Technische Universität Berlin
Fakultät IV Elektrotechnik und Informatik
Institut für Automatisierungstechnik
Fachgebiet Regelungstechnik
Projekt Analyse und Synthese von Regelungssystemen
Sommersemester 2019

Abschlussbericht zum Schwerpunktprojekt

SiD
Simulation of a Virtual Swarm of Kinematically Coupled Drones

erstellt von
Jan-Philip Wirsching Matr. Nr. 357498
Aarohan Jain Matr. Nr. 375232

Betreuer:
Dr.-Ing. Thomas Seel
Karsten Eckhoff

10. August 2019

Selbständigkeitserklärung

Hiermit erklären wir, dass wir das vorliegende Bericht selbständig verfasst und keine anderen als die angegebenen Hilfsmittel benutzt haben. Die Stellen des Berichts, die anderen Quellen im Wortlaut oder dem Sinn nach entnommen wurden, sind durch Angaben der Herkunft kenntlich gemacht. Dies gilt auch für Zeichnungen, Skizzen, bildliche Darstellungen sowie für Quellen aus dem Internet.

10. August 2019

_____

Jan-Philip Wirsching

_____

Aarohan Jain

Inhaltsverzeichnis

1 Einleitung

In diesem Projekt wurden mehrere Drohnen mit MATLAB simuliert. Außerdem folgt eine Umsetzung von äußeren Kräften, die es erlauben Drohnen zu koppeln. Mit dem Ergebnis können die Auswirkungen von gekoppelten Quadrocoptern und deren neuen Flugeigenschaften ermittelt werden.
Ein Ausblick mit dem Ergebnis des Projektes kann sein, dass man ein durchgängigen Flug eines Flugtaxis untersuchen kann. Um das genannte Taxi werden mehrere Quadrocopter angedockt, die für den Flug zuständig sind. Während des Fluges werden die entladenen Quadrocopter ausgetauscht. Die entladenen Drohnen fliegen zu einer in der Nähe liegende Ladestation zurück und werden für den nächsten Einsatz aufgeladen.

2 Quadrocopter

Für das Projekt wurden verschiedene Modelle zur Beschreibung eines Quadrocopters untersucht. Das gewählte Modell [4] ist linearisiert. Somit ist es einfacherer das Modell in eine S-Funktion zu implementieren. Für die Linearisierung wurden ein paar Annahmen getroffen. Die Orientierungswinkel (Roll, Pitch, Yaw) werden als klein angenommen. Die Winkel werden zwischen 25° und -25° liegen. (2.1) Somit kann man eine Kleinwinkelnäherung anwenden. Dadurch können die Sinus und Kosinus Terme als Konstanten betrachtet werden. Hiermit erlangt man eine Linearisierung des Modells. Im Modell werden Matrizen erzeugt, die eine intrinsische Zustandsrückkopplung aufweisen. Die Zustände sind die Roll-Pitch-Yaw Winkel mit deren Geschwindigkeiten und die Position mit dessen Geschwindigkeiten. Zusätzlich wird die Gravitation als Zustand betrachtet. Als Teilausgang bekommt man die geregelte Positionen und Orientierungen, welche mit dem Eingang (Referenzen) verglichen werden. Somit wird der Quadrocopter die vorgegebene Referenz verfolgen.

$$\begin{aligned} \sin(\theta) &\approx \theta \\ \cos(\theta) &\approx 1 \end{aligned} \tag{2.1}$$

3 Zustandsraummodell des Quadrocopters

3.1 Kontinuierliches Zustandsraummodell

Das gewählte Zustandsraummodell [4] besitzt 13 Zustände. Die ersten 6 Zustände beschreiben die Eulerwinkel ($\varphi = x_1$, $\theta = x_3$, $\psi = x_5$), als auch die jeweilige Winkelgeschwindigkeit. Die nächsten 6 Zustände beschreiben die Position (x =x_7, y =x_9, z = x_{11}) und die Geschwindigkeit des Quadrocopters. Der 13. Zustand ist die Gravitation. Da es keine Abhängigkeit vom Ausgang und Eingang herrscht, bleibt der 13. Zustand konstant. Dieser Zustand wurde zusätzlich hinzugefügt.
Der Eingang u = $\omega = [\omega_1^2, \omega_2^2, \omega_3^2, \omega_4^2]^T$ beschreibt die quadratische Winkelgeschwindigkeit der 4 Rotoren von der Drohne. Diese Eingänge erzeugen die entsprechende Auftriebskräfte für den Quadrocopter.
Die 6 Ausgänge beinhalten die Informationen für die Position und der Orientierung der Drohne.
Mithilfe des Pole Placement Verfahrens wurde der PID-Regler entworfen. Die Pole wurden aus dem gewählten Paper [4] entnommen und an den Stellen

$$\begin{aligned} z_1 &= -2 + 1.96j \\ z_2 &= -2 - 1.96j \\ z_{3,4} &= -2.0 \\ z_{5,6} &= -4.0 \\ z_{7,8} &= -5.0 \\ z_{9,10} &= -8.0 \\ z_{11,12} &= -3.0 \\ z_{13} &= 0 \end{aligned} \tag{3.1}$$

festgelegt. Für den 13. Zustand wurde der Pol in dem Ursprung gelegt. Da der Zustand konstant (vom Eingang und Ausgang unkontrollierbar) ist, wurde ein grenzstabiler Pol zugewiesen.

Die erstellten Matrizen, die den modifizierten Regelkreis beschreiben: (3.2) (3.3)

$$[\dot{x}] = \underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ A_x & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & A_y & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & A_z & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -g \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}}_{A} [x] + \underbrace{\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \frac{-lk}{I_{xx}} & 0 & \frac{lk}{I_{xx}} \\ 0 & 0 & 0 & 0 \\ \frac{-lk}{I_{yy}} & 0 & \frac{lk}{I_{yy}} & 0 \\ 0 & 0 & 0 & 0 \\ \frac{b}{I_{zz}} & \frac{-b}{I_{zz}} & \frac{b}{I_{zz}} & \frac{-b}{I_{zz}} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ k & k & k & k \\ 0 & 0 & 0 & 0 \end{bmatrix}}_{B} [u] \tag{3.2}$$

$$[y] = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}}_{C} [x] \tag{3.3}$$

Die Konstanten l, k, b, m, g beschreiben den Radius des Quadrocopters, den Auftriebs- und Luftwiderstand, die Masse des Quadrocopters und die Schwerkraft. I_{xx}, I_{yy}, I_{zz} beschreiben die Trägheit für jede Achse.

3.2 Diskretisiertes Zustandsraummodell

Nach der Umsetzung des kontinuierlichen Zustandsraummodell folgt die Diskretisierung mithilfe MATLAB. Dieses wird notwendig sein, falls man das Modell als ein reales Modell betrachten würde, das auf einem digitalen Mikrocontroller basiert.
Es wird die MATLAB-Funktion *place* verwendet, um von den kontinuierlichen Matrizen (A, B) und einen vorgegebenen Polvektor P (3.1) die "Feedback Gain" Matrix K

zu erzeugen (3.4). Die Matrix G wird berechnet, sodass im Modell die Quadrocopter-Referenzen als Eingang verwendet werden können (3.5).

$$K = place(A, B, P) \tag{3.4}$$

$$G = -B^{-1} \cdot (A - B \cdot K) \cdot C^{-1} \tag{3.5}$$

Mithilfe dieser Matrizen werden neue Zwischenmatrizen erzeugt, die eine Zustandsrückkopplung aufweisen, und helfen dabei, die diskreten Matrizen zu erzeugen. (3.6)

$$\begin{aligned} A_{neu} &= A - B \cdot K \\ B_{neu} &= B \cdot G \end{aligned} \tag{3.6}$$

Die Matrix A_{neu} wird in einer approximierenden Reihe eingesetzt, um die diskrete A-Matrix A_d zu bekommen. Da das System von 13. Ordnung ist, wird angenommen, dass nach 100 Durchläufe die Terme verschwinden. Die Matrix B_d wird in Abhängigkeit von A_d berechnet. (3.7)

$$\begin{aligned} A_d &= \sum_{k=0}^{\infty} A_{neu}^k \frac{T_s^k}{k!} \\ B_d &= (A_{neu})^{-1} \cdot (A_d - I) \cdot B_{neu} \\ C_d &= C \\ D_d &= 0 \end{aligned} \tag{3.7}$$

Schließlich erhält man das folgende diskrete System: (3.8)

$$\begin{aligned} x[n+1] &= A_d \cdot x[k] + B_d \cdot u[k] \\ y[k] &= C_d \cdot x[k] \end{aligned} \tag{3.8}$$

4 Erste Simulationsergebnisse

Um die Funktionalität des Modells zu überprüfen, wurde durch MATLAB mehrere Verläufe des Quadrocopters simuliert.
Für die Simulation wurden die Konstanten aus dem Paper [4] übernommen. Die Luftwiderstandskoeffizienten betragen jeweils $A_x = A_y = A_z = 0.25$. Der Auftriebs- und Luftwiderstand liegt bei $k = 2.98 \cdot 10^{-6}$ und $b = 1.14 \cdot 10^{-7}$. Die Trägheit der einzelnen Achsen wurde mit den Werten $I_{xx} = 0.00497$, $I_{yy} = 0.0055$ und $I_{zz} = 0.01$ definiert. Der Radius des Quadrocopters wurde auf 0.175m gelegt.
Die Abtastzeit befindet sich bei $T_s = 0.04$s, um mit dem WebSocket kompatibel zu sein.

Der Referenzweg (4.1) für die Drohne verläuft, sodass sie in eine Höhe von 1m hochfliegt. Außerdem wird sie sich in einem Radius von 8m im Kreis fliegen. Die Orientierungswinkel werden in beiden Referenzen jeweils auf 0 eingestellt.

$$ref = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 8 \cdot \sin(\frac{2 \cdot \pi \cdot t}{10}) \\ 8 \cdot \cos(\frac{2 \cdot \pi \cdot t}{10}) \\ 1 \end{bmatrix} \tag{4.1}$$

In der Abbildung 1 sieht man den Verlauf für die Flughöhe. Es lässt sich gut erkennen, dass nach wenigen Sekunden die gewünschte Höhe von 1m erreicht wurde. Außerdem erkennt man, dass es keine Regelabweichung gibt.

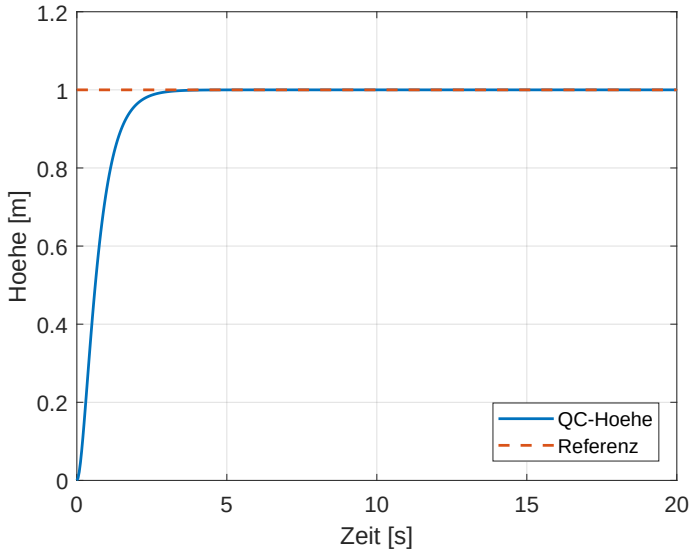


Abbildung 1: Flughöhe eines Quadrocopters

In der Abbildung 2 wird die Flugbahn einer Drohne in einem 3D-Plot dargestellt. In dieser Darstellung sieht man gut, wie der Referenzweg definiert wurde. Die Flugbahn der Drohne weist zur vorgegebenen Referenz eine kleine Abweichung. Trotzdem liefert es ein zufriedenstellendes Ergebnis.
In der Abbildung 3 sieht man, wie die Orientierungswinkel der Drohnen sich verhalten. Der Pitchwinkel liegt innerhalb der 1. Sekunde bei 55°, da die Drohne sich zuerst vom Ursprung zu der kontinuierlichen Kreisbewegung auf der xy-Ebene mit dem Radius von 8m regeln muss. Da die Verläufe nach paar Sekunden sinusförmig verlaufen, kann man

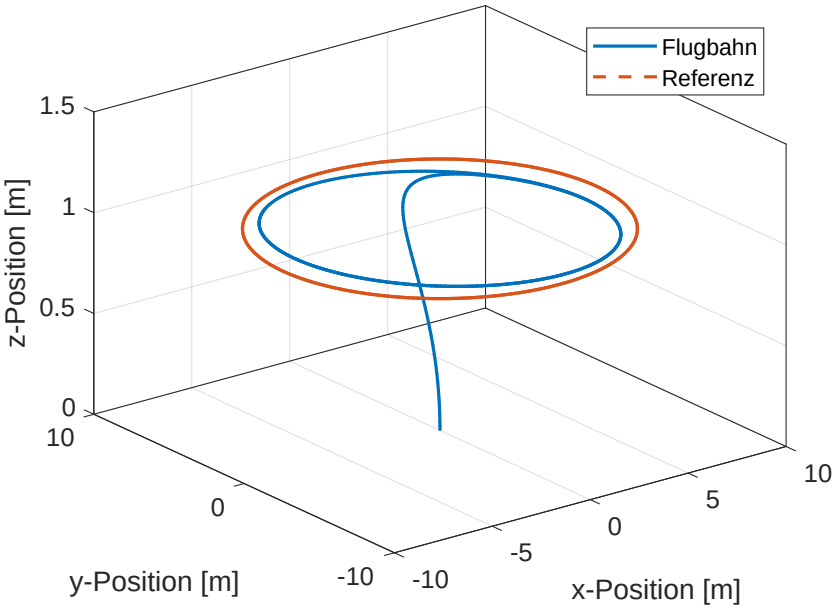


Abbildung 2: Flugbahn eines Quadrocopters

sich sicher sein, dass die Drohne im Kreis fliegt. Die Kleinwinkelnäherung wird nach der Regelung der Position eingehalten, alle Orientierungswinkel haben einen kleineren Betrag als 25°.

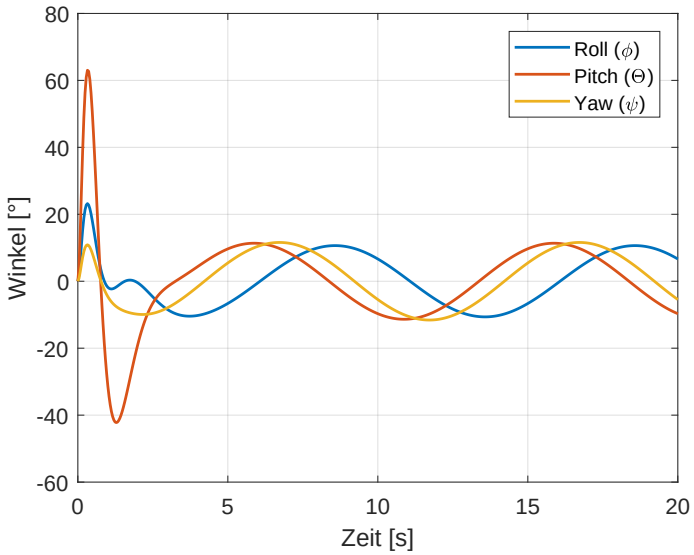


Abbildung 3: Orientierung eines Quadrocopters

5 Wichtige Bausteine des Projekts

Das skizzierte Modell (Abb. 4) wurde in Simulink umgesetzt. In dem Eingang des Modells wird eine Referenz, wie z.B. die Referenz aus (4.1), vorgegeben. Die erstellten Matrizen aus dem Reglerentwurf und die Reglerdynamik wurde in einer S-Funktion geschrieben. Die S-Funktion ist ein Block in Simulink, die in diesem Projekt die ganze Reglerdynamik enthält. Der Code kann entweder in C, C++, Fortan oder direkt mit der Schreibweise von MATLAB geschrieben werden. Die Vorteile mit dieser Variante sind unter anderem, dass man alles in einem Block schreiben kann und dies hilft der Übersichtlichkeit in Simulink. Außerdem ist die Implementierung leichter. Auch ist es in der Entwicklung schneller. [2]
Zusätzlich gibt es ein Block, die die Kraft berechnet, wenn sich zwei oder mehrere Quadrocopter berühren. Die Ermittlung der Kräfte wird in den Kapitel 7 und 8 näher erläutert. Falls eine Kraft auftritt, wird diese Kraft zum Eingang der S-Funktion geführt, um die Position nach Bewirkung der Kräfte zu regeln. Somit ergibt sich eine Rückkopplung.
Der Ausgang mit der Position und der Orientierung wird zum WebSocket weitergeleitet, um das Ergebnis auf einem Webapp darzustellen.
Der WebSocket ist die Verbindung zwischen den generierten Code und Babylon. Da der WebSocket nur mit einem C++ Code kompatibel ist, musste der Code von der S-Funktion in C++ generiert werden.
Babylon ist ein Webapp in Javascript zur visuellen Darstellung von 3D-Objekten. Die Java-Dateien wurden angepasst, um mehrere Quadrocopter-Objekte zu erzeugen, sodass in der Darstellung die äußeren Kräften erkennbar sind.

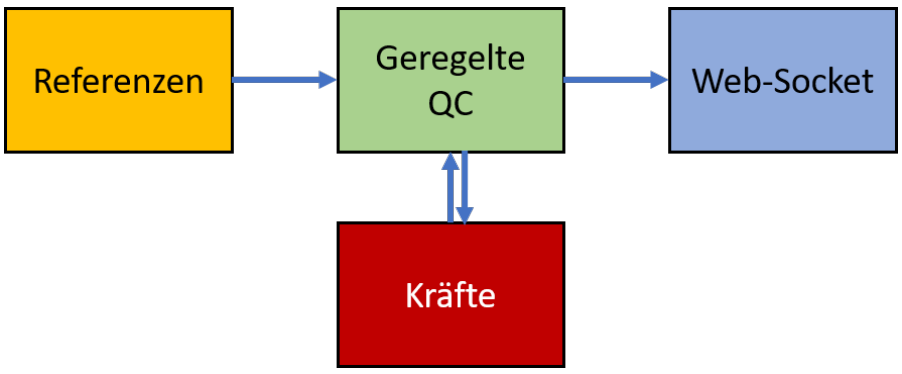


Abbildung 4: Skizze zum Aufbau des Simulinkmodells

6 Aufbau der einzelnen Simulinkblöcke

In der Abbildung 5 ist das gesamte Simulink-Modell zu sehen. In die Eingänge des großen Blocks sind die Referenzwege der 3 Quadrocopter, als auch die berechneten Kräfte bei einem Zusammenstoß oder einer Anziehung von mindestens 2 Drohnen, zu sehen. Die ersten 3 Eingänge sind die Orientierungswinkel und die darauf folgende 3 Eingänge sind die Position der Drohne. Aus dem großen Block gelangen unter anderem die berechneten Quaternionen, die tatsächlichen Positionen und die Orientierung der Drohnen. Am 2. Ausgang befindet sich die jeweilige Drehrate der Drohnen und am 4. Ausgang liegen die jeweiligen Geschwindigkeiten.
Die berechneten Positionen und Quaternionen werden zum einen zum WebSocket geleitet und zum anderen für die Kräfteberechnung. Die berechneten Kräfte werden mit einer Zeitverzögerung in den großen Block rückgekoppelt.
In dem Block der Kräfteberechnung befindet sich die Funktion, die die Kräfte bei einem Zusammenstoß oder einer Anziehung von mindestens 2 Drohnen berechnet (Abb.6). In die Funktion gelangen die berechneten Quaternionen und Positionen der Drohnen. Außerdem gelangt in dem Eingang der Radius einer Drohne. Aus der Funktion gelangen zwei Kräfte. Die erste Kraft beschreibt das Kraftverhalten bzgl. der $x - y - z$ Achsen. Die zweite Kraft soll das Verhalten bzgl. Roll, Pitch, Yaw beschreiben.
In dem großen Block liegt für jede Drohne jeweils ein S-Funktion Builder (Abb.6 (a)) und die Funktionen für die Umrechnung von Eulerwinkel zu Quaternionen (Abb.6 (b)). Die S-Funktion beinhaltet die im Skript berechneten Matrizen. Das geregelte System ist in einem Block beschrieben. Außerdem können die Anfangszustände eingestellt und die rückgekoppelten Kräfte addiert werden. Die MATLAB-Funktion EulertoQuat rechnet die geregelten Orientierungen von Euler Winkeln (Roll-Pitch-Yaw) zu Quaternionen um, um mit dem Web-Socket kompatibel zu sein. Die Umrechnung ist in der Gleichung (6.1) zu sehen. [3]

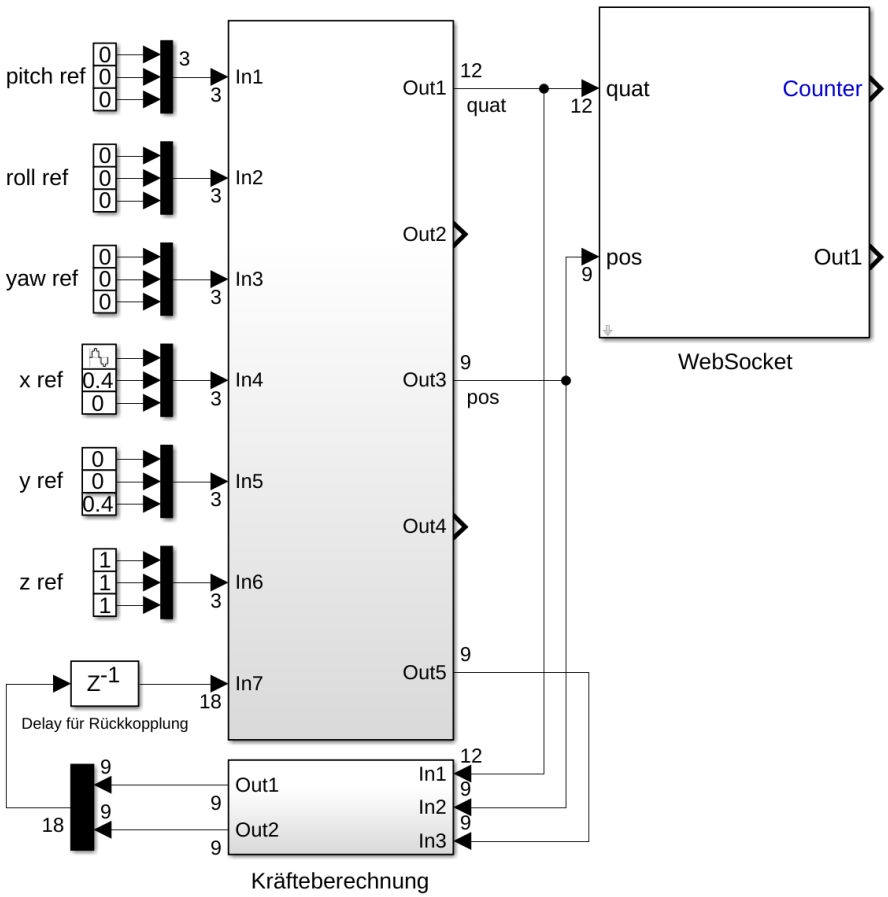


Abbildung 5: Aufbau des Simulinkmodells

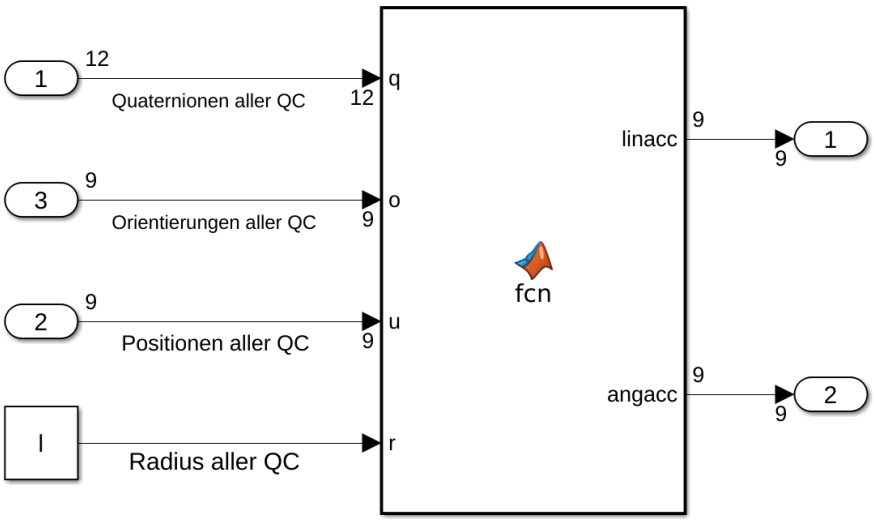
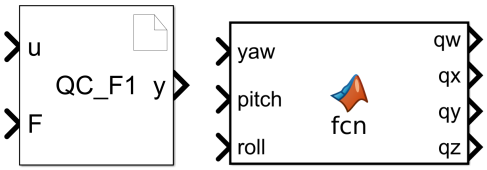


Abbildung 6: Funktionsblock für die Kräfteberechnung



(a) (b)

Abbildung 7: (a) S-Function Builder, (b) Umrechnung von Eulerwinkel zu Quaternionen

$$
\begin{aligned}
cy &= \cos(\psi/2) \\
sy &= \sin(\psi/2) \\
cp &= \cos(\theta/2) \\
sp &= \sin(\theta/2) \\
cr &= \cos(\varphi/2) \\
sr &= \sin(\varphi/2)
\end{aligned} \tag{6.1}
$$

$$
\begin{aligned}
q_w &= cy \cdot\ cp \cdot\ cr + sy \cdot sp \cdot sr \\
q_z &= cy \cdot\ cp \cdot\ sr - sy \cdot sp \cdot cr \\
q_y &= sy \cdot\ cp \cdot\ sr + cy \cdot sp \cdot cr \\
q_x &= sy \cdot\ cp \cdot\ cr - cy \cdot sp \cdot sr
\end{aligned}
$$

7 Konzept von äußeren Kräfte an Quadrocoptern

Um die Kollision zu verwirklichen, wurde um der Position des Quadrocopter eine imaginäre Kugel integriert. Die Kugel besitzt den Radius eines Quadrocopters. Falls die Drohnen sich berühren, wird der Abstand der beiden Positionen kleiner sein als die Summe von den 2 Radien der Drohnen. Mit der Eindringtiefe d (7.1) kann bestimmt werden, wie stark die Quadrocopter sich berühren. Je kleiner der Abstand zwischen den Mittelpunkten der Drohnen ist, desto stärker werden die Drohnen sich abstoßen.
Die Richtung der Kraft $\overrightarrow{R}$ (7.2) lässt sich bestimmen, wenn man die 2 Positionen der Quadrocopter subtrahiert und durch die Norm der beiden Positionen teilt.
Es wird angenommen, dass es eine quadratische Beziehung zwischen der Kraft $\overrightarrow{F_i}$ und der Eindringtiefe d gibt. Diese Beziehung realisiert eine annehmbare realistische Wirkung der Kraft. Außerdem hängt die Kraft noch von deren Richtung ab. Es wird auch eine Konstante k eingesetzt, um die Kraft mit einem linearen Faktor etwa verstärken bzw.

dämpfen zu können. (7.3)

$$d = 2r - ||(p_i - p_j)||_2 \tag{7.1}$$

$$\overrightarrow{R} = \frac{\overrightarrow{p_i} - \overrightarrow{p_j}}{||(p_i - p_j)||_2} \tag{7.2}$$

$$\overrightarrow{F_i} = k \cdot d^2 \cdot \overrightarrow{R} \tag{7.3}$$

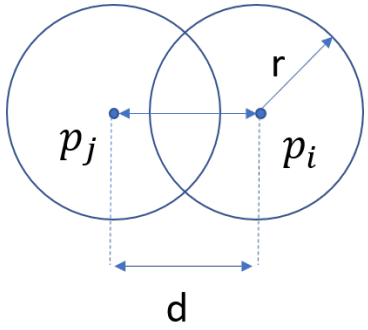


Abbildung 8: Konzeptzeichnung zur Berechnung für äußeren Kräfte zwischen 2 Quadrocoptern

8 Konzept der Kopplung von mehreren Quadrocoptern

Damit die Drohnen miteinander koppeln können, werden an den Drohnen Positionen definiert an den die Drohnen miteinander koppeln können.
Erstens wird die Position eines 'Magneten' (definierte Positionen zum Kopplen von Drohnen) bzgl. der Position des QC festgelegt. Jede Position eines QC befindet sich in einen eigenen Koordiantensystem. Deshalb muss es in ein globales Koordinatensystem umgewandelt werden, damit jede Position im gleichen System befindet. Dafür wird die Methode quaternionRotate verwendet (8.1), um die lokale Position eines Magneten auf dem Quadrocopter $d_{m,i}$ anhand des Quaternionvektors vom Quadrocopter q_i^n zu drehen. Das Ergebnis ist ein Vektor. Dies wird zusammen mit der globalen Position des Quadrocopters p_i^n addiert, um die globale Position vom Magneten, $p_{m,i}^n$, zu erhalten.
Die anziehende Kraft lässt sich durch die Formel (8.2) berechnen.
Um diese Kraft in der S-Funktion implementieren zu können, wird sie mithilfe einer Konstante in eine Beschleunigung a_i umgewandelt (8.3). Weiter wird wieder die Methode quaternionRotate in (8.4) verwendet, um aus der berechneten Kraft die Drehmoment der

betroffenen Quadrocopter zu bestimmen. Anschließend werden in der Gleichung (8.5) die Drehmomente von Euler-Winkeln zu z-y-x-achsenbezogene Winkel umgewandelt (8.6).[1] Die Formeln gelten ab den Zeitpunkt, indem der Abstand von mindestens 2 Magneten kleiner gleich der 3-fachen Radienweite einer Drohne ist. Ist dies der Fall sind die Magnete aktiviert.

$$p_{m,i}^n = p_i^n + \text{quaternionRotate}(q_i^n, d_{m,i}) \tag{8.1}$$

$$\overrightarrow{F_i} = \frac{(\overrightarrow{p}_{m,j}^n - \overrightarrow{p}_{m,i}^n)}{||p_{m,j}^n - p_{m,i}^n||_2} \cdot \frac{1}{||p_{m,j}^n - p_{m,i}^n||_2^2} \tag{8.2}$$

$$\overrightarrow{a_i} = c_1 \overrightarrow{F_i} \tag{8.3}$$

$$\overrightarrow{\tilde{\alpha}_i} = c_2 \cdot \text{quaternionRotate}(q_i^n, d_{m,i}) \times \overrightarrow{F_i} \tag{8.4}$$

$$\overrightarrow{\alpha_i} = \text{conversion}(\overrightarrow{\tilde{\alpha}_i}) \tag{8.5}$$

$$\begin{bmatrix} \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{bmatrix} = \begin{bmatrix} 1 & \tan(\Theta)\sin(\Phi) & \cos(\Phi)\tan(\Theta) \\ 0 & \cos(\Phi) & -\sin(\Phi) \\ 0 & \sin(\Phi)/\cos(\Theta) & \cos(\Phi)/\cos(\Theta) \end{bmatrix} \cdot \overrightarrow{\alpha_i} \tag{8.6}$$

9 Auswertung

Zuerst wird der entworfene Regler ausgewertet. In der Abbildung 2 ist zu sehen, dass es einen deutlichen Unterschied zwischen der Referenz und der eigentlichen Flugbahn gibt. In den weiteren Simulationsergebnissen wurde eine Referenz untersucht, in denen die Quadrocopter nur in der x- oder y-Richtung bewegen sollen. Es ist aber zu erkennen, dass die Quadrocopter eine kleine aber merkbare elliptische Bewegung besitzen. Das bedeutet, dass der Regler nicht perfekt ist. Die erstellten Reglermatrizen können verbessert werden, um die Anforderungen genauer zu erfüllen.
Nun wird die Methode bewertet, die die Kräfte realisiert. Die Wechselwirkungen zwischen zwei beliebigen Quadrocoptern in einem theoretischen Schwarm wurden einigermaßen effizient implementiert. Es werden alle möglichen Kombinationen von 2 Quadrocoptern durch verschachtelte for-Schleifen auf eine eingestellte Bedingung überprüft. Allerdings müssen die Dimensionen der Kraftmatrizen angepasst werden, wenn die Anzahl

der Drohnen erhöht wird. Die Komplexität der Funktion ist (mindestens) $O(n^2)$. Damit steigt der Aufwand sehr schnell durch eine Erhöhung der Anzahl von Quadrocoptern.
Die Implementierung der Orientierungskraft war nicht erfolgreich gewesen. Die Drohnen drehen sich nicht in die Richtung des gegenüberliegenden Magneten. Sie haben aber um die x- und y-Achse gedreht. Dies ist jedoch die falsche Richtung. Der Grund dafür könnte sein, dass die Reihenfolge von Drehungen falsch angenommen wurden (es gab keine genauen Angaben im Paper [4]). Damit könnte die Rotationsmatrix in (8.6) falsch gewählt worden sein.
Außerdem wurde für die Kraftberechnung eine imaginäre Kugel um die Drohne platziert. Diese ist bei einer Kollision ungenauer, als wenn man ein Quader um der Drohne legt.
Es wurden einige Ziele des Projektes nicht erreicht. Das erste ist die Implementierung des Blendermodells anstelle des vorgegeben Quadrocopters in Babylon. Auch wurde nicht mehrere Regler miteinander verglichen. Außerdem lässt sich nicht das Terminal im Webapp ansteuern und die Konstanten können für die Kräfte im Webapp nicht verändert werden, um verschiedene Effekte zu sehen.

10 Zusammenfassung

Ziel dieser Arbeit war es ein Modell eines Quadrocopters in MATLAB zu verwirklichen und darauf aufbauend sollten mehrere Drohnen miteinander gekoppelt werden. Hierfür wurde ein vorhandenes Simulinkmodell verwendet, der zuvor das Flugverhalten von verschiedenen Übertragungsfunktionen simuliert. Für das Projekt wurde das entnommene Modell [4] mithilfe eines S-Function Builder entworfen. Anschließend mussten die Javascript Dateien an das neue Modell angepasst werden. Damit die Drohnen sich gegenseitig abstoßen und anziehen können, wurde daraufhin ein Kraftmodell entworfen.
Abschließend lässt sich sagen, dass das Projekt teilweise erfolgreich war, einen Schwarm von Drohnen virtuell zu simulieren, und die Kräfte dazwischen zu betrachten. Allerdings gibt es noch Ziele, die nicht erreicht werden konnten, welche die Funktionalität des Modells verbessern können.

Literatur

[1] Angular velocity conversions. https://davidbrown3.github.io/2017-07-25/EulerAngles/. Zugriff: 05.08.2019.

[2] Implementing s-functions. https://www.mathworks.com/help/simulink/sfg/implementing-s-functions.html. Zugriff: 01.07.2019.

[3] DM Henderson. Shuttle program. euler angles, quaternions, and transformation matrices working relationships. 1977. Zugriff: 20.05.2019.

[4] TS Tengis and A Batmunkh. State feedback control simulation of quadcopter model. In *2016 11th International Forum on Strategic Technology (IFOST)*, pages 553–557. IEEE, 2016. Zugriff: 10.07.2019.