# Pflichtenheft: Bilderkennung

# Inhaltsverzeichnis

[1. Zielbestimmung](#h.3dtrjt8e33of)

[1.1 Muss-Kriterien](#h.gfj1vhfcj863)

[1.2 Soll-Kriterien](#h.cglwz74rc9og)

[1.3 Kann-Kriterien](#h.fv6ygx7p8abi)

[1.4 Abgrenzungskriterien](#h.ieiswnk5jhj5)

[2 Produkteinsatz](#h.ejt6x6fo8an3)

[2.1 Anwendungsbereiche](#h.8f7yqs7t6vmf)

[2.2 Zielgruppe](#h.x02l0f8967ef)

[2.3 Betriebsbedingungen](#h.ivyd8syl8blf)

[3. Produktumgebung](#h.qjm3apu98qmu)

[3.1 Software](#h.z56pejo0a154)

[3.2 Hardware](#h.qj49wkv4ac7s)

[4. Produktfunktionen](#h.acao0flu6xu8)

[5. Produktdaten](#h.9lisx19vl4f2)

[5.1 Non-persistente Daten](#h.kp2pld4a0n9y)

[5.2 Persistente Daten](#h.2mhldc49dafm)

[6. Produktleistungen](#h.eewuvk2k2vs3)

[7. Qualitätsbestimmungen](#h.xogg4nfwcopn)

[8. Testszenarien und Testfälle](#h.wjvwcispp4f6)

[9. Benutzeroberfläche](#h.6pocof99bv28)

[10. Entwicklungsumgebung](#h.wvcjguiwzd7)

[10.1 Software](#h.su1yj8ie2827)

[10.2 Hardware](#h.7aphb7ugb9m4)

# 1. Zielbestimmung

Das Ziel der Studienarbeit ist eine wissenschaftliche Diskussion über einsetzbare Verfahren zur Bildanalyse um Objekte innerhalb eines Bildes zu Erkennen und Kategorisieren. Zusätzlich soll eine Anwendung entwickelt werden, welche eine Server-Einbettung ermöglicht um eine solche Objekterkennung innerhalb einer Serveranwendung umsetzen zu können.

Muss-Kriterien sind auf jeden Fall verpflichtend, Soll-Kriterien sind so weit wie möglich zu implementieren, Kann-Kriterien sind optional und werden implementiert, wenn noch Zeit übrig ist.

## 1.1 Muss-Kriterien

* Die Anwendung enthält ein funktionsfähiges Neuronales Netz.
* Dieses neuronale Netz ermöglicht die Erkennung und Lokalisierung von Objekten innerhalb eines Bildes. Aufgrund dieser Funktionalität wird dieses neuronale Netz in folgenden als *object recognition application* bezeichnet.
* Das neuronale Netz enthält eine Eingabe- und Ausgabe-Schnittstelle für Bilder.
* Die Architektur des neuronalen Netzes muss dokumentiert sein.
* Das neuronale Netz kann mit Hilfe von „supervised-training“ trainiert werden.
* Die Anwendung muss mit einer Datei zur grundlegenden vortrainierten Intelligenz ausgeliefert werden.
* Der Algorithmus zum Training (supervised) der Daten soll das ihm gegebene Trainingsmaterial maximal ausnutzen.

## 1.2 Soll-Kriterien

* Das bereits trainierte neuronale Netz soll nachträglich mit weiteren Daten trainierbar sein.
* Die erkannten Objekte sollen kategorisiert werden können. Hierzu soll eine eigene interne neue Anwendung entwickelt werden, welche ebenfalls ein vergleichbares Verfahren des maschinellen Lernens nutzt. Zur Abgrenzung soll dieses im Folgenden als *object identification application* bezeichnet werden.
* Die Anwendung erlaubt eine parallelisierte Verarbeitung mehrerer zu untersuchenden Bilddateien.
* Hierzu verfügt diese über eine Konfigurationsschnittstelle, um die maximale Zahl an zu verwendeten Prozessen *(Threads)* festzulegen.

## 1.3 Kann-Kriterien

* Die Anwendung soll in einen Web-Server eingebunden werden, welcher eine Schnittstelle zur Eingabe von Bilddaten, sowie eine Schnittstelle zur Ausgabe der analysierten Daten ermöglicht.
* Es wird eine iPhone-App zur Fotographie entwickelt, welche die erfassten Bilder an den oben genannten Web-Server übertragen kann. Weiterhin soll diese die von der Server-Anwendung erarbeiteten und ausgegebenen Daten empfangen und entsprechend darstellen können.
* Die Umsetzung des Trainings enthält eine Methode zur Kalkulation auf einer Grafikkarte, anstelle des Prozessors, um somit eine höhere Berechnungsgeschwindigkeit zu realisieren.
* Die Anwendung kann auch lokal auf einem iPhone (iOS9+) ausgeführt werden.

## 1.4 Abgrenzungskriterien

* Das Bilderkennungs-Anwendung enthält lediglich das neuronale Netz mit den trainierbaren Parametern, eine Ein- und Ausgabeschnittstelle zur Bildanalyse, sowie eine Möglichkeit eine Menge von gekennzeichneten Daten zum Training zu verwenden.
* Die Web-Server-Anwendung ermöglicht selbst keine Bilderkennung. Es wird lediglich eine Ausführung der oben beschriebene „object recognition & identification“ – Anwendung gewährleistet.
* Es gibt keine Einschränkung für den Benutzer, es gibt keine kostenpflichtigen Erweiterungen.
* Es können nur Objekte wiedererkannt werden, welche bereits durch ein umfangreiches Trainingsverfahren dem neuronalen Netz vorgestellt wurden. Um einen Vergleichsfaktor zur Einschätzung der Leistungsfähigkeit des neuronalen Netzes zu generieren, soll eine feste Menge an Objekten zur Trainings- und Testphasen festgelegt werden. Anhand dieser orientiert sich die Entwicklung des neuronalen Netzes. Für die Erkennung weiterer Objekte wird vom Endanwender die selbstständige Nutzung der Trainingsschnittstelle vorausgesetzt.
* Weiterhin wird eine den Trainingsdatensätzen ähnliche Qualität der Bilder zur Analyse vorausgesetzt.

# 2 Produkteinsatz

## 2.1 Anwendungsbereiche

Die Anwendung dient der automatisierten Verarbeitung von unstrukturierten Informationen in Form von Bildern, um eine hohe personelle Belastung zur iterativen Verarbeitung dieser zu vermeiden. Dies soll weiterhin die automatische Suche nach Begriffen von nicht näher bezeichneten Bildern ermöglichen. Denkbar wäre eine nachträgliche Indexierung der Ermittelten Objektinformationen um die Anwendung in eine Bilder-Datenbank zu integrieren.

## 2.2 Zielgruppe

Die App ist sowohl für den privaten, als auch für den kommerziellen Gebrauch geeignet. Insbesondere Entwickler und Betreiber von Bilder-Datenbanken erhalten somit eine zeitsparende Möglichkeit, Bilder nicht ihren Informationen nach manuell zu beschreiben, sondern ein automatisiertes Verfahren zur Ermittlung der benötigten Informationen zu nutzen. Hierzu wird eine automatisierte textbasierte Suchbegriff-Indizierung ermöglicht.

## 2.3 Betriebsbedingungen

* Der Betrieb der Anwendung setzt eine Common Intermediate Language RunTime voraus.
* Der Betrieb des Servers setzt eine .NET-fähige Serverumgebung mit Internet-Verbindung voraus.
* Der Betrieb der App setzt eine funktionierende Kamera des iPhones voraus.

# 3. Produktumgebung

## 3.1 Software

Die Anwendung wird für die .NET Laufzeitumgebung der Version 4.5 oder höher entwickelt.

## 3.2 Hardware

Unterstützt wird jede Hardware, welche über eine Common Intermediate Language RunTime verfügt.

Weiterhin wird eine Hardware-Umgebung vorausgesetzt, welche dem Umfang der Anwendung genug physikalische und virtuelle Ressourcen zur Verfügung stellt.

# 4. Produktfunktionen

|  |  |
| --- | --- |
| /F10/ | Einstellungen ändern: Anzahl der Threads |

Der Benutzer der Anwendung kann die Anzahl der zu benutzenden gleichzeitig arbeitenden Threads über eine Schnittstelle festlegen. Als Default-Wert wird hierbei die von der .NET RunTime maximal zur verfügbar gestellten virtuellen Prozessorkernen.

|  |  |
| --- | --- |
| /F20/ | Schnittstelle zur Eingabe von Bildinformationen |

Das neuronale Netz verfügt über eine universelle Eingabe-Schnittstelle über welche Bild-Informationen eingelesen werden können. Um die Anwendung kompatibel zu halten, ist neben den üblichen Bildformaten *Portable Network Graphics*, *JPEG*, *Tagged Image File Format* und *Bitmap* auch eine Reindatendarstellung in Form eines zweidimensionalen Daten-Arrays möglich.

|  |  |
| --- | --- |
| /F30/ | Analysieren von Bildinformationen |

Die der Anwendung zur Verfügung gestellten Bildinformationen sollen durch einen internen Mechanismus gemäß den abgebildeten Objekten ausgewertet werden. Hierbei werden die auf dem Bild vorhandenen Objekt erkannt und in innerhalb der Form eines geometrischen Rechtecks erfasst. Die Rechtecke werden hierbei durch die Koordinaten auf dem Bild beschrieben.

|  |  |
| --- | --- |
| /F40/ | Schnittstelle zur Ausgabe der Analyse |

Die analysierten Daten sollen über eine Ausgabe-Schnittstelle an den ursprünglich aufrufenden Prozess übertragen werden. Hierbei sollen die Informationen der Objekte in eine Klasse gekapselt werden, welche die vier Koordinatenpunkte des Rechtecks, sowie optional die Informationen der zu erkennenden Objekte beinhält.

|  |  |
| --- | --- |
| /F40/ | Initialisierung der Kanten der *object recognition* |

Der Anwender erhält die Möglichkeit die Intelligenz, abgebildet durch die Kantengewichte, mit einem Standart-Wert zu initialisieren. Hierbei wird keine trainierte Version, sondern der tatsächlich untrainierter Initialwert „1“ auf alle Kantengewichte gesetzt.

|  |  |
| --- | --- |
| /F50/ | Trainieren des neuronalen Netzes |

Über eine Schnittstelle, welche jeweils ein Bild, sowie die hierzu korrekt analysierten Informationen enthält, soll das neuronale Netz trainiert werden können. Hierzu wird ein supervised-learning Verfahren angewendet.

|  |  |
| --- | --- |
| /F60/ | Export der Intelligenz |

Die in den Kantengewichten hinterlegte Intelligenz soll in eine namentlich freiwählbare Binär- oder XML-Datei exportierbar sein.

|  |  |
| --- | --- |
| /F70/ | Import der Intelligenz |

Über eine Schnittstelle werden die oben genannten Binär- oder XML-Dateien zur Initialisierung des neuronalen Netzes importiert.

# 

# 7. Qualitätsbestimmungen

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Anforderung* | *sehr wichtig* | *wichtig* | *neutral* | *irrelevant* |
| Zuverlässigkeit |  |  | x |  |
| Stabilität |  | x |  |  |
| Performance |  |  |  | x |
| Benutzerfreundlichkeit |  |  |  | x |
| Oberfläche |  |  |  | x |
| Funktionalität | x |  |  |  |
| Portabilität |  | x |  |  |
| Wartbarkeit |  |  | x |  |

# 8. Testszenarien und Testfälle

Es werden zwei Datenmengen von Bilder unterschieden. Die erste sei hierbei die Trainingsdatenmenge, welche alle Daten zum trainieren einsetzt. Die zweite Datenmenge ist die Testdatenmenge, welche zur Überprüfung der Leistungsfähigkeit der Anwendung genutzt werden soll. Dieser Leistungstest ist insbesondere daher wichtig, da dieser den Faktor zur Bemessung und somit Vergleichsmerkmal innerhalb der Entwicklung abbildet. Dieser Faktor gilt als maßgeblicher Leitfaden für die Entwicklungsrichtung des neuronalen Netzes.

Hierfür soll die standardisierte *NORB Object Recognition DataSet* (nachfolgend *NORB*) Datenmenge verwendet werden, welches sowie Trainingsdaten als auch Testdaten enthält. Durch den hohen Umfang dieser können zwar Leistungstest feingranular durchgeführt werden, jedoch erhöht eine hohe Menge an Daten ebenfalls die zeitliche Dauer des Trainings. In der frühen Entwicklungsphase soll vorerst ein grobgranularer Vergleich ausreichen. Daher soll in jener Phase vorerst die minimierte Version *small NORB* genutzt werden, um eine effizientere Entwicklung zu ermöglichen.

Alle entwickelten App-Funktionen werden umfangreich getestet. Getestet wird:

* Scannen einer Visitenkarte via Kamera
* Scannen einer Visitenkarte via Fotogalerie
* Korrektur der erkannten Informationen durch Benutzer
* Speichern eines neuen Kontakts
* Bearbeiten eines vorhandenen Kontakts
* Ändern der Einstellungen
* Starten, Pausieren und Beenden der App ohne Datenverlust
* Abbruch eines Scanvorgangs via Kamera
* Abbruch eines Scanvorgangs via Fotogalerie
* Abbruch bei der Korrektur der erkannten Informationen

# 

# 

# 9. Schnittstellen

# 10. Entwicklungsumgebung

## 10.1 Software

Die App wird in der Programmiersprache *Swift und Objective-C* entwickelt. Verwendete Softwares sind:

* OS X 10.10 (Yosemite)
* Xcode 6.0
* Github Desktop zur git Versionskontrolle
* Balsamiq Mockups for Desktop 2

## 10.2 Hardware

Zum Testen stehen die folgenden Geräte zur Verfügung:

* iPhone 5
* iPad (4. Generation)