Streszczenie

Planer ruchu pojazdu z Systemem Automatycznego Parkowania

Streszczam.

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumyeirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diamvoluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Słowa kluczowe: slowo1, slowo2, ...

Abstract

Vehicle path planner with Automatic Parking System

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumyeirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diamvoluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumyeirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diamvoluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Keywords: keyword1, keyword2, ...

Warszawa, dnia .....

Oświadczenie

Oświadczam, że pracę magisterską pod tytułem „Planer ruchu pojazdu z Systemem Automatycznego Parkowania", której promotorem jest prof. dr hab. inż Krzysztof Marciniak wykonałam/em samodzielnie, co poświadczam własnoręcznym podpisem.

.....

Spis treści

Wstęp **11**

1. Cel pracy i dostępne publikacje **13**

1.1. Parking Path Programming Strategy for Automatic Parking System 13

1.2. Easy Path Planning and Robust Control for Automatic Parallel Parking 13

2. Analiza wymagań **14**

2.1. Wymagania funkcjonalne 14

2.1.1. Ekran startowy 14

2.1.2. Edytor map 14

2.1.3. Edytor pojazdów 15

2.1.4. Planer ruchu 15

2.1.5. Wizualizacja 17

2.1.6. Ustawienia 18

2.2. Wymagania niefunkcjonalne 18

3. Technologia **19**

3.1. Qt 5.8 19

3.2. Boost 1.64 20

3.3. NanoVG 20

3.4. OpenGL 21

4. Analiza ruchu pojazdu **22**

4.1. Geometria Ackermana 22

4.2. Wyznaczanie minimalnego promienia skrętu 22

4.3. Wyznaczanie okręgów skrętu 23

5. Algorytmy planowania ruchu i parkowania **24**

5.1. Algorytm planowania ruchu 24

5.1.1. Idea algorytmu 24

5.1.2. Pseudokod algorytmu 30

5.1.3. Testy i dalszy rozwój algorytmu 31

5.2. Algorytm parkowania 31
5.2.1. Analiza problemu parkowania równoległego 31
5.2.2. Idea algorytmu 32
5.2.3. Pseudokod algorytmu 32
5.2.4. Testy i dalszy rozwój algorytmu 33
6. Instrukcja użytkownika 34
7. Rozdział pokazowy 35
7.1. Przykładowa sekcja/podrozdział 35
7.1.1. Podsekcja 35
7.2. Tabele i rysunki 36

Wstęp

Jednym z największych wyzwań jakie czekają na kursantów ubiegających się o wydanie dokumentu prawa jazdy jest nauka manewru parkowania równoległego. Problem ten nie dotyczy tylko młodych kierowców. Niejednokrotnie osoby posiadające prawo jazdy od kilku lat mają problem aby poprawnie zaparkować samochód w dość wąskim miejscu, a wykonanie idealnego manewru parkowania równoległego w jednym lub maksymalnie dwóch ruchach jest prawdziwą sztuką. W dzisiejszych czasach z pomocą przychodzą zaawansowane systemy parkowania automatycznego, które są już na wyposażeniu wszystkich wiodących marek produkujących samochody osobowe.

Zazwyczaj mamy do czynienia z półautomatycznym systemem wspomagającym parkowanie (asystentem), który sam znajduje wystarczającą do zaparkowania ilość miejsca, a także pomaga wykonać samą czynność odpowiednio sterując kierownicą. Jedynym zadaniem kierowcy jest odpowiednie operowanie pedałami gazu i hamulca. Jest to absolutnie podstawowa wersja asystenta parkowania, którą oferują nawet najprostsze i najtańsze rozwiązania.

W przypadku bardziej zaawansowanych systemów parkowanie jest wykonywane w pełni automatycznie, a niekiedy możliwe jest nawet opuszczenie pojazdu przez kierowcę. Rozwiązania najbardziej zaawansowane dostarczają możliwość automatycznego wjazdu lub wyjazdu pojazdu z garażu bez obecności kierowcy w samochodzie.

Jak w praktyce wygląda działanie asystenta parkowania? Po pierwsze kierowca musi wybrać interesujący go tryb parkowania (prostopadłe lub równoległe), gdyż system zazwyczaj nie jest w stanie samodzielnie tego określić. Następnie rozpoczyna się proces wyszukiwania wolnego miejsca w czasie którego trzeba w miarę wolno jechać do przodu (zwykle maksymalnie 20-30 km/h). Gdy stosowne miejsce zostanie odnalezione komputer prosi o zatrzymanie się i najczęściej o włączenie biegu wstecznego. Od tego momentu komputer przejmuje pełną lub część kontroli nad pojazdem, w zależności od zaawansowania systemu. Oczywiście kierowca w dowolnym momencie może przerwać działanie systemu i samodzielnie kontynuować wykonywanie manewru. Cały proces parkowania zajmuje od kilkunastu do kilkudziesięciu sekund.

Zupełnie inną kategorię systemów wspomagających kierowcę stanowią autopiloty. Obecnie najwięksi producenci aut na świecie testują swoje rozwiązania , a rezultaty tych testów są bardzo obiecujące. Dla przykładu w grudniu 2016 roku konwój ciężarówek z powodzeniem zakończył testowy przejazd po Europie. Ciężarówki takich firm jak DAF, Daimler, IVECO, MAN, Scania i Volvo wyruszyły z różnych miejsc w Europie, m.in. ze Szwecji, Belgii i Niemiec. Za cel obrały Rotterdam w Holandii. Ciężarówki jechały samodzielnie w kolumnach składających się z 2-3 pojazdów i zachowywały stałe tempo jazdy. Były połączone ze sobą za pomocą Wi-Fi. To ciężarówka z przodu ustalała trasę i prędkość, a pozostałe za nią podążały. Nad komputerem czuwał jednak człowiek, który siedział na miejscu kierowcy. Wszystkie ciężarówki szczęśliwie dojechały do Rotterdamu.

Innym przykładem ukazującym potencjał jaki drzemie w samochodowych autopilotach jest sytuacja jaka miała miejsce również w grudniu 2016 roku w Holandii. Kierowca poruszający się pojazdem marki Tesla Model X uniknął kolizji na autostradzie dzięki wczesnej reakcji systemu autopilota, który wykonał hamowanie, podczas gdy dla kierowcy niebezpieczeństwo nie było jeszcze widoczne przez kolejny ułamek sekundy. Widać więc, że zaawansowane systemy, w które obecnie wyposażane są samochody nie tylko podnoszą komfort jazdy, ale niekiedy potrafią ratować również ludzkie życie.

Niniejsza praca magisterska miała na celu opracowanie systemu, który umożliwia bezkolizyjny przejazd pojazdu z wyznaczonego miejsca i orientacji do miejsca docelowego z uwzględnieniem modułu automatycznego parkowania. Użytkownik otrzymuje do dyspozycji edytor pozwalający na tworzenie/edycję map po których ma poruszać się pojazd, planer ruchu, który odpowiedzialny jest za wyznaczenie ścieżki, biorąc pod uwagę różne ustawienia wprowadzane przez użytkownika oraz moduł wizualizacji, który jest dostępny zarówno w postaci wizualizacji 2D jak i 3D.

W celu stworzenia aplikacji wykorzystana została technologia C++, natomiast interfejs użytkownika został wykonany z wykorzystaniem zestawu bibliotek Qt. Do stworzenia edytorów oraz wizualizacji 2D wykorzystano bibliotekę NanoVG. Wizualizacja 3D została przygotowana w całości w technologii OpenGL.

1. Cel pracy i dostępne publikacje

Tematem niniejszej pracy dyplomowej było opracowanie planera ruchu pojazdu wraz z systemem automatycznego parkowania. Podczas implementacji systemu wykorzystano umiejętności zdobyte na studiach drugiego stopnia, specjalność „Projektowanie systemów CAD/CAM", jak również informacje zawarte w kilku publikacjach dostępnych w Internecie. Ich krótki przegląd przedstawiono poniżej.

1.1. Parking Path Programming Strategy for Automatic Parking System

Autorami tej publikacji są Jian-Min Wang, Sen-Tung Wum Chao-Wei Ke oraz Bo-Kai Tzeng. W publikacji opisany został algorytm wyznaczania ścieżki umożliwiającej zaparkowanie pojazdu w miejscu przeznaczonym do parkowania równoległego. Rozwiązanie zaproponowane przez autorów opiera się na założeniu, że parkowanie będzie odbywać się w jednym manewrze. Takie założenie wymusza więc, ze miejsce parkingowe musi mieć odpowiednią długość. Sam manewr parkowania sprowadza się do ruchu pojazdu po dwóch okręgach, które z założenia mają mieć identyczne promienie. Na początku pojazd porusza się po pierwszym z okręgów, a następnie następuje zmiana okręgu w momencie gdy tylna oś pojazdu znajduje się na wysokości początku miejsca parkingowego. Rozwiązanie przedstawione w tej publikacji stało się podstawą algorytmu parkowania zaimplementowanego na potrzeby tej pracy.

Autorzy publikacji poszli o krok dalej i poza opisem samego algorytmu parkowania zaprezentowali jego działanie w praktyce. W tym celu skonstruowali miniaturowy model samochodu, który wyposażyli w odpowiednie czujniki. Parkowanie pojazdu przebiegło pomyślnie, jednak ta część publikacji nie była już istotna z punktu widzenia niniejszej pracy.

1.2. Easy Path Planning and Robust Control for Automatic Parallel Parking

2. Analiza wymagań

2.1. Wymagania funkcjonalne

Przygotowana aplikacja powinna zostać podzielona na 5 wymienionych poniżej modułów i dla każdego z nich spełniać określone wymagania funkcjonalne.

2.1.1. Ekran startowy

Ekran startowy to pierwszy widok ukazywany w aplikacji po jej uruchomieniu przez użytkownika. Powinien on zawierać informacje mówiące o tym czym jest ta aplikacja i w jakim celu powstała. Ponadto powinna być możliwość łatwego rozpoczęcia pracy z aplikacją bezpośrednio z ekranu startowego.

2.1.2. Edytor map

Głównym zadaniem edytora map jest umożliwienie użytkownikowi stworzenia nowej mapy o określonych wymiarach i zapisania jej do pliku. Ponadto możliwe powinno być również wczytanie uprzednio zapisanej mapy i dalsza jej edycja.

Po utworzeniu nowej mapy użytkownik powinien mieć możliwość dodawania następujących rodzajów obiektów:

- budynków – podstawowym „budynkiem" powinna być prosta kostka sześcienna. Jest to wymaganie wstępne mające na celu ułatwienie procesu wytwarzania i testowania aplikacji. Po pomyślnym zaimplementowaniu obsługi zwykłego sześcianu jako budynku do aplikacji powinno zostać dodane kilka innych budynków posiadających własne siatki i tekstury,
- dekoracji – użytkownik powinien mieć możliwość umieszczania na mapie elementów dekoracyjnych takich jak drzewo, ławka itp. Na początek jednak podobnie jak w przypadku budynków wystarczy zaimplementować obsługę prostego sześcianu jako obiektu dekoracyjnego,
- miejsc parkingowych – powinno być możliwe dodawanie dwóch rodzajów miejsc parkingowych, przeznaczonych do parkowania równoległego oraz parkowania prostopadłego. Po-

nownie należy najpierw zaimplementować dwa typy będące prostymi powierzchniami, a następnie dodać modele o określonych wymiarach i z nałożonymi teksturami,

- pojazdów - .

Dla każdego z dodawanych obiektów użytkownik powinien mieć możliwość ustalenia jego nazwy oraz wykonywania na nim następujących operacji:

- przesuwania (translacji)
- rotacji
- skalowania

Operacja skalowania nie jest zalecana dla obiektów ze zdefiniowanymi siatkami i teksturami, czyli niebędących prostymi sześcianami.

Dodatkowo podczas umieszczania obiektu na mapie oraz późniejszej jego edycji powinno się sprawdzać, czy obiekt ten nie koliduje z żadnym innym obiektem już umieszczonym na mapie oraz czy w całości mieści się w granicach mapy.

Kolejnym wymaganiem dla edytora map jest dodanie drzewa obiektów zawierającego wszystkie dodane do mapy obiekty. Przy tworzeniu drzewa powinien zostać uwzględniony podział na następujące gałęzie: budynki, dekoracje, miejsca parkingowe.

Ostatnim wymaganiem jest umożliwienie użytkownikowi zaznaczania dodanego już obiektu w celu jego ponownej jego edycji zarówno z poziomu drzewa obiektów jak i poprzez najechanie nad obiekt kursorem myszki i jego wybór.

2.1.3. Edytor pojazdów

2.1.4. Planer ruchu

Głównym zadaniem planera ruchu jest obliczanie trajektorii po jakiej ma poruszać się pojazd pomiędzy wskazanymi przez użytkownika położeniami lub miejscami parkingowymi. Planowanie ruchu powinno być dostępne dla następujących wariantów przejazdu:

- położenie -> położenie
- miejsce parkingowe -> położenie
- położenie -> miejsce parkingowe
- miejsce parkingowe -> miejsce parkingowe.

Przed rozpoczęciem wyznaczania ścieżki, po której będzie poruszał się pojazd, użytkownik powinien skonfigurować następujące rzeczy:

- wybór mapy,
- wybór pojazdu,
- ustawienie położenia początkowego i końcowego.

Następnie powinna aktywować się opcja wyznaczania ścieżki, która do tej pory powinna być nieaktywna. Po jej wybraniu użytkownik powinien mieć możliwość skonfigurowania kolejnych parametrów, tym razem wykorzystywanych przez planer ruchu. W skład tych parametrów wchodzą:

- procentowa wartość informująca algorytm o jaki procent szerokości pojazdu mają zostać powiększone przeszkody,
- rodzaj algorytmu planowania (do wyboru Krzywa sklejana B-Spline lub Metoda wpisanych łuków),
- w przypadku wyboru algorytmu wykorzystującego wpisane łuki, informacja czy pojazd może wykorzystywać dowolne czy tylko dopuszczalne dla swojej skrętności łuki,
- jakość sprawdzania kolizji pojazdu z innymi obiektami na mapie.

Po zakończeniu wyznaczania ścieżki użytkownik powinien zostać poinformowany o powodzeniu lub niepowodzeniu tej operacji.

Kolejnym wymaganiem dotyczącym planera ruchu jest to, że w dowolnym momencie pracy powinna być możliwość resetu lub zapisania obecnych efektów pracy lub wczytania uprzednio zapisanych danych z pliku.

Po zakończeniu obliczeń użytkownik powinien mieć możliwość włączania i wyłączania wyświetlania następujących obiektów w celu zaprezentowania kolejnych kroków działania algorytmu planowania ruchu:

- powiększonych przeszkód na mapie,

- grafu Voronoi,
- pełnego grafu Visibility-Voronoi,
- wyznaczonej łamanej będącej ścieżką pomiędzy zadanymi położeniami lub miejscami parkingowymi (o ile istnieje),
- wyznaczonych ścieżek parkowania (o ile takie istnieją),
- finalnie wyznaczonej ścieżki (o ile istnieje).

2.1.5. Wizualizacja

Głównym zadaniem modułu wizualizacji jest wyświetlanie symulacji przejazdu samochodu zarówno w trybie 2D jak i 3D. Moduł wizualizacji powinien być wyposażony w listę symulacji, na której znajdują się uprzednio wczytane symulacje, które są gotowe do wyświetlenia użytkownikowi. Symulacje mogę być dodawane na listę na dwa sposoby:

- import z modułu planera ruchu, pod warunkiem, że została tam przygotowana prawidłowa symulacja,
- wczytywane z uprzednio zapisanych plików.

Po zaznaczeniu symulacji na liście powinny aktywować się następujące opcje:

- usunięcie symulacji z listy,
- wyświetlenie szczegółowych informacji o symulacji,
- możliwość odtworzenia/pauzowania lub zastopowania wybranej symulacji.

Ponadto powinna być możliwość wyczyszczenia listy wczytanych symulacji oraz ustawienia czasu trwania symulacji przejazdu.

Okno ze szczegółowymi informacjami na temat symulacji powinno zawierać następujące informacje:

- długość znalezionej ścieżki,
- informację o jaki procent szerokości pojazdu powiększono obiekty na mapie,
- informację na temat wykorzystanego algorytmu planowania ruchu,

- informację, czy pojazd może wykonywać tylko dopuszczalne skręty,
- informację na temat dokładności sprawdzania kolizji wykorzystanej przy planowaniu ruchu.

2.1.6. Ustawienia

2.2. Wymagania niefunkcjonalne

Poza wymienionymi powyżej wymaganiami funkcjonalnymi aplikacja powinna spełniać również pewne wymagania niefunkcjonalne. Najważniejszym z nich jest poprawne działanie aplikacji na systemach operacyjnych z rodziny Windows. Ponadto aplikacja powinna łatwo dać się tłumaczyć na inne języki oraz posiadać wbudowany język polski i angielski, a jej interfejs powinien być responsywny i prezentować się dobrze na monitorach o różnych rozdzielczościach.

3. Technologia

W tym rozdziale zostaną przedstawione technologie, które wykorzystano w celu realizacji wymagań opisanych w poprzednim rozdziale.

Całość aplikacji została napisana w języku C++ z wykorzystaniem zestawu bibliotek ułatwiających tworzenie graficznego interfejsu użytkownika – Qt w wersji 5.8. Ponadto w celu implementacji zapisywania i wczytywania map oraz symulacji, a także do konstrukcji grafu Voronoi wykorzystano kolekcję bibliotek Boost w wersji 1.64. Edytory graficzne takie jak edytor map czy planer ruchu zostały zaimplementowane z wykorzystaniem biblioteki NanoVG, natomiast do wizualizacji 3D użyta została technologia OpenGL w połączeniu z kilkoma innymi bibliotekami, co zostanie opisane poniżej.

3.1. Qt 5.8

Qt to zestaw przenośnych bibliotek i narzędzi programistycznych dedykowanych dla języków C++, QML i Java. Podstawowym składnikiem bibliotek wchodzących w skład Qt są klasy służące do budowy graficznego interfejsu użytkownika. Interfejs można budować pisząc kod lub przy pomocy edytora graficznego. Podczas stylizacji poszczególnych kontrolek możliwe jest użycie styli CSS. Poniżej zaprezentowano przykład stylizacji przycisku oraz zmiany jego wyglądu na akcje najechania myszką nad przycisk oraz jego wciśnięcia. W podobny sposób można zmieniać wygląd dowolnego elementu wchodzącego w skład interfejsu użytkownika.

```
// przykład qpushbutton
// może jakiś obrazek
```

Począwszy od wersji 4.0 Qt zawiera również narzędzia służące do tworzenia programów konsolowych i serwerów. W opracowanym systemie Qt został wykorzystany tylko do stworzenia graficznego interfejsu użytkownika.

3.2. Boost 1.64

Boost jest zbiorem bibliotek programistycznych rozszerzających możliwości języka C++. Dziedziny zastosowania Boost są bardzo szerokie. Pakiet dostarcza m.in. biblioteki ogólnego przeznaczenia (inteligentne wskaźniki, wyrażenia regularne), biblioteki stanowiące warstwę abstrakcji dla systemu operacyjnego (obsługa systemów plików czy wielowątkowości), jak i narzędzia przeznaczone głównie dla innych twórców bibliotek i zaawansowanych programistów języka C++ (np. biblioteka metaprogramowania MPL). Kilka bibliotek wchodzących w poczet Boost zostało włączonych do pierwszego raportu technicznego komitetu standaryzacyjnego C++, który jest zbiorem proponowanych rozszerzeń biblioteki standardowej języka C++.

W przygotowanej aplikacji kolekcja bibliotek Boost została wykorzystana do implementacji zapisywania i wczytywania map oraz symulacji, a także do konstrukcji grafu Voronoi na podstawie przygotowanej przez użytkownika mapy. Przykład wykorzystania biblioteki Boost do serializacji obiektu został przedstawiony poniżej.

// przykład

3.3. NanoVG

NanoVG jest kompaktową biblioteką wspierającą renderowanie grafiki wektorowej z antyaliasingiem dla OpenGL. Interfejs biblioteki jest bardzo zbliżony do interfejsu jaki dostarcza obiekt Canvas z HTML5 dzięki czemu osoby znające HTML5 mogą niemal natychmiast rozpocząć efektywną pracę z biblioteką NanoVG. Biblioteka ta powstała z myślą o dostarczeniu niewielkiego i prostego API, przy pomocy którego możliwe będzie tworzenie skalowalnych interfejsów użytkownika, wizualizacji czy wszelkiego rodzaju edytorów. Przykład użycia biblioteki przedstawiono poniżej.

// przykład

Potencjał biblioteki NanoVg ukazuje przykładowy interfejs użytkownika przygotowany przez jej autorów. Poniżej przedstawiono zrzut ekranu prezentujący ten interfejs.

//zdj

3.4. OpenGL

OpenGL obok DirectX i Vulkan API jest podstawową niskopoziomową biblioteką graficzną 3D, obsługiwaną przez wszystkie liczące się systemy operacyjne oraz większość dostępnych na rynku procesorów graficznych. Niezależność od platformy sprzętowej oraz ogólnie dostępna specyfikacja czyni z OpenGL standard powszechnie wykorzystywany przez producentów oprogramowania użytkowego i gier. Zestaw funkcji OpenGL składa się z 250 podstawowych wywołań, umożliwiających budowanie złożonych trójwymiarowych scen z podstawowych figur geometrycznych.

Dzięki wykorzystaniu OpenGL w opracowanej aplikacji możliwe było stworzenie wydajnej wizualizacji 3D. Jako, że czysty OpenGL skupia się na dostarczaniu API służącego do komunikacji z procesorem karty graficznej i nie posiada takich narzędzi jak np. parsery plików z siatkami, to konieczne było użycie dodatkowych, dedykowanych bibliotek dla OpenGL. Najważniejsze z tych bibliotek wymieniono poniżej:

- SOIL
- ASSIMP

Użycie powyższych bibliotek w połączeniu z OpenGL jest w dzisiejszych czasach standardem.

4. Analiza ruchu pojazdu

Przy analizie ruchu pojazdu w opracowanym systemie uwzględniona została tzw. geometria Ackermana (lub mechanizm kompensacji Ackermana), aby w możliwie realistyczny sposób opisać ruch prawdziwego pojazdu. Ponadto przyjęto założenie, że w każdym momencie ruchu pojazd porusza się z prędkością, która nie powoduje poślizgu bocznego. W poniższej analizie jako pozycję pojazdu przyjęto środek jego tylnej osi, natomiast w aplikacji ze względów technicznych jako środek pojazdu przyjmowany jest geometryczny środek otaczającego go prostokąta. Nie zmienia to jednak w żaden sposób opisywanych metod i specyfiki problemu.

4.1. Geometria Ackermana

Oczywistym jest fakt, iż podczas ruchu pojazdu w linii prostej oba przednie (skrętne) koła pozostają równoległe do siebie, a ich kąt skręcenia wynosi 0 stopni. Jednak w przypadku pokonywania nawet najmniejszego zakrętu, kąty skręcenia obu kół stają się różne. Za odpowiedni kąt skręcenia każdego z kół odpowiada mechanizm kompensacji Ackermana. Zastosowanie tego mechanizmu w pojazdach jest niezbędne, aby mogły one poruszać się po krzywiźnie bez poślizgu kół jezdnych, który mógłby spowodować utratę kontroli nad pojazdem, a w najlepszym wypadku przedwczesne zużycie się opon.

4.2. Wyznaczanie minimalnego promienia skrętu

Minimalny promień skrętu każdego pojazdu zależny jest od maksymalnego kąta skręcenia jego przednich kół, a precyzyjniej od maksymalnego kąta skręcenia koła znajdującego się po wewnętrznej stronie skrętu (co wynika z geometrii Ackermana). Samo pojęcie minimalnego promienia skrętu również wymaga doprecyzowania, gdyż można rozważać wiele różnych promieni skrętu, w zależności od przyjętego punktu odniesienia na pojeździe. W poniższej analizie wyznaczone zostaną trzy promienie skrętu:

- dla tylnego koła znajdującego się po wewnętrznej stronie skrętu (Rmin),

- dla przedniego koła znajdującego się po zewnętrznej stronie skrętu (Rmax),
- dla środka poruszającego się pojazdu (R).

Na poniższym rysunku zaznaczone zostały poszukiwane promienie.

// rysunek

Do wykonania obliczeń zakładamy, że znamy rozstaw osi oraz rozstaw kół pojazdu, a także kąt skręcenia wewnętrznego koła przedniego, który przy wyznaczaniu minimalnego promienia skrętu musi być równy maksymalnemu kątowi skrętu kół dla danego pojazdu (na ogół wartość ta waha się w granicach 36-42 stopnie). Zaznaczone na rysunku promienia obliczamy w następujący sposób:

// obliczenia

4.3. Wyznaczanie okręgów skrętu

Pojazd znajdujący się w określonym położeniu posiada zawsze dwa okręgi po których natychmiast może zacząć się poruszać przy zadanym kącie skrętu kół w jedną lub drugą stronę. W systemie przyjęto założenie, że koła są zawsze maksymalnie skręcone, tak aby pojazd jak najwięcej poruszał się po liniach prostych, a zakręty pokonywał jak najszybciej.

W poprzednim podrozdziale została opisana metoda wyznaczania różnych promieni skrętu. Do opisu pojedynczego okręgu skrętu oprócz jego promienia potrzebna jest informacja na temat położenia jego środka. Uzyskanie takiej informacji nie stanowi większego problemu i środki obu okręgów można obliczyć w następujący sposób:

// obliczenia

Proces wyznaczania środka okręgu skrętu dla pojazdu został przedstawiony na poniższym rysunku.

// rysunek

5. Algorytmy planowania ruchu i parkowania

Podczas implementacji aplikacji opracowano algorytm planowania ruchu pojazdu pomiędzy zadanymi położeniami oparty na krzywych sklejanych oraz na wpisywaniu łuków w łamaną złożoną z krawędzi grafu oraz algorytm parkowania równoległego i prostopadłego. Opracowane algorytmy zostały opisane w dalszej części tego rozdziału.

5.1. Algorytm planowania ruchu

Algorytmy planowania ruchu pojazdu można podzielić na dwa rodzaje - takie w których możliwe jest sprowadzenie pojazdu do punktu poprzez odpowiednie powiększenie przeszkód oraz takie w których tego zrobić nie można. Algorytmy pierwszego rodzaju można stosować przy założeniu, że pojazd, dla którego planowana jest trajektoria może obracać się w miejscu, wokół uprzednio zdefiniowanej osi obrotu. W takim wypadku przy obliczaniu trajektorii najczęściej wykorzystywane są tzw. grafy widoczności. Jeśli zaś sprowadzenie pojazdu do punktu nie jest możliwe to do wyznaczenia trajektorii można wykorzystać graf Voronoi.

Ponieważ w niniejszej pracy rozważany jest ruch pojazdu możliwie jak najbardziej zbliżony do ruchu prawdziwego samochodu, to algorytmy oparte o grafy widoczności zostaną pominięte i rozważane będą tylko algorytmy, w których niemożliwe jest zastąpienie pojazdu pojedynczym punktem.

5.1.1. Idea algorytmu

Podstawowymi parametrami wejściowymi dla algorytmu planowania ruchu pojazdu są następujące obiekty:

- mapa
- pojazd - zawiera informację o maksymalnym skręcie kół, rozstawie osi oraz rozstawie kół,
- pozycja początkowa, pozycja końcowa.

Zdefiniowane powyżej parametry są niezbędne do rozpoczęcia planowania ruchu pojazdu. Sam proces wyznaczania trajektorii w najprostszej wersji można podzielić na następujące etapy:

- jeśli położeniem początkowym lub końcowym jest miejsce parkingowe, to należy wyznaczyć trajektorię parkowania zgodnie z algorytmem, który zostanie opisany w następnym podrozdziale,
- utworzyć graf Voronoi, dla zadanej przez użytkownika mapy oraz dodać do niego wierzchołki reprezentujące położenie początkowe i końcowe oraz krawędzie do tych wierzchołków,
- wyznaczyć ścieżkę w grafie pomiędzy wierzchołkiem początkowym i końcowym, np. za pomocą algorytmu A*,
- na podstawie wyznaczonej łamanej, którą tworzą krawędzie wchodzące w skład ścieżki w grafie wyznaczyć trajektorię dopuszczalną dla pojazdu - przy pomocy krzywej sklejanej lub wpisując łuki pomiędzy kolejne odcinki łamanej.

Skuteczność opisanego powyżej algorytmu w praktyce była bardzo niewielka, dlatego na potrzeby tej pracy przebadano i opracowano następujące modyfikacje przedstawionego powyżej schematu:

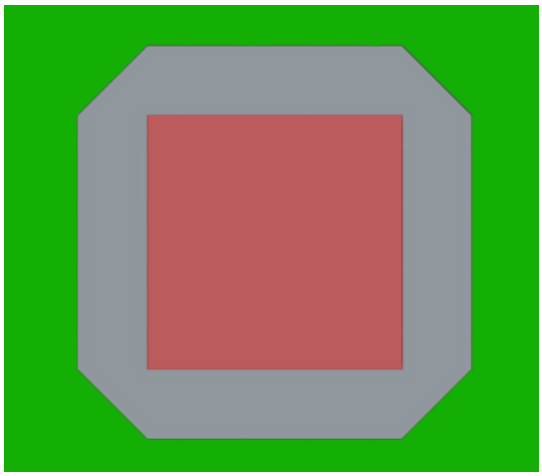
- powiększanie przeszkód,
- modyfikacja grafu Voronoi i ewentualne uzupełnienie go o dodatkowe wierzchołki i krawędzie,
- iteracyjne sprawdzanie i eliminacja kolizji.

W następnych podrozdziałach zostaną opisane wymienione powyżej modyfikacje, a także zostanie opisana metoda konstrukcji finalnej ścieżki oparta na krzywej sklejanej oraz wpisywaniu łuków pomiędzy odcinki łamanej.

Powiększanie przeszkód

Celem powiększania przeszkód na mapie jeszcze przed wyznaczeniem grafu Voronoi jest minimalizacja ryzyka kolizji pojazdu z danym obiektem, poprzez oddalenie krawędzi grafu od brzegów obiektu. Algorytm powiększania obiektów mapy jest bardzo prosty i sprowadza się do przesunięcia każdego wierzchołka wzdłuż normalnej do danej krawędzi o odległość zdefiniowaną przez użytkownika (procentowa wartość szerokości pojazdu). Efektem zastosowania

takiego algorytmu powiększania jest powstawanie przeszkód, które mają aż 8 wierzchołków ze standardowych przeszkód czworokątnych. Dzieje się tak, gdyż tak naprawdę każdy wierzchołek modyfikowanej przeszkody wejściowej jest przesuwany dwukrotnie, najpierw wzdłuż normalnej do jednej krawędzi, a następnie wzdłuż normalnej do krawędzi do niej prostopadłej. Przykład przeszkody przed powiększeniem i po powiększeniu zaprezentowano na obrazku poniżej.

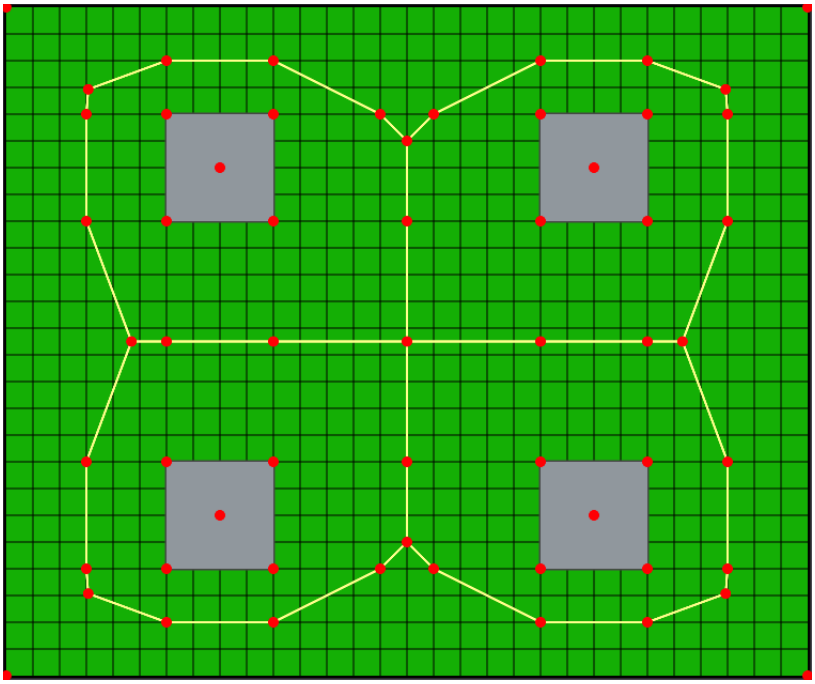


Rysunek 5.1: Przykład powiększania przeszkody

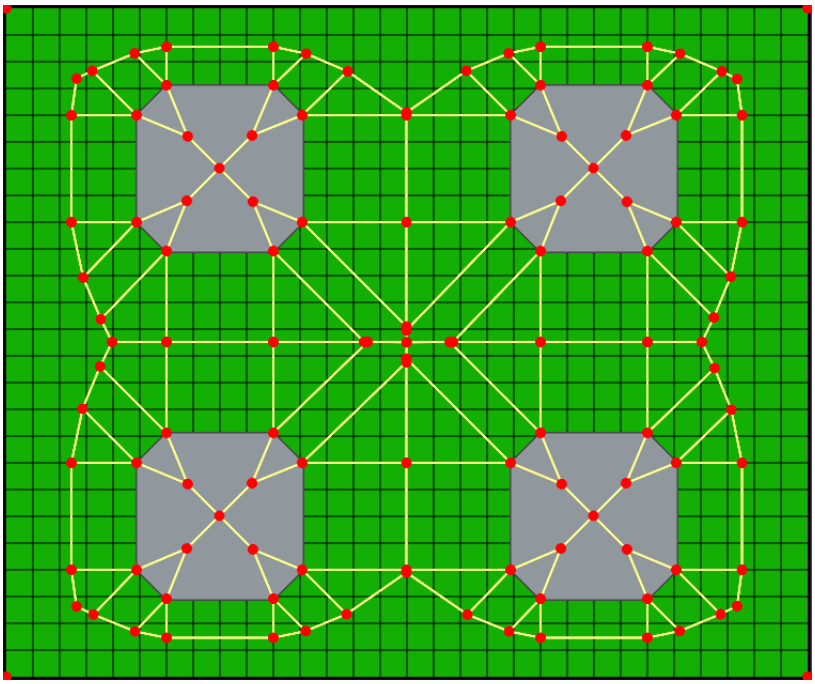
Dodatkowym efektem powiększenia przeszkód jest zupełnie inna postać wyznaczonego grafu Voronoi. Graf wyznaczony dla mapy ze standardowymi przeszkodami przed powiększeniem jest dużo prostszy i na pierwszy rzut oka ćzytelniejszy", jednak bardziej wartościowy z punktu widzenia algorytmu planowania ruchu jest graf wyznaczony dla mapy z powiększonymi przeszkodami, ponieważ zawiera więcej wierzchołków i krawędzi, oraz minimalizuje ryzyko kolizji z przeszkodami poprzez zachowanie odpowiedniej odległości krawędzi od przeszkód. Porównanie obu grafów przedstawiają poniższe rysunki 5.2 oraz 5.3.

Konstrukcja grafu i wyszukiwanie ścieżki

Jak nie trudno się domyślić konstrukcja grafu, w którym wyszukiwana będzie ścieżka pomiędzy wierzchołkiem startowym i końcowym rozpoczyna się od wyznaczenia grafu Voronoi dla zadanej mapy (z powiększonymi przeszkodami lub nie). Ponieważ doświadczenia empiryczne pokazały, że sam graf Voronoi bardzo często zawiera zbyt mało wierzchołków i krawędzi, aby możliwe było wyznaczenie sensownej trajektorii dla pojazdu należało w jakiś sposób wzbogacić ten graf o nowe wierzchołki i krawędzie. Podstawowy pomysł polega na dodaniu krawędzi pomiędzy wszystkimi wierzchołkami, które są dla siebie na wzajem "widoczne", tzn. krawędź pomiędzy tymi wierzchołkami nie przechodzi przez żaden obiekt mapy (lub powiększony obiekt mapy). Przykładowy graf z dodatkowymi krawędziami przedstawia poniższy rysunek.

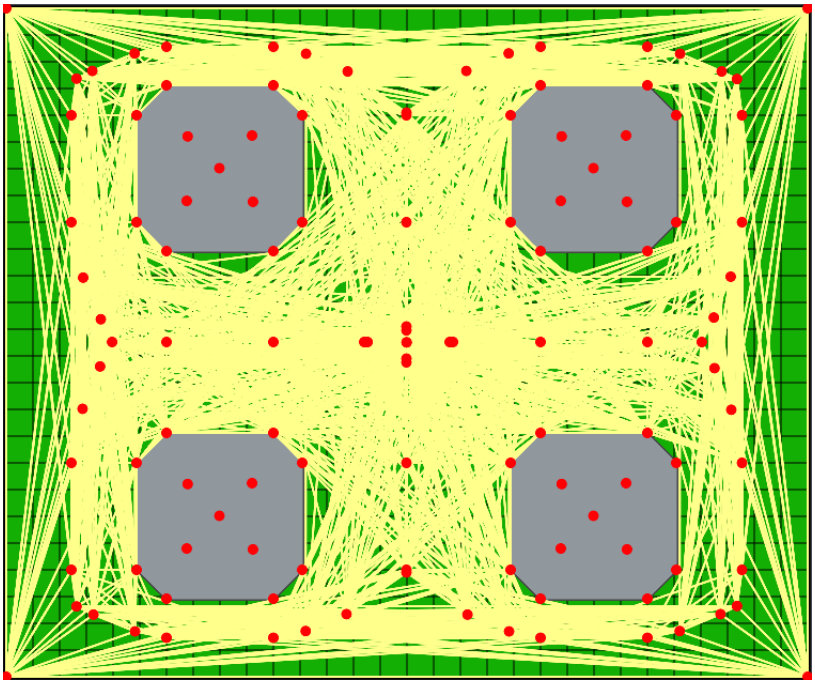


Rysunek 5.2: Graf Voronoi dla mapy z przeszkodami przed powiększeniem



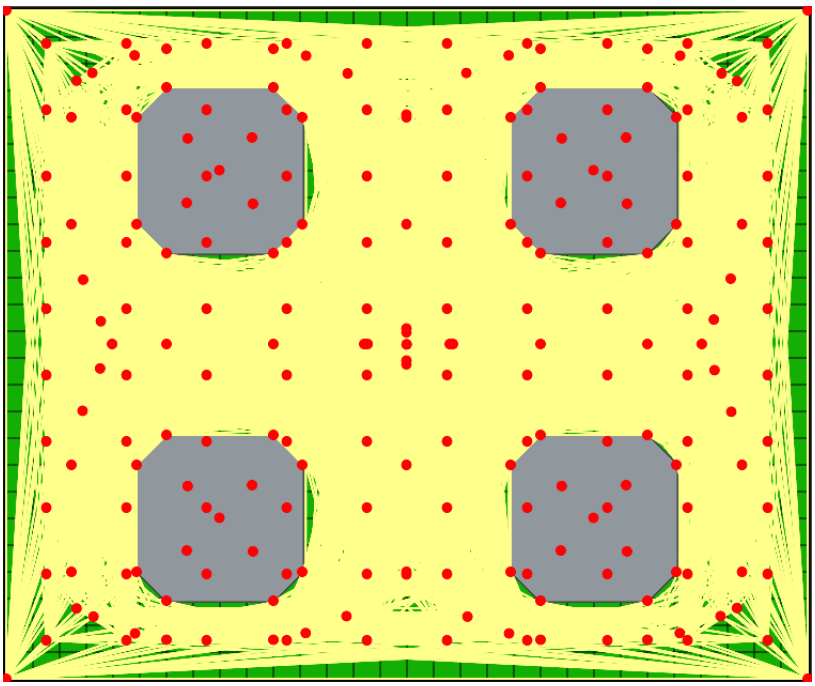
Rysunek 5.3: Graf Voronoi dla mapy z przeszkodami po powiększeniu

W znakomitej większości przypadków taka modyfikacja grafu jest wystarczająca i pozwala na wyznaczenie sensownej trajektorii. Zdarzają się jednak sytuacje, w których okazuje się, że w grafie wyznaczonym powyższą metodą wciąż jest zbyt mało wierzchołków i krawędzi, żeby np. pojazd mógł się przecisnąć pomiędzy gęsto i ciasno rozlokowanymi przeszkodami. W tej sytuacji rozwiązaniem może być dodanie nowych, sztucznych wierzchołków oraz krawędzi pomiędzy



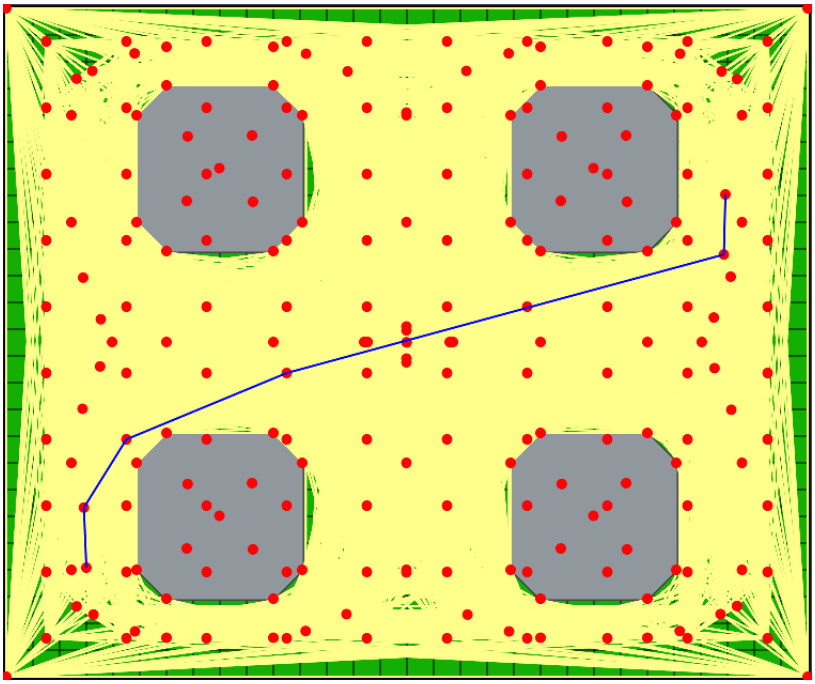
Rysunek 5.4: Graf Voronoi z dodatkowymi krawędziami

tymi wierzchołkami, a wszystkimi wierzchołkami, dla których krawędź nie będzie kolidować z powiększonymi przeszkodami mapy. Graf będący wynikiem takiej operacji (dodane po 10 wierzchołków w pionie i w poziomie) przedstawiono na poniższym rysunku. Czytelność takiego grafu jest już jednak praktycznie zerowa, co wynika z ogromnej liczby krawędzi, które są tak gęsto rozmieszczone, iż niemal w całości pokrywają całą dostępną przestrzeń na mapie.



Rysunek 5.5: Graf Voronoi z dodatkowymi wierzchołkami i krawędziami

Ostatnim krokiem w konstrukcji grafu jest dodanie do niego dwóch wierzchołków początkowych połączonych krawędzią (krawędź początkowa) oraz dwóch wierzchołków końcowych połączonych krawędzią (krawędź końcowa). Następnie należy połączyć końcowy wierzchołek krawędzi początkowej ze wszystkimi możliwymi wierzchołkami grafu. To samo należy zrobić z początkowym wierzchołkiem krawędzi końcowej. Przykładową ścieżkę znalezioną w grafie dla pozycji startowej znajdującej się w lewym dolnym rogu mapy i pozycji końcowej znajdującej się w prawym górnym rogu mapy przedstawiono na poniższym rysunku.



Rysunek 5.6: Ścieżka wyznaczona w grafie pomiędzy położeniem początkowym i końcowym

Konstrukcja finalnej ścieżki za pomocą krzywej B-sklejanej

Krzywa B-sklejana jest jedną z najczęściej stosowanych reprezentacji parametrycznych krzywych sklejanych. Krzywą B-sklejaną charakteryzują dwa parametry:

- **n** - stopień sklejanych krzywych wielomianowych (w przypadku niniejszej pracy stopień krzywych równy 3),

- **m** - liczba podprzedziałów, na których definiowane są kolejne części krzywej.

Konstrukcja finalnej ścieżki za pomocą wpisywania łuków pomiędzy odcinki łamanej

Wykrywanie i eliminacja kolizji

W zaimplementowanej aplikacji mechanizm wykrywania i eliminacji kolizji może zostać w dowolnym momencie włączony lub wyłączony przez użytkownika oraz to użytkownik określa jego dokładność. Jak nietrudno się domyślić wraz ze wzrostem dokładności wykrywania kolizji i ich eliminacji wzrasta też ilość czasu, która jest potrzebna na wykonanie niezbędnych obliczeń.

Sam proces wykrywania kolizji jest dość prosty. Idea jest taka, że symulujemy ruch pojazdu po wyznaczonej w poprzednich krokach trajektorii uaktualniając jego stan do stanu jaki miałby znajdując się w danym punkcie ścieżki. Następnie algorytm sprawdza, czy występuje kolizja pojazdu z którąś z przeszkód znajdujących się na mapie. Jeśli nie, to wszystko jest w porządku i sprawdzane jest kolejne położenie. Jeśli natomiast kolizja wystąpiła, to odnajdywana jest krawędź grafu, która z największym prawdopodobieństwem odpowiada za kształt danego fragmentu ścieżki i następuje usunięcie tej krawędzi z grafu. Następnie wyznaczana jest nowa trajektoria i cały proces zaczyna się od początku. Liczba punktów ścieżki dla których sprawdzana jest kolizja pojazdu z przeszkodami jest definiowana przez użytkownika i od liczby tych punktów zależy jakość i czas działania algorytmu wykrywania i eliminacji kolizji. Dla uporządkowania poniżej przedstawiono pseudokod opisanego algorytmu.

Pseudokod 5.1.1.1. Wykrywania i eliminacji kolizji

```
dla liczby położeń zdefiniowanych przez użytkownika
    zaktualizuj stan pojazdu do określonego położenia
    sprawdź kolizję pojazdu z przeszkodami znajdującymi się na mapie
    jeśli nie występuje kolizja
        kontynuuj
    w przeciwnym wypadku
        znajdź krawędź, która odpowiada za dany fragment wynikowej ścieżki
        usuń znalezioną krawędź z grafu
        wyznacz na nowo ścieżkę i rozpocznij proces wykrywania i eliminacji kolizji
        od początku
```

5.1.2. Pseudokod algorytmu

W tym podrozdziale przedstawiony zostanie pseudokod algorytmu planowania ruchu pojazdu.

5.1.3. Testy i dalszy rozwój algorytmu

5.2. Algorytm parkowania

5.2.1. Analiza problemu parkowania równoległego

Podczas próby zaparkowania równoległego pojazdu w wybranym miejscu parkingowym możliwe jest wystąpienie następujących sytuacji: - miejsce parkingowe jest za małe i zaparkowanie pojazdu w ogóle nie jest możliwe, - miejsce parkingowe jest dostatecznie duże i możliwe jest zaparkowanie w jednym manewrze, - miejsce parkingowe jest wystarczającej wielkości aby pojazd się w nim zmieścił, ale wymaga wykonania kilku ruchów w przód i w tył. Oczywiście przy definiowaniu powyższych przypadków nie są brane pod uwagę dodatkowe ograniczenia, które mogą wystąpić w otoczeniu miejsca parkingowego i np. uniemożliwić parkowanie prostopadłe w pojedynczym manewrze. Przy projektowaniu algorytmów parkowania przyjęto założenie, że obszar wokół miejsc parkingowych, po którym porusza się pojazd już podczas wykonywania manewru parkowania jest bezkolizyjny, a jeśli zdefiniowana przez użytkownika mapa nie zapewnia takowej bezkolizyjności zwracany jest komunikat, ze zaparkowanie w wybranym miejscu nie jest możliwe. Wyprowadzone zostaną teraz matematyczne warunki, jakie muszą wystąpić w każdej z wyżej opisanych sytuacji: - miejsce parkingowe jest zbyt małe - sytuacja ma miejsce, gdy długość lub szerokość miejsca parkingowego jest mniejsza niż długość lub szerokość wybranego pojazdu - miejsce parkingowe jest dostatecznie duże, by możliwe było zaparkowanie w pojedynczym manewrze – aby wyprowadzić warunki matematyczne należy najpierw przeanalizować ruch pojazdu podczas wykonywania manewru parkowania. W tym celu przedstawiono poniższy rysunek. //rysunek Na powyższym rysunku zaznaczone zostały okręgi, minimalny i maksymalny, jakie zarysowuje pojazd podczas parkowania. Do obliczenia minimalnej długości miejsca parkingowego potrzebnej do wykonania parkowania w pojedynczym manewrze przyjmijmy następujące oznaczenia: - Rmin – promień minimalnego okręgu wyznaczanego przez tylne koło znajdujące się po wewnętrznej stronie skrętu - Rmax – promień maksymalnego okręgu wyznaczanego przez przednie koło znajdujące się po zewnętrznej stronie skrętu - L - minimalna długość miejsca parkingowego pozwalająca na parkowanie w pojedynczym manewrze Minimalną długość miejsca parkingowego można obliczyć w następujący sposób: // obliczenia - miejsce parkingowe jest wystarczające, aby pojazd mógł w nim zaparkować wykonując kolejno ruchy w przód i w tył – jak nie trudno się domyślić, taka sytuacja ma miejsce, gdy długość miejsca parkingowego jest większa od długości pojazdu, ale jednocześnie mniejsza niż minimalna długość potrzebna do zaparkowania w pojedynczym manewrze.

5.2.2. Idea algorytmu

Parametrami wejściowymi dla algorytmu wyznaczania ścieżki parkowania są następujące dane: - mapa, - pojazd, - wybrane miejsce parkingowe (zawiera ono dozwolony typ parkowania, tzn. równoległe lub prostopadłe). Działanie samego algorytmu rozpoczyna się od sprawdzenia, która z opisanych w poprzednim podrozdziale sytuacji ma miejsce. Jeśli miejsce parkingowe jest zbyt małe to algorytm kończy działanie i zwraca informację, że zaparkowanie w danym miejscu nie było możliwe. Następnym krokiem jest sprawdzenie czy mamy do czynienia z parkowaniem równoległym, czy prostopadłym oraz czy naszym celem jest wyznaczenie ścieżki wjazdowej, czy wyjazdowej z miejsca parkingowego. Jednak zamiast rozważać osobno w jaki sposób wygenerować ścieżkę parkowania umożliwiającą wjazd, a w jaki sposób ścieżkę umożliwiającą wyjazd z miejsca parkingowego wystarczy rozważyć tylko sytuację wyjeżdżania z danego miejsca. Takie podejście jest wystarczające, ponieważ mając wyznaczoną ścieżkę, umożliwiającą wyjazd z miejsca parkingowego możliwe jest uzyskanie ścieżki wjazdowej poprzez odwrócenie kolejności elementów wchodzących w skład tej ścieżki. W przypadku parkowania prostopadłego wyznaczenie ścieżki parkowania jest zadaniem trywialnym i sprowadza się do ustawienia pojazdu idealnie na środku miejsca parkingowego, a następnie wycofywania pojazdu, aż do momentu, gdy cały znajdzie się na zewnątrz. Jeśli zadaniem algorytmu było wyznaczenie ścieżki wyjazdowej to jego działanie zostanie w tym momencie zakończone. W przypadku, gdy poszukiwana była ścieżka wjazdowa wystarczy odwrócić parametry wyznaczonej linii, po której ma poruszać się pojazd. W obu przypadkach wyznaczona ścieżka składa się z pojedynczej linii. Trudniejszym zadaniem jest wyznaczenie ścieżki parkowania równoległego. Na początek podobnie jak poprzednio należy ustawić pojazd na środku miejsca parkingowego. W następnym kroku pojazd jest wycofywany, aż do momentu, jego tylna krawędź pokryje się z tylną krawędzią miejsca parkingowego (tutaj można rozważać sytuację idealną gdy tylna krawędź pojazdu zrównuje się z tylną krawędzią miejsca lub dać jakąś tolerancję, aby do tej krawędzi nie dojeżdżał i zachowywał określony odstęp). Gdy pojazd znajduje się już w takim ustawieniu rozpoczyna się główna pętla algorytmu, w której pojazd porusza się po kolejnych fragmentach okręgów.

5.2.3. Pseudokod algorytmu

W tym podrozdziale przedstawiony zostanie pseudokod opisanego powyżej algorytmu parkowania oraz innych procedur, które są wykorzystywane w tym algorytmie.

// pseudokod

5.2.4. Testy i dalszy rozwój algorytmu

Testy algorytmu parkowania przebiegły pomyślnie i potwierdziły prawidłowe działanie algorytmu, zgodne z oczekiwaniami przy uprzednio przyjętych założeniach. Poniżej zaprezentowano przykładowe ścieżki parkowania, wygenerowane zarówno dla parkowania prostopadłego jak i równoległego: 1. Parkowanie prostopadłe - wyznaczona ścieżka parkowania, gdy otoczenie miejsca parkingowego jest bezkolizyjne // rysunek - sytuacja, w której algorytm zwraca informację, że zaparkowanie nie jest możliwe, choć w rzeczywistości taki manewr dałoby się wykonać (przeszkoda znajduje się zbyt blisko wyznaczonej ścieżki parkowania i powoduje kolizję z poruszającym się po tej ścieżce pojazdem). //rysunek 2. Parkowanie równoległe: - szerokie miejsce parkingowe – parkowanie w pojedynczym manewrze // rysunek - wąskie miejsce parkingowe – parkowanie w kilku manewrach //rysunek - ekstremalnie wąskie miejsce parkingowe – parkowanie w bardzo dużej liczbie manewrów, ale wciąż możliwe

Działanie algorytmu wyznaczania ścieżek parkingowych pokrywa wszystkie sytuacje jakie mogą wystąpić podczas wykonywania manewru parkowania, gdy otoczenie wokół miejsca parkingowego jest bezkolizyjne. Jednak w przypadku, gdy jakiś obiekt znajduje się zbyt blisko ścieżki parkowania i powoduje kolizję z poruszającym się po ścieżce pojazdem, to algorytm zwraca informację, że zaparkowanie w tym miejscu nie jest możliwe, choć w rzeczywistości, często dałoby się taką przeszkodę ominąć i zakończyć manewr parkowania pomyślnie. Dalszym krokiem w rozwoju algorytmu byłoby więc wykrywanie takich sytuacji i próba omijania przeszkód, aby możliwe było pomyślne dokończenie manewru.

6. Instrukcja użytkownika

7. Rozdział pokazowy

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumyeirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diamvoluptua.

At vero eos et accusam et justo duo dolores et ea rebum.

7.1. Przykładowa sekcja/podrozdział

Definicja 7.1 (Definicja). *Definicją* nazywamy wypowiedź o określonej budowie, w której informuje się o znaczeniu pewnego wyrażenia przez wskazanie innego wyrażenia należącego do danego języka i posiadającego to samo znaczenie.

7.1.1. Podsekcja

Poniżej podsekcji nie schodzimy.

Definicja 7.2 (Definicja). *Równaniem* nazywamy formę zdaniową postaci $t_1 = t_2$, gdzie t_1, t_2 są termami przynajmniej jeden z nich zawiera pewną zmienną.

Przykład 7.3. Przykładem równania jest

$$2 + 2 = 4. \tag{7.1}$$

Jeśli nie chcemy numerka, piszemy

$$2 + 2 = 4.$$

Równanie (7.2) jest fałszywe. Referencje (i kilka innych rzeczy) działają po dwukrotnym przekompilowaniu tex-a.

$$\int_0^1 x \, dx = \frac{3}{2}. \tag{7.2}$$

Twierdzenie 7.4 jest bardzo ciekawe.

Twierdzenie 7.4 (Twierdzenie Pitagorasa). Niech będzie dany trójkąt prostokątny o przyprostokątnych długośći a i b oraz przeciwprostokątnej długości c. Wtedy

$$a^2 + b^2 = c^2.$$

Dowód: Dowód został zaprezentowany w [1] oraz [2]. Czyli w sumie mogę napisać, że w [1, 2]. Albo że łatwo widać. □

Wniosek 7.5. Doszedłem do jakiegoś wniosku i daję temu wyraz.

Uwaga 7.6. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumyeirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diamvoluptua. At vero eos et accusam et justo duo dolores et ea rebum.

Lemat 7.7 (Lemacik). Ten lemat jest nie na temat.

Dowód: Dowód przez indukcję. □

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumyeirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diamvoluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumyeirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diamvoluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

7.2. Tabele i rysunki

Opcjonalny argument środowisk table i figure
h - bez przemieszczenia, dokładnie w miejscu użycia (uzyteczne w odniesieniu do niewielkich wstawek);
t - na górze strony;
b - na dole strony;
p - na stronie zawierającej wyłącznie wstawki;
! - ignorując większość parametrów kontrolujacych umieszczanie wstawek, przekroczenie wartosci, których może nie pozwolić na umieszczanie nastepnych wstawek na stronie.

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumyeirmod tempor

Rysunek 7.1: Obrazek zrobiony w LaTeXu

bla	blabla	blablabla
bla	blabal	blablabla
ble	bleble	blebleble

Tablica 7.1: Pełny opis znajdujący się pod tabelą

invidunt ut labore et dolore magna aliquyam erat, sed diamvoluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumyeirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diamvoluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Rysunek 7.2: Jakiś obrazek

Definicja 7.8. Niech $A \neq \emptyset$, $n \in \mathbb{N}$. Każde przekształcenie $f : A^n \to A$ nazywamy *n-arną operacją* lub *działaniem* określonym na A. 0-arne operacje to wyróżnione stałe.

Definicja 7.9 (Algebra). Parę uporządkowaną (A, F), gdzie $A \neq \emptyset$ jest zbiorem, a F jest rodziną operacji określonych na A, nazywamy *algebrą* (lub *F-algebrą*). Zbiór A nazywa się *zbiorem elementów*, *nośnikiem* lub *uniwersum* algebry (A, F), a F *zbiorem operacji elementarnych*.

Stwierdzenie 7.10. Stwierdzam więc ostatnio, że doszedłszy do granicy, pozostaje mi tylko przy tej granicy biwakować albo zawrócić, możliwie też szukać przejścia czy wyjścia na nowe obszary.

Bibliografia

[1] A. Aaaaa, *Tytuł*, Wydawnictwo, rok, strona-strona.

[2] J. Bobkowski, S. Dobkowski, *Blebleble*, Magazyn nr, rok, strony.

[3] C. Brink, *Power structures*, Algebra Universalis 30(2), 1993, 177-216.

[4] F. Burris, H. P. Sankappanavar, *A Course of Universal Algebra*, Springer-Verlag, New York, 1981.

Wykaz symboli i skrótów

nzw.	nadzwyczajny
$*$	operator gwiazdka
$\sim$	tylda

Spis rysunków

5.1 Przykład powiększania przeszkody 26
5.2 Graf Voronoi dla mapy z przeszkodami przed powiększeniem 27
5.3 Graf Voronoi dla mapy z przeszkodami po powiększeniu 27
5.4 Graf Voronoi z dodatkowymi krawędziami 28
5.5 Graf Voronoi z dodatkowymi wierzchołkami i krawędziami 28
5.6 Ścieżka wyznaczona w grafie pomiędzy położeniem początkowym i końcowym . . 29

7.1 Obrazek zrobiony w LaTeXu 37
7.2 Logo MiNI 37

Spis tabel

7.1 Opis skrócony 37

Spis załączników

1. Załącznik 1
2. Załącznik 2

Załącznik 1, załącznik 2