

Bildqualitätsverbesserung für Bilder von Dokumenten

Algorithm Engineering 2023 Projekt Paper

Mario Baars
Friedrich Schiller University Jena
Germany
mario.louis.baars@uni-jena.de

Eric Günl
Friedrich Schiller University Jena
Germany
eric.günl@uni-jena.de

ABSTRACT

In diesem Paper wird über die Konzeption, Entwicklung und Optimierung eines Bildqualitätsverbesserungsprogramm berichtet. Es werden die verwendeten Algorithmen, die Prototypisierung in Python, bis hin zu einer optimierten, parallelisierten Version des Programms in C++ beleuchtet. Darüber hinaus werden Performance, Grad der Effektivität und die Design Entscheidungen des Programms diskutiert.

Entstanden ist hierbei ein plattform-übergreifendes Programm, welches system- und komplizierungs-agnostisch schlechte Bilder von Dokumenten in eine gut druckbare Form überführt. Durch die Portabilität und gute Performance, kann dieses Programm effektiv in der Kommandozeile für die Bildqualitätsverbesserung verwendet werden. Darüber hinaus ermöglicht das Bereitstellen von zwei Parametern als Kommandozeilenoption, den Nutzer ad-hoc ideale Parameter für das zuverbessernde Bild zufinden, ohne das Programm erneut kompilieren zu müssen.

KEYWORDS

Image Enhancement, Programm Optimization, Portability

1 ZIEL DES PROJEKTS

Das Projekt Bildverbesserung für aufgenommene Dokumente hat das Ziel ein Foto von einem schriftlichen Dokument, welches im ppm P3 Format vorliegt, so zu verbessern, dass Schrift und Hintergrund in einem hohen Kontrast zueinander stehen. Dabei ist die best mögliche Verbesserung, dass für jeden Pixel entschieden wird, ob er zur Schrift oder zum Hintergrund gehört, wie das bei einem Dokument in einem Texteditor der Fall wäre. Wir haben uns dazu entschieden tatsächlich für jeden Pixel zu entscheiden ob er zur Schrift oder zum Hintergrund gehört und keine Grauwertabstufungen in unserem Ausgabebild (auch im ppm P3 Format) zu erlauben. Dies ist durch Bildrauschen, Helligkeitsgradienten, oder Defokussierung eine herausfordernde Aufgabe die wahrscheinlich meist keine 100%ige Trefferquote für jeden Pixel haben wird. Dennoch versuchen wir mit unserem Programm so nah wie möglich an dieses Ziel heranzukommen.

2 PROGRAMM

Das komplette Programm wurde in Python entwickelt. Die hier aufgezählten Filter und viele weitere, die nicht effizient oder wirksam waren, wurden in Python implementiert und getestet. Danach wurde das Programm in C++ übersetzt. Nachdem es mit allen Funktionen in C++ implementiert war, wurde das Programm optimiert.

Das Programm lässt sich in grob in 4 Bereiche unterteilen.

AEPRO 2023, March 15, 2023, Jena, Germany. Copyright ©2023 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

2.1 Einlesen des Bildes

Die ppm Datei wird in einen Buffer geladen. Danach werden einzeln das Dateiformat, die Dimensionen und der Maximale Wert eingelesen. Danach wird der Buffer Pixel für Pixel durchlaufen und aus den RGB Werten ein Grauwert errechnet. Dies geschieht mit den Color conversion Werten von *OpenCV*. Die errechneten Werten werden in einem 2D Array gespeichert.

2.2 Algorithmen

2.2.1 *Moving Normalization Filter*. Dieser Filter bewirkt, dass dunkle und helle Gebiete im Bild aneinander angeglichen werden. Ein möglicher Helligkeitsgradient der über das gesamte Bild verläuft wird mit diesem Filter geglättet, was in der Abbildung 1 zusehen ist. Der obere Graph zeigt mögliche Originaldaten, bei welchen der Wert über den gesamten Bereich sinkt. Die 2 Kerben bei 20 und 60 zeigen hier die geringeren Werte (wo im Bild die schwarze Schrift wäre) an, die detektiert werden sollen. Mit einem festen Schwellwert ist dies nicht möglich. Deshalb teilt man jeden einzelnen Wert durch den Mittelwert seiner Umgebungswerte. Dies normalisiert den Hintergrund auf den Wert 1 ohne dabei örtliche Gradienten wie die Schrift auszulöschen. Der untere Graph zeigt, dass die Kerben immer noch deutlich detektierbar sind und der Hintergrund einen festen Wert angenommen hat. Um die Kerben herum ist der Wert sogar etwas höher als der Hintergrundwert, was zusätzlich nützlich ist um Schrift von Hintergrund zu separieren [2].

2.2.2 *Adaptiver Schwellwert Filter*. Der Schwellwert um zu entscheiden, ob der Pixel Hintergrund oder Schrift ist, wird in unserem Algorithmus adaptiv anhand von 2 lokalen Variablen und 3 Parametern errechnet. Dazu wird das Array in Bereiche aufgeteilt. Von diesen Bereichen wird die Varianz des schon gefilterten Bildes berechnet. Dies ist die erste Variable der Schwellwertformel. Für die zweite Variable wird aus dem original Bild ein Histogramm linearisiertes Bild erzeugt. Dies ist notwendig, da besonders dunkle und auch sehr helle Bilder einen ähnlichen Wertebereich haben müssen, damit die Schwellwertformel für alle Bilder funktioniert. Das Histogramm linearisierte Bild wird in die selben Bereiche unterteilt und für jeden Bereich wird der maximale Wert ermittelt. Dies ist die zweite Variable der Formel. Die Formel lautet:

$$S = m \cdot \text{Var} + n + \sqrt{b_{\max}} \cdot f \quad (1)$$

Var ist die Varianz. b_{\max} ist der maximale wert des linearisierten Bildes. Die Parameter sind:

- m : Faktor (Gewicht) für die Varianz
- n : Summand um die zu justieren
- f : Faktor für die Wurzel aus der maximalen Helligkeit

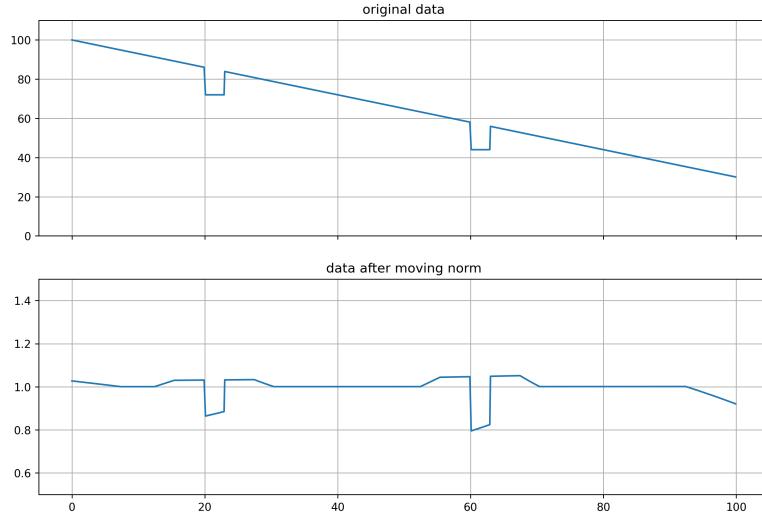


Figure 1: Die Datenverteilung bevor (oben) und nach (unten) der Anwendung des *Moving Normalization Filter*.

Diese Formel ist durch umfangreiches Testen von Variablen, die einen Einfluss auf den Schwellwert haben entstanden. Die Parameter wurden so ausgewählt, dass für die Beispielbilder das bestmögliche Ergebnis nach unserer visuellen Beurteilung herauskam. Sowohl der Parameter n als auch m sind bei Aufruf des Programms einstellbar, um den Schwellwert für das jeweilige Bild anzupassen. Dabei sollte der Wert n zwischen 0.2 und 0.9 liegen. Falls des Verbesserte Bild zu viele dunkle Stellen aufweist sollte der n Wert gesenkt werden. Wenn jedoch Teile der Schrift nicht erkannt werden und als Hintergrund detektiert werden muss der n Wert erhöht werden. Selbiges gilt für den m Wert. Hier liegt der Bereich zwischen 0.2 und 0.6.

Im Programm wurde kein Filter zur Rauschunterdrückung implementiert, da nach einigen Test mit zum Beispiel einem adaptiven Mittelwertfilter, welcher die globale und lokale Varianz des Bildes berücksichtigt, festgestellt wurde, dass keine signifikante Verbesserung des Bildes aufgetreten ist. Die meisten Bilder von Dokumenten sind nicht in einem Maß verrauscht, sodass Schrift nicht vom Hintergrund unterscheidbar wäre [1].

2.3 Erstellen einer Ausgabe ppm Datei

Erstellen einer Ausgabe ppm Datei Das erzeugte boolesche Array mit nur Einsen und Nullen wird Stück für Stück mittels eines Buffers der Größe 80KBit in eine ppm P3 Datei geschrieben.

2.4 Optimierung

Mit Zeitmessungen der einzelnen Bereiche konnte für 3 Versionen des Programms die Laufzeiten verglichen werden. Dabei wurden das Programm in Python, das nicht optimierte Programm in C++ und das optimierte Programm in C++ verglichen: Als Bild wurde dabei

eine ppm P3 Datei mit 18.4 MB verwendet. Die Laufzeitmessungen wurden auf dem selben PC, welcher 4 Kerne besitzt, durchgeführt.

Table 1: Performance zwischen den unterschiedlichen Iterationen des Bilderverbesserungsprogramms in Millisekunden.

Stadium Funktion	Python	C++	C++ optimiert
Laden des Bildes	910	471	116
Moving Norm Filter	69500	5118	1203
Adaptiver Schwellwert Filter	2270	277	295
Speichern des Bilds	680	474	210
Gesamtes Programm	73360	6341	1825

Wie in der Tabelle 1 zu sehen ist verbessert sich die Performance des Python Programms zum C++ Programm um mehr als das 10-fache. Auch die Optimierung des C++ codes bringt eine Performance Steigerung von 250%.

2.4.1 Beschleunigung des Ladevorgangs der ppm Datei in ein Array. In der nicht optimierten Variante des C++ Programms wird jeder Wert einzeln aus der ppm Datei in eine Pixel Struktur geschrieben, welche in ein 2D Array gespeichert wird. Danach wird dieses Array durchlaufen und für jeden RGB Wert der Grauwert berechnet und in ein neues Array geschrieben. Die optimierte Version lädt die gesamte ppm Datei in einen Buffer. Dieser Buffer wird dann Char für Char durchlaufen und die RGB Werte in Grauwerte umgerechnet und in ein Array gespeichert. Dies bringt eine performance Steigerung von 300%.

2.4.2 Parallelisierung des Moving Norm Filters. Um diese Funktion zu optimieren wurden 2 mal 2 for Schleifen mit omp for

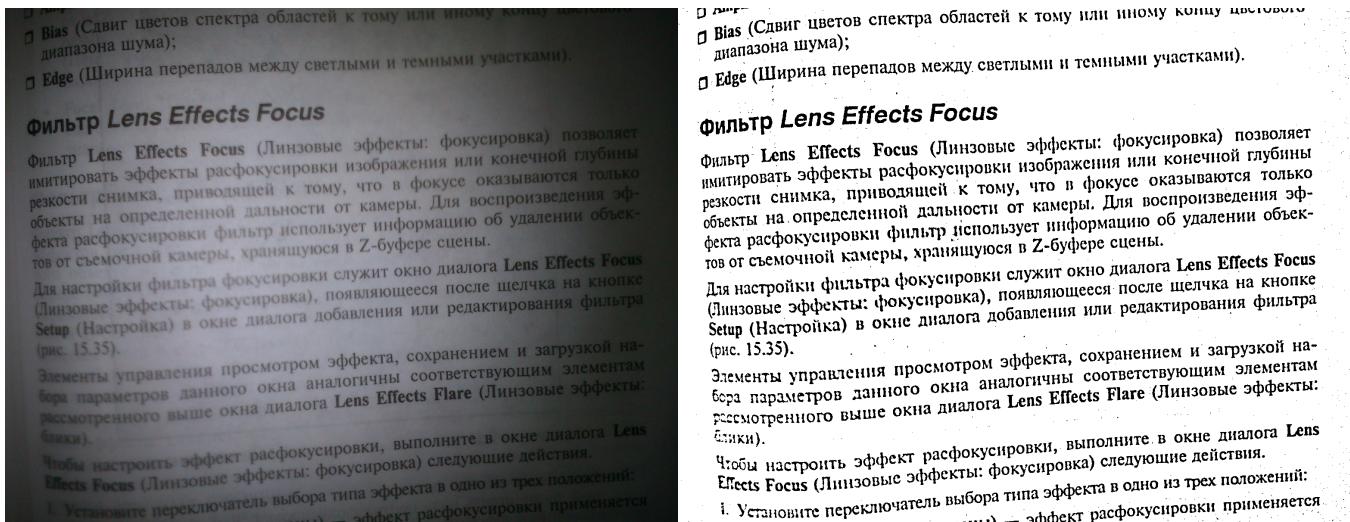


Figure 2: Vergleich zwischen dem bereitgestellten Bild (links) und dem durch das Programm verbesserte Bild (rechts). Die verwendeten Parameter des *Adaptive Thresholds* für die Verbesserung waren $m = 0.5$ und $n = 0.84$.

`collapse(2)` `schedule(static)` parallelisiert. Je nach Anzahl der Kerne des PCs kann dies zu unterschiedlichen Performance Steigerungen führen. Die Optimierung dieser Funktion ist am wichtigsten, da Sie auch der Bottleneck des Programms ist.

2.4.3 Beschleunigung des Speicherns der resultierenden Bild Datei. Auch bei dieser Funktion werden die Daten zur Optimierung der Performance in einen Buffer geschrieben, welcher dann in die Datei geschrieben wird. Dies bringt eine Verbesserung im Vergleich zur Variante des Wert für Wert in die Datei schreiben um 100%.

Zudem können die Flags `-Ofast` und `-ffast-math` genutzt werden um das schon optimierte Programm zu verbessern. Mit diesen beiden Flags erhält man eine Laufzeit des kompletten Programms von 422 ms und damit eine weitere Performanzsteigerung von 300%.

2.5 Tests

Die Natur des Programms deutet eher auf die Verwendung von Systemtests statt Unit-Tests oder Integrationstests. Das Kernstück der Programme sind die Algorithmen, die in den vorigen Abschnitten genauer beschrieben wurden. Diese Algorithmen sind so designed, dass sie bei einem gültigen Eingabebild im P3 ppm-Format auch gültige Zwischenergebnisse liefern. Somit zeigt sich die bedeutsame Gültigkeit der Funktionen erst durch die Anwendung aller Algorithmen in einem integralen System. Deshalb wurde das Programm anhand von der Anwendung auf verschiedenen beschafften Eingabebildern getestet.

2.6 CMake

Für maximale Portabilität wurde CMake gewählt, um dessen plattformübergreifende Eigenschaften auszunutzen. Es gibt 2 CMakeLists.txt Dateien in den die Instruktionen für das Bauen des Programms zu finden sind. Beim Aufruf der CMakeLists.txt in dem selben Ordner wie die main.cpp Datei wird die sekundäre CMakeLists.txt Datei, die die image_enhancer.lib Bibliothek baut, aufgerufen.

- Bias (Сдвиг цветов спектра областей к тому или иному концу цветового диапазона шума);
- Edge (Ширина перепадов между светлыми и темными участками);
- Edge (Ширина перепадов между светлыми и темными участками).

Фильтр Lens Effects Focus

Фильтр Lens Effects Focus (Линзовые эффекты: фокусировка) позволяет имитировать эффекты расфокусировки изображения или конечной глубины резкости снимка, приводящий к тому, что в фокусе оказываются только объекты на определенной дальности от камеры. Для воспроизведения эффекта расфокусировки фильтр использует информацию об удалении объектов от съемочной камеры, хранящуюся в Z-буфере сцены.

Для настройки фильтра фокусировки служит окно диалога Lens Effects Focus (Линзовые эффекты: фокусировка), появляющееся после щелчка на кнопке Setup (Настройка) в окне диалога добавления или редактирования фильтра (рис. 15.35).

Элементы управления просмотром эффекта, сохранением и загрузкой набора параметров данного окна аналогичны соответствующим элементам рассмотренного выше окна диалога Lens Effects Flare (Линзовые эффекты: блеск).

Чтобы настроить эффект расфокусировки, выполните в окне диалога Lens Effects Focus (Линзовые эффекты: фокусировка) следующие действия:

- i. Установите переключатель выбора типа эффекта в одно из трех положений:

ii. Установите переключатель выбора типа эффекта — эффект расфокусировки применяется

Die minimal Anforderungen dieses Programms zu kompilieren sind ein C/C++ Compiler, mit C++14-Standard und die CMake Version 3.9.

3 ERGEBNISSE

Wie in Abbildung 3 zu sehen ist hängt die Qualität des Bildes stark von den Parametern ab, die übergeben werden. Je dunkler das Bild, desto geringer müssen die Parameter n und m gewählt werden. Wenn man nach der Nutzung des Enhancers nicht zufrieden ist, sollte man einen anderen Parametersatz ausprobieren. Zu beachten ist auch das der Parameter n einen größeren Einfluss auf das Ergebnis hat als der Parameter m . Für jedes der 3 Testbilder lässt sich jedoch schnell ein Parametersatz finden, für den der Algorithmus sehr gut zwischen Schriftpixeln und Hintergrundpixeln unterscheiden kann.

3.1 Beurteilung der Ergebnisbildqualität

Helligkeitsgradienten, welche über das gesamte Bild verlaufen werden erfolgreich geglättet, ohne dabei lokalen Kontrast, wie den von Schrift zu Hintergrund zu verringern. Der adaptive Schwellwertfilter kann bei richtiger Parameterwahl die Schrift vom Hintergrund trennen. Somit kann ein Bild generiert werden, welches mit guter Qualität gedruckt werden kann, wie in Abbildung 2 zusehen ist. In dunklen Bereichen des Bildes kann es auch bei guter Parameterwahl zu vereinzelten schwarzen Pixeln im Hintergrund kommen.

4 ZUSAMMENFASSUNG

In diesem Paper wurden die verwendeten Algorithmen und die Entwicklungsstufen eines Programms zur Bildqualitätsverbesserung, das schlecht ausgeleuchtete Bilder in eine gut druckbare (hoher Kontrast zwischen Hinter- und Vordergrund) Form bringt, dargestellt. Weiterhin wurde die Laufzeit über die Optimierung des Programms auf ein Bruchteil, des anfänglichen Python-Prototypen gesenkt und über die plattform-übergreifende Natur des Programms durch die Verwendung von CMake berichtet. Die Funktion und Effektivität

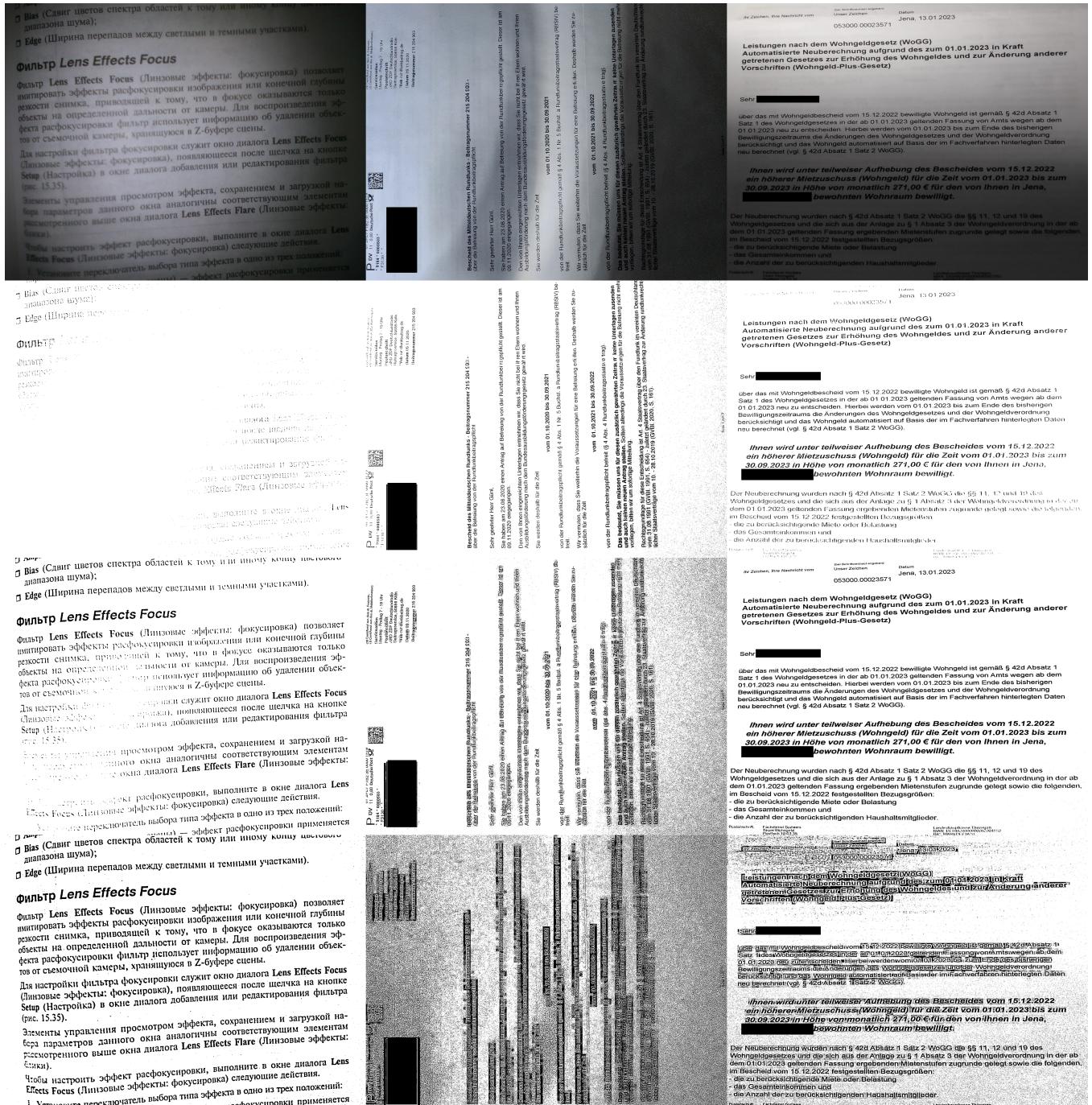


Figure 3: Vergleich zwischen den Bildqualitätsverbesserung von unterschiedlichen Testbildern. Von oben nach unten: Original Bild, Verbesserter Bilder mit den Parametern: $m = 0.3, n = 0.3$; $m = 0.45, n = 0.7$; $m = 0.5, n = 0.84$.

des Programms wurde anhand von verschiedene Testbilder und der einstellbaren Paramter des *Adaptiven Schwellenwerts* getestet und deren Einfluss auf die Qualität des Ergebnisses gezeigt.

REFERENCES

- [1] Joachim Denzler. 2021. Vorlesungsnotizen Rechnersehen I.
 - [2] Eric Günl. 2021. Qualitätssicherung von mikrostrukturierten Werkzeugeinsätzen durch vollflächigen Nachweis der optischen Funktionalität. Unveröffentlicht, unter NDA.