Masterproef industriële ingenieurswetenschappen 2019-2020

Optimalisatie van het industrieel robotfreesproces door filteren en toevoegen van punten in het robotpad

Ward Habraken

Master IW energie

Jonas Nelis

Master IW energie

Situering

KUKA is een producent van **industriële robots** en ontwikkelt nieuwe technologieën die gebruik maken van deze robots. Deze masterproef gaat door in de vestiging in Houthalen, KUKA Automatisering + Robots N.V. Eén van deze technologieën wordt hier ontwikkeld, namelijk het **frezen** met behulp van een industriële robot. De robot volgt hierbij een pad van punten gebruikmakende van **lineaire bewegingen met luswerking** ofwel **spline-bewegingen**.

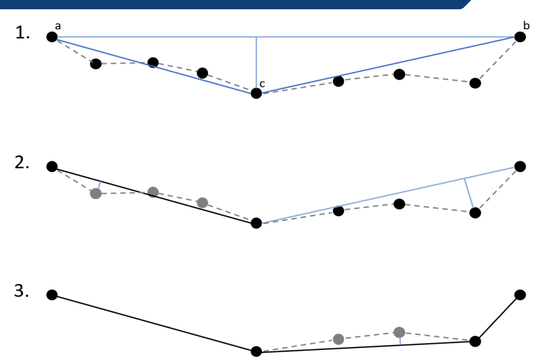
Figuur 1: Freesproces [1]

Probleem- en doelstelling

Het frezen met behulp van een robotarm, weergegeven in figuur 5, is een traag proces dat een hoge nauwkeurig vereist. Hierbij kan het proces versneld worden door niet cruciale punten te **filteren** die binnen een gegeven tolerantie liggen. Om de nauwkeurigheid na het filteren te behouden, moeten punten **toegevoegd** worden. Tijdens het filteren moet er niet alleen rekening gehouden worden met de stand van de robot maar ook met oriëntatie.

Hoe kan het freesproces versneld worden met behoud van nauwkeurigheid binnen een gegeven tolerantie?

Douglas-Peucker



Figuur 2: Werking Douglas-Peucker-algoritme

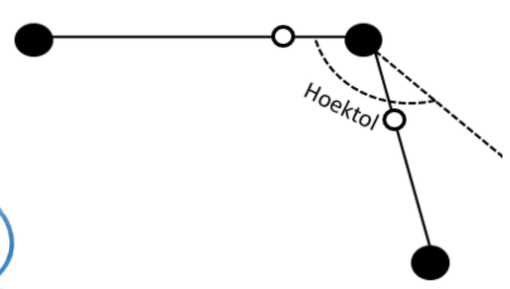
Het Douglas-Peucker-algoritme, weergegeven in figuur 2, wordt gebruikt om de **punten te filteren** met behulp van een meegegeven **tolerantie**.

1. Het begin- en eindpunt worden behouden. Het algoritme berekent de loodrechte afstand van de lijn *ab* tot de overige punten.
2. Indien de grootste afstand groter is dan tolerantie, wordt dit punt behouden. Vervolgens wordt opnieuw de afstand berekend, deze keer van de lijnen *ab* en *ac* tot de punten die hiertussen liggen.
3. Indien de grootste afstand kleiner is dan de tolerantie, mogen alle punten hiertussen verwijderd worden.

Hoekalgoritme

Het hoekalgoritme, weergegeven in figuur 3, wordt gebruikt om **afrondingen**, die door de luswerking of spline-beweging van de robot ontstaan, te **voorkomen**. Er zijn drie mogelijke situaties:

1. De hoek is kleiner dan de tolerantie → het punt voor en na het hoekpunt wordt bijgehouden.
2. De hoek is groter dan de tolerantie, de vorige en volgende hoek zijn kleiner dan de tolerantie → het punt voor en na het hoekpunt wordt bijgehouden.
3. De vorige hoek, de hoek en de volgende hoek zijn groter dan de tolerantie → er moet niets gebeuren



Figuur 3: Hoekalgoritme

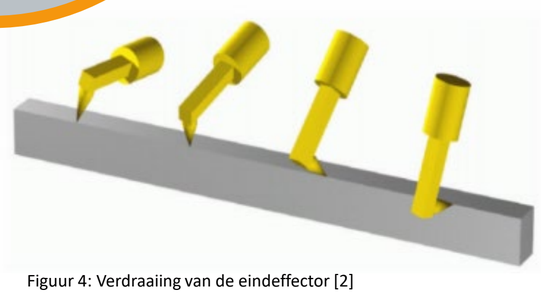
Algoritmes

Oriëntatie-algoritme

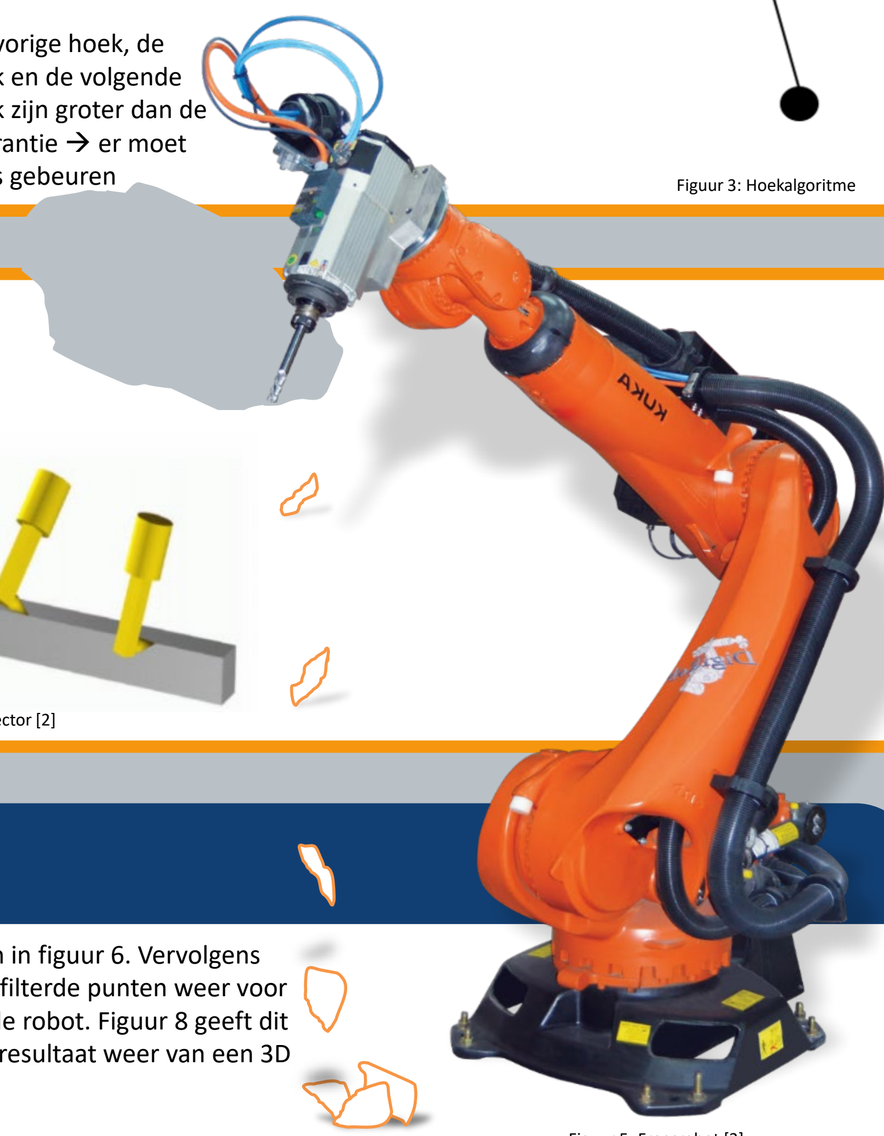
Het oriëntatie-algoritme wordt gebruikt om te **voorkomen** dat er een te **grote hoekverdraaiing** van de eindeffector voorkomt tussen opeenvolgende punten. Figuur 4 geeft de verdraaiing van de eindeffector weer.

Het algoritme werkt als volgt:

1. Het eerste punt wordt behouden.
2. De verdraaiing van de eindeffector wordt berekend tussen het eerste en het tweede punt.
3. Indien de verdraaiing kleiner is dan de tolerantie, wordt de verdraaiing tussen het eerste en derde punt berekend.
4. Dit proces wordt herhaald tot de verdraaiing groter is dan de tolerantie. Het punt, waarbij dit voorkomt, wordt behouden en vervolgens wordt de verdraaiing berekend tussen dit punt en de volgende punten.



Figuur 4: Verdraaiing van de eindeffector [2]

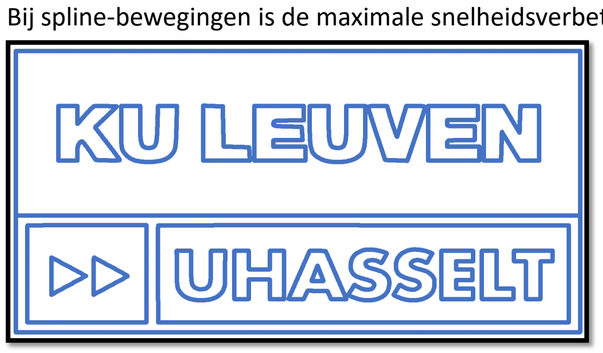


Figuur 5: Freesrobot [3]

Resultaat

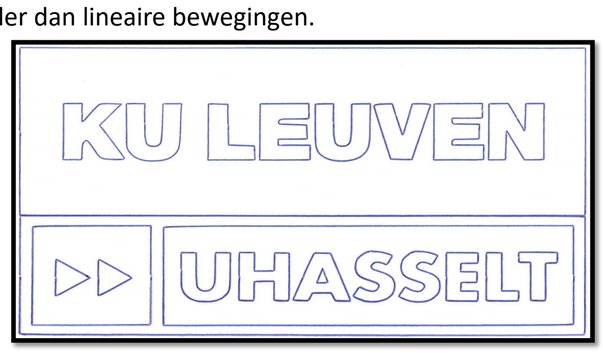
Het programma is geschreven in **C#** met behulp van Visual Studio. Als eerste worden er punten in **2D** gegenereerd zoals weergegeven in figuur 6. Vervolgens worden de punten gefilterd, dit kan zowel voor **lineaire** bewegingen met luswerking als voor **spline**-bewegingen. Figuur 7 geeft de gefilterde punten weer voor lineaire bewegingen. Hierna wordt er een pen bevestigd aan de freeskop en worden de gefilterde punten getekend met behulp van de robot. Figuur 8 geeft dit resultaat weer. Ten slotte worden vorige stappen herhaald voor punten in **3D** waarna de robot de figuur uitfreest. Figuur 9 geeft een resultaat weer van een 3D freestest.

Door gebruik te maken van het Douglas-Peucker-algoritme zal de nauwkeurigheid altijd binnen de gegeven tolerantie blijven. De snelheidsverbetering is afhankelijk van het aantal originele punten en de tolerantie. Met een tolerantie van 0,1 mm en lineaire bewegingen zijn **snelheidsverbeteringen** tot **74%** mogelijk. Bij spline-bewegingen is de maximale snelheidsverbetering slechts **34%**, maar spline-bewegingen zijn over het algemeen sneller dan lineaire bewegingen.



Figuur 6: Uhasselt en KU Leuven logo uitgezet in punten

Figuur 7: De punten van de logo's na het filteren



Figuur 8: 2D lineair en spline test

Figuur 9: 3D freestest

Promotoren / Copromotoren: Prof. dr. ir. Johan Baeten
Dr. ir. Wim Persoons

[1] WNT, "THE EVOLUTION IN TROCHOIDAL MILLING." https://www.wnt.com/en-cn/cutting-tools/news-detail/the-evolution-in-trochoidal-milling-1351.html (geopend 8 mei, 2020).
[2] KUKA, "KUKA System Software," vol. 7, p. 359, 2019.
[3] Metaradam. "Industrial Robot Machine KUKA Milling." https://www.hiclipart.com/free-transparent-background-png-clipart-mhjwn (geopend 12 oktober, 2019).