Print Monitoring

Anmerkungen / toDo:

* Am Anfang den Testaufbau erklären
* Kapiteltiefe nur auf 2 Ebenen?

# 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.

# 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 Schritt umgesetzt. Das fertige Tool soll schlussendlich im MakerStudio der FHNW Brugg-Windisch eingesetzt werden.

# Umsetzung

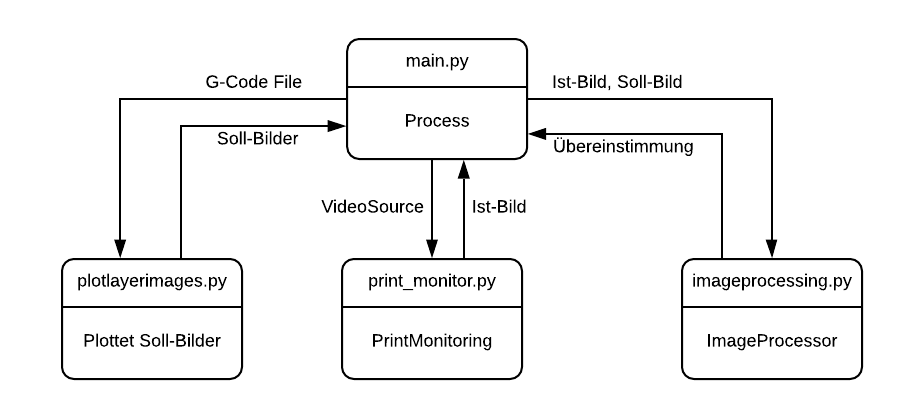


Abbildung 1: Software Architektur

Das Tool wurde in Python umgesetzt. Um eine Überwachung zu starten oder eine neue Kamera zu konfigurieren, wurde in der main.py ein Command-Line-Interface enzwickelt.

Wie das CLI verwendet wird und welche Packages installiert werden müssen wird in dem separaten Dokument «readme.pdf» erklärt. Reproduktion der Tests

# Vorbereitungen

* Aufbau der Kamera (Testaufbau)
* Erstellen des G-codes
* Konfigurieren der Kameraposition (ROI)

## Konfiguration

Um das Tool für eine neue Kamera oder eine neue Videodatei verwenden zu können, muss dies einmalig manuell konfiguriert werden. Die ROI des Druckers und die ROI des Druckkopfes müssen ausgewählt werden. Die Konfiguration wird danach in der Datei settings.ini abgelegt. Genaue Instruktionen befinden sich in der Datei *readme.pdf*.

## Hardware

**Raspberry**

**Kamera**

**Beleuchtung**

## 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 verarbeitet. 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 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] ([https://github.com/zhangyaqi1989/G-code-Reader](https://github.com/zhangyaqi1989/Gcode-Reader))

## 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:

Der G-code wird so generiert, dass der Kopf des Druckers nach jedem gedruckten Layer an seine Ausgangsposition fährt. Diese Ausgangsposition muss nach dem Positionieren der Kamera einmalig manuell auf dem Bild ausgewählt werden. Siehe Dokument «Anleitung» Kapitel «Konfiguration»

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:
   1. STARTING
   2. LAYER\_PRINTING
   3. LAYER\_FINISHED
   4. PRINT\_FINISHED

Sobald der Status LAYER\_FINISHED erreicht wird, wird das aktuelle Frame als IST-Bild an das main.py übergeben, wo es dann mit dem passenden Soll-Bild verglichen wird.

## Berechnen der Übereinstimmung

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

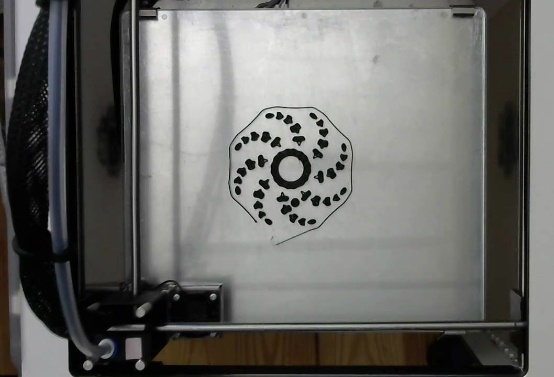
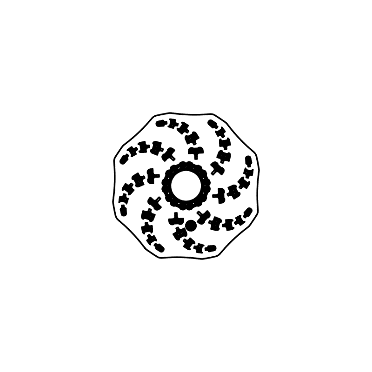


Abbildung 2: Soll-Bild Abbildung 3 Ist-Bild

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

### Freistellen des Drucks

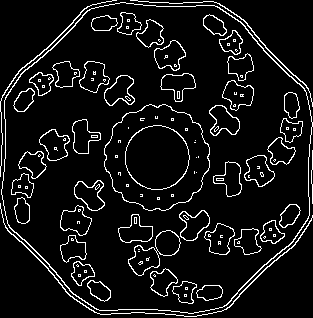
Zuerst muss der Druck auf dem Ist-Bild freigestellt werden. Dazu wird vor Beginn des Drucks durch das PrintMonitoring ein Bild des leeren Druckers angefertigt. Dieses Bild wird als Maske für das Ist-Bild verwendet werden. Um dies zu erreichen wird ein Canny-Edge-Dedector auf das Bild angewendet. Die 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 Druckes.

### 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 Druckes angewendet wird, anstatt auf das ganze Binärbild von Druck und Template. (<https://docs.opencv.org/2.4/modules/imgproc/doc/object_detection.html?highlight=matchtemplate#cv2.matchTemplate>)

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. [2] (<https://www.pyimagesearch.com/2015/01/26/multi-scale-template-matching-using-python-opencv/>)

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 4 dargestellt.

Abbildung 6 Ist-Bild

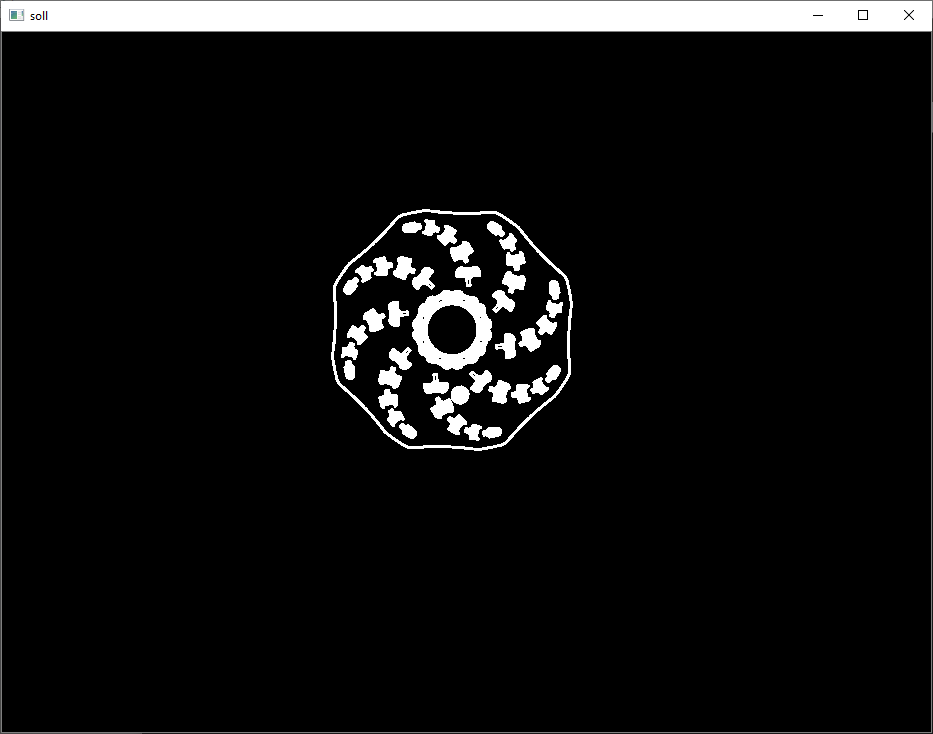
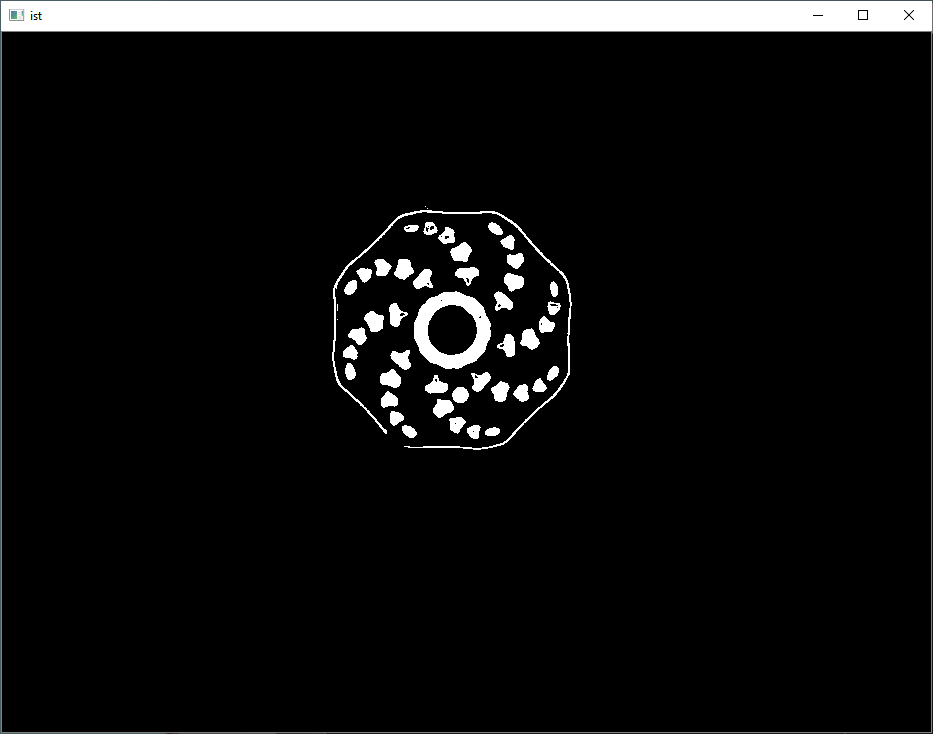


Abbildung 5 Resultierendes Soll-Bild mit Druck an Position von Druck in Ist-Bild

Zur Berechnung der Übereinstimmung werden die beiden binären Ist- und Soll-Bilder mit einem bitweisen XOR verglichen. Daraus entsteht ein Differenzbild (Abbildung 6), woraus sich schlussendlich die prozentuale Übereinstimmung berechnen.



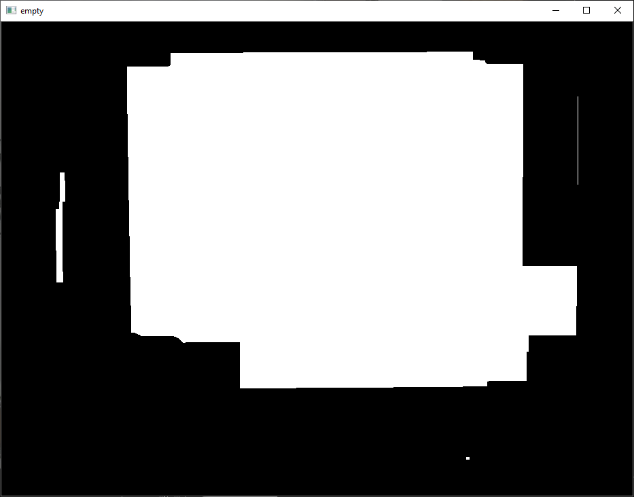
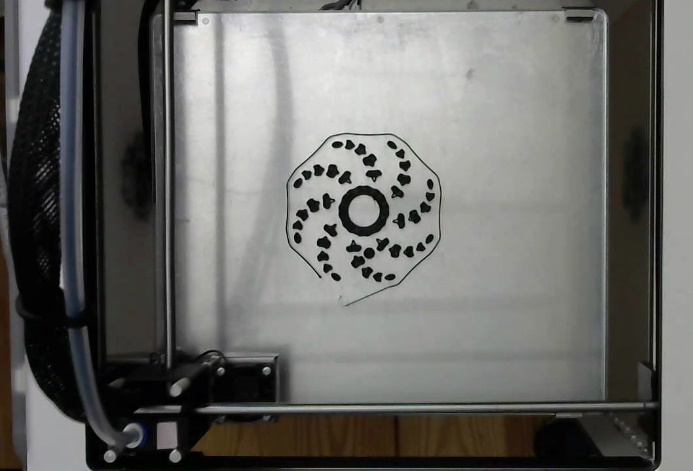
Abbildung 7 Differenzbild

## Tests

|  |  |  |  |
| --- | --- | --- | --- |
| TestNr 1.  VideoSource: testprint4.mp4  G-code: Octpus\_PreProcessed.G-code | | | |
| LayerNr. | Ist | Soll | Resultat |
| 1 |  |  | 0.929  -> OK |
| 2 |  |  | 0.901  -> OK |
| 3 |  |  | 0.884  -> OK |
| 4 |  |  | 0.871  -> OK |
| 5 |  |  | 0.851  -> OK |
| 6 |  |  | 0.846  -> 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 85%, wo der Druck abgebrochen wurde. | | | |

# 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 Druckes 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.

Zudem kann nicht die komplette Druckplatte überwacht werden. Aufgrund der gewählten Methode zur Freistellung des Druckes, wird die maximale Grösse des Druckes um zirka 5 – 10 cm kleiner. In Abbildung 7 ist der berduckbare Bereich hell dargestellt.

Abbildung 8 Maskierter Druckbereich

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.

# Abbildungsverzeichnis

[Abbildung 1: Soll-Bild Abbildung 2 Ist-Bild 3](#_Toc42503808)

[Abbildung 3 Template (links) und gefundene Position und Grösse in Ist-Bild (rechts) 4](file:///C:\Users\Manuel\Documents\repos\bverproject\Dokumentation.docx#_Toc42503809)

[Abbildung 4 Resultierendes Soll-Bild mit Druck an Position von Druck in Ist-Bild 4](file:///C:\Users\Manuel\Documents\repos\bverproject\Dokumentation.docx#_Toc42503810)

[Abbildung 5 Ist-Bild 4](file:///C:\Users\Manuel\Documents\repos\bverproject\Dokumentation.docx#_Toc42503811)

[Abbildung 6 Differenzbild 5](#_Toc42503812)

# Quellen

[1] Y. Zhang, *zhangyaqi1989/Gcode-Reader*. 2020.

[2] „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).

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

Librarie Versionen

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