、空调、汽车...

(2) 编程



乔布斯：“每个人都应学习编程，因为它教你如何思考”

程序员的层次

- 架构师
  - 核心能力：抽象思维与抽象能力
  - 宏观的全局掌控能力
- 工程师
  - 软件工程
- 搬砖师（“码农”）
  - 编写代码

代码质量的评判维度

- 可阅读性：方便代码流转
- 可扩展性 / 可维护性：方便修改功能，添加新功能
- 可测试性：质量管理
- 可复用性：简化后续功能开发的难度

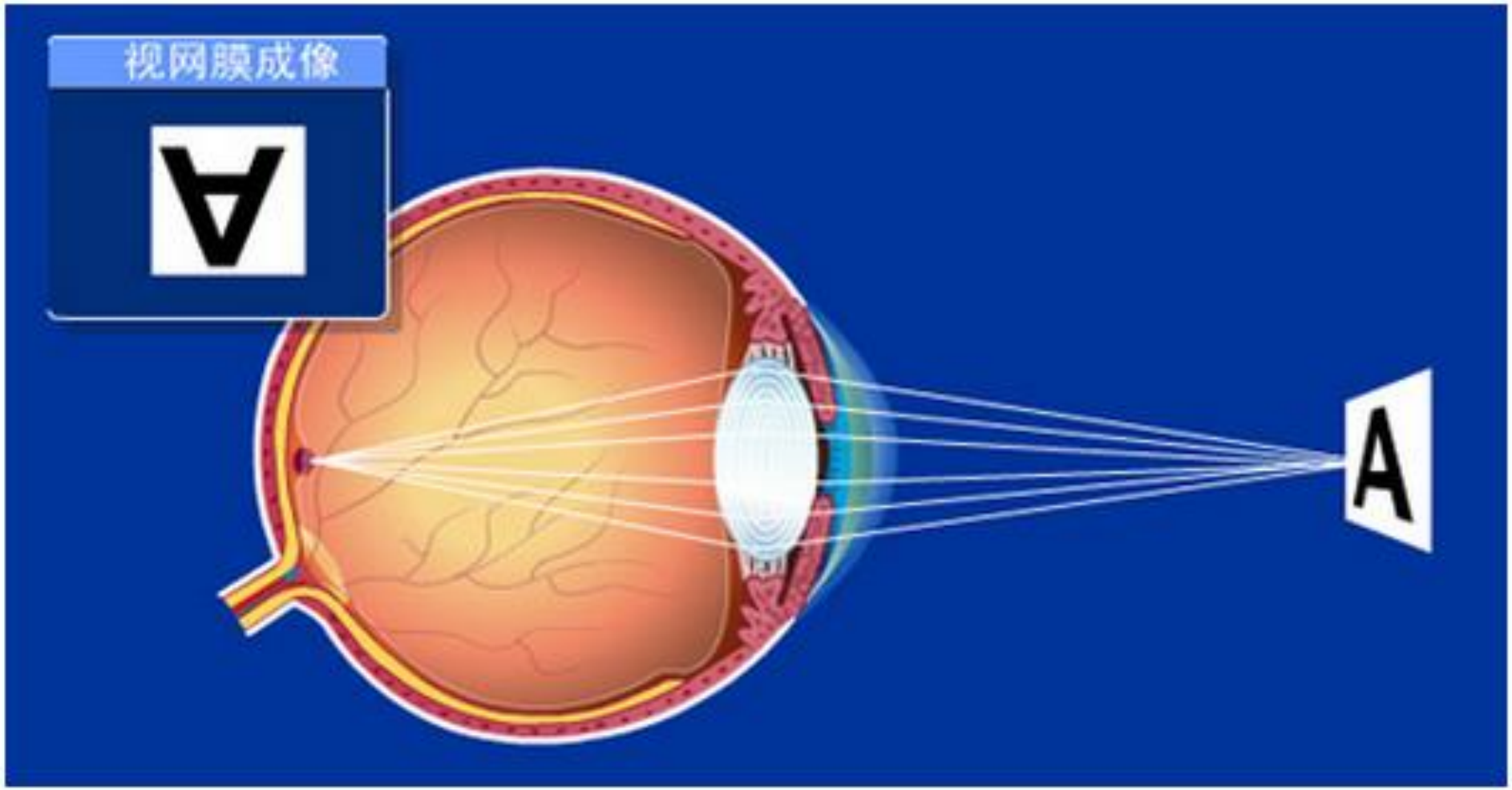
关于编程

- 好习惯
  - 代码规范：程序员的“脸面”
  - 程序注释：边写代码边注释，日志…
  - git
- 熟练C++指针、内存分配与释放、避免内存泄漏
- 调试程序：思路和逻辑清晰、多实践!
  - debug工具
    - F9, F5, F10, F11, ^F5, ^F10, Shift+F10, Shift+F11, …
    - 各种查看窗口: stack, variable, watch…
  - 使用std::cout不断输出要检查的内容
- 程序员的终身修养
  - 泛型编程、模式设计、软件工程、架构师…

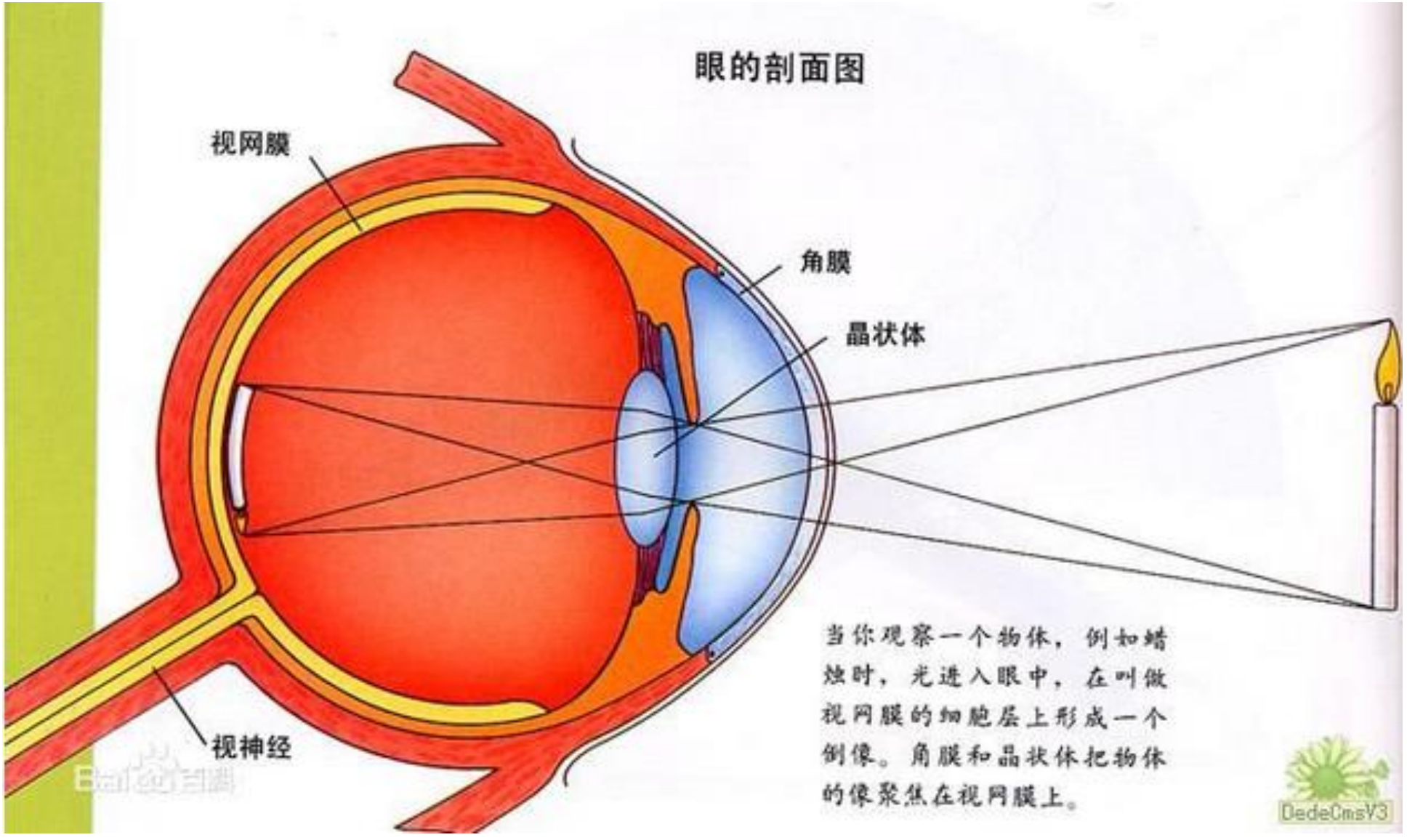
(3) 数字图像

什么是图像?

视网膜成像：上亿个感光细胞

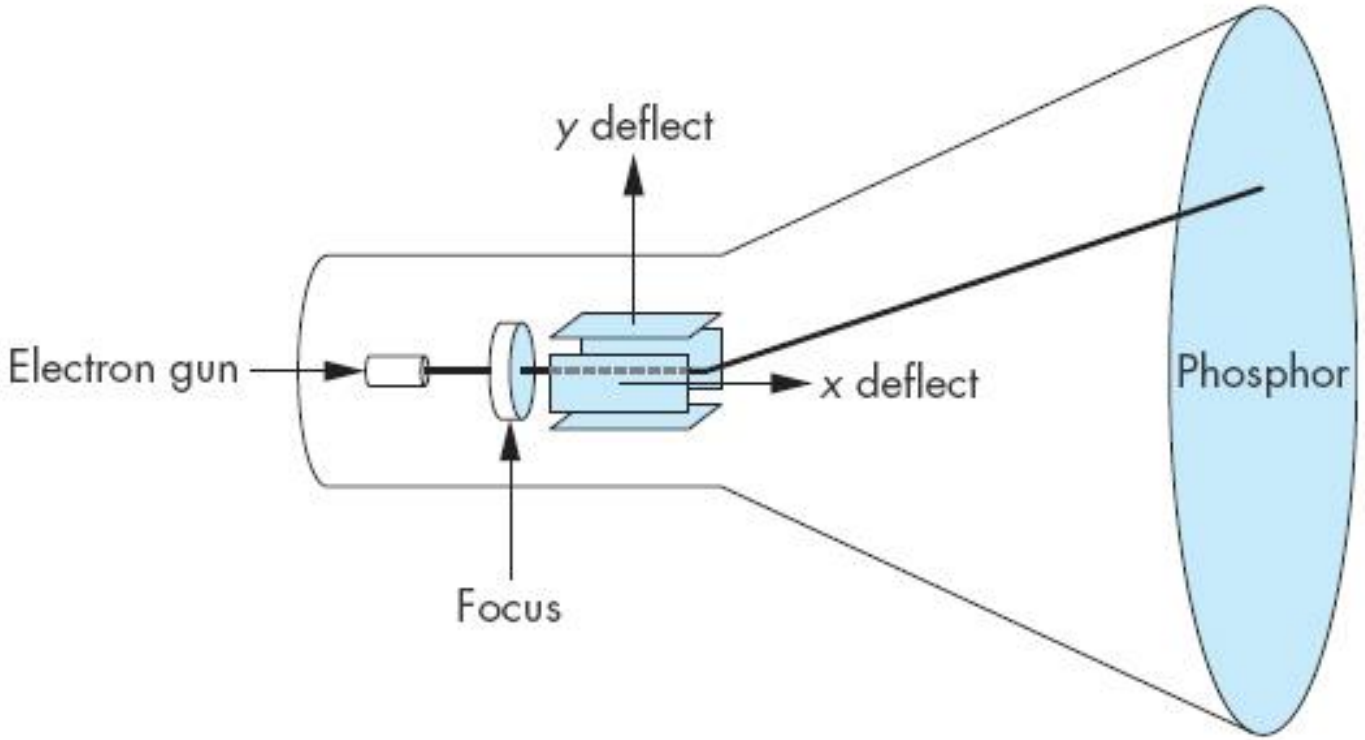


视网膜成像：上亿个感光细胞

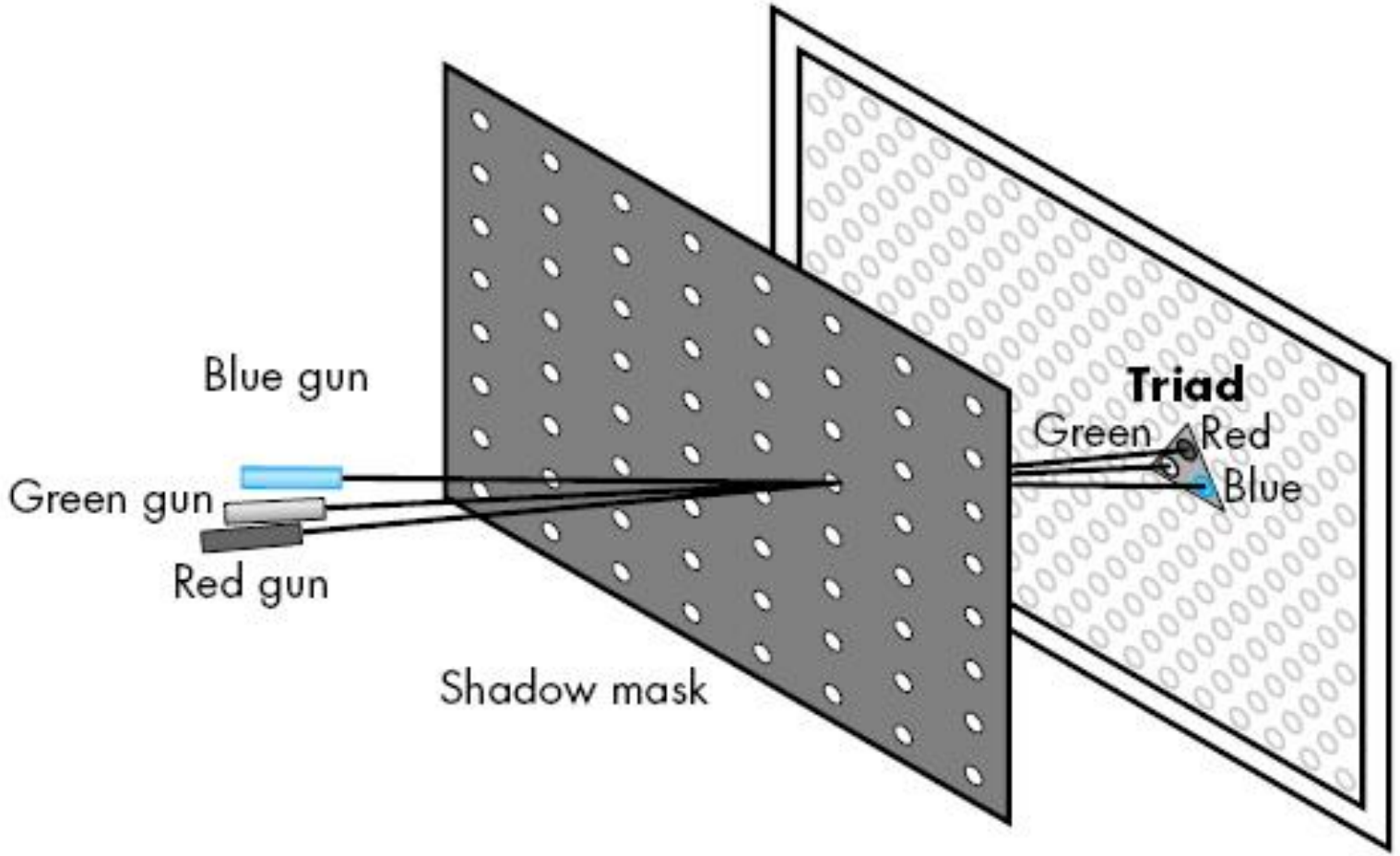


阴极射线显像管显示器 (CRT)

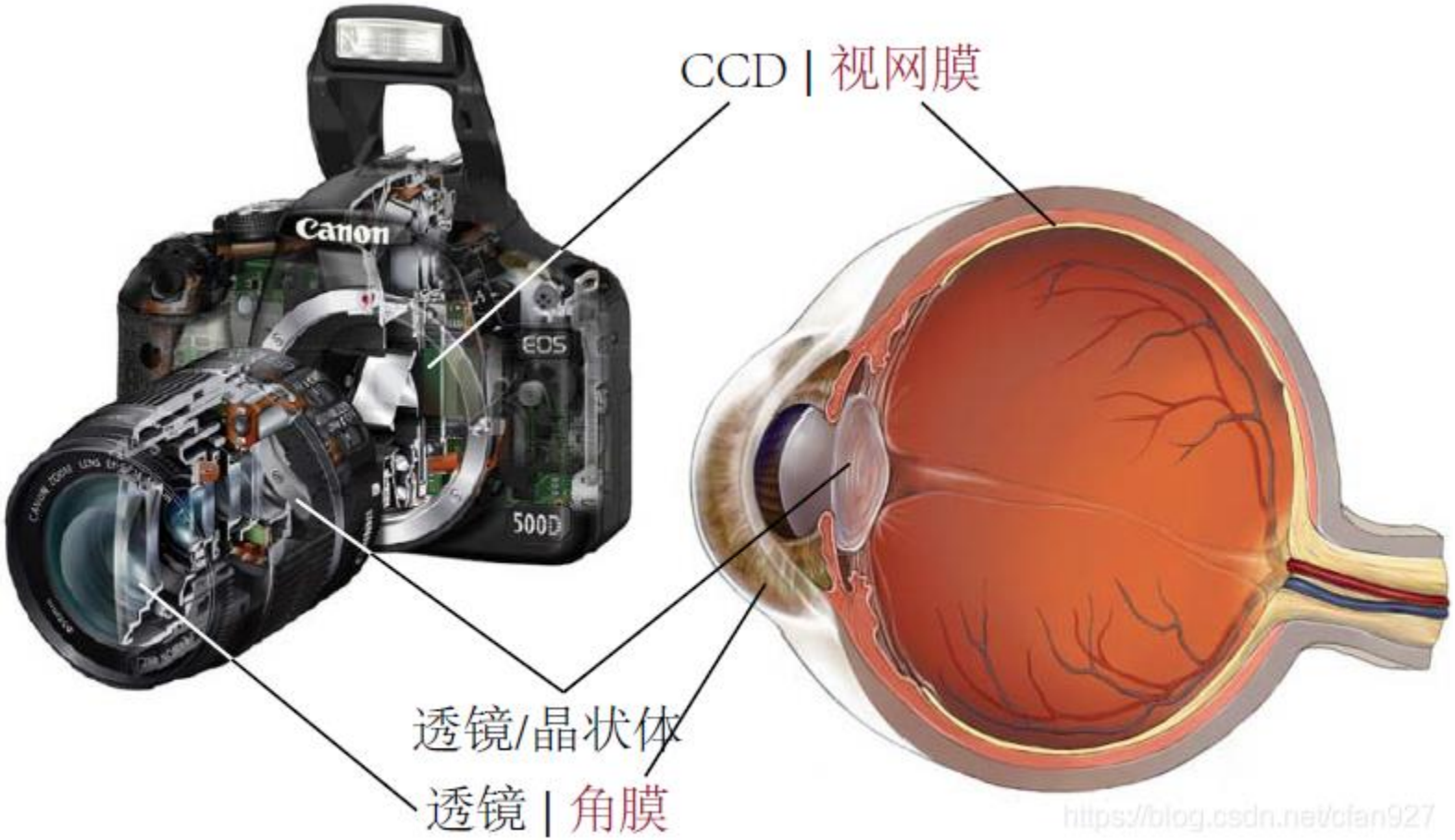
- 逐个将“点”打在屏幕上的相应位置
  - 逐行逐列扫描



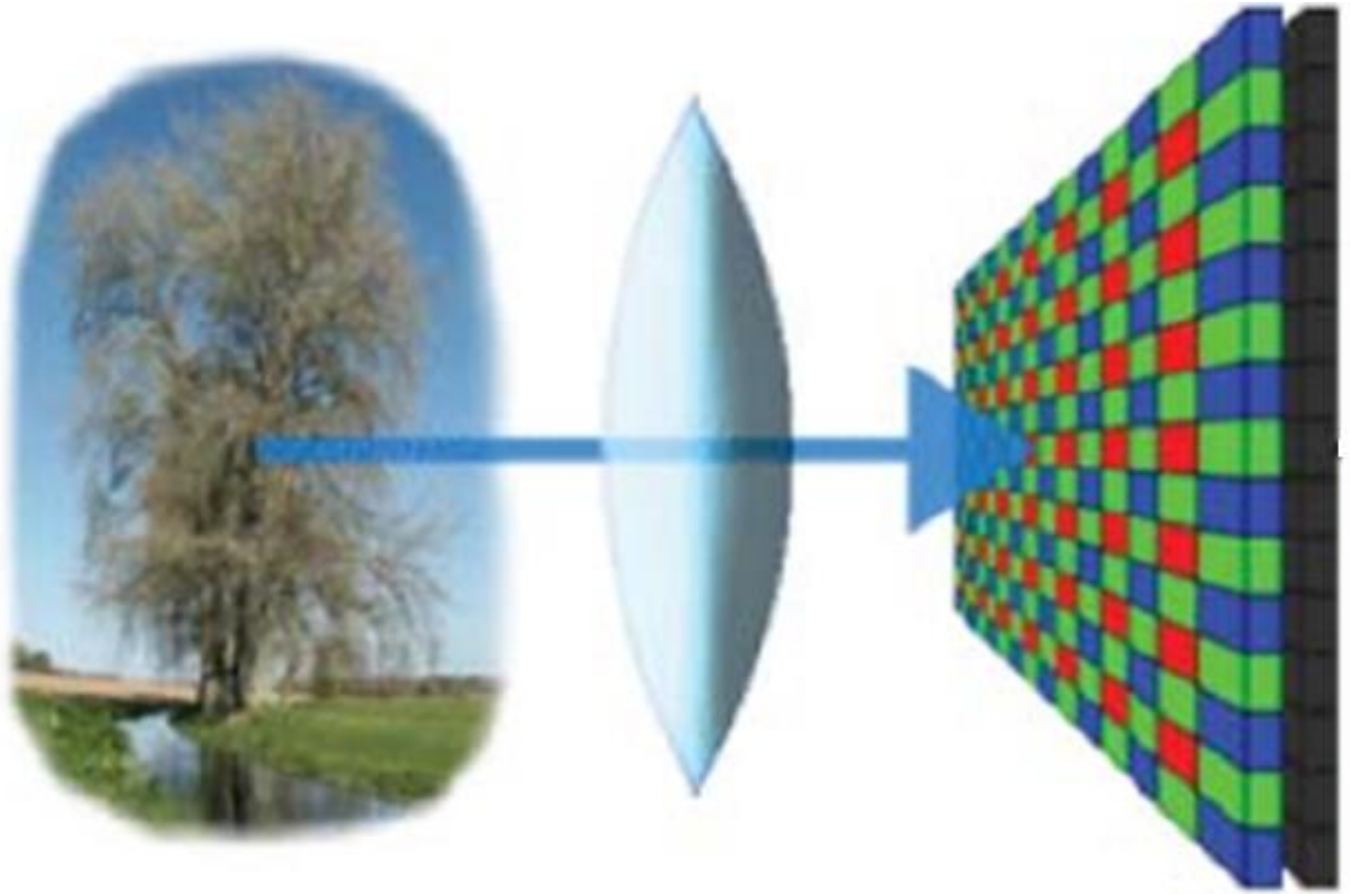
彩色CRT显示器（光栅显示器）



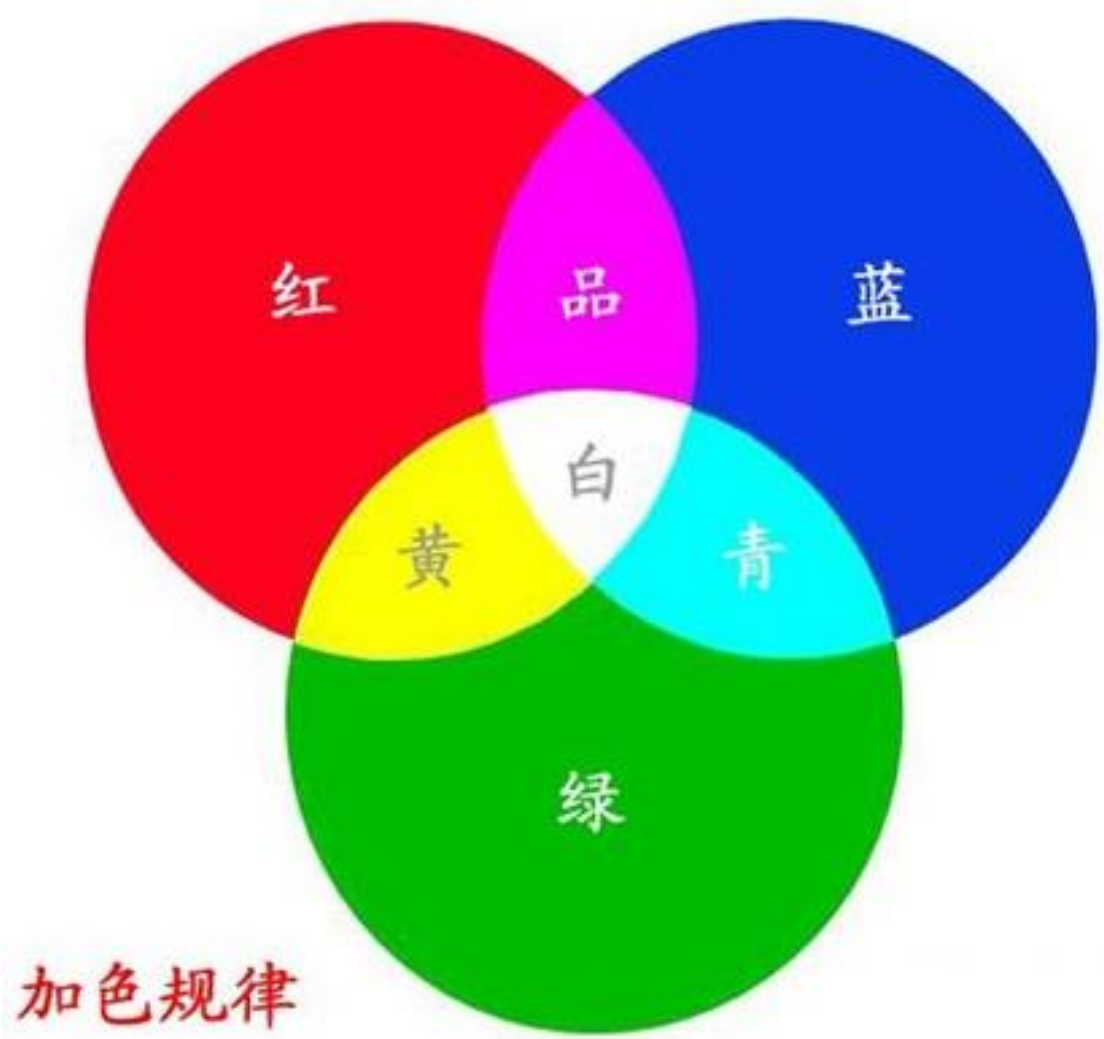
图像的抽象表达？



数字图像

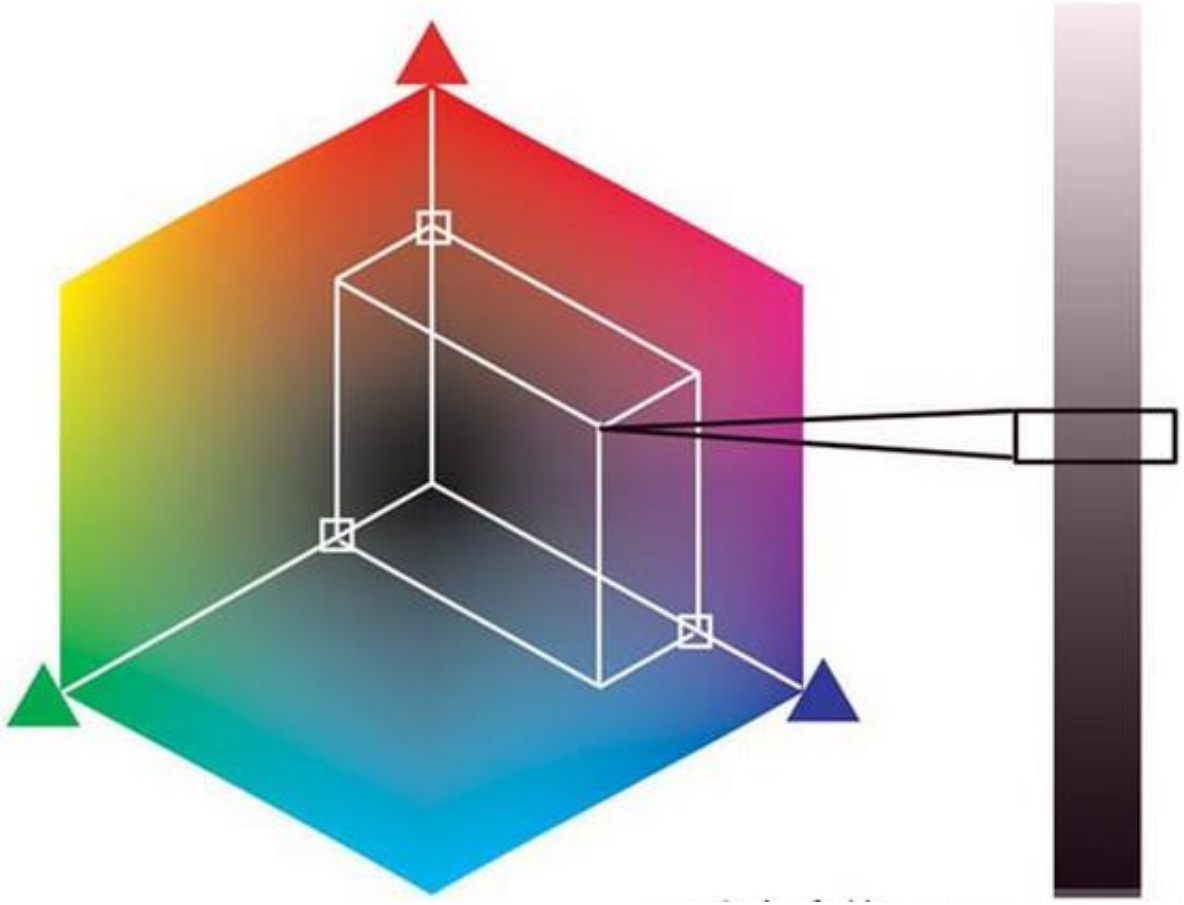


颜色三原色：红、绿、蓝



RGB颜色空间

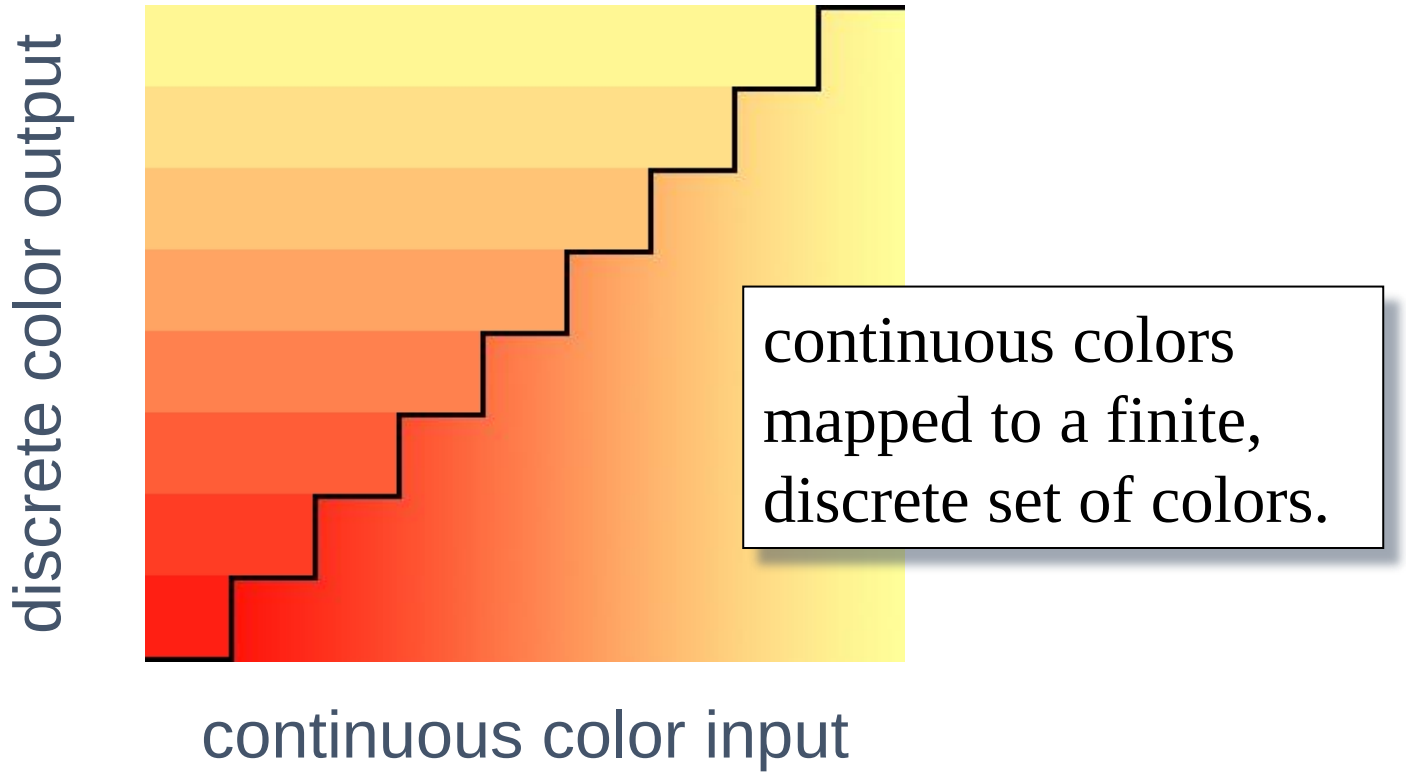
- Color = a R + b G + c B



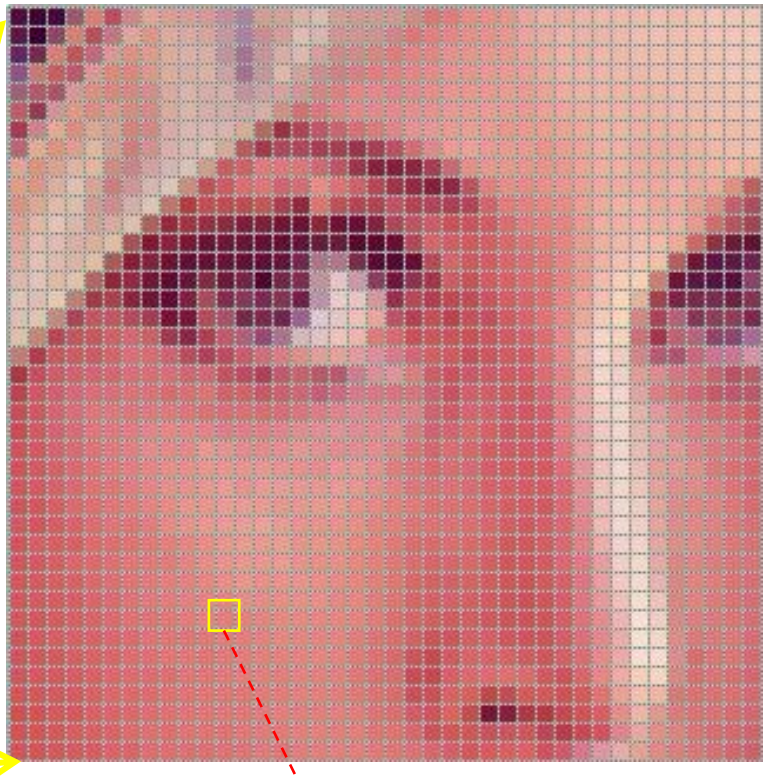
Color Spaces: Different Basis

- RGB
- CMY
- CIE XYZ
- s $l\alpha\beta$

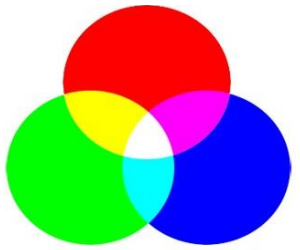
数字图像：连续空间的离散采样



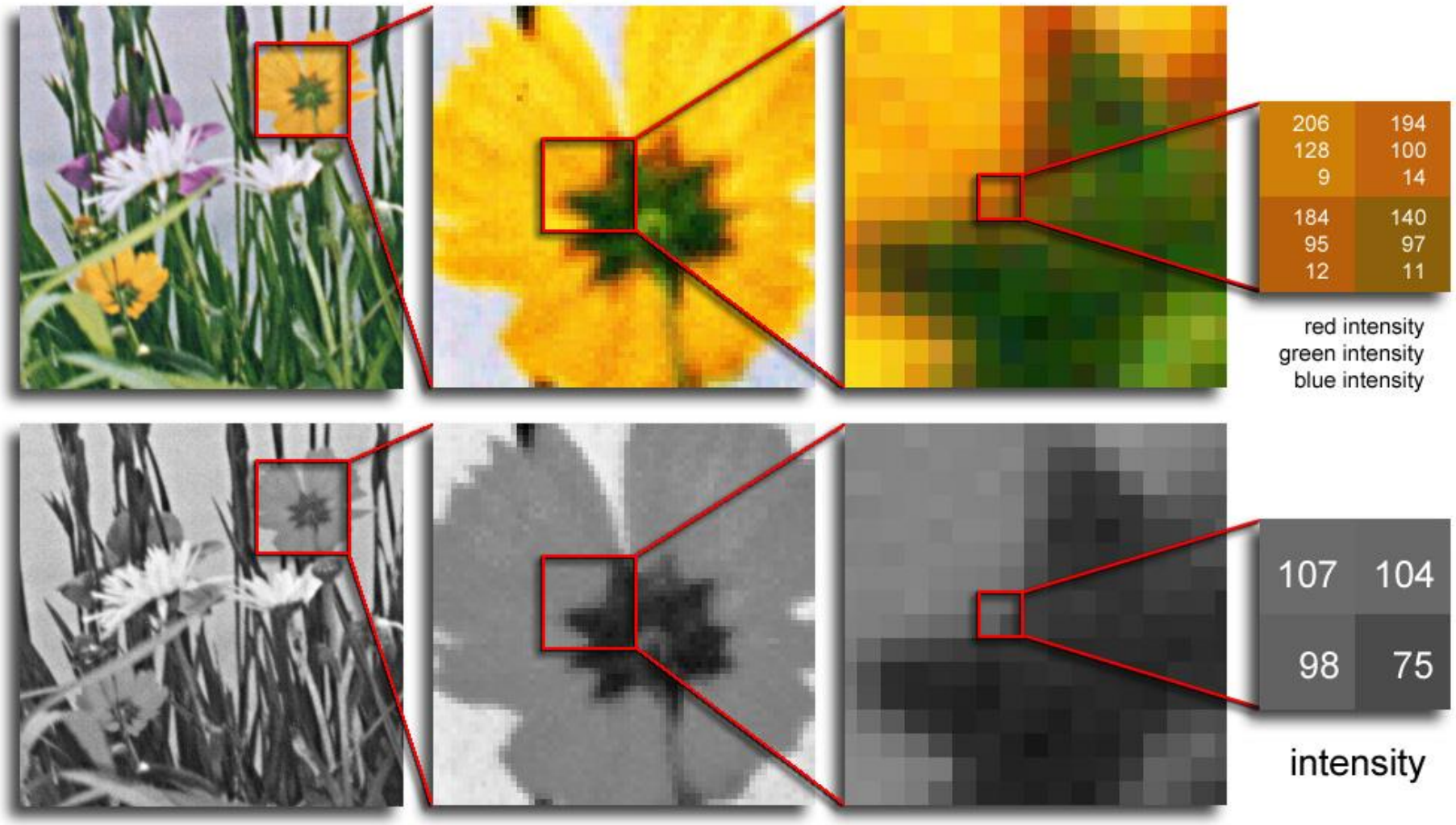
数字图像



0.6 R + 0.3 G + 0.1 B



Color Image and Gray Image



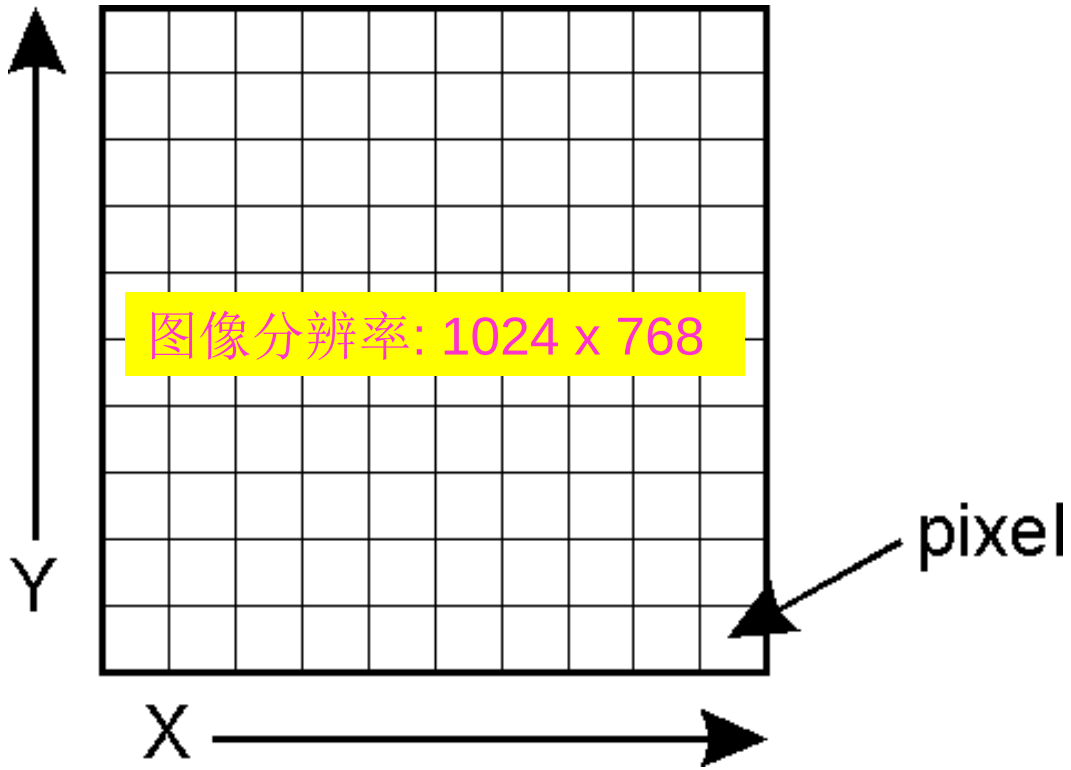
图像的抽象

- 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

- 所有像素的值!

图像的抽象

- 矩阵表达，分辨率



图像编程

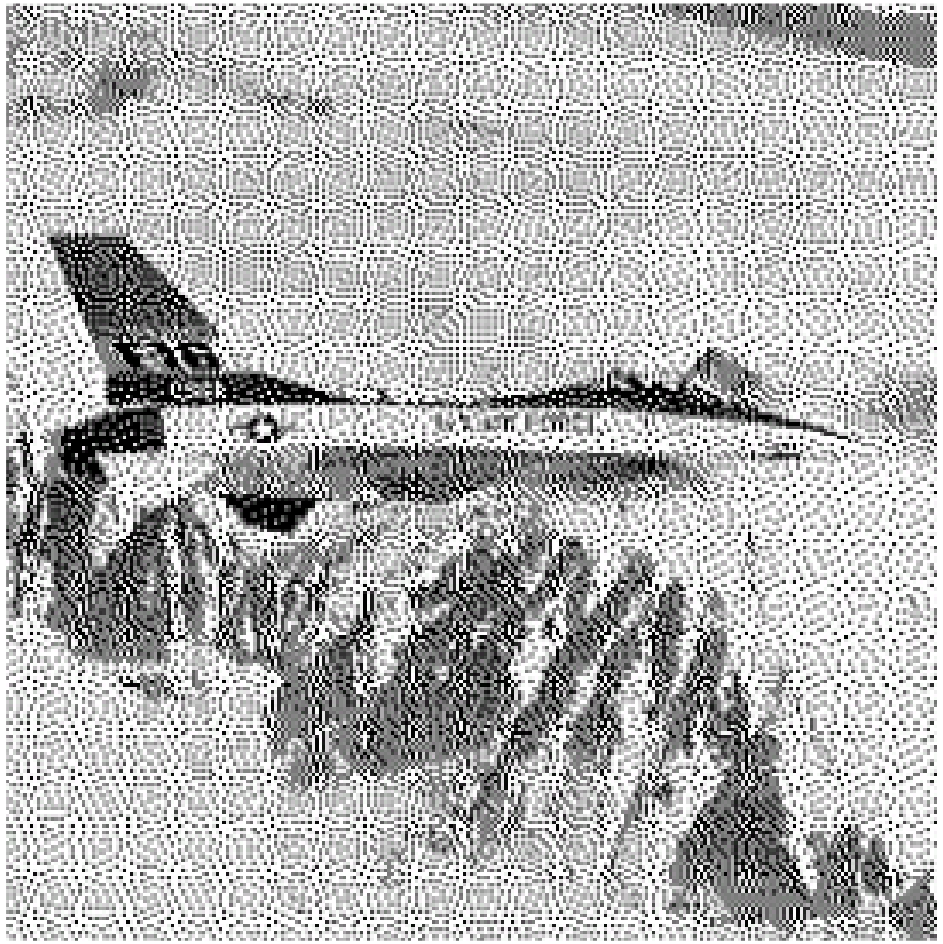
- Matlab

- C/C++
- OpenCV (Open Computer Vision)

图像的种类

- Binary images (0 or 1)
- Gray images (0~255)
- Color images
  - 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）

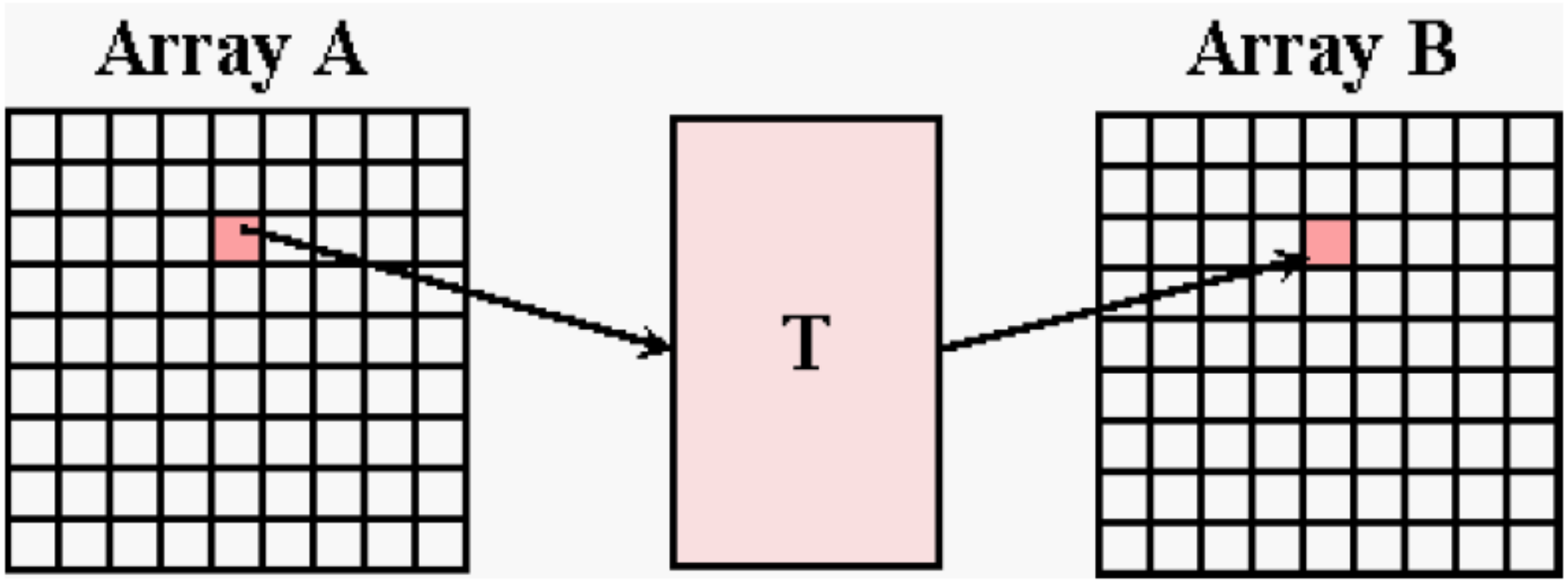
- 24 bits per pixel, and the three channels R G B are three gray images respectively
  - Each of the channels is encoded by 8 bits

- 32 bits
  - (R, G, B, a)

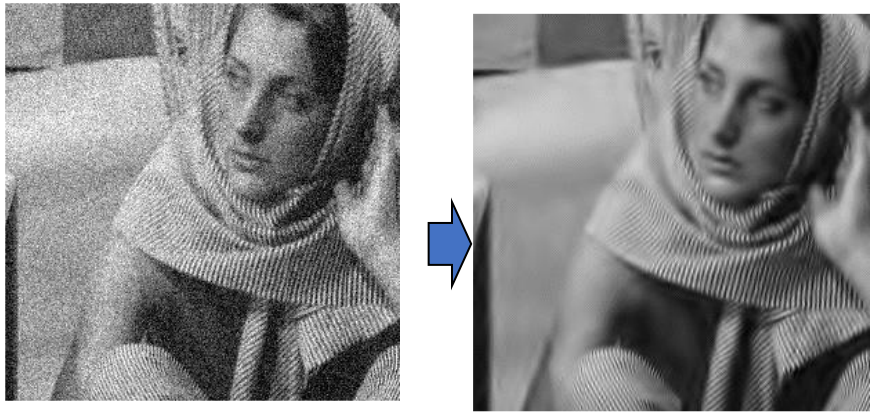
Color Components

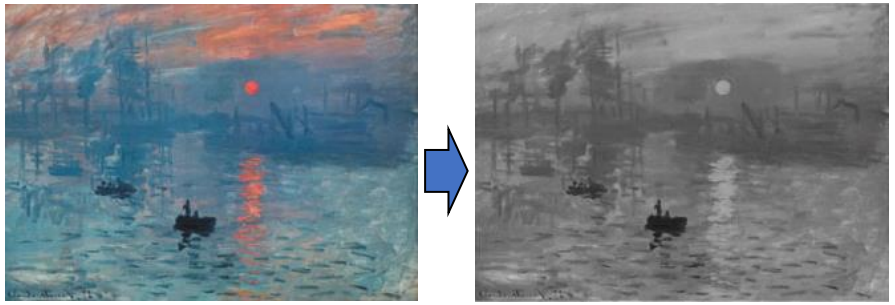
图像处理（图像变换）

$$\mathbf{B}[x, y] = T\big[\mathbf{A}[x, y]\big]$$



图像处理的例子

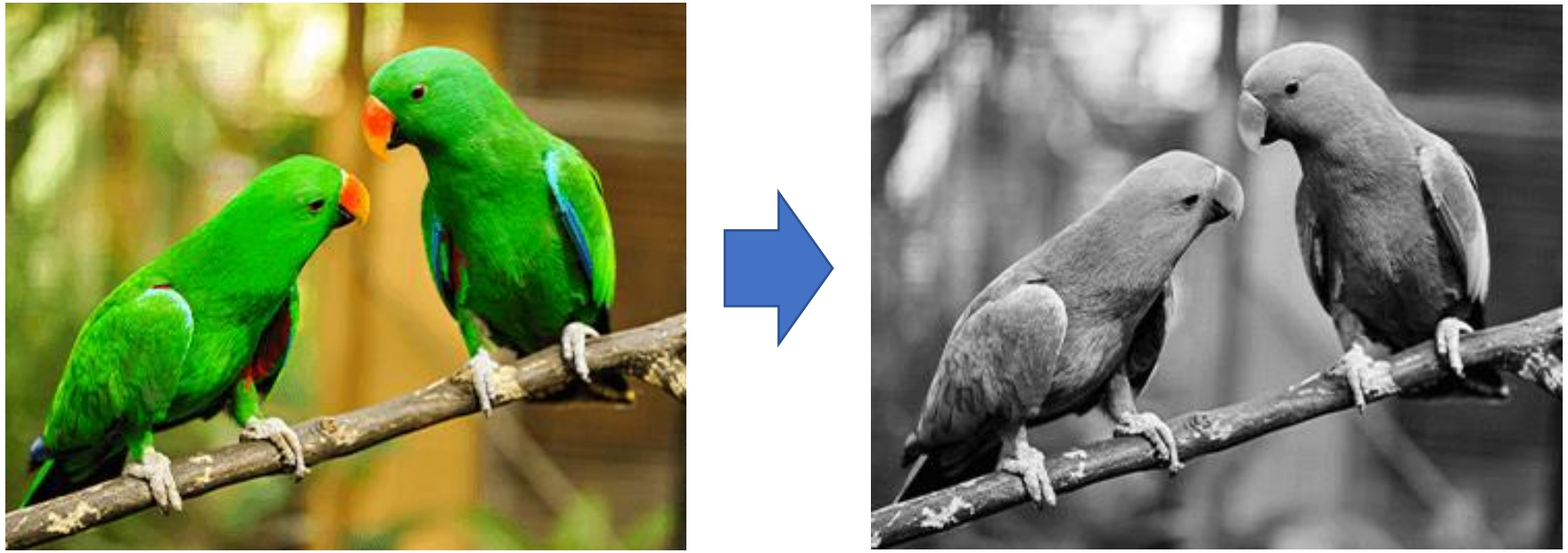





Since 1699, when French explorers
landed at the great bend of the
Mississippi River and celebrated
the first Mardi Gras in North America,
New Orleans has brewed a fascinating
melange of cultures. It was French,
then Spanish, then French again, then
sold to the United States. Through all
these years, and even into the 1900s,
others arrived from everywhere:
Acadians (Cajuns), Africans, indige-

例子：彩色图像灰度化 (Color2Gray)

- 应用?



https://www.rapidtables.com/convert/image/rgb-to-grayscale.html
https://ieeexplore.ieee.org/abstract/document/5445596
https://users.cs.northwestern.edu/~ago820/color2gray/

问题:
彩色图像灰度化
哪个结果好?

方法1结果

输入图像

方法2结果



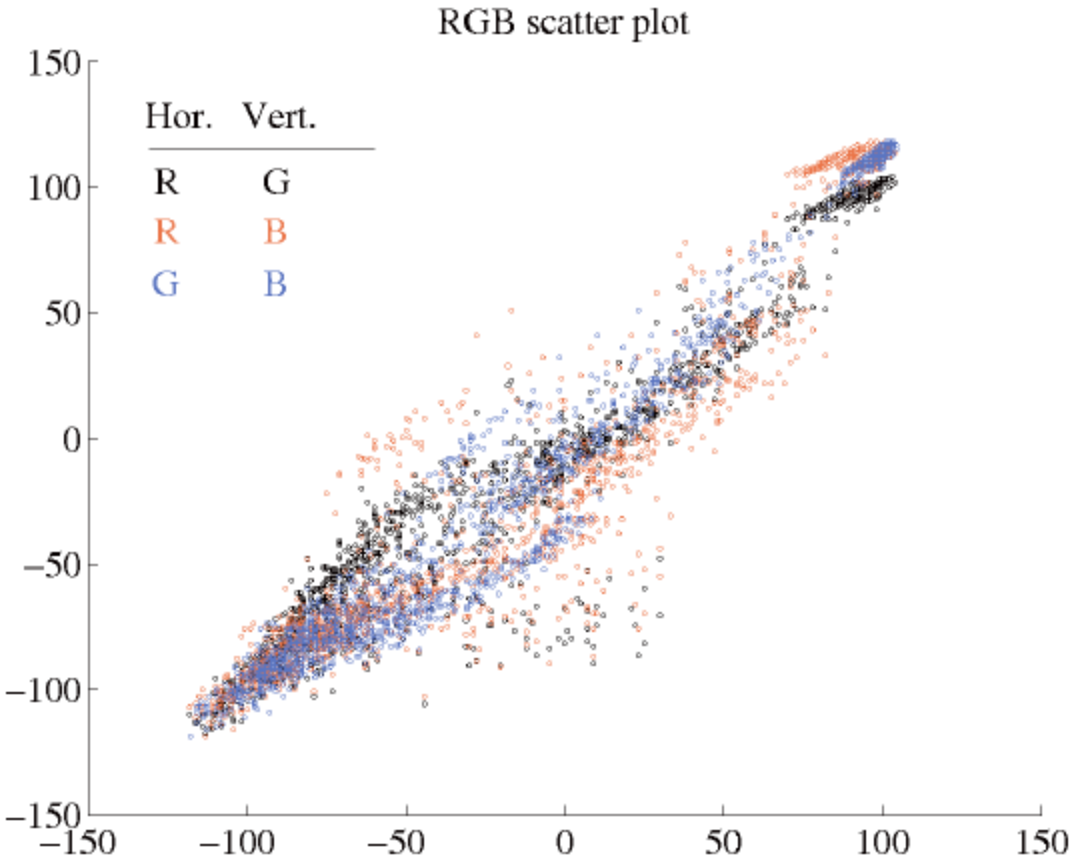


Thunder Bay
130
Isle Royale
National
Park
Lake

Thunder Bay
130
Isle Royale
National
Park
Lake

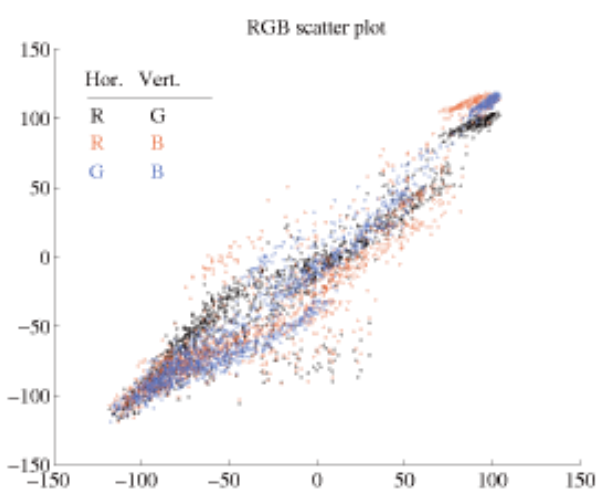
Thunder Bay
130
Isle Royale
National
Park
Lake

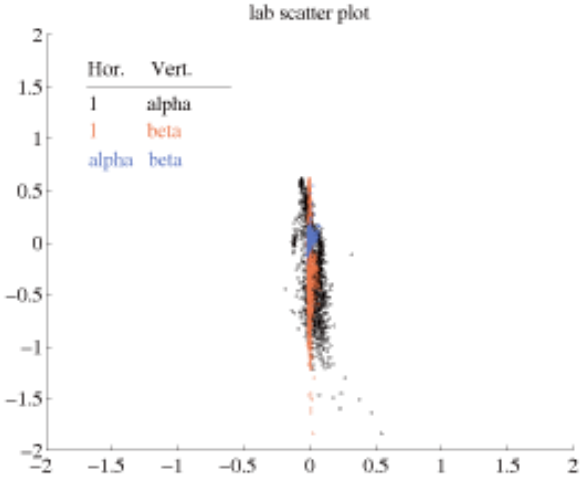
建模：图像--3D空间的点集

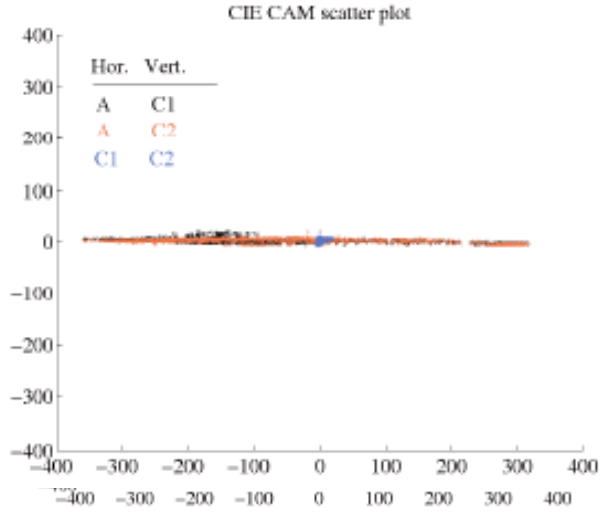


Color Spaces: Different Basis

- RGB
- CMY
- CIE XYZ
- s $l\alpha\beta$







RGB

$l\alpha\beta$

CIE

Example: RGB→ $l\alpha\beta$

$$\begin{bmatrix} L \\ M \\ S \end{bmatrix} = \begin{bmatrix} 0.3811 & 0.5783 & 0.0402 \\ 0.1967 & 0.7244 & 0.0782 \\ 0.0241 & 0.1288 & 0.8444 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

$$\begin{aligned} \boldsymbol{L} &= \log L \\ \boldsymbol{M} &= \log M \\ \boldsymbol{S} &= \log S \end{aligned}$$

$$\begin{bmatrix} l \\ \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{3}} & 0 & 0 \\ 0 & \frac{1}{\sqrt{6}} & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & -2 \\ 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{L} \\ \mathbf{M} \\ \mathbf{S} \end{bmatrix}$$

例子2：图像颜色风格迁移（“学习”）

Reinhard et al. Color transfer between images. IEEE CG&A, 2001.
https://ieeexplore.ieee.org/abstract/document/946629

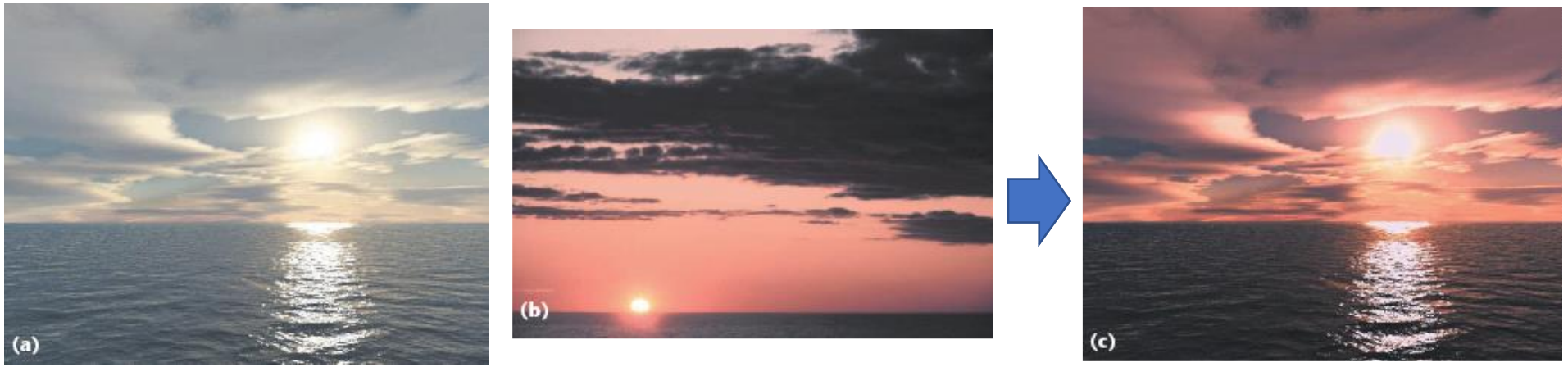
数学模型及算法

- 使点云位置、形状相似（均值和方差变换）

$$l^* = l - \langle l \rangle \qquad l' = \frac{\sigma_t^l}{\sigma_s^l} l^*$$

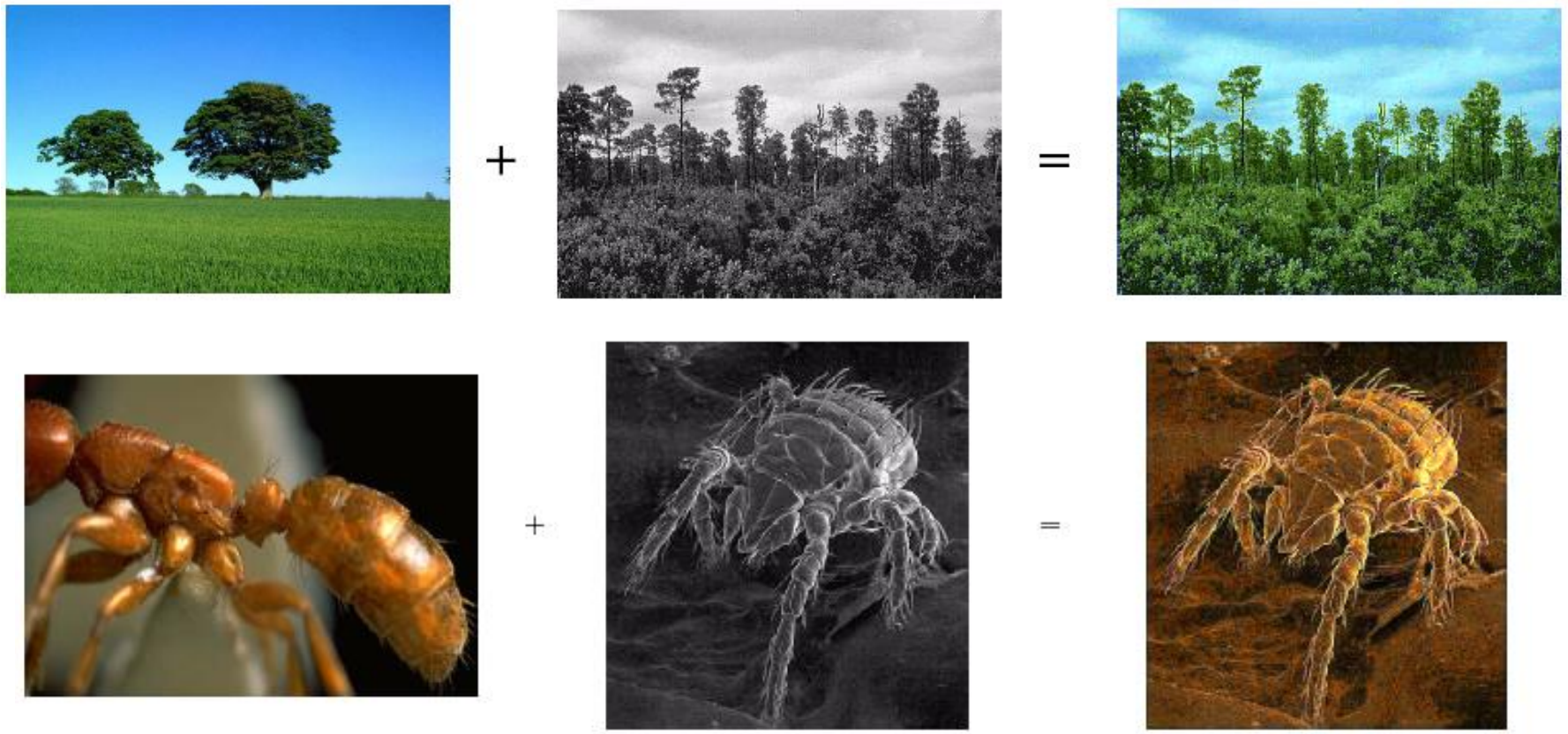
$$\alpha^* = \alpha - \langle \alpha \rangle \qquad \alpha' = \frac{\sigma_t^\alpha}{\sigma_s^\alpha} \alpha^*$$

$$\beta^* = \beta - \langle \beta \rangle \qquad \beta' = \frac{\sigma_t^\beta}{\sigma_s^\beta} \beta^*$$



[Reinhard et al. Color Transfer between Images. IEEE CG&A, 2001]

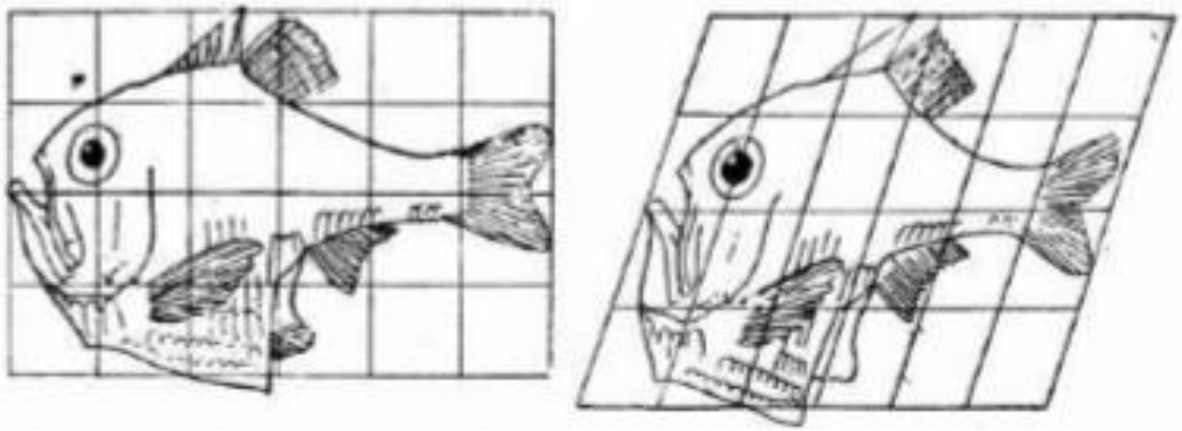
例子3：灰度图像上色



https://zhuanlan.zhihu.com/p/202553515

图像变形Image Warping

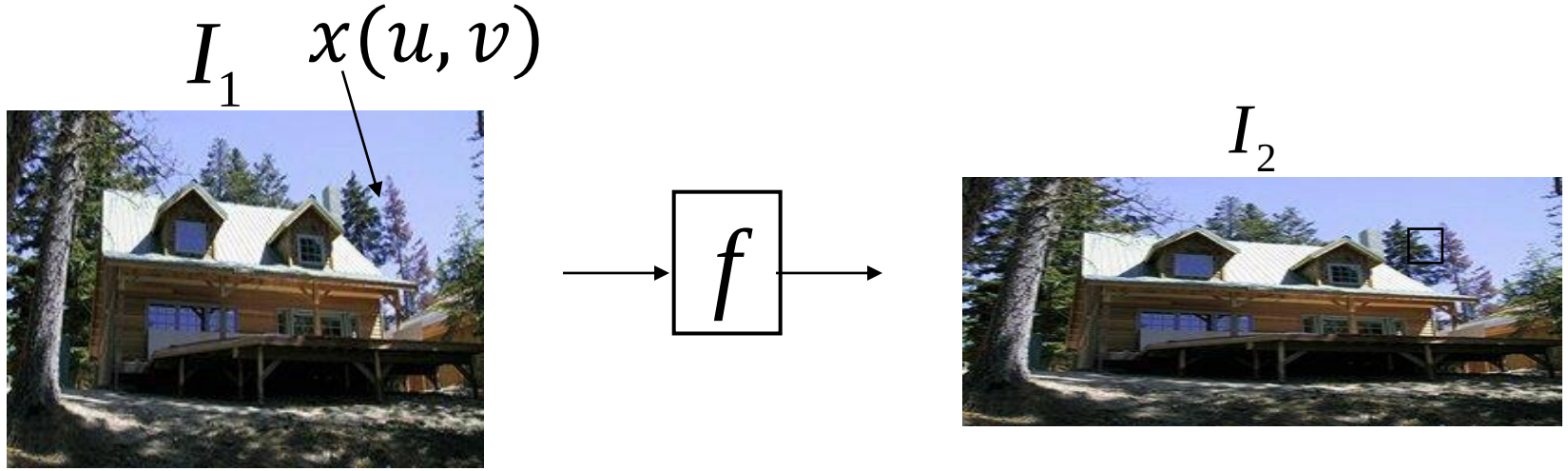
- 通过操控图像定义域，改变图像中物体的几何形状，实现图像整体或局部的变形

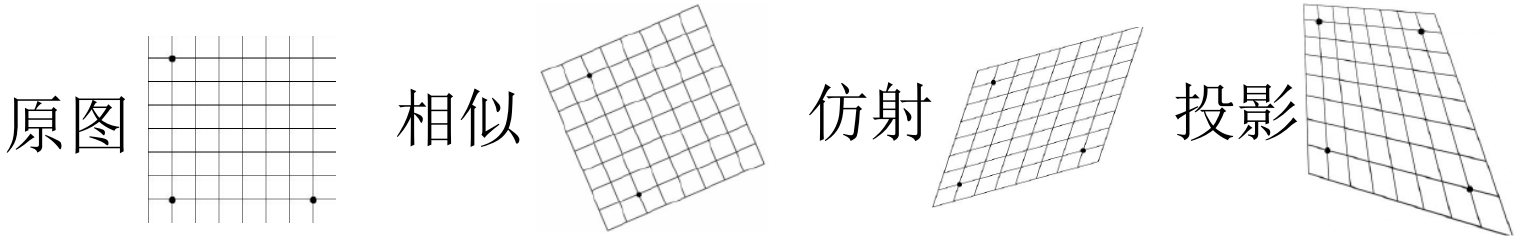


图像变形Image Warping

- 根据变形函数逐像素改变输入图像，生成变形后的图像

$$I_2(f(x(u,v))) = I_1(x(u,v))$$





图像变形Image Warping

- 相似变换

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} a & b \\ -b & a \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} c \\ d \end{bmatrix}$$

- 二维缩放、旋转和平移变换的组合。
- 允许一个正方形被转换成任何旋转的矩形。
- 线之间的夹角被保留
- 自由度为4（a，b，c，d）
- 逆变换是相同的表示（相似性）

图像变形Image Warping

- 仿射变换

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix}$$

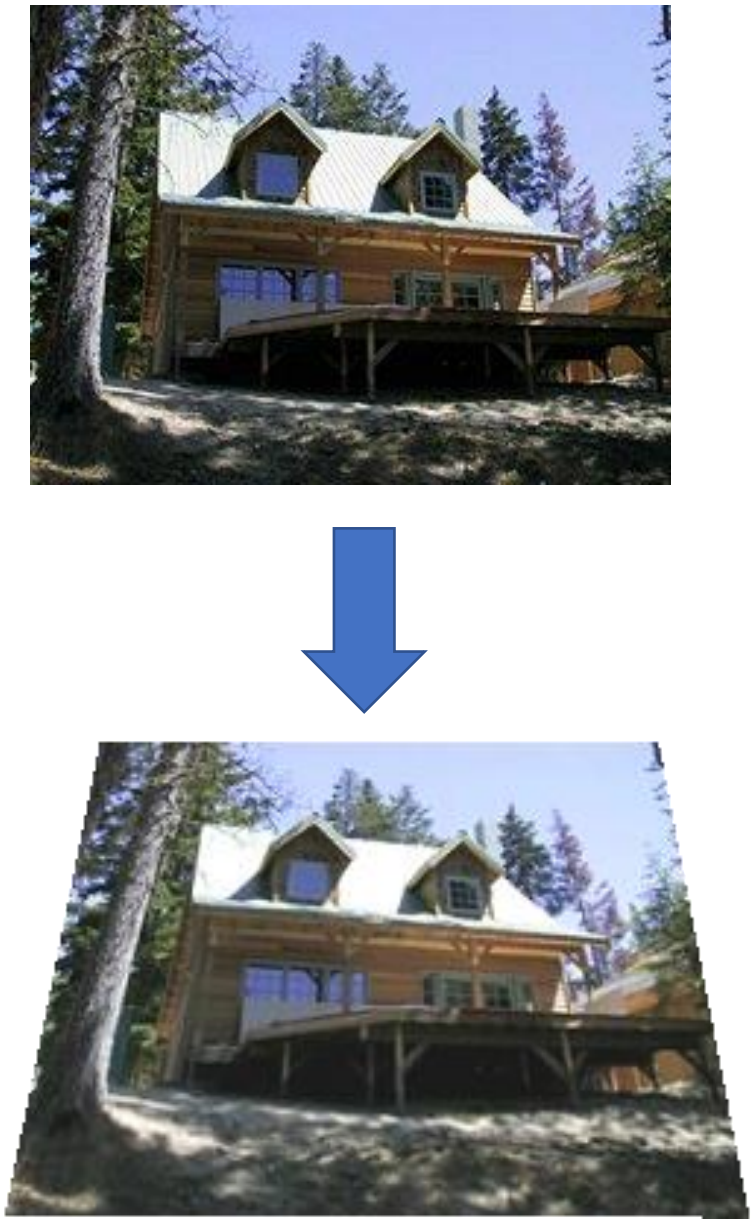
- 二维缩放、旋转、剪切和平移变换的组合。
- 允许一个正方形被转换成任何平行四边形。
- 自由度为6（a，b，c，d, e, f）
- 逆变换是相同的表示（相似性）

图像变形Image Warping

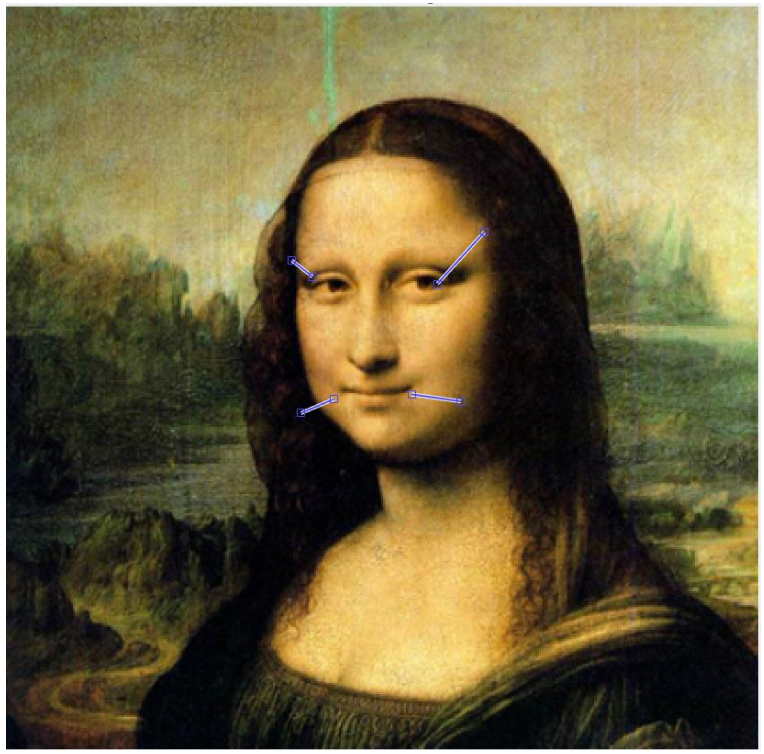
- 投影变换

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \sim \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

  - 如果$g = h = 0$，那么为一个特殊情况的仿射
  - 允许一个正方形被扭曲成任何四边形
  - 自由度为8（a-h）
  - 逆是同一形式（也就是投影）。



图像变形Image Warping



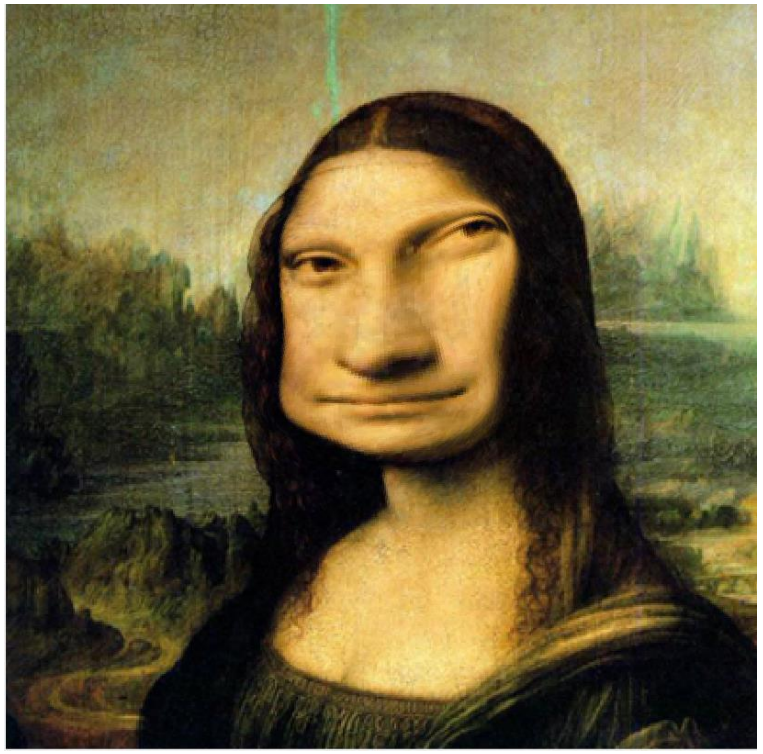
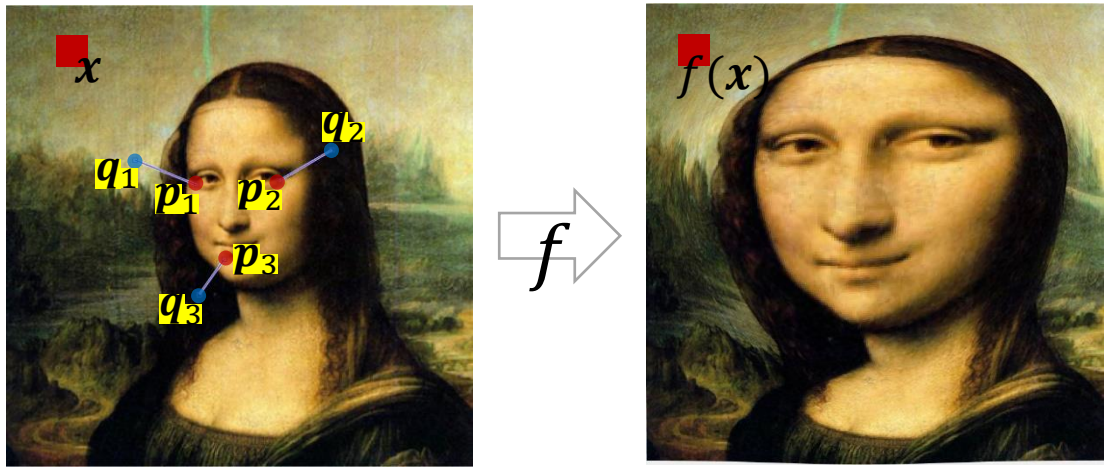


Image Warping using RBF



$$f(\boldsymbol{x}) = \sum_i b_i(\boldsymbol{x})\boldsymbol{a}_i$$

$$b_i(x) = \frac{1}{|x - p_i|^2 + d}$$

$$f(\boldsymbol{x}) = A\boldsymbol{x} + \boldsymbol{c} + \sum_i b_i(\boldsymbol{x})\boldsymbol{a}_i$$

$$f(\boldsymbol{x}) = \boldsymbol{x} + \sum_i b_i(\boldsymbol{x})\boldsymbol{a}_i$$

内容感知的图像缩放

- **概念**
  - 图像在不同终端显示时，面临画面尺寸变化
  - 通过缩放减少或者扩展图像的大小去适应不同的显示屏幕

图像缩放

Content Aware

Uniform Scaling

内容感知的图像缩放

- **图像处理**
  - （不）等比例缩放、裁剪
- **图形处理**
  - 内容感知的画面增删或变形

不等比例缩放

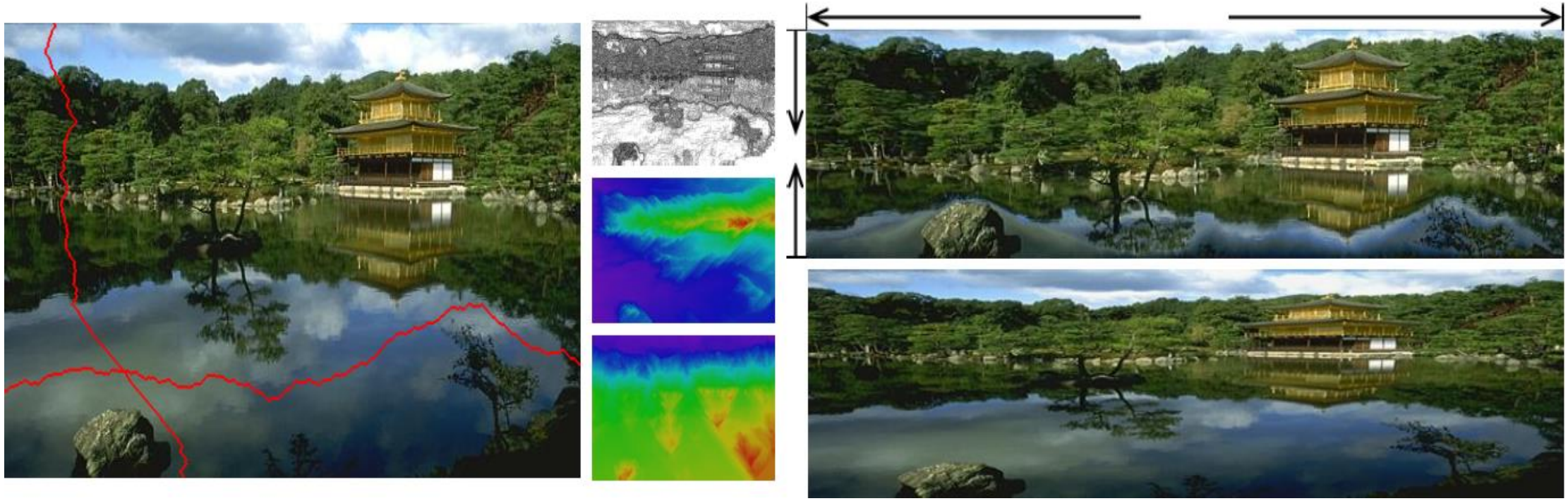
等比例缩放

裁剪

Seam Carving

图像缩放方法

- **Seam Carving**
  - **思想：**通过在图像中增加水平和竖直方向连续的缝隙进行扩大或缩小图像
  - seam：图像中连通的低能量像素通路，并且每行或者每列只包含一个像素

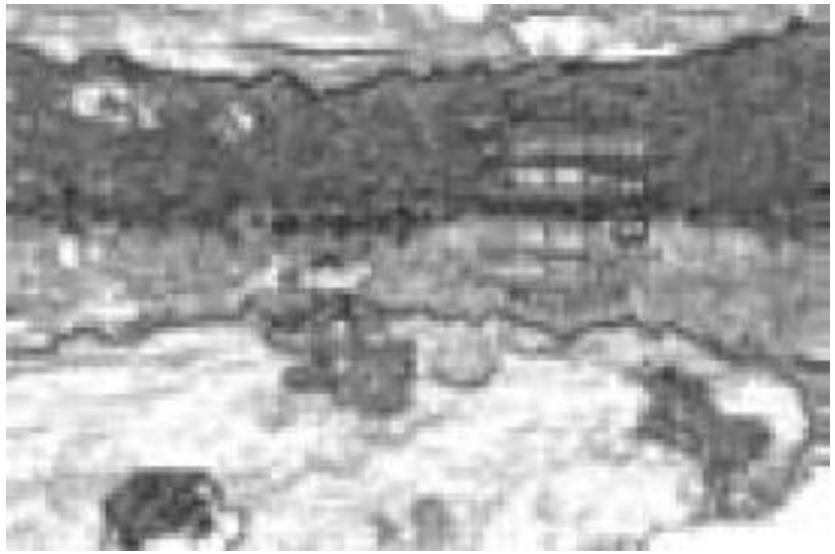


Seam Carving

- **方法**
  - 定义Seam(缝隙)为图像中穿过较低视觉显著性区域的连线

视觉显著性 $$e(I) = \left|\frac{\partial I}{\partial x}\right| + \left|\frac{\partial I}{\partial y}\right|$$

缝隙



梯度图

Seam Carving

- 方法
  - 定义Seam(缝隙)为图像中穿过较低视觉显著性区域的连线
  - 动态规划寻找符合条件的缝隙

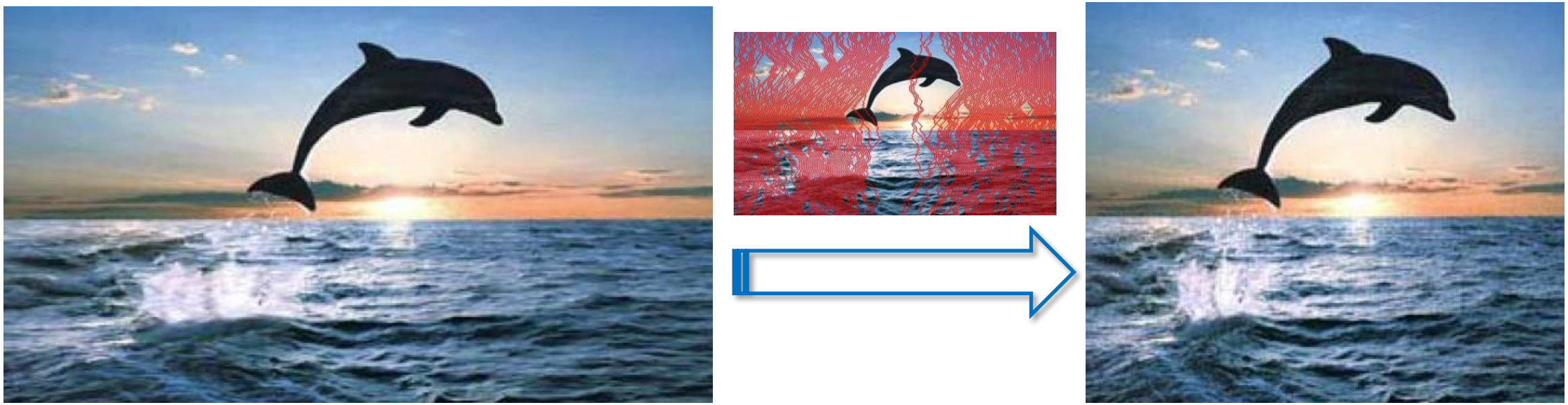
竖直缝隙 $s = \{s_i\}_{i=1}^{n} = \{x(i)\}_{i=1}^{n}$ $\forall i,\ |x(i) - x(i-1)| \leq 1$

最优缝隙 $s^* = \min_s E(s) = \min_s \sum_{i=1}^{n} \mathrm{e}(\mathrm{I}(s_i))$

$$M(i,j) = \mathrm{e}(i,j) + \min(M(i-1,j-1), M(i-1,j), M(i-1,j+1))$$

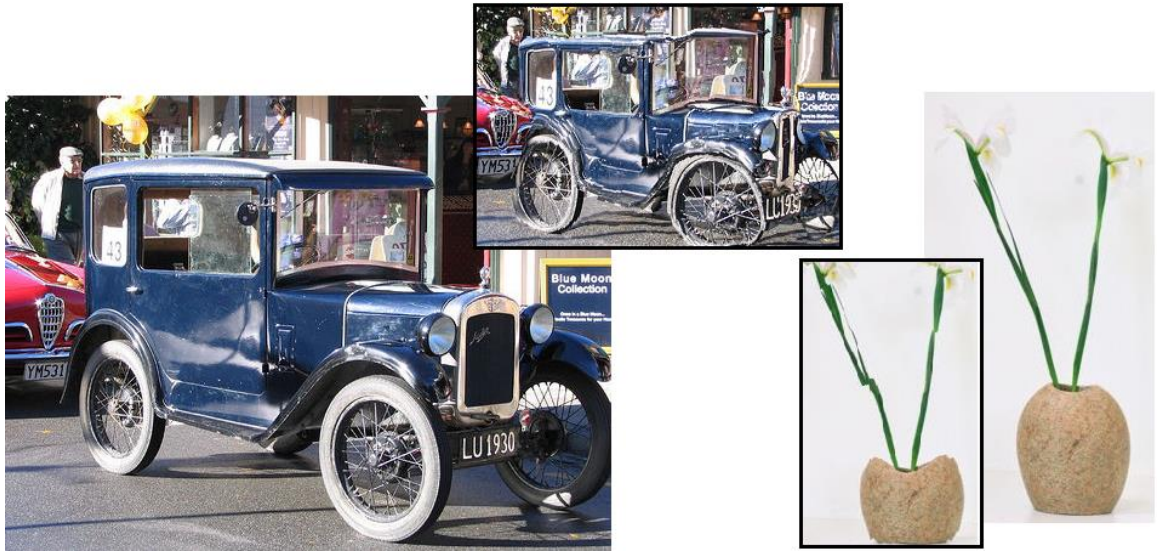
Seam Carving

- 方法
  - 定义Seam(缝隙)为图像中穿过较低视觉显著性区域的连线
  - 动态规划寻找符合条件的缝隙
  - 删除缝隙，调整图像尺寸



Seam Carving

- 结果

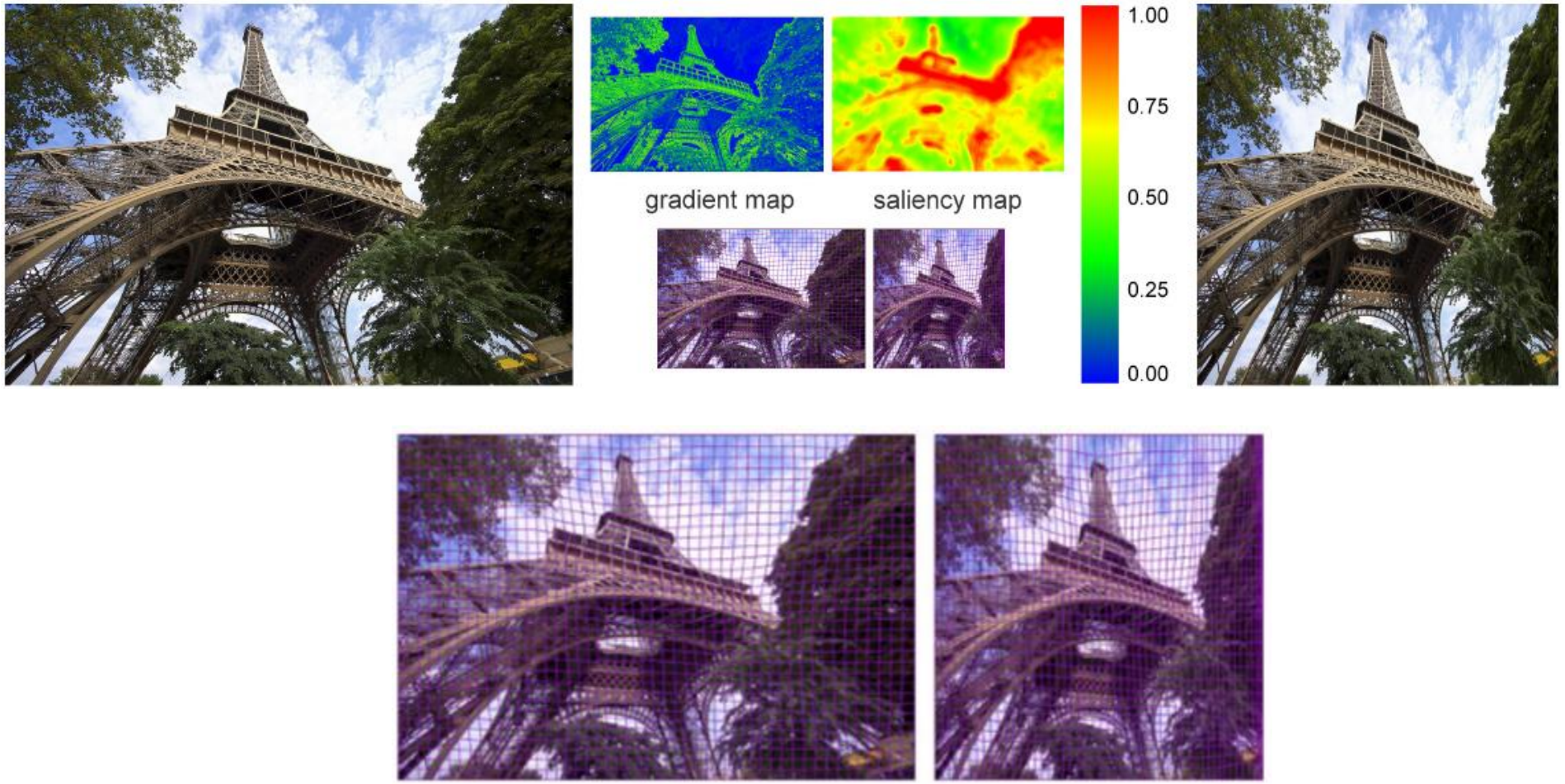


局限性

1. 效率低
2. 难以保持图像结构

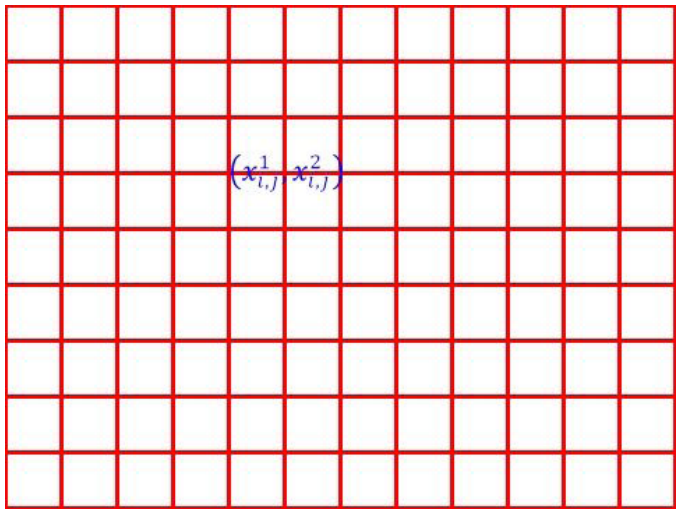
网格变形驱动的图像缩放

- **思想：**以目标尺寸为约束，对原始图像进行结构保持的变形



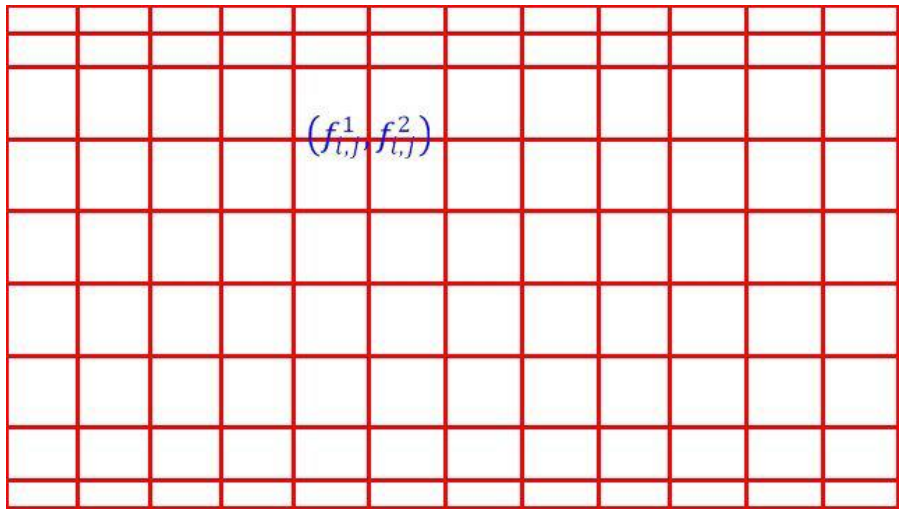
网格变形驱动的图像缩放

Embedding



Low-distortion Deformation

$$\min \sum_q D_q$$



Texture Mapping

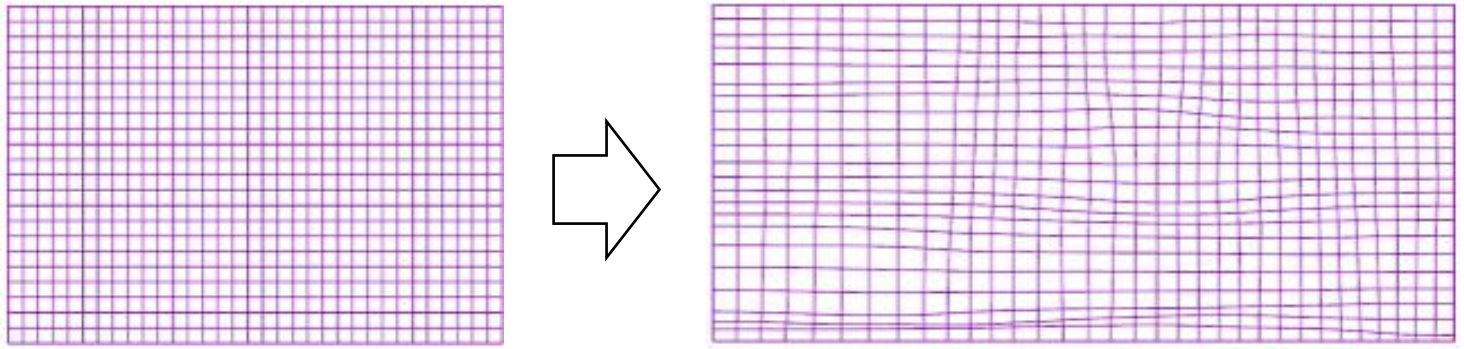
Distortions/dissimilarity

Conformal distortion
(stretch)

Isometric distortion
(Stretch + Scaling)

网格变形驱动的图像缩放

- **方法：** 网格变形驱动图像缩放

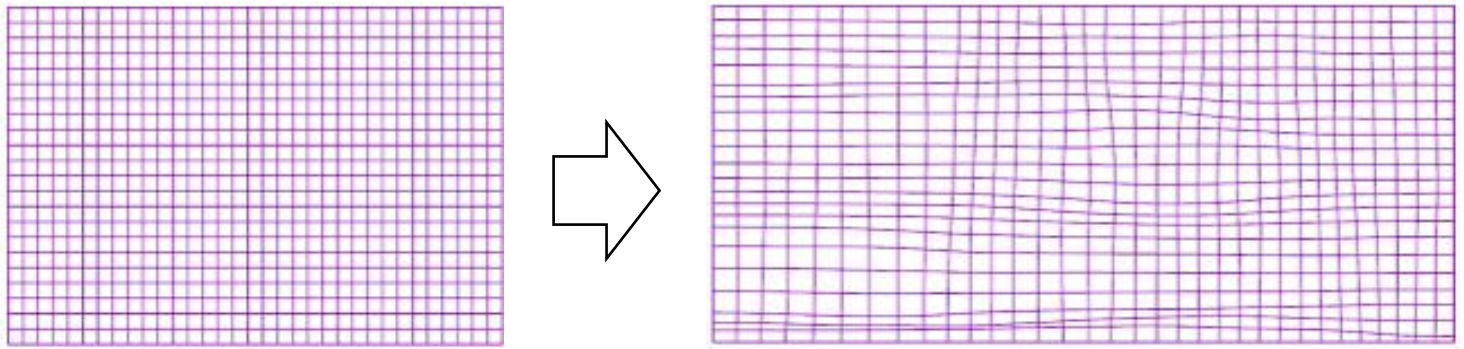


$f: \mathrm{v}_i \rightarrow \mathrm{v}_i'$

$\mathrm{v}' = s_f \mathrm{v} + \mathrm{t}$

网格变形驱动的图像缩放

- **方法：**网格变形驱动图像缩放 $D_l(f)$



均匀缩放 $D_u(f) = \sum_{\{i,j\}\in \mathrm{E}} \left\| \left(\mathrm{v}_i' - \mathrm{v}_j'\right) - s_f\left(\mathrm{v}_i - \mathrm{v}_j\right) \right\|^2$

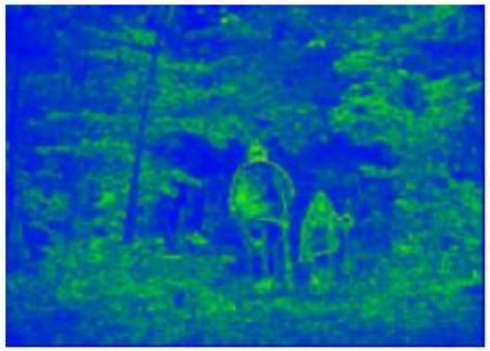
比例缩放 $D_l(f) = \sum_{\{i,j\}\in \mathrm{E}} \left\| \left(\mathrm{v}_i' - \mathrm{v}_j'\right) - l_{ij}\left(\mathrm{v}_i - \mathrm{v}_j\right) \right\|^2$

边界条件 $\mathrm{v}_0' = (0,0)^T \qquad \mathrm{v}_{\mathrm{end}}' = (n', m')^T \qquad \mathrm{v}_{i,y(x)}' = i/m'(n')$

网格变形驱动的图像缩放

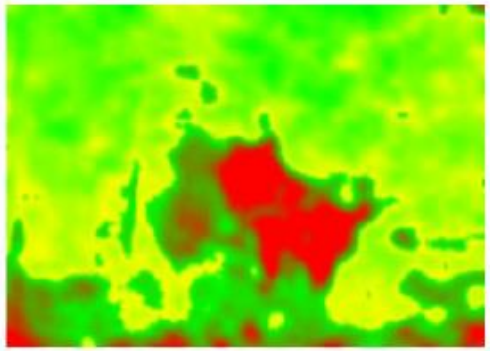
- **结构**
  - 图像梯度+图像显著度
    - 梯度：图像局部结构分部
    - 显著度：视觉注意区域
  - 变形时保持结构变化尽可能少

原始图



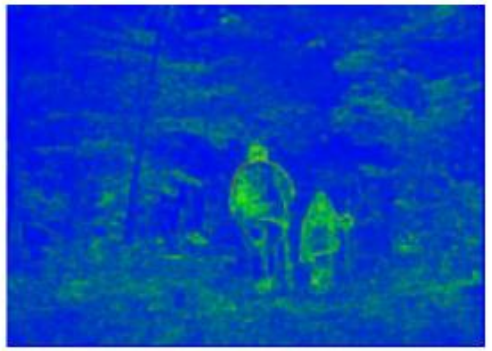
梯度图

X



显著图

=

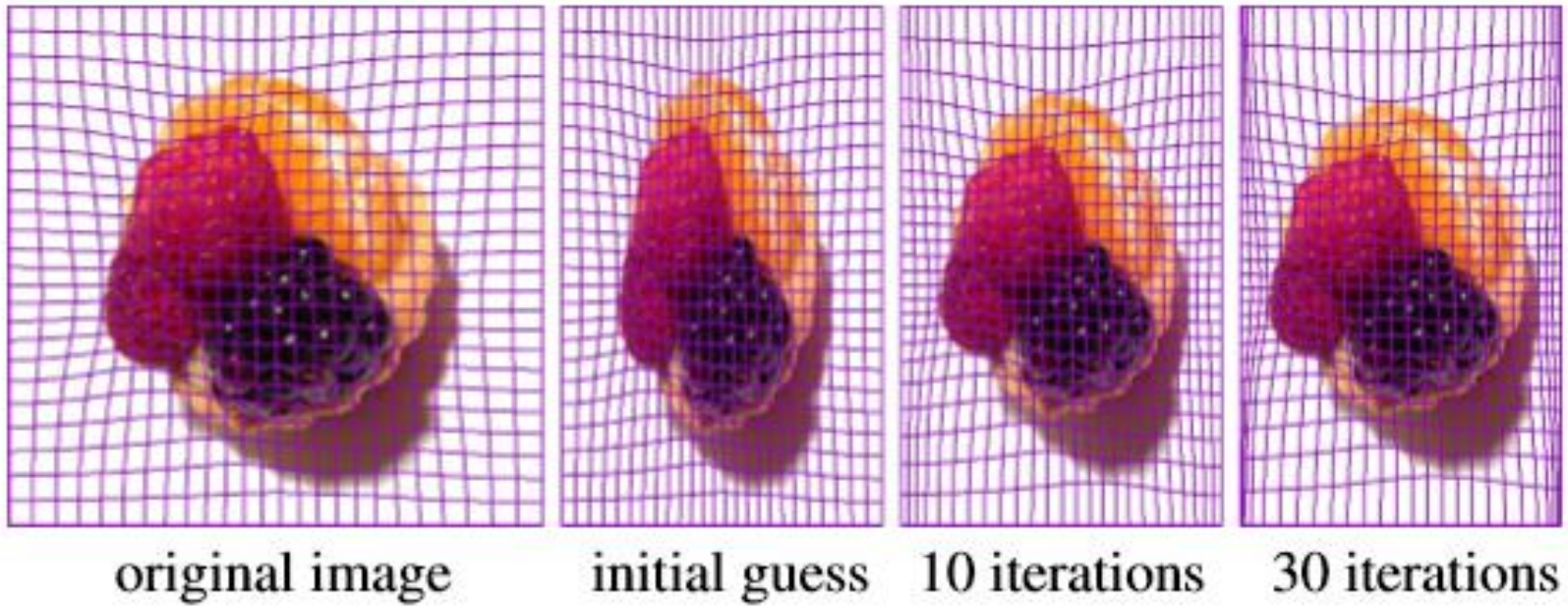


结构图

网格变形驱动的图像缩放

- **求解**
  - 自适应网格设置：更具结构重要性图放置网格顶点
  - 简单缩放作为初值：

$$\mathrm{v}' = s_f \mathrm{v} + \mathrm{t}$$



网格变形驱动的图像缩放

- 结果

原始图

缝隙增删

网格变形

局部线结构扭曲

中国科学技术大学

University of Science and Technology of China

谢　谢！