HTL Perg für EDV & Organisation

Machlandstraße 48,

4320 Perg,

Österreich

Diplomarbeit

Mobile Face and Feature Detecting Photo Assessment

Eingereicht von: Simon Hinterholzer, Alexander Ortner – 5AHIF 2014/2015

Betreuungslehrer: Dipl.-Ing. Christian Aberger

Auftraggeber: Aberger Software GmbH

Eidesstattliche Erklärung

Perg am, _____

Hiermit erklären wir eidesstattlich, die hier vorliegende Diplomarbeit selbstständig und ohne fremde Hilfe verfasst und ausschließlich die angegebenen Quellen beziehungsweise Hilfsmittel benutzt zu haben und wörtliche oder inhaltliche Stellen aus jenen Quellen auch als solche erkenntlich gemacht haben.

Simon Hinterholzer	Alexander Ortner

Danksagung

An dieser Stelle möchten wir uns bei all jenen Personen bedanken, die uns bei der Erstellung dieser Diplomarbeit mit Rat und Tat zur Seite standen und uns unterstützten.

Primär gilt unser Dank unserem Professor Herrn Dipl.-Ing. Christian Aberger, der uns während der gesamten Dauer der Diplomarbeit unterstützte und viele Ideen und Lösungen für unsere Arbeit beisteuerte.

Weiters möchten wir uns auch bei unseren Familien bedanken, die uns diesen Weg ermöglicht haben und vor allem immer für uns dagewesen sind und uns eine große Hilfe waren.

Impressum

Schule

- HTBLA Perg für EDV und Organisation
- Machlandstraße 48
- 4320 Perg

Schuljahr

- 2014/2015

Klasse

- 5AHIF

Projekttitel

- MFFDPA
- Mobile Face and Feature Detecting Photo Assessment

Projektteam

- Simon Hinterholzer
- Alexander Ortner

Betreuungslehrer

- Dipl.-Ing. Christian Aberger

Inhaltsverzeichnis

1. Kurzbeschreibung

Aufgabe der Diplomarbeit war es, eine Applikation für das Betriebssystem iOS zu entwickeln. Diese soll in der Lage sein, aus einer Menge von Fotos die besten Bilder herauszusuchen, um diese dem Benutzer vorzuschlagen. Diese Fotos werden aufgrund bestimmter Kriterien (Schärfe, Kontrast, Gesichtserkennung) ausgewählt.
Der User hat nach der automatischen Auswahl der Fotos noch die Möglichkeit, von der Applikation gewählte Fotos zu löschen oder mit einem anderen Bild zu ersetzen.

Das angestrebte Ziel ist es, den User beim Auswählen seiner besten Fotos auf dem Smartphone zu unterstützen. Diese Fotos können dann direkt weiter verwendet werden zum Bestellen oder zum Gestalten von Fotodruckprodukten.
Mit ein paar einfachen Einstellungen zu Beginn, wählt die Applikation automatisch die qualitativ hochwertigsten Fotos aus dem ausgewählten Zeitraum aus.

Des Weiteren kann diese Applikation in andere Programme eingebaut werden, welche mit Bildern arbeiten. Als Beispiel kann ein Bestellsystem für Fotodrucke oder ähnliches herangezogen werden.

2. Abstract

The task of this diploma is to develop an application for the iOS operation system. This application should be able to pick out the best images from a set of photos and present it to the user. These photos are selected due to certain criteria like focus, contrast and face recognition.
After the automatic selection of the images the user is able to delete or replace a selected picture by another photo of the quantitative results.

Our desired target is to aid the user by choosing the greatest pictures on his smartphone. Then these photos can be used to order a photo print product.
With adjusting a few simple settings at the beginning this application automatically picks the best photos of high quality out of the selected timespan.

Furthermore this application can be included into other programs, which work with pictures like an ordering system for photo printing.

3. Stakeholder im Projekt

3.1. Das Team

3.1.1. Simon Hinterholzer

Persönliche Daten	
Name:	Simon Georg Franz Hinterholzer
Geburtsdatum:	12.September 1994
Adresse:	Winden 56
Ort:	4311 Schwertberg
E-Mail:	simonhinterholzer@hotmail.com

Erfahrungen		
Ausbildung:	2001-2005 Volksschule Schwertberg 2005-2009 Hauptschule Schwertberg 2009-2015 HTBLA Perg	
Praktika & Erfahrung:	Juli 2011	Praktikum bei der MERCKENS Karton- und Pappenfabrik GmbH
	Juli - August 2012	Praktikum bei der MERCKENS Karton- und Pappenfabrik GmbH
	Juli 2013	Praktikum bei der MERCKENS Karton- und Pappenfabrik GmbH

Fähigkeiten	
Sprachen:	Deutsch – Muttersprache Englisch – fließend in Wort und Schrift
EDV - Technologien:	Betriebssysteme: • Microsoft Windows • Microsoft Server 2012 R2 • Linux Sprachen: • C# • Java Sonstige: • SQL, PL/SQL • Oracle APEX • SAP • Microsoft Office

3.1.2. Alexander Ortner

Persönliche Daten	
Name:	Alexander Franz Ortner
Geburtsdatum:	13.November 1994
Adresse:	Nelkenweg 9
Ort:	4310 Mauthausen
E-Mail:	alexander.ortner@gmx.at

Erfahrungen		
Ausbildung:	2001-2005 Volksschule Mauthausen 2005-2009 Hauptschule Mauthausen 2009-2015 HTBLA Perg	
Praktika & Erfahrung:	Juli 2011	Praktikum bei der ENGEL Austria GmbH
	Juli 2012	Praktikum bei der GRZ IT Gruppe in Linz
	Juli 2013	Praktikum bei der ENGEL Austria GmbH
	Juli 2014	Praktikum bei der GRZ IT Gruppe in Linz

Fähigkeiten	
Sprachen:	Deutsch – Muttersprache Englisch – fließend in Wort und Schrift
EDV - Technologien:	Betriebssysteme: • Microsoft Windows • Microsoft Server 2012 R2 • Linux Sprachen: • C# • Java Sonstige: • SQL, PL/SQL • Oracle APEX • SAP • Microsoft Office

3.2. Motivation

Viele kennen das Problem. Man kommt vom Urlaub nach Hause und hat Unmengen an Fotos geknipst von denen einige mehr und andere weniger gut sind. Doch wie stellt man nun fest, welche der Fotos man auch gedruckt haben will? Wieder muss man sich entscheiden, welche der Bilder gelungen sind.

Der Fokus dieser Diplomarbeit liegt darauf, ein Programm zu entwickeln, welches in der Lage ist, anhand einer Bewertungsfunktion zu entscheiden, welche Fotos die besten Eigenschaften aufweisen. Diese werden daraufhin dem Benutzer vorgeschlagen.

Mit unserem Programm wollen wir vor allem jene Personen unterstützen, welche keine Lust beziehungsweise wenig Zeit haben, sich mit der Auswahl der besten Fotos aus einer großen Fotomenge auseinanderzusetzen.

3.3. Betreuungslehrer

Betreut wird unsere Diplomarbeit von Professor Christian Aberger, welcher zugleich seit der vierten Klasse unser vielgeschätzter Java Programmierlehrer ist. Durch seine hilfsbereite und sympathische Art konnte er uns immer wieder dazu antreiben, das Bestmögliche zu geben sowie das eine oder andere Mal unter die Arme zu greifen.

Persönliche Daten	
Name:	Christian Aberger
Adresse:	Softwarepark 37
Ort:	4232 Hagenberg
E-Mail:	christian@aberger.at

Zusätzlich zu seiner Arbeit als Professor in der HTL in Perg, ist er auch noch Geschäftsführer der Aberger Software GmbH, welche auch unseren Auftraggeber darstellt.

3.4. Auftraggeber

Abbildung 1: Aberger Sofware GmbH

Wie bereits erwähnt, ist unser Auftraggeber die Aberger Software GmbH mit Sitz in Hagenberg im Mühlkreis. Unser Ansprechpartner ist Herr Martin Brandner.

Kontakt	
Name:	Aberger Software GmbH
E-Mail:	office@aberger.at
Adresse:	Softwarepark 37
Ort:	4232 Hagenberg
Tel.:	+ 43 720 348 450
Web:	www.aberger.at

Gegründet wurde die Aberger Software GmbH im Jahre 1991, mit Standort in Maierhof als ein Ein-Mann-Betrieb. Im Herbst 2007 übersiedelte sie in das AMSEC-Gebäude in Hagenberg und operiert seitdem von dort aus. Die derzeitige Anzahl der Beschäftigten beläuft sich auf 13 Personen.

Die Aberger Software GmbH beschäftigt sich primär mit der Softwareentwicklung. Eine Entwicklung ist z.B. ein Fotobestellsystem, welches es ermöglicht ohne eine Softwareinstallation, Fotos vom eigenen Rechner schnell hochzuladen und diese in gedruckter Form zu bestellen.

4. Entstehung und Planung

4.1. Thema und Aufgabenstellung

4.1.1. Aktueller Zustand (Ist-Zustand)

Nach einem erholsamen Urlaub, bei dem man viel erlebt und dementsprechend auch viele Fotos mit dem Smartphone gemacht hat, steht man nun vor der großen Aufgabe, diese Fotos zu sortieren bzw. die besten Aufnahmen herauszusuchen, falls man diese in einem Fotobuch verewigen möchte. Somit muss jedes Foto einzeln begutachtet und mit anderen Bildern verglichen werden, um das Schönste herauszufiltern.

4.1.2. Problembereich

Durch diese aufwendige Arbeit wird sehr viel Zeit in Anspruch genommen, welche für wichtigere Aufgaben verwendet werden könnte.

4.2. Zielsetzung

4.2.1. Technische Ziele

Folgende technische Ziele sollen durch MFFDPA verfolgt werden:

- Unser Programm soll auf jedem iOS7 Gerät oder höher laufen.
  - Jedes iPhone mit iOS Betriebssystem
  - Auf jedem Tablet mit iOS Betriebssystem
  - Sowie auch auf dem iPod.
- MFFDPA soll einfach zu bedienen bzw. der Ablauf leicht verständlich sein.
- Weiters kann MFFDPA um weitere Funktionen wie z.B. Einbinden von GPS-Funktionen erweitert werden.
- Ein großes Ziel von uns ist jedoch die Verwendung neuer und uns noch fremder Technologien wie z.B. iOS oder OpenCV.

4.2.2. Generelle Ziele des Projektes

Generelle Ziele die von uns verfolgt werden:

- Die besten Bilder anhand folgender Kriterien auswählen.
  - Schärfe
  - Kontrast
  - Gesichtserkennung
- Es soll eine hohe Performance gewährt werden
- Der Ablauf soll so einfach wie möglich gehalten werden.
- Dem User soll so viel Arbeit wie nur möglich abgenommen werden.

4.3. Funktionsumfang

4.3.1. Überblick Gesamtsystem

In dem folgenden Use-Case Diagramm wird das Grundsystem grafisch dargestellt.
Zu beachten ist, dass die Schritte „Fotoabzüge bestellen“ bzw. der Schritt „Zahlung“ nicht von uns realisiert werden. Unsere Applikation dient lediglich der Beschaffung bzw. Sortierung der richtigen Fotos und fungiert zwischen diesen beiden Schritten.

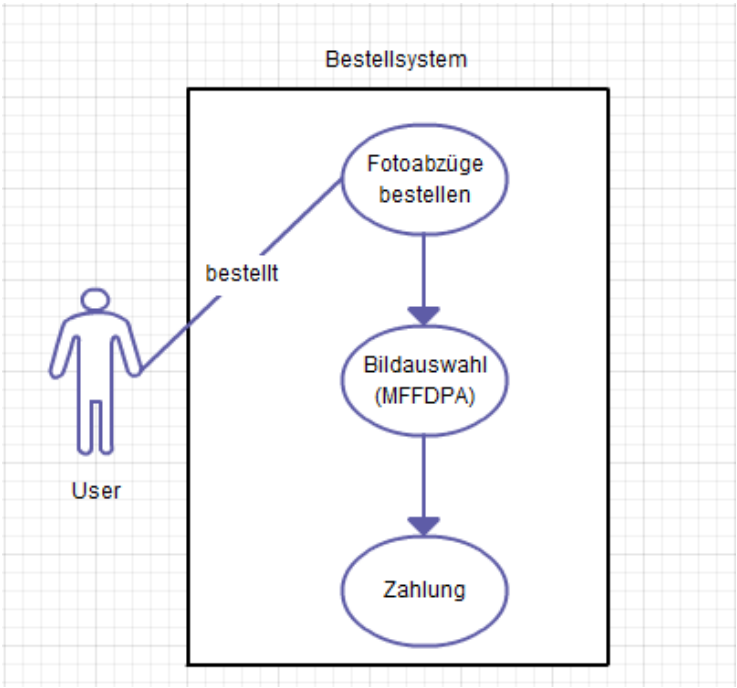

Abbildung 2: Überblick Gesamtsystem

4.3.2. Funktionalitäten MFFDPA

Das folgende Use-Case Diagramm stellt unsere Diplomarbeit in drei einfachen Schritten dar.

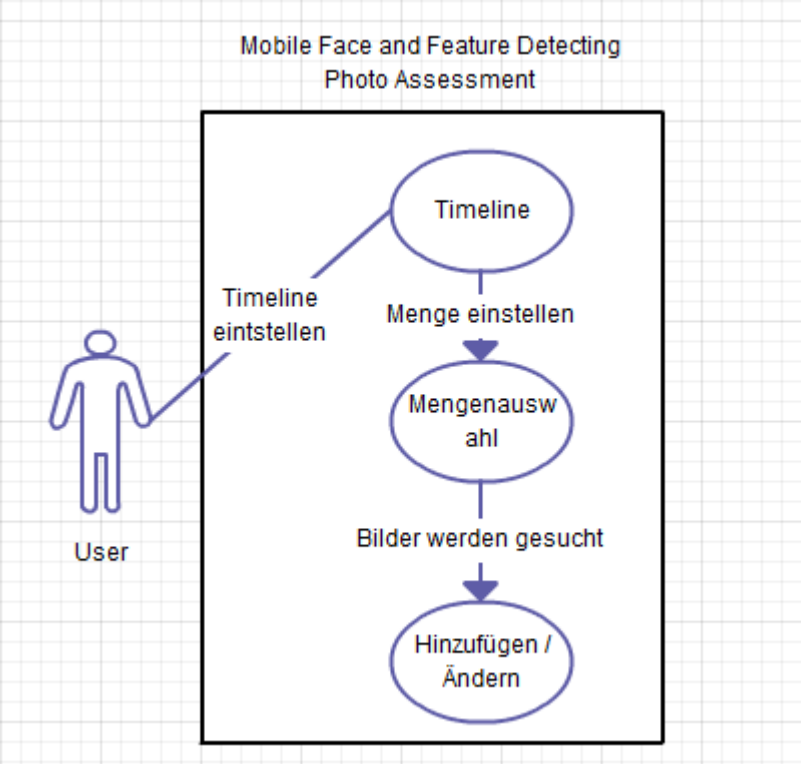

Abbildung 3: Funktionalitäten MFFDPA

4.3.2.1. Die Auswahl des Fotoordners anhand einer Timeline

Als erster Schritt werden alle Fotos, welche der User in seiner Galerie hat, in einer Timeline aufgelistet bzw. dargestellt. Der User soll nun die Möglichkeit haben diese Timeline auf einen bestimmten Zeitraum einzugrenzen. Diese Funktion realisieren wir anhand der Programmiersprache Objective-C. Hat der Benutzer seinen bevorzugten Zeitraum ausgewählt folgt der nächste Schritt.

4.3.2.2. Auswahl der Zielmenge anhand eines Schiebereglers

Im nächsten Schritt kann der Benutzer anhand eines Schiebreglers die bevorzugte Ergebnismenge an Fotos angeben die von unserer Applikation vorgeschlagen werden sollen.

4.3.2.3. Automatische Auswahl der Bilder

Nach dem Einstellen der gewünschten Ergebnismenge wird die automatische Bewertung der einzelnen Fotos gestartet. Jedes einzelne Bild wird durch die Kriterien Farbraum, Kontrast, Schärfe sowie Gesichtserkennung benotet. Diese Bewertung bzw. Begutachtung der einzelnen Bilder erfolgt durch die Library OpenCV. Jene Fotos, welche am besten bewertet werden, bekommt der User als Ergebnis präsentiert. Ein weiteres Kriterium ist die Verteilung der Fotos anhand von GPS-Koordinaten, damit am Ende nicht alle Fotos von nur einer Location sind.

4.3.2.4. Hinzufügen und Ändern einzelner Bilder

Im letzten Schritt hat der Benutzer noch die Möglichkeit einzelne Bilder entfernen bzw. durch andere ersetzen zu lassen. Soll ein Foto ersetzt werden, rückt das Bild mit der nächstbesten Bewertung in die Ergebnismenge.

4.3.3. Funktionalität Farbraum

Bei der Funktionalität des Farbraumes ging es in unserer Aufgabenstellung darum, ein Bild anhand seiner Farben zu bewerten.
Dabei galt, je mehr Farben auf dem Bild vorhanden sind, desto besser wird das Bild bewertet. Zur Bewertung nahmen wir jedoch nicht alle möglichen Farben, sondern reduzierten das Farbspektrum auf 256 mögliche Farben, da die Farben für das menschliche Auge relativ ähnlich sind und wir so bewusst auf eine nuancenhafte Abstufung verzichteten.

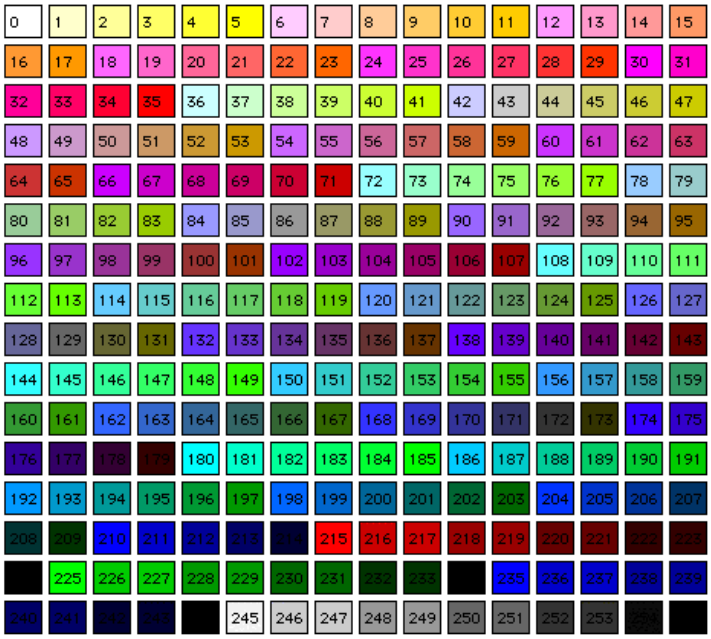

Abbildung 4: Farbspektrum mit 256 Farben

4.3.4. Funktionalität Kontrast

Als Kontrast bezeichnet man in der Fotografie die Unterschiede zwischen hellen und dunklen Bereichen eines Bildes.
Ob der Kontrast nun gut oder schlecht ist oder zu dem betreffenden Bild passt, liegt in diesem Fall im Auge des Betrachters wie man auf den folgenden Bildern erkennen kann.

Abbildung 5: Silhouttenartiges Bild durch verstärkten Kontrast

Abbildung 6: kontrastreiches Originalbild

Abbildung 7: Fades Bild durch reduzierten Kontrast

Da es äußerst schwierig ist den Computer dazu zu bringen den Kontrast eines Bildes zu bewerten, ob dieser zum Bild passt oder ob die Stimmung des Bildes dadurch besser wird, haben wir uns entschieden, bei der Berechnung einen höheren Kontrastwert besser zu bewerten als einen geringeren.

4.3.5. Funktionalität Schärfe

Bei der Schärfe gibt es in der Fotografie zwei verschiedene Definitionen, die geklärt werden müssen.
Zum einen hätten wir da die tatsächliche Schärfe, die auch physikalisch auf einem Bild vorhanden ist.
Zum anderen gibt es den sogenannten Schärfeeindruck, von dem man spricht, wenn das Bild „anscheinend“ scharf ist, dies jedoch nur so wirkt.

Abbildung 8: Absichtliche Unschärfe des Bildes

Im unserem Fall wollen wir auf die tatsächliche Schärfe schließen, da es äußerst schwierig wäre, dem Rechner beizubringen, was Menschen als „scharf“ empfinden, obwohl es nicht wirklich scharf ist.

4.3.6. Funktionalität Gesichtserkennung

Unter Gesichtserkennung versteht man die Analyse der sichtbaren Merkmale der Vorderseite des Kopfes.

Unterscheiden muss man hier, ob man das Gesicht, dass man erkennt, einer Person zuordnen will, sozusagen eine Identifikation, oder ob man darauf prüft, ob auf dem verarbeiteten Bild überhaupt ein Gesicht vorhanden ist.

Die Identifikation eines Gesichtes, sprich die Zuordnung von diesem, basiert auf den biometrischen Merkmalen eines Gesichtes, wie zum Beispiel dem Abstand der Augen und so weiter, und hat allerlei Möglichkeiten, wie sie uns helfen kann. Einerseits kann sie bei der Aufklärung von Verbrechen helfen, indem zum Beispiel Videos von Überwachungskameras ausgewertet werden und die Gesichter der Personen, die darauf zu sehen sind, mit Bildern von bereits vorbestraften Menschen verglichen werden und so schnell ermittelt werden kann, ob es sich beim Täter um jemanden handelt, der bereits aktenkundig ist, was die Ermittlung um einiges erleichtert. Andererseits ist diese Art von Gesichtserkennung auch in der Ahnenforschung hilfreich, da sie die Merkmale des Gesichtes erkennen kann und mit den Merkmalen anderer Gesichter vergleicht, was Aufschluss darüber geben kann, ob die zwei Personen miteinander verwandt sind.

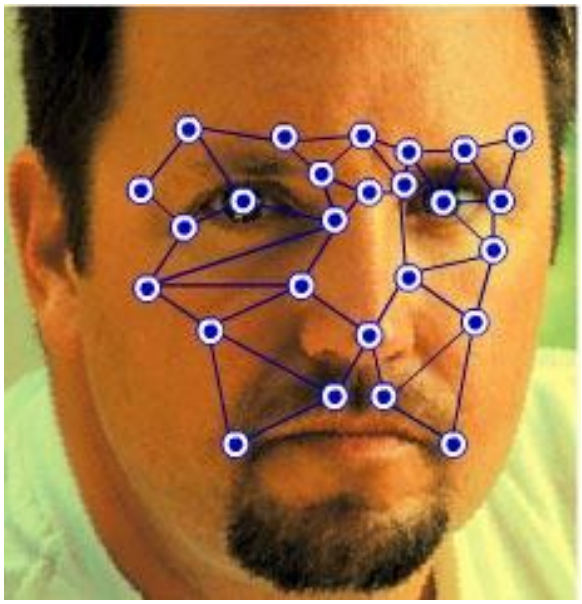

Abbildung 9: Identifizierung einer Person anhand biometrischer Merkmale

Die Überprüfung, ob überhaupt ein Gesicht auf dem Bild zu erkennen ist, nennt man Gesichtsdetektion. Diese Technologie findet ihre Anwendung zum Beispiel bei Bildsuchmaschinen, wobei mit Hilfe der Detektion überprüft wird, ob sich auf dem Bild überhaupt Gesichter befinden, da diese andernfalls verworfen werden, und schon seit einiger Zeit bei Digitalkameras, bei denen sofort am Display angezeigt wird, ob und wo sich auf dem Bild Gesichter befinden.

Abbildung 10: Beispiel der Gesichtserkennung

Umgesetzt werden kann Gesichtserkennung im zwei- oder auch im dreidimensionalen Raum.

Unsere Anwendung arbeitet im zweidimensionalen Raum, da dies die einzige Möglichkeit bei Bildern ist. Dabei werden vom Programm markante Punkte wie Augen, Nase, Kinn oder der Mund gesucht. Wurden diese Punkte gefunden, so geht das Programm davon aus, dass ein Gesicht gefunden wurde. Um ein Gesicht in 2D zu identifizieren, werden die Abstände zwischen den gefundenen Punkten herangezogen und mit bereits analysierten und gespeicherten Bildern verglichen.

Problematisch bei der Gesichtserkennung im zweidimensionalen Raum ist jedoch, dass sich das Ergebnis abhängig von Belichtung und anderen Störfaktoren verfälschen kann. Zum Beispiel kommt es vor, dass Gesichter von Personen mit längeren Haaren, die das Gesicht eventuell teilweise verdecken, nicht als solche erkannt werden.

Diese Probleme gibt es im dreidimensionalen Raum eigentlich nicht mehr, da hier viel mehr Faktoren berücksichtigt werden können.

So können hier beispielsweise die Erhebung bzw. die Länge der Nase mit in die Analyse einfließen, was es außerdem ermöglicht, dass die Gesichtserkennung im dreidimensionalen Raum nicht nur Frontalaufnahmen verarbeiten kann, sondern auch Gesichter entdecken und identifizieren kann, welche nur schräg oder von der Seite aufgenommen wurden.

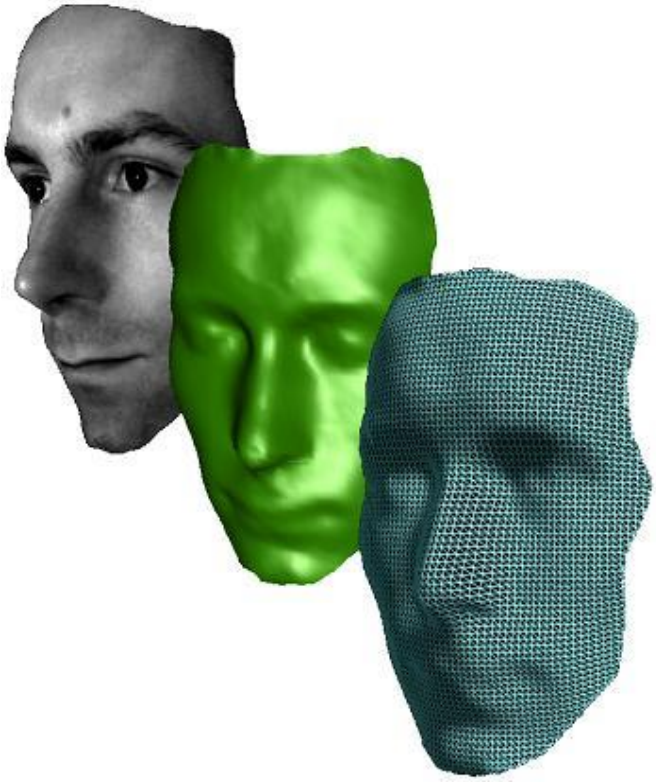

Abbildung 11: Beispielbild 3D-Gesichtserkennung

Ist dies der Fall, werden zum Beispiel das Kinn, die Nase und auch die Form des Ohrs zur Analyse und Identifikation genutzt.

4.3.7. Funktionalität GPS-Koordinaten

Jedes Foto welches geschossen wird speichert Unmengen von Daten, darunter auch GPS-Koordinaten. Diese Koordinaten sollen nun in unserer Applikation verwendet werden um bei der Auswahl der Fotos eine gleichmäßige Verteilung der verschiedenen Locations zu gewährleisten. Um diese Funktion zu realisieren sollen wir uns anhand dem Voronoi-Diagramm ein Beispiel nehmen.

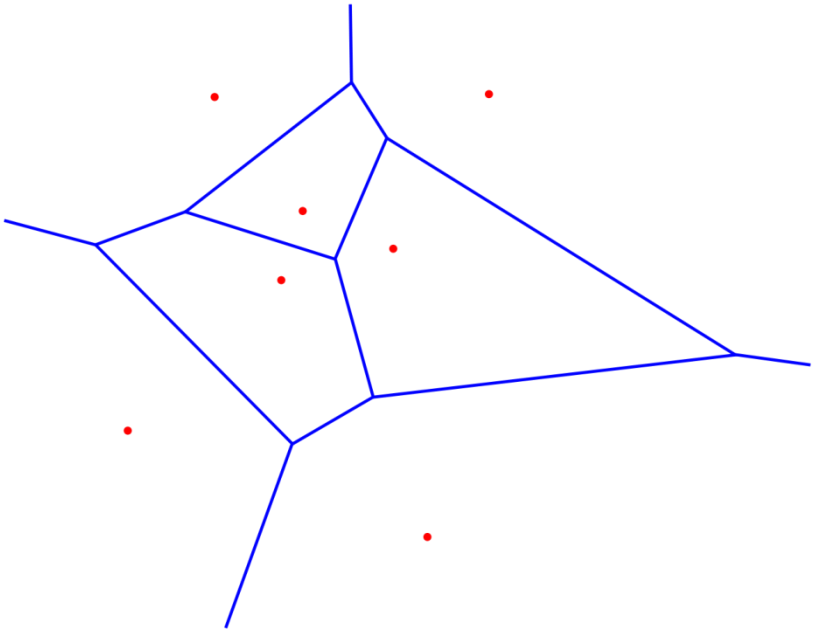

Abbildung 12: Voronoi-Diagramm

Wie in der Grafik oberhalb ersichtlich, wird der Raum in Regionen unterteilt. Diese Regionen werden auch Voronoi-Regionen genannt. Jede Region hat ein sogenanntes Zentrum, welches sich aus einer Menge von Punkten herausbildet. Die Region bildet sich im Anschluss dadurch, dass die einzelnen Punkte dem Zentrum der Region näher liegen als dem Zentrum einer anderen Region. Dadurch bilden sich die einzelnen Grenzen und die Region entsteht.

4.3.8. Grafische Benutzeroberfläche

Im Folgenden werden die einzelnen Teile der Grafischen Benutzeroberfläche (GUI) näher erläutert.

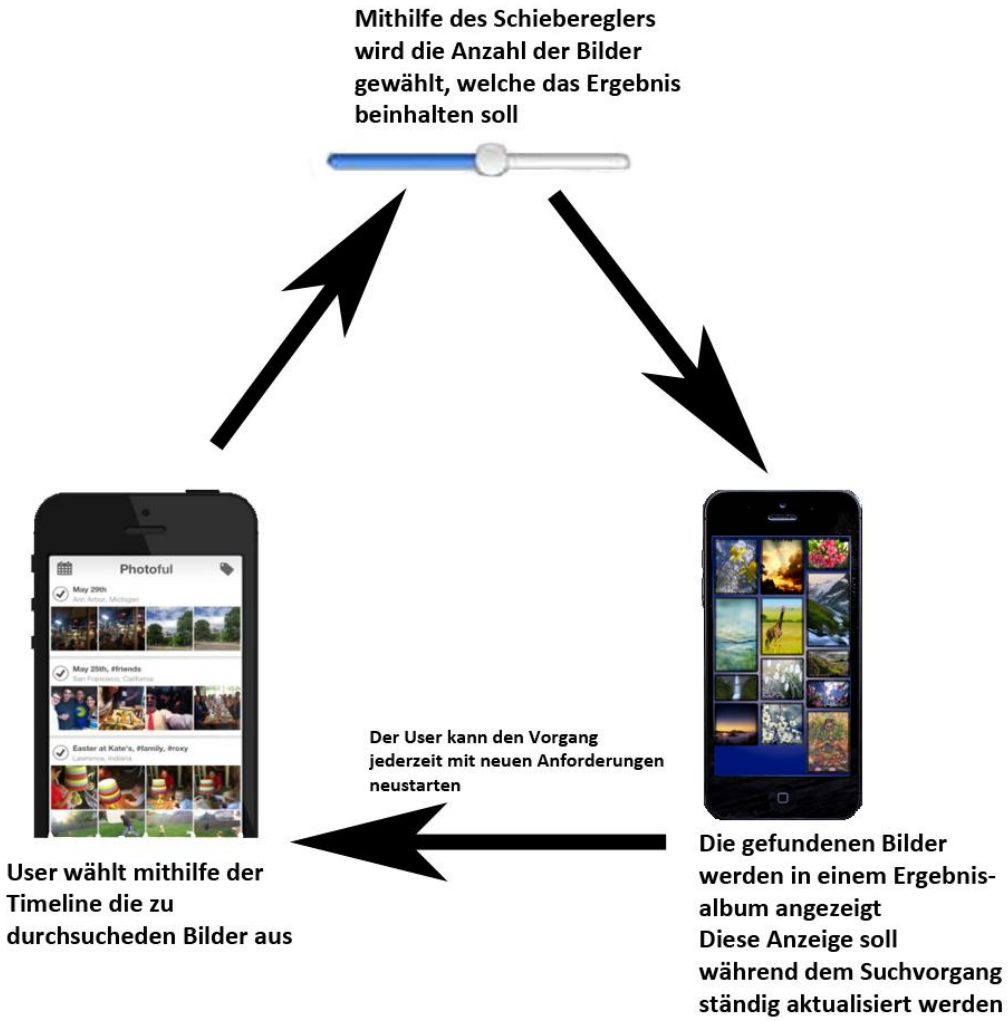

Abbildung 13: Grafische Benutzeroberfläche

4.3.8.1. TimeLine

Nachdem der User unsere Applikation gestartet hat wird ihm eine TimeLine mit allen seinen Fotos aus der Galerie dargestellt. Die TimeLine wird untergliedert in Jahre, Monate und Tage. Mithilfe der Zoomfunktion kann in die einzelnen Abschnitte hinein- bzw. herausgezoomt werden um den Zeitraum zu bestimmen welchen er für die Fotoauswahl verwenden möchte.

4.3.8.2. Slider

Sobald der Zeitraum gewählt wurde, muss als nächster Schritt die Ergebnismenge angegeben werden. Diese realisieren wir mittels eines Sliders.

Beispiel:
Der User hat den Zeitraum August und September ausgewählt. Dieser Zeitraum enthält 350 Fotos. Mittels des Sliders kann er nun eine Menge zwischen 0 und 350 bestimmen. Wählt er als Menge 50 Bilder aus werden im die fünfzig besten Bilder aus diesem Zeitraum herausgesucht.

4.3.8.3. Bild entfernen oder ändern

Im letzten Schritt wird dem Benutzer nun die automatisch bewertete Ergebnismenge präsentiert. Dieser kann nun die einzelnen Bilder begutachten und falls nötig die gelieferte Ergebnismenge anhand zweier zusätzlicher Funktionen bearbeiten.

Bild entfernen:
Das Bild wird aus der Ergebnismenge entfernt und es wird kein Ersatz Bild für das gelöschte eingefügt.

Bild ändern:
Das derzeitige Bild wird entfernt und stattdessen wird das nächstbeste Bild d.h. das Foto mit der nächstbesten Bewertung eingefügt.

4.4. Ressourcenplanung

4.4.1. Hardware

4.4.1.1. Smartphone (iPhone)

- Damit unsere Applikation genützt werden kann wird ein Gerät mit iOS-Betriebssystem benötigt. Vorranging haben wir uns auf das iPhone spezifiziert, mit welchem der Benutzer in der Lage ist sich aus einer Menge von Fotos die besten heraussuchen zu lassen. Die benötigte Mindestversion ist iOS7.

Abbildung 14: iPhone

4.4.1.2. Tablet (iPad)

- Weiters kann die Applikation auch auf dem iPad ausgeführt werden, was im Vergleich zum iPhone eine erleichterte Handhabung aufgrund der Größe des Bildschirmes hat. Ein großer Vorteil der für die Benutzung auf einem Tablet spricht ist das ein besserer Überblick gewährt wird. Somit kann unsere Applikation auf jedem beliebigen Tablet ausgeführt werden, sofern dieses ein iOS Betriebssystem (Version 7.x) hat.

Abbildung 15: iPad

4.4.1.3. iPod Touch

- Ein iPod wird im Grunde dazu verwendet um überall seine persönliche Lieblingsmusik mit dabei zu haben. Der von Apple entwickelte iPod Touch besitzt zusätzlich auch eine eingebaute Kamera um Fotos zu knipsen. Deswegen kann unsere Applikation auch auf dem iPod Touch ausgeführt werden sofern dieser die Mindestversion von iOS7 hat.

Abbildung 16: iPod Touch

4.4.1.4. MacBook Air

- Aufgrund der Tatsache, dass die Applikation für iOS ausgelegt ist, mussten wir einen Teil unserer Diplomarbeit auf dem MacBook Air entwickeln. Dies betraf vor allem den Teil der GUI sowie die Einbindung des C++ Codes für die Bildbewertung. Dieses Gerät bekamen wir von der Schule bereitgestellt, wodurch unsere Arbeit erheblich erleichtert wurde.

Abbildung 17: MacBook Air

4.4.2. Software

4.4.2.1. Visual Studio

- Visual Studio ist eine Entwicklungsumgebung für Hochsprachen und wurde von Microsoft entwickelt, sowie im Jahre 1997 veröffentlicht. Die aktuelle Version ist Visual Studio 2013.
- Wir haben die Entwicklungsumgebung vor allem verwendet, um die Bewertungskriterien der einzelnen Bilder (d.h. Schärfe, Kontrast, Gesichtserkennung) auszuwerten. Diese Methoden wurden in der Programmiersprache C++ unter Einbindung der öffentlichen Library von OpenCV von uns ausprogrammiert.

Abbildung 18: Visual Studio

4.4.2.2. XCode

- Die Entwicklungsumgebung XCode, welche von Apple herausgegeben wurde, wird vor allem für die Entwicklung von Programmen für „Mac OS X“ und „iOS“ verwendet. Dabei ist XCode vor allem für die beiden Programmiersprachen Swift und Objective-C entwickelt worden.
- Letzteres, d.h. Objective-C, kommt auch bei unserer Applikation zum Einsatz. Die gesamte Benutzeroberfläche sowie die Funktionen der Timeline bzw. des Schiebereglers sind in dieser Sprache implementiert.

Abbildung 19: XCode

4.4.2.3. OpenCV

- OpenCV ist eine freie Programmbibliothek, welche von Intel entwickelt wurde. Die darin enthaltenen Algorithmen sind primär für die Bildbearbeitung und für maschinelles Sehen ausgelegt. Ein großer Vorteil, der für die Nutzung von OpenCV spricht, ist die Geschwindigkeit sowie eine große Auswahl an Algorithmen welche aus neuesten Forschungsergebnissen entwickelt wurden.
- In unserer Applikation ist OpenCV vor allem für die Bewertung der Bilder zuständig. (Schärfe, Kontrast, Gesichtserkennung)

Abbildung 20: OpenCV

4.4.2.4. iOS (Betriebssystem)

- Das Betriebssystem iOS wurde von der Apple Inc. entwickelt und ist vor allem für mobile Geräte wie das iPhone, das iPad und den iPod gedacht. Die erste Version von iOS wurde im Juni 2007 zusammen mit dem iPhone veröffentlicht. Die aktuelle iOS Version ist 8.2 (Stand Februar 2015).

Abbildung 21: iOS

4.4.2.5. VMware-Workstation

- VMware Workstation, welches vom Unternehmen VMware entwickelt wurde, ist eine Software um Betriebssystem auf dem eigenen Rechner zu virtualisieren. Positiv an einer solchen Virtualisierung ist vor allem, dass damit nicht der eigene Rechner gefährdet wird. Negativer Aspekt dieser Methode ist vor allem die verminderte Schnelligkeit dieser virtualisierten Betriebssysteme.
- Zu Beginn der Programmierung hatten wir ein Mac-OS Betriebssystem auf einer VMware virtualisiert, welches jedoch aufgrund der fehlenden Leistung unserer Rechner eine enorme Belastung, sowie sehr langsam war. Aufgrund dessen stiegen wir dann auf ein MacBook Air um, dass von der Schule zur Verfügung gestellt wurde.

Abbildung 22: VMware Workstation

4.5. Vorgehensplan

Der folgende Plan gibt eine grobe Übersicht, wie wir bei der Erstellung der Diplomarbeit vorgegangen sind.

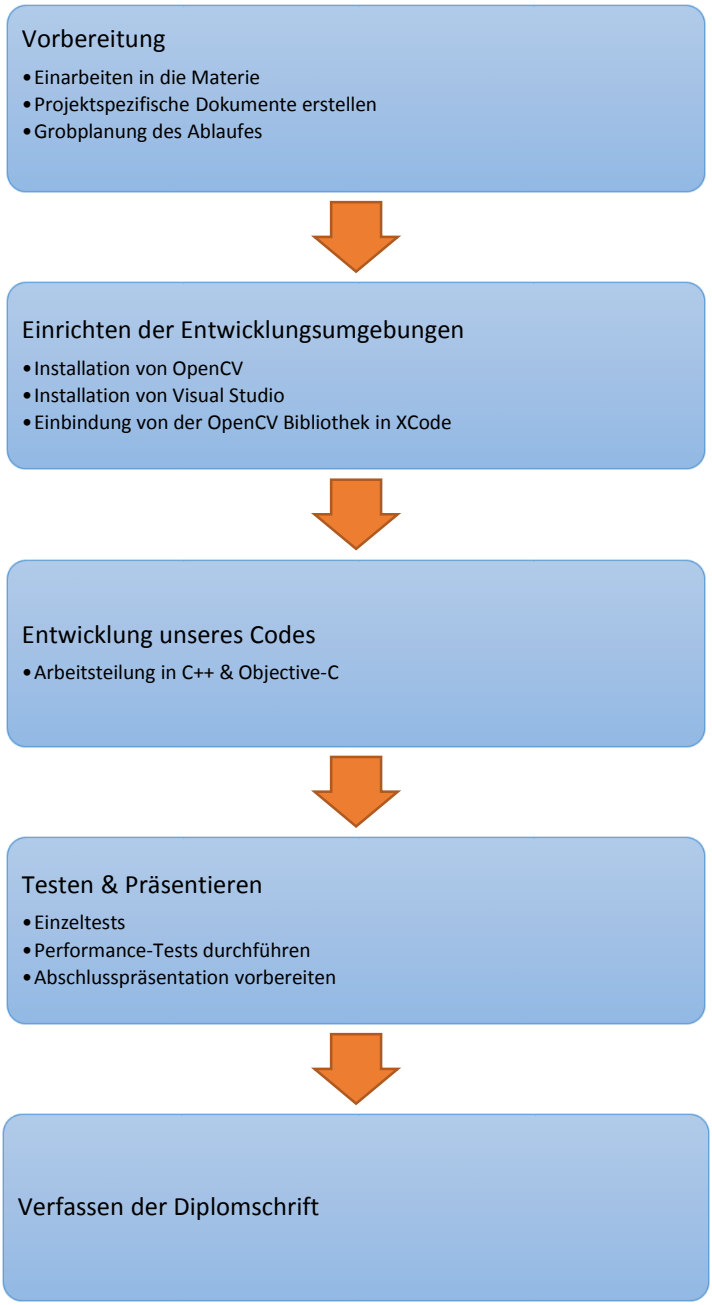

Abbildung 23: Vorgehensplan

In den nächsten Abschnitten werden die einzelnen Punkte bzw. Aufgaben genauer erläutert.

4.5.1. Vorbereitung

Die Idee für unsere Diplomarbeit bekamen wir von unserem Java Programmierprofessor Herrn Dipl.-Ing. Christian Aberger bereits im Sommer 2014, während der letzten Monate unserer 4.Klasse an der HTL Perg.

Aufgrund von Terminschwierigkeiten in den Sommerferien, welche durch Ferialarbeiten und Urlaubsreisen anfielen, begannen wir am Anfang der 5. Klasse damit, die Rahmenbedingungen der Diplomarbeit, in Kooperation mit unserem Auftragsgeber, abzustecken.

Sobald diese klar waren, überlegten wir gemeinsam, wie wir die benötigten Funktionen umsetzen wollten. Dazu machten wir uns erst einmal mit der, uns vom Auftraggeber empfohlenen, Bibliothek „OpenCV“ vertraut um herauszufinden wie wir diese am besten nutzen konnten um unsere Ziele zu erreichen.

Gleichzeitig wollten wir uns auch mit dem Betriebssystem iOS und dessen Programmiersprache Objective-C vertraut machen, da die von uns implementierten Funktionen die Nutzer von iPhones, iPods und iPads beim Auswählen ihrer Fotos unterstützen sollten.

Darum entschlossen wir uns, dass wir die Arbeit unter uns aufteilen sollten um möglichst effizient an die Sache heranzugehen.

So beschäftigte sich Alexander Ortner mit iOS und Objective-C um später alle Funktionen für Nutzer der genannten Produkte in eine handliche und leicht zu bedienende mobile Applikation zu verpacken.

Gleichzeitig machte sich Simon Hinterholzer an die Arbeit, sich mit OpenCV vertraut zu machen um die benötigten Bewertungsmerkmale auf den Bildern zu finden, diese auszuwerten und aufgrund der Ergebnisse eine Punktezahl zu vergeben, mithilfe derer dann auf dem mobilen Gerät eine Reihenfolge der besten Bildern angezeigt werden kann.

Da wir nach einiger Zeit der Einarbeitung ungefähr wussten, wie wir die Technologien benutzen konnten, machten wir uns daran, ein erstes Konzept zu erstellen, auf das wir hinarbeiten wollten.

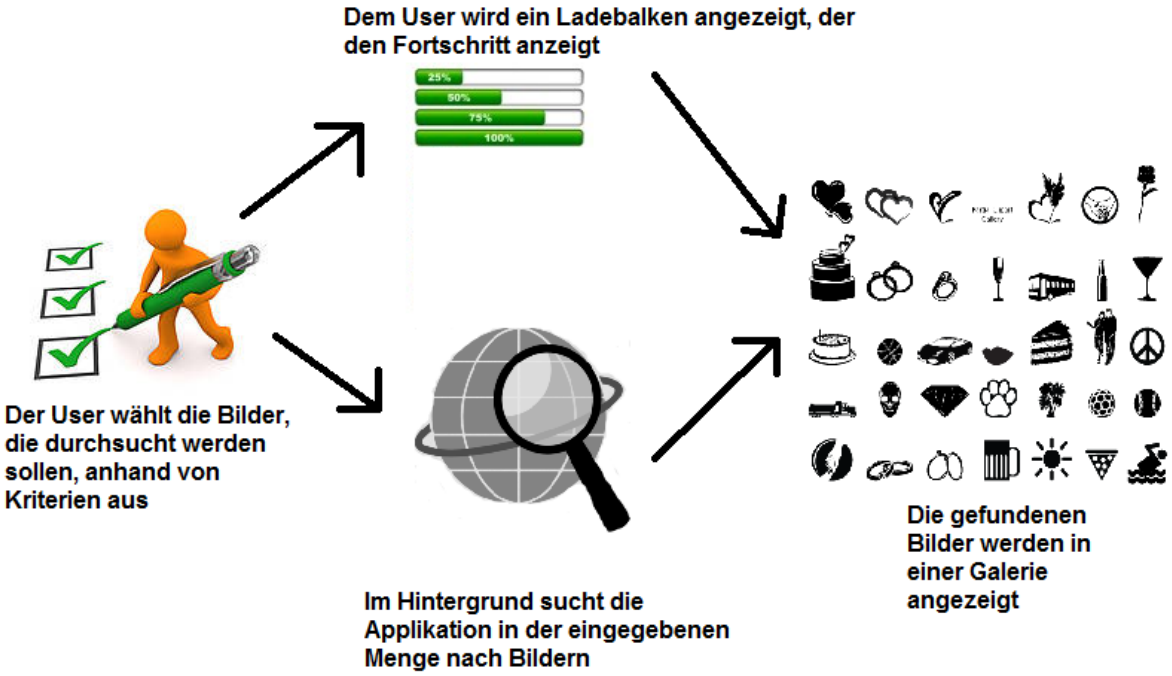

Abbildung 24:erstes Konzept für die geplante Entwicklung

Nun da wir ein Bild vor Augen hatten, was aus unserer Arbeit der nächsten Monate entstehen sollte, machten wir uns daran, für das Projekt benötigte Dokumente zu erstellen. Dazu zählten zum Beispiel das Projekthandbuch, in dem festgehalten wurde, wie wir das Projekt bzw. die Diplomarbeit abwickeln zu gedenken, das Pflichtenheft, eine Auflistung der Funktionen die unser Endergebnis beinhalten sollte, oder eine vorläufige ToDo-Liste, die wir ständig aktualisierten um uns einen Überblick zu verschaffen, was als Nächstes getan werden sollte.

4.5.2. Einrichten der Entwicklungsumgebungen

4.5.2.1. Visual Studio

Um uns mit der Technologie OpenCV vertraut zu machen war es nötig die Bibliothek erst einmal in die, von uns gewählte und auch empfohlene, Programmierumgebung Visual Studio einzubinden. Dabei galt es auf einige Dinge Rücksicht zu nehmen und gewisse Einstellungen vorzunehmen.

Wie man dabei vorgeht damit alles glatt läuft haben wir hier kurz zusammengefasst.

Zuerst muss darauf geachtet werden unter welcher Systemarchitektur man entwickeln möchte. Um das herauszufinden, öffnet man das Startmenü über einen Klick auf das Windows-Symbol, das sich ganz links am unteren Bildschirmrand befindet.

Danach tätigt man einen Rechtsklick auf die Schaltfläche „Computer“ und wählt im sich öffnenden Kontextmenü „Eigenschaften“.

Nun öffnet sich ein Fenster das den Unterpunkt „System“ der Systemsteuerung anzeigt. Hier sehen wir das auf dem Gerät, welches wir verwenden um unser Programm zu entwickeln, das Betriebssystem „Windows 7 Home Premium“ mit einer 64-bit-Systemarchitektur installiert ist.

Wir werden aber trotzdem auf die 32-bit-Version zurückgreifen, da so die Verwendung in Verbindung mit Visual Studio unkomplizierter ist. Nun fahren wir fort indem wir auf „Erweiterte Systemeinstellungen klicken.

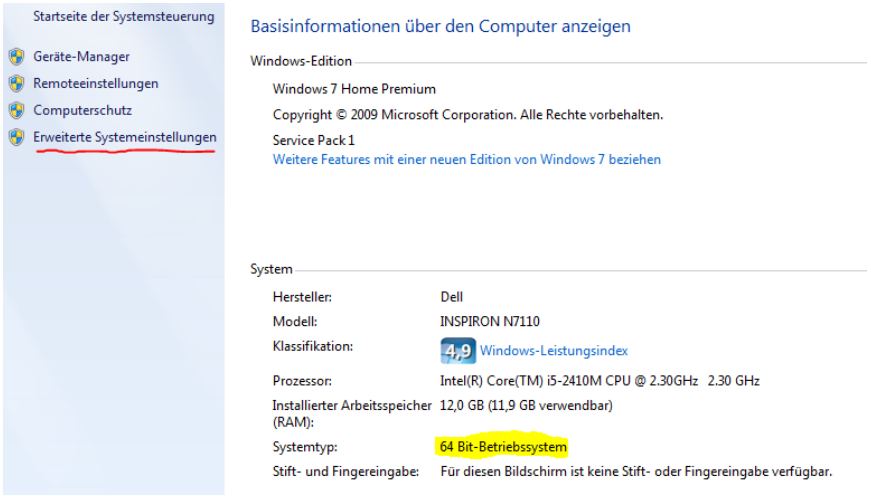

Abbildung 25: Übersicht über das System

Damit unser System weiß wo die Bibliothek liegt, ist es notwendig eine Umgebungsvariable zu erstellen die den Pfad enthält der zum Ablageort der OpenCV-Bibliothek führt. Deswegen klicken wir erst einmal auf „Umgebungsvariablen…“.

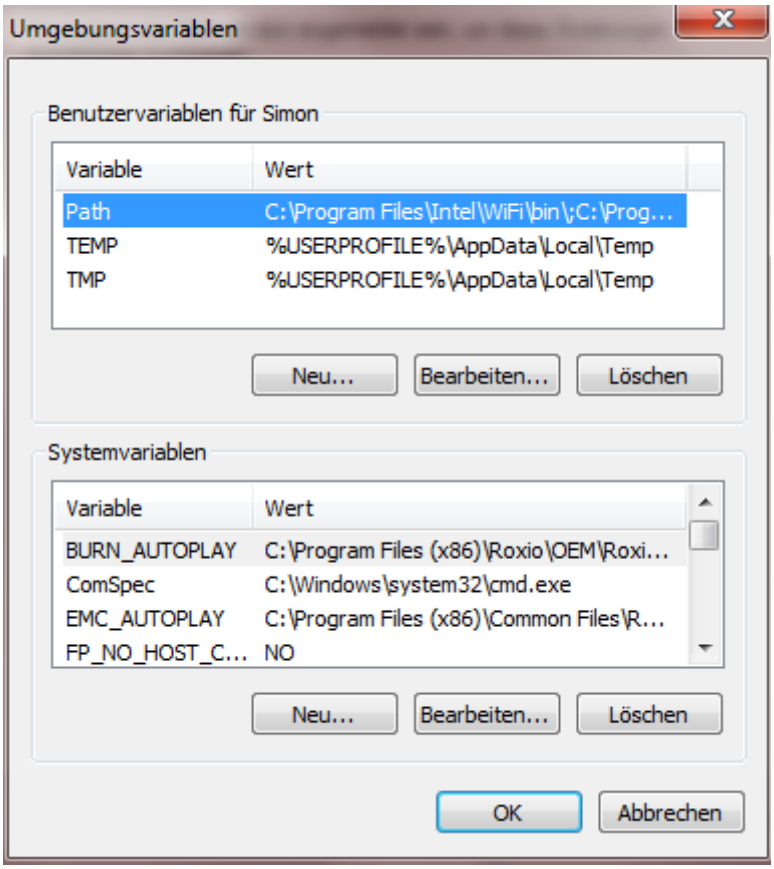

Abbildung 26: Hier werden die Umgebungsvariablen für das System verwaltet

Hier legen wir eine neue Variable, mit den Namen „OPENCV_DIR“ und dem Inhalt „C:\opencv\build\“, an.

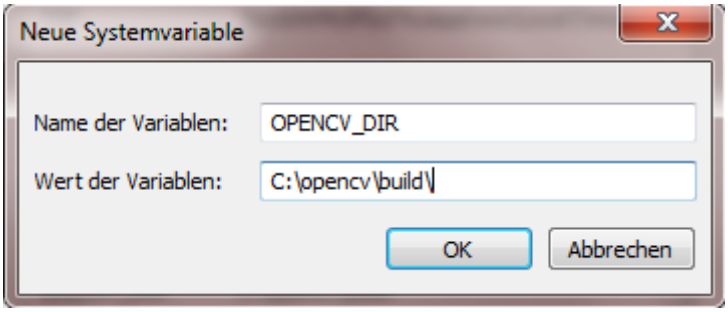

Abbildung 27: Anlegen einer neuen Variable

Außerdem erweitern wir die Systemvariable „Path“. Dazu markieren wir diese und klicken dann auf „Bearbeiten…“ und fügen im vorhandenen Text einfach, durch ein Semikolon getrennt, „%OPENCV_DIR%\x86\vc12\bin“ ein.

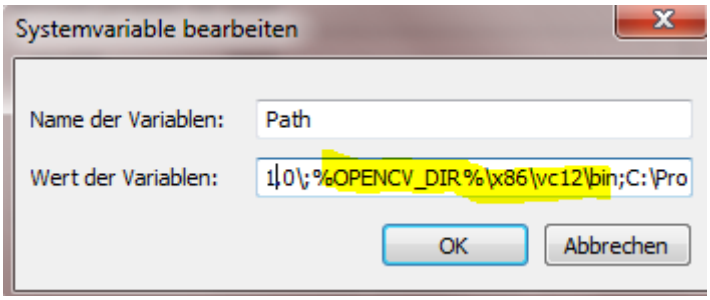

Abbildung 28: Hier wird die PATH-Variable editiert

Nachdem die Umgebungsvariablen passend erstellt und angepasst wurden, können wir damit beginnen ein neues Projekt in der Programmierumgebung Visual Studio anzulegen.

Dazu klicken wir im Menüband links oben auf File und dann New Project.

Im Fenster das sich öffnet wählen wir unter Visual C++, als Projekttyp die Win32-Console Application. Nachdem wir unserem Projekt einen Namen gegeben haben, bestätigen wir das Erstellen mit einem Klick auf den Button mit der Aufschrift OK.

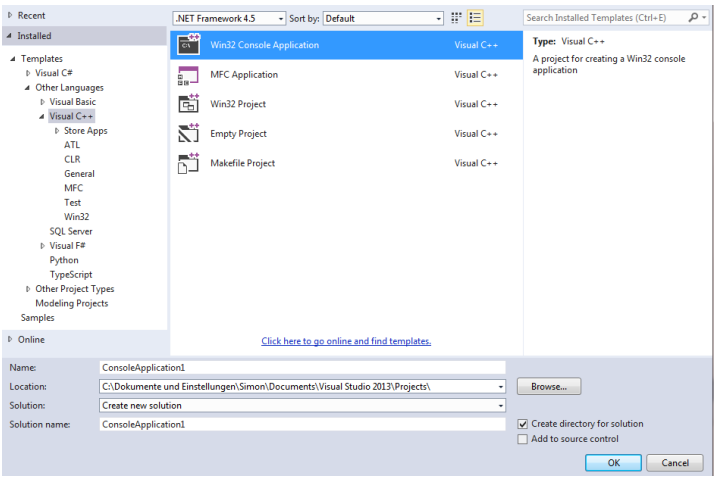

Abbildung 29: Maske zur Projekterstellung

Wenn das Projekt fertig aufgebaut ist sehen wir uns den Solution Explorer an. Dort rechtsklicken wir den Projektnamen und wählen Properties.

Im nächsten Fenster stellen wir bei Configuration, links oben, auf All Configurations und navigieren dann über Configuration Properties zu C/C++ und weiter zum Unterpunkt General. Dort tragen wir bei Additional Include Directories die Zeichenkette $(OPENCV_DIR)\include ein.

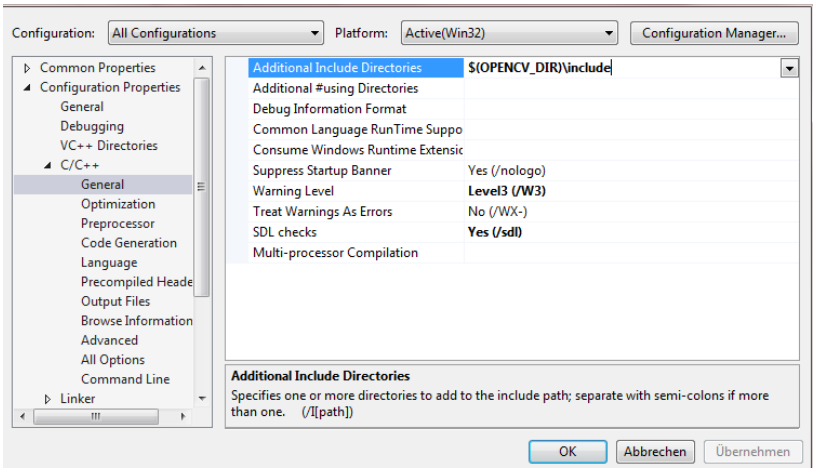

Abbildung 30: Eintragen der zusätzlichen Include Directories

Danach weiter in den Unterpunkt Linker und dort wiederum auf General, wo wir unter Additional Library Directories die Zeichenkette $(OPENCV_DIR)\x86\vc12\lib eintragen.

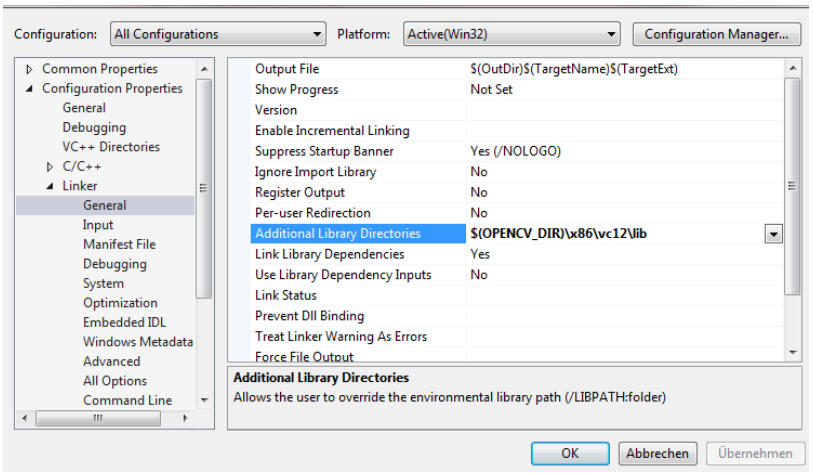

Abbildung 31: Eintragen der zusätzlichen Library Directories

Anschließend wechseln wir zum Punkt Input.

Hier müssen wir uns zwischen Release und Debug Configuration unterscheiden. Vom Vorgehen her ändert sich dabei nichts, jedoch ändert sich eine Kleinigkeit bei den Daten die wir eintragen.

Wir wählen nun eine der beiden Konfigurationen und klicken bei den Additional Dependencies auf den Pfeil und wählen <Edit…>.

Im Fenster, das sich öffnet, tragen wir nun, je nach Konfiguration, folgende Daten ein:

Release Configuration	Debug Configuration
opencv_calib3d2410.lib	opencv_calib3d2410d.lib
opencv_contrib2410.lib	opencv_contrib2410d.lib
opencv_core2410.lib	opencv_core2410d.lib
opencv_features2d2410.lib	opencv_features2d2410d.lib
opencv_flann2410.lib	opencv_flann2410d.lib
opencv_gpu2410.lib	opencv_gpu2410d.lib
opencv_highgui2410.lib	opencv_highgui2410d.lib
opencv_imgproc2410.lib	opencv_imgproc2410d.lib
opencv_legacy2410.lib	opencv_legacy2410d.lib
opencv_ml2410.lib	opencv_ml2410d.lib
opencv_nonfree2410.lib	opencv_nonfree2410d.lib
opencv_objdetect2410.lib	opencv_objdetect2410d.lib
opencv_photo2410.lib	opencv_photo2410d.lib
opencv_stitching2410.lib	opencv_stitching2410d.lib
opencv_superres2410.lib	opencv_superres2410d.lib
opencv_ts2410.lib	opencv_ts2410d.lib
opencv_video2410.lib	opencv_video2410d.lib
opencv_videostab2410.lib	opencv_videostab2410d.lib

Wie man sieht ähneln sich die eingetragenen Daten, jedoch fällt vielleicht auf, dass bei der Debug Configuration jeweils ein kleines ‚D' angehängt wird.

Zum Schluss bestätigen wir unsere vorgenommenen Änderungen mit einem Klick auf die Schaltfläche Übernehmen. Nun können wir mit der Programmierung beginnen.

4.5.3. Entwicklung unseres Codes

Nach dem Einlesen in die neuen Technologien und dem Durcharbeiten von Tutorials fühlten wir uns in der Lage zu Phase zwei überzugehen, der Entwicklung des Codes für unsere Anwendung.

Grob lässt sich diese Phase in folgende Punkte einteilen:

- **OpenCV-Bibliothek implementieren**
  - So erhielten wir die Funktionen, mit denen wir unsere Bewertungsmerkmale bestimmen konnten.
    - z.B. die Möglichkeit ein Histogramm zu berechnen, um so den Farbraum zu bestimmen

- **Implementieren der Berechnung der einzelnen Bewertungsmerkmale**
  - Da wir uns einig waren, dass wir die Bilder anhand von Schärfe, Kontrast, Farbraum und den darauf vorhandenen Gesichtern bewerten wollten, mussten wir ebendiese bestimmen und messen. Dazu war es notwendig, für jedes Merkmal einen Wert zu bestimmen, anhand dessen wir die einzelnen Bilder bewerten konnten.
    - z.B. die Fläche die ein Gesicht in Pixel auf dem Bild einnimmt im Verhältnis zu den Gesamtpixeln aus denen das Bild besteht

- **Bewertungssystem**
  - Damit wir nun Punktezahlen an die Bilder vergeben konnten, mussten wir uns überlegen, wie wir die einzelnen Bewertungsmerkmale gewichten wollten.

- **Design der Benutzeroberfläche**
  - Damit die Nutzer auch etwas mit unserer Applikation anfangen können, ist es von Nöten, dass wir ihnen auch die Möglichkeit der Interaktion bieten. So wurde eine GUI konzipiert, die möglichst einfach und intuitiv zu bedienen ist.

- **Implementierung der Benutzeroberfläche**
  - Anschließend an das Design wurde die Benutzeroberfläche in der Programmiersprache Objective-C implementiert.

- **Zusammenführen der beiden Technologien**
  - Zum Schluss mussten wir den von uns, in Objective-C und OpenCV, geschaffenen Code zusammenführen, um eine vollständige mobile Anwendung zu erhalten.

Während dieser großen Phase war es notwendig, oft den Rat unseres Betreuungslehrers Herrn Dipl.-Ing. Christian Aberger einzuholen, um etwaige Probleme zu besprechen. In solch einem Fall wies uns Herr Aberger auf so manchen Fehler hin und zeigte uns mögliche Lösungswege auf. Diese Treffen hatten aber nicht nur den Sinn einer Fehlerkontrolle und Hilfestellung. Während diesen Meetings konnten wir besprechen was uns noch fehlte und was wir als nächstes in Angriff nehmen sollten.

4.5.4. Testen & Präsentieren

4.5.4.1. Testen

Unsere Applikation sollte später von diversen Kunden genutzt werden, dementsprechend sollte unsere Arbeit fehlerfrei und ohne große Probleme ausführbar sein.
Durch den Faktor der schlechten Zeiteinteilung konnten wir unsere Applikation leider nicht ausführlich testen und somit keine Garantie auf eine fehlerfreie Ausführung gewährleisten.

4.5.4.2. Präsentieren

Jedes Jahr wird an unserer Schule der sogenannte „P@bs-Award“ abgehalten an welchem die Projekte, sowie Diplomarbeiten der Maturanten zuerst einem Komitee und danach der Öffentlichkeit präsentiert werden.
An diesem nahmen wir mit unserer Diplomarbeit selbstverständlich auch teil und versuchten unser Bestes für die Präsentation zu geben.

4.5.5. Verfassen der Diplomarbeit

Da alle Punkte, welche die Arbeit umfasste, detailliert erklärt und schriftlich festgehalten werden mussten, nahm das Verfassen der Diplomarbeit relativ viel Zeit in Anspruch.

5. Umsetzung

5.1. Funktionalitäten

5.1.1. Farbraum

Beim Bewertungsmerkmal Farbraum geht es darum herauszufinden wie viele verschiedene Farben auf dem Bild vorkommen. In unserem Fall gilt dabei je mehr Farben auf dem Bild vorhanden sind desto besser.

Dazu berechnen wir uns zuerst das Histogramm des Bildes.

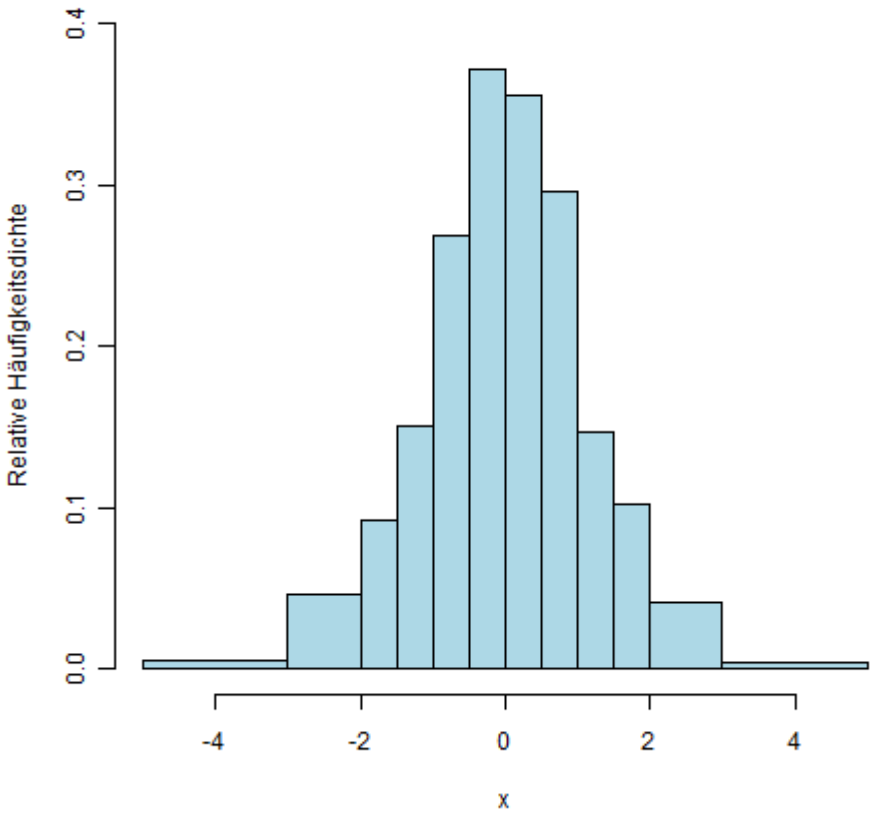

Abbildung 32: Beispiel für ein Histogramm

Ein Histogramm stellt die Häufigkeitsverteilung von bestimmten Merkmalen grafisch dar. Zur Berechnung müssen die Daten in sogenannte „Bins“ (Klassen) eingeteilt werden. Deren Größe kann dabei variabel oder festgelegt sein. Die Höhe der Rechtecke stellt dabei die Häufigkeitsdichte dar.

In unserem Fall teilen wir das Histogramm in 256 Bins auf da wir, aufgrund der RGB-Farben, 256 mögliche Werte zur Verfügung haben. Jedes Bin enthält nun die Anzahl der Pixel deren Farbwert dem Bin-Wert entsprechen.

Mithilfe des berechneten Histogramms, können wir nun die Summen der Pixel auslesen, die einen gemeinsamen Farbwert aufweisen, da das Histogramm diese für uns bildet.

Gibt es Farbwerte deren Pixelsumme unter einem prozentuellem Grenzwert der von der Bildgröße abhängig ist liegen, scheiden diese aus da sie wahrscheinlich zu geringe Flächen füllen um auf einem Bild ins Gewicht zu fallen, da diese mit freiem Auge vermutlich gar nicht erkennbar sind.

5.1.2. Kontrast

Zur Bestimmung des Kontrastes eines Bildes verwenden wir in unserem Programm den Sobel-Operator. Dieser wird häufig in der Bildverarbeitung verwendet und dient zur Kantendetektion wobei er unter Zuhilfenahme der Faltung zum Algorithmus wird.

Prinzipiell wird das Bild zuerst in Graustufen umgewandelt und danach zweimal durchlaufen. Einmal vertikal und einmal horizontal.

Dabei werden jedes Mal die Kanten, also die Änderung der Pixelfarbe, markiert. Diese zwei Ergebnisse werden dann kombiniert und mit Absolutwerten dargestellt. So kann man (und die Maschine, in diesem Fall der Computer) die Umrisse der auf dem Bild vorhandenen Objekte erkennen.

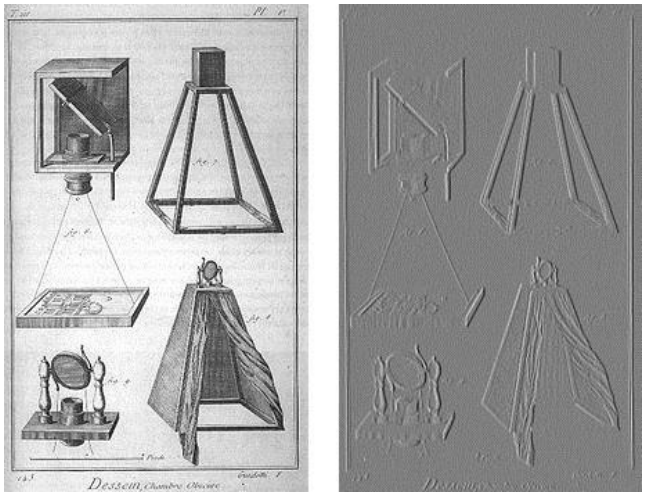

Abbildung 33: v.l.n.r.: Originalbild, Horizontaldurchlauf

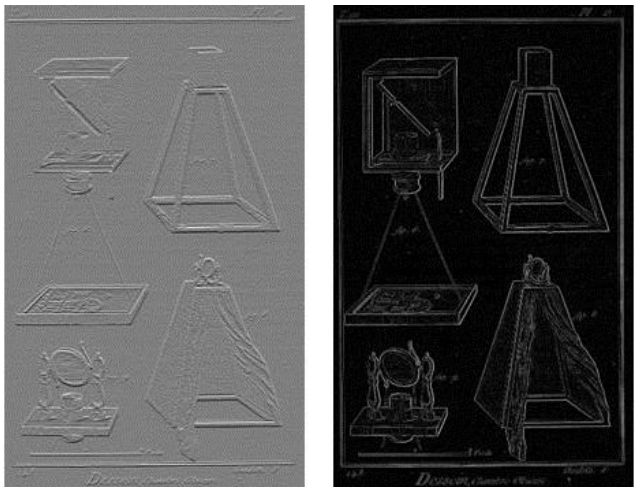

Abbildung 34: v.l.n.r. Vertikaldurchlauf, Kombinationsbild

5.1.3. Schärfe

Um die Schärfe von unseren Bildern herauszufinden machten wir uns auf die Suche nach einem geeigneten Verfahren. Nachdem wir einige Algorithmen zusammengetragen hatten, galt es herauszufinden welcher von diesen für unsere Zwecke am besten geeignet wäre.
Zu Testzwecken wurde ein Beispielbild verwendet welches in zwei Varianten zu Verfügung stand. Variante eins ist scharf und Variante zwei unscharf.
Nun wandten wir die verschiedenen Algorithmen auf dieses Bild an und achteten darauf bei welchem wir die besten Ergebnisse bekommen würden, damit wir diese auch unterscheiden könnten.

Abbildung 36: Scharf gestelltes Beispielbild

```
Modified Laplacian: 13.0643
Variance of Laplacian 842.346
Tenengrad: 6603.01
Normalized Graylevel Variance: 36.7041
```

Abbildung 35: Ergebnisse für das scharfe Bild

Abbildung 38: Nicht scharf gestelltes Beispielbild

```
Modified Laplacian: 5.15426
Variance of Laplacian 128.374
Tenengrad: 2359.01
Normalized Graylevel Variance: 33.2995
```

Abbildung 37: Ergebnisse für das nicht scharfe Bild

Dabei erhielten wir von jedem Algorithmus völlig unterschiedliche Werte.
Nachdem wir dies mit einigen Bildern wiederholt hatten entschieden wir uns für den „Variance of Laplacian“-Algorithmus. Dabei wird der Laplace-Operator verwendet. Hier zeigte sich deutlich, dass OpenCV eine tolle Technologie zur Bildverarbeitung ist, da der Laplace-Operator darin bereits implementiert ist und wir so schnell zu einem Ergebnis kommen konnten.

5.1.4. Gesichtserkennung

Wir konzentrierten uns bei unserer Anwendung auf die Gesichtsdetektion, da es für uns ein wichtiges Bewertungsmerkmal ist, ob sich Gesichter auf dem Bild befinden und wenn ja, wie viel Platz das Gesicht oder die Gesichter einnehmen da wir uns entschlossen haben, dass ein Foto mit großem und gut erkennbaren Gesicht besser ist als ein Foto mit kleinem und schwer zu erkennenden Gesicht.

Wir versuchen auf den zu bewertenden Bilden mithilfe von sogenannten „Haarcascades“ (→Haar-Detektor-Klassifikatoren) Gesichter zu entdecken.

Bei dieser Technik wird ein rechteckiges Suchfenster über das Bild geschoben. Die Größe dieses Rechteckes ändert sich nach jedem Suchdurchlauf. Während sich die Fläche nun über das Bild bewegt wird geprüft, ob innerhalb dieses Feldes ein Objekt vorhanden ist nach dem gesucht wird. Diese Überprüfung wird von einem Detektor durchgeführt der anhand von rechteckigen Blöcken auf Merkmale prüft.

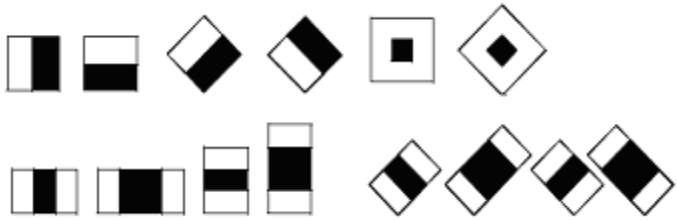

Abbildung 39:mögliche Merkmale

Um den Wert eines solchen Merkmals zu bestimmen subtrahiert man die Summe der Pixel die sich in den schwarzen Blöcken befinden, von der Summe der Pixel die sich in den weißen Blöcken befinden.

Jede dieser Suchformen kann, mit skalierbarer Größe, beliebig im Suchfenster angeordnet werden.

5.2. Technologie Objective C

Abbildung 40: Objective C

Objective-C ist eine Erweiterung der Programmiersprache C und in der Lage objektorientiert zu Programmieren. Weiters ist es die primäre Sprache von Cocoa und GNUstep. Durch die objektorientierte Erweiterung ist es Objective-C möglich auch andere imperative Sprachen anzuwenden wie z.B. Objective-C++, welches wir unter anderem auch verwenden, da dies die Mischung mit C++ Code gestattet.

5.2.1. Geschichte

Entwickelt wurde Objective-C in den 80iger Jahren von Brad Cox und Tom Love. Im Jahre 2006 wurde Objective-C 2.0 von Apple veröffentlicht, welches unter anderem Verbesserungen in der Syntax und in der Laufzeit-Performance aufwies. Weiters wurde auch 64-Bit-Plattformen unterstützt sowie ein modernes Speichermanagement implementiert.

5.2.2. Eigenschaften von Objective-C

5.2.2.1. Dynamisches Binden

- Diese Eigenschaft von Objective-C sticht besonders hervor. Methoden werden dynamisch gebunden und somit ist eine Polymorphie auch außerhalb einer Klassenhierarchie möglich. Das heißt eine Methode kann aufgrund seines Namens von Objekten einer beliebigen Klasse ausgeführt werden. Einzige Voraussetzung ist das die Methode von der Klasse implementiert wird. Damit muss der Aufrufer die Klasse nicht kennen bzw. muss sie nicht in einer Basisklasse definieren.

5.2.2.2. Dynamische Typisierung

- Aufgrund der Dynamischen Bindung ist es nicht notwendig, dass der Absender die Klasse des Empfängers kennt. Dafür existiert eine „id“, welche für jedes Instanz Objekt einer Klasse stehen kann. Damit sind einzelne Instanzen stets typisiert und gehören genau einer Klasse an. Kurz zusammengefasst kann man sagen, dass Objective-C zwar streng typisiert dies jedoch dynamisch macht.

5.2.2.3. Nachrichten

- In Objective-C werden Methoden und Nachrichten strikt getrennt, somit spricht man nicht von Methodenaufrufen. Es gibt nur einen Nachrichtensender und Nachrichtenempfänger (Sender und Receiver). Welche Methode aufgerufen wird entscheidet der Receiver, welcher als erstes versucht eine gleichnamige Methode zu finden. Dazu wird der Nachrichtenname, für den es einen eigenen Datentyp (Selector) gibt, herangezogen.
- Daraus entsteht die Möglichkeit ohne Eingabe von Sourcecode ganze Objektgraphen & Bedienungsoberflächen zu gestalten. Das heißt der Programmierer muss keinen Sourcecode dafür schreiben wodurch ein Großteil der Programmierarbeit entfällt.

5.2.2.4. Kategorien

- Werden bestehende Klassen um Methoden erweitert, spricht man von Kategorien. Diese Methoden, welche in Kategorien enthalten sind enthalten auch Instanzen die von fremdem Code erzeugt worden sind, auch wenn der fremde Code diese Kategorie nicht kennt.
- Man kann diesen Kategorien auch Instanzen hinzufügen, welche aus einem bereits bestehenden Code stammen und im Grunde die Kategorie auch nicht kennen. Der Vorteil von Kategorien besteht darin, dass es möglich ist fremden Klassen Methoden hinzuzufügen.

5.2.2.5. Protokolle

- In Objective-C lassen sich Sätze von Methoden in Protokollen zusammenfassen. Diese Eigenschaft kann man mit der Programmiersprache Java in Bezug auf die Verbindung mit Interfaces vergleichen.

5.2.2.6. Runtime Type Information

- Jedes Objekt führt einen Verweis auf seinem Typ mit. Diese geschieht mittels RTTI (Runtime Type Information), durch welches man zur Laufzeit jedes Objekt seiner Klasse zuordnen kann. Darüber hinaus sind alle Informationen über Instanz Variablen sowie implementierte Methoden in der Klasse enthalten.
- Durch diese Eigenschaft kann der Receiver der Nachricht eine Methode zuweisen. Auf der anderen Seite kann auch der Absender eine Abfrage starten um Information über die implementierten Methoden zu erhalten.

5.2.2.7. Klassenobjekte

- In Objective-C ist die Bezeichnung „Exemplar einer Klasse“ aufgrund der Tatsache, dass es nicht nur Instanz Objekte sondern auch Klassenobjekte gibt mehrdeutig. Erkennen kann man eine Klassenmethode im Sourcecode anhand eines „+“, welches der Methode vorangestellt ist. Der Unterschied dieser Methoden in der Schreibweise besteht darin, dass eine Instanz Methode „-class“ und eine Klassenmethode „+class“ verwendet um das erwünschte Klassenobjekt zu bekommen.

5.2.3. Vererbung in Objective C

Das Konzept der Vererbung von Objective-C ist an Smalltalk angelehnt. Unterstützt wird daher nur Einfachvererbung. Es existiert eine genaue Vererbungshierarchie. Die Klasse Object bildet dabei die Wurzel. Alle Methoden bzw. Instanz Variablen, welche nicht explizit als geschützt deklariert wurden, werden von den oberen Klassen geerbt.

Methodenaufrufe werden ebenfalls wie in Smalltalk ausgeführt. Die Nachricht wird in der Vererbungshierarchie nach oben weitergereicht, falls der Receiver keine passende Methode für die Nachricht finden kann.

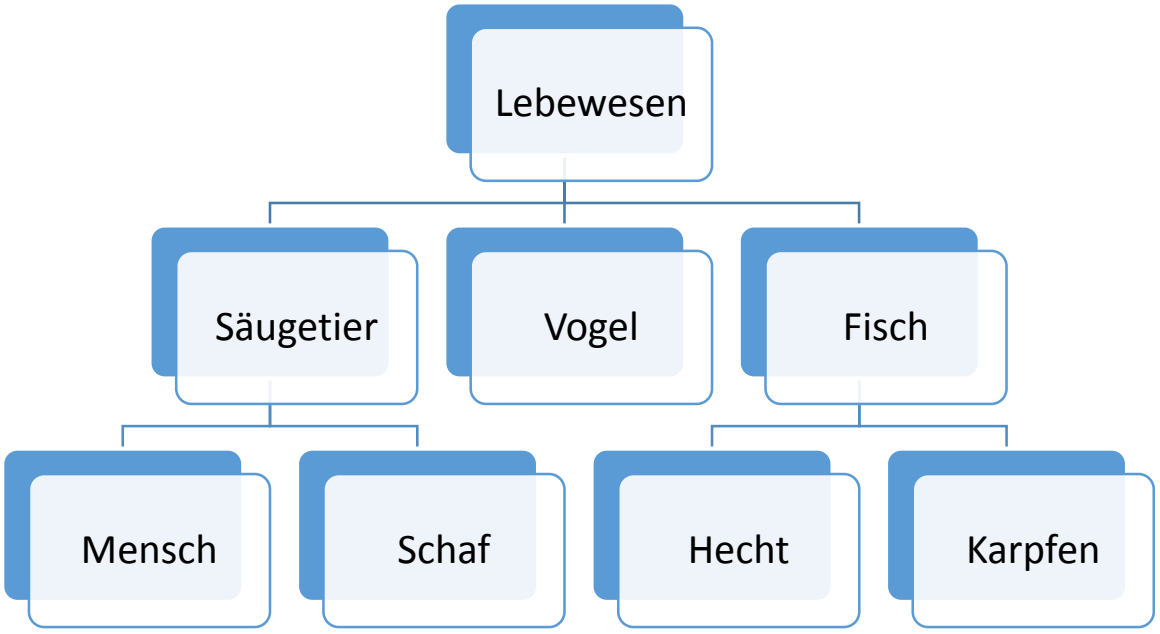

Abbildung 41: Grundlegendes Beispiel für Vererbung

An der Grafik oberhalb kann man ein grundlegendes Beispiel der Vererbung erkennen.

5.3. Technologie iOS Betriebssystem

Abbildung 42: Apple iOS

5.3.1. Geschichte

Im Jahre 2005 beginnt die Reise von iOS mit der Idee zur Entwicklung eines Tablet-Computers. Diese Idee wurde jedoch von Steve Jobs, dem Gründer von Apple, abgelehnt und stattdessen an der Entwicklung eines Telefons begonnen. Am 9 Januar 2007 wurde das Betriebssystem zum ersten Mal zusammen mit dem iPhone auf der MacWorld Conference and Expo präsentiert. Dieses iPhone hatte jedoch einen recht spärlichen Funktionsumfang und bot keine Funktionen die einem modernen Smartphone gerecht werden könnten.
Um auch externen Entwicklern die Chance zu ermöglichen eigene Apps zu erstellen wurde am 6 März 2008 das SDK für iOS veröffentlicht. Diese selbstprogrammierten Apps können dann im App Store angeboten werden.
Im November 2012 wurde das iOS-Design komplett durch Jonathan Ive, den Chef-Designer, neugestaltet. Dadurch entstand das buntere Betriebssystem iOS 7, welches sich am Flat Design(grafisch minimalistischer Gestaltungsstil) orientierte.

5.3.2. Der Lebenszyklus einer App

Eine App entsteht durch das Zusammenspiel des Systems, welches ein Grundgerüst liefert und dem Benutzer, der dieses Grundgerüst mit seinem eigenen Code füllt. Das heißt, das System liefert alle notwendigen Eigenschaften damit eine App ausgeführt werden kann und der Programmierer gestaltet die App nach seinem Belieben. Um zu verstehen wie diese iOS Infrastruktur arbeitet werden die wichtigsten Hauptpunkte im Folgenden beschrieben.

5.3.2.1. Die Main Funktion

Jede C basierte Applikation startet mit der „main“ Funktion. Der Unterschied zu iOS ist, dass die Main-Funktion nicht selbst geschrieben werden muss, sondern automatisch von XCode generiert wird. Diese automatisch implementierte Funktion sollte nicht verändert werden.

```
#import <UIKit/UIKit.h>
#import "AppDelegate.h"

int main(int argc, char * argv[])
{
    @autoreleasepool {
        return UIApplicationMain(argc, argv, nil, NSStringFromClass([AppDelegate
class]));
    }
}
```

Abbildung 43: iOS Main Funktion

Die UIApplicationMain Funktion lädt das User Interface von den verfügbaren storyboard files und ruft danach den eigens programmierten Code auf. Das einzige um das sich der Benutzer kümmern muss sind die storyboard files und den eigenen Sourcecode.

5.3.2.2. Aufbau einer App

iOS App nutzen die Model-View-Controller Architektur wie in der folgenden Grafik ersichtlich. Diese Architektur trennt die logische Daten- bzw. Betriebsebene von der visuellen Darstellungsebene der Daten ab. Diese Architektur ist vor allem notwendig um Applikationen auf verschiedenen Geräten sowie in verschiedenen Auflösungen darstellen zu können. Das Herzstück dieser Architektur ist die UIApplication, welche die Interaktion zwischen dem System und anderen Objekten in einer App ermöglicht.

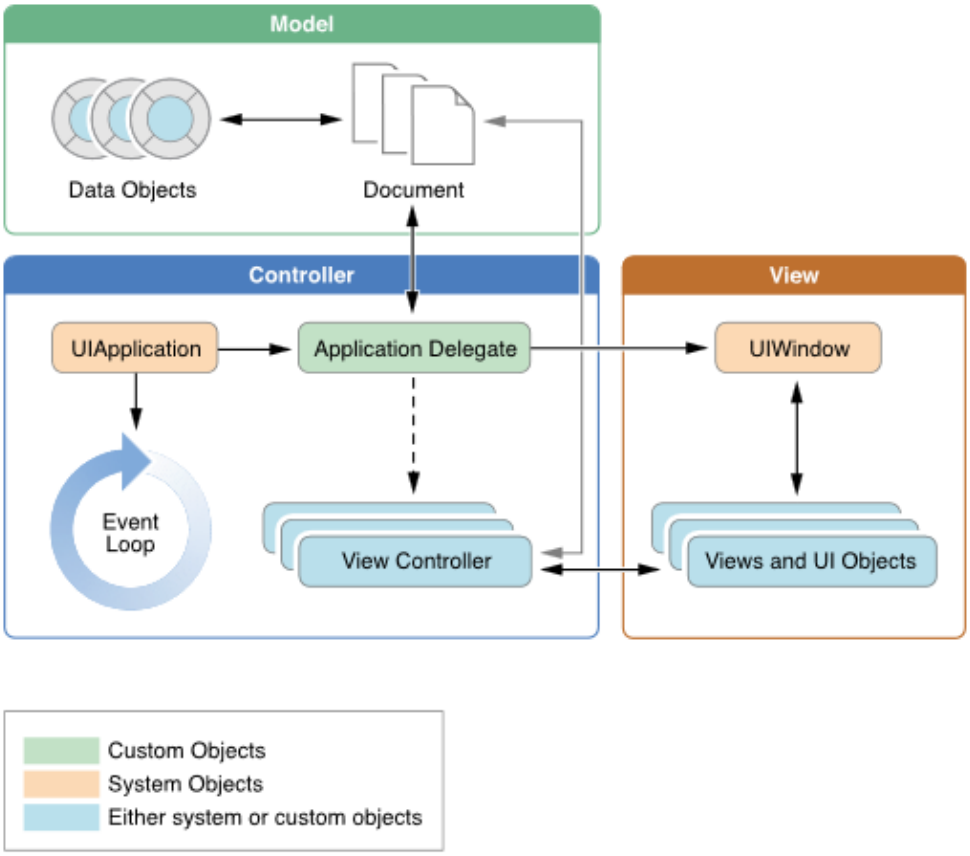

Abbildung 44: Architektur einer iOS App

- UIApplication Object:
  - Die UIApplication verwaltet die Event Loop sowie andere App Eigenschaften. Weiters benachrichtigt sie andere Objekte, wenn zum Beispiel der User auf den Touch Screen drückt.

- App Delegate Object:
  - Diese Funktion ist das Herzstück des eigen programmierten Codes. App delegate arbeitet zusammen mit der UIApplication um die Initialisierung, Statuswechsel sowie Steuerung von high-level Events der App zu behandeln.

- Documents:
  - Documents können benutzt werden um die data-model Objects in der Applikation zu verwalten. Diese sind zwar nicht zwingend erforderlich, eröffnen aber die Möglichkeit seine Daten in einem einzigen Dokument oder in mehrere Dokumente zusammenzufassen.

- Data Model Objects:
  - Die Data Model Objects sind dafür verantwortlich die Daten der App zu speichern

- View Controller Objects:
  - Der View Controller ist primär für die Präsentation der App-Daten auf dem Bildschirm verantwortlich. Dabei verwaltet er eine Single-View(Einzelansicht) mit der Sammlung der subviews(Unteransichten). Soll eine Ansicht angezeigt werden ist der View Controller dafür zuständig diese auf dem Bildschirm anzuzeigen.
  - Als Basis-Klasse existiert der UIViewController der über allen anderen View-Controller Objects steht. Dieser hat die benötigten Basiseigenschaften wie z.B. Laden der Ansichten, Anzeigen der Ansichten, Rotieren des Bildschirmes wenn das Gerät gedreht wird und viele weitere System Eigenschaften.

- UIWindow Object:
    - Das UIWindow Object koordiniert die auf dem Bildschirm angezeigten Ansichten. Die meisten Applikationen haben nur eine Ansicht, welche im Hauptfenster präsentiert wird. Doch es kann auch sein das eine zusätzliche Ansicht z.B. für den Inhalt verwendet wird.
    - Um jetzt den Inhalt der Applikation zu ändern, wird ein View Controller verwendet welcher die View(Ansicht) ändert. Das Fenster selbst wird niemals gewechselt sondern nur die View.

- View- und Control Objects:
    - Views und Controls werden dazu verwendet um den Inhalt der Applikation darzustellen. Eine View ist ein Objekt, welches Inhalt in einen gewissen Rahmen visualisiert und auf Events in diesem Rahmen reagiert. Controls sind zum Beispiel buttons(Knöpfe) oder text fields(Textfelder)

5.3.2.3. Die Hauptschleife

Die Hauptschleife einer Applikation verwaltet alle vom User ausgelösten Events. Wird eine Applikation gestartet so aktiviert das UIApplication object die Hauptschleife. Diese agiert im main-thread der App damit gewährleistet werden kann, dass die Aktionen, welche vom Benutzer ausgelöst werden, in der richtigen Reihenfolge ausgeführt werden.

Sobald der User mit dem Gerät interagiert bzw. ein Event auslöst wird dieses vom Operating System über einen Port in die Event queue weitergeleitet. Dort wird ein Event nach dem anderen in der Hauptschleife ausgeführt. Das UIApplication object ist das erste Objekt, welches die Aufgabe erhält. Es entscheidet welche Methode bzw. was ausgeführt werden soll. In folgender Grafik ist dieser Ablauf gut dargestellt. Hierbei handelt es sich um ein sogenanntes touch Event.

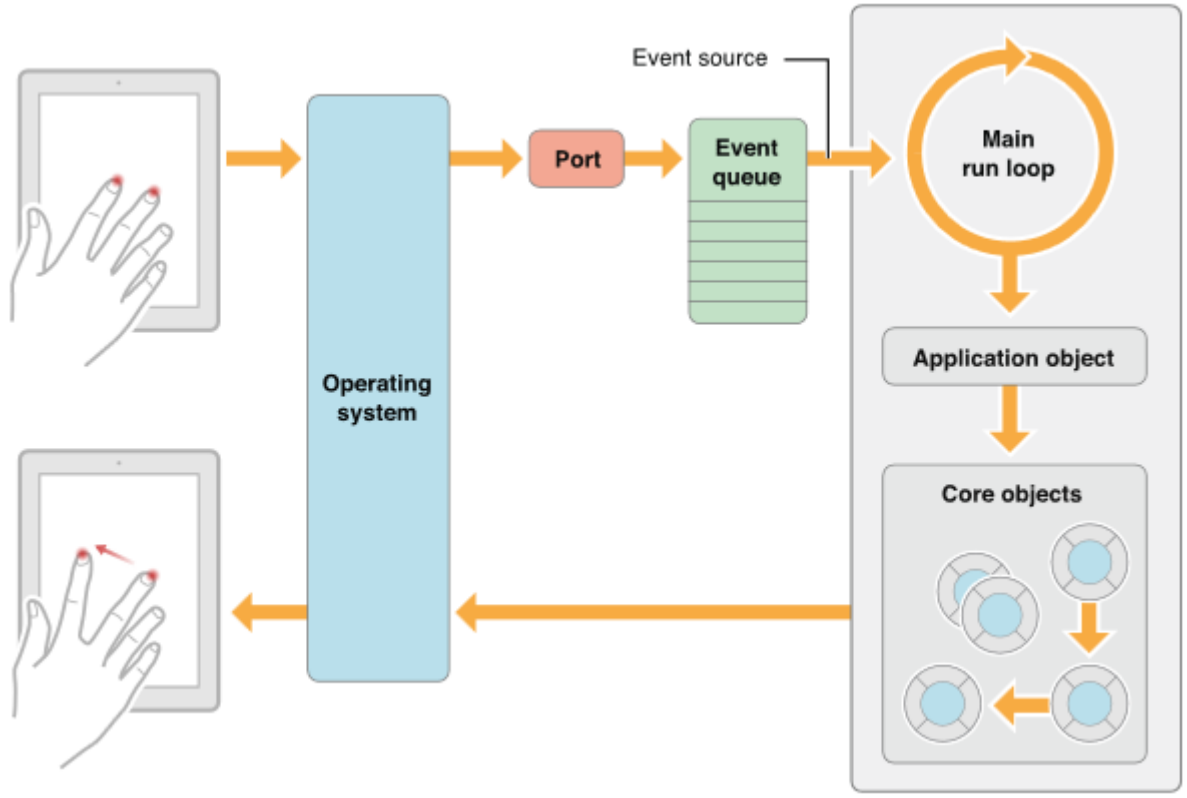

Abbildung 45: Eventbehandlung in der Hauptschleife

5.3.2.4. Ausführungsstatus einer App

Jede Applikation befindet sich immer in einem Statuszustand, welche in folgender Grafik bzw. Aufzählung näher erläutert werden.

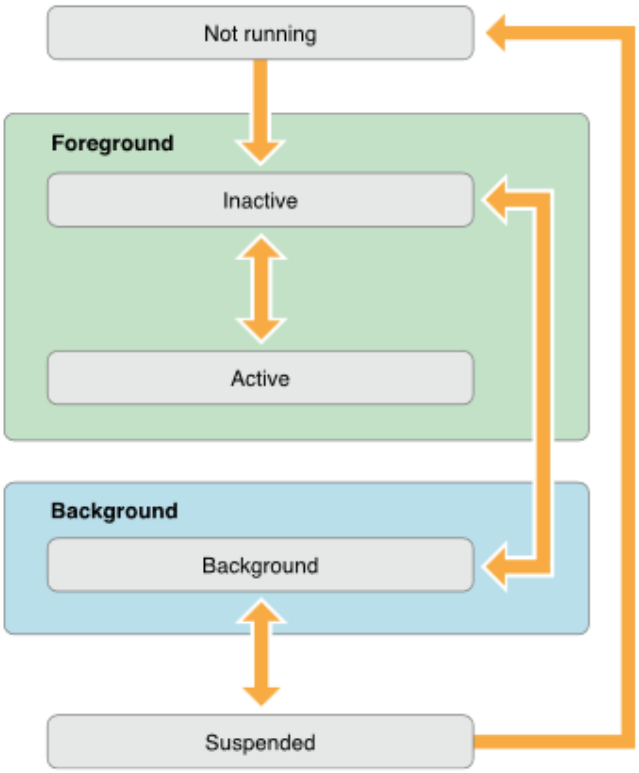

Abbildung 46: Status einer App

- **Not running:**
  - Die Applikation wird nicht ausgeführt bzw. wurde vom System beendet.

- **Inactive:**
  - Die Applikation läuft im Vordergrund erhält jedoch zurzeit keine Events. Grund dafür kann sein das im Hintergrund ein anderer Code ausgeführt wird. Normalerweise bleibt eine App nur kurz in diesem Modus.

- **Active:**
  - Die App läuft im Vordergrund und empfängt Events. Dies stellt den normalen Modus einer Applikation dar.

- **Background:**
  - Die Applikation befindet sich im Hintergrund und führt Code aus. Die meisten Applikationen betreten diesen Status kurz während sie beendet werden. Fordert eine App eine extra Zeit an kann es sein, dass sie sich auch eine längere Zeitdauer im Hintergrund Status befindet.

- **Suspended:**
  - Die App befindet sich im Hintergrund führt dabei aber keinen Code aus. Das System macht das automatisch. Das heißt während sich die Applikation in diesem Status befindet besetzt diese zwar Speicher aber führt keinen Code aus.
  - Der Vorteil an diesem Status ist der, dass im Falle einer Speicherknappheit Apps einfach in diesen Status versetzt werden um Platz für Applikationen zu machen die im Vordergrund laufen.

Es ist auch möglich einen Status mit einer entsprechenden Methode zugunsten seiner eigenen Wünsche für die App zu beeinflussen.

- *application:willFinishLaunchingWithOptions:*
  - Mit dieser Methode kann man während die App starten Code ausführen.

- *application:didFinishLaunchingWithOptions:*
  - In dieser Methode kann man noch Optionen ausführen bevor die Applikation dem User angezeigt wird.

- *applicationDidBecomeActive:*
  - Durch diese Methode kann man seiner Applikation mitteilen, dass sie im Vordergrund stehen wird.

- *applicationWillResignActive:*
  - Wird verwendet um mitzuteilen, dass die App den Vordergrund verlassen wird.

- *applicationDidEnterBackground:*
  - Die Applikation läuft im Hintergrund und kann jederzeit in den suspended Status wechseln.

- *applicationWillEnterForeground:*
  - Die App wechselt vom Hintergrund zum Vordergrund, sie ist jedoch noch nicht aktiv.

- *applicationWillTerminate:*
  - Die Applikation wird beendet. Diese Methode wird nicht aufgerufen wenn der Status der App suspended ist.

5.3.2.5. Beenden einer Applikation

Applikationen müssen jederzeit für ein mögliches Beenden des Programmes bereit sein, d.h. es bleibt keine Zeit Benutzer Daten zu speichern oder andere kritische Aufgaben zu erledigen. Dass eine App durch das System beendet wird ist ganz normal. Das System macht das vor allem um Speicher frei zu räumen, sodass andere Applikationen, welche der User öffnet, ausgeführt werden können. Weiters beendet das System Apps, welche seit einem gewissen Zeitraum nicht benützt worden sind oder nicht mehr auf events reagieren.

Befindet sich eine Applikation im Status suspended so erhält diese keine Information falls sie beendet wird. Das System beendet den Prozess und macht somit den belegten Speicher wieder frei.

Wird eine App von einem Benutzer beendet, so hat es denselben Effekt wie das Beenden einer Applikation mit dem Status suspended. Der Prozess wird gekillt und es wird keine Information an die App gesendet.

5.3.2.6. Threads und Concurrency

Im Grunde erstellt das System den Hauptthread und der Programmierer kann eigene Threads hinzufügen um andere Aufgaben zu erfüllen. Die bevorzugte Technik für iOS Apps ist es den Grand Central Dispatch (GCD) sowie andere asynchrone Interfaces anstatt eigenen Threads zu verwenden. Mit dieser Technik definiert man die Arbeit, sowie die Reihenfolge in welcher diese erledigt werden soll, und das System entscheidet, anhand der verfügbaren CPUs, wie es am besten erledigt wird. Überlässt man diese Aufgabe dem System wird erstens der Code vereinfacht und zweitens eine bessere Leistung erzielt.

5.4. Probleme

5.4.1. Probleme bei der Gesichtserkennung

5.4.1.1. Bewertungsproblem bei Bildern auf denen keine Gesichter vorhanden sind.

Problem:
Wird ein Foto bewertet auf dem kein Gesicht gefunden wurde bzw. möglicherweise kein Gesicht vorhanden ist, fehlt diesem Bild ein zusätzlicher Bewertungspunkt. D.h. das Bild schneidet in der Bewertung schlechter ab. Das Problem besteht nun darin das z.B. ein Landschaftsfoto in den anderen Bewertungskriterien sehr gut abschneidet, doch aufgrund des fehlenden Bewertungspunktes als nicht gut eingestuft wird und somit dem User nicht in der Ergebnismenge präsentiert wird obwohl dieser das Foto mit großer Wahrscheinlichkeit selbst ausgewählt hätte.

Lösung:
Gelöst wurde dieses Problem, indem wir das Bewertungssystem an eben diesen Fall anpassten. Damit wir auch mit einem Bewertungsmerkmal weniger auf ähnliche Punkte kommen, haben wir die einzelnen Kriterien so gewichtet, dass schlussendlich die Bewertungen des Farbraums, des Kontrasts, der Schärfe und der Gesichtserkennung einen ähnlichen Wert annehmen. So wird verhindert, dass Bilder, auf denen z.B. viele Gesichter erkannt werden, die jedoch unscharf sind und wenige Farben haben, besser eingestuft werden als farbenfrohe Landschaftsbilder die einen abwechslungsreichen Kontrast und eine gute Schärfe bieten.

5.4.1.2. Es werden Gesichter erkannt, die keine sind

Problem:

Als wir die ersten Tests mit Beispielbildern durchführten, fiel uns auf, dass es vereinzelt vorkam, dass auf den Bildern Gesichter gefunden wurden die keine waren. So wurde etwa auf einem Vorhang oder einem Baum ein „Gesicht“ eingezeichnet, das der Algorithmus aufgrund der Haarcascades als solches identifizierte. Dies war ein Problem da wir so bei der Bewertung vollkommen falsche Werte geliefert bekommen würden.

Lösung:

Die Lösung des Problems war ebenso genial wie simpel. Wir überlegten uns was ein Gesicht zu einem Gesicht machen würde und kamen zu dem Schluss, dass zu einem Gesicht Augen gehören würden. So entschieden wir uns die gefundenen Gesichter noch einmal von dem Algorithmus durchlaufen zu lassen, jedoch ließen wir ihn diesmal nach Augen Ausschau halten. Fand er in dem Feld, in dem er ein Gesicht vermutete keine dazugehörigen Augen, so wurde dieses verworfen. Nun hatten wir zum Schluss eine verlässliche Liste mit gefundenen Gesichtern und dazu die Information, wieviel Platz sie auf dem Bild einnahmen was wir als Bewertungsgrundlage verwendeten.

Beispiel:

Abbildung 47: Hier wird zwischen den Fahnen ein Gesicht gefunden

5.4.1.3. Gesichter werden nicht erkannt

Problem:

Trotz Vorhandensein eines Gesichtes auf einem Foto, wird dieses von dem Algorithmus nicht erkannt. Dieses Problem kann durch eine mögliche Verdeckung eines Auges hervorgerufen werden oder aber auch wenn eine Person seitlich steht und somit der Algorithmus keine zwei Augen als Anhaltspunkte heranziehen kann. Dieser Fehlerfall wirkt sich natürlich wieder auf die Bewertung eines Bildes aus. (→Verweis auf das Problem: Bewertungsproblem bei Bildern auf denen keine Gesichter vorhanden sind)

Lösung:

Da wir bereits bei Bildern, auf denen überhaupt keine Gesichter vorhanden waren vor dem Problem standen, da ihnen ein Bewertungskriterium fehlte, konnten wir diese beiden Probleme auf einmal lösen, indem wir das Bewertungssystem so anpassten, wie im Punkt „5.4.1.1. Bewertungsproblem bei Bildern auf denen keine Gesichter vorhanden sind“ erklärt wurde.

Beispiel:

Abbildung 48: Der Algorithmus erkennt hier ein Gesicht nicht

5.4.2. Sonstige Probleme in unserer Diplomarbeit

5.4.2.1. Probleme bei der Aufwandschätzung

Problem:
Als wir uns zu Beginn in die Technologien einarbeiteten, wirkte es auf uns so, als wären diese in der Lage sämtliche Aufgaben im Handumdrehen zu erfüllen. So schätzten wir den Aufwand viel geringer ein als dieser eigentlich war. Aufgrund dieser fälschlichen Annahme gingen wir eher entspannt an die Arbeit ohne uns großen Stress zu machen. Diese Falscheinschätzung fiel uns jedoch im Laufe der Diplomarbeit in den Rücken sodass wir terminliche Probleme bekamen sowie auch die Umsetzung unserer Ziele gefährdete.

Lösung:
Um die Diplomarbeit dennoch so gut es geht voran zu treiben, konzentrierten wir uns vor allem auf die Hauptfunktionen der Applikation. Das heißt wir vernachlässigten kleinere Funktionen die für uns als entbehrlich eingestuft wurden.

5.4.2.2. Erlass der GPS basierten Fotoauswahl

Problem:
Aufgrund des enormen Verzuges, welchen wir während der Diplomarbeit erlitten, mussten wir feststellen, dass die Aufgabe die Fotoauswahl aufgrund von GPS Kriterien zu beeinflussen nicht mehr realisierbar ist.

Lösung:
Nach Absprache mit Herrn Aberger, beschlossen wir, diesen Teil der Diplomarbeit als mögliche Erweiterung für eine zukünftige Arbeit anzusehen und damit diesen Teil zu streichen.

5.5. Arbeitsweise

5.5.1. Arbeitsteilung

In der folgenden Grafik ist ersichtlich, welchen Hauptaufgabenbereichen sich die Diplomanden jeweils primär gewidmet haben.

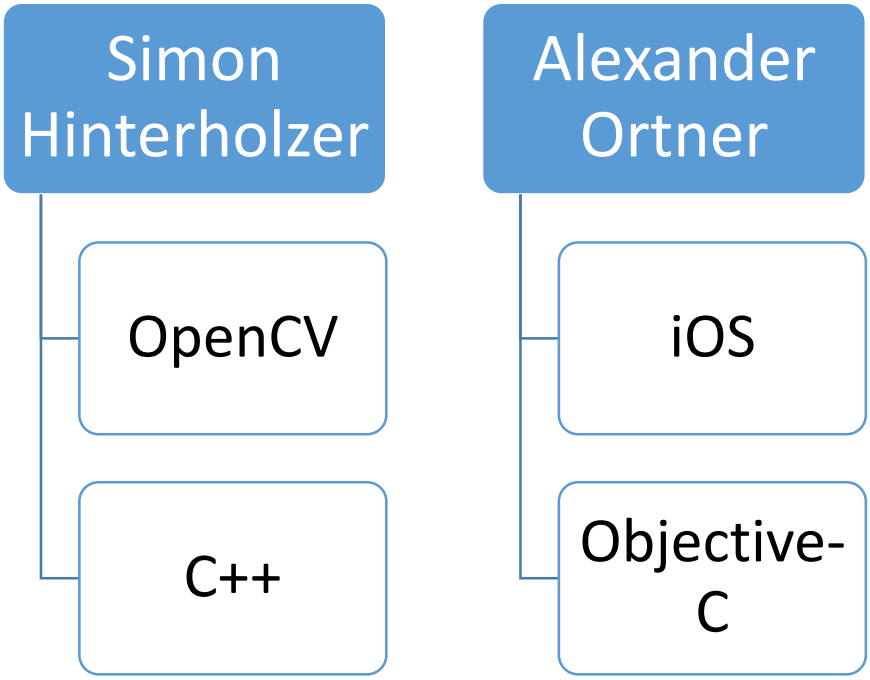

Abbildung 49: Arbeitsteilung

- Wie in der Grafik oberhalb ersichtlich beschäftigte sich Herr Simon Hinterholzer vorrangig mit der Implementierung des Bewertungsvorganges. Dazu war es notwendig, dass er sich mit Besonderheiten der Bildverarbeitung vertraut machte und sich in die Materie „OpenCV“ und deren Verwendung einarbeitete. Hinzu kamen etwaige benötigte Kenntnisse in der Programmiersprache C++.

- Herr Alexander Ortner hingegen, war für die grafische Darstellung und Interaktion mit dem User verantwortlich. Um dies zu bewerkstelligen arbeitete er sich in die Programmiersprache Objective-C ein.

6. Aufwandsverteilung

In folgender Grafik kann man unsere Aufwandsverteilung begutachten. Diese teilt sich in verschiedene Unterpunkte auf. Die Dokumentation, Einarbeitung, Programmierung, Erstellung der Diplomschrift, Testen sowie Vorbereitung für die Präsentation.

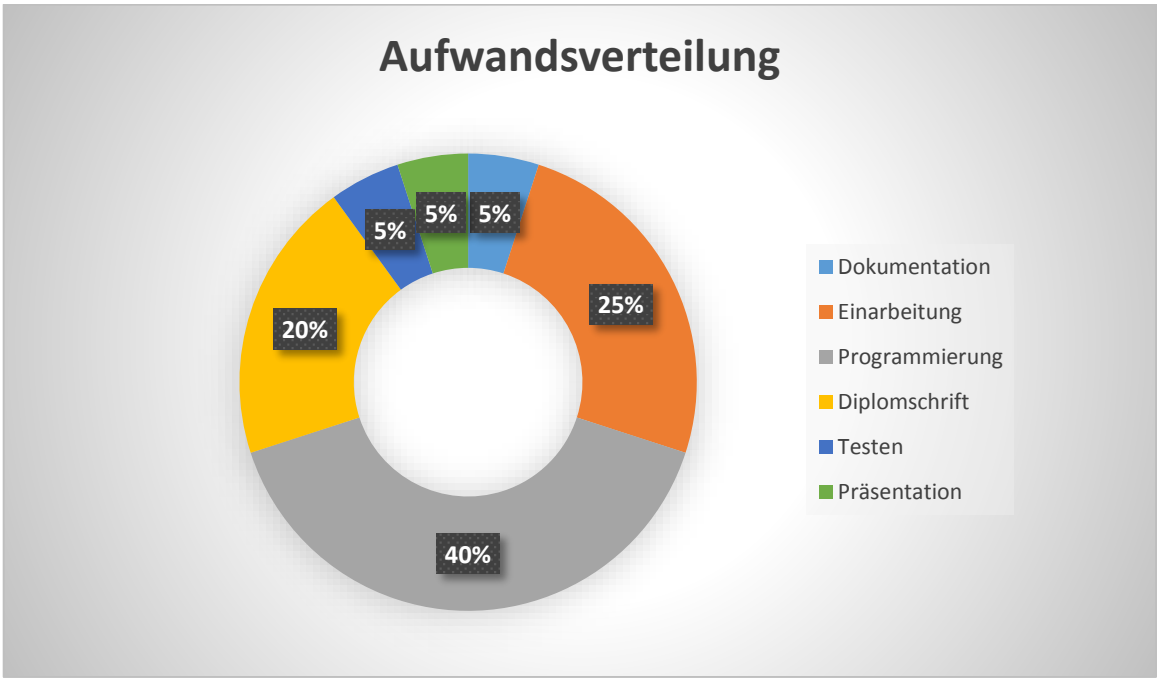

Abbildung 50: Aufwandsverteilung

- Die drei größten Teile die bei unserer Diplomarbeit angefallen sind, erstens die Einarbeitung in die für uns neue und vor allem fremde Materie von OpenCV, beziehungsweise iOS mit Objective-C. Zweitens die Programmierung des Programmes selbst welches den größten Teil der Diplomarbeit ausgemacht hat und drittens die Erstellung bzw. das Schreiben der Diplomarbeitsschrift.

- Die Dokumentation machte einen kleinen Aufwand in unserer Arbeit aus. Dies war die anfängliche Erstellung der ersten Dokumente, welche besonders zum Projektmanagement zählen, wie z.B. das Pflichtenheft, Projekthandbuch sowie Projektstrukturplan.

- Ein weiterer kleinerer Punkt war das Testen der Applikation, welches wir aufgrund fehlender Zeit vernachlässigt haben.

- Letzter Punkt ist noch die Präsentation der Diplomarbeit bei der schulinternen P@bs - Veranstaltung.

7. Glossar

MFFDPA	Mobile Face and Feature Detecting Photo Assessment Bezeichnung unserer Diplomarbeit
Slider	zu Deutsch Schieberegler Wird verwendet um eine Zahl in einer bestimmten Zahlenmenge festzulegen.
Timeline	Anzeige der Fotos nach dem Datum sortiert. zu Deutsch: Zeitlinie
GUI	Graphical User Interface zu Deutsch: Grafische Benutzeroberfläche
VM	Virtual Machine zu Deutsch: Virtuelle Maschine
Location	Bezeichnet einen gewissen Ort.
Instanz	kann man auch als Objekt einer Klasse bezeichnen.
Sourcecode	Der Quellecode eines Computerprogrammes
GPS	Global Positioning System zu Deutsch: Globales Positionsbestimmungssystem
Performance	zu Deutsch: Leistung
Voronoi-Diagramm	mathematisch; Zerlegung eines Raumes in Regionen
Dependencies	zu Deutsch: Abhängigkeiten
Laplace-Operator	mathematischer Operator; linearer Differentialoperator innerhalb der mehrdimensionalen Analysis →verwendet zur Schärfebestimmung
Haarcascades	Technik zur Objektsuche in einem Bild → Gesichtserkennung
Sobel-Operator	einfacher Kantendetektions-Filter; häufige Anwendung in der Bildverarbeitung; →verwendet zur Kontrastanalyse

8. Quellenverzeichnis

8.1. Dokumente und PDFs

- https://developer.apple.com/library/ios/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/iPhoneAppProgrammingGuide.pdf
- https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/ProgrammingWithObjectiveC/ProgrammingWithObjectiveC.pdf
- http://www.guettel.com/download/Face-Recognition--Gesichterkennung-Computer.pdf

8.2. Webseiten

8.2.1. Wikipedia

- http://de.wikipedia.org/wiki/OpenCV
- http://en.wikipedia.org/wiki/OpenCV
- http://de.wikipedia.org/wiki/Apple_iOS
- http://en.wikipedia.org/wiki/IOS
- http://de.wikipedia.org/wiki/Microsoft_Visual_Studio
- http://de.wikipedia.org/wiki/Xcode
- http://de.wikipedia.org/wiki/VMware_Workstation
- http://de.wikipedia.org/wiki/Kontrast
- http://en.wikipedia.org/wiki/Contrast_%28vision%29
- http://de.wikipedia.org/wiki/Sch%C3%A4rfe_%28Fotografie%29

8.2.2. Diverse sonstige Seiten

- http://opencv.org/
- https://developer.apple.com/library/ios/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/Introduction/Introduction.html
- https://pmougin.wordpress.com/2008/03/13/some-nice-features-of-the-objective-c-language/
- https://atomicobject.com/resources/handbook-of-software/objective-c-language

9. Abbildungsverzeichnis

10. Auszug Pflichtenheft

10.1. Musskriterien & Abgrenzungskriterien

10.1.1. Musskriterien

10.1.1.1. User-Interface

- Der User kann anhand der Timeline wählen, welche Fotos vom Programm durchsucht und bewertet werden
  - Der User kann wählen, wie viele Bilder er von dem Programm vorgeschlagen bekommt.
  - Nach dem Suchen der Bilder werden diese dem User in einer Art Galerie angezeigt.
  - Wird zu Beginn mit den ersten Fotos aufgefüllt
  - Wird während dem Suchen ständig aktualisiert
  - Der User kann aus der angezeigten Galerie Fotos löschen.
  - Der User kann der Galerie auch eigens gewählte Fotos hinzufügen.

10.1.1.2. Applikation

- Die Applikation läuft unter iOS.
- Die Applikation sucht Bilder aus den vorgegebenen Alben. Die Auswahl dieser geschieht aufgrund von verschiedenen Kriterien:
  - Schärfe des Bildes
  - Kontrast
  - Belichtung
  - Sind auf dem Bild Gesichter vorhanden?
  - Sind auf dem Bild vorhandene Gesichter gut erkennbar?

10.1.2. Abgrenzungskriterien

- Nur für die Plattform iOS ab der Version 7.x gedacht
- Es werden nur Ressourcen des Mobiltelefons genutzt, keine externen.

11. Auszug Projekthandbuch

11.1. Meilensteine

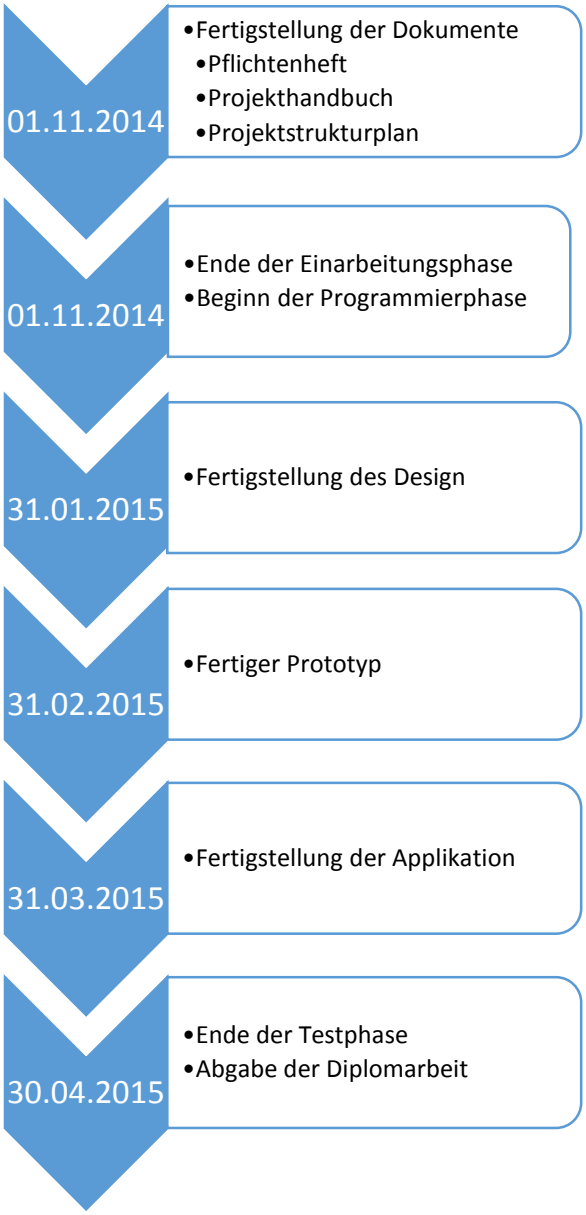

Abbildung 51: Meilensteine

Aufgrund unserer schlechten Zeiteinteilung, sowie Unterschätzung diverser Arbeiten, konnten wir einige Meilensteine nicht termingerecht einhalten und kamen somit ihn Verzug.

12. Resümee und persönliche Erfahrung

12.1.1. Resümee Simon Hinterholzer

Durch die Diplomarbeit konnte ich viele Erfahrungen sammeln.
Ich lernte das Arbeiten mit neuen Technologien und deren Vorzüge kennen und zu schätzen.
So machte ich die Erfahrung, dass zum Beispiel die Bibliothek OpenCV tolle Möglichkeiten bietet und ich mir auch vorstellen könnte, nach dem Abschluss der HTL Perg mit dieser, oder ähnlichen Produkten, zu arbeiten.

Andererseits habe ich auch gelernt, dass es zwar wichtig ist, einen genauen Plan zu haben, aber noch viel wichtiger ist es, diesen auch so umzusetzen wie man es sich beim Planen vorgestellt hat. Man sollte bei Projekten immer bedenken, dass die Zeit eventuell schneller vergeht als man denkt.

Ich bin froh, dass ich die Möglichkeit hatte diese Diplomarbeit zu absolvieren, da ich hoffe, dass mir die gemachten Erfahrungen im späteren Berufsleben helfen werden. Außerdem konnte ich so sehen, wieviel Arbeit wirklich in einem Softwareprojekt steckt, was ich bisher unterschätzt hatte.

12.1.2. Resümee Alexander Ortner

Diese Diplomarbeit war für mich reich an Erfahrung und eine große Herausforderung. Ich lernte wie wichtig es ist eine gute Zeiteinteilung zu haben, sowie diese strikt einzuhalten und dass man eine solche Aufgabe nicht auf die leichte Schulter nehmen kann.

Besonders die Arbeit mit der neuen Sprache Objective C sowie dem Betriebssystem iOS brachte mich an meine Grenzen. Es war für mich eine neue Erfahrung, da ich vor dieser Diplomarbeit noch nie eine dieser Technologien verwendet habe. Auch werde ich in näherer Zukunft diese nicht verwenden, da ich in der Programmiersprache Java eher meine Stärken sehe.

Alles in allem konnte ich mir durch diese Arbeit sehr viel neues Wissen aneignen und hoffe dieses auch im späteren Berufsleben einsetzen zu können.

Zu guter Letzt möchten wir noch einmal einen großen Dank an alle Personen aussprechen, die uns bei dieser Arbeit geholfen und unterstützt haben.

Simon Hinterholzer
&
Alexander Ortner

„Man merkt nie, was schon getan wurde, man sieht immer nur, was noch zu tun bleibt.“
Marie Curie