OpenCV:

Was ist OpenCV?

Eine in C/C++ geschriebene freie Programmbibliothek mit Algorithmen für die Bildverarbeitung und maschinelles Sehen. Es existiert seit 18 Jahren und ist 200 MB groß

Wie ist OpenCV aufgebaut?

Es hat eine modulare Struktur. Das heißt, es existieren mehrere Bibliotheken in einem Paket, welche alle verschiedene Funktionen beinhalten:

Kernfunktionalität:

Einfache Datenstrukturen wie multidimensionale Arrays und standard-Funktionen

Bildverarbeitung:

Lineare und nicht-lineare Bildfilterung

Geometrische Bildbearbeitung

Farb-Raum-Abhängigkeit

Video:

Videoanalyse-Tools wie Bewegungseinschätzung, Hintergrundeliminierung oder Objekterkennungs-Algorithmen

Calib3d:

Mehrsicht-Geometrie-Algorithmen

Features2d:

Erkennung hervorstechender Teile

Objektdefinierung

Highgui:

Einfache Benutzeroberflächenfähigkeiten

Video I/O:

Schnittstelle für Videoaufnahme und -codierung bzw. -decodierung

Gpu:

Grafikbeschleunigte Algorithmen von verschiedenen Modulen

Warum OpenCV?

Da es neben C++ und Java auch mit Python funktioniert, es am verbreitetsten am Markt ist und auch die meiste Entwicklungszeit in diese Bibliothek investiert wurde

Wie funktioniert das System von OpenCV?

Ein aufgenommenes Bild wird von einem Python-Programm in eine gewisse Anzahl von Rechtecken unterteilt, welche wiederum in ein Array gepackt werden. Änderungen werden dementsprechend erkannt und interpretiert

Was kann mit OpenCV erkannt bzw. gemacht werden?

Gesichter, Gesten, Objekte erkennen

Eigenbewegungseinschätzung

Segmentierung sowohl im 2D- als auch im 3D-Bereich

Maschinelles Lernen (Entscheidungsbäume, Klassifizierung, …)

Vor- und Nachteile:

API-Konzept:

Schnittstelle zur Anwendungsprogrammierung

Das Gesteninterpretationsprogramm:

Es werden die Gestenbibliothek cv2, numpy (Programmbibliothek zur Handhabung von Vektoren) und math (für Berechnung von Konturen) als Header benötigt.

Mit cv2.VideoCapture(0) wird die Aufnahme gestartet

In der Endlos-Schleife wird bei jedem Durchlauf überprüft, ob die Aufnahme beendet ist

Dann wird eine Form erstellt, welche je nach Durchlaufszahl einen der vier Sektoren umrahmt

Dann wird das Bild mit dafür von OpenCV zur Verfügung gestellten Funktionen in Helligkeitsstufen unterteilt

Dadurch können Körper erkannt werden. Es wird derjenige Körper fixiert, welcher am handähnlichsten erscheint.

Um diese wird eine Kontur gelegt. Die mittlere Bewegungsgeschwindigkeit von Formpunkten dieser Kontur wird aufgezeichnet und je nachdem, ob diese in die Ecke der Aufnahme geht oder nicht wird die Sektorenkennzahl oder eine 0 ausgegeben.

Was zeichnet die Programmierung in Python aus?

Knapp und gut lesbar

Objektorientiert

Strukturierung durch Einrückung

: statt {}

Kein Semikolon

Wie wird Bildmaterial verarbeitet?

Die einzelnen Bilder der Aufnahme werden eingelesen, dieses wird dann in Graustufen unterteilt. Diese Graustufen werden zur benötigten Unterscheidung von der Hand zum Hintergrund in schwarz oder weiß zerteilt. Dann werden die überbleibenden Weiß-Stellen untersucht und der Bereich, welcher am ehesten einer Hand ähnelt, wird herausgenommen und weiter behandelt.

Kann OpenCV auch mit anderen Rechner-Plattformen arbeiten?

Ja, durch OpenCL

APDS-9960:

Warum dieser Sensor?

Preiswert

Entspricht Anforderungen (Reichweite von 10 bis 20 cm, erkennbare Gesten in alle sechs Richtungen, Lichtabhängigkeit, Detektivität)

Kann Umgebungslicht und RGB-Licht aufnehmen

Abstandserkennung

Gestenerkennung

Arbeitet mit 3,3 V (Versorgung über Arduino)

I2C-Interface

Einfach zu entwickeln (kann mit Arduino kommunizieren, da Header für C/C++-Programmierung beigelegt ist

Liest alle 30ms neue Daten ein

Gestenerkennung, Abstandsmessung und Lichterkennung werden über Zustandsmaschine gesteuert

Was für Alternativen hätte es gegeben?

ZX-Gestensensor: Dies kann jedoch nur links-rechts-Bewegungen erkennen und Abstände messen, daher hätten wir in von oben herabschauend installieren müssen.

Skywriter HAT: hat nur 5 cm Reichweite

MGC3x30 GestIC: arbeitet mit elektrischem Feld. Hat aber, bis auf den Fakt, dass es Gesten genauer erkennen kann, keine Vorteile und eine geringe Reichweite bei hohem Preis

Warum kann man mit diesem Sensor nur genau diese Gesten erkennen?

Grund ist die Struktur gesture\_data\_type im Header, welche nur Daten in 32er-Arrays in alle sechs Richtungen eingibt, dazu noch ob Schwelle überschritten wurde und Anzahl an Gesten

Gibt es Möglichkeiten, andere Gesten zu implementieren?

Ja, da die Richtungsdaten in 32-Bit-Arrays gespeichert werden, welche theoretisch in Richtungsvektoren umgewandelt werden können.

Jedoch lässt dies der Header nur schwer zu, da nicht klar ersichtlich ist, wie die enum auf zusätzliche Gestentypen erweiterbar ist, wodurch wir uns dafür entschieden haben, stattdessen einfach den Sensor um 45° zu versetzen

Ich probierte zwar ein Auslösen dieser Geste unter der Bedingung, dass d\_data und r\_data der Struktur gesture\_data\_type einen gewissen Array-Wert überschreiten, aber auch das funktionierte nicht.

Nein, aber:

Sensitivität ist bearbeitbar

Zeitbereich bis zur nächsten Detektion (30, 60, 120 oder 240 ms)

Der Header:

Wie funktioniert ein Infrarotsensor allgemein?

Erkennt Infrarotstrahlung (0,7-1 µm Wellenlänge)

Eine IR-LED gibt einen Strahl derartiger Wellenlänge aus. Dieser wird von Objekten reflektiert und kommt dementsprechend zu den Photodioden zurück, welche neben Abstand auch Bewegung aus dem rückkehrenden Strahl herausnehmen können. Diese Informationen kommen von er Phasenverschiebung des rückkommenden Strahls.

Wie kommuniziert der Sensor mit dem Arduino?

Über ein Arduino-Programm (C-basiert), in welchem der Header vom APDS-9960 inkludiert ist. Die Aufnahmen des Sensors werden über I2C an das Arduino-Board weitergegeben

Gestensteuerungsprogramm:

Programmierung allgemein:

Konstruktor: wird aufgerufen, wenn Klasse oder Struktur erstellt wird.

Objekt:

Funktion:

Objektorientierung:

Klasse:

Android-App:

Wie kann die WiFi-Verbindung über die App aktiviert werden?

Wie kann über die App mit einem Netzwerk verbunden werden?

Text für Präsentation:

Folie 1:

Meine sehr geehrten Damen und Herren, sehr geehrter Herr Abteilungsvorstand und sehr geehrtes Notenpräsidium. Ich trete heute zur Diplomarbeitsverteidigung als Teil meiner Reife- und Diplomprüfung an. Wie Sie sehen können war ich Teil von Reflecton. Wir haben es uns als Ziel gesetzt, ein neuartiges Smart Mirror-System von Grund auf zu entwickeln. Mein Teil des großen Ganzen war hierbei einerseits die Entwicklung einer Android-App zur Steuerung und Individualisierung des Spiegelinhalts und andererseits die Entwicklung eines Bedienkonzeptes, mit welchem ohne großen Aufwand am Spiegel Detailansichten der einzelnen Applikationen geöffnet werden können.

Folie 2:

Dabei werde ich auf folgende Punkte zu sprechen kommen. Zuerst werde ich über die Herangehensweise sprechen. Das heißt, eine Auflistung von möglichen Bedienungstypen vor dem Spiegel und warum diese für unsere Anwendung mehr oder weniger passend sind. Danach spreche ich über OpenCV. Ein erstes Resultat meiner Nachforschungen, welches jedoch nicht die endgültige Lösung war. Darauffolgend komme ich zur finalen Lösung, dem Infrarotsensor. Dabei werde ich über die Funktionsweise des Sensors und das zugehörige Interpretationsprogramm sprechen. Zu guter letzt spreche ich noch über die Android-App, welche ebenso wie die iOS-App zur Konfiguration der Spiegelinhalte dient.

Folie 3:

Doch zuerst will ich kurz zusammenfassen, wie der analytische Weg hin zur Gestensteuerung grob ausgesehen hat. Die erste Überlegung war es, eine Tastatur oder eher noch ein zweites Display, welches eine Tastatur beinhaltet und mit Touchscreen funktioniert, zu implementieren. Dies hätte zwar die Bedienung erheblich vereinfacht, jedoch wäre ein zusätzliches Gerät sowohl unpraktisch als auch unästhetisch in Verbindung mit der Räumlichkeit, in der das Endprodukt verbaut wird. Ein weiterer Gedankengang war es, den Spiegel selbst mit einem Touchscreen zu versehen. Dabei kamen jedoch die Probleme auf, dass dies Verschmierungen zur Folge hätte und ein Touchscreen durch ein halbdurchlässiges Spiegelglas grundsätzlich nicht funktionieren könnte. Darum dachte ich weiter und kam auf die Idee, eine Sprachsteuerung zu verwenden. Da diese Technologie jedoch noch nicht zur Gänze ausgereift ist hinsichtlich Dialektik und Tonlagen-Erkennung, habe ich beschlossen hierin keine Zeit zu investieren, da der Eigenaufbau einen Riesenaufwand bedeutet hätte. Schlussendlich kam ich auf die Gestensteuerung. Sie hat den Vorteil, dass sie in unserer Anwendung aufgrund der wenigen unterschiedlichen Befehle verschiedene Gesten ohne große Schwierigkeiten auseinanderhalten kann und diese auch weltweit gleich sind, sodass Dialektik wegfällt. Des Weiteren ist sie einfach in den Spiegel integrierbar, da nur ein kleines Aufnahmegerät wie ein Sensor oder eine Kamera benötigt wird. Damit kommen wir nun auch zum ersten Entwicklungsversuch.

Folie 4:

Da mir vom vorjährigen HWE-Projekt noch ein Raspberry Pi 3B inklusive NoIR-Kamera zur Verfügung stand, fokussierte ich mich in erster Linie darauf, diese für diesen Zweck ebenfalls zu verwenden. Auf der Suche nach einer dementsprechend geeigneten Gestenbibliothek wurde mir schnell klar, dass über 90% derartiger Anwendung mit der auf Python basierenden Bibliothek OpenCV realisiert wurden. OpenCV kann neben den von uns benötigten Handgesten auch Gesichter, Gegenstände, Bewegung im Raum oder ganze Körper erkennen und interpretieren. Dazu muss lediglich ein Python-Programm erstellt werden, in welchem die Header-Datei von OpenCV inkludiert ist. In diesem Programm ist eine while(1)-Schleife der Hauptdarsteller. In dieser Endlos-Schleife wird durchgehend innerhalb des Kamerabildes zwischen vier Bereichen gewechselt. Das heißt, zu einem gewissen Zeitpunkt achtet das Programm bspw. auf den Sektor linksoben, 50 ms später auf den Sektor rechtsoben. So kann jede Wischgeste in einen der vier Sektoren innerhalb von maximal 200 ms erkannt werden. schneller ist es leider nicht möglich, da es nicht möglich ist, vier Bereiche gleichzeitig zu fokussieren oder zumindest habe ich nach eingehender Suche nicht herausgefunden wie und der Erkennungsprozess ohnehin bis zu 50 ms in Anspruch nimmt. Ein weiterer kleiner Nachteil dieses Systems ist, dass die Wischgeste ungefähr von der Aufnahmenmitte ausgehen muss, um erkannt werden zu können. Wenn sie, für den Fall einer Wischgeste nach linksoben oder rechtsoben, 10-20 cm weiter oben beginnt, könnte die Hand aus dem Kamerabild verschwinden, bevor die Wischgeste erkannt wird, um dieselbe Distanz weiter unten würde sie in einen anderen Sektor wandern, der eine Geste in diese Richtung nicht erkennen würde. Ein weiteres Problem, das im Vorhinein noch nicht bedacht wurde, ist die Störung der Privatsphäre bei der Aufnahme mittels Kamera. Gerade im Badezimmerbereich, einem Hauptanwendungsbereich des Produkts, würde sich wohl so ziemlich jeder unwohl fühlen, wenn eine Kamera andauernd das Geschehen aufnimmt. Daher begab ich mich wieder auf die Suche in der Hoffnung, eine humanere Lösung zu finden.

Folie 5:

Und das gelang mir auch. Hier sehen Sie den nun im Spiegelrahmen auf der unteren Vorderseite verbauten Sensor. Er nennt sich APDS-9960 und ist ein Infrarotsensor. Der schwarze Quader inmitten der oberen Befestigungslöcher nimmt die Infrarotstrahlung von umliegenden Körpern auf und gibt die Informationen über eine I2C-Schnittstelle über die an den unteren Kontakten befestigten Leitungen an einen Mikrocontroller, in diesem Fall ein HTL UNO-Board, weiter. Dieses verarbeitet dann innerhalb eines Arduino-Programms, in welchem die Header-Datei dieses Sensors inkludiert ist, die ankommenden Daten und interpretiert Bewegungen von erkannten Körpern. Dabei gibt es vordefinierte Richtungsgesten in beide Richtungen aller drei Dimensions-Achsen. Da wir jedoch Gesten in die vier Diagonalen brauchten, um die Applikationen in den Ecken zu öffnen, wurde der Infrarotsensor um 45° versetzt eingebaut. Dies war auch nötig, weil die Gestenbibliothek nur über großen Aufwand erweiterbar gewesen wäre. Ich habe es probiert, indem ich mit einem Logic-Analyzer die Datenfolgen der einzelnen Gesten analysierte und ähnliche in die Header-Datei implementierte. Jedoch ohne Erfolg. Dennoch ist dieses Modell die bestmögliche Lösung, da sie günstig in der Anschaffung ist und gerade genug kann, um unseren Anforderungen gerecht zu werden.

Folie 6:

Kommen wir nun zur Android-App. Diese besteht aus vier Fenstern. Beim Öffnen der App kommt man zur Start-Anwendung, wie hier bei diesem Screenshot vom Emulator auch zu sehen ist. Hier kann man seine Benutzerdaten eingeben, um sich dann über den Sign in-Button einzuloggen oder auf Wanna Join? klicken, um sich zu registrieren. Dabei kommt man zur Registrierungs-Anwendung, in welcher man seine E-Mail bei unserer Datenbank gemeinsam mit einem Passwort, welches den klassischen Kriterien entsprechen muss, anmelden. Über beide Wege kommen Sie als Nutzer daraufhin zur Widgets-Anwendung. In dieser können Sie die Anordnung der Widgets Ihren Wünschen anpassen. Rechtsunten in dieser Anwendung befindet sich ein Zahnrad, welches über klicken zur Einstellungen-Anwendung führt. In dieser können Sie WLAN auf Ihrem Handy ein- bzw. ausstellen und sich mit einem Netzwerk verbinden.

Folie 7:

Vielen Dank für Ihre Aufmerksamkeit.