Universiteit van Amsterdam

Rapport

Rapport Multimedia Eindproject

27 juni 2018

Tutor:
Engel Hamer

Practicumgroep:
C2

Cursus:
Multimedia

Student:
Mounir el Kirafi
11879106

1 Introductie

Voor het projectblok multimedia is er besloten een app te maken die Optical Character Recognition (OCR) implementeert.

Deze app kan een foto nemen van een sudoku in een krant of een boek. Vervolgens wordt deze foto omgezet tot een grid met 81 vakjes waarbij elk van die vakjes door een algoritme gaat om te bepalen welk getal erin staat.

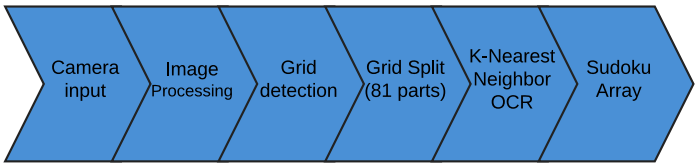
Vervolgens wordt de array aan getallen die hiermee zijn gekregen ingevuld in een in-app sudoku. De gebruiker heeft dan een aantal mogelijkheden: De sudoku zelf invullen door te typen of Speech to Text, of het laten oplossen door onze snelle sudoku solver.

Mijn persoonlijke bijdrage aan dit project zit in het visuele gedeelte. Ik heb gezorgd voor het herkennen van de sudoku en de getallen.

2 Methode

OCR is een interessant onderwerp en er zijn veel verschillende manieren voor. Er is Google Vision gebaseerd op Google's cloud services, OpenCV, Tesseract en meer. In dit project is er de keuze gemaakt voor OpenCV. Dit omdat Google vision een creditcard nodig heeft voor in ieder geval een trial, en OpenCV gebruikervriendelijker is dan Tesseract.

De sudoku herkenning bestaat uit zes stappen, te zien in figuur een, hier onder.



Figuur 1: Sudoku herkenning stappen

Het processing van de input afbeelding wordt gedaan door de volgende stappen:

- Grayscale transformatie
- Inversion (bitwise not met zichzelf)
- Blur (Gaussian Blur)
- Tresholding

Deze stappen maken het vinden van de grid makkelijker door de voornaamste figuren en lijnen in een foto naar voren te halen. Dit gebeurt voornamelijk bij het tresholden, waarbij pixels afhankelijk van hun intensiteit zwart of wit worden.

Vervolgens wordt de sudoku grid gevonden door de alle contouren in de foto te vinden. Na deze contouren te hebben gevonden wordt er eerst gesorteerd op alle vierzijdige contouren, en vervolgens op de grootste contour. Dit neemt aan dat de sudoku de focus van de afbeelding is, en dat er dus niet een groter vierkant naast is. Zie figuur drie en vier onderaan ter illustratie.

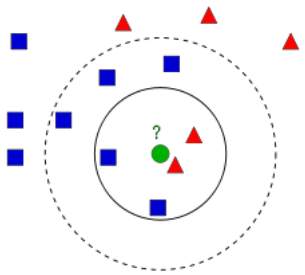
Als een na laatste komt de image herkenning. Dit wordt gedaan door de sudoku op te splitsen in 81 stukken, elk een getal. Elk van deze getallen worden gegeven aan het algoritme dat getallen kan herkennen: het K-Nearest Neighbor algoritme.

Ten slotte worden de gevonden getallen terug gegeven als een array van getallen.

2.1 Algoritme

Het OCR algoritme kent meerdere vormen: **Nearest Neighbor**, **K-Nearest Neighbor** en **weighted K-Nearest Neighbor**. Ter illustratie wordt figuur twee gebruikt.

Het scenario is als volgt: Er zijn blauwe en rode huizen. Er is een groene huis bij gekomen en die moet geclassificeerd worden.



Figuur 2: Nearest Neighbor illustratie

Met Nearest Neighbor, wordt er gekeken naar welk huis het dichtsbij is. In dit geval wordt het dus een rood huis. Met K-nearest neighbor wordt er een getal ***k*** meegegeven. Deze bepaald naar hoeveel huizen in de buurt wordt gekeken. In het geval van het getal 3 is het nog steeds een rood huis, maar in het geval van getal 5 is het een blauw huis. Modified k-nearest neighbor is hetzelfde als k-nearest neighbor, behalve dat de huizen die dichterbij zijn zwaarder meetellen.

Dit voorbeeld is vergelijkbaar met OCR. De verschillende kleuren zijn in deze situatie de getallen en het groene huisje het getal dat binnenkomt.

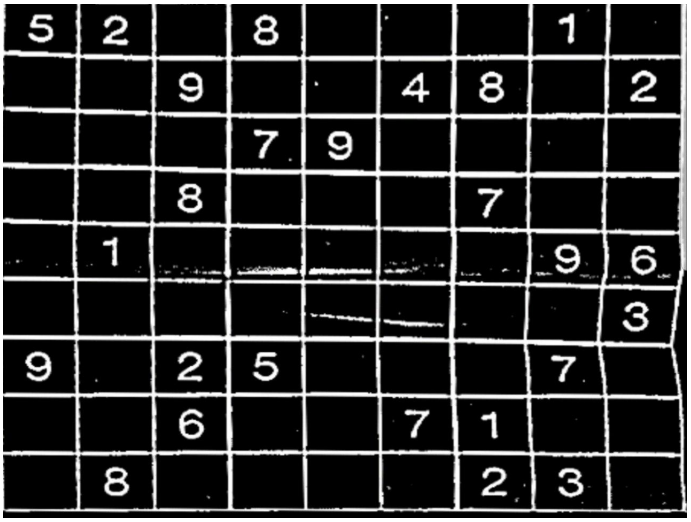
Verder zijn er ook nog algoritmes om deze algoritme te gebruiken. Er is een ***Brute force*** en ***KD Tree*** algoritme. De brute force algoritme kijkt domweg naar alle naburige klassen (elk klasse is een ander kleur uit het vorig voorbeeld). KD Tree kijkt naar een bepaald huis, en van daar naar de dichtstbijzijnde. Neem punt A als startpunt neemt, punt B als een punt dat ver van punt A is en punt C dat dichtbij punt B is. Hieruit valt te concluderen dat punt C ver van A is. Als er dan een punt D is dit dichtbij punt A is, hoeft de afstand tot C niet berekend te worden. Die valt automatisch weg tegen punt D. Op deze manier wordt een boom gebouwd met aftakkingen van de verschillende punten.

2.2 Procedure

Voor het implementeren van deze algoritme moet er een database zijn om het binnenkomende getal tegen te checken. Hiervoor is een foto gebruikt met 5000 handgeschreven versies van elk getal, van 0 tot 9. Hiervan wordt een model gebouwd. Vervolgens wordt het getal bewerkt door het recht te zetten en in het midden te zetten. Ten slotte wordt de nearest neighbor algoritme aangeroepen met behulp van het gemaakte model. Hier komt een getal uit. Dit wordt terug gegeven als een array van getallen dat de sudoku representeert.

2.3 Software en apparatuur

Dit project is gedaan in Android Studio op Windows 10. Als android apparaat is een Samsung Galaxy S8+ gebruikt.



Figuur 3: Verwerkte input afbeelding

Figuur 4: Extracted cell

Teamgenoten: Abdelilah Ahbari, Liam Zuiderhoek, Mund Vetter. Groep in19, app ARSS.