# Masterproef Semi-Autonomous Mobile Robot Navigation in Populated Envirments

Kevin Denis

25 november 2016

## Doelstelling masterproef

Figuur 1a: Een niet te drukke situatie in Brussel Noord, maar nog steeds dynamisch. Eigen foto.

Figuur 1b: Een drukkere omgeving, deze keer in Brussel Centraal. Door Fabian318 (wikipedia)

De doelstelling van deze masterproef is om een lokale padplanner te ontwikkelen die paden genereert in een dynamische omgeving[1]. Voorbeelden van zulke omgevingen zijn weergegeven in fig. 1a en 1b. In beide omgevingen moet de padplanner ook de dynamiek van zijn omgeving kunnen schatten om botsingsvrije pad op te kunnen stellen. Deze complexe lokale paden bestaan uit vierde en vijfde orde Bézierkrommes. Het voorgesteld algoritme is een uitbreiding van een al bestaande padplanner die enkel circulaire paden genereert. Deze schiet tekort in sommige situaties, b.v.b. wanneer de rolstoel naast een deur is. Dit algoritme heeft als hoofddoel om toegepast te kunnen worden op een semi-autonome rolstoelen, welke bijkomende vereisten met zich meebrengt. Er moet niet één mogelijk pad uit dit algoritme komen maar meerdere. Dit, omdat deze gegenereerde paden als hypothesen gebruikt moeten worden voor de navigatie-intentie van de gebruiker. Het algoritme moet snel werken, (doel : 5 Hz) zodat de gebruiker niet merkt dat er vertraging is tussen zijn commando en het uitvoeren daarvan. Uiteindelijk moet het algoritme op een multi-resolutie niveau werken, deze moet op korte afstand de invloed van het draaien van het zwenkwiel in rekening brengen en de dynamische randvoorwaarden van van de rolstoel zelf (maximale versnelling/vertraging, maximale kromming van pad, enz.).

### Objectieven

1. Ontwerpen van een lokale padplanner gebaseerd op Bézierkromme, geformuleerd als een COP.
2. Deze padplanner moet dynamische obstakels kunnen ontwijken.
3. Deze padplanner moet de dynamische beperkingen van de rolstoel in rekening brengen.
4. Deze padplanner moet snel genoeg uitgevoerd worden zodat de gebruiken geen last heeft van vertraging. Doel : 5Hz.

[1] Deze paden moeten "veilig" zijn voor de gebruiker en de personen om zich heen. Een mogelijke interpretatie hiervan is *sociaal aanvaardbaar* rij-gedrag. Daarmee wordt er bedoelt dat in een drukke station, het aanvaardbaar is om lichtjes tegen iemand te botsen. De implementatie hiervan is natuurlijk een uitdaging op zich.

# Week 0

Hierin is het werk gemaakt in de weken voor het begin van de academiejaar gebundeld.

## Doel

- Concreet doel voor thesis tegen eind september.
- Literatuur studie over het implementeren van splines in pad planning.
- Leren werken met gitlab.

## Bereikt

- Concreet doel voor thesis tegen eind september.

  Eerste visie is neergeschreven. Deze zal tijdens week 1 verfijnt worden.

- Literatuur studie over het implementeren van splines in pad planning.

  Methodes voor het afvlakken van paden die op een andere manier berekend zijn.

  - Bezier Curves : curve gaat door begin en eindpunt. Raaklijn begin en eindpunt hangt af van punt 2 en n-1. Dit is handig, want ik ben volledig vrij in het bepalen van het begin en eind oriëntatie van de curve. Begin = huidige pose, eind = gewenste pose.
  - Splines : elke controle punt heeft hetzelfde gewicht. Graad bepaald continuïteit van de punten. Er zou makkelijk gekozen kunnen worden voor het gebruik van cubische splines. Deze zorgen ervoor dat de functie in elk punt continue is tot de 2de afgeleide (pad is C2 continu). Er is natuurlijk een duidelijk verschil tussen heb gebruik maken van spline methodes of af te vlakken en om te interpoleren.
  - NURBS : cfr. splines maar controle punt heeft een vrij te kiezen gewicht

## Opmerkingen voor volgende week

- Benchmark maken om meerdere methodes (Bezier Curves, Splines en NURBS) uit te testen, zodat werking, parameters, vrijheid en rekentijd gekend zijn.
- Leren werken met gitlab.

## Wat ik heb gelezen

- „Design and Evaluation of a Lookup-Table Based Collision-Checking Approach for Fixed Sets of Mobile Robot Paths" [8]
- „Manipulaiting B-Spline Based Paths for Obstacle Avoidance in Autonomous Ground Vehicles" [7]
- „Analysis of a Spline Based, Obstacle Avoiding Path Planning Algorithm" [6]
- „Path Planning Based on Bézier Curve for Autonomous Ground Vehicles" [5]
- „Curvature-Continuous Trajectory Generation with Corridor Constraint for Autonomous Ground Vehicles" [4]
- „Continuous Path Smoothing for Car-Like Robots Using B-Spline Curves" [9]
- „Implementation of the Spline Method for Mobile Robot Path Control" [10]
- „Optimal Trajectory Generation for Car-Type Mobile Robot Using Spline Interpolation" [14]

# Week 1 - 2

## Doel

- Opmerkingen van Meneer Bruyninckx toevoegen aan huidig visie doelstelling masterproef
- Een Configuration-Space implementatie in MATLAB uitwerken.
- Verder uitwerking van het stuksgewijs toevoegen van Bézier Curves, en graad van continuiteit garanderen.

## Bereikt

- Opmerkingen van Meneer Bruyninckx toevoegen aan huidig visie doelstelling masterproef
- Verder uitwerking van het stuksgewijs toevoegen van Bézier Curves.

## Cubic Spline Interpolation

**Pro's**

- Polynoom gaat door alle gegeven punten.
- Oriëntatie kan op een indirecte manier gegeven worden, door de waarde van de helling van de begin en eindpunt op te geve
- Toevoegen van punt zal enkel voor een lokale verandering zorgen (groot voordeel tegenover bvb Lagrange Interpolatie)

**Con's**

- Werkt enkel met stijgende waarde van x.
- Spline kan niet evenwijdig met x-as beginnen en evenwijdig met y as eindigen

Mogelijke oplossing voor deze problemen : Door spline in meerdere delen te berekenen en het lokaal assenstelsel te roteren kan dit probleem opgelost worden

## Cubic Spline Approximation

**Pro's**

- Op eerste zicht niet logisch, maar door grote gewichten op begin en eindpunt te zetten zorgt men voor paden die door begin en eindpunt gaan
- Pad zal kleinere krommingen bevatten, comfortabeler voor de passagier
- Door met de interpoleer gewichten variëren creëert men licht verschillende paden.

**Con's**

- Zelfde opmerkingen als bij Cubic Spline Interpolation

## Bézier curves

### Pro's

Cubic Bézier Curve lijkt een ideale oplossing, omdat :

- Helling curve (dus oriëntatie robot) wordt bepaald door punt 2 en n-1.
- Bézier curves kunnen ook aan elkaar bevestigd worden. Voorbeeld algoritme: De Casteljau Construction ("Bézier Spline").
- Wel ervoor zorgen dat G-2 en C-2 continue is (zeker wanneer curve uit meerdere Bézier curves opbouw)

### OPM

- Indien ik C2 continu Bézier curves aan elkaar toe wil voegen, en nog steeds vrij wil zijn voor positie en oriëntatie van begin en eindpunt, MOET ik 5de orde (4de graad) Bézier curves gebruiken.

Maar, moet dit echt? Zou een controller niet makkelijk deze niet continue versnelling op kunnen vangen?

## Controle punten

Tot nu toe heb ik niet het probleem bekeken van waar ik de punten krijg, die ik gebruik als controle punt voor de splines. Deze zouden, onder andere kunnen komen van een globale pad planner. Kleine variaties op deze punten en orientaties zullen er voor zorgen dat ik meerdere paden kan maken
Een andere mogelijkheid zou zijn dat ik enkel de huidige positie en oriëntatie, doel en de free-space heb, en hiermee paden moet genereren. Dit laat veel meer vrijheid toe voor het genereren van paden

## Concrete vragen

1. Mijn dilemma is het volgende : moet ik mij enkel in Bézier curves verdiepen, of proberen een twee oplossing te implementeren, dus zowel Cubic Spline Approximation en Bézier curves ?
2. Is C-2 continu eigenlijk een vereiste? Of is G-2 continu genoeg? Dit zou overeenkomen als een "lichte"storing voor de controller; versnelling in dezelfde richting maar niet zelfde amplitude
3. Meneer Bruyninckx, zou u de term "dichtheid van paden"kunnen uitleggen?

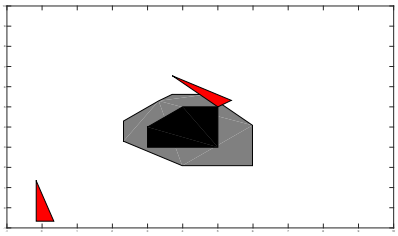
## Opmerkingen voor volgende week

- Een concrete methode uitwerken voor het genereren van punten als input voor de locale pad planner.
- Invloed stad van castor wielen op pad
- Soccer robots van Eindhoven University of Technology bekijken.
- Planning Algorithms van Steven M. LaValle doornemen
- Doornemen masterproef van Karel Belaen (gebruik van splines voor het afvlakken van paden)

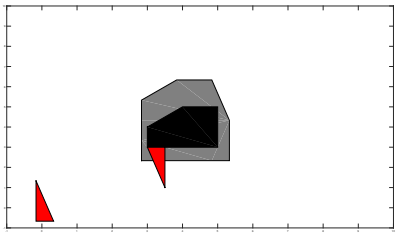
# Week 3

## Bereikt

- Een Configuration-Space implementatie in MATLAB.
  IN: (convex) polygon robot en (convex) polygon obstakel UIT : mogelijke botsingsvrije "posen" die de robot kan aannemen. Methode toegepast van Computational Geometry: Algorithms and Applications (Minkowski Sum en convhull (matlab functie)). *Was zeker geen vereiste, maar ik vond het gewoon interessant*.
- Verschillende manieren Bézier spline te berekenen met als gedachte zo snel mogelijke berekening, nodig voor implementatie. *Ook vrij vroeg, maar belangrijk in mijn ogen, zodat ik weet op welke verschillende manieren zo'n curve berekend kunnen worden en hun rekentijd*.
- Snelle lectuur van discrete padplanners (LaValle).
- Eerste draft van algoritme doorsturen en bespreken voor de implementatie in MATLAB.



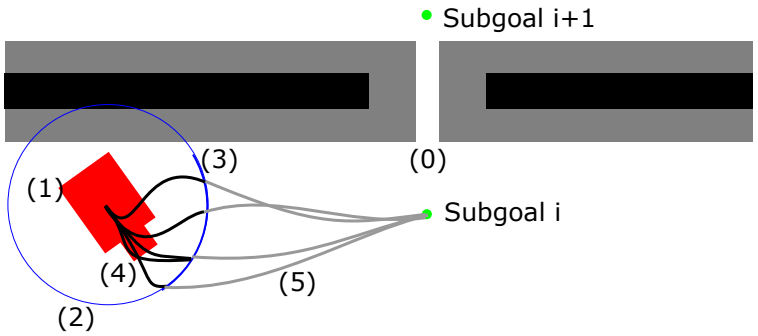
Figuur 2a: Rood: robot. Zwart : Obstakel. Grijs : Obstacle Space, deze is zowel afhankelijk van $(x, y, \theta)$



Figuur 2b: Rood: robot. Zwart : Obstakel. Grijs : Obstacle Space, deze is zowel afhankelijk van $(x, y, \theta)$

## Algoritme voor locale padplanner

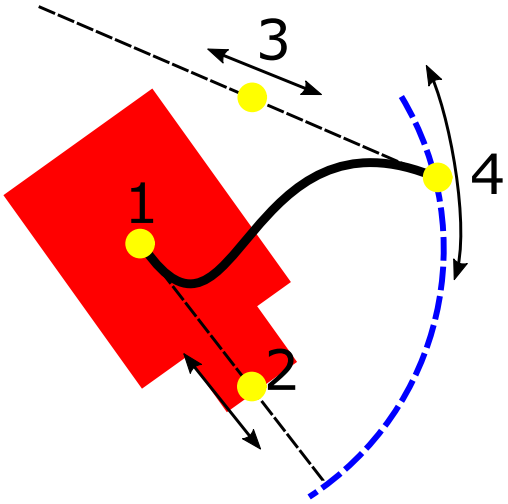
### Overzicht




Figuur 3: Overzicht locale padplanner. Rood: robot. Zwart : Obstakel en locale paden. Grijs : 2de Bézier curve (eindput 1ste curve tot subgoal) en Obstacle Space, deze is zowel afhankelijk van $(x, y, \theta)$ (deze is op deze figuur niet correct, maar enkel toegevoegd ter illustratie).

0. Het algoritme verwacht de configuration space waarin de robot zich kan verplaatsen $(x, y, \theta)$, zodat deze kan nakijken of het gegenereerde paden botsingsvrij is. Er moet ook een subgoal zijn waar de robot zich naar toe moet verplaatsen. Eens dat de robot dicht genoeg is bij deze subgoal, moet deze naar de volgende gaan.
1. Huidige positie en oriëntatie van de robot $(x, y, \theta)$.
2. Een cirkel rond de robot legt de maximale afstand die de splines mogen afleggen vergelen met de huidige positie.
3. Enkel een bepaald deel van de cirkel (vet blauw) zal uiteindelijk gebruikt mogen worden waar de eindpunten van de splines zich zullen bevinden.
4. Splines worden gegenereerd met als startpunt de positie van de robot en zijn oriëntatie. Er zijn dus beperkingen op de plaats van de controle punten voor de Bézier curves. Op dit moment word er met cubische Bézier curves, deze bestaan uit 4 controle punten. De eigenschappen van een Bézier curves zijn dat ze door punt 1 en n (=4) gaan, en de oriëntatie (1ste afgeleide) word bepaald door punt 2 en n-1 (=3). De volgende parameters zijn vast :
   (a) Plaats van punt 1 op actuele positie van robot.
   (b) Richtingscoëfficiënt tussen punten 1 en 2 (= actuele oriëntatie robot).
   (c) Richtingscoëfficiënt tussen punt 3 en 4 moet er voor zorgen dat oriëntatie robot op cirkel "meer" wijst naar subgoal als in zijn begin oriëntatie.
   (d) Plaats van punt 4 : op een deel van cirkel.

   De volgende parameters zijn vrij :
   (a) Punt 2 op de rechte bepaald door de oriëntatie en beginpunt van de robot.
   (b) Punt 3 op de rechte bepaald door de eind oriëntatie en punt 4.
   (c) Punt 4 op specifiek deel van de cirkel.
5. De berekende Bézier curves in (4) zullen natuurlijk met een bepaalde costfunctie (dicht bij user-intention, lage kromming, afstand, etc.) met elkaar vergeleken worden. Men zou ook de Bézier cruve kunnen genereren tot de subgoal (in het grijs) en daar informatie van gebruiken voor het overwegen van welke locale pad het beste is (afstand, kromming, etc.).




Figuur 4: Uitleg over de plaats van de vrije controle punten. Rood: robot. Zwart : locale pad gemaakt dankzij een Bézier Curve. Geel : ligging van controle punt voor deze specifieke Bézier curve. Stippellijnen duiden aan hoe controle punten 2, 3 en 4 mogelijk van plaats kunnen veranderen. De term resolutie betekend het aantal mogelijke liggingen op de rechte/kromme.

**Motivatie en specifieke uitleg voor positie controle punten**

**Waarom deze blauwe cirkel ?** Het idee van de cirkel kwam door de volgende reden : om een Bézier curve te kunnen maken zijn er controle punten nodig. Om er voor te zorgen dat men niet in het wilde weg controle punten begin te genereren en om het accent tot waar de controle punten zich kunnen bevinden, was de cirkel in mijn ogen een goede oplossing.

**Hoe kiest men nu deze deel van de cirkel die gebruikt wordt ?** Ik denk dat ik mij hier kan laten inspireren door de Vector Field Histogram (normal, + en *) van Borenstein en Koren, deze hebben een interessante implementatie voor het kiezen van vrije sectoren. Op deze manier zou ik dus vrije sectoren op de blauwe cirkel rond de robot kunnen kiezen.

**Waarom nu die 2de Bézier curve ?** Ik denk dat er uit deze 2de Bézier curve nuttige informatie kan gehaald worden. Het antwoord in een zekere zin de vraag *"Hoe ver (positie en orientatie) ben ik van het subgoal"*.

**En de resolutie ?** Er zijn verschillende parameters die een bepaalde resolutie kunnen hebben (de plaatsen van punten 2 en 3 op de rechte bepaald door respectievelijk punten 1 en 4) en de plaats van punt 4 op de cirkel.

## Opmerkingen voor volgende week

- Dynamische obstakel ontwijking literatuur, kijken hoe ik de ideeën kan implementeren in deze padplanner.
- Literatuur doornemen die in de opmerkingen van Meneer Demeester staat.
- Verder werken aan draft algoritme padplanner na feedback.
- Bijwerken van referentie bestand voor literatuur !!!

# Week 4-5

## Bereikt

- Lokale padplanner formuleren als een COP, gegeven begin- en eindpositie en oriëntatie, voor een Bézierkromme van orde $n-1$. $n$ is hier het aantal controlepunten.
- Eerste stappen in gitlab.

## Overzicht van padplanner

Illustratie van de padplanner geformuleerd als een COP in MATLAB, met CASADI [1] en OPTISTACK (dit is een front-end/wrapper van CASADI voor MATLAB).

- Parameter
  De parameter die wordt geoptimaliseerd is de positie van de $n$ controlepunten nodig voor het definiëren van de Bézierkromme van orde $n-1$.

$$z = [x_1, y_1, \ldots, x_n, y_n]^T \tag{1}$$

$$B(t) = BzierCurve(x, y, t) \tag{2}$$

  Met :

$$t = 0 : \Delta t : 1, \text{ dimensieloze parameter voor de constructie van de Bézierkromme} \tag{3}$$

- Kostfunctie (minimaliseren) (uit [3] [2])

$$f(z) = \int_0^1 \kappa(t)^2 dt + \alpha \int_0^1 s(t) dt \tag{4}$$

  Met :

$$\kappa(t) = \frac{B'(t) \times B''(t)}{\|B'(t)\|^3}, \text{ kromming} \tag{5}$$

$$s(t) = \int_0^1 \sqrt{B_x'^2 + B_y'^2} dt, \text{ lengte van de Bézierkromme} \tag{6}$$

$$\alpha, \text{ het relatieve gewicht tussen buiging-energie en lengte van de kromme}^3 \tag{7}$$

- Randvoorwaarden. *Voorlopig enkel geometrisch*.
  - Begin- en eindpunt liggen vast

$$[x_1, y_1] = [x1, y1] \tag{8}$$

$$[x_n, y_n] = [xn, yn] \tag{9}$$

  - Oriëntatie op begin- en eindpunt liggen vast

$$y_2 = \tan(\theta_1)(x_2 - x_1) \tag{10}$$

$$y_{end-1} = \tan(\theta_{end})(x_{end} - x_{end-1}) \tag{11}$$

[2] hier wordt er geïntegreerd over de dimensieloze parameter $t$. Niet te verwarren met de tijd. In [3] wordt er over $ds$ geïntegreerd, dit heb ik voorlopig nog niet geïmplementeerd.

[3] dit is niet helemaal correct, men moet hiervoor $\kappa(t)^2$ integreren over $s$ en niet over $t$.

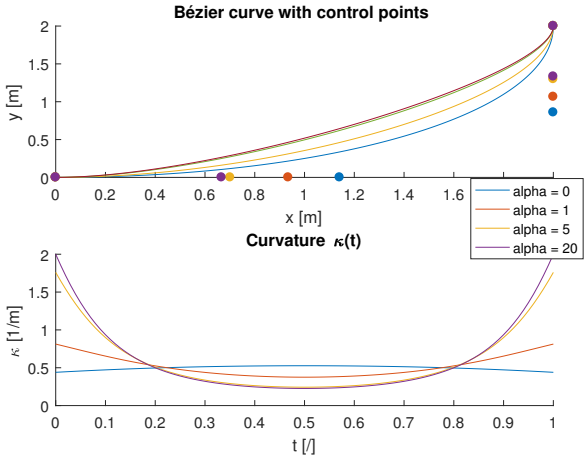
– Maximale toegelaten kromming

$$\kappa(t) \leqslant \kappa_{max} \tag{12}$$

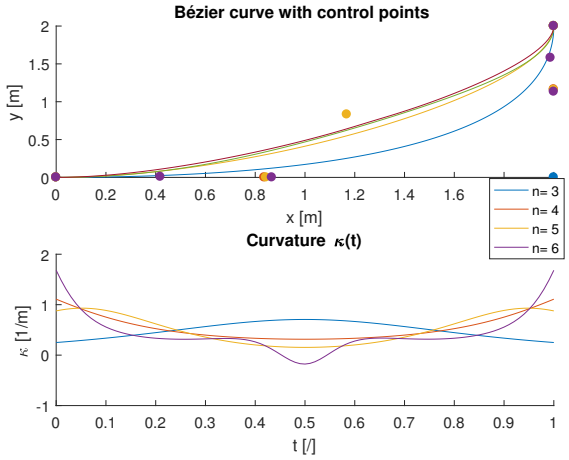
– controlepunten $2 : n-1$ moeten binnen de rechthoek blijven gedefinieerd door punten $1$ en $n$.

$$[x_1, y_1] <= [x_{2:n-1}, y_{2:n-1}] <= [x_n, y_n] \tag{13}$$

In fig. 5a en fig. 5b kan men figuren vinden van de optimalisatie resultaten met toenemende waarde van $\alpha$ en $n$ respectievelijk.




Figuur 5a: Optimalisatie met verschillende waardes voor $\alpha$. De kromming neemt toe en de lengte van de kromme daalt met toenemende waarde voor $\alpha$, totdat deze de maximale waarde $\kappa_{max} = 2$ bereikt.




Figuur 5b: Optimalisatie met verschillende waardes voor $n$. Zowel de buiging-energie als de lengte van de kromme dalen met het toenemen $n$. De rekentijd neemt wel toe met toenemende $n$.

## Wat ik heb gelezen

- „Path Planning for Mobile and Hyper-Redundant Robots Using Pythagorean Hodograph Curves" [3]
- „Real-Time Motion Planning in the Presence of Moving Obstacles" [12]
- „Optimal Trajectory Generation for Car-Type Mobile Robot Using Spline Interpolation" [14]
- *A Primer on Bézier Curves* [11]
- *Computer Aided Geometric Design* [13]

## Opmerkingen voor volgende week

- Obstakel ontwijking toevoegen [12] en referentie hierin (Hypervlak).
- Gitlab aanvullen met eventueel literatuur, meeting rapport en logboek.
- Update doelstellingen masterproef en 4 concrete meetbare objectieven.

# Week 6-7

## Bereikt

- Lokale padplanner formuleren als een COP, gegeven begin- en eindpositie en oriëntatie, voor een Bézierkromme van orde $n-1$. $n$ is hier het aantal controlepunten. Deze houdt nu rekening met één obstakel. Deze moet ook convex zijn. De methode steunt op de theorema van de scheidende hypervlakken, uitgelegd in [2].
- Update doelstellingen masterproef en 4 concrete meetbare objectieven.

## Overzicht van padplanner

Illustratie van de padplanner geformuleerd als een COP in MATLAB vgl. (14). De parameter die wordt geoptimaliseerd is de positie van de $n$ controlepunten nodig voor het definiëren van de Bézierkromme van orde $n-1$.

Voorlopig is de obstakel-ontwijker een zeer dure operatie (van 0.2s naar 1.2s uitvoertijd). Dit kom omdat in het optimaliseringsproces op elke punt van Bézierkromme controleert of deze ver genoeg van het obstakel is. In [12], is er een elegantere methode (en ook sneller) ontwikkeld. Deze methode is nog niet geïmplementeerd.

$$\underset{z}{\text{minimize}} \quad f(z) = \int_0^l \kappa(t)^2 ds + \alpha \int_0^1 s(t)dt \text{ (uit [3])} \tag{14a}$$

$$\text{subject to} \quad [x_1, y_1] = [x1, y1], \tag{14b}$$

$$[x_n, y_n] = [xn, yn], \tag{14c}$$

$$y_2 = \tan(\theta_1)(x_2 - x_1), \tag{14d}$$

$$y_{n-1} = \tan(\theta_n)(x_n - x_{n-1}), \tag{14e}$$

$$\kappa(t)^2 \leqslant \kappa_{max}^2, \tag{14f}$$

$$a^T B(t) \leqslant b \text{ (uit [2], zie ook fig. 6)}, \tag{14g}$$

$$lb_x \leqslant x_i \leqslant ub_x, \tag{14h}$$

$$lb_y \leqslant y_i \leqslant ub_y. \tag{14i}$$

Met :

$t = 0 : \Delta t : 1,$ dimensieloze parameter voor de constructie van de Bézierkromme

$z = [x_1, y_1, \ldots, x_n, y_n]^T$

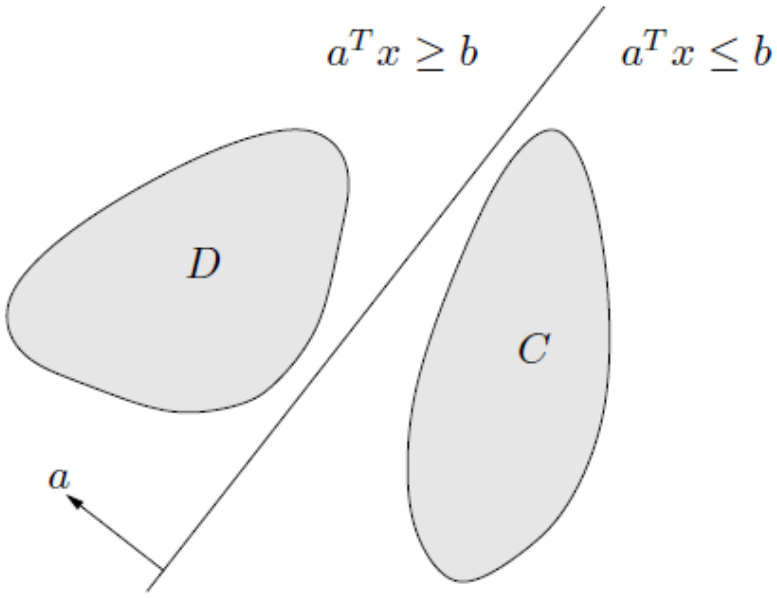
$B(t) = BezierCurve(x, y, t)$

$\kappa(t) = \dfrac{B'(t) \times B''(t)}{\|B'(t)\|^3},$ kromming

$s(t) = \displaystyle\int_0^1 \sqrt{B_x'^2 + B_y'^2} dt,$ lengte van de Bézierkromme $s(t=1) = l$

$\alpha,$ het relatieve gewicht tussen buiging-energie en lengte van de kromme

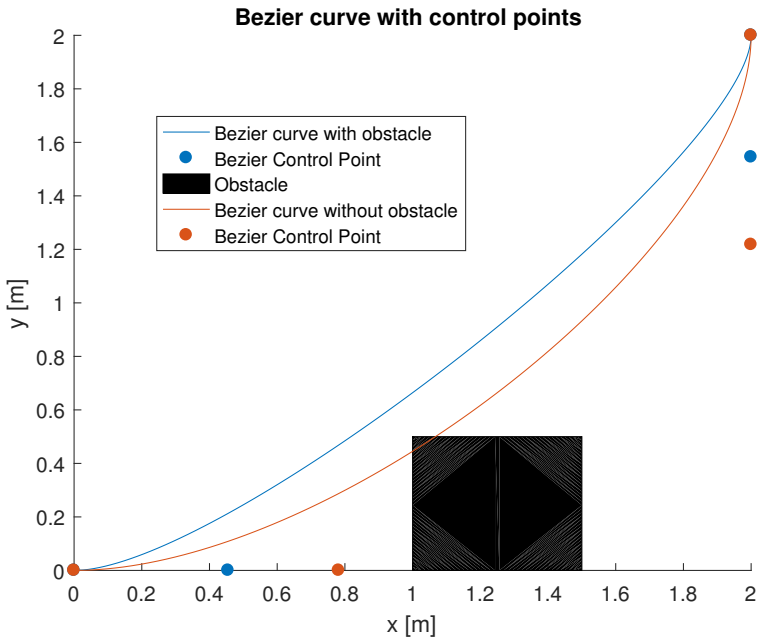
- Randvoorwaarden. *Voorlopig enkel geometrisch.*
  - Begin- en eindpunt liggen vast vgl. (14b) en (14c)
  - Oriëntatie op begin- en eindpunt liggen vast vgl. (14d) en (14e)
  - Maximale toegelaten kromming vgl. (14f)
  - Er moet een hypervlak getekend kunnen worden tussen elk punt van de kromme en het obstakel vgl. (14g).




Figuur 6: Tussen twee niet overlappende convexe sets C en D is er altijd een scheidende hypervlak, bepaald voor $a^T x = b$. $a$ is hier de vector bepaald door de twee dichtstbijzijnde punten in C en D. $b$ de offset van deze vlak. In deze toepassing is C de robot en is voorlopig tot een cirkel herleid. D is hier een obstakel gevormd door een convexe polygoon.

– Controlepunten moeten binnen een bepaalde grens blijven vgl. (14h) en (14i)

In fig. 7 kan men de resultaten van het optimaliseringsprobleem vinden met en zonder rekening te houden met een obstakel.




Figuur 7: Optimalisatie met en zonder rekening houden van het obstakel, respectievelijk in het blauw en rood.

## Wat ik heb gelezen

- *Convex Optimization* [2]
- „Real-Time Motion Planning in the Presence of Moving Obstacles" [12]

## Opmerkingen voor volgende week

- Dynamica toevoegen, dus ook "hoe gaat men het pad volgen"(feedforward, feedback, mogelijke inspiratie in [12] en case-studie *Optimization of Mechatronic Systems*).
- Van een cirkelvormige geometrie naar een convexe polynoom geometrie voor het obstakel-ontwijking.
- Afspraak maken met promotors en mentors.
- Gitlab aanvullen met eventueel literatuur, meeting rapport en logboek.

# Referenties


[1] Joel Andersson. *A General-Purpose Software Framework for Dynamic Optimization (Een Algemene Softwareomgeving Voor Dynamische Optimalisatie)*. Leuven, 24 okt 2013. 186 p. ISBN: 978-94-6018-750-6. URL: https://lirias.kuleuven.be/handle/123456789/418048 (bezocht op 29-10-2016).

[2] Stephen Boyd en Lieven Vandenberghe. *Convex Optimization*. Cambridge: Cambridge university press, 2004. 716 p. ISBN: 0-521-83378-7.

[3] H. Bruyninckx en D. Reynaerts. „Path Planning for Mobile and Hyper-Redundant Robots Using Pythagorean Hodograph Curves". In: *, 8th International Conference on Advanced Robotics, 1997. ICAR '97. Proceedings*. , 8th International Conference on Advanced Robotics, 1997. ICAR '97. Proceedings. Jul 1997, p. 595–600. DOI: 10.1109/ICAR.1997.620243.

[4] J. w Choi, R. E. Curry en G. H. Elkaim. „Curvature-Continuous Trajectory Generation with Corridor Constraint for Autonomous Ground Vehicles". In: *49th IEEE Conference on Decision and Control (CDC)*. 49th IEEE Conference on Decision and Control (CDC). Dec 2010, p. 7166–7171. DOI: 10.1109/CDC.2010.5718154.

[5] J. w Choi, R. Curry en G. Elkaim. „Path Planning Based on Bézier Curve for Autonomous Ground Vehicles". In: *World Congress on Engineering and Computer Science 2008, WCECS '08. Advances in Electrical and Electronics Engineering - IAENG Special Edition of the*. World Congress on Engineering and Computer Science 2008, WCECS '08. Advances in Electrical and Electronics Engineering - IAENG Special Edition of the. Okt 2008, p. 158–166. DOI: 10.1109/WCECS.2008.27.

[6] John Connors en Gabriel Elkaim. „Analysis of a Spline Based, Obstacle Avoiding Path Planning Algorithm". In: *2007 IEEE 65th Vehicular Technology Conference - VTC2007-Spring*. 2007 IEEE 65th Vehicular Technology Conference - VTC2007-Spring. biblatex: connors2007. San Diego, CA, apr 2007, p. 2565–2569. DOI: 10.1109/VETECS.2007.528.

[7] John Connors en Gabriel Elkaim. „Manipulaiting B-Spline Based Paths for Obstacle Avoidance in Autonomous Ground Vehicles". In: *Proceedings of the 2007 National Technical Meeting of The Institute of Navigation*. Proceedings of the 2007 National Technical Meeting of The Institute of Navigation. San Diego, CA, jan 2007, p. 1081–1088. URL: https://www.ion.org/publications/abstract.cfm?articleID=7204.

[8] Eric Demeester e.a. „Design and Evaluation of a Lookup-Table Based Collision-Checking Approach for Fixed Sets of Mobile Robot Paths". In: *International Symposium on Robotics* (2012).

[9] Mohamed Elbanhawi, Milan Simic en Reza N. Jazar. „Continuous Path Smoothing for Car-Like Robots Using B-Spline Curves". In: *J Intell Robot Syst* 80.1 (8 jan 2015), p. 23–56. ISSN: 0921-0296, 1573-0409. DOI: 10.1007/s10846-014-0172-0. URL: http://link.springer.com.kuleuven.ezproxy.kuleuven.be/article/10.1007/s10846-014-0172-0 (bezocht op 26-09-2016).

[10] H. Eren, Chun Che Fung en J. Evans. „Implementation of the Spline Method for Mobile Robot Path Control". In: *Proceedings of the 16th IEEE Instrumentation and Measurement Technology Conference, 1999. IMTC/99*. Proceedings of the 16th IEEE Instrumentation and Measurement Technology Conference, 1999. IMTC/99. Deel 2. 1999, 739–744 vol.2. DOI: 10.1109/IMTC.1999.776966.

[11] Mike Kamermans. *A Primer on Bézier Curves*. 13 jun 2013. URL: http://pomax.github.io/bezierinfo (bezocht op 30-10-2016).

[12] Tim Mercy, Wannes Van Loock en Goele Pipeleers. „Real-Time Motion Planning in the Presence of Moving Obstacles". In: Benelux Meeting on Systems and Control. Soesterberg, Netherlands, 24 mrt 2016.

[13] Thomas W. Sederberg. *Computer Aided Geometric Design*. 28 sep 2016. 288 p. URL: http://tom.cs.byu.edu/~557/text/cagd.pdf (bezocht op 30-10-2016).

[14] Rahee Walambe e.a. „Optimal Trajectory Generation for Car-Type Mobile Robot Using Spline Interpolation". In: *IFAC-PapersOnLine*. 4th IFAC Conference on Advances in Control and Optimization of Dynamical Systems ACODS 2016 49.1 (1 jan 2016), p. 601–606. ISSN: 2405-8963. DOI: 10.1016/j.ifacol.2016.03.121. URL: http://www.sciencedirect.com/science/article/pii/S2405896316301215 (bezocht op 26-09-2016).