KI-Praxis VI

Teil 6

Gesichtserkennung und -identifizierung

Ein Anwendungsbereich der Künstlichen Intelligenz (KI), der viele Menschen schon immer besonders fasziniert hat, ist die Fähigkeit von Robotern und Computern, Personen bzw. menschliche Gesichter zu erkennen und dann mit den betreffenden Personen zu interagieren. In diesem Artikel soll mit vergleichsweise einfacher Hardware wie einem Raspberry Pi, einer Pi-Camera, einer speziellen 200°-Weitwinkelkamera oder einer USB-Webcam ein Gesichts- bzw. Personenerkennungssystem aufgebaut werden. Dabei muss zwischen zwei verschiedenen Varianten unterschieden werden. Die Gesichtserkennung erkennt, dass sich menschliche Gesichter im Blickfeld einer Kamera befinden. Diese können dann entsprechend automatisch gekennzeichnet werden. Eine wesentlich anspruchsvollere Aufgabe ist die Wiedererkennung bzw. Identifizierung von Gesichtern oder Personen. Hierbei steht die Identifikation aufgrund individueller Gesichtsmerkmale im Vordergrund. Speziell das letztere Verfahren birgt erhebliches gesellschaftliches und soziales Gefährdungspotenzial. Auch darauf wird am Ende des Beitrags eingegangen.

Gesichter erkennen und identifizieren

Wenn ein modernes Smartphone via FaceID durch einen einfachen Blick in die Frontkamera entsperrt wird, ist bereits Künstliche Intelligenz im Einsatz. Lässt man Google oder Apple die Urlaubsfotos sortieren, basieren die Ergebnisse bereits wieder auf Machine-Learning-Algorithmen. Ob daher beispielsweise die letztgenannte Anwendung wirklich eine gute Idee ist, muss jeder für sich selbst entscheiden. Auch bei PCs und Laptops kann man sich bereits via Webcam mit dem eigenen Gesicht anmelden. Inwieweit die dabei gesammelten biometrischen Daten wirklich „sicher" sind, sei dahingestellt.

Die ersten Ansätze zur Entwicklung von Gesichtserkennungssystemen stammen bereits aus den 1960er-Jahren. Die Gesichtserkennung ist seitdem

eine der wichtigsten Anwendungen der Künstlichen Intelligenz. Aufgrund ihrer hohen wirtschaftlichen Bedeutung und vielfältiger Anwendungen wurde sie in den letzten 50 Jahren intensiv untersucht.

Häufig werden die beiden Begriffe Gesichtserkennung (engl. „face detection") und Gesichtsidentifizierung (engl. „face recognition") synonym verwendet. Es gibt jedoch einige grundlegende Unterschiede. Die Definition der Gesichtserkennung bezieht sich auf ein System oder einen Algorithmus, mit dem das Vorhandensein von Gesichtern auf Bildern oder in Videos festgestellt werden kann. Hierfür werden heute typischerweise die Methoden des Maschinellen Lernens (ML) eingesetzt. Diese Bilder dürfen dabei neben den Gesichtern auch Landschaften, Gebäude und andere Teile des Menschen wie etwa Beine, Schultern und Arme etc. enthalten.

Die Restriktionen aus den Anfängen der Gesichtserkennung, bei denen die Gesichter nur vor gleichmäßigen Hintergründen erkannt wurden, sind inzwischen hinfällig. Eine der ersten weit verbreiteten Anwendungen der Gesichtserkennung waren Autofokussysteme in elektronischen Kameras. Hier war es belanglos, zu welcher Person das Gesicht gehörte, es musste nur sicher erkannt werden. Weitere Anwendungen sind etwa die Ermittlung der Anzahl von Personen in einem bestimmten Bereich. Neben sicherheitsrelevanten Aufgaben rückten hier erstmals auch Marketingperspektiven ins Blickfeld.

Die Gesichtsidentifizierung ist dagegen in der Lage, die Identität einer bestimmten Person festzustellen. Ursprünglich war die Zugangskontrolle zu sensiblen Bereichen eine der wichtigsten Anwendungen dieser Variante. Inzwischen wird das Verfahren aber auch bei alltäglichen Aufgaben wie etwa dem Entsperren von Laptops, Handys oder Tablets eingesetzt.

Viele Gesichtserkennungsalgorithmen beginnen mit der Suche nach menschlichen Augen. Diese bilden eine sogenannte Talregion und sind so eines der am einfachsten zu erkennenden Merkmale. Sobald die Augen erkannt wurden, versucht der Algorithmus typischerweise andere Gesichtsbereiche wie Augenbrauen, Mund oder Nase zu erfassen. Sobald die erste Vermutung, dass ein Gesicht erkannt wurde, bestätigt ist, können weitere Tests zur Anwendung kommen. Diese überprüfen dann, ob tatsächlich ein menschliches Antlitz erkannt wurde oder ob es sich um eine zufällige Verteilung von Merkmalen handelt, die einem menschlichen Gesicht ähneln. Die Gesichtsidentifizierung ist damit letztendlich ein biometrisches Verfahren, das weit über das Erfassen eines menschlichen Gesichts im Sichtfeld einer Kamera hinausgeht.

Die Gesichtserkennung erreicht zwar meist keine hundertprozentige Präzision. Dafür kann jedoch sehr exakt angegeben werden, wie groß die Wahrscheinlichkeit ist, dass ein Gesicht mit einer bestimmten Person in einer Datenbank übereinstimmt (Bild 1).

Mit der Entwicklung von Bilderkennungsverfahren, die in der Lage sind, Gesichter zu analysieren, tauchten schnell auch Fragen zur Sicherheit dieser Technik auf. Letztendlich kann man diese Gefahren nur mit umfangreichem eigenem Wissen und umfassendem Sachverstand korrekt einschätzen.

Datenschutz und Persönlichkeitsrechte

Da die bei einem reinen Gesichtserkennungssystem erfassten Bilder nicht mit einer Datenbank abgeglichen werden, müssen auch keine persönlichen biometrischen Daten gespeichert werden. Ohne Datenspeicherung sind Datenschutzverletzungen der betroffenen Personen weniger wahrscheinlich. Dennoch ermöglichen es Erkennungssysteme etwa Geschäftsinhabern, ihren Kunden ein besseres Kundenerlebnis zu bieten. So kann das System allgemeine Informationen über das Gesicht wie Alter oder Geschlecht identifizieren, ohne spezifisch zu bestimmen, wer die Person ist. Wenn festgestellt wird, dass die überwiegende Mehrheit der Kunden in einer Filiale aus einer bestimmten demografischen Gruppe stammt, können Produktauswahl und Marketing entsprechend optimiert werden.

Bild 1: Gesichtserkennung, Identifizierung und Augmented Reality

Weitere Anwendungen finden sich in Bereichen, in denen gewisse Schutzbedürfnisse oberste Priorität haben. Sicherheitsrelevante Umgebungen, wie etwa in militärischen Einrichtungen, sind hier klassische Beispiele. Dort genügt es oft bereits, die Anwesenheit unerwünschter Personen zu erkennen, um geeignete Abwehrmaßnahmen zu ergreifen. Eine spezifische Erkennung ist nicht unbedingt erforderlich.

Hinweis:
Um auf die Belange des Datenschutzes eindringlich hinzuweisen, wurden alle Personenabbildungen in diesem Artikel verfremdet. Das Training und Testen der beschriebenen Systeme muss aber natürlich mit unverfremdetem Bildmaterial durchgeführt werden.

Grundlagen und Anwendungen

Nach den ersten Ansätzen der maschinellen Gesichtserkennung dauerte es über 30 Jahre, bis mit einem Algorithmus namens „Eigengesichter" die ersten wirklich nützlichen Ergebnisse erzielt wurden. Seither gewinnt das Thema jedoch immer mehr an Aufmerksamkeit und man kann von einer glänzenden Zukunft dieses Forschungsbereichs ausgehen. Zweifellos werden Sicherheitsanwendungen in vielen Bereichen eine zentrale Rolle spielen. Hierbei kann sowohl die Gesichtserkennung als auch die -identifizierung zum Einsatz kommen.

Ein interessantes Beispiel sind Flughafensicherheitssysteme. Die Gesichtsidentifizierung wird verwendet, um die Passagierkontrolle zu automatisieren. So können polizeilich gesuchte Personen oder potenziell Verdächtige identifiziert werden, bevor sie eine Straftat, wie etwa eine Flugzeugentführung, ausführen können.

Die reine Gesichtserkennung kann dagegen genutzt werden, um Passagierzahlen zu erfassen. Damit können die Ströme der Reisenden optimal gelenkt und gesteuert werden. Größere Personenansammlungen oder lange Warteschlangen werden automatisch und schnell erfasst, um etwa zusätzliche Schalter oder Abfertigungseinrichtungen öffnen zu können.

Im kommerziellen Einsatz kann die Gesichtserkennung auch verwendet werden, um die Identifizierung von Personen zu beschleunigen. So sind Systeme vorstellbar, die Kunden erkennen, sobald sie eine Bank- oder Versicherungsfiliale betreten. Ein Mitarbeiter kann den Kunden dann bereits mit seinem Namen begrüßen und Daten vorbereiten, bevor er tatsächlich zum Schalter oder Büro des Angestellten gelangt.

Aktive Werbetafeln könnten ihren Inhalt an die vorbeikommenden Personen anpassen. Nach der Analyse der Personen würden sich Werbespots an Geschlecht, Alter oder gar den persönlichen Stil anpassen.

Hierbei können jedoch durchaus bereits die ersten Konflikte mit Datenschutzgesetzen auftreten. Private Unternehmen haben im Allgemeinen nicht das Recht, Personen an öffentlichen Orten zu fotografieren oder zu filmen.

Mit dem Einsatz von 3D-Kameras kann die Technik nochmals wesentlich verbessert werden. Diese Systeme erzielen dank ihrer Fähigkeit, dreidimensionale Bilder eines Gesichts aufzunehmen eine nochmals deutlich verbesserte Treffergenauigkeit. Einfache Kamerasysteme können bereits mit Fotografien getäuscht werden. In der praktischen Umsetzung mit der Pi-Camera (s. Materialliste) ist dies leicht überprüfbar. Dieses System kann kaum zwischen einem Foto und der Person selbst unterscheiden.

Die Gesichtserkennung wird in Zukunft zweifellos immer häufiger eingesetzt. Die in den letzten Jahren generierten Datenmengen ermöglichen immer neue Wege zur Analyse der erfassten Information. Das Maschinelle Lernen wird somit einerseits Wege finden, um diese Informationen sinnvoll zu nutzen. Andererseits sollte man auch niemals die Gefahren dieser Technologien vergessen. Es ist daher immer nützlich und sinnvoll, wenn sich möglichst viele Leute auch praktisch mit den Methoden auseinandersetzen. Nur so kann verhindert werden, dass einige wenige die neue Macht der KI missbrauchen.

Methoden und Algorithmen

Für die Gesichtserkennung können zunächst die klassischen Methoden des Maschinellen Lernens und der Objektdetektion eingesetzt werden (Bild 2). Letztendlich ist auch ein menschliches Gesicht nichts anderes als ein Objekt mit speziellen Eigenschaften. Anstelle der im Beitrag „Maschinelles Handschriftenlesen“ [1] verwendeten Zahlenbilder könnte man so etwa eine große Anzahl von Gesichtern als Trainingsbasis verwenden. Das Internet mit seinen Anwendungen wie Youtube oder Facebook liefert hier nahezu unerschöpfliche Datenquellen. Die Daten

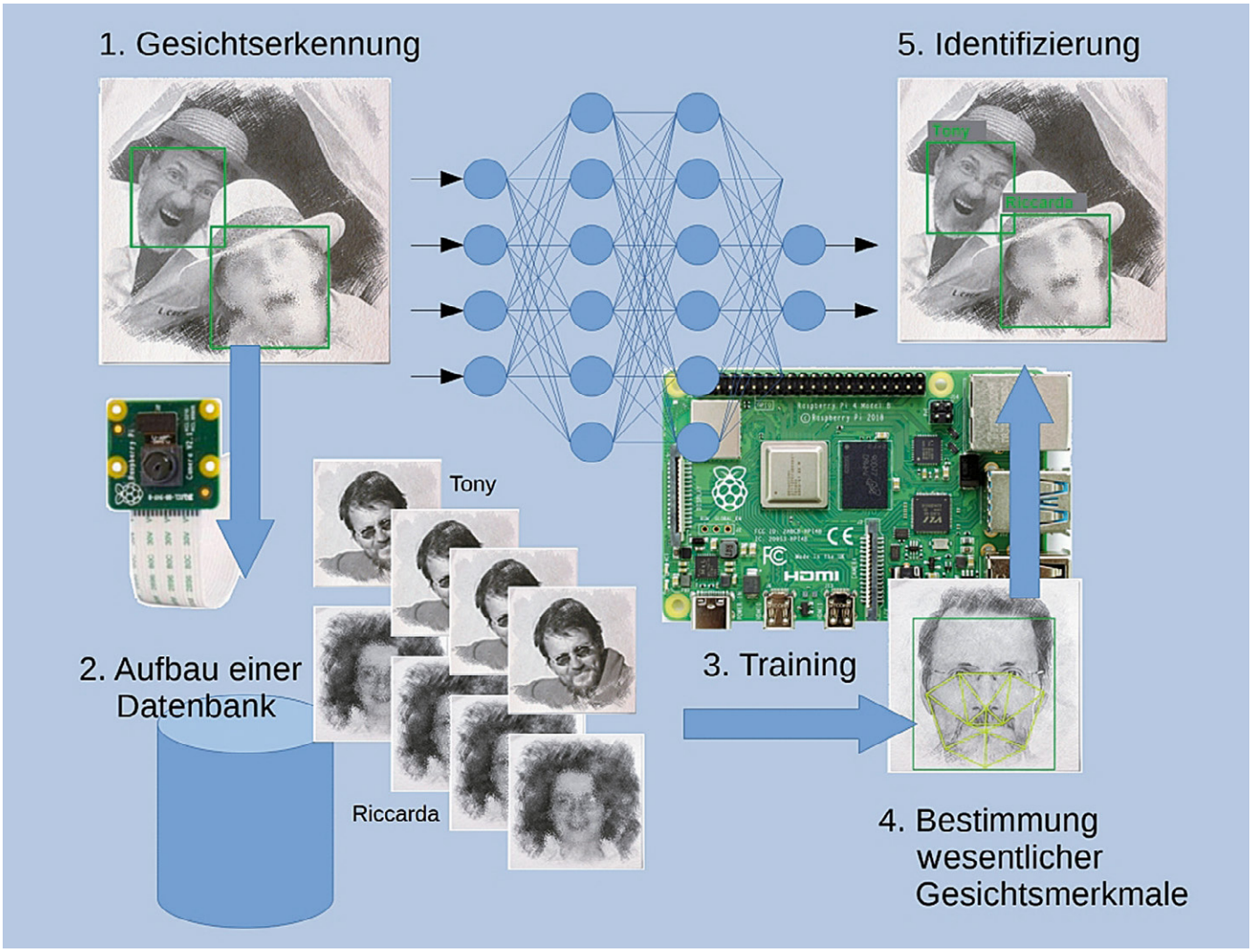

Bild 2: Prinzipielles Verfahren der Gesichtserkennung und -identifizierung

werden von den bekannten Internetkonzernen auch intensiv genutzt, da viele Anwender de facto unwissentlich ihre Zustimmung zur Verwendung ihrer persönlichen Daten gegeben haben.

Die Fortschritte bei der Gesichtserkennung sind also zu einem nicht unwesentlichen Teil dem Boom der sozialen Medien zu verdanken. Die dort angesammelten riesigen Datenmengen erhalten eine Unzahl von Gesichtsbildern, oftmals sogar zusammen mit einer großen Anzahl an mehr oder weniger persönlichen Zusatzinformationen.

Um zu verstehen, wie ein Algorithmus Gesichter erkennt, kann man sich zunächst fragen, wie Menschen eigentlich ein Gesicht erkennen. Auf den meisten Frontalbildern von menschlichen Gesichtern finden sich zwei Augen, eine Nase, Lippen, Stirn, Kinn, Ohren und Haare. Diese prinzipiellen Bestandteile sind praktisch immer vorhanden. Gesichter unterscheiden sich jedoch auch voneinander. Wo genau liegen also die Unterschiede?

Eine zusätzliche Problematik ergibt sich daraus, dass sich das Gesicht ein und derselben Person durch Emotionen (Lachen, Ärger ...) stetig verändert. Zudem haben das Alter, Kosmetik, Haarschnitt oder Bartwuchs einen oft nicht unwesentlichen Einfluss. Darüber hinaus erzeugt bereits die Änderung des Betrachtungswinkels häufig bereits ein deutlich anderes Gesichtsbild.

Bei genauerer Betrachtung zeigt sich jedoch, dass es einige Merkmale in jedem Gesicht gibt, die von Alter, Emotion und Orientierung etc. weitgehend unabhängig sind (Bild 3). Einige Forschungsgruppen begannen daher mit den Methoden des unbeaufsichtigten Lernens, um zunächst verschiedene Arten von Gesichtern zu klassifizieren und zu identifizieren. Andere verfolgten den Ansatz mit umfangreichem Training von Modellen, basierend auf jedem einzelnen Merkmal des Gesichts. Mit den klassischen Methoden der Datenanalyse wie

- Hauptkomponentenanalyse (PCA)
- lineare Diskriminanzanalyse
- unabhängige Komponentenanalyse
- Gabor-Filter

konnten dann erste Erfolge erzielt werden. Insbesondere Gabor-Filter erlangten besondere Bedeutung, da sie wichtige Merkmale wie Augen, Nase oder Mund in einem Bild lokalisieren können. Sie sind daher sowohl für die Gesichtserkennung als auch für die Identifikation einsetzbar.

Die klassische Methode, Gesichtsbilder Pixel für Pixel zu vergleichen, ist dagegen wenig effektiv. Zum Beispiel würden Hintergrund- und Haarpixel kaum auswertbare Informationen liefern. Zudem müssten für den direkten Vergleich alle Gesichter in allen Bildern perfekt ausgerichtet sein, um überhaupt brauchbare Ergebnisse zu erzielen.

Um dieses Problem zu lösen, erstellt der PCA-Algorithmus eine Reihe von Hauptkomponenten, die als „Eigengesichter" bezeichnet werden. Aus mathematischer Sicht sind Eigengesichter mit Eigenvektoren vergleichbar. Genauso wie jeder Vektor aus einer Summe von Basisvektoren erzeugt werden kann, könnte mit einem ausreichend großen Datensatz theoretisch jedes menschliche Gesicht durch eine spezielle Kombination von Eigengesichtern dargestellt werden. Der Eigengesichtsalgorithmus bildet damit die Grundlage der Gesichtserkennung.

Andere Analysemethoden bauen auf der so definierten Basis auf. So werden Gabor-Filter verwendet, um die wichtigsten Merkmale im Gesicht zu identifizieren. Anschließend kann ein Eigengesichtsalgorithmus verwendet werden, um diese Merkmale zu vergleichen. Aktuell stehen mehrere Open-Source-Bibliotheken zur Verfügung, in denen eine oder mehrere dieser Methoden implementiert wurden.

Gesichtserkennung in der Praxis

Als praktisches Anwendungsbeispiel soll ein Raspberry Pi (s. Materialliste) so eingerichtet und trainiert werden, dass damit Mitglieder der Familie oder des Freundeskreises maschinell erkannt werden.

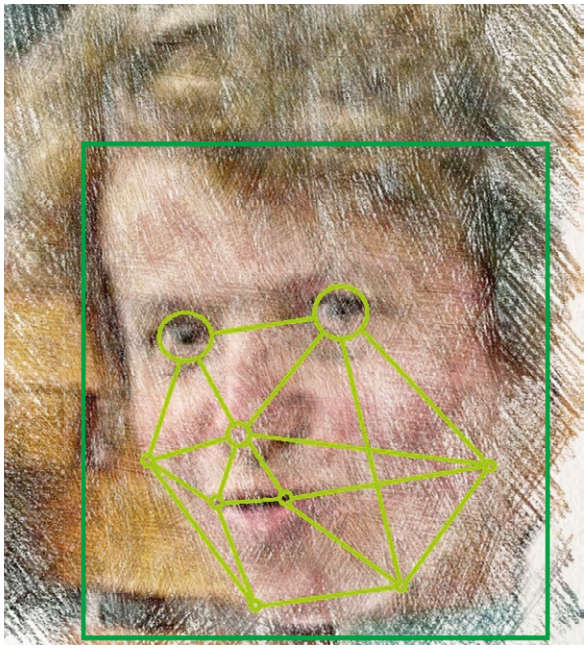

Bild 3: Wesentliche Gesichtsmerkmale

Schließlich kann mit einem selbst gebauten „Biometriescanner" sogar ein Türschloss realisiert werden, das nur bekannten Gesichtern die Tür öffnet.

OpenCV ist eine der beliebtesten Bibliotheken im Bereich der Objekt- und Gesichtserkennung. Der Kern der Bibliothek wurde ursprünglich in C/C++ geschrieben. Dies ist bei KI- bzw. ML-Anwendungen nicht ungewöhnlich, da C immer noch zu den schnellsten und effizientesten Programmiersprachen gehört. Später wurden dann die zugehörigen Python-Bibliotheken erstellt, die ein einfaches Einbinden in diese bei KI-Entwicklungen vorherrschende Sprache erlaubt.

Das Programmpaket verwendet Algorithmen für Maschinelles Lernen, um nach Gesichtern in einem Bild zu suchen. Da Gesichter sehr komplex aufgebaut sind und erheblichen individuellen Variationen unterliegen, gibt es keinen einfachen Test, der via „if-Entscheidung" ein Gesicht in einem beliebigen Bild finden könnte. Stattdessen gibt es eine Vielzahl von Mustern und Funktionen, die aufeinander abgestimmt sind. Die Algorithmen unterteilen das Problem, ein Gesicht zu identifizieren, in mehrere Teilaufgaben, von denen jede für einen Maschinenalgorithmus leicht zu lösen ist. Diese zugehörigen Algorithmen werden auch als „Klassifikatoren" bezeichnet.

Für eine effektive Gesichtserkennung können mehrere Tausend Klassifikatoren erforderlich sein. Ein typischer Gesichtserkennungsalgorithmus beginnt z. B. oben links in einem Bild und bewegt einen Erfassungsrahmen über kleine Bildelemente hinweg nach unten. In jedem einzelnen Bildblock wird dabei ständig überprüft, ob es sich bei dem aktuell erfassen Muster um ein Gesicht handeln könnte. Da Tausende Tests pro Block erforderlich sein können, müssen Millionen von Berechnungen durchgeführt werden. So erklären sich auch die erheblichen Rechenleistungen, die für Bilderkennungsverfahren erforderlich sind. Im Gehirn werden diese Infor-

mationen weitgehend parallel verarbeitet, sodass Millionen von Neuronen in der Summe die gleiche Rechenleistung erbringen wie eine einzelne, im Gigahertzbereich getaktete CPU.

Für eine möglichst effiziente Bilderkennung verwendet OpenCV sogenannte „Kaskaden". Wie bei einer Reihe von Wasserfällen wird das Problem so in mehrere Stufen unterteilt. Für jeden Bildblock wird ein grober, aber schneller Test durchgeführt. Liefert eine dieser Prüfungen ein positives Ergebnis, wird eine etwas detailliertere Untersuchung durchgeführt und so weiter. Der Algorithmus kann 30 bis 50 dieser Stufen oder Kaskaden nacheinander ausführen. Nur wenn alle Stufen mit positiven Resultaten durchlaufen wurden, wird die Erkennung eines Gesichts bestätigt.

Ein wesentlicher Vorteil dieses Verfahrens ist, dass der Großteil eines Bildes in den ersten Phasen nur negative Ergebnisse liefert. Dies bedeutet, dass keine Zeit damit verschwendet wird, in jedem Block sämtliche Funktionen zu testen. Damit kann der Algorithmus wesentlich beschleunigt werden. Gesichtserkennungsverfahren können so auch auf weniger leistungsfähigen Systemen nahezu in Echtzeit durchgeführt werden.

Da die Gesichtserkennung eine der wichtigsten Anwendungen des Maschinellen Lernens ist, verfügt OpenCV über eine Reihe integrierter Kaskaden, mit denen neben Gesichtern auch Augen, Hände oder Beine usw. erkannt werden. Zudem ist es sogar möglich, die Stimmungslage einer Person zu deuten, indem beispielsweise die Form des Mundes analysiert wird. Auf diesen Prinzipien basieren auch die bekannten Verfahren in Digitalkameras, die den Verschluss automatisch auslösen, sobald das Lächeln einer Person in Bildbereich detektiert wurde.

Raspberry Pi als Spion

Dank der hohen Effizienz der OpenCV-Algorithmen kann sogar mit der relativ bescheidenen Rechenleistung eines Raspberry Pi eine brauchbare Gesichtserkennung durchgeführt werden. Hierfür werden die folgenden Software-Pakete benötigt:

- OpenCV: Open-Source-Softwarebibliothek zur Verarbeitung von Bildern und Videos in Echtzeit. Das Paket enthält auch einige Algorithmen zum maschinellen Lernen.

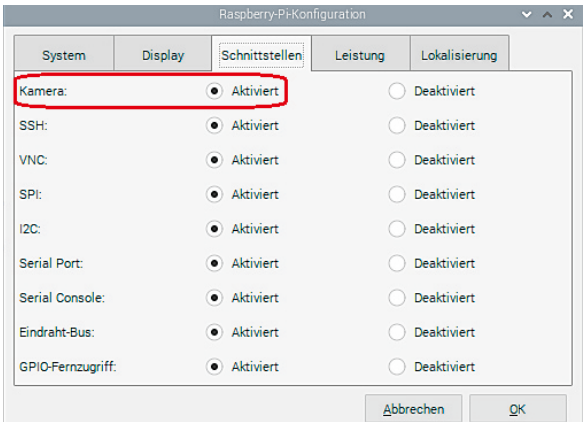

Bild 4: Aktivieren der Kamera

- face_recognition: Hiermit können Gesichtserkennungen durchgeführt und Referenzrahmen um Gesichter erstellt werden.
- Imutils: Dieses Paket enthält eine Reihe praktischer Funktionen zur Beschleunigung von OpenCV-Algorithmen auf dem Raspberry Pi.

Die einzelnen Befehle und Anweisungen zur Installation sind in der Datei „openCV_install.txt" zusammengefasst [2]. So können sie direkt in das Terminalfenster des Raspberry Pi kopiert werden. Das fehlerträchtige Abtippen kann damit entfallen.

Nach der Installation dieser Bibliotheken kann der Raspberry Pi dann anhand einer Reihe von Personenbildern trainiert werden. Die Bilder werden automatisch zu einem Trainingsdatensatz zusammengestellt. Danach kann der Raspberry Pi in einem Live-Video-Strom nicht nur die ihm nun bekannten Personen als Gesichter erkennen, sondern diese auch identifizieren.

Eine mögliche Anwendung wäre hier beispielsweise ein Erkennungssystem für das Tragen von Gesichtsmasken. Das System könnte nach entsprechendem Training feststellen, ob eine Person eine Gesichtsmaske trägt oder nicht. Falls die Maske fehlt, kann die betreffende Person, etwa durch eine SMS, zum Tragen einer Maske aufgefordert werden. Hier erkennt man bereits die vielfältigen Möglichkeiten, die derartige System eröffnen. Jedem sollte deshalb bewusst sein, dass hier neben den legalen auch durchaus illegale oder zumindest unethische Anwendungen möglich sind. An dieser Stelle ist daher die folgende Warnung angebracht.

Hinweis: Alle in diesem Beitrag vorgestellten Geräte, Verfahren und Methoden sind ausschließlich für den persönlichen und privaten Gebrauch bestimmt. Die beteiligten Personen sind darüber zu informieren, dass Daten gesammelt werden. Alle geltenden nationalen, staatlichen und kommunalen Gesetze sind vollumfänglich einzuhalten. Zudem ist beispielsweise zu bedenken, dass bereits Videoaufnahmen in den öffentlichen Bereichen meist illegal sind.

Hardware-Voraussetzungen und Installation

Für die Umsetzung der hier vorgestellten Projekte sind die folgenden Hardwarekomponenten erforderlich:

- Raspberry Pi 4–8 GB RAM mit Netzteil und microSD-Karte mit mindestens 32 GB
- Pi-Camera, 200°-Weitwinkelkamera (s. Materialliste) oder USB-Webcam

Falls der Raspberry Pi im Stand-alone-Betrieb arbeiten soll, werden zusätzlich Tastatur, Maus, ein Monitor und ein passendes HDMI-Kabel benötigt.

Die Installation und Einrichtung der für die Gesichtserkennung erforderlichen Pakete und Bibliotheken erfordert fortgeschrittene Kenntnisse und Erfahrungen im Umgang mit dem Raspberry Pi. Als Zeitaufwand sollten selbst erfahrene Nutzer mindestens zwei Stunden einplanen. Die genaue Dauer hängt unter anderem von der Download-Geschwindigkeit des verfügbaren Internetanschlusses ab.

Da sich die Bibliotheksversionen in diesem aktuell sehr aktiven Gebiet rasch ändern, ist hier ein gewisses Maß an Experimentierfreude notwendig, da es durchaus vorkommen kann, dass eine neuere Programmversion nicht mit den hier vorgestellten Varianten kompatibel ist. Als grundlegende Betriebssystemvariante für den Raspberry Pi kommt Raspberry Pi OS 10 zum Einsatz. Sobald es zu Verfügung steht, kann mit der Aktivierung der Pi-Camera begonnen werden. Wie immer sollte zunächst

```
sudo apt-get update & sudo apt-get upgrade
```

auf der Kommandozeile ausgeführt werden, bevor ein größeres neues Projekt gestartet wird. Dann beginnt die eigentliche Installation:

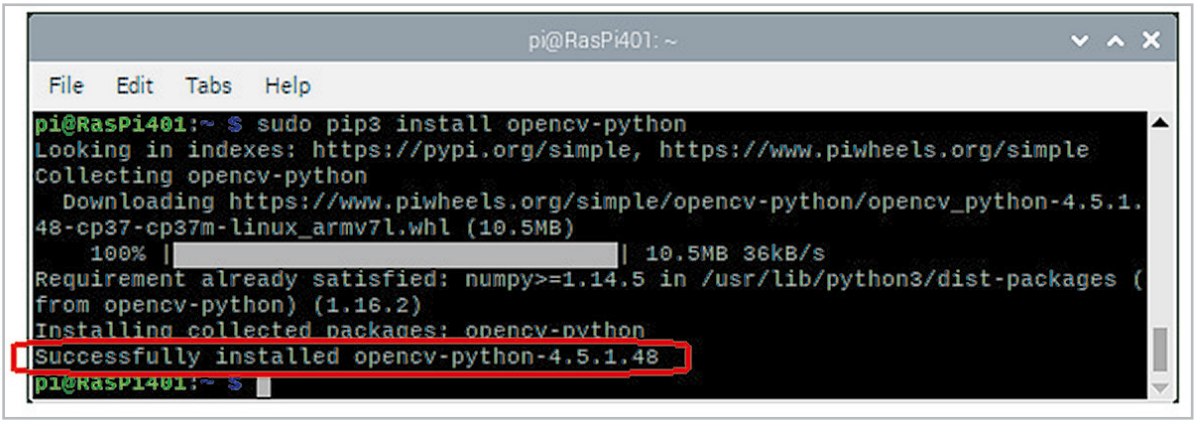

Bild 5: Erfolgreiche Installation von opencv-python

1. Aktivieren von „Kamera" in der Raspberry Pi-Konfiguration (Bild 4).
   Nach Drücken von OK sollte der Raspberry Pi neu gestartet werden.
2. Installieren der Pi-Camera über das Terminal mit diesem Befehl:
   pip install picamera [Array]

Mit der Anweisung

raspistill -o image.jpg

kann die Kamera getestet werden.

Nach der Ausführung des Befehls findet sich das aufgenommene Bild im Home-Verzeichnis (/home/pi).

Mit dem Programm

CamCheck_1V0.py

(s. [2]) kann die Videofunktion der Kamera mit einem Python-Programm überprüft werden.

Nach dem Start des Programms öffnen sich zwei aktive Videofenster. Eines zeigt die Farb-, das andere eine Schwarz-Weiß-Version des Videostroms der angeschlossenen Kamera.

Über das Terminal wird nun OpenCV installiert. Dazu wird jeder der Befehle in der Installationsdatei in das Terminalfenster eingegeben. Eventuelle Aufforderungen oder Nachfragen („Möchten Sie fortfahren? (j/n)") müssen mit der j-Taste und der Eingabetaste quittiert werden.

Bild 5 zeigt die erfolgreiche Installation der Version 4.5.1 von OpenCV auf dem Raspberry Pi.

Nun kann die Installation von face_recognition:

sudo pip3 install face-recognition

und imutils erfolgen:

sudo pip3 install imutils

Der Download dieser Pakete kann wieder bis zu einer Stunde in Anspruch nehmen.

Nachdem die Programmpakete erfolgreich installiert wurden, steht dem praktischen Einsatz nichts mehr im Weg.

In einem ersten Schritt soll versucht werden, Gesichter im Blickfeld einer Kamera zu lokalisieren.

Insbesondere für Überwachungsaufgaben kann eine Weitwinkelkamera vorteilhaft eingesetzt werden. Die Joy-it-200°-Weitwinkel-Version (s. Bild 6) leistet hier gute Dienste. Sie wird genauso wie die normale Pi-Camera mit dem Raspberry Pi verbunden, bietet jedoch ein wesentlich größeres Gesichtsfeld (s. Bild 9).

Python erkennt Gesichter, Augen und ein Lächeln

Die Erkennung menschlicher Gesichter gehört zu den wichtigsten geistigen Leistungen, die ein neuer Erdenbürger erlernt. Die Erkennung bzw. Unterscheidung von Gesichtern wird vom menschlichen Gehirn normalerweise bereits innerhalb der ersten Lebensmonate erworben. Neugeborene interessieren sich von Geburt an mehr für Gesichter als für irgendwelche anderen Objekte in ihrem Umfeld. Sogar Ungeborene wenden sich bereits einem auf den Bauch der Mutter projiziertem Lichtmuster zu, wenn dieses das Grundmuster eines Gesichts aufweist.

Innerhalb des Bereichs des maschinellen Lernens zählt die Gesichtserkennung deshalb zu den bedeutsamsten biometrischen Verfahren. Sie wird häufig sicherheitstechnisch und kriminalistisch eingesetzt. Ihr Hauptzweck ist die Identifikation oder Authentifizierung von Personen.

Typischerweise dient die technische, computergestützte Gesichtserkennung der Zutrittskontrolle zu sicherheitsempfindlichen Bereichen. Aber auch für die Suche nach Dubletten in Datenbanken wird das Verfahren eingesetzt. So kann beispielsweise in Melderegistern Identitätsdiebstahl aufgedeckt werden.

OpenCV stellt eine Reihe von Prozeduren zur Verfügung, welche die maschinelle Gesichtserkennung in verschiedenen Varianten ermöglichen. Das Programmpaket enthält sowohl einen Trainer als auch einen Detektor. Hier soll zunächst der Detektor im Vordergrund stehen. OpenCV enthält bereits viele vortrainierte Klassifikatoren, z. B. für

- Gesichter
- Augen
- ein Lächeln

Diese XML-Dateien stehen z. B. unter [3] zur Verfügung und werden in einem Ordner unter

face_detection_recognition/HaarCascades

gespeichert.

Dann kann bereits mit wenigen Zeilen Code ein sehr effektives Gesichtserkennungssystem aufgebaut werden.

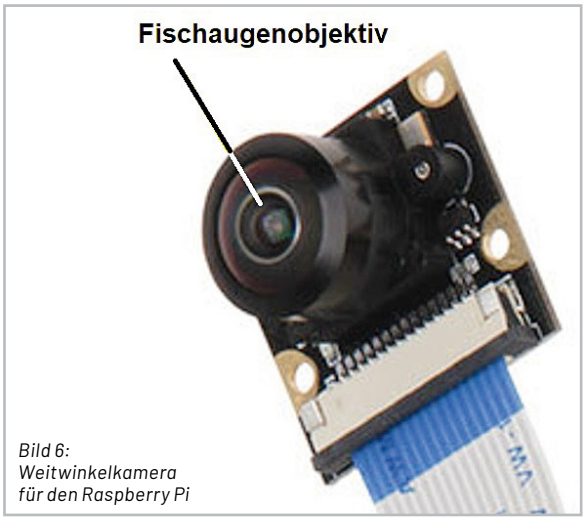

Bild 6: Weitwinkelkamera für den Raspberry Pi

Das Python-Programm (FaceDetector_1V1.py) dazu ist ebenfalls im Download-Paket [2] enthalten und fällt sehr kompakt aus:

```
import cv2

cascade="HaarCascades/haarcascade_frontalface_default.xml"
detector = cv2.CascadeClassifier(cascade)

green=(0,255,0)
framewidth=3

cap = cv2.VideoCapture(0)
cap.set(3,320) # set Width
cap.set(4,240) # set Height

while True:
    ret, img = cap.read()
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = detector.detectMultiScale(gray,scaleFactor=1.1,minNeighbors=5,minSize=(30,30))
    # print("numer of faces detected:", len(faces))
    for (x,y,w,h) in faces:
        cv2.rectangle(img,(x,y),(x+w,y+h),green,framewidth)
    cv2.imshow("Press 'q' quit",img)
    key = cv2.waitKey(1) & 0xFF
    if key == ord("q"):
        print("bye...")
        break
cap.release()
cv2.destroyAllWindows()
```

Nach dem Laden der OpenCV-Library wird die zu verwendende Kaskade aus dem oben genannten Verzeichnis geladen:

```
cascade="HaarCascades/haarcascade_frontalface_default.xml"
```

Danach wird der zugehörige Klassifikator als „detector" definert:

```
detector = cv2.CascadeClassifier(cascade)
```

Der Videostrom der aktiven Kamera wird über

```
cap = cv2.VideoCapture(0)
cap.set(3,320) # set Width
cap.set(4,240) # set Height
```

geöffnet. Dabei wird zusätzlich noch die Auflösung des Videobilds festgelegt. Die Parameter „3" bzw. „4" der Funktion cap.set stehen für die Breite und Höhe des Videostroms in Pixeleinheiten. Es ist zu beachten, dass hohe Auflösungen die Bildverarbeitung erheblich verlangsamen. Mit der hier verwendeten 0,7-Megapixel-Auflösung (320 x 240) lässt sich jedoch fast Echtzeitverarbeitung erreichen. Mit

```
ret, img = cap.read()
```

wird der Videostrom der Kamera in Einzelbilder (Frames) zerlegt. Die Funktion read() liest direkt aus der angegebenen Videoquelle, in diesem Beispiel der Webcam. Die Rückgabewerte sind:

- der tatsächlich gelesene Videoframe, also ein Frame pro Schleifendurchlauf
- ein Rückgabecode (ret)

Der Rückgabecode zeigt beispielsweise an, ob tatsächlich Frames gelesen werden. Wird aus einer Videodatei gelesen, ist damit das Ende des Videos detektierbar. Beim Lesen von der Webcam spielt dies keine Rolle, da der Videostream praktisch „endlos" läuft. In der Hauptschleife werden die Videobilder zunächst in eine Schwarz-Weiß-Version umgewandelt, da die Haar-Kaskaden damit effizienter arbeiten. Dann wird der Detektor auf den so erzeugten S/W-Videostrom angewendet:

```
faces = detector.detectMultiScale(gray,scaleFactor=1.1,minNeighbors=5,minSize=(30,30))
```

Der Detektor verwendet hier neben dem Videostrom selbst noch drei weitere Parameter:

1. scaleFactor: Für eine schnellere Erkennung kann der Wert auf bis zu 1,4 erhöht werden, allerdings besteht dann zunehmend das Risiko, dass Gesichter übersehen werden.
2. minNeighbors: Hierfür sind Werte von 3 bis 6 optimal; höhere Werte führen zu weniger Erkennungen, dafür aber mit besserer Erkennungsgenauigkeit.
3. minSize – Minimal detektierte Gesichtsgröße. Kleinere Objekte werden ignoriert.

Normalerweise liefert ein Wert von 30 x 30 Pixeln gute Ergebnisse.

Sobald ein Gesicht erkannt wurde, werden vier Parameter zurückgegeben:

x: x-Koordinate oberer linker Bildrand
y: y-Koordinate oberer linker Bildrand
w: Breite des erkannten Gesichts im Videostrom in Pixeln
h: Höhe des erkannten Gesichts im Videostrom in Pixeln

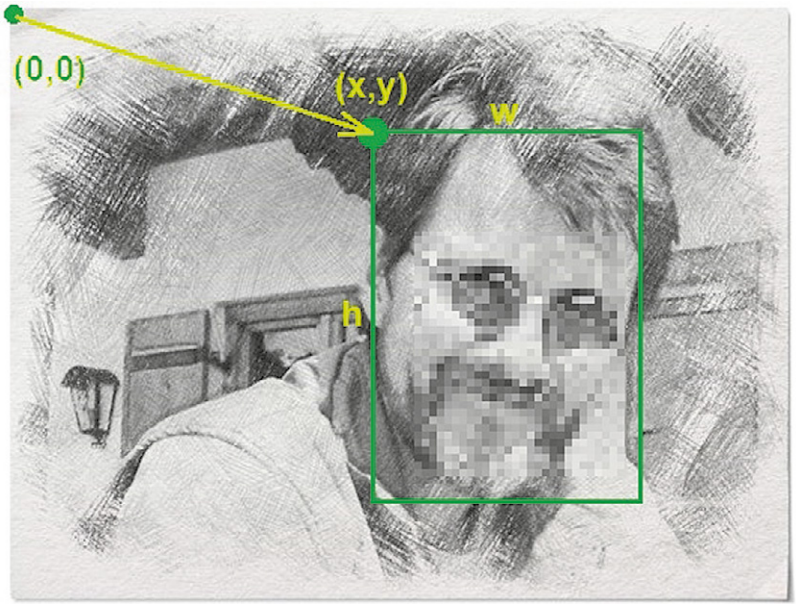

Bild 7: Definition des Detektorrahmens

Über die Funktion

```
cv2.rectangle(img,(x,y),(x+w,y+h),green,framewidth)
```

wird ein Rahmen mit den entsprechenden Koordinaten in der gewünschten Farbe (hier grün) und der Stärke „framewidth" um jedes erkannte Gesicht gezogen (Bild 7).

Die verbleibenden Programmzeilen dienen lediglich dem kontrollierten Programmabbruch.

Über die for-Schleife:

```
for (x,y,w,h) in faces:
```

werden alle von „faces" erkannten Gesichter abgefragt. Daher ist auch die Erfassung mehrerer Gesichter in einem Bild kein Problem für das Programm (Bild 8).

Über die Länge des Rückgabevektors von „faces" kann die Anzahl der im Bild detektierten Gesichter ausgegeben werden (die Zeile ist im obigen Code-Beispiel zunächst auskommentiert und kann bei Bedarf aktiviert werden):

```
print("numer of faces detected:", len(faces))
```

Damit lassen sich sehr interessante Anwendungen umsetzen. So kann man beispielsweise die Anzahl der Gäste auf einer Party im Laufe der Zeit bestimmen. Natürlich ist diese Funktion auch im kommerziellen Bereich von großer Bedeutung. Unter anderem lässt sich etwa die Anzahl der Kunden in einem Laden zeitaufgelöst verfolgen. Aber auch die Anzahl der Personen auf einem Volksfest oder auf einer öffentlichen Versammlung kann damit vollautomatisch erfasst werden. Andererseits zeigt sich aber auch hier bereits wieder, welche Gefahren die neuen Technologien mit sich bringen. Schon mit einfachsten Mitteln wird damit ein umfassendes Überwachungspotenzial eröffnet.

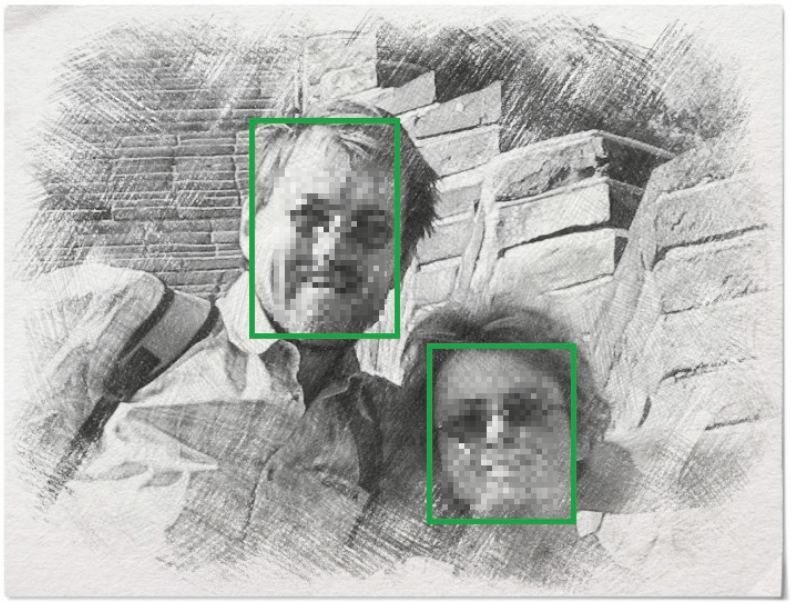

Bild 8: Auch mehrere Gesichter werden problemlos erfasst.

Wie man durch eigene Experimente leicht feststellen kann, werden bei dieser Variante auch Fotos von Gesichtern problemlos erkannt. Dies zeigt auch gleich eine wesentliche Problematik des Verfahrens auf. Bei Sicherheitsanwendungen könnte man das System mittels eines Lichtbilds sehr leicht täuschen. Hier zeigt sich die Überlegenheit von 3D-Messverfahren. Diese können mithilfe von zwei oder mehr Kameras ein einfaches zweidimensionales Foto sofort von einer realen dreidimensionalen Person unterscheiden.

Nachdem nun die Gesichtserkennung und -zählung problemlos möglich ist, kann man sich im nächsten Schritt der Identifizierung von Personen zuwenden. Die folgenden Abschnitte zeigen jedoch, dass dieses Verfahren nochmals deutlich aufwendiger ist als die einfache Gesichtserkennung.

Für Überwachungsaufgaben kann die 200°-Weitwinkelkamera vorteilhaft eingesetzt werden. Diese gestattet es, wesentlich größere Bereiche abzudecken (s. Bild 9).

Allerdings müssen hier wieder die gesetzlichen Bestimmungen eingehalten werden. So ist sicherzustellen, dass keine öffentlichen Flächen von der Kamera erfasst werden. Deshalb ist hier der folgende Hinweis angebracht:

Hinweis:
Bitte beachten Sie beim Einsatz von Kameras stets die aktuell gültigen gesetzlichen Bestimmungen und Datenschutzverordnungen!
Die Überwachung öffentlicher Räume ist Privatpersonen nicht gestattet.

Bild 9 zeigt als Beispiel die Überwachung eines privaten Zugangswegs mit der Weitwinkelkamera.

Erstellen von Testbildern und Trainieren

Um die Gesichtserkennung zur Gesichtsidentifizierung zu erweitern, sind zwei weitere Schritte erforderlich:

1. Aufnahme von Testbildern und Erstellung eines Testdatensatzes
2. Trainieren des Netzes mit diesem Datensatz

Zunächst müssen dazu die notwendigen Python-Programme:

- CollectTestPICs_1V0.py
- TrainFaces_1V0.py
- FaceRecognizer_1V0.py

aus dem Downloadpaket [2] auf den Raspberry Pi kopiert werden.

Bild 9: Personenerfassung mit einer Weitwinkelkamera

Dies kann entweder wieder über den Download auf einen PC und die anschließende Übertragung per USB-Stick oder mit dem direkten Download auf den Raspberry Pi erfolgen.

takeTestPICs.py *

```
1  import cv2
2
3  name = 'Riccarda'
4
5  cam = cv2.VideoCapture(0)
6  cv2.namedWindow("press space to take a photo", cv2.WINDOW_NORMAL)
7  cv2.resizeWindow("press space to take a photo", 500, 300)
8  img_counter = 0
9  print("press 'q' to quit")
10 while True:
11     ret, frame = cam.read()
12     if not ret:
13         print("failed to grab frame")
14         break
15     cv2.imshow("press space to take a photo", frame)
```

Bild 10: Der Name der zu erkennenden Person wird im Programm „CollectTestPICs_1V0.py" eingetragen.

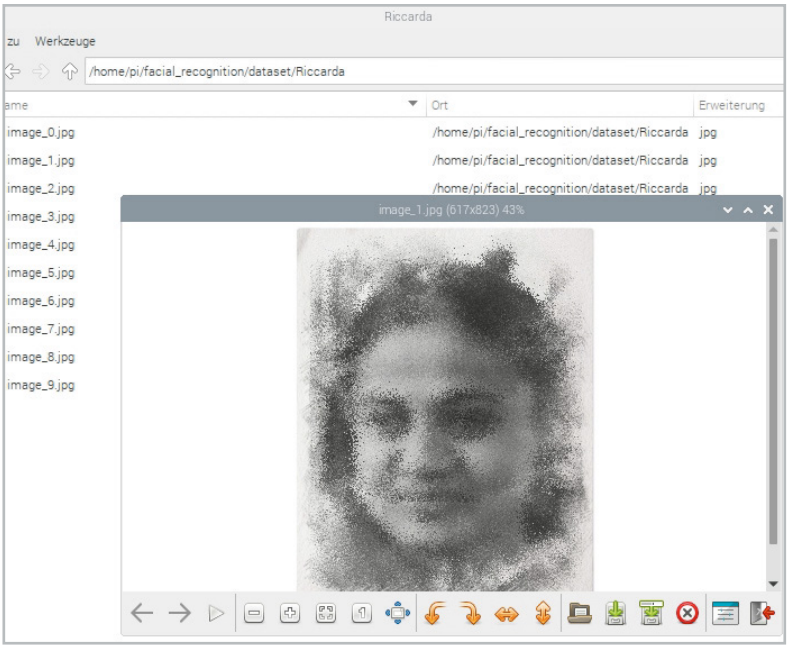

Bild 11: Die aufgenommenen Testbilder im Dateimanager mit Beispielbild

Für die Aufnahme der Testbilder muss zunächst ein Ordner mit dem Namen der jeweiligen Testperson erstellt werden, z. B.

/home/pi/face_detection_recognition/ ⮐
dataset/Riccarda

In der Thonny IDE [4] wird dann das Programm

CollectTestPICs_1V0.py

geöffnet. Dort wird der Name (z. B. „Riccarda") innerhalb der Anführungszeichen eingetragen (Bild 10). Die Namen im Datensatzordner und in der Datei müssen hier natürlich exakt übereinstimmen.

Nach dem Starten des Programms in der Thonny IDE öffnet sich ein Fenster mit einem Videostream der Webcam. Dies kann auch auf einem Raspberry Pi 4 mehrere Sekunden in Anspruch nehmen.

Nun muss die verwendete Kamera auf das Gesicht der Testperson ausgerichtet werden. Durch Drücken der Leertaste wird jetzt eine Fotoserie aufgenommen. Etwa zehn Fotos sollten für einen ersten Test ausreichend sein. Idealerweise wird der Kopf dabei leicht gedreht, gehoben oder gesenkt, um verschiedene perspektivische Winkel aufzunehmen. Brillenträger sollten einige Bilder mit und ohne Brille erstellen, sodass das Gesicht später in beiden Fällen erkannt wird. Über die „q"-Taste (für „quit") kann das Programm beendet werden. Anschließend kann man die Fotos im Dateimanager überprüfen (Bild 11).

Falls gewünscht, kann die Prozedur für weitere Personen wiederholt werden. Nach der Zusammenstellung des Trainingsdatensatzes erfolgt im nächsten Schritt das Training des Models. Dazu muss lediglich das Programm

TrainFaces_1V0.py

ausgeführt werden.

Es dauert ungefähr 3 bis 4 Sekunden, bis der Raspberry Pi ein Foto des Datensatzes analysiert hat. Für einen Datensatz mit 20 Bildern (z. B. zwei Personen mit jeweils zehn Bildern) werden also ungefähr ein bis zwei Minuten benötigt. Die so gewonnenen Trainingsdaten werden schließlich in der Datei „encodings.pickle" abgespeichert.

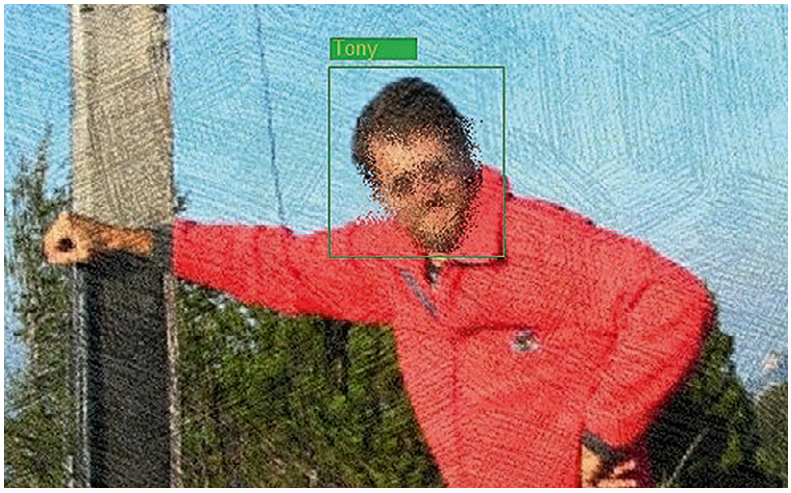

Bild 12: Identifizierung von Personen mit dem Raspberry und der PiCam

Selbsterkenntnis

Nach Abschluss der Vorarbeiten kann nun die eigentliche Gesichtserkennung mit

FaceRecognizer_1V0.py

gestartet werden. Nach wenigen Sekunden öffnet sich wieder die Webcam-Ansicht. Befindet sich ein Gesicht im Blickfeld der Kamera, wird dieses automatisch analysiert. Wurde das Gesicht erkannt, wird es mit einem Rechteck markiert. Wenn das Modell korrekt trainiert wurde, erscheint dazu auch der Name der Person (Bild 12). Wurden noch weitere Gesichter trainiert, werden diese ebenfalls erkannt. Falls es sich um ein unbekanntes Gesicht handelt, wird dieses mit „unknown" markiert. Das Programm kann wieder mit „q" beendet werden.

Das eigene Gesicht als Schlüssel

Abschließend soll an dieser Stelle nochmals von den speziellen Fähigkeiten des Raspberry Pi Gebrauch gemacht werden. Im Gegensatz zu klassischen PCs oder Laptops ist dieser in der Lage, über I/O-Pins elektronische Geräte direkt anzusteuern. Damit kann man die Gesichtsidentifizierung zu einem Biometriescanner erweitern. Dieser erlaubt es dann, eine Tür zu öffnen, wenn eine bestimmte Person erkannt wurde. Im Code muss dazu lediglich ein bestimmter Pin (z. B. GPIO 23) geschaltet werden, sobald ein bestimmtes Gesicht erkannt wurde:

```
if "Tony" in names:
    print(names)
    GPIO.output(23, GPIO.HIGH)
else:
    GPIO.output(23, GPIO.LOW)
```

Diese Erweiterung ist im Programm „FaceKey_1V0.py" enthalten. Auf der Hardwareseite muss nun lediglich ein elektromechanischer Türöffner angeschlossen werden. Sobald nun das Gesicht einer zutrittsberechtigten Person detektiert wird, öffnet sich das betreffende Schloss. Allen anderen bleibt der Zugang verwehrt.

Hinweis:
Bei praktischen Anwendungen sollte man immer bedenken, dass diese einfache Gesichtsidentifikation keinerlei Sicherheitsstandards erfüllt. Für die Sicherung von Haus- oder Wohnungstüren sind in jedem Fall professionelle Systeme erforderlich.

Fazit und Ausblick

Die automatische Personenidentifizierung muss stets mit einer gewissen Skepsis betrachtet werden. Auch aus diesem Grunde ist es von höchster Wichtigkeit, die Potenziale und Möglichkeiten auf diesem Gebiet möglichst genau zu kennen.

Dieser Beitrag zeigt, dass bereits mit einfachen Mitteln Personenidentifizierungssysteme aufgebaut werden können. Dabei kann nicht nur erkannt werden, dass sich ein Gesicht im Blickfeld einer Kamera befindet. Vielmehr können nach entsprechendem Training einzelne Personen bzw. Gesichter sogar identifiziert werden. Damit wird natürlich auch einem potenziellen Missbrauch Tür und Tor geöffnet. Illegale Personenüberwachung oder die Erstellung von Bewegungsprofilen ist mit moderner Technik bereits mit geringem finanziellen Aufwand möglich. Ob sich diese Entwicklungen zum Fluch oder zum Segen der Menschheit entwickeln, muss sich erst noch zeigen.

Im nächsten Beitrag zu dieser Serie wird es um das Trainieren eigener Netze gehen. Bislang wurden überwiegend fertig trainierte Modelle eingesetzt. Diese waren bis zu einem gewissen Maße lernfähig. So konnten dem Gesichtserkennungsmodell neue Personen hinzugefügt werden. Dennoch waren die Modelle auf bestimmte vordefinierte Aufgaben beschränkt. Beim Trainieren eigener Netze wird man dagegen nochmals wesentlich flexibler. Damit wird es auch möglich, spezielle Aufgaben zu lösen. So können etwa Systeme entwickelt werden, welche die Erkennung von Tieren oder speziellen Tierarten, das Sortieren von Waren oder Objekten wie z. B. Werk- oder Spielzeugen oder die Erfassung von Handgesten ermöglichen. ELV

Material	Artikel-Nr.
Raspberry Pi 4 Model B, 8 GB RAM	250567
Joy-IT 200°-Weitwinkelkamera	145132
Standard 5-MP-Kamera für Raspberry Pi	145134

i Weitere Infos

[1] Download-Paket zu diesem Beitrag: Artikel-Nr. 252589
[2] Fachbeitrag Handschrifterkennung, ELVjournal 5/2021: Artikel-Nr. 252233
[3] https://github.com/Mjrovai/OpenCV-Face-Recognition/tree/master/FaceDetection/Cascades
[4] Machine Learning mit PC, Raspberry Pi und MaixDuino, G. Spanner, elektor-Verlag (2021)

Alle Links finden Sie auch online unter: de.elv.com/elvjournal-links