Politechnika Wrocławska

Wydział Elektroniki, Fotoniki i Mikrosystemów

KIERUNEK: Automatyka i Robotyka (AIR)

PRACA DYPLOMOWA INŻYNIERSKA

TYTUŁ PRACY:

Budowanie map otoczenia dla robota mobilnego w środowisku ROS

AUTOR:

Marcel Konieczny

PROMOTOR:

dr inż. Wojciech Domski

WROCŁAW 2023

Streszczenie

Celem pracy było zbadanie możliwości efektywnego budowania map otoczenia na podstawie dwuwymiarowego rozkładu normalnego, wykorzystując dane skanera laserowego 2D. Na początku pracy opisane zostało środowisko symulacyjne oraz architektura projektu. Projekt został wykonany w środowisku symulacyjnym ROS oraz Gazebo. Następnie opisany został proces przetwarzania oraz grupowania danych skanera laserowego. Kolejno opisana została parametryzacja otrzymanych grup na podstawie dwuwymiarowego rozkładu normalnego. Otrzymane grupy parametrów są wykorzystywane w celu budowania mapy otoczenia. W trakcie pracy badawczej zostały zaimplementowane dodatkowe założenia, dotyczące przetwarzania danych, w celu osiągnięcia najbardziej dokładnej reprezentacji budowanego otoczenia. Na końcu pracy porównano mapy wygenerowane bez żadnych ograniczeń do tych wygenerowanych dla ograniczenia np. ruchu robota. Na podstawie otrzymanych wyników zauważono, że najdokładniejsze mapy zostały wygenerowane gdy robot się nie przemieszczał.

Summary

The major objective of this thesis is to investigate the possibility of effectively environment map building on the basis of the two-dimensional normal distribution, using 2D laser scanner data. At the beginning of the thesis, the simulation environment and project architecture were described. The project was made in simulation enviroment: ROS and Gazebo. Then, the process of processing and grouping the laser scanner data was described. Next the parameterization of the obtained groups on the basis of the two-dimensional normal distribution was described. The obtained groups of parameters are used to build a map of the environment. During the research work, additional assumptions regarding data processing were implemented in order to achieve the most accurate representation of the built environment. At the end of the research, maps generated without any restrictions were compared to those generated with limit e.g. robot movement. It was noticed that the most accurate maps were generated when the robot was not moving.

Słowa kluczowe: roboty mobilne, skaner laserowy, klasteryzacja danych, transformacja rozkładów normalnych, budowanie mapy

Keywords: mobile robots, laser scanner, data clustering, Normal Distributions Transform, map building

Spis treści

Rozdział 1

Wstęp

Roboty mobilne stają się coraz popularniejsze, świadczyć mogą o tym dane z rynku robotów mobilnych, którego wartość w 2018 roku była szacowana na 18.7 mld USD. Na podstawie przeprowadzonych analiz założono, że do 2023 roku wzrośnie ona do wartości 54 mld USD [8]. Tego typu roboty wykorzystywane są w różnych dziedzinach życia człowieka, zaczynając od prostych robotów sprzątających mieszkania, po zaawansowane konstrukcje robotyczne wysyłane w kosmos w celu eksploracji oraz badania struktury planet. Poprzez możliwość uniwersalnego dostosowania, roboty mobilne znalazły także zastosowanie w walce z epidemią COVID-19, pozwalając na kontakt (np. sprawdzenie temperatury ciała) z osobą chorą bez konieczności przebywania w jej otoczeniu [2].

Jednym z podstawowych zadań robota mobilnego jest ruch. Robot może być sterowany przez operatora lub poruszać się autonomicznie. Ruch autonomiczny sprawia, że robot sam podejmuje decyzje, w które miejsce należy się przemieścić, aby wykonać zlecone zadanie, nie byłoby to możliwe bez mapy otoczenia oraz lokalizacji. Jedną z najbardziej popularnych technik wykorzystywanych do poruszania się autonomicznym robotem jest SLAM (z ang. *Simultaneous Localization and Mapping*). Wykorzystanie algorytmu typu SLAM wyróżnia się tym, że w czasie rzeczywistym, na podstawie danych z swoich czujników, robot generuje mapę otoczenia i określa swoje położenie [7].

Zauważyć można, że budowanie map jest jednym z podstawowych problemów z którymi muszą się zmagać autonomiczne roboty mobilne. Bez mapy robot nie jest w stanie efektywnie samodzielnie się poruszać. Wraz z rozwojem technologii można przetwarzać dane coraz szybciej, nowe czujniki dostarczają coraz dokładniejsze pomiary. Implementując odpowiednie algorytmy robot jest w stanie generować dokładne mapy otoczenia. Jednym z sposobów budowania map otoczenia jest klasteryzacja danych.

Klasteryzacja danych jest podstawowym algorytmem związanym z eksploracją danych. Tego typu metody koncentrują się na przetwarzaniu dużej ilości danych w celu uzyskania pewnych informacji o badanym środowisku, na przykład: pewnych regularności, ograniczeń [5],[3]. Stosując klasteryzację grupujemy otrzymane dane na wybraną ilość podzbiorów według charakterystyki danej informacji. Tak przedstawione grupy pozwalają na analizę otrzymanych danych.

Robot mobilny wyposażony w czujniki odległości dostarcza dane na temat otoczenia, w którym się znajduje. Stosując klasteryzację można przetwarzać otrzymywane dane w celu zmniejszenia ilości zajmowanej przez nie pamięci. Dzieląc otrzymywane informacje na kilka grup i następne opisanie ich za pomocą kilku parametrów, pozwala na zaoszczędzenie pamięci oraz przechowanie większej ilości informacji w bardziej efektowny sposób. W przypadku robotów mobilnych należy pamiętać, że posiadamy ograniczoną ilość dostępnych zasobów.

Pomimo zauważalnych korzyści płynących z zastosowania algorytmów klasteryzacji danych, należy pamiętać, że cechują się one przeważnie złożonością czasową $O(N^2d)$, dla dużej ilości próbek robot może gubić informacje na temat skanowanego pomieszczenia ponieważ nie będzie w stanie tak szybko przetworzyć wszystkich danych [4]. Kolejną wadą stosowania algorytmów klasteryzacji do budowania mapy jest utrata pewnych informacji, algorytm dzieląc za każdym razem otoczenie na kilka grup, może nie uwzględnić wąskiego przejścia, które wystąpiło na ścianie skanowanego pomieszczenia. Stosując algorytmy klasteryzacji należy więc określić jaka będzie dla użytkownika optymalna dokładność mapy, ponieważ czym większa dokładność, tym większe zapotrzebowanie na pamięć i czas wykonania algorytmu.

Jedym z sposobów klasteryzacji jest zastosowanie algorytmu transformacji rozkładów normalnych (z ang. *Normal Distributions Transform, NDT*). Implementacja tego algorytmu w celu budowy mapy opiera się na podzieleniu skanowanej przestrzeni na zdefiniowaną ilość komórek. Wykonane przez robota pomiary odległości zostają przedstawione jako punkty, i następnie umieszczone zostają w odpowiednich komórkach. Po przydzieleniu punktów do odpowiednich komórek, komórki zostają sparametryzowane na podstawie dwukrotnego rozkładu normalnego. Zaawansowane implementacje algorytmu NDT pozwalają nie tylko na budowanie mapy otoczenia, ale także na lokalizowanie robota wykonującego skan na podstawie już wykonanych pomiarów [1]. Mając na uwadze, że czujniki umieszczone na robocie wykonują pomiary obarczone pewnymi błędami, zastosowanie kilku metod pomiaru niweluje powstałe błędy.

1.1 Teza

Wykorzystanie transformacji opartej na reprezentacji danych za pomocą rozkładu normalnego pozwala na efektywne tworzenie map otoczenia z informacji zbieranych przez robota mobilnego wyposażonego w skaner laserowy 2D.

Rozdział 2

Konfiguracja środowiska symulacyjnego

Symulacja jest popularnym sposobem prowadzenia badań. Realistyczne odwzorowanie własności fizycznych oraz niskie koszty, są głównymi zaletami tego typu badań. Wstępna symulacja danego modelu pozwala określić czy stworzona konstrukcja zadziała w rzeczywistym środowisku, w przypadku źle dobranych parametrów danych elementów badania symulacyjne zapobiegają utracie zasobów i czasu.

W celu sprawdzenia działania algorytmu NDT do budowania mapy otoczenia zdecydowano się skorzystać z ROS (z ang. *Robot Operating System*) oraz Gazebo.

ROS jest to open sourcowe środowisko dostosowane do symulacji oraz sterowania robotami. Uniwersalność, nieustający rozwój, dostęp do wielu bibliotek oraz możliwość implementacji własnych funkcjonalności w Pythonie lub w C++, czyni to narzędzie popularnym systemem symulacji robotycznych. Architektura tego systemu składa się paczek. Użytkownik ma do swojej dyspozycji wiele gotowych paczek, które może łączyć jak również dodawać do nich swoje elementy w zależności od zapotrzebowania. Każdy z pakietów może zawierać: węzły, inne pakiety czy oprogramowanie zaimplementowane przez użytkownika. Podział na poszczególne paczki został zastosowany w celu przejrzystości i łatwej przenośności pomiędzy urządzeniami [6].

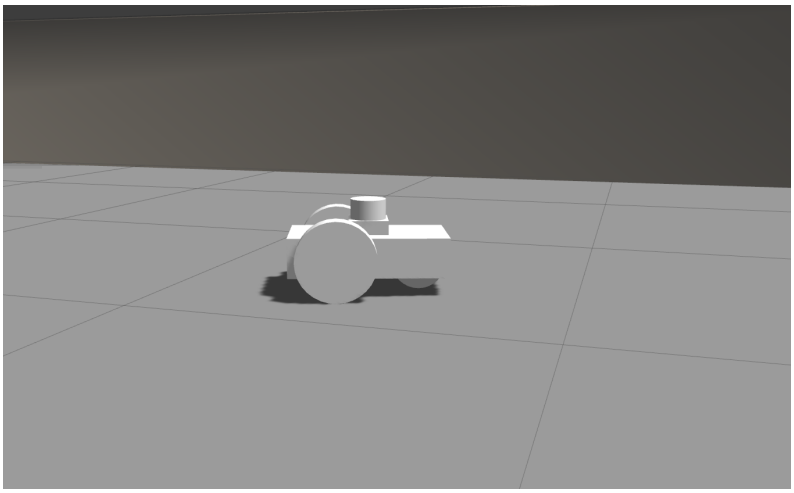
Elementem, który na ogół znajduje się w paczce jest węzeł, jest to oddzielny proces odpowiedzialny za kontrolowanie pewnych zdefiniowanych czynności takich jak: obsługa lasera, sterowanie robotem czy przetwarzanie danych i tworzenie mapy otoczenia. Węzły pomiędzy sobą przekazują dane za pomocą tematów, w celu otrzymania danych przez pewien węzeł musi on zasubskrybować odpowiedni temat, na przykład: węzeł lasera publikuje dane na pewnym temacie, węzeł odpowiedzialny za przetwarzanie danych chcąc uzyskać dane lasera musi zasubskrybować ten temat. Od chwili subskrypcji za każdym razem, gdy laser opublikuje dane na temacie, otrzymywać je będzie węzeł przetwarzania danych. Istotnym elementem ROS-a jest zarządca, odpowiada on za wskazanie węzłom lokalizacji, gdzie publikuje dany temat. W chwili, gdy węzeł uzyska informacje, gdzie znajduje się dany temat zostaje on zasubskrybowany i przepływ informacji odbywa się bez zarządcy [6]. W projekcie został wykorzystany ROS melodic.

W celu symulacji robota oraz otoczenia wykorzystano program Gazebo, program ten jest dostosowany do pracy razem z ROSem i jest udostępniony w ramach darmowej licencji. Duża liczba dostępnych wtyczek oferuje między innymi łatwą obsługę skanera laserowego czy kamery.

Tworząc środowisko symulacyjne utworzono 2 pakiety: `simulation` oraz `data_handle`. Pakiet `simulation` jest wykorzystywany w celu: wizualizacji robota w środowisku symulacyjnym, poruszania się robota oraz obsługi lasera. Pakiet `data_handle` jest wykorzystywany w celu: przetwarzania danych lasera oraz budowy mapy otoczenia. Każdy z

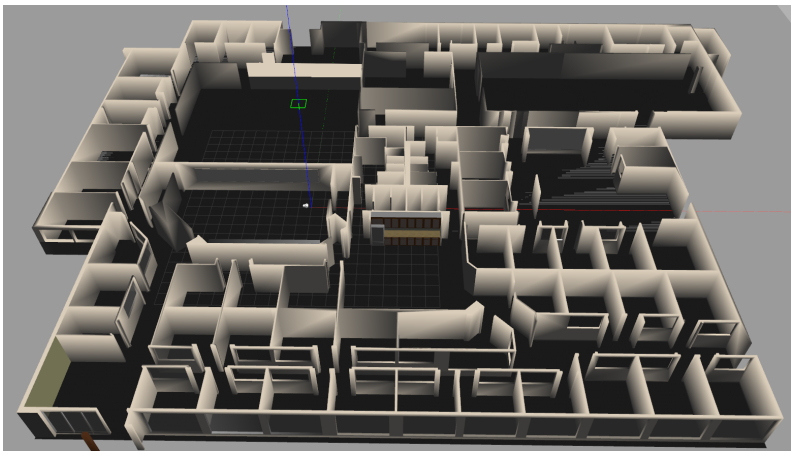
pakietów jest uruchamiany z osobnego pliku uruchomieniowego. Taka struktura pozwala między innymi na dokonywanie zmian w pakiecie związanym z przetwarzaniem danych bez konieczności wyłączania symulacji.

Tworząc projekt zdecydowano się wykorzystać robota klasy (2,0) (posiada dwa niezależne koła napędowe oraz brak niezależnych kół skrętnych). Model robota został zdefiniowany w pliku myrobot.xacro. W pliku tym zostały uwzględnione wszystkie wymiary robota, rodzaje połączeń pomiędzy elementami oraz macierze inercji poszczególnych elementów. Plik przedstawiający robota jest wczytywany przez Gazebo za pomocą pakietu `ros_gazebo_pckg`, pakiet ten łączy ROSa i Gazebo.



Rysunek 2.1: Model robota widziany w Gazebo

W celu monitorowania oraz możliwości sterowania robotem wykorzystano wtyczkę do Gazebo `differential_drive_controller`. Mając zdefiniowane odległości pomiędzy kołami napędowymi oraz ich średnicę, wtyczka ta pozwala na sterowanie robotem poprzez publikowanie komend na temacie `/cmd_vel`. Jest ona odpowiedzialna także za publikowanie informacji o przemieszczeniu i pozycji robota na temacie `/odom`. Wykorzystując program konsolowy `teleop_twist_keyboard` użytkownik może publikować dane na temacie `/cmd_vel`, jest więc w stanie sterować robotem i przemieszczać go w wybrane miejsca. Zdecydowano, że domyślnie robot będzie się poruszać po ogólnodostępnej mapie `Willow Garage` widocznej na rysunku 2.2.

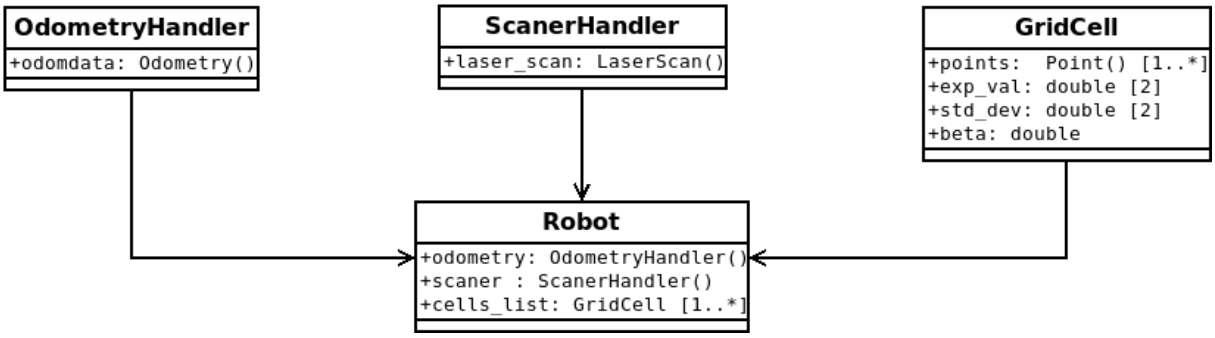


Rysunek 2.2: Mapa Willow Garage z robotem umieszczonym w lewym górnym rogu

Robot wyposażony został w skaner laserowy. W celu uzyskiwania danych z czujnika laserowego użyto wtyczki `hakuyo_laser`. W projekcie wykorzystano laser 2D, który zwraca 720 odczytów. Pomiary są wykonywane z częstotliwością 10 Hz i dokładnością do 0.01 metra. Wtyczka ta publikuje zmierzone wartości na temacie `/laser`. Opisane zależności pomiędzy tematami można przedstawić jako schemat blokowy (rys. 2.3).

Rysunek 2.3: Zależności pomiędzy tematami

Klasteryzacja danych w celu budowania mapy została zaimplementowana we wcześniej wspomnianym pakiecie `data_process`. Dołączając pakiet `rospy` możliwe było użycie języka Python, język ten cechuje się prostotą oraz dużą liczbą bibliotek. Zdecydowano, że w celu odbierania danych z tematów zostaną utworzone 2 klasy: `OdometryHandler` oraz `LaserHandler`. Klasa `OdometryHandler` subskrybuje temat `/odom` i przechowuje tylko aktualną pozycję. Dane te pozwalają na bieżące monitorowanie położenia robota. Klasa `LaserHandler` subskrybuje temat `/laser`, przechowuje ona także informacje, ile próbek laser mierzy oraz kąt w jakim laser skanuje. Obie klasy są elementami klasy `Robot`. Klasa `Robot` odpowiada także za przetworzenie danych, tak aby każdy z pomiarów mógł być przedstawiony jako punkt o współrzędnych (x,y) i następnie przydziela uzyskane punkty do odpowiednich grup. Grupy są zdefiniowane jako klasa `GridCell`. Klasa `GridCell` za pomocą algorytmu NDT parametryzuje przechowywane punkty uzyskując: wartości oczekiwane (μ_x i μ_y), odchylenia standardowe (σ_x i σ_y) oraz kąt pochylenia punktów względem osi X (β). Po wykonanej operacji grupowania punktów oraz parametryzacji, na podstawie uzyskanych wyników budowana jest mapa otoczenia zbudowana z elips. Zależności pomiędzy przedstawionymi klasami zostały przedstawione na rysunku 2.4.



Rysunek 2.4: Zależności pomiędzy klasami

Rozdział 3

Budowa mapy

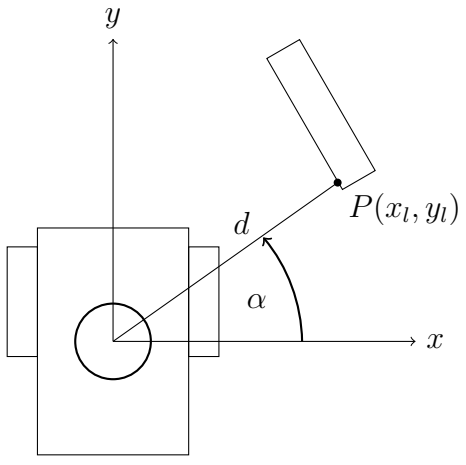
3.1 Przetwarzanie oraz grupowanie danych

Temat `/laser` dostarcza 720 próbek podczas jednego pomiaru. Znając aktualne położenie robota, wartości próbek oraz kąt skanowania, można przedstawić każdą z próbek pomiarowych jako punkt o współrzędnych (x, y). W celu uzyskania współrzędnych globalnych na początku obliczone zostają współrzędne lokalne punktu względem robota (3.1). Wiedząc, że pomiarów będzie 720 oraz że kąt skanowania jest równy 360°, wyznaczono, że wartość kąta α wzrasta o 0.5° wraz z obliczaniem każdej kolejnej próbki. Na rysunku 3.1 przedstawiony został pomiar dla jednej próbki.

$$\begin{cases} x_l = cos(\alpha) \cdot d \\ y_l = sin(\alpha) \cdot d \end{cases}, \tag{3.1}$$

gdzie:

- d to zmierzona odległość,
- α to wartość kąta odchylenia względem pierwszej próbki.



Rysunek 3.1: Robot w lokalnym układzie współrzędnych

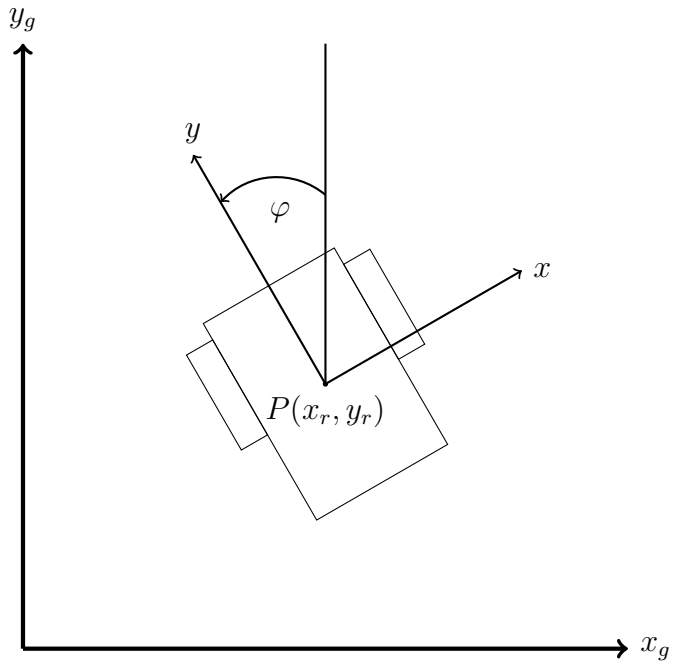
Mając obliczony punkt w lokalnym układzie współrzędnych (x_l, y_l) oraz odczytane położenie robota (x_r, y_r), za pomocą (3.2) obliczone zostają współrzędne globalne próbki.

Wykorzystując fakt, że laser został umieszczony w punkcie (0,0) układu lokalnego robota, położenie robota oraz punkt względem którego wykonywane są pomiary, są sobie równe.

$$\begin{cases} x = cos(\varphi) \cdot x_l - sin(\varphi) \cdot y_l + x_r \\ y = sin(\varphi) \cdot x_l + cos(\varphi) \cdot y_l + y_r \end{cases}, \tag{3.2}$$

gdzie:

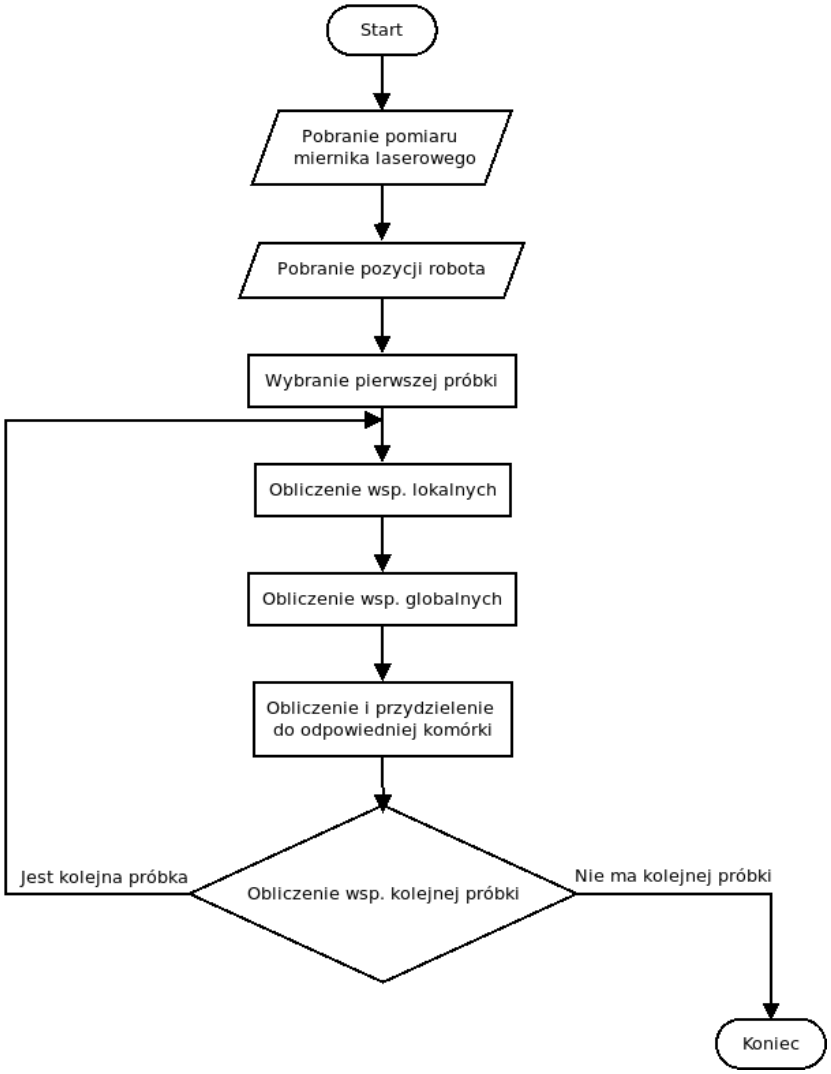
- x_l , y_l to obliczone współrzędne lokalne,
- x_r, y_r to współrzędne położenia robota,
- φ to kąt odchylenia robota względem osi Y.



Rysunek 3.2: Robot w globalnym układzie współrzędnych

Realizując budowanie mapy w oparciu o algorytm NDT zdecydowano, że budowana mapa będzie podzielona na sektory w kształcie kwadratu o długości boku 0,5 metra. Uzyskane punkty w procesie przetwarzania danych skanera laserowego trafiają do odpowiednich komórek. Każda z komórek jest reprezentowana przez obiekt typu `GridCell`. W celu optymalnego zarządzania pamięcią komputera, odpowiednie kwadraty są tworzone jedynie, gdy przynajmniej jeden punkt należy do danego kwadratu. Powyższe operacje zostają wykonane dla każdej próbki z pomiaru lasera.

Algorytm zaimplementowany w celu przetwarzania próbek lasera pobiera pomiar lasera, następnie pobrana zostaje pozycja robota. Uzyskując 720 próbek, w pętli każda zostaje przetworzona i przydzielona do odpowiedniej komórki. W tym celu dla pojedynczej próbki zostają obliczone współrzędne lokalne i następnie współrzędne globalne. Otrzymany punkt, na podstawie swoich wartości, zostaje przydzielony do odpowiedniej komórki. Po umieszczeniu punktu w komórce zostaje przetworzona kolejna próbka, jeżeli wszystkie próbki zostały przetworzone algorytm kończy swoje działanie. Schemat blokowy algorytmu przedstawiony został na rysunku 3.3.



Rysunek 3.3: Schemat blokowy operacji przetwarzania próbek lasera

3.2 Budowa mapy

Mając rozdzielone punkty pomiędzy poszczególnymi komórkami, zaimplementowane rozwiązanie rozpoczyna proces budowania mapy. W tym celu przetworzone zostają dane w każdej z komórek, tak aby można je było przedstawić jako elipsy.

Parametryzując przechowywane w komórce `m` punktów, na początku obliczona zostaje wartości oczekiwane μ_x i μ_y (3.3), uzyskane wartości odzwierciedlać będą środek elipsy.

$$\begin{cases} \mu_x = \frac{1}{m} \sum_{i=0}^{m} x_i \\ \mu_y = \frac{1}{m} \sum_{i=0}^{m} y_i \end{cases}. \tag{3.3}$$

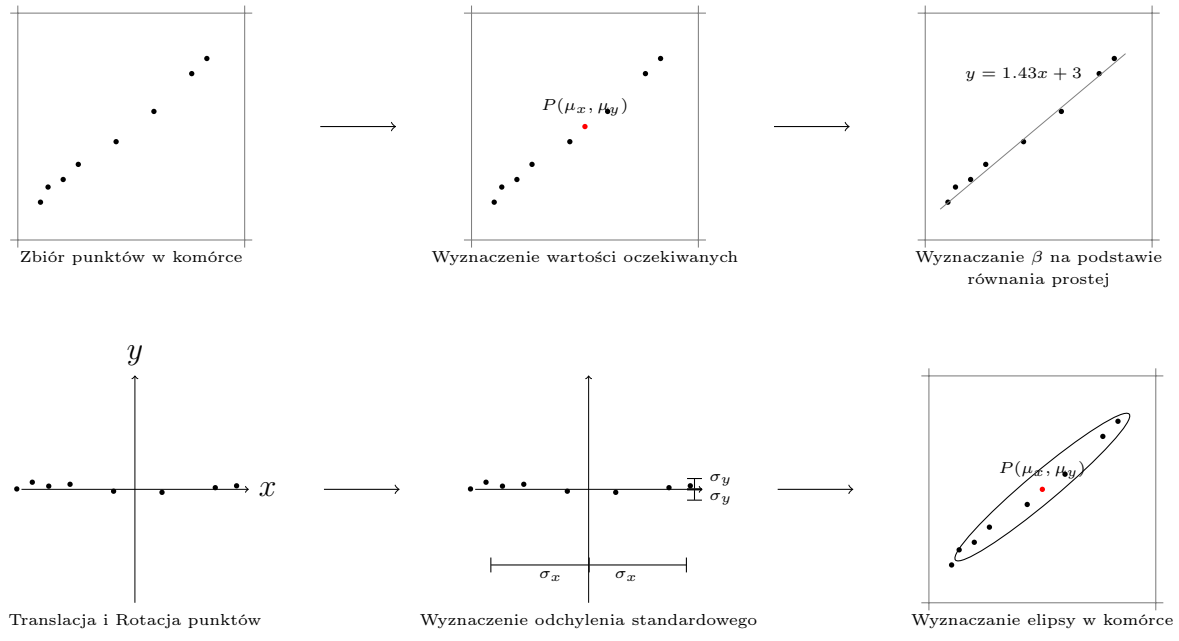
Następnie na podstawie punktów w komórce aproksymowane zostaje równanie prostej, w celu przedstawienia zbioru punktów jako prosta o równaniu $y = ax + b$. Wartość współczynnika a pozwala na wyznaczenie kąta nachylenia prostej względem osi X. Zdecydowano, że równanie prostej będzie wyznaczane na podstawie skrajnych punktów. Stosując taki sposób wyznaczania prostej, komputer jest w stanie szybko obliczyć wartości współczynników a i b, pozwala to na przetworzenie wielu komórek szybciej, niż gdyby prosta była

wyznaczana na podstawie wszystkich punktów w komórce. Taki sposób jest jednak nieodporny na błędy pomiarów, w przypadku błędnego pomiaru jednego z dwóch punktów (skrajnych), równanie prostej nie będzie odzwierciedlało rzeczywistego położenia punktów w komórce. Mając wyznaczony kąt β nachylenia prostej do osi X oraz wartości oczekiwane μ_x i μ_y przeprowadzona zostaje translacja punktów o P$(-\mu_x, -\mu_y)$, aby prosta, aproksymowana na postawie punktów w komórce, przechodziła przez początek układu współrzędnych. Następnie wykonana zostaje rotacja wszystkich punktów w komórce o kąt $-\beta$, względem punktu $P(0,0)$, tak aby punkty częściowo pokrywały się z osią X układu współrzędnych. Nie wykonując operacji translacji oraz rotacji punków, generowane elipsy posiadałyby błędne wymiary półosi.

Mając przeprowadzoną translację oraz rotację wszystkich punktów można wyznaczyć odchylenie standardowe σ_x oraz σ_y (3.4). Dodatkowo założono, że odchylenie standardowe na podstawie którego będą generowane elipsy będzie średnią starej wartości σ i nowo obliczonej wartości. Takie założenie sprawia, że wpływ na wartość odchylenia standardowego mają także poprzednie wartości pomiarów. W przypadku gdy otrzymane wartości jednego z pomiarów jest obarczona błędem, poprzednio obliczona wartość będzie miała wpływ na wymiary półosi elipsy. Podczas pierwszego obliczania wartości σ w danej komórce, wartości otrzymane na podstawie (3.4) są wartościami końcowymi.

$$\begin{cases} \sigma_x = \sqrt{\frac{\sum_{i=0}^{m}(x_i-\mu_x)^2}{m-1}} \\ \sigma_y = \sqrt{\frac{\sum_{i=0}^{m}(y_i-\mu_y)^2}{m-1}} \end{cases}. \tag{3.4}$$

Na podstawie odchylenia standardowego wyznaczane są mała oraz wielka półoś elipsy. Operacja budowania mapy na przykładzie pojedynczej komórki widoczna jest na diagramie blokowym (rys. 3.4).

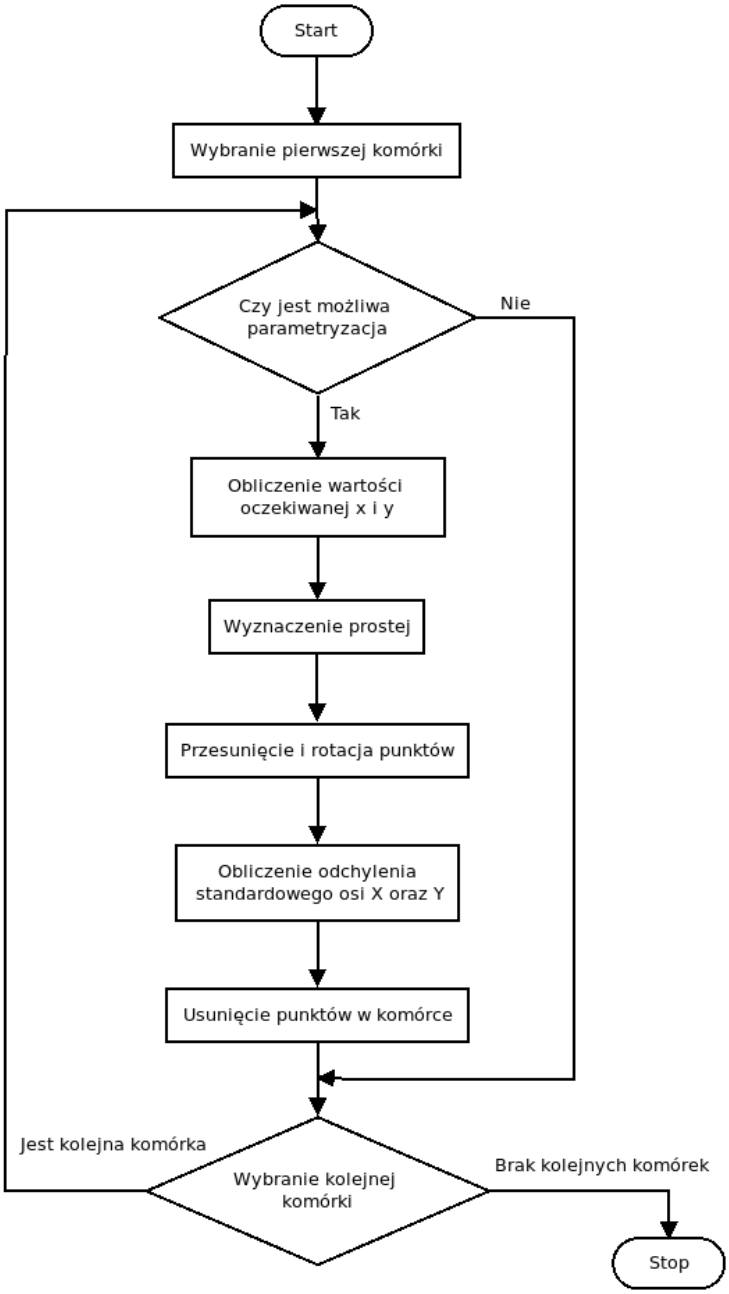


Rysunek 3.4: Proces wyznaczania elipsy

Po zakończeniu operacji parametryzacji usunięte zostają wszystkie punkty, które zawierała dana komórka. Przechowywane są jedynie informacje na temat: odchylenia standardowego względem osi X (σ_x) i Y (σ_y), kąt nachylenia β oraz wartości oczekiwane μ_x i

μ_y, na podstawie tych wartości generowane są elipsy.

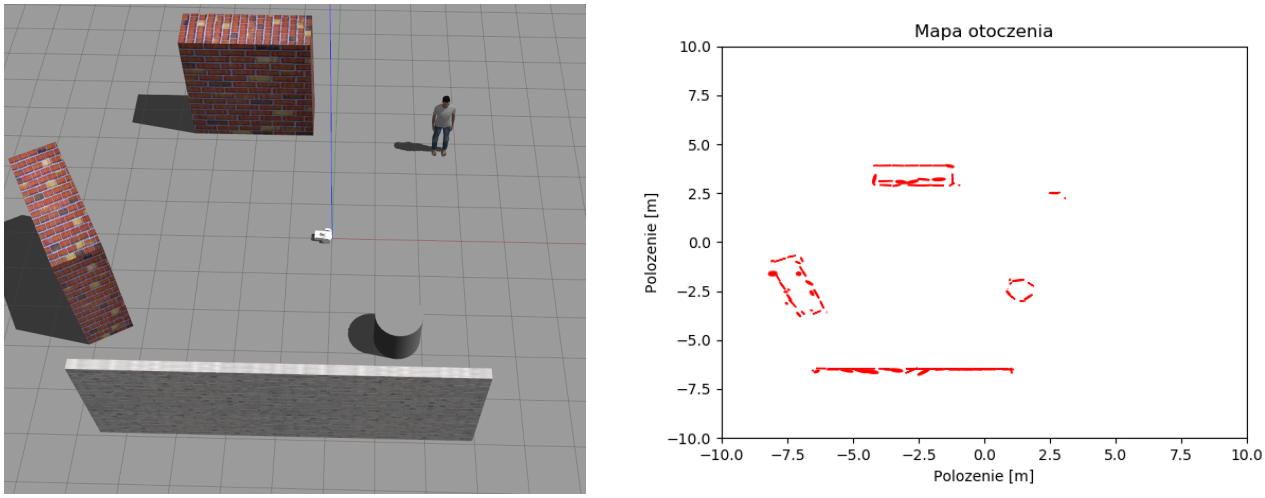
Algorytm zastosowany w celu parametryzacji wszystkich komórek opiera się na pętli, która na początku sprawdza czy komórka może być sparametryzowana. Jeśli w komórce nie ma wystarczającej ilości punktów (2 – tyle punktów potrzebnych jest do wyznaczenia równania prostej), wybierana jest kolejna komórka. Jeżeli w komórce znajdują się przynajmniej 2 punkty: obliczone zostają wartości oczekiwane, wyznaczone równanie prostej, przeprowadzona zostaje translacja oraz rotacja punktów w komórce. Na podstawie zrotowanych punktów, wyliczone zostają odchylenia standardowe, a następnie usunięte zostają wszystkie punkty z komórki. Po przeprowadzonej parametryzacji wybrana zostaje kolejna komórka. Jeżeli każda komórka została sprametryzowana algorytm kończy działanie. Kolejna parametryzacja komórek odbędzie się po przetworzeniu kolejnych próbek lasera. Opisany przebieg parametryzacji umieszczono na schemacie blokowym widocznym na rysunku 3.5.



Rysunek 3.5: Schemat blokowy parametryzacji komórek

W celu wizualizacji mapy złożonej z elips, zdecydowano się skorzystać z biblioteki języka Python – `matplotlib`. Zaimplementowana tam funkcja `Ellipse` pozwala na generowanie elips poprzez podanie wartości: punkt środka elipsy, wielkość mniejszej i większej półosi oraz kąta pochylenia. Jako parametry podano więc wartość oczekiwaną (punkt środka elipsy), odpowiednio pomnożone odchylenie standardowe punktów na osi X i Y (wielkości półosi) oraz kąt pochylenia. Początkowo zdecydowano się generować elipsy z półosiami będącymi dwukrotnościami odchylenia standardowego, takie założenie sprawiło, że pomiędzy poszczególnymi elipsami powstawały wolne przestrzenie. Użytkownik nieświadomy sposobu działania zastosowanego algorytmu budowania mapy, mógłby zostać wprowadzony w błąd. Z tego powodu zdecydowano się na zastosowanie trzykrotności odchylenia standardowego, co sprawiło, że tak powstałe elipsy minimalnie się zazębiają. Generowane ściany nie posiadają znaczących ubytków. Wielkość wizualizowanej mapy domyślnie była przystosowana do wymiarów 10 metrów na 10 metrów. W momencie, w którym były generowane elipsy poza początkowo zdefiniowanym polem, wielkość wizualizowanego terenu się zwiększa.

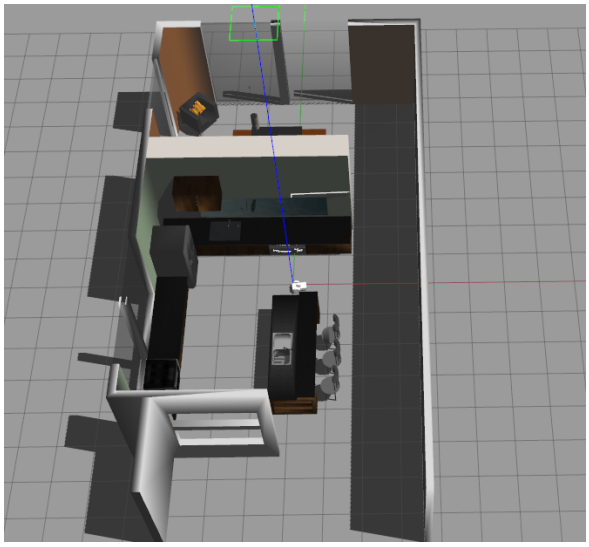
W celu sprawdzenia zaimplementowanego algorytmu klasteryzacji na początku umieszczono na wolnej przestrzeni kilka przeszkód i poprowadzono robota pomiędzy przeszkodami (rys. 3.6). Następnie umieszczono robota w przestrzeni przedstawiającej 2 pokoje zawierające dużo szczegółów (meble z wgłębieniami, stoły czy krzesła) (rys. 3.7). Na końcu przeprowadzono badania działania algorytmu na mapie Willow Garage (rys. 3.8). Po lewej stronie widoczne są środowiska symulacyjne w programie Gazebo (rys. 3.6a÷ 3.8a). Po prawej stronie widoczna jest wygenerowana mapa (rys. 3.6b ÷ 3.8b).



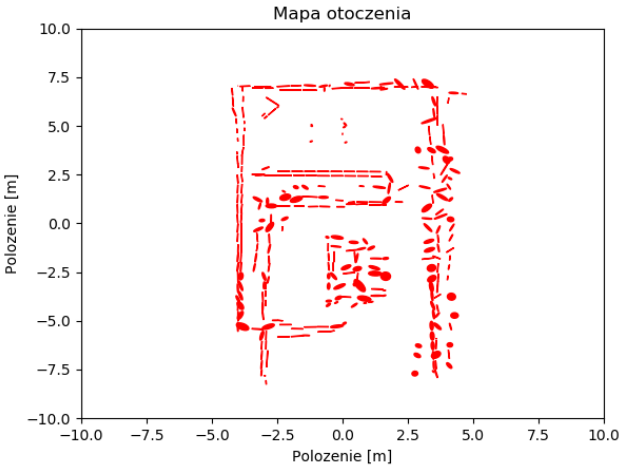
(a) Przestrzeń w gazebo

(b) Wygenerwana mapa

Rysunek 3.6: Budowa mapy dla prostego otoczenia

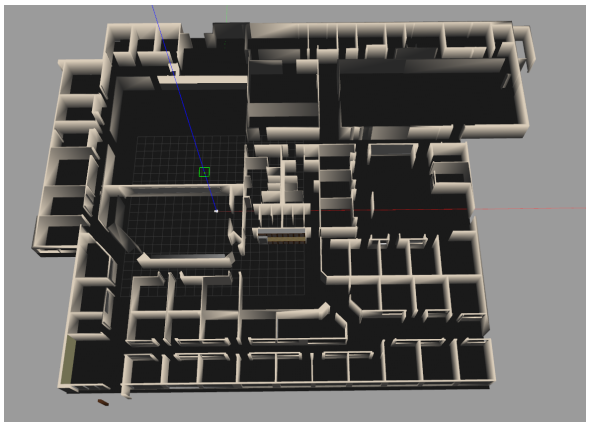


(a) Przestrzeń w gazebo

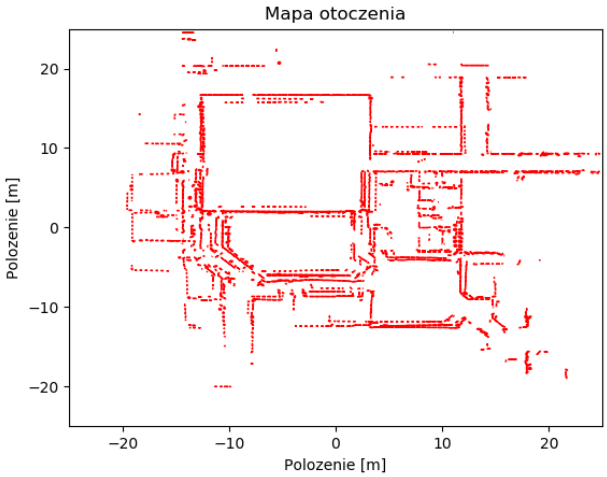


(b) Wygenerwana mapa

Rysunek 3.7: Budowa mapy dla pomieszczeń z wieloma szczegółami



(a) Przestrzeń w gazebo



(b) Wygenerwana mapa

Rysunek 3.8: Budowa mapy Willow Garage

Na podstawie wygenerowanych map (rys. 3.6 ÷ 3.8), można zauważyć, że miejscowo algorytm generował te same przestrzenie (ściany, obiekty) przesunięte, prawdopodobnie wynika to z wykonywania pomiarów laserem podczas obrotu robota. Przetworzenie tak wykonanych pomiarów wyznaczało błędne wartości współrzędnych punktów co w rezultacie sprawiało, że punkty trafiały do nieprawidłowych komórek. W celu rozwiązania problemu zablokowano możliwość budowania mapy w czasie obrotu robota. Pomiary lasera są uwzględniane tylko podczas postoju lub ruchu liniowego robota.

Stwierdzono również, że jedną z przyczyn generowania elips z błędnym pochyleniem może być źle obliczona wartość współczynnika a, na podstawie, której jest obliczane pochylenie elipsy. Z tego powodu zdecydowano się zaimplementować równanie aproksymacji prostej na podstawie punktów w komórce:

$$\begin{pmatrix} b \\ a \end{pmatrix} = (X^T X)^{-1} X^T y, \tag{3.5}$$

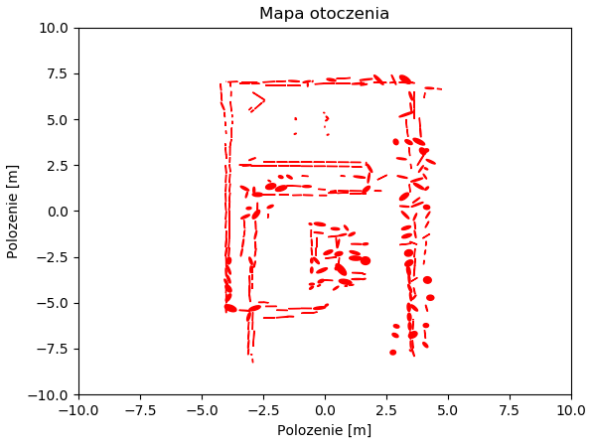
gdzie:

- X jest to macierz składająca się z jedynek w pierwszej kolumnie oraz wartości x, punktów znajdujących się w komórce w drugiej kolumnie,
- y jest to wektor wartości y, punktów znajdujących się w komórce,
- a oraz b są to współczynniki równania prostej.

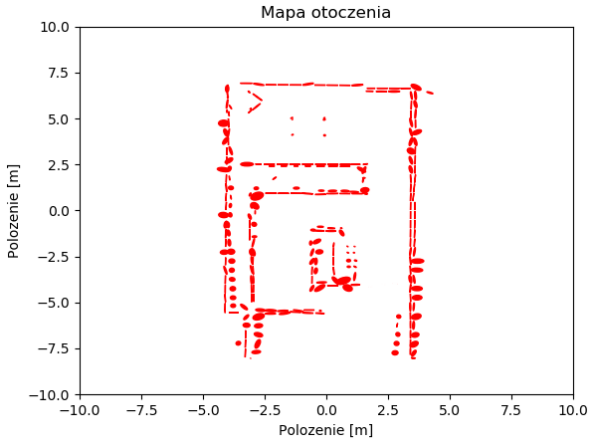
Rozwiązując równanie otrzymano wektor składający się z współczynników aproksymowanej prostej o równaniu $y = ax + b$. W odróżnieniu do pierwotnie zaimplementowanego rozwiązania, wykorzystanie regresji liniowej pozwoli na wyznaczenie współczynników prostej na podstawie wszystkich punktów w komórce.

Dodatkowo założono, aby punkty w komórkach były przetwarzane jedynie wtedy, gdy znajdzie się w nich przynajmniej 10 punktów. Wyznaczenie współczynników prostej oraz odchylenia standardowego na podstawie większej ilości punktów powinno dokładniej przedstawiać rzeczywiste wartości. Wówczas występuje mniejsze prawdopodobieństwo związane z błędnym wygenerowaniem elips.

Po zaimplementowaniu przedstawionych rozwiązań, wybudowano kolejne mapy otoczenia które zestawiono z poprzednio wybudowanymi mapami bez założeń (rys. 3.9 i 3.10). Zdecydowano że po lewej stronie będą widoczne mapy wygenerowane bez ograniczeń (rys. 3.9a, 3.10a), po prawej z nowo zaimplementowanymi ograniczeniami (rys. 3.9b, 3.10b).

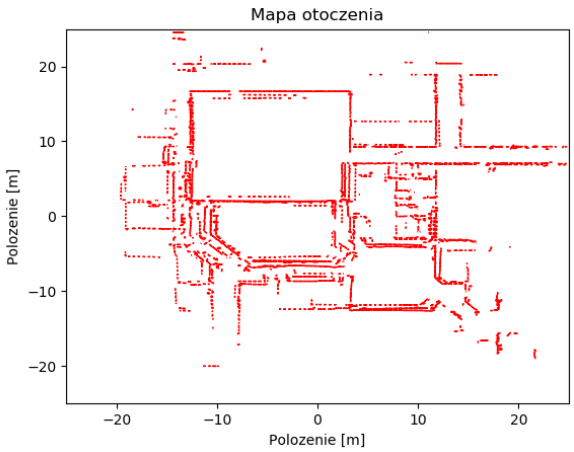


(a) Mapa bez dodatkowych założeń

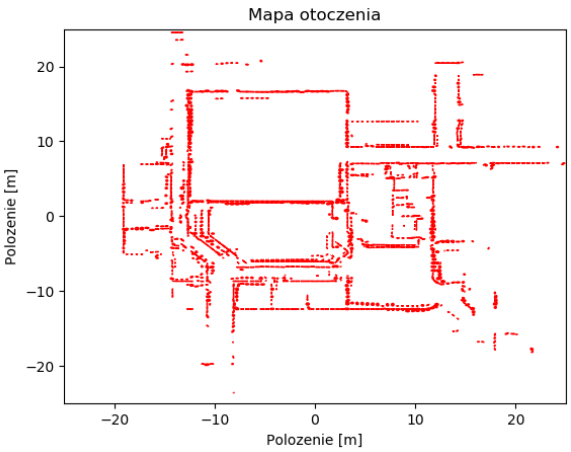


(b) Mapa z założeniami: nie przetwarzanie danych w trakcie obrotu robota, wykorzystanie regresji liniowej do wyznaczenia współczynników prostej, parametryzacja gdy znajduje się przynajmniej 10 punktów w komórce

Rysunek 3.9: Zestawienie map z dużą liczbą szczegółów



(a) Mapa bez dodatkowych założeń

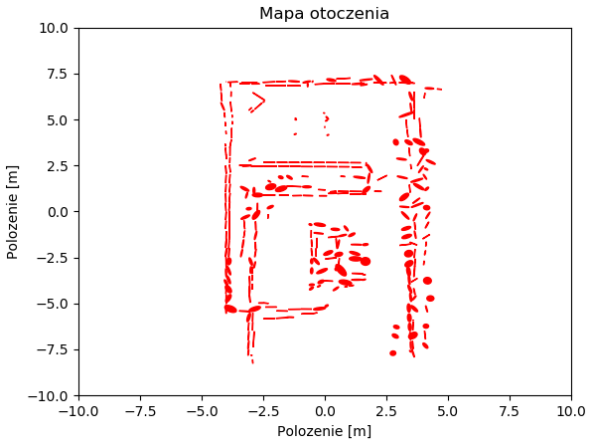


(b) Mapa z założeniami: nie przetwarzanie danych w trakcie obrotu robota, wykorzystanie regresji liniowej do wyznaczenia współczynników prostej, parametryzacja gdy znajduje się przynajmniej 10 punktów w komórce

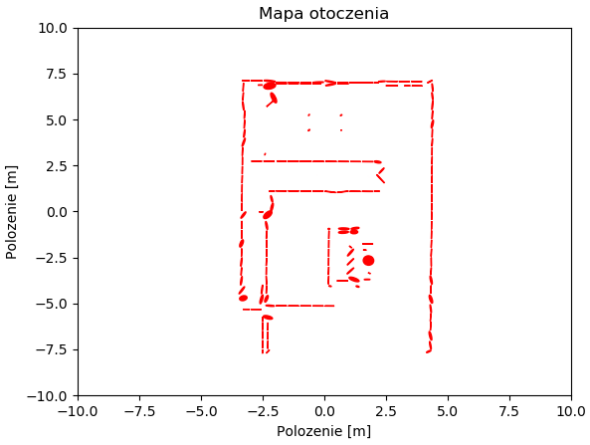
Rysunek 3.10: Zestawienie wybudowanych map Willow Garage

Na podstawie wygenerowanych wykresów można zauważyć, że mapy otoczenia wygenerowane bez ograniczeń (rys. 3.9a, 3.10a), są mniej dokładne niż mapy wygenerowane z nowymi założeniami (rys. 3.9b, 3.10b).

Widząc poprawę dokładności map, zdecydowano się zaimplementować dodatkowe założenie, na podstawie którego przetwarzanie pomiarów następuje tylko wtedy, gdy robot nie będzie się poruszał. Dodano również założenie sprawiające, że parametry (wartości oczekiwane, odchylenia standardowe oraz kąt obrotu), na podstawie których są generowane elipsy, będą średnią arytmetyczną pięciu ostatnich operacji parametryzacji danych komórek. Wybudowane mapy otoczenia zestawiono z mapami wybudowanymi bez żadnych założeń (rys. 3.11 i 3.12). Po lewej stronie umieszczono mapy bez ograniczeń (rys. 3.11a, 3.12a) , po prawej stronie wygenerowano mapy z nowym ograniczeniem (rys. 3.11b, 3.12b).

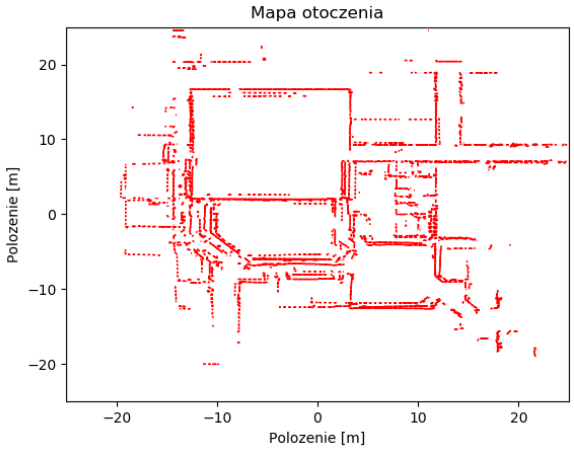


(a) Mapa bez dodatkowych założeń

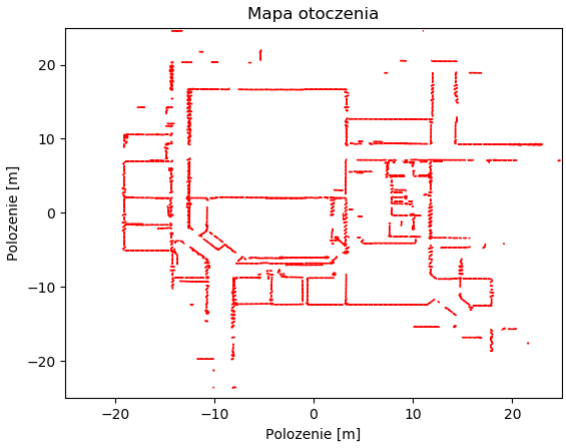


(b) Mapa z założeniami: przetwarzanie danych tylko gdy robot się nie porusza, wyznaczanie elips na podstawie 5 ostatnich parametryzacji

Rysunek 3.11: Zestawienie map z dużą liczbą szczegółów



(a) Mapa bez dodatkowych założeń



(b) Mapa z założeniami: przetwarzanie danych tylko gdy robot się nie porusza, wyznaczanie elips na podstawie 5 ostatnich parametryzacji

Rysunek 3.12: Zestawienie map Willow Garage

Na podstawie przeprowadzonych badań stwierdzono, że najdokładniejsze mapy powstają, gdy robot skanuje pomieszczenie nie poruszając się oraz gdy elipsy generowane są na podstawie średniej arytmetycznej pięciu ostatnich operacji parametryzacji w danej komórce. Zaimplementowane założenia sprawiają, że budowane mapy są dokładne. Należy jednak pamiętać, że każde zatrzymanie w celu skanowania generuje dodatkowy czas przeznaczony na budowę mapy danego pomieszczenia.

Rozdział 4

Podsumowanie

Zgodnie z tezą przedstawioną w pierwszym rozdziale pracy można stwierdzić, że jest możliwe wykorzystanie transformacji opartej na reprezentacji danych za pomocą rozkładu normalnego w celu efektywnego wykonania mapy otoczenia na podstawie informacji zbieranych przez robota mobilnego wyposażonego w skaner laserowy 2D. Taki rodzaj budowania mapy pozwala na przedstawienie jej pojedynczego fragmentu (w postaci elipsy) za pomocą kilku parametrów. Tak zbudowana mapa zajmuje mniej pamięci niż mapa wykonana w technice rastrowej.

W ramach wykonania pracy inżynierskiej zapoznano się z możliwościami ROSa oraz Gazebo. W środowisku ROS stworzono model robota klasy (2,0) a następnie z wykorzystaniem Gazebo przeprowadzono symulację. Wtyczki dostarczone do programu Gazebo pozwoliły na sterowanie robotem oraz na zamodelowanie skanera laserowego 2D. Zdecydowano się podzielić mapę na obszary w kształcie kwadratu o boku długości 0.5 metra. Za pomocą odpowiednich przekształceń matematycznych, każda z próbek, wykonana podczas jednego pomiaru skanera, trafia do odpowiedniego obszaru na mapie. Po przedstawieniu każdej z próbek jako punktu oraz przydzieleniu do odpowiednich komórek na mapie, na podstawie rozkładu normalnego o dwóch parametrach wyznaczono: wartości oczekiwane μ_x i μ_y oraz odchylenia standardowe σ_x oraz σ_y. Po wyznaczeniu wartości oczekiwanej, odchylenia standardowego oraz kąta odchylenia próbek od osi X, wygenerowano na podstawie uzyskanych parametrów elipsy. Elipsy zostają umieszczone na przestrzeni, która odzwierciedla rzeczywiste otoczenie budując mapę. Porównując otrzymane mapy z rzeczywistym otoczeniem, zdecydowano się na wprowadzenie dodatkowych założeń (wykorzystanie regresji liniowej do wyznaczenia współczynników prostej, nieprzetwarzanie danych, gdy robot się obraca oraz ustawienie minimalnej ilości punktów, aby dane mogły być parametryzowane) pozwalających na eliminację błędów związanych z błędnym obliczaniem współrzędnych punktów czy pochyleniem elips. Uzyskane z nowymi założeniami mapy były dokładniejsze. Przeprowadzając dodatkowe badania zauważono, że przetwarzanie danych pochodzących ze skanera laserowego tylko wtedy, gdy robot się nie przemieszcza oraz generowanie elips na podstawie wartości średnich arytmetycznych 5 ostatnich parametryzacji komórek, poprawia dokładność generowanych map. Implementacja opisanych założeń pozwoliła na generowanie dokładnych map otoczenia.

Na podstawie wykonanych badań można zauważyć, że wraz z rezygnacją budowania mapy w czasie ruchu robota, poprawiła się dokładność ich reprezentacji. Jedną z możliwości dalszego rozwoju projektu jest implementacja rozwiązania wspierającego budowanie dokładnych map w czasie poruszania się robota. Tego typu rozwinięcie projektu może być zrealizowane poprzez zaimplementowanie algorytmu pozwalającego na kompensacje wykonanych pomiarów w oparciu o estymację położenia robota.

Analizując badania symulacyjne można zauważyć, że podczas wykonywania pomiarów wybrane punkty zostają umieszczone do błędnej komórki. Prawdopodobnie wynika to położenia lub ruchu robota w czasie wykonywania pomiarów. W celu zniwelowania błędów w przyszłości, można rozwinąć projekt o analizę już powstałych komórek, tak aby na podstawie aktualnych odczytów oraz położenia robota eliminować błędnie wykonane pomiary.

Jedną z wad przeprowadzania symulacji jest idealizacja pomiarów czy brak błędów związanych z niedokładnościami zastosowanych elementów elektronicznych. W celu sprawdzenia algorytmu w rzeczywistym otoczeniu, można zbudować rzeczywistego robota, podobnego do wykorzystanego w projekcie inżynierskim, a następnie sprawdzić na nim działanie zastosowanego algorytmu. Pozwoli to określić czy zaimplementowane rozwiązania i przeprowadzone badania symulacyjne mają odzwierciedlenie w rzeczywistym świecie.

Literatura

[1] P. Biber, W. Strasser. The normal distributions transform: a new approach to laser scan matching. *International Conference on Intelligent Robots and Systems*, strony 2743–2748, Las Vegas, NV, USA, 2003.

[2] M. Cardona, A. Palma, J. Manzanares. Covid-19 pandemic impact on mobile robotics market. *IEEE Andean Conference*, strony 1–4, Quito, Ecuador, 2020.

[3] W. J. Frawley, G. Piatetsky-Shapiro, C. J. Matheus. Knowledge discovery in databases: An overview. *AI Magazine*, 13(3):57, 1992.

[4] D. Huang, C.-D. Wang, J.-S. Wu, J.-H. Lai, C.-K. Kwoh. Ultra-scalable spectral clustering and ensemble clustering. *IEEE Transactions on Knowledge and Data Engineering*, 32(6):1212–1226, 2020.

[5] A. K. Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31(8), 2010.

[6] L. Joseph, J. Cacace. *Mastering ROS for Robotics Programming - Second Edition: Design, Build, and Simulate Complex Robots Using the Robot Operating System*. Packt Publishing Limited, 2018.

[7] J. Li, L. Cheng, H. Wu, L. Xiong, D. Wang. An overview of the simultaneous localization and mapping on mobile robot. *International Conference on Modelling, Identification and Control*, strony 358–364, Wuhan, China, 2012.

[8] MarketsandMarkets Reasearch. Mobile robot market by operating environment, 2018. `https://www.marketsandmarkets.com/Market-Reports/mobile-robots-market-43703276.html`.

Załącznik A

Do pracy załączono płytę DVD zawierającą w poszczególnych katalogach:

`/Praca_inzynierska.pdf` — wersja cyfrowa pracy,

`/Kod_zrodlowy` — kod źródłowy oprogramowania środowiska symulacyjnego oraz budującego mapę otoczenia,

`/Wygenerowane_mapy` — wygenerowane mapy otoczenia.