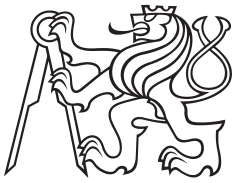
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra řídicí techniky

Plánování trajektorie pro autonomní vozidla

Antonín Hruška

Vedoucí: Mgr. Matěj Novotný
Obor: Robotika a Kybernetika
Květen 2020

Poděkování

Děkuji Mgr. Matěji Novotnému za vedení mé bakalářské práce, zvláště za jeho cenné rady, věcné připomínky a vstřícnost, se kterou mě provázel po celou dobu tvorby. V neposlední řadě i za jeho lidský přístup a životní pohled, kterým mě během práce obohatil a inspiroval.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškerou použitou literaturu. Dále prohlašuji, že tato bakalářská práce nebyla využita k získání jiného nebo stejného titulu.

V Praze, 22. května 2020

Abstrakt

Bakalářská práce se zabývá plánováním trajektorie pro autonomní vozidla. Jejím cílem je seznámit laického čtenáře s širší problematikou plánování trajektorie pro neholonomní robotická vozidla, která se pohybují v dvoudimenzionálním prostoru. Přiblížit mu typické reprezentace plánování trajektorie a představit přístupy, jež jsou v praxi běžně využívány.

Práce odkáže na řešení pomocí časoprostorové mřížky, k jejíž tvorbě jsou využity η-spline křivky. K reprezentaci překážek je využívána struktura bitmapy. Tento přístup umožňuje jednoduchou reprezentaci dynamických překážek a real-time využití v praxi.

Výstupem je implementace tohoto přístupu pomocí `Python` a `C` kódu a názorná simulace vzorové situace, která ověří funkčnost implementace.

Klíčová slova: autonomní vozidla, plánování trajektorie, prostorová stavová mřížka, diskrétní mapová mřížka, η-spline

Vedoucí: Mgr. Matěj Novotný
Český institut informatiky, robotiky a kybernetiky, ČVUT v Praze (CIIRC), Jugoslávských partyzánů 1580/3, Praha 6

Abstract

The subject of this thesis is trajectory planning for autonomous vehicles. An issue of a trajectory planning is theoretically introduced with a focus on two-dimensional nonholonomic robotic vehicles, including typical representations and approaches.

The solution to the issue of trajectory planning is presented, using the spatiotemporal state lattices and creating η-spline spatial trajectories. The obstacle collision avoidance is implemented through a bitmap structure. This approach allows simple representation of dynamic obstacles and computing solution to a particular problem in a real-time manner.

The outcome of this thesis is an implementation of the described solution through the `Python` and `C` code. A solved example is attached to prove basic functionality.

Keywords: car-like robot, path planning, space state lattice, occupancy grid, η-spline

Title translation: Trajectory planning for autonomous vehicles

Obsah

Obrázky

Tabulky

ctuthesis t1606152353

Kapitola 1

Úvod

Plánování trajektorie je jedním z významných odvětví průmyslové robotiky, neboť robot se v realném světě *musí* pohybovat, aby úspěšně plnil zadané úkoly. Uplatňuje se tak v každé činnosti, kde jsou roboti využíváni, ať už se jedná o autonomní vozidla, automatizované výrobní linky, zásobování, či složité medicínské operace. Objektivně se tak jedná o klíčovou část celé robotiky a je ihned po hardwarové části nutným předpokladem k běžnému využití robota. Úlohu plánování trajektorie se pokusíme čtenáři představit pomocí populárního *piano mover's problem*, který je jednoduchou instancí zmíněného problému. Cílem úlohy je přestěhovat piano (robota) v patře (prostředí) z obývacího pokoje (počátečního bodu) do ložnice (koncového bodu) tak, abychom se vyhnuli jakémukoliv střetu s nábytkem či stěnou (překážkami). Pomineme-li váhu takového piana, složitost úkolu spočívá v netriviální *geometrii* piana a *volném* prostředí, jež je vytyčeno překážkami. Pokud ve formulaci úlohy záleží pouze na tom, *zda* je robot schopen přejít z počáteční do koncové konfigurace, lze plánování trajektorie chápat jen jako formu vyhýbání se překážkám. Výsledkem je pak řešení diferenciálních rovnic modelu robota, splňující okrajové podmínky. Mnohem častěji však záleží i na tom, *jak* bude výsledná trajektorie vypadat. Takovouto formulaci lze modelovat takzvanými *omezeními*. Tato omezení se zvlášť využívají v prostředí, kde dochází k interakci s člověkem a jednoduché bezkolizní řešení nemusí splňovat lidské požadavky. Typickým příkladem je jízdní komfort v autě, kde je důležitá plynulost manévru a pohodlí posádky.[18]

Plánování trajektorie je obecně podřízeno schopnostem robota a jednotlivá řešení jsou odvislá od konkrétního typu robota. V našem případě považujeme za robota automobil, který budeme popisovat modelem diferenciálních rovnic. Jedná se tak o plánování trajektorie za diferenciálních omezení [kapitola 14

36] s důrazem na tvar trajektorie. Významnost této teorie je dána silnou poptávkou průmyslu, neboť automobil je celosvětově nejrozšířenějším dopravním prostředkem. Její poznatky lze však využít i pro roboty v průmyslu s podobnou konstrukcí, jako jsou například ještěrky a vysokozdvižné vozíky. Pro hlubší seznámení se s problematikou plánování trajektorie je vhodná již zmíněná kniha [36] či [33] nebo [32]. V [29] se lze seznámit s modely auta za učelem řízení.
Nejdříve se v teoretické části seznámíme s reprezentací statických a dynamických překážek (2.1), moderní teorií reprezentace plánování (2.2) a přehledem metod plánování trajektorie pro autonomní vozidla (2.3). Zmíníme také různé diferenciální modely aut (2.4). V praktické části představíme zvolený algoritmus a podrobně popíšeme jeho koncept a klíčové myšlenky (3.1). Ve výsledcích (3.2) uvedeme vzorovou situaci a řešení algoritmu, na kterém ověříme použitelnost algoritmu v praxi.

Kapitola 2

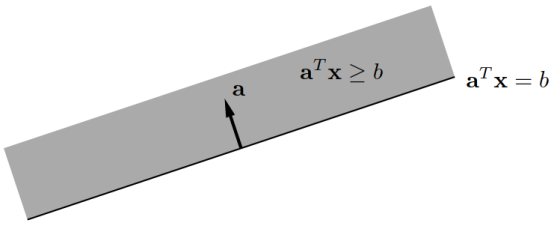
Teorie

V této kapitole najdeme základní kameny „moderní[1]“ teorie plánování trajektorie autonomních vozidel. V částech (2.1) a (2.2) se seznámíme se základními notacemi a v oddílu (2.3) pak probereme typické metody a reprezentace. V (2.4) probereme základní matematické modely mobilních robotů s důrazem na automobily.

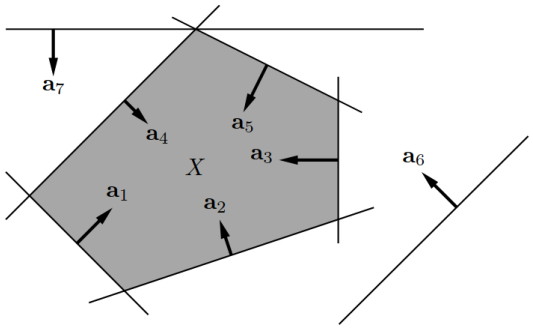
2.1 Geometrická reprezentace

Cílem tohoto oddílu je představit vhodnou reprezentaci *objektů* (robota, překážek) nutných pro plánování trajektorie. Předlohou k tomuto textu byla [kapitola 3 36]. Jako přímočarý přístup se jeví zvolit geometrickou reprezentaci, neboť jak překážky, tak robota lze chápat jako tuhé těleso. Opomeňme nyní fakt, že v některých formulacích stačí robota aproximovat pouze hmotným bodem. Tuhé těleso zaujímá nenulový objem, a má tudiž *tvar*, který lze reprezentovat *hranicí* či *vnitřkem*. Oba přístupy jsou běžné a preferují se podle konkrétní formulace problému.
Svět $\mathcal{W}$, ve kterém se robot vyskytuje, chápeme jako euklidovský prostor $\mathbb{R}^n$, kde $n \in \{2, 3\}$. *Oblast překážek* $\mathcal{O} \subset \mathcal{W}$ značí množinu všech bodů, jež se nachází uvnitř libovolné překážky. *Volným* prostorem pak rozumíme množinu $\mathcal{F} = \mathcal{W} \setminus \mathcal{O}$.
V následujících částech představíme geometrické primitivy, ze kterých půjde systematicky vystavět $\mathcal{O}$. Tato reprezentace musí dostatečně popisovat reálný

[1] Reprezentovanou pomocí konfiguračního prostoru, jež byl představen v 70. letech.



Obrázek 2.1: Polorovina Zdroj: [61]



Obrázek 2.2: Konvexní mnohostěn Zdroj: [61]

svět, zároveň však musí být jednoduchá pro efektivní výpočet počítačem. Z definice $\mathscr{O}$ bude snažší tvary reprezentovat pomocí jejich *vnitřků*.

Mnohostěny. Aproximace reálných objektů mnohostěny je v robotice běžně používána [21]. Jako výchozí stavební primitiv budeme považovat poloprostor (2.1). Poloprostor H definujeme

$$H = \{\boldsymbol{x} \in \mathscr{W} | \boldsymbol{a}\boldsymbol{x} \geq b\}. \tag{2.1}$$

Konvexní mnohostěn (2.2) pak vznikne konečným průnikem poloprostorů

$$X = H_1 \cap H_2 \cap \ldots \cap H_m = \{\boldsymbol{x} \in \mathscr{W} | \boldsymbol{a}_i\boldsymbol{x} \geq b_i \, \forall i = 1, \ldots, m\}. \tag{2.2}$$

Případné komplikovanější útvary (např. nekonvexní mnohostěny) zkonstruujeme pomocí konečných kombinací sjednocení, průniků či rozdílů. Zdůrazněme, že mnohostěnem může být dle definice prázdná množina. Znaménko v definici poloprostoru H lze libovolně bez újmy nahradit, ovšem obdobný výsledek lze docílit i vhodným použitím množinového rozdílu.
Díky De Morganovým zákonům lze výslednou nekonvexní či nesouvislou oblast $\mathscr{O}$ zapsat jako konečné sjednocení konečně mnoha průniků poloprostorů H. Je vhodné upozornit, že reprezentace $\mathscr{O}$ kovexními mnohostěny není jednoznačná. Ačkoliv by bylo výhodné minimalizovat počet takovýchto mnohostěnů, najít rozklad s minimálním počtem je NP složité [36]. Existuje však mnoho dekompozičních algoritmů s polynomialní složitostí [12].

Semialgebraické útvary. Přeformulujeme-li (2.1) na

$$H = \{\boldsymbol{x} \in \mathscr{W} | \boldsymbol{a}\boldsymbol{x} - b \geq 0\} = \{\boldsymbol{x} \in \mathscr{W} | f(\boldsymbol{x}) \geq 0\}, \tag{2.3}$$

získáme vhodný zápis, který lze přirozeně rozvinout. V případě semialgebraických útvarů nahradíme afinní funkci f polynomem s reálnými koeficienty. Skutečně dochází k přirozenému zobecnění, neboť i afinní funkce je polynomem. Pokud bychom se omezili pouze na polynomy druhého řádu, hranice těchto útvarů jsou kvadriky, resp. kuželosečky, mezi které například patří i kulová plocha, resp. kružnice.

Bitmapy. Další možnou reprezentací překážek jsou bitmapy. Zjednodušeně se jedná o diskretizaci spojitého světa $\mathscr{W}$ s libovolným rozlišením ϵ, která je popsána funkcí

$$\phi : \mathscr{W} \to \mathbb{Z}^n \tag{2.4}$$

$$\phi^{-1}(\boldsymbol{i}) = \prod_{j}^{n} [i - \epsilon, i_j + \epsilon) \tag{2.5}$$

kde i_j jsou složky $\boldsymbol{i}$. Překážky $\mathscr{O}$ můžeme reprezentovat pomocí n dimenzionální matice $M = (m_{i_1,i_2,\ldots,i_n})$:

$$m_{i_1,i_2,\ldots,i_n} = \begin{cases} 1 & \phi^{-1}(\boldsymbol{i}) \cap \mathscr{O} \neq \emptyset \\ 0 & \text{jinak} \end{cases} \quad , \boldsymbol{i} = \begin{pmatrix} i_1 & i_2 & \ldots & i_n \end{pmatrix}^T$$

Spojitost s předchozími modely je přímočará, neboť ϕ „rozsekalo“ $\mathscr{W}$ na krychle, resp. čtverce s délkou hrany 2ϵ. Výhoda této metody spočívá hlavně v přímočarém převodu reálných dat ze senzorů a jednoduché reprezentace, narozdíl od semialgebraických modelů, kde najít vhodný popis $\mathscr{O}$ může být komplikované.

Ostatní metody.

Nonuniform rational B-splines (NURBS). NURBS lze zapsat jako

$$C(u) = \frac{\sum_{i=0}^{n} \omega_i P_i N_{i,k}(u)}{\sum_{i=0}^{n} \omega_i N_{i,k}(u)}, \tag{2.6}$$

kde $\omega_i \in \mathbb{R}$ jsou váhy a P_i jsou kontrolní body. $N_{i,k}(u)$ jsou bázové funkce B-spline stupně k:

$$N_{i,k}(u) = \left(\frac{u - t_i}{t_{i+k} - t_i}\right) N_{i,k-1}(u) + \left(\frac{t_{i+k+1} - u}{t_{i+k+1} - t_{i+1}}\right) N_{i+1,k-1}(u).$$

Základ rekurze je $N_{i,0}(u) = 1$ pro $t_i \leq u < t_{i+1}$ a $N_{i,0}(u) = 0$ jinak [50].

Neboť se jedná o druh spline křivek, jsou zadané parametricky (narozdíl od implicitně zadaných algebraických útvarů (2.3)). Mimo reprezentaci překážek se tyto křivky uplatňují ještě v generování *hladkých* trajektorií, či v počítačové grafice [54].

Zobecněné válce. Zobecněné válce jsou přímočarým zobecněným „klasických" válců. Místo středové osy podstavy lze použít 3D spline-křivku $(x(s), y(s), z(s))$ parametrizovanou $s \in [0, 1]$ a poloměr rovněž paramterizovat (spojitou) funkcí $r(s)$ podel křivky. Jednou z výhod této reprezentace je jednoduchá interpretace člověkem (narozdíl např. od NURBS).

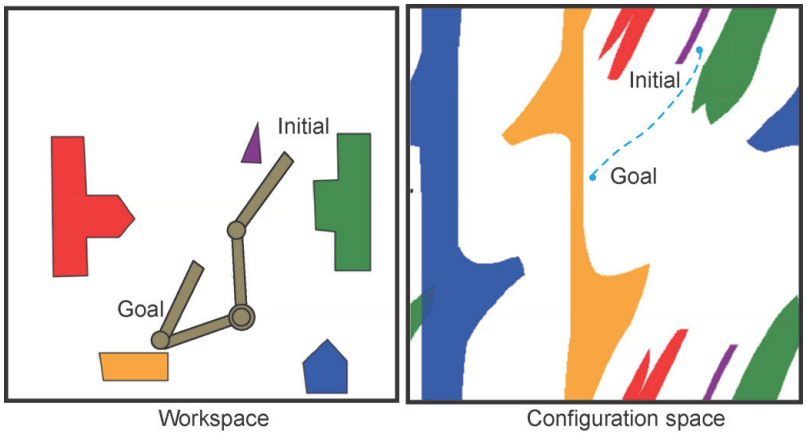
2.2 Konfigurační prostor

Konfigurační prostor (znám též jako C-space) byl poprvé představen na konci 70. let Lozano-Perézem and Wesleym [43, 42] a lze jej považovat za základní koncept novodobého plánování trajektorie.
Z matematického pohledu je konfigurační prostor varieta, jejíž struktura odpovídá konstrukci robota a obvykle je jeho dimenze rovna počtu stupňů volnosti robota. C-space lze také považovat za speciální stavový prostor, známý z teorie řízení systémů. [kapitola 4 36] Robot je v C-space znázorněn jednoznačně svou *konfigurací*.
Například pro autonomní vozidlo je konfigurace složena z pozice $(x, y) \in \mathbb{R}^2$ a azimutu $\theta \in [0, 2\pi)$. Konfigurační prostor C-space $= \mathbb{R}^2 \times \mathbb{S}$ je tedy válcem. Pro robotická ramena jsou prvky konfigurace typicky úhly a posuny, které lze řídit pomocí pohonu: $(\theta_1, \theta_2, \dots)$ [16]. Plánování trajektorie se tak zjednodušuje pouze na nalezení trajektorie z počátečního bodu do bodu koncového v konfiguračním prostoru [41].
Obdobně, jako jsme rozdělili *svět* $\mathscr{W}$ na *oblast překážek* $\mathscr{O}$ a její doplněk, i C-space lze rozdělit. Konfigurace blokované překážkami budeme značit C_{obs} a robotu dostupné konfigurace naopak C_{free}.
Formálněji: předpokládejme, že *robot* $\mathscr{A}$ a *oblast překážek* $\mathscr{O}$ jsou reprezentovány semialgebraickými útvary z 2.1. Nechť q je konfigurace robota $\mathscr{A}$. Pak $C_{obs} \subset C$ je definován jako

$$C_{obs} = \{q \in C | \mathscr{A}(q) \cap \mathscr{O}\} \tag{2.7}$$

a C_{free} jako

$$C_{free} = C \setminus C_{obs}. \tag{2.8}$$



Obrázek 2.3: C-space a svět $\mathscr{W}$ Zdroj: [46]

Ačkoliv je definice C_{obs} korektní, její explicitní vyjádření může být pro složitější druhy robotů komplikované a časově náročné (PSPACE-složité)[2][8] (viz 2.3). Toto však nemusí být jediné úskalí, zvláště u neholonomních systémů, neboť dle Mingueze: „ *I v absenci překážek je plánování přípustných trajektoriích pro neholonomní vozidla složité. Přesné řešení je známo pouze pro určité druhy systému, ale většina systému zůstává bez přesného vyjádření řešení. Obecně však lze používat i přibližná řešení.* “[s. 829 44][3]
Obdobně jako v 2.1, i zde lze využít k reprezentaci konfiguračního prostoru mnohostěny, či semialgebraické struktury [9]. Téměř nemožné je pak přesně konstruovat C_{obs} pro vysokodimenzionální C. S dnešním rozvojem neuronových sítí by však tento problém mohl být překonán a konfigurační prostor by mohl být odhadnut neuronovým klasifikátorem [46]. Jiným přístupem je reprezentace C-space přímo bez vyjádření C_{obs}.
Oba tyto přístupy vytvořily významné třídy algoritmů, s nimiž se seznámíme v následujícím oddílu 2.3. Algoritmům založeným na explicitním vyjádření C_{obs} se občas říká *přesné*, či *kombinatorické* [kapitola 6 36], neboť nedochází k aproximaci C_{obs}. Opakem jsou *vzorkovací* algoritmy, jež reprezentují C strukturami grafů [26], či stromů [30] a C_{obs} je v nich vyjádřen pouze implicitně, například pomocí příznakové funkce [16], která vychází z definice 2.7. Některé efektivní implementace takovýchto funkcí lze nalézt v [40].

[2] To implikuje NP složité.
[3] Překlad autora

2.3 Přehled algoritmů plánování trajektorie

2.3.1 Úvod

V této kapitole se seznámíme s běžnými koncepty algoritmů pro plánování trajektorie. Pro přehlednost uvádíme pouze některé přístupy, nejedná se tak o kompletní výčet. Nejdříve v 2.3.1 probereme teoretické koncepty, které se používají pro hodnocení algoritmů. Další odstavce se zaměří na konkrétní algoritmy. Ty lze rozdělit dle způsobu řešení na tři podkategorie. První kategorií jsou algoritmy, jež řeší rozklad spojitého C_{free} na menší oblasti (Cell decomposition - 2.3.2). Další kategorie naopak využívá poznatky z kalkulu a problem formulují jako hledání minima spojité funkce nad C-space (Artifical Potential Field - 2.3.2). Poslední kategorie je založena na *vzorkování*, jež lze dále rozdělit na deterministické (State & Control lattice - 2.3.2) a nedeterministické (Probabilistic Road Map - 2.3.2, Rapidly exploring Random Tree - 2.3.2). Text se bude především věnovat posledně zmíněné kategorii, neboť i prezentovaný algoritmus do ní patří. Ten však bude probrán v samostatné kapitole 3.1.

O úplnosti algoritmů. Dříve než postupně zmíníme nejznámější algoritmy z oblasti plánování trajektorie, přiblížíme pojem *úplnosti* algoritmu a jeho slabší notace, jež nám pomohou kvalitativně zhodnotit a srovnat jednotlivé přístupy.

Definice 2.1 (Úplnost algoritmu)**.** Algoritmus je úplný, zaručuje-li, že pro libovolný (přípustný) vstup vrátí vždy správné řešení v konečném čase. Neexistuje-li správné řešení pro přípustný vstup, algoritmus v konečeném čase tuto informaci oznámí.

Definice 2.2 (Úplnost algoritmu v rozlišení)**.** Algoritmus je úplný v rozlišení, zaručuje-li, že pro libovolný (přípustný) vstup vrátí vždy správné řešení v konečném čase, pakliže takovéto řešení existuje. Neexistuje-li řešení, algoritmus může běžet libovolně dlouho.

Definice 2.3 (Pravděpodobnostní úplnost algoritmu)**.** Algoritmus je pravděpodobnostně úplný, pakliže s dostatečným počtem vzorků se pravděpodobnost nalezení správného řešení blíží k 1.

Již v 2.2 jsme zmínili, že plánovací algoritmy lze členit na *přesné* a *vzorkovací*. Všechny *přesné* algoritmy mají vlastnost *úplnosti*. U *vzorkovacích* algortimů se budeme muset smířit se slabší notací.
Vzorkuje-li algoritmus libovolně *hustě*, je *úplný v rozlišení*. Mnoho vzorkovacích algoritmů volí sve vzorky *náhodně* s daným rozdělením. Pakliže je dané rozdělení vhodné (rozumějme algoritmus vzorkuje hustě s pravděpodobností 1),

je algoritmu *pravděpodobnostně úplný*. Ačkoliv jsou vzorkovací algoritmy *neúplné*, obvykle dosahují v běžné praxi lepších výsledků a zvláště v posledních letech jsou hojně využívany [16].

Poznámka 2.4 (O hustotě). Pojem *husté* vzorkování zde není definováno a lze jej chápat intuitivně, formálně však vychzí z definice *hůsté* množiny v topologii. Notorickým příkladem husté množiny jsou racionální čísla $\mathbb{Q}$ v $\mathbb{R}$. Ve stejném smyslu se mohou *vzorky* algoritmu blížit k libovolné *konfiguraci* za předpokladu, že se počet iterací blíží k nekonečnu.

2.3.2 Algoritmy

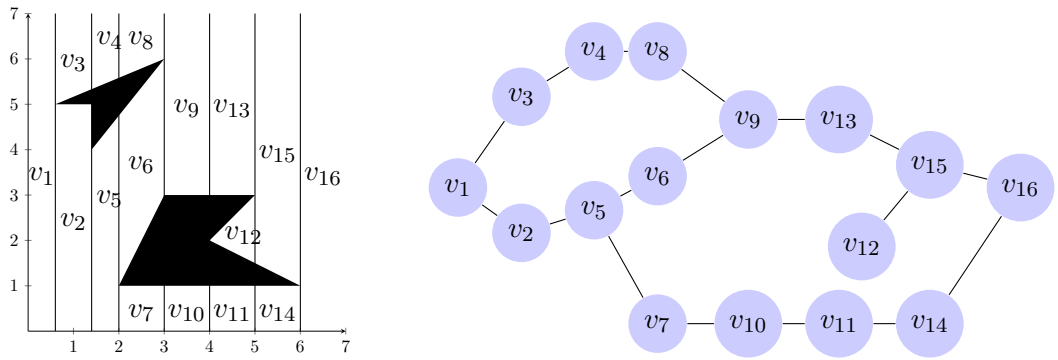
Cell Decomposition. Koncept dekompozice[4] je nadčasový a moderní člověk ho používá na denní bázi. Jedná se tak o obecnou myšlenku, která lze uplatnit i na problém plánování trajektorie. V takovém případě dekomponujeme C_{free}. Takovýto rozklad nám umožní diskretizaci spojitého problému a následné využití teorie grafu. v závislosti na způsobu a kvalitě rozkladu C_{free} rozlišujeme několik druhů dekompozic. Všechny však mají stejný princip:

- Rozložení C_{free} na soubor jednotlivých, *nepřekrývajících se, jednoduchých* oblastí $R = \{C_1, C_2, C_3, C_4, \ldots\}$.
- Vytvoření grafu $G(V = R, E)$ zachovavající topologii souboru (hrany reprezentují „sousedství“ oblastí).
- Nalezení cesty v G spojující oblast s počateční konfigurací s oblastí s konfigurací koncovou.

Jednoduchými oblastmi rozumím v tomto případě takové, jež splňují následné vlastnosti[s. 253 36]:

1. Uvnitř oblasti je pohyb robota mezi libovolnými dvěma body dostatečně triviální a není náročné ho vypočítat.
2. *Sousedství* oblastí lze při tvorbě grafu jednoduše rozhodnout.
3. Pro q_{init} a q_{final} by nemělo být složité určit, které oblasti je obsahují.

[4] Neboli rozkladu složitých objektů na objekty jednodušší (jejichž složení opět tvoří původní objekt).



Obrázek 2.4: Lichoběžníková dekompozice

Z pohledu kvality rozkladu lze rozlišit mezi rozklady *úplnými* a *přibližnými*. *Úplné* rozklady splňují vlastnost[5]

$$C_{free} = \bigcup_i C_i$$

a alogritmy, jež je používají jsou *úplné*. Nalezní takových rozložení však nemusí být jednoduché, a proto se používají i rozklady *přibližné*.[32]
Pro názornost uvedeme *lichoběžníkový* rozklad, jež se řadí mezi *úplné* rozklady a *quadtree* rozklad, který je *přibližným* rozkladem. Pro hlubší seznámení se s *úplnými* rozklady doporučujeme [kap. 6 36] či [kap. 5 32], pro rozklady *přibližné* lze využít [kap. 6 32].

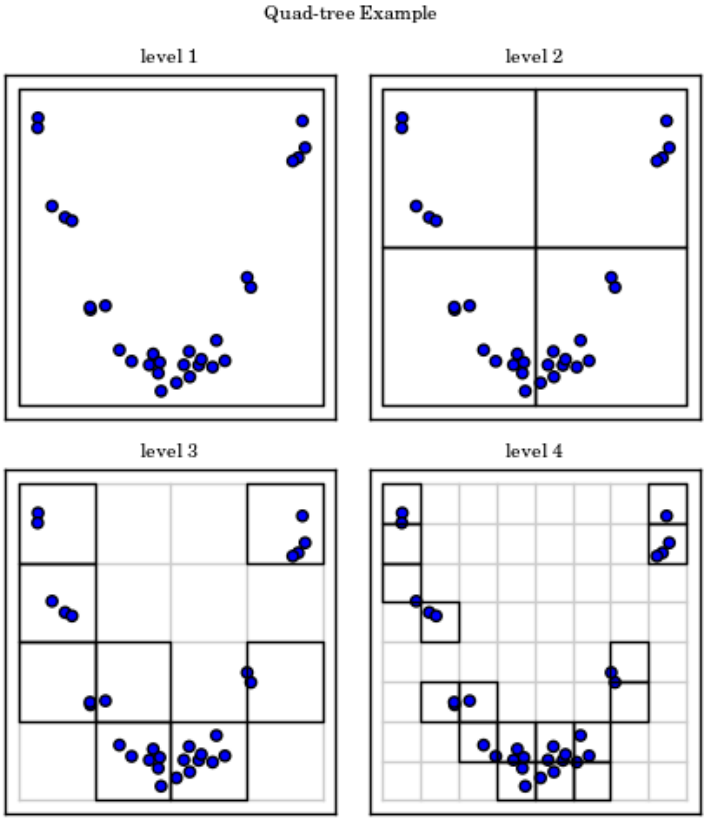
Příklad 2.5 (Lichoběžníkový rozklad). Pro jednoduchost předpokládejme, že holonomní robot $\mathscr{A}$ je reprezentován jako bod v 2D prostoru:

$$C = \mathscr{W} = \mathbb{R}^2$$
$$C_{obs} = \bigcup_i^n H_i$$
$$H_i \bigcap H_j = \emptyset \forall\, i, j \leq n, i \neq j,$$

kde H_i reprezentuje mnohoúhelník (viz 2.1). Lichoběžníkový rozklad je přímočarý. Pro každý vrchol v každého mnohostěnu H_i reprezentující C_{obs}, vytvoříme vertikální[6] přímku $x = x_v$. Tyto přímky rozdělí C_{free} do konvexních oblastí R, jež jsou reprezentovány vrcholy grafu G s hranami odpovídajícími přechody mezi nimi. Nad takovýmto grafem lze již použít algoritmy na hledání minimální cesty (viz 2.4).

Příklad 2.6 (quadtree rozklad). Pro jednoduchost předpokládejme, že $C = \mathscr{W} = \mathbb{R}^2$. Princip quadtree rozkladu je opakované rovnoměrné rozdělování C-space vždy na 4 části (viz 2.5, 2.6). v každé další iteraci se rozděluje část, jež vznikla v předchozí. Toto dělení probíhá, dokud se nedovrší daného rozlišení ϵ. Celkový počet iterací značme n.

[5] Nebo $\mathrm{cl}(C_{free}) = \bigcup_i C_i$
[6] Lze zvolit i horizontální dělení.



Obrázek 2.5: Quad tree - Rovnoměrné dělení Zdroj: [18]

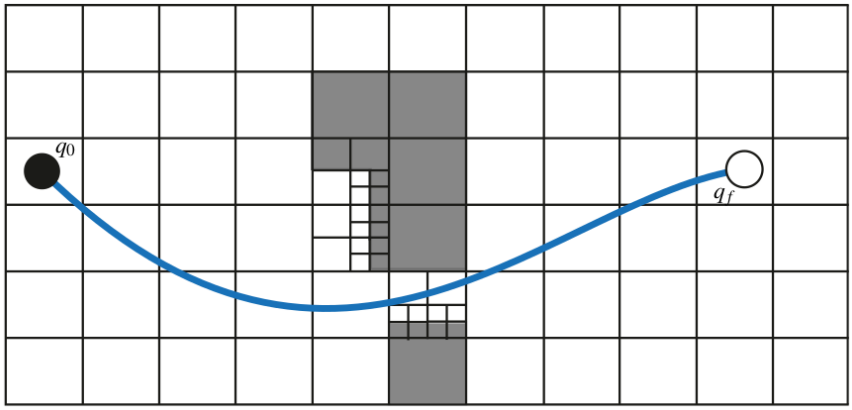
Buď lze dělit všechny nově vzniklé části, nebo pouze ty, jež obsahují hranici překážek

$$C_{i,j} \cap C_{obs} \neq \emptyset \wedge C_{i,j} \cap C_{free} \neq \emptyset, \quad i \in \{1, .., n\}, j \in \{1, 2, 3, 4\}.$$

První způsob vede na reprezentaci pomocí bitmap 2.1. Druhý naopak zpusobuje, že jednotlivé buňky jsou v oblasti hranice překážek velmi jemně dělené, naopak ve volném prostoru jsou „velké“.Ačkoliv je tato dekompozice pouze *přibližná*, aproximuje C_{free} libovolně přesně.

Probabilistic Roadmaps (PRM). V roce 1996 byl představen nový algoritmus pro plánování trajektorie Kavarakiovou et al.[26] pro vysokodimenzionální C-space.[7]. Tento algoritmus spadá do *vzorkovacích* algoritmů a je založen na *náhodném* vzorkování konfigurace v C-space. Tento algoritmus je *pravděpodobnostně úplný* (za předpokladu vhodného rozdělení)[26] s exponenicálním poklesem pravděpodobnosti neúspěchu za předpokladu růstu vzorků [25]. Algoritmus má dvě fáze: učící fázi (offline) a řešící fázi (online). Nejdříve se

[7]Připomněme i českou stopu v podobě Petra Švestky, jenž je spoluautorem.



Obrázek 2.6: Quad tree - Cílené dělení Zdroj: [53]

v účící fázi seznamuje s okolním prostředím pomocí náhodného vzorkování konfiguračního prostoru. Pakliže je zvolená konfigurace bezkolizní, je propojena s již existující mapou pomocí *lokálního plánovače*. Vzniká tak graf, jehož uzly jsou bezkolizní konfigurace. Hrany pak klíčují schopnost robota přejít z jedné konfigurace na druhou (toto je zjištěno lokálním plánovačem). V druhé fázi je zadán počáteční a koncový bod. Ty jsou přidány do grafu a pomocí grafových technik je nalezena optimální cesta. Algoritmus je představen pomocí pseudokódu 1.
Algoritmus ke svému fungování potřebuje jednak vhodnou (pseudo)metriku d

$$d : C \times C \mapsto \mathbb{R}_{\geq 0} \tag{2.9}$$

měřící C-space a již zmíněný lokální plánovač l, jež rozhoduje o proveditelnosti přechodu mezi konfiguracemi q_1, q_2:

$$l : C \times C \mapsto \{0, 1\} \tag{2.10}$$

$$l(q_1, q_2) = \begin{cases} 1 & \text{trajektorie je kolizní} \\ 0 & \text{trajektorie je bez kolize,} \end{cases} \tag{2.11}$$

kde zvolená trajektorie je vlastností lokálního plánovače l.

Příklad 2.7 (Metriky). Mezi typicky používanými metrikami jsou p-normy, zvlaště

$$L_1 \text{ norma} : \sum_i |x_i - y_i|$$

$$L_2 \text{ norma} : \sqrt{(\sum_i (x_i - y_i)^2)}\,.$$

Pro robotická ramena s rotačními klouby je pak vhohodná metrika

$$d(\theta_1, \theta_2) = \min(|\theta_1 - \theta_2|, 2\pi - |\theta_1 - \theta_2|)$$

Poznámka 2.8. Lokální plánovač typicky volí za spojnici bodů q_A a q_B úsečku, kterou z rozlišením ϵ vzorkuje a kontroluje body $q_A = q_1, q_2, \ldots, q_{n-1}, q_n = q_B$

oproti kolizi pomcí příznakové funkce (viz [40]). Tato volba je vhodná pro plánování trajektorie holonomníních robotů (robotická ramena a manipulátory).

Algorithm 1 Probabilistic Road Map algorithm

function PHASE ONE(Maxiter,$k, M, G(V, E, \omega)$)
  $n \leftarrow 0$
  for $i \leq$ Maxiter **do**
    Náhodně vygeneruj konfiguraci q_{new}
    if $q_{new} \in C_{free}$ **then**
      Najdi k nejbližších bodů q_j v grafu $G(V, E)$ pomocí metriky d.
      for $j \leq k$ **do**
        if $l(q_{new}, q_j) =$ True **then**
          Přidej vrchol q_j a hranu (q_{new}, q_j) do grafu $G(V, E)$
          Oceň hranu $\omega : (q_{new}, q_j) \rightarrow \mathbb{R}$
          $n \leftarrow n + 1$
          if $n \geq M$ **then break**
          end if
        end if
      end for
    end if
  end for
  return $G(V, E, \omega)$
end function
function PHASE TWO($M, q_{init}, q_{final}, G(V, E, \omega)$)
  Pomocí d a l najdi nejbližších M bodů q_j bodu q_{init} v grafu $G(V, E)$ a přidej ho do $G(V, E)$.
  if $q_{init} \notin V$ **then**
    return Failure
  end if
  Pomocí d a l najdi nejbližších M bodů q_j bodu q_{final} v $G(V, E)$ a přidej ho do $G(V, E)$.
  if $q_{final} \notin V$ **then**
    return Failure
  end if
  return Nejlevnější cesta P v grafu $G(V, E) : \omega(P) \leq \omega(P^*) \, \forall P^* \in G$
end function

Velmi důležitým aspektem PRM je náhodné rozdělení, jímž se řídí selekce *vzorků*. Ačkoliv je rovnoměrné rozdělení obecně vhodné pro plánování trajektorie, na jeho úskalí lze narazit zvláště v případech úzkých průjezdů, jež jsou na konfiguraci robota náročné, ale splnitelné. Jako celkově vhodné rozdělení pro širokou škálu překážek se jeví rozdělení založené na Haltonových posloupnostech.[5] Upozorněme však, že Haltonova posloupnost je deterministická. Tento algoritmus je vhodný do statického prostředí (typicky průmyslová výroba), neboť mapu lze vygenerovat pouze jednou a následný počet dotazů (q_{init}, q_{final}) v druhé fázi může být libovolně mnoho. Nevýhodou však je, že

výsledná trajektorie není zpravidla idealní. To je případně řešeno *zahlazováním* trajektorie, typicky v závislosti na hodnotícím kriteriu a konstrukci konkrétního robota Přehled způsobu zahlazování trajektorie je zveřejněn v [54]. Případně lze použít optimalizovaný alogritmus PRM*, jenž byl představen Karamanem a Frazzolim v článku [24]. Vhodné srovnání různých parametrů PRM lze nalézt v [15]. Zdůrazněme, že ačkoliv je tato metoda na první pohled nevhodná pro neholonomní systémy (viz 2.8), byly představeny různé lokální plánovače PRM, zvláště pro systémy podobné automobilům ([59],[20]).

Rapidly exploring random trees (RRT). Dalším *vzorkovacím* algoritmem založeným na pravděpodobnostním přístupu a náhodě je RRT [34] a jeho vylepšená verze RRT-connect [30]. Oba tyto algoritmy jsou *pravěděpodobnostně úplné* [34] s exponenciálním poklesem pravděpodobnosti selhání [14] a veskrze velmi podobné PRM. Ke svému běhu opět potřebují *(pseudo)metriku d* (2.9) a *lokální plánovač l* (2.10). Algoritmus RRT má oproti PRM pouze jednu fázi. Nejdříve zakoření strom T v počáteční konfiguraci q_{init} a pak opakovaně volí náhodné konfigurace $q_{new} \in C$-space, které se pak pomocí d a l snaží přidat k nejbližšíším listu stromu $q_{leaf} \in T$[8].Výsledkem je pak cesta P ve stromu T, která vede k listu, který je „blízko“ q_{final}. Stručný pseudokód je nastíněn v 2.
RRT-connect je nadstavbou, která využívá možnosti zakořenit strom T' i v q_{final}. Postupně tak střídavě rostou oba stromy T a T' dokud jejich listy „nejsou dostatečně blízko“. V tom případě se pak propojí a vzniklá cesta mezi q_{init} a q_{final} je výstupem RRT-connect.
Povšimněme si, že změníme-li dvojci (q_{init}, q_{final}), celý strom (stromy) se můsí utvářet znovu a algoritmus běží od začátku. To zhoršuje použitelnost RRT ve statickém prostředí (narozdíl od PRM). Tato nevýhoda je však v dynamickém prostředí potlačena změnou okolního prostředí. Obdobně jako PRM trpí na „neidealitu“ výsledné cesty. Tento problem byl vyřešen Karamanem a Frazzolim v článku [24], který představil optimalizované algoritmy PRM* a RRT*. Jak RRT, tak RRT* se hojně uplatnily pro plánování trajektorie neholonomních sytémů ([14, 35, 19, 39]) a těší se velké oblibě i dnes. V [45] je představena *real-time* verze RT-RTT*, jež je schopná online přeplánovávat a využívá již vytvořené stromy při pohybu robota.

Artifical Potential Fields (APF). Další z přístupů k řešení plánování trajektorie je metoda potenciálních polí, jež byla poprvé v této oblasti představena O. Khatibem [28]. Metoda využívá myšlenku potenciálních polí, jež je dobře známá z teorie fyziky (gravitace, či elektrické pole) a lze shrnout následovně:

[8] Zdůrazněme, že struktura stromu vzniká způsobem přidávání nových vrcholů q_{new}, konkrétně tím, že přidáváme vždy pouze jednu hranu (q_{new}, q^*_{leaf}), $q^*_{leaf} = \arg\min_{q \in T} d(q, q_{new})$

Algorithm 2 Rapidly Exploring Random Tree algorithm

function GROW TREE(Maxiter,δ, q_{init})
  Inicializuj strom $T(V,E)$, kde $V = \{q_{init}\}$, $E = \{\}$
  for $i \leq$ Maxiter **do**
    Náhodně vygeneruj konfiguraci q_{new}
    if $q_{new} \in C_{free}$ **then**
      Najdi nejbližší list q_{leaf} ve stromu $T(V,E)$ pomocí metriky d.
      if $d(q_{new}, q_{leaf}) > \delta$ **then**
        if $\exists\, q \in C_{free} : q \in \tau_{(q_{new},q_{leaf})} \wedge d(q, q_{leaf}) \leq \delta$ **then**
          $q_{new} = q$ ▷ $\tau_{(x,y)}$ značí trajektorii mezi x a y
        else
          continue
        end if
      end if
      if $l(q_{new}, q_{leaf}) =$ True **then**
        $V = v \cup \{q_{leaf}\}$
        $E = E \cup \{(q_{new}, q_j)\}$
      end if
    end if
  end for
end function
function EVALUTE($\delta, q_{final}, T(V,E)$)
  if $\exists q_{leaf} \in V : d(q_{final}, q_{leaf}) \leq \delta$ **then**
    $q_{final} = q_{leaf}$
    return Cesta $P : (q_{init}, \ldots, q_{final})$ v grafu $T(V,E)$
    ▷ Graf T je strom s kořenem q_{init}, tedy $\exists! P$
  else
    return Failure
  end if
end function

> „*Robot se pohybuje v poli sil. Žádaná konfigurace robota je pólem přitažlivé síly a stěny překážek naopak působí na robota odpudivou silou.* “[28][9]

Metoda ke svému fungovaní potřebuje funkci $p : C \mapsto \mathbb{R}_{\geq 0}$, jež zobecňuje vzdálenost robota od překážky v C-space. Vhodné funkce k takovéto reprezentaci lze najít v [40].Této metodě se občas přisuzuje přívlatek „Sensor-based“ právě kvůli nezastupitelné roli senzorů přiblížení, jež v praktické implementaci reprezentují p. Potenciální pole f pak lze vyjádřit složením kladné a záporné síly $f = f_a + f_r$, kde f_a reprezentuje přitažlivou sílu a f_r sílu zápornou[10](viz 2.8)

[9] Překlad autora.
[10] Indexy r a s jsou odvozeny z anglických slov „attractive“ a „repulsive“.

Příklad 2.9 (Kladná síla). Pro jednoduchost předpokládejme, že holonomní robot $\mathscr{A}$ je reprezentován jako bod v 2D prostoru, a tedy $C = \mathscr{W} = \mathbb{R}^2$. Pak kladnou sílu f_a lze konstruovat následovně

$$f_a : \mathbb{R}^2 \mapsto \mathbb{R}$$
$$f_a(\boldsymbol{x}) = \eta \, ||\boldsymbol{x} - \boldsymbol{x_f}||^2,$$

kde $\boldsymbol{x}$ reprezentuje aktuální pozici robota a $\boldsymbol{x_f}$ reprezentuje finální pozici robota. Parametr $\eta \geq 0$ je škálovací faktor. f_a je tedy pozitivně definitní kvadratická norma s minimem v x_f, viz (2.7a).

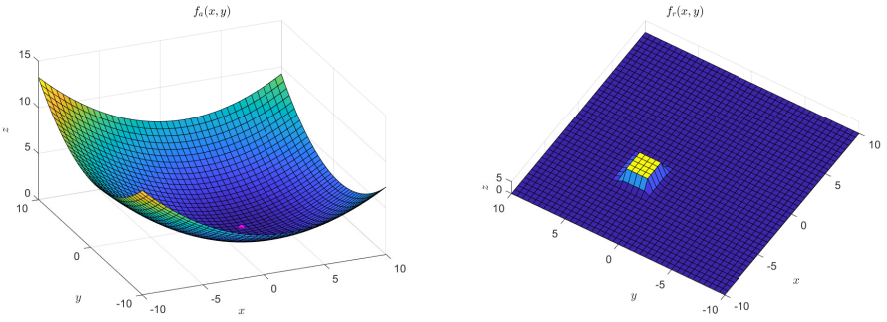
Příklad 2.10 (Záporná síla). Pro jednoduchost předpokládejme, že holonomní robot $\mathscr{A}$ je reprezentován jako bod v 2D prostoru, a tedy $C = \mathscr{W} = \mathbb{R}^2$. Zapornou sílu f_r konstrujeme následovně

$$f_r : \mathbb{R}^2 \mapsto \mathbb{R}$$
$$f_r(\boldsymbol{x}) = \begin{cases} \rho(\frac{1}{p(\boldsymbol{x})} - \frac{1}{d_0})^2 & p(\boldsymbol{x}) \leq d_0 \\ 0 & p(\boldsymbol{x}) \geq d_0 \end{cases},$$

kde $\rho \geq 0$ je škálovací faktor. Pro větší názornost viz (2.7b). Při konstrukci f_r jsme využili vlastnosti kladných reálných čísel $\mathbb{R}_{>0}$:

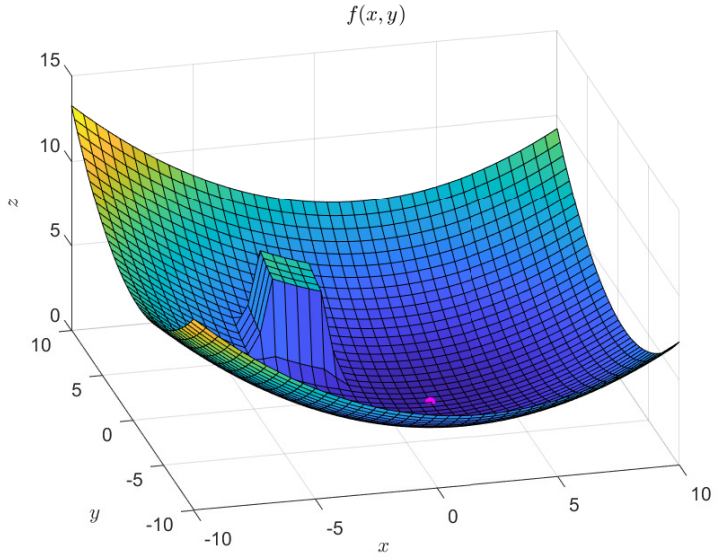
$$x \leq y \implies \frac{1}{x} \geq \frac{1}{y} \quad \forall x, y \in \mathbb{R}_{>0}$$

K nalezení $\boldsymbol{x_f}$ a trajektorie pak lze využít gradientní metodu, či libovolnou další iterační metodu na hledání volného lokálního minima na $f(\boldsymbol{x})$. Zdůrazněme několik úskalí prezentovaného přístupu. Jedním z nich je problém při aplikaci metody na neholonomní roboty, neboť směr sestupu nemusí odpovídat proveditelné trajektorii robota. Toto lze však vyřešit konečnou množinou přípustných pohybů robota a vybrání takového pohybu, jež minimalizuje potenciální pole f, či nalezením cesty pomocí APF a pak zpětnovazebním řízením podel dané trajektorie [31]. Přeformulujeme-li takovýto přístup, jedná se o *vzorkovací deterministický* algoritmus, jež je obohacen o heuristiku, která je reprezentována právě potenciálním polem f. Mnohem významější překážkou je však uvíznutí algoritmu v lokálním minimu f. Právě tento aspekt je silným omezením dané metody, neboť je důvodem *neúplnosti* algoritmu. Nejčastějším řešením je kombinace s dalším plánovacím algoritmem, který je schopen robota vyvést z případného lokálního minima. Navíc, pokud je cesta nalezena, není nikterak zaručena její optimalita. Lze však upravit f tak aby zohledňovala příslušná kritéria [13]. Naopak pozitivem tohoto přístupu je vysoká rychlost výpočtu a možnost snížení dimenze prohledávaného prostoru oproti konfiguračnímu C-space pomocí níže uvedné generalizace.
Předpokládejme, že robot má vysokou dimenzi v C-space (typicky složité průmyslové stroje). Místo exaktního vyjádření C_{obs} lze na robotovi zvolit n



(a) : Kladná síla f_a **(b) :** Záporná síla f_r

Obrázek 2.7: Složky potenciálního pole



Obrázek 2.8: Výsledné potenciální pole f

bodů a pro každý bod vypočítat nad konfiguračním prostorem C zápornou sílu $f_{ri} : C \mapsto \mathbb{R}$. Každá síla f_{ri} vychází z funkce $p_i : C \mapsto \mathbb{R}_{\geq 0}$, jež zobecňuje vzdálenost i-tého bodu od překážek. Celkové pole $f = f_a + \sum_i f_{ri}$ a jeho gradient $\nabla_q f$ pak popisuje výslednou trajektorii.

State & Control lattice approach. Tento přístup byl představen Pivtoraikem v roce 2005 [51] a přináší plánování pro *neholonomní diferenciální* systém přímo v C-space pomocí konečného počtu „speciálních“ pohybů. „Speciálními“ pohyby myslíme takové, jež odpovídají difereneciálním omezením systému, budeme je nazývat *proveditelnými*[11]. Doposud zmíněné algoritmy popisovaly kinematiku robota pomocí algebraicko-goniometrických

[11] Z anglického *feasible*.

Algorithm 3 Control based sampling

Diskretizuj prostor vstupu U a vytvoř soubor řídících primitv $u = (u_1, u_2, u_3, \ldots)$
function GROW TREE($u, \epsilon, q_{final}, q_{init}$)
  Inicializuj strom $T(V, E)$, kde $V = \{q_{init}\}$, $E = \{\}$
  Inicializuj seznam Frontier = $\{q_{start}\}$
  $q_{cur} = q_{start}$
  while $|q_{cur} - q_{final}| < \epsilon$ **do**
    q_{cur} = Frontier.pop() ▷ Seřazení lze ovlivnit heuristikou
    for $u_i \in u$ **do**
      $q_{new} = q_{cur} + \Delta q$
      if $\tau_{(q_{new}, q_{cur})}$ je „volná“ **then**
        $V = v \cup \{q_{new}\}$ ▷ $\tau_{(x,y)}$ značí trajektorii mezi x a y
        $E = E \cup \{(q_{new}, q_{cur})\}$
        Frontier.append(q_{new})
      end if
    end for
  end while
  return Cesta $P : (q_{init}, \ldots, q_{final})$ v grafu $T(V, E)$
    ▷ Graf T je strom s kořenem q_{init}, tedy $\exists! P$
end function

rovnic a tiše předpokládaly, že je robot holonomní systém a každý stav lze řídit nezávisle na jiných. Skutečně běžné robotické manipulátory jsou konstruovány tak, aby tuto vlastnost splňovaly (Kartézský robot, SCARA robot, Stewartova plošina). V [22] se lze s danou problematikou seznámit blíže. Důležitým aspektem je „snadné“ řešení inverzní kinematické úlohy. Zatímco dosud stačilo najít inverzi algebraicko-goniometrických rovnic, v případě diferenciálních systémů je potřeba najít inverzi k diferenciálním rovnicím.[27] Automobil (obecně mobilní systémy) patří do kategorie neholonomních robotů, v oddílu (2.4) jsou představeny matematické modely, jež slouží k jeho reprezentaci. Matematický model je obecně reprezentován soustavou diferenciálních a algebraických rovnic:

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) \\ \boldsymbol{y} &= \boldsymbol{h}(\boldsymbol{x}), \end{aligned} \tag{2.12}$$

kde $\boldsymbol{x}$ označuje stav modelu a $\boldsymbol{y}$ značí výstup systému. Vstup systému značíme symbolem $\boldsymbol{u}$. Poznamenejme že zde používaný pojem konfigurační prostor se v teorii řízení označuje stavový prostor a pojmy stav a konfigurace tak splývají.

Control lattice. Představme nejdříve přístup pomocí Control lattice approach. Jeho významnou výhodou je, že lze použít na libovolný difereneciální systém. Naopak nevýhodou je, že není obecně schopen řešit plánování pro

libovolnou dvojici stavů (q_{init}, q_{final}). Algoritmus nejdříve zvolí ze stromu T konfiguraci q. Danou konfiguraci rozvine o řidící vstup $\hat{\boldsymbol{u}}$:

$$\Delta q = \Delta \boldsymbol{x} = \int_{t}^{t+\Delta t} \boldsymbol{f}(\boldsymbol{x}, \hat{\boldsymbol{u}}) dt \tag{2.13}$$

Pokud je cesta podel řídící trajektorie bezkolizní, je nová konfigurace

$$q_{new} = q + \Delta q$$

přidána do stromu T. Přiblížíli se řešení q_{new} dostatečně blízko q_{final}, algoritmus ukončí prohledávání a vrátí cestu grafem T. Pseudokod je implementován v (3).
Významným atributem je volba vstupu $\hat{\boldsymbol{u}}$. Při Control lattice approach se dle daného vzoru (např. rovnoměrně) diskretizuje prostor vstupů U, a vznikne tak konečný soubor řídících primitv.
Díky povaze volby vzniká struktura *stromu*, navíc lze využít heuristiky k zrychlení prohledávání. V těchto aspektech je představovaný algoritmus podobný RRT-(2.3.2).

State lattice. Oproti Control lattice approach se State lattice aprroach podrobně zabývá modelem systému a jeho diferenciální omezení. Důsledkem je, že se každý typ systému studuje odděleně, a vznikají tak specifická řešení. Automobilová oblast byla v tomto ohledu zkoumána v posledních desetiletí a existují řešení pro různá kritéria a různé matematické modely (aut či kolových systémů) [36, kap. 15],[11, 55, 7, 4, 51, 48]. Máme-li reprezentaci *optimálních*[12] pohybů, lze pro každou dvojci konfigurace p, q rozhodnout, zda existuje (optimální) trajektorie $\tau_{(p,q)}$, pro kterou jsme schopni přímo vyjádřit vstup diferencialního systému u. Případně je trajektorie *proveditelná* a stačí zapojit zpětnovazební řízení [36, kap. 8,14],[23] [13]. Zvolíme-li mřížku v C-space s daným rozlišením $\Delta q \in \mathbb{R}^n$:

$$\Delta q = \begin{pmatrix} \Delta q_1 \\ \Delta q_2 \\ \vdots \\ \Delta q_n \end{pmatrix},$$

lze předpočítat soubor základních pohybů. Vyhodou tohoto přístupu je jednak schopnost libovolně přesně aproximovat prostor C-space, narozdíl od Control lattice approach, kde tranzitní konfigurace nelze předem dobře odhadnout. Lze tak zaručit, že q_{final} je v této mřížce. Postup algoritmu je dále totožný s Control lattice approach.

[12] dle zvoleného kritéria

[13] Při skutečné aplikaci řídících a plánovacích systému na palubu robota je použití zpětné vazby více než vhodné, neboť je schopna vyřešit nejistoty a vlastnosti robota, jež nejsou modelovány (síla motorů, tuhost karosérie, nekvalitní součástky z výroby a další nejistoty).

Jsme-li schopni konstruovat optimální trajektorie $\tau_{(p,q)}$ pro libovolné konfigurace p a q, zdálo by se, že lze využít i jiné metody plánování trajektorie (PRM, RRT). Uskalím je v tomto případě časová náročnost tvorby $\tau_{(p,q)}$, zvláště pro komplikovanější systémy s vyšší dimenzí C-space, jež může znemožnit použití algoritmů, jež počítá trajektorii v on-line fázi. Naopak soubor základních pohybů lze určit dopředu v off-line fázi a State lattice approach tak v on-line fázi tvoří pouze graf dle aktuálních překážek v okolí.

Tvořením mřížek se oba algoritmy řadí mezi *vzorkovací*. Čím hustější je dělení v mřížce, tím přesněji algoritmy popisují skutečné možnosti robota, avšak prodlužuje se výpočetní doba. Tento aspekt je pro *deterministické vzorkovací* algoritmy typický a je vždy nutné zvážit potřebnou přesnost a časovou náročnost.

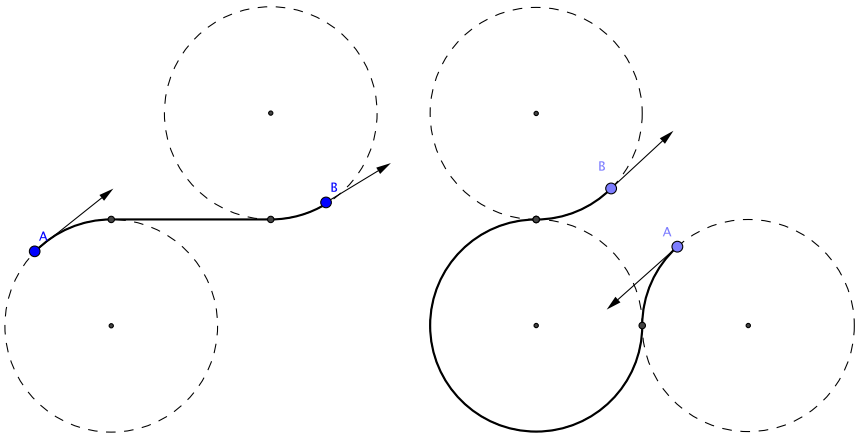
2.4 Modely automobilu

V této části stručně probereme matematické modely pro automobily a jim podobné robotické platformy.

Dubins' car. Nejprimitivnějším matematickým modelem, jímž lze aproximovat kinematiku automobilu, je Dubins' car. Systém je popsán diferenciálními rovnicemi

$$\begin{aligned} \dot{x} &= V\cos(\theta) \\ \dot{y} &= V\sin(\theta) \\ \dot{\theta} &= u, \end{aligned} \tag{2.14}$$

kde $\Big(x, y\Big)^T$ je pozice automobilu, θ je azimut automobilu (vůči ose x). Vstup u odpovídá „míře zatáčení“ a je omezen minimálním poloměrem zatáčení $|u| \leq \frac{V}{r_{min}}$. Automobil se může pohybovat pouze konstantní rychlostí V a pouze dopředu. Lester Dubins ve svém článku [11] dokázal, že *nejkratší* trajektorie, po kterých se tyto autmobily mohou pohybovat, jsou složeny z 3 základních pohybů: „zatočit doprava“-R, „zatočit doleva“-L, „jet rovně“-S a jsou jimi tyto sekvence: 'RSR', 'RSL', 'LSR', 'LSL', 'RLR', 'LRL', kde rádius zatočení $r = r_{min}$. Tyto trajektorie jsou též známy jako Dubinsovy (2.9).



(a) : RSL Zdroj: [1] **(b) :** RLR Zdroj: [2]

Obrázek 2.9: Dubinsovy pohyby

Reeds-Sheep' car. Obohatíme-li Dubins' car o možnost jízdy dozadu, získáme Reeds-Sheep' car. Systém je popsán rovnicemi

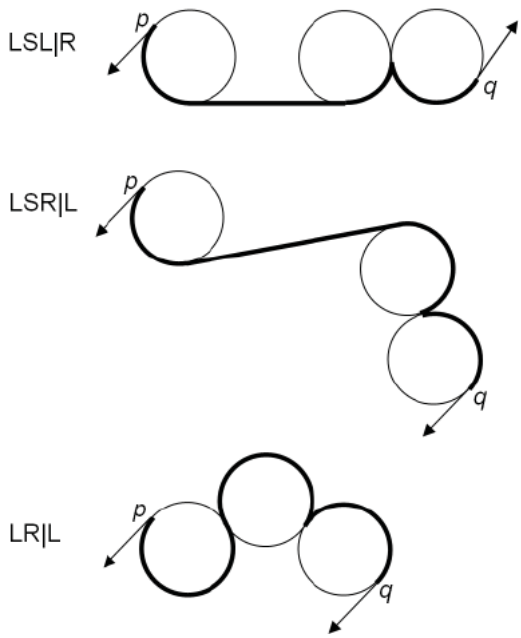
$$\begin{aligned} \dot{x} &= V\cos(\theta) \\ \dot{y} &= V\sin(\theta) \\ \dot{\theta} &= u_1 \cdot u_2, \end{aligned} \tag{2.15}$$

kde stavy odpovídají stavům z (2.14). Vstup u_2 odpovídá vstupu u v Dubins' car (2.14). Vsutp $u_1 \in \{-1, 1\}$ a odpovídá možnosti jízdě dozadu [kap. 15 36]. Reeds a Sheep ukázali v článku [55], že existuje pouze 48 různých sekvencí řízení pro konstrukci *nejkratší* trajektorie $\tau_{(p,q)}$ spojující libovolné dvě konfigurace p a q. Základní slovník je obohacen o symbol '|', jež reprezentuje „změnu směru" a výsledné pohyby jsou podrobně rozebrny v [s. 887 36]. Sussmann a Tang v [58] dokázali, že lze zredukovat celkový počet sekvencí z 48 na 46.

Differential Drive. Ne všechny systémy mají strukturu a pohon podobné čtyřkolovým vozidlům. Existují i platformy, jež mají každé kolo poháněno samostatně, jsou jimi například robotický vysavač, či Segway. Systémy lze popsat rovnicemi

$$\begin{aligned} \dot{x} &= \frac{r_w}{2}(u_l + u_r)\cos(\theta) \\ \dot{y} &= \frac{r_w}{2}(u_l + u_r)\sin(\theta) \\ \dot{\theta} &= \frac{r_w}{d}(u_r - u_l), \end{aligned} \tag{2.16}$$

kde $\Big(x, y\Big)^T$ odpovídá středu robota, θ je azimut robota. Parametry robota jsou poloměr kol r_w a rozchod kol d. Vstupy jsou úhlové rychlosti $u_l = \omega_w^l$ a



Obrázek 2.10: Ukázka pohybů pro Reeds-Sheep car Zdroj: [6]

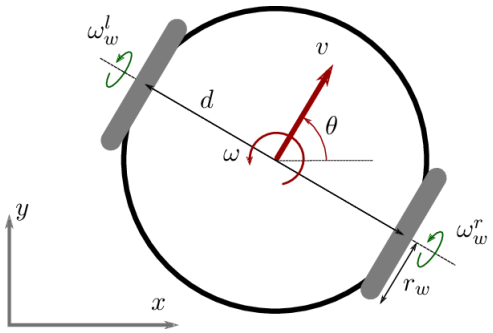
Pohyb	u_r	u_l
„dopředu“	ω_{max}	ω_{max}
„dozadu“	$-\omega_{max}$	$-\omega_{max}$
„zatáčet doleva“	$-\omega_{max}$	ω_{max}
„zatáčet doprava“	ω_{max}	$-\omega_{max}$

Tabulka 2.1: Základní pohyby pro Balkcom-Masonovy křivky

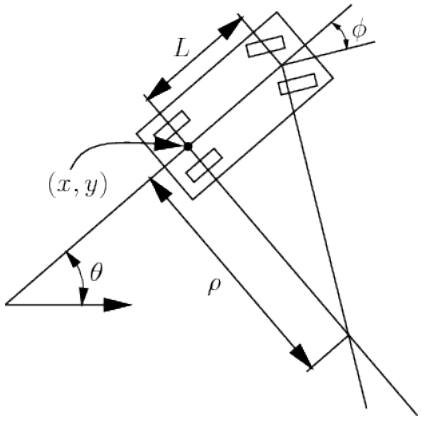
$u_r = \omega_w^l$ na levém a pravém kole (viz 2.11). Balkcom a Mason v [4] ukázali, že existují časově optimální[14] trajektorie, které lze složit z 4 základních primitiv. Pro jednoduchost předpokládejme, že $u_i \in [-\omega_{max}, \omega_{max}]$, $i \in r, l$. Základní primitivy jsou uvedeny v tabulce (2.1). Navíc každá optimální trajektorie je složena z nejvýše 5 základních primitivů. Mnohé pohyby jsou však symetrické a minimální počet variací pro řízení lze významně snížit [4].

Bicycle model. Jak Dubins' car tak Reeds-Sheep' car modely reprezentovaly automobil jako jednoosé vozidlo, a zanedbávaly tak část dynamiky plynoucí z dvouosé struktury, jež je pro automobil typická. Bicycle model tento aspekt zahrnuje, a používá se tak jako výchozí model pro simulaci běžného automobilu.

[14] Optimalizace nelze formulovat jako minimalizace dráhy, neboť robot může rotovat na místě.



Obrázek 2.11: Differential Drive Zdroj: [60]



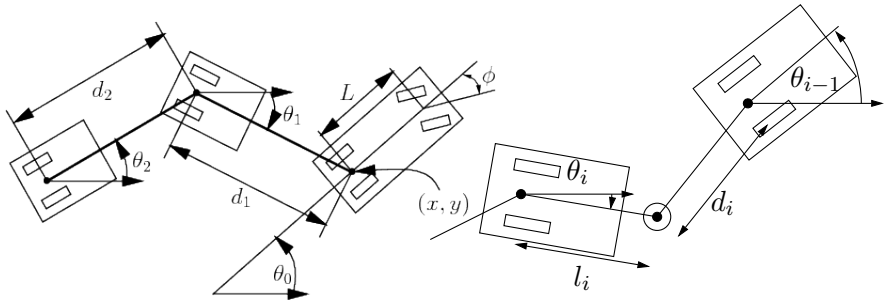
Obrázek 2.12: Bicycle model Zdroj: [36]

Stavové rovnice jsou

$$\begin{aligned} \dot{x} &= v \cdot \cos(\theta) \\ \dot{y} &= v \cdot \sin(\theta) \\ \dot{\theta} &= \frac{v}{L} \cdot \tan(\phi), \end{aligned} \tag{2.17}$$

kde $\left(x, y\right)^T$ je poloha středu zadní osy automobilu, θ je azimut autmobilu. Řídící vstupy $\boldsymbol{u} = \left(v, \quad \phi\right)^T$ odpovídají rychlosti automobilu v a úhlu natočení volantu (viz 2.12). Parametr L je rozvor vozidla. Stavové rovnice lze přepsat tak, aby $\left(x, y\right)^T$ odpovídalo poloze středu vozidla. Představený model splňuje pouze kinematická omezení, nikoliv dynamická, lze však zobecnit na model dynamický. Kvalitativní rozbor a porovnání kinematického a dynamického bicycle modelu se skutečným autem je uveden v [29, 52].

Trailers. Přirozeným rozšířením modelů aut je systém auto-přívěs, jež počítá s libovolným počtem sekvenčně řazených přívěsů za auto. Ačkoliv tato situace není typická na běžných komunikacích, je častá například na letišti, či



(a) : Přímo zapojené přívěsy **(b) :** Přívěsy s kloubem
Zdroj: [36]

Obrázek 2.13: Přívěsy

v průmyslově výrobě. Typicky jsou přívěsy za sebou zapojeny přímo, takové zapojení vede na systém:

$$\begin{aligned}
\dot{x} &= v \cdot \cos(\theta) \\
\dot{y} &= v \cdot \sin(\theta) \\
\dot{\theta}_0 &= \frac{v}{L} \cdot \tan(\phi) \\
\dot{\theta}_1 &= \frac{v}{d_1} \cdot \sin(\theta_1 - \theta_0) \\
&\vdots \\
\dot{\theta}_i &= \frac{v}{d_i}\Big(\prod_{j=1}^{i-1} \cos(\theta_{j-1} - \theta_j)\Big) \sin(\theta_{i-1} - \theta_i) \\
&\vdots \\
\dot{\theta}_n &= \frac{v}{d_n}\Big(\prod_{j=1}^{n-1} \cos(\theta_{j-1} - \theta_j)\Big) \sin(\theta_{n-1} - \theta_n),
\end{aligned} \tag{2.18}$$

kde proměné respektují značení Bicycle modelu (2.17). θ_i je azimut i-tého přívěsu a d_i je délka ramena i-tého vozítka, pro lepší představu viz (2.13a). Rovnice (2.18) předpokládají celkově n zapojených vozíků. S každým vozíkem se tedy konfigurační prostor systému zvyšuje o jednu dimenzi. Pokud je spojení přívěsu kloubové (viz 2.13b), rovnice se změní následovně:

$$\begin{aligned}
\dot{\theta}_i &= \frac{1}{l_i}\Big(\sin(\theta_{i-1} - \theta_i) v_i - \cos(\theta_{i-1} - \theta_i)\, \dot{\theta}_i\, d_i\Big) \\
v_i &= \cos(\theta_{i-1} - \theta_i) v_{i-1} + \sin(\theta_{i-1} - \theta_i)\, \dot{\theta}_i, d_i
\end{aligned} \tag{2.19}$$

kde v_i je rychlost i přívěsu a d_i je délka přední části a l_i délka zadní části i. ramene, které je rozděleno kloubem. Rovnice jsou natolik komplikované, že je nelze převést do zřetězeného stavu jako v (2.18).
Vlastnosti a chování obou typů přívěsu jsou hlouběji studovány a porovnány v [37], odkud byly převzaty.

Kapitola 3

Praktická část

3.1 Algoritmus

V tomto oddíle představíme algoritmus, jenž jsme se rozhodli implementovat. Implementace a výsledky jsou uvedeny v 3.2. Algoritmus stojí především na článku [63], dále pak byly použity myšlenky z [62, 48, 49]. Algoritmus se řadí mezi *deterministické vzorkovací* a využívá koncept prostorové mřížky (viz 2.3.2), jenž rozšiřuje o časový rozměr. Algoritmus plánuje G^2 hladké trajektorie[1], jež jsou nepřímo optimalizované vůči normálovému ryvu. Zároveň implementuje efektivní detekci kolizí pro statické a dynamické překážky. V základní verzi je implementován pro kinematický Bicycle modelem (viz 2.17) a zaručuje spojité vstupy řízení (v, ϕ). Výstupem algoritmu je bezkolizní trajektorie, ze které lze analyticky vyjádřit vstupy řízeného systému (v, ϕ) Algoritmus tak lze využít jako generátor trajektorie pro předovazební řízení v celkové architektuře rozhodovací logiky.

3.1.1 Matematický model a optimální trajektorie

Matematický model. Algoritmus ze své podstaty potřebuje matematický model pro základní soubor trajektorií (viz 2.3.2). Naším modelem bude kinematický Bicycle model, jenž byl představen v předchozí části (viz 2.17).

[1] Tyto trajektorie jsou pro automobily splnitelné.

Připomeňme jen krátce stavové rovnice:

$$\begin{aligned} \dot{x} &= v \cdot \cos(\theta) \\ \dot{y} &= v \cdot \sin(\theta) \\ \dot{\theta} &= \frac{v}{L} \cdot \tan(\phi), \end{aligned} \tag{3.1}$$

Teoretické intermezzo o geometrii. Nežli představíme η-spline křivky, v krátkosti se seznámíme s vybranými pasážemi z teorie diferenciální geometrie, jež jsou nutným minimem k pochopení následujícího oddílu, obsáhlý pohled do teorie diferenciální geometrie je představen v [38]. Zároveň předpokládáme, že čtenář má základní znalost infinitezimálního kalkulu a algebry.

Definice 3.1. Rovinnou orientovanou křivkou rozumíme parametrizovanou vektorovou funkci

$$\begin{aligned} \boldsymbol{p}(t) : [t_0, t_1] &\to \mathbb{R}^2 \\ t &\mapsto \begin{pmatrix} x(t) \\ y(t) \end{pmatrix}, \end{aligned}$$

jež začíná v $\boldsymbol{p}(t_0) = \Big(x(t_0), \quad y(t_0)\Big)^T$ a končí v bodě $\boldsymbol{p}(t_1) = \Big(x(t_1), \quad y(t_1)\Big)^T$.

Tato definice lze přirozeně zobecnit do prostoru $\mathbb{R}^n$. Značením $\dot{\boldsymbol{p}}(t)$ rozumíme první derivaci $\boldsymbol{p}$ dle proměnné t:

$$\dot{\boldsymbol{p}}(t) = \Big(\tfrac{dx}{dt}, \quad \tfrac{dy}{dt}\Big)^T.$$

Je-li $||\dot{\boldsymbol{p}}(t)|| > 0 \quad \forall\, t$ říkáme, že je křivka regulární.

Definice 3.2. Nechť $\boldsymbol{\gamma}(t) : [t_0, t_1] \to \mathbb{R}^n$ je orientovaná křivka. Je-li $\dot{\boldsymbol{\gamma}}(t)$ spojitá na intervalu $[\tau_0, \tau_1] \subset [t_0, t_1]$, pak délku l oblouku křivky $\boldsymbol{\gamma}(t)$ mezi body τ_0 a τ_1 definujeme pomocí vzorce:

$$l \equiv \int_{\tau_0}^{\tau_1} ||\boldsymbol{\gamma}'(t)||\, dt \tag{3.2}$$

Definice 3.3. Nechť $\boldsymbol{\gamma}(t)$, $t \in [t_0, t_1]$ je orientovaná křivka. Funkci s:

$$\begin{aligned} s :& [t_0, t_1] \to R_{\geq 0} \\ s(\tau) &\equiv \int_{t_0}^{\tau} ||\dot{\boldsymbol{\gamma}}(t)||\, dt \end{aligned} \tag{3.3}$$

nazveme obloukem křivky $\boldsymbol{\gamma}(t)$.

Ačkoliv je oblouk křivky s definován korektně (3.3), výpočet definičního integrálu nemusí být nikterak jednoduchý. Oblouk křivky je vždy neklesající

funkce, což si lze i geometricky intuitivně představit. Důkaz plyne přímo z definice (3.3), z nezápornosti normy a vlastnosti integrálu. Pokud je navíc křivka $\boldsymbol{\gamma}(t)$ regulární, je oblouk křivky s rostoucí funkce a existuje k ní tedy inverze s^{-1} (neboť je ryze monotónní). Tedy pro každou délku l křivky $\boldsymbol{\gamma}(t)$ existuje právě jedna hodnota τ, taková, že křivka $\boldsymbol{\gamma}(t)|_{[t_0,\tau]}$ má délku l. Reparamaterizujeme-li křivku jejím obloukem, její velikost rychlosti $||\dot{\boldsymbol{\gamma}}(s)||$ je konstantně rovna jedné. Praktický příklad následuje níže.

Příklad 3.4 (Příklad parametrizace obloukem). Nechť křivka $\boldsymbol{\gamma}$ je parametrizována následovně:

$$\begin{aligned}\boldsymbol{\gamma} &: [0,1] \to \mathbb{R}^2 \\ t &\mapsto \left(t, \quad -\sqrt{1-t^2}\right)^T\end{aligned}$$

Z definice (3.3) vypočítáme délku oblouku

$$\begin{aligned}\hat{s} = s(\tau) &= \int_0^\tau ||\boldsymbol{\gamma}'(t)||\, dt = \int_0^\tau || \left(1, \quad \tfrac{t}{\sqrt{1-t^2}}\right)^T ||\, dt \\ &= \int_0^\tau \sqrt{1^2 + \frac{t^2}{1-t^2}} dt = \int_0^\tau \sqrt{\frac{1}{1-t^2}}\, dt \\ &= \arcsin\tau, \quad \tau \in [0,1].\end{aligned}$$

$\arcsin(x)$ je na intervalu $[0,1]$ ryze monotónní funkce, jejíž inverzí je funkce $\sin(x)$, tedy

$$\tau = \sin(\hat{s})$$

Reparametrizujeme-li $\boldsymbol{\gamma}(\hat{s})$:

$$\begin{aligned}\boldsymbol{\gamma} &: [s(0), s(1)] = [\arcsin(0), \arcsin(1)] = [0, \frac{\pi}{2}] \to \mathbb{R} \\ \hat{s} &\mapsto \left(\sin(\hat{s}), \quad -\sqrt{1-\sin(\hat{s})^2}\right) = \left(\sin(\hat{s}), \quad -\cos(\hat{s})\right)\end{aligned}$$

mělo by platit, že oblouk reparametrizované křivky $s_{\boldsymbol{\gamma}(\hat{s})}(x) = x$. Ověřme výpočtem:

$$\begin{aligned}s(x) &= \int_0^x || \left(-\cos(\hat{s}), \quad \sin(\hat{s})\right) ||\, d\hat{s} = \int_0^x \sqrt{\cos(\hat{s})^2 + \sin(\hat{s})^2}\, d\hat{s} \\ &= \int_0^x 1\, d\hat{s} = x - 0 = x.\end{aligned}$$

Zadefinujme ještě pojmy křivost a geometrická spojitost, které budou v textu užity. Oba koncepty jsou geometrické, a tudiž by nemělo záležet na zvolené parametrizaci křivky.

Definice 3.5. Orientovaná křivka $\boldsymbol{\gamma}$ je třídy G^n (má geometrickou spojitost řádu $n \in \mathbb{N} : n > 1$), pokud existuje lokální regulární parametrizace třídy C^n v okolí každého bodu této křivky.

Matematický koncept křivosti křivky má za cíl odpovídat naivní představě o „křivosti“ křivky. Mělo by se jednat o veličinu, jež nám v libvoloném bodě křivky popíše, jak příliš křivka v jeho okolí mění svůj směr. Zřejmě bychom rádi tvrdili, že přímka je rovná, a tudiž má křivost 0 (podel celé křivky). Navíc kružnice by zřejmě měly mít neměnnou křivost, neboť nelze geometricky rozlišit bod na kružnici bez souřadného systému. Navíc s rostoucím poloměrem r se kružnice v okolí (pevně zvoleného) bodu stále více přimiká k tečné přímce, a proto je v bodě „rovnější“. Zřejmě navíc nemůže záležet na parametrizaci křivky, neboť se jedná o geometrickou vlastnost, nikoliv vlastnost parametrizace. Uveďme příklad, jenž nabídne kandidáta na křivost:

Příklad 3.6. Nechť $\boldsymbol{\gamma}(t)$ je obecná přímka v $\mathbb{R}^2$:

$$\begin{aligned}\boldsymbol{\gamma}(t) : \mathbb{R} &\to \mathbb{R}^2 \\ t &\mapsto \Big(at, \quad bt\Big)^T\end{aligned}$$

Její první derivace $\dot{\boldsymbol{\gamma}}(t)$ je rovna:

$$\dot{\boldsymbol{\gamma}}(t) = \Big(a, \quad b\Big)^T$$

Její druhá derivace $\ddot{\boldsymbol{\gamma}}(t)$ je pak

$$\ddot{\boldsymbol{\gamma}}(t) = \frac{d}{dt}\Big(a, \quad b\Big)^T = \mathbf{0},$$

a tudiž je velikost druhé derivace $||\ddot{\boldsymbol{\gamma}}(t)|| = 0 \quad \forall\, t \in [0,1]$.

Výsledek se zdá slibný, neboť je $||\ddot{\boldsymbol{\gamma}}(t)||$ pro přímky konstantně rovna nule, tedy splňuje první požadovanou vlastnost křivosti. Ověřme další požadavek, plynoucí z úvahy o kružnici a nezávislosti na parametrizaci:

Příklad 3.7. Nechť $\boldsymbol{\gamma}$ je kružnice v $\mathbb{R}^2$ se poloměrem $r \in \mathbb{R}$ s libovolnou parametrizací $\phi(t)$:

$$\begin{aligned}\phi(t) &: [0,1] \to [a,b] \subset \mathbb{R} \\ \boldsymbol{\gamma}(u) &: [a,b] \to \mathbb{R}^2 \\ \boldsymbol{\gamma}(t) &= (\boldsymbol{\gamma} \circ \phi)(t) \\ t &\mapsto \Big(r\,\cos(\phi(t)), \quad r\,\sin(\phi(t))\Big)^T\end{aligned}$$

První derivaci $\dot{\boldsymbol{\gamma}}(t)$ lze rozepsat dle řetízkového pravidla:

$$\begin{aligned}\dot{\boldsymbol{\gamma}}(t) &= \Big(-r\dot{\phi}(t)\sin(\phi(t)) \quad r\dot{\phi}(t)\cos(\phi(t))\Big)^T \\ ||\dot{\boldsymbol{\gamma}}(t)|| &= r\dot{\phi}(t)\end{aligned}$$

Velikost druhé derivace $||\ddot{\boldsymbol{\gamma}}(t)||$ je pak

$$\begin{aligned} ||\ddot{\boldsymbol{\gamma}}(t)|| &= || \begin{pmatrix} -r\dot{\phi}\cos(\phi(t)) - r\ddot{\phi}(t)\sin(\phi(t)) & -r\dot{\phi}(t)\sin(\phi(t)) + r\ddot{\phi}(t)\cos\phi(t) \end{pmatrix}^T || \\ &= \left(r\dot{\phi}(t)\right)^2 + \left(r\ddot{\phi}(t)\right)^2 \end{aligned} \tag{3.4}$$

Požadavek na nezávislost křivosti vylučuje našeho kandidáta $||\ddot{\boldsymbol{\gamma}}(t)||$, neboť je velikost $||\ddot{\boldsymbol{\gamma}}(t)||$ závislá na parametrizaci vztahem (3.4). Povšimněme si, že problém vzniká již při první derivaci křivky,neboť platí:

$$||\dot{\boldsymbol{\gamma}}(t)|| > ||\dot{\boldsymbol{\gamma}}(u)|| \implies ||\ddot{\boldsymbol{\gamma}}(t)|| > ||\ddot{\boldsymbol{\gamma}}(u)|| \quad \forall\, \boldsymbol{\gamma}(t), \boldsymbol{\gamma}(u) \in \mathbb{R}^n : \boldsymbol{\gamma}(t) = \boldsymbol{\gamma}(u).$$

Ovšem zaručíme-li, aby parametrizace byly stejně rychlé, tedy velikosti derivací $||\dot{\boldsymbol{\gamma}}||$ by se v libovolném bodě křivky rovnaly, zajistili bychom, že i $||\ddot{\boldsymbol{\gamma}}||$ by se pro stejné křivky rovnala. Tento požadavek lze jednoduše vyřešit, pokud křivky budeme parametrizovat obloukem. Pak skutečně v libovolném bodě křivky budou velikosti tečných vektorů stejné, neboť budou rovné právě jedné. Definujme tedy křivost křivky:

Definice 3.8. Nechť $\boldsymbol{\gamma}$ je parametrizovaná obloukem s, definujeme její křivost κ v bodě $\boldsymbol{\gamma}(s)$ jako velikost druhé derivace křivky dle s:

$$\kappa \equiv ||\ddot{\boldsymbol{\gamma}}(s)||$$

Vypočtěme křivost kružnice:

Příklad 3.9 (Křivost kružnice). Nechť $\boldsymbol{\gamma}(s)$ je kružnice v $\mathbb{R}^2$ se poloměrem $r \in \mathbb{R}$ a středem v bodě (c_1, c_2) parametrizována obloukem s:

$$\begin{aligned} \boldsymbol{\gamma}(s) : [0, 2\pi r] &\to \mathbb{R}^2 \\ s &\mapsto \left(c_1 + r\,\cos(\tfrac{s}{r}),\quad c_2 + r\,\sin(\tfrac{s}{r})\right)^T \end{aligned}$$

První derivaci $\dot{\boldsymbol{\gamma}}(s)$:

$$\dot{\boldsymbol{\gamma}}(s) = \left(-\sin(\tfrac{s}{r}),\quad \cos(\tfrac{s}{r})\right)^T$$

Křivost κ je pak rovna:

$$\begin{aligned} \kappa &= ||\ddot{\boldsymbol{\gamma}}(s)|| \\ &= || \left(-\tfrac{1}{r}\cos(\tfrac{s}{r}),\quad -\tfrac{1}{r}\sin(\tfrac{s}{r})\right)^T || \\ &= \frac{1}{r} \end{aligned} \tag{3.5}$$

Slabou stránkou výše uvedené definice křivosti κ je použití parametriazace obloukem, která není obvykle snadno použitelná k běžnému výpočtu. Je však potřeba výpočítávat křivost běžně i z obecných parametrizací. K tomu poslouží následující tvrzení:

Tvrzení 3.10. Předpokládejme, že $\boldsymbol{\gamma}(t)$ je regulární křivka v $\mathbb{R}^2$ s libovolnou parametrizací t. Pak pro její křivost κ platí

$$\kappa = \frac{|\dot{\boldsymbol{\gamma}}(t) \cdot \ddot{\boldsymbol{\gamma}}(t)|}{|\dot{\boldsymbol{\gamma}}(t)|^3} = \frac{\ddot{y}\dot{x} - \ddot{x}\dot{y}}{(\dot{x}^2 + \dot{y}^2)^{\frac{3}{2}}}. \tag{3.6}$$

Důkaz. Parametrizujme křivku s obecnou parametrizací $\boldsymbol{\gamma}(t), t \in [t_0, t_1]$ jejím obloukem s (to lze provést vždy) a označme příslušný parametr s. Tedy $s = s(t)\, t \in [t_0, t_1]$, a $\boldsymbol{\gamma}(t) = \boldsymbol{\gamma}(s(t))$, $t \in [t_0, t_1]$. Zřejmě platí:

$$\frac{d\,\boldsymbol{\gamma}(s(t))}{dt} = \frac{d\,\boldsymbol{\gamma}(s(t))}{ds}\frac{ds}{dt}$$

a

$$\begin{aligned}\frac{d^2\,\boldsymbol{\gamma}(s(t))}{dt^2} &= \frac{d}{dt}\left(\frac{d\,\boldsymbol{\gamma}(s(t))}{ds}\frac{ds}{dt}\right) \\ &= \frac{d^2\boldsymbol{\gamma}(s(t))}{ds^2}\left(\frac{ds}{dt}\right)^2 + \frac{d\boldsymbol{\gamma}(s(t))}{ds}\frac{d^2\,s}{dt^2}\end{aligned}$$

K přehlednějšímu zápisu využijeme domluvenou notaci:

$$\dot{\boldsymbol{\gamma}}(t) = \dot{\boldsymbol{\gamma}}(s)\dot{s}(t) \tag{3.7}$$

$$\ddot{\boldsymbol{\gamma}}(t) = \ddot{\boldsymbol{\gamma}}(s)\dot{s}(t)^2 + \dot{\boldsymbol{\gamma}}(s)\ddot{s}(t) \tag{3.8}$$

Neboť platí $||\dot{\boldsymbol{\gamma}}(s)|| = 1$ (z parametrizace oblouku), plyne z rovnice (3.7):

$$\dot{s}(t) = ||\dot{\boldsymbol{\gamma}}(t)|| \tag{3.9}$$

Dále platí

$$\dot{s}(t)\ddot{s}(t) = \ddot{\boldsymbol{\gamma}}(t) \cdot \dot{\boldsymbol{\gamma}}(t) \tag{3.10}$$

neboť z (3.9) plyne:

$$\begin{aligned}\dot{s}(t)^2 &= \dot{\boldsymbol{\gamma}}(t) \cdot \dot{\boldsymbol{\gamma}}(t) \,/\, \frac{d}{dt} \\ 2\dot{s}(t)\ddot{s}(t) &= \ddot{\boldsymbol{\gamma}}(t) \cdot \dot{\boldsymbol{\gamma}}(t) + \dot{\boldsymbol{\gamma}}(t) \cdot \ddot{\boldsymbol{\gamma}}(t) \\ 2\dot{s}(t)\ddot{s}(t) &= 2\dot{\boldsymbol{\gamma}}(t) \cdot \ddot{\boldsymbol{\gamma}}(t) \\ \dot{s}(t)\ddot{s}(t) &= \ddot{\boldsymbol{\gamma}}(t) \cdot \dot{\boldsymbol{\gamma}}(t)\end{aligned}$$

z definice křivosti a rovnic (3.8) a (3.7) můžeme psát:

$$\begin{aligned}\kappa = ||\ddot{\boldsymbol{\gamma}}(s)|| &= \frac{||\ddot{\boldsymbol{\gamma}}(t) - \dot{\boldsymbol{\gamma}}(s)\ddot{s}(t)||}{\dot{s}(t)^2} = \frac{||\ddot{\boldsymbol{\gamma}}(t) - \frac{\dot{\boldsymbol{\gamma}}(t)}{\dot{s}(t)}\ddot{s}(t)||}{\dot{s}(t)^2} \\ &= \frac{||\dot{s}(t)^2\ddot{\boldsymbol{\gamma}}(t) - \ddot{s}(t)\dot{s}(t)\dot{\boldsymbol{\gamma}}(t)||}{\dot{s}(t)^4} \\ &= \frac{||\,(\dot{\boldsymbol{\gamma}}(t) \cdot \dot{\boldsymbol{\gamma}}(t))\,\ddot{\boldsymbol{\gamma}}(t) - (\ddot{\boldsymbol{\gamma}}(t) \cdot \dot{\boldsymbol{\gamma}}(t))\,\dot{\boldsymbol{\gamma}}(t)||}{||\dot{\boldsymbol{\gamma}}(t)||^4}\end{aligned}$$

Využejeme-li vektorovou „BAC-CAB“ identitu

$$\boldsymbol{B}(\boldsymbol{A}\cdot\boldsymbol{C})-\boldsymbol{C}(\boldsymbol{A}\cdot\boldsymbol{B})=\boldsymbol{A}\times(\boldsymbol{B}\times\boldsymbol{C}),$$

můžeme psát:

$$\kappa=\frac{||\dot{\boldsymbol{\gamma}}(t)\times(\ddot{\boldsymbol{\gamma}}(t)\times\dot{\boldsymbol{\gamma}}(t))\,||}{||\dot{\boldsymbol{\gamma}}(t)||^4} \tag{3.11}$$

$$=\frac{||\dot{\boldsymbol{\gamma}}(t)||\;||\,(\ddot{\boldsymbol{\gamma}}(t)\times\dot{\boldsymbol{\gamma}}(t))\,||}{||\dot{\boldsymbol{\gamma}}(t)||^4} \tag{3.12}$$

$$=\frac{||\,(\ddot{\boldsymbol{\gamma}}(t)\times\dot{\boldsymbol{\gamma}}(t))\,||}{||\dot{\boldsymbol{\gamma}}(t)||^3} \tag{3.13}$$

Rozepíšeme-li vektorovoý zápis (3.13) do složek, získáme tvar

$$\kappa=\frac{\sqrt{(\ddot{z}\dot{y}-\ddot{y}\dot{z})^2+(\ddot{x}\dot{z}-\ddot{z}\dot{x})^2+(\ddot{y}\dot{x}-\ddot{x}\dot{y})^2}}{(\dot{x}^2+\dot{y}^2+\dot{z}^2)^{\frac{3}{2}}} \tag{3.14}$$

Za předpokladu, že $\boldsymbol{\gamma}(t)$ je rovinná křivka, lze bez újmy na obecnosti předpokládat, že $z=0$ a výraz (3.14) se zjednoduší na tvar:

$$\kappa=\frac{\ddot{y}\dot{x}-\ddot{x}\dot{y}}{(\dot{x}^2+\dot{y}^2)^{\frac{3}{2}}} \tag{3.15}$$

□

Dále se seznamíme s 2-dimenzionálním Frenet - Serretovým souřadným systémem[2] (též známý jako **TN** souřadný systém).

Definice 3.11. Nechť $\boldsymbol{\gamma}(s)$ je orientovaná křivka v $\mathbb{R}^2$ parametrizovaná obloukem s. Tečný vektor $\mathbf{T}$ a normálový vektor $\mathbf{N}$ definujeme následujícími vzorci:

$$\begin{aligned}\mathbf{T}&\equiv\frac{d\boldsymbol{\gamma}(s)}{ds}\\ \mathbf{N}&\equiv\frac{\frac{d\mathbf{T}}{ds}}{||\frac{d\mathbf{T}}{ds}||}\end{aligned} \tag{3.16}$$

Tvrzení 3.12. Vektory $\mathbf{T},\mathbf{N}$ tvoří ortonormální souřadný system

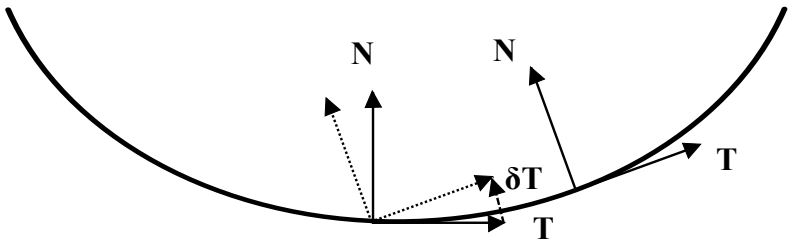
Důkaz. Z definice parametrizace křivky $\boldsymbol{\gamma}(s)$ obloukem s vyplývá, že derivace $\dot{\boldsymbol{\gamma}}(s)$ je jednotkový vektor pro libovolné s, tedy $\mathbf{T}$ je jednotkový vektor a $\mathbf{N}\perp\mathbf{T}$, neboť:

$$1=||\mathbf{T}||^2=\mathbf{T}\cdot\mathbf{T}\;/\;\frac{d}{ds}$$

$$0=\frac{d}{ds}(\mathbf{T}\cdot\mathbf{T})=\frac{d\mathbf{T}}{ds}\cdot\mathbf{T}+\mathbf{T}\cdot\frac{d\mathbf{T}}{ds}=2\frac{d\mathbf{T}}{ds}\cdot\mathbf{T}\implies\mathbf{N}\perp\mathbf{T}$$

Navíc, $\mathbf{N}$ má jednotkovou délku přímo z definice (3.16). □

[2] Běžná definice Frenet-Serretovi báze (**TNB** souřadný systém) je definována v $\mathbb{R}^3$, nám však stačí definice v $\mathbb{R}^2$.



Obrázek 3.1: Planární křivka a **TN** souřadný systém Zdroj: [3]

Poznámka 3.13. Z definice (3.16) a předchozího tvrzení vyplývá, že druhá derivace $\ddot{\boldsymbol{\gamma}}(s)$ je vždy kolmá na první derivaci křivky $\dot{\boldsymbol{\gamma}}(s)$, je-li křivka parametrizována obloukem s. Tedy křivost $\kappa \equiv ||\ddot{\boldsymbol{\gamma}}(s)||$ obecně odpovídá změně orientace první derivace $\dot{\boldsymbol{\gamma}}(t)$ pro libovolnou parametrizaci t.

η-spline křivky. V [48, 49] jsou představeny spline křivky[3] pátého řádu (též η-spline křivky), které jsou geometricky dvakrát diferencovatelné a parametrizovány čtveřicí $\boldsymbol{\eta} = \begin{pmatrix}\eta_1 & \eta_2 & \eta_3 & \eta_4\end{pmatrix}^T \in \mathbb{R}_{>0} \times \mathbb{R}_{>0} \times \mathbb{R} \times \mathbb{R} = \mathcal{H}$:

$$\boldsymbol{\tau}(u) = \begin{pmatrix} x(u) \\ y(u) \end{pmatrix} = \begin{pmatrix} x_0 + x_1 u + x_2 u^2 + x_3 u^3 + x_4 u^4 + x_5 u^5 \\ y_0 + y_1 u + y_2 u^2 + y_3 u^3 + y_4 u^4 + y_5 u^5 \end{pmatrix}, \tag{3.17}$$

kde $u \in [0, 1]$ a

$$\begin{aligned}
x_0 &= x_A \\
x_1 &= \eta_1 \cos\theta_A \\
x_2 &= \frac{1}{2}(\eta_3 \cos\theta_A - \eta_1^2 \kappa_A \sin\theta_A) \\
x_3 &= 10(x_B - x_A) - (6\eta_1 + \frac{3}{2}\eta_3)\cos\theta_A - (4\eta_2 - \frac{1}{2}\eta_4)\cos\theta_B \\
&\quad + \frac{3}{2}\eta_1^2 \kappa_A \sin\theta_A - \frac{1}{2}\eta_2^2 \kappa_B \sin\theta_B \\
x_4 &= -15(x_B - x_A) - (8\eta_1 + \frac{3}{2}\eta_3)\cos\theta_A + (7\eta_2 - \eta_4)\cos\theta_B \\
&\quad - \frac{3}{2}\eta_1^2 \kappa_A \sin\theta_A + \eta_2^2 \kappa_B \sin\theta_B \\
x_5 &= 6(x_B - x_A) - (3\eta_1 + \frac{1}{2}\eta_3)\cos\theta_A - (3\eta_2 - \frac{1}{2}\eta_4)\cos\theta_B \\
&\quad + \frac{1}{2}\eta_1^2 \kappa_A \sin\theta_A - \frac{1}{2}\eta_2^2 \kappa_B \sin\theta_B
\end{aligned} \tag{3.18}$$

[3] Spline křivka je po částech polnomiální křivka, která je na hranici intervalů spojitá. Případně lze vyžadovat i diferencovatelnost až n-tého řádu.

$$
\begin{aligned}
y_0 &= y_A \\
y_1 &= \eta_1 \sin\theta_A \\
y_2 &= \frac{1}{2}(\eta_3 \sin\theta_A - \eta_1^2 \kappa_A \cos\theta_A) \\
y_3 &= 10(y_B - y_A) - (6\eta_1 + \frac{3}{2}\eta_3)\sin\theta_A - (4\eta_2 - \frac{1}{2}\eta_4)\sin\theta_B \\
&\quad - \frac{3}{2}\eta_1^2\kappa_A\cos\theta_A + \frac{1}{2}\eta_2^2\kappa_B\cos\theta_B \\
y_4 &= -15(y_B - y_A) + (8\eta_1 + \frac{3}{2}\eta_3)\sin\theta_A + (7\eta_2 - \eta_4)\sin\theta_B \\
&\quad + \frac{3}{2}\eta_1^2\kappa_A\cos\theta_A - \eta_2^2\kappa_B\cos\theta_B \\
y_5 &= 6(y_B - y_A) - (3\eta_1 + \frac{1}{2}\eta_3)\sin\theta_A - (3\eta_2 - \frac{1}{2}\eta_4)\sin\theta_B \\
&\quad - \frac{1}{2}\eta_1^2\kappa_A\cos\theta_A + \frac{1}{2}\eta_2^2\kappa_B\cos\theta_B,
\end{aligned}
\tag{3.19}
$$

kde (x_A, y_A, θ_A) odpovídá počáteční konfiguraci automobilu a κ_A odpovídá *křivosti* $\boldsymbol{\tau}$ v počátečním bodě $\boldsymbol{\tau}(0)$, analogicky index B značí parametry v koncovém bodě. Rozšíříme konfigurační prostor pro kinematický model C_{kin} o křivost κ. Vznikne tak 4 dimenzionální prostor C s konfiguracemi $q \in C$, jež odpovídají okrajovým podmínkám η-spline křivek. Křivku, která vede z konfigurace p do konfigurace q budeme značit $\boldsymbol{\tau}_{(p,q)}$.
V [49] je podrobné odvození rovnic (3.18) a (3.19) z okrajových podmínek $\boldsymbol{\tau}$. Článek zároveň obsahuje důkaz spojitosti druhé derivace $\ddot{\boldsymbol{\tau}}(t)$ pro libovlnou přípustnou čtveřici parametrů $\boldsymbol{\eta}$ a libovolnou dvojci konfigurací $p, q \in C$.

Optimalizace. Neboť přípustných η-spline křivek spojujících dvě libovolné konfigurce $\tau_{(p,q)}$ je nespočetně mnoho, je z nich potřeba dle vhodného kritéria vybrat. Kritérium představené v [48] je $|\frac{d\kappa}{ds}|$, kde s je oblouk křivky $\tau_{(p,q)}$ a κ je její křivost. Ukažme, jak lze minimalizaci $|\frac{d\kappa}{ds}|$ interpretovat z hlediska automobilu a jeho řízení. Z modelu auta (2.17) a (2.12) lze psát:

$$
\begin{aligned}
\theta(t) &= \arctan\frac{\dot{y}(t)}{\dot{x}(t)} \,/\, \frac{d}{dt} \\
\dot{\theta}(t) &= \frac{1}{1 + (\frac{\dot{y}(t)}{\dot{x}(t)})^2} \cdot \left(\frac{\dot{y}(t)}{\dot{x}(t)}\right)' \\
\dot{\theta}(t) &= \frac{\dot{x}(t)^2}{\dot{x}(t)^2 + \dot{y}(t)^2} \cdot \frac{\ddot{y}(t)\dot{x}(t) - \dot{y}(t)\ddot{x}(t)}{\dot{x}(t)^2} \\
\dot{\theta}(t) &= \frac{\ddot{y}(t)\dot{x}(t) - \dot{y}(t)\ddot{x}(t)}{\dot{x}(t)^2 + \dot{y}(t)^2} \\
\dot{\theta}(t) &= \sqrt{\dot{x}(t)^2 + \dot{y}(t)^2} \cdot \frac{\ddot{y}(t)\dot{x}(t) - \dot{y}(t)\ddot{x}(t)}{(\dot{x}(t)^2 + \dot{y}(t)^2)^{\frac{3}{2}}}
\end{aligned}
$$

Z definice křivosti (3.6) a modelu (2.17) lze poslední rovnost přepsat na

$$\dot{\theta} = v \ \cdot \kappa \tag{3.20}$$

Vyjádříme-li úhel ϕ z rovnic (2.17):

$$\phi = \arctan\left(\frac{\dot{\theta}L}{v}\right) = \arctan(L\kappa). \tag{3.21}$$

Za předpokladu malé křivosti κ lze tvrdit, že představené kritérium přibližně odpovídá velikost derivace úhlu natočení volantu dle času $|\frac{d\phi}{dt}|$, neboť

$$\frac{d\phi}{dt} = \frac{d}{dt}\arctan(L\kappa) = L\frac{d\kappa}{dt}\frac{1}{(L\kappa)^2+1} \overset{\kappa \ll 1}{\approx} L\frac{d\kappa}{dt} \tag{3.22}$$

Předpoklad malé křivosti je pro plánovaní trajektorie automobilů platný, neboť pro automobily a systémy s přívěsem je minimální poloměr zatáčení standardně $r_{min} > 10\,\text{m}$. Z rovnice (3.5) navíc víme, že křivost kružnic je $\kappa = \frac{1}{r}$, neboli $\kappa < 0,1$ pro běžné automobily. Lze tedy předpokládat, že $\frac{d\phi}{dt} \propto \frac{d\kappa}{dt}$. Navíc dle pravidla zřetězení lze psát

$$\frac{d\kappa}{dt} = \frac{d\kappa}{ds}\frac{ds}{dt} = \frac{d\kappa}{ds}v. \tag{3.23}$$

Předpokládme-li, že rychlost v se podel trajektorie mění minimálně, či je dokonce konstantní, lze psát $\frac{d\phi}{ds} \propto \frac{d\phi}{dt} \propto \frac{d\kappa}{dt}$ [48].
Cílem je tedy minimalizovat normálový ryv $j_n = \frac{da_n}{dt}$[4] po celé trajektorii $\tau_{(p,q)}$, resp. minimalizovat rychlé změny v řízení $\frac{d\phi}{dt}$.
Optimalizační úlohu formulujeme

$$\min_{\boldsymbol{\eta}\in\mathcal{H}} \max_{u\in[0,1]} \left|\frac{d\kappa}{ds}\right| \tag{3.24}$$

za podmínek

$$||\dot{\tau}(u)|| > 0 \quad \forall\, u \in [0,1]. \tag{3.25}$$

Jedná se o úlohu, jejíž suboptimální hodnoty lze nalézt pomocí SQP iteračních metod, či algoritmem, jež byl představen Guarinom a Piazzim v [17].

3.1.2 Soubor primitiv a časoprostorová mřížka

Navážeme-li na předchozí kapitolu, kde jsme představili 4-dimenzionální konfigurační prostor C s konfigurací q:

$$q = \begin{pmatrix} x, & y, & \theta, & \kappa \end{pmatrix}^T \in C$$

[4] a_n značí normálové zrychlení autmobilu

a rozšíříme jej o časovou proměnou t tak, abychom popsali stav automobilu jednoznačně, vznikne 7 dimenzionální prostor C_{time} (2D pozice, 2D rychlost, 2D zrychlení, čas) [63]. V našem případě popíšeme robota konfigurací q_t následovně:

$$q_t = \begin{pmatrix} x, & y, & \theta, & v, & a_t, & a_n, & t \end{pmatrix}^T \in C_{\text{time}},$$

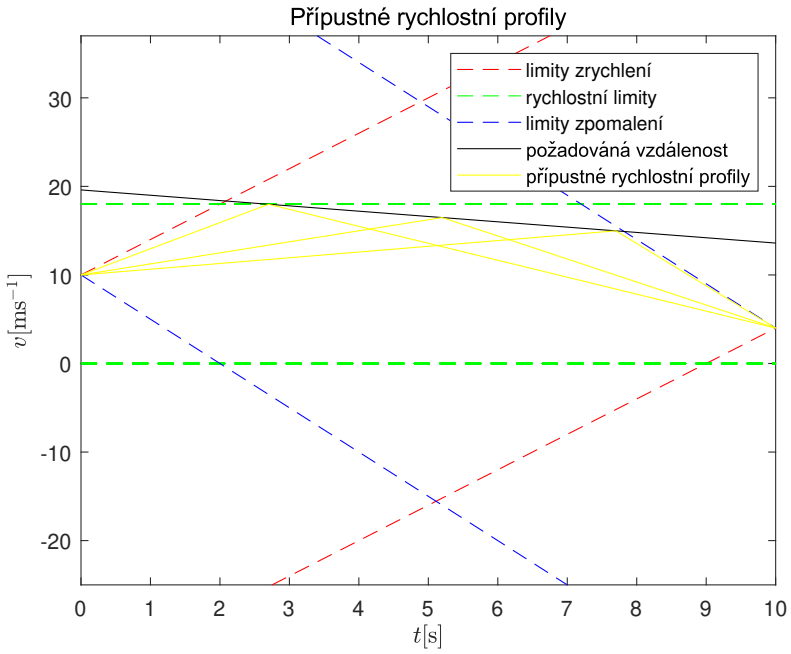
kde (x, y) odpovídá pozici, θ odpovídá azimutu automobilu a v tečné rychlosti autmobilu ve směru určeném úhlem θ. Dvojce (a_t, a_n) pak určuje zrychlení automobilu, kde a_t je tečné zrychlení a a_n normálové zrychlení. Konečně t označuje čas.
Přidáním časové informace jsme velikost dimenze konfiguračníního prostoru rozšířili z 3 resp. 4 na 7, čímž jsme zvýšili náročnost tvorby mřížky, která se stává časově neefektivní. V [63] je představena efektivní metoda tvorby mřížky pro běžné automobily na silnici, již lze využít i v našem případě tovární haly za předpokladu splnění následujících podmínek:

- Šířka vyhrazeného jízdního pruhu pro vozidlo neumožňuje otočení vozidla. Vozidlo se může tedy otáčet pouze na místech k tomu určených.
- Vozidlo je schopno se pohybovat po prostorových spline-křivkách $\boldsymbol{\tau}(t)$ s po částech afinním profilem rychlosti v.

V takovém případě lze efektivně zredukovat strukturu mřížky z 7 dimenzionálního prostoru na 4 dimenzionální prostor, předpokládáme-li, že všechny uzly v mřížce budou mít nulovou křivost $\kappa = 0$, nulové zrychlení $a = 0$ a budou směřovat vždy ve směru vozovky $\theta = \theta_{\text{vozovka}}$.
Při tvorbě trajektorie $\boldsymbol{\nu}_{(q,p)} \subset C_{\text{time}}$ spojující konfigurace $q, p \in C_{\text{time}}$ nejdříve zkonstrujeme optimální trajektorii $\boldsymbol{\tau}(u) \in C$ nezávislé na časovém profilu. Nad touto trajektorií pak vytvoříme po částech afinní profil rychlosti v tak, aby byly splněny okrajové podmínky rychlosti a celková dráha odpovídala délce křivky $\boldsymbol{\gamma}(u)$:

$$t_q < t_p \tag{3.26}$$
$$v(t_q) = v_q \tag{3.27}$$
$$v(t_p) = v_p \tag{3.28}$$
$$\int_{t_q}^{t_p} v = \int_0^1 ||\boldsymbol{\gamma}'(u)||\, du. \tag{3.29}$$

Tato postupná konstrukce sice nezaručuje, že výsledné trajektorie $\boldsymbol{\nu}_{(q,p)}$ jsou optimální vůči zvolenému kritériu, zjednodušuje však celkovou strukturu základního souboru a zaručuje, že trajektorie $\boldsymbol{\nu}_{(q,p)}$ je dynamicky splnitelná. Navíc při vhodném $\Delta q \in C_{\text{time}}$ lze zaručit, že soubor primitiv bude stále obsahovat křivky $\boldsymbol{\nu}_{(q,p)}$ s konstantním profilem rychlosti v, a tudiž optimální vzhledem k $\frac{d\kappa}{ds}$ (z rovnice 3.23). Uvažujme, že rychlostní profil je složen ze dvou



Obrázek 3.2: Rychlostní profily

affiních funkcí a splňuje okrajové podmínky a podmínku ujeté vzdálenosti. Na obrázku (3.2) je provedena vizualizace. Tato formulace dovoluje více řešení, ze kterých lze vybírat. Přirozené kritérium je minimalizovat zrychlení, neboť je přímo úměrné spotřebě vozidla či plynulosti pohybu. Kritérium minimalizující celkovou velikost zrychlení je

$$f(t) = |a_1| + |a_2| = |\frac{s(t) - v(t_q)}{t - t_q}| + |\frac{v(t_p) - s(t)}{t_p - t}|, \tag{3.30}$$

kde $s(t)$ odpovídá funkci, jež popisuje rychlost nutnou pro splnění podmnínky dráhy (3.29) (černá v (3.2)). Dalším vhodnými kritériem optimalizace může být minimalizace rozdílu zrychlení a_1 a a_2:

$$g(t) = |a_1 - a_2| = |\frac{s(t) - v(t_q)}{t - t_q} - \frac{v(t_p) - s(t)}{t_p - t}| \tag{3.31}$$

Ve finální implementaci jsou trajektorie optimalizovány dle kritéria $g(t)$.

3.1.3 Graf

Grafové reprezentace standardně využívají algoritmy na nalezení nejkratší cesty, mezi které patří Dijkstrův algoritmus či A*. Případně vylepšený Stentzův D* [56] či Focussed D* [57], jež taktéž patří do stejné třídy. Všechny

tyto algoritmy udržují navštívené uzly uspořádané, což v časověprostorových mřížkách není nutné, neboť čas takové uspořádání přirozeně tvoří. Tedy v grafu nad časoprostorovou mřížkou nelze vytvořit cyklus, neboť se nelze vrátit v čase. Vzniklý graf je tedy orientovaný a acyklický a lze na něj uplatnit algoritmus na nalezení nejkratší cesty s časovou složitostí $O(n)$ narozdíl od dříve zmíněných algoritmů, jež jsou nejlépe $O(n \log n)$. Hlubší teoretický popis algoritmu pro orientované a acyklické grafy je uveden v [10, 47].[63]

3.1.4 Detekce kolizí

Algoritmus používá detekci kolizí, jež je založena na bitmapách (viz 2.1) a inspiruje se v článku [62]. Dříve, než-li popíšeme proces detekce, odvodíme podobu C_{obs} pro automobil založený na Bicycle-model s nenulovou šířkou w a délkou l. Začněme však postupně jednoduššími příklady.

Příklad 3.14 (Bod v rovině). Předpokládejme na okamžik, že automobil nemá obsah, je reprezentován pouze bodem (x, y) a překážky jsou reprezentovány mnohostěny (viz 2.1). Pak se libovolná překážka $O \in \mathscr{O} \subset \mathscr{W}$ do konfiguračního prostoru Bicycle modelu C_{kin} promítne nezávisle na azimutu vozu θ, neboť je automobil v nedovolené konfiguraci právě tehdy, když $(x, y) \in \mathscr{O}$. Prostor $C_{obs} \subset C_{\text{kin}}$ má pak podobu:

$$C_{obs} = \mathscr{O} \times [0, 2\pi)$$

a transformace je tudiž triviální.

Příklad 3.15 (Kruh v rovině). Přidejme do předchozí úvahy předpoklad, že automobil má tvar kruhu. Pak je vozidlo reprezentováno bodem (x, y) a poloměrem kruhu r. Vozidlo je v kolizi s libovolnou překážkou $O \in \mathscr{O} \subset \mathscr{W}$, pakliže bod (x, y) je vzdálen od překážky O méně jak r:

$$d((x, y), O) < r$$

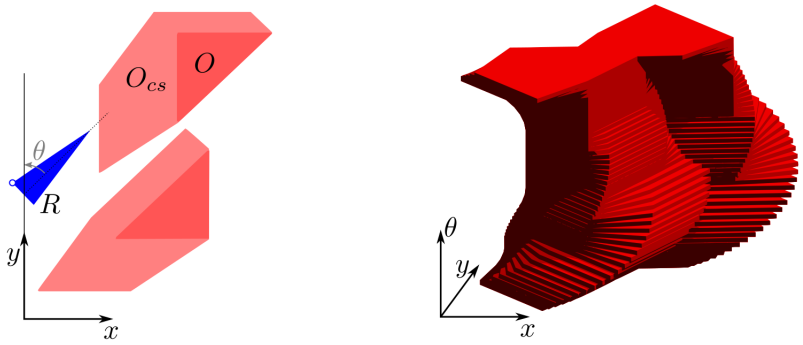
Překážka se tedy „nafoukne“ o r. Opět je kolizní konfigurace nezávislá vůči azimutu θ. To je způsobeno tím, že kruh T je invariantní vůči rotaci kolem svého středu, tedy platí

$$R_{(\alpha)}(T) = T \quad \forall \alpha \in [0, 2\pi],$$

kde $R_{(\alpha)}$ symbolizuje rotaci o úhel α.

Příklad 3.16 (Pevně fixované těleso v rovině). Vraťme se k příkladu bodu v rovině (3.14). Představme si nyní holonomního robota s netriviálním tvarem $R \in \mathscr{W}$, který je fixně orientován v prostoru pod úhlem θ. Robot se může pohybovat libovolným směrem. Výsledný prostor $C_{obs} = O_{cs}$ lze popsat pomocí Minkovského sumy, jež je pro dvě množiny $A, B \subset \mathbb{R}^n$ definována následovně:

$$A \oplus B = \{\boldsymbol{a} + \boldsymbol{b} | \boldsymbol{a} \in A, \boldsymbol{b} \in B\}$$



(a) : Fixované těleso v rovině **(b) :** Automobil v rovině

Obrázek 3.3: C_{obs} prostor pro kinematický Bicycle model Zdroj: [63]

V našem případě tedy:

$$C_{obs} = R \oplus \mathscr{O}$$

Minkovského suma zobecňuje úvahu z příkladu (3.15), kdy nyní překážky „nafukujeme“ o tvar tělesa R. Pro lepší představu viz (3.3a).

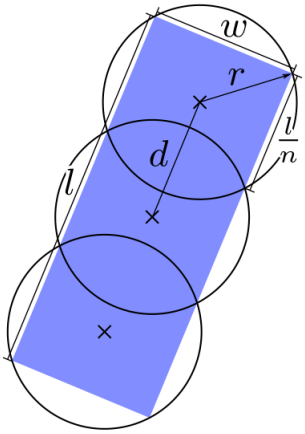
K určení C_{obs} pro autombil vyjedeme z příkladu (3.16). Uvědomíme-li si, že automobil je pro daný úhel θ pevně fixované těleso, stačí tvar automobilu T sečíst Minkovského sumou pro každý úhel $\theta \in [0, 2\pi)$ s oblastí překážek $\mathscr{O}$:

$$C_{obs} = \bigcup_{\theta \in [0,2\pi]} T_\theta \oplus \mathscr{O}, \tag{3.32}$$

kde $T_\theta = R_{(\theta)}(T)$ je tvar T zrotován o úhel θ v bodě středu zadní nápravy (x, y). Vizualizace s diskretizovanými hodnotami θ je v (3.3b). Detekce kolizí je extenzivně využívaná utilita libovolného algoritmu, zvlášťe pak algoritmů založených na tvorbě mřížek. Tvoří tak uzké hrdlo programu. Implementovat algoritmus naivně a pro každý stav automobilu detekovat kolizi online v běhu je tak nemyslitelné. Cílem je přesunout výpočet a tvorbu co nejvíce do offline fáze, zefektivnit zápis a reprezentaci překážek v počítačové struktuře. Nejpřirozenější reprezentace pro počítač jsou bitmapy. Již v (2.1) jsme popsali jejich princip, zde zdůrazněme, že výhodou je i jednoduchý zápis nestrukturovaných dat ze senzorů přímo do struktury, se kterou algoritmus je schopen účinně počítat a snižujeme tak nároky na online výpočet programu, který počítá s dynamicky měnící se mapou. Dalším zefektivněním bude vhodná reprezentace auta.
S představou tvaru C_{obs} rozvineme myšlenku z příkladu (3.15). Pokud rozložíme tvar automobilu s šířkou w a délkou l (obdélník) na kružnice D se stejným poloměrem r (resp. vytvoříme jeho pokrytí kružnicemi, viz (3.5)), zjednodušíme reprezentaci automobilu z obdélníku na jednotlivé body S, které odpovídají středům kružnic a tvoří jádro I (viz 3.4).
Pak stačí každou překážku „nafouknout“ o poloměr r a kontrolovat pouze I na kolizi. Toto zefektivní výpočet, neboť orientace automobilu θ je skryta

(a) : Aproximace automobilu **(b) :** Disk D **(c) :** jádro I

Obrázek 3.4: Automobil a rozklad na disk a jádro Zdroj: [63]



Obrázek 3.5: Pokrytí autmobilů kružnicemi Zdroj: [63]

pouze v jádře I a „nafouknoutí“ překážek je tak pro libovolný úhel θ stejné a stačí provést jen jednou [63]. Tato myšlenka je zaznamenána na obrázku (3.6).

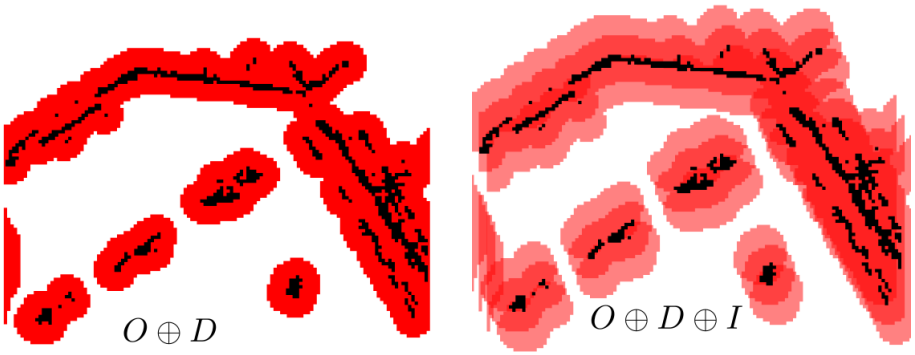
Pokrytí kružnicemi, jež je na obrázku (3.5), je možno dosáhnout pomocí vzorců[5]:

$$\begin{aligned} r &= \frac{1}{2}\sqrt{\frac{l^2}{n^2} + w^2} \\ d &= 2\sqrt{r^2 - \frac{w^2}{4}} \end{aligned} \tag{3.33}$$

Inkrementální algoritmus, jenž je schopen aproximovat kružnici pomocí bodů v bitmapě s rozlišením ϵ je podrobně představen v [63], my zde jen ukážeme pseudokód[6] v (4).

Výsledný postup detekce je následující: Automobil je v offline fázi pokryt n kružnicemi dle rovnic (3.33), Pro každou křivku $\tau(t)$ ze souboru primitv předpočítáme (v m jejich bodech) konfigurace robota q. Z azimutu θ a rozlišení bitmapy ϵ vypočteme obraz jádra I v bitmapě (pro každý bod podel $\boldsymbol{\tau}(t)$). K výsledné trajektorii si uložíme těchto m bodů, kde každý bod je

[5] V originálním článku [63] jsou tyto rovnice chybně.
[6] Pseudokod uvádený v [63] je chybný.



(a) : „Nafouknutí“ překážek o kruh D

(b) : „Nafouknutí“ překážek o celý automobil $D \oplus I$

Obrázek 3.6: Automobil a rozklad na disk a jádro Zdroj: [63]

Algorithm 4 Disk approximation

```
function DRAW CIRCLE(r)
    x = round(r)
    y = 0
    F = x(x − 1) − r^2 + 5/4
    while y ≤ x do
        F = F + 2y + 3
        y = y + 1
        if F > 0 then
            F = F − 2x + 2
            Vytvoř obdélník [−x, −y] × [x, y]
            Vytvoř obdélník [−y, −x] × [y, x]
            x = x − 1
        end if
    end while
end function
```

reprezentován n-ticí bodů reprezentující I a časem průjezdu t:

$$\begin{bmatrix} \begin{pmatrix} x_{11} & y_{11} & t_1 \\ x_{21} & y_{21} & t_1 \end{pmatrix} & \begin{pmatrix} x_{12} & y_{12} & t_1 \\ x_{22} & y_{22} & t_1 \end{pmatrix} & \dots & \begin{pmatrix} x_{1n} & y_{1n} & t_1 \\ x_{2n} & y_{2n} & t_2 \end{pmatrix} \\ & & \vdots & \\ \begin{pmatrix} x_{m1} & y_{m1} & t_1 \end{pmatrix} & \begin{pmatrix} x_{m2} & y_{m2} & t_m \end{pmatrix} & \dots & \begin{pmatrix} x_{mn} & y_{mn} & t_m \end{pmatrix} \end{bmatrix}. \tag{3.34}$$

Při online výpočtu pak každou překážku „nafoukneme“ o kruh s poloměrem r z (3.33) (při zápisu do bitmapy). Trajektorie je bezkolizní, pakliže všech m bodů příslušné trajektorie je bezkolizních.

3.2 Implementace a Výsledky

Implementace a knihovny. Algoritmus byl implementován v `Pythonu` a kritická on-line část v `C`[7]. Kód je optařen komentáři přímo v kódu a nebude podrobněji komentován.
Off-line část, jež je implementována v `Python` využívá `Numpy` a `Scipy` knihovny pro tvorbu η-spline křivek, optimalizaci a její reprezentaci. Dále je využit modul `re` a `pickle` při vyčítání a ukládání dat. Další knihovny, jež jsou v kódu využity, nejsou nutné pro běh programu.
Program v `C` využívá knihovnu třetí strany, v níž jsou implementované základní algoritmické struktury. Knihovna je dostupná z `https://github.com/fragglet/c-algorithms`.Tuto závislost lze překonat doimplementováním vlastní struktury `halda`. Program `C` dále využívá základní knihovny a matematickou knihovnu.

Výsledky. Simulace programu probíhala na osobním počítači s procesorem Intel Core i5-3337U s taktem 1,80 GHz. V simulaci uvažujeme triviální rovnou cestu s délkou 200 m a šířkou 10 m a maximálním simulačním časem $T = 60$ s. Parametry diskretizace a grafu jsou uvedeny v tabulkách (3.1) a (3.2). Tabulka (3.1) odpovídá základnímu souboru pohybů, respektive popisuje přechodové hrany v grafu. Dva uzly v grafu q_1, q_2 mohou být propojeny pouze pokud jejich rozdíl menší než přechodové maximum $\Delta_{\max} q > \Delta q = q_2 - q_1$. Tato množina je očištěna o přechody, jež nejsou uskučitelné modelem auta,jehož parametry jsou uvedeny v (3.3).

V simulaci má vozidlo najít co nejrychlejší bezkolizní trajektorii ze své počáteční polohy do požadované polohy:

$$\tilde{q}_{start} = \begin{pmatrix} 8 \\ 4 \end{pmatrix} \longrightarrow \begin{pmatrix} 120 \\ 8 \end{pmatrix} = \tilde{q}_{final}$$

Vozidlo je v simulaci předjížděno dvěma dalšími vozidly, kterým se výhýbá a zpomaluje, aby jim umožnil předjetí. Ve vizualizaci (3.7) a (3.8) je graficky znázorněna nalezená trajektorie. Zelené body reprezentují okamžité[8] polohy překážek v čase t. Modrý obdélník reprezentuje vlastní vozidlo. Algoritmus našel pro tento vstup výslednou trajektorii, která lze dosáhnout během 17 s. Offline fáze tvroby grafu trvala do 30 sekund bez optimalizace s pomocí vyhledávací tabulky ideálních parametrů $\boldsymbol{\eta}$ (3.1.1). Offline fáze bez této tabulky trvá v řádu desítek minut. Online fáze implementovaná v `C` běžela

[7] Kód je přiložen k práci
[8] Vizualizace zobrazuje v čase t všechny polohy překážek pro čas $[t - 0,5; t + 0,5]$

$\Delta_{\max} t$	8	[s]
$\Delta_{\max} x$	24	[m]
$\Delta_{\max} y$	4	[m]
průměrný faktor větvení		≈ 153

Tabulka 3.1: Lokální propojenost grafu

veličina	symbol	Δ	jednotka	max hodnota	# hodnot
čas	t	2	[s]	60	31
podelná pozice	x	8	[m]	200	26
příčná pozice	y	2	[m]	10	6
rychlost	v	2	[ms^{-1}]	8	5
# uzlů					24180

měřítko bitmapy	1:10
časové měřítko bitmapy	1:4

Tabulka 3.2: Základní parametry algoritmu

0,31 s. Finální rekonstrukce cesty implementovaná v `Pythonu` trvala 0,5 s. Více příkladu zde v práci neuvadíme, lze však pomocí přiloženého kódu generovat náhodné příklady a upravovat zadání dle vůle čtenáře.

3.3 Závěr

Implementovali jsme v minimálním rozsahu algoritmus představený v (3.2). Algoritmus je implementován pro nalezení nejrychlejší trajektorie. Ovšem struktura řešení dovoluje používat libovolnou heuristiku a vážící funkci, jež lze zadat externě. Tato změna by zároveň neměla výrazně zhoršit výpočetní čas, neboť algoritmus prohledává dostupnou část z q_{init} kompletně. Toto lze navíc uplatnit, nelze-li se dostat přímo do požadované konfigurace q_{final}. Lze se totiž postupně dotazovat na blízké konfigurace q v okolí q_{final}, čímž lze zaručit alespoň přiblížení k žádanému stavu. To zároveň implikuje, že časová náročnost algoritmu není závislá na prostorově-časové vzdálenosti. Algoritmus předpokládá, že vozidlo má plnou informaci o překážkách a jejich trajektoriích v daném časovém úseku, což lze v továrním prostředím zajistit vytvořením komunikační sítě mezi pohybujícími se vozidly, případně vhodnou úpravou dat ze senzorů a odhadováním trajektorie[9]. Algoritmus lze využít i pro více entit, pakliže předpokládáme, že entity mají prioritu, která tvoří lineární uspořádání (nad množinou všech entit). Jinak je vhodný pouze pro plánování na úrovni jednotlivých entit a nelze jej jednoduše zobecnit pro hledaní optimálních cest pro více vozidel se stejnou prioritou. Bohužel se

[9] Například použitím neuronové sítě.

veličina	symbol	hodnota	jednotky
délka	l	5	[m]
šířka	w	2	[m]
maximální rychlost	$v_{\max}$	10	$[\mathrm{ms}^{-1}]$
minimální rychlost	$v_{\min}$	0	$[\mathrm{ms}^{-1}]$
maximální akcelerace	$a_{\max}$	4	$[\mathrm{ms}^{-2}]$
minimální akcelerace	$a_{\min}$	-5	$[\mathrm{ms}^{-2}]$
maximální úhel natočení kol	$\phi_{\max}$	$\frac{\pi}{6}$	[rad]

Tabulka 3.3: Parametry modelu automobilu

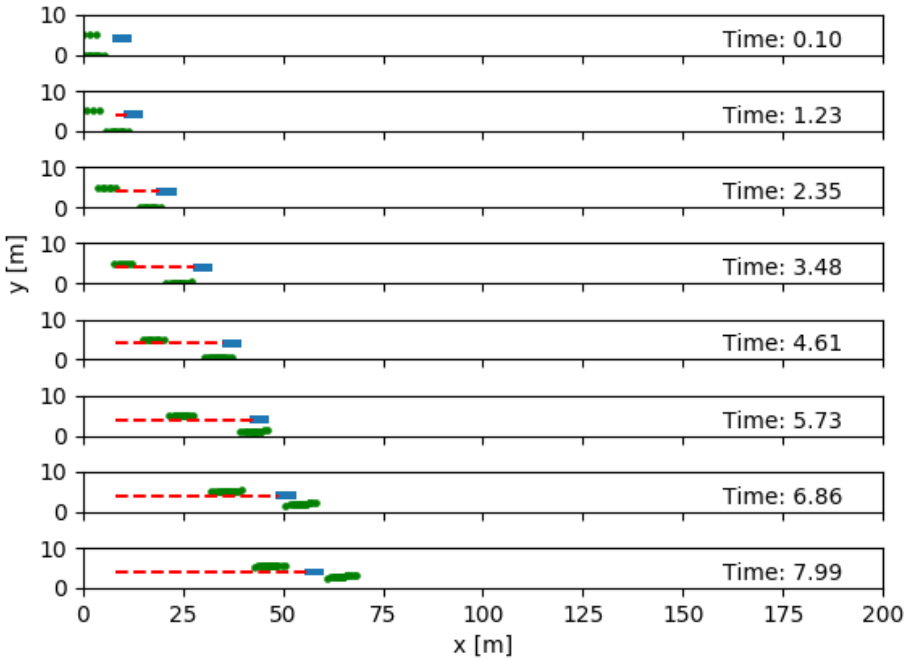
nepodařilo z časových důvodů implementovat celý algoritmus v nativním `C`, který by oproti `Pythonu` šel paralelizovat. I bez vhodné paralelizace by došlo k zrychlení algoritmu, jež by dovolovalo hustější mřížku, či plánování na delší časový horizont.

3.3.1 Budoucí práce

Ačkoliv se nepodařilo (opět z časových důvodů) zobecnit prezentovaný algoritmus na systémy s více přívěsy, které odpovídají struktuře milk-run vozidel, podařilo se implementovat alespoň částečné kroky, jež prokazují, že představený algoritmus lze za určitých předpokladů použít i pro složitější systémy, jako je milk-run. Konkrétní implementace[10] předpokládá, že jednotlivé přívěsy mají řiditelné nápravy a jsou schopny kopírovat původní prostorovou trajektorii vedoucího vozu. Ačkoliv se jedná o silný požadavek na milk-run, řiditelné nápravy jsou běžně v praxi implementovány a jsou poskytovány s algoritmy řízení[11].
Konfigurační prostor robota C_{time} je nutné rozšířit na C_{trailer}, abychom byli schopni popsat stav celého milk-runu. Pro jednoduchost nyní předpokládejme, že délka milk-runu je menší než nejkratší η-spline křivka, která reprezentuje přechod $\tau_{(p,q)}$ mezi dvěma stavy q, p v mřížce nad C_{time}. V takovém případě je stav milk-runu jednozančně určen takovýmto přechodem $\tau_{(p,q)}$. Stačí tedy konfiguraci $q_{\text{time}} \in C_{\text{time}}$ rozšířit o stav c, abychom získali novou konfiguraci $q_{\text{trailer}} \in C_{\text{trailer}}$:

$$C = C_{\text{time}} \times \mathbb{R}$$
$$q = q_{\text{time}} \cup \{c\}.$$

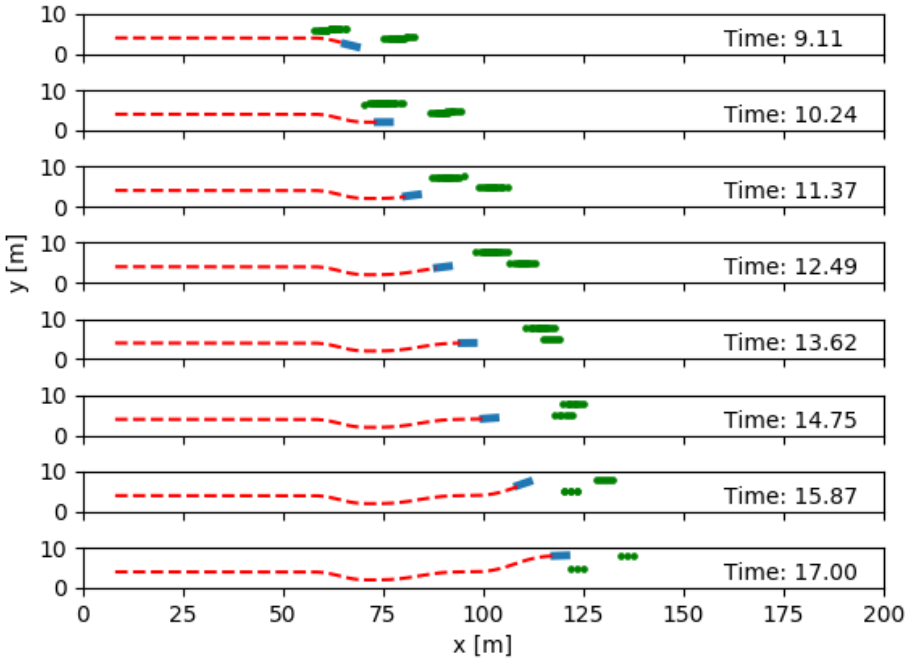
[10] Částečná implementace je přiložena ve zdrojovém kódu.
[11] Například `https://www.jungheinrich.cz/produkty/manipulacni-technika/privesy`



Obrázek 3.7: Vzorový příklad - část první

V obecném případě dojde pouze ke zvýšení dimenze nového stavu c. Dimenze bude odpovídat počtu nejkratších trajektorií[12], které budou stačit k jednoznačnému popisu stavu milk-runu. Popis je jednoznačný, je-li součet délek jednotlivých trajektorií větší než délka milk-runu.
Detekce kolize pak lze snadno rozšířit, neboť vozíky stále drží trajektorii prvního vozu, a lze tak již vypočítané body (3.34) využít jednoduchým posunutím v čase. Tento přístup však vyžaduje, že jsou body (3.34) vzorkovány rovnoměrně dle délky trajekorie. To v momentální implementaci však neplatí a je nutné vzorkování upravit. Momentální výsledky ukazují, že časová náročnost výpočtu v on-line fázi je přímo úměrná velikosti prohledávaného grafu. Výzvou budoucí práce je tedy nalézt vhodnou strukturu kánonických pohybů, zjednodušit mřížku nad C_{trailer}, či provést paralelizaci výpočtu. To je možné, neboť to dovoluje struktura výpočtu. Trivilním řešením časové náročnosti je pak snížit hustotu vzorkování mřížky, či zmenšit celkový prostor, který je grafem prohledáván. To způsobí snížení velikosti grafu, což se přímo odrazí ve výpočetní době. Částečné výsledky ukazují, že zvolený přístup je uskutečnitelný a má smysl se mu nadále věnovat.

[12] Ve smyslu prostorové vzdálenosti.



Obrázek 3.8: Vzorový příklad - část druhá

Příloha A

Bibliografie

[1] Salix alba. URL: https://commons.wikimedia.org/w/index.php?curid=46816719.

[2] Salix alba. URL: https://commons.wikimedia.org/w/index.php?curid=46816720.

[3] Salix alba. URL: https://commons.wikimedia.org/w/index.php?curid=2656277.

[4] Devin J Balkcom a Matthew T Mason. "Time optimal trajectories for bounded velocity differential drive vehicles". In: *The International Journal of Robotics Research* 21.3 (2002), s. 199–217.

[5] Michael Branicky et al. "Quasi-randomized path planning". In: sv. 2. Ún. 2001, 1481–1487 vol.2. ISBN: 0-7803-6576-3. DOI: 10.1109/ROBOT.2001.932820.

[6] Marcus Brazil et al. "Demonstrating efficiency gains from installing truck turntables at crushers". In: (lis. 2015).

[7] Xuân-Nam Bui et al. "Shortest path synthesis for Dubins non-holonomic robot". In: *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*. IEEE. 1994, s. 2–7.

[8] John Canny. *The complexity of robot motion planning*. MIT press, 1988.

[9] John Canny et al. *On the complexity of kinodynamic planning*. Tech. zpr. Cornell University, 1988.

[10] Thomas H Cormen et al. *Introduction to algorithms*. 2009.

[11] Lester E Dubins. "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents". In: *American Journal of mathematics* 79.3 (1957), s. 497–516.

[12] David Eppstein. "Graph-Theoretic Solutions to Computational Geometry Problems". In: *Graph-Theoretic Concepts in Computer Science*. Ed. Christophe Paul a Michel Habib. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, s. 1–16. ISBN: 978-3-642-11409-0.

[13] Brett Fajen et al. "A Dynamical Model of Visually-Guided Steering, Obstacle Avoidance, and Route Selection". In: *International Journal of Computer Vision* 54 (srp. 2003), s. 13–34. DOI: 10.1023/A:1023701300169.

[14] Emilio Frazzoli, Munther A. Dahleh a Eric Feron. "Real-Time Motion Planning for Agile Autonomous Vehicles". In: *Journal of Guidance, Control, and Dynamics* 25.1 (2002), s. 116–129. DOI: 10.2514/2.4856. eprint: https://doi.org/10.2514/2.4856. URL: https://doi.org/10.2514/2.4856.

[15] Roland Geraerts a Mark H. Overmars. "A Comparative Study of Probabilistic Roadmap Planners". In: *Algorithmic Foundations of Robotics V*. Ed. Jean-Daniel Boissonnat et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, s. 43–57. ISBN: 978-3-540-45058-0. DOI: 10.1007/978-3-540-45058-0_4. URL: https://doi.org/10.1007/978-3-540-45058-0_4.

[16] Héctor H González-Banos, David Hsu a Jean-Claude Latombe. "Motion planning: Recent developments". In: *Autonomous Mobile Robots: Sensing, Control, Decision-Making and Applications* (2006).

[17] Corrado Guarino Lo Bianco a Aurelio Piazzi. "A hybrid algorithm for infinitely constrained optimization". In: *International Journal of Systems Science - IJSySc* 32 (led. 2001), s. 91–102. DOI: 10.1080/00207720150210878.

[18] Steffen Heinrich. *Planning universal on-road driving strategies for automated vehicles*. New York, NY: Springer Berlin Heidelberg, 2018. ISBN: 9783658219543.

[19] Robin Hess, Florian Kempf a Klaus Schilling. "Trajectory Planning for Car-Like Robots using Rapidly Exploring Random Trees*". In: *IFAC Proceedings Volumes* 46.29 (2013). 3rd IFAC Symposium on Telematics Applications, s. 44–49. ISSN: 1474-6670. DOI: https://doi.org/10.3182/20131111-3-KR-2043.00018. URL: http://www.sciencedirect.com/science/article/pii/S1474667015343603.

[20] David Hsu et al. "Randomized Kinodynamic Motion Planning with Moving Obstacles". In: *The International Journal of Robotics Research* 21.3 (2002), s. 233–255. DOI: 10.1177/027836402320556421. eprint: https://doi.org/10.1177/027836402320556421. URL: https://doi.org/10.1177/027836402320556421.

[21] Bernard Chazelle. "Approximation and decomposition of shapes". In: *Algorithmic and Geometric Aspects of Robotics* 1 (1985), s. 145–185.

[22] Reza N. Jazar. *Vehicle dynamics. theory and application*. 2nd ed. New York: Springer, 2014. ISBN: 978-1-4614-8543-8.

[23] Nitin R. Kapania a J. Christian Gerdes. "Design of a feedback-feedforward steering controller for accurate path tracking and stability at the limits of handling". In: *Vehicle System Dynamics* 53.12 (2015), s. 1687–1704. DOI: 10.1080/00423114.2015.1055279. eprint: https://doi.org/10.1080/00423114.2015.1055279. URL: https://doi.org/10.1080/00423114.2015.1055279.

[24] Sertac Karaman a Emilio Frazzoli. *Sampling-based Algorithms for Optimal Motion Planning*. 2011. arXiv: 1105.1186 [cs.RO].

[25] L. E. Kavraki, M. N. Kolountzakis a J. -. Latombe. "Analysis of probabilistic roadmaps for path planning". In: *IEEE Transactions on Robotics and Automation* 14.1 (1998), s. 166–171.

[26] Lydia E Kavraki et al. "Probabilistic roadmaps for path planning in high-dimensional configuration spaces". In: *IEEE transactions on Robotics and Automation* 12.4 (1996), s. 566–580.

[27] Alonzo Kelly a Bryan Nagy. "Reactive Nonholonomic Trajectory Generation via Parametric Optimal Control". In: *The International Journal of Robotics Research* 22.8 (čvc 2003), s. 583–601.

[28] Oussama Khatib. "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots". In: *Autonomous Robot Vehicles*. Ed. Ingemar J. Cox a Gordon T. Wilfong. New York, NY: Springer New York, 1990, s. 396–404. DOI: 10.1007/978-1-4613-8997-2_29. URL: https://doi.org/10.1007/978-1-4613-8997-2_29.

[29] Jason Kong et al. "Kinematic and dynamic vehicle models for autonomous driving control design". In: *2015 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2015, s. 1094–1099. ISBN: 978-1-4673-7266-4. DOI: 10.1109/IVS.2015.7225830. URL: http://ieeexplore.ieee.org/document/7225830/ (cit. 31. 03. 2020).

[30] James J Kuffner a Steven M LaValle. "RRT-connect: An efficient approach to single-query path planning". In: *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*. Sv. 2. IEEE. 2000, s. 995–1001.

[31] K. J. Kyriakopoulos, P. Kakambouras a N. J. Krikelis. "Potential fields for nonholonomic vehicles". In: *Proceedings of Tenth International Symposium on Intelligent Control*. 1995, s. 461–465.

[32] Jean-Claude Latombe. *Robot Motion Planning*. USA: Kluwer Academic Publishers, 1991. ISBN: 079239206X.

[33] Jean-Paul Laumond. *Robot motion planning and control*. New York: Springer, 1998. ISBN: 3540762191.

[34] Steven M. Lavalle. *Rapidly-Exploring Random Trees: A New Tool for Path Planning*. Tech. zpr. 1998.

[35] Steven M. LaValle a Jr. James J. Kuffner. "Randomized Kinodynamic Planning". In: *The International Journal of Robotics Research* 20.5 (2001), s. 378–400. DOI: 10.1177/02783640122067453. eprint: https://doi.org/10.1177/02783640122067453. URL: https://doi.org/10.1177/02783640122067453.

[36] Steven Michael LaValle. *Planning algorithms*. New York: Cambridge University Press, 2006. ISBN: 978-0521862059.

[37] JH Lee et al. "A Passive Multiple Trailer System with Off-axle Hitching". In: *International Journal of Control Automation and Systems* 2 (zář. 2004), s. 289–297.

[38] John M Lee. *Introduction to Smooth Manifolds*. Springer, 2013. ISBN: 978-1-4419-9981-8.

[39] Steve Levine. "Sampling-based Planning Algorithms Applied to Bicycles". In: ().

[40] Ming Lin a Dinesh Manocha. "Collision And Proximity Queries". In: *Handbook of Discrete and Computational Geometry, Third Edition* (lis. 2003). DOI: 10.1201/b10636-12.

[41] Stephen R. Lindemann a Steven M. LaValle. "Current Issues in Sampling-Based Motion Planning". In: *Robotics Research. The Eleventh International Symposium*. Ed. Paolo Dario a Raja Chatila. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, s. 36–54. ISBN: 978-3-540-31508-7.

[42] Tomás Lozano-Pérez. "Spatial Planning: A Configuration Space Approach". In: *Autonomous Robot Vehicles*. Ed. Ingemar J. Cox a Gordon T. Wilfong. New York, NY: Springer New York, 1990, s. 259–271. ISBN: 978-1-4613-8997-2. DOI: 10.1007/978-1-4613-8997-2_20. URL: https://doi.org/10.1007/978-1-4613-8997-2_20.

[43] Tomás Lozano-Pérez a Michael A. Wesley. "An Algorithm for Planning Collision-Free Paths among Polyhedral Obstacles". In: *Commun. ACM* 22.10 (říj. 1979), s. 560–570. ISSN: 0001-0782. DOI: 10.1145/359156.359164. URL: https://doi.org/10.1145/359156.359164.

[44] Javier Minguez, Florent Lamiraux a Jean-Paul Laumond. "Motion Planning and Obstacle Avoidance". In: led. 2008, s. 827–852. DOI: 10.1007/978-3-540-30301-5_36.

[45] Kourosh Naderi, Joose Rajamäki a Perttu Hämäläinen. "RT-RRT*: a real-time path planning algorithm based on RRT*". In: lis. 2015, s. 113–118. DOI: 10.1145/2822013.2822036.

[46] Jia Pan a Dinesh Manocha. "Efficient Configuration Space Construction and Optimization for Motion Planning". In: *Engineering* 1.1 (2015), s. 046–057. ISSN: 2095-8099. DOI: https://doi.org/10.15302/J-ENG-2015009. URL: http://www.sciencedirect.com/science/article/pii/S2095809916300443.

[47] David J Pearce a Paul HJ Kelly. "A dynamic topological sort algorithm for directed acyclic graphs". In: *Journal of Experimental Algorithmics (JEA)* 11 (2007), s. 1–7.

[48] Aurelio Piazzi a Corrado Guarino Lo Bianco. "Quintic G2-splines for trajectory planning of autonomous vehicles". In: ún. 2000, s. 198–203. ISBN: 0-7803-6363-9. DOI: 10.1109/IVS.2000.898341.

[49] Aurelio Piazzi et al. "Quintic G2-Splines for the Iterative Steering of Vision-Based Autonomous Vehicles". In: *Intelligent Transportation Systems, IEEE Transactions on* 3 (dub. 2002), s. 27–36. DOI: 10.1109/6979.994793.

[50] Les Piegl a Wayne Tiller. *The NURBS Book (2nd Ed.)* Berlin, Heidelberg: Springer-Verlag, 1997. ISBN: 3540615458.

[51] Mihail Pivtoraiko a Alonzo Kelly. "Efficient constrained path planning via search in state lattices". In:

[52] Philip Polack et al. "The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles?" In: červ. 2017, s. 812–818. DOI: 10.1109/IVS.2017.7995816.

[53] *Quad Tree*. 2012. URL: https://www.astroml.org/book_figures_1ed/chapter2/fig_quadtree_example.html.

[54] Abhijeet Ravankar et al. "Path smoothing techniques in robot navigation: State-of-the-art, current and future challenges". In: *Sensors* 18.9 (2018), s. 3170.

[55] J. A. Reeds a L. A. Shepp. "Optimal paths for a car that goes both forwards and backwards." In: *Pacific J. Math.* 145.2 (1990), s. 367–393. URL: https://projecteuclid.org:443/euclid.pjm/1102645450.

[56] Anthony Stentz. "Optimal and efficient path planning for partially known environments". In: *Intelligent Unmanned Ground Vehicles*. Springer, 1997, s. 203–220.

[57] Anthony Stentz. "The Focussed D* Algorithm for Real-Time Replanning". In: *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*. IJCAI'95. Montreal, Quebec, Canada: Morgan Kaufmann Publishers Inc., 1995, s. 1652–1659. ISBN: 1558603638.

[58] Hector J. Sussmann a Guoqing Tang. "Shortest paths for the Reeds-Shepp car: A worked out example of the use of geometric techniques in nonlinear optimal control". In: *Rutgers Univ., Tech. Rep. SYNCON* (1991).

[59] Petr Svestka a Mark Overmars. "Motion Planning for Carlike Robots Using a Probabilistic Learning Approach". In: *I. J. Robotic Res.* 16 (dub. 1997), s. 119–143. DOI: 10.1177/027836499701600201.

[60] Horatiu George Todoran. "Optimal Local Path-Planning and Control for Mobile Robotics". Dis. Ún. 2018.

[61] Tomáš Werner. "Optimalizace. Elektronická skripta předmětu B0B33OPT". Praha, 2020. URL: https://cw.fel.cvut.cz/old/_media/courses/a4b33opt/opt.pdf.

[62] Julius Ziegler a Christoph Stiller. "Fast collision checking for intelligent vehicle motion planning". In: *2010 IEEE Intelligent Vehicles Symposium*. IEEE. 2010, s. 518–522.

[63] Julius Ziegler a Christoph Stiller. "Spatiotemporal state lattices for fast trajectory planning in dynamic on-road driving scenarios". In: lis. 2009, s. 1879–1884. DOI: 10.1109/IROS.2009.5354448.

Příloha B

Seznam použitých symbolů a zkratek

Symbol	Význam
$\mathbb{N}$	přirozená čísla
$\mathbb{Z}$	celá čísla
$\mathbb{Q}$	racionální čísla
$\mathbb{R}$	reálná čísla
$\mathbb{R}_{\geq 0}$	nezáporná reálná čísla
$\mathbb{R}_{>0}$	kladná reálná čísla
$\mathbb{S}$	kružnice
$\oplus$	Minkovského suma
$\propto$	přímo úměrné
$\approx$	přibližně rovno
$\neq$	není rovno
$\equiv$	definujeme
$\ll$	řádově menší (v bázi deset)
$\mathrm{cl}(\cdot)$	uzávěr množiny M
$R_{(\alpha)}(\cdot)$	rotace v rovině o úhel α
$\mathscr{A}$	robot
$\mathscr{W}$	svět (prostředí, ve kterém se robot vyskytuje)
$\mathscr{O}$	oblast překážek (ve světě)
$\mathscr{F}$	volný prostor (ve světě)
C	konfigurační prostor
C_{obs}	oblast překážek (v konfiguračním prostoru)
C_{free}	volný prostor (v konfiguračním prostoru)
$\Delta\,\cdot$	diference
$\nabla_x\,\cdot$	gradient dle souřadnice x
$O(\cdot)$	asymptotická časová složitost
log	přirozený logaritmus

Zkratka	Význam
C-space	konfigurační prostor
NP	nedeterministicky polynomiální (třída složitosti)
AFP	artifical potential field
PRM	probabilistic road map
RRT	rapidly exploring random trees
S&C lattice	state and control lattice

Příloha C

Zdrojový kód

```
root
├── Simple_car
│   ├── readme.txt
│   ├── run_computation.sh
│   ├── C
│   ├── Py
│   └── Data
├── Trailers
│   ├── readme.txt
│   ├── run_computation.sh
│   ├── C
│   ├── Py
│   └── Data
└── libs
    └── c-algorithms-master.zip
C
├── Makefile
├── main.c
├── load_scripts.c
├── load_scripts.h
├── time_space_lattice.c
├── time_space_lattice.h
└── params.h
```

```
Py
├── opt_functions
│   ├── __pycache__
│   ├── __init__.py
│   ├── confun.py
│   └── objfun.py
├── __pycache__
├── collision_avoidance.py
├── create_canonical_set.py
├── create_obstacles.py
├── ctrl_models.py
├── get_final_path.py
├── minimize_acceleration.py
├── quintic_splines.py
├── set_params.py
└── spacetime_config.py
```

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Hruška** Jméno: **Antonín** Osobní číslo: **474562**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra řídicí techniky**

Studijní program: **Kybernetika a robotika**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Plánování trajektorie pro autonomní vozidla

Název bakalářské práce anglicky:

Trajectory planning for autonomous vehicles

Pokyny pro vypracování:

1) Seznámit se s problematikou plánování trajektorií pro různé typy autonomních vozidel, seznámit se s různými modely reprezentace vozidel a stavového prostoru.
2) Pro konkrétní úlohu – plánování trajektorie autonomního systému vozíků pro milk-run při výrobní lince – vybrat funkční model a navrhnout algoritmus plánování trajektorie.
3) Vybraný model spolu s plánovačem implementovat.
4) Ověřit funkčnost celého systému na zadaných instancích a vyhodnotit jeho použitelnost a případné rezervy.

Seznam doporučené literatury:

[1] Steven M. LaValle - Planning Algorithms, Cambridge University Press, (2006)
[2] Heinrich Steffen - Planning Universal On-Road Driving Strategies for Automated Vehicles. AutoUni – Schriftenreihe, vol 119. Springer, Wiesbaden, (2018)
[3] Ó'Dúnlaing, Colm & Yap, Chee - A "Retraction" Method for Planning the Motion of a Disc, J. Algorithms. 6. 104-111. 10.1016/0196-6774(85)90021-5. (1985)

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Mgr. Matěj Novotný, katedra matematiky FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **10.01.2020** Termín odevzdání bakalářské práce: **22.05.2020**

Platnost zadání bakalářské práce:
do konce letního semestru 2020/2021

Mgr. Matěj Novotný	prof. Ing. Michael Šebek, DrSc.	prof. Mgr. Petr Páta, Ph.D.
podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. Datum převzetí zadání

Podpis studenta