TECHNICAL REPORT

Raspberry Pi AWB determination methods and their statistical analyses

Kim Miikki[*] School of Chemical Engineering,
Aalto University, Espoo, Finland

July 5, 2023

Abstract

The determination of a correct auto white balance (AWB) parameters in photography are essential for good color reproducibility and invariance in the same lightning conditions. The easiest way to determine AWB gains is to read the camera's AWB gain values. However, this paper demonstrates that the gray card calibration method is more accurate than the built-in AWB method. The major disadvantage of gray card calibration is the time consumption compared to an AWB reading. Two optimized gray card calibration methods are also presented as a much faster methods than the traditional and slow exhaustive search method.

Keywords: auto white balance, AWB, camera, photography.

1. Introduction

Before capturing photos or videos with a digital camera, it is important to determine the white balance of the scene to ensure a realistic color cast. A continuous AWB is not suitable for cases where determination of color changes is needed. Hence, the white balance must be locked for color casting consistency. On a Raspberry Pi camera, this can be done by giving fixed gain values for the red and blue channels and disabling the camera's default AWB function. The red gain (rgain) and blue gain (bgain) values amplifies the sensitivity of respective color channels, so they can be used to modify color temperature of the image in different lightning conditions[1].

---

* E-mail address: kim.miikki@aalto.fi

For determination of rgain and bgain values they can be directly read via the Raspberry Pi AWB algorithm. Another way to get these values, is to use the Gray World Theory (GWT), where the average values of each color channels are integrated to gray, i.e. $R_{avg} = G_{avb} = B_{avg}$ [2]. A gray or white card is used as a reference color for AWB calibration. This study compares two different iteration methods for determining bgain and rgain values. The first method is an accurate and exhaustive search calibration and the second is a faster calibration optimized with the Nelder-Mead method[3].

2. Materials and Methods

Raspberry Pi OS (Legacy) with desktop (Debian: 10 (buster), System: 32-bit, Release: 3.5.2023) was used for the camera software and python scripts.[4] The python functions for controlling the camera was imported from the picamera[5] library. The calibration programs in this report can be found from the author's GitHub repository[6].

2.1. Raspberry PI AWB algorithm

The Raspberry Pi's AWB algorithm uses a simple Bayesian approach to determine rgain and bgain values by defining a feasible set of illuminants and then selecting the illuminant from the set which most likely produce the resulting colors.[7] The acquired calibration values are used for adjusting and locking the white balance when capturing images or recording videos.

The Raspberry Pi AWB values can be read with the `'awb_gains.py'` program. The gains are obtained with the awb_gains function, which is a member of the picamera[8] (python module for the Raspberry Pi camera) object.

For repeated trials, a program named '`awbgains-rtrials.py`' can be used.

2.2. Exhaustive search AWB calibration

Exhaustive search calibration is a brute-force method to find the global minimum in 2 dimensional matrix, which rows and columns represent all possible combinations of RB-tuples (rgain, bgain -tuple). The matrix size can be controlled by adjusting red and blue gains steps. This method requires an outer and an inner loop in order to get all possible RB-tuples in the matrix. For each RB-tuple the camera AWB is set to its rgain and bgain values, a frame is captured, region of interest (ROI) is cropped, The R_{avg}, G_{avg} and B_{avg} are calculated and finally a color distance value is calculated with formula (1).

$$distance = \frac{|R_{avg} - G_{avg}| + |R_{avg} - B_{avg}| + |G_{avg} - B_{avg}|}{3} \qquad (1)$$

The position of the smallest distance value in the matrix gives the optimal rgain and bgain values. The next search area is then built around the optimal RB-tuple position, by reducing the search ranges for the rgain and bgain values. Also the rgain and bgain steps are divided by 10, and the next iterations are performed until the desired level of accuracy is achieved.

When using `‘mapgains.py -c’` python program 5 iterations will be performed, and rgain and bgain values are obtained with 4 decimals. Only shutter speed is asked when starting the program with the `'-c'.` Everything else is done automatically in this mode.

2.3. Nelder-Mead optimized AWB calibrations

In Nelder-Mead method the optimal rgain and bgain values are determined by optimizing an objective function by a downhill simplex method. Two different objective functions are used in this paper: the first search for the optimal gain values when the mean absolute error (MAE, formula 1) is as small as possible (i.e. minimum), and the second method search for the minimum of the mean squared error (MSE) in RGB color distributions. The MSE is calculated with the formula (2):

$$MSE = \frac{1}{N+1} \sum_{i=0}^{N} \frac{(R_i - G_i)^2 + (R_i - B_i)^2 + (G_i - B_i)^2}{3}, \qquad (2)$$

where $N = 255$ and R_i, G_i and B_i are color counts of value i for the respective color channels.

The seed values for rgain and bgains, which are required for the optimization algorithm, are both read with the Raspberry Pi awb_gains function.

Nelder-Mead MAE optimization can be performed with '`awbgains.py`' and the MSE optimization with '`awbgains-mse.py`'. Both programs accepts following arguments: -ss, -i and -n. The -ss requires the shutter speed in µs, -i specifies the maximum iterations and -n sets the number of repetition trials.

2.4. Determining the grayness level or normality of distributions

Normal distributed rgains and bgains, based on pseudo-random numbers, can be generated with ‘`gen-normgains.py`’. The mean rgain and bgain can be set with -r and -b arguments, and their standard deviations with respective -rs and -rb arguments. The trials count is set using the -n argument. For reproducibility the seed is set to 1 (the value can be changed from the source code). The distribution is saved in a CSV file, the statistics are displayed on screen and the distribution graphs are generated.

The grayness of an image can be evaluated with the ‘`grayness.py`’. The image is analyzed by giving its filename as a positional argument for the python script. The ‘`perfect_gray_threshold`’ variable is defined in the source code (default = 2000), but it’s value should be changed the reflect the required gray level based on: image size and degree of noise in the image. This program generates a color distribution plots, decision if the image is perfectly gray or not, and a MSE value which is an indicator of the grayness. A smaller value corresponds to better grayness.

Different calibration methods can be visually compared with the ‘`awbgains-plotter.py`’. It plots 1-5 distributions on the same graph with confidence interval (CI) rectangles for the distributions’ mean coordinates.

Distribution normality can be estimated with the ‘`awbgains-normality.py`’. It reads the first *awbgains-trials-* file, creates histogram plots fro the gains, Q-Q plots for both gains against normal distribution, tolerance limits (TI) plots for rgain and bgain values. Also a Shapiro-Wilk test is perform in order the accept or reject H0 hypothesis, where H0 is a normal distribution (probably) and the alternate hypothesis HA is not a normal distribution.

Two series of trials can be examined with the ‘`awbgains-ksanalysis.py`’. This program compares the distribution of all distances of coordinates between two sets. The test is very sensitive, so only a small number of trials is recommended (i.e. 10). The test is based on Kolmogorov-Smirnov test. The program reads the two first csv files which starts with the string ‘`awbgains-trials-`’. As a result, distribution histograms are created, a cumulative density graph is generated and similarity of distribution is accepted or rejected based on the p-value.

3. Results

One of the most important thing before conducting a calibration, is that the lightning environment should be as constant as possible. In particular, the color temperature (CT) should not vary at all, when determining the gain values for the red and the blue channels. In general, calibration should be avoided during the lamp's warm-up ramp. Figure 1 a) shows how the illuminance of a 3000K LED lamp changes over time after it has been turned on, b) and its RGB analysis as a function of time.

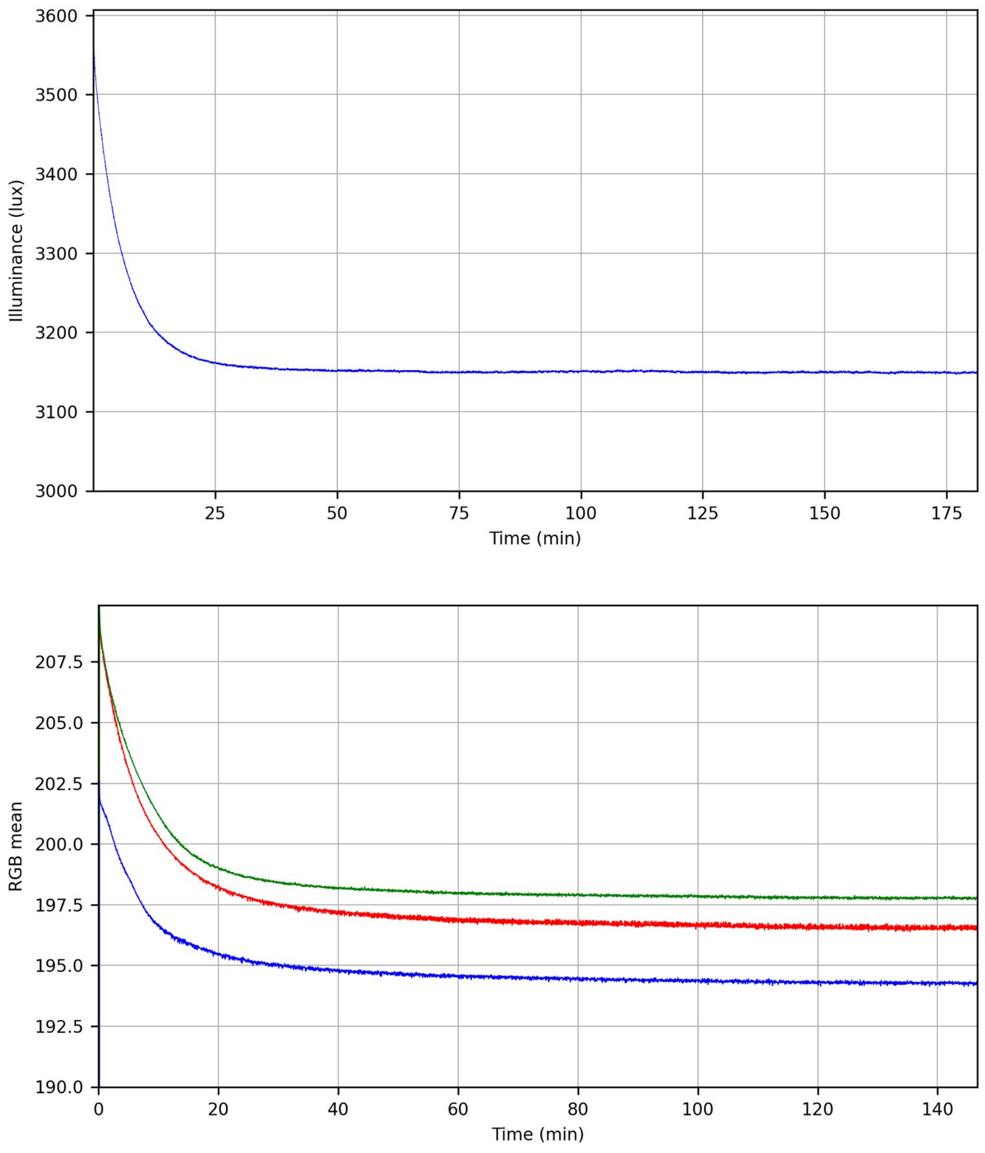
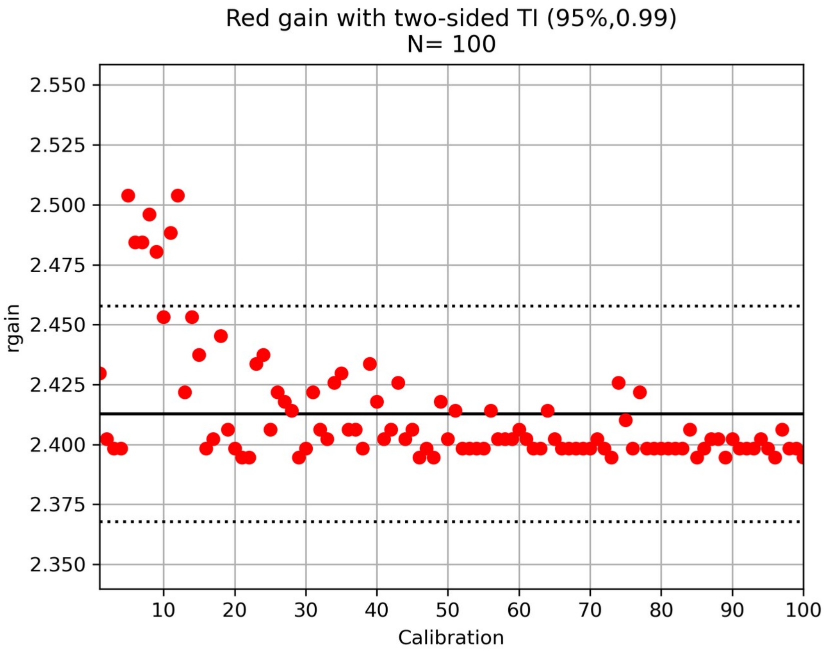


Fig. 1. a) Illuminance as a function of time for a 3000 K LED lamp, b) RGB analysis of the colors captured from the lamp.

The lamp case shows that the lamp should be on at least 1 hour before calibration. The rgain and bgain are presented in figure 2 right after the lamp was turned on, and the calibration was performed with *awbgains-rtrials.py*.



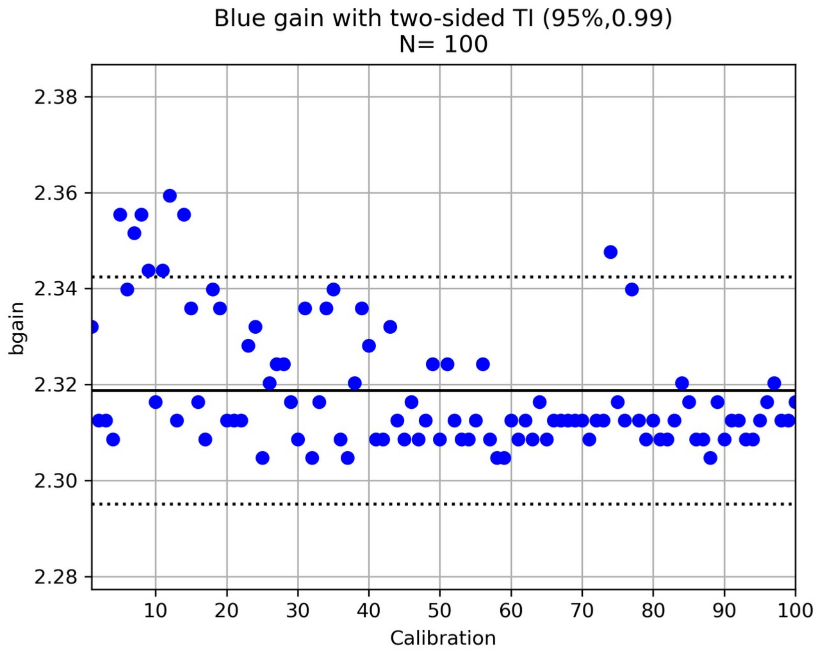


Figure 2. AWB gains of a warming led lamp.

3.1. Raspberry PI AWB algorithm

When performing repeated trials (e.g. reading of AWB gain values) a delay is required for the automatic gain adjustment to settle. When the lighting condition is constant, the distribution of AWB gains is discrete, which can be seen in Figure 3.

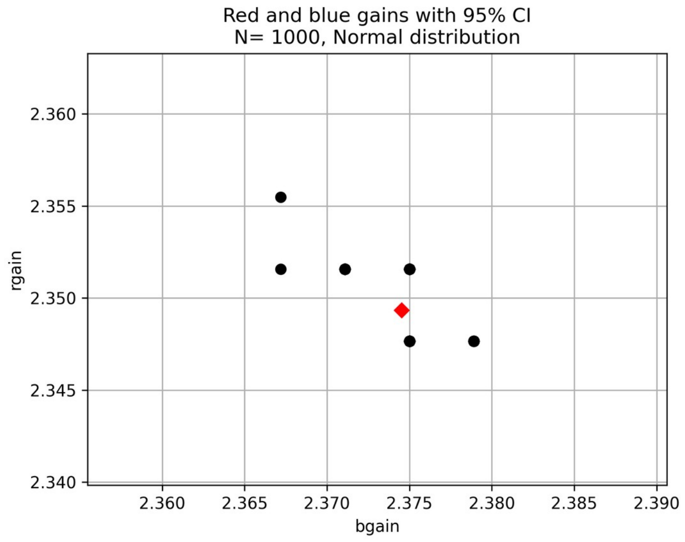


Figure 3. Raspberry Pi AWB trials with N = 1000.

In addition to the previous figure, the normality of the red and blue gain distributions can be verified with the awbgains-normality.py program. In the Shapiro-Wilk test, the p-value was 0 for both gains, hence both distributions are not normal distributions. In Figure 3 and 4, there is no noticeable drift, which is an indication of stable colored light.

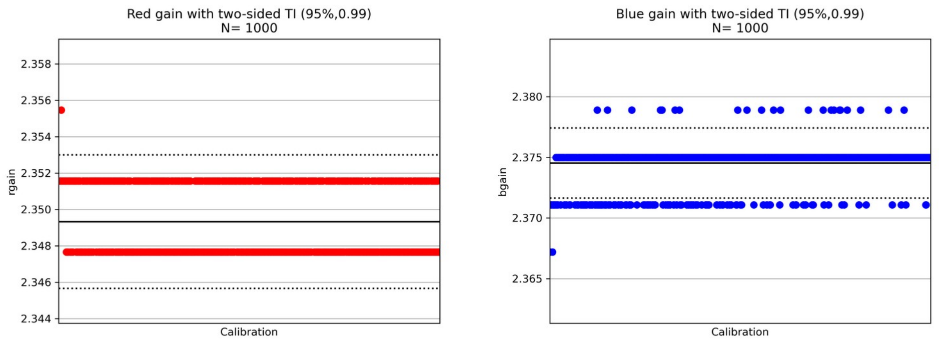


Figure 4. Time series of rgain and bgain with N = 1000 and interval = 2 s.

3.2. Exhaustive search AWB calibration

This method exhaustively searches for the global minimum of the color distance defined in formula 1. Typically, each iteration provides a tenfold precision for the gain values, so that the last round of iteration yields 4 decimals for the gain values at minimum color distance calculated by formula 1. The main disadvantage is the time consumed by this method, which is typically several minutes.

The exhaustive search is very suitable for the optimization, due to the high amount of local minima in the rgain, bagain-space, which are illustrated in figure 5.

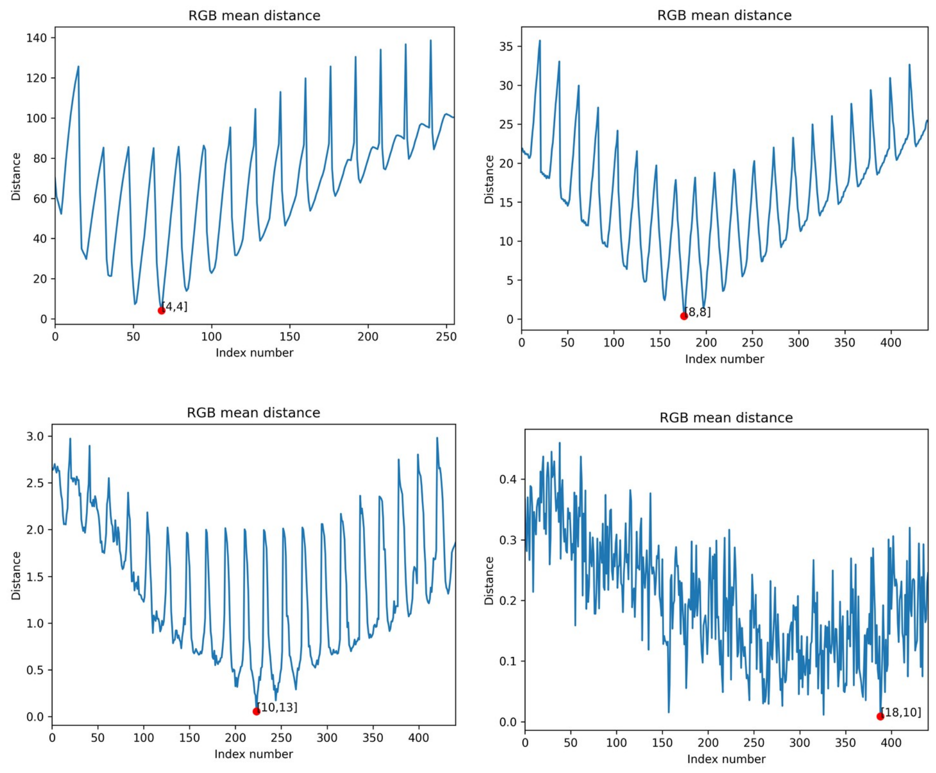


Figure 5. Mean color distances in 4 calibration iterations.

Repeated calibration experiments (N = 30) were performed with a 3000 K LED lamp by using '`mapgains.py -c`' at shutter speed of 1000 µs. From the first calibration, the seed values rgain, bgain for the rest of the calibrations were obtained. This values are printed on the terminal, and can also be found in the last file with the prefix: `rgbmap-cal-`. Here is a part of the output:

```
Calibration
-----------
Red gain : 2.3038
Blue gain: 2.3349
Minimal distance: 0.002
```

The remaining 29 calibrations were then performed using this command: `mapgains.py -c -pw`. The pw argument is a "precision wide" calibration when the target gain values are known. The advantage of this kind of calibration is that two iteration rounds can be skipped. But to ensure that the global minimum is found, the search area is also expanded.

In order to examine how the gains are distributed, the repeated calibration data can be easily analyzed by coping all files with prefix 'rgbcals-data-' into a analysis directory, and then executing this command: `awbgains.py -a rgbcals-data`. The results of the repeated trials are: rgain= 2.303 [95% CI: 2.302,2.303] and bgain= 2.334 [95% CI: 2.334,2.335]. The results are stored in the generated log file. Also TI (figure 6), CI and distribution (figure 7) graphs are created.

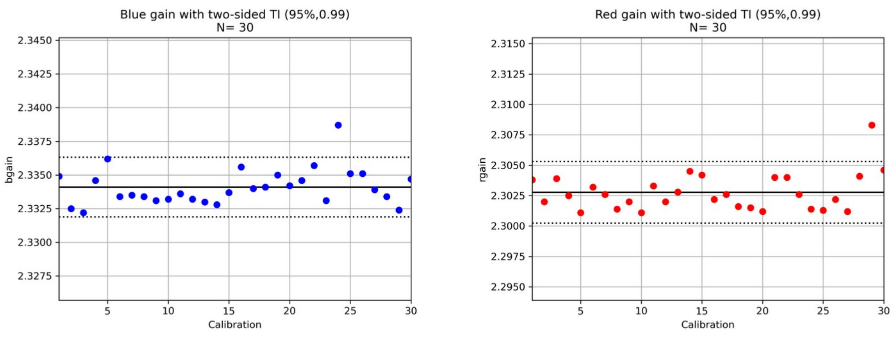


Figure 6. Tolerance interval plots for 99 % of rgain and bgain values with 95 % confidence.

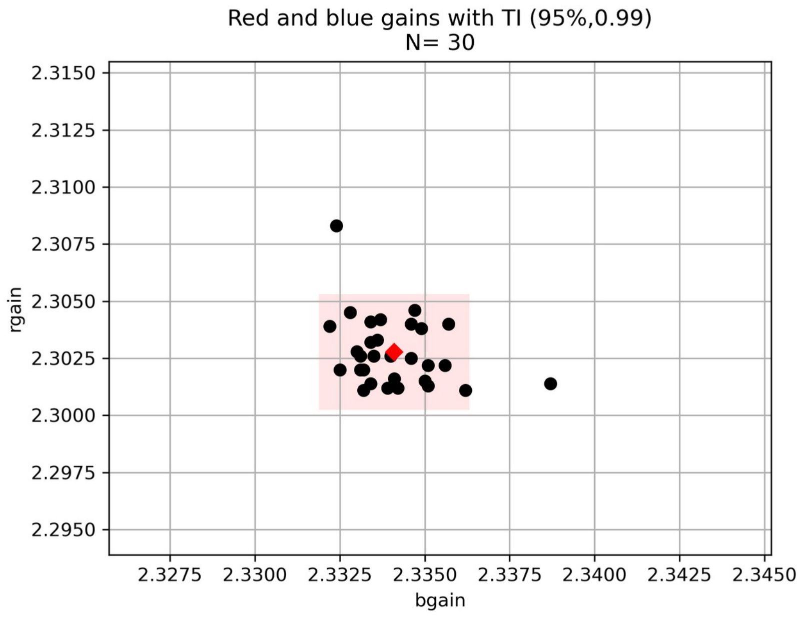


Figure 7. Red and blue gains distributions in a TI plot.

Normality of the gain distributions can be estimated with `'awbg-normality-py'`. The Shapiro-Wilk test rejected the normality for these trials. The p_{rgain}= 0.001 and p_{bgain}= 0.009 which are under 0.05, so the normal distribution hypothesis must be rejected for the both gains. Two outlier coordinates, as seen in figure 7, distorts the distribution and Q-Q plots (figures 8, 9).

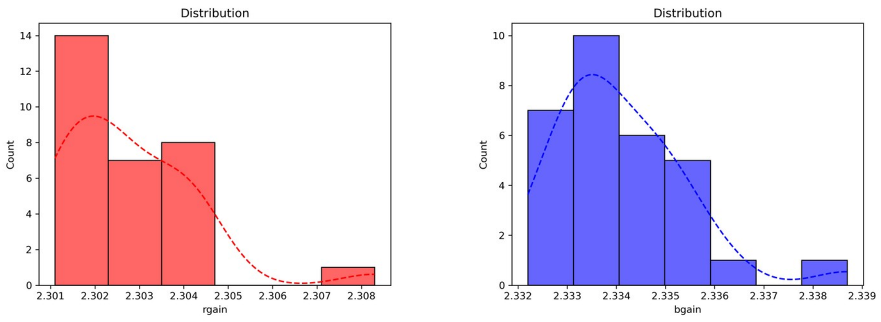


Figure 8. Red and blue gain distribution plots.

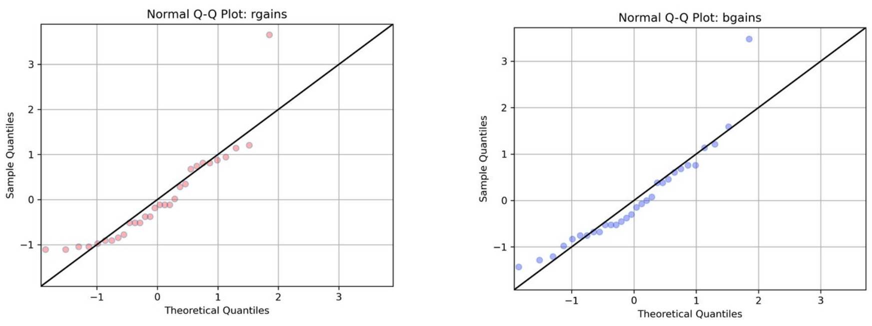


Figure 9. Q-Q plots for rgain and bgain.

The `mapgains.py` captures a sample image with the optimal rgain and bgain values. Its grayness can also be estimated with '`grayness.py`'. The MSE= 6890 for the last calibration at trial 30, which is over the threshold value (= 2000). Hence, the perfect gray hypothesis is rejected. The differences between the RGB channel distributions are shown in Figure 10.

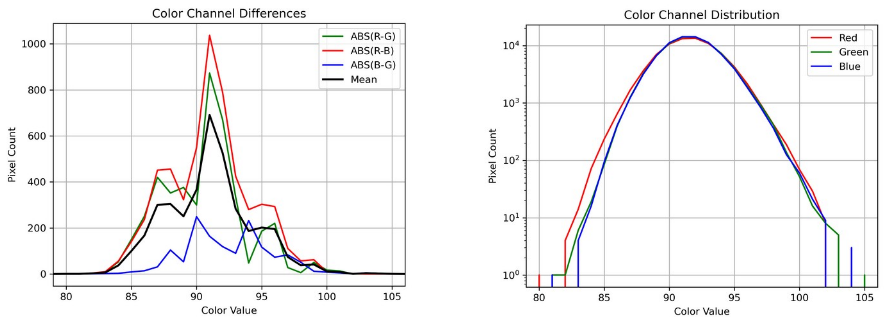


Figure 10. Distribution of color count differences and absolute color counts per channel.

Although these 30 trials did not show a normal distribution, the deviation from it does not appear to be very significant if outliers from the distribution are removed.

3.3. Nelder-Mead optimized AWB MAE calibration

This method optimize the MAE defined in formula (1) by using Nelder-Mead method. The 3000 K LED lamp was calibrated 30 times with the following command: '`awbgains.py -i 20 -n 30 -figs -ss 1000`'. Maximum iterations was set to 20, calibrations to 30 and the shutter speed to 1 ms (= 1000 μs). The convergence of both gain values to the optimal values are shown in Figure 11.

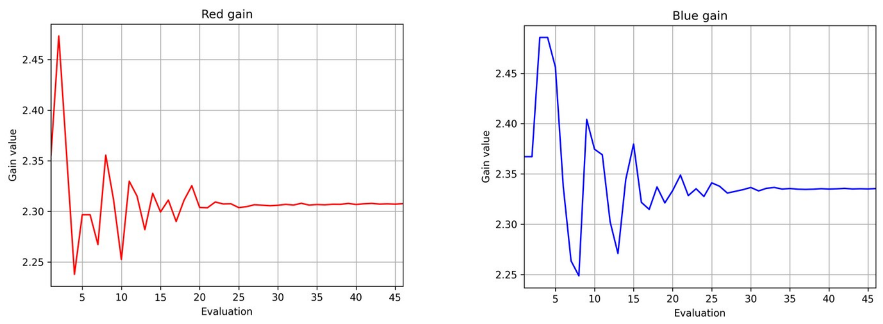


Figure 11. Convergence of rgain and bagin to optimal values using the Nelder-Mead MAE method.

The seed values for the gains were read from the Raspberry Pi AWB function. Duration of 30 trials, including generation of graphs, was 10.5 min. This calibration reached the minimum after 20 evaluation which is shown in figure 12.

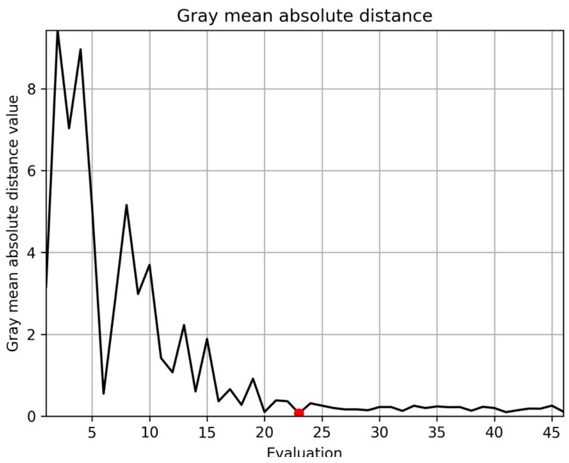


Figure 12. Convergence of MAE. Minimum MAE found after 20 evaluations.

The results of the repeated trials can be found in the generated log file: rgain= 2.303 [95% CI: 2.302,2.303] and bgain= 2.334 [95% CI: 2.333,2.334]. The obtained gains were same in this method and the exhaustive search method.

Normality of the gain distributions were estimated with `'awbgains-normality.py'`. The Shapiro-Wilk test rejected the normality for these trials. The p_{rgain}= 0.009 and p_{bgain}= 0.043. Both p values are under 0.05, so the normal distribution hypothesis are rejected for the both gains. The distributions are illustrated in figure 13.

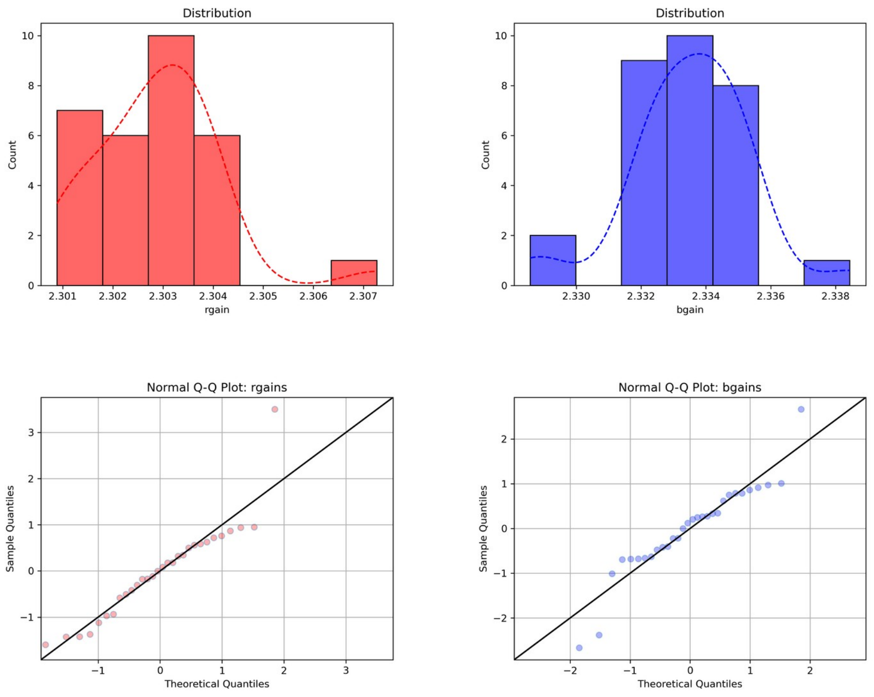


Figure 13. Nelder-Mead MAE method: rgain and bgain distribution on Q-Q plots.

The red and blue gains distributions are shown in figure 14 as coordinates with their tolerance intervals as a rectangle.

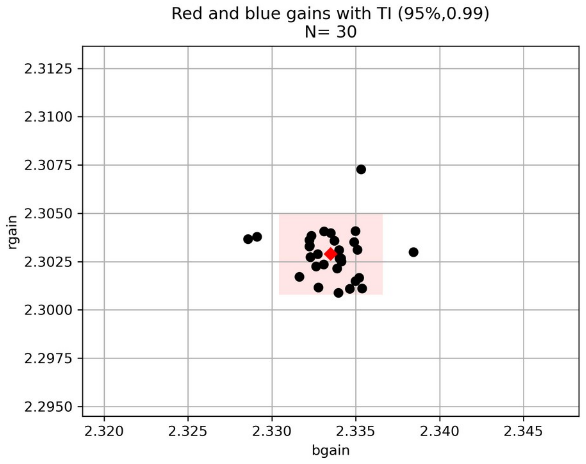


Figure 14. Nelder-Mead MAE method: gains distribution and their tolerance intervals.

A sample image (figure 15) captured by `awbgains.py` was analyzed with `grayness.py`. The MSE walue was 4431 and grayness with threshold 2000: not gray. The analysis results are shown in figure 16.

Figure 15. A sample image captured after calibration 30/30 with rgain= 2.303 and bgain= 2.333.

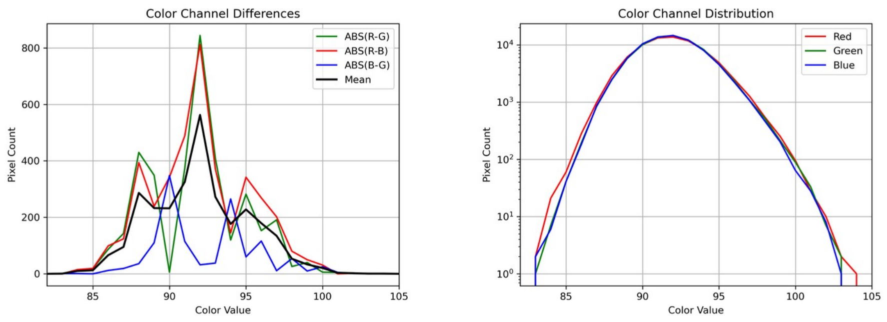


Figure 16. Distribution of color count differences and absolute color counts per channel.

The Nelder-Mead MAE is significantly faster method than the exhaustive search. Typically, the duration of one calibration is less than 20 s. The accuracy of the gains can be improved by conducting a series of trials, and calculating the average values of the calibrations. The awbgains.py supports multiple trials method, and following output was generated after the calibrations:

```
AWBgains log file created on 2023-06-26 04:29:46.667825

Shutter speed: 1000 μs
Iterations: 20

Trials: 30
p     : 0.95
Method: Student's t-distribution
```

```
Optimal distances:
Min   : 0.007
Max   : 0.064
Mean  : 0.025
Median: 0.022
SEM   : 0.003

rgain= 2.303±0.0
bgain= 2.334±0.001

rgain= 2.303 [95% CI: 2.302,2.303]
bgain= 2.334 [95% CI: 2.333,2.334]

Total time elapsed: 0:10:25.868558
```

3.4. Nelder-Mead optimized AWB MSE calibration

Another way to find the optimal gain values, is to change them objective function from MAE to MSE (formula 2). The seed values for rgain and and bgain are acquired from the Raspberry Pi AWB function. Thereafter ta calibration on proceeded in same way as AWB MAE, except for the objective function.

A 3000 K LED lamp was calibrated 30 times with the following command: '`awbgains-mse.py -i 20 -n 30 -figs -ss 1000`'. Maximum iterations was set to 20, calibrations to 30 and the shutter speed to 1 ms. The convergence of both gain values to the optimal values are shown in Figure 17.

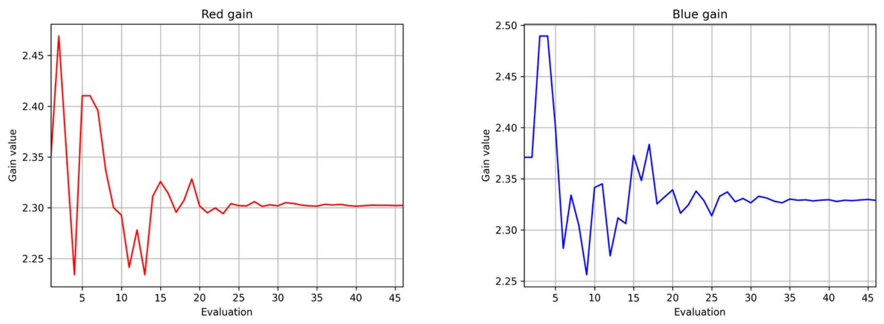


Figure 17. Convergence of rgain and bagin to optimal values using the Nelder-Mead MSE method.

With this method, the optimal gain values are also reached quickly. The decreasing trend of the MSE is shown in figure 20.

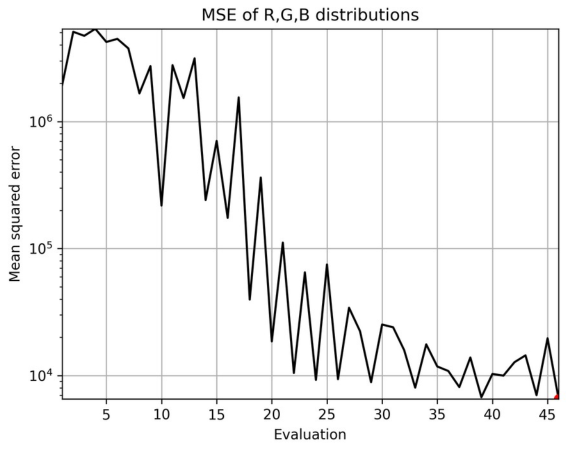


Figure 20. Convergence of MSE. Minimum is found after 45 evaluations.

The results of the repeated trials are: rgain= 2.303 [95% CI: 2.303,2.304] and bgain= 2.332 [95% CI: 2.331,2.333]. The obtained gains were very close to the exhaustive and AWB MAE methods. Minimum MSE found was 1816 which is under the threshold for non-gray verdict. Hence, at least one of the 30 calibrations was classified as perfect gray. The best value of color distance (formula 1) was 0.008.

Normality of the gain distributions were estimated with '`awbgains-normality.py`'. The Shapiro-Wilk test rejected the normality for these trials. The p_{rgain}= 0.011 and p_{bgain}= 0.015. Both p values are under 0.05, so the normal distribution hypothesis are rejected for the both gains. The distributions are shown in figure 21.

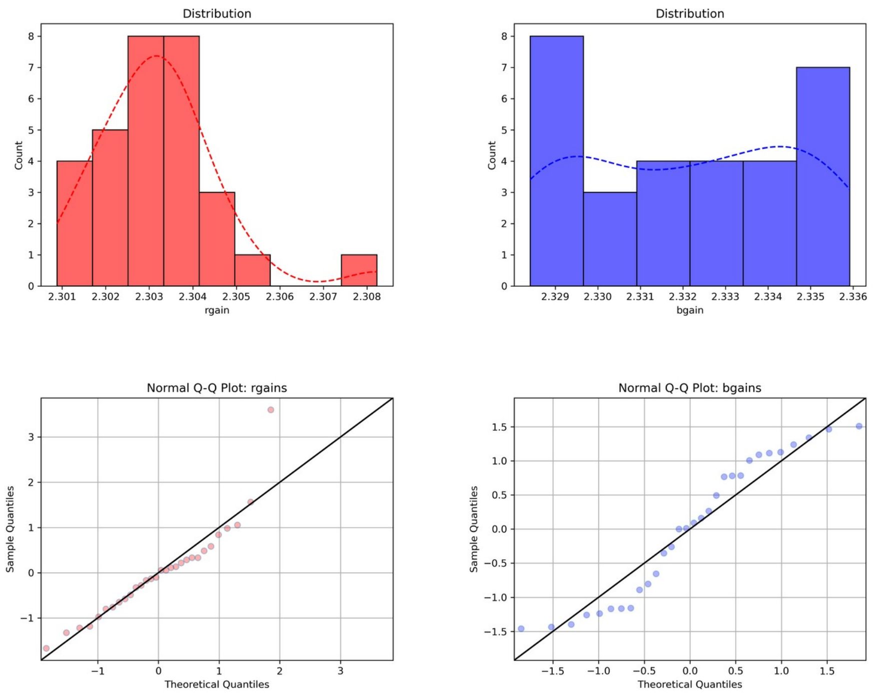


Figure 21. Nelder-Mead MSE method: rgain and bgain distribution on Q-Q plots.

The red and blue gains distributions are shown in figure 22 as calibration series with their tolerance intervals.

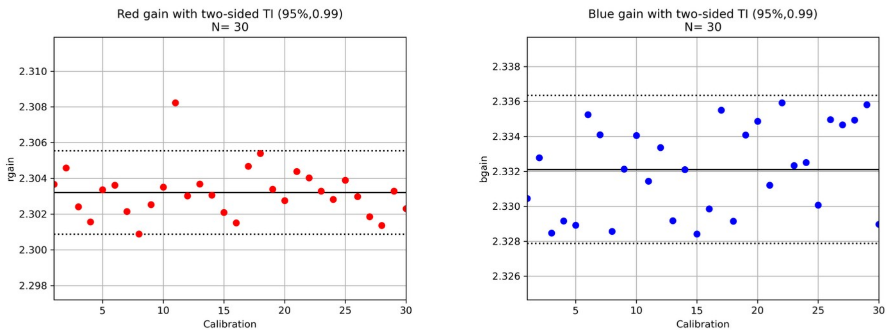


Figure 22. Nelder-Mead MSE method: gains distribution and their tolerance intervals as a function of calibration.

Finally the last sample image (figure 23) of 30 MSE calibrations was analyzed with `grayness.py`. The MSE was 4600, therefore the grayness verdict was: not gray. The color count differences are shown in figure 24.

Figure 23. A sample image captured after calibration 30/30 with rgain= 2.302 and bgain= 2.329.

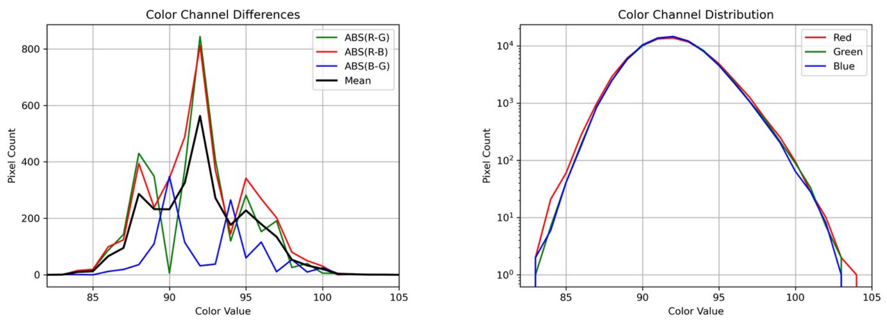


Figure 24. Distribution of color count differences and absolute color counts per channel.

Again, rgains and bgains were not normally distributed, perfect gray was not achieved, but gain values were similar to the other two calibration methods.

This is also a fast method. The calibrations were completed in 0:10:32.

3.5. Statistical comparison of AWB methods

Different ways to determining AWB gain values were compared based on the number of trials (N). The results are presented in the next two subsections.

3.5.1. Comparison of different methods for AWB calibration with N=10 trials

The average rgain and bgain values of the optimal AWB gain values are obtained by repeated trials per method. In the first comparison between the methods, the number of repeated trials was 10, which is practical for repetitions due to the relatively low time consumption.

The gain distributions and their means are shown in figure 25. From the zoomed image, the RPI AWB set was removed because its mean gain coordinate position differed significantly from the mean coordinates of the other methods.

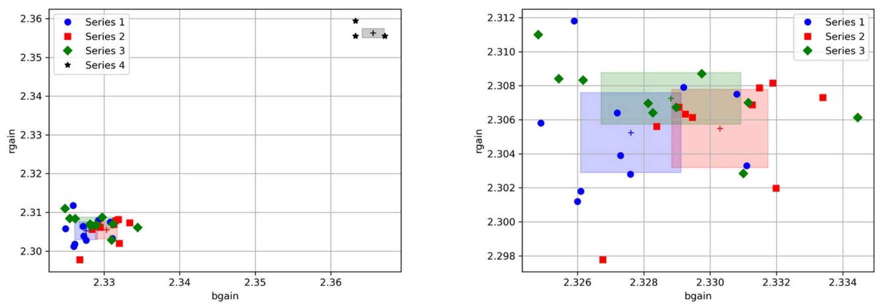


Figure 25. Gain distributions and their mean coordinates of four different methods with N = 10, CI = 95 % (1 = mapgain, 2 = MAE, 3 = MSE, 4 = RPI AWB).

All but the AWB RPI method were compared pairwise with the following scripts:

```
awbgains-hypom.py
awbgains-hypoc.py
awbgains-ksanalysis.py
```

The distributions (length between coordinates / series) for each pairwise comparison are shown in figures 26-28.

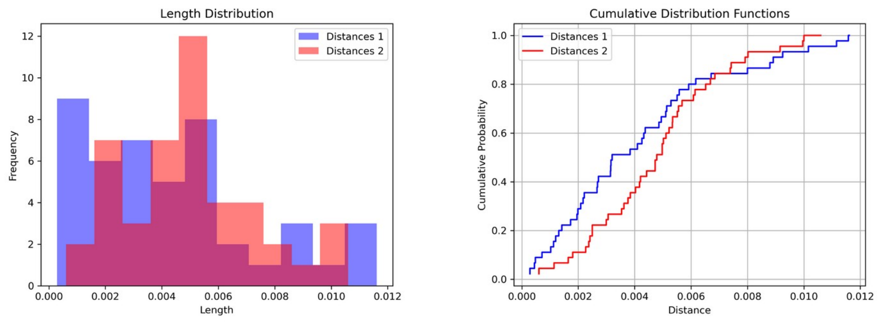


Figure 26. Pairwise comparison of MAE (1) and mapgains (2) distributions and CDFs with N = 10.

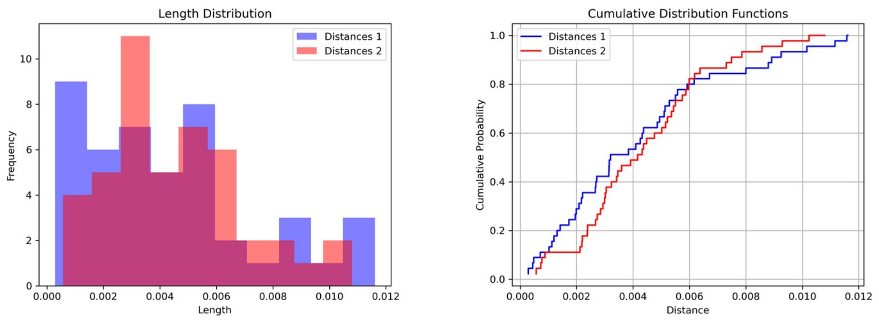


Figure 27. Pairwise comparison of MAE (1) and MSE (2) distributions and CDFs with N = 10.

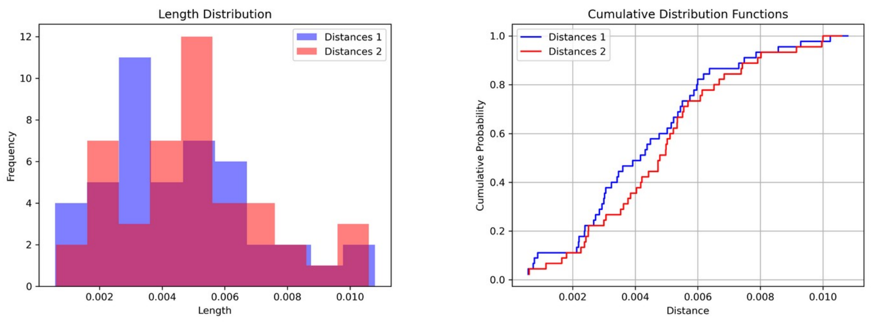


Figure 28. Pairwise comparison of MSE (1) and mapgains (2) distributions and CDFs with N = 10.

Three type of hypothesis were used in order to determining are the mean gains same, the mean gain coordinates same and the length distributions same. The results of these hypothesis are summarized in the following table:

Methods	**prgain**	p_{rgain}	$\mu_{rgain1}=\mu_{rgain2}$	p_{bgain}	$\mu_{bgain1}=\mu_{rgain2}$	$\mu_{coord1}=\mu_{coord2}$	**D**	$p_{distribution}$
MAE-mapgain		0.893	+	0.015	-	-	0.244	0.136
MAE-MSE		0.264	+	0.149	+	+	0.200	0.332
MSE-mapgain		0.076	+	0.243	+	+	0.177	0.480

Table 1. Pairwise student t-test between pairs of the methods (MAE, mapgain = exhaustive search, MSE). Plus sign is accepted hypothesis and minus sign rejected. Kolmogorov-Smirnov test results are presented in three last columns.

The KS-test indicates similar distributions for the three tests. There are no significant differences for the red gain in these three calibration methods. The blue gain differs significantly between MAE and mapgains methods with N=10. Hence, a the same mean gain coordinates hypothesis has to be rejected between MAE and mapgains methods. However, MAE-MSE and MSE-mapgain methods mean gain coordinates did not differ significantly from each other.

3.5.2. *Comparison of different methods for AWB calibration with N=30 trials*

In the first comparison between the methods, the number of repeated trials was 30. The mean gains per AWB methods are shown with 95 % confidence intervals in figure 29.

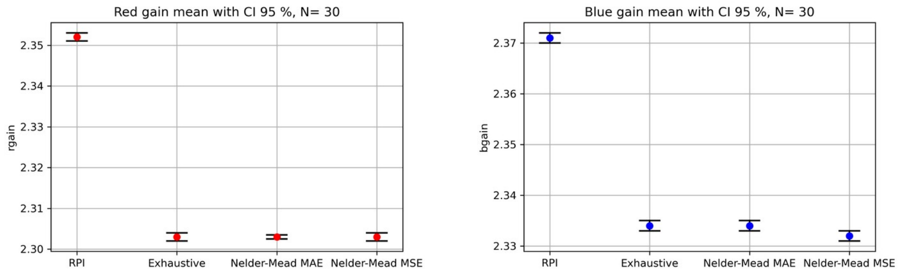


Figure 29.Read and blue gain means with N = 30, 95 % CI of four different AWB calibrations.

Again the RPI AWB method differs from the three other calibration methods. Hence, only three similar results are compared, which distributions are shown in figure 30.

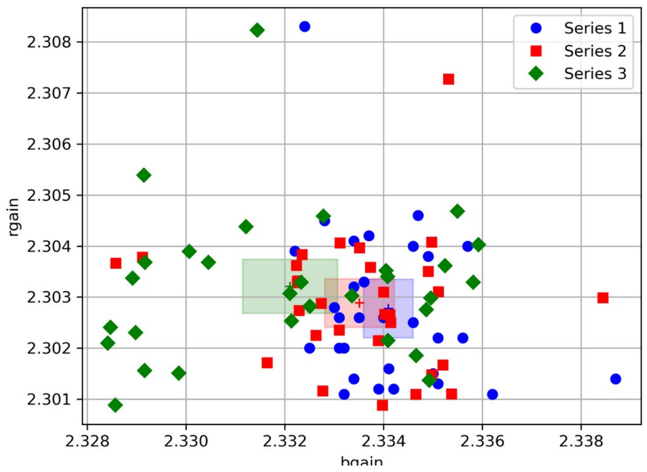


Figure 30. Gain distributions and their mean coordinates of three different methods with N = 30, CI = 95 % (1 = exhaustive search (mapgains), 2 = MAE, 3 = MSE).

The distributions (length between coordinates / series) for each pairwise comparison are shown in figures 31-33.

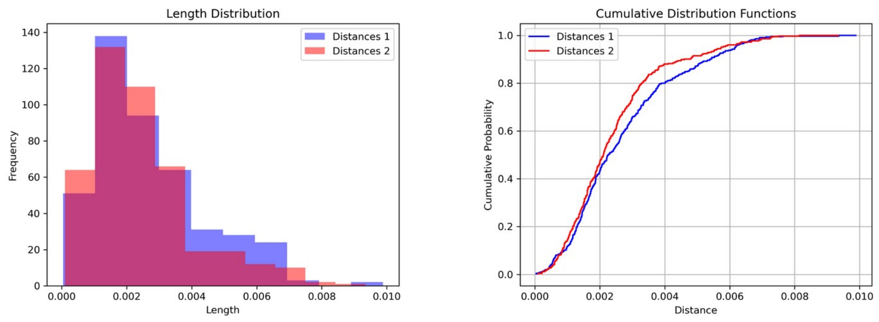


Figure 31. Pairwise comparison of MAE (1) and mapgains (2) distributions and CDFs with N = 30.

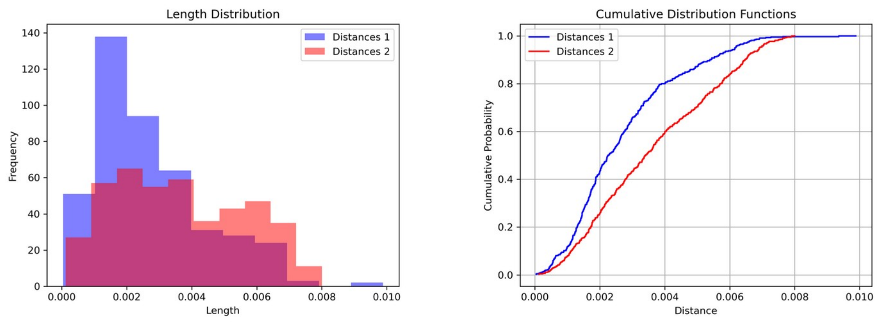


Figure 32. Pairwise comparison of MAE (1) and MSE (2) distributions and CDFs with N = 30.

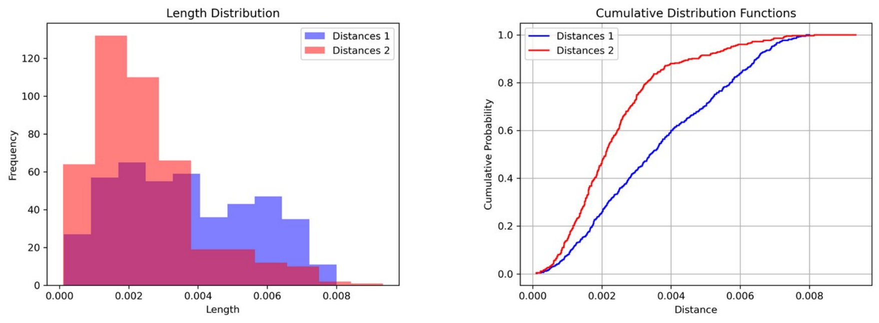


Figure 33. Pairwise comparison of MSE (1) and mapgains (2) distributions and CDFs with N = 30.

Three type of hypothesis were used in order to determine if the mean gains are same, the mean gain coordinates are same and the similarity of the length distributions. The results of these hypothesis are summarized in the following table:

Methods	**prgain**	**p_{rgain}**	**$\mu_{rgain1}=\mu_{r\text{-}gain2}$**	**p_{bgain}**	**$\mu_{bgain1}=\mu_{rgain2}$**	**$\mu_{coord1}=\mu_{co\text{-}ord2}$**	**D**	**$p_{distribution}$**
MAE-mapgain		0.745	+	0.161	+	+	0.103	0.019
MAE-MSE		0.360	+	0.042	-	-	0.237	4.2E-11
MSE-mapgain		0.252	+	0.001	-	-	0.319	3.4E-21

Table 2. Pairwise student t-test between pairs of the methods (MAE, mapgain = exhaustive search, MSE). Plus sign indicates an accepted hypothesis and minus sign rejected. Kolmogorov-Smirnov test results are presented in three last columns.

When increasing the trials to N=30, all KS-tests for similarity of distributions were rejected. However, MAE-mapgains KS-test gave a p-value of 0.019 which is not very far from the required 0.05. The two other test's p-values were extremely small, so the distributions similarity must be rejected. All methods, except the RPI AWB, had similar mean values for the red gain, but only MAE-mapgain comparison between mean blue gain was accepted. Hence, the mean coordinates for the gains are same in these two methods when N=30.

4. Conclusions

Several methods were presented how to obtain auto white balance values values with a Raspberry Pi camera system. The RPI AWB function differs significantly from the three other methods examined. The exhaustive search (mapgains) is a robust and slow method in order to obtain the calibration values rgain and bgain, with 4 decimals. By using a Nelder-Mead optimization, the iterations required for an AWB calibration can be significantly reduced. The accuracy of the of these fast iteration methods can be enhanced by using repeated calibrations. Good mean values for the gains can be achieved with N=5-10, without consuming to much time compared with the exhaustive search which typically requires several minutes to perform.

In this work, it was found that the RPI AWB values are highly discrete in an uniform unchanging lightning environment. The discreteness of the gain values can be used as proof of stability of the lightning conditions, especially when no drift of the values can be detected.

A method for determining perfect gray was presented, but the threshold or correct calculation of the threshold is not yet clear.

Whether the AWB calibration method is normally distributed or not can be determined using the `awbgains-normalty.py` program. Other python programs prefixed with 'awbgains' were introduced to capture and statistically analyze AWB calibrations.

Based on the results, the following recommendations are given: 1) RPI AWB can be used for immediate and coarse gain values, 2) MAE or MSE method for most calibrations and c) exhaustive search method to check the correctness of other methods or if 4 decimal places are required for AWB gain values.

References

[1] *The official raspberry pi camera guide 2020 the official raspberry pi camera guide 2020*. (2020). Cambridge, England: Raspberry Pi Press.

[2] Zapryanov, G., Ivanova, D., & Nikolova, I. (2012). Automatic white balance algorithms for digital still cameras – a comparative study. Information Technologies and Control, 1, 16-22.

[3] Nelder, J. A., & Mead, R. (1965). A simplex method for function minimization. *The computer journal*, *7*(4), 308-313.

[4] Raspberry Pi Ltd. (n.d.). *Operating system images*. Raspberry Pi. Retrieved June 14, 2023, from https://www.raspberrypi.com/software/operating-systems/

[5] Jones, D. (2017, February 2). *picamera — Picamera 1.13 Documentation*. Retrieved June 14, 2023, from https://picamera.readthedocs.io/en/release-1.13/

[6] Miikki, K. (2023, June 11). *GitHub - kmiikki/rpi-camera: Raspberry Pi High Speed Camera*. GitHub. Retrieved June 14, 2023, from https://github.com/kmiikki/rpi-camera/

[7] Raspberry Pi Camera Algorithm and Tuning Guide. Raspberry Pi Datasheets . (2021). June 13, 2023. https://datasheets.raspberrypi.com/camera/raspberry-pi-camera-guide.pdf

[8] Jones, D. (2017, February 25). *picamera*. PyPI. https://pypi.org/project/picamera/