Planowanie ścieżek metrycznych i planowanie ruchu

* Zdefiniuj Cspace, relaksację ścieżki, stronniczość digitalizacji, obsesję na punkcie podrzędnych celów i warunek zakończenia.

* Reprezentuj środowisko wewnętrzne za pomocą uogólnionego wykresu Voronoi, regularnej siatki lub drzewa czwórkowego i utwórz wykres odpowiedni do planowania ścieżki.

* Zastosuj algorytm wyszukiwania A* do wykresu, aby znaleźć optymalną ścieżkę między dwoma lokalizacjami.

* Wyjaśnij różnice między zmianą ciągłą a zmianą sterowaną zdarzeniami.

* Wyjaśnij, w jaki sposób algorytm wyszukiwania D* realizuje ciągłe ponowne planowanie.

* Opisz problem poruszającego się po fortepianie i dlaczego różni się on od planowania trasy.

* Wyjaśnij podobieństwa i różnice między algorytmami szybkiego eksploracji drzewa losowego (RRT) a algorytmami typu A*.

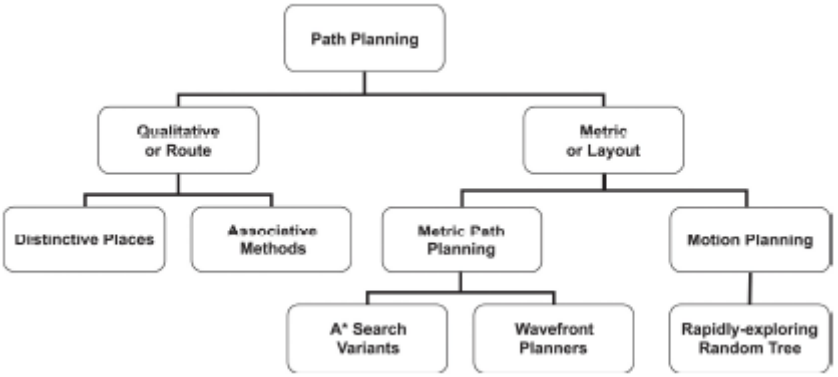
* Wymień kryteria oceny narzędzia do planowania ścieżki, a jeśli otrzymałeś opis narzędzia do planowania ścieżki lub ruchu, użyj kryteriów, aby ocenić użyteczność narzędzia do planowania.

Przegląd

Planowanie trasy jest próbą odpowiedzi na pytanie nawigacyjne: Jaka jest najlepsza droga? zakładając, że cel jest znany i istnieje mapa. W poprzedniej części przedstawiono nawigację topologiczną jako jedną z dwóch form nawigacji. Nawigacja topologiczna opierała się na pamięci przestrzennej zorganizowanej jako trasy między punktami orientacyjnymi, podczas gdy inna forma nawigacji, nawigacja metryczna, przewidywała, że układy środowiska będą rezydować w pamięci przestrzennej. W tym momencie czytelnik może chcieć bardziej szczegółowej odpowiedzi na pytanie Jaka jest różnica między nawigacją topologiczną a nawigacją metryczną / planowaniem ścieżki? Z praktycznego punktu widzenia wdrażania kolejne pytanie brzmi: Co jest powszechnie używane lub co działa wystarczająco dobrze? Bardziej subtelne pytanie brzmi: Ile planowania ścieżki potrzebujesz? Zagnieżdżony hierarchiczny kontroler podziału deliberacji opisany w Części 12 sugeruje, że planowanie jest rzadkie. Czy jednak przy zaawansowanych zasobach obliczeniowych istnieje jakaś wada ciągłego przemyślanego ponownego planowania?

KOMPONENTY PLANERA ŚCIEŻKI METRYCZNEJ

Postaramy się odpowiedzieć na wszystkie te pytania. Najpierw spróbuje rozróżnić metody topologiczne i metryczne, przedstawiając cztery różne sytuacje, w których nawigacja topologiczna jest niewystarczająca. Ponieważ planery ścieżki metrycznej i ruchu składają się z dwóch elementów – reprezentacji (struktury danych) i algorytmu – zostaną omówione osobno. Najpierw omówimy, w jaki sposób planiści ścieżek dzielą świat na strukturę lub mapę nadającą się do planowania ścieżek, zwaną przestrzenią konfiguracji. Intencją każdej reprezentacji jest reprezentowanie tylko istotnych cech lub odpowiedniej konfiguracji istotnych pod względem nawigacyjnym obiektów w interesującej nas przestrzeni; dlatego termin przestrzeń konfiguracyjna. Następnie zagłębimy się w popularne algorytmy planowania ścieżki, skupiając się na A* i popularnym wariancie wyszukiwania D*. Algorytmy wyszukiwania mogą ogólnie przeszukiwać dowolną reprezentację przestrzeni konfiguracyjnej, chociaż, podobnie jak w przypadku każdego algorytmu, niektóre metody działają lepiej w połączeniu z określonymi strukturami danych. Rysunek



przedstawia trzy kategorie planowania ścieżki metrycznej i ruchu. Algorytmy ścieżki lub trasy dzielą się na dwie szerokie kategorie: te, które traktują planowanie ścieżki jako problem przeszukiwania grafów przy użyciu wariantu algorytmu A*, oraz algorytmy propagacji czoła fali, które traktują planowanie ścieżki jako problem z kolorowaniem grafiki. Skupimy się na A* i jego popularnym wariancie D*. Nnastępnie skupimy się na planowaniu ruchu w celu manipulacji i innych sytuacjach wrażliwych na poza; zazwyczaj używają one wariantu algorytmu szybko ewoluującego drzewa losowego (RRT). Na koniec Część dotyczy realizacji ścieżki oraz kwestii przeplatania reakcji i planowania. Niezależnie od tego, jakiego algorytmu używamy, zawsze pojawia się kwestia, kiedy planować, a zwłaszcza jak przeplatać reakcję i planowanie. Czy planer powinien być uruchamiany ponownie po każdej aktualizacji czujnika, czy też plan powinien być modyfikowany tylko wtedy, gdy świat się zmieni lub wydajność znacznie się pogorszy?

Cztery sytuacje, w których nawigacja topologiczna nie wystarcza

Istnieją co najmniej cztery typowe sytuacje, w których nawigacja topologiczna nie jest odpowiednia. Jedna sytuacja ma miejsce, gdy przestrzeń między punktem początkowym a miejscem docelowym nie jest łatwo wyodrębniona na oznaczone lub percepcyjnie odrębne regiony. Na przykład bezzałogowy statek powietrzny na niebie może mieć bardzo mało percepcyjnych punktów orientacyjnych. Druga sytuacja ma miejsce, gdy wybór trasy wpływa na kontrolę lub koszty energii pojazdu. Na przykład szybko poruszający się robot może nie być w stanie wykonać ciasnego skrętu z dużą prędkością, ale mógłby wykonać ten skręt, gdyby jechał wolniej. Robot może być w stanie wykonać szerszy skręt bez poświęcania prędkości, a tym samym przebyć większą odległość, ale do miejsca docelowego dotrzeć później. Teoretycznie kompensacja zakrętów przy dużych prędkościach mogłaby zostać włączona do lokalnej strategii sterowania, ale wymagałoby to eksperymentowania i poznania właściwej dynamiki pojazdu, ryzykując rozbicie platformy. W idealnym przypadku robot mógłby generować bezpieczny wybór tras bezpośrednio z mapy a priori. Innym przykładem jest robot naziemny, który może preferować trasy o łagodnych zboczach lub niewymagające energochłonnego wspinania się, czy też pojazdy latające, które wybierają loty minimalizujące wiatry boczne. Trzeci przypadek może wystąpić, gdy celem ścieżki jest umożliwienie pokrycia obszaru przez czujnik. Na przykład bezzałogowy pojazd morski może wymagać przeprowadzenia skanowania obszaru oceanu przez kosiarkę w celu znalezienia zatopionego statku lub wraku samolotu. Innym przykładem może być UAV mierzący teren lub plac budowy. W takich przypadkach trasa zostanie określona jako sekwencja lokalizacji lub punktów trasy w ramce współrzędnych. Czwarta sytuacja może mieć miejsce, gdy ważna jest pozycja robota, czy to podczas dotarcia do celu, czy też w trakcie pokonywania obszaru. Rozważ zsunięcie pianina w dół schodów, gdzie pianino musi być odwrócone na jedną stronę i obrócone, aby przejść przez ciasne przestrzenie; trafnie nazywa się to problemem porucznika fortepianu. Ponieważ prawdziwe roboty nie są holonomiczne, problem osób poruszających się po pianinach szybko pojawia się w ciasnych lub

zagraconych przestrzeniach. Problem z pianinem pojawia się również przy przenoszeniu ramienia robota z jednego miejsca do drugiego pomiędzy bałaganem. Ruch ramion lub nóg robota jest ogólnie nazywany planowaniem ruchu, aby odróżnić go od ruchów makro całej platformy. Poza może być również ważna w zastosowaniach kryjących. W badaniach lotniczych ogólna jakość badania poprawia się, jeśli zdjęcia obszaru są robione z tej samej wysokości iz kamerą prostopadłą do ziemi. Jednak podczas skanowania kosiarki bezzałogowy statek powietrzny ze stałym skrzydłem musi skręcić. Oznacza to, że kamera nie jest już skierowana prosto w dół i wysokość może się nieznacznie zmienić. Dlatego też bezzałogowe statki powietrzne z nieruchomymi skrzydłami zawierały przeregulowanie, aby zapewnić wystarczającą odległość, aby UAV mógł skręcić i powrócić do prawidłowej pozycji przed powrotem do pomiarów.

PUNKTY TRASY

Te cztery sytuacje prowadzą do innego smaku planowania ścieżki, zwanego planowaniem ścieżki metrycznej i planowaniem ruchu. W celu zapewnienia przeglądu, planowanie ścieżki metrycznej i planowanie ruchu będzie określane ogólnie jako metody zorientowane na metrykę lub układ. W metodach metrycznych robot posiada a priori mapę otoczenia i generuje plan odpowiednich ruchów, aby osiągnąć cel lub stopień pokrycia. Metody metryczne generalnie faworyzują techniki, które wytwarzają optymalną ścieżkę lub sekwencję ruchów, zgodnie z pewną miarą najlepszej, podczas gdy metody jakościowe w rozdziale 13 zadowalały się stworzeniem trasy z możliwymi do zidentyfikowania punktami orientacyjnymi lub bramami. Kolejną różnicą jest to, że ścieżki metryczne zwykle rozkładają ścieżkę na podcele składające się z punktów orientacyjnych. Te punkty orientacyjne są najczęściej stałymi lokalizacjami lub współrzędnymi (x,y) lub lokalizacjami i pozycjami. Lokalizacje te mogą mieć znaczenie matematyczne, ale mogą nie mieć sensownego nawiązania do obiektów lub punktów orientacyjnych na świecie, np. „idź 15 metrów, skręć – 90°” wymaga lokalizacji i mapowania w porównaniu do „przejdź około 15 metrów do punktu orientacyjnego” L, skręć w lewo”, co wymaga wykrycia punktów orientacyjnych. Terminy „optymalny” i „najlepszy” mają poważne konsekwencje dla robotyki. Powiedzenie, że ścieżka jest optymalna, implikuje porównanie. Jak zobaczymy, niektóre metody metryczne są w stanie wytworzyć optymalną ścieżkę, ponieważ uwzględniają wszystkie możliwe ścieżki między punktami. Co zaskakujące, optymalna ścieżka może nie wydawać się optymalna dla ludzkiego oka; na przykład matematycznie optymalna ścieżka świata podzielonego na kafelki lub siatki może być bardziej poszarpana i nieregularna niż prosta. Możliwość tworzenia i porównywania wszystkich możliwych ścieżek zakłada również, że planowanie ma dostęp do istniejącej (lub a priori) mapy świata. Równie ważne, planowanie ścieżki z użyciem mapy a priori zakłada, że mapa jest dokładna i aktualna. Jako takie, metody metryczne są kompatybilne z deliberacją, podczas gdy metody jakościowe działają dobrze z bardziej reaktywnymi systemami. Jako funkcja deliberatywna, metody metryczne są często nękane wyzwaniami w zakresie reprezentacji świata, radzenia sobie z dynamicznymi zmianami i niespodziankami oraz złożonością obliczeń.

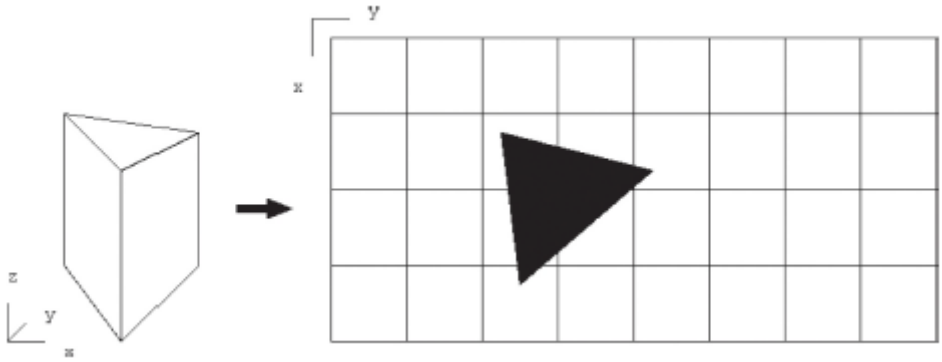
PRZESTRZEŃ KONFIGURACJI (CSPACE)

Fizyczne roboty kosmiczne i przeszkody, w których istnieją, można traktować jako przestrzeń światową. Przestrzeń konfiguracyjna, w skrócie Cspace, to struktura danych, która pozwala robotowi określić położenie (lokalizację i orientację) siebie oraz wszelkich innych obiektów i robota.

STOPNIE SWOBODY

Dobra reprezentacja w Cspace zmniejsza liczbę wymiarów, z którymi musi się zmagać planista. Weź pod uwagę, że potrzeba sześciu wymiarów (zwanych również stopniami swobody lub DOF), aby dokładnie określić, gdzie znajduje się obiekt. Osoba może określić położenie obiektu jako zbiór współrzędnych (x, y, z) w pewnym układzie odniesienia. Ale przedmiot jest trójwymiarowy; ma przód i

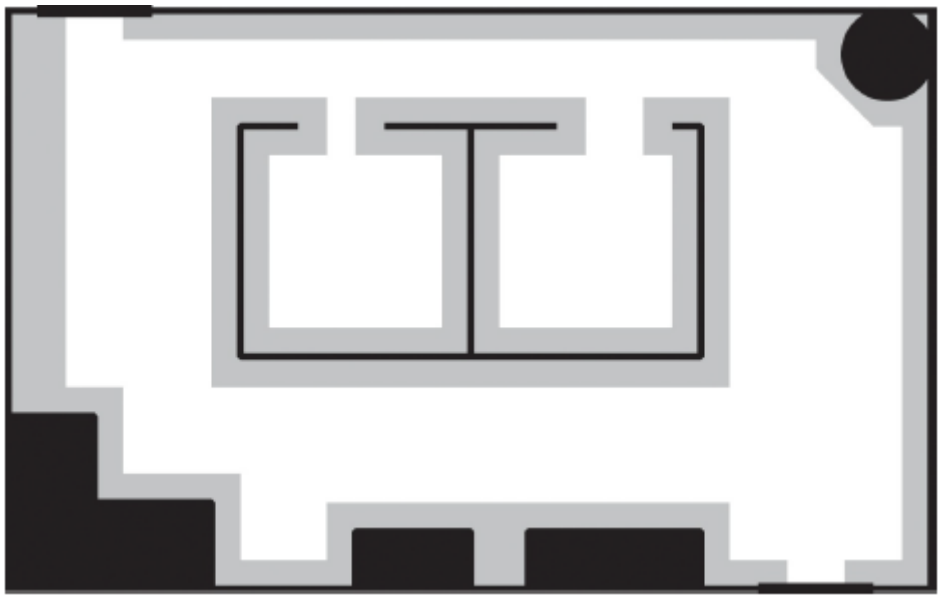
tył, górę i dół. Potrzebne są jeszcze trzy stopnie, aby określić, gdzie zwrócony jest przód krzesła, niezależnie od tego, czy jest przechylony, czy nie, czy nawet do góry nogami. Są to kąty Eulera (wymawiane „ojlera”), φ, θ, γ, znane również jako pochylenie, odchylenie i przechylenie. Sześć stopni swobody to więcej niż potrzeba w przypadku mobilnego robota naziemnego w większości przypadków do planowania ścieżki. Współrzędną z (wysokość) można wyeliminować, jeśli robot i wszystkie obiekty leżą na podłodze. Jednak współrzędna z będzie interesująca, jeśli robot jest pojazdem powietrznym lub podwodnym. Podobnie kąty Eulera mogą być niepotrzebne. Kogo obchodzi, w którą stronę zwrócony jest robot, jeśli wszystko, co chce zrobić, to zaplanować ścieżkę wokół przeszkody? Ale skok planetarnego łazika lub zbocze nadchodzącego wzgórza może mieć kluczowe znaczenie dla misji na skalistym terenie. Rysunek przedstawia transformację obiektu do przestrzeni Cspace.



Ogólnie rzecz biorąc, metryczne algorytmy planowania ścieżki dla robotów mobilnych zakładały tylko dwa DOF, translację i rotację. Świat jest opisany w dwóch wymiarach (x, y), a robot może poruszać się tylko do przodu lub do tyłu oraz skręcać w płaszczyźnie (x, y). Liczba różnych reprezentacji Cspace jest zbyt duża, aby uwzględnić tutaj więcej niż próbkowanie zgrubne. Poniższe podrozdziały zawierają szczegółowe informacje na temat trzech ważnych reprezentacji Cspace: map łąk, GVG i siatek. Jednak warto zacząć od przeglądu trzech reprezentacji. Jedna grupa metod dzieli świat na nieregularne wielokątne regiony pustych przestrzeni, takie jak mapy łąk i uogólnione grafy Voronoi (GVG). Mapy łąk nie były często używane w ciągu ostatnich kilku lat, ale są intuicyjne i ilustrują najlepsze praktyki uprawy przeszkód i relaksacji ścieżki oraz sposoby uzyskania ścieżki poprzez przeszukiwanie geometrii. Mapy łąkowe ilustrują również główne problemy związane z metodami zorientowanymi na region. Jednym z nich jest to, że wielokąty generowane przez algorytm niekoniecznie są unikalne; dlatego inny algorytm lub punkt początkowy wygeneruje inną mapę. Innym problemem jest to, że nie gwarantuje się, że granice będą łatwe do niezawodnego postrzegania przez robota, a tym samym wiedząc, gdzie się znajduje. Jednak GVG i inne metody zorientowane na region przeżywają odrodzenie, ponieważ wykrywanie zasięgu i gęste chmury punktów sprawiają, że ekstrakcja objętości jest bardziej niezawodna i spójna. Zamiast próbować znaleźć wielokąty otwartej przestrzeni, alternatywą jest reprezentacja brutalna, w której przestrzeń jest podzielona w pewien regularny, powtarzalny sposób, a następnie każdy podział jest oznaczony jako pusty lub zajęty. Zasadniczo te metody nakładają się na siatkę na całym świecie. Regularna siatka zapewnia proste partycjonowanie. Niestety, im mniejszy rozmiar elementu siatki, tym wyższa rozdzielczość przestrzeni konfiguracyjnej, ale im więcej elementów należy przeszukać, zwiększając czas obliczeniowy i przestrzeń potrzebną do planowania. W ten sposób pojawiły się rekurencyjne metody partycjonowania siatki, w szczególności drzewa czwórkowe i ósemkowe.

Meadow map

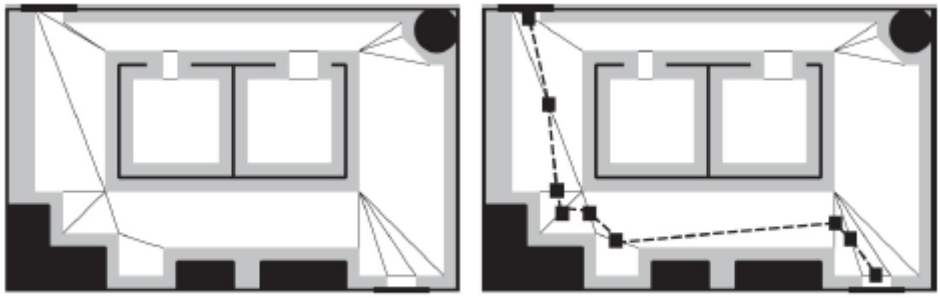
Wiele wczesnych algorytmów planowania ścieżki opracowanych dla robotów mobilnych zakładało, że robot będzie z wyprzedzeniem dysponował bardzo dokładną mapą świata. Ta mapa mogłaby zostać w jakiś sposób zdigitalizowana, a następnie robot mógłby zastosować różne algorytmy, aby przekonwertować mapę na odpowiednią reprezentację Cspace. Przykładem reprezentacji Cspace, która może być użyta z mapą a priori, jest meadow mapi lub hybrydowy model wolnej przestrzeni grafu wierzchołków.



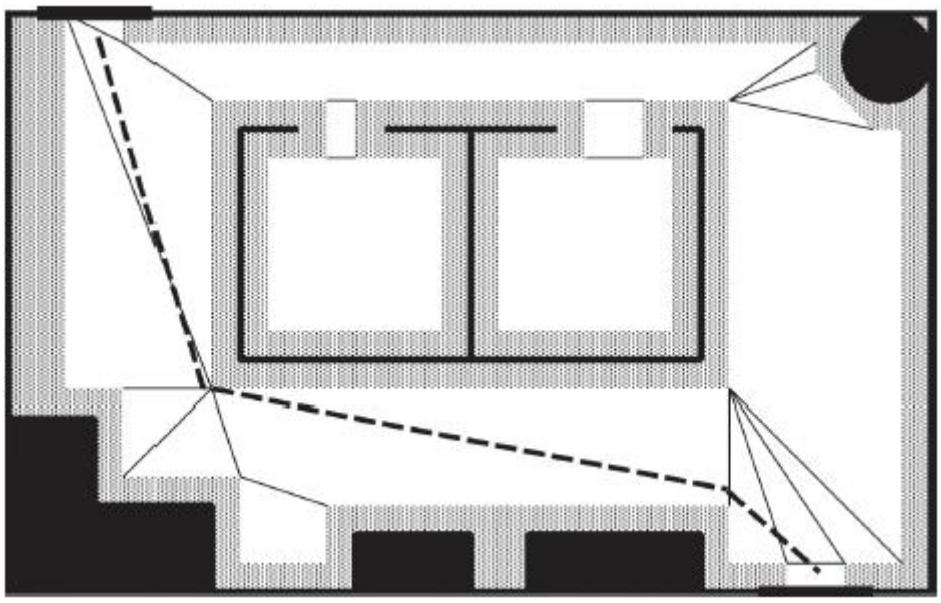
Meadow map przekształcają wolną przestrzeń w wypukłe wielokąty. Wielokąty wypukłe mają ważną właściwość: jeśli robot wystartuje na obwodzie i przejdzie w linii prostej do dowolnego innego punktu na obwodzie, nie wyjdzie z wielokąta. Wielokąt reprezentuje bezpieczny obszar, przez który robot może się poruszać. Problem planowania ścieżki staje się kwestią wybrania najlepszej serii wielokątów do przejścia. Meadow map nie są tak powszechne w robotyce, ale służą do zilustrowania zasad wpływania na przestrzeń konfiguracyjną, a następnie planowania przez nią ścieżki. Kroki programowania są proste. Najpierw planista zaczyna od metrycznego rozplanowania przestrzeni świata. Większość planistów zwiększy rozmiar każdego obiektu, dodając rozmiar robota do granicy obiektu; pozwala to planistom traktować robota tak, jakby był punktem, a nie obiektem 2D. Zauważ, że planiści ścieżki domyślnie zakładają pojazdy holonomiczne od pierwszego kroku. Następnym krokiem w algorytmie planowania ścieżki jest skonstruowanie wielokątów wypukłych poprzez uwzględnienie odcinków linii pomiędzy parami interesujących obiektów. W przypadku map wnętrz są to zazwyczaj narożniki, przejścia, granice obiektów i tak dalej. Algorytm może następnie określić, która kombinacja linii dzielą wolną przestrzeń na wypukłe wielokąty. Mapa łąk jest już technicznie kompletna, ale nie ma formatu umożliwiającego planowanie ścieżek. Każdy wypukły wielokąt reprezentuje bezpieczne przejście dla robota. Ale jest jeszcze trochę pracy do zrobienia. Niektóre z odcinków linii tworzących obwód nie są połączone z innym wielokątem (tj. są częścią ściany), więc powinny być niedostępne dla algorytmu planowania. Ponadto, jak widać na powyższym rysunku, niektóre odcinki linii są dość długie. Miałoby to wpływ na całkowitą długość ścieżki, na której robot przecina wielokąt. Projektantowi trudno jest zdyskretyzować ciągły odcinek linii. Tak więc problemem staje się określenie punktów kandydujących na wielokącie. Jednym z rozwiązań jest znalezienie środka każdego odcinka linii, który graniczy z innym wielokątem. Zauważ, że każdy z tych punktów środkowych staje się węzłem, a jeśli krawędzie zostaną narysowane między nimi, wyłania się graf nieskierowany. Algorytm planowania ścieżki określi najlepszą ścieżkę do obrania.

ŚCIEŻKA RELAKSACJI

Jedna wada mapy łąki, a właściwie każdej reprezentacji Cspace, jest widoczna na rysunku.



Każda wybrana ścieżka będzie nieco postrzępiona. Każdy z punktów przegięcia jest zasadniczo punktem orientacyjnym. Jednym z wyników procesu partycjonowania jest to, że wolna przestrzeń jest dzielona w sposób, który ma sens geometryczny, ale niekoniecznie działa to dla robota, który musi podróżować przez tę przestrzeń. Po co iść do połowy korytarza, a potem skręcić w kąt? Podział może być matematycznie optymalny na papierze, ale w praktyce wydaje się to wręcz głupie. Chuck Thorpe, pracując na Carnegie Mellon University, opracował rozwiązanie dla ścieżek generowanych z dowolnego rodzaju dyskretyzacji wolnej przestrzeni.



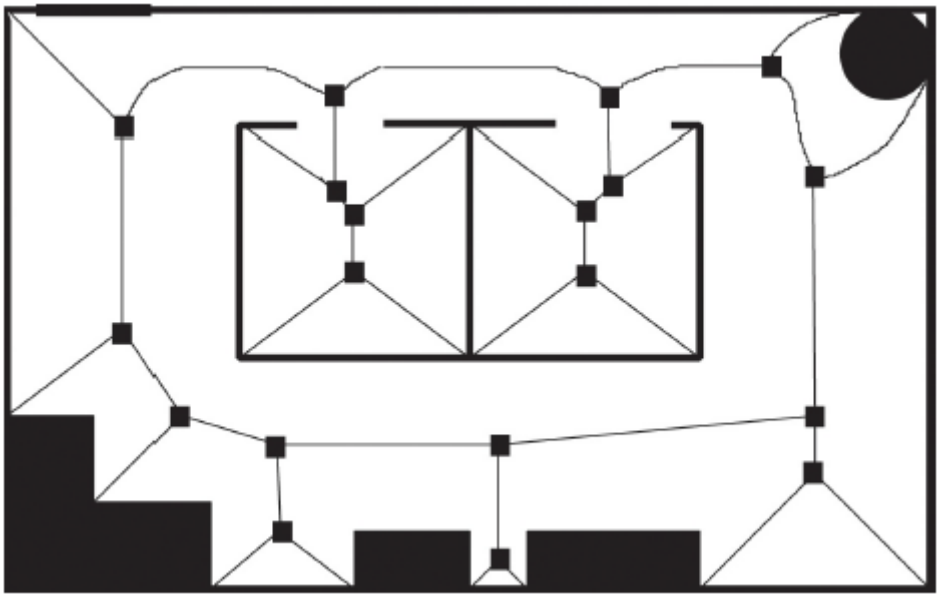
Wyobraź sobie, że ścieżka jest ciągiem. Teraz wyobraź sobie, że ciągniesz za oba końce, aby napiąć sznurek (techniczna nazwa to relaksacja ścieżki). Ten algorytm napinający łańcuch usunie większość załamań ze ścieżki bez naruszania właściwości strefy bezpiecznej wielokątów wypukłych. Mapy łąkowe mają trzy problemy, które ograniczają ich użyteczność. Po pierwsze, technika generowania wielokątów jest złożona obliczeniowo. Ale co ważniejsze, wykorzystuje artefakty mapy do określenia granic wielokątów, a nie rzeczy, które można wyczuć. Jeśli robot nie ma dokładnej lokalizacji, skąd wie, że znajduje się w połowie długiej, pozbawionej cech charakterystycznych hali i powinien skręcić o 30°? Trzecią poważną wadą jest to, że nie jest jasne, jak zaktualizować lub naprawić diagramy, ponieważ robot wykrywa rozbieżności między mapą a priori a światem rzeczywistym.

UOGÓLNIONY WYKRES WORONOJA (GVG)

Uogólniony graf Woronoja lub GVG to popularny mechanizm reprezentujący Cspace i generujący graf. W przeciwieństwie do mapy łąkowej, GVG można skonstruować, gdy robot wchodzi do nowego środowiska, tworząc w ten sposób mapę topologiczną, jak pokazuje Howie Choset z zarnegie Mellon University.

KRAWĘDŹ WORONOJA

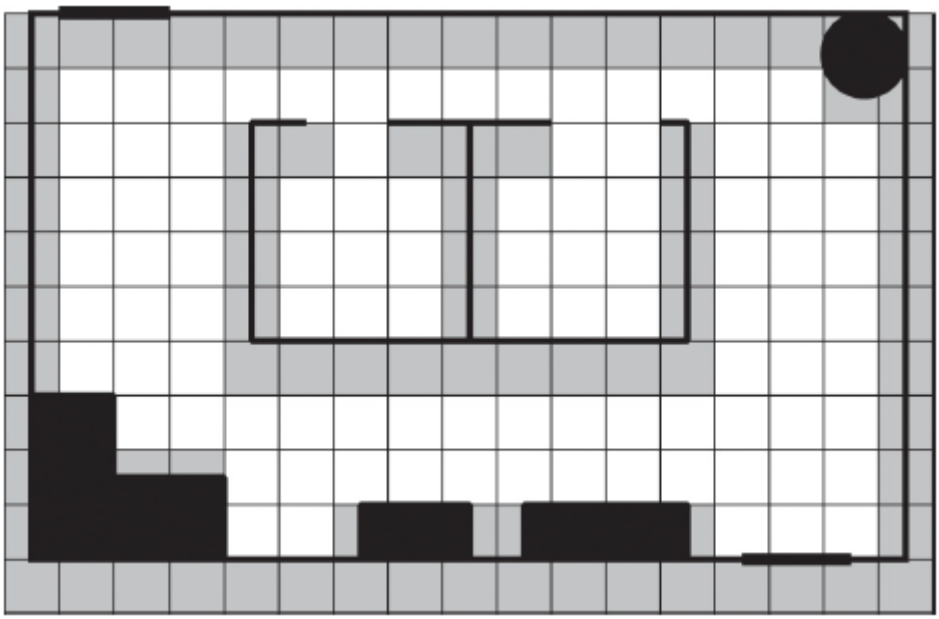
Podstawową ideą GVG jest wygenerowanie linii, zwanej krawędzią Woronoja, w równej odległości od wszystkich punktów. Jak widać na rysunku , linia ta biegnie środkiem korytarzy i otworów.



Punkt, w którym spotyka się wiele krawędzi Voronoi, jest znany jako wierzchołek Woronoja Zauważ, że wierzchołki często odpowiadają fizycznym konfiguracjom, które można wykryć w środowisku. Ułatwia to robotowi podążanie ścieżką wygenerowaną przez GVG, ponieważ istnieje niejawna strategia lokalnego sterowania pozwalająca na zachowanie równej odległości od wszystkich przeszkód. Jeśli robot podąża za krawędzią Voronoi, nie zderzy się z żadną wymodelowaną przeszkodą, ponieważ pozostaje „pośrodku”. Eliminuje to konieczność powiększania granic przeszkód, jak w przypadku mapy łąkowej. Krawędzie służą jako autostrady lub główne arterie. Należy również zauważyć, że zakrzywione krawędzie w GVG nie mają znaczenia dla teorii grafów ani algorytmów grafów. Tylko długość, a nie fizyczna rzeczywistość krawędzi ma znaczenie.

ZWYKŁA SIATKA

Inną metodą podziału przestrzeni świata jest regularna siatka. Metoda siatki regularnej nakłada siatkę kartezjańską 2D na przestrzeń świata, jak pokazano.



Jeśli w obszarze objętym elementem siatki znajduje się jakikolwiek obiekt, ten element jest oznaczony jako „zajęty”. Dlatego regularne siatki są często określane jako siatki zajętości.

4-POŁĄCZENI SĄSIEDZI

8-POŁĄCZENI SĄSIEDZI

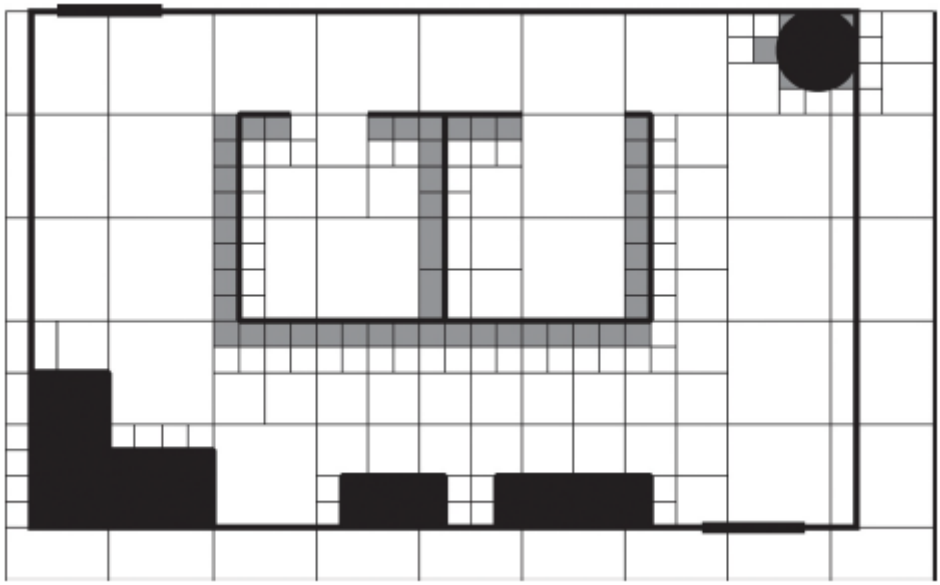
Regularne siatki są łatwe do nałożenia. Środek każdego elementu w siatce może stać się węzłem, przekształcając siatkę w silnie połączony wykres. Siatki są uważane za połączone 4 lub 8, w zależności od tego, czy pozwalają na rysowanie łuku po przekątnej między węzłami.

NASTAWIENIE CYFRYZACJI

Niestety zwykłe siatki nie są pozbawione problemów. Po pierwsze, wprowadzają stronniczość digitalizacji, co oznacza, że jeśli obiekt wpadnie w nawet najmniejszą część elementu siatki, cały element jest oznaczony jako „zajęty”. Prowadzi to do marnowania miejsca i bardzo postrzępionych obiektów. Aby zredukować marnowaną przestrzeń, regularne siatki do pomieszczeń wewnętrznych są często drobnoziarniste, rzędu 4 do 6 cali kwadratowych. Ta drobna ziarnistość oznacza wysoki koszt pamięci masowej i dużą liczbę węzłów, które należy wziąć pod uwagę algorytmowi planowania ścieżki.

DRZEWO CZWÓRKOWE

Czwórki są wariantem zwykłych siatek, które starają się uniknąć marnowania przestrzeni



Drzewo czwórkowe to siatka rekurencyjna. Reprezentacja zaczyna się od elementów siatki reprezentujących duży obszar, być może 64 cale kwadratowe (8 na 8 cali). Jeśli obiekt znajduje się w części siatki, ale nie w całości, algorytm Cspace dzieli element na cztery (stąd nazwa „czwórka") mniejsze siatki, każda o powierzchni 16 cali kwadratowych. Jeśli obiekt nie wypełnia określonego podelementu, algorytm dokonuje kolejnego rekurencyjnego podziału tego elementu na cztery kolejne podelementy, reprezentujące obszar o powierzchni 4 cali kwadratowych. Trójwymiarowe drzewo czworokątne nazywa się octree.

Planowanie ścieżki metrycznej

Metryczne metody planowania miejsc docelowych lub tras opierają się na mapie a priori. Metody grafowe przekształcają mapę a priori w podobną do grafu przestrzeń C, a następnie używają algorytmu przeszukiwania grafów, aby znaleźć najkrótszą ścieżkę na grafie między węzłem początkowym a docelowym. A* to klasyczny algorytm wyszukiwania grafów AI, z wariantem D* dostosowanym do robotów. Alternatywą jest wykorzystanie algorytmów kolorowania grafów z grafiki.

A* i planiści w oparciu o wykresy

WĘZEŁ POCZĄTKOWY

WĘZEŁ CEL

Jak widać wcześniej, większość reprezentacji Cspace można przekonwertować na wykresy. Oznacza to, że ścieżka między węzłem początkowym a węzłem docelowym może być obliczona za pomocą algorytmów przeszukiwania grafów. Algorytmy wyszukiwania grafów pojawiają się w sieciach i problemach z routingiem, więc jest to klasa algorytmów dobrze rozumiana przez informatyków. Jednak wiele z tych algorytmów wymaga, aby program odwiedził każdy węzeł na grafie w celu określenia najkrótszej ścieżki między węzłem początkowym a docelowym. Odwiedzenie każdego węzła może być wykonalne obliczeniowo w przypadku słabo połączonego wykresu, takiego jak wyprowadzony z diagramu Voronoi, ale jest to obliczeniowo kosztowne w przypadku silnie połączonego wykresu, takiego jak wygenerowany ze zwykłej siatki. W związku z tym istnieje duże zainteresowanie stosowaniem planerów ścieżek, które wykonują wyszukiwanie typu „rozgałęzione i powiązane"; to znaczy planiści, którzy odcinają ścieżki, które nie są optymalne. Oczywiście sztuka polega na tym, aby wiedzieć, kiedy przycinać!

A* ALGORYTM WYSZUKIWANIA

Algorytm wyszukiwania A* jest klasyczną metodą obliczania optymalnych ścieżek dla robotów holonomicznych. Wywodzi się z algorytmu wyszukiwania A. Wyszukiwanie zostanie najpierw przedstawione na przykładzie, a następnie zostanie wprowadzone A* (jak to się powszechnie nazywa) na podstawie tego przykładu. Oba zakładają mapę metryczną, w której położenie każdego węzła jest znane we współrzędnych bezwzględnych, a krawędzie wykresu wskazują, czy możliwe jest przechodzenie między tymi węzłami. Algorytm wyszukiwania A tworzy optymalną ścieżkę, zaczynając od węzła początkowego, a następnie przechodząc przez wykres do węzła docelowego. Generuje optymalną ścieżkę przyrostowo; ateach update, bierze pod uwagę węzły, które można dodać do ścieżki i wybiera najlepszy. Wybiera „prawy" węzeł do dodania do ścieżki za każdym razem, gdy rozszerza ścieżkę („prawy węzeł" jest bardziej formalnie znany jako prawdopodobny ruch). Sercem metody jest wzór (lub funkcja oceny) do pomiaru wiarygodności węzła:

f(n) = g(n) + h(n)

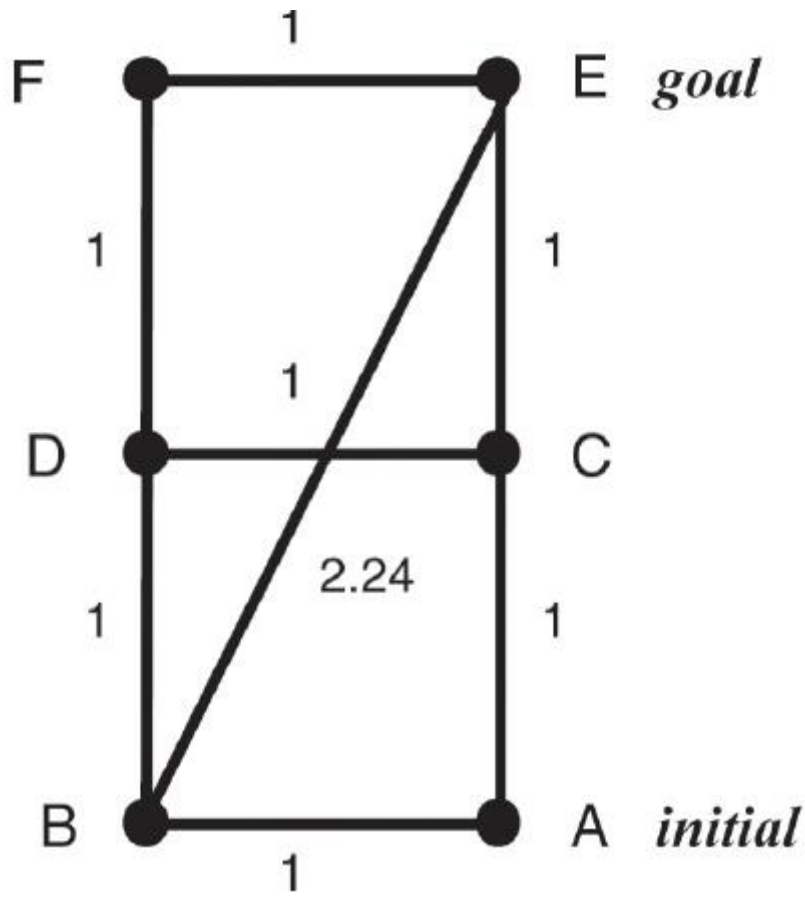
gdzie:

* f(n) mierzy, jak dobry jest ruch do węzła n

* g(n) mierzy koszt dotarcia do węzła n z węzła początkowego. Gdy A rozszerza się od początkowego węzła na zewnątrz, jest to odległość dotychczas wygenerowanej ścieżki plus odległość od krawędzi do węzła n

* h(n) reprezentuje najniższy koszt dojścia od n do celu

Zastanów się, jak formuła jest używana w poniższym przykładzie. Załóżmy, że reprezentacja Cspace dała wykres na rysunku



Algorytm wyszukiwania A zaczyna się w węźle A i tworzy strukturę podobną do drzewa decyzyjnego w celu określenia możliwych węzłów, które może rozważyć dodanie do swojej ścieżki. Do wyboru są tylko dwa węzły: B i C. Aby określić, który węzeł jest właściwym do dodania, algorytm wyszukiwania ocenia prawdopodobieństwo dodania B lub C, patrząc na krawędzie. Prawdopodobieństwo dodania B jako następnego ruchu jest następujące:

f(B) = g(B) + h(B) = 1 + 2.24 = 3.24

gdzie g(B) to koszt przejścia z A do B, a h(B) to koszt z B do celu E. Prawdopodobieństwo dodania C to:
f(C) = g(C) + h(C) = 1 +1 = 2.0

gdzie g(C) to koszt przejścia z A do C, a h(C) to koszt przejścia z C do E. Ponieważ f(C) > f(B), ścieżka powinna przebiegać z A do C.

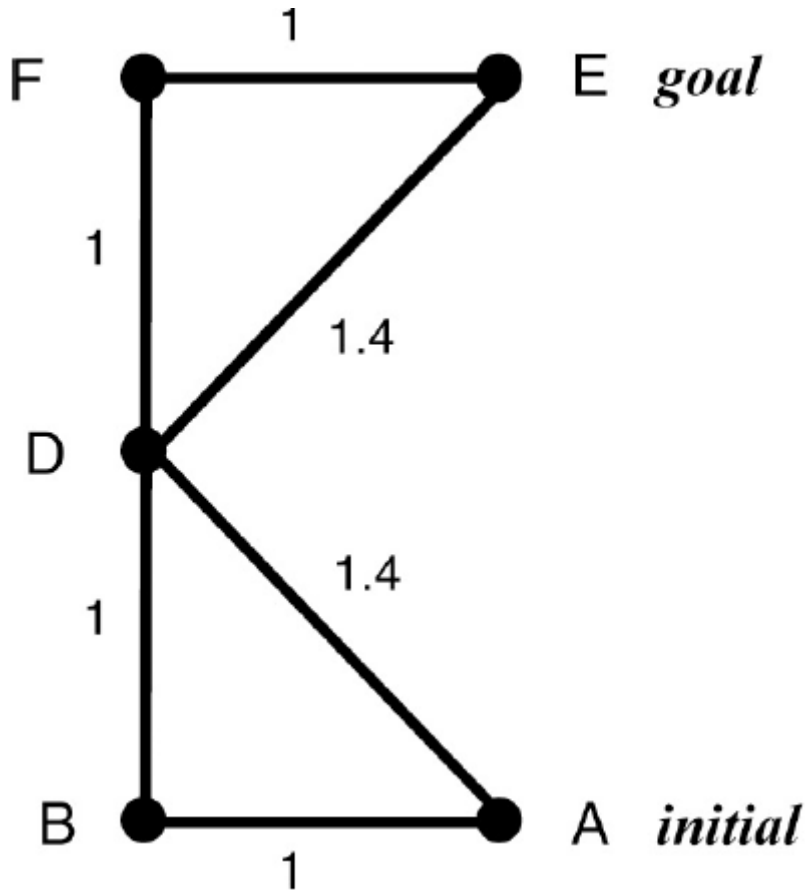
Ale to zakłada, że h(n) jest znane w każdym węźle. Oznacza to, że algorytm musi wykonać rekurencję, aby znaleźć poprawną wartość h(n). Oznacza to, że algorytm musi odwiedzić wszystkie węzły. Wyszukiwanie gwarantuje uzyskanie optymalnej ścieżki, ponieważ porównuje ze sobą wszystkie możliwe ścieżki. Wyszukiwanie* ma interesujące podejście do zmniejszania liczby ścieżek, które muszą zostać wygenerowane i porównane: algorytm porównuje możliwe ścieżki z najlepszą możliwą ścieżką, nawet jeśli w rzeczywistym świecie nie ma takiej ścieżki. Algorytm szacuje h zamiast sprawdzać, czy rzeczywiście istnieje segment ścieżki, który może dotrzeć do celu w tej odległości. Oszacowanie można następnie wykorzystać do określenia, które węzły są najbardziej obiecujące i kiedy pewne ścieżki nie mają szans na osiągnięcie celu krótszą ścieżką niż inne kandydatki, a zatem należy je usunąć z wyszukiwania. Pod A* funkcja oceny staje się:

f*(n) = g*(n) + h*(n)

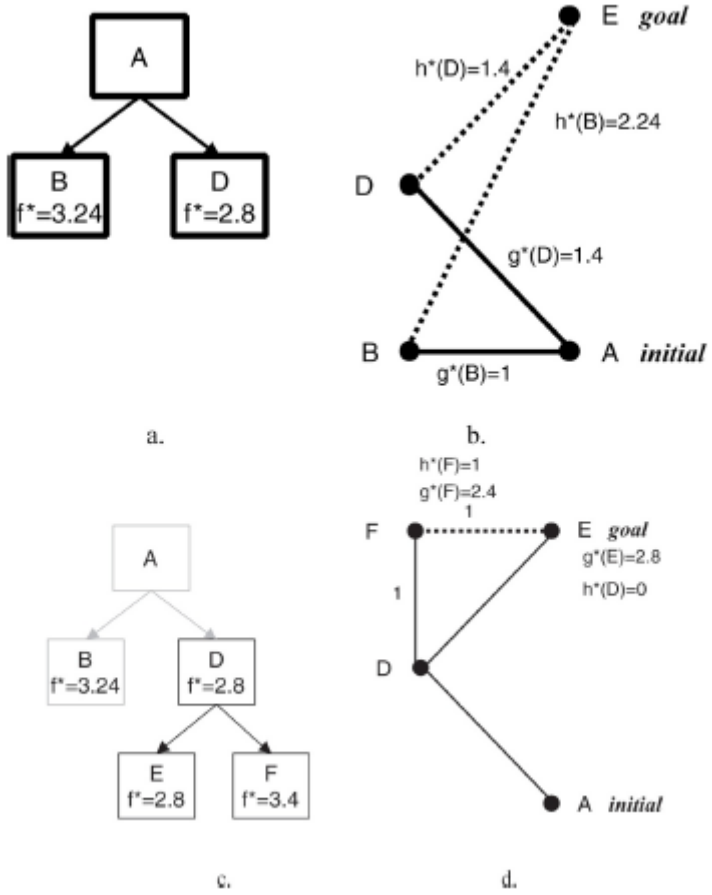
WARUNEK DOPUSZCZALNOŚCI

FUNKCJA HEURYSTYCZNA

gdzie * oznacza, że funkcje są oszacowaniami wartości, które zostałyby dołączone do oceny wyszukiwania A. W planowaniu ścieżki g*(n) jest tym samym co g(n): koszt przejścia z początkowego węzła do n, który jest znany poprzez przyrostowe budowanie ścieżki. h*(n) to prawdziwa różnica. Jaki jest więc sposób oszacowania kosztu przejścia od n do celu? Co więcej, jak możemy być pewni, że oszacowanie będzie wystarczająco dokładne, aby ostatecznie nie wybrać ścieżki, która nie jest naprawdę optymalna? Można to zrobić upewniając się, że h*(n) nigdy nie będzie mniejsze niż h(n). Ograniczenie, że h*(n) ≤ h(n) nazywa się warunkiem dopuszczalności. Ponieważ h*(n) jest wartością szacunkową, nazywa się ją również funkcją heurystyczną, ponieważ wykorzystuje praktyczną regułę, aby zdecydować, który węzeł jest najlepszy do rozważenia. Na szczęście istnieje naturalna funkcja heurystyczna służąca do szacowania kosztu od n do celu: odległość euklidesowa (linia prosta). Przypomnij sobie, że lokalizacje każdego węzła są znane niezależnie od krawędzi. Dlatego łatwo jest obliczyć odległość w linii prostej między dwoma węzłami. Odległość w linii prostej jest zawsze najkrótszą drogą między dwoma punktami, z wyjątkiem krzywizny ziemi i tak dalej. Ponieważ rzeczywista droga nigdy nie może być krótsza, warunek dopuszczalności jest spełniony. Aby zobaczyć, jak A* używa tego do faktycznego wyeliminowania węzłów wizytujących, rozważ rysunek 14.10. Podobnie jak w przypadku wyszukiwania A, pierwszym krokiem w A* jest rozważenie wyborów z początkowego węzła.



Do wyboru są B i D, które można traktować jako drzewo poszukiwań, jak pokazano na rysunku a lub jako podzbiór oryginalnego wykresu (patrz rysunek b).



Niezależnie od tego, jak jest wizualizowany, każdy węzeł jest oceniany w celu określenia, czy jest to najbardziej prawdopodobny ruch. Powyższy rysunek pokazuje, co algorytm „widzi" w tym momencie wykonywania. Wybory do oceny to:

f*(B) = g*(B) + h*(B) = 1 + 2.24 = 3.24

f*(D) = g*(D) + h*(D) =1.4 + 1.4 = 2.8

Ścieżka biegnąca od A − D−? − E może być krótsza niż ścieżka biegnąca od A − B−? − E. Czyli D jest najbardziej prawdopodobnym węzłem. Zauważ, że A* nie może wyeliminować ścieżki przechodzącej przez B, ponieważ algorytm nie może „zobaczyć" ścieżki, która faktycznie biegnie z D do E i określić, czy rzeczywiście jest ona możliwie najkrótsza. W kroku 2, A* powtarza się (powtarza ocenę) od D, ponieważ D jest najbardziej prawdopodobny, jak pokazano na rysunku powyżej. Dwie opcje z D to E i F, które są oceniane dalej to

f*(E) = g*(E) + h*(E) = 2.8 + 0 = 2.8

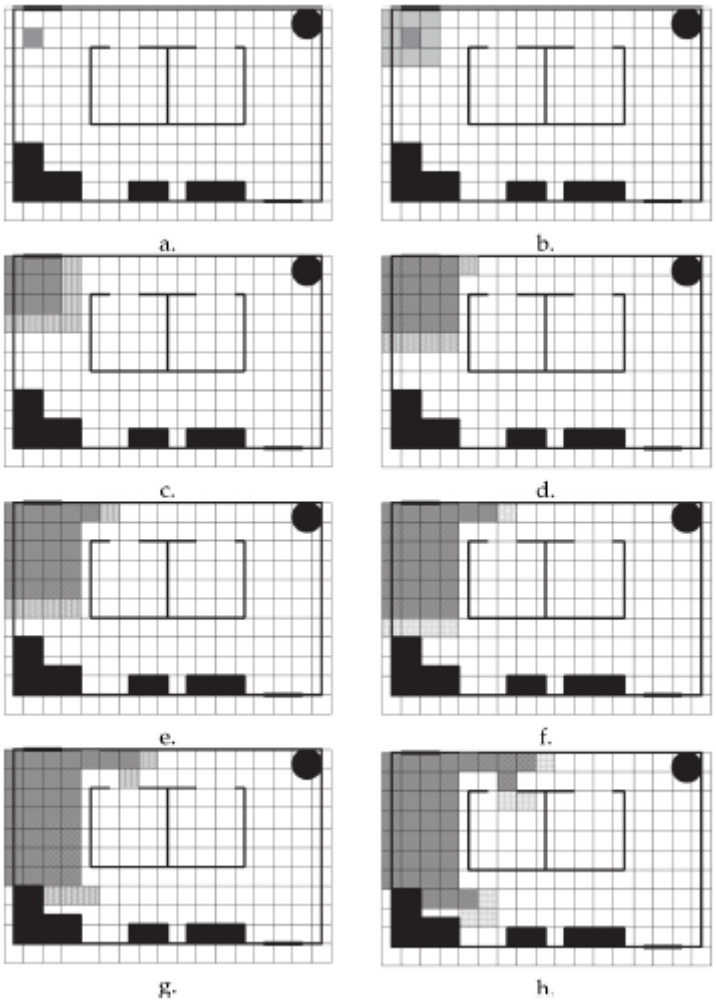
f*(F) = g*(F) + h*(F) = 2.4 + 1.0 = 3.4

Teraz algorytm widzi, że E jest najlepszym wyborem spośród liści drzewa wyszukiwania, w tym gałęzi do B. (Gdyby B był najlepszym wyborem, algorytm zmieniłby gałęzie). E jest lepszym wyborem niż F i B Kiedy algorytm rozwija E, zauważa, że E jest celem, więc algorytm jest zakończony. Optymalna ścieżka to A − D − E i nie musieliśmy jawnie rozważać A − B − F − E. Istnieją inne sposoby na ulepszenie opisanej do tej pory procedury. f*(F) nie trzeba było obliczać, jeśli algorytm przygląda się swoim wyborom i widzi, że jeden z nich jest celem. Każdy inny wybór ścieżki musi być dłuższy, ponieważ krawędzie nie mogą być ujemne, więc D − F − E musi być dłuższa niż D − E. Innym ważnym spostrzeżeniem jest to, że

każda ścieżka od A do E musi przechodzić przez D, więc gałąź B drzewa poszukiwań mogła zostać przycięta. Oczywiście w powyższym przykładzie algorytm nigdy nie miał okazji tego zauważyć, ponieważ B nigdy nie zostało rozszerzone. W przypadku większego wykresu łatwo sobie wyobrazić, że może wystąpić przypadek, w którym po kilku rozszerzeniach przez D, liść w B w drzewie wyszukiwania może okazać się najbardziej prawdopodobny. Wtedy algorytm rozszerzyłby A i zobaczył, że wybór padł na D. Ponieważ D wystąpiło już w innej gałęzi, z tańszym g*(D), gałąź B można było bezpiecznie przyciąć. Jest to szczególnie przydatne, gdy A* jest stosowane do silnie połączonego wykresu utworzonego ze zwykłej siatki w porównaniu do rzadkich wykresów, które występują w aplikacjach do planowania niezrobotyzowanego. Jedną z bardzo atrakcyjnych funkcji narzędzia do planowania ścieżki A* jest to, że można go używać z dowolną reprezentacją Cspace, którą można przekształcić w wykres. Główny wpływ, jaki Cspace ma na planner A*, to liczba obliczeń potrzebnych do znalezienia ścieżki. Ograniczeniem A* jest to, że bardzo trudno jest użyć go do planowania ścieżki, gdy podczas generowania ścieżki należy wziąć pod uwagę inne czynniki niż odległość. Na przykład odległość w linii prostej może obejmować teren skalisty lub piasek, co stanowi zagrożenie dla robota. Podobnie robot może chcieć unikać przejeżdżania przez wzgórza w celu oszczędzania energii, ale jednocześnie może chcieć zjeżdżać w dół, gdy tylko jest to możliwe, z tego samego powodu. Aby uwzględnić wpływ terenu na koszty ścieżki, należy zmienić funkcję heurystyczną h*(n). Przypomnijmy jednak, że nowa funkcja heurystyczna musi nadal spełniać warunek dopuszczalności: h*≤ h. Jeśli nowe h* przyniesie najgorszy koszt energii lub koszt bezpieczeństwa, będzie to dopuszczalne, ale nieszczególnie przydatne przy przycinaniu ścieżek. Ponadto, zdobywanie energii podczas zjazdów oznacza w istocie posiadanie na wykresie krawędzi z ujemną wagą, z którą A* nie może sobie poradzić. (Wagi ujemne stanowią interesujący problem, ponieważ robot może wielokrotnie wchodzić w pętlę toczenia się w dół wzgórza, ponieważ jest to rozwiązanie energooszczędne; jednak to rozwiązanie nie pozwala na żaden postęp w przód! Algorytmy typu Bellman-Ford radzą sobie z tym sytuacja.)

Planery oparte na Wavefront

Style propagacji Wavefront planistów dobrze nadają się do reprezentacji typu siatki, ale nigdy nie zyskały popularności w robotyce. Podstawową zasadą jest to, że front fali uważa przestrzeń C za materiał przewodzący, którego ciepło promieniuje od węzła początkowego do węzła docelowego. W końcu ciepło rozprzestrzeni się i osiągnie cel, jeśli istnieje sposób, jak pokazano w sekwencji na rysunku
.



Inną analogią dla planistów wavefront jest kolorowanie regionów w grafice, gdzie kolor rozprzestrzenia się na sąsiednie piksele. Dwa interesujące aspekty propagacji czoła fali to i) optymalna ścieżka od wszystkich elementów siatki do celu może być obliczona jako efekt uboczny oraz ii) wszelkie koszty poruszania się po terenie można uwzględnić jako inną przewodność. Propagacja Wavefront daje mapę, która wygląda jak potencjalne pole. Jednym z wielu typów narzędzi do planowania ścieżki z frontem falowym jest algorytm Trulla opracowany przez Kena Hughesa146. Wykorzystuje on podobieństwa z potencjalnym polem, dzięki czemu ścieżka sama w sobie reprezentuje to, co powinien zrobić robot, tak jakby ścieżka była obserwacją czujnika.

Wykonywanie zaplanowanej ścieżki

Większość algorytmów planowania ścieżki jest stosowanych w ściśle określonym stylu jednorazowo, reaktywnie wykonywanym. Prawie wszystkie techniki dzielą ścieżkę na segmenty; nawet planista wavefront faktycznie tworzy lokalizację celu (punkt drogi) dla każdego wektora kierunkowego. Dekompozycja ścieżki na segmenty jest dobrze dopasowana do architektury, w której kartograf przekazuje segmenty ścieżki lub całą ścieżkę sekwencerowi. Sekwencer może następnie zastosować serię zachowań typu „przejście do celu", dezaktywując i ponownie inicjując zachowanie po osiągnięciu nowego celu podrzędnego. Niestety są dwa problemy z reaktywnym wykonaniem ścieżki metrycznej zgodnie z opisem powyżej: obsesja podrzędna i brak oportunistycznego przeplanowania.

OBSESJA CELÓW POMOCNICZYCH

WARUNEK ZAKOŃCZENIA

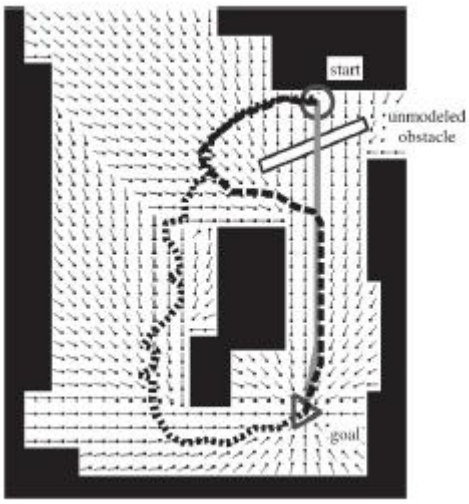
Obsesja podcelowa pojawia się, gdy robot spędza zbyt dużo czasu i energii, próbując osiągnąć dokładną pozycję podcelową, a dokładniej, gdy warunki zakończenia są ustawione z nierealistyczną tolerancją. Problem z ustawieniem warunku zakończenia dla celów cząstkowych najlepiej opisać na przykładzie. Załóżmy, że następny punkt orientacyjny znajduje się w lokalizacji (35, 50). Jeśli robot ma enkodery wału lub GPS, ustalenie, czy robot dotarł do punktu drogi, powinno teoretycznie być proste. W praktyce bardzo trudno jest robotowi, nawet robotowi holonomicznemu, dotrzeć dokładnie w dowolne miejsce, ponieważ robotowi trudno jest precyzyjnie się poruszać. Robot może dosięgnąć (34,5, 50). Zachowanie pokazuje, że cel jest teraz o 0,5 metra do przodu, a robot porusza się ponownie, próbując dosięgnąć (35, 50). W tym ruchu może przestrzelić i wylądować na (35,5,50,5). Teraz musi się obracać i ruszać ponownie, powodując ruchy kołyszące, które mogą trwać przez kilka minut. To marnuje czas i energię oraz sprawia, że robot wydaje się nieinteligentny. Problem pogłębiają pojazdy nieholonomiczne, które muszą cofać się, aby skręcić w określone miejsce. Tworzenie kopii zapasowej prawie zawsze wprowadza więcej błędów w nawigacji. Aby poradzić sobie z obsesją na punkcie podrzędnych celów, wielu robotyków programuje w swoich zachowaniach polegających na przejściu do celu tolerancję na warunek zakończenia w celu osiągnięcia celu. Powszechną heurystyką robotów holonomicznych jest umieszczenie tolerancji +/- szerokości robota. Jeśli więc cylindryczny robot holonomiczny o średnicy 1,0 metra otrzyma cel (35, 50), zatrzyma się, gdy $34,5 < x < 35,5$ i $49,5 < y < 50,5$. Nie ma wspólnej heurystyki dla robotów nieholonomicznych, ponieważ manewrowość każdej platformy jest inna. Bardziej dokuczliwym aspektem obsesji podrzędnej jest sytuacja, w której cel jest zablokowany, a robot nie może osiągnąć warunku zakończenia. Rozważmy na przykład bramkę podrzędną na przeciwległym końcu hali od robota, ale hala jest zablokowana i nie ma sposobu na obejście. Ponieważ robot działa reaktywnie, niekoniecznie zdaje sobie sprawę, że nie robi postępów. Jednym z rozwiązań jest oszacowanie przez sekwencer maksymalnego dopuszczalnego czasu, w którym robot dotrze do celu. Ta ocena może być zaimplementowana jako parametr zachowania (zakończ z kodem błędu po n sekundach) lub jako wyzwalacz stanu wewnętrznego. Zaletą tego ostatniego rozwiązania jest to, że kod może stać się częścią monitora, prowadząc do pewnej formy samoświadomości.

Przeplanowanie

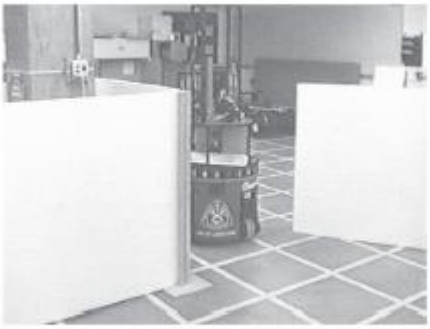
Z obsesją na punkcie celów pośrednich wiąże się fakt, że w realizacji planów często brakuje oportunistycznych usprawnień. Załóżmy, że robot kieruje się do celu pośredniego 2, gdy niemodelowana przeszkoda zawraca robota z zamierzonej ścieżki. Załóżmy teraz, że z nowej pozycji robot może dostrzec cel podrzędny 3. W klasycznym stylu implementacji zagnieżdżonego kontrolera hierarchicznego robot nie szukałby celu podrzędnego 3, więc kontynuowałby przejście do celu podrzędnego 2, bardziej optymalne byłoby, gdyby robot skierował się prosto do podcelu 3. Potrzeba oportunistycznego przeplanowania pojawia się również wtedy, gdy mapa a priori okazuje się nieprawidłowa. Co się stanie, gdy robot odkryje, że jest wysyłany przez błotnistą ziemię? Próba reaktywnego poruszania się po błotnistym terenie wydaje się nieinteligentna, ponieważ wybór lewej lub prawej strony może mieć poważne konsekwencje, jeśli robot utknie w błocie. Zamiast tego robot powinien zwrócić kontrolę do Kartografa, który zaktualizuje swoją mapę i przeplanuje. Pytanie brzmi: Skąd robot wie, że za bardzo zboczył z zamierzonej ścieżki i musi zmienić plan? Istnieją dwa podejścia do ponownego planowania. Jednym z nich jest ciągłe przeplanowywanie, zasadniczo narzucanie hierarchicznego cyklu Sens, Plan, Działanie. Drugim jest ponowne zaplanowanie, gdy wystąpi jakieś zdarzenie, wyjątek lub wskazanie, że wykonanie planu nie działa. Replanowanie sterowane zdarzeniami może być stosowane w planie hybrydowym, a następnie w architekturze Sense-Act, ale wymaga to dodatkowego monitorowania deliberatywnego.

D* ALGORYTM WYSZUKIWANIA

Algorytm wyszukiwania D* jest popularnym przykładem ciągłego przeplanowania, natomiast rozszerzenie algorytmu Trulla jest przykładem przeplanowania sterowanego zdarzeniami. Obaj planiści zaczynają od mapy a priori i obliczają optymalną ścieżkę z każdej lokalizacji do celu. D* robi to poprzez wcześniejsze przeszukanie A* z każdej możliwej lokalizacji do celu; to przeformułowanie przekształca A* z algorytmu najkrótszej ścieżki z jednym źródłem w algorytm dla wszystkich ścieżek. Algorytmy wszystkich ścieżek są obliczeniowo drogie i czasochłonne, ale nie stanowi to problemu, ponieważ ścieżki są obliczane, gdy robot rozpoczyna misję i siedzi nieruchomo. Ponieważ Trulla jest planistą typu wavefront, generuje optymalną ścieżkę między wszystkimi parami punktów w Cspace jako efekt uboczny obliczania ścieżki od lokalizacji początkowej do celu. Obliczenie optymalnej ścieżki z każdej lokalizacji do celu faktycznie pomaga w reaktywnym wykonaniu ścieżki. Oznacza to, że jeśli robot może zlokalizować się na mapie a priori, może odczytać optymalny podcel dla przejścia do celu przy każdej aktualizacji. Jeśli robot musi wykonać szeroki obrót, aby ominąć niemodelowaną przeszkodę na rysunku, robot zostanie automatycznie przekierowany na optymalną ścieżkę bez konieczności ponownego planowania.



a.


b.

Zwróć uwagę, jak ścieżka metryki staje się wirtualnym czujnikiem, kierującym przejściem do zachowania celu poprzez zastąpienie bezpośrednich danych czujnika. Jest to przykład interakcji między komponentami deliberatywnymi (planowanie ścieżki) i reaktywnymi (wykonawczymi) architektur hybrydowych.

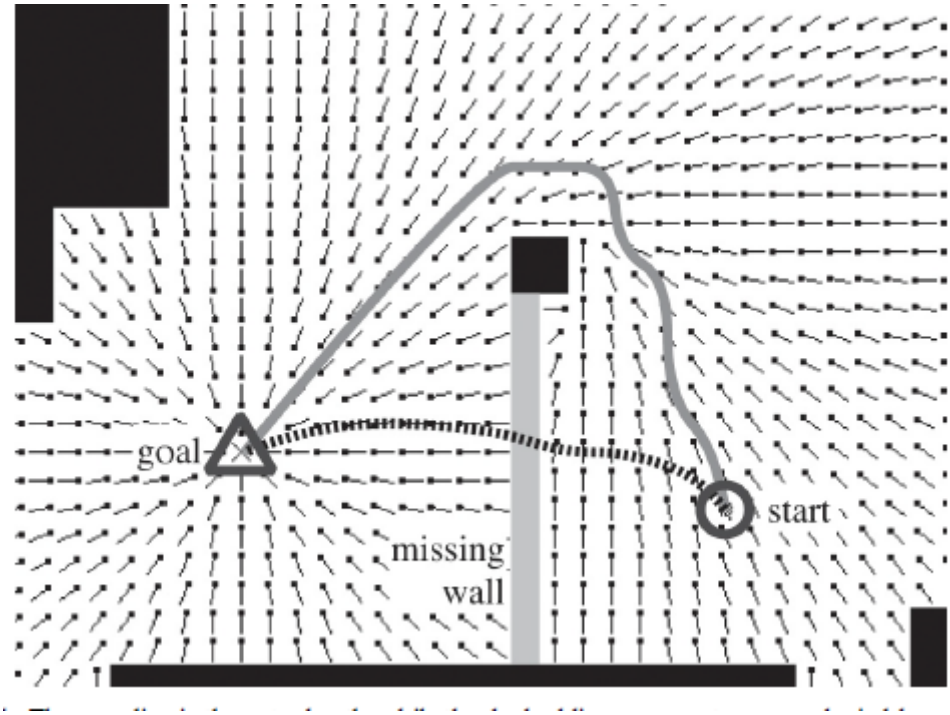
CIĄGŁA PRZEMIANA

Takie podejście eliminuje obsesję na punkcie podrzędnych celów, ponieważ robot może reaktywnie zmieniać „optymalne” ścieżki i oportunistycznie przemieszczać się do bliższego punktu. Jak większość rzeczy w życiu, zbyt wiele dobrych rzeczy jest złych. Jednak w pewnym momencie liczba niemodelowanych przeszkód może zmusić robota do uwięzienia lub wędrowania, zmieniając cele pośrednie, ale nie robiąc prawdziwych postępów. Rozwiązaniem D* tego problemu jest ciągła aktualizacja mapy i dynamiczna naprawa ścieżek A* dotkniętych zmianami na mapie. D* reprezentuje jedną skrajność na skali przeplanowania: ciągłe przeplanowanie. Ciągłe przeplanowanie ma dwie wady. Po pierwsze, robot z wbudowanym procesorem i ograniczeniami pamięci, taki jak łazik planetarny, może być zbyt kosztowny obliczeniowo, aby był praktyczny. Po drugie, ciągłe zmiany planowania w dużym stopniu zależą od jakości wykrywania. Jeśli robot wykryje niemodelowaną przeszkodę w czasie T1, oblicza nową ścieżkę i dokonuje dużej korekty kursu. Jeśli robot nie wykryje już tej przeszkody w czasie T2, ponieważ pierwszy odczyt był fantomem z szumu czujnika, ponownie obliczy kolejną dużą

korektę kursu. Rezultatem może być robot, który ma bardzo gwałtowne ruchy i faktycznie potrzebuje więcej czasu, aby dotrzeć do celu.

ZMIANA WYDARZENIA

W przypadku planowania ścieżki z wbudowanymi procesorami i zaszumionymi czujnikami, pożądane byłoby posiadanie pewnego rodzaju schematu sterowanego zdarzeniami, w którym zdarzenie zauważalne przez system reaktywny wywołałoby ponowne planowanie. Trulla używa iloczynu skalarnego zamierzonego wektora ścieżki i rzeczywistego wektora ścieżki. Gdy rzeczywista ścieżka odchyla się o 90° lub więcej, iloczyn skalarny wektora ścieżki i rzeczywistego wektora, po którym porusza się robot, staje się 0 lub ujemny. Dlatego iloczyn skalarny działa jak afordancja wyzwalająca ponowne planowanie: robot nie musi wiedzieć, dlaczego zbacza z kursu, tylko że wyraźnie zboczył z kursu. Jest to bardzo dobre w sytuacjach, które utrudniałyby postęp na pierwotnie obliczonej ścieżce, w efekcie w sytuacjach, w których świat rzeczywisty jest mniej podatny na osiągnięcie zamierzonego celu. Ale nie radzi sobie z sytuacją, w której rzeczywisty świat jest rzeczywiście bardziej przyjazny. Na rysunku 14.14 przeszkoda, o której sądzi się, że tam jest, tak naprawdę nie istnieje.



Robot mógłby osiągnąć znaczne oszczędności w nawigacji, oportunistycznie pokonując lukę. Taki oportunizm wymaga od robota dostrzeżenia, że świat jest naprawdę korzystniejszy niż pierwotnie modelowano. Ciągły replaner, taki jak D*, ma wyraźną przewagę, ponieważ automatycznie zauważy zmianę w świecie i odpowiednio zareaguje, podczas gdy Trulla nie zauważy korzystnej zmiany ponieważ nie doprowadzi to do odchylenia ścieżki. Otwarte pytanie badawcze, czy istnieją afordancje na dostrzeganie korzystnych zmian w świecie, które pozwalają robotowi na oportunistyczną optymalizację jego ścieżki.

Planowanie ruchu

Planowanie ruchu wywodzi się z gałęzi robotyki zajmującej się manipulatorami przemysłowymi, gdzie dynamika robota nie może być wyabstrahowana przez założenia holonomiczności, a kształt robota – lub części, którą niesie – nie może być zignorowany. Jako przykład tego, dlaczego robot może nie być holonomiczny, rozważmy, czy robot przenosi 225 kg części z jednej strony fabryki na drugą z dużą

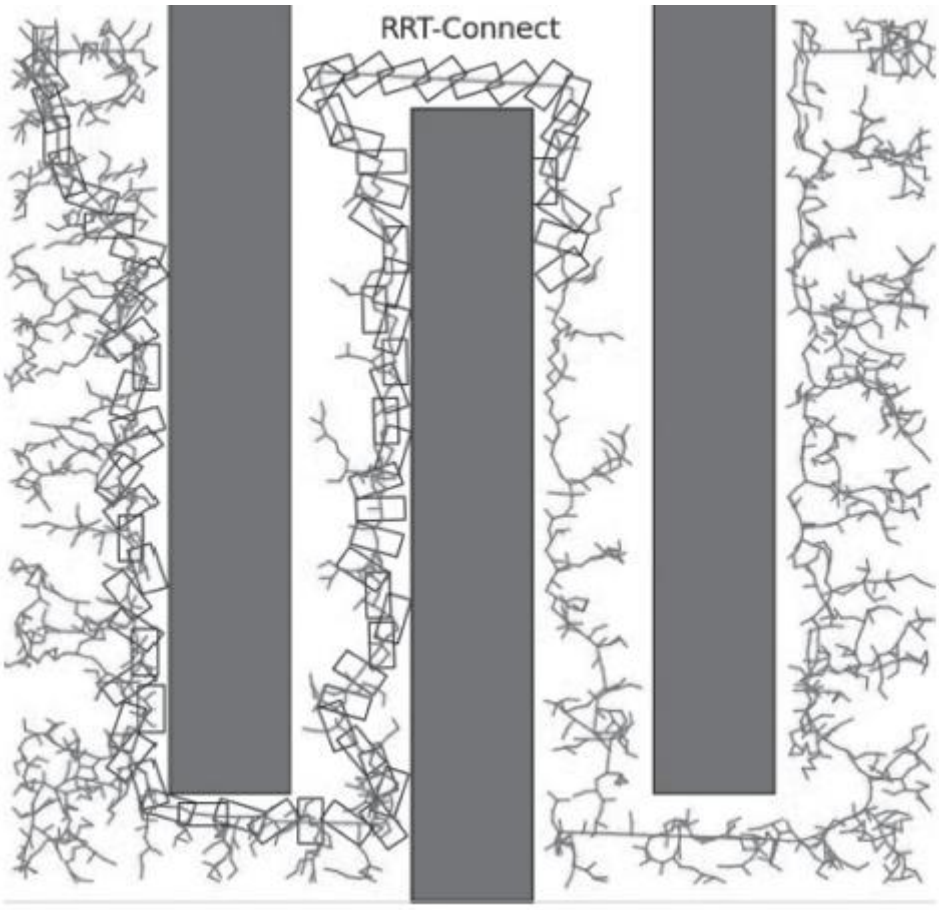
prędkością. Może być konieczne zwolnienie i skręcenie szerszego zakrętu lub ryzyko przewrócenia się. Jako przykład tego, dlaczego kształt jest ważny, weźmy pod uwagę, że robot może potrzebować przesunąć ramię przez otwór, a następnie podnieść je i ominąć przeszkody w celu złożenia części. W takich sytuacjach, w których przestrzeń planowania ma dużą liczbę wymiarów do rozważenia, tradycyjne metody planowania ścieżek generowania tras nie działają, a zatem wchodzą w zakres badań zwany planowaniem ruchu. Planowanie ruchu uwzględniające dynamikę pojazdu jest ogólnie traktowane jako problem sterowania, który modyfikuje planowanie i wykonanie ścieżki, dlatego nie jest omawiany w tej książce. Jednak planowanie ruchu, które obejmuje różne pozy i ograniczenia, jest często uwzględniane w sztucznej inteligencji i zostanie omówione poniżej.

PROBLEM PORUSZAJĄCY SIĘ FORTEPIANEM

Klasa aplikacji, w których nie można wyabstrahować pozy, jest często określana jako problem osób poruszających się na fortepianie, ponieważ poruszanie się robota w bardzo zagraconym środowisku 3D jest analogiczne do grupy osób, które planują obracanie i odwracanie fortepianu na każdym kroku w kolejności wnieść pianino przez drzwi, z mieszkania, w dół po schodach, przez kolejne drzwi i tak dalej. Analogia do poruszania się po fortepianie powinna jasno wskazywać, że pozycja robota w dowolnym punkcie ścieżki jest ważna, a zatem robot nie może być traktowany jako okrągła kropka ani przestrzeń konfiguracyjna zmniejszona do trzech stopni swobody. Powinno to również zilustrować, że możliwe kombinacje lokalizacji i pozycji są zasadniczo nieskończone. Na szczęście problem operatora fortepianu staje się możliwy do rozwiązania, jeśli celem jest po prostu wygenerowanie planu bezkolizyjnego, ale niekoniecznie optymalnego. W końcu celem jest wyciągnięcie pianina z mieszkania i zejście po schodach nieuszkodzone, a nie zrobienie tego optymalnie i szybko. Powszechnie akceptowaną metodą generowania bezkolizyjnej ścieżki jest użycie wariantu szybko ewoluującego algorytmu drzewa losowego. Mówiąc bardziej formalnie, w zagadnieniu poruszającego się po fortepianie, planowanie ruchu musi uwzględniać położenie i pozę (położenie, pozę) robota w każdym punkcie przestrzeni, a także to, czy możliwe jest przemieszczenie i obracanie w celu poruszania się (locationi, posei). do (locationi+1, posei+1) bez kolizji. Jednym ze sposobów spojrzenia na planowanie ruchu jest myślenie o każdym punkcie w przestrzeni jako o zestawie możliwych ważnych pozycji, w których robot może się znajdować bez dotykania ściany lub obiektu. Planowanie ścieżki musi uwzględniać wszystkie możliwe pozy, a nie tylko lokalizację. Najwyraźniej przestrzeń poszukiwań jest bardzo duża i należy ją wyabstrahować do czegoś, co jest bardziej łatwe do obliczenia. Dyskretowanie świata w regularną siatkę 3D lub ośmiornicę pomaga zmniejszyć przestrzeń poszukiwań, ale każdy element ma dużą liczbę póz. Następnie pojawia się kwestia przejść: jeśli robot jest w (locationa, posea) może być tylko pięć pozycji, w które może się przekształcić w (locationb, poseb), ale jeśli robot jest w (locationc, posec), może być nieskończony zestaw poprawnych póz w $location_b$.

SZYBKIE ODKRYWANIE LOSOWYCH DRZEW (RRT)

Uderzające inne podejście, zwane szybkim badaniem drzew losowych (RRT), oparte na pracy Stevena Lavalle, wykorzystuje losowość do próbkowania przestrzeni i póz. Rysunek ilustruje losowe drzewo generowane przez jeden wariant o nazwie RRT-Connect oraz częściową ścieżkę obliczoną przez drzewo.



Podstawowy algorytm wygląda następująco:

* Zacznij od mapy a priori i węzła początkowego (lokalizacja, pozycja), który służy jako korzeń drzewa.

* Losowo próbkuj przestrzeń i wygeneruj listę kandydujących (lokalizacja, pozycja) węzłów.

* Losowo wybierz węzeł kandydujący z listy i sprawdź, czy węzeł zawiera prawidłową pozę, w której robot lub część nie znajduje się wewnątrz ściany lub obiektu. Jeśli węzeł nie jest prawidłowy, powtarzaj, aż znajdziesz prawidłowy węzeł. Zauważ, że metody sprawdzają tylko, czy losowo wybrana pozycja (lokalizacja, pozycja) jest prawidłowa, nie wyszukują wszystkich prawidłowych pozycji w lokalizacji.

* W przypadku prawidłowego węzła kandydującego sprawdź, czy istnieje bezkolizyjna ścieżka między najbliższym wierzchołkiem w drzewie a węzłem, to znaczy, czy robot może fizycznie przemieszczać się i obracać od wierzchołka w drzewie do węzła kandydującego bez uderzając w cokolwiek. Jeśli tak, dodaj węzeł do drzewa. Jeśli nie, odrzuć kandydujący węzeł i wybierz inny.

* Kontynuuj, aż drzewo będzie zawierało cel.

* Po zbudowaniu drzewa zaplanuj ścieżkę przez drzewo.

Kryteria oceny planistów ścieżki i ruchu

W tym momencie powinno być jasne, że projektant robotyki ma do wyboru szeroką gamę technik, obejmujących prawie 30 lat badań. Jak w przypadku wszystkich innych robotów, wybór techniki zależy od niszy ekologicznej, w której robot będzie pracował. Minimalne kryteria oceny przydatności ścieżki lub planera ruchu to:

1. Złożoność. Czy algorytm jest zbyt wymagający obliczeniowo lub wymaga dużej ilości miejsca, aby można go było wykonać lub mieścić się w granicach robota? Wielu planistów UAV pracuje na laptopie przed lotem, ponieważ nie ma komputera pokładowego lub komputer pokładowy jest zbyt ograniczony.

2. Wystarczająco reprezentuje teren. Wielu badaczy pracuje w pomieszczeniach, które są płaskie. Roboty zewnętrzne mogą pracować w trudnym terenie ze stromymi wzniesieniami lub w niepożądanych obszarach, takich jak śliski piasek lub błoto. Jeśli algorytm planowania ścieżki jest zbudowany w celu generowania ścieżek z reprezentacji binarnej (obszar jest nawigowany lub nie), może to doprowadzić robota do kłopotów, jeśli zostanie zastosowany w bardziej zróżnicowanym środowisku.

3. Wystarczająco przedstawia fizyczne ograniczenia platformy robota. Roboty mają ograniczenia fizyczne. Najgłębszym ograniczeniem, które wpływa na planowanie ścieżki, jest to, czy robot jest holonomiczny, czy nie. Przypomnij sobie, że roboty holonomiczne mogą obracać się w miejscu, a prędkość może zmieniać się natychmiast.

4. Kompatybilny z warstwą reaktywną. Planiści ścieżki są z definicji ostrożni, ale warstwa reaktywna będzie odpowiedzialna za wykonanie ścieżki. Pożądana jest technika, która upraszcza tę transformację w wykonalne działania.

5. Obsługuje poprawki do mapy i przeplanowanie. Planowanie ścieżki i ruchu wymaga mapy a priori, która może okazać się bardzo błędna. Na przykład robot Quince, użyty w wypadku nuklearnym Fukushima Daiichi, został wysłany, by wspiąć się po schodach na trzecie piętro budynku Eactor; odkrył jednak, że spoczniki schodów były węższe niż pokazano na rysunkach powykonawczych, a zatem nie mogły się obracać. Robot musiał porzucić misję i wycofać się

schody. Dlatego robot może zacząć od jednej mapy, odkryć, że mapa jest nieprawidłowa i musi zaktualizować mapę i zmienić jej plan. Oczywiście pożądane są techniki takie jak D*, które pozwalają na naprawę istniejącego planu, a nie na złomowanie i obliczanie od zera.

Podsumowanie

Metryczne planowanie ścieżki przekształca przestrzeń świata w przestrzeń konfiguracji lub Cspace, która ułatwia optymalne planowanie ścieżki dla trasy. Istnieje wiele różnych sposobów przedstawiania obszaru lub objętości przestrzeni, ale wszystkie przekształcają przestrzeń w widok „z lotu ptaka”, niezależnie od pozycji i punktu widzenia robota. Reprezentacje Cspace zazwyczaj dają w wyniku wykres lub drzewo odpowiednie do wyszukiwania A*. Siatki regularne są obecnie najpopularniejszą reprezentacją Cspace w praktyce, chociaż uogólnione grafy Voronoi mają interesujące właściwości. Metody A* ogólnie zakładają, że robot jest holonomiczny, a metody Cspace, które zwiększają przeszkody do rozmiarów robota, pomagają uczynić to założenie wykonalnym. Metoda Cspace może prowadzić do błędów dyskretyzacji, a tym samym do ścieżek z fałszywymi zakrętami; można je wyeliminować za pomocą algorytmu relaksacji ścieżki lub algorytmu napinania struny. Metody metryczne często pomijają kwestię wykonania zaplanowanej ścieżki, zwłaszcza w odniesieniu do wpływu szumu czujnika lub niepewności lokalizacji. Dwa problemy związane z przeplataniem planowania ścieżki metryki z wykonaniem reaktywnym to obsesja na punkcie podrzędnych celów i kiedy należy ponownie zaplanować. Optymalne techniki planowania ścieżki dla map ustalonych a priori są dobrze zrozumiałe, ale mniej jasne jest, jak zaktualizować lub naprawić ścieżkę (ścieżki) bez rozpoczynania od nowa, jeśli robot napotka znaczące odchylenie od mapy a priori. Jednym z rozwiązań uchwyconych przez algorytm D* jest ciągłe przeplanowanie, jeśli pozwalają na to zasoby, a niezawodność czujnika jest wysoka; innym jest przeplanowanie sterowane zdarzeniami, które

wykorzystuje afordancje do wykrycia, kiedy należy przeplanować. Być może jeszcze poważniejszym pominięciem ograniczeń świata fizycznego w planowaniu tras jest to, że popularne narzędzia do planowania tras generowania tras mają zastosowanie tylko do pojazdów holonomicznych działających w stosunkowo otwartych przestrzeniach. Zamiast tego planowanie ruchu bada, jak zaplanować ruchy robota w bardzo zagraconej, wielowymiarowej przestrzeni, w której liczy się dynamika pojazdu i poza. Szybko eksplorująca losowa klasa algorytmów drzewa jest popularna w rozwiązywaniu problemu poruszenia fortepianu. Jeśli chodzi o sztuczną inteligencję, planowanie ścieżki i ruchu to w rzeczywistości problemy wyszukiwania, w których celem jest skuteczne znalezienie odpowiedzi w stogu siana wiedzy a priori. W tym przypadku odpowiedzią jest sekwencja punktów lub ruchów, które można powiązać z drzewem lub wykresem. Ten rodzaj planowania wykresów ma inny smak niż klasyczne machinacje planowania i rozwiązywania problemów widziane w Strips. Chociaż planowanie ruchu wykorzystuje losowość, nie jest to wnioskowanie. Przypomnijmy, że wnioskowanie wykorzystuje deliberację w celu dodania brakujących informacji, dostarczenia brakującej relacji między zestawami danych lub koncepcjami lub wywnioskowania nowej wiedzy. Planowanie ruchu wykorzystuje losowe próbkowanie, aby pomóc w trudnym obliczeniowo przeszukiwaniu tego, co można jawnie przedstawić, a nie wypełniać luki między znanym a nieznanym. Wracając do pytań postawionych we wstępie, odpowiedź na pytanie: Jaka jest różnica między nawigacją topologiczną a nawigacją metryczną lub planowaniem ścieżek? polega na tym, że nawigacja topologiczna koncentruje się na zapisanych, wykrytych trasach, podczas gdy metody metryczne polegają na wykorzystaniu map a priori do generowania optymalnych ścieżek lub sekwencji ruchów. W rozdziale opisano kilka różnych przestrzeni konfiguracyjnych i algorytmów prowadzących do praktycznego pytania: co jest powszechnie używane lub co działa wystarczająco dobrze? W praktyce najpowszechniejszymi algorytmami generowania tras są warianty A* lub D*, które uwzględniają rzeczywiste nieholonomiczne cechy robota oraz jego prędkość i trajektorię. Algorytmy te zazwyczaj wykorzystują zwykłą przestrzeń konfiguracyjną siatki. W planowaniu ruchu popularna jest klasa algorytmów RRT. Na koniec, ile planowania ścieżki potrzebujesz? ma wiele możliwych odpowiedzi. Z pewnością jedną ważną kwestią jest to, czy należy wziąć pod uwagę dynamikę lub pozę robota. Jeśli robota można aproksymować jako holonomicznego i działa on w stosunkowo otwartych przestrzeniach, takich jak drogi i korytarze, algorytm A* jest wystarczający do planowania trasy. Jeśli nie, potrzebne jest planowanie ruchu. Drugą kwestią jest to, czy pożądanym wyjściem jest trasa do miejsca docelowego, czy ścieżka, która maksymalizuje zasięg czujnika w danym obszarze. Trzecią kwestią jest przeplanowanie. Prawo Moore'a doprowadziło do sytuacji, w których obliczeniowo możliwe jest przeplanowanie na każdym kroku. Może to umożliwić osobie planującej ścieżkę wykrywanie i planowanie wokół przeszkód lub zmian na świecie, a nie reagowanie na nie. Jednak projektant może chcieć zachować rozróżnienie między namysłem a reakcją, a zatem przeplatanie planowania i wykonania może być bardziej eleganckie. Metryczne planowanie ścieżki i ruchu zależy od posiadania mapy świata i od tego, że robot może zlokalizować się na tej mapie podczas wykonywania planu. Reprezentacje i algorytmy w przestrzeni Cspace często nie biorą pod uwagę sposobu reprezentowania i wnioskowania o typach terenu, a przypadki specjalne, takie jak robot faktycznie oszczędzający lub generujący energię podczas zjazdu ze wzgórza, są zwykle ignorowane.