Pomiary Automatyka Robotyka, ISSN 1427-9126, R. 23, Nr 4/2019, 5–18, DOI: 10.14313/PAR_234/5

Agentowa struktura wielomodalnego interfejsu do Narodowej Platformy Cyberbezpieczeństwa, część 2.

Włodzimierz Kasprzak, Wojciech Szynkiewicz, Maciej Stefańczyk, Wojciech Dudek, Maksym Figat, Maciej Węgierek, Dawid Seredyński, Cezary Zieliński
Politechnika Warszawska, Wydział Elektroniki i Technik Informacyjnych, Instytut Automatyki i Informatyki Stosowanej, Nowowiejska 15/19, 00-665 Warszawa

Streszczenie: Ten dwuczęściowy artykuł przedstawia interfejs do Narodowej Platformy Cyberbezpieczeństwa (NPC). Wykorzystuje on gesty i komendy wydawane głosem do sterowania pracą platformy. Ta część artykułu przedstawia strukturę interfejsu oraz sposób jego działania, ponadto prezentuje zagadnienia związane z jego implementacją. Do specyfikacji interfejsu wykorzystano podejście oparte na agentach upostaciowionych, wykazując że podejście to może być stosowane do tworzenia nie tylko systemów robotycznych, do czego było wykorzystywane wielokrotnie uprzednio. Aby dostosować to podejście do agentów, które działają na pograniczu środowiska fizycznego i cyberprzestrzeni, należało ekran monitora potraktować jako część środowiska, natomiast okienka i kursory potraktować jako elementy agentów. W konsekwencji uzyskano bardzo przejrzystą strukturę projektowanego systemu. Część druga tego artykułu przedstawia algorytmy wykorzystane do rozpoznawania mowy i mówców oraz gestów, a także rezultaty testów tych algorytmów.

Słowa kluczowe: Narodowa Platforma Cyberbezpieczeństwa, rozpoznawanie obrazu, rozpoznawanie gestów, rozpoznawanie mowy, rozpoznawanie mówcy

1. Wprowadzenie

1.1. Wielomodalny interfejs człowiek-komputer

Prace badawcze poświęcone wielomodalnym interfejsom człowiek-komputer są prowadzone od ponad 40 lat [37]. Celem tych badań jest opracowanie metod i technik interakcji ludzi z komputerem w pełni wykorzystujących sposoby naturalnej komunikacji i interakcji człowieka z otoczeniem. Interfejsy wielomodalne charakteryzują się dwiema podstawowymi cechami: łączą wiele typów danych oraz przetwarzają te dane w czasie rzeczywistym przy określonych ograniczeniach czasowych [10].

System "Put-That-There" [3] opracowany w MIT (USA) jest powszechnie uważany za pierwszy praktyczny pokaz możliwości, jakie daje wielomodalny interfejs. W systemie tym były łączone dwa rodzaje wejść: głosowe oraz gesty, które umożliwiały użytkownikowi siedzącemu na krześle naturalną i efektywną interakcję z systemem wizualizacji danych przestrzennych. System akceptował polecenia typu „utwórz tutaj zielony kwadrat", „zmniejsz to" lub „umieść to tam". Żadne z tych poleceń nie może być poprawnie zinterpretowane, gdy głos i gesty będą rozważane niezależnie, dopiero ich poprawnie zinterpretowana wielomodalna kombinacja tworzy proste i ekspresywne polecenie, które jest naturalne dla użytkownika.

W latach 80. i 90. ubiegłego wieku powstało wiele prototypowych systemów z wielomodalnymi interfejsami wykorzystujących zarówno wejścia audio (rozpoznawanie mowy), jak i wizyjne (rozpoznawanie gestów rąk, głowy, ciała, ruchu warg, wyrazu twarzy) [10, 17, 36, 49].

Obecnie rozwijane są nowe rodzaje wielomodalnych interfejsów człowiek-komputer, które są określane jako „Cognitive Immersive Rooms" [9, 57]. Są to pomieszczenia, które „słyszą", „widzą", interpretują polecenia i ruchy użytkownika lub grupy użytkowników i odpowiednio na nie reagują.

Wielomodalność interfejsu zapewnia dogodniejszy dla człowieka sposób komunikacji z maszyną i powinna też zwiększać skuteczność i poprawność takiej komunikacji. Posługiwanie się takimi formami przekazu jak gesty i mowa jest naturalnym dla człowieka sposobem przekazywania swoich zamiarów i poleceń. Druga zaleta wielomodalności wynika z faktu, że żadna automatyczna analiza danych pochodzących z czujników nie zapewnia poprawności rozpoznania przekazu człowieka w 100%. Poprawnie wykonana kombinacja danych pocho-

Autor korespondujący:
Cezary Zieliński, c.zielinski@ia.pw.edu.pl

Artykuł recenzowany
nadesłany 15.07.2019 r., przyjęty do druku 16.09.2019 r.

dzących z różnych modalności interfejsu powinna zwiększyć poprawność rozpoznania poleceń w porównaniu do każdej pojedynczej modalności.

W proponowanym rozwiązaniu realizujemy trzy modalności: rozpoznawanie gestów, rozpoznawanie mowy i rozpoznawanie mówcy.

1.2. Rozpoznawanie gestów

Na potrzeby interakcji z komputerem człowiek może wykonywać gesty różnymi częściami ciała – dając znaki ręką [22], ruchem głowy lub całym ciałem [35] lub mimiką twarzy [56].

Zasadniczo przez gest rozumie się akcję wykonywaną przez człowieka charakteryzującą się ruchem. W niniejszej pracy nazwiemy to **gestem dynamicznym** [39]. Do rozpoznawania gestu potrzebna jest analiza sekwencji czasowej obrazów. W odróżnieniu, przez **gest statyczny** rozumieć będziemy określony układ elementu ciała człowieka, pozostający w niezmienionym stanie podczas procesu akwizycji obrazu. Zasadniczo do rozpoznania takiej pozy wystarczy pojedynczy obraz.

Procedura rozpoznawania gestu obejmuje akwizycję danych (obraz lub sekwencja obrazów), detekcję istotnych cech w obrazie/obrazach i zastosowanie technik uczenia maszynowego do wytrenowania klasyfikatora do rozpoznawania zadanych gestów [53]. Klasyczne rozwiązania w zakresie detekcji cech w pojedynczych obrazach polegają na wykrywaniu lokalnych charakterystyk takich, jak: cechy Haara, cechy HoG i deskryptory punktów kluczowych [27]. Typowo stosowane klasyfikatory cech numerycznych to LDA (liniowa analiza dyskryminacyjna), klasyfikator kaskadowy, SVM (maszyna wektorów nośnych) i MLP (wielowarstwowy perceptron) [20]. W analizie sekwencji obrazów stosuje się dodatkowo informację o ruchu (wyznaczając tzw. optyczny potok lub wektory ruchu dyskretnych elementów obrazu), a typowym klasyfikatorem sekwencji obserwacji w czasie są modele HMM (Ukryte Modele Markowa) [19].

W ostatnich latach na popularności zyskały rozwiązania oparte na technice głębokich sieci neuronowych. Sieć splotowa CNN wyznacza nieliniowe przekształcenie zadanego obszaru obrazu do wektora cech a warstwy wyjściowe realizują klasyfikację [15, 25]. Na potrzeby rozpoznawania gestów dynamicznych, typowy dotąd model HMM może być zastąpiony siecią rekurencyjną LSTM (ang. *long short-term memory*), specjalnie zaprojektowaną do rozpoznawania sekwencji obserwacji w czasie [50]. Przegląd metod rozpoznawania gestów w obrazach RGB można znaleźć w szeregu pracach, np. [4, 15, 39], natomiast przegląd metod rozpoznawania gestów w obrazach RGB-D zawarty jest m.in. w pracach [6, 40].

Praktycznie wszystkie z przedstawionych metod zostały już w mniejszym lub większym stopniu zaimplementowane w różnych bibliotekach bądź pakietach programistycznych. Zależnie od tego, który fragment systemu ma zostać zaimplementowany, wykorzystać można różne ich zestawienia. Przy pomocy biblioteki OpenPose [5] możliwe jest wykrycie sylwetki człowieka w obrazie oraz dopasowanie do niej charakterystycznych punktów węzłowych (stawów), takich jak nadgarstki, łokcie, ramiona itp., uzyskując w ten sposób szkieletową reprezentację operatora. Mając dane o położeniu poszczególnych punktów w czasie (ich trajektorie), można wykorzystać metody dostępne w bibliotece Gesture Recognition Toolkit [11] do klasyfikacji sekwencji gestów. Biblioteka ta zawiera implementacje metod klasyfikacji gestów opisanych wcześniej, wymaga jednak podania wyznaczonych zestawów cech. Oprócz położeń punktów charakterystycznych sylwetki można wykorzystać dokładny model samej dłoni, o ile dostępny jest dla niej obraz z głębią w wysokiej rozdzielczości [48].

XKin [38] używa ukrytych modeli Markowa (HMM) oraz obrazu głębi w celu wyznaczenia pozycji operatora oraz klasyfikacji 16 gestów dynamicznych, natomiast gesty statyczne (znaki amerykańskiego języka migowego) rozpoznawane są przy użyciu cech tekstury kolorowej. Do rozpoznawania znaków języka migowego można wykorzystać również DeepHand [24] – rozwiązanie oparte na sieciach splotowych, wytrenowanych w oparciu o około milion obrazów kolorowych z różnymi gestami.

Prototyp interaktywnego narzędzia InterSec [34] do trójwymiarowej wizualizacji zawiera „naturalny" interfejs użytkownika, umożliwiający sterowanie wizualizacją za pomocą gestów. Danymi wejściowymi dla systemu rozpoznawania gestów są odczyty z czujników Kinect i LEAP Motion oraz ekranu wielodotykowego. Dzięki temu możliwe jest jednoczesne wykonanie kilku gestów za pomocą dłoni i ciała w celu realizacji różnych zadań.

1.3. Rozpoznawanie mowy

Automatyczne rozpoznawanie mowy (ang. *Automatic Speech Recognition*) jest przedmiotem badan od około 60 lat. Na obecny stan rozwoju metod rozpoznawania mowy zasadniczy wpływ miało wprowadzenie modeli stochastycznych HMM (ang. *Hidden Markov Models*) pod koniec lat 80. XX wieku [2, 42] i zastosowanie głębokich sieci neuronowych DNN, począwszy od około 2005 r. [41]. Początkowo sieci te stosowano do klasyfikacji pojedynczych ramek sygnału w terminach podfonemów, pozostawiając modelom HMM zadanie rozpoznawania sekwencji obserwacji, co przyjmowało postać hybrydowego rozwiązania DNN-HMM. Jednak wraz z rozwojem specjalizowanych głębokich sieci rekurencyjnych zapewniających modelowanie opóźnień czasowych, w ostatnich kilku latach sieci neuronowe stopniowo przejmują także rolę pełnioną dotąd przez HMM. Skutkuje to tzw. systemami „end-to-end" bezpośrednio odwzorowującymi akustyczną sekwencję wejściową w symboliczną sekwencję wyjściową [13, 7].

Można wymienić kilka implementacji systemu rozpoznawania mowy publicznie dostępnych o otwartym kodzie źródłowym [31]. W początkach XXI wieku powstał modułowy system Sphinx-4 (Sun Microsystems, 2004) [52]. Wyznacza on pewien standard implementacji podstawowych algorytmów do analizy mowy i jednocześnie zapewnia strukturę szkieletową („framework") do tworzenia systemów analizy mowy w języku Java. Dostępne też było popularne narzędzie HTK do symbolicznej analizy sygnału mowy z wykorzystaniem modelu HMM [54]. Nowszym projektem z tej dziedziny jest KALDI [18], biblioteka o charakterze badawczym w języku C++ obejmująca nowo proponowane rozwiązania.

Były lub są dostępne „mniejsze" rozwiązania obejmujące wybrane zakresy funkcji analizy mowy. MARF (ang. *Modular Audio Recognition Framework*) [30] to biblioteka w języku Java, dość dobrze udokumentowana. Ostatnia stabilna wersja pochodzi z 2007 r. – nie jest już rozwijana. Do wyznaczania cech LPC lub MFCC dla ramek sygnału mowy istnieje szereg bibliotek. Biblioteka Loudia7 [26] ma otwarty i dobrze udokumentowany kod (licencja GPLv3). Biblioteka FFTW8 [8] (licencja GPL) przeznaczona jest do obliczeń szybkiej transformaty Fouriera. Dostępna jest implementacja funkcji marszczenia czasu FastDTW [46] (w języku Java), służąca do dopasowania dwóch sekwencji wektorów cech o różnej długości. Autorzy niniejszej pracy zrealizowali też w przeszłości własne implementacje wybranych funkcji w języku C++ lub Java: VAD, wyznaczanie cech MFCC, klasyfikator DTW, klasteryzacja cech, rozpoznawania z użyciem modeli HMM [20, 21]. W zakresie analizy akustycznej warto też sięgnąć do rozwiązań sprawdzonych w obu zagadnieniach analizy mowy – rozpoznawania komend, jak i mówców. Takim rozwiązaniem jest biblioteka SPro (Speech Signal Processing Toolkit) [14] z licencją „MIT License".

1.4. Rozpoznawanie mówcy

Przez rozpoznawanie mówcy rozumie się zazwyczaj dwa sposoby analizy sygnału mowy [29]:

– Identyfikacja mówcy – użytkownik nie musi udowadniać tożsamości; system decyduje, który model mówcy jest najbardziej zbliżony do mowy wejściowej → wymaga N porównań aktualnej obserwacji z modelem;
– Weryfikacja mówcy – użytkownik musi najpierw podać swoją tożsamość, a następnie system sprawdza, czy jest ona prawidłowo rozpoznana → wymagane jest jedno porównanie obserwacji z modelem.

Systemy rozpoznawania mówcy obejmują następujące rozwiązania:
– Podstawowy sposób nazywany jest UBM-GMM [44, 45]. Polega on na utworzeniu stochastycznych modeli dla każdego rejestrowanego mówcy i dla mówcy „średniego", mających postać mieszanin rozkładów Gaussa. Rozpoznawanie polega na określeniu odstępu miar wiarygodności dopasowania obserwacji do modelu mówcy i modelu średniego.
– Rozwiązanie nazywane GMM-SVM polega na utworzeniu modelu mówcy w postaci superwektorów cech przez złożenie reprezentantów wszystkich klastrów Gaussa danego mówcy. Superwektory są następnie rzutowane na podprzestrzeń w celu redukcji wymiaru, a do rozpoznawania stosowane są klasyfikatory SVM.
– Rozwiązanie o nazwie „Joint Factor Analysis" polega na zastosowaniu analizy czynnikowej w przestrzeni superwektorów cech [29]. Wyznaczane są jednocześnie czynniki i podprzestrzenie zależne od mówców oraz czynniki zakłóceń – zależne od kanału wejściowego i szumu środowiska. W praktyce proces uczenia przebiega sekwencyjnie – najpierw wyznacza się dominujące kierunki zmienności odpowiadające czynnikom zależnym od mówców, następnie po „zamrożeniu" tych czynników znajduje się czynniki modelujące zmienność kanału nagrywającego, a na koniec znajduje czynniki modelujące szum środowiska.
– Podejścia oparte na wyznaczaniu tzw. I-wektorów polegają na zastosowaniu liniowej analizy dyskryminacyjnej (LDA) lub tzw. „probabilistycznego LDA" (PLDA) jako klasyfikatorów [33]. I-wektory są wyznaczane w wyniku analizy czynnikowej wykonywanej w przestrzeni superwektorów. W pierwszym rozwiązaniu następuje potem redukcja wymiaru nowej przestrzeni i klasyfikacja za pomocą LDA. W drugiej metodzie prowadzona jest najpierw kolejna dekompozycja na czynniki, tym razem przestrzeni i-wektorów, modelująca zakłócenia i klasyfikacja stosująca zaawansowane stochastyczne miary odległości.
– Głębokie sieci neuronowe znalazły również zastosowanie do rozpoznawania mówców [8, 41]. Prace badawcze dotyczą wykorzystania tych sieci do modelowania mówcy – znajdywania w procesie uczenia nieliniowego przekształcenia cech zastępującego model mieszanin Gaussa – a także na etapie dopasowania obserwacji z modelem [28].

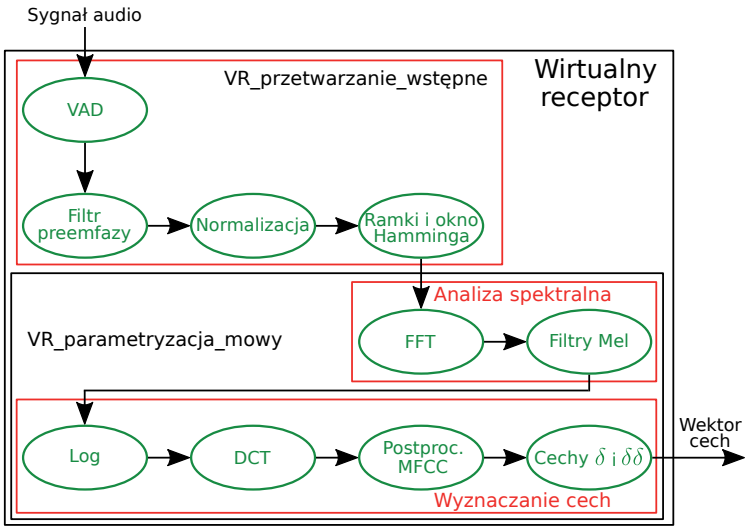
W zakresie technik modelowania i rozpoznawania mówcy dostępna jest biblioteka „open source" projektu ALIZE [1] (Uniwersytet w Avignon) na licencji GNU Lesser General Public License (LGPL). Korzysta ona z wyżej wymienionej biblioteki SPro [14] do analizy akustycznej sygnału mowy i implementuje podstawowe rozwiązanie UBM-GMM. Jej rozszerzeniem jest platforma biometryczna Mistral [32], która oprócz ALIZE zawiera też szereg nowszych technik stosowanych w rozpoznawaniu mówców (np. metodę GMMSVM i analizę czynnikowa przestrzeni superwektorów).

2. Algorytmy wykorzystywane przez moduł audio

Zasadniczo jedynym agentem zajmującym się przetwarzaniem sygnałów mowy jest a_{audio}, w którego skład wchodzą: podsystem sterowania c_{audio}, wirtualny receptor $r_{audio,mic}$, wirtualny efektor $e_{audio,ui}$, receptor rzeczywisty $R_{audio,mic}$ oraz efektor rzeczywisty $E_{audio,ui}$. Wirtualny receptor jest odpowiedzialny za przetworzenie sygnału głosowego w celu rozpoznania zarówno mówcy, jak i wydanej komendy. Sposób przetwarzania sygnału głosowego przez receptor wirtualny $r_{audio,mic}$ opisano w podsekcjach 2.1, 2.2, 2.3 i 2.4.

2.1. Przetwarzanie wstępne

Pierwsze kroki przetwarzania komendy głosowej wykonywane są w dziedzinie czasu i obejmują (rys. 1): detekcję głosu VAD (ang. *Voice Activity Detection*), a konkretnie odróżnienie chwil, w których użytkownik mówi od tych, kiedy wykrywany jest jedynie szum, filtrację prowadzącą do wzmocnienia wysokich częstotliwości sygnału (filtr preemfazy) w celu redukcji stosunku szumu do sygnału użytecznego, normalizację jego amplitudy, segmentację sygnału na ramki, a następnie pomnożenie zawartości tych ramek przez okna Hamminga w celu zredukowania wpływu skończonego czasu trwania sygnału w ramce na jego spektrogram [43].



Rys. 1. Funkcje wirtualnego receptora $r_{audio,mic}$ agenta audio a_{audio}: przetwarzanie wstępne i parametryzacja pojedynczych (krótkookresowych) ramek sygnału mowy
Fig. 1. Audio agent virtual receptor functions: preprocessing and parametrization of single (short-time) speech frames

2.2. Parametryzacja mowy

Kolejny etap przetwarzania odbywa się w dziedzinie częstotliwości. Rozpoczyna się od wykonania szybkiej transformaty Fouriera (FFT) sygnału w ramce, a następnie stosuje się szereg filtrów pasmowych. Pierwsza czynność określa spektrum sygnału w każdej ramce, na podstawie którego wyznaczane są współczynniki Fouriera, a druga wyznacza cechy melspektralne dla ramki sygnału mowy MFC (ang. *Mel-Frequency Coefficients*) i polega na przekształceniu amplitudy współczynników Fouriera za pomocą trójkątnych filtrów pasmowych rozmieszczonych równomiernie na skali mel, z końcowym zlogarytmowaniem wyników takich filtracji [47]. Skala mel określa zależność subiektywnie odbieranej wysokości dźwięku od rzeczywistej częstotliwości tego dźwięku.

Wykonanie przekształcenia kosinusowego powoduje transformację współczynników do dziedziny cepstralnej. Cepstrum to odwrotna transformata Fouriera widma sygnału wyrażonego w skali logarytmicznej. Przekształcenia w tej dziedzinie polegają na odjęciu wektora wartości średnich i na filtracji w dziedzinie cepstralnej (ang. *liftering*), w ten sposób tworząc znormalizowane współczynniki melcepstralne MFCC (ang. *Mel-Frequency Cepstral Coefficients*). Wektor cech uzupełniony zostaje o gradienty tych współczynników względem czasu i wartość energii sygnału w ramce [47].

2.3. Modelowanie i rozpoznawanie komend

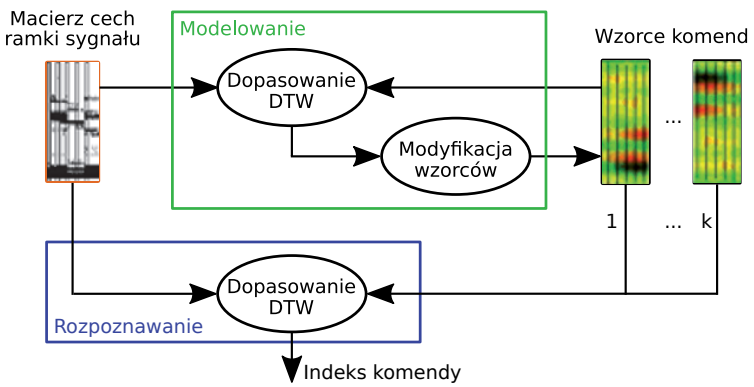
Moduł audio umożliwia tworzenie modeli akustycznych dwóch typów:
– modeli poszczególnych komend głosowych;
– modeli głosu poszczególnych mówców.

Współwystępowanie obu typów modeli w jednym systemie rozpoznawania komend i identyfikacji mówców pozwala na specyficzne rozwiązanie problemu modelowania i rozpoznawania komend głosowych. Istnieje bowiem możliwość korzystania z tej samej bazy próbek mowy dla utworzenia modeli obu typów. Dodatkową okolicznością jest ograniczona liczba komend głosowych, które mogą być reprezentowane w pojedynczej instalacji interfejsu. W takiej sytuacji przyjęto założenie upraszczające polegające na tym, że słownik komend ma charakter „zamknięty" – rozpoznawane są jedynie te komendy, dla których istnieją próbki głosowe, pobrane od zarejestrowanych mówców. Oczywistym faktem jest, że dla zbudowania modeli mówców potrzebne są ich rzeczywiste próbki głosowe. Przyjęto, że będą nimi wypowiedzi zawierające reprezentowane w systemie komendy głosowe. To założenie motywowane jest obserwacją, że identyfikacja mówcy jest skuteczniejsza, gdy rozpoznawanie jego głosu odbywa się w oparciu o te same komendy, co uzyskane uprzednio w procesie tworzenia modeli mówców. Powyższe założenie odnośnie modeli mówców pozwala z kolei na przyjęcie założenia o zamkniętym słowniku dla procesu uczenia modeli komend głosowych.

Model każdej komendy ma postać macierzy cech (sekwencji wektorów cech dla kolejnych ramek sygnału w czasie) – każda kolumna takiej macierzy jest jednym wektorem cech. Liczba wektorów cech jest zmienna i zależy od długości wypowiedzianej komendy. Model jednej komendy powstaje w wyniku dopasowania do siebie i uśrednienia indywidualnych macierzy cech tworzonych dla próbek uczących zawierających daną komendę (rys. 2). Rozpoznawanie komendy polega na porównaniu macierzy cech pozyskanej z aktualnie analizowanego fragmentu sygnału z modelami komend. Kluczowym algorytmem w procesie dopasowania w obu fazach pracy – modelowania i rozpoznawania – jest własna wersja algorytmu „dynamicznego marszczenia czasu" (ang. *Dynamic Time Warping*) [55].

2.4. Modelowanie i rozpoznawanie mówców

Funkcje modelowania i rozpoznawania mówcy są odmianą typowego rozwiązania zwanego UBM-GMM (ang. *Universal Background Model – Gaussian Mixture Model*). Wektory cech X^t wyznaczone dla kolejnych ramek t sygnału mowy tworzą punkty w wielowymiarowej przestrzeni. W wyniku klasteryzacji (grupowania) wektorów cech uzyskanych dla wszystkich mówców powstaje uniwersalny model UBM Λ^{ubm} o postaci N mieszanin funkcji Gaussa. Zasadniczo podobnie tworzone są



Rys. 2. Funkcje wirtualnego receptora agenta audio: modelowanie i rozpoznawanie komend głosowych
Fig. 2. Audio agent virtual receptor functions: modelling and recognition of spoken commands

modele dla indywidualnych mówców, gdy rozpatruje się jedynie próbki wypowiedzi każdego mówcy z osobna. Na koniec modelowania dodatkowo przesuwa się klastry modeli mówców w kierunku odpowiadających im klastrom modelu UBM w sytuacji, gdy zbiór próbek dla mówcy jest mało liczny. Niech $X^t(s)$ oznacza wektor cech dla ramki t sygnału mowy danego mówcy s. Niech m_h^{ubm} oznacza środek klastra cech o indeksie h w modelu UBM, a $m_h(s) = \varepsilon_h\{X^t(s)\}$ jest środkiem najbliższym mu klastra cech dla nagrań mówcy s. Dla wyznaczenia nowego środka klastra w modelu mówcy $m_h(s)$ stosujemy współczynnik α_h, który jest bliski 1 wtedy, gdy zawiera dużo danych, albo zdąża do zera wtedy, gdy liczba danych dla mówcy jest mała:

$$m_h(s) = \alpha_h \ \varepsilon_h\{X(s)\} + (1-\alpha_h)\, m_h^{\text{ubm}}$$

Powyższa modyfikacja dotyczy jedynie środków klastrów. Macierze kowariancji dla klastrów wyznaczonych dla próbek danego mówcy pozostają bez zmian. W powyższym wzorze macierze kowariancji nie występują w sposób bezpośredni, gdyż są niezmienne, a wzór ten podaje jedynie modyfikację wektora średnich. Ocena dopasowania modelu danego mówcy ze zbiorem cech dla ramek aktualnego sygnału ma charakter względny, gdyż porównuje się ze sobą dwa wyniki dopasowania aktualnej próbki mowy – raz, z modelem mówcy, a dwa – z modelem uniwersalnym UBM. Dopiero różnicę tych ocen, wyrażonych w skali logarytmicznej, porównuje się z zadanym progiem. Niech X^t będzie zbiorem wektorów cech pozyskanych z aktualnej wypowiedzi, s – identyfikatorem mówcy, Λ^s – modelem mówcy s, Λ^{ubm} – uniwersalnym modelem wszystkich mówców. Identyfikacja opiera się na mierze dopasowania obserwacji z modelem mówcy (ang. *score*), będącej różnicą dwóch wartości (logarytmów) wiarygodności:

$$S(X^t \mid \Lambda^s, \Lambda^{\text{ubm}}) = \log p(X^t \mid \Lambda^s) - \log p(X^t \mid \Lambda^{\text{ubm}})$$

3. Algorytmy wykorzystywane przez moduł wizyjny

Za przetwarzanie obrazów odpowiedzialny jest agent a_{vision}. Wirtualny receptor r_{vision} agreguje obrazy uzyskane ze stereopary kamer oraz przetwarza je w opisany w podsekcjach 3.1, 3.2, 3.3 i 3.4 sposób. Następnie wirtualny receptor r_{vision} przekazuje podsystemowi sterowania c_{vision} pozycje dłoni, aby ten po przekształceniu opisanym w podsekcjach 3.5 i 3.6 Przesłał wyniki analizy modułowi prezentacji, by ten w końcu odwzorował je na pozycje kursorów na ekranie. W tym celu rozpoznawane są zarówno gesty statyczne jak i dynamiczne.

3.1. Rektyfikacja obrazu

Wirtualny receptor r_{vision} wykorzystując parametry kalibracyjne układu kamer (zarówno dotyczące pojedynczych kamer jak i relacji między nimi) poddaje obrazy przekształceniu prostującemu. Parametry pozyskiwane są z podsystemu sterowania, który z kolei pobiera je z agenta $a_{\text{vision-database}}$. W wyniku z obrazu usuwane są zniekształcenia wprowadzane przez układ optyczny, a pozycja obu obrazów normalizowana jest w taki sposób, że poszczególne linie poziome na obu obrazach odpowiadają sobie. Tak przygotowana para obrazów jest wykorzystana do wyznaczenia mapy głębi.

3.2. Wyznaczanie mapy głębi

Oba obrazy uzyskane ze stereopary poddawane są operacji wyznaczenia mapy niezgodności (ang. *disparity map*), na której określane jest względne przesunięcie między widokiem lewym a prawym tego samego obiektu w przestrzeni. Wykorzystywany jest w tym celu algorytm SGBM (ang. *Semi-Global Block*

Matching) [16]. Uzyskane różnice położeń (wyrażone w pikselach obrazu) mogą być następnie bezpośrednio przeliczone na odległość danego punktu od kamery (wyrażoną w metrach), a z niej wyznaczyć można już pozostałe dwie współrzędne kartezjańskie.

3.3. Detekcja obszarów dłoni i twarzy operatora

Obliczanie mapy niezgodności jest w ogólności zadaniem dość czasochłonnym, a jego złożoność zależna jest od rozmiaru obrazu wejściowego. W celu minimalizacji czasu przetwarzania do wyznaczania głębi przekazywane są jedynie obszary zainteresowania. Dla interfejsu NPC są nimi dłonie oraz twarz operatora. Ich wykrycie w obrazie RGB jest realizowane przez operacje segmentacji danych.

Pierwszym krokiem jest wykrycie w obrazie twarzy operatora. Stosowany do tego jest klasyczny detektor kaskadowy wykorzystujący cechy Haara [51]. Jeśli twarz była poprawnie wykryta w poprzednich obrazach, w kolejnych jej pozycja jest jedynie śledzona. W wykrytym obszarze wykonywane jest dopasowanie cech twarzy: oczu, nosa, ust oraz linii podbródka [23]. Na ich podstawie wyznaczany jest jej owal, a z niego określana jest barwa skóry operatora.

W kolejnym kroku, znając rozkład barwy skóry operatora, dokonywana jest segmentacja obrazu w celu znalezienia dłoni. Podobnie jak w przypadku twarzy, po ich wstępnym wykryciu w następnych obrazach są one jedynie śledzone w celu przyspieszenia obliczeń.

3.4. Lokalizacja dłoni w układzie odniesienia

Mając wyznaczone obszary zainteresowania, tj. twarz i obie dłonie, są one przekazywane do algorytmów wyznaczania głębi, które w tym przypadku działają znacząco szybciej. Ostatecznie wirtualny receptor r_{vision} przekazuje do podsystemu sterowania c_{vision} pozycje punktów charakterystycznych dłoni wyznaczonych względem układu odniesienia związanego ze stereoparą.

3.5. Gesty statyczne

Gesty statyczne to takie, których znaczenie związane jest z pozycją nieruchomych dłoni operatora. Są one przekształcane przez podsystem sterowania c_{vision} na kody zwyczajowo generowane przez urządzenia wejściowe, takie jak np. mysz komputowa. Przykładowo, wykrywane są następujące gesty: dłoń otwarta to gest neutralny, zaciśnięta pięść oznacza kliknięcie, natomiast palec wyciągnięty do góry lub w dół powoduje przewijanie ekranu. Wynik rozpoznawania gestu statycznego jest przesyłany do modułu prezentacji wraz z pozycją dłoni.

Do rozpoznawania gestów statycznych pierwotnie wykorzystany został detektor kaskadowy używający cech Haara. Został on wytrenowany na sześciu gestach (dłoń otwarta, pięść, wskazywanie, kciuk w górę, kciuk w dół oraz palce ułożone w literę „V"). Do uczenia wykorzystano zbiór zawierający w sumie 3331 obrazów. Wyniki tego klasyfikatora nie były jednak na odpowiednio wysokim poziomie, dlatego przygotowany został klasyfikator wykorzystujący splotowe sieci neuronowe (CNN). Architektura sieci składa się z siedmiu warstw splotowych, zgrupowanych w trzy bloki, między którymi wykonywana jest operacja redukcji za pomocą funkcji maksimum (ang. *max pooling*). Sieć redukuje kolejno obraz wejściowy o wielkości 32 px × 32 px aż do uzyskania sześciu macierzy wyjściowych w ostatniej warstwie splotowej. Na wyjściu znajduje się sześć neuronów odpowiadających sześciu nauczonym gestom, z których wartość każdego z nich wyznaczana jest przez operację uśrednienia odpowiedniej płaszczyzny z ostatniej warstwy splotowej. Ostatnim elementem sieci jest *softmax*, normalizujący wyjścia do prawidłowego rozkładu prawdopodobieństwa (rys. 3). Klasyfikator ten daje zdecydowanie lepsze wyniki, więc zdecydowano się ostatecznie na jego wykorzystanie.

3.6. Gesty dynamiczne

Gesty dynamiczne to takie, dla których istotna jest trajektoria ruchu dłoni, a więc takie jak machnięcia w lewo bądź prawo. W tym celu podsystem sterowania c_{vision} wykorzystuje algorytm dynamicznego marszczenia czasu [12]. Uczenie gestów wymaga zebrania odpowiedniej liczby danych wzorcowych. Ze względu na znacznie mniejszą wymiarowość problemu dopasowania trajektorii (w porównaniu do zadania rozpoznawania obrazów), wystarcza niewielka liczba (poniżej 10) pomiarów referencyjnych. Rozpoznane gesty dynamiczne przekazywane są do modułu prezentacji, jako identyfikator gestu. Identyfikatory następnie tłumaczone są na skróty klawiaturowe przez moduł prezentacji.

Gesty dynamiczne interpretowane są tylko wtedy, kiedy użytkownik używa lewej ręki, tj. prawa dłoń znajduje się poza obszarem roboczym. Od tego momentu kolejne pozycje lewej dłoni są zapisywane w pamięci wewnętrznej do bufora o pojemności 50 pozycji. Po każdej zapisanej pozycji cały bufor jest interpretowany jako trajektoria i dopasowywany do niego jest jeden z nauczonych gestów dynamicznych. Po rozpoznaniu gestu o odpowiednio wysokim współczynniki podobieństwa jego identyfikator przesyłany jest do agenta odpowiedzialnego za sterowanie wizualizacją, czyli do a_{prez}. W celu uniknięcia wielokrotnego wywołania tego samego gestu w krótkim okresie (w kilku następujących po sobie obrazach) po poprawnym rozpoznaniu gestu dynamicznego przez kolejną sekundę rozpoznawanie jest zawieszone – następuje akwizycja pozycji dłoni, ale gesty nie są interpretowane.

4. Testy rozpoznawania mowy i mówców

4.1. Baza próbek mowy

Wykonano dwie sesje nagrań, w dwóch różnych miejscach, w dużym odstępie czasu.

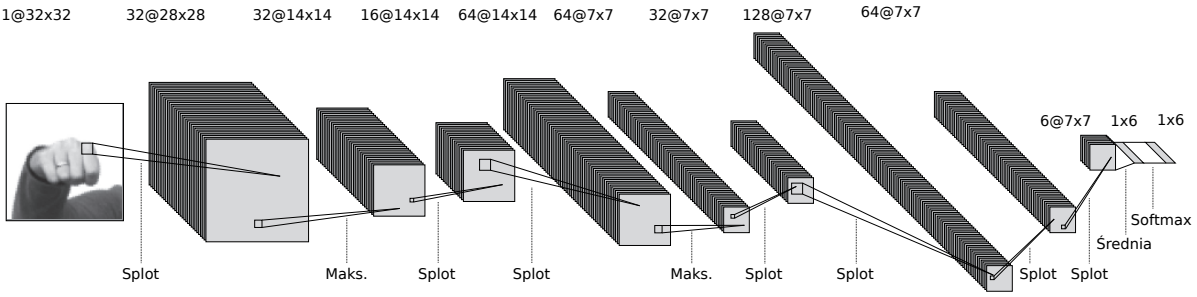
Do nagrań w pierwszej sesji wykorzystano następujące trzy mikrofony stacjonarne: Sennheiser MKE 600 (zamontowany na statywie), the t.bone EM 9600 (trzymany w ręku) i Shure SM58 LE (zamontowany na statywie) oraz moduł bezprzewodowy AKG WMS 40 Mini Sport ISM3 z mikrofonem nagłownym Samson DE10.

Do nagrań w czasie drugiej sesji korzystano z czterech mikrofonów: Sennheiser MKE 600, Shure SM58, mikrofonu konferencyjnego AKG CGN321 STS (nowy, niewystępujący w pierwszej sesji) i nagłownego AKG Samson DE10.

W pierwszej sesji zebrano nagrania próbek mowy dla 11 mówców – dla każdego mówcy zebrano minimum po dziewięć próbek stacjonarnymi mikrofonami i po sześć próbek mikrofonem nagłownym, dla każdej z 10 komend głosowych. Komendy nagrywano po kolei, do pojedynczego pliku, a następnie dokonano podziału na poszczególne komendy. Każdą komendę nagrano przynajmniej trzykrotnie, każdym z trzech mikrofonów stacjonarnych. Dla mikrofonu nagłownego komendy nagrano co najmniej sześciokrotnie dla każdego mówcy. W grupie 11 mówców było ośmiu mężczyzn i trzy kobiety.

W drugiej sesji nagrań zebrano próbki nagrań tych samych 10 komend głosowych od siedmiu mówców (sami mężczyźni), z których dwóch już występowało w pierwszej sesji, a pozostałych pięciu – nie. Dla każdego mówcy nagrano przynajmniej sześć próbek każdej komendy trzema mikrofonami stacjonarnymi i sześć próbek mikrofonem nagłownym.

Lista komend głosowych, dla których zostały zebrane próbki mowy, to: `obraz ogólny`, `najważniejsze zagro-`



Rys. 3. Struktura sieci neuronowej odpowiedzialnej za klasyfikację gestów statycznych
Fig. 3. Structure of the neural network for static gesture classification

żenia, `uwaga gość`, `po gościu`, `ostatnie zgłoszenia`, `mapa zagrożeń`, `sektor transportowy`, `sektor medyczny`, `sektor bankowy` i `sektor rządowy`.

Tym samym utworzono cztery bazy próbek:

Baza 1A – pierwsza sesja, mikrofony stacjonarne zawiera próbki pochodzące z trzech mikrofonów stacjonarnych – przynajmniej po sześć próbek na komendę i mówcę, w tym po dwie z każdego mikrofonu. Wybrano do testów próbki dla 11 mówców × 10 komend × 6 próbek, czyli 660 próbek;

Baza 1B – druga sesja, mikrofony stacjonarne próbki mowy siedmiu mówców – po sześć próbek dla każdej z 10 komend;

Baza 2A – pierwsza sesja, mikrofon nagłowny zawiera próbki pochodzące z mikrofonu nagłownego – 11 mówców × 10 komend × 6 próbek. W sumie także zebrano 660 próbek;

Baza 2B – druga sesja, mikrofon nagłowny próbki mowy siedmiu mówców – po sześć próbek dla każdej z 10 komend nagranych mikrofonem nagłownym.

Procesem nagrań kierował serwisant działający w roli administratora systemu, a uczestniczyli w nim kolejno rejestrowani bezpośredni użytkownicy. Do zebrania próbek korzystano jedynie z modułu audio.

4.2. Sposób przeprowadzenia testów

System rozpoznawania komend wykorzystuje nauczony wcześniej model. Danymi służącymi uczeniu modelu były próbki głosu grupy 11 osób, nagranych w pierwszej sesji. Moduł audio wyposażono w kilkanaście modeli komend – po jednym specjalizowanym dla indywidualnego mówcy, po jednym modelu dla kobiet i jednym dla mężczyzn oraz jednym ogólnym. Model ogólny powstaje na podstawie próbek nagrań głosu wszystkich osób z uczestniczących w nagraniach. Modele specjalizowane oparte są na próbkach nagrań kolejnych pojedynczych mówców. Testy modułu audio przeprowadzone były na ogólnym modelu mówcy i na modelach pojedynczych mówców.

4.2.1. Testy off-line

Wyróżniamy dwa rodzaje testów – wykonane w warunkach **off-line** i **on-line**. Pierwsze z nich wykonano na zbiorze wcześniej nagranych próbek uczących i testowych składających się na cztery bazy nagrań, określonych powyżej jako „1A, 2A" i „1B, 2B".

— Dla baz 1A i 2A przyjęto podział każdego zbioru próbek na dwa podzbiory – trzy próbki komendy każdego mówcy brały udział w procesie uczenia (tworzenia) modeli, a pozostałe sześć wchodziło w skład podzbioru testowego. Próbki baz 1B i 2B przeznaczono jedynie do testowania z modelami utworzonymi dla 1A i 2A.

4.2.2. Testy on-line

Drugi typ testów (on-line) prowadziła osoba, która nie była zarejestrowana w pierwszej sesji, tzn. nie istniał dla niej model głosowy mówcy ani też jej specjalizowany model komend. Dlatego testy rozpoznawania w tym trybie wykonano jedynie z wykorzystaniem ogólnego modelu komend. Podczas rozpoznawania tego mówcy właściwym wynikiem powinien być zawsze identyfikator „0", co oznacza „brak identyfikacji" zarejestrowanego mówcy.

Testy on-line polegały na wypowiedzeniu przez użytkownika wszystkich 10 poleceń. Każde polecenie wypowiadane było po 10 razy. Testy zostały powtórzone dla każdego z czterech mikrofonów wybranych do testów (podanych wyżej). Mówca znajdował się w odległości od kilku do 20 cm od mikrofonów.

4.3. Wyniki rozpoznawania komend w trybie off-line

Procesem tworzenia modeli komend i rozpoznawania komend kierował serwisant, działając w roli administratora, a realizował go agent audio a_{audio}.

W bazie 1A dla każdego z 11 mówców i każdej z 10 komend zebrano przynajmniej 9 próbek wypowiedzi. Każdy zbiór 9 próbek tej samej wypowiedzi, tego samego mówcy, został podzielony na część treningową (próbki 1–3) i część testową (próbki 4–9). Stworzono model komend i modele mówców w oparciu o zbiór próbek treningowych. Przetestowano skuteczność rozpoznawania komend i mówców w oparciu o zbiór testowych próbek.

Każda próbka była rozpoznawana niezależnie od pozostałych.

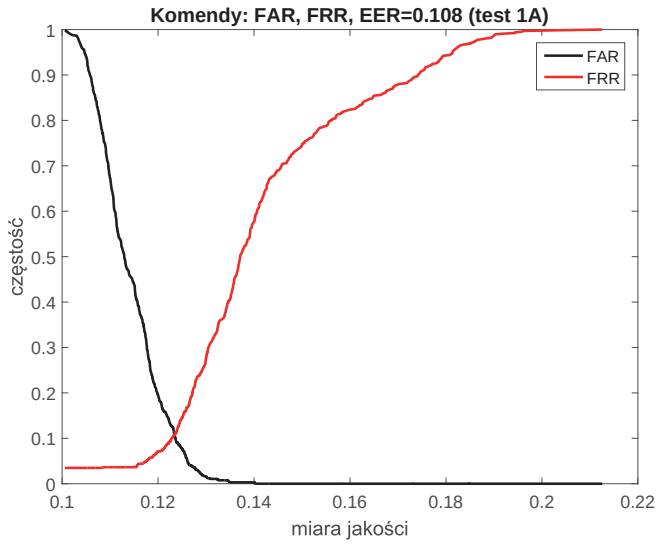
Bezwzględna skuteczność rozpoznawania komend (tzn. przy braku warunku spełniania określonego progu jakości) przy jednym zbiorczym modelu dla wszystkich mówców wyniosła, zależnie od podziału zbioru na próbki uczące i testowe, 92,4% – 93,9% (dla próbek z bazy 1A) i 89,4% – 90,1% (dla próbek z bazy 1B).

Kolejny test polegał na dołączeniu próbek testowych z bazy 1B do 1A. W większości (pięciu z siedmiu) byli to inni mówcy niż rejestrowani w pierwszej sesji, dla których utworzono modele komend i mówców. Wyniki zbiorcze rozpoznawania komend nieznacznie pogorszyły się. Dla połączonych baz 1A i 1B odnotowano bezwzględną skuteczność 90,1% – 90,4%.

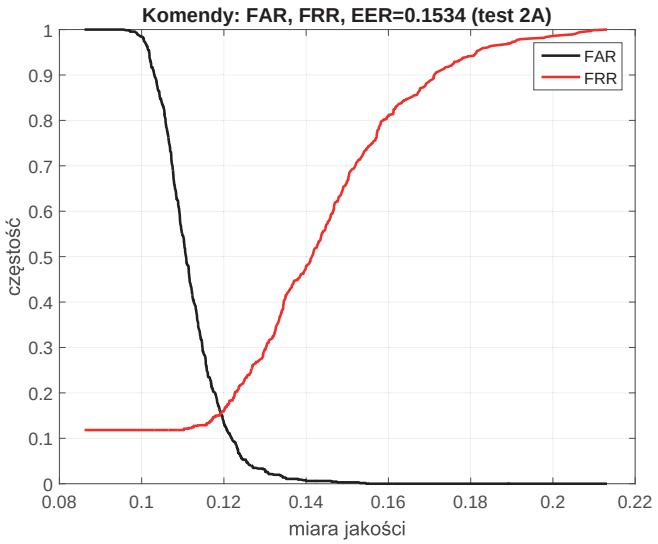
Charakterystyka systemu za pomocą kryteriów biometrycznych podana jest na rysunku 4 (dla bazy 1A) i na rysunku 5 (dla połączonych baz 1A i 1B). Punkt EER – punkt równowagi częstości błędnej akceptacji FAR (ang. *false acceptance rate*) i częstości błędnego odrzucenia FRR (ang. *false rejection rate*) – wyniósł odpowiednio 0,108 (dla bazy 1A) i 0,159 (dla połączonych baz 1A i 1B).

Podobnie zrealizowano testy dla baz próbek 2A i 2B pochodzących z mikrofonu nagłownego, z tym że tu część treningowa i część testowa zawierały obie po trzy próbki dla każdej komendy każdego mówcy. Wyniki rozpoznawania komend były o około 5% gorsze niż dla baz 1A i 1B. Dla obu baz 2A i 2B odnotowano podobną bezwzględną skuteczność rozpoznawania komend wynoszącą około 87%, a punkt EER wyniósł 0,154 (dla bazy 2A) i 0,216 (dla połączonych baz 2A i 2B) (rys. 6, rys. 7).

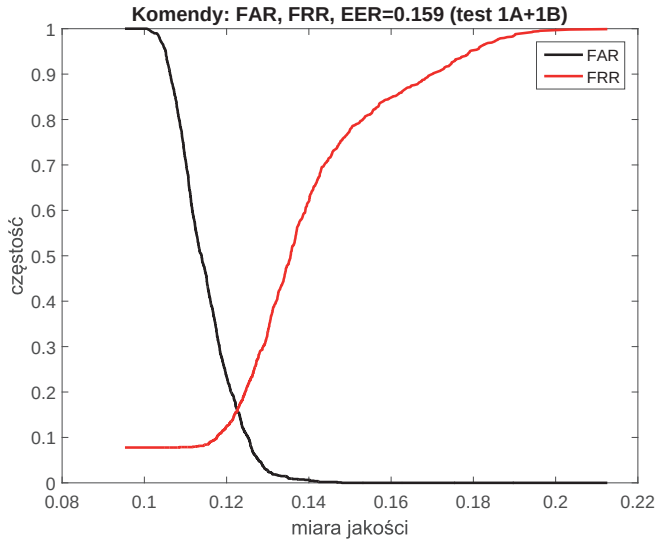
Jeżeli założymy teraz, że mówca każdej próbki testowej został prawidłowo zidentyfikowany, można wtedy stosować



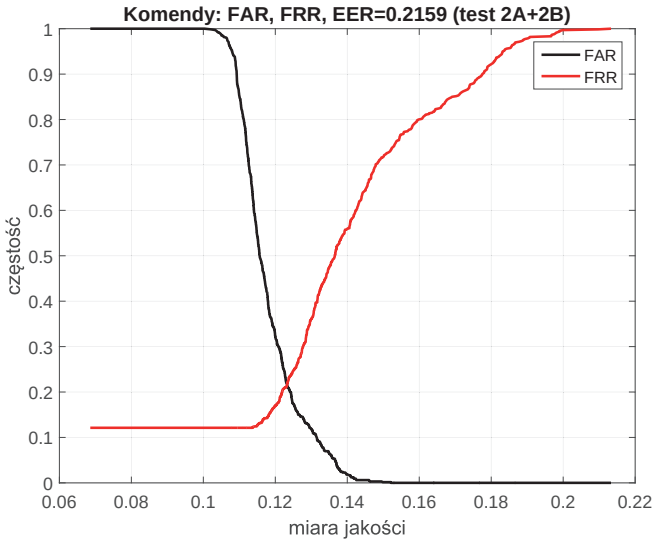
Rys. 4. Wykres krzywych FAR i FRR dla procesu rozpoznawania komend dla próbek w bazie 1A
Fig. 4. FAR and FRR plots for spoken command recognition for samples from database 1A



Rys. 6. Wykres krzywych FAR i FRR dla procesu rozpoznawania komend dla próbek w bazie 2A
Fig. 6. FAR and FRR plots for spoken command recognition for samples from database 2A



Rys. 5. Wykres krzywych FAR i FRR dla procesu rozpoznawania komend łącznie dla próbek w bazach 1A i 1B
Fig. 5. FAR and FRR plots for spoken command recognition for samples from joined databases 1A and 1B



Rys. 7. Wykres krzywych FAR i FRR dla procesu rozpoznawania komend łącznie dla próbek w bazach 2A i 2B
Fig. 7. FAR and FRR plots for spoken command recognition for samples from joined databases 2A and 2B

własny model komend danego mówcy. Przy takim postepowaniu skuteczność rozpoznawania zwiększyła się średnio do 91,2% (dla połączonych baz 1A i 1B) i 87,8% (dla połączonych baz 2A i 2B).

Przy ocenie skuteczności rozpoznawania komend należy uwzględnić, że zbiór uczący był stosunkowo nieliczny, a jakość próbek nagrywanych różnymi kanałami akwizycji (o zmiennych ustawieniach czułości) była bardzo zróżnicowana.

4.4. Wyniki rozpoznawania komend w trybie on-line

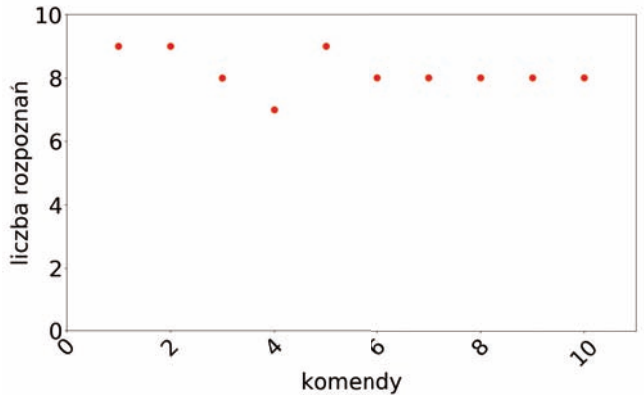
Do testów on-line dysponowano dwoma mikrofonami stacjonarnymi – Sennheiser MKE 600 i Shure SM58 – oraz mikrofonem nagłownym AKG LE Samson DE10. Każda z 10 komend była powtarzana 10-krotnie dla każdego z mikrofonów. Próg decyzyjny ustawiony był na wartość miary jakości wynosząca 0,12, czyli w okolicy punktu EER (odczytanego z wykresów skuteczności biometrycznej w testach off-line). To oznacza, że nie każda próba wydania komendy kończyła się rozpoznaniem jednej z 10 komend. Nie przyjęto jednak żadnych dalszych ograniczeń dla samego nagrania, np. spełniania przez nagranie wymogu minimalnej i maksymalnej średniej energii. Tym samym zbyt ciche lub zbyt głośne nagrania w praktyce zdarzały się, co wpływało negatywnie na proces ich rozpoznawania. Można uznać, że testowano działanie systemu w wersji on-line, pracującego w symulowanych warunkach rzeczywistych.

Wykresy (a)–(c) na rys. 8 obrazują prawidłowość rozpoznania poszczególnych komend przy stosowaniu każdego z trzech dostępnych mikrofonów. Najwięcej poprawnych rozpoznań uzyskano dla mikrofonów Sennheiser MKE 600 i Shure SM58. Najmniej rozpoznań stwierdzono dla mikrofonu nagłownego AKG Samson. Stwierdzono następujące częstości prawidłowej akceptacji GAR (ang. *genuine acceptance rate*): 82% (Sennheiser MKE 600), 82% (Shure SM58) i 73% (nagłowny AKG Samson). Dla porównania, przy progu 0,12 w testach on-line uzyskano dla

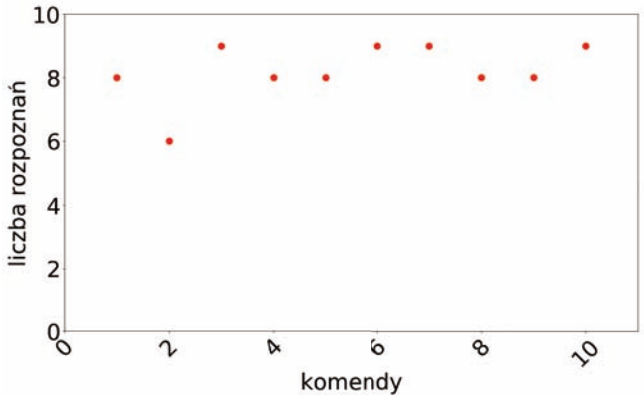
obu połączonych baz 1A i 1B (mikrofony stacjonarne) średnią skuteczność 84,1% a dla połączonych baz 2A i 2B (mikrofony nagłowne) – około 78%. Różnica skuteczności rozpoznania w spodziewanym punkcie EER między testem off-line a on-line wyniosła około 3–5% na korzyść testu off-line.

Wykresy (a)–(b) na rys. 9 są macierzami pomyłek (ang. *confusion matrix*). Jako „dane" oznaczono komendy, które były wypowiadane przez mówcę, zaś jako „rozpoznanie" oznaczono komendy, które były rozpoznane przez moduł audio. Jak można było oczekiwać, większość pomyłek dotyczy komendy nr 7, która jest zbliżona do komend nr 8–10.

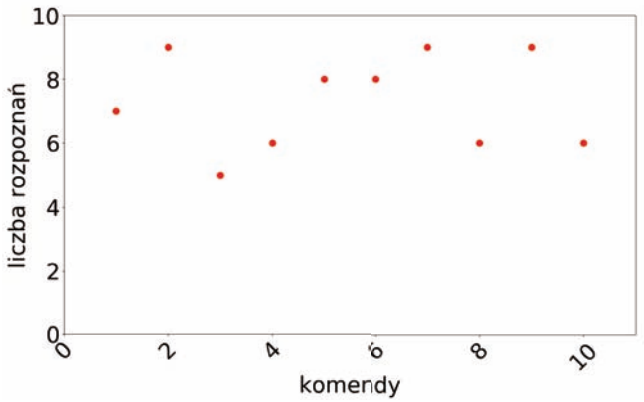
(a) Sennheiser MKE 600 – 82 rozpoznań na 100 prób



(b) Shure SM58 – 82 rozpoznań na 100 prób

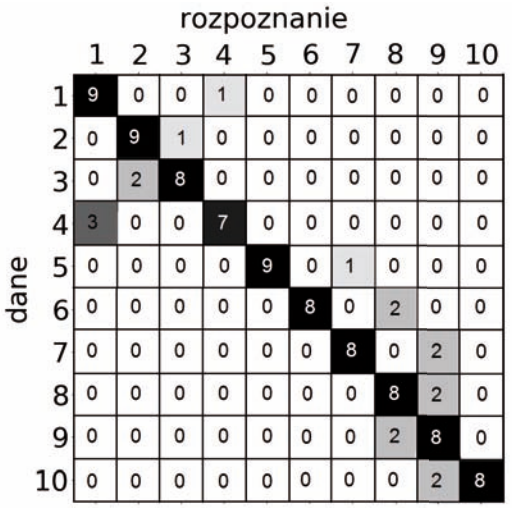


(c) AKG (nagłowny) – 73 rozpoznań na 100 prób



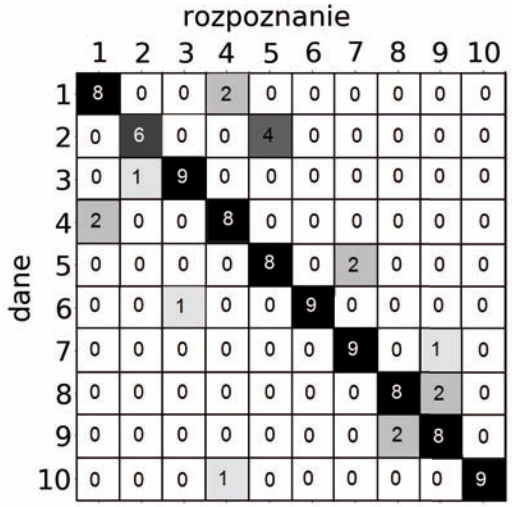
Rys. 8. Suma rozpoznań prawidłowych i nieprawidłowych (każda komenda wypowiedziana została 10 razy)
Fig. 8. Sum of proper and improper recognitions (each command repeated 10 times)

(a) Sennheiser MKE 600



dane \ rozpoznanie	1	2	3	4	5	6	7	8	9	10
1	9	0	0	1	0	0	0	0	0	0
2	0	9	1	0	0	0	0	0	0	0
3	0	2	8	0	0	0	0	0	0	0
4	3	0	0	7	0	0	0	0	0	0
5	0	0	0	0	9	0	1	0	0	0
6	0	0	0	0	0	8	0	2	0	0
7	0	0	0	0	0	0	8	0	2	0
8	0	0	0	0	0	0	0	8	2	0
9	0	0	0	0	0	0	0	2	8	0
10	0	0	0	0	0	0	0	0	2	8

(b) Shure SM58



dane \ rozpoznanie	1	2	3	4	5	6	7	8	9	10
1	8	0	0	2	0	0	0	0	0	0
2	0	6	0	0	4	0	0	0	0	0
3	0	1	9	0	0	0	0	0	0	0
4	2	0	0	8	0	0	0	0	0	0
5	0	0	0	0	8	0	2	0	0	0
6	0	0	1	0	0	9	0	0	0	0
7	0	0	0	0	0	0	9	0	1	0
8	0	0	0	0	0	0	0	8	2	0
9	0	0	0	0	0	0	0	2	8	0
10	0	0	0	1	0	0	0	0	0	9

Rys. 9. Macierze pomyłek dla testu rozpoznawania 10 komend
Fig. 9. Confusion matrices from the recognition test of 10 commands

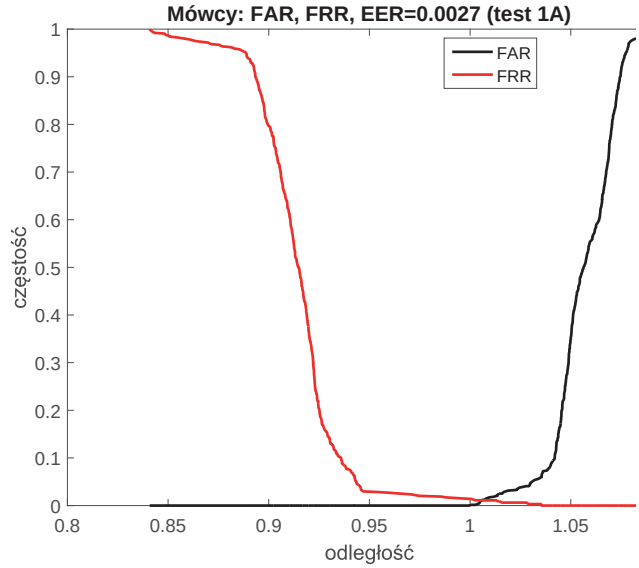
4.5. Wyniki rozpoznawania mówców w trybie off-line

Identyfikacja mówcy na podstawie pojedynczych słów zasadniczo nie jest nigdy wystarczająco skuteczna. Dlatego zastosowano procedurę identyfikacji mówcy wykonywaną wtedy, gdy znane są wypowiedzi co najmniej trzech mówców. Decyzję podejmowano po akumulacji wyniku rozpoznania dla minimum trzech kolejnych próbek, ale maksymalnie dziesięciu kolejnych próbek. Proces testowania był podzielony na niezależnie od siebie wykonywane rozpoznawanie próbek każdego mówcy z osobna. Dla jednego mówcy dysponowano 30 próbkami (komendami). Wyniki rozpoznania dwóch pierwszych próbek nie prowadzą do decyzji o identyfikacji, lecz są wewnętrznie akumulowane. Po dodaniu wyniku rozpoznania trzeciej próbki następuje pierwsza decyzja o identyfikacji mówcy. Teraz dodanie wyniku każdej kolejnej próbki prowadzi do kolejnych decyzji, a liczba akumulowanych wyników rośnie aż osiągnie liczbę dziesięciu próbek, czyli do wyczerpania maksymalnej pojemności pamięci pojedynczych wyników. Wtedy najnowszy wynik zastępuje w tej pamięci wynik najstarszy – innymi słowy stworzono bufor cykliczny. Każda kolejna decyzja podejmowana jest na podstawie akumulacji wyników rozpoznania 10 pojedynczych próbek. Podsumowując, liczba próbek jednego mówcy, wynosząca 30, prowadzi do 28 decyzji o identyfikacji mówcy.

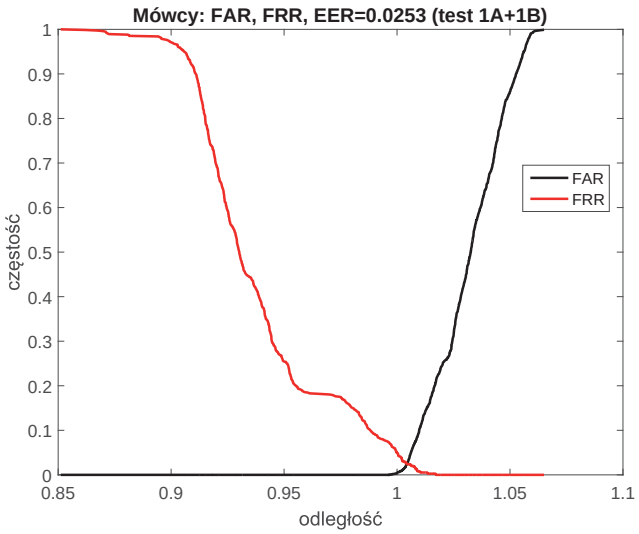
Jako sukces uznano sytuację, gdy próbka danego mówcy została przypisana poprawnie do mówcy nazwanego *genuine* (autentyczny), czyli nie została przypisana do żadnego z pozostałych znanych mówców pełniących wtedy rolę *imposter* (podszywającego się). Jednak warunkiem poprawnej identyfikacji jest także, aby odstęp próbki od modelu mówcy *genuine* był niższy (o zadaną względną odległość 2%, czyli zastosowano próg odległości 0,98) od odległości do modelu UBM, reprezentującego „wszystkich mówców". Jako „brak decyzji" zaznaczono sytuację, gdy wynik dla mówcy *genuine* jest najlepszy spośród zarejestrowanych mówców, ale pozostaje zbyt bliski wynikowi dla modelu UBM, lub też, gdy „wygrywa" model UBM.

Charakterystyka systemu za pomocą kryteriów biometrycznych podana jest na rys. 10 i 11. W podanych warunkach (dla progu względnego odstępu 0,98) stopa poprawnej identyfikacji mówcy wynosiła około 97% dla bazy 1A i około 89% dla bazy 2A, w obu przypadkach przy zerowej stopie błędnej akceptacji. Należy zauważyć, że uzyskano wysoką skuteczność identyfikacji, mimo iż typowe wyniki odległości dla mówców *genuine* były jedynie o kilka procent lepsze (średnio 5,9–6,4%) od wyników odległości dla modelu uniwersalnego mówcy UBM. Odstęp wyników dla mówców *imposter* od mówców *genuine* był zawsze większy o kilka procent od odstępu dla modelu UBM, a to oznacza, że możliwe było nawet zastosowanie progu 1,0, co zwiększyłoby stopę poprawnej identyfikacji do niemal 100%.

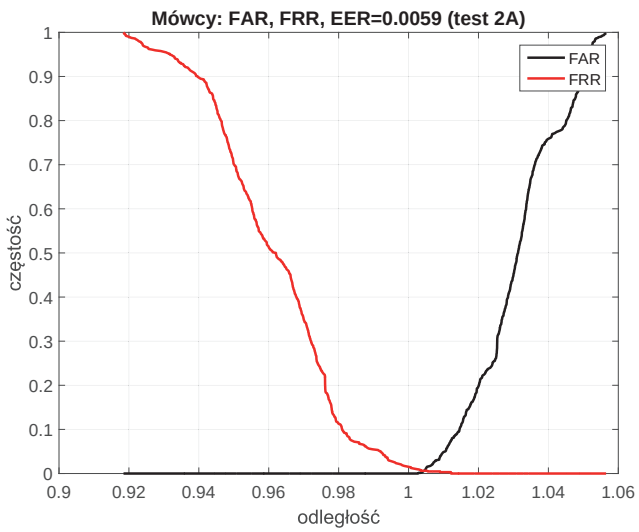
Po dołączeniu baz próbek 1B (do 1A) i 2B (do 2A) obserwujemy (spodziewane) pogorszenie się wyników rozpoznawania mówców wyrażające się wzrostem wartości EER i przesunięciem się centrów klastrów (głównie dotyczy to próbek z bazy 2B względem próbek z bazy 2A). Charakterystyka systemu testowanego łącznie na bazach 1A i 1B za pomocą kryteriów biometrycznych podana jest na rys. 12, a testowanego na bazach 2A i 2B – na rys. 13.



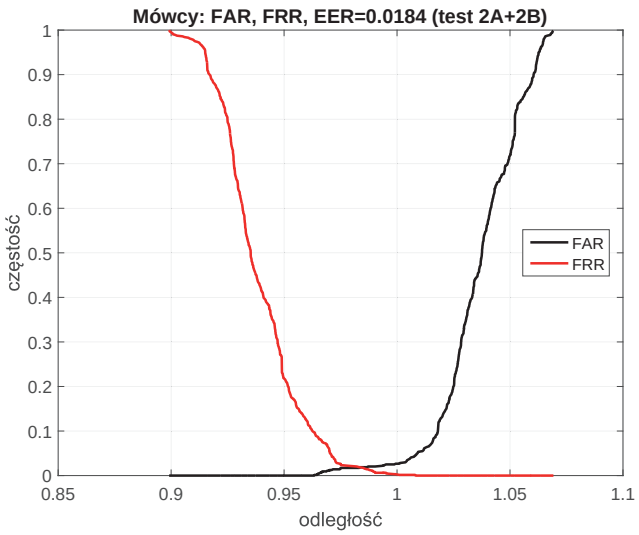
Rys. 10. Wykres krzywych FAR i FRR dla procesu rozpoznawania mówcy dla próbek w bazie 1A
Fig. 10. FAR and FRR plots for speaker recognition for samples from database 1A



Rys. 12. Wykres krzywych FAR i FRR dla procesu rozpoznawania mówcy łącznie dla próbek w bazach 1A i 1B
Fig. 12. FAR and FRR plots for speaker recognition for samples from joined databases 1A and 1B



Rys. 11. Wykres krzywych FAR i FRR dla procesu rozpoznawania mówcy dla próbek w bazie 2A
Fig. 11. FAR and FRR plots for speaker recognition for samples from database 2A

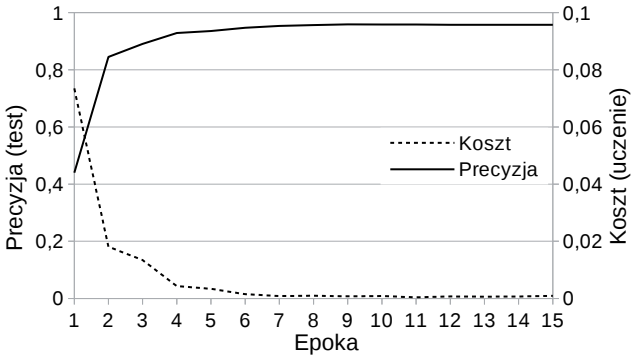


Rys. 13. Wykres krzywych FAR i FRR dla procesu rozpoznawania mówcy łącznie dla próbek w bazach 2A i 2B
Fig. 13. FAR and FRR plots for speaker recognition for samples from joined databases 2A and 2B

5. Wyniki rozpoznawania gestów

Cześć systemu odpowiedzialna za rozpoznawanie gestów dynamicznych wykorzystuje bezpośrednio dobrze przetestowane moduły z biblioteki GRT [12]. Z tego względu w tej sekcji przedstawione zostaną jedynie testy części odpowiedzialnej za rozpoznawanie statycznych gestów dłoni (ich przykłady przedstawiono na rys. 14).

Rys. 14. Przykłady gestów statycznych
Fig. 14. Sample static gestures



Rys. 15. Proces trenowania sieci CNN do rozpoznawania gestów statycznych
Fig. 15. Training of the static gesture classifier CNN

5.1. Uczenie gestów statycznych

Sieć neuronowa, przygotowana do rozpoznawania gestów statycznych, została wytrenowana na zbiorze ponad 3 tysięcy zdjęć. Do testowania przygotowano zbiór dodatkowych 2 tysięcy obrazów, których sieć nie widziała w trakcie uczenia. Obrazy zbierano w dwóch sesjach czasowych od 3 różnych operatorów.

5.2. Wyniki rozpoznawania – macierz pomyłek

Proces uczenia sieci został przedstawiony na rys. 15. Po 15 epokach sieć osiągnęła skuteczność powyżej 95% (na zbiorze testowym), dalsze uczenie prowadziło do przeuczenia i spadku jakości rozpoznawania.

Analiza macierzy pomyłek (rys. 16) pokazuje, że występują pewne niedoskonałości, natomiast ogólna jakość rozpoznawania jest na akceptowalnym poziomie. Dodatkowa poprawa jakości rozpoznawania jest możliwa do osiągniecia w przyszłości przez analizę czasową gestów i akceptację ich dopiero wtedy, gdy wystąpią przez kilka klatek pod rząd.

rozpoznanie

dane	0	1	2	3	4	5
0	335	0	3	0	0	0
1	0	251	1	0	0	0
2	0	0	252	0	45	0
3	0	0	8	178	0	0
4	0	0	13	0	235	0
5	0	0	3	0	0	397

Rys. 16. Macierz pomyłek dla zbioru testowego gestów statycznych
Fig. 16. Confusion matrix for the static gesture classifier

5.3. Testy użytkowania modułu

Oprócz ilościowych testów jakości rozpoznawania przeprowadzono także testy jakościowe pracy z systemem. Podczas testów użytkownik został poproszony o wykonanie różnych operacji, pokrywających możliwości modułu rozpoznawania gestów oraz podobnych do tych wykonywanych podczas codziennej pracy przy użyciu myszy. Operacje te zawierały m.in. nawigowanie na mapie (gesty dwuręczne), przeglądanie stron internetowych na wielu kartach (generujące kody skrótów klawiaturowych), nawigowanie w systemie (gesty jednoręczne) itp.

Wszystkie zaimplementowane funkcje działały poprawnie i pozwalały na skuteczne wykonanie zadań. Użytkownicy jednak zauważyli, że interakcja z systemem była męcząca. Po pierwsze, zaobserwowano trudności z precyzyjnym trafieniem w obiekty. W związku z tym wprowadzono osobny gest do aktywnego wskazywania obiektu, który rozwiązał ten problem. Po drugie, użytkownik odczuwał dyskomfort przy długotrwałym trzymaniu rąk w górze. W celu rozwiązania tego problemu zaproponowano dwa usprawnienia. Pierwsze z nich wprowadziło strefę aktywną sterowania. Jeśli operator przeniesie ręce poza nią, może je swobodnie opuścić, bez ryzyka wykonania niechcianej akcji na ekranie. Drugim usprawnieniem jest możliwość zdefiniowania przestrzeni roboczej w taki sposób, aby podnoszenie rąk nie było konieczne. To jednak może skutkować zmniejszeniem efektywnej rozdzielczości przy sterowaniu kursorami.

6. Podsumowanie i wnioski

Zaprezentowano wielomodalny interfejs do sterowania prezentacją wyników pracy Narodowej Platformy Cyberbezpieczeństwa. Do jego specyfikacji wykorzystano metodykę opartą na agentach upostaciowionych, jednocześnie wykazując, że jest ona przydatna nie tylko do realizacji systemów robotycznych. Dzięki zastosowaniu sieci współpracujących agentów struktura powstałego interfejsu ma naturę otwartą. To umożliwia względnie proste rozszerzenie struktury, a przez to zwiększenie jej możliwości – nowych trybów porozumiewania się z interfejsem. Ponadto obecne tryby mogą być rozszerzane przez dodawanie nowych komend głosowych oraz gestów sterujących, w tym wykonywanych głową oraz sterowanie wyrazem twarzy.

Podstawowymi sposobami komunikacji użytkownika z interfejsem są polecenia wydawane albo głosem albo gestem. Wprowadzono też bezpieczny tryb dostępu do platformy wykorzystujący identyfikację mówcy. Obecnie interfejs przechodzi fazę testowania. Wstępne testy wykazały wysoką skuteczność rozpoznawania gestów.

Nowatorskim elementem modułu audio jest połączenie rozpoznawania komend z identyfikacją mówcy w jednym module. Pozwala to najpierw zidentyfikować mówcę, lub nawet tylko określić, czy mówca jest kobietą czy mężczyzną, aby następnie w procesie rozpoznawania komend korzystać z dedykowanego modelu komend dla danego mówcy lub odrębnego modelu

Rys. 17. Użytkownik wykonujący testy funkcjonalne
Fig. 17. User performing functional tests

dla kobiet/mężczyzn. Wyniki testów jednoznacznie wskazują, że skuteczność rozpoznania komend rośnie przy stosowaniu dedykowanego modelu mówcy lub modelu przypisanego płci. Ponieważ trudno jest spodziewać się, że będziemy dysponowali dużymi zbiorami próbek pozyskanymi od mówców, aby polepszyć skuteczność rozpoznawania komend głosowych oraz mówców, wartości i liczba parametrów dobierane są eksperymentalnie.

Rozpoznawanie gestów statycznych zostało zrealizowane za pomocą stosunkowo niewielkiej sieci neuronowej. Eksperymenty z uczeniem zostały przeprowadzone na wielu sieciach różniących się między sobą m.in. strukturą. Spośród nauczonych sieci wybrano te o najwyższej skuteczności, jednakże uzyskane wyniki nieznacznie różniły się między poszczególnymi sieciami. Istotny wpływ na rezultaty uczenia miał zbiór trenujący, dlatego dalsze prace skoncentrują się na przygotowaniu większego i bardziej różnorodnego zbioru trenującego. Należy podkreślić, że przedstawiono wyniki rozpoznawania gestów statycznych bez ich analizy czasowej. Oznacza to, że gesty są rozpoznawane w pojedynczych obrazach w sekwencji, niezależnie od innych. Dodanie zależności czasowej między obrazami w sekwencji powinno podnieść skuteczność rozpoznawania gestów statycznych. W ramach przeprowadzonych badań nad dwoma metodami uczenia, uzyskaliśmy lepsze rezultaty dla współczesnych splotowych sieci neuronowych niż dla detektora kaskadowego. Z punktu widzenia inżyniera, przygotowanie danych trenujących oraz przeprowadzenie procedury uczenia wymaga podobnego wysiłku dla obu metod, stąd zalecamy używanie sieci neuronowych do zadań podobnych do opisanych w niniejszej pracy.

Podziękowania

Praca wykonana w ramach projektu CYBERSECIDENT/369195/I/NCBR/2017, współfinansowanego przez Narodowe Centrum Badań i Rozwoju w ramach programu CyberSecIdent.

Bibliografia

1. Alizé, opensource speaker recognition. [http://alize.univ-avignon.fr].
2. Benzeghiba M., De Mori R., Deroo O., Dupont S., Erbes T., Jouvet D., Fissore L., Laface P., Mertins A., Ris C., Rose R., Tyagi V., Wellekens C., *Automatic speech recognition and speech variability: A review.* "Speech Communication", Vol. 49, No. 10–11, 763–786, 2007, DOI: 10.1016/j.specom.2007.02.006.
3. Bolt R.A., *"Put-that-there": Voice and gesture at the graphics interface.* ACM SIGGRAPH Computer Graphics, Vol. 14, No. 3, 1980, 262–270, DOI: 10.1145/800250.807503.
4. Borges P.V.K., Conci N., Cavallaro A., *Video-based human behavior understanding: A survey.* "IEEE Transactions on Circuits and Systems for Video Technology", Vol. 23, No. 11, 1993–2008, 2013, DOI: 10.1109/TCSVT.2013.2270402.
5. Cao Z., Simon T., Wei S., Sheikh Y., *Realtime Multi-Persson 2D Pose Estimation using Part Affinity Fields* , IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 1302–1310, 2017.
6. Chen L., Wei H., Ferryman J., *A survey of human motion analysis using depth imagery.* "Pattern Recognition Letters", Vol. 34, No. 15, 1995–2006, 2013, DOI: 10.1016/j.patrec.2013.02.006.
7. Chiu C., Sainath T.N., Wu Y., Prabhavalkar R., Nguyen P., Chen Z., Kannan A., Weiss R.J., Rao K., Gonina E., Jaitly N., Li B., Chorowski J., Bacchiani M., *State-of-the-art speech recognition with sequence-to-sequence models.* IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 4774–4778, 2018, DOI: 10.1109/ICASSP.2018.8462105.
8. Chung J., Nagrani A., Zisserman A., *Voxceleb2: Deep speaker recognition.* INTERSPEECH 2018.
9. Divekar R.R., et al. *CIRA: An architecture for building configurable immersive smart-rooms.* K. Arai, S. Kapoor,

R. Bhatia (eds.), *Intelligent Systems and Applications. IntelliSys 2018*, Vol. 869 serii *Advances in Intelligent Systems and Computing*, 76–95, Springer, 2019.
10. Dumas B., Lalanne D., Oviatt S.. *Human Machine Interaction*, Vol. 5440 serii *Lecture Notes in Computer Science*, rozdz. *Multimodal Interfaces: A Survey of Principles, Models and Frameworks*, 3–26. Springer, 2009.
11. Gillian N., Paradiso J.A., *The gesture recognition toolkit*. "Journal of Machine Learning Research", Vol. 15, No. 1, 3483–3487, 2014.
12. Gillian N.E., Knapp R.B., O'Modhrain M.S., *Recognition of multivariate temporal musical gestures using n-dimensional dynamic time warping*. NIME, 2011.
13. Graves A., Jaitly N.. *Towards end-to-end speech recognition with recurrent neural networks*. ICML'14 Proceedings of the 31st International Conference on International Conference on Machine Learning, Vol. 32, II-1764–II-1772.
14. Gravier G., SPro: Speech Signal Processing Toolkit, 2010.
15. Herath S., Harandi M., Porikli F., *Going deeper into action recognition: A survey*. "Image and Vision Computing", Vol. 60, 4–21, 2017, DOI: 10.1016/j.imavis.2017.01.010.
16. Hirschmuller H., *Stereo processing by semiglobal matching and mutual information*. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 30, No. 2, 328–341, 2008, DOI: 10.1109/TPAMI.2007.1166.
17. Jaimes A., Sebe N.. *Multimodal human–computer interaction: A survey*. "Computer Vision and Image Understanding", Vol. 108, No. 1, 116–134, 2007, DOI: 10.1016/j.cviu.2006.10.019.
18. Kaldi. *The kaldi project*. [http://kaldi.sourceforge.net/index.html].
19. Kapuscinski T., Oszust M., Wysocki M., Warchol D., *Recognition of hand gestures observed by depth cameras*. "International Journal of Advanced Robotic Systems", Vol. 12, No. 4, 2015, DOI: 10.5772/60091.
20. Kasprzak W., *Rozpoznawanie obrazów i sygnałów mowy*. Oficyna Wydawnicza Politechniki Warszawskiej, 2009.
21. Kasprzak W., Przybysz P., *Stochastic modelling of sentence semantics in speech recognition*. Computer Recognition Systems 4, Vol. 95 serii *Advances in Intelligent and Soft Computing*, 737–746, Berlin, Heidelberg, 2011, Springer-Verlag, DOI: 10.1007/978-3-642-20320-6_75.
22. Kasprzak W., Wilkowski A., Czapnik K., *Hand gesture recognition based on free-form contours and probabilistic inference*. "International Journal of Applied Mathematics and Computer Science", Vol. 22, No. 2, 437–448, 2012, DOI: 10.2478/v10006-012-0033-6.
23. Kazemi V., Sullivan J., *One millisecond face alignment with an ensemble of regression trees. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1867–1874, 2014, DOI: 10.1109/CVPR.2014.241.
24. Koller O., Ney H., Bowden R., *Deep Hand: How to Train a CNN on 1 Million Hand Images When Your Data is Continuous and Weakly Labelled*. IEEE Conference on Computer Vision and Pattern Recognition, 3793–3802, 2016, DOI: 10.1109/CVPR.2016.412.
25. Krizhevsky A., Sutskever I., Hinton G.E., *ImageNet classification with deep convolutional neural networks*. NIPS'12 Proceedings of the 25th International Conference on Neural Information Processing Systems, Vol. 1, 1097–1105, 2012.
26. *Loudia. The loudia library*. [https://github.com/rikrd/loudia].
27. Lowe D.G., *Distinctive image features from scale-invariant keypoints*. "International Journal of Computer Vision, Vol. 60, No. 2, 91–110, 2004, DOI: 10.1023/B:VISI.0000029664.99615.94.
28. Lukic Y., Vogt C., Dürr O., Stadelmann T., *Speaker identification and clustering using convolutional neural networks*. IEEE 26th InternationalWorkshop on Machine Learning for Signal Processing (MLSP), 2016, DOI: 10.1109/MLSP.2016.7738816.
29. Mak M.-W., Chien J.-T., *Machine learning for speaker recognition*. INTERSPEECH 2016 Tutorial, [www.eie.polyu.edu.hk/mwmak/papers/IS2016-tutorial.pdf], 2016.
30. MARF, [marf.sourceforge.net].
31. Matarneh R., Maksymova S., Lyashenko V., Belova N.V., *Speech recognition systems: A comparative review*. "Journal of Computer Engineering", Vol. 19, No. 5, 71–79, 2017.
32. *Mistral. The mistral biometric platform*. [http://mistral.univ-avignon.fr].
33. Nayana P., Mathew D., Thomas A., *Comparison of Text Independent Speaker Identification Systems using GMM and i-Vector Methods*. "Procedia Computer Science", Vol. 115, 47–54, 2017, DOI: 10.1016/j.procs.2017.09.075.
34. Nunnally T., Uluagac A.S., Beyah R., *InterSec: An interaction system for network security applications*. IEEE International Conference on Communications (ICC), 7132–7138, 2015, DOI: 10.1109/ICC.2015.7249464.
35. Oikonomopoulos A., Patras I., Pantic M., *Spatiotemporal localization and categorization of human actions in unsegmented image sequences*, "IEEE Transactions on Image Processing", Vol. 20, No. 4, 1126–1140, 2010, DOI: 10.1109/TIP.2010.2076821.
36. Oviatt S., *Ten myths of multimodal interaction*. Communications of the ACM, Vol. 42, No. 11, 74–81, 1999, DOI: 10.1145/319382.319398.
37. Oviatt S., Schuller B., Cohen P., Sonntag D., Potamianos G., Kruger A. (eds.), *The Handbook of Multimodal-Multisensor Interfaces,* Vol. 1: *Foundations, User Modeling, and Common Modality Combinations*. ACM Books Series. Association for Computing Machinery (ACM), 2017.
38. Pedersoli F., Benini S., Adami N., Leonardi R., *XKin: an open source framework for hand pose and gesture recognition using kinect*. "The Visual Computer", Vol. 30, No. 10, 1107–1122, 2014, DOI: 10.1007/s00371-014-0921-x.
39. Poppe R., *A survey on vision-based human action recognition*. "Image and Vision Computing", Vol. 28, No. 6, 976–990, 2010, DOI: 10.1016/j.imavis.2009.11.014.
40. Presti L.L., La Cascia M., *3D skeleton-based human action classification: A survey*. "Pattern Recognition", Vol. 53, 130–147, 2016, DOI: 10.1016/j.patcog.2015.11.019.
41. Purwins H., Li B., Virtanen T., Schlüter J., Chang S., Sainath T., *Deep learning for audio signal processing*. "IEEE Journal of Selected Topics in Signal Processing", Vol. 13, No. 2, 206–219, 2019, DOI: 10.1109/JSTSP.2019.2908700.
42. Rabiner L.R., Juang B.-H., *Historical Perspective of the Field of ASR/NLU*, 521–538. Springer, Berlin, Heidelberg, 2008.
43. Rabiner L.R., Schafer R.W., *Introduction to digital speech processing*. Foundations and Trends in Signal Processing, 1(1):1–194, 2007.
44. Reynolds D., Quatieri T.F., Dunn R.B., *Speaker verification using adapted Gaussian mixture models*. "Digital Signal Processing", Vol. 10, No. 1–3, 19–41, 2000, DOI: 10.1006/dspr.1999.0361.
45. Reynolds D., Rose R.C., Dunn R.B., *Robust text-independent speaker identification using Gaussian mixture speaker models*. "IEEE Transactions on Speech and Audio Processing", Vol. 3, No. 1, 72–83, 1995, DOI: 10.1109/89.365379.
46. Salvador S., Chan P., *FastDTW: Toward accurate dynamic time warping in linear time and space*.
47. Schafer R.W., *Homomorphic Systems and Cepstrum Analysis of Speech*, 161–180. Springer, Berlin, Heidelberg, 2008.
48. Sinha A., Choi C., Ramani K., *DeepHand: Robust hand pose estimation by completing a matrix imputed with deep features*. Proceedings of the IEEE Conference on Computer

Vision and Pattern Recognition, 4150–4158, 2016, DOI: 10.1109/CVPR.2016.450.
49. Turk M., *Multimodal interaction: A review*. "Pattern Recognition Letters", Vol. 36, 189–195, 2014, DOI: 10.1016/j.patrec.2013.07.003.
50. Ullah A., Ahmad J., Muhammad K., Sajjad M., Baik S.W., *Action recognition in video sequences using deep bi-directional LSTM with CNN features*. "IEEE Access", Vol. 6, 1155–1166, 2017, DOI: 10.1109/ACCESS.2017.2778011.
51. Viola P., Jones M., *Rapid object detection using boosted cascade of simple features*. IEEE Conference on Computer Vision and Pattern Recognition, 2001, DOI: 10.1109/CVPR.2001.990517.
52. Walker W., Lamere P., Kwok P., Raj B., Singh R., Gouvea E., Wolf P., Woelfel J., *Sphinx-4: A flexible open source framework for speech recognition*. SMLI TR2004-0811, SUN Microsystems Inc., 2004, [http://cmusphinx.sourceforge.net/sphinx4].
53. Weinland D., Ronfard R., Boyer E., *A survey of vision-based methods for action representation, segmentation and recognition*. "Computer Vision and Image Understanding", Vol. 115, No. 2, 224–241, 2011, DOI: 10.1016/j.cviu.2010.10.002.
54. Woodland P., Evermann G., Gales M., *HTK book*. Cambridge University Engineering Department (CUED). 2000–2006.
55. Young S., *HMMs and Related Speech Recognition Technologies*, 539–558. Springer, Berlin, Heidelberg, 2008.
56. Yu Z., Zhang C., *Image based static facial expression recognition with multiple deep network learning*. Proceedings of the 2015 ACM on International Conference on Multimodal Interaction, 435–442, ACM, 2015, DOI: 10.1145/2818346.2830595.
57. Zhao R., Wang K., Divekar R., Rouhani R., Su H., Ji Q., *An immersive system with multi-modal human-computer interaction*. 13th IEEE International Conference on Automatic Face Gesture Recognition (FG 2018), 517–524, 2018, DOI: 10.1109/FG.2018.00083.

Agent Structure of Multimodal User Interface to the National Cybersecurity Platform – Part 2

Abstract: This two part paper presents an interface to the National Cybersecurity Platform utilising gestures and voice commands as the means of interaction between the operator and the platform. Cyberspace and its underlying infrastructure are vulnerable to a broad range of risk stemming from diverse cyber-threats. The main role of this interface is to support security analysts and operators controlling visualisation of cyberspace events like incidents or cyber-attacks especially when manipulating graphical information. Main visualization control modalities are gesture- and voice-based commands. Thus the design of gesture recognition and speech-recognition modules is provided. The speech module is also responsible for speaker identification in order to limit the access to trusted users only, registered with the visualisation control system. This part of the paper focuses on the structure and the activities of the interface, while the second part concentrates on the algorithms employed for the recognition of: gestures, voice commands and speakers.

Keywords: National Cybersecurity Platform, image recognition, gesture recognition, speech recognition, speaker recognition

prof. dr hab. inż. Włodzimierz Kasprzak
W.Kasprzak@elka.pw.edu.pl
ORCID: 0000-0002-4840-8860

Jest profesorem na Wydziale Elektroniki i Technik Informacyjnych (WEiTI) Politechniki Warszawskiej (PW). W PW pracuje od 1997 r. Obecnie jest kierownikiem Zespołu Percepcji Maszyn w IAiIS. Jego zainteresowania badawcze obejmują algorytmy i techniki automatycznej analizy obrazów i sygnału mowy oraz metody rozpoznawania wzorców i uczenia maszynowego. Jest autorem/współautorem ponad 160 publikacji naukowych i członkiem towarzystw naukowych IEEE, IEEE CI Soc., IEEE SMC Soc., TPO (IAPR), DAGM.

dr hab. inż. Wojciech Szynkiewicz
W.Szynkiewicz@elka.pw.edu.pl
ORCID: 0000-0001-6348-1129

Adiunkt w Instytucie Automatyki i Informatyki Stosowanej na Wydziale Elektroniki i Technik Informacyjnych Politechniki Warszawskiej. Zajmuje się zagadnieniami sterowania i programowania robotów, autonomicznej nawigacji, planowania zadań i cyberbezpieczeństwa robotów. Autor i współautor ponad 90 publikacji w wymienionych obszarach badawczych.

mgr inż. Maciej Stefańczyk
maciej.stefanczyk@pw.edu.pl
ORCID: 0000-0001-9948-6319

Absolwent Wydziału Elektroniki i Technik Informacyjnych Politechniki Warszawskiej. W 2010 r. uzyskał tytuł inżyniera, w 2011 r. tytuł magistra inżyniera, oba z wyróżnieniem. W 2011 r. rozpoczął prace nad doktoratem dotyczącym zastosowania aktywnej wizji wraz z systemami opartymi na bazie wiedzy w systemie sterowania robotów. Od 2012 r. pracuje na Politechnice Warszawskiej, początkowo jedynie przy realizacji projektów, a obecnie na stanowisku asystenta. Główne zainteresowania naukowe obejmują zastosowanie informacji wizyjnej zarówno w robotyce, jak i systemach rozrywki komputerowej.

mgr inż. Wojciech Dudek
wojciech.dudek@pw.edu.pl
ORCID: 0000-0001-5326-1034

Jest doktorantem i asystentem naukowym w Politechnice Warszawskiej w Instytucie Automatyki i Informatyki Stosowanej. Pracował w międzynarodowych grantach badawczych, m.in. RAPP (7PR Komisji Europejskiej) i INCARE (Active and Assisted Living JP Komisji Europejskiej). Jego zainteresowania naukowe obejmują systemy sterowania usługowymi robotami mobilnymi, ich lokalizacje, nawigacje i harmonizacje ich zadań.

mgr inż. Maksym Figat
maksym_figat@pw.edu.pl
ORCID: 0000-0002-1898-0540

Absolwent Wydziału Matematyki i Nauk Informacyjnych. W 2012 r. uzyskał tytuł inżyniera, a w 2013 r. tytuł magistra (z wyróżnieniem) na specjalizacji „Projektowanie systemów CAD/CAM". Jest doktorantem na Wydziale Elektroniki i Technik Informacyjnych Politechniki Warszawskiej (PW). W PW pracuje od 2014 r. w projektach badawczych krajowych i międzynarodowych, a od 2018 r. na stanowisku asystenta badawczo-dydaktycznego. Jego zainteresowania badawcze koncentrują się na zagadnieniach związanych z językami dziedzinowymi, specyfikacja systemów robotycznych, automatyczna generacja kodu sterowników robotów na podstawie formalnej specyfikacji.

mgr inż. Maciej Węgierek
wegierek.maciej@gmail.com
ORCID: 0000-0003-0779-255X

Jest doktorantem w Politechnice Warszawskiej w Instytucie Automatyki i Informatyki Stosowanej. Pracował w międzynarodowym grancie badawczym INCARE (Active and Assisted Living JP Komisji Europejskiej). Jego zainteresowania naukowe obejmują specyfikacje wymagań, struktury oraz zasady działania sterowników robotów.

mgr inż. Dawid Seredyński
dawid.seredynski@pw.edu.pl
ORCID: 0000-0003-2528-6335

Jest doktorantem na Wydziale Elektroniki i Technik Informacyjnych (WEiTI) Politechniki Warszawskiej (PW). W PW pracuje od 2013 r. w projektach badawczych krajowych i międzynarodowych, a od 2018 r. na stanowisku asystenta naukowo-dydaktycznego. Jego zainteresowania badawcze obejmują zagadnienia związane z planowaniem i realizacja złożonych zadań przez roboty usługowe.

prof. dr hab. inż. Cezary Zieliński
c.zielinski@ia.pw.edu.pl
ORCID: 0000-0001-7604-8834

Jest profesorem zwyczajnym na Wydziale Elektroniki i Technik Informacyjnych (WEiTI) Politechniki Warszawskiej (PW). W PW pracuje od 1985 r., a od 2008 r. również w Przemysłowym Instytucie Automatyki i Pomiarów PIAP. W PW sprawował funkcje: prodziekana ds. nauki i współpracy międzynarodowej WEiTI (2002–2005), zastępcy dyrektora ds. naukowych Instytutu Automatyki i Informatyki Stosowanej (IAiIS) (2005–2008), dyrektora IAIS (2008–2016) oraz prodziekana ds. ogólnych WEiTI (2016–). Od 1996 r. jest kierownikiem Zespołu Robotyki w IAiIS. Od 2007 r. jest członkiem Komitetu Automatyki i Robotyki Polskiej Akademii Nauk. Jego zainteresowania badawcze koncentrują się na zagadnieniach związanych z programowaniem i sterowaniem robotów.