AUTOMATYKA • 2010 • Tom 14 • Zeszyt 3/1

Dominik Sankowski*, Sławomir Jeżewski*, Maciej Łaski*

Lokalizacja robota mobilnego w czasie rzeczywistym na podstawie danych ze skanera laserowego

1. Wprowadzenie

W niniejszej publikacji rozważa się problem lokalizacji robota mobilnego [1] w nieznanym otoczeniu. Problem ten w robotyce mobilnej znany jest pod nazwą SLAM (*Simultaneous Localization And Mapping*) [2, 3] i jest on fundamentem dla autonomicznego poruszania się oraz orientacji w terenie. W klasycznym ujęciu SLAM, robot wykorzystuje różnego rodzaju czujniki do zbierania informacji o otoczeniu. W miarę odkrywania nowych obszarów przeprowadza proces lokalizacji, opierając się na zgromadzonych danych i dodaje nowe informacje dotyczące aktualnej pozycji do bazy wiedzy (mapy). Na jej podstawie może wyznaczać ścieżki ruchu i decydować o kierunku jazdy.

1.1. Określenie problemu

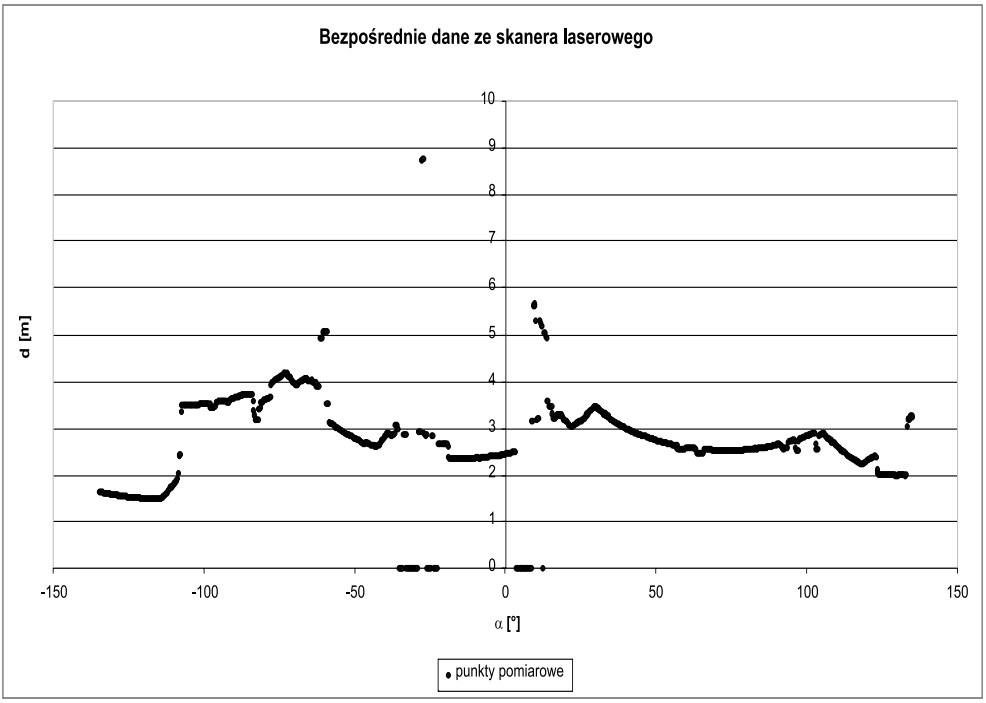
W literaturze problem lokalizacji robota mobilnego jest często omawiany. Większość autorów podejść stosuje metodę obliczeń wsadowych (off-line), tzn.: robot zatrzymuje się, wykonuje pomiar, przeprowadza lokalizację i przemieszcza się w inne miejsce. Innym podejściem jest zebranie pomiarów poprzez zdalne sterowanie robotem, a tworzenie mapy i lokalizacja odbywają się w późniejszym czasie. Wstępnie otrzymane dane są później używane do autonomicznego poruszania się robota. Taka organizacja obliczeń zwalnia autorów z obowiązku opracowania optymalnych czasowo algorytmów i doboru odpowiedniego sprzęty obliczeniowego.

Autorzy tej publikacji skupili się na lokalizacji robota w czasie rzeczywistym. Przez czas rzeczywisty rozumiana jest zdolność przetworzenia wszystkich danych przychodzących od czujników w trakcie poruszania się robota.

W ściślejszym ujęciu rozważany jest problem lokalizacji robota mobilnego za pomocą skanera laserowego [4, 5]. Urządzenie to generuje promień światła, który odbija się od

* Katedra Informatyki Stosowanej, Politechnika Łódzka

obiektów w otoczeniu i powracając niesie informację o odległości do przeszkody. Dzięki wykorzystaniu obrotowego zwierciadła skaner potrafi liczyć odległości do różnych obiektów znajdujących się w płaszczyźnie prostopadłej do osi obrotu lustra. Dane odbierane z lasera są zbiorem pomierzonych w konkretnej chwili czasowej odległości **d(i)**, dla danego kąta obrotu lustra **α(i)**, gdzie *i* jest numerem próbki w zbiorze (rys. 1).



Rys. 1. Wykres danych pochodzących od skanera laserowego d(α)

Taka para wartości jednoznacznie identyfikuje punkt w układzie biegunowym. Na rysunku 2 pokazano przekształcone za pomocą przekształcenia (1) punkty układu biegunowego do postaci kartezjańskiej. Taka operacja daje odzwierciedlenie otoczenia robota.

$$P(i)=\begin{cases} x(i)=d(i)\cdot\cos\left(\alpha(i)\right) \\ y(i)=d(i)\cdot\sin\left(\alpha(i)\right) \end{cases} \tag{1}$$

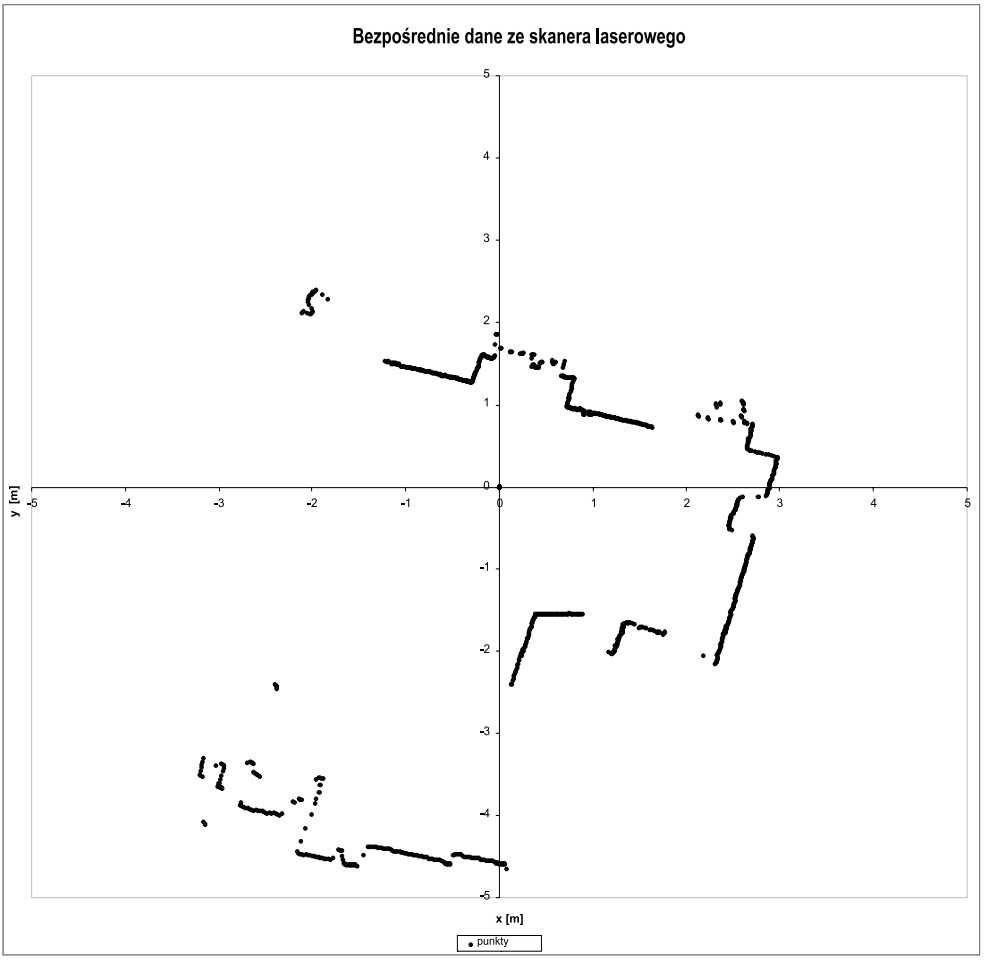
Powyżej został zobrazowany skan otoczenia wykonany skanerem laserowym. Jest to obraz rzeczywistego pomieszczenia, w którym znajdował się robot.

Podczas poruszania się robot zbiera kolejne dane o otoczeniu, a skaner nieprzerwanie działa i przesyła informację o odległościach do przeszkód znajdujących się wokół robota.

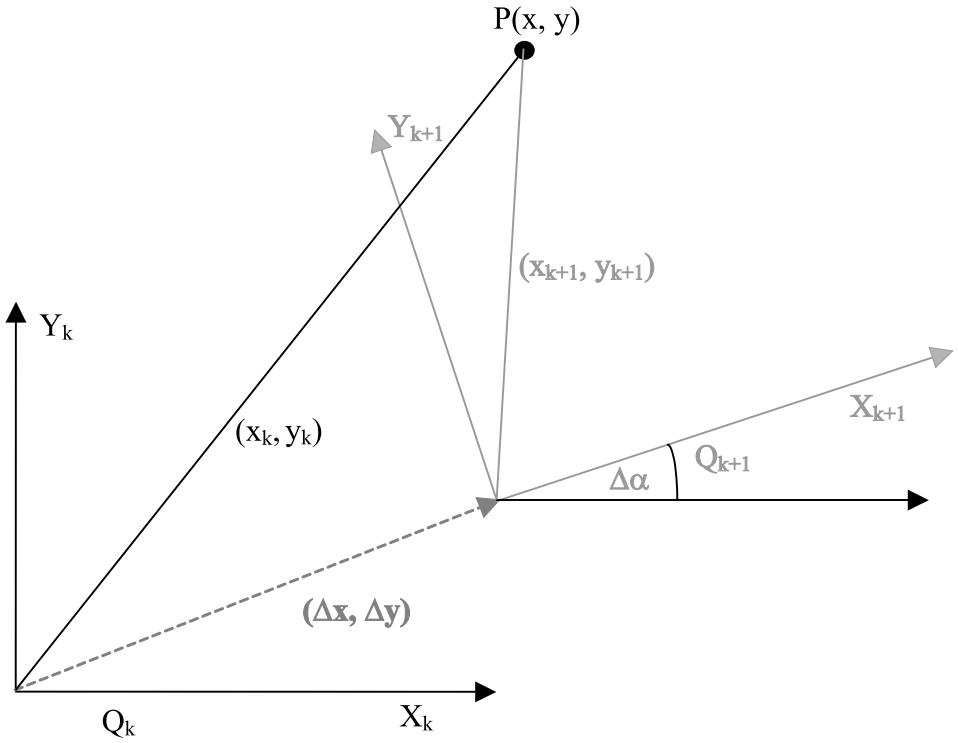
Znając translację (Δx, Δy) i rotację robota ($\Delta\alpha$), w przestrzeni definiuje się transformację (2), pokazaną na rysunku 3, która przekształci punkty jednego skanu Q_k w drugi skan Q_{k+1}.

$$\begin{bmatrix} x_{k+1}(i) \\ y_{k+1}(i) \end{bmatrix} = \begin{bmatrix} \cos(\Delta\alpha) & -\sin(\Delta\alpha) \\ \sin(\Delta\alpha) & \cos(\Delta\alpha) \end{bmatrix} \begin{bmatrix} x_k(i) \\ y_k(i) \end{bmatrix} + \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \tag{2}$$

W powyższym wzorze x_k i y_k oznaczają kolejne punkty skanu Q_k. Natomiast x_{k+1} i y_{k+1} są punktami obliczonymi za pomocą wzoru (2) i powinny pokrywać się z rzeczywistymi punktami odebranymi od skanera laserowego jeśli zmienimy pozycję robota o (Δx, Δy, $\Delta\alpha$).



Rys. 2. Wizualizacja skanu z rysunku 1 przekonwertowanego na układ kartezjański



Rys. 3. Transformacja pomiędzy dwoma skanami Q_k i Q_{k+1}

1.2. Algorytm ICP

Problem lokalizacji robota mobilnego polega na wyznaczeniu transformacji (Δx, Δy) oraz rotacji ($\Delta\alpha$). Istnieje kilka metod rozwiązania tego problemu, takie jak filtry cząsteczkowe [6, 7] czy lokalizacja na podstawie geometrii otoczenia [8]. Autorzy tej publikacji skupią się na algorytmie ICP, który został po raz pierwszy zaprezentowany w roku 1992 [9]. Został on użyty, jako metoda służąca do rejestracji trójwymiarowych kształtów we wszystkich sześciu stopniach swobody. Celem algorytmu było odnalezienie transformacji pozwalającej przekształcić zarejestrowany trójwymiarowy obiekt tak, aby jak największą częścią pokrywał się z poprzednią rejestracją tego obiektu z innego punktu widzenia.

Algorytm ten jest iteracyjny i w każdym kolejnym kroku zbliża rozwiązanie do lokalnego optimum. Zasadą działania jest obliczanie minimalnych odległości euklidesowych pomiędzy punktami dwóch obiektów i w kolejnych iteracjach dąży się do minimalizacji sumy tych odległości. Cechy tego algorytmu bardzo dobrze można wykorzystać do lokalizacji robota mobilnego opartej o skaner laserowy, gdyż dane pochodzące od lasera są zbiorem punktów.

1. Posiadając dwa kolejne skany odebrane od skanera laserowego Q_k i Q_{k+1} wyznacza się początkowe wartości (Δx, Δy, $\Delta\alpha$). Przeważnie przyjmują one wartość zero, ale można je wyznaczać za pomocą estymacji prędkości lub przy użyciu innych czujników robota, takich jak enkodery kwadraturowe.
2. Następnie używając przekształcenia ze wzoru (2) wyznacza się transformację zbioru Q_{k+1} w Q'_k. W ostatniej iteracji Q'_k powinien pokrywać się z Q_k.

$$\begin{bmatrix} x'_k(i) \\ y'_k(i) \end{bmatrix} = \begin{bmatrix} \cos(\Delta\alpha) & -\sin(\Delta\alpha) \\ \sin(\Delta\alpha) & \cos(\Delta\alpha) \end{bmatrix} \begin{bmatrix} x_{k+1}(i) \\ y_{k+1}(i) \end{bmatrix} + \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$$

3. Kolejnym krokiem jest odnalezienie najbliższych sąsiadów pod względem odległości w zbiorach Q_k i Q'_k. Definiuje się pewną funkcję $F(i)$ określającą indeks, dla którego $(x'_k(i), y'_k(i))$ jest najbliższy punktowi $(x_k(F(i)), y_k(F(i)))$.
4. Definiowana jest funkcja błędu:

$$E = \sum_{i=1}^{n} \left(\left(x_k\left(F(i)\right) - x'_k(i) \right)^2 + \left(y_k\left(F(i)\right) - y'_k(i) \right)^2 \right) \tag{3}$$

5. Dalej dokonuje się minimalizacji funkcji E dla każdej z poszukiwanych wartości:

$$\begin{aligned} \Delta x &= \frac{\partial E}{\partial x} \\ \Delta y &= \frac{\partial E}{\partial y} \\ \Delta\alpha &= \frac{\partial E}{\partial \alpha} \end{aligned} \tag{4}$$

6. Z kolei, jeżeli zmiana wartości funkcji błędu między iteracjami jest większa od zadanego parametru t, powraca się do punktu 2 i wykonuje kolejną iterację z nowymi wartościami szukanych parametrów.

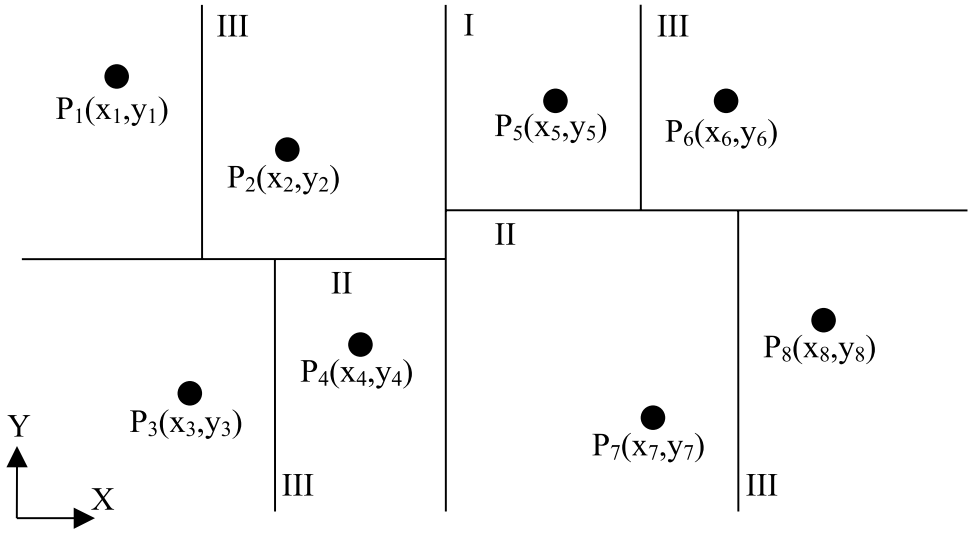
1.3. Drzewo KD

Najbardziej czasochłonną operacją podczas realizacji algorytmu ICP jest obliczanie najbliższych sąsiadów pomiędzy dwoma zbiorami. Dlatego autorzy skupili się na optymalizacji metody wyszukiwania najbliższych sąsiadów, wykorzystując do tego celu właściwości iteracyjne algorytmu:

– Jeden ze zbiorów nigdy nie jest przekształcany, więc można go wykorzystać do zbudowania struktury przyspieszającej obliczenia.

– Kolejne iteracje przekształcają skan o mały krok, więc nowy sąsiad powinien znajdować się w pobliżu punktu określonego, jako najbliższy sąsiad z poprzedniej iteracji. W ostatnich iteracjach, kiedy oba zbiory prawie się pokrywają, bardzo dużo punktów będzie miało tego samego sąsiada, co w poprzedniej iteracji.

Drzewo KD [10] (rys. 4) jest strukturą przyspieszającą obliczenia najbliższego sąsiada poprzez dzielenie przestrzeni na podprzestrzenie względem kolejnych wymiarów. Taki podział powoduje powstanie lokalnie posortowanych zbiorów, na których można wykonać operację binarnego poszukiwania.



Rys. 4. Przykładowe drzewo KD

Zaproponowana przez autorów optymalizacja tego drzewa polega na przechowywaniu dodatkowych informacji dotyczących AABB (*Axis Aligned Bounding Box*) dla wszystkich węzłów. Patrząc na rysunek 4, w trakcie pierwszego podziału oznaczonego rzymską cyfrą *I*, zbiór punktów $\{P_1, P_2, P_3, P_4\}$ nie jest ograniczony z góry, z dołu i z lewej strony. Z prawej natomiast istnieje ograniczający punkt P_5. Informacja ta jest znacząca i znana podczas rekursywnego generowanie drzewa, dlatego nie wymaga dodatkowych obliczeń. W miarę generowania kolejnych poziomów drzewa dodawane są pozostałe wartości ograniczające konkretny węzeł.

Przetrzymywanie wyżej wymienionych danych ma znaczenie ze względu na drugą własność algorytmu ICP. Poszukiwanie najbliższego sąsiada za pomocą drzewa KD odbywa się przy użyciu analizy top-down (z góry na dół). Powoduje to konieczność przeszukania wszystkich poziomów drzewa. W prezentowanym podejściu algorytm ICP wykonuje jedną klasyczną iterację, w trakcie której zapamiętuje wskaźniki do odnalezionych liści

drzewa. Dzięki temu każdą kolejną iterację można przyspieszyć, dzięki uruchomieniu algorytmu poszukiwania od wcześniej zapamiętanego najbliższego sąsiada. W tym przypadku drzewo przeszukiwane jest przy użyciu analizy bottom-up (z dołu do góry), a ciężar obliczeń spada na testowanie przechowywanych AABB dla danego węzła.

Ta optymalizacja może prowadzić do znacznego przyspieszenia działania algorytmu ICP. Badania metody zostaną zamieszczone w dalszej części publikacji.

2. Badania algorytmu

Opracowane algorytmy zostały testowane na dwóch typach danych: pochodzących ze środowiska symulacyjnego oraz z rzeczywistego robota. Symulację przeprowadzono robota w środowisku Microsoft Robotics Developer Studio [11]. Dane z niej uzyskane są syntetyczne i odzwierciedlają wirtualne otoczenie miejskie. Dzięki użyciu silnika fizyki Nvidia PhysX w systemie symulacji są one zbliżone do sygnałów rzeczywistych. Dane pochodzące od rzeczywistego robota, który zbudowany jest w oparciu o sześciokołową platformę jezdną. Układ powstaje w Katedrze Informatyki Stosowanej Politechniki Łódzkiej w ramach grantu Ministerstwa Nauk i Szkolnictwa wyższego z 2009 r.

Przedstawiany robot mobilny wyposażony jest w czujnik laserowy potrafiący skanować otoczenie i podawać odległości do przedmiotów znajdujących się w otoczeniu robota w zakresie 270^{o} z rozdzielczością $0{,}25^{o}$. Zakres pomiarowy tego czujnika wynosi 30 metrów. Najważniejszym parametrem jest czas jednego pełnego pomiaru wynoszący 25 milisekund. Przyjęto, że jest to czas, w którym algorytm lokalizacji powinien wykonać obliczenia i rozpocząć oczekiwanie na nowe dane.

Robot mobilny podlega ograniczeniom w postaci wielkości oraz ilości energii, jaką dysponuje. Dlatego należy optymalnie dobrać jednostkę obliczeniową, która będzie wykonywała obliczenia, ale zarazem będzie niewielkich rozmiarów i będzie pobierała niewiele prądu. Z tego powodu w rzeczywistym robocie, który jest wykorzystany do przeprowadzenia badań, zdecydowano się na minikomputer klasy Intel Atom. Jest on wyposażony w procesor Intel Atom o częstotliwości taktowania 1,66 GHz i 1 GB pamięci RAM DDR2. Wszystkie wyniki ilościowe zaprezentowane w niniejszej publikacji odniesione są do tej platformy sprzętowej.

2.1. Przebieg badań

Badanie przeprowadzono na trzech zestawach danych:

– Pierwszy zestaw pochodził ze środowiska symulacyjnego. Dane te pochodzą od poruszającego się robota w symulowanym otoczeniu miejskim. W skład otoczenia wchodziły: budynki, latarnie i ławki.

– Kolejny zestaw został pobrany od rzeczywistego skanera znajdującego się na robocie. Podczas zbierania danych robot znajdował się w pomieszczeniu i nie poruszał się. W skład otoczenia wchodziły: ściany (wraz z nierównościami typu: gniazdka, listwy itp.), ławka z komputerem, krzesło i dwa serwery typu mainframe.
– Ostatni prezentowany zestaw pochodzi od tego samego skanera laserowego. Tym razem jednak robot poruszał się pomiędzy trzema pomieszczeniami połączonymi korytarzem. W skład otoczenia wchodziły: biurka, krzesła, stanowiska pomiarowe i inne przedmioty znajdujące się w typowym biurze.

Dla każdego z wyżej wymienionych zestawów danych wykonano porównanie pod względem czasu wykonania przy użyciu lokalizacji bez struktury przyspieszającej, z drzewem KD oraz z drzewem KD posiadającym pamięć o poprzedniej iteracji. Czas w systemie linuks badany był funkcją systemową *gettimeofday*. Wyniki zawierają minimalny, średni i maksymalny czas wykonania wszystkich lokalizacji dla zebranych danych.

Następnie wykonano dokładne porównanie każdej iteracji algorytmu dla trzeciego zestawu danych. Wybrano ten zestaw, ponieważ jest odebrany z rzeczywistego skanera i w rzeczywistych warunkach.

3. Wyniki

3.1. Porównanie algorytmu ICP z użyciem optymalizacji i bez użycia optymalizacji

Tabela 1
Obliczenia lokalizacji wykonane dla pierwszego zastawu danych

Algorytm	Minimalny czas działania [s]	Średni czas działania [s]	Maksymalny czas działania [s]	Odchylenie standardowe [s]	Procent lokalizacji poniżej 25 ms
ICP bez optymalizacji	0,554604	0,824109	1,354978	0,104225	0
ICP z drzewem KD	0,012432	0,022993	0,060852	0,007899	66,35
ICP z drzewem KD z pamięcią	0,004502	0,013165	0,046149	0,008191	89,0

Z tabeli 1 wynika jednak, iż algorytm ICP z pamięcią o ostatniej iteracji radził sobie najlepiej. Bardzo ważnym wskaźnikiem jest maksymalny czas działania, który mówi, ile danych algorytm traci w przypadku, kiedy lokalizacja nie powiedzie się w ciągu 25 milisekund. Dla powyższych danych w najgorszym przypadku algorytm ICP traci trzy skany, a ICP z pamięcią tylko dwa.

Tabela 2
Obliczenia lokalizacji wykonane dla drugiego zestawu danych

Algorytm	Minimalny czas działania [s]	Średni czas działania [s]	Maksymalny czas działania [s]	Odchylenie standardowe [s]	Procent lokalizacji poniżej 25 ms
ICP bez optymalizacji	1,221614	1,268628	2,011277	0,069640	0
ICP z drzewem KD	0,028079	0,030927	0,064248	0,004326	0
ICP z drzewem KD z pamięcią	0,014675	0,017661	0,047292	0,004534	90,4535

W otoczeniu zamkniętym, gdzie robot był ograniczony czterema ścianami, zazwyczaj wszystkie próbki skanera laserowego trafiały w jakąś przeszkodę. Dlatego można oczekiwać gorszych rezultatów niż wcześniej przedstawione dla danych symulacyjnych.

Z tabeli 2 wynika, iż algorytm ICP z drzewem KD zupełnie nie radzi sobie z lokalizacją w czasie rzeczywistym. Za każdym razem należałoby pobierać od skanera laserowego, co drugi skan, a w najgorszym przypadku, co trzeci. Algorytm ICP z pamięcią działał w czasie rzeczywistym w 90 procentach przypadków, a w najgorszym przypadku musiałby pominąć tylko jeden zestaw danych.

Tabela 3
Obliczenia lokalizacji wykonane dla trzeciego zestawu danych

Algorytm	Minimalny czas działania [s]	Średni czas działania [s]	Maksymalny czas działania [s]	Odchylenie standardowe [s]	Procent lokalizacji poniżej 25 ms
ICP bez optymalizacji	1,191564	1,356930	2,200160	0,069982	0
ICP z drzewem KD	0,027480	0,034254	0,071496	0,005587	0
ICP z drzewem KD z pamięcią	0,012975	0,020903	0,051788	0,006628	78,8507

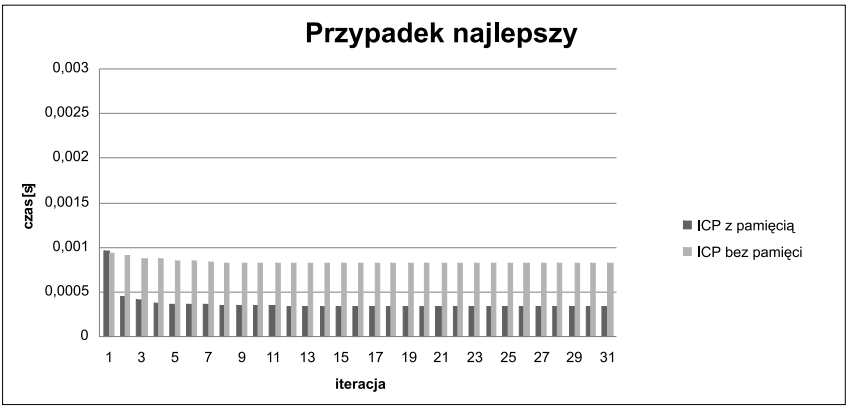
Dla trzeciego zestawu danych sytuacja jest bardzo podobna jak w poprzednim przypadku. Mniejsza ilość poprawnych lokalizacji mogła być spowodowana eksploracją przestrzeni i odkrywaniem nowych obszarów, w których skany nie pokrywały się całkowicie.

W tym przypadku maksymalny czas działania algorytmu ICP z pamięcią jest większy niż 50 ms. Powoduje to konieczność odrzucenia trzech skanów pochodzących od lasera, jednak obiecującą informacją, którą można odczytać z tabeli 3 jest fakt, iż czas przekraczający 50 ms wynosi tylko 2 ms, a średni czas wykonania lokalizacji pozostawia 4 ms oczekiwania na nowe dane. Dzięki temu ciężar obliczeń, które przekroczyły wyznaczone ramy czasowe można przenieść na kolejne wywołanie algorytmu. Może to być jednak niebezpieczne, jeżeli wystąpi kilka z rzędu niepoprawnych lokalizacji.

Uwzględniając, że działanie algorytmu odbywało się pod kontrolą systemu operacyjnego na procesorze z jednym rdzeniem, można przyjąć, że część lokalizacji była dłuższa ze względu na czas wywłaszczony przez system operacyjny. Problem ten dotyczy wszystkich trzech badanych przypadków.

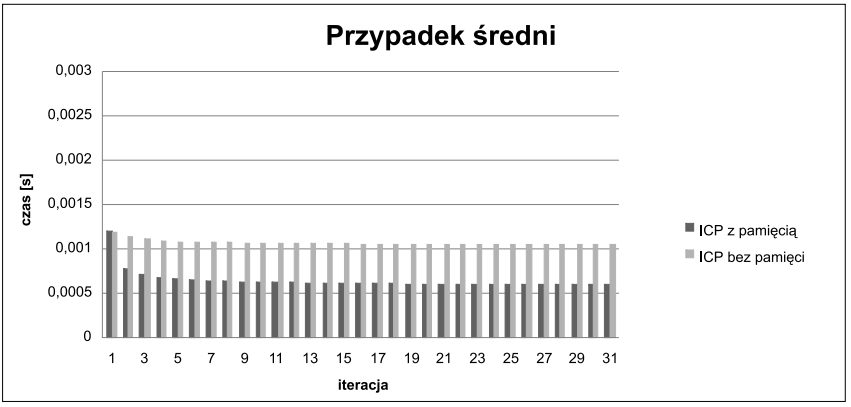
3.2. Porównanie pojedynczej iteracji dla algorytmu ICP bez pamięci i ICP z pamięcią

Poniżej zobrazowany został przebieg pojedynczej lokalizacji.



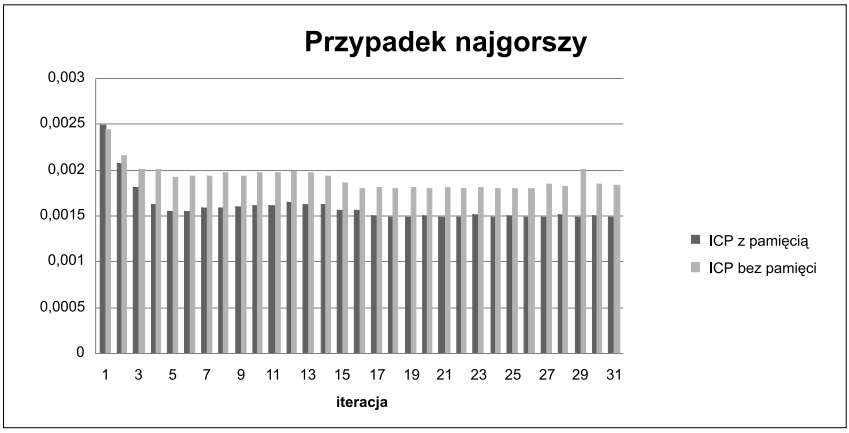
Rys. 5. Lokalizacja, podczas której algorytm uzyskał najlepszy czas

Na rysunku 5 zobrazowana została lokalizacja, w której algorytm ICP osiągnął najlepszy czas wynoszący: 0,012975 sekundy.



Rys. 6. Uśredniony czas ze wszystkich lokalizacji

Na rysunku 6 pokazano uśredniony czas kolejnych iteracji. Średnio pojedyncza lokalizacja trwała 0,020903 sekundy.



Rys. 7. Lokalizacja, podczas której algorytm uzyskał najgorszy czas

Na rysunku 7 pokazano najgorszy przypadek, w którym algorytm lokalizacji nie zmieścił się w wyznaczonych ramach czasowych. Czas ten wynosił 0,051788 sekundy.

Na osi *X* zaznaczone są kolejne iteracje odbywające się podczas pojedynczej lokalizacji i ich czas trwania na osi *Y*. Jak widać, czas trwania algorytmu ICP z pamięcią zmniejsza się w trakcie trwania procesu, co potwierdza założenie z punktu 1.3.

W każdym z trzech powyższych przypadków algorytm przeszukiwania drzewa z pamięcią jest szybszy od algorytmu bez tej optymalizacji.

4. Podsumowanie i wnioski

Bardzo często zdarzało się, że przedmioty wchodzące w skład symulowanego otoczenia miejskiego były oddalone od robota o więcej niż 30 metrów. W takiej sytuacji, aby utrzymać zgodność z rzeczywistym laserem, symulowany laser nie podawał poprawnej próbki. Z tego powodu część danych została odrzucona jeszcze na etapie zbierania danych. Jest to powodem uzyskania nieco lepszych wyników czasowych niż w zaprezentowanych poniżej przypadkach.

Dzięki wykorzystaniu struktury przyspieszającej oraz proponowanej przez autorów optymalizacji drzewa KD można bardzo przyspieszyć wykonywanie się algorytmu lokalizacji. Jest to bardzo ważne za względu na problem postrzegania otoczenia przez robota mobilnego – im więcej przetworzonych danych, tym większa wiedza jest dostępna dla algorytmów wyższego poziomu, np.: nawigacyjnych. Kolejnym istotnym parametrem jest użycie procesora. Na dzień dzisiejszy występuje ograniczenie co do wybranej jednostki obliczeniowej, natomiast wraz z wzrostem technologii procesory będzie można zamienić na wydajniejsze. Użycie optymalizacji algorytmu lokalizacji pozwoli na wykorzystanie pozostałej części mocy obliczeniowej na inne zadania.

Rozwój technologii lokalizacji zmierza także w stronę lokalizacji trójwymiarowej, która do poprawnego działania wymaga przetworzenia znacznie większej ilości danych. Zaproponowane algorytmy i optymalizacje mogą zostać wykorzystane także do celów lokalizacji trójwymiarowej, co może prowadzić do uzyskania przetwarzania w czasie rzeczywistym także przy użyciu tego podejścia.

Rys. 8. Mapa uzyskana z trzeciego zestawu pomiarowego

Na rysunku 8 została pokazana mapa wygenerowana w czasie rzeczywistym za pomocą opisywanego algorytmu ICP. Jedynymi danymi źródłowymi były informacje pochodzące od skanera laserowego. Punkt (0, 0) umieszczony jest w miejscu, gdzie robot rozpoczął badanie przestrzeni. Na mapie tej można zauważyć geometrię pomieszczeni oraz obiekty, które się w niej znajdowały. Na podstawie takich danych algorytmy wyższego poziomu

będą podejmowały decyzję o ścieżce ruchu czy omijaniu przeszkód. Czarną linią zaznaczone zostały punkty, w których robot znajdował się podczas zbierania danych. Wszystkie te punkty także są wyznaczone za pomocą algorytmu ICP.

Publikacja została wykonana w ramach grantu naukowego MNiSW z roku 2008–2010.

Współautor Maciej Łaski jest stypendystą programu: „Innowacyjna dydaktyka bez ograniczeń – zintegrowany rozwój Politechniki Łódzkiej – zarządzanie uczelnią, nowoczesna oferta edukacyjna i wzmacnianie zdolności do zatrudniania, także osób niepełnosprawnych”.

Literatura

[1] Jeżewski S., Sankowski D., Dadan W., *Koncepcja autonomicznego robota pola walki przeznaczonego do zadań zwiadu i wykrywania min*. ISSN: 1429-3447, 2009.

[2] Nuchter A., *The Simultaneous Localization and Mapping Problem with Six Degrees of Freedom.* ISSN 1610-7438, 2009.

[3] Stachniss C., *Robotic Mapping and Exploration.* ISSN 1610-7438, 2009.

[4] Jensfeld P., Christensen H.I., *Pose Tracking Using Laser Scanning and Minimalistic Environmental Models.* IEEE Transactions on Robotics and Automation, vol. 17, No. 2, April 2001.

[5] Aghamohammadi A.A., Taghirad H.D., Tamjidi A.H., Mihankhah E., *Feature-Based Laser Scan Matching For Accurate and High Speed Mobile Robot Localization.* Advanced Robotics and Automated Systems (ARAS), 2007.

[6] Howard A., *Multi-robot Simultaneous Localization and Mapping using Particle Filters.* In Robotics: Science and Systems, 2005.

[7] Dieter F., Thrun S., Burgard W., Dellaert F., *Particle Filters for Mobile Robot Localization*. 2001.

[8] Wang C., Thorpe C., *Simultaneous Localization and Mapping with Detection and Tracking of Moving Objects*. Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA), 2003, 842–849.

[9] Besl P.J., McKay N.D., *A Method for Registration of 3-D Shapes*. IEEE Transaction on Pattern Analysis and Machine Intelligence, vol. 14, No. 2, February 1992.

[10] Havran V., *Heuristic Ray Shooting Algorithms*. Ph.D. Thesis, November 2000.

[11] Jeżewski S., Łaski M., *Przegląd i porównanie środowisk symulacji robotów mobilnych*. ISSN: 1429-3447, 2009.