

Lib-Indexer

EDBV WS 2019/2020: AG_C_3

Anand Eichner (11808244)

Laurenz Edmund Fiala (11807869)

Anna Nieto-Berezhinskaya (01223066)

Aleksandar Vucenovic (01635282)

6. Januar 2020

1 Gewählte Problemstellung

1.1 Ziel

Das Projekt soll Bücher in einem Bücherregal erkennen, in Bücher-Koordinaten umwandeln und nach ihrem Label abspeichern.

1.2 Eingabe

JPG-Bild eines Bücherregals mit Büchern, auf denen eindeutige TU-Bibliotheks-Labels (schwarz-auf-weiß) kleben.

1.3 Ausgabe

visuell:



Abbildung 1: Input-Bild

textuell:

Strukturierte Klartext-Datei mit Inhalt:

- Standort der Bücher (in Büchern zum Ursprung - links-oben)
- Vier Pixel-Vektoren, die ein Label in einem Viereck umschließen

1.4 Voraussetzungen und Bedingungen

- Die Bücher müssen gerade (+/- 5°) stehen.
- Das Bild darf nicht mehr als 30° von der Waagrechte abweichen.
- Das Bild muss eine für die Texterkennung der Labels ausreichende Auflösung aufweisen (Abhängig von der Entfernung).
- Das Bild muss farbig sein.
- Das Bild muss ausreichend hell sein.
- Ein Weißabgleich muss durchgeführt worden sein.

Neue Anforderungen:

- Der Hintergrund der Labels muss weiß sein (Intensität muss über dem globalen Otsu-Threshold liegen)

1.5 Methodik

Methodik- Pipeline

1. Hough-Transformation
Finden der Regalfächer zum Korrigieren der Perspektive
2. Perspektivenkorrektur
tuwe Mittels Transformationsmatrix aus HT berechnet

3. Eckenerkennung
Finden der Ecken von Labels
4. Integral imaging
Finden von Labels innerhalb eines Akzeptanzbereichs, es werden nur Bereiche zwischen verschiedenen, zuvor erkannten, Ecken überprüft
5. Eigene Heuristik
Einordnen von Labels in Buch-Koordinaten
6. Optical Character Recognition
Erkennen von Text auf den Labels in den zuvor erkannten Bereichen

1.6 Evaluierungsfragen

Interaktion zwischen den Komponenten:

- *Werden die Regalfächer korrekt erkannt?*
Voraussetzungen: Seite der Regalfächer, die zur Kamera zeigt, ist schwarz.
Ergebnis: An jedem Fach liegt eine Gerade an.
- *Wird die Perspektive korrekt angepasst?*
Voraussetzungen: Korrekte Geraden der Regalfächer.
Ergebnis: Bücher und Labels sind im Bild weitestgehend rechteckig.
- *Werden alle Labels erkannt?*
Voraussetzungen: Perspektivenkorrigiertes Bild
Ergebnis: Bounding-Boxes der gefundenen Labels
- *Sind die Bounding Boxes der gefundenen Labels korrekt? Ist der gesamte Text darin enthalten?*
Voraussetzungen: Korrekt erkanntes Label oder ein false-positive.
Ergebnis: Vier Vektoren, die den gesamten Text umschließen (und nicht mehr). Bei false-positives ist das Ergebnis nicht relevant, jedoch sollte es nicht zu groß sein (z.B. das gesamte Bild überdecken).
- *Werden die Labels korrekt in Bücher-Koordinaten umgewandelt?*
Voraussetzungen: Bounding Boxes der Labels sind korrekt.
Ergebnis: Bücher-Koordinatensystem als 2D-Array mit Ursprung links-oben.
- *Werden die Labels der TU-Bibliothek korrekt gelesen und in Text umgewandelt?*
Voraussetzungen: Label Bounding-Boxes wurden korrekt berechnet (enthalten keinen unnötigen Text).
Ergebnis: String-Repräsentation des Labels. Erwartete Korrektheit: > 90% für typische Datensätze.

- *Wird die Wahrscheinlichkeit der Label-Korrektheit angemessen berechnet?*
 Voraussetzungen: Korrekt in Text umgewandeltes Label.
 Ergebnis: Floating-point Wert im Intervall [0, 1]. Alle Labels mit Wahrscheinlichkeit 0 wurden entfernt.

1.7 Zeitplan

Meilenstein	abgeschlossen am	Arbeitsaufwand in h
Prototyp	10.11.	30
Hough-Transformation	17.11.- 20.12.	50 2
Perspektivenkorrektur	17.11.- 23.12.	6 20
Labelerkennung	25.11.- 14.11.	25
Labelerkennung - Verbesserungen	25.11.- 5.1.	30
Labels in Buch-Koordinaten	1.12.- 4.1.	4 0.2
Optical Character Recognition	10.12.- 3.1.	110 40
Labels filtern	15.12. nicht abgeschl.	6 nicht abgeschl.
Daten in Output-Format umwandeln	5.1.	4 0
Tests	18.12.- 5.1.	5
Evaluierung	20.12. 5.1.	5 4

2 Arbeitsteilung

Name	Tätigkeiten
Anna Nieto-Berezhinskaya	Harris Eckendetektor (zu feinfühlig und kann deshalb nicht eingesetzt werden)
Anand Eichner	houghTransform (+ lokale Funktionen), PerspectiveCorrection (+ lokale Funktionen),
Laurenz Edmund Fiala	label_detection (+ lokale Funktionen)
Aleksandar Vucenovic	ocr (+ locals), NCC, SSD_naive (+ lokale Funktionen) preprocessing (+ lokale Funktionen)

3 Methodik

3.1 Perspektivenkorrektur

Im Vorverarbeitungsschritt wird das Eingabebild in ein Graustufenbild umgewandelt und mit einem vertikalen und einem horizontalen Sobelfilter convoluted um Kantenbilder zu erhalten, dann wird der Betrag dieser Werte genommen um Kanten in beide Richtungen zu finden. Um dem Rauschen im Bild entgegenzuwirken werden alle verbundobjekte unter einer bestimmten Größe ausgefiltert um nur lange Kanten zu behalten. Bei den Vertikalen werden zusätzlich nur die 20 größten Verbundobjekte ausgewählt um dem Rauschen durch die vielen Vertikalen entgegenzuwirken.

Die beiden Kantenbilder werden mit hilfe einer Hough-Transformation in den Parameterraum transformiert. Von dort werden folglich einige Maxima ausgelesen um einige Geraden im Bild zu ermitteln.

Aus den Linienscharen werden jeweils zwei Vertikale und zwei Horizontale Ausgewählt um ein Rechteck im Raum darzustellen. Die vier Eckpunkte dieses Rechtecks werden vor der geometrischen Verzerrung perspektivisch Korrekt hochskaliert um die unnötige erstellung redundanter Daten zu verhindern. Mit den skalierten Eckpunkten und den eigentlichen Eckpunkten des Bildes wird nun eine perspektivische Transformationsmatrix gebildet und auf das Bild angewendet.

3.2 Optical Character Recognition

Image Thresholding (Adaptive)

Das Binarizing des Labels fungiert als Noise Reduktion, zusätzlich ermöglicht es Template Matching mit weiß/schwarzen Bitmaps, da es ein schwarz/weiß Bild zurückliefert.

Dilation

Dilation ermöglicht eine korrekte Regionproperties Funktion, da es die Löcher in den Components füllt, und diese damit sauber verbindet.

Regionprops

Regionprops gibt uns die Centroids und Bounding boxes von collected components, die in unserem Fall potentielle character sind. Die Methode wird benutzt um die Charakter zu segmentieren, in dem man das Originalbild mit den Koordinaten der Bounding Boxes cropped. Zustätzlich verschafft es uns eine Sortiermethode, mithilfe der Centroids können wir die Wörter nämlich richtig von links oben nach rechts unten sortieren, da Centroids die XY-Koordinaten der Mitte eines Buchstabens im Bild liefert.

Prewitt Edge-Detection

Prewitt Edge-Detection wird als preprocessing step für die Dilation und Hough Transformation genutzt, da diese ein Kantenbild als Eingabe brauchen und Prewitt dieses liefert.

Normalized Cross Correlation (Template Matching) [3]

Der nccor/NCC Algorithmus ist ein sehr weit verbreiteter Algorithmus für das Template Matching. In Matlab selbst ist er als normxcorr2-Funktion durchführbar. Er wird verwendet, um die Korrelation zwischen template und original image zu finden, und normalisiert um richtige Ergebnisse unabhängig von Brightness-Unterschieden zu liefern. Das Ergebnis ist eine Korrelationsmatrix, die Werte zwischen -1 und 1 enthält, wobei 0 die kleinste Korrelation repräsentiert und 1 die größte (-1, falls das Original oder Template negiert ist). Wir suchen das globale Maxima in der Matrix, da es den besten Korrelationswert liefert, und vergleichen diesen mit allen Werten der Templates. Das Template mit dem größten Korrelationswert wird schließlich als matched character

weitergegeben.

Sum of Squared Differences (Template Matching)

Der SSD Algorithmus wird auch, wie der nxcorr2-Algorithmus, zum Template Matching verwendet. Der SSD Algorithmus beruht darauf, dass, wenn man ein Template über das Originalbild legt, und die Summe der quadrierten Differenzen der Pixelwerte bildet, ein möglichst kleiner Wert bei einem Match herauskommen sollte (da sich die Werte sehr ähnlich sein müssen). Wir haben eine naivere Variante implementiert, da unser Template und Bild immer gleich groß sind und immer beide Bilder die Buchstaben zur Gänze umschließen. Daher müssen wir keine Regionen absuchen undOffsets ausprobieren.

3.3 Label Detection

Um die Labels zu erkennen, haben wir uns aufgrund der verschiedenen Label-Dimensionen am Anfang des Projekts dazu entschieden, mittels Integral Imaging (Summed Area Table) zu bestimmen, ob ein Bereich als Label gelten kann oder nicht.

Der Prototyp war auch recht erfolgversprechend. Jedoch sind wir im Verlauf des Projekts auf einige Limitationen bei dieser Herangehensweise gestoßen.

Es hat sich herausgestellt, dass es oft nicht ganz trivial ist, bei verschiedenen Auflösungen die korrekten Ecken mit dem Harris-Eckendetector zu finden. Hier spielt das Preprocessing, welches ein einfacher Gaußfilter ist, eine große Rolle. Auch die Größe des Gauß-Konvolutionskernels für den Harris-Detektor ist wichtig. Aufgrund von Zeitmangel gegen Ende des Projekts, konnten diese Werte leider nicht mehr so genau bestimmt werden und deshalb ist das Ergebnis leider nicht so genau, wie eigentlich möglich bzw. gewünscht.

Harris-Eckendetektor [1]

Der Harris-Eckendetektor wird genutzt, um die Ecken der einzelnen Labels zu erkennen. Ursprünglich war geplant, diese Methode selbst zu implementieren, jedoch war die selbst implementierte Variante zu sensibel und das konnte aufgrund von fehlenden Projektmitgliedern nicht mehr ausgebessert werden.

Summed Area Table / Integral Imaging [2]

Das Integral Imaging wird benutzt, um die Intensität der einzelnen Label-Kandidaten effizient berechnen zu können. Im Programmverlauf wird das genutzt, um das Verhältnis von dunklen zu hellen Bildabschnitten zu berechnen und damit bestimmen zu können, ob ein Bereich als Label gelten kann oder nicht.

Otsu-Threshold [4]

Es wird ein globaler Threshold benötigt um Labels vom Hintergrund abzugrenzen. Das Integral Image beruht auf dem Ergebnis der Bildmaske dieses Thresholds. Ein Otsu-Threshold eignet sich sehr gut um dynamisch in jedem Bild einen akzeptablen Threshold zu ermitteln.

Der Threshold wird auch lokal pro Label-Kandidat angewandt, um den oft sehr hellgrauen Text vom Hintergrund (dem Label) abzugrenzen. Damit ist das Ergebnis des Programms wesentlich zuverlässiger.

4 Implementierung

4.1 Main

Um das Framework auszuführen muss main.m mit dem Pfad zum Bild als einziges Argument ausgeführt werden. Die main Funktion gibt dann den json String zurück.

Beispelaufruf: main('Dataset/01.jpg');

4.2 Perspektivenkorrektur

Für die Perspektivenkorrektur ist die PerspectiveCorrection.m Datei zuständig. Diese nutzt einige Builtins für das Preprocessing und eine selbstgeschriebene Implementierung der Hough-Transformation (houghTransform.m). Zum Skalieren der Eckpunkte wurden einige Funktionen implementiert um geometrische Operationen mit Linien in Parameterform durchzuführen. Bei der Hough-Transform wird nur ein Wertebereich von -30 bis +30 Grad genutzt um den Parameterraum möglichst klein zu halten. Weiters wird nur nach 5 Maxima gesucht, dies bietet ein gutes Mittelding zwischen schlechten Ergebnissen und zu wenigen Ergebnissen. Es werden bei den Vertikalen Kanten die jeweils 20 Größten behalten dies bietet auch hier wieder ein gutes Mittelding zu viel Rauschen und einer zu geringen Chance eine gute Kante zu erhalten.

4.3 Label Detection

Die Erkennung der Labels wird in label_detection.m gehandhabt.

Zuerst werden mit einem Harris-Eckendetektor (MATLAB-builtin) die Ecken im Bild gefunden [1]. Danach wird ein globaler Otsu-Threshold berechnet [4], der genutzt wird, um eine binäre Bildmaske zu erstellen, wobei alle Pixel mit größerer Intensität als der Threshold, auf 1 gesetzt werden, und alle anderen auf 0.

Mit dieser Maske wird wiederum ein Summed Area Table erstellt [2].

Für alle gefundenen Ecken wird nun versucht Nachbarn zu finden, die ein plausibles Label ergeben könnten. Dabei wird aus dem Vorschitt der Perspektivenkorrektur die Höhe der Bücherregale berechnet und daraus die ungefähren Label-Dimensionen abgeleitet. Diese konstanten Umrechnungsfaktoren wurden durch das Analysieren von Bildern abgelesen und fix in das Programm eingetragen. So ist zum Beispiel der Faktor von Regalhöhe zu Labelhöhe 0.135. Da manche Bücher weiter hinten stehen oder leicht aus dem Regal herausstehen, wird für die Label-Dimensionen ein Akzeptanzbereich erstellt,

in dem sich alle im Bild befindlichen Labels befinden sollten.

Für alle gefundenen und erfolgreichen Ecken-Nachbar-Kombinationen wird nun ein Label-Kandidat erstellt - das ist der Bereich zwischen den zwei Punkten. Es wird überprüft, ob der Schwarzanteil dem eines typischen Labels entspricht.

Für diesen Schwarzanteil wird zuerst ein lokaler Label-Threshold, ebenfalls mit der Methode von Otsu, berechnet. Dann werden die schwarzen Randbereiche mittels der MATLAB-Funktion `imclearborder` bereinigt. Das hilft auch sehr bei schief stehenden Büchern.

Weiters wird die Summe der weißen Bereiche im Label, mittels Integral Imaging (und dem zuvor erstellten Summed Area Table), berechnet.

Als Ergebnis erhalten wir somit die Anzahl der Pixel mit lokalen dunklen Bereichen und globalen hellen Bereichen. Diese zwei Werte werden dann in Verhältnis zueinander gestellt und das Ergebnis muss zwischen 7 und 25 liegen (die Werte wurden fix eingetragen und mittels Testen bestimmt).

Falls, diese Überprüfung erfolgreich ist, nehmen wir an, dass der betrachtete Bereich ein Label ist.

Zum Schluss werden noch alle gefundenen Labels eliminiert, die innerhalb eines anderen Labels liegen. Es wurde auch versucht überschneidende Labels zu verwerfen, jedoch gäbe es hier ein Problem: wenn zwei Bücher extrem nah beieinander stehen, kann es passieren, dass damit mehrere Labels zu einem verschmolzen werden und somit korrekte Information verloren geht.

Anmerkung: *Wieso benutzen wir den globalen Weißwert und den lokalen Schwarzwert?* Wenn wir den lokalen Weißwert (lokaler Threshold) nutzen würden, verlieren wir jegliche Relation zum Originalbild. Als Ergebnis bekäme man dann viel zu viele false-positives, weil dann auch dunkelgraue Bereiche eventuell als weiß gelten würden.

4.4 Optical Character Recognition

Normalized Cross Correlation (Template Matching)

Der `nxcorr2` Algorithmus wurde mithilfe einer Publikation implementiert [3]. Die Preconditions beinhalten, dass man zwei Binärbilder vergleicht, und dass das Template gleich groß oder kleiner als das Originalbild ist. Zuerst habe ich die lokalen Summen berechnet, was uns durch einen Blogpost von Matlab zu ihrer `normxcorr2` Implementierung vorgeschlagen wurde¹. Die Berechnung dieser lokalen Summen ist von einem veröffentlichten Matlab Code auf Matworks übernommen, da dieser komplizierte Optimierungen beinhaltet, wie das sequentielle Berechnen der cumulative sums². Weiters muss die initiale Korrelationsmatrix mithilfe von zwei Fouriertransformationen berechnet werden, wie in

¹<https://blogs.mathworks.com/steve/2006/05/02/fast-local-sums/>

²<https://www.mathworks.com/matlabcentral/fileexchange/24925-fast-robust-template-matching>

der Publikation beschrieben wird. Nach langem rumprobieren mit der fft-Funktion von Matlab, was nicht funktionierte, haben wir bei anderen Implementierung nachgesehen, und gesehen dass man fft2 verwenden muss, und diesen Teil des Codes verändert (und an den autoren gecreditted). Die Korrelationsmatrix wird dann mithilfe von den Durchschnittswerten des Templates normiert. Zu guter letzt wird die mathematische Formel, die in der Publikation beschrieben wird implementiert um die normierte Matrix zu erhalten.

Sum of Squared Differences

Der SSD Algorithmus ist wie schon beschrieben in einer simpleren Variante implementiert, da unsere Preconditions das ermöglichen. Man legt das Template über das Bild, dass in unserem Fall gleich groß ist, und benutzt die übereinanderliegenden Pixelwerte um eine quadrierte Differenz dieser zu berechnen. Schließlich summiert man diese und erhältet einen Wert, der pseudonormiert wird, um schönere/kleinere Werte zu erhalten.

OCR Die Optical Character Recognition ist eine Zusammensetzung aus dem Preprocessing und einer der gewählten Template Matching Varianten. Dabei werden alle Patches/Character, die vom Preprocessing zurückgegeben werden mit den Templates verglichen, und das jeweils beste Template wird als matched character angenommen.

5 Evaluierung

5.1 Datensatz

Insgesamt 14 Bilder von Bücherregalen in der Universitätsbibliothek, aufgenommen mit einem Xiaomi Mi MIX 2S.

Auflösung : 4032 x 3024 px, und alle anderen Einstellungen auf Auto.

5.2 Evaluierungsfragen

Interaktion zwischen den Komponenten

- *Werden die Regalfächer korrekt erkannt?*

Die Erkennung von Regalfächern wurde aufgrund von Schwierigkeiten bei der korrekten Erkennung dieser durch eine ungefähre Heuristik approximiert. Dies reicht aus um die Bücher in Zeilen einzuordnen.

- *Wird die Perspektive korrekt angepasst?*

Die Perspektive wird bei allen Bildern im Datensatz in akzeptablem Maß korrigiert.

- *Werden alle Labels erkannt?*

Es gibt mehr Probleme mit False-positives als mit nicht erkannten Labels. In seltenen Fällen wird ein Label nicht erkannt, in den meisten Fällen werden die Labels mehrfach markiert.

- Sind die Bounding Boxes der gefundenen Labels korrekt? Ist der gesamte Text darin enthalten?

Wir erhalten viele false-positives beziehungsweise unvollständige Bounding-areas, diese fallen jedoch bei der Optical Character Recognition durch, da aufgrund von fehlendem Texts oder eines zu kleinen Bereichs entweder zu wenige Zeichen, oder aufgrund des Rauschens von zu großen Bereichen zu viele Zeichen erkannt werden.

- Werden die Labels korrekt in Bücher-Koordinaten umgewandelt?
Bücher werden aufgrund ihrer Labelkoordinaten in Zeilen eingesortiert.
- Werden die Labels der TU-Bibliothek korrekt gelesen und in Text umgewandelt?
Aufgrund der vielen False-positives aus dem vorherigen Schritt und der hohen Variabilität der einzelnen Zeichen auf dem Bild, erhalten wir oft zu viele, zu wenige, oder gar die falschen Zeichen.
(Sehr beliebt sind "Y" und "L" was falsche Zeichen angeht).
- Wird die Wahrscheinlichkeit der Label-Korrekttheit angemessen berechnet?
Wurde aufgrund von zu niedriger Variabilität entfernt. Die Konfidenz war immer im gleichen Bereich, auch bei offensichtlich falschen Resultaten.

6 Schlusswort

(max. 1 Seite)

Hier fasst ihr Ergebnisse Eures Projekt zusammen:

Welche Schlussfolgerung lässt sich ziehen? Gibt es offene Probleme? Wie lässt sich Eure Lösung noch verbessern? etc.

Literatur

- [1] Mike Stephens Chris Harris. A combined corner and edge detector. *Alvey Vision Conference*, 1988. Zugriff 28.10.2019; Verantwortliche: Anna Nieto-Berezhinskaya, Laurenz E. Fiala.
- [2] Franklin C. Crow. Summed-area tables for texture mapping. 1984. Zugriff 12.11.2019; Verantwortliche: Laurenz E. Fiala.
- [3] J. P. Lewis. Fast normalized cross-correlation. 1995. Zugriff 5.1.2019; Verantwortliche: Aleksandar Vucenovic.
- [4] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *Computer Graphic*, 1979. Zugriff 5.1.2019; Verantwortliche: Laurenz E. Fiala.