HTWK Leipzig
Fachbereich IMN
Vorlesung Softwaretechnik
Wintersemester 04/05
Datum: 13.01.04

LV Softwaretechnik

Pflichtenheft des Projekts „wxOCR“

Matthias Jauernig
Michael Lahl
03IN1

1. Zielbestimmung:

Mit der Anwendung „wxOCR“ soll ein plattformunabhängiges Open-Source-Programm entwickelt werden, welches eine Texterkennung aus Bilddateien heraus mit Hilfe eines Neuronalen Netzes realisiert.

1.1 Musskriterien:

- Gliederung in 2 Teilprogramme: mit dem ersten (wxOCR_netTrainer) wird ein Neuronales Netz (siehe 14. zur Begriffsklärung) auf bestimmte Mustersätze angelernt, was durch den Entwickler geschehen sollte, da dieser Prozess eine gewisse Zeit dauern kann; mit dem zweiten (wxOCR_main oder allgemein wxOCR) kann ein Endbenutzer die eigentliche Texterkennung durchführen
- Der Benutzer kann ein zu verarbeitendes Bild wählen
- Text (= Buchstaben + Zahlen + Satzzeichen, siehe 14. zur Einschränkung) wird aus dem Bild erkannt und ausgegeben
- Erkannter Text kann markiert, kopiert und gespeichert werden
- Zuverlässige Erkennung der Schrift Arial bzw. einer Schriftart ohne Serifen, Schriftgröße 14pt, Schriftbreite normal, Schriftlage normal, Schriftfarbe schwarz auf weißem Hintergrund bei exakt horizontaler Ausrichtung der Textzeile
- Open-Source, Quelltext unter GPL
- Plattformunabhängig, inkl. betreffender GUI (speziell: Windows/Linux)
- Deutschsprachige Benutzungsoberfläche (allerdings mit Hinblick auf Internationalisierung eine an Englisch angelehnte Variablen-/Klassen-/Funktionen-Benennung)

1.2 Wunschkriterien:

- Projektseite auf sourceforge.net erstellen
- Unterschiedliche Textarten/-größen und -farben können erkannt werden
- Erkennung von Text aus bis zu einem gewissen Grad „schiefen“ Textzeilen
- Behandlung von Bildern/Grafikelementen als nicht erkennbaren Text
- Bis zu gewissem Grad kann Text mit Kerning erkannt werden (siehe 14. zur Begriffsklärung)
- Englischsprachige Benutzungsoberfläche

1.3 Abgrenzungskriterien:

- Kein Erkennen von Handschrift oder von Arial „zu stark“ abweichenden Schriftarten
- Kein Erkennen von Schrift, welche Ligaturen besitzt (siehe 14. zur Begriffsklärung)
- Keine Schaffung eines absolut ausgereiften Systems zur Texterkennung, welches mit kommerziellen Lösungen konkurrieren könnte

2. Produkteinsatz:

Das Produkt dient dem Nutzer dazu, ein (z.B. von ihm gescanntes) Bild in vom Rechner verarbeitbaren Text umzuwandeln und in einer Textdatei speichern zu können.

2.1 Anwendungsbereiche:

- Bildverarbeitungs-/Bildaufbereitungsbereich

2.2 Zielgruppen:

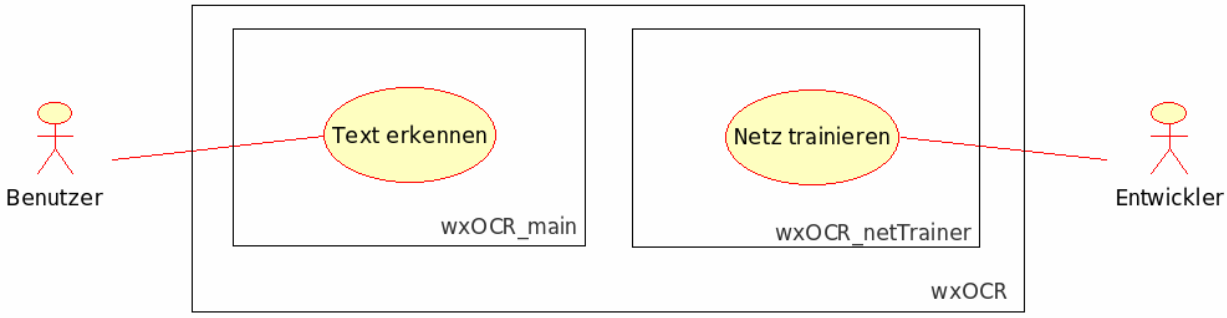
- Endbenutzer von Scannersoftware, daher intuitive/benutzerfreundliche Umgebung
- Software-Entwickler und Studenten informatischer Studiengänge, die Teile der Software in ihren Projekten nutzen wollen

2.3 Betriebsbedingungen:

- Büro-/Heimumgebung
- Kurze Betriebszeiten, Anwendung: selten

3. Produktübersicht:

Use-Case-Diagramm (Muss-Kriterien)



4. Produktfunktionen:

4.1 Geschäftsprozesse:

Geschäftsprozess	Text erkennen
Ziel	Text wird aus Bild erkannt
Kategorie	Primär
Vorbedingung	Neuronales Netz muss mit wxOCR_netTrainer antrainiert worden sein
Nachbedingung Erfolg	Benutzer hat erkannten Text ausgegeben bekommen
Nachbedingung Fehler	Fehlermeldung, System im Ausgangszustand
Akteure	Benutzer
Auslösendes Ereignis	Benutzer startet die Software
Beschreibung	1.) Auswahl der Bilddatei 2.) Anwahl des Menüpunkts „Text erkennen“ 3.) Benutzer kann erkannten Text markieren, kopieren oder speichern
Alternativen	1a) Fehlermeldung, falls Bilddaten inkorrekt 2a) Fehlermeldung, falls Neuronales Netz nicht geladen werden konnte bzw. kein Text erkannt wurde

Geschäftsprozess	Netz trainieren
Ziel	Neuronales Netz wird auf Mustersatz antrainiert
Kategorie	Primär
Vorbedingung	Es sind auf der Festplatte zu trainierende Mustersätze vorhanden
Nachbedingung Erfolg	Neuronales Netz wurde trainiert und abgespeichert
Nachbedingung Fehler	Fehlermeldung an Entwickler, System in Zustand vor Ausführen des fehlerhaften Befehls
Akteure	Entwickler
Auslösendes Ereignis	Benutzer startet Applikation
Beschreibung	1.) Auswahl einer Bilddatei, die die Muster enthält bzw. einer zuvor abgespeicherten Musterdatei 2.) Neuronales Netz trainieren lassen 3.) Trainiertes Neuronales Netz abspeichern
Alternativen	1a) Fehlermeldung, falls Bilddaten inkorrekt 2a) Erkannte/geladene Muster in Datei speichern 2b) Geladenes Muster anzeigen lassen

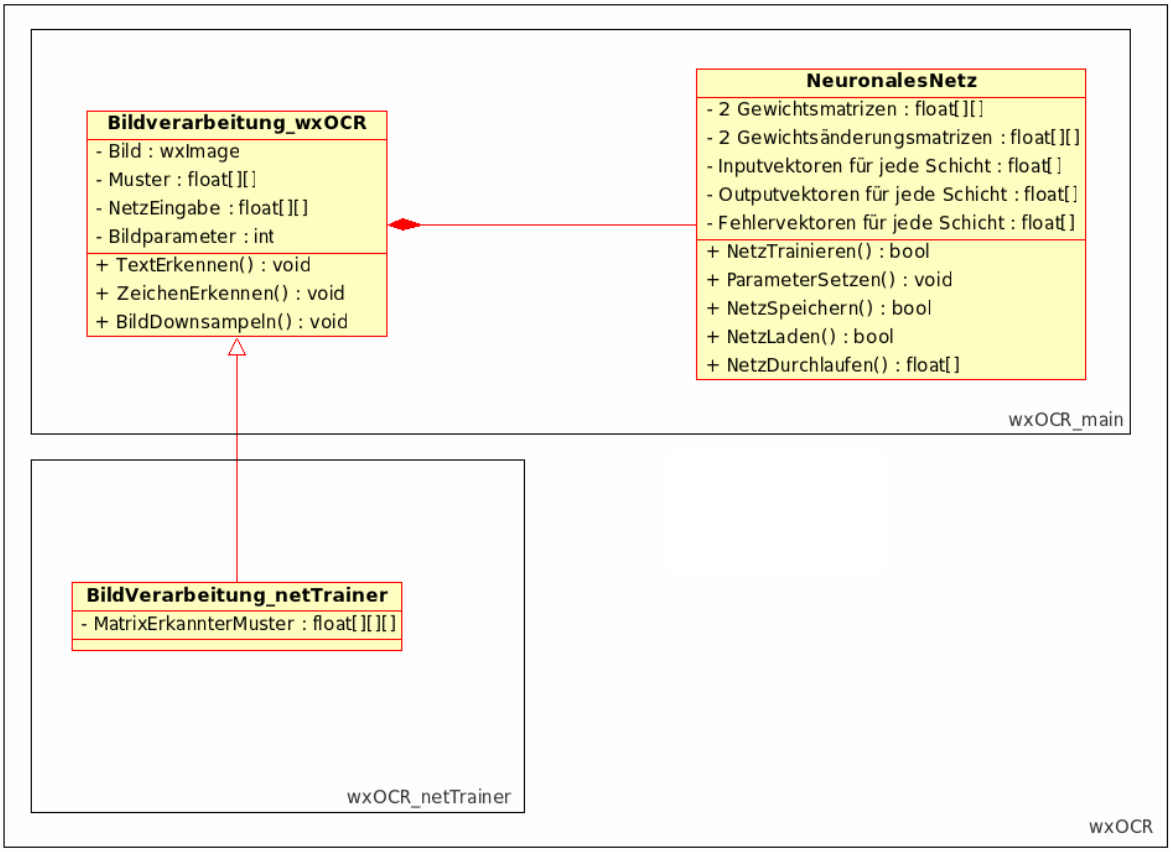
5. Produktdaten:

5.1 Dauerhaft zu speichernde Daten:

- Speicherung in Dateien, keine Datenbank-Anbindung
- Neuronales Netz
- Muster in Bilddatei oder Musterdatei

5.2 Zur Laufzeit anfallende Daten:

5.2.1 Klassendiagramm:



5.2.2 Bilddaten:

- Bilddaten: 1 Objekt für das Bild mit Farbwerten, Bildgröße etc.
- Musterdaten: 1 2-dim. float-Matrix, welche ein Muster mit seinen Graustufenwerten speichert
- 1 Matrix, welche die Netzeingabe als fixe Größe aus der dynamischen Matrix eines Musterdatums repräsentiert

5.2.3 Daten des Künstlichen Neuronalen Netzes:

- Ein Künstliches Neuronales Netz mit 3 Schichten (siehe 14. zur Begriffsklärung)
- 2 Gewichtsmatrizen, Ein-/Ausgabe- und versteckte Neuronen
- weitere Daten zur internen Organisation des Netzes (Matrizen, Vektoren, Skalare)

5.2.4 Textdaten:

- 1 Zeichenkette für den erkannten Text

6. Produktleistungen:

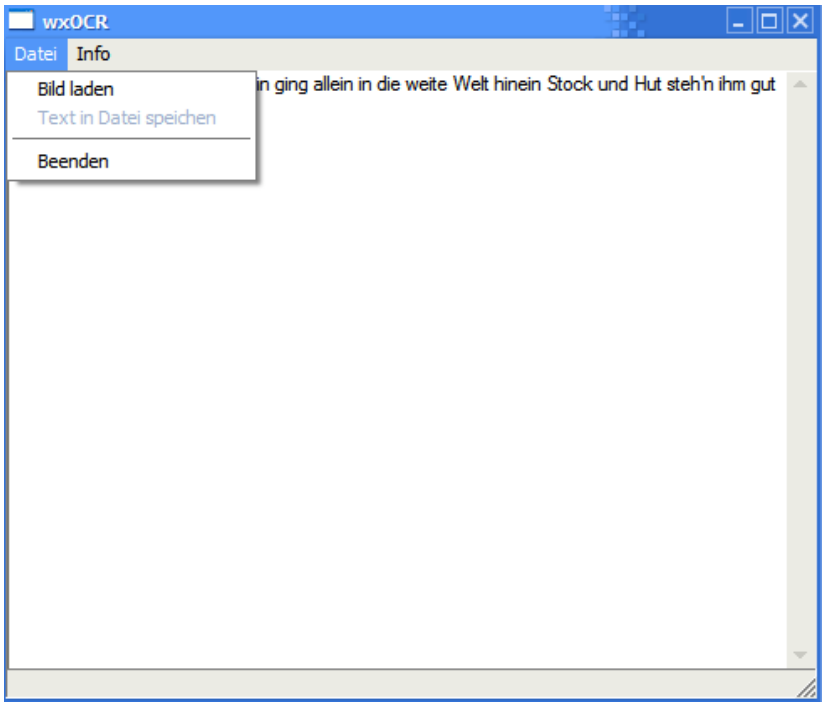
- Als Künstliches Neuronales Netz werde ein vorwärts gerichtetes Netz verwendet. Da der Lernprozess mittels Backpropagation-Algorithmus eine dem Benutzer unzumutbar lange Zeitspanne erfordern würde, wird dieser Prozess vor der Nutzung des Programms wxOCR_main (bzw. wxOCR) vom Entwickler durch ein Hilfsprogramm (wxOCR_netTrainer) vorgenommen, welches das Netz repräsentative Muster-Datensätze lernen lässt und die daraus entstehenden Gewichtsmatrizen in einer Datei speichert. Diese wird dann von wxOCR geladen, sodass sich kein signifikanter Zeitverzug bemerkbar machen sollte.
- Die Erkennung des Textes aus der Bilddatei sollte bei einer A4-Seite nicht länger als 10s dauern (abhängig von der Textlänge).

7. Qualitätsanforderungen: (in Bezug auf beide Teilapplikationen, primär wxOCR_main)

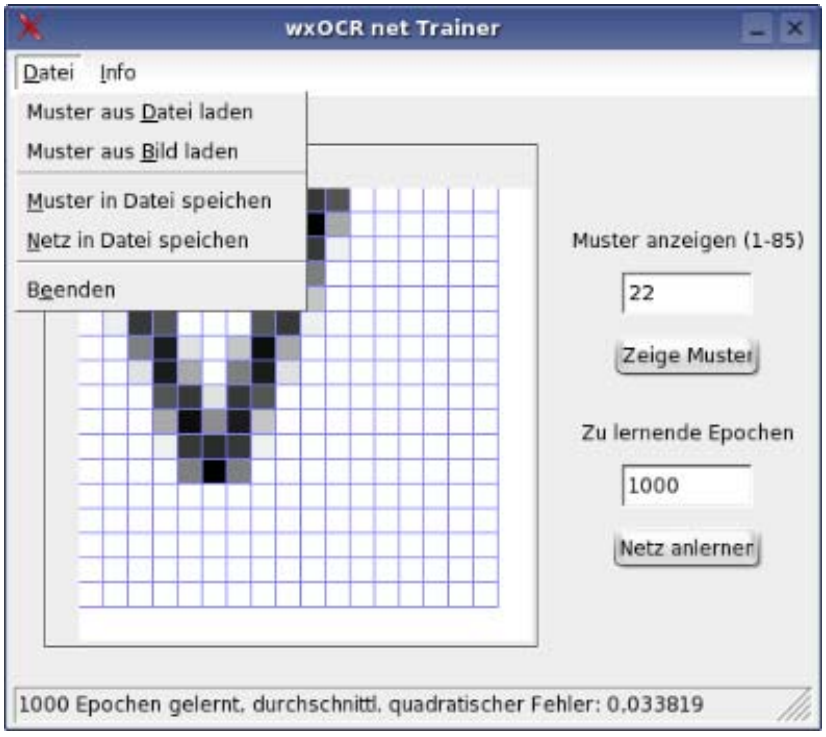
Produktqualität	sehr gut	gut	normal	nicht relevant
Funktionalität				
Richtigkeit		X		
Stabilität			X	
Zuverlässigkeit				
Reife			X	
Fehlertoleranz			X	
Benutzbarkeit				
Verständlichkeit	X			
Erlernbarkeit		X		
Bedienbarkeit	X			
Effizienz				
Zeitverhalten		X		
Verbrauchsverhalten			X	
Änderbarkeit				
Analysierbarkeit	X			
Modifizierbarkeit	X			
Prüfbarkeit	X			
Übertragbarkeit				
Anpassbarkeit	X			
Konformität	X			
Austauschbarkeit	X			

8. Benutzungsoberfläche:

- Die Bedienungsoberfläche ist auf Mausbedienung auszulegen.
- wxOCR_main: (hier: Prototyp unter Windows)



- wxOCR_netTrainer: (hier: Prototyp unter Linux mit KDE 3.3)



- Bis auf Dialoge zur Dateiauswahl und die Anzeige von Informationen sonst keine weiteren Dialoge, auf deren Darstellung hier näher eingegangen werden müsste.

9. Nichtfunktionale Anforderungen:

- Verwendung der GUI-Bibliothek wxWidgets für Erreichen der Plattformunabhängigkeit (siehe 14. zur Begriffsklärung)
- Bereitstellung einer Dokumentation zur Benutzungsoberfläche und einer Entwickler-Dokumentation zu den einzelnen implementierten C++ Klassen
- Changelog anlegen zur Speicherung einer Änderungs-Historie

10. Technische Produktumgebung:

Das Produkt läuft stets in einer grafischen Umgebung.

10.1 Software:

- Primär Windows 9x/NT/200x/XP
- Linux mit KDE/GNOME und GTK+
- Sekundär alle weiteren von wxWidgets unterstützten Betriebssysteme (Unix mit Motif, OS/2, MacOS)

10.2 Hardware:

- IBM PC kompatibel

11. Spezielle Anforderungen an die Entwicklungsumgebung:

11.1 Software:

- C++ - Compiler
- Installierte Bibliothek wxWidgets v2.4.2

12. Gliederung in Teilprodukte:

- Teilprodukt 1: Realisierung der Software wxOCR_netTrainer als Basis für das Hauptprogramm wxOCR_main (wxOCR); mit diesem „Trainer“ wird das Neuronale Netz auf einen gewissen Basisdatensatz an Mustern trainiert
- Teilprodukt 2: Realisierung der Software wxOCR_main, welche die eigentliche Benutzerschnittstelle darstellt und die Bilderkennung ausführt
- wxOCR Version <1: Implementierung der Musskriterien
- wxOCR Version <2: Inkrementelle Implementierung der Wunschkriterien

13. Ergänzungen:

- wxWidgets-unabhängiger Teil nach ANSI-C++ Norm
- Quelltext-Lizenz: GPL

14. Glossar:

- *Künstliches Neuronales Netz (KNN):*
  Für unsere Software soll ein vorwärts gerichtetes KNN verwendet, welches mit dem Backpropagation-Algorithmus inklusive Lernrate und Impulsfaktor trainiert werden soll. Bei einem so abgegrenzten KNN handelt es sich um einen gerichteten Graphen, dessen Kanten gewichtet sind (Speicher) und dessen Knoten (Prozessoren) die Neuronen darstellen, die eine Eingabe verarbeiten und mit den Gewichten entsprechend aufsummiert eine Ausgabe liefern. Unser Neuronales Netz soll aus 3 Schichten bestehen: einer Eingabeschicht von Neuronen, einer verborgenen Schicht und einer Ausgabeschicht. Jede dieser Schichten besitzt eine problemabhängige Anzahl von Neuronen, je nachdem wie die jeweiligen Datenvektoren dimensioniert sind. Ein- und Ausgabeschicht stellen dabei die Schnittstelle zur Umwelt (dem aufrufenden Programmteil) dar, die verborgene Schicht dient zur internen Speicherung und Verarbeitung von Daten im Netz (i.A. können auch mehrere verborgene Schichten existieren). Vorwärts gerichtet („feed forward“) heißt, dass jedes Neuron einer Schicht mit jedem Neuron der vorigen Schicht (also Eingabeschicht mit verborgener und diese mit der Ausgabeschicht) verbunden ist und es keine Rückführungen gibt (was in anderen Netzmodellen zur Anwendung kommt). Legt man jetzt an die Eingabeneuronen einen geeigneten Datenvektor an, so liefert die Ausgabeschicht die dazu gehörige Ausgabe. Durch Anpassen der Gewichte kann man diese Ausgabe beeinflussen. „Lernen“ im Sinne von Backpropagation bedeutet, dass ein überwachtes Lernen stattfindet. Man legt eine Eingabe an und greift die Ausgabe ab. Daraus errechnet sich ein Fehler zur tatsächlichen Ausgabe und entsprechend diesem Fehler kann man die Gewichte im Netz anpassen. Lernt man so zyklisch eine geeignete Menge von Datenvektoren an, so minimiert sich der Fehler nach einer gewissen Anzahl von Durchläufen, zu einer gelernten Eingabe kann das Netz somit eine korrekte Ausgabe liefern. Neuronale Netze sind vor allem deshalb interessant, weil sie „verrauschte“ Eingabedaten korrekt zuzuordnen wissen. Daher haben wir uns auch entschieden sie für diese Software einzusetzen.
- *„Text“:*
  Soll in unserem Fall folgendes Alphabet sein:
  A={'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z', 'Ä', 'Ö', 'Ü', 'ß', 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z', 'ä', 'ö', 'ü', '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', '.', '?', '!', ',' ,'-', '+', '=', '*', '/', '(', ')', '”', '<', '>', '%', '&'}, card(A)=85
- *Kerning:*
  Dieser Begriff, auch „Unterschneiden“ genannt, findet Anwendung zur Verbesserung des Schriftbildes. Dabei werden Buchstaben näher aneinander gestellt, sodass sie sich vertikal überschneiden. Dies fördert den Textfluss und verstärkt die Lesbarkeit der Textzeile.
- *Ligaturen:*
  Stellen wie das „Kerning“ eine Methode zur Verbesserung des Schriftbildes dar. Dabei werden 2 oder 3 Buchstaben so eng aneinander gestellt, dass sie miteinander verschmelzen, z.B. „fl“ oder „qj“. Hier ist es schwierig einen Algorithmus zu entwickeln, der diese beiden Zeichen auch wirklich nicht als nur ein Zeichen erkennt.
- *wxWidgets:* (http://wxwidgets.org)
  Hierbei handelt es sich um eine Art Grafikbibliothek zur Erzeugung von plattformunabhängigen GUIs. Genauer gesagt ist es ein „Wrapper“, der je nach Betriebssystem GUI-Aufrufe in die jeweiligen Betriebssystem-Calls umwandelt. So wird unter Windows in MFC-, unter Linux in GTK+- und unter Unix in Motif-Calls gewandelt. Dies macht diese frei verfügbare Bibliothek zu einem mächtigen Werkzeug der plattformübergreifenden Programmierung.