****

**计算机学院暑期实习报告**

题 目：3D食品打印算法与优化

姓名 ：

指导教师 ：

日期：2016/7/30

# 1 编程环境

操作系统：Microsoft Windows 10

开发平台：Visual Studio 2010

开发语言：C++

辅助工具：Opencv 2.1, Face++

# 2问题背景

将消费者的照片通过3D食品打印成为可口而充满艺术性的食品（如（d）所示），是一个充满个性和趣味的创意。

(a) (b) (c) (d)

图1 3D食品打印

如上图所示，（a-b）为3D食品打印的结果。而由于用于食品打印的材料为相应食材，和普通3D打印机采用的abs、sla等材料在制作技术、加热温度、凝固条件上均有较大不同，因此需要结合需求进行针对性的改进。如（c）所示，使用糖、巧克力或者奶油等材料进行打印，其线条宽度较大，反复堆叠进行多层打印容易走形，效果较差。我们目标实现的效果即为图（d）所示的糖画。整体构型精巧充满艺术性并且可以粘在竹签上完整的取下来。因此为了良好的成型效果因此需要单层打印并避免路径的反复交叉引起外观的变化。此外，为了打印后能够完成的取下不断裂，形状中需要彼此相连不断裂。因此，若需要打印个人照片，需要对以下两方面进行处理：

1、需要简化图片的内容，因为食品打印难以实现丰富的细节，一般采用简单的线来表示形状。因此需要将图片内容进行处理，保持其主体内容的同时将图片内容简化为简单的线表示。

2、做到线条相互连通一笔完成的前提下，需要对于打印路径进行规划。3D食品打印使用的打印材料为相应食材，比如糖浆巧克力等。由于食材在加热温度，凝固条件，难以凝固成型效果上都与较常用的abs或者sla有较大差距，其路径规划也不相同。需要在

路径之间流出更多间隙。单层打印效果较好，若路径存在交叉则在交叉处的厚度会显出增加，影响美观。故需要尽量避免路径交叉。

# 3 概述

通过图像中的线条、颜色、阴影等，一幅图像可以表达非常丰富的信息。本文利用的Coherent Line Drawing技术就是从图像中提取出连续的、平滑的风格化线条，来捕捉和表达图像中的形状和轮廓信息。

为了达到提取轮廓的目的，首先想到的是边缘检测（Edge Detection）的一系列算法，但是边缘检测算法并不能完全满足我们的要求，例如常用的Canny边缘检测算法其结果图离散线条粗糙繁杂且无法准确提取人物面部特征。

Henry Kant et al.在2007年提出了一种Coherent Line Drawing[1]技术，该技术将问题分为两步，首先构造图像的边缘切向流（ETF），再沿着流向作DoG（Flow-based DoG），进而提取轮廓。

脸部轮廓3D食品打印的两个关键点：

* 保持其面部特征轮廓的同时简化图片细节。

利用face++提供的API进行脸部特征点提取，可得到83个包含脸部轮廓，眉毛，眼睛，鼻子，嘴巴的特征点。在Coherent Line Drawing算法前做预处理，对脸部轮廓特征点线性插值并在灰度图中增强脸部轮廓的灰度值。在算法后采取八连通深度搜索，设定线条像素点阈值对线条筛选删除。保持面部特征轮廓的同时可简化图片细节。

* 线条相互连通一笔画的前提下，对线条细化，抽取骨架对打印路径进行规划。

预先采用八连通深度搜索对连通域进行标记，依次枚举两两连通域间的最短距离，利用最小生成树prim算法进行连通域互连，达到互连边的权值之和最小。经多次测试，此做法连通后对原图片的破坏程度较小，可达到连通与美观的平衡。最后采用形态学的抽取骨架方法，将线条细化为一个像素点等宽。经深度遍历记录连通域内各点即可得到一个带方向的路径，设置刷新频率则能动态画出图片中的人物肖像。

关键词：Coherent Line Drawing，边缘检测，最小生成树，抽取骨架

# 4 算法描述

概述中提到的Henry Kang提出的算法。如图2所示，算法分为两大步骤，第一步是构造参考图像的平滑方向场，称为ETF（Edge Tangent Flow），第二步再对ETF进行基于方向场流向的各向异性高斯滤波，称为FDoG（Flow-based Difference-of-Gaussians）根据阈值进行二值化后即得参考图像的线绘图。

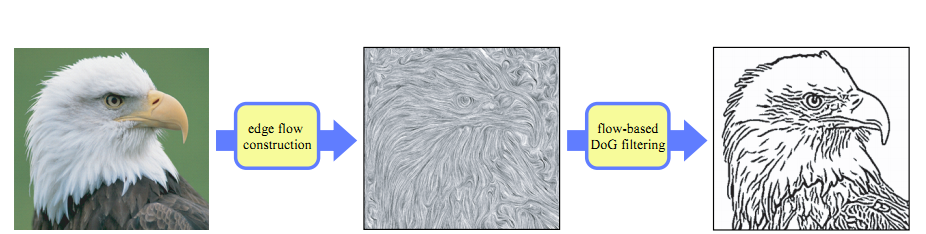


图2 ETF，FDoG效果图

梯度方向是图像变化最剧烈的方向，与其垂直的方向可看做局部边缘的切向，即“流向”。算法首先使用Sobel算子得到参考图像的梯度场，然后在每一像素x处计算与其梯度向量垂直的向量，得到了初始流场。再迭代地对流场做2~3次非线性平滑，使得重要的边缘方向被保留，而次要的方向与相关的重要方向对齐，这样就能得到更平滑的方向场。ETF的滤波核函数如下：

 (1)  
 (2)

 (3)

 (4)

其中，为空间距离权值，控制梯度值之差的影响，y与x的梯度值相差越大，这一权值也越高，控制方向的影响，梯度向量越趋近于平行这一权值越大，越趋近于垂直则越小。

本算法采用基于方向场流向的DoG（FDoG）来生成线绘图，其滤波核如图3所示。

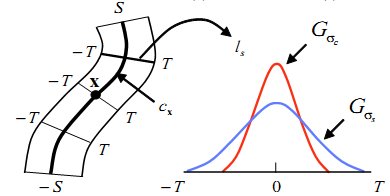


图3滤波核

从参考图像生成线条的过程与边缘检测有很大的相似之处，因此许多边缘检测的方法可以应用到算法中来，这一算法就采用了Winnemoller et al.的DoG方法[2]来得到边缘的轮廓线条。对于DoG的原理，我们是这样理解的：如果对图像作二阶微分，即施加Laplacian算子，那么边缘就出在穿0点处，而Laplacian对于噪声十分敏感，在施加Laplacian操作前，先要进行高斯滤波，即Laplacian of Gaussian（LoG），而这里所使用的DoG则是对LoG的简化和近似。

如图3所示，在每一点处，沿着图像变化最剧烈的方向，即梯度方向，施加一维的DoG：

 (5)

 (6)

的大小由决定，为的1.6倍，T的值也由决定，因此，是一个非常关键的参数，它控制着线宽。

曲线S代表的是方向场的流向，为了得到更连续的线条，需要再沿着S进行一次高斯卷积：

 (7)

也是一个非常关键的参数，控制线条的连续性。

最后，再根据设定的阈值对图像进行二值化，就得到了最后的结果：

 (8)

# 5 算法实现

**1、数据结构**

（1）ETF 类

class ETF

{

private:

char\* gray;//参考图像灰度图

double\* gradient;//梯度场

double\* gradientMag;//梯度值

double\* tangent;//方向流场

double max\_grad;//最大梯度值

int width;

int height;

int wideStep;

int iterations;//迭代次数

int radius;//滤波核大小

public:

~ETF();

void Init(char\* img, double\* tan, int w, int h, int it, int r);//初始化参数

void ETFTransfer();//构造ETF

private:

void GetTangent();//计算初始ETF

void RefineTangent();//计算平滑ETF

void RefineTangent2();//不同的实现方式计算平滑ETF

inline double DistanceSquare(int x1, int y1, int x2, int y2);

inline double Descartes(double x1, double y1, double x2, double y2);

inline double Norm(double x, double y);

inline void SobelFilter(double ul, double um, double ur, double ml, double mr, double ll, double lm, double lr, double\* x, double\* y, double\* mag, double\* max\_grad);//Sobel算子

inline void Normalize(double\* x, double\* y);

};

（2）FDoG类

class FDoG

{

private:

char\* image;//参考图像

char\* result;//结果图像

double\* tangent;//ETF

int width;

int height;

int wideStep;

double sigmaC;//控制线宽

double sigmaS;

double sigmaM;//控制线条连续性

double noise;

double threshold;

int deleteline;//控制简化程度

int extend;//控制脸部轮廓清晰度

public:

~FDoG();

void Init(char\* img, char\* res, double\* tan, int w, int h, double sc, double sm, double p, double t, int d,int e);//初始化参数

void FDoGTransfer();//进行FDoG操作

void DecreaseLine();//进行去除细节内容操作

void DFS(int i,int j);//阈值遍历

void \_DFS(int i,int j);//删除线条

void kDFS(int i,int j);//面部轮廓特征点标记

void DFS\_(int i,int j);//连通域标记

void prim(int u0)//最小生成树

void CountEdge();//求得连通域间最小距离

private:

double\* MakeGaussians(int\* n, double sigma);//根据sigma生成一维高斯

inline double Gaussian(double x, double mean, double sigma);

inline int round(double x);

};

（3）Parameters结构

struct Parameters//算法所需所有输入参数

{

int smooth\_times;

int iteration\_times;

int radius;

double sigmaC;

double sigmaM;

double noise;

double threshold;

int deleteline;

int extend;

};

（4）DrawLine类

class DrawLine

{

private:

IplImage\* src\_img;//参考图像

IplImage\* res\_img;//结果图像

unsigned int width;

unsigned int height;

double\* tangent;//方向流场

struct Parameters param;//参数

public:

DrawLine();

~DrawLine();

void SetParameters(int st, int it, int rr, double sc, double sm, double pp, double tt,int d,int e);//设定参数

void SetSrcImg(IplImage\* img);//设定源图像

IplImage\* GetSrcImg();//返回源图像

double\* GetETFImg();//返回ETF

IplImage\* GetResultImg();//返回结果图像

void TransferImage();//进行Coherent Line Drawing

};

**2、关键函数说明**

（1）class DrawLine::TransferImage()

在这一函数中进行Coherent Line Drawing的两个主要步骤。首先将输入的源图像转换为灰度图。然后声明一个ETF类的对象，调用ETF的Init函数将图像数据和参数传入，再调用ETF的ETFTransfer函数进行ETF的构造。再声明FDoG类对象，调用其Init函数设置结果指针和参数，并调用FDoGTransfer进行FDoG操作，最后调用DecreaseLine进行简化操作，block和prim进行连通域标记和最小生成树的构造。

（2）class ETF::GetTangent()

该函数在每一点处调用inline void SobelFilter(double ul, double um, double ur, double ml, double mr, double ll, double lm, double lr, double\* x, double\* y, double\* mag, double\* max\_grad);，使用Sobel算子计算参考图像的梯度场和归一化的梯度值，再在每一点处，求出与该点梯度向量垂直的流场方向向量。其中，图像边缘处的值取与其紧靠的内侧行或列的值，四个角点处则取平均。

（3）class ETF::RefineTangent()

在该函数中迭代地根据算法定义的核函数对ETF进行平滑

（4）class FDoG::FDoGTransfer()

在该函数中分两步实现FDoG。首先在每一点处沿着该点的梯度方向作一维DoG，并将结果保存，然后从该点出发，沿着方向流场向正反两个方向遍历，并对经过的所有点的DoG作高斯卷积。S和T的大小是由和决定的，其做法是在double\* MakeGaussians(int\* n, double sigma);中使用sigma作为参数，且x从0开始递增地调用inline double Gaussian(double x, double mean, double sigma);计算高斯函数，直到时高斯函数值小于等于某个阈值。然后分配一个大小为的数组，存放所有大于该阈值的高斯函数值，这样不仅决定了卷积核的大小，同时也求出了在每一点处应乘上的高斯函数值。

（5）class FDoG::void DecreaseLine();

该函数分两部分，一部分为调用kDFS()函数进行Face++脸部轮廓特征点的标记，一部分为调用DFS()函数进行连通线条像素点统计，小于阈值并且不是含有特征点的线条通过调用\_DFS()函数进行删除。

（6）class FDoG:: void CountEdge()

该函数第一步调用DFS\_进行连通域标记并统计连通域个数，第二步构造图论模型，分别对任意两个连通线条枚举各自的像素点，取其最近的两点间距离

（7）class FDoG:: void prim(int u0)

在block()函数求得各连通域的最小距离后。该函数即朴素的prim最小生成树算法，将各连通域连通，形成一棵树。其中连接两个连通域的直线算法采用了任意斜率的bresenham算法。

（8）Mat thinImage(const cv::Mat & src, const int maxIterations = -1)

细化线条为一个像素点等宽。src为输入图像,用cvThreshold函数处理过的8位灰度图像格式，元素中只有0与1，1代表有元素，0代表为空白

参数说明：

参数Smooth Times控制高斯平滑的次数，对于受噪声影响的图像，如Lena有明显的改善效果；

参数Iteration Times为ETF迭代次数，一般控制在2~3次为宜，保持效果和计算速度的平衡；

参数Neighbor Radius为Box Filter半径大小，一般保持为5即可；

参数Sigma C控制线宽，该值越大，线条越粗；

参数Sigma M控制连续性，该值越大，连续性越好；

参数Noise Level为噪声等级，结果的质量对该值的变化非常敏感，一般设在0.99为宜；

参数Threshold为进行二值化时的阈值，该值越大线条连续性越强

参数deleteline为进行删除线条时的阈值，该值越大图片细节越少，一般设置为300-500为宜；

参数 extend为标记脸部特征点的阈值，由于Face++得到的特征点不一定落在目标轮廓区域内，所以要采取范围扩展，涵盖面部轮廓区域。该值越大删除操作后留下的面部轮廓越清晰，一般取20-30为宜。

图4 Coherent Line Drawing效果

**脸部关键点提取**

通过调用Face++[5]的API获得目标图片的83个人脸关键点。

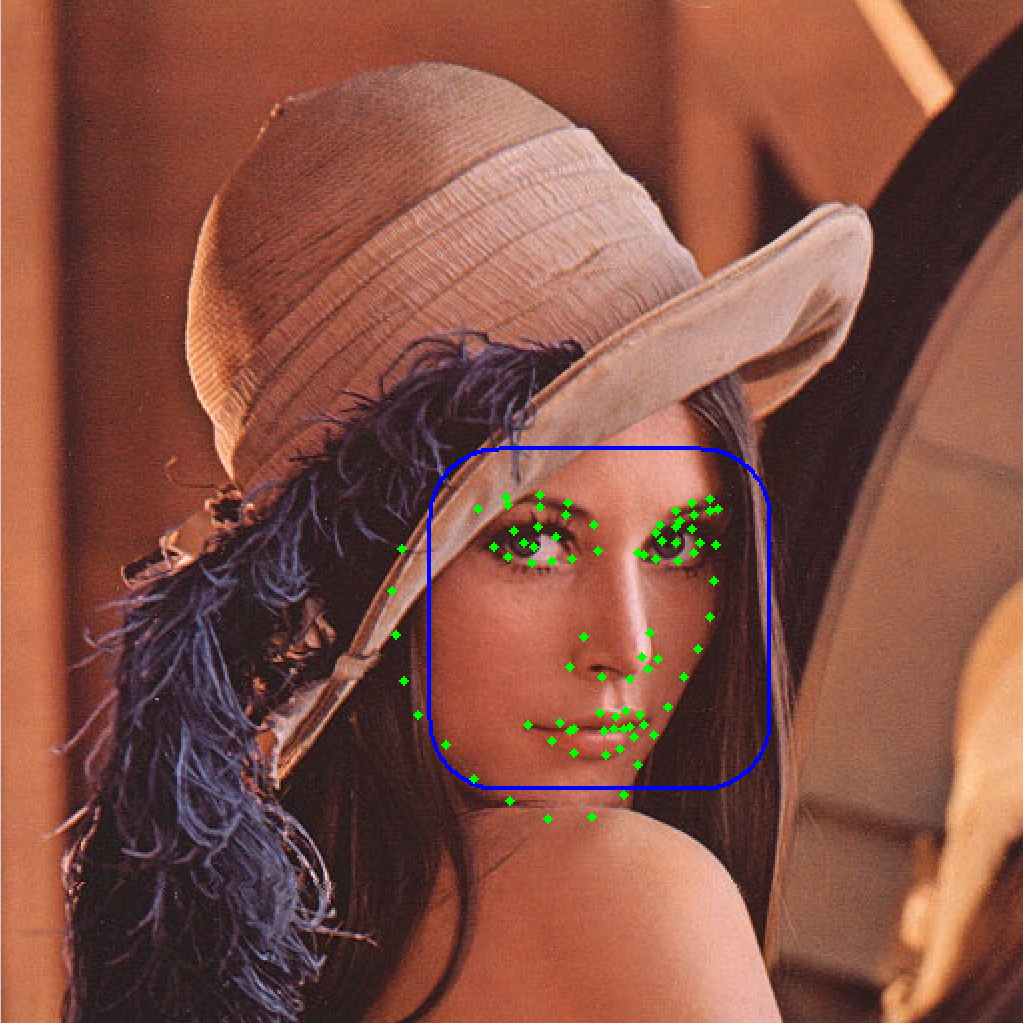


图5 Face++脸部轮廓关键点提取

脸部轮廓特征点的生成顺序为：第一个点为下巴，接着是9个由上到下的左侧脸部轮廓关键点和9个由上到下的右侧脸部轮廓关键点，利用线性插值，其中直线画法采用任意斜率的Bresenham算法[4]可以得到脸部轮廓的线条。如下图所示：

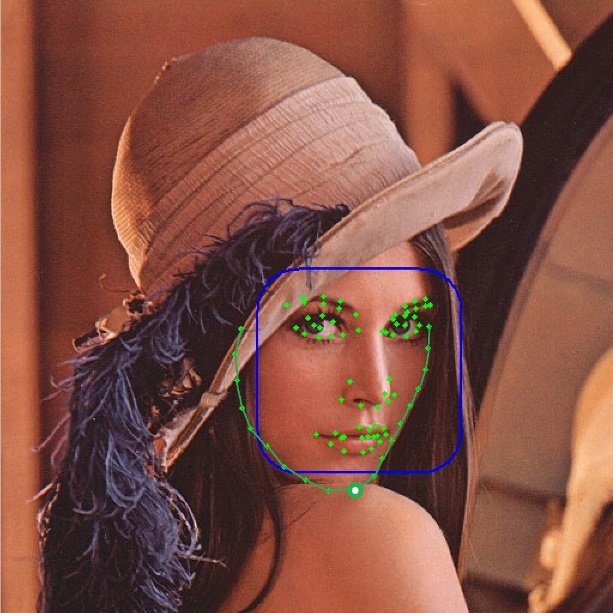


图6 Face++脸部轮廓关键点插值

为了得到更为清晰的脸部轮廓线条，在ETF，FDoG处理前彩色图转灰度图的步骤后预处理，加强Face++人脸关键点插值后得到的脸部轮廓的灰度值，提高其梯度值，便于ETF，FDoG算法提取轮廓。

**简化图片内容：**

通过ETF，FDoG算法得到二值图，预先标记含有Face++人脸关键点的连通线条，然后对其二值图中每一个像素点避开标记的人脸关键点线条采取八连通深度搜索遍历，统计每一个连通线条的像素点的个数，并设置阈值(算法中取300)，删除像素点数低于300的连通线条，从而简化图片。如图7：

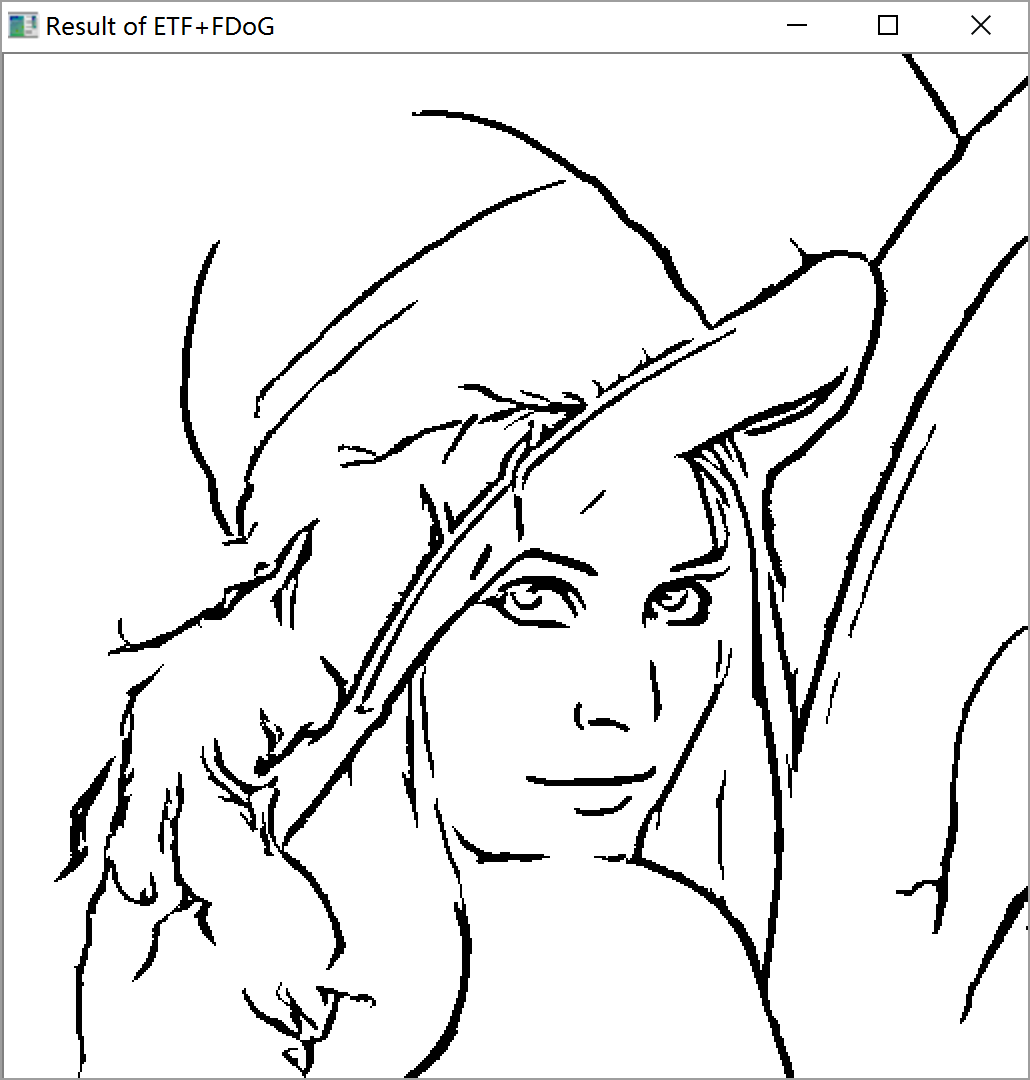
 

图7 简化图片

**线条相互连通形成一笔画：**

预先采取八连通深度搜索遍历对连通域进行标记，将各个离散的连通线条抽象为一个点，进而可将问题抽象为一个图论问题，依次枚举两两连通域间的最短距离，并记录达到最短距离时对应的两点坐标。此时连通域间的边已求出，最小生成树prim算法进行距离最近的两个连通域互连，达到互连边的权值之和最小。经多次测试，此做法连通后对原图片的破坏程度较小，做到了连通与美观的平衡。

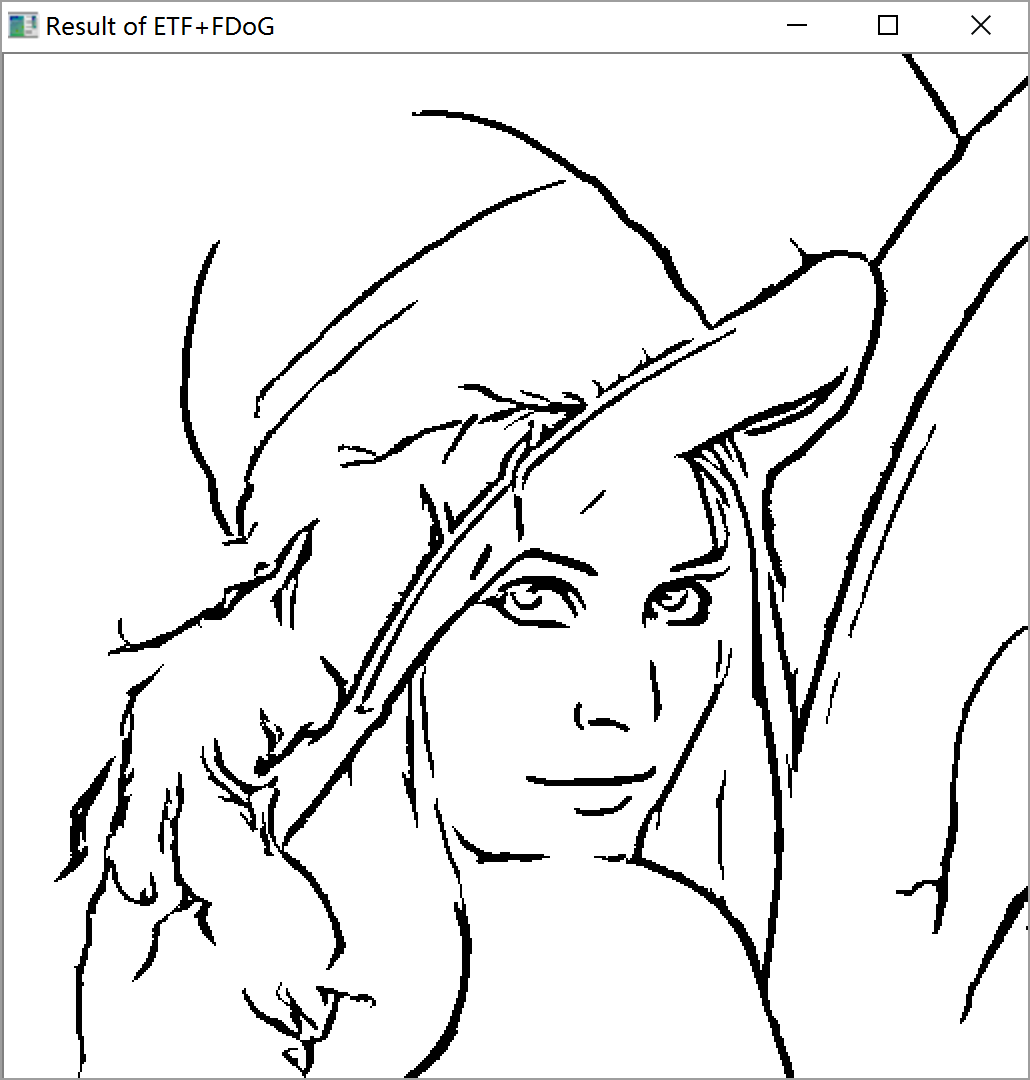
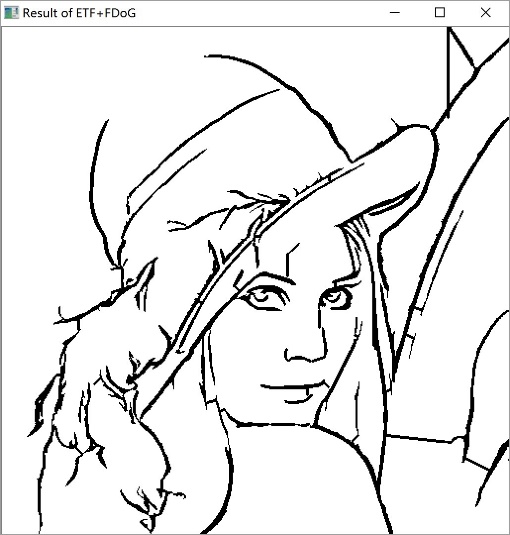
 

图8 一笔画

本用例共35个连通域，连通后成为一个整体。

**线条细化抽取骨架：**

采用形态学的抽取骨架方法，不断腐蚀将线条细化为一个像素点等宽。经深度遍历记录连通域内各点即可得到一个带方向的路径。

图9 线条细化抽取骨架

设置点的半径大小及刷新频率，将路径上的点重绘则能动态画出图片中的人物肖像。见demo.mp4

图10 动态重绘图

# 6 算法评价

**优点及展望：**

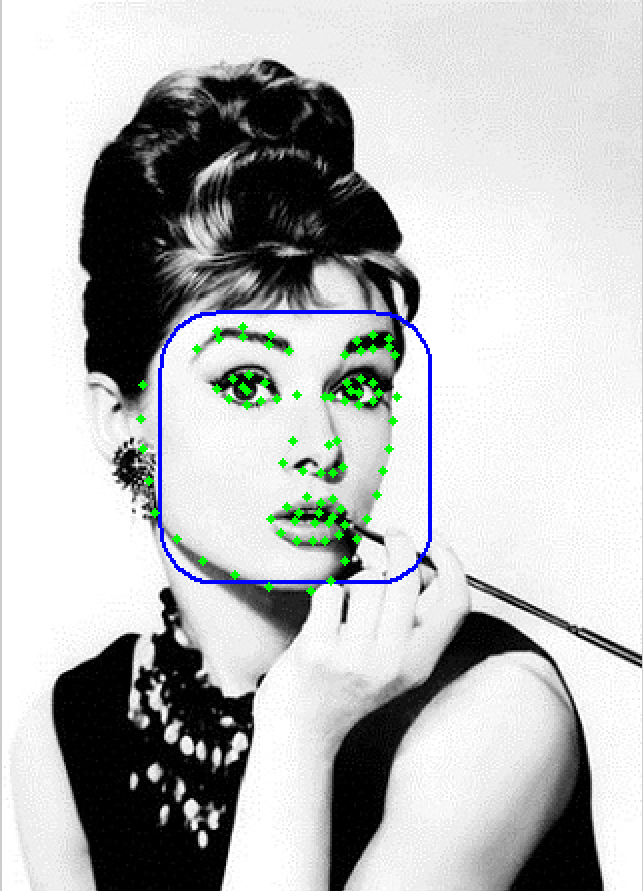
本次实习基本实现了照片轮廓提取，保留面部特征轮廓的同时简化细节，图片连通域互连为一笔画可实现重要区域线条填补，线条细化抽取骨架，深度搜索顺序动态画出照片中人物肖像。

**缺点及改进：**

([Pedersen 2006](#_ENREF_16))在2006年提出将图片抽取为迷宫的方法，为我们设计打印机打印路径提供了新思路，由于3D食品打印技术特点，需要尽量避免打印路径的交叉和重合。也可以用来实现对于风景照片或者肖像的路径设计。

由于实习时间有限，本算法需要一定的人为参数阈值设定。原计划采用AAM Library或ASM算法来获取人物面部轮廓特征点，但效果并不理想。离散连通域连通的过程并没有做到首尾无重叠路径无交叉的连接有待进一步改进。

其他测试用例：

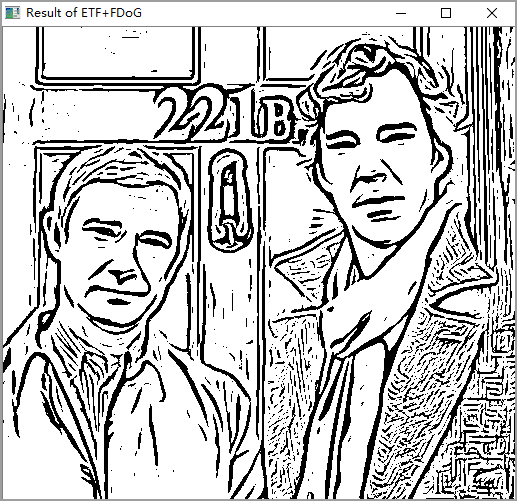
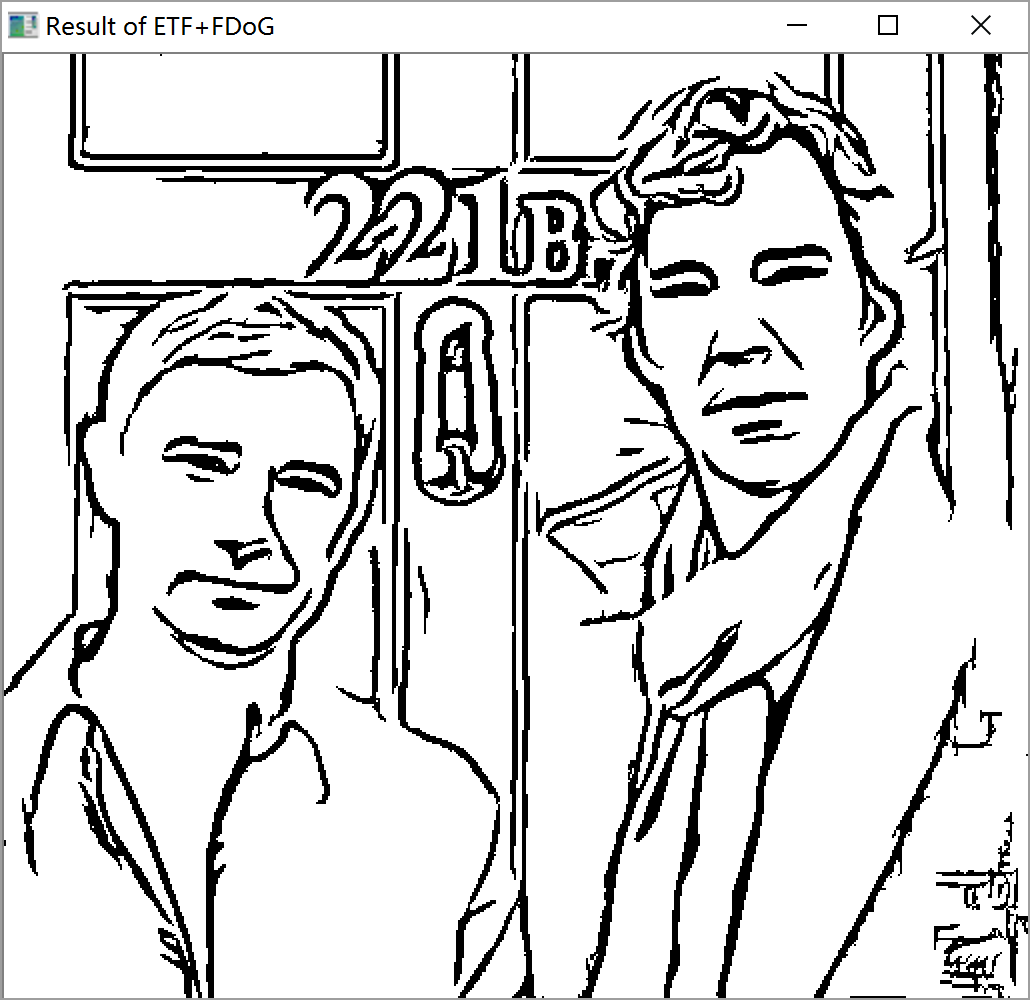
 

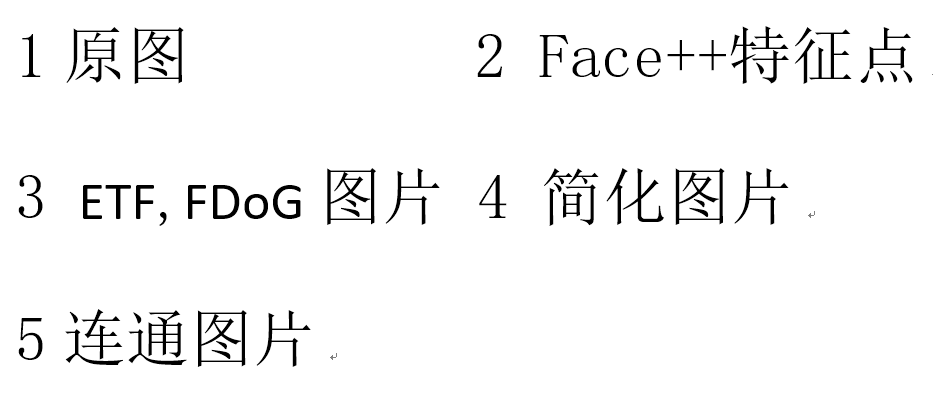
 



双人用例：

# 7 参考文献：

[1] H. Kang, S. Lee, and C.K. Chui, "Coherent line drawing," Proc. Non-Photorealistic Animation and Rendering (NPAR '07), pp. 43-50, Aug. 2007.

[2] Winnemoller. H, Olsen. S, and Gooch. B, 2006. "Real time video abstraction," Proc. ACM SIGGRAPH 06, 1221-1226.

[3](美)Gary Rost Bradski 学习OpenCV[M].于仕琪译.北京:清华大学出版社,2009:165-175

[4]孙家广,胡事民.计算机图形学基础教程(第2版)[M].北京：清华大学出版社,2012

[5]<http://www.faceplusplus.com.cn/demo-landmark/>