Computer Vision HW3

612415013 蕭宥羽

1. **程式碼及方法**

#include <opencv2/opencv.hpp>

#include <vector>

#include <iostream>

#include <string>

#include <filesystem>

**Include library**

cv::Mat image,background,original;

std::vector<cv::Point> points; //point容器保存點擊事件的座標點

**設置圖片及容器points，用來存放圖片及點擊事件的座標點**

void onMouse(int event, int x, int y, int flags, void\* param){

  if(event==cv::EVENT\_LBUTTONDOWN){

    points.push\_back(cv::Point(x,y));

    cv::circle(original,cv::Point(x,y),3,cv::Scalar(0,255,0),-1);

    std::cout<<"["<<x<<","<<y<<"]"<<std::endl;

    cv::imshow("Image",original);

  }

}

**將點擊位置的座標存放到容器當中，並將座標顯示到終端機上**

void drawPolygo(){

    background.copyTo(original);

    cv::Mat mask = cv::Mat::zeros(original.size(), CV\_8UC1);

    std::vector<std::vector<cv::Point>> pts = { points };

    cv::fillPoly(mask, pts, cv::Scalar(255));

    cv::polylines(original, pts, true, cv::Scalar(255, 0, 0), 2);

    cv::imshow("Image",original);

}

**將原始導入的圖片original複製到background中，並將點擊的點連起來**

void shrinkPolygon(cv::Mat& grad\_x,cv::Mat& grad\_y,double ALPHA,double BETA,double GAMMA){

  //定義搜索區域大小

  const int searchRegionSize=1;

  //定義能量函數的權重

  const double alpha=ALPHA;

  const double beta=BETA;

  const double gamma=GAMMA;

  for(int i=0;i<points.size();i++){

    double minEnergy = INT\_MAX;// 初始化最小能量

    cv::Point point\_minus\_1 = (i==0)?points[points.size()-1]:points[i-1]; //p\_i-1

    cv::Point point\_plus\_1 = (i==points.size()-1)?points[0]:points[i+1];  //p\_i+1

    cv::Point newPoint;

      for(int x=-searchRegionSize;x<=searchRegionSize;x++){

        for(int y=-searchRegionSize;y<=searchRegionSize;y++){

        cv::Point point\_s = cv::Point(points[i].x+x,points[i].y+y);   //p\_i+s

        cv::Point g\_point\_s;  //p\_i+s 遍歷點的梯度

        g\_point\_s.x=grad\_x.at<double>(point\_s.y,point\_s.x);

        g\_point\_s.y=grad\_y.at<double>(point\_s.y,point\_s.x);

        double E\_cont = cv::norm(point\_s-point\_minus\_1);

        double E\_curv = cv::norm(point\_minus\_1-2\*point\_s+point\_plus\_1);

        double E\_img = cv::norm(g\_point\_s);

        double totalEnergy=alpha\*E\_cont+beta\*E\_curv+gamma\*E\_img;

        if(totalEnergy<minEnergy){

          minEnergy=totalEnergy;

          newPoint=point\_s;

        }

      }

    }

    points[i]=newPoint;

    cv::circle(original,cv::Point(points[i].x,points[i].y),3,cv::Scalar(0,255,0),-1); //繪製新的點

  }

}

**grad\_x grad\_y為圖像的梯度、ALPHA BETA GAMMA 能量函數的權重、searchRegionSize 搜索**

**區域的大小、double E\_cont = cv::norm(point\_s - point\_minus\_1);: 表示連續性項，即相鄰頂點之間的距離、double E\_curv = cv::norm(point\_minus\_1 - 2 \* point\_s + point\_plus\_1);:表示曲率項，即頂點附近的曲率。double E\_img = cv::norm(g\_point\_s);: 表示圖像梯度項。minEnergy設為一個很大的數，totalEnergy=alpha\*E\_cont+beta\*E\_curv+gamma\*E\_img計算區域中的能量，如果totalEnergay小於minEnergy則minEnergy就會被更新，用這種方法來算出區域中最小的能量，然後將點更新到能量最小的地方，便可以完成收縮。**

cv::VideoWriter createVideoWriter(const std::string& filename, const cv::Size& frameSize, double fps) {

    int fourcc = cv::VideoWriter::fourcc('X', '2', '6', '4'); // 使用X264編碼

    cv::VideoWriter writer(filename, fourcc, fps, frameSize);

    if (!writer.isOpened()) {

        std::cerr << "Error: Unable to open video file for writing." << std::endl;

        exit(EXIT\_FAILURE);

    }

    return writer;

}

void saveFramesToVideo(const std::string& imgname, const std::vector<cv::Mat>& frames, double fps) {

  std::filesystem::path imgPath(imgname);

  std::string videoFilename="../result/result"+imgPath.stem().string()+"\_video.avi";

  cv::VideoWriter writer = createVideoWriter(videoFilename, frames[0].size(), fps);

  for (const auto& frame : frames) {

      writer.write(frame);

  }

  writer.release();

}

**將過程存成.avi的影片**

void saveImage(const std::string& imgname,const cv::Mat& img){

  std::filesystem::path imgPath(imgname);

  std::string imageFilename="../result/result"+imgPath.stem().string()+"\_image.jpg";

  cv::imwrite(imageFilename,img);

}

**存圖片的function**

void pic(std::string imgname,double ALPHA,double BETA,double GAMMA){

  original = cv::imread(imgname);

  cv::resize(original,original,cv::Size(original.cols/2,original.rows/2),0,0);

  background = original.clone();

  cv::cvtColor(original,image,cv::COLOR\_RGB2GRAY);

  cv::GaussianBlur(image,image,cv::Size(1,1),0,0);

  cv::Mat grad\_x,grad\_y;

  cv::Sobel(image,grad\_x,CV\_64F,1,0); //計算x方向梯度

  cv::Sobel(image,grad\_y,CV\_64F,0,1); //計算y方向梯度

  cv::imshow("Image",original);

  cv::setMouseCallback("Image",onMouse,0);  //觸發點擊事件

  cv::waitKey(0);

  //add a vector to save the image of each frame

  std::vector<cv::Mat> frames;

  //add the initial image to frames

  frames.push\_back(original.clone());

  if (points.size() >= 3){

    for(int i=0;i<100;i++){

      drawPolygo();

      frames.push\_back(original.clone());

      cv::waitKey(3);

      shrinkPolygon(grad\_x,grad\_y,ALPHA,BETA,GAMMA);

    }

  }

  saveImage(imgname,original);

  saveFramesToVideo(imgname,frames,30.0);

  points.clear();

}

**把圖片先灰階接著做高斯模糊，然後計算出圖片的梯度，呼叫前面所寫的function，實現繪圖**

**及收縮的功能，並將影片及圖片存起來**

int main(int argc, char\* argv[]) {

  pic("../test\_img/1.jpg",0.2,0.1,0.7);

  pic("../test\_img/3.jpg",0.2,0.1,0.7);

  pic("../test\_img/5.jpg",0.2,0.1,0.7);

  return 0;

}

**輸入不同圖片，**0.2,0.1,0.7**這個組合這三張照片都有不錯的**

**二結果**

|  |  |
| --- | --- |
| Image1 | Result1 |
| Image3 | Result3 |
| Image5 | Result5 |

**觀察結果:像是的電風扇這張圖的倒影會影響邊緣的偵測，所以結果沒有那麼好。**