=========== Updated 2023 Feb. 5 , if you have problem using the latest opencv (4.4.x.x etc), try to install an older version, follow the following methods

conda> pip uninstall opencv-python

conda> pip uninstall opencv-contrib-python

conda > pip install opencv-python==3.4.16.59,

conda > pip install opencv-contrib-python==3.4.16.59

Inside CSE-CUHK, use :> pip install --proxy <http://proxy.cse.cuhk.edu.hk:8000> opencv-python , etc

* =====
* You may try the following samples from
* <https://github.com/opencv/opencv/tree/4.x/> (download the zip file and find the sample codes)
* conda> python camshift.py 0 # camshift using camera 0 , use mouse to select area to track
* conda> python lk\_track.py 0 # lk tracker for using camera 0 (laptop camera)
* conda> python calibrate.py #using input image files from ..\data, you my use your own images

//////////////////////////////////////////////////////////////////////////

Download the Camshift lk\_track etc sample code in zip from OpenCV Github repository ( <https://github.com/opencv/opencv> ) (C++ or Python),

After unzip , python samples codes are in \opencv-4.x\samples\python

Also download <http://www.cse.cuhk.edu.hk/~khwong/www2/cmsc5711/lk_track_save_data.py>

And save into \opencv-4.x\samples\python

/////////////// lk\_track : Lk\_track test: 2022.3.12 ////////////////////////////

cd into sample \opencv-4.x\samples\python, in anaconda>

conda>python lk\_track.py 0 #0 is for laptop camera

conda>python lk\_track1.py 0 #0 is for laptop camera

conda>python lk\_track\_save\_data.py 0 #can save data into a data file "lk\_output.txt"

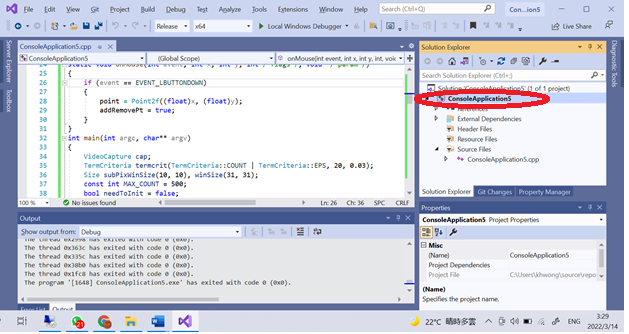
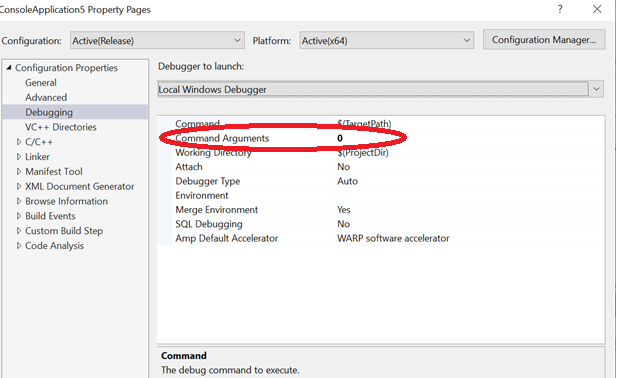
/////////////// camshift ////////////////////////////

conda> python camshift.py 0

* ======Python ========== For anaconda-python , 2020 Feb 1 ====================
* I recommend you to use anaconda-python , (you may follow the instruction to install anacondas -windows / python:  <https://www.anaconda.com/distribution/#download-section> ), then you can install the latest opencv-python by

1. Win10-search (lower-left-bottom) “anaconda-prompt (right-click-Open-location), Right click (run as administrator).
2. In anaconda-prompt :>> conda install -c conda-forge opencv
3. Now Opencv-python is installed. You may use Anaconda-spyder to edit/test python-opencv programs. For example , try the testing programs in <https://sites.google.com/site/hongslinks/opencv-1/opencv>
4. If you have indentation problem when running python programs, copy and paste your code into Note++ (<https://notepad-plus-plus.org/downloads/>), at top menu, click-on: View/show symbol/Show white space and tab. Make sure the tabs are replaced by four spaces. Mixing of tabs and spaces may not be allowed for Python, so either use tabs or 4-spaces in your program for indentation.
5. Camera Calibration demo/tutorial : <https://sites.google.com/site/hongslinks/opencv-1/camera-calibration-opencv-python>

======C++ ======= For cpp users through Visual studio , 2022 March 13 =================

* To enter arguments to a program under visual studio IDE
* In the visual studio main window-top bar , select View/solution explorer
* On the right, right click the red elliptical part (console application)
* 
* Select “debugging” enter argument, see the red-elliptical part in the figure
* 

( This document applies to both x32 and x64 project settings)

This document has 3 parts

1. Setting up your windows system
2. Setting up your visual studio
3. Examples
   1. Camshift (object tracking)
   2. Lkdemo (feature tracking)
   3. Camera calibration

2022 March 13, switching to opencv-4.5.5-vc14\_vc15 (applies also to a new OPENCV installation)

1. if you are using opencv3xx, change opencv3xx to opencv455 in the following passage.
2. Download from <https://opencv.org/opencv-4-0-0.html> , or
3. <https://sourceforge.net/projects/opencvlibrary/files/latest/download>
4. download opencv-4.5.5-vc14\_vc15.exe ,and run it.
5. You will get a directory opencv, rename to c:\opencv455 in my case.

**Part A**

1. For opencv400, Set: windows System Variable (type in win10 start: Edit the system environment) in environment-variables, User variables for YOUR\_NAME : OPENCV\_DIR should contain “c:\opencv455\build\x64\vc15” ~~not c:\opencv455\build\x64\vc15\~~ . Then, your system will use opencv455.

Graphical illustration is as follows

Double click “Path” on the right side of the above window (lower part), you will see the following. Assume you use python3.5, add “%OPENCV\_DIR%\bin”

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

**Part B (seems opencv455 has no 32 bit version, only x64)**

Setting up your visual studio (2013, 2017, 2018 , 2019 etc)

<https://www.tektutorialshub.com/visual-studio/how-to-download-and-install-visual-studio-2019/>

You may try the latest visual studio if you prefer. After installation open visual studio, start a “new project”.

From the top Microsoft visual studio menu (top)

select View>>other windows >>Property manager

step B.1:------------------------------ (same for x32, x64 and all versions of opencv):

c/c++ general

$(OPENCV\_DIR)\..\..\include\

Graphical user interface, text, application

Description automatically generated

Step B.2:------------------------------- (same for x32, x64 and all versions of opencv):

linker general

Additional library directories: $(OPENCV\_DIR)\lib

Graphical user interface, text, application, email

Description automatically generated

Step B.3:------------------------------- (same for x32, x64 and all versions of opencv):

(change this if you use a new opencv version)

Linker/input/additional dependencies:

opencv\_world455d.lib;%(AdditionalDependencies)

For debug mode use opencv\_world455d.lib

Graphical user interface, text, application, email

Description automatically generated

step B.4:------------------------ ( x32, x64 are different).

Check platform is x64 or x32, depending on your system setting. If you set up both x32 x64,

Then you can choose x32 or x64

Step B.5:------------------------(depends on your visual studio version)

make sure first line of the windows opencv cpp program is

#include <iostream>

~~or~~

~~#include "pch.h" // if using vs2017~~

~~Or~~

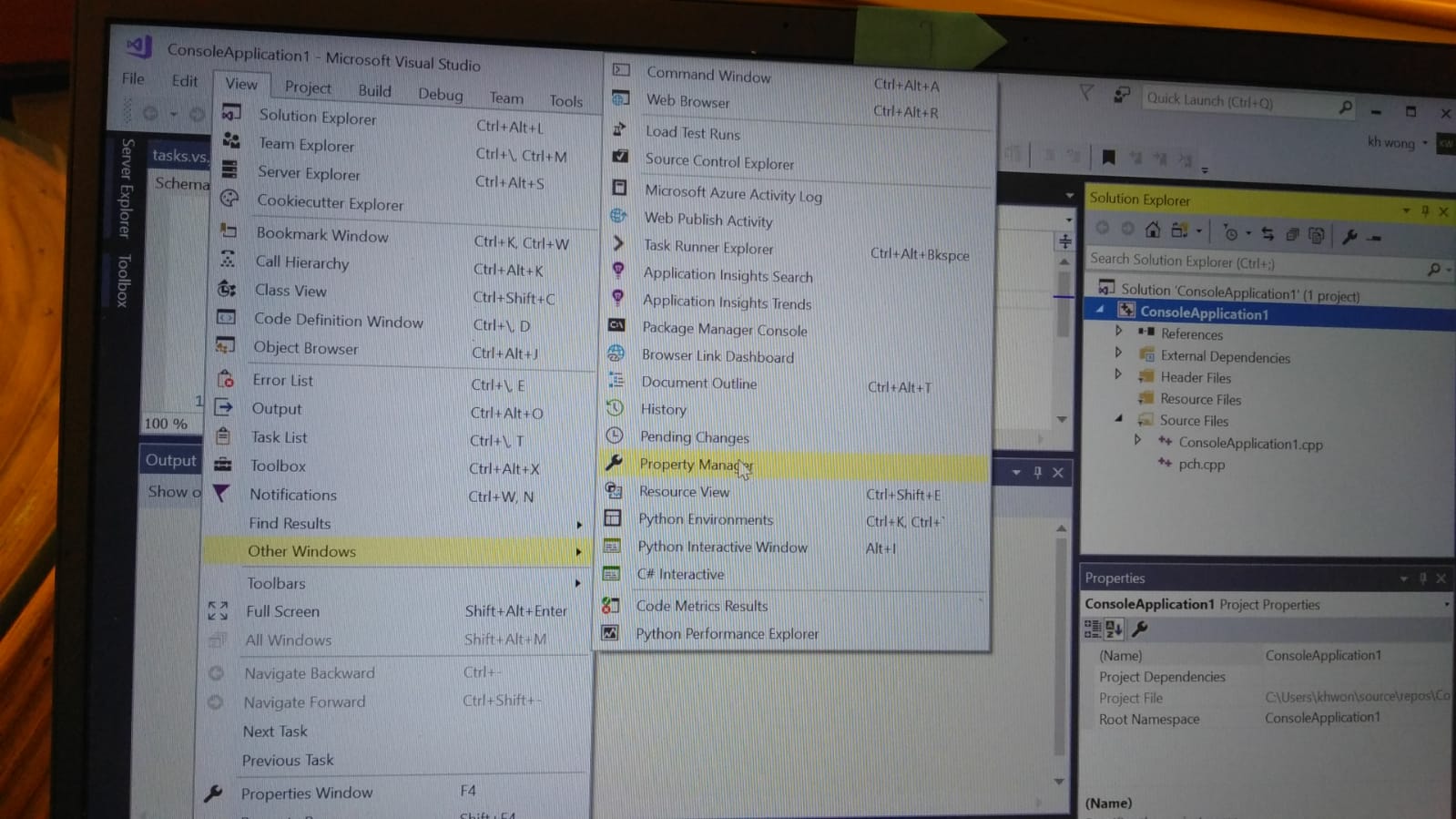
~~#include "stdafx.h" //if you are using vs2013 etc.~~

// older version; // older version; // older version; // older version; // older version; // older version;

Part B (using pictorial illustration)// older version

Setting default parameters for opencv320 for vs2017 projects, 2018 Oct 18

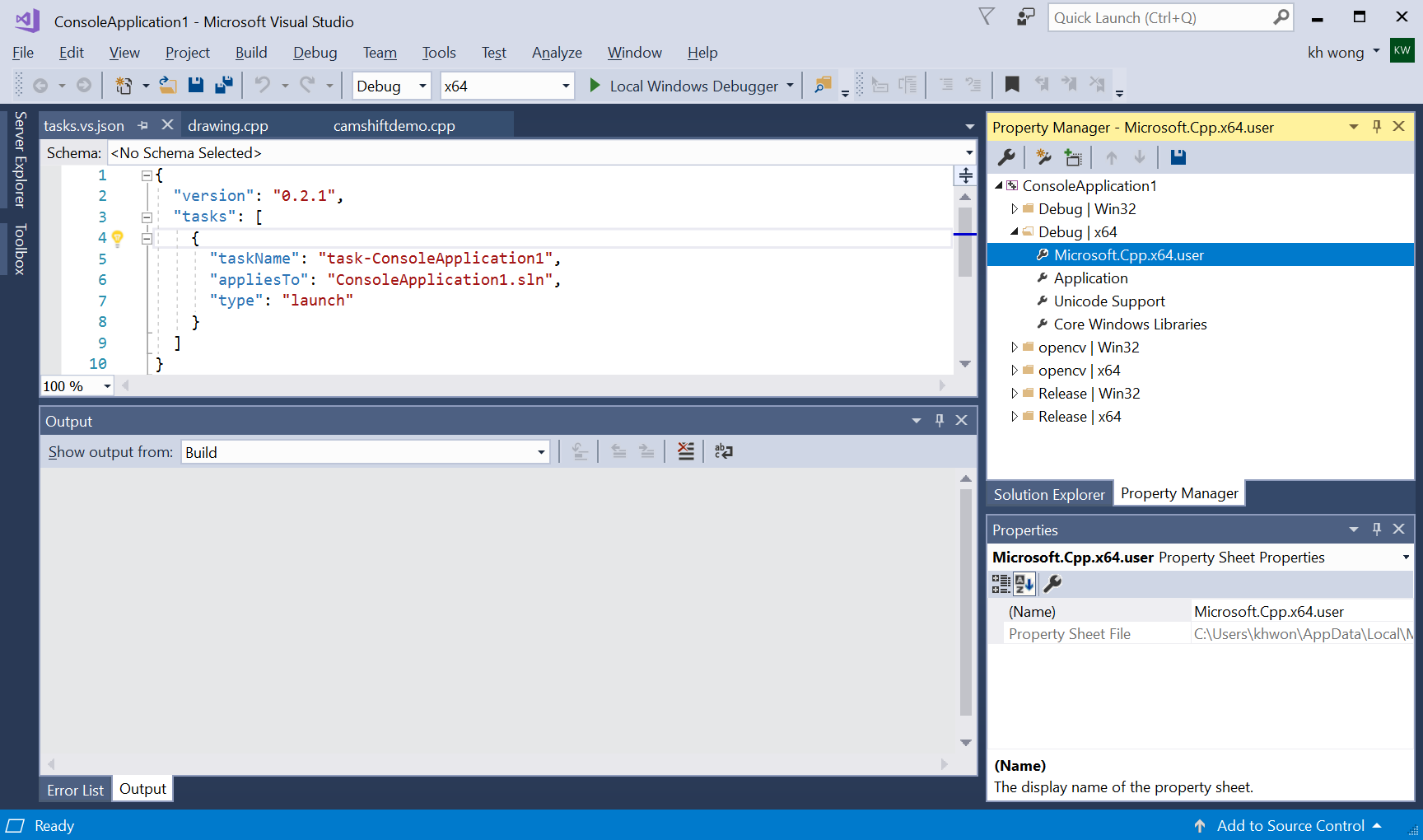
Create a new project



Part B (using pictorial illustration)

1. From the top Microsoft visual studio menu (top)

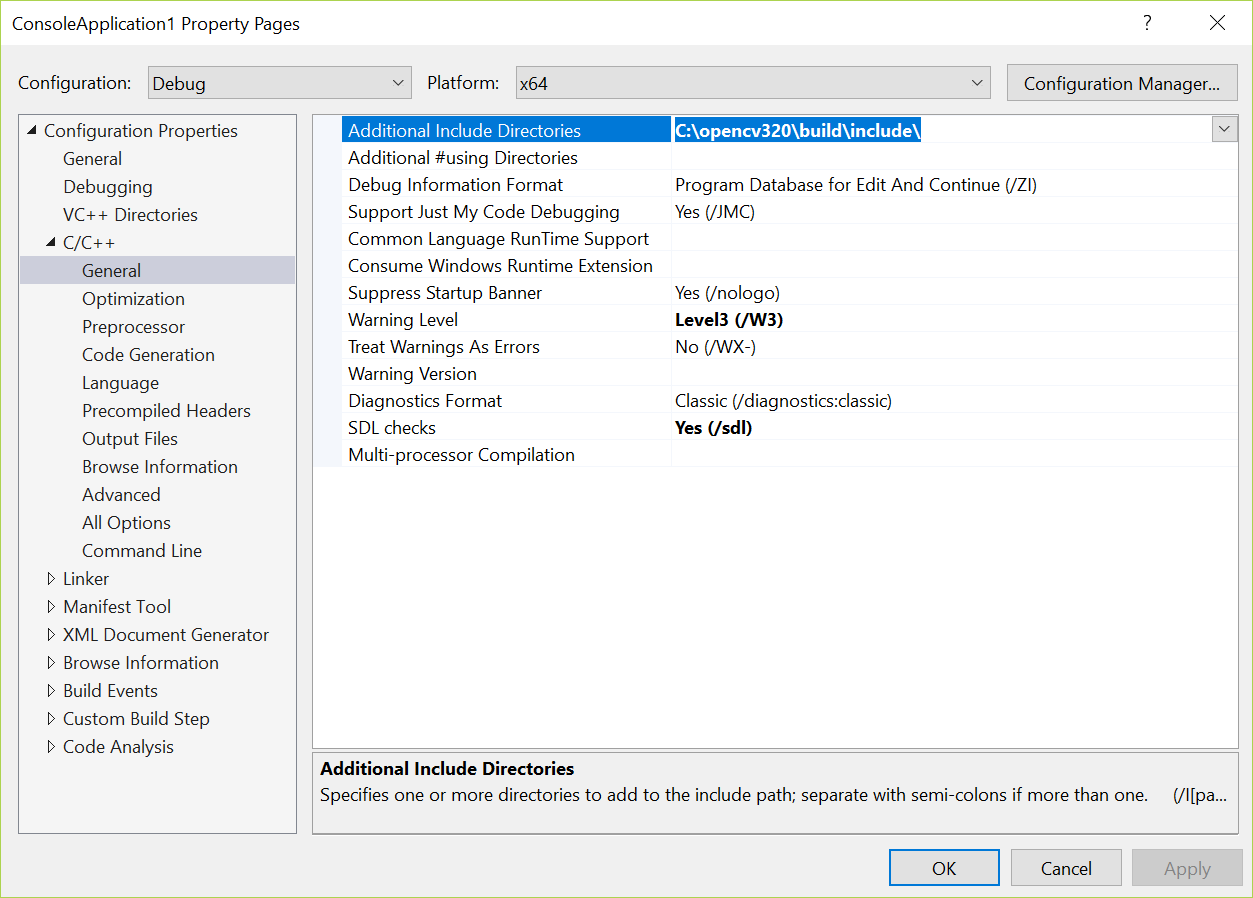
select View>>other windows >>Property manager



In Property Manager (right top) open Debug x64>> right-click Microsoft.Cpp.x64.user, do the followings 4 steps.

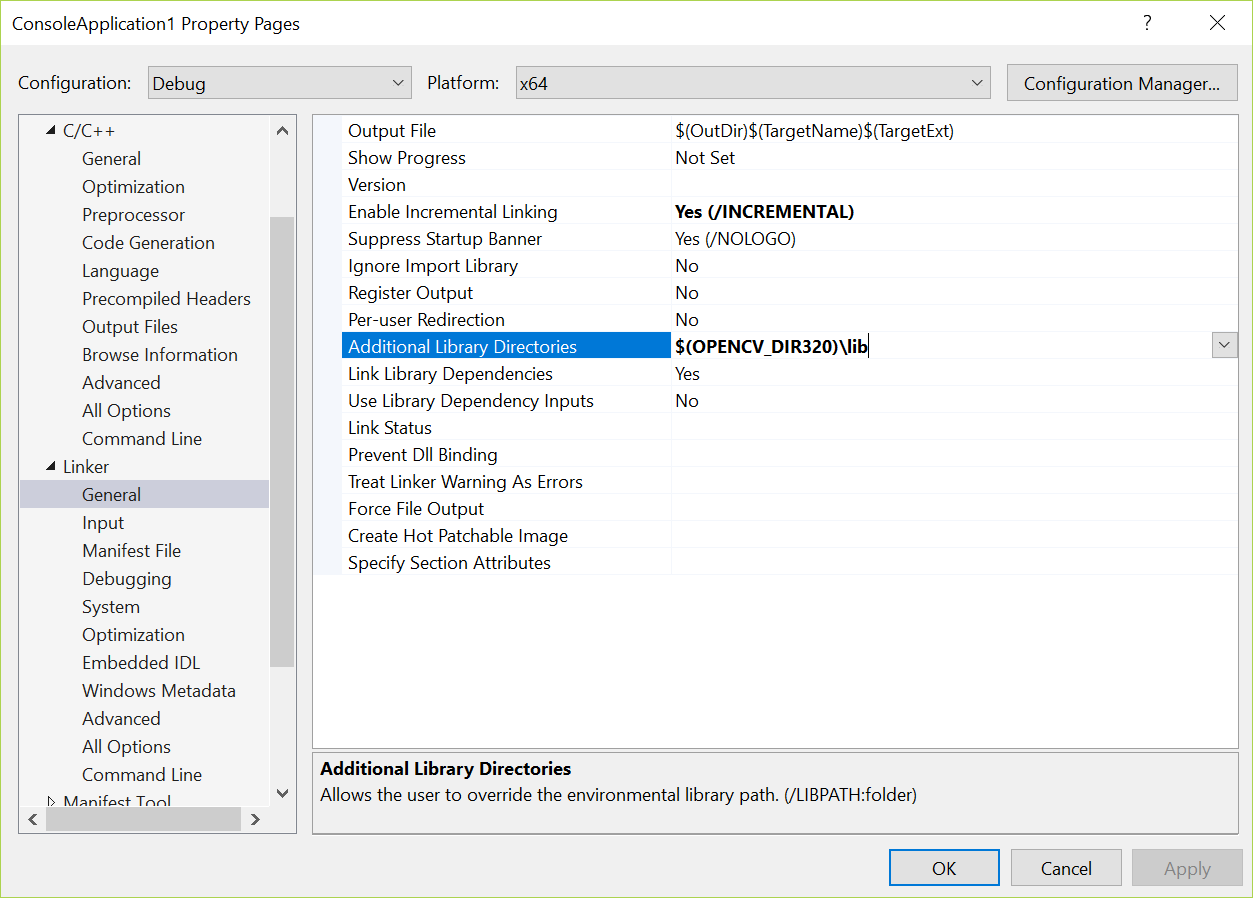
(can do the same for x32 , if you want to have x32 projects)

Step1:



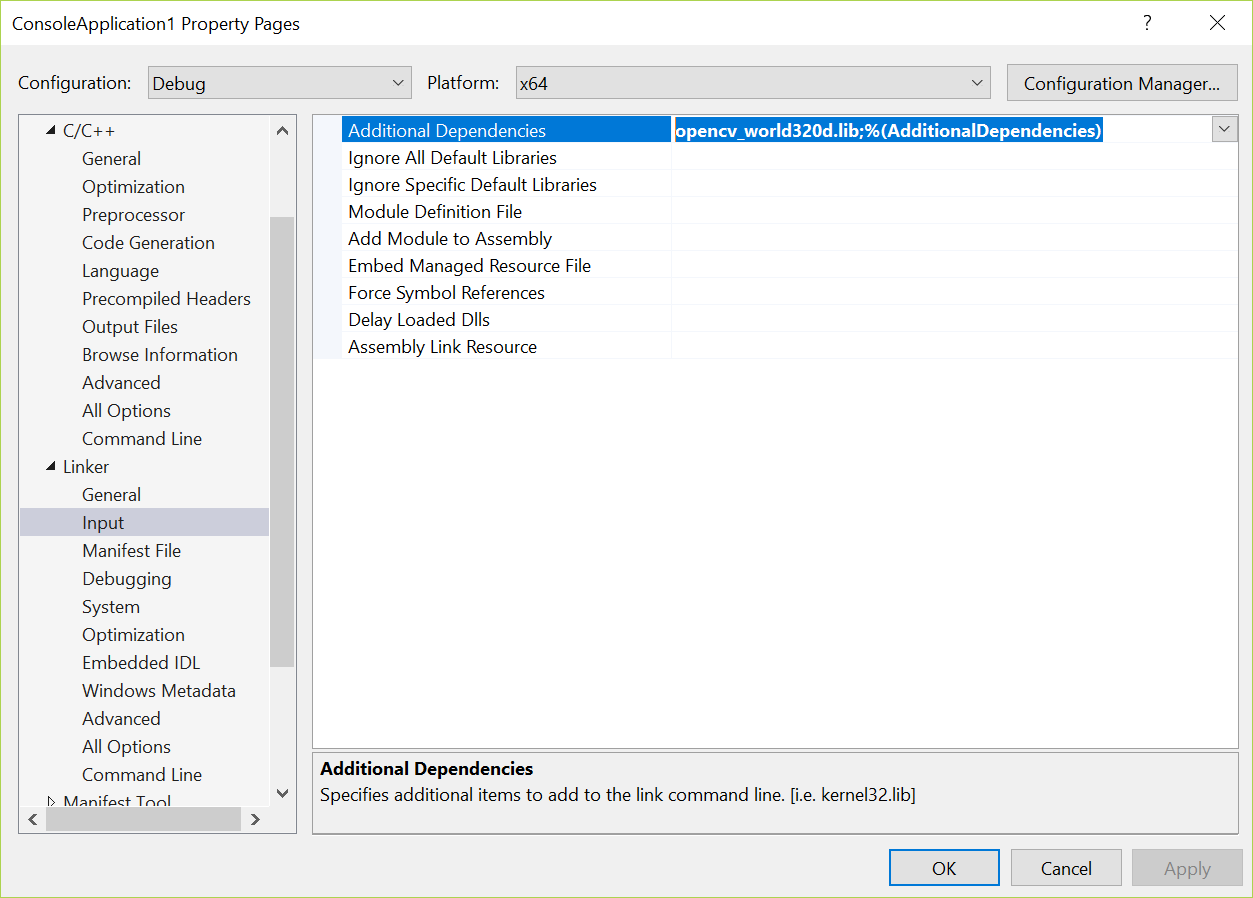
Additional include directory : C:\opencv320\build\include\

Step2:



Additional library directories: $(OPENCV\_DIR320)\lib

Step3:



Additional dependencies: opencv\_world320d.lib;%(AdditionalDependencies)

Step4:

1. Check in My\_code.cpp the first line is

#include "pch.h"

1. It is a x64 project or x32 project, use the correct one.

--------------------------------------Done-----------------------------

Part C, examples

Example code 1 (camshift: from C:\opencv400\sources\samples\cpp\ camshiftdemo.cpp)

// test2.cpp : This file contains the 'main' function. Program execution begins and ends there.

//

#include "pch.h"#include <opencv2/core/utility.hpp>

#include "opencv2/video/tracking.hpp"

#include "opencv2/imgproc.hpp"

#include "opencv2/videoio.hpp"

#include "opencv2/highgui.hpp"

#include <iostream>

#include <ctype.h>

using namespace cv;

using namespace std;

Mat image;

bool backprojMode = false;

bool selectObject = false;

int trackObject = 0;

bool showHist = true;

Point origin;

Rect selection;

int vmin = 10, vmax = 256, smin = 30;

// User draws box around object to track. This triggers CAMShift to start tracking

static void onMouse(int event, int x, int y, int, void\*)

{

if (selectObject)

{

selection.x = MIN(x, origin.x);

selection.y = MIN(y, origin.y);

selection.width = std::abs(x - origin.x);

selection.height = std::abs(y - origin.y);

selection &= Rect(0, 0, image.cols, image.rows);

}

switch (event)

{

case EVENT\_LBUTTONDOWN:

origin = Point(x, y);

selection = Rect(x, y, 0, 0);

selectObject = true;

break;

case EVENT\_LBUTTONUP:

selectObject = false;

if (selection.width > 0 && selection.height > 0)

trackObject = -1; // Set up CAMShift properties in main() loop

break;

}

}

string hot\_keys =

"\n\nHot keys: \n"

"\tESC - quit the program\n"

"\tc - stop the tracking\n"

"\tb - switch to/from backprojection view\n"

"\th - show/hide object histogram\n"

"\tp - pause video\n"

"To initialize tracking, select the object with mouse\n";

static void help()

{

cout << "\nThis is a demo that shows mean-shift based tracking\n"

"You select a color objects such as your face and it tracks it.\n"

"This reads from video camera (0 by default, or the camera number the user enters\n"

"Usage: \n"

" ./camshiftdemo [camera number]\n";

cout << hot\_keys;

}

const char\* keys =

{

"{help h | | show help message}{@camera\_number| 0 | camera number}"

};

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

{

VideoCapture cap;

Rect trackWindow;

int hsize = 16;

float hranges[] = { 0,180 };

const float\* phranges = hranges;

CommandLineParser parser(argc, argv, keys);

if (parser.has("help"))

{

help();

return 0;

}

int camNum = parser.get<int>(0);

cap.open(camNum);

if (!cap.isOpened())

{

help();

cout << "\*\*\*Could not initialize capturing...\*\*\*\n";

cout << "Current parameter's value: \n";

parser.printMessage();

return -1;

}

cout << hot\_keys;

namedWindow("Histogram", 0);

namedWindow("CamShift Demo", 0);

setMouseCallback("CamShift Demo", onMouse, 0);

createTrackbar("Vmin", "CamShift Demo", &vmin, 256, 0);

createTrackbar("Vmax", "CamShift Demo", &vmax, 256, 0);

createTrackbar("Smin", "CamShift Demo", &smin, 256, 0);

Mat frame, hsv, hue, mask, hist, histimg = Mat::zeros(200, 320, CV\_8UC3), backproj;

bool paused = false;

for (;;)

{

if (!paused)

{

cap >> frame;

if (frame.empty())

break;

}

frame.copyTo(image);

if (!paused)

{

cvtColor(image, hsv, COLOR\_BGR2HSV);

if (trackObject)

{

int \_vmin = vmin, \_vmax = vmax;

inRange(hsv, Scalar(0, smin, MIN(\_vmin, \_vmax)),

Scalar(180, 256, MAX(\_vmin, \_vmax)), mask);

int ch[] = { 0, 0 };

hue.create(hsv.size(), hsv.depth());

mixChannels(&hsv, 1, &hue, 1, ch, 1);

if (trackObject < 0)

{

// Object has been selected by user, set up CAMShift search properties once

Mat roi(hue, selection), maskroi(mask, selection);

calcHist(&roi, 1, 0, maskroi, hist, 1, &hsize, &phranges);

normalize(hist, hist, 0, 255, NORM\_MINMAX);

trackWindow = selection;

trackObject = 1; // Don't set up again, unless user selects new ROI

histimg = Scalar::all(0);

int binW = histimg.cols / hsize;

Mat buf(1, hsize, CV\_8UC3);

for (int i = 0; i < hsize; i++)

buf.at<Vec3b>(i) = Vec3b(saturate\_cast<uchar>(i\*180. / hsize), 255, 255);

cvtColor(buf, buf, COLOR\_HSV2BGR);

for (int i = 0; i < hsize; i++)

{

int val = saturate\_cast<int>(hist.at<float>(i)\*histimg.rows / 255);

rectangle(histimg, Point(i\*binW, histimg.rows),

Point((i + 1)\*binW, histimg.rows - val),

Scalar(buf.at<Vec3b>(i)), -1, 8);

}

}

// Perform CAMShift

calcBackProject(&hue, 1, 0, hist, backproj, &phranges);

backproj &= mask;

RotatedRect trackBox = CamShift(backproj, trackWindow,

TermCriteria(TermCriteria::EPS | TermCriteria::COUNT, 10, 1));

if (trackWindow.area() <= 1)

{

int cols = backproj.cols, rows = backproj.rows, r = (MIN(cols, rows) + 5) / 6;

trackWindow = Rect(trackWindow.x - r, trackWindow.y - r,

trackWindow.x + r, trackWindow.y + r) &

Rect(0, 0, cols, rows);

}

if (backprojMode)

cvtColor(backproj, image, COLOR\_GRAY2BGR);

ellipse(image, trackBox, Scalar(0, 0, 255), 3, LINE\_AA);

}

}

else if (trackObject < 0)

paused = false;

if (selectObject && selection.width > 0 && selection.height > 0)

{

Mat roi(image, selection);

bitwise\_not(roi, roi);

}

imshow("CamShift Demo", image);

imshow("Histogram", histimg);

char c = (char)waitKey(10);

if (c == 27)

break;

switch (c)

{

case 'b':

backprojMode = !backprojMode;

break;

case 'c':

trackObject = 0;

histimg = Scalar::all(0);

break;

case 'h':

showHist = !showHist;

if (!showHist)

destroyWindow("Histogram");

else

namedWindow("Histogram", 1);

break;

case 'p':

paused = !paused;

break;

default:

;

}

}

return 0;

}

-----example code 2-------------------------

// lkdemo1.cpp : This file contains the 'main' function. Program execution begins and ends there.

# //Note if you see error C2666: “cv::goodFeaturesToTrack”: The 8th paraneter :‘0’, is ambiguous. For newer compiler you need to change it to (bool)0,

//goodFeaturesToTrack(gray, points[1], MAX\_COUNT, 0.01, (int)10, Mat(), 3, 0, 0.04);

// change to

// goodFeaturesToTrack(gray, points[1], MAX\_COUNT, 0.01, (int)10, Mat(), 3, (bool)0, 0.04);

#include "pch.h"

#include "opencv2/video/tracking.hpp"

#include "opencv2/imgproc.hpp"

#include "opencv2/videoio.hpp"

#include "opencv2/highgui.hpp"

#include <iostream>

#include <ctype.h>

using namespace cv;

using namespace std;

static void help()

{

// print a welcome message, and the OpenCV version

cout << "\nThis is a demo of Lukas-Kanade optical flow lkdemo(),\n"

"Using OpenCV version " << CV\_VERSION << endl;

cout << "\nIt uses camera by default, but you can provide a path to video as an argument.\n";

cout << "\nHot keys: \n"

"\tESC - quit the program\n"

"\tr - auto-initialize tracking\n"

"\tc - delete all the points\n"

"\tn - switch the \"night\" mode on/off\n"

"To add/remove a feature point click it\n" << endl;

}

Point2f point;

bool addRemovePt = false;

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

{

if (event == EVENT\_LBUTTONDOWN)

{

point = Point2f((float)x, (float)y);

addRemovePt = true;

}

}

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

{

VideoCapture cap;

TermCriteria termcrit(TermCriteria::COUNT | TermCriteria::EPS, 20, 0.03);

Size subPixWinSize(10, 10), winSize(31, 31);

const int MAX\_COUNT = 500;

bool needToInit = false;

bool nightMode = false;

cv::CommandLineParser parser(argc, argv, "{@input||}{help h||}");

string input = parser.get<string>("@input");

if (parser.has("help"))

{

help();

return 0;

}

if (input.empty())

cap.open(0);

else if (input.size() == 1 && isdigit(input[0]))

cap.open(input[0] - '0');

else

cap.open(input);

if (!cap.isOpened())

{

cout << "Could not initialize capturing...\n";

return 0;

}

namedWindow("LK Demo", 1);

setMouseCallback("LK Demo", onMouse, 0);

Mat gray, prevGray, image, frame;

vector<Point2f> points[2];

for (;;)

{

cap >> frame;

if (frame.empty())

break;

frame.copyTo(image);

cvtColor(image, gray, COLOR\_BGR2GRAY);

if (nightMode)

image = Scalar::all(0);

if (needToInit)

{

// automatic initialization

//goodFeaturesToTrack(gray, points[1], MAX\_COUNT, 0.01, (int)10, Mat(), 3, 0, 0.04);

//goodFeaturesToTrack(gray, points[1], MAX\_COUNT, 0.01, (int)10, Mat(), 3, (bool)0, 0.04);

cornerSubPix(gray, points[1], subPixWinSize, Size(-1, -1), termcrit);

addRemovePt = false;

}

else if (!points[0].empty())

{

vector<uchar> status;

vector<float> err;

if (prevGray.empty())

gray.copyTo(prevGray);

calcOpticalFlowPyrLK(prevGray, gray, points[0], points[1], status, err, winSize,

3, termcrit, 0, 0.001);

size\_t i, k;

for (i = k = 0; i < points[1].size(); i++)

{

if (addRemovePt)

{

if (norm(point - points[1][i]) <= 5)

{

addRemovePt = false;

continue;

}

}

if (!status[i])

continue;

points[1][k++] = points[1][i];

circle(image, points[1][i], 3, Scalar(0, 255, 0), -1, 8);

}

points[1].resize(k);

}

if (addRemovePt && points[1].size() < (size\_t)MAX\_COUNT)

{

vector<Point2f> tmp;

tmp.push\_back(point);

cornerSubPix(gray, tmp, winSize, Size(-1, -1), termcrit);

points[1].push\_back(tmp[0]);

addRemovePt = false;

}

needToInit = false;

imshow("LK Demo", image);

char c = (char)waitKey(10);

if (c == 27)

break;

switch (c)

{

case 'r':

needToInit = true;

break;

case 'c':

points[0].clear();

points[1].clear();

break;

case 'n':

nightMode = !nightMode;

break;

}

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

cv::swap(prevGray, gray);

}

return 0;

}

Example 3

Camera calibration example:

Create a new project with camercal.cpp as source

// C:\opencv400\sources\samples\cpp\calibration.cpp: This file contains the 'main' function. Program execution begins and ends there.

//modification needed

1. Add this line to the top #pragma warning(disable : 4996) //\_CRT\_SECURE\_NO\_WARNINGS
2. cameraId=1 //change from cameraId = 0 //line 375
3. compile to generate calibration.exe and see where the program is, ie …….\x64\Debug
4. run windows/cmd, change directory to the directory …….\x64\Debug where yu can find calibration.exe
5. print the checker board pattern using the file below, stick it on a hard board <http://www.cse.cuhk.edu.hk/~khwong/www2/cmsc5711/ChessBoard.pdf>
6. run > calibration.exe -w=4 -h=5 -s=0.025 -o=camera.yml -op –oe
7. press ‘g’ to start, after calibration result is in the text file camera.yml
8. my test shows data: [ 6.0961138039145885e+02, 0., 3.4361528997983709e+02, 0., 6.0553104137756031e+02, 2.5534656363748795e+02, 0., 0., 1. ], meaning intrinsic matrix

mint=[609.6 0 343.6

1. 605.5 255.3

0 0 1]

#include "pch.h"

#pragma warning(disable : 4996) //\_CRT\_SECURE\_NO\_WARNINGS

//#include <iostream>

#include "opencv2/core.hpp"

#include <opencv2/core/utility.hpp>

#include "opencv2/imgproc.hpp"

#include "opencv2/calib3d.hpp"

#include "opencv2/imgcodecs.hpp"

#include "opencv2/videoio.hpp"

#include "opencv2/highgui.hpp"

#include <cctype>

#include <stdio.h>

#include <string.h>

#include <time.h>

#include <iostream>

using namespace cv;

using namespace std;

const char \* usage =

" \nexample command line for calibration from a live feed.\n"

" calibration -w=4 -h=5 -s=0.025 -o=camera.yml -op -oe\n"

" \n"

" example command line for calibration from a list of stored images:\n"

" imagelist\_creator image\_list.xml \*.png\n"

" calibration -w=4 -h=5 -s=0.025 -o=camera.yml -op -oe image\_list.xml\n"

" where image\_list.xml is the standard OpenCV XML/YAML\n"

" use imagelist\_creator to create the xml or yaml list\n"

" file consisting of the list of strings, e.g.:\n"

" \n"

"<?xml version=\"1.0\"?>\n"

"<opencv\_storage>\n"

"<images>\n"

"view000.png\n"

"view001.png\n"

"<!-- view002.png -->\n"

"view003.png\n"

"view010.png\n"

"one\_extra\_view.jpg\n"

"</images>\n"

"</opencv\_storage>\n";

const char\* liveCaptureHelp =

"When the live video from camera is used as input, the following hot-keys may be used:\n"

" <ESC>, 'q' - quit the program\n"

" 'g' - start capturing images\n"

" 'u' - switch undistortion on/off\n";

static void help()

{

printf("This is a camera calibration sample.\n"

"Usage: calibration\n"

" -w=<board\_width> # the number of inner corners per one of board dimension\n"

" -h=<board\_height> # the number of inner corners per another board dimension\n"

" [-pt=<pattern>] # the type of pattern: chessboard or circles' grid\n"

" [-n=<number\_of\_frames>] # the number of frames to use for calibration\n"

" # (if not specified, it will be set to the number\n"

" # of board views actually available)\n"

" [-d=<delay>] # a minimum delay in ms between subsequent attempts to capture a next view\n"

" # (used only for video capturing)\n"

" [-s=<squareSize>] # square size in some user-defined units (1 by default)\n"

" [-o=<out\_camera\_params>] # the output filename for intrinsic [and extrinsic] parameters\n"

" [-op] # write detected feature points\n"

" [-oe] # write extrinsic parameters\n"

" [-oo] # write refined 3D object points\n"

" [-zt] # assume zero tangential distortion\n"

" [-a=<aspectRatio>] # fix aspect ratio (fx/fy)\n"

" [-p] # fix the principal point at the center\n"

" [-v] # flip the captured images around the horizontal axis\n"

" [-V] # use a video file, and not an image list, uses\n"

" # [input\_data] string for the video file name\n"

" [-su] # show undistorted images after calibration\n"

" [-ws=<number\_of\_pixel>] # Half of search window for cornerSubPix (11 by default)\n"

" [-dt=<distance>] # actual distance between top-left and top-right corners of\n"

" # the calibration grid. If this parameter is specified, a more\n"

" # accurate calibration method will be used which may be better\n"

" # with inaccurate, roughly planar target.\n"

" [input\_data] # input data, one of the following:\n"

" # - text file with a list of the images of the board\n"

" # the text file can be generated with imagelist\_creator\n"

" # - name of video file with a video of the board\n"

" # if input\_data not specified, a live view from the camera is used\n"

"\n");

printf("\n%s", usage);

printf("\n%s", liveCaptureHelp);

}

enum { DETECTION = 0, CAPTURING = 1, CALIBRATED = 2 };

enum Pattern { CHESSBOARD, CIRCLES\_GRID, ASYMMETRIC\_CIRCLES\_GRID };

static double computeReprojectionErrors(

const vector<vector<Point3f> >& objectPoints,

const vector<vector<Point2f> >& imagePoints,

const vector<Mat>& rvecs, const vector<Mat>& tvecs,

const Mat& cameraMatrix, const Mat& distCoeffs,

vector<float>& perViewErrors)

{

vector<Point2f> imagePoints2;

int i, totalPoints = 0;

double totalErr = 0, err;

perViewErrors.resize(objectPoints.size());

for (i = 0; i < (int)objectPoints.size(); i++)

{

projectPoints(Mat(objectPoints[i]), rvecs[i], tvecs[i],

cameraMatrix, distCoeffs, imagePoints2);

err = norm(Mat(imagePoints[i]), Mat(imagePoints2), NORM\_L2);

int n = (int)objectPoints[i].size();

perViewErrors[i] = (float)std::sqrt(err\*err / n);

totalErr += err \* err;

totalPoints += n;

}

return std::sqrt(totalErr / totalPoints);

}

static void calcChessboardCorners(Size boardSize, float squareSize, vector<Point3f>& corners, Pattern patternType = CHESSBOARD)

{

corners.resize(0);

switch (patternType)

{

case CHESSBOARD:

case CIRCLES\_GRID:

for (int i = 0; i < boardSize.height; i++)

for (int j = 0; j < boardSize.width; j++)

corners.push\_back(Point3f(float(j\*squareSize),

float(i\*squareSize), 0));

break;

case ASYMMETRIC\_CIRCLES\_GRID:

for (int i = 0; i < boardSize.height; i++)

for (int j = 0; j < boardSize.width; j++)

corners.push\_back(Point3f(float((2 \* j + i % 2)\*squareSize),

float(i\*squareSize), 0));

break;

default:

CV\_Error(Error::StsBadArg, "Unknown pattern type\n");

}

}

static bool runCalibration(vector<vector<Point2f> > imagePoints,

Size imageSize, Size boardSize, Pattern patternType,

float squareSize, float aspectRatio,

float grid\_width, bool release\_object,

int flags, Mat& cameraMatrix, Mat& distCoeffs,

vector<Mat>& rvecs, vector<Mat>& tvecs,

vector<float>& reprojErrs,

vector<Point3f>& newObjPoints,

double& totalAvgErr)

{

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

if (flags & CALIB\_FIX\_ASPECT\_RATIO)

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

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

vector<vector<Point3f> > objectPoints(1);

calcChessboardCorners(boardSize, squareSize, objectPoints[0], patternType);

objectPoints[0][boardSize.width - 1].x = objectPoints[0][0].x + grid\_width;

newObjPoints = objectPoints[0];

objectPoints.resize(imagePoints.size(), objectPoints[0]);

double rms;

int iFixedPoint = -1;

if (release\_object)

iFixedPoint = boardSize.width - 1;

rms = calibrateCameraRO(objectPoints, imagePoints, imageSize, iFixedPoint,

cameraMatrix, distCoeffs, rvecs, tvecs, newObjPoints,

flags | CALIB\_FIX\_K3 | CALIB\_USE\_LU);

printf("RMS error reported by calibrateCamera: %g\n", rms);

bool ok = checkRange(cameraMatrix) && checkRange(distCoeffs);

if (release\_object) {

cout << "New board corners: " << endl;

cout << newObjPoints[0] << endl;

cout << newObjPoints[boardSize.width - 1] << endl;

cout << newObjPoints[boardSize.width \* (boardSize.height - 1)] << endl;

cout << newObjPoints.back() << endl;

}

objectPoints.clear();

objectPoints.resize(imagePoints.size(), newObjPoints);

totalAvgErr = computeReprojectionErrors(objectPoints, imagePoints,

rvecs, tvecs, cameraMatrix, distCoeffs, reprojErrs);

return ok;

}

static void saveCameraParams(const string& filename,

Size imageSize, Size boardSize,

float squareSize, float aspectRatio, int flags,

const Mat& cameraMatrix, const Mat& distCoeffs,

const vector<Mat>& rvecs, const vector<Mat>& tvecs,

const vector<float>& reprojErrs,

const vector<vector<Point2f> >& imagePoints,

const vector<Point3f>& newObjPoints,

double totalAvgErr)

{

FileStorage fs(filename, FileStorage::WRITE);

time\_t tt;

time(&tt);

struct tm \*t2 = localtime(&tt);

char buf[1024];

strftime(buf, sizeof(buf) - 1, "%c", t2);

fs << "calibration\_time" << buf;

if (!rvecs.empty() || !reprojErrs.empty())

fs << "nframes" << (int)std::max(rvecs.size(), reprojErrs.size());

fs << "image\_width" << imageSize.width;

fs << "image\_height" << imageSize.height;

fs << "board\_width" << boardSize.width;

fs << "board\_height" << boardSize.height;

fs << "square\_size" << squareSize;

if (flags & CALIB\_FIX\_ASPECT\_RATIO)

fs << "aspectRatio" << aspectRatio;

if (flags != 0)

{

sprintf(buf, "flags: %s%s%s%s",

flags & CALIB\_USE\_INTRINSIC\_GUESS ? "+use\_intrinsic\_guess" : "",

flags & CALIB\_FIX\_ASPECT\_RATIO ? "+fix\_aspectRatio" : "",

flags & CALIB\_FIX\_PRINCIPAL\_POINT ? "+fix\_principal\_point" : "",

flags & CALIB\_ZERO\_TANGENT\_DIST ? "+zero\_tangent\_dist" : "");

//cvWriteComment( \*fs, buf, 0 );

}

fs << "flags" << flags;

fs << "camera\_matrix" << cameraMatrix;

fs << "distortion\_coefficients" << distCoeffs;

fs << "avg\_reprojection\_error" << totalAvgErr;

if (!reprojErrs.empty())

fs << "per\_view\_reprojection\_errors" << Mat(reprojErrs);

if (!rvecs.empty() && !tvecs.empty())

{

CV\_Assert(rvecs[0].type() == tvecs[0].type());

Mat bigmat((int)rvecs.size(), 6, rvecs[0].type());

for (int i = 0; i < (int)rvecs.size(); i++)

{

Mat r = bigmat(Range(i, i + 1), Range(0, 3));

Mat t = bigmat(Range(i, i + 1), Range(3, 6));

CV\_Assert(rvecs[i].rows == 3 && rvecs[i].cols == 1);

CV\_Assert(tvecs[i].rows == 3 && tvecs[i].cols == 1);

//\*.t() is MatExpr (not Mat) so we can use assignment operator

r = rvecs[i].t();

t = tvecs[i].t();

}

//cvWriteComment( \*fs, "a set of 6-tuples (rotation vector + translation vector) for each view", 0 );

fs << "extrinsic\_parameters" << bigmat;

}

if (!imagePoints.empty())

{

Mat imagePtMat((int)imagePoints.size(), (int)imagePoints[0].size(), CV\_32FC2);

for (int i = 0; i < (int)imagePoints.size(); i++)

{

Mat r = imagePtMat.row(i).reshape(2, imagePtMat.cols);

Mat imgpti(imagePoints[i]);

imgpti.copyTo(r);

}

fs << "image\_points" << imagePtMat;

}

if (!newObjPoints.empty())

{

fs << "grid\_points" << newObjPoints;

}

}

static bool readStringList(const string& filename, vector<string>& l)

{

l.resize(0);

FileStorage fs(filename, FileStorage::READ);

if (!fs.isOpened())

return false;

size\_t dir\_pos = filename.rfind('/');

if (dir\_pos == string::npos)

dir\_pos = filename.rfind('\\');

FileNode n = fs.getFirstTopLevelNode();

if (n.type() != FileNode::SEQ)

return false;

FileNodeIterator it = n.begin(), it\_end = n.end();

for (; it != it\_end; ++it)

{

string fname = (string)\*it;

if (dir\_pos != string::npos)

{

string fpath = samples::findFile(filename.substr(0, dir\_pos + 1) + fname, false);

if (fpath.empty())

{

fpath = samples::findFile(fname);

}

fname = fpath;

}

else

{

fname = samples::findFile(fname);

}

l.push\_back(fname);

}

return true;

}

static bool runAndSave(const string& outputFilename,

const vector<vector<Point2f> >& imagePoints,

Size imageSize, Size boardSize, Pattern patternType, float squareSize,

float grid\_width, bool release\_object,

float aspectRatio, int flags, Mat& cameraMatrix,

Mat& distCoeffs, bool writeExtrinsics, bool writePoints, bool writeGrid)

{

vector<Mat> rvecs, tvecs;

vector<float> reprojErrs;

double totalAvgErr = 0;

vector<Point3f> newObjPoints;

bool ok = runCalibration(imagePoints, imageSize, boardSize, patternType, squareSize,

aspectRatio, grid\_width, release\_object, flags, cameraMatrix, distCoeffs,

rvecs, tvecs, reprojErrs, newObjPoints, totalAvgErr);

printf("%s. avg reprojection error = %.7f\n",

ok ? "Calibration succeeded" : "Calibration failed",

totalAvgErr);

if (ok)

saveCameraParams(outputFilename, imageSize,

boardSize, squareSize, aspectRatio,

flags, cameraMatrix, distCoeffs,

writeExtrinsics ? rvecs : vector<Mat>(),

writeExtrinsics ? tvecs : vector<Mat>(),

writeExtrinsics ? reprojErrs : vector<float>(),

writePoints ? imagePoints : vector<vector<Point2f> >(),

writeGrid ? newObjPoints : vector<Point3f>(),

totalAvgErr);

return ok;

}

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

{

Size boardSize, imageSize;

float squareSize, aspectRatio = 1;

Mat cameraMatrix, distCoeffs;

string outputFilename;

string inputFilename = "";

int i, nframes;

bool writeExtrinsics, writePoints;

bool undistortImage = false;

int flags = 0;

VideoCapture capture;

bool flipVertical;

bool showUndistorted;

bool videofile;

int delay;

clock\_t prevTimestamp = 0;

int mode = DETECTION;

int cameraId = 1;

vector<vector<Point2f> > imagePoints;

vector<string> imageList;

Pattern pattern = CHESSBOARD;

cv::CommandLineParser parser(argc, argv,

"{help ||}{w||}{h||}{pt|chessboard|}{n|10|}{d|1000|}{s|1|}{o|out\_camera\_data.yml|}"

"{op||}{oe||}{zt||}{a||}{p||}{v||}{V||}{su||}"

"{oo||}{ws|11|}{dt||}"

"{@input\_data|0|}");

if (parser.has("help"))

{

help();

return 0;

}

boardSize.width = parser.get<int>("w");

boardSize.height = parser.get<int>("h");

if (parser.has("pt"))

{

string val = parser.get<string>("pt");

if (val == "circles")

pattern = CIRCLES\_GRID;

else if (val == "acircles")

pattern = ASYMMETRIC\_CIRCLES\_GRID;

else if (val == "chessboard")

pattern = CHESSBOARD;

else

return fprintf(stderr, "Invalid pattern type: must be chessboard or circles\n"), -1;

}

squareSize = parser.get<float>("s");

nframes = parser.get<int>("n");

delay = parser.get<int>("d");

writePoints = parser.has("op");

writeExtrinsics = parser.has("oe");

bool writeGrid = parser.has("oo");

if (parser.has("a")) {

flags |= CALIB\_FIX\_ASPECT\_RATIO;

aspectRatio = parser.get<float>("a");

}

if (parser.has("zt"))

flags |= CALIB\_ZERO\_TANGENT\_DIST;

if (parser.has("p"))

flags |= CALIB\_FIX\_PRINCIPAL\_POINT;

flipVertical = parser.has("v");

videofile = parser.has("V");

if (parser.has("o"))

outputFilename = parser.get<string>("o");

showUndistorted = parser.has("su");

if (isdigit(parser.get<string>("@input\_data")[0]))

cameraId = parser.get<int>("@input\_data");

else

inputFilename = parser.get<string>("@input\_data");

int winSize = parser.get<int>("ws");

float grid\_width = squareSize \* (boardSize.width - 1);

bool release\_object = false;

if (parser.has("dt")) {

grid\_width = parser.get<float>("dt");

release\_object = true;

}

if (!parser.check())

{

help();

parser.printErrors();

return -1;

}

if (squareSize <= 0)

return fprintf(stderr, "Invalid board square width\n"), -1;

if (nframes <= 3)

return printf("Invalid number of images\n"), -1;

if (aspectRatio <= 0)

return printf("Invalid aspect ratio\n"), -1;

if (delay <= 0)

return printf("Invalid delay\n"), -1;

if (boardSize.width <= 0)

return fprintf(stderr, "Invalid board width\n"), -1;

if (boardSize.height <= 0)

return fprintf(stderr, "Invalid board height\n"), -1;

if (!inputFilename.empty())

{

if (!videofile && readStringList(samples::findFile(inputFilename), imageList))

mode = CAPTURING;

else

capture.open(samples::findFileOrKeep(inputFilename));

}

else

capture.open(cameraId);

if (!capture.isOpened() && imageList.empty())

return fprintf(stderr, "Could not initialize video (%d) capture\n", cameraId), -2;

if (!imageList.empty())

nframes = (int)imageList.size();

if (capture.isOpened())

printf("%s", liveCaptureHelp);

namedWindow("Image View", 1);

for (i = 0;; i++)

{

Mat view, viewGray;

bool blink = false;

if (capture.isOpened())

{

Mat view0;

capture >> view0;

view0.copyTo(view);

}

else if (i < (int)imageList.size())

view = imread(imageList[i], 1);

if (view.empty())

{

if (imagePoints.size() > 0)

runAndSave(outputFilename, imagePoints, imageSize,

boardSize, pattern, squareSize, grid\_width, release\_object, aspectRatio,

flags, cameraMatrix, distCoeffs,

writeExtrinsics, writePoints, writeGrid);

break;

}

imageSize = view.size();

if (flipVertical)

flip(view, view, 0);

vector<Point2f> pointbuf;

cvtColor(view, viewGray, COLOR\_BGR2GRAY);

bool found;

switch (pattern)

{

case CHESSBOARD:

found = findChessboardCorners(view, boardSize, pointbuf,

CALIB\_CB\_ADAPTIVE\_THRESH | CALIB\_CB\_FAST\_CHECK | CALIB\_CB\_NORMALIZE\_IMAGE);

break;

case CIRCLES\_GRID:

found = findCirclesGrid(view, boardSize, pointbuf);

break;

case ASYMMETRIC\_CIRCLES\_GRID:

found = findCirclesGrid(view, boardSize, pointbuf, CALIB\_CB\_ASYMMETRIC\_GRID);

break;

default:

return fprintf(stderr, "Unknown pattern type\n"), -1;

}

// improve the found corners' coordinate accuracy

if (pattern == CHESSBOARD && found) cornerSubPix(viewGray, pointbuf, Size(winSize, winSize),

Size(-1, -1), TermCriteria(TermCriteria::EPS + TermCriteria::COUNT, 30, 0.0001));

if (mode == CAPTURING && found &&

(!capture.isOpened() || clock() - prevTimestamp > delay\*1e-3\*CLOCKS\_PER\_SEC))

{

imagePoints.push\_back(pointbuf);

prevTimestamp = clock();

blink = capture.isOpened();

}

if (found)

drawChessboardCorners(view, boardSize, Mat(pointbuf), found);

string msg = mode == CAPTURING ? "100/100" :

mode == CALIBRATED ? "Calibrated" : "Press 'g' to start";

int baseLine = 0;

Size textSize = getTextSize(msg, 1, 1, 1, &baseLine);

Point textOrigin(view.cols - 2 \* textSize.width - 10, view.rows - 2 \* baseLine - 10);

if (mode == CAPTURING)

{

if (undistortImage)

msg = format("%d/%d Undist", (int)imagePoints.size(), nframes);

else

msg = format("%d/%d", (int)imagePoints.size(), nframes);

}

putText(view, msg, textOrigin, 1, 1,

mode != CALIBRATED ? Scalar(0, 0, 255) : Scalar(0, 255, 0));

if (blink)

bitwise\_not(view, view);

if (mode == CALIBRATED && undistortImage)

{

Mat temp = view.clone();

undistort(temp, view, cameraMatrix, distCoeffs);

}

imshow("Image View", view);

char key = (char)waitKey(capture.isOpened() ? 50 : 500);

if (key == 27)

break;

if (key == 'u' && mode == CALIBRATED)

undistortImage = !undistortImage;

if (capture.isOpened() && key == 'g')

{

mode = CAPTURING;

imagePoints.clear();

}

if (mode == CAPTURING && imagePoints.size() >= (unsigned)nframes)

{

if (runAndSave(outputFilename, imagePoints, imageSize,

boardSize, pattern, squareSize, grid\_width, release\_object, aspectRatio,

flags, cameraMatrix, distCoeffs,

writeExtrinsics, writePoints, writeGrid))

mode = CALIBRATED;

else

mode = DETECTION;

if (!capture.isOpened())

break;

}

}

if (!capture.isOpened() && showUndistorted)

{

Mat view, rview, map1, map2;

initUndistortRectifyMap(cameraMatrix, distCoeffs, Mat(),

getOptimalNewCameraMatrix(cameraMatrix, distCoeffs, imageSize, 1, imageSize, 0),

imageSize, CV\_16SC2, map1, map2);

for (i = 0; i < (int)imageList.size(); i++)

{

view = imread(imageList[i], 1);

if (view.empty())

continue;

//undistort( view, rview, cameraMatrix, distCoeffs, cameraMatrix );

remap(view, rview, map1, map2, INTER\_LINEAR);

imshow("Image View", rview);

char c = (char)waitKey();

if (c == 27 || c == 'q' || c == 'Q')

break;

}

}

return 0;

}