Pracownia problemowa magisterska (PPMGR)

Autor pracy dyplomowej: **Michał Piotrak**

Opiekun pracy dyplomowej: **mgr inż. Krzysztof Chabko**

Temat pracy dyplomowej:

Realizacja systemu do rozpoznawania gestów języka migowego

Wprowadzenie

Głównym celem każdego systemu do rozpoznawania gestów danego języka migowego jest zapewnienie jak najwyższej jakości jego tłumaczenia na formę pisemną albo tekstową (lub odwrotnie) bez udziału ludzkiego tłumacza. Wraz z rozwojem nauk dotyczących m.in. komputerowego przetwarzania obrazu, uczenia maszynowego, czy przetwarzania języka naturalnego, mamy okazję zaobserwować na rynku coraz więcej interesujących rozwiązań danego problemu, które spełniają powyższy cel w zadowalającym stopniu. Ponadto, tego rodzaju system wyposażony również w mechanizm rozpoznawania mowy, może zapewnić interaktywną komunikację między osobami słyszącymi a niedosłyszącymi.

Dana praca zawiera ogólny przegląd projektów systemów do tłumaczenia określonego języka migowego. Pozwolił mi on wyodrębnić wstępne wymagania, które będę się starał zrealizować w ramach projektu pracy dyplomowej. Warto wspomnieć, że będzie on korzystał z doświadczeń, wynikających z pracy inżynierskiej, która w głównej mierze polegała na wykorzystaniu możliwości kontrolera Kinect.

Opis istniejących rozwiązań

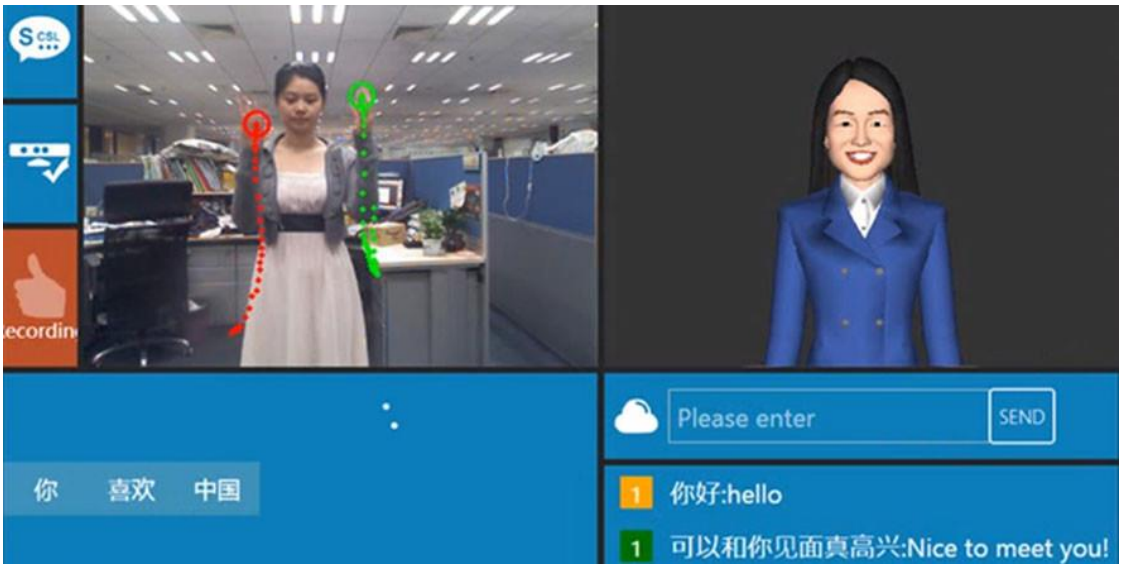
Większość współczesnych rozwiązań problemu tłumaczenia języka migowego wykorzystuje różnego rodzaju kamery, służące do rejestracji obrazu, który następnie jest przetwarzany, w celu identyfikacji jakiegoś gestu, który jest zarejestrowany w danym systemie jako element języka migowego. Jednak, pierwsze tego rodzaju systemy opierały swoją funkcjonalność na zastosowaniu tzw. sztucznej dłoni. Przykładem takiego podejścia może być projekt **SignAloud,** który wykorzystuje parę rękawiczek, wyposażonych w specjalne czujniki, które rejestrują pozycję oraz ruch dłoni. Następnie, te dane są przesyłane za pośrednictwem technologii Bluetooth do komputera centralnego, który realizuje proces ich przetwarzania. Polega on na dopasowaniu otrzymanych danych do jakiegoś gestu, z którym związane jest wcześniej zdefiniowane słowo lub fraza. Po tym działaniu, otrzymane słowo lub ich ciąg przekazywane

jest do syntezatora mowy, którego zastosowanie zapewnia możliwość zrozumienia przez osobę słyszącą treści, którą chce przekazać osoba niedosłysząca. Niestety, rękawice nie zostały wyposażone w mechanizm rejestracji głosu, przez co nie gwarantują dwukierunkowej komunikacji między osobą słyszącą a głuchą.

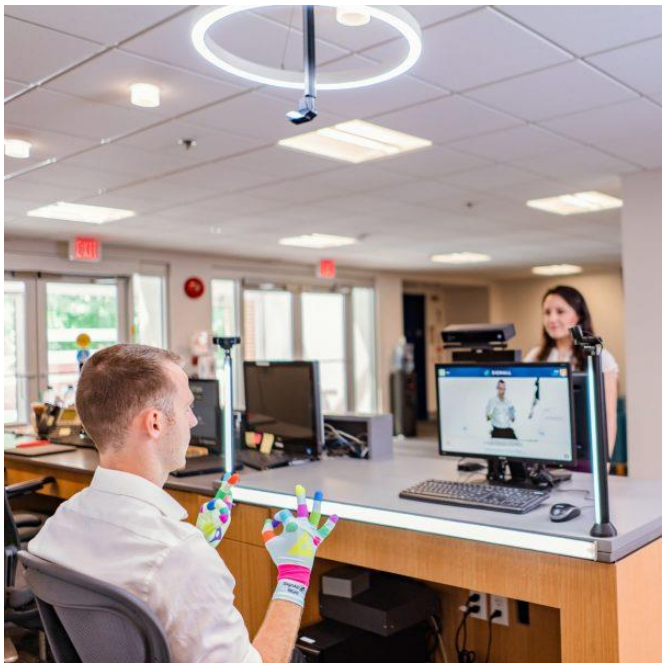
Wzajemną komunikację za to zapewnia system, będący wynikiem współpracy naukowców z Chińskiej Akademii Nauk i specjalistów surdopedagogiki z Uniwersytetu w Pekinie w ramach zespołu należącego do oddziału Microsoft Research firmy Microsoft - **Kinect Sign Language Translator**. Jak sama nazwa projektu wskazuje, wykorzystuje on kontroler Kinect do rejestracji obrazu. System opiera swoje działanie na dwóch trybach:

- tryb tłumacza (*translator mode*) - polega na przełożeniu danego gestu na odpowiadające mu słowo pisane oraz odwrotnie. Potrafi wychwycić tor ruchu dłoni osoby posługującej się językiem migowym przy pomocy technik rozpoznawania wzorców i obrazów oraz uczenia maszynowego.
- tryb komunikacji (*communication mode*) - potrafi tłumaczyć całe zdania oraz wyposażony jest w specjalny avatar 3D, dzięki któremu jest możliwa wzajemna komunikacja między osobą słyszącą a niedosłyszącą. Cały proces wygląda tak, że w przypadku, gdy osoba słysząca chce przekazać wiadomość, to system rejestruje jej głos, przetwarza go i zostaje wykorzystany wcześniej wspomniany avatar 3D, który przedstawia odpowiadające treści wiadomości gesty, zrozumiałe dla osoby głuchej. W odwrotnym kierunku, przetwarzany jest obraz, zawierający gest wykonany przez osobę niedosłyszącą, w wyniku czego system prezentuje wiadomość tekstową, którą może odczytać jej odbiorca.

System pierwotnie został uruchomiony dla chińskiego języka migowego, ale aktualnie trwają prace nad rozwojem projektu dla odmiany amerykańskiej. Warto również wspomnieć, że dane urządzenie jest dalej prototypem, przez co nie gwarantuje wysokiej jakości tłumaczenia.



Za system charakteryzujący się większą precyzją należy uznać projekt **SignAll** - rozwiązanie dostarczane przez węgierską firmę Dolphio Technologies. System również może działać w



oparciu o kontroler Kinect czy po prostu inną kamerę, wyposażoną w czujnik głębi. Jednak, największą różnicą w porównaniu do mechanizmu udostępnionego w ramach projektu Kinect Sign Language Translator są specjalne rękawice, które musi założyć osoba, posługująca się językiem migowym. Dzięki ich zastosowaniu, osiągnięto większą precyzję przy tłumaczeniu gestów, które polegają na wykorzystaniu palców u dłoni. Ogólnie, cały system opiera się na różnych technikach rozpoznawania obrazów oraz przetwarzania języka naturalnego.

Określenie wymagań

Na podstawie dokonanego przeglądu istniejących rozwiązań oraz własnych doświadczeń związanych z realizacją pracy inżynierskiej opracowałem poniższą specyfikację wymagań, przedstawiającą funkcjonalność oraz sposób działania systemu, który będę się starał zaprojektować w ramach pracy magisterskiej.

Wymagania funkcjonalne:

1. System będzie rozpoznawał zdefiniowane gesty wzorowane na amerykańskim języku migowym (*American Sign Language*), czyli tym samym opierał się na amerykańskiej odmianie języka angielskiego.
2. System będzie działał w oparciu o zaprojektowany słownik, w którym danemu wyrazowi będzie odpowiadał dany gest. Ze zdefiniowanych słów zostaną także utworzone różne frazy, które będą związane z pewną kombinacją gestów.
3. System będzie przekładał dany gest na wiadomość w formie tekstowej, która zostanie zaprezentowana użytkownikowi.
4. System będzie umożliwiał przetwarzanie mowy użytkownika na wiadomość tekstową, która zostanie jemu zaprezentowana. W ten sposób zostanie zagwarantowana wzajemna komunikacja osoby słyszącej z osobą głuchą, przy założeniu, że użytkownik niedosłyszący posiadł umiejętność czytania. Ta funkcjonalność zostanie zrealizowana poprzez zintegrowanie z zewnętrznym API, udostępnionym w ramach katalogu usług Cognitive Services firmy Microsoft - Speech To Text API.

Wymagania niefunkcjonalne:

1. System będzie mógł współpracować z kontrolerem Kinect 2.0. Będzie wykorzystywał dane, udostępnione przez czujnik głębi.

2. Identyfikacja gestów będzie ograniczona wyłącznie do śledzenia rąk czy dłoni. System nie będzie brał pod uwagę m.in. ruch warg ust, który jest istotny przy niektórych gestach, wchodzących w skład języka migowego.
3. System będzie projektowany z myślą o wykonywaniu tłumaczenia w czasie rzeczywistym.
4. System zostanie zaimplementowany przy użyciu języka C#.

Bibliografia

1. http://www.dani89mc.com/thesis.html
2. http://globalaccessibilitynews.com/2017/01/27/students-invented-gloves-that-convert-sign-language-into-speech/
3. https://www.microsoft.com/en-us/research/blog/kinect-sign-language-translator-part-1/
4. http://www.signall.us/product_info/
5. https://azure.microsoft.com/en-us/services/cognitive-services/speech-to-text/
6. https://en.wikipedia.org/wiki/Machine_translation_of_sign_languages

Pracownia dyplomowa magisterska

Autor pracy dyplomowej: **Michał Piotrak**

Opiekun pracy dyplomowej: **mgr inż. Krzysztof Chabko**

Temat pracy dyplomowej:

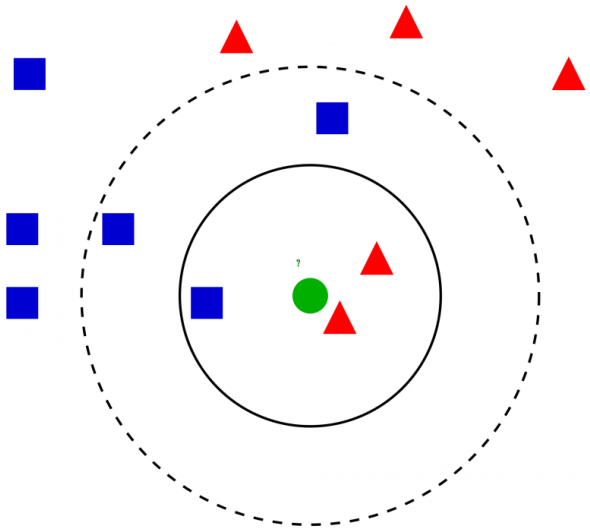
Realizacja systemu do rozpoznawania gestów języka migowego

Wprowadzenie

Aktualny etap prac nad projektem realizowanym w ramach pracy magisterskiej skupił się przede wszystkim nad problemem klasyfikacji wykonywanego gestu przez użytkownika jako przykładu konkretnego gestu języka migowego, który został zdefiniowany w systemie. Podczas poszukiwania metod do rozwiązania przedstawionego problemu, skupiłem się przede wszystkim na algorytmach uczenia maszynowego z nadzorem. W praktyce umożliwiają one dostarczenie wytrenowanego modelu na danych etykietowanych, który następnie można wykorzystać do różnych zadań na danych nieetykietowanych. Jednym z takich zadań jest właśnie proces klasyfikacji, polegający na przypisaniu dostarczonego na wejściu modelu (klasyfikatora) przykładu do danej klasy, która jest wyodrębniona w modelu.
Oczywiście, w projektowanym klasyfikatorze gestów języka migowego, na jedną klasę będą przypadać przykłady jednego konkretnego gestu. W wyborze metody klasyfikacji starałem się mieć na uwadze to, że cały proces rozpoznawania gestu będzie odbywał się w czasie rzeczywistym, dlatego realizacja procesu klasyfikacji powinna być jak w najmniejszym stopniu złożona, ale zarazem skuteczna. Z tego względu, zdecydowałem się na wykorzystanie jednej z podstawowych metod uczenia maszynowego z nadzorem, czyli algorytmu k najbliższych sąsiadów. Do określania podobieństwa pomiędzy poszczególnymi przykładami zostanie użyta dynamiczna transformata czasowa. Obydwa wspomniane mechanizmy, wraz z ich potencjalnymi usprawnieniami, zostały opisane w kolejnych rozdziałach raportu.

Algorytm k najbliższych sąsiadów

Ten sposób klasyfikacji wykorzystuje w sposób bezpośredni przykłady ze zbioru trenującego do określenia klasy obiektu testującego. Jego oznaczenie jako egzemplarza danej klasy polega na sprawdzeniu przynależności klasowej k najbliższych próbek trenujących i wybraniu klasy, która jest reprezentowana przez największą liczbę przykładów. Najlepiej będzie to zademonstrować na poniższym przykładzie:



Należy przyjąć, że zielonym kółkiem została przestawiona lokalizacja przykładu, którego klasyfikator ma przydzielić do konkretnej klasy. Niebieskie kwadraty i czerwone trójkąty odzwierciedlają próbki ze zbioru trenującego. Okrąg z ciągłym obwodem ilustruje sytuację, gdy parametr algorytmu k = 3. Wtedy rozpatrywane są tylko 3 najbliższe obiekty do zielonego kółka i w takiej sytuacji zostanie mu przypisana klasa reprezentowana przez czerwony trójkąt. Postępując analogicznie, okrąg z przerywanym obwodem przedstawia przypadek, gdy k = 5. W tym przypadku, przykład testujący zostaje skojarzony z klasą równoważną dla niebieskiego kwadratu.

Przy implementacji algorytmu, bardzo możliwa będzie konieczność rozwiązania pewnych problemów skutkujących na efektywności działania programu:

- duża złożoność pamięciowa i obliczeniowa – w celu jej ograniczenia prawdopodobnie będzie trzeba wykorzystać algorytmy kondensacji zbioru uczącego, które usuwają próbki niemające wpływu na przebieg granicy decyzyjnej,
- nadmierne dopasowanie do zbioru trenującego – przy dużej liczbie cech branych pod uwagę w procesie klasyfikacji, istnieje ryzyko zbyt dużego dopasowania się klasyfikatora do danych uczących, przez co może popełniać błędy w klasyfikacji próbek testujących. Z tego powodu warto zastosować analizę głównych składowych (ang. Principal Component Analysis – w skrócie metoda PCA) albo inną pokrewną metodę, aby pozbyć się tych cech, które nie decydują o wyniku klasyfikacji.

Do przygotowania klasyfikatora najprawdopodobniej zostanie wykorzystany model udostępniony przez bibliotekę OpenCV lub Accord.NET.

Dynamiczna transformata czasowa

Jak wcześniej zostało wspomniane, do porównywania sekwencji czasowych, reprezentujących konkretne gesty zostanie zastosowana dynamiczna transformata czasowa (ang. Dynamic Time Warping – w skrócie DTW). Jest to algorytm wykorzystywany zarówno w procesie rozpoznawania mowy, jak i w systemach wideo oraz grafice komputerowej. Przyjmując, że mamy zdefiniowane dwa sygnały czasowe, pozwala znajdować w nich podobne kształty o różnych fazach. Aby objaśnić to w sposób bardziej obrazowy, opisana właściwość jest

wykorzystywana np. w sytuacji, gdy trzeba rozpoznać dane słowo w dwóch wypowiedziach, które zostały wykonane z różną szybkością.
Cały algorytm rozpoczyna się od wyliczenia tzw. macierzy odległości lokalnych, która reprezentuje odległości pomiędzy wszystkimi parami elementów zdefiniowanych szeregów. Do mierzenia tej odległości można stosować różne metryki – np. euklidesową czy Manhattan. Po zbudowaniu danej macierzy następuje proces znalezienia ścieżki dopasowania przebiegającej przez obszary o małym koszcie – oznaczającym, że w danym miejscu porównywane przebiegi są podobne. Ogólnie rzecz biorąc, określenie kosztu bierze się stąd, że wspomniana wcześniej ścieżka dopasowania jest wyznacza przy pomocy Programowania Dynamicznego. Po jej znalezieniu, mamy także obliczony jej koszt wyznaczenia. Jeśli wartość kosztu jest mała, to znaczy, że badane szeregi są do siebie podobne.
Odnośnie implementacji algorytmu, to aktualnie są rozważane przeze mnie dwie możliwości:

- własna implementacja,
- wykorzystanie gotowej metody, udostępnionej w ramach biblioteki Accord.NET.

Aktualizacja wymagań projektu

Większość wymagań odnośnie projektu pozostaje bez zmian w porównaniu z raportem z Pracowni Problemowej Magisterskiej (PPMGR). Pojawiły się jednak pewne zmiany, związane z większą wiedzą w zakresie problemu klasyfikacji.

Wymagania funkcjonalne:

1. System będzie rozpoznawał zdefiniowane gesty wzorowane na amerykańskim języku migowym (*American Sign Language*), czyli tym samym opierał się na amerykańskiej odmianie języka angielskiego.
2. System będzie działał w oparciu o zaprojektowany słownik, w którym danej frazie będzie odpowiadał dany gest.
3. System będzie przekładał dany gest na konkretną frazę, która zostanie zaprezentowana użytkownikowi.
4. System będzie umożliwiał przetwarzanie mowy użytkownika na wiadomość tekstową, która zostanie jemu zaprezentowana. W ten sposób zostanie zagwarantowana wzajemna komunikacja osoby słyszącej z osobą głuchą, przy założeniu, że użytkownik niedosłyszący posiadł umiejętność czytania. Ta funkcjonalność zostanie zrealizowana poprzez zintegrowanie z zewnętrznym API, udostępnionym w ramach katalogu usług Cognitive Services firmy Microsoft - Speech To Text API.
5. System udostępni funkcjonalność dodawania nowego gestu wraz z równoważną jemu frazą do wcześniej zdefiniowanego słownika.

Wymagania niefunkcjonalne:

1. System będzie mógł współpracować z kontrolerem Kinect 2.0. Będzie wykorzystywał dane, udostępnione przez czujnik głębi.

2. Identyfikacja gestów będzie ograniczona wyłącznie do śledzenia rąk czy dłoni. System nie będzie brał pod uwagę m.in. ruch warg ust, który jest istotny przy niektórych gestach, wchodzących w skład języka migowego.
3. System będzie projektowany z myślą o wykonywaniu tłumaczenia w czasie rzeczywistym.
4. System zostanie zaimplementowany głównie przy użyciu języka C#. Ewentualnie zostaną użyte inne narzędzia do procesu tworzenia własnego klasyfikatora.

Plan realizacji projektu

Poniżej postanowiłem umieścić plan swoich najbliższych działań, na których się skupię w projektowaniu omawianego systemu:

1. Realizacja własnych algorytmów niezbędnych do procesu przetwarzania obrazu, który będzie miał na celu przygotowanie danej klatki do etapu klasyfikacji. Jest to krok konieczny, ponieważ przy pracy inżynierskiej korzystałem z metod udostępnionych przez zewnętrzną bibliotekę AForge.NET. Wiązało się to z kopiowaniem obrazu z obiektu dostarczonego przez środowisko UWP do obiektu, z którego korzystała metoda omawianej biblioteki. Była to operacja czasochłonna, dlatego lepiej byłoby jej uniknąć.
2. Przygotowanie środowiska do tworzenia słownika systemu, czyli zapisywania konkretnej pary gest – fraza. Dana funkcjonalność ma uprościć proces tworzenia zbioru trenującego dla klasyfikatora.
3. Implementacja metody dynamicznej transformaty czasowej oraz jej testy.
4. Przygotowanie klasyfikatora.

Bibliografia

1. http://www.dani89mc.com/thesis.html
2. https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/
3. https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761
4. http://www.mini.pw.edu.pl/~rafalkoj/www/?download=DTW.pdf

Przygotowanie pracy dyplomowej magisterskiej (PDYM)

Autor pracy dyplomowej: **Michał Piotrak**

Opiekun pracy dyplomowej: **mgr inż. Krzysztof Chabko**

Temat pracy dyplomowej:

Realizacja systemu do rozpoznawania gestów języka migowego

Wprowadzenie

Raport skupia się przede wszystkim na kwestii wyboru rodzaju gestów, które będą rozpoznawane przez opracowany system oraz na prezentacji koncepcji realizacji procesu wydobycia informacji o lokalizacji oraz kształcie dłoni, która potencjalnie występuje w przetwarzanym obrazie. Przygotowywane rozwiązanie powinno umożliwiać śledzenie 7 charakterystycznych punktów, na które przypadają 5 palców, środek dłoni oraz nadgarstek. Poprzez projektowanie kolejnego elementu systemu, wyciągnąłem należyte wnioski, które poskutkowały aktualizacją wymagań stawianych szykowanemu mechanizmowi. Postanowiłem również przemyśleć zagadnienie związane z weryfikacją mojego rozwiązania, dlatego zamieściłem w dany tekście rozdział odnoszący się do wstępnych planów badań działania systemu. Na pewno będą one dotyczyły sprawdzenia samego procesu ekstrakcji dłoni z obrazu, jak i jakości opracowanego klasyfikatora, który zgodnie z poprzednim raportem będzie opierał się na wykorzystaniu algorytmu k najbliższych sąsiadów oraz dynamicznej transformaty czasowej. Na końcu niniejszej pracy podsumowuje swoje ostatnie wykonane czynności w ramach projektu oraz wspominam o harmonogramie najbliższych prac.

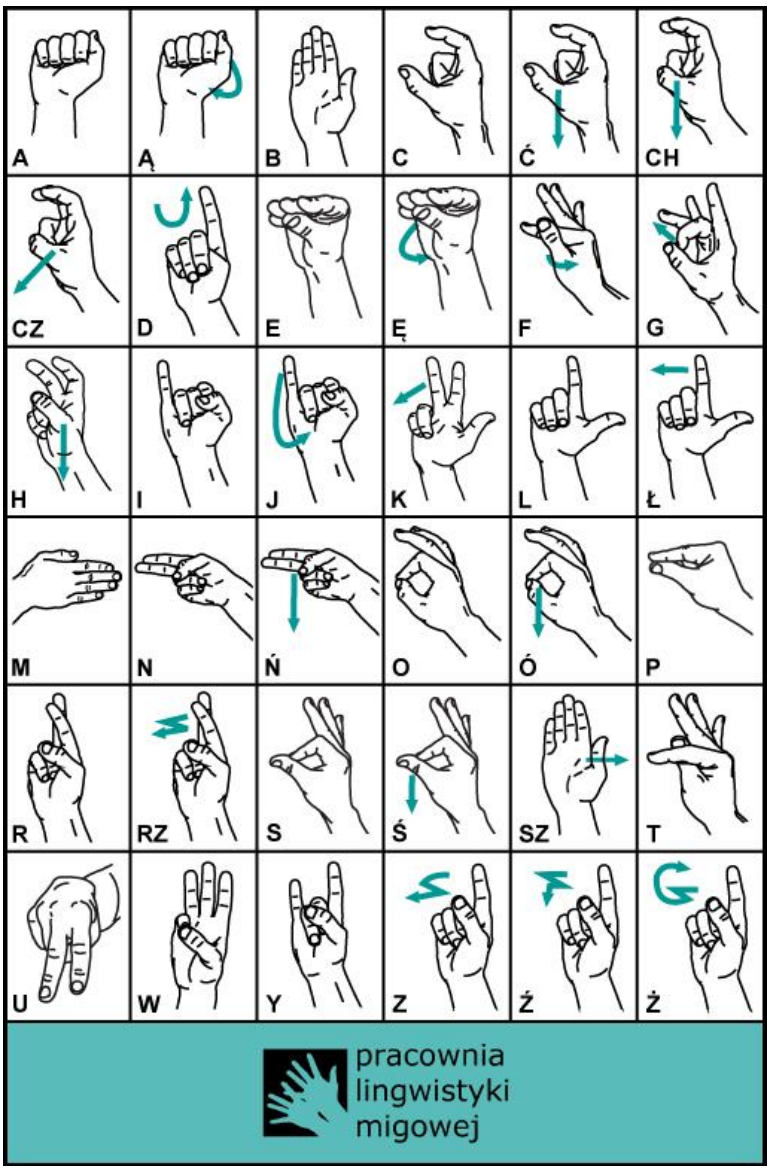
Podział znaków w języku migowym

Obowiązujący podział znaków zostanie zaprezentowany na przykładzie Polskiego Języka Migowego (PJM), na którym ostatecznie zdecydowałem się bazować. Gesty wchodzące w skład PJM dzielą się na znaki ideograficzne i znaki daktylograficzne. Te pierwsze należy traktować jako znaki pojęciowe, które oznaczają konkretne słowo lub zwrot. Przez ten fakt, stanowią one podstawę komunikacji dla osób posługujących się omawianym językiem migowym. Podczas wykonywania tego rodzaju gestów, ręka jest ustawiona w określonym miejscu względem ciała.

Znaki daktylograficzne pełnią raczej funkcje pomocnicze i uzupełniające. Daktylografia polega na porozumiewaniu się za pomocą odpowiednich układów palców jednej lub obydwu dłoni. Ten rodzaj gestów został użyty do określenia liter alfabetu, liczb, znaków interpunkcyjnych itp. Wykonując dane znaki, mamy możliwość przekazywania imion, nazw, skrótowców czy po prostu wyrazów, dla których nie ma znaku ideograficznego, poprzez literowanie. Warto także zaznaczyć, że komunikacja opierająca się wyłącznie na znakach ideograficznych nie jest do końca poprawna pod względem gramatycznym, w szczególności dla takich języków jak język polski. Precyzując, jest po prostu pozbawiona zakończeń fleksyjnych. Z tego powodu, jeśli chcemy zachować pełną zgodność przekazu migowego z ustnym należałoby używać znaki migowe w szyku gramatycznym języka ojczystego, dodając

za pomocą alfabetu palcowego końcówki fleksyjne. Oczywiście, dana forma komunikacji jest zwyczajnie wolniejsza od podstawowej, dlatego raczej wykorzystuje się ją w celach dydaktycznych, a nie użytkowych.

Rozpoznawanie gestów ideograficznych wiązałoby się z realizacją mechanizmu wydobycia uproszczonego szkieletu użytkownika, dzięki któremu można byłoby ustalić położenie ręki względem ciała. Niestety, wykonanie tej funkcjonalności w sposób satysfakcjonujący wiąże się z implementacją złożonych algorytmów, dlatego, żeby zwiększyć prawdopodobieństwo powodzenia projektu, postanowiłem skupić się na wykrywaniu znaków daktylograficznych, czyli śledzeniu wyłącznie ruchu dłoni i jej palców. W takiej sytuacji, naturalnym wyborem wydaje się dostarczenie systemu, który będzie potrafił rozpoznawać znaki utożsamiane z określonymi literami. Za źródło danych gestów posłuży poniższa tablica, która została udostępniona przez Pracownię Lingwistyki Migowej Wydziału Polonistyki Uniwersytetu Warszawskiego:



Jak wynika z informacji zawartych na powyższym obrazku, niektóre gesty są odmianami innych, tylko w wersji dynamicznej, która polega na nie tylko odpowiednim ułożeniu palców, ale także wykonaniu właściwego ruchu dłonią (np. litery N i Ń). Z tego powodu mamy nieformalny podział tych znaków na statyczne i dynamiczne. Oczywiście, moje rozwiązanie powinno wykrywać obydwa rodzaje gestów.

Realizacja wykrywania dłoni

Prezentowany algorytm detekcji dłoni w scenie ma dostarczać uproszczony model szkieletu, w którym będzie oznaczone 7 punktów charakterystycznych – 5 palców, środek dłoni oraz nadgarstek. Opracowany schemat działań ma na razie charakter wstępnych planów, dlatego możliwe jest, że będzie on modyfikowany w przyszłości, wraz z postępem prac. Warto także odnotować, że dany algorytm będzie wykorzystywał wyłącznie obraz pochodzący z kamery głębokości. Poniżej zamieściłem ogólną listę kolejnych kroków przygotowanego mechanizmu wraz z krótkimi wyjaśnieniami:

1. Konwersja mapy głębokości z formatu 11 bitowego na 8 bitowy. Po prostu pierwszy format nie jest wydajny do dalszej pracy oraz także niemożliwy do wizualizacji.
2. Usunięcie zbędnych szumów występujących w obrazie oraz jego wygładzenie. Zostanie to zrealizowane przy użyciu filtru Gaussa oraz medianowego.
3. Segmentacja obszaru dłoni. Wykonanie tej operacji najprawdopodobniej niestety będzie wymagała od strony użytkownika ułożenia dłoni w środku sceny oraz w płaszczyźnie mniej więcej równoległej do czujnika głębokości. Dzięki temu będzie można oszacować poziom głębokości, w którym znajduje się dłoń oraz następnie wydzielić piksele ulokowane w nim albo w niedalekim sąsiedztwie. Otrzymane piksele zostaną pokolorowane na kolor szary, a nie biały, ze względu na ułatwienie czynności wykonywanych w następnych krokach.
4. Zastosowanie operatora zamknięcia, w celu pozbycia się różnych „dziur" pojawiających się w wydzielonym obszarze dłoni.
5. W wyniku przeprowadzenia segmentacji mogliśmy dostać również różne inne elementy nie należące do dłoni, a nawet z nią nie połączone. Należy je wyeliminować i można tego dokonać poprzez odpowiednie wykorzystanie algorytmu Flood Fill do pokolorowania pikseli należących do ręki na kolor biały. Jeśli uzyskamy taki efekt, wtedy można przeprowadzić zwykłą segmentację, która wyodrębni jedynie obszar należący do dłoni
6. Użycie algorytmu Canny'ego do detekcji krawędzi i wyznaczenia konturu dłoni.
7. Posiadając informacje o lokalizacji i kształcie konturu, można określić tzw. Region Of Interest (w skrócie ROI), który byłby tożsamy z obszarem naszej dłoni. Dany ROI mógłby mieć kształt zaokrąglonego prostokąta czy nawet elipsy, która informowałaby o lokalizacji naszej dłoni. W takiej sytuacji, łatwo analizowałoby się przypadki, gdy dłoń czy jej fragment znalazł się poza kadrem kamery.
8. Kopiowanie wcześniej otrzymanego konturu na nową bitmapę i ewentualne zastosowanie filtru Gaussa w celu jego wygładzenia.

9. Użycie operatora morfologicznego Hit-And-Miss, aby uzyskać szczuplejsze niektóre fragmenty dłoni, jak np. palce. Ta operacja jest zapoczątkowaniem procesu ekstrakcji szkieletu dłoni z otrzymanego obiektu.
10. Wykorzystanie metody Euclidean Distance Transform, która dostarcza specjalną macierz, zawierającą wyliczone odległości poszczególnych pikseli znajdujących się w obszarze dłoni do jej konturu. Największe wartości w tej macierzy wyznaczają przebieg uproszczonego szkieletu.
11. Zastosowanie operatora Laplace'a, aby wydobyć wyżej opisany szkielet.
12. Dokonanie normalizacji otrzymanego szkieletu, aby była możliwa jego wizualizacja w obrazie binarnym.
13. Posłużenie się operatorem dylacji albo bardziej zaawansowaną metodą Probalitic Hough Line Transform, w celu połączenia niektórych linii szkieletu, które mogły być od siebie odizolowane.
14. Wykrycie „kątów" występujących w szkielecie za pomocą algorytmu detekcji Harrisa. Wspomniane „kąty" są równoważne z punktami charakterystycznymi. Oczywiście, dana metoda może nanieść trochę szumów albo niedokładności na obraz, dlatego prawdopodobne jest zastosowanie po niej jakichś dodatkowych operacji.
15. Wyznaczanie otoczki wypukłej dla każdego wcześniej otrzymanego punktu charakterystycznego. Dane działanie pozwoli połączyć niektóre punkty w większy obiekt, który będzie odbierany jako 1 punkt charakterystyczny. Ten zabieg powinien być szczególnie ważny dla obszaru środka dłoni, ponieważ w jego okolicach najprawdopodobniej może dojść do różnych zakłóceń.
16. Dwupoziomowa filtracja wygenerowanych otoczek, która powinna zakończyć się wyznaczeniem 7 punktów charakterystycznych, czyli 5 palców, środka dłoni oraz nadgarstka.
17. Na końcu, danym punktom powinna zostać przyznana odpowiednia etykieta, która by je bezpośrednio identyfikowała. Ten etap wiąże się z analizą lokalizacji tych punktów względem siebie oraz jak zmienia się ich położenie wraz z ruchem ręki.

Można także dodać, że przedstawiony algorytm, przy pewnych modyfikacjach, można byłoby użyć do uzyskania szkieletu całego ciała. Jednak, na początku lepiej skupić się na prostszym przypadku dotyczącym dłoni, w celu upewnienia się o słuszności swoich pomysłów. Opisane rozwiązanie niesie ze sobą pewne ograniczenia. Zostaną one ujęte w podanych wymaganiach systemowych oraz założeniach projektowych.

Aktualizacja wymagań i założeń projektu

Ze względu na pewne wnioski wyciągnięte podczas opracowywania projektu mechanizmu śledzenia ruchu dłoni, lista wymagań odnoszących się do systemu zostanie w niektórych miejscach zmodyfikowana. Postanowiłem także wyodrębnić sekcję związaną z założeniami projektowymi, aby nie były one utożsamiane z terminem wymagania.

Wymagania funkcjonalne:

1. System zagwarantuje możliwość detekcji minimum jednej dłoni użytkownika. Jej położenie będzie definiowane przez 7 punktów charakterystycznych, na które przypadają wszystkie palce, środek dłoni oraz nadgarstek. Prezentacja tych punktów będzie polegała na demonstracji uproszczonego szkieletu dłoni.
2. System będzie działał w oparciu o zaprojektowany słownik, w którym danemu gestowi będzie odpowiadała konkretna fraza.
3. System będzie przekładał gest na frazę, która zostanie zaprezentowana użytkownikowi.
4. System udostępni funkcjonalność dodawania nowego gestu wraz z równoważną jemu frazą do wcześniej zdefiniowanego słownika.

Wymagania niefunkcjonalne:

1. System będzie mógł współpracować z kontrolerem Kinect 2.0 oraz konsolą Xbox One.
2. System będzie projektowany z myślą o wykonywaniu tłumaczenia w czasie rzeczywistym.
3. System będzie wykrywał tylko gesty oparte na ruchu dłoni oraz jej palców, czyli znaki daktylograficzne.
4. System będzie mógł rozpoznawać zdefiniowane gesty tylko wtedy, gdy użytkownik będzie znajdować się od kamery w odległości 0,5 – 2 m. Ten zabieg ma na celu umożliwić śledzenie ruchów palców, które przy większej odległości mogą nie być po prostu widoczne.

Założenia projektowe:

1. System powinien rozpoznawać wszystkie znaki znajdujące się w tzw. Polskim Alfabecie Palcowym. Dotyczy to zarówno gestów statycznych, jak i dynamicznych. Ewentualnie, mniejszym priorytetem zostaną wyróżnione gesty, w których palce są w płaszczyźnie pionowej do kamery (np. litera E). Po prostu opracowany schemat detekcji gestów może nie radzić sobie z takimi przypadkami.
2. System prawdopodobnie będzie wymagał od użytkownika mniej więcej dokładnego ulokowania jego dłoni, np. w środku sceny. Ten zabieg będzie miał na celu uprościć proces segmentacji, który miałby wyodrębnić dłoń z reszty obrazu.
3. Dłoń użytkownika nie powinna znajdować się w bliskiej odległości od tułowia, czy innego elementu ciała. Te ograniczenie związane jest z faktem, że wykrywanie dłoni będzie realizowane wyłącznie poprzez przetwarzanie danych z czujnika głębi. Zrezygnowałem z równoległej obsługi danych z kamery RGB do rozpoznawania obszaru o barwie zbliżonej do koloru ludzkiej skóry, ponieważ w pewien sposób ogranicza to rozwiązanie do sytuacji, kiedy mamy odpowiednie oświetlenie, użytkownik nie ma na sobie rękawiczek itd.

System zostanie zaimplementowany głównie przy użyciu języka C#, w środowisku UWP. Ewentualnie zostaną użyte inne narzędzia do procesu tworzenia własnego klasyfikatora.

Proponowane badania działania systemu

Jak było wspominane na samym początku raportu, ten rozdział dotyczy wstępnych planów badań weryfikacyjnych jakość działania mechanizmu śledzenia ruchu ręki oraz przygotowanego klasyfikatora gestów. W przypadku tej pierwszej funkcjonalności, na pewno będą sprawdzane różne kombinacje związane z ułożeniem palców, które niekiedy będą wymagały „schowania" jakiegoś palca czy wyeksponowania wszystkich. Bez wątpienia trzeba będzie przeprowadzić testy polegające na obrocie dłoni czy usytuowaniu jej w innej płaszczyźnie niż mniej więcej równoległej do kamery. Poddana badaniu zostanie także skuteczność detekcji gestów w różnych odległościach.

Co do klasyfikatora, chciałbym dokonać porównania jego skuteczności dla przypadków, gdy proces jego nauki będzie polegał na pokazywaniu określonych gestów przez jedną osobą lub przez większą liczbę osób. Z pewnością, dobrze byłoby zagwarantować również, żeby przykłady testowe były wykonywane przez innych użytkowników niż tych, którzy brali udział w trenowaniu klasyfikatora. Na pewno znajdzie się także miejsce na zestawienie skuteczności rozpoznawania znaków statycznych a dynamicznych.

Wykonane prace w tym semestrze

W ostatnim okresie zajmowałem się przede wszystkim implementacją różnych algorytmów z zakresu grafiki komputerowej, których użyłem do stworzenia własnego detektora ruchu w trakcie realizacji pracy inżynierskiej. Jednak wtedy opierałem się na stosowaniu gotowych już rozwiązań udostępnionych przez bibliotekę AForge.NET. Niestety, to podejście okazało się kosztowne, jeśli chodzi o wydajność opracowanego mechanizmu, ponieważ wymuszało kopiowanie całej bitmapy otrzymanej od kontrolera Kinect do formatu akceptowalnego przez daną bibliotekę. Z tego względu postanowiłem sam zaimplementować użyte operatory, filtry czy algorytmy, aby uzyskać po prostu lepsze rezultaty. Przy okazji, chciałem wykorzystać opisywany detektor do aktualnie szykowanego systemu. Najprawdopodobniej nie zostanie on w całości zastosowany, lecz jego składowe pewnie tak. Precyzując, udało mi się przygotować poniższe mechanizmy:

- zmienianie rozmiaru obrazu przy pomocy interpolacji dwuliniowej,
- konwersja do obrazu w odcieniach szarości,
- pikselizacja obrazu wejściowego,
- filtr do mieszania dwóch różnych obrazów wejściowych w celu uzyskania jednego na wyjściu,
- wyliczanie różnicy dwóch obrazów,
- progowanie,
- operator dylatacji,
- algorytm Grahama do wyszukiwania otoczki wypukłej dla podanego zbioru punktów.

Warto także dodać, że powyższe mechanizmy zostały poddane testom, polegającym na porównywaniu wyniku ich działania z rezultatami otrzymanymi poprzez wywołanie ich odpowiedników z biblioteki AForge.NET.

Aktualizacja planu realizacji projektu

Poniżej postanowiłem umieścić plan swoich najbliższych działań dotyczących realizacji projektu:

1. Implementacja mechanizmu śledzenia dłoni oraz jego testy.
2. Przygotowanie środowiska do tworzenia słownika systemu, czyli zapisywania konkretnej pary gest – fraza. Dana funkcjonalność ma uprościć proces tworzenia zbioru trenującego dla klasyfikatora.
3. Implementacja metody dynamicznej transformaty czasowej oraz jej testy.
4. Przygotowanie klasyfikatora.
5. Przeprowadzenie różnych badań związanych z działaniem klasyfikatora.

Bibliografia

1. http://media.statsoft.nazwa.pl/_old_dnn/downloads/marnik.pdf
2. http://poradnik-logopedyczny.pl/komunikacja-alternatywna/komunikacja-alternatywna/249/daktylografia.html
3. http://www.slownikpjm.uw.edu.pl/page/opjm
4. http://www.dani89mc.com/thesis.html
5. https://hub.packtpub.com/hand-gesture-recognition-using-kinect-depth-sensor/
6. https://pdfs.semanticscholar.org/19ca/f2ef4e2509b734679b9c5330e8d4d62ab96b.pdf