Mgr inż. Szymon Moliński
POLITECHNIKA OPOLSKA, WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI I INFORMATYKI,
INSTYTUT UKŁADÓW ELEKTROMECHANICZNYCH I ELEKTRONIKI PRZEMYSŁOWEJ

Zastosowanie algorytmów OCR i zewnętrznych baz danych w zagadnieniach komunikacji robota humanoidalnego NAO

Streszczenie: *Artykuł prezentuje program rozwijający komunikację między robotami humanoidalnymi a ludźmi. Cel badań został osiągnięty za pomocą algorytmów pozwalających na odczytywanie wydrukowanych tekstów i wykonywanie na ich podstawie założonych czynności. Wiąże się to z wykorzystaniem konkretnych technik przetwarzania obrazów, jak: wyodrębnianie cech, rozpoznawanie wzorców i ich klasyfikacja oraz z komunikacją między robotem a zewnętrzną bazą danych.*

The OCR algorithms and external databases usage in the subject of humanoid NAO robot communication

Summary: *This article presents the program which expands the robot-human communication level. The aim of research was achieved by usage of the algorithms for reading printed information and performing actions based on the read text. It is connected to particular image processing fields like feature extraction, pattern recognition and classification as well as to the communication between robot and external database.*

WPROWADZENIE

Jednym z podstawowych zmysłów człowieka jest wzrok. Nasz gatunek rozróżnia barwy z wąskiego pasma spektrum elektromagnetycznego, nazywanego światłem widzialnym. Większość wytworów techniki i kultury, które niosą za sobą informacje, jest tworzona w taki sposób, by potrzebne informacje można było odczytywać przy pomocy zmysłu wzroku.

Wykonanie zdjęcia lub sfilmowanie obiektu przez robota jest odpowiednikiem widzenia, jednakże samo odczytanie i zakodowanie informacji do postaci dyskretnej w większości przypadków nie można uznać za ostatnią fazę „widzenia" przy naśladowaniu zmysłu wzroku człowieka [10]. Istotny jest również kontekst i informacja stojąca za danymi znakami lub obrazami. W zależności od typu robota dane odbierane przez systemy wizyjne powinny prowadzić do wybranych akcji, na przykład wykonania polecenia zapisanego na karcie albo naliczenia ceny na podstawie kodu kreskowego. Przykładem jest również użycie specjalnych kart z poleceniami w celu zasygnalizowania robotowi, co ma wykonać pod nieobecność właściciela [7]. Wykorzystanie systemów *OCR* (*Optical Character Recognition*) przez roboty przemysłowe jest już również realizowane [9].

Reasumując można wyróżnić kilka aspektów uzasadniających naukę czytania robotów: **zdrowotny** – związany z pomocą dla osób niedowidzących i niewidomych, gdzie przykładowo analizuje recepty lekarskie sprawdzając poszczególne leki w medycznych bazach danych [1]; **pracy**, gdzie robot odczytuje polecenia ze znaków w swoim otoczeniu [7] oraz **rozrywkowy**.

W artykule przedstawiono wykorzystanie przetwarzania obrazów i łączenia się robota z internetowymi bazami danych w aspekcie rozrywkowym, jednakże na bazie opracowanych algorytmów można go z powodzeniem przekształcić tak, żeby robot wykonywał również inne zadania.

ROBOT NAO I NARZĘDZIA PROGRAMISTYCZNE

Humanoidalny robot *NAO* został opracowany przez firmę *Aldebaran Robotics* z Francji. Premiera robota nastąpiła w 2006 roku, do 2015 roku sprzedano około 5000 sztuk robotów *NAO*, które używane są w wielu instytucjach edukacyjnych i badawczych na całym świecie [8]. Sukces *NAO* wziął się najprawdopodobniej z możliwości jakie oferuje robot: posiada aż 25 stopni swobody, akcelerometry, czujniki siły nacisku, sonar, dwie kamery HD, emitery IR, cztery mikrofony i głośniki, przy tym swój własny system operacyjny (*NAOqi* oparty na Linuksie) i narzędzia umożliwiające programowanie

wysokopoziomowe, oparte na metodzie *przeciągnij i upuść* oraz niskopoziomowe, wspierające najpopularniejsze języki programowania, gdzie możliwe jest tworzenie własnych algorytmów niezdefiniowanych w *firmware* [8]. Na komercyjny sukces robota mógł przełożyć się również *design* robota, ponieważ przypomina on humanoidalne dziecko (rys. 1). Najważniejszymi elementami sprzętowymi robota ze względu na rodzaj postawionego problemu są dwie kamery. Obie mają taką samą specyfikację, przedstawioną w dokumentacji robota. Możliwe jest ustawienie wielu parametrów związanych ze wspomaganiem sprzętowym albo programowym w celu uzyskania najlepszych ujęć w zależności od warunków oświetlenia, tła czy dostępnej pamięci robota. Ułatwia to zagadnienia takie jak odszumianie obrazu, jednakże nie zniwelowało całkowicie etapu filtracji obrazu.

Rys. 1. Robot NAO

Przetwarzanie zdjęć wykonanych przez robota dokonywało się w głównej mierze przy wykorzystaniu narzędzia jakim jest biblioteka *OpenCV,* która znacznie przyspiesza obróbkę i analizę wielowymiarowych sygnałów. Najważniejszym etapem analizy jest rozpoznawanie znaków, liter, w skrócie nazywane *OCR* (*Optical Character Recognition*) [5]. W zależności od danych wejściowych (ich typu reprezentacji) można wyróżnić kilka podstawowych metod ekstrakcji. W tabeli 1 zebrano najważniejsze z nich – podkreślono te, które wykorzystano w trakcie realizacji algorytmów. Robot *NAO* wykonywał zdjęcia kolorowe lub w odcieniach szarości, dlatego doprowadzenie obrazu do postaci binarnej wymagało wstępnej filtracji i odszumiania oraz progowania (*threshold*). Operacje ułatwiał fakt, że fraza zapisana była czarnymi literami na białym tle, co daje duży kontrast (rys. 2). Posiadając dane w postaci binarnej przechodzi się do etapu wyodrębniania pojedynczych obiektów (wypełnionych znaków) i ich porównywania z wzorcami.

Rys. 2. Przykładowy tekst odczytywany przez NAO

Wzorcem mogą być same obiekty, wtedy porównuje się piksel po pikselu jedną ilustrację (badaną) z bazową i oblicza się na tej podstawie średni błąd kwadratowy. Wzorcami mogą być również własności geometryczne liter – wybrane przez autora przedstawiono w dalszej części artykułu. W tabeli 1 zestawiono najważniejsze techniki pozwalające na ekstrakcję znaków w zależności od typu danych wejściowych: jako obrazy wypełnione (w odcieniach szarości lub czarno-białe) albo jako wektory (szkielety) poszczególnych liter.

Tabela 1

Metody ekstrakcji znaków w zależności od typu reprezentacji [2]

Ilustracja w odcieniach szarości	**Reprezentacja dwuwartościowa**		**Wektor (szkielet)**
	Wypełniony znak	**Kontur**	
Template matching	**Template matching**	-	Template matching
Deformable templates	-	-	Deformable templates
Unitary Transforms	Unitary Transforms		Graph description
	Projection histograms	Contour profiles	Discrete features
Zoning	Zoning	Zoning	Zoning
Geometric moments	**Geometric moments**	Spline curve	-
Zernike moments	Zernike moments	Fourier descriptors	Fourier descriptors

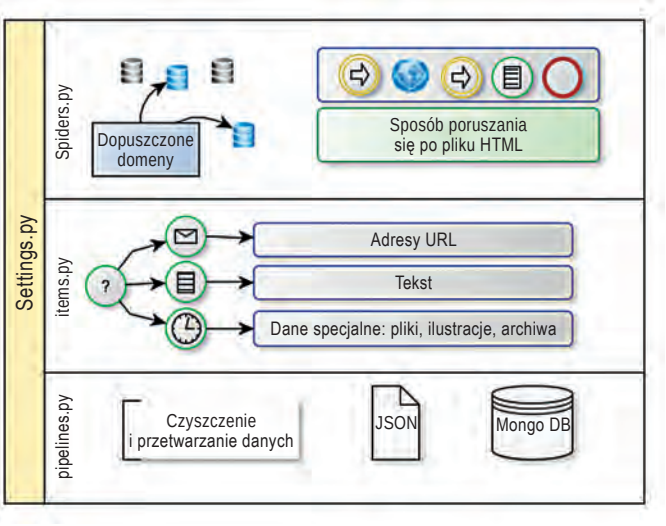
Znaki odtworzone w trakcie analizy obrazu rastrowego są następnie porównywane do ciągów znaków w bazie. Każdy ciąg może aktywować nowy algorytm robota (gdyby był to robot do pracy) albo może zostać poddany analizie w źródłach sieciowych (przykładowo składniki leku). Dla celów demonstracyjnych robot odczytywał z wydrukowanych tekstów tytuły filmów nawiązujących do robotyki, by następnie przeszukiwać internetowego bloga [XX] w poszukiwaniu recenzji, które ściągał do swojej pamięci

i wokalizował przy użyciu wbudowanego głośnika. Operację przeszukiwania zasobów internetu można przeprowadzić na wiele sposobów, ale podobnie jak *OpenCV* i język *C++* sprawdzają się przy przetwarzaniu obrazów tak przy przeszukiwaniu stron internetowych można wykorzystać bibliotekę *Scrapy* [XX] w języku *Python*.

Scrapy ułatwia *Web Scraping* [z angielskiego, dosłownie: „zeskrobywanie stron"] – to technika wydobywania informacji ze stron internetowych w celu ich dalszego przetwarzania i analizowania [11]. *Scrapy* umożliwia tworzenie aplikacji do przeglądania i pobierania treści ze stron internetowych. Cele, które mogą zostać osiągnięte przy pomocy tego narzędzia to na przykład archiwizacja danych, które pojawią się cyklicznie na danych serwisach, *data mining* albo badania naukowe.

Program w *Scrapy* składa się z kilku podstawowych elementów: *Spiders* lub *Crawlers*, czyli tzw. „Pająków", które wyszukują żądane przez nas informacje na stronach internetowych i poruszają się po nich (używając do tego standardu *xml*), *items,* czyli obiektów, które pobierane są ze stron oraz *pipelines,* czyli strumieni, którymi można ściągać dane na dysk. Wszystkie elementy spięte są przez plik *Setting,* czyli ustawienia (na przykład określających kolejność oraz typ ściąganej treści). Schematycznie przedstawiono to na rysunku 3.

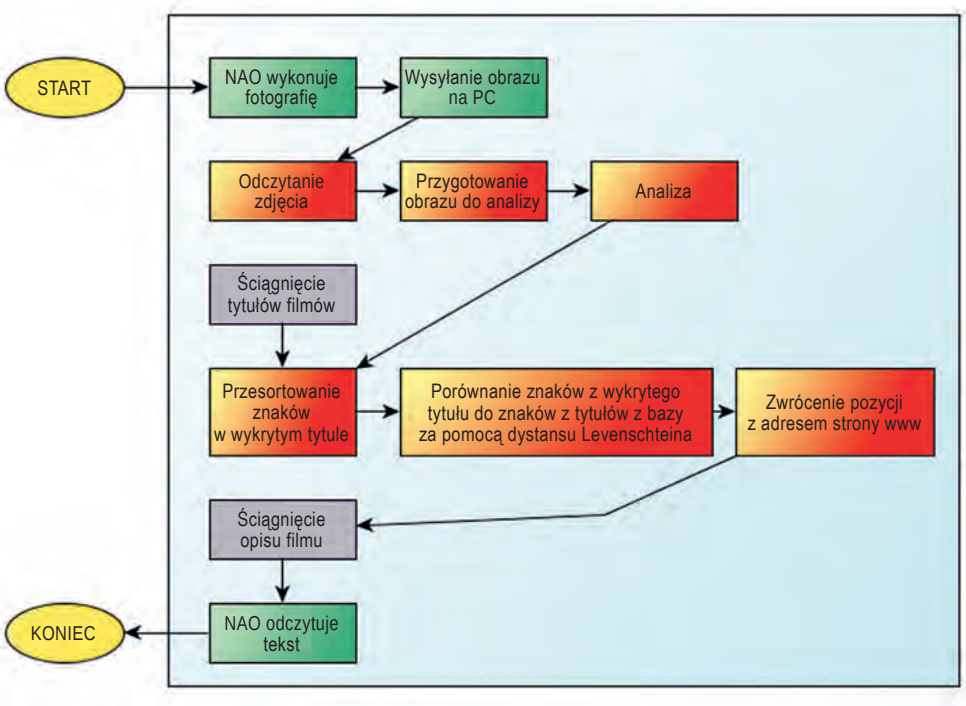
Ściągnięcie danych i ich odczytanie jest ostatnią fazą algorytmu, który zaprezentowano w następnym rozdziale.



Rys. 3. Zasada tworzenia skryptu do pobierania zawartości stron internetowych w Scrapy; grafika własna na podstawie dokumentacji do biblioteki

ALGORYTM

Program przebiega w kilku etapach. Pierwszy to inicjacja akcji wykonania fotografii przez robota, następnie wysłanie jej przez sieć bezprzewodową do jednostki obliczeniowej wykonującej pozostałe kroki. Zdjęcie jest przechwytywane, rozpoczyna się faza wstępnej obróbki, na którą składa się filtracja i odszumianie oraz ekstrakcja znaków – znaki są analizowane i sortowane według kolejności alfabetycznej. W międzyczasie ściągane są tytuły filmów



Rys. 4. Podstawowy algorytm odczytywania tekstu i wykorzystywania go jako klucza do baz danych

(polecenia) z bloga (bazy danych) i również są sortowane według alfabetu. Następuje faza porównania przesortowanych ciągów i na tej podstawie zwracana jest pozycja z adresem strony www z recenzją danego filmu (treść wykonywana). Jest ona pobierana i transferowana do pamięci robota *NAO*. Robot odczytuje głośno tekst. Schematycznie przedstawiono to na rysunku 4, gdzie pierwszy i ostatni wiersz algorytmu wykonywane są przez sprzęt i oprogramowanie robota, drugi i czwarty wiersz to operacje w *C++* i *OpenCV* (drugi) oraz *C++* i *STL* (czwarty), a wiersze trzeci i piąty związane są z operacjami *Web Scraping* i inicjowane są one przez skrypty *Pythona* i *Scrapy*.

Najistotniejsze elementy programu to:

- Algorytmy *OCR* – służące do obrazowania i przetwarzania tekstu z postaci dwuwymiarowej, rastrowej macierzy natężeń na ciąg znaków;
- Algorytmy *Web Scraping* – służące do automatycznej komunikacji z bazami danych w internecie i pozyskiwania informacji ze stron internetowych;
- Algorytmy przetwarzające łańcuchy znaków – służące do wyszukiwania, przetwarzania i porównywania fraz.

ALGORYTMY OCR ZAIMPLEMENTOWANE W PROGRAMIE

Algorytmy *OCR* służące do analizy poszczególnych znaków opierały się na uprzednim przygotowaniu i sklasyfikowaniu poszczególnych wielkich liter alfabetu angielskiego czcionki *Calibri* o wysokości między 20 a 120 pikseli. Dla każdej litery utworzono szablon. Na rysunku 5 przedstawiono przykładowy szablon dla litery „F".

Rys. 5. Szablon znaku „F" czcionki Calibri wykorzystany w algorytmie

Z szablonu wycięto poszczególne znaki w celu przygotowania masek do analizy średniego błędu kwadratowego oraz określenia podstawowych danych geometrycznych i statystycznych o danej literze w funkcji jej wymiarów.

Podstawowe własności geometryczne liter uwzględnione na etapie klasyfikacji obiektów, oprócz średniego błędu kwadratowego, to:

- stosunek długości wektora łączącego początek układu współrzędnych ze środkiem masy litery do długości wektora łączącego początek układu współrzędnych ze środkiem symetrii (A1),
- ilość pikseli znaku do pikseli tła najmniejszego prostokąta, w który wpisany jest znak (A2),
- pole powierzchni prostokąta, w który wpisany jest znak do pola powierzchni najmniejszego wielokąta opisującego literę (A3),
- stosunek długości do szerokości prostokąta, w który wpisany jest znak (A4),
- momenty *Hu* (M1-M7).

Cztery pierwsze cechy geometryczne zostały dobrane przez autora samodzielnie, jako stosunkowo łatwe do zaimplementowania i przedstawienia zależności. Niezmienniki *Hu* można pozyskać przy użyciu funkcji wbudowanej w *OpenCV*.

Stosunek długości wektora łączącego początek układu współrzędnych ze środkiem masy litery do długości wektora łączącego początek układu współrzędnych ze środkiem symetrii prostokąta opisującego literę jest pierwszą własnością geometryczną wykorzystaną w przygotowanym programie. Wartość ta zmienia się nieznacznie w funkcji rozmiarów znaku (patrz rys. 7). Jednakże jest nieodporna na rotację i zniekształcenia wywołane różnymi kątami obserwacji, stąd istotną rolą algorytmu przygotowującego obraz jest wykrycie kąta przesunięcia rysunku i obrót do odpowiedniej pozycji (ułatwiają to horyzontalne linie, patrz rys. 2).

Wartość pierwszego wskaźnika geometrycznego liczono według równania:

$$A1 = \frac{\|C\|}{\|S\|}, \tag{1}$$

gdzie:

A1 – stosunek środka masy litery do środka symetrii prostokąta, w który wpisana jest litera,
$\|C\|$ – moduł wektora środka masy litery (centroidu),
$\|S\|$ – moduł wektora środka symetrii prostokąta, w który wpisany jest znak.

Pozycję środka masy obliczono na podstawie sumy:

$$C_x = \frac{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y) * x}{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)}, \qquad C_y = \frac{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y) * y}{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)} \tag{2}$$

gdzie:

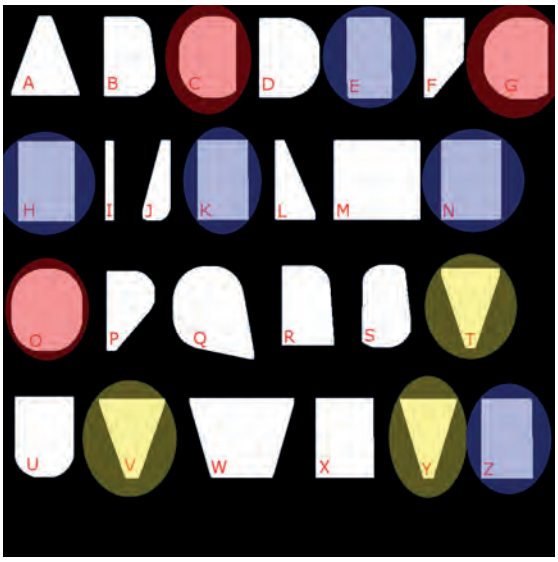
$C_{x,y}$ – składowe wektora środka masy,
f(x,y) – natężenie w danym punkcie – wartości od 0 do 255,
M, N – wymiary macierzy tworzącej ilustrację (M – kolumny, N – wiersze).

Jedną z najłatwiejszych do ekstrakcji właściwości geometrycznych i statystycznych jest **stosunek** białych pikseli (255) do czarnych (0), czyli **powierzchni zajmowanej przez literę do powierzchni tła**. Własność ta tyczy się obrazów binarnych.

$$A2 = \frac{\sum B(i,j)}{\sum C(i,j)}; \qquad (3)$$
$$B(i,j) = 255, \qquad C(i,j) = 0.$$

Jak widać na rysunku 7 właściwość ta (A2) jest stabilna przy zmianach rozmiaru litery: to znaczy, że wartość zmienia się liniowo i w niewielkim stopniu.

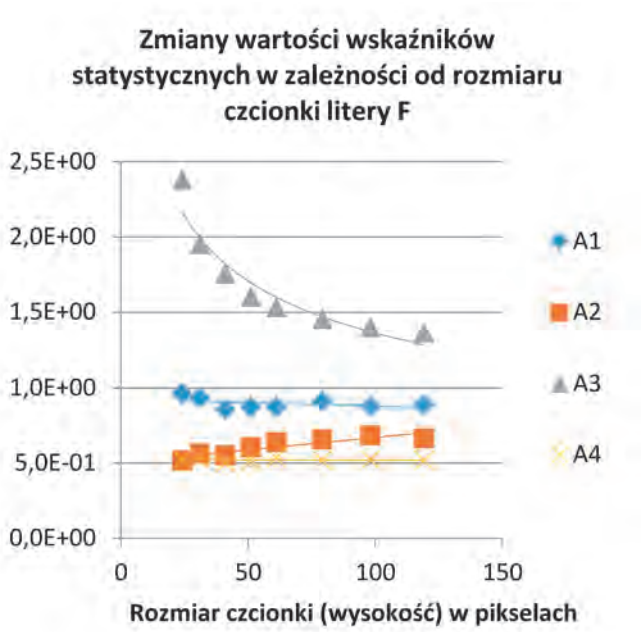
Wiele liter ma we wnętrzu przestrzeń, dla której granicą są krzywe tworzące znak. Z tego względu do ich rozpoznawania może przysłużyć się określenie **stosunku pikseli prostokąta opisującego literę do liczby pikseli najmniejszego wielokąta opisującego literę (A3)**. Poszczególne wielokąty przedstawiono na rysunku 7. Nie jest to metoda, która zwraca wypełnienie wnętrza w sensie stricte, ponieważ litera jest wcześniej zniekształcana i przybliżana do wielokąta zamkniętego. Jest to najbardziej zawodna metoda. Dwa podstawowe powody: duża wrażliwość na zakłócenia i zniekształcenia obrazu oraz na skalowanie, a także duże podobieństwo między niektórymi wielokątami zamykającymi litery, co przedstawiono na rysunku 6.



Rys. 6. Podobieństwo między wielokątami opisującymi litery alfabetu (E-H-K-N-Z, C-G-O, T-V-Y)

Ostatnim ze wskaźników geometrycznych wykorzystanych przez autora, jest **stosunek wysokości do szerokości litery (A4)**, w którą wpisany jest znak.

Rysunek 7 przedstawia pomiary zmiany wielkości wskaźników w zależności od zmian rozmiaru czcionki – jest to o tyle istotne, że niektóre z nich różnią się w funkcji wysokości znaku, co uniemożliwia zwykłe uśrednienie i ustalenie jednego przedziału wartości, które może przybierać wskaźnik. Z tego względu algorytm uwzględnia również pomiary wielkości wykrytych znaków i wartości geometryczne porównywane są do najbliższej rozmiarem, zbadanej wcześniej litery (według szablonu przedstawionego na rysunku 5).



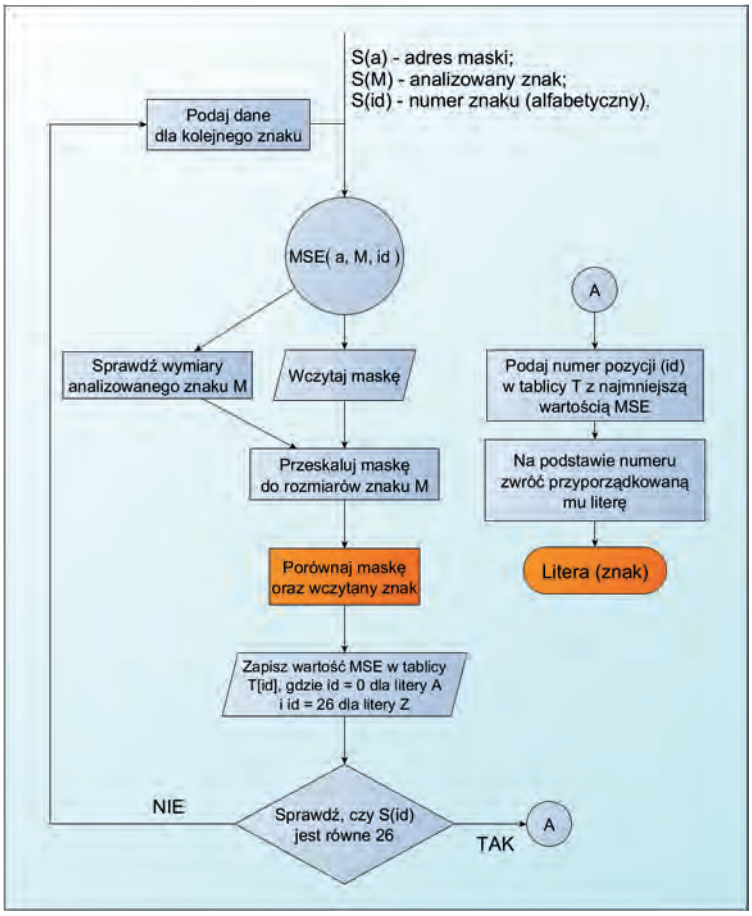
Rys. 7. Zmiany wartości parametrów statystycznych w zależności od rozmiaru czcionki dla litery F, gdzie: A1- A4 – poszczególne własności geometryczne

Kolejną grupą wskaźników użytych w programie były ***niezmienniki Hu***. W celu detekcji obiektów można wykorzystać kilka typów momentów, najważniejsze są jednak takie, które posiadają odporność na translację, rotację, skalę oraz skrzywienie obiektu. *Hu* wyznaczył sześć momentów odpornych na rotację, translację oraz skalowanie i jeden odporny na skrzywienie – a dokładnie odbicie lustrzane. Poszczególne momenty przedstawione w pracy [4]:

- $M_1 = \eta_{20} + \eta_{02}$
- $M_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2$
- $M_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2$
- $M_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2$
- $M_5 = (\eta_{30} - 3\eta_{12}) * (\eta_{30} + \eta_{12}) * [(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + (3\eta_{21} - \eta_{03}) * (\eta_{21} + \eta_{03}) * [(3\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]$
- $M_6 = (\eta_{20} - \eta_{02}) * [(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + 4\eta_{11} * (\eta_{30} + \eta_{12}) * (\eta_{21} + \eta_{03})$
- $M_7 = (3\eta_{21} - \eta_{03}) * (\eta_{30} + \eta_{12}) * [(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + (\eta_{30} - 3\eta_{12}) * (\eta_{21} + \eta_{03}) * [3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]$

gdzie η_{pg} to znormalizowany moment centralny.

Niezmienniki niższych rzędów (M1-M4) zmieniały się w sposób liniowy wraz ze zmianą wielkości litery. Momenty wyższych rzędów (M5-M7) nie nadawały się bezpośrednio do analizy ze względu na to, że ich zmienność w funkcji wielkości znaku dało przybliżyć się tylko funkcją wielomianową 5 rzędu.



Rys. 8. Schemat blokowy algorytmu porównania dwóch znaków (ilustracji) za pomocą średniego błędu kwadratowego (MSE)

Już w fazie projektowania i wyboru danych własności geometrycznych i statystycznych można było zauważyć ich niedoskonałość. Z tego względu podjęto środki zaradcze w samym algorytmie: przygotowane teksty napisane są dużą czcionką (powyżej 50 pikseli wysokości), w fazie przygotowania ilustracji jest ona obracana do odpowiedniej pozycji i odszumiana z pikseli, które mogłyby zaszkodzić odczytowi, w fazie porównywania przyrównuje się do zbioru wartości najbliższego wymiarami do odczytanej ilustracji a nie do wartości średniej ze wszystkich odczytów.

Algorytm opiera się przede wszystkim na obliczeniu *MSE* (*Mean Square Error*) i porównywaniu maski z literą, własności statystyczne i geometryczne pełnią rolę dyskryminatora trafień, które nie są szumem.

Średni błąd kwadratowy to jedno z podstawowych narzędzi w analizie sygnałów. Służy do porównywania podobieństwa dwóch sygnałów, tego w jakim stopniu jeden z nich (często przetworzony) odwzorowuje drugi (oryginalny lub zakładany) [6]. Załóżmy, że $\mathbf{x} = \{x_i | i = 1, 2, \ldots, N\}$ oraz $\mathbf{y} = \{y_i | i = 1, 2, \ldots, N\}$ to dwa skończone i dyskretne sygnały (lub w uogólnieniu na więcej wymiarów obrazy), gdzie N to liczba próbek sygnału (pikseli w przypadku ilustracji) a x_i oraz y_i to wartości *i-tych* próbek w zbiorach **x** oraz **y**. W takim razie *MSE* między sygnałami zdefiniujemy jako [6]:

$$MSE(x,y) = \frac{1}{N}\sum_{i=1}^{N}(x_i - y_i)^2. \quad (4)$$

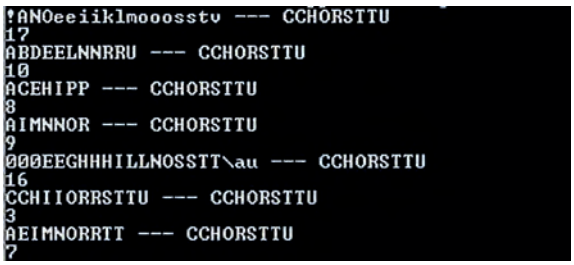
Uproszczony algorytm przedstawiający obliczenia *MSE* przedstawia rysunek 8.

MSE stał się podstawowym wyznacznikiem, a pozostałe narzędzia klasyfikacji miały na celu usunięcie fałszywych pozytywnych wyników, które mogłyby być jedynie szumem i obiektami podobnymi do liter.

POZOSTAŁE ALGORYTMY

Oprócz algorytmów *OCR* w programie użyto również skryptów algorytmów sortujących i porównujących dystans między ciągami znaków. Porównanie dystansu było dodatkowym zabezpieczeniem, ponieważ znaki wykrywane przez algorytm *OCR* nie zawsze były sortowane od lewej do prawej (tak jak czyta człowiek). Stąd wygodniejsze było przesortowa-

nie wszystkich znaków według kolejności alfabetycznej i ich porównanie do przesortowanych tytułów filmów z bloga, na którym znajdowały się recenzje. Przykład przedstawiono na rysunku 9 (dla frazy „SHORT CIRCUIT").



Rys. 9. Anagramy w kolejności alfabetycznej; po prawej – znaki wykryte, po lewej – znaki z tytułów na stronie naoczyta.wordpress.com, cyfry to odległość edycyjna między ciągami znaków

Algorytm porównujący ciągi tekstowe to Dystans *Levenshteina* (inaczej odległość edycyjna) [3].

W toku pracy zastosowano jeszcze skrypty przeszukujące strony internetowe opisane we wcześniejszej części artykułu poświęconej bibliotece *Scrapy*.

WYNIKI

Pomiary wykonano przy pełnych warunkach oświetleniowych gabinetu (świetlówki w oprawach, około 2 metry na suficie) jak i bez dodatkowego, sztucznego źródła światła (jedynie naturalne oświetlenie z okna skierowanego na północ, w pogodny dzień z niewielką liczbą chmur na niebie). Robot *NAO* wykonywał zdjęcia przy rozdzielczości 1280x960 pikseli z odległości od 15 do 90 centymetrów przy tekście o wysokości znaku 1,5 cm, co przy obrazie wykonanym z odległości 15 centymetrów było w przybliżeniu równe 140 pikselom.

Do 70 centymetrów od tekstu robot odczytuje znaki poprawnie niezależnie od tego czy sztuczne oświetlenie – świetlówki w oprawach, na wysokości ok. 2 metrów – jest włączone czy nie. Przy 80 centymetrach tytuł jest interpretowany poprawnie tylko przy włączonym świetle.

Film, na którym zarejestrowano jak robot *NAO* czyta, znajduje się pod adresem: **https://naoczyta.wordpress.com/2015/10/06/nao-likes-movies-too/**

WNIOSKI

Zwiększenie możliwości komunikacyjnych robotów humanoidalnych o rozpoznawanie poleceń i tekstów zapisanych przez ludzi jest wykonalne przy użyciu znanych algorytmów i metod detekcji obiektów – gdzie nie trzeba uciekać się do zagadnień związanych z głębokim uczeniem się maszynowym. Wymaga to jednak nałożenia kilku ograniczeń: do rozmiaru i kroju czcionki, sposobu przedstawiania fraz, jak i wykorzystania dodatkowej jednostki obliczeniowej w celu realizacji projektu.

Przedstawiony algorytm stanowi prototyp na bazie którego można tworzyć usprawnione programy. Elementy, które przede wszystkim wymagają poprawy to możliwość wykonywania programu lokalnie, za pomocą sprzętu robota, a nie dodatkowego komputera oraz większa liczba czcionek, które mogą zostać obsłużone przez robota.

Umiejętność czytania przez humanoidalne roboty otwiera wiele możliwości praktycznego zastosowania: robot może pomagać osobom niedowidzącym, pracować, uczyć się, odczytywać teksty przy obsłudze klientów. Łącząc czytanie z przeszukiwaniem internetowych baz danych można poszerzyć zakres działań robota lub ilość danych, które może pozyskać w trakcie czytania.

LITERATURA

[1] BHARGAVA A., NATH K. V., SACHDEVA P., SAMEL M.: Reading Assistant for the Visually Impaired. International Journal of Current Engineering and Technology, Vol. 5, No. 2, ss. 1050-1055.

[2] DUE TRIER Ø., JAIN A., TAXT T.: Feature extraction methods for character recognition – a survey. Pattern Recognition, vol. 29, No. 4, ss. 641-662.

[3] GILLELAM M.: Levenshtein Distance, in Three Flavors. http://people.cs.pitt.edu/~kirk/cs1501/Pruhs/Spring2006/assignments/editdistance/Levenshtein%20Distance.htm

[4] HU MING KUEI: Visual Pattern Recognition by Moment Invariants. IEEE Transactions on Information Theory - TIT, vol. 8, no. 2, ss. 179-187.

[5] MORI S., SUEN C., YANAMOTO K.: Historical Review of OCR research and development. Proceedings of IEEE, vol. 80, ss. 1029-1058.

[6] WANG Z., BOVIK A.C.: Mean Squared Error: Love It or Leave It? IEEE Signal Processing Magazine, Vol. 26, ss. 98-117.

[7] ZHAO S., NAKAMURA K., ISHII K., IGARASHI T.: Magic Cards: A Paper Tag Interface for Implicit Robot Control. Proceedings of the ACM Conference on Human Factors in Computing Systems, CHI2009, ss. 173-182.

[8] http://www.solidworks.pl/aktualno-ci/robot-nao-nowy-przyjaciel-osob-starszych-i-chory-a4114910.

[9] http://www.kuka-robotics.com/en/solutions/branches/food/

[10] https://en.wikipedia.org/wiki/Visual_perception

[11] https://en.wikipedia.org/wiki/Web_scraping