Projekt autonomicznego robota rozpoznającego znaki.

Karolina Bożek, Magdalena Pęksa

20 czerwca 2017

Spis treści

1 Wstęp

1.1 Ogólna charakterystyka implementowanego modelu.

Celem naszego projektu było porównanie algorytmów uczących się rozpoznawania znaków drogowych w warunkach symulujących te panujące na drodze, kiedy kamera rejestrująca obraz przemieszcza się, obiekty mogą być częściowo niewidoczne a oświetlenie niesprzyjające detekcji koloru.
Po wstępnej obróbce zdjęcia, w skład której wchodziły metody wymienione w publikacji, takie jak Color Enhancement, Gaussian Blur oraz binaryzacja, przeprowadzone zostały dwa etapy klasyfikacji.
Wpierw użyto liniowego klasyfikatora SVM, którego celem było rozpoznanie kształtu (w naszym przypadku koła, ponieważ model został wytrenowany na znakach nakazu). W kolejnym etapie do rozpoznawania konkretnego znaku (nakaz jazdy w prawo, lewo; nakaz jazdy prosto) został wykorzystany algorytm Random forest oraz k-neighbours z algorytmem k-d-trees.
Klasyfikatory uczono przy użyciu deskryptora HOG, histogramy przedstawiające rozkład gradientów tzn rozkład orientacji krawędzi.
Zbiór trenujący składał się z autentycznych zdjęć wykonanych w warunkach drogowych, testowanie natomiast polegało na przetwarzaniu klatek z kamery zamocowanej do robota, w czasie rzeczywistym w różnych warunkach oświetlenia.

2 Opis danych

Zbiór treningowy modelu to niemiecka baza znaków drogowych GTSRB.
http://benchmark.ini.rub.de/?section=gtsrb&subsection=dataset
Składa się ona z ponad 50 tysięcy obrazków, podzielona jest na sety zawierające po 30 zdjęć danego znaku. Na każdym z nich znajduje się dokładnie jeden znak, którego powierzchnia zajmuje co najwyżej 90% całkowitej powierzchni obrazka. Warunek ten umożliwia zastosowanie metod opartych o detekcję krawędzi (co najmniej 5 pikseli marginesu). Nie jest natomiast konieczne, aby znak znajdował się w centralnej części obrazka. Obrazki mają różne rozmiary, najmniejszy to 15x15 pikseli, natomiast największy 250x250 pikseli. Zapisane są w formacie PPM (Portable Pixmap P6).

Rysunek 1: Przykład obrazka wykorzystywanego w naszym zbiorze trenującym.

3 Opis użytych metod

Dane zostały poddane następującemu procesowi obróbki: segmentacja, detekcja oraz klasyfikacja. Finalnie proces segmentacji polega na Color Enhancement dla koloru niebieskiego (kolor znaków nakazu) oraz innych filtrów morfologicznych, mających na celu jak najdokładniejsze wyszczególnienie obszaru zajmowanego przez znak drogowy.

Zbinaryzowany obraz przekazywany jest dalej do modułu odpowiedzialnego za wyodrębnienie znalezionego obszaru (algorytm Sliding Window wraz z Pyramids).
Następnie pozyskane dane zostają wykorzystane do stworzenia histogramów, na których uczony jest trenowany model.
Wytrenowano klasyfikatory zarówno do rozpoznawania kształtu znaku jak i do klasyfikacji obiektu na jego powierzchni (zwrot strzałki). Podobnie jak w wymienionej publikacji przetestowane zostały różne filtry morfologiczne takie jak Color Enhancement, Chromatic Filter, Top-hat oraz Bottom-hat.

Color Enhancement polega na ekstrakcji pikseli, w których dominuje jeden z kanałów RGB. W przypadku znaków nakazu, jest to kolor niebieski (0, 0, 255). Zastosowanie Chromatic Filter ma na celu wyostrzenie konturu kształtu na powierzchni znaku. Algorytm top-hat wyodrębnia jasne piksele z dużym kontrastem w stosunku do lokalnego oświetlenia, bottom-hat działa analogicznie, ale dla ciemnych pikseli.
Zasadniczo oba algorytmy polegają na zastosowaniu erozji oraz dylatacji.

Dla każdego piksela RGB $x = \{x_R, x_G, x_B\}$ w obrazie:

$$f_R(x) = max(0, \frac{min(x_G - x_B, x_B - x_R)}{s}) \text{ gdzie } s = x_R + x_G + x_B \tag{1}$$

Wzór na Color Enhancement użyty w projekcie 2.

$$f(R, G, B) = \frac{\|R - G\| + \|G - B\| + \|B - R\|}{3D} \tag{2}$$

gdzie D jest poziomem ekstracji koloru achromatycznego
Wzór na Chromatic filter, w projekcie użyto $D = 20$ oraz ekstrakcji $f(R, G, B) > 1$ 2.

$$I_T = I - I_o,\ I_o = I_D \circ I_E \tag{3}$$

gdzie I_o to operacja morfologicznego otwarcia obrazu
I to obraz wejściowy
I_D to operacja dylatacji
I_E to operacja erozji
Wzór na Top hat użyty w projekcie4.

$$I_B = I_C - I,\ I_C = I_E \circ I_D \tag{4}$$

gdzie I_o to operacja morfologicznego otwarcia obrazu
I to obraz wejściowy
I_D to operacja dylatacji
I_E to operacja erozji
Wzór na Top bottom użyty w projekcie4.

Naszym celem było przetestowania zaimplemetowanych modeli w praktyce.
Do tego posłużył nam robot jeżdżący NXT mindstorm wraz z komórką, która została użyta jako kamera sieciowa łącząca się z naszym programem.
W związku z duża ilością, kosztowanych operacji naszego modelu, klatki z kamery były odczytywane w osobnym wątku.
Korzystałyśmy przy tym z biblioteki nxt-python. Nasz robot łączył się z komputerem za pomocą kabla usb, na którym wcześniej odpaliliśmy serie treningową.
W celu swobodnego ruchu robota według zakwalifikowanych znaków program sterował dwoma silnikami nxt odpowiednio na portach A i C.
Metody $nxt.SynchronizedMotors(self.motor_c, self.motor_a, 100)$
oraz $nxt.SynchronizedMotors(self.motor_a, self.motor_c, 100)$
ustawiają moc odpowiednio silnikowi C oraz A i mogąc wywołać odpowiednio skręt w lewo oraz w prawo przy zastosowaniu metody $nxt.turn()$.

Rysunek 2: Konstrukcja hardwaru.

4 Ewaluacja wyników

Tablica 1: Porównanie wyników dla algorytmów RandomForest oraz K-neighbours

Random forest 32	Random forest 50	K-neighbours 32	K-neighbours 50
Straight : Straight	Left : Left	Left : Right	Left : Left
Right : Right	Straight : Straight	Right : Right	Straight : Left
Straight : Straight	Right : Straight	Right : Right	Straight : Straight
Right : Right	Right : Right	Right : Right	Straight : Straight
Right : Left	Right : Right	Left : Left	Right : Right
Straight : Straight	Straight : Straight	Right : Right	Left : Left
Right : Left	Left : Left	Straight : Straight	Straight : Straight
Straight : Straight	Left : Left	Right : Right	Right : Right
Left : Left	Right : Straight	Right : Right	Straight : Straight
Right : Right	Left : Left	Left : Right	Straight : Straight
80 %	80%	80%	90%

Powyższe wyniki testowane są w optymalnych warunkach oświetleniowych oraz kąta widzenia robota. Przy zmiennym oświetleniu (tzn rożnym ułożeniu cień) dokładność wyników spadała nawet do 50 %.

5 Demonstracja wyniku i działania programu:

https://github.com/pymagda/ML_project/blob/master/nxt_signs.mp4

6

Literatura

[1] http://www.sciencedirect.com/science/article/pii/S0921889012001236