Міністерство освіти і науки України
Національний Університет «Києво-Могилянська Академія»
Кафедра математики

Планування Руху для Безпілотних Автомобілів
Motion Planning for Self-Driving Cars

Курсова робота
за спеціальністю "Прикладна математика" 6.040301

Керівник курсової роботи
К.ф.-м.н., доцент. Чорней Р.К.

_____
(підпис)
"__"_____ 2020 р.

Виконав студент
Кравченко І.В.
"__"_____ 2020 р.

Київ 2020

Зміст

Вступ

У сучасному світі автомобіль є одним з найбільш поширених видів транспорту і являється невід'ємною частиною життя більшості людей. Можливість людині індивідуально планувати власне пересування, не спираючись на зовнішні обставини, а також відносно швидко долати великі відстані, протягом багатьох століть вважалася нездійсненною мрією, але приблизно в другій половині 20-го століття автомобілі стали доступними більшості людей, здійснивши революцію у транспортному світі. У зв'язку з технічним прогресом якість автомобілів постійно покращується, роблячи їх більш безпечними, зручними та швидкими. Однак існує ряд проблем, які ускладнюють процес експлуатації авто, або роблять його незручним чи навіть небезпечним для водія та пасажирів та оточуючих учасників руху. Головною з таких проблем є людський фактор. За статистикою, близько 94% всіх дорожньо-транспортних пригод відбуваються через помилку людини. Друга проблема пов'язана із тим, що процес водіння автомобіля може займати істотну частину дня людини, і час, проведений за кермом, буде непродуктивний та виснажуючий. Боротьба із цими проблемами і є головною ціллю розробки безпілотних автомобілів. По-перше, поведінка безпілотного автомобіля позбавлена людського фактору: авто не втомлюється і не відволікається, тим самим забезпечуючи безпеку як пасажирам так й іншим учаникам руху. По-друге, під час пересування за допомогою безпілотного автомобіля, пасажири можуть проводити час набагато більш продуктивно, наприклад, їсти, читати та писати електронні листи, вчитися або працювати тощо. Сучасні комп'ютери щорічно збільшують обчислювальні потужності, доступні інженерам та науковцям, що дозволяє розглядати і вирішувати все більший клас задач. Завдяки цьому, на сьогоднішній день, запровадження повністю безпілотних автомобілів у буденне життя людства є лише питанням часу.

У цій роботі розглянуто проблему планування пересування для безпілотного автомобіля. Роботу розбито на 3 розділи:

1. У першому розділі буде поставлено та формалізовано задачу безпі-

лотного водіння, представлено ієрархічну модель автоматизованого пересування, а також узгоджено цілі та обмеження.

2. У другому розділі буде розгнянута проблема побудови маршруту від поточного місцезнаходження авто до точки призначення методами картографічної навігації, описано правила взаємодії безпілотного авто з іншими учасниками руху.Також побудовано модель планування поведінки безпілотного авто, яка дозволяє автомобілю обирати доречні сценарії поведінки залежно від обставин навколишнього середовища.

3. У третьому розділі буде поставлена задача локального планування, що полягає у прокладенні траєкторії від поточного місцезнаходження авто до пункту призначення у середі зі статичними перешкодами. Кінець розділу присвячено побудові глядкої траєкторії, що робитиме подорож комфортною для пасажирів.

Розділ 1

Задача Планування

1.1 Ієрархічна модель

Задача планування руху безпілотного автомобіля полягає в тому, щоб якнайшвидше подолати шлях із поточного місцезнаходження до місця призначення безпечним та комфортним шляхом, дотримуючись правил дорожнього руху. Очевидно, що така задача є дуже непростою, адже системі керування потрібно контролювати та оптимузівати відносно велику кількість критеріїв, а саме прокладати маршрут, обирати коректний сценарій водіння, реагувати на зміни навколишнього середовища(зокрема, статичні да динамічні прешкоди), генерувати оптимальну траєкторію пересування та відповідати за контроль швидкісного профілю. Для того, щоб спростити задачу, її розбивають на різні рівні абстракції, кожен з яких відповідає за деяку задачу і являє собою оптимізаційну проблему, і розв'язують ці задачі, рухаючи за рівнем абстракції від найбільшого до найнижчого.
Задачі виглядають наступним чином(за рівнем абстракції від найвищого до найнижчого):

1. Планування місії - навігація по карті, генерація оптмального маршруту по дорожним сполученням від точки поточного перебування авто до місця призначення.

2. Планування поведінки - обрання оптимального маневру для виконання відповідно до поведінки інших учасників дорожнього руху, правил на даній ділянці дороги.

3. Локальне планування - генерація кінематично можливого, оптимального маршруту без перешкод на шляху, а також генерація зручного швидкісного профілю для комфортного пересування.

1.2 Цільові функції

Після представлення ієрархічної моделі, ми звели задачу безпілотного керування до послідовності оптимізаційних задач. Для того, щоб оцінити ефективність рішень для кожної задачі а також отримати можливість порівнювати між собою ефективність різних рішень, необхідно запровадити деяку міру оптимальності, т. зв. цільову функцію, оптимуми якої нас цікавитимуть.

При плануванні шляху, найпростішим та найбільш інтуїтивним є бажання досягти місця призначення найбільш ефективним шляхом. Тому однією з найбільш розповсюджених цілей буде мінімізація довжини шляху, який автомобіль прокладе від точки перебування до місця призначення. Тоді цільова функція матиме вигляд довжини дуги між точкою відправлення та пунктом призначення і виражатиметься наступним чином:

$$s_f = \int_{x_i}^{x_f} \sqrt{1 + (\frac{dy}{dx})^2}\ dx \ \rightarrow min$$

Де s_f = накопичена довжина дуги, $x_i = x$-координата початкової точки знаходження авто(initial), $x_f = x$-координата кінцевої точки знаходження авто(final).

При плануванні швидкісного профілю, логічно хотіти потрапити до місця призначення якнайшвидше, тобто мінімізувати тривалість подорожі. Накопичений час при подорожуванні вздовж деякої дуги довжини s_f, згенерованої зазделегідь, знаходиться наступним чином:

$$T_f = \int_0^{s_f} \frac{1}{v(s)}\ ds \rightarrow min$$

У наступну чергу, задля забезпечення комфорту пасажирів під час поїздки, бажано зробити швидкість руху якомога більш плавною.

Означення 1.2.1 *Нехай для деякої задачі планування задано функцію зміщення автомобіля x, де $x(s)$ = положення авта в момент s. Тоді* ***ривком (англ. jerk)*** *називається швидкість зміни прискорення автомобіля відносно часу, тобто $J(s) = \dddot{x}(s)$.*

Для забезпечення плавності руху, прискорення автомобіля має змінюватися максимально повільно. Тоді буде доречним розглянути наступну

цільову функцію:

$$\int_0^{s_f} ||J(s)||^2 \, ds \; = \int_0^{s_f} ||\dddot{x}(s)||^2 \, ds \to min$$

Вона мінімізуватиме накопичений ривок, якого набуватиме авто під час руху вздовж траєкторії довжини s_f.

Розділ 2

Високорівневе планування

2.1 Планування місії

Планування місії являється найвищою за рівнем абстрактності задачею керування, і її основна ціль полягає в тому, щоб прокласти оптимальний маршрут з поточного місцезнаходження авта до місця призначення, базуючись на високорівневій інформації, такій як сполученість дорогами пунктів на карті, швидкісні обмеження тощо, нехтуючи більш низькорівневою інформацією про статичні та динамічні перешкоди, кінематичні та динамічні обмеження і т.д. Користуючись високоструктурованістю картографічних даних, можна доволі просто побудувати ефективну математичну модель для вирішення даної задачі, а саме відтворити наявну інформацію у вигляді зваженого графа, де множина вершин - роздоріжжя, а множина ребер - сегменти доріг, що їх сполучають, а ваги ребер - довжинам відповідних доріг. Таким чином, задача прокладення оптимального маршруту зводиться до пошуку шляху з вершини-початку до вершини-призначення, такого, що сума ваг ребер цього шляху буде мінімальною.

Класичним алгоритмом пошуку найкоротшого шляху по зваженому графу є алгоритм Дейкстри, який, у свою чергу, є модифікацією звичайного пошуку в глибину. Ключова різниця полягає в основній структурі даних: якщо для пошуку в глибину вершини зберігалися у черзі, то для алгоритму Дейкстри вершини зберігаються в мінімальній купі, де вони відсортовані по спаданню за вагою найкоротшого шляху з вершини-початку.

Algorithm 1: Алгоритм Дейкстри для (G, s, t)

```
open ← MinHeap();
closed ← Set() ;
predecessors ← Dict();
open.push(s, 0);
while not open.isEmpty() do
    u, uCost ← open.pop();
    if isGoal(u) then
        return extractPath(u, predecessors);
    end
    foreach v ∈ u.successors() do
        if v ∈ closed then
            continue;
        end
        uvCost ← edgeCost(G, u, v);
        if v ∈ open then
            if uCost + uvCost < open[v] then
                open[v] ← uCost + uvCost;
                predeccessors[v] ← u;
            else
                open.push(v, uCost + uvCost);
                predeccessors[v] ← u;
            end
        end
        closed.add(u);
    end
end
```

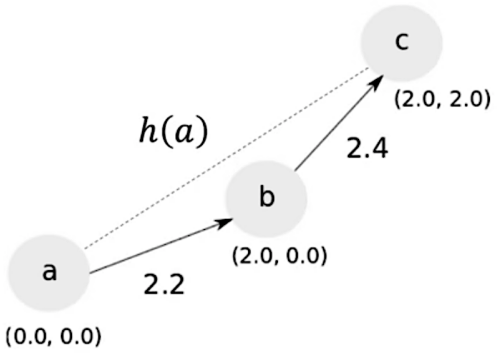
Однак алгоритм Дейкстри має складність $O(n^2)$, через що погано масштабується і не дуже добре підходить для розв'язку задачі планування місії, оскільки множини роздоріж та доріг зазвичай являються дуже великими. Для цього використовується покращення алгоритму Дейкстри, а саме алгоритм A^*. Покращення базуэться на припущенні, що відстань між вершинами графа(представлених роздоріжжями) можна обрахувати за допомогою Евклідової метрики.

$$h(v) = ||t - v||$$

Ця формула обраховує норму $h(v)$ (англ. heuristic) вектора-відстані між початковою вершиною v і ціллю t. Таке наближення зазвичай є доволі ефективним для припущення щодо істиної відстані між двома вершинами. Таким чином, приблизна відстань до цілі, яку нам залишиться

подолати при відвідуванні вершини v, дорівнює $h(v)$. Варто зауважити, що дане наближення буде майжде завжди *недооцінювати* відстань, що насправд залишиться подолати, адже $||x + y|| \leq ||x|| + ||y||$.

Рис. 2.1: Наближення відстані для A^*



Algorithm 2: Алгоритм A^* для (G, s, t)

```
Зеленим кольором позначено відмінності A* від алгоритму
 Дейкстри;
open ← MinHeap();
closed ← Set() ;
predecessors ← Dict();
open.push(s, 0);
while not open.isEmpty() do
    u, uCost ← open.pop();
    if isGoal(u) then
        return extractPath(u, predecessors);
    end
    foreach v ∈ u.successors() do
        if v ∈ closed then
            continue;
        end
        uvCost ← edgeCost(G, u, v);
        if v ∈ open then
            if uCost + uvCost + h(v) < open[v] then
                open[v] ← uCost + uvCost + h(v);
                costs[v] ← uCost + uvCost;
                predeccessors[v] ← u;
            else
                open.push(v, uCost + uvCost);
                costs[v] ← uCost + uvCost;
                predeccessors[v] ← u;
            end
        end
        closed.add(u);
    end
end
```

2.2 Взаємодія з динамічними об'єктами

2.2.1 Прогнозування руху

Ціллю прогнозування руху динамічних об'єктів є фіксація динамічних перешкод, та розрахування їх позиції, швидкості та напрямку руху. Ця задача є дуже важливою, адже вона дозволяє планувати маневри для

коректної взаємодії з динамічними об'єктами, уникати зіткнень під час руху запланованою траєкторією.

Для того, щоб задачу прогнозування руху динамічних перешкод було можливо розглянути з математичної точки зору, робляться деякі припущення стосовно поведінки учасників дорожнього руху, що робить аналіз їх переміщення суттєво простішим для прогнозування. Ці припущення можна роздиліти на дві категорії: перші роблять припущення щодо позиції об'єкта, інші - щодо його швидкості. Спершу розглянемо припущення про позицію об'єкта:

- Транспортні засоби, що рухаються вздовж даної смуги, зазвичай слідуватимуть цій смузі й надалі.
- Зміна смуг транспортним засобом муже бути спрогнозована через спостерігання індикаторного світла(поворотники тощо).
- На роздоріжжях, для прогнозування можливих шляхів пересування об'єкту, розглядаються всі можливі в даній ситуації траєкторії(зазвичай їх небагато, тому обчислювальної проблеми ця задача не становить).

Припущення щодо швидкості об'єкта:

- Якщо авто наближається до повороту зі значною кривизною, воно сповільнюватиметься.
- Регуляторні елементи впливають на швидкість учасників руху (світлофори, обмеження швидкості, знаки тощо).

2.2.2 Час до зіткнення

Час до зіткнення являється важливою мірою безпечності пасажирів автомобіля та інших учасників дорожнього руху, і відіграє ключову роль при виборі між потенційно виконуваними маневрами, залежно від поточної обстановки середовища. Знаючи про наявність потенційних зіткнень та час до їх виникнення, система керування має можливість краще планувати безпечні маневри у середовищі, або приготуватися до аварійних дій за необхідності. Задача полягаэ в тому, щоб, відстежуючи пересування учасників дорожнього руху, спочатку знайти точки, де можуть виникнути зіткнення, а потім знайти час до цих зіткнень.

Існує два підходи для розв'язання даної задачі: заснований на симуляції та заснований на наближенні. Підхід, що заснований на симуляції, полягає в тому, що прораховує на деякий час вперед траєкторії руху

учасників руху, базуючись на відомих даних про їх місцезнаходження, швидкість та напрям руху. Підхід, заснований на наближенні, з певною мірою точності обраховує слід(множина точок, на на площині, які займатиме авто) для деякого проміжку часу, і шукає перетини цих слідів, які відповідатимуть за точки зіткнень. У кожного метода є свої переваги та недоліки: симуляційний підхід дає набагато більш точні результати, але є доволі обчислювально дорогим, у той час як підхід, заснований на наближенні, є набагато менш вибагливим з точки зору обчислювальних потужностей, але ціною менш точних результатів. У цій роботі розглядатиметься симуляційний підхід. Алгоритм працює наступним чином: протягом деякого часу, для кожного учаснику руху обраховується місце у просторі, яке він займає. Це місце наближується за допомогою примітивних геометричних фігур, у переважній більшості випадків - кола. Потім покроково обраховується місцезнаходження кожного учасника руху, і якщо деякі кола-буфери перетинаються або дотичні, маємо потенційну точку зіткнення.

Algorithm 3: Algorithm Constant Velocity $TTC(D, T, dt, N_C)$

Data: D - список динамічних об'єктів, dt - час між кроками симуляції, N_c - кількість кіл для апроксимації

Result: P_{coll} - множина точок зіткнення, TTC(time to collision) - список часів до зіткнень

$t \leftarrow 0$;
$x_0 = x_{obj}$;
while $t < T$ **do**
    $t = t + dt$;
    for $i \in 1, \ldots, |D|$ **do**
        $d_i.x_t \leftarrow PositionEstimation(d_i, t)$;
        for $i \in 1, \ldots, |D|$ **do**
            $d_j.x_t \leftarrow PositionEstimation(d_i.x_t, d_j.x_t, N_C)$;
            if $P_{coll,i,j}$ **then**
                $TTC \leftarrow t$;
return P_{coll}, TTC

2.3 Планування поведінки

Система планування поведінки планує множину високорівневих дій та маневрів для забезпечення безпечного та ефективного досягнення місії водіння при різноманітних ситуаціях. Вона бере до уваги правила дорожнього руху, статичні та динамічні об'єкти довколо автомобіля. Система

планування поведінки має вміти виконувати наступні маневри:

1. Підтримувати швидкість - зберігати поточну швидкість пересування.

2. Слідувати за лідером - підлаштовувати свою швидкість під швидкість автомобіля попереду та підтримувати з ним безпечну дистанцію.

3. Сповільнюватися до зупинки - почати сповільнювати рух та зупинитися до заданої точки.

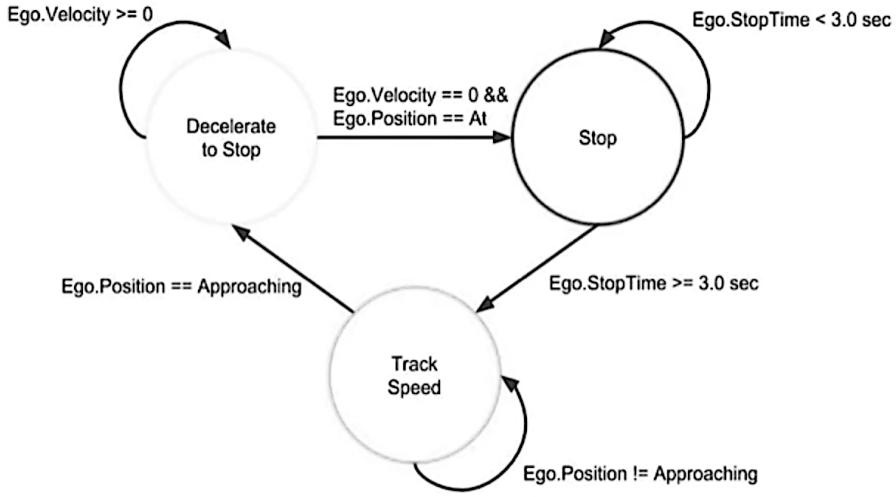
4. Зупинка - залишати автомобіль нерухомим в поточному місці.

5. Злиття - приєднатися або перейти на іншу смугу дороги.

Система повертає маневр для виконання, беручи до уваги обмеження, такі як ідеальний шлях(переважно центр поточної смуги), обмеження швидкості на даній ділянці дороги, границі смуги, місця для зупинки(світлофори, перехрестя, знаки "стоп"), оточуючі учасники дорожнього руху.

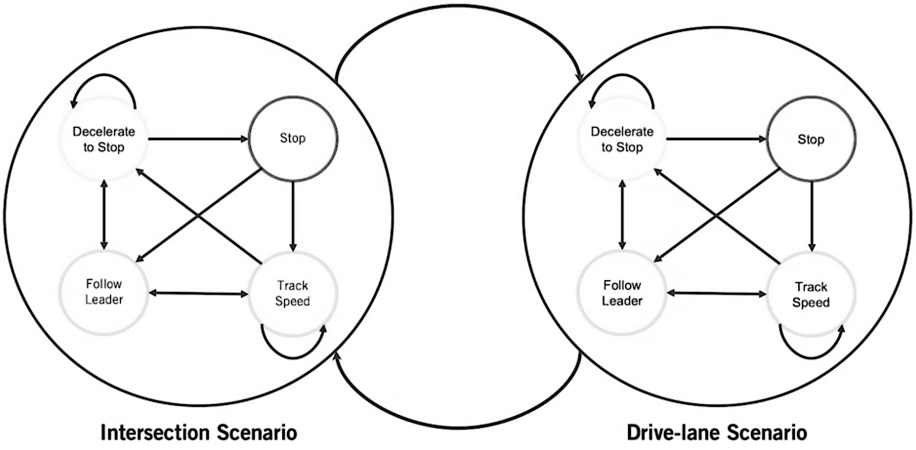
Класичним підходом для моделювання системи планування поведінки є скінченний автомат, де множина станів - це множина маневрів, а функція переходу задає перехід від виконання одних маневрів до інших, де перехід здійснюється за виконання деякої умови. Перевагами такого підходу до реалізації системи планування поведінки є обмежена кількість перевірок правил(оскільки перевіряти потрібно лише умови переходу з поточного стану), спрощення складних маневрів до більш простих, чітка структура та простота реалізації. Однак у такого підходу є і свої недоліки: при збільшенні кількості станів, прослідковувати зв'язки між ними та задавати правила переходу стає набагато складніше; також модель скінченного автомату не надає можливості реагувати не неоднозначні ситуації та ситуації, що не були явно передбачені інженерами. Один скінченний автомат дуже добре підходить для моделювання якогось конкретного сценарію водіння, наприклад їзда через перехрестя, але побудова одного скінченного автомату для обробки всіх сценаріїв водночас не є оптимальним варіантом, оскільки при доволі великій кількості станів стається "вибух правил коли задавати правила переходу для кожного стану та відстеження зв'язків між ними стає дуже важкою задачею. Також одночасна обробка великої кількості критеріїв переходу є обчислювально вибагливою задачею. В результаті таку систему дуже важко моделювати та підтримувати. Для вирішення даної проблеми моделювання системи планування поведінки відбувається за допомогою

Рис. 2.2: Скінченний автомат для маневру з 3 станів



ієрархічного скінченного автомату, у якого кожен суперстан являється скінченним автоматом. Тоді задача зводиться до того, щоб для кожного маневру створити власний скінченний автомат, який буде виконувати поставлену задачу, а потім створити глобальний скінченний автомат, кожен стан якого буде маневром, і задати правила переходу від одного маневру до інших.

Рис. 2.3: Ієрархічний скінченний автомат для двох маневрів з 4 станів



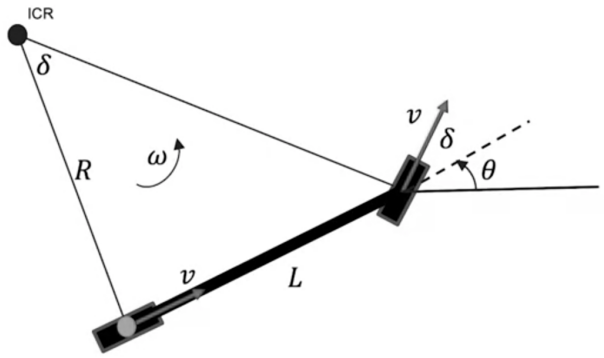
Розділ 3

Планування траєкторії

3.1 Кінематична модель

Автомобіль являється системою, стан та можливості якої підпорядковуються деяким фізичним та технічним законам і обмеженням. Фізичні обмеження, що діють на авто, мають дві природи: кінематичну та динамічну. Кінематичні обмеження зумовлені геометрією автомобіля та особливостями конструкції, у той час як динамічні обмеження представлені силами, що діють на автомобіль в даний момент часу(моменти, прискорення, крутний момент(torque) тощо). Для моделювання при відносно невеликіх швидкостях достятньо розглядати лише кінематичну модель. Для моделювання кінематичних властивостей автомобіля використовують кінематичну велосипедну модель, яка являє собою вісь із переднім та заднім колесами, кожне з яких відповідає за передню і задню пари коліс автомобіля відповідно. Напрям руху автомобіля задається кутом θ від-

Рис. 3.1: Кінематична велосипедна модель для автомобіля



носно осі OX інерційної системи координат. Кут повороту(steering angle) позначається як δ і дорівнює куту між вектором швидкості(velocity) v для переднього колеса та віссю автомобіля. Швидкість задається за допомогою вектора, який вказує в сторону, куди повернуте кожне колесо. Оскільки керованою є лише передня вісь автомобіля, швидкість заднього колеса завжди вказуватиме у напрямку вздовж осі авта. У якості точки, яка відображатиме положення автомобіля на площині, можна будь-яку точку на осі, але зазвичай береться середня точка заднього колеса, що відповідає за задню вісь автомобіля. Тоді можна обрахувати миттєвий центр обертання за допомогою насутпного рівняння:

$$\dot{\theta} = \omega = \frac{v}{R} \tag{3.1}$$

Також, з подібних трикутників, що утворюються сторонами L, R та v, δ, справедливим буде наступне відношення:

$$\tan\delta = \frac{L}{R} \tag{3.2}$$

Об'єднавши 3.1 та 3.2, отримуємо наступне рівняння швидкості обертання:

$$\dot{\theta} = \omega = \frac{v}{R} = \frac{v\tan\delta}{L}$$

За допомогою цих рівнянь можна описати кінематичну велосипедну модель для точки на задній осі (x_r, y_r). Компоненти вектору швидкості в напрямку осей OX, OY та швидкості обертання θ дорівнюють відповідно:

$$\dot{x_r} = v\cos\theta$$

$$\dot{y_r} = v\sin\theta$$

$$\dot{\theta} = \frac{v\tan\delta}{L}$$

Стан моделі у довільний момент можна описати за допомогою поточного місцезнаходження (x, y), напрямку руху θ та кута повороту δ, а керування системою відбувається шляхом зміни швидкості v та швидкість зміни кута повороту ϕ. Керування кутом повороту через швидкість його зміни зумовлена тим, що в реальності кут повороту неможливо миттєво радикально змінити, тому краще ввести обмеження для швидкості його зміни. Формалізуючи наведені міркування, отримуємо наступне:
Стан: $[x, y, \theta, \delta]^T$, Вхідні дані: $[v, \phi]^T$.

В кінцевому результаті отримуємо систему рівнянь, що пов'язує стан системи та вхідні дані:

$$\dot{x_r} = v\cos\theta$$
$$\dot{y_r} = v\sin\theta$$
$$\dot{\theta} = \frac{v\tan\delta}{L}$$
$$\dot{\delta} = \phi$$

Обчислювати на кожному кроці напряму систему диференціальних рівнянь доволі складно та невигідно, тому задля спрощення даної ситуації, кожне диференціальне рівняння дискретизується:

$$x_n = \sum_{i=0}^{n-1} v_i\cos(\theta_i)\Delta t = x_{n-1} + v_{n-1}\cos(\theta_{n-1})\Delta t$$

$$y_n = \sum_{i=0}^{n-1} v_i\sin(\theta_i)\Delta t = x_{n-1} + v_{n-1}\sin(\theta_{n-1})\Delta t$$

$$\theta_n = \sum_{i=0}^{n-1} \frac{v_i\tan(\delta_i)}{L}\Delta t = \theta_{n-1} + \frac{v_{n-1}\tan(\delta_{n-1})}{L}\Delta t$$

Таким чином, обчислення становляться більш ефективними, а рекурсивність сум дозволяє не обчислювати щоразу одні й ті самі значення, а додавати нову інформацію до збереженої протягом попередніх ітерацій.

3.2 Реактивне планування у статичному середовищі

Головна ціль задачі прокладання траекторії полягає в тому, щоб, маючи інформацію про статичне середовище, прокласти через нього шдях без перешкод, обраховуючи потрібні для цього вхідні дані для керування кінематичною велосипедною моделлю.

3.2.1 Алгоритм розгортання траекторії

Алгоритм розгортання траекторії є класичним підходом до генерації задовільної траекторії руху без перешкод у статичному середовищі. Суть алгоритма полягає в тому, щоб варіюючи деякий параметр вхідних даних, прокласти з поточного місцезнаходження деяку множину потенційних траекторій, потім серед них відсіяти ті, що мають перешкоди на

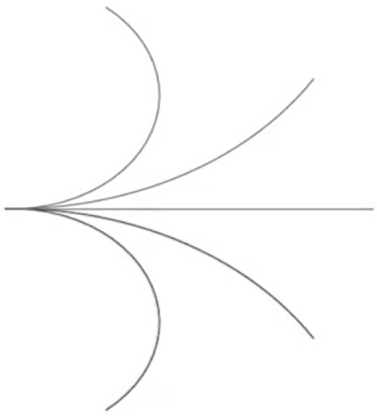
своєму шляху, і під кінець із множини залишених траекторій обрати ту, що накраще наближає авто до цілі.

Першим кроком є утворення множини потенційних траекторій. Кожна з траекторій отримується шляхом надання системі вхідних даних із фіксованими значеннями(фіксована швидкість і кут повороту) та прогнозування поведінки системи під дією цих даних протягом досить невеликого проміжку часу. Чим більша кількість потенційних траекторій, тим більш гнучною буде система, однак менша кількість траекторій веде до зниження обчислювальної вартості операції генерації множини траекторій.

Відсіявши траекторії, на шляху яких виникли перешкоди, потрібно певним чином оцінити решту траекторій та вибрати серед них найкращу. Одним з класичних методів оцінки є відстань від кінця прокладеної траекторії x_n до точки призначення x_{goal}:

$$D = ||x_n - x_{goal}||$$

Рис. 3.2: Множина траекторій для автомобіля при постійній швидкості та зміні кута повороту



Для підвищення зручності траекторії, побудованої за допомогою алгоритму розгортання, ефективним засобом буде накладення деяких обмежень на множину вхідних даних для керування ситсемою. Нехай стан системи задається вектором $[x, y, \theta]^T$, де θ - поточний напрям руху, що задається у вигляді кута між віссю машини та віссю OX, а вхідні дані подаються у вигляді вектора $[v, \delta]^T$, де v - швидкість, δ - кут повороту, на значення якого накладаються деякі обмеження, зазвичай зумовлені обмеженнями дороги та конструкції автомобіля. Тоді система описується

наступними рівняннями:

$$\dot{x_r} = v\cos\theta$$

$$\dot{y_r} = v\sin\theta$$

$$\dot{\theta} = \frac{v\tan\delta}{L}$$

$$\delta_{min} \leq \delta \leq \delta_{max}$$

$$v_{min} \leq v \leq v_{max}$$

Для підвищення комфортності поїздки, можна додати обмеження на прямолінійне($\ddot{x}$) та кутове($\ddot{\theta}$) прискорення:

$$\ddot{x}_{min} \leq \ddot{x} \leq \ddot{x}_{max}$$

$$\ddot{\theta}_{min} \leq \ddot{\theta} \leq \ddot{\theta}_{max}$$

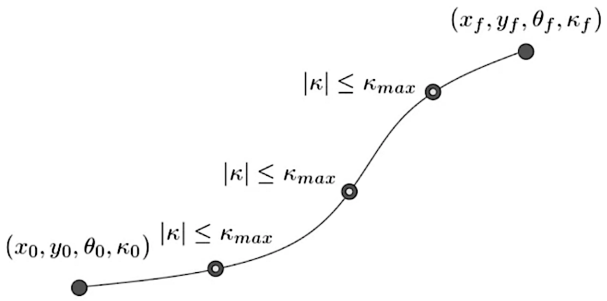
Варто зазначити, що введення додаткових обмежень хоча й робить можливі до виконання траєкторії більш комфортними для пасажирів, але воно також звужує множину потенційних траєкторій, що знижує маневренність авта.

3.3 Гладке локальне планування

Задача гладкого локального планування полягає в тому, щоб, маючи поточну позицію (x_0, y_0), напрям руху θ_0 і кривизну траєкторії κ_0, знайти шлях до кінцевої точки (x_f, y_f) з кінцевим напрямком руху θ_f і кривизною траєкторії κ_f, що задовольняють кінематичні обмеженням. У контексті оптимізаційної задачі, початкові та кінцеві значення авта можуть слугувати за граничні умови. Єдиним кінематичним обмеженням буде обмеження максимальної кривизни траєкторії $\forall x \in (x_0, x_f) : |\kappa(x)| \leq \kappa_{max}$. Забезпечити виконання даної умови для кожної точки з (x_0, x_f) - дуже складна обчислювально громіздка задача, тому для спрощення траєкторія розбиваєтья деякою невеликою множиною точок, і задовільнення умови щодо кривизни перевіряється тільки в них. Для спрощення вигляду оптимізаційної задачі, траєкторія буде задана як параметрична крива у вигляді поліноміальної спіралі, яка являє собою функцію, що задається як залежність кривизни спіралі від довжини дуги.

$$\kappa(s) = a_3 s^3 + a_2 s^2 + a_1 s + a_0$$

Рис. 3.3: Граничні умови для задачі гладкого планування



Тоді інші параметри системи в момент s виражаються через кривизну κ наступним чином:

$$\theta(s) = \theta_0 + \int_0^s a_3 s'^3 + a_2 s'^2 + a_1 s' + a_0 ds = \theta_0 + a_3 \frac{s^4}{4} + a_2 \frac{s^3}{3} + a_1 \frac{s^2}{2} + a_0 s$$

$$x(s) = x_0 + \int_0^s \cos(\theta(s'))ds'$$

$$y(s) = y_0 + \int_0^s \sin(\theta(s'))ds'$$

Незважаючи на складний процес виводу параметрів θ, x, y з кривизни κ, така функція все ж є відносно зручною, оскільки дозволяє швидко та зручно перевіряти чи задовольняє траекторія умови кривизни $|\kappa(x)| \leq \kappa_{max}$ для даного x. Значним недоліком такого підходу є відсутність явного рішення для $x(s), y(s)$, через що доводиться наближати їх значення за правилом Сімпсона. Для знаходження найкращої траекторії потрібно побудувати оціночний функціонал, який прийматиме на вхід альтернативні параметризації траекторій та перевірятиме, чи задовольняють вони крайнім умовам задачі, а також обрати найкращу за деяким критерієм (зазвичай - мінімізація кривизни).

Розділ 4

Висновки

У даній роботі було поставлено та формалізовано математичною мовою задачу безпілотного водіння, представлено ієрархічну модель планування, та розглянуто кожен рівень планування, а також наведено деякі приклади цільових функцій для оптимізації задач водіння різних рівнів абстракції. Навігаційна задача планування місії зведена до пошуку по зваженому графу алгоритмом A^*, наведено механізми для планування поведінки та руху серед інших динамічних об'єктів. Проілюстровано один із головних способів реалізації системи планування поведінки - за допомогою ієрархічного скінченного автомату. Серед більш низькорівнених задач планування траєкторії руху, було розглянуто та побудовано кінематичну велосипедну модель автомобіля як керовану систему з вектором-станом і вектором із вхідними даними. Також було наведено класичний алгоритм прокладання маршруту у статичному середовищі - метод розгортання траєкторії. Насамкінець, у роботі було коротко описано основні засади та підходи до побудови гладкої параметризованої траєкторії для забезпечення комфортної подорожі пасажирам.

Бібліографія

[1] Steven. M. Lavalle. *Planning Algorithms*. Cambridge University Press, 2006.

[2] P. Polack, F. Altche, B. Dandrea-Novel, and A. D. L. Fortelle *The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles*. 2017 IEEE Intelligent Vehicles Symposium (IV), 2017.

[3] C. Urmson, C. Baker, J. Dolan, P. Rybski, B. Salesky, W. Whittaker, D. Ferguson, and M. Darms. *Autonomous Driving in Traffic: Boss and the Urban Challenge*. AI Magazine, vol. 30, no. 2, p. 17, 2009.