Print Monitoring

1 AUSGANGSLAGE

Die meisten FDM-3D-Drucker besitzen keine Fehlerüberwachung. Im Falle eines Druckfehlers, wird der Druckprozess nicht unterbrochen. So wird Filament verschwendet oder - der Drucker kann sogar beschädigt werden. Ziel des Projektes ist es, die Druckqualität eines FDM-3D-Druckers durch eine Kamera live zu überwachen. Dadurch kann bei einem fehlerhaften Druckvorgang der Druck automatisiert pausiert oder abgebrochen werden.

2 ZIEL

Bevor ein Druck gestartet wird, soll für jeden Layer des Drucks ein Soll-Bild geplottet werden. Während dem Drucken wird dann nach jedem gedruckten Layer ein Ist-Bild aufgenommen. Aus dem Ist- und Soll-Bild wird dann eine prozentuale Übereinstimmung berechnet. Anhand dieser Übereinstimmung lässt die Druckqualität evaluieren. Grundsätzlich soll das Tool Livebilder einer Webcam verarbeiten. Im Rahmen dieser Projektarbeit soll zudem auch das Verwenden von gespeicherten Aufzeichnungen möglich sein.

In einem ergänzenden Projekt soll eine Kommunikation zwischen Drucker und beispielsweise einem Raspberry PI implementiert werden. Ziel wird sein, einen Fernzugriff auf den Drucker und so den Druckvorgang ermöglichen (starten, stoppen eines Auftrags).

Das sekundäre Ziel ist nicht Teil dieser Arbeit und wird in einem zweiten Projekt umgesetzt. Das fertige Tool soll schlussendlich im MakerStudio der FHNW Brugg-Windisch eingesetzt werden.

3 VORBEREITUNGEN

3.1 AUFBAU DER KAMERA UND BELEUCHTUNG (TESTAUFBAU)

Um Schatten auf dem Druckbett zu verringern, kommt ein LED-Strahler zum Einsatz, welcher den Druck indirekt beleuchtet. Zum Aufzeichnen der Bilder wird eine Webcam senkrecht über dem Drucker eingesetzt, welche an einer Aufhängung befestig wurde.

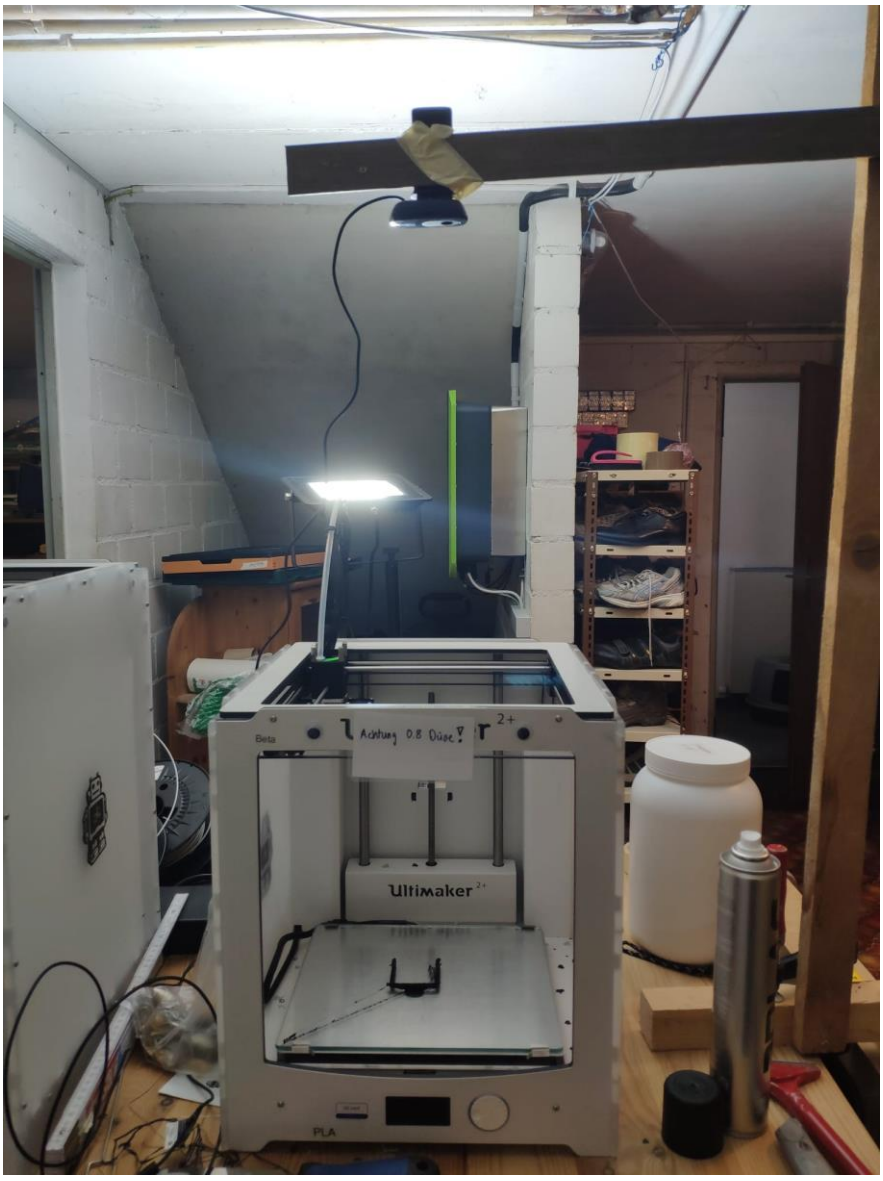


Abbildung 1: Testaufbau der Projektarbeit

3.2 KONFIGURATION

Um das Tool für eine neue Kamera(-positon) oder eine neue Videodatei verwenden zu können, muss das Tool einmalig manuell konfiguriert werden. Die ROI (region of interest) des Druckers und die ROI des Druckkopfes müssen von Hand ausgewählt werden. Die Konfiguration wird danach in der Datei config.ini abgelegt. Genaue Instruktionen befinden sich in der Datei *readme.md*.

Reproduktion der Testresultate

Um die dokumentierten Resultate zu reproduzieren, folgen Sie bitte den Instruktionen der GitLab-Seite (*readme.md)*.

4 UMSETZUNG

Das Tool wurde in Python umgesetzt. Um eine Überwachung zu starten oder eine neue Kamera zu konfigurieren, wurde das main.py mit einem Command-Line-Interface ergänzt.

Wie das CLI verwendet wird und welche Packages installiert werden müssen wird in dem separaten Dokument «readme.md» auf Github erklärt.

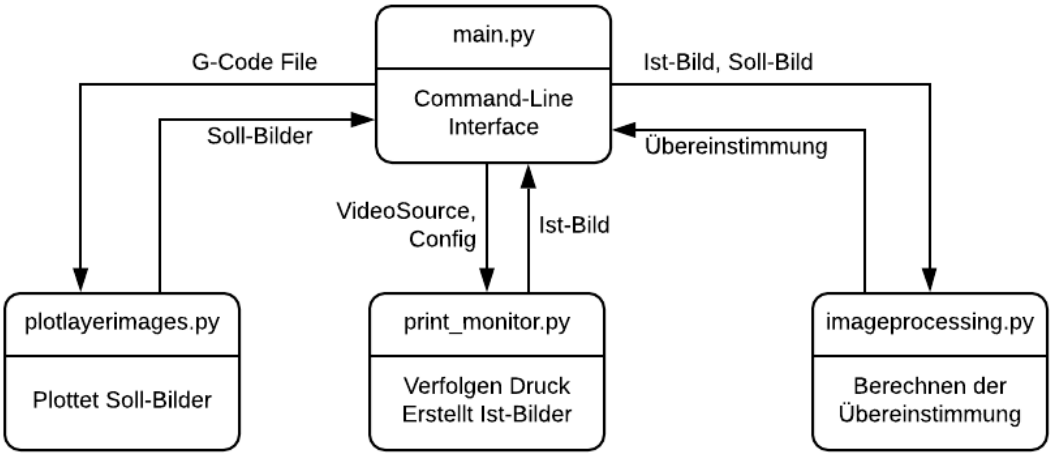


Abbildung 2: Software Architektur

4.1 Plotten der Soll-Bilder

Um ein 3D-Modell drucken zu können, wird die zu druckende Datei (normalerweise .stl) von einem Slicer-Programm (Beispielsweise Cura) in einen G-code gewandelt. Die G-code-Datei besteht aus einer simplen Sammlung von X, Y und Z Koordinaten, welche vom Druckkopf schrittweise angefahren werden sollen, bis schlussendlich das komplette Modell gedruckt ist. In unserem Fall wird der G-code so erweitert, dass der Druckkopf nach jedem Layer zurück in die Startposition fährt.

In der Datei «plotlayerimages.py» wird dieser G-code gelesen, die X und Y Koordinaten werden mittels MatPlotLib Schicht für Schicht geplottet und als Bild abgespeichert. Somit entsteht eine Sammlung von Soll-Bildern. Für das Plotten des G-codes wurde eine Library von *zhangyaqi1989* verwendet [1]

4.2 Erstellen der Ist-Bilder

Sobald sämtliche Soll-Bilder generiert wurden, wird der G-code dem Drucker übergeben, der Druckauftrag wird gestartet und die Überwachung des Druckvorgangs mittels «print_monitor.py» wird initialisiert. Das Script erkennt, wann ein Layer fertig gedruckt wurde. Das Überwachen des Drucks funktioniert folgendermassen:

Für jedes Video-Frame werden folgende Schritte ausgeführt:

1. Konvertieren des Bildes in den **HSV-Bereich**
2. Mittels OpenCV einen **Color-Treshhold** auf das Bild anwenden. Um den Druckkopf freizustellen. Danach ein **Errode** und **Dilate** um das Binärbild zu bereinigen.
3. Das berechnete Binärbild wird mit den Binärbildern der letzten drei Frames **And-Verknüpft.** So können weitere Fehler eliminiert werden und eine höhere Stabilität wird erreicht.
4. Es wird überprüft, ob sich die BoundingBox der Binärbildes innerhalb oder ausserhalb des definierten Ausgangspositions-Bereichs befindet.
5. Die gewonnene Information wird als StateMachine abgebildet, welche dann eine Aussage über den Status des Druckers macht. Die möglichen Drucker-Zustände sind:
    a. STARTING
    b. LAYER_PRINTING
    c. LAYER_FINISHED
    d. PRINT_FINISHED

Sobald der Status LAYER_FINISHED erreicht wird, wird das aktuelle Frame als Ist-Bild gespeichert und mit dem passenden Soll-Bild verglichen (siehe folgendes Kapitel).

4.3 BERECHNEN DER ÜBEREINSTIMMUNG

Aus den vorhergehenden beiden Schritten wurden ein Soll und Ist-Bild für einen gedruckten Layer generiert.

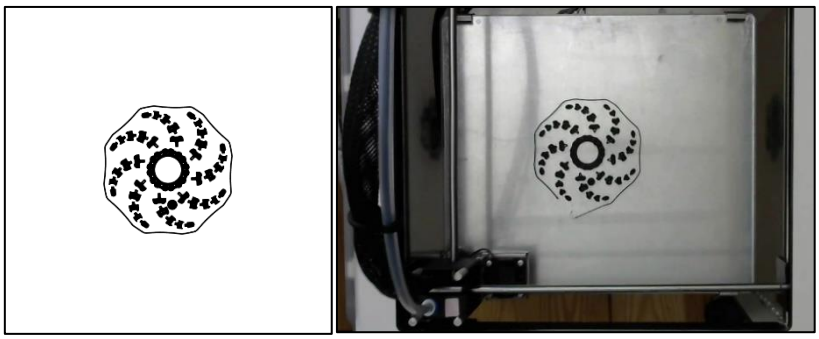


Abbildung 3: Soll-Bild (links) und Ist-Bild (rechts)

So können jetzt in der Datei «imageprocessing.py» mit der Funktion getMatch(..) diese beiden Bilder verglichen werden, um eine prozentuale Übereinstimmung zu berechnen.

4.3.1 Freistellen des Drucks

Zuerst muss der Druck auf dem Ist-Bild freigestellt werden. Dazu wird das zu Beginn erstellte Bild des leeren Druckers als Maske für das Ist-Bild verwendet. Um dies zu erreichen wird ein Canny-Edge-Dedector auf das Bild ohne Druck angewendet. Die berechneten Kanten werden dann mittels einem Dilate vergrössert (Code in imageprocessing.getMask(…)).

Sobald die Maske erstellt ist, wird ein Canny-Edge-Dedector auf das Ist-Bild angewendet. Danach werden die Kanten des Ist-Bildes maskiert und es entsteht ein freigestelltes Bild der Kanten des Drucks.

4.3.2 Berechnen der Übereinstimmung

Um Aussagen zur Druckqualität zu machen, muss man die Differenz von Soll- und Ist-Bild auswerten können. Zu diesem Zweck werden die Kanten des Soll-Bildes detektiert und das Bild wird auf sein BoundingRect zugeschnitten. Das zugeschnittene Soll-Bild wird nun als Template für das OpenCV Templatematching verwendet. Der Templatematching Algorithmus funktioniert folgendermassen: Das Template wird Pixel für Pixel über das maskierte Ist-Bild geschoben. Nach jedem Pixel wird die Überlappung zwischen Template und Ist-Bild berechnet. Als Resultat wird die maximale Überlappung sowie die Koordinaten der maximalen Überlappung ausgegeben. Verschiedene Versuche ergaben, dass der Algorithmus zuverlässiger funktioniert, wenn er lediglich auf die Kanten des Drucks angewendet wird, anstatt auf das ganze Binärbild von Druck und Template. [2]

Abbildung 4 Template (links) und gefundene Position und Grösse in Ist-Bild (rechts)

Um das TemplateMatching skalierungsinvariant und somit stabiler umzusetzen, wird der Algorithmus für verschiedene Skalierungen des Template durchgeführt. Als Resultat wird der grösste Match aller Durchgänge verwendet. [3]

Somit sind nun die exakte Position und Grösse des Drucks im Ist-Bild bekannt.

Mit dieser Information wird nun das Soll-Bild entsprechen des Ist-Bildes skaliert und positioniert. Das Resultierende Soll-Bild ist in Abbildung 6 dargestellt.

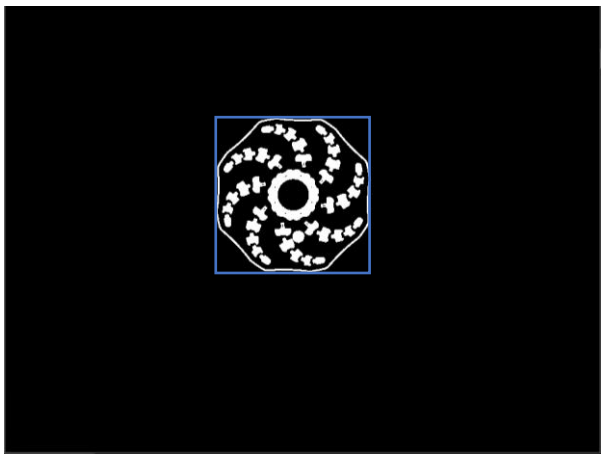


Abbildung 5 Resultierendes Soll-Bild mit Druck an Position von Druck in Ist-Bild

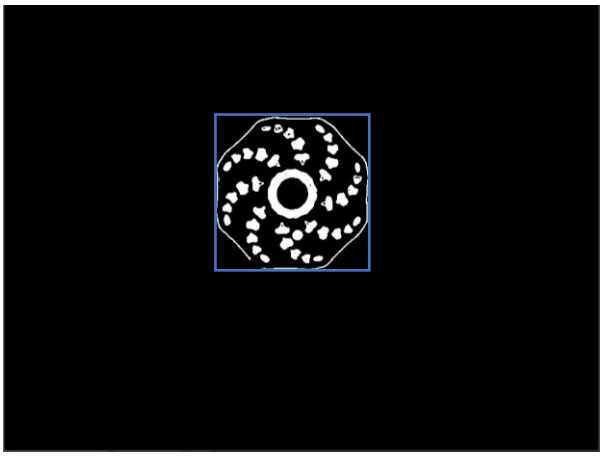


Abbildung 6 Ist-Bild

Zur Berechnung der Übereinstimmung werden die beiden binären Ist- und Soll-Bilder mit einem bitweisen XOR verglichen. Daraus entsteht ein Differenzbild (Abbildung 8), woraus sich schlussendlich die prozentuale Übereinstimmung berechnen.

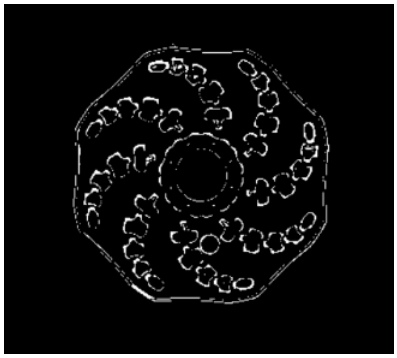
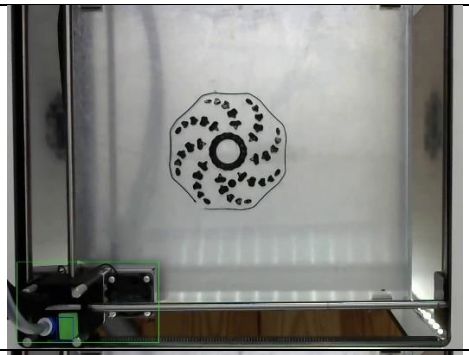
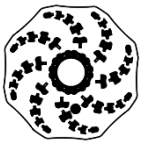
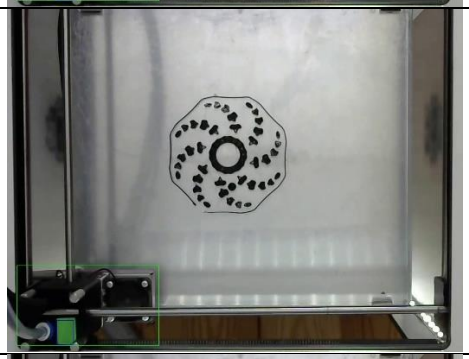
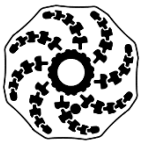
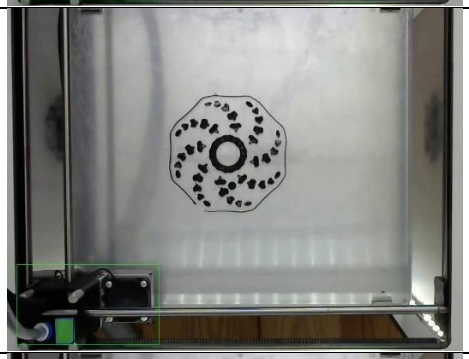
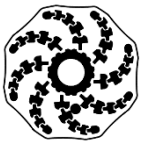
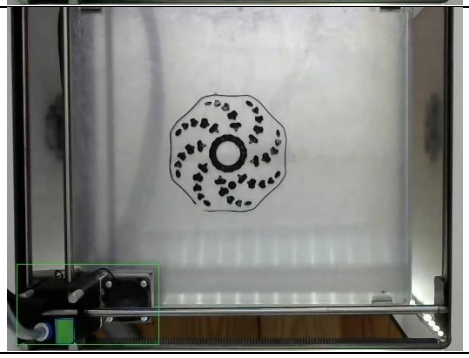
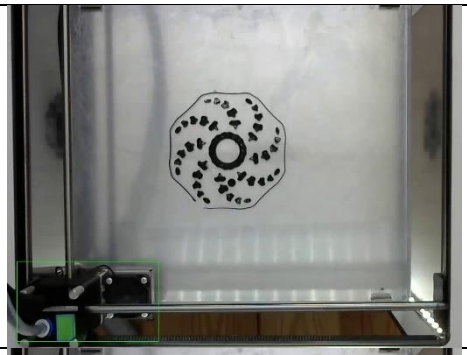
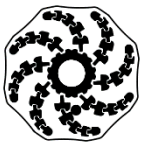
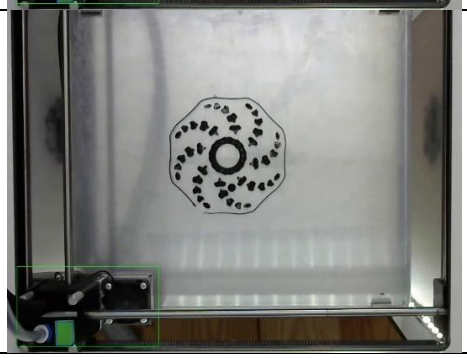
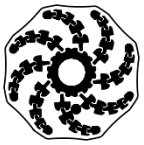


Abbildung 7 Differenzbild

4.4 TESTS

TestNr 1.
VideoSource: octopus.mp4
G-code: octopus.gcode

LayerNr.	Ist	Soll	Resultat
1			0.686 -> OK
2			0.610 -> OK
3			0.591 -> OK
4			0.560 -> OK

5			0.558 -> OK
6			0.547 -> Druck wird abgebrochen

Test erfolgreich.

Ab Mitte des 2. Layers war die Düse des Druckers verstopft und es konnte kein Material mehr aufgetragen werden. Der Druckfehler nahm mit jedem Layer zu und erreichte den Grenzwert von 55%, wo der Druck abgebrochen wurde.

5 FAZIT

Die Projektarbeit wurde stabil und zuverlässig umgesetzt. Häufige Druckfehler, wie zum Beispiel eine verstopfte Düse, zu wenig Filament oder fehlende Stützstrukturen werden erkannt. Somit kann das Tool in einer produktiven Umgebung eingesetzt werden, um den Verschleiss der Drucker und das Verschwenden von Filament zu verkleinern.

Druckfehler, welche auf der Draufsicht des Drucks nicht sichtbar sind, können aber nicht erkannt werden. Das Tool könnte deshalb noch mit einer zweiten Kamera, welche seitlich auf den Druck gerichtet ist, erweitert werden, um eine komplette Qualitätsüberwachung zu garantieren.

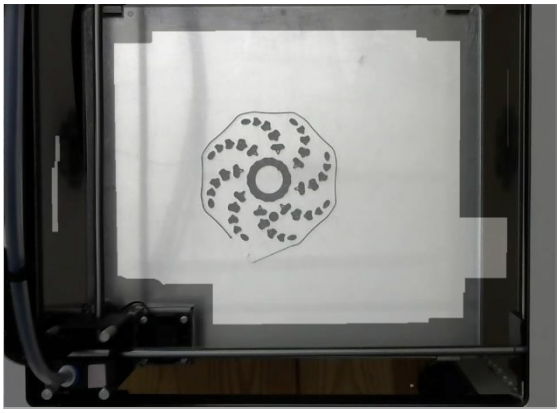


Abbildung 8 Maskierter Druckbereich

Zudem kann nicht die komplette Druckplatte überwacht werden. Aufgrund der gewählten Methode zur Freistellung des Drucks, wird die maximale Grösse des Drucks um zirka 5 – 10 cm kleiner. In Abbildung 9 ist der berduckbare Bereich hell dargestellt.

Zudem funktioniert das Freistellen nur bei dunkeln Filamenten, welche ausreichend Kontrast zu der Druckplatte haben, genügend stabil. Um weitere Farben zu unterstützen, müssten die Grenzwerte des verwendete Edge-Dedectors und der Treshholds dynamischer gewählt werden.

Mögliche Verbesserungen

- Landmarke / QRCode anstelle des grünen Klebers. So können auch nicht schwarze Filamente besser verwendet werden.
- Es wird nur die Anzahl der übereinstimmenden Pixel betrachtet und nicht die Form als Ganzes.
- Treshholds dynamischer wählen

6 ABBILDUNGSVERZEICHNIS

7 QUELLEN

[1] Y. Zhang, „zhangyaqi1989/Gcode-Reader“, Juni 08, 2020. https://github.com/zhangyaqi1989/Gcode-Reader (zugegriffen Juni 10, 2020).

[2] „Object Detection — OpenCV 2.4.13.7 documentation“. https://docs.opencv.org/2.4/modules/imgproc/doc/object_detection.html?highlight=matchtemplate#cv2.matchTemplate (zugegriffen Juni 11, 2020).

[3] „Multi-scale Template Matching using Python and OpenCV“, *PyImageSearch*, Jan. 26, 2015. https://www.pyimagesearch.com/2015/01/26/multi-scale-template-matching-using-python-opencv/ (zugegriffen Juni 10, 2020).

8 REFERENZIERUNG DER PYTHON LIBRARIES:

Open-CV

Open Source Computer Vision Library, Itseez, 2015, https://github.com/itseez/opencv

Matplotlib

Thomas A Caswell, Michael Droettboom, Antony Lee, John Hunter, Eric Firing, David Stansby, … Jan Katins. (2020, March 18). matplotlib/matplotlib: REL: v3.2.1 (Version v3.2.1). Zenodo. http://doi.org/10.5281/zenodo.3714460

Numpy

Travis E. Oliphant. A guide to NumPy, USA: Trelgol Publishing, (2006).

Stéfan van der Walt, S. Chris Colbert and Gaël Varoquaux. The NumPy Array: A Structure for Efficient Numerical Computation, Computing in Science & Engineering, 13, 22-30 (2011), DOI:10.1109/MCSE.2011.37 (publisher link)

Seaborn

Michael Waskom, Olga Botvinnik, Drew O'Kane, Paul Hobson, Saulius Lukauskas, David C Gemperline, … Adel Qalieh. (2017, September 3). mwaskom/seaborn: v0.8.1 (September 2017) (Version v0.8.1). Zenodo. http://doi.org/10.5281/zenodo.883859

Librarie Versionen

Instruktionen betreffend der Reproduzierbarkeit sind auf GitLab im readme.md festgehalten.

Versionen:

matplotlib==3.2.1
numpy==1.18.5
opencv-python==4.2.0.34
pandas==1.0.4
seaborn==0.10.1