Hieronder een aanvulling door Joep Suijs naar aanleiding van overleg met Jan Heynen i.v.m. de presentie over odometrie

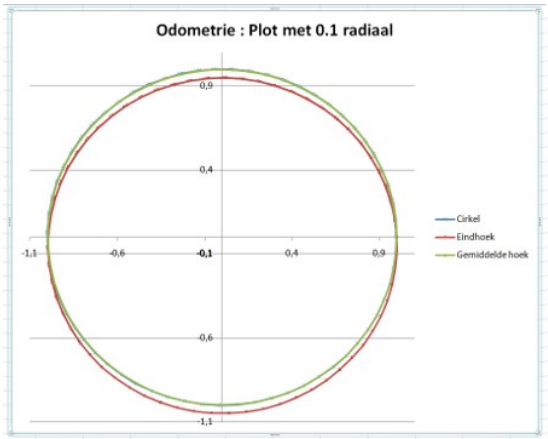
Bij de presentatie constateerde Jan dat de code de nieuwe positie bepaalt door eerst de hoek bij te werken (rotatie) en daarna XY bij te werken op basis van de afgelegde weg (translatie). Zijn methode is gebaseerd op het gemiddelde van de hoek voor en na rotatie, volgens een paper dat hij heeft gevonden.

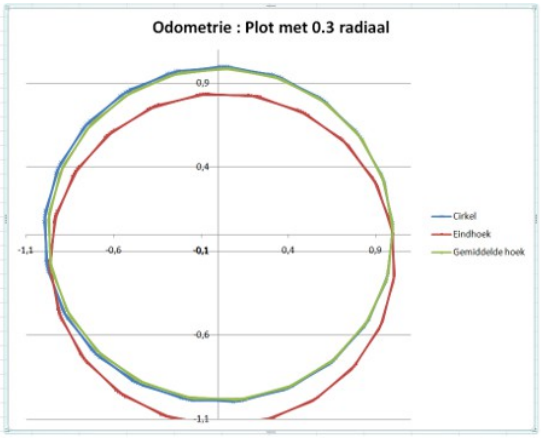
De ideale situatie is dat je boog berekent op basis van de translatie en rotatie samen, maar dat is tamelijk reken-intensief. Een rechte lijn (lijnstuik) is een vereenvoudiging hiervan. De richting van de lijn kan het gemiddelde van de begin- en eind-hoek zijn.
Ooit was dat ook mijn aanpak, maar op een bepaald moment was de hypothese dat dit niet nodig was, omdat de meerdere stappen achter elkaar de fout opheffen. Bij testen gaven beide methode geen verschil, dus sindsdien gebruikte ik de vereenvoudigde methode.

Naar aanleiding van de presentatie hebben Jan en ik beide een simulatie gemaakt van de 3 varianten (boog, gemiddelde hoek en eind-hoek) voor een robot die een cirkel rijdt. De conclusie is dat de methodes voor een hele cirkel geen verschil geven. En voor de boog vs gemiddelde hoek geldt dat de verschillen gedurende de hele cirkel verwaarloosbaar zijn.
Maar... met een kwart cirkel rijden is de afwijking van mijn vereenvoudiging duidelijk waarneembaar doordat de hele cirkel iets is gedraaid ten opzichte van het assenstelsel. In de praktijk kan de fout (voor een LimaPlanck) oplopen tot ruim 1 cm. Reden genoeg om dit aan te passen!

Hieronder de grafieken die Jan gemaakt heeft en een link naar de aanpassing van de code op Github.

Dank aan Jan voor de grafieken en - meer nog - voor het sparren over dit onderwerp. Want het heeft de odometrie weer wat nauwkeuriger gemaakt.
Groeten,
Joep





https://github.com/jsuijs/LiMaPlanck/commit/b9abdd68cdfdcf31dc792263a4e78814b38b30d3

Toelichting door Jan Heynen.

In de spreadsheet heb ik inderdaad 2 plots gemaakt, de eerste doet telkens een berekening als het hoekverschil 0,1 rad vedraagt, de 2de pas na 0,3 rad. In de praktijk hangt dit af van de resolutie van je encoders. Bij mij robot is dat 0,07 mm/tic. Ik kan dus makkelijk elke 0,1 rad een nieuwe pos. berekenen.