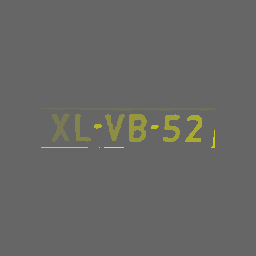
  
**Figuur 4 de scheiding tussen voor en achtergrond**

Het enige probleem dat je nu nog hebt is dat je de buitenrand van je plaat vaak ook meeneemt. Deze raak je kwijt door propagatie toe te passen. Om de karakters te kunnen scheiden labelen we de objecten.

**   
Figuur 5 de losse objecten**

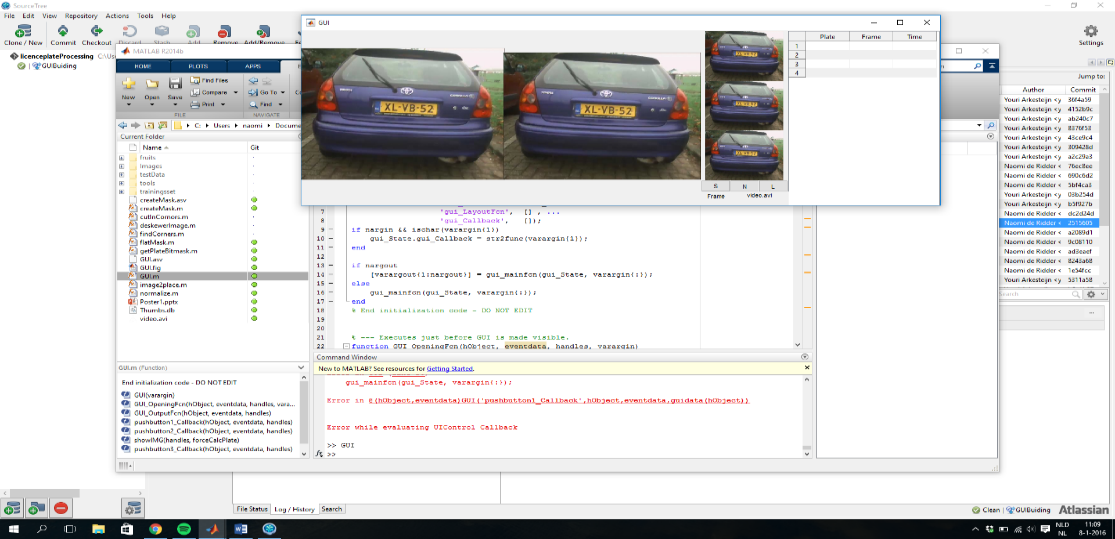
**Sampledata**Voordat we aan de letterherkenning kunnen beginnen hebben we sample data nodig. Per karakter dat voorkomt in een kentekenplaat wil je een binaire afbeelding hebben waarbij het karakter in het midden staat. Belangrijk is dat alle sampledata en de karakters die je uit de kentekenplaten haalt even groot zijn (± 100\*200 px).

**Letterherkenning**Na het verzamelen van testdata wil je puntsgewijs alle sample data met de objecten in je kentekenplaat vermenigvuldigen. Als je dit gedaan hebt wil je de som hiervan, delen door de som van de samples om te kijken welke sampledata het meest overeenkomt met het object in de kentekenplaat. Hoe hoger het percentage, des te meer overeenkomstigheid.

Hier ontstaan echter wel problemen bij karakters die erg veel op elkaar lijken zoals de 5 en de s. Dit kun je op te lossen door te kijken naar de plaats van het streepje, deze scheidt namelijk letters van cijfers. Door te kijken naar het karakter ernaast kun je bepalen of het een letter of cijfer moet zijn.

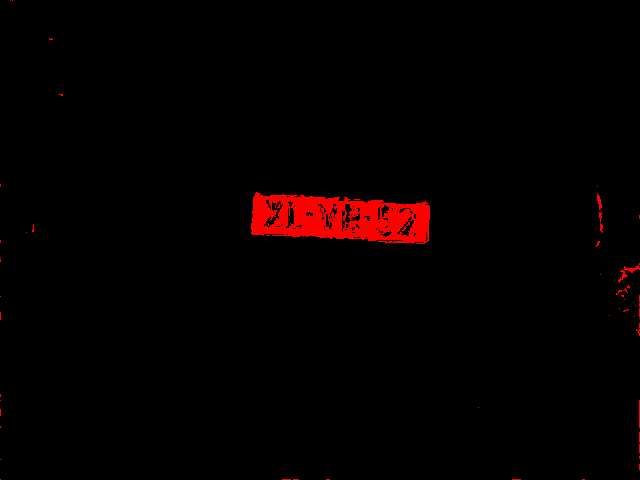
**De opdracht**We hebben de opdracht gekregen om met behulp van beeldverwerking een programma te maken dat kentekenplaatnummers herkent. Dit programma is gemaakt met Matlab en zal o.a. gebruik maken van segmentatie, edge detection en object separation.

**GUI**In de GUI is de video te zien waarop je kentekenplaatherkenning gaat toepassen. Je kunt een video inladen en deze starten, stoppen of het volgende frame laten zien m.b.v. 3 knoppen. Ook zie je op de video een rode rand om de kentekenplaat heen.

**Figuur 1 de GUI**

**Pixels herkennen die mogelijk van een kentekenplaat zijn**Voor het zoeken naar pixels die in een kentekenplaat zitten kunnen we zoeken naar geelaccenten. Om te zorgen dat dit proces belichtingsinvariant is doen we dit op de genormaliseerde data. Deze geel accenten worden als kentekenplaat herkend als ze in bepaalde constante waardes vallen. Deze RGB waarden zijn bepaald met behulp van de tool colorThresholder

**De locatie van de kentekenplaat bepalen**Nu hebben we een matrix waarin de pixels die mogelijk van een kentekenplaat zijn. Om dit beeld iets vrijer van gaten te maken passen we een closing toe. Na de closing zoeken we het grootste object omdat er veelal extra pixels zijn die ook als geel gemarkeerd worden, de ruis.

  
**Figuur 2 gele pixels en de locatie van de kentekenplaat**

**De kentekenplaat recht zetten ten opzichte van het beeld, en alleen de kentekenplaat overhouden**Omdat deze kentekenplaten ook vanuit velen hoeken en rotaties geschoten kunnen worden willen we de kentekenplaat rechtzetten voordat we object herkenning en seperation willen doen.

We doen dit doormiddel van het vinden van de hoeken van de kentekenplaat die we gevonden hebben en dit vlak te roteren en projecteren naar een nieuwe beeldruimte waar alleen de inhoud van de plaat recht op staat. Hierna zorgen we ervoor dat we alleen de kentekenplaat overhouden door de nieuwe hoeken te vinden en de image op deze plek uit te snijden.   
****  
**Figuur 3 de rechtgezette kentekenplaat**  
 **Het scheiden van de objecten en de achtergrond van de kentekenplaat**Nadat we deze kentekenplaat correct uit het beeld hebben gehaald is het belangrijk om een goede scheidingsmethode te vinden die belichtingsinvariant is en de zwarte letters van de gele achtergrond kan scheiden. Dit doen we door een binair beeld van de kentekenplaat te maken.

**Kentekenplaatherkenning**