Eyetracker zum Selbstbau:

Einsteigermodell und Profiversion – nutzbar mit Scratch bis zu anspruchsvoller maschineller Bildverarbeitung mit openCV

Der Eyetracker basiert auf einer sehr einfachen, preiswerten Endoskop-Kamera mit USB-Anschluss und kann in zwei Varianten gebaut werden: für Einsteiger ohne Lötarbeiten und 3D-Druck oder als verbesserte Version unabhängig vom Umgebungslicht mit Infrarot-Filter und 3D-gedrucktem Gehäuse.

benötigtes Material: ca 20-25 € **Zeitaufwand: 45min-3h je nach Variante**

- Endoskopkamera mit USB-Anschluss (ca 8 €; zB Versandhandel)
- Sicherheitsbrille mit Gläsern, die ausgetauscht werden können (z.B. Conrad Art.-Nr. 886256)
- 15-20 cm Kupferdraht 2,5mm2 (zB aus NYM3 Verlegeleitungen, Baumarkt)
- Kabelbinder

verbesserte Version zusätzlich:

- 3mm IR-LED 940nm (zB. Conrad Art.-Nr. 154380)
- Widerstand 150 Ohm
- kleines Stück belichteter Analog-Film
- 3D-Druck des Gehäuses (2 Teile)
- Epoxid-2K-Kleber
- Schrumpfschlauch, Durchmesser ca. 8mm, 20cm Länge

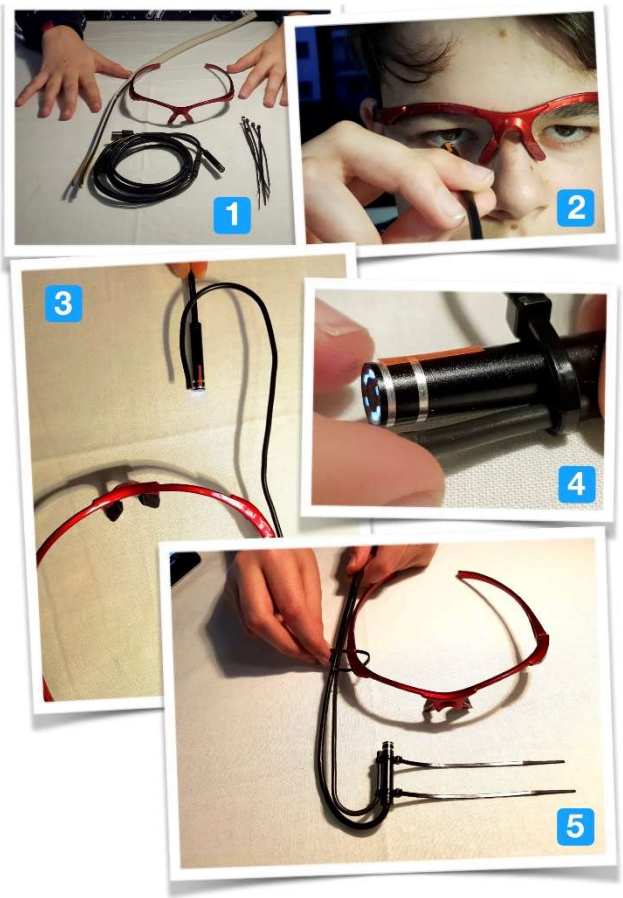
Werkzeug für die verbesserte Version:

Lötkolben, Cuttermesser, Seitenschneider, optimal Heißluftgebläse (zum Schrumpfen)

Einleitung

Eyetracker beobachten mit einer Kamera das Auge und erfassen die Bewegungen des Augapfels und das Blinzeln der Augenlieder. Entgegen unserer alltäglichen Wahrnehmung erfolgt diese Bewegung sprunghaft (Sakkaden) und nur in den Momenten, in denen ein Punkt fixiert wird, kann eine bewußte Wahrnehmung erfolgen. In welcher Reihenfolge und mit welcher Dauer einzelne Elemente eines Bildes oder einer realweltlichen Szene angesehen werden, gibt wichtige Hinweise auf deren Bedeutung und die Informationsverarbeitung im Gehirn. Die Auswertung der Augenposition erfolgt dabei automatisiert. Hochwertige, professionelle Systeme für Forschungsanwendungen können oft beide Augen gleichzeitig erfassen und bieten eine hohe Abtastrate, so dass auch schnelle Bewegungen erfasst werden können. Die Bandbreite der Anwendungsmöglichkeiten ist entsprechend groß und reicht von medizinischen Themen über Informationsverarbeitung von Reizen bis hin zu einfacheren Marketingstudien und Optimierung von Webseiten, Werbeflächen und Produktpräsentationen.
Die in dieser Anleitung vorgestellten Möglichkeiten, einen eigenen Eyetracker zu bauen und für Projekte zu nutzen, erreichen nicht die hohe Abtastrate (in erfassen Bildern pro Sekunde, fps), sind aber für Schülerinnen und Schüler einfach nachzubauen und auch in eigenen Programmen nutzbar. So können z.B. Spiele oder Roboter mit einer grafischen Programmierung in Scratch gesteuert werden – oder man verwendet openCV für anspruchsvollere Projekte.

Variante für Einsteiger



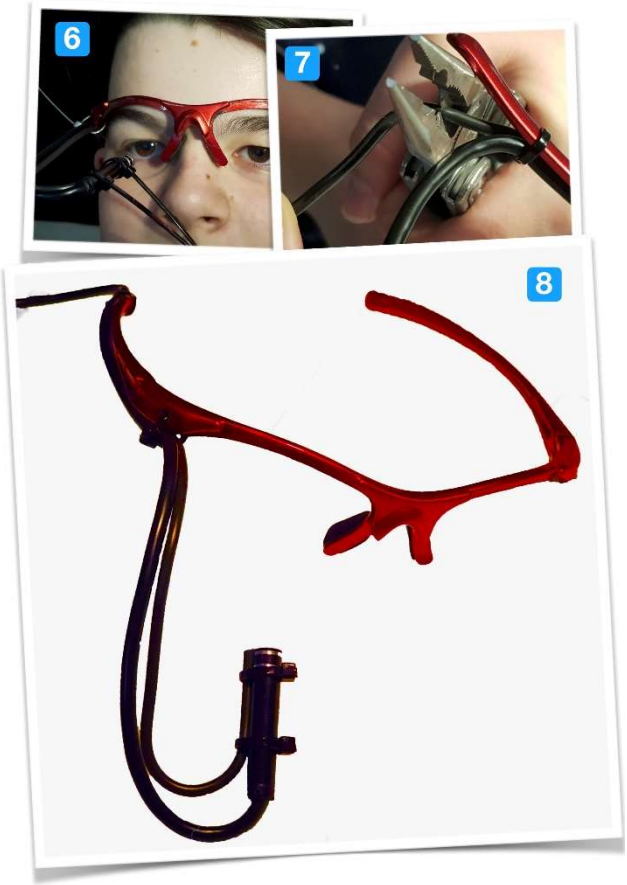
1 Übersicht über benötigtes Material

2 Position der Endoskop-Kamera zum Auge festlegen:
Kamera sollte direkt vor der Pupille sitzen:
- seitliche Abweichung (Abstand von der Nase)
- Höhe (Abstand von den Augenbrauen) beachten

3 Entfernung der Endoskop-Kamera zum Auge in dieser Position optimieren (großes Bild vom gesamten Auge in der Videovorschau am Computer)

4 Endoskop-Kamera so drehen, dass das vom Auge übertragene Bild aufrecht ist, dann Oberseite mit Klebestreifen markieren. Anschließend isolierten Kupferdraht mit Kabelbinder befestigen.

5 Kamera am Brillengestell befestigen: Draht und USB-Kabel gemeinsam mit Kabelbindern befestigen



6 Eyetracker aufsetzen, vorsichtig mit den Kabelbindern – letzte Korrektur der Position (Endoskop-Kamera direkt vor der Pupille, Auge vollständig und möglichst groß im Videobild)

7 Kabelbinder mit einer Zange ganz fest anziehen, überstehenden Draht abschneiden … nicht das USB-Kabel 😉

8 fertiger Eyetracker:
mit Hilfe eines Drahtes befestige Endoskop-Kamera an dem Gestell einer Schutzbrille ohne Gläser

Der nun fertige Eyetracker ist eine USB-Kamera, die sich an Windows, Mac und Linux-Computern (z.B. Raspberry Pi) als Kamera anschließen lässt. In Scratch oder openCV kann damit nun die Augenposition ausgewertet werden.

Variante für Fortgeschrittene

1 benötigtes Material

2 Mit einer Zange vorsichtig das zylinderförmige Gehäuse der Endoskop-Kamera entfernen.
Das Bildwandler-Modul sitzt vorne senkrecht auf der Platine.

3 Das kleine Kameramodul vorsichtig von der Platine lösen (Klebetropfen hält es in Position). Die dünnen Leiterbahnen nicht beschädigen. Für den Einbau in das 3D-gedruckte Gehäuse muss der Klebetropfen mit einem Cutter entfernt werden.
Auf dem Kameramodul sitzt eine winzige kreisrunde Platine mit den weißen LEDs zur Beleuchtung – diese bitte entfernen und die Leiterbahnen durchtrennen.

4 Linse aus dem Kameramodul herausdrehen – der IR-Sperrfilter schimmert rötlich.

5 Mit einem Cutter vorsichtig den IR-Sperrfilter (Rückseite der Line) entfernen. Bei veränderter Beleuchtung schimmert der Filter grünlich

6 3D-Drucke erstellen, Ausschnitt für Kühlkörper ggf anpassen (Platine hat eine silberne Fläche, die recht warm wird. Zur Sicherheit soll die per Kühlkörper abgeleitet werden).
Ein Stück belichteten Films auf die Kameraöffnung des 3D-gedruckten Deckels kleben.

7 IR-LED in den Deckel einkleben, Anschlüsse der LED mit den zusätzlichen Kabeln (blau = Kathode, gelb = Anode) vom der Anschlussleitung verbinden. Sie können von der Platine abgelötet werden.
Anschließend alles im Gehäuse festkleben.
Neben dem Anschlusskabel einen Draht zur Stabilisierung mit einkleben.

8 Kamera-Kabel am Gehäuse zur Leuchtregulierung durchtrennen, Schrumpfschlauch über Kabel und Draht ziehen und verschrumpfen, anschließend am Brillengestell befestigen.
USB-Kabel und Kamerakabel ohne die Leuchtregulierung wieder verbinden (Widerstand für die LED zwischen gelb und +5V einlöten, blau mit GND verbinden).

Diese Variante ist aufwändiger, weil das kleine Kameramodul um 90° gedreht werden muss, damit alles flach und seitlich aus dem Blickfeld herausgeführt werden kann.
Es ist auch etwas schwierig, die IR-LED und die Kameraelektronik in dem kleinen Gehäuse unterzubringen.

Bitte beachten:

Die IR-LED beleuchtet das Gesicht und das Auge aus großer Nähe. Der Vorwiderstand und die LED sind so gewählt, dass die LED nicht mit voller Leistung leuchtet. Trotzdem gilt folgender

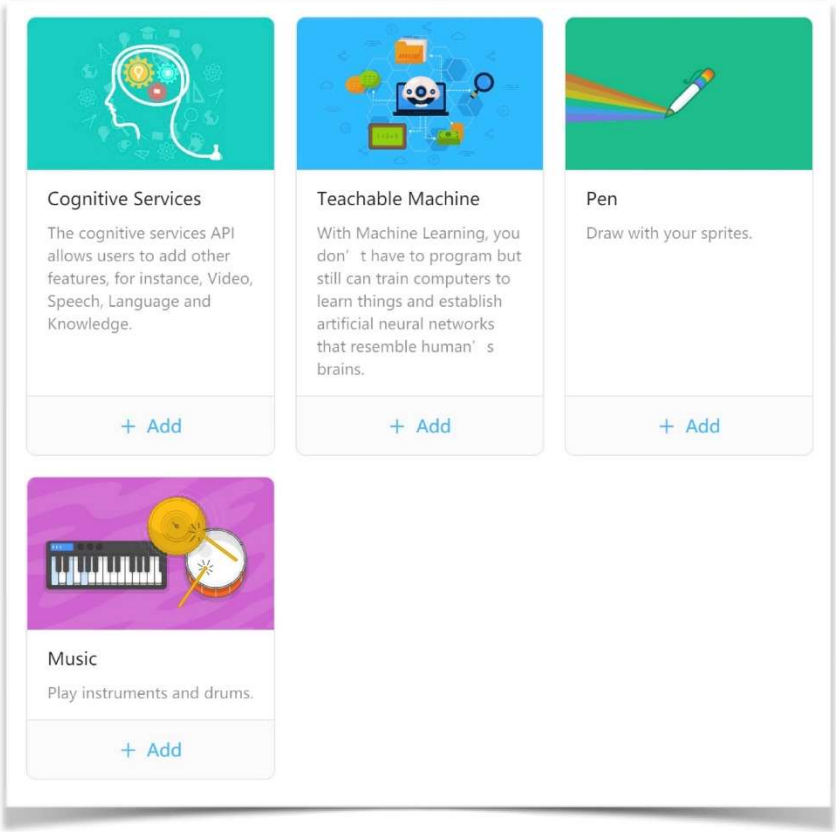
Warnhinweis:
Die Benutzung dieser Anleitung und die Umsetzung der darin enthaltenen Informationen erfolgt ausdrücklich auf eigenes Risiko. Verein und Autor können für etwaige Unfälle und Schäden jeder Art, die sich beim Nachbau oder der Nutzung ergeben, aus keinem Rechtsgrund eine Haftung übernehmen. Haftungsansprüche gegen Verein und Autor für Schäden materieller oder ideeller Art, die durch die Nutzung oder Nichtnutzung der Informationen bzw. durch die Nutzung

fehlerhafter und/oder unvollständiger Informationen verursacht wurden, sind grundsätzlich ausgeschlossen. Rechts- und Schadenersatzansprüche sind daher ausgeschlossen. Die Anleitung inklusive aller Inhalte wurde unter größter Sorgfalt erarbeitet. Verein und Autor übernemen jedoch keine Gewähr für die Aktualität, Korrektheit, Vollständigkeit und Qualität der bereitgestellten Informationen. Druckfehler und Falschinformationen können nicht vollständig ausgeschlossen werden. Verein und Autor übernehmen keine Haftung für die Aktualität, Richtigkeit und Vollständigkeit der Inhalte der Anleitung, ebenso nicht für Druckfehler. Es kann keine juristische Verantwortung sowie Haftung in irgendeiner Form für fehlerhafte Angaben und daraus entstandenen Folgen vom Verein bzw. Autor übernommen werden.

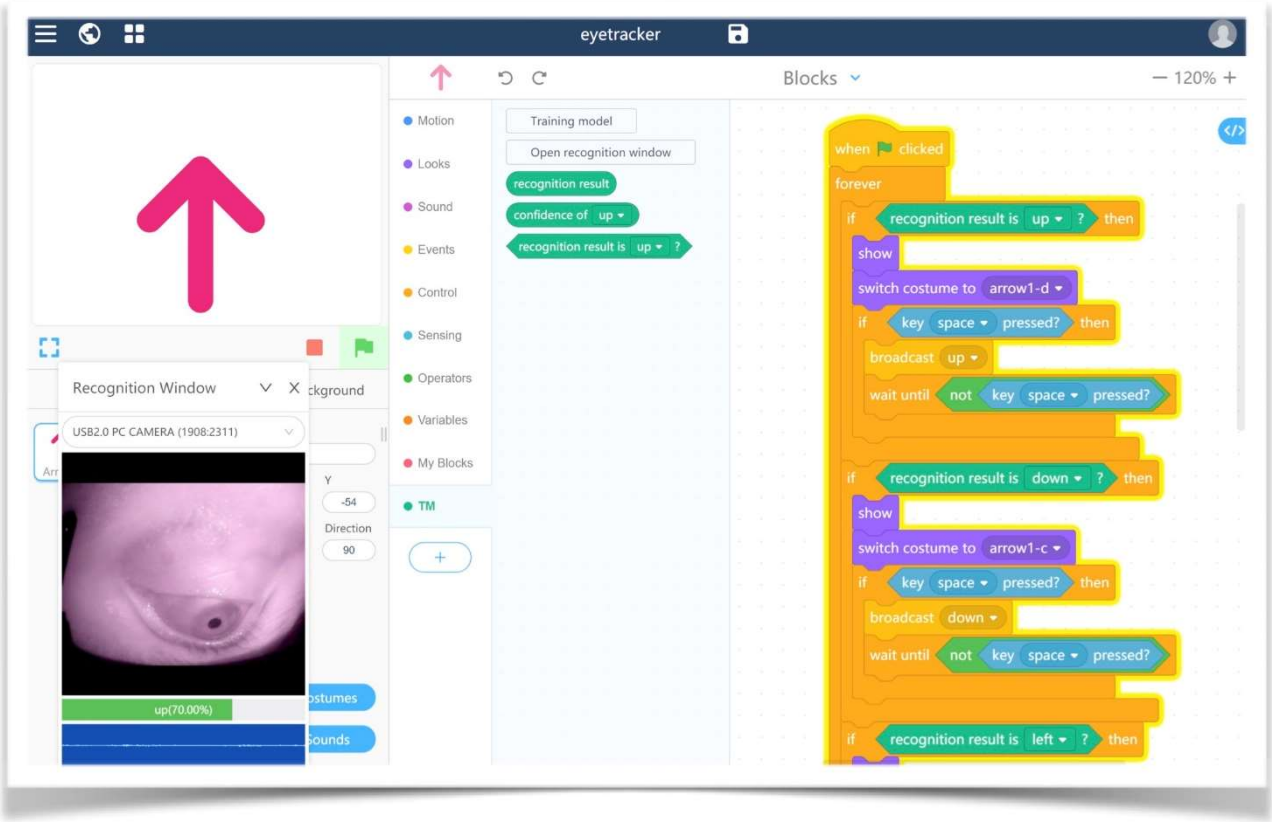
Programmierung mit Scratch

Scratch ist eine grafische Programmiersprache, die sich hervorragend für den Einstieg mit jungen Schülerinnen und Schülern eignet, z.B. im Übergang Grundschule-weiterführende Schulen. Die Programmiersprache wird vom MIT entwickelt und Anfang 2019 in der dritten Version (Scratch 3) veröffentlicht. Über Erweiterungen können zusätzliche Funktionen implementiert werden, z.B. für die Verarbeitung von Sprachbefehlen oder von Bildern.
Der chinesische Robotikhersteller Makeblock bietet eine auf Scratch3 basierende Programmierumgebung an, in die sich sehr leicht eine maschinelle Bildverarbeitung integrieren lässt. Die Software kann unter folgendem Link kostenlos installiert werden:
https://www.makeblock.com/software/mblock5

Ergänzen Sie dann die Erweiterung zum Trainieren von neuronalen Netzen ("Teachable Machine"):



Anschließend können Sie im Bereich Sprites in Scratch das neuronale Netzwerk in Ihre Programmierung einbinden. Erstellen Sie ein Modell mit 5 Ausgängen, wenn Sie die Blickrichtungen oben/unten, rechts und links für eine Roboterfahrt nutzen möchten (Blick geradeaus als 5. Kategorie – keine Fahrtrichtung vorgegeben).
Für die Ansteuerung des Roboters benutzen Sie Nachrichten zwischen Sprites und dem Roboter-Programmierbereich:



Wenn der Eyetracker an der rechten Seite montiert ist, ist das Bild der Kamera um 180° gedreht – zu sehen in der oberen Abbildung. Hier spielt das keine Rolle, da der Trainingsalgorithmus die Bilder an sich verarbeitet, unabhängig von der Lage des Auges. Für die Auswertung mit Python und openCV kann das Bild einfach gedreht werden.
In der Scratch-Programmierung ist nur ein Ausschnitt gezeigt, weil sich die Struktur wiederholt: Falls eine Richtung erkannt wurde, wird als Sprite (Costume/Kostüm) ein Pfeil in die entsprechende Richtung angezeigt. So bekommt der Benutzer ein visuelles Feedback, welche Bewegungsrichtung erkannt wurde. Wenn dann die Leertaste gedrückt ist, während eine Richtung erkannt wurde, erfolgt per Nachricht (Broadcast) die Information der Blickrichtung an andere Programmteile – z.B. für die Fahrtrichtung eines Roboters. Blickt der Benutzer geradeaus, wird nichts angezeigt und ein Stop-Befehl übertragen.

Objektorientierte Programmierung mit Python und openCV

In Python können Sie die Kamera wie jedes Video-Device ansprechen. Die weitere Verarbeitung der Bilddaten ist dann mit wenigen Zeilen Code schon möglich… hier ein Beispiel, um die Pupille zu finden:

```
1.  # -*- coding: utf-
2.
3.  # openCV importieren
4.  import cv2
5.  # wird in dem Beispiel nicht benutzt, aber für weitere Experiemten sinnvoll...
6.  import numpy as np
7.
8.  # 1. Videodevice (USB) verwenden; bei mehreren Kameras die korrekte Nummer auswählen
9.  cap = cv2.VideoCapture(0)
10.
11. while True:
12.
13.     video = False           # abwarten bis ein Bild eingelesen wurde
14.     while video == False:
15.        video, frame = cap.read()  # aktuellen Frame einlesen
16.     # hier kann die weitere Verarbeitung der Augenbewegung erfolgen ;-)
17.     # als Beispiel: Suche nach Pupille (kreisförmige Struktur)
18.
19.     #Eingangsbild um 180 Grad drehen
20.     (h, w) = frame.shape[:2]
21.     center = (w / 2, h / 2)
22.     M = cv2.getRotationMatrix2D(center, 180, 1.0)
23.     frame = cv2.warpAffine(frame, M, (w, h))
24.
25.     #Bild nach Grau wandeln, Rauschen entfernen, Helligkeit anpassen
26.     gray = cv2.cvtColor(frame,cv2.COLOR_RGB2GRAY)
27.     blur = cv2.medianBlur(gray, 5)
28.     cl1 = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(3,3))
29.     clahe = cl1.apply(blur)
30.     # Kantenfinder canny edge
31.     pupille = cv2.Canny(clahe,50,200)
32.
33.     #Kreise im Bild finden: Radius zwischen 10 und 30
34.     circles = cv2.HoughCircles(blur ,cv2.HOUGH_GRADIENT,1,20,param1=50,param2=30,minRadius
    =10,maxRadius=30)
35.     #nur wenn überhaupt ein Kreis erkannt wurde
36.     if circles is not None:
37.       circles = np.round(circles[0, :]).astype("int") #Fließekomma-
    Werte in Integer wandeln
38.       print ('integer',circles)   # Ausgabe zur Kontrolle, z.B. Anpassung der Radien
39.       for (x,y,r) in circles:
40.           # alle gefundenen Kreise einzeichnen im Ausgangsbild
41.             cv2.circle(frame, (x, y), r, (0, 0, 255), 2)
42.     cv2.imshow('Frame', frame)  #Ursprungsbild mit gefundenen Kreisen
43.     cv2.imshow('Auge', pupille)  #Canny-Edge zeigen
44.     cv2.imshow('blur', blur)    #entrauschtes Bild anzeigen
45.     if cv2.waitKey(1) & 0xFF == ord('q'):
46.       break
```