Important Notes:Camera Calibration Prototype

Notes:

1. Explore using different “find chessboard pattern” functions in OpenCV:
2. findChessBoardCorners()

**Description:** The function attempts to determine whether the input image is a view of the chessboard pattern and locate the internal chessboard corners. The function returns a non-zero value if all of the corners are found and they are placed in a certain order (row by row, left to right in every row). Otherwise, if the function fails to find all the corners or reorder them, it returns 0. For example, a regular chessboard has 8 x 8 squares and 7 x 7 internal corners, that is, points where the black squares touch each other. The detected coordinates are approximate, and to determine their positions more accurately, the function calls cornerSubPix. You also may use the function cornerSubPix with different parameters if returned coordinates are not accurate enough.

**Restrictions:** The function requires white space (like a square-thick border, the wider the better) around the board to make the detection more robust in various environments. Otherwise, if there is no border and the background is dark, the outer black squares cannot be segmented properly and so the square grouping and ordering algorithm fails. It is possible that we will need to calibrate the camera with a chessboard that contains more whitespace around the borders to successfully return corners of interior points

1. findChessBoardCornersSB()

**Description:** The function is analog to [**findChessboardCorners**](https://docs.opencv.org/4.5.5/d9/d0c/group__calib3d.html#ga93efa9b0aa890de240ca32b11253dd4a) but uses a localized radon transformation approximated by box filters being more robust to all sort of noise, faster on larger images and is able to directly return the sub-pixel position of the internal chessboard corners. The Method is based on the paper [**[62]**](https://docs.opencv.org/4.5.5/d0/de3/citelist.html#CITEREF_duda2018) "Accurate Detection and Localization of Checkerboard Corners for Calibration" demonstrating that the returned sub-pixel positions are more accurate than the one returned by cornerSubPix allowing a precise camera calibration for demanding applications.In the case, the flags [**CALIB\_CB\_LARGER**](https://docs.opencv.org/4.5.5/d9/d0c/group__calib3d.html#gga9007b81edae8e7ead89219b316c109fba8dc6198c5d486c78911a60cba58d0835) or [**CALIB\_CB\_MARKER**](https://docs.opencv.org/4.5.5/d9/d0c/group__calib3d.html#gga9007b81edae8e7ead89219b316c109fba9a4682e44762f5fc90b34f6ec95cce0e) are given, the result can be recovered from the optional meta array. Both flags are helpful to use calibration patterns exceeding the field of view of the camera. These oversized patterns allow more accurate calibrations as corners can be utilized, which are as close as possible to the image borders. For a consistent coordinate system across all images, the optional marker (see image below) can be used to move the origin of the board to the location where the black circle is located.

**Restrictions:** The function requires a white boarder with roughly the same width as one of the checkerboard fields around the whole board to improve the detection in various environments. In addition, because of the localized radon transformation it is beneficial to use round corners for the field corners which are located on the outside of the board.

1. findCirclesGrid()

**Description**: The function attempts to determine whether the input image contains a grid of circles. If it is, the function locates centers of the circles. The function returns a non-zero value if all of the centers have been found and they have been placed in a certain order (row by row, left to right in every row). Otherwise, if the function fails to find all the corners or reorder them, it returns 0.

**Restrictions:** The function requires white space (like a square-thick border, the wider the better) around the board to make the detection more robust in various environments.

**Current Issues/Troubleshooting**

1. “findchessboardcorners()” and “findchessboardcornersSB()” methods are not correctly identifying position of all interior corner points in chessboard and thus returning “None” and “False” for “ret” and “Corners” respectively

* **Potential Solutions**: Print out other chessboard pattern images with moderate to large amount of whitespace on perimeter of board and pass into algorithm to determine if corners can be identified