The Armed Quadcopter

Ashwin Varghese Kuruttukulam[1]

***Abstract*— The primary aim of this report is to model and simulate an autonomous quadcopter with a 3 degree of freedom arm. We first discuss the standard quadcopter dynamics model and coordinates conventions. Following this we discuss the design considerations, forward and inverse kinematics of the 3 degree of freedom arm placed on the quadcopter. Following this we discuss how the position controller design of the quadcopter.**

Finally, we validate the inverse and forward kinematics by simulating the *armed quadcopter* using the VREP simulation environment[2]. The simulation code is available at: https://github.com/ashwinvk94/armed_quadcopter

I. INTRODUCTION

Quadcopters have shown to be useful for a variety of tasks ranging from landscape mapping to search and rescue. Amazon showcased its delivery drone system recently and the DJI matrice drone was used to perform thermal inspections on windmills.

In comparison with other modes of transportation, flying vehicles have a distinct advantage of being able to control its own position in 3 dimensional space rather than be limited to a plane such as the ground or sea. Additionally, holonomic systems, such as quadcopters and multicopters have the capability to quickly change their direction of motion, allowing very fast and agile flight.

Quad copters on their own do not usually meaningfully interact with their environment and are currently mostly used for applications such as surveying and photography. Attaching an arm attached to a quadcopter allows it to meaningfully interact with its environment.

Fig. 1. Amazon Air, DJI Matrice

II. HISTORY OF QUADCOPTERS

The earliest recorded use of an unmanned aerial vehicle for occurred on July 1849, when the Austrian imperial forces besieging Venice attempted to float some 200 paper hot air balloons each carrying a 24-30 pound armament. Aerial drones have been thought of even before, as seen from the numerous drawings of aerial vehicles Leonardo Da Vinci Quadcopters [3].

[1]A. V. Kuruttukulam is with Faculty of Systems Engineering, University of Maryland, MD 20742, USA ashwinvk94 at gmail.com

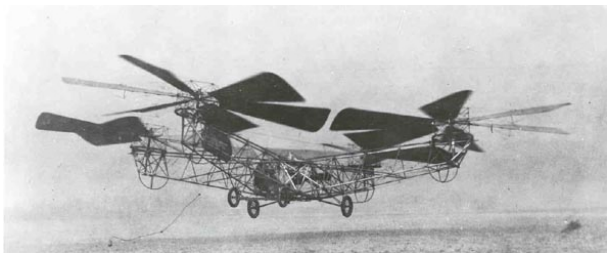

Fig. 2. De Bothezat Helicopter, 1923

Quadcopters are much older than most people expect. Quadcopters were among the first vertical take-off and landing vehicles (VTOLs). Engineers developed quadcopters to solve the problems that helicopter pilots had with making vertical flights. One of the first was the de Bothezat helicopter [4] and it was an experimental quadrotor helicopter built for the United States Army Air Service by George de Bothezat in the early 1920s, and was said at the time to be the first successful helicopter [5]. However,they did not become very popular due to many reasons,among which were mechanical complexity, relatively large size and weight(since they were driven by engines), and difficulties in control.

III. DESIGN

A. Quadcopter Design

In the multicopter family of designs, the quadcopter is the most simple design and is the most popular layout for a whole lot of reasons. Quadcopters are symmetrical and embody the simplest principle of operation for controlling roll, pitch, yaw and motion. Simply varying the speed of each of the motors provides full 3D motion and rotation and hover capability.

Although the quadcopter design lacks the efficiency and stability of other VTOL systems such as helicopter, at a small scale the high power-density electric motors, power semiconductors and the low manufacturing cost makes them very popular.

B. Arm Design

In the hover state, the quadcopter has the ability to reach any position in 3D space and attain any yaw angle orientation. But as per to its design, non-zero roll and pitch angles will cause the quadcopter to move from its current position. Consecutively, the roll and pitch angles are constrained to be zero at hover state.

If we attach a static gripper to the drone, the quadcopter would be able to move this gripper to any position and yaw orientation, but not all orientations. 2 revolute joints which

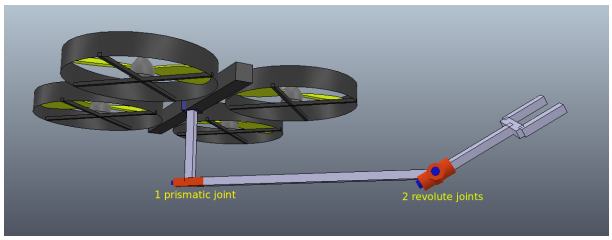

Fig. 3. Arm Design

will enable the rotation of the gripper in the directions of the roll and pitch axes allows the quadcopter to move the gripper to any position and orientation. Now the quadcopter can move the gripper to all possible positions and orientations.

Additionally, in order to ensure that the gripper does not interfere with the quadcopter body or propellers during the gripping action, a prismatic joint facing the front of the quad, offset vertically down by a distance h_1 was added. Using this, the gripper can be extended when needed and retracted when not in use. This will allow for more agile flight(when retracted) since the torque generated by the grippers weight when it is retracted would be lesser than when it is extended.

C. Gripper Design

As we have mentioned before, a gripper is attached to the end-effector of the arm. Since there was a number of available gripper designs in the VREP simulator, we chose one as per the following design criteria. For simplicity, we assume that we will be gripping cylindrical and cuboid shaped objects.

One of the main design criteria for the gripper, is that ideally, it should not interfere with the object when it is in the open state. Since we will always be approaching the objects from the front side, this can achieved by choosing a gripper whose fingers is behind the X-Y plane at the grasping point, as shown in the fig 4, when it is in the open state.

In addition to this, the gripper should be able to grasp object of various shapes and sizes. Also, the gripper should be of minimum weight in order to increase the flight time of the quadcopter.

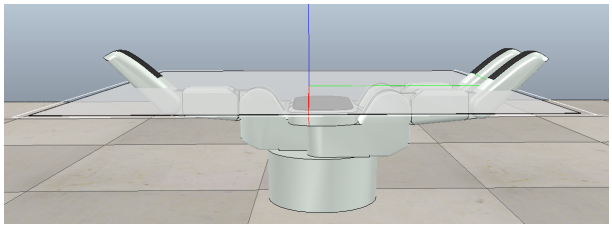

Fig. 4. Gripper XY plane

The BarrettHand [6] has three articulated fingers and a palm as illustrated in fig 5 which act in concert to trap the target object firmly and securely within a grasp consisting of seven coordinated contact vectors one from the palm plate and one from each link of each finger. The gripper also contains a torqueswitch mechanism which shifts the torque to the required finger joint. In addition, two of the three fingers in the BarettHand, have a variable finger angle, allowing for even further grasping configurations. This report does not go in how to set the finger angle, but this could be done by approximating the object shape using a computer vision algorithm. This allows it to grasp a variety of objects. For real-world applications, the Model T from the Yale open Hand Project [14], would be a cheap, light and reliable alternative to the BarrettHand.

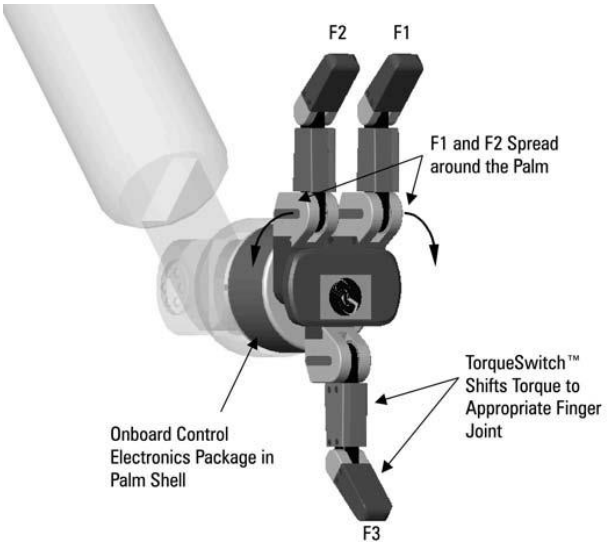

Fig. 5. The BarrettHand

IV. Modeling

A. Assumptions

Although simulations try to make the physics as close to the real-world conditions as possible. A few assumptions have been made in this project in order to make things simpler. The assumptions are listed below,

- The 4 motors are placed at the vertices of a square diagonal length 2L
- All motors have infinite thrust
- All the 4 motors exert the thrusts in the same direction.
- The center of mass of the quadcopter is at the center of the aforementioned square
- The Ground effect is neglected
- Air drag is neglected
- The position and orientation of the object with respect to frame 0 is known (In a real-world scenario, this can be found through computer vision algorithms).
- the shape of the object to be pick up is either a cylinder, or a cuboid.
- We assume that we have perfect position control of each joint, ie, we can set any of the joints of the quadcopter to any position we desire.
- The object can be gripped using friction
- Since a large variation of θ_2^* would cause the arm to collide with quadcopter itself, we limit the joint 2 to the range, $[-90, +90]$ degrees.

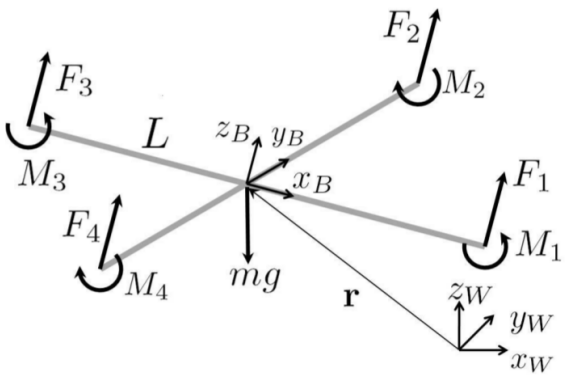

Fig. 6. The Quadcopter Model

- Since the gripper we are using is symmetrically about its y-axis as shown in fig 4, we limit we limit the joint 3 to the range, $[-90, +90]$ degrees.

B. Quadcopter Model

We use the quadcopter model mentioned in [7], because it simplifies the control equations for the quadcopter. The quadcopter is assumed be propelled by four motors placed at the ends of a square with a diagonal length of $2L$. The world frame, W is defined by x_W,y_W and z_W, where z_W points up. As shown in Figure 6, x_B coincides with the forward direction and z_B is perpendicular to the square which the motors lie on. Motor 1 lies on the positive x_B-axis, 2 on the positive y_B-axis, 3 on the negative x_B-axis and 4 on the negative y_B-axis. The $Z-X-Y$ Euler angles model the rotation from the world frame,W to the Body Frame, B. First we rotate about z_W by the yaw angle, ψ, then we rotate about x_W by ϕ and in the end we rotate about the y_B axis by the pitch angle, θ.

We can then generate the rotation matrix from B to W as,

$$M = \begin{bmatrix} c\psi c\theta - s\phi s\psi s & -c\phi s\psi & c\psi s\theta + c\theta s\phi s\psi \\ c\theta s\psi + c\psi s\phi s\theta & c\phi c\psi & s\psi s\theta - c\psi c\theta s\phi \\ -c\phi s\theta & s\phi & c\phi c\theta \end{bmatrix}$$

Please note that in this report, $c\theta$ represents $cos(\theta)$ and $s\theta$ represents $sin(\theta)$ We also assume motors M1 and M3 to act in the $-z_B$ direction, and motors M2 and M4 to act in the z_B direction, as per the right hand rule.

C. Motor Model

Through experimentation, it has been found that a rotor with an angular velocity ω_i produces a, vertical force F_i according to,

$$F_i = k_F \omega_i^2$$

. In addition, each rotor produces a momentM_i according to,

$$M_i = k_M \omega_i^2$$

The constant k_M is determined to be about $1.5 \times 10^{-9} \frac{Nm}{rpm^2}$. by matching the performance of the simulation of the simulation of real result. The exact relationship between the actual and commanded motor speed is a complicated function of the motor controller and the propeller and motor dynamics. The true performance is a function of the speed of the rotor and whether the speed is increasing or decreasing. However, for simplicity a simple first order motor model is used for controller development and simulation throughout this work. The rotor speed is approximately related to the commanded speed by a first-order differential equation

$$\dot{\omega_i} = k_m(\omega_1^{des} - \omega_1)$$

. This motor gain k_m is found to be about $20s^-1$by matching the performance of the simulation to the real system. The desired angular velocity ${\omega_i}^{des}$, are limited to a minimum and maximum value determined through experimentation to the approximately $1200rpm$ and $7800rpm$

D. Arm Model

In order to grasp any object in the world with the gripper on the quadcopter, we have to find the position of joints which would enable us to achieve that position and orientation. This means that we have to perform inverse kinematics on the arm. Since this arm only consist of 3 degrees of freedom, the transformation matrix from the base frame T_3^0 was found and then the joint angles and positions were found algebraically. The forward kinematics was found as follows,

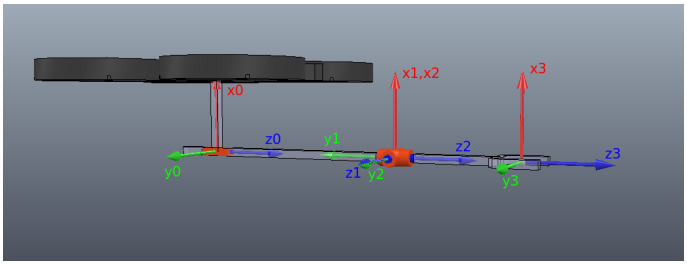

Fig. 7. The Quadcopter Model

1) Forward Kinematics: The frames were chosen as shown in fig 7 . Please note that the frame F_0, placed at the origin of joint 1, was used as the base frame. All frames were placed as per the DH conventions.

For a transformation, from frame 0 to 1, the parameter d is the distance between the origin o_0 and the intersection of the x_1 axis with z_0 measured along the z_0 axis. θ is the angle between x_0 and x_1 measured in a plane normal to z_0 . Finally, The DH table for the arm is given below. Where d_1^*, θ_2^* and θ_3^* are the joint angles for the first second and the third joint.

	θ	d	a	α
0-1	0	$l_1 + d_1^*$	0	-90
1-2	θ_2^*	0	0	90
2-3	θ_3^*	l_2	0	0

DH table

The following transformation matrices were found from the above DH Table. Please note that the transformation matrix representation format is as per [12]

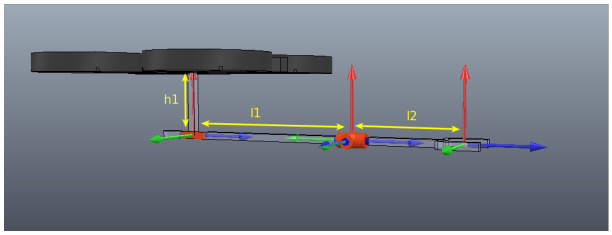

Fig. 8. Link lengths

$$T_1^0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & l_1 + d_1^* \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_2^1 = \begin{bmatrix} c\theta_2^* & 0 & s\theta_2^* & 0 \\ s\theta_2^* & 0 & -c\theta_2^* & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_3^2 = \begin{bmatrix} c\theta_3^* & -s\theta_3^* & 0 & 0 \\ s\theta_3^* & c\theta_3^* & 0 & 0 \\ 0 & 0 & 1 & l_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Multiplying these three matrices together, we get the transformation from frame 0 to frame 3 as,

$$T_3^0 = \begin{bmatrix} c\theta_2^* c\theta_3^* & -c\theta_2^* s\theta_3^* & s\theta_2^* & l_2 s\theta_2^* \\ s\theta_3^* & c\theta_3^* & 0 & 0 \\ -c\theta_3^* s\theta_2^* & s\theta_2^* s\theta_3^* & c\theta_2^* & l_2 c\theta_2^* \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2) Inverse Kinematics: Inverse Kinematics refers to the problem of finding the joint position given the position and orientation of the end-effector. In our system, as part of inverse kinematics, we will have to calculate the values of $\theta_2^*, \theta_3^*, d_1^*, x_{quad}, y_{quad}$ and z_{quad}

From the assumption that the position and orientation of the object with respect to the first joint is known, we would get a transformation matrix, that would be equivalent to T_3^0.

By analyzing the second column, second row element and the third column, third row element in T_3^0 we can find the values for $\theta_3^* s$ and $\theta_2^* s$ respectively as follows,

$$\theta_2^* = atan2(T_3^0{}_{1,2}, T_3^0{}_{3,3})$$

$$\theta_3^* = atan2(T_3^0{}_{2,1}, T_3^0{}_{2,2})$$

In order to find the first joint position, let us assume that the position of the object with respect to the world frame is, $[x_{obj}, y_{obj}, z_{obj}]$. The position setpoints(with respect to the world frame) given to the the quadcopter are:

$$x_{quad} = x_{obj} - l_1 - l_2 c\theta_2^* - d_{rt}$$

$$y_{quad} = y_{obj}$$

$$z_{quad} = z_{obj} + h_1 - l_2 s\theta_2^*$$

Where h_1 is the vertical offset of joint 1 from the quadcopter center as showon in fig 8 and d_{rt} is the distance we retract joint 1 while approaching the quadcopter. h_1 is equal to 0.1m and d_{rt} was taken as 0.2. As mentioned before, in order to ensure that the gripper doesnt interfere with the quadcopter body or propellers we first approach the object to be picked up such that gripper is retracted,ie, $d_1^* = 0$. After waiting for the controller to stabilize the quadcopter to the given setpoint, we then set $d_1^* = d_{rt}$ and gripper frame will coincide with the object frame.

V. Link Weight Thrust Compensation

In this section, we calculate the extra thrust that is required to to be generated by the motors in order to compensate for the weight and torque generated by the the effect of gravity on the arm links. We first assume that the masses of links with lengths h_1, l_1 and l_2 have masses m_0,m_1 and m_2 respectively.

Since the link h1 is exactly below the center of mass of the quadcopter, it only generates a linear force at the center of mass of the quadcopter. But the links l_1 and l_2 are offset from the center of mass in the horizontal plane. And hence the weight of these links will generate a torque at the center of mass of the quadcopter. This torque would be function of θ_2^* and d_1^* since the position of the links with respect to the center of mass is dependent on these joint parameters. The total downward force caused by the weights of the links would be constant and would be given by,

$$F_z = -(m_0 + m_1 + m_2)g$$

This extra force will be provided by all the four motors equally.

Now we calculate the torques generated by the weights m_1 and m_2. The center of mass of the links are always offset from the center of mass of the quadcopter along a new axis, which is a frame that is obtained by rotating the quadcopter frame by +45 degrees along the z_B axis. The position along this x-axis would be given by,

$$l_{1com} = l_1 + d_1^*$$

$$l_{2com} = l_1 + d_1^* + l_2 c\theta_2^*$$

Correspondingly, the torque generated by these weights at the center of mass of the quadcopter would be,

$$\tau_{m1} = m_1 g(l_1 + d_1^*)$$

$$\tau_{m2} = m_2 g(l_1 + d_1^* + l_2 c\theta_2^*)$$

In order to counter this torque, the 2 rotors in the front as per the above mentioned x-axis would have to generate extra torque while the 2 motors in the back would have to create lower thrust. Such a thrust change of F from one motor would hence create a torque, $F\sqrt{L}$. Therefore, the thrust change required by each motor due to the torques due to the link weights would be

$$F_\tau = (m_1 g(l_1 + d_1^*) + m_2 g(l_1 + d_1^* + l_2 c\theta_2^*))/4\sqrt{L}$$

The total thrust change required by the motors is given by,

$$F_{1change} = F_\tau + Mg/4$$

$$F_{2change} = F_\tau + Mg/4$$

$$F_{3change} = -F_\tau + Mg/4$$

$$F_{3change} = -F_\tau + Mg/4$$

where, $M = (m_0 + m_1 + m_2)$

VI. Small Angle Control

Small angle control starts with the assumption that the quadcopters roll, pitch orientation does not vary much from the hover condition. And this assumption is valid for most practical autonomous quadcopter applications, where the drone is not expected to follow very aggressive trajectories. The small angle controller mentioned in [7] uses 2 nested feedback controllers as shown in Fig 9. The low level the attitude controller, which controls the roll, pitch and yaw angles of the quadcopter runs at about 1 KHz[8]. Inertial Measurement Sensors are used to obtain feedback data for the attitude controller.

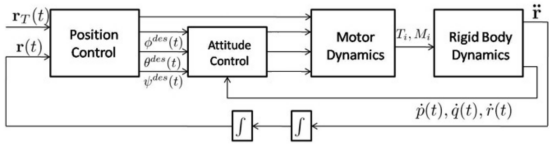

Fig. 9. Small angle control [7]

The second level nested control is the position controller whose output is fed to the attitude controller. In real-world scenarios, GPS, Camera based localization or Motion Capture Systems are used for position feedback.

In addition to these two controller, for our simulation we added a velocity controlled in between as shown in fig **??**. The velocity controller helps by constraining the velocity of the quadcopter when the position setpoints are changed quickly.

A. Control Model

In most quadcopter control models, we assume that the motor dynamics are relatively fast, ie, the motor quickly reaches its angular speed setpoint. Including the motor increases the complexity without a lot of improvement in performance.

The state of the model is taken as the position and velocity of the center of mass and the orientation and the angular velocity.

$$X = [x, y, z, \phi, \theta, \psi, \dot{x}, \dot{y}, \dot{z}, p, q, r]^T$$

Here orientation is only locally parametrized by the Euler angles. The input is

$$u = [u1, u2, u3, u4]^T$$

where $u1$ is the total thrust from the propellers, and $u1, u2, u4$ are the moments about the body frame axes.

B. Attitude Control

The attitude control in a quadcopter controls the orientation of a quadcopter. To achieve a stable flight, the roll and pitch angles need to be changed in order to follow the required trajectory. Several different control methods can be used to achieve stability. Most commercial systems use LQR or PID controllers. The first controller to be used for the purpose of attitude control in the quadcopter system was the PID, later followed by the LQR controller and very recently a combination of optimal control schemes, PID controllers and other advanced controllers are being used.

1) Proportional-Integral-Derivative Controller: The Proportional-Integral-Derivative or the PID controller gives the simplest solution to the various real-world problems. Both the transient and steady state response are taken are of with its three-term functionality i.e. proportional, integral and derivative. The PID is a control loop feedback system mechanism used in the majority of the Industrial control System. Since the invention of the PID controller both its usages and popularity has increased due to the advances in digital technology. The equation for the PID controller is given by

$$u(t) = K(e(t) + \frac{1}{T_i}\int_t^0 e(\tau)d\tau + T_d\dot{e}(t))$$

where $e(t) = y_{sp} - y$ is the error between the measured process variable and the reference signal, often call the set point T_i is the integral time and t_d is the derivative time. If the process variable cannot be measured, then we take the observation of the estimates of the measurable outputs.

C. PID Attitude Control

For any quadcopter to be in the stable and hover state the total force generated by the Quadcopter should be equal to the total forces acting on it i.e. the force must be equal to the product of its mass and gravity on the object, this is given by

$$F_i = mg$$

where F_i is the total force generated by the rotors of the Quadcopter. The Attitude control for the small angle control is discussed in this section. The attitude control is used to track trajectories in $SO(3)$ that are close to the nominal hover state where the roll and pitch angles are small. From the equation of general angular acceleration in the Motor model, if we assume that the products of Inertia are small which are Ideally the products of inertial are zero because the axes ares close to the principle axes and because of symmetry $I_{xx} \approx I_{yy}$ then the euler equation becomes

$$I_{xx}\dot{p} = u_2 - qr(I_{zz} - I_{yy})$$

$$I_{yy}\dot{q} = u_3 - pr(I_{xx} - I_{zz})$$

$$I_{zz}\dot{r} = u_4$$

we can also assume the component of the angular velocity in the z_B direction , r is small so the rightmost terms from the first two equations above, which are products involving the value of r is small compared to the other terms. We note that near the nominal hover state $\dot{\phi} \approx p, \dot{\theta} \approx q$ and $\dot{\psi} \approx r$. For these reasons we can use simple proportional derivative control laws that take the form

$$u_{2,des} = k_{p,\phi}(\phi^{des} - \phi) + k_{d,\phi}(p^{des} - p)$$

$$u_{3,des} = k_{p,\theta}(\theta^{des} - \theta) + k_{d,\theta}(q^{des} - q)$$

$$u_{4,des} = k_{p,\psi}(\psi^{des} - \psi) + k_{d,\psi}(r^{des} - r)$$

the vector of the desired rotor speeds can be found from the desired net force ($u_{1,des}$ and the moments ($u_{2,des}, u_{3,des}$ and $u_{4,des}$) by inverting we get the inverted matrix.

D. Velocity Control

Although a velocity controller is not mentioned in the small angle control scheme in [7], practical implementations of quadcopter controllers such as [9] uses a velocity controller in between the position and attitude controllers. The cascaded controller scheme used for our simulations is shown in Fig **??**. The equations for the velocity PID outputs are given by

$$V_{out,x} = k_{p,v_x}(v_x^{des} - v_x) + k_{d,v_x}(\dot{v_x}^{des} - \dot{v_x})$$

$$V_{out,y} = k_{p,v_y}(v_y^{des} - v_y) + k_{d,v_y}(\dot{v_y}^{des} - \dot{v_y})$$

$$V_{out,z} = k_{p,v_z}(v_z^{des} - v_z) + k_{d,v_z}(\dot{v_z}^{des} - \dot{v_z})$$

Where v_x,v_y and v_z are the feedback values and $V_{out,x}$,$V_{out,y}$ and $V_{out,z}$ are the outputs of the velocity controller.

The velocity controller is a simple PID control for each position axis. The feedback is real-world systems are usually obtained by air flow sensors or by differentiating the position odometry values.

E. Position Control

In this section we will be talking about the position control involved with the small angle control. The presentation of two representative position control methods that uses the roll and pitch angles as the inputs to the system, using a method similar to the Back-stepping approach. The back-stepping is a process in control system designed to stabilize special cases of non-linear systems. The Hover controller is used for station-keeping or maintaining the position at a desired x, y and z location. The second tracks a trajectory in the three dimensions

1) Hover Controller: Here we use the pitch and roll angle to control position in the x_W and y_W plane, u_4 to control the yaw-angle we are trying to track. Note that $\psi_T(t) = \psi_0$ is the desired yaw angle. The command accelerations,$\ddot{r_i}^{des}$ are calculated from the PID feedback of the position error $e_i = (r_{i,T} - r_i)$ as

$$(\ddot{r_{i,T}} - \ddot{r_i}^{des}) + k_{d,i}(\dot{r_{i,T}} - \dot{r_i}) +$$

$$k_{p,i}(r_{i,T} - r_i) + k_{i,i}\int(r_{i,T} - r_i) = 0$$

where $\dot{r_{i,T}} = \ddot{r_{i,T}}$ this equality becomes zero for the hover state. Then we linearize the following equation

$$m\ddot{r} = [0 \;\; 0 \;\; -mg]^T + R_B[0 \;\; 0 \;\; F_B]$$

to get the relationship between the desired accelerations and roll and pitch angles

$$\ddot{r_1}^{des} = g(\theta^{des} cos\psi_T + \phi^{des} sin\psi_T)$$

$$\ddot{r_2}^{des} = g(\theta^{des} sin\psi_T - \phi^{des} cos\psi_T)$$

$$u_{i,des} = m\ddot{r_3}^{des}.$$

the position control loop for the above hover controller runs at the rate date is received from the input which is normally around 100 Hz, while the linear inner attitude control runs normally at around 1 kHz. There is the usual trade-off in optimizing the control gains between the speed of response and stability, for tightly optimized or "stiff" controller the horizontal positioning errors are within the tolerance level which is set to 2cm and the value of error in the vertical direction is 0.6 cm. However, this set of gains leads to a relatively small basin of attraction. By optimizing the gains for a softer response we can increase the size of this basin. The quadcopter operation was much smoother after perturbing the value to represent the loosely optimized controller.

F. PID tuning

As mentioned in [7], in order to tune the controller we consider the equation of motion about the x-axis,

$$I_{xx}\ddot{\phi} + k_{d,\phi}\dot{\phi} + k_{p,\phi}\phi = 0$$

Comparing this to the standard second order system equation,

$$\ddot{\phi} + 2\xi\omega_n\dot{\phi} + \omega_n^2\phi = 0$$

Hence we can start set the initial tuning parameters to

$$k_{p,\phi} = I_{xx}\omega_n^2$$

and

$$k_{d,\phi} = 2I_{xx}\xi\omega_n$$

But during implementation, we discovered that it was easier to manually tune the controller by setting $k_{d,\phi} = 0$ and then tuning $k_{p,\phi}$. Then after finding the value for $k_{p,\phi}$ at which the quadcopter gives constant oscillations, we increase $k_{d,\phi}$ so as to remove these oscillations

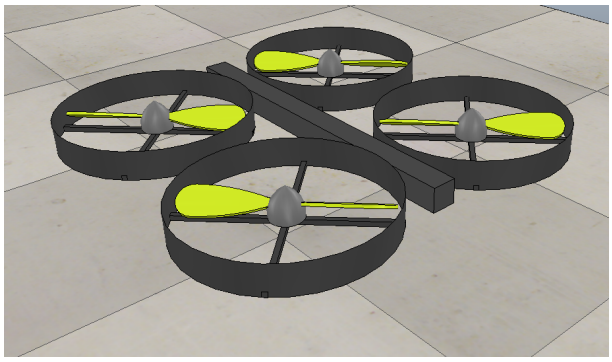

Fig. 10. Quadcopter Model

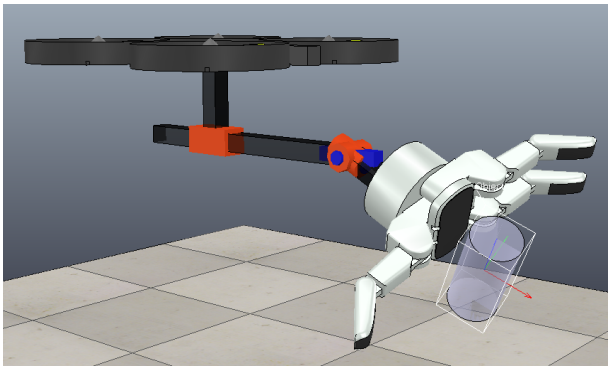

Fig. 12. Inverse Kinematics Validation

VII. Simulation and Validation

In this section we present the simulation of the quadcopter+arm setup mentioned above. The simulation was done using the VREP simulation environment[2]. The python remote API was used to simplify software development. VREP contains a model of a quadcopter, which simulated the forces generated by a motor and propeller, by applying a force and corresponding torque at the locations of the motor. An arm was attatched as mentioned before in order to validate the quadcopter+arm system. The model is controlled using the small angle control scheme mentioned before. As shown in $fig.9.$, the quadcopter was controlled using two cascaded position and attitude PID controllers, which were explained in detail before. Each lower level of control was tuned first before moving onto the next controller.

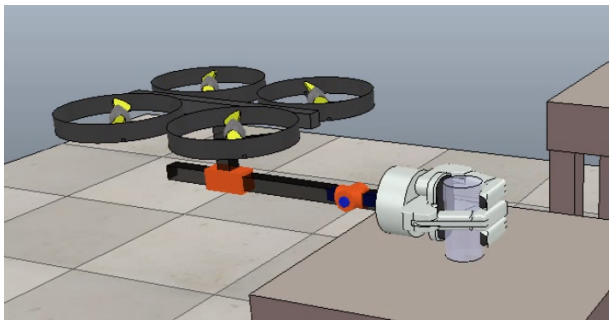

Fig. 11. Grasping

The position controller takes about 20 seconds(in simulation time) in order to converge to a position setpoint with an error less than 0.01m. Hence, after giving the position setpoint to the controller, we wait for 20 seconds, before extending the prismatic joint in the arm from d_1^* to d_{rt}.

A. Inverse Kinematics Validation

As we had mentioned before, given the position and orientation of the object to be gripped, the inverse kinematics equations mentioned before were used to set the joint positions for the arm and the position of the quadcopter.

In order to validate that the Inverse Kinematics is working correctly, we change the physical position and orientation of the object and the quadcopter automatically follows the position such that the gripper end is at the object. In the first test, we vary the position of the object. In the video, it is clearly evident that the quadcopter automatically moves to the position required to grip the object.

In the second test, we vary the orientation of the object. In the video we see that the joint angles adjust so that the object can be gripped correctly. We also notice that at certain orientation cahnges, the position of the quadcopter also changes. This is expected, since the postion of the quadcopter depends of the values for $c\theta_2^*$.

This validation can be replicated on most computers using the open source code provided at: https://github.com/ashwinvk94/armed_quadcopter

Videos from the simulation can be found at https://youtu.be/F-4HzJd8PGQ and https://youtu.be/TAcblA4842w

VIII. Conclusion

In conclusion, we have described the kinematics and design that is involved in creating a quadcoper with a 3-dof arm. We have also shown that this arm allows the quadcopter to move the gripper to any position and orientation. Although we have shown that it is kinematically possible to pick and place objects, it is important to not that we have not taken into consideration the energy required to do so.

References

[1] Furrer, Fadri Burri, Michael Achtelik, Markus Siegwart, Roland. (2016). RotorS A Modular Gazebo MAV Simulator Framework. Studies in Computational Intelligence. 625. 595-625. 10.1007/978-3-319-26054-9_23.

[2] E. Rohmer, S. P. N. Singh, M. Freese, "V-REP: a Versatile and Scalable Robot Simulation Framework," IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, 2013

[3] https://airandspace.si.edu/stories/editorial/leonardo-da-vinci-and-flight

[4] Young, Warren R. (1982). The Helicopters. The Epic of Flight. Chicago: Time-Life Books. p. 28.

[5] J.G. Leishman, The Breguet-Richet Quad-Rotor Helicopter of 1907.

[6] William Townsend, (2000) "The BarrettHand grasper programmably flexible part handling and assembly", Industrial Robot: An International Journal, Vol. 27 Issue: 3, pp.181-188, https://doi.org/10.1108/01439910010371597

[7] Mellinger, Daniel Warren, "Trajectory Generation and Control for Quadrotors" (2012). Publicly Accessible Penn Dissertations. 547. https://repository.upenn.edu/edissertations/54

[8] D. Gurdan, J. Stumpf, M. Achtelik, K. Doth, G. Hirzinger, and D. Rus. Energy- efficient autonomous four-rotor flying robot controlled at 1 khz. In Proc. of the IEEE Int. Conf. on Robotics and Automation, Roma, Italy, April 2007 .

[9] Meier, Lorenz Tanskanen, Petri Fraundorfer, Friedrich Pollefeys, Marc. (2012). The PIXHAWK open-source computer vision framework for MAVs. ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. XXXVIII-1/C22. 13-18. 10.5194/isprsarchives-XXXVIII-1-C22-13-2011.

[10] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," 2011 IEEE International Conference on Robotics and Automation, Shanghai, 2011, pp. 2520-2525.doi: 10.1109/ICRA.2011.5980409

[11] Quigley, Morgan Conley, Ken P Gerkey, Brian Faust, Josh Foote, Tully Leibs, Jeremy Wheeler, Rob Y Ng, Andrew. (2009). ROS: an open-source Robot Operating System. ICRA Workshop on Open Source Software. 3.

[12] Robot Dynamics and Control,Second Edition,Mark W. Spong, Seth Hutchinson, and M. Vidyasagar

[13] Erwin Coumans. 2015. Bullet physics simulation. In ACM SIGGRAPH 2015 Courses (SIGGRAPH '15). ACM, New York, NY, USA, pages. DOI: https://doi.org/10.1145/2776880.2792704

[14] R. R. Ma, L. U. Odhner, A. M. Dollar "Yale OpenHand Project: Optimizing Open-Source Hand Designs for Ease of Fabrication and Adoption," IEEE Robotics Automation Magazine, vol. 24(1), pp. 32-40, 2017