WoR-Robots Applicatie

HAN Arnhem

561399

MWJ.Berentsen@student.han.nl

Berentsen M.W.J.

6 april 2017

Inhoudsopgave

1 Inleiding

Voor de eindopdracht van WoR-Robots is er de opdracht om een applicatie te leveren die aan de hand van een simpele TUI (Text User Interface) items kan oppakken en op een opgegeven plek neerleggen. Om deze opdracht uit te kunnen voeren wordt ROS (Robot Operating System) gebruikt om de arm aan te sturen, en er wordt gebruik gemaakt van OpenCV voor de beeldherkenning. Alle code die door de student geschreven wordt zal bestaan uit C++11

2 Applicatie design

2.1 Het berekenen van de beweging van de robotarm

Om het pad te bereken die de robotarm moet afleggen moet er eerst een configuratie gevonden worden. Dit omdat de robot niet aangestuurd wordt met coördinaten maar hij de hoeken van de arm moet ontvangen. Allereerst zal er nu eerst verklaard worden hoe er van co/ördinaten naar een configuratie gerekend wordt.

2.1.1 Het berekenen van de nieuwe configuratie

Voor het berekenen van de configuratie word er eerst een twee dimensionaal punt bepaald vanaf het zijaanzicht. In het twee dimensionale stelsel wordt er gewerkt met coördinaten X en Y. Deze coödinaten zetten we om naar een configuratie met behulp van het volgende algoritme dit daarna per punt wordt behandeld.

while (e is too far from g and found θ_{nieuw} is within solutionspace)
Compute $J(e, \theta)$ for the current pose θ
Compute J^{-1}
$\Delta e = \beta(g - e)$
$\Delta\theta = J^{-}1 \times \Delta e$
$\theta_{nieuw} = \theta_{oud} + \Delta\theta$
Compute new e vector

1. while (e is too far from g)and found θ_{nieuw} is within solutionspace)
   Zolang het gevonden eindpunt e nog niet ongeveer gelijk is aan het doel g wordt het volgende uitgevoerd.

Voor het berekenen van de configuratie naar een x,y coördinaat wordt het forward kinematics gebruikt. Dit levert de volgende twee formules op:
$x = x_0 + l_1 \cdot sin(\theta_1) + l_2 \cdot sin(\theta_1 + \theta_2) + l_3 \cdot sin(\theta_1 + \theta_2 + \theta_3)$
$y = y_0 + l_1 \cdot cos(\theta_1) + l_2 \cdot cos(\theta_1 + \theta_2) + l_3 \cdot cos(\theta_1 + \theta_2 + \theta_3)$

Als de gevonden configuratie overigens niet in de oplossingsruimte valt wordt er naar een volgende oplossing gezocht.

2. Compute $J(e, \theta)$ for the current pose θ
   De Jacobi wordt afgeleid vanuit de ingaande onafhankelijke variabelen x, y en de uitgaande afhankelijke variablen $\theta_1, \theta_2, \theta_3$ ofwel $\mathbb{R}^3 \rightarrow \mathbb{R}^2$. Dit levert de volgende Jacobi matrix op: $J = \begin{bmatrix} \frac{dx}{d\theta_0} & \frac{dx}{d\theta_1} & \frac{dx}{d\theta_2} \\ \frac{dy}{d\theta_0} & \frac{dy}{d\theta_1} & \frac{dy}{d\theta_2} \end{bmatrix}$

   De afgeleiden die in deze matrix staan worden op de volgende manier berekend:
   $\frac{dx}{d\theta_0} = l_1 \cdot cos(\theta_1) + l_2 \cdot cos(\theta_1 + \theta_2) + l_3 \cdot cos(\theta_1 + \theta_2 + \theta_3)$
   $\frac{dx}{d\theta_1} = l_2 \cdot cos(\theta_1 + \theta_2) + l_3 \cdot cos(\theta_1 + \theta_2 + \theta_3)$
   $\frac{dx}{d\theta_2} = l_3 \cdot cos(\theta_1 + \theta_2 + \theta_3)$
   $\frac{dy}{d\theta_0} = -l_1 \cdot sin(\theta_1) - l_2 \cdot sin(\theta_1 + \theta_2) - l_3 \cdot sin(\theta_1 + \theta_2 + \theta_3)$
   $\frac{dy}{d\theta_1} = -l_2 \cdot sin(\theta_1 + \theta_2) - l_3 \cdot sin(\theta_1 + \theta_2 + \theta_3)$
   $\frac{dy}{d\theta_2} = -l_3 \cdot sin(\theta_1 + \theta_2 + \theta_3)$

3. Compute J^{-1}
   Om de inverse van deze matrix te berekenen word de pseudo inverse gebruikt. Dit resulteert in de volgende formule: $J^- = A^T \cdot (A \cdot A^T)^-$

4. $\Delta e = \beta(g - e)$
   Δe = doel coördinaten - huidige coördinaten $(g - e) \cdot \beta$

5. $\Delta\theta = J^- \cdot \Delta e$
   Hier wordt de inverse van de Jacobi (J^-) uit stap 3 maal het verschil van de coödinaten (Δe) uit stap 4 gedaan om het schil in de configuratie te benaderen.

6. $\theta_{nieuw} = \theta_{oud} + \Delta\theta$
   Tel hier de het verschil in de configuratie bij de oude configuratie bij op.

7. Compute new e vector
   Gebruik forward kinematics om te controleren of de gevonden configuratie klopt

2.1.2 het bepalen van de coördinaten

Om tot een berekening komen moeten er coördinaten bekend zijn van het op te pakken object en het doel. Hiervoor wordt er vanuit gegaan dat het vinden van deze objecten gewoon lukt en dit pixel coördinaten oplevert. In het programma wordt gebruik gemaakt van aruco. Dit is een libary van opencv die gemakelijk herkeningspunten aanbiedt. Als eerste wordt er op basis van deze vondst de verhouding tussen pixels en milimeters berekend. Het is bekend dat de geprinte aruco 42mm is dus het pixel naar mm ratio is dus $Aruco_{width}/42$ Dit resulteert in een ratio die vanaf nu R genoemd wordt. Vervolgens kan de positie correctie berekend worden ervan uit gaand dat de aruco precies op de coördinaten $(10, 0)$ ligt. De correctie voor het Y coördinaat is heel simpel dit is het Y punt waarop aruco gevonden is. De correctie voor het X coördinaat moet een kleine correctie krijgen omdat deze 10cm voor de base van de robot ligt. De berekening hiervoor is $X = Aruco_x + (10_{cm}/R)$

Nu deze informatie allemaal bekend is word kan deze gebruik worden in de positie bepaling van de objecten. De coördinaten van het object kunnen nu als volgt berekend worden:

$\delta X = Object_X - Correctie_x * R.$
$\delta Y = Object_Y - Correctie_x * R.$
$schuinezijde = \sqrt{(\delta X^2 + \delta Y^2)}.$
$Coördinaat_X = schuinezijde.$
$Coördinaat_Y = -baseheight$ Dit omdat het blok altijd op de grond ligt.
De hoek waaronder de base moet draaien wordt berekend met de inverse tangens. $tan^-(\delta Y/\delta X)$.

Dit wordt samengevoegd in een pakket kan naar de robot software gestuurd kan worden die met inverse kinematica de stand van arm vanaf het zijaanzicht berekend en de de hoek van de base daaraan toevoegt om op de juiste positie uit te komen.

2.1.3 Het bepalen van het pad

Voor het bepalen van het pad wordt wegens tijdsnood een simpel pad aangehouden van 4 punten. Eerst gaat de robotarm naar zijn op te pakken object. Vervolgens wordt deze zelfde configuratie aangenomen met een verkleining van de shoulder waardoor het object opgetild wordt. Vervolgend gaat de arm naar de configuratie van het doel met nog steeds de ophoging van de shoulder. Als laatste zal de arm naar de configuratie gaan van het doel (De witte cirkel) en daar het object loslaten.

2.2 Afwegingen

In het bepalen van het pad zijn er afwegingen gemaakt in het maken van de software deze zullen hieronder behandeld worden.

2.2.1 berekingen van het pad vanaf zijaanzicht met 2 dimensies

Voor het berekenen van de configuratie hebben is er een versimpeling gemaakt naar twee dimensies en een draaiing van de base. Dit doordat het rekenen met 4DOF naar 3 dimensies nog te complex was. Wel is er al een uitgerekend wat de jacobiaan is voor deze berekening voordat besloten is dit niet te gebruiken.

$$x = x_0 + l_1 \cdot cos(\theta_1) \cdot cos(\theta_0) + l_2 \cdot cos(\theta_1 + \theta_2) \cdot cos(\theta_0) + l_3 \cdot cos(\theta_1 + \theta_2 + \theta_3) \cdot cos(\theta_0)$$

$$y = y_0 + l_1 \cdot cos(\theta_1) \cdot sin(\theta_0) + l_2 \cdot cos(\theta_1 + \theta_2) \cdot sin(\theta_0) + l_3 \cdot cos(\theta_1 + \theta_2 + \theta_3) \cdot sin(\theta_0)$$

$$z = z_0 + l_1 \cdot sin(\theta_1) + l_2 \cdot sin(\theta_1 + \theta_2) + l_3 \cdot sin(\theta_1 + \theta_2 + \theta_3)$$

$$J = \begin{bmatrix} \frac{dx}{d\theta_0} & \frac{dx}{d\theta_1} & \frac{dx}{d\theta_2} & \frac{dx}{d\theta_3} \\ \frac{dy}{d\theta_0} & \frac{dy}{d\theta_1} & \frac{dy}{d\theta_2} & \frac{dy}{d\theta_3} \\ \frac{dz}{d\theta_0} & \frac{dz}{d\theta_1} & \frac{dz}{d\theta_2} & \frac{dz}{d\theta_3} \end{bmatrix}$$

$$\frac{dx}{d\theta_0} = -l_1 \cdot cos(\theta_1) \cdot sin(\theta_0) - l_2 \cdot cos(\theta_1 + \theta_2) \cdot sin(\theta_0) - l_3 \cdot cos(\theta_1 + \theta_2 + \theta_3) \cdot sin(\theta_0)$$

$$\frac{dx}{d\theta_1} = -l_1 \cdot sin(\theta_1) \cdot cos(\theta_0) - l_2 \cdot sin(\theta_1 + \theta_2) \cdot cos(\theta_0) - l_3 \cdot sin(\theta_1 + \theta_2 + \theta_3) \cdot cos(\theta_0)$$

$$\frac{dx}{d\theta_2} = -l_2 \cdot sin(\theta_1 + \theta_2) \cdot cos(\theta_0) - l_3 \cdot sin(\theta_1 + \theta_2 + \theta_3) \cdot cos(\theta_0)$$

$$\frac{dx}{d\theta_3} = -l_3 \cdot sin(\theta_1 + \theta_2 + \theta_3) \cdot cos(\theta_0)$$

$$\frac{dy}{d\theta_0} = -l_1 \cdot cos(\theta_1) \cdot cos(\theta_0) - l_2 \cdot cos(\theta_1 + \theta_2) \cdot cos(\theta_0) - l_3 \cdot cos(\theta_1 + \theta_2 + \theta_3) \cdot cos(\theta_0)$$

$$\frac{dy}{d\theta_1} = -l_1 \cdot sin(\theta_1) \cdot sin(\theta_0) - l_2 \cdot sin(\theta_1 + \theta_2) \cdot sin(\theta_0) - l_3 \cdot sin(\theta_1 + \theta_2 + \theta_3) \cdot sin(\theta_0)$$

$$\frac{dy}{d\theta_2} = -l_2 \cdot sin(\theta_1 + \theta_2) \cdot sin(\theta_0) - l_3 \cdot sin(\theta_1 + \theta_2 + \theta_3) \cdot sin(\theta_0)$$

$$\frac{dy}{d\theta_3} = -l_3 \cdot sin(\theta_1 + \theta_2 + \theta_3) \cdot sin(\theta_0)$$

$$\frac{dz}{d\theta_0} = 0$$

$$\frac{dz}{d\theta_1} = l_1 \cdot cos(\theta_1) + l_2 \cdot cos(\theta_1 + \theta_2) + l_3 \cdot cos(\theta_1 + \theta_2 + \theta_3)$$

$$\frac{dz}{d\theta_2} = l_2 \cdot cos(\theta_1 + \theta_2) + l_3 \cdot cos(\theta_1 + \theta_2 + \theta_3)$$

$$\frac{dz}{d\theta_3} = l_3 \cdot cos(\theta_1 + \theta_2 + \theta_3)$$

2.2.2 berekingen van het pad met inverse kinematica zonder A*

Voor het berekenen wordt er geen A* gebruikt. Dit omdat het ondanks effectief heel sloom is. De inverse kinematica die gebruikt is levert snel configuraties op maar kan niet garanderen als er niks gevonden dat er ook geen mogelijkheid was. Er is gekozen voor dit algoritme vanwege de snelheid.

3 Verloop opdracht

Bij de start van de opdracht is er eerst besproken hoe de samenwerking moest verlopen. Daar is gekozen voor het gebruik van Github, deze keuze is gebaseerd op het feit dat het gratis is en het versie beheer bekend is voor de studenten. Vervolgens is er geïnventariseerd wat er al beschikbaar was. Zo is er door Berentsen, M al een applicatie geschreven die de figuren kan herkennen en in kaart brengen en door Tunc, A een applicatie die via ROS de arm kan aansturen. Deze zijn met een lichte aanpassing direct bruikbaar in deze opdracht en worden dan ook beiden gebruikt.