11.05.2021

Temat pracy magisterskiej: **System rozpoznawania gestów języka migowego**

Autor: **Michał Piotrak**

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

Wstęp

Pierwsza część raportu skupi się na krótkim wyjaśnieniu problemu z występującymi szumami na mapie głębokości pochodzącej z sensora Kinect, które uniemożliwiły wykonanie w sposób efektywny ekstrakcji dłoni z tego rodzaju obrazu. Następnie, przedstawię nową koncepcję wykonywanego projektu, który opiera się na wykorzystaniu „danych szkieletowych" pochodzących z kontrolera Kinect. Ta decyzja projektowa wpłynęła również na to, że docelową platformą przygotowywanego systemu będzie komputer PC z systemem operacyjnym Windows 10.

W wielu akapitach tego raportu będę odnosił się do wcześniejszych koncepcji zawartych w załączonym pliku *Piotrak_PDYM.pdf*, który zawiera wszystkie wcześniejsze ustalenia dotyczące projektu. Według bibliografii, dany dokument został oznaczony symbolem [1] i w miejscach powoływania się na niego jest wspominana również strona, na której znajduje się fragment, do którego się w danym miejscu odwołuje.

Wyjaśnienie problemu z szumami

Zgodnie z wcześniejszym założeniem, opracowywany system miał rozpoznawać znaki daktylograficzne odpowiadające literom z polskiego alfabetu [1] (strona 10). Do wykonywania tego rodzaju gestów potrzebny jest mechanizm śledzący ruch jednej dłoni oraz jej palców. Ze względu na to, że na konsoli Xbox One nie było możliwości pobierania modelu szkieletu śledzonego użytkownika z kontrolera Kinect, została podjęta próba wyodrębnienia obszaru dłoni z otrzymywanej mapy głębokości oraz następnie jej szkieletyzacji. Dzięki temu, byłaby możliwość śledzenia zmian położenia charakterystycznych punktów dłoni, takich jak palce, nadgarstek itp.

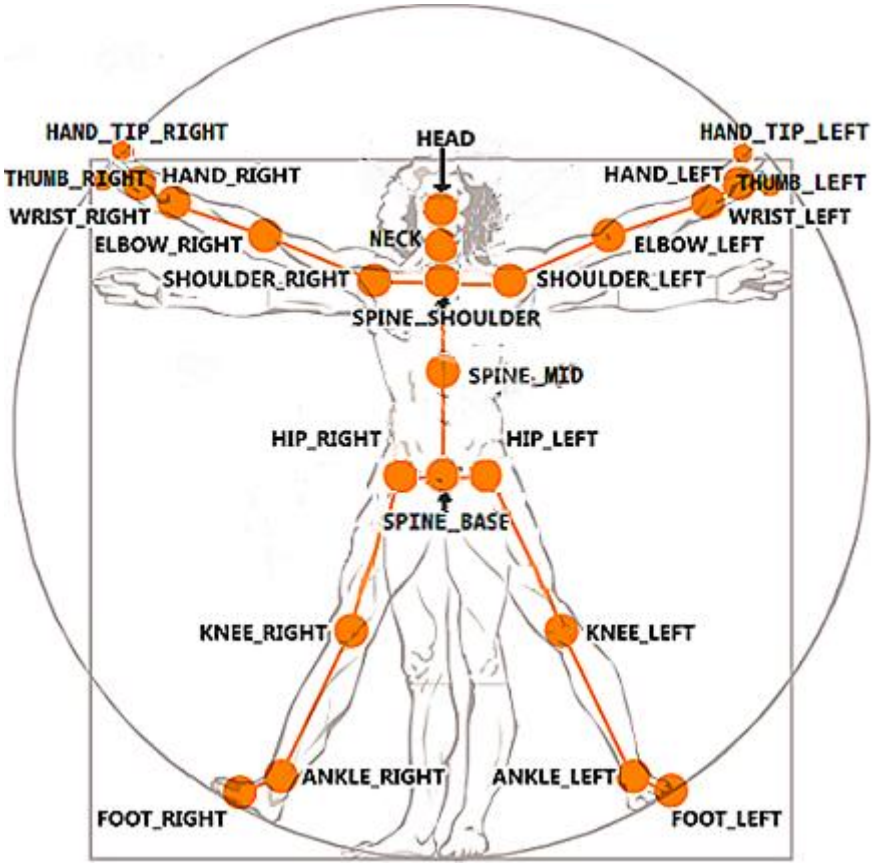
Niestety, nie udało się uzyskać satysfakcjonujących wyników odnośnie procesu ekstrakcji dłoni z mapy głębokości. Problem polega na tym, że kontroler Kinect nie zawsze zwraca prawidłowe wartości dotyczących odległości danego punktu do kontrolera. Widać to na nagraniu *BallSegmentation.mp4* załączonym w pliku *XboxRecords.zip*. Przemieszczający się biały kwadrat oznacza punkt, w którym w teorii znajduje się obiekt najbliżej położony kontrolera (pomijane są wartości równe 0 oraz poszukiwania takiego obiektu ograniczamy do środkowego obszaru kadru). Z nagranego kadru wynika, że za obiekt położony najbliżej powinna zostać uznana piłka. Jednak okazuje się, że Kinect zwraca w środkowym obszarze punkty, których przypisana wartość na mapie głębokości jest mniejsza niż dla punktów stanowiących obszar piłki. Wszystko wskazuje na to, że te punkty można uznać za pewien rodzaj szumu, który jest charakterystyczny dla kamer typu ToF (ang. *Time of Flight*). Więcej informacji o specyfice tego szumu można znaleźć na stronie [2] albo załączonej pracy [3].

Na nagraniu *BallSegmentationWithMedianFilter.mp4* można zaobserwować próbę zredukowania danego szumu za pomocą filtru medianowego. Niestety, dany zabieg nie przyniósł dobrych rezultatów i ogólnie rzecz biorąc, różne kombinacje zastosowania tego rodzaju filtru oraz filtru Gaussa zakończyły się niepowodzeniem. Dodatkowo, zaobserwowano spory narzut czasowy.

W wcześniej wspomnianych źródłach można także znaleźć opracowane przez ich autorów metody eliminacji tego rodzaju szumu. Jednak, ze względu na ich złożoność oraz obawę, że efektywność całego mechanizmu nie będzie akceptowalna z perspektywy użytkownika, postanowiłem zrezygnować z dalszych prób rozwijania systemu na konsolę Xbox One na rzecz komputera PC, w przypadku którego jest możliwość operowania na modelu szkieletu użytkownika dostarczanego przez kontroler Kinect.

Nowe podejście

Tak jak zostało wspomniane w poprzednim rozdziale, najważniejszą zmianą koncepcyjną w przygotowywanym rozwiązaniu było przejście z środowiska międzyplatformowego technologii UWP na tradycyjne podejście desktopowe, wykorzystując technologię WPF. Dzięki temu, mogłem skorzystać z oprogramowania *Kinect for Windows SDK 2.0* [4], które umożliwia pozyskanie modelu szkieletu użytkownika, którego ruchy są śledzone przez kontroler Kinect. Dany model pozwala uzyskać informacje o lokalizacji 25 punktów charakterystycznych, które zostały przedstawione na poniższej grafice:



Ze względu na fakt, że Kinect 2.0 nie wspiera śledzenia wszystkich palców u dłoni, w opracowywanym rozwiązaniu skupię się jednak na wybranych znakach ideograficznych

z amerykańskiej odmiany języka migowego (ang. *American Sign Language*, w skrócie *ASL*). Dlatego w procesie klasyfikacji konkretnego gestu wykonywanego przez użytkownika, system będzie brał pod uwagę następujące punkty charakterystyczne (poprzez * oznaczone są punkty, co do których mam jeszcze pewne wątpliwości i ostatecznie mogą nie zostać wykorzystane):

- ELBOW_LEFT i ELBOW_RIGHT,
- WRIST_LEFT i WRIST_RIGHT,
- HAND_LEFT i HAND_RIGHT,
- THUMB_LEFT i THUMB_RIGHT*,
- HAND_TIP_LEFT i HAND_TIP_RIGHT*.

Zaprojektowana aplikacja skupi się na śledzeniu ruchu wyłącznie jednego użytkownika. W przypadku znalezienia się w kadrze więcej niż jednej osoby, proces klasyfikacji gestu zostanie przerwany oraz aplikacja wygeneruje odpowiedni komunikat. Poza tym, dalej zakładam, że system umożliwi dynamiczne dodawanie kolejnych fraz do słownika, lecz przede wszystkim skupię się na znakach przedstawionych w następnym rozdziale.

W celu krótkiej prezentacji aktualnego stanu projektu, załączam do raportu nagranie *SkeletonPreview.avi*, które powinno znajdować się w pliku *PCRecords.zip*. Na nagraniu można zaobserwować, że wyróżnione wyżej punkty charakterystyczne są oznaczone kolorem fioletowym.

Wybrane gesty do rozpoznania

Wybierając poniższe znaki, chciałem, żeby były one jak najbardziej uniwersalne oraz umożliwiały weryfikację różnych scenariuszy (np. gest wykonywany za pomocą obu rąk, gest zawierający dłoń zaciśniętą). Dla każdego wybranego gestu dołączam nagranie z prawidłową jego realizacją oraz spis własnych uwag, które spostrzegłem w trakcie testów śledzenia ruchu realizowanego przez moją aplikację. Testy wykonywałem przy oddalonym o ok. 1,5 metra kontrolerze Kinect, przy równomiernym oświetleniu pokoju.

Znaczenie	**Nagranie gestu**
Hello	https://www.signingsavvy.com/sign/hello
Goodbye	https://www.signingsavvy.com/sign/goodbye
Please	https://www.signingsavvy.com/sign/please
Sorry	https://www.signingsavvy.com/sign/sorry
Thank you	https://www.signingsavvy.com/sign/THANK%20YOU/426/1
Yes	https://www.signingsavvy.com/sign/YES/493/1
No	https://www.signingsavvy.com/sign/NO/291/1
Bed (going to sleep)	https://www.signingsavvy.com/sign/BED/43/1
House	https://www.signingsavvy.com/sign/house
Book	https://www.signingsavvy.com/sign/book
Eat	https://www.signingsavvy.com/sign/EAT/151/1
Drink	https://www.signingsavvy.com/sign/DRINK/119/1

Znaczenie	Uwagi
Hello	• Gest pokazywany z wykorzystaniem jednej ręki. • Przez większy okres czasu wykonywania gestu, wszystkie punkty charakterystyczne są prawidłowo odwzorowane.
Goodbye	• Gest pokazywany z wykorzystaniem jednej ręki. • Mechanizm zwraca, że dłoń cały czas otwarta. • Podczas wykonywania gestu, wszystkie punkty charakterystyczne są prawidłowo odwzorowane.
Please	• Gest pokazywany z wykorzystaniem jednej ręki. • Problem z tym, że gest jest wykonywany, dotykając klatki piersiowej, przez co nie do końca prawidłowo są wykrywane punkty charakterystyczne. • Przy większym oddaleniu od klatki piersiowej – lepsze rezultaty śledzenia ruchu.
Sorry	• Gest pokazywany z wykorzystaniem jednej ręki. • Problem z tym, że gest jest wykonywany, dotykając klatki piersiowej, przez co jeszcze gorzej niż w przypadku znaku „*Please*” są wykrywane punkty charakterystyczne. • Przy większym oddaleniu od klatki piersiowej – lepsze rezultaty śledzenia ruchu.
Thank you	• Gest pokazywany z wykorzystaniem jednej ręki. • Dość dobre rezultaty, czasem gubi punkt określający zakończenie dłoni.
Yes	• Gest pokazywany z wykorzystaniem jednej ręki. • Prawidłowa lokalizacja punktów charakterystycznych, czasem gubi lokalizację kciuka. • Mechanizm prawidłowo zwraca, że dłoń zaciśnięta.
No	• Gest pokazywany z wykorzystaniem jednej ręki. • Nie w pełni można śledzić ten gest, ponieważ kontroler Kinect nie śledzi ruchu palca wskazującego. • Prawidłowa lokalizacja punktów charakterystycznych, czasem gubi lokalizację kciuka.
Bed (going to sleep)	• Gest pokazywany z wykorzystaniem dwóch rąk. • Jedynie punkty określające łokcie są zgodne ze stanem faktycznym. • Najprawdopodobniej gest do odrzucenia albo do pokazania, że nie zawsze Kinect zapewnia odpowiednią jakość śledzenia ruchu użytkownika.
House	• Gest pokazywany z wykorzystaniem dwóch rąk. • Przez większy okres czasu wykonywania gestu, wszystkie punkty charakterystyczne są prawidłowo odwzorowane.
Book	• Gest pokazywany z wykorzystaniem dwóch rąk. • Przez większy okres czasu wykonywania gestu, wszystkie punkty charakterystyczne są prawidłowo odwzorowane. • Czasem gubi lokalizację kciuka. • Czasem nie wykrywa, że dłoń jest zaciśnięta.

Eat	• Gest pokazywany z wykorzystaniem jednej ręki. • Porównanie do znaku „*Thank you*" – są dość podobne i może to być dobry test dla klasyfikatora. • Dość dobre rezultaty, czasem gubi punkt określający kciuka.
Drink	• Gest pokazywany z wykorzystaniem jednej ręki. • Dość dobre rezultaty, czasem gubi punkt określający kciuka.

Postać próbki do klasyfikacji gestu oraz proces ich pozyskiwania

Żeby wykonanie danego gestu było możliwe do sklasyfikowania, należy je opisać poprzez przyjęty wcześniej zbiór parametrów (cech). Po dokonaniu przeglądu proponowanych zestawów parametrów w podobnych rozwiązaniach postanowiłem bazować na poniższej propozycji przygotowanej w załączonej pracy [5]:

Table 1. Proposed features, extracted from the skeletal joints (features marked with ∗, are calculated using only HandLeft and/or HandRight).

Feature Name	Frames Involved	Equation
Spatial angle	F_2, F_1	$\arccos \frac{\mathbf{v}_2^{(J)} \cdot \mathbf{v}_1^{(J)}}{\left\|\mathbf{v}_2^{(J)}\right\| \cdot \left\|\mathbf{v}_1^{(J)}\right\|}$
Spatial angle	F_N, F_{N-1}	$\arccos \frac{\mathbf{v}_N^{(J)} \cdot \mathbf{v}_{N-1}^{(J)}}{\left\|\mathbf{v}_N^{(J)}\right\| \cdot \left\|\mathbf{v}_{N-1}^{(J)}\right\|}$
Spatial angle	F_N, F_1	$\arccos \frac{\mathbf{v}_N^{(J)} \cdot \mathbf{v}_1^{(J)}}{\left\|\mathbf{v}_N^{(J)}\right\| \cdot \left\|\mathbf{v}_1^{(J)}\right\|}$
Total vector angle	$F_1, \ldots, F_N$	$\sum_{i=1}^{N} \arccos \left(\frac{\mathbf{v}_i^{(J)} \cdot \mathbf{v}_{i-1}^{(J)}}{\left\|\mathbf{v}_i^{(J)}\right\| \left\|\mathbf{v}_{i-1}^{(J)}\right\|} \right)$
Squared total vector angle	$F_1, \ldots, F_N$	$\sum_{i=1}^{n} \arccos \left(\frac{\mathbf{v}_i^{(J)} \cdot \mathbf{v}_{i-1}^{(J)}}{\left\|\mathbf{v}_i^{(J)}\right\| \left\|\mathbf{v}_{i-1}^{(J)}\right\|} \right)^2$
Total vector displacement	F_N, F_1	$\left\|\mathbf{v}_N^{(J)} - \mathbf{v}_1^{(J)}\right\|$
Total displacement	$F_1, \ldots, F_N$	$\sum_{i=1}^{n} \left\|\mathbf{v}_i^{(J)} - \mathbf{v}_{i-1}^{(J)}\right\|$
Maximum displacement	$F_1, \ldots, F_N$	$\max_{i=2,..,N} \left(\left\|\mathbf{v}_i^{(J)} - \mathbf{v}_{i-1}^{(J)}\right\| \right)$
Bounding box diagonal length *	$F_1, \ldots, F_N$	$\sqrt{a_{B(\mathcal{V}^{(\mathcal{J})})}^2 + b_{B(\mathcal{V}^{(\mathcal{J})})}^2}$
Bounding box angle *	$F_1, \ldots, F_N$	$\arctan \frac{b_{B(\mathcal{V}^{(\mathcal{J})})}}{a_{B(\mathcal{V}^{(\mathcal{J})})}}$

Table 2. Symbols used throughout this paper and their description.

Symbol	Definition
J	a given joint
J_c, J_p	child/parent joint of J, respectively
F_i	a given video frame, $i = 1, \ldots, N$
$\mathbf{v}_i^J$	vector of 3D coordinates of J at F_i
$v_{x,i}^{(J)}, v_{y,i}^{(J)}, v_{z,i}^{(J)}$	the 3D coordinates of $\mathbf{v}_i^J$
$\mathcal{J}$	the set if all joints
$\mathcal{V}^{\mathcal{J}}$	the set of all vectors $\mathbf{v}_i^J, J \in \mathcal{J}, i = 1, 2, \ldots, N$
$B(\bullet)$	a 3D bounding box of a set of vectors
$a_{B(\bullet)}, b_{B(\bullet)}$	the lengths of the sides of $B(\bullet)$

Oczywiście, docelowy zbiór cech może się różnić od powyższego. Na pewno chciałbym również zawrzeć informację o tym, jaki jest stan dłoni w konkretnej klatce (zaciśnięta, otwarta).

Odnośnie procesu wytwarzania danych uczących, zostanie wzięte pod uwagę to, żeby wykonanie danego gestu nie miało niepotrzebnych przerw czasowych lub już po jego zakończeniu pokazywania rejestrowany nie był dalszy ruch użytkownika. Z tego powodu, najlepszym sposobem pozyskiwania danych uczących wydaje mi się współpraca dwóch osób, gdzie jedna osoba pokazuje dany znak, a druga decyduje o momencie rozpoczęcia i zakończenia nagrania. Ewentualnie, rozważana jest opcja uruchamiania trybu dodawania nowego przykładu do klasyfikatora poprzez wykonanie predefiniowanego gestu, którego potencjalne wystąpienie będzie sprawdzane z poziomu kodu. Dana możliwość wydaje się być wygodniejsza w organizacji procesu generacji danych uczących, ale może przynieść mniej dokładne próbki. Chciałbym również dodać możliwość weryfikacji wykonania gestu przez użytkownika, który będzie mógł zdecydować, czy ma on zostać dodany do klasyfikatora czy nie.

Metoda klasyfikująca

Najprawdopodobniej zostanę przy algorytmie k najbliższych sąsiadów, który opisywałem już wcześniej w dokumencie [1] (strona 5). W przypadku uzyskania dość niesatysfakcjonujących rezultatów podejmę próbę zastosowania algorytmu SVM. Do zaprojektowania klasyfikatora gestów zastosuje narzędzia udostępnianie przez darmową bibliotekę do uczenia maszynowego *Scikit – learn* [6].

Problem porównywania sekwencji czasowych – dynamiczna transformata czasowa

Ze względu na to, że dany gest można wykonać z różną szybkością, pojawia się problem porównywania tych przebiegów czasowych. Jednak, tutaj z pomocą przychodzi również wcześniej opisywany przeze mnie algorytm dynamicznej transformaty czasowej (ang. *Dynamic Time Warping*, w skrócie *DTW*), który będę chciał zastosować. Więcej informacji na jego temat można znaleźć w dokumencie [1] (strona 6).

Wstępny plan przypadków testowych

1. Klasyfikator gestów sprawdzany przez osobę, która brała udział w przygotowaniu zbioru uczącego (szacuję, że będę w stanie zorganizować 4 – 5 osoby do procesu przygotowania zbioru uczącego).
2. Klasyfikator gestów sprawdzany przez osobę, która nie brała udziału w przygotowaniu zbioru uczącego.

Obydwa przypadki testowe dotyczyłyby wszystkich wybranych znaków oraz zrealizowane byłyby one w podobnych warunkach (np. odległość od kontrolera, pozycja użytkownika, oświetlenie).

Planowany harmonogram prac

Poniżej zamieszczam założony przeze mnie harmonogram prac. Po każdym etapie postaram się Panu przesyłać kolejny raport ze zrealizowanych postępów:

- Do końca maja – realizacja mechanizmu rejestracji wykonania gestu oraz wyliczenie wszystkich parametrów potrzebnych do procesu klasyfikacji.
- Do końca czerwca – przygotowanie środowiska do wytworzenia klasyfikatora oraz zastosowanie dynamicznej transformaty czasowej.
- Do końca lipca – opracowanie zbiorów uczących oraz rozpoczęcie etapu testowania.
- Do końca sierpnia – zakończenie etapu testowania, ostatnie poprawki, najpóźniej od połowy sierpnia rozpoczęcie pisania samej pracy dyplomowej.
- Do 16.09 – złożenie pracy dyplomowej.

Bibliografia

1. Dokument *Piotrak_PDYM.pdf* zawierający wszystkie wcześniejsze założenia, koncepcje, pomysły dotyczące projektu.
2. https://www.hindawi.com/journals/jece/2016/6587162/
3. *Noise aware depth denoising for a time-of-flight camera*, Jiyoung Jung, Joon-Young Lee, In So Kweon
4. https://docs.microsoft.com/en-us/previous-versions/windows/kinect/dn782035(v=ieb.10)
5. *Real-Time Arm Gesture Recognition Using 3D Skeleton Joint Data*, Georgios Paraskevopoulos, Evaggelos Spyrou, Dimitrios Sgouropoulos, Theodoros Giannakopoulos, Phivos Mylonas
6. https://scikit-learn.org