



Tesseract.js

Pure Javascript Multilingual OCR

Sebastian Brock, 26.06.2023

Web Technologien SS2023

Was ist Tesseract?

- Freie, offene Software zur Texterkennung aus Bildern
- **OCR** - Optical Character Recognition
- Nicht nur lateinische Schriftzeichen, sondern auch z.B. chinesische oder kyrillische Schriften erkennbar
- Dient einer Vielzahl von OCR-Programmen als Basis
- Basisprogramm via CLI ausführbar

OCR - Die trockene Definition

- Ziel: Aus einem physisch vorhandenen Dokument sollen Texte als solche digital erkannt werden
- Scanner oder Kameras liefern lediglich Rastergrafiken als Grundlage (bestehend aus Pixeln)
- ➔ Optische Texterkennung erkennt aus Pixeldaten Buchstaben als solche und wandelt sie in den entsprechenden Code einer Textcodierung zu (z.B. ASCII)

Historisches zu OCR

- **1913:** „Optophon“ wandelte Schriftzeichen in bestimmte Töne um, um Blinden das Lesen zu ermöglichen
- **1914:** Erste OCR-Ähnliche Maschine von Emanuel Goldberg entwickelt (Umwandlung von Schriftzeichen in Telegrafен-Code)
- **1976:** Kurzweil Reading Machine
 - Texterkennung in jeder Schriftart
 - Eingabe via Scanner



Raymond Kurzweil, 1976

Historischer OCR Fail - JBig2

- Scanner der Firma Xerox nutzten JBig2, um gescannte Dokumente zu komprimieren
- Durch „Pattern Matching“ wird für jedes Schriftzeichen nur eine Darstellung gespeichert
- Bug: Schriftzeichen wurden fehlerhaft erkannt
- 2013 durch David Kriesel öffentlich gemacht
- 2015: JBig2 durch das BSI als **unzulässig für rechtssichere Scans** eingestuft

Vorher	Nachher
7 113569370251	11356937025
113569470251	113569470251
113669471251	113669471251
113669571251	113669581251
113669581261	113669581261
114669581262	114669581262
114670581262	114670581262
115670681262	115670681262

Auszug aus „Traue keinem Scan, den du nicht selbst gefälscht hast“ von David Kriesel

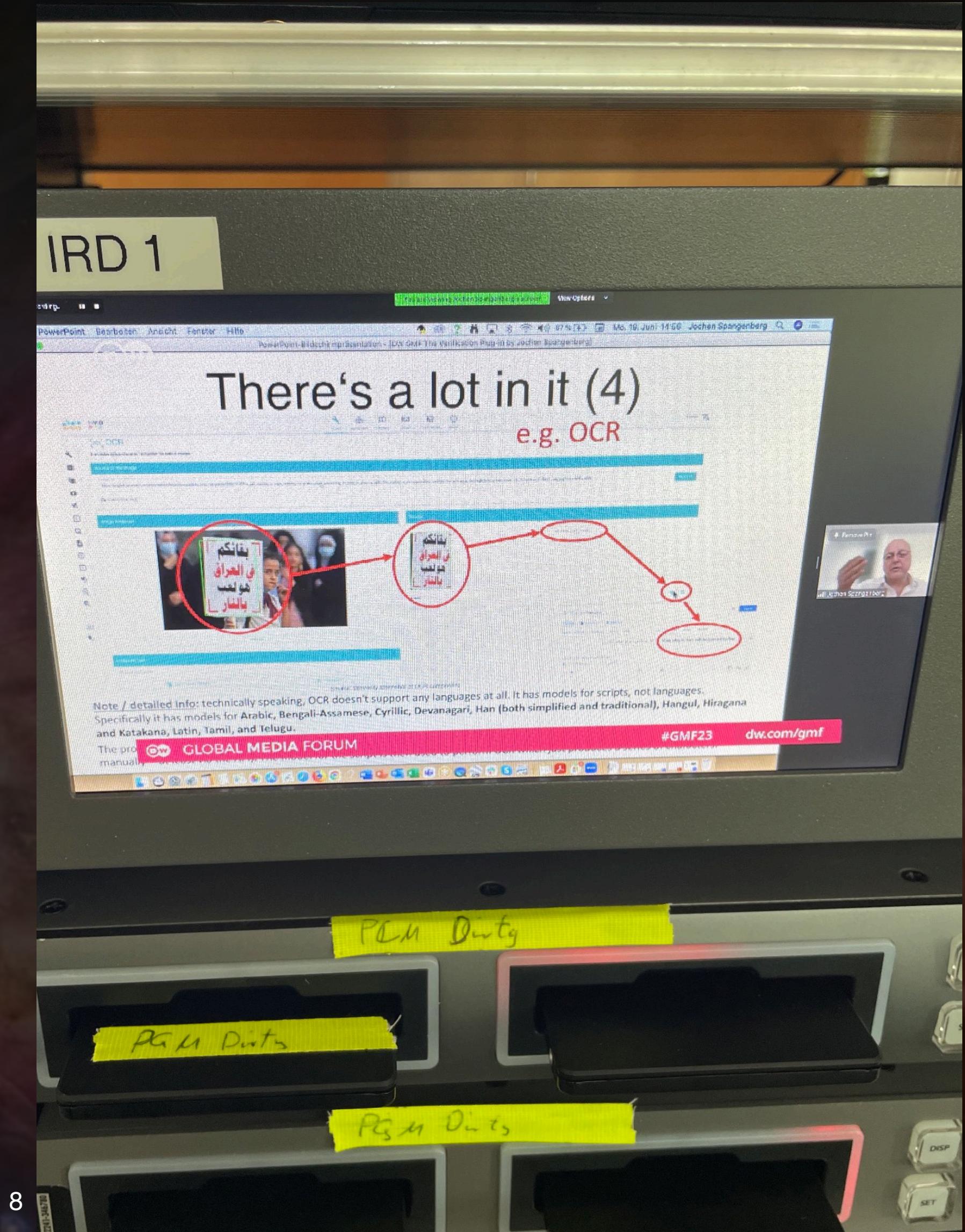
Historie Tesseract

- **1984 - 1994** Entwicklung bei HP für Scanner durch Ray Smith
- **2005** Google übernimmt das Projekt, es wird Open Source
- **2006** Weiterentwicklung als Grundlage von Google Books
- **2011** Erkennung indischer Sprachfamilien
- **2012** Erkennung arabischer und hebräischer Texte
- **2016** Nutzung eines neuronalen Netzes zur Texterkennung
- **2018** Google nicht mehr aktiv an Entwicklung beteiligt

Wo ist Tesseract im Einsatz?

- Google-Produkte:
 - Mobile Geräte (Android)
 - Texterkennung in Videos
 - Gmail - Spamerkennung auf Bildern
- Paperless NGX (Dokumentendigitalisierung)
- Wolfram Mathematica
- 28+ GUIs
- Viele weitere Projekte, Erweiterungen, Plugins etc.

Wo ist Tesseract im Einsatz? (2)



Funktionsweise

- 1) Umrisse erkannter Komponenten (Textblöcke) werden berechnet
- 2) Anhand der Umrisse werden *Blobs* organisiert, welche jeweils eine Zeile Text enthalten
- 3) Einzelne Wörter werden durch Analyse der Zeichenabstände erkannt
- 4) Texterkennung 1: Jedes Wort, welches definitiv erfolgreich erkannt wurde, wird als Trainingsdaten für den *adaptive classifier* genutzt
- 5) Texterkennung 2: Sollte der classifier erst gegen Ende etwas gelernt haben, was am Anfang nützlich gewesen wäre, kann dies nun genutzt werden

Abstract

Das nachträgliche Austauschen eines Schauspielers kann innerhalb der Filmindustrie verschiedene Möglichkeiten besitzen. Hierzu wird zu Beginn der Begriff "Actor Replacement" eingeführt, der eine Eigenkreation darstellt und das nachträgliche Ausstauschen eines Motivs aus bereits produziertem Material betitelt. Auf der einen Seite werden bekannte Schauspieler aus großen Filmreihen, wie beispielsweise "Star Wars", immer älter oder sind bereits verstorben. Ihre gespielten Charaktere sollen allerdings weiterhin auf der Filmleinwand auftauchen können, sodass hierzu einige audiovisuelle Methoden und Verfahren notwendig sind. Auf der anderen Seite kann es möglich sein, dass ein Film in der Vergangenheit spielt und ein Schauspieler entsprechend jünger dargestellt werden muss und dadurch nur spezifische Elemente des Schauspielers ausgetauscht werden müssen.

Ziel dieser Ausarbeitung ist die Erkennung solcher Methoden und die Beantwortung der Forschungsfrage: "Mit welchen audiovisuellen Techniken können in der Filmproduktion Schauspieler ersetzt werden?". Hierzu werden traditionelle Techniken der Filmindustrie zum nachträglichen Austauschen eines Schauspielers ergründet, sowohl Schwierigkeiten, als auch Unterschiede zwischen diversen Methoden analysiert. Anhand einer praxisorientierten Perspektive sollen diese im Anschluss demonstriert und validiert werden. Der Schwerpunkt dieser Arbeit liegt auf der praxisorientierten Anwendung einer ausgewählten Methodik zum nachträglichen Austauschen eines Schauspielers.

Erkannte Methoden werden in drei Stufen eingeteilt und somit das Drei-Stufen-Modell konstruiert. Jede dieser Stufen repräsentiert eine Erweiterung des Actor Replacements um einen weiteren Faktor:

Stufe 1 - Integration eines Motivs in eine bestehende Szene

Stufe 2 - Substitution eines Motivs in einer bestehenden Szene

Stufe 3 - Substitution spezifischer Elemente eines Motivs in einer bestehenden Szene

Stufe 1 umfasst Möglichkeiten, eine Person oder ein Objekt nachträglich in eine bereits bestehende Szene zu integrieren. Die Stufe 2 erweitert das Ganze um die Notwendigkeit, in bereits produziertem Material eine Person oder ein Objekt durch ein anderes Motiv auszutauschen. Darauf baut Stufe 3 auf, allerdings soll hier keine ganze Person

Wort kann nicht erkannt werden

Classifier lernt durch erfolgreiche Worterkennung etwas, was bei der Erkennung des nicht erkannten Wortes helfen könnte

Durchlauf 1 fertig



Abstract

Das nachträgliche Austauschen eines Schauspielers kann innerhalb der Filmindustrie verschiedene Möglichkeiten besitzen. Hierzu wird zu Beginn der Begriff "Actor Replacement" eingeführt, der eine Eigenkreation darstellt und das nachträgliche Ausstauschen eines Motivs aus bereits produziertem Material betitelt. Auf der einen Seite werden bekannte Schauspieler aus großen Filmreihen, wie beispielsweise "Star Wars", immer älter oder sind bereits verstorben. Ihre gespielten Charaktere sollen allerdings weiterhin auf der Filmleinwand auftauchen können, sodass hierzu einige audiovisuelle Methoden und Verfahren notwendig sind. Auf der anderen Seite kann es möglich sein, dass ein Film in der Vergangenheit spielt und ein Schauspieler entsprechend jünger dargestellt werden muss und dadurch nur spezifische Elemente des Schauspielers ausgetauscht werden müssen.

Ziel dieser Ausarbeitung ist die Erkennung solcher Methoden und die Beantwortung der Forschungsfrage: "Mit welchen audiovisuellen Techniken können in der Filmproduktion Schauspieler ersetzt werden?". Hierzu werden traditionelle Techniken der Filmindustrie zum nachträglichen Austauschen eines Schauspielers ergründet, sowohl Schwierigkeiten, als auch Unterschiede zwischen diversen Methoden analysiert. Anhand einer praxisorientierten Perspektive sollen diese im Anschluss demonstriert und validiert werden. Der Schwerpunkt dieser Arbeit liegt auf der praxisorientierten Anwendung einer ausgewählten Methodik zum nachträglichen Austauschen eines Schauspielers.

Erkannte Methoden werden in drei Stufen eingeteilt und somit das Drei-Stufen-Modell konstruiert. Jede dieser Stufen repräsentiert eine Erweiterung des Actor Replacements um einen weiteren Faktor:

Stufe 1 - Integration eines Motivs in eine bestehende Szene

Stufe 2 - Substitution eines Motivs in einer bestehenden Szene

Stufe 3 - Substitution spezifischer Elemente eines Motivs in einer bestehenden Szene

Stufe 1 umfasst Möglichkeiten, eine Person oder ein Objekt nachträglich in eine bereits bestehende Szene zu integrieren. Die Stufe 2 erweitert das Ganze um die Notwendigkeit, in bereits produziertem Material eine Person oder ein Objekt durch ein anderes Motiv auszutauschen. Darauf baut Stufe 3 auf, allerdings soll hier keine ganze Person

Wort wird nun korrekt erkannt

Durchlauf 2 fertig,
Texterkennung abgeschlossen

Volume 69, pages 872–879.

Fig. 1. An example of a curved fitted baseline.



Fig. 2. A fixed-pitch chopped word.

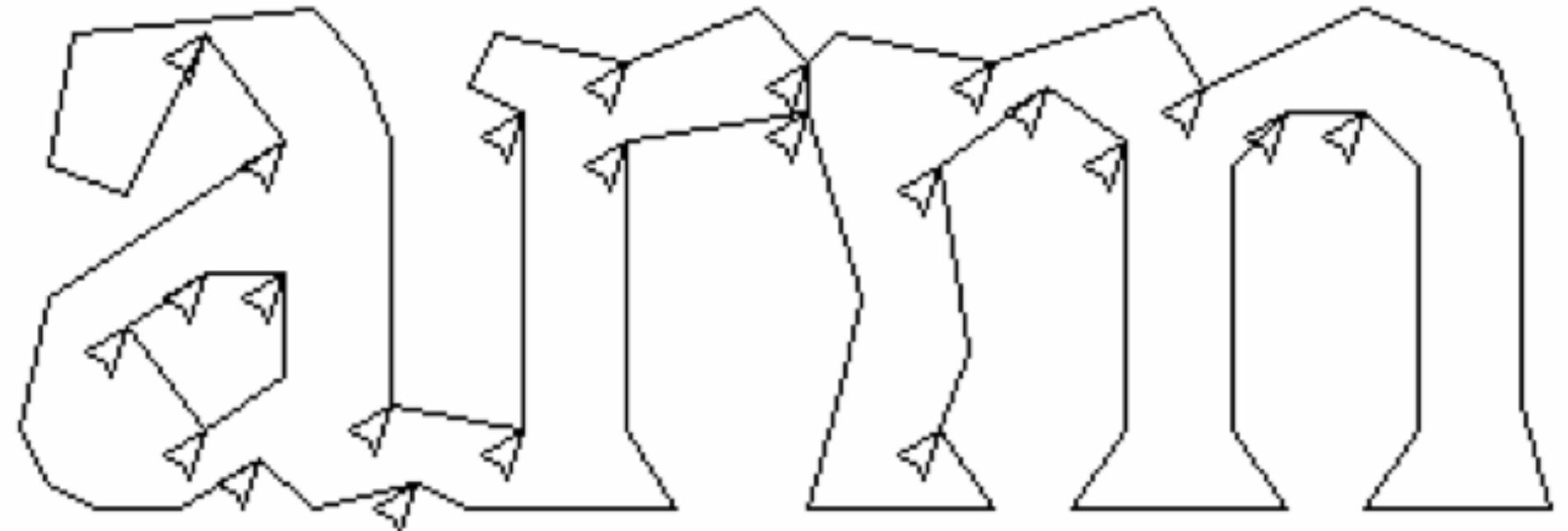


Fig. 4. Candidate chop points and chop.

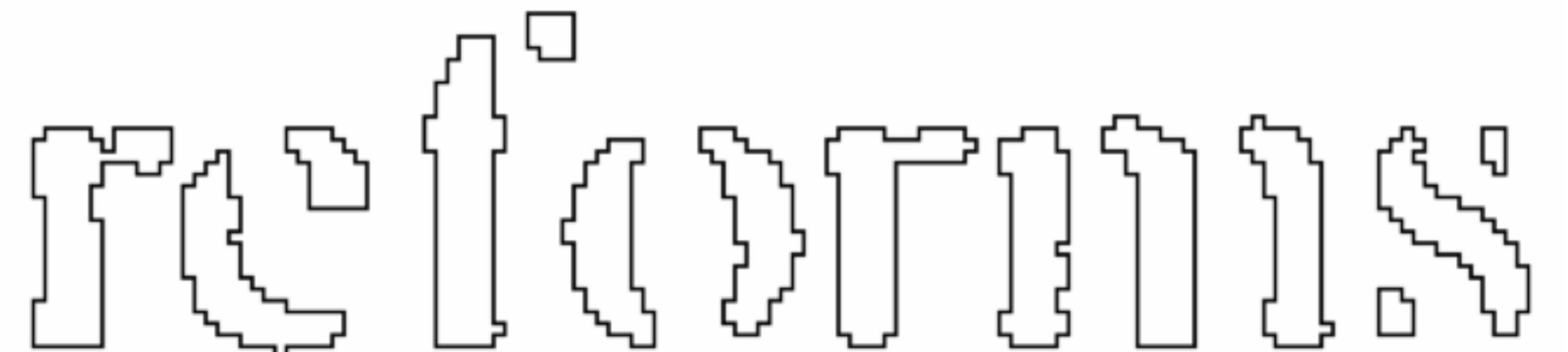


Fig. 5. An easily recognized word.

Kleine Meta-Aufgabe - Level 1

<https://cutt.ly/7wek5r20>

NICHT abtippen! :)

Kleine Meta-Aufgabe - Level 2

gotos.in/you+have+to+write+a+really+really+long+url+to+get+to+650+characters.+like+seriously+you+have+no+idea+how+long+it+has+to+be+650+characters+is+absolutely+freaking+enormous.+You+can+fit+sooooooooooooooo+ooooooooooooo+much+data+into+650+characters.+My+hands+are+getting+tired+typing+this+many+characters.+I+didnt+even+realise+how+long+it+was+going+to+take+to+type+them+all.+So+many+characters.+I'm+bored+now+so+I'll+just+copy+and+paste.+I'm+bored+now+so+I'll+just+copy+and+paste+1ke+.I'm+bored+now+so+I'll+just+copy+and+paste.I'm+bored+now+so+I'll+just+copy+and+paste.I'm+bored+now+so+I'll+just+copy+and+paste.I'm+bored+now+so+I'll+just+copy+and+paste.+It+has+to+be+freaking+enormously+freaking+enormous

Tesseract.Js

- Port der Tesseract-Engine in JavaScript
- Wurde mithilfe von Emscripten erstellt (Assemblercode zu WebAssembly)
- Teil des **Project Naptha**
- Verfügbar via NPM, Script aber auch direkt einbindbar
- Erster Alpha-Release: **14. Mai 2019**
- Erster Production-Release: **19. Dezember 2019**
- ~ **55.000** wöchentliche NPM-Downloads

Tesseract.Js - Einbindung

```
import Tesseract from 'tesseract.js';

Tesseract.recognize(
  'https://tesseract.projectnaptha.com/img/eng_bw.png',
  'eng',
  { logger: m => console.log(m) }
).then(({ data: { text } }) => {
  console.log(text);
})
```

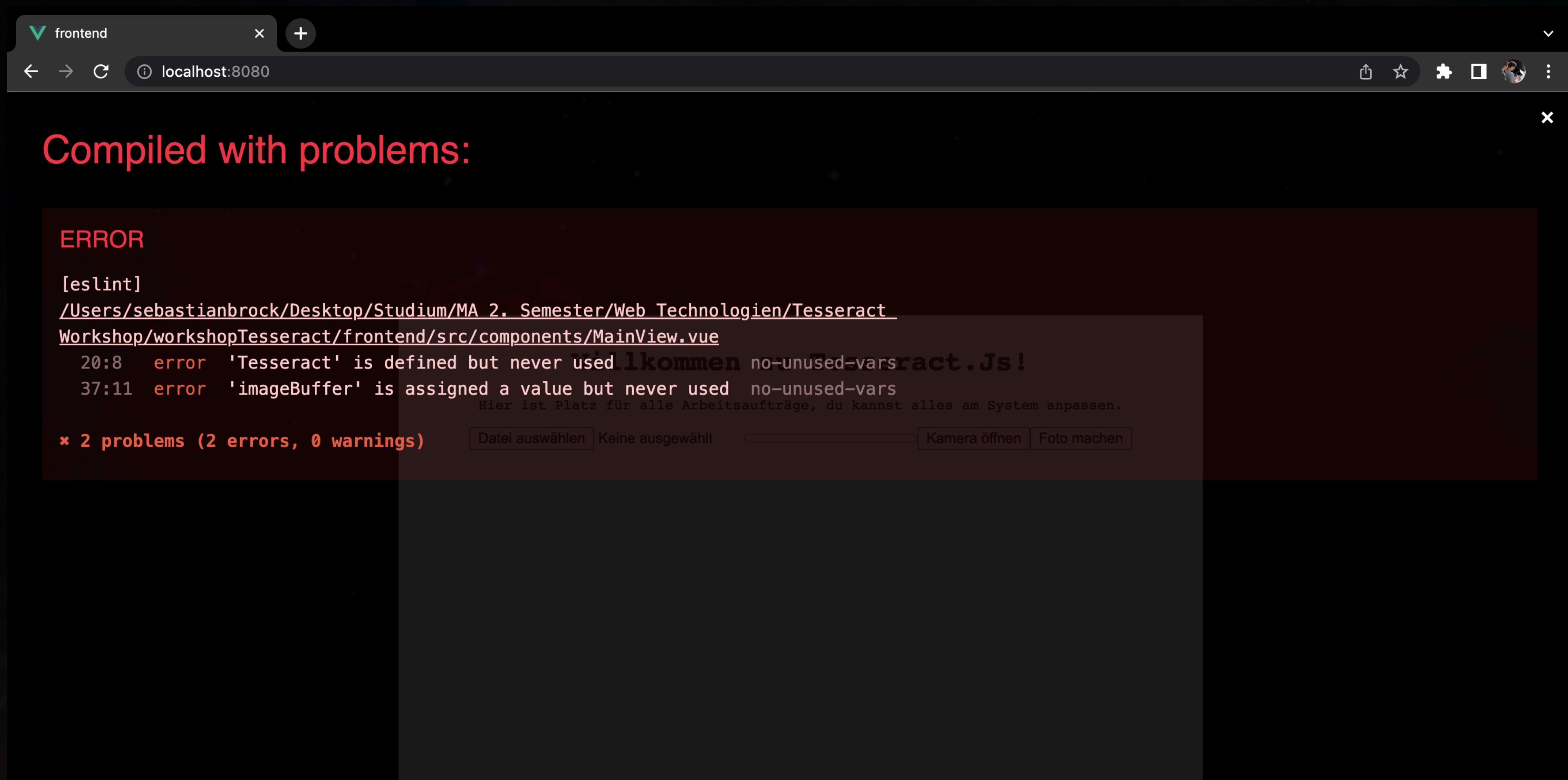
Praxisteil

Aufgabe 0 (10 Min)

- Repo lokal klonen
- NPM-Pakete installieren
- Im Terminal in Ordner „frontend“ wechseln
- Lokal ausführen via **npm run serve**
- Der Compiler-Fehler ist gewollt!

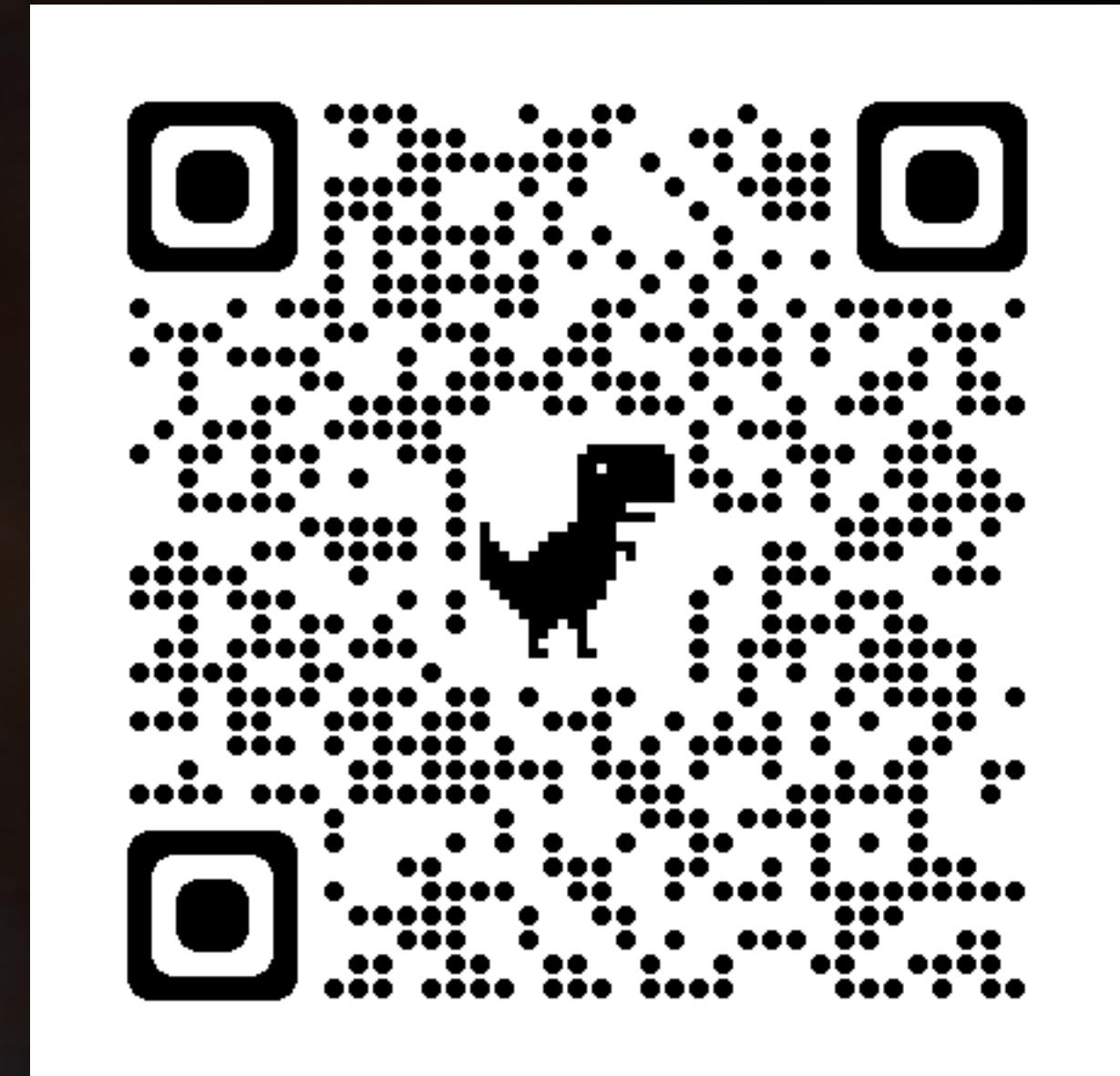


[github.com/sebastianbroc/
workshopTesseract](https://github.com/sebastianbroc/workshopTesseract)



Aufgabe 1 (15 Min)

- Tesseract.Js Texterkennung via Datei-Upload an der vorgesehenen Stelle im Code einbinden
- Die Dokumentation des Tesseract.Js Repos ist hierfür sinnvoll
- Ergebnis der Texterkennung anschließend im Frontend anzeigen



github.com/naptha/tesseract.js/#readme

Aufgabe 2 (15 Min)

- Fortschritt der Texterkennung bisher noch nicht für den Nutzer sichtbar
- Wie könnte der Fortschritt im Frontend dargestellt werden?
- Überlege dir, wie der Fortschritt der Texterkennung basierend auf den Rückgaben Tesseracts auf der Seite dargestellt werden könnte!

Aufgabe 3 (20 Min)

- Texterkennung ist bisher nur auf hochgeladenen Dateien möglich
- Nun soll die Webcam eingebunden werden, um Fotos von Dokumenten machen zu können
- Rahmencode bereits vorhanden!
- Auskommentierte Zeilen im Template müssen nun genutzt werden

Aufgabe 4 (20 Min)

- Stichwort: Experimentieren!
 - Texterkennung auf verschiedenen Dokumenten ausprobieren
 - Ausgedruckte Texte
 - Handschriftliche Objekte (zB Notizzettel)
 - Stilisierte Schriftzeichen
 - Gleiche Artefakte gerne auch mit dem Smartphone scannen
- ➡ Unterscheidet sich das Ergebnis? Wo liegen die Grenzen der Software?

Abschluss - Fazit

- Wie gut hat die Einbindung und Nutzung generell funktioniert?
- Wie gut hat Tesseract im Vergleich zum Smartphone abgeschnitten?
- Würdest du Tesseract.Js selbst in Betracht ziehen?



Vielen Dank für's Mitmachen!

Quellen und weiterführende Links

- <https://tesseract-ocr.github.io/tessdoc/User-Projects---3rdParty.html>
- <https://static.googleusercontent.com/media/research.google.com/de//pubs/archive/33418.pdf>
- <https://www.veraai.eu/home>
- <https://gmf-event.com/program-details/181/>