Technische Universität Berlin
Fakultät IV Elektrotechnik und Informatik
Fachgebiet Regelungssysteme
Leiter: Prof. Dr.-Ing. Jörg Raisch

Schwerpunktprojekt
Analyse und Synthese von Regelungssystemen
Leiter: Dr. Thomas Schauer

Regelung eines Quadrocopters

erstellt von Team A:
Serdar Gareayaghi Matr. Nr. 374183
Onur Akdemir Matr. Nr. 375959
Vrontos Apostolos Matr. Nr. 367770
Ongun Türkcüoglu Matr. Nr. 371690

11. März 2019

Erklärung

Hiermit erkläre ich, dass ich wesentliche Teile zum vorliegenden Protokoll beigetragen habe. Weiterhin erklären wir als Gruppe, dass keine weiteren Personen an der Anfertigung des Protokolls beteiligt waren. Die verwendeten Quellen werden am Ende des Dokuments benannt.

_____

Serdar Gareayaghi

_____

Onur Akdemir

_____

Vrontos Apostolos

_____

Ongun Türkcüoglu

Inhaltsverzeichnis

Verwendete Variablen

ψ: Yaw Winkel

ϕ: Roll Winkel

θ: Pitch Winkel

T: Thrust

F: gesamte Schubkraft

m: Masse des Quadrocopters (27 gramms)

x, y, z: Koordinaten für die Position des Quadrocopters

1 Einleitung

Das Ziel dieses Projekts ist die Positionierung eines Nano Quadrocopters (Crazyflie 2.0). Der Crazyflie 2.0 Nano Quadcopter ist eine fliegende Plattform, die ideal für Entwickler geeignet ist. Er hat eine geringe Größe und ein kleines Gewicht und dadurch ist er sowohl für den Freienbereich ,wie andere große Quadrocoptern, als auch für den Innenbereich geeignet. Er ist bereit zum fliegen nach seine 4 Motoren befestigt wurden.

Die Positionierung wird durch Lage- und Rotationsregelung ermöglicht. Zur Positionsbestimmung wird eine Intel RealSense D435 – Tiefenkamera verwendet. Mit diesem Kamera kann man intern die Entfernung eines Objekts erkennen.

Als Betriebssystem wird Ubuntu 18 LTS und als Programmiersprache wird Python verwendet. Um die Bildverarbeitung zu ermöglichen werden AR-Code benutzt. Durch der Bildverarbeitung wird die 3D-Position des Quadrocopters und die Pitch(ϕ), Roll(θ) und Yaw(ψ) Winkeln genommen. Diese Werte sind gebraucht um ein geeigneter Regelkreis zu Implementieren. Die Verbindung zwischen der Bildverarbeitung und Regelung also alle Kommunikationen werden mittels ZMQ(zeromq) ermöglicht.

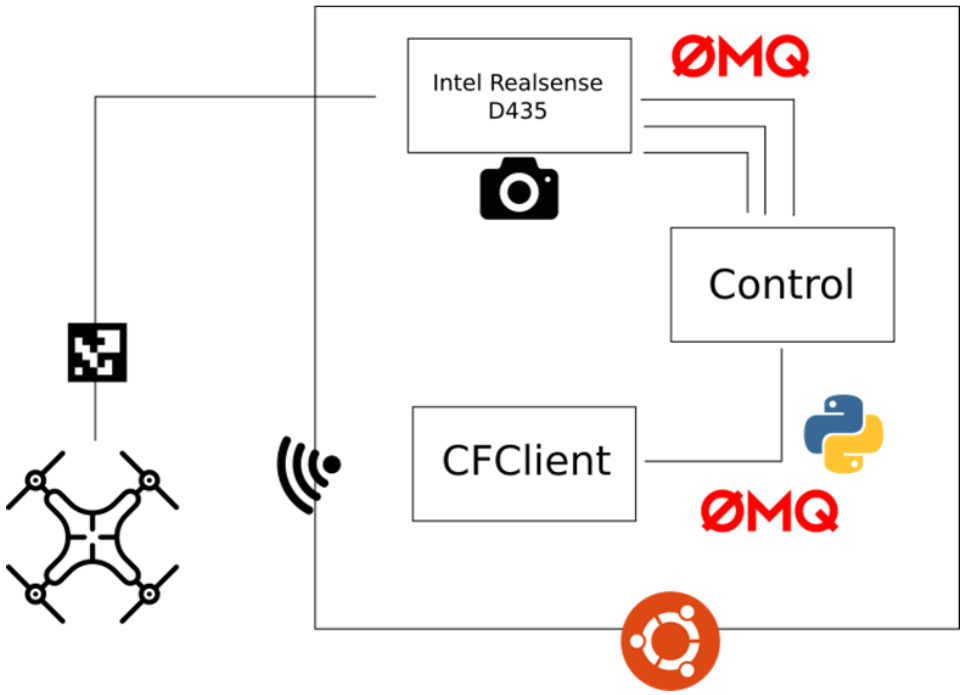

Abbildung 1: Übersicht des Blockschaltbilds

2 Bildverarbeitung

Sämtliche Quadrocopters (auch Crazyflie 2.0) haben externe Module, die die Erfassung der Position der Drohne ermöglichen. Jedoch ist noch eine Variante; die Positionierung durch eine externe Kamera. Intel RealSense D435 verfügt ein Stereo-Bild, wobei die Tiefeninformation durch Farbencodierung ermittelt wird. Im Vergleich zur normalen 2D-Kameras kann man deswegen zusätzlich zu den X-Y-Koordinaten noch die Z-Koordinate ableiten, welche für die Regelung eines Quadrocopters essentiel ist.
Für die Bildverarbeitung gibt es viele Ansätze, die jeweils mit Vor- und Nachteilen verbunden sind. Im Rahmen dieses Projekt haben wir im Wesentlich zwei unterschiedliche Ansätze ausprobiert: Bildverarbeitung durch Farbenerkennung und Bildverarbeitung durch AR-Markers.

2.1 Bildverarbeitung durch Farbenerkennung

Die Farbenerkennung ist sehr verbreitet und relativ leicht zu implementieren. Exemplarische Codes und Tutorials können im Internet gefunden werden, und dieses Teil beschäftigt sich nur mit dem Konzept, Details sind ausgelassen.
RealSense D435 kann gleichzeitig das 2D-Bild und das Stereo-Bild streamen. Die beiden Bilder sind normalerweise nicht deckungsgleich, deswegen muss man sie anpassen, bevor die Tiefeninformation extrahiert werden kann. Der RGB-Farbraum (Red, Green, Blue) ist die übliche Farbencodierung für Videos und Bilder, aber der Farbraum ist für die Farbenerkennung schlecht geeignet, denn die Licht-Bedingungen nicht gut definiert sind. Deswegen ist eine Übersetzung zu dem HSV-Farbraum (Hue, Saturation and Value) in der Regel gewünscht. In dem HSV-Farbraum kann einen Bereich definiert werden, welcher die gewünschte Farbe maskiert. Im Idealfall würde die Kamera nur den Bereich filtern, und durch Cropping kann die Z-Information von dem Stereo-Bild ermittelt werden.

Theoretisch ist dieses Verfahren sehr eindeutig und leicht zu implementieren, jedoch die Implementierung muss starke Begrenzungen annehmen. Das größte Problem dabei ist die Lichtabhängigkeit des gewünschten Farbenbereichs. Der Bereich ist bei der einfachen Implementierung statisch und kann nicht robust gegen unterschiedlichen Lichtbedingungen reagieren. Stochastische Algorithmen wie das Meanshift-Algorithmus, welches die stärkste Lichtamplitude folgt, sind robuster, jedoch immer noch von dem statischen Lichtbereich abhängig und rechenintensiv. Aus diesen Gründen wird bei diesem Projekt AR-Markers verwendet.

2.2 Bildverarbeitung durch AR-Markers

AR-Markers (Augmented-Reality) sind eine Alternative für die Bildverarbeitung. AR-Markers stammen in der Regel aus einem vordefinierten Dictionary, und jeder Marker ist mit einem ID verbunden. Solange das 2D-Bild scharf und hoch-aufgelöst ist, wird der Marker von der Kamera erkannt. Außerdem können mit solcher Markers unterschiedliche Eigenschaften abgeleitet werden, wie die Kamera-Positionierung, Rotations- und Translationsmatrizen des Markers bzw. der Kamera. Für die Regelung eines Quadrocopters sind diese Eigenschaften wichtig, weil die Bestimmung der Orientierung der Drohne dadurch erleichtert wird. Theoretisch muss man in dem Fall keine Stereo-Kamera benutzen, aber solche Kameras haben schon interne Parameters, die die Implementierung optimieren.

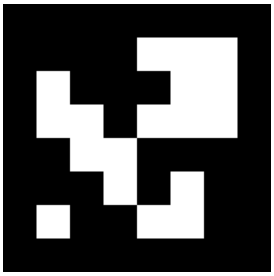

Abbildung 2: AR-Marker aus ArUco-Markers

Der Framework OpenCV2 für Python hat die eigenständige Bibliothek *cv2.aruco*, mit der die komplette Erstellung und Detektion eines Markers möglich ist. Randbedingungen für die Erkennung eines AR-Markers sind:

- A priori-Wissen der reellen Länge eines ArUco-Markers.
- Die Eckpunkte und die Kanten eines Markers müssen in jedem Frame erkennbar sein.
- Die Kamera muss eine hohe Auflösung haben und im Idealfall soll die Abtastrate hoch sein. Der Grund dafür hat seine Wurzeln in der zweiten Bedingung, denn ein unscharfes Bild bedeutet in der Regel, dass die Eckpunkte nicht mehr richtig erkennbar sind.

Der Framework verfügt Funktionen für die ArUco-Marker-Detection. Dabei sind die folgende zwei Funktionen wichtig:

2.2.1 cv2.aruco.detectMarkers

Die Funktion *detectMarkers()* ist die Erkennung des Markers in dem Frame. Nachdem ein Marker richtig erkannt ist, werden die IDs und die **Eckpunkte (corners)** der detektierten Markers ausgegeben. Die Hilfsfunktion *drawMarker()* zeichnet das Gebiet des Markers. Die Abbildung 3 veranschaulicht einen detektierten Marker.

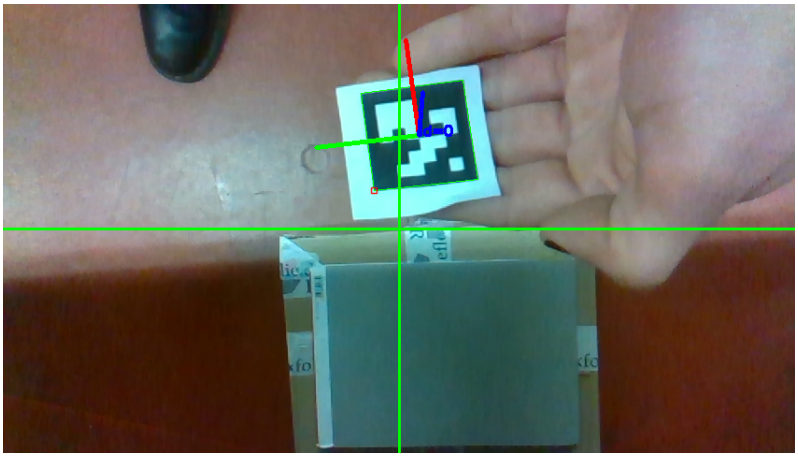

Abbildung 3: Detected AR-Marker

2.2.2 cv2.aruco.estimatePoseSingleMarkers

Die Funktion *estimatePoseSingleMarkers()* nimmt als Argument die Eckpunkte (aus der ersten Funktion detectMarkers()) des detektierten Markers und die Kameraparameters. Die Kameraparameter werden normalerweise (im Fall einer 2D-Kamera) mithilfe einer anderen Funktion (cv2.aruco.calibrateCameraAruco) gewonnen, aber im Fall einer Stereo-Kamera können wir diese Parameter direkt ablesen. Die Kameraparameter unterscheiden sich in zwei Arten, die **extrinsische** und **intrinsische** Kameraparameter.

Extrinsische Parameter sind die Kameramatrix und die Distortionsmatrix. Die Kameramatrix ist eine 3x3-Matrix, wobei die Parameter f_x und f_y die Breite und die Höhe des Bildes in Pixels beschreiben, und die Parameter c_x und c_y den Fokus(Brennpunkt) der Kamera in Pixels beschreiben. Die Distortionsmatrix beinhaltete Information über die Verzerrung des Bildes wegen den Linsen. Wenn die Distortionsmatrix eine Nullmatrix ist, dann gibt es keine Verzerrung, welche der Fall für Realsense D435 ist.

$$\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

Intrinsische Parameter sind die Translationsmatrix und die Rotationsmatrix der Kamera. Der Translationsvector representiert die Position des Weltursprung in dem Kamera-Koordinatensystem, und die Rotationsmatrix kann als Richtung der Weltachsen in dem Kamera-Koordinatensystem interpretiert werden. Wie schon erwähnt, können wir in unserem Fall die beiden intrinsischen Parameter von der Kamera direkt auslesen.

Das heißt, wir benutzen bei diesem Projekt das Stereo-Bild nicht explizit, aber die Parameter, die wir benutzen werden auch für die Erstellung des Tiefenbildes verwendet. Die Abbildung 4 zeigt das Ergebnis aus der kompletten Bildverarbeitung. Der Code ist in der Github-Repository unter dem Link: https://github.com/onguntoglu/projekt_quad_wise1819

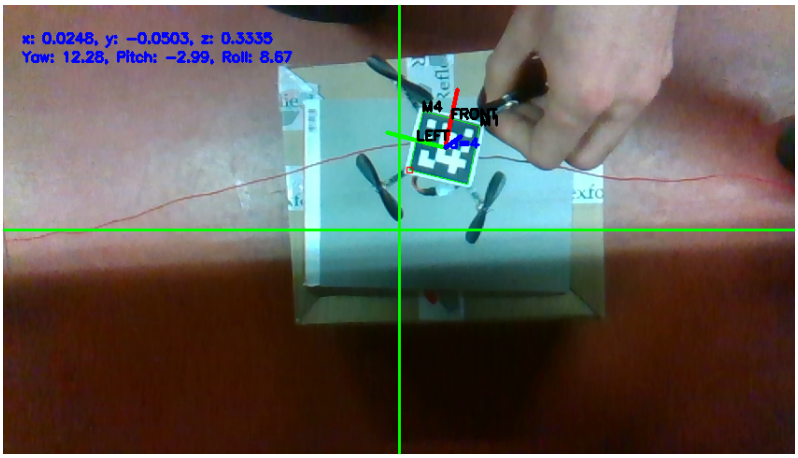

Abbildung 4: Bildverarbeitung mit ArUco-Markers

2.3 intrinsische Tait-Bryan-Winkel

Für eine gute Erkenntnis der Berechnungen von intrinsische Winkel, es muss zwischen extrinsische und intrinsische Drehung unterschieden. Wie in der Abbilding 5 gesehen, sind die extrinsische Drehungen um die feste ursprungliche Achsen, das heißt, dass alle Drehungen unabhängig von der Folge um die ursprungliche Achsen sind. An der Gegenseite, bei der intrinsischen Drehungen, ist die Folge wesentlich, da zwei gleiche Drehungen in umgekehrte Folgen ganz unterschiedliche Rotationen bezeichnen.

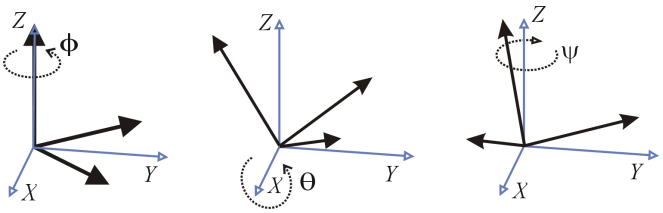

Abbildung 5: Drehung mit Euler Winkel $(\phi, \theta, \psi) = (45^o, 30^o, -60^o)$, in Z-X-Z extrinsische Konvektion[1]

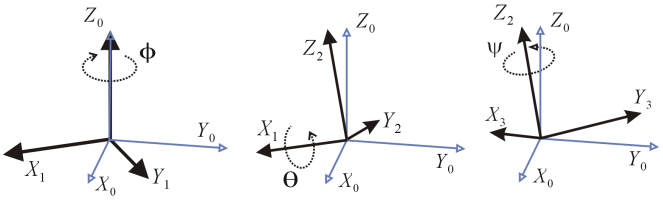

Abbildung 6: Drehung mit Euler Winkel $(\phi, \theta, \psi) = (-60^o, 30^o, 45^o)$, in Z-X'-Z" intrinsiche Konvektion[2]

In dem Projekt die Kameraachsen sind als die extrinsische Achsen zu verstehen, da diese Achsen fest und ursprunglich sind. Beziehungsweise die Roll-, Pitch- und Yaw-Achsen bezüglich des Quadrocopters sind die intrinsische Achsen. Der Marker liefert aber einen Rotationsvektor mit extrinsische Drehungen. Deswegen wird aus dem Rotationsvektor die intrinsischse Winkel berechnet.

Bei der intrinsischen Drehung läuft die nächstfolgende Drehungen um die neue rotierte Achsen. Jede extrinsische Drehung entspricht die intrinsische Drehung mit dieselbe Winkel aber in der umgekehrte Folge. Ein Beispiel ist in der Abbildungen 5 und 6 veranschaulicht. Die extrinsische Drehung für $(\phi, \theta, \psi) = (45^o, 30^o, -60^o)$ in Z-X-Z Konvektion liefert das gleiche Ergebnis wie der intrinsische Drehung für $(\phi, \theta, \psi) = (-60^o, 30^o, 45^o)$ in Z-X'-Z" Konvektion.

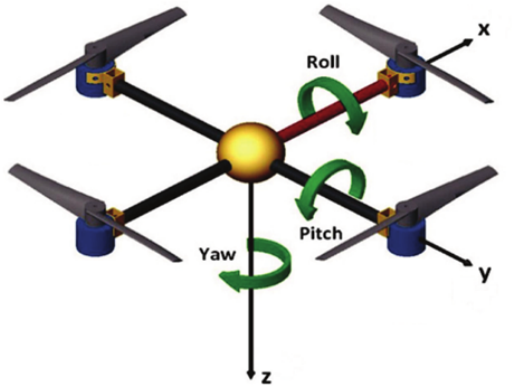

Abbildung 7: die intrinsische Achsen des Quadrocopters

Aus dem Rotationsvektor v_{ext} wird mittels Rodrigues-Formel die extrinsische Rotationsmatrix M_{ext} bestimmt.
Durch RQ-Decomposition der extrinsischen Rotationsmatrix werden elementare Rotationsmatrizen M_z, M_y, M_x für die ursprungliche Achsen erstellt.
Um die intrinsische Rotationsmatrix zu ermitteln werden die elementare Rotationsmatrizen in der umgekehrte Folge, also in der Z-Y'-X" Konvektion, multipliziert.
Aus der intrinsische Rotationmatrix M_{int} werden die intrinsische Winkeln Roll(ϕ), Pitch(θ) und Yaw(ψ) erhalten.

$$\text{RodriguesFormel}(v_{ext}) = M_{ext}$$
$$\text{RQ-Decomposition}(M_{ext}) = M_z M_y M_x$$
$$M_{int}(\phi, \theta, \psi) = M_x M_y M_z$$

3 Flugdynamik von Crazyflie 2.0

In der Formeln 1-3 wird das Modell des Quadrocopters bezüglich Thrust, Roll, Pitch und Yaw beschrieben. Die nichtlineare und die vereinfachte Modellierungen wurden von Liangs Masterarbeit[3] genommen. Die Flugdynamik des Quadrocopters hat folgende Form :

$$\ddot{z} = (cos\theta \cdot cos\phi) \cdot \frac{F}{m} - g \tag{1}$$

$$\ddot{x} = (sin\psi \cdot sin\phi + cos\psi \cdot cos\phi \cdot sin\theta) \cdot \frac{F}{m} \tag{2}$$

$$\ddot{y} = (-cos\psi \cdot sin\phi + sin\theta \cdot sin\psi \cdot cos\phi) \cdot \frac{F}{m} \tag{3}$$

Das Simulink-Modell mit der Übertragungsfunktion für Thrust ist in der Abbildung 8 dargestellt. Die Berechnung der Übertragunsfunktion wird in dem Teil Regelung erläutert. Die Roll und Pitch Winkel werden immer als kleine Winkel eingestellt, deshalb kann die Sinus und Kosinus Funktionen für kleine Winkel mit Taylor-Entwicklung vereinfacht.

$$sin(x) = x - \frac{1}{6}x^3 + \frac{1}{120}x^5 - .. \approx x$$
$$cos(x) = 1 - \frac{1}{2}x^2 + \frac{1}{24}x^4 - ... \approx 1 - \frac{1}{2}x^2$$

Das Modell vereinfacht sich zu:

$$\ddot{z} = (cos\theta \cdot cos\phi) \cdot \frac{F}{m} - g \tag{4}$$

$$\ddot{x} = (sin\psi \cdot \phi + cos\psi \cdot (1 - \frac{1}{2}\phi^2) \cdot \theta) \cdot \frac{F}{m} \tag{5}$$

$$\ddot{y} = (-cos\psi \cdot \phi + \theta \cdot sin\psi \cdot (1 - \frac{1}{2}\phi)) \cdot \frac{F}{m} \tag{6}$$

Dieses Modell wurde für einen konstanten Yaw Winkel = -90 linearisiert für eine vereinfachte Regelung der Höhe und Roll bzw. Pitch Winkel. Nach dem vereinfachten Modell hängt die x-Achse von der Roll Winkel und die y-Achse von der Pitch Winkel.
Die Einstellung der zur Crayzflie-Client gelieferte Winkel und Schubkräfte läuft als interne Regelung der Motor-Kräfte.

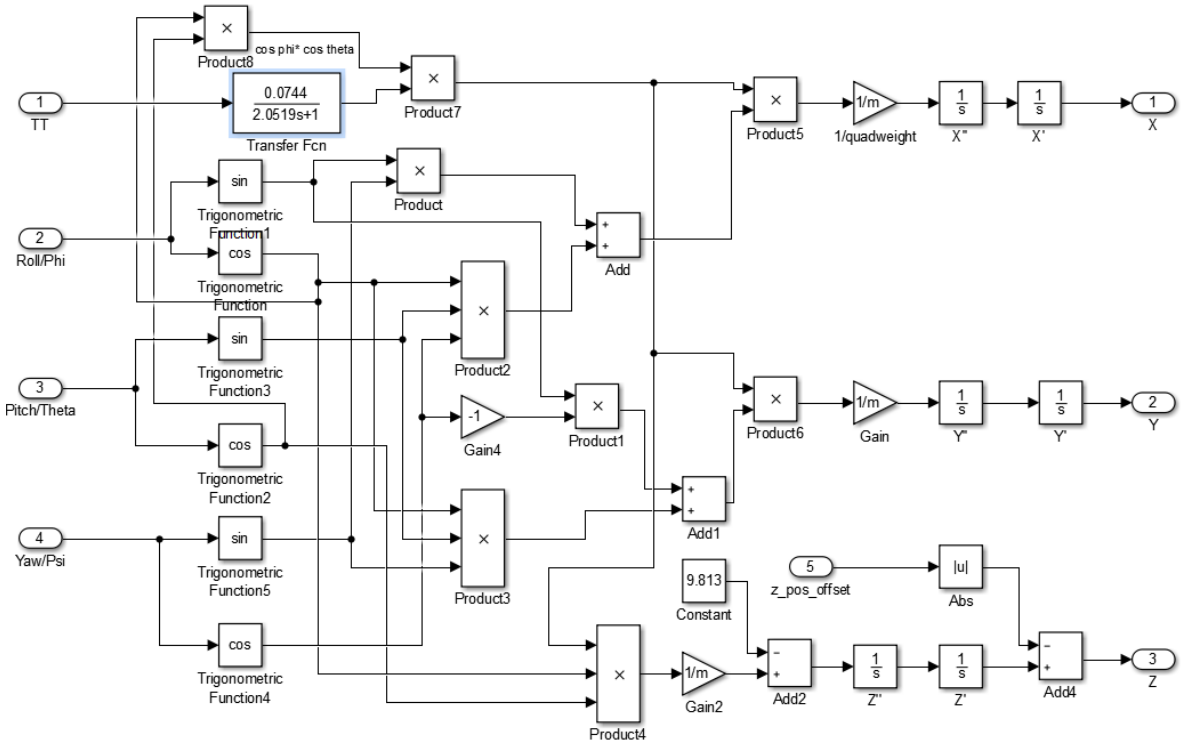

Abbildung 8: Die Quadrocopter-Block im Simulink

4 Regelung

4.1 System Identifikation

Für die Höhenregelung soll das System zwischen Höhe (z-Koordinate) und Thrust (T) identifiziert werden. Die Schubkraft überwindet für kleine Thrust Werte die Schwerkraft nicht, was zu keiner Änderung der Höhe z führt. Diesen Punkt, bei dem die Schubkraft die Schwerkraft überwindet heißt Hovering State. In diesem Punkt gilt:

$$T - m \cdot g = 0$$

Es wird auch angenommen, dass alle Winkel beim Hovering State gleich Null sind, weil für die Höhenregulung nicht von Interesse sind.
Nachdem der Quadrocopter lang genug in dem Sichtbereich des Kameras fliegen konnte, konnte Daten über Thrust und Z-Position ermittelt werden. In der Abbildung 9 wird der Zusammenhang von Thrust und Höhe gezeigt.

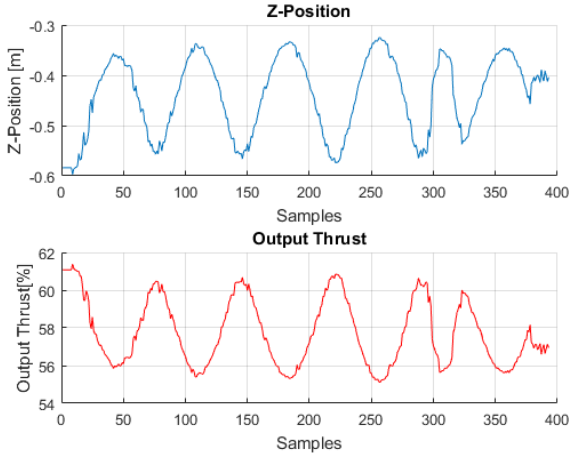

Abbildung 9: Die Schubkraft und Z-Positionierung

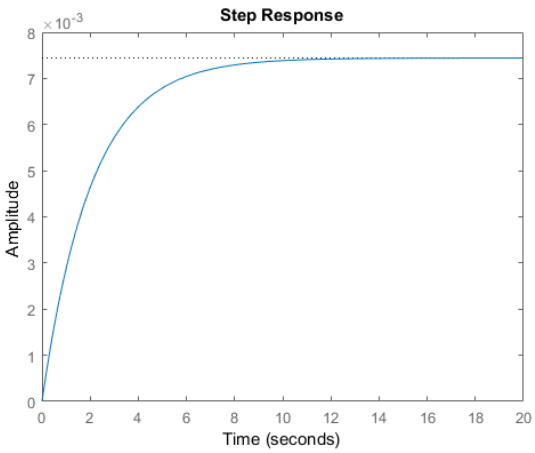

Abbildung 10: ermittelte Transfer-Funktion für Thrust und Z-Position

Aus dieser Daten wurde eine Übertragungsfunktion mittels Matlab System-Identification Tool berechnet. Die Sprungantwort der Übertragungsfunktion ist in der Abbildung 10 geplottet. Die Übertragungsfunktion lautet:

$$\frac{K}{sT+1} \text{ mit } K \approx 0.00744 \text{ und } T \approx 2.05189$$

4.2 Höhen-und Lageregelung

Die Regelung folgt mit drei PID-Reglern jeweils für Höhe-, Pitch(y-Achse)-und Roll(x-Achse) Regelung.

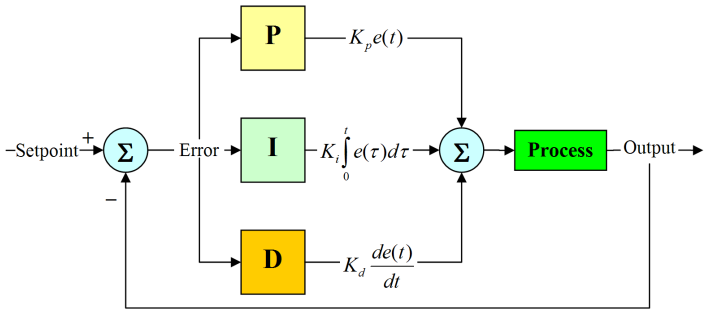

Abbildung 11: PID-Regler[4]

Die allgemeine Form der PID-Regler ist in der Abbilding 11 zu sehen. Erstens wird der Fehler zwischen der Stellgröße und dem Ausgang bestimmt. Proportionalanteil bestimmt wie stark der Regler zum entsprechenden Fehler reagiert, während der Integralanteil alle vergangene Fehlern summiert. Der D-Anteil reagiert auf der Ableitung des momentanen Fehlers. Da die Fehlern für alle Reglern in Weltkoordinaten gegeben werden, reagiert der D-Anteil als momentan ausgegebene Geschwindigkeit.
Die Höhenregelung reagiert auf den Fehler der Höhe, also der z-Koordinate. Die Pitchregelung auf den Fehler der y-Koordinate und die Rollregelung der x-Koordinate. Die Eingangsgrößen sind für alle Reglern in Meter-Einheiten gegeben. Die Ausgang der Höhenregelung wird zu Crazyflie-Client als Prozent und die Winkel als Grad geliefert. Bei der Einstellung der Abtastrate der Reglern muss geachtet werden, dass die Abtastrate des Kameras nicht überschritten wird. Infolge wird die Abtastrate für alle drei Reglern als 40 Hz gewählt für eine möglichstschnelle Regelung.
Die notwendige Schubkraft für die Hovering-State ist ca. %59 der gesamte Schubkrafts. Das System des Quadrocopters reagiert zur schnelle Veränderungen sehr aggresiv und tritt aus dem Sichtbereich des Kameras aus. Deshalb werden die P-I-D Anteile so gewählt, dass zur Crazyflie gelieferte Werte möglichst kleine Veränderungen haben. Um eine lange Anstiegszeit in der Höhenregelung zu vermeiden, werden in Addition zur PID-Regler eine Feedforward-Konstante von %59 Schubkraft benutzt. Der Höhenregler kompensiert die notwendige Schubkraft für die Stellgröße und idealerweise stellt die benötigte Schubkraft für die Hovering-State.

Die Crazyflie, die in diesem Projekt benutzt wird, hat wesentliche Instabilitäten, die die Pitch und Roll Winkel beeinflüssen. Die Gründe könnte an der Ungleichgewicht an der Lage des Akkus oder der Rotoren liegen. Die Pitch- und Rollreglern können am Fluganfang diese Instabilitäten nicht schnell genug überwinden, da beim Fluganfang auch die Schubkraft noch nicht stabil ist, welche die Lageregelung sehr stark beeinflusst. Deshalb wird bei Pitchregler ein Feedforward-Konstante von -0.25^o und bei Rollregler von 0.25^o benutzt. Das Blockschaltbild der Regelung im Simulink sieht wie in der Abbildung 12 aus.

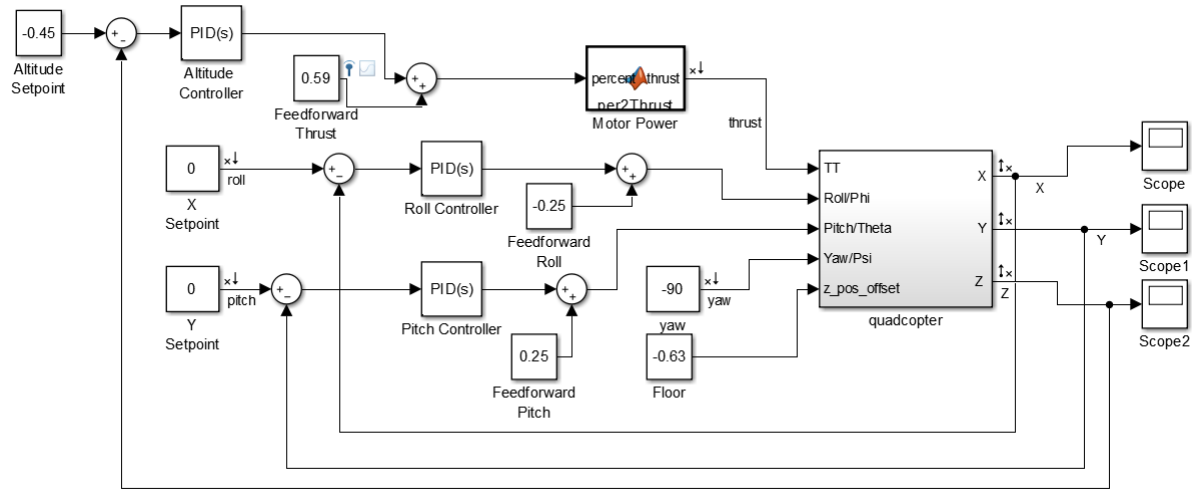

Abbildung 12: Das Blockschaltbild der Regelung

Die Simulation liefert leider keine bemerkungsvolle Ergebnisse, welche zu einer falschen Identifikation der Schubkraft-Höhe System zurückzuführen ist. Deshalb werden die Reglerkoeffizienten nach vielen Versuchen und Auswertungen wie folgt eingestellt:

Höhenregelung: $P_z = 0.03$, $I_z = 0.03$, $D_z = 0.075$, $Feedforward_z = 0.59$

Ausgang-Limitierung$_z$ $= [0.585, 0.625]$

Pitchregelung: $P_\theta = 30$, $I_\theta = 10$, $D_\theta = 15$, $Feedforward_\theta = 0.25^o$

Rollregelung: $P_\phi = 12$, $I_\phi = 8$, $D_\phi = 5$, $Feedforward_\phi = -0.25^o$

5 Ergebnisse und Auswertung

Die folgende Abbildungen 13, 14 und 15 gehören zu demselben Versuch, die auch als Video(output2.avi) in der Github-Repository unter dem folgenden Link verfügbar ist: https://github.com/onguntoglu/projekt_quad_wise1819
Als Schutz wurde der Quadrocopter an der Seiten mit Seilen verbunden, die die Bewegung in der x-Achse wesentlich begrenzen. Allerdings die Wirkungen der Seilen sind sehr wenig auf der Bewegungen in der y- und z-Achse.

5.1 Bilverarbeitung

In dem Versuch wurde der Copter ca. 60 cm entfernt vom Kamera platziert. In der Entfernung funktioniert die Erkennung von AR-Code sehr gut mit der Abtastrate 60 Hz obwohl eine Auflosüng von 848x480 verwendet wurde. Die Kameraabtastrate reicht sicherlich für eine präzise Regelung. Für eine Entfernung mehr als 80 cm wurde der Kamera nicht geprüft. Für weitere Entfernungen wurde eine bessere Auflösüng notwendig, was in dem Projekt mit Python nicht eingestellt konnte, da die Abtastrate der Kamera bei höhere Auflösüngen wesentlich klein für eine gute Regelung war.

5.2 Höhenregelung

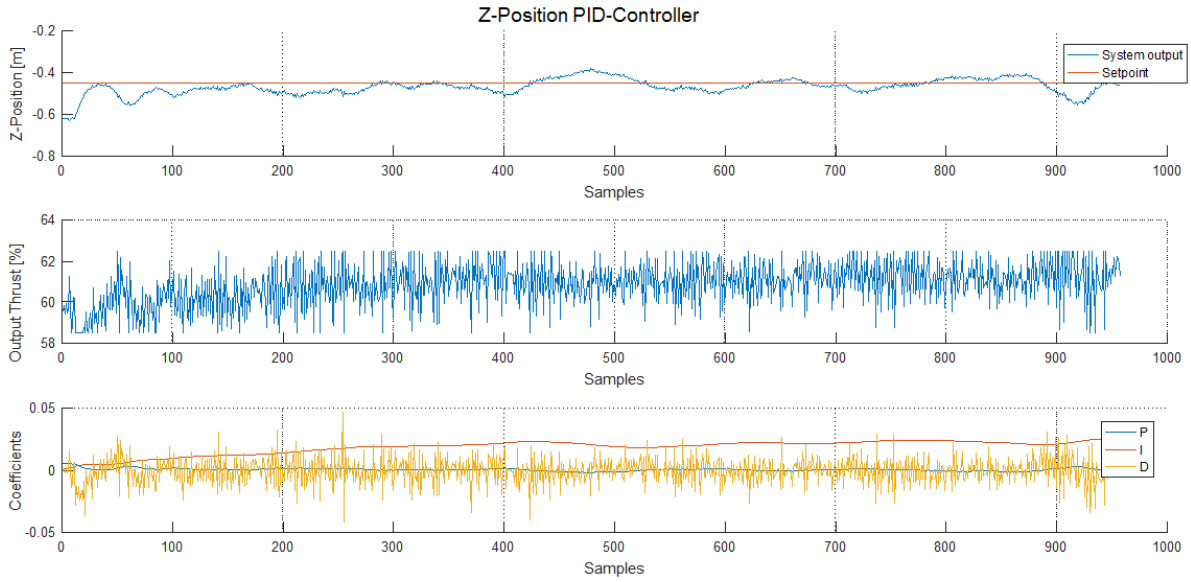

Abbildung 13: Z-Position

Bei dem Versuch steht die Crazyflie am Anfang auf dem Boden, ca. 63 cm entfernt vom Kamera. Die Stellgröße ist 45 cm, also der Copter muss sich ca. 18 cm hochheben. In der Abbildung 13 wird jeweils die Lage des Quadrocopters an der z-Kordinate, die eingestellte Schubkraft und die Einfluss von P-I-D Anteile veranschaulicht. Die Abbildung zeigt, dass der P-Anteil die schwächste und der D-Anteil die stärkste Wirkung hat. Am Anfang schwingt der Quadrocopter stark, da der Fehler innerhalb eine Sekunde fast aufgehoben aber die notwendige Schubkraft für Hovering-State noch nicht erreicht ist. Mit der Zeit verkleinert sich die Schwingung, da der I-Anteil zur Ermittlung der Hovering-Schubkraft hilft. Obwohl eine Schwingung von -3, +3 cm um die Stellgröße gibt, die Höhe ist befriedigend ausgeregelt, also der Regler zeitigt einen Erfolg.

5.3 Lageregelung

5.3.1 X-Positionierung mittels Roll Winkel

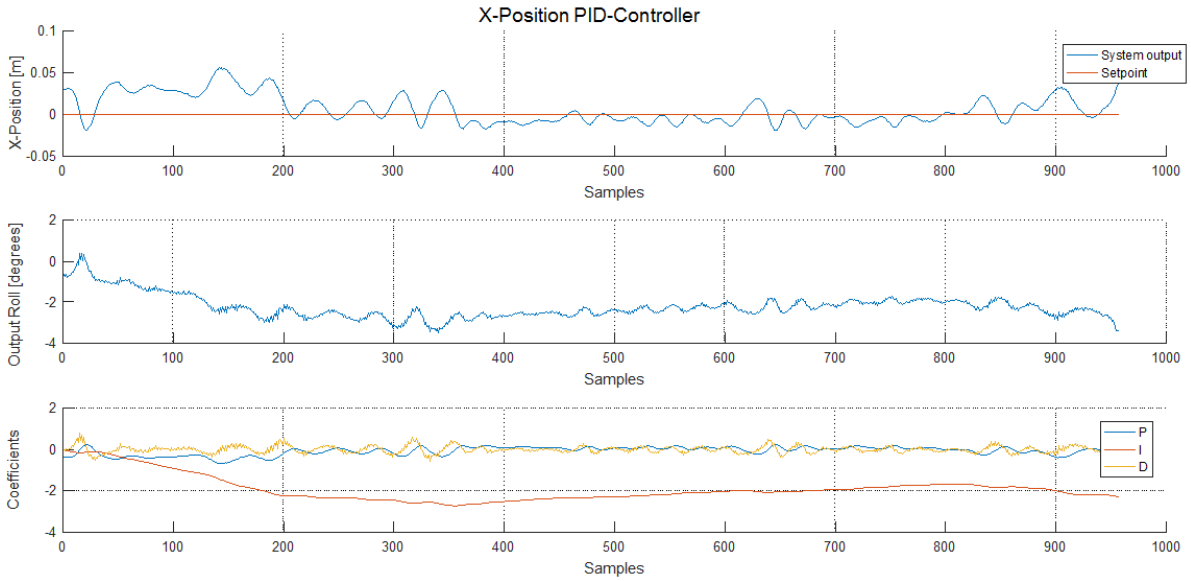

Abbildung 14: X-Position

Bei der X-Position ist die Stellgröße der Ursprung. Wie vorhererhwähnt ist der Copter an der Seiten mit Seilen verbunden, die die Bewegung in der x-Achse deutsam begrenzen. Der Copter kann in der x-Achse maximal +5 cm und minimal -5 cm erreichen. Es sind große Schwingungen zu Beginn auffällig. Die Schwingungen entstehen wegen der Instabilität des Copters nach dem Start der 4 Motoren. Am Anfang erreicht der Copter seine maximale Begrenzung +5 cm aber ab diesem Punkt werden die Begrenzungen bis die Auschaltung des Copters niemals erreicht. Es könnte sein, dass der Copter an

diesem Zeitpunkt, wo er die maximale Begrenzung erreicht, ohne die Seilen nicht schnell genug der Fehler abdecken und aus dem Sichtbereich der Kamera austreten könnte. Trotzdem wird ab dem entsprechenden Zeitpunkt der Roll-Winkel gut genug geregelt mit minimale Abweichungen der X-Lage um der Ursprung. Wenn auf die P-I-D Anteile Rücksicht genommen wird, es zeigt sich, dass der P- und D-Anteile nicht stark genug auf den Fehler reagiert haben, um gegen die momentane Fehler schnell zu kompenzieren. Trotzdem erreicht der I-Anteil schnell einen relativ stabilen Wert um die Instabilität an der x-Achse auszugleichen. Wie in der Abbildung gesehen ein Roll-Winkel von ca. 2^o stabilisiert den Copter an dem Ursprung mit wenig Schwingung. Es ist schwer zu bestimmen, was diese Instabilität verursacht, da auch Einflüsse der Seilen vorhanden sind.

5.3.2 Y-Positionierung mittels Pitch Winkel

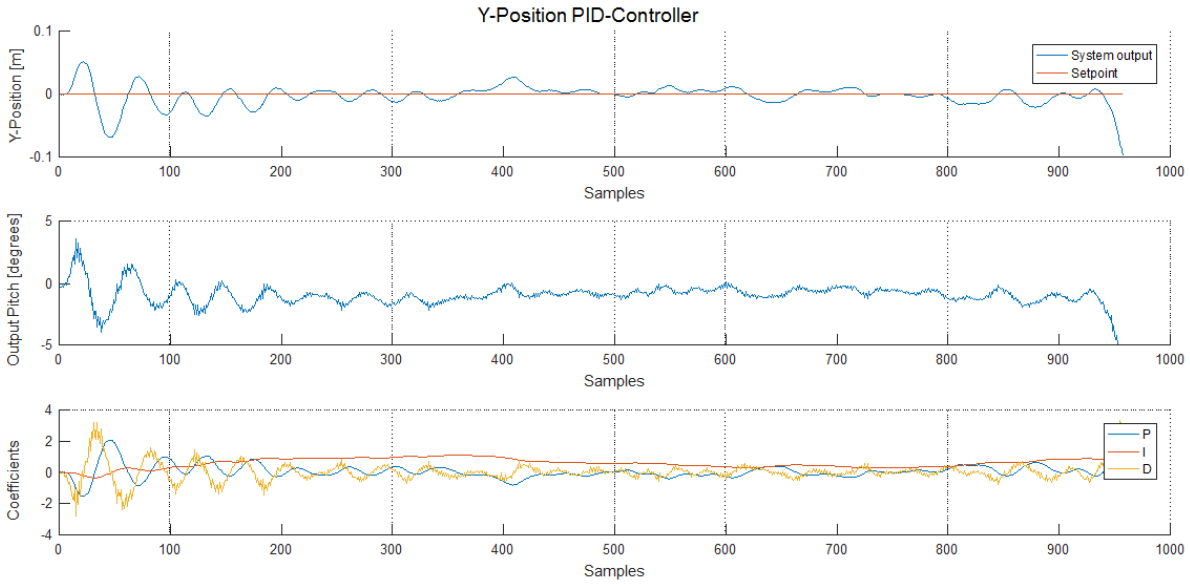

Abbildung 15: Y-Position

Die benutzte Seilen beeinflussen der y-Achse fast gar nicht und die Ergebnisse können ohne Zweifel wahrgenommen werden. Bei der Y-Position ist die Stellgröße wieder der Ursprung. Am Anfang enstehen größe Schwingungen, da parallel auch die Höhenregelung läuft, welche die instabile Veränderungen der Schubkraft verursacht. Anstatt die zum Beginn entstehende Abweichungen, verkleinern sich die Amplitude der Schwingungen deutlich mit der Zeit, da eine stabile Schubkraft erreicht wird. Infolge kann der Pitch-Regler den Fehler einfach abdecken. Der Copter hatte ohne Pitchregelung eine Neigung

in der negative Richtung an der y-Achse zu fliegen. Wie gesehen die P- und D-Anteile reagieren sehr schnell gegen die momentane Fehlern, während der I-Anteil einen konstante Wert erreicht, welche gegen die entsprechende Neigung wirkt.

6 Erweiterungsmöglichkeiten

Zu dem Projekt gibt es noch viele Erweiterungsmöglichkeiten, wie eine höhere Auflösüng der Kamera, die auch eine höhe Abtastrate ermöglicht. Mit einer höheren Auflösung könnte theoretisch die Bildverarbeitung für höhere Entfernungen ermöglicht.

Zur Regelung könnte andere Reglerentwurfsverfahren wie RST-Reglerentwurf und LQ-Regler untergesucht werden, die für die Regelung von Crazyflie bessere Ergebnisse liefern können.

Zu den Erweiterungsmöglichkeiten gehört auch das Korrigieren der Yaw-Drifts. Da der Yaw Winkel stark driftet, wird die Regelung des Copters noch schwerer und führt auch dazu, dass der Copter schnell aus dem Blick der Kamera kommt. Es wurde versucht eine Yaw-Regelung zu realisieren, aber leider ohne Erfolg. Dieses Problem kann korrigiert werden, indem ein stabiler Copter verwendet wird.

7 Zusammenfassung

In diesem Projekt wurde ein Quadrocopter Crazyflie 2.0 für die Höhe- und Lageregelung untersucht. Die Verbindungen zwischen der Geräte und verschidenen Pythonskripte laufen problemlos für die Erstellung eines erfolgreichen Regelkreises. Die Regelung der Höhe und der Y-Positionierung erfolgt gut aber es ist schwer eine zweifellose Schlussfolgerung über die X-Poisitonsregelung zu ziehen. Obwohl beim Fluganfang starke Instabilitäten auftreten, überwinden die Reglern diese schnell und der Copter erreicht eine stabile Lage, inder nur schwache Schwingungen auftreten. Trotzdem ist die ganze Verhaltensregelung noch nicht erreicht, da die Yaw-Regelung noch nicht zur Verfügung steht. Mit der Berücksichtigung der Erweiterungsmöglichkeiten könnte die Regelung des Quadrocopters noch verbessert werden.

Literatur

[1] https://commons.wikimedia.org/wiki/File:EulerX.png

[2] https://commons.wikimedia.org/wiki/File:EulerG.png

[3] Liang Shi, "Positionsregelung Eines Nano-Quadrocopters", 11.06.2015

[4] https://www.controleng.com/wp-content/uploads/sites/2/2014/08/CTL1408_WEB_F1_PID-Valin_Fig1_PID-control_loopslider.jpg