中山大学计算机学院算法设计与分析本科生实验报告

课程名称：算法设计与分析

教学班级	专业（方向）	学号	姓名
2班	计算机科学与技术	21307174	刘俊杰

一、实验内容

应用最大流最小割集思想对图像进行分割。

二、实验思想

在课堂上我们学习了最大流和最小割的概念,并学会了如何求解最大流和最小割，本实验的目标是将最大流最小割集思想应用到图像分割。

如何将将最大流最小割集思想应用到图像分割:

将最大流最小割算法应用于图像分割的基本思想是利用图割问题的最小割，来划分图中的像素点，从而实现对图像的分割。

1. **图割的本质：** 图割问题是在图中找到一组最小容量的边，将图分为两个不相交的部分，即源和汇。这与图像分割的目标是找到一个最小的代价（容量）来分离图像区域是相符的。
2. **图像分割的问题建模：** 将图像分割问题建模为图割问题，其中图的节点表示图像中的像素，图的边表示像素之间的关系。这样，通过调整边的容量，我们可以控制图割的位置，从而实现图像的分割。
3. **最小割的求解：** 最大流最小割算法通过在网络中找到最大流，隐含地找到了最小割。在图像分割的背景下，最小割将图像划分为两个区域，一个表示前景，一个表示背景，通过最小割的位置，可以实现对图像的分割。
4. **图像的连通性：** 图像中相邻像素之间有着一定的连通性，这正好与图割问题的建模相吻合。最小割将通过具有最小容量的边来划分图，这会在图像中保留一些连通的结构，而不是将图随机地分割成两半。

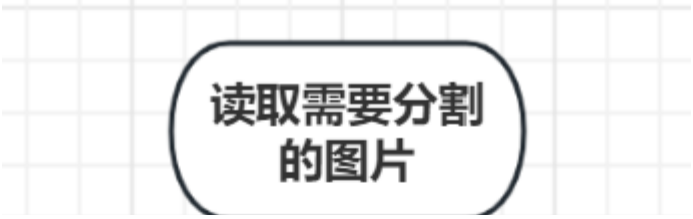
参考资料

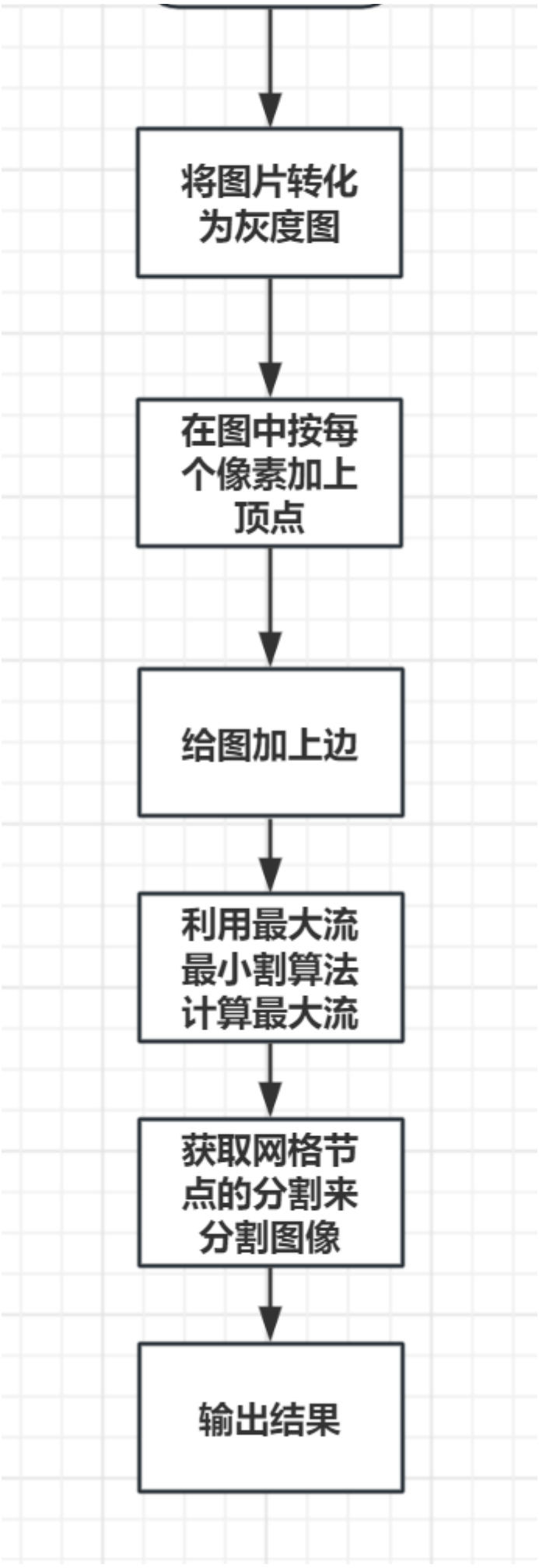
为了了解最大流和最小割应用到图像分割的原理,我参考了如下资料:

https://xhy3054.github.io/2021/01/24/2021-01-24-graph-cut/

- 该博客详细讲解了最大流和最小割应用到图像分割的方法

三、流程图




将图片转化为灰度图
在图中按每个像素加上顶点
给图加上边
利用最大流最小割算法计算最大流
获取网格节点的分割来分割图像
输出结果

四、源代码

代码实现

1.转为灰度图

```
# 阈值
threshold = 220
# 加载图像并将其转换为灰度图像
original_img = cv2.imread('D:\\d_code\\algorithm\\Lab4\\7.png')
img = cv2.cvtColor(original_img, cv2.COLOR_BGR2GRAY)
img = 255 * (img > threshold).astype(np.uint8)
```

2.创建图并加上点和边

```
#创建图
g = maxflow.Graph[int]()

#按每个像素加顶点
nodeids = g.add_grid_nodes(img.shape)

# 添加终端边图像像素是从源节点到图像像素的边的容量
# 反转的图像像素是从图像像素到汇节点的边的容量
g.add_grid_tedges(nodeids, img, 255-img)
```

3.计算最大流,输出分割结果

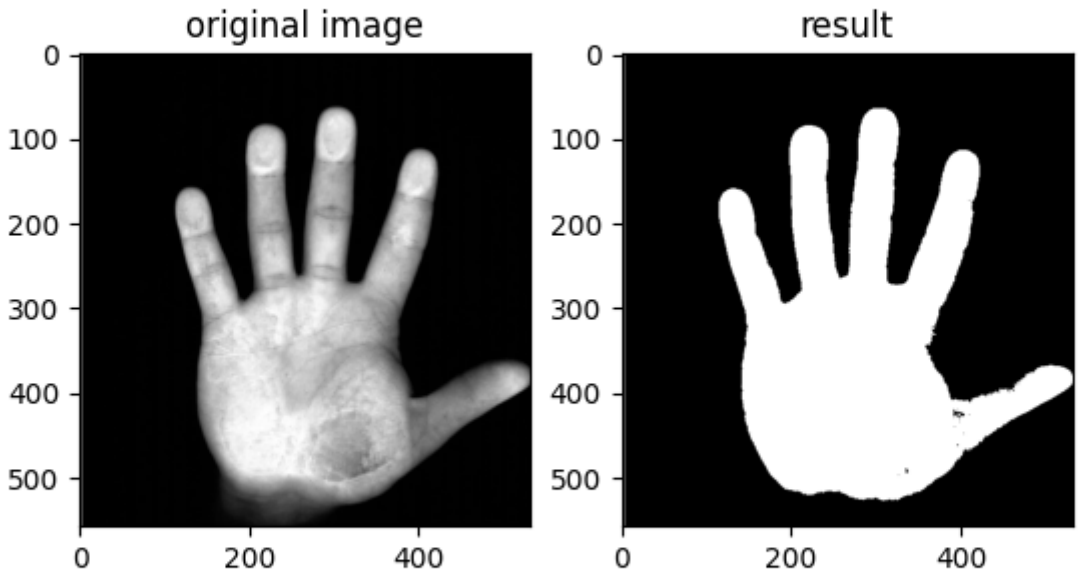
```
# 计算最大流
g.maxflow()

# 得到分割后的图像
sgm = g.get_grid_segments(nodeids)
img_segment = np.logical_not(sgm).astype(np.uint8) * 255

# 将 BGR 转换为 RGB 顺序  因为plt.imshow()使用 RGB（红绿蓝）通道顺序
img_rgb = cv2.cvtColor(original_img, cv2.COLOR_BGR2RGB)

# 输出结果
plt.subplot(121)
plt.imshow(img_rgb)
plt.title('original image')
plt.subplot(122)
plt.title('result')
plt.imshow(img_segment, cmap='gray')
plt.show()
```

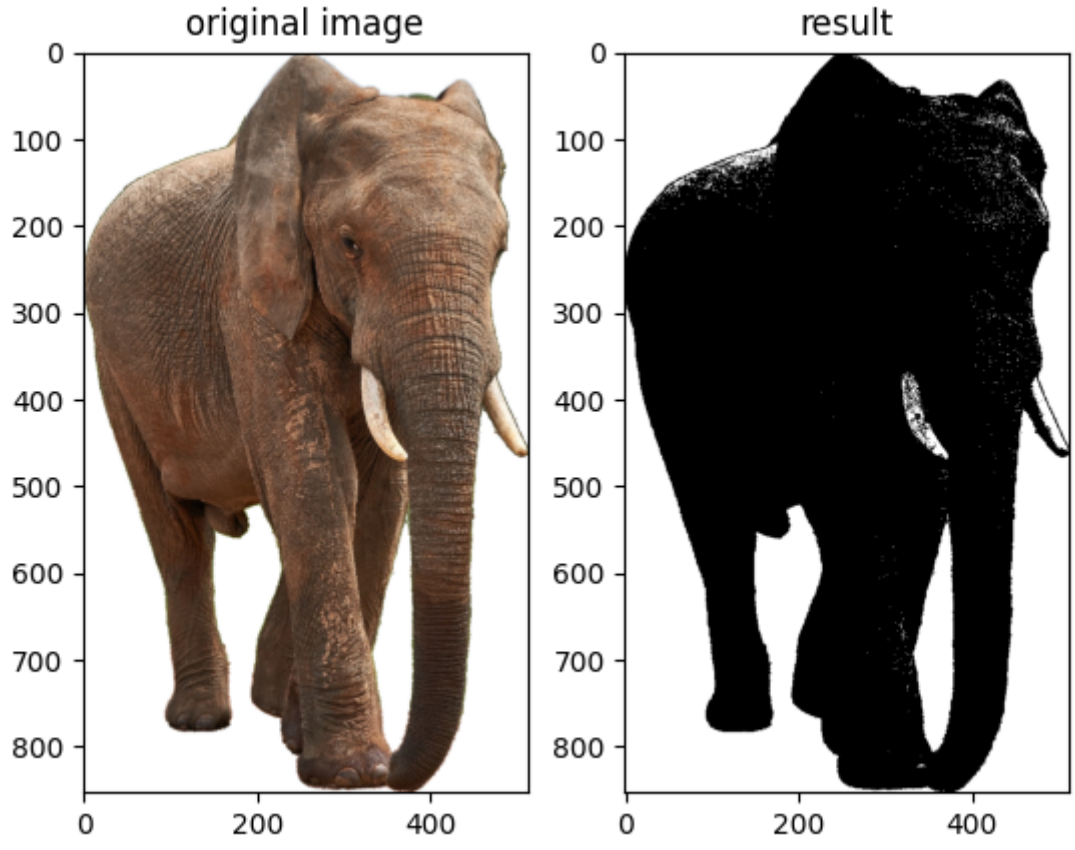
完整代码

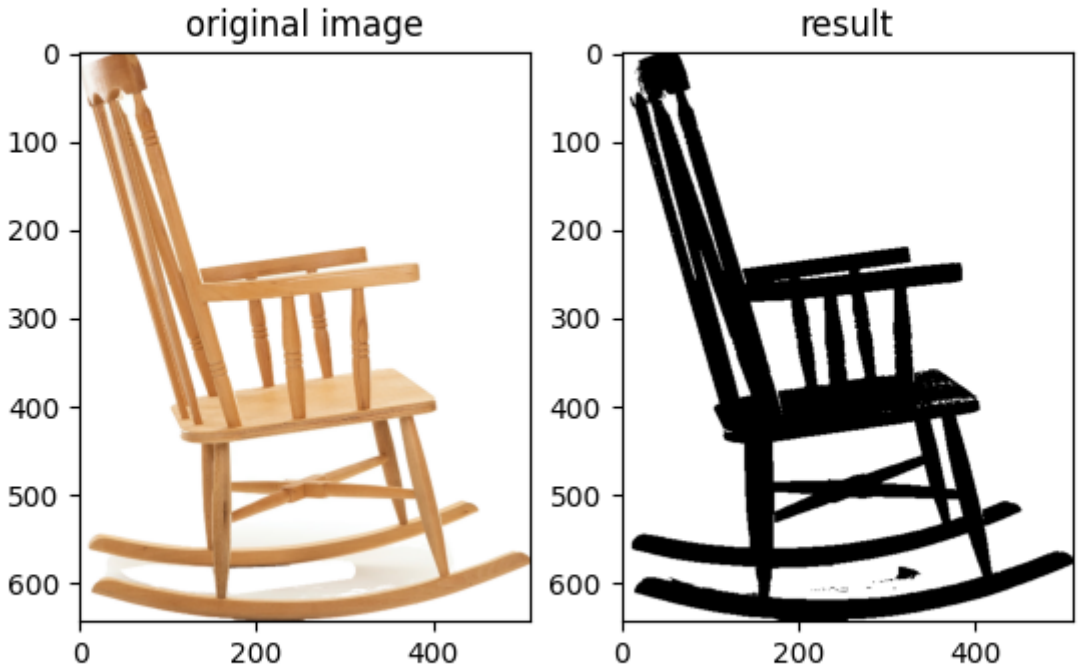
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
import maxflow

# 阈值
threshold = 220
# 加载图像并将其转换为灰度图像
original_img = cv2.imread('D:\\d_code\\algorithm\\Lab4\\7.png')
img = cv2.cvtColor(original_img, cv2.COLOR_BGR2GRAY)
img = 255 * (img > threshold).astype(np.uint8)

#创建图
g = maxflow.Graph[int]()

#按每个像素加顶点
nodeids = g.add_grid_nodes(img.shape)

# 添加终端边图像像素是从源节点到图像像素的边的容量
# (255-img)反转的图像像素是从图像像素到汇节点的边的容量
g.add_grid_tedges(nodeids, img, 255-img)

# 计算最大流
g.maxflow()

# 得到分割后的图像
sgm = g.get_grid_segments(nodeids)
img_segment = np.logical_not(sgm).astype(np.uint8) * 255

# 将 BGR 转换为 RGB 顺序  因为plt.imshow()使用 RGB（红绿蓝）通道顺序
img_rgb = cv2.cvtColor(original_img, cv2.COLOR_BGR2RGB)

# 输出结果
plt.subplot(121)
plt.imshow(img_rgb)
plt.title('original image')
plt.subplot(122)
plt.title('result')
plt.imshow(img_segment, cmap='gray')
plt.show()
```

五、输出结果

由于每张图片分割目标的通道强度不同,有些地方会偏亮,可以调整代码中的阈值来改善分割效果


original image
result
0
100
200
300
400
500
0
200
400


original image
result
0
100
200
300
400
500
600
700
800
0
200
400



可以看到最大流最小割算法可以实现图像分割