浙江大學

报告题目：GrabCut 图像分割与 Border Matting

课程名称：	计算摄影学
指导教师：	章国锋/周晓巍
姓名：	***
学号：	317010****
专业：	****
学院：	计算机科学与技术学院
邮箱：	317010****@zju.edu.cn
日期：	2019 年 6 月 22 日

目录

一、 GrabCut 项目简介

GrabCut 是微软研究院的一个课题，主要功能是图像分割。作为一个经典的图像分割方法，它利用图像中的纹理信息和边界信息，结合手工交互和自动的 Graph-Cut 算法来对静态图像进行高效的前景/背景分割。

在实现前后景分割时，GrabCut 首先定义混合多高斯模型，而后根据用户的交互操作初始化 GMM 变量，计算得到单高斯函数的权重系数和均值，并得到协方差，应用最小割求解进行估计分割，而后重复训练 GMM 直至收敛。

相比于 GraphCut ，GrabCut 方法有以下改进：

1. GraphCut 的前景与背景的模型是灰度直方图，而 GrabCut 取代为 RGB 三通道的高斯混合模型；
2. GraphCut 是能量最小化是一次性达到的，而 GrabCut 取代为一个不断进行分割估计和模型参数学习的交互迭代过程；
3. GrabCut 支持不完整标记，比如比如可以只标记背景，之后迭代步骤会对随后抽取的像素的临时标签进行处理；若是分割不理想，可以加入编辑步骤，即手动标记前景或背景，此为可选项，而 GraphCut 必须由用户提供前景和背景的一些种子。

为了解决 GrabCut 方法得到的硬分割图像的边缘效果问题，同时开发了 Border Matting 方法处理分割图像的边界，使之更加自然平滑。

二、 基础理论综述

在 Grabcut 项目中，应用的基础理论有 Graphcut 算法和高斯混合模型 (GMM)。以下给出这两种基础理论说明和选择其作为基础理论并应用的原因。并对在此基础上发展的 Grabcut 算法和处理边缘的 Border Matting 算法进行理论分析。

2.1 Graphcut 算法

Graph cuts 是一种十分有用和流行的能量优化算法，在计算机视觉领域普遍应用于前背景分割（Image segmentation）、立体视觉（stereo vision）、抠图（Image matting）等。

(1) Graphcut 概述

Graphcut 把图像分割问题与图的最小割（mincut）问题相关联。首先用一个无向图 G=<V，E> 表示要分割的图像，V 和 E 分别是顶点（vertex）和边（edge）的集合。此处的 Graph 和普通的 Graph 稍有不同。普通的图由顶点和边构成，如果边的有方向的，这样的图被则称为有向图，否则为无向图，且边是有权值的，不同的边可以有不同的权值，分别代表不同的物理意义。而 Graph Cuts 图是在普通图的基础上多了 2 个顶点，这 2 个顶点分别用符号”S”和”T”表示，统称为终端顶点。其它所有的顶点都必须和这 2 个顶点相连形成边集合中的一部分。所以 Graph Cuts 中有两种顶点，也有两种边。

Graph Cuts 中的 Cuts 是指这样一个边的集合，很显然这些边集合包括了上面 2 种边，该集合中所有边的断开会导致残留”S”和”T”图的分开，所以就称为“割”。如果一个割，它的边的所有权值之

和最小，那么这个就称为最小割，也就是图割的结果。而福特-富克森定理表明，网路的最大流 maxflow 与最小割 mincut 相等。所以由 Boykov 和 Kolmogorov 发明的 max-flow/min-cut 算法就可以用来获得 s-t 图的最小割。这个最小割把图的顶点划分为两个不相交的子集 S 和 T，其中 s S，t T 和 S T=V 。这两个子集就对应于图像的前景像素集和背景像素集，那就相当于完成了图像分割。

这一算法的最大优势就是可以可以通过标记一些点集为前景和背景，实现最优分界线的自动生成。比较契合图像分割的应用场景。

(2) Graphcut 工作流程

对于一个图像分割问题而言，定义 z 为图像数据，α 为灰度值，0 代表背景，而 1 代表前景，θ 为图像背景和前景的分类方法，它分别由前景和背景的灰度直方图构成。

$$\underline{\theta} = \{h(z;\alpha), \alpha = 0, 1\} \tag{1}$$

对于这样的像素标记问题，把目标（s-node）的 label 设为 1，背景（t-node）的 label 设为 0，这个过程可以通过最小化图割来最小化能量函数得到。那很明显，发生在目标和背景的边界处的 cut 就是我们想要的（相当于把图像中背景和目标连接的地方割开，那就相当于把其分割了）。同时，这时候能量也应该是最小的。假设整幅图像的标签 label（每个像素的 label）为 L= l1,l2,,, lp ，其中 li 为 0（背景）或者 1（目标）。那假设图像的分割为 L 时，图像的 Gibbs 能量项可以表示为：

$$\mathbf{E}(\underline{\alpha}, \underline{\theta}, \mathbf{z}) = U(\underline{\alpha}, \underline{\theta}, \mathbf{z}) + V(\underline{\alpha}, \mathbf{z}) \tag{2}$$

其中数据项 U 在给定直方图模型的情况下评估不透明度 α 和图像数据 z 的匹配程度，它被定义为：

$$U(\underline{\alpha}, \underline{\theta}, \mathbf{z}) = \sum_{n} -\log h\left(z_n; \alpha_n\right) \tag{3}$$

而平滑项 V 则表示为：

$$V(\underline{\alpha}, \mathbf{z}) = \gamma \sum_{(m,n)\in\mathbf{C}} \operatorname{dis}(m,n)^{-1} \left[\alpha_n \neq \alpha_m\right] \exp -\beta \left(z_m - z_n\right)^2 \tag{4}$$

在公式 4中 C 是相邻像素对的集合，dis (•) 是相邻像素的欧几里德距离。在实践中，如果像素是水平/垂直或对角相邻（8 路连接），则可以通过将像素定义为邻居来获得良好的结果。常量 β 被定义为：

$$\beta = \left(2\left\langle \left(z_m - z_n\right)^2 \right\rangle\right)^{-1} \tag{5}$$

此处公式 5中 $< \cdot >$ 代表在一个图像集上的期望。

将能量公式定义完整后，图像分割问题就可以被定义为一个全局能量函数最小化的问题，

$$\hat{\underline{\alpha}} = \arg\min_{\underline{\alpha}} \mathbf{E}(\underline{\alpha}, \underline{\theta}) \tag{6}$$

应用最小割算法进行图像分割，就能够得到基础的硬分割结果。

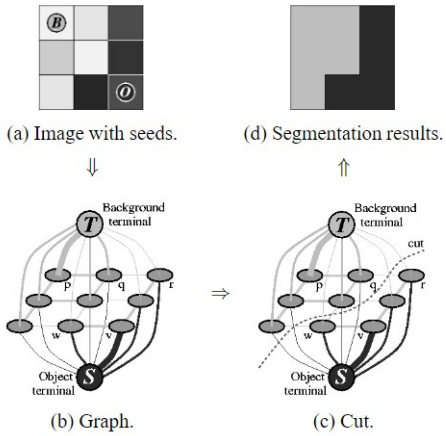


图 1: min-cut 示意图

(3) min-cut/max-flow 算法

最大流/最小割（Max-Flow/Min-Cut）在解决计算机视觉中的能量方程最小化问题的强大，最早发现是 Greig 于 1989 年发表的文章：Exact Maximum A Posteriori Estimation for Binary Images。

通过能量项构建无向图，借助与最小割算法，我们就可以实现对像素点的分类，即图像的分割，留下的边都是相似且联系紧密的邻域像素之间的，详细的关于最小割算法可以应用于能量方程最小化问题的证明不在此赘述。

2.2 高斯混合模型 (GMM)

高斯混合模型，英文全称：Gaussian mixture model，简称 GMM。高斯混合模型就是用高斯概率密度函数 (二维时也称为：正态分布曲线) 精确的量化事物，将一个事物分解为若干基于高斯概率密度函数行程的模型。

(1) 高斯混合模型的特性

就单高斯模型 (SGM) 而言，多维变量 x 服从高斯分布时，它的概率密度函数为：

$$N(x;u,\Sigma)=\frac{1}{\sqrt{2\pi|\Sigma|}}\exp\left[-\frac{1}{2}(x-u)^T\Sigma^{-1}(x-u)\right] \tag{7}$$

此时 x 是维度为 d 的列向量;u 是模型期望;Σ 是模型方差。

在实际应用中，u 通常用样本均值来代替，Σ 通常用样本方差来代替。每个类别都有自己的 u 和 Σ，把 x 代入公式 7，当概率大于一定阈值时我们就认为 x 属于该类。从几何上讲，单高斯分布模型在二维空间应该近似于椭圆，在三维空间上近似于椭球。但实际上在很多分类问题中，属于同一类别的样本点并不满足“椭圆”分布的特性，它们的分类可能由更加复杂的多种特性决定的，因此很难得到基于单一特征的优良分布。

由于单高斯模型的这一巨大缺陷，人们提出了混合高斯模型：

$$\Pr(x)=\sum_{k=1}^{K}\pi_k N\left(x;u_k,\Sigma_k\right) \tag{8}$$

其中的任意一个高斯分布 $N(x;u_k,\Sigma_k)$ 叫作这个模型的一个 component；K 需要事先确定好，代表 component 个数；而 π_k 则为权值系数，模型内所有 component 的权值系数之和为 1。

每个 GMM 由 K 个 Gaussian 分布组成，每个 Gaussian 称为一个“Component”，这些 Component 线性加成在一起就组成了 GMM 的概率密度函数。当 K 足够大时，GMM 可以用来逼近任意连续的概率密度分布。

(2) 模型参数的求解

GMM 是一种聚类算法，每个 component 就是一个聚类中心。即在只有样本点，不知道样本分类 (含有隐含变量) 的情况下，需要计算出模型参数 (π,u 和 Σ)。

本次实验中用到的聚类算法为 K-means 算法，故以下主要介绍该算法的基本概念。k-means 算法是一种得到最广泛使用的聚类算法。它是将各个聚类子集内的所有数据样本的均值作为该聚类的代表点。

k-means 计算过程：

1. 随机选择 k 个类簇的中心；
2. 计算每一个样本点到所有类簇中心的距离，选择最小距离作为该样本的类簇；
3. 重新计算所有类簇的中心坐标，直到达到某种停止条件 (迭代次数/簇中心收敛/最小平方误差)。

期望最大化（Expectation Maximization）是在含有隐变量（latent variable）的模型下计算最大似然的一种算法。我们可以用 EM 算法推导 K-means：k-means 算法是高斯混合聚类在混合成分方差相等，且每个样本仅指派一个混合成分时候的特例；k-means 中每个样本所属的类就可以看成是一个隐变量，在 E 步中，我们固定每个类的中心，通过对每一个样本选择最近的类优化目标函数，在 M 步，重新更新每个类的中心点，该步骤可以通过对目标函数求导实现，最终可得新的类中心就是类中样本的均值。

(3) 高斯混合模型与图像分割

由于在图像分割问题中，前景和背景的颜色信息和纹理信息往往都不是单一的，而是多样的、复杂的，所以单高斯模型往往会面临分类准确率低或适用范围小等问题。为了解决这一图像分类问题，能够统计并反映更多特征信息的混合高斯模型得到了应用。

在 Graphcut 这一图像分割算法中，以单高斯模型得到的灰度直方图为分类依据进行最大似然估计，这不可避免地会导致重要的颜色信息的丢失，而实际上，颜色信息可以为图像分割的效果带来很大的提升。又由于构建足够的颜色空间直方图是不可行的，因此采用高斯混合模型 GMM。GMM 是 k 个组件（component）的满协方差（full-covariance) 高斯混合，通常 k=5。

2.3 GrabCut 算法概述

作为一种应用了 GraphCut 图割算法和混合高斯模型的图像分割方法，GrabCut 和 GraphCut 的实现过程十分相似。下面给出 GrabCut 的理论概述。

由于引入了 GMM，拥有 K 个分量 (component)，表示为 $\mathbf{k}=\{k_1,\ldots,k_n,\ldots,k_N\}$。通常 K 取值为 5。每个像素依据特征归属于特定的 component。因此 GrabCut 的能量函数相比 GraphCut 多了这一变量，变为：

$$\mathbf{E}(\underline{\alpha},\mathbf{k},\underline{\theta},\mathbf{z})=U(\underline{\alpha},\mathbf{k},\underline{\theta},\mathbf{z})+V(\underline{\alpha},\mathbf{z}) \tag{9}$$

数据项 U 被定义为各个单高斯模型之和：

$$U(\underline{\alpha},\mathbf{k},\underline{\theta},\mathbf{z})=\sum_n D\left(\alpha_n,k_n,\underline{\theta},z_n\right) \tag{10}$$

此处 $D\left(\alpha_n,k_n,\underline{\theta},z_n\right)=-\log p\left(z_n|\alpha_n,k_n,\underline{\theta}\right)-\log\pi\left(\alpha_n,k_n\right)$，并且 $p(\cdot)$ 是高斯概率分布，而 $\pi(\cdot)$ 表示权值系数，因此有：

$$\begin{aligned}D\left(\alpha_n,k_n,\underline{\theta},z_n\right)=&-\log\pi\left(\alpha_n,k_n\right)+\tfrac{1}{2}\log\det\Sigma\left(\alpha_n,k_n\right)\\&+\tfrac{1}{2}\left[z_n-\mu\left(\alpha_n,k_n\right)\right]^{\top}\Sigma\left(\alpha_n,k_n\right)^{-1}\left[z_n-\mu\left(\alpha_n,k_n\right)\right]\end{aligned} \tag{11}$$

所以对于这样一个分割模型，需要用到以下参数：

$$\underline{\theta}=\{\pi(\boldsymbol{\alpha},k),\mu(\alpha,k),\Sigma(\alpha,k),\alpha=0,1,k=1\ldots K\} \tag{12}$$

除了数据项以外为了保证前景和背景分割效果，GrabCut 和 GraphCut 同样引入平滑项 V 作为能量函数的一部分，更好利用纹理信息实现分割：

$$V(\underline{\alpha},\mathbf{z})=\gamma\sum_{(m,n)\in\mathbf{C}}\left[\alpha_n\neq\alpha_m\right]\exp-\beta\left\|z_m-z_n\right\|^2 \tag{13}$$

至此 GrabCut 的能量函数的定义也已经完整，为了实现能量最小化的目标，我们首先训练混合高斯模型，而后同样应用最小割算法，对图像进行硬分割。

2.4 Border Matting 算法概述

Border Matting 算法概述上一步的 GrabCut 得到的分割结果被称作硬分割，而 Bonier Matting 算法通过“边界消光”机制，使得硬分割边界周围的一个条带透明，从而达到图像边界更加自然的目的。在 Border Matting 算法中，首先需要找到的是图像的边界信息，因为要处理的部分就足图像的边界区域, 让图像的过渡更加自然。一般选用图像的 mask 来使用 Canny 算子进行边缘检测。获取到保存了边缘信息的图像后，再对图像进行轮廓参数化，找到同屈于一个连续轮廓上的像素点，并且保存所有轮廓像素点的信息, 为每个像素点分配一个单独的编号，因为后续需要在每一个轮廓像素点的基础上进行扩展，以找到整个需要处理的条带。条带查找一般使用宽度搜索的方式，获得到距离每个中心像素点的一定距离以内的像素点，共同组成图像前景的边界条带。

能量函数如下：

$$E=\sum_{n\in T_U}\tilde{D}_n\left(\alpha_n\right)+\sum_{t=1}^{T}\tilde{V}\left(\Delta_t,\sigma_t,\Delta_{t+1},\sigma_{t+1}\right) \tag{14}$$

参数 Δ 和 α 通过获取。

其中平滑项的计算方法如下：

$$\tilde{V}\left(\Delta,\sigma,\Delta',\sigma'\right)=\lambda_1\left(\Delta-\Delta'\right)^2+\lambda_2\left(\sigma-\sigma'\right)^2 \tag{15}$$

其中 λ_1=50, λ_2= 1000 平滑项的作用是使得 α 的值随着 t 的增加而平滑地变化，以达到图像的边缘逐渐模糊的效果。在使用动态规划算法进行嚴优的参数值计算的时候，通常将 Δ 离散化为 30 个等级，将 σ 离散化为 10 个等级，在计算中我们使用将每一个 Δ 与 σ 都代入到轮廓像素中进行计算，直到最后找到使得能量函数取值最小的 Δ 与 σ 的值并保存。

数据项 D 的计算方法如下：

$$\tilde{D}_{\mathrm{n}}\left(\alpha_n\right)=-\log\mathbf{N}\left(z_{\mathrm{n}};\mu_{t(n)}\left(\alpha_n\right),\Sigma_{t(n)}\left(\alpha_n\right)\right) \tag{16}$$

其中 N 代表高斯概率密度函数，均值和方差分别为：

$$\begin{aligned}\mu_t(\alpha)&=(1-\alpha)\mu_t(0)+\alpha\mu_t(1)\\ \Sigma_t(\alpha)&=(1-\alpha)^2\Sigma_t(0)+\alpha^2\Sigma_t(1)\end{aligned} \tag{17}$$

最后 α 通过 Sigmoid 函数进行计算，其中每个像素使用了其 Δ 与 σ 值，获取到整张图像的 alphaMask 后，将其与原图像进行融合，就得到了 Border Matting 后的结果。

三、 我的工作

在本项目中，我主要实现了 GrabCut 算法的核心步骤和 Border Matting，并完成了简单易用的用户界面，用户界面支持完成 GrabCut 和实现 Border Matting 所需要的基本交互。

用户界面的设计已经在给出的项目框架中完成，交互逻辑与 Open Source Computer Vision Library(OpenCV) 自带的库函数 GrabCut 相同 (如表1)。

表 1: Hot keys

keys	function
ESC	quit the program
r	restore the original image
n	next iteration
left mouse button	set rectangle
CTRL+left mouse button	set GC_BGD pixels
SHIFT+left mouse button	set CG_FGD pixels
CTRL+right mouse button	set GC_PR_BGD pixels
SHIFT+right mouse button	set CG_PR_FGD pixels
b	do Border Matting

3.1 实现 GrabCut

GrabCut 和 GraphCut 同样作为一种依靠能量优化实现图像分割的算法，同样有一个需要优化的 Gibbs 能量函数公式 9，GrabCut 同样以最小化这一能量函数为图像分割目标。

为了实现能量函数的最小化完成图像分割，GrabCut 的具体实现流程分为下方几个步骤。

(1) 定义混合高斯模型

从公式 8，即混合高斯模型公式可以知道，只要确定了三个参数：权重系数 π、均值 、协方差 ，就可以根据当前像素点的 bgr 值确定当前像素属于前景和背景的概率 D(x)，所以在 GMM 类中，定义三个指针，分别表示权重系数，均值和协方差。因为当前像素用 BGR 值表示，所以均值其实为 3 个 double 数，再加上 K=5(5 个单高斯函数组成的多高斯混合函数)，总共 15 个双精度值，而权重系数则为 5 个双精度值，cov 共 3*3*5=45 个双精度值。据此,GMM 的定义为：

```
class GMM
{
public:
  GMM(int componentsCount = 5);// k equal to 5
  void initLearning();
  void addPixel(int compID, const cv::Vec3d color);
  void endLearning();
  static GMM matToGMM(const cv::Mat &model);
  cv::Mat GMMtoMat() const;
  int getComponentsCount() const; //get the number of components
  double operator()(const cv::Vec3d color) const;
  int mostPossibleComponent(const cv::Vec3d color) const;
private:
  std::vector<Component> components;
  std::vector<double> coefs; // Pi_k
  int totalSampleCount;
};
```

而 GMM 每个 component(单高斯模型) 的定义如下：

```
class Component
{
public:
  Component();
  Component(const cv::Mat &modelComponent);
  cv::Mat exportModel() const;
  void initLearning();
  void addPixel(cv::Vec3d color);
  void endLearning(); //end training
  double operator()(const cv::Vec3d &color) const;
  int getCompPixelCount() const;
private:
  cv::Vec3d mean;   // BGR value
  cv::Matx33d cov;  // Multivariate covariance matrix
  cv::Matx33d inverseCov; // Inverse of multivariate covariance matrix
  double covDeterminant; // multivariate covariance determinant
  int totalPiexelCount; //determine the weight
};
```

(2) 初始化 GMM 变量

定义两个变量 bgdGMM 和 fgdGMM 分别表示和前景的混合多高斯模型。首先根据选定的四边形框来初始化 mask 图像，四边形框外的像素是背景，值为 GC_BGD，四边形内的像素可能是前景，值为 GC_PR_FGD。之后，根据 mask 图像，读入样本数据。前景 GMM 的样本数据 (包含可能的前景与确定的前景) 放在变量 fgdSamples 中，背景 GMM 的样本数据 (包含可能的背景与确定的背景) 放入变量 bgdSamples 中。fgdSamples 和 bgdSamples 中存放得都是一些 bgr 颜色值。之后根据 kmeans 聚类算法，计算得到当前像素属于前景或背景混合多高斯变量中的第几个单高斯函数，结果放在 bgdSamples, fgdSamples 中，值为 0-4。

(3) 计算得到权重系数、均值和协方差

权重系数 $\phi_j(\pi_k)$ 为属于某个单高斯函数的采样像素数量除以所有采样像素的数量。每个单高斯函数的均值 μ_j 为所有属于该函数的采样像素颜色和除以属于该函数的颜色采样数量。协方差 Σ_j 的公式则比较复杂。

$$\begin{aligned}\phi_j &:= \frac{1}{m}\sum_{i=1}^{m} w_j^{(i)} \\ \mu_j &:= \frac{\sum_{i=1}^{m} w_j^{(i)} x^{(i)}}{\sum_{i=1}^{m} w_j^{(i)}} \\ \Sigma_j &:= \frac{\sum_{i=1}^{m} w_j^{(i)} \left(x^{(i)}-\mu_j\right)\left(x^{(i)}-\mu_j\right)^T}{\sum_{i=1}^{m} w_j^{(i)}}\end{aligned} \tag{18}$$

得到 GMM 的权重系数、均值和协方差的值后，根据数据项 (Data Term) 的公式，Gibbs 能量公式的第一项可以被表示出来并进行计算，判断一个像素（由 color=（B,G,R）三维 double 型向量来表示）属于这个 GMM 混合高斯模型的概率。

$$\begin{aligned}D\left(\alpha_n, k_n, \underline{\theta}, z_n\right) &= -\log \pi\left(\alpha_n, k_n\right) + \tfrac{1}{2}\log\det\Sigma\left(\alpha_n, k_n\right) \\ &+\tfrac{1}{2}\left[z_n - \propto\left(\alpha_n, k_n\right)\right]^{\top}\Sigma\left(\alpha_n, k_n\right)^{-1}\left[z_n - \propto\left(\alpha_n, k_n\right)\right]\end{aligned} \tag{19}$$

(4) 计算平滑项 V

GrabCut 的 Gibbs 能量函数的第二项，即平滑项和 GraphCut 中的平滑项定义基本相同，若两个邻域像素，norm2 范数差值过大就添加惩罚值，以保证分割出来的图像内部更加平滑。β 是用来调整当图像对比度较高或者较低时，两个邻域像素的差别对整个能量函数的影响程度的，例如在低对比度时，两个邻域像素的差别可能就会比较小，这时候需要乘以一个较大的 beta 来放大这个差别，在高对比度时，则需要缩小本身就比较大的差别，所以我们需要分析整幅图像的对比度来确定参数 β, 如公式 20所示。

$$\begin{aligned}V(\underline{\alpha}, \mathbf{z}) &= \gamma\sum_{(m,n)\in\mathbf{C}}\left[\alpha_n \neq \alpha_m\right]\exp -\beta\left\|z_m - z_n\right\|^2 \\ \beta &= \left(2\left\langle\left(z_m - z_n\right)^2\right\rangle\right)^{-1}\end{aligned} \tag{20}$$

(5) 调用 maxFlow 库进行分割

通过计算得到的能量项构建图，图的顶点为像素点，图的边由两部分构成：每个顶点与 Sink 汇点 t（代表背景）和源点 Source（代表前景）连接的边。根据 mask 来判断是前景后景，分别加点和加边，借助与最小割算法，我们就可以实现对像素点的分类，即图像的分割，使得能量函数最小化公式 21。

$$\min_{\{\alpha_n:n\in T_U\}} \min_{\mathbf{k}} \mathbf{E}(\underline{\alpha}, \mathbf{k}, \underline{\theta}, \mathbf{z}) \tag{21}$$

(6) 总结基本流程

GrabCut 的迭代图像分割：

1. 初始化 GMM 的 components，为每个像素分配确定的单高斯模型：

$$k_n := \arg\min_{k_n} D_n\left(\alpha_n, k_n, \theta, z_n\right)$$

2. 通过图像数据训练混合高斯模型的参数；

$$\underline{\theta} := \arg\min_{\theta} U(\underline{\alpha}, \mathbf{k}, \underline{\theta}, \mathbf{z})$$

3. 进行估计分割，利用 min cut 算法实现：

$$\min_{\{\alpha_n:n\in T_U\}} \min_{\mathbf{k}} \min_{\mathbf{k}} \mathbf{E}(\underline{\alpha}, \mathbf{k}, \underline{\theta}, \mathbf{z})$$

4. 从第一步开始重复，直到收敛。

3.2 实现 Border Matting

通过 GrabCut 完成图像硬分割后，有一个不可避免的问题出现了，由于在实现硬分割时，图像的 alpha 通道数值不为 0 即为 1，这样的设计方式会在很大程度上影响分割后图像的边缘效果，容易有毛刺等边缘不自然不平滑的现象出现。为了解决这一问题，提出了 Border Matting 算法来处理图像边缘轮廓。

Border Matting 算法

(1) 能量函数最小化

为了简化实现的代码，在本项目中没有采用动态规划算法实现能量函数的最小化，而是使用了贪心的策略。在本项目中，根据论文的建议将 Δ 离散化为 30 个等级 (值从 0 到 29)，将 σ 离散化为 10 个等级（值从 0 到 9），在计算中我们使用将每一个可能的 Δ 与 σ 都代入到轮廓像素中进行计算，得到对应于这些参数条件下的能量函数值。直到最后找到使得能量函数取值最小的 Δ 与 σ 的值并保存。代码实现细节如下：

```
double min = INFINITY;
for (int deltalevel = 0; deltalevel < 30; deltalevel++)
{
```

```
 for (int sigmalevel = 0; sigmalevel < 10; sigmalevel++)
 {
  double grayValue =
      valueColor2Gray(Image.at<Vec3b>(contourVector[i].pointInfo.y,
      contourVector[i].pointInfo.x));
  double D = dataTerm(contourVector[i].pointInfo, grayValue, deltalevel,
      sigmalevel, contourVector[i].pointInfo.nearbyInfo);
  for (int j = 0; j < contourVector[i].neighbor.size(); j++)
  {
   point &p = contourVector[i].neighbor[j];
   D += dataTerm(p, valueColor2Gray(Image.at<Vec3b>(p.y, p.x)), deltalevel,
       sigmalevel, p.nearbyInfo);
  }
  double V = lamda1 * (deltalevel - delta)*(deltalevel - delta) + lamda2 *
      (sigma - sigmalevel)*(sigma - sigmalevel);
  if (D + V < min)
  {
   min = D + V;
   contourVector[i].pointInfo.delta = deltalevel;
   contourVector[i].pointInfo.sigma = sigmalevel;
  }
 }
}
```

(2) 计算 alpha 通道值并融合

再完成能量函数最小化后，得到了每个像素的 Δ 与 σ 值，通过 Sigmoid 函数进行计算，获取到整张图像的 alphaMask 后，创建一幅带有 alpha 通道的 RGB 图像矩阵，并将原图的 RGB 值和计算得到的 alpha 通道值存入该图像矩阵中，就完成了原图与消光带融合，得到 Border Matting 后的结果。为图片的边缘添加消光效果。

(3) 遇到的困难和解决方案

在实现 Border Matting 算法时，我也遇到了许多关于数据格式，图像通道方面的问题，但都通过查阅文档等方式解决了，其中最为困扰我的，就是对于不同级的 Δ 与 σ 该如何取值。因为在之前了解到对视差等级的定义的时候，视差的值并非直接等于等级，而是表述为：

$$d_k^0 = d_{\min} + \frac{k}{50} \cdot (d_{\max} - d_{\min}), \qquad k = 0, \ldots, 50 \tag{22}$$

针对此问题，原论文“*GrabCut*”—*Interactive Foreground Extraction using Iterated Graph Cuts* 中也没有给出详尽的表述，但在最后的实现中，为了实现的便捷，还是在代码中直接以等级值对应于的 Δ 与 σ 的值进行应用，最后也取得了一定的效果。

四、 测试与分析

4.1 测试环境

表 2: 测试环境

item	detail
CPU	Intel® Core™ i7-6700HQ CPU 2.60GHz
RAM	16.0GB DDR4 2133MHz
hard disk	SSD 256GB
OS	Windows 10 Pro 64-bit
IDE	Microsoft Visual Studio Community 2017
OpenCV version	2.4.13

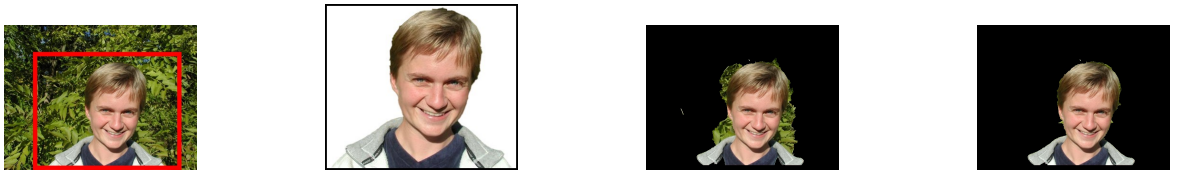
4.2 GrabCut 分析测试

(1) 与论文分割效果进行对比

对于 GrabCut 的图像分割效果进行测试，主要对比了原论文 "*GrabCut*" —*Interactive Foreground Extraction using Iterated Graph Cuts* 中示例图片的分割效果与现实现的 GrabCut 所达到的分割效果。

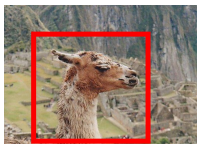
(a) 原图. (b) 论文中分割效果. (c) GrabCut 一次迭代. (d) GrabCut 二次迭代.

图 2: 图像分割效果对比 1

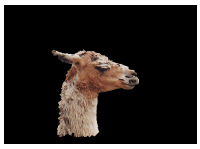


(a) 原图. (b) 论文中分割效果. (c) GrabCut 一次迭代. (d) GrabCut 二次迭代.

图 3: 图像分割效果对比 2


(a) 原图.


(b) 论文中分割效果.


(c) GrabCut 一次迭代.

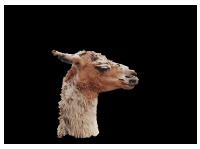

(d) GrabCut 二次迭代.

图 4: 图像分割效果对比 3

通过以上与论文中例图的三组效果对比我们可以看到，经过一次迭代分割的图像效果相对较差，但在经过交互操作二次迭代后，分割效果基本趋于一致。

(2) 与 OpenCV 库函数分割效果和运行时间进行对比

由于 OpenCV 库函数 GrabCut 与我在项目框架中实现的 GrabCut 在进行了两次及以上的迭代过程后效果都趋于论文中的例图效果，故在本次测试中比较的主要为两者均进行一次迭代后的效果。对比结果如下所示：


(a) 原图.


(b) OpenCV 一次迭代.


(c) GrabCut 一次迭代.

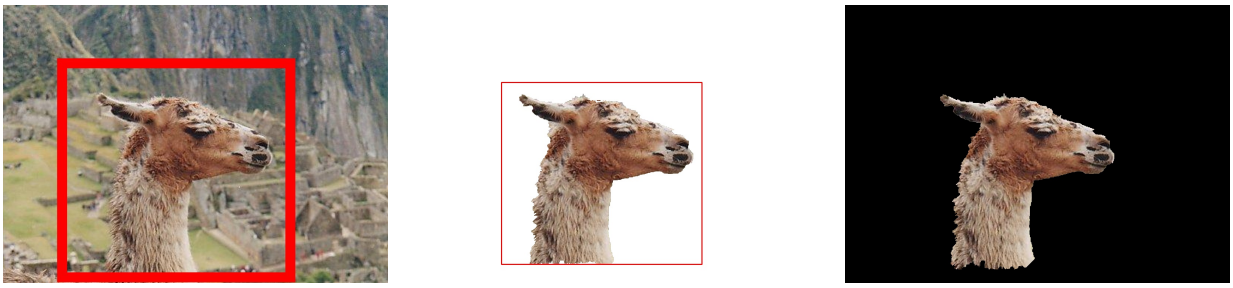
图 5: 图像分割效果对比 4


(a) 原图.


(b) OpenCV 一次迭代.


(c) GrabCut 一次迭代.

图 6: 图像分割效果对比 5



(a) 原图. (b) OpenCV 一次迭代. (c) GrabCut 一次迭代.

图 7: 图像分割效果对比 6

表 3: 运行时间对比表

图片	项目中实现的 GrabCut	OpenCV 库函数 GrabCut
图 5	0.472s	0.494s
图 6	1.750s	1.310s
图 7	6.480s	4.597s

(3) GrabCut 结果分析

在与论文示例图片的比较中我们可以看到本次课程项目中实现的 GrabCut 在第一次进行迭代图像分割时，往往会保留一些边缘处的图像，尤其是在图 3 和图 4 中我们可以明显看到有相当数量的应当属于背景的像素点被保留了下来，同时在图 2 中我们也可以看到花图案内部存在这一些像素点被判定为背景像素的情况。

针对图 2 中可以看到花朵图案内部存在这一些像素点被判定为背景像素这一问题，首先应当是由于花朵上的露珠的颜色信息与花朵本身的颜色信息差异较大这一客观因素。同时这样的特点也使得露珠对应的像素与噪点像素的特性十分接近。通过查阅相关文献资料后，我了解到，在经典的图像分割算法中，为了避免图像的噪点 (或类似于噪点的像素) 对图像分割效果可能产生的不利影响，在真正进行分割之前，往往会选择第一步将图像进行滤波处理。而在本次项目的 GrabCut 图像分割算法的实现中，并没有对原始图像进行任何处理，直接将未经滤波的原图的图像数据作为像素点的分类依据，这必然会带来对于噪点像素 (或类似于噪点的像素) 容易误判的问题。因此，经过分析推测，如果添加滤波步骤，在很大程度上有可能改善这一问题。

而图 3 和图 4 中出现的相当数量的应当属于背景的像素点被保留在前景中这一问题，通过对实现过程的梳理，推测有两个可能的原因：

1. 高斯混合模型在提取前景和背景的特征时，由于确定的背景仅为矩形框外的像素，如果框外的背景像素和框内的背景像素差异很大，可能导致框内的部分背景像素的特征被提取为前景的特征，此时就会导致有大量应当属于背景的像素点被保留在前景中；

2. 还有可能由于在前景中有与背景的颜色或纹理信息十分相近的部分，导致在使用 GMM 判断像素的标签，返回最大分量时，认为该背景像素点更加有可能时前景部分，使之在分割时成为了源点，导致部分的背景被保留了下来。

针对这两个可能的原因进行分析后，推测能利用为 GMM 添加更多的 components 来帮助 GMM 更好地逼近图像像素真实的分布情况，实现对图像更多更细特征的提取，得到更加精确的分类，但我认为最佳的解决方法还是添加用户交互来对一些分割不合理的部分进行人工区分，因为过多的 components 会大大增加资源消耗和资源负担，大大弱化 GrabCut 的实际意义，而用户交互的方式成本要小得多。

在与 OpenCV 库函数中的 GrabCut 进行对比时，我们可以看到运行的时间比较相近，但是就最后的分割结果来看，在第一次迭代时，调用 OpenCV 库函数中的 GrabCut 得到的结果对背景的分离效果更加好，但是也有把部分前景同样分离的问题。针对此情况，我推测有两个可能的原因：

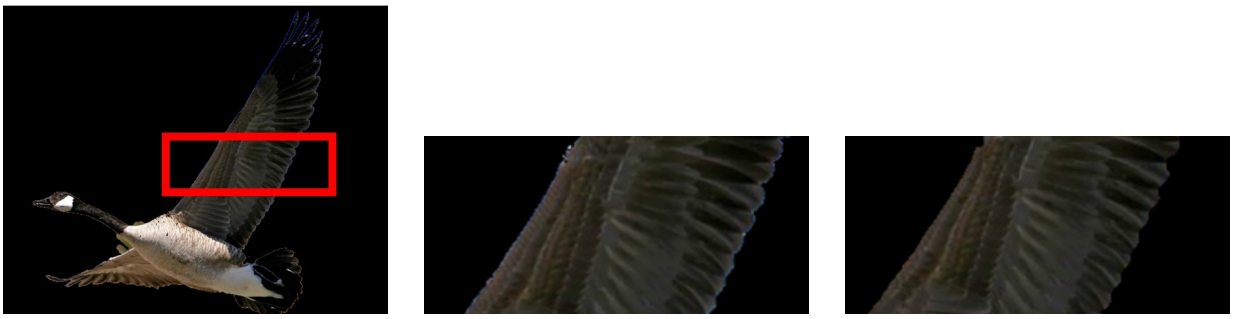
1. 可能矩形框的选择或其他交互操作的细微差别导致最后得到的高斯混合模型不同；
2. 可能混合高斯模型分量的学习停止条件有所差异导致得到的分量不同，进而相同像素的标签可能不同。

4.3 Bordering Matting 测试

(1) Bordering Matting 结果与分析

(a) 原图中的位置. (b) BM 前. (c) BM 后.

图 8: Border Matting 前后效果对比 1



(a) 原图中的位置. (b) BM 前. (c) BM 后.

图 9: Border Matting 前后效果对比 2

通过上面两个例子可以清晰地看到，Border Matting 确实具有一定的消光效果，可以让图像的边缘变得更加自然和平滑，但同时也不难发现，经过 BM 处理后，图像的边缘更加模糊，推测是由于融

合 alpha 通道值，产生的另一方面效果，因为原本的 α 值不为 1，即为 0，而经过消光处理后，边缘的透明度有了变化。

但与此同时，我们也发现对比论文中放大得到的边缘效果，本项目中实现的情况还是与之有所差距。并且对于一些特例的图片，Border Matting 对前景的边缘处理效果并不理想，尤其是对于动物毛发的边缘，容易被直接截去，或弥散为整个色块。

(a) 原图. (b) BM 前. (c) BM 后.

图 10: Border Matting 前后效果对比 3

针对此问题，推测是在能量最小化的实现中有所缺陷，或是通过 Sigmoid 函数进行计算，获取到整张图像的 alphaMask 这一过程存在没有解决的问题。在实现中直接以等级值对应于的进行应用可能也是出现这些问题的一个重要原因。

五、 总结与展望

5.1 总结

在本次实验中，我尝试通过阅读论文，和课程项目中给出的框架说明，实现了了 GrabCut 算法的核心步骤和 Border Matting，使之能够完整简单的图像前景/背景分割任务。

这是我第一次尝试以阅读论文，并复现的方式来学习一种图像处理算法，在项目框架中详尽的指导，和论文清晰的理论步骤引领下，通过实现 GrabCut 的过程，帮助我对交互式图像分割与抠像有了更加深入的理解，通过本次实验，我也对 GMM 提取多维特征，利用最小割算法解决能量最小化问题等方面有了真正明晰的理解。没有了详尽的文档说明的引领，实现 Border Matting 时，我遇到了不少困难，尽管最终实现的边缘处理效果还是不够理想，但通过不断发现问题，解决问题的过程，我也是获益匪浅。

5.2 进一步改进的展望

在实现本次项目的过程中，我也发现了一些可以改进的要点，可以作为进一步工作的方向：

1. 在对图像进行 GrabCut 分割算法之前，可以先对图像进行滤波，避免背景中一些零星的颜色与前景相近的像素被判断为前景，也可以减少图像噪点对分割可能的影响。
2. 可以进一步理解 Border Matting 的过程，优化能量最小化过程的实现方法，进一步探索 Δ 与 σ 的取值方法来提高边缘处理效果。

参考文献

[1] ROTHER, C., KOLMOGOROV, V., AND BLAKE, A. Grabcut- interactive foreground extraction using iterated graph cut. In Proc. of ACM SIGGRAPH, 2004,309-314.

[2] BOYKOV, Y., AND JOLLY, M.-P. Interactive graph cuts for optimal boundary and region segmentation of objects in n-d images. In ICCV, 2001, 105-112.

分工情况：单人完成

成员：3170102492 夏豪诚