FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Dokumentenkamera und Belegleser für „SISI“

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Medizinische Informatik

eingereicht von

Christian Hinterer
Matrikelnummer 0927843

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung
Betreuer: Ao.Univ.Prof. Dipl.-Ing. Dr. techn. Wolfgang Zagler
Mitwirkung: Projektass. Dipl.-Ing. Christian Beck

Wien, 02.11.2016

(Unterschrift Verfasser) (Unterschrift Betreuer)

Erklärung zur Verfassung der Arbeit

Hiermit erkläre ich, dass ich diese Arbeit selbstständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Christian Hinterer

7022 Schattendorf, 02.11.2016

Ort, Datum, Unterschrift

Kurzzusammenfassung

Bargeldloser Zahlungsverkehr, speziell e-banking, birgt für viele Menschen, insbesondere ältere, die es zeitlebens gewohnt waren, bar zu bezahlen und ihre Bankgeschäfte direkt mit einem bzw. einer Bankangestellten abzuwickeln, große Probleme. Im Rahmen dieser Diplomarbeit wird gezeigt, wie das am Zentrum für Angewandte Assistierende Technologien der TU Wien entwickelte „Senioren Terminal - SISI“ bzw. LUI erweitert werden kann, um älteren Menschen Unterstützung zukommen zu lassen, wenn diese ihren Zahlungsverkehr auf erleichterte Art und Weise von zu Hause aus erledigen wollen. Kern der Arbeit ist es, Belege, welche in Papierform vorhanden sind, mit einer einfachen Webcam zu erfassen, mittels Bildbearbeitung die Qualität zu erhöhen, mittels Bilderkennung den Beleg selbst sowie dessen Layout zu erkennen und mittels OCR (Optical Character Recognition) den benötigten Inhalt aus dem Bild zu extrahieren. Ziel ist eine eigenständige Lösung, welche allerdings über eine definierte Schnittstelle als Treiber von dem „Senioren Terminal – SISI“ bzw. der darauf laufenden universellen Benutzer-Schnittstelle LUI verwendet werden kann. Die dabei an LUI übertragenen Informationen können dann in weiteren Schritten, unter Absprache mit Kreditinstituten und unter Verwendung von definierten Schnittstellen, zu e-banking-Applikationen exportiert werden. Dies ist allerdings von dieser Diplomarbeit abgegrenzt.

Die als Treiber implementierte Lösung ist mit Embarcadero Delphi implementiert, die Kommunikation mit den „Consumer-Applikationen“ erfolgt unter Verwendung von „Named Pipes“. Im produktiven Betrieb läuft der Treiber als Prozess ohne grafische Benutzeroberfläche, diese kann allerdings im Modus „Testbetrieb“ eingeblendet werden, um die einzelnen Prozessschritte nachvollziehen zu können. Für die Durchführung der OCR-Logik sind einige vorhandene Komponenten evaluiert worden, um eine effiziente Komponente in Relation zu den Kosten zu wählen.

Neben der Diplomarbeit als Studie der Machbarkeit, Dokumentation der Erkenntnisse, sowie Dokumentation der Implementierung, besteht das Ergebnis aus dem Treiber selbst.

Zudem wurde eine Studie mit der zu erwartenden Benutzergruppe, nämlich den Benutzern und Benutzerinnen von LUI durchgeführt, um die Benutzerfreundlichkeit sowie die Benutzerakzeptanz abschätzen zu können.

Abstract

Electronic-Commerce, especially Electronic Banking includes several usage challenges, which are difficult to handle for some users, in particular for older people. These users are used to do cash payments and to handle their financial business with the support of banking employees directly at the bank. Scope of this thesis is to show, how the “Senioren Terminal – SISI” respectively LUI, developed at the Centre for Applied Assistive Technologies of the University of Technology Vienna, can be extended for giving support to older people to handle their financial businesses from their homes with an easier method than the usual e-banking-applications. Core of the thesis is to capture banking receipts with a common webcam, increasing the pictures quality, filtering the receipts layout and most important using OCR (optical character recognition) to extract the needed data. The aim is to develop a driver, which communicates with LUI, respectively with the universal user interface LUI, which is installed on the terminal, using a defined interface. Thereafter, the data can be exported to e-banking-applications of various banks, though this mechanism is excluded from this thesis.

The driver is implemented with Embarcadero Delphi, for communication with LUI, Named Pipes are used. The driver is a process without GUI (graphical user interface), though a GUI can be used for testing purposes to monitor the process steps. For the OCR, several existent components were evaluated to extract an efficient one in relation to its price.

Besides this thesis as feasibility study, documentation of perceptions and documentation of implementation, the result includes the driver itself.

Furthermore, interviews with people out of the expected user group were done to determine usability and user acceptance.

Inhaltsverzeichnis

1. Einleitung

Bargeldloser Zahlungsverkehr im Allgemeinen und e-Banking (Tele- und Internetbanking) im Speziellen, sind für den größeren Teil der älteren Bevölkerung immer noch schwierige oder nicht durchführbare Tätigkeiten.

Für all jene, die diese Technologien zur Abwicklung ihrer Bankgeschäfte von Beginn an verwendet haben, sind sie Routine. Für Menschen, die es allerdings Jahrzehnte gewohnt waren, Bargeld und Belege in Papierform zu verwenden, ist die Umstellung schwierig und stellt die Benutzer und Benutzerinnen vor teilweise unlösbare Probleme. Dies führt von Skepsis über ängstliche Handhabung bis hin zur Verweigerung dieser Technologien zur bargeld- und papierlosen Abwicklung der Finanzangelegenheiten.

Zudem besitzen viele dieser Menschen nicht die erforderlichen technischen Mittel wie herkömmliche PCs oder Smartphones, Anbindung an das Internet bzw. sie nutzen die Geräte nicht, auch wenn sie diese besitzen.

Laut „Statistika GmbH“ nutzten bis Mai 2016 72% aller Österreicher und Österreicherinnen im Alter von 60 bis 69 Jahren das Internet, im Alter von 70 und älter sind es nur noch 48% [1]. Laut „Statistik Austria“, im Auftrag des österreichischen Bundeskanzleramts, besitzen 19% der österreichischen Haushalte keinen Computer. Bei 45% davon handelt es sich um Haushalte von 65-74 jährigen Menschen. Als Gründe nannten 77% der befragten Personen, dass kein Internetzugang gewollt ist, 29% gaben als Begründung „fehlende Kenntnisse“ an [2]. Laut „Statistik Austria“ nutzten 2015 59,5% aller Menschen zwischen 55 und 74 Jahren das Internet. 47% der gleichen Altersgruppe nutzte das Internet unterwegs, also via Mobiltelefon, Laptop, Netbook oder Tablet [3]. Laut „Statistika GmbH“ nutzten 2014 nur 48% der gesamten Bevölkerung das Internet für online-banking Dienste [4]!

Die genannten Statistikauszüge zeigen, dass immer noch ein großer Teil der Bevölkerung, speziell im Alter von 60 Jahren und älter, Technologien wie Computer, Internet, mobile Geräte und online-banking nicht nutzt.

Der Trend im Finanzwesen entwickelt sich allerdings in Richtung Rationalisierung des Schalterdienstes und zur Verwendung elektronischer Geräte bzw. zur Implementierung automatisierter Prozesse. Daher werden in den nächsten Jahren Lösungen von Bedarf sein, welche die Verwendung von Zahlscheinen und anderen Papierbelegen vorsehen, die Verarbeitung allerdings ohne bzw. mit möglichst wenigen bis hin zu gar keinen Benutzereingaben automatisiert vornehmen.

Am AAT (Zentrum für Angewandte Assistierende Technologien) des Instituts für Gestaltungs- und Wirkungsforschung der TU (Technische Universität) Wien wird ein „Senioren Terminal – SISI“ bzw. LUI entwickelt, welches auf Basis eines Touchscreen-PCs umgesetzt wird. Dieses Terminal bietet Anwendungen wie Telefon und Videophone, Kalender mit Erinnerungsfunktion, Tagesablaufplanung, Verbindung zu sozialen Netzwerken und andere Funktionen an. Es soll um eine Anwendung erweitert werden, welche es Benutzern und Benutzerinnen ermöglicht, auf einfache Art und Weise e-banking zu verwenden. Dies soll erreicht werden, indem die Benutzer und Benutzerinnen möglichst wenige Eingaben zu tätigen haben. Die Benutzer und Benutzerinnen können wie gewohnt Belege des Finanzwesens verwenden, welche in Papierform vorhanden sind.

Diese Diplomarbeit befasst sich mit der Erfassung der Belege und der möglichst fehlerfreien Extraktion der auf dem Beleg vorhandenen Informationen.

Die Belege werden von einem Bilderfassungsgerät (Kamera/Scanner) erfasst und die Bildinhalte werden mittels OCR (Optical Character Recognition) bzw. ICR (Intelligent Character Recognition) analysiert und ausgewertet. Die auf diese Weise erfassten Daten werden dem Benutzer oder der Benutzerin am Bildschirm zur Kontrolle angezeigt und können bearbeitet werden. Ab diesem Zeitpunkt können die Daten weiter verarbeitet werden, zum Beispiel über etwaige von Kreditinstituten zur Verfügung gestellte Schnittstellen, dies ist allerdings nicht Gegenstand dieser Diplomarbeit.

Besondere Augenmerke der Lösung liegen dabei auf:

1. Einfacher und komfortabler Bedienbarkeit
2. Geringer Anzahl an notwendigen Benutzereingaben, im Idealfall dem Ausbleiben von textuellen Benutzereingaben
3. Aufgrund der Sensibilität der Bankdaten:
    a. Einem hohen Grad an Datenkorrektheit
    b. Der Möglichkeit, die erfassten Daten zu kontrollieren und zu bestätigen

Daraus ergibt sich folgende Fragestellung:

Ist es möglich, die am AAT entwickelte Benutzerschnittstelle LUI, dem Stand der Technik entsprechend zu erweitern, um es benutzerfreundlich zu ermöglichen, in Papierform vorhandene Zahlscheine automatisiert zu erfassen und die enthaltenen Informationen korrekt und vollständig zu filtern, um diese zur weiteren Verarbeitung, zum Beispiel mittels e-banking, über eine direkte Schnittstelle zur Bank zur Verfügung zu stellen?

Die Herausforderungen liegen dabei unter anderem bei der unterschiedlichen Qualität und den unterschiedlichen Eigenschaften wie Bildhelligkeit, Kontrast, Sättigung und Weißabgleich der erfassten Bilder. Da die Erfassung mit Bildgeräten wie Webcams aufgrund der Umgebungslichtquellen wie Raumbeleuchtung, Tages- bzw. Sonnenlicht keine einheitliche Belichtung garantieren können, entfällt die Erkennung von Blindfarben, wie sie bei der Verarbeitung von Zahlscheinen grundsätzlich eingesetzt wird. Dies erschwert die Auffindung der entsprechenden Felder am Zahlschein, welche die gewollten Informationen enthalten.

Eine weitere Herausforderung liegt an den schlechten Ergebnissen von OCR Engines, wenn die Bildeingabe in entsprechend unzureichender Qualität erfolgt.

Zuletzt entfällt die Möglichkeit, die extrahierten Informationen durch Vergleiche mit vorhandenen Informationen zumindest zum Teil auf Korrektheit zu überprüfen. So können zum Beispiel Kreditinstitute den eingegebenen Namen und den IBAN mit Hilfe des Datensatzes aus dem Kundenstamm auf deren Zusammengehörigkeit überprüfen, um auf diese Weise die Fehlerhäufigkeit zu minimieren.

2. Ausgangssituation

Die Ausgangssituation besteht aus folgenden Teilen:

- Vorhandene Methoden zur Abwicklung von Bankgeschäften bzw. der Verarbeitung von Zahlscheinen allgemein
- Existente Lösungen zur elektronischen Abwicklung von Bankgeschäften bzw. der Verarbeitung von Zahlscheinen (e-banking-Lösungen)
- Vorhandene Hardware- und Software-Komponenten
- Die zu erwartende Benutzergruppe. Diese besteht aus den Benutzern und Benutzerinnen von LUI (des Senioren Terminals). Es handelt sich somit zum größten Teil um ältere Menschen und um Menschen mit körperlichen Einschränkungen.

Gegenwärtig existieren mehrere Möglichkeiten, um Bankgeschäfte im Allgemeinen bzw. Überweisungen im Speziellen zu tätigen. Folgend sind die gängigsten Varianten kurz beschrieben:

1. Die durch die Benutzer und Benutzerinnen eigenständige elektronische Abwicklung der Bankgeschäfte unter Einsatz von mobilen Applikationen, welche auf Tablets oder Smartphones installiert werden. Hierbei wird die App, wie im mobilen Bereich üblich, direkt über das Internet auf das Gerät geladen und installiert. Die Authentifizierung erfolgt analog zu den Web-basierten Anwendungen aus Punkt 2, also mit durch die Bank an ihre Kunden und Kundinnen ausgehändigten Zugangsdaten zur Anmeldung und mit TANs zur Bestätigung von Transaktionen. Die Durchführung von Überweisungen erfolgt grundsätzlich ebenfalls analog zu web-basierten Anwendungen, also durch Eingabe der Empfängerdaten sowie des Betrages und Bestätigung der Eingaben. Die dabei angebotenen Funktionen beschränken sich zumeist auf folgende:
    a. Übersicht über die vorhandenen Konten
    b. Übersicht über die Umsätze der vorhandenen Konten
    c. Tätigung von Überweisungen (In- und Ausland)
    d. Kontrollmappe (Übersicht über die getätigten Transaktionen)
    e. Anzeigen der Kontodetails
    f. Kontaktaufnahme zur Bank (zu Geschäfts- bzw. Hotline-Zeiten)
    g. Erteilung eines Auftrags zur Kartensperrung

    Einige dieser mobilen Applikationen bieten zusätzliche Funktionen für die erleichterte Durchführung von Überweisungen an. Dies zumeist durch den Einsatz von im Gerät verbauten Bilderfassungsgeräten. Details dazu werden im Kapitel *5.2. State of the Art und Related Work* beschrieben.

2. Die durch die Benutzer und Benutzerinnen eigenständige elektronische Abwicklung der Bankgeschäfte via Browser-basierten Webanwendungen, welche durch die jeweilige Bank gehostet werden. Dies ist die vermehrt eingesetzte e-banking-Technologie im Privatbereich. Die Kunden und Kundinnen können bei ihrer Bank Zugangsdaten für die Webapplikation der Bank anfordern und sich mit diesen bei der Webapplikation authentifizieren. Somit stehen den Kunden und Kundinnen alle Funktionen, welche die Webapplikation der Bank anbietet zur Verfügung. Die Durchführung von monetären Transaktionen wie Überweisungen erfordert eine weitere Authentifizierung durch die

Eingabe einer TAN (Transaktionsnummer). TANs sind Einmalpasswörter, die von der Bank an die Kunden und Kundinnen entweder bei Bedarf direkt an ihre bei der Bank registrierte Mobilfunknummer, oder über eine zusätzliche App, gesendet werden. Eine Alternative zur Verwendung von TANs bietet die elektronische Signatur über die Bürgerkarte. Möchte der Kunde bzw. die Kundin eine Überweisung tätigen, so muss er bzw. sie sich also auf der Webseite der Bank durch Eingabe von Zugangsdaten authentifizieren, folgend die Funktion Überweisung wählen, die Empfängerdaten eingeben, zumindest Empfängername, IBAN (International Bank Account Number) und Betrag, ggf. BIC (Business Identifier Code), Verwendungszweck oder Zahlungsreferenz und die Eingabe abschließend durch einen Klick auf den entsprechenden Knopf bestätigen. Folgend muss die Transaktion durch Eingabe der geforderten TAN oder durch die elektronische Signatur mittels Bürgerkarte bestätigt werden und erneut ein Klick auf den entsprechenden Knopf abgesetzt werden. Die dabei angebotenen Funktionen beschränken sich zumeist auf folgende:

   a. Auftragsmappe (Erstellung von Vorlagen, Daueraufträgen, Lastschriftaufträgen, Abschöpfungsaufträgen)
   b. Limit-Konfiguration (Einstellungen zu Betrag-Limits betreffend Kartenabhebung In-/Ausland, Hausbank), Überweisungen (In-/Ausland), Daueraufträgen, Kartenzahlung (tägliche & wöchentliche Limits)
   c. Anforderung von Kontoauszügen
   d. Umsatzreklamationen
   e. Wertpapier-Handling (Depots/Fonds, Einkauf/Verkauf)
   f. Produkteröffnung (Kontoeröffnung, Anforderung von Krediten – Vorberechnungen und Vereinbarung von persönlichen Terminen mit dem Bankberater bzw. der Bankberaterin)

3. Die konventionelle Abwicklung der alltäglichen Bankgeschäfte in persönlicher Kooperation mit einem/einer Angestellten der Bank. Dabei werden die zu erledigenden Tätigkeiten dem/der Bankangestellten persönlich mitgeteilt bzw. in vielen Fällen mittels Dokumenten in Papierform (Zahlungsanweisungen, Überweisungsscheine, Schecks und andere) übergeben. Dabei entstehende Informationen werden durch den Bankangestellten bzw. die Bankangestellte als elektronische Daten erfasst und verarbeitet.

4. Viele Banken stellen für Ihre Kunden bzw. Kundinnen neben den Selbstbedienungsterminals zur Geldabhebung und zum Kontoauszugsdruck auch Selbstbedienungsterminals für Überweisungen, sogenannte Überweisungsterminals, zur Verfügung. Diese Terminals verfügen meist über einen Touchscreen, eine Tastatur und eine Spezialtastatur zur Eingabe des PIN-Codes. Zum Teil sind diese Geräte auch mit Scannern zur elektronischen Erfassung von Zahlscheinen ausgestattet. Bei der Verwendung dieser Terminals in den Bankfilialen können die Kunden bei Bedarf die Unterstützung eines Bankangestellten in Anspruch nehmen.

5. Die durch die Benutzer und Benutzerinnen eigenständige elektronische Abwicklung der Bankgeschäfte unter Einsatz von lokal am PC installierten Applikationen, welche im Hintergrund via Internet die Transaktionen an den Server der Bank weitergeben.

Diese klassischen Client-Server-Architekturen finden hauptsächlich im gewerblichen Bereich Einsatz und erfordern die Installation und Konfiguration einer Anwendung auf den lokalen Clients oder Server des Kunden. Der Datenaustausch mit dem Server der Bank erfolgt dann nicht wie in Punkt 2 via HTTPS (Hyper Text Transfer Protocol Secure), sondern über eigene von den Banken entwickelte Anwendungsprotokolle, welche ebenfalls TCP (Transmission Control Protocol) nützen.
Hierzu müssen die Daten in allen Fällen manuell vom Benutzer bzw. der Benutzerin eingegeben werden. In großen Unternehmen kommen die Daten zwar oft bereits elektronisch zur Anwendung, z.B. von ERP-Systemen, müssen dann allerdings im Quellsystem manuell eingegeben und gepflegt werden.

Jegliche komplexe Bankgeschäfte wie Vereinbarung von Rahmenbedingungen, Erstellung von Portfolios, Produktanpassungen, Beratungen, Verhandlung etc. werden nach wie vor persönlich mit dem Bankberater oder der Bankberaterin durchgeführt.

Aus technischer Sicht besteht die Ausgangssituation aus folgenden Komponenten:

1. Windows-basierte Tablet-PCs. Diese haben eine integrierte Webcam sowie einen Touchscreen. Auf alle anderen Komponenten der Tablets wird nicht explizit eingegangen, da diese den Anforderungen an die Hardware hinsichtlich der Entwicklungen dieser Diplomarbeit in jedem Fall genügen. Auf diesen Tablets wird LUI installiert (siehe nächster Punkt).

2. LUI: LUI ist ein hochflexibles, modernes, robustes und multimodales Benutzerschnittstellen-System, entwickelt von der Gruppe fortec. Es wurde entwickelt, um die immer wiederkehrenden Anforderungen komplexer, technischer Systeme an das UI (User Interface) abzudecken. Abhängig vom Einsatzbereich und den Vorlieben und Fähigkeiten der Benutzer und Benutzerinnen, bietet das System eine hochgradige Konfigurierbarkeit. Multimodale Ein- und Ausgabemöglichkeiten sind bereits in der Grundausstattung integriert.
   Ein generischer Kern setzt Konfigurations- und Designanweisungen um. Der Kern kann über Treiber angepasst und erweitert werden und bietet verschiedenste Schnittstellen, sowohl für die Einbindung in ein System als auch zur Beeinflussung der Umgebung [5].
   Um die Umsetzbarkeit der Anforderungen, welche in dieser Diplomarbeit behandelt werden, überprüfen zu können, muss ein Treiber implementiert werden. Der Treiber läuft als eigenständige Komponente und kann prinzipiell von jedem Programm angesprochen werden, welches die Schnittstelle zur Kommunikation mit dem Treiber einhält. Die Kommunikation des Treibers zu anderen Komponenten wird nach den Vorgaben implementiert, welche es möglich machen, den Treiber in LUI einzubinden.

3. Logitech Pro Webcam C920: Für die Teststellung dieser Diplomarbeit wurde diese Webcam heran gezogen. (siehe *Abbildung 1 - Kameravorrichtung mit Logitech Pro Webcam C920*)
   Diese Webcam unterstützt
   a. *Full HD in 720p (bis zu 1280x720 Pixel)*
   b. *Full HD in 1080p (bis zu 1920x1080 Pixel)*
   c. *H.264-Videokompression*

   d. *ein Carl Zeiss®-Objektiv mit 20-stufigem Autofokus*
   e. *zwei integrierte Stereomikrofone*
   f. *automatische Belichtungsanpassung*
   g. *Hi-Speed USB 2.0 Schnittstelle*
   h. *Universalhalterung mit Stativgewinde* [6]

4. Am Markt vorhandene OCR-Komponenten. Um diese in eigenen Softwareentwicklungen nutzen und erweitern zu können, muss zwingend ein SDK (Software Development Kit) im Lieferumfang enthalten sein. Welche Komponenten es gibt, wie sich diese unterscheiden, welche für diese Diplomarbeit verwendet wird und wie es zu dieser Entscheidung kommt, wird im Kapitel 4.3. Evaluierung existenter OCR Engines beschrieben.

5. Am Markt vorhandene Programmiersprachen, Entwicklungsumgebungen und 3rd party Bibliotheken.

Abbildung 1 - Kameravorrichtung mit Logitech Pro Webcam C920 (Draufsicht)

Abbildung 2 - Kameravorrichtung mit Logitech Pro Webcam C920 (Seitenansicht)

Abbildung 3 - Kameravorrichtung mit Logitech Pro Webcam C920 (Schrägansicht)

3. Zielsetzung, Anforderungen und Abgrenzung

Folgend wird auf die Zielsetzung eingegangen, speziell auf den zu implementierenden Prozess, auf die technischen Anforderungen, welche zum Erzielen der Lösung eingehalten werden müssen, sowie auf die Abgrenzung der Inhalte dieser Arbeit.

3.1. Zielsetzung

Grundsätzliches Ziel ist, LUI um eine Lösung zu erweitern, welche es auch älteren Menschen ohne Kenntnisse im Umgang mit PCs, Smartphones und e-banking-Technologien erlaubt, e-banking zu nutzen. Dies soll erreicht werden, indem die Lösung keine textuellen Eingaben und nur wenige Klicks erfordert und somit leicht zu verwenden ist. Um dies zu erreichen, müssen die Daten, welche der Benutzer bzw. die Benutzerin in herkömmlichen e-banking Applikationen eingibt, elektronisch automatisiert erfasst werden.

Der Prozess zur Verwendung dieser Lösung soll möglichst ähnlich zum Prozess sein, welchen die Benutzer und Benutzerinnen kennen und gewohnt sind. Hierzu verwenden die Benutzer und Benutzerinnen nach wie vor ihre Belege in Papierform. Diese Dokumente sollen durch eine Kamera erfasst, analysiert und ausgewertet werden. Die alternative Verwendung eines Scanners wurde per Vorgabe ausgeschlossen, da Scanner nicht in der Standardausstattung der eingesetzten Terminals enthalten sind.

Diese Diplomarbeit umfasst die Implementierung eines Treibers, welcher Zahlscheine automatisiert erfasst und auswertet. Die erfassten Daten werden an LUI übergeben, wo sie dem Benutzer bzw. der Benutzerin angezeigt werden und manuell bearbeitet werden können. Durch die Implementierung weiterer Komponenten für LUI können die Daten dann weiter verarbeitet werden, zum Beispiel zur Weitergabe der Daten an eine etwaige Schnittstelle zu einer Bank bzw. einem Kreditinstitut. Dies wurde exemplarisch in der Arbeit mit dem Titel „Paper to Byte – Vom Zahlschein zum Online Banking – Ein Konzept für ältere Erwachsene“ von Robert Koch umgesetzt [7].

Die Benutzer und Benutzerinnen sollen ausschließlich die ihnen bekannte Oberfläche LUI bedienen müssen. Die gesamte Funktionalität soll automatisiert im Hintergrund abgewickelt werden. Im Idealfall muss der Benutzer bzw. die Benutzerin nur den Zahlschein vor der Kamera positionieren und den Start-Knopf der Anwendung am Touchscreen drücken. Nach erfolgreicher Verarbeitung des Zahlscheines werden die erfassten Informationen am Bildschirm angezeigt.

3.2. Anforderungen

Die geforderte Lösung soll als Treiber implementiert werden, verfügt also selbst über keine grafische Benutzeroberfläche. Über eine definierte Schnittstelle kann die Funktionalität des Treibers theoretisch von allen Programmen konsumiert werden, welche sich an die Schnittstellen-Vereinbarung halten. Da der Treiber primär entwickelt wird, um in LUI integriert zu werden, wird die Schnittstelle durch LUI indirekt definiert.

Dies betrifft sowohl die Technologie mit der zwischen dem Treiber und LUI kommuniziert wird, sowie das Schema der Nachrichten, welche zwischen Treiber und LUI kommuniziert werden.

LUI bietet grundsätzlich 3 Möglichkeiten zur Kommunikation mit anderen Komponenten:

1. DLL-Kommunikation
2. Kommunikation via „Named Pipes“
3. Representational State Transfer (REST)

Die Definition der Nachrichten, welche kommuniziert werden können, basiert auf der Extensible Markup Language (XML). Hierfür wurde bei der Entwicklung des LUI eine generische Struktur definiert welche alle möglichen Nachrichtenstrukturen definiert. Innerhalb dieser Strukturen können die Nachrichten innerhalb spezieller XML-Tags dynamisch sein. Der Aufbau aller möglichen Nachrichten ist dem Benutzerhandbuch von LUI zu entnehmen. Auf Grund der Vielzahl an Anforderungen an die Nachrichtenstruktur wird keine XML Schema Definition (XSD) eingesetzt [8].

Das Design der Oberfläche in LUI, welches für die Ansteuerung des Treibers erstellt wird, muss von der Implementierung des Treibers zu 100% entkoppelt sein. Dies hat den Vorteil, dass das Design von LUI geändert werden kann, um es an verschiedene Benutzer und Benutzerinnen und Anforderungen anpassen zu können, ohne dabei die Implementierung des Treibers anpassen zu müssen. Die hierfür benötigten Mechanismen sind bereits im Kern von LUI umgesetzt. Beim Erstellen der Oberfläche in LUI müssen lediglich die Vorgaben des Treibers hinsichtlich der Objektidentifikation eingehalten werden. Nach dem Start des Treibers durch LUI werden dem Treiber die LUI Komponenten mitgeteilt, an welche der Treiber die erfassten Informationen übergeben soll.

Die Kommunikation zwischen Treiber und LUI muss folgende Inhalte abdecken:

1. Nachrichten von LUI an den Treiber (siehe *Abbildung 27 - Flowchart: Message-Queue Reading-Thread*)
    a. Beim Start von LUI wird der Treiber gestartet indem die entsprechende Binärdatei gestartet wird.
    b. Unmittelbar nach dem Start des Treibers sendet LUI die IDs der Komponenten an den Treiber, an welche dieser die vom Zahlschein erfassten Informationen übergibt. LUI kann dem Treiber auch IDs von Komponenten übergeben, welche den aktuellen Prozessfortschritt sowie eine Beschreibung des aktuellen Prozessschrittes anzeigen können.
    c. Startbefehl zur Verarbeitung. Erhält der Treiber den Startbefehl, erfasst dieser ein Bild über eine am Terminal verfügbare Webcam und führt die entsprechenden Verarbeitungsschritte zur Analyse des Bildes durch.

   d. Abbruchbefehl. Erhält der Treiber den Abbruchbefehl, unterbricht er die Verarbeitung, verwirft die bis zu diesem Zeitpunkt erstellten Zwischenergebnisse und wartet auf erneute Eingaben seitens LUI.
   e. Beendigung des Treibers. Durch entsprechende Bedienung von LUI wird der Treiber beendet.

2. Nachrichten vom Treiber an LUI
   a. Übermittlung des aktuellen Prozess- bzw. Arbeitsfortschrittes.
   b. Übermittlung der erhobenen Daten nach erfolgreicher Verarbeitung des Eingabebildes. Da in dieser aktuellen Entwicklung und Überprüfung der Machbarkeit, ausschließlich Zahlscheine verarbeitet werden, handelt es sich hierbei um folgende Informationen:
      i. Name des Empfängers bzw. der Empfängerin
      ii. International Bank Account Number (IBAN) des Empfängers bzw. der Empfängerin
      iii. Business Identifier Code (BIC) des Empfängers bzw. der Empfängerin
      iv. Betrag
      v. Zahlungsreferenz
      vi. Verwendungszweck
      vii. Name des Auftraggebers bzw. der Auftraggeberin
      viii. IBAN des Auftraggebers bzw. der Auftraggeberin

Abbildung 4 - Template S€PA Zahlungsanweisung zeigt ein durch das Projekt S€PA (Single Euro Payments Area) definiertes Template einer Zahlungsanweisung. Der im Zuge dieser Arbeit entwickelte Treiber verarbeitet nur Zahlscheine, welche sich an dieses Template halten.

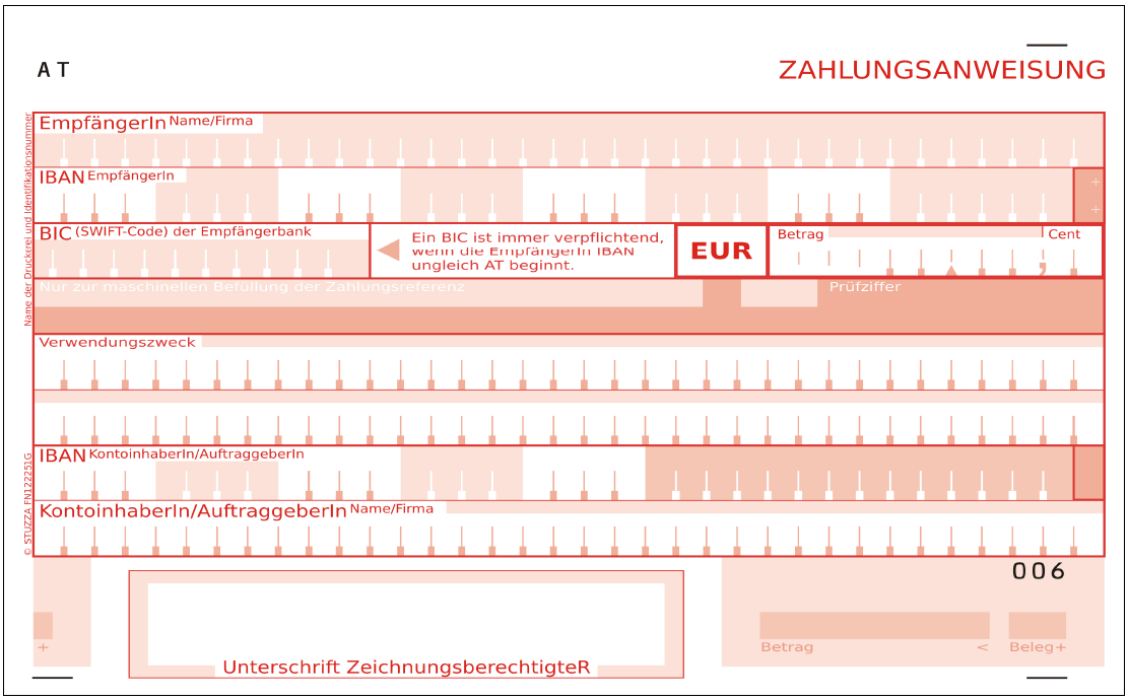

Abbildung 4 - Template S€PA Zahlungsanweisung [9]

Für die Entwicklung des Treibers wurden folgende Vorgaben seitens der Arbeitsgruppe fortec definiert:

- Entwicklung unter Einsatz der Programmiersprache Object Pascal. (Das hierfür bevorzugte Entwicklungswerkzeug ist Embarcadero Delphi, entwickelt von Borland und 2008 an Embarcadero Technologies Inc. mit Sitz in San Francisco verkauft.)
- Erfassung der Bilder mit einer handelsüblichen Webcam. Als Kamera für diese Entwicklung wurde die Webcam „Logitech Pro Webcam C920“ festgelegt.
- Einsatz geeigneter Methoden zur Verbesserung der Bildqualität.
- Entwicklung oder Einsatz von Algorithmen zur Layout-Erkennung der Zahlscheine.
- Erkennung und Ausgleich von Bildschrägläufen und Rotation.
- Entwicklung oder Einsatz von Algorithmen zur Filterung der Text-Bestandteile der Zahlscheine.
- Auswahl einer geeigneten OCR Komponente.
- Verarbeitung der Zahlscheine mit der gewählten OCR Komponente.
- Implementierung einer Kommunikationsschnittstelle zu LUI.
- Implementierung von Logging.

Um die Treiberintegration in LUI testen zu können, muss eine Oberfläche in LUI erstellt werden. Da die zu erwartenden Benutzer und Benutzerinnen von LUI vorwiegend ältere Menschen sind und diese oft unter bestimmten Beeinträchtigungen leiden, ist auf einen hohen Grad an Benutzerfreundlichkeit zu achten.

Folgende Liste zeigt mögliche Beeinträchtigungen die bei der Erstellung des LUI Designs in diesem Fall zu berücksichtigen sind. Diese sind visuelle Beeinträchtigungen [10]:

- Verlust von Kontrast- und Farbsensitivität
- Verminderung von Tiefenwahrnehmung und Sehschärfe
- Nachlassende Akkomodationsbreite
- Verzögerte Dunkelanpassung
- Abnahme des peripheren Sehens
- Verschlechterung der Wahrnehmung schneller Bewegungen
- Verschlechterung der Sehstärke

Auf Grund dieser möglichen Beeinträchtigungen sollte ein Kontrastverhältnis zwischen Text und Hintergrund von 50:1 angestrebt werden [11]. Weiters ist auf eine einheitliche Farbcodierung sämtlicher Icons zu achten [12]. Da mit zunehmendem Alter Blau- und Gelbtöne oft nicht ausreichend unterschieden werden können ist auf diese Farbkombination zu verzichten, um ausreichend Kontrast zu erzielen [13].

Hinsichtlich Schriften wird eine Mindestgröße von 12-14 Punkt, respektive 4,2-5 Millimeter empfohlen. Zudem sollten gut leserliche Schriftarten wie Times New Roman, Arial, Helvetica oder Verdana im Design eingesetzt werden [11] [14]. Auf Animationen sollte möglichst verzichtet werden, da das periphere Sehen der Benutzer und Benutzerinnen eingeschränkt sein könnte [15]. Um auf die unterschiedlichen Bedürfnisse der einzelnen Anwender und Anwenderinnen eingehen zu können, sollte das Design austausch- oder anpassbar sein [10].

Durch kognitive und motorische Einschränkungen sind Eingabegeräte wie Tastaturen, Mäuse und Stifte für die beeinträchtigten Menschen ungeeignet. Alternativ sollten Touchscreens eingesetzt werden, da hierbei der Kontakt zu den manipulierten Objekten intuitiver und natürlicher ist [16] [17]. Touchscreens vereinigen zudem Ein- und Ausgabe und sind auch für Anfänger ohne viel Training gut verwendbar [18]. Auf Grund ungewollter Berührungen von Objekten des Touchscreens durch den Nutzer oder die Nutzerin sollte es möglich sein die Position von GUI-Objekten im Design zu verändern [19].

Ein weiterer wichtiger Punkt des Designs ist die Beschriftung von Icons, zusätzlich zur bildlichen Darstellung, um dem Benutzer bzw. der Benutzerin deren Funktion eindeutig zu vermitteln [20].

Aufgrund der mangelnden Erfahrung der Benutzer und Benutzerinnen mit Informationstechnologien, ist generell auf eine einfache und intuitive Bedienung zu achten.

3.1. Abgrenzung

Zeitgleich zu dieser These bezüglich der Machbarkeit der beschriebenen Anforderungen hinsichtlich Integration einer Dokumentenkamera inklusive Belegleser in LUI, wird am AAT ein nahezu wartungsloser Drucker entwickelt. Dieser Drucker, inklusive des dafür benötigten Treibers, ist Gegenstand zweier anderer Entwicklungen, kann aber über die gemeinsame Integration in LUI letztendlich für das Drucken von Informationen betreffen Dokumentenkamera bzw. als Belegdrucker verwendet werden.

Die Arbeiten zur Entwicklung diese Druckers sind „Online-Banking-Belegdrucker für Senior/-innen“ von Goran Pavlovic [21], welcher einen möglichen Designentwurf erstellte und

„Technische Umsetzung eines Telebanking-Belegdurckers für Senioren/-innen“ von Clemens Eisserer [22], welcher die tatsächliche Entwicklung der Drucker-Hardware durchführte.

Weiters ist explizit zu erwähnen, dass diese Diplomarbeit ausschließlich über die Umsetzbarkeit der Anforderungen bezüglich Dokumentenkamera und Belegleser und deren Integration in LUI handelt, jedoch nicht über die direkte Anbindung an eine Bank oder ein Kreditinstitut. Die Daten werden lediglich einer etwaigen weiteren Entwicklung, welche die Anbindung an das Bankenwesen als Inhalt hat, zur Verfügung gestellt. Die Anbindung an eine Bank oder ein Kreditinstitut, inklusive der damit verbundenen Kriterien betreffend Datensicherheit und Einhaltung aller vorgegebenen Gesetze, Normen und Richtlinien, sind nicht Inhalt oder Teil dieser Arbeit. Zeitgleich zu dieser Diplomarbeit wird am AAT allerdings eine exemplarische Anbindung an zwei Kreditinstitute in der Arbeit „Paper to Byte – Vom Zahlschein zum Online Banking – Ein Konzept für ältere Erwachsene“ von Robert Koch implementiert [7].

Die in dieser Arbeit mit der Dokumentenkamera verarbeiteten Dokumente bzw. Zahlscheinbelege beschränken sich auf vom Projekt S€PA definierte Zahlungsanweisungen (siehe *Abbildung 4 - Template S€PA Zahlungsanweisung*).

4. Interviews mit Benutzern und Benutzerinnen von LUI

Eine weitere Anforderung laut Aufgabenstellung ist die Durchführung von Interviews mit den Benutzern und Benutzerinnen von LUI. Das Ziel hierbei ist die Abschätzung der Benutzerakzeptanz und die Erhebung von etwaigen Gründen für das Nicht-benutzen von vorhandenen e-banking-Lösungen, um diese Erkenntnisse wenn möglich direkt bei der Entwicklung berücksichtigen zu können.

Hierfür wurden 13 Personen befragt, darunter 5 Frauen im Alter von 75, 78, 80, 80 und 87 Jahren und 8 Männer im Alter von 76, 82, 82, 83, 83, 85, 86 und 88 Jahren.

Folgende Fragen wurden der Interviewgruppe dabei gestellt:

1. Nutzen Sie derzeit e-banking?
2. Wenn Frage 1 mit „Ja“ beantwortet wurde: Verwenden Sie hierfür einen Internet-Browser oder eine App auf Ihrem Smartphone?
3. Wenn Frage 1 mit „Nein“ beantwortet wurde: Würden Sie e-banking nutzen, wenn es für Sie einfach verständlich und anwendbar wäre?
4. Würden Sie e-banking mit dem Szenario dieser Diplomarbeit nutzen? (Hierfür wurde der Interviewgruppe die Lösung dieser Diplomarbeit gezeigt, siehe Kapitel 5.6. Integration in LUI)
5. Wenn Frage 4 mit „Nein“ beantwortet wurde: Welche Gründe sprechen für Sie gegen den Einsatz dieser Lösung?
6. Welche Möglichkeit zum Identitätsnachweis würden Sie bevorzugen?
7. Welche Informationen betreffend Überweisung sollten am Bildschirm sofort angezeigt werden?

Die Interviews wurden durch Katrin Gebhart im Zuge ihre Bachelorarbeit „Zurück zum Papier“ durchgeführt [23].

Abbildung 5 - Ergebnis der Frage 1: e-banking Nutzung zeigt, dass gegenwärtig nur 8% der Befragten ihre Bankgeschäfte unter Einsatz einer E-Banking-Technologie abwickeln, wobei 31% der Befragten nicht wussten, was e-banking tatsächlich ist.

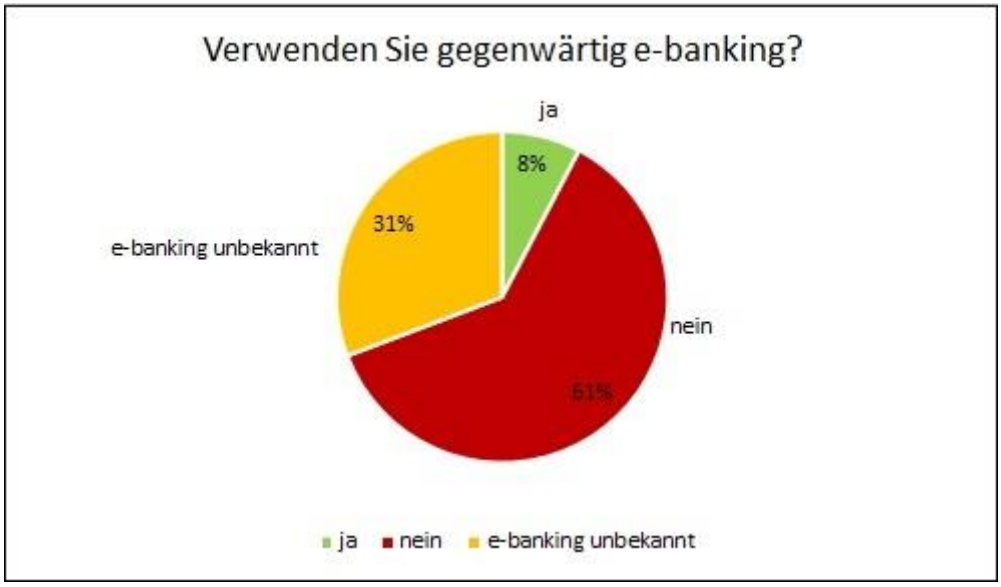

Abbildung 5 - Ergebnis der Frage 1: e-banking Nutzung

Diese Erkenntnis deckt sich weitgehend mit den Studien der Statistika GmbH und der Statistik Austria (siehe Kapitel 1. Einleitung).

In der zweiten Frage wurden alle gegenwärtigen E-Banking-Benutzer befragt, ob sie für die Durchführung eine eigene Applikation (Desktop oder mobile App) einsetzen oder direkt über die Web-Applikation ihrer Bank arbeiten. 100% der Befragten verwenden das Web-Interface ihrer Bank. Da allerdings nur eine Person generell e-banking anwendet, hat diese Statistik aus praktischer Sicht keinen Aussagewert.

In der dritten Frage wurden alle Personen, welche e-banking gegenwärtig nicht nutzen, gefragt, ob Sie es nutzen würden, wenn es für Sie verständlich und leicht anwendbar wäre. 12 der 13 befragten Personen antworteten mit „Nein“, 1 Person mit „Vielleicht, wenn ich einen Internetzugang hätte“. Die Gründe hierfür waren durchwegs die Skepsis gegenüber unbekannten Technologien und die Angst der fehlenden Unterstützung im Fehlerfall.

Bei der dritten Frage wurden die interviewten Personen befragt, ob sie sich vorstellen können, mit dem hier implementierten Szenario künftig e-banking zu verwenden. Dazu wurde den befragten Personen das Szenario inklusive der Benutzeroberfläche in LUI vorgestellt.

Auf die Frage, ob sich die Personen vorstellen könnten, unter Verwendung dieser Anwendung künftig ihre Bankgeschäfte abzuwickeln, antworteten 10 Personen (77%) mit „Nein“ und nur 3 Personen (23%) mit „Ja“ oder „Vielleicht“ (siehe *Abbildung 6 - Ergebnis der Frage 4: E-Banking mit vorgestelltem Szenario*). Dieses Ergebnis zeigt, wie wichtig es ist, dass Lösungen dieser Art für die Benutzer und Benutzerinnen einfach zu bedienen sind und diese das Gefühl haben, jederzeit zu wissen was passiert.

Abbildung 6 - Ergebnis der Frage 4: E-Banking mit vorgestelltem Szenario

Die Gründe warum sich die Mehrheit der Befragten gegen eine Verwendung aussprach, sind in *Abbildung 7 - Ergebnis der Frage 5: Gründe gegen die Verwendung dieser Applikation* dargestellt.

Abbildung 7 - Ergebnis der Frage 5: Gründe gegen die Verwendung dieser Applikation

Während die Anwendung für 30% zu technisch ist und 30% der Befragten sich mit keinen neuen Technologien auseinander setzten möchten, gaben jeweils 10% an, dass sie sich entweder zu alt dazu fühlen, es ihnen zu umständlich sei, die Kinder die Bankgeschäfte für sie erledigen oder sie es bevorzugen, im Problemfall lieber einen Bankangestellten zur Verfügung zu haben, um die Dinge zu klären.

Daraus ergeben sich für mich zwei Schlüsse. Erstens ist die starke Vereinfachung der Verwendung von e-banking trotzdem noch zu technisch und mit zu großem Neuheitscharakter für die Benutzer verbunden und die Scheu zur Verwendung somit immer noch aufrecht.

Im Kontrast dazu der zweite Rückschluss: Für Benutzer der LUI Komponente könnte dieses Szenario sehr schnell zur Gewohnheit werden, da das einheitliche Design von LUI den Benutzern bzw. Benutzerinnen eine einfach bedienbare und standardisierte Anwendung suggeriert, egal ob sie e-banking oder ein anderes, von LUI angebotenes Feature, verwenden.

Ähnliche Ergebnisse wie die Antworten aus der Frage in *Abbildung 7 - Ergebnis der Frage 5: Gründe gegen die Verwendung dieser Applikation*, lieferte eine Umfrage der Futurezone Technology News. Dieser Studie zufolge besitzen 90% der über 60-jährigen ein Mobiltelefon und knapp die Hälfte davon nutzen täglich das Internet [24].

Wenn auch die Altersgruppe in der Befragung dieser Diplomarbeit mehr als 10 Jahre über dem Alter der untersuchten Gruppe in der Marktstudie von Futurezone ist [24], zeigen sich doch ähnliche Muster, welche nachfolgend beschrieben sind.

Die Marktstudie beinhaltet zudem, dass Senioren und Seniorinnen auf dem Technik-Sektor keine Sonderbehandlung wollen, sprich weder dem Marketing noch Produkten, zugeschneidert auf Senioren, positiv gegenüber stehen. Laut der Studie räumen alle Teilnehmer den „Respekt Computern gegenüber“ und die anfängliche Angst „nicht alles richtig zu machen“, ein [24].

Betreffend Online-Banking deckt sich die Einstellung der Teilnehmer und Teilnehmerinnen der Marktstudie, mit der Einstellung der Befragten in dieser Diplomarbeit.

Hierzu äußerten sich die Befragten mit der Antwort, dass Online-Banking trotz langjähriger Computernutzung inklusive Internet tabu sei. Die Gründe hierfür sind die Scheu vor der Verwendung persönlicher Informationen wie Kontonummer etc. über das Internet. Für den Umgang mit dieser Art von Informationen fühlen sich die Betroffenen wiederum nicht vertraut genug mit der Technik [24].

Auch dies unterstreicht wieder die Forderung nach einer Lösung, die den Benutzern und Benutzerinnen die Skepsis hinsichtlich des Umgangs mit neuen Technologien wie auch die Angst, etwas falsch zu machen, nimmt. Und LUI löst genau diese Forderung, indem es eine einheitliche Bedienoberfläche für unzählige Anwendungen schafft.

Auf die Frage, welche Möglichkeit die Benutzer und Benutzerinnen für den Identitätsnachweis bevorzugen würden, wählten 100% die elektronische Unterschrift, vergleichbar mit den Paketdiensten und dem Zustellservice der Post, gegenüber den Alternativen von TAN-Codes per Post, der Bankomatkarte und TAN-Codes auf elektronischem Wege zu empfangen (siehe *Abbildung 8 - Ergebnis der Frage 6: Möglichkeiten zum Identitätsnachweis*).

Abbildung 8 - Ergebnis der Frage 6: Möglichkeiten zum Identitätsnachweis

Frage 6 bezieht sich auf etwaige zukünftige Implementierungen von Schnittstellen zu Banken bzw. Kreditinstituten. Tatsächlich obliegen die Möglichkeiten zum Identitätsnachweis natürlich nur der Bank selbst, da diese die Schnittstelle zu Ihrem System selbst definiert.

Bei der Frage, welche der von der Zahlungsanweisung extrahierten Informationen am ersten Ergebnisbildschirm angezeigt werden sollten, war sich die befragte Gruppe einig und sprach sich geschlossen für den Empfängernamen, den IBAN des Empfängers und den Betrag aus (siehe *Abbildung 9 - Ergebnis der Frage 7: Angezeigte Informationen*).

Abbildung 9 - Ergebnis der Frage 7: Angezeigte Informationen

Zusammenfassend zeigten die Interviews wie wichtig es ist eine Lösung zu entwickeln, die den Benutzern und Benutzerinnen vertraut erscheint und welche einfach bedienbar ist. Andernfalls ist die Benutzerakzeptanz zu gering, wie sich auch in einer Marktstudie der *future-zone* zeigte. Ersteres wird durch die Integration in LUI erreicht. Die einfache Bedienbarkeit muss bei der Erstellung des Designs und der Implementierung dieser Lösung berücksichtigt werden. Wenn möglich sollte die Lösung ohne textuelle Benutzereingaben auskommen. Des Weiteren ist es wichtig den Anwendern und Anwenderinnen durch eine leichte Bedienbarkeit zu zeigen, dass kein besonderes, technisches Know-How notwendig ist um diese Lösung zu verwenden und um zu wissen wie die Daten verarbeitet werden.

5. Realisierung

Die Überprüfung der Machbarkeit der geforderten Lösung beginnt mit einer Definition des methodischen Vorgehens sowie einer Beschreibung der eingesetzten Prozesse und Techniken.

Danach folgt ein Überblick über den Stand der Technik von:

- Vorhandenen, mit der Lösung dieser Diplomarbeit vergleichbaren Lösungen bzw. Lösungsansätzen (E-banking Applikationen, speziell jenen mit Scanfunktionen)
- OCR im Finanzwesen
- OCR- und ICR (Intelligent Character Recognition) Algorithmen und Funktionsweisen

Durch Evaluierungen von am Markt vorhandenen, OCR Engines und einer Gegenüberstellung dieser anhand definierter Attribute, wird der Auswahlprozess einer geeigneten OCR Lösung gezeigt.

Anhand einer prototypischen Implementierung der Anforderungen wird die Machbarkeit der Aufgabenstellung überprüft. Dabei wird die Implementierung dokumentiert, die erfolgreich umgesetzten Anforderungen aufgezeigt, die entstandenen Probleme illustriert sowie mögliche Lösungen diskutiert.

Ausserdem wird die Qualität der erzielten Ergebnisse anhand der prozentuell korrekt erkannten Inhalte aufgezeigt, insbesondere in Relation zu unterschiedlichen Lichtverhältnissen bei den Bildaufnahmen. Dies wird anhand von Tests durchgeführt.

Zuletzt wird eine mögliche Integration des Treibers in LUI aufgezeigt.

5.1. Methodik

Aus der Analyse der Ausgangssituation und der Definition der Zielsetzung ergeben sich folgende Hauptaugenmerke:

1. Da es sich bei der zu erwartenden Benutzergruppe hauptsächlich um ältere Menschen mit wenig oder keinem Bezug zu automatisierter Informationsverarbeitung handelt, ergibt sich die Forderung nach einer einfach bedienbaren Lösung. Aus diesem Grund wird die Architektur der Lösung so definiert, dass diese in Form eines Treibers implementiert wird. Der Benutzer bzw. die Benutzerin bedient lediglich die ihm bzw. ihr bekannte Oberfläche von LUI, welches im Hintergrund den Dienst des entwickelten Treibers in Anspruch nimmt. Auf diese Weise wird der Benutzer bzw. die Benutzerin mit keiner zusätzlichen Bedienoberfläche konfrontiert. Durch den flexiblen Aufbau von LUI kann die Bedienoberfläche auch an verschiedene Benutzeranforderungen angepasst werden. *Abbildung 10 - LUI Design mit Treiberschnittstellen* zeigt das Design von LUI mit den Schnittstellen zu Treibern. Die Anbindung dieser

Implementierung an LUI erfolgt über die Konfiguration eines LUI-Pipe-Treibers.

2. Um den Funktionsumfang der Texterkennung (OCR) zu erheben, muss der Stand der Technik von OCR und die damit verbundenen Technologien mittels Literaturrecherche erhoben und die existenten OCR Engines analysiert, getestet und evaluiert werden. Der Vergleich der OCR Engines wurde in mehreren Schritten durchgeführt:
    a. Vorauswahl, bedingt durch die Plattformabhängigkeit. Nicht für Windows entwickelte Engines werden ausgeschlossen.
    b. Vorauswahl, bedingt durch die Notwendigkeit eines SDK im Lieferumfang der Engine.
    c. Festlegung von KO-Kriterien bzw. Mindestfunktionalität.
    d. Vergleich der Funktionsumfänge, welche von den jeweiligen Herstellern beworben werden, mit den KO-Kriterien und Elimination ungenügender Engines.
    e. Durchführung von Funktionstests der Engines mit von den Herstellern angeforderten Testversionen. Hierfür werden die Engines mit den gleichen Eingaben getestet. Um die Ergebnisse vergleichen zu können, wird für alle Ergebnisse die Damerau-Levenshtein-Distanz des jeweiligen Ergebnisses und des Eingabetextes berechnet und auf einen Prozent-Wert skaliert.
    f. Gegenüberstellung der erreichten Ergebnisse, der Preise sowie der nutzbaren Funktionsumfänge aller Lösungen.

3. Um die Benutzerakzeptanz an technische Lösungen sowie der geplanten Lösung dieser Diplomarbeit zu ermitteln, wird ein GUI-Design für LUI entwickelt und zur Durchführung von Benutzerbefragungen verwendet.

4. Um zu überprüfen, ob die mit der Webcam erstellten Bilder den Anforderungen der OCR Engine an die Bildqualität genügt, werden damit erzeugte Bilder mit der gewählten OCR Engine analysiert. Hierfür werden Zahlscheine unter verschiedenen Bedingungen hinsichtlich Helligkeit, Kontrast, Belichtung und Farbsättigung abgelichtet und mit der OCR Engine verarbeitet. Auf diese Weise werden empirisch die Anforderungen an die Eingabequalität definiert und gleichzeitig die Einsatzfähigkeit der Kamera evaluiert.

5. Basierend auf folgenden Kriterien wird ein Software Design erstellt:
    a. Um parallele zwischen folgenden Tasts zu erreichen, wird Multi-Threading verwendet:
        i. Ansteuerung der Kamera und Verarbeitung der erfassten Bilddaten
        ii. Logging
        iii. Kommunikation mit LUI zum Empfangen von Befehlen von LUI, zum Senden von extrahierten Bankdaten an LUI und zum Senden des aktuellen Prozessfortschritts an LUI
        iv. Verarbeitung der Bilder mittels OCR Engine, wenn diese asynchrone Aufrufe unterstützt

   b. Ein späterer Tausch der OCR Engine ohne einen aufwändigen Umbau der Implementierung muss gewährleistet werden.
   c. Die Kommunikationsschnittstelle zwischen Treiber und LUI soll mit „Named Pipes“, unter Verwendung von DLLs oder unter Verwendung von REST erfolgen.
   d. Zu Testzwecken soll der Treiber ein GUI beinhalten, welches optional angezeigt werden kann.
   e. Der Treiber muss alle von LUI registrierten GUI Komponenten mit Daten versorgen.

6. Recherche für geeignete Bildbearbeitungsalgorithmen, speziell im Hinblick auf Merkmalserkennung in Bildern.

7. Implementierung des Treibers mit Embacadero Delphi.

8. Um die Funktionalität des Treibers und somit den Erfolg der Implementierung zu überprüfen, werden Tests durchgeführt. Diese Tests umfassen die Verwendung des Treibers mit unterschiedlichen Zahlscheinen und unter unterschiedlichen Belichtungssituationen. Anhand der Ergebnisse dieser Tests wird auch die Fragestellung dieser Arbeit beantwortet bzw. der prozentuelle Anteil der korrekt erkannten und extrahierten Zeichen abgeschätzt.

9. Die Umsetzung, sowie dabei aufgetretene Probleme und Erkenntnisse werden diskutiert, dokumentiert und mögliche Lösungsansätze werden mittels Recherche ermittelt.

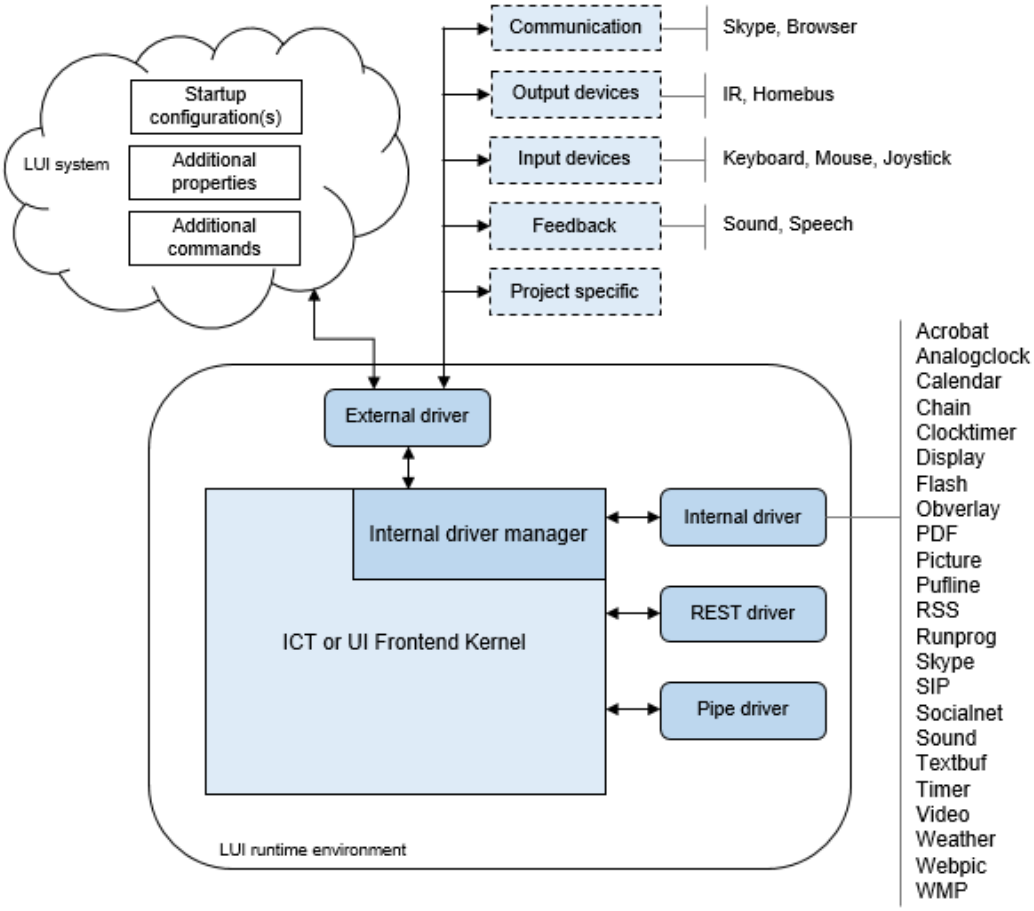

Abbildung 10 - LUI Design mit Treiberschnittstellen

5.2. State of the Art und Related Work

Bezüglich der Entwicklung und des Einbaus einer für ältere Menschen benutzerfreundlichen E-Banking-Applikation in LUI gibt es noch keine Vorversuche.

In den folgenden Unterkapiteln wird auf den gegenwärtigen Stand der Technik hinsichtlich OCR und ICR Algorithmen und Funktionsweisen eingegangen. Zudem werden ähnliche Lösungen, wie im Zuge dieser Diplomarbeit entwickelt, vorgestellt und getestet. Damit in Verbindung wird auch auf den Stand der Technik von Bilderfassungsgeräten, OCR im Finanzwesen, OCR-Schriften, sowie auf Prinzipien welche bei der Zahlscheinverarbeitung eingesetzt werden, nämlich das der Blindfarbe und das der Positionsmarkierungen, eingegangen.

5.2.1. OCR- und ICR-Verfahren

OCR, im deutschen Sprachraum auch optische Zeichenerkennung genannt, behandelt die Texterkennung in informationstechnisch verarbeiteten Bildern. Ein Teil dieser Arbeit ist die OCR Verarbeitung von Zahlscheinen mittels einer vorhandenen OCR Engine. Viele am Markt vorhandene Engines bietet unzählige Features, welche im Zuge des OCR Prozesses verschiedenste Arten von Bildbearbeitung durchführen. Dies mit dem Ziel die Ergebnisse der eigentlichen Texterkennung zu verbessern bzw. zu ermöglichen. Da eine Anforderung an die Entwicklung diese Treibers die leichte Austauschbarkeit der eingesetzten OCR Engine ist, dürfen neben der eigentlichen Texterkennung keine Features der eingesetzten OCR Engine verwendet werden. Hintergrund hierfür sind die ungleichen Funktionsumfänge der vorhandenen OCR Engines, neben der eigentlichen Zeichenerkennung.

Um alle möglichen beziehungsweise die für diese Arbeit notwendigen Bildbearbeitungsschritte zur Optimierung der OCR Ergebnisse zu kennen, wurde der Stand der Technik hinsichtlich OCR und ICR Verfahren analysiert. Ausserdem enthalten viele OCR Engines Funktionalität zur Nachbearbeitung der erkannten Zeichen, um die Qualität der Ergebnisse auf Grund von Wahrscheinlichkeiten und Textkontekt zu verbessern. Dieses Potential könnte eventuell auf die Ergebnisse dieser Arbeit angewandt werden oder zumindest im Hinblick auf die Weiterentwicklung dieser Arbeit aufbereitet werden.

Der Gesamtprozess der OCR Verarbeitung lässt sich in mehrere Schritte unterteilen. Die Literatur bietet keine einheitliche Einteilung in Prozessschritte. Eine mögliche Unterteilung des Gesamtprozesses ist folgender Auflistung zu entnehmen, wobei die einzelnen Prozessschritte jeweils Applikations-spezifische Teilschritte enthalten können:

1. **Bild-Vorverarbeitung**: Bei der Vorverarbeitung werden die relevanten Bildbereiche ermittelt, also jene, die den zu extrahierenden Text enthalten. Weiters werden Bildtransaktionen und Bildfilter angewandt. Eine mögliche Gliederung in Teilschritte, applikationsspezifisch nicht zwingend in der gleichen Reihenfolge und auch nur teilweise notwendig, ist:
    a. Qualitative Aufwertung der Bildinformation: In dieser Arbeit wird versucht den Kontrast zu erhöhen, die Bildhelligkeit und Bildsättigung anzupassen, sowie Störinformation aus dem Bild zu Filtern.
    b. Bildanalyse (Layout- und Textanalyse): Erkennung des Layouts und des Textes. Dazu gehören Zeilen, Blöcke, Absätze, Barcodes, QR Codes, Linien und andere geometrische Figuren sowie jegliche andere, mögliche Bildteile. In dieser Arbeit muss das Layout des Zahlscheines in Form der roten Rechtecke sowie der bedruckte Text erkannt werden.
    c. Erkennung der Grundlinie des Textes: Die Grundlinie des Textes ist die gedachte Linie, auf der alle Zeichen einer Zeile aufgesetzt sind. In dieser Arbeit nicht notwendig, da die horizontale Ausrichtung anhand des roten Layouts der Zahlscheine umgesetzt ist. (siehe nächster Punkt)
    d. „De-skewing“ bzw. Rotation des Bildes: Ziel ist es, das Bild vor der Zeichenerkennung so auszurichten, dass der zu erkennende Text, horizontal am Bild positioniert ist. Dies kann allgemein unter Berücksichtigung der Grundlinie des Textes umgesetzt werden. Applikations-spezifisch kann die

Rotation auch anhand bekannter Linien oder anderer Layout-Merkmale umgesetzt werden. In dieser Arbeit zum Beispiel anhand des rechteckigen Layouts der Zahlscheinfelder.

e. Entfernung aller Bildinformationen, welche nicht den Text repräsentieren. In dieser Arbeit muss das rote Layout der Zahlscheine entfernt werden. Dazu zählen Feldbeschriftungen der Zahlscheinfelder, der Rahmen der die Zahlscheinfelder begrenzt, der rote Hintergrund der Zahlscheinfelder sowie Störpixel, welche durch die Kameraaufnahme des Bildes entstanden sind.
f. Zeichensegmentierung: Abhängig von der OCR Engine müssen Zeichen getrennt werden, falls diese durch Störinformationen zusammenhängend erscheinen. In dieser Arbeit nicht notwendig, da die Zahlscheine mit OCR-B Schrift bedruckt sind und mit ausreichend hoher Auflösung abgelichtet werden, so entfällt der Fehlerfall von zusammenhängenden Zeichen.
g. Skalierung der Zeichengröße: Abhängig von der OCR Engine werden die besten Ergebnisse bei der Texterkennung, abhängig von der Zeichengröße in Pixel, erzielt. Vergrößerung von Zeichen kann allerdings bei fehlender Bildinformation zu Qualitätsverlusten führen.
h. Binarisierung des Bildes: OCR Engines verarbeiten grundsätzlich binäre Bilder, das heisst jedes Pixel kann entweder den Wert „0“, welcher die Farbe „Schwarz“ repräsentiert, oder den Wert „1“, welcher die Farbe „Weiss“ repräsentiert, annehmen. 8-Bit Grauwertbilder, bei denen jedem Pixel ein Wert von 0-255 zugeordnet ist, und 24 Bit Farbwertbilder, bei denen jedem Pixel jeweils drei Werte von 0-255, welche die Grundfarben Rot, Grün und Blau repräsentieren, zugeordnet sind, müssen vor der Texterkennung in Binärbilder umgewandelt werden.

2. **Text- bzw. Zeichenerkennung**: In diesem Schritt wird versucht, die Textzeichen auf dem Bild zu erkennen und zu extrahieren. Abhängig von der Implementierung der OCR Engine werden hierfür verschiedene Techniken, wie zum Beispiel Feature Extraction, Matrix Matching, Structural Analysis oder KNN (Künstliche Neuronale Netze) angewandt.
   Matrix Matching konvertiert jedes Zeichen in ein Muster, dargestellt durch eine Matrix. Dieses Rastergrafik-Muster des gefundenen Zeichens wird dann mit den Zeichen im *Stored Font Model*, dem Datenspeicher bekannter Zeichen, verglichen. Stimmt das gefundenen Muster mit einem gespeicherten überein, ist das Zeichen identifiziert. *Matrix Matching* eignet sich sehr gut für monotypische, einheitliche Seiten, also Eingaben mit gleicher Schrift und einheitlichem Layout. *Fuzzy Logic* erweitert *Matrix Matching* um Vergleiche von Bereichen beim Vergleichen der Raster. Wird kein eindeutiges Zeichen mittels *Matrix Matching* gefunden, wird mittels *Fuzzy Logic* das ähnlichste Zeichen aus dem *Stored Font Model* gewählt. *Feature Extraction* definiert die Zeichen über definierte Charakteristika wie Höhe, Breite, Dichte, Schleifen, Linien, Halme und andere Eigenschaften. *Feature Extraction* eignet sich sehr gut für Laserdrucke und Bilder von hoher Auflösung und Qualität. *Structural Analysis* identifiziert die Zeichen aufgrund von Zeichenteilen. *Structural Analysis* eignet sich für Bilder von schlechter Qualität. Neural Networks simulieren das menschliche

Neuralsystem. Die Pixel des Bildes werden mit einem Index bekannter Pixelmuster verglichen. Neuronale Netze eignen sich am Besten für spezifische Anforderungen wie das Erkennen von Trends in Grafiken [25].

3. **Fehlerkorrektur**: Die Fehlerkorrektur kann mit unterschiedlichen Verfahren durchgeführt werden.
    a. Eine Möglichkeit ist es, den extrahierten Text mit linguistischen Verfahren zu überprüfen. Je nach eingesetzter Sprache existieren andere linguistische Definitionen. Mit diesem Verfahren können Fehler zwar teilweise erkannt werden, jedoch schwer korrigiert. Zudem können Fehler auftreten, welche die linguistischen Regeln einer Sprache trotzdem erfüllen.
    b. Ein weiteres Verfahren ist der Abgleich der gefundenen Wörter mit einem definierten Wörterbuch. Mit diesem Verfahren können einzelne Zeichenfehler korrigiert werden. Wenn im Wörterbuch ein Wort existiert, dass dem gefundenen zum Beispiel bis auf ein Zeichen gleicht, so ist die Wahrscheinlichkeit unter Umständen hoch, dass es sich bei diesem Zeichen um einen Zeichenfehler der Texterkennung handelt. Je größer die Länge der verglichenen Wörter, desto wahrscheinlicher ist die Korrektheit der Korrektur.
    c. Manuelle Korrektur: Die Ergebnisse werden vom Benutzer bzw. der Benutzerin manuell überprüft und gegebenenfalls korrigiert, unter Umständen unter Zuhilfenahme des verarbeiteten Dokumentes.

4. **Nachbearbeitung**: Die Nachbearbeitung enthält Tätigkeiten zur weiteren Verarbeitung der extrahierten textuellen Information. Zum Beispiel die Codierung des Ausgabeformats, das Speichern der Informationen, das Anzeigen der Informationen, oder jegliche andere applikationsspezifische Verarbeitung der Daten. In dieser Arbeit werden die Daten als strings (Zeichenketten) vom Treiber an LUI gesendet, um dort angezeigt beziehungsweise weiter verarbeitet werden zu können.

OCR Engines beinhalten immer noch viele Schwächen, für welche es noch keine konkreten Lösungen gibt. Vor allem Dokumente von schlechter Bildqualität oder niedriger Auflösung erschweren die korrekte Zeichenerkennung. Viele Engines verwenden „Stored Font Models“. Das bedeutet, dass gespeicherte Muster zum Vergleich der gefundenen Muster herangezogen werden. Das birgt vor allem Probleme, wenn die Dokumente mit Bildstörungen behaftet sind oder eine andere Schrift verwendet wird als die Schriften, die im „Stored Font Model“ enthalten sind. Somit ist diese Methode auch für die Verwendung von Handschriften ungeeignet. Um diese Problematik zu umgehen, wurden Ansätze entwickelt, um Dokument-spezifische „Font Models“ zu erzeugen [26].

Anhand der Beschreibung der möglichen Prozessschritte lassen sich unter anderem folgende Einflussfaktoren auf die Qualität der OCR Verarbeitung ableiten:

- Qualität der Layouterkennung: Werden die Positionen der Bildteile, welche den Text enthalten, nicht erkannt, so kann auch der Text nicht extrahiert werden
- Schriftart des zu erkennenden Textes
- Wird *Matrix* oder *Pattern Matching* eingesetzt, so sind die Ergebnisse neben der Bildqualität auch maßgeblich von der Qualität der Muster-Datenbank abhängig

- Wird *Feature Extraction* eingesetzt, so sind die Ergebnisse neben der Bildqualität auch von der Qualität der definierten Merkmale abhängig. Indirekt hat dadurch auch die Eingabesprache einen Einfluss auf die Ergebnisse. Manche Sprachen beziehungsweise deren Zeichen und Symbole bieten besser unterscheidbare Merkmale als andere. Gleiches gilt auch für *Structural Analysis*.
- Qualität des Algorithmus zur Zeichenerkennung
- Qualität des Algorithmus zur Fehlererkennung beziehungsweise –korrektur
- Auflösung des Eingabebildes
- Bildeigenschaften des Eingabebildes: Kontrast, Helligkeit und Sättigung beziehungsweise Intensität

Ein maßgeblicher Schritt zur Verbesserung der Ergebnisse von OCR Verarbeitung besteht somit in der Verwendung von qualitativ hochwertigen Bilddaten in Form von hohem Kontrast, Vermeidung von Unter- und Überbelichtung und Vermeidung von Störinformation durch schlechte Aufnahmebedingungen wie falscher Belichtung, Bewegungsunschärfe, schlechtem Fokus und Auflösungen unter 2 Megapixel. Auflösungen von 2 Megapixel entsprechen bei der Ablichtung von Zahlscheinen ca. 300 dpi (Dots Per Inch), unter der Voraussetzung, dass der Zahlschein annährernd den gesamten Bildbereich abdeckt. In Anwendungsfällen wie dieser Diplomarbeit, in denen dies nicht möglich ist, sollte mit Algorithmen zur Bildbearbeitung versucht werden, die Qualität entsprechend aufzuwerten.

Die zweite Möglichkeit, OCR Verarbeitung zu verbessern, besteht in der Verbesserung des Texterkennungs-Algorithmus. Hierzu existieren viele Ansätze in allen Kategorien (feature extraction, structrual analysis, KNN, pattern matching), jedoch gibt es keine allgemein gültige Lösung. Die besten Ergebnisse werden erreicht, wenn die Wahl der OCR Technik vom Anwendungsfall abhängig gemacht wird. Unter Berücksichtigung des Anwendungsfalls können auch Optimierungen und Verbesserungen des Erkennungsalgorithmus umgesetzt werden. Folgend werden einige Ansätze aus den letzten Jahren diskutiert.

Sushruth, Gunasheela, Thejus, Vinay und Sudhir entwickelten in Indien eine OCR Methode namens „i“. Ziel von „i“ war die Schaffung einer einfachen und schnellen OCR Komponente, welche Schrift-unabhängig und Schriftgrößen-unabhängig ist. Das Besondere an „i“ ist, dass es keine Datenbanken und Bibliotheken mit Vergleichsmatrizen benötigt, sondern einen einzigartigen Algorithmus zur Erkennung von Zeichen verwendet. Implementiert mit MATLAB, hat „i“ bei einem Test mit 500 Bildern eine Genauigkeit von 100% bei den Schriften Arial, Times New Roman und Courier New erreicht. „i“ segmentiert einzelne Zeichen und analysiert dabei Eigenschaften wie Start- und Endpunktkoordinaten von einzelnen Buchstabensegmenten. Über gefundene Eigenschaften einzelner Zeichensegmente wird auf das jeweilig gefunde Zeichen rückgeschlossen [27]. „i“ erreichte zwar hohe Präzision bei den genannten Schriftarten, ist aber darauf angewiesen, dass die verschiedenen Zeichen einer Schriftart genügend distinktive Eigenschaften aufweisen. Weiters ist der Ansatz optimal für den Einsatzzweck von OCR ohne die Verwendung von Vergleichsdatenbanken, beschränkt sich aber auf diesen Einsatzzweck. Die Erkennung von Zeichen in Bildern mit schlechten Belichtungen, niedrigem Kontrast oder schlechten Auflösungen wurde nicht explizit untersucht.

In „Learning on the Fly: Font-Free Approaches to Difficult OCR Problems”, Andrew Kae und Erik Learned-Miller diskutieren ebenfalls einen Ansatz einer OCR-Methode, bei welcher ebenfalls weder Vergleichsdaten noch Trainingsdaten verwendet werden. Anstatt auf reine Zeichendefinitionen zu achten, werden Zeichenfolgen und Sprachstatistiken berücksichtigt.

Dieser Ansatz erfordert mehrere Iterationen. Im ersten Schritt werden Zeichen definiert, welche eindeutig zugeordnet werden können. In den nächsten Durchgängen wird der Kontext der bereits gefundenen Zeichen für die Evaluierung weiterer Zeichen verwendet [28]. Durch die Notwendigkeit mehrerer Durchgänge ist diese Methode bei großen Datenmengen wesentlich langsamer als andere Methoden. Des weiteren muss ein Kontext gefunden werden, um bei den weiteren Durchgängen den Grad der Zeicherkennung erhöhen zu können. Diese Erkennungmethode eignet sich daher nur für bedingte Anwendungsfälle, da keine unzusammenhängenden, beliebigen Zeichenfolgen erkannt werden können.

Für ICR werden in der Literatur zwei verschiedene Definitionen verwendet. In der einen Definition bezeichnet ICR die Fehlerkorrektur auf Zeichenebene. Hierbei wird das vermeintlich erkannte Zeichen verworfen und gegebenenfalls einer erneuten Erkennung unterzogen, welche einer anderen Konfiguration unterliegt. In der anderen Definition ist ICR eine Weiterentwicklung von HWR (Handwriting Recognition), bei der parallel zur Erkennung ein Lernprozess stattfindet, um die Genauigkeit für zukünftige Erkennungsprozesse zu erhöhen. Um Dokumente, welche aus Wörtern einer bestimmten Sprache bestehen, zu verarbeiten, kommt zusätzlich IWR (Intelligent Word Recognition) zum Einsatz. IWR arbeitet mit vollständigen Wörten im Gegensatz zu ICR und OCR, welche auf Zeichenebene operieren. Zeichenerkennung von handgeschriebenen Zeichen zählt nach wie vor zu den ungelösten Problemen des Sektors automatisierte Zeichenerkennung. Es existieren jedoch einige vielversprechende Ansätze, welche folgend kurz diskutiert werden. Diese setzen größtenteils KNN (Künstliche Neuronale Netze; englisch ANN – Artificial Neural Networks) ein. Bezogen auf das österreichische Finanzwesen bietet keine Bank eine Applikation, welche auch handgeschriebene Zahlscheine verarbeiten kann. (siehe Kapitel *Applikationen zur OCR-Verarbeitung von Zahlscheinen im Österreichischen Finanzwesen*). Nachfolgend wird auf vielversprechende Ansätze zur Handschriftenerkennung der letzten Jahre eingegangen.

Automatisierte, elektronische Erkennung von handgeschriebenen Zeichen ist Teil vieler Anwendungsszenarien. Unter anderem in Lesehilfen für erblindete Personen sowie bei der Analyse von Bankbelegen und anderen sturkturierten Dokumenten. Pradeep, Srinivasan und Himavathi ändern die Größe aller Zeichen auf 30x20 (600) Pixel und nutzen diese dann zum Training von neuronalen Netzen. Für die Klassifizierung des gelesenen Zeichens wird ein „feed forward back propagation neural network“ verwendet, welches die 600 Pixel des Zeichens analysiert. Von allen umgesetzten neuronalen Netzen hat jenes mit 2 Schichten zu jeweils 100 Neuronen mit 90% die besten Ergebnisse geliefert [29].

Patil und Shimpi verwenden in „Handwritten English character recognition using neural network“ ebenfalls neuronale Netze, den „Feed Forward Algorithmus“ und den „Back Propagation Algorithmus“ für Training, Fehlerkalkulation und Anpassung der Gewichtungen. Ihre Ergebnisse haben ebenfalls gezeigt, dass ein „Mulitlayer Perceptron Neuronales Netz“ mit einem versteckten „Layer“, trainiert mit dem „Error Back Propagation Algorithmus“, bessere Erkennungsgenauigkeit liefert und auch weniger Speicher auslastet. Die Ergebnisse lieferten eine Genauigkeit von 70%, gemessen an den Eingaben [30]. Auch wenn der Ansatz der neuronalen Netze und die Verwendung von „Error Back Progagation“ Vorgehensweisen vielversprechende Ergebnisse und Grundlagen liefern, so ist dabei allerdings zu berücksichtigen, dass die Eingabedaten bei den Versuchen hochwertig gewählt wurden. Daraus folgt ein unter Umständen deutlich schlechteres Ergebnis bei Verwendung von Bildern mit hohem Anteil an Störinformation. Des Weiteren sind Ergebnisraten von 70% für Einsatzzwecke mit hohen Anforderungen an die Korrektheit der Erkennung zu niedrig.

In „End-to-End Text Recognition with Convolutional Neural Networks" verwenden die Autoren neuronale Netze mit unbeaufsichtigtem „Feature Learning", anstatt ein Modell zu entwickeln, welches manuell, mit großen Mengen an definierten "Features" befüllt werden muss. Kombiniert werden die Ergebnisse mit einem Lexikon-Abgleich [31]. Somit eignet sich dieses Verfahren nur für Eingaben, die in Lexika stehen, nicht für beliebige Zeicheneingaben wie zum Beispiel dem IBAN von Zahlungsanweisungen. Durch das automatisierte, unbeaufsichtigte „Feature Learning" anstatt der manuellen Definition eignet sich dieses Verfahren auch nicht für Handschriften, kann allerdings die Performance und die Genauigkeit von OCR erhöhen.

Pradeep, Srinivasan und Himavathi entwickelten 2011 eine neue Methode zur Handschriften-Erkennung. Zwar verwenden die Autoren wie die meisten vergleichbaren Lösungen, „Multilayer Feed Forward Neural Network", allerdings verwenden die Autoren „Diagonal Based Feature Extraction" anstatt vertikaler oder horizontaler „Feature Extraction", um die Handschriften-Eigenschaften zu extrahieren. Für das Training des neuronalen Netzes werden in dieser Arbeit 50 verschiedene handschriftliche Alphabete von verschiedenen Personen in das neuronale Netz gepflegt. Bei der diagonalen „Feature Extraction" werden alle Pixel eines Zeichens (in dieser Lösung 90x60) in Zonen von 10x10 Pixel zerlegt. Dann werden die 19 Diagonalen jedes 10x10 Rasters extrahiert und der Durschnitt wird berechnet. Dieser Durchschnitt repräsentiert die jeweilige Zone. Bei 90x60 Pixel pro Zeichen und 10x10 Pixel pro Zone ergeben sich 54 Zonen und somit auch „Features". Zusätzlich werden 9 und 6 Features durch den Durchschnitt der vertikalen und horizontalen Zonenwerte berechnet. Somit ergeben sich pro Zeichen 69 Werte, zusammengesetzt aus den 54 Zonen, den 9 Werten der Spaltendurchschnitte und den 6 Werten der Zeilendurchschnitte. Erste Tests lieferten eine Genauigkeit von bis zu 98% [32].

2011 präsentierten 4 Mitarbeiter der IDSIA eine Lösung unter Verwendung von neuronalen Netzen, welche eine Fehlerrate von nur 0.27% erreichte. Allerdings bestanden die Texte ausschließlich aus Zeichen der „MNIST database of handwritten digits", mit welcher die neuronalen Netze trainiert wurden [33]. Um die große Menge an Trainingsdaten verarbeiten zu können, wurden hoch-performante Gaming-Grafikkarten verwendet – diese Lösung lässt sich somit nicht auf Standard-IT-Infrastrukturen einsetzen.

Auch wenn „künstliche Neuronale Netze" vermehrt bei der Verarbeitung von Hand-geschriebenen Zeichen Anwendung finden, können diese auch für OCR von Maschinen-Schriften eingesetzt werden.

Sameeksha Barve beschreibt die Verwendung von KNN für OCR in „Optical Character Recognition Using Artificial Neural Network". Die Eingaben in das OCR „Feature Extraction System" werden gemeinsam mit den Ausgaben in ein KNN gespeist. Der Back Propagation Algorithmus kombiniert die drei Bereiche Lerneffekt (Training), Fehlerkalkulation und flexible Gewichtung [34]. Wird ein schlechter Algorithmus zur „Feature Extraction" verwendet oder schlechte „Features" zur Zeichenerkennung, wird folglich auch die Klassifikation der gefundenen Zeichen im neuronalen Netzwerk schlecht. Barve ist es nicht gelungen, die beiden Eigenschaften Genauigkeit und Geschwindigkeit in ihrem Verhältnis zu optimieren. Folglich liefert diese Implementierung je nach Konfiguration entweder sehr genaue Ergebnisse zu Lasten der Geschwindigkeit oder umgekehrt.

Die Erhebung des Standes der Technik hinsichtlich OCR Verfahren zeigte die notwendigen Bildbearbeitungsschritte, bevor die Bilder mit der OCR Engine verarbeitet werden, um die best mögliche Qualität der Ergebnisse zu erzielen. Außerdem zeigte sich mit welchen OCR Verfahren die besten Ergebnisse bezogen auf die Eigenschaften Bildqualität, Layout und

Schriften der Eingabebilder erzielen lassen. Eine weitere Erkenntnis sind die existenten Schwächen von aktuellen OCR Verfahren. Daraus lässt sich ein eventueller, zukünftiger Tausch der OCR Verarbeitung in dieser Arbeit ableiten.

5.2.2. Applikationen zur OCR-Verarbeitung von Zahlscheinen im Österreichischen Finanzwesen

Der Trend im Finanzwesen entwickelt sich seit einigen Jahren in Richtung automatisierter Auftragsverarbeitung und Reduktion der Bankangestellten. Es existieren bereits Banken, die ihre Dienste ausschließlich via Internet anbieten. Diese Internetbanken gehören zu den sogenannten Direktbanken. Direktbanken sind Banken ohne eigenes Filialnetz und somit auch ohne direktem Kontakt zu den Kunden. Neben dem Vorteil, dass den Kunden die Dienstleistungen der Bank somit 24 Stunden am Tag zur Verfügung stehen, müssen die Kunden auch keine Niederlassung der Bank aufsuchen, sondern können ihre Geschäfte von jedem Ort mit Internet-Konnektivität, erledigen. Seitens der Bank werden dadurch Kosten eingespart, da keine Filialen und keine Schalterangestellten bezahlt werden müssen. Die Kostenersparnis kommt wiederum den Kunden in Form von besseren Konditionen zu Gute.

Im Zuge der Erhebung des Standes der Technik, wurden vorhandene Lösungen zur automatisierten Zahlscheinverarbeitung analysiert. Nach der Recherche über vorhandene Lösungen, wurde eine Applikation, die der Bawak P.S.K., getestet. Ziel war die Analyse der am Markt angebotenen Funktionalitäten, die Erhebung des Zusammenhangs zwischen Hardwareanforderungen an die Kamera und der Ergebnisse der Verarbeitung, die Analyse der Qualität der Ergebnisse, sowie die Beurteilung der Benutzerfreundlichkeit. Auf Grund der im Finanzwesen vorhandenen, branchenspezifischen Geheimhaltung konnten leider keine Informationen über die eingesetzten OCR Verfahren und Algorithmen recherchiert werden.

5.2.2.1. Überblick über vorhandene Lösungen

Unter anderem sind folgende Direktbanken in Österreich ansässig:

- Bankhaus Denzel (http://www.denzelbank.at)
- Bankdirekt.at (Tochter der Raiffeisenlandesbank Oberösterreich) (https://www.bankdirekt.at/eBusiness/01_template1/1132762101969358860-NA-NA-NA-2-NA-NA-NA.html)
- Direktanlage.at (Hello bank! By BNP PARIBAS) (https://www.hellobank.at/)
- Easybank (Tochter der Bawag P.S.K.) (https://www.easybank.at/easy)
- Generalibank.at (Teil der Generali Gruppe) (http://www.generalibank.at/)
- ING-DiBa (https://www.ing-diba.at/)

Im Gegensatz zu den Direktbanken betreiben die sogenannten Filialbanken Niederlassungen, an denen die Abwicklung der angebotenen Dienste durch persönlichen Kontakt zwischen Bankangestellten und Kunden abgewickelt werden.

In Österreich existieren unter anderem folgende Filialbanken:

- Raiffeisen (http://www.raiffeisen.at/oesterreich/NA-NA-NA-30-NA.html)
- Bank Austria (https://www.bankaustria.at/)
- Sparkasse (http://www.sparkasse.at/sgruppe/)
- Volksbank (https://www.volksbank.at/)
- Bawag P.S.K. (https://www.bawagpsk.com/BAWAGPSK/PK)

Um die Konkurrenzfähigkeit aufrecht zu erhalten, bieten jedoch alle 5 Banken aus dieser Aufzählung ihre Services ebenfalls via online-banking-Diensten an. Alle 5 sowohl via browserbasierten Applikationen als auch via Apps für Tablets und Smartphones.

Wenn auch mit unterschiedlichem Funktionsumfang, enthalten alle Apps dieser 5 Banken eine Funktion zum Scannen von Zahlscheinen. Zur Umsetzung dieser Funktionalität wird die im Gerät verbaute Kamera herangezogen, um den Zahlschein ablichten zu können. Die Apps der *Volksbank* [35] sowie der *Bank Austria* [36] bieten zumindest die Funktionalität zum Scannen von QR-Codes.

Laut einer Untersuchung der Futurezone Technology News Österreich wird der Einsatz von Zahlungsanweisungen in den nächsten Jahren zurückgehen und teilweise oder gesamt durch den Einsatz von QR-Codes (Quick Response Codes) auf Rechnungen und Erlagscheinen ersetzt werden [37].

Die notwendigen Standards, um QR-Codes überhaupt sicher und zuverlässig im sensiblen Finanzmarkt einzusetzen, wurden in Österreich bereits durch die STUZZA (Studiengesellschaft für Zusammenarbeit im Zahlungsverkehr) ausgearbeitet. In dieser sind die größten österreichischen Kreditinstitute sowie die Nationalbank vertreten. Darüber hinaus trägt die Plattform auch Sorge, dass die Vorgaben der europäischen Normungsinstitute berücksichtigt und umgesetzt werden. Nachdem schon mehrere Banken mobile E-Banking-Apps anbieten, mit denen QR-Codes eingescannt und verarbeitet werden können, geht die Erste Bank nun einen Schritt weiter. Mitte Dezember 2013 wurden sämtliche Selbstbedienungsterminals mit einer QR-Code-Lesefunktion aufgerüstet. Zahlscheine, auf denen ein QR-Code aufgedruckt ist, hinter dem sich eine Zahlungsanweisung verbirgt, können ohne Zutun über das Zahlscheinfach vom Gerät verarbeitet werden [37].

Die Vorteile dieser Lösung liegen an der einfachen Anwendbarkeit und der schnellen Übertragung ohne Übertragungsverlust. Neben der einfachen Anwendung für Zahlende, ermöglichen QR Codes eine schnelle Verarbeitbarkeit für Zahlungsdienstleister und eine sichere Zuordnung durch den Zahlungsempfänger [38].

Die Vorteile von QR-Codes gegenüber der bisherigen Vorgehensweise des Ablichtens der Dokumente und des anschließenden Verarbeitens mittels OCR sind eindeutig. Die Verwendung von QR-Codes inkludiert automatisch eine Überprüfung auf Korrektheit. Lesefehler können somit fast zu 100% ausgeschlossen werden. Außerdem ist die Verarbeitungsgeschwindigkeit von QR-Codes wesentlich höher [37].

Während sich diese Form der Zahlungsmethode in anderen Ländern wie zum Beispiel Japan schon vor Jahren durchgesetzt hat, bleibt der große Durchbruch in Österreich nach wie vor

aus. Zwar bieten sowohl die österreichischen Filialbanken wie auch Direktbanken die Verarbeitung von QR-Codes auf Zahlscheinen an, jedoch liegt es nach wie vor an den Unternehmen diese Methode auch umzusetzen. Hierfür müssten sämtliche Unternehmen und sonstige Institutionen eine QR-Code Infrastruktur bei sich implementieren und alle ausgegebenen Rechnungen und/oder Zahlscheine mit entsprechenden QR-Codes bedrucken.

Dies würde eine automatisierte Verarbeitung wie sie in dieser Arbeit umgesetzt wird theoretisch obsolet machen, jedoch selbst bei vollständiger Umsetzung der Methode mit QR-Codes wäre diese für ältere Menschen immer noch schwer anzuwenden, da der Umgang mit Smartphones und Tablets für viele dieser Menschen schwierig ist bzw. viele dieser Menschen nicht über diese Technologien verfügen. Auch die Anwendung selbst ist nicht ganz einfach, da man die Kamera aktivieren muss und manuell in gewissem Abstand und parallel zum QR-Code positionieren muss. Für Benutzer und Benutzerinnen welche diese oder ähnliche Techniken nicht alltäglich verwenden, erscheinen sie zum Teil sehr mühsam, sind schwierig oder schlichtweg nicht durchführbar.

Da es in Österreich nach wie vor eher unüblich ist, Zahlscheine mit QR-Codes zu bedrucken, wurde in dieser Arbeit von der Verwendung einer Infrastruktur für QR-Codes abgesehen. Der Einsatz von QR-Codes sollte in der Zukunft jedoch nicht vernachlässigt werden. Da diese für den Benutzer wesentlich leichter zu scannen sind als Zahlscheinteile und ganze Zahlscheine, würde sich diese Methode vor allem für ältere Menschen und Menschen mit taktilen Einschränkungen besser eignen.

Die Apps der Raiffeisen, der Sparkasse und der Bawag P.S.K. bieten neben der Funktion des Scannens von QR-Codes noch weitere Scanfunktionen an. Die App der Raiffeisen bietet zusätzlich einen Scanner für Zahlscheine wie auch einen IBAN-Scanner [39].

Die App der Sparkasse bietet die sogenannte „Scan & Pay“ Funktion, welche neben QR-Codes auch ganze Zahlscheine scannen kann. Die Hinweise der Sparkasse zum Nutzen dieser Funktion sind [40]:

1. Halten Sie das Smartphone beim Scan möglichst still
2. Achten Sie auf ausreichende Beleuchtung
3. Scannen Sie von oben und nicht schräg
4. Es können nur gedruckte und keine handschriftlichen Vermerke übernommen werden
5. Überprüfen und gegebenenfalls ergänzen Sie folgende vom Scan übernommenen Felder
    a. Empfänger
    b. Referenz
    c. Kundendaten
    d. Betrag
    e. Bankleitzahl bzw. BIC
    f. Kontonummer bzw. IBAN

Die App der Bawag P. S. K. bietet ebenfalls folgende Features [41]:

1. Feature: **„Zahlen mit QR (Quick Response) Code“**: Hierbei wird die im Gerät eingebaute Kamera genutzt, um den auf den Zahlschein gedruckten QR Code zu erfassen. Nach erfolgreicher Erfassung werden die durch den QR Code repräsentierten Daten in die Eingabefelder geladen und können vom Benutzer noch manuell bearbeitet werden.

2. Feature: **„IBAN-Scanfunktion“**: Hierbei wird die im Gerät eingebaute Kamera genutzt, um den, maschinell auf den Zahlschein gedruckten IBAN zu erfassen. Der erfasste IBAN wird dann in das vorgesehe Eingabefeld geladen und kann vom Benutzer noch manuell bearbeitet werden. Die restlichen Überweisungsdaten müssen vom Benutzer manuell eingegeben werden.

3. Feature: **„Scan & Transfer“**: Hierbei wird die im Gerät eingebaute Kamera genutzt, um alle maschinell auf den Zahlschein gedruckten Informationen zu erfassen. Nach erfolgreicher Erfassung werden die Daten in die Eingabefelder geladen und können vom Benutzer bzw. der Benutzerin noch manuell bearbeitet werden.

5.2.2.2. Testen einer vorhandenen Lösungen

Im Zuge dieser Arbeit wurden alle 3 Optionen der App der Bawag P.S.K. mit folgenden Geräten getestet. Ziel war es zu ermitteln ab welchen Kameraauflösungen die App verwertbare Ergebnisse liefert. Außerdem die Ermittlung der Qualität von vorhandenen Lösungen, sowie vor allem die Benutzerfreundlichkeit. Da es sich bei der Zielgruppe dieser Arbeit um ältere Menschen und Menschen mit Einschränkungen handelt, wurde erhoben ob vorhandene Lösungen unter Einsatz einer Smartphone-Kamera leicht zu bedienen sind, oder einiges an Benutzererfahrung erfordern.

5.2.2.2.1. Testgeräte

1. Testgerät: **Smartphone Sony Xperia Go** inklusive 5 Megapixel Kamera (siehe *Abbildung 11* - Sony Xperia go Smartphone mit 5 Megapixel Kamera)

Abbildung 11 - Sony Xperia go Smartphone mit 5 Megapixel Kamera [42]

2. Testgerät: **Smartphone Sony XPeria Z3 Compact** inklusive 20.7 Megapixel Kamera (siehe *Abbildung 12* - Sony Xperia Z3 compact Smartphone mit 20.7 Megapixel Kamera)

Abbildung 12 - Sony Xperia Z3 compact Smartphone mit 20.7 Megapixel Kamera [43]

3. Testgerät: **Tablet Samsung Tab 3** inklusive 3.2 Megapixel Kamera (siehe *Abbildung 13* - Samsung Tablet Tab 3 mit 3.2 Megapixel Kamera)

Abbildung 13 - Samsung Tablet Tab 3 mit 3.2 Megapixel Kamera [44]

5.2.2.2.2. Testszenarien

Durch die 3 zu testenden Features mit jeweils 3 Testgeräten ergeben sich folgende 9 Testszenarien:

1. Testszenario: Bawag P.S.K. „Zahlen mit QR Code“ mit Sony Xperia go Smartphone (5 Megapixel Kamera)
2. Testszenario: Bawag P.S.K. „Zahlen mit QR Code“ mit Sony Xperia Z3 compact Smartphone (20.7 Megapixel Kamera)
3. Testszenario: Bawag P.S.K. „Zahlen mit QR Code“ mit Samsung Tablet Tab 3 (3.2 Megapixel Kamera)
4. Testszenario: Bawag P.S.K. IBAN-Scanfunktion mit Sony Xperia go Smartphone (5 Megapixel Kamera)
5. Testszenario: Bawag P.S.K. IBAN-Scanfunktion mit Sony Xperia Z3 compact Smartphone (20.7 Megapixel Kamera)
6. Testszenario: Bawag P.S.K. IBAN-Scanfunktion mit Samsung Tablet Tab 3 (3.2 Megapixel Kamera)
7. Testszenario: Bawag P.S.K. „Scan and Transfer“ mit Sony Xperia go Smartphone (5 Megapixel Kamera)
8. Testszenario: Bawag P.S.K. „Scan and Transfer“ mit Sony Xperia Z3 compact Smartphone (20.7 Megapixel Kamera)

9. Testszenario: Bawag P.S.K. „Scan and Transfer“ mit Samsung Tablet Tab 3 (3.2 Megapixel Kamera)

Alle Tests wurden bei Tageslicht, sowohl mit wie auch ohne Sonneneinstrahlung und am Abend mit Raumlicht getestet. Ohne den Einsatz einer zusätzlichen Lichtquelle, in Form der in den Geräten verbauten Lampen, sind alle Tests fehlgeschlagen. Hauptgrund hierfür ist die Positionierung des Gerätes über dem abzulichtenden Zahlschein, welche zu einem Schatten führt, da sich das Gerät zwischen Lichtquelle und Zahlschein befindet. Folgend sind nur Testszenarien dokumentiert, bei denen eine zusätzliche Lichtquelle eingesetzt wurde, um ausreichende und gleichmäßige Beleuchtung zu schaffen.

Die Testszenarien 1 – 3 (Scannen von QR Codes) haben fehlerfrei funktioniert. Die Gründe hierfür sind, dass QR Codes entwickelt wurden, um gescannt zu werden und durch ihre automatische Fehlerkorrektur sehr robust sind und dass die zu scannende Information (der QR Code) den größten Teil des erfassten Bildbereichs einnimmt und dieser somit mit einer hohen Anzahl an Bildinformation erfasst werden kann, was die Weiterverarbeitung erleichtert.

Die Testszenarien 4 – 9 wurden mit folgenden Zahlscheinen durchgeführt:

- *Abbildung 14 - Bawag P.S.K. App Test-Zahlschein Nr. 1*
- *Abbildung 15 - Bawag P.S.K. App Test-Zahlschein Nr. 2*
- *Abbildung 16 - Bawag P.S.K. App Test-Zahlschein Nr. 3*

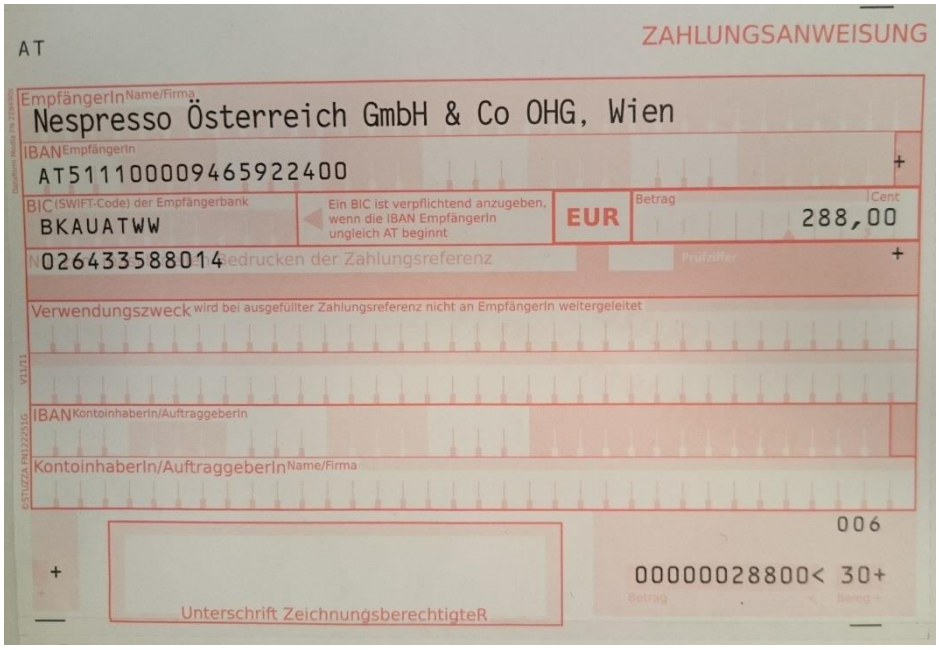

Abbildung 14 - Bawag P.S.K. App Test-Zahlschein Nr. 1

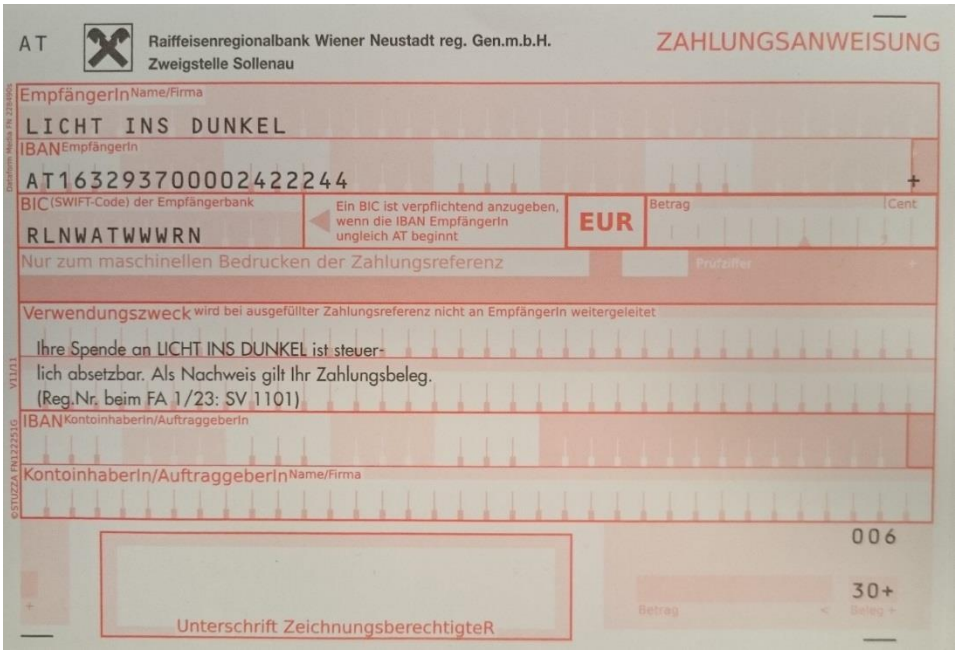

Abbildung 15 - Bawag P.S.K. App Test-Zahlschein Nr. 2

Abbildung 16 - Bawag P.S.K. App Test-Zahlschein Nr. 3

Für die Testszenarien 4 – 6 stellt die Bawag P.S.K. folgende Beschreibung zur Verfügung:

Abbildung 17 - IBAN-Scanner (links mit Smartphone, rechts mit Tablet) zeigt die Anleitung zur Benutzung des IBAN-Scanners der Bawag P.S.K. App.

Abbildung 17 - IBAN-Scanner (links mit Smartphone, rechts mit Tablet) [45]

Dazu werden folgende Tipps seitens der Bawag P.S.K. zur Verfügung gestellt [45]:

1. *Versuchen Sie durch langsame Auf- und Abbewegungen die richtige Position zu finden.*
2. *Das „AT“ des Anzeigefeldes zeigt Ihnen an, dass nur österreichische IBAN gescannt werden können.*
3. *Versuchen Sie daher mal das Smartphone weiter weg zu halten und die IBAN mittig im Anzeigefeld zu positionieren.*
4. *Achten Sie auf ausreichende und gute Beleuchtung (aktivieren Sie gegebenenfalls das Licht.)*
5. *Scannen Sie nicht schräg, sondern gerade von oben.*
6. *Es können nur gedruckte IBAN übernommen werden.*

Die Testszenarien 4 und 5 haben bei allen 3 Testzahlscheinen (*Abbildung 5 – 7*) fehlerfrei funktioniert. Analog zu den QR Codes ist auch in diesen Szenarien die zu scannende Information (der IBAN) im Verhältnis zum erfassten Bildbereich sehr groß und wird somit wieder mit einer hohen Anzahl an Bildinformation erfasst, was die Weiterverarbeitung erleichtert.

Testszenario 6 (IBAN-Scan mit Tablet Tab3) ist fehlgeschlagen. Die App zeigte den Fehlerdialog „Die Funktion ist leider nicht verfügbar!“. Da die App die Funktion allerdings inklusive Anwendungsbeschreibung anbietet, sowohl im GUI der App wie auch in der Beschreibung auf der Webseite der Bawag P.S.K., konnte die Ursache hierfür nicht festgestellt werden. Eine diesbezügliche Anfrage an den technischen Support der Bawag P.S.K. blieb unbeantwortet.

Für die Testfälle 7 – 9 stellt die Bawag P.S.K. folgende Beschreibung zur Verfügung:

Abbildung 18 - Scan & Transfer (links mit Smartphone, rechts mit Tablet) zeigt die Anleitung zur Benutzung des IBAN-Scanners der Bawag P.S.K. App.

Abbildung 18 - Scan & Transfer (links mit Smartphone, rechts mit Tablet) [45]

Dazu werden folgende Tipps seitens der Bawag P.S.K. zur Verfügung gestellt [45]:

1. *Positionieren Sie den Zahlschein innerhalb der Markierung, er sollte diesen Bereich möglichst ausfüllen.*
2. *Versuchen Sie durch langsame Auf- und Abbewegungen die passende Position zu finden.*
3. *Achten Sie auf ausreichende und gute Beleuchtung.*
4. *Scannen Sie nicht schräg, sondern gerade von oben.*
5. *Es können nur gedruckte, keine handgeschriebenen Einträge übernommen werden.*

Testszenario 7 (Scan & Transfer mit Sony Xperia go, 5 Megapixel Kamera) lieferte keine nutzbaren Ergebnisse. Entweder wurde der zu scannende Zahlschein nicht erkannt oder die App stürzte ab, die Gründe hierfür waren nicht reproduzierbar. Jeder der 3 Testzahlscheine (*Abbildung 5 – 7*) wurde 10 Mal versucht zu scannen. Die insgesamt 30 Versuche führten zu 11 nicht reproduzierbaren App-Abstürzen und zu 19 Versuchen, bei denen der Zahlschein nicht erkannt wurde. Die 19 Versuche wurden jeweils nach 2 Minuten Versuchsdauer abgebrochen. Da für die Versuchsreihe optimale Beleuchtungsverhältnisse geschaffen wurden, ist davon auszugehen, dass die Auflösung der Kamera zu niedrig ist.

Im Gegensatz zum Scan von QR Codes und zum IBAN-Scan, ist in den Testszenarien 7 – 9 die zu scannende Information im Verhältnis zum erfassten Bildbereich sehr klein, da in diesen Szenarien der gesamte Erlagschein erfasst werden muss.

Testszenario 8 (Scan & Transfer mit Sony Xperia Z3 compact, 20.7 Megapixel Kamera) lieferte teilweise nutzbare Ergebnisse. Jeder der 3 Testzahlscheine (*Abbildung 5 – 7*) wurde 5 Mal versucht zu scannen. Die Ergebnisse pro Zahlschein waren nicht zu 100% richtig, aber bei allen 5 Versuchen identisch.

Ergebnisse bei der Verarbeitung von Zahlschein Nr. 1 (*Abbildung 14 - Bawag P.S.K. App Test-Zahlschein Nr. 1*):

- Der Inhalt der Felder „IBAN“, „Zahlungsreferenz“ und „Betrag“ wurde richtig erkannt.
- Der Inhalt des Feldes „BIC“ wurde nicht erfasst.
- Der Inhalt des Feldes „Empfänger“ wurde bis auf zwei Zeichenfehler richtig erkannt.

Ergebnisse bei der Verarbeitung von Zahlschein Nr. 2 (*Abbildung 15 - Bawag P.S.K. App Test-Zahlschein Nr. 2*):

- Der Inhalt der Felder „Empfänger“ und „IBAN“ wurde richtig erkannt.
- Der Inhalt des Feldes „BIC“ wurde nicht erfasst.
- Der Inhalt des Feldes „Verwendungszweck“ wurde falsch erkannt; Grund hierfür ist, dass die für den Druck verwendete Schrift nicht dem OCR Standard entspricht (siehe *Abbildung 15 - Bawag P.S.K. App Test-Zahlschein Nr. 2*).

Ergebnisse bei der Verarbeitung von Zahlschein Nr. 3 (*Abbildung 16 - Bawag P.S.K. App Test-Zahlschein Nr. 3*):

- Der Inhalt des Feldes „IBAN“ wurde richtig erkannt.
- Der Inhalt des Feldes „BIC“ wurde nicht erfasst.
- Der Inhalt des Feldes „Empfänger“ wurde bis auf einen Zeichenfehler richtig erkannt.

Die Unterschiede zwischen den Testszenarien 7 und 8 ergeben sich durch die unterschiedlichen Kameraauflösungen. Da die zu scannende Information im Verhältnis zum erfassten Bildbereich sehr klein ist, muss die Auflösung des Bilderfassungsgerätes dementsprechend größer sein, um die erfassten Informationen mit einer entsprechend hohen Anzahl an Bildinformation zu erfassen.

Testszenario 9 (Scan & Transfer mit Samsung Tablet Tab3, 3.2 Megapixel Kamera) ist, wie auch Testszenario 6, fehlgeschlagen. Die App zeigte, analog zu Testszenario 6, den Fehlerdialog „Die Funktion ist leider nicht verfügbar!“. Da die App die Funktion allerdings inklusive Anwendungsbeschreibung anbietet, sowohl im GUI der App wie auch in der Beschreibung auf der Webseite der Bawag P.S.K., konnte die Ursache hierfür nicht festgestellt werden. Eine diesbezügliche Anfrage an den technischen Support der Bawag P.S.K. blieb auch hier unbeantwortet.

5.2.2.2.3. Zusammenfassung der Testergebnisse und Schlussfolgerungen

Tabelle 1 - Bawag P.S.K. Scan Features: Zusammenfassung der Ergebnisse zeigt die Zusammenfassung der Ergebnisse:

	QR Code Scan	IBAN Scan	Scan & Transfer
Sony Xperia Z3 compact 20.7 Megapixel Kamera	✓	✓	✓ (IBAN und Betrag) ✗ (Empfängername, Verwendungszweck)
Sony Xperia go 5 Megapixel Kamera	✓	✓	✗
Samsung Tablet Tab3 3.2 Megapixelkamera	✓	✗	✗

Tabelle 1 - Bawag P.S.K. Scan Features: Zusammenfassung der Ergebnisse

Schlussfolgerungen aus den 9 Testszenarien:

- Ohne Einsatz einer zusätzlichen Lichtquelle oder der im Gerät verbauten Lampe hat kein Verfahren funktioniert

Alle weiteren Schlussfolgerungen beziehen sich auf Tests mit zusätzlicher Lichtquelle:

- Scannen von QR Codes hat einwandfrei funktioniert. Sowohl mit Smartphones wie auch mit dem Tablet und auch bei niedrigen Kameraauflösungen von 2 Megapixel
- Scannen des IBAN hat mit den Smartphones ebenfalls funktioniert. Allerdings muss man Übung mit der Handhabung haben, da das Smartphone sehr ruhig und parallel zum Zahlschein gehalten werden muss. Diese Funktion eignet sich somit nur bedingt für ältere Anwender und Anwenderinnen und unter Umständen gar nicht für Anwender und Anwenderinnen mit Einschränkungen betreffend der Arme, der Hände oder des Sehvermögens
- Scannen des gesamten Zahlscheines hat nur mit einem Gerät funktioniert, welches über eine hohe Kameraauflösung verfügt. Die Testszenarien wurden mit 20.7 Megapixel durchgeführt. Diesbezüglich wäre eine genauere Studie notwendig, um die minimalerforderliche Auflösung zu bestimmen, da vom App Hersteller keine genauen Informationen hinsichtlich technischer Anforderungen bereitgestellt werden. Die Handhabung dieser Funktion ist zudem schwieriger als die IBAN-Scan-Funktion und erfordert einen sehr genauen Einsatz des Smartphones. Aus meiner Sicht eignet sich diese Funktion nicht für ältere Menschen. Dies führt zum Schluss, dass die in dieser Diplomarbeit entwickelte Lösung vorhandenen Bedarf seitens der Anwender und Anwenderinnen deckt. Neben dem Bedarf der genauen Positionierung des Smartphones über dem Zahlschein, muss diese Position auch mehrere Sekunden bis unter Umständen Minuten mit dem Smartphone gehalten werden. Zudem ist der Inhalt des Zahlscheines am Display schwer zu erkennen, was den Einsatz für Menschen mit Sehbehinderungen erschwert bis unmöglich macht. Außerdem funktioniert diese Funktion selbst bei bestmöglicher Anwendung durch den Benutzer bzw. die Benutzerin trotzdem nicht einwandfrei, da nicht alle Feldinhalte des Zahlscheines richtig extrahiert werden

Die Testergebnisse unterstreichen die Forderung nach Lösungen, welche auch von älteren Menschen und Menschen mit Einschränkungen leichter verwendet werden können. Selbst bei geübter Handhabung der Geräte und wenn diese sehr ruhig gehalten werden, hat die getestete Applikation nur teilweise funktioniert. Die Kamera musste annähernd exakt parallel zum Zahlschein gehalten werden, der Abstand zwischen Kamera und Zahlschein musste genau eingehalten werden und die am Display vorgezeichnete Schablone eines Zahlscheins musste exakt über dem abzulichtenden Zahlschein positioniert werden. Nur dann hat die Applikation den Zahlschein erkannt. Schon die Initialisierung der Kamera erforderte es, dass Gerät ca. 20 Sekunden ruhig zu halten. Daraus lässt sich die Wichtigkeit ableiten, in dieser Implementierung eine fixierte Kamera einzusetzen. Zudem zeigte sich, dass die Verarbeitung von ganzen Zahlscheinen bei Auflösungen unter 5 Megapixel nicht funktioniert.

Ein weiteres Einsatzgebiet von OCR im Finanzwesen sind die Selbstbedienungsterminals bzw. die Überweisungsterminals in Bankfilialen. Diese bieten die gleiche Funktionalität wie die Ergebnisse dieser Diplomarbeit, nämlich die automatisierte Erfassung und Verarbeitung von Zahlscheinen. Aufgrund der Geheimhaltung der Spezifikationen, Abläufe und Implementierungen durch die Banken und Dienstleister, welche diese Geräte verkaufen, implementieren und warten, ist das Wissen allerdings nicht wissenschaftlich zugänglich.

OCR im Finanzwesen, beziehungsweise OCR angewandt auf Zahlungsanweisungen, erfolgen im österreichischen Finanzwesen unter vordefinierten und kontrollierten Bedingungen. Dieser Anwendungsfall vereinfacht die automatisierte Verarbeitung von Zahlungsanweisungen in mehreren Hinsichten. Diese sind in der folgenden Aufzählung erläutert:

- Da die zu verarbeitenden Dokumente in eine dafür vorgesehene Maschine eingezogen werden und somit nicht beliebig vor dem Aufnahmegerät positioniert werden können, ist die Positionierung bei allen Anwendungsfällen ident. Durch die Normierung der Zahlungsanweisungen durch die Sepa und die immer exakt gleiche Positionierung der Zahlungsanweisungen bei der Ablichtung, entfällt die Linien- und Rechteckserkennung. Die Maschinen, mit denen in den Banken OCR auf Zahlungsanweisungen angewandt wird, können somit fast auf Pixel genau die Felder bestimmen, in denen sie nach den geforderten Informationen suchen müssen.
- Durch den Einzug der Dokumente in die Maschine ist zudem die Belichtung bei der Erstellung der Bilder immer die Gleiche. Dies ermöglicht eine auf den Anwendungsfall optimierte Konfiguration der Kamera.
- Einhergehend mit der Belichtung können auch Kontrast, Bildhelligkeit und Farbsättigung für den Anwendungsfall optimiert werden.
- Die einheitliche Belichtung beziehungsweise der Einsatz von speziellen Scannern zur Bilderfassung, ermöglichen ausserdem das leichte Ausblenden gewisser Layout-Teile, wie zum Beispiel den roten Rahmen von Zahlungsanweisungen, durch die Technik der Blindfarben.
  Als Blindfarbe wird das gewollte Ausblenden einer bestimmten Primärfarbe (rot, grün oder blau) beim Scannen bezeichnet. Mit Hilfe einer Blindfarbe können zum Beispiel vorgedruckte Linien und Rahmen auf Formularen oder Fragebögen beim Scannen ausgeblendet werden. Dies ermöglicht zum einen eine bessere Komprimierung der Bilder und gewährleistet zum anderen eine optimale Auswertung mit Hilfe von OCR oder ICR. Zur Verwendung einer Blindfarbe wird üblicherweise ein Graustufenscanner mit einer

speziellen Lampe ausgerüstet oder ein Farbscanner verwendet. Moderne Dokumentenscanner verfügen heutzutage alle über die Möglichkeit zur Auswahl einer Blindfarbe [46].
Als Raster- bzw. Textdruckfarbe sowie für die Bezeichnung ZAHLUNGSANWEISUNG ist ausschließlich eine rote Blindfarbe gemäß Spezifikation S+V J 25083[1] *(Aufrasterung der Feldunterlegungen sowie Feldbeschriftung) zu verwenden. Der Andruck der Währung „EUR" ist zwingend.*
Blindfarben sind spezielle Druckfarben, die keine Schwarzpartikel enthalten und durch eine entsprechende Abstimmung von Beleuchtung, Filter und optoelektronischen Empfängern (Scannern) ausgeblendet werden können. Insbesondere ist darauf zu achten, dass beim Bezug der Farben die Eigenschaft „Scannergeeignete OCR-Blinddruckfarbe" eingehalten wird. Der Rastervordruck muss beständig gegenüber nachfolgender Verarbeitung mittels Laserdruckern sein [9].
Auf Grund der Vorgaben für diese Arbeit konnte kein Scanner eingesetzt werden. Sollten die Senioren Terminals künftig mit Scannern ausgestattet werden, wäre es sinnvoll die Lösung dementsprechend zu adaptieren, um bessere Verarbeitungsergebnisse zu erzielen.

- Auch der Abstand vom Dokument zur Kamera ist immer der Gleiche, somit muss auch der Kamerawinkel nie verändert werden.
- Auch der Fokus der Kamera kann ebenfalls genau auf diesen Anwendungsfall konfiguriert werden und muss dann nicht mehr verändert werden.
- Da die Überweisungsterminals Zugriff auf die Geschäftsdaten der Bank haben, können unmittelbar nach dem Scanvorgang Überprüfungen der Daten auf Korrektheit durchgeführt werden, zum Beispiel ob die erfassten Informationen hinsichtlich „Empfängername" und „Empfänger IBAN" dem gleichen Konto zugeordnet sind.

Überweisungsterminals sind für die Verarbeitung von Zahlscheinen, welche in schwarz und in OCR-B-Schrift vorbedruckt sind, vorgesehen. Nur das Feld „Verwendungszweck" darf laut Spezifikation mit anderen Schriften bedruckt werden. Die getesteten Überweisungsterminals, die getesteten Apps und die Lösung dieser Arbeit funktionieren aber trotzdem auch mit anderen Schriften.

5.2.3. Zusammenfassung und Erkenntnisse des Standes der Technik für diese Arbeit

Zusammengefasst ergaben sich folgende Schlussfolgerungen aus der Erhebung des Standes der Technik von OCR und ICR Verfahrung und Algorithmen, sowie der Untersuchung vorhandener Lösungen zur automatisierten Verarbeitung von Zahlscheinen:

- Um bestmögliche Ergebnisse bei der Texterkennung zu erzielen, sind eine qualitative Aufwertung der Bilder, eine Layout-Analyse, der Schräglaufausgleich, das Ausblenden

[1] Gemäß OCR-Farbreihe von Sinclair & Valentine – z.B. Huber OCR Rot 521.

von nicht notwendigen Bildbestandteilen, die Skalierung der Zeichengröße, sowie die Binarisierung des Bildes notwendig.

- Abhängig vom Layout und der Qualität der zu verarbeitenden Bilder und von den eingesetzten Schriften, eignen sich verschiedene OCR Algorithmen. Während sich *Matrix Matching* für monotypische Eingaben eignet, erzielt *Feature Extraction* die besten Ergebnisse bei der Verarbeitung von Laserdrucken und generell Bildern mit hohen Auflösungen und hoher Qualität. *Structural Analysis* eignet sich zur Verarbeitung von Bildern mit schlechter Qualität. Spezifische Anforderungen wie zum Beispiel die Erkennung von Trends in Grafiken lassen sich am besten mit Neuronalen Netzen implementieren.
- Die erzielten Ergebnisse lassen sich mittels Fehlerkorrektur-Verfahren auf Zeichen- wie auch auf Wortebene verbessern.
- OCR Verfahren beinhalten immer noch viele Schwächen. Um zukünftig bessere Ergebnisse in dieser Arbeit zu erzielen, ist die Entwicklung bei der Textverarbeitung zu berücksichtigen und die eingesetzte OCR Engine gegebenenfalls zu tauschen.
- Vorhandene Lösungen zur automatisierten Zahlscheinverarbeitung unter Einsatz von mobilen Geräten sind schwer zu bedienen und ungeeignet für ältere Menschen und Menschen mit Einschränkungen.
- Der Einsatz von QR Codes würde die automatisierte Verarbeitung von Zahlscheinen aus Sicht der Bedienung massiv vereinfachen. Im Österreichischen Finanzwesen hat diese Technologie bis jetzt jedoch wenig Einzug gefunden.
- In dieser Arbeit muss eine fixierte Kamera eingesetzt werden.
- Vorhandene mobile Applikationen zur Zahlscheinverarbeitung benötigen Bilder von Kameraauflösungen größer als 5 Megapixel und Beleuchtung des Aufnahmebereiches.
- Die Verarbeitung der Zahlscheinfelder für Empfänger bzw. Empfängerin, IBAN, Betrag und Zahlungsreferenz liefert gute Ergebnisse, wenn eine entsprechend hohe Kamera-Auflösung zur Verfügung steht.
- Mit konstanter Belichtung des Aufnahmefeldes und darauf resultierendem konstantem Kontrast, konstanter Bildhelligkeit und Farbsättigung können die Ergebnisse der OCR Verarbeitung maßgeblich verbessert werden, da die Parameter zur Bildbearbeitung bestmöglich konfiguriert werden können.
- Der Einsatz von Scannern und dem damit verbundenen Konzept der Blindfarbe auf Erlagscheinen könnte die manuelle Layout-Erkennung ersetzen.

5.3. Evaluierung existenter OCR Engines

Um eine passende OCR Engine für diese Machbarkeitsstudie zu finden, wurde eine Internetrecherche hinsichtlich am Markt vorhandener OCR Engines durchgeführt.

Die Internetrecherche wurde mit folgenden Suchbegriffen durchgeführt:

- ocr engine
- ocr engine windows
- ocr sdk
- ocr delphi
- ocr open source
- ocr software
- ocr freeware

Folgende Liste enthält die recherchierten und untersuchten Engines in alphabetischer Reihenfolge:

- ABBYY FineReader Engine OCR SDK
  https://www.abbyy.com/de-de/finereader/
- Hyland Software (ehemals AnyDoc Software)
  https://www.onbase.com/de-DE/uber-hyland/akquisitionen/anydoc-software#.V6sJ2fmLSUk
- Asprise C/C++/Delphi OCR and Barcode Recognition
  https://asprise.com/royalty-free-library/ocr-api-for-java-csharp-vb.net.html
- CuneiForm
  http://cognitiveforms.com/products_and_services/cuneiform
- Dynasoft OCR SDK
  http://www.dynamsoft.com/Products/.net-ocr-component.aspx
- FreeOCR
  http://www.free-ocr.com/de.html
- GOCR
  http://jocr.sourceforge.net/
- Leadtools
  https://www.leadtools.com/
- MeOCR
  http://www.meocr.com/
- Microsoft Office Document Imaging
  https://support.microsoft.com/de-at/kb/982760
- Microsoft Office OneNote 2007
  https://support.office.com/de-de/article/Einf%C3%BChrung-in-Microsoft-Office-OneNote-2007-cd06e378-87e2-45b6-96af-7b384599f3d4
- Nicomsoft OCR SDK
  https://www.nicomsoft.com/
- Nuance OmniPage Capture SDK
  http://www.nuance.de/for-individuals/by-product/omnipage/index.htm
- Ocrad
  https://www.gnu.org/software/ocrad/
- OCRFeeder
  https://apps.ubuntu.com/cat/applications/precise/ocrfeeder/

- OCRopus
  https://github.com/tmbdev/ocropy
- OpenRTK – ExperVision OCR SDK
  http://www.expervision.com/ocr-sdk-toolkit/openrtk-ocr-toolkit-sdk
- Puma.NET
  http://pumanet.codeplex.com/
- Lexmark (ehemals Readsoft)
  http://www.lexmark.com/en_us/better-together.html
- Recogniform OCR Engine
  http://www.recogniform.com/ocr.htm
- Scantron
  http://www.scantron.com/
- Screenworm
  http://www.screenworm.de/
- SimpleOCR
  http://www.simpleocr.com/
- SmartScore
  http://www.musitek.com/
- Tesseract
  https://github.com/tesseract-ocr/tesseract

5.3.1. Vorauswahl der OCR Engines

Da es sich bei LUI um eine Windows Applikation handelt und somit auch der Treiber dieser Diplomarbeit unter Windows implementiert wird, wurden im ersten Evaluierungsschritt alle Engines ausgeschlossen, welche nicht für Windwos verfügbar sind. Diese sind:

- *OCRFeeder (Linux)* [47]
- *OCRopus (Linux)* [48]
- *Screenworm (Mac OS X)* [49]

Aufgrund der Forderung nach lokaler, programmatischer OCR Verarbeitung, wurden im zweiten Evaluierungsschritt alle Engines ausgeschlossen, welche kein SDK beinhalten. Diese sind:

- *AnyDoc Software* [50]
- *FreeOCR* [51]
- *GOCR* [52]
- *Microsoft Office Document Imaging* [53]
- *Microsoft Office OneNote 2007* [54]
- *Readsoft* [55]
- *Scantron* [56]

- *SimpleOCR* [57]
- *SmartScore* [58]

Die verbleibenden Engines wurden auf ihre Verwendbarkeit mit Delphi untersucht. Hierfür müssen die Engines eine mit Embarcadero Delphi verwendbare API anbieten. Dies kann in Form von C Bibliotheken, Zugriff via DLL Funktionen oder Zugriff via COM Objekten gegeben sein.

Tesseract OCR Engine wurde zwischen 1985 und 1995 von HP entwickelt und wird derzeit von Google weiter entwickelt. Tesseract unterstützt die Plattformen Linux, Windows, VC++, CygWin und Mac OSX und läuft unter der Apache 2.0 license. Die ursprüngliche Source wurde in C implementiert, danach wurde allerdings der größte Teil in C++ implementiert [59]. Um Tesseract-OCR in Delphi verwenden zu können, muss die Source verändert werden und anschließend erneut in C kompiliert werden. Versuche, bei denen dies unter Erhalt der gesamten Tesseract Funktionalität gelang, sind bis jetzt fehlgeschlagen [60].

CuneiForm, eine Softwarekomponente des russischen Unternehmens Cognitive Technologies wurde 2008 zur Freeware gemacht. Die Sourcen wurden unter dem Open Source Projekt OpenOCR veröffentlicht. Die Webseite zur Koordination des Projekts (http://OpenOCR.orf) ist allerdings bereits ausser Betrieb. Weiters bietet die Software keinen adäquaten SDK und die letzte Version wurde 2009 erstellt [61]. Diese Komponente schied somit aus.

MeOCR ist eine lizenzfreie Windows Applikation, welche die PUMA OCR Engine .NET API nützt. Die dabei von von MeOCR angebotene API ist somit die PUMA.NET API. Dabei erweitert MeOcr die PUMA.NET Lösung um einige Ausgabeformate wie HTML, RTF und DOCX (Worddokumente) [62]. Da diese Formate in dieser Implementierung nicht verwendet werden schied MeOCR aus dem Auswahlprozess und PUMA.NET wurde untersucht.

Puma.NET is a wrapper library for Cognitive Technologies CuneiFrom recognition engine that makes it easy to incorporate OCR functionality in any .NET Framework 2.0 (or higher) application [63]. Da PUMA.NET ein Wrapper der Cognitive Technologies CuneiForm recognition engine ist, diese allerdings aus genannten Gründen für diese Arbeit nicht einsetzbar ist, folgt, dass auch PUMA.NET für diese Arbeit keinen Nutzen bringt.

Dynamosoft bietet zwei Produkte inklusive SDK zur Bildverarbeitung. *Dynamic Web TWAIN* eignet sich zur Bildverarbeitung in Webapplikationen, *Dynamic .NET TWAIN* zur Bildverarbeitung in Anwendungen, implementiert mit dem Microsoft .NET Framework. Beide Anwendungen können, verbunden mit Zusatzkosten, mit einem OCR AddOn ausgestattet werden [64]. *Dynamic Web TWAIN* eignet sich nicht, da im Zuge dieser Diplomarbeit keine Webapplikation implementiert wird. *Dynamic .NET TWAIN* bietet ausschließlich ein SDK für Microsoft .NET und kann somit nicht direkt in Delphi Projekte eingebunden werden. Die Einbindung via COM ist umständlich und mühsam und die Laufzeit der Anwendung wird dabei negativ beeinflusst. Die Engine könnte somit für den Prototypen dieser Arbeit verwendet werden und wird in Evidenz gehalten. Sollte keine effizientere Lösung gefunden werden, kann auch dieses SDK zurück gegriffen werden.

Bezüglich Ocrad stellten sich die im Vorfeld gefundene Information, dass es sich hierbei um eine Engine handelt, welche auf der Windows Plattform nutzbar ist, als Fehler heraus. Ocrad – The GNU OCR ist Teil des GNU (GNU's Not Unix) Betriebssystems und ein Kommandozeilenprogramm, welches aber auch als Backend verwendet werden kann [65]. Die Windows Plattform wird somit nicht unterstützt und die Komponente scheidet aus.

Nach der Analyse der OCR Engines bezüglich Nutzbarkeit mit Embarcadero Delphi, verblieben folgende sieben Engines zur weiteren Evaluierung:

- *ABBYY FineReader Engine OCR SDK* [66]
- *Asprise C/C++/Delphi OCR and Barcode Recognition* [67]
- *Leadtools OCR SDK Technology* [68]
- *Nicomsoft OCR SDK* [69]
- *Nuance OmniPage Capture SDK* [70]
- *OpenRTK – ExperVision OCR SDK* [71]
- *Recogniform OCR Engine* [72]

Im Zuge der weiteren Evaluierung der verbliebenen Produkte wurden folgende Informationen erhoben:

- Funktionalität
- Lizenzmodelle
- Kosten der Anschaffung des notwendigen Lizenzmodells
- Anforderungen der Engine an die Bildqualität des zu verarbeitenden Bildes
- Möglichkeit, eine kostenfreie Trialversion zu testen

Es folgt eine Übersicht der, von den Produktherstellern ausgewiesenen, maßgebenden Produktleistungen, bezogen auf die definierten Anforderungen.

ABBYY FineReader Engine OCR SDK bietet Features zur Bildvorbearbeitung, inklusive Bildskalierung, Zuschneiden von Bildern, Bildrotation, Schräglaufbehebung, Spiegelung und Invertierung. Daneben können Algorithmen zur Bildsäuberung, Filterung und Binarisierung angewandt werden. Textblöcke können erkannt, aber auch manuell definiert werden. Linienerkennung ist laut Produktbeschreibung nicht inkludiert. Unterstützt wird unter anderem ICR, OMR (Optical Mark Recognition), OCR-A und OCR-B. Wörterbücher können verwendet werden – dabei werden zur Zeit 202 Sparchen unterstützt. Eine kostenlose Testversion ist verfügbar [66].
Eine Entwicklerlizenz kostet € 4900,00 und die Kosten für die kleinste Laufzeitlizenz betragen 420,00 pro 25000 verarbeiteten Seiten [73].

Asprise C/C++/Delphi OCR and Barcode Recognition unterstützt keine explizite Bildbearbeitung, da es für den Einsatzzweck von gescannten Dokumenten vorgesehen ist. Laut Produktbeschreibung enthält der Funktionsumfang auch keine Linien- und Rechteckserkennung und ICR wird ebenfalls nicht unterstüzt. Das Produkt unterstützt 20+ Sprachen. Das für diese Arbeit benötigte Lizenzmodell, und gleichzeitig das im Umfang geringste der drei angebotenen, kostet $ 4998,00 [67]. Da dieses Produkt für die Verarbeitung von gescannten Dokumenten vorgesehen ist und somit hohe Anforderungen an die Qualität der zu verarbeitenden Bilder setzt, eignet es sich nicht für die Verwendung für den Anwendungsfall dieser Arbeit.

LEADTOOLS OCR SDK Technology unterstützt OCR, ICR, MICR (Magnetic Ink Character Recognition), 40 Sprachen, die Verwendung von Wörterbüchern, Bildsegmentierung und Zonenerkennung, Noise-Erkennung und –Entfernung, Document Pre-processing, Barcode-Scanning und Fast TWAIN. Eine Demo-Version ist verfügbar. Der Preis für den LEADTOOLS Recognition Imaging SKD beträgt $ 3995,00 [68].

Nicomsoft OCR SDK ist ein Produkt des russischen Unternehmens Nicomsoft. Von allen evaluierten Lösungen ist dies die Einzige, die sich auf die Implementierung eines SDKs spezialisiert und keine Frontend-Software dazu anbietet. Die Komponente enthält folgende, ausgewiesene Features:

- *Bilderfassung über WIA (Windows Image Acquisition) und TWAIN* [69]
- *Bildschräglaufkorrektur* [69]
- *Bilddrehung* [69]
- *Bildinvertierung* [69]
- *Auto-Scaling* [69]
- *Bildbinarisierung zur Helligkeitsverbesserung und Kontrasterhöhung* [69]
- *Layout-Analyse und die Möglichkeit, Bildteile zu definieren* [69]
- *Linien- und Rechteckerkennung* [69]
- *ICR* [69]
- *Definierbarkeit eines Eingabealphabets* [69]
- *Verwendung von benutzerdefinierten Wörterbüchern* [69]
- *Verarbeitung von Bildern aus und in den Speicher* [69]

Eine kostenlose Testversion ist verfügbar und wird mit zahlreichen Beispiel-Anwendungen ausgeliefert, um die Funktionalität testen zu können und gleichzeitig die Verwendung des SDKs anhand des Source-Codes nachvollziehen zu können. Die Kosten für eine Einzelentwickler-Lizenz betragen $399,00 [69].

Nuance OmniPage Capture SDK der Firma Nuance bietet sowohl eine C/C++ API wie auch eine ActiveX Schnittstelle, welche den Funktionsumfang der C-Schnittstelle, sowie zusätzliche Features für effizientes Dokumentenmanagement, anbietet. Die Bildvorbearbeitung inkludiert Bilddrehung, Bildbegradigung, Bildspiegelung, Bildreinigung sowie eine Verbesserung der Bildauflösung. Capture SDK enthält 12 Erkennungmodule, unter anderem für Maschinenschrift (OCR), Handschrift (ICR) und OCR-B. Zoneninhaltsprüfung sowie Textprüfung sind ebenfalls enthalten. Linien- und Rechteckerkennung werden nicht explizit beworben. Eine kostenlose Testversion ist verfügbar. Das Produkt ist ab € 3995,00 erhältlich [70]. Anforderung an die benötigte Bildqualität konnten nicht in Erfahrung gebracht werden.

OpenRTK – ExperVision OCR SDK bietet die Features Bildbinarisierung, Bildrotierung, Schräglauferkennung, Linienerkennung, Bildgeräuschfilterung und auch die Möglichkeit zur Definition von Bildteilbereichen. ICR und die Definition eines Eingabealphabets wird nicht unterstützt. Die Engine setzt ein Open API Design um [71]. Preise zum Lizenzmodell sind nur auf Anfrage erhältlich. Anforderungen der Engine an die Bildqualität sind keine ausgewiesen. Die Möglichkeit zu einer Trialversion besteht. Aufgrund der ausbleibenden Möglichkeit, ICR zu testen, wird diese Engine aus dem Evaluierungsverfahren ausgeschlossen.

Recogniform OCR Engine verkauft neben ihren Produkten auch die Funktionalität dieser, aufgeteilt auf 24 Module in Form von DLLs (Dynamic Linkes Libraries) und COM (Component Object Model) Objekten [72]. Mit folgender Zusammenstellung könnte die gesamte benötigte Funktionalität abgedeckt werden:

- *OCR Omnifont Recognition Engine, Kosten € 400,00* [72]
- *OCR-B Recognition Engine, Kosten € 2000,00* [72]
- *ICR Handwritten Recognition Engine, Kosten € 600,00* [72]
- *Lines Removal Library, Kosten € 600,00* [72]

- *Dynamic Thresholding Binarization Library, Kosten € 500,00* [72]
- *Handwritten/Printed Fields Classification Library, Kosten: keine produktive Version verfügbar* [72]
- *Deskew Library, Kosten € 599,00 Royalties Free* [72]

Die Kosten für die zumindest notwendigen Module „OCR Omnifont Recognition Engine“, „OCR-B Recognition Engine“ und „ICR Handwritten Recognition Engine“ ergeben eine Summe von € 3000,00.

Die genauere Evaluierung der Engines ergab, dass OpenRTK – ExperVision OCR SDK nicht optimal zum Anwendungsfall passt, und dass die Engine Asprise C/C++/Delphi OCR and Barcode Recognition keine Verarbeitung von nicht-gescannten Dokumenten vorsieht, im Anwendungsfall dieser Diplomarbeit jedoch ausschließlich nicht-gescannte Dokumente verarbeitet werden.

Folgende Tabelle gibt eine Übersicht über die verbleibenden fünf Engines.

Hersteller	**Trial**	**Lizenz**	**Kosten**
Abbyy	Ja	Entwickler + Laufzeit	4.900,00 EUR + 420,00 EUR/25000 pp.
Nuance	Ja	Entwickler	3.995,00 EUR
Nicomsoft	Ja	Entwickler	399,00 USD (~ 307,00 EUR*)
Recogniform	Ja	-	3.000,00 EUR + 1.699,00 EUR
Leadtools	Ja	Entwickler	3995,00 USD (~ 3.073,00 EUR*)

* es wurde ein Kurs von 1 EUR = 1.3 USD verwendet

Tabelle 2 - Preisgegenüberstellung evaluierter OCR Engines

In *Tabelle 2 - Preisgegenüberstellung evaluierter OCR Engines* ist deutlich zu sehen, dass das Produkt der Firma Nicomsoft bei weitem die billigste Lösung anbietet. Eine Begründung hierfür konnte nicht recherchiert werden. Sollten die Engines eine annähernd gleiche Qualität erbringen, liegt dies wohl am Verkaufsmodell. Nicomsoft ist das einzige Produkt, bei dem nur der SDK existiert und keine Frontend Anwendungen zum Verkauf angeboten werden. Recogniform hat bei der Produktimplementierung die Funktionalitäten auf 24 unabhängige Module aufgeteilt. Der Kunde kann neben den Hauptprodukten eine beliebige Zusammenstellung an Bibliotheken erwerben, welche die 24 Module implementieren. OmniPage und Nuance bieten zahlreiche Frontend Produkte an, welche ihre Engines verwenden. Zusätzlich bieten sie allerdings ihre SDKs auch explizit zum Verkauf an, ohne eine Frontend Applikation zu erwerben.

Recogniform OCR Engine wurde aus dem weiteren Evaluierungsprozess ausgeschlossen, da es keine preislichen Vorteile gegenüber den anderen Lösungen bringt, die modulare Aufteilung allerdings zu einer weniger performanten und komplexeren Implementierung führen würde.

5.3.2. Testen der verbleibenden Engines

Von den restlichen vier Engines wurden Testversionen von den Herstellerseiten geladen, um diese in Livetests untersuchen zu können. Hierfür wurden die gleichen Bilder mit allen vier Engines verarbeitet und die Ergebnisse mittels „Damerau Levenshtein Distance“ zum Eingabetext vergleichbar gemacht.

Levenshtein Distance – A distance between two strings, not necessarily of the same length, given by the minimum number of symbol insertions, deletions and substitutions required to transform one string into the other [74].

Um einen leichter zu vergleichbaren Wert zu erhalten, wird die *Levenshtein Distance* auf Prozent skaliert, indem die *Distance* durch die Länge der Zeichenkette dividiert wird, das Ergebnis von 1 subtrahiert und dann mit 100 multipliziert wird. Da der Nachkommateil vernachlässigt werden kann, wird dieser gerundet. Das Ergebnis beinhaltet die prozentuelle Übereinstimmung der beiden Zeichenketten, für die die Distanz berechnet wurde.

Vor der Durchführung der Tests wurden 4 verschiedene Gruppen bzgl. Eingabebilder definiert. Diese Gruppen sollten als Richtlinie für das Testen dienen:

- Zahlschein mit Maschinendruck, von hoher Qualität, erreicht durch scannen der Belege mit 300 dpi
- Zahlschein, handschriftlich ausgefüllt, von hoher Qualität, erreicht durch scannen der Belege mit 300 dpi
- Zahlschein mit Maschinendruck, erfasst mit der Webcam, welche für diese Teststellung verwendet wird
- Zahlschein, handschriftlich ausgefüllt, erfasst mit der Webcam, welche für diese Teststellung verwendet wird

Da die Ergebnisse innerhalb einer Gruppe und Engine ziemlich ident waren und dadurch auf die anderen Tests innerhalb der gleichen Guppe mit der gleichen Engine geschlossen werden kann, werden folgend immer die Ergebnisse einer Gruppe für einen Zahlschein dargestellt. Aufgrund der unbekannten Qualität der zukünftig verarbeiteten Zahlscheine hinsichtlich Knitterfalten und Verschmutzung, der nicht vorhersehbaren Bedingungen bei der Belegerfassung, hinsichtlich Belichtung und der nicht vorhersehbaren Bedienung der Kamera, hinsichtlich der Genauigkeit der Benutzer beim Deponieren der Zahlungsanweisungen, können Ergebnisse im produktiven Betrieb nur sehr ungenau prognostiziert werden.

Abbildung 19 - Scan eines Zahlscheines mit 300 dpi, zeigt einen Beleg mit 300 dpi eingescannt.

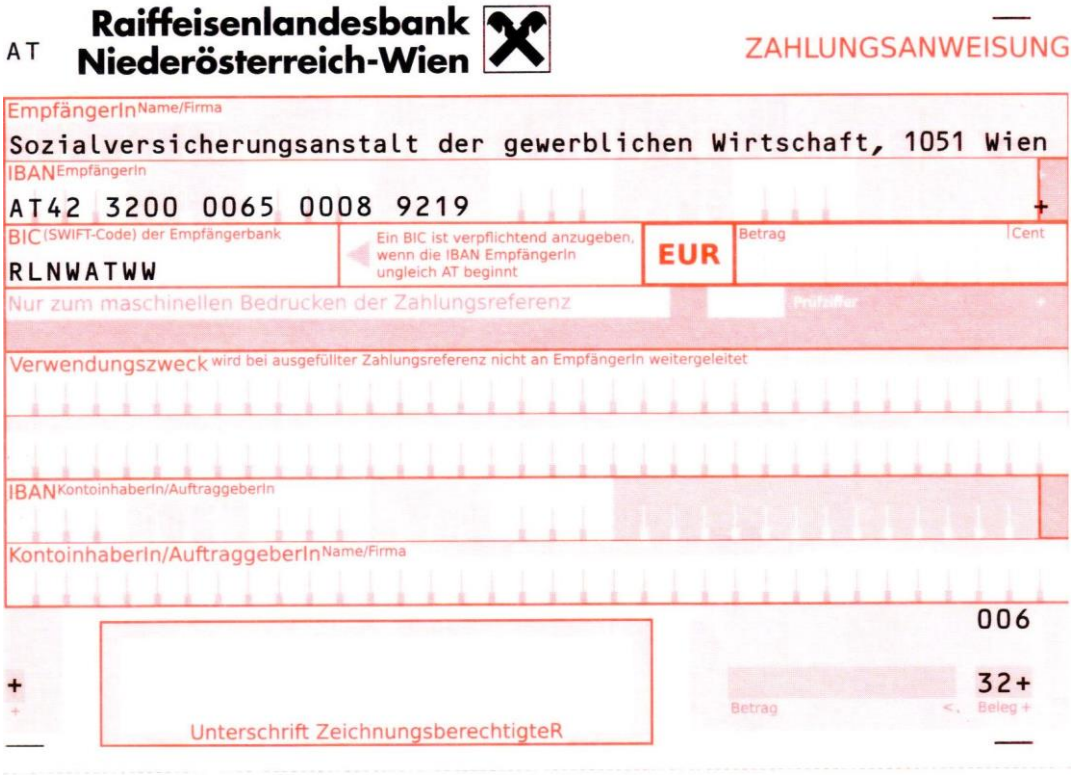

Abbildung 19 - Scan eines Zahlscheines mit 300 dpi

Bei Scans von hoher Qualität, wie in *Abbildung 19 - Scan eines Zahlscheines mit 300 dpi*, erreichten alle Engines fast immer 100% Genauigkeit. Wenn dies nicht der Fall war, musste die Länge der verglichenen Zeichenkette berücksichtigt werden. Da die Genauigkeit aufgrund der richtigen Zeichenanzahl berechnet wird, ist die Abweichung bei Fehlern in kurzen Zeichenketten sehr groß. Wenn zum Beispiel ein Zeichen eines 7-stelligen BIC-Codes falsch gelesen wird, beträgt die Übereinstimmung natürlich nur noch 86%.

Abbildung 20 - Scan eines Zahlscheines mit 300 dpi, Hand geschrieben, blau und *Abbildung 21 - Scan eines Zahlscheines mit 300 dpi, Hand geschrieben, schwarz* zeigen mit 300 dpi gescannte und mit Hand ausgefüllte Zahlungsanweisungen. Bei diesem Anwendungsfall erreichte die Abbyy Engine die besten Ergebnisse mit 70% bis 80%, je nachdem, mit welcher Genauigkeit die Zahlungsanweisung ausgefüllt war. Die Engine von Nuance erreichte ebenfalls 70% bis 80% Genauigkeit. Die Engines von Recogniform und Leadtools erreichten Ergebnisse zwischen 60% und 70%. Die Nicomsoft Engine erreichte bei Buchstaben nur bis zu 30%, bei Ziffern ebenfalls bis zu 80%.

Bei allen Engines musste allerdings explizit definiert werden, dass es sich beim Eingabebild um handgeschriebene Informationen handelt. Des weiteren musste der Bildbereich, welcher die handgeschriebene Information enthält, mittels Rechteck, durch 4 Eckpunkte, definiert werden.

Abbildung 20 - Scan eines Zahlscheines mit 300 dpi, Hand geschrieben, blau

Abbildung 21 - Scan eines Zahlscheines mit 300 dpi, Hand geschrieben, schwarz

Abbildung 22 - Zahlschein mit Webcam erfasst zeigt eine Zahlungsanweisung in der Qualität, mit welcher im Fall dieser Teststellung gearbeitet wird. In dieser Testgruppe erreichten alle Engines 80% bis 100%, abhängig von der Bildqualität und der Belichtung. 100% wurden allerdings nur von der Abbyy Engine erreicht, dies aber nur in 40% der Fälle.

Abbildung 22 - Zahlschein mit Webcam erfasst

Abbildung 23 - Zahlschein mit Webcam erfasst, Hand geschrieben zeigt eine mit der Webcam erfasste Zahlungsanweisung, welche handschriftlich ausgefüllt wurde. Hierbei erreichten die Engines von Abbyy, Nuance und Recogniform zwischen 50% und 70%. Bei den Ziffern lagen die Ergebnisse der Nicomsoft Engine im gleichen Bereich, hierfür musste allerdings definiert werden, dass der Bildbereich ausschließlich Ziffern enthält. Bei den Buchstaben, waren mit der Nicomsoft Engine nur maximal 20% korrekt, in den meisten Fällen mit bis zu 10% allerdings deutlich weniger. Die Erkennungsrate der Leadtools Engine lag bei 40% bis 50%.

Abbildung 23 - Zahlschein mit Webcam erfasst, Hand geschrieben

Zusammenfassend erzielten alle Engines ausgezeichnete Erfolge bei höchster Bildqualität und Maschinen-bedruckten Eingaben. Bei Maschinen-bedruckten Eingaben auf Belegen, welche mit der Webcam erfasst wurden, waren die Ergebnisse ebenfalls bei allen Engines größtenteils über 90%. Lediglich bei sehr schlechten Lichtverhältnissen gab es auch Ergebnisse im Bereich 80-90%.

Bei handschriftlich ausgefüllten Eingaben waren die Ergebnisse insgesamt unbrauchbar für den Anwendungsfall dieser Arbeit. Begründet wird dies durch die kleine Menge an verarbeiteten Daten pro Zahlungsanweisung. Wenn nach dem Aufwand für Belegpositionierung und Bildverarbeitung vom Benutzer bzw. von der Benutzerin zusätzlich der Aufwand betrieben werden muss, die Ergebnisdaten genau zu kontrollieren und sehr viel nachzubessern, erhöht sich der Gesamtaufwand gegenüber dem Anwendungsfall, die Daten von Beginn an manuell einzutippen.

Für den produktiven Einsatz müssen zwei weitere Dinge berücksichtigt werden. Erstens kann das Eingabebild vor der Verarbeitung mit OCR in seiner Qualität erhöht werden. Dies könnte zu Verbesserungen der Ergebnisse führen. Zweitens wurden alle Tests mit optimalen Eingaben an die Engines durchgeführt, da der ausführende Tester die Belege von den Engine Frontend Applikationen angezeigt bekam und die Engines optimal konfigurierte. Da der Nicomsoft OCR SDK kein Frontend besitzt, wurde dieser mit einer im Trial mitgelieferten Testapplikation getestet. Dies umfasste die genaue manuelle Lokalisierung der Bildteile mit den Texten sowie die Deklaration dieser Bildteile bezüglich Eingabeformat – zum Beispiel die Konfiguration an der Engine, dass es sich bei spezifischen Feldern um handschriftliche Eingaben handelt.

Diese optimale Konfiguration kann in dem Anwendungsfall dieser Arbeit nicht von den Benutzern bzw. Benutzerinnen durchgeführt werden, da dies gegen das Ziel der vereinfachten Bedienbarkeit sprechen würde.

Die Evaluierung der Engines führte zu folgenden Erkenntnissen, welche zur Entscheidungsfindung herangezogen wurden:

- Die Ergebnisse bei Maschinen-bedruckten Belegen sind bei allen getesteten Engines ähnlich.
- Die Nicomsoft Engine ist bei der Verarbeitung handschriftlicher Belege deutlich schlechter.
- Alle drei Engines sind, bezogen auf den Anwendungsfall dieser Arbeit, ungenügend bei der Verarbeitung handschriftlicher Belege.
- Die Ergebnisse werden schlechter, wenn die Engines betreffend Eingabebild, vom Benutzer weniger genau definiert und konfiguriert werden.
- Die Ergebnisse werden besser, wenn die Bildqualität nach der Erfassung des Bildes verbessert werden kann.
- Die Kosten von Nicomsoft OCR betragen, je nach EUR-USD-Wechselkurs, nur ungefähr zwischen 6% und 10% der Kosten der anderen Engines.

Durch die Testergebnisse wird die Verarbeitung von handschriftlich ausgefüllten Zahlscheinen von den Anforderungen abgegrenzt. Diese könnte jedoch als Inhalt von weiteren Untersuchungen herangezogen werden (siehe Kapitel 7. Ausblick).

Aufgrund der ähnlichen Ergebnisse bei Maschinen-bedruckten Belegen, sowie des Budgets für diese Arbeit, kombiniert mit dem signifikant niedrigeren Preis des Nicomsoft OCR SDKs, wurde dieser für die OCR Verarbeitung dieser Teststellung gewählt.

5.4. Treiber Implementierung

Folgend ist die gesamte Implementierung dokumentiert. Der Sourceode ist der beigelegten CD zu entnehmen. Die Applikation läuft in einem Prozess, wahlweise mit oder ohne GUI. Mit GUI kann die Applikation „stand-alone“ verwendet werden, sprich als eigenständiges Programm, welches auf Benutzereingaben im GUI reagiert. Ohne GUI läuft das Programm wie ein Treiber für LUI, welches wiederum auf den „Senioren Terminals“ läuft. Dabei wird der implementierte Treiber von LUI gesteuert. Der Benutzer arbeitet in diesem Szenario mit LUI, weiß also nicht, dass er indirekt den Treiber verwendet. In diesem Anwendungsfall wird der Treiber von LUI gestartet und die angebotenen Funktionen konsumiert. Die Unterscheidung bezieht sich rein auf die Bedienung, der Rest der Implementierung bleibt gleich.

Der Treiber bedient das am Computer oder Terminal angeschlossene Bilderfassungsgerät oder wahlweise eines davon, wenn mehrere verfügbar sind. Dies kann ein „Plug-and-Play-Gerät“, aber auch eine integrierte Komponente sein. Ausschlaggebend ist, dass die erforderlichen Treiber des Geräts am Betriebssystem installiert sind. Der Treiber führt eine Bildaufbereitung durch, um die Qualität des Dokumentes auf dem erfassten Bild zu erhöhen. Im Anschluss wird das erfasste Dokument in ein Raster unterteilt, um die einzelnen

Eingabefelder mittels OCR untersuchen zu können. Um OCR durchführen zu können, muss die entsprechende Engine am Computer, beziehungsweise am Terminal, installiert sein, beziehungsweise die erforderlichen Bibliotheken verfügbar sein, damit die im OCR SDK verwendeten Prozeduren und Funktionen auch entsprechende Funktionalität beinhalten.

LUI bietet 3 Möglichkeiten mit anderen Softwarekomponenten zu kommunizieren:

1. Named Pipes
2. DLLs
3. REST (Representational State Transfer)

Da REST zum Zeitpunkt der Implementierung noch nicht in LUI integriert war, verblieben die zwei Alternativen. Dieser Treiber verwendet *Named Pipes*, da es mit Named Pipes im Vergleich zu DLLs einfacher ist mit executables zu kommunizieren. Named Pipes sind unabhängig von Funktionsschnittstellen und der Treiber kann direkt über die Pipe von LUI aus gestartet werden.

A named pipe *is a named, one-way or duplex pipe for communication between the pipe server and one or more pipe clients. All instances of a named pipe share the same pipe name, but each instance has its own buffers and handles, and provides a separate conduit for client/server communication. The use of instances enables multiple pipe clients to use the same named pipe simultaneously.*
Any process can access named pipes, subject to security checks, making named pipes an easy form of communication between related or unrelated processes.
Any process can act as both a server and a client, making peer-to-peer communication possible. As used here, the term pipe server refers to a process that creates a named pipe, and the term pipe client refers to a process that connects to an instance of a named pipe.
Named pipes can be used to provide communication between processes on the same computer or between processes on different computers across a network [75].

Da LUI als Pipe Server agiert, verhält sich die Implementierung dieses Treibers wie ein Pipe Client. Das bedeutet LUI erzeugt zwei Pipes und der Treiber verbindet sich zu diesen. Hierbei dient eine Pipe zum Senden und eine zum Empfangen von Daten. Auch wenn es hinsichtlich Pipe Kommunikation möglich wäre, den Treiber auf einem anderen System laufen zu lassen als LUI, befinden sich in dieser Implementierung beide Komponenten am gleichen System. In dieser Implementierung startet LUI den Treiber. Der Dateipfad der Treiber-Startdatei ist in der LUI Konfiguration hinterlegt. Wenn der Treiber gestartet wird, verbindet er sich auf die von LUI bereitgestellte Pipe und die Kommunikation läuft.

Läuft der Treiber als eigenständige Anwendung, werden die extrahierten Informationen in eine Logdatei geschrieben, läuft der Treiber über LUI, werden die Informationen über die Pipe an LUI geschickt und dort zur Validierung durch den Benutzer angezeigt beziehungsweise zur weiteren Verarbeitung zwischengespeichert.

5.4.1. Design

Abbildung 24 - Klassendesign, zeigt die implementierten Klassen und deren Beziehungen (engl. association) zueinander. Ein Klassendiagramm, inklusive Definitionen der beinhalteten Prozeduren, Funktionen, Attribute, Properties und der Datenkapselung all dieser Klassenbestandteile, ist sowohl der Implementierung beigefügten Dokumentation wie auch dem Kapitel *a.1. Design des Treibers (Klassendiagramm)* zu entnehmen.

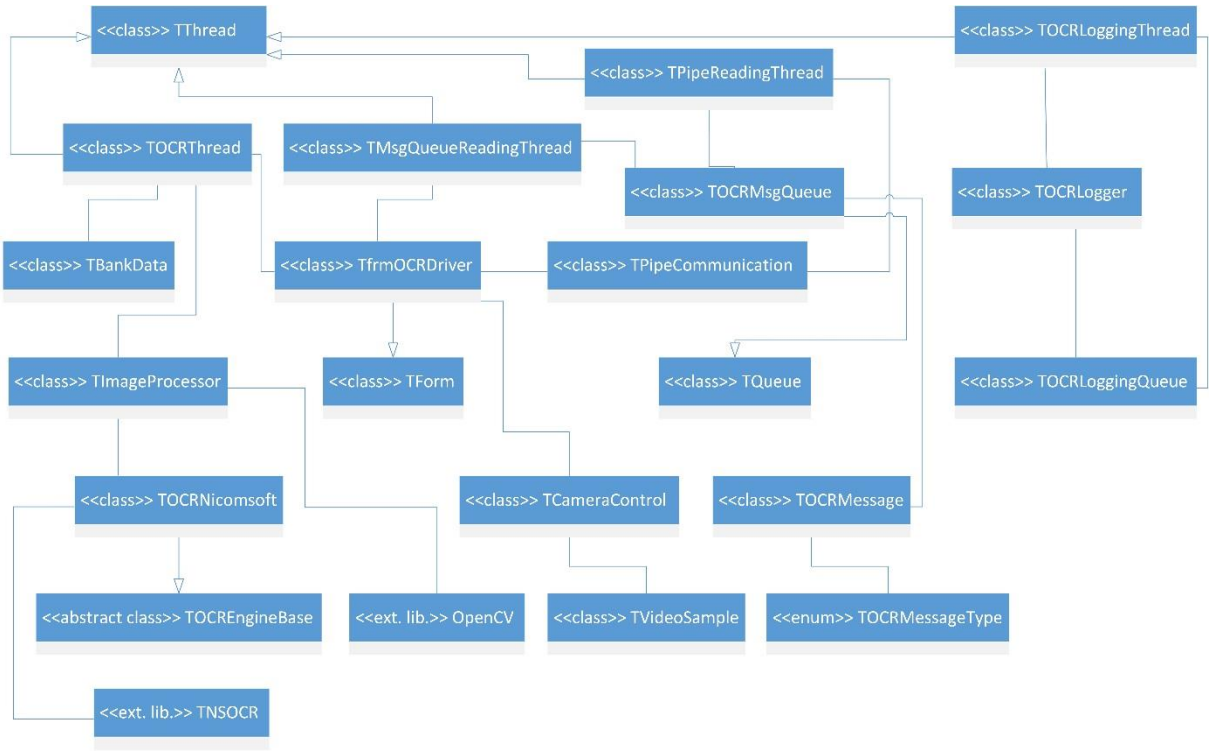

Abbildung 24 - Klassendesign

Folgende Auflistung enthält alle im Treiber implementierten Klassen inklusive kurzer Beschreibung. In Kapitel *5.4.2.Programmablauf* wird der Programmablauf anhand von Ablaufdiagrammen dargestellt. Zudem werden Implementierungsdetails erläutert, verwendete Algorithmen offengelegt und alternative Lösungsansätze diskutiert. Die Zwischenergebnisse der Schritte von Bild- sowie OCR-Verarbeitung werden anhand von Abbildungen dargestellt.

Die Treiberimplementierung ist auf folgende Units und Klassen aufgeteilt:

- Unit *OCRDriverGUI*
    - Klasse *TfrmOCRDriver*: ist die Einstiegsklasse der Applikation. Sie enthält ein GUI, welches im Testmodus angezeigt, im Live-Modus ausgeblendet wird. Die Steuerung und Koordination der Treiberfunktionalitäten findet statt. Objekt-Instanzen der Klassen für internes Logging (*TOCRLogger*), für Kommunikation mit LUI via Named Pipes (*TPipeCommunication*), für Abarbeitung der Treiber-

internen Nachrichten (*TMsgQueueReadingThread*), für die Webcam-Steuerung (*TCameraControl*) wie auch für die Bild- und OCR-Verarbeitung (*TImageProcessor*) werden verwaltet

  - Klasse *TMsgQueueReadingThread*: liest die Treiber-interne message queue (TOCRMsgQueue) und verarbeitet die Nachrichten je nach Nachrichten-Typ
  - Klasse *TOCRThread*: führt die Bild- und die OCR-Verarbeitung aus. Grundsätzlich beides asynchron zur Kommunikation mit LUI, zur Treiber-internen Nachrichtenverarbeitung und zum Treiber-internen Logging. Die OCR-Verarbeitung kann jedoch nur asynchron stattfinden, wenn dies die dafür verwendete Engine unterstützt

- Unit *ocrLogging*
  - Klasse *TOCRLogger*: implementiert Applikations-internes Logging unter Verwendung einer *queue* (*TOCRLoggingQueue*) und eines Lese-Threads (*TOCRLoggingThread*)
  - Klasse *TOCRLoggingQueue*: *queue* um *logging messages* zu puffern
  - Klasse *TOCRLoggingThread*: liest die Treiber-interne *logging queue* (*TOCRLoggingQueue*) und schreibt alle Logging-Einträge in einem konfigurierbaren Intervall auf die Festplatte
- Unit *ocrMessageProcessing*
  - Klasse *TOCRMsgQueue*: speichert alle Treiber-internen Nachrichten (*TOCRMessage*). Dies können entweder von LUI via *Named Pipe* empfangene Nachrichten sein, Nachrichten welche an LUI gesendet werden sollen oder Nachrichten mit Treiber-internen Befehlen
  - Klasse *TOCRMessage*: Container für Treiber-interne Nachrichten
  - Enumeration *TOCRMessageType*
- Unit *ocrPipeCommunication*
  - Klasse *TPipeCommunication*: implementiert Kommunikation mit LUI via *Named Pipes*
  - Klasse *TPipeReadingThread*: liest die geöffnete *incoming pipe* zu LUI und speichert die gelesenen Nachrichten in die Treiber-interne *message queue* (*TOCRMsgQueue*)
- Unit *ocrBankDataContainer*
  - Klasse *TBankData*: Container für die extrahierten Überweisungsinformationen
- Delphi *unit ocrVideoSampleGrabber*
  - Sammlung an Klassen, welche den Zugriff auf Videoaufnahmegeräte via *DirectShow* bereitstellen [76]
- Unit *ocrCameraControl*

- Klasse *TCameraControl*: bietet eine Schnittstelle zur Konfiguration und Steuerung von Videoaufnahmegeräten unter Verwendung der *unit ocrVideoSampleGrabber* für den Hardware-Zugriff

- Unit *ocrImageProcessing*
  - Klasse *TImageProcessor*: führt Bildbearbeitung durch:
    - Bildnormalisierung
    - Berechnung von Helligkeit, Kontrast, Hue, Sättigung, Value und Mittelwerte der RGB-Werte des Bildes
    - Bestimmung von Schwellenwerten und Bildbinarisierung
    - Linien- und Rechteck-Erkennung in den Bilddaten
    - Winkelberechnungen und Bildrotation
    - Bild-Cropping
    - Morphologische Transformationen
    - Aufrufe der OCR-Verarbeitung für Bildfragmente
- Unit *ocrEngineBaseClass*
  - Klasse *TOCREngineBase*: Basisklasse für OCR *engines*
- Unit *ocrEngineNicomsoft*
  - Klasse *TOCRNicomsoft*: Schnittstelle zur Verwendung des *Nicomsoft OCR SDK*. Bietet Funktionalität zur Verwendung der Engine.
    Um eine andere OCR Engine in dieser Treiber-Implementierung zu verwenden muss eine Klasse analog zu *TOCRNicomsoft* implementiert werden. Diese muss ebenfalls von *ocrEngineBaseClass* erben und somit folgende Methoden überschreiben:
    - *LoadDllAndInitializeEngine*: lädt die Bibliothek der eingesetzten Engine und initialisiert die Engine
    - *DoOcr(TBitmap)*: führt die OCR-Verarbeitung auf das als Funktionsargument übergebene Bitmap-Objekt aus
  - In der Klasse *ocrImageProcessing* muss beim Erzeugen der konkreten, von *ocrEngineBaseClass* abgeleiteten, Objektinstanz die entsprechende, mit der eingesetzten OCR Engine korrespondierende, Klasse verwendet werden.
- Unit *ocrCommandsAndConfig*: enthält global definierte, standardisierte Kommandos zum Nachrichtenaustausch zwischen dieser Implementierung und LUI und eine Sammlung an Konfigurationsparametern für diesen Treiber die verwendet werden, wenn die entsprechenden Parameter nicht via Konfigurationsdatei übergeben werden. Folgende Liste enthält diese Konfigurationsparameter:
  - Name der *Named Pipes*
  - Suffix für die *incoming pipe*
  - Suffix für die *outgoing pipe*

- Verzeichnis, in das Debug-Informationen gespeichert werden
- Flag, zum De-/Aktivieren des Treiber-internen Logging
- Name des Log-Files
- Logging-Intervall indem die *logging queue* auf die Festplatte geschrieben wird (in Millisekunden)
- Flag zum De-/Aktivieren des Logging von verfügbaren Videoaufnahmegeräten und deren Auflösung
- Flag zum De-/Aktivieren der Speicherung von Zwischenergebnissen der Bildbearbeitung
- Flag zum De-/Aktivieren des Logging der extrahierten Überweisungsinformation
- Interface-Name des Videoaufnahmegerätes, welches zur Bildaufnahme herangezogen werden soll
- Index der Kamera-internen Auflösung (siehe Kapitel *5.7. Camera-Resolution-Tool für Videoaufnahmegeräte*)
- Flag zum De-/Aktivieren der Nutzung der Kamera-Standardkonfiguration
- Helligkeitswert der Kamera (0-255)
- Kontrastwert der Kamera (0-255)
- Sättigungswert (Farbintensität) der Kamera (0-255)
- Schärfewert der Kamera (0-255)
- Flag zum De-/Aktivieren des automatischen Weißabgleichs der Kamera
- Weißabgleichswert der Kamera (2000-6500)
- Flag zum De-/Aktivieren der automatischen *backlight compensation* der Kamera
- Flag zum De-/Aktivieren der Benützung eines eigenen Thread für die OCR Verarbeitung der Engine

- Unit *ocrCommandsAndConfig*
  - Klasse *TOCRConfiguration*: liest die gewünschte Konfiguration beim Programmstart aus einer Konfigurationsdatei; bei fehlender Konfigurationsdatei oder einzelnen fehlenden Konfigurationsparametern in der Datei werden die Standardkonfigurationsparameter aus der Unit verwendet

5.4.2. Programmablauf

Nachfolgend wird der Programmablauf illustriert. Um asynchrone Verarbeitung zu gewährleisten, wurde Multi-Threading verwendet. Das Hauptprogramm, die Kommunikation mit LUI, das

Logging und die Bildbearbeitung laufen jeweils in einem eigenen Thread. Wenn die eingesetzte OCR engine asynchrone Verarbeitung unterstützt, erfolgt auch die OCR-Verarbeitung asynchron. Um zwischen den einzelnen Thread zu kommunizieren, wurde eine Message Queue implementiert. Um die Nachrichten der Message Queue zu verarbeiten wurde ebenfalls ein eigener Thread implementiert. Um die Queue *thread-safe* zu implementieren wurden kritische Abschnitte (engl. *critical section*) eingesetzt. Auf diese Weise kann gewährleistet werden, dass nur ein Thread zur gleichen Zeit auf die Queue zugreift und die in der Queue enthaltenen Daten somit einen konsistenten Datenbestand abbilden.

In traditional operating systems, each process has an address space and a single thread of control. In fact, that is almost the definition of a process. Nevertheless, in many situations, it is desirable to have multiple threads of control in the same address space running in quasi-parallel, as though they were (almost) separate processes (except for the shared address space) [77].

Abbildung 25 - Flowchart: Driver Application Start zeigt die Prozessschritte, welche unmittelbar nach dem Programmstart ausgeführt werden.

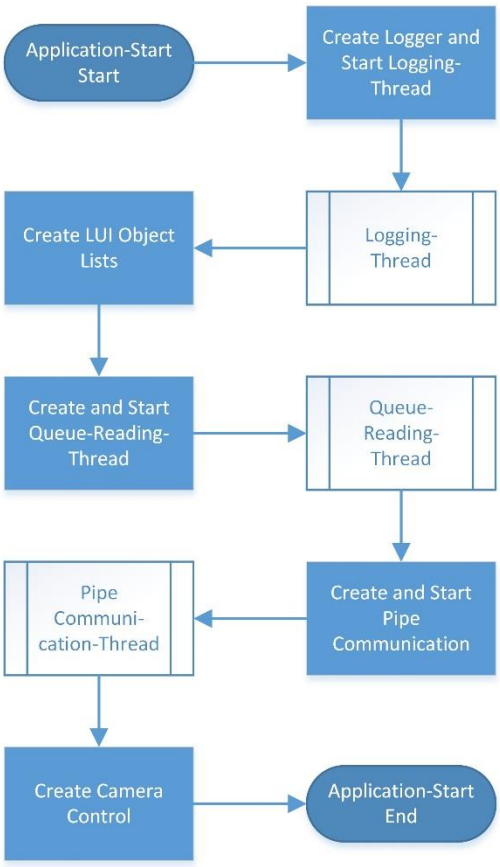

Abbildung 25 - Flowchart: Driver Application Start

Nach Instanzierung und Start des Treiber-internen Loggers (siehe Kapitel *5.4.2.1. Logging*) werden Listen für LUI Objekte erstellt. Da das Design von LUI zu 100% von diesem Treiber

abstrahiert ist, müssen die in LUI verwendeten GUI-Objekte zur Laufzeit an den Treiber übermittelt werden, damit dieser die entsprechenden Felder kennt, in welche die vom Zahlschein abstrahierten Informationen geschrieben werden sollen.

Im Anschluss wird der Queue-Reading-Thread instanziert und gestartet. Dieser liest die Treiber-interne *message queue* und verarbeitet die darin enthaltenen Nachrichten. Wenn die *message queue* verfügbar ist, kann eine Instanz zur Pipe Kommunikation erstellt und gestartet werden. Der Pipe Communication-Thread liest Daten von der Named Pipe zu LUI und schreibt die gelesenen Daten in die Treiber-interne *message queue* (siehe Kapitel *5.4.2.2. Message Queuing and Pipe Communication*).

Zuletzt wird eine Instanz zur Steuerung der am PC/Tablet-PC/Terminal angeschlossenen Bild-/Video-Aufnahmegeräten erstellt (siehe Kapitel *5.4.2.3. Kamerasteuerung*).

5.4.2.1. Logging

Abbildung 26 - Flowchart: Logging Thread zeigt die Prozessschritte des Treiber-internen Logging-Mechanismus.

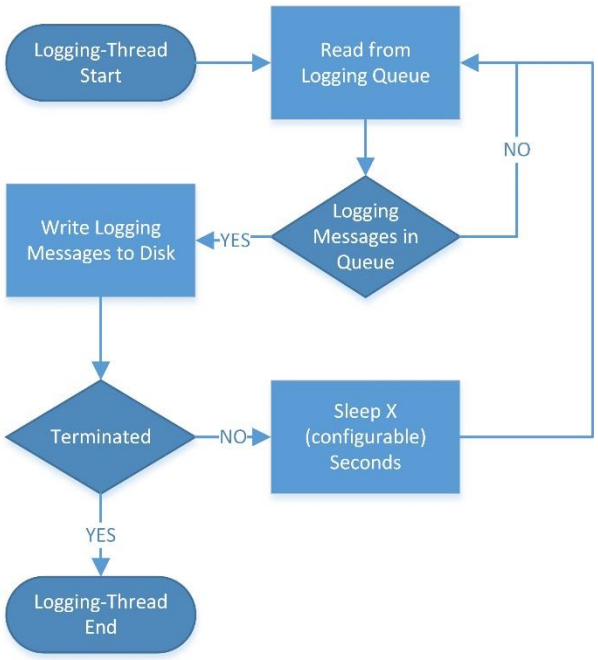

Abbildung 26 - Flowchart: Logging Thread

Bis zur Beendigung des Threads werden alle Nachrichten der *logging queue* in die, in der Konfiguration definierten Datei, geschrieben. Danach wird der Thread pausiert, um keine CPU-

Ressourcen unnötig zu blockieren. Um die *thread-safety* der *logging queue* zu gewährleisten wurden kritische Abschnitte eingesetzt.

Die Konfiguration des Loggers kann in der Konfigurationsdatei vorgenommen werden.

5.4.2.2. Message Queuing and Pipe Communication

Abbildung 27 - Flowchart: Message-Queue Reading-Thread stellt den Ablauf der Treiber-internen Nachrichten-Verarbeitung dar.

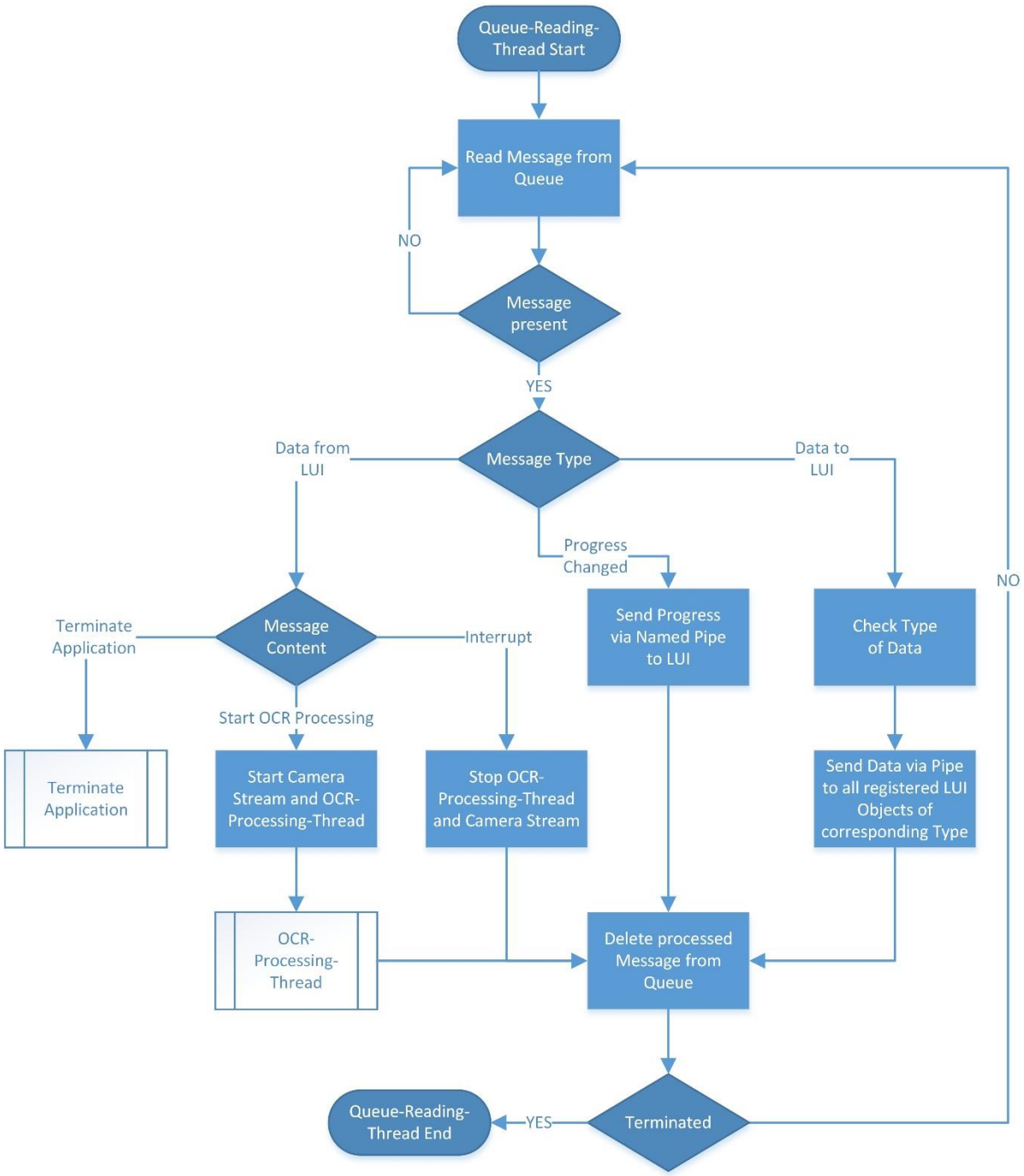

Abbildung 27 - Flowchart: Message-Queue Reading-Thread

Bis zur Beendigung des Threads werden die Nachrichten von der Treiber-internen *message queue* gelesen und entsprechend des Nachrichten-Typs verarbeitet. Hierbei werden drei verschiedene Nachrichten-Typen unterschieden:

1. *Message Type* „Data from LUI“: Nachrichten dieses Typs wurden von der eingehenden Named Pipe von LUI gelesen und zur Verarbeitung in die *message queue* gespeichert. Die Nachrichten von LUI sind im Format XML. Zur Bearbeitung wurde die Bibliothek *NativeXml* verwendet [78]. Es werden drei verschiedene Inhalte unterschieden:
    a. „Terminate Application“: der Befehl, die Treiber-Applikation zu beenden; dieser Befehl wird indirekt ausgeführt, wenn zumindest eine der beiden Named Pipes von bzw. zu LUI geschlossen ist (siehe Kapitel *5.4.2.4. Application Termination*)
    b. „Start OCR Processing“: der Befehl, die Treiber-interne Bild- und OCR-Verarbeitung durchzuführen. Dazu gehören das Erfassen eines Bildes von der Kamera, das Vorverarbeiten des Bildes im Sinne von Entfernung von Störsignalen, das Auffinden des gesuchten Bildteils (Informationsfelder des Zahlscheines) im Zuge der Rechteckserkennung, das Filtern des Bildtextes sowie das Verarbeiten der entsprechenden Bildteile mit einer OCR Engine (siehe Kapitel *5.4.2.5. Image and OCR Processing*)
    c. „Interrupt“: der Befehl, die Treiber-interne Bild- und OCR-Verarbeitung abzubrechen. Dieser Befehl wird indirekt im Zuge von LUI-seitigen Unterbrechungen des Menüs, welches die OCR-Treiber-Funktionalität anbietet, gegeben. Hierbei wird zum Beispiel die LUI-seitige Anzeige der OCR-Verarbeitung durch eingehende Telefonanrufe, medizinische Notfälle oder sonstiges unterbrochen

2. *Message Type* „Progress Changed“: Der Prozessfortschritt der Bild- und OCR-Verarbeitung wird laufend in die Treiber-interne *message queue* gespeichert. Beim Lesen aus der *queue* wird der Prozessfortschritt im Testmodus am GUI angezeigt. Im Live-Modus wird der Prozessfortschritt zusätzlich über die ausgehende Named Pipe an LUI gesendet, um dort angezeigt werden zu können

3. *Message Type* „Data to LUI“: Nachrichten dieses Typs werden über die ausgehende Named Pipe an LUI geschrieben. Hierbei handelt es sich um Funktionsaufrufe von LUI-internen Funktionen oder um Informationen, welche in LUI angezeigt werden sollen. Je nach Unterscheidung der zu sendenden Information ist der Nachrichtaufbau bzw. –inhalt verschieden (siehe Kapitel *5.6. Integration in LUI*)

Abbildung 28 - Flowchart: Pipe Communication-Thread zeigt den Ablauf des Threads welcher die eingehende Named Pipe von LUI liest. Hierbei werden bis zur Beendigung des Threads die gelesenen Nachrichten in die Treiber-interne *message queue* geschrieben. Die Pipe Kommunikation wird gestartet, indem zwei Named Pipes erstellt und geöffnet werden. In dieser Anwendung werden die Pipes von LUI geöffnet und der Treiber verbindet sich zu diesen. Die notwendigen Informationen betreffend der beiden Pipes sind:

- Name der Pipe
- InSuffix
- OutSuffix

Diese werden in der LUI Konfiguration definiert und müssen in der Konfiguration des Treibers ident definiert werden, um einen Kommunikationskanal öffnen zu können.

Die Konfiguration der Pipe Namen kann in der Konfigurationsdatei der Treiber-Applikation vorgenommen werden.

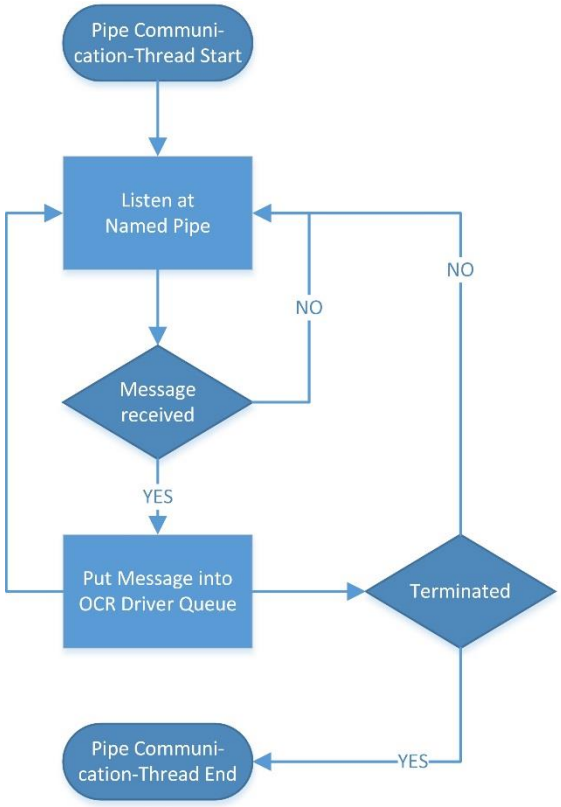

Abbildung 28 - Flowchart: Pipe Communication-Thread

5.4.2.3. Kamerasteuerung

Um die am PC/Tablet-PC/Terminal angeschlossenen Videoaufnahmegeräte verwenden zu können, müssen diese über deren Treiber angesprochen werden. Da es sich bei dem Treiber dieser Lösung um eine Windows-Anwendung handelt, wurde die Kamerasteuerung mit dem *Multimedia Framework und API DirectShow* umgesetzt. *DirectShow* war ursprünglich Teil des *DirectX SDK*, ist jetzt aber Teil des *Windows SDK*, ehemals *Plattform SDK*.

Für die Kommunikation mit dem Treiber der Kamera wurde eine von *M. Braun* [76] entwickelte Portierung der C++ Beispiele des Microsoft DirectX 9.0 SDKs verwendet. Diese implementiert die entsprechenden Interfaces und registriert die entsprechenden Callback-Methoden. Die externen Quelltext-Aufrufe werden mit Standard-Aufrufen (engl. standard call) durchgeführt. Im Zuge dieser Arbeit wurde die Implementierung von *M. Braun* um ein Speichermanagement erweitert.

Das Videoaufnahmegerät und die Auflösung die verwendet werden sollen, müssen in der LUI-Konfiguration des Treibers vorgenommen werden (siehe Kapitel *5.6. Integration in LUI*). Die

verfügbaren Videoaufnahmegeräte und deren verwendbare Auflösungen können mit einem, explizit dafür implementierten, Tool, dem *CameraResolutionTool*, ausgelesen werden (siehe Kapitel *5.7. Camera-Resolution-Tool für Videoaufnahmegeräte*). Falls der Treiber ohne LUI verwendet wird, kann die Konfiguration der Kamera in der Konfigurationsdatei vorgenommen werden.

Die Einstellungen der Kamera betreffend Aufnahmekonfiguration (Helligkeit, Kontrast, Zoom etc.) können ebenfalls in der Konfigurationsdatei vorgenommen werden.

5.4.2.4. Application Termination

Abbildung 29 - Flowchart: Application Termination zeigt die notwendigen Prozessschritte um die Treiber-Applikation unter Freigabe des allokierten Speichers ordnungsgemäß zu beenden.

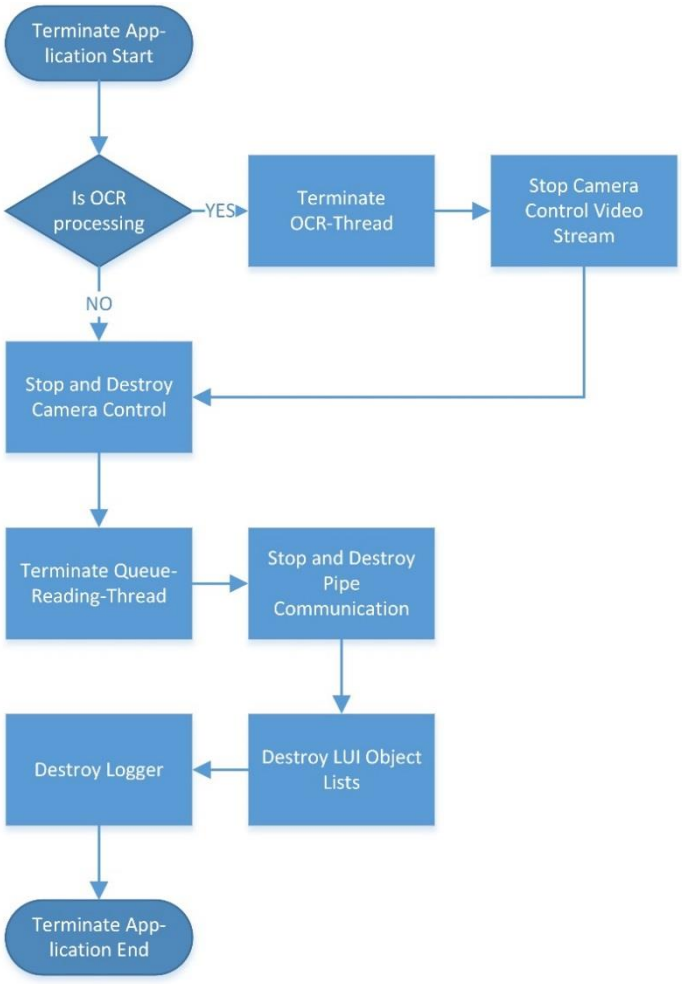

Abbildung 29 - Flowchart: Application Termination

Hierfür muss der Thread zur Bild- und OCR-Verarbeitung beendet werden, falls dieser läuft. Zudem wird der Stream zur Kamera geschlossen, falls dieser offen ist, und die Instanz der Kamerasteuerung wird freigegeben. Folgend wird der Queue-Reading-Thread beendet und die Instanz freigegeben. Nach dem Schließen der Named Pipes und dem Freigeben der Instanzen werden die Listen, mit denen LUI Objekte verwaltet werden, freigegeben. Zuletzt wird die Instanz des Loggers freigegeben und die Applikation beendet.

5.4.2.5. Image and OCR Processing

Auf die erfassten Bilder der Erlagscheine müssen folgende Prozessschritte angewandt werden:

1. Vorverarbeitung des Bildes. Da die Bilder unter verschiedenen Bedingungen hinsichtlich der vorhandenen Lichtquellen erfasst werden und die Bilder somit in den Eigenschaften Helligkeit, Kontrast, Bildrauschen und Farbsättigung variieren, muss versucht werden, dies möglichst gut auszugleichen. Dieser Schritt inkludiert folgende Punkte:
    a. Erkennung/Verbesserung von Bildhelligkeit und Kontrast
    b. Ausgleich von Überbelichtungen/Normalisierung des Farbspektrums/Erhöhung der Bildschärfe
    c. Verringerung von Bildstörungen/Bildrauschen

2. Erkennung des maßgeblichen Bildbereichs, nämlich des Rechteckes, das die Informationen enthält, welche extrahiert werden sollen. Da die Bilder unter verschiedenen Bedingungen hinsichtlich der Positionierung vor dem Aufnahmegerät erfasst werden, kann von keiner einheitlichen Struktur der Bildinhalte auf den Aufnahmebildern ausgegangen werden. Durch die unterschiedliche Erfassung hinsichtlich der Belichtung und Positionierung können auch die Mechanismen der Blindfarbe und der Positionsmarkierungen nicht genutzt werden. Die Analyse des Layouts muss somit manuell durchgeführt werden. Dieser Schritt inkludiert folgende Punkte:
    a. Erkennung des Layouts mit Hilfe von Mechanismen wie Farbfiltern und Kantendetektion, gefolgt von Objekterkennung wie z.B. Linien- oder Rechteckserkennung oder sonstiger Merkmalerkennung
    b. Erkennung und Ausgleich von Bildschrägläufen bzw. Bildrotation

3. Berechnung der relevanten Bildbereiche, in denen die gesuchten Informationen auf den Zahlscheinen gedruckt sind

4. Qualitative Aufbereitung der relevanten Bildbereiche, um die bedruckte Schrift möglichst gut erkennbar für die OCR Verarbeitung zu machen

5. Verarbeitung der relevanten Bildbereiche mit einer OCR Engine

Dabei werden sämtliche Prozessschritte, sowie dabei erzielte Ergebnisse und aufgetretene Fehler im Applikations-Log dokumentiert.

Abbildung 32 - Flowchart: Image processing zeigt die Prozessschritte zur Durchführung der zentralen Treiberaufgaben. Diese umfassen das Erfassen eines Bildes von einem offenen Stream zu einem verfügbaren Videoaufnahmegerät, die Vorverarbeitung des Bildes im Sinne der Anwendung von Störfiltern, Bildbearbeitungsalgorithmen, Rechteckserkennung und geometrischer Transformation und die Texterkennung durch Verarbeitung der Bilddaten mittels OCR, unter Verwendung einer externen, eingebundenen OCR Engine.

Der detaillierte Ablauf der einzelnen Schritte, die verwendeten Algorithmen und Techniken, dabei getroffene Entwicklungsentscheidungen, alternative Lösungsansätze, sowie Zwischenergebnisse der Verarbeitung sind in Folge beschrieben und dargestellt.

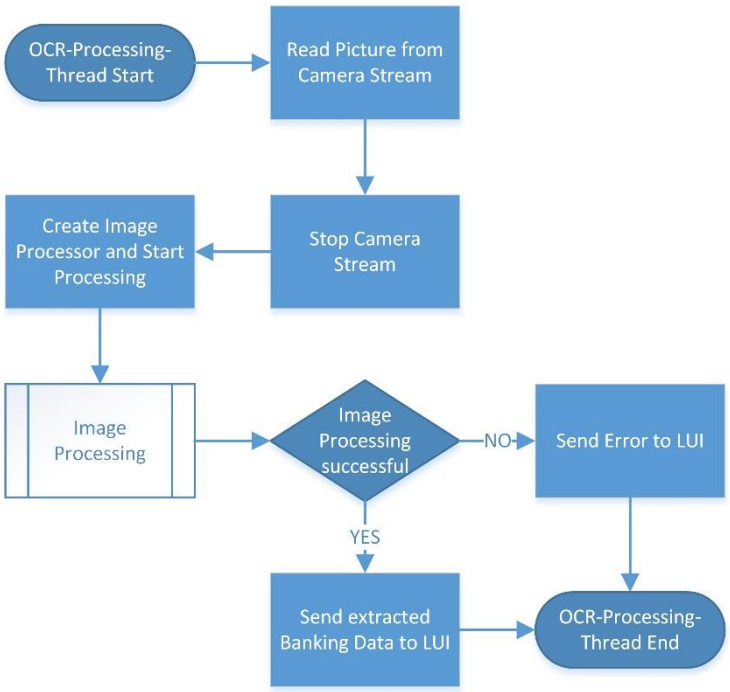

Abbildung 30 - Flowchart: OCR-Processing-Thread

Die gesamte Bildverarbeitung läuft in einem eigenen Thread, dem OCR-Processing-Thread, um die asynchrone Verarbeitung parallel zur Kommunikation mit LUI zu ermöglichen. Lediglich die asynchrone Verwendung der OCR-Verarbeitung hängt von der eingesetzten OCR Engine ab, da diese asynchrone Verarbeitung unterstützen muss, um die aufrufende Applikation nicht zu blockieren. Zur Bildbearbeitung wurde die Bibliothek *OpenCV*[2] verwendet. Hierfür wurde der *Delphi wrapper*, der *OpenCV library header files*, von Laentir Valetov und Mikhail Grigorev eingebunden [79]. *OpenCV* wurde gewählt da es eine dem Stand der Technik entsprechende

[2] http://opencv.org/

freeware Komponente ist. Die *OpenCV* Bibliotheken beinhalten eine Vielzahl an Filter und Algorithmen zur Bildbearbeitung, gelten als etabliert und stabil und sind in Delphi leicht einzubinden, da es sich um C und C++ Bibliotheken handelt.

Im ersten Prozessschritt wird ein Bild vom offenen Stream zur eingesetzten Kamera gelesen. Im Anschluss wird der Stream zur Kamera geschlossen, da dieser nicht länger benötigt wird. Bei einem zukünftigen, erneuten Aufruf der Verarbeitung wird der Stream erneut geöffnet.

Abbildung 31 - Input Image captured from Cam zeigt ein von der Webcam dieser Teststellung (Logitech Pro Webcam C920) erfasstes Bild. Das erfasste Bild wird in das Format 24-Bit-RGB konvertiert, falls es nicht bereits in diesem Format erfasst wurde. 24-Bit-RGB bedeutet, dass jeder Pixel durch drei Farbkanäle (Werte), stellvertretend für Rot (engl. red), Grün (engl. green) und Blau (engl. blue) definiert ist. Jeder dieser drei Werte ist durch 8 Bit codiert, dies ergibt pro Wert einen Wertebereich von 0 bis 255.

Abbildung 31 - Input Image captured from Cam

Zu beachten ist die im Papier gefaltete Kante, welche entlang der Unterkante des Unterschriftsfeldes vorhanden ist. Diese führt dazu, dass der Bildbereich unter der Kante bei der Aufnahme weniger belichtet und somit im Bild dunkler ist. Veränderungen an der Papieroberfläche wie diese müssen bei der Bildbearbeitung berücksichtigt werden.

Im nächsten Schritt wird eine Instanz des Treiber-internen *Image Processors (TOcrImageProcessor)* erstellt. Diesem wird das zuvor gelesene Bild übergeben. Die Verarbeitung wird gestartet (siehe *Abbildung 32 - Flowchart: Image processing*).

Wenn die Bild- und OCR-Verarbeitung erfolgreich war, werden die extrahierten Überweisungsinformationen via ausgehender Named Pipe an LUI gesendet. Bei nicht erfolgreicher Verarbeitung wird ein entsprechender Fehler geloggt und an LUI gesendet.

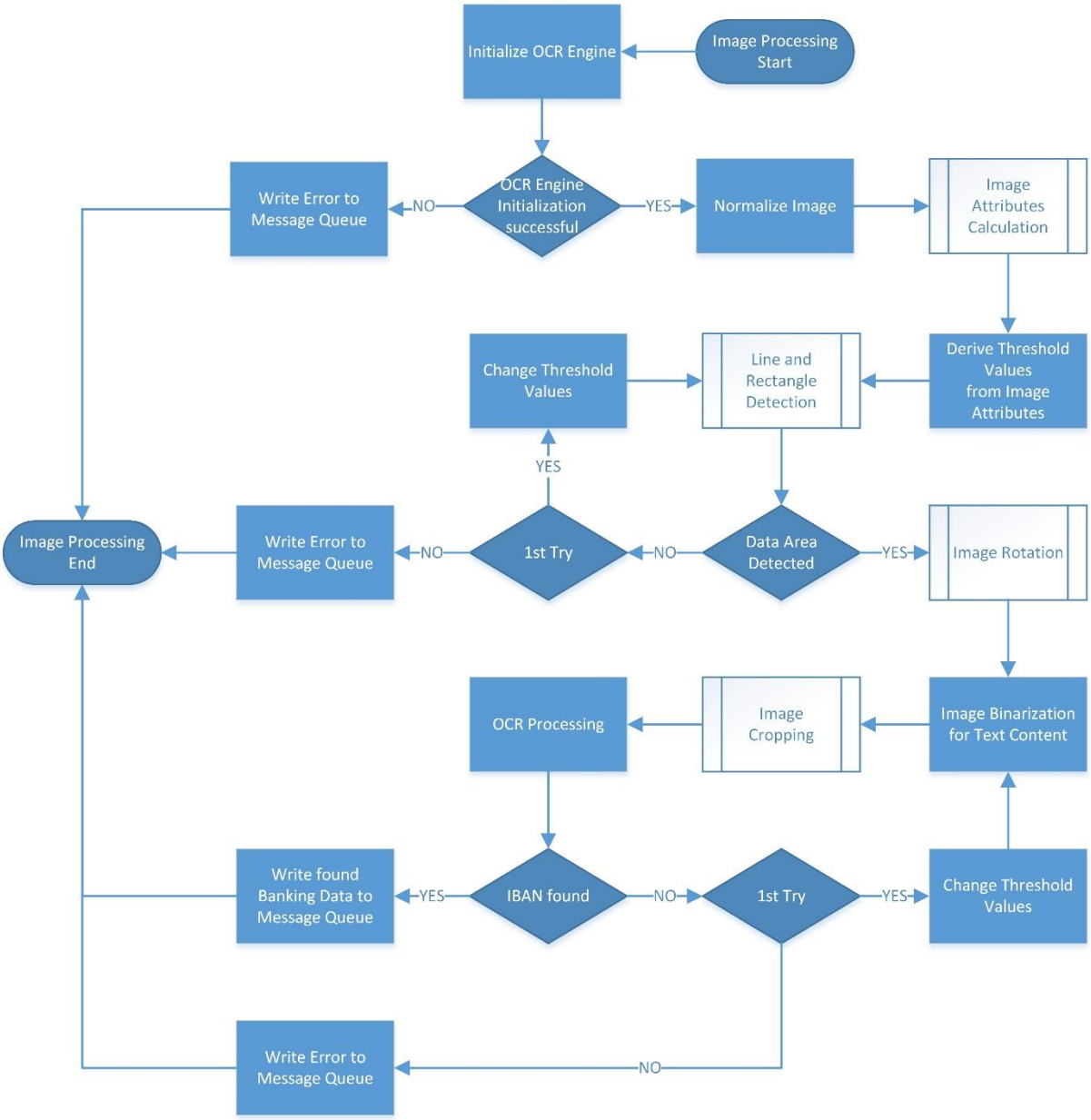

Abbildung 32 - Flowchart: Image processing

Bevor die Bild- und OCR-Verarbeitung durchgeführt wird, wird die OCR Engine geladen und initialisiert. Das hat den Zweck, dass die Speicher-intensive Bildverarbeitung bei OCR Engine-seitigem Fehler nicht durchgeführt wird. In diesem Fall wird ein Fehler geloggt und an LUI übermittelt.

5.4.2.5.1. Bildnormalisierung

Konnte die OCR Engine erfolgreich geladen und initialisiert werden, wird das Eingabebild normalisiert (engl. *normalized*). Hierbei wird der Wertebereich aller Pixel pro Farbkanal auf den maximalen Wertebereich (0-255) skaliert, indem jedes Pixel mit der linearen Formel

$$g(x) = \alpha * f(x) + \beta$$

neu berechnet wird. Hierbei ist *g(x)* der neu berechnete Wert des Pixels, f(x) der ursprüngliche Wert des Pixels, α eine multiplikative Konstante und β eine additive Konstante. Die beiden Konstanten errechnen sich hierbei wie folgt:

$$\alpha = 255 / (a - b)$$

wobei a der höchste auftretende Pixelwert des Farbkanals und b der niedrigste auftretende Pixelwert des Farbkanals ist. Beträgt der höchste Wert 255 und der niedrigste Wert 0, kann dieser Farbkanal nicht auf die Skala 0-255 skaliert werden, da bereits alle Werte genützt werden.

$$\beta = - \alpha * f(x)$$

Mit der additiven Konstante β wird der durch α berechnete 255 große Wertebereich auf den Startwert 0 und den Endwert 255 skaliert.

Kamera-erfasste Bilder, welche unter Überbelichtung erfasst wurden, nutzen den Wertebereich von 0-255 zum Beispiel nicht aus. *Abbildung 33 - Normalized Image* zeigt ein überbelichtet erfasstes Bild vor (in der Abbildung oben) und nach (in der Abbildung unten) der Normalisierung. Bei der bedruckten Schrift ist die Erhöhung des Kontrasts deutlich erkennbar. Eine Erhöhung des Kontrasts führt dazu, dass helle Farben heller und gleichzeitig dunkle Farben dunkler werden. Der Grenzwert zwischen hell und dunkel ergibt sich hierbei aus der linearen Umwandlung, nämlich jener Pixelwert für den f(x) = g(x) gilt.

Abbildung 33 - Normalized Image

5.4.2.5.2. Berechnung von Bildeigenschaften

Nach der Normalisierung des Eingabebildes werden folgende Bildeigenschaften errechnet (siehe *Abbildung 34 - Flowchart: Image Attributes Calculation*):

- Helligkeit
- Kontrast
- Durchschnittlicher *Hue*-Wert
- Durchschnittlicher *Saturation*-Wert
- Durchschnittlicher *Value*-Wert

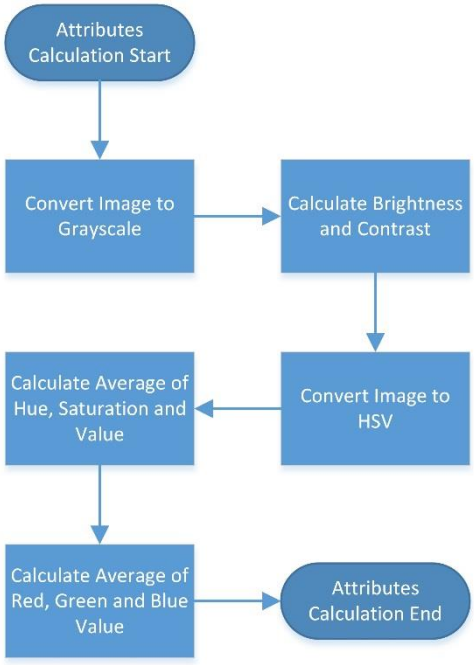

Abbildung 34 - Flowchart: Image Attributes Calculation

Um die Helligkeit und den Kontrast des Bildes berechnen zu können, muss es zuvor in ein Grauwertbild (engl. *grayscale image*) umgewandelt werden. In einem Grauwertbild ist jedem Pixel anstatt der drei Werte der Farbkanäle RGB ein Grauwert des Wertebereichs 0-255 zugeordnet. Dieser Grauwert berechnet sich dabei wie folgt:

Grauwert = 0.299 * Rotwert + 0.587 * Grünwert + 0.114 * Blauwert

Abbildung 35 - Grayscale Image zeigt das Bild aus *Abbildung 31 - Input Image captured from Cam* umgewandelt in ein Grauwertbild.

Die Helligkeit des Bildes ist der Durchschnitt der Grauwerte aller Pixel.

Der Kontrast des Bildes ist die durchschnittliche Standardabweichung aller Pixel vom Durchschnitt der Grauwerte (von der Helligkeit).

Abbildung 35 - Grayscale Image

Um die durchschnittlichen Werte von *Hue*, *Saturation* und *Value* des Bildes berechnen zu können, muss es zuvor vom RGB-Farbraum in den HSV-Farbraum umgewandelt werden.

Hue repräsentiert dabei die Farbe auf einem Farbkreis von 360° oder einem Farbkreis indem die 360° auf einen größeren oder einen kleineren Bereich skaliert wurden.

Saturation definiert die Farbsättigung des Bildes. Die Sättigung wird grundsätzlich in Prozent angegeben, wobei 0% grau, 50% halb-gesättigt und 100% voll-gesättigt (reine Farbe) repräsentieren. Wahlweise können die Prozentangaben auch auf eine andere Skala abgebildet werden, wie zum Beispiel auf Werte zwischen 0 und 1, oder auf Werte zwischen 0 und 255.

Value definiert die Bildhelligkeit. Die Helligkeit wird grundsätzlich in Prozent angegeben, wobei 0% keine Helligkeit, 100% volle Helligkeit repräsentiert. Wahlweise können die Prozentangaben auch auf eine andere Skala abgebildet werden, wie zum Beispiel auf Werte zwischen 0 und 1, oder auf Werte zwischen 0 und 255.

Hierfür werden die Werte für Hue, Saturation und Value pro Pixel wie folgt berechnet:

Value = MAX-Wert(R, G, B)

Saturation = (Value – MIN-Wert(R, G, B) / Value (wenn Value ≠ 0)

Saturation = 0 (wenn Value = 0)

Hue = 60 * (G – B) / (Value – MIN-Wert(R, G, B)) (wenn Value = R)

Hue = 120 + 60 * (B – R) / (Value – MIN-Wert(R, G, B)) (wenn Value = G)

Hue = 240 + 60 * (R – G) / (Value – MIN-Wert(R, G, B)) (wenn Value = B)

Von den Werten *Hue*, *Saturation* und *Value* werden jeweils die Durchschnittswerte von allen Pixel berechnet.

Von den RGB-Werten werden ebenfalls die Durchschnittswerte von allen Pixel berechnet.

Abbildung 36 - HSV Image zeigt das Bild aus *Abbildung 33 - Normalized Image* konvertiert in den HSV-Farbraum.

Abbildung 36 - HSV Image

Die errechneten Werte für Helligkeit, Kontrast, Hue, Saturation und Value werden zur Bestimmung der Grenzwerte (engl. *threshold values*) für verschiedene Bildbearbeitungsalgorithmen herangezogen. Die Beschreibung folgt an entsprechenden Stellen.

5.4.2.5.3. Erkennung des Zahlschein-Datenfeldes

Im nächsten Schritt wird versucht, das Datenfeld des Zahlscheines zu ermitteln. Hierfür muss eine Merkmalserkennung (engl. *feature extraction*) durchgeführt werden, um die Linien/Konturen (engl. *contours*) zu finden, welche das Datenfeld begrenzen. Um die gesuchten Merkmale im Bild zu finden, muss zuerst ein Kantenbild des Bildes erstellt werden. Ein Kantenbild

ist ein Binärbild (engl. *binary image*), welches die Pixel des Bildes aufgrund definierter Merkmale in zwei Gruppen unterteilt. Diese Merkmale können unter anderem durch Farbe, Helligkeit, Kontrast und Sättigung definiert sein.

Die Zuteilung kann mit zwei unterschiedlichen Methoden durchgeführt werden. Die erste Methode ist die Einteilung aller Pixel aufgrund von definierten Grenzwerten der untersuchten Merkmale. Hierbei enthält eine Gruppe die Pixel, welche die definierten Merkmale aufweisen (je nach Definition unter oder über einem definierten Grenzwert liegen bzw. zwischen zwei definierten Grenzwerten liegen), die andere Gruppe die Pixel, welche die Merkmale nicht aufweisen. Bei der zweiten Methode werden die Kanten als Merkmalsänderungen des Pixels gegenüber benachbarter Pixel interpretiert und dementsprechend in einem Binärbild gespeichert. Ein Binärbild ist ein Bild, in dem jedem Pixel einer von zwei möglichen Werten zugeordnet ist. *Abbildung 37 - Flowchart: Line and Rectangle Detection* zeigt die Prozessschritte dieser Arbeit hinsichtlich Linien- und Rechtecks-Erkennung.

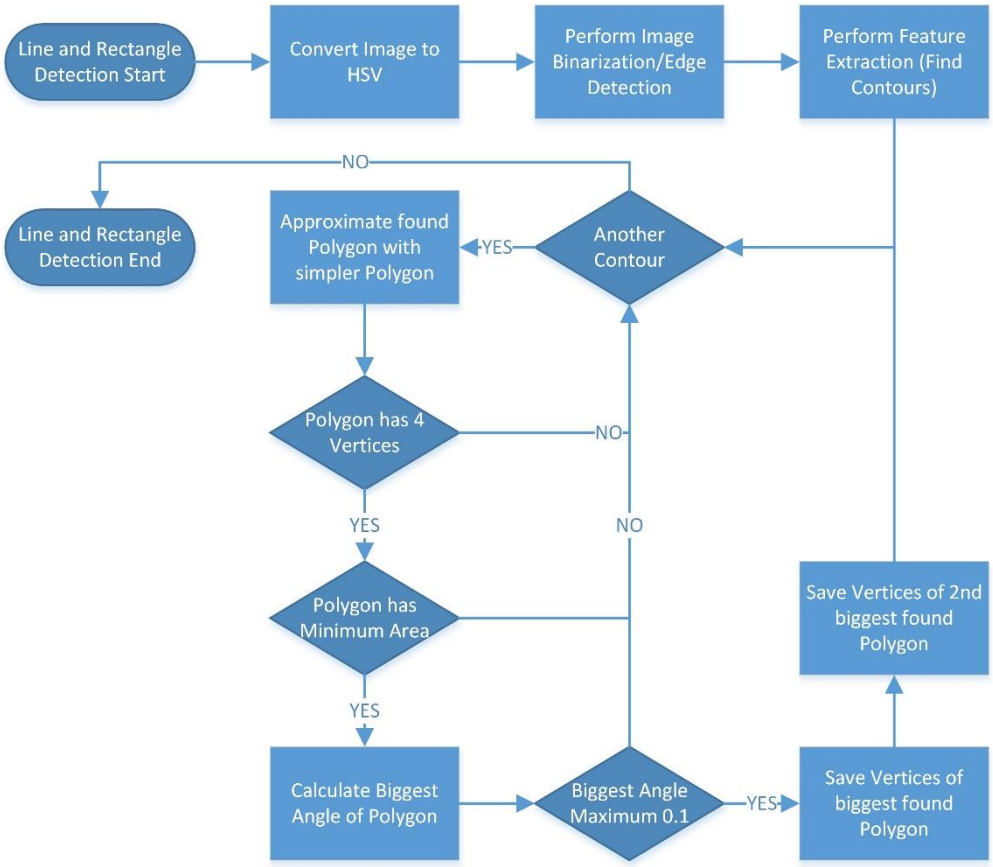

Abbildung 37 - Flowchart: Line and Rectangle Detection

Im Fall dieser Diplomarbeit eignet sich die rote Farbe des Außenrahmens für die Merkmalsfindung. Obwohl die rote Farbe der Zahlscheine grundsätzlich für genau diesen Anwendungsfall als Blindfarbe gewählt wurde, ist dies in dieser Arbeit nicht nutzbar, da die Eigenschaft der Blindfarbe durch das Ablichten mit der Webcam verloren geht – hierfür müsste ein spezieller

Scanner verwendet werden (siehe Kapitel *Applikationen zur OCR-Verarbeitung von Zahlscheinen im Österreichischen Finanzwesen*). Hinzu kommt, dass die Lichtbedingungen bei jedem Einsatz dieses Treibers unterschiedlich sein können. Dies führt dazu, dass die rote Rahmenfarbe der Zahlscheine nicht statisch definiert werden kann. Es muss versucht werden die Rahmenfarbe, oder zumindest einen definierten Farbbereich dieser, von den errechneten Werten für Helligkeit, Kontrast und den durchschnittlichen HSV-Werten abzuleiten. Der gesuchte Farbbereich wird seinerseits auch im HSV-Farbraum definiert, da sich dieser besser für die Trennung von Farben eignet als der RGB-Farbraum.

Da die Farbe „Rot“ im HSV-Farbraum jeweils am unteren und am oberen Ende der *Hue* Skala definiert ist, müssen zwei Farbbereiche definiert werden.

Um diese zwei HSV-Bereiche für die Rahmenfindung zu definieren wurde eine Versuchsreihe bestehend aus 200 Tests durchgeführt. Hierfür wurden Zahlscheine unter verschiedenen Bedingungen hinsichtlich Belichtung, Helligkeit, Kontrast und Sättigung (Farbintensität) abgelichtet und verarbeitet. Die unterschiedlichen Belichtungsszenarien wurden durch Variation des Umgebungslichtes erreicht. Hierfür wurden folgende *fuzzy* Bereiche definiert:

- Direkte Sonneneinstrahlung auf den zu erfassenden Bildbereich
- Indirekte Sonneneinstrahlung: dies bedeutet, dass die Sonne die Tageshelligkeit maßgeblich erhellt hat, der zu erfassende Bildbereich allerdings nicht der direkten Sonneneinstrahlung ausgesetzt war
- Tageslicht bei Bewölkung
- Abenddämmerung ohne Raumbeleuchtung
- Abenddämmerung mit Raumbeleuchtung
- Nacht ohne Raumbeleuchtung
- Nacht mit Raumbeleuchtung
- Nacht mit expliziter Ausleuchtung des zu erfassenden Bildbereich; hierfür wurden 4 unterschiedliche Beleuchtungsstufen eingesetzt

Die genauen Werte der Versuchsreihe hinsichtlich Lichtstärke und Lichtstrom sind nicht bekannt.

Die unterschiedlichen Bedingungen hinsichtlich Helligkeit, Kontrast und Sättigung wurden durch Änderungen der Kamerakonfiguration erreicht.

Die Vorgehensweise der Tests war wie folgt:

1. *Fuzzy* Bereich für das Lichtverhältnis erheben und dokumentieren
2. Kamerakonfiguration für Helligkeit, Kontrast und Sättigung durchführen (der Wertebereich dieser drei Größen war bei der verwendeten Kamera jeweils 0-255) und dokumentieren. Es wurden jeweils die Werte 50, 128 und 220 verwendet, somit ergeben sich neun Konstellationen der Werte. Die Verwendung der Werte 50, 128 und 220 wurden manuell auf Grund der visuellen Darstellung des Kamerastreams gewählt
3. Erfassung eines Zahlscheines mit der Kamera, unter den definierten Einstellungen
4. Dokumentation der errechneten Bildattribute Helligkeit, Kontrast, durchschnittlicher *Hue*, durchschnittlicher *Value* und durchschnittliche *Saturation* für das erfasste Bild

5. Iterative Durchführung der Punkte 6. Und 7.
6. Versuchsweise Festlegung der HSV-Threshold-Grenzen für die Binarisierung und anschließende Verarbeitung
7. Dokumentation der Ergebnisse aus Punkt 6. für die 3 Szenarien:
    a. Untere Threshold-Werte für HSV bei denen die Verarbeitung „noch“ brauchbare Ergebnisse liefert
    b. Obere Threshold-Werte für HSV bei denen die Verarbeitung „noch“ brauchbare Ergebnisse liefert
    c. Optimale Threshold-Werte für die Verarbeitung (nicht zwingend die Mittelwerte von unteren und oberen Grenzen aus Punkt a. und b.)

Da aus den Tests kein direkter Zusammenhang zwischen den Bildwerten Helligkeit, Kontrast und durchschnittliche HSV Werte des Bildes und den HSV-Werten des Thresholdings für die optimalen Ergebnisse abgeleitet werden konnte, wurden folgende Regeln zur Festlegung der HSV-Grenzwerte festgelegt:

Bei der Rahmenfindung anhand der roten Farbe sind die gesuchten Wertebereiche für *Hue* 0-10 und 170-180, wenn die Gesamtskala 0-180 beträgt. Diese entsprechen der Definition für die Farbe „Rot“. Die Threshold-Werte für *Hue* können somit grundsätzlich konstant gewählt werden. Im Fall dieser Arbeit werden die Bereiche jedoch auf 0-15 und 165-180 ausgedehnt, wenn der durchschnittliche *Hue* Wert des Bildes unter 65, oder die Bildhelligkeit unter 170 beträgt. Grund hierfür ist die Verschiebung der „Rot“-Werte bei Bildern mit relativ niedriger Bildhelligkeit.

Da die Farbe selbst an sich unabhängig von der Helligkeit ist und die Helligkeitswerte verschiedener Bereiche eines Bildes nicht konstant sein müssen, wird für *Value* die gesamte Skala genutzt. Im Fall dieser Implementierung 0-255. Ein abgelichteter Zahlschein könnte zum Beispiel eine Kante durch einen Knick enthalten. Somit wäre der durch die Kante von der Kamera abgewandte Bereich des Zahlscheines, im Bild dunkler, als der der Kamera zugewandte Teil des Zahlscheines.

Somit bleibt die ausschlaggebende Größe des Thresholding die *Saturation*. Die Skala für *Saturation* reicht in dieser Implementierung von 0-255. Da die reine rote Farbe eine *Saturation* von 255 hat, ist der obere Grenzwert des gesuchten *Saturation*-Bereichs 255.

Bei einem Bildkontrast von über 700 und einer Bildhelligkeit von über 170, wird der untere Grenzwert für *Saturation* wird wie folgt gewählt:

Bei einer durchschnittlichen Sättigung des Bildes von unter 15, wird der untere Grenzwert mit 100 definiert.

Bei einer durchschnittlichen Sättigung des Bildes zwischen 15 und 30, wird der untere Grenzwert mit 135 definiert.

Bei einer durchschnittlichen Sättigung des Bildes zwischen 30 und 45, wird der untere Grenzwert mit 150 definiert.

Bei einer durchschnittlichen Sättigung des Bildes zwischen 45 und 60, wird der untere Grenzwert mit 165 definiert.

Bei einer durchschnittlichen Sättigung des Bildes von über 60, wird der untere Grenzwert mit 180 definiert.

Bei einem Bildkontrast von unter 700 oder einer Bildhelligkeit von unter 170, wird der untere Grenzwert für *Saturation* mit 80 definiert. Zudem werden in diesen Fällen die Wertebereiche für *Hue* auf 0-20 und 160-180 ausgedehnt.

Mit den definierten Werten wird versucht, das Eingabebild zu filtern, um ein Kantenbild von möglichst nur Bildbestandteilen der Rahmenfarbe zu erhalten. Hierfür werden die HSV-Werte der Bildpixel mit den definierten Bereichen der HSV-Werte verglichen, um passende, sowie unpassende Pixel, jeweils einem der beiden Binärwerte zuzuordnen.

Wenn die Rahmenfindung mit den definierten Grenzwerten nicht erfolgreich durchgeführt werden konnte, wird ein neuer Grenzwertbereich für *Saturation* festgelegt und die Rahmenfindung erneut durchgeführt. Hierbei wird der *Saturation*-Bereich um 30 vergrößert, der untere Grenzwert von *Saturation* somit um 30 reduziert. Ist die Erkennung ein zweites Mal nicht erfolgreich, wird die Verarbeitung abgebrochen und ein Fehler wird zurückgegeben.

5.4.2.5.3.1. Binarisierung des Eingabebildes

Die zwei definierten HSV-Bereiche ergeben die zwei Binärbilder *Abbildung 38 - Binary Image of lower Hue Scope* und *Abbildung 39 - Binary Image of upper Hue Scope. Abbildung 40 - Binary Image of both Hue Scopes* zeigt die beiden Binärbilder übereinander gelegt (addiert).

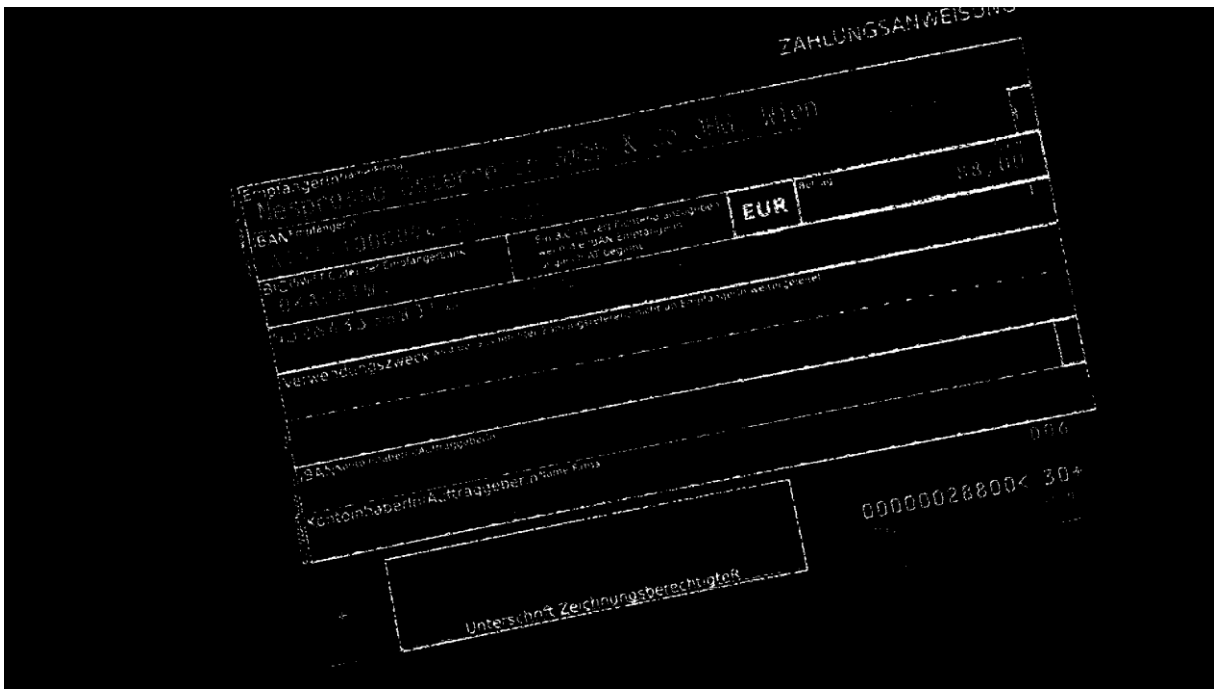

Abbildung 38 - Binary Image of lower Hue Scope

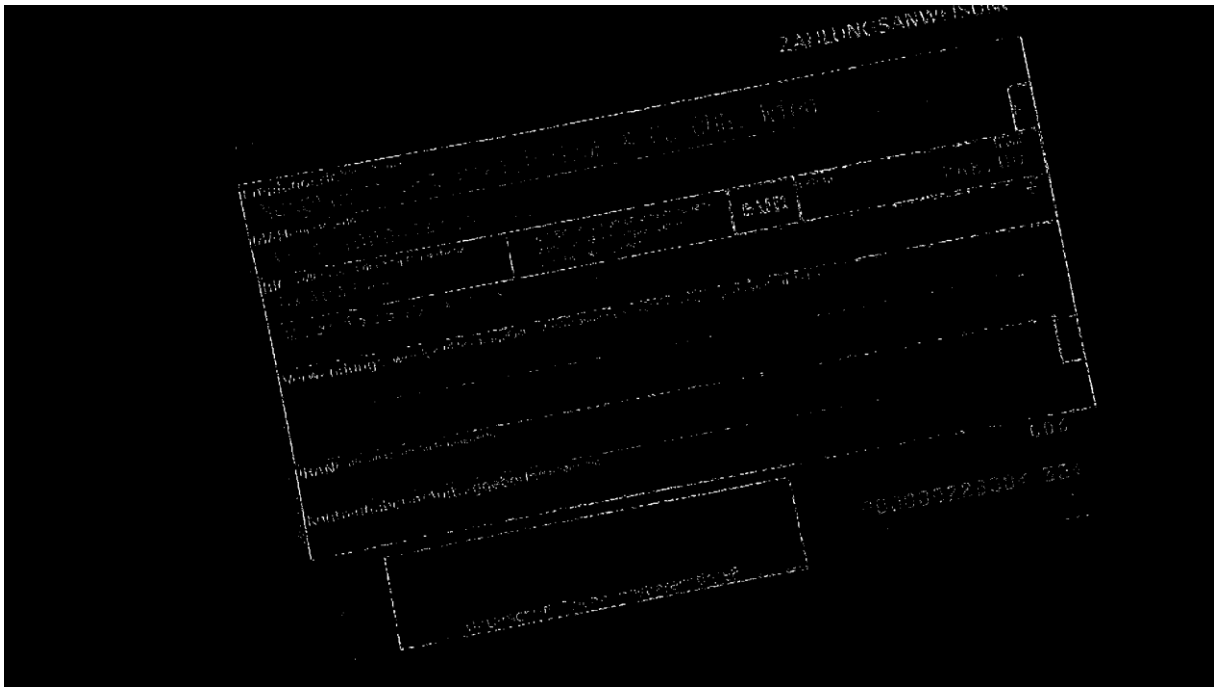

Abbildung 39 - Binary Image of upper Hue Scope

Abbildung 40 - Binary Image of both Hue Scopes

Wenn die HSV-Bereiche zum Erstellen der Binärbilder zu groß gewählt werden, haben die Binärbilder weniger Aussagewert, da diese dann weit mehr Bildelemente als den roten Rahmen des Datenfeldes des Zahlscheines enthalten und somit das Merkmal ungenauer definieren. Werden die HSV-Bereiche zu klein gewählt, werden unter Umständen weniger Bildelemente gefunden als für die Detektion des Rahmens notwendig sind. Werden die HSV-Bereiche des Filters schlecht gewählt, oder weist der Rahmen am Bild nicht genügend unterscheidbare Charakteristika gegenüber seiner Umgebung auf, kann der Rahmen nicht detektiert werden. In weniger extremen Fällen kann der Rahmen zum Beispiel Unterbrechungen aufweisen, wenn gewisse Bildbereiche des Rahmens nicht in die definierten HSV-Bereiche der Filter fallen. Um die Liniendetektion in diesen Fällen trotzdem zu ermöglichen, müssen diese Lücken geschlossen werden.

Deswegen wird das Binärbild im Anschluss an die Binarisierung (engl. *binarization*) erweitert (engl. *dilated*). Diese Operation ist den morphologischen Transformationen (engl. *morphological transformations*) zugeordnet und wird folgend im Zuge der morphologischen Öffnung (engl. *morphological opening*) erläutert.

Abbildung 41 - Both Image Scopes after morphological Dilation zeigt das Bild aus *Abbildung 40 - Binary Image of both Hue Scopes* nach der morphologischen Erweiterung (engl. *morphological dilation*).

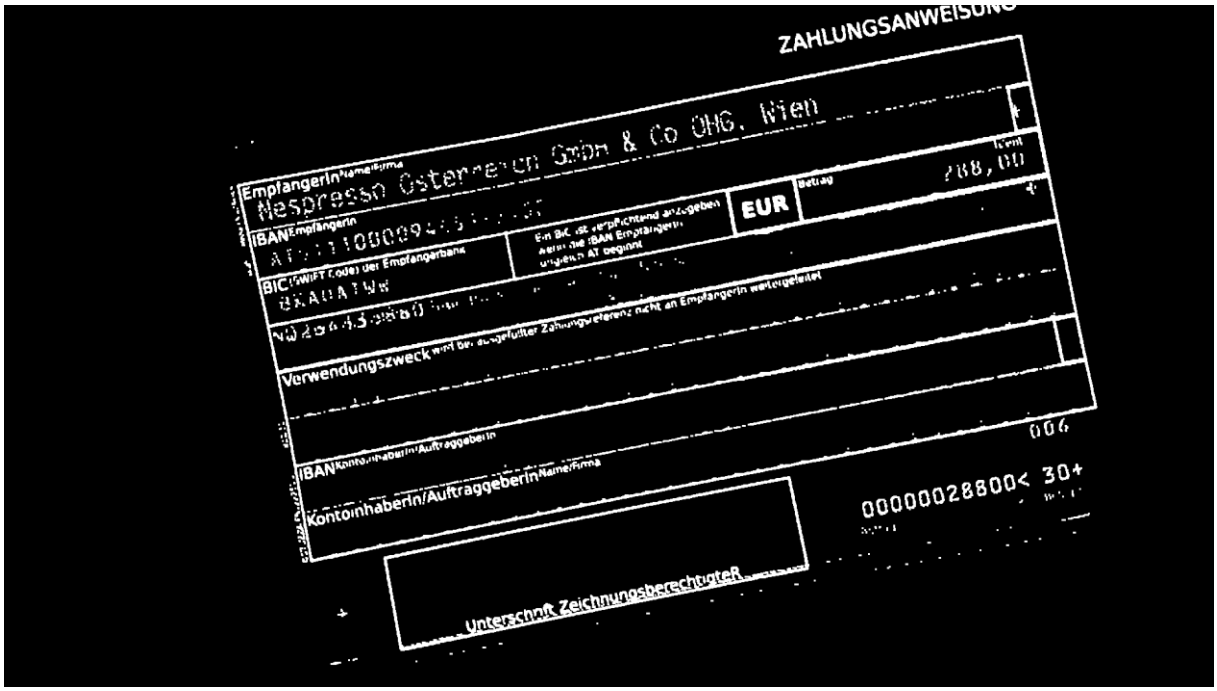

Abbildung 41 - Both Image Scopes after morphological Dilation

Eine alternative Methode zur Kantendetektion (Erstellung des Kantenbildes) aufgrund von Grenzwerten wäre Thresholding aufgrund der Helligkeit (engl. *brightness thresholding*).

Thresholding is a technique used for segmentation, which separates an image into two meaningful regions: foreground and background, through a selected threshold value T. If the image

is a grey image, T is a positive real number in the range of [0,….,K], Where, K >= 0. So, thresholding may be viewed as an operation that involves tests against a value T [80].

Thresholding aufgrund der Helligkeit ist die einfachste Methode. Hierbei wird das Binärbild erstellt, indem allen Pixel unter einem bestimmten Helligkeits-Grenzwert (engl. *threshold*) ein Wert zugewiesen wird, allen anderen Pixel der andere Wert. *Abbildung 42 - Image Thresholding with Thresholds 50 (left up), 100, 150, 200* zeigt das Eingabebild aus *Abbildung 33 - Normalized Image,* bearbeitet mit den Thresholds 50 (links oben), 100 (rechts oben), 150 (links unten) und 200 (rechts unten). Für diese Arbeit ist diese einfache Methode ungeeignet, da neben der Bildhelligkeit auch die rote Farbe des Rahmens genutzt werden kann. Würde man den Threshold so wählen, dass der rote Rahmen erhalten bleiben würde, blieben automatisch auch alle dunkleren Zeichen wie die schwarzbedruckte Schrift erhalten.

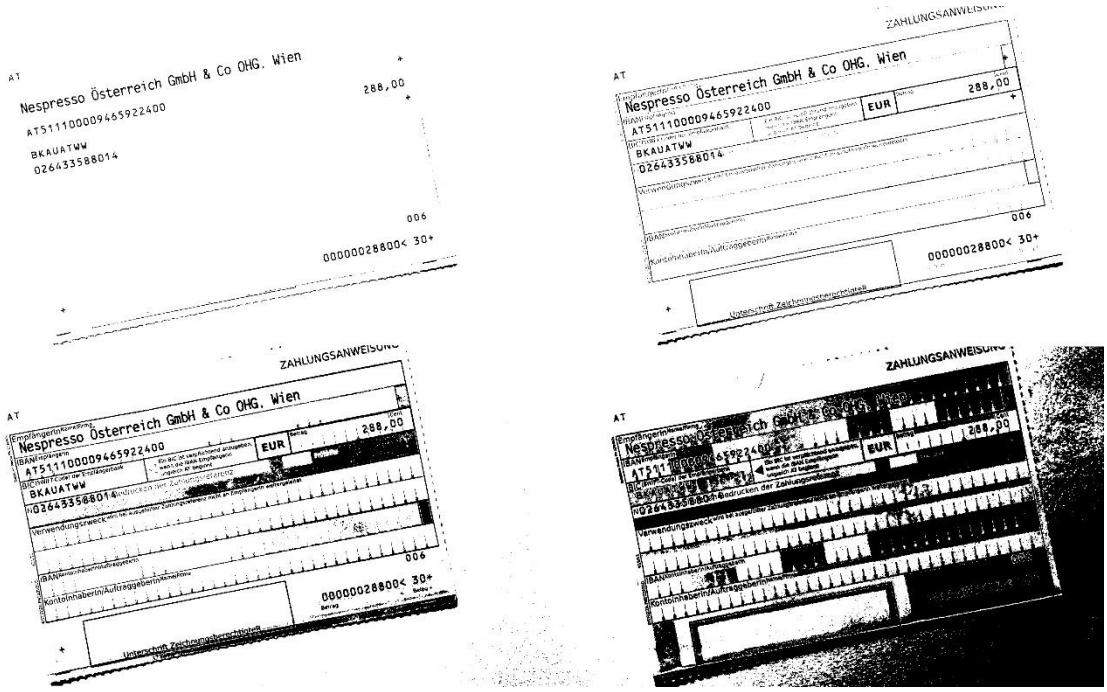

Abbildung 42 - Image Thresholding with Thresholds 50 (left up), 100, 150, 200

Neben diesen einfachen Methoden der Kantendetektion durch den Einsatz von Grenzwerten bezüglich Farbe, Helligkeit oder Sättigung wurden bereits viele Methoden der zweiten Gruppe (Kantendefinition aufgrund von lokalen Merkmalsänderungen) implementiert.

Die meisten dieser Methoden basieren auf den klassischen Operatoren und Algorithmen *Roberts*, *Sobel*, *Scharr*, *Prewitt*, *Frei-Chen*, *Laplacian* und *Canny* [81].

Der Sobel-Operator ist ein diskreter Differenzial-Operator. Er berechnet eine Annäherung des Gradienten einer Bildintensitätsfunktion. Der Operator kombiniert Gauss‘sche Glättung und Differentialrechnung. Glätten ist die mathematische Berechnung von Kurvenannäherungen mit geringerer Krümmung. Das Ziel beim Glätten ist die Anwendung von Näherungspolynomen,

welche ein optimales Verhältnis von Krümmungsverringerung zu Änderungen der Ausgangskurve ergeben. Während die Krümmung möglichst verringert werden soll, soll die Kurve selbst möglichst unverändert bleiben. Im ersten Schritt berechnet der Sobel-Operator zwei Derivate:

Horizontale Änderungen: Berechnung durch Faltung des Eingabebildes I mit einem ungeraden Kernel. Die Standardglättung der Größe 3 sieht wie folgt aus:

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * I$$

Abbildung 43 - Standard horizontal Sobel Kernel Size 3 [82]

Vertikale Änderungen: Berechnung durch Faltung des Eingabebildes I mit einem ungeraden Kernel. Die Standardglättung der Größe 3 sieht wie folgt aus:

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * I$$

Abbildung 44 - Standard vertical Sobel Kernel Size 3 [82]

Anschließend wird für jeden Punkt des Bildes eine Annäherung des Gradienten in diesem Punkt berechnet. Dies kann mit einer der folgenden beiden Formeln aus *Abbildung 45 - Calculation of Gradient during Sobel* berechnet werden.

$$G = \sqrt{G_x^2 + G_y^2} \qquad G = |G_x| + |G_y|$$

Abbildung 45 - Calculation of Gradient during Sobel [82]

Abbildung 46 - Sobel Operator with Kernel Sizes 1 (left up), 3, 5, 7 zeigt das Eingabebild aus *Abbildung 33 - Normalized Image* nach Anwendung des Sobel-Operators mit den Kernelgrößen 1 (links oben), 3 (rechts oben), 5 (links unten) und 7 (rechts unten). Gut erkennbar ist, dass bei zunehmender Größe des eingesetzten Kernels zunehmend viele Störpixel als Kanten wahrgenommen werden. Der Sobel-Operator könnte für diese Arbeit verwendet werden, liefert jedoch deutlich ungenauere Ergebnisse als das Thresholding der roten Farbe und deren Sättigung. Zudem kann der Sobel-Operator nur auf einen Farbkanal angewandt werden. Aufgrund der RGB-Zusammensetzung der mit Webcam abgelichteten Bilder müsste der Sobel-Operator auf alle drei Farbkanäle angewandt werden und die Ergebnisse unter Umständen separat weiter verarbeitet werden.

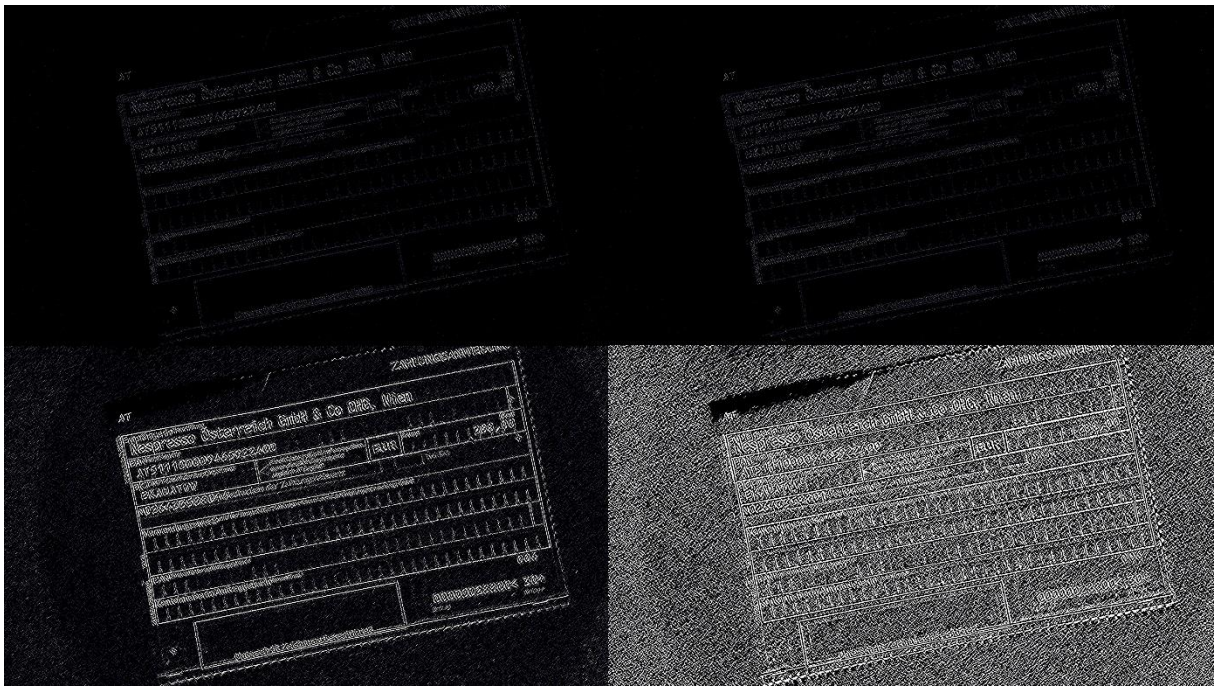

Abbildung 46 - Sobel Operator with Kernel Sizes 1 (left up), 3, 5, 7

Ein weiterer Algorithmus zur Kantendetektion ist der Canny Algorithmus. Entwickelt von John F. Canny im Jahr 1986, auch *optimal detector* genannt, verfolgt der Canny Algorithmus die Erfüllung folgender Bedingungen [83]:

- Niedrige Fehlerrate: Ausschließliche Erkennung von tatsächlichen Kanten
- Gute Lokalisierung: Die Distanz zwischen erkannten Kantenpixel und tatsächlichen Kantenpixel soll minimiert werden
- Minimale Resonanz: Nur eine Antwort des Detektors pro Kante

Der Canny Algorithmus besteht grundsätzlich aus vier Schritten [84]:

1. Reduktion von Bildstörungen: Dies erfolgt durch Glättung unter Verwendung eines Gauss'schen Kernel der Größe 5 (siehe *Abbildung 47 - Gaussian Filter Kernel with Size 5*).
2. Gradient des Bildes berechnen (siehe Sobel-Operator)
3. Non-maximum supression: Die gefundenen Kanten werden auf die Stärke von 1 Pixel reduziert, indem entlang der Kanten nur das stärkste Pixel behalten, alle anderen verworfen werden.
4. Hysterese: Mit dem Hysterese Verfahren wird versucht, in den durch Berechnung gefundenen Kanten die tatsächlich existenten Kanten zu filtern. Dies geschieht unter Verwendung von einem oberen und einem unteren Grenzwerte. Folgende Bedingungen werden für alle Kantenpixel überprüft (siehe *Abbildung 48 - Canny Hysteresis*):
    a. Ist der Gradient des Pixels größer als der obere Grenzwert, wird das Pixel behalten.

b. Ist der Gradient des Pixels kleiner als der untere Grenzwert, wird das Pixel verworfen.

c. Ist der Gradient des Pixels zwischen den Grenzwerten und ist das Pixel mit einem Pixel aus Punkt a. benachbart, wird das Pixel behalten (John Canny empfiehlt ein Verhältnis von oberem zu unterem Grenzwert zwischen 2 zu 1 und 3 zu 1).

$$K = \frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix}$$

Abbildung 47 - Gaussian Filter Kernel with Size 5

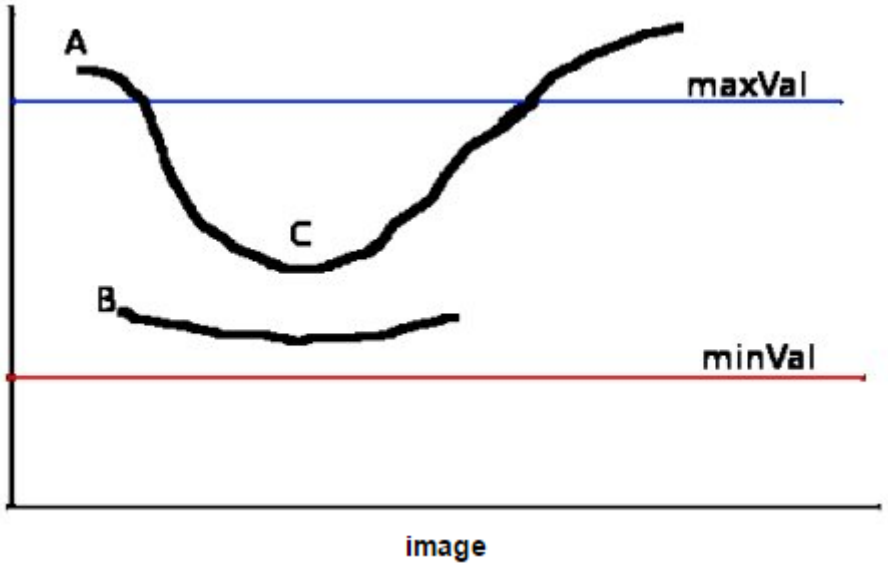

Abbildung 48 - Canny Hysteresis [85]

Abbildung 49 - Canny Edge Detector with Thresholds 80 & 160 & Kernel 3 zeigt das Eingabebild aus *Abbildung 33 - Normalized Image* nach Anwendung des Canny Algorithmus mit der Kernelgröße 3, dem oberen Grenzwert 160 und dem unteren Grenzwert 80. Genau wie der Sobel-Operator könnte auch der Canny Algorithmus für diese Arbeit verwendet werden. Doch genau wie der Sobel-Operator liefert Canny schlechtere Ergebnisse als die Untersuchung des Bildes nach der roten Rahmenfarbe. Weiters müsste auch bei Verwendung von Canny das Bild in allen Farbkanälen separat verarbeitet werden. Außerdem erfordert Canny, abhängig vom Eingabebild, die Festlegung der Grenzwerte, um verwertbare Ergebnisse zu erhalten.

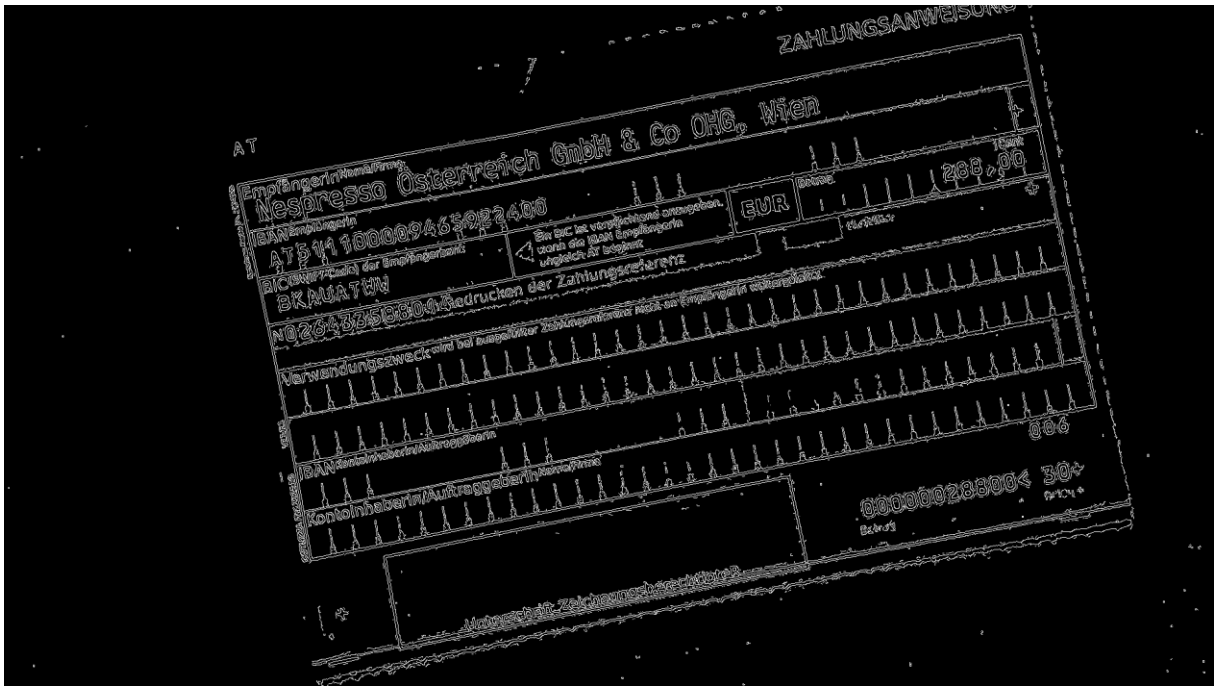

Abbildung 49 - Canny Edge Detector with Thresholds 80 & 160 & Kernel 3

5.4.2.5.3.2. Rechtecks-Erkennung im Binärbild

Das erstellte Kantenbild kann nun zu Linien-, Vierecks- oder sonstiger Merkmalserkennung verwendet werden. Bei großen abzubildenden Dokumenten bzw. bei Dokumenten mit unebenen, kurvigen Bereichen macht es unter Umständen Sinn, das Bild durch Rektifizierung vor zu verarbeiten, um Kurven auf Linien bzw. Vierecke abzubilden.

In „Goal-Oriented Rectification of Camera-Based Document Images" setzen die Autoren eine "coarse-to-fine strategy" zur Rektifizierung der Bilder ein. Im ersten Schritt wird eine grobe Rektifizierung durchgeführt, um kurvige Bereiche des Bildes auf ein Rechteck abzubilden. Der erste Schritt beruht auf der Ausrichtung des Textes im Bild. Im zweiten Schritt wird der Bildbereich auf Wortebene normalisiert, um lokale Verzerrungen zu beseitigen. Die Rektifizierung wird mit einem definierten Transformationsmodell durchgeführt. Alternativen wären Interpolationsverfahren und parametrische Verfahren. Experimente zeigten Steigerungen der Genauigkeit von OCR Engines bis zu 40% auf Zeichenebene und auf Wortebene gegenüber dem Einsatz der gleichen OCR Engines ohne eine vorrangegangene Rektifizierung der Bildverzerrungen [86].

In dieser Arbeit ist die Rektifizierung nicht notwendig, da die geringe Größe der Zahlscheine auch bei leichten Biegungen nur zu kleinen Abweichungen führt und somit die OCR Verarbeitung nicht maßgeblich beeinflusst. Sollten die Zahlscheine sehr große Verformungen aufweisen, können diese auch durch Rektifizierung nicht mehr korrekt ausgeglichen werden. Somit

wäre eine Rektifizierung nur in sehr wenigen Fällen sinnvoll und in allen anderen Fällen würde der Rektifizierungsprozess unnötigerweise Ressourcen verbrauchen.

Das Lokalisieren und die Bestimmung der Orientierung von Linien in Bildern ist ein wichtiger Prozess in den Feldern CV (Computer Vision), Bildverarbeitung und auch OCR, speziell auf Formulare angewandt. Für binarisierte Bilder ist die klassische Methode die Hough Transformation, welche als spezielle Radon Transformation gilt [87].

Linienerkennung in Rastergrafiken wird grundsätzlich unter Einsatz der Hough Transformation durchgeführt. Hough parametrisiert Linien durch zwei Parameter, dem „slope“ und „Y-Achsen-Schnittpunkt“. Der Hough Algorithmus benutzt ein zweidimensionales Array. Dieses wird Akkumulator genannt und dazu verwendet, Linien zu identifizieren. Linien werden durch $r = x*\cos\theta + y*\sin\theta$ definiert. Eine weitere Parameterisierung ist die θ-ρ Parameterisierung von Duda und Hart, welche die Hough Transformation verbessert. Linienerkennung wird auch bei Kamerakalibrierung, Erkennung von geometrischen Objekten, bar codes, QR Codes und Augmented Reality Markern eingesetzt [88].

Abbildung 50 - Result of Hough Transformation on Binary Edge Image mapped on orignial Input Image zeigt das Ergebnis der Hough Transformation, angewandt auf das Kantenbild aus *Abbildung 41 - Both Image Scopes after morphological Dilation*, abgebildet auf das Eingabebild aus *Abbildung 31 - Input Image captured from Cam*. Die mit *Hough* gefundenen Linien im Kantenbild sind im ursprünglichen Eingabebild blau eingezeichnet.

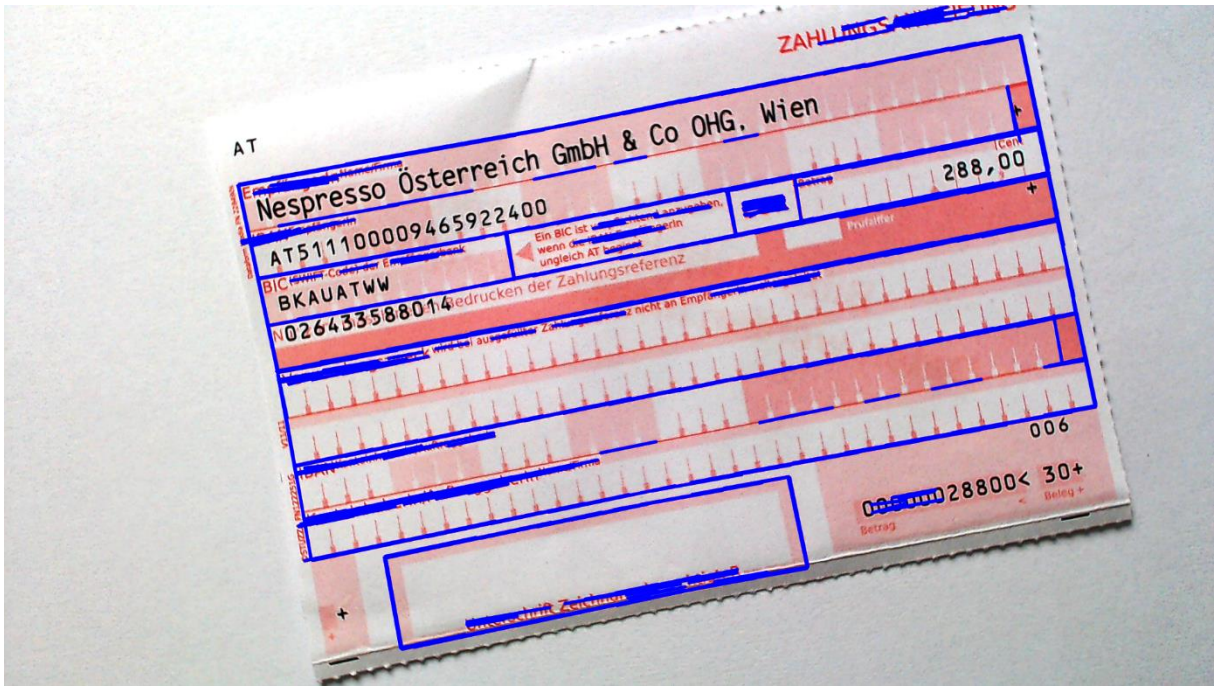

Abbildung 50 - Result of Hough Transformation on Binary Edge Image mapped on orignial Input Image

Hough Transformation könnte in dieser Arbeit zur Liniendetektion angewandt werden. Da in dieser Arbeit allerdings Vierecke und nicht nur Linien gefunden werden müssen, müssten die

Ergebnisse der Hough Transformation weiter verarbeitet werden, um in allen gefundenen Linien, mögliche Vierecke zu detektieren. Aus diesem Grund wird die Verwendung eines Algorithmus zur Erkennung von geschlossenen Graphen bevorzugt.

Die Methoden zur Rechteckserkennung können in zwei Kategorien unterteilt werden. Die erste Kategorie enthält Methoden basierend auf der Hough Transformation oder basierend auf Methoden welche wiederum auf der Hough Transformation basieren. Die zweite Kategorie basiert auf einer linearen Kombination von Methoden. Zhu und Qingzhi haben einen Algorithmus für „closed graph disjoint chain code tracking“ entwickelt. Im Prinzip vergleicht man das aktuelle Pixel mit den acht benachbarten Pixel und nimmt das benachbarte Pixel gegebenenfalls in die „Tracking List“ auf. Der Algorithmus wird mit den neu aufgenommenen Pixel fortgeführt. Erreicht man ein bereits besuchtes Pixel, handelt es sich offenbar um einen geschlossenen Graphen. Anhand der beteiligten Pixel kann evaluiert werden, ob es sich bei dem Graphen um ein Rechteck handelt [89].

Satoshi Suzuki und Keiichi Abe haben ebenfalls einen Algorithmus entwickelt, um Konturen in digitalisierten Bildern zu finden. Dieser ist im Paper *„Topological Structural Analysis of Digitized Binary Images by Border Following“,* dokumentiert [90]. Eigentlich sind es zwei Algorithmen. Beide verarbeiten Binärbilder. Der erste findet alle Konturen eines Bildes, sowohl innere wie auch äußere. Bei äußeren Konturen handelt es sich um den äußeren Rahmen von Objekten des Binärbildes. Innere Konturen existieren nur bei geschlossenen Objekten. Enthält das Binärbild zum Beispiel einen Ring, werden 2 Konturen gefunden, nämlich die äußere, sowie die innere Begrenzung des Ringes. Jeder Pixel einer gefundenen Kontur grenzt somit an zwei Seiten an Pixel der Kontur, an einer Seite an einen Pixel des gefundenen Objektes im Binärbild und an einer Seite an den des gefundenen Objekts abgewandten Pixel. Der zweite Algorithmus ist eine Abwandlung des ersten und findet nur die äußersten Konturen des Binärbildes. Hierbei werden nicht nur die inneren Konturen von geschlossenen Objekten verworfen, sondern sämtliche gefundenen Konturen innerhalb eines geschlossenen Objektes. Enthält das Bild zum Beispiel zwei Ringe, wobei einer vom anderen ohne Schnittpunkt umschlossen ist, findet der Algorithmus nur die äußere Kontur des äußeren Ringes.

Für diese Implementierung ist die Verwendung des zweiten Algorithmus von Suzuki und Abe naheliegend, da der Anwendungsfall die Auffindung des äußersten Rechteckes von Zahlscheinen vorsieht. Die Vorgehensweise des Algorithmus ist hierbei wie folgt:

Um mögliche Startpixel zu finden wird das gesamte Raster des binären Bildes nach Pixel mit dem Wert „1“ gescannt, die ein benachbartes Pixel mit dem Wert „0“ aufweisen. Startpixel können zudem nur Pixel sein die nicht innerhalb von bereits gefundenen Konturen liegen. Das Startpixel wird samt seiner benachbarten Pixel mit dem Wert „0“ in die Liste der besuchten Pixel aufgenommen. Ausgehend vom Startpixel werden dann alle Nachbarpixel gelesen. Beträgt der Wert des gelesenen Pixels ebenfalls „1“ und hat dieser ein Nachbarpixel mit dem Wert „0“, das gleichzeitig ein Nachbarpixel eines Nachbarpixels des zuvor gelesenen Pixels mit dem Wert „1“ ist, so wird dieses, inklusive seiner Nachbarpixel mit dem Wert „0“, in die Liste der besuchten Pixel aufgenommen. Bei jedem Einfügen in die Liste wird überprüft, ob eines der benachbarten Pixel mit einem der bereits gespeicherten Pixel des Wertes „0“, seinerseits benachbart ist. Wenn ja wurde eine Kontur gefunden. Der Algorithmus wird ausgehend von der Liste der besuchten Pixel weiter geführt. Werden keine neuen Nachbarpixel mit dem Wert „1“ gefunden wird ein neues Startpixel gesucht. Wird kein neues Startpixel gefunden ist der Algorithmus zu Ende [90].

Abbildung 51 - Contour Detection by Border Following (Satoshi Suzuki, Keiichi Abe) zeigt das Ergebnis von *border following* nach Satoshi Suzuki und Keiichi Abe, angewandt auf das Kantenbild aus *Abbildung 41 - Both Image Scopes after morphological Dilation*. Die oberen zwei Bilder aus *Abbildung 51 - Contour Detection by Border Following (Satoshi Suzuki, Keiichi Abe)* zeigen das Ergebnis des Algorithmus von Suzuki und Abe, angewandt mit der Option, alle Konturen zu finden Die unteren beiden Bilder zeigen das Ergebnis des Algorithmus mit der Option, nur äußere Konturen zu finden. Innere Konturen sind entweder Konturen, welche eine Verbindung zu einer äußeren Kontur haben, aber nicht zu den äußeren Punkten der Kontur gehören, oder Konturen, die keine Verbindung zu einer äußeren Kontur haben, allerdings von einer äußeren Kontur umschlossen sind. Die rechten Bilder aus *Abbildung 51 - Contour Detection by Border Following (Satoshi Suzuki, Keiichi Abe)* zeigen die gefundenen Konturen abgebildet auf das ursprüngliche Eingabebild aus *Abbildung 31 - Input Image captured from Cam.* Die inneren Konturen mit Berührung zu äußeren Konturen sind im rechten oberen Bild der *Abbildung 51 - Contour Detection by Border Following (Satoshi Suzuki, Keiichi Abe)* grün eingezeichnet, die restlichen Konturen blau.

Um Speicher bei der Speicherung der Konturen zu sparen, kann der Algorithmus unter Verwendung von einfacher Approximation angewandt werden. Hierbei werden sämtliche gefundene horizontale, vertikale und diagonale Segmente nur durch ihre Endpunkte identifiziert und gespeichert. Verwendet man keine Approximation, werden alle zur Kontur gehörenden Punkte gespeichert. Das gefundene Viereck des Datenfeldes des Zahlscheines besteht in diesem Beispiel bei einer Auflösung von zwei Megapixel aus mehreren Tausend Punkten. Im optimalsten Fall könnte es auf die vier Eckpunkte komprimiert werden. Bei schlechtem Kontrast an sich oder schlechten Ergebnissen bei der Erstellung des Kantenbildes, zum Beispiel als Folge von schlechtem Kontrast, kann das Viereck aber zumindest auf wenige Zig Punkte reduziert werden.

Abbildung 51 - Contour Detection by Border Following (Satoshi Suzuki, Keiichi Abe)

Das linke untere Bild der *Abbildung 51 - Contour Detection by Border Following (Satoshi Suzuki, Keiichi Abe)* zeigt eine Abbildung der Informationen, wie sie in diesem Treiber weiter verarbeitet werden. Die im Bild angezeigten Konturen sind in Form von Sequenzen aus den zur Kontur gehörenden Punkten im Speicher.

Grundsätzlich wird zur OCR Verarbeitung nur das Rechteck benötigt, welches die Daten der Überweisung enthält. Da der Zahlschein allerdings von (0° bis 360] rotiert sein kann, wird ein weiterer Bezugspunkt benötigt, von welchem auf die Rotation des Zahlscheines geschlossen werden kann. Da das Unterschriftsfeld, laut Spezifikation der Zahlscheine durch die S€PA, auf jedem Zahlschein an der gleichen Position enthalten sein muss, wurde dies als Referenzpunkt für die Rotation des Zahlscheines gewählt.

Ziel ist es somit, aus allen gefundenen äußeren Konturen die zwei Vierecke für das Daten- und das Unterschriftsfeld zu finden und die jeweils vier Eckpunkte zu speichern, falls die zwei Felder vorhanden sind und gefunden wurden. Um dies zu überprüfen, müssen alle gefundenen Konturen iteriert werden.

Da es sich bei den gesuchten Feldern um Vierecke handelt, diese allerdings durch Ungenauigkeiten betreffend Wölbungen des Zahlscheinpapiers, Kamerawinkel bei der Aufnahme und Konturendetektion nicht exakt aus vier Eckpunkten bestehen, müssen die gefundenen Konturen an ein Viereck approximiert werden. Dies erfolgt unter Verwendung des *Ramer Douglas Peuker algorithm*.

Der *Ramer Douglas Peuker algorithm* approximiert eine Linie, definiert durch n Punkte, durch eine Linie bestehend aus weniger Punkten, wobei jene Punkte eliminiert werden, die bei Beibehaltung die Approximation am wenigsten verändern würden. Das Maß dieser Veränderung wird durch die Toleranz ε definiert. Beginnend bei den beiden Endpunkten, teilt der Algorithmus die Linie in zwei Segmente, wobei jener Punkt als Teiler gewählt wird, der von der direkten Verbindung der Endpunkte am weitesten entfernt liegt. Liegt diese Entfernung innerhalb der Toleranz ε, ist der Algorithmus zu Ende, sonst werden die vorhandenen Liniensegmente rekursiv ihrerseits in Teilsegmente zerlegt, bis kein Punkt existiert, welcher ausserhalb der Toleranz ε liegt [91].

Diese Lösung verwendet den Ramer-Dougles-Peuker Algorithmus mit einer Genauigkeit von 0,02. Dies entspricht einer Abweichung von 2% zwischen dem Umfang der ursprünglichen Kontur und dem Umfang der approximierten Kontur. Beträgt die Abweichung mehr als 2%, ist die geometrische Verformung des Zahlscheines, zumindest aber die Abweichung der Ergebnisse der Konturendetektion, so groß, dass die OCR Engine keine für diese Lösung brauchbaren Ergebnisse liefert. Der Wert von 2% wurde durch das Durchführen von *trial and error tests* ermittelt.

Nach der Approximation der Konturen werden alle approximierten Konturen verworfen, welche nicht exakt aus vier Punkten bestehen, also all jene, die keine Vierecke sind. Zudem werden alle Konturen verworfen, deren Fläche kleiner als 10000 Pixel ist. Dies ist deswegen notwendig, da durch die vielen Störpixel in Webcam-Bildern viele Konturen gefunden werden, welche nur aus wenigen Pixel bestehen. Würden diese Konturen bei der weiteren Verarbeitung berücksichtigt werden hätte dies negative Auswirkungen auf die Performance. Die minimal benötigte Anzahl an Pixel für eine viereckige Kontur entspricht acht.

Der Wert 10000 wurde wie folgt ermittelt: Das Unterschriftenfeld eines Zahlscheines nimmt ungefähr 8% der gesamten Zahlscheinfläche in Anspruch. Bei großen Distanzen zwischen Webcam und abgelichteten Zahlschein vermindert sich der Anteil des Unterschriftenfeldes zur

Gesamtfläche des abgelichteten Bildes entsprechend. Bei einer maximalen Kameradistanz zur Aufnahmefläche von 20 Zentimetern ergibt sich ein Zahlschein zu Gesamtbild-Verhältnis von ungefähr 1 zu 4. Daraus folgt, dass der Anteil des Unterschriftenfeldes in diesem Extremwertfall circa 2% der gesamten Bildfläche betragen würde. Bei einer minimal erforderlichen Auflösung der Kamera von 1 Megapixel, wären das 20000 Pixel. Um Sicherheitsbuffer zu berücksichtigen, bei zum Beispiel leicht größerem Abstand zwischen Kamera und Aufnahmefläche als 20 Zentimetern oder etwas kleineren Auflösungen als 1 Megapixel oder Rundungsfehlern in der Berechnung, wurde der berechnete Wert auf 10000 halbiert. Bei Verwendung von viel kleineren Auflösungen als 1 Megapixel oder viel größeren Abständen zwischen Kamera und Aufnahmefläche ist diese Lösung auf Grund der sich daraus ergebenden minderen Bildqualitäten nicht mehr verwendbar.

Zuletzt werden alle Vierecke verworfen, deren kleinster Winkel kleiner als 84,261° (größer als 0,1 im Bogenmaß) ist, um eine Approximation des gefundenen Vierecks an ein Rechteck, zu gewährleisten. Hierfür wird der Cosinus eingesetzt, um jeweils den Winkel zwischen den zwei Seiten, der vier Eckpunkte, zu berechnen. Seien *a* und *b* die zwei Vektoren, wobei *a* durch die Punkte P1 und P3 definiert ist und *b* durch die Punkte P2 und P3 definiert ist, die den Winkel ρ einschließen, dann errechnet sich der Winkel (im Bogenmaß) wie folgt:

$$\rho = \frac{a \cdot b}{|a| * |b|}$$

Tests haben gezeigt, dass 84° der kleinste Winkel ist bei dem der erfasste Zahlschein beziehungsweise das ermittelte Datenfeld am Zahlschein noch korrekt mittels OCR verarbeitet werden kann. Bei kleineren Winkeln werden, je nach Position der bedruckten Schrift am Zahlschein, Teile des Datenfeldes abgeschnitten und können somit nicht mehr an die OCR Engine zur weiteren Verarbeitung übergeben werden.

Von allen verbleibenden Konturen werden die größten zwei gespeichert. Diese sind im idealsten Fall das Datenfeld und das Unterschriftenfeld des Zahlscheines. Werden keine zwei passenden Konturen gefunden, war die Erkennung der Felder nicht erfolgreich. Beim ersten Fehlversuch werden die Parameter zur Erstellung des Kantenbildes verändert und die Erkennung wiederholt (siehe *Abbildung 32 - Flowchart: Image processing*). Beim zweiten Fehlversuch wird die Verarbeitung abgebrochen und ein Fehler geloggt und an LUI gesendet. *Abbildung 52 - Two biggest detected Rectangles* zeigt die bei erfolgreicher Erkennung gefundenen Vierecke, definiert durch jeweils vier Eckpunkte und abgebildet auf dem ursprünglichen Eingabebild aus *Abbildung 31 - Input Image captured from Cam*.

Abbildung 52 - Two biggest detected Rectangles

5.4.2.5.4. Bildrotation

Mit den Informationen der beiden gefundenen Vierecke kann die geometrische Transformation durchgeführt werden. *Abbildung 53 - Flowchart: Image Rotation* zeigt die dafür notwendigen Prozessschritte.

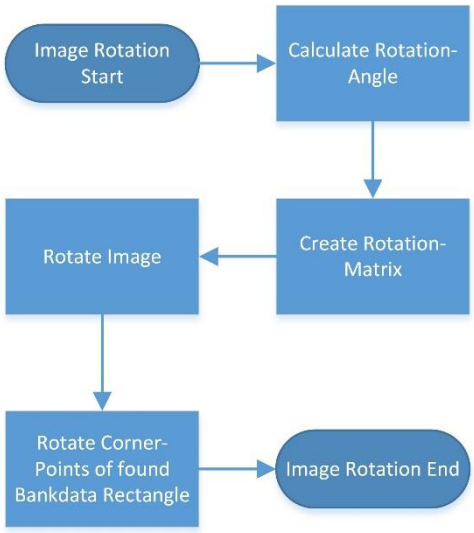

Abbildung 53 - Flowchart: Image Rotation

Im ersten Schritt muss der Rotationswinkel berechnet werden. Hierfür werden die beiden Schwerpunkte der gefundenen Vierecke berechnet. Diese sind jeweils durch den Schnittpunkt der Diagonalen definiert. Im Anschluss werden die Seitenlängen des größeren Rechteckes berechnet. Hierfür wird der *Pythagoräische Lehrsatz eingesetzt*, wobei die Delta der X- und Y-Koordinaten der zwei Punkte, welche die Seite definieren, als Katheten verwendet werden. Mit diesen Informationen kann hergeleitet werden, in welchem der folgenden vier Rotationsbereiche sich das Bild befindet:

1. Das Bild ist 0-90° rotiert, wenn, ausgehend vom Punkt mit der kleinsten X-Koordinate (des größeren Vierecks), die linke Seite größer ist als die rechte Seite, und die Y-Koordinate des Schwerpunktes des größeren Vierecks kleiner ist als die Y-Koordinate des Schwerpunktes des kleineren Vierecks

2. Das Bild ist 91-180° rotiert, wenn, ausgehend vom Punkt mit der kleinsten X-Koordinate (des größeren Vierecks), die linke Seite kleiner oder gleich ist als die rechte Seite, und die X-Koordinate des Schwerpunktes des größeren Vierecks kleiner oder gleich ist als die X-Koordinate des Schwerpunktes des kleineren Vierecks

3. Das Bild ist 181-270° rotiert, wenn, ausgehend vom Punkt mit der kleinsten X-Koordinate (des größeren Vierecks), die linke Seite größer ist, als die rechte Seite, und die Y-Koordinate des Schwerpunktes des größeren Vierecks größer oder gleich ist als die Y-Koordinate des Schwerpunktes des kleineren Vierecks

4. Das Bild ist 271-360° rotiert, wenn, ausgehend vom Punkt mit der kleinsten X-Koordinate (des größeren Vierecks), die linke Seite kleiner oder gleich ist als die rechte

Seite, und die X-Koordinate des Schwerpunktes des größeren Vierecks größer ist als die X-Koordinate des Schwerpunktes des kleineren Vierecks

Der Rotationswinkel wird mit dem *Arkustangens* von *(Gegenkathete / Ankathete)* berechnet. Das Ergebnis der Arkustangens-Funktion wird mit (360 / (2 * Pi)) multipliziert, um vom Bogenmaß ins Gradmaß umzuwandeln. Abhängig vom Rotationsbereich, muss der Winkel im Gradmaß vom Winkelteilkreis subtrahiert werden. Dies ergibt die folgenden Berechnungen passend zu den obigen vier Rotationsbereichen:

1. Rotationswinkel = 270 – (360 / (2 * Pi) * arctan(Gegenkathete / Ankathete)
2. Rotationswinkel = 270 – (360 / (2 * Pi) * arctan(Gegenkathete / Ankathete)
3. Rotationswinkel = 180 – (360 / (2 * Pi) * arctan(Gegenkathete / Ankathete)
4. Rotationswinkel = 90 – (360 / (2 * Pi) * arctan(Gegenkathete / Ankathete)

Mit dem Rotationwinkel kann nun folgende 2x3 Rotationsmatrix erstellt werden:

$$\begin{bmatrix} \alpha & \beta & (1-\alpha) * center.x - \beta * center.y \\ -\beta & \alpha & \beta * center.x + (1-\alpha) * center.y \end{bmatrix} \quad \text{mit} \quad \begin{matrix} \alpha = scale * \cos angle \\ \beta = scale * \sin angle \end{matrix}$$

Die Werte *center.x*, *center.y* und *scale* sind wie folgt definiert. *center.x* und *center.y* sind die Koordinaten des Punktes, um welchen rotiert wird. In dieser Arbeit wird um den Bildmittelpunkt rotiert, somit ist *center.x* gleich der halben Bildbreite und *center.y* gleich der halben Bildhöhe. *scale* bezieht sich auf den Zoomfaktor. In dieser Arbeit wird der *scale* mit 1 definiert, da das Bild in seiner Größendarstellung unverändert bleiben soll.

Mit der Rotationsmatrix kann nun die affine Transformation durchgeführt werden. Hierbei wird jeder Bildpunkt mit der Rotationsmatrix multipliziert. *Abbildung 54 - Rotated Image* zeigt das Ergebnis der Rotation des Bildes aus *Abbildung 52 - Two biggest detected Rectangles*.

Abbildung 54 - Rotated Image

Die zuvor gespeicherten Koordinaten der vier Eckpunkte des Vierecks mit den Überweisungsdaten müssen ebenfalls rotiert werden, damit diese zu dem rotierten Bild passen. Diese werden ebenfalls mit der Rotationsmatrix multipliziert. Im Anschluss werden die Koordinaten der vier Eckpunkte berechnet, um ein Rechteck zu erhalten. Hierfür werden jeweils die Mittelwerte der Distanzen zwischen den X- und Y-Koordinaten von benachbarten Punkten berechnet.

5.4.2.5.5. OCR-Verarbeitung der einzelnen Zahlscheinfelder

Nach der Rotation des Bildes muss dieses in die einzelnen Datenfelder des Zahlscheines zerschnitten werden. Im Anschluss wird jedes Datenfeld mit der OCR Engine verarbeitet, um die entsprechende textuelle Information zu extrahieren. *Abbildung 55 - Flowchart: Image Cropping* zeigt die Vorgehensweise der Bildverarbeitung, um die einzelnen Bildteile zu erhalten.

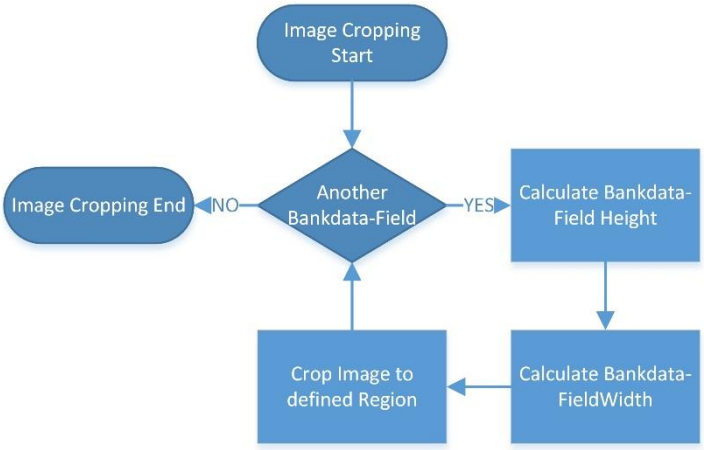

Abbildung 55 - Flowchart: Image Cropping

Im ersten Schritt wird das Bild auf das, durch die vier Eckpunkte definierte, Datenfeld des Zahlscheines zugeschnitten (siehe *Abbildung 56 - Cropped Input Image*).

Abbildung 56 - Cropped Input Image

Für das Filtern der Textteile des Bildes wird wiederum der HSV-Farbraum verwendet (siehe *Abbildung 57 - Cropped Image as HSV*).

Abbildung 57 - Cropped Image as HSV

Aufgrund der Spezifikation zur Verwendung von schwarzer OCR-B Druckfarbe wird das Bild nach dunklen Anteilen gefiltert. Die Filter-Grenzwerte werden hierbei wie folgt definiert:

Analog zur Festlegung der Grenzwerte für die Rahmenfindung werden auch hierfür die errechneten Bildwerte und die damit durchgeführten Tests herangezogen. Aufgrund dessen konnte dies im Zuge der Teststudie für die Rahmenerkennung durchgeführt werden.

Da die gesuchten Bildbereiche „möglichst schwarz“ sind, „Schwarz“ aber Farb-unabhängig ist, wird für den *Hue* Wert die gesamte Skala verwendet. In dieser Arbeit ist dies 0-180.

Da „Schwarz“ auch Sättigungs-unabhängig ist, wird auch für die Sättigung die gesamte Skala verwendet. In dieser Arbeit ist dies 0-255.

Somit bleibt die ausschlaggebende Größe des Thresholding der *Value*. Die Skala für *Value* reicht in dieser Implementierung von 0-255. Da die reine schwarze Farbe einen *Value* von 0 hat und im Gegensatz dazu die weiße Farbe einen Value von 255, ist der untere Grenzwert des gesuchten *Value*-Bereichs 0.

Bei einem Bildkontrast von über 700 und einer Bildhelligkeit von über 170, wird der obere Grenzwert für *Value* wird wie folgt gewählt:

Bei einer durchschnittlichen Sättigung des Bildes von unter 20, wird der obere Grenzwert mit 135 definiert.

Bei einer durchschnittlichen Sättigung des Bildes zwischen 20 und 30, wird der obere Grenzwert mit 125 definiert.

Bei einer durchschnittlichen Sättigung des Bildes zwischen 30 und 40, wird der obere Grenzwert mit 115 definiert.

Bei einer durchschnittlichen Sättigung des Bildes zwischen 40 und 50, wird der obere Grenzwert mit 105 definiert.

Bei einer durchschnittlichen Sättigung des Bildes zwischen 50 und 60, wird der obere Grenzwert mit 100 definiert.

Bei einer durchschnittlichen Sättigung des Bildes zwischen 60 und 70, wird der obere Grenzwert mit 90 definiert.

Bei einer durchschnittlichen Sättigung des Bildes von über 70, wird der obere Grenzwert mit 80 definiert.

Bei einem durchschnittlichen *Hue* Wert von unter 55, wird der untere Grenzwert für *Value* um 10 verringert. Bei einem durchschnittlichen *Hue* Wert von unter 40, wird der untere Grenzwert für *Value* ein weiteres Mal um 10 verringert. Grund hierfür ist der niedrigere *Value* Wert der Schrift bei relativ dunklen Bildern.

Bei einem Bildkontrast von unter 700 wird der obere Grenzwert für *Value* folgend definiert:

Liegt die Bildhelligkeit unter 200 wird der Grenzwertwert mit 90 definiert. Liegt die Bildhelligkeit über 200 wird der Grenzwert bei einer Bildsättigung von über 20, mit 160, bei einer Bildsättigung von unter 20, mit 100 definiert.

Abhängig von der Genauigkeit der eingesetzten Grenzwerte entsteht ein Binärbild wie auf den Abbildungen, *Abbildung 58 - Binary Image of Text with Noise* und *Abbildung 59 - Binary Image of Text without Noise*, ersichtlich. Ziel ist die Binarisierung um ausschließlich die Schrift zu erhalten. Aufgrund der variierenden Bildeigenschaften, die durch das Nuten der Webcam bei unterschiedlichen Belichtungsbedingungen entstehen, können keine exakten Grenzwerte festgelegt werden. Um keine zur Schrift gehörenden Pixel zu filtern, muss der Grenzwert für die Helligkeit daher tendenziell höher gewählt werden. Wird der Grenzwert zu niedrig gewählt werden unter Umständen Pixel gefiltert welche die bedruckte Schrift abbilden. Diese können nicht mehr hergestellt werden. Wird der Grenzwert höher gewählt werden diese Pixel nicht aus dem Bild gefiltert, es bleiben allerdings Störpixel im Bild, die das Ergebnis der OCR Verarbeitung

verschlechtern würden. Diese können jedoch durch morphologische Öffnung des Bildes gefiltert werden. Bei zu hohem Grenzwert können diese Störpixel auch durch morphologische Öffnung nicht mehr gefiltert werden und das Bild ist für eine weitere Verarbeitung mittels OCR unbrauchbar.

Wenn die Erkennung der Textteile des Bildes mit den definierten Grenzwerten nicht erfolgreich durchgeführt werden konnte, wird ein neuer Grenzwertbereich für *Value* festgelegt und die Erkennung erneut durchgeführt. Hierfür wird bei Bildern mit Kontrast über 700 und Helligkeit über 170 der obere Grenzwert für *Value* um 20 erhöht. Bei Bildern mit Kontrast unter 700 oder Helligkeit unter 170 wird der obere Grenzwert für *Value* um 10 reduziert. Ist die Erkennung ein zweites Mal nicht erfolgreich, wird die Verarbeitung abgebrochen und ein Fehler wird zurückgegeben.

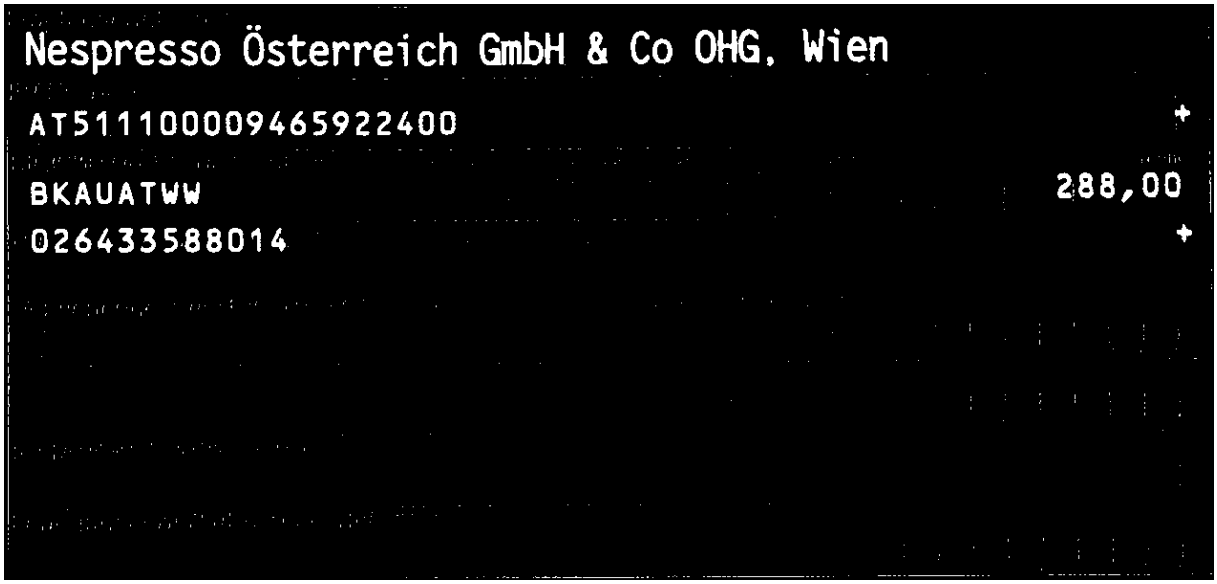

Abbildung 58 - Binary Image of Text with Noise

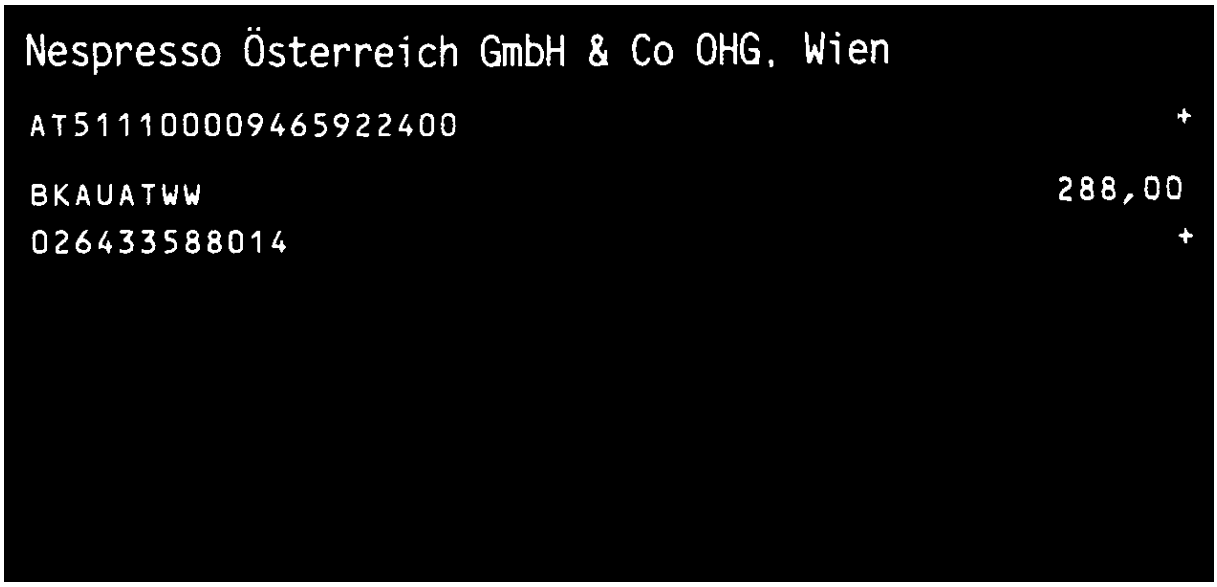

Abbildung 59 - Binary Image of Text without Noise

Morphologisches Öffnen ist das aufeinander folgende Erodieren und Erweitern eines Bildes unter Verwendung eines definierten Kernels einer definierten Größe. Beim Erodieren werden jene Pixel abgetragen, die in ihrer Umgebung keine Pixel der gleichen Färbung (in dieser Arbeit weiß) haben. Je größer der Kernel gewählt wird, desto größere Bildteile werden beim Erodieren gefiltert. In dieser Arbeit ist der einzig einsetzbare Kernel der kleinst-mögliche Kernel, nämlich jener der Größe 3x3. Bei größeren Kernels würden Teile der Schrift ebenfalls soweit abgetragen werden, dass diese beim darauf folgenden Erweitern nicht wieder erstellt werden würden. Das morphologische Erweitern des Bildes ist die gegensätzliche Verarbeitung des Abtragens/Erodierens. Unter Einsatz eines Kernels werden die vorhandenen, eingefärbten Bildteile erweitert. *Abbildung 60 - Image after Eroding* zeigt das Bild aus *Abbildung 58 - Binary Image of Text with Noise* nach dem Erodieren, *Abbildung 61 - Eroded Image after Dilation* zeigt das erodierte Bild aus *Abbildung 60 - Image after Eroding* nach der morphologischen Erweiterung.

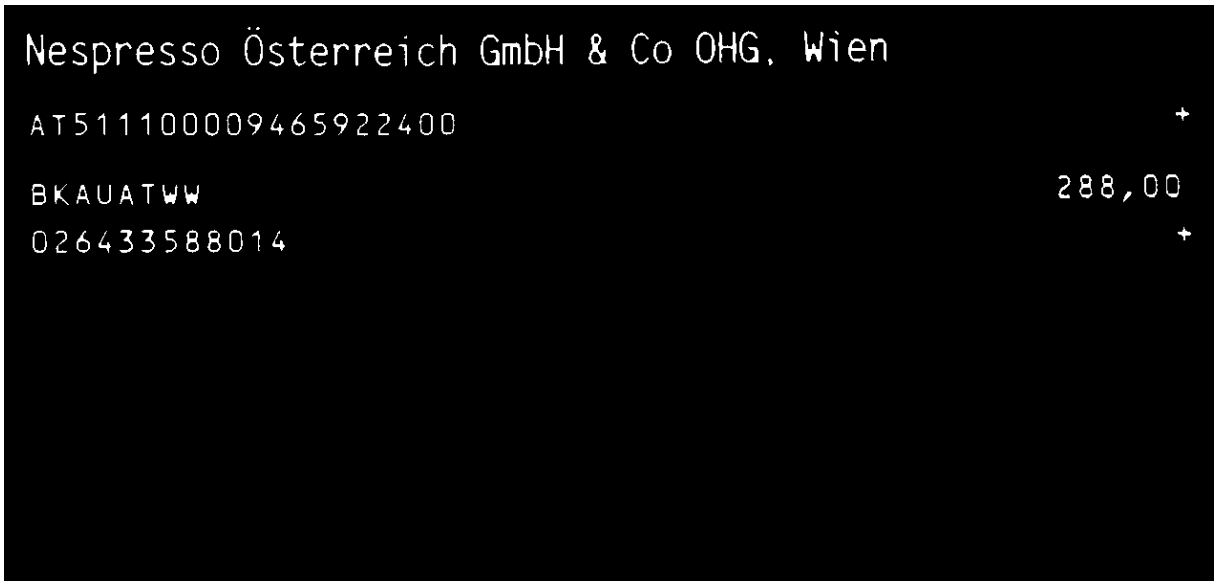

Abbildung 60 - Image after Eroding

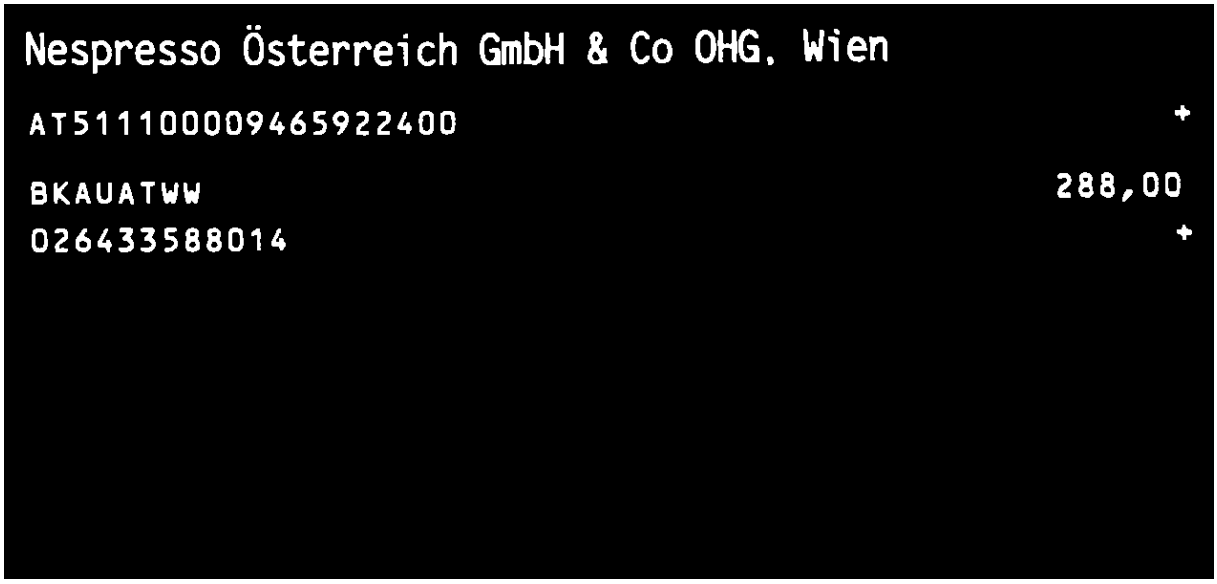

Abbildung 61 - Eroded Image after Dilation

Im Anschluss an die Binarisierung und gegebenenfalls morphologische Öffnung wird das Bild des Zahlscheines nach der Spezifikation der Stuzza in die einzelnen Datenfelder zerlegt. Zahlscheine haben Abmessungen von entweder 147,32mm oder 148,5mm Breite und entweder 101,6mm oder 105mm Höhe. Da die Dimensionen des verarbeiteten Zahlscheines nicht bekannt sind, wird dieser mittels prozentuellen Werten wie folgt in die einzelnen Felder zerlegt:

- Empfängername
    - Höhe: 12,5% der Gesamthöhe, beginnend bei 0% der Gesamthöhe

    - Breite: 100% der Gesamtbreite, beginnend bei 0% der Gesamtbreite
- IBAN des Empfängers
    - Höhe: 12,5% der Gesamthöhe, beginnend bei 12,5% der Gesamthöhe
    - Breite: 97,2% der Gesamtbreite, um etwaige Positionsmarkierungen nicht zu lesen, beginnend bei 0% der Gesamtbreite
- BIC
    - Höhe: 12,5% der Gesamthöhe, beginnend bei 25% der Gesamthöhe
    - Breite: 31,34% der Gesamtbreite, beginnend bei 0% der Gesamtbreite
- Betrag
    - Höhe: 12,5% der Gesamthöhe, beginnend bei 25% der Gesamthöhe
    - Breite: 31,34% der Gesamtbreite, beginnend bei 68,66% der Gesamtbreite
- Zahlungsreferenz
    - Höhe: 12,5% der Gesamthöhe, beginnend bei 37,5% der Gesamthöhe
    - Breite: 97,2% der Gesamtbreite, um etwaige Positionsmarkierungen nicht zu lesen, beginnend bei 0% der Gesamtbreite
- Verwendungszweck
    - Höhe: 25% der Gesamthöhe, beginnend bei 50% der Gesamthöhe
    - Breite: 100% der Gesamtbreite, beginnend bei 0% der Gesamtbreite
- IBAN des Absenders
    - Höhe: 12,5% der Gesamthöhe, beginnend bei 75% der Gesamthöhe
    - Breite: 97,2% der Gesamtbreite, um etwaige Positionsmarkierungen nicht zu lesen, beginnend bei 0% der Gesamtbreite
- Name des Absenders
    - Höhe: 12,5% der Gesamthöhe, beginnend bei 87,5% der Gesamthöhe
    - Breite: 100% der Gesamtbreite, beginnend bei 0% der Gesamtbreite

Abbildung 62 - Image of Segregated Payment Form Fields zeigt die Bilder der einzelnen Datenfelder des Zahlscheines.

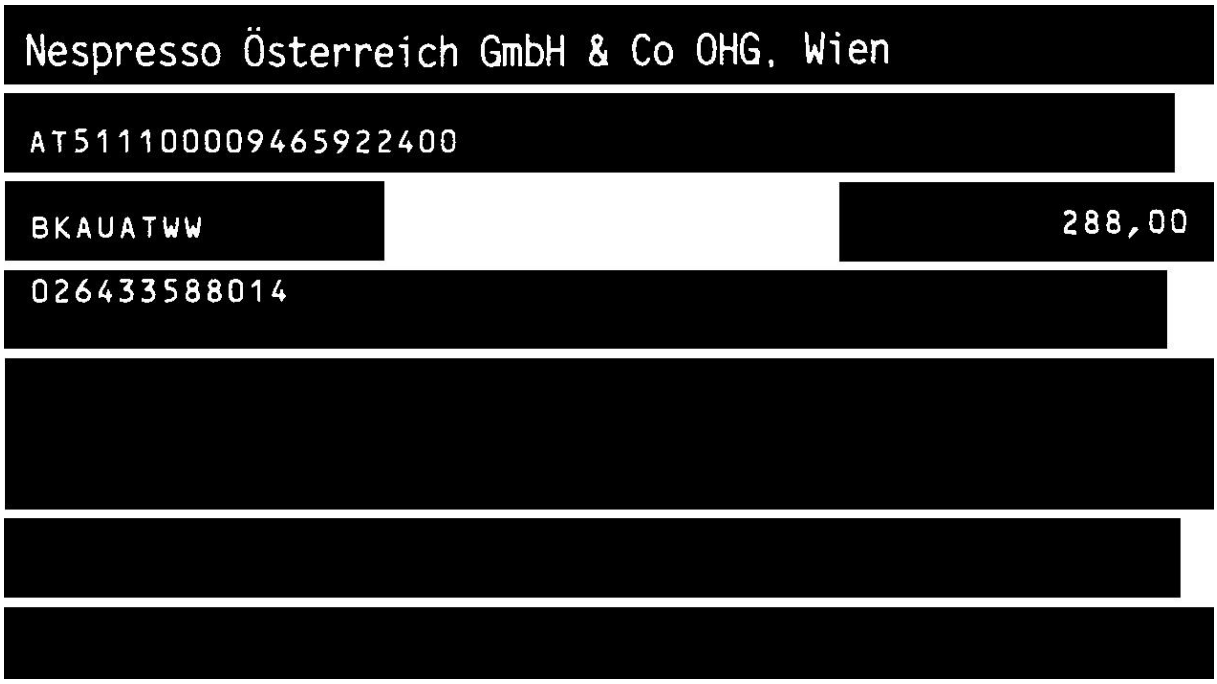

Abbildung 62 - Image of Segregated Payment Form Fields

Die einzelnen Bilder der Datenfelder werden einzeln mit der OCR Engine *Nicomsoft OCR SDK* verarbeitet.

Die Nicomsoft OCR Engine bietet im Namensraum *NSOCR* [92] 6 Objekttypen, mit denen die gesamte Funktionalität der Engine verwendbar ist. Diese sind:

```
NSOCR  : TNSOCR;
CfgObj : HCFG;
OcrObj : HOCR;
ImgObj : HIMG;
SvrObj : HSVR;
BlkObj : HBLK;
```

Über eine Objektinstanz der Klasse TNSOCR werden die anderen fünf Objekte verwendet. Bevor dies möglich ist, muss die Nicomsoft OCR Engine geladen und initialisiert werden [92]. Hierfür sind folgende zwei Aufrufe notwendig:

```
NSOCR := TNSOCR.Create('NSOCR.dll');
NSOCR.Engine_InitializeAdvanced(Cfg Obj, OcrObj, ImgObj);
```

Mit der Methode Engine_InitializeAdvanced wird die Nicomsoft Standard-Konfiguration geladen [92]. Diese kann dann beliebig an den eigenen Anwendungsfall angepasst werden.

Konnte die Engine erfolgreich initialisiert werden, stellen die fünf anderen Objekttypen folgende Funktionalität zur Verfügung:

- ***CFG object.*** *"Config" object stores and manages OCR settings. Refer to NSOCR Configuration section for more information about OCR settings. In most cases only one CFG object is required to be created and all other created objects will use it to access OCR settings. Creation of several CFG objects is necessary only if different settings are used for several OCR objects* [92].
- ***OCR object.*** *"OCR" object handles OCR process, it manages all resources that are necessary for OCR. In most cases only one OCR object is required to be created and it will process images one by one. Creation of several OCR objects is necessary only if several images must be processed at once in multi-threaded application* [92].
- ***IMG object.*** *"Image" object handles image for OCR. Image can be loaded from file on disk, from file in memory or from memory bitmap. Image can contain one or several pages. Many image file formats are supported: BMP, JPEG, PNG, TIFF, GIF and so on. PDF is supported via GhostScript library, refer to PDF support section for more information* [92].
- ***BLK object.*** *"Block" object handles image zone for OCR. This object is used to specify a zone size, position, type, get recognized text of a zone and so on* [92].
- ***SVR object.*** *"Saver" object handles saving OCR results to a file. Several formats are supported: ANSI TXT, Unicode TXT, RTF, PDF. Additional saving options are supported: page size selection, image-over-text feature for PDF, DPI selection for images, formatting options and so on* [92].
- ***SCAN object.*** *"Scan" object performs scanning images from TWAIN-compatible devices. Scanned image is placed to IMG object or can be saved to a file. Scanned image can contain one or several pages, it is possible to select scanning device, select DPI, color depth and other scanning settings* [92].

Das Aufrufen der OCR Engine mit den einzelnen Bildteilen liefert den in den Bildern enthaltenen Text als Zeichenkette zurück. Im Idealfall zu 100% korrekt wie in der folgenden Aufzählung, passend zum Eingabebild aus *Abbildung 31 - Input Image captured from Cam*:

- Empfänger: Nespresso Österreich GmbH & Co OHG. Wien
- Empfänger IBAN: AT511100009465922400
- BIC: BKAUATWW
- Betrag: 288,00
- Zahlungsreferenz: 026433588014
- Verwendungszweck:
- Auftraggeber:
- Auftraggeber IBAN:

Die Korrektheit der Daten kann nicht maschinell validiert werden. Lediglich der extrahierte IBAN kann auf Validität geprüft werden. Der Aufbau des IBAN ist wie folgt:

- 2-stelliger Ländercode
- 2-stellige Prüfsumme
- BBAN (länderspezifische Aneinanderreihung von unterschiedlichen Feldern und unterschiedlich langen Feldern des gleichen Typs, wie Bankleitzahl, Kontotyp, Kontonummer, Kontrollziffer, Regionalcode, Stelle der Filialnummer)

Für die Berechnung der Validität wird der 2-stellige Ländercode hinter den BBAN gereiht. Die Buchstaben werden hierbei so codiert, dass A = 10, B = 11,…Z= 35 ist. Die 2-stellige Prüfsumme wird hinter den Ländercode gereiht. Die durch diese Codierung entstandene Zahl muss bei der Division durch 97 einen Restwert von 1 aufweisen, um einen validen IBAN abzubilden [93].

Werden keine Textdaten in den Bildern gefunden, war die Erkennung, entweder aufgrund falscher Bilddaten oder aufgrund mangelhafter Bilddaten, nicht erfolgreich. Als Referenz wird das IBAN-Feld verwendet, da dieses eine sehr wichtige Information für Überweisungen enthält und zudem auf Validität geprüft werden kann. Beim ersten Fehlversuch werden die Parameter zur Erstellung des Binärbildes, welches nur die textuellen Inhalte abbilden soll, verändert und die Erkennung wiederholt (siehe *Abbildung 32 - Flowchart: Image processing*). Beim zweiten Fehlversuch wird die Verarbeitung abgebrochen und ein Fehler geloggt und an LUI gesendet.

Nach Beendigung der Verarbeitung, unabhängig ob erfolgreich oder nicht, kann eine erneute Verarbeitung gestartet werden.

5.5. Testen der Treiberfunktionalität

Um die Implementierung zu testen wurden die in der Folge beschriebenen Tests durchgeführt.

Da die Ergebnisse des Treibers maßgeblich von der Qualität von Bildrotation, der Rechteckserkennung, der Erkennung der Schrift und der Verarbeitung eben dieser mittels OCR Engine abhängen, beschränken sich die Tests auf die Verarbeitung des gleichen Zahlscheines unter verschiedenen Aufnahmebedingungen. Hierfür wird, mit Ausnahme des Tests der Bildrotation, pro Test jeweils das erfasste Bild, die errechneten Bildattribute Helligkeit, Kontrast und durchschnittliche HSV-Werte und die erkannten Textteile angegeben.

Korrekt erkannte Zeichen werden „schwarz“, inkorrekte Zeichen „rot“ und „fett“ dargestellt. Die erkannten Informationen des Feldes Verwendungszweck können vernachlässigt werden, da diese in beliebiger Schrift (nicht zwingend OCR-B) auf den Zahlschein gedruckt sind.

Die Testfälle 2-7 verarbeiten Zahlscheine die unter verschiedenen Beleuchtungssituationen abgelichtet wurde.

Die Testfälle 8-15 verarbeiten Zahlscheine die bei bestimmten Kamerakonfigurationen hinsichtlich Helligkeit, Kontrast und Sättigung abgelichtet wurden.

Test 1: Testen der Bildrotation aller vier Rotationsbereiche

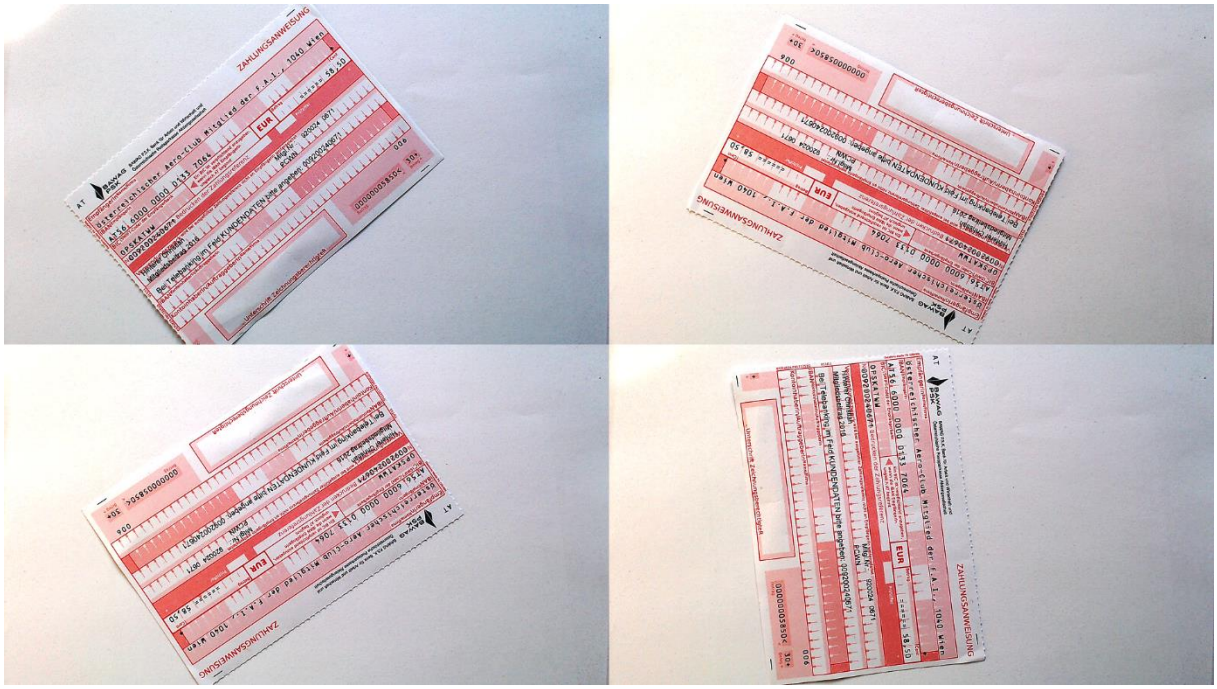

Abbildung 63 - Zahlscheine mit unterschiedlichen Rotationsbereichen

Abbildung 64 - Zahlscheine nach Rotationskorrektur

Die Rotation funktioniert für alle Rotationsbereiche korrekt. Voraussetzung hierfür ist die Erkennung des Rechtecks am Zahlschein, welches die Überweisungsdaten enthält.

Test 2: Verarbeitung eines bei Sonneneinstrahlung abgelichteten Zahlscheines

Abbildung 65 - Bei Sonneneinstrahlung abgelichteter Zahlschein

Berechnete Bildhelligkeit:	208
Berechneter Bildkontrast:	1222
Berechneter durchschnittlicher *Hue* Wert des Bildes:	95
Berechneter durchschnittlicher *Saturation* Wert des Bildes:	36
Berechneter durchschnittlicher *Value* Wert des Bildes:	226

Extrahierte Informationen:

Empfängername:	österreichischer Aero-Club Mitglied der F.A.I., 1040 Wien
IBAN:	AT566000000001337064 (valid)
BIC:	OPSKATWW
Betrag:	58,50
Zahlungsreferenz:	009200240671
Verwendungszweck:	Hinterer Christian Mitgl.Nr : 920024 0671 Mitgliedsbeitrag 2016 PCWN Bei Telebanking im Feld KUNDENDATEN bitte angeberr 009200240671

Die Daten am Zahlschein wurden bis auf ein Zeichen im Feld Verwendungszweck korrekt erkannt.

Test 3: Verarbeitung eines bei Tageslicht ohne direkte Sonneneinstrahlung abgelichteter Zahlschein

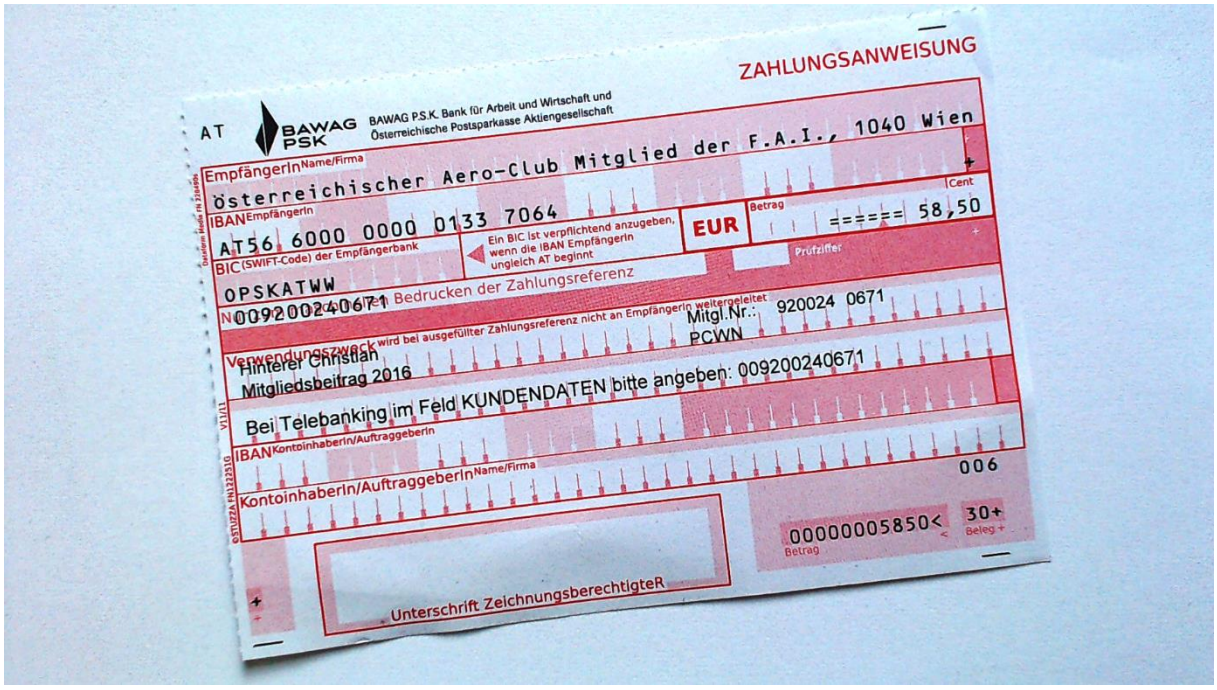

Abbildung 66 - Bei Tageslicht ohne Sonneneinstrahlung abgelichteter Zahlschein

Berechnete Bildhelligkeit:	202
Berechneter Bildkontrast:	886
Berechneter durchschnittlicher *Hue* Wert des Bildes:	50
Berechneter durchschnittlicher *Saturation* Wert des Bildes:	53
Berechneter durchschnittlicher *Value* Wert des Bildes:	228

Extrahierte Informationen:

Empfängername:	österreichischer Aero-Club Mit**cAj**ed der F.A.I., 1040 Wien
IBAN:	AT566000000001337064 (valid)
BIC:	OPSKATWW
Betrag:	58,50
Zahlungsreferenz:	009200240671
Verwendungszweck:	Hinterer Christian Mitgl.**hl**r.: 920024 0671
	--Mi**l**gliedsbeitrag 2016 PCWN

Bei Telebanking im Feld KUNDENDATEN bitte angeben: 009200249671

Test 4: Verarbeitung eines in der Dämmerung ohne Raumlicht abgelichteter Zahlschein

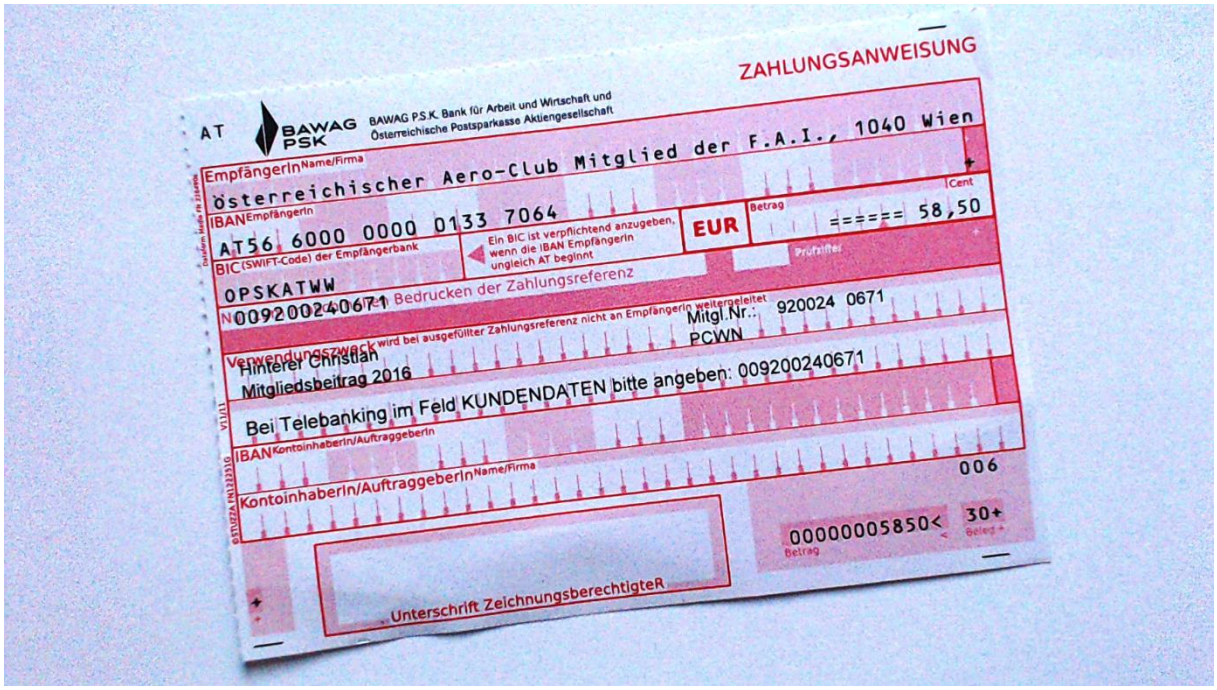

Abbildung 67 - In der Dämmerung ohne Raumlicht abgelichteter Zahlschein

Berechnete Bildhelligkeit:	202
Berechneter Bildkontrast:	809
Berechneter durchschnittlicher *Hue* Wert des Bildes:	126
Berechneter durchschnittlicher *Saturation* Wert des Bildes:	60
Berechneter durchschnittlicher *Value* Wert des Bildes:	238

Extrahierte Informationen:

Empfängername:	österreichischer Aero-Club Mitglied der F.A.I., 1040 Wien
IBAN:	AT566000000001337064 (valid)
BIC:	OPSKATWW
Betrag:	58,50
Zahlungsreferenz:	009200240671
Verwendungszweck:	Hinterer Christtan Mdgl Nr 920024 0671
	Mlt9lledsbeltra9 2016 PCWN
	Bei Telebanking im Feld KUNDENDATEN bitte angeben 009200240671

Test 5: Verarbeitung eines in der Dämmerung mit Raumlicht abgelichteter Zahlschein

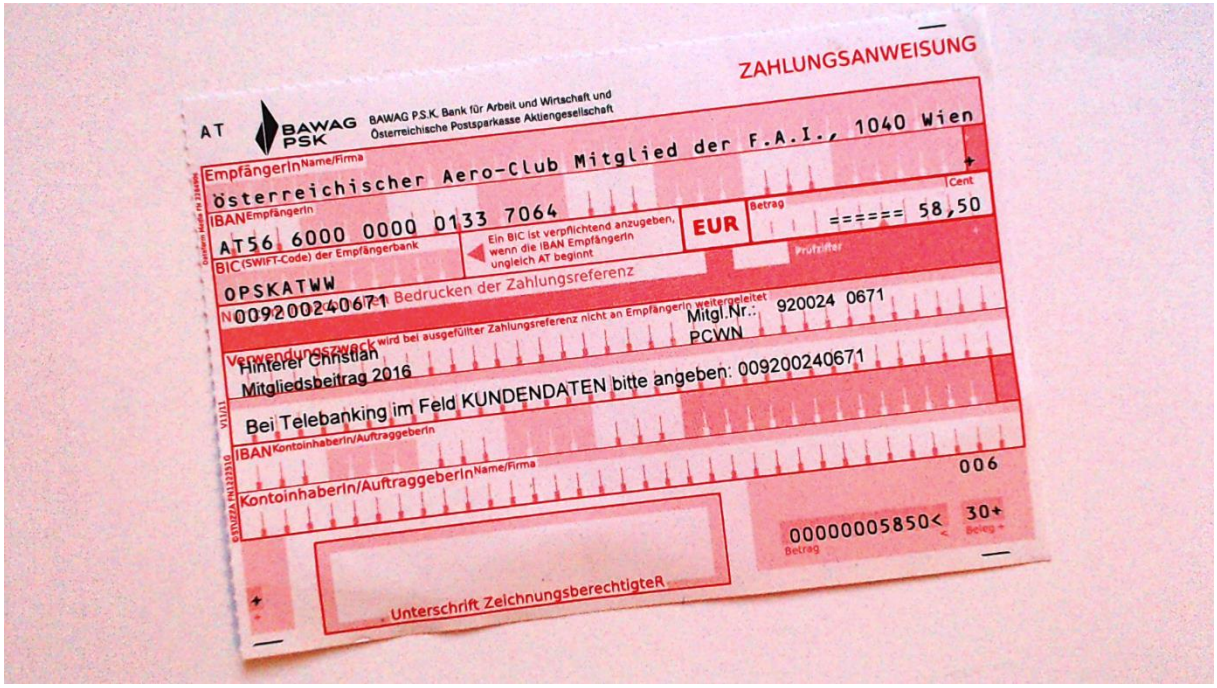

Abbildung 68 - In der Dämmerung mit Raumlicht abgelichteter Zahlschein

Berechnete Bildhelligkeit:	187
Berechneter Bildkontrast:	759
Berechneter durchschnittlicher *Hue* Wert des Bildes:	121
Berechneter durchschnittlicher *Saturation* Wert des Bildes:	56
Berechneter durchschnittlicher *Value* Wert des Bildes:	231

Extrahierte Informationen:

Empfängername:	österreichischer Aero-Club Mitgliecl der F.A.I., 1040 Wien
IBAN:	AT566000000001337064 (valid)
BIC:	OPSKATWW
Betrag:	58,50
Zahlungsreferenz:	009200240671
Verwendungszweck:	Hinterer Christian Mitgl.Nr.: 920024 0671 Mitgliedsbeitrag 2016 PCWN Bei Telebanking im Feld KUNDENDATEN bitte angeben: 009200240671

Test 6: Verarbeitung eines im dunklen Raum unter Verwendung einer direkten Beleuchtung auf das Zahlscheinfeld abgelichteten Zahlscheins

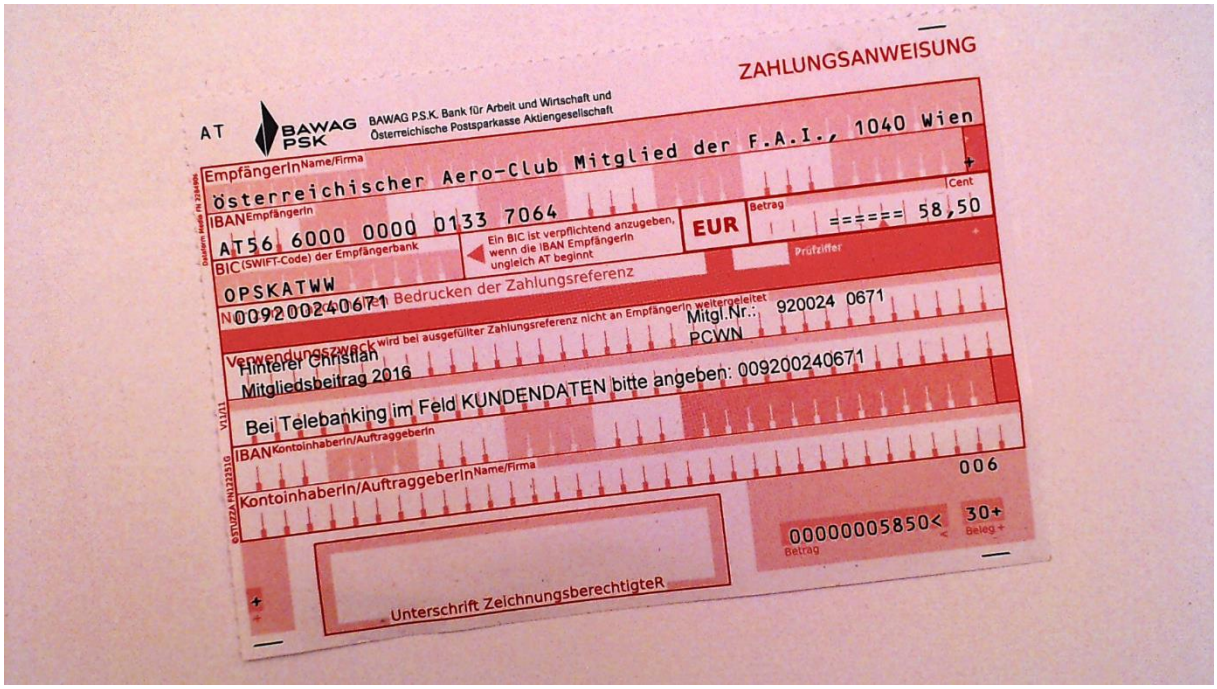

Abbildung 69 - Im dunklen Raum unter expliziter Beleuchtung des Zahlscheinfeldes abgelichteter Zahlschein

Berechnete Bildhelligkeit:	179
Berechneter Bildkontrast:	954
Berechneter durchschnittlicher *Hue* Wert des Bildes:	122
Berechneter durchschnittlicher *Saturation* Wert des Bildes:	73
Berechneter durchschnittlicher *Value* Wert des Bildes:	218

Extrahierte Informationen:

Empfängername:	österreichischer Aero-Club Mitqlieci der F.A.I., 1040 Hien
IBAN:	AT566000000001337064 (valid)
BIC:	OPSKATWW
Betrag:	58,50
Zahlungsreferenz:	009200240671
Verwendungszweck:	Hinterer Christlan Mitgl.Nr.: 920024 0671 Mitgliedsbeitrag 2016 PCWN Bei Telebanking im Feld KUNDENDATEN bitte angeben: 009200240671

Test 7: Verarbeitung eines im dunklen Raum abgelichteten Zahlscheines, unter Verwendung der Kamera-Autobelichtung

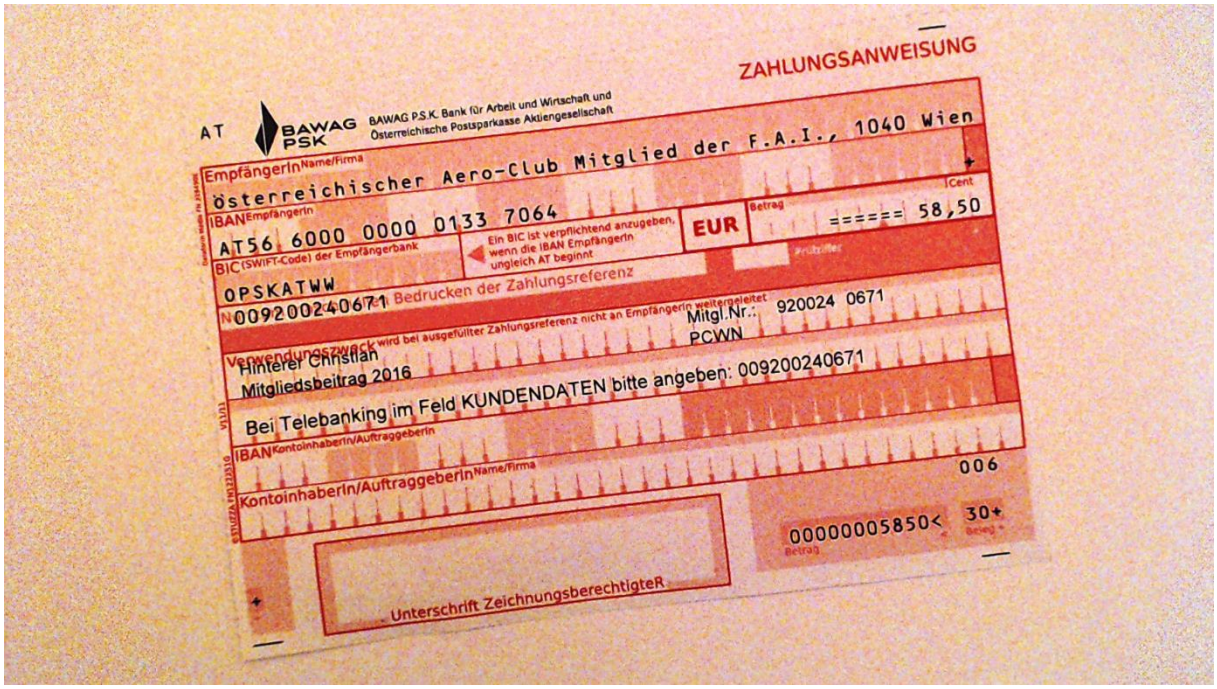

Abbildung 70 - Im dunklen Raum abgelichteter Zahlschein, unter Einsatz der Kamera-Autobelichtung

Berechnete Bildhelligkeit:	189
Berechneter Bildkontrast:	901
Berechneter durchschnittlicher *Hue* Wert des Bildes:	112
Berechneter durchschnittlicher *Saturation* Wert des Bildes:	94
Berechneter durchschnittlicher *Value* Wert des Bildes:	232

Extrahierte Informationen:

Empfängername:	österreichischer Aero-Club Mit**m**lied der F.A.I., 1040 **kl**ien
IBAN:	AT566000000001337064 (valid)
BIC:	OPSKATWW
Betrag:	58,50
Zahlungsreferenz:	009200240671
Verwendungszweck:	Hinterer Chr**t**stian Mitgl Nr 920024 0671
	lvhlghedsbeitrag 2016 PCWN
	Bei Telebanking im Feld KUNDENDATEN bitte angeben 009200240671

Test 8: Verarbeitung eines Zahlscheins, abgelichtet bei niedrig konfigurierter Helligkeit, hohem Kontrast und hoher Sättigung

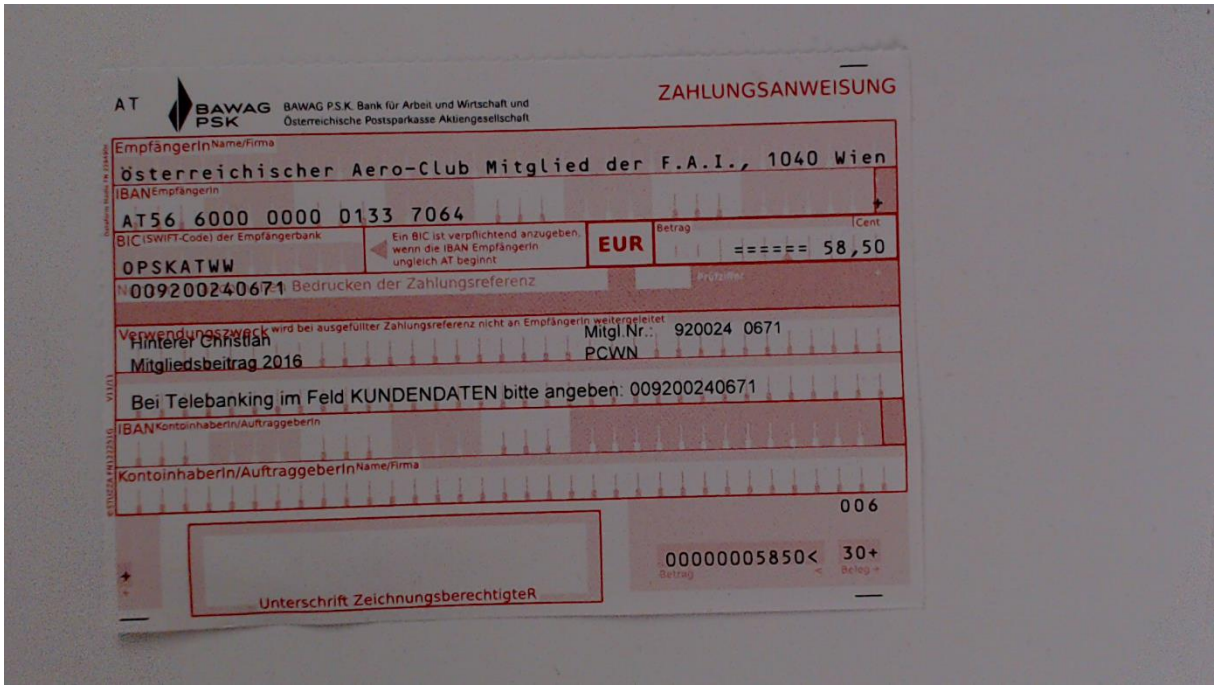

Abbildung 71 - Zahlschein, abgelichtet bei niedrig konfigurierter Helligkeit, hohem Kontrast und hoher Sättigung

Berechnete Bildhelligkeit:	146
Berechneter Bildkontrast:	266
Berechneter durchschnittlicher *Hue* Wert des Bildes:	126
Berechneter durchschnittlicher *Saturation* Wert des Bildes:	33
Berechneter durchschnittlicher *Value* Wert des Bildes:	158

Extrahierte Informationen:

Empfängername:	österreichischer Aero-Club MitgLied der F.A.I., 1040 Wien.
IBAN:	AT566000000001337064 (valid)
BIC:	OPSKATWW
Betrag:	58,50
Zahlungsreferenz:	I 009200240671
Verwendungszweck:	Hinterer thriitiän Mitgl Nr 920024 0671
	Milghedsbeitr-ag 2016 . - PCWN - . -
	Bet Telebanking im Feld KUNDENDATEN bit.t,e ang,eben ,0,09.?09.24gp.!1 , ,

Durch die geringe Helligkeit kombiniert mit dem hohen Kontrast, werden bei „dünnen“ Schriften wie im Feld Verwendungszweck einige Zeichen falsch erkannt und einige Bildstörungen fälschlicherweise als Zeichen erkannt. Bei diesem Szenario ist das Thresholding zur Extraktion der Schrift schwer umzusetzen, da die Bildwerde der Schrift und die Bildwerte der dunkelroten Beschriftung der Zahlscheinfelder ähnliche Attribute aufweisen.

Test 9: Verarbeitung eines Zahlscheins, abgelichtet bei niedrig konfigurierter Helligkeit, mittlerem Kontrast und hoher Sättigung

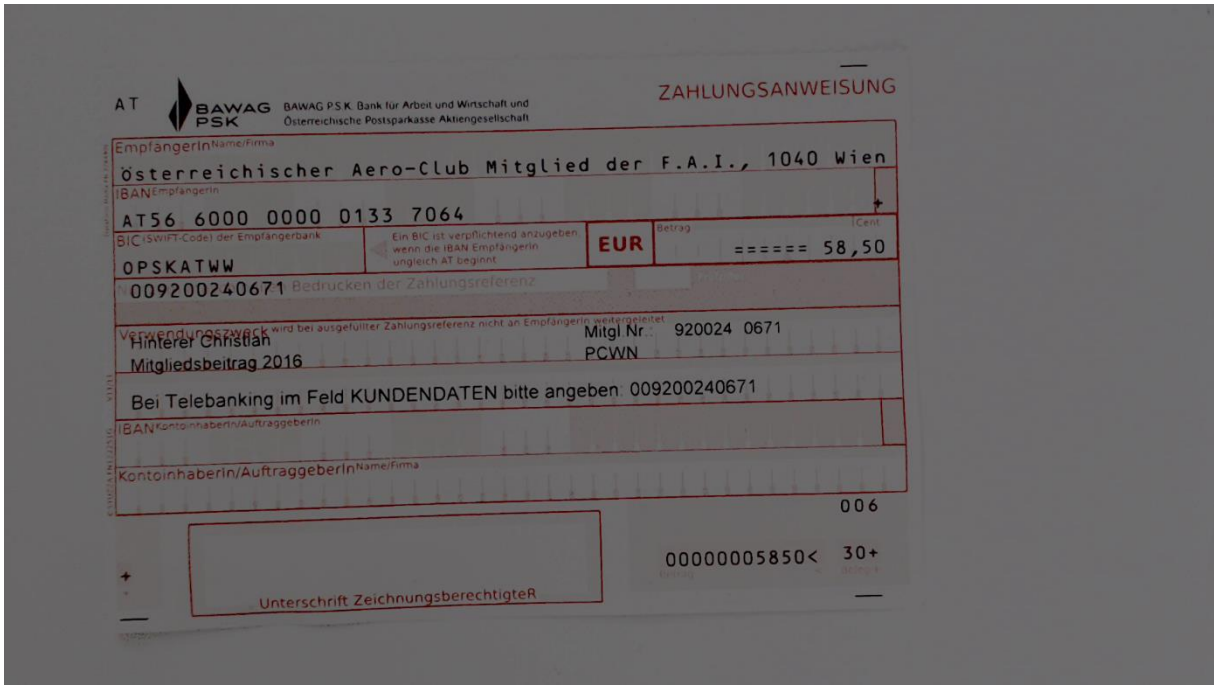

Abbildung 72 - Zahlschein, abgelichtet bei niedrig konfigurierter Helligkeit, mittlerem Kontrast und hoher Sättigung

Berechnete Bildhelligkeit:	153
Berechneter Bildkontrast:	106
Berechneter durchschnittlicher *Hue* Wert des Bildes:	114
Berechneter durchschnittlicher *Saturation* Wert des Bildes:	15
Berechneter durchschnittlicher *Value* Wert des Bildes:	158

Extrahierte Informationen:

Empfängername:	österreichischer Aero-Club Mitglied der F.A.I., 1040 Wien
IBAN:	**KU**66000000001337064 (invalid)
BIC:	**.,0**PS**,**KAT**,**W**,**W

Betrag:	58,50
Zahlungsreferenz:	009200240671
Verwendungszweck:	Hinteret Crmstiah hA!tgljedsbelKag 291-6 - .-
	Mitgl Nr 920024 0671 PCWN
	Bet Telebanking im Feld KUNDENDATEN bitte angeben ,0,09200240671 ,, ,,

Bei mittlerem Kontrast kombiniert mit geringer Helligkeit können einige Zeichen nicht korrekt erkannt werden.

Test 10: Verarbeitung eines Zahlscheins, abgelichtet bei niedrig konfigurierter Helligkeit, niedrigem Kontrast und mittlerer Sättigung

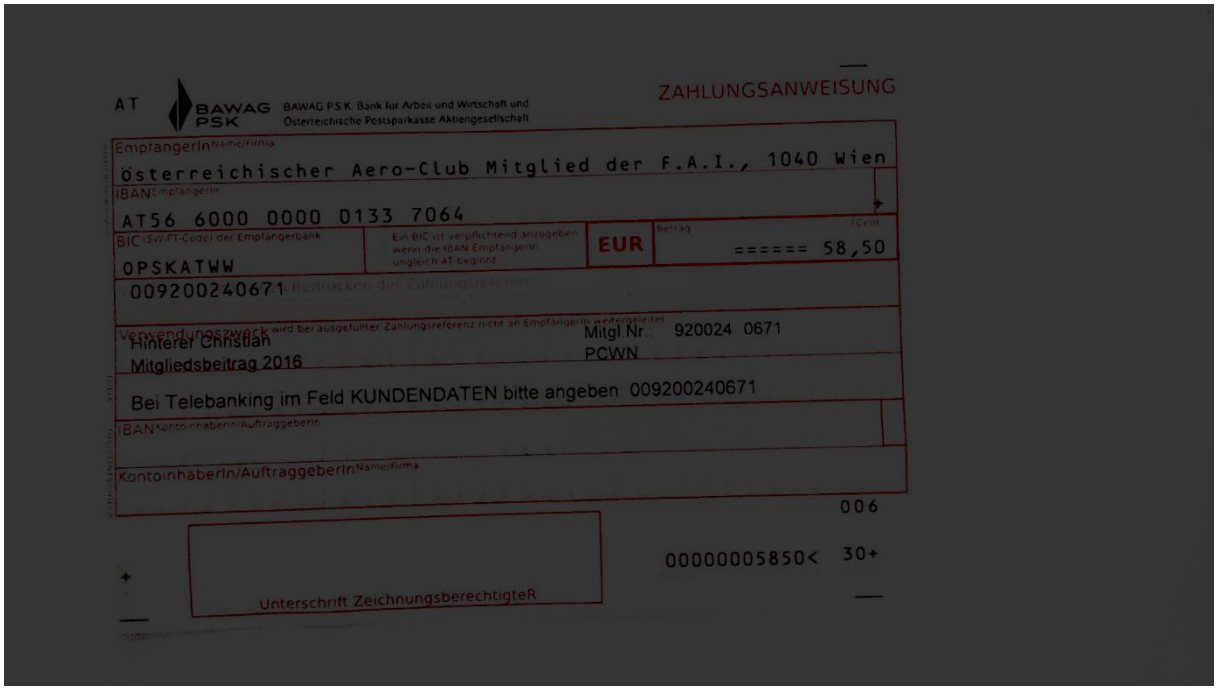

Abbildung 73 - Zahlschein, abgelichtet bei niedrig konfigurierter Helligkeit, niedrigem Kontrast und mittlerer Sättigung

Berechnete Bildhelligkeit:	115
Berechneter Bildkontrast:	71
Berechneter durchschnittlicher *Hue* Wert des Bildes:	17
Berechneter durchschnittlicher *Saturation* Wert des Bildes:	11
Berechneter durchschnittlicher *Value* Wert des Bildes:	118

Extrahierte Informationen:

Empfängername:	osterreichischer Aero-Club Mitglied aer F.A.I., 1Ö4Ö 4ien
IBAN:	AT566000000001337064 (valid)
BIC:	OPSKATWW
Betrag:	58,5
Zahlungsreferenz:	cjU9?0324067:
Verwendungszweck:	Hinteret ehristjah Mitgi Nr. 92ÖO24 067i
	.. Mitglie9sbeikag 2QW -. - - .. - - - --. -P-QWN-- - . -- -
	Bei Telebanking im Feld KUNEEU,+rEy bitt,e,a,U.g.e9,e? ,0,0920.0?40.6!1 - -

Niedriger Kontrast bei niedriger Bildhelligkeit führt zu vielen Erkennungsfehlern.

Test 11: Verarbeitung eines Zahlscheins, abgelichtet bei hoher konfigurierter Helligkeit, mittlerem Kontrast und mittlerer Sättigung

Abbildung 74 - Zahlschein, abgelichtet bei hoher konfigurierter Helligkeit, mittlerem Kontrast und mittlerer Sättigung

Berechnete Bildhelligkeit:	239
Berechneter Bildkontrast:	228
Berechneter durchschnittlicher *Hue* Wert des Bildes:	127

Berechneter durchschnittlicher *Saturation* Wert des Bildes: 24

Berechneter durchschnittlicher *Value* Wert des Bildes: 252

Extrahierte Informationen:

Empfängername: Osterreichischer Aero-Club Mitgtied der F.A.I., 1040 Wien

IBAN: AT566000000001337064 (valid)

BIC: OPSKATWW

Betrag: 58,50

Zahlungsreferenz: 009200240671

Verwendungszweck: Mtnlerer Christian Mitgl Nr 920024 0671

Mitgiieds.bedrag 2016 PCWN

Bei Telebanking im Feld KUNDENDATEN bitte angeben 009200240671

Test 12: Verarbeitung eines Zahlscheins, abgelichtet bei hoher konfigurierter Helligkeit, niedrigem Kontrast und hoher Sättigung

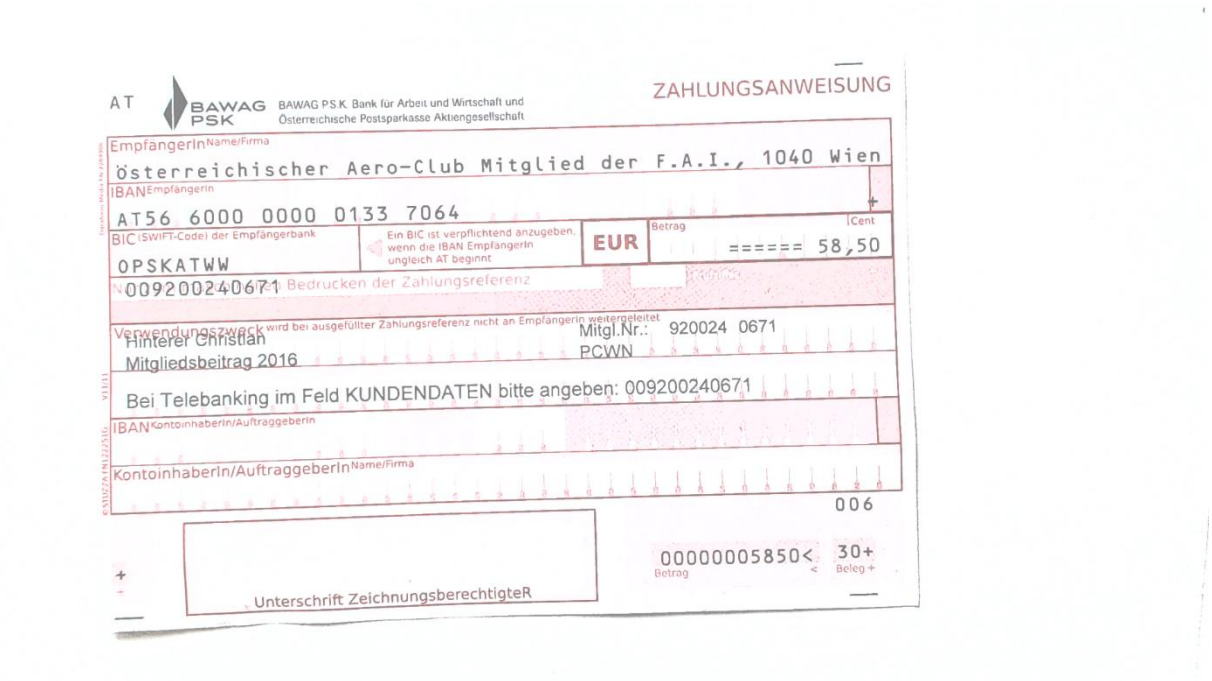

Abbildung 75 - Zahlschein, abgelichtet bei hoher konfigurierter Helligkeit, niedrigem Kontrast und hoher Sättigung

Berechnete Bildhelligkeit: 244

Berechneter Bildkontrast: 327

Berechneter durchschnittlicher *Hue* Wert des Bildes: 42

Berechneter durchschnittlicher *Saturation* Wert des Bildes: 8

Berechneter durchschnittlicher *Value* Wert des Bildes: 247

Extrahierte Informationen:

Empfängername: **7** Österreichischer Aero-C**t**ub Mitglied der F.A.I**-**, 1040 Wien

IBAN: AT566000000001337064**H** (invalid)

BIC: OPSKATWW

Betrag: 58,50,

Zahlungsreferenz: 009200240671

Verwendungszweck: Hinterer Christ**l**an -- Mitg**h**edsbeitrag 2016

jv1itgl.Nr**:**: 920024 0671 **-** PCWN

Bei Telebanking im Feld KUNDENDATEN bitte angeben: 009200240671

In diesem Szenario musste sowohl die Rechtecks- wie auch die Texterkennung zweimal durchgeführt werden, da die ursprünglichen Binarisierungsvorgänge zu viele notwendige Pixel filterten. Thresholding mit größeren Bildbereichen hat in diesem Szenario funktioniert, unter Umständen können aber zu viele Störpixel im Bild verbleiben.

Test 13: Verarbeitung eines Zahlscheins, abgelichtet bei niedrig konfigurierter Helligkeit, niedrigem Kontrast und mittlerer Sättigung

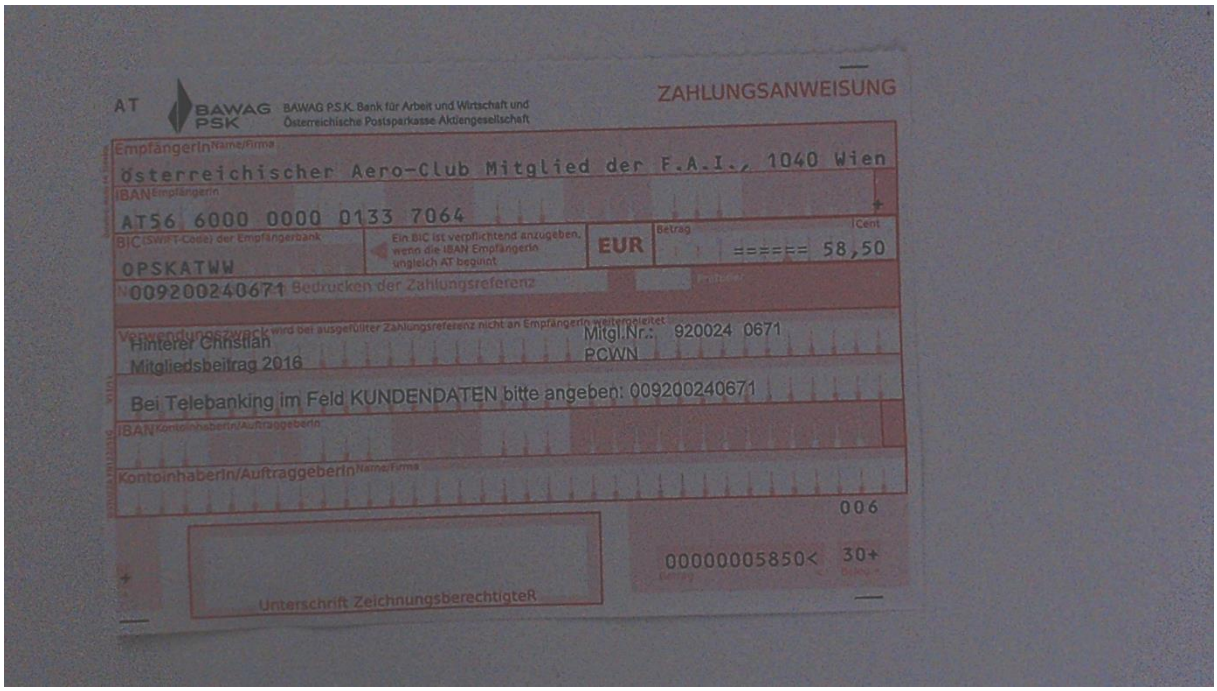

Abbildung 76 - Zahlschein, abgelichtet bei niedrig konfigurierter Helligkeit, niedrigem Kontrast und mittlerer Sättigung

Berechnete Bildhelligkeit:	115
Berechneter Bildkontrast:	324
Berechneter durchschnittlicher *Hue* Wert des Bildes:	113
Berechneter durchschnittlicher *Saturation* Wert des Bildes:	46
Berechneter durchschnittlicher *Value* Wert des Bildes:	131

Extrahierte Informationen:

Empfängername:	**: . . .- . ...**österreichischer Aero-Club Mitglied der F.A.**!**., 1040 **Id**ien
IBAN:	**Ä**T**S**66000000001337064 (invalid)
BIC:	OPSKATWW
Betrag:	58,50
Zahlungsreferenz:	009200240671
Verwendungszweck:	**.ri**int**C**re**t** Ch**n**st**t**an **. . .** .Mitgl.Nr.:. 920024 0671 **i** **. --.** M**%**liedsbe**jfc**ag 20.16**..- .... :- .;-.:-j-.-j- I--jj- AQN**N**-J- -- --- .. -- --.--. --- .- -(tltf** Bei Telebanking im Feld **5gy1?EyDATE.!! E,!!,2 a.!.g-?-p-2.2i**-9-9.20.**9,?g.?e**1**- .- .- - -, .- .i**

Bei geringer Helligkeit und niedrigem Kontrast wird unter Umständen auch die IBAN Information nicht richtig erkannt. Trotz zweimaliger Verarbeitung der Texterkennung konnte der IBAN nicht korrekt erkannt werden.

Test 14: Verarbeitung eines Zahlscheins, abgelichtet bei niedrig konfigurierter Helligkeit, niedrigem Kontrast und hoher Sättigung

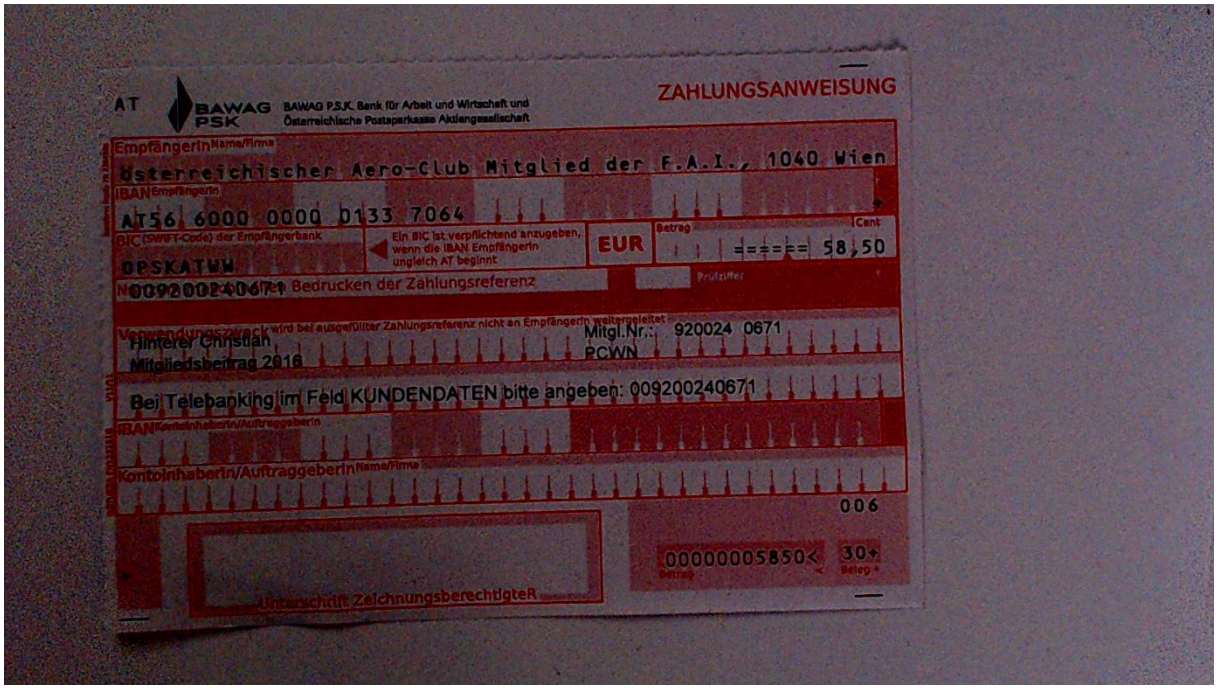

Abbildung 77 - Zahlschein, abgelichtet bei niedrig konfigurierter Helligkeit, niedrigem Kontrast und hoher Sättigung

Die Verarbeitung konnte nicht durchgeführt werden. Auf Grund des mangelnden Kontrastes und der mangelnden Helligkeit des Bildes, konnte das entsprechende Rechteck nicht gefiltert werden.

Test 15: Verarbeitung eines Zahlscheins, abgelichtet bei hoher konfigurierter Helligkeit, niedrigem Kontrast und niedriger Sättigung

Abbildung 78 - Zahlschein, abgelichtet bei hoher konfigurierter Helligkeit, niedrigem Kontrast und niedriger Sättigung

Die Verarbeitung konnte nicht durchgeführt werden. Auf Grund des mangelnden Kontrastes und der mangelnden Sättigung des Bildes, bei gleichzeitig überhöhter Helligkeit, konnte das entsprechende Rechteck nicht gefiltert werden.

5.6. Integration in LUI

Um die Integration des Treibers zu testen, wurde folgend beschriebenes Design für LUI erstellt. Das Design wurde passend zum vorhandenen LUI Design unter den Gesichtspunkten Benutzerfreundlichkeit und einfache, intuitive Bedienung entworfen.

The LUI is a combination of several programs and configuration files that act together to present an environment for the user to interact with. Appearance and behavior of the UI is set via configuration files. Different configuration files can be used for different users or for different applications [8].

Die Kommunikation zwischen Treiber und LUI erfolgt mittels Named Pipes unter Verwendung eines *pipe driver*. *Abbildung 79 - General Structure of the LUI* zeigt die Architektur von LUI.

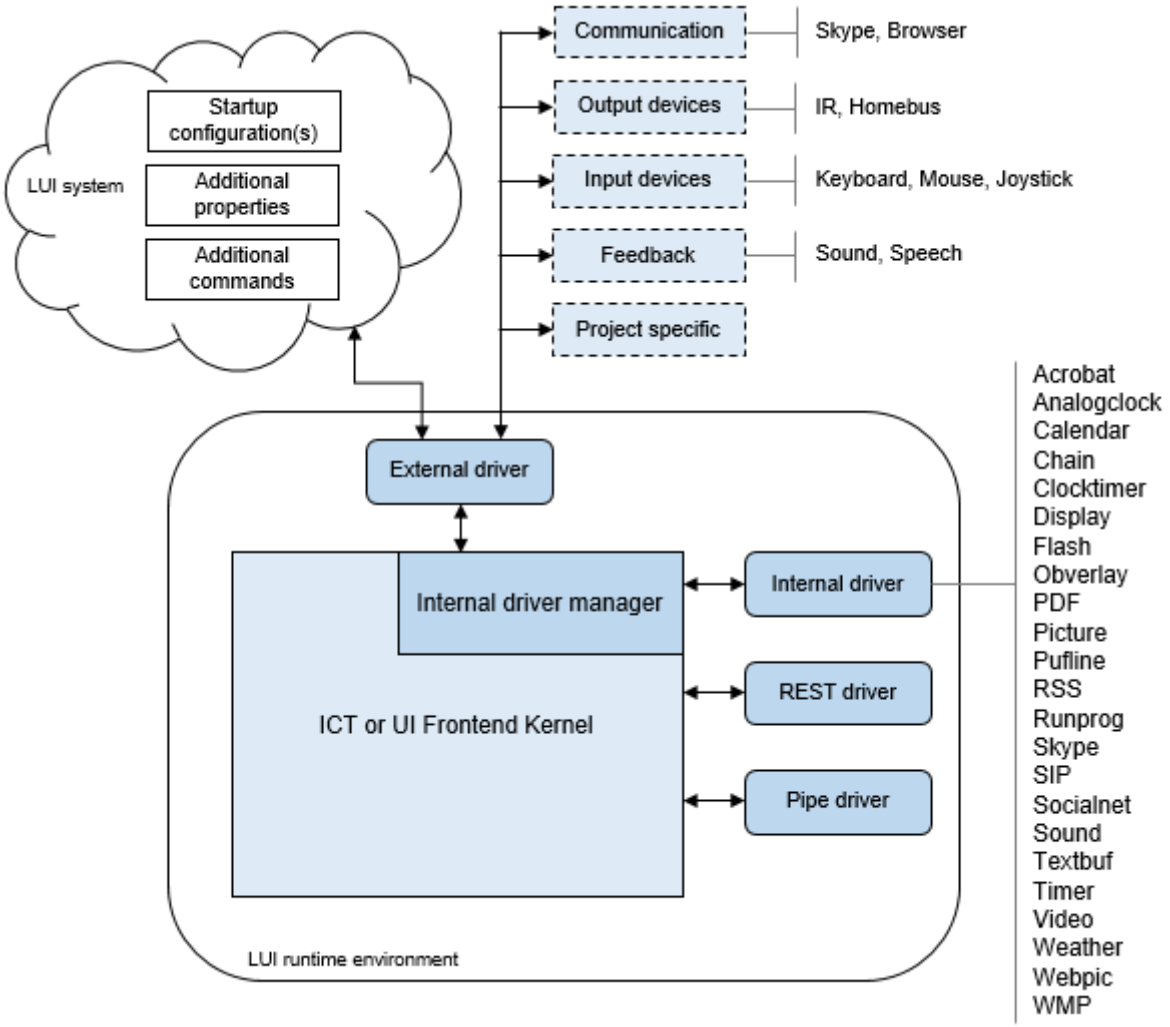

Abbildung 79 - General Structure of the LUI [8]

Die LUI runtime environment besteht aus dem Hauptprogramm LUI.exe, welches den Kernel und einige eingebettete Treiber beinhaltet. Das Hinzufügen von weiteren Treibern ist möglich [8].

Die Implementierung dieser Arbeit ist genauso ein Treiber, mit welchem man die Gesamtfunktionalität von LUI erweitern kann. Außerdem muss ein *UI Configuration file* erstellt, oder ein vorhandenes erweitert, werden, um den Treiber in LUI einzubetten und die Konfiguration bezüglich LUI-Oberfläche zur Verwendung der Treiber-Funktionalität zu konfigurieren. Um eine navigierbare Benutzeroberfläche für den Treiber in LUI zu erstellen muss die Menüstruktur von LUI erweitert werden. *Abbildung 80 - Principle Menu Structure of LUI* [8] zeigt den prinzipiellen Aufbau des LUI Bildschirm-Layouts. Ein *menu* selbst bildet dabei eine Bildschirmseite ab indem es als Kontainer für *entries* agiert. *Entries* sind ebenfalls nicht sichtbar und agieren ebenfalls als Kontainer. Ein *entry* kann seinerseits wieder *entries* oder einzelne *objects* enthalten. *Objects* sind die am Bildschirm angezeigten sichtbaren entities [8].

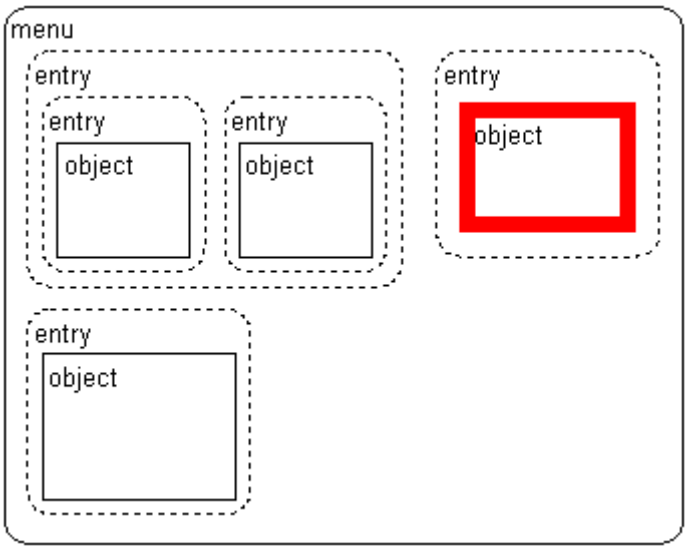

Abbildung 80 - Principle Menu Structure of LUI [8]

Pro angezeigter Oberfläche muss ein Menü erstellt werden. Zum Testen der Integration wurde eine Oberfläche aus sieben Ansichten (Menüs) entworfen:

1. Der aktive LUI Startbildschirm (*Abbildung 81 - LUI Designentwurf: Startbildschirm*): Der Benutzer bzw. die Benutzerin hat die Möglichkeit den Knopf für die Funktionalität „Überweisung“ zu klicken.
2. Auswahl des Auftraggeberkontos (*Abbildung 82 - LUI Designentwurf: Auftragskonto*): Bei mehreren gespeicherten Auftraggeberkonten kann der Benutzer bzw. die Benutzerin durch Klick auf den IBAN in die 3. Ansicht wechseln.
3. Auftraggeberkonto ändern (*Abbildung 83 - LUI Designentwurf: Auftragskonto ändern*): Alle gespeicherten Auftraggeberkonten werden angezeigt. Durch Klick auf den IBAN kann der Benutzer bzw. die Benutzerin das entsprechende Konto wählen.
4. OCR Verarbeitung starten und „laufend“ (*Abbildung 84 - LUI Designentwurf: OCR Verarbeitung starten* und *Abbildung 85 - LUI Designentwurf: OCR Verarbeitung laufend*): Durch Klick auf den Start-Button kann der Benutzer bzw. die Benutzerin die Verarbeitung starten. Während der Verarbeitung wird der Prozessfortschritt am Bildschirm angezeigt. Bei Fehlern der Verarbeitung wird eine entsprechende Fehlermeldung am Bildschirm angezeigt. Nach erfolgreicher Verarbeitung wechselt die Anwendung in die 5. Ansicht.
5. OCR Ergebnisse 1 von 2 (*Abbildung 86 - LUI Designentwurf: Ergebnisanzeige Seite 1/2*): Anzeige der extrahierten Informationen für „Empfängername“, „IBAN des Empfängers“, „Validität des IBAN“ und „Betrag“. Durch Klick auf die entsprechende Information wird in die 7. Ansicht gewechselt.
6. OCR Ergebnisse 2 von 2 (*Abbildung 87 - LUI Designentwurf: Ergebnisanzeige Seite 2/2*): Anzeige der extrahierten Informationen für „BIC“, „Zahlungsreferenz“ und „Verwendungszweck“. Druch Klick auf die entsprechende Information wird in die 7. Ansicht gewechselt.

7. OCR Ergebnisse bearbeiten (*Abbildung 88 - LUI-Designentwurf: Daten editieren*): Manuelle Bearbeitung der gefundenen Information.

Die Abbildungen 63, 64, 65, 68 und 69, welche die Anzeige der Überweisungsdaten hinterlegen, stammen aus dem Design von Robert Koch, entstanden im Zuge seiner Arbeit *„Paper to Byte“* [7].

Um das Design von LUI modular entwerfen zu können, muss die Struktur der Menüs zur Laufzeit an den Treiber übermittelt werden. Die *LUI objects* die Informationen vom Treiber anzeigen sollen müssen einen *custom tag* enthalten. Dieser *custom tag* muss seinerseits einen *objecttype tag* mit einem der folgenden Inhalte enthalten:

- bank_process_status: zur Anzeige von Fehlermeldungen und des aktuellen Prozessschrittes
- bank_progress: zur Anzeige des Prozessfortschrittes der OCR- und Bildverarbeitung
- bank_receiver: zur Anzeige der Information „Empfängername“
- bank_receiver_iban: zur Anzeige der Information „IBAN des Empfängers“
- bank_iban_validity: zur Anzeige der Information „der Validität des extrahierten IBAN“
- bank_bic: zur Anzeige der Information „BIC“
- bank_amount: zur Anzeige der Information „Betrag“
- bank_reference: zur Anzeige der Information „Zahlungsreferenz“
- bank_purpose: zur Anzeige der Information „Verwendungszweck“

Weiters kann der Treiber folgende *commands* an LUI übermitteln:

- camunavailable: wenn die konfigurierte Kamera nicht verfügbar ist
- ocrengineunavailable: wenn die OCR Engine nicht verfügbar ist
- ocrsuccessful: wenn die OCR Verarbeitung erfolgreich war
- ocrunsuccessful: wenn die OCR Verarbeitung nicht erfolgreich war
- ibanvalid: wenn der extrahierte IBAN valide ist
- ibaninvalid: wenn der extrahierte IBAN nicht valide ist

Um den Treiber zu starten muss seitens LUI folgendes Startkommando an den Treiber übermittelt werden:

```
<command type="pipedriver" ignoreerror="true">
  <driver>[Name der Pipe]</driver>
  <data>START_OCR</data>
</command>
```

Ein weiterer Teil der Konfiguration beinhaltet die Definition der *Named Pipe* zur Kommunikation zwischen Treiber und LUI. Die Konfiguration der *pipe* umfasst auch die Definition der Startdatei des Treibers. Der Mechanismus ist bereits in LUI implementiert. LUI startet den Treiber, respektive die konfigurierte Datei. Sobald der Treiber läuft, werden zwischen LUI und dem Treiber die zwei pipes (eingehende und ausgehende) aufgebaut bzw. verbunden.

Zuletzt müssen in der LUI Konfiguration des Treibers der Name und die Auflösung der zu verwendenden Kamera angegeben werden. Um den Kameratreiber-internen Namen, sowie

die von der Kamera unterstützen Auflösungen vom Kameratreiber auszulesen, kann der LUI Designer die implementierte Anwendung aus Kapitel *5.7. Camera-Resolution-Tool für Video-aufnahmegeräte* einsetzen. Der Name der Kamera, sowie der Index für die Auflösung sind in den *tags camname* und *resolutionindex* des *pipe drivers* „OCRpipe“ anzugeben.

Da die Kamera dieser Teststellung auf die eingesetzte Distanz zum abzulichtenden Dokument den Fokus nicht automatisch berechnen kann, muss dieser manuell konfiguriert werden.

Die Beschreibung zur Erstellung eines LUI Designs für den Treiber ist der, dem Testdesign beigefügten Beschreibung, bzw. in weitere Folge dem *ISUI Manuel* zu entnehmen.

Die Konfiguration für das entworfene Design dieser Arbeit ist der beigelegten CD zu entnehmen.

Abbildung 81 - LUI Designentwurf: Startbildschirm

Abbildung 82 - LUI Designentwurf: Auftragskonto

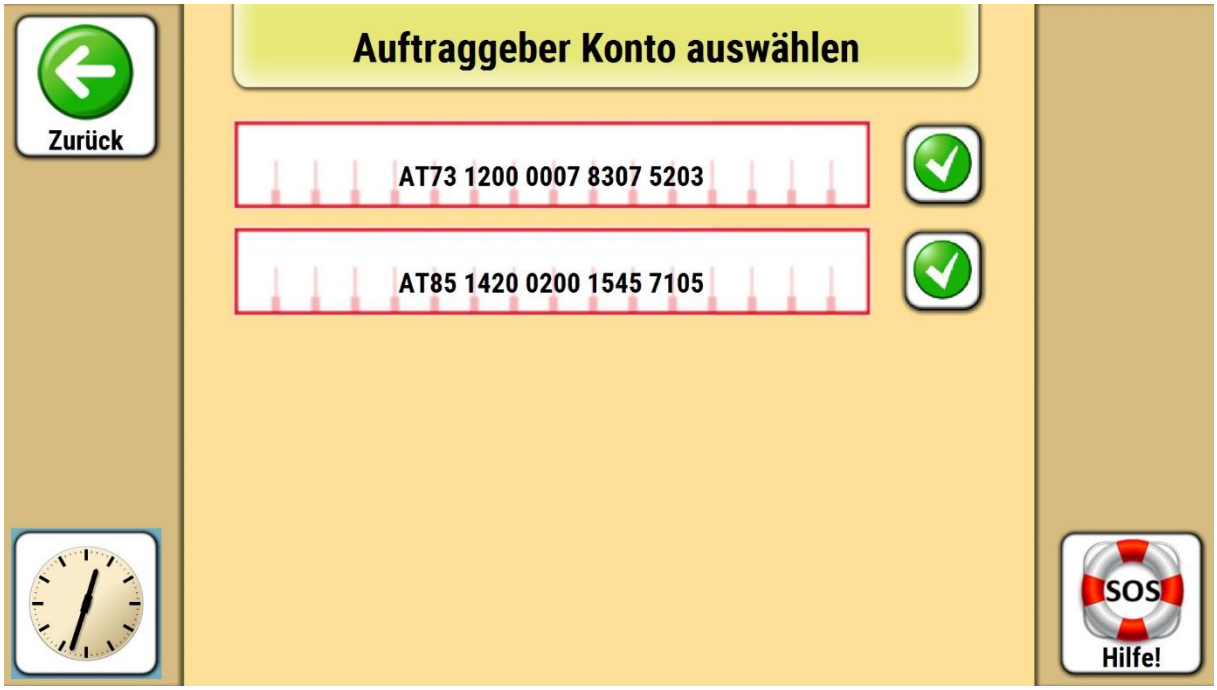

Abbildung 83 - LUI Designentwurf: Auftragskonto ändern

Abbildung 84 - LUI Designentwurf: OCR Verarbeitung starten

Abbildung 85 - LUI Designentwurf: OCR Verarbeitung laufend

Abbildung 86 - LUI Designentwurf: Ergebnisanzeige Seite 1/2

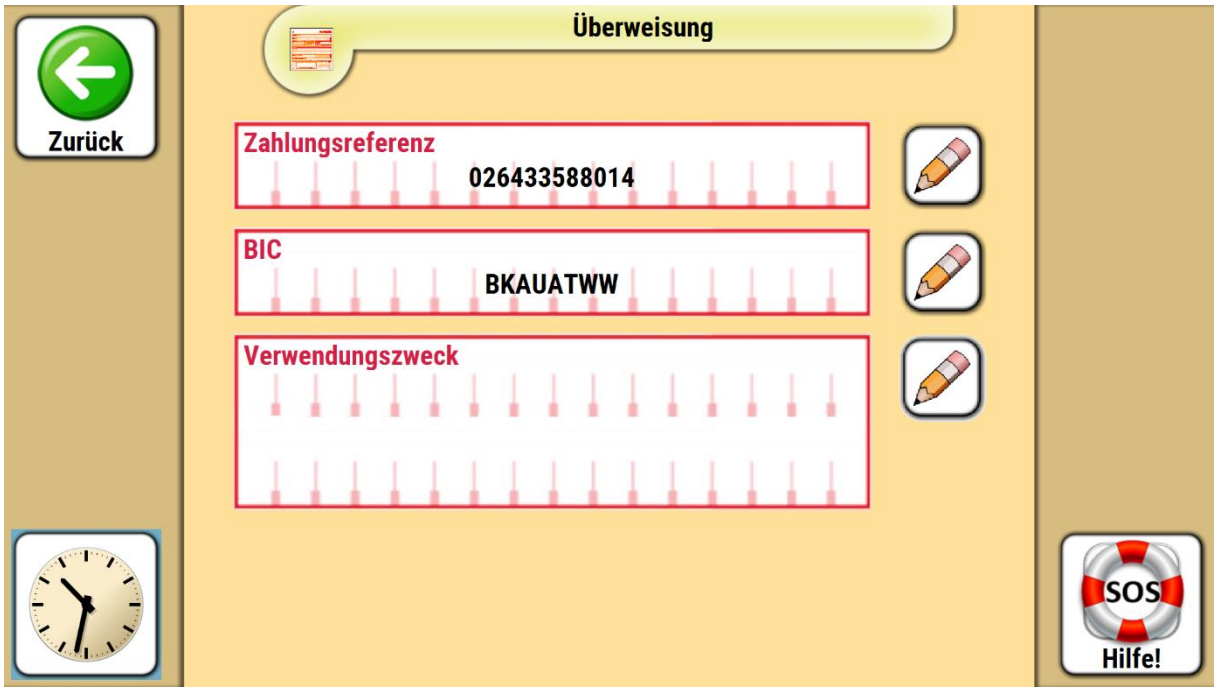

Abbildung 87 - LUI Designentwurf: Ergebnisanzeige Seite 2/2

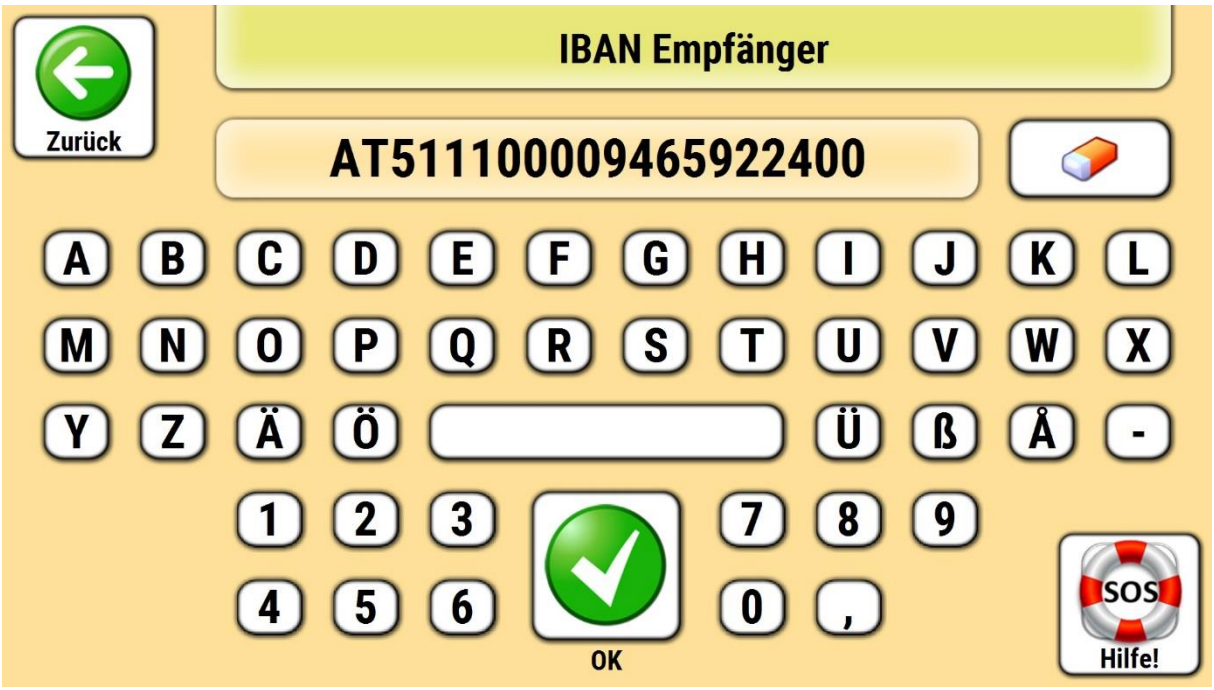

Abbildung 88 - LUI-Designentwurf: Daten editieren

5.7. Camera-Resolution-Tool für Videoaufnahmegeräte

Die Integration des OCR Treibers, in LUI, benötigt die Konfiguration des Windows-Treiber-internen Gerätenamen der Kamera, sowie die Auflösung, die verwendet werden soll. Um es dem LUI-Designer zu erleichtern, diese Informationen zu beschaffen, wurde eine Anwendung implementiert, das *Camera-Resolution-Tool*. Diese Anwendung nützt die gleiche DirectShow-Schnittstelle, wie der entwickelte OCR Treiber, um die Information vom Kameratreiber bzw. den Kameratreibern auszulesen. *Abbildung 89 - Screenshot des CameraResolutionTool* zeigt die Oberfläche des Tools.

Im Kombinationsfeld (engl. *combo box*) werden alle verfügbaren Geräte angezeigt. Der angezeigte Name ist gleichzeitig der Name, welcher in der LUI-Konfigruation als *CameraDeviceName* verwendet werden muss. Wählt man eine Kamera aus dem Kombinationsfeld, werden im Listenfeld (engl. *list box*) alle verfügbarn Auflösungen dieses Gerätes angezeigt. Der Index aus der ersten Spalte des Listenfeldes ist gleichzeitig der Index, welcher in der LUI-Konfiguration als *CameraResolutionIndex* verwendet werden muss. Mit dem Kontrollkästchen (engl. *check box*) kann man die angezeigten Auflösungen, auf jene des RGB-Formats eingrenzen. Der OCR Treiber unterstützt neben unkomprimierten Daten, die Kodierungen *YUY2*, *YUYV*, *YUNV*, *MJPG*, *I420*, *YV12* und *IYUV*.

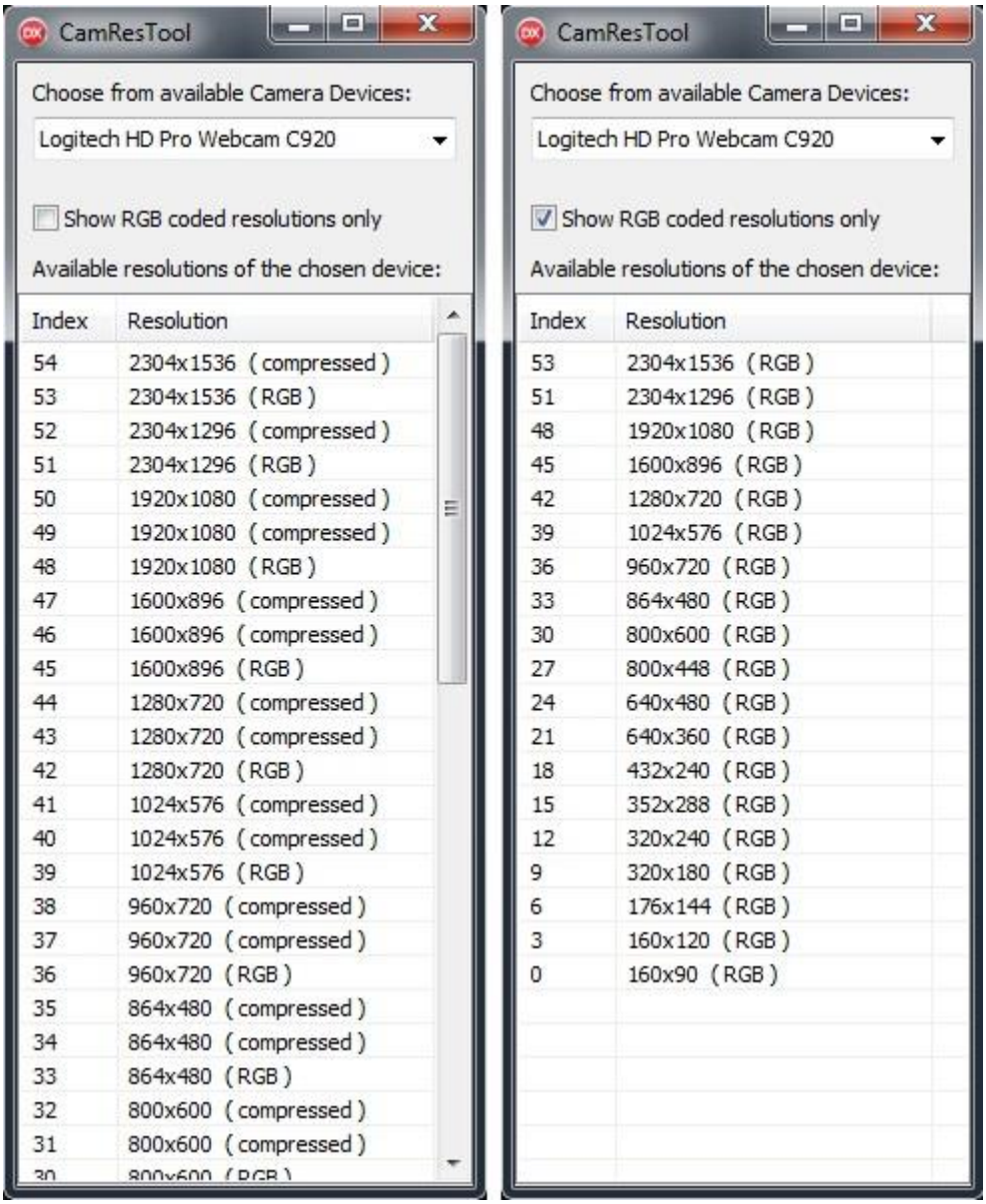

Abbildung 89 - Screenshot des CameraResolutionTool

6. Zusammenfassung und Schlussfolgerung

Die Analyse des Senioren Terminals und die im Zuge dieser Arbeit entstandene Implementierung haben gezeigt, dass das Senioren Terminal grundsätzlich, entsprechend der in dieser Arbeit gestellten Anforderungen, erweitert werden kann.

Nach der Erfassung des Standes der Technik bezüglich OCR und ICR Algorithmen, sowie von OCR im Finanzwesen, existenter ähnlicher Lösungen und Bildbearbeitungstechniken, wurden am Markt erhältliche OCR Engines hinsichtlich definierter Kriterien evaluiert. Die Kriterien leiteten sich aus dem in dieser Arbeit vorgegebenen Kontext der Anforderungen ab. Wesentliche Merkmale waren der Preis hinsichtlich des erhältlichen Lizenzmodelles sowie die Verwendbarkeit auf der Windows-Plattform und die Kompatibilität mit Embarcadero Delphi. OCR Engine Kandidaten wurden mit gescannten und mit Webcam-erfassten Zahlscheinen, sowohl mit OCR-Schrift bedruckt wie auch mit Handschrift ausgefüllt, getestet, um die Qualität der Ergebnisse beurteilen zu können. Für die Entwicklung dieser Arbeit wurde die Nicomsoft OCR Engine herangezogen. Die Kosten der Nicomsoft Engine betrugen im Vergleich zu den anderen untersuchten Engines nur etwa 6-10%. Unter anderem, weil die Engine kein Frontend enthält, sondern ausschließlich einen SDK. Hinsichtlich OCR liefert die Nicomsoft Engine qualitativ gleichwertige Ergebnisse wie andere teurere Engines, weist allerdings bei ICR wesentliche Schwächen auf. Diese Schwächen konnten allerdings vernachlässigt werden, da auch die anderen Engines bei der Verarbeitung von Handschriften fehlerbehaftete Ergebnisse liefern und somit in dieser Arbeit nicht eingesetzt werden können. Die ICR Funktionalitäten der Engines beruhen größtenteils auf Feedback-Schleifen. Das bedeutet, dass die falsch erkannten Zeichen vom Benutzer korrigiert werden müssen und dass diese Korrekturen an die Engine rückgemeldet werden, um einen Lernprozess zu erreichen und somit die Erkennungswahrscheinlichkeit der zukünftigen Verarbeitungen erhöhen. Da die Anforderungen allerdings beinhalten, weitgehend auf Benutzereingaben zu verzichten, wurde die Verwendung von ICR abgegrenzt.

Die Implementierung ist mit Embarcadero Delphi umgesetzt. Es handelt sich um einen Treiber für die Windows-Plattform, welcher via Named Pipes mit seinen Consumer-Applikationen wie zum Beispiel LUI, das auf dem Senioren Terminal läuft, kommuniziert. Der dynamisch konfigurierbare, modulare Aufbau von LUI erforderte eine dementsprechende Treiber-seitige Berücksichtigung, da das aktuelle Design von LUI zur Laufzeit an den Treiber übergeben wird.

Durch den Einsatz von LUI wird ein einheitliches *look and feel* für die Benutzer und Benutzerinnen erreicht, unabhängig davon, welche Applikation indirekt über LUI verwendet wird. Eine wichtige Anforderung an die Lösung war die Minimierung der Benutzereingaben, damit eine möglichst einfache und benutzerfreundliche Bedienung gewährleistet werden kann. Dieses Ziel wurde erreicht. Im fehlerfreien Fall müssen die Benutzer und Benutzerinnen nur einen Zahlschein vor der Kamera deponieren und zwei Klicks in LUI absetzen, ohne dabei textuelle Eingaben zu tätigen. Die Integration in LUI erfolgt über die von LUI zur Verfügung gestellte Funktionalität, der Konfiguration via XML-Konfigurationsdateien. Nach erfolgreicher Verarbeitung werden die extrahierten Daten in LUI angezeigt und können weiter verarbeitet werden, zum Beispiel im Zuge der Übermittlung an ein Kreditinstitut. Dies wurde parallel zu dieser Arbeit in der Diplomarbeit „Paper to Byte“ von Robert Koch exemplarisch umgesetzt [7]. Da die Benutzergruppe den Umgang mit LUI gewohnt ist und die in dieser Arbeit implementierte Lösung in LUI eingebunden werden kann, wurde zusätzlich ein erhöhtes Maß an Benutzerfreundlichkeit und Benutzertoleranz erreicht.

Die Treiberimplementierung umfasst das Erfassen einen Bildes von einer am Terminal angeschlossenen Kamera, indem über DirectShow der Windows-Treiber angesprochen wird.

Im Anschluss wird der Kontrast des erfassten Bildes durch Normalisierung (engl. *normalization*) der Bildwerte auf die gesamte verfügbare Skala von 0-255 erhöht. Weiters wird ein Gauss'scher Störfilter angewandt, um Pixelstörungen (engl. *noise*) zu verringern.

Danach wird versucht, im erfassten Bild mittels Merkmalserkennung (engl. *feature extraction*) das rote Rechteck zu erkennen, welches am Zahlschein die Überweisungsdaten enthält. Hierfür wurden die Kantendetektions-Algorithmen *thresholding*, *Sobel* und *Canny* untersucht, um aus dem Eingabebild ein binäres Kantenbild zu erstellen. Gewählt wurde *thresholding* aufgrund von Farbe und Farbsättigung, da die rote Farbe der Zahlscheine für diesen Einsatzzweck, wenn auch unter akkurater Erfassung der Blindfarbe, vorgesehen ist. Dieses Kantenbild wird dann zur Auffindung von geschlossenen Konturen herangezogen. Hierfür wird der *border following algorithm* von Suzuki und Abe verwendet. Alternativ dazu könnte die *Hough Transformation* eingesetzt werden, um Linien im Bild zu finden, welche dann im Anschluss an *Hough*, für die Auffindung des Rechteckes untersucht werden könnten. Im nächsten Schritt werden die gefundenen Konturen mit dem *Ramer Douglas Peuker algorithm* durch Konturen mit weniger Eckpunkten approximiert. Alle Konturen, welche nicht durch ein Viereck approximiert werden können, werden verworfen.

Bei erfolgreicher Auffindung des Rechteckes wird das Bild durch affine Transformation rotiert, um den Text parallel zu den horizontalen Bildrändern auszurichten.

Nach der geometrischen Transformation wird das Bild erneut gefiltert, um alle Bestandteile bis auf den maschinell-bedruckten Text zu verwerfen. Bei etwaigen Bildstörungen wird im Anschluss an die Binarisierung (engl. *binarization*) eine morphologische Öffnung (engl. *morphological opening*) auf das Bild angewandt.

Zuletzt werden die einzelnen Datenfelder des Zahlscheines mit der Nicomsoft OCR Engine verarbeitet, um die textuellen Inhalte aus dem Bild zu extrahieren. Nach erfolgreicher Verarbeitung werden die Informationen an LUI übertragen, andernfalls ein passender Fehlerhinweis.

Die Qualität der erreichten Ergebnisse ist, abhängig von den Bedingungen hinsichtlich Belichtung und Sonneneinstrahlung, nicht konstant. Die Kombination aus schlechter Belichtung und qualitativ minderwertigen Eingaben in Form von verschmutzten oder zerknitterten Zahlungsanweisungen führt zu Ergebnissen von unter 80% betreffend Vollständigkeit und Richtigkeit der erfassten Daten. Bei „normalen" Bedingungen, wie gut ausgeleuchteten Räumen sowie Verhinderung von Unter- und Überbelichtung, liegt die Erkennungsrate bei 90-100%.

Eigenschaften, welche die Qualität gegenüber den auf dem Bankensektor eingesetzten Überweisungsterminals vermindern, sind

- nicht konstante Belichtung bei der Erfassung der Belege – dies führt zu unterschiedlichen Ausgangssituationen bei der Bildaufbereitung und der Bildverarbeitung. Weiters kann das Konzept der Blindfarbe somit nicht in dieser Arbeit genutzt werden
- nicht konstante Positionierung des Beleges vor dem Bilderfassungsgerät und damit verbundene Unterschiede bei der Rastererkennung sowie der generellen Unterscheidung von Belegtypen (diese Arbeit beschränkt sich auf die Erfassung von Zahlungsanweisungen)

- die eingesetzte Hardware – in den Geräten zur Belegerfassung bei den Banken – sind hochauflösende, exakt für den Anwendungsfall konfigurierte, Scanner. Für diese Teststellung wurde eine Webcam verwendet
- *Data Mining*, welches von den Banken aufgrund der gespeicherten Daten durchgeführt werden kann. Banken können aufgrund der Relation zwischen IBAN des Empfängers und Namen des Empfängers auf etwaig falsch erfasste Zeichen rückschließen

Im Zuge der Arbeit wurde eine Studie mit Personen aus der erwarteten Benutzergruppe, nämlich den Benutzern und Benutzerinnen von LUI durchgeführt, um die Benutzerfreundlichkeit sowie die Benutzerakzeptanz abschätzen zu können. Diese Studie ergab, dass nach wie vor viele ältere Menschen vor dem Gebrauch von e-banking zurückschrecken. Die Gründe hierfür waren vielseitig. Grundlage der meisten Gründe war die Skepsis, neue, automationsgestützte Technologien zu verwenden. Die Kombination dieser Skepsis mit der Wichtigkeit von Finanzangelegenheiten lässt viele ältere Menschen davor zurückschrecken, sich mit diesen neuen Technologien auseinander zu setzen. Für die Zielgruppe der Menschen mit Einschränkungen in der Motorik ist die Lösung gut einsetzbar, da diese keine Probleme mit dem Einsatz von automationsgestützten, modernen Technologien assoziieren.

Um den Treiber in LUI integrieren und testen zu können, wurde ein LUI Design entwickelt. Im Zuge dieser Design-Entwicklung muss der Designer die Kamera-Schnittstelle sowie den Index der gewollten Auflösung konfigurieren. Um dies zu erleichtern wurde ein Tool, das Camera-Resolution-Tool entwickelt, welches alle verfügbaren Bild- und Videoaufnahmegeräte und deren angebotene Auflösungen inklusive der internen Indizes anzeigt.

Grundsätzlich konnte mit dieser Arbeit die Durchführbarkeit der Anforderungen anhand eines Prototypen gezeigt werden. Ob der Funktionsumfang einer dauerhaften, benutzerfreundlichen Nutzung genügt und ob das gewünschte Maß an Benutzerakzeptanz damit erreicht werden kann, muss anhand einer langfristigen Benutzerstudie im Live-Betrieb evaluiert werden.

7. Ausblick

In der folgenden Aufzählung wird auf Punkte eingegangen, welche nicht Gegenstand dieser Arbeit waren oder im Zuge der Arbeit abgegrenzt wurden. Weiters Anforderungen, die nicht oder nur ungenügend umgesetzt werden konnten und weitere Funktionsmerkmale, deren Umsetzung Vorteile bringen könnten.

1. Die Distanz zwischen eingesetzter Webcam und Zahlschein ist zu gering, um die Auto-Fokus-Funktion der Webcam nutzen zu können. Daher muss der Fokus manuell konfiguriert werden. Dies hat den Nachteil, dass der Fokus nicht mehr stimmt, wenn die Entfernung zwischen Kamera und Zahlschein verändert wird. Mögliche Lösungen wären das Austauschen der Kamera oder das Entwickeln einer (verschließbaren) Vorrichtung, um die zu verarbeitenden Belege immer an der gleichen Position mit der gleichen Belichtung und dem gleichen Abstand zur Kamera zu erfassen. Dies hätte sowohl den Vorteil einer konstanten Bildqualität wie auch konstante Koordinaten bei der Rastererkennung des am Beleg abgebildeten Layouts.

2. Der automatische Weißabgleich der Kamera funktioniert nicht konstant. Dies führt zu Helligkeitsschwankungen sowie Farbschleiern bei den erfassten Bildern. Zudem funktioniert der Weißabgleich nach verschiedensten Beleuchtungssituationen zum Teil überhaupt nicht mehr, die erfasten Bilder sind dann mit einem blauen oder einem roten Schleier überzogen, abhängig von der Einstellung des Weißabgleichs. Dieser muss dann erneut kalibriert werden. Mögliche Lösungen wären auch hier die Ansätze aus Punkt 1.

3. Die unterschiedlichen Beleuchtungssituationen wie „wenig bis kein Licht“, Raumlicht, zusätzliche Beleuchtung am Standort der Kamera, Lichtabstrahlung vom Monitor, Tageslicht und Sonnenlicht führen zu unterschiedlichen Helligkeits- und Kontrastmerkmalen der erfassten Bilder. Dies verhindert eine konstante Definition von Werten zur Filterung der Bilder nach rotem Rahmen und schwarzer Schrift. Dies führt wiederum zu schlechteren Ergebnissen bei der Merkmalserkennung und verhindert die Textextraktion entweder ganz oder verschlechtert die Korrektheit der Ergebnisse. Im Zuge der Arbeit wurde versucht, von den Helligkeits-, Kontrast-, Farb- und Sättigungswerten der erfassten Bilder auf Filterwerte zu schließen, die konstante Ergebnisse erreichen. Dies konnte nicht zu 100% schlüssig umgesetzt werden. Obwohl die Filter einen großen Teil an Anwendungsfällen verarbeiten können, bleiben Fälle, in denen die Verarbeitung nicht oder mit zu wenig Genauigkeit funktioniert. Mögliche Alternativen wären die Verbesserung der Filter-Algorithmen bzw. die Verbesserung der Parameterdefinition für die Filter und die Entwicklung einer Vorrichtung (siehe Punkt 1.).

4. Unterschiedliche Beleuchtungssituation in einem erfassten Bild, zum Beispiel wenn der Zahlschein beim Ablichten zur Hälfte in der Sonne liegt, können von diesem Treiber nicht verarbeitet werden. Eine mögliche Lösung hierfür wäre ebenfalls die Konstruktion

einer Vorrichtung (siehe Punkt 1.).

5. Um die OCR Verarbeitung zu verbessern, könnten in Zukunft alternative Engines oder Algorithmen eingesetzt werden.

6. Die in dieser Arbeit verarbeiteten Dokumente müssen mit Maschinenschrift bedruckt sein, damit die gewünschten Informationen extrahiert werden können. In einer Weiterentwicklung dieses Prototypen kann ICR eingebaut werden, um auch handschriftlich ausgefüllte Dokumente verarbeiten zu können. Hierfür sind allerdings zusätzliche Algorithmen und Technologien notwendig. Unter anderem kann für ICR die Entwicklung von neuronalen Netzen eingesetzt werden. Eine andere Möglichkeit ist die Umsetzung eines Lernmechanismus, um die Erkennung der Benutzer-Handschriften in der ICR Engine zu verbessern. Hierfür müsste allerdings nach einem eigenen Konzept geforscht werden, da die aktuellen Schnittstellen der Engines hinsichtlich des Lernprozesses mehrere Benutzereingaben benötigen, dies allerdings durch die Anforderungen dieser Arbeit ausgeschlossen wird.

7. Diese Teststellung beschränkt sich auf die Verarbeitung von Zahlungsanweisungen. Zukünftig könnte dieser Prototyp hinsichtlich der Erkennung und Verarbeitung von anderen Bankbelegen erweitert werden.

8. Um wiederkehrende Empfänger von Zahlungsanweisungen schneller verarbeiten zu können, beziehungsweise Fehlerquellen auszuschließen, könnte in zukünftigen Weiterentwicklungen eine Historie über die verarbeiteten Informationen geführt werden. Um Datensicherheit zu gewährleisten, müsste allerdings Verschlüsselung eingesetzt werden. Damit sind jedoch zusätzliche Aufwände für Schlüsselmanagement und Wartung der eingesetzten Verschlüsselung notwendig, um zu gewährleisten, dass die eingesetzte Verschlüsselung dem Stand der Technik entspricht und somit nur sehr unwahrscheinlich kompromittiert werden kann.

9. Um die praxisnahe Einsatzfähigkeit der Lösung sowie die Benutzerakzeptanz zu evaluieren, müsste eine langfristige Studie hinsichtlich Einsatz der Lösung im Live-Betrieb durchgeführt werden.

10. Um die extrahierten Informationen weiterverarbeiten zu können, muss eine Schnittstelle entwickelt werden, welche unter Kooperation mit Kreditinstituten die Anbindung an die jeweiligen Bankensysteme ermöglicht. Hauptaugenmerk muss hierbei auf Datenschutz, Verschlüsselung und die jeweilig einzuhaltenden Gesetze, Normen und Richtlinien gelegt werden. Parallel zu dieser Arbeit wurde von Robert Koch in der Arbeit „Paper to Byte“ eine Weiterverarbeitung der Daten unter Nutzung des DOM (Document Object Model) einer Webseite erstellt [7].

11. In LUI könnte die Verwendung des zu diesem Prototypen parallel entwickelten Belegdruckers konfiguriert werden.

Abkürzungsverzeichnis

AAT	Zentrum für Angewandte Assistierende Technologien
ANN	Artificial Neural Networks
BBAN	Basic Bank Account Number
BIC	Business Identifier Code
COM	Component Object Model
CV	Computer Vision
DLL	Dynamic Links Library
DOM	Document Object Model
DPI	Dots Per Inch
GNU	GNU's Not Unix (rekursives Akronym)
HSV	Hue Saturation Value
HTTPS	Hyper Text Transfer Protocol Secure
HWR	Handwriting Recognition
IBAN	International Bank Account Number
ICR	Intelligent Character Recognition
LUI	Hochflexible multimodale Benutzerschnittstelle
IWR	Intelligent Word Recognition
KNN	Künstliche Neuronale Netze
LUI	Lokales Benutzerterminal (engl. Local User Interface)
MICR	Magnetic Ink Character Recognition
OCR	Optical Character Recognition
OMR	Optical Mark Recognition
PIN	Personal Identifier Number
QR	Quick Response
RAM	Random Access Memory
REST	Representational State Transfer
RGB	Rot Grün Blau (engl. Red Green Blue)
S€PA	Single Euro Payments Area
SDK	Software Development Kit
SISI	Senioren Terminal
STUZZA	Studiengesellschaft für Zusammenarbeit im Zahlungsverkehr

TCP	Transmission Control Protocol
TU	Technische Universität
UI	User Interface
WIA	Windows Image Acquisition
XML	Extensible Markup Language
XSD	XML Schema Definition

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[1] Statistika GmbH, „Statistika - Das Statistik Portal," [Online]. Available: http://de.statista.com/statistik/daten/studie/298276/umfrage/internetnutzer-in-oesterreich-nach-zielgruppen/. [Zugriff am 07 2016].

[2] Bundeskanzleramt Österreich, „Bundeskanzleramt Österreich," [Online]. Available: https://www.bka.gv.at/DocView.axd?CobId=49806. [Zugriff am 07 2016].

[3] Statistik Austria, „Statistik Austria - der Informationsmanager," [Online]. Available: http://www.statistik.at/web_de/statistiken/energie_umwelt_innovation_mobilitaet/informationsgesellschaft/ikt-einsatz_in_haushalten/index.html. [Zugriff am 07 2016].

[4] Statistika GmbH, „Statistika GmbH - Online Banking Nutzung," [Online]. Available: http://de.statista.com/statistik/daten/studie/431727/umfrage/nutzung-des-internets-fuer-online-banking-in-oesterreich/. [Zugriff am 07 2016].

[5] AAT, TU Wien, „AAT TU Wien - ISUI Homepage," [Online]. Available: http://www.aat.tuwien.ac.at/isui.index.html. [Zugriff am 14 07 2014].

[6] Logitech, „Logitech HD Pro Webcam C920 - Technische Daten," Logitech, [Online]. Available: http://www.logitech.com/de-at/product/hd-pro-webcam-c920?crid=34. [Zugriff am 14 7 2014].

[7] R. Koch, *Paper to Byte – Vom Zahlschein zum Online Banking – Ein Konzept für ältere Erwachsene,* TU Wien, 2016.

[8] AAT, TU Wien, *IS-UI Configuration Manual,* Vienna University of Technology - Institute "integrated study", 2001.

[9] Erste Bank der oesterreichischen Sparkassen AG (Erste Bank Oesterreich), „Erste Bank der oesterreichischen Sparkassen AG (Erste Bank Oesterreich)," [Online]. Available: https://www.sparkasse.at/noe/Downloads/255598b0-7b01-4a8d-a87e-73ed0c07aa10/sepa-n-zahlungsanweisung.pdf. [Zugriff am 11 2013].

[10] W. Ijsseltsteijn, H. Nap, Y. d. Kort und K. Poes, „Digital Game Design for Elderly Users," *FuturePlay,* 2007.

[11] A. D. Fisk, W. A. Rogers und N. Charness, „Designing for Older Adults: Principles and Creative Human Factors Approaches (Human Factors & Aging)," 2009.

[12] J. t. Voort, J. Radstaat, M. Douma, L. Clarijs und R. A. u. S. Shahid, „Social Engagement in Elderly Care Homes: Towards Designing an Application to Reduce Social Loneliness," *HCII 2015 Posters, Part I,* pp. 327-333, 2015.

[13] S. P. Marshall, „Interaction Science and the Aging User: Techniques to Assist in Design and Evaluation," *UAHCI/HCII 2013, Part II,* pp. 133-141, 2013.

[14] E. Sciarretta, A. Ingrosso, V. Volpi und A. O. u. R. Grimaldi, „Elderly and Tablets: Considerations and Suggestions About the Design of Proper Applications," *ITAP 2015, Part I,* pp. 509-518, 2015.

[15] C. Jian, H. Shi, F. Schafmeister, C. Rachuy, N. Sasse, H. Schmidt, V. Hoemberg und N. v. Steinbüchel, „Touch and Speech: Multimodal Interaction for Elderly Persons," *BIOSTEC,* pp. 385-400, 2012.

[16] D. Williams, M. A. U. Alam, S. I. Ahamed und W. Chu, „Considerations in designing human-computer interfaces for elderly people," in *13th International Conference on Quality Software*, 2013.

[17] A. Holzinger, „Finger Instead of Mouse: Touch Screens as a Means of Enhancing Universal Access," *User Interfaces for All,* pp. 387-397, 2003.

[18] A. Vasconcelos, P. A. Silva, J. Caseiro und L. F. Teixeira, „Designing tablet-based games for seniors: the example of CogniPlay, a cognitive gaming platform," in *4th International Conference on Fun and Games*, 2012.

[19] S. Harada, D. Sato, H. Takagi und C. Asakawa, „Characteristics of Elderly User Behavior on Mobile Multi-touch Devices," *INTERACT,* pp. 323-341, 2013.

[20] F. Arab, Y. Malik und B. Abdulrazak, „Evaluation of PhonAge: An Adapted Smartphone Interface for Elderly People," *INTERACT,* pp. 547-554, 2013.

[21] G. Pavlovic, *Online-Banking-Belegdrucker für Senior/-innen,* TU Wien, 2014.

[22] C. Eisserer, *Technische Umsetzung eines Telebanking-Belegdurckers für Senioren/-innen,* TU Wien, 2016.

[23] K. Gebhart, *Zurück zum Papier,* TU Wien, 2016.

[24] Futurezone Technology News, „futurezone.at," 11 03 2014. [Online]. Available: http://futurezone.at/digital-life/senioren-wollen-keine-seniorenhandys/55.444.449. [Zugriff am 2014].

[25] S. Singh, „Optical Character Recognition Techniques: A Survey," *Journal of Emerging Trends in Computing and Information Sciences,* 06 2013.

[26] Kae, Huang, Doersch und Learnded-Miller, „Improving State-of-the-Art OCR through High-Precision Document-Specific Modelung," University of Massachusetts Amherst, 2010.

[27] S. Sushruth, G. Gunasheela, D. Thejus, D. S. Vinay und R. R. Sudhir, „"i" - A novel algorithm for Optical Character Recognition (OCR)," Bangalore, Indien, 2013.

[28] A. Kaw und E. Learned-Millter, „Learning on the Fly: Font-Free Approaches to Difficult OCR Problems," Dept. of Computer Science, University of Massachusetts, Amherst MA, 2009.

[29] J. Pradeep, E. Srinivasan und S. Himavathi, „Neural Network based Handwritten Character Recognition system without feature extraction," in *International Conference on Computer, Communication and Electrical Technology, ICCCET*, Indien, 2011.

[30] V. Patil und S. Shimpi, „Handwritten English character recognition using neural network," *Computer Science and Engineering,* 2011, Ausgabe 41.

[31] T. Wang, D. J. Wu, A. Coates und A. Y. Ng, „End-to-End Recognition with Convolutional Neural Networks," in *21st International Conference on Pattern Recognition (ICPR 2012)*, Tsukubqa, Japan, 2012.

[32] J. Pradeep, E. Srinivasan und S. Himavathi, „Diagonal based Feature Extraction for Handwritten Alphabets Recognition System using Neural Network," *International Journal of Computer Science & Information Technology (IJCSIT),* 20 2011.

[33] D. C. Ciresan, U. Meier, L. M. Gambardella und J. Schmidhuber, „Convolutional Neural Network Committees For Handwritten Character Classification," in *International Conference on Document Analysis and Recognition*, 2011.

[34] S. Barve, „Optical Character Recognition Using Artificial Neural Network," *International Journal of Advanced Research in Computer Engineering & Technology,* 04 06 2012.

[35] Volksbank, „Mobile Banking App," [Online]. Available: https://www.volksbank.at/private/electronic-banking/mobile-banking. [Zugriff am 08 2016].

[36] Bank Austria, „Mobile Banking App," [Online]. Available: http://bankaustria.at/mobilebankingapp/#feature-6. [Zugriff am 08 2016].

[37] Futurezone Technology News Österreich, „QR-Codes auf Rechnungen machen Erlagscheine obsolet," *Futurezone Technology News Österreich,* 26 12 2013.

[38] STUZZA, „Zahlungsverkehr mit QR-Codes," [Online]. Available: https://www.stuzza.at/de/zahlungsverkehr/qr-code.html. [Zugriff am 08 2016].

[39] Raiffeisen, „Raiffeisen ELBA-App," [Online]. Available: http://www.raiffeisen.at/oesterreich/1006622331426_1006623304603_100662484086 9-1006624840869-NA-30-NA.html. [Zugriff am 08 2016].

[40] Sparkasse, „E-Banking Services, Scan & Pay," [Online]. Available: https://netbanking.sparkasse.at/hilfe/e-banking-services/mobiles-netbanking-app/ScanAndPay. [Zugriff am 08 2016].

[41] Bawag P.S.K., „Mobile App Funktionen," [Online]. Available: https://www.bawagpsk.com/BAWAGPSK/PK/ebanking/120210/ebanking-mobile.html. [Zugriff am 07 2016].

[42] Amazon.de, „Amazon - Sony Xperia go," [Online]. Available: https://www.amazon.de/Sony-Xperia-GO-ST27i-Smartphone/dp/B00MOIE5PC/ref=sr_1_3?ie=UTF8&qid=1470743046&sr=8-3&keywords=sony+xperia+go. [Zugriff am 08 2016].

[43] Amazon.de, „Amazon - Sony Xperia Z3 compact," [Online]. Available: https://www.amazon.de/Sony-Compact-Smartphone-Touch-Display-Speicher/dp/B00N9O8NPS/ref=sr_1_1?s=telephone&ie=UTF8&qid=1470743133&sr=1-1&keywords=sony+xperia+z3+compact. [Zugriff am 08 2016].

[44] Amazon.de, „Amazon - Samsung Tab3," [Online]. Available: https://www.amazon.de/Samsung-interner-Speicher-Megapixel-

Android/dp/B00DIYS93A/ref=sr_1_1?s=ce-de&ie=UTF8&qid=1470743176&sr=1-1&keywords=samsung+tab+3. [Zugriff am 08 2016].

[45] Bawag P.S.K., „"Scan & Transfer" und IBAN-Scan,“ [Online]. Available: https://www.bawagpsk.com/linkableblob/BAWAGPSK/270426/882ae30ec080c951bb298226bd87a80f/scanner-phone-tablet-data.pdf. [Zugriff am 07 2016].

[46] OCR Systeme GmbH, „OCR Systeme GmbH - Glossar,“ [Online]. Available: http://www.ocr-systeme.de/ocrallg.htm. [Zugriff am 08 2016].

[47] Ubuntu, „OCRFeeder,“ [Online]. Available: https://apps.ubuntu.com/cat/applications/precise/ocrfeeder/ . [Zugriff am 05 2014].

[48] T. Breuel, „OCRopus,“ [Online]. Available: https://github.com/tmbdev/ocropy . [Zugriff am 05 2014].

[49] screenworm.de, „screenworm,“ [Online]. Available: http://www.screenworm.de/. [Zugriff am 08 2016].

[50] OnBase by Hyland, „AnyDoc Software,“ [Online]. Available: https://www.onbase.com/de-DE/uber-hyland/akquisitionen/anydoc-software#.V6sJ2fmLSUk. [Zugriff am 08 2016].

[51] free-ocr.com, „FreeOCR,“ [Online]. Available: http://www.free-ocr.com/de.html. [Zugriff am 05 2014].

[52] GOCR, [Online]. Available: http://jocr.sourceforge.net/. [Zugriff am 05 2014].

[53] Microsoft, „Microsoft Office Document Imaging,“ [Online]. Available: https://support.microsoft.com/de-at/kb/982760. [Zugriff am 05 2014].

[54] Microsoft, „Microsoft Office OnNote 2007,“ [Online]. Available: https://support.office.com/de-de/article/Einf%C3%BChrung-in-Microsoft-Office-OneNote-2007-cd06e378-87e2-45b6-96af-7b384599f3d4 . [Zugriff am 05 2014].

[55] Lexmark (ehemals Readsoft), „Readsoft,“ [Online]. Available: http://www.lexmark.com/en_us/better-together.html . [Zugriff am 08 2016].

[56] Scantron, „Scantron,“ [Online]. Available: http://www.scantron.com/. [Zugriff am 08 2016].

[57] SimpleSoftware, „SimpleOCR,“ [Online]. Available: http://www.simpleocr.com/. [Zugriff am 05 2014].

[58] Musitek, „SmartScore,“ [Online]. Available: http://www.musitek.com/. [Zugriff am 08 2016].

[59] Google Inc., „Google Tesseract-OCR project,“ [Online]. Available: http://code.google.com/p/tesseract-ocr/. [Zugriff am 05 2014].

[60] Google Inc., „Google Tesseract-OCR Issues,“ [Online]. Available: http://code.google.com/p/tesseract-ocr/issues/detail?id=88. [Zugriff am 05 2014].

[61] Cognitive Technologies, „Cognitive Forms - Cuneiform,“ [Online]. Available: http://cognitiveforms.com/products_and_services/cuneiform.

[62] MeOCR, „MeOCR,“ [Online]. Available: http://www.meocr.com/index.php. [Zugriff am 05 2014].

[63] PUMA.NET open source project, „Codeplex Puma.NET project,“ [Online]. Available: http://pumanet.codeplex.com/. [Zugriff am 05 2014].

[64] Dynamsoft, „Dynamsoft Dynamic .NET TEAIN,“ [Online]. Available: http://www.dynamsoft.com/Products/.Net-TWAIN-Scanner.aspx. [Zugriff am 05 2014].

[65] GNU.org, „Ocrad - The GNU OCR,“ [Online]. Available: http://www.gnu.org/software/ocrad/. [Zugriff am 05 2014].

[66] Abbyy, „Abbyy FineReader OCR SDK,“ [Online]. Available: http://www.abbyy.de/ocr-sdk-windows/ocr-schritte/. [Zugriff am 05 2014].

[67] Asprise, „Asprise OCR and Barcode Recognition,“ [Online]. Available: http://asprise.com/royalty-free-library/c-c++-delphi-ocr-for-windows-mac-linux-download.html. [Zugriff am 05 2014].

[68] LEAD Technologies Inc., „LEADTOOLS,“ [Online]. Available: https://www.leadtools.com/sdk/professional-ocr.htm. [Zugriff am 05 2014].

[69] Nicomsoft, „Nicomsoft OCR SDK - List of Features,“ [Online]. Available: http://www.nicomsoft.com/products/ocr/features/. [Zugriff am 05 2014].

[70] Nuance, „Nuance OmniPage Capture SDK,“ [Online]. Available: http://www.nuance.de/for-business/by-product/omnipage/csdk/index.htm. [Zugriff am 05 2014].

[71] Exper-OCR Inc., „OpenRTK - ExperVision OCR SDK,“ [Online]. Available: http://www.expervision.com/ocr-sdk-toolkit/openrtk-ocr-toolkit-sdk#Introduction. [Zugriff am 06 2014].

[72] Recogniform Technologies, „Recogniform Technologies - Products,“ [Online]. Available: http://www.recogniform.com/products.htm. [Zugriff am 05 2014].

[73] Abby, *Preisauskunft,* Mai 2014.

[74] G. A. Stephen, String Searching Algorithms, 1994.

[75] Microsoft, „Microsoft Developer Network - Named Pipes,“ Microsoft, [Online]. Available: https://msdn.microsoft.com/en-us/library/windows/desktop/aa365590%28v=vs.85%29.aspx. [Zugriff am 12 2013].

[76] M. Braun, „DirectX Delphi WebCam Capture,“ [Online]. Available: http://www.delphibasics.info/home/delphibasicsprojects/directxdelphiwebcamcapturexample. [Zugriff am 11 2014].

[77] A. S. Tanenbaum und B. Herbert, „Modern Operating Systems,“ Pearson, 2015, p. 97.

[78] simdesign, „NativeXml: A native Delphi XML parser and writer,“ [Online]. Available: http://www.simdesign.nl/nativexml.html. [Zugriff am 21 2 2014].

[79] M. G. Laentir Valetov, „Project Delphi-OpenCV. Translation of OpenCV library header files in Delphi,“ [Online]. Available: https://github.com/Laex/Delphi-OpenCV. [Zugriff am 08 12 2015].

[80] A. Mukherjee und S. Kanrar, „Enhancement of Image Resolution by Binarization,“ International Journal of Computer Applications, 2010.

[81] D. R. W. &. R. S. Ochawar, „Overview on Edge Detection Methods,“ in *International Conference on Electronic Systems, Signal Processing and Computing Technologies*, 2014.

[82] OpenCv, „OpenCV - Sobel derrivates,“ [Online]. Available: http://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/sobel_derivatives/sobel_derivatives.html. [Zugriff am 05 2016].

[83] OpenCv, „OpenCv - Canny Algorithmus,“ [Online]. Available: http://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/canny_detector/canny_detector.html. [Zugriff am 04 2016].

[84] J. Canny, „A Computational Approach to Edge Detection,“ in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1986.

[85] OpenCv, „OpenCv - Canny Tutorial,“ [Online]. Available: http://docs.opencv.org/trunk/da/d22/tutorial_py_canny.html#gsc.tab=0. [Zugriff am 08 2016].

[86] N. Stamatopoulos, B. Gatos, I. Pratikakis und S. J. Perantonis, „Goal-Oriented Rectification of Camera-Based Document Images,“ *IEEE Transactions on Image Processing,* 04 04 2011.

[87] N. Aggarwal und W. C. Karl, „Line Detection in Images Through Regularized Hough Transform,“ *IEEE Transactions on Image Processing,* 03 2006.

[88] M. Dubská, J. Havel und A. Herout, *Real-Time Detection of Lines using Parallel Coordinates and OpenGL,* Brno University of Technology: 2013 ACM - Association for Computing Machinery, Inc., 2011.

[89] Y. Zhu und Z. Qingzhi, „Rectangle Detection by the Chain-code Tracing,“ IEEE, Nan Yang, Ching, 47300, 2011.

[90] K. A. Satoshi Suzuki, „Topological Structural Analysis of Digitized Binary Images by Border Following,“ *Computer Vision, Graphics, and Image Processing,* pp. 32-46, 1985.

[91] T. P. David Douglas, „Algorithms for the reduction of the number of points required to represent a digitized line or its caricature,“ *The Canadian Cartogrpher,* pp. 112-122, 1973.

[92] Nicomsoft, *Nicomsoft OCR: Developer's Guide,* (c) 2009-2013.

[93] IBAN.de, „IBAN.de - Prüfsummenberechnung,“ [Online]. Available: https://www.iban.de/iban-pruefsumme.html. [Zugriff am 10 2014].

a. Anhang

Im Anhang sind folgende Informationen enthalten:

a.1. Design des Treibers (Klassendiagramm)

a.1. Design des Treibers (Klassendiagramm)