Paweł PAWŁOWSKI, Michał JESKE, Szymon KWIATKOWSKI, Natalia KOŁODZIEJCZYK

Politechnika Poznańska, Wydział Automatyki, Robotyki i Elektrotechniki, Instytut Automatyki i Robotyki, Zakład Układów Elektronicznych i Przetwarzania Sygnałów

doi:10.15199/48.2021.02.13

Zrobotyzowane mapowanie przestrzeni z wykorzystaniem czujnika LIDAR

Streszczenie. *W pracy zaprezentowano technikę zrobotyzowanego mapowania przestrzeni. Do testów zaprojektowano i wykonano dwukołowy nieholonomiczny robot mobilny z napędem różnicowym i jednym kołem samonastawnym. Robot, skanując otoczenie za pomocą czujnika LIDAR, umożliwia generowanie dwuwymiarowych map pomieszczeń oraz trójwymiarowych modeli otoczenia. Przedstawiono przykładowe mapy otoczenia uzyskane podczas testów w trybie sterowania ręcznego oraz pracy autonomicznej.*

Abstract*. The paper presents a technique of robotic space mapping. A two-wheeled nonholonomic mobile robot with a differential drive and a self-adjusting wheel was designed and manufactured for testing. The robot scans the environment using the LIDAR sensor and generates two-dimensional maps of rooms and three-dimensional models of the surrounding. Generated maps of the environment obtained during tests in manual control mode and autonomous operation mode are presented. (**Robotic space mapping using LIDAR sensor**).*

Słowa kluczowe: LIDAR, robot mobilny, mapowanie przestrzeni, SLAM.
Keywords: LIDAR, mobile robot, space mapping, SLAM.

Wprowadzenie

Roboty mobilne mają zastosowanie w wielu obszarach gospodarki. Do głównych ich zadań należy wykonywanie prac ciężkich i monotonnych (np. transport towarów w magazynach), prac w środowisku o wysokiej szkodliwości lub wykonywanych w warunkach ekstremalnych, bądź niebezpiecznych dla człowieka. Jednym z tych zastosowań, rozpatrywanym w niniejszej pracy, jest rozpoznawanie terenu i sporządzanie map przestrzeni w otoczeniu nieprzyjaznym dla człowieka: w ciemności, w pomieszczeniach trudnodostępnych, zagrożonych zawaleniem lub wybuchem (np. po katastrofach budowlanych), zadymionych oraz w takich, w których znajdują się substancje trujące. W przypadkach zagrożenia zrobotyzowane przygotowanie mapy otoczenia może usprawnić akcję ratowniczą bez narażania dodatkowych osób na rozpoznanie terenu.

Do tworzenia map terenu można wykorzystać czujniki wizyjne lub dalmierze. Czujniki wizyjne umożliwiają szybkie zobrazowanie dużego obszaru w przestrzeni, ale pomiary odległości wymagane do wyznaczania map otoczenia są utrudnione, wymagają kalibracji, skomplikowanych obliczeń i są obarczone błędami. Z kolei dalmierze realizują dokładne pomiary punktowe, ale odwzorowanie przestrzeni wymaga bardzo wielu pomiarów, co istotnie wydłuża czas skanowania.

Najprostsze techniki pomiaru odległości polegają na wykorzystaniu ultradźwiękowych czujników odległości (należą do nich np. samochodowe czujniki parkowania), umożliwiając pomiar z dokładnością centymetrową w zakresie do około 15 metrów. Zdecydowanie dokładniejsze są czujniki laserowe, które umożliwiają pomiary w dużo większym zakresie (nawet kilkuset metrów) i z większą dokładnością (milimetrową i mniejszą). Specjalizowane dalmierze laserowe są stosowane nawet do pomiaru obiektów na orbicie Ziemi (ang. satellite laser ranging). Niestety, są one dużo większe i zdecydowanie droższe.

Użyty w niniejszej pracy czujnik LIDAR (ang. light infrared detection and ranging) plasuje się w grupie cenowej, oferowanej dokładności i zasięgu pomiędzy dalmierzami ultradźwiękowym a laserowymi. Zasada działania jest zbliżona do dalmierza laserowego. Czujniki te charakteryzują się kompaktową budową i zakresami pomiarowymi od kilkunastu do kilkuset metrów przy dokładności centymetrowej [1, 8]. Znajdują one zastosowanie do pomiarów odległości przy równoczesnej lokalizacji i mapowaniu otoczenia (SLAM, ang. simultaneous localization and mapping) [2].

W pracy [2] autorzy za pomocą czujnika LIDAR udokładniają pomiary uzyskane z kamery stereowizyjnej. W pracy [3] pomiary czujnikiem LIDAR umożliwiły robotowi unikanie kontaktu z przeszkodami i skanowanie przestrzeni w dwóch wymiarach (2D). Skanowanie przestrzeni trójwymiarowej (3D) zrealizowano na platformie Intel NUC, LiDAR 2D z kamerą RGB-D (z wbudowanym skanerem przestrzeni trójwymiarowej). Czujnik LIDAR wykorzystano również w komercyjnym niewielkim robocie mobilnym [4]. Podobnie, autor pracy [5] do nawigacji robota mobilnego wykorzystał dalmierz pracujący z paśmie podczerwieni.

W niniejszej pracy zaprezentowano technikę zrobotyzowanego mapowania przestrzeni. Do testów zaprojektowano i wykonano dwukołowy nieholonomiczny robot mobilny, z napędem różnicowym i jednym kołem samonastawnym. Robot, poruszając się, umożliwia skanowanie otoczenia za pomocą czujnika LIDAR z równoczesną generacją dwuwymiarowych map pomieszczeń oraz trójwymiarowych modeli otoczenia z wykorzystaniem techniki SLAM. Robot może poruszać się w trybie sterowania ręcznego lub pracy autonomicznej. Przedstawiono konstrukcję robota i opis jego oprogramowania w środowisku ROS (ang. robot operating system) oraz przykładowe mapy wygenerowane przez oprogramowanie robota.

Czujnik LIDAR

LIDAR jest optycznym miernikiem odległości zaliczanym do kategorii czujników ToF (ang. time-of-fligt), dokonujących pomiaru czasu propagacji fali. Składa się on z modułu generującego wiązkę światła oraz teleskopu. Źródłem światła jest dioda półprzewodnikowa (laserowa), typowo wytwarzająca falę z zakresu podczerwieni. Wysyła ona krótkie impulsy światła w określonym kierunku. Światło to, po dotarciu do celu, ulega odbiciu i rozproszeniu. Promienie odbite są obserwowane za pomocą teleskopu, a następnie rejestrowane poprzez detektor, który bada przesuniecie fazowe między falą wysłaną a odebraną oraz dodatkowo natężenie odebranego sygnału. Odległość między czujnikiem a obiektem mierzonym można wyliczyć ze wzoru (1)

$$D = \frac{c}{2} \cdot \frac{T\Delta\varphi}{2\pi} \tag{1}$$

gdzie: D – odległość, c – prędkość propagacji fali świetlnej, T – okres wysyłanej fali światła, $\Delta\varphi$ – przesunięcie fazowe pomiędzy sygnałem nadawanym a odebranym [7, 8].

Aby wygenerować mapę otoczenia należy dokonać pomiarów odległości czujnika LIDAR do obiektów w otoczeniu zmieniając kąty pomiarowe w przestrzeni. W tym celu wykonano w technologii druku 3D głowicę umożliwiającą obrót czujnika LIDAR w dwóch płaszczyznach. Do obrotu głowicy wykorzystano dwa serwomechanizmy modelarskie SG90. Ich zaletą jest przede wszystkim mały rozmiar i dokładność pozycjonowania. Użyte serwomechanizmy pracują w zakresie kątów 180º oraz charakteryzują się momentem 2,5 kg·cm. Ponieważ wstępne badania serwomechanizmów wykazały, że ich charakterystyka odbiega od danych z noty katalogowej producenta, dlatego każdy serwomechanizm został indywidualnie skalibrowany.

Wykonanie pełnego zakresu ruchu serwomechanizmu, odpowiedzialnego za obrót w poziomie, wymaga ustawienia jego wału w pozycji 0º, a wału serwomechanizmu odpowiedzialnego za obrót w pionie, w pozycji 180º, po czym obraca się stopniowo czujnik w poziomie. Tym sposobem możliwe jest wykonanie pełnego skanu w zakresie 360º na jednym poziomie (skanu 2D). Procedura skanu 3D przebiega w podobny sposób: skany na jednym poziomie są wykonywane jako skany 2D, ale po każdym skanie nachylenie w pionie zwiększane jest o 1 stopień. Zmiana o 1 stopień jest realizowana do kąta 45º. Powyżej tego kąta pomiary są wykonywane co 2 stopnie. Dzięki temu pełen skan zostaje wykonany szybciej, przy wystarczającej gęstości pomiarów. Zastosowany czujnik LIDAR TFmini Plus [8] umożliwia wykonywanie pomiarów z maksymalną szybkością 1000 pomiarów na sekundę. Jednak im częściej są dokonywane pomiary, tym są obarczone większymi błędami [6, 8]. Po przeprowadzeniu szczegółowych badań dokładności pomiarów zastosowanego czujnika LIDAR, ustalono jego szybkość na 250 pomiarów na sekundę. Stanowiło to kompromis miedzy dużą powtarzalnością odczytów (dokładnością), a możliwie krótkim czasem wykonania pełnego pomiaru 2D lub 3D. Tak dobrana szybkość skanowania wykorzystuje równocześnie maksymalną prędkość obrotową serwomechanizmów. Dalsze zredukowanie liczby pomiarów nie zmniejszyłoby więc czasu skanowania przestrzeni, a zmniejszyłoby gęstość punktów pomiarowych.

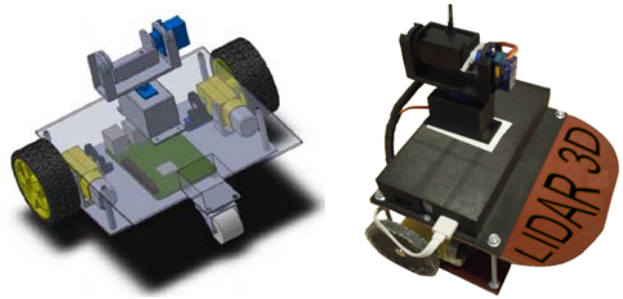
Po przeliczeniu, efekt końcowy stanowią punkty odzwierciedlające położenie powierzchni skanowanych obiektów w otoczeniu, zwane potocznie chmurą punktów. Ostatecznie skanowanie otoczenia 2D zajmuje 3,7 sekundy (z rozdzielczością 0,5º), a 3D wymaga 101 sekund (z rozdzielczością w pionie i poziomie 1º i zakresie w pionie od 0 do 45º; pomiar 3D może być wykonywany do dowolnego kąta nachylenia od podłoża zadanego przez użytkownika.)

Robot mapujący przestrzeń

Zaprezentowaną głowicę z czujnikiem LIDAR, umożliwiającą skanowanie przestrzeni umieszczono na specjalnie zaprojektowanym robocie mobilnym. Konstrukcję nośną robota zaprojektowano w programie SolidWorks. Platforma robota została wykonana w technice druku 3D na wzór typowego dwukołowego robota nieholonomicznego z jednym kołem swobodnym z przodu (rys. 1). Konstrukcja ta umożliwia stabilne prowadzenie pojazdu przy pomocy dwóch kół napędowych i napędu różnicowego.

Wymiary robota zostały dobrane w taki sposób, aby wszystkie pozostałe komponenty (poza głowicą czujnika LIDAR i baterią zasilającą), zmieściły się w podstawie robota, a jednocześnie zajmowały jak najmniejszą przestrzeń.

Do napędu robota wykorzystano dwa silniki prądu stałego. Wybrano silniki modelarskie o zintegrowanej przekładni 1:120. Sterowanie silnikami realizuje wzmacniacz mocy TB6612 w układzie mostka H. Wyposażony jest w tranzystory MOSFET o $R_{DS(on)}$ = 0,5 Ω. Prędkość obrotowa silników ustalana jest za pomocą modulacji PWM.

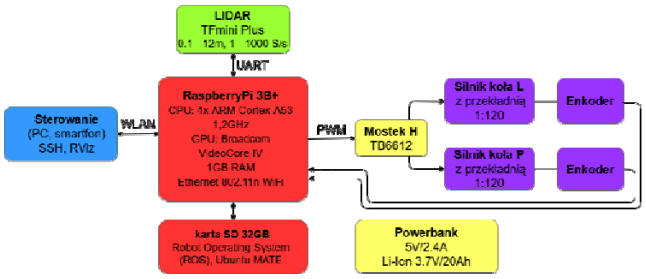


Rys. 1. Trójwymiarowy model robota wykonany w programie SolidWorks i zrealizowana konstrukcja z czujnikiem LIDAR

Sterownik robota i oprogramowanie

Moduł sterujący robota zrealizowano z wykorzystaniem platformy Raspberry Pi 3B+. Zawiera on układ SoC Broadcom BCM2837 (CPU: 4x ARM Cortex-A53, 1,2GHz, GPU: Broadcom VideoCore IV), 1GB pamięci RAM, interfejsy sieciowe 10/100Mbit/s Ethernet, 2,4GHz 802.11n WiFi. Oprogramowanie zapisano na karcie MicroSDHC 32GB.

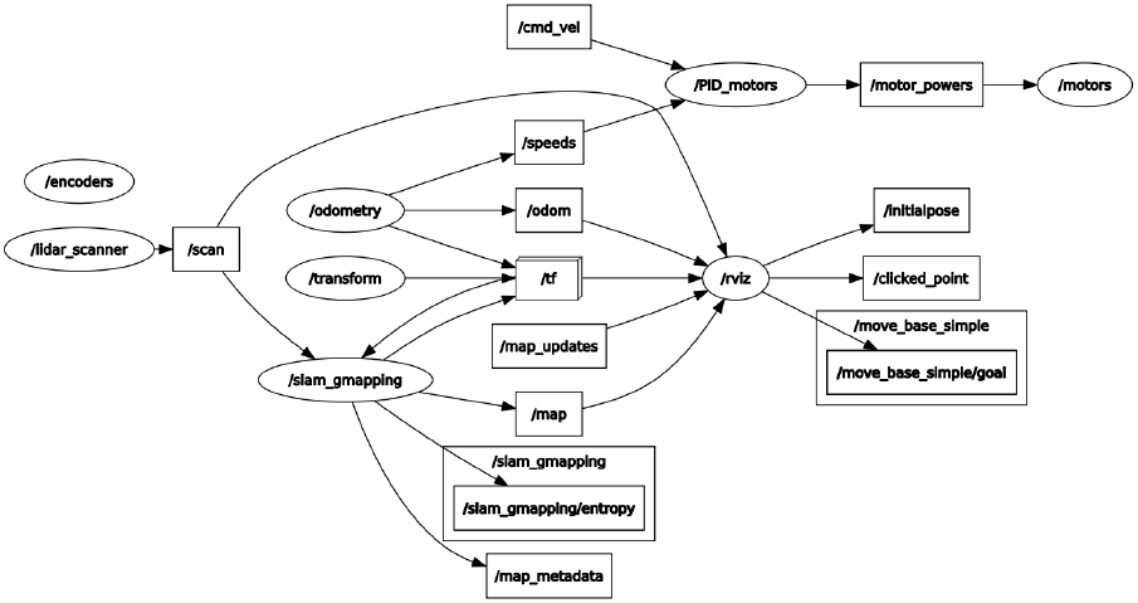
Sterownik robota realizuje kilka zadań. Steruje on pracą czujnika LIDAR i serwomechanizmami ustalającymi kąty pomiaru, odczytuje wyniki pomiarów, wyznacza mapy otoczenia, a także może dodatkowo wysyłać wyniki przez interfejs sieciowy do dalszego przetworzenia. Realizuje także napęd robota: odbiera rozkazy sterujące napędem i przekształca je w sygnały kontrolujące silniki napędowe. Obsługuje także czujniki pomiarowe (enkodery) umieszone na kołach robota wykorzystywane w odometrii. Schemat blokowy robota przedstawiono na rys. 2.



Rys. 2. Schemat blokowy robota skanującego przestrzeń

Programy realizujące powyższe funkcje pracują pod kontrolą platformy ROS w wersji „Melodic Morenia”, która jest dedykowana do programowania robotów. Platformę ROS zainstalowano na systemie operacyjnym Ubuntu MATE. Oprogramowanie ROS, mimo mylącej nazwy, nie jest systemem operacyjnym, tylko platformą programistyczną do oprogramowywania robotów. Jest zbiorem narzędzi, bibliotek i założeń, które upraszczają pisanie złożonych programów sterujących robotem, niezależnie od wybranej platformy. ROS umożliwia również sprawne pisanie programów niskopoziomowych do obsługi sprzętu, a także przekazywanie wiadomości między procesami i prezentację graficzną połączeń między napisanymi programami w formie grafu (rys. 3). Programy w

ROS można pisać w różnych językach (np. C++, Python), używając do tego specjalnych bibliotek, takich jak roscpp czy rospy. Użytkownik może również posłużyć się gotowymi paczkami programów, napisanymi przez innych programistów. Kody źródłowe bibliotek wydanych na licencji BSD (ang. Berkeley Software Distribution) można edytować, dopasowując do własnych potrzeb.



Rys. 3. Graf struktury oprogramowania ROS robota mapującego przestrzeń [6]

Podstawowymi jednostkami programowymi w ROS są węzły (ang. nodes). Zazwyczaj są to pojedyncze programy, mające sprecyzowane działanie (na rys. 3 przedstawione jako elipsy). Węzły mogą komunikować się asynchronicznie lub synchronicznie.

Komunikacja asynchroniczna odbywa się poprzez interfejs publish/subscribe, który polega na tym, że węzły deklarują subskrypcję lub publikowanie wiadomości na pewien temat (ang. topic, prostokąty na rys. 3). Kierunek przesyłania wiadomości jest zaznaczony strzałką. Wiadomości składają się z jednej lub większej liczby zmiennych typu podstawowego, tablic lub innych wiadomości. Informacje przesyłane są za pomocą protokołu TCPROS, który jest oparty na stosie TCP/IP. Taki sposób przesyłania zapewnia modułowość projektu i umożliwia łatwą rozbudowę.

Komunikacja synchroniczna jest opisana przez interfejs klient/serwer. Oferuje ona komunikację pojedynczych węzłów i działa na zasadzie pytanie-odpowiedź. Wymiana informacji tą metodą opiera się na usługach (ang. services). W ROS usługi są zdefiniowane jako para wiadomości: jedna jest pytaniem, druga odpowiedzią. Wybrany węzeł oferuje usługę o pewnej nazwie, inne węzły żądają danego serwisu poprzez wysłanie wiadomości i oczekują na odpowiedź. Po otrzymaniu wiadomości zwrotnej kontynuują działanie programu.

Opracowana struktura oprogramowania wykorzystuje kilka paczek programów. Najważniejszą z nich jest slam_gmapping. Zawiera ona kilka bloków. Węzeł odometry realizuje odometrię, wykorzystując do tego dane zapewnione przez usługę węzła encoders. Odometria jest obliczana z okresem 100 ms na podstawie średniego czasu pomiędzy zmianami stanów zastosowanego czujnika szczelinowego umieszczonego na kołach napędowych. Program encoders wykrywa przejścia ze stanu wysokiego na niski oraz z niskiego na wysoki, co przy zastosowanym sprzęcie daje 40 przejść stanów na pełen obrót koła. Obliczone prędkości obrotowe kół są publikowane w temacie /speeds, a pozycja względem punktu początkowego w temacie /odom oraz w specjalnym temacie /tf zawierającym wszystkie transformaty między układami robota. Realizuje je węzeł transform, który ze zdefiniowaną częstotliwością umieszcza w temacie /tf transformatę między układem bazowym robota a układem czujnika LIDAR.

Węzeł lidar_scanner jest odpowiedzialny za komunikację z czujnikiem LIDAR. Subskrybuje on tematy /is_moving oraz /make_3DScan. W pierwszym z nich przesyłana jest wiadomość informująca o tym, czy robot się porusza – w tym wypadku skany nie są przesyłane po to, aby uniknąć błędów spowodowanych przesunięciem bazy. Z kolei, w przypadku pojawienia się wiadomości, wykonywany jest skan 3D do zadanego kąta nachylenia od podłoża. Wyniki pomiarów 2D są przesyłane na kanale /scan w formacie LaserScan. Jest to format danych opracowany specjalnie dla czujników typu LIDAR. Mapy 3D są publikowane jako chmura punktów na kanale /map3D.

Robot skanujący oferuje możliwość autonomicznego wykonywania map otoczenia. Algorytm autonomicznego mapowania jest zawarty w węźle autonomus_path_requester. Subskrybuje on tematy: /map, /move_base, /status i /auto_map_start. Po uruchomieniu, przeszukuje mapę w celu znalezienia wolnych punktów graniczących z punktami, o których nie ma informacji. Zaznacza je na drugiej mapie, a reszta punktów pozostaje niezmieniona. W przypadku znalezienia takich miejsc, kreuje podwójną listę, w której są zawarte ich indeksy i odległości od robota. Lista jest sortowana według odległości, a następnie algorytm próbuje wytyczyć trasę do najbliższego miejsca. Jeśli okazuje się to niemożliwe, próbuje z kolejnymi miejscami z listy. Po znalezieniu punktu, o którym posiada ograniczone informacje, wysyła wiadomość /move_base_simple/goal, w celu przemieszczenia się do kolejnego punktu. Algorytm jest powtarzany, dopóki użytkownik nie prześle wiadomości wyłączającej robota na kanale /auto_map_start.

Wybór trybów pracy jest realizowany w węźle user_interface. Pozwala on uruchomić skanowanie 3D, włączyć lub wyłączyć automatyczne mapowanie, a także zapisać wyznaczoną mapę 2D w formacie. pgm lub .yaml, oraz zapisać ostatni skan 3D w formacie .pcd. Zmiana trybów wykonywana jest poprzez wysyłanie odpowiednich wiadomości, a zapis − poprzez chwilowe uruchomienie programu map_saver z paczki map_server dla mapy 2D lub programu pointcloud_to_pcd z paczki pcl_ros dla mapy 3D. Ponieważ węzeł pointcloud_to_pcd jako wejścia wymaga formatu PointCloud2 a program laser_scan publikuje skany w formacie PointCloud, dlatego do konwersji użyto zmodyfikowanego programu point_cloud_converter.

Mapowanie terenu może być realizowane także w trybie ręcznego teleoperowania robotem. W tym przypadku przez węzeł user_interface uruchamiany jest program teleop.

Do komunikacji z robotem użyto bezprzewodowej sieci WiFi, a sterownik Raspberry PI skonfigurowano jako punkt dostępowy, tworzący sieć lokalną. Dodatkowo umożliwiono także komunikację poprzez protokół szyfrowany SSH.

Testy generowania map otoczenia

Problem generowania map otoczenia polega na zebraniu właściwej liczby pomiarów i odpowiednim ich przetworzeniu. Pomiary uzyskiwane z obrotowej głowicy LIDAR powstają w układzie sferycznym, więc należy je przekształcić do przestrzeni kartezjańskiej poprzez rzutowanie. Należy zwrócić uwagę, że robot jest mobilny, więc, aby dokładnie odwzorować skanowane otoczenie w układzie bezwzględnym (nieruchomym), konieczne jest posiadanie informacji o ruchu i położeniu robota. Można je wyznaczać z odometrii podczas ruchu lub wykorzystując, a następnie transformując wcześniej zmierzone punkty. W niniejszej pracy zaproponowano równoczesne użycie obu technik. Rozwiązanie to zarówno przyspiesza jak i udokładnia obliczenia. Wyznaczone w statycznym układzie kartezjańskim punkty mogą zostać wyświetlone jako gotowa mapa 2D lub model 3D z użyciem odpowiedniego interpretera.

W celu sprawdzenia poprawności działania programu generującego mapy 2D, wykonano szereg testów. Pierwsze pomiary wykonywano w kilku punktach oddalonych od siebie minimum o 1 m linii w prostej. Podczas testów sprawdzano jednorodność wykonanej mapy, tzn. to czy po wytworzeniu mapy z punktu A, przemieszczeniu się z punktu A do punktu B oraz wykonaniu kolejnego skanu z punktu B, wykonane pomiary odpowiednio nakładają się na wcześniej utworzoną mapę. Weryfikowano ciągłość krawędzi (ścian pomieszczeń) oraz zmierzone odległości. Aby zniwelować liczbę zmiennych wpływających na dokładność pomiarów, wstępne testy przeprowadzono w pomieszczeniach o ograniczonej liczbie krzywizn i przeszkód, np. na korytarzu (rys. 4).

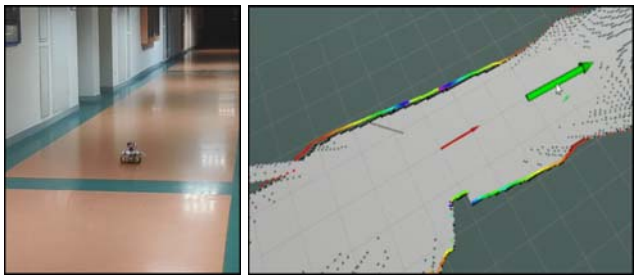
Zaprezentowana na rys. 3 mapa pochodzi z początku testów (przed optymalizacją algorytmów). Można na niej zauważyć punkty, przy których zmierzona odległość wybiega poza skalę. Pewną osobliwością, na którą natknięto się podczas przeprowadzania próby, były drzwi i ściany szklane.

Po każdorazowym wykonaniu pojedynczego skanu pomiary zostawały zapisane w postaci punktów na mapie. Następnie robot przemieszczał się, transponowany był punkt referencyjny, jakim jest położenie głowicy czujnika, a kolejny skan rozszerzał mapę. Okazało się, że niedokładności odometrii, na podstawie której wyznaczano położenie robota, wynikające głównie z poślizgu kół, powodują powstawanie zniekształceń na mapach wykonywanych z wielu punktów położenia przemieszczającego się robota (por. rys. 5a, 5b).

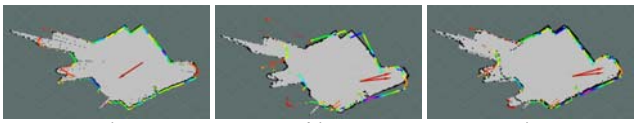
Po zidentyfikowaniu źródeł niedokładności wyznaczania położenia robota (przede wszystkim poślizgu kół) i zminimalizowaniu ich wpływu, a także wykorzystaniu łączonej techniki wykorzystującej odometrię i transformowanie kolejnych punktów pomiarowych wraz z dopasowaniem do poprzednich skanów udało się istotnie poprawić dokładność tworzenia map otoczenia robota (rys. 5c).

Dalszym testom poddano uzupełnianie mapy 2D i sprawdzenie wykonywania skanów w trybie jazdy teleoperowanej oraz autonomicznej. W trybie operowania zdalnego wykonano skan 2D spreparowanego otoczenia w kształcie litery L, po którym poruszał się robot (rys. 6). Jak można zauważyć (rys. 6a), robot z pierwotnej pozycji nie był w stanie zeskanować wnęki na końcu pomieszczenia. Dopiero po przemieszczeniu do odpowiedniego punktu oraz wykonaniu kolejnego skanu, mapa została prawidłowo uzupełniona. Warto zwrócić uwagę na lukę w narożniku wewnętrznym, którą robot wykrył i zarejestrował w postaci pojedynczych punktów uciekających poza obrys litery L.

W trybie autonomicznym robot miał za zadanie utworzyć skan pierwotny, a następnie samodzielnie określić punkt, w którym należałoby wykonać kolejny skan, w taki sposób, aby uzyskać kompletną mapę. W trakcie procesu uzupełniania mapy, robot samodzielnie obrał 3 punkty, w których wykonał skan. Pierwsze dwa znajdowały się w przybliżeniu na środku pomieszczenia. Po wykonaniu pierwszego przemieszczenia i skanu robot przesunął się w stronę dłuższej krawędzi korytarza i ponowił pomiary. Skany uzyskane w trybie autonomicznym oraz trybie ręcznym okazały się zbliżone, jednak tryb autonomiczny zajął robotowi więcej czasu (wykonał on trzy skany zamiast dwóch).

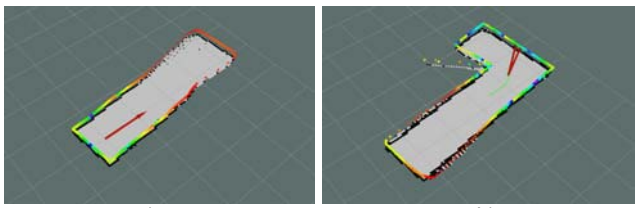


Rys. 4. Robot skanujący otoczenie i wygenerowana mapa 2D [6]



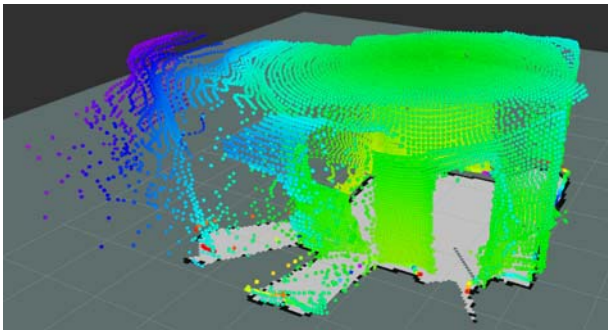
a) b) c)

Rys. 5. Korekta orientacji mapy względem pozycji robota: a) mapa wykonana na podstawie jednego skanu, b) mapa niedokładnie uzupełniona drugim skanem wykonanym po przemieszczeniu robota, c) mapa udokładniona drugim skanem (zoptymalizowane algorytmy lokalizacji robota i uzupełniania mapy) [6]



a) b)

Rys. 6. Autonomiczne uzupełnianie niewidocznych obszarów skanowania 2D: a) skan wykonany z jednego punktu początkowego, b) mapa uzupełniona drugim skanem po przemieszczeniu robota [6]



Rys. 7. Mapowanie 3D w postaci chmury punktów [6]

Robot w trybie pracy autonomicznej posiada dodatkową funkcję omijania przeszkód. Jest ona bardzo pożądana, szczególnie w nieznanym otoczeniu. Aby zbadać, czy robot, wyznaczając ścieżkę ruchu do zadanego punktu, jest w stanie poprawnie rozpoznać i ominąć potencjalne przeszkody na trasie ruchu, przeprowadzono dodatkowe testy. Przygotowano odseparowane środowisko na planie kwadratu. Na jego środku umiejscowiono przeszkodę. Zadano robotowi ruch po przekątnej w taki sposób, aby trasa najkrótsza była trasą kolizyjną z przeszkodą. Robot, wykonując pierwszy skan, wykrył przeszkodę (w oprogramowaniu robota zdefiniowano minimalną odległość na jaką może się zbliżać do przeszkód). Następnie wyznaczył nową trajektorię oraz wykonał ruch okrążający przeszkodę, utrzymując odpowiednią odległość zarówno od ścian jak i od przeszkody, w ten sposób zaliczając test.

Po wykonaniu serii testów sprawdzających tworzenie map 2D, zbadano funkcjonalność tworzenia map 3D. Konstruowanie mapy trójwymiarowej jest realizowane poprzez nałożenie chmury punktów wyznaczonych przy pomiarach pod kolejnymi kątami nachylenia do podłoża na mapę 2D. Rysunek 7 prezentuje wizualizację 3D w programie RViz pomieszczenia z rys. 5 na podstawie wykonanego przez robota skanu 3D. Zwizualizowane pomieszczenie powstaje z chmury odpowiednio położonych i pokolorowanych punktów. Do kolorowania punktów zastosowano tzw. skalę hutniczą, która przedstawia za pomocą koloru odległości punktów od punktu obserwacji. Mapowanie 3D może być bardzo użyteczne w przypadku konieczności pozyskania informacji o wysokości pomieszczeń czy też stopnia ich zniszczenia podczas pomiarów katastrof budowlanych, bądź badania możliwości wprowadzenia sprzętu ratunkowego.

Podsumowanie

Zaprezentowana technika zrobotyzowanego mapowania przestrzeni z wykorzystaniem czujnika LIDAR umieszczonego na autonomicznym robocie mobilnym wraz z dedykowanym oprogramowaniem może samodzielnie dostarczyć dane reprezentujące skany i mapy otoczenia w formacie 2D i 3D w formie tzw. chmury punktów. Jako wynik procesów skanowania 3D, chmury punktów mogą być wykorzystywane do wielu celów, w skanowania przestrzeni, ale także do tworzenia trójwymiarowych modeli CAD do produkcji (wydruku) części, a także kontroli jakości oraz do wielu innych aplikacji wymagających wizualizacji, animacji lub renderowania. Zbudowany robot może także wspomóc rekonstrukcję niedostępnych albo nieznanych rzutów pomieszczeń lub większych przestrzeni. Ograniczeniami w pracy robota są strome wzniesienia i progi, a także wymagany czas wykonywania dokładnych skanów. Tryb pracy autonomicznej robota umożliwia mapowanie bez wcześniejszej znajomości otoczenia.

Praca sfinansowana z subwencji na 2020r.

***Autorzy**: dr inż. Paweł Pawłowski, inż. Michał Jeske, inż. Natalia Kołodziejczyk, inż. Szymon Kwiatkowski, Politechnika Poznańska, Wydział Automatyki, Robotyki i Elektrotechniki, Instytut Automatyki i Robotyki, Zakład Układów Elektronicznych, i Przetwarzania Sygnałów, , ul. Jana Pawła II 24, 60-965 Poznań, E-mail: pawel.pawlowski@put.poznan.pl*

LITERATURA

[1] S. Liu, Benewake TFmini vs.TFmini plus, different sensors?, Ardupilot – forum https://discuss.ardupilot.org/t/benewake-tfminivs-tfmini-plus-different-sensors/37878/10 (dostep 09.04.2020)

[2] J. Park, J. Y. Kim, B. Kim and S. Kim, Global Map Generation using LiDAR and Stereo Camera for Initial Positioning of Mobile Robot, *2018 International Conference on Information and Communication Technology Robotics (ICT-ROBOT)*, Busan, 2018, 1-4.

[3] D. Ghorpade, A. D. Thakare, S. Doiphode, Obstacle Detection and Avoidance Algorithm for Autonomous Mobile Robot using 2D LiDAR," *2017 International Conference on Computing, Communication, Control and Automation (ICCUBEA)*, Pune, 2017, 1-6.

[4] M. Wise, T. Foote TurtleBot, strona produktu, https://www.turtlebot.com, (dostep 09.04.2020)

[5] D. Sierociuk, Nawigacja robota mobilnego przy użyciu dalmierza na podczerwień, *Przegląd elektrotechniczny*, 04/2004, 362-364

[6] M. Jeske, N. Kołodziejczyk, S. Kwiatkowski, Mobilny robot mapujacy przestrzen 3D w technologii LIDAR, Praca dyplomowa inżynierska, Politechnika Poznańska, 2020

[7] J. Illade-Quinteiro, V. M. Brea, P. López, D. Cabello, G. Doménech-Asensi, Distance Measurement Error in Time-of-Flight Sensors Due to Shot Noise, *Sensors 2015*, 15, DOI:10.3390/s150304624, 4624-4642.

[8] Dokumentacja czujnika LIDAR TFmini PLUS, https://cdn.sparkfun.com/assets/1/4/2/1/9/TFmini_Plus_A02_Product_Manual_EN.pdf (dostep 09.04.2020).