

程，具有一定的參考價值，感興趣的小夥伴們可以參考一下

相機標定：簡單的說，就是獲得相機引數的過程。引數如：相機內參數矩陣，投影矩陣，旋轉矩陣和平移矩陣等

什麼叫相機引數？

簡單的說，將現實世界中的人、物，拍成一張影象（二維）。人或物在世界中的三維座標，和影象上對應的二維座標間的關係。表達兩種不同維度座標間的關係用啥表示？用相機引數。

相機的成像原理

先來看一下，相機的成像原理：

[首頁](#)[MySQL教學](#)[網站技巧](#)[網路程式設計](#)[軟體程式設計](#)[資料庫](#)[作業系統](#)[其它](#)[軟體程式設計](#)

ITREAD01

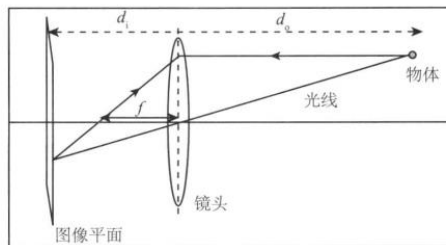
聊一聊OpenCV相

[首頁](#)[科技](#)[程式語言](#)[軟體程式設計](#) · 發表 2018-01-10

NT\$

馬_

這篇文章主要為大家詳細介紹了OpenCV相機標定的相關資料，即獲得相機引數的過



如圖所示，這是一個相機模型。將物體簡化看成一個點。來自物體的光，通過鏡頭，擊中影象平面（影象感測器），以此成像。 d_0 是物體到鏡頭的距離， d_i 是鏡頭到影象平面的距離， f 是鏡頭的焦距。三者滿足以下關係。

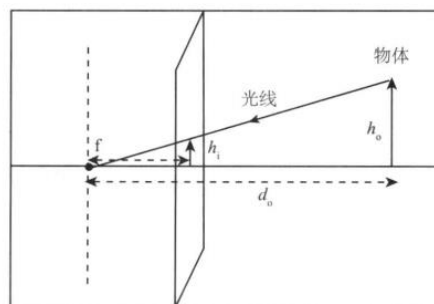
$$\frac{1}{f} = \frac{1}{d_0} + \frac{1}{d_i}$$

現在，簡化上面的相機模型。

將相機孔徑看成無窮小，只考慮中心位置的射線，這樣就忽視了透鏡的影響。然後由

於 d_0 遠遠大於 d_i ，將影像平面放在焦距處，這樣物體在影像平面上成像為倒立的影像（沒有透鏡的影響，只考慮從中心的孔徑進入的光線）。這個簡化的模型就是針孔攝像機模型。然後，我們在鏡頭前，將影像平面放在焦距距離的位置，就可以簡單獲得一個筆直的影像（不倒立）。當然，這只是理論上的，你不可能將影像感測器從相機裡拿出來，放在鏡頭前面。實際應用中，針孔攝像機應該是將成像後的影像倒過來，以獲得正立的影像。

到此，我們獲得了一個簡化的模型，如下圖：



h_0 是物體的高， h_i 是影象上物體的高， f 是焦距（距離）， d_0 是影象到鏡頭的距離。四者滿足如下關係：

$$h_i = f \frac{h_0}{d_0}$$

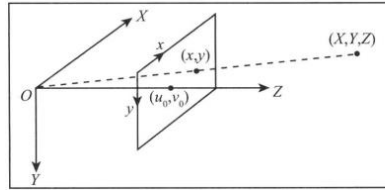
(1)

物體在影象中的高度 h_i ，和 d_0 成反比。也就是說，離鏡頭越遠，物體在影象中越小，離得越近越大（好吧，這句話是廢話）。

但通過這個式子，我們便能夠預測三維中的物體，在影象（二維）中的位置。那麼怎麼預測？

相機標定

如下圖所示，根據上面簡化的模型，考慮三維世界中的一個點，和其在影象（二維）中的座標關係。



(X, Y, Z) 為點的三維座標，(x, y) 為其通過相機成像後在影象（二維）上的座標。 u_0 和 v_0 是相機的中心點（主點），該點位於影象平面中心（理論上是這樣。但實際的相機會有幾個畫素的偏差）現在只考慮 y 方向上，由於需要將三維世界中的座標，轉換為影象上的畫素（影象上的座標，實際上是畫素的位置），需要求 y 方向上焦距等於多少個畫素（用畫素值表示焦距）， P_y 表示畫素的高，焦距 f （米或毫米）。垂直畫素表示的焦距為

$$f_y = f / p_y$$

根據式子(1)，只考慮y方向。我們三維世界中得點，在影象(二維)中y的座標。

$$y = \frac{f_y Y}{Z} + v_0$$

同理，得到x的座標。

$$x = \frac{f_x X}{Z} + u_0$$

現在，將上圖中的座標系的原點O，移動到影象的左上角。由於(x, y)是關於(u0, v0)的偏移，上面表示影象(二維)中點的座標的式子不變。將式子以矩陣的形式重寫，得。

$$s \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

其中，等式左邊的第一個矩陣，叫做“相機內參數矩陣”，第二個矩陣叫（投影矩陣）。

更為一般的情況，開始時的參考座標系不位於主點（中心點），需要額外兩個引數“旋轉向量”和“平移向量”來表示這個式子，這兩個引數在不同視角中是不一樣的。整合後，上述式子重寫為。

$$s \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_1 & r_2 & r_3 & t_1 \\ r_4 & r_5 & r_6 & t_2 \\ r_7 & r_8 & r_9 & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

校正畸變

通過相機標定，獲得了相機引數後，可以計算兩個對映函式（x座標和y座標），它們分別給出了沒有畸變的影象座標。將畸變的影象重新對映成為沒有畸變的影象。

程式碼：

做相機標定時，一般用標定板(棋盤)拍攝一組影象，利用這些影象提取角點，通過角點在影象中得座標和三維世界中的座標（通常自定義3維座標），計算相機引數。

```
1  std::vector<cv::Point2f> imagePoints;
2  //提取標定影象角點，
3  cv::findChessboardCorners(image, boardSize, //角點
4                             imageCorners);
5
```

函式calibrateCamera完成相機標定工作。

```
1  cv::calibrateCamera(imagePoints, //二
2                      imageSize, //影象大小
3                      camerMatrix, //相機內參
4                      disCoeffs, //投影係數
5                      rvecs, //旋轉
6                      tvecs, //平移
7                      flag //標記opencv標定方法
8                      );
9
```

計算畸變引數，去畸變

```
1  //計算畸變引數
2  cv::initUndistortRectMaps(imagePoints, imageSize, cameraMatrix,
3                             distCoeffs, cv::Mat(), cv::Mat());
```

```

4   map1, //x對映函
5   map2 //y對映函式
6   );
7   //應用對映函式
8   cv::remap(image,
9   undistorted, //去
10  map1, map2, cv::INTER_LINEAR);

```

現在整合程式碼。

示例：

標頭.h

```

1  #include<opencv2/calib3d.hpp>
2  #include<opencv2/core.hpp>
3  #include<opencv2/imgproc.hpp>
4  #include<opencv2/highgui.hpp>
5  #include <opencv2/video.hpp>
6  #include<string>
7  #include<vector>
8  class CameraCalibration
9  {
10 private:
11     //世界座標
12     std::vector<Point3f> worldPoints;
13     //影象座標
14     std::vector<Point2f> imagePoints;
15     //輸出矩陣
16     cv::Mat cameraMatrix;
17     cv::Mat distCoeffs;
18     //標記
19     int flag;
20     //去畸變引數
21     cv::Mat map1,

```

```
22 //是否去畸變
23 bool mustInitl
24
25 ///儲存點資料
26 void addPoints
27 {
28     imagePoints
29     objectPoints
30 }
31 public:
32 CameraCalibrat
33 //開啟棋盤圖片
34 int addChessbo
35 {
36     std::vector<
37     std::vector<
38     //輸入角點的
39     for (int i =
40     {
41         for (int i
42         {
43             objectCo
44         }
45     }
46     //計算角點在
47     cv::Mat imag
48     int success
49     for (int i =
50     {
51         image = cv
52         //找到角點
53         bool found
54         cv::corner
55         imageCor
56         cv::Size
57         cv::Size
58         cv::Terr
```

```

59         30, 0.1);
60         if (imageCorners.empty())
61         {
62             addPointToCorners(imageCorners);
63             success = true;
64         }
65         //畫出角點
66         cv::drawContours(image, imageCorners, -1, Scalar::all(0), 1);
67         cv::imshow("Image", image);
68         cv::waitKey(10);
69     }
70     return success;
71 }
72
73 //相機標定
74 double calibrateCamera(const vector<Point2f>& imagePoints,
75                         const vector<Point3f>& objectPoints,
76                         Size imageSize, CameraModel model,
77                         const vector<Point2f>& distCoeffs,
78                         const vector<Point2f>& rvecs,
79                         const vector<Point2f>& tvecs,
80                         const vector<Point2f>& rvecs2,
81                         const vector<Point2f>& tvecs2,
82                         const vector<Point2f>& rvecs3,
83                         const vector<Point2f>& tvecs3,
84                         const vector<Point2f>& rvecs4,
85                         const vector<Point2f>& tvecs4,
86                         const vector<Point2f>& rvecs5,
87                         const vector<Point2f>& tvecs5,
88                         const vector<Point2f>& rvecs6,
89                         const vector<Point2f>& tvecs6,
90                         const vector<Point2f>& rvecs7,
91                         const vector<Point2f>& tvecs7,
92                         const vector<Point2f>& rvecs8,
93                         const vector<Point2f>& tvecs8,
94                         const vector<Point2f>& rvecs9,
95                         const vector<Point2f>& tvecs9,

```

```

96     }
97     //常成員函式，獲取
98     cv::Mat getCar
99     cv::Mat getDis
100 };

```

源.cpp

```

1  #include"標頭.h"
2  #include<iomanip>
3  #include<iostream>
4  int main()
5  {
6      CameraCalibrator
7      cv::Mat image;
8      std::vector<std::
9      cv::namedWindow
10     for (int i = 1;
11     {
12         ///讀取圖片
13         std::stringst
14         s << "D:/imag
15         std::cout <<
16
17         filelist.push
18         image = cv:::
19         cv::imshow("
20         cv::waitKey(
21     }
22     //相機標定
23     cv::Size boardS
24     Cc.addChessboar
25     Cc.calibrate(ir
26
27     //去畸變
28     image = cv::im

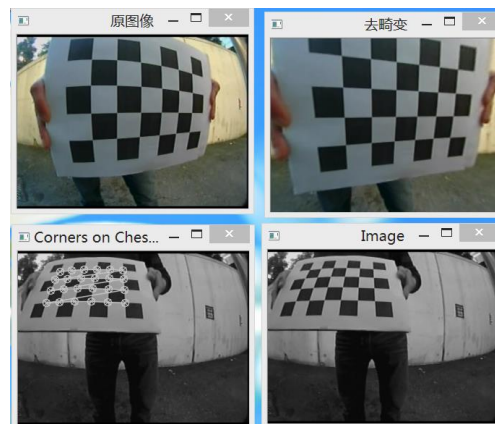
```

```

29     cv::Mat uImage=
30     cv::imshow("原圖",uImage);
31     cv::imshow("去畸變",uImage);
32     //顯示相機內參數
33     cv::Mat cameraMatrix;
34     std::cout << "相機內參數:" << endl;
35     std::cout << cameraMatrix << endl;
36     std::cout << cameraMatrix << endl;
37     std::cout << cameraMatrix << endl;
38
39     cv::waitKey(0);
40 }

```

實驗結果：



```

C:\Windows\system32\cmd.exe
D:\Images\chessboards\chessboard04.jpg
D:\Images\chessboards\chessboard05.jpg
D:\Images\chessboards\chessboard06.jpg
D:\Images\chessboards\chessboard07.jpg
D:\Images\chessboards\chessboard08.jpg
D:\Images\chessboards\chessboard09.jpg
D:\Images\chessboards\chessboard10.jpg
D:\Images\chessboards\chessboard11.jpg
D:\Images\chessboards\chessboard12.jpg
D:\Images\chessboards\chessboard13.jpg
D:\Images\chessboards\chessboard14.jpg
D:\Images\chessboards\chessboard15.jpg
D:\Images\chessboards\chessboard16.jpg
D:\Images\chessboards\chessboard17.jpg
D:\Images\chessboards\chessboard18.jpg
D:\Images\chessboards\chessboard19.jpg
D:\Images\chessboards\chessboard20.jpg
D:\Images\chessboards\chessboard21.jpg
D:\Images\chessboards\chessboard22.jpg
Camera intrinsic: 3x3
172.654 0 157.829
0 184.195 110.635
0 0 1
中文(简体) - 手心输入法 半

```

標籤： 軟體程式設計 C語言

您可能也會喜歡...

[Python使用OpenCV進 Android實現呼叫系統](#)

[行標定](#)

[相簿與相機設定頭像](#)

[並儲存在本地及伺服器](#)

[器](#)

[Android中通過訪問本地相簿或者相機設定](#)

[Android 相機相簿許可](#)

[權設定方法](#)

[權設定方法](#)

[使用者頭像例項](#)

[android 7自定義相機](#)

[預覽及拍照功能](#)

[Python+樹莓派+YOLO Android 自定義相機及](#)

[打造一款人工智慧照](#)

[分析原始碼](#)

[相機](#)

[Android自定義照相機](#)

[的例項](#)

[Android 用 camera2](#)

[iOS開發-自定義相機](#)

[API 自定義相機](#)

[例項\(仿微信\)](#)

[Spring Boot實戰之](#)

[Android中關於自定義](#)

[netty-socketio實現簡](#)

[相機預覽介面拉伸問](#)

[題](#)

[題](#)

[單聊天室\(給指定使用](#)

[者推送訊息\)](#)

[Android自定義相機實](#)

[現定時拍照功能](#)

[Android使用系統自帶的相機實現一鍵拍照功能](#)

[Android自定義相機實現自動對焦和手動對焦](#)

[Android自定義相機介面的實現程式碼](#)

[Android實用控制元件自定義逼真相機光圈View](#)

[Android自定義照相機Camera出現黑屏的解決方法](#)

[Android自定義照相機詳解](#)

[Android實現從本地相簿/相機拍照後裁剪圖片並設定頭像](#)

[首頁](#)

[Python教學](#)



ITREAD01.COM © 2018. 版權所有。

看以看到，相機內參數矩陣為

172.654 、 0 、 157.829
0 、 184.195 、 118.635
0 、 0 、 1

以上就是本文的全部內容，希望對大家的學習有所幫助，也希望大家多多支援itread01.com。

