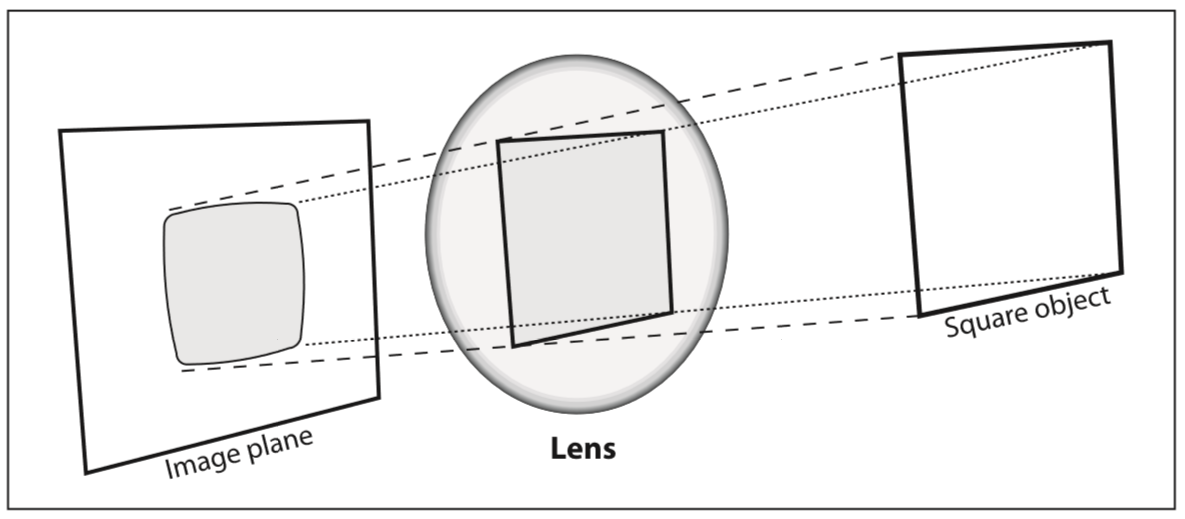
Camera Models and Calibration

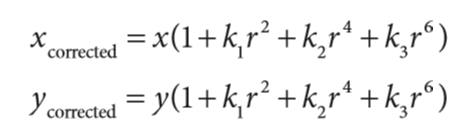
Camera ကို အသံုးျပဳၿပီး တကယ္႔လက္ေတြ႕ သံုးဖက္ျမင္ကမာၻမွာ တိက်တဲ႔ တိုင္းတာတြက္ခ်က္ျခင္းေတြ ျပဳလုပ္ႏိုင္ဖို႔ Camera calibration ျပဳလုပ္ေပးရန္ အေရးတႀကီး လိုအပ္ပါတယ္။ Camera calibration ဆိုတာ သခၤ်ာနည္းပညာကို အသံုးျပဳၿပီး ကင္မရာ ရဲ႕ လြဲမွားမႈေတြကို ျပဳျပင္ေပးတာပဲ ျဖစ္ပါတယ္။

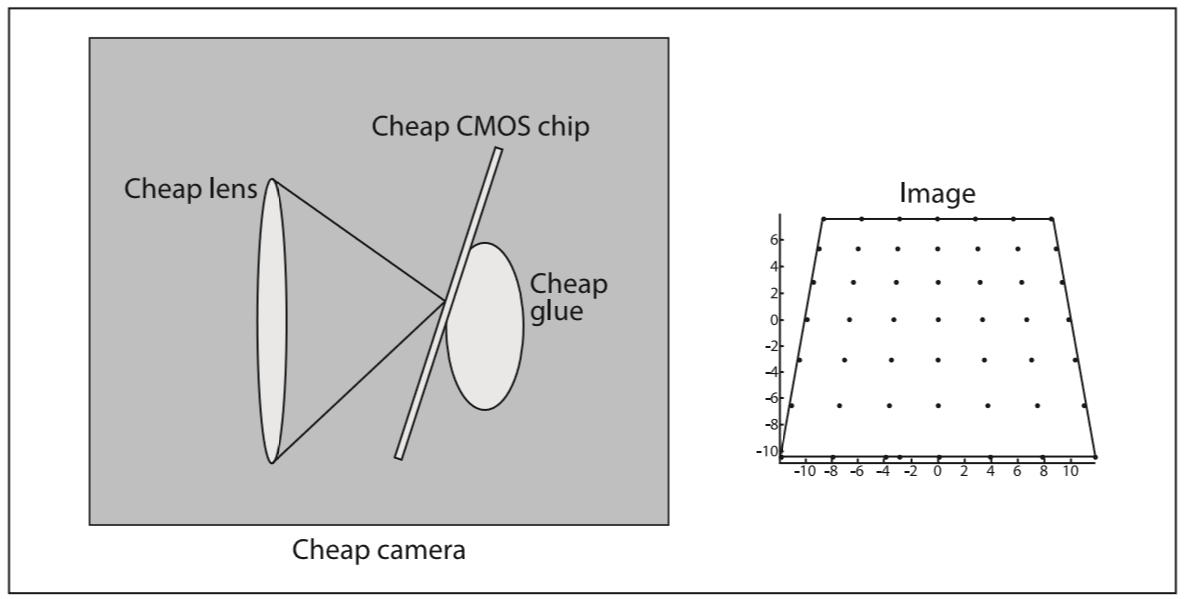
ကင္မရာ၏ geometry model နဲ႔ distortion model ႏွစ္ခုကို ကင္မရာရဲ႕ intrinsic parameters ေတြလို႔ ေခၚပါတယ္။ ထို႔ေနာက္ homography transform ကို အသံုးျပဳၿပီး ကင္မရာ၏ အေျခခံ ျပဳမူေဆာင္ရြက္ပံုႏွင္႔ လြဲမွားမႈမ်ား ကို ေလ႔လာျခင္း၊ ျပဳျပင္ေဆာင္ရြက္မႈမ်ားကို ျပဳလုပ္ႏိုင္ပါသည္။

Lens Distortions



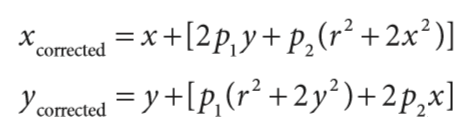
ကင္မရာမွန္ဘီလူးကို ေသြးရာတြင္ သခၤ်ာနည္းက်က် parabolic ပံုျဖစ္ေအာင္ ေသြးရန္ ခက္ခဲ၍ spherical အလြယ္တကူ ျပဳလုပ္ အသံုးျပဳေလ႔ ရွိၾကပါသည္။ ထို႔ေၾကာင္႔ မွန္ဘီလူးႏွင္႔ ပံုရိပ္ကို သခၤ်ာနည္းက်က် တည္႔မတ္ရန္ ခက္ခဲပါသည္။ ထို႔ေၾကာင္႔ Radial distortions ျဖစ္ေပၚလာရျခင္း ျဖစ္ပါသည္။ ၄င္းျဖစ္စဥ္တြင္ ပံုရိပ္၏ အစြန္းဖက္ပိုင္းတြင္ ပံုပ်က္ျခင္း ျဖစ္ေပၚတတ္ပါသည္။ barrel သို႔မဟုတ္ fish-eye effect ဟု ေခၚပါသည္။ ေစ်းေပါေသာ web cam မ်ားတြင္ အျဖစ္မ်ားပါသည္။ ေအာက္မွာ အဲဒီပံုပ်က္ျခင္းကို ကုစားတဲ႔ ေဖာ္ျမဴလာေပးထားပါတယ္။ x နဲ႔ y က မူရင္း ပံုပ်က္ေနတဲ႔ point ျဖစ္ၿပီး xcorrected နဲ႔ ycorrected ကေတာ႔ ျပင္ဆင္ၿပီး point ပဲ ျဖစ္ပါတယ္။





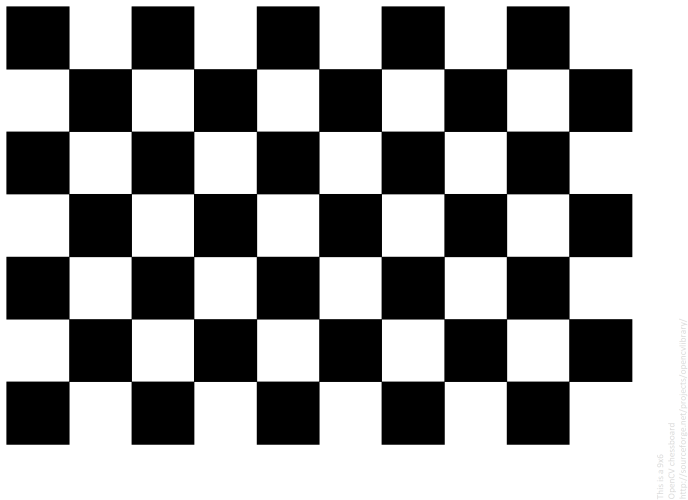
ပံု - tangential distortion

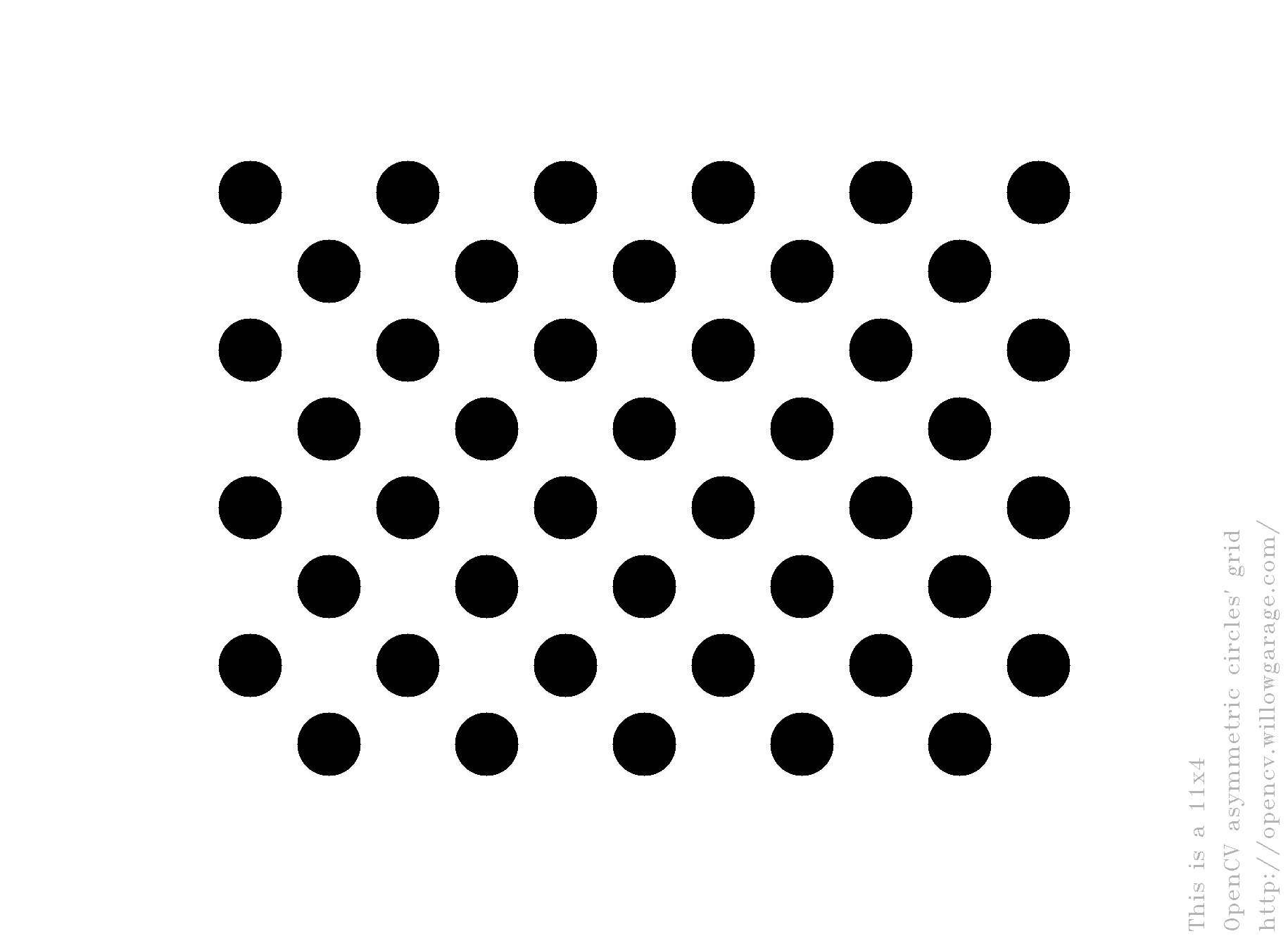
ေနာက္တစ္ခ်က္က tangential distortion ျဖစ္ပါတယ္။ ကင္မရာကို ထုတ္လုပ္ခ်ိန္မွာ မွန္ဘီလူးကို ပံုရိပ္ဖမ္းကိရိယာႏွင္႔ မ်ဥ္းၿပိဳင္ ျဖစ္ေအာင္ မတပ္ဆင္ႏိုင္တဲ႔ အတြက္ ျဖစ္ေပၚလာရျခင္းပါ။ အဲဒီ ပံုပ်က္ျခင္းကို ကုစားတဲ႔ ေဖာ္ျမဴလာကို ေပးထားပါတယ္။



ဒါေၾကာင္႔ ပံုပ်က္ျခင္းနဲ႔ ပက္သက္တဲ႔ coefficients စုစုေပါင္း ငါးခု ျဖစ္လာပါတယ္။ (k1,k2,k3,p1,p2)။ ၄င္းတို႔ အားလံုးကို စုေပါင္းၿပီး distortion vector လို႔ ေခၚပါတယ္။ ၄င္းကို 5\*1 matrix တစ္ခုနဲ႔ ကိုယ္စားျပဳ ေဖာ္ျပေလ႔ ရွိပါတယ္။

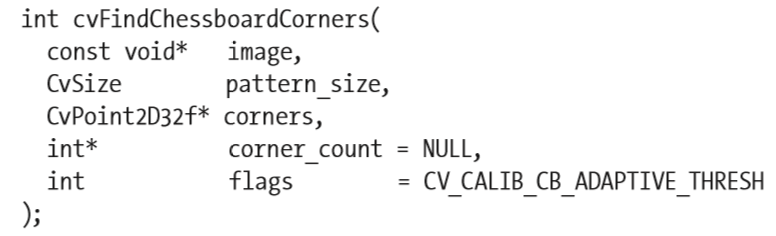
OpenCV မွာ calibration ျပဳလုပ္ဖို႔ cvCalibrateCamera2() ကို အသံုးျပဳရပါတယ္။ ဒီနည္းကို အသံုးျပဳဖို႔ အတြက္ အတိုင္းအတာနဲ႔ ပံုစံ တိတိက်က် သိေနတဲ႔ structure တစ္ခုရဲ႕ အေနအထားမတူတဲ႔ ပံုေပါင္းမ်ားစြာကို အသံုးျပဳရမွာပါ။ က်ားကြက္ပံုနဲ႔ စက္ဝိုင္းပံုေတြကို အသံုးျပဳေလ႔ ရွိပါတယ္။ ေအာက္မွာ နမူနာပံုေတြ ျပထားေပးပါတယ္။





အထက္ပါ ပံုေတြကို လွည္႔ပတ္ ေနရာေျပာင္းၿပီး ရိုက္ကူးျခင္းအားျဖင္႔ ပံုတစ္ပံုျခင္းကို ရိုက္ကူးစဥ္မွာ ရွိတဲ႔ ကင္မရာရဲ႕ relative location နဲ႔ orientation ေတြသာမက camera ရဲ႕ intrinsic parameters ေတြကို တြက္ခ်က္ရယူႏိုင္မွာ

အဲဒီလို ျပဳလုပ္ရာမွာ က်ားကြက္ရဲ႕ corners ေတြကို ရွာႏိုင္ဖို႔ cvFindChessboardCorners() ဆိုတဲ႔ function ကို အသံုးျပဳႏိုင္ပါတယ္။



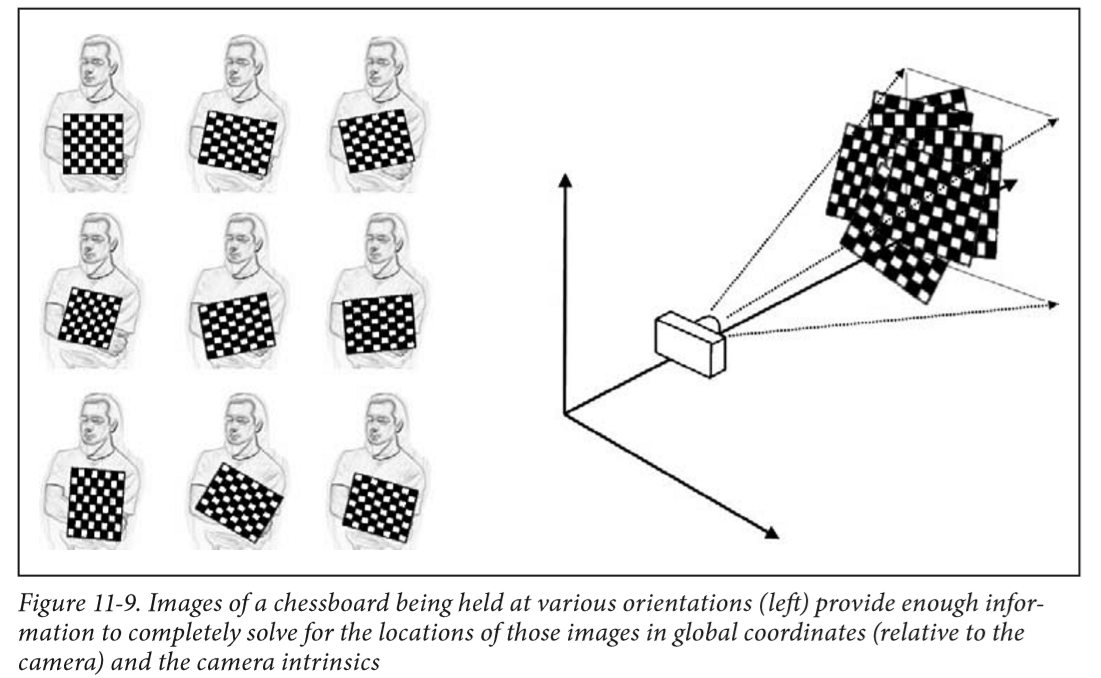
အဲဒီ function ရဲ႕ ပထမ argument က က်ားကြက္ပံု တစ္ပံု

ဒုတိယ က pattern\_size - က်ားကြက္ရဲ႕ ေဒါင္လိုက္နဲ႔ အတန္းလိုက္ corners ေတြ

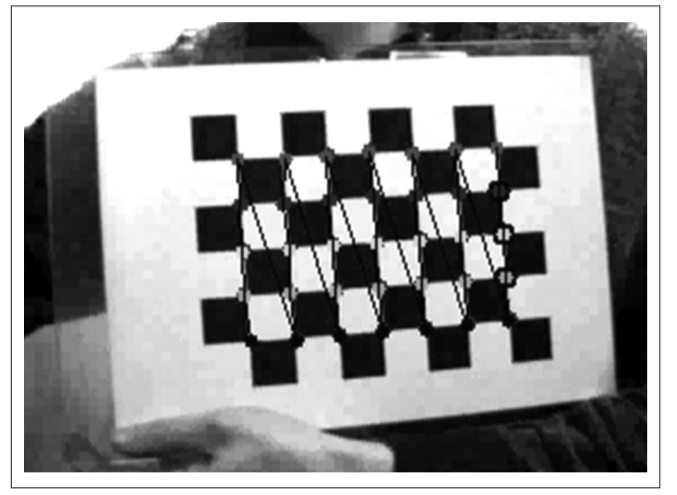
တတိယ က corners (pointer to array) corner location ကို သိမ္းဖို႔

စတုတၳက corner\_count က ထည္႔ခ်င္လဲထည္႔ မထည္႔ခ်င္လဲရတယ္။ NULL ေပးထားတယ္။ ေတြ႕တဲ႔ corner အေရအတြက္ကို သိမ္းဖို႔

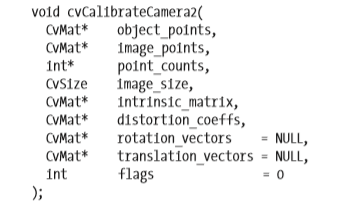
ေနာက္ဆံုး တစ္ခုက flags - corner ေတြ႕ရင္ non\_zero မေတြ႕ရင္ zero



ကၽြန္ေတာ္တို႔ က်ားကြက္ရဲ႕ corners ေတြကို ပံုဆြဲၿပီး ျပျခင္းျဖင္႔ ေတြ႕ရွိေၾကာင္း ရွင္းလင္းေအာင္ လုပ္ေပးႏိုင္တယ္။ ဒါကို cvDrawChessboardCorners() သံုးၿပီး ျပဳလုပ္ႏိုင္ပါတယ္။



Calibration ျပဳလုပ္ဖို႔ ေအာက္ပါ function ကို အသံုးျပဳႏိုင္ပါတယ္။



object\_points - မွာ calibration pattern corner coordinates ေတြ ပါဝင္တယ္၊ ျပထားတဲ႔ ပံုအားလံုးရဲ႕ တန္ဖိုးေတြ ပါဝင္တယ္. အဲဒီ ပက္တန္ေတြ စုေပါင္းၿပီး အဲဒီပံု ဘယ္လို ေပၚသင္႔တယ္ဆိုတာကို ေဖာ္ျပေပးတယ္။ တကယ္တမ္း ရွိရမယ္႔ ပံုကို ခန္႔မွန္းလို႔ရေအာင္။ က်ားကြက္ အျပားကို သံုးထားလို႔ Z coordinate တန္ဖိုးကို zero ေပးထားၿပီး ရိုးရွင္းေအာင္ လုပ္ထားတယ္။ က်ားကြက္ ပက္တန္ တစ္ခုကိုပဲ view အမ်ိဳးမ်ိဳးသံုးၿပီး ထည္႔သြင္းထားတာ ျဖစ္လို႔ တစ္ႀကိမ္တြက္ၿပီး အျခား view ေတြအတြက္ပါ ရွာလို႔ရတယ္။ a vector of vectors of calibration pattern points in the calibration pattern coordinate space.

image\_points - မွာ corners ေတြရဲ႕ တန္ဖိုးေတြပါဝင္တယ္။ အဲဒါေတြရဖို႔ က်ားကြက္ေထာင္႔ေတြကို ႀကိဳရွာထားရတယ္။ အဲဒီက တန္ဖိုးေတြကို သံုးထားတာ။ a vector of vectors of the projections of calibration pattern points.

point\_counts – ျပထားတဲ႔ ပံုအားလံုးရဲ႕ point အားလံုးရဲ႕ တန္ဖိုးေတြ။ a vector of integers, containing as many elements, as the number of views of the calibration pattern. Each element is the number of points in each view. Usually, all the elements are the same and equal to the number of feature points on the calibration pattern.

imageSize – ကင္မရာက ရတဲ႔ ဗီဒီယုိ၊ ဒါမွမဟုတ္ ဓာတ္ပံုက အရြယ္အစား

cameraMatrix – aspect ratio ကို အေသထားခ်င္တယ္ဆိုရင္ fx ကို zero ထားရမယ္

cameraMatrix = Mat::eye(3, 3, CV\_64F);

**if**( s.flag & CV\_CALIB\_FIX\_ASPECT\_RATIO )

cameraMatrix.at<double>(0,0) = 1.0;

distCoeffs – အေျဖလိုခ်င္တာ။ အေျဖအေနနဲ႔ ထြက္လာမွာ။ ေလာေလာဆယ္ zero နဲ႔ initialize လုပ္ေပးထားရမယ္။

distCoeffs = Mat::zeros(8, 1, CV\_64F);

rotation\_vectors – အေျဖလိုခ်င္တာ။ အေျဖအေနနဲ႔ ထြက္လာမွာ။ Output vector of rotation vectors (see [Rodrigues()](http://www.swarthmore.edu/NatSci/mzucker1/opencv-2.4.10-docs/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html" \l "void Rodrigues(InputArray src, OutputArray dst, OutputArray jacobian)" \o "void Rodrigues(InputArray src, OutputArray dst, OutputArray jacobian)) ) estimated for each pattern view.

translation\_vectors – အေျဖလိုခ်င္တာ။ အေျဖအေနနဲ႔ ထြက္လာမွာ။ Output vector of translation vectors estimated for each pattern view.

flags –

Different flags that may be zero or a combination of the following values:

CV\_CALIB\_USE\_INTRINSIC\_GUESS cameraMatrix contains valid initial values of fx, fy, cx, cy that are optimized further. Otherwise, (cx, cy) is initially set to the image center ( imageSize is used), and focal distances are computed in a least-squares fashion. Note, that if intrinsic parameters are known, there is no need to use this function just to estimate extrinsic parameters. Use [solvePnP()](http://www.swarthmore.edu/NatSci/mzucker1/opencv-2.4.10-docs/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html" \l "bool solvePnP(InputArray objectPoints, InputArray imagePoints, InputArray cameraMatrix, InputArray distCoeffs, OutputArray rvec, OutputArray tvec, bool useExtrinsicGuess, int flags)" \o "bool solvePnP(InputArray objectPoints, InputArray imagePoints, InputArray cameraMatrix, InputArray distCoeffs, OutputArray rvec, OutputArray tvec, bool useExtrinsicGuess, int flags)) instead.

CV\_CALIB\_FIX\_PRINCIPAL\_POINT The principal point is not changed during the global optimization. It stays at the center or at a different location specified when CV\_CALIB\_USE\_INTRINSIC\_GUESS is set too.

CV\_CALIB\_FIX\_ASPECT\_RATIO The functions considers only fy as a free parameter. The ratio fx/fy stays the same as in the input cameraMatrix . When CV\_CALIB\_USE\_INTRINSIC\_GUESS is not set, the actual input values of fx and fy are ignored, only their ratio is computed and used further.

CV\_CALIB\_ZERO\_TANGENT\_DIST Tangential distortion coefficients (p\_1, p\_2) are set to zeros and stay zero.

CV\_CALIB\_FIX\_K1,...,CV\_CALIB\_FIX\_K6 The corresponding radial distortion coefficient is not changed during the optimization. If CV\_CALIB\_USE\_INTRINSIC\_GUESS is set, the coefficient from the supplied distCoeffs matrix is used. Otherwise, it is set to 0.

CV\_CALIB\_RATIONAL\_MODEL Coefficients k4, k5, and k6 are enabled. To provide the backward compatibility, this extra flag should be explicitly specified to make the calibration function use the rational model and return 8 coefficients. If the flag is not set, the function computes and returns only 5 distortion coefficients.

#include "opencv2/highgui/highgui.hpp"

#include "opencv2/imgproc/imgproc.hpp"

#include "opencv2/objdetect/objdetect.hpp"

#include "opencv2/core/core.hpp"

#include "opencv2/opencv.hpp"

#include "opencv/cv.h"

#include <iostream>

#include <stdio.h>

#include <conio.h>

using namespace cv;

using namespace std;

CascadeClassifier car\_cascade;

int main(int argc, const char\*\*argv)

{

string car\_cascade\_name = "d:/cars3.xml";

//CascadeClassifier car1\_cascade;

CascadeClassifier car\_cascade;

//XML ဖိုင္ကို ဆြဲတင္

car\_cascade.load(car\_cascade\_name);

if (car\_cascade.load(car\_cascade\_name))

{

printf("success\nXML LOADED\n");

}

if (!car\_cascade.load(car\_cascade\_name))

{

printf("--(!)Error loading\n"); return -1;

}

VideoCapture cap("D://car.avi"); // open the video file for reading

if (!cap.isOpened()) // if not success, exit program

{

cout << "Cannot open the video file" << endl;

return -1;

}

//cap.set(CV\_CAP\_PROP\_POS\_MSEC, 300); //start the video at 300ms

double fps = cap.get(CV\_CAP\_PROP\_FPS); //get the frames per seconds of the video

//Video ထဲက frames per second ကို ဖတ္

cout << "Frame per seconds : " << fps << endl;

int pt1, pt2, pt3, pt4;

Rect r;

int x;

int y;

int h;

int w;

Mat frame;

Mat frame1;

Mat frame2;

while (1)

{

bool bSuccess = cap.read(frame); // read a new frame from video

if (!bSuccess) //if not success, break loop

{

cout << "Cannot read the frame from video file" << endl;

break;

}

//Frame ကို grayscale ေျပာင္း

cvtColor(frame, frame1, CV\_BGR2GRAY);

//Grayscale image ရဲ႕ histogram ကို equalize ျပဳလုပ္တာ

equalizeHist(frame1, frame2);

std::vector<Rect> cars;

//ကားကို detect ျပဳလုပ္

car\_cascade.detectMultiScale(frame2, cars, 1.05, 2, 0 | CV\_HAAR\_SCALE\_IMAGE, Size(30, 30));

printf("%d cars are found.\n", cars.size());

for (size\_t i = 0; i < cars.size(); i++)

{

r = cars[i];

pt1 = r.x;

pt2 = r.y;

pt3 = r.x + r.width;

pt4 = r.y + r.height;

printf("a car is found at Rect(%d,%d,%d,%d).\n", r.x, r.y, r.width, r.height);

printf("pt1=%d,pt2=%d,pt3=%d,pt4=%d\n", pt1, pt2, pt3, pt4);

// Mat carROI=img\_gray(cars[i]);

x = cars[i].x;

y = cars[i].y;

h = y + cars[i].height;

w = x + cars[i].width;

rectangle(frame, Point(x, y), Point(w, h), Scalar(255, 255, 0), 2, 1, 0);

}

imshow("detectvideo", frame);

imshow("grayscalevideo", frame1); //show the frame in "MyVideo" window

imshow("histogramvideo", frame2);

if (waitKey(30) == 27) //wait for 'esc' key press for 30 ms. If 'esc' key is pressed, break loop

{

cout << "esc key is pressed by user" << endl;

break;

}

}

waitKey(0);

return 0;

}

// Given a view of a checkerboard on a plane, view that image and a

// list of others frontal parallel to that plane

//

// This presumes that you have previously callibrated your camera and stored

// an instrinics and distortion model for your camera.

//

// console application.

// Gary Bradski Oct 3, 2008

//

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* License:\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Oct. 3, 2008

Right to use this code in any way you want without warrenty, support or any guarentee of it working.

BOOK: It would be nice if you cited it:

Learning OpenCV: Computer Vision with the OpenCV Library

by Gary Bradski and Adrian Kaehler

Published by O'Reilly Media, October 3, 2008

AVAILABLE AT:

http://www.amazon.com/Learning-OpenCV-Computer-Vision-Library/dp/0596516134

Or: http://oreilly.com/catalog/9780596516130/

ISBN-10: 0596516134 or: ISBN-13: 978-0596516130

OTHER OPENCV SITES:

\* The source code is on sourceforge at:

http://sourceforge.net/projects/opencvlibrary/

\* The OpenCV wiki page (As of Oct 1, 2008 this is down for changing over servers, but should come back):

http://opencvlibrary.sourceforge.net/

\* An active user group is at:

http://tech.groups.yahoo.com/group/OpenCV/

\* The minutes of weekly OpenCV development meetings are at:

http://pr.willowgarage.com/wiki/OpenCV

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

#include "opencv2/highgui/highgui.hpp"

#include "opencv2/imgproc/imgproc.hpp"

#include "opencv2/objdetect/objdetect.hpp"

#include "opencv2/core/core.hpp"

#include "opencv2/opencv.hpp"

#include "opencv/cv.h"

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <conio.h>

#include <math.h>

#include <vector>

#include <iostream>

using namespace cv;

using namespace std;

int birdEye1(int argc, char\* argv[]) {

//if (argc != 4) return -1;

// INPUT PARAMETERS:

//

//int board\_w = atoi(argv[1]); //inner corners per row

//int board\_h = atoi(argv[2]); //inner corners per column

int board\_w = 3; //inner corners per row ဘုတ္ရဲ႕ အတြင္းေထာင္႔ အလ်ားလိုက္ အေရအတြက္။ အျပင္က //ေထာင္႔ ၂ ခုကို ခ်န္ၿပီး ေရတြက္ရတယ္

int board\_h = 4; //inner corners per column ဘုတ္ရဲ႕ အတြင္းေထာင္႔ ေဒါင္လ္ိုက္ အေရအတြက္။ အျပင္က

//ေထာင္႔၂ခုကို ခ်န္ၿပီး ေရတြက္ရတယ္

int board\_n = board\_w \* board\_h;

CvSize board\_sz = cvSize(board\_w, board\_h); //board Size ကို width နဲ႔ hight ကေန တြက္ယူတာ

//Hard coded intrinsics for the camera

//Calibration ကေန ရလာတဲ႔ အခ်က္အလက္၊ ကင္မရာရဲ႕ အေနအထားကို ေဖာ္ျပႏိုင္တဲ႔ အခ်က္အလက္

//3\*3 Matrix နဲ႔ သိမ္းတယ္။

Mat intrinsicMat = (Mat\_<double>(3, 3) <<

418.7490, 0., 236.8528,

0., 558.6650, 322.7346,

0., 0., 1.);

//Hard coded distortions for the camera

//Calibration ကေန ရလာတဲ႔ အခ်က္အလက္၊ ပံုပ်က္ျခင္းနဲ႔ ဆိုင္တဲ႔ အခ်က္အလက္

//CvMat ထဲကို cvmSet သံုးၿပီး ထည္႔တယ္။

CvMat\* distortion = cvCreateMat(1, 4, CV\_32F);

cvmSet(distortion, 0, 0, -0.0019);

cvmSet(distortion, 0, 1, 0.0161);

cvmSet(distortion, 0, 2, 0.0011);

cvmSet(distortion, 0, 3, -0.0016);

//Image ေတြကို initialize လုပ္ထားတာ

IplImage\* image = 0;

IplImage\* gray\_image = 0;

//argv[3] ကို command prompt ကေန မRUN ခ်င္လို႔ hardcoded လုပ္ၿပီး အေသ ထည္႔ေပးလိုက္တာ

argv[3] = "d:\\YBr5y.png";

//တကယ္လို႔ ပံုကိုဖြင္႔တာ အဆင္မေျပခဲ႔ရင္ Error message ျပၿပီး ထြက္မယ္

if ((image = cvLoadImage(argv[3])) == 0) {

//if ((image = cvLoadImage("d:\\YBr5y.png")) == 0) {

printf("Error: Couldn’t load %s\n", argv[3]);

return -1;

}

//ဖြင္႔လို႔ ရသြားရင္ ပံုကို Grayscale ေျပာင္းမယ္

// 8 - က bit depth of image elements , 1 - က channels အေရအတြက္။ grayscale ေျပာင္းမွာမို႔လို႔ 1 ပဲထား

gray\_image = cvCreateImage(cvGetSize(image), 8, 1);

cvCvtColor(image, gray\_image, CV\_BGR2GRAY);

// UNDISTORT OUR IMAGE

//ရလာတဲ႔ ပံုကို ပံုပ်က္ျခင္းေတြကေန ျပင္ဆင္မယ္

IplImage\* mapx = cvCreateImage(cvGetSize(image), IPL\_DEPTH\_32F, 1);

IplImage\* mapy = cvCreateImage(cvGetSize(image), IPL\_DEPTH\_32F, 1);

CvMat intrinsic(intrinsicMat);

//This initializes rectification matrices

// ***intrinsicMatrix* ([IntPtr](http://msdn2.microsoft.com/en-us/library/5he14kz8" \t "_blank))**

//The camera matrix (A) [fx 0 cx; 0 fy cy; 0 0 1]

// ***distortionCoeffs* ([IntPtr](http://msdn2.microsoft.com/en-us/library/5he14kz8" \t "_blank))**

//The vector of distortion coefficients, 4x1 or 1x4 [k1, k2, p1, p2].

// ***mapx* ([IntPtr](http://msdn2.microsoft.com/en-us/library/5he14kz8" \t "_blank))**

//The output array of x-coordinates of the map

// ***mapy* ([IntPtr](http://msdn2.microsoft.com/en-us/library/5he14kz8" \t "_blank))**

//The output array of y-coordinates of the map

cvInitUndistortMap(

&intrinsic,

distortion,

mapx,

mapy

);

IplImage \*t = cvCloneImage(image);

// Rectify our image

//ပံုျပန္ျပင္မယ္

cvRemap(t, image, mapx, mapy);

// GET THE CHESSBOARD ON THE PLANE

//က်ားကြက္ကို ယူမယ္

cvNamedWindow("Chessboard");

CvPoint2D32f\* corners = new CvPoint2D32f[board\_n]; //2 Dimension ကေန 3 Dimension

int corner\_count = 0;

//က်ားကြက္ရဲ႕ အတြင္းေထာင္႔ေတြကို ရွာမယ္၊ ေပးထားတဲ႔ ေထာင္႔ အေရအတြက္ မွားရင္ အကုန္မွားမယ္

int found = cvFindChessboardCorners(

image,

board\_sz,

corners,

&corner\_count,

CV\_CALIB\_CB\_ADAPTIVE\_THRESH | CV\_CALIB\_CB\_FILTER\_QUADS

);

if (!found){

printf("Couldn’t aquire chessboard on %s, "

"only found %d of %d corners\n",

argv[3], corner\_count, board\_n

);

return -1;

}

//Get Subpixel accuracy on those corners:

// ပိုတိက်ေအာင္ လုပ္တယ္

cvFindCornerSubPix(

gray\_image,

corners,

corner\_count,

cvSize(11, 11),

cvSize(-1, -1),

cvTermCriteria(CV\_TERMCRIT\_EPS | CV\_TERMCRIT\_ITER, 30, 0.1)

);

//GET THE IMAGE AND OBJECT POINTS:

// We will choose chessboard object points as (r,c): roll နဲ႔ column ပံုစံ ေရြး

// (0,0), (board\_w-1,0), (0,board\_h-1), (board\_w-1,board\_h-1).

//

CvPoint2D32f objPts[4], imgPts[4];

imgPts[0] = corners[0];

imgPts[1] = corners[board\_w - 1];

imgPts[2] = corners[(board\_h - 1)\*board\_w];

imgPts[3] = corners[(board\_h - 1)\*board\_w + board\_w - 1];

objPts[0].x = 0; objPts[0].y = 0;

objPts[1].x = board\_w - 1; objPts[1].y = 0;

objPts[2].x = 0; objPts[2].y = board\_h - 1;

objPts[3].x = board\_w - 1; objPts[3].y = board\_h - 1;

// DRAW THE POINTS in order: B,G,R,YELLOW

//ကာလာခြဲၿပီး ဆြဲမယ္ အျပာ၊ အစိမ္း၊ အနီ၊ အဝါ အစဥ္လိုက္

cvCircle(image, cvPointFrom32f(imgPts[0]), 9, CV\_RGB(0, 0, 255), 3); //blue

cvCircle(image, cvPointFrom32f(imgPts[1]), 9, CV\_RGB(0, 255, 0), 3); //green

cvCircle(image, cvPointFrom32f(imgPts[2]), 9, CV\_RGB(255, 0, 0), 3); //red

cvCircle(image, cvPointFrom32f(imgPts[3]), 9, CV\_RGB(255, 255, 0), 3); //yellow

// DRAW THE FOUND CHESSBOARD

//ေတြ႕တဲ႔ ဘုတ္ ေထာင္႔ေတြကို ဆက္ဆြဲမယ္

cvDrawChessboardCorners(

image,

board\_sz,

corners,

corner\_count,

found

);

cvShowImage("Chessboard", image);

// FIND THE HOMOGRAPHY

//

CvMat \*H = cvCreateMat(3, 3, CV\_32F);

cvGetPerspectiveTransform(objPts, imgPts, H);

Mat homography = H;

cvSave("Homography.xml", H); //We can reuse H for the same camera mounting

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*GENERATING 3X4 MATRIX\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// LET THE USER ADJUST THE Z HEIGHT OF THE VIEW

//

float Z = 23;

int key = 0;

IplImage \*birds\_image = cvCloneImage(image);

cvNamedWindow("Birds\_Eye");

// LOOP TO ALLOW USER TO PLAY WITH HEIGHT:

//

// escape key stops

//

while (key != 27) {

// Set the height

//

CV\_MAT\_ELEM(\*H, float, 2, 2) = Z;

// COMPUTE THE FRONTAL PARALLEL OR BIRD’S-EYE VIEW:

// USING HOMOGRAPHY TO REMAP THE VIEW

//ပံုကို ျပန္ျပင္ဆြဲဖို႔ homography ကို ျပန္သံုးမယ္

// CV\_INTER\_LINEAR | CV\_WARP\_INVERSE\_MAP | CV\_WARP\_FILL\_OUTLIERS ေတြက bird //eye view ကို ျဖစ္ေစတယ္

cvWarpPerspective(

image,

birds\_image,

H,

CV\_INTER\_LINEAR | CV\_WARP\_INVERSE\_MAP | CV\_WARP\_FILL\_OUTLIERS

);

cvShowImage("Birds\_Eye", birds\_image);

//imwrite("/home/lee/bird.jpg", birds\_image);

key = cvWaitKey();

if (key == 'u') Z += 0.5;

if (key == 'd') Z -= 0.5;

}

return 0;

}