



AKADEMIA GÓRNICZO-HUTNICZA

WYDZIAŁ FIZYKI I INFORMATYKI STOSOWANEJ

ANALIZA I PRZETWARZANIA OBRAZÓW

Detekcja pasa ruchu

Autor:
Maciej Kubicki

Prowadzący:
dr inż. Mariusz Jędrychowski

30 czerwca 2017

Spis treści

1	Wstęp	2
1.1	Temat	2
1.2	Wykorzystane technologie	2
2	Rozwiązanie	3
2.1	Algorytm	3
2.2	Aplikacja	6
3	Kilka screenów wyników	7
4	Wnioski	13

1 Wstęp

1.1 Temat

Tematem projektu jest realizacja aplikacji służącej do rozpoznawania pasa ruchu. Jako pas ruchu rozumiem ograniczające go linie. Zagadnienie to jest obecnie dość popularne - w związku z budową/rozwojem autonomicznych samochodów.

1.2 Wykorzystane technologie

Projekt zrealizowany został jako aplikacja Qt w wersji 5.6 w C++. Dodatkowo posługiwałem się biblioteką OpenCV.

2 Rozwiązanie

2.1 Algorytm

Problem ten jest dość skomplikowany, zwłaszcza, gdy zależny nam na dokładnych wynikach, jednak sam algorytm może być przedstawiony w kilku krokach:

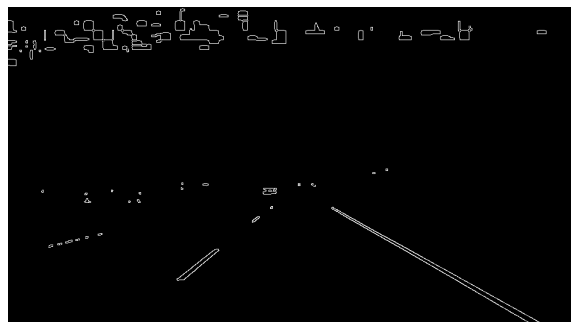
1. Krokiem pierwszym jest wyodrębnienie koloru żółtego i białego z obrazu - w takich kolorach można spotkać pasy na jezdni. W tym celu posługuję się progowaniem - kolor biały można łatwo uzyskać wybierając wysokie jasności z obrazu w skali szarości, natomiast kolor żółty odpowiednio dobierając próg, jednak tym razem progowanie jest dokonywane z przestrzeni HSV. Po wykryciu wspomnianych kolorów na otrzymanym obrazie wykrywam krawędzie stosując algorytm Canny'ego, będzie to niezbędne w kolejnych krokach.



Rysunek 1: Obraz wejściowy

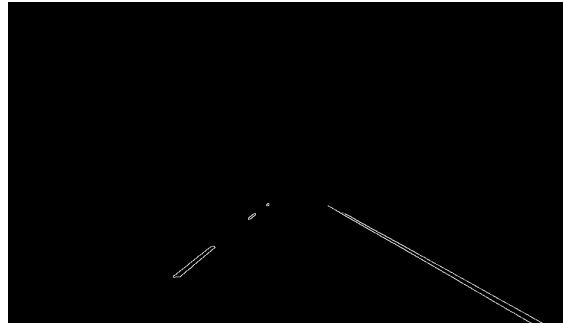


Rysunek 2: Obraz po progowaniu



Rysunek 3: Obraz po wykryciu krawędzi

2. Jak widzimy po efektach wykryte zostało również tło. W tym kroku naszym celem jest pozbycie się go - co wbrew pozorom nie jest trudne. Na podstawie ustawienia kamery oraz rozdzielczości dobieramy odpowiedni region zainteresowania, który bardzo dobrze się sprawdza. W moim przypadku ten region jest stały dla danego ustawienia kamery i rozdzielczości, jednak w bardziej zaawansowanych implementacjach prawdopodobnie region ten jest modyfikowany w zależności od ustawienia kierownicy, bądź też w momencie jazdy pod górę. W moim przypadku dobierany jest czworokątny region, z którego dwa dolne punkty są narożnikami ramki obrazu z kamery, a dwa górne ustawiam w zależności od rozdzielczości nagrania. Oto efekt:



Rysunek 4: Obraz z ROI

3. Kolejny krok to zastosowanie probabilistycznej transformaty Hougha w celu wykrycia linii. W tym kroku po raz kolejny musimy zastanowić się nad odpowiednim doбором parametrów. Dzięki transformacji znajdujemy odcinki, z których zbudowane są lewy i prawy pasy ruchu. W celu ich odróżnienia - zakładamy, że odcinki po lewej stronie połowy szerokości ramki to lewa linia pasa ruchu, a po prawej to prawa linia pasa ruchu.

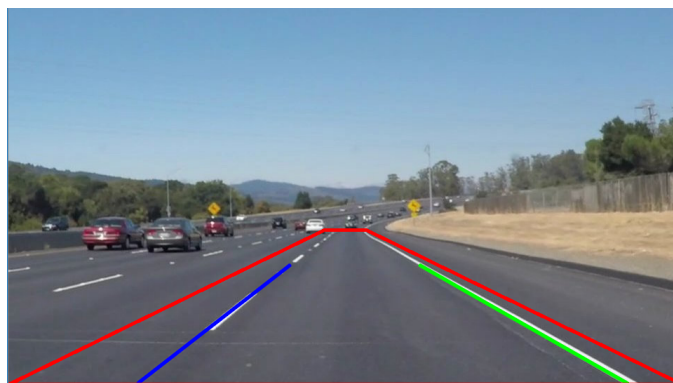


Rysunek 5: Transformata Hougha naniesiona na obraz początkowy

4. Ostatnim krokiem jest rysowanie lewej i prawej linii pasa. W przypadku gdy pas ruchu z sąsiednim jest oddzielony linią przerywaną nie wystarczy narysowanie odcinków otrzymanych z transformaty Hougha, więc do punktów lewej i prawej linii dopasowuję krzywe liniowe, które są krawędziami pasa ruchu. W tym punkcie należałoby się zastanowić czy innego rodzaju krzywe nie byłyby lepszym rozwiązaniem. Krzywe wyznaczające pas ruchu rysowane są na podstawie punktów ustalonych na podstawie rozdzielczości nagrania i ROI - można się zastanowić nad składaniem tych krzywych(ograniczających pas ruchu) z większej ilości punktów.



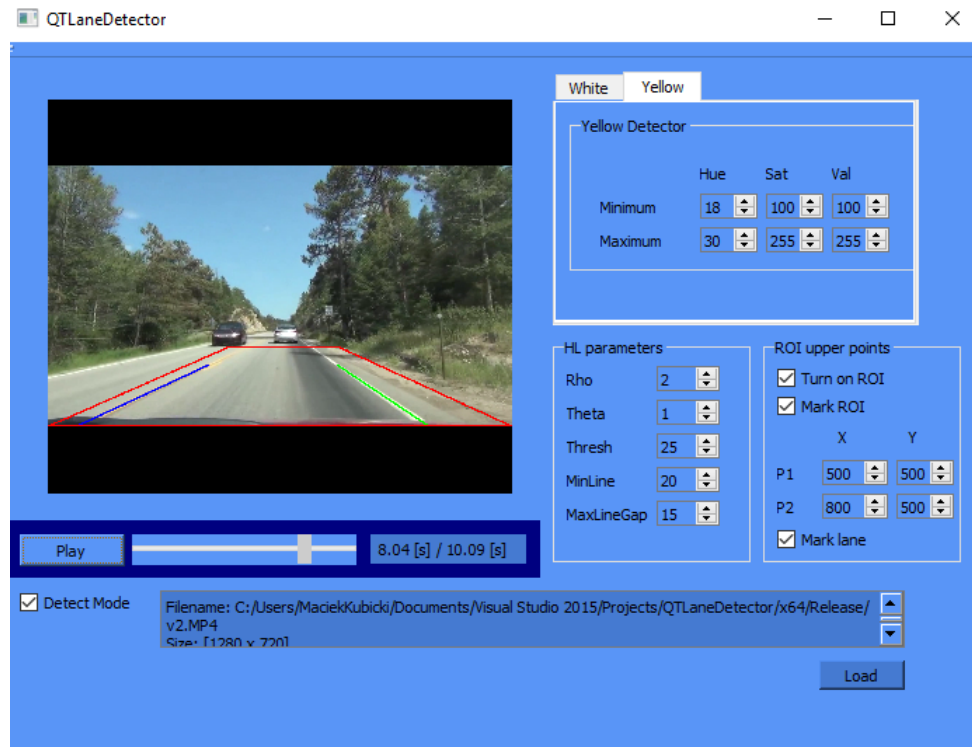
Rysunek 6: Wykryty pas ruchu



Rysunek 7: Wykryty pas ruchu z zaznaczeniem ROI

2.2 Aplikacja

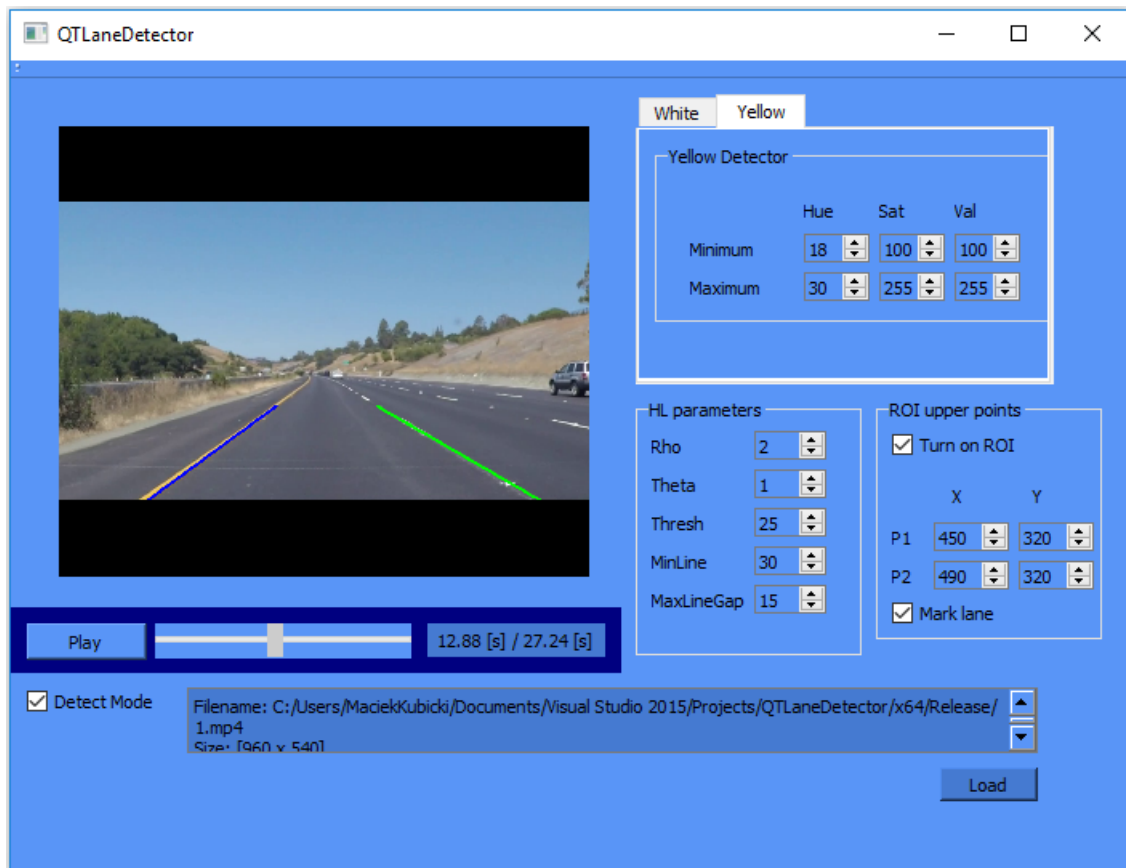
W ramach projektu przygotowałem prostą aplikację przy pomocy, której możemy przetestować różne parametry pojawiające się przedstawionym powyżej algorytmie. Możemy ustawić odpowiednie progi przy wydobywaniu koloru białego i żółtego. Możemy odpowiednio dobrać parametry transformaty Hougha, a także dobrać dwa górne punkty ROI. W interfejsie mamy też 4 checkboxy - pierwszy **Detect Mode** uruchamia transformatę Hougha, kolejny **Turn on ROI** uruchamia ROI, następny **Mark ROI** zaznacza ROI, a ostatni **Mark lane** rysuje dopasowaną krzywą do wyniku transformaty Hougha.



Rysunek 8: Wykryty pas ruchu

Interfejs pozwala zmieniać parametry jedynie, gdy odtwarzania jest zatrzymane. W interfejsie zawarte są mechanizmy walidacji parametrów wprowadzanych przez użytkownika. Domyślne parametry dają niezłe rezultaty w przypadku filmików 1 i 2 dołączonych do projektu. W przypadku 3, który ma inną rozdzielczość należy zmodyfikować wartości w sekcji **ROI upper points**(przykład na dole). To samo dotyczy się filmików v1 i v2(również przykłady na dole).

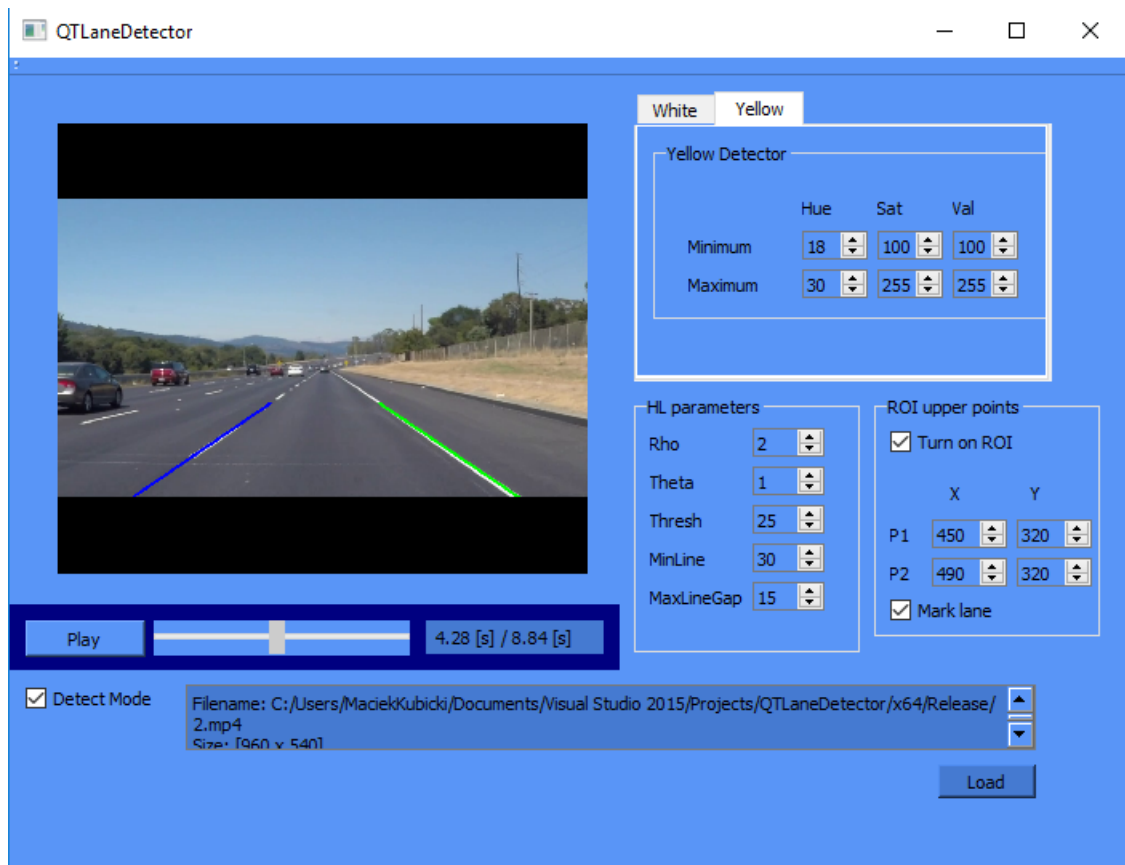
3 Kilka screenów wyników



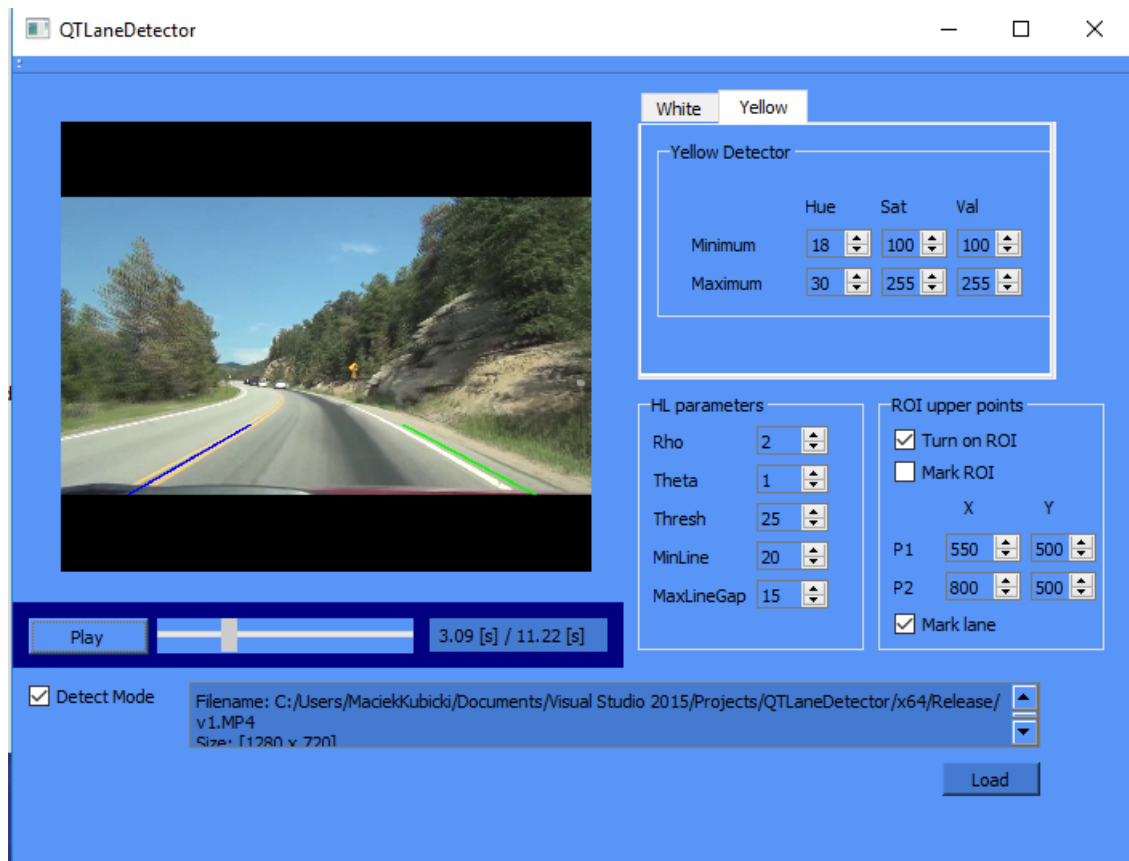
Rysunek 9: Wykryty pas ruchu



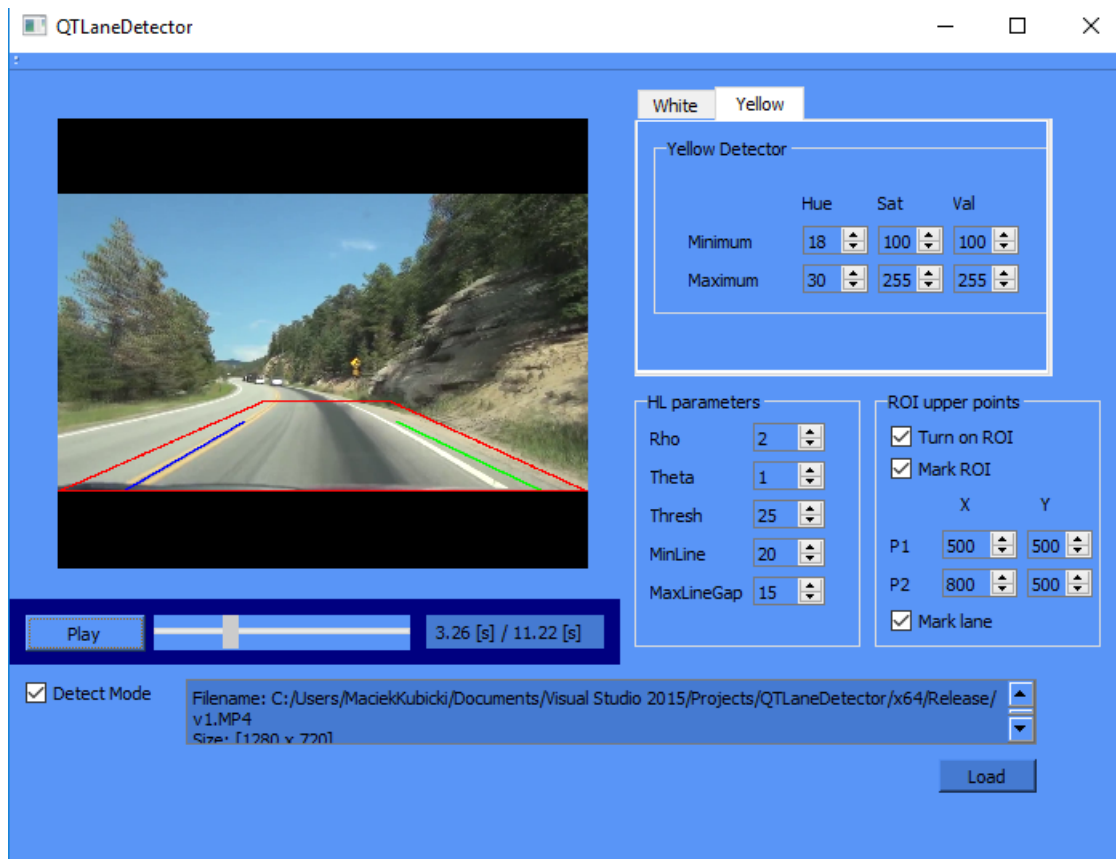
Rysunek 10: Wykryty pas ruchu + parametry dla 3



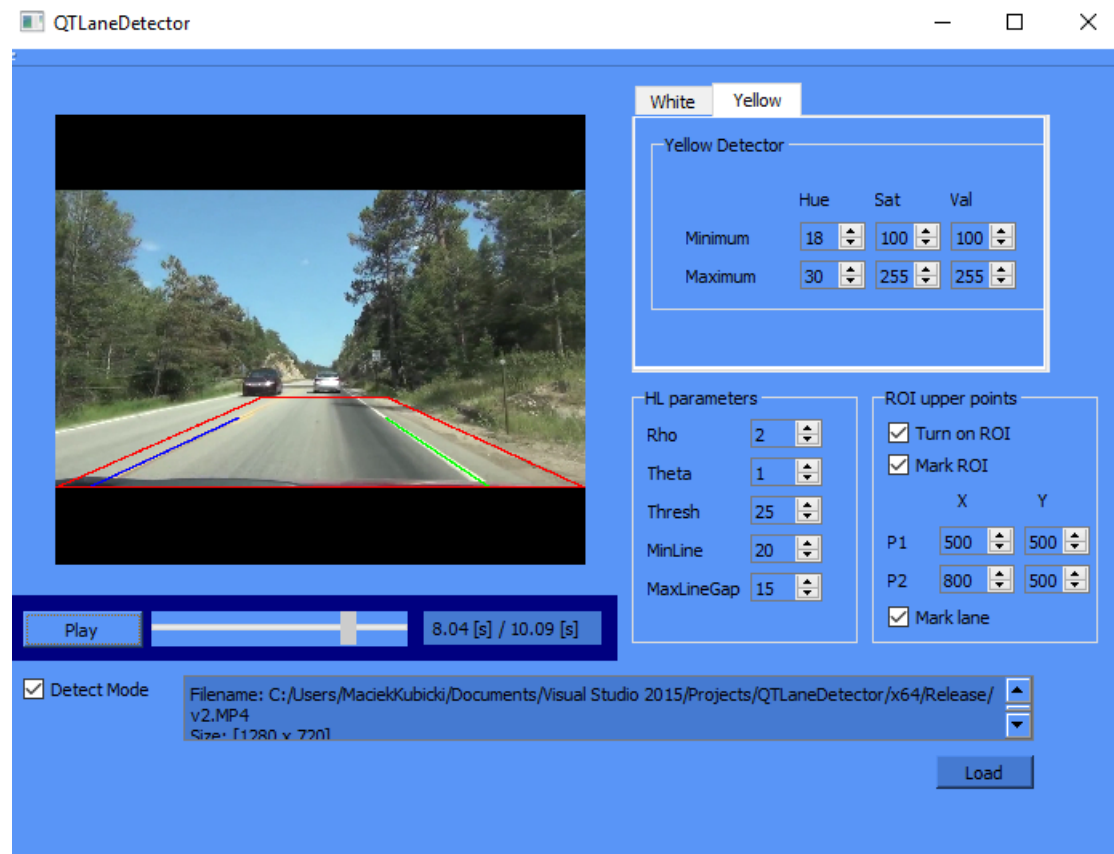
Rysunek 11: Wykryty pas ruchu



Rysunek 12: Wykryty pas ruchu + parametry v1



Rysunek 13: Wykryty pas ruchu + parametry v1



Rysunek 14: Wykryty pas ruchu + parametry v2

4 Wnioski

Podsumowując otrzymane efekty są w mojej ocenie dobre. Wszystkie załączone pliki wideo dają w miarę poprawne wyniki. W przypadku wideo **v1** pod koniec rysowanie lewej krzywej jest zaburzone poprzez wykrycie maski - to pokazuje, że miejsce umieszczenie kamery jest ważne i powinno być dobrze przemyślane. Widzę kilka punktów w których można rozwinąć projekt. Pierwszym z nich jest wykrywanie aut, które można by oznaczyć na obrazie wyjściowym, jednak ważniejsze byłoby odjęcie ich w pierwszym kroku zaprezentowanego wyżej algorytmu. Kolejnym pomysłem jest usprawnienie polegające na ustalaniu punktów ROI jedynie na podstawie rozdzielczości nagrania, jednak to byłoby ciężkie ze względu na to, że kamera może być ustawiona na wiele różnych sposobów. Wydaje mi się, że ciekawą opcją byłoby dodanie możliwości w GUI do przechwyty obrazu z kamery i przetestowaniu projektu w warunkach rzeczywistych.