《计算机视觉-openCV应用技术》

第四章 深度估计与分割

李策

中国矿业大学（北京）计算机科学与技术系

E-mail: celi@cumtb.edu.cn

提纲

第四章 深度估计与分割

4.1 创建模块

4.2 捕获深度摄像头的帧

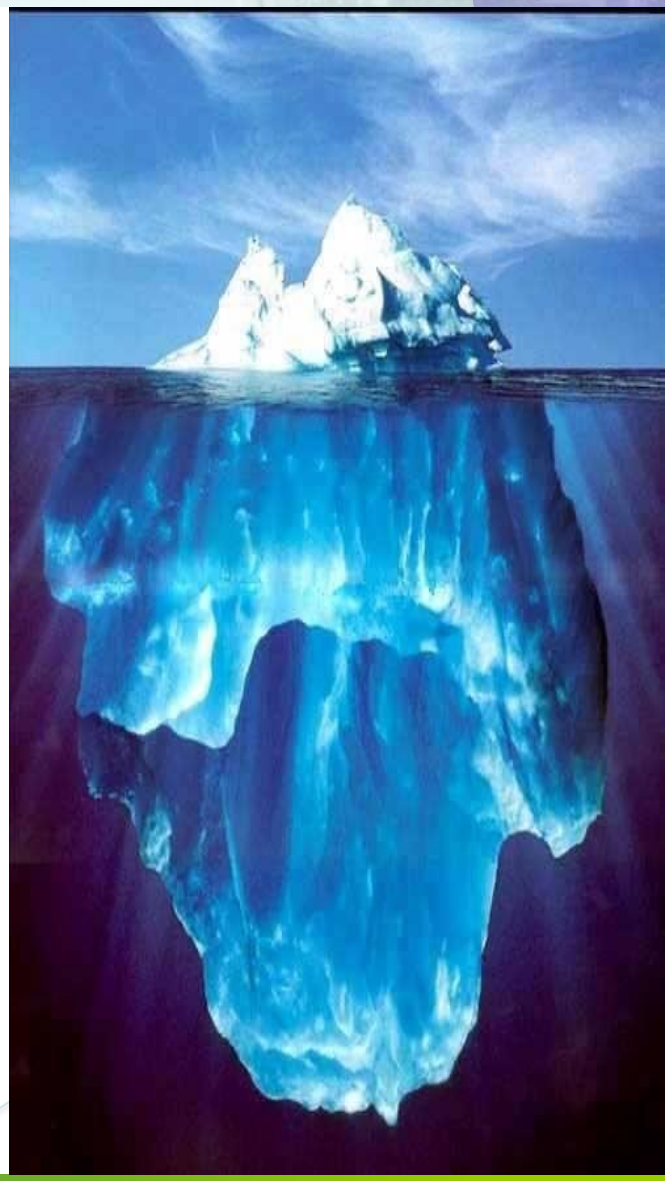
4.3 从视差图得到掩模

4.4 对复制操作执行掩模

4.5 使用普通摄像头进行深度估计

4.6 使用分水岭和GrabCut算法进行物体分割

4.7 总结



4.1 创建模块

1、Cameo.py中捕获和处理深度摄像头数据的代码可以在其他地方重用，所以应该把这部分代码分离，放到一个新的模块中。

创建新文件depth.py，需要在文件中做以下引用声明：

```
import numpy
```

2、修改以前的rects.py文件，将复制操作限制在一个矩形中的非矩形子区域，在rects.py文件中增加以下声明：

```
import numpy
import utils
```

3、在Cameo.py文件中加如下声明：import depth

4.2 捕获深度摄像头的帧

深度相关（depth-related）通道的概念，具体如下：

1、深度图：灰度图像，该图像的每个像素值都是摄像头到物体表面之间距离的估计值。

2、云点图：彩色图像，该图像的每种颜色都对应一个（x、y或z）维度空间。

4.2 捕获深度摄像头的帧

3、视差图：灰度图像，该图像的每个像素值代表物体表面的立体视差。立体视差：假如将从不同视角观察同一场景得到的两张图像叠放在一起，这很可能让人感觉是两张不同的图像，在这个场景中，针对两张图像中两个孪生的物体之间任意一对相互对应的两个像素点，可以度量这些像素之间的距离，这个度量就是立体视差。

4、有效深度掩模：表明一个给定的像素的深度信息是否有效（非零值表示有效，零值表示无效）。

4.3 从视差图得到掩模

建立Cameo的目的：精确地估计面部区域。

1、面部区域的矩形估计： FaceTracker函数和一幅普通的彩色图像

2、判断噪声：通过相应的视差图分析矩形区域，即离得太远或太近不是面部的一部分，可以通过去除这些噪声来精炼面部区域。但是只能够在数据有效的情况下做这个测试，就像有效深度掩模所表明的那样。

下面写一个生成掩模的函数，使得面部矩形中不想要的区域的掩模值为0，想要的区域掩模值为1。

这个函数会将视差图、有效深度掩模和一个矩形作为参数。实现：

4.3 从视差图得到掩模

```
def createMedianMask(disparityMap, validDepthMask, rect = None):
    """Return a mask selecting the median layer, plus shadows."""
    if rect is not None:
        x, y, w, h = rect
        disparityMap = disparityMap[y:y+h, x:x+w]
        validDepthMask = validDepthMask[y:y+h, x:x+w]
    median = numpy.median(disparityMap)
    return numpy.where((validDepthMask == 0) | \
                       (abs(disparityMap - median) < 12),
                       1.0, 0.0)
```

4.4 对复制操作执行掩模

copyRect()函数，它仅将源图像中的指定矩形区域复制到目标图像中。

编辑copyRect()函数，以增加一个掩模参数。这个参数可能为None，这种情况就会使用复制操作原来的实现。否则就需要确保掩模和图像有相同的通道数。假定掩模有一个通道而图像有三个通道（BGR），则可使用numpy.array的repeat()方法和reshape()方法来为掩模复制通道。实现如P57下：

```
def copyRect(src, dst, srcRect, dstRect, mask = None,
             interpolation = cv2.INTER_LINEAR):
    """Copy part of the source to part of the destination."""
```

4.4 对复制操作执行掩模

还需要修改swapRects()函数，该函数使用copyRect()来完成一组矩形区域的循环交换。

修改：增加一个新的参数masks，这是一组要分别传给copyRect()函数的掩模。实现如P58：

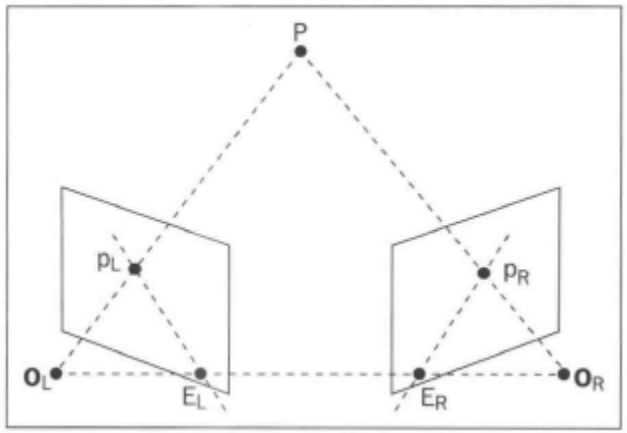
```
def swapRects(src, dst, rects, masks = None,
              interpolation = cv2.INTER_LINEAR):
    """Copy the source with two or more sub-rectangles swapped."""
```

4.5 使用普通摄像头进行深度估计

深度摄像头将传统摄像头和一个红外传感器相结合来帮助摄像头区别相似物体并计算它们与摄像头之间的距离。

立体视觉：计算机视觉的一个分支，它从同一物体的两张不同图像提取三维信息。

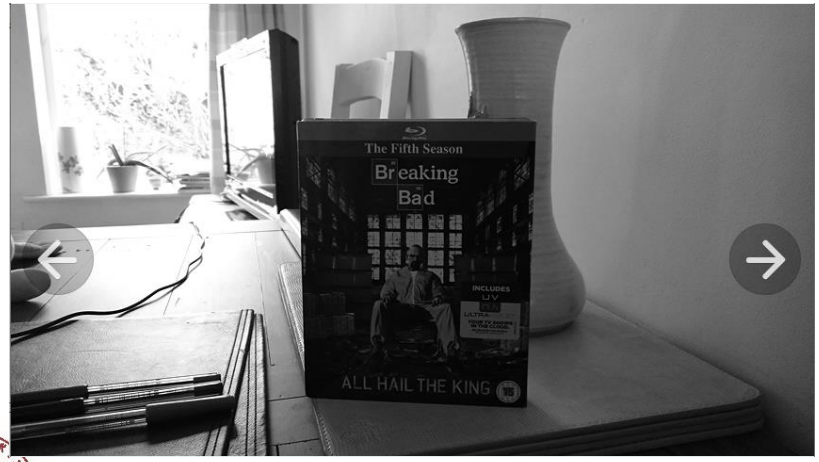
极几何：它跟踪从摄像头到图像上每个物体的虚线，然后在第二张图像做同样的操作，并根据同一个物体对应的线的交叉来计算距离。如图所示：

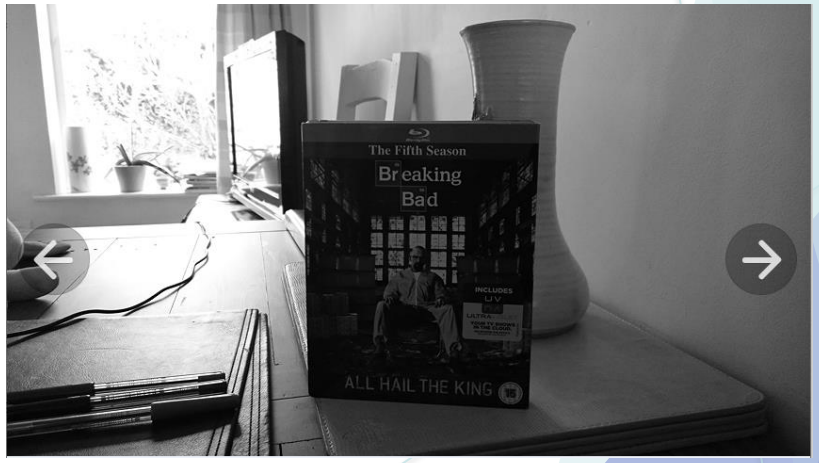


4.5 使用普通摄像头进行深度估计

OpenCV使用极几何来计算所谓的视差图，它是对图像中检测到的不同深度的基本表示。这样就能够提取出一张图片的前景部分而抛弃其余部分。

首先需要同一物体在不同视角下拍摄的两幅图像，但是要注意这两幅图像是距物体相同距离拍摄的，否则计算将会失败，视差图也就没有意义。





4.6 使用分水岭和GrabCut算法进行物体分割

StereoSGBM主要是从二维图片中得到三维信息。而GrabCut能很好实现这个功能的工具。GrabCut算法的实现步骤为：

1）在图片中定义含有（一个或多个）物体的矩形。

2）矩形外的区域被自动认为是背景。

3）对于用户定义的矩形区域，可用背景中的数据来区别它里面的前景和背景区域。

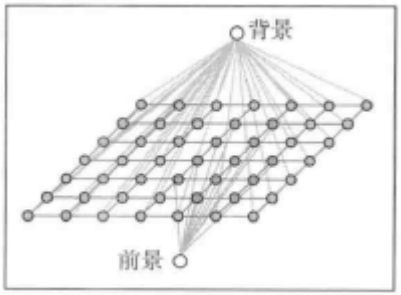
4）用高斯混合模型（Gaussians Mixture Model，GMM）来对背景和前景建模，并将未定义的像素标记为可能的前景或背景。

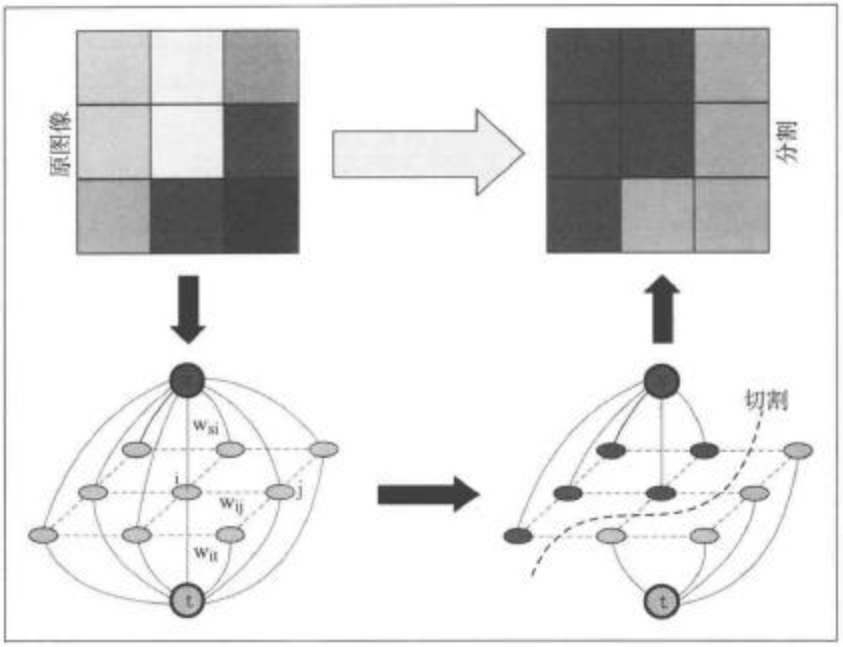
5）图像中的每一个像素都被看作通过虚拟边与周围像素相连接，每条边都有一个属于前景或背景的概率，这基于它与周围像素颜色上的相似性。

4.6 使用分水岭和GrabCut算法进行物体分割

6）每一个像素（即算法中的节点）会与一个前景或背景节点连接，这与下图类似：

7）在节点完成连接后（可能与背景或前景连接），若节点之间的边属于不同终端（即一个节点属于前景，另一个节点属于背景），则会切断它们之间的边，这就能将图像各部分分割出来。下图能很好说明该算法：





4.6.1 用GrabCut进行前景检测的例子

用GrabCut进行前景检测的例子

创建一个脚本来实例化GrabCut，进行分割，然后把结果与原图像一起展示：

4.6.2 使用分水岭算法进行图像分割

分水岭算法叫作分水岭是因为它里面有水的概念。把图像中低密度的区域（变化很少）想象成山谷，图像中高密度的区域（变化很多）想象成山峰。开始向山谷中注入水直到不同的山谷中的水开始汇聚。为了阻止不同山谷的水汇聚，可以设置一些栅栏，最后得到的栅栏就是图像分割。

下图为罗勒叶子的图像，它是香蒜沙司最重要的原料之一：

4.6.2 使用分水岭算法进行图像分割

现在希望通过图像分割将这些罗勒叶子从白色背景中分离出来：

A.导入numpy、cv2、matplotlib，加载叶子图像：

```
import numpy as np
import cv2
from matplotlib import pyplot as plt


img = cv2.imread('images/basil.jpg')
gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
```

4.6.2 使用分水岭算法进行图像分割

B. 将颜色转为灰度之后，为图像设一个阈值，可将图像分为两个部分：黑色部分和白色部分：

```
ret, thresh =
cv2.threshold(gray,0,255,cv2.THRESH_BINARY_INV+cv2.THRESH_
OTSU)
```

C. morphologyEx变换来去除噪声数据，这是一种对图像进行膨胀之后再进行腐蚀的操作，它可以提取图像特征：

```
kernel = np.ones((3,3),np.uint8)
opening = cv2.morphologyEx(thresh,cv2.MORPH_OPEN,kernel,
iterations = 2)
```

4.6.2 使用分水岭算法进行图像分割

D. 通过对 morphologyEx 变换之后的图像进行膨胀操作，可得到大部分是背景的区域：

```
sure_bg = cv2.dilate(opening,kernel,iterations=3)
```

E. 反之，可以通过distanceTransform来获取确定的前景区域。也就是说，这是图像中最可能是前景的区域，越是远离背景区域的边界的点越可能属于前景。在得到distanceTransform操作的结果之后，应用一个阈值来决定哪些区域是前景，这样得到正确结果的概率很高：

```
dist_transform = cv2.distanceTransform(opening,cv2.DIST_L2,5)
ret, sure_fg =
cv2.threshold(dist_transform,0.7*dist_transform.max(),255,0)
```

4.6.2 使用分水岭算法进行图像分割

F. 确定前景和背景中重合的区域，可从sure_bg与sure_fg的集合相减来得到：

```
sure_fg = np.uint8(sure_fg)
unknown = cv2.subtract(sure_bg,sure_fg)
```

G. 通过connectedComponents 函数，在不同的山谷之间设定栅栏

```
ret, markers = cv2.connectedComponents(sure_fg)
```

H. 背景区域加1，将unknow区域设为0

```
markers = markers+1
markers[unknown==255] = 0
```

4.6.2 使用分水岭算法进行图像分割

I. 把栅栏染成红色：

```
markers = cv2.watershed(img,markers)
img[markers == -1] = [255,0,0]
```

4.7 总结

本章介绍了：

- 二维输入中得到三维信息
- 深度摄像头、极几何、立体图像
- 视差图的计算
- 两种方法：GrabCut和分水岭来进行图像分割

下一章重点：特征描述符和关键点检测