**基于OpenCV2.0的光流算法跟踪实现**

**一、原理**

在OpenCV中使用光流算法主要用到calcOpticalFlowPyrLK函数，该函数输入前后两帧图像和输入的跟踪点计算出在新一帧图像中跟踪点的新位置。对每一帧图像进行上述计算，即可对运动物体进行跟踪。各帧之间进行跟踪，难免有跟踪失败的情况，因及时添加特征点，检测特征点的函数使用goodFeaturesToTrack函数。

**二、实现**

#include "stdafx.h"

#include <opencv2\opencv.hpp>

#include <iostream>

using namespace std;

using namespace cv;

int \_tmain(int argc, \_TCHAR\* argv[])

{

VideoCapture cap(0);

Mat frame,image,curgGray,prevGray;

vector<Point2f> trackingPoints[2];

bool pointTrackingFlag = false;

while (true)

{

cap >> frame;

if (frame.empty())

{

break;

}

cvtColor(frame, curgGray, CV\_BGR2GRAY);

if (trackingPoints[0].size() > 20 )

{

vector<uchar> statusVector;

vector<float> errorVector;

if (prevGray.empty())

{

curgGray.copyTo(prevGray);

}

calcOpticalFlowPyrLK(

prevGray, //输入，前一帧图像

curgGray, //输入，当前帧图像

trackingPoints[0], //输入，前一帧跟踪点

trackingPoints[1], //输出，跟踪到的点

statusVector, //成功标志，成功为1，失败为0

errorVector //跟踪误差

);

int count = 0;

int minDist = 2;

//对跟踪到的点进行筛选

for (int i = 0; i < trackingPoints[1].size(); i++)

{

//如果两个点位置相差太小，认为没有移动

if (norm(trackingPoints[0][i] - trackingPoints[1][i]) < minDist)

{

continue;

}

//如果跟踪失败，放弃该点

if (!statusVector[i])

{

continue;

}

//保存筛选后的点

trackingPoints[1][count++] = trackingPoints[1][i];

//绘制跟踪结果

int radius = 8;

int thickness = 2;

int lineType = 8;

circle(frame, trackingPoints[1][i], radius, Scalar(0, 255, 0),

thickness, lineType);

}

trackingPoints[1].resize(count);

std::swap(trackingPoints[0], trackingPoints[1]);

std::swap(prevGray, curgGray);

trackingPoints[1].clear();

imshow("TRACKING", frame);

waitKey(1);

}

else

{

goodFeaturesToTrack(

curgGray, //输入，灰度图像

trackingPoints[0], //检测到的特征点

100, //最大特征点数量

0.02, //质量等级

10 //特征点间最小距离

);

}

}

return 0;

}

