Team 16 – License Plate Recognition

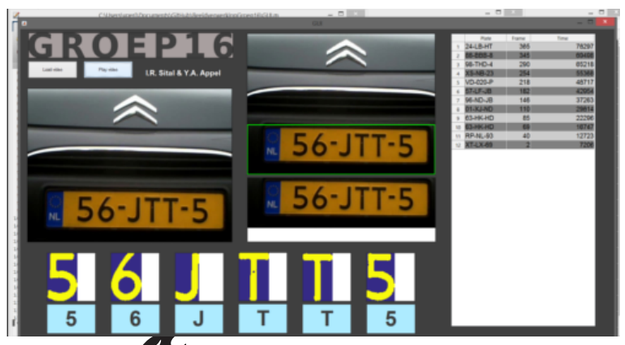
Week 4 – Yoeri Appel, Ishan Sital

Intro

De opdracht was om een systeem te implementeren welke cijfer/letter combinaties van kentekens kan herkennen in videos. Om het probleem van de kenteken herkenning aan te pakken zijn we gestart met het maken van een test set van kentekens. M.b.v. deze test set hebben we scatterplots gemaakt om zo thresholds te vinden voor de segmentatie van de kentekens. Iteratief hebben we de thresholds telkens bijgesteld en meer eigenschappen van een plaatje mee laten wegen.

Nadat de segmentatie van het kenteken werkte is er gestart om de individuele karakters van het kenteken te segmenteren. Hiertoe hebben we wederom thresholding gebruikt. Vervolgens is er pattern recognition gebruikt om de daadwerkelijke tekens te bepalen. Ter controle hebben we de regels van NL kentekens opgezocht om een controle uit te voeren of er valide kentekens herkend worden.

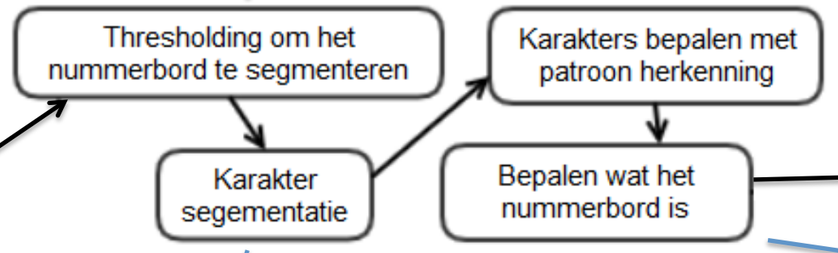
Als laatste hebben we aan de GUI gesleuteld om zo de juiste resultaten te tonen. Hierbij kwam naar voren dat we een wisseling van scene's moesten kunnen bepalen om zo te bepalen wanneer we starten met het herkennen van een nieuw kenteken en de False Positives en False Negatives kunnen voorkomen.



Kenteken segmentatie

Het eerste wat we willen doen is het kenteken te scheiden van de rest van het plaatje. Doormiddel van scatterplots op de verschillende kleurkanalen hebben we bepaald met welke thresholds we deze scheiding konden maken. We hebben eerst een median filter toegepast op het plaatje om zo de ruis te verwijderen, maar de edges te behouden. Uiteindelijk bleek dat thresholding niet "alleen" het kenteken uit het plaatje segmenteerde. Om dit wel te doen hebben we van alle losse segmenten die overbleven de eigenschappen bepaald en vervolgens een selectie gedaan van 1 segment welke voldeed aan de voorwaarden van het kenteken. Denk hierbij aan de intensiteit van het segment, de verhouding, de breedte en de oriëntatie.

Conversie van RGB naar HSI kost veel computertijd en de uiteindelijke thresholding op het Intensiteit kanaal leverde uiteindelijk weinig verbetering op waardoor we hebben besloten om deze threshold achterwege te laten.



Karakter segmentatie

In deze stap was het de bedoeling dat we het gesegmenteerde nummerbord pakte en uit dat plaatje 6 karakters segmenteerde.
We hebben dit wederom met thresholding gedaan, maar wel op een iets andere manier. Elk nummerbord heeft net een andere kleur (door bijvoorbeeld belichting), maar op elk nummerbord kan je de karakters herkennen doordat de letters een donkere kleur hebben en de achtergrond een lichte. Daarom hebben we ervoor gekozen om het input plaatje om te zetten in een grijswaarden plaatje en vervolgens met de ingebouwde matlab functie graythresh gebruikt om een threshold te bepalen. Graythresh maakt gebruik van Otsu's methode bij het bepalen van de threshold. We hebben ook nog een sharpening filter op de input gebruikt om de randen te versterken en na thresholding hebben we bepaald welke objecten de karakters waren door naar bepaalde eigenschappen van de objecten te bekijken (hoogte/breedte verhouding, grootte, locatie in het beeld, etc). Nadat we de 6 objecten hebben bepaald, hebben we gekeken naar de afstand tussen deze objecten om te bepalen wat voor soort nummerbord het is (bijv. '99-XX-99' of '99-XXX-9').

Karakter herkenning

Bij deze stap moesten we bepalen welk karakter een plaatje representeerde. Hiervoor hebben we gebruikt gemaakt van patroon herkenning. We hebben het plaatje opgedeeld in een aantal delen en vervolgens voor elk deel gekeken hoeveel deze afweek van onze opgeslagen voorbeelden. Het aantal delen hebben we zo gekozen dat ze klein genoeg waren om zodat we vormen konden herkennen, maar groot genoeg om het effect van ruis te minimaliseren. Een van de weinige karakters waar ons programma moeite mee had om het goed te herkennen waren R en H. Om deze reden hebben we een speciale controle toegevoegd wanneer 1 van deze karakters werd herkend. Deze controle let vooral op de gebieden midden-boven en rechts-midden, omdat dit de gebieden zijn waarin de meeste verschillen zitten bij deze 2 karakters. Deze stap haalde ook vaak karakters zoals D en 0 door elkaar, maar doordat we de volgende stap hebben toegevoegd maakte dit niet uit.

Bepalen van nummerbord

Sommige karakters zijn bijna onmogelijk om uit elkaar te houden ((D,O), (8,B), etc), maar gelukkig hebben de Nederlandse nummerborden duidelijke regels.
In deze stap controleren we of de 6 karakters een geldig nummerbord vormen en als dit niet het geval is, of we een paar karakters kunnen veranderen zodat ze wel een geldig nummerbord vormen. Voor elk gesegmenteerd karakter hebben we een aantal karakters bepaald en voor elk van deze karakters hebben we het percentage dat dit plaatje overeen kwam met het karakter. Als de 6 karakters met de hoogste percentages geen geldig nummerbord vormen, vervangen we karakters zodat de reeks wel een geldig nummerbord vormt. Hierbij zorgen we dat de som van de percentages zo groot mogelijk blijft.

Evaluatie

We hebben teveel tijd besteed aan het krijgen van perfecte kenteken segmentatie, want achteraf bleek dat dit helemaal niet zo perfect hoeft te zijn. We hebben verschillende technieken geprobeerd zoals Sobel edge detectie, maar dit leverde geen verbetering op. Daarnaast hebben we geprobeerd om te filteren in het HSI domein, maar dit hielp weinig en het omzetten van RGB naar HSI gaf performance problemen. Nadat we dit in orde hadden gingen de rest van de stappen vlot en op het eind hebben we geprobeerd om het systeem te tweaken om de False Positives.

Het uiteindelijke resultaat is dat ons systeem ongeveer 80% scoort op de herkenning van Cat I+II video's