# Sterowanie robotem za pomocą gestów
## z wykorzystaniem czujnika ruchu

**ZYGFRYD WIESZOK, ŁUKASZ LIPKA**

Politechnika Śląska

**Słowa kluczowe:**

czujnik ruchu, zdalne sterowanie, Kinect, sterowanie gestami

**Keywords:**

motion sensor, remote control, Kinect, gesture control

**Artykuł recenzowany**

**Streszczenie**

*W niniejszym artykule opisano projekt, w którym przedstawiona została metoda bezdotykowego sterowania robota z zastosowaniem czujnika ruchu Kinect. Obiektem sterowanym jest jeżdżący robot składający się z: podwozia, silnika oraz modułu do komunikacji. Sterowanie odbywa się za pomocą sensora Microsoft Kinect wraz z odpowiednią analizą gestów sterujących. Przedstawiono algorytm sterowania oraz propozycje rozwiązania problemów wynikających z charakterystyki metody sterowania. Poruszono kwestię dopasowania algorytmu do budowy fizycznej osoby sterującej, zwiększenie dokładności sterowania w kluczowych zakresach oraz problemu śledzenia wielu osób znajdujących się przed czujnikiem.*

**Abstract**

*This paper describes the project that demonstrates the use of the Kinect motion sensor to control the robot using hand gestures. The controlled object is robot comprising of: the chassis, electronic components, engine and communication module. The control is done using the Microsoft Kinect sensor and proper analysis of controls gesture.*

### Wstęp

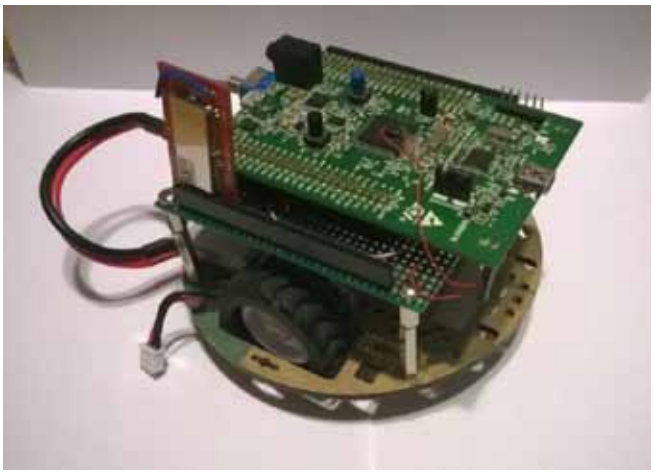
Rozwój technologii pozwala nam budować zaawansowane technologicznie roboty,jednak nie wszystkie są całkowicie autonomiczne. Duża część robotów wymaga interakcji z użytkownikiem, która jest prowadzona za pomocą dedykowanych kontrolerów. Projektowanie kontrolerów oraz interfejsów użytkownika jest bardzo ważnym elementem budowy robota, ponieważ w dużej mierze od niego zależy efektywność oraz jakość pracy osoby obsługującej urządzenie. Kontrolery powinny zapewniać możliwie najprostszą i najbardziej intuicyjną metodę sterowania. Bardzo ważne są wymagania definiowane przez warunki pracy, w jakich urządzenie jest używane. W pewnych zastosowaniach istotny jest minimalny kontakt fizyczny użytkownika z kontrolerem robota, ponieważ może być wymagana wysoka sterylność np. w warunkach medycznych. W niniejszym artykule zbadano możliwość wykorzystania sterowania bezdotykowego do kontroli urządzenia. Zostało to zrealizowane za pomocą czujnika ruchu, analizującego ruchy oraz gesty wykonywane przez osobę sterującą. Wykorzystano do tego czujnik ruchu Kinect produ-

kowany przez firmę Microsoft [1]. W celu zbadania wspomnianej metody sterowania został zbudowany prosty robot, poruszający się zgodnie z gestami wykonywanymi przez użytkownika.

## Sprzęt

Robot został zbudowany [8] z następujących części:

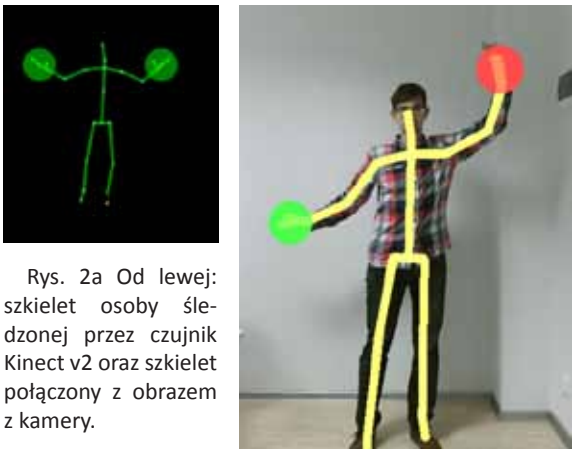
- uniwersalnego podwozia Pololu RPC04A
- 2x silnik Pololu HP 30:1 z wałem dwustronnym [4]
- 2x przedłużone mocowanie do mikrosilników Pololu
- 2x koła Pololu 42 x 19 mm
- kulkę podporową ½”
- 2x enkodery optyczne mocowane na wale silnika
- sterownik silników **TB6612FNG** [7]
- układ elektroniczny oparty o procesor **STM32F407** firmy STMicroelectronics[5]
- moduł komunikacji Bluetooth BTM-222[6]



Rys. 2 Zbudowany prototyp robota.

W rozwiązaniu zastosowano czujnik Kinect w wersji drugiej, zapewniający możliwość śledzenia ruchów ciała osób znajdujących się przed czujnikiem. Tworzy on wirtualny szkielet postaci zawierający do 26 ruchomych kości reprezentujących postawę każdej śledzonej osoby [3]. Poza śledzeniem ruchów ludzi, Kinect dostarcza również wiele strumieni danych, które mogą zostać wykorzystane w aplikacjach, m. in. jest to obraz w rozdzielczości 1080p, mapa głębokości obrazu, obraz dostarczany z kamery podczerwonej pozwalający np. monitorować warunki oświetleniowe lub pomagający w detekcji ruchów twarzy, strumień dźwiękowy pozwalający za pomocą odpowiednich sterowników na rozpoznawanie mowy.

W przedstawionym przykładzie została wykorzystana możliwość śledzenia dłoni użytkownika, których ruchy przetwarzane są na odpowiednie rozkazy sterujące robotem. Sterowanie robotem jest realizowane poprzez śledzenie położenia dłoni w stosunku do reszty ciała kontrolującej osoby. Dokument ten pokaże dokładną analizę algorytmu sterowania i funkcji czujnika Kinect w rozwiązaniu.



Rys. 2a Od lewej: szkielet osoby śledzonej przez czujnik Kinect v2 oraz szkielet połączony z obrazem z kamery.

## Różnice pomiędzy wersjami czujnika Kinect

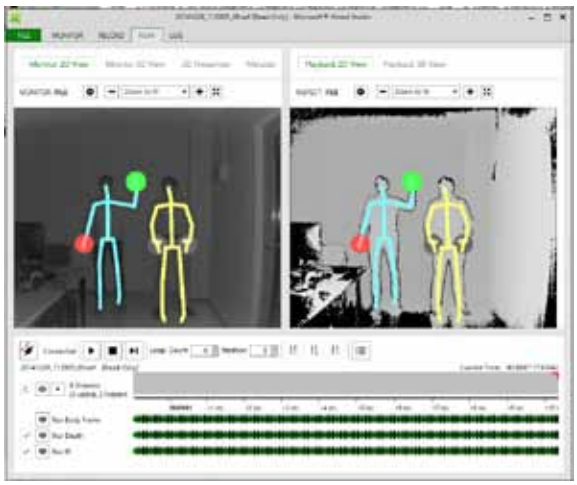
Obecnie na rynku są dostępne dwie wersje czujnika Kinect produkowanego przez firmę Microsoft – starsza wersja Kinect v1 (premiera listopad 2010) oraz nowsza wersja czujnika Kinect v2, która została zaprezentowana w lipcu 2014. W czasie pracy nad aplikacją do sterowania robotem zostały użyte obie wersje czujnika. W czasie badań wykazano, że nowsza wersja czujnika Kinect jest znacząco lepsza od swojego poprzednika. Dużą zaletą Kinect v2 jest znacząco wyższa dokładność obrazu, dostarczanego przez czujnik głębokości, co pozwoliło na lepsze odzwierciedlenie ruchów śledzonych osób. Została również zwiększona ilość śledzonych kości z 20 do 26, dzięki czemu ruchy śledzonych osób są bardziej naturalne. Liczba śledzonych osób wzrosła z dwóch do sześciu osób. Bardzo ważną funkcjonalnością wykorzystaną w tym projekcie, zapewnianą przez czujnik Kinect v2 jest śledzenie stanu dłoni użytkownika – możliwe jest rozpoznanie otwartej lub zamkniętej dłoni. Funkcjonalność ta została wykorzystana, aby możliwe było łatwe rozpoczęcie sterowania oraz jego przerwanie (skutkujące zatrzymaniem robota) poprzez zamknięcie lub otwarcie dłoni [10].


Rys. 3 U góry: Czujnik Kinect v1 [8] i v2 (Microsoft); na dole: obraz głębokości dostarczany przez czujnik Kinect v1 i v2 [9].

## Oprogramowanie wspierające czujnik Kinect

Ważną zaletą czujników Kinect jest oprogramowanie dostarczane przez firmę Microsoft, ponieważ zapewnia ono łatwy dostęp do wszystkich strumieni danych dostarczanych przez czujnik (obraz z kamery RGB, obraz kamery podczerwonej, obraz głębokości) oraz bardzo ważne dla tego projektu wyznaczone pozycje oraz orientacje kości śledzonych osób. Biblioteka pozwalająca na dostęp do tych informacji została przygotowana dla platformy .Net bazując na systemie zdarzeń, dzięki czemu w naturalny sposób może zostać wykorzystana do tworzenia aplikacji w języku C# [9].

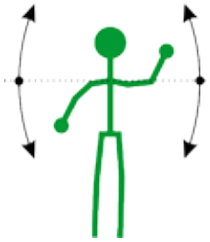
Programem wpierającym tworzenie aplikacji jest program Kinect Studio pozwalający na podgląd oraz wizualizację danych dostarczanych przez urządzenie. Najbardziej przydatną funkcją tego narzędzia jest możliwość nagrywania danych z czujnika i odtworzenie ich w sposób, umożliwiający symulację działania sensora Kinect. Funkcja ta jest bardzo użyteczna w czasie tworzenia aplikacji ponieważ nie jest wymagane podłączenie do fizycznego urządzenia dodatkowo zapewnia ona również możliwość odtwarzania wybranych fragmentów nagrania. Nagrania pochodzące z programu Kinect Studio mogą być również wykorzystane w programie Gesture Builder, w którym nagrane sekwencje ruchów są używane do maszynowego uczenia gestów.



Rys. 4 Program Kinect Studio.

## Analiza algorytmu sterowania robotem

Sterowanie robotem zostało zrealizowane poprzez śledzenie położenia dłoni użytkownika w stosunku do barków użytkownika. Każda dłoń odpowiada za sterowanie jednym silnikiem, odpowiednio prawa dłoń steruje prawym silnikiem, lewa dłoń steruje lewym silnikiem. Kierunek oraz prędkość obrotów silnika jest ustalana na podstawie położenia dłoni względem położenia barku. Bark jest punktem zerowym, w którym prędkość obrotów wynosi 0. Przesunięcie dłoni w górę skutkuje zwiększeniem prędkości obrotów do przodu, natomiast obniżenie dłoni wiąże się z zwiększeniem obrotów do tyłu. Prędkość jest ustalana analogowo poprzez podnoszenie lub opuszczanie dłoni od barku do maksymalnego wychylenia dłoni w górę lub w dół. Wykonanie gestu sterującego zostało zilustrowane na Rys. 5.



Rys. 5 Sterowanie silnikami robota.

Szkielet postaci dostarczamy przez czujnik Kinect pozwala uzyskać informacje na temat pozycji każdej kości w przestrzeni 3D. Posiadając dostęp do trójwymiarowego wektora pozycji barku, łokcia oraz nadgarstka względem środka sceny, możliwe jest wyznaczenie długości ręki, która definiuje maksymalne wychylenie. Długość ręki można wyliczyć ze wzoru na odległość w przestrzeni 3D, wyznaczając odległość pomiędzy barkiem a łokciem, a następnie sumując ją z odległością pomiędzy łokciem a dłonią[10].

Wektor reprezentujące ramię może zostać wyznaczony za pomocą wzoru:

(1) $$r = \vec{b} - \vec{k},$$

gdzie *b* jest wektorem reprezentującym pozycję stawu barkowego, *k* jest wektorem reprezentującym pozycją stawu łokciowego.

Długość ramienia *r* użytkownika jest wyznaczana wzorem:

(2) $$|\vec{r}| = \sqrt{r_x^2 + r_y^2 + r_z^2}$$

Wektor reprezentujący przedramię *p* jest wyrażony wzorem:

(3) $$\vec{p} = \vec{k} - \vec{d},$$

gdzie *k* jest wektorem pozycji stawu łokciowego, *d* jest wektorem reprezentującym pozycję dłoni.

Długość przedramienia *p* jest wyrażona wzorem:

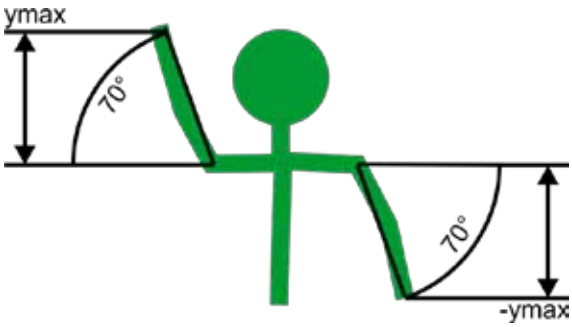
(4) $$|\vec{p}| = \sqrt{p_x^2 + p_y^2 + p_z^2}$$

Długość ręki *l* obliczana jest za pomocą wzoru:

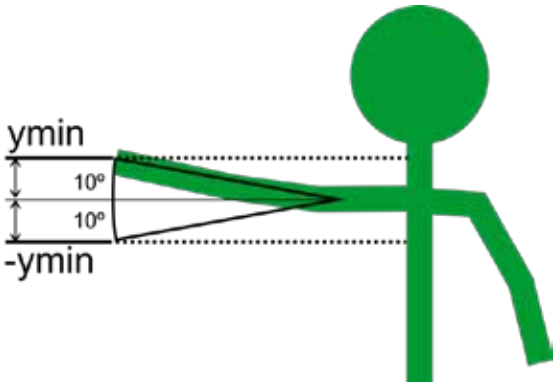
(5) $$l = |\vec{r}| + |\vec{p}|$$

Informacja o długości ręki mówi o maksymalnej wysokości, na jaką może zostać podniesiona lub opuszczona dłoń w stosunku do barku. Takie podejście jest jednak narażone na błąd wynikający z dokładności czujnika, a także z anatomii, ponieważ nie każdy użytkownik może w taki sposób podnieść rękę. W takiej sytuacji należy wprowadzić pewne przybliżenie i złagodzenie stawianych wymagań. W tym celu zastosowano odpowiednie obliczenie zakładające, że maksymalna prędkość obrotów zostanie uzyska-

na gdy dłoń znajdzie się na wysokości wyznaczonej przez pozycję dłoni przy wyprostowanej ręce tworzącej kąt 70 stopni z linią barków (Rys. 5). Ustalono również dolny próg poniżej którego szybkość wynosi 0. Próg ten jest określany za pomocą kąta tworzonego przez wyprostowaną rękę, a osią x barku. Kąt ten został określony jako 10 stopni (Rys. 5).




Rys. 5 Maksymalne zakresy pracy analogowej.




Rys. 5 Martwa strefa wyłączona z sterowania.

Minimalna wartość przesunięcia dłoni w stosunku do pozycji barku $y_{min}$, poniżej której prędkość sterowanego silnika wynosi 0, wyrażona jest wzorem:

(6) $$y_{min} = l * sin(10°)$$

Maksymalna wartość przesunięcia dłoni w stosunku do pozycji barku $y_{max}$, powyżej której prędkość sterowanego silnika wynosi 100 wyrażona jest wzorem:

(7) $$y_{max} = l * sin(70°)$$

Prędkość pracy silnika jest wyliczana na podstawie pozycji dłoni w stosunku od pozycji barku na osi y, w zakresie opisanym przez $y_{min}$ oraz $y_{max}$. Metoda została przedstawiona wzorem:

(8) $$V(y_t) = \{0,\ |y_t| < y_{min}\ 100,\ y_t > y_{max} - 100,\ y_t < -y_{max}\ \frac{(y_t - y_{min})*100}{y_{max} - y_{min}},\ y_{min} < y_t < y_{max}\ \frac{(y_t + y_{min})*100}{y_{max} - y_{min}},\ -y_t,$$

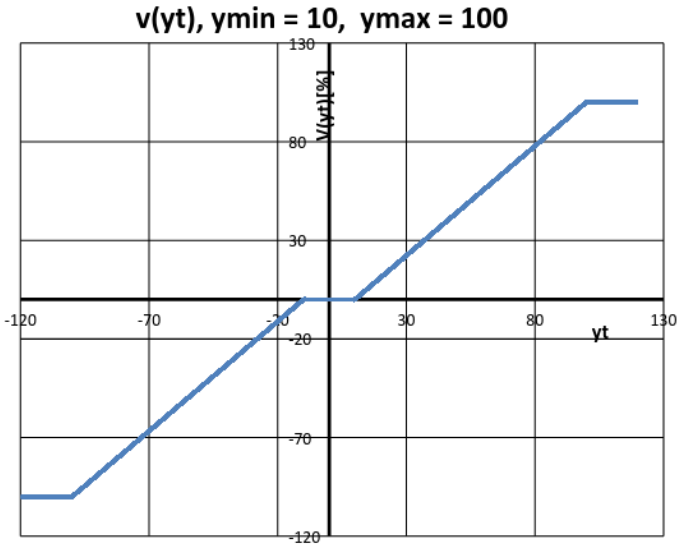
gdzie $V(y_t)$ to prędkość używana do sterowania silnikiem, *y_t* jest wyznaczane za pomocą wzoru:

(9) $$y_t = d_y - b_y$$

gdzie $y_t$ to przesunięcie dłoni w stosunku do barku w osi y, $d_y$ to składowa y wektora pozycji dłoni, $b_y$ to składowa y wektora pozycji barku.

W celu wyznaczenia prędkości obrotów silnika za pomocą wzoru (8), obliczana jest odległość pomiędzy pozycją dłoni a pozycją barku w osi y, następnie jeśli pozycja dłoni znajduje się blisko barku – odległość nie przekracza $y_{min}$ prędkość ustalana jest na 0. W przypadku gdy odległość dłoni od barku wykracza poza przyjętą maksymalną wartość odległości ($y_{min}$ lub $y_{max}$), prędkość ustalana jest na 100. W sytuacji gdy dłoń znajduje się w przedziale $(-y_{max},-y_{min})U(y_{min},y_{max})$ prędkość silnika wyznaczana jest z proporcji poprzez podzielenie odległości dłoni od barku przez maksymalne wychylenie. Ze względu na występowanie „martwej strefy" w przedziale $(-y_{min},y_{min})$ konieczne jest pomniejszenie licznika oraz mianownika o $y_{min}$. W przypadku ujemnej odległości dłoni w stosunku do barku (prędkość do tyłu), licznik jest powiększany o $y_{min}$. Wykres funkcji został zaprezentowany na Rys. 5.

W czasie testów zaobserwowano, że silniki najczęściej pracują w zakresie małych prędkości (20%-50% prędkości silnika), ponieważ w tym zakresie wykonywanie manewrów takich jak skręcanie jest najpłynniejsze oraz najdokładniejsze. Jednocześnie okazało się, że poruszanie ręką blisko barku, czyli w interesującym zakresie, jest niewygodne i mało precyzyjne dlatego konieczne okazało się zwiększenie dokładności w tym zakresie. Dodatkowo zauważono, że nie jest wymagana dokładność przy pracy z dużymi prędkościami (prędkość powyżej 70%), dlatego postanowiono zmienić funkcję wyznaczającą prędkość pracy silnika z liniowej, na funkcję nieliniową wyrażoną wzorem ogólnym:

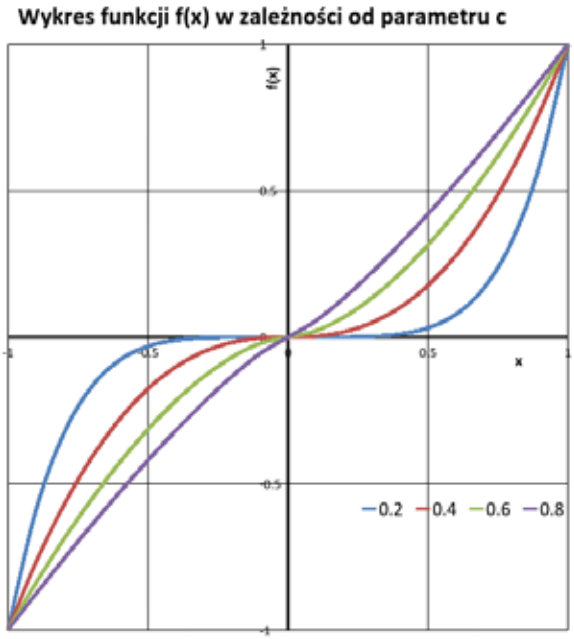



Rys. 5 Powyżej wykres funkcji wyznaczającej prędkość.

(10) $$f(x) = Sign(x) * |x|^{\frac{1}{c}}, \qquad c \in (0; 1),$$

gdzie *c* to stała określająca nieliniowość funkcji.

Współczynnik c we wzorze (10) jest stałą pozwalającą na modyfikację nieliniowości funkcji. Należało dobrać go tak, aby najlepiej spełniał przedstawione wcześniej wymagania. W tym celu przygotowano wykres różnych współczynników c, który został przedstawiony na Rys. 5.

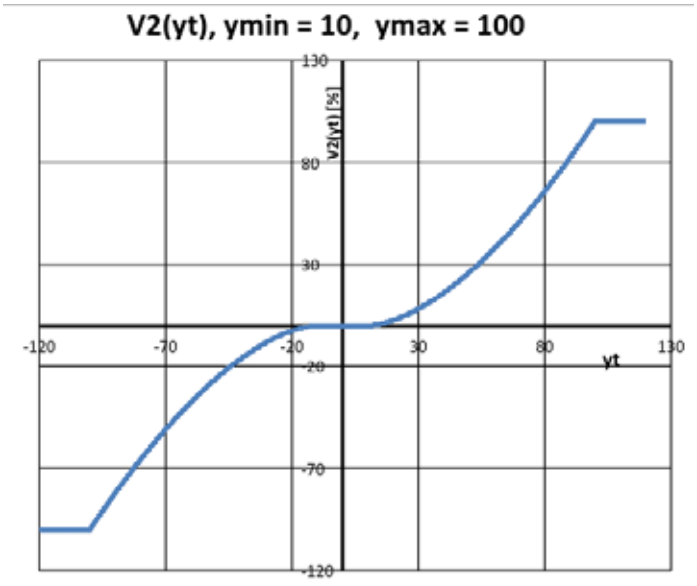



Rys. 5 Wpływ czynnika c na funkcję prędkości.

Na podstawie analizy wykresu, jako parametr c przyjęto wartość 0.4. Po empirycznym zbadaniu zachowania algorytmu sterowania z przyjętym parametrem, został on zmieniony na wartość 0.6, ponieważ dawał on wyższą dokładność sterowania.

Zmodyfikowana funkcja wyznaczania prędkości silnika (oznaczona jako $V_2(y\_t)$) została zaprezentowana za pomocą wzoru:

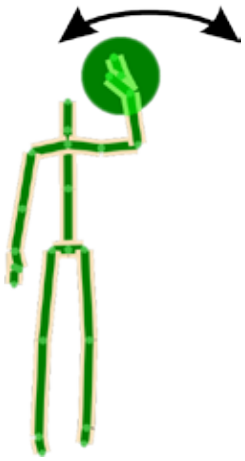
(11) $$V_2(y_t) = \{0,\ |y_t| < y_{min}\ 100,\ y_t > y_{max} - 100,\ y_t < -y_{max} \left|\frac{(y_t - y_{min})}{y_{max} - y_{min}}\right|^{\frac{1}{0.6}} * 100,\ y_{min} < y_t < y_{max} \left|\frac{(y_t + y_{min})}{y_{max} - y_{min}}\right|^{\frac{1}{0.6}}$$




Rys. 5 Wykres zmodyfikowanej funkcji wyznaczającej prędkość.

Funkcja wyznaczania prędkości wykorzystuje takie same zakresy oraz warunki brzegowe jak funkcja (8), natomiast w normalnym zakresie pracy $(-y_{max}, -y_{min})U(y_{min}, y_{max})$, stosunek położenia dłoni oraz barku jest podnoszony do potęgi 1/0.6. Wykres zmodyfikowanej funkcji wyznaczania prędkości został przedstawiony na Rys. 5.

Innym problemem wynikającym z użycia czujnika Kinect okazało się śledzenie wielu osób przez czujnik, ponieważ osobą sterującą robotem, zostaje pierwsza osoba w liście śledzonych osób – wiązało się to z problemami gdy w polu widzenia czujnika pojawiła się inna osoba, gdyż przejmowała ona sterowanie robotem. Wyeliminowanie tego problemu polegało na zidentyfikowaniu osoby kontrolującej robota, oraz zapamiętaniu jej identyfikatora, od tego momentu, tylko ta osoba może sterować robotem. Identyfikacja osoby sterującej polega na wykonaniu odpowiedniego gestu dłonią – gest ten został przedstawiony na rysunku Rys. 5. Zaproponowany gest jest zgodny z zaleceniami gestów dla aplikacji wykorzystujących Kinect opisanych w dokumencie [2].



Rys. 5 Wykonanie gestu.

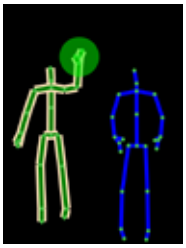
W programach wykorzystujących czujniki Kinect istnieją dwa podstawowe sposoby rozpoznawania gestów – heurystyczny lub za pomocą maszynowego uczenia gestów.

Metoda heurystyczna polega na zaimplementowaniu odpowiednich funkcji śledzących ruchy użytkownika oraz rozpoznających odpowiednie sekwencje tych ruchów jako wykonanie gestu. Wyznaczenie tych sekwencji należy do programisty. Podejście to sprawdza się w przypadku prostych gestów np. podniesienie dłoni powyżej głowy, w przypadku bardziej złożonych gestów wiąże się ono jednak z dużym nakładem pracy i kodu.

Drugą metodą jest metoda maszynowego uczenia – należy przygotować kilka lub kilkanaście (w zależności od złożoności gestu) sekwencji ruchów pozytywnych oraz negatywnych (przedstawiających niepoprawne wykonanie gestu), a następnie przy pomocy algorytmów regułowych wyznaczane są reguły opisujące wystąpienie gestu. W tym przypadku istnieją również 2

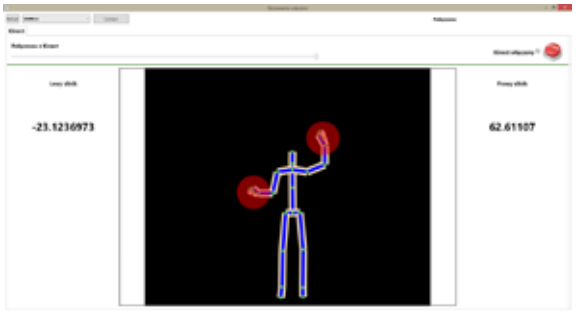
metody detekcji gestu – dyskretna oraz ciągła – w programie Visual Gesture Builder działają one w oparciu o algorytmy, odpowiednio **Adaptive Boosting** oraz **Random Forest Regression**. W przypadku gestów dyskretnych otrzymujemy informację na temat tego czy gest został wykonany czy nie, oraz jaki jest poziom zaufania do tej oceny, natomiast w przypadku detekcji ciągłej otrzymujemy informację o procencie wykonania gestu.

W projekcie zdecydowano się na użycie metody maszynowego uczenia gestu wykorzystując detekcję dyskretną. W tym celu wykorzystano narzędzie Kinect Studio aby nagrać sekwencje ruchów, a następnie wykorzystano Visual Gesture Builder, aby oznaczyć sekwencje ruchów jako gesty pozytywne oraz negatywne. Do poprawnego rozpoznania gestu konieczne było nagranie 6 klipów zawierających w sobie kilkukrotne wykonanie poprawnego oraz niepoprawnego gestu, został również nagrany dodatkowy klip umożliwiający przetestowanie wygenerowanych reguł. Ogólnie oznaczono 2016 klatek z przykładami pozytywnych oraz 4926 klatek z przykładami negatywnych. Po wygenerowaniu reguł, przetestowano wynikowy model za pomocą przygotowanego wcześniej zbioru testowego, składającego się z 470 klatek. Dokładność klasyfikacji zbioru testowego (błąd średnio kwadratowy – RMS) wyniosła 0.6834408. Zdolności klasyfikacyjne wygenerowanego modelu zostały przetestowane za pomocą funkcji **Live-Preview** dostępnej w programie Visual Gesture Builder. Funkcja ta pozwala na żywo obserwować stopień zaufania dla aktualnie wykonywanego gestu. Stopień zaufania dla poprawnie wykonanego gestu w trakcie sesji testowej zawierał się w przedziale pomiędzy 70% a 90%. Minimalny stopień zaufania dla gestu przyznającego użytkownikowi możliwość sterowania robotem, został ustawiony na 50%, co w badaniach okazało się wystarczające. Na rysunku Rys. 5 przedstawiono sytuację rozpoznania gestu, po której zostaje podświetlona osoba aktualnie kontrolująca urządzenie.



Rys. 5 Rozpoznanie gestu.

Ważną funkcjonalnością wprowadzoną przez Kinect v2 jest możliwość śledzenia stanu dłoni tj. czy dłoń użytkownika jest zamknięta czy otwarta. Wspomniana możliwość została wykorzystana w programie, aby umożliwić użytkownikowi natychmiastowe zatrzymanie silnika oraz poruszanie rękoma nie powodujące uruchamiania silników. Ręce użytkownika nie są używane do sterowania robotem dopóki nie zostaną one zaciśnięte w pięść, co pozwala użytkownikowi np. opuścić ręce nie powodując włączenia silników, dopiero po zaciśnięciu dłoni, ich pozycja jest przeliczana na sygnały sterujące silnikami. Funkcja ta jest również wykorzystywana przy detekcji gestu rozpoznania użytkownika sterującego, ponieważ gest ten musi zostać wykonany otwarta dłonią, co pozwala uniknąć detekcji tego gestu w czasie sterowania robotem, gdy pięści są zaciśnięte.




Rys. 5 Aplikacja sterująca robotem za pomocą czujnika Kinect.

## Podsumowanie

Czujnik Kinect posiada duży potencjał w dziedzinie sterowania urządzeniami wymagającymi interakcji z użytkownikiem. Taka metoda sterowania okazała się intuicyjna dla użytkowników. Zaskoczeniem była precyzja takiej metody sterowania pozwalając na wykonywanie skomplikowanych manewrów wykonywanych przez robota. Po realizacji projektu zauważono wiele zastosowań czujnika Kinect w kontrolowaniu otaczających nas urządzeń i robotów. Szczególnie docenianą cechą jest możliwość sterowania bezdotykowego, dzięki czemu może on być zastosowany w miejscach gdzie dotykanie pulpitu sterowniczego może być ryzykowne lub wymagana jest wysoka sterylność np. może być on wykorzystany do sterowania aparaturą znajdującą się na sali operacyjnej, nie wymagającej dużej precyzji – włączanie lamp lub dostępu do bazy danych zawierających wyniki badań pacjenta (np. przeglądanie zdjęć rentgenowskich).

Czujnik może również zostać wykorzystany do sterowania manipulatorów poprzez odzwierciedlanie ruchów użytkownika. Dodatkowo daje on możliwość połączenia tej metody sterowania z okularami Oculus Rift, dając operatorowi obraz 3D z kamer monitorujących pracę manipulatora oraz jego otoczenia. Taka metoda sterowania wymaga jednak wypracowania zaawansowanych algorytmów, pozwalających na eliminację zakłóceń wynikających z zasady działania czujnika Kinect (analiza obrazu) oraz błędów operatora.

## Bibliografia

1. http://www.microsoft.com/en-us/kinectforwindows/ (dostęp: 06.01.2015)
2. Microsoft Corporation, Human Interface Guidelines for Kinect 2.0 http://download.microsoft.com/download/6/7/6/676611B4-1982-47A4-A42E--4CF84E1095A8/KinectHIG.2.0.pdf (dostęp: 06.01.2015)
3. Webb J., Ashley J.:Beginning Kinect Programming with the Microsoft Kinect SDK. Apress 2012.
4. Pololu, Micro Metal Gearmotor HP motor characteristic, https://www.pololu.com/product/1093 (dostęp: 14.12)
5. StMicroelectronics, STM32F4Discovery user manual, http://www.st.com/st-web-ui/static/active/en/resource/technical/document/user_manual/DM00039084.pdf
6. Tomasz Starak, Minimoduł Bluetooth z układem BTM-222, ELEKTRONIKA PRAKTYCZNA 8/2011, s. 56-57
7. Toshiba. Dokumentacja sterownika TB6612FNG. Pololu. [dostęp: 07 01 2015.] https://www.pololu.com/file/0J86/TB6612FNG.pdf.
8. Cook David. Budowa robotów dla początkujących. : Helion, 2009.
9. Gregory S. MacBeth, C# Programmer's Handbook, Publication Date: October 29, 2003, ISBN13: 978-1-59059-270-0
10. Z. Wieszok, K. Tokarz, Mobilna platforma zdalnie programowana, Politechnika Śląska, Gliwice, styczeń 2015, s.10-35