图像处理

1

图像处理技术文档 - 特征提取与轮廓检测

本文档详细描述了使用形态学操作和轮廓检测技术从二值图像中提取特征的图像处理流程。此流程主要用于分析和提取图像中的结构化特征，适用于需要精细操作的图像处理任务。以下是具体的步骤和方法说明。

1. 图像读取与二值化

首先，从指定路径读取图像文件，并将其加载为灰度图像。然后，应用阈值操作以进行二值化处理，以便于后续的形态学操作。

```
fn1 = "images/task1.5.png"
I1 = cv.imread(fn1, cv.IMREAD_GRAYSCALE)
if not isinstance(I1, np.ndarray) or
I1.data == None:
    print("Error reading file \"
{}\"".format(fn1))
else:
    ret, I1 = cv.threshold(I1, 127, 255,
cv.THRESH_BINARY_INV)
```

2. 形态学预处理

使用形态学操作移除二值图像中的小对象，这有助于减少噪声并突出主要特征。

```
I1clean1 = bwareaopen(I1, 200)
```

3. 开运算

应用形态学开运算（先腐蚀后膨胀）以平滑对象的边界，断开狭窄的连接点，并消除小的突出部分。此处选择进行15次迭代，以达到显著的平滑效果。

```
element3x3 =
cv.getStructuringElement(cv.MORPH_ELLIPSE
, (3, 3))
I1clean4 = cv.morphologyEx(I1clean1,
cv.MORPH_OPEN, element3x3, iterations=15)
```

4. 轮廓检测

通过对开运算处理后的图像进行一次轻微的腐蚀操作，从而提取对象的内部轮廓。这有助于突出显示对象的边界和内部结构。

```
I1clean5 = I1clean4 - cv.morphologyEx(I1clean4, cv.MORPH_ERODE, element3x3, iterations=1)
```

5. 结果展示

展示处理过程中的不同阶段图像，包括原始二值图像、去除小对象后的图像、经过15次开运算迭代后的图像，以及最终提取的内部轮廓图像。

```
ShowImages([
    ("Source", I1),
    ("bwareaopen", I1clean1),
    ("Open 15 iterations", I1clean4),
    ("Internal contour", I1clean5)
], 3)
```

本技术文档提供了详细的图像处理步骤，旨在通过形态学操作和轮廓提取技术来分离和分析图像中的特定特征，适用于图像分析、机器视觉和数字图像处理等多种应用场景。

2

图像处理技术文档 - 特征区域提取

本文档详细介绍了一种图像处理技术，使用多个形态学操作来提取图像中的特定特征区域。此方法适用于需要从复杂背景中分离特定特征的场景。以下是详细的步骤和方法解释。

1. 图像读取与预处理

首先从指定路径读取彩色图像。如果图像读取失败，将显示错误信息。成功读取后，将图像转换为灰度图像，为进一步处理做准备。

```
fn4 = "images/task_2.2.png"
I4 = cv.imread(fn4, cv.IMREAD_COLOR)
if not isinstance(I4, np.ndarray) or I4.data == None:
  print("Error reading file \"{}\"".format(fn4))
else:
  I4gray = cv.cvtColor(I4, cv.COLOR_BGR2GRAY)
```

2. 二值化处理

使用Otsu的方法自动确定最佳阈值，将灰度图像转换为二值图像，便于后续的形态学操作。

```
ret, I4m = cv.threshold(I4gray, 0, 255, cv.THRESH_OTSU)
```

3. 形态学预处理

使用形态学操作来填充二值图像中的小空洞并消除噪点。

```
I4clean1 = ~bwareaopen(~I4m, 200)  # 填充
I4clean1 = bwareaopen(I4clean1, 100)  # 消除噪点
```

4. 形态学开运算

通过先腐蚀后膨胀的形态学开运算进一步清理和平滑图像边界。

```
element3x3 = cv.getStructuringElement(cv.MORPH_ELLIPSE, (3, 3))
I4clean2 = cv.morphologyEx(I4clean1, cv.MORPH_OPEN, element3x3, iterations=3)
```

5. 腐蚀操作

使用更大的腐蚀核深度腐蚀，进一步突出图像中的主要特征。

```
element7x7 =
cv.getStructuringElement(cv.MORPH_ELLIPSE
, (7, 7))
I4centers = cv.morphologyEx(I4clean2,
cv.MORPH_ERODE, element7x7,
iterations=10,
borderType=cv.BORDER_CONSTANT,
borderValue=(0))
```

6. 边界提取

通过逐步腐蚀和开运算迭代处理，精确提取特征边界。

```
I4border = ~I4centers
I4tmp = I4border.copy()
while cv.countNonZero(I4tmp) > 0:
  I4tmp_erode = cv.erode(I4tmp,
element3x3,
borderType=cv.BORDER_CONSTANT,
borderValue=(255))
  I4tmp_open =
cv.morphologyEx(I4tmp_erode,
cv.MORPH_OPEN, element5x5,
borderType=cv.BORDER_CONSTANT,
borderValue=(0))
  subset = cv.bitwise_and(~I4tmp_open,
I4tmp_erode)
  I4border2 = cv.bitwise_or(subset,
I4border2)
  I4tmp = I4tmp_erode
```

7. 特征区域分离

使用掩模将处理后的图像中的特征区域与背景分离。

```
I4m_split = cv.bitwise_and(I4clean2, ~I4border2)
I4m_split_resized = cv.resize(I4m_split, (I4.shape[1], I4.shape[0]))
I4split = I4.copy()
I4split[I4m_split_resized < 128] = 0
```

8. 结果展示

展示原始图像、二值化后的图像、分离出的特征区域以及对应的形态学处理后的图像。

```
ShowImages([("Source", I4),
            ("Source morphology", I4m),
            ("Separated", I4split),
            ("Separated morphology", I4m_split_resized)], 2)
```

本技术文档提供了一系列图像处理步骤，旨在通过形态学操作和边界提取技术来分离图像中的特定区域，适用于需要精确图像分析和处理的多种应用场景。

3

图像处理技术文档 - 分水岭算法

本文档详细介绍了一种图像处理技术，使用分水岭算法进行图像分割，主要目的是提取图像的重要特征，例如前景和背景，并对特定区域进行标记。下面逐步解释了整个图像处理流程。

1. 图像读取与显示

首先，从指定路径读取彩色图像。如果图像读取失败，系统会显示错误信息。成功读取后，会展示原始图像。

```
fn5 = "images/task_2.2.png"
I5 = cv.imread(fn5, cv.IMREAD_COLOR)
if not isinstance(I5, np.ndarray) or I5.data is None:
  print("Error reading file")
else:
  ShowImages([("Source", I5)])
```

2. 灰度转换与二值化

将彩色图像转换为灰度图像，再应用Otsu的二值化方法将图像从灰度转换为黑白，以便更好地处理图像。

```
I5gray = cv.cvtColor(I5,
cv.COLOR_BGR2GRAY)
ret, I5bw = cv.threshold(I5gray, 0, 255,
cv.THRESH_BINARY + cv.THRESH_OTSU)
ShowImages([("Grayscale", I5gray),
("Morphology", I5bw)], 2)
```

3. 图像清理

使用形态学操作去除图像中的噪声和小的不相关区域。通过开运算和自定义的结构元素来进一步增强图像结构的清晰度。

```
I5bw_clean = ~bwareaopen(~I5bw, 200)
I5bw_clean = bwareaopen(I5bw_clean, 100)
element5x5 =
cv.getStructuringElement(cv.MORPH_ELLIPSE
, (5, 5))
I5bw_clean = cv.morphologyEx(I5bw_clean,
cv.MORPH_OPEN, element5x5, iterations=3)
ShowImages([("Cleaned up", I5bw_clean)],
1)
```

4. 距离变换与前景提取

对清理后的图像使用距离变换，计算每个前景像素到最近背景像素的距离。利用这些距离值进行阈值处理，突出前景。

```
I5dist = cv.distanceTransform(I5bw_clean, cv.DIST_L2, 5)
ret, I5fg = cv.threshold(I5dist, 0.6 * I5dist.max(), 255, 0)
I5fg = (I5fg.astype(np.float32) / I5fg.max() * 255).astype(np.uint8)
ShowImages([("Distance", I5dist.astype(np.float32) / I5dist.max()), ("Foreground", I5fg)], 2)
```

5. 连通域分析

使用连通域分析方法标记图像中的独立前景区域，以便在接下来的步骤中使用分水岭算法对它们进行分割。

```
n_markers, I5markers_fg = cv.connectedComponents(I5fg)
ShowImages([("Foreground", I5fg), ("Foreground markers", cv.applyColorMap((I5markers_fg.astype(np.float32) * 255 / n_markers).astype(np.uint8), cv.COLORMAP_JET))], 2)
```

6. 分水岭算法应用

应用分水岭算法对前景标记的图像进行分割。这一步是将图像中的每个前景区域明确分割出来，使其成为单独的区域。

```
I5markers = I5markers_fg.copy()
I5markers = cv.watershed(I5, I5markers)
I5markers_bg =
np.zeros_like(I5markers_fg)
I5markers_bg[I5markers == -1] = 255
ShowImages([("Foreground watershed",
cv.applyColorMap((I5markers.astype(np.flo
at32) * 255 / ret).astype(np.uint8),
cv.COLORMAP_JET)), ("Background",
I5markers_bg)], 2)
```

7. 最终视觉呈现

在图像中标记分水岭线，并对最终图像中的分割结果进行可视化。

```
I5result = I5.copy()
I5result[I5markers == -1] = (255, 0, 0)
ShowImages([("Markers after watershed",
cv.applyColorMap((I5markers.astype(np.flo
at32) * 255 / (ret +
1)).astype(np.uint8), cv.COLORMAP_JET)),
("Image with borders", I5result)])
```

本技术文档提供了图像处理的详细步骤，可以应用于需要图像分割的多种场景。通过这些步骤，可以有效地从复杂背景中提取有用的图像前景信息。