BIULETYN INSTYTUTU AUTOMATYKI I ROBOTYKI
NR 33, 2012

Zastosowanie kontrolera ruchu i środowiska Matlab do budowy interfejsów typu NUI

Tomasz PAŁYS

Instytut Teleinformatyki i Automatyki WAT,
ul. Gen. S. Kaliskiego 2, 00-908 Warszawa
tomasz.palys@wat.edu.pl

STRESZCZENIE: W artykule opisano zastosowanie kontrolera ruchu Microsoft Xbox 360 Kinect i środowiska Matlab do prototypowania interfejsów, które umożliwiają sterowanie za pomocą gestów. Opisano architekturę systemu oraz sposób komunikacji z kontrolerem ruchu i środowiskiem Matlab. Zaproponowano sposób generowania kodu w języku C++ na podstawie skryptów Matlaba, a także sposób dołączenia go do budowanego systemu.

SŁOWA KLUCZOWE: kontroler ruchu, Microsoft Kinect, Matlab Coder, naturalny interfejs użytkownika, NUI

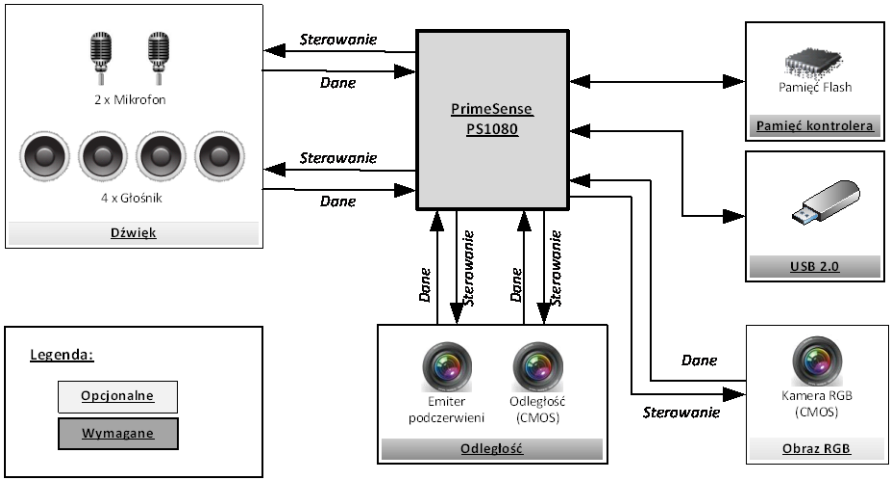
1. Wprowadzenie

Obecnie coraz większym zainteresowaniem cieszą się naturalne interfejsy użytkownika (ang. *Natural User Interface*, NUI). Interfejsy tego typu ułatwiają sterowanie i zwiększają komfort pracy operatora (użytkownika), wykorzystując naturalne sposoby komunikowania się pomiędzy ludźmi. Przedstawione opisy badań naukowych w literaturze przedmiotu, odnoszą się do sposobów modelowania interakcji operator-system oraz metodyk projektowania interfejsów NUI [12]. Dostępne publikacje naukowe poruszają również problem konstruowania i implementacji algorytmów rozpoznawania mowy czy gestów [7], [14], [15], [16].

Jednym z rodzajów interfejsów NUI są interfejsy, które umożliwiają sterowanie za pomocą gestów. Potencjalnie mogą one ułatwić funkcjonowanie osobom niepełnosprawnym lub ułatwić skuteczną rehabilitację, a także zwiększyć funkcjonalność systemów sterowania, czy wirtualnej rzeczywistości. Do ich konstruowania nadają się tzw. kontrolery ruchu, które są dostępne

w sprzedaży od kilku lat. Obecne na rynku są dostępne kontrolery firmy *Microsoft* i *Asus*, które zostały zbudowane na bazie mikrokontrolera firmy *PrimeSense PS1080* [11].

Schemat logiczny kontrolera, zbudowanego na bazie mikrokontrolera *PS1080*, przedstawiono na rys. 1. Wymaganymi elementami kontrolera są emiter światła laserowego w zakresie podczerwieni i matryca CMOS (ang. *Complementary Metal Oxide Semiconductor*), nazywana kamerą głębokości. W wyniku ich zastosowania, możliwe jest oszacowanie współrzędnych każdego punktu obrazu cyfrowego. Wykorzystując to, firma *PrimeSense* opracowała i opatentowała metodę oszacowania współrzędnych ludzkiego szkieletu w dyskretnej przestrzeni 3D [3]. Opcjonalnie, do kontrolera mogą zostać dołączone mikrofony, głośniki oraz kamera RGB.



Rys. 1. Schemat logiczny kontrolera ruchu [11]

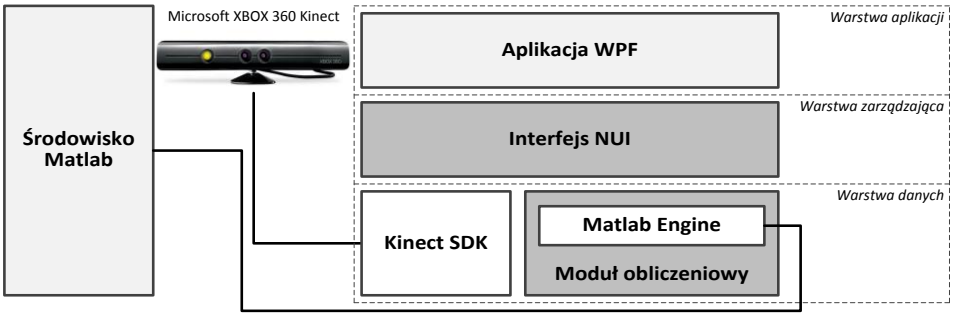
Efektem współpracy pomiędzy firmą *PrimeSense* a firmą *Microsoft*, było skonstruowanie kontrolera ruchu *Microsoft Xbox 360 Kinect*. Pierwotnie planowano wykorzystać go jako dedykowany kontroler konsoli *Microsoft Xbox360*. Jednak ze względu na jego właściwości możliwe jest zastosowanie go do budowy interfejsu NUI.

Istnieje wiele propozycji wykorzystania kontrolera ruchu do budowy systemów lub interfejsów NUI, umożliwiających sterowanie za pomocą gestów. Dotyczą one jednak wybranych środowisk programistycznych. Znane są systemy [5], w których możliwe jest wykorzystanie kontrolera tylko w środowisku do obliczeń naukowo – inżynierskich *Matlab*, firmy *MathWorks*. Inne propozycje dotyczą głównie wykorzystania kontrolera w środowisku programistycznym *Microsoft Visual Studio 2010* [6]. Powyższe rozwiązania mają swoje wady i zalety. Dlatego proponuje się połączyć zalety obu tych

rozwiązań. Środowisko *Matlab* można wykorzystać do opracowania algorytmów rozpoznawania gestów i ich testowania. Pozostałą część systemu proponuje się zaimplementować w środowisku programistycznym *Microsoft Visual Studio 2010* tak, aby spełnić wymagania stawiane przed systemami czasu rzeczywistego. Istotne jest opracowanie metody komunikacji i przekazywania danych, pomiędzy środowiskiem *Matlab* a środowiskiem programistycznym. Należy dodać, że środowisko Matlab umożliwia automatyczne wygenerowanie kodu w języku C++ oraz biblioteki statycznej. Pozwoli to na zmniejszenie kosztów i skrócenie czasu budowy prototypu systemu rozpoznawania gestów.

2. System sterowania za pomocą gestów

Założono, że system sterowania za pomocą gestów, będzie uruchamiany na komputerze typu desktop, pracującym pod kontrolą systemu operacyjnego *Microsoft Windows 7*. Do jego budowy wykorzystano środowisko programistyczne *Microsoft Visual Studio 2010* oraz kontroler ruchu *Microsoft Xbox 360 Kinect*. Dodatkowo przyjęto założenie, że algorytmy rozpoznawania gestów, zostaną opracowane w środowisku *Matlab*.



Rys. 2. Trójwarstwowa architektura systemu sterowania za pomocą gestów

Na rys. 2 przedstawiono trójwarstwową architekturę zbudowanego systemu, który umożliwia sterowanie za pomocą gestów. W pierwszej warstwie, warstwie danych, znajdują się dwie biblioteki statyczne, które zostały zaimplementowane w kodzie niezarządzanym (ang. *unmanaged code*). Warstwę zarządzającą stanowi moduł *Interfejs NUI*, natomiast trzecią warstwą jest *Aplikacja WPF* (ang. *Windows Presentation Foundation*). Elementy te zostały zaimplementowane w kodzie zarządzanym (ang. *managed code*).

Za pozyskanie danych odpowiada kontroler *Microsoft Xbox 360 Kinect*. Umożliwia on oszacowanie współrzędnych szkieletu ludzkiego ciała w dyskretnej przestrzeni 3D oraz pozyskanie strumieni dźwięku i wideo. Dane

w postaci strumieni przekazywane są do modułu *Interfejs NUI*, za pośrednictwem funkcji zaimplementowanych w bibliotece *Microsoft Kinect SDK*. Biblioteka zawiera sterowniki i niezbędne funkcje, które umożliwiają implementację programów w języku C# albo C++. Po otrzymaniu danych *Interfejs NUI* wykorzystuje *Moduł obliczeniowy*, aby wyznaczyć wszystkie żądane parametry i zrealizować zaimplementowane algorytmy. Obliczenia są realizowane w środowisku *Matlab*, za pośrednictwem biblioteki *Matlab Engine*. Funkcje tej biblioteki można wykorzystać podczas implementacji w języku C++. Na podstawie wyznaczonych parametrów rozpoznawane są gesty a odpowiednie sygnały sterujące są przekazywane do aplikacji, która została zaimplementowana w języku C#, z wykorzystaniem technologii WPF. Zastosowanie tej technologii umożliwia szybkie i łatwe zobrazowanie wyników rozpoznawania, a w konsekwencji weryfikację zaimplementowanych algorytmów rozpoznawania gestów.

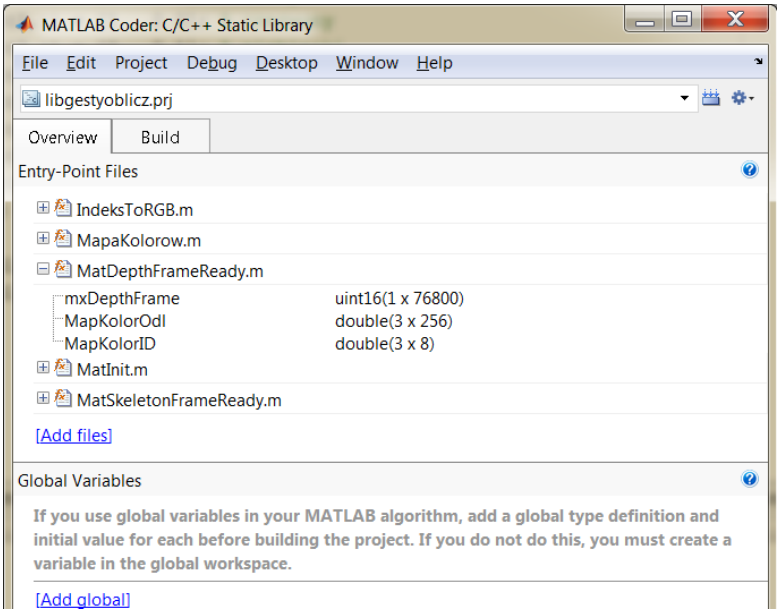
Tab. 1. Ustawienia właściwości projektu „Interfes NUI”[1]

Rodzaj	Nazwa	Wartość
General		
Configuration Type		Dynamic Library (.dll)
Common LanguageRuntime Support		Common Language Runtime Support (/clr)
C++		
General	Additional Include Directories	$(KINECTSDK10_DIR)inc; $(MATLAB7_DIR)extern\include;
Code generation	Runtime Library	Multi-threaded Debug DLL (/MDd)
Precompiled Header	Precompiled Header	Not Using Precompiled Headers
Linker		
General	Additional Library Directories	$(KINECTSDK10_DIR)lib\amd64; $(MATLAB7_DIR)extern\lib\win64\microsoft;
Input	Additional Dependencies	Kinect10.lib; libmx.lib; libeng.lib;

Interfejs NUI został zaimplementowany jako oddzielna biblioteka dynamiczna typu CLR (ang. *Common Language Runtime*). Umożliwia to dołączenie jej do innych aplikacji napisanych w kodzie zarządzanym. Najważniejsze ustawienia dla kompilatora języka C++ i programu konsolidującego (ang. *Linker*) przedstawiono w tab. 1. Ustawienia uwzględniają zarówno programową obsługę kontrolera ruchu, jak i współpracę ze środowiskiem *Matlab*. Dodatkowo należy jeszcze uwzględnić to,

[1] Aby ustawienia przedstawione w tabeli zostały poprawnie zinterpretowane przez środowisko programistyczne, należy w systemie zdefiniować zmienną środowiskową *MATLAB7_DIR* o wartości *C:\Program Files\MATLAB\R2011b*.

że konstruowanie algorytmów będzie odbywało się dwuetapowo. Pierwszy etap będzie polegał na konstruowaniu algorytmów w środowisku *Matlab* i ich testowaniu. Drugi etap, to wygenerowanie biblioteki statycznej za pomocą przybornika *Matlab Coder* [9] i dołączenie jej do budowanego systemu. Dlatego proponuje się wykorzystać kompilację warunkową. W pierwszej kolejności zdefiniować identyfikator (`#define`) `MatlabEngine`. Następnie kod, który będzie wykonywany w pierwszym etapie, umieścić pomiędzy dyrektywą `#ifdef MatlabEngine` a dyrektywą `#else`. Z kolei fragment kodu, wykonywany podczas drugiego etapu (jeżeli zostanie usunięty identyfikator `MatlabEngine`), umieścić pomiędzy dyrektywą `#else` a `#endif`.



Rys. 3. Okno przybornika *Matlab Coder*

W celu automatycznego wygenerowania biblioteki statycznej, na podstawie skryptów *Matlaba*, należy wykonać komendę `coder`. Pojawi się okno (rys. 3), które umożliwi dodanie odpowiednich skryptów do projektu biblioteki. Następnie przechodząc do zakładki *Build* należy ustawić żądane opcje projektu i wygenerować niezarządzaną bibliotekę statyczną oraz odpowiadający jej kod w języku C++, który można dołączyć do projektu systemu rozpoznawania gestów. Należy dodać, że nie dla wszystkich funkcji *Matlaba* można automatycznie wygenerować kod za pomocą przybornika *Matlab Coder*. Jednak, zgodnie z zapowiedzią firmy *MathWorks*, możliwości przybornika będą stopniowo zwiększane [9].

3. Kontroler Microsoft Xbox 360 Kinect

Istotnym elementem systemu, umożliwiającego sterowanie za pomocą gestów, jest kontroler ruchu. Parametry techniczne kontrolera ruchu *Microsoft Xbox 360 Kinect* przedstawiono w tab. 2. Kontroler można podłączyć do komputera klasy PC za pośrednictwem portu USB i specjalnego zasilacza. Minimalne wymagania sprzętowe są następujące [4]:
- dwurdzeniowy procesor 2,66 GHz o architekturze x86 lub x64,
- karta grafiki wspierająca *DirectX 9.0c*,
- 2 GB pamięci RAM,
- dedykowany port USB 2.0,
- zasilacz umożliwiający podłączenie kontrolera do portu USB.

Tab. 2. Parametry techniczne kontrolera ruchu Microsoft Xbox 360 Kinect[2]

Parametr	Wartość
Kąt widzenia w poziomie	57^0
Kąt widzenia w pionie	43^0
Mechaniczne pochylenie kontrolera w pionie	$\pm 27^0$
Max. szybkość pozyskiwania obrazów ruchomych	30 klatek/s
Domyśla rozdzielczość strumienia wideo	640 × 480
Domyśla rozdzielczość strumienia głębokości	640 × 480
Format kodowania dźwięku	16 - bit PCM (mono)
Częstotliwość próbkowania dźwięku	16 kHz
Liczba mikrofonów	4
Konwersja analogowo-cyfrowa	24 bity
Filtracja cyfrowa dźwięku	eliminacja echa i szumów

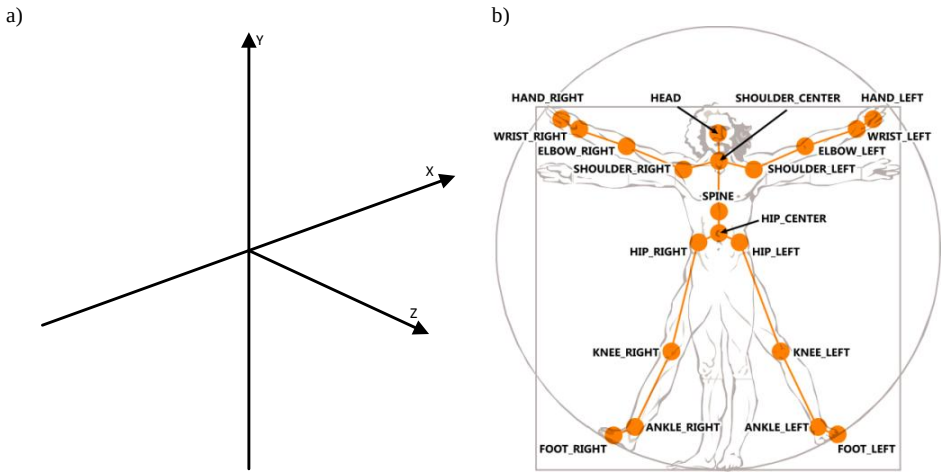
Z kolei minimalne wymagania programowe są następujące [4]:
- system operacyjnego *Microsoft Windows 7*,
- środowisko uruchomieniowego *.NET Framework 4*,
- środowisko programistyczne *Microsoft Visual Studio 2010*,
- pakiet programistyczny *Microsoft Kinect SDK 1.0* (zawiera sterowniki).

Kontroler *Microsoft Xbox 360 Kinect* umożliwia pozyskanie następujących strumieni danych: współrzędnych ludzkiego szkieletu, odległości od kontrolera ruchu, audio oraz wideo. Współrzędne ludzkiego szkieletu są wyznaczane w prawoskrętnym układzie współrzędnych, który przedstawiono na rys. 4a (oś *Z* pokrywa się z osią optyczną kamery głębokości). Z kolei na rys. 4b

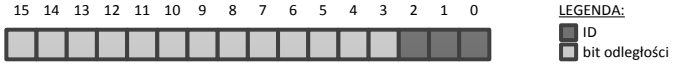
[2] http://msdn.microsoft.com/en-us/library/hh855355.aspx

przedstawiono charakterystyczne punkty ludzkiego szkieletu, których współrzędne są wyznaczane przez kontroler. Kontroler może zidentyfikować sześć osób, ale maksymalnie może „śledzić" dwie osoby, czyli dwa szkielety.

Szacowanie odległości odbywa się w przedziale od 400 mm do 4 000 mm. W tym celu zastosowano technikę oświetlania przestrzeni 3D tzw. światłem strukturalnym (ang. structured light) [14], [12]. Emiter światła laserowego generuje siatkę punktów w zakresie podczerwieni. Po odbiciu się promieni światła podczerwonego od obserwowanego obiektu, na matrycy CMOS powstaje obraz rastrowy. Analizując powstałe zniekształcenia w obrazie rastrowym, można wyznaczyć kształt obiektu oraz oszacować jego odległość od kontrolera ruchu. Odległość wyznaczana jest dla każdego punktu obrazu (30 klatek/s) o rozdzielczości: 320×240, 640×480 i 80×60. Następnie do aplikacji przekazywany jest strumień szesnastobitowych liczb, których interpretację przedstawiono na rys. 5. Pierwsze trzy, najmniej znaczące bity, to indeks szkieletu. Indeks umożliwia wyznaczenie obszaru zajmowanego przez „śledzoną osobę" (maksymalnie dwie). Pozostałe trzynaście bitów to szacunkowa odległość, która podawana jest w milimetrach.



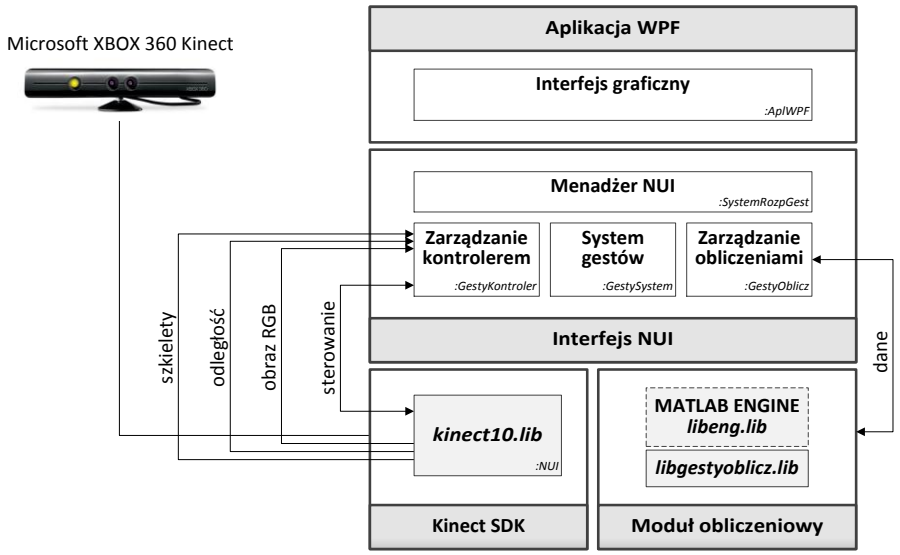
Rys. 4. Śledzenie szkieletu przez kontroler ruchu [13]
a) przestrzeń szkieletu; b) charakterystyczne punkty ciała (szkieletu)



Rys. 5. Interpretacja „strumienia odległości" [2]

Dodatkowo, dzięki kontrolerowi mamy możliwość pozyskania strumienia wideo. Istnieje możliwość pozyskania obrazu w standardzie RGB o rozdzielczości 1280 × 960 piksela (12 klatek/s) albo 640 × 480 (30 klatek/s). Można również pozyskać obraz wideo w standardzie YUV o rozdzielczość 640 × 480 (15 klatek/s). Kontroler umożliwia również pozyskanie 16-bitowych próbek dźwięku, zakodowanych zgodnie ze standardem PCM (mono). Warto również dodać, że w połączeniu z *pakietem Microsoft Speech Platform* [10], możliwe jest zbudowanie interfejsu NUI, który umożliwia sterowanie systemem czasu rzeczywistego za pomocą komend głosowych (również języka polskiego). Jednak z punktu widzenia budowanej aplikacji najbardziej istotne są informacje o odległości i szkieletach. Dlatego w dalszej części artykułu, informacje o strumieniu audio i wideo, zostaną pominięte.

4. Interfejs NUI umożliwiający sterowanie za pomocą gestów

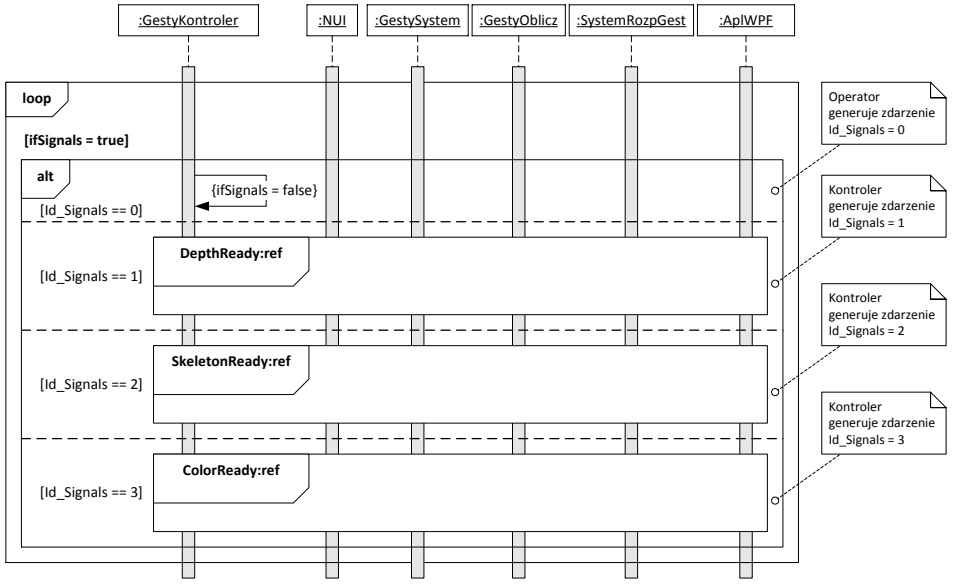


Rys. 6. Schemat blokowy systemu umożliwiającego sterowanie za pomocą gestów

Na rys. 6 przedstawiono trójwarstwową architekturę systemu umożliwiającego sterowanie za pomocą gestów. W pierwszej warstwie, *warstwie danych*, znajdują się dwa moduły. Pierwszy moduł *Kinect SDK*, to biblioteka statyczna *kinect10.lib*. Biblioteka umożliwia programowe sterowanie i obsługę kontrolera. Drugi moduł, *moduł obliczeniowy*, to biblioteka statyczna *libeng.lib*. Umożliwia ona realizację obliczeń w środowisku *Matlab*. W wyniku tego można łatwo i szybko opracować algorytmy, które realizują obliczenia na potrzeby systemu rozpoznawania gestów. Ostatecznie proponuje

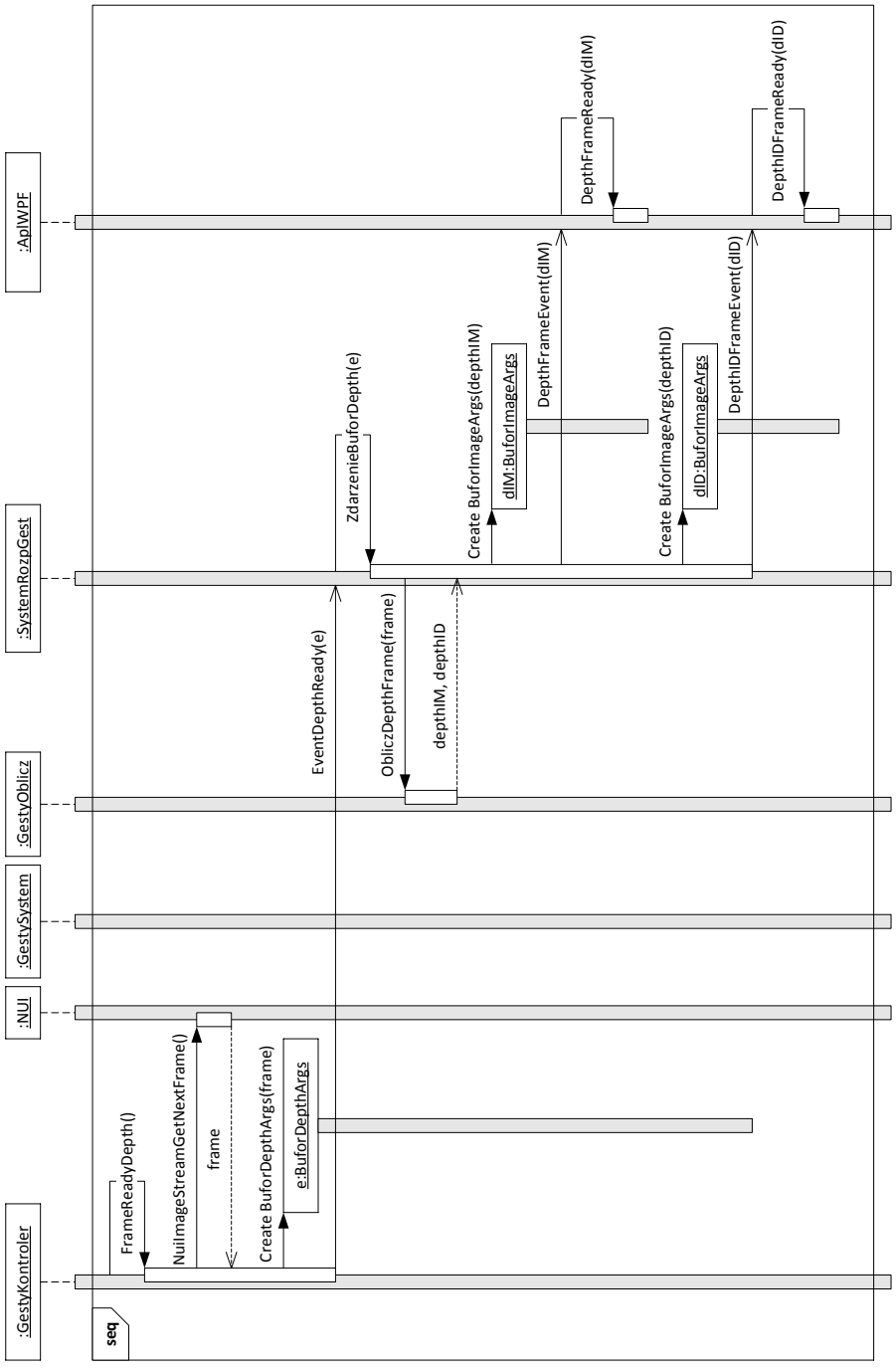
się zamienić bibliotekę *libeng.lib* na bibliotekę statyczną *libgestyoblicz.lib*. Biblioteka ta powinna być zaimplementowana w języku C++ i zawierać funkcje realizujące algorytmy opracowane w środowisku *Matlab*. Można ją wygenerować automatycznie na podstawie skryptów *Matlaba* za pośrednictwem przybornika *Matlab Coder*.

W warstwie drugiej, czyli *warstwie zarządzającej*, znajduje się *Interfejs NUI*. Jego głównym elementem jest moduł *Menadżer NUI*. Jednym z zadań menadżera jest pozyskiwanie danych z kontrolera i sterowanie nim, co odbywa się za pośrednictwem modułu *Zarządzanie kontrolerem*. Po otrzymaniu danych z kontrolera, menadżer zleca wykonanie obliczeń modułowi *Zarządzanie obliczeniami*. Uzyskane parametry przekazuje do modułu *System gestów*. Na podstawie parametrów uzyskanych w kolejnych chwilach czasu *System gestów* podejmuje decyzję o rozpoznaniu gestu. W przypadku rozpoznania gestu, moduł przekazuje taką informację menadżerowi, który z kolei powinien przekazać ją do warstwy aplikacji, czyli do modułu *Aplikacja WPF*.

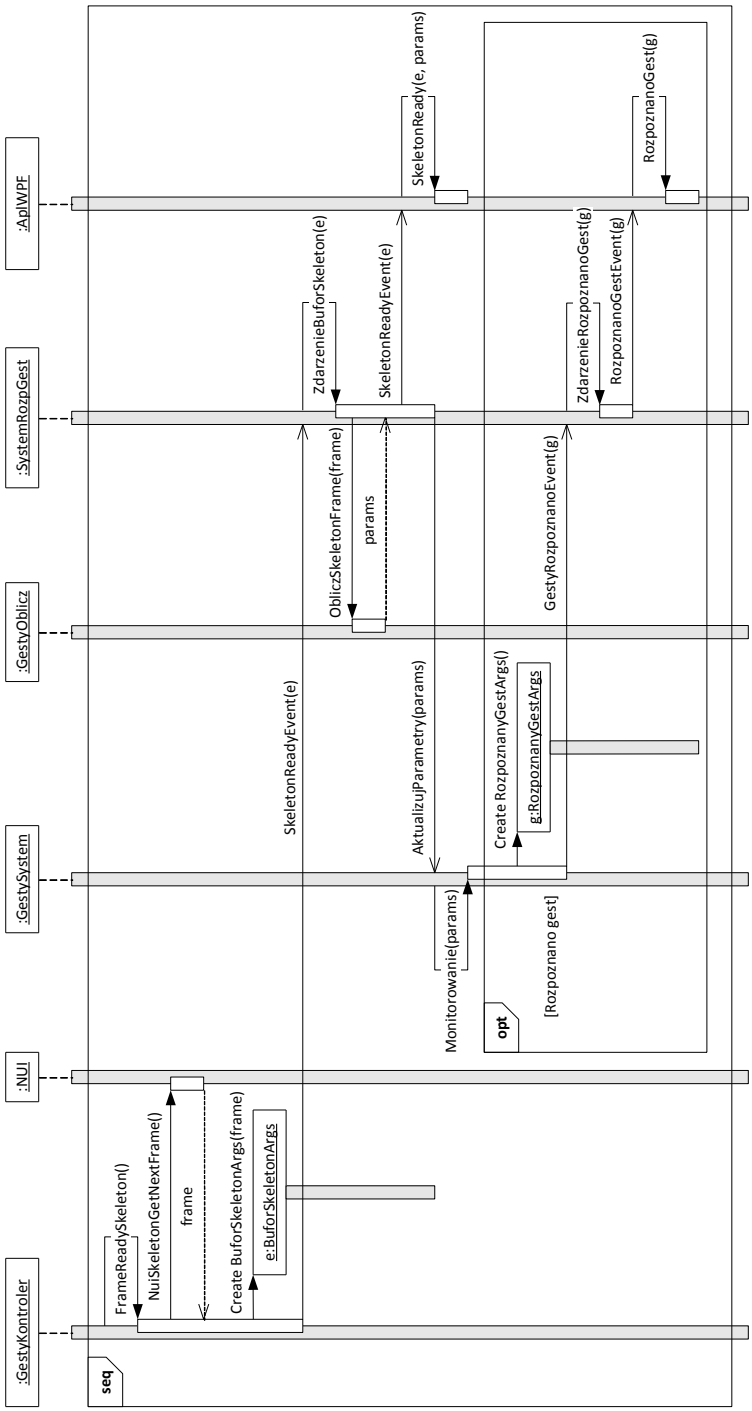


Rys. 7. Ogólny diagram sekwencji

Komunikacja pomiędzy modułami systemu odbywa się za pomocą mechanizmu zdarzeń. Ogólny diagram sekwencji przedstawiono na rys. 7. Przyjęto, że operator generuje zdarzenie (*Id_Signals = 0*), które kończy działanie aplikacji. Pozostałe zdarzenia są generowane przez kontroler. Zdarzenie *DepthReady* (*Id_Signals = 1*), generowane jest w chwili pozyskania informacji przez kamerę głębokości. Szczegóły związane ze sposobem obsługi tego zdarzenia przedstawiono na rys. 8.



Rys. 8. Obsługa zdarzenia związanego z pozyskaniem informacji z kamery głębokości



Rys. 9. Obsługa zdarzenia związanego z pozyskaniem informacji o szkielecie

Ponieważ obsługa zdarzenia *ColorReady* (*Id_Signals = 3*), związanego z pozyskaniem obrazu RGB jest podobna, pominięto szczegółowy diagram opisujący obsługę tego zdarzenia. W przypadku pozyskania informacji o śledzonych szkieletach, obsługę zdarzenia *SkeletonReady* (*Id_Signals = 2*) przedstawiono rys. 9.

Rys. 10. Okno główne aplikacji WPF

Na rys. 10 przedstawiono przykładowe okno aplikacji WPF, która umożliwia zobrazowanie wyników rozpoznawania oraz testowanie zaimplementowanych algorytmów. W przypadku tego systemu gesty są rozpoznawane na podstawie wartości kąta skierowanego, pomiędzy linią kręgosłupa a linią jaką tworzy łokieć i nadgarstek ręki. System rozpoznaje odchylenie każdej ręki co $45^0 \pm 10^0$. W lewej części okna przedstawiono parametry i wynik rozpoznania. Środkowa część okna odpowiada za zobrazowanie informacji pozyskanych z kamery głębokości, a prawa z kamery RGB. Zbudowany w ten sposób interfejs NUI, można dołączyć do dowolnej aplikacji działającej w środowisku uruchomieniowym *.NET Framework 4.0*.

5. Podsumowanie

W artykule przedstawiono trójwarstwową architekturę systemu czasu rzeczywistego z interfejsem NUI, który umożliwia sterowanie za pomocą gestów. Połączono możliwości środowiska programistycznego *Microsoft Visual*

Studio 2010 i środowiska do obliczeń naukowo – inżynierskich *Matlab*, firmy *MathWorks*. System umożliwia szybkie i łatwe konstruowanie oraz testowanie algorytmów rozpoznawania gestów. Dodatkowo pozwala na zmniejszenie kosztów budowy prototypu.

Pierwsza warstwa systemu, warstwa danych (rys. 6), zawiera biblioteki, które zaimplementowano w kodzie niezarządzanym. Dzięki temu zminimalizowano czas obsługi zdarzeń związanych z kontrolerem i realizowanymi obliczeniami, co jest szczególnie istotne w przypadku systemów czasu rzeczywistego. Druga warstwa została zaimplementowana w kodzie zarządzanym, jako biblioteka typu CLR. Pozwala to dołączyć interfejs do innych aplikacji uruchamianych we wspólnym środowisku uruchomieniowym. Trzecia warstwa sytemu, warstwa aplikacji, została zaimplementowana z wykorzystaniem technologii WPF. Podział na warstwy i moduły umożliwia łatwe dostosowanie systemu do nowych wymagań. Dzięki temu można szybko dostosować system do współpracy z innym kontrolerem ruchu oraz zwiększyć liczbę rozpoznawanych gestów.

Literatura

[1] BOOCH G., RUMBAUGH J., Jacobson I., *Inżynieria oprogramowania. UML przewodnik użytkownika*. WNT, Warszawa, 2001.

[2] *Depth Stream*, MSDN Library, http://msdn.microsoft.com/en-us/library/jj131028.aspx.

[3] *Extraction of skeletons from 3D maps*, `http://www.freepatentsonline.com/y2011/0052006.html.

[4] *Kinect for Windows SDK*, MSDN Library, http://msdn.microsoft.com/en-us/library/hh855347.aspx.

[5] *Kinect Matlab*, MathWorks File Exchange, http://www.mathworks.com/matlabcentral/fileexchange/30242.

[6] *Kinect*, CodePlex, http://www.codeplex.com/site/search?query=kinect&ac=8.

[7] KRAISS K.F., *Advanced Man-Machine Interaction*. Springer-Verlag GmbH, Berlin, 2006.

[8] *Language Features for Targeting the CLR*, MSDN Library, http://msdn.microsoft.com/en-us/library/xey702bw.aspx.

[9] *MATLAB Coder*, MathWorks, http://www.mathworks.com/products/matlab-coder/.

[10] *Microsoft Speech Platform*, MSDN Library, http://msdn.microsoft.com/en-us/library/hh361572.aspx.

[11] *PS1080 system on chip*, PrimeSense Natural Interactions, http://www.primesense.com/en/company-profile/114-the-ps1080.

[12] SIKORSKI M., *Interakcja człowiek-komputer*. Wyd. PJWSTK, Warszawa, 2010.

[13] *Tracking Users with Kinect Skeletal Tracking*, MSDN Library, http://msdn.microsoft.com/en-us/library/jj131025.aspx.

[14] WANG J. I INNI, *3D scene reconstruction by multiple structured-light based commodity depth cameras*, 2012 IEEE International Conference on Acoustics, Speech and Signal Processing, 2012, pp. 5429-5432.

[15] WILKOWSKI A., *A HMM-based system for real-time gesture recognition in movie sequences*, 2008 Conference on Human System Interaction, 2008, pp. 737-742.

[16] WYSOCKI M. I INNI, *Rozpoznawanie gestów wykonywanych rękami w systemie wizyjnym*. Oficyna Wydawnicza Politechniki Rzeszowskiej, Rzeszów, 2011.

Use of the motion and MATLAB environment to build a NUI interfaces

ABSTRACT: The problem of the use of motion controller Microsoft Xbox 360 Kinect and the Matlab environment for prototyping interfaces that allow control by gestures is considered in the paper. The architecture of a created system, including software communication with the motion controller and the Matlab environment, has been described. The way of generating and using C++ code based on Matlab scripts is presented.

KEYWORDS: motion controller, Microsoft Kinect, Matlab Coder, natural user interfaces, NUI

Praca wpłynęła do redakcji: 5.10.2012