Regelung eines Quadrocopters über eine Scilab-Xcos Schnittstelle

Stefan Fabian Ruppin, Fachgebiet Regelungssysteme, TU Berlin
stefan.f.ruppin@campus.tu-berlin.de
Dr. Thomas Schauer, Fachgebiet Regelungssysteme, TU Berlin
schauer@control.tu-berlin.de

Abstrakt

Ziel dieser Arbeit ist die Kombination eines Open Source Quadrocopters mit dem Open Source Scilab-Tool Xcos für die Realisierung einer Indoor-Höhenregelung. Xcos ermöglicht das flexible Zusammenschalten verschiedener Blöcke mit unterschiedlichen Funktionen zu großen Systemen. Um eine Höhenregelung im Raum über Xcos zu realisieren, wurden Blöcke zur Steuerung (Tastatureingaben) und zur Befehlskommunikation mit dem Quadrocopter in C++ implementiert. Als Umgebung wurde Ubuntu 13.10 mit der HART Toolbox benutzt. Des Weiteren wurde eine kamerabasierte Positionsbestimmung mithilfe der Microsoft Kinect Kamera realisiert und in Xcos integriert. Anschließend wurde die Höhendynamik experimentell identifiziert und ein diskreter RST-Regler implementiert. Die Qualität des Reglers wurde abschließend in einem Testflug validiert.

1 Gesamtsystem

In Abbildung 3 ist der Aufbau des Gesamtsystems zu sehen. Die zentrale Einheit stellt ein Computer mit Scilab/Xcos [7] dar. An diesem werden die Kamera sowie das Funkmodul Crazyradio [4] per USB angeschlossen. Die Integration dieser Hardware in Xcos erfolgt durch den Entwurf eigener Blöcke, das beinhaltet die Berechnungsroutine sowie das Interface. Auf diesem Computer soll später auch die Höhenregelung und Steuerung über die Tastatur erfolgen.
Die getätigten Berechnungen werden zu Befehlen umgesetzt und diese an das Funkmodul Crazyradio übermittelt. Dieses überträgt die Befehle mithilfe eines eigenen Kommunikationsprotokoll [3] über einen $2,4$ GHz Funkkanal schließlich zum Quadrocopter Crazyflie. Der Quadrocopter versucht die empfangenen Befehle anschließend umzusetzen. Dafür besitzt er eine eigene interne Lageregelung, welche die Sollwerte realisiert. Die interne Lageregelung wird mit der originalen Firmware des Quadrocopters geliefert.
Die Kamera wird währenddessen Bilder des Quadrocopters aufnehmen und dieses per USB an das Xcos-Diagramm übertragen. Über einen farbigen Marker am Quadrocopter

kann dann die Position im Raum ermittelt werden. Die Positionsinformation kann dann zur Bestimmung der nächsten Steuersignale verwendet werden.

Abbildung 1: Crazyflie (mit Marker)

Abbildung 2: Crazyradio

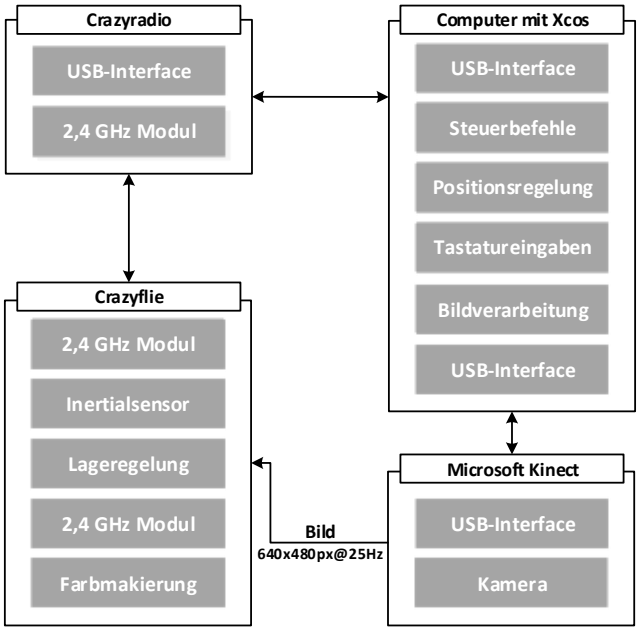


Abbildung 3: Schema des Gesamtsystems

2 Schnittstellenblöcke in Xcos

Für die direkte Kontrolle eines Quadrocopters über Xcos war es nötig, eigene Blöcke zur Kommunikation und Positionsbestimmung zu entwickeln. Die Blockentwicklung wurde mithilfe der HART Toolbox von Holger Nahrstaedt [9] unter Ubuntu 13.10 durchgeführt. Dabei wurden drei neue Blöcke entwickelt.

2.1 Crazyflie-Block

Dieser Block dient der Kommunikation mit einem 9×9 cm großen Quadrocopter des Typs Crazyflie, wie er von der Firma bitcraze [2] hergestellt und verkauft wird. Der Block basiert auf der C++-Bibliothek libcflie von Jan Winkler [10]. Realisiert wurde das Empfangen von Messwerten der Quadrocopter-internen Sensorik und das Senden von Sollwinkelvorgaben für Roll, Pitch, Yaw sowie einer Schubvorgabe.

2.2 Kamerablock

Zur Positionsbestimmung im Raum wurde eine Microsoft Kinect Kamera mithilfe eines Blockes in Xcos integriert. Der Block stellt eine Verbindung mit der Kinect her und kann das aufgenommene Bild nach einen bestimmten Farbmaker durchsuchen und diesen dann verfolgen. So wird eine Poisitionsbestimmung im Raum ermöglicht, welche 3D Koordinaten des Quadrocopters liefert. Dies ermöglicht eine Positionsermittlung mit 25-30

Hz direkt in Xcos. Die Funktionsweise der Positionsbestimmung kann in Abbildung 4 nachvollzogen werden. Zur Ansteuerung der Kinect unter Ubuntu wurde das 3D-Sensor-Framework OpenNI [6] benutzt und zur Bildverabeitung unter C wurde OpenCV [5] verwendet.

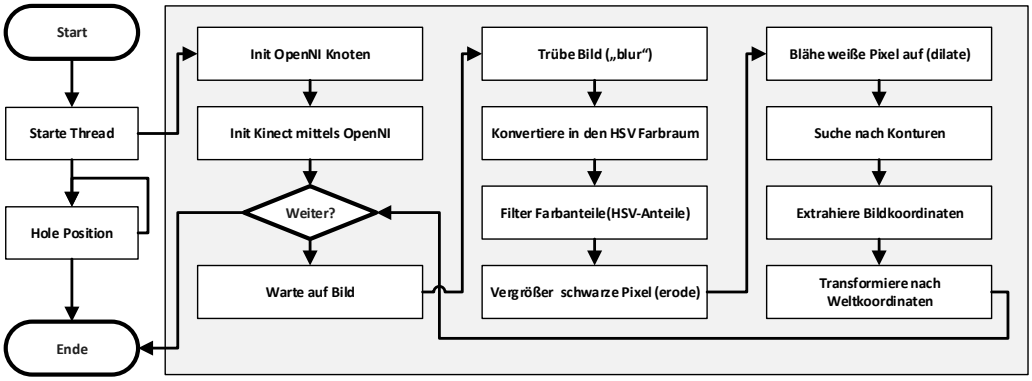


Abbildung 4: Ablauf der Positionsbestimmung

2.3 Tastaturblock

Als letztes wurde noch ein Tastaturblock entwickelt, der es ermöglicht, mit dem Xcos Diagramm während der Simulation direkt zu interagieren. Dies ermöglicht das Steuern des Quadrocopters sowie die Kontrolle von Diagrammparametern wie z.B. der Reglerparameter. Als Basis für die Implementierung wurde die C++-Bibliothek Allegro [1] verwendet, da sie das einfache Erstellen von kompletten Tastaturabbildern zulässt.

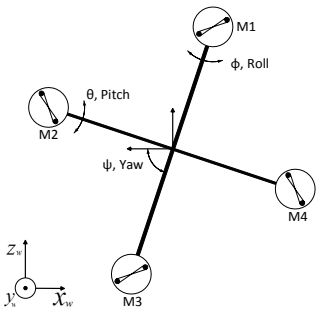
3 Modell und Regler für die Höhendynamik

Für die Identifikation der Höhendynamik wird ein mathematisches Grundmodell gesucht, welches in der Lage ist, die Systemausgänge mit den Systemeingängen zu verbinden. Als Systemeingang für die Höhenregelung sind die Neigungswinkel „Roll“ und „Pitch“ sowie der Schub T relevant. Es wird also ein Modell gesucht, welches eine Verbindung zwischen diesen drei Systemeingängen und dem Systemausgang, der Höhe y_w, herstellt.

Eine ausführliche Literaturrecherche hat gezeigt, dass es ausreichend ist, für eine Positionsregelung den Quadrocopter als Punktmasse zu betrachten. Das Modell, welches Denys Bohdanov in seiner Masterarbeit „Quadrotor UAV Control for Vision-based Moving Target Tracking Task“ [8] benutzt, nimmt den Quadrocopter als Punktmasse an und setzt die Neigungswinkel und den Schub mit der Höhe in Verbindung.

Für den Quadrocopter sollte mit obrigen System die Höhendynamik geregelt werden. Dazu wurde das Modell (3) geeignet entkoppelt. Dabei wurden die Neigungswinkel des Quadrocopters für den Schwebefulg als nahezu Null angesehen ($\phi = \theta = 0$). Über eine

$$m\ddot{x}_w = (\sin\psi\sin\theta + \cos\psi\cos\theta\sin\phi)\cdot T \quad (1)$$
$$m\ddot{z}_w = (-\cos\psi\sin\theta + \sin\phi\sin\psi\cos\theta)\cdot T \quad (2)$$
$$m(\ddot{z}_w + g) = \cos\theta\cos\phi\cdot T \quad (3)$$



experimentelle Systemidentifikation wurde dann nach dem passendem Verstärkungsfaktor gesucht und folgendes finales Modell für den Quadrocopter Crazyflie erhalten.

$$\ddot{y}_w = c\cdot T - g \quad (4)$$
$$c = 0,0001994\ \frac{\mathrm{m}}{\mathrm{s}^2} \quad (5)$$

Für den Reglerentwurf wurde der Doppelintegrator (4) zunächst mit einer Abtastfrequenz von 25Hz zeitdiskretisiert. Anschließend wurde ein digitaler RST-Reglerentwurf, wie er in „Computer-Controlled Systems: Theory and Design“ [11] beschrieben ist, mit Integralanteil, Stellgrößenbeschränkung und Anti-Wind-Up [11] durchgeführt und in Simulationen verifiziert. Die allgemeine Regelerstruktur ist in Abbildung 5 dargestellt.

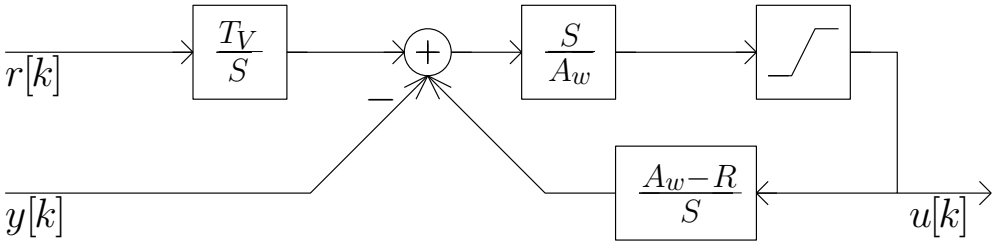


Abbildung 5: Regelerstruktur nach [11]

4 Testflug zur Entwurfsvalidierung

Die Funktionstüchtigkeit des Gesamtsystems mit allen entwickelten Blöcken sollte verifiziert werden. Dazu wurde ein Versuchsstand errichtet und ein Testflug mit aktiver Höhenregelung durchgeführt. Dabei wurden Messwerte aufgenommen, mit denen sich das Ergebnis bewerten lässt.

Ein Test des Reglers in der Realität zeigt, dass der Regler in der Lage ist, den Quadrocopter sauber auf Referenzsprünge reagieren zu lassen. Der Regler ist in der Lage, die Höhe des Quadrocopters zu stabilisieren und um einen Arbeitspunkt zu regeln. Die Systemantwort bei einer Änderung der Höhenreferenz um -20 cm ist in Abbildung 6 zu sehen. Ausgangspunkt für den Sprung ist eine Flughöhe von $y_w = 18\ cm$ über der Kameramitte (Höhe im Kamerakoordinatensystem). Dabei wird eine Anstiegszeit von $t_r = 2,76$ s erreicht. Dies ist im Vergleich mit der Simulation und der im Entwurf angesetzten Anstiegszeit von 3 s geringfügig schneller. Diese Abweichung könnte sich damit erklären

lassen, dass die initiale Flughöhe des Quadrocopters sich im Testflug ungefähr 1 cm unter dem Wert der Sollgröße befindet und der Regler seine Vorgabe somit schneller erreichen kann. Vergleicht man Messung und Simulation jedoch direkt (s. Abbildung 7), ist zu er-

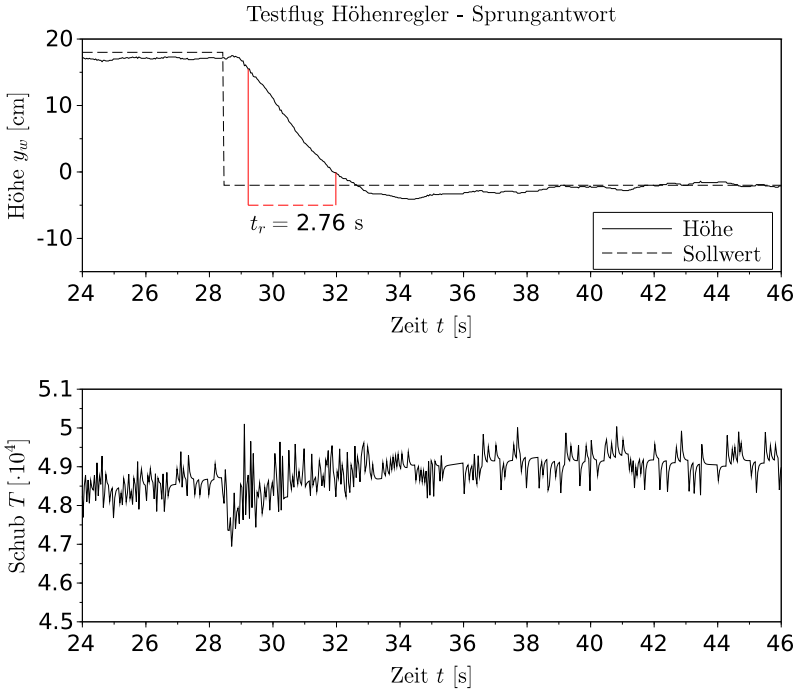


Abbildung 6: Testflug: Sprungantwort

kennen, dass die Starthöhe keinen großen Einfluss hat, sondern der Quadrocopter in der Realität schneller zu sinken scheint und schwerer abzufangen ist. Eine mögliche Ursache hierfür könnte der eigene Abwind (engl. down-dash) des Quadrocopters sein, welcher das so genannte Wirbelringstadium hervorruft. Hierbei sinkt der Quadrocopter in seinem eigenen Abwind und wird durch diesen zusätzlich beschleunigt. Dies resultiert während des horizontalen Singfluges in einem Auftriebsverlust und erfordert folglich mehr Schub für das Bremsmanöver. Dieses Verhalten ist in der Simulation nicht abgebildet. Eine andere mögliche Ursache wäre eine Ungenauigkeit in der Parameteridentifikation.
Das Unterschwingen im Testflug lag bei $\approx 10.5\%$ und damit deutlich höher als in der Simulation, wohingegen das Überschwingen bei positiven Sollgrößensprüngen bei nur $\approx 7\%$ lag.

5 Zusammenfassung

Unter Verwendung von Open Source Hard- und Software konnte eine preiswerte Quadrocopter-Versuchsanordnung realisiert werden. Der Entwurf und die Testung der beschriebenen Höhenregelung eignen sich gut als Bestandteile eines Praktikumsversuches in der universitären Lehre. In Zukunft sollen Mehrgrößenregler für die Realisierung von Flugmanövern entwickelt werden. Ferner ist die Ansteuerung und Regelung mehrerer Quadrocopter im Schwarm denkbar.

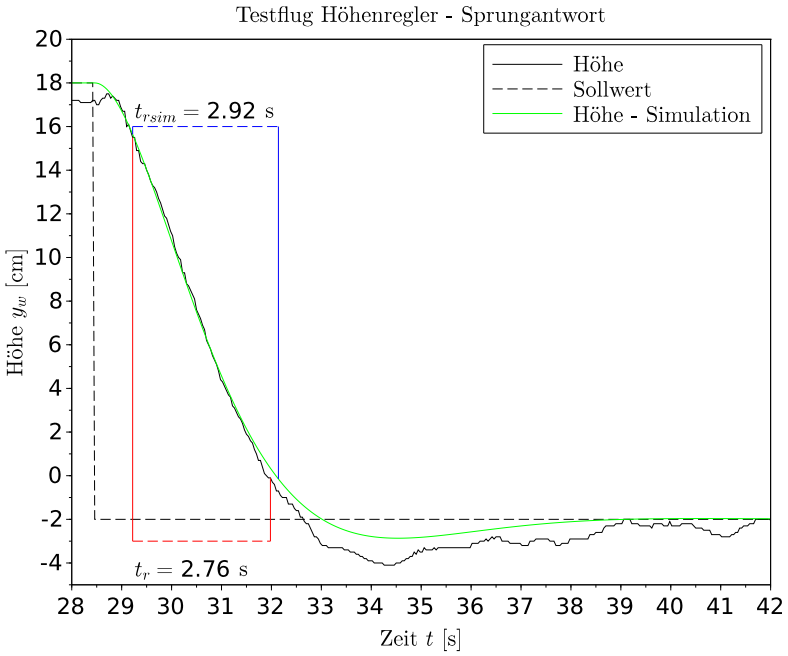


Abbildung 7: Sprungantwort: Testflug vs. Stimulation

Literatur

[1] *Allegro*. `http://alleg.sourceforge.net/`, Abruf: 15.04.2014 22:15

[2] *bitcraze*. `http://www.bitcraze.se/`, Abruf: 15.04.2015 22:05

[3] *Crazy RealTime Protocol*. `http://wiki.bitcraze.se/projects:crazyflie:crtp`, Abruf: 15.04.2014 22:31

[4] *Crazyradio*. `http://www.bitcraze.se/crazyradio/`, Abruf: 15.04.2015 22:05

[5] *OpenCV Homepage*. `http://opencv.org/`, Abruf: 15.04.2014 21:45

[6] *Openni Nite Homepage*. `http://www.openni.org/files/nite/`, Abruf: 15.04.2014 22:00

[7] *Scilab*. `http://www.scilab.org/scilab/features/xcos`, Abruf: 15.04.2015 22:07

[8] Bohdanov, Denys: *Quadrotor UAV Control for Vision-based Moving Target Tracking Task*, University of Toronto, Diplomarbeit, 2012

[9] Nahrstaedt, Holger: *HART Toolbox sourceforge page*. `http://hart.sourceforge.net/`. Version: 2008-2013, Abruf: 20.04.2014 14:18

[10] Winkler, Jan: *libcflie*. `https://github.com/fairlight1337/libcflie`. Version: 2013, Abruf: 13.04.2013 18:24

[11] Åström, Karl J. ; Wittenmark, B.: *Computer-Controlled Systems: Theory and Design, Third Edition*. Dover Publications, 2011 (Dover Books on Electrical Engineering Series). – ISBN 978–0–486–48613–0