# 实验四：机器视觉

## 实验目的

* 机器视觉
* 机器语音

## 实验原理

## 机器视觉

### 1.1 机器视觉概述

#### 1.1.1 机器视觉系统分类

视觉系统就是用机器代替人眼来做测量和判断。视觉系统是指通过机器视觉产品（即图像摄取装置，分 [CMOS](https://baike.baidu.com/item/CMOS) 和[CCD](https://baike.baidu.com/item/CCD" \t "https://baike.baidu.com/item/%E6%9C%BA%E5%99%A8%E8%A7%86%E8%A7%89%E7%B3%BB%E7%BB%9F/_blank) 两种）将被摄取目标转换成图像信号，传送给专用的图像处理系统，根据像素分布和亮度、颜色等信息，转变成数字化信号；图像系统对这些信号进行各种运算来抽取目标的特征，进而根据判别的结果来控制现场的设备动作。是用于生产、装配或包装的有价值的机制。它在检测缺陷和防止缺陷产品被配送到消费者的功能方面具有不可估量的价值。

[机器视觉系统](https://baike.baidu.com/item/%E6%9C%BA%E5%99%A8%E8%A7%86%E8%A7%89%E7%B3%BB%E7%BB%9F)的特点是提高生产的柔性和自动化程度。在一些不适合于人工作业的危险工作环境或人工视觉难以满足要求的场合，常用机器视觉来替代人工视觉；同时在大批量工业生产过程中，用人工视觉检查产品质量效率低且精度不高，用机器视觉检测方法可以大大提高[生产效率](https://baike.baidu.com/item/%E7%94%9F%E4%BA%A7%E6%95%88%E7%8E%87)和生产的自动化程度。而且[机器视觉](https://baike.baidu.com/item/%E6%9C%BA%E5%99%A8%E8%A7%86%E8%A7%89/7414484)易于实现信息集成，是实现计算机集成制造的基础技术。可以在最快的生产线上对产品进行测量、引导、检测、和识别，并能保质保量的完成生产任务。

#### 1.1.2 机器视觉系统组成

视觉系统一般由照明光源、镜头、摄像头、图像处理单元构成。目前常用的有由数字摄像机构成的视觉系统：需要连接PC借助于PC上运行的处理软件来完成工作；有智能摄像机构成的视觉系统：处理单元和软件集成在摄像机中。

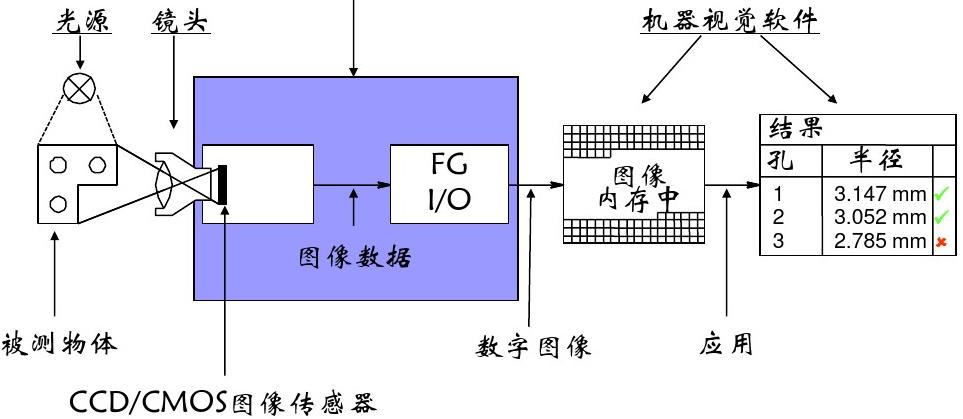


图 数字摄像机构成的视觉系统

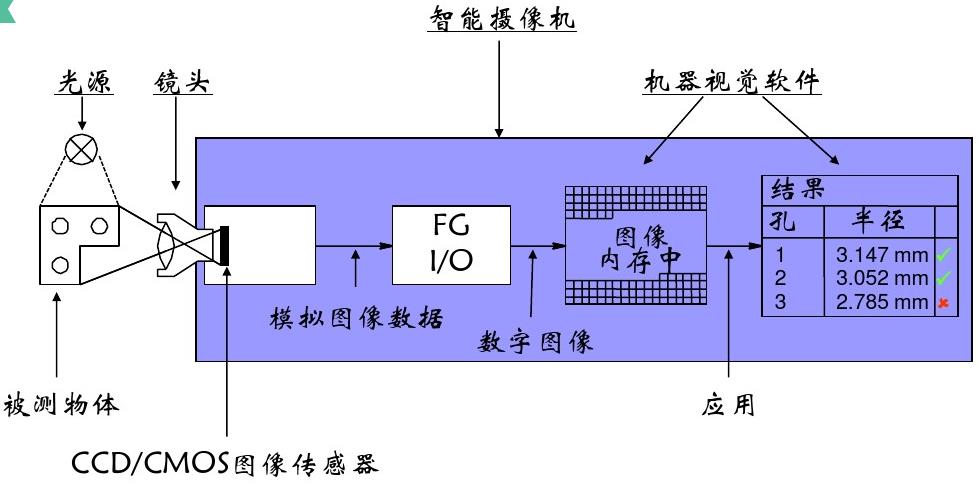


图 智能摄像机构成的视觉系统

##### 1.1.2.1 照明光源

照明是影响机器视觉系统输入的重要因素，它直接影响输入数据的质量和应用效果。由于没有通用的机器视觉照明设备，所以针对每个特定的应用实例，要选择相应的照明装置，以达到最佳效果。光源可分为可见光和不可见光。常用的几种可见光源是白帜灯、日光灯、水银灯和钠光灯。可见光的缺点是光能不能保持稳定。如何使光能在一定的程度上保持稳定，是实用化过程中急需要解决的问题。另一方面，环境光有可能影响图像的质量，所以可采用加防护屏的方法来减少环境光的影响。

照明系统按其照射方法可分为：背向照明、前向照明、结构光和频闪光照明等。其中，背向照明是被测物放在光源和摄像机之间，它的优点是能获得高对比度的图像。前向照明是光源和摄像机位于被测物的同侧，这种方式便于安装。结构光照明是将光栅或线光源等投射到被测物上，根据它们产生的畸变，解调出被测物的三维信息。频闪光照明是将高频率的光脉冲照射到物体上，摄像机拍摄要求与光源同步。

照明光源用于对被测物打光，体现出被测物的特征。使用不同的光源和照明技术可以达到较为理想的成像效果以突出被测物的特征。

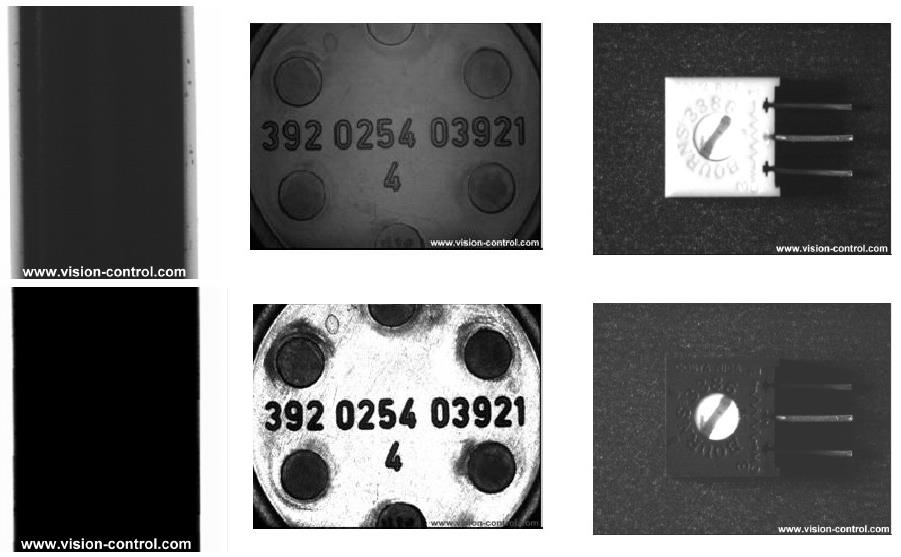


图 光源选型对成像的影响

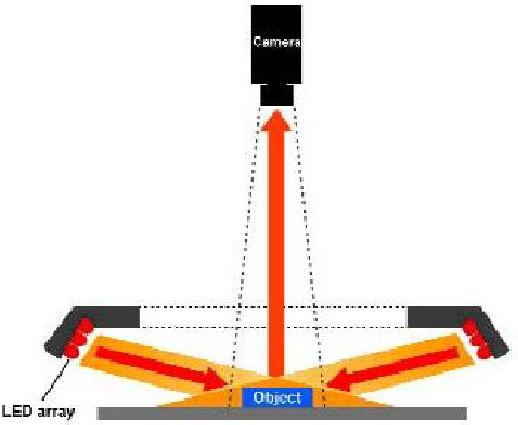


图 低角度照

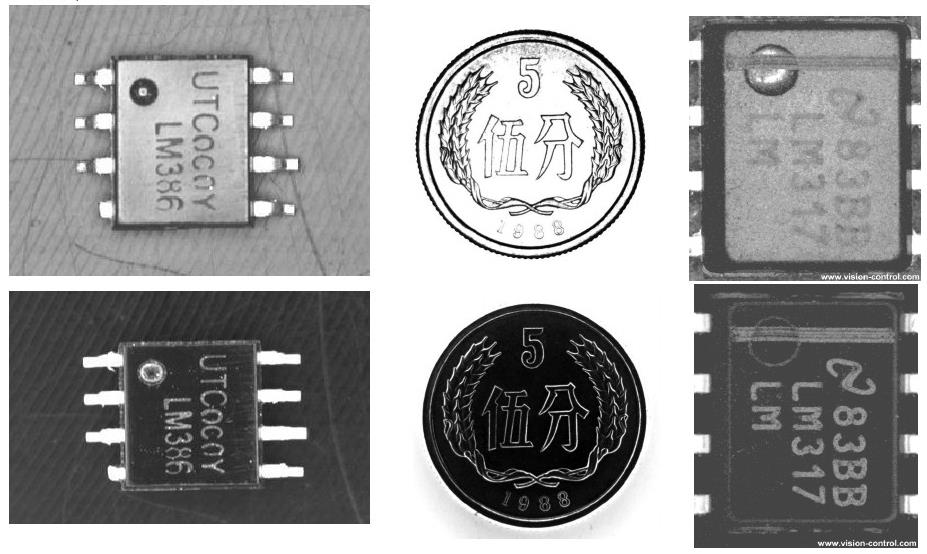


图 低角度照明效果

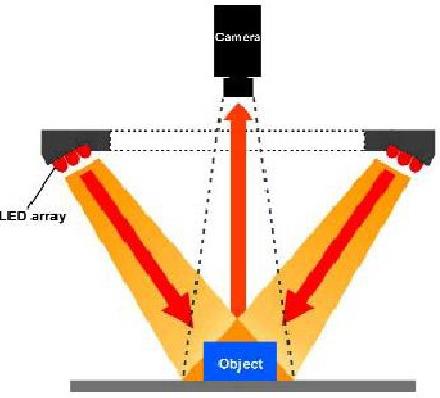


图 前向直射照明



图 前向直射照明效果

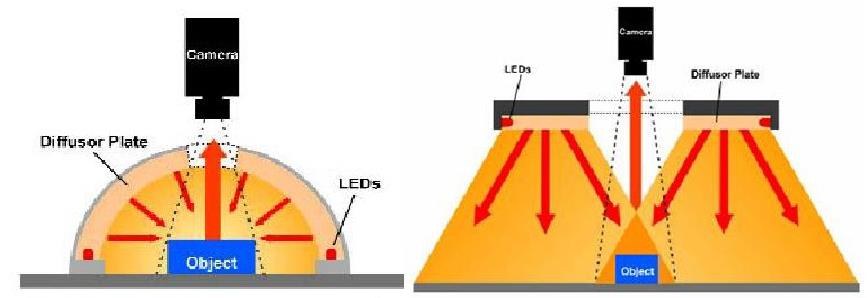


图 前向漫反射照明

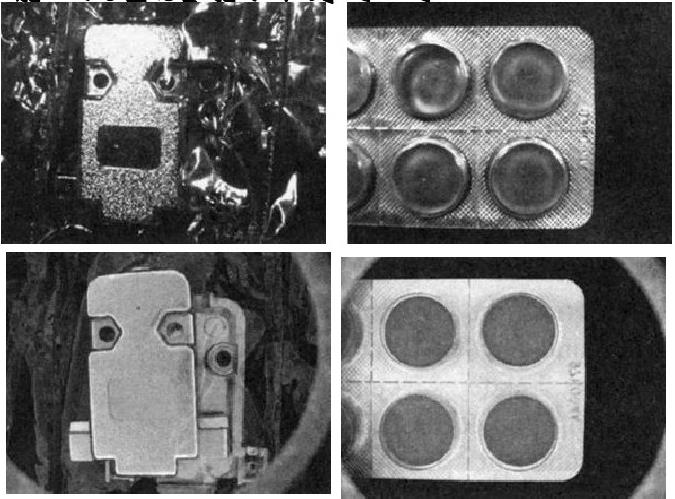


图 前向漫反射照明效果

##### 1.1.2.2 镜头

镜头是摄像机用于生成影像的光学部件，其选型和好坏直接决定了成像的效果。根据镜头的性能及外形区分，目前有P型、E型、L型和自动[变焦镜头](https://baike.baidu.com/item/%E5%8F%98%E7%84%A6%E9%95%9C%E5%A4%B4" \t "https://baike.baidu.com/item/%E9%95%9C%E5%A4%B4/_blank)等类型。具体如下：

1. P型镜头

（1） 自动定位镜头，本身瞳焦已经调节好，需要检验从最大倍率到最小倍率的[清晰度](https://baike.baidu.com/item/%E6%B8%85%E6%99%B0%E5%BA%A6" \t "https://baike.baidu.com/item/%E9%95%9C%E5%A4%B4/_blank)，是否一致、是否清晰。

（2） 检验同轴度，即最大倍率到最小倍率取像在同一位置，不能偏移或偏移太大，均视为不良品，必需重新更换镜头。

（3）光学放大倍率为0.7—4.5X，即0.7倍到4.5倍之间共九种倍率。

（4） 清晰度根据校正块、实际对象成像反映来进行判断。

2. E型镜头

（1） 此镜头为普通工业镜头，需要手动调节瞳焦，在机台安装好以后，手动调节使用最大倍率和最小倍率时，图像同样的清晰，如果不能调节清晰度视为不良品，如果调节后镜头有晃动等不稳定因素存在，也视为不良品。

（2）检验[同轴度](https://baike.baidu.com/item/%E5%90%8C%E8%BD%B4%E5%BA%A6" \t "https://baike.baidu.com/item/%E9%95%9C%E5%A4%B4/_blank)，即最大倍率到最小倍率取像在同一位置，不能偏移或偏移太大，均视为不良品，必需重新更换镜头。

（3）光学放大倍率为0.7—4.5X。

（4）清晰度根据校正块、实际对象成像反映来进行判读。

3. L型镜头

（1）此镜头为普通工业镜头，需要手动调节瞳焦，在机台安装好以后，手动调节使最大倍率和最小倍率时，图像同样的清晰，如果不能调节清晰度视为不良品，如果调节后镜头有晃动等不稳定因素存在，也视为不良品。

（2）检验同轴度，即最大倍率到最小倍率取像在同一位置，不能偏移或偏移太大，均视为不良品，必需重新更换镜头。

（3）光学放大倍率为0.7—4.5X。

（4）清晰度根据校正块、实际对象成像反映来进行判读。

4. 自动变焦镜头

（1） 为自动定位镜头本身瞳焦已经调节好，需要检验从最大倍率到最小倍率的清晰度，是否一致、是否清晰。

（2） 检验同轴度，即最大倍率到最小倍率取像在同一位置，不能偏移或偏移太大，均视为不良品，必需重新更换镜头。

（3） 光学放大倍率为0.7—4.5X。

（4） 清晰度根据校正块、实际对象成像反映来进行判读。

关于镜头需要了解的概念主要有：工作距离、焦距、视场角及景深等。

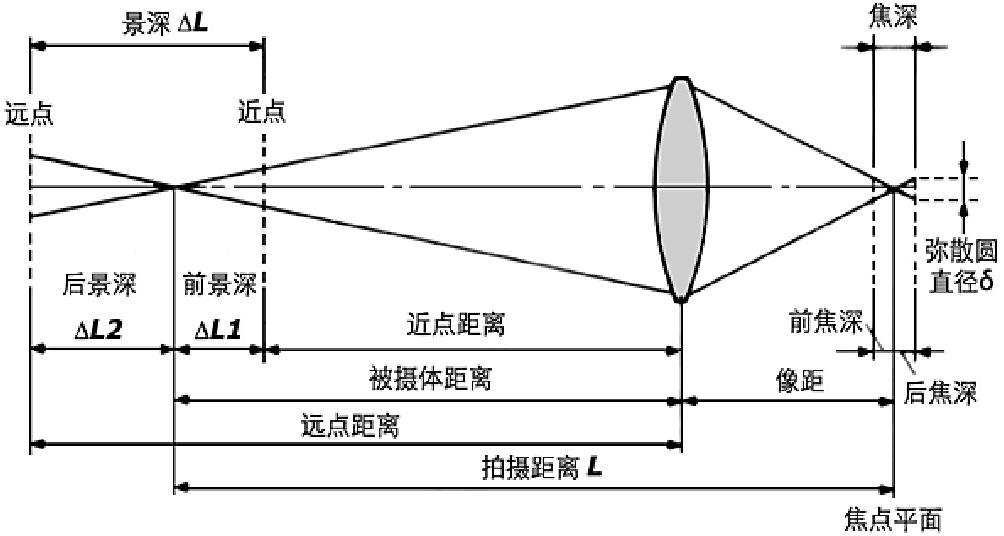


图 镜头成像原理1

焦距：镜片到感光芯片之间的距离，通常用f表示；

工作距离：镜片到被测物之间的距离，也叫物距，通常用D表示；

景深：可成像的远点和近点之间的距离称为景深；

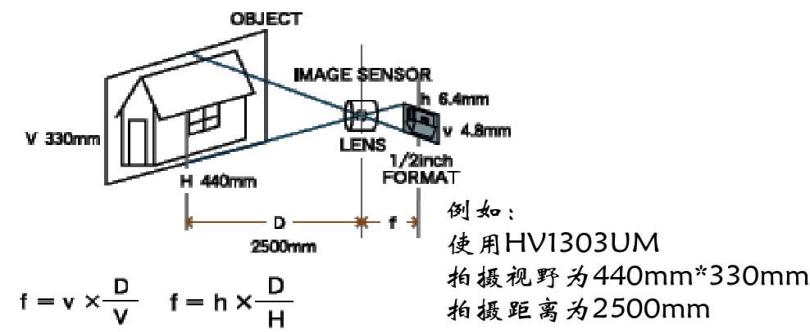


图 镜头成像原理2

镜头的视场角：镜头的视场角决定了镜头的视野范围，也就是说在相同工作距离的情况下，视场角越大视野范围越大，而从上图可以看出视场角是受到镜头焦距影响的，焦距越小则视场角越大，视野范围也越大。

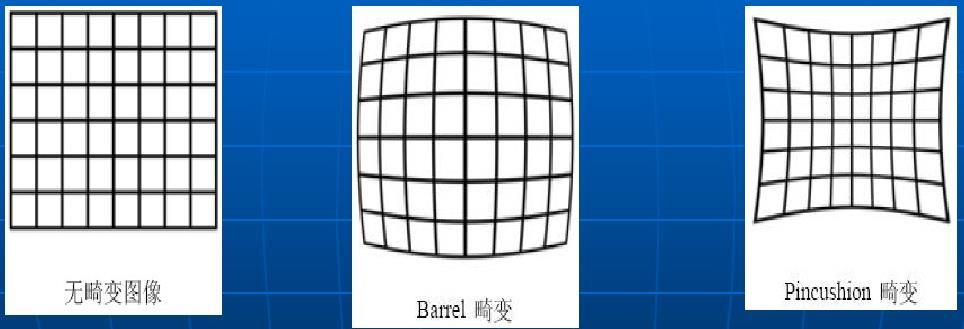


图 镜头的畸变

镜头的畸变：几乎所有的镜头都存在畸变，只是大小不同。一般如果畸变小于2%时人的肉眼是分辨不出的，如果畸变小于感光芯片的一个像素时摄像机也分辨不出。

##### 1.1.2.3 摄像机

摄像机是将通过镜头聚焦于像平面（感光芯片处）的光线生成图像的设备。关于摄像机需要了解的基本概念：像素、分辨率、帧率等。

像素：是指由图像的小方格组成的，这些小方块都有一个明确的位置和被分配的色彩数值，小方格颜色和位置就决定该图像所呈现出来的样子。

分辨率：摄像机的分辨率指摄像机的感光芯片上水平方向和垂直方向的感光晶元数的乘积。

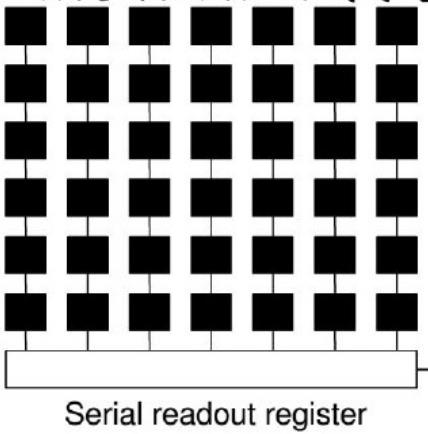


图 摄像机感光芯片示意图

通常表达是说摄像机分辨率为1600 X 1200，其乘积为1920000，摄像机的像素为200万。像素和分辨率指标直接影响摄像机成像的精细程度，因此在视觉系统中需要根据需要合理选择摄像机。

帧率：是指摄像机连续成像的速率，在视觉系统中捕捉移动物体图像时，帧率越高成像效果越好，帧率较低时移动物体会形成拖影。

#### 1.1.3 机器视觉工作原理

##### 1.1.3.1 目标物体识别原理

目标物识别的算法有很多，这里介绍一种常用的、典型的、也较为简单的算法原理——模板匹配法。

模板匹配的简要流程：

1）模板W开始位置在图像f的左上角

当W的中心位于f的边界上时，围绕f的边界需要进行填充。填充通常会被限制为模板宽度的一半。

2）模板从左上角向右下角开始移动（相当于遍历）

每次移动一个像素的位置，过程中可以得到模板和图像各位置间的相似度。

3）生成结果图像

相似度量值保存到结果图像中，在结果图像中的每个位置都包含了匹配度量值。

4）找到匹配位置

根据所用的匹配算法不同，对结果图像进行处理分析，找到匹配的位置。

不同的模板匹配算法：

1）基于灰度值的模板匹配算法

SAD:计算模板与图像之间差值的绝对值总和：



SSD:计算模板和图像之间差值的平方和：



在光照维持不变的情况下，SAD和SSD相似度量的结果很好。当光照发生变化时，效果不理想。

2）基于边缘的模板匹配算法

该算法图像中的边缘不会（或者很少）受光想变化的影响，并且当存在混乱和遮挡时，图像的边缘特征仍能保持大量的有用信息。

3）基于形状的模板匹配算法

该算法的相似度量考虑的是模板内像素的梯度向量，并通过计算梯度向量的內积总和最小值确定最佳匹配位置，稳定性和可靠性都比较优越。

实际应用中通常会使用成熟的算法库中的算法来实现模板匹配，比如OpenCV、Halcon等。

##### 1.1.3.2 视觉标定原理

1）相机固定

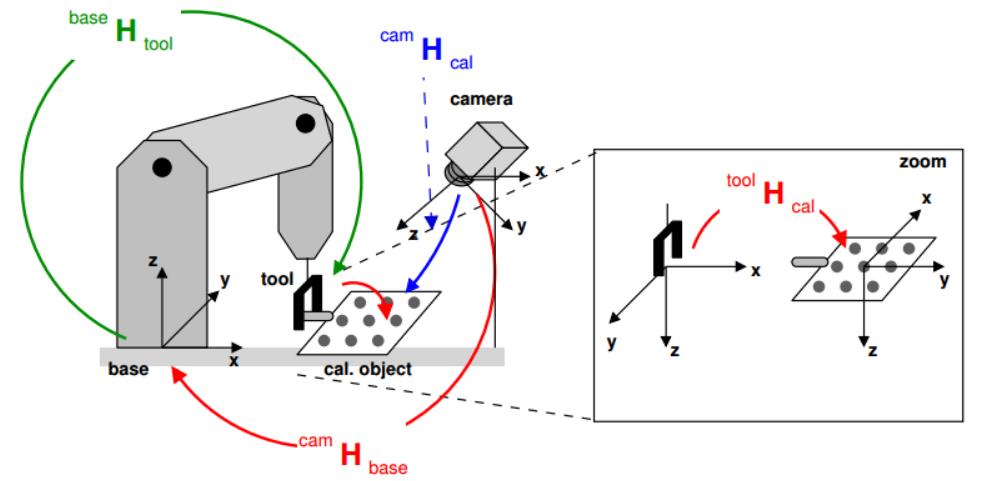


图 相机固定安装的标定

标定的最终结果，即得到base与cam的空间位置关系，即一个4\*4的变换矩阵。



2）相机固定在机械臂上

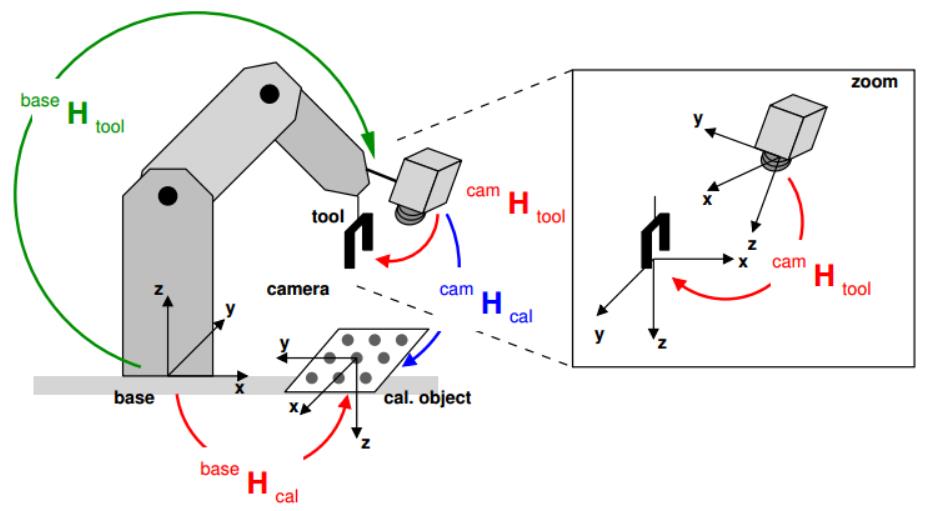


图 相机固定在机械臂上的标定



#### 1.1.4 机器视觉的应用现状

在国外，机器视觉的应用普及主要体现在半导体及电子行业，其中大概40%-50%都集中在半导体行业。具体如PCB印刷电路：各类生产印刷电路板组装技术、设备；单、双面、多层线路板，覆铜板及所需的材料及辅料；辅助设施以及耗材、油墨、药水药剂、配件；电子封装技术与设备；丝网印刷设备及丝网周边材料等。

电子生产加工设备：电子元件制造设备、半导体及集成电路制造设备、元器件成型设备、电子工模具。机器视觉系统还在质量检测的各个方面已经得到了广泛的应用，并且其产品在应用中占据着举足轻重的地位。除此之外，机器视觉还用于其他各个领域。

而在中国，视觉技术的应用开始于90年代，因为行业本身就属于新兴的领域，再加之机器视觉产品技术的普及不够，导致以上各行业的应用几乎空白。目前国内机器视觉大多为国外品牌。国内大多机器视觉公司基本上是靠代理国外各种机器视觉品牌起家，随着机器视觉的不断应用，公司规模慢慢做大，技术上已经逐渐成熟。

随着经济水平的提高，3D机器视觉也开始进入人们的视野。3D机器视觉大多用于水果和蔬菜、木材、化妆品、烘焙食品、电子组件和医药产品的评级。它可以提高合格产品的生产能力，在生产过程的早期就报废劣质产品，从而减少了浪费节约成本。这种功能非常适合用于高度、形状、数量甚至色彩等产品属性的成像。

在行业应用方面，主要有制药、包装、电子、汽车制造、半导体、纺织、烟草、交通、物流等行业，用机器视觉技术取代人工，可以提供生产效率和产品质量。例如在物流行业，可以使用机器视觉技术进行快递的分拣分类，不会出现大多快递公司人工进行分拣，减少物品的损坏率，可以提高分拣效率，减少人工劳动。

### 1.2 机器人图像处理与分析

#### 1.2.1 图像提取

##### 1.2.1.1 SURF特征提取

SURF算法为每个检测到的特征定义了位置和尺度,尺度值可用于定义围绕特征点的窗口大小。不论物体的尺度在窗口是什么样的,都将包含相同的视觉信息,这些信息用于表示特征点以使得它们与众不同。

在特征匹配中,特征描述子通常用于N维向量,在光照不变以及少许透视变形的情况下很理想。另外,优质的描述子可以通过简单的距离测量进行比较,比如欧氏距离。因此,它们在特征匹配算法中,用处是很大的。

在 OpenCV中,使用SURF进行特征点描述主要是 draw Matches方法和BruteForceMatcher类的运用,下面让我们一起来认识它们。

1.绘制匹配点: drawMatches()函数

drawMatches用于绘制出相匹配的两个图像的关键点,它有以下两个版本的

C++函数原型。

C++: void drawMatches(const Mat& imgl,

constvector<KeyPoint>& keypoints1,

const Mat& img2,

constvector<Keypoint>& keypoints2,

constvector<DMatch>& matcheslto2,

Mat& outImg,

const Scalar&matchColor=Scalar:: all(-1),

const Scalar&singlePoint Color=Scalar:: all(-1),

const vector<char>&matchesMask=vector<char>(),

intflags=DrawMatchesFlags:: DEFAULT)

C++: void drawMatches(const Mat& imgl,

constvector<Keypoint>& keypoints1,

const Mat& img2,

constvector<KeyPoint>& keypoints2,

const vector<vector<DMatch>>&matches 1to2,

Mat& outImg,

const Scalar&matchColor=Scalar:: all(-1),

const Scalar&singlePointColor=Scalar:: all(-1),

constvector<vector<char>>& matchesMask=vector<vector<char>>(),

Intflags=DrawMatchesFlags:: DEFAULT)

除了第五个参数 matcheslto2和第九个参数 matchesMask有细微的差别以外,

两个版本的基本上相同。下面统一讲解。

* 第一个参数, constMat&类型的imgl,第一幅源图像。
* 第二个参数, const vector< KeyPoint>&类型的 keypoints1,根据第一幅源图

像得到的特征点,它是一个输出参数。

* 第三个参数, const mat&类型的img2,第二幅源图像。
* 第四个参数, const vector≤ Keypoint>&类型的 keypoints2,根据第二幅源图

像得到的特征点,它是一个输出参数。

* 第五个参数, matches lto2,第一幅图像到第二幅图像的匹配点,即表示每个图1中的特征点都在图2中有一一对应的点。
* 第六个参数,Mat&类型的 outImg,输出图像,其内容取决于第五个参数标识符 falgs。
* 第七个参数, const Scalar&类型的 matchColor,匹配的输出颜色,即线和关

键点的颜色。它有默认值 Scalar:all(-1),表示颜色是随机生成的。

* 第八个参数, const Scalar&类型的 singlePointColor,单一特征点的颜色,它也表示随机生成颜色的默认值 Scalar:all(-1)。
* 第九个参数, matchesMask,确定哪些匹配是会绘制出来的掩膜,如果掩膜为空,表示所有匹配都进行绘制。
* 第十个参数,int类型的flags,特征绘制的标识符,有默认值DrawMatchesFlags ::DEFAULT。可以在如下这个 DrawMatchesFlags结构体中选取值:

struct DrawMatchesFlags

{

enum

{

DEFAULT=0,//创建输出图像矩阵(使用Mat:: create)。使用现存的输出图像

绘制匹配对和特征点。且对每一个关键点,只绘制中间点

DRAW\_OVER\_OUTIMG=1,//不创建输出图像矩阵,而是在输出图像上绘制匹配对

NOT\_DRAW\_SINGLE\_POINTS=2,//单点特征点不被绘制

DRAW\_RICH\_KEYPOINTS=4//对每一个关键点,绘制带大小和方向的关键点圆圈

};

};

2.BruteForceMatcher类源码分析

接下来,我们看看本节中会用到的 BruteForceMatcher类的源码分析, BruteForceMatcher类用于进行暴力匹配相关的操作。在......\opencv\sources\ modules\ legacy\ include\opencv2\legacy\ legacy. hpp路径下,可以找到BruteForceMatcher类的定义。

template<class Distance>

class CV\_EXPORTS BruteForceMatcher: publicBEMatcher

{

public:

BruteForceMatcher( Distance d= Distance()):

BFMatcher(Distance:: normType, false){(void)d;}

virtual ~BruteForceMatcher(){}

};

其公共继承自 BFMatcher类。而 BFMatcher类位于… opencv\sources\ modules\features2d\ linclude\lopencv2\features2d\features2d.hpp

路径之下,我们一起看看其代码

class CV\_EXPORTS\_W BFMatcher :publicDescriptorMatcher

{

//......

};

发现公共其继承自 DescriptorMatcher类,再次进行溯源,找到 DescriptorMatcher类的定义。

class CV\_EXPORTS\_W DescriptorMatcher: public Algorithm

{

//......

};

可以发现, DescriptorMatcher类是继承自它们“德高望重的祖先” Algorithm基类的。

而用 BruteForceMatcher类时用到最多的 match方法,是它从 DescriptorMatcher

类那里继承而来。定义如下。

//为各种描述符找到一个最佳的匹配(若掩膜为空)

CV\_WRAP void match( const Mat& queryDescriptors, const

Mat&trainDescriptors

CV\_OUTvector<DMatch>& matches, const Mat& mask=Mat ())const

##### 1.2.1.2 ORB特征提取

1.ORB算法概述

ORB为是 ORiented Brief的简称,是 brief算法的改进版。ORB于2011年在《ORB

an efficient alternative to Sift or SurF》这篇文章中被提出。此文章的摘要中说,ORB算法比sift算法效率高两个数量级,而在计算速度上,ORB是sift的100倍,是surf的10倍。而江湖上流传的说法是,ORB算法综合性能在各种测评里相较于其他特征提取算法是最好的

若想要引出ORB( ORiented Brief)算法,要由 Brief描述子入手,下面,就来先介绍 Brief描述子。

2.相关概念认知

a.关于 Brief描述子

Brief是 Binary Robust Independent Elementary Features的缩写。这个特征描述子是由EPFL的 Calonder在ECCV2010上提出的,主要思路就是在特征点附近随机选取若干点对,将这些点对的灰度值的大小,组合成一个二进制串,并将这个二进制串作为该特征点的特征描述子。

BRIEF的优点在于速度,而缺点也相当明显。

* 不具备旋转不变性。
* 对噪声敏感
* 不具备尺度不变性。

而ORB算法就是试图解决上述缺点中的1和2提出的一种新概念。

b.关于尺度不变性

ORB没有试图解决尺度不变性(因为FAST本身就不具有尺度不变性),但是这样只求速度的特征描述子,一般都是应用在实时的视频处理中的,这样的话就可以通过跟踪还有一些启发式的策略来解决尺度不变性的问题。

c.关于计算速度

经过统计,ORB算法的执行速度是SIFT的100倍,是SURF的10倍。

3.ORB类相关源码简单分析

ORB, Orb FeatureDetector, OrbDescriptorExtractor这三个类,也是完全等价的。然后我们继续溯源，发现ORB类同样继承自Feature2D类：

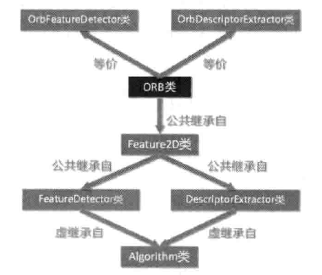
class CV\_EXPORTS\_W ORB public Feature2D

{

//......

};

接下来的分析思路,经过分析,可以得到如下图所示的类关系脉络图。



ORB类相关联系脉络图

#### 1.2.2 图像预处理

##### 1.2.2.1 线性滤波

1.平滑处理

平滑处理( smoothing)也称模糊处理( bluring),是一种简单且使用频率很高的图像处理方法。平滑处理的用途有很多,最常见的是用来减少图像上的噪点或者失真。在涉及到降低图像分辨率时,平滑处理是非常好用的方法。

2.图像滤波与滤波器

图像滤波,指在尽量保留图像细节特征的条件下对目标图像的噪声进行抑制,是图像预处理中不可缺少的操作,其处理效果的好坏将直接影响到后续图像处理和分析的有效性和可靠性。

消除图像中的噪声成分为图像的平滑化或滤波操作。信号或图像的能量大部分集中在幅度谱的低频和中频段,而在较高频段,有用的信息经常被噪声淹没。因此一个能降低高频成分幅度的滤波器就能够减弱噪声的影响。

图像滤波的目的有两个:一个是抽出对象的特征作为图像识别的特征模式；另一个是为适应图像处理的要求,消除图像数字化时所混入的噪声。而对滤波处理的要求也有两条:一是不能损坏图像的轮廓及边缘等重要信息；二是使图像清晰视觉效果好。

平滑滤波是低频增强的空间域滤波技术。它的目的有两类:一类是模糊;另一类是消除噪音。

空间域的平滑滤波一般采用简单平均法进行,就是求邻近像元点的平均亮度值。邻域的大小与平滑的效果直接相关,邻域越大平滑的效果越好,但邻域过大,平滑也会使边缘信息损失的越大,从而使输出的图像变得模糊,因此需合理选择邻域的大小。

关于滤波器,一种形象的比喻是:可以把滤波器想象成一个包含加权系数的窗口,当使用这个滤波器平滑处理图像时,就把这个窗口放到图像之上,透过这个窗口来看我们得到的图像。

滤波器的种类有很多,在新版本的 OpenCV中,提供了如下5种常用的图像平滑处理操作方法,它们分别被封装在单独的函数中,使用起来非常方便。

* 方框滤波 -Box Blur函数
* 均值滤波(邻域平均滤波)—Blur函数
* 高斯滤波— -Gaussian blur函数
* 中值滤波— median Blur函数
* 双边滤波— bilateral Filter函数

3.线性滤波器的简介

线性滤波器:线性滤波器经常用于剔除输入信号中不想要的频率或者从许多频率中选择一个想要的频率。

几种常见的线性滤波器如下。

* 低通滤波器:允许低频率通过;
* 高通滤波器:允许高频率通过;
* 带通滤波器:允许一定范围频率通过;
* 带阻滤波器:阻止一定范围频率通过并且允许其他频率通过;
* 全通滤波器:允许所有频率通过,仅仅改变相位关系;
* ·陷波滤波器(Band- Stop Filter):阻止一个狭窄频率范围通过,是一种特殊带阻滤波器。

4.滤波和模糊

关于滤波和模糊,大家往往在初次接触的时候会弄混淆:“一会儿说滤波,会儿又说模糊,似乎不太清楚。”

没关系,在这里,我们就来分析一下,为大家扫清障碍。

上文已经提到过,滤波是将信号中特定波段频率滤除的操作,是抑制和防止干扰的一项重要措施。

为了方便说明,就拿我们经常用的高斯滤波来作例子吧。滤波可分低通滤波和高通滤波两种:高斯滤波是指用高斯函数作为滤波函数的滤波操作,至于是不是模糊,要看是高斯低通还是高斯高通,低通就是模糊,高通就是锐化。

其实说白了是很简单的：

* 高斯滤波是指用高斯函数作为滤波函数的滤波操作;
* 高斯模糊就是高斯低通滤波。

5.邻域算子与线性邻域滤波

邻域算子(局部算子)是利用给定像素周围的像素值的决定此像素的最终输出值的一种算子。而线性邻域滤波就是一种常用的邻域算子,像素的输出值取决于输入像素的加权和,具体过程如下图所示。

邻域算子除了用于局部色调调整以外,还可以用于图像滤波,以实现图像的平滑和锐化,图像边缘增强或者图像噪声的去除。本节我们介绍的主角是线性邻域滤波算子,即用不同的权重去结合一个小邻域内的像素,来得到应有的处理效果。

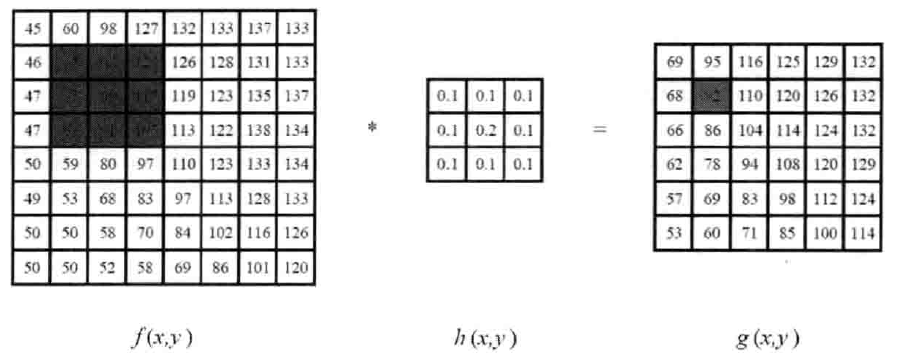


图 邻域滤波(卷积)

图注:邻域滤波(卷积)——左边图像与中间图像的卷积产生右边图像。目标图像中蓝色标记的像素是利用原图像中红色标记的像素计算得到的。

线性滤波处理的输岀像素值g(i,j）是输入像素值的加权和,如下



其中的h(k,I),我们称其为“核”,是滤波器的加权系数,即滤波器的“滤波系数”。

上面的式子可以简单写作:



其中f表示输入像素值，h表示加权系数“核”，g表示输出像素值。

在新版本的 OpenCV中,提供了如下三种常用的线性滤波操作,它们分别被封装在单独的函数中,使用起来非常方便。

* ·方框滤波— boxblur函数
* ·均值滤波—blur函数
* 高斯滤波— Gaussian Blur函数

下面我们来对它们进行一一介绍。

6.方框滤波（box Filter）

方框滤波( box Filter)被封装在一个名为 boxblur的函数中,即 boxblur函数的作用是使用方框滤波器( box filter)来模糊一张图片,从src输入,从dst输出。

函数原型如下。

C++: void boxFilter (InputArray src, OutputArray dst, int depth, Size ksize, Point anchor=Point(-1,-1), boolnormalize=true, int borderType=BORDER DEFAULT)

参数详解如下。

* ·第一个参数, InputArray类型的src,输入图像,即源图像,填Mat类的对象即可。该函数对通道是独立处理的,且可以处理任意通道数的图片。但需要注意,待处理的图片深度应该为CV\_8U、CV\_16U、CV\_16S、CV\_32F以及CV\_64F之一。
* 第二个参数, OutputArray类型的dst,即目标图像,需要和源图片有一样的尺寸和类型。
* 第三个参数,int类型的 depth,输出图像的深度,-1代表使用原图深度即 src.depth()
* ·第四个参数,Size类型(对Size类型稍后有讲解)的ksize,内核的大小。一般用Size(w,h)来表示内核的大小,其中w为像素宽度,h为像素高度。Size(3,3)就表示3x3的核大小,Size(5,5)就表示5x5的核大小。
* 第五个参数, Point类型的 anchor,表示锚点(即被平滑的那个点)。注意它有默认值 Pointe-1,1)。如果这个点坐标是负值的话,就表示取核的中心为锚点,所以默认值 Pointe(-1,-1)表示这个锚点在核的中心。
* 第六个参数,bool类型的 normalize,默认值为true,一个标识符,表示内核是否被其区域归一化( normalized)了。
* 第七个参数,int类型的 border Type,用于推断图像外部像素的某种边界模式。有默认值 BORDER\_DEFAULT,我们一般不去管它。

Box Filter函数方框滤波所用的核表示如下。



其中：



上式中f表示原图,h表示核,g表示目标图,当 normalize=true的时候,方框滤波就变成了我们熟悉的均值滤波。也就是说,均值滤波是方框滤波归一化( normalized)后的特殊情况。其中,归一化就是把要处理的量都缩放到一个范围内,比如(0,1),以便统一处理和直观量化。而非归一化( Unnormalized)的方框滤波用于计算每个像素邻域内的积分特性,比如密集光流算法( dense optical flow algorithms)中用到的图像倒数的协方差矩阵( covariance matrices of image derivatives)。

如果我们要在可变的窗口中计算像素总和,可以使用 integral函数。

7.均值滤波

均值滤波,是最简单的一种滤波操作,输出图像的每一个像素是核窗口内输入图像对应像素的平均值(所有像素加权系数相等),其实说白了它就是归一化后的方框滤波。我们在下文进行源码剖析时会发现,blur函数内部中其实就是调用了一下 box Filter。

下面开始讲均值滤波的内容吧。

1. 均值滤波的理论简析

均值滤波是典型的线性滤波算法,主要方法为邻域平均法,即用一片图像区域的各个像素的均值来代替原图像中的各个像素值。一般需要在图像上对目标像素给岀一个模板(内核),该模板包括了其周围的临近像素(比如以目标像素为中心的周围8(3x3-1)个像素,构成一个滤波模板,即去掉目标像素本身)。再用模板中的全体像素的平均值来代替原来像素值。即对待处理的当前像素点(x,y),选择一个模板,该模板由其近邻的若干像素组成,求模板中所有像素的均值,再把该均值赋予当前像素点(x,y),作为处理后图像在该点上的灰度点g(x,y),即g(x,y)=1m∑f(x,y),其中m为该模板中包含当前像素在内的像素总个数。

1. 均值滤波的缺陷

均值滤波本身存在着圊有的缺陷,即它不能很好地保护图像细节,在图像去噪的同时也破坏了图像的细节部分,从而使图像变得模糊,不能很好地去除噪声。

1. 在 OpenCV中使用均值滤波——bLur函数

blur函数的作用是:对输入的图像sre进行均值滤波后用dst输出。blur函数在 OpenCV官方文档中,给出的其核是这样的:



这个内核一看就明了,就是在求均值,即blur函数封装的就是均值滤波。bur函数的原型如下。

C++: void blur (InputArray src, OutputArraydst, size ksize, Pointanchor-Point(-1,-1), int borderType=BORDER\_DEFAULT)

* ·第一个参数, InputArray类型的src,输入图像,即源图像,填Mat类的对象即可。该函数对通道是独立处理的,且可以处理任意通道数的图片。但需要注意的是,待处理的图片深度应该为CV\_8U、CV\_16U、CV\_16S、CV\_32F以及CV\_64F之一。
* ·第二个参数, OutputArray类型的dst,即目标图像,需要和源图片有一样的尺寸和类型。比如可以用Mat: :Clone,以源图片为模板,来初始化得到如假包换的目标图。
* ·第三个参数,Size类型(对Size类型稍后有讲解)的ksize,内核的大小。一般写作Sie(w,h)来表示内核的大小(其中,w为像素宽度,h为像素高度)。Sie(3,3)就表示3×3的核大小,Size(5,5)就表示5×5的核大小。
* 第四个参数, Point类型的 anchor,表示锚点(即被平滑的那个点),注意它有默认值 Point(-1,-1)。如果这个点坐标是负值,就表示取核的中心为锚点,所以默认值 Point(-1,-1)表示这个锚点在核的中心。
* 第五个参数,int类型的 border Type,用于推断图像外部像素的某种边界模式。有默认值 BORDER\_DEFAULT,我们一般不去管它。

8.高斯滤波

1. 高斯滤波的理论简析

高斯滤波是一种线性平滑滤波,可以消除高斯噪声,广泛应用于图像处理的减噪过程。通俗地讲,高斯滤波就是对整幅图像进行加权平均的过程,每一个像素点的值,都由其本身和邻域内的其他像素值经过加权平均后得到。高斯滤波的具体操作是:用一个模板(或称卷积、掩模)扫描图像中的每一个像素,用模板确定的邻域内像素的加权平均灰度值去替代模板中心像素点的值。

大家常说高斯滤波是最有用的滤波操作,虽然它用起来效率往往不是最高的。高斯模糊技术生成的图像,其视觉效果就像是经过一个半透明屏幕在观察图像这与镜头焦外成像效果散景以及普通照明阴影中的效果都明显不同。高斯平滑也用于计算机视觉算法中的预先处理阶段,以增强图像在不同比例大小下的图像效果(参见尺度空间表示以及尺度空间实现)。从数学的角度来看,图像的高斯模糊过程就是图像与正态分布做卷积。由于正态分布又叫作高斯分布,所以这项技术就叫作高斯模糊。

图像与圆形方框模糊做卷积将会生成更加精确的焦外成像效果。由于高斯函数的傅里叶变换是另外一个高斯函数,所以高斯模糊对于图像来说就是一个低通滤波操作。

高斯滤波器是一类根据高斯函数的形状来选择权值的线性平滑滤波器。高斯平滑滤波器对于抑制服从正态分布的噪声非常有效。一维零均值高斯函数如下。



其中,高斯分布参数 Sigma决定了高斯函数的宽度。对于图像处理来说,常用二维零均值离散高斯函数作平滑滤波器。

二维高斯函数如下。



1. 高斯滤波: GaussianB|ur函数

GaussianBlur函数的作用是用高斯滤波器来模糊一张图片,对输入的图像src进行高斯滤波后用dst输出。它将源图像和指定的高斯核函数做卷积运算,并且支持就地过滤( In-placefiltering)。

C++: void GaussianBlur(InputArray src, OutputArray dst, size ksize,

double sigmax, double sigmaY=0, intborderType=BORDER DEFAULT)

* ·第一个参数, InputArray类型的src,输入图像,即源图像,填Mat类的对象即可。它可以是单独的任意通道数的图片,但需要注意的是,其图片深度应该为CV\_8U、CV\_16U、CV\_16S、CV\_32F以及CV\_64F之一。
* 第二个参数, OutputArray类型的dst,即目标图像,需要和源图片有一样的尺寸和类型。比如可以用Mat::Clone,以源图片为模板,来初始化得到如假包换的目标图。
* ·第三个参数,Size类型的ksize高斯内核的大小。其中 size. width和size.height可以不同,但它们都必须为正数和奇数,或者是零,这都由sigma计算而来。
* ·第四个参数, double类型的 sigmaX,表示高斯核函数在Ⅹ方向的的标准偏差。
* ·第五个参数, double类型的 sigmaY,表示高斯核函数在Y方向的的标准偏差。若 sigmaY为零,就将它设为 sigmaX;如果 sigmaX和 sigmaY都是0,那么就由ksize. width和kize. height计算出来。

为了结果的正确性着想,最好是把第三个参数Size、第四个参数 sigmaX和第五个参数 sigmaY全部指定到。

* ·第六个参数,int类型的 border Type,用于推断图像外部像素的某种边界模式。有默认值 BORDER\_DEFAULT,我们一般不去管它。

9.线性滤波相关 OpenCV源码剖析

1. OpenCV中 boxFilter函数源码解析

void cv:: boxFilter( InputArray\_src, OutputArray dst, int ddepth,Size ksize, Point anchor

bool normalize, int borderType)

{

Mat src=\_src. getMat();//复制源图的形参Mat数据到临时变量,用于稍后的操作

int sdepth=src.depth(),cn=src. channels();//定义int型临时变量,代表源图深度的sdepth,源图通道的引用cn

//处理ddepth小于零的情况

if( ddepth <0)

ddepth=sdepth;

\_dst.create(src.size(), CV\_MAKETYPE(ddepth,cn));//初始化目标图

Mat dst=\_dst. getMat();//复制目标图的形参Mat数据到临时变量,用于稍后的操作

//处理 border Type不为 BORDER\_CONSTANT且 normalize为真的情况

if( borderType !=BORDER CONSTANT&& normalize)

{

if( src.rows == 1)

ksize. height =1;

if( src.cols ==1)

ksize. width =1;

}

//若之前有过 HAVE TEGRA OPTIMIZATION优化选项的定义,则执行宏体中的 tegra优化

版函数并返回

#ifdef HAVE\_TEGRA\_OPTIMIZATION

if( tegra:: box(src, dst, ksize, anchor, normalize, borderT'ype))

return;

#endif

//调用 FilterEngine滤波引擎,正式开始滤波操作

Ptr<FilterEngine> f=createBoxFilter(src.type(),dst.type(),ksize,anchor,normalize, borderType);

f->apply( src, dst );

}

其中,Ptr是用来动态分配的对象的智能指针模板类。可以发现,函数的内部代码思路是很清晰的:先复制源图的形参Mat数据到临时变量,定义一些临时变量,再处理ddepth小于零的情况,接着处理 border Type不为 BORDER\_CONSTANT且normalize为真的情况,最终调用 FilterEngine滤波引擎创建一个 BoxFilter,正式开始滤波操作。

这里的 Filter Engine是 OpenCV图像滤波功能的核心引擎,我们有必要详细剖析看其源代码。

1. FilterEngine类解析: OpenCV图像滤波核心引擎

Filter Engine类是 OpenCV关于图像滤波的主力军类,是 OpenCV图像滤波功能的核心引擎。各种滤波函数如blur、 Gaussian blur,其实是就是在函数末尾处定义了一个Ptr< Filter Engine>类型的f,然后f-> apply(src,dst)了一下而已。

这个类可以把几乎所有的滤波操作施加到图像上,它包含了所有必要的中间缓存器。有很多和滤波相关的 create系函数的返回值直接就是Pr< FilterEngine>。

比如：

* cv::create SeparableLinearFilter()
* cv::createLinear Filter(), cv:: createGaussianFilter(), cv: : createDerivFilter()
* cv::createBoxFilter()
* cv::createMorphology Filter()

下面给出其中一个函数的原型。

Ptr<FilterEngine>createLinearFilter (int srcType, int dstType,

InputArray kernel, Point\_anchor=Point (-1, -1), double delta=0, int

rowBorderType= BORDER\_DEFAULT, intcolumnBorderType=-1, const Scalar&bordervalue=Scalar())

上面提到过,其中的Ptr是用来动态分配的对象的智能指针模板类,而尖括号里的模板参数就是 Filter Engine。

使用 Filter Engine类可以分块处理大量的图像,构建复杂的管线,其中就包含些进行滤波阶段。如果我们需要使用预先定义好的的滤波操作,有cv::filter2D()、cv::erode()和cv::dilate()可以选择,它们不依赖于 FilterEngine,在自己函数体内部就实现Filter Engine提供的功能;不像其他诸如我们今天讲的blur系列函数,依赖于FilterEngine引擎。

我们看下其类声明经过详细注释的源码,如下。

class CV\_EXPORTS FilterEngine

{

public:

//默认构造函数

FilterEngine();

//完整的构造函数。\_fi1ter2D、\_rowFilter和 \_columnFilter之一,必须为非空

FilterEngine(const Ptr<BaseFilter>& \_filter2D,

constPtr<BaseRowFilter>& \_rowFilter,

constPtr<BaseColumnFilter>&\_columnFilter,

int srcType, int dstType, intbufType,

int rowBorderType=BORDER\_REPLICATE,

Int\_columnBorderType=-1,

const Scalar& borderValue=Scalar());

//默认析构函数

virtual~FilterEngine();

//重新初始化引擎。释放之前滤波器申请的内存

void init(const Ptr<BaseFilter>&\_filter2D,

constPtr<BaseRowFilter>&\_rowFilter,

constPtr<BasecolumnFilter>&\_columnFilter,

int srcType, int dstType, intbufType,

Int\_rowBorderType= BORDER\_REPLICATE,

Int\_columnBorderType=-1,

const Scalar&\_borderValue=Scalar());

//开始对指定了ROI区域和尺寸的图片进行滤波操作

virtual int start(Size wholeSize, Rect roi, int maxBufRows=-1);

//开始对指定了ROI区域的图片进行滤波操作

virtual int start(const Mat& src, const Rect&srcRoi=Rect(0,0,-1,-1),

bool isolated=false, intmaxBufRows=-1);

//处理图像的下一个 srcCount行(函数的第三个参数)

virtual int proceed (const uchar\* src, int srcStep, int srcCount,uchar\* dst, intdststep);

//对图像指定的ROI区域进行滤波操作,若 SrcRoi=(0,0,-1,-1),则对整个图像进行滤波操作

virtual void apply( const Mat& src, Mat& dst,

const Rect&srcRoi=Rect(0,0,-1,-1),

Point dstofs=Point(0,0),

bool isolated=false);

//如果滤波器可分离,则返回true

boolisSeparable() const{ return (const BaseFilter\*)filter2D ==0;}

//返回输入和输出行数

int remainingInputRows() const;

intremainingOutputRows() const;

//一些成员参数定义

int srcType, dstType, bufType;

Size ksize;

Point anchor;

int maxWidth;

Size wholesize;

Rect roi;

int dxl, dx2;

int rowBorderType, columnBorderType;

vector<int> borderTab;

int borderElemSize;

vector<uchar> ringbuf;

vector<uchar> srcRow;

vector<uchar> constBorderValue;

vector<uchar> constBorderRow;

int bufstep, startY, startYo, endY, rowCount, dsty;

vector<uchar\*> rows;

Ptr<BaseFilter> filter2D;

Ptr<BaseRowFilter> rowFilter;

Ptr<BasecolumnFilter> columnFilter;

};

1. OpenCV中blur函数源码剖析

我们可以在 OpenCV的安装路径 \sources\modules\imgprocl\src下的smooth.cpp源文件中找到blur的源代码,下面一起来看 OpenCV中blur函数定义的真面目。

void cv::blur(InputArray src, OutputArray dst, Size ksize, Point anchor, int border Type)

//调用 boxFilter函数进行处理

boxFilter( src, dst, -l, ksize, anchor, true, borderType );

可以发现,在blur函数内部调用了一个 boxFilter函数,且第6个参数为true,也就是我们上文所说的 normalize=true,即均值滤波是均一化后的方框滤波。

10.OpenCV中 GaussianBlur函数源码剖析

下面我们看一下 OpenCV中 Gaussian Blur函数源代码:

void cv::GaussianBlur( InputArray\_src, OutputArray\_dst, Size ksize, double sigmal, doublesigma2,

int borderType)

{

//复制形参Mat数据到临时变量,用于稍后的操作

Mat src =\_src. getMat();

\_dst. create( src.size(), src.type());

Mat dst= \_dst. getMat ();

//处理边界选项不为 BORDER\_CONSTANT时的情况

if( borderType != BORDER\_CONSTANT)

{

if( src.rows ==1)

ksize. height = 1;

if( src.cols == 1)

Ksize.width= 1;

}

//若ksize长宽都为1,将源图复制给目标图

if( ksize.width ==1&& ksize.height ==1

{

src.copyTo(dst);

return;

}

//若之前有过HAVE\_ TEGRA\_ OPTIMIZATION优化选项的定义,则执行宏体中的 tegra优化

版函数并返回

#ifdef HAVE\_TEGRA\_OPTIMIZATION

if(sigmal==0&& sigma2 ==0 & &tegra:: gaussian(src, dst, ksize, borderType))

return;

#endif

//如果 HAVE\_PP&&( IPP\_VERSION\_MAJOR>=7为真,则执行宏体中语句

#if defined HAVE\_IPP &&(IPP\_VERSION\_MAJOR >= 7)

if(src.type()== CV\_32FCl && sigmal==sigma2 &&ksize.width==ksize.height&&sigmal ! =0.0)

{

IppiSize roi ={src.cols, src rows)};

int bufSize =0;

IppiFilterGaussGetBufferSize\_32f \_C1R(roi, ksize. width,&bufsize);

AutoBuffer<uchar> buf(bufSize+128);

if( ippiFilterGaussBorder 32f CIR((const Ipp32f

\*)src.data,(int)src. step,

(Ipp32f \*)dst.data, (int)dst. step,

roi,ksize.width, (Ipp32f)sigmal,

(IppiBorderType)borderType,0.0,

alignPtr(&buf[o], 32))>=0)

return;

}

#endif

//调用滤波引擎,正式进行高斯滤波操作

Ptr<FilterEngine>f= createGaussianFilter( src.type(), ksize, sigmal,

sigma2, borderType );

f->apply( src, dst );

}

11.线性滤波核心API函数

a. 方框滤波: boxFilter函数

box Filter的函数作用是使用方框滤波( box filter)来模糊一张图片,由src输入,dst输出。

函数原型如下。

C++: void boxFilter(InputArray src, OutputArray dst, int ddepth, Size ksize, Point anchor=Point(-1,-1), boolnormalize=true, int borderType=BORDER DEFAULT)

参数详解如下。

* 第一个参数: InputArray类型的src,输入图像,即源图像,填Mat类的对象即可。该函数对通道是独立处理的,且可以处理任意通道数的图片。但需要注意,待处理的图片深度应该是CV\_8U、CV\_16U、CV\_16S、CV\_32F、CV\_64F之一。
* 第二个参数: OutputArray类型的dst,即目标图像,需要和源图片有一样的尺寸和类型。
* 第三个参数:int类型的ddepth,输出图像的深度。“-1”代表使用原图深度,即src.depth()
* 第四个参数:Size类型的ksize,内核的大小。一般用Size(w,h)的写法来表示内核的大小(其中,w为像素宽度,h为像素高度)。例如,Size(3,3)表示3x3的核大小;Size(5,5)就表示5x5的核大小。
* 第五个参数: Point类型的 anchor,表示锚点(即被平滑的那个点),注意它有默认值 Pointe(-1,-1)。如果这个点坐标是负值的话,就表示取核的中心为锚点,所以默认值 Point(-1,-1)表示这个锚点在核的中心。
* 第六个参数:bool类型的 normalize,默认值为true,一个标识符,表示内核是否被其区域归一化( normalized)了。
* 第七个参数:int类型的borderType,用于推断图像外部像素的某种边界模

式。有默认值 BORDER\_DEFAULT,我们一般不去管它。

b. 均值滤波:blur函数

blur的作用是对输入的图像src进行均值滤波后用dst输出。

函数原型如下。

C++: void blur (InputArray src, OutputArraydst, Size ksize, Point

anchor=Point(-1,-1), int borderType=BORDER\_DEFAULT)

参数详解如下。

* 第一个参数: InputArray类型的src,输入图像,即源图像,填Mat类的对象即可。该函数对通道是独立处理的,且可以处理任意通道数的图片。但需要注意,待处理的图片深度应该为CV\_8U、CV\_16U、CV\_16S、CV\_32F、CV\_64F之一。
* 第二个参数: OutputArray类型的dst,即目标图像,需要和源图片有一样的尺寸和类型。比如可以用Mat: :Clone,以源图片为模板,来初始化得到如假包换的目标图。
* 第三个参数:Size类型(对Size类型稍后有讲解)的ksize,内核的大小。一般用Size(w,h)的写法来表示内核的大小(其中w为像素宽度,h为像素高度)。例如,Size(3,3)表示3x3的核大小,Size(5,5)就表示5x5的核大小。
* 第四个参数: Point类型的 anchor,表示锚点(即被平滑的那个点),注意它有默认值 Pointe(-1,-1)。如果这个点坐标是负值的话,就表示取核的中心为锚点,所以默认值 Point(-1,1)表示这个锚点在核的中心。
* 第五个参数:int类型的 borderType,用于推断图像外部像素的某种边界模式。有默认值 BORDER\_DEFAULT,我们一般不去管它。

c. 高斯滤波: GaussianBlur函数

Gaussian Blur函数的作用是用高斯滤波器来模糊一张图片,对输入的图像src进行高斯滤波后用dst输出。

函数原型如下

C++: void GaussianBlur(InputArray src, OutputArray dst, Size ksize,

double sigmaX, double sigmaY=0, intborderType= BORDER\_DEFAULT)

参数详解如下。

* 第一个参数: Input Array类型的src,输入图像,即源图像,填Mat类的对象即可。它可以是单独的任意通道数的图片。但需要注意,图片深度应该为CV\_8U、CV\_16U、CV\_16S、CV\_32F、CV\_64F之一。
* 第二个参数: OutputArray类型的dst,即目标图像,需要和源图片有一样的尺寸和类型。比如可以用Mat: :Clone,以源图片为模板,来初始化得到如假包换的目标图。
* 第三个参数:Size类型的ksize高斯内核的大小。其中ksize. width和ksize.heigt可以不同,但它们都必须为正数和奇数,也可以为零。它们都是由 sigma计算而来的。
* 第四个参数: double类型的 sigmaX,表示高斯核函数在X方向的的标准偏差。
* 第五个参数: double类型的 sigmaY,表示高斯核函数在Y方向的的标准偏差。若 sigmaY为零,就将它设为 sigmaX,如果 sigmaX和 sigmaY都是0,那么就由 ksize.width和 ksize .height计算出来。为了结果的正确性着想,最好是把第三个参数Size、第四个参数 sigmaX和第五个参数 sigmaY全部指定到。
* 第六个参数:int类型的 borderType,用于推断图像外部像素的某种边界模式。注意它有默认值 BORDER\_DEFAULT。

##### 1.2.2.2 非线性滤波

正如我们在上节中讲到的,线性滤波可以实现很多种不同的图像变换。而非线性滤波,如中值滤波器和双边滤波器,有时可以达到更好的实现效果。

1.非线性滤波概述

在2.2.2.1节中,我们所考虑的滤波器都是线性的,即两个信号之和的响应和它们各自响应之和相等。换句话说,每个像素的输出值是一些输入像素的加权和线性滤波器易于构造,并且易于从频率响应角度来进行分析。

然而,在很多情况下,使用邻域像素的非线性滤波会得到更好的效果。比如在噪声是散粒噪声而不是高斯噪声,即图像偶尔会出现很大的值的时候,用高斯滤波器对图像进行模糊的话,噪声像素是不会被去除的,它们只是转换为更为柔和但仍然可见的散粒。这就到了中值滤波登场的时候了。

2.中值滤波概述

中值滤波( Median filter)是一种典型的非线性滤波技术,基本思想是用像素点邻域灰度值的中值来代替该像素点的灰度值,该方法在去除脉冲噪声、椒盐噪声的同时又能保留图像的边缘细节。

中值滤波是基于排序统计理论的一种能有效抑制噪声的非线性信号处理技术,其基本原理是把数字图像或数字序列中一点的值用该点的一个邻域中各点值的中值代替,让周围的像素值接近真实值,从而消除孤立的噪声点。这对于斑点噪声( speckle noise)和椒盐噪声(salt-and- pepper noise)来说尤其有用,因为它不依赖于邻域内那些与典型值差别很大的值。中值滤波器在处理连续图像窗函数时与线性滤波器的工作方式类似,但滤波过程却不再是加权运算。

中值滤波在一定的条件下可以克服常见线性滤波器,如最小均方滤波、方框滤波器、均值滤波等带来的图像细节模糊,而且对滤除脉冲干扰及图像扫描噪声非常有效,也常用于保护边缘信息。保存边缘的特性使它在不希望岀现边缘模糊的场合也很有用,是非常经典的平滑噪声处理方法。

* 中值滤波与均值滤波器比较

优势:在均值滤波器中,由于噪声成分被放入平均计算中,所以输出受到了噪声的影响。但是在中值滤波器中,由于噪声成分很难选上,所以几乎不会影响到输出。因此同样用3×3区域进行处理,中值滤波消除的噪声能力更胜一筹。中值滤波无论是在消除噪声还是保存边缘方面都是一个不错的方法。

劣势:中值滤波花费的时间是均值滤波的5倍以上。

3.双边滤波

双边滤波( Bilateral filter)是一种非线性的滤波方法,是结合图像的空间邻近度和像素值相似度的一种折中处理,同时考虑空域信息和灰度相似性,达到保边去噪的目的,具有简单、非迭代、局部的特点。

双边滤波器的好处是可以做边缘保存( edge preserving)。以往常用维纳滤波或者高斯滤波去降噪,但二者都会较明显地模糊边缘,对于高频细节的保护效果并不明显。双边滤波器顾名思义,比高斯滤波多了一个高斯方差 sigma-d,它是基于空间分布的高斯滤波函数,所以在边缘附近,离得较远的像素不会对边缘上的像素值影响太多,这样就保证了边缘附近像素值的保存。但是,由于保存了过多的高频信息,对于彩色图像里的高频噪声,双边滤波器不能够干净地滤掉,只能对于低频信息进行较好地滤波。

在双边滤波器中,输出像素的值依赖于邻域像素值的加权值组合,公式如下。



而加权系数w(i，j，k，l)取决于定义域核和值域核的乘积。

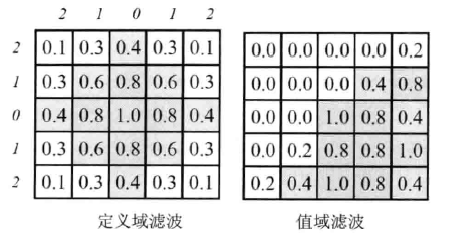
其中定义域核表示如下。



值域核表示如下。



定义域滤波和值域滤波如下图所示。



两者相乘后,就会产生依赖于数据的双边滤波权重函数,如下。



4.非线性滤波相关核心API函数

a. 中值滤波: medianB|ur函数

medianBlur函数使用中值滤波器来平滑(模糊)处理一张图片,从src输入,结果从dst输出。对于多通道图片,它对每一个通道都单独进行处理,并且支持就地操作( In-placeoperation)。函数原型如下。

C++: void medianBlur(InputArray src, OutputArray dst, int ksize)

参数详解如下。

* 第一个参数, InputArray类型的src,函数的输入参数,填1、3或者4通道的Mat类型的图像。当ksize为3或者5的时候,图像深度需为CV\_8U、CV\_16U、CV\_32F其中之一,而对于较大孔径尺寸的图片,它只能是CV\_8U。
* 第二个参数: OutputArray类型的dst,即目标图像,函数的输出参数,需要和源图片有一样的尺寸和类型。我们可以用Mat: Clone,以源图片为模板,来初始化得到如假包换的目标图。
* 第三个参数:int类型的ksize,孔径的线性尺寸( aperture linear size),注意这个参数必须是大于1的奇数,比如:3、5、7、9……

b. 双边滤波: bilateralFilter函数

此函数的作用是用双边滤波器来模糊处理一张图片,由src输入图片,结果于dst输出。函数原型如下。

C++: void bilateralFilter(InputArray src, OutputArraydst, int d, double sigmaColor, double sigmaSpace, int borderType=BORDER\_DEFAULT)

参数详解如下。

* 第一个参数: InputArray类型的sre,输入图像,即源图像,需要为8位或者浮点型单通道、三通道的图像。
* 第二个参数: OutputArray类型的dst,即目标图像,需要和源图片有一样的尺寸和类型。
* 第三个参数:int类型的d,表示在过滤过程中每个像素邻域的直径。如果这个值被设为非正数,那么 OpenCV会从第五个参数 sigmaSpace来计算出
* 第四个参数: double类型的 sigma Color,颜色空间滤波器的 sigma值。这个参数的值越大,就表明该像素邻域内有越宽广的颜色会被混合到一起,立生较大的半相等颜色区域。
* 第五个参数: double类型的 sigmaSpace,坐标空间中滤波器的 sigma值,坐标空间的标注方差。它的数值越大,意味着越远的像素会相互影响,从而使更大的区域中足够相似的颜色获取相同的颜色。当d>0时,d指定了邻域大小且与 sigmaSpace无关。否则,d正比于 sigmaSpace

第六个参数:int类型的 border Type,用于推断图像外部像素的某种边界模式。注意它有默认值 BORDER\_DEFAULT。

##### 1.2.2.3 形态学滤波

1.形态学概述

形态学( morphology)一词通常表示生物学的一个分支,该分支主要研究动植物的形态和结构。而我们图像处理中的形态学,往往指的是数学形态学。下面一起来了解数学形态学的概念。

数学形态学( Mathematical morphology)是一门建立在格论和拓扑学基础之上的图像分析学科,是数学形态学图像处理的基本理论。其基本的运算包括:二值腐蚀和膨胀、二值开闭运算、骨架抽取、极限腐蚀、击中击不中变换、形态学梯度、Top-hat变换、颗粒分析、流域变换、灰值腐蚀和膨胀、灰值开闭运算、灰值形态学梯度等。

简单来讲,形态学操作就是基于形状的一系列图像处理操作。 OpenCV为进行图像的形态学变换提供了快捷、方便的函数。最基本的形态学操作有两种,分别是:膨胀( dilate)与腐蚀( erode)。

膨胀与腐蚀能实现多种多样的功能,主要如下。

* 消除噪声
* 分割( isolate)出独立的图像元素,在图像中连接(join)相邻的元素
* 寻找图像中的明显的极大值区域或极小值区域
* 求出图像的梯度。

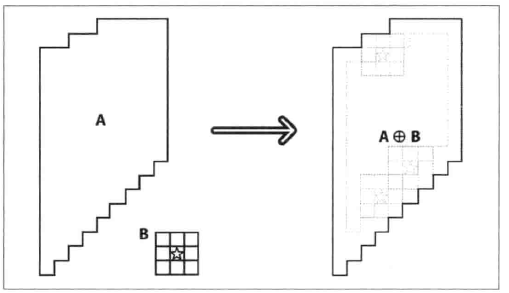
腐蚀和膨胀是对白色部分(高亮部分)而言的,不是黑色部分。膨胀是图像中的高亮部分进行膨胀,类似于“领域扩张”,效果图拥有比原图更大的高亮区域:腐蚀是原图中的高亮部分被腐蚀,类似于“领域被蚕食”,效果图拥有比原图更小的高亮区域。

2.膨胀

膨胀( dilate)就是求局部最大值的操作。从数学角度来说,膨胀或者腐蚀操作就是将图像(或图像的一部分区域,称之为A)与核(称之为B)进行卷积。

核可以是任何形状和大小,它拥有一个单独定义出来的参考点,我们称其为锚点( anchorpoint)。多数情况下,核是一个小的,中间带有参考点和实心正方形或者圆盘。其实,可以把核视为模板或者掩码。

而膨胀就是求局部最大值的操作。核B与图形卷积,即计算核B覆盖的区域的像素点的最大值,并把这个最大值赋值给参考点指定的像素。这样就会使图像中的高亮区域逐渐增长,如下图所示。这就是膨胀操作的初衷。

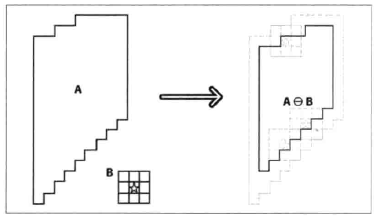


膨胀操作示例图

3.腐蚀

大家应该知道,膨胀和腐蚀( erode)是相反的一对操作,所以腐蚀就是求局部最小值的操作。

我们一般都会把腐蚀和膨胀进行对比理解和学习。下文就可以看到,两者的函数原型也是基本一样的。腐蚀操作示例如下图所示。



腐蚀操作示例图

4.相关 Open CV源码分析溯源

在… \openc\sources\modules\imgprocl\src\morph. cpp路径中,我们可以发现erode(腐蚀)函数和 dilate(膨胀)函数的源码,如下。

void cv:: erode( InputArray src, OutputArraydst, InputArray kernel,Point anchor, int

iterations,int borderType, constScalar& borderValue)

//调用 morphop函数,并设定标识符为 MORPH\_ERODE

morphop( MORPH\_ERODE, src, dst, kernel, anchor, iterations,borderType, bordervalue );

void cv: :dilate( InputArray src, OutputArray dst, InputArray kernel,Point anchor, int iterations,

int borderType, constScalar& bordervalue)

//调用 morphop函数,并设定标识符为 MORPH\_DILATE

morphop( MORPH\_DILATE, src, dst, kernel, anchor, iterations, borderType, bordervalue );

}

可以发现, erode和 dilate这两个函数内部就是调用了一下 morphOp,只是它们们调用 morphO时,第一个参数标识符不同:一个为 MORPH\_ERODE(腐蚀),一个为MORPH\_DILATE(膨胀)。

5.相关核心API函数讲解

1. 膨胀:dilate函数

dilate函数使用像素邻域内的局部极大运算符来膨胀一张图片,从src输入由dst输出。支持就地(in- place)操作。

函数原型如下。

C++: void dilate(

InputArray src,

OutputArray dst,

InputArray kernel,

Point anchor=Point(-1,1),

int iterations=1,

int borderType= BORDER\_CONSTANT,

const Scalar& borderValue=morphology DefaultBordervalue()

);

参数详解如下。

* 第一个参数, InputArray类型的src,输入图像,即源图像,填Mat类的对象即可。图像通道的数量可以是任意的,但图像深度应为CV\_8U、CV\_16U、CV\_16S、CV\_32F或CV\_64F其中之一。
* 第二个参数, OutputArray类型的dst,即目标图像,需要和源图片有一样的尺寸和类型。
* 第三个参数, InputArray类型的 kernel,膨胀操作的核。当为NULL时,表

示的是使用参考点位于中心3×3的核。

我们一般使用函数 getStructuringElement配合这个参数的使用。

Get StructuringElement函数会返回指定形状和尺寸的结构元素(内核矩阵)。其中, get StructuringElement函数的第一个参数表示内核的形状,有如下三种形状可以选择。

* 矩形: MORPH\_RECT;
* 交叉形: MORPH\_CROSS;
* 椭圆形: MORPH\_ELLIPSE。

而 getStructuringElement函数的第二和第三个参数分别是内核的尺寸以及锚

点的位置。

一般在调用 erode以及 dilate函数之前,先定义一个Mat类型的变量来获得

getstructuringElement函数的返回值。对于锚点的位置,有默认值Poin(-1,-1),表示锚点位于中心。此外,需要注意,十字形的 element形状唯一依赖于锚点的位置,而在其他情况下,锚点只是影响了形态学运算结果的偏移。

getStructuringElement函数相关的调用示例代码如下。

int g\_nStructElementsize=3;//结构元素(内核矩阵)的尺寸

//获取自定义核

Mat element=getStructuringElement(MORPH\_RECT,

Size(2\*g\_nStructElementsize+1, 2\*g\_nStructElementSize+1),

Point( g\_nStructElementSize, g\_nstructElementSize ));

调用之后,我们可以在接下来调用 erode或 dilate函数时,在第三个参数填保存了 getStructuringElement返回值的Mat类型变量。对应于上面的示例,就是element变量。

* 第四个参数, Point类型的 anchor,锚的位置,其有默认值(-1,-1),表示锚位于中心。
* 第五个参数,int类型的 iterations,迭代使用 erode()函数的次数,默认值为1。
* 第六个参数,int类型的 borderType,用于推断图像外部像素的某种边界模式。注意它有默认值 BORDER\_DEFAULT。
* 第七个参数, const scalar&类型的 borderValue,当边界为常数时的边界值,有默认值 morphologyDefaultBorderValue(),一般不用去管它。需要用到它时,可以看官方文档中的 createMorphologyFilter()函数,以得到更详细的解释。

使用 erode函数,一般只需要填前面的三个参数,后面的四个参数都有默认值,而且往往会结合 getStructuringElement一起使用。

1. 腐蚀: erode函数

erode函数使用像素邻域内的局部极小运算符来腐蚀一张图片,从src输入由dst输出。支持就地(in- place)操作。

看一下函数原型,如下。

C++: void erode(

InputArray src,

OutputArray dst,

InputArray kernel,

Point anchor=Point (-1,-1),

int iterations=1,

int borderType=BORDER\_CONSTANT,

const Scalar& borderValue=morphologyDefaultBordervalue()

);

参数详解如下。

* 第一个参数, InputArray类型的src,输入图像,即源图像,填Mat类的对象即可。图像通道的数量可以是任意的,但图像深度应为CV\_8U、CV\_16U、CV\_16S、CV\_32F或CV\_64F其中之一。
* 第二个参数, OutputArray类型的dst,即目标图像,需要和源图片有一样的尺寸和类型。
* 第三个参数, InputArray类型的 kernel,腐蚀操作的内核。为NULL时,表示的是使用参考点位于中心3x3的核。一般使用函数 getStructuringElement 配合这个参数的使用。 getStructuring Element函数会返回指定形状和尺寸的结构元素(内核矩阵,具体看上文中 dilate函数的第三个参数讲解部分。
* 第四个参数, Point类型的 anchor,锚的位置。其有默认值(-1,-1),表示锚位于单位( element)的中心,一般不用管它。
* 第五个参数,int类型的 Iterations,迭代使用 eroded()函数的次数,默认值为1。
* 第六个参数,int类型的 borderType,用于推断图像外部像素的某种边界模式。注意它有默认值 BORDER\_DEFAULT。
* 第七个参数, const Scalar&类型的 borderValue,当边界为常数时的边界值,有默认值 morphologyDefaultBorderValue(),一般不用去管它。需要用到它时,可以看官方文档中的 createMorphologyFilter()函数以得到更详细的解释。

同样的,使用 erode函数,一般只需要填前面的三个参数,后面的四个参数都有默认值。而且往往结合getStructuringElement一起使用。

##### 1.2.2.4 基于OpenCV的边缘检测

1. 边缘检测的一般步骤

1. 滤波

边缘检测的算法主要是基于图像强度的一阶和二阶导数,但导数通常对噪声很敏感,因此必须采用滤波器来改善与噪声有关的边缘检测器的性能。常见的滤波方法主要有高斯滤波,即采用离散化的高斯函数产生一组归一化的高斯核,然后基于高斯核函数对图像灰度矩阵的每一点进行加权求和。

1. 增强

增强边缘的基础是确定图像各点邻域强度的变化值。增强算法可以将图像灰度点邻域强度值有显著变化的点凸显岀来。在具体编程实现时,可通过计算梯度幅值来确定。

1. 检测

经过增强的图像,往往邻域中有很多点的梯度值比较大,而在特定的应用中这些点并不是要找的边缘点,所以应该采用某种方法来对这些点进行取舍。实际工程中,常用的方法是通过阈值化方法来检测。

另外,需要注意,下文中讲到的 Laplacian算子、 sobel算子和 Scharr算子都是带方向的,所以,示例中我们分别写了X方向、Y方向和最终合成的的效果图。

2. canny 算子

1. canny算子简介

Canny边缘检测算子是 John F. Canny于1986年开发出来的一个多级边缘检测算法。更为重要的是,Canny创立了边缘检测计算理论( Computational theory ofedge detection),解释了这项技术是如何工作的。Canny边缘检测算法以 Canny的名字命名,被很多人推崇为当今最优的边缘检测的算法。

其中,canny的目标是找到一个最优的边缘检测算法,让我们看一下最优边缘检测的三个主要评价标准。

* 低错误率:标识出尽可能多的实际边缘,同时尽可能地减少噪声产生的误报。
* 高定位性:标识出的边缘要与图像中的实际边缘尽可能接近。
* 最小响应:图像中的边缘只能标识一次,并且可能存在的图像噪声不应标识为边缘。

为了满足这些要求,caany使用了变分法,这是一种寻找满足特定功能的函数的方法。最优检测用4个指数函数项的和表示,但是它非常近似于高斯函数的一阶导数。

1. Canny边缘检测的步骤
2. 【第一步】消除噪声

一般情况下,使用高斯平滑滤波器卷积降噪。以下显示了一个size=5的高斯内核示例:



1. 【第二步】计算梯度幅值和方向

此处,按照 Sobel滤波器的步骤来操作。

①运用一对卷积阵列(分别作用于x和y方向)

②使用下列公式计算梯度幅值和方向





而梯度方向一般取这4个可能的角度之一——0度，45度，90度，135度。

1. 【第三步】非极大值抑制

这一步排除非边缘像素,仅仅保留了一些细线条(候选边缘)。

1. 【第四步】滞后阈值

这是最后一步, Canny使用了滞后阈值,滞后阈值需要两个阈值(高阈值和低阈值):

①若某一像素位置的幅值超过高阈值,该像素被保留为边缘像素。

②若某一像素位置的幅值小于低阈值,该像素被排除。

③若某一像素位置的幅值在两个阈值之间,该像素仅仅在连接到一个高于高阈值的像素时被保留。

1. Canny边缘检测:canny()函数

Canny函数利用canny算子来进行图像的边缘检测操作。

C++: void Canny(InputArray image, OutputArray edges, double thresholdl,

double threshold2, int aperturesize=3, bool L2gradient=false)

* 第一个参数, InputArray类型的 image,输入图像,即源图像,填Mat类的对象即可,且需为单通道8位图像。
* 第二个参数, OutputArray类型的 edges,输出的边缘图,需要和源图片有一样的尺寸和类型。
* 第三个参数, double类型的 threshold1,第一个滞后性阈值。
* 第四个参数, double类型的 threshold2,第二个滞后性阈值。
* 第五个参数,int类型的 apertureSize,表示应用 Sobel算子的孔径大小,其

有默认值3。

* 第六个参数,bool类型的L2 gradient,一个计算图像梯度幅值的标识,有默认值 false。

需要注意的是,这个函数阈值1和阈值2两者中较小的值用于边缘连接,而较大的值用来控制强边缘的初始段,推荐的高低阈值比在2:1到3:1之间。

3. sobel算子

1. sobel算子的基本概念

Sobel算子是一个主要用于边缘检测的离散微分算子( discrete differentiation operator)。它结合了高斯平滑和微分求导,用来计算图像灰度函数的近似梯度。在图像的任何一点使用此算子,都将会产生对应的梯度矢量或是其法矢量。

1. sobel算子的计算过程

我们假设被作用图像为I然后进行如下操作。

(1)分别在x和y两个方向求导。

①水平变化:将I与一个奇数大小的内核Gx进行卷积。比如,当内核大小为3时,Gx的计算结果为:



②垂直变化:将:I与一个奇数大小的内核进行卷积。比如,当内核大小为3时,计算结果为



(2)在图像的每一点,结合以上两个结果求出近似梯度:



另外有时,也可用下面更简单的公式代替:



1. 使用 Sobel算子:Sobel()函数

Sobel函数使用扩展的 Sobel算子,来计算一阶、二阶、三阶或混合图像差分。

C++: void Sobel{

InputArray src,

OutputArray dst,

int ddepth,

int dx,

int dy,

int ksize=3

double scale=l,

double delta=0,

int borderType=BORDER\_DEFAULT );

(1)第一个参数, InputArray类型的src,为输入图像,填Mat类型即可。

(2)第二个参数, OutputArray类型的dst,即目标图像,函数的输出参数,需要和源图片有一样的尺寸和类型。

(3)第三个参数,int类型的 ddepth,输出图像的深度,支持如下 src.depth()和ddepth的组合:

* 若src. depth()=CV\_8U,取ddepth=-1/CV\_16S/CV\_32F/CV\_64F
* 若src. depth()=CV\_16U/CV\_16S,取ddepth=-l/CV\_32F/CV\_64F
* 若 src. depth()=CV\_32F,取 ddepth=-1/CV\_32F/CV\_64F
* 若 src. depth()=CV\_64F,取 ddepth=-l/CV\_64F

(4)第四个参数,int类型dx,x方向上的差分阶数。

(5)第五个参数,int类型dy,y方向上的差分阶数。

(6)第六个参数,int类型 ksize,有默认值3,表示 Sobel核的大小;必须取1、3、5或7。

(7)第七个参数, double类型的 scale,计算导数值时可选的缩放因子,默认值是1,表示默认情况下是没有应用缩放的。可以在文档中查阅 getDerivKernels 的相关介绍,来得到这个参数的更多信息。

(8)第八个参数, double类型的delta,表示在结果存入目标图(第二个参数dst)之前可选的 delta值,有默认值0。

(9)第九个参数,int类型的 borderType,边界模式,默认值为 BORDER\_DEFAULT。这个参数可以在官方文档中 borderInterpolate处得到更详细的信息。

一般情况下,都是用 ksize× ksize内核来计算导数的。然而,有一种特殊情况——当 size为1时,往往会使用3x1或者1x3的内核。且这种情况下,并没有进行高斯平滑操作。

一些补充说明如下。

①当内核大小为3时, Sobel内核可能产生比较明显的误差(毕竟, Sobel 算子只是求取了导数的近似值而已)。为解决这一问题, OpenCV提供了 Scharr函数,但该函数仅作用于大小为3的内核。该函数的运算与 Sobel函数一样快,但结果却更加精确,其内核是这样的:

②因为 Sobel算子结合了高斯平滑和分化( differentiation),因此结果会具有更多的抗噪性。大多数情况下,我们使用 sobel函数时,取【 xorder=1, yorder=0, ksize=3】来计算图像X方向的导数,【 xorder=0, yonder=1, ksize=3】来计算图像y方向的导数。

计算图像X方向的导数,取【 xorder=1, yorder=0, ksize=3】情况对应的内核:



而计算图像Y方向的导数,取【 xorder=0, yorder=1, ksize=3】对应的内核:



4. Laplacian算子

1. Laplacian算子简介

Laplacian算子是n维欧几里德空间中的一个二阶微分算子,定义为梯度grad的散度div。因此如果f是二阶可微的实函数,则f的拉普拉斯算子定义如下。

(1)f的拉普拉斯算子也是笛卡儿坐标系xi中的所有非混合二阶偏导数求和。

(2)作为一个二阶微分算子,拉普拉斯算子把C函数映射到C函数。对于k≥2,表达式(1)(或(2)定义了一个算子△:C(R)→C(R);或更一般地,对于任何开集Ω,定义了一个算子△:C(Ω)→C(Ω)。

根据图像处理的原理可知,二阶导数可以用来进行检测边缘。因为图像是“二维”,需要在两个方向进行求导。使用 Laplacian算子将会使求导过程变得简单。

Laplacian算子的定义：



需要说明的是,由于 Laplacian使用了图像梯度,它内部的代码其实是调用了Sobel算子的。

1. 计算拉普拉斯变换: Laplacian()函数

Laplacian函数可以计算出图像经过拉普拉斯变换后的结果

C++: void Laplacian(InputArray src, OutputArray dst, int ddepth, int ksize=1, double scale=1, double delta=0,intborderType=BORDER\_DEFAULT );

* 第一个参数, InputArray类型的 Image,输入图像,即源图像,填Mat类的对象即可,且需为单通道8位图像。
* 第二个参数, Output Array类型的 edges,输出的边缘图,需要和源图片有一样的尺寸和通道数。
* 第三个参数,int类型的ddept,目标图像的深度。
* 第四个参数,int类型的ksize,用于计算二阶导数的滤波器的孔径尺寸,大小必须为正奇数,且有默认值1。
* 第五个参数, double类型的 scale,计算拉普拉斯值的时候可选的比例因子,

有默认值1。

* 第六个参数, double类型的 delta,表示在结果存入目标图(第二个参数dst)之前可选的 delta值,有默认值0。
* 第七个参数,int类型的 borderType,边界模式,默认值为 BORDER\_DEFAULT。这个参数可以在官方文档中 borderInterpolateO处得到更详细的信息。

Laplacian()函数其实主要是利用 sobel算子的运算。它通过加上 sobel算子运算出的图像x方向和y方向上的导数:来得到载入图像的拉普拉斯变换结果。

其中, sobel算子( ksize>1)如下:



而当ksize=1时, Laplacian()函数采用以下3x3的孔径:



5. scharr滤波器

我们一般直接称scharr为滤波器,而不是算子。上文已经讲到,它在 OpenCv中主要是配合 Sobel算子的运算而存在的。下面让我们直接来看看其函数讲解。

1. 计算图像差分:Scharr()函数

使用 Scharr滤波器运算符计算x或y方向的图像差分。其实它的参数变量和Sobel基本上是一样的,除了没有ksize核的大小。

C++: void Scharr(

InputArray src,//源图

OutputArray dst,//目标图

int ddepth,//图像深度

int dx,//x方向上的差分阶数

int dy,//y方向上的差分阶数

double sca1e=1,//缩放因子

double delta=0,// delta值

intborderType= BORDER\_DEFAULT)//边界模式

(1)第一个参数, InputArray类型的src,为输入图像,填Mat类型即可。

(2)第二个参数, OutputArray类型的dst,即目标图像,函数的输出参数,需要和源图片有一样的尺寸和类型。

(3)第三个参数,int类型的ddepth,输出图像的深度,支持如下 src.depth() 和 ddepth的组合:

* 若src.depth()=CV\_8U, 取ddepth=-1/CV\_16S/CV\_32F/CV\_64F
* 若 src.depth()=CV\_16U/CV\_16S,取 ddepth=-l/CV\_32F/CV\_64F
* 若 src.depth()=CV\_32F,取 ddepth=-l/CV\_32F/CV\_64F
* 若 src.depth()=CV\_64F,取 ddepth=-l/CV\_64F

(4)第四个参数,int类型dx,x方向上的差分阶数。

(5)第五个参数,int类型dy,y方向上的差分阶数。

(6)第六个参数, double类型的 scale,计算导数值时可选的缩放因子,默认值是1,表示默认情况下是没有应用缩放的。我们可以在文档中查阅getDerivKernels的相关介绍,来得到这个参数的更多信息。

(7)第七个参数, double类型的 delta,表示在结果存入目标图(第二个参数dst)之前可选的dlta值,有默认值0。

(8)第八个参数,int类型的 borderType,边界模式,默认值为 BORDER\_DEFAULT。这个参数可以在官方文档中 borderInterpolate处得到更详细的信息。

不难理解,如下两者是等价的,即:

Scharr(src, dst, ddepth, dx, dy, scale, delta, borderType);与Sobel(src, dst, ddepth, dx, dy,

CV \_SCHARR, scale, delta, borderType);

#### 1.2.3 图像分割

##### 1.2.3.1 图像金字塔与图片尺寸缩放

1. 关于图像金字塔

图像金字塔是图像中多尺度表达的一种,最主要用于图像的分割,是一种以多分辨率来解释图像的有效但概念简单的结构。

图像金字塔最初用于机器视觉和图像压缩,一幅图像的金字塔是一系列以金字塔形状排列的,分辨率逐步降低且来源于同一张原始图的图像集合。其通过梯次向下采样获得,直到达到某个终止条件才停止采样。

金字塔的底部是待处理图像的高分辨率表示,而顶部是低分辨率的近似。

我们将一层一层的图像比喻成金字塔,层级越高,则图像越小,分辨率越低。

一般情况下有两种类型的图像金字塔常常出现在文献和以及实际运用中。它们分别是:

* 高斯金字塔( Gaussianpyramid)—用来向下采样,主要的图像金字塔。
* 拉普拉斯金字塔( Laplacianpyramid)—用来从金字塔低层图像重建上层未采样图像,在数字图像处理中也即是预测残差,可以对图像进行最大程度的还原,配合高斯金字塔一起使用。

两者的简要区别在于:高斯金字塔用来向下降采样图像,而拉普拉斯金字塔则用来从金字塔底层图像中向上采样,重建一个图像。

要从金字塔第i层生成第i+1层(我们将第i+1层表示为),我们先要用高斯核对;进行卷积,然后删除所有偶数行和偶数列,新得到图像面积会变为源图像的四分之一。按上述过程对输入图像执行操作就可产生出整个金字塔。

当图像向金字塔的上层移动时,尺寸和分辨率会降低。 OpenCV中,从金字塔中上一级图像生成下一级图像的可以用 PryDown,而通过 PryUp将现有的图像在每个维度都放大两遍。

图像金字塔中的向上和向下采样分别通过 OpenCV的函数 pyrUp和 pyrDown实现。

概括起来就是:

* 对图像向上采样— pyrUp函数；
* 对图像向下采样— pyrDown函数。

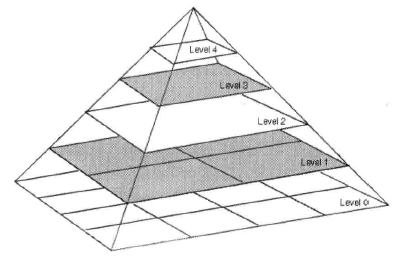
这里的向下与向上采样,是针对图像的尺寸而言的(和金字塔的方向相反),向上就是图像尺寸加倍,向下就是图像尺寸减半。

但需要注意的是, PryUp和 PryDown不是互逆的,即 PryUp不是降采样的逆操作。这种情况下,图像首先在每个维度上扩大为原来的两倍,新增的行(偶数行)以0填充。然后给指定的滤波器进行卷积(实际上是一个在每个维度都扩大为原来两倍的过滤器)去估计“丢失”像素的近似值。

PryDown()是一个会丢失信息的函数。为了恢复原来更高的分辨率的图像,我们要获得由降釆样操作丢失的信息,这些数据就和拉普拉斯金字塔有关系了。

1. 高斯金字塔

高斯金字塔是通过高斯平滑和亚采样获得一些列下采样图像,也就是说第K层高斯金字塔通过平滑、亚采样就可以获得K+1层高斯图像。高斯金字塔包含了系列低通滤波器,其截止频率从上一层到下一层以因子2逐渐增加,所以高斯金字塔可以跨越很大的频率范围。金字塔的图像如下图所示;



图像金字塔演示图

另外,每一层都按从下到上的次序编号,层级(表示为尺寸小于第i层)。

1. 对图像的向下取样

为了获取层级为的金字塔图像,我们采用如下方法：

1. 对图像进行高斯内核卷积;
2. 将所有偶数行和列去除。

得到的图像即为的图像。显而易见,结果图像只有原图的四分之一。通过对输入图像;〔原始图像)不停迭代以上步骤就会得到整个金字塔。同时我们也可以看到,向下取样会逐渐丢失图像的信息。

以上就是对图像的向下取样操作,即缩小图像。

1. 对图像的向上取样

如果想放大图像,则需要通过向上取样操作得到,具体做法如下。

1. 将图像在每个方向扩大为原来的两倍,新增的行和列以0填充。
2. 使用先前同样的内核(乘以4)与放大后的图像卷积,获得“新增像素”的近似值。

得到的图像即为放大后的图像,但是与原来的图像相比会发觉比较模糊,因为在缩放的过程中已经丢失了一些信息。如果想在缩小和放大整个过程中减少信息的丢失,这些数据就形成了拉普拉斯金字塔。

接下来一起看一下拉普拉斯金字塔的概念。

1. 拉普拉斯金字塔

下式是拉普拉斯金字塔第i层的数学定义:



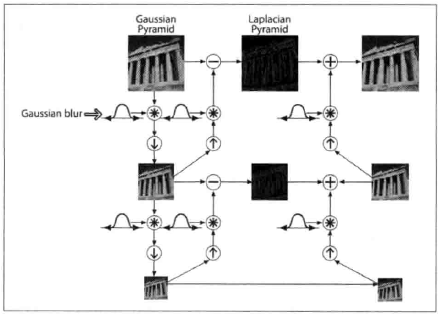
式中的表示第i层的图像。而UP()操作是将源图像中位置为(x,y)的像素映射到目标图像的(2x+1,2y+1)位置,即在进行向上取样。符号表示卷积,为的高斯内核。

我们下文将要介绍的 pryUp,就是在进行上面这个式子的运算。

因此,我们可以直接用 OpenCV进行拉普拉斯运算:= -PyrUp()

也就是说,拉普拉斯金字塔是通过源图像减去先缩小后再放大的图像的一系列图像构成的。

整个拉普拉斯金字塔运算过程可以通过下图来概括.



拉普拉斯金字塔运算过程

所以,我们可以将拉普拉斯金字塔理解为高斯金字塔的逆形式。

另外再提一点,关于图像金字塔非常重要的一个应用就是图像分割。图像分割的话,先要建立一个图像金字塔,然后对和的像素直接依照对应的关系,建立起“父与子”关系。而快速初始分割可以先在金字塔高层的低分辨率图像上完成,然后逐层对分割加以优化。

1. 图像金字塔相关API函数

图像金字塔相关API函数主要是pyrUp、 pyrDown这一对,下面分别对其进行讲解。

1. 向上采样: pyrUp()函数

pyrUp()函数的作用是向上采样并模糊一张图像,说白了就是放大一张图片。

C++: void pyrUp(InputArray src, OutputArraydst, const size&

dstsize=Size(), int borderType= BORDER\_DEFAULT)

第一个参数, InputArray类型的src,输入图像,即源图像,填Mat类的对

象即可。

第二个参数, OutputArray类型的dst,输出图像,和源图片有一样的尺寸

和类型。

第三个参数, const size&类型的dstsize,输出图像的大小;有默认值 Size(),

即默认情况下,由Size(src.cols\*2, src rows\*2)来进行计算,且一直需要

满足下列条件:

|dstsize. width-src.cols\*2 I <=(dstsize.width.mod2)

I dstsize. height -src. rows\*2 I <=(dstsize. heiht mod2)

第四个参数,int类型的 borderType,边界模式,一般不用±管它。

pyrUp函数执行高斯金字塔的采样操作,其实它也可以用于拉普拉斯金字塔的。

首先,它通过插入可为零的行与列,对源图像进行向上取样操作,然后将结果与 pyrDown()乘以4的内核做卷积。

1. 采样: pyrDown（）函数

pyrDown()函数的作用是向下采样并模糊一张图片,说白了就是缩小一张图片。

C++: void pyrDown(InputArray src, OutputArray dst, const size& dstsize=Size(), int borderType=BORDER\_DEFAULT)

* 第一个参数, InputArray类型的src,输入图像,即源图像,填Mat类的对象即可。
* 第二个参数, OutputArray类型的dst,输出图像,和源图片有一样的尺寸和类型。
* 第三个参数, const Size&类型的 dstsize,输出图像的大小;有默认值 Size(), 即默认情况下,由 Size size( src. cols+1)/2,(src,rows+1)/2)来进行计算,且一直需要满足下列条件:

I dstsize. width \*2-src.cols |<=2

I dstsize heigth\*2-src.rows | <=2

该 pyrDown函数执行了高斯金字塔建造的向下采样的步骤。首先,它将源图像与如下内核做卷积运算:



接着,它通过对图像的偶数行和列做插值来进行向下采样操作。

##### 1.2.3.2 阈值化

在对各种图形进行处理操作的过程中,我们常常需要对图像中的像素做出取舍与决策,直接剔除一些低于或者高于一定值的像素。

阈值可以被视作最简单的图像分割方法。比如,从一副图像中利用阈值分割出我们需要的物体部分(当然这里的物体可以是一部分或者整体)。这样的图像分割方法基于图像中物体与背景之间的灰度差异,而且此分割属于像素级的分割。为了从一副图像中提取出我们需要的部分,应该用图像中的每一个像素点的灰度值与选取的阈值进行比较,并作出相应的判断。注意:阈值的选取依赖于具体的问题。即物体在不同的图像中有可能会有不同的灰度值。

一旦找到了需要分割的物体的像素点,可以对这些像素点设定一些特定的值来表示。例如,可以将该物体的像素点的灰度值设定为“0”(黑色),其他的像素点的灰度值为“255”(白色)。当然像素点的灰度值可以任意,但最好设定的两种颜色对比度较强,以方便观察结果。

OpenCV2.X中, Threshold()函数(基本阈值操作)和 adaptiveThreshold()函数(自适应阈值操作)可以完成这样的要求。它们的基本思想是:给定一个数组和一个阈值,然后根据数组中的每个元素的值是高于还是低于阈值而进行一些处理。下面,我们将对这两个函数分别进行剖析。

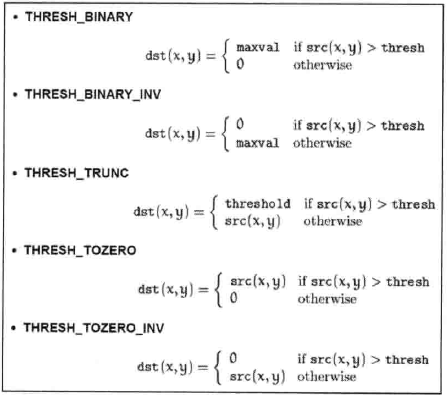
1.固定阈值操作: Threshold()函数

函数 Threshold()对单通道数组应用固定阈值操作。该函数的典型应用是对灰度图像进行阈值操作得到二值图像,( compare()函数也可以达到此目的)或者是去掉噪声,例如过滤很小或很大象素值的图像点。

C++: double threshold(InputArray src, OutputArray dst, double thresh,double maxval,

int type)

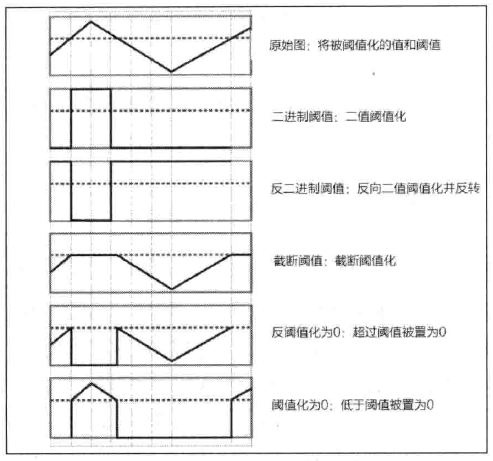
* 第一个参数, InputArray类型的src,输入数组,填单通道,8或32位浮点类型的Mat即可。
* 第二个参数, OutputArray类型的dst,函数调用后的运算结果存在这里,即这个参数用于存放输出结果,且和第一个参数中的Mat变量有一样的尺寸和类型
* 第三个参数, double类型的 thresh,阈值的具体值。
* 第四个参数, double类型的 maxval,当第五个参数阈值类型type取 CV \_THRESH\_BINARY或CⅤ\_THRESH\_BINARY\_INV时阈值类型时的最大值(对应地, OpenCV2中可以为 CV\_THRESH\_BINARY和CV\_THRESH\_BINARY\_INV)。
* 第五个参数,int类型的type,阈值类型。 threshold()函数支持的对图像取阈值的方法由其确定,具体用法如下图。



Threshold(）函数中阈值类型选项对应的操作

上述标识符依次取值分别为0,1,2,3,4。

而针对上述公式, 一一对应的图形化的阈值描述如图6.71所示( THRESH\_BINARY对应二进制阈值,依次类推)。



Threshold()函数中不同阈值类型的操作结果

2.自适应阈值操作: adaptiveThreshol()函数

adaptiveThreshold()函数的作用是对矩阵釆用自适应阈值操作,支持就地操

作。函数原型如下。

C++: void adaptiveThreshold(InputArray src, OutputArray dst, double

maxValue, int adaptiveMethod, int thresholdType, int blockSize, double C)

* 第一个参数, InputArray类型的src,输入图像,即源图像,填Mat类的对

象即可,且需为8位单通道浮点型图像。

* 第二个参数, OutputArray类型的dst,函数调用后的运算结果存在这里,

需和源图片有一样的尺寸和类型。

* 第三个参数, double类型的 maxValue,给像素赋的满足条件的非零值。

具体看下面的讲解。

* 第四个参数,int类型的 adaptiveMethod,用于指定要使用的自适应阈值算

法,可取值为 ADAPTIVE\_THRESH\_MEAN\_C或 ADAPTIVE\_THRESH\_GAUSSIAN\_C。

* 第五个参数,int类型的 thresholdType,阈值类型。取值必须为

THRESH\_BINARY、 THRESH\_BINARY INV其中之一。

* 第六个参数,int类型的 blockSize,用于计算阈值大小的一个像素的邻域尺

寸,取值为3、5、7等。

* 第七个参数, double类型的C,减去平均或加权平均值后的常数值。通常

其为正数,但少数情况下也可以为零或负数。

adaptiveThreshold（）函数根据如下公式,将一幅灰度图变换为一幅二值图。

当第五个参数“阈值类型” thresholdType取值为 THRESH\_BINARY时,公

式如下。



当第五个参数“阈值类型” thresholdType取值为 THRESH\_BINARY\_INV时,

公式为:



而其中的T(xy)为分别计算每个单独像素的阈值,取值如下。

对于ADAPTIVE\_THRESH\_MEAN\_C方法,阈值T(x,y)为 blockSize×blockSize邻域内(x,y)减去第七个参数C的平均值。

对于ADAPTIVE\_THRESH\_GAUSSIAN\_C方法,阈值T(x,y)为 blockSize×blockSize邻域内(x,y)减去第七个参数C与高斯窗交叉相关(( cross-correlation with a Gaussian window))的加权总和。

#### 1.2.4 图像表示与描述

将一副图像分割为区域后，接下来通常要对分割区域加以表示和描述，是“自然状态的”像素更适合计算机处理。

描述是较抽象地表示目标。好的描述应在尽可能区别不同目标的基础上对目标的尺度、平移、旋转等不敏感，这样的描述比较通用。

描述可分为对边界的描述和对区域的描述。此外，边界和边界或区域和区域之间的关系也常需要进行描述。

表示和描述是密切联系的。表示的方法对描述很重要，因为它限定了描述的精确性；而通过对目标的描述，各种表示方法才有实际意义。

表示和描述又有区别，表示侧重于数据结构，而描述侧重于区域特性以及不同区域间的联系和差别。

表示是直接具体地表示目标，好的表示方法应具有节省存储空间、易于特征计算等优点。

描述是较抽象地表示目标。好的描述应在尽可能区别不同目标的基础上对目标的尺度、平移、旋转等不敏感，这样的描述比较通用。

描述可分为对边界的描述和对区域的描述。此外，边界和边界或区域和区域之间的关系也常需要进行描述。

##### 1.2.4.1 表示方法

1.链码

a.概念

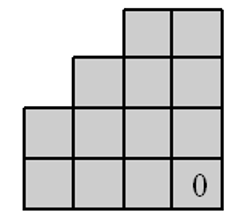
链码是对边界点的一种编码表示方法，其基本思想是利用一系列具有特定长度和方向的相连的直线段来表示目标的边界。



举例：若设起始点O的坐标为（5，5），则分别用如下4方向和8方向链码按逆时针顺序表示区域边界：

4方向链码： （5， 5）1 1 1 2 3 2 3 2 3 0 0 0； 

8方向链码： （5， 5）2 2 2 4 5 5 6 0 0 0。



b.链码表示的特点

1. 只有边界的起点需用绝对坐标表示，其余点都可只用接续方向来代表偏移量；
2. 与用坐标值相比，链码表达可大大减少边界表示所需的数据量。

c.存在的问题

直接对分割所得的目标边界编码，有可能出现如下问题：

1. 产生的码串通常很长；
2. 噪声等干扰会导致小的边界变化而使链码发生与目标整体形状无关的较大变动。

2.多边形近似

a.问题的引出

实际应用中的数字边界常由于噪声、采样等的影响而有许多较小的不规则处，这些不规则处常对链码和边界段表达产生较明显的干扰影响。

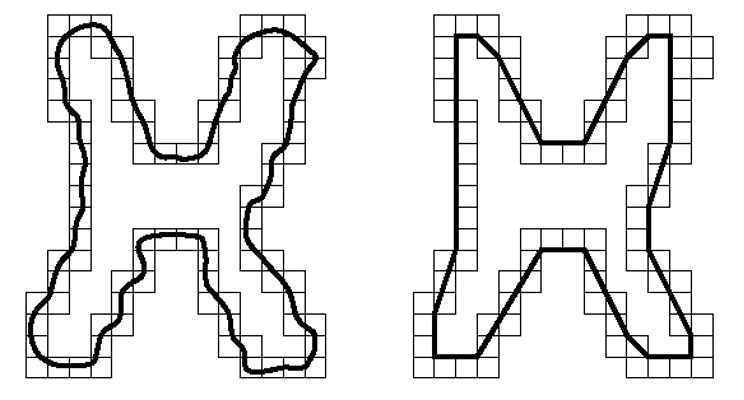
b.多边形方法的基本思想

多边形是一系列线段的封闭集合，它可用来逼近大多数使用的曲线到任意的精度。

在实际中多边形表达的目的：要用尽量少的线段来代表边界并保持边界的基本形状，从而用较简单的形式来表达和描述边界。

c.基于收缩的最小周长多边形法

将边界看成是有弹性的线，将组成边界的像素系列的内外边各看成一堵墙，如将线拉紧则可到最小周长多边形。



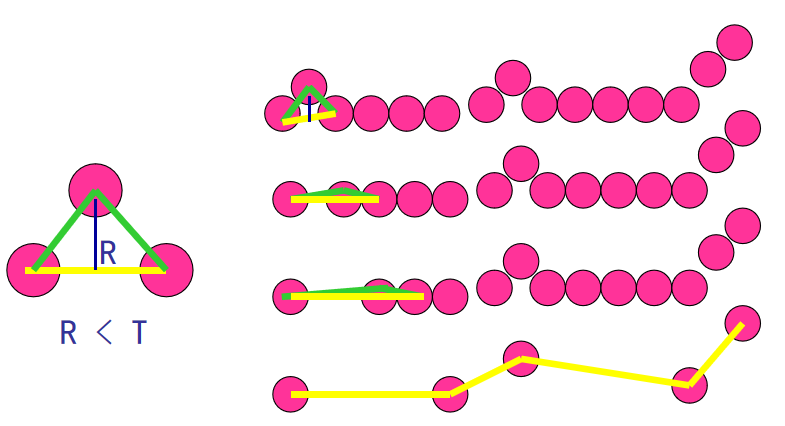
d.聚合技术

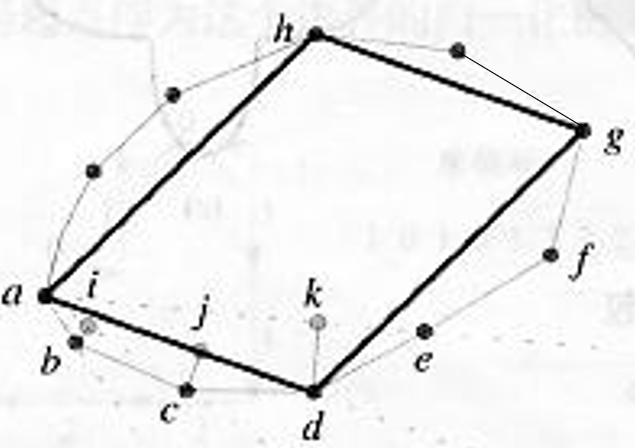
算法步骤：

1）沿着边界选两个相邻的点对，计算首尾连接直线段与原始折线段的误差R。

2）如果误差R小于预先设置的阈值T。去掉中间点，选新点对与下一相邻点对，重复1）；否则，存储线段的参数，置误差为0，选被存储线段的终点为起点，重复1）2）。

3）当程序的第一个起点被遇到，程序结束。





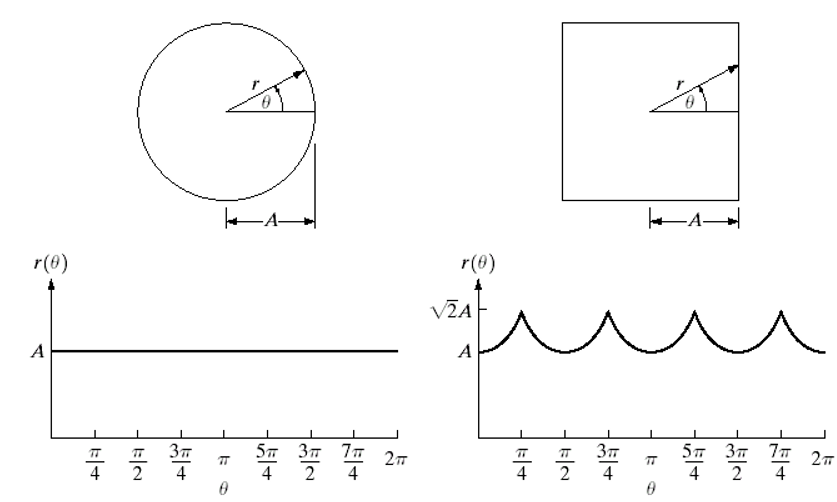
3.标记

a.基本思想

标记是边界的1-D泛函表达，其基本思想是把2-D的边界用1-D的较易描述的函数形式来表达。

b.最简单的标记方法

先对给定的物体求出质心，然后把边界点与质心的距离作为角度的函数就得到一种记。



c.存在问题

函数过分依赖于旋转和比例的变化。

d.改进措施-----旋转不变

1. 选择离质心最远的点作为起点；
2. 使用差分链码的方法。

e.改进措施-----比例不变

对函数进行正则化，使函数值总是分布在相同的值域里，比如说[0，1]。

* 利用长短轴进行正则化；
* 利用所有边界样本进行正则化。

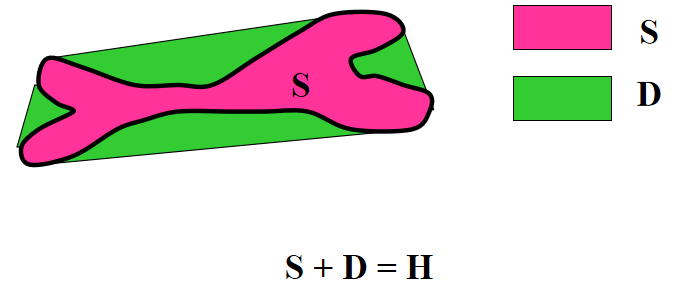
4.边界分段

a.基本概念

1. 一个任意集合S（区域）的凸起外缘H是：包含S

的最小凸起的集合。

1. H-S的差的集合被称为集合S的凸起补集D。



b. 分段算法：

给进入和离开凸起补集D的变换点打标记来划分边界段。

c. 优点：不依赖于方向和比例的变化

d. 存在问题

噪音的影响，导致出现零碎的划分。

e. 改进措施

先平滑边界，然后再分段。

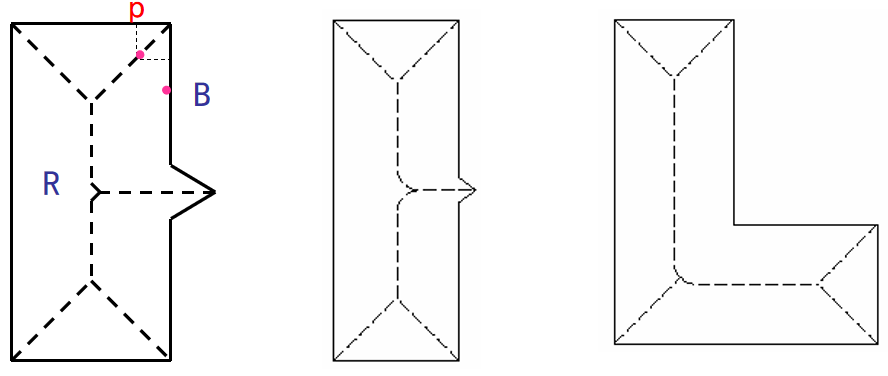
5.骨架

（1）基本思想

表示一个平面区域结构形状的一种重要方法是把它削减成图形。这种削减可以通过细化（也称为抽骨架）算法，获取区域的骨架来实现。

（2）Blum的中轴变换方法（MAT）

设：R是一个区域，B为R的边界点，对于R中的点p，找p在B上“最近”的邻居。如果p有多于一个的邻点，称它属于R的中轴（骨架）。



##### 1.2.4.2 边界描绘子

1.一些简单的描绘子

a.边界的长度

1. 定义：区域的边界长度。
2. 计算方法
3. 周长用边界所占面积表示， 也即边界点数之和， 每个点占面积为1的一个小方块。
4. 当把像素看作一个个点时，则周长用链码表示。此时，当链码值为奇数时，其长度记作 ;当链码值为偶数时，其长度记作1。即周长p表示为

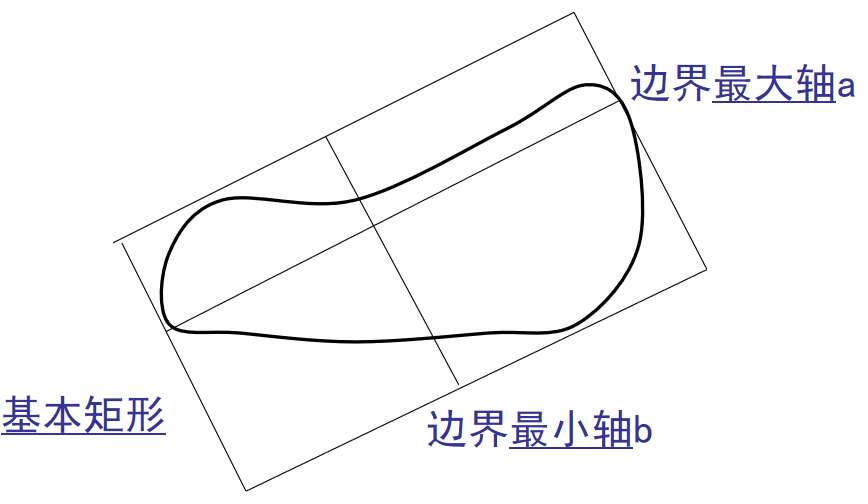


b.边界的直径



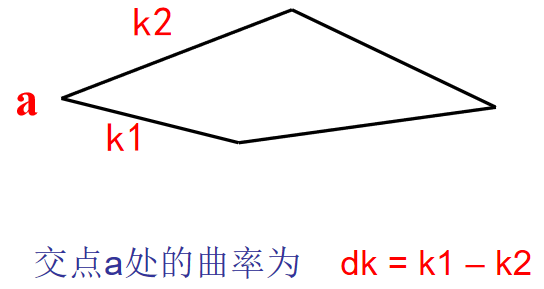
c.边界线的离心率：长轴和短轴的比率。

1. 边界最大轴a：是连接距离最远的两个点的线段。
2. 边界最小轴b：与最大轴垂直，且其长度确定的包围盒刚好包围边界。
3. 基本矩形: 包围边界的矩形。

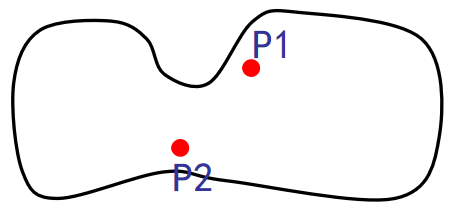


d.曲率

定义为斜率的改变率，描述了边界上各点沿边界方向的变化量。用相邻边界线段（描述为直线）的斜率差作为在边界线交点处的曲率描述子。



在一个边界点的曲率的符号描述了边界在该点的凹凸性。如果曲率大于零，则曲线凹向朝着该点法线的正向。如果曲率小于零，则曲线凹向朝着该点法线的负方向。



2.形状数

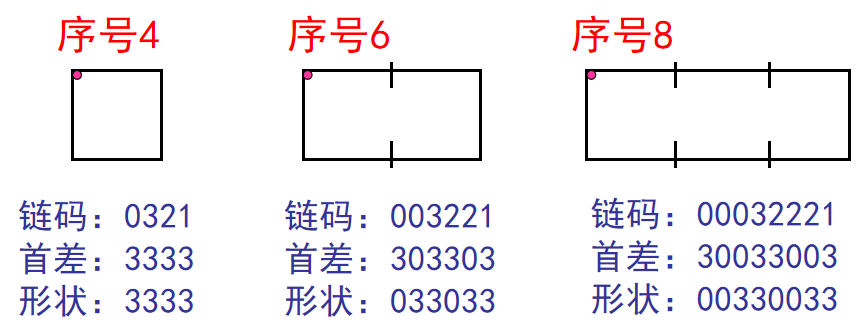
a. 定义：形状数是链码的最小值的差分码。

例如：基于4—方向的链码为:10103322,差分码为：33133030,形状数为：03033133。

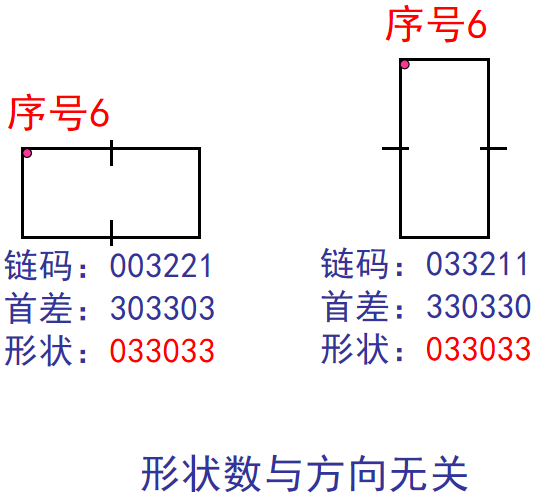
b. 形状数的阶(order)

形状数序列的长度(即链码的个数)。对闭合曲线，阶总是偶数。对凸形区域，阶对应边界外包矩形的周长。

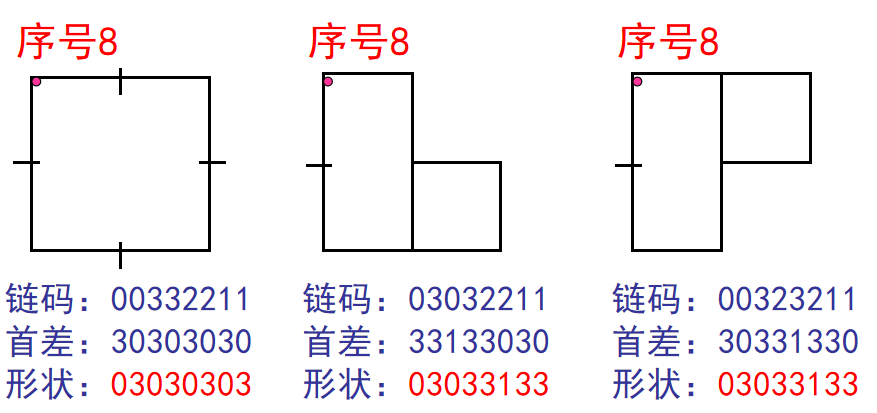
序号为4、6、8的形状数举例：



序号为6的形状数举例：



序号为8的形状数举例：



3.傅里叶描绘子

a. 基本方法：

1. 将XY平面中的曲线段转化为复平面上的1个序列，从而用复数的形式来表示给定边界上每个点（x,y）。对1个由N个点组成的封闭边界，从任一点开始绕边界1周就得到1个复数序列：



1. 进行离散傅立叶变换



系数a(u)被称为边界的傅立叶描述子。

对离散s(k)的傅里叶变换为



复系数a(u)称为边界的傅里叶描绘子。

系数的反向傅里叶变换为：



取前P个系数代替所有的傅里叶系数。，P一般选为2的指数次方的整数。





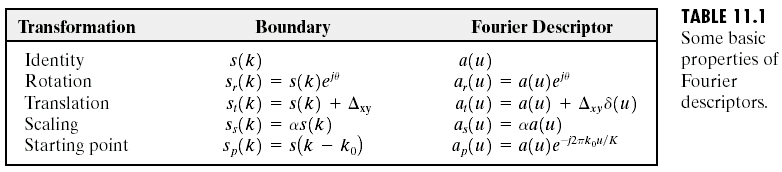
在近似边界中，存在同样数目的点，但重建每个点时并不实用同样多的项。高频元素能很好的解释细节，而低频分量决定整体形状。

b. 使用价值

1. 较少的傅立叶描述子（如4个），就可以获取边界本质的整体轮廓；
2. 这些带有边界信息的描述子，可以用来区分明显不同的边界。

c. 优点

1. 使用复数作为描述符，对于旋转、平移、放缩等操作和起始点的选取不十分敏感。
2. 几何变换的描述子可通过对函数作简单变换来获得。



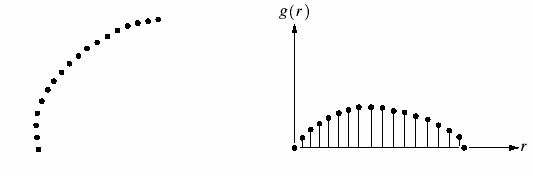
4.统计矩

a. 基本思想：

将描述形状的任务减少至描述一个一维函数，边界线段的形状可以通过简单的统计矩进行定量的描述，如均值、方差和高阶矩。

b. 统计矩的定义

1. 把边界当作直方图函数，将g(r)归一化为单位面积下的函数并把它做成直方图，就是讲作为产生值的概率。



1. 定义：

统计距 ： 均值： 

这里K是边界上点的数目,是边界的矩量，例如，二阶距用来衡量r的均值曲线分布，三阶距用来衡量以均值作为参考的对称性。

1. 矩量的优点

A、实现是直接的；

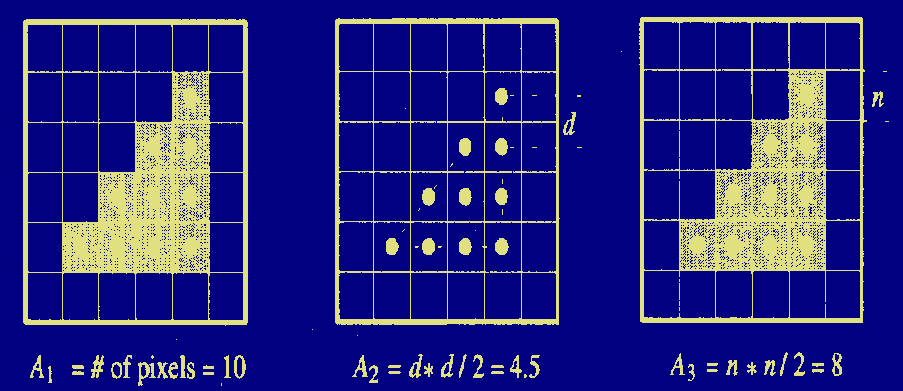
B、附带了一种关于边界形状的“物理”解释

C、对于旋转的不敏感性

D、为了使大小比例不敏感，可以通过伸缩r的范围来将大小正则化。

##### 1.2.4.3 区域描绘子

1. 一些简单的描绘子
2. 区域面积：区域中像素的数目。



1. 区域周长：区域边界的长度。
2. 致密度：(周长)2/面积 。



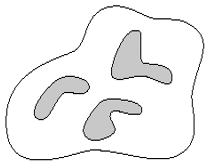
1. 其他简单描绘子

如最大值、最小值、中值、均值、重心、方差等。

2. 拓扑描绘子

拓扑学研究一种图像在没有撕裂和连接的情况下（橡皮伸展变形），不受任何变形影响的性质。

1. 拓扑描绘子由区域内连通分量数来定义



一个有3个连通分量的区域

1. 拓扑描绘子由欧拉数来定义

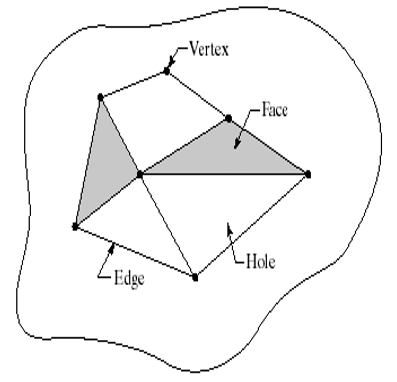
孔洞数H、连通分量的数目C、欧拉数E



1. 拓扑网络与欧拉数的关系



（V顶点数、Q边数、F面数）



V=7，Q=11，F=2，一个连通区域和3个孔

3.纹理

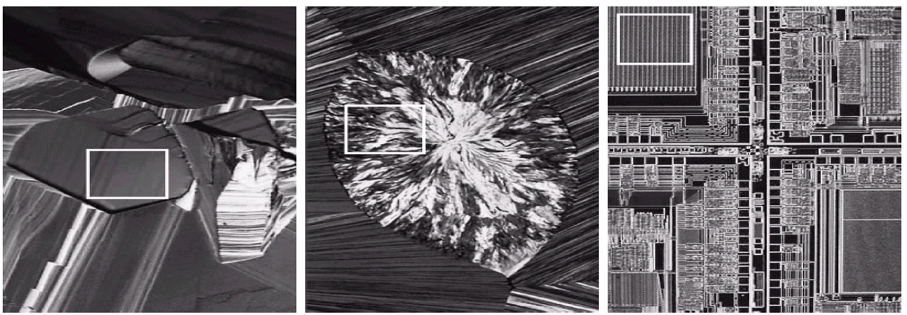
a. 区域描绘的一种重要方法是量化其纹理内容

1. 定义

纹理是由许多相互接近的、 互相编织的元素构成， 它们富有周期性。



纹理描绘子提供了对平滑度、粗糙度和规律性等特性的度量。



上图从左到右的白色方块标记平滑、粗糙且有规则的纹理。这些光学显微镜下的图像分别是一块超导体、人类的胆固醇和一个微处理器。

1. 纹理描绘子分类
2. 统计方法。（采用统计方法对纹理进行分析。）
3. 结构方法。（从图像结构的观点出发，则认为纹理是结构，由纹理基元按一定规律排列而成。 采用句法结构方法。 ）
4. 频谱法。（分析纹理的频域特征）

b.统计方法

1. 描述纹理：用一幅图像或区域灰度级直方图的统计矩

令z为一个代表 灰度级的随机变量，为对应的直方图。则n阶中心矩为：





零阶矩：

一阶矩：

二阶矩：，表示灰度对比度的量度，可以用于构造有关平滑度的描绘子：



三阶矩：，表示直方图偏斜度的量，左偏为负，右偏为正。

四阶矩：，表示相关平直度的量。

五及更高阶的矩不容易与直方图形状联系起来，但他们提供对纹理更进一步的描述。

1. 其他的基于直方图的纹理量度

一致度：

平均熵：

因为p在区间[0,1]内取值并且这些值的和为1，所以度量U对所有灰度级都相等（极大的一致性）的图像有最大值并且从那里开始递减。

熵是可变性的度量，对于一个不变的图像，其值为0.

说明：

均值在每个区域的平均灰度级并且它只作为亮度的大致概念时才是有用的，并不是真实的纹理。

标准差可以提供更为丰富的信息，使用这个度量可以清晰地显示出粗糙的纹理 ；R和标准差所衡量的事物在本质上是相同的。

三阶矩确定直方图的对称性和是否向左（负值）和向右（正值）歪斜很有用，即可以度量图像的灰度级相对于均值是偏向暗的一方还是亮的一方。

一致性：平滑的一致性好，粗糙的一致性比较低

熵值与一致性的度量是相反的。

1. 灰度共生矩阵

基本方法：取图像中任意一点(x，y)及偏离它的另一点(x+a, y+b)，设该点对的灰度值为(，)。对于整幅图像，统计出每—种(，)值出现的概率p(，)，并排列成方阵，称为联合概率矩阵，也叫做共生矩阵。再由共生矩阵计算五个统计量。

1. 具体步骤:

由原始图像生成满足位置算子P的点对矩阵；A矩阵的元素是出现（其位置由P指定）的灰度级为的点相对于灰度级为的点的次数。

  P：在右下方的一个像素

1. 计算发生的概率（点对数/图像中满足P的点对总数）



1. 计算下列统计量，以描述灰度共生矩阵的“内容”特性。

令n为图像中满足P的点对总数目。矩阵C是通过用n除A中的每一个元素得到的，则为概率估计。矩阵C为灰度级共生矩阵。

最大概率：

（表示对P的最强响应）

元素差异的k阶矩：



一致性：

（当都相等时，有最大值）

熵：

（当 c的所有元素有最大的 随机性时，有最大值）

c.结构性方法

1. 基本思想

图像中各个部分间的结构关系是二维的，而串是一维的，期望找到一种方法把二维关系转化为一维的串。

1. 通过产生规则来生成结构

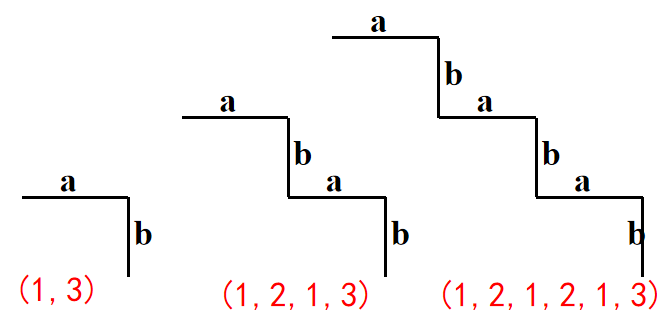
阶梯结构关系

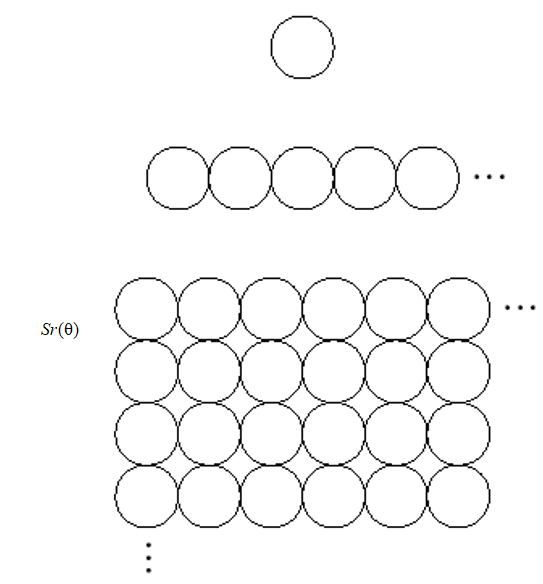
1) S->aA

2) A->bS

3) A->b

其中S、A是变量





1. 频谱方法

A、对纹理描述有用的傅里叶频谱的三个特征

1. 傅立叶频谱中突起的峰值对应纹理模式的主要方向; 
2. 这些峰在频域平面的位置对应模式的基本周期;
3. 如果利用滤波把周期性成分除去， 剩下的非周期性部分可用统计方法描述。

对于刚才提到的频谱特征的检测和解释，使用函数S（r，θ）的极坐标表达比较简单，S是频谱函数，r和θ是坐标系中的变量 。

对于两个一维函数：

对固定的θ值分析可以得到沿着自原点的辐射方向上的频谱所表现得特性（比如存在的尖峰）。反之，分析固定r值的，可得到沿着以原点为圆心的圆形上的特性。

B、频谱的整体性描述





4.二维函数的矩

1. 二维连续函数的（p+q）阶矩



单值性定力表明：如果f(x,y)是分段连续的并且仅在xy平面内有限的部分具有非零值，则存在各阶矩，并且矩的序列（）由f(x,y)唯一决定。相反，（）也唯一地决定了f(x,y)。

1. 二维连续函数的中心矩（平移不变）





1. 二维离散函数（如数字图像）的中心矩



1. 归一化中心矩（平移、尺度不变）



##### 1.2.4.4 关系描绘子

主要目的：以重写规则的形式在边界和区域中获取基本的重复模式。

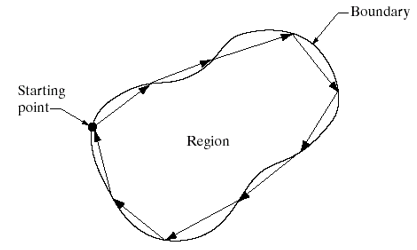
重写规则：



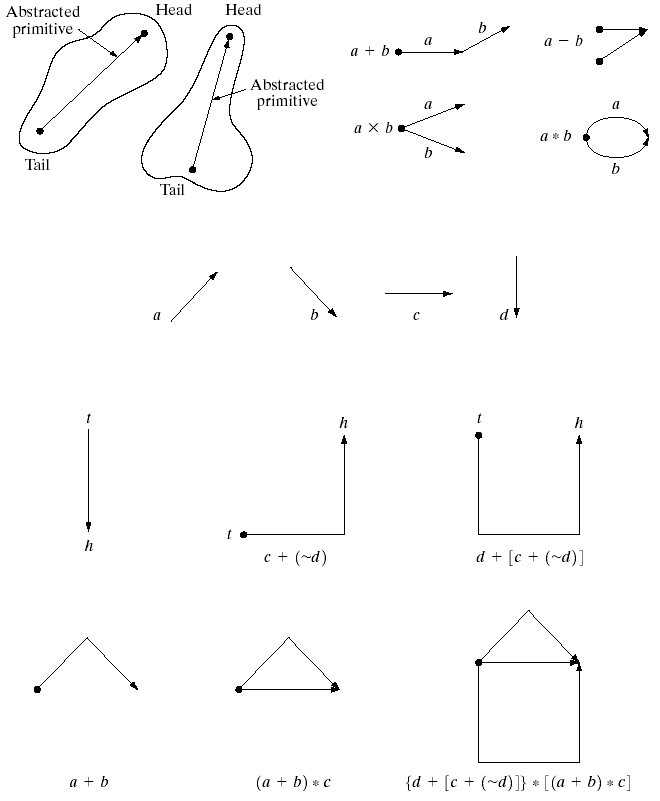


串是一维结构，用串描述图像的方法即将二维位置关系转变为一维形式。

* 一种方法是沿着对象的轮廓线用指定了方向和长度的线段进行编码。



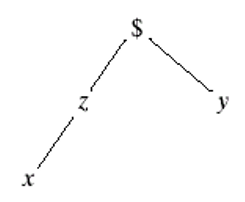
* 另一种方法是用定向线段描述一幅图像的部分区域，这种定向线段的连接方式多样。



串描述最适合应用于以首尾连接的图元连通性情况。但有时从纹理或其他描绘子看来相似的区域却是不连接的，这就需要有描述这种情况的技术。用于处理这种情况的有效方法之一就是树描绘子。

树描绘子：

一棵树T是一个或多个节点的有限集合。



树中有两类重要信息：

* 关于节点的信息
* 一个节点与它相邻节点关系的信息

总结：

从图像分割出来的对象或区域的表示方法和描绘方式是大多数图像自动处理过程的前期步骤。例如，这些描绘构成了将在下一章中展开讨论的目标识别方法的输入。从各种方法中挑选哪一种方法使由所面对的问题决定的，目的就是悬着能够“抓住”对象或对象之间本质差异的描绘子，而对于各种因素的变化，比如位置、尺寸和方向，应尽可能地保留这种差异的独立性。

#### 1.2.5 图像识别与分类

##### 1.2.5.1 图像分析

图像分析的定义：

图像分析是一个发现、辨认和理解模式的过程，这些模式都与执行与图像相关的任务有关。

图像分析的目标:

计算机图像分析的主要目的之一是，赋予某些具有感觉能力的机器，以类似人的大脑的能力。例如OCR。

自动图像分析系统行为能力的概念化分类:

* 获取、发现信息：从背景中提取相关信息
* 学习、应用知识：抽象、归纳信息特征的学习过程，并应用到新的对象中。
* 构造、推理知识：从不完整的信息中构造推论出新的知识，并加以应用。

##### 1.2.5.2 图像分析系统

图像分析技术分类的三种基本范畴：

* 低级处理：图像获取、预处理，不需要智能
* 中级处理：图像分割、表示与描述，需要智能
* 高级处理：图像识别、解释，缺少理论，为降低难度，设计的更专用。

图像识别与解释的基本方法：

* 识别的统计分类方法：用向量形式表达模式；分派模式向量到不同的 模式类。
* 识别的结构方法：用符号匹配，模式被表示为符号形式（如形状树、串和树）
* 图像解释的方法：图像解释技术是基于谓词逻辑、语义网络和特定产品的系统。

##### 1.2.5.3 模式与模式类

1.模式的定义

模式是图像中的一个对象或者某些感兴趣本质的数量或机构的描述。

模式是由一个或多个描述子来组成，换句话说，模式是一个描述子的序列（名词“特征”经常被用来代替描述子）

模式是一组特征或一组描述子。

2.模式类的定义

模式类是具有某些公共特征的模式的系列，模式类用,, 表示，m是类的个数。

3.模式识别的定义

根据图像中对象的特征组成的模式，确定对象是属于哪一个模式类，即为模式识别。

4.常用的模式序列

1. 模式向量

模式向量用粗体的小写字母表示，如x,y形式如下：



其中每一个 代表第i个描述子，n是这种描述子的数量。模式向量被表示为第一列或表示为一列或表示成

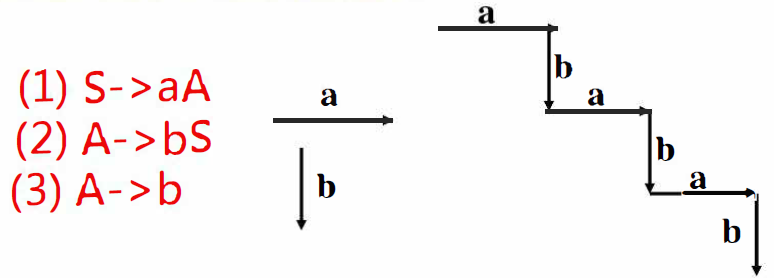


其中，T指出是秩。

1. 模式串

用千以对象特征的结构或空间关系作为模式的识别。

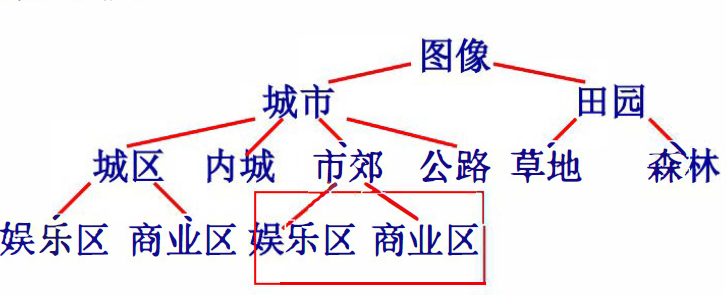
模式串举例：梯状的模式



1. 模式树

以分层目录结构排序的模式类，一般多采用树结构。

模式树举例：



##### 1.2.5.4 决策论法

1. 决策论法的基本概念

a. 决策论识别法的定义

设：模式向量：

对于：M个模式类,,，

寻找m个决策函数,,，具有这样的特性：如果模式x属于模式类，那么：



换句话说，如果一个未知模式x属于第i个模式类，把x代入所有的决策函数，的数值最大。

b. 决策边界的定义

对于模式x,如果决策函数值有：



此x值，被称为与的决策边界。

通常用一个单一的函数标识两个类之间的决策边界，定义为：



如果，x属于类，

如果，x属于类。

1. 匹配
2. 最小距离分类器
3. 算法思想

* 对于M个模式类为每一个模式类确定一个原型模式向量
* 对于一个未知模式向量x，如果x与的距离最小，就称，x属于。

1. 最小距离分类器定义：
2. 计算模式类的原形向量：



其中是属于模式类的模式向量的个数。

通过计算已知属于的模式向量的各分量的均值得到原形模式向量

1. 计算x与的距离



其中是欧几里德范式（平方与开方）

1. 决策

如果，

就说：x属于。

改成求最大的标准形式，决策函数为:



如果，

那么：x属于。

1. 类与之间的决策边界是:



1. 相关匹配
2. 相关匹配的基本思想
3. 用样板子图像直接作为模式（不是用描述子）。
4. 通过子图像与原图像直接进行相关计算，把相关计算作为决策函数。
5. 相关计算获得最大值的位置，就被认为匹配成功。

2）算法描述

决策函数是相关函数



对图像的每一个点进行相关计算，只计算重叠部分。

问题：在边界处将失去准确性，其误差与子图像的尺寸成正比。

3） 改进

相关函数对振幅的变化太敏感，f(x,y)加倍，c(s,t)也加倍。用相关系数函数代替相关函数



的值域为（-1,1）

4) 对旋转和比例变化的分析

* 当被匹配图像中，对象的尺寸和角度与模式不此方法将失效。
* 尺寸的正则化，解决空间比例的间题。 正则化模板与原图。
* 如果知道原图像的旋转角度，我们可以通过旋转原图像，对齐模式解决。
* 如果被匹配的对象的角度任意，有结论：此方法不能用于这种间题。

5) 关于空域的计算

相关函数，可以在频域计算。



但在何种情况下，没有更有优势的理论根据。相关系数方式只能在空域进行。

##### 1.2.5.5 结构法

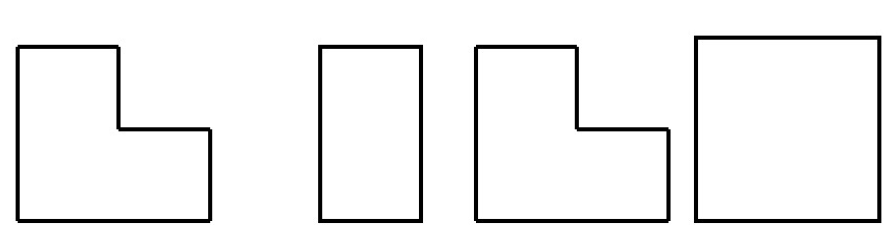
决策法，通过量化的方法处理模式，最大限度地忽略了模式形状的内在结构关系。

结构法，则力求通过准确地抓住这些不同模式类的内在结构关系来进行模式识别。

1. 匹配形状数

a. 匹配形状数的基本思想

通过比较两个对象边界的形状数的相似程度，来匹配对象。例如：



未知模式 原形模式类

b. 基本概念

1. 两个区域边界的相似级别k的定义：

相同形状数的最大序号：即：

当考虑用4向链码表示的封闭区域边界的形状数时，A和B具有相似级别k，如果满足

这里s表示形状数，下标表示序号。

1. 两个区域边界A和B形状数的距离D(A,B)

相似级别的倒数：D(A,B)=1/K

距离满足如下性质：



1. 算法思想

* 用不同密度的网格划分边界区域，获得不同序数的形状数。
* 如果使用相似级别k，k越大说明越相似。
* 如果使用相似距离D,D越小说明越相似
* 可以利用相似数来进行判别。

2. 串匹配

1. 串匹配的基本思想

比较两个边界的串编码的相似程度，来进行匹配

1. 三个基本概念

设：两个区域边界A和B已分别被编码为串。

1. 两个串的匹配数M:

当时我们说发生了一个匹配。令M代表A、B中匹配的总数。

1. 不匹配的符号数量Q:



这列的是字符串的长度。当且仅当A和B完全相同时，Q=0。

1. A和B相似度的简便衡量R:



因此，当A和B完全匹配时，;

当A和B中任何字符都不匹配时，

M=0,R=0

1. 算法思想

* 由于匹配是逐字符进行的
* 选择一个好的开始点， 可以大大减少计算量。任何将两个串规则为相同字符开头的方法都是有效的，只要这种方法不是穷举起点。
* 最大的R给出了最好的匹配。

### 1.3 相机标定

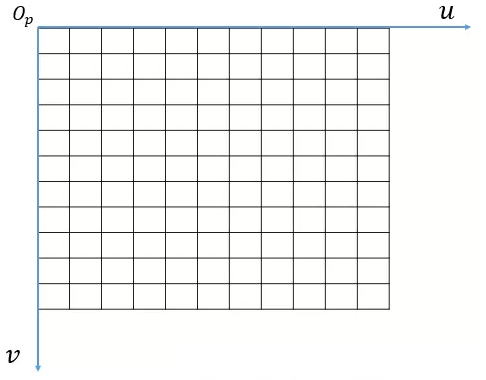
#### 1.3.1 相机标定概述

在图像测量过程以及机器视觉应用中，为确定空间物体表面某点的三维几何位置与其在图像中对应点之间的相互关系，必须建立相机成像的几何模型，这些几何模型参数就是相机参数。在大多数条件下这些参数必须通过实验与计算才能得到，这个求解参数的过程就称之为相机标定（或摄像机标定）。无论是在图像测量或者机器视觉应用中，相机参数的标定都是非常关键的环节，其标定结果的精度及算法的稳定性直接影响相机工作产生结果的准确性。因此，做好相机标定是做好后续工作的前提，提高标定精度是科研工作的重点所在。

#### 1.3.2相机标定原理

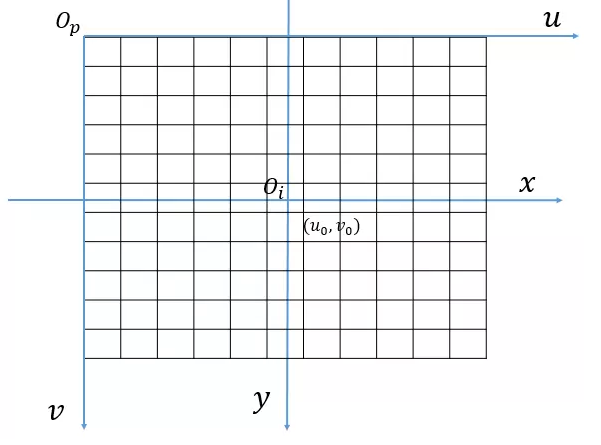
##### 1.3.2.1 各坐标系的定义

1.像素坐标系：像素坐标就是像素在图像中的位置。一般像素坐标系的左上角的顶点就是原点，水平向右是u，垂直向下是v轴。



像素坐标系

2.图像坐标系：在像素坐标系中，每个像素的坐标是用像素来表示的，然而，像素的表示方法却不能反应图像中物体的物力尺寸，因此，有必要将像素坐标转换为图像坐标。将像素坐标系的原点平移到图像的中心，就定为图像坐标系的原点，图像坐标系的x轴与像素坐标系的u轴平行，方向相同，而图像坐标系的y轴与像素坐标系的v轴平行，方向相同。



图像坐标系

在图中，假设图像中心的像素坐标是（u0,v0），相机中感光器件每个像素的物力尺寸是dx \* dy，那么，图像坐标系的坐标（x,y）与像素坐标系的坐标（u,v）之间的关系可以表示为：



写成矩阵的形式为：



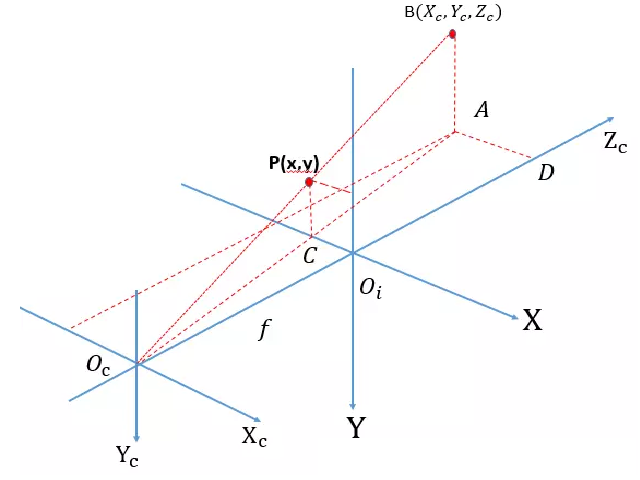
改写为齐次坐标的形式为：



3.相机坐标系：相机坐标系是以相机的光轴作为Z轴，光线在相机光学系统的中心位置就是原点Oc（实际上就是透镜的中心）,相机坐标系的水平轴Xc与垂直轴Yc分别于图像坐标系的X轴和Y轴平行。在图中，相机坐标系的原点与图像坐标系的原点之间的距离OcOi之间的距离为f(也就是焦距)。



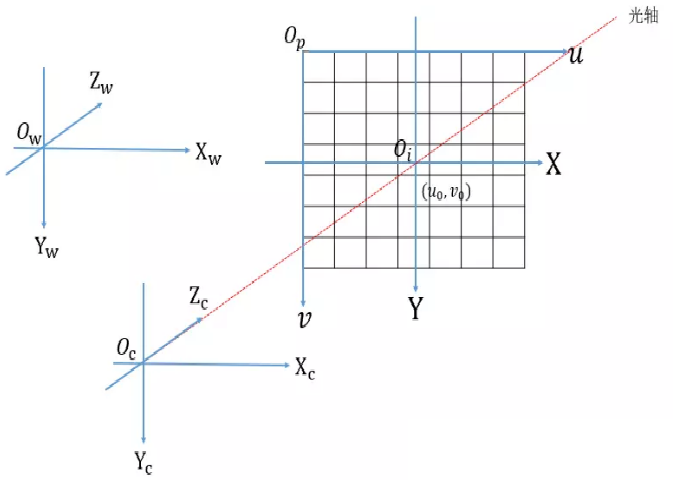
图像坐标系  
  
 上图中，如果有一个物体成像到图像坐标系，则可以用下图来表示（B点是相机坐标系中物体的点坐标，P是图像坐标系中成像的坐标）：



可以知道相机坐标系与图像坐标系的关系为：



4.世界坐标系：世界坐标系是图像与真实物体之间的一个映射关系。如果是单目视觉的话，主要就是真实物体尺寸与图像尺寸的映射关系。如果是多目视觉的话，那么就需要知道多个相机之间的关系，这个关系就需要在同一个坐标系下进行换算。在下图中，世界坐标系的原点是Ow,而Xw,Yw,Zw轴并不是与其他坐标系平行的，而是有一定的角度，并且有一定的平移。



世界坐标系

当对相机坐标系安装一定的参数，分别绕着X，Y，Z轴做平移和旋转后，就得到在世界坐标系中的坐标。 平移表示：



而对于旋转，可以采用如下的方法，给定一个基本旋转矩阵和基本矩阵：





对于三坐标轴旋转，当绕着X轴旋转时，保持基本矩阵的第1列不变，有如下的旋转矩阵：



当绕着Y轴旋转时，保持基本矩阵的第2列不变，有如下的旋转矩阵：



当绕着Z轴旋转时，保持基本矩阵的第3列不变，有如下的旋转矩阵：



那么，整个相机坐标系到世界坐标系的变换公式为：



##### 1.3.2.2 相机的内参与外参

通过前面的几个步骤，我们已经得到了各个坐标系之间的相互转换关系，进一步的就可以得到从像素坐标系到世界坐标系的变换关系：



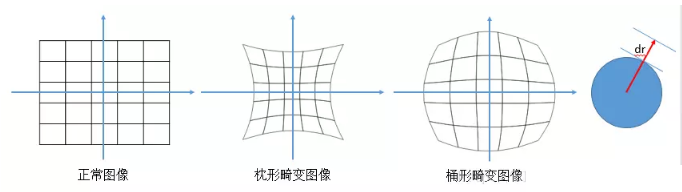
公式中，红色的框框就表示相机的外参，可以看到，外参是相机相对于世界坐标系的旋转和平移变换关系。内参是相机固有的属性，实际上就是焦距，像元尺寸。同时还可以看到，公式中有一个Zc，它表示物体离光学中心的距离。这也就说明，在标定的时候，如果物体在距离相机的不同位置，那么我们就必须在不同的位置对相机做标定。简单点来理解就是，当物体离相机远的时候，在图像上就很小，那么一个像素代表的实际尺寸就大，当物体离相机近的时候，那么成像效果就大，一个像素代表的实际物体尺寸就小。因此，对于每一个位置都需要去标定。

##### 1.3.2.3 图像畸变及畸变矫正

畸变的英文单词是distortion。从英文的意思来看就是物体看起来是不正常的，比如说形状改变了，扭曲了或者其他的变化。造成图像畸变的原因有很多，总结起来可以分为两类：径向畸变和切向畸变。

1.径向畸变

可以这样来理解，对于透镜而言，以透镜的中心作为原点，往外是透镜的半径的方向，当光线越靠近中心的位置，畸变越小，沿着半径方向远离中心的时候，畸变越大。典型的径向畸变有桶形畸变和枕形畸变。如下图所示。



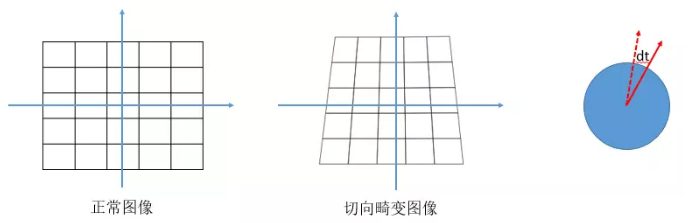
径向畸变的矫正公式如下：



式中，（x,y）是理想的无畸变的坐标（图像坐标系），（,）是畸变后图像像素点的坐标，而且r²=x²+y²。

2.切向畸变

切向畸变可以这样理解，当透镜与成像平面不平行时，就产生了畸变，类似于透视变换。



切向畸变的矫正公式如下：



3.畸变矫正

通过上面介绍的径向畸变和切向畸变模型，可以得到两个模型最终作用于真实图像后的矫正模型。



#### 1.3.3 相机标定方法

相机标定方法有：传统相机标定法、主动视觉相机标定方法、相机自标定法。具体如下：

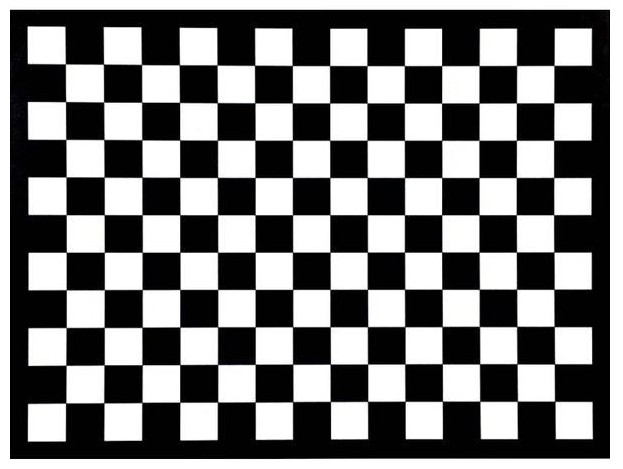
传统相机标定法需要使用尺寸已知的标定物，通过建立标定物上坐标已知的点与其图像点之间的对应，利用一定的算法获得相机模型的内外参数。根据标定物的不同可分为三维标定物和平面型标定物。三维标定物可由单幅图像进行标定，标定精度较高，但高精密三维标定物的加工和维护较困难。平面型标定物比三维标定物制作简单，精度易保证，但标定时必须采用两幅或两幅以上的图像。传统相机标定法在标定过程中始终需要标定物，且标定物的制作精度会影响标定结果。同时有些场合不适合放置标定物也限制了传统相机标定法的应用。

目前出现的自标定算法中主要是利用相机运动的约束。相机的运动约束条件太强，因此使得其在实际中并不实用。利用场景约束主要是利用场景中的一些平行或者正交的信息。其中空间平行线在相机图像平面上的交点被称为消失点，它是射影几何中一个非常重要的特征，所以很多学者研究了基于消失点的相机自标定方法。自标定方法灵活性强，可对相机进行在线定标。但由于它是基于绝对二次曲线或曲面的方法，其算法鲁棒性差。

基于主动视觉的相机标定法是指已知相机的某些运动信息对相机进行标定。该方法不需要标定物，但需要控制相机做某些特殊运动，利用这种运动的特殊性可以计算出相机内部参数。基于主动视觉的相机标定法的优点是算法简单，往往能够获得线性解，故鲁棒性较高，缺点是系统的成本高、实验设备昂贵、实验条件要求高，而且不适合于运动参数未知或无法控制的场合。

#### 1.3.4 相机标定步骤

标定模板（标定板 Calibration Target）　在机器视觉、图像测量、摄影测量、三维重建等应用中，为校正镜头畸变；确定物理尺寸和像素间的换算关系；以及确定空间物体表面某点的三维几何位置与其在图像中对应点之间的相互关系，需要建立相机成像的几何模型。通过相机拍摄带有固定间距图案阵列平板、经过标定算法的计算，可以得出相机的几何模型，从而得到高精度的测量和重建结果。而带有固定间距图案阵列的平板就是标定模板（标定板 Calibration Target）。



标定模板

相机标定步骤：

1、打印一张棋盘格，把它贴在一个平面上，作为标定物。

2、通过调整标定物或摄像机的方向，为标定物拍摄一些不同方向的照片。

3、从照片中提取棋盘格角点。

4、估算理想无畸变的情况下，五个内参和六个外参。

5、应用最小二乘法估算实际存在径向畸变下的畸变系数。

6、极大似然法，优化估计，提升估计精度。

### 1.4 机器人视觉伺服系统

#### 1.4.1 机器人视觉伺服系统概述

视觉伺服的定义：人类对于外部的信息获取大部分是通过眼睛获得的，千百年来人类一直梦想着能够制造出智能机器，这种智能机器首先具有人眼的功能，可以对外部世界进行认识和理解。人脑中有很多组织参与了视觉信息的处理，因而能够轻易的处理许多视觉问题，可是视觉认知作为一个过程，人类却知道的很少，从而造成了对智能机器的梦想一直难以实现。随着照相机技术的发展和计算机技术的出现，具有视觉功能的智能机器开始被人类制造出来，逐步形成了机器视觉学科和产业。所谓机器视觉，美国制造工程师协会机器视觉分会和美国机器人工业协会的自动化视觉分会给出的定义是：“机器视觉是通过光学的装置和非接触的传感器自动地接收和处理一个真实物体的图像，以获得所需信息或用于控制机器人运动的装置。”

机器视觉作为与人眼类似的机器仿生系统，从广义角度凡是通过光学装置获取真实物体的信息以及对相关信息的处理与执行都是机器视觉，这就包括了可见视觉以及非可见视觉，甚至包括人类视觉不能直接观察到的、物体内部信息的获取与处理等。

机器人视觉伺服系统是机器视觉和机器人控制的有机结合，是一个非线性、强耦合的复杂系统，其内容涉及图像处理、机器人运动学和动力学、控制理论等研究领域。随着摄像设备性能价格比和计算机信息处理速度的提高，以及有关理论的日益完善，视觉伺服已具备实际应用的技术条件，相关的技术问题也成为当前研究的热点。

#### 1.4.2 视觉伺服系统的机构

对机器人视觉伺服系统的研究是机器人领域中的重要内容之一，其研究成果轨线跟踪等问题中。通常所说的机器视觉是指：自动获取并分析图像，以得到一组可对景物描述的数据或控制某种动作的数据。而视觉伺服则不同于机器视觉，它利用机器视觉的原理对图像进行自动获取与分析，以实现对机器人的某项控制为目的。正是由于系统以实现某种控制为目的，所以视觉伺服系统中的图像处理过程必须快速准确。

视觉伺服系统采用视觉反馈环形成闭环，在视觉反馈环中抽取某种图像特征。图像特征可以是点、曲线、图像上的某一区域等，比如，它可以是点在图像平面的坐标位置，或投影面的形心及其惯量的高次幂。

##### 1.4.2.1 视觉伺服系统分类

视觉伺服的控制策略主要基于以下两个问题：

1）是否采用分层控制结构？即机器人是否需要闭环关节控制器？进一步说，就是系统的视觉反馈是为机器人的关节控制闭环提供输入量，还是由视觉控制器直接控制机器人各关节。

2）误差输入量是以机器人所在空间的三维坐标表示，还是以图像特征?

按控制策略2）区分，视觉伺服系统分为两类：基于位置的控制系统（position-based control，又称3D视觉伺服，3Dvisualservoing)，基于图像的控制系统（image-base control，或称2D视觉伺服，2Dvisualservoing）。由于基于位置和基于图像的视觉伺服各有其优缺点，于是近年有学者综合上述两类视觉伺服系统的优点，设计出2-1/2D视觉伺服系统。

按控制策略1）区分，视觉伺服系统可分为动态观察移动系统和直接视觉伺服。前者采用机器人关节反馈内环稳定机械臂，由图像处理模块计算出摄像机应具有的速度或位置增量，反馈至机器人关节控制器；后者则由图像处理模块直接计算机械壁各关节运动的控制量。

##### 1.4.2.2 视觉伺服系统的控制结构

1.基于位置的视觉伺服控制结构

在基于位置的控制系统中，输入量以三维笛卡尔坐标表示（又称3D伺服控制），多数基于位置的视觉伺服系统采用具有5~6个自由度的机械臂作为摄像机的运动载体。系统的视觉反馈环首先从图像中提取图像特征，然后利用图像相对位置的估计值与其期望值相比较后，产生的位置误差量送入笛卡尔坐标控制模块。根据是否采用关节控制闭环，基于位置的视觉伺服系统分为动态观察移动系统和直接视觉伺服两类。

需要指出的是，在机器人手一眼系统中，摄像头与目标的相对位置是通过机械手末端坐标与固定坐标系的关系矩阵T（T已知)间接获得。如果T存在误差，则机械手末端的位置估计也将有误差，且此误差不能被系统观察到，所以在某些情况下（如机器人抓取或跟踪物体时），系统可能会操作失败。但若使系统在检测目标的同时，也检测机械手的末端位置，则上述误差将得到修正。只能观察目标的手一眼系统称为末端开环系统，而能同时观察目标和机械手末端位置的手一眼系统称为末端闭环系统。

现在基于位置的控制系统多数为动态观察移动系统，其原因如下：

①视觉系统较低的采样速率使得对机器人的控制成为复杂的非线性动态

控制问题，而动态观察移动系统将机器人的运动学上的奇异点问题与视觉控制分开，使机器人成为理想的笛卡尔运动设备。

②多数机器人系统中含有能够接受以笛卡尔坐标表示的位置增量或速度的对外接口。

对基于位置的视觉伺服系统的具体应用有：Corke等采用固定于机械臂末端的单摄像机（已校准），对刚体的二维运动进行视觉跟踪；G.Verghese等则利用它探讨对三维运动的目标进行视觉跟踪的问题；Peter.K.Allen等基于该类系统研究出可实时跟踪并抓取移动物体的双目机器人手一眼系统；Papanikolopulos等根据此类系统研究运动物体速度未知的情况下，在二维空间内（假设深度已知）实时跟踪非几何形体运动目标的方法。

尽管基于位置的视觉伺服系统在实际应用中较为便利，但它存在如下缺点：

①依赖于摄像机及机械臂的标定精度，对标定参数误差敏感，有时还依赖于目标模型的正确性；

②对目标图像没有任何控制，意味着在跟踪过程中，目标可能逃离摄像机的视觉范围。

2.基于图像的视觉伺服控制结构

基于图像的控制系统又称2D视觉伺服。此类系统的控制策略基于当前图像特征f与理想图像特征f\*间的误差之上，因而对摄像机和机械臂的校准误差、目标模型误差具有较强的鲁棒性，正好克服了基于位置的视觉伺服系统的缺点。按控制策略2），基于图像的视觉伺服系统也有动态观察一移动系统和直接视觉伺服两类。

在基于图像的控制系统中，图像特征f通常是一些目标特征点的图像坐标集合，误差量即为（f\*一f），当e（t）=0时，跟踪达到要求。由于e（t)是在图像上（二维空间）定义，而摄像机运动控制器的输入量定义在摄像机可能运动的范围之内（三维空间），所以基于图像的控制法则必须找出表示图像特征参数变化量与摄像空间位置变化量的关系，这一关系即图像雅可比矩阵J（imageJacobian）。

例函数或复杂函数，用来调节f趋向f\*，为机器人控制器输出的摄像头运动速度，是J的伪逆阵的估计值。对基于图像的伺服视觉系统的研究很多，如蒋平等直接利用图像误差来跟踪目标，他们采用手一眼系统，首先拍摄一幅理想目标图像，而后对运动目标进行注视跟踪，使实时采样的目标图像收敛于理想目标图像，该系统的控制规则由图像差反馈和物体运动自适应补偿组成，可以完成“眼注视”这种具有局部收敛性的运动目标跟踪，且具有良好的准确性和鲁棒性。其他基于图像的控制方案如采用局部位置估计、自适应深度估计、图像雅可比矩阵估计等。

基于图像的控制系统的主要缺点为：

①计算J需要估计目标深度，而深度估计一直是计算机视觉中的难点；

②摄像机位置可能收敛于局部最小点，而非理想值；

③跟踪过程中，图像雅可比矩阵可能存在奇异值，使系统不稳定，此外，

保证系统全局稳定性的充分条件为度J[f(t)，Z(t)]>0，t此条件在实际应用中难以实现。

3.2-1/2D视觉伺服的控制结构

在总结上述两种视觉伺服系统的优缺点后，E.Malis等人提出以目标特征点的图像坐标误差Ep（以二维图像空间表示），和摄像机旋转误差Aue(以三维笛卡空间表示）作为控制系统的输入量，从而产生一种新的视觉伺服系统一2-1/2D视觉伺服系统。2-1/2D视觉伺服系统首先选取目标的特征点，根据特征点在摄像机的当前图像坐标系和理想图像坐标系（分别对应摄像机的当前位置和理想位置）中的成像点Pe和Pe\*迭代求取两图像坐标系的关系矩阵H及图像坐标误差Ep。由H求得ue(u为摄像机的旋转轴，0为摄像机绕u轴旋转的角

度），ue与已知的摄像机理想位置参数ue\*相比较，得到的摄像机旋转误差量Aue送入控制器。文献详细论述了如何由H矩阵求取摄像机部分位置参数，并指出，若目标为一平面，则求取H矩阵是一个线性问题，至少需要4对不共线的特征点，但当目标为一非平面时，求取H矩阵就成为一个非线性问题。

选择Ep和Aue作为误差的好处是：

①向量ue可控制摄像机的方向，则可表示摄像机旋转角速度Q的矩阵函数，且此矩阵函数在整个工作空间无奇异点，不仅提高了系统的稳定性，而且保证系统在整个工作空间向理想位置收敛，从而使得在跟踪过程中，不论摄像机的初始位置如何，目标始终保持在摄像机的视觉范围内；

②Ep是二维图像坐标误差，它作为控制系统的输入量可以保证系统在其校准误差下的全局稳定性。

2-1/2D视觉伺服系统也存在一些缺点：

①需要进行特征点匹配；

②求解H矩阵是一个计算量很大的迭代过程；

③比2D视觉伺服系统更易受图像噪声的影响。

#### 1.4.3 图像处理

##### 1.4.3.1 图像特征

图像特征主要有图像的颜色特征、纹理特征、形状特征和空间关系特征。

颜色特征是一种全局特征，描述了图像或图像区域所对应的景物的表面性质；纹理特征也是一种全局特征，它也描述了图像或图像区域所对应景物的表面性质；形状特征有两类表示方法，一类是轮廓特征，另一类是区域特征，图像的轮廓特征主要针对物体的外边界，而图像的区域特征则关系到整个形状区域；空间关系特征，是指图像中分割出来的多个目标之间的相互的空间位置或相对方向关系，这些关系也可分为连接/邻接关系、交叠/重叠关系和包含/包容关系等。

##### 1.4.3.2 计算目标的空间位置

视觉测量技术是建立在机器视觉研究基础上的一门新兴技术, 是根据摄像机获得的视觉信息对目标的位置和姿态进行的测量 , 主要研究从二维图像或三维笛卡尔空间信息的映射和视觉测量系统的构成等 。视觉测量具有结构简单、便于移动、数据采集迅速等优点。视觉测量按所用视觉传感器数量可以分为单目视觉测量、双目视觉( 立体视觉)测量和三( 多) 目视觉测量等。其中, 双目视觉测量和近景摄影测量的理论基础和主要研究内容是一样的。单目视觉测量是指仅利用一台相机或摄像机拍摄单张像片来进行测量工作。因其仅需一台视觉传感器, 所以, 该方法的优点是操作方便、结构简单、相机标定也简单, 同时还避

免了立体视觉中的视场小、立体匹配难的不足 , 因而, 近年来这方面的研究比较活跃, 主要集中在对运动物体的检测与跟踪、三维重建等。

但是, 目前针对工业领域内空间目标位置测量的单目视觉测量技术的研究还比较少, 本文在结合科研项目的基础上, 利用单目视觉采集图像, 通过图像处理技术, 结合直线与平面相交法来对工业生产过程中的作业平台上的目标进行定位, 达到了较高的精确性, 满足了工业生产要求。

1.单目摄像机设定

摄像机标定是三维机器视觉的必要步骤, 用以从摄像机获取的图像信息中得到三维物体的位置、形状等几何信息。在许多情况下, 机器视觉的整体性能主要依靠摄像头标定的精度。

摄像机标定就是获得摄像机内部几何和光学特性即内部参数, 以及摄像机在大地坐标系中的位置和方向, 即外部参数。建立适当的数学模型, 得到二维计算机图像坐标与三维大地坐标之间的关系, 然后计算摄像机内外部参数的精确值。

1.1摄像机模型

利用摄像机针孔模型来描述三维空间的点到计算机图像帧缓冲区像素点的转换关系。摄像机针孔模型是基于共线原理的, 空间中物体的每一个点通过投影中心以直线的方式投射到图像平面。针孔型摄像机模型如图。

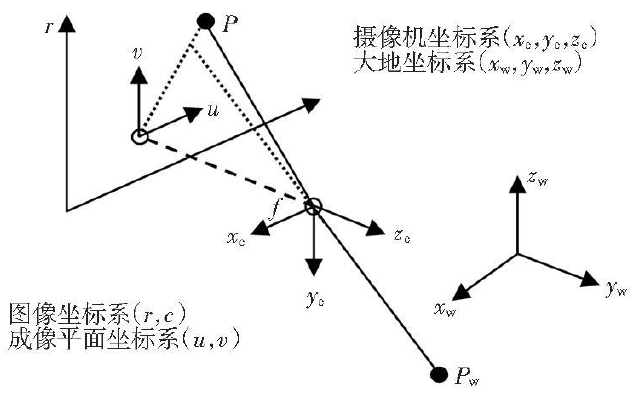


图 针孔型摄像机模型

设(, ,  )是大地坐标中某目标点P 的三维坐标,(, ,  ) 是同一点P 在摄像机坐标系中的三维标, ( ,)是在理想针孔模型下P 点的图像坐标, ( ,)由透镜变形引起的偏离( ,) 的实际图像坐标。( u, v ) 是P 点在成像平面坐标系下的坐标。从(, ,  ) 到 ( ,) 的完整变换可以分为4步 :

1. 三维空间刚体位置变换



式中R 为3 \*3 的旋转矩阵,T 为3\*1的平移矩阵





1. 针孔模型下的理想投影变换( 透视变换)



1. 畸变模型



式中 k 为畸变系数。

1. 成像平面坐标系到图像坐标系的变换公式为



式中 ( , )为计算机图像中心坐标, 也叫做主点坐标。

1.2参数标定

需要标定的外参数为旋转阵R, 平移向量T; 内参数=fx, fy , ( , ), k, 其中, fx = f /dx, fy = f /dy。

利用OPENCV 开发的摄像机标定程序来进行参数标定。标定采用的是一张7 􀀁 7的平面棋盘格标定板, 如图2所示, 摄像机只需在不同的角度抓取几张平面标定模板的图片, 就可以实现对摄像机的标定。显然, 由于本程序的原理是采用最小二乘法, 所以, 抓得图越多, 标定的结果就越精确。

程序的基本步骤如下:

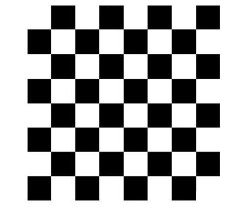


图2 平面棋盘格标定板

1)读取一组(多余10张)图像数据;

2)用cv F ind Chessboa rd Corners( )筛选图像; 将读入的一组图像数据分别代入cv Find Chessboa rd Corne rs( )函数,如果返回值是1, 则表示在该幅图像上提取的角点数目和设定的相同, 提取角点成功; 若为0, 则表示角点提取失败,该幅图要舍弃;

3)将用cv F ind Chessboard Corne rs( )得到的图像像素坐标系中坐标值代入Find Corne r SubP ix( ) 函数, 进一步精确得到角点亚像素级的坐标值;

4)将角点在大地坐标系的坐标值和在图像坐标系中的坐标值代入cv Ca librate C am era 2( )函数, 最后得到摄像机的内外部参数值。

2.空间目标位置测量

已知空间平面􀀁, 空间直线L, 当直线与平面的位置的关系是相交但并非重合时, 则该直线与平面只存在一个交点, 即图3所示的P 点。

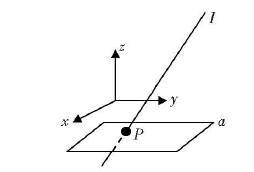


图3 空间直线与平面相交示意图

如果直线经过2个已知点P1 (, ， ) , P 2 (, ， ), 则用参数方程表示直线L 的方程如下：



平面α的方程可用下式表示



则交点P 的值为





现在, 将这种关系引入对空间目标的位置测量中, 根据图1, 空间中的任意一点都是通过投影中心在成像平面上成像的, 那么, 这两者之间的连线就是所说的光线, 可以通过确定这条光线上的2个点来写出这条光线的方程, 然后跟所定义的大地坐标系中的平面相交, 则可以得到空间目标的坐标点。

由图1可以看出: 所要找的第一个点是摄像机的投影中心, 即摄像机坐标系中的坐标原点[ 0, 0, 0 ]^ T。第二点则是要求空间位置点的对应点, 由于在实际的处理中, 得到的是图像坐标系里的坐标((, ) , 所以, 先要将其从图像坐标系转换到成像平面坐标系中, 考虑到要消除畸变的影响,可以采用等式( 3) 和式( 4) 来完成这个转换。为了得到该点在成像平面上对应的空间点坐标, 还需要考虑其在摄像机坐标系中的z 方向的值, 成像平面与光心的距离为f, 因此, 第二个点的坐标为[ u, v, f ]^T。根据等式( 5)得到摄像机坐标系中的这条线如下



由于测量的是目标的大地坐标, 则需要将直线L 从摄像机坐标系变换到大地坐标系, 实际上就是把构成直线的2个点 和 变换到大地坐标系下。根据式( 1)得



等式中, 是等式( 1 )中R 的逆矩阵。则转换后的2个点分别为 = T= -  T,  = (  - T )，于是直线L 可以重新表示为



在实际的测量中, 假定目标所放置的平面的方程为z=0, 将平面方程代入式( 10), 即可得到目标的空间位置。由于使用的平面的标定板, 所以, 测量平面的位姿可以通过标定得到, 即如果将平面标定板直接放置在测量平面上, 那么, 此时标定得到的关于这幅图像的外参基本上就等同于测量平面的位姿。另外还必须考虑的是, 因为标定板存在一定的厚度, 所以, 外参所确定的大地坐标系还必须沿其Z 轴正方形再移动标定板的厚度, 则式( 9) 中从大地坐标到摄像机坐标转换的变化: T变为RD + T, 其中, D = 。

##### 1.4.3.3 构造图像雅克比矩阵

基于图像雅可比矩阵的方法是目前在智能机器人视觉伺服研究领域使用最为广泛的一类方法。假设f∈Rm为图像特征参数矢量（f为在图像特征空间中相应的速度量)，p∈Rm为机器人末端执行器在任务空间中的坐标（p为任务空间中相应的速度量),则图像雅可比矩阵为从机器人任务空间到图像特征空间的一个映射变换，如式（1）所述：



此模型首先由Weiss 提出,即为图像雅可比矩阵,也可称之为特征敏感阵、变换阵、B矩阵、Compositc Jacobian 及visua-motor Jacobian等。它反映了机器人运动空间与所选择的图像特征空间的微分映射关系。

假定一个 6DOF 机器人手眼系统，P为固定在此末端执行器上一点,为该点在摄像机坐标系中的坐标. 根据图 1 所示小孔成像模型 ,该点在摄像机图像平面上的坐标[u, v]T为



式中*:λ*为摄像机焦距*.，*

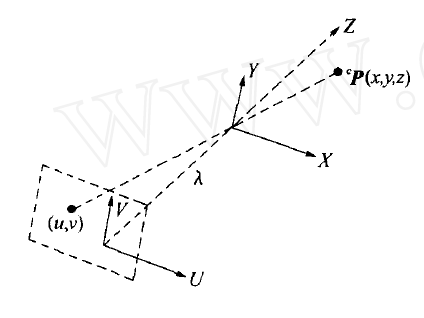


图 小孔成像模型

而机器人末端执行器相对于一个固定的摄像机坐标系中作平移运动和旋转运动根据机器人末端执行器的运动,可以得到如下关系:



将代入式（4）可得到c P 的微分：



由式（3）可得：



经过整理*,*最后可以得到如下等式：



定义系统图像特征为将机器人空间运动p定义(即将机器人看作理想的笛卡儿运动器件,不考虑关节运动控制) ,可以得到系统的图像雅可比矩阵如下



从式（9）可以看出,图像雅可比矩阵的值与机器人末端执行器相对于摄像机坐标系的当前位置及摄像机内部参数有关,并随执行器位置的变化而变化.所以,有必要对图像雅可比矩阵进行进一步处理。

##### 1.4.3.4 滤波与预测

对于基于位置控制的视觉伺服系统，从二维图像平面提取得到三维的目标位置，必然是带有噪声的；所以系统必须对视觉处理后的信号进行数字滤波；另一方面，视觉采样的速率一般较慢（典型的采样频率是30Hz)，而机器人的伺服驱动频率一般为1000Hz，因此视觉处理造成的时延不可避免；同时，运动目标的位置在时刻变化，因此，对运动目标操作的基于位置控制系统中应有预测器以补偿时延.在选择预测滤波器时，主要的约束是对于运动特性及噪声特性的了解，同时考虑计算的成本.一般来说，对运动目标拦截、抓取，打击等操作，希望根据已获得的离散时间序列目标位置信息，分析其轨迹、速度加速度等运动参数，以规划机器人在何时、何地接近目标并实施操作。

对于运动方程易于建模的情况，如目标保持匀速直线运动或抛物运动，可直接求解运动方程参数来实现预测.比如PUMA560机器人拦截并抓取平面直线运动的乒乓球时，对视觉图像处理得到的目标的位置信息，可以采用二阶滤波器进行数字滤波，然后建立直线运动方程对目标实现运动估计。

对于运动目标运动方程不易确定的情况，一般，操作分两步进行，首先跟踪目标，在末端执行器接近目标且达到一定精度时，再实施抓取等操作，这样策略下多采用两种预测方法：一种是建立自回归模型ARM，对位置信息的时间序列进行分析；另一种是采用Kalman滤波器，对运动参数估计。

Kalman滤波器是在动态系统中定义一线性状态空间模型，然后用输出公式得到系统的最优的状态估计.获得好的预测精度的条件是干扰和测量噪声都是零均值白噪声.在对机器人视觉图像预测时，大多将目标位姿及其导数作为状态向量.由于目标的动态特性不可知，因此选用常速度模型，即在一个采样周期内速度不变，但速度又是状态向量的一部分，它的值也被估计，每次被更新。

Kalman滤波器的输出要求是线性的，而视觉处理的输出是目标在图像平面的坐标，它与状态向量(目标在机器人坐标系中位姿速度)的关系是非线性的。Kalmmn预测器对速度的变化也比较敏感，更适宜运动目标速度变化缓慢（如匀速）的情况。

为使视觉伺服机器人能对速度不连续或运动轨迹变化大的场合，可以建立两个基本的运动模型：直线运动、圆周运动.Kalman滤波器也基于此模型建立，通过设置合适的阀值误差，在基本模型间进行转换而实现精度更高的运动估计.这种方法应用到长周期的运动估计中，能明显地提高精度。

#### 1.4.4 视觉控制器设计

视觉控制器是机器人视觉伺服系统控制的核心，它根据位置误差或图像误差，采用一定的控制算法，实现对机器人的运动控制.依据机器人和视觉系统的性能及任务的复杂性，可以采用不同的设计方法。

##### 1.4.4.1 PID控制法

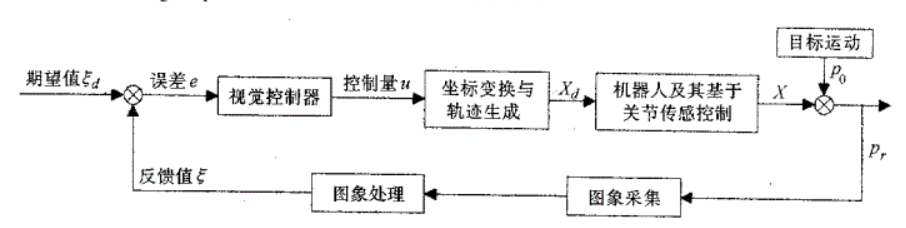


图 基于位置的动态look-and-move视觉伺服系统结构

选择PID控制器是最经典的方法。在图中，误差可以在工作空间、关节空间或图像特征空间(基于图像控制结构）表示，而机器人的控制为直角空间或机器人关节空间的位置或速度运动指令，因此，在进行PID控制之前，有必要进行空间转换或机器人逆运动学计算，需用到机器人微分运动雅可比矩阵或图像雅可比矩阵。因此，将机器人当成一被控对象，输入为直角空间或机器人关节空间的位置或速度运动指令，不考虑其动态特性，则依据视觉反馈误差e可建立控制律：



式中，u为控制量输入，，分别为比例、积分、微分的三个系数矩阵.多数情况下，直接根据误差e，进行坐标变换和轨迹规划后送入机器人控制器，这相当于，取单位矩阵的纯比例控制.

由于e，u可以在图像特征空间、工作空间、关节空间中表示，而机器人的控制量输入通常在工作空间或关节空间中表示，所以需要用到必要的空间变换关系，即机器人本身的微分雅可比矩阵Jmbot以及图像雅可比矩阵Jimage。Jmbot通常可以很好地求取，而Jimage中的每一元素都是工作空间位姿向量r的非线性函数，在任务执行过程中随着r的变化，需要在线实时调整.计算Jimage要用到目标的深度信息，这通常难以求取，所以有很多系统采用离线计算好的期望J\*image，并在运行中使用恒定的J\*image。计算Jimage可以依据物体的已知CAD模型，采用类似标定的方法获取深度信息或者利用测矩传感器或立体视觉的方法：也可以离线用神经网络来学习图像特征变化量与关节变化量之间的非线性匹配关系或在线自适应调整。

##### 1.4.4.2 基于图像差的视觉控制器

目前，几乎所有的视觉伺服系统都需要从图像中抽取特征来获取反馈信息，视觉仍只作为闭环控制中图像特征的一个映射工具而孤立于环外.实际的控制是以这一映射量作为控制目标，因此控制精度直接依赖于特征抽取的精度。基于Vidyasagar教授的视觉与控制集成的思想，即跳过图像特征抽取过程直接利用目标图像与实际图像间的反馈引导机器人运动，而提出了基于图像差的图像直接反馈方法(见图)，其闭环系统的参考输入及被控量均为二维图像。

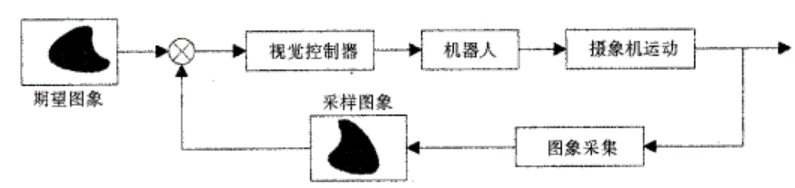


图 基于图像差的图像直接反馈控制

定义期望图像与实时采集的图像之间的图像误差为：



基于这一二维图像误差，不同的任务选用不同的李雅普诺夫函数，按照李雅普诺夫法设计不同控制律。

##### 1.4.4.3 考虑动态特征的自适应视觉控制器

为完成对运动目标的操作，传统的方法是对目标的三维位置重建，或是对目标图像分析，以控制机器人运动接近目标实施操作.这类方法不可避免地需要一些运动学参数或摄像机焦距等参数，而我们知道，人类完成此类的操作，并不需要位置重建或其它信息，基于此，提出了视觉伺服机器人自适应控制器，以克服模型参数的不确定及摄像机标定误差等引起的问题。

另一方面，由于机器人系统的非线性及动态特性，在执行高速度等任务时，难以达到满意的控制结果因此，应建立精确的动力学模型，研究其动力学特性对系统的影响。

自适应视觉控制器正是考虑了模型参数不确定性及系统动态性能两方面的因素而设计的，它通常是结合机器人的动力学方程直接设计机器人的关节力矩控制律。

Liu Hsu建立固定摄像机的平面机器人系统，研究其视觉伺服的自适应控制方法，首先建立参数化的动力学方程，同时根据摄像的投影关系，得到目标的特征点的变化与其关节速度的关系，这个关系中隐含着不确定参数：焦距、摄像机与操作平面的距离摄像机坐标系与机器人坐标系的夹角等，最终建立模型参考自适应控制器，并通过递阶控制或非线性PI控制求解，使得参考模型与系统模型的误差趋于零。

此外，还有视觉阻抗控制，任务函数法，基于状态空间描述的视觉控制器设计等方法，在此不再过多介绍。

#### 1.4.5 存在的问题

由于机器人视觉包括许多自成一体的研究领域，如图像处理、计算机视觉、运动学、动力学、控制理论及实时计算等，所以要考虑的问题比较多。其中的视觉部分，与计算机视觉中的active vision，shape from stereo vision, shape from motion 等有相似之处。但计算机视觉与机器人视觉研究的最终目的不同：前者主要研究视觉检验，精度要求高，速度不是主要问题；而机器人视觉主要研究在视觉引导下机器人对环境的作用，有实时性的要求。根据目前情况，机器人视觉应在以下几个方面进一步加强研究：

1)图像特征的选择问题。视觉伺服的性能密切依赖于所用的图像特征，特征的选择不仅要考虑识别的指标，还要考虑控制指标。从控制的观点看，用冗余特征可抑制噪声的影响，提高视觉伺服的性能，但又会给图像处理增加难度。因此如何选择性能最优的特征，如何处理特征以及如何评价特征，都是需要进一步研究的问题。针对任务有时可能需要从一套特征切换到另一套，可以考虑把全局特征与局部特征结合起来。

2)结合计算机视觉及图像处理的研究成果，建立机器人视觉系统的专用软件库。在视觉伺服中，需要进行图像采集、图像处理、特征抽取及由二维信息

重构三维信息等，要处理的数据量较大，算法复杂多样。如果有这样的软件平台，在进行视觉伺服任务时，就可以少走弯路.当然更希望生产出性能价格比较高的相关硬件。

3）加强系统的动态性能研究.目前的研究多集中于根据图像信息确定期望的机器人运动这一环节上，而对整个视觉伺服系统的动态性能缺乏研究。

4）利用智能技术的成果。虽然神经网络在机器人视觉伺服中已得到应用，但多数都是针对具体物体的具体特征，或只进行了仿真实验，还有待于进一步的研究.考虑到人类看到并拿起某个物体时，事先并没有在数字上准确计算物体的位置，而是通过不断地观察、判断和推理，其中包含学习和模糊推理的内容，由此可以考虑用模糊神经技术解决机器人视觉伺服问题。

5）利用主动视觉的成果。主动视觉是当今计算机视觉和机器视觉研究领域中的一个热门课题。它强调的是视觉系统与其所处环境之间的交互作用能力，因而有可能使被动感知方式下一些病态问题转化为良态问题，使非线性问题转化为线性问题。

6）多传感器融合问题。视觉传感器具有一定的使用范围，如能有效地结合其它传感器，利用它们之间性能互补的优势，便可以消除不确定性，取得更加可靠、准确的结果。

基于图像的视觉伺服方法虽然已取得了瞩目成果，但其自身的缺陷限制

了其应用范围，其主要缺点有：

图像雅可比矩阵通常与机械手到图像平面的垂直距离（也称为深度)有关，而深度信息估计一直是计算机视觉中的难点。对于摄像机固定的场合，深度可通过机器人运动学及摄像机标定参数来求取，而对于眼在手结构，深度的精确求取比较困难。

图像雅克比矩阵一般只能在局部范围内才能保证其精度，因此摄像机位置可能收敛于局部最小点，而非理想值。

跟踪过程中图像雅克比矩阵可能存在奇异性问题。当机械手位于任务空间的某些位姿时，图像雅可比矩阵可能会降秩，导致利用逆雅可比矩阵求取的控制律不稳定。

在摄像机运动过程中，需要在线估计图像雅可比矩阵且估计精度难以提高。此外，保证系统全局稳定性的充分条件是J\*J>0，J广为J的伪逆，此条件在实际中难以保证。

## 2. ROS机器视觉实践

### 2.1 驱动USB摄像头

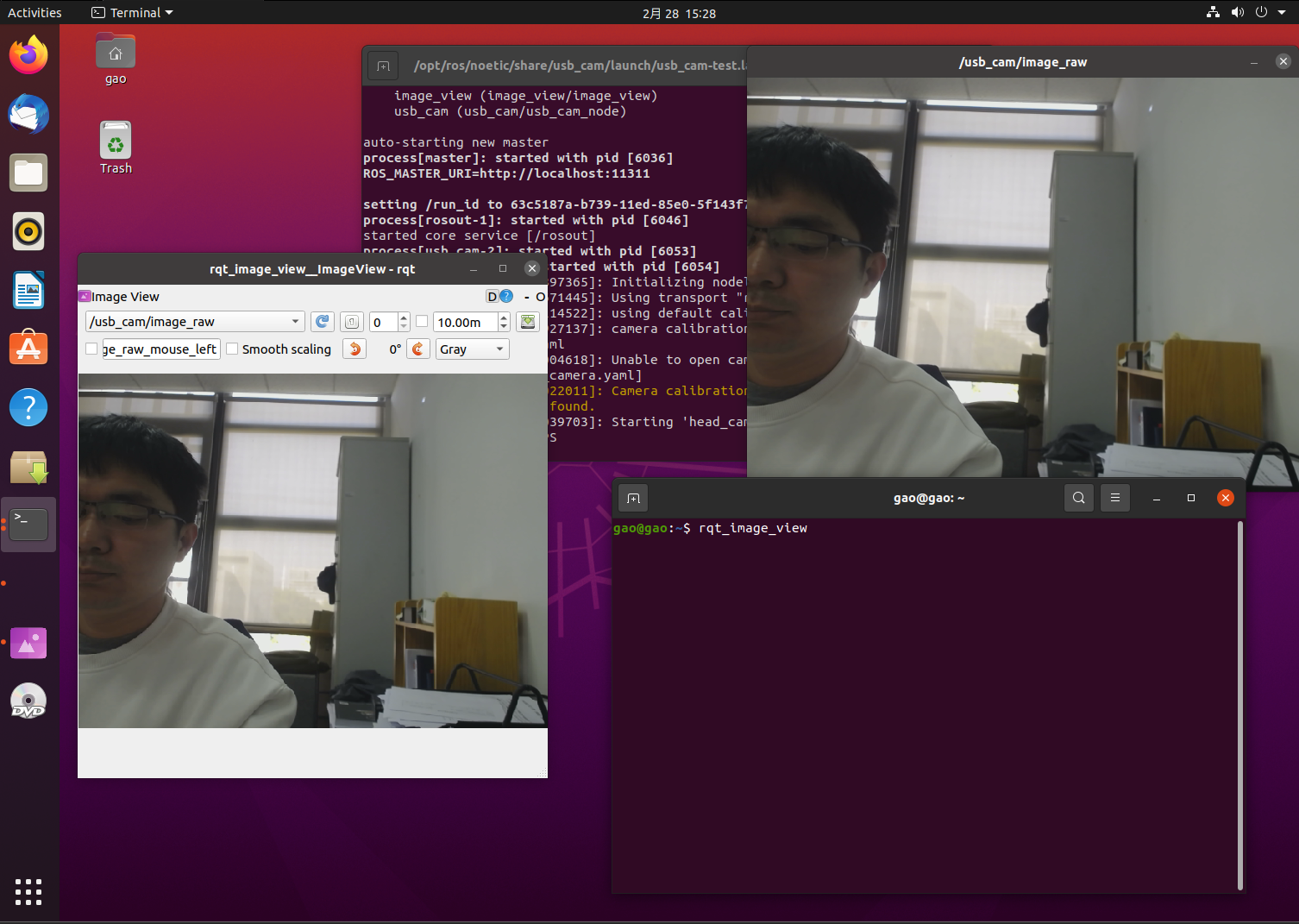
ROS中使用USB摄像头非常容易，可以直接使用usb\_cam功能包驱动。

sudo apt-get install ros-noetic-usb-cam #安装usb\_cam功能包

安装完成之后就可以用以下命令启动摄像头：

roslaunch usb\_cam usb\_cam-test.launch

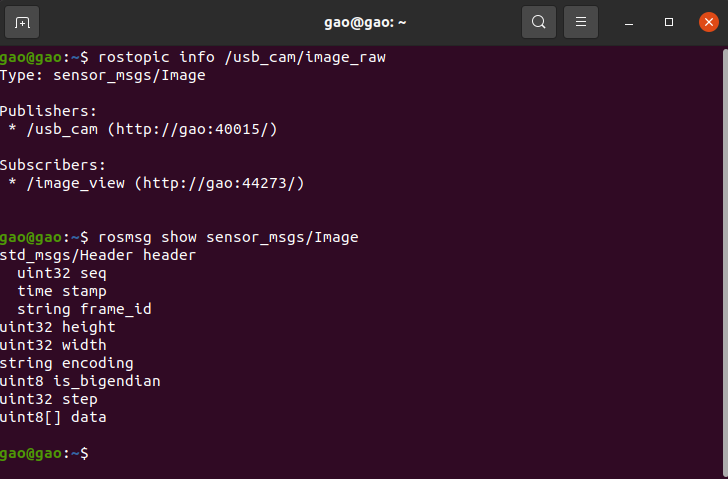
rqt\_image\_view



查看话题信息和图像消息

rostopic info /usb\_cam/image\_raw

rosmsg show sensor\_msgs/Image



图像数据具体如下

header：消息头

height(width)：图像纵向(横向)分辨率

encoding：图像的编码格式

is\_bigendian：图像数据的大小端存储模式

### 2.2 摄像头标定

由于摄像头内部以及外部的一些原因，生成的物体图像通常会发生畸变，为了避免数据源造成的误差，需要对摄像头参数进行标定。(此处需要制作一个标定靶)

sudo apt-get install ros-noetic-camera-calibration #安装摄像头标定功能包

roslaunch robot\_vision usb\_cam.launch

rosrun camera\_calibration cameracalibrator.py --size 8x6 --square 0.024 image:=/usb\_cam/image\_raw camera:=/usb\_cam

其中第三条命令中cameracalibrator.py标定需要以下几个参数：

size：标定棋盘的内部的角点个数

square：这个参数对应每个棋盘格的边长，单位为m

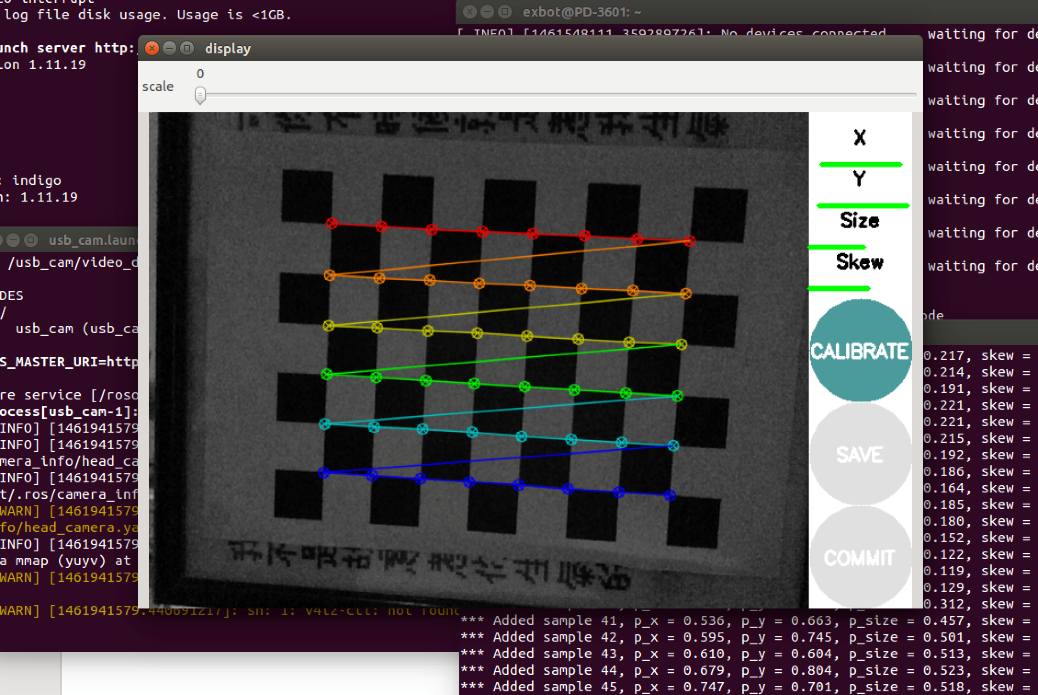
image和camera：设置摄像头发布的图像话题

上面的命令启动成功之后，将制作好的标定靶放置在视野范围内，就可以看到下图所示的界面。

界面正常出现后，我们需要手持棋盘板在不同的距离不同的角度位置让程序识别棋盘，所以需要确保棋盘全部保持出现在图像中。大致的几个位置如下图（转自官网）显示：



在每个不同位置保持片刻，等棋盘被彩色高亮标示后即可移动。在移动过程中可以看到窗口右上角的几个进度条在增长，同时颜色也渐渐趋向绿色。当右侧的calibration按钮亮起之后，表示我们已经采集够了标定所需数据，点击按钮便会自动计算并显示结果。



### 2.3 OpenCV及人脸识别

基于OpenCV库，可以快速的开发机器视觉方面的应用，而ROS中集成了OpenCV库的相关接口功能包，使用下面的命令即可安装：

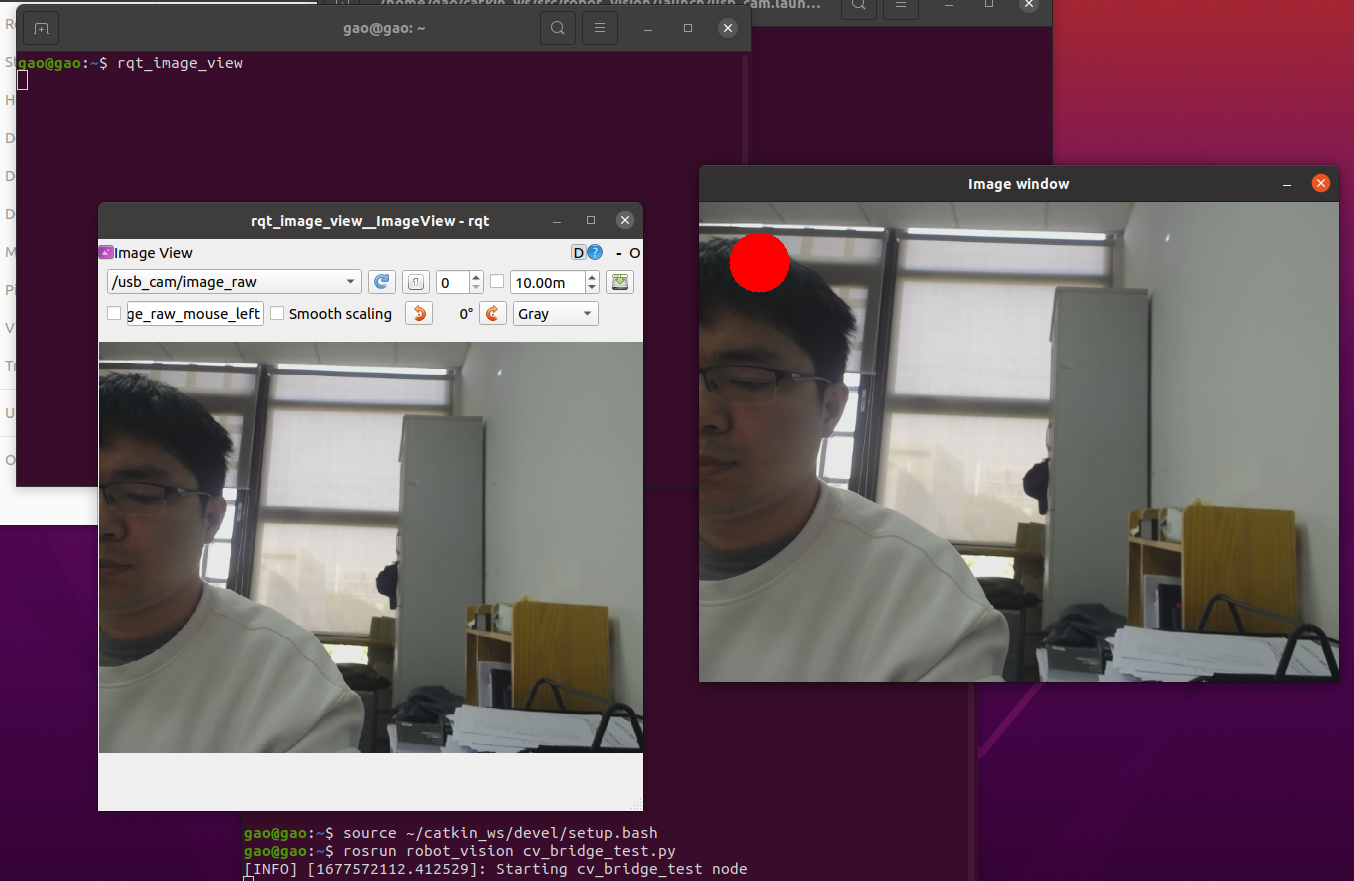
sudo apt-get install ros-melodic-vision-opencv libopencv-dev python-opencv

通过以下命令启动该例程（此处需要将py文件的权限改为可执行文件）

roslaunch robot\_vision usb\_cam.launch

rosrun robot\_vision cv\_bridge\_test.py

rqt\_image\_view



实现上述功能的源码robot\_vision/scripts/cv\_bridge\_test.py的步骤中首先是导入相应的模块，然后就是定义一个subscriber接收原始的图像消息，再定义一个Publisher发布OpenCV处理之后的图像消息。最后就是一些ROS图像消息与OpenCV图像数据之间的转换。

import cv2

from cv\_bridge import CvBridge,CvBridgeError #导入相应模块

定义一个Subscriber接受原始图像消息，再定义一个Publisher发布OpenCV处理后的图像消息，还需要一个CvBridge的句柄，便于调用相关转换接口。

self.image\_pub = rospy.Publisher("cv\_bridge\_image",Image,queue\_size = 1)

self.bridge = CvBridge()

self.image\_sub = rospy.Subscriber("/usb\_cam/image\_raw",Image,self.callback)

在这个例程的源码中主要的是imgmsg\_to\_cv2()、cv2\_to\_imgmsg()这两个接口函数的使用方法。

此处来进行ROS和OpenCV的结合来实现人脸识别的机器视觉应用。OpenCV中的人脸识别算法首先将获取的图像进行灰度转化、边缘处理和噪声过滤；然后缩小图像、直方图均衡化，匹配分类器放大相同倍数直到大于检测图像，则返回匹配结果。

使用下面命令来启动摄像头，然后运行face\_detector.launch启动人脸识别功能：

roslaunch robot\_vision usb\_cam.launch

roslaunch robot\_vision face\_detector.launch

启动完成之后就可以在摄像头中看到相应的效果，设别到人脸的区域使用绿色矩形框标识。

下面为face\_detector.py的源码主要分为三部分。

1、初始化部分

主要完成ROS节点、图像、识别参数的设置。

def \_\_init\_\_(self):

rospy.on\_shutdown(self.cleanup);

# 创建cv\_bridge

self.bridge = CvBridge()

self.image\_pub = rospy.Publisher("cv\_bridge\_image", Image, queue\_size=1)

# 获取haar特征的级联表的XML文件，文件路径在launch文件中传入

cascade\_1 = rospy.get\_param("~cascade\_1", "")

cascade\_2 = rospy.get\_param("~cascade\_2", "")

# 使用级联表初始化haar特征检测器

self.cascade\_1 = cv2.CascadeClassifier(cascade\_1)

self.cascade\_2 = cv2.CascadeClassifier(cascade\_2)

# 设置级联表的参数，优化人脸识别，可以在launch文件中重新配置

self.haar\_scaleFactor = rospy.get\_param("~haar\_scaleFactor", 1.2)

self.haar\_minNeighbors = rospy.get\_param("~haar\_minNeighbors", 2)

self.haar\_minSize = rospy.get\_param("~haar\_minSize", 40)

self.haar\_maxSize = rospy.get\_param("~haar\_maxSize", 60)

self.color = (50, 255, 50)

# 初始化订阅rgb格式图像数据的订阅者，此处图像topic的话题名可以在launch文件中重映射

self.image\_sub = rospy.Subscriber("input\_rgb\_image", Image, self.image\_callback, queue\_size=1)

2、ROS图像回调函数

当接收到摄像头发布的RGB图像数据之后进入回调函数，将图像转换成OpenCV的数据格式，然后处理之后调用人脸识别的功能函数，最后发布结果。

def image\_callback(self, data):

# 使用cv\_bridge将ROS的图像数据转换成OpenCV的图像格式

try:

cv\_image = self.bridge.imgmsg\_to\_cv2(data, "bgr8")

frame = np.array(cv\_image, dtype=np.uint8)

except CvBridgeError, e:

print e

# 创建灰度图像

grey\_image = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

# 创建平衡直方图，减少光线影响

grey\_image = cv2.equalizeHist(grey\_image)

# 尝试检测人脸

faces\_result = self.detect\_face(grey\_image)

# 在opencv的窗口中框出所有人脸区域

if len(faces\_result)>0:

for face in faces\_result:

x, y, w, h = face

cv2.rectangle(cv\_image, (x, y), (x+w, y+h), self.color, 2)

# 将识别后的图像转换成ROS消息并发布

self.image\_pub.publish(self.bridge.cv2\_to\_imgmsg(cv\_image, "bgr8"))

3、人脸识别

人脸识别部分直接调用了OpenCV提供的人脸识别接口，与库中的人脸特征进行匹配。

def detect\_face(self, input\_image):

# 首先匹配正面人脸的模型

if self.cascade\_1:

faces = self.cascade\_1.detectMultiScale(input\_image,

self.haar\_scaleFactor,

self.haar\_minNeighbors,

cv2.CASCADE\_SCALE\_IMAGE,

(self.haar\_minSize, self.haar\_maxSize))

# 如果正面人脸匹配失败，那么就尝试匹配侧面人脸的模型

if len(faces) == 0 and self.cascade\_2:

faces = self.cascade\_2.detectMultiScale(input\_image,

self.haar\_scaleFactor,

self.haar\_minNeighbors,

cv2.CASCADE\_SCALE\_IMAGE,

(self.haar\_minSize, self.haar\_maxSize))

return faces

## 实验内容

### 完成人脸识别

### 完成动作检测

## 参考文献

1. <http://wiki.ros.org/>
2. https://wiki.ros.org/ROS/Tutorials