阿洲的程式教學
關於Qt、OpenCV、影像處理演算法

# 相機校正(Camera calibration)

這邊介紹如何使用OpenCV提供的函式，對照相機進行校正。校正時我們必須先準備圖表，並用照相機在不同視角和距離對這個圖表拍照，使用時大約拍攝10到20張，以這些影像當作我們的輸入資料，目前OpenCV的支持三種類型的校準圖表，分別是：1.經典的黑白色棋盤，2.對稱圓形圖案，3.非對稱圓形圖案。

OpenCV在作相機校正時，有考量到徑向(radial)和切向(tangential)兩個因素，徑向失真會產生桶狀或類似魚眼效果的失真，對於一個校正前的像素點(X，Y)，校正後的位置在($X_{corrected}$，$Y_{corrected}$)，以下為徑向失真校正前後位置的關係：

$$x_{corrected} = x(1 + k_1r^2 + k_2r^4 + k^3r^6$$
$$y_{corrected} = y(1 + k_1r^2 + k_2r^4 + k^3r^6$$

切向失真主要發生在鏡頭和拍攝物體面不是平行，由以下的公式進行切向失真校正：

$$x_{corrected} = x + [2p_1y + p_2(r^2 + 2x^2)]$$
$$y_{corrected} = y + [2p_1(r^2 + 2y^2) + 2p_2x]$$

綜合以上兩個公式，OpenCV主要考量以下五個參數的矩陣進行校正：

$$Distortion_{coefficients} = (k_1 \quad k_2 \quad p_1 \quad p_2 \quad k_3)$$

**OpenCV** 找棋盤角點

bool findChessboardCorners(InputArray image, Size patternSize, OutputArray corners, int flags = CALIB_CB_ADAPTIVE_THRESH+CALIB_CB_NORMALIZE_IMAGE)

- image：輸入圖，必須為8位元的灰階或彩色影像。
- patternSize：棋盤的尺寸，patternSize = Size(points_per_row,points_per_colum) = Size(columns,rows)。

- corners：輸出角點。
- flags：旗標，CV_CALIB_CB_ADAPTIVE_THRESH表示使用區域自適應濾波進行二值化，CV_CALIB_CB_NORMALIZE_IMAGE表示二值化前先呼叫equalizeHist()。
- 有找到棋盤角點返回true，否則返回false。

**OpenCV** 找精確角點

void cornerSubPix(InputArray image, InputOutputArray corners, Size winSize, Size zeroZone, TermCriteria criteria)

- image：輸入圖。
- corners：輸入角點，會更新成更精確的位置。
- winSize：搜尋範圍，假設輸入Size(5,5)，則會搜尋5*2+1的範圍，也就是11×11的尺寸
- criteria：迭代搜尋的終止條件。

**OpenCV** 相機校正

double calibrateCamera(InputArrayOfArrays objectPoints, InputArrayOfArrays imagePoints, Size imageSize, InputOutputArray cameraMatrix, InputOutputArray distCoeffs, OutputArrayOfArrays rvecs, OutputArrayOfArrays tvecs, int flags=0, TermCriteria criteria=TermCriteria( TermCriteria::COUNT+TermCriteria::EPS, 30, DBL_EPSILON))

- objectPoints：校正座標系統的位置，輸入型態為std::vector< std::vector< cv::Vec3f > >。
- imagePoints：校正點的投影，輸入型態為std::vector< std::vector< cv::Vec2f > >，imagePoints.size()和objectPoints.size()必須相同，且imagePoints[i].size()和objectPoints[i].size()也必須相同。
- imageSize：影像尺寸。
- cameraMatrix：輸出3×3浮點數的相機矩陣。
- distCoeffs：輸出的畸變參數，(k_1, k_2, p_1, p_2[, k_3[, k_4, k_5, k_6]]) 的4、5或8個元素。

**OpenCV** 校正矩陣

void initUndistortRectifyMap(InputArray cameraMatrix, InputArray distCoeffs, InputArray R, InputArray newCameraMatrix, Size size, int m1type, OutputArray map1, OutputArray map2)

- cameraMatrix：輸入的相機矩陣。
- distCoeffs：輸入的畸變參數。
- newCameraMatrix：新的相機矩陣。
- size：沒有畸變影像的尺寸。
- m1type：map1的型態，可以為CV_32FC1或CV_16SC2。
- map1：第一個輸出矩陣。
- map2：第二個輸出矩陣。

我們程式碼用以下步驟進行相機校正：

1. 多次拍攝圖表，並將影像存在硬碟作為輸入資料，這邊選用14張圖表影像。
2. 尋找每張影像的校正點，為了得到更精確的位置，另外呼叫cornerSubPix()函式，TermCriteria 代表終止條件，這邊是用迭代次數30次，最小精度0.1當作終止條件。
3. 輸入每張影像校正後的校正點位置。
4. 假設這張影像的每個校正點都有找到，依我們的輸入圖來說，也就是24個點都有找到的話，就把這張影像的校正點存入當作輸入的資料點。
5. 呼叫calibrateCamera()得到相機矩陣cameraMatrix，以及畸變矩陣distCoeffs，這兩個矩陣會在後續校正時使用。
6. 呼叫initUndistortRectifyMap()得到映射位置矩陣map1、map2。
7. 得到map1和map2後，即可對實際輸入的影像進行映射得到校正後的影像。
8. 可到此處下載圖檔：下載圖檔

### calibration.h

```
#include <cstdio>
#include <opencv2/opencv.hpp>
#include <vector>
#include <string>

using namespace cv;
using namespace std;

class CameraCalibrator{
private:
    vector<string> m_filenames;
```

```
    Size m_borderSize;
    vector<vector<Point2f> > m_srcPoints;
    vector<vector<Point3f> > m_dstPoints;
public:
    void setFilename();
    void setBorderSize(const Size &borderSize);
    void addChessboardPoints();
    void addPoints(const vector<Point2f> &imageCorners, const
vector<Point3f> &objectCorners);
    void calibrate(const Mat &src, Mat &dst);
};
```

## calibration.cpp

```
#include "calibration.h"

void CameraCalibrator::setFilename(){
    m_filenames.clear();
    m_filenames.push_back("chessboard01.jpg");
    m_filenames.push_back("chessboard02.jpg");
    m_filenames.push_back("chessboard03.jpg");
    m_filenames.push_back("chessboard04.jpg");
    m_filenames.push_back("chessboard05.jpg");
    m_filenames.push_back("chessboard06.jpg");
    m_filenames.push_back("chessboard07.jpg");
    m_filenames.push_back("chessboard08.jpg");
    m_filenames.push_back("chessboard09.jpg");
    m_filenames.push_back("chessboard10.jpg");
    m_filenames.push_back("chessboard11.jpg");
    m_filenames.push_back("chessboard12.jpg");
    m_filenames.push_back("chessboard13.jpg");
    m_filenames.push_back("chessboard14.jpg");
}

void CameraCalibrator::setBorderSize(const Size &borderSize){
    m_borderSize = borderSize;
}

void CameraCalibrator::addChessboardPoints(){
    vector<Point2f> srcCandidateCorners;
    vector<Point3f> dstCandidateCorners;
    for(int i=0; i<m_borderSize.height; i++){
        for(int j=0; j<m_borderSize.width; j++){
            dstCandidateCorners.push_back(Point3f(i, j, 0.0f));
        }
    }

    for(int i=0; i<m_filenames.size(); i++){
        Mat image=imread(m_filenames[i],CV_LOAD_IMAGE_GRAYSCALE);
```

```
        findChessboardCorners(image, m_borderSize, srcCandidateCorners);
        TermCriteria param(TermCriteria::MAX_ITER + TermCriteria::EPS,
30, 0.1);
        cornerSubPix(image, srcCandidateCorners, Size(5,5), Size(-1,-1),
param);
        if(srcCandidateCorners.size() == m_borderSize.area()){
            addPoints(srcCandidateCorners, dstCandidateCorners);
        }
    }
}

void CameraCalibrator::addPoints(const vector<Point2f> &srcCorners,
const vector<Point3f> &dstCorners){
    m_srcPoints.push_back(srcCorners);
    m_dstPoints.push_back(dstCorners);
}

void CameraCalibrator::calibrate(const Mat &src, Mat &dst){
    Size imageSize = src.size();
    Mat cameraMatrix, distCoeffs, map1, map2;
    vector<Mat> rvecs, tvecs;
    calibrateCamera(m_dstPoints, m_srcPoints, imageSize, cameraMatrix,
distCoeffs, rvecs, tvecs);
    initUndistortRectifyMap(cameraMatrix, distCoeffs, Mat(), Mat(),
imageSize, CV_32F, map1, map2);
    remap(src, dst, map1, map2, INTER_LINEAR);
}
```

## main.cpp

```
#include "calibration.h"

int main(){
    CameraCalibrator myCameraCalibrator;
    myCameraCalibrator.setFilename();
    myCameraCalibrator.setBorderSize(Size(6,4));
    myCameraCalibrator.addChessboardPoints();

    Mat src = imread("chessboard09.jpg",0);
    Mat dst;
    myCameraCalibrator.calibrate(src, dst);

    imshow("Original Image", src);
    imshow("Undistorted Image", dst);
    waitKey();
    return 0;
}
```



回到首頁

回到OpenCV教學

參考資料：

OpenCV 2 計算機視覺編成手冊 P.190~P.198

2016-01-04 阿宅 OpenCV, 影像間的投影關係 calibrateCamera, Camera calibration, cornerSubPix, findChessboardCorners, initUndistortRectifyMap

自豪的採用 WordPress