<https://blog.dgut.top/2020/07/20/opencv-biaoding/>

<https://blog.dgut.top/2020/07/15/python-aruco/>

|  |  |  |  |
| --- | --- | --- | --- |
| White Balance Auto | Toggle | On/Off | **awb\_mode** |
| White Balance Setting | 2x Range | Red, Blue (0.0-8.0) | **awb\_gains** |
| Brightness | Range | (0-100) | **brightness** |
| Contrast | Range | (-100, 100) | **contrast** |
|  |  |  | **exposure\_mode** |
| Exposure Compensation | Range | (-25,25) | **exposure\_compensation** |
|  | Read Only Int |  | **exposure\_speed** |

Calibration script:

1. Line up Robot where center of wheelbase is directly over center of calibration page.
2. Robot drives back 1 meter. Verify distance
3. Average the 3D location of all the ArUco Markers. This point should be exactly 0 along the x-axis, whatever the camera is reading is the angular error. Save this as an x offset.
4. Using Y and Z, calculate how far away the point is from the camera. This value should be sqrt(1+(camera height)^2). Perform division to get a scale factor for all the points.
5. Apply scale factor to markers, generate a plane using possibly [this](https://www.ilikebigbits.com/2017_09_25_plane_from_points_2.html) method
6. Save scale factor, plane, and x-offset in a Yaml, Origin of plane?

Point Localization Script

1. Load values saved from the Yaml file
2. Apply X-offset to point, then scale
3. Calculate line from origin to ArUco marker after offset
4. Calculate intersection of that line to the plane