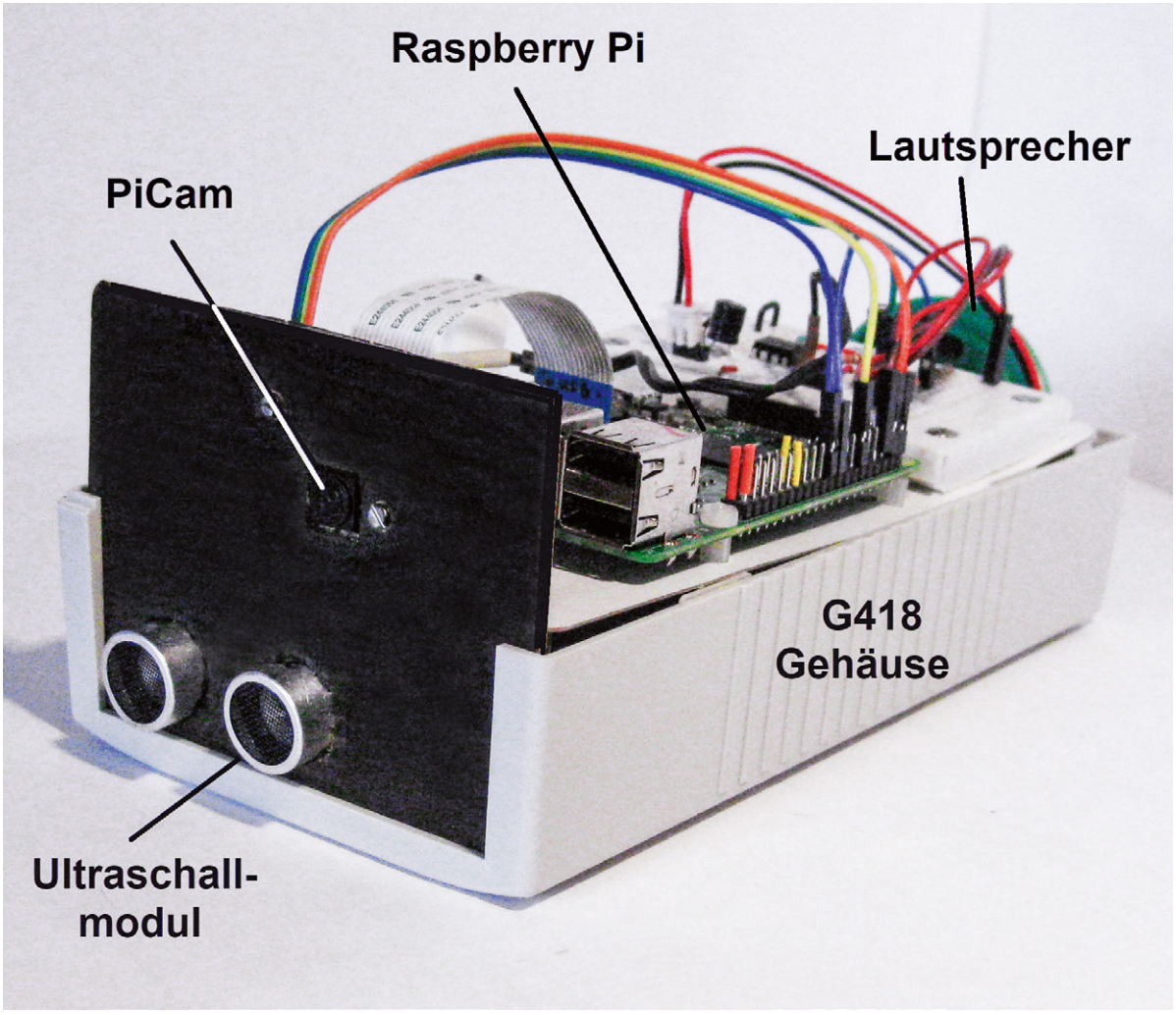
KI-Praxis V

Teil 5

Objekterkennung mit einer See-and-Talk-Box

Gegenstände und Objekte zu erkennen und zu identifizieren zählt zu den zentralen Aufgaben der Künstlichen Intelligenz. Vor allem für die Entwicklung autonomer Fahrzeuge ist eine zuverlässige Erfassung anderer Fahrzeuge, Personen und Verkehrszeichen von entscheidender Bedeutung. Rapide Fortschritte in der Forschung führten zu Softwaresystemen, die es gestatten, dieses Problem auch auf Kleinrechnern anzugehen. In diesem Beitrag wird zunächst die schrittweise Einrichtung der Open-Source-Plattform für maschinelles Lernen TensorFlow Lite auf dem Raspberry Pi erläutert. Im Anschluss daran werden mittels eines darauf aufbauenden KI-Systems Objekterkennungsmodelle ausgeführt und getestet.



See-and-Talk-Box

In Kombination mit der Sprachausgabe aus dem letzten Beitrag kann mithilfe der Objekterkennung eine See-and-Talk-Box aufgebaut werden. Dieses Gerät ist in der Lage, optisch erfasste Objekte oder Gegenstände akustisch zu beschreiben. Befindet sich beispielsweise eine Orange (Apfelsine) im Sichtfeld der Kamera, so wird die Information „I see an orange" ausgegeben. Derartige Geräte können beispielsweise für blinde oder stark sehbehinderte Menschen eine wertvolle Hilfe sein.

Neben der Original-PiCam wird hierfür auch eine Weitwinkelkamera zum Einsatz kommen, da sie aufgrund ihres wesentlich größeren Gesichtsfelds auch für Überwachungsaufgaben bestens geeignet ist.

Hardware-Voraussetzungen

Für die praktische Umsetzung der Projekte sind die folgenden Hardware-Komponenten erforderlich (siehe dazu auch „Material" am Ende des Beitrags):

- Raspberry Pi 4 mit 8 GB RAM
- PiCam oder Weitwinkelkamera Joy-it 200°
- Aktivboxen/Audioverstärker mit angeschlossenem Lautsprecher

Zusätzlich kann auch ein Ultraschallmodul eingesetzt werden. Dann lassen sich sogar die Entfernungen zu den vom Kameramodul erfassten Objekten mit hoher Präzision angeben.

Für den Aufbau der See-and-Talk-Box empfiehlt sich ein passendes Gehäuse, das alle Komponenten inklusive der Stromversorgung aufnehmen kann. Eine Variante ist das Kunststoff-Element-Gehäuse G418 (s. Material), das auch im Titelbild zu sehen ist.

TensorFlow Lite auf dem Raspberry Pi 4

Die Rechenleistung eines Raspberry Pi 4 mit einem Arbeitsspeicher von 8 GB ist durchaus ausreichend, um KI-basierte Objekterkennungen nahezu in Echtzeit auszuführen. Besonders geeignet für diese Aufgabe ist die Light-Version von TensorFlow. Diese ist seit einiger Zeit auch für „Embedded Systems" wie den Raspberry Pi verfügbar.

TensorFlow Lite (TFLite) basiert auf einer Reihe von Tools, welche die effiziente Anwendung von maschinellem Lernen auf kleinen und mobilen Geräten ermöglicht. TFLite-Modelle laufen auf dem Raspberry Pi 4 deutlich schneller als die konventionellen TensorFlow-Varianten. Zudem ist das Einrichten von TensorFlow Lite auf dem Raspberry Pi wesentlich einfacher als die Installation eines klassischen TensorFlow-Pakets. Um mit TensorFlow Lite beispielsweise eine Objekterkennung einzurichten, sollten die folgenden fünf Schritte abgearbeitet werden:

1. Aktualisieren des Raspberry Pi
2. Laden eines Repositorys und Erstellen einer virtuellen Umgebung
3. Installieren von TensorFlow und OpenCV und der zugehörigen Abhängigkeiten
4. Einrichten eines Objekterkennungsmodells
5. Ausführen des Modells

Der erste Schritt, die Aktualisierung, erfolgt über die bekannten Anweisungen:

```
sudo apt-get update
sudo apt-get dist-upgrade
```

Je nachdem wie lange es her ist, seit der Pi das letzte Mal aktualisiert wurde, kann ein vollständiges Update zwischen wenigen Minuten und über einer Stunde in Anspruch nehmen.

Im zweiten Schritt muss das erforderliche GitHub-Repository auf den Raspberry Pi geladen werden. Dieses enthält alle erforderlichen Software-Komponenten für die Ausführung von TensorFlow Lite. Zudem ist ein Shell-Skript integriert, das die komplette Installation vereinfacht. Das Laden des Repositorys erfolgt über diese Anweisung:

```
git clone https://github.com/EdjeElectronics/TensorFlow-↩
Lite-Object-Detection-on-Android-and-Raspberry-Pi.git
```

Mit dem Clone-Befehl werden alle erforderlichen Dateien in einen Ordner namens „TensorFlow-Lite-Objecdetection ..." heruntergeladen. Dieser Ordner kann der Einfachheit halber in z. B. „TFlite" oder einen mit einer anderen, kürzeren Bezeichnung umbenannt werden:

```
mv TensorFlow-Lite-Object-Detection-on-Android-↩
and-Raspberry-Pi TFlite
```

Anschließend wird in dieses neue Verzeichnis gewechselt:

```
cd TFlite
```

Im Folgenden wird nur in diesem neu erstellten Verzeichnis:

```
/home/pi/TFlite
```

gearbeitet.

Die virtuelle Umgebung

Als Nächstes wird eine sogenannte virtuelle Umgebung namens „TFlite-env" erstellt. Durch die Verwendung von virtuellen Umgebungen (Virtual Environments) werden Konflikte zwischen verschiedenen Bibliotheksversionen vermieden. Wird z. B. eine spezielle TensorFlow-Version in einer eigenen Umgebung installiert, können andere Varianten auf dem Pi installiert bleiben, ohne dass es zu unerwünschten „Nebenwirkungen" kommt.

Für eine virtuelle Umgebung muss zunächst das erforderliche Paket installiert werden:

```
sudo pip3 install virtualenv
```

Anschließend wird ein Ordner namens „TFlite" erstellt und in diesen Ordner gewechselt. Danach kann die eigentliche Umgebung namens „TFlite-env", mit

```
python3 -m venv TFlite-env
```

erstellt werden. Damit wird auch automatisch ein Ordner mit dem Namen TFlite-env innerhalb des Verzeichnisses TFlite erstellt. In den Ordner TFlite-env werden alle für diese Umgebung relevanten Paketbibliotheken installiert.

Bild 1: Die virtuelle Umgebung TFlite ist aktiv.

Bild 2: Die Pakete sind erfolgreich installiert.

Nun muss die virtuelle Umgebung noch über

source TFlite-env/bin/activate

aktiviert werden. Der Quellbefehl

TFlite-env/bin/enabled

muss immer im Verzeichnis

/home/pi/tflite

ausgeführt werden, um die Umgebung neu zu aktivieren. Ob die TFlite-Umgebung tatsächlich aktiv ist, kann am Vorsatz „TFlite-env" vor dem Pfad in der Eingabeaufforderung festgestellt werden (Bild 1).

Paketinstallation

Als Nächstes sind die folgenden Paketbibliotheken zu installieren:

- TensorFlow
- OpenCV
- die erforderlichen Abhängigkeiten für beide Pakete

Prinzipiell ist OpenCV keine unbedingte Voraussetzung für die Arbeit mit TensorFlow Lite. Für die Objekterkennung ist das Paket jedoch erforderlich, um mit der PiCam Bilder zu erfassen und die Ergebnisse in Form von Grafikfenstern darstellen zu können.

Die Anweisungssequenz für die Installation der oben genannten Pakete findet sich wie immer im Download-Paket zu diesem Beitrag [1]. Die einzelnen Anweisungen können so wieder direkt aus der Datei

install_TFlite_openCV.txt

in das Terminal des Raspberry Pi kopiert werden.

Im Laufe der Installation werden Dateien mit einem Datenvolumen von etwa 400 MB heruntergeladen. Dies kann beispielsweise bei einer 16-MBit-Internetverbindung bis zu einer halben Stunde in Anspruch nehmen. Die erfolgreiche Installation kann mit *pip freeze* überprüft werden (Bild 2). Man erkennt, dass nur tatsächlich in der virtuellen Umgebung installierte Pakete sichtbar sind.

Das Erkennungsmodell

Nun muss noch das passende Erkennungsmodell eingerichtet werden. Hier soll zunächst ein von Google bereitgestelltes TFLite-Beispielmodell zum Einsatz kommen. Alternativ wäre es jedoch auch möglich, ein eigenes Modell zu trainieren. Darauf wird jedoch erst in einem späteren Beitrag näher eingegangen.

Einem Erkennungsmodell sind zwei Dateien zugeordnet:

– die Erkennung.tflite-Datei (das Modell selbst)
– die labelmap.txt-Datei (Label-Map für das Modell)

Tabelle 1

Leistungsmerkmale der Weitwinkelkamera im Vergleich zur Standard-PiCam

Model	5 MP Weitwinkel Kameramodul	Standard-PiCam
Chip	OV5647	OV5647
Auflösung	5 Megapixel; 2952x1944	5 Megapixel; 2952x1944
Betrachtungswinkel	200°	75,7°
Brennweite	0,87 mm	3,6 mm (einstellbar)
Blende	2,0	1,8
Abm. (B x H x T)	25 x 16,5 x 24 mm	36 x 9 x 26,5 mm

Google stellt ein SSDLite-MobileNet-v2-Objekterkennungsmodell zur Verfügung, das mit dem MSCOCO-Dataset (MicroSoft Common Objects in Context) trainiert und für die Ausführung auf TensorFlow Lite adaptiert wurde. Es kann 80 verschiedene Objekte wie

Personen, Tiere,
Fahrzeuge,
Alltagsgegenstände
(Tische, Stühle, Topfpflanzen, Geschirr),
Obst usw.

erkennen und identifizieren. Welche Objekte im Einzelnen erkannt werden, kann in der labelmap.txt-Datei überprüft werden. Werden in diese Datei beispielsweise die deutschsprachigen Begriffe eingegeben, so werden die Objekte mit den entsprechenden deutschen Ausdrücken bezeichnet. Die Abbildungen in den folgenden Abschnitten wurden teilweise unter Verwendung einer deutschsprachigen Labelmap erstellt.

Das Modell kann über

wget https://storage.googleapis.com/ ⮐
download.tensorflow.org/models/tflite/ ⮐
coco_ssd_mobilenet_v1_1.0_quant_ ⮐
2018_06_29.zip

geladen werden. Das Entpacken und Speichern in einen Ordner namens „Sample_TFLite_model" erfolgt über

unzip coco_ssd_mobilenet_v1_1.0_quant_ ⮐
2018_06_29.zip-d Sample_TFlite_model

Der Befehl erstellt den Ordner automatisch. Damit steht das Beispielmodell zu Anwendung bereit.

Anschließen und aktivieren der Kamera

Anschluss und Inbetriebnahme der Pi-Kamera wurde bereits in Teil 3 „Handschrifterkennung" [2] der Beitragserie ausführlich erläutert. Details können dort bei Bedarf nachgeschlagen werden. Hier sollen deshalb nur noch einmal kurz die wichtigsten Schritte dargelegt werden.

Der Anschluss der Kamera erfolgt über die 15-polige serielle MIPI-Schnittstelle. Um das 15-polige Flachbandkabel des Kameramoduls mit dem Board zu verbinden, zieht man den oberen Teil des CSI-Steckverbinders (CSI = Camera Serial Interface) etwas nach oben, steckt dann das Flachbandkabel mit der blauen Markierung zum Ethernet-Anschluss hinein und drückt den Verschluss wieder nach unten.

Die Aktivierung der Kamera erfolgt über das Konfigurationstool raspi-config. Dort wird einfach die Kamera auf „Enable" gesetzt. Abschließend ist ein Reboot erforderlich. Mit

raspistill -o image.jpg

kann die Kamera getestet werden. Das aufgenommene Bild befindet sich im Verzeichnis home/pi.

Neben der klassischen PiCam kann auch eine Weitwinkelkamera eingesetzt werden (s. Materialliste). Das wesentlich größere Gesichtsfeld dieser Kameraversion kann seine Vorteile insbesondere bei Überwachungsaufgaben ausspielen.

Tabelle 1 fasst die Leistungsmerkmale dieser Kameraversion im Vergleich zur Standard-PiCam zusammen.

Bild 3: Gesichtsfeld der Standard-PiCam (ca. 76° horizontal)

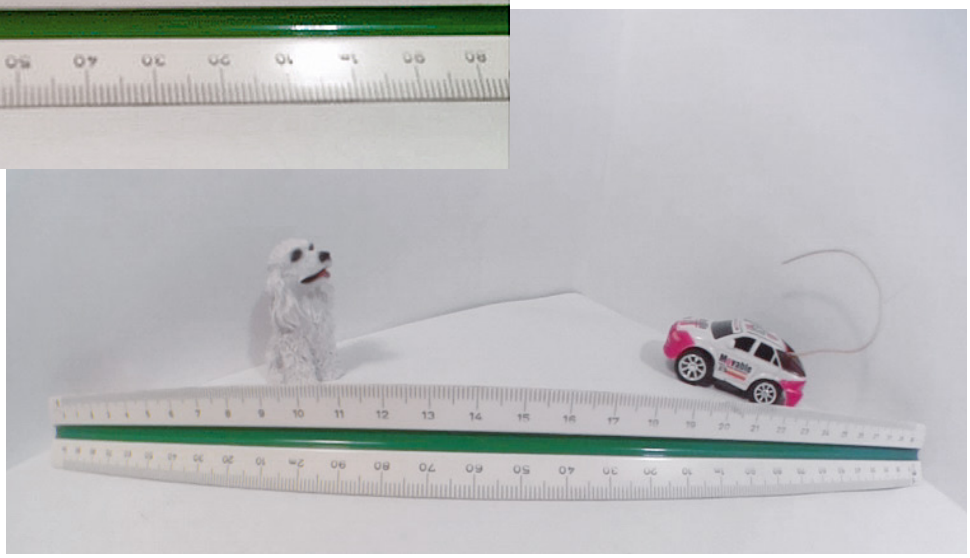


Bild 4: Gesichtsfeld der 200°-Weitwinkelkamera

Bild 3 und Bild 4 zeigen die Gesichtsfelder der beiden Kameras im direkten Vergleich. Man erkennt, dass die Weitwinkelkamera einen deutlich größeren Bereich optisch abdecken kann. Dies kann sowohl für die Überwachung von Räumen oder Fluren als auch in der Robotik oder für autonome Fahrzeuganwendungen ein wesentlicher Vorteil sein.

Das TFLite-Modell in Aktion

Für die Ausführung des Objekterkennungsmodells ist noch ein passendes Python-Programm erforderlich. Diese findet sich im Download-Paket:

TFlite_detection_PiCam.py

Nachdem das Programm auf dem Raspberry Pi in das Verzeichnis

/home/pi/Tflite

kopiert wurde, muss in dieses Verzeichnis gewechselt werden: cd TFlite

Dann wird die virtuelle Umgebung aktiviert:

source TFlite-env/bin/activate

Im Anschluss daran kann die Objekterkennung gestartet werden.

python3 TFlite_detection_PiCam_1V1.py

Um möglichst viel Speicher und Rechenleistung freizugeben, sollten alle anderen Anwendungen auf dem Raspberry Pi geschlossen werden.

Nach einigen Augenblicken der Initialisierung wird ein Fenster mit dem Kamera-Feed angezeigt. Um erkannte Objekte herum werden Begrenzungsrahmen und Beschriftungen angezeigt (Bild 5).

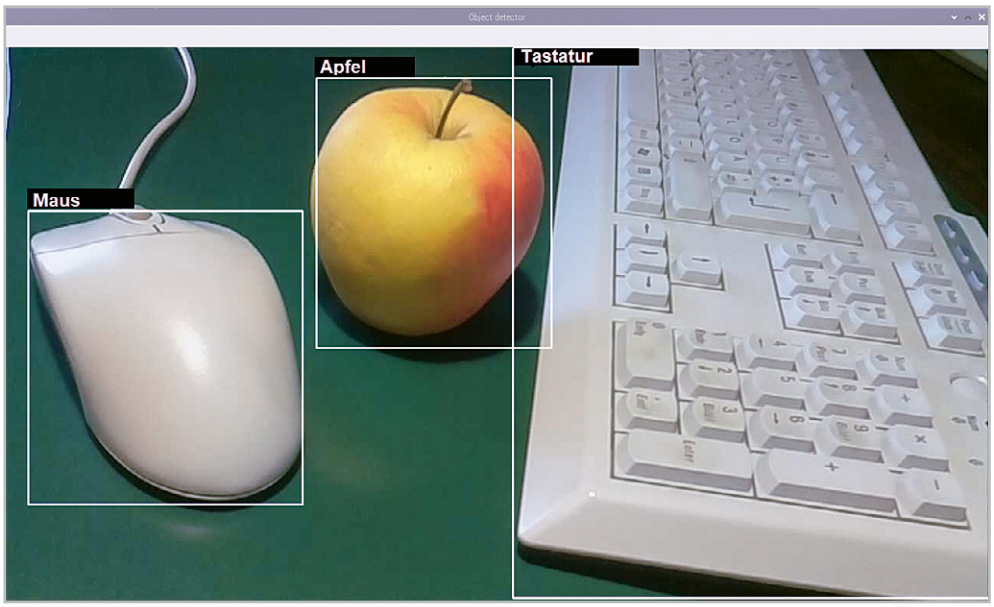


Bild 5: Erfolgreiche Objekterkennung

Man erkennt, dass die Identifizierung von alltäglichen Bürotensilien kein Problem darstellt. Auch bei Obstsorten erweist sich der Algorithmus geradezu als Experte (Bild 6). Entscheidend ist nicht etwa nur die Farbe: auch ein gelber Apfel wird nicht als Banane erkannt.

Auch bei Verwendung der Weitwinkelkamera werden bekannte Objekte korrekt klassifiziert (Bild 7).

Dies zeigt die hohe Qualität des Systems. Letztendlich kann die Fähigkeit, Gegenstände auch unter erschwerten Bedingungen zu erkennen, als Maß für die „Intelligenz" eines Systems betrachtet werden. Bild 7 verdeutlicht, dass der MSCOCO-Algorithmus gelernt

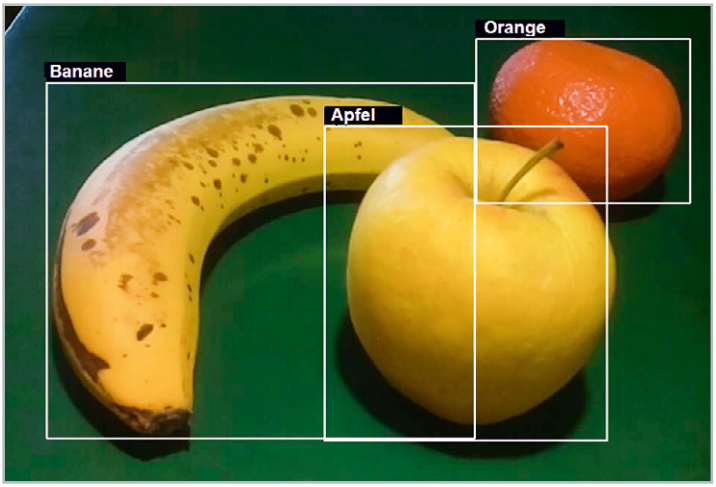


Bild 6: Das Erkennen verschiedener Obstsorten ist kein Problem für den Algorithmus.

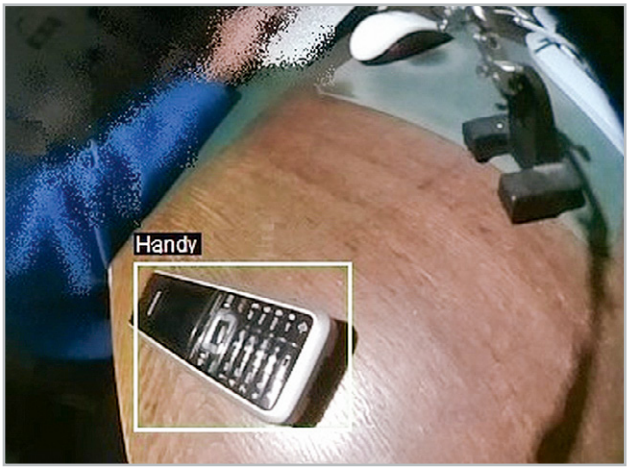


Bild 7: Trotz Verzerrungen werden Objekte auch mit der Weitwinkelkamera erkannt.

hat, ein Handy auch als solches zu identifizieren, wenn es aufgrund der Weitwinkelabbildung keine klassische Rechteckform mehr aufweist. Damit kann die Weitwinkelkamera insbesondere bei Überwachungsaufgaben und in der Robotik gute Dienste leisten. Die Frame-Rate erreicht je nach Komplexität des aktuellen Bildes Werte von bis zu 20 Bildern pro Sekunde. Dieser Wert kommt einer Echtzeiterfassung durchaus recht nahe. Für eine einfache Video-Überwachungsanwendung ist diese Frame-Rate jedoch ohnehin meist vollkommen ausreichend.

Bild 8 zeigt, dass die Erkennung von verschiedenen Alltagsgegenständen vom Algorithmus mühelos gemeistert wird. Einzelobjekte werden also bereits

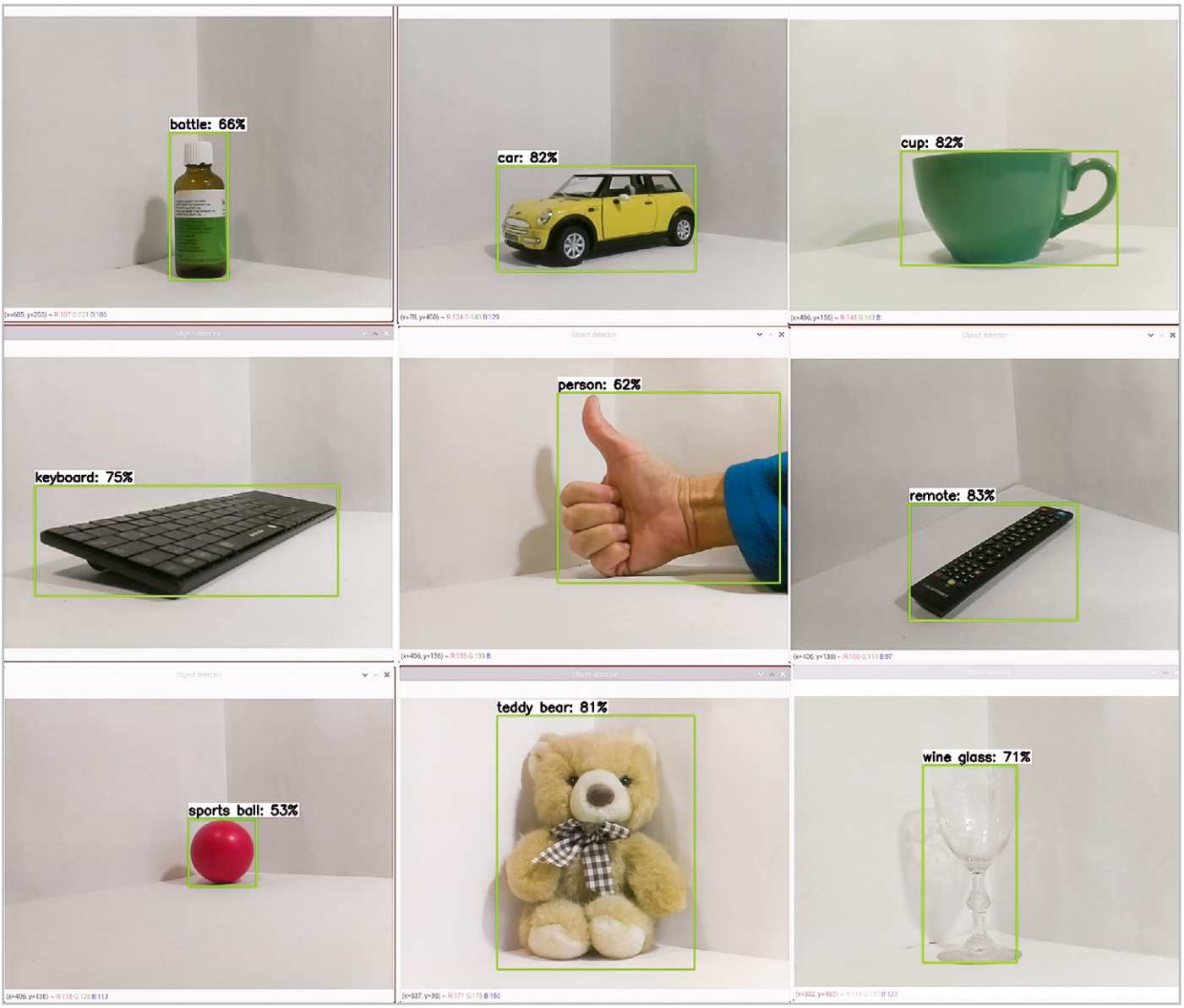


Bild 8: Verschiedenste Alltagsgegenstände werden sicher erkannt.

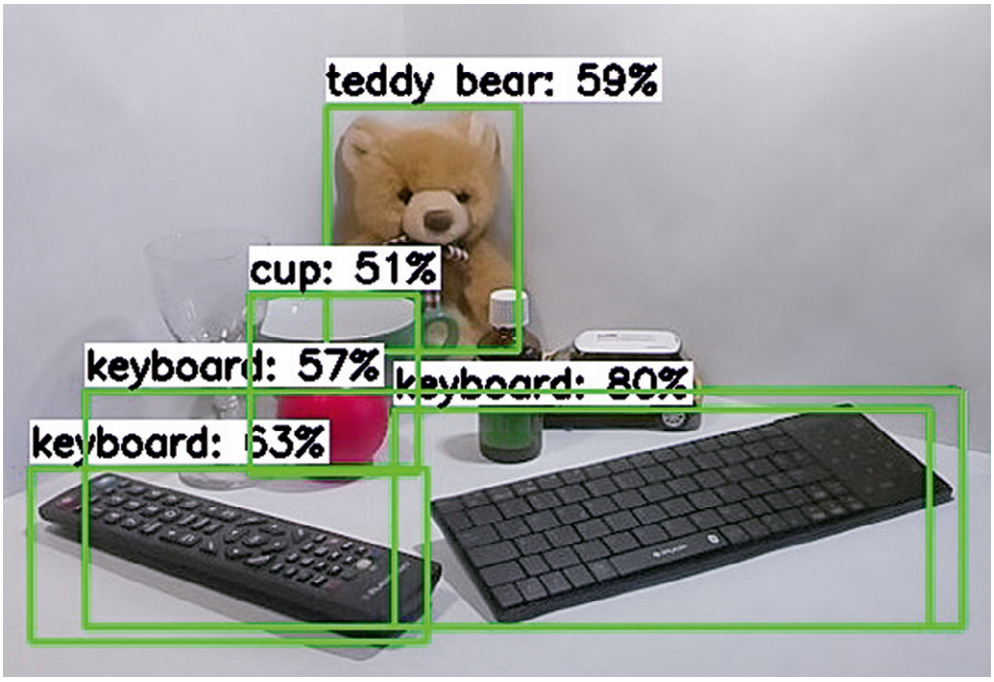


Bild 9: Bei mehreren Objekten erreicht das System seine Grenze.

mit hoher Trefferwahrscheinlichkeit erkannt. Sobald allerdings mehrere bekannte Objekte im Blickfeld erscheinen, erreicht das System seine Grenzen, wie Bild 9 zeigt. Einige Objekte, wie etwa das Weinglas und das teilweise verdeckte Auto werden auch nach längerer Zeit nicht erkannt. Die Fernbedienung wird zwar erfasst, jedoch fehlerhaft als Tastatur kategorisiert. Zudem werden die Fernbedienung und die Tastatur zusammengefasst als weitere „Tastatur" eingeordnet.

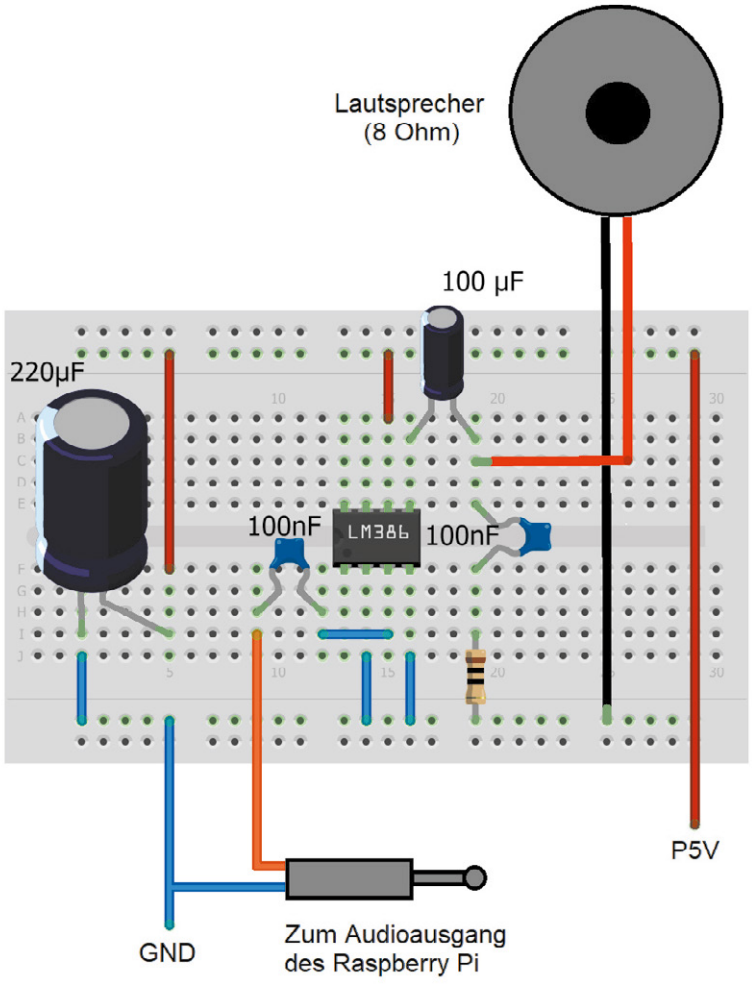


Bild 10: Kompakter Audioverstärker

Das System wäre also beispielsweise für den Einsatz als zentrales Erkennungssystem in einem autonomen Fahrzeug sicher noch nicht geeignet. Es ist dennoch erstaunlich, dass selbst ein Einplatinenrechner wie der Raspberry Pi bereits in der Lage ist, Multiobjekterkennungen durchzuführen.

Sag, was du siehst! Die See-and-Talk-Box

Bereits im vierten Teil („Spracherkennung und Sprachsynthese", [3]) zu dieser Beitragserie wurden die Fähigkeiten des Raspberry Pi zur Ausgabe menschlicher Sprache diskutiert. Diese Möglichkeit soll auch hier nochmals eingesetzt werden.

Nachdem eine erfolgreiche Objekterkennung installiert ist, kann man die Information über erkannte Gegenstände, Verkehrszeichen oder Fahrzeuge etc. auch akustisch ausgeben. Hierzu muss lediglich eine Aktivbox an die 3,5-mm-Audio-Buchse des Raspberry Pi angeschlossen werden. Alternativ ist auch ein im Eigenbau erstellter Audioverstärker mit Lautsprecher verwendbar (Bild 10). Diese Variante bietet den Vorteil, dass sie einfacher in ein kompaktes Gehäuse integriert werden kann (s. Titelbild des Beitrags). Der eingezeichnete rechte 100-nF-Kondensator sowie der 10-Ω-Widerstand (Farbringkombination braun – schwarz –- schwarz) können unter Umständen entfallen. Sie dienen lediglich dazu, die Schwingneigung des Verstärkers zu reduzieren.

Bild 11 zeigt einen Schaltplan für einen Gesamtaufbau, der bereits auch die optionale Ultraschallerweiterung für den nächsten Abschnitt enthält. Für die Übertragung des Audiosignals ist ein Kanal ausreichend, da die Sprachausgabe ohnehin in Mono erfolgt. Bei Verwendung einer externen Aktivbox können natürlich beide Kanäle (R & L) verwendet werden.

Auf der Softwareseite muss lediglich das Objekterkennungsprogramm um die Sprachausgabe erweitert werden. Hierfür kommt das bereits aus dem letzten Beitrag [3] bekannte eSpeak-Modul zum Einsatz. (Bei Bedarf können die notwendigen Details

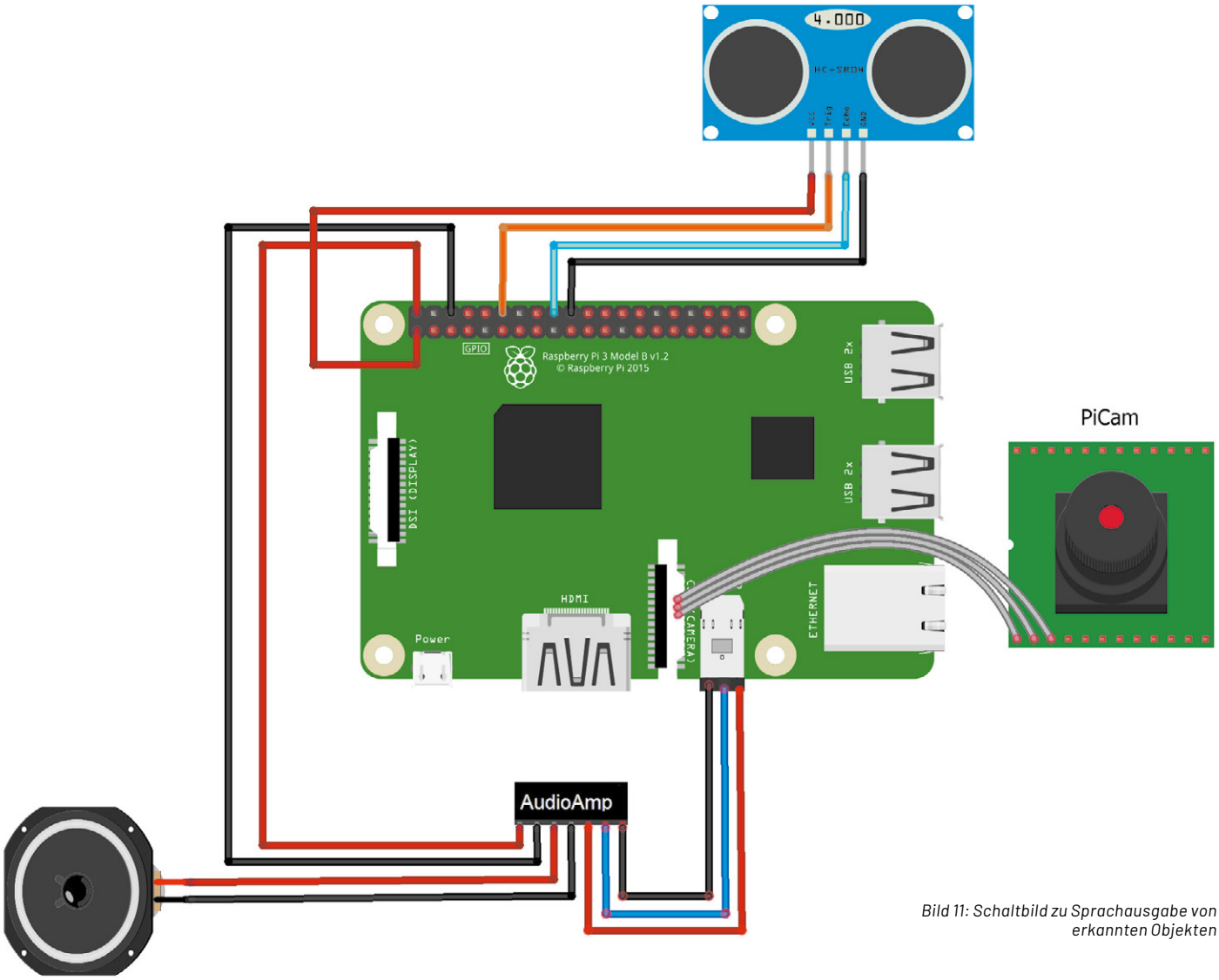


Bild 11: Schaltbild zu Sprachausgabe von erkannten Objekten

zur Installation des Softwarepakets im oben genannten Beitrag nachgelesen werden.) Ist das Paket erfolgreich installiert, können durch das Einfügen der Zeilen

```
os.system("espeak "I see a'")
speak = str(object _ name)
os.popen('espeak "' + speak + '" --stdout | aplay 2> /dev/null').read()
```

nach der Objekterfassung die einzelnen Objekte der Reihe nach akustisch ausgegeben werden. Das gesamte Programm ist im Download-Paket unter

TFLite_detection_PiCam_speaker_1V0.py

verfügbar. Der Start des Programms muss wieder in der oben erstellten virtuellen Umgebung erfolgen.

Eine so entstehende „See-and-Talk-Box" kann beispielsweise für blinde oder sehbehinderte Menschen eine wichtige Alltagshilfe sein. Auch auf kommerzieller Ebene werden derartige Systeme in zunehmenden Maßen für die Unterstützung von Menschen mit entsprechenden Einschränkungen eingesetzt. Natürlich kommt hier speziell entwickelte Hardware zum Einsatz. Die Geräte können damit wesentlich kompakter gebaut werden als die hier vorgestellte Raspberry-Pi-Version.

So ist es bereits möglich, das gesamte System in eine Brille zu integrieren, die sich äußerlich kaum von einer gewöhnlichen Sehhilfe unterscheidet. Dennoch enthält das System eine Kamera, das Objekterkennungsmodul und eine Audioausgabe im Ohrenbügel der Brille.

Mit der einfachen Objekterkennung ist jedoch noch lange nicht das Ende der Möglichkeiten erreicht. So sind etwa wie Brillen am Kopf tragbare Miniaturcomputer verfügbar, die neben der akustischen Ausgabe auch über ein integriertes optisches Display verfügen, das am Rand des Sichtfelds auf einem Brillenrahmen montiert ist. Damit können auch allgemeine Informationen und zusätzliche Daten zu den von der Kamera erkannten Objekten eingeblendet werden.

Eines der ersten kommerziell auf diesem Gebiet verfügbaren Geräte war das Google-Glass-System. Google Glass ist ohne Zweifel ein höchst innovatives Anwendungsprojekt für KI-Verfahren. Allerdings zeigt das Gerät auch sehr deutlich die Gefahren der neuen Technologie. Von besonderer Bedeutung ist die Gefährdung der Privatsphäre beobachteter Personen. Die Brille ist zweifellos in der Lage, unauffällig die Umgebung des Trägers auszuspähen und Aufzeichnungen aller Art drahtlos auf einen Server zu übertragen.

Auch wenn das hier vorgestellte System nicht so leistungsfähig ist wie kommerziell entwickelte Geräte, so zeigt es doch, dass man bereits mit einfachen Mitteln ein brauchbares Objekterfassungssystem aufbauen kann. Da der Raspberry Pi im Gegensatz

zu einem Laptop oder PC auch über frei programmierbare I/O-Pins verfügt, könnte das System sogar für IoT-Projekte eingesetzt werden. So wäre es beispielsweise möglich, Obstsorten automatisch mit einem an den Raspberry Pi angeschlossenen Roboterarm (wie etwa dem Joy-IT-Bausatz Roboterarm „Grab-it", ELVshop Artikel-Nr. 133285) zu sortieren.

Ultraschall-Entfernungsmessung

Man kann sogar noch einen Schritt weiter gehen und das System mit einem Ultraschall-Entfernungsmesser ausrüsten. Damit ist es dann in gewissem Sinne sogar seinem menschlichen Gegenspieler überlegen. Zwar ist auch der Mensch in der Lage, Entfernungen aufgrund seiner Erfahrungen und des stereoskopischen Sehens mit zwei Augen abzuschätzen. Ein Ultraschallmodul kann jedoch Entfernungen von mehreren Metern zentimetergenau erfassen.

Hinsichtlich des Ultraschall-Entfernungsmessers ist zu beachten, dass dieser auch für den Betrieb mit 3,3 V ausgelegt sein muss. Alternativ kann ein konventioneller Wandler für 5 V verwendet werden. In diesem Fall wird jedoch zusätzlich ein 3,3 V/5 V-Pegelwandler in den Trigger-/Echo-Signalleitungen benötigt, da die I/O-Pins des Raspberry Pi nur für 3,3 V ausgelegt sind.

Wird der Eigenbau-Verstärker aus Bild 10 verwendet, kann der gesamte Aufbau in einem G418-Gehäuse (s. „Material") untergebracht werden. Das Titelbild dieses Beitrags zeigt einen Aufbauvorschlag dazu.

Das Programm

TFLite_detection_webcam_distance_speaker_1V0.py

enthält die zusätzlichen Anweisungen zur Auswertung des Ultraschall-Entfernungsmessers.

Nach der Initialisierung der Pins:

```
TRIG=18 #trigger pin of HC-SR04 ultrasound module
ECHO=24 #echo pin of HC-SR04 ultrasound module
GPIO.setmode(GPIO.BCM) #or GPIO.setmode(GPIO.BOARD)
GPIO.setup(TRIG, GPIO.OUT)
GPIO.setup(ECHO, GPIO.IN)
```

kann die Entfernung eines Objektes in Zentimetern über die Routine

```
def distance():
    GPIO.output(TRIG, 1)
    time.sleep(0.00001)
    GPIO.output(TRIG, 0)
    while GPIO.input(ECHO)==0:
        start _ time=time.time()
    while GPIO.input(ECHO)==1:
        stop _ time=time.time()
    return (stop _ time-start _ time)*340/2*100 #calculate distance from time
```

berechnet werden. Dabei wird die Schallgeschwindigkeit in Luft (340 m/s) verwendet:

```
return (stop _ time-start _ time)*340/2*100 #calculate distance from time
```

Der Faktor 2 ergibt sich aus der Tatsache, dass die Hin- und Rücklaufzeit zum Objekt berücksichtigt werden muss.

Nach der Erfassung und Erkennung eines Objekts kann auch dessen Entfernung über eSpeak akustisch ausgegeben werden. Hierbei ist zu beachten, dass hier immer das Objekt mit der größten Reflexionsfläche die Entfernungsangabe dominiert. So werden zwar kleine Objekte im Vordergrund durchaus vom KI-System korrekt erkannt, der Ultraschallsensor liefert jedoch die Entfernung des Hintergrundobjektes dazu. Mittels schärfer gebündelter Ultraschallquellen ließe sich hier eine gewisse Verbesserung erzielen. Allerdings ist eine detaillierte Ultraschallidentifizierung einzelner Gegenstände aus prinzipiellen Gründen kaum möglich.

Aus diesem Grund kommen bei autonomen Fahrzeugen auch überwiegend Laser-Scanner zum Einsatz. Diese erlauben eine tatsächliche dreidimensionale Abbildung der Umgebung. Die Entfernungsinformation muss also nicht mehr zusätzlich gewonnen werden, da sie direkt vom Lasersystem geliefert wird.

Für einfache Anwendungen, etwa im Bereich der Sehhilfen, ist ein mit preisgünstigen Ultraschallsensoren versehenes Gerät allerdings durchaus praktikabel. Aber auch in der Automatisierungstechnik und in der Robotik kommen Anwendungen, die die Vorteile der KI-gestützten Bilderkennung und der Ultraschall-Entfernungsmessung kombinieren, immer häufiger zum Einsatz.

Die komplette „See-and-Talk-Box" ist auf einem YouTube-Video in Aktion zu sehen [4].

Fazit und Ausblick

In diesem Beitrag wurde gezeigt, dass es selbst mit Einplatinenrechnern wie dem Raspberry Pi möglich ist, fortgeschrittene KI-Anwendungen umzusetzen. Mit den geeigneten Softwarebibliotheken können so Objekterkennungsverfahren genutzt werden, die vor wenigen Jahren noch komplexe Hochleistungsrechner ausgelastet hätten. Das frei verfügbare MSCOCO-Dataset erlaubt es einem Raspberry Pi sogar, mit angeschlossener Kamera 80 verschiedene Alltagsobjekte sehr präzise zu erkennen.

Zusammen mit einer Sprachausgabe können die erkannten Objekte auch akustisch ausgegeben werden. Es entsteht eine See-and-Talk-Box, die beispielsweise für sehbehinderte Menschen eine wertvolle Hilfe sein kann. Wird zusätzlich noch ein Ultraschall-Entfernungsmesser integriert, so kann sogar die Entfernung der Objekte präzise erfasst und angegeben werden.

Im nächsten Beitrag wird die allgemeine Objekterfassung zu einer Gesichtserkennung erweitert und ausgebaut. Damit lassen sich nicht nur Gesichter im Blickfeld einer Kamera erkennen, vielmehr ist es sogar auch möglich, einzelne Gesichter bzw. Personen zu identifizieren.

Ob sich diese Technologien als Fluch oder als Segen für die Menschheit entpuppen werden, muss die Zukunft erst noch zeigen. Allerdings kann es nur von Vorteil sein, wenn sich so viele Menschen wie möglich mit diesen Themen intensiver befassen, da sie nur so in der breiten Öffentlichkeit richtig eingeschätzt werden können. ELV

Material	Artikel-Nr.
Raspberry Pi 4 Model B, 8 GB RAM	250567
Kunststoff-Element-Gehäuse G418	030452
Ultraschallmodul HC-SR04	122121
Eventuell elektronische Kleinteile und ein Breadboard (s. Bild 10) oder eine Aktivbox	

i Weitere Infos

[1] Download-Paket zu diesem Beitrag: Artikel-Nr. 252461
[2] ELVjournal 5/2021, KI-Praxis III - Handschrifterkennung: Artikel-Nr. 252233
[3] ELVjournal 6/2021, KI-Praxis IV - Spracherkennung und Sprachsynthese: Artikel-Nr. 252343
[4] Video auf YouTube, See-and-Talk-Box in Aktion: https://www.youtube.com/watch?v=4ZotNeC77Yc

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