



Hochschule Konstanz
Technik, Wirtschaft und Gestaltung

Signale, Systeme und Sensoren

Kalibrierung von digitalen Kameras

T. Schoch, L. Stratmann

Konstanz, 29. April 2019

Zusammenfassung (Abstract)

Thema:	Kalibrierung von digitalen Kameras	
Autoren:	T. Schoch	tobias.schoch@htwg-konstanz.de
	L. Stratmann	luca.stratmann@htwg-konstanz.de
Betreuer:	Prof. Dr. Matthias O. Franz Jürgen Keppler Christoph Kaiser	mfranz@htwg-konstanz.de juergen.keppler@htwg-konstanz.de ch241kai@htwg-konstanz.de

In dem zweiten Versuch der Versuchsreihe des Semesters haben wir die Eigenschaften von digitalen Kameras untersucht. Deshalb erfolgt mittels der Python Bibliothek OpenCV eine Kalibrierung der Kamerasensoren.

Wir nehmen mit einer Webcam ein Grauwertkeil auf und berechnen den Durchschnitt und die Standartabweichung jeder einzelnen Grauwertstufe. Zudem werden wir 10 Dunkelbilder und 10 Weißbilder aufnehmen und von diesen den pixelweisen Mittelwert der 10 Dunkel- und Weißbilder berechnen. Wir werden mithilfe des durchschnittlichen Dunkelbildes das thermische Ausleserauschen entfernen aus dem Bild des Grauwertkeils entfernen. Dadurch bleibt nur noch der Offset bzw. der Dunkelstrom jedes Pixels übrig und wir haben damit jeden Nullpunkt des Dunkelbildes bestimmt. In der dritten Aufgabe normieren wir das Weißbild und dividieren wir dieses durch den Abzug des Dunkelbildes von dem korrigierten Eingangs- bild. Dadurch erhalten wir die tatsächliche Intensität des einfallenden Lichtes. Im letzten Teil der Aufgabe überprüfen wir die Bilder auf funktionsuntüchtige Pixel wie zum Beispiel Hot, Stuck und Deadpixels.

Inhaltsverzeichnis

Abbildungsverzeichnis	IV
Tabellenverzeichnis	V
Listingverzeichnis	VI
1 Einleitung	1
2 Versuch 1: Aufnahme und Analyse eines Grauwertkeiles	2
2.1 Fragestellung, Messprinzip, Aufbau, Messmittel	2
2.2 Messwerte	4
2.3 Auswertung	5
2.4 Interpretation	7
3 Versuch 2: Aufnahme eines Dunkelbildes	8
3.1 Fragestellung, Messprinzip, Aufbau, Messmittel	8
3.2 Messwerte	9
3.3 Auswertung	10
3.4 Interpretation	11
4 Versuch 3: Aufnahme eines Weißbildes	12
4.1 Fragestellung, Messprinzip, Aufbau, Messmittel	12
4.2 Messwerte	13
4.3 Auswertung	14
4.4 Interpretation	16
5 Versuch 4: Aufnahme eines Weißbildes	17
5.1 Fragestellung, Messprinzip, Aufbau, Messmittel	17
5.2 Messwerte	18

5.3	Auswertung	19
5.4	Interpretation	21
Anhang		22
A.1	Quellcode	22
A.1.1	Quellcode Versuch 1	22
A.1.2	Quellcode Versuch 2	27
A.1.3	Quellcode Versuch 3	30
A.1.4	Quellcode Versuch 4	36
A.2	Kameraeinstellungen für OpenCV	40

Abbildungsverzeichnis

2.1	Aufbau im Labor	3
2.2	Das Bild des Grauwertkeiles	4
2.3	Die einzelnen Grauwerte und das zusätzliche Gesamtbild	5
2.4	Standardabweichung und Durchschnitt der jeweiligen Grauwerte	7
3.1	Eines der 10 aufgenommenen Dunkelbilder	9
3.2	Grauwertkeil und kontrastmaximierte Bild	10
3.3	Unterschied zwischen dem richtigen und dem korrigierten Bild	10
4.1	Weißbild	13
4.2	Grauwertkeil und kontrastmaximierte Bild	14
4.3	Dunkelbild subtrahiert vom Weißbild	14
4.4	Unterschied zwischen dem richtigen und dem korrigierten Bild	15
5.1	Dunkelbild mit Dead- und Stuckpixel	18
5.2	Dunkelbild mit Dead- und Stuckpixel	18
5.3	Unterschied zwischen dem richtigen und dem korrigierten Bild	19
5.4	Dunkelbild mit Dead- und Stuckpixel	19
5.5	Unterschied zwischen dem richtigen und dem korrigierten Bild	21
6.6	Kameraeinstellungen für OpenCV	40

Tabellenverzeichnis

2.1	Messwerte Kalibrierung	4
2.2	Messwerte Kalibrierung	6
5.1	Messwerte Kalibrierung	20
5.2	Messwerte Kalibrierung	20

Listingverzeichnis

6.1	Bild einlesen von der Webcam und Bildeinstellungen	22
6.2	Bild in Grauwerte aufteilen	23
6.3	LaTeX Tabelle mit Mittelwert Hexwert und Standartabweichung	25
6.4	Pixelweisen Mittelwert der 10 Dunkelbilder berechnen und Bild ausgeben .	27
6.5	Bild vom Grauwertkeil vom Dunkelbild abziehen	29
6.6	Pixelweisen Mittelwert der 10 Weißbilder berechnen und Bild ausgeben .	30
6.7	Das berechnete Weißbild minus das Dunkelbild	32
6.8	Bild normieren	33
6.9	korrigiertes Bild erneut aufteilen in die einzelnen Grauwerte	34
6.10	Mittelwert Hexwert und Standartabweichung berechnen	35
6.11	Deadpixels berechnen	36
6.12	Stuckpixels berechnen	38

1

Einleitung

In dem zweiten Versuch der Versuchsreihe in Signale, Systeme und Sensoren geht es um die Kalibrierung von Kameras.

Die Kalibrierung erfolgt mithilfe unseres Python Skriptes und der OpenCV Bibliothek. So wird nach der Kalibrierung und der Justierung ein Grauwertkeil aufgenommen und ausgewertet.

So kann man über ein Bild mehr herausfinden, als auf den ersten Blick zu sehen ist.

Während des Versuches werden wir so das Bild in die einzelnen Grauwerte des Grauwertkeiles aufteilen und über diese die Standardabweichung, den Mittelwert, sowie den Hexwert zu berechnen.

So können wir bei einem Bild außerdem noch das thermische Rauschen entfernen, beziehungsweise die Vignettierung relativieren. Kontrastmaximiert ist es so besonders interessant die Vignettierung zu sehen.

Durch die erhaltenen Daten wie zum Beispiel die Intensität des Lichteinfalls, können wir unter Anderem die Standardabweichung, den Mittelwert und den Hexwert berechnen. Eine weitere Möglichkeit ist durch die Normierung des durchschnittlichen Weißbildes die Intensivität des Lichteinfalls zu berechnen. Außerdem kann man wenn welche vorhanden sind, Stuck-, Hot- und Deadpixel beobachten.

Im großen Ganzen ist es wirklich sehr interessant, wieviele unscheinbare Informationen man aus einem Bild erhalten kann, wenn man sie richtig auswertet. Welche Informationen tatsächlich vorhanden sind und wie interessant diese auch tatsächlich sind, werden wir im Laufe der folgenden 4 Versuche noch feststellen.

2

Versuch 1: Aufnahme und Analyse eines Grauwertkeiles

2.1 Fragestellung, Messprinzip, Aufbau, Messmittel

Im zweiten Versuch verwenden wir eine Webcam um einen Grauwertkeil aufzunehmen. Das Foto wird senkrecht aufgenommen, indem die Webcam an eine Metallhalterung angebracht wird. Nachdem alles richtig positioniert wurde, so dass die Grauwertstufen möglichst parallel zum Bildrand verlaufen, nehmen wir mittels einem Python Skript das wir programmiert haben aus *task1.1* und der OpenCV Bibliothek ein Bild auf. Ebenfalls können wir die Belichtungs- und Sättigungsparameter mit *.get(entsprechende Kennzahl)*. Das Bild soll mittels des Skriptes in das .png Format erstellt werden, da das .jpg Format verlustbehaftet ist. Zudem sollten wir die Einstellungen der Kamera notieren, sowie den Abstand von Grauwertkeil und Kamera.

Außerdem sollen wir ein Python Skript schreiben, dass nun die Unerbilder aus dem Bild des Grauwertkeils ausliest und in externe Dateien speichert. Dabei sollen die Unterbilder möglichst viele Pixel der jeweiligen Stufe umfassen ohne die Stufenränder zu berühren, da diese sonst die Standartabweichung erhöhen und den Durchschnitt des Grauwertes verfälschen. Zudem soll der Mittelwert und die Standartabweichung der einzelnen Unterbilder gemacht werden.

Hier kann man einige von den Gegenstände sehen, welche man im Versuch benötigt:

- Kamerahalterung
- Webcam
- Grauwertkeil
- PyCharm
- Papier

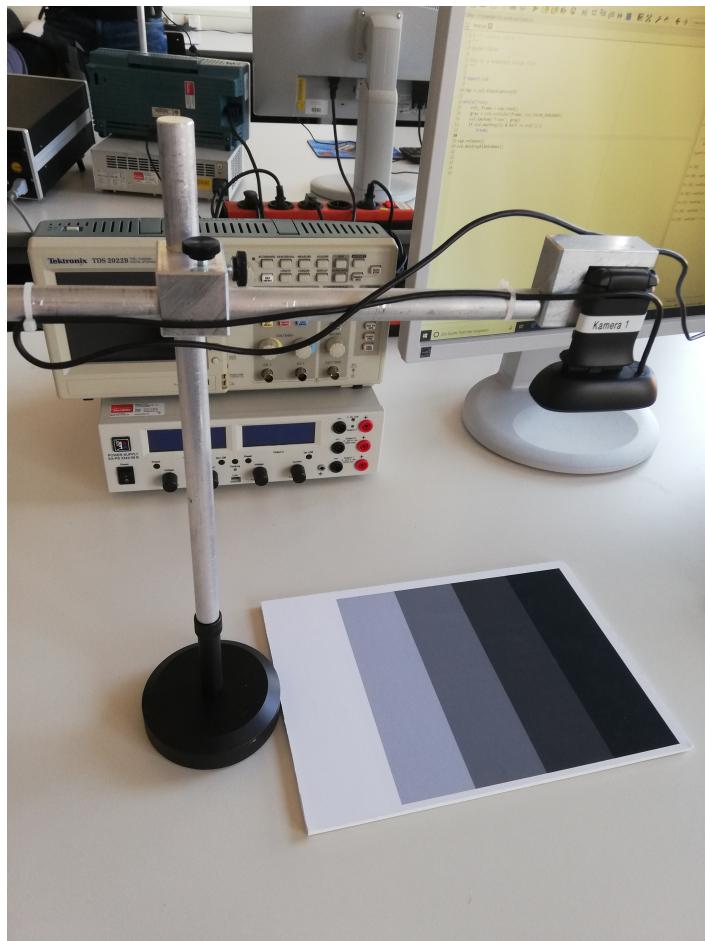


Abbildung 2.1: Aufbau im Labor

2.2 Messwerte

Durch die Kalibrierung im Skript und die Justierung der Geräte und des Grauwertkeils haben wir ein Bild dessen gemacht.

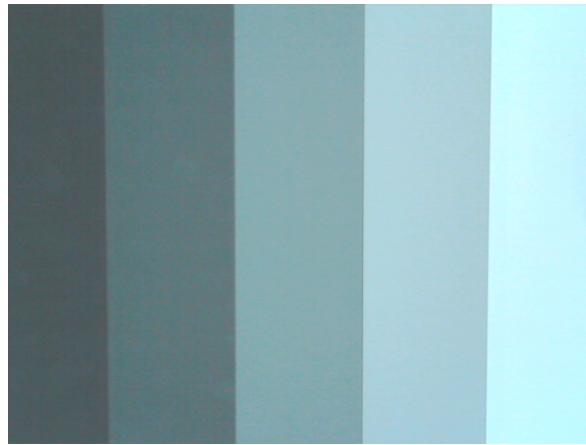


Abbildung 2.2: Das Bild des Grauwertkeiles

Tabelle [2.2] zeigt die Kalibrierungen, die wir vorgenommen haben, dass die Kamera stets die selben Werte hat. Dies wurde mit `.set(3, 640)`. Dabei wird im ersten Teil, hier als 3 definiert die Einstellung gesucht. In diesem Fall die Frame Breite. Im zweiten Teil wird die Frame Breite definiert. Hier als 640 Pixel.

Beschreibung	Wert	Wertebereich
Frame Width	640	Je nach Bedarf
Frame Height	480	Je nach Bedarf
Brightness	133	0 - 255
Contrast	32	0 - 255
Saturation	32	0 - 255
Gain	20	0 - 255
Exposure	-4	-1 - -7
White Balance	10000	0 - 10000

Tabelle 2.1: Messwerte Kalibrierung

2.3 Auswertung

Da das Bild einen starken Blaustich hat, haben wir die Datei im Nachhinein mit `cv2.IMREAD_GRAYSCALE` in Schwarz Weiß gefärbt. In den folgenden Abbildung sind die einzelnen Aufteilungen vom Bild des Grauwertkeils dargestellt. Hierfür wurde zuerst in der Python Datei *task1.2* mit der Pythonbibliothek *cv2* das Bild des Grauwertkeils mit manuellen Grenzwerten in ihre einzelnen Grauwerte aufgeteilt. Die Dateien (a) bis (e) sind die einzelnen Grauwerte. (f) ist das gesamte Bild des Grauwertkeils.

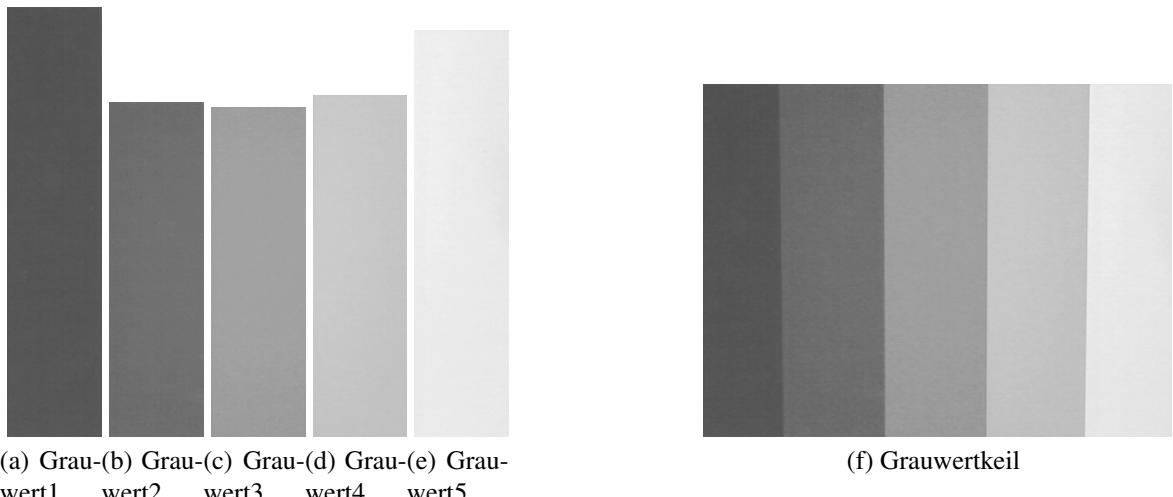


Abbildung 2.3: Die einzelnen Grauwerte und das zusätzliche Gesamtbild

Mittels `cv2.mean()` haben wir nun den durchschnittlichen Grauwert der einzelnen Graustufen berechnet und mit `np.std()` die Standardabweichung. Zudem haben wir den Hexwert mit matplotlib berechnet. Das Ganze haben wir so in die Konsole ausgegeben, dass wir direkt den Konsolenoutput in LaTeX einfügen konnten. Die Tabelle ist bei den Messergebnissen Tabelle 2.2 einsehbar.

Tabelle [4.2] zeigt die in Python berechneten Werte für die einzelnen Graustufen die einzelnen Grauwerte, den Hexwert, sowie die Standardabweichung

Graustufe	Grauwert	Hex-Value	Standardabweichung
Graustufe 1	86.39761904761905	#161616	2.082341260388545
Graustufe 2	112.00498456790123	#1d1d1d	2.6116529874626044
Graustufe 3	161.2048205596107	#292929	2.2861194924984116
Graustufe 4	199.60105744949496	#333333	2.968662423351764
Graustufe 5	235.58106563421828	#3c3c3c	2.582844019720398

Tabelle 2.2: Grauwert, Hexwert und Standardabweichung für die einzelnen Grauwerte

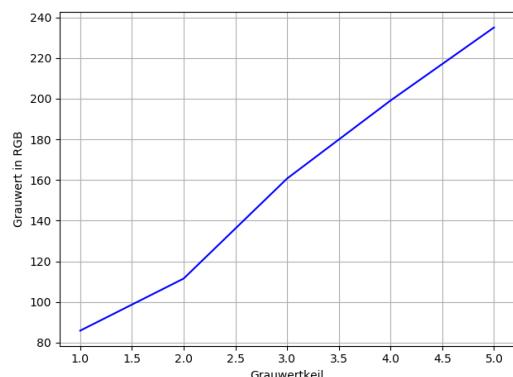
2.4 Interpretation

Durch einen Messfehler ist der Grauwertkeil in farbig statt in Schwarz Weiß aufgenommen worden. Dies haben wir durch die Funktion `cv2.IMREAD_GRAYSCALE` behoben. Die einzelnen Abschnitte der Grauwerte sind ungleich hoch, da die einzelnen Grauwerte nicht alle parallel nach unten verlaufen und sonst noch Pixel von anderen Grauwerten beinhalten würden.

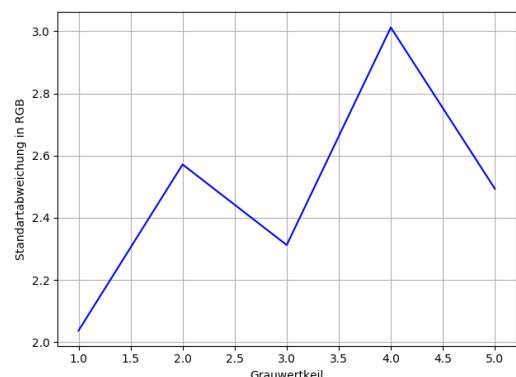
Die Werte auf der y-Achse sind die Grauwerte in RGB, während die x-Achse den Grauwertkeil von 1 bis 5 darstellt. Der Grauwert könnte von 0 für schwarz bis 255 für weiß gehen.

Die Ausschwenkung des zweiten Grauwertes könnte daran liegen, dass dunkle Farben durch Sensorrauschen und seitlichen Lichteinfluss dunkler wirken, als sie tatsächlich sind. Eine andere Theorie ist, dass die Grauwerte auf dem Keil nicht linear dunkel sind, sondern der Grauwert einfach dunkler ist als im Schnitt. So hat der zweite Wert die zweitgröte Standardabweichung sämtlicher Grauwerte.

Im Gesamten kann man aber sagen, dass die Farben eine relativ geringe Standardabweichung haben die in RGB-Werten zwischen 2,08 und 2,96 liegt.



(a) Durchschnittlicher Grauwert



(b) Standardabweichung des Grauwerts

Abbildung 2.4: Standardabweichung und Durchschnitt der jeweiligen Grauwerte

3

Versuch 2: Aufnahme eines Dunkelbildes

3.1 Fragestellung, Messprinzip, Aufbau, Messmittel

Im Versuch 2 sollten wir die Kamera so abdecken, dass das Bild komplett schwarz erscheint. Dies haben wir mit einem schwarzen Mousepad gemacht. Mit den selben Kameraeinstellungen wie bei der Aufnahme des Grauwertkeils sollten wir nun 10 Bilder machen. Dies haben wir in Python mit einer simplen for-Schleife verwirklicht. Danach sollten wir wieder in Python die 10 Bilder einlesen, in *double* umwandeln und den pixelweisen Mittelwert berechnen.

Da die Webcam standartmäßig nur Farbbilder liefert, müssen wir noch die Bilder in Grauwertbilder umwandeln. Da man nun 307200 Mittelwerte für den Durchschnitt der 10 Bilder erhält, speichern wir diese in einem Vektor um die for-Schleife effizienter zu machen. Aus den sämtlichen Mittelwerten wiederrum, haben wir uns ein Bild erstellen lassen und dieses kontrastmaximiert dargestellt. Danach haben wir von jedem Pixel im Grauwertkeil den entsprechenden Mittelwertspixel abgezogen. Dieses korrigierte Bild wurde daraufhin gespeichert.

3.2 Messwerte

Hier ist eines der 10 aufgenommenen Dunkelbilder, die durch das Pythonskript gecaptured wurden.



Abbildung 3.1: Eines der 10 aufgenommenen Dunkelbilder

3.3 Auswertung

Um die Webcam abzudecken, haben wir das im Labor für die PC's verwendete Mousepad genommen. Mit der Funktion `cv2.equalizeHist(image)` erhalten wir ein kontrastmaximierte Bild. Dieses ist in der Abbildung 3.2 einsehbar. Das Bild hat einen konstanten Grauwert von 71.



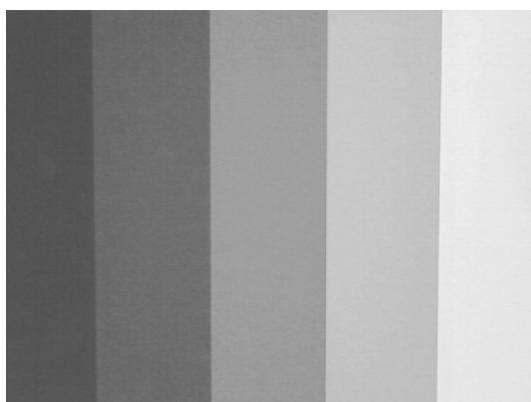
(a) Aufnahme des Grauwertkeils



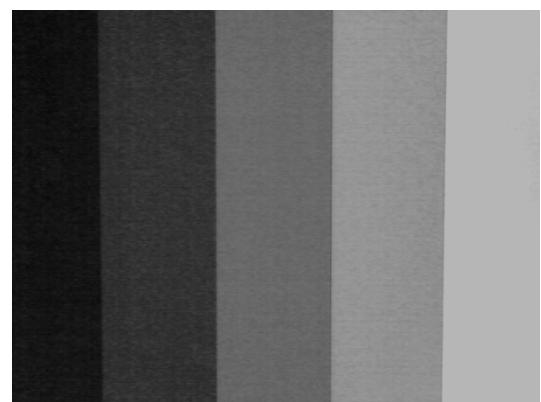
(b) kontrastmaximierte Bild

Abbildung 3.2: Grauwertkeil und kontrastmaximierte Bild

Nachdem wir das Bild des pixelweisen Durchschnittes und das Bild des Grauwertkeils eingelesen haben und in ein Schwarz-Weiß Bild umgewandelt haben, subtrahieren wir vom Bild des Grauwertkeiles das Bild des pixelweisen Durchschnittes. Dabei erhalten wir die Abbildung in 3.2.b.



(a) Aufnahme des Grauwertkeils



(b) Korrigiertes Bild

Abbildung 3.3: Unterschied zwischen dem richtigen und dem korrigierten Bild

3.4 Interpretation

Da das kontrastmaximierte Bild keine Veränderungen aufweist im Vergleich zum normalen Durchschnittsbild wissen wir, dass es keine Kontraste gibt und das Bild überall dieselbe Farbe besitzt. Das Bild hat nicht den Grauwert 0 sondern 71, da wir die Belichtungseinstellungen von dem Bild des Grauwertkeiles verwendet haben.

Aufgrunddessen haben wir die Gewissheit, dass wir die Kamera vollständig abgedeckt haben, ohne dass es einen externen Lichteinfluss gab. Schlichtweg würde das kontrastmaximierte Bild darstellen, dass es keine Störfaktoren in der Webcam gibt und alle denselben Nullpunkt haben.

Durch die Subtraktion des durchschnittlichen Dunkelbildes, erhalten wir den Nullpunkt jedes einzelnen Pixels. Durch diesen Vorgang könnten wir theoretisch das thermische Rauschen entfernen. Da unser durchschnittliches Dunkelbild keine Unterschiede aufweist, wissen wir jedoch, dass es kein thermisches Rauschen gibt.

4

Versuch 3: Aufnahme eines Weißbildes

4.1 Fragestellung, Messprinzip, Aufbau, Messmittel

Im dritten Versuch nehmen wir ein weißes Blatt Papier und legen dieses auf den Grauwertkeil. Dadurch haben wir den selben Abstand wie zum Grauwertkeil. Dabei wird die Belichtung auf 30-50% der Hellsättigung eingestellt.

Hierbei sollten keine Schatten oder andere Störfaktoren auftreten. Auch hier nehmen wir mit unserem Programm 10 Bilder auf.

Mittels einem Python Skript berechnen wir daraufhin den Mittelwert jedes einzelnen Pixels. Durch den Mittelwert eliminieren wir das thermische Rauschen. Von den Mittelwerten des weißen Bildes werden wir das Dunkelbild abziehen und erhalten daraus das resultierende Weißbild. Das Bild geben wir zudem kontrastmaximiert dar.

Das berechnete Weißbild wird nun normiert mit dem Mittelwert 1.

Der Grauwertkeil wird mit dem Dunkelbild subtrahiert und danach mit dem normierten Weißbild dividiert.

Durch das kontrastmaximierte Bild können wir nun so die Intensität des einfallenden Lichts betrachten.

4.2 Messwerte

Hier ist eines der aufgenommenen Bilder vom weißen Blatt Papier. Leider ist durch einen Aufnahmefehler das Bild blau geworden. Dieses Problem haben wir durch das einlesen im Python Programm durch folgenden Code behoben:

```
1 grey = cv2.imread('data/Versuch1a.png', cv2.IMREAD_GRAYSCALE)
```

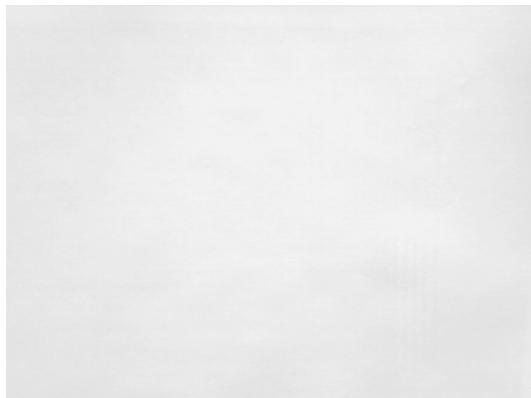
Durch *cv2.IMREAD_GRAYSCALE* wird das Bild in ein Schwarz-Weiß Bild umgewandelt.



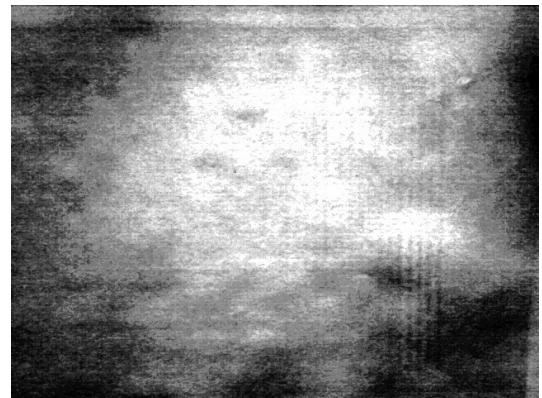
Abbildung 4.1: Eines der 10 aufgenommenen Weißbilder

4.3 Auswertung

Nun haben wir in zwei for-Schleifen für jeden Pixel den Durchschnitt aller 10 Bilder ausgewertet und in ein neues Bild geschrieben. Das untenstehende linke Bild zeigt den Pixelweisen Durchschnitt. Das rechte der beiden Bilder ist das linke Bild kontrastmaximiert.



(a) Pixelweiser Mittelwert der einzelnen Bilder



(b) kontrastmaximierte Bild der Weißbilder

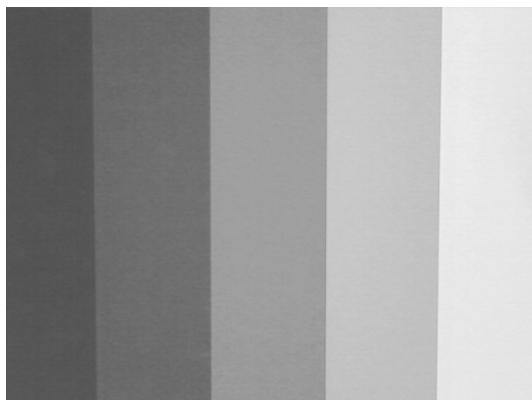
Abbildung 4.2: Mittelwert der Weißbilder und kontrastmaximierte Bild

Um die tatsächliche Intensität des Lichtes zu bestimmen und die Sensitivitäten der einzelnen Pixel, mussten wir zuerst das das Weißbild mit dem Dunkelbild subtrahieren.

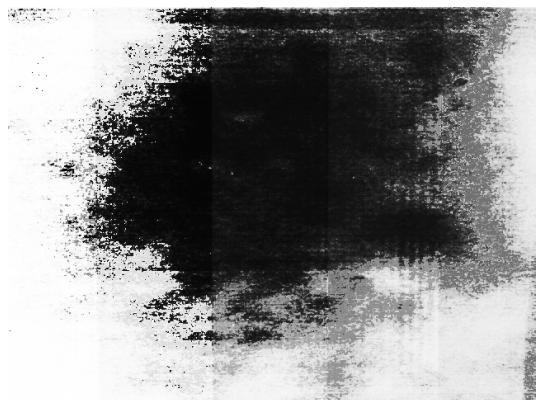


Abbildung 4.3: Dunkelbild subtrahiert vom Weißbild

Danach war es die Aufgabe das Weißbild zu normieren, dass der Mittelwert 1 ist. Das durch Abzug des Dunkelbildes korrigierte Eingangsbild wird anschliessend durch das normierte Weißbild dividiert.



(a) Grauwertkeil



(b) Bild mit der Intensität des Lichteinfalles

Abbildung 4.4: Unterschied zwischen dem Grauwertkeil und dem Bild mit der Intensität des Lichteinfalles

4.4 Interpretation

Das Bild hat ebenfalls wie die Aufnahme des Grauwertkeils einen starken Blautstich. Dies haben wir gelöst durch eine Konvertierung ins Blaue.

Bei der Auswertung haben wir das kontrastmaximierte Bild neben dem des durchschnittlichen Weißbildes, sieht man schön die Verstärkung. Hier sieht man eine Vignettierung des Bildes d.h. die Optik der Kamera übträgt die Helligkeit nicht gleichmäßig auf den Sensor.

Dies wird sichtbar durch eine Abdunkelung des Bildes zu den Rändern hin. Nachdem man Weißbild minus Dunkelbild berechnet hat, bekommt man ein dunkleres Weißbild, bei dem man immer noch leicht die Vignettierung beobachten kann.

In dem letzten Experiment wurde die Intensität des einfallenden Lichtes berechnet. In schwarz kann man nun gut betrachten, wie das Licht eingefallen ist.

Die Kamera war auf ihrer Halterung so zum Fenster gerichtet, dass das Licht von oben eingefallen ist.

Auch hier ist wieder eine Art Vignettierung zu sehen in weiß. Der Lichteinfall stimmt auch mit dem kontrastmaximierten Bild von vorher überein, da am oberen Teil des Bildes die Vignettierung nicht so stark ist bzw. das Licht von oben einfällt.

5

Versuch 4: Aufnahme eines Weißbildes

5.1 Fragestellung, Messprinzip, Aufbau, Messmittel

Im vierten Versuch war es die Aufgabe funktionsuntüchtige Pixel, wie zum Beispiel Hot, Stuck und Deadpixel zu entdecken und zu markieren. Deadpixels sind Pixel die immer auf ihrem niedrigsten Wert steckenbleiben.

Stuck Pixels bleiben immer auf ihrem höchsten Wert, während Hot Pixels in die Sättigung gehen bei längeren Belichtungszeiten. Zudem sollen wir mithilfe des Programmes aus Aufgabe 3 das Bild des Grauwertkeiles korrigieren und es speichern.

Dies haben wir jedoch bereits schon in Aufgabe 3 gelöst. Danach sollten wir mit dem korrigierten Bild so verfahren wie in Aufgabe 1, indem wir das Bild mit den selben Einstellungen wie in Aufgabe 1 das Bild in 5 Unterbilder aufteilen und den durchschnittlichen Grauwert, den Hexwert und die Standartabweichung berechnen.

Diesen werden wir im späteren Verlauf noch mit den Messungen aus Aufgabe 1 vergleichen.

5.2 Messwerte

In der unteren Abbildung untersuchen wir ein Weißbild auf funktionsuntüchtige Pixel.

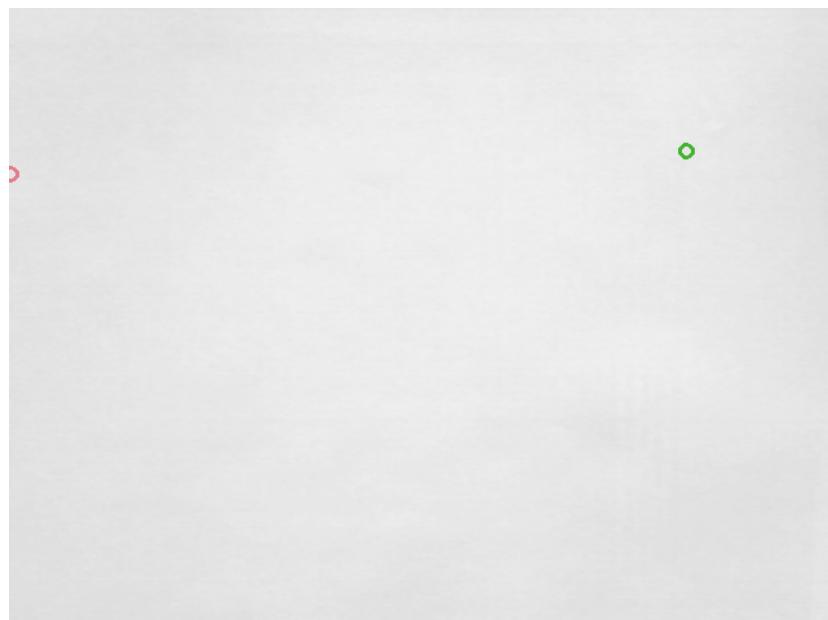


Abbildung 5.1: Weißbild mit Dead- und Stuckpixel

In der unteren Abbildung untersuchen wir ein Dunkelbild auf funktionsuntüchtige Pixel.



Abbildung 5.2: Dunkelbild mit Dead- und Stuckpixel

5.3 Auswertung

Mittels der OpenCV Bibliothek und der damit verbundenen Funktion `cv2.circle` haben wir einen Kreis gezeichnet. Mittels einer For-Schleife werden alle 307200 Pixel untersucht auf den dunkelsten und den hellsten Punkt. Das Dunkelbild wurde ebenfalls mit der Funktion

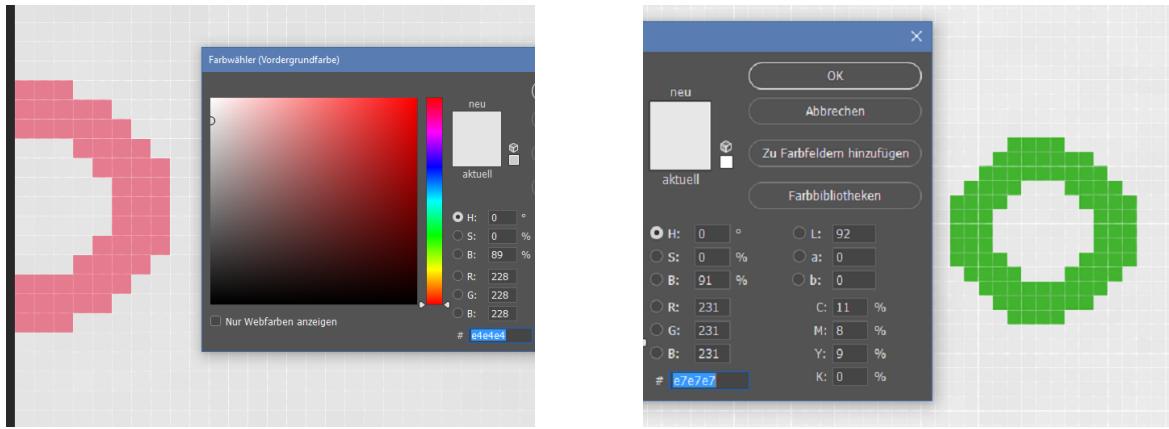


Abbildung 5.3: Unterschied zwischen dem hellsten und dem dunkelsten Bild

`cv2.circle` bearbeitet. Das Ergebnis sehen sie in der Abbildung 5.4.

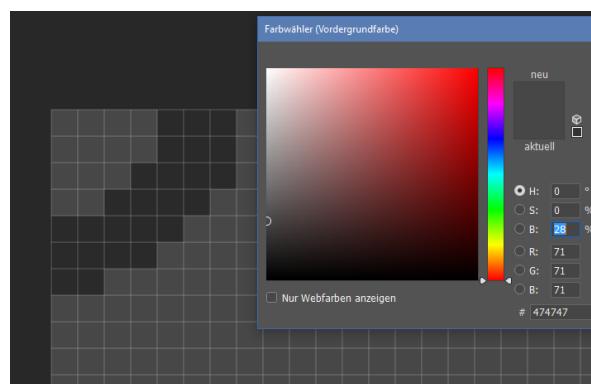


Abbildung 5.4: Dunkelbild mit Dead- und Stuckpixel

Der mit dem normierten Weißbild korrigierte Graukeil, wird nochmal mit dem Skript aus der ersten Aufgabe berechnet. In dieser Tabelle ist der Mittelwert, der HexValue und die Standardabweichung für die Intensität des Lichteinfalls.

Stufe	Mittelwert	Hex-Value	Standartabweichung
Stufe 1	241.6711111111112	#3e3e3e	36.794374014010515
Stufe 2	123.12765432098767	#1f1f1f	101.62025094391961
Stufe 3	60.187013381995136	#0f0f0f	73.25401533025126
Stufe 4	93.80972222222222	#181818	83.6389175516743
Stufe 5	160.29564564564564	#292929	78.10787025296499

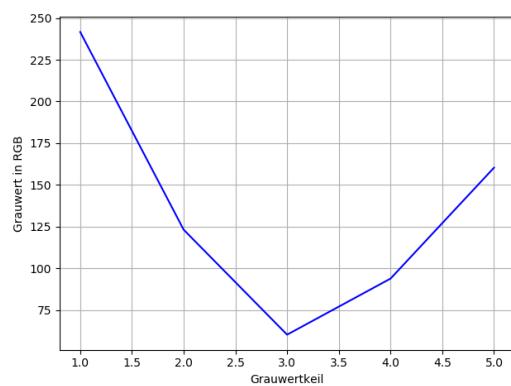
Tabelle 5.1: Mittelwert, Hexwert und Standardabweichung für die Intensivität des Lichteinfalls

5.4 Interpretation

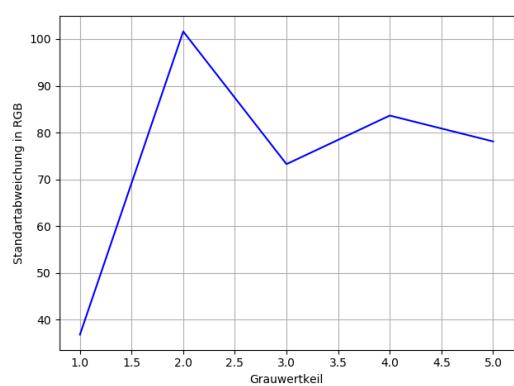
Um die beiden Extrempunkte wurde ein roter Kreis um den dunkelsten Punkt gezeichnet und ein grüner Kreis um den hellsten Punkt. Da weder ein sehr dunkler Punkt noch ein extrem weißer außergewöhnlicher Punkt vorkommt im Bild, gibt es im Weißbild weder Stuck-, noch Deadpixels. Auch das Dunkelbild weißt keine Dead- oder Stuckpixels auf. Das gesamte Bild hat einen durchgehenden Grauwert von 71. So ist sowohl der höchste als auch der kleinste Farbwert bei 71.

Das korrigierte Bild mit dem Programm aus Aufgabe 3 wurde bereits in dem dritten Versuch gelöst und ausgewertet.

Da das Licht von oben in den Sensor der Kamera eingetreten ist, ist wie zu erwarten eine 'U' Form zu sehen. Der Wert im dritten Graukeil ist am niedrigsten in der Abbildung 5.5.a, da dieser auch am meisten Schwarz Anteil hat. Links und rechts davon steigt der Wert wieder an, durch die Vignettierung und den unterschiedlichen Lichteinfall. Die Standartabweichung ist wie zu erwarten beim zweiten und vierten Wert am höchsten, da diese wie in der Abbildung 4.4.b einen hohen Lichteinfall haben, aber trotzdem durch die Vignettierung eine nicht gleichmäßige Verteilung auf den Kamerasensor geschieht. Der erste Wert hat die geringste Standartabweichung da dieser am meisten von dem Lichteinfall entfernt ist.



(a) Durchschnitt der Intensität des Lichteinfalls



(b) Standartabweichung der Intensität des Lichteinfalls

Abbildung 5.5: Unterschied zwischen dem [Durchschnitt und der Standartabweichung der Intensität des Lichteinfalls]

Anhang

A.1 Quellcode

A.1.1 Quellcode Versuch 1

```
1 import cv2
2
3 # ----- Aufgabe1.1 ----- #
4
5 cap = cv2.VideoCapture(0)
6
7 cap.set(3, 640)
8 cap.set(4, 480)
9 cap.set(10, 200)
10 cap.set(11, 32)
11 cap.set(12, 32)
12 cap.set(14, 20)
13 cap.set(15, -4)
14 cap.set(17, 10000)
15
16 while (True):
17     ret, frame = cap.read()
18     gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
19     cv2.imshow("frame", gray)
20     for x in range(1, 11):
21         cv2.imwrite("data/bild" + str(x) + ".png", frame)
22     if cv2.waitKey(1) & 0xFF == ord("q"):
23         break;
24
25 # cv2.imwrite("data/bild.png", frame)
26
27 print("frame width: " + str(cap.get(3)))
28 print("frame height: " + str(cap.get(4)))
```

```

29 print("-----")
30 print("brightness: " + str(cap.get(10)))
31 print("contrast: " + str(cap.get(11)))
32 print("saturation: " + str(cap.get(12)))
33 print("-----")
34 print("gain: " + str(cap.get(14)))
35 print("exposure: " + str(cap.get(15)))
36 print("-----")
37 print("white balance: " + str(cap.get(17)))
38
39 cap.release()
40 cv2.destroyAllWindows()

```

Listing 6.1: Bild einlesen von der Webcam und Bildeinstellungen

```

1 import numpy as np
2 import cv2
3
4 # ----- Aufgabe1.2 -----
5
6 # Vector um Grenz- und Breitenwert zu speichern
7 vec = np.zeros((5, 2))
8 crop = ["crop1", "crop2", "crop3", "crop4", "crop5"]
9
10 # Grauwertkeil einlesen – Bild ist warum auch immer blau , deshalb in Grau umwandeln
11 image = cv2.imread('data/Versuch1a.png', cv2.IMREAD_GRAYSCALE)
12
13 # Grenz- und Breitenwerte für Bild 1
14 vec[0, 0] = 0
15 vec[0, 1] = 105
16 # Grenz- und Breitenwerte für Bild 2
17 vec[1, 0] = 111
18 vec[1, 1] = 135
19 # Grenz- und Breitenwerte für Bild 3
20 vec[2, 0] = 249
21 vec[2, 1] = 137
22 # Grenz- und Breitenwerte für Bild 4
23 vec[3, 0] = 389
24 vec[3, 1] = 132
25 # Grenz- und Breitenwerte für Bild 5
26 vec[4, 0] = 529
27 vec[4, 1] = 111
28

```

```

29 # for-Schleife um auf alle Bilder von 1-5 zuzugreifen
30 for z in range(1, 6):
31
32     # Ab welchem Pixel in der Höhe y soll begonnen werden
33     y = 0
34     # Ab welchem Pixel in der Breite x soll begonnen werden
35     x = int(vec[z-1, 0])
36     # Wie tief soll der Pixel gehen in h
37     h = 480
38     # Wie breit soll der Pixel gehen in w
39     w = int(vec[z-1, 1])
40
41     # Schneiden des Bildes mit den Variablen von oben
42     crop[z-1] = image[y:y + h, x:x + w]
43     # geschnittene Bilder anzeigen
44     cv2.imshow("Crop" + str(z), crop[z-1])
45     # geschnittene Bilder exportieren
46     cv2.imwrite("bild" + str(z) + ".png", crop[z-1])
47
48     # Auf den Knopfdruck 0 warten bevor
49     cv2.waitKey(0)

```

Listing 6.2: Bild in Grauwerte aufteilen

```

1 import numpy as np
2 import cv2
3 import matplotlib
4 import matplotlib.pyplot as plt
5
6 # ----- Aufgabe1.3 -----
7
8 vec = np.zeros((10, 3))
9
10 # LaTeX Tabellenformat
11 print("\hline")
12 print("Stufe & Mittelwert & Hex-Value & Standartabweichung \\\\")

13
14 # for-Schleife um jede der 5 Dateien zu erreichen
15 for x in range(1, 6):
16     # einlesen der aufgeteilten Bilder
17     image = cv2.imread("data/bild" + str(x) + ".png")
18
19     # Durchschnitt des aufgeteilten Bildes bekommen
20     b, g, r, a = cv2.mean(image)
21     # bgr durch 1000 da die hexfunktion Werte zwischen 0 und 1 braucht
22     b1 = b / 1000
23     g1 = g / 1000
24     r1 = r / 1000
25     # Hex Wert berechnen
26     hex = matplotlib.colors.to_hex([b1, g1, r1])
27
28     # LaTeX Tabellenformat für jede Stufe
29     print("\hline")
30     print("Stufe " + str(x) + " & " + str((b + g + r) / 3) + " & " + hex + " & " + str(np.std(image)) + " \\\\")

31
32     vec[x-1, 0] = x
33     vec[x-1, 1] = (b + g + r) / 3
34     vec[x-1, 2] = np.std(image)
35
36 # Darstellung des Grauwerts in einem Graphen
37 plt.plot([vec[0, 0], vec[1, 0], vec[2, 0], vec[3, 0], vec[4, 0]],
38          [vec[0, 1], vec[1, 1], vec[2, 1], vec[3, 1], vec[4, 1]], 'b')
39 plt.ylabel('Grauwert in RGB')
40 plt.xlabel('Grauwertkeil')
41 plt.grid(True)

```

```

42 plt.show()
43
44 # Darstellung der Standartabweichung in einem Graphen
45
46 plt.plot([vec[0, 0], vec[1, 0], vec[2, 0], vec[3, 0], vec[4, 0]],
47           [vec[0, 2], vec[1, 2], vec[2, 2], vec[3, 2], vec[4, 2]], 'b')
48 plt.ylabel('Standartabweichung in RGB')
49 plt.xlabel('Grauwertkeil')
50 plt.grid(True)
51 plt.show()
52
53 print("\hline")

```

Listing 6.3: LaTeX Tabelle mit Mittelwert Hexwert und Standartabweichung

A.1.2 Quellcode Versuch 2

```
1 import numpy as np
2 import cv2
3
4 # ----- Aufgabe2.1 ----- #
5
6 # Vectoren für die 10 Dunkelbildern
7 vec1 = np.zeros((480, 640))
8 vec2 = np.zeros((480, 640))
9 vec3 = np.zeros((480, 640))
10 vec4 = np.zeros((480, 640))
11 vec5 = np.zeros((480, 640))
12 vec6 = np.zeros((480, 640))
13 vec7 = np.zeros((480, 640))
14 vec8 = np.zeros((480, 640))
15 vec9 = np.zeros((480, 640))
16 vec10 = np.zeros((480, 640))
17 # Vector für das Durchschnittsbild
18 average = np.zeros((480, 640))
19
20 # String für die einzelnen Dateinamen
21 blackpic = ["black1", "black2", "black3", "black4", "black5", "black6",
22             "black7", "black8", "black9", "black10"]
23 # Vektornamen in einem Vektor
24 vector = [vec1, vec2, vec3, vec4, vec5, vec6, vec7, vec8, vec9, vec10]
25
26 # for-Schleife um auf alle 10 Dateien zuzugreifen
27 for x in range(0, 10):
28     # Dunkelbilder einlesen
29     blackpic[x] = cv2.imread('data/' + blackpic[x] + '.png', cv2.IMREAD_GRAYSCALE)
30
31 # for-Schleife um auf alle 307200 Pixel zuzugreifen
32 for y in range(0, 480):
33     for z in range(0, 640):
34         # Vektoren der Dateien füllen mit den Bildpixelvalues
35         vector[x][y, z] = blackpic[x][y, z]
36
37 # for-Schleife um auf alle 307200 Pixel zuzugreifen
38 for y in range(0, 480):
39     for z in range(0, 640):
40         # Mean auf 0 zurück zu setzen
```

```

41 mean = 0
42 #for-Schleife um auf alle 10 Dateien zuzugreifen
43 for file in range(0, 10):
44     # Pixel aus allen Dateien welche auf der selben Stelle liegen addieren
45     mean += vector[file][y, z]
46 # Pixel nun durch die Anzahl der Dateien teilen um den Mittelwert des Pixels zu erlangen
47 average[y, z] = float(mean / 10)
48
49 # Datei einlesen um sie zu überschreiben
50 image = cv2.imread("data/blackaverage.png", 0)
51 #for-Schleife um auf alle 307200 Pixel zuzugreifen
52 for x in range(0, 480):
53     for y in range(0, 640):
54         # Bild erstellen mit den Mittelwert der Pixels
55         image[x, y] = average[x, y]
56
57 # Bild exportieren
58 cv2.imwrite("data/blackaverage.png", image)
59 # Bild anzeigen
60 cv2.imshow('image', image)
61
62 # Bild kontrastmaximiert darstellen
63 image_contrast = cv2.equalizeHist(image)
64
65 # kontrastmaximierte Bild exportieren und anzeigen
66 cv2.imwrite("data/contrastblackaverage.png", image_contrast)
67 cv2.imshow('image_contrast', image_contrast)
68 cv2.waitKey(0)
69 cv2.destroyAllWindows()

```

Listing 6.4: Pixelweisen Mittelwert der 10 Dunkelbilder berechnen und Bild ausgeben

```

1 import numpy as np
2 import cv2
3
4 # ----- Aufgabe2.2 -----
5
6 # Vector für das Korrekturbild
7 korrigiertes_bild = np.zeros((480, 640))
8
9 # Bild des Grauwertkeils einlesen und in Schwarz–Weiß umwandeln
10 grey = cv2.imread('data/Versuch1a.png', cv2.IMREAD_GRAYSCALE)
11 # Bild des Pixelweisen Mittelwerts einlesen
12 blackavg = cv2.imread('data/blackaverage.png')
13
14 # for-Schleife um auf alle 307200 Pixel zuzugreifen
15 for y in range(0, 480):
16     for z in range(0, 640):
17         # Auf Grauwert des Pixels zugreifen des Grauwertkeils
18         r1, g1, b1 = grey[y, z]
19         # Auf Grauwert des Pixels zugreifen des Dunkelbildes
20         r2, g2, b2 = blackavg[y, z]
21         # Grauwertkeil – Dunkelbild
22         korrigiertes_bild[y, z] = int(r1) - int(r2)
23
24 # Datei einlesen um sie zu überschreiben
25 image = cv2.imread("data/korrigiertes_bild.png")
26 # for-Schleife um auf alle 307200 Pixel zuzugreifen
27 for x in range(0, 480):
28     for y in range(0, 640):
29         # Bild erstellen mit den Mittelwert der Pixels
30         image[x, y] = korrigiertes_bild[x, y]
31
32 # Bild exportieren und anzeigen
33 cv2.imwrite("data/korrigiertes_bild.png", image)
34 cv2.imshow('image', image)
35 cv2.waitKey(0)
36 cv2.destroyAllWindows()

```

Listing 6.5: Bild vom Grauwertkeil vom Dunkelbild abziehen

A.1.3 Quellcode Versuch 3

```
1 import numpy as np
2 import cv2
3
4 # ----- Aufgabe3.1 ----- #
5
6 # Vectoren für die 10 Weißbildern
7 vec1 = np.zeros((480, 640))
8 vec2 = np.zeros((480, 640))
9 vec3 = np.zeros((480, 640))
10 vec4 = np.zeros((480, 640))
11 vec5 = np.zeros((480, 640))
12 vec6 = np.zeros((480, 640))
13 vec7 = np.zeros((480, 640))
14 vec8 = np.zeros((480, 640))
15 vec9 = np.zeros((480, 640))
16 vec10 = np.zeros((480, 640))
17 # Vector für das Durchschnittsbild
18 average = np.zeros((480, 640))
19
20 # String für die einzelnen Dateinamen
21 whitepic = ["white1", "white2", "white3", "white4", "white5", "white6",
22             "white7", "white8", "white9", "white10"]
23 # Vektornamen in einem Vektor
24 vector = [vec1, vec2, vec3, vec4, vec5, vec6, vec7, vec8, vec9, vec10]
25
26 # for-Schleife um auf alle 10 Dateien zuzugreifen
27 for x in range(0, 10):
28     # Weißbilder einlesen
29     whitepic[x] = cv2.imread('data/' + whitepic[x] + '.png', cv2.IMREAD_GRAYSCALE)
30
31 # for-Schleife um auf alle 307200 Pixel zuzugreifen
32 for y in range(0, 480):
33     for z in range(0, 640):
34         # Vektoren der Dateien füllen mit den Bildpixelvalues
35         vector[x][y, z] = whitepic[x][y, z]
36
37 # for-Schleife um auf alle 307200 Pixel zuzugreifen
38 for y in range(0, 480):
39     for z in range(0, 640):
40         # Mean auf 0 zurück zu setzen
```

```

41 mean = 0
42 #for-Schleife um auf alle 10 Dateien zuzugreifen
43 for file in range(0, 10):
44     # Pixel aus allen Dateien welche auf der selben Stelle liegen addieren
45     mean += vector[file][y, z]
46 # Pixel nun durch die Anzahl der Dateien teilen um den Mittelwert des Pixels zu erlangen
47 average[y, z] = float(mean / 10)
48
49 # Datei einlesen um sie zu überschreiben
50 image = cv2.imread("data/whiteaverage.png", 0)
51 #for-Schleife um auf alle 307200 Pixel zuzugreifen
52 for x in range(0, 480):
53     for y in range(0, 640):
54         # Bild erstellen mit den Mittelwert der Pixels
55         image[x, y] = average[x, y]
56
57 # Bild exportieren
58 cv2.imwrite("data/whiteaverage.png", image)
59 # Bild anzeigen
60 cv2.imshow('image', image)
61
62 # Bild kontrastmaximiert darstellen
63 image_contrast = cv2.equalizeHist(image)
64
65 # kontrastmaximierte Bild exportieren und anzeigen
66 cv2.imwrite("data/contrastwhiteaverage1.png", image_contrast)
67 cv2.imshow('image_contrast', image_contrast)
68 cv2.waitKey(0)
69 cv2.destroyAllWindows()

```

Listing 6.6: Pixelweisen Mittelwert der 10 Weißbilder berechnen und Bild ausgeben

```

1 import numpy as np
2 import cv2
3
4 # ----- Aufgabe3.2 -----
5
6 # Vektor um die Werte der Subtraktion zu speichern
7 white_minus_black = np.zeros((480, 640))
8
9 # Weiß und Schwarz einlesen
10 whiteavg = cv2.imread('data/whiteaverage.png')
11 blackavg = cv2.imread('data/blackaverage.png')
12 image = cv2.imread("data/whiteminusblack.png")
13
14 # for-Schleife um auf alle 307200 Pixel zuzugreifen
15 for y in range(0, 480):
16     for z in range(0, 640):
17         # Grau-Werte berechnen
18         r1, g1, b1 = whiteavg[y, z]
19         r2, g2, b2 = blackavg[y, z]
20         # Weiß – Schwarz
21         white_minus_black[y, z] = int(r1) - int(r2)
22
23 # for-Schleife um auf alle 307200 Pixel zuzugreifen
24 for x in range(0, 480):
25     for y in range(0, 640):
26         # neue Pixel überschreiben von weiß – schwarz einfügen
27         image[x, y] = white_minus_black[x, y]
28
29 # Bild exportieren und anzeigen
30 cv2.imwrite("data/whiteminusblack.png", image)
31 cv2.imshow('image', image)
32
33 # Bild kontrastmaximierte darstellen
34 image_contrast = cv2.equalizeHist(image)
35
36 # Bild exportieren und anzeigen
37 cv2.imwrite("data/contrastwhiteaverage2.png", image_contrast)
38 cv2.imshow('image_contrast', image_contrast)
39 cv2.waitKey(0)
40 cv2.destroyAllWindows()

```

Listing 6.7: Das berechnete Weißbild minus das Dunkelbild

```

1 import numpy as np
2 import cv2
3
4 # ----- Aufgabe3.3 -----
5
6 # Vektor zum Speichern des korrigierten Bildes
7 korrigiertes_bild = np.zeros((480, 640))
8
9 # Bild einlesen und evtl in Schwarz Weiß einlesen
10 grey = cv2.imread('data/Versuch1a.png', cv2.IMREAD_GRAYSCALE)
11 blackavg = cv2.imread('data/blackaverage.png')
12 whiteaverage = cv2.imread('data/whiteaverage.png')
13 whiteminusblack = cv2.imread('data/whiteminusblack.png')
14
15 # for-Schleife um auf alle 307200 Pixel zuzugreifen
16 for y in range(0, 480):
17     for z in range(0, 640):
18         # (Weiß-Schwarz) - avg(Weiß-Schwarz)
19         r, g, b = whiteaverage[y, z] - np.mean(whiteaverage)
20         # Grau-Werte berechnen
21         r1 = grey[y, z]
22         r2, g2, b2 = blackavg[y, z]
23         # Weiß - Schwarz
24         korrigiertes_bild[y, z] = int(r1) - int(r2)
25         # /((Weiß-Schwarz) - avg(Weiß-Schwarz))
26         korrigiertes_bild[y, z] /= r
27
28 # Bild einlesen als Grundlage
29 image = cv2.imread("data/korrigiertes_bild.png")
30 # for-Schleife um auf alle 307200 Pixel zuzugreifen
31 for x in range(0, 480):
32     for y in range(0, 640):
33         # neue Pixel überschreiben von weiß - schwarz einfügen
34         image[x, y] = korrigiertes_bild[x, y]
35
36 # Bild exportieren und anzeigen
37 cv2.imwrite("data/korrigiertes_bild2.png", image)
38 cv2.imshow('image', image)
39 cv2.waitKey(0)
40 cv2.destroyAllWindows()

```

Listing 6.8: Bild normieren

```

1 import numpy as np
2 import cv2
3
4 # ----- Aufgabe3.4 -----
5
6 # Vector um Grenz- und Breitenwert zu speichern
7 vec = np.zeros((5, 2))
8 crop = ["crop1", "crop2", "crop3", "crop4", "crop5"]
9
10 # Bild einlesen und in Schwarz – Weiß umwandeln
11 image = cv2.imread('data/korrigiertes_bild2.png', cv2.IMREAD_GRAYSCALE)
12
13 # Grenz- und Breitenwerte für Bild 1
14 vec[0, 0] = 0
15 vec[0, 1] = 105
16 # Grenz- und Breitenwerte für Bild 2
17 vec[1, 0] = 111
18 vec[1, 1] = 135
19 # Grenz- und Breitenwerte für Bild 3
20 vec[2, 0] = 249
21 vec[2, 1] = 137
22 # Grenz- und Breitenwerte für Bild 4
23 vec[3, 0] = 389
24 vec[3, 1] = 132
25 # Grenz- und Breitenwerte für Bild 5
26 vec[4, 0] = 529
27 vec[4, 1] = 111
28
29 # Auf alle 5 Dateien zugreifen
30 for z in range(1, 6):
31
32     # Ab welchem Pixel in der Höhe y soll begonnen werden
33     y = 0
34     # Ab welchem Pixel in der Breite x soll begonnen werden
35     x = int(vec[z-1, 0])
36     # Wie tief soll der Pixel gehen in h
37     h = 480
38     # Wie breit soll der Pixel gehen in w
39     w = int(vec[z-1, 1])
40
41     # Schneiden des Bildes mit den Variablen von oben

```

```

42 crop[z - 1] = image[y:y + h, x:x + w]
43 # geschnittene Bilder anzeigen
44 cv2.imshow("Crop" + str(z), crop[z - 1])
45 # geschnittene Bilder exportieren
46 cv2.imwrite("korrigiert" + str(z) + ".png", crop[z-1])
47
48 cv2.waitKey(0)

```

Listing 6.9: korrigiertes Bild erneut aufteilen in die einzelnen Grauwerte

```

1 import numpy as np
2 import cv2
3 import matplotlib.pyplot as plt
4 import matplotlib
5
6 # ----- Aufgabe3.4b -----
7
8 # Vektor zum speichern
9 vec = np.zeros((5, 4))
10
11 # LaTeX Table format
12 print("\hline")
13 print("Stufe & Mittelwert & Hex-Value & Standartabweichung \\\\")

14
15 # for-Schleife um jede der 5 Dateien zu erreichen
16 for x in range(1, 6):
17     # einlesen der aufgeteilten Bilder
18     image = cv2.imread("data/korrigiert" + str(x) + ".png")
19
20     # Durchschnitt des aufgeteilten Bildes bekommen
21     b, g, r, a = cv2.mean(image)
22     # bgr durch 1000 da die hexfunktion Werte zwischen 0 und 1 braucht
23     b1 = b / 1000
24     # Hex Wert berechnen
25     hex = matplotlib.colors.to_hex([b1, b1, b1])
26
27     # LaTeX Tabellenformat für jede Stufe
28     print("\hline")
29     print("Stufe " + str(x) + " & " + str((b + g + r) / 3) + " & " + str(hex)
30           + " & " + str(np.std(image)) + " \\\\")

31
32     vec[x - 1, 0] = x
33     vec[x - 1, 1] = (b + g + r) / 3

```

```

34     vec[x - 1, 2] = np.std(image)
35
36 # Darstellung des Grauwerts in einem Graphen
37 plt.plot([vec[0, 0], vec[1, 0], vec[2, 0], vec[3, 0], vec[4, 0]],
38           [vec[0, 1], vec[1, 1], vec[2, 1], vec[3, 1], vec[4, 1]], 'b')
39 plt.ylabel('Grauwert in RGB')
40 plt.xlabel('Grauwertkeil')
41 plt.grid(True)
42 plt.show()
43
44 # Darstellung der Standartabweichung in einem Graphen
45
46 plt.plot([vec[0, 0], vec[1, 0], vec[2, 0], vec[3, 0], vec[4, 0]],
47           [vec[0, 2], vec[1, 2], vec[2, 2], vec[3, 2], vec[4, 2]], 'b')
48 plt.ylabel('Standartabweichung in RGB')
49 plt.xlabel('Grauwertkeil')
50 plt.grid(True)
51 plt.show()
52
53 print("\hline")

```

Listing 6.10: Mittelwert Hexwert und Standartabweichung berechnen

A.1.4 Quellcode Versuch 4

```

1 import numpy as np
2 import cv2
3
4 # ----- dead -----
5
6 # Values for dark x & y
7 cir1 = 1000
8 cir2 = 1000
9 # Values for light x & y
10 cir3 = 0
11 cir4 = 0
12 # actual pixelvalue
13 value = 0
14 # highscores
15 high1 = 1000
16 high4 = 0
17

```

```

18 # Bild einlesen
19 white1 = cv2.imread('data/white1.png', cv2.IMREAD_GRAYSCALE)
20 image = cv2.imread("data/zzz.png")
21
22 #for-Schleife um auf alle 307200 Pixel zuzugreifen
23 for x in range(0, 480):
24     for y in range(0, 640):
25         # den aktuellen Pixelwert in value speichern
26         value = white1[x, y]
27         # dunkelsten x Punkt mit aktuellem Wert überprüfen
28         if (value < high1):
29             cir1 = x
30             cir2 = y
31             #falls dunkler neuer Highscore
32             high1 = value
33         # hellsten x Punkt mit aktuellem Wert überprüfen
34         if (value > high4):
35             cir3 = x
36             cir4 = y
37             #falls heller neuer Highscore
38             high4 = value
39
40 # Hellster und Dunkelster Pixel ausgeben
41 print("Der dunkelste Punkt liegt bei: " + str(cir1) + " " + str(cir2))
42 print("Der hellste Punkt liegt bei: " + str(cir3) + " " + str(cir4))
43
44 # Kreise bei den dunkelsten bzwh hellsten Pixeln zu machen
45 cv2.circle(image, (cir2, cir1), 5, (142, 123, 228), 2)
46 cv2.circle(image, (cir4, cir3), 5, (46, 179, 66), 2)
47
48 # Bild anzeigen und exportieren
49 cv2.imshow('image', image)
50 cv2.imwrite("zzz.png", image)
51 cv2.waitKey(0)
52 cv2.destroyAllWindows()

```

Listing 6.11: Deadpixels berechnen

```

1 import numpy as np
2 import cv2
3
4 # ----- dead2 ----- #
5
6 # Values for dark x & y
7 cir1 = 1000
8 cir2 = 1000
9 # Values for light x & y
10 cir3 = 0
11 cir4 = 0
12 # actual pixelvalue
13 value = 0
14 # highscores
15 high1 = 1000
16 high4 = 0
17
18 # Bild einlesen
19 white1 = cv2.imread('data/black2.png', cv2.IMREAD_GRAYSCALE)
20 image = cv2.imread("data/zzz.png")
21
22 # for-Schleife um auf alle 307200 Pixel zuzugreifen
23 for x in range(0, 480):
24     for y in range(0, 640):
25         # den aktuellen Pixelwert in value speichern
26         value = white1[x, y]
27         # dunkelsten x Punkt mit aktuellem Wert überprüfen
28         if (value < high1):
29             cir1 = x
30             cir2 = y
31             # falls dunkler neuer Highscore
32             high1 = value
33             # hellsten x Punkt mit aktuellem Wert überprüfen
34             if (value > high4):
35                 cir3 = x
36                 cir4 = y
37                 # falls heller neuer Highscore
38                 high4 = value
39
40 # Hellster und Dunkelster Pixel ausgeben
41 print("Der dunkelste Punkt liegt bei: " + str(cir1) + " " + str(cir2))

```

```
42 print("Der hellste Punkt liegt bei: " + str(cir3) + " " + str(cir4))
43
44 # Kreise bei den dunkelsten bzw hellesten Pixeln zu machen
45 cv2.circle(white1, (cir2, cir1), 5, (142, 123, 228), 2)
46 cv2.circle(white1, (cir4, cir3), 5, (46, 179, 66), 2)
47
48 # Bild anzeigen und exportieren
49 cv2.imshow('white1', white1)
50 cv2.imwrite("zzz2.png", white1)
51 cv2.waitKey(0)
52 cv2.destroyAllWindows()
```

Listing 6.12: Stuckpixels berechnen

A.2 Kameraeinstellungen für OpenCV

<u>Values</u>	
Brightness	133
Contrast	32
Saturation	32
gain	20
exposure	-4
whitebalance	10000

Abstand von Kamera zu Grammweicht :
- 31.5 cm

Abbildung 6.6: Kameraeinstellungen für OpenCV