UNIVERSITÉ Concordia UNIVERSITY
FACULTY OF ENGINEERING AND COMPUTER SCIENCE
Department of Computer Science and Software Engineering

COMP498G/691G COMPUTER VISION

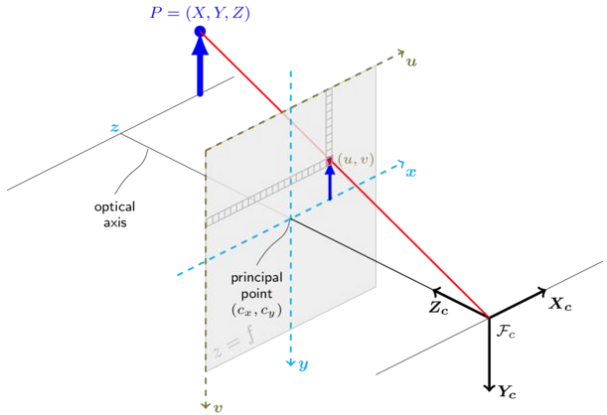
TUTORIAL 4
Camera Calibration

OpenCV Camera Model

OpenCV camera calibration is based on a pinhole camera model.

$$s\, m' = A[R|t]M'$$

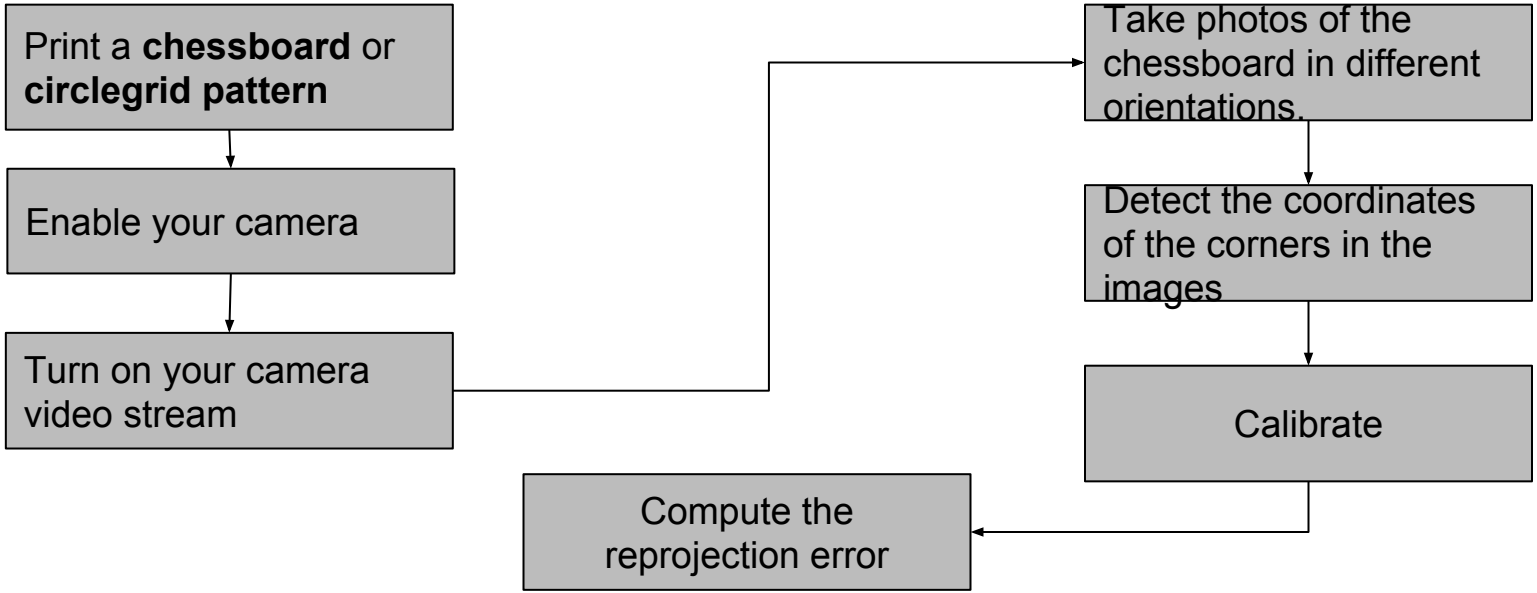
$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$



camera model

http://docs.opencv.org/3.1.0/dc/dbb/tutorial_py_calibration.html

Camera Calibration Procedure



http://docs.opencv.org/3.1.0/dc/dbb/tutorial_py_calibration.html

OpenCV Calib3d library

Calibrate Interface

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))

notes:

objpoints: std::vector<std::vector<cv::Point3f> >

imgpoints: std::vector<std::vector<cv::Point2f> >

http://docs.opencv.org/3.1.0/dc/dbb/tutorial_py_calibration.html

OpenCV Calib3d library

Internal implementation:

Solver for
Ax=b and Ax=0 type
of problems.

Finds an object pose from 3D-2D point correspondences:
bool solvePnP(InputArray **objectPoints**, InputArray **imagePoints**, InputArray **cameraMatrix**, InputArray **distCoeffs**, OutputArray **rvec**, OutputArray **tvec**, bool **useExtrinsicGuess**=false, int **flags**=ITERATIVE)

Or Finds an object pose from 3D-2D point correspondences using the RANSAC scheme
void solvePnPRansac(InputArray **objectPoints**, InputArray **imagePoints**, InputArray **cameraMatrix**, InputArray **distCoeffs**, OutputArray **rvec**, OutputArray **tvec**, bool **useExtrinsicGuess**=false, int **iterationsCount**=100, float **reprojectionError**=8.0, int **minInliersCount**=100, OutputArray **inliers**=noArray(), int **flags**=ITERATIVE)

http://docs.opencv.org/3.1.0/dc/dbb/tutorial_py_calibration.html

OpenCV Calib3d library

Find chessboard features

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

void drawChessboardCorners(InputOutputArray **image**, Size **patternSize**, InputArray **corners**, bool **patternWasFound**)

http://docs.opencv.org/3.1.0/dc/dbb/tutorial_py_calibration.html

OpenCV Calib3d library

Find circlegrid features

bool findCirclesGrid(InputArray **image**, Size **patternSize**, OutputArray **centers**, int **flags**=CALIB_CB_SYMMETRIC_GRID, const Ptr<FeatureDetector>& **blobDetector**=new SimpleBlobDetector())

void drawChessboardCorners(InputOutputArray **image**, Size **patternSize**, InputArray **corners**, bool **patternWasFound**)

http://docs.opencv.org/3.1.0/dc/dbb/tutorial_py_calibration.html

OpenCV Calib3d library

Now we take the advantage of the features2d library, refine the corners.

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

http://docs.opencv.org/3.1.0/dc/dbb/tutorial_py_calibration.html

OpenCV Calib3d library

The last step: checking your calibration quality

```
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)
{
/** fill up your code here **/
}
```

http://docs.opencv.org/3.1.0/dc/dbb/tutorial_py_calibration.html

OpenCV Camera Calibration example

https://github.com/opencv/opencv/tree/master/samples/cpp/tutorial_code/calib3d/camera_calibration

OpenCV Calib3d library

Other Interesting Function in Calib3d

1) Project object coordinates to image coordinates

void projectPoints(InputArray **objectPoints**, InputArray **rvec**, InputArray **tvec**, InputArray **cameraMatrix**, InputArray **distCoeffs**, OutputArray **imagePoints**, OutputArray **jacobian**=noArray(), double **aspectRatio**=0)

2) Find an initial camera matrix

Mat initCameraMatrix2D(InputArrayOfArrays **objectPoints**, InputArrayOfArrays **imagePoints**, Size **imageSize**, double **aspectRatio**=1.)

3) Decompose projection matrix to rotation and camera matrix

void decomposeProjectionMatrix(InputArray **projMatrix**, OutputArray **cameraMatrix**, OutputArray **rotMatrix**, OutputArray **transVect**, OutputArray **rotMatrixX**=noArray(), OutputArray **rotMatrixY**=noArray(), OutputArray **rotMatrixZ**=noArray(), OutputArray **eulerAngles**=noArray())

....

http://docs.opencv.org/3.1.0/dc/dbb/tutorial_py_calibration.html

OpenCV Calib3d library

Other Interesting Function in Calib3d (Continued)

4) **fisheye: OpenCV's fisheye camera model**

5) undistort your image (not in calib3d library)

void undistort(InputArray **src**, OutputArray **dst**, InputArray **cameraMatrix**, InputArray **distCoeffs**, InputArray **newCameraMatrix**=noArray())

http://docs.opencv.org/3.1.0/dc/dbb/tutorial_py_calibration.html

CONCORDIA.CA