  
HTL - Perg

Höhere Abteilung für EDV und Organisation

Diplomarbeit

Mushroom Identifier

Projektteam: Jakob Froschauer

Hakan Abbas

Markus Arbeithuber

Projektbetreuer: Prof. Dipl.-Ing. Christian Aberger

Bearbeitungszeitraum: 01.10.2016 – 30.04.2017

**Eidesstattliche Erklärung**

Hiermit versichern wir, die vorliegende Arbeit selbständig, ohne fremde Hilfe und ohne Benutzung anderer als der von uns angegebenen Quellen angefertigt zu haben. Alle Stellen, die wörtlich oder sinngemäß aus fremden Quellen direkt oder indirekt übernommen wurden, sind als solche gekennzeichnet.

Perg, \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Unterschrift \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Name)

Perg, \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Unterschrift \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Name)

Danksagung

An dieser Stelle möchten wir uns bei allen Personen bedanken, die uns bei der Enstehung der Diplomarbeit unterstützen und uns zur Seite standen.

Besonderer Dank gilt unserem Diplomarbeitsbetreuer Prof. Dipl.-Ing. Christian Aberger, der uns die gesamte Projektdauer für technische und organisatorische Fragen zur Verfügung stand.

Herzlichen Dank!

[**Impressum** 4](#_Toc476208007)

[1. Schule 4](#_Toc476208008)

[2. Schuljahr 4](#_Toc476208009)

[3. Klasse 4](#_Toc476208010)

[4. Projektname 4](#_Toc476208011)

Impressum

1. Schule

HTBLA Perg für Informatik

Marchlandstraße 48

4320 Perg

1. Schuljahr

2016/2017

1. Klasse

5AHIF

1. Projektname

Mushroom Identifier

# Projektleiter

Jakob Froschauer

Projektteam

Hakan Abbas

Markus Arbeithuber

# Betreuungslehrer

Dipl.-Ing. Christian AbergerEinleitung

Kurzfassung

Die Diplomarbeit Mushroom Identifier ist während des fünften Jahrgangs von Hakan Abbas, Markus Arbeithuber und Jakob Froschauer im Zuge der Reife- und Diplomprüfung an der Technischen Bundeslehranstalt Perg erstellt worden.

Die mobile Anwendung soll die Zukunft des Pilz Lexikons darstellen. Bei der Suche im Wald begegnet man oft Pilzen, die man nicht ohne schwere, klobrige Pilz Lexika erkennen kann. Auch mit diesen Büchern ist es schwierig in vertretbarer Zeit den gesuchten Pilz zu identifizieren. Mit der App Mushroom Identifier soll dieses Problem der Vergangenheit angehören. Zur Funktionsweise: Der Benutzer wird dazu aufgerufen, ein Pilzfoto aus der Vogelperspektive auszuwählen oder ein neues zu schießen. Daraufhin wird unter anderem die Farbe und die Form des Pilzes erkannt. Auch auffällige Eigenschaften (zum Beispiel viele kleine Punkte in einem großen roten Kreis => Fliegenpilz) werden automatisch erkannt.

Die dazu nötigen Vergleichsdaten werden lokal mit der App mitgeliefert um das Problem des schlechten Internet Empfangs im Wald zu umgehen.

Mit jeder erkannten Eigenschaft verringert sich die Zahl der in Frage kommenden Pilze. Wenn am Ende der Bilderkennung noch kein Pilz feststeht werden die Unterschiede der noch in Frage kommenden Pilze durch Ja/Nein Benutzerfragen abgefragt. Darüber hinaus wird auch durch Maschinelles Lernen festgestellt, ob es sich überhaupt um einen Pilz handeln kann. Wir sind davon überzeugt, dass durch diese Applikation:

1.

2. die Identifikation von Pilzen so leicht und frustfrei wie noch nie von statten geht. Außerdem hat man dieses "Lexikon" im Gegensatz zu der Bücherform immer in der Hosentasche eingesteckt.

Motivation

Das gesamte Projektteam sind begeisterte Pilzsammler, jedoch trauten wir uns nur bei dem Eierschwammerl auch zuzugreifen, da der Identifikationsprozess von Pilzen bisher ausgesprochen aufwendig ist. Schwere Pilzlexika mussten mitgeschleppt werden. Dieser aufwendige Prozess soll vereinfacht werden. Ein einfaches Foto mit dem Smartphone und eventuell ein paar JA/NEIN Fragen zu beantworten soll reichen um Pilze eindeutig zu identifizieren.

Darüber hinaus wird durch die modernen Technologien ein jüngeres Publikum angesprochen, das sich am Pilzsammlern begeistern kann und somit die Vorzüge

Technische Grundlagen

Technologien

Java (Android)

Swift/Objective-C (IOS)

C++ (OPEN CV)

Android Studio (Android)

XCode (IOS)

Visual Studio (OPEN CV)

Ist eine von dem Unternehmen Microsoft aangebotenen Entwicklungsumgebung für verschiedene Hochsprachen (darunter C, C++, C#, Phyten, HTML, JavaScript und Typescript)

Für unsere Tätigkeiten wird C++ mit der Programmbibliothek OPENCV verwendet.

OpenCV (Open Source Computer Vision)

Diese ist eine in C++ geschriebene Bibliothek, welche ursprünglich von Intel entwickelt wurde, inzwischen jedoch quelloffen unter eine BSD-Lizenz entwickelt wird. Sie umfasst unter anderem Algorithmen für 3D-Funktionalität Gesichtsdetektion, verschiedenste Filter (z. B. Gauß). Mit ihr werden Applicationen erstellt, die sich allgemein mit Computer Vision beschäftigen

Computer Vision

Beschreibt die computergestützte Lösung von Aufgaben, die sich an den Fähigkeiten des menschlichen visuellen Sehens orientiert. Wird bei der Automatisierungstechnik bei der Qualitätssicherung, bei Radarfallen bis hin zum selbstfahrenden Auto und in der Sicherheitstechnik eingesetzt.

Mensch-Computer-Interaction

Aufgetretene Probleme

1. Offizieller OpenCV Installationsguide ist veraltet und aus jetziger Sicht sehr inkorrekt <http://docs.opencv.org/2.4/doc/tutorials/introduction/windows_install/windows_install.html>

Lösung: aktuellere Guides gesucht gefunden => Problem: nicht für die aktuellste Version=> Guide für meine Version angepasst.

1. In OpenCV wird nicht der RGB (Standard) Farbraum, sondern der BGR Farbraum verwendet => führte zu vermeintlich falscher Ergebnisse => Tipp in Forum führte mich zu der Lösung

<http://stackoverflow.com/questions/8932893/accessing-certain-pixel-rgb-value-in-opencv>

1. Der HSV Farbraum ist in OpenCV nur bis 180 (um in ein uchar zu passen) gehend, normale Farbraumumrechner (Word, GIMP, Photoshop, Internetrechner) rechnen mit 360 => Lösung => eigenen Farbraumumrechner für OPENCV gesucht und gefunden

<http://www.shervinemami.info/colorConversion.html>

1. Im HSV Farbraum ist die Farbe Rot (z. B. Fliegenpilz) 2-Geteilt=>

Lösung 1. Zuerst linken Farbraum untersuchen, dann rechten Farbraum untersuchen dann beide Bilder zusammenfügen

Lösung 2. Man kann über H Wert 181=1 und so weiter => geht allerdings nur bis 255 (um in uchar zu passen)

1. Fliegenpilze (eigentlich rund) sind mit weißen Flecken (auch teilweise an den Rändern übersäht, diese weiße Flecken werden nicht miterkannt und bei der Untersuchung nach Runden Figuren nicht als Rund erkannt => PROBLEM
2. Nicht gewusst, dass Circle Transformation den Canny selbstständig durchfürht. Davor habe ich ihn selbstständig davor gemacht

Der Identifiaktionsprozess besteht aus insgesamt drei Phasen:

Die Bilderkennung

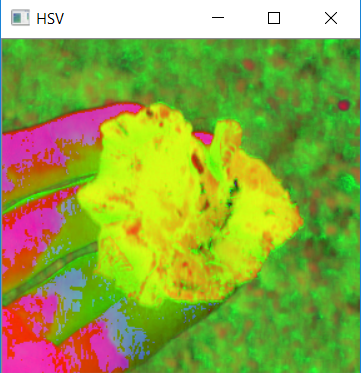
Das maschinelle Lernen

Benutzerfragen

Funktionsweise Bilderkennung

Zuerst wählt der Benutzer auf dem Smartphone einen quadratischen Bereich aus, in dem sich der Pilz befindet. Dann wird in diesem Quadrat ein Bereich im Zentrum nach der Farbe untersucht um. Der Mittelwert aus den erkannten Farben wird daraufhin mit den Daten aus dem XML File verglichen. Alle Pilze, die eine Ähnliche Farbe besitzen kommen in die nähere Auswahl.

Sollte die Farbe einzigartig sein (wie z. B. beim Grünspantäuschling) kann es vorkommen, dass der Erkennungsprozess ab diesem Punkt abgeschlossen ist.

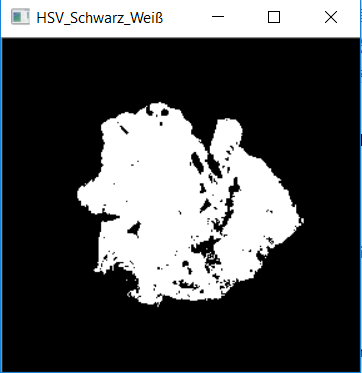
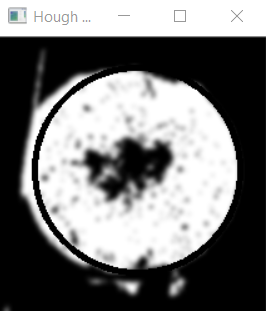


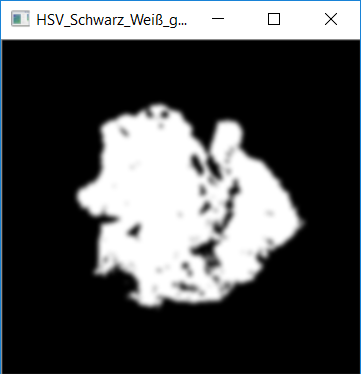
Wenn dem nicht der Fall ist wird das Bild in den HSV Farbraum konvertiert, da dieser wesentlich unempfindlicher für verschiedene Lichtquellen ist als der RGB/BGR-Farbraum.

Daraufhin wird nur die Farbe des Pilzes herausgefiltert, sodass der Pilz weiß und der Rest Schwarz dargestellt wird.

Der Pilz wird daraufhin noch Weichgezeichnet und auf diesem Bild wird ein Kreiserkennungsalgorithmus durchgeführt.

Dazu werden mit dem Canny Edge Detector bearbeitet. Und mit dem Ergebnis nach Kreisen gesucht. Bei dem Eierschwammerl werden (zurecht) keine Pilze gefunden. Bei dem immer runden Ruchsigen Rötelritterling wird dagegen zurecht ein Kreis erkannt.





Verwendete Funktionen

imread("C:\\Users\\Jakob\\Documents\\Visual Studio 2015\\Projects\\OPENCV31\\data\\falscher\_pfifferling.jpg");

//umwandlung von BGR in HSV

cv::cvtColor(image, hsv\_image, cv::COLOR\_BGR2HSV);

inRange(hsv\_image, Scalar(mushlist[i].hsv\_v[0], mushlist[i].hsv\_v[1], mushlist[i].hsv\_v[2]), Scalar(mushlist[i].hsv\_b[0], mushlist[i].hsv\_b[1], mushlist[i].hsv\_b[2]), hsv\_first);

// Gasusscher Weichzeichner

GaussianBlur(src\_gray, src\_gray, Size(9, 9), 2, 2);

Canny(detected\_edges, detected\_edges, lowThreshold, lowThreshold\*ratio, kernel\_size);

HoughDetection(src\_gray, src\_gray, 99, 41);

//AUSGABE//////////////////////////////////////////////////////////////////////////////

namedWindow("Pilz\_ohne\_Aenderung", WINDOW\_AUTOSIZE); // Pilz ohne Änderung

int detectAndDisplay(Mat frame); //Maschinelles Lernen; Fliegenpilzerkennung

vector<Pilz> readxml(); //Lesen der PilzXML

void CannyThreshold(int, void\*); //Umrisse werden erkannt

vector<Pilz> oneornull(vector<Pilz> mushlist2, wstring question); // 1/0 Entscheidungsfragen

vector <Pilz> roundornot(vector <Pilz> mushlist, int amountofcircles); //ist der Pilz Rund oder nicht?

vector <Pilz> questions(vector <Pilz>mushlist); //Ausführliche Entscheidungsfragen

int HoughDetection(const Mat& src\_gray, const Mat& src\_display, int cannyThreshold, int accumulatorThreshold); //Hough Circle Detection