

# 湖南科技大学课程教案

## (章节、专题首页)

授课教师: 王志喜

职称: 副教授

单位: 计算机科学与工程学院

课程名称	计算机图形图像技术
章节、专题	数字图像处理概述
教学目标及基本要求	了解数字图像处理的研究内容和应用领域, 掌握图像数据格式, 熟悉用Dev-C++开发OpenCV应用的方法。
教学重点	图像数据格式, 用Dev-C++开发OpenCV应用
教学难点	图像数据格式
教学内容与时间分配	(1) 数字图像处理的研究内容和主要应用 (0.4课时) (2) 图像数据格式 (0.9课时) (3) OpenCV简介 (0.4课时) (4) 用Dev-C++开发OpenCV应用 (1.3课时) 共计3课时。
习题	第10.5.1节 (基础知识题) 和10.5.2节 (程序设计题)。

# 第2篇 数字图像处理

## 第10章 数字图像处理概述

### 10.1 数字图像处理的研究对象和应用领域

#### 10.10.1 数字图像

数字图像处理的研究对象是数字图像。

##### 1. 图像的含义

图像这个词包含的内容很广，凡是记录在纸上，拍摄在照片上，显示在屏幕上的所有具有视觉效果的画面都可以称为图像。

## 2. 图像的灰度信息

图像的灰度信息是指图像中各点处的颜色深浅程度信息（也就是明暗程度信息或亮度信息）。对于灰度图像来说，灰度就是黑白程度等级（也就是明暗程度等级或亮度）；对于彩色图像来说，情况也一样，因为任何彩色图像都可以分解成红、绿、蓝三种灰度图像，因此彩色图像的灰度指的是这三种灰度图像的灰度。

## 3. 图像的分类

根据图像记录方式的不同，图像可分为两大类。

- 模拟图像。通过某种物理量（光、电）的强弱变化来记录图像上各点的灰度信息（如模拟相机和模拟电视等）。
- 数字图像。用数值记录图像的灰度信息。

## 10.1.2 数字图像处理

### 1. 数字图像处理

所谓数字图像处理，就是指利用数字计算机及其他相关数字技术，对数字图像进行某种运算和处理，使图像中的某部分信息更加突出，以适应某种特殊需求，达到某种预期目的。例如，使褪色模糊了的照片重新变清晰，从医学显微图像中提取有意义的细胞特征等。

### 2. 基本的数字图像处理技术

若未加特别说明，图像处理通常是指使用计算机的数字图像处理。基本的数字图像处理技术包括图像数据格式、图像变换、图像增强、图像解析和图像数据压缩等。

### 3. 数字图像处理的研究内容<sup>1</sup>

数字图像处理的研究内容通常包括以下几个方面。

- 图像的数字化。研究如何把一幅连续的光学图像表示成一组数值，既不失真又便于计算机分析处理。
- 图像的增强。增强图像中的有用信息，削弱干扰和噪音，以便于观察、识别和进一步分析、处理。增强后的图像不一定和原图像一致。
- 图像恢复。使褪色或模糊的图像复原。复原的图像要尽可能和原图像保持一致。
- 图像编码。在满足指定保真度的前提下，简化图像表示，压缩表示图像的数据，便于存储和传输。
- 图像分析。对图像中的各种对象进行分割、分类、识别、描述、解释等。

## 10.1.3<sup>#</sup>. 数字图像处理与相关学科的关系

与数字图像处理密切相关的学科主要有计算机图形学、计算几何、计算机视觉和模式识别等。它们之间的关系如图10-1所示。

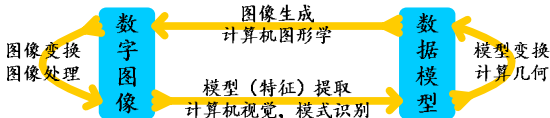


图10-1 数字图像处理与相关学科的关系

- 计算机图形学着重讨论怎样将数据模型变成数字图像。
- 图像处理着重研究图像的压缩存储和去除噪音等问题，以突出图像中感兴趣的部分或特征。
- 模式识别重点讨论如何从图像中提取数据和模型。
- 计算几何着重研究数据模型的建立、存储和管理。

随着技术的发展和应用的深入，这些学科的界限越来越模糊，各学科相互渗透、融合。一个较完善的应用系统通常综合利用了各个学科的相关技术。

## 10.1.4 数字图像处理的部分应用领域<sup>[1]</sup>

### 1. 航空和航天领域

在航空和航天领域，美国喷气推进实验室对月球、火星照片的处理是数字图像处理技术首次获得实际应用。此外，数字图像处理技术还广泛应用在飞机遥感和卫星遥感技术中。目前，世界各国都在利用陆地卫星获取的图像进行土地测绘、资源调查（如森林调查和水资源调查等）、气象监测、灾害检测（如病虫害检测和环境污染检测等）、矿产勘察和军事侦察等方面的工作。另外，在空中交通管制以及机场安检视频监控中，数字图像处理技术也得到了广泛的应用。

### 2. 生物和医学领域

数字图像处理技术在生物和医学领域的应用十分广泛，无论是临床诊断还是病理研究都采用数字图像处理技术，而且很有成效。数字图像处理技术的直观、无创伤和安全方便的优点获得了普遍接受。除了最成功的X射线和CT技术之外，还有对医用显微图像的处理分析等方面，如红细胞和白细胞分类、染色体分析、癌细胞识别等。

### 3. 军事和公安领域

在军事领域，图像处理和识别主要用于导弹的精确制导，各种侦察照片的判读，具有图像传输、存储和显示的军事自动化指挥系统，飞机、坦克和军舰模拟训练系统等。

在公安领域，图像处理和识别主要用于公安业务图片的判读分析，指纹识别和人脸鉴别，不完整图片的复原，以及交通监控与事故分析等。高速公路不停车自动收费系统中的车辆和车牌的自动识别都是数字图像处理技术成功应用的例子。

### 4. 工业和工程领域

在工业和工程领域中，数字图像处理技术也有着十分广泛的应用。例如，自动装配线中零件的质量检测和分类，印刷电路板疵病检查，弹性力学照片的应力分析，流体力学图片的阻力和升力分析，邮政信件的自动分拣，在有毒或放射性环境中识别物体的形状和排列状态，在先进制造技术中采用机器视觉等。

机器视觉作为智能机器人的重要感觉器官，主要进行三维景物的理解和识别。机器视觉主要用于军事侦察和危险环境的自主机器人，邮政、医院和家庭服务的智能机器人，装配线中工件的识别和定位，太空机器人的自动操作等。



## 5. 通信领域

当前通信的主要发展方向是声音、文字、图像和数据结合的多媒体通信，也就是将电话、电视和计算机以三网合一的方式在数字通信网上传输。其中以图像通信最为复杂和困难，原因是图像的数据量十分巨大。例如，传送彩色电视信号的速率就高达100Mbit/s以上。要将这样高速率的数据实时传送出去，必须采用某种编码技术来压缩信息的比特量。

## 6. 电子商务与文化艺术领域

在当前呼声甚高的电子商务中，数字图像处理技术也大有可为，如身份认证、产品防伪、水印技术等。

数字图像处理技术在文化艺术领域中的应用主要有电视画面的数字编辑，动画的制作，电子图像游戏，纺织工艺品设计，服装设计与管理，发型设计，文物资料照片的复制与修复，运动员动作分析与评分等。

## 10.2 图像数据

### 10.2.1 图像在计算机上的存在形式

#### 1. 位图

数字图像在计算机上是以位图形式存在的。位图是一个矩形点阵，每一个点都称为一个像素。像素是数字图像中的基本单位。一幅  $W \times H$  大小的图像，是由  $W \times H$  个明暗不等的像素组成的。

#### 2. 灰度值

在数字图像中，各像素的明暗程度由一个称为灰度值的数值标识。

通常将白色的灰度值定义为255，黑色的灰度值定义为0。由黑到白之间的明暗度均匀地划分成256个等级，每个等级由一个相应的灰度值定义，这样就定义了一个256个等级的灰度表。

任何一幅用这个灰度表记录的图像，它的每一个像素的灰度值都是由0~255之间的某一个数值标定的。显然，描述一个像素需要用8位二进制数据。

### 3. 灰度图像的处理

对于一幅灰度图像来说，256个等级的灰度变化足以描述它的各个细微部分。如果采用少于256个等级的灰度表，将发现图像上原来很清楚的细微部分会比较模糊，这是由于记录图像的信息不够而引起的。反之，如果采用多于256等级的灰度表，虽然从理论上说图像的表现会变得更加细致入微，但是，因为人眼很难分辨256个等级以上的灰度变化，采用多于256个等级的灰度表只会无益地增加图像的数据量。所以，采用256个等级的灰度表是比较理想的。

根据上面的讨论，每一个灰度图像中的像素都分别由一个字节记录。

### 4. 彩色图像的处理

在彩色图像中，每个像素需用三字节数据记录。这是因为任何彩色图像都可以分解成红、绿、蓝三个灰度图像，任何一种其他颜色都可以由这三种颜色混合而成。

在数字图像处理中，彩色图像的处理通常通过对三个灰度图像分别进行处理来实现。

## 10.2.2 图像的采样

### 1. 图像采样和灰度级量化的含义

通常的图像，如一幅画、一张照片，都能由一个二维连续函数  $f(x, y)$  来描述。其中  $(x, y)$  是图像平面上任意一个二维坐标点， $f$  指出该点颜色的深浅。

为了便于用计算机处理图像，图像  $f(x, y)$  必须对空间和颜色深浅的幅度都进行数字化。空间坐标  $(x, y)$  的数字化称为图像采样，而颜色深浅幅度的数字化称为灰度级量化。

## 2. 图像的点阵表示

假定连续图像  $f(x, y)$  被等距离取点采样形成一个  $W \times H$  的矩形点阵，则  $f(x, y)$  可用下式表示。

$$f(x, y) \approx \begin{pmatrix} f_{0,0} & f_{0,1} & \cdots & f_{0,W-1} \\ f_{1,0} & f_{1,1} & \cdots & f_{1,W-1} \\ \cdots & \cdots & \cdots & \cdots \\ f_{H-1,0} & f_{H-1,1} & \cdots & f_{H-1,W-1} \end{pmatrix}$$

上式右边的矩阵就是一幅数字图像，每一个元素称为一个像素。

## 3. 记录图像所需位数的计算

图像进行上述数字化处理时，关键是要决定  $W$  和  $H$  的大小及允许给每个像素赋予离散灰度值的灰度级数  $G$ 。为了便于处理，通常将灰度级数  $G$  设计成2的正整数幂，即  $G = 2^m$ 。这样，记录一幅图像所需的位数为  $b = W \times H \times m$ 。

#### 4. 采样点数和灰度级数的选取

图像的分辨率与采样点数和灰度级数紧密相关。

- 采样点数和灰度级数越大，数字图像就越接近原来的连续图像。
- 随着采样点数和灰度级数的增大，存储图像的空间以及处理图像所需的时间也同时迅速增加。
- 减少采样点数（缩小  $W$  值和  $H$  值）将使图像趋于模糊；降低信息记录位数  $m$  会使图像质量劣化。
- 要评价一幅图像质量的优劣通常很困难，因为对于图像质量的要求是随不同的应用目的而变的。
- 根据经验，要使一幅数字图像具有黑白电视画面的画质，应使  $W, H \geq 512, m = 8$ 。

## 5. 用数字化扫描仪对图像进行数字化处理

在用数字化扫描仪对图像进行数字化处理时，通常要给出分辨率和灰度级。

- 灰度级。由每个像素所需的位数给出，一般选择1, 4, 8, 24等值。
- 分辨率。通常由dpi值给出。dpi表示在每英寸长度上采样的像素数。例如，对一幅5×5平方英寸的图像进行扫描，如果选择100dpi，将生成一幅500×500像素的图像。

## 10.2.3 图像的数据格式

### 1. 像素数据的记录方法

图像是由排成矩形点阵的像素组成的。因此把一幅图像记录到文件中时，必须同时记录下各像素在点阵中的位置及像素的灰度值。但是实际上可以利用各像素在文件中的记录位置来暗示像素在图像点阵中的位置，这样就可以省去记录像素位置坐标的数据量，而各像素的数据只用来记录其灰度值。

在一个存储一幅  $W \times H$  图像的数据文件中， $W \times H$  个像素数据通常按照下列方式排列。

- 最初  $W$  个数据对应第1行从左到右的  $W$  个像素；
- 第  $W + 1$  到  $2W$  个数据对应第2行从左到右的  $W$  个像素；
- .....
- 最后  $W$  个数据对应第  $H$  行从左到右的  $W$  个像素。



## 2. 文件头

必须在文件中注明图像的尺寸（宽度与高度，单位为像素），以便在读取数据时能够根据该尺寸正确构造图像的二维点阵。

图像的尺寸通常记录在文件头中，文件头是有关图像整体的信息数据块，除记录图像的尺寸外，还记录诸如像素的位长、图像的颜色表等有关信息。文件头之后才是图像数据。

图像数据的文件格式随着图像的各种信息的内容取舍与记录次序的不同而异，其中，关于图像数据的记录方式基本相同，主要的差异在于文件头的内容。

这里只介绍目前应用较广，比较常见的BMP文件格式。

### 3. BMP文件格式

基于调色板的BMP文件由文件头、信息头、调色板和图像的像素数据四个部分组成，而真彩色的BMP文件则由文件头、信息头和图像的BGR像素数据组成。如图10-2所示。

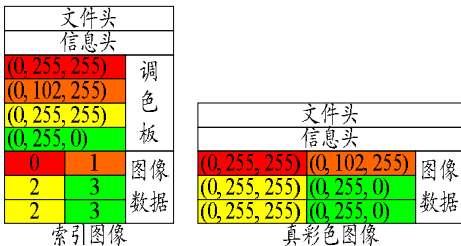


图10-2 BMP文件格式

(1) 文件头。文件头是一个BITMAPFILEHEADER数据结构，其内容如下  
(bfSize和bfOffBits各4字节，其余各2字节，共14字节)。

- **bfType**。图像文件类型，固定为BM。
- **bfSize**。图像文件大小。
- **bfReserved1**。保留，应设定为0。
- **bfReserved2**。保留，应设定为0。
- **bfOffBits**。图像数据偏移量，从文件起始位置到图像数据的距离。

文件头		调色板
信息头		
(0, 255, 255)		
(0, 102, 255)		
(0, 255, 255)		图像数据
(0, 255, 0)		
0	1	
2	3	
2	3	

索引图像

文件头		图像 数据
信息头		
(0, 255, 255)	(0, 102, 255)	
(0, 255, 255)	(0, 255, 0)	
(0, 255, 255)	(0, 255, 0)	

真彩色图像

(2) 信息头。信息头是一个BITMAPINFOHEADER数据结构，包含与设备无关的点阵位图的尺寸和颜色格式等信息，内容如下（biPlanes和biBitCount各2字节，其余各4字节，共40字节）。

- **biSize**。本数据结构的大小。
- **biWidth**。图像的宽度（列数，单位为像素）。
- **biHeight**。图像的高度（行数，单位为像素）。
- **biPlanes**。目标设备的平面数（帧数），固定为1。
- **biBitCount**。每个像素所需位数（位深度），通常选用1、4、8或24。
- **biCompression**。图像数据的压缩格式，通常选用BI\_RGB。
- **biSizeImage**。实际图像数据的大小（单位为字节，BI\_RGB格式为0）。
- **biXPelsPerMeter**。目标设备水平分辨率参考值（单位为像素/米，默认为0）。
- **biYPelsPerMeter**。目标设备垂直分辨率参考值（单位为像素/米，默认为0）。
- **biClrUsed**。图像实际使用的颜色数目，默认值0表示所有颜色。
- **biClrImportant**。显示图像时需要的颜色数目，默认值0表示所有颜色。

文件头		调色板
信息头		
(0, 255, 255)		
(0, 102, 255)		
(0, 255, 255)		
(0, 255, 0)		
0	1	图像数据
2	3	
2	3	

索引图像

文件头		图像 数据
信息头		
(0, 255, 255)	(0, 102, 255)	
(0, 255, 255)	(0, 255, 0)	
(0, 255, 255)	(0, 255, 0)	

真彩色图像

(3) 调色板。调色板也叫颜色表，是一个可变长的列表，列表的长度（列表中项的数目）由信息头中的biBitCount值决定（ $2^{\text{biBitCount}}$ ）。调色板中的每一项是一个RGBQUAD对象，数据结构RGBQUAD的内容如下（各1字节，共4字节）。

- **rgbBlue**。蓝色分量。
- **rgbGreen**。绿色分量。
- **rgbRed**。红色分量。
- **rgbReserved**。保留，必须为0。

(4) 图像数据。

- BMP图像像素数据的存储顺序是由下往上，从左往右。
- 像素数据的每一行都必须是4字节对齐的，即每一行像素数据的长度（单位为字节）都必须是4的倍数，如果不是4的倍数则必须补足。
- 虽然BMP文件的像素数据有BI\_RLE8及BI\_RLE4等几种压缩格式，但是BMP文件通常不采用压缩格式。
- 真彩色图像没有调色板，像素数据直接使用BGR颜色值。

文件头 信息头	
(0, 255, 255)	调色板
(0, 102, 255)	
(0, 255, 255)	
(0, 255, 0)	
0	1
2	3
2	3

索引图像

文件头 信息头	
(0, 255, 255)	(0, 102, 255)
(0, 255, 255)	(0, 255, 0)
(0, 255, 255)	(0, 255, 0)

真彩色图像

## 4. BMP文件格式举例

以大小为255×255像素的图像为例，分别说明单色BMP图像文件、16色BMP图像文件、256色BMP图像文件和真彩色图像文件的文件格式。

(1) 单色BMP图像文件。由文件头、信息头、调色板和像素数据四部分组成。

- 文件头。共5项。

bfType	bfSize	bfReserved1	bfReserved2	bfOffBits
BM	8222	0	0	62

- 信息头。共11项。

biSize	biWidth	biHeight	biPlanes	biBitCount	biCompression	biSizeImage	...
40	255	255	1	1	0	8610	0

- 调色板。共2项。

(0, 0, 0, 0)	(255, 255, 255, 0)
--------------	--------------------

- 像素数据。略。

(2) 16色BMP图像文件。由文件头、信息头、调色板和像素数据四部分组成。

- 文件头。共5项。

bfType	bfSize	bfReserved1	bfReserved2	bfOffBits
BM	32758	0	0	118

- 信息头。共11项。

biSize	biWidth	biHeight	biPlanes	biBitCount	biCompression	biSizeImage	...
40	255	255	1	4	0	32640	0

- 调色板。共16项。

(0, 0, 0, 0)	(0, 0, 128, 0)	(0, 128, 0, 0)	(0, 128, 128, 0)
(128, 0, 0, 0)	(128, 0, 128, 0)	(128, 128, 0, 0)	(128, 128, 128, 0)
(192, 192, 192, 0)	(0, 0, 255, 0)	(0, 255, 0, 0)	(0, 255, 255, 0)
(255, 0, 0, 0)	(255, 0, 255, 0)	(255, 255, 0, 0)	(255, 255, 255, 0)

- 像素数据。略。

(3) 256色BMP图像文件。由文件头、信息头、调色板和像素数据四部分组成。

- 文件头。共5项。

bfType	bfSize	bfReserved1	bfReserved2	bfOffBits
BM	66358	0	0	1078

- 信息头。共11项。

biSize	biWidth	biHeight	biPlanes	biBitCount	biCompression	biSizeImage	...
40	255	255	1	8	0	65280	0

- 调色板。共256项。

(0, 0, 0, 0)	(0, 0, 128, 0)	(0, 128, 0, 0)	(0, 128, 128, 0)
(128, 0, 0, 0)	(128, 0, 128, 0)	(128, 128, 0, 0)	(192, 192, 192, 0)
(192, 220, 192, 0)	(240, 202, 166, 0)	(0, 32, 64, 0)	(0, 32, 96, 0)
(0, 32, 128, 0)	(0, 32, 160, 0)	(0, 32, 192, 0)	(0, 32, 224, 0)
...	...	...	...

- 像素数据。略。



(4) 真彩色BMP图像文件。由文件头、信息头和像素数据三部分组成。

- 文件头。共5项。

bfType	bfSize	bfReserved1	bfReserved2	bfOffBits
BM	195894	0	0	54

- 信息头。共11项。

biSize	biWidth	biHeight	biPlanes	biBitCount	biCompression	biSizeImage	...
40	255	255	1	24	0	195840	0

- 像素数据。略。

## 10.3 OpenCV简介

### 10.3.1 OpenCV概述<sup>♫</sup>

#### 1. 什么是OpenCV

OpenCV是一个开源的计算机视觉库，包含数百种数字图像处理和计算机视觉方面的通用算法。

OpenCV提供基于C的OpenCV 1.x API和基于C++的OpenCV 2.x API这两种API。

自OpenCV 2.4发布以来，OpenCV推荐使用基于C++的OpenCV 2.x API，而不是使用基于C的OpenCV 1.x API，OpenCV 4甚至还废弃了一些C API。

本书使用OpenCV 2.x API，并简称为OpenCV。

## 2. OpenCV的常用模块

OpenCV具有模块化结构，包含多个共享库或静态库。下面是一些常用的模块。

- 核心功能模块（core）。定义基本数据结构的模块，包括密集的多维数组和所有其他模块使用的基本功能。
- 高层用户交互模块（imgcodecs, highgui）。包括图像文件的读写和一个易于使用的简单UI界面。
- 图像处理模块（imgproc）。包括图像的滤波、几何变换、形态学处理、颜色空间转换和直方图计算等图像处理函数。
- 视频处理与分析模块（video, videoio）。包括视频序列的读写、运动估计和对象跟踪等算法的实现。
- 二维特征模块（features2d）。包括图像的特征检测器、特征描述符和描述符匹配器等。
- 对象检测模块（objdetect）。检测预定义的对象，如杯子、汽车和人脸等。
- 3D模块（calib3d）。包括摄像机校准、对象姿态估计、立体匹配、三维信息重建等算法。
- 机器学习模块（ml）。一组用于分类、回归和聚类等方面的类和算法。

## 10.3.2 OpenCV的命名约定

### 1. 名字空间

OpenCV中的C++类和函数等资源都定义在命名空间cv内，有两种方法可以访问。

- 在使用这个命名空间的相关资源时带上cv前缀，如cv::Mat表示使用命名空间cv中的Mat。
- 在代码开头的适当位置加上“using namespace cv;”，使得在使用这个命名空间的相关资源时只要不产生歧义就不需要带上cv前缀。

## 2. 函数命名

使用驼峰命名法。

<Action><Target>

Action=核心功能（例如获取get，输出put）

Target=操作目标（例如轮廓Contour，多边形Polygon）

例如，putText表示在图像中输出一串文本。

## 3. 类型命名

使用Pascal命名法。

<Type>[Mod]

Type=类型核心名字（例如矩阵Mat，点Point）

Mod=可选修饰词（例如说明成员类型）

例如，Point2f表示32位浮点数二维点坐标。

## 4. 矩阵数据类型

矩阵的数据类型包含位深度和通道数。通道数就是每个元素数据的分量数（例如，真彩色BMP图像的通道数为3），位深度就是每个元素中各通道的基本数据类型（例如，真彩色BMP图像的位深度为字节）。

OpenCV中说明矩阵数据类型的格式为“CV\_<位数>{SUF}C<通道数>”。其中，S表示带符号整数，U表示无符号整数，F表示浮点数。

例如，CV\_8UC1表示单通道的8位无符号数矩阵，CV\_16SC2表示双通道的16位符号数矩阵，CV\_32FC3表示3通道的32位浮点数矩阵。

格式中的“CV\_<位数>{SUF}”称为位深度，表示每个元素中各通道的基本数据类型。

### 【注】

- 位深度只能是CV\_8U、CV\_8S、CV\_16U、CV\_16S、CV\_32S、CV\_32F、CV\_64F或CV\_USRTYPE1（共8种选择）。
- 当通道数为1时，格式中的“C<通道数>”可以省略。例如，CV\_8UC1和CV\_8U都可以表示单通道的8位无符号数矩阵。

## 5. 头文件

一般来说，对于OpenCV，需要使用哪一模块的内容就使用哪个模块作为头文件。例如，如果需要使用Mat类，因为Mat类属于core模块，那么头文件就是opencv2/core/core.hpp，如果需要使用imread()或imshow()等高层交互函数，那么头文件就是opencv2/highgui/highgui.hpp。

一种更直接的方法就是只使用头文件opencv2/opencv.hpp，因为该文件引入了OpenCV中所有模块的hpp文件。

### 10.3.3 OpenCV程序实例

下列程序读入并显示一幅彩色图像。程序运行结果如图10-3所示。

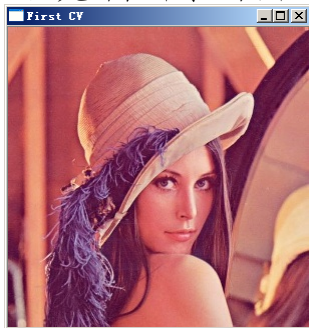


图10-3 OpenCV程序实例



```
// FirstCV.Cpp
#include<opencv2/opencv.hpp>
using namespace cv;

int main()
{   Mat im = imread("lena.jpg"); // 载入彩色图像
    if(im.empty()) return -1; // 载入失败
    namedWindow("First CV"); // 创建窗口
    imshow("First CV", im); // 显示图像
    waitKey(); // 等待按键
}
```

## 10.4 用Dev-C++开发OpenCV应用<sup>♫</sup>

### 10.4.1 必备工具的下载和安装

#### 1. Dev-C++的下载和安装

Dev-C++原始版本为Bloodshed Dev-C++，最高版本号为4.9.9.2，4.9.9.2以后的版本为Orwell Dev-C++。Orwell Dev-C++的下载地址为

<http://orwelldevcpp.blogspot.com/>

从该网站下载最新的不带编译器的Orwell Dev-C++。下载完成后直接运行安装程序，按照提示完成安装。注意，安装完成以后不要立即运行Orwell Dev-C++。

## 2. 编译器的下载和安装

使用Orwell Dev-C++编写程序通常使用GCC编译器。这里选用MinGW-W64 GCC-8.1.0的i686-posix-dwarf（编译OpenCV必须使用POSIX线程支持的版本），下载地址是

[https://sourceforge.net/projects/mingw-w64/files/Toolchains%20targetting%20Win32/Personal%20Builds/mingw-builds/8.1.0/thread-s-posix/dwarf/i686-8.1.0-release-posix-dwarf-rt\\_v6-rev0.7z](https://sourceforge.net/projects/mingw-w64/files/Toolchains%20targetting%20Win32/Personal%20Builds/mingw-builds/8.1.0/thread-s-posix/dwarf/i686-8.1.0-release-posix-dwarf-rt_v6-rev0.7z)

下载完成后解压该文件，将解压得到的mingw32文件夹复制到Dev-C++的安装目录（例如C:\Dev-Cpp）。

然后运行Orwell Dev-C++，按照提示完成Orwell Dev-C++的配置和编译器的默认配置。

**【注】**为了方便编译OpenCV，这里需要将MinGW的可执行文件目录（例如C:\Dev-Cpp\mingw32\bin）加入Path环境变量，并且，还需要保证Path环境变量中只有这一份GCC的可执行文件目录。

### 3. cmake-3.13.2-win32-x86

用于编译OpenCV，下载地址为

<https://github.com/Kitware/CMake/releases/download/v3.13.2/cmake-3.13.2-win32-x86.msi>

下载完成后直接运行安装程序，按照提示完成安装。

## 10.4.2 OpenCV的编译

### 1. 下载源程序文件

OpenCV 4.5.5是OpenCV 4系列的一个比较成熟的版本，是OpenCV官网于2021年12月30日发布的。OpenCV 4.5.5源程序文件的下载地址为

<https://codeload.github.com/opencv/opencv/zip/4.5.5>

下载完成后解压缩该文件，记住解压缩位置，例如D:\OpenCV-4.5.5。

## 2. 配置项目

(1) 启动CMake (cmake-gui)。可以从开始菜单启动。

(2) 分别指定OpenCV源文件位置(例如D:\OpenCV-4.5.5)和目标文件位置(例如D:\CV455)，如图10-4所示。

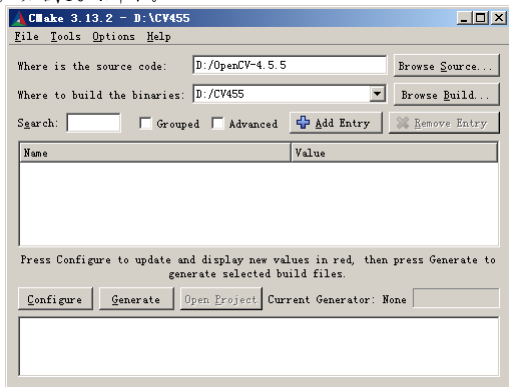


图10-4 源文件和目标文件位置

(3) 点击Configure，弹出选择环境和编译器的对话框。在列表选取MinGW Makefiles，编译器选用本机默认的编译器。如图10-5所示。

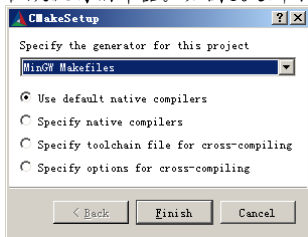


图10-5 选择环境和编译器

(4) 点击Finish，等待选项窗格中出现一片红色的选项。如图10-6所示。

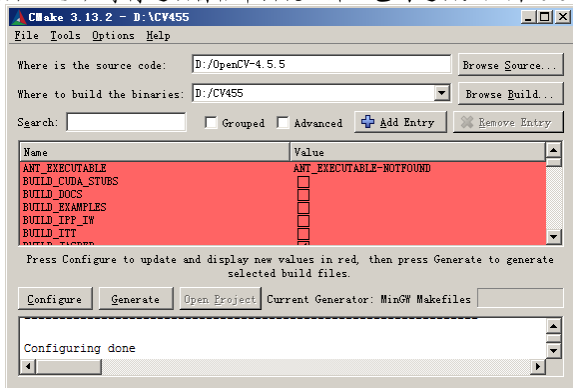


图10-6 初始选项



(5) 如果希望目标程序能够在Windows XP机器上运行，则需要找到OPENCV\_CMAKE\_MACRO\_WIN32\_WINNT选项，将该选项的值从0x0601（对应Windows 7）改成0x0501（对应Windows XP）。如图10-7所示。可以使用Search框搜索该选项。

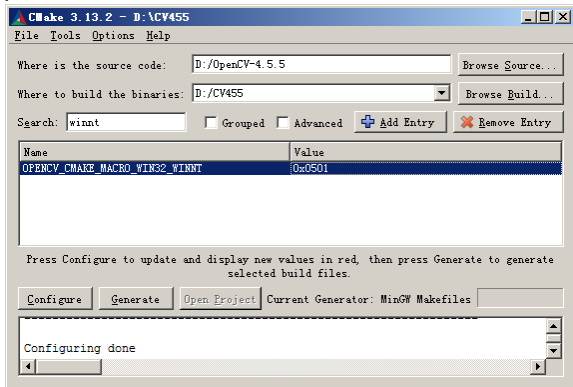


图10-7 修改OPENCV\_CMAKE\_MACRO\_WIN32\_WINNT选项

(6) 再次点击Configure，等待红色选项消失，显示Configuring done。如图10-8所示。

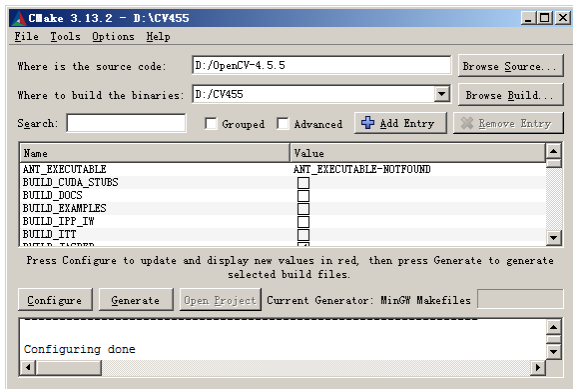


图10-8 完成配置

(7) 点击Generate, 生成项目相关文件。完成后会显示Generating done。如图10-9所示。

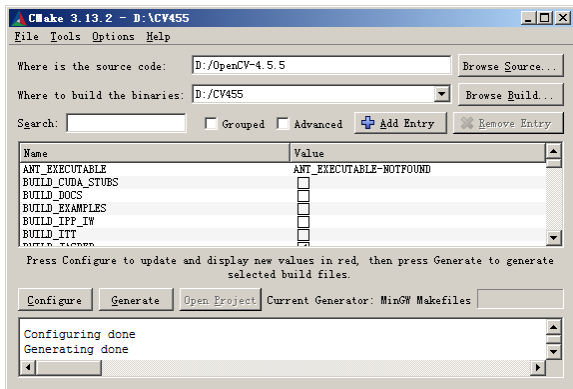


图10-9 生成相关文件

### 3. 编译项目

打开命令行窗口，将工作目录切换到目标文件夹（例如D:\CV455）。输入命令mingw32-make install，开始编译。如图10-10所示。

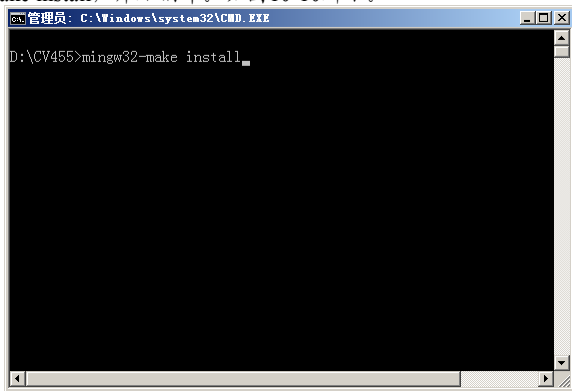
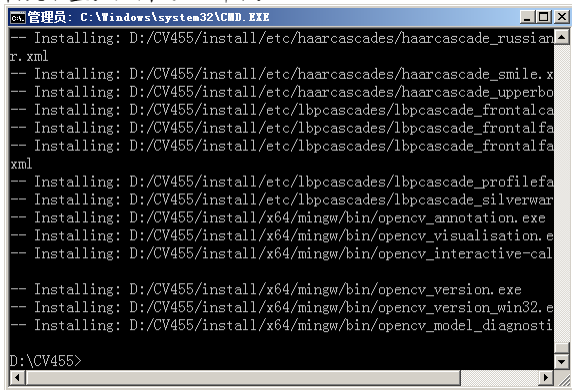


图10-10 开始编译

编译过程比较耗时，大约1到2小时。编译完成以后，mingw32-make install命令将自动完成相关配置。如图10-11所示。



```
管理员: C:\Windows\system32\CMD.EXE
-- Installing: D:/CV455/install/etc/haarcascades/haarcascade_russian
r.xml
-- Installing: D:/CV455/install/etc/haarcascades/haarcascade_smile.x
-- Installing: D:/CV455/install/etc/haarcascades/haarcascade_upperbo
-- Installing: D:/CV455/install/etc/lbpcascades/lbpcascade_frontalca
-- Installing: D:/CV455/install/etc/lbpcascades/lbpcascade_frontalfa
-- Installing: D:/CV455/install/etc/lbpcascades/lbpcascade_frontalfa
xml
-- Installing: D:/CV455/install/etc/lbpcascades/lbpcascade_profilefa
-- Installing: D:/CV455/install/etc/lbpcascades/lbpcascade_silverwar
-- Installing: D:/CV455/install/x64/mingw/bin/opencv_annotation.exe
-- Installing: D:/CV455/install/x64/mingw/bin/opencv_visualisation.e
-- Installing: D:/CV455/install/x64/mingw/bin/opencv_interactive-cal

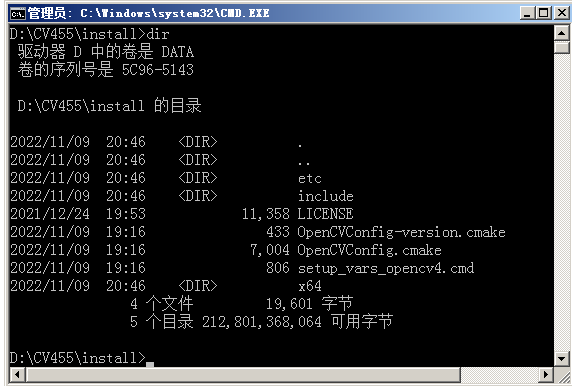
-- Installing: D:/CV455/install/x64/mingw/bin/opencv_version.exe
-- Installing: D:/CV455/install/x64/mingw/bin/opencv_version_win32.e
-- Installing: D:/CV455/install/x64/mingw/bin/opencv_model_diagnosti

D:\CV455>
```

图10-11 完成配置

## 4. 安装

完成以后，所有目标文件位于目标文件夹的install子目录中（例如，如图10-12所示的D:\OpenCV455\install）。可以从该目录或其子目录中找到bin、include和lib这三个文件夹，将这三个文件夹复制到待安装目录即可（例如C:\Dev-Cpp）。



```
管理员: C:\Windows\system32\CMD.EXE
D:\CV455\install>dir
驱动器 D 中的卷是 DATA
卷的序列号是 5C96-5143

D:\CV455\install 的目录

2022/11/09  20:46    <DIR>          .
2022/11/09  20:46    <DIR>          ..
2022/11/09  20:46    <DIR>          etc
2022/11/09  20:46    <DIR>          include
2021/12/24  19:53             11,358 LICENSE
2022/11/09  19:16             433 OpenCVConfig-version.cmake
2022/11/09  19:16             7,004 OpenCVConfig.cmake
2022/11/09  19:16             806 setup_vars_opencv4.cmd
2022/11/09  20:46    <DIR>          x64
               4 个文件          19,601 字节
               5 个目录 212,801,368,064 可用字节

D:\CV455\install>
```

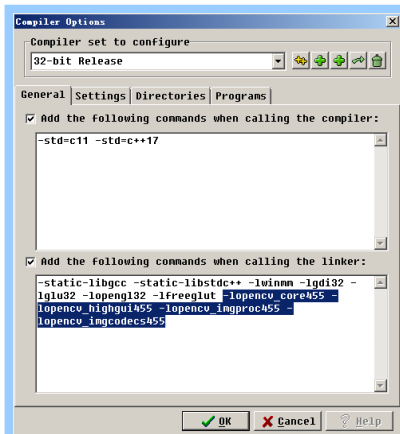
图10-12 目标文件位置

## 10.4.2 编译器设置与IDE的使用

### 1. 编译器设置

(1) 设置方法。从Tools菜单中选择Compiler Options打开Compiler Options，在Compiler Options中完成编译器设置。

(2) 语言支持。使用C++编写的OpenCV应用程序一般都需要使用到C++11新增的特性，可以使用选项-std=c++11使得编译器支持C++ 11。因为GCC-8.1.0支持C++ 17，本书使用了选项-std=c++17（如图10-13所示）。



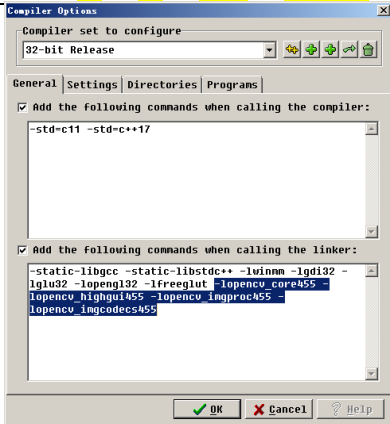
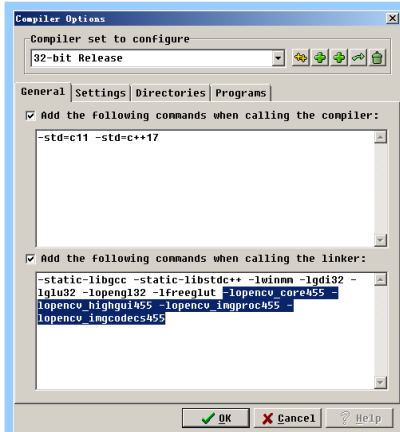


图10-13 语言支持和链接库设置



(3) 链接库。在Dev-C++中编译OpenCV图像处理应用程序需要包括libopencv\_core455.dll.a、libopencv\_highgui455.dll.a、libopencv\_imgproc455.dll.a和libopencv\_imgcodecs455.dll.a等静态链接库，可以使用-lopencv\_core455、-lopencv\_highgui455、-lopencv\_imgproc455和-lopencv\_imgcodecs455等选项设置这些链接库文件（如图10-13所示）。

为了保证能够链接到C++标准库，还需加上-static-libstdc++(如图10-13所示)。



(4) 可执行文件路径。首先从Directories页中选择Binaries，然后将编译和运行应用程序所需要的可执行文件的路径添加到列表中，最后调整这些路径的顺序（如图10-14所示）。

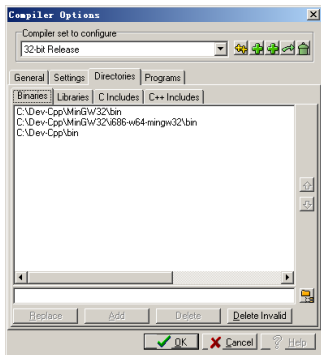


图10-14 可执行文件路径设置

(5) 链接库文件路径。首先从Directories页中选择Libraries，然后将编译应用程序所需要的链接库文件的路径添加到列表中，最后调整这些路径的顺序（如图10-15所示）。

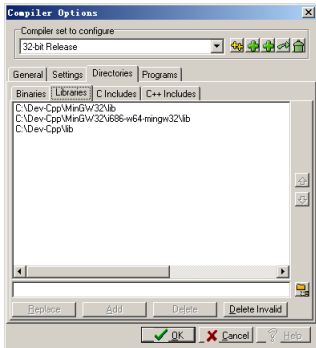


图10-15 链接库文件路径设置

(6) C Include文件路径。首先从Directories页中选择C Includes，然后将编译应用程序所需要的C Include文件的路径添加到列表中，最后调整这些路径的顺序（如图10-16所示）。

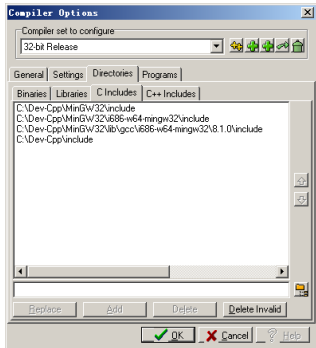


图10-16 C Include文件路径设置

(7) C++ Include文件路径。首先从Directories中选择C++ Includes，然后将编译应用程序所需要的C++ Include文件的路径添加到列表中，最后调整这些路径的顺序（如图10-17所示）。

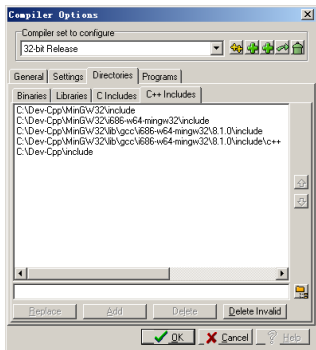


图10-17 C++ Include文件路径设置

(8) 编译器文件。在Programs页中指定编译器各主要组成文件的文件名（如图10-18所示）。

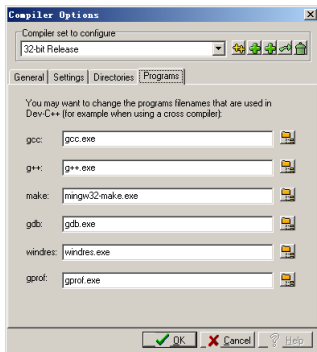


图10-18 编译器文件路径设置

## 2. IDE的使用方法

在完成开发环境的安装和编译器的设置以后，使用Dev-C++开发OpenCV应用的方法非常简单，打开Dev-C++集成开发环境，创建一个空白源程序文件，编写源程序、编译、调试、运行。

## 10.5 练习题

### 10.5.1 基础知识题

1. 请计算大小为 $720 \times 450$ 的真彩色图像至少需要占用多少字节的存储空间？（结果为972,000）
2. 大小为 $720 \times 450$ 的真彩色图像保存为真彩色BMP图像文件，请计算该文件的长度是多少字节。（结果为972,054）
3. 大小为 $450 \times 720$ 的真彩色图像保存为真彩色BMP图像文件，请计算该文件的长度是多少字节。（结果为973,494）



## 10.5.2 程序设计题

1. 随意从互联网下载一幅真彩色图像，使用OpenCV编写一个读入并显示该图像的程序，要求使用真彩色图像和灰度图像两种方式显示。相关函数的使用请参阅例题和OpenCV手册。
2. 请根据BMP文件的格式编写一个C++函数 `vector<uchar> loadBmp24(const string &path)`。该函数用于从一个真彩色BMP文件中读取图像数据，保存在uchar数组中（可以忽略非真彩色BMP文件）。