Raport z przedmiotu: Rozpoznawanie obrazów
Prowadzący: dr hab. inż. Krzysztof Krawiec

Temat: Speed Camera

Roman Kamyk, inf66257, ISWD <roman.kamyk@gmail.com>
Mateusz Pachocki, inf66295, ISWD <mateusz.pachocki@gmail.com>

13 czerwca 2007
zajęcia: śr. 11:45, s. 45

1 Cel projektu

Celem projektu było stworzenie aplikacji, która na podstawie sekwencji wideo będzie dokonywała pomiaru predkości poruszających się obiektów - pojazdów. Prędkość pojazdu będzie estymowana jedynie na podstawie informacji zawartej w obrazie wideo.

Najważniejszym kryterium oceny poprawności działania aplikacji jest błąd pomiaru. W tym celu konieczne jest dysponowanie testowym obrazem wideo, dla którego znana jest prędkość poruszających się pojazdów.

Danymi niezbędnymi do realizacji projektu są sekwencje wideo, na których widoczne będą pojazdy w ruchu. Dobre rezultaty otrzymuje się dla sekwencji wideo w skali szarości, w rozdzielczości 320x240 przy trzech klatkach na sekundę. Z tego wynika, że w celu ustalenia prędkości poruszającego się pojazdu nie trzeba koniecznie dysponować obrazem wideo wysokiej jakości. W celu pewnego ułatwienia, do testów wykorzystane zostały sekwencje, w których kamera znajduje się na pewnej wysokości nad trasą pojazdów. W ten sposób uzyskujemy odpowiednio długi tor ruchu oraz odpowiednią wielkość śledzonych obiektów.

2 Osadzenie projektu w dziedzinie rozpoznawania obrazów

W realizacji projektu zostało zastosowane podejście image-driven (inaczej bottom-up), ponieważ na podstawie obrazów pozyskanych z sekwencji wideo tworzymy opis ich zawartości za pomocą algorytmów detekcji krawędzi, czy też śledzenia poruszających się obiektów. W projekcie nie ma elementów uczenia maszynowego, gdyż nie było takiej potrzeby. Pozyskana z obrazu informacja o jego cechach służy wyłącznie do obliczeń matematycznych, których wynikiem jest prędkość poruszającego się pojazdu.

3 Zastosowane podejście

Podejście zastosowane przy realizacji projektu polegało na wykryciu poruszającego się obiektu, a następnie przypisanie temu obiektowi śledzonych punktów (z ang. *trackers*). Punkty te są śledzone za pomocą algorytmu Lucas-Kanade. Mając co najmniej 8 punktów, które nie są współpłaszczyznowe, można dokonać estymacji prędkości za pomocą algorytmu *Eight Point*. Szczegółowe działanie algorytmu oraz jego implementacja zostały opisane w rozdziale 6.

4 Trudności

Głównymi trudnościami w realizacji projektu były:

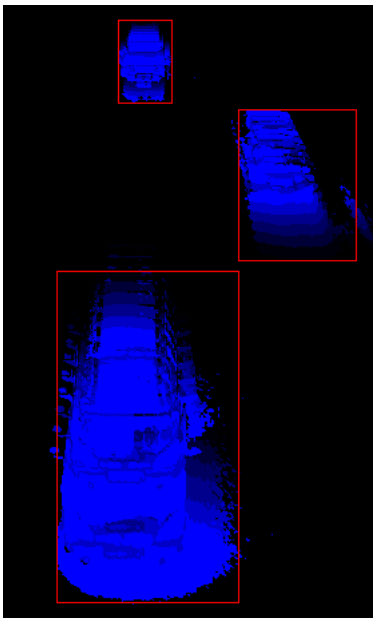
- brak dokładnej wiedzy o odległości pojazdu od kamery,
- zniekształcenia wynikające z przektałcenia perspektywicznego,
- szum w obrazie wideo wynikający z technologii jego pozyskiwania,
- cień poruszającego się pojazdu.

5 Środowisko realizacji projektu

Projekt został napisany w języku *Python* z wykorzystanie biblioteki *OpenCV*. Projekt nie posiada graficznego interfejsu użytkownika, gdyż użytkownik nie ma żadnego wpływu na parametry aplikacji ani na sposób jej działania. Wyświetlana jest oryginalna sekwencja wideo z zaznaczonymi rozpoznanymi pojazdami, punktami śledzonymi oraz wektorem przemieszczenia.

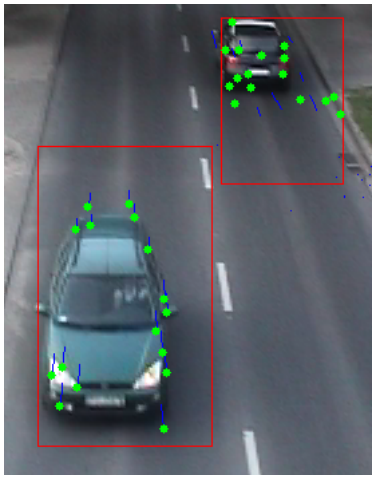
6 Implementacja

Pierwszym etapem w estymacji prędkości było wykrycie pojazdów w sekwencji wideo. W tym celu użyliśmy zwykłej różnicy pomiędzy kolejnymi klatkami, a obraz wynikowy poddaliśmy dodatkowo binaryzacji tak, aby widoczne były tylko poruszające się obiekty. Obrazy te zostały też poddane filtrowaniu Gaussowskiemu w celu usunięcia szumów. W ten sposób otrzymaliśmy *motion history image* (MHI), gdzie jasność piksela odpowiada nowości w czasie zajścia ruchu – rys. 1.



Rysunek 1: Obraz uzyskany za pomocą *motion history image*.

Mając już znalezione poruszające się obiekty, otoczyliśmy je prostokątami, tworząc coś w rodzaju *region of interest* (ROI) – rys. 2. Zabieg ten jest konieczny przy doborze punktów do śledzenia. Punkty dobierane są za pomocą funkcji z biblioteki *OpenCV – FindGoodFeaturesToTrack*. Dzięki zastosowaniu ROI punkty nie są wyszukiwane w całym obrazie, a jedynie w miejscach, które nas interesują. W kolejnych iteracjach (dla kolejnych klatek) dokonywane jest jedynie sprawdzenie, czy dla danego ROI zostały znalezione punkty do śledzenia. Jeśli takich punktów nie ma lub ich liczność jest mniejsza niż 8, to dokonuje się ponownego poszukiwania punktów.



Rysunek 2: Pojazdy otoczone ROI. Kolorem czerwonym oznaczone są ROI, zielonym punkty śledzone, niebieskim wektor przesunięcia.

Gdy dysponujemy *trackerami* w ROI, możemy przystąpić do śledzenia ich w sekwencji wideo. Do tego celu został wykorzystany algorytm Lucas-Kanade, w wersji oferowanej przez bibliotekę *OpenCV*. Kiedy posiadamy pozycje punktów z klatki aktualnej oraz poprzedniej, możemy przystąpić do oszacowania prędkości. Jak już zostało wspomniane, w tym celu korzystamy z algorytmu *Eight Point*. Nazwa algorytmu pochodzi od założenia, że do poprawnego działania potrzebne jest co najmniej 8 punktów. Schemat działania algorytmu jest następujący:

1. Konstruowany jest system składający się z n punktów, gdzie $n \geqslant 8$. Niech macierz A będzie macierzą współczynników o wymiarze $n \times 9$ oraz $A = UDV^T$ będzie rozkładem SVD macierzy A.
2. Wartości macierzy F (*fundamental matrix*) są składowymi kolumn V odpowiada-

jącymi najmniejszym wartościom własnym macierzy A.

3. Obliczany jest rozkład na wartości osobliwe macierzy F wg wzoru:

$$F = UDV^T \tag{1}$$

4. Najmniejsza wartość osobliwa na przekątnej macierzy D ustawiana jest na 0. Niech D' będzie poprawioną macierzą.

5. Poprawiona estymata F, F', dana jest wzorem:

$$F' = UD'V^T \tag{2}$$

Liczenie macierzy *fundamental* w ten sposób jest jednak dość kłopotliwe i obarczone błędami numerycznymi, głównie ze względu na rozkład SVD. Dlatego stosujemy funkcję z biblioteki *OpenCV – cvFindFundamentalMat*. Funkcja ta oblicza macierz fundamentalną dla danych punktów przed i po przesunięciu. Do obliczenia tej macierzy używana jest jedna z czterech dostępnych metod (w zależności od dostępnej liczby punktów – n):

- algorytm *Seven Point* dla $n = 7$,
- algorytm *Eight Point* dla $n \geqslant 8$,
- algorytm *RANSAC* dla $n \geqslant 8$,
- algorytm *LMedS* dla $n \geqslant 8$.

Algorytmy te były testowane na specjalnej sekwencji, na której sześcian poruszał się ruchem jednostajnym w jednym kierunku. Algorytm *Seven Point* dawał najsłabsze rezultaty, otrzymane macierze różniły się bardzo znacznie od siebie. Pozostałe algorytmy dały bardzo zbliżone rezultaty, więc zdecydowaliśmy się na opisany powyżej algorytm *Eight Point* ze względu na przyzwoite wyniki i szybkość działania.

Posiadając macierz F możemy wyznaczyć macierz E (*essential matrix*). Wyjaśnienia wymaga znaczenie obu tych macierzy. Dla *essential matrix* zachodzi równanie:

$$p_r^T E p_l = 0 \tag{3}$$

p_r oraz p_l są wektorami punktów w przestrzeni kamery. Dla *fundamental matrix* zachodzi następujące równanie:

$$\hat{p}_r^T F \hat{p}_l = 0 \tag{4}$$

W tym przypadku $\hat{p}_r$ oraz $\hat{p}_l$ są wektorami współrzędnych punktów w przestrzeni obrazu. Między tymi macierzami zachodzi zależność:

$$F = M_r^{-T} E M_l^{-1} \tag{5}$$

Macierze M_r oraz M_l są macierzami wewnętrznych parametrów kamery.

Gdy za pomocą algorytmu *Eight Point* mamy wyznaczoną macierz F, to korzystając z zależności 5 możemy wyznaczyć macierz E. *Essential matrix* posiada następującą właściwość:

$$E^T E = \begin{bmatrix} T_y^2 + T_z^2 & -T_x T_y & -T_x T_z \\ -T_y T_x & T_z^2 + T_x^2 & -T_y Tz \\ -T_z T_x & -T_z T_y & T_x^2 + T_y^2 \end{bmatrix} \tag{6}$$

Elementy tej macierzy to wartości przemieszczeń punktu w płaszczyźnie xyz. To jeszcze nie umożliwia wyznaczenia prędkości poruszającego się obiektu. W tym celu macierz daną równaniem 6 należy poddać normalizacji. Z 6 można wyznaczyć ślad EE^T:

$$Tr(E^T E) = 2\|\mathbf{T}\|^2 \tag{7}$$

Stąd podzielenie każdego elementu *essential matrix* przez:

$$N = \sqrt{Tr(E^T E)/2} \tag{8}$$

będzie równoważne z normalizacja wektora przesunięcia do wektora jednostkowego. Używając normalizacji danej równaniem 8 można zapisać następującą zależność:

$$\hat{E}^T \hat{E} = \begin{bmatrix} 1 - \hat{T}_x^2 & -\hat{T}_x \hat{T}_y & -\hat{T}_x \hat{T}_z \\ -\hat{T}_y \hat{T}_x & 1 - \hat{T}_y^2 & -\hat{T}_y \hat{T}_z \\ -\hat{T}_z \hat{T}_x & -\hat{T}_z \hat{T}_y & 1 - \hat{T}_z^2 \end{bmatrix} \tag{9}$$

Posiadając znormalizowaną macierz E możemy wyliczyć prędkość z elementów na jej przekątnej.

Jak już zostało wspomniane w rozdziale „Środowisko realizacji projektu", całość została zaimplementowana w języku *Python* z wykorzystaniem biblioteki *OpenCV*. Poniżej zostały wymienione i krótko opisane najważniejsze funkcje aplikacji:

update_mhi – dokonuje aktualizacji *motion history image*. Obraz ten musi być aktualizowany w każdej klatce, aby nie utracić informacji o żadnym z poruszających się obiektów. Jako parametry przyjmuje aktualną klatkę oraz próg binaryzacji obrazu.

find_points_in_roi – znajduje punkty do śledzenia w danym *region of interest*. Oprócz wspomnianej już wcześniej funkcji *cvGoodFeaturesToTrack* wykorzystana jest tutaj również funkcja *cvFindCornerSubPix*, aby znaleźć narożniki obiektu, ponieważ są one najlepszymi punktami do śledzenia. Jako parametry przyjmuje obraz w skali szarości oraz wymiary ROI.

trace_points – dokonuje śledzenia znalezionych wcześniej punktów za pomocą algorytmu Lucas-Kanade.

clean_up – usuwa *trackery*, które w jakiś sposób znalazły się poza ROI. Może to być spowodowane cieniem pojazdu lub niedokładnościami algorytmu śledzącego.

compute_fund – oblicza macierz fundamentalną na podstawie przekazanych punktów.

compute_translation – oblicza wektor przesunięcia na podstawie macierzy *essential*.

compute_speed – dokonuje wszystkich opisanych w tym rozdziale przekształceń matematycznych oraz zbiera cząstkowe wyniki z pozostałych funkcji.

7 Wnioski

Algorytm zastosowany do estymacji prędkości jest algorytmem, który sprawdza się głównie w stereowizji. Jednak przy traktowaniu dwóch kolejnych klatek jak obrazów z dwóch różnych kamer, można go z powodzeniem stosować także do estymacji prędkości poruszających się obiektów. Na niedokładności pomiarów mają wpływ głównie obliczenia numeryczne, gdyż zastosowane podejście wymaga wielu operacji macierzowych, które, jak np. rozkład SVD, są dość niestabilne numerycznie. Dzięki temu, że aplikacja została napisana w języku *Python* jej testowanie, czy wprowadzanie zmian było bardzo szybkie.

8 Proponowane ulepszenia

Aby uzyskać bardziej stabilne wyniki przesunięcia można by zastosować liczenie średniego przesunięcia w danej klatce obiektu i aktualizowanie go w kolejnych, ewentualnie odrzucając wyniki znacznie odstające (powiedzmy o rząd wielkości).

Alternatywnym podejściem mogła być estymacja prędkości na podstawie przesunięcia w pikselach wyliczonego w przestrzeni obrazu, a następnie odwrócenie transformacji perspektywicznej w celu przeniesienia tego przesunięcia do przestrzeni kamery. To podejście wymagałoby mocnych założeń w postaci znajomości odległości niektórych punktów obrazu od kamery. W najprostszym przypadku, gdy kamera widzi jedynie jedną płaszczyznę — z drogą, wystarczy znajomość skrajnych widocznych punktów asfaltu i odległości między nimi. Posiadając taką wiedzę można przekształcić wyliczone przesunięcie, a następnie przeskalować je, aby uzyskać rzeczywistą prędkość.