Technische Universität Dresden

Fakultät für Elektrotechnik und Informationstechnik

Institut für Biomedizinische Technik

Versuchsanleitung Praktikum Mikrorechentechnik II
Versuch GMM-3

Bildaufnahme und -verarbeitung mit OpenCV und Python

Stand: 03.05.2024

Ansprechpartner:

Matthieu Scherpf, Dipl.-Ing.
Institut für Biomedizinische Technik
Fakultät Elektrotechnik und Informationstechnik
TU Dresden
Fetscherstraße 29, 01307 Dresden
Email: Matthieu.Scherpf@tu-dresden.de
Tel.: +49 (0)351-463 32118

Inhaltsverzeichnis

1 Einführung

1.1 Ziel des Versuchs

In diesem Versuch sollen Methoden der digitalen Bildverarbeitung mit Python demonstriert werden. Das Ziel des Versuches ist es, Bilder von einer Webcam einzulesen, durch eigene Bildverarbeitungsalgorithmen zu verändern und auf dem Bildschirm darzustellen. Dabei wird mit der Verwendung externer Bibliotheken (OpenCV-Python) der Funktionsumfang von Python erweitert. Am Beispiel der Bildverarbeitung wird gezeigt, wie man komplexe Aufgaben durch Abstraktion mit wenig und gut lesbarem Quellcode bewältigen kann.

1.2 Methodik

Zur Programmierung während des Versuches wird Python verwendet. Sie werden die entsprechenden Anweisungen in einzelnen Befehlssequenzen umsetzen. Dabei werden Sie Ihre Befehlssequenzen in eine bereits existierende Main-Funktion eintragen. Diese übernimmt bereits die Bildübertragung von der Kamera in den Rechner. Ihre Befehle werden nach und nach die Main-Funktion ergänzen. Nach jeder Versuchsaufgabe erfolgt eine neue Testphase. Im Kolloquium werden auch die Hardwarekomponenten des Versuches und ihre Funktionen besprochen.

1.3 Coronabedingte Onlinedurchführung

Sollte der Praktikumsversuch sowie das Kolloquium nicht in Präsenz stattfinden können, dann ist der Versuch am eigenen Rechner zu bearbeiten. Hierfür wird davon ausgegangen, dass in der Praktikumsgruppe mindestens ein Rechner mit einer Webcam ausgestattet ist und entsprechend für die Bearbeitung der Aufgabe genutzt werden kann. Für die Bewertung ist der finale Quellcode einzureichen. Die Benotung findet dann auf Basis des digital durchgeführten Kolloquiums sowie der Qualität des eingereichten Quellcodes statt. Haben Sie diesbezüglich Fragen, können Sie sich per E-Mail (Matthieu.Scherpf@tu-dresden.de) an den Praktikumsbetreuer wenden.

Im folgenden finden Sie Informationen zur Installation von Python sowie des Quellcode-Grundgerüsts für die Versuchsdurchführung.

1.3.1 Installation von Python

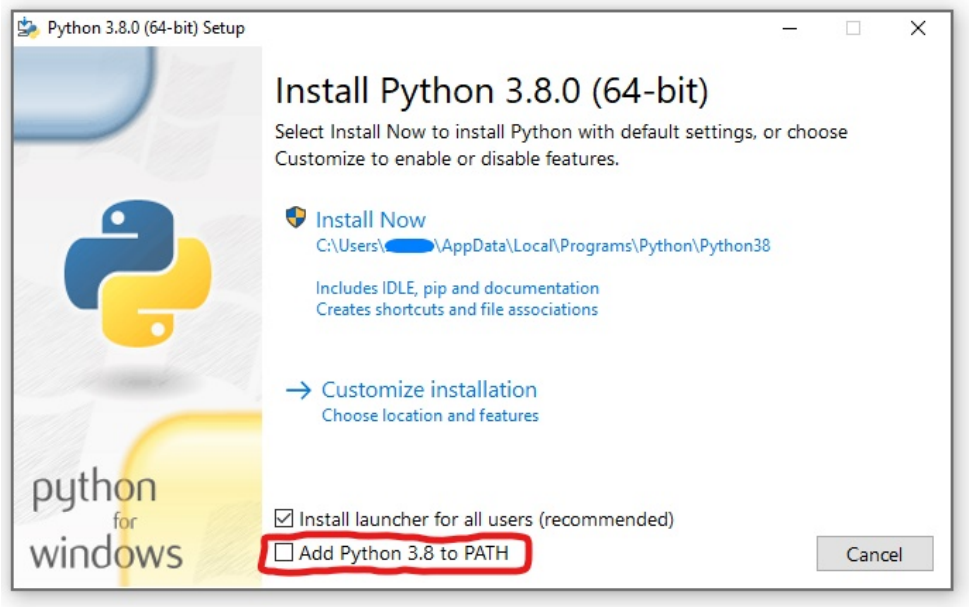

Abbildung 1: Bei der Installation muss das Häckchen gesetzt werden, um Python über das Terminal starten zu können.

Laden Sie sich die folgende Windows installer Datei für Python 3 herunter, welche Sie unter benanntem Link (https://www.python.org/ftp/python/3.8.6/python-3.8.6-amd64.exe) finden können. Führen Sie anschließend den heruntergeladenen installer aus um Python 3 für Windows zu installieren (nutzen

Sie für alle Optionen die Standardvorgaben des installers - **zusätzlich ist das Kästchen „Add Python 3.8 to PATH“ anzukreuzen, siehe Abbildung 1**). Testen Sie, ob die Installation erfolgreich war, indem Sie ein Terminal öffnen (unter Windows *cmd* oder *Eingabeaufforderung* genannt) und dort *python* eingeben. Startet jetzt der Python-Interpreter (siehe Abbildung 2), dann hat alles funktioniert. Weiterhin können Sie unter [4] Hilfe für die Installation zu finden.

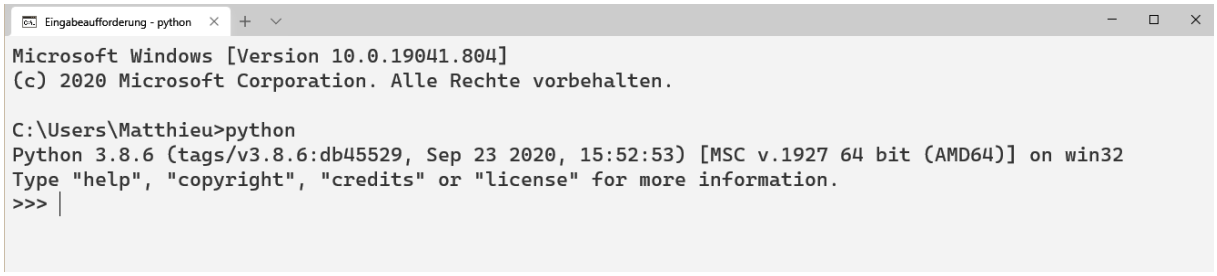

Abbildung 2: Terminal nach Eingabe des *python*-Befehls.

Sollten Sie ein anderes Betriebssystem verwenden (macOS, Linuxdistribution), dann sind die Schritte prinzipiell die gleichen. Es muss natürlich der installer für das entsprechende Betriebssystem heruntergeladen werden, anstatt des oben angegebenen.

Bevor Sie den Praktikumsbetreuer um Hilfe bitten, sollten Sie zuerst versuchen im Internet mögliche Lösungen zu finden bzw. Ihre Gruppenmitglieder um Hilfe zu bitten. In der Regel lassen sich auf diese Weise die meisten Probleme schnell beheben.

1.3.2 Installation des Pakets *mrt2pkgGMM3*

Nachdem Python erfolgreich installiert wurde, ist nun noch der Quellcode erforderlich, auf dem dieser Praktikumsversuch aufbauen soll. **Dieser kann als zip-Archiv unter benanntem Link (`https://cloudstore.zih.tu-dresden.de/index.php/s/N4GCQ6pbdLCMP6c`) heruntergeladen werden**. Das Archiv muss vor der Installation entpackt werden. Öffnen Sie anschließend ein neues Terminal und navigieren Sie mittels folgendem Befehl zum eben entpackten Ordner:

```
cd /path/to/folder
```

Dann nutzen Sie den vorherig automatisch mitinstallierten python-package-manager (kurz pip) um alle notwendigen Bibliotheken zu installieren:

```
python -m pip install -r requirements.txt
```

Hierzu zählt unter anderem *opencv-python*, welche zum Auslesen der Webcam sowie der weiteren Verarbeitung der Bilddaten benötigt wird.

1.3.3 Installation eines Editors

Um nun Pythoncode schreiben zu können, sollte noch ein entsprechender Editor heruntergeladen werden. Es wäre theoretisch auch möglich, hierfür einfach den Windows standard Texteditor zu verwenden, allerdings gilt es sprachspezifische Regeln einzuhalten, wofür ein Editor entsprechende Hilfen liefert (bspw. die korrekte Einrückung von Codezeilen). Als Editor empfiehlt sich *Visual Studio Code* (https://code.visualstudio.com/). Die Installation und Nutzung dieses Editors ist im Internet gut beschrieben und wird hier nicht weiter erläutert (siehe [6]).

1.4 Umgang mit dem Terminal

Der einfachste Weg Pythoncode auszuführen ist über das Terminal (auch Kommandozeile oder Eingabeaufforderung genannt). Um ein Pythonskript, also eine Datei mit der Endung *.py* auszuführen, starten Sie entweder ein Terminal beim Ordner, wo auch die auszuführende Datei liegt oder starten Sie ein Terminal und navigieren Sie dann mit dem Befehl aus 1.3.2 zum Zielordner. Anschließend können Sie das Pythonskript mit folgendem Befehl starten. Hilfe für die Nutzung des Terminals unter Windows können Sie unter [5] finden.

```
python skriptname.py
```

Eine eigenständige Kompilierung ist in Python nicht notwendig, da der Programmcode zur Laufzeit kompiliert wird. Das bedeutet, dass der Programmcode direkt zum Start automatisch in den entsprechenden Maschinencode übersetzt wird.

2 Inhalt

2.1 Exkurs in die digitale Bildverarbeitung

Die Bildverarbeitung ist ein Spezialgebiet der digitalen Signalverarbeitung. Sie beschäftigt sich mit der Aufnahme, Diskretisierung, Speicherung und Verarbeitung der Bilder. Digitale Bildverarbeitung ist allgegenwärtig, in der Industrieautomatisierung und Prozesssteuerung, ebenso wie in Verbrauchergeräten (Digitalkameras, Smartphones, Webcams). Digitale Bilder können auf vielerlei Art erzeugt werden. Im Folgenden wird als Bild stets die Abbildung von sichtbarem Licht gemeint. Digitale Bilder können jedoch auch aus anderen physikalischen Größen oder Benutzereingaben errechnet werden (Ultraschall, Magnetresonanztomographie, Radar o.ä.).

In Abbildung 3 ist der Ablauf der Aufnahme eines digitalen Videos, also einer Bildfolge, schematisch dargestellt. Die Einzelelemente des Bildsensors - die Pixel - quantisieren einfallendes Licht. Dadurch kann die Bildinformation eines Objektes diskretisiert werden. Die diskreten Werte können dann digital gespeichert werden.

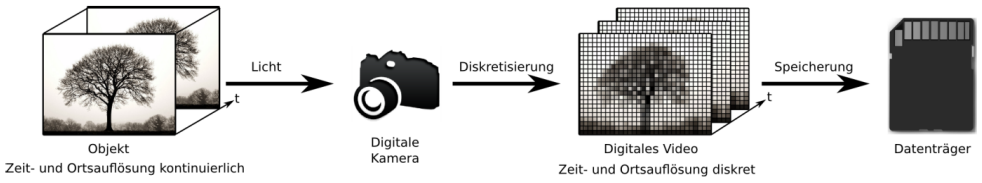

Abbildung 3: Ablauf der Aufnahme eines digitalen Bildes.

Im Falle eines graustufen Bildes werden Helligkeitswerte gespeichert, die bei der Wiedergabe auf einem Bildschirm der Helligkeit der einzelnen Bildpunkte entsprechen. Bei einem Farbbild wird das Licht gefiltert, bevor es den Sensor erreicht. So kann für jeden Bildpunkt das Mischungsverhältnis mehrerer Grundfarben bestimmt werden. Im Allgemeinen werden dazu rote, grüne und blaue Farbfilter verwendet. Im RGB-Farbraum werden damit für jeden Bildpunkt genau drei Helligkeiten gespeichert. Bei der Wiedergabe entsprechen diese der Helligkeit, mit der die einzelnen Subpixel eines Bildschirmes leuchten (siehe Abbildung 4). Durch den geringen Abstand zueinander können die einzelnen Subpixel vom menschlichen Auge nicht aufgelöst werden. Es entsteht ein Farbeindruck, welcher der additiven Mischung der Grundfarben entspricht.

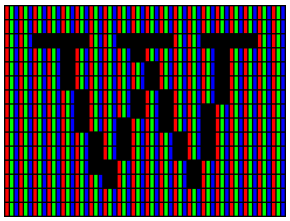

Abbildung 4: Diskretisierung von Bildern - vom Pixel zum Subpixel.

Die möglichen Verarbeitungsschritte, die über die Bildakquise, Speicherung und Darstellung hinausgehen sind sehr vielfältig. Geläufige Beispiele sind das Filtern (z.B. Weichzeichnen) oder Skalieren (Ändern der Größe) der Bilder. Komplexere Algorithmen sind notwendig um den Inhalt digitaler Bilder zu analysieren. Die Fortschritte in der Miniaturisierung der Rechentechnik erlauben es, anspruchsvolle Bildverarbeitungsaufgaben, wie das Erkennen von Gesichtern, auf günstiger Hardware in Echtzeit durchzuführen.

2.2 Einführung in OpenCV mit Python

Die *Open Source Computer Vision Library* ist eine Bibliothek von C-Funktionen für die Bildverarbeitung. Die durch *OpenCV* bereitsgestellten Funktionen vereinheitlichen den Zugriff auf Kamerahardware und

erleichtern den Umgang mit komplexen Datentypen wir mehrkanaligen Bildern und Videos. OpenCV ist hardwareunabhängig. Dadurch lässt sich die Bibliothek unter verschiedenen Prozessorarchitekturen und Betriebssystemen nutzen (Windows, Linux, macOS, Android, iOS). Weiterhin existieren Schnittstellen zu vielen bedeutenden Programmiersprachen wie C++, C#, Java und Python.

2.2.1 Auslesen der Kamera mit OpenCV

Listing 1: Aulesen der Kamera mit OpenCV in Python. Entnommen aus dem für dieses Praktikum vorgesehenen Paket *mrt2pkgGMM3*.

```
import cv2

if __name__ == '__main__':
    cam_stream = cv2.VideoCapture(0)
    cv2.namedWindow('Demo', cv2.WINDOW_AUTOSIZE)
    while True:
        check, frame = cam_stream.read()
        frame = cv2.medianBlur(frame, 11)
        cv2.imshow('Demo', frame)
        if cv2.waitKey(1) == 27:
            break
    cam_stream.release()
    cv2.destroyAllWindows()
```

Listing 1 zeigt ein kurzes, aber vollständiges Python Programm, das OpenCV verwendet. Das Verständnis dieses Programmteils ist Voraussetzung zum Lösen der Aufgaben. OpenCV wird ständig weiterentwickelt. Deswegen muss darauf geachtet werden, auf welche Version sich die im Internet angebotenen Dokumentation und Beispiele beziehen. Dieser Versuch baut auf Version 4.5 auf.

2.2.2 Hardwareinteraktion

Durch die große Menge an unterschiedlichen Verbraucher- und Industriekameras existiert eine nur schwer überschaubare Menge an Standards, Treibern und Softwareschnittstellen. OpenCV verbirgt die Kommunikation mit der Kamera vor dem Programmierer und bietet stattdessen ein einheitliches und einfach zu bedienendes Interface an. Damit können einzelne Bilder (Frames) sowohl von verschiedenen Kameras, als auch aus Videodateien gelesen werden. Dieses Interface präsentiert sich als Objekt vom Typ *cv2.VideoCapture()*. Nach erfolgreicher Initialisierung können durch Aufruf dessen Methode *read()* einzelne Bilder von der Kamera abgerufen werden.

2.2.3 Wie „sieht“ der Computer ein Bild?

Digitale Bilder können in einer Vielzahl von Formaten gespeichert werden. Im Rahmen dieses Versuches genügt es das Standardformat von OpenCV zu verstehen. Dabei handelt es sich um 24 Bit Farbbilder mit den drei Farbkanälen rot, grün und blau. Obwohl dieser Typ in der OpenCV Dokumentation als RGB-Format bezeichnet wird, liegen die Helligkeitswerte der einzelnen Kanäle in der Reihenfolge BGR im Arbeitsspeicher. Dies wird in Abbildung 5 veranschaulicht.

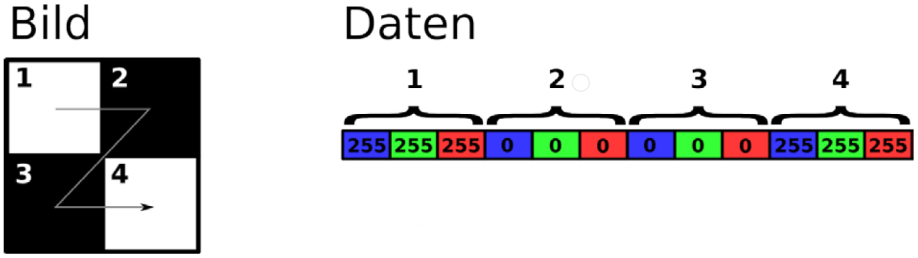

Abbildung 5: Ablage von RGB Bildern im Arbeitsspeicher bei Nutzung von OpenCV.

2.2.4 Bildverarbeitung mit OpenCV

OpenCV stellt viele Funktionen bereit, um Bilder zu analysieren und zu manipulieren. Diese können direkt mit den von der Kamera ausgelesenen Bildern (i.e. *array*) umgehen. Ein Beispiel dafür ist die Funktion *cv2.medianBlur()*, welche in Listing 1 genutzt wird. Sie erwartet ein Bild (vom Typ *numpy.ndarray*) als Quelle. Durch direkten Zugriff auf die Bilddaten können beliebige eigene Algorithmen verwirklicht werden. Sie können Hilfe für die Nutzung von OpenCV mit Python unter [3] finden.

2.2.5 Nutzung einer graphischen Oberfläche

OpenCV verfügt über Funktionen zum Erstellen einer graphischen Benutzeroberfläche (GUI), die als *High-level GUI* bezeichnet wird. Diese kann zur Darstellung von Bildern verwendet werden und erfasst Benutzereingaben über Maus und Tastatur. Um ein Fenster zu erstellen, genügt ein Aufruf der Funktion *cv2.namedWindow()*. Beim Erzeugen wird dem Fenster ein Name zugewiesen, durch den es im weiteren Programmverlauf identifiziert wird. Ein Bild kann dem Fenster durch die Funktion *cv2.imshow()* übergeben werden. Graphische Benutzeroberflächen werden im Allgemeinen durch Ereignisse (Events) gesteuert. Das bedeutet, dass ein Programmteil für die Ein- und Ausgabe dieser Ereignisse verantwortlich sein muss. In der High-level GUI übernimmt die Funktion *cv2.waitKey()* diese Rolle. Entgegen dem Namen liest diese Funktion nicht nur Tastatureingaben aus, sondern übernimmt die gesamte Ereignisverwaltung. Aus diesem Grund muss sie aufgerufen werden, wann immer sich der Inhalt eines Fensters ändern soll. Der Funktion kann eine Wartezeit (Timeout) übergeben werden. Diese bestimmt, wie lange auf eine Tastatureingabe gewartet wird, bevor das Programm weiterläuft.

2.3 Hardware

Je nachdem, ob das Praktikum als Präsenz- oder Onlineveranstaltung durchgeführt wird, kommt entweder die Webcam in Ihrem Computer oder speziell für das Praktikum vorgesehene Hardware zum Einsatz. Sollte das Praktikum als Onlineveranstaltung stattfinden, bedenken Sie, dass das Praktikum als Gruppenarbeit angesetzt ist. Sollten Sie also keine Webcam zur Verfügung haben oder ähnliche Probleme auftreten, sind diese in aller Regel im Rahmen der Gruppe lösbar. Für den Fall einer Durchführung im Rahmen einer Präsenzveranstaltung, steht Ihnen eine *Logitech StreamCam* (https://www.logitech.com/de-de/product/streamcam#specification-tabular) zur Verfügung. Die Kamera kann maximal mit einer Auflösung von 1920x1080 Pixeln bei 30 Bildern pro Sekunde ausgelesen werden.

3 Gute Praxis bei der Programmierung

Da nicht nur das Kolloquium, sondern auch das Erstellen bzw. Erweitern des Programms mit dem geforderten Funktionsumfang Teil der Bewertung ist, werden hier noch Tipps für die gute Praxis beim Programmieren gegeben. Sie sollten im Rahmen dieses Praktikums folgende Punkte beachten:

- Fügen Sie an den erforderlichen Stellen Kommentare ein, sodass Ihr vorgehen nachvollzogen werden kann.
- Versuchen Sie den Code möglichst lesbar (also einfach) zu halten.
- Achten Sie auf eine konsistente und verständliche Benennung von Variablen und ggf. Funktionen und Klassen (sollten für dieses kleine Projekt nicht notwendig sein).
- Achten Sie darauf, dass Ihr Code in der letzten Version, die Sie abgeben, funktioniert. Das heißt, dass alle drei Aufgabenteile (*aufgabe_x.py*) erfolgreich ausgeführt werden können und die entsprechenden Aufgabenteile umgesetzt sind.

4 Vorbereitungsaufgaben

Beantworten Sie die folgenden Fragen (jeder Student einzeln) in Stichpunkten schriftlich. Die Antworten bilden die Grundlage für das Kolloquium und sind Bestandteil des Protokolls. Ohne diese schriftlichen Vorbereitungen findet das Praktikum nicht statt!

- Welche Bildaufnahmesensoren wurden vor Einsatz der CCD und CMOS Sensoren verwendet? Nennen Sie einige davon!
- Wie funktioniert die Bildaufnahme und -ausgabe der Bilddaten auf einem CCD Sensor?

- Wie funktioniert die Bildaufnahme und -ausgabe der Bilddaten auf einem CMOS Sensor?
- Vergleichen Sie kurz beide Bauformen (CCD, CMOS) in Bezug auf ihre praktische Anwendbarkeit.
- Wie wird mit Bildaufnahmesensoren eine Farbaufnahme ermöglicht (Grundprinzipien)?
- Welche technischen Anwendungen der elektronischen Bildaufnahme sind Ihnen bekannt?
- Nennen Sie speziell Anwendungen von Kameras in der Medizin.
- Nennen Sie die wichtigsten Baugruppen einer WebCam und erläutern deren Funktion.
- Nennen Sie den Unterschied zwischen Helligkeitswerten und Graustufen.
- Welche Größe (in Byte) hat ein Farbbild der Auflösung 128x128 mindestens?
- Wie kann man mit einem Programm aus einem Farbbild ein graustufen Bild erzeugen?

5 Versuchsdurchführung

5.1 Aufgabenstellung

Im Verlauf des Praktikums sind folgende Aufgaben zu lösen. Lösen Sie die Aufgabe der Reihe nach und nutzen Sie die hierfür vorgesehenen Dateien/Skripte. Sofern möglich, können Sie den Code aus der vorherigen Aufgabe einfach in die neue Aufgabe übernehmen/kopieren. **Bitte protokollieren Sie die Lösung jeder Aufgabe mithilfe eines Screenshots ihres Bildschirms.**

Einführung: Führen Sie das Beispielprogramm *example.py* über das Terminal aus (siehe 1.4).

Aufgabe 1 (nutzen Sie hierfür die Datei *aufgabe_1.py*)

a. Verändern Sie das Programm so, dass ein zweites Fenster zur Anzeige geöffnet wird. In diesem sollen die modifizierten Bilder der nächsten Teilaufgabe dargestellt werden.

b. Stellen Sie im zweiten Fenster nur den grünen Farbkanal des originalen Bildes dar.

Aufgabe 2 (nutzen Sie hierfür die Datei *aufgabe_2.py*)

a. Verändern Sie das Programm so, dass ein zweites Fenster zur Anzeige geöffnet wird. Nutzen Sie hierfür die Klasse *CustomWindow* und übergeben Sie ihr regelmäßig das darzustellende Bild. Diese Klasse stellt die Funktionen *imshow* sowie *namedWindow*, welche sich genauso wie über OpenCV nutzen lassen, zur Verfügung.

b. Wenden Sie auf das in diesem Fenster dargestellte Bild den Schwellwertfilter aus *thresholdfilter.py* an indem Sie die Funktion *apply_threshold* auf das Bild anwenden.

c. Justieren Sie die gegebenen Schwellwerte im Fenster so, dass möglichst effektiv die Gesichtshaut (möglichst keine Haare oder Hintergrund) während der Aufnahme erkannt wird. Dokumentieren Sie die optimalen Schwellwerte.

Aufgabe 3 (nutzen Sie hierfür die Datei *aufgabe_3.py*): Bauen Sie auf dem letzten Stand von Aufgabe 2 auf. Instanziieren Sie ein Objekt der Klasse *PPGExtractor* bevor die Schleife ausgeführt wird. Übergeben Sie dieser Instanz regelmäßig das Originalbild nach Schwellwertanwendung. Nach ca. 10 Sekunden sollte das aus den Hautpixeln geschätzte Photoplethysmogramm der letzten 3 Sekunden angezeigt werden. Zusätzlich wird im Terminal in einem definierten Zeitintervall die geschätzte Herzrate ausgegeben. Die Bestimmung der Parameter ist hier sehr einfach umgesetzt und entsprechend Störanfällig (bspw. durch Bewegung oder Beleuchtungsschwankungen). Wie und warum das funktioniert, sollten Sie mit Ihrem Praktikumsbetreuer besprechen.

5.2 Hinweise zur Durchführung

Für die bewertete Versuchsdurchführung tragen Sie sich bitte in folgende Umfrageliste ein: https://dudpoll.inf.tu-dresden.de/vyzh3BOlMQ/. Melden Sie sich mit Ihrer Praktikumsgruppe für einen Termin an. In dieser bewerteten Versuchsdurchführung werden die Vorbereitungsaufgaben (siehe 4) sowie die Lösungen zu den jeweiligen Aufgaben aus 5.1 geprüft.

6 Literatur

[1] Klaus R., Käser H.: Grundlagen der Computertechnik,
Vdf Hochschulverlag AG an der ETH Zürich, 1998

[2] Stotz, Dieter: Computergestützte Audio- und
Videotechnik, Springer Verlag, Berlin Heidelberg New York, 1995

[4] How-to install python: `https://realpython.com/installing-python/`

[3] OpenCV Python tutorials: `https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_tutorials.html`

[5] How-to use windows command line with python: `https://docs.python.org/3/faq/windows.html#how-do-i-run-a-python-program-under-windows`

[6] How-to use python with visual studio code: `https://code.visualstudio.com/docs/python/python-tutorial`

7 Arbeits- und Brandschutzhinweise

Vorbeugende Maßnahmen:

- Die Praktikumsteilnehmer haben sich so zu verhalten, daß Gefahrensituationen und Unfälle vermieden werden.
- Die Befugnis zum Bedienen und Nutzen von Geräten ist auf den zugewiesenen Praktikumsplatz beschränkt.
- Eingriffe in die zum Praktikumsaufbau gehörenden Geräte sind nicht erlaubt.
- Der Anschluss und der Betrieb privater Geräte in den Praktikumsräumen ist verboten.
- Defekte an Geräten oder Gebäudeeinrichtungen sind unverzüglich dem Betreuer mitzuteilen. Betroffene Geräte sind außer Betrieb zu nehmen. Andere Personen sind vor Gefahren zu warnen. Den Anweisungen der Praktikumsbetreuer bzw. anderer aufsichtsführender Personen ist unbedingt Folge zu leisten.
- Betriebsfremde dürfen sich nur mit Erlaubnis des Praktikumsbetreuers in den Praktikumsräumen aufhalten.
- Rauchen und Umgang mit offenem Feuer ist nicht gestattet.
- Nach Ende des Praktikums ist der Arbeitsplatz in sauberem und aufgeräumten Zustand zu hinterlassen.
- Außergewöhnliche Ereignisse bzw. besondere Vorkommnisse sind umgehend dem Betreuer oder dem diensthabenden Assistenten zu melden.

8 Quellcode

example.py > ...

```python
"""
---------------------------------------------------------------------------
Created: 08.02.2021, 13:04
---------------------------------------------------------------------------
Author: Matthieu Scherpf
Email: Matthieu.Scherpf@tu-dresden.de
Website: https://becuriouss.github.io/matthieuscherpf/
---------------------------------------------------------------------------
Purpose: Für Praktikum Mikrorechentechnik 2 - Versuch GMM 3 - Bildaufnahme und
-verarbeitung mit OpenCV und Python
---------------------------------------------------------------------------
"""

# ----------------------------------------------------------------------------
# Import benötigter python Pakete bzw. Module
# ----------------------------------------------------------------------------
import cv2  # OpenCV Paket für Python

# wenn dieses Skript über den Pythoninterpreter im Terminal gestartet wird,
dann wird der folgende Code ausgeführt
if __name__ == '__main__':
    # ------------------------------------------------------------------------
    # Dem Konstruktor muss die ID (ein integer) der Kamera übergeben werden.
    Wenn nur eine Kamera am System angeschlossen ist, ist die ID <0>.
    Eventuell muss dieser Wert auf <1> gesetzt werden, sofern das System bspw.
    Front- und Rückkamera besitzt -> ausprobieren.
    # ------------------------------------------------------------------------
    cam_stream = cv2.VideoCapture(0)

    # ------------------------------------------------------------------------
    # OPTIONAL (zum experimentieren): Parameter können gesetzt werden (mit
    welcher Auflösung soll die Kamera ausgelesen werden, mit wie vielen
    Bildern pro Sekunde, etc.); Die Parameter werden nicht zwingend von der
    verwendeten Kamera unterstützt und es kann sein, dass die Kamera nicht
    mehr korrekt ausgelesen werden kann!
    # ------------------------------------------------------------------------
    # cam_stream.set(cv2.CAP_PROP_FRAME_WIDTH, 1920)  # defaults to: 640
    # cam_stream.set(cv2.CAP_PROP_FRAME_HEIGHT, 1080)  # defaults to: 480
    # cam_stream.set(cv2.CAP_PROP_FPS, 30)  # defaults to: 30

    # ------------------------------------------------------------------------
    # Initialisiere ein Fenster, zur anzeige der Bilder; Das Fenster passt
    sich automatisch der Größe bzw. der Auflösung der Aufnahme an; Das Fenster
    wird später über den ersten Übergabeparameter <'Demo'> referenziert
    # ------------------------------------------------------------------------
    cv2.namedWindow('Demo', cv2.WINDOW_AUTOSIZE)

    # ------------------------------------------------------------------------
    # Lese die Bilder der Kamera aus, bis die "ESC" Taste betätigt wird (Das
    Fenster muss hierbei angewählt (im Fokus) sein.)
    # ------------------------------------------------------------------------
    while True:
        # --------------------------------------------------------------------
        # lese das nächste verfügbare frame der Kamera ein; Es werden zwei
        Werte zurückgegeben: boolean check (True, wenn das frame erfolgreich
        ausgelesen wurde, was hier nicht weiter geprüft wird; False sonst),
        frame (ein array, was die Werte des ausgelesenen frames enthält)
        # --------------------------------------------------------------------
        check, frame = cam_stream.read()

        # --------------------------------------------------------------------
        # --------------------------------------------------------------------
        # Verarbeitung des eingelesenen Bildes
        # --------------------------------------------------------------------
        # --------------------------------------------------------------------
        # Anwendung eines Medianfilters mit einer Filtergröße von 11x11 Pixeln
        frame = cv2.medianBlur(frame, 11)

        # --------------------------------------------------------------------
        # Anzeigen des manipulierten frames
        # --------------------------------------------------------------------
        cv2.imshow('Demo', frame)

        # --------------------------------------------------------------------
        # Prüfen, ob eine Nutzereingabe getätigt wurde um das Programm zu
        beenden; 27 entspricht der Escape-Taste (siehe ASCII Tabelle: http://
        www.asciitable.com/)
        # --------------------------------------------------------------------
        if cv2.waitKey(1) == 27:
            break

    # ------------------------------------------------------------------------
    # Wenn das Programm via Escape-Taste beendet wurde muss die Kamera
    hierrüber informiert werden und noch geöffnete Fenster sauber geschlossen
    werden
    # ------------------------------------------------------------------------
    cam_stream.release()
    cv2.destroyAllWindows()

```

aufgabe_1.py > ...

```python
"""
-------------------------------------------------------------------------
Created: 08.02.2021, 13:04
-------------------------------------------------------------------------
Author: Matthieu Scherpf
Email: Matthieu.Scherpf@tu-dresden.de
Website: https://becuriouss.github.io/matthieuscherpf/
-------------------------------------------------------------------------
Purpose: Für Praktikum Mikrorechentechnik 2 - Versuch GMM 3 - Bildaufnahme und
-verarbeitung mit OpenCV und Python
-------------------------------------------------------------------------
"""

# -----------------------------------------------------------------------
# Import benötigter python Pakete bzw. Module
# -----------------------------------------------------------------------
import cv2  # OpenCV Paket für Python
from thresholdfilter import apply_threshold

# wenn dieses Skript über den Pythoninterpreter im Terminal gestartet wird,
dann wird der folgende Code ausgeführt
if __name__ == '__main__':
    # -------------------------------------------------------------------
    # Dem Konstruktor muss die ID (ein integer) der Kamera übergeben werden.
    Wenn nur eine Kamera am System angeschlossen ist, ist die ID <0>.
    Eventuell muss dieser Wert auf <1> gesetzt werden, sofern das System bspw.
    Front- und Rückkamera besitzt -> ausprobieren.
    # -------------------------------------------------------------------
    cam_stream = cv2.VideoCapture(0)

    # -------------------------------------------------------------------
    # OPTIONAL (zum experimentieren): Parameter können gesetzt werden (mit
    welcher Auflösung soll die Kamera ausgelesen werden, mit wie vielen
    Bildern pro Sekunde, etc.); Die Parameter werden nicht zwingend von der
    verwendeten Kamera unterstützt und es kann sein, dass die Kamera nicht
    mehr korrekt ausgelesen werden kann!
    # -------------------------------------------------------------------
    # cam_stream.set(cv2.CAP_PROP_FRAME_WIDTH, 1920)  # defaults to: 640
    # cam_stream.set(cv2.CAP_PROP_FRAME_HEIGHT, 1080)  # defaults to: 480
    # cam_stream.set(cv2.CAP_PROP_FPS, 30)  # defaults to: 30

    # -------------------------------------------------------------------
    # Initialisiere ein Fenster, zur anzeige der Bilder; Das Fenster passt
    sich automatisch der Größe bzw. der Auflösung der Aufnahme an; Das Fenster
    wird später über den ersten Übergabeparameter <'Demo'> referenziert
    # -------------------------------------------------------------------
    cv2.namedWindow('Demo', cv2.WINDOW_AUTOSIZE)

    # -------------------------------------------------------------------
    # Lese die Bilder der Kamera aus, bis die "ESC" Taste betätigt wird (Das
    Fenster muss hierbei angewählt (im Fokus) sein.)
    # -------------------------------------------------------------------
    while True:
        # ---------------------------------------------------------------
        # lese das nächste verfügbare frame der Kamera ein; Es werden zwei
        Werte zurückgegeben: boolean check (True, wenn das frame erfolgreich
        ausgelesen wurde, was hier nicht weiter geprüft wird; False sonst),
        frame (ein array, was die Werte des ausgelesenen frames enthält)
        # ---------------------------------------------------------------
        check, frame = cam_stream.read()

        # ---------------------------------------------------------------
        # ---------------------------------------------------------------
        # Verarbeitung des eingelesenen Bildes
        # ---------------------------------------------------------------
        # ---------------------------------------------------------------

        # ---------------------------------------------------------------
        # Anzeigen des manipulierten frames
        # ---------------------------------------------------------------
        cv2.imshow('Demo', frame)

        # ---------------------------------------------------------------
        # Prüfen, ob eine Nutzereingabe getätigt wurde um das Programm zu
        beenden; 27 entspricht der Escape-Taste (siehe ASCII Tabelle: http://
        www.asciitable.com/)
        # ---------------------------------------------------------------
        if cv2.waitKey(1) == 27:
            break

    # -------------------------------------------------------------------
    # Wenn das Programm via Escape-Taste beendet wurde muss die Kamera
    hierrüber informiert werden und noch geöffnete Fenster sauber geschlossen
    werden
    # -------------------------------------------------------------------
    cam_stream.release()
    cv2.destroyAllWindows()

```

aufgabe_2.py > ...

```python
"""
-------------------------------------------------------------------------
Created: 08.02.2021, 13:04
-------------------------------------------------------------------------
Author: Matthieu Scherpf
Email: Matthieu.Scherpf@tu-dresden.de
Website: https://becuriouss.github.io/matthieuscherpf/
-------------------------------------------------------------------------
Purpose: Für Praktikum Mikrorechentechnik 2 - Versuch GMM 3 - Bildaufnahme und -verarbeitung mit OpenCV und Python
-------------------------------------------------------------------------
"""

# -------------------------------------------------------------------------
# Import benötigter python Pakete bzw. Module
# -------------------------------------------------------------------------
import cv2  # OpenCV Paket für Python
from thresholdfilter import apply_threshold
from customWindow import CustomWindow as cw

# wenn dieses Skript über den Pythoninterpreter im Terminal gestartet wird, dann wird der folgende Code ausgeführt
if __name__ == '__main__':
    # -------------------------------------------------------------------------
    # Dem Konstruktor muss die ID (ein integer) der Kamera übergeben werden. Wenn nur eine Kamera am System angeschlossen ist, ist die ID <0>. Eventuell muss dieser Wert auf <1> gesetzt werden, sofern das System bspw. Front- und Rückkamera besitzt -> ausprobieren.
    # -------------------------------------------------------------------------
    cam_stream = cv2.VideoCapture(0)

    # -------------------------------------------------------------------------
    # OPTIONAL (zum experimentieren): Parameter können gesetzt werden (mit welcher Auflösung soll die Kamera ausgelesen werden, mit wie vielen Bildern pro Sekunde, etc.); Die Parameter werden nicht zwingend von der verwendeten Kamera unterstützt und es kann sein, dass die Kamera nicht mehr korrekt ausgelesen werden kann!
    # -------------------------------------------------------------------------
    # cam_stream.set(cv2.CAP_PROP_FRAME_WIDTH, 1920)  # defaults to: 640
    # cam_stream.set(cv2.CAP_PROP_FRAME_HEIGHT, 1080)  # defaults to: 480
    # cam_stream.set(cv2.CAP_PROP_FPS, 30)  # defaults to: 30

    # -------------------------------------------------------------------------
    # Initialisiere ein Fenster, zur anzeige der Bilder; Das Fenster passt sich automatisch der Größe bzw. der Auflösung der Aufnahme an; Das Fenster wird später über den ersten Übergabeparameter <'Demo'> referenziert
    # -------------------------------------------------------------------------
    cv2.namedWindow('Demo', cv2.WINDOW_AUTOSIZE)

    # -------------------------------------------------------------------------
    # Lese die Bilder der Kamera aus, bis die "ESC" Taste betätigt wird (Das Fenster muss hierbei angewählt (im Fokus) sein.)
    # -------------------------------------------------------------------------
    while True:
        # -------------------------------------------------------------------------
        # lese das nächste verfügbare frame der Kamera ein; Es werden zwei Werte zurückgegeben: boolean check (True, wenn das frame erfolgreich ausgelesen wurde, was hier nicht weiter geprüft wird; False sonst), frame (ein array, was die Werte des ausgelesenen frames enthält)
        # -------------------------------------------------------------------------
        check, frame = cam_stream.read()

        # -------------------------------------------------------------------------
        # -------------------------------------------------------------------------
        # Verarbeitung des eingelesenen Bildes
        # -------------------------------------------------------------------------
        # -------------------------------------------------------------------------

        # -------------------------------------------------------------------------
        # Anzeigen des manipulierten frames
        # -------------------------------------------------------------------------
        cv2.imshow('Demo', frame)

        # -------------------------------------------------------------------------
        # Prüfen, ob eine Nutzereingabe getätigt wurde um das Programm zu beenden; 27 entspricht der Escape-Taste (siehe ASCII Tabelle: http://www.asciitable.com/)
        # -------------------------------------------------------------------------
        if cv2.waitKey(1) == 27:
            break

    # -------------------------------------------------------------------------
    # Wenn das Programm via Escape-Taste beendet wurde muss die Kamera hierrüber informiert werden und noch geöffnete Fenster sauber geschlossen werden
    # -------------------------------------------------------------------------
    cam_stream.release()
    cv2.destroyAllWindows()

```

aufgabe_3.py > ...

```python
"""
-------------------------------------------------------------------------
Created: 08.02.2021, 13:04
-------------------------------------------------------------------------
Author: Matthieu Scherpf
Email: Matthieu.Scherpf@tu-dresden.de
Website: https://becuriouss.github.io/matthieuscherpf/
-------------------------------------------------------------------------
Purpose: Für Praktikum Mikrorechentechnik 2 - Versuch GMM 3 - Bildaufnahme und -verarbeitung mit OpenCV und Python
-------------------------------------------------------------------------
"""

# ----------------------------------------------------------------------------
# Import benötigter python Pakete bzw. Module
# ----------------------------------------------------------------------------
import cv2  # OpenCV Paket für Python
from thresholdfilter import apply_threshold
from customWindow import CustomWindow as cw
from ppgextractor import PPGExtractor

# wenn dieses Skript über den Pythoninterpreter im Terminal gestartet wird, dann wird der folgende Code ausgeführt
if __name__ == '__main__':
    # ----------------------------------------------------------------------------
    # Dem Konstruktor muss die ID (ein integer) der Kamera übergeben werden. Wenn nur eine Kamera am System angeschlossen ist, ist die ID <0>. Eventuell muss dieser Wert auf <1> gesetzt werden, sofern das System bspw. Front- und Rückkamera besitzt -> ausprobieren.
    # ----------------------------------------------------------------------------
    cam_stream = cv2.VideoCapture(0)

    # ----------------------------------------------------------------------------
    # OPTIONAL (zum experimentieren): Parameter können gesetzt werden (mit welcher Auflösung soll die Kamera ausgelesen werden, mit wie vielen Bildern pro Sekunde, etc.); Die Parameter werden nicht zwingend von der verwendeten Kamera unterstützt und es kann sein, dass die Kamera nicht mehr korrekt ausgelesen werden kann!
    # ----------------------------------------------------------------------------
    # cam_stream.set(cv2.CAP_PROP_FRAME_WIDTH, 1920)  # defaults to: 640
    # cam_stream.set(cv2.CAP_PROP_FRAME_HEIGHT, 1080)  # defaults to: 480
    # cam_stream.set(cv2.CAP_PROP_FPS, 30)  # defaults to: 30

    # ----------------------------------------------------------------------------
    # Initialisiere ein Fenster, zur anzeige der Bilder; Das Fenster passt sich automatisch der Größe bzw. der Auflösung der Aufnahme an; Das Fenster wird später über den ersten Übergabeparameter <'Demo'> referenziert
    # ----------------------------------------------------------------------------
    cv2.namedWindow('Demo', cv2.WINDOW_AUTOSIZE)

    # ----------------------------------------------------------------------------
    # Lese die Bilder der Kamera aus, bis die "ESC" Taste betätigt wird (Das Fenster muss hierbei angewählt (im Fokus) sein.)
    # ----------------------------------------------------------------------------
    while True:
        # ----------------------------------------------------------------------------
        # lese das nächste verfügbare frame der Kamera ein; Es werden zwei Werte zurückgegeben: boolean check (True, wenn das frame erfolgreich ausgelesen wurde, was hier nicht weiter geprüft wird; False sonst), frame (ein array, was die Werte des ausgelesenen frames enthält)
        # ----------------------------------------------------------------------------
        check, frame = cam_stream.read()

        # ----------------------------------------------------------------------------
        # ----------------------------------------------------------------------------
        # Verarbeitung des eingelesenen Bildes
        # ----------------------------------------------------------------------------
        # ----------------------------------------------------------------------------

        # ----------------------------------------------------------------------------
        # Anzeigen des manipulierten frames
        # ----------------------------------------------------------------------------
        cv2.imshow('Demo', frame)

        # ----------------------------------------------------------------------------
        # Prüfen, ob eine Nutzereingabe getätigt wurde um das Programm zu beenden; 27 entspricht der Escape-Taste (siehe ASCII Tabelle: http://www.asciitable.com/)
        # ----------------------------------------------------------------------------
        if cv2.waitKey(1) == 27:
            break

    # ----------------------------------------------------------------------------
    # Wenn das Programm via Escape-Taste beendet wurde muss die Kamera hierrüber informiert werden und noch geöffnete Fenster sauber geschlossen werden
    # ----------------------------------------------------------------------------
    cam_stream.release()
    cv2.destroyAllWindows()

```