STUDIA INFORMATICA 2015
Volume 36 Number 3 (121)

Piotr CZEKALSKI, Marcin GOLENIA, Łukasz LIPKA, Krzysztof TOKARZ
Politechnika Śląska, Instytut Informatyki

USING GESTURE AND VOICE COMMANDS FOR THE TRIBOT ROBOT CONTROL

Summary. Presented project integrates seamlessly modern device control methods into one, solid solution. The Project is in touch-less control algorithm to the robotics, considered as a technology sampler for feature industrial usage. It implements gesture and voice recognition based solution to control the mobile Tribot robot driving over flat, two dimensional surface. It integrates Microsoft Kinect sensor, Lego Mindstorms NXT robot and a PC computer all together. It also provides voice controlled calibration of the human to machine interface.

Keywords: gesture control, mobile robot, speech recognition, NXT, Kinect

INTEGRACJA STEROWANIA GESTAMI I ROZPOZNAWANIA GŁOSU W ZASTOSOWANIACH DO STEROWANIA ROBOTAMI MOBILNYMI

Streszczenie. W dokumencie opisano projekt, w którym zintegrowano nowoczesne metody sterowania bezdotykowego robotem mobilnym przy użyciu gestów oraz rozpoznawania głosu. Przedmiotem sterowania jest robot zbudowany na platformie Lego Mindstorms NXT, poruszający się po dwuwymiarowej przestrzeni. Rozwiązanie integruje sensor Microsoft Kinect do sterowania robotem oraz metodę kalibracji położenia użytkownika za pomocą rozpoznawania komend głosowych.

Słowa kluczowe: sterowanie gestami, roboty mobilne, rozpoznawanie głosu, NXT, Kinect

1. Introduction

Currently, many tasks are performed by autonomous robotic systems, but the remote control is still in charge of most tasks performed by robots, particularly when expected conditions of the operation environment cannot be foreseen thus hard to be held by the regular,

non-adaptable control algorithms. Some of such problems can be held with success by AI-based control algorithms, but all of them are still unable to work as flexibly as humans do. Nowadays the development of gesture based control systems is hitting both the commercial market and military one, i.e. [14][15]. On the other hand, there is a rapid development of the household robotics and interaction methods using advanced techniques integrating various components, i.e. speech recognition, face and hand tracking, pose and pointing posture recognition and others [16]. When it comes to the distant controlling of the mobile robot, usually kind of the joystick or direction pad controller devices were used as a first choice input source. This paper represents a novel method of controlling mobile robots over the 2D surface using speech and gestures. The robot is a popular differential-drive Tribot robot [1] built with Lego Mindstorms NXT robotics development set [7]. The control over robot is achieved both by gestures and voice recognition. The paper describes a method, design, implementation and experiments performed by the utility of the physical devices and appropriate control software.

2. Outline

The system is composed of the three major parts:

- A touch-less input device, capable to record operator's speech and track its skeleton – the Microsoft Kinect sensor [8], connected to the PC through wired, USB connection (requires separate power supply, however).
- A PC computer with a Bluetooth communication device (the NXT robot is capable to be driven both over Bluetooth and USB interfaces, however, for the mobile platform it is better to control robot wirelessly than over wired connection). The Bluetooth communication stack delay impact has been analyzed and presented in the chapter **Communication and timing analysis**.
- LEGO Mindstorms NXT robotics platform – the robot model is a differential-drive Tribot robot [1][2] (see Fig. 1). Tribot is running over 2D surface and operated via two independent NXT Servo Motors [2][9], one per wheel. The wheels are settled directly on the NXT Servo Motor shaft, thus wheel RPM is equal to the NXT Servo Motor RPM.

PC Control Application works in two modes – the Calibration Mode and the Control Mode (including three internal states for setting particular range values). To process to the control mode, it is necessary to perform all calibration steps, as presented below. Calibration and switching between modes is performed by means of the voice recognition – user is expected to provide a fixed set of the words (a grammar). The voice recognition based state machine of the software control is presented on the Fig. 2.

Fig. 1. LEGO Mindstorms NXT Tribot robot
Rys. 1. Robot Tribot, platforma LEGO Mindstorms NXT

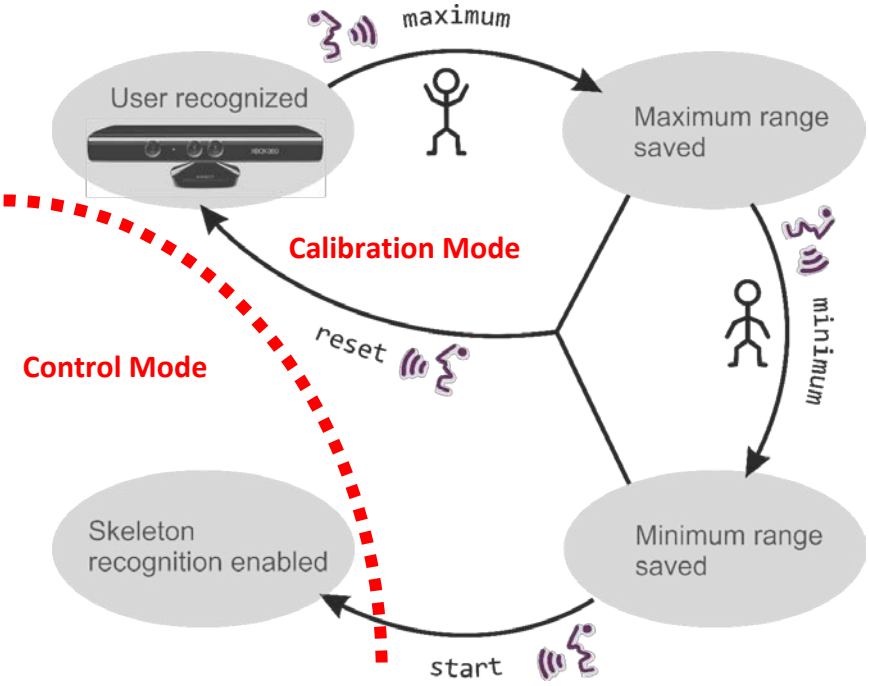

Fig. 2. PC control application – voice recognition driven state machine
Rys. 2. Aplikacja sterująca PC – maszyna stanów sterowana rozpoznawaniem głosu

The system works as follows: user runs the PC application which checks if the Microsoft Kinect and NXT Robot are present and ready to use. If so, the user performs short calibration process determining the area in which user can control the robot to set maximum and minimum levels, so it is capable to fit any user along with its range of movement of its

hands. The zero level is calculated by the application. The voice commands recognized by the application that relate to the calibration mode are: Maximum, Minimum and Reset. When a Start command is recognized, it enables control of the robot motors by transforming position of the hands (according to the zero level) to the Servo Motor power. This process is described in the subsection **Control algorithms**.

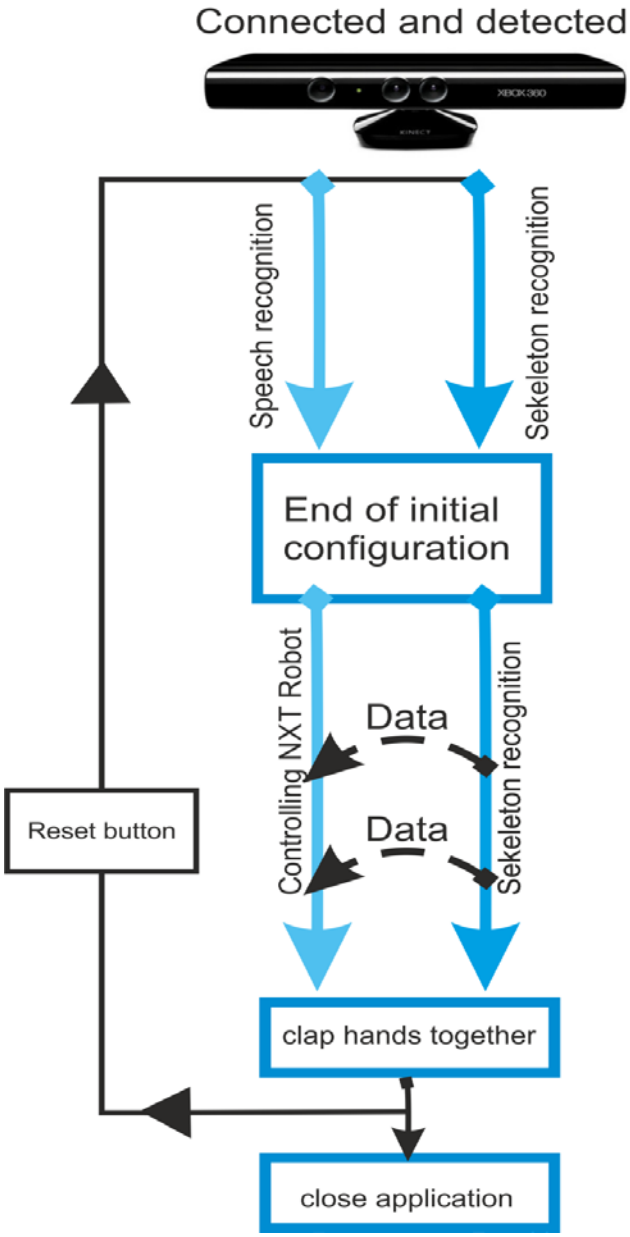

Fig. 3. Voice and gesture recognition control schema
Rys. 3. Sterowanie rozpoznawaniem głosu i gestami

The robot control idea is rather simple – raising hands above zero level increases the value of power being sent to the NXT Servo Motor, analogously lowering the hand below

zero level causes the robot motor receive a negative value of power so the wheel in this case rotates in the opposite direction. In Fig. 4 the mentioned values can be seen together along with current hands flat coordinates. Clapping hands together stops robot control, and enables again speech recognition mode, so the user can start the robot again or just close the application (see Fig. 3).

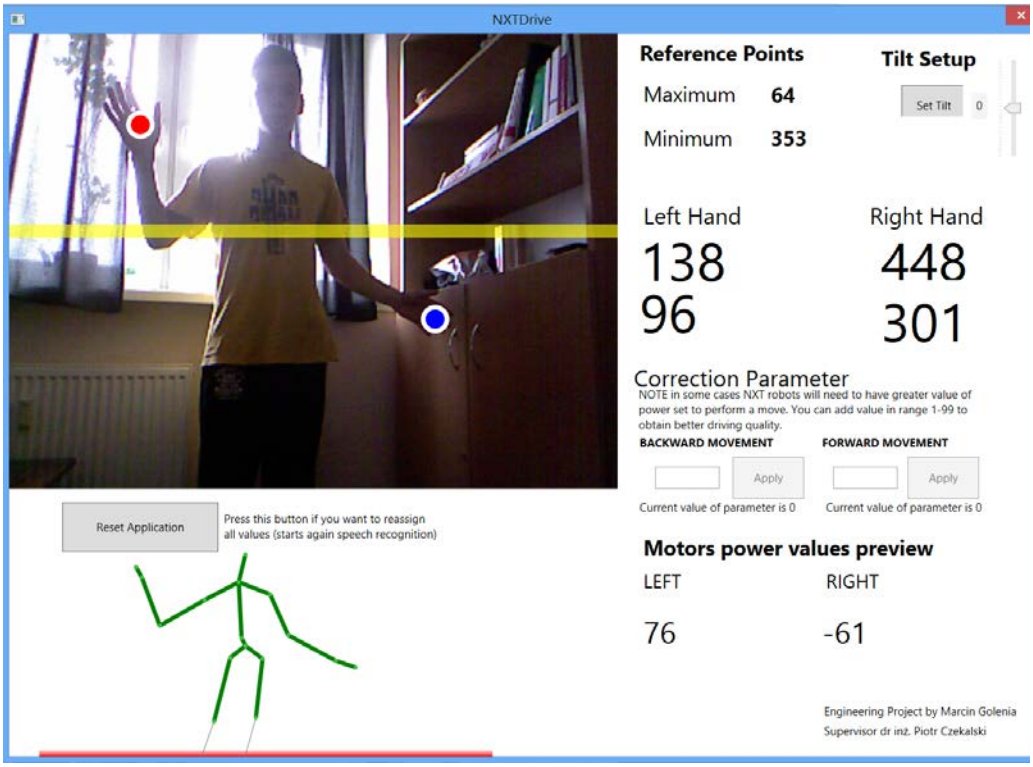

Fig. 4. PC control application
Rys. 4. Aplikacja sterująca, uruchamiana na komputerze PC

3. System Construction

System construction can be split into major hardware and related software parts, as juxtaposed below:

- Kinect – Recognizes the skeleton, sends images from camera and sound from microphone array (also embedded into the Kinect device) to the PC computer. It acts as a touch-less input device, delivering video stream (visible range), skeleton tracking stream - up to two persons simultaneously, audio stream - along with sound source direction. The system uses video stream, skeleton tracking (only the first skeleton in the skeleton collection is taken into the consideration), audio stream for voice recognition and video stream to present the current situation “as seen” by the Kinect sensor to provide valuable feedback to the user and then simplify calibration and robot control.

- PC Computer – is a central point of the system, running the control algorithms. It processes the received data to configure the control environment and send appropriate values to the robot over a Bluetooth connection, using LCP (LEGO Control Protocol) [12].
- Robot – is an output of the system – the NXT Intelligent Brick [13] transforms received data to control the power of the Tribot's [3] two connected NXT Servo Motors [9].
- The software solution was implemented using .NET framework and appropriate libraries, including WPF and hardware-related SDKs:
- Mindsqualls NXT – a powerful library that enables remote management of any NXT robot including Bluetooth and USB connections. This library delivers only remote controlling – it does not provide autonomous programming mode. It provides an interface to control the NXT Intelligent Brick from the PC-running application written in the C# language (actually this library can be mixed with any .NET compatible language, including VB.NET and F#). It delivers LEGO Control Protocol implementation that is a standard for the remote NXT control, enabling management of the NXT Servo Motors, including rotation direction, rotation speed and power (among others) [12].
- Microsoft Kinect SDK – Provides classes to work with the Kinect sensor, including skeleton tracking structures, video and audio streams. It also enables remote tilting of the Kinect sensor - presented solution does not support this feature, but it is very easy to extend the PC application with this feature, however.
- Microsoft Speech Platform – Provides classes to work with speech recognition captured by any Windows audio source, here delivered by Kinect sensor. It is necessary to deliver appropriate grammar to enable recognition of the words.

An application which merges all mentioned technologies was written in WPF (Windows Presentation Foundation) technology using C# and XAML [10], where the interface is separated from the logic part as in MVVM (Model-View-ViewModel) schema of creating rich user interfaces [11]. The user interface was not the priority anyway, so it is kept as simple, as possible - it contains just one form, as presented on the Fig. 4.

The MVVM model enables the ability to quickly change the UI, i.e. two separate forms for the Calibration Mode and the Control Mode with ease. The MVVM model was the natural choice as most of the controls and samples delivered with the Microsoft Kinect Sensor use this schema. It is also considered as industrial and programming standard nowadays as one of the important features of this model is able to test the model separately from the view that enables automated testing of the code [11].

4. Speech Recognition Control Interface

A speech interface is used to calibrate input interface and switch between application modes.

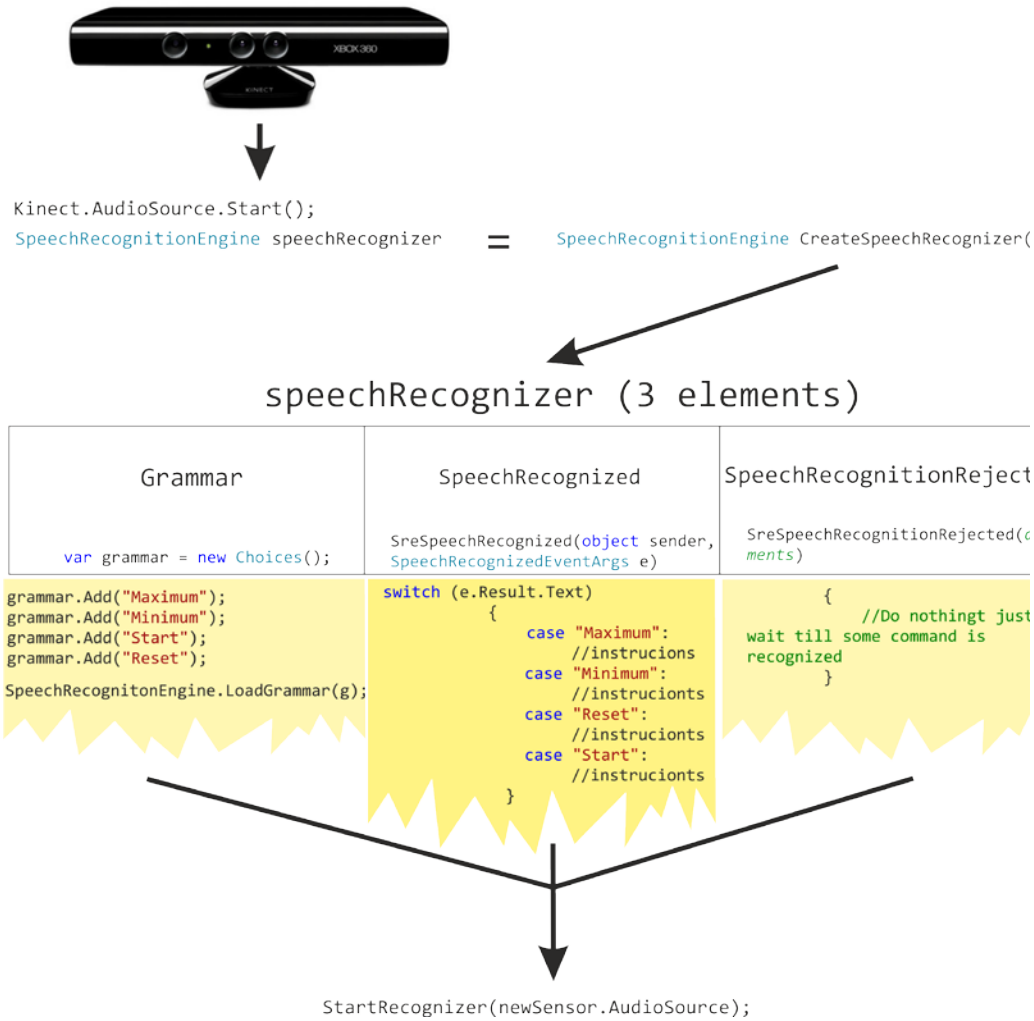

Fig. 5. Szczegóły implementacji zdarzeń rozpoznawania mowy
Rys. 5. Speech recognition events implementation details

The speech recognizer is created in the following way: first an audio stream for the use of Kinect is created and then the speech recognition engine is added to the Kinect. Next a grammar needs to be created and added to the engine. In this case grammar consists just out of only four strings – Minimum, Maximum, Reset, and Start. The speech recognition process together with C# code fragments are presented on the Fig. 5. The speech recognition engine also has only two methods / event handlers implemented that is right enough to deliver the required features in this case:

- *SreSpeechRecognized* occurs when the speech was successfully recognized and contains instructions that the application should do in the next. In this case simple switch structure

was used with four case sections, one for each voice command (*Start, Reset, Maximum* and *Minimum*).

- The second *SreSpeechRecognitionRejected* method occurs when speech was unrecognized. In case of this application this method does nothing – the application simply waits till it recognizes known command. *StartRecognizer* method runs the recognition engine.

As speech recognition interface uses event model, the speech recognition is running asynchronously to the rest of the application [4][8].

5. Gesture Recognition Control Interface

5.1. Gesture Recognition

To begin Gesture recognition creating an object of *Skeleton* class is needed. Kinect is up to track 6 skeletons and it is doing it all the time [4]. Using current SDK release (v.1.5 at the moment of writing this paper) Kinect sensor is capable to track up to two skeletons simultaneously [8] and deliver it as a list of the objects to the programmer – this limitation comes possibly from the limits of the Xbox 360 console, as Kinect for PC and Kinect for XBox do not differ much in functionalities (see **Summary** chapter for more information). The presented application requires only one operator to be present. In case there are more persons present and successfully recognized by the Kinect sensor, only the first one that stands closest to the sensor is in charge of controlling the robot. To work with only one skeleton, an array of Skeleton class objects should be created and then the first one should be returned to work with. It was done using dedicated function named *getFirstSkeleton()* where the skeleton is obtained using simple LINQ query [5]. Having the skeleton assigned to the desired object, further steps could be taken. Using *Skeleton* properties it is necessary to identify appropriate attributes that identify left and right hand location (its X and Y coordinates). This is performed indirectly through depth view delivered by the Kinect (depth is considered as pre-processed video stream here). *DepthImageFrame* is a frame used specifically for depth images. It provides access to coordinates for each dimension (axis), the format and the pixel data of the single depth frame, as seen by the sensor and allows for mapping of coordinates between *Skeleton* frames and *Color* frames. Within this access, object of structure *DepthImagePoint* can be obtained. It represents X, Y axis coordinates of the interested skeleton point over the video frame (see Fig. 6 for the coordinates system explanatory) and the depth value in the depth frame structure that represent the Z axis – a third dimension [6]. Unfortunately for the developers, the functionality of the Z-axis representing the depth value are very limited here, as it provides only a discrete set of the values: near, medium and far. This limitation is subject to be changed in future releases of the Kinect SDK. Two *DepthImagePoint* variables are

used – one for left hand and the other for right one. To get them a SDK's *MapFromSkeleton-Point()* function is used in which case the type Joint is specified. Joint coordinates of the hands are used to calculate values of power for robot motors and to identify "clap hands" gesture that finishes Control Mode of the PC application.

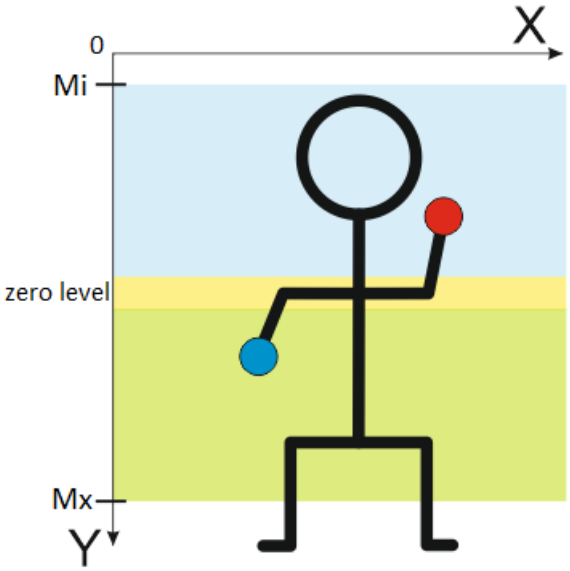

Fig. 6. Coordinates (note that Y-axis values are increasing downwards)
Rys. 6. Układ współrzędnych (uwaga – wartości osi Y wzrastają w dół)

5.2. Control Algorithms

Once the user has defined the maximum and minimum values that denotes the range of one's hands, then said start, a zero level z (observe yellow strip presented on Fig. 4 and Fig. 6) value is calculated and presented to the operator, using simple averaging formula Eq. 1:

$$Z = \frac{Mi - Mx}{2} + Mx \tag{1}$$

where *Mx* denotes the maximum value of the Y-axis coordinate, *Mi* denotes minimum Y-axis coordinate value.

The theoretical relation between the NXT Interactive Servo Motor rotation speed and power setting is considered to be linear versus power setting provided by the controller with no stall. This approach is a theoretical one with no respect to the hardware characteristics and physical parameters of the NXT Interactive Servo Motor - this relation is true for the unloaded motor. So far some computation has to be performed on those values to drive motors as NXT Interactive Servo Motor under the load (the robot's weight) present stall area even up to 40% of the base power [9]. Once started, the rest of the curve still can be approximated as linear however [9]. This leads to the introduction of the linear correction coefficient in the following Eq. 2 and Eq. 3. This parameter was estimated by the tests as observed the mini-

mum power value delivered to the robot to achieve slight movement. In any case the robot construction is such the +100% of the power is the full forward rotation with the maximum speed, and -100% is backwards. The conversion of the Y-axis coordinates of a hand with the motor power values expressed in % (from -100 to 100) is to be performed as the main loop of the control algorithm. It occurs whenever checking the user's hand is above or below zero level area otherwise robot stops. In case the *Y* coordinate is located within the area below the zero level, the raw motor power *Mpr* delivered to the motor is calculated using formula, (2) otherwise using formula (3):

$$Mpr = (-1) \cdot \frac{(Hy - Z \cdot 1.05) \cdot 100}{Mi - 1.05 \cdot Z} - cP \tag{2}$$

$$Mpr = \frac{(0.95 \cdot Z - Hy) * 100}{0.95 \cdot Z - Mx} + cP \tag{3}$$

where *Hy* stands for the current Y-axis reading of the hand, *z* – Y-axis value zero level, *Mi* – Y-axis maximum value (as calibrated in Calibration Mode, see Fig. 6), *Mx* – Y-axis minimum value (as calibrated in Calibration Mode, see Fig. 6), *cP* – linear correction coefficient (obtained from the set of the tests performed on the real system). The raw motor power is calculated independently for each motor: *LMpr* and *RMpr* for left and right motor, respectively. Each motor is controlled with one hand and is subject to further correction by the Acceleration Restriction and Gyration Speed Restriction as presented in the chapters 5.3 and 5.4. To support the robot's physical limitation on curving and acceleration.

5.3. Acceleration Restriction

Experiments performed, presented necessity to introduce Acceleration Restrictions. The Kinect sensor delivers position of the joints that provide data to the control algorithm on all three dimensions. During work user is requested to remain in more-less stable position, operating by hands only. Due to the limited resolution of the Kinect's depth camera that delivers base information to calculate joint position it is essential to maintain the motor power range between -100 and +100 for both *LMpr* and *RMpr* (in general: *Mpr*). There are various situations those values can be over exceeded i.e. when user moves closer to the sensor or rises hand higher than during initial arms range calibration. To limit the value of the *Mpr* (a power range *LMpr* and *RMpr* for left and right motors respectively), simple cutoff procedure has been introduced (4) by the conditional statement, to obtain limited *CMpr* (*LCMpr* and *RCMpr* for left and right motors respectively) values out of the range given by the endpoint +100 and -100 to the correct [-100…100] interval:

$$CMpr = \begin{cases} sign(Mpr) \cdot 100 & when\ abs(Mpr) > 100 \\ Mpr & otherwise \end{cases} \tag{4}$$

5.4. Gyration Speed Restriction

While controlling the Tribot robot, sharp turns may cause it to fall due to its height. This may happen whenever there is a large rotational speed difference between motors, i.e. they are running full speed in opposite directions to cause the robot to pivot with high angular speed as when user rises one hand maximum upwards (*Mx*) while holding the other one maximum downwards (*Mi*) as on Fig.6.

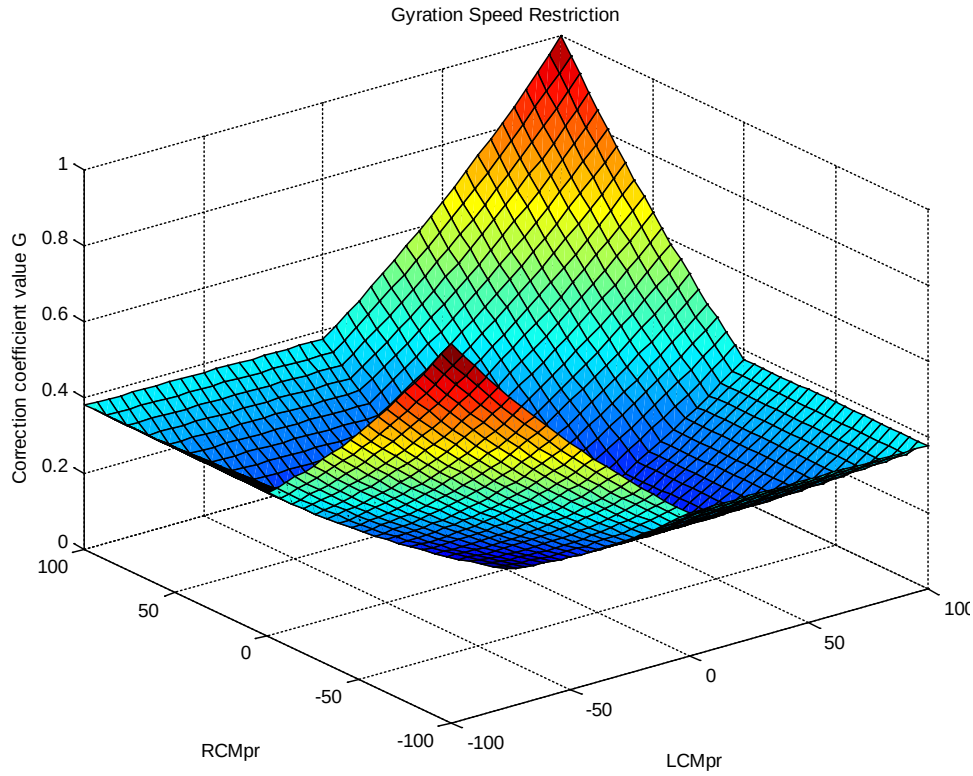

Fig. 7. Gyration Speed Restriction correction coefficient *G*
Rys. 7. Współczynnik korekcji *G* dla ruchu obrotowego

This situation requires limitation on the rotation speed. On the other hand, when the user holds both hands up or down, the control algorithm should let the robot travel forward or backwards with full speed. To target this problem, a simple Gyration Speed Restriction has been introduced that calculated correction coefficient for both engines, lowering their rotation angular speed when necessary. The algorithm bases on the comparison of the absolute ΔM (5) and relative Δm (6) rotation speed of both engines:

$$\Delta M = \left| LCMpr - RCMpr \right| \tag{5}$$

$$\Delta m = \left| LCMpr \right| + \left| RCMpr \right| \tag{6}$$

The correction coefficient is given by the following formula (7):

$$G = \frac{e^{\frac{\Delta M + \Delta m}{400}} - 1}{e - 1} \tag{7}$$

The Fig. 7 presents correction coefficient *G* vs normalized hand position on [-100…100] interval. One may observe that when the rotation speed, direction of both motors contradict, the motor power is reduced to about 38% of the maximum power to prevent robots from losing its stability while when rotation speeds agree, the robot is able to operate on full forward or full backward speed.

5.5. Communication and Timing Analysis

In case of the remote control of the mobile device, it is necessary to analyze the timings of the commands sent to the robot with respect to the accuracy and response from the physical device, i.e. to stop the robot immediately, when the obstacle is seen by the operator. So how much further will robot go, after the initial gesture of stopping the robot was presented by the operator, before it stops physically?

The Kinect is connected to the PC computer so the delay of the USB communication, image processing of the Kinect drivers and PC application is subject of almost no impact to the solution, assuming a modern PC computer is used. The major impact on the command route is Bluetooth communication between PC and the robot.

The experiments performed by the research team present that PC application are capable of sending no less than one command per each 25ms. As LCP ensures delivery and execution of the command, it is obvious that the worst control case occurs when driving robot full speed forward, then issuing the stop command immediately after the full speed run. To stop both engines the time of the 50ms is necessary (two NXT Interactive Servo Motors, one command per each).

The maximum rotation speed of the NXT Interactive Servo Motor is about 160rpm when unloaded [9]. As robot uses direct drive (1:1 ratio), the maximum driving speed when using regular tires (5.6cm diameter) is about 0.4691445 m/s so in every 50ms robot is capable to travel about 2.35cm after the command is issued, before robot stops. This value should be increased a bit because of the robot inertia.

5.6. Summary

Presented solution was successfully implemented. Delays were almost unnoticeable and driving precision better than expected, so this solution control method and software may be employed and extended in many robotic platforms. This solution also applies to the caterpillar based robots directly, however a slight correction of the *cP* parameter (see Eq. 2 and Eq.

3) may be required then as the weight and center of gravity and the weight distribution of the robot changes along with the construction.

Interestingly, the authors of the paper found that both PC Kinect and Xbox Kinect devices present the same set of functions as seen through the Kinect SDK for PC computers, thus both may be used interchangeably.

BIBLIOGRAPHY

1. Benedettelli D.: Creating Cool MINDSTORMS NXT Robots. Apress, 2008.
2. Kelly J. F.: Advanced NXT: The Mayan Adventure. Apress, 2006.
3. Perdue D. J.: The Unofficial LEGO Mindstorms NXT Inventor's Guide. No Starch Press, Syngress, 2007.
4. Webb J., Ashley J.: Beginning Kinect Programming with the Microsoft Kinect SDK. Apress 2012.
5. Cisek J.: Tworzenie nowoczesnych aplikacji graficznych. Helion, Gliwice 2012.
6. Grzejszczak T., Mikulski M., Szkodny T., Jędrasiak K.: Gesture Based Robot Control. [w:] Proceedings of International conference Computer vision and graphics. ICCVG 2012, Warsaw, Poland, September 24-26, 2012., p. 407÷413, Lecture Notes in Computer Science; vol. 7594 Springer, Berlin 2012.
7. The LEGO Group: LEGO Mindstorms official website: http://mindstorms.lego.com, 2012.
8. Kinect for Windows website: http://www.microsoft.com/en-us/kinectforwindows/, Microsoft Corporation, 2013.
9. Hurbain P., NXT motor characteristics. http://www.philohome.com/nxtmotor/ nxtmotor.htm, 2013.
10. MacDonald M.: Pro WPF in C# 2010: Windows Presentation Foundation in .NET 4. 3rd edition, Apress, 2010.
11. Yosifovich P.: Windows Presentation Foundation 4.5 Cookbook, Packt Publishing, 2012.
12. The LEGO Group: Bluetooth Developer Kit. http://mindstorms.lego.com/en-us/support/files/default.aspx, 2013.
13. The LEGO Group: NXT Intelligent Brick. http://mindstorms.lego.com/en-us/products/default.aspx#9841, 2013.
14. 3Gear: 3Gear gesture PC control. http://www.threegear.com/, 2013.
15. Controlling robots using wrist band – MYO. https://developer.thalmic.com/apply/, 2013.

16. Stiefelhagen R., Fugen C. et al.: Natural human-robot interaction using speech, head pose and gestures. IEEE/RSJ International Conference on Intelligent Robots and Systems, Sendai, Japan, 2004, p. 2422÷2427, vol. 3, 2004.

Omówienie

W publikacji przedstawiono spójną technologię sterowania modelem robota mobilnego, skonstruowanego jako przykładowe rozwiązanie, pozwalające przetestować nowoczesne interfejsy użytkownika, obejmujące przekazywanie poleceń w sposób bezdotykowy oraz z wykorzystaniem mechanizmów rozpoznawania mowy. Docelowo opisana technologia może znaleźć zastosowanie w rozwiązaniach przemysłowych. Projekt integruje różnorodne technologie w jedno spójne rozwiązanie, wykorzystując możliwości urządzeń przetwarzających obraz w gesty sterujące oraz dodatkowo jest uzupełniony o funkcje pozwalające na rozpoznanie wydawanych przez użytkownika komend głosowych kontrolujących działanie systemu. W artykule opisano zastosowanie sterowania za pomocą gestów do kontrolowania poruszania się robota mobilnego w dwuwymiarowej przestrzeni płaskiej wraz z uwzględnieniem zjawisk fizycznych, takich jak tarcie, uślizg oraz konieczność stabilizacji poruszania się robota w trakcie pokonywania łuków. Urządzenie integruje w spójną całość technologie Microsoft Kinect (sensor wejściowy sterowania bezdotykowego), Lego Mindstorms NXT (implementację modelu robota) i komputer PC (rys. 2). Wykorzystując gesty wykonywane przez użytkownika (rys. 6), wydawane są komendy sterujące mobilnym robotem trójkołowym klasy Tribot, podczas gdy komendami głosowymi dokonywana jest kalibracja interfejsu użytkownika oraz zmiana trybów pracy algorytmu kontrolującego (rys. 2 i 3). Artykuł przedstawia algorytm sterowania w postaci równań, opisujących sterowanie mocą silników w funkcji gestów użytkownika (równania (1), (2), (3)) wraz z funkcjami zapewnienia płynnego przyspieszania oraz stabilności robota podczas pokonywania łuków z wykorzystaniem współczynnika ograniczenia prędkości ruchu obrotowego opisanego równaniami (4), (5), (6), (7) (rys. 7). Model obejmuje również wprowadzone niezbędne korekcje wynikające z przeprowadzonych doświadczeń w świecie rzeczywistym, w którym należało uwzględnić m.in. tarcie i interakcje pomiędzy elementami systemu. Zaprezentowane rozwiązanie zostało z sukcesem zaimplementowane i przetestowane w laboratorium ZMiTAC Instytutu Informatyki Politechniki Śląskiej.

Address

Piotr CZEKALSKI: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16, 44-100 Gliwice, Polska, piotr.czekalski@polsl.pl.

Marcin GOLENIA: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16, 44-100 Gliwice, Polska, marcingolenia@gmail.com.

Łukasz LIPKA: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16, 44-100 Gliwice, Polska, lukasz.lipka@polsl.pl.

Krzysztof TOKARZ: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16, 44-100 Gliwice, Polska, krzysztof.tokarz@polsl.pl.