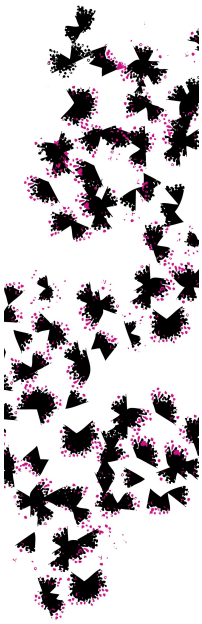

Design of a contact-mode controller for applying a notable force on a surface using an aerial manipulator

J.J. (Jim) Hoekstra

BSc Report

Committee:
H.W. Wopereis MSc
Dr.ir. M.A. van der Hoef
Dr. R. Carloni

May 2016

009RAM2016
Robotics and Mechatronics
EE-Math-CS
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

Contents

1 Introduction **3**

2 Preliminary Experiments **4**

3 System Specifications **5**

4 Dynamics During Physical Contact with a Vertical Surface **7**
4.1 Exerted Force 8
4.2 Rotations in the Horizontal Plane 8
4.3 Rotations in the Vertical Plane 12

5 Controller Architecture **12**
5.1 Contact Yaw Controller 12
5.2 Contact Pitch Controller 15
5.3 Pitch Controller 16

6 Simulation of the System **16**
6.1 Simulation of the Dynamics 16
6.2 Simulation of the Controller 17

7 Experiments **18**
7.1 Rewriting the Controller 18
7.2 Testing the Controller 20

8 Discussion of the Results **22**

9 Conclusion **23**

10 Future Work **23**
10.1 Suggestions for future research 23
10.2 Including the effect of friction in the simulation 24

Abstract

One of the main tasks in research towards aerial robotics is to enable the use of multirotor UAVs for dynamic aerial interaction and airborne maintenance tasks. This work presents a novel control algorithm for stabilizing multirotor-UAVs applying high physical contact forces on the environment. By combining state feedback on the roll and yaw angle, and making use of the dynamics of physical contact, an integrated controller is designed and tuned using the LQR method. Experiments have been performed using this controller, during which a contact force of more than 15 N was exerted against a vertical surface for several minutes, demonstrating the effectiveness of the algorithm.

1 Introduction

Unmanned Aerial Vehicles (UAVs) are playing an increasingly bigger role in society. The technology has become much cheaper in recent years and therefore more suited to a wide range of applications. For instance, Amazon is experimenting with using UAVs to deliver packages within 30 minutes. Other examples are the use of UAVs in rescue operations [1], inspection of structures such as power lines [2], aerial photography etc..

The interaction between UAVs and their environment is a new and rapidly growing research field [3]. The challenge is to enable UAVs to perform tasks that are difficult or dangerous for humans, such as maintainance or cleaning work at high altitude. Having UAVs to perform these tasks is expected to reduce the cost of maintainance, as well as the risk of injury to humans [4].

The European H2020 Project AEROWORKS focuses on developing a team of aerial robotic workers, equiped with manipulators, that can perform maintenance procedures. For these procedures, airborne manipulation of the environment is required. Manipulation of the environment can take several forms, such as the gripping and transporting of objects [5], the assembly of a structure [6], or the application of a force against the environment [7]. This research is a part of the Aeroworks project, and the focus of this part is on applying a force to an environment using a UAV. The application of a force to an object is relevant for tasks like cleaning, coating, and inspecting objects and surfaces.

Previous research has focused on stabilizing a UAV while exerting a force against a vertical wall through a manipulator [3, 4, 7, 8]. Although positive results were achieved, the contact forces remained relatively small (around 2 Newton). In a study on impacts between a UAV and its environment, a contact force of up to 50 Newton was reached, but it could not be held for longer than a fraction of a second [9].

The aim of this research is to improve the control algorithms for UAVs in physical interaction with their environment, in order to ensure stability under significant contact forces for extended periods of time.

2 Preliminary Experiments

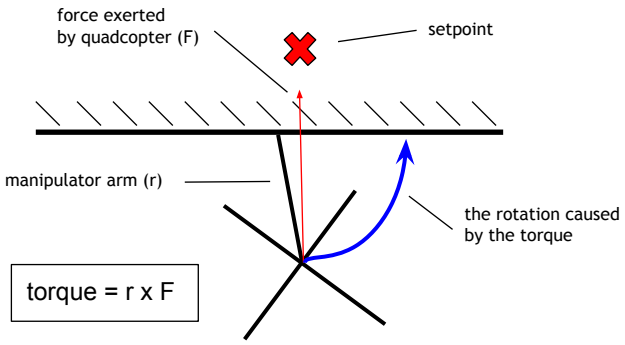

Figure 1: Illustration of the torque arising when the setpoint is not in line with the manipulator arm

We have performed preliminary experiments. In these experiments it was attempted to exert a force against a wall using the control algorithms normally used for free flight, i.e. position and attitude control. It was observed that once a significant force was applied (> 2N) to the wall, the UAV became instable in yaw direction and started turning towards the wall. Two possible explanations for this instability are offered here, which are useful in developing a new control approach:

- Because a position controller is used, with a setpoint slightly into the wall, the exerted force is no longer necessarily in line with the manipulator arm when the UAV's yaw angle is unequal to zero. This causes a torque around the contact point between the manipulator and the wall in yaw direction (see figure 1). Note that this moment will only increase once the UAV starts rotating.
- The same attitude controller is used in free flight and in contact mode. However, this controller may not have been powerful enough to correct the yaw angle once it starts deviating from its setpoint, because the moment of inertia in contact mode around the contact point is much greater than the moment of inertia in free flight around the center of the UAV (Steiner's theorem). As a test, it was attempted to increase the gains on the yaw PD controller, so the controller would respond to an error with a larger torque. However, even maximum torque the UAV could generate was not enough to stabilize the system.

The system always appears to become instable in the horizontal plane, never in the vertical plane. This might be due to the fact that vertical disturbances are easier to reject using the thrust. The balance between thrust and gravity can

be seen as an actuator for rotations in the vertical plane, and is relatively easy to control. In the horizontal plane, such an actuator is not available. Therefore, the controller proposed should focus on stabilizing rotations in the horizontal plane.

3 System Specifications

In free flight, the attitude of the quadcopter is given by its roll, pitch and yaw angle. These angles describe its orientation around the x,y and z axis (see figure 2). In contact, however, it is useful to use a different set of axes to describe the orientation: x',y', and z' (see figure 3). The reasons for this will be explained later.

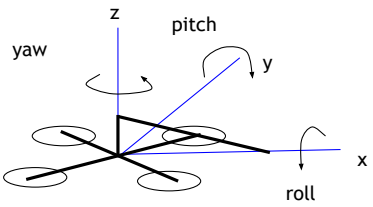

Figure 2: The axes of rotation in free flight

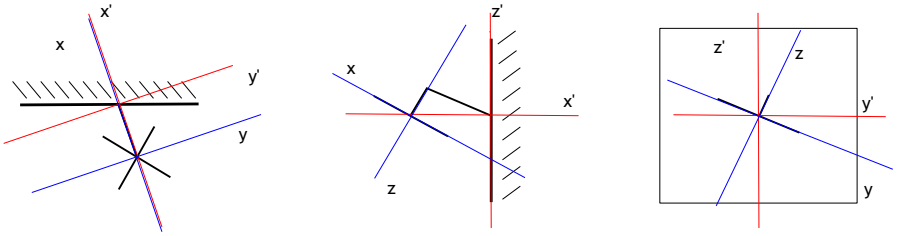

Figure 3: the axes of rotation in free flight (blue) and in contact mode (red)

Rotations are described as follows:

- around the x-axis: roll, or ϕ
- around the y-axis: pitch, or θ
- around the z-axis: yaw, or ψ
- around the x'-axis: contact roll, or Φ
- around the y'-axis: contact pitch, or Θ
- around the z'-axis: contact yaw, or Ψ
- derivatives of rotations are called rotation rates

Explanation of a few symbols that will be used throughout the report:

- m = mass
- l = length
- h = height
- COM = center of mass
- I = moment of inertia
- F = force

- R = rotation matrix
- τ = torque

The UAV used for this project is custom build. The attitude controller is implemented on an onboard Pixhawk controller, and the position controller is implemented on an external computer. The angle of the manipulator is actuated by a Dynamixal servo motor. The total thrust each motor can generate is 8.5 Newton using 8 inch propellers. Figure 4 shows a picture of the UAV with the manipulator mounted on top.

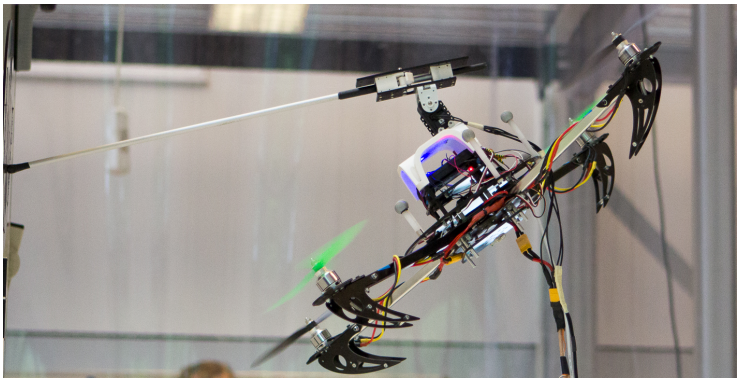

Figure 4: Picture of the UAV with manipulator

Measurements were done on the UAV in order to develop an accurate model of the dynamics:

- $m_{\text{uav (including manipulator)}} = 1.4\ kg$
- $m_{\text{uav arm}} = 0.2\ kg$
- $m_{\text{manipulator arm}} = 0.033\ kg$
- $l_{\text{uav arm}} = 0.335\ m$
- $l_{\text{manipulator arm}} = 0.5\ m$
- $h_{\text{manipulator arm}} = 0.18\ m$
- $r = l_{\text{COM to contact point}} = 0.5\ m$

Assuming that a quarter of the mass of an arm is located at the end of the arm, due to the motor and the leg, and the rest is distributed over the length of the arm, the moment of inertia of the UAV around its axes of rotation can be approximated (the mass of the manipulator arm is neglected in these calculations).

$$I_\phi = 4(\tfrac{1}{2}\tfrac{3}{4} m_{\text{arm}} (\tfrac{1}{2}\sqrt{2} \cdot l_{\text{arm}})^2 + \tfrac{1}{4} m_{\text{arm}} (\tfrac{1}{2}\sqrt{2} \cdot l_{\text{arm}})^2) = 0.028 \text{ kg m}^2$$
$$I_\theta = I_\phi = 0.028 \text{ kg m}^2$$
$$I_\psi = 4(\tfrac{1}{2}\tfrac{3}{4} m_{\text{arm}} l_{\text{arm}}^2 + \tfrac{1}{4} m_{\text{arm}} l_{\text{arm}}^2) = 0.056 \text{ kg m}^2$$

$$I_\Phi = I_\phi = 0.028 \text{ kg m}^2$$

$I_\Theta = I_\theta + m_{\text{UAV}} r^2 = 0.378 \text{ kg m}^2$
$I_\Psi = I_\psi + m_{\text{UAV}} r^2 = 0.406 \text{ kg m}^2$

4 Dynamics During Physical Contact with a Vertical Surface

In order to develop a simplified model of the dynamics of the UAV in contact with a wall, the following assumption is made: there is enough friction between the end-effector and the wall so that the end-effector does not move with respect to the wall when in contact. This assumption is based on the idea that the end-effector exerts a relatively high normal force (w.r.t. the parallel forces) on the wall, and that the friction coefficient between the two materials is sufficiently high. This assumption allows us to model the UAV with manipulator as a body of which the position of the end-effector is fixed.

Stable interaction is defined here as a situation in which the UAV is exerting a force against the wall, without rotations around the contact point and without slipping over the wall.

In order to achieve stable interaction, there should be no net torque around the (fixed) contact point, since that would cause a rotation around the contact point. This implies that the net force the UAV is exerting should always be in the direction of the end-effector (see figure 5). In addition to this requirement, it is desirable to keep the forces parallel to the wall low, otherwise the assumption that the friction force is high enough to prevent the UAV from slipping over the wall might not hold anymore.

Figure 5 illustrates the implications of these requirements. First, if the force the UAV is exerting a force that is not in line with the manipulator arm, this will result in a rotation around the contact point. Although this might be undesirable if the UAV is in perfect equilibrium, it can be used to reach an equilibrium or to counteract disturbances. Second, if the contact yaw and contact pitch angle are not equal to zero, the parallel forces that arise might cause the UAV to slip.

An important property of the system in contact mode is that torques around the center of mass (COM) of the UAV, created by differences in rotor thrusts, no longer result in rotations around the COM but in rotations around the contact point. The rotations around the contact point can be split into rotations in the vertical plane and rotations in the horizontal plane (as illustrated in figure 3). The reason to split these is that rotations in the vertical plane are relatively easy to control using the thrust, as opposed to rotations in the horizontal plane (as explained in section 2). This was apparent in the preliminary experiments, during which instability only occured in the horizontal plane, and never in the vertical plane. Figure 3 illustrates the axes around which the equations of motion are to be found (in red).

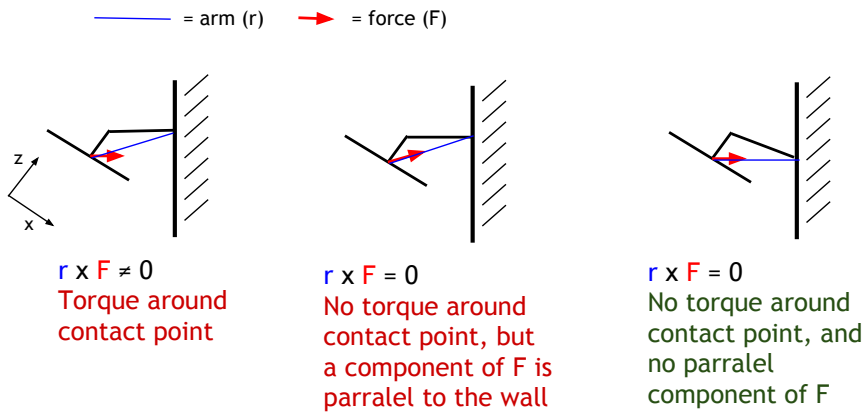

Figure 5: Possible configurations of the UAV when in contact with a wall

4.1 Exerted Force

The UAV is an underactuated system, and can only exert a linear force in the z-direction. This means that the UAV needs a nonzero pitch angle in order to exert a force in the x'-direction. Assuming that the UAV does not accelerate in z'-direction while in contact flight and does not exert significant parallel foces on the wall (i.e. the thrust exactly compensates for gravity), the pitch angle can be related to the force in x'-direction as follows (see figure 8 for an illustration):

$$F_x(\theta) = F_z \cdot \tan(\theta)$$

So to calculate what the pitch angle should be for a certain force setpoint, we can rewrite this equation to:

$$\theta(F_x) = \arctan(\tfrac{F_x}{F_z})$$

4.2 Rotations in the Horizontal Plane

In order to understand what causes rotations in the horizontal plane (the x'y' plane), the equation of motion is derived. Rotations in the x'y' plane correspond to rotations around the z' axis. The total torque around the z' axis consists of two terms: the effect of the torques in the body-fixed frame around the contact point, and the effect of the net force on the COM.

First, the projection of the torques in the body-fixed frame on the z' axis are determined. This is done using a rotation matrix. This rotation matrix only contains the roll and pitch angle, since the yaw angle is the same in both coordinate systems (both coordinate systems are shown in figure 3). The resulting torques are described with the subscript "t", as they are the result of

the torques in the COM.

$C_\theta = \cos\theta$
$S_\phi = \sin\phi$
etc.

$$R = RyRx \tag{1}$$

$$R = \begin{bmatrix} C_\theta & S_\theta S_\phi & S_\theta C_\phi \\ 0 & C_\phi & -S_\phi \\ -S_\theta & C_\theta S_\phi & C_\theta C_\phi \end{bmatrix} \tag{2}$$

$$\begin{bmatrix} \tau_{\Phi \mathrm{t}} \\ \tau_{\Theta \mathrm{t}} \\ \tau_{\Psi \mathrm{t}} \end{bmatrix} = R \begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} C_\theta \tau_\phi + S_\theta S_\phi \tau_\theta + S_\theta C_\phi \tau_\psi \\ C_\phi \tau_\theta - S_\phi \tau_\psi \\ -S_\theta \tau_\phi + C_\theta S_\phi \tau_\theta + C_\theta C_\phi \tau_\psi \end{bmatrix} \tag{3}$$

Then, the effect of the net force exerted on the COM on the torque around the contact point is determined. The same transformation is applied to the force in order to find its components in the contact coordinate system. These resulting torques are described with the subscript "f", as they are the result of the force in the COM. See figure 6 for an illustration of how $\tau_{\Theta \mathrm{f}}$ is determined, and figure 7 for an illustration of how $\tau_{\Psi \mathrm{f}}$

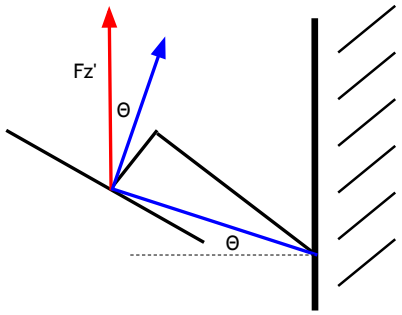

Figure 6: Illustration of the effect of a contact pitch angle on the torque around the y' axis

$$F = \begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix} \tag{4}$$

$$\begin{bmatrix} F_{x'} \\ F_{y'} \\ F_{z'} \end{bmatrix} = R \begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} = \begin{bmatrix} S_\theta C_\phi T \\ -S_\phi T \\ C_\theta C_\phi T - mg \end{bmatrix} \tag{5}$$

$$\begin{bmatrix} \tau_{\Phi \mathrm{f}} \\ \tau_{\Theta \mathrm{f}} \\ \tau_{\Psi \mathrm{f}} \end{bmatrix} = r \begin{bmatrix} 0 \\ F_{z'} C_\Theta \\ -F_{y'} \end{bmatrix} \tag{6}$$

The total torque around the z' axis can now be described as (see figure 7):

$$\tau_\Psi = \tau_{\Psi \mathrm{t}} + \tau_{\Psi \mathrm{f}} = I_\Psi \ddot{\Psi} = \tau_\Psi - F_{y'} r \tag{7}$$

$$I_\Psi \ddot{\Psi} = -S_\theta \tau_\phi + C_\theta S_\phi \tau_\theta + C_\theta C_\phi \tau_\psi + S_\phi T r \tag{8}$$

In order to design a controller for this system, it will be approximated by a linear system. This is done because analysis and control design are much easier for linear systems, and because a controller designed based on a linear system performs sufficiently well in many cases [10]. ϕ is assumed to stay small during contact flight, so $\sin\phi \approx \phi$ and $\cos\phi \approx 1$. However, in the term that contains τ_θ we will assume that $\sin\phi \approx 0$. Doing this will enable us to write the system in state space form, which is useful for designing the controller. Note that this is not an unreasonable approximation, since τ_θ stays small ($< 1\ Nm$) compared to Tr (has to be at least $mgr \approx 7\ Nm$ to compensate for gravity, as illustrated in figure 8).

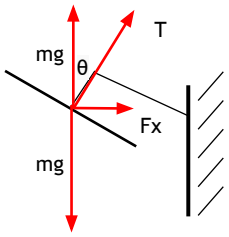

Figure 7: Illustration of the resulting torque (left)
Figure 8: Illustration of the relation between T, gravity, and F_x (right)

Unlike ϕ, θ will not stay small during contact, but vary with the applied force. This makes it harder to accurately approximate the non-linear terms containing θ. For now, θ will be approximated as an average value between the extremes it will most likely take on during contact flight. It is assumed that the contact force will not be lower than 2 Newton, and not higher than 15 Newton, which corresponds to two θ values (as calculated in eq. 11).

In this project, it is chosen to limit the contact force to 15 Newton, because this corresponds to a pitch angle of 0.829 radians. This angle approaches the maximum angle of 1.159 rad (see eqn 10) at which the UAV can still provide enough upward thrust to compensate for gravity ($F_{z'} = 0$). As each propeller can provide 8.5 Newton of thrust, the maximum pitch angle is computed as follows:

$$F_{z'} = C_\theta C_\phi T - mg = 0 \tag{9}$$

assuming $\phi \approx 0$

$$\theta_{max} = \arccos(\frac{mg}{4 \cdot 8.5}) \approx 1.159\ rad \tag{10}$$

It is not desirable to get too close to this maximum value, because that would mean the propellers cannot provide more thrust, and the system would become uncontrollable due to actuator constraints.

For the linearized sytem, θ is then approximated as follows (eq. 12). It remains to be validated whether this approximation is accurate enough, this will later be done using simulations and experiments.

$$\theta(F_{sp}) = \arctan(\frac{F_{sp}}{F_z}) \approx \arctan(\frac{F_{sp}}{m \cdot g}) \tag{11}$$

$\theta_{min}(F_{sp} = 2) \approx 0.145$
$\theta_{max}(F_{sp} = 15) \approx 0.829$

$$\theta \approx \frac{0.829 + 0.145}{2} \approx 0.487 \tag{12}$$

$\cos(\theta) \approx 0.884$
$\sin(\theta) \approx 0.468$

$$T = \frac{mg}{\cos(\theta)\cos(\phi)} \approx \frac{mg}{0.884} \approx 15.5 \tag{13}$$

When plugging these numbers into equation 8, the following equation of motion is obtained:

$$\ddot{\Psi} = -1.15\tau_\phi + 2.18\tau_\psi + 19.1\phi \tag{14}$$

Because the roll angle is a variable in this equation, and thus has an effect on the contact yaw angle, its equation of motion should also be taken into account when designing the controller. This equation is:

$$I_\phi \ddot{\phi} = \tau_\phi \tag{15}$$

$$\ddot{\phi} = 35.7\tau_\phi \tag{16}$$

And in state space form:

$$\begin{bmatrix} \dot{\Psi} \\ \ddot{\Psi} \\ \dot{\phi} \\ \ddot{\phi} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 19.1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \Psi \\ \dot{\Psi} \\ \phi \\ \dot{\phi} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ -1.15 & 2.18 \\ 0 & 0 \\ 35.7 & 0 \end{bmatrix} \begin{bmatrix} \tau_\phi \\ \tau_\psi \end{bmatrix} \tag{17}$$

4.3 Rotations in the Vertical Plane

The same approach can be used for rotations in the vertical plane (the x'z' plane). Rotations in the x'z' plane correspond to rotation around the y' axis. In this case, however, the torque around the y' axis is independent of the torque around the y axis. This is due to the extra degree of freedom in the manipulator arm: the angle of the manipulator arm is controlled by a servo motor in such a

way that the contact pitch angle remains zero (a more detailed explanation of how the manipulator arm is controlled can be found in section 5.3). Therefore this servo motor effectively prevents rotations around the y axis to have an effect on rotations around the y' axis. This means that only the resultant force on the COM has an effect on the torque around the y' axis. The equation of motion then becomes:

$$\tau_\Theta = \tau_{\Theta \mathrm{f}} = I_\Theta \ddot{\Theta} = r C_\Theta (C_\phi C_\theta T - mg) \tag{18}$$

5 Controller Architecture

The controller that is the result of this project consists of three seperate parts. The innovative part of this controller is the part that stabilizes rotations around the z' axis, which, as explained before, was the biggest problem in contact flight. This part will from now on be called the 'Contact Yaw Controller'. Second, there is a part that stabilizes rotations around the y' axis, which can be accomplished by a more conventional controller. This part will be called the 'Contact Pitch Controller'. Third, there is a part that regulates the force that is exerted perpendicular to the wall using feedback on the pitch angle. This part will be called the 'Pitch Controller'.

In figure 9, the full controller structure is shown. Every controller has a setpoint, which it is 'trying' to reach (..._sp). Every controller also has a control gain matrix (K_..) which determines how it responds to deviations from the setpoint. And every controller has a matrix that scales the setpoints in order to ensure zero steady-state error (N_..). All states of the system (Ψ,$\dot{\Psi}$,ϕ,$\dot{\phi}$,θ,$\dot{\theta}$,z,$\dot{z}$) are measurable using either IMUs or a camera system like OptiTrack. All rotations and rotation rates are measured in body-fixed frame so no additional transformation is required.

5.1 Contact Yaw Controller

Because all states are measured, full state feedback can directly be applied to the system, without having to design estimators [10]. In full state feedback, the inputs of the system are computed as a linear combination of the states of the system [10]. Since this system has two inputs, τ_ϕ and τ_ψ, the control law is given by the following equation, in which K_Ψ is a 2x4 matrix, and N_Ψ a 2x2 matrix.

$$\begin{bmatrix} \tau_\phi \\ \tau_\psi \end{bmatrix} = N_\Psi \begin{bmatrix} \Psi_{sp} \\ \Psi_{sp} \end{bmatrix} - K_\Psi \begin{bmatrix} \Psi \\ \dot{\Psi} \\ \phi \\ \dot{\phi} \end{bmatrix} \tag{19}$$

A common way to solve a control problem like this is using a linear quadratic regulator (LQR) [11]. Given a multi-input linear system (eq. 17), and a quadratic cost function dependent on the states of the system and the inputs to

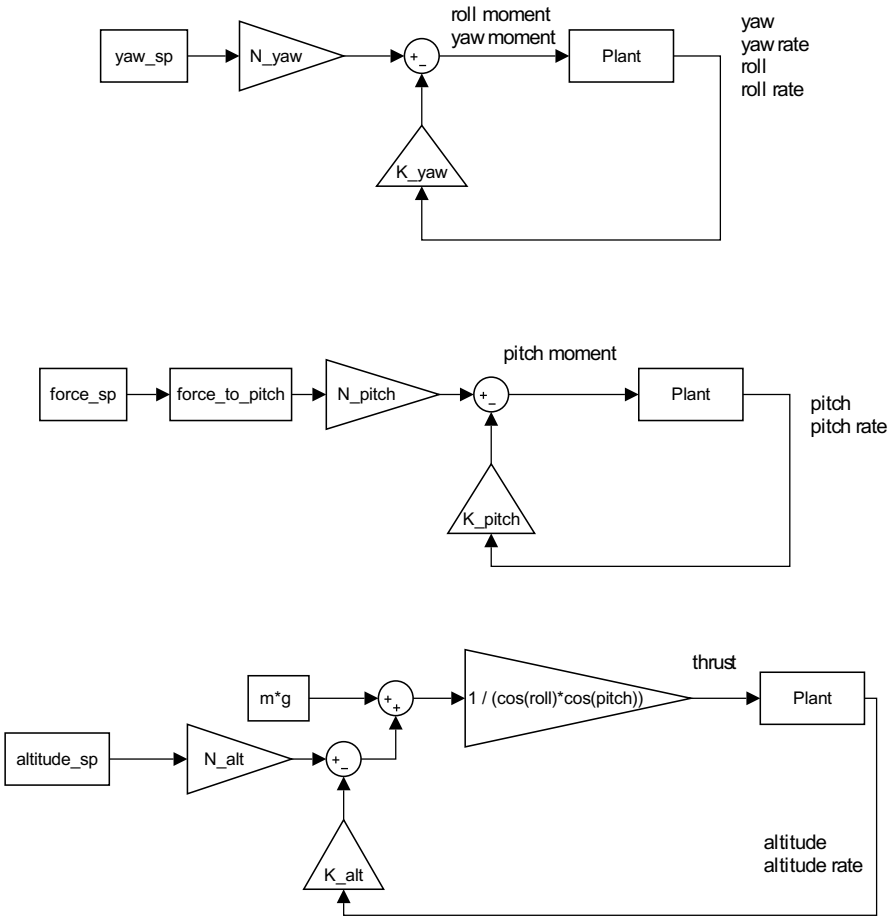

Figure 9: Structure of the complete controller for contact flight

the system, the LQR computes a control gain matrix K_Ψ that minimizes this cost function [12].

The lqr function MATLAB is used here to compute the optimal control gains [13]. MATLAB's lqr is a function of 4 matrices. The A and B matrices represent the state space, and are based on eq. 17. The Q and R matrices represent the parameters of the quadratic cost function. Q represents the weight placed on each state of the system, and R represents the cost placed on each output of the controller (in this case τ_ϕ and τ_ψ). The matrices used for this controller design are shown here:

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 19.1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{20}$$

$$B = \begin{bmatrix} 0 & 0 \\ -1.15 & 2.18 \\ 0 & 0 \\ 35.7 & 0 \end{bmatrix} \tag{21}$$

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{22}$$

$$R = \begin{bmatrix} 1 & 0 \\ 0 & 5 \end{bmatrix} \tag{23}$$

The Q-matrix is chosen this way because it is considered equally important to track references on the yaw angle and the roll angle. Although the main goal of the controller is to track a yaw reference, from the equations of motion it shows that a roll angle has a relatively large effect on the torque around the yaw axis. Therefore tracking a roll angle setpoint is considered as important as tracking a yaw angle setpoint. The cost matrix R is chosen like this, with a higher cost on the yaw input, because a yaw torque costs a lot more energy than a roll torque due to quadcopter dynamics. Moreover, the maximum yaw torque is much lower than the maximum roll torque for the same difference in rotor speed.

Using these matrices as inputs in MATLAB's lqr function gives the following K-matrix:

$$K_\Psi = \begin{bmatrix} K1 & K2 & K3 & K4 \\ K5 & K6 & K7 & K8 \end{bmatrix} = \begin{bmatrix} 0.9888 & 0.5503 & 2.9306 & 0.4204 \\ 0.0667 & 0.0275 & 0.0870 & 0.0076 \end{bmatrix} \tag{24}$$

This will result in a controller that evaluates both the contact yaw and roll angles, as well as their derivatives in order to compute a torque around the yaw and roll axis. And because the control of the contact yaw and roll are combined in this controller, the result is that a roll angle is used to compensate for an error in the contact yaw angle. It can be seen from the equations of motion that this is an efficient way of controlling the contact yaw angle, since a roll angle has a large effect on the contact yaw angle.

The reference signal needs to be premultiplied with the following N_Ψ matrix (due to the structure of the controller), in order to ensure zero steady-state error (this becomes clear when the structure of the controller is rewritten, as in figures 14 and 15).

$$N_\Psi = \begin{bmatrix} 0.9888 & 0 \\ 0 & 0.0667 \end{bmatrix} \tag{25}$$

5.2 Contact Pitch Controller

As mentioned before, the contact pitch angle can be controlled by the balance between thrust and gravity. The resulting force in the vertical direction will cause a torque around the y' axis. Rather than controlling the contact pitch angle directly, it is chosen to control the angle by controlling the attitude.

This has the same result as long as the contact pitch angle remains small. Because the contact point is fixed, a change in altitude will correspond to a change in contact pitch. Although this relation is not linear, for small contact pitch angles it can be approximated to be linear. Furthermore, altitude and contact pitch are both actuated by a force in z-direction. The advantage of controlling altitude instead of contact pitch is that altitude is directly measured by the Optitrack system. Measuring the contact pitch, however, would require sensing the position of the end-effector and performing additional calculations.

The setpoint for the altitude is kept equal to what is was in free flight. This means that the altitude, and therefore the contact pitch angle, is kept constant during contact flight. A PD-controller is used to control altitude, in combination with a feedforward controller. The feedforward controller compensates for gravity (taking into account the orientation of the UAV), and the PD-control ensures the setpoint is reached and compensates for disturbances. This controller was tuned during test flight and the following gains were determined:

$$K_\Theta = \begin{bmatrix} 0.12 \\ 0.4 \end{bmatrix} \tag{26}$$

It should be made clear that the thrust signal in this controller is not expressed in Newton, but is a number between 0 and 1. This is an unfortunate consequence of using the onboard Pixhawk controller to control the motor signals. These gains are tuned specifically for this controller, and therefore probably will not perform well when used in a simulation in which the thrust signal is expressed in Newton. A future improvement of this system would be to omit the Pixhawk controller, and perform the transformation between thrust signal and motor signal in the Contact Pitch Controller presented here. However, due to time limitations this is not considered a priority for this project.

The contact pitch angle is also controlled in another way. The manipulator arm has one degree of freedom. Using a servo motor its orientation the xz-plane can be changed. This degree of freedom is used to ensure the end-effector is always exactly at the same height as the COM i.e. the contact pitch is angle is zero. However, this servo motor is not strong enough to achieve this during contact flight and therefore the altitude controller described before is also required to control the contact pitch angle.

5.3 Pitch Controller

As mentioned before, to exert a force the UAV needs to have a pitch angle. A PD-controller is used in this case to control the pitch, which is the same controller that is used in free flight. This controller was tuned during free

flight, and the same gains were used in contact flight. The following gains were determined:

$$K_\theta = \begin{bmatrix} 12.9 \\ 2.0 \end{bmatrix} \tag{27}$$

6 Simulation of the System

6.1 Simulation of the Dynamics

A simulation of the system dynamics described in the previous section was made. The goal of the simulation is to serve as a first test for whether the controller stabilizes the system or not before testing it in the lab, thereby saving a lot of time and spare parts.

At first I developed a very elaborate simulation, also taking into account the dynamics of free flight, the dynamics of collisions with the wall, the gyroscopic effects of the propellers, the numerical problems involved with collisions simulated in discrete time etc.. Later I realized that it might be more valuable to limit the scope of the simulation to one specfic situation, in this case the contact with the wall. The reason for this is that the simulation became so complex it was hard to find the source of a problem when something unexpected happened. Also, many of these simulated effects did not appear to have an effect on the problem that is investigated, i.e. what causes the instability around the z' axis. However, the time spent working on the elaborate simulation was not lost time, because I gained a lot of insight into the dynamics of the system and it enabled me to develop the solution presented in this report.

The simulation of contact flight includes the dynamics described in the previous section, taking into account the contact yaw, contact pitch, roll, and pitch angle. The simulation uses the non-linear equations. The Euler method is used to integrate the differential equations, with a timestep of 1 millisecond. Several different methods and timesteps were tried but the difference in outcome was negligible, so in the end the simplest method was chosen, i.e. the Euler method. The outputs of the controller are fed into the system at 50 Hz, which is a reasonable (minimum) rate for controllers for UAVs of this size. Lastly, a disturbance is added to $\ddot{\Psi}$, $\ddot{\Theta}$, $\ddot{\phi}$, and $\ddot{\theta}$ in the form of white gaussian noise with a power of 20 dBW, corresponding to angular accelerations of up to $40\frac{rad}{s^2}$. The power of the disturbance was chosen so that the system becomes instable when a position controller is used similar to the one used in the preliminary experiments.

6.2 Simulation of the Controller

The controller described in section 5 was tested in the simulation that was described in the previous section. Figure 10 shows the reponse of the system on a step change on the contact yaw setpoint. Here it can be seen that the controller 'uses' a roll angle to change the contact yaw angle. The system reaches its setpoint after about two seconds, and then settles into its new equilibrium. Also,

the contact pitch angle stays close to zero (as its setpoint is always zero). The pitch setpoint is kept at its determined average value of 0.487 in this simulation.

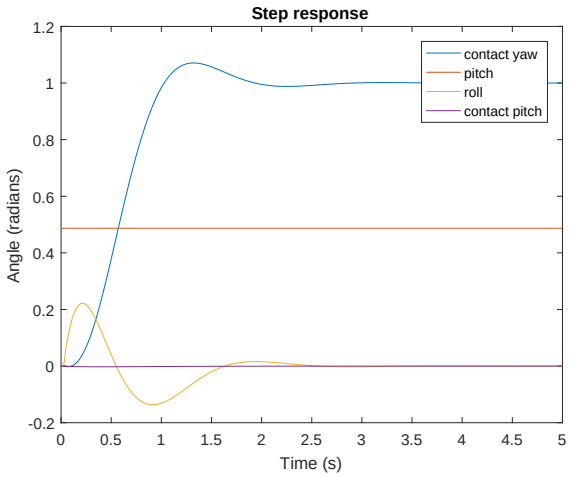

Figure 10: Step response

In figure 11 the result of the disturbances in the simulation can be seen. It can be seen that the yaw controller stabilizes the contact yaw and roll angles around zero, for different pitch value. Also, the pitch is varied between its minimum (0.145 rad) and maximum (0.826 rad) expected value, to validate that the controller stabilizes the system for all these pitch values.

In figure 12 it can be seen that the altitude controller stabilizes the contact pitch around zero. Note that different values for the control gains were used in the simulation than what was stated in section 5.2. The reason for this is also explained there.

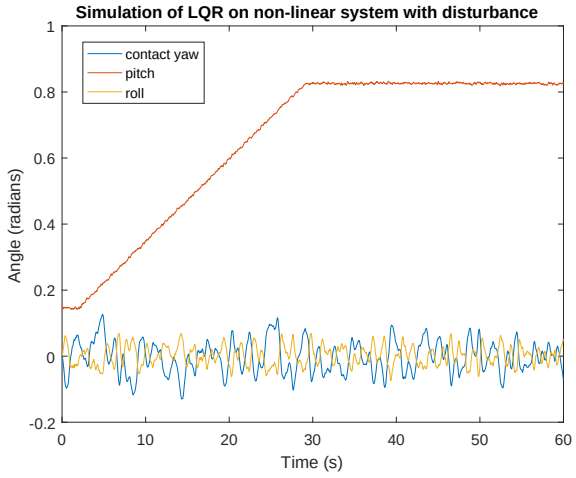

Figure 11: Effect of the controller on the roll and contact yaw

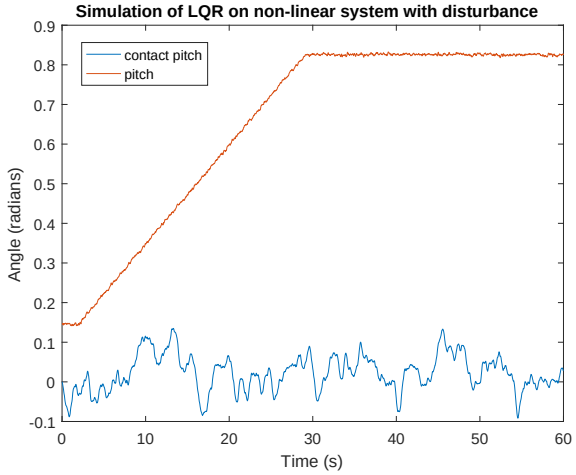

Figure 12: Effect of the controller on the contact pitch

7 Experiments

7.1 Rewriting the Controller

As described previously, the outputs of the contact yaw controller are torques (in the body-fixed frame). However, the onboard Pixhawk controller used in the experiments cannot take torques as inputs, only attitude setpoints, attitude rate setpoints, or raw motor values. In this case, it is chosen to use the attitude rate setpoints, because it gives more freedom than using attitude setpoints, and

it takes care of mapping forces to motor signals, alleviating this need from the proposed contact controller.

The rate controller on the Pixhawk is a simple proportional controller and its schematic looks like this:

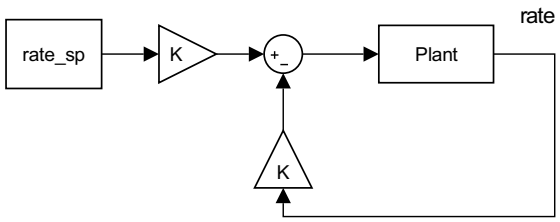

Figure 13: Proportional rate controller on the Pixhawk

To adapt the Contact Yaw Controller to output rate setpoints instead of torques, the structure of the controller can be viewed as shown in figure 14. Note that only half of the yaw controller is shown here, the part that outputs the roll torque. In the way it is drawn now, the right part of this controller corresponds to a proportional rate controller, and the on-board rate controller of the Pixhawk can fullfill this part.

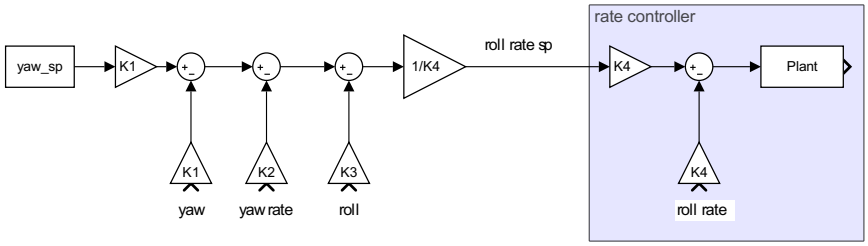

Figure 14: Illustration of the seperation of the Contact Yaw Controller (the part that outputs τ_ϕ) and the roll rate controller

To adapt the Contact Yaw Controller to match this model, the following changes need to be made. The control of the roll rate (corresponding to the control gain K4) is now performed in the rate controller on the Pixhawk. Therefore it should be removed from the proposed Contact Yaw Controller. Also, the output of the controller should then be divided by K4, to make the signal a rate setpoint. The same can be done for the other half of the controller, the part that outputs the yaw torque (see figure 15). In this case the gains K5..K8 are involved, and K6 should be removed from the Contact Yaw Controller since that is the gain corresponding to feedback on the yaw rate. In short, the rate

setpoints are determined as follows:

$$\begin{bmatrix} \dot{\phi}_{sp} \\ \dot{\psi}_{sp} \end{bmatrix} = \begin{bmatrix} \frac{1}{K4} & 0 \\ 0 & \frac{1}{K6} \end{bmatrix} \left(N\Psi \begin{bmatrix} \Psi_{sp} \\ \Psi_{sp} \end{bmatrix} - \begin{bmatrix} K1 & K2 & K3 & 0 \\ K5 & 0 & K7 & K8 \end{bmatrix} \begin{bmatrix} \Psi \\ \dot{\Psi} \\ \phi \\ \dot{\phi} \end{bmatrix} \right) \tag{28}$$

The same is done for the Pitch Controller, as it also controls a rotation. It is, however, not necessary for the Contact Pitch Controller, because this controller outputs a thrust signal, which can directly be used by the Pixhawk controller.

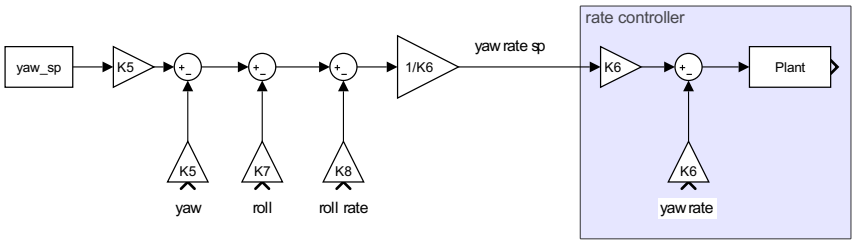

Figure 15: Illustration of the seperation of the Contact Yaw Controller (the part that outputs τ_ψ) and the yaw rate controller

7.2 Testing the Controller

To verify the controller, a test environment was constructed. An industrial 6-DOF force-torque sensor was mounted on a vertically placed table, creating an area for the UAV to push against. The force sensor measures the force in three direction, one perpendicular to the table (x-direction), and two parallel to the table (see figure 16 for a picture of the set-up). A regular position controller was used to approach this area, and once the UAV came very close we manually switched to the contact controller. The force setpoint was gradually increased from 2 Newton to 15 Newton, and the UAV achieved stable interaction, lasting a few minutes. This experiment was repeated several times, with similar results. In figure 17 the orientation of the UAV during contact is plotted, and in figure 18 the measured force is plotted.

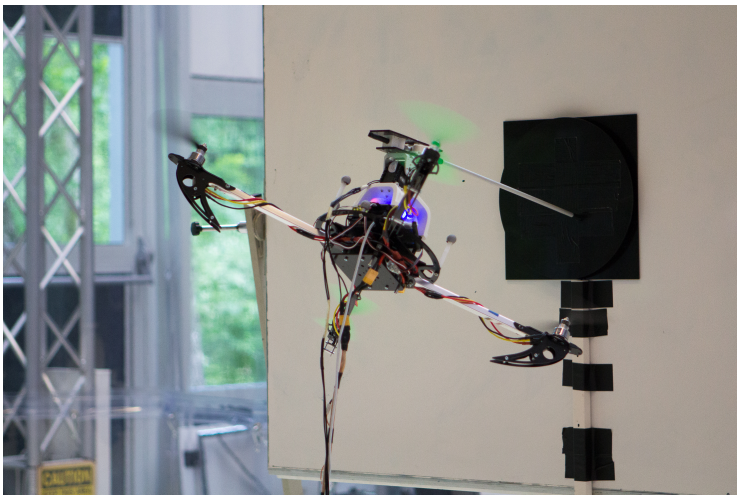

Figure 16: Picture of the setup of the experiment

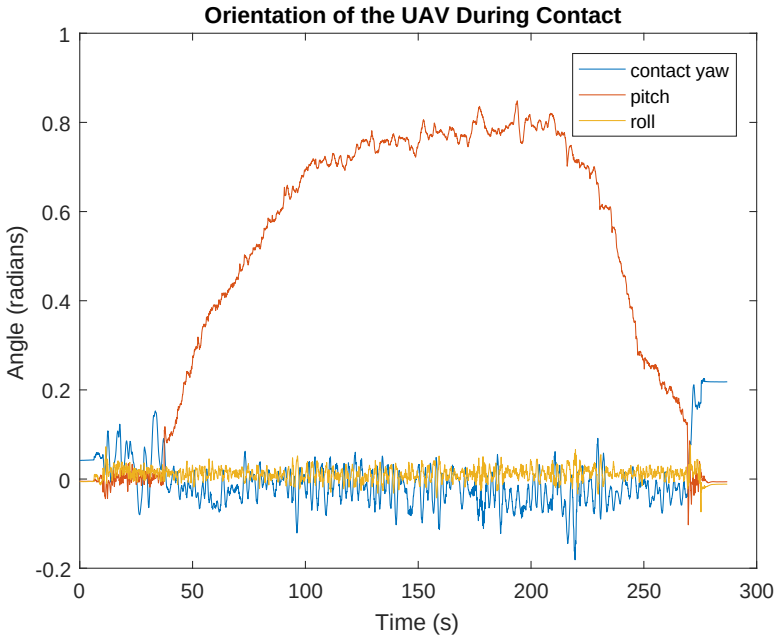

Figure 17: Orientation of the UAV during contact

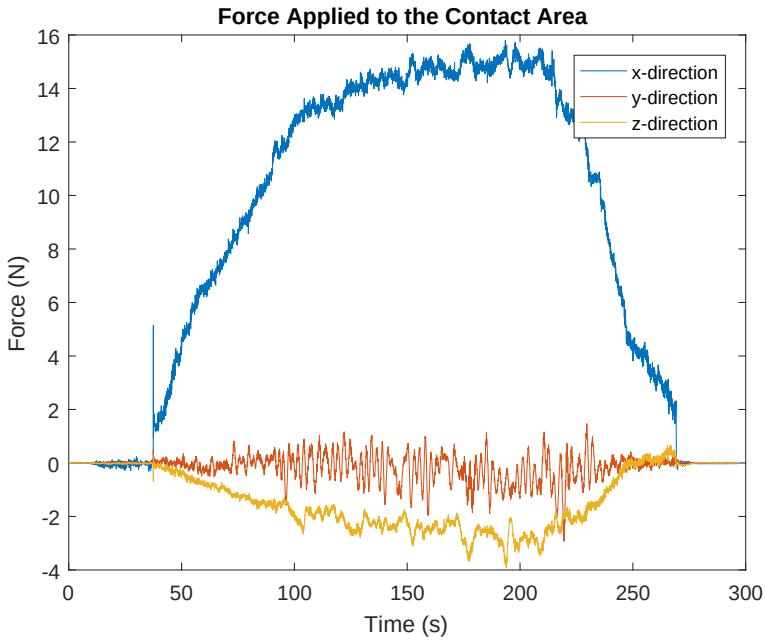

Figure 18: Force applied to the contact area

8 Discussion of the Results

From the experiments it is clear that the controller stabilizes the system and that a force is exerted perpendicular to the contact area. It can be seen in the figures that the force in x-direction is indeed varied between 2 and 15 Newton. This indicates that the Pitch Controller succesfully tracks a force setpoint.

The force in y-direction shows why it was necessary to make the assumption of high friction in order to design this controller. The UAV uses a roll angle to compensate for a contact yaw error, thereby exerting a force parallel to the wall. But because the friction is sufficiently high, the end-effector does not slide over the wall, but stays where it is as the COM turns around the contact point.

The small negative force in the z-direction indicates that the contact pitch angle was not exactly zero. In other words, the end-effector was not perfectly aligned with the COM. This was probably due to the algorithm that computed the angle for the servo motor that controlled the manipulator. An exisiting algorithm made by a previous user of this UAV was used for this in order to save time, but it could be improved in the future. In this case it was not a problem, because the friction with the wall was sufficiently high so the end-effector did not slip over the wall.

The results show that, using this controller, it is possible to exert a high force against a vertical surface with a quadcopter. For reasons mentioned earlier, the exerted force in this experiment was limited to 15 Newton. The limit of this controller was not reached in the experiment, which was intended. However,

that does mean that this controller might be able to stabilize the system under higher contact forces as well. No conclusions about this can be drawn yet, as experiments with higher are not yet performed. It is expected, however, that this algorithm would perform equally well on more powerful quadrotors (since the controller design is based on physical principles that hold for all quadrotors). If this is the case, the contact force can be further increased without approaching the maximum thrust that can be generated.

Lastly, the controller was now tested on a UAV which was still partially controlled by the onboard Pixhawk controller. Although the controller structure was designed to limit the influence of the Pixhawk controller as much as possible, it is not an optimal solution. In the future, this controller should be implemented in a way such that it is independent of any other controllers in order to optimize the performance.

9 Conclusion

In this report, a controller is presented for applying a significant force to a vertical surface using a quadcopter. Stable interaction is achieved using this controller with a contact force of 15 Newton, over a time period of several minutes. This contact force is significantly higher than what was reached in previous research. In previous research the highest contact force that could be held for longer than a few seconds, was around 3 Newton [3, 4, 7, 8, 9]. This control algorithm is a step forward in the development of UAVs that are able to perform tasks that require physical interaction with their environment, and future steps are suggested in this report.

10 Future Work

10.1 Suggestions for future research

The controller presented in this report is a step towards UAVs that are capable of autonomously inspecting and cleaning surfaces. A next step in this development could be to adapt this controller to enable a UAV to move over a wall, while exerting a force perpendicular to it. This ability, combined with a type of brush on the end-effector, would enable a UAV to clean a surface.

At the moment this is not possible with this controller, as it uses the friction with the wall to control its orientation, and would therefore not work if the friction force is lower than the force it is exerting parallel to the wall.

A possible approach to solving this problem could be found in the design of the manipulator. Specifically, if the friction coefficient between the end-effector and the wall could be made variable, it would make controlling movement parallel to the wall much simpler. In that case, the friction coefficient could be set high when the orientation of the UAV needs to be changed, and low when the position of the UAV on the wall needs to be changed. Effectively, this would mean placing a type of 'brake' on the end-effector.

It is also suggested that this controller is tested on different types of multi-rotors, to confirm its generalizibility. When testing this controller on a different system, however, it should be kept in mind that the control gains might need to be adapted to the parameters of the system at hand. The methods presented in this report can be used to realize this. Furthermore, it should be tested whether the maximum contact force is only limited by actuator constraints, or whether unexpected destabilizing effects occur when the contact force is increased.

10.2 Including the effect of friction in the simulation

The simulation described in section 6 can be adapted to take the friction between the end-effector and the wall into account. This will be useful in developing and verifying a controller that will enable the UAV to move over the wall.

To achieve this, the COM and the end-effector are modelled as two seperate points, connected to eachother by a spring (see figure 19). If the stiffness of the spring is set to be sufficiently high, this spring will resemble the rigid stick that the manipulator arm actually is. The advantage of modelling the manipulator as a spring is that the forces acting on the COM and the end-effector can be considered seperately. That means that the contact point does no longer need to be at a fixed position.

The simulation is for now made two-dimensional, since movement in the horizontal plane is predicted to be harder to control, for the same reasons as why rotations in the horizontal plane are harder to control. However, the simulation can be easily extended to three dimensions if this is deemed necessary. Two new axes are defined for this simulation, the world-frame X and Y axes (as illustrated in figure 19). The body frame axes, and the contact frame axes are still the same as described in section 3.

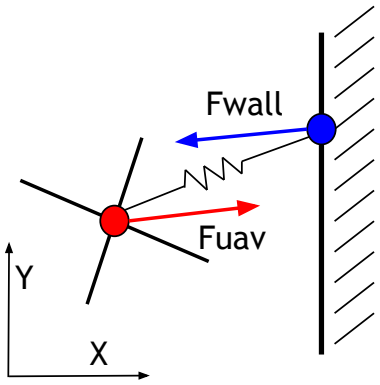

Figure 19: The axes of rotation in free flight

The force the UAV is exerting, F_{UAV}, is computed by rotationg the thrust to the worldframe:

$$F_{\text{UAV}} = RzRyRx \begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix} \tag{29}$$

The acceleration of the COM and the end-effector is computed using the following equations:

$$m_{\text{com}} \begin{bmatrix} \ddot{X}_{\text{com}} \\ \ddot{Y}_{\text{com}} \end{bmatrix} = F_{\text{UAV}} - F_{\text{spring}} \tag{30}$$

$$m_{\text{end effector}} \begin{bmatrix} \ddot{X}_{\text{end effector}} \\ \ddot{Y}_{\text{end effector}} \end{bmatrix} = F_{\text{wall}} + F_{\text{spring}} \tag{31}$$

Where F_{UAV} and F_{wall} are two-dimensional vectors containing the force components in X and Y direction. F_{spring} is the vector of the spring force, and is computed as follows:

$$d = \begin{bmatrix} X_{\text{end effector}} - X_{\text{com}} \\ Y_{\text{end effector}} - Y_{\text{com}} \end{bmatrix} \tag{32}$$

$$l = \sqrt{d[1]^2 + d[2]^2} \tag{33}$$

$$f_{\text{spring}} = k(l - r) \tag{34}$$

$$F_{\text{spring}} = f_{\text{spring}} \frac{1}{l} d \tag{35}$$

In equation 34, f_{spring} is the magnitude of the spring force, k the spring constant, and r the normal length of the manipulator arm i.e. the length of the spring at rest. In equation 35, the force vector is calculated by multiplying the force magnitude by the normalized d vector (representing the direction from the COM to the manipulator).

The contact yaw angle is then computed using the relative position of the end effector and the COM:

$$\Psi = \arctan \frac{d[1]}{d[2]} \tag{36}$$

Since the end effector cannot move through the wall, the reaction force of the wall in X direction, $F_{\text{wall}X}$, is equal in magnitude to the spring force exerted on the wall in X direction, $F_{\text{spring}X}$. The reaction force of the wall in Y direction, however, is computed as a friction force, with μ being the friction coefficient, and m a factor correcting the direction of the force:

$$F_{\text{wall}Y} = m\mu F_N = m\mu F_{\text{spring}X} \tag{37}$$

$$m = \begin{cases} 1 & \text{if } \Psi < 0 \\ -1 & \text{if } \Psi \geq 0 \end{cases}$$

In the simulation, a distinction is made between the static friction coefficient μ_s, and the kinetic friction coefficient μ_k.

The total equation for computing the reaction force of the wall is given by:

$$F_{\text{wall}} = \begin{bmatrix} -F_{\text{spring}X} \\ m\mu F_{\text{spring}X} \end{bmatrix} \tag{38}$$

To complete these equations of motion, the torques around the COM and their effect on rotations around the contact point should also be taken into account. However, because of time restrictions, this could not be done in the current project. Once this is done, the new equation of motion for Ψ can replace the old one in the simulation, and the simulation can be used to develop and test a controller that enables a UAV to move over a wall while exerting a force.

References

[1] P. Doherty, P. Rudol, *A UAV Search and Rescue Scenario with Human Body Detection and Geolocalization*, AI 2007: Advances in Artificial Intelligence, pp. 1-13, DOI: 10.1007/978-3-540-76928-6_1, 2007.

[2] Z. Li, Y. Liu, R. Hayward, J. Zhang and J. Cai, *Knowledge-based power line detection for UAV surveillance and inspection systems*, Image and Vision Computing New Zealand, IVCNZ 2008 23rd International Conference Christchurch, pp. 1-6, DOI: 10.1109/IVCNZ.2008.4762118, 2008.

[3] M. Fumagalli, R. Naldi, A. Machelli, F. Forte, A.Q.L. Keemink, S. Stramigioli, R. Carloni, L. Marconi, *Developing an Aerial Manipulator, physical interaction with the environment*, IEEE Robotics & Automation Magazine, pp. 41-50, DOI: 10.1007/978-3-540-76928-6_1, 2014.

[4] A. Albers, S. Trautmann, T. Howard, T.A. Nguyen, M. Frietsch, C. Sauter, *Semi-autonomous Flying Robot for Physical Interaction with Environment*, IEEE Conference on Robotics, Automation and Mechatronics, pp. 441-446, DOI: 10.1007/978-3-540-76928-6_1, 2010.

[5] V. Ghadiok, J. Goldin and W. Ren, *Autonomous indoor aerial gripping using a quadrotor*, 2011 IEEE International Conference on Intelligent Robots and Systems (IROS), pp. 4645-4651, DOI: 10.1109/IROS.2011.6094690, 2011.

[6] Q. Lindsey, D. Mellinger, V. Kumar, *Construction of cubic structures with quadrotor teams*, Proc. Robotics: Science & Systems VII, 2011.

[7] M. Fumagalli, R. Naldi, A. Macchelli, R. Carloni, S. Stramigioli, L. Marconi, *Modeling and control of a flying robot for contact inspection*, 2012 IEEE International Conference on Intelligent Robots and Systems (IROS), pp. 3532-3537, 2012.

[8] J.L. Scholten, M. Fumagalli, S. Stramigioli, R. Carloni, *Interaction control of an UAV endowed with a manipulator*, 2013 IEEE International Conference on Robotics and Automation (ICRA), pp. 4910-4915, 2013.

[9] T. Bartelds, A. Capra, S. Hamaza, S. Stramigioli and M. Fumagalli, *Compliant Aerial Manipulators: Toward a New Generation of Aerial Robotic Workers*, IEEE Robotics and Automation Letters, pp. 477-483, DOI: 10.1109/LRA.2016.2519948, 2016.

[10] G. Franklin, J.D. Powell, A. Emami-Naeini, *Feedback Control of Dynamic System*, Addison-Wesley Publishing Company, Third Edition, 1993.

[11] Control Tutorials for Matlab and Simulink, University of Michigan, taken from http://ctms.engin.umich.edu/CTMS/index.php?example= AircraftPitch&section=ControlStateSpace#5 on 30-06-2016.

[12] E.D. Sontag, *Mathematical Control Theory: Deterministic Finite Dimensional Systems*, Chapter 8.2: Linear System with Quadratic Cost, Second Edition, Springer, New York, 1998.

[13] MATLAB lqr function description, taken from http://nl.mathworks.com/help/control/ref/lqr.html on 30-06-2016