# Anforderungsanalyse

Projekt: FDFR – Face Detection Face Recognition

Inhalt:

## 1. Zielsetzung

FDFR soll, gegeben ein Bild, die darauf befindlichen Personen erkennen und identifizieren.
Die Priorität liegt, neben einer schnellen Bearbeitung, in erster Linie auf einer möglichst zuverlässigen Identifikation auch unter erschwerten Bedingungen wie etwa ungünstige Lichtverhältnisse oder teilweise verdeckten Gesichtern.

Die Software soll eine Person anhand eines Bildes identifizieren können, sofern die besagte Person vorher in einer dafür vorgesehenen Datenbank erfasst wurde.
Hierfür wird ein schwarz-weiß-Bild (farbige Bilder werden konvertiert) übergeben. Anschließend werden mithilfe verschiedener Prozesse alle Gesichter auf diesem Bild lokalisiert. Wird kein Gesicht lokalisiert, wird ein oder werden mehrere Bild-Bearbeitungs-Algorithmen die Bild-Qualität so manipulieren, dass der nachfolgende Algorithmus zur Lokalisierung eine höhere Erfolgschance hat. Diese Vor-Bearbeitung der Bilder kann auch grundlegend auf alle eingehenden Bilder angewandt werden, sollte aber je nach Komplexität und damit einhergehender Laufzeitbeeinträchtigung nur sparsam verwendet werden. Tests zur Gesamtlaufzeit des Systems unter verschiedenen Architekturen können die finale Vorgehensweise dabei erleichtern.
Bei mindestens einem lokalisiertem Gesicht, wird dieses/ werden diese einzeln aus dem Gesamt-Bild extrahiert und getrennt voneinander weiterverarbeitet. Die Schritte hierbei sind für jedes Gesicht gleich:

a) Erfassung von 'Landmarken' auf dem Gesicht
b) Rotation, Translation und Zentrierung des Gesichts (Frontalisierung)
c) Skalierung auf einheitliche Größe (bspw. 250x250 px)
d) Setzen aller Nicht-Gesichts-Pixel auf schwarz

Die so erhaltene einheitliche Matrix kann in einen eindimensionalen Spalten-Vektor (bzw. 1x3 dimensional, da 24bit also 3-Channel Grauwerte) überführt und anschließend in ein hierfür ausgelegtes Convolutional-Neural-Network (CNN) übergeben werden. Dieses CNN transformiert den Input in einen ebenfalls eindimensionalen Output-Vektor, welcher als Merkmals-Vektor des Bildes interpretiert werden kann.
Mithilfe diese Merkmal-Vektors ist es möglich einen Vergleich zu anderen Merkmals-Vektoren aufzustellen und somit ein bereits gespeichertes Gesicht wiederzuerkennen. Für diesen Vergleich wird der Einfachheit wegen vorerst ein reiner Distanz-Abgleich mithilfe der Euklidischen Distanz durchgeführt und ein Treshhold bestimmt dabei die maximale Distanz, innerhalb welcher von der gleichen Person ausgegangen werden kann.

## 2. Funktionale Anforderungen

Es wird eine möglichst robuste, aber auch möglichst schnelle Wiedererkennung von vorher gespeicherten Personen auf einem Bild angestrebt.
Das Projekt wird in der Programmiersprache PYTHON (3.6.2) realisiert unter Verwendung bereits umgesetzter Frameworks und Modulen. Zu den wichtigsten zählen dabei OpenCV (3.3.0+contrib), Tensorflow (1.2.0), Theano (0.9.0), Keras (2.0.5), dlib (19.6.1) und natürlich grundlegende wie NumPy (1.13.1) oder SciPy (0.19.1).
Die oben genannten Architektur kann sich im Laufe der Entwicklung ändern. Grundsätzlich besteht die Software aber aus folgenden Elementen:

- Bild-Bearbeitung (A)
- Gesichts-Lokalisierung (B)

- Gesichts-Erkennung (C)

Die genannten Komponenten müssen dabei nicht zwingend in dieser Reihenfolge eingebettet werden. Vielmehr kommen Bild-Bearbeitungs-Algorithmen (A) sowohl anfänglich zur Verbessrung der Erkennungsrate zum Einsatz, wie auch nach der Erkennung, um beispielsweise ein Gesicht zu Frontalisieren (ausrichtung des Gesichtes nach vorne), zu zentrieren und zu skalieren.
Die Gesichts-Lokalisierung (B) wird unumgänglich immer vor der Gesichts-Erkennung (C) stattfinden wobei es nicht auszuschließen ist, dass Funktionen von B auch während des Prozesses C noch von Nutzen sind.
Faktisch gibt es noch weitere, in dieser Aufzählung vernachlässigbare Module die der Vorverarbeitung dienen, beispielsweise um Daten zu konvertieren, zu labeln, Module zu testen oder ähnliches.

## 2.1 Bild-Bearbeitung (Modul A)

Das Bild-Bearbeitungs-Modul umfasst Algorithmen zur Manipulation von Bild-Daten, wodurch nachfolgende Algorithmen in ihrer Funktion unterstützt werden sollen.

### 2.1.1 Qualitätsverbesserung

Bilder mit schlechten Kontrasten oder Lichtverhältnissen erschweren oft die Erkennung von darauf befindlichen Objekten. Mithilfe verschiedener Verfahren kann die Qualität deutlich verbessert werden.
Durch einen Histogrammausgleich wird der Kontrast erhöht und die Erkennungsrate drastisch verbesert. Es gibt verschiedene Arten, wobei hier zunächst die entsprechende Funktion von OpenCV verwendet wird (Einfacher Histogrammausgleich). Ein adaptiver Histogrammausgleich lieferte bei mehreren Test schlechtere Ergebnisse.
Dieses Verfahren ist bisher ausreichend Effektiv, wobei das Hinzufügen weiterer Verfahren möglicherweise für andere Daten oder Umstände effektiver ist.
In Betracht gezogen werden können unter Anderem noch:
Gammakorrektur, Histrogrammspreizung, Medianfilter

### 2.1.2 Frontalisierung

Sobald ein Gesicht erkannt wurde, muss es in ein einheitliches Format überführt werden.
Die Frontalisierung ist eigentlich ein mehrstufiger Prozess.
Zuerst wird der Ausschnitt mit dem Gesicht aus dem Original-Bild geschnitten.
Anschließend wird das Gesicht vermessen (z.B. Landmarken) und frontalisiert.
Das frontalisierte Gesicht kann nun skaliert und zentriert werden, wobei dieser Schritt auch vor der Frontalisierung passieren kann. Die Nicht-Gesichts-Pixel werden einheitlich mit schwarz belegt, um Rauschen zu minimieren.

## 2.2 Gesichtsidentifikation (Modul B)

Das Modul B soll auf gegebenem Bild Gesichter identifizieren bzw. lokalisieren. Hierzu wird momentan eine Funktion von OpenCV verwendet, die sich auf die Benutzung von bereits vortrainierten Haarcascades stützt.
Wird hiermit kein Gesicht erkannt, sollten entweder komplexere Algorithmen zur Qualitätsverbesserung des Bildes angewandt werden und/ oder komplexere Algorithmen zur Gesichtidentifikation.

### 2.3 Gesichtserkennung (MODUL C)

Die Gesichtserkennung soll ein lokalisiertes Gesicht mit bereits bekannten Gesichtern vergleichen und eine verlässliche Aussage darüber treffen können, ob und wem dieses Gesicht zugeordnet werden kann. Umgesetzt wird die, indem das zuvor lokalisierte und extrahierte Gesicht durch ein CNN in einen Merkmalsvektor überführt wird. Zunächst wird hierfür ein vortraineirtes CNN genutzt (VGG 16). Der resultierende Merkmalsvektor ist dabei eine reduzierte, aber möglichst eindeutige Repräsentation des ursprünglichen Inputs. Personen in der Datenbank werden mit dem gleichen Verfahren abgebildet, wobei eine Person mehrere Merkmalsvektoren besitzen kann, um verschiedene Bedingungen (Licht, Pose, ..) abzudecken. Der so erhaltene Vektor kann also mit den schon existierenden Verglichen werden. Anfangs wird das Gesicht dabei der Person zugeordnet, deren gespeicherter Vektor die geringste euklidische Distanz zu dem neuen Vektor hat und zudem einen manuell festgelegten Grenzwert dabei nicht überschreitet.

## 3. Nicht-funktionale Anforderungen

Das gesamte Projekt sollte mit maximal möglicher Flexibilität geplant und umgesetzt werden. Ziel ist es schnell einen stehenden Workflow bereitzustellen, der alle funktionalen Anforderungen minimal befriedigt. Die modulare Aufbauweise soll es dabei vereinfachen zukünftige Änderungen unkompliziert in das bestehende System zu integrieren und über eine zentrale Stelle die gewünschten Architektur-Optionen zu bestimmen. So soll einerseits ein Wechsel zwischen mehreren Algorithmen mit dem gleichen Zweck sowie das erstellen von Erweiterungen leicht umsetzbar sein.

### 3.1 Modularität

Das System wird in mehrere Module unterteilt, welche wiederum aus mehreren Modulen bestehen können. Eine zentrale Schnittstelle soll alle verwendeten Module steuern und verwalten. Alle nötigen Einstellungen werden in dieser zentralen Schnittstelle vorgenommen und während der Ausführung von hier an weitere Module übertragen. Die zentrale Schnittstelle legt dabei die grundlegende Architektur bzw. den grundlegenden Ablauf vor. Die Reihenfolge der Modul-Aufrufe wird hier bestimmt, ebenso wie die Art, wie einzelne Module das übergebene Problem lösen.
Jedes Modul soll so aufgebaut sein, dass wenigen öffentliche Funktionen ein Problem übergeben werden, zusammen mit Parametern zur Art der Lösung. Diese öffentliche Funktion ruft daraufhin private Funktionen auf, um das Problem zu lösen wobei die privaten Aufrufe bestimmt sind durch die mitgegebenen Parameter. Abschließend wird das gelöste Problem zurückgegeben.

### 3.2 Verfügbarkeit

Das beschrieben System soll durch einen expliziten Aufruf verfügbar sein und nicht vor beenden seiner Aufgabe abbrechen. Mögliche Fehler sollen sichtbar gemeldet werden, aber sofern möglich den Programmfluss nicht stören. Schwerwiegende Fehler erfordern möglicherweise eine Nutzer-Interaktion zur Festlegung des weiteren Vorgehens oder im schlimmsten Fall den Neustart des Programms.
Besondere Auflagen wie ständige Verfügbarkeit sind nicht erforderlich.

## 3.3 Änderbarkeit

Die Software soll mit geringem Aufwand verändert werden können. Hierfür wird eine hohe Modularität gewählt und besondere Rücksicht bei der Programmierung genommen. Änderungen sollten ausschließlich durch Erweiterungen, nicht aber durch direkte Änderungen in bestehenden Funktionen erfolgen.

## 3.4 Performanz

Die Performanz des Systems spielt eine wichtige Rolle, wenngleich sie lediglich an zweiter Stelle, nach der Änderbarkeit, steht. Nach einem flexiblen System, ist ein System, dass Anfragen in Echtzeit bearbeiten kann, oberstes Ziel der Implementierung. Alle Algorithmen sollen mit größtmöglicher Effizienz umgesetzt werden und auf handelsüblichen Maschinen funktionieren.

## 3.5 Sicherheit

Es gelten keine besonderen Sicherheitsansprüche.

## 3.6 Testbarkeit

Die einzelnen Module müssen für sich abgeschlossen und testbar sein. Regelmäßige Tests sollen eine reibungslose Zusammenführung ermöglichen. Dabei wird zwischen Unit-Tests für einzelne Funktionen und Integration-Tests für ganze Module unterschieden. Beide Tests werden angewendet, wobei nicht jede, aber jede komplexere Funktion getestet werden soll.

## 3.7 Bedienbarkeit

Die Software soll von Fachkräften mit entsprechendem Vorwissen möglichst intuitiv bedienbar sein. Der Schwerpunkt liegt allerdings klar auf dem funktionellen Aspekt und es wird demnach kein gesonderter Aufwand betrieben die Nutzerfreundlichkeit zu optimieren.