# MOMAV
# Marco's Omnidirectional Micro Aerial Vehicle

Marco Ruggia

***Abstract*—This paper presents a number of developments related to omnidirectional aerial vehicles, meaning vehicles with decoupled translational and rotational dynamics. In particular it concerns itself with multicopters that actively rotate their arms to achieve omnidirectionality. Proposed are, a highly symmetrical placement of rotor-arms, a small-scale rotating arm assembly that allows for continuous rotation, and a SQP based control allocation algorithm. These approaches are implemented in a purpose-built drone and results of testflights are presented.**

## I. Introduction

Omnidirectional aerial vehicles are able to exert forces and torques independently, enabling them to decouple translational and rotational dynamics. This characteristic has been found to be particularly useful in the many manipulation tasks, when aerial vehicles need to be in contact with the environment [3, 7]. Less commonly they can also be used as an alternative to gimbals for visual inspection and surveying applications.

Omnidirectional aerial vehicles are most commonly of the multicopter kind and come in mainly two designs. Fixed-tilt designs where the rotors are fixedly tilted in a nonparallel fashion [4, 5], and variable-tilt designs where the rotor arms can be actively rotated [2]. This paper proposes possible improvements to the existing variable-tilt designs. In particular three contributions are presented:

- A highly symmetrical rotor-arm configuration in the shape of an octahedron, including its practical implementation.
- A small-scale rotating arm assembly providing continuous rotation by use of a modified hobbyist servo motor and a custom slip ring.
- A control allocation algorithm based on Sequential Quadratic Programming (SQP), allowing for the use of easily tunable objective functions.

These contributions are implemented in a purpose-built drone (Fig. 1) and testflights are performed to showcase its capabilities. Since this is a hobbyist project, a special emphasis was placed on keeping overall cost of the drone low and on limiting the required tools and materials to commonly available ones. As a result, the drone is relatively simple and easy to build.

## II. Body Geometry

One commonality of the current generation of variable-tilt omnidirectional aerial vehicles, is that they use rotating arms that are all coplanar. In this configuration the load carrying capacity and efficiency depend highly on the orientation,

Fig. 1: Prototype drone *MOMAV*, used to evaluate proposals

which can be an undesirable trait. Instead, this project proposes a more symmetrical placement of the arms.

To compare various geometries, the maximum load carrying efficiency is used, meaning the fraction of total available thrust that can be applied upwards for any given orientation. This value is not a particularly good metric, as multicopter rarely operate at or near maximum load, but it's thought to be roughly comparable to efficiency under more reasonable conditions.

$x_i$: rot. axis of arm $i$, $q$: body orientation

$$n_i = \text{normalize}\left( \left(q x_i \times (0,0,1)^T\right) \times q x_i \right)$$
$$\text{effic.} = \frac{1}{N} \sum_{i=1}^{N} n_i \cdot (0,0,1)^T \tag{1}$$

A natural starting point for choosing symmetrical arm placements are the vertices of the platonic solids. The options with a reasonable number of arms are: a tetrahedron (4), an octahedron (6) or a cube (8). All of these provide more than than the 6 degrees of freedom required for omnidirectionality, providing two per arm (angle & throttle). Out of these three the octahedron is chosen as a project specific trade-off between manufacturability, cost and efficiency (Fig. 2).

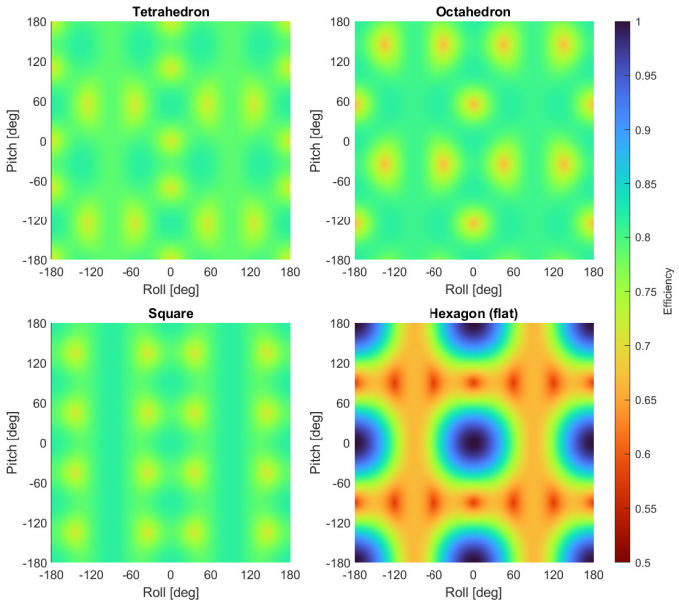


Fig. 2: Maximum load carrying efficiency vs. orientation

The octahedral body (Fig. 3) is constructed using six flat half-rings where each one follows two consecutive edges of the octahedron. Each vertex is then at a meeting point of one center and two ends of the half-rings. The arms are made with two parallel sheets placed perpendicularly to the passing half-ring at each vertex. Two halved joints on the passing half-ring prevent the arm from moving up-down, while the two half-ring ends prevent it from moving left-right. Twisting is addressed by connecting the three edges of each face of the octahedron with hexagons, creating a truncated octahedron shape. Relying mainly on interlocking contacts for connections instead of friction from screws was found to lead to a more rigid construction with no noticeable play.

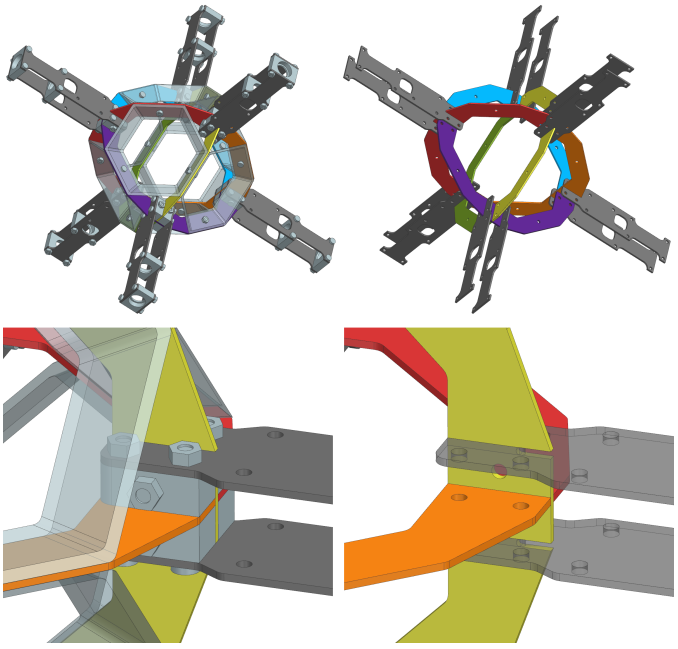

Fig. 3: Detailed views of the octahedral body implementation

## III. Rotating Arm Assembly

A second noteworthy part of this project is the rotating arm assembly. It's function is to position the propellers at any given angle around the arms. To maintain the high degree of symmetry, a requirement of this assembly is to allow for continuous rotation. This decision has the added benefit of simplifying the flight controller. In return, two challenges appear. How can the continuous angular positioning be achieved and how can power then be transmitted to the propellers.

### A. Arm Actuator

The proposed solution for the arm actuators is largely driven by the low-cost constraint. While suitable small-scale and high-accuracy actuators are available, none are particularly affordable. To overcome this challenge a ~100° limited servo, commonly found in model aircraft, is modified to support continuous rotation.

The servo used is the *KST MS325*. Unlike most servos of this type it uses a magnetic encoder as feedback for position control, instead of a potentiometer. This gives the servo the mechanical ability to be rotated continuously. The encoder used is the *AMS AS5600*, configured by default to output an analog voltage emulating a potentiometer. The stock feedback control circuit is then the same as in potentiometer based servos. To achieve continuous rotation this control circuit is replaced by an *ATSAMD21* microcontroller and a *TI DRV8870* motor driver. The encoder is then connected trough its I2C interface, to unlock the full 360° measuring range and to provide the many well known advantages over an analog interface. Feedback control is implemented with a PID loop running on the microcontroller in a way that allows for setpoints outside the 0°-360° range. To avoid overshoots a *proportional on measurement* [1] approach is used. Tuning is performed manually by trial and error.

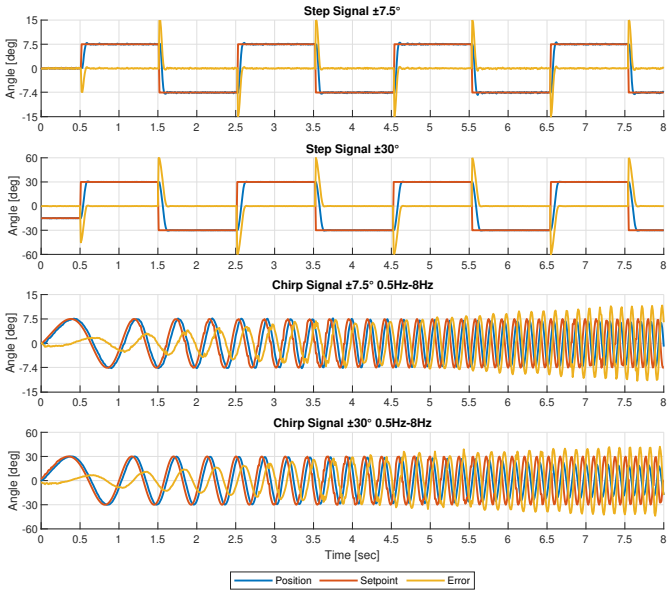


Fig. 4: Servo arm responses to step and chirp signals with propeller spinning at 80% throttle (~15N thrust)

| Parameter | Value |
|---|---|
| Proportional gain (on measurement) | 7.6 / rad |
| Integral gain | 220.0 / (rad·sec) |
| Derivative gain | 0.05 / (rad/sec) |
| PID loop rate | 1 kHz |
| Encoder polling rate | 4 kHz |
| Encoder exp. smoothing constant | 0.2 |
| Output range | ±1.0 |

TABLE I: Parameters of arm servo control loop

This approach to arm actuation has produced very satisfactory results, considering the relatively low cost and small size. Some key figures are a peak velocity of 2.5 rev/sec, peak torque of 0.51 Nm and the fast dynamic behaviour as shown in Figure 5. It appears that the main contributor to tracking error is the maximum velocity, as there are no significant time delays and the acceleration is comparatively fast.

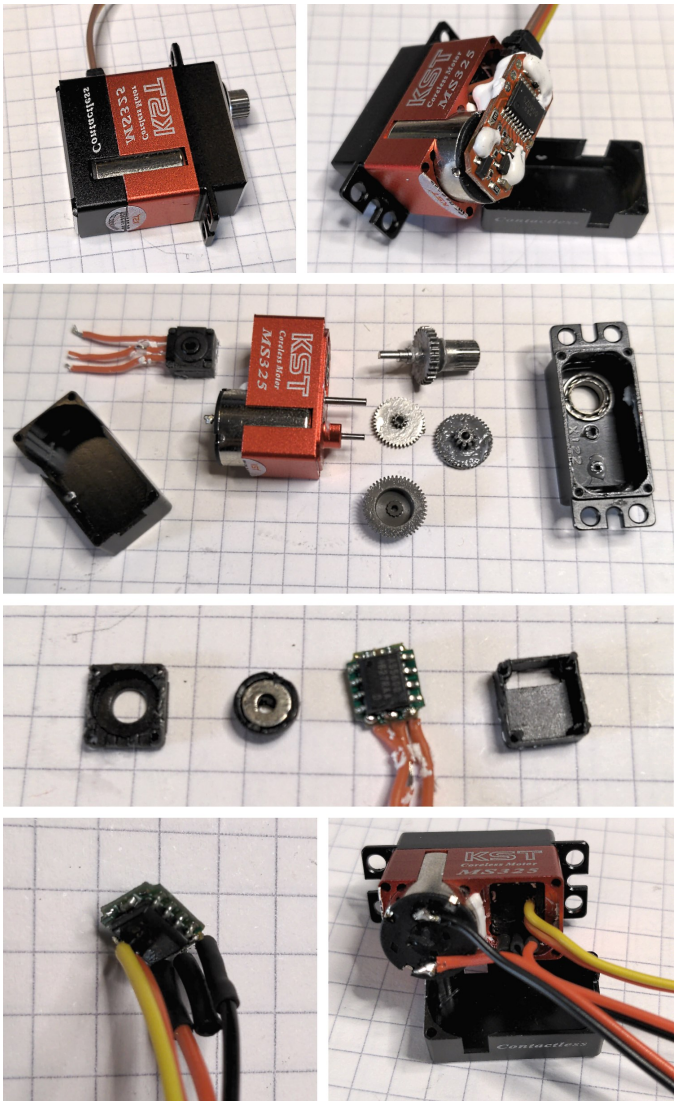

Fig. 5: Modification steps of the *KST MS325* servo for continuous rotation, not including the new control board

### B. Slip Ring

The other challenge of the rotating arm assembly is to transmit electrical power from the stationary part of the arm to the rotating part for the propeller motor. Such a component is commonly called a slip-ring. Like for the arm actuators, it is difficult to find one in the correct size, cost & power rating range. This project proposes a solution using brushes harvested from the ubiquitous *RS550* DC motors (Fig. 6).

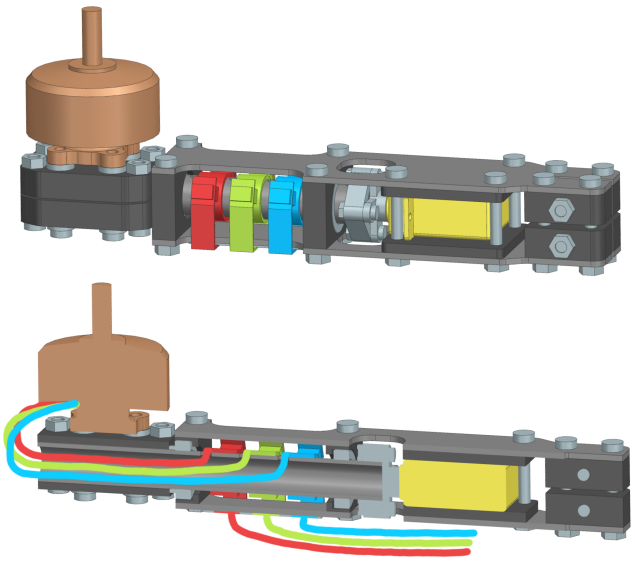

Fig. 6: Rotating arm assembly with a section view showing internal wiring of the slip-rings

In practice three slip-rings are built for each phase of the propeller motors, as this makes packaging easier than the alternative of having slip rings on the other side of the motor controllers. Each phase has two brushes soldered to a carrier circuit board such that their contacts push against a rotating tube. The tube is isolated by electrical tape and under the contact patches three copper rings are glued. Next to each ring a hole is drilled into the tube through which an electrical wire is routed and soldered to the side of the copper ring. These wires connect to the propeller motor, while the carrier board goes on to connects to the motor controller.

A test is performed to determine what the capabilities of this assembly are. For this purpose six slip-rings are individually subjected to constant currents ranging from 1 A to 5 A and the voltage drop-off is measured to calculate power loss. The low maximum tested current is due to limits of the used power supply. During all tests a small USB fan is used to cool the slip-rings, similarly to how the propeller would.

Results of these tests are plotted in Figure 7. For the here discussed drone, 5A current per motor is roughly what's needed for hover. That current over two slip-rings at a time induces a loss of about 4 W, which is 3.2% of total power. This result is acceptable for the given drone, but is probably improvable with shorter brushes and more care in producing consistent contact patches.

Concerns on heat dissipation at higher currents are recognized but could not be easily tested.

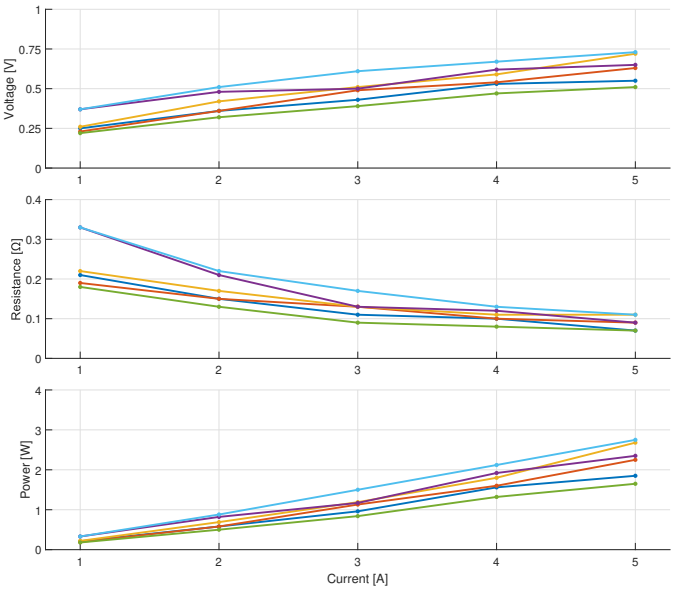


Fig. 7: Results of the slip-ring power loss tests on six samples

## IV. Control Allocation

The last aspect of this project that is presented is the control allocation algorithm. It's goal is to calculate the target arm angles and throttle values, such that some total force and torque setpoint is met. This system is highly overactuated with 12 variables (6 arm angles & 6 throttles) and only 6 constraints (3 forces & 3 torques). For this reason a numerical optimization approach is use where the constraints have to be satisfied ($G = 0$) and the remaining degrees of freedom are chosen such that they minimize some objective function ($O$).

$$(u^*, a^*) = \text{argmin } O(u, a) \text{ s.t. } G(u, a, q, F, M) = 0 \quad (2)$$

| Name | Description | Type |
|---|---|---|
| **Inputs:** | | |
| $q$ | body orientation quaternion | $\in \mathbb{H}$ |
| $F$ | desired body force | $\in \mathbb{R}^3, [N]$ |
| $M$ | desired body torque | $\in \mathbb{R}^3, [Nm]$ |
| **Outputs:** | | |
| $a_i$ | angle of arm $i$ | $\in \mathbb{R}, [rad]$ |
| $u_i$ | motor throttle of arm $i$ | $\in \mathbb{R}, (0, 1)$ |
| **Constants:** | | |
| $r_i$ | arm endpoint | $\in \mathbb{R}^3, [m]$ |
| $x_i$ | arm rotation axis | $\in \mathbb{R}^3, \|\cdot\| = 1$ |
| $z_i$ | arm zero angle thrust direction | $\in \mathbb{R}^3, \|\cdot\| = 1$ |
| $s_i$ | motor spin direction | $\in \{-1, 1\}$ |
| $\mu$ | motor thrust constant | $\in \mathbb{R}, [N/1]$ |
| $\tau$ | motor torque constant | $\in \mathbb{R}, [Nm/1]$ |
| **Intermediaries:** | | |
| $n_i$ | arm thrust direction | $\in \mathbb{R}^3, \|\cdot\| = 1$ |
| $f_i$ | arm force | $\in \mathbb{R}^3, [N]$ |
| $m_i$ | arm torque | $\in \mathbb{R}^3, [Nm]$ |

TABLE II: Variable names and descriptions

For this particular implementation the objective function $O$ is chosen as the sum of throttle values squared with a weight $w_u$ and the arm angle velocities squared with a weight $w_a$, aiming to minimize both. A penalty weight $w_p$ is used to push the throttles towards a range $u_{pl}$ to $u_{ph}$, with the goal of keeping them within achievable limits without the added complexity of inequality constraints.

$$\begin{aligned} O = \sum_{i=1}^{6} \; & w_u u_i^2 + \frac{w_a}{\Delta t}(a_i - a_{i,prev})^2 \\ & + (w_p(u_i - u_{pl})^2)_{u_i < u_{pl}} \\ & + (w_p(u_i - u_{ph})^2)_{u_i > u_{ph}} \end{aligned} \quad (3)$$

The constraint $G$ is defined as the difference between the desired body force/torque $F, M$ and the sum of force/torques $f_i, m_i$ achieved by the inputs $u_i, a_i$. When they match, it follows that $G = 0$ and the constraint is satisfied. Note that in the following equations quaternions are multiplied with vectors and the result is defined to be the rotated vector.

$$n_i = \begin{bmatrix} \sin(a_i/2) \\ x_i \cos(a_i/2) \end{bmatrix}_{\in \mathbb{H}} z_i \quad (4)$$

$$f_i = \mu u_i n_i \quad (5)$$

$$m_i = \mu u_i (r_i \times n_i) + \tau u_i s_i n_i \quad (6)$$

$$G = \begin{bmatrix} \sum_{i=1}^{6}(f_i) - q^{-1}F \\ \sum_{i=1}^{6}(m_i) - q^{-1}M \end{bmatrix} \quad (7)$$

Now that the optimization problem is fully defined, a few characteristics can be noted:

- The problem is nonlinear due to the trigonometric functions in the constraint $G$, suggesting an iterative solving approach.
- The objective $O$ is a convex function, meaning there is no risk of converging to a maximum instead of a minimum.
- The constraint $G$ is not convex, meaning a solver could converge to a suboptimal solution that e.g. asks an arm to rotate more than half a turn. Sensibly limiting the step-size is in practice enough to avoid this problem.

The method chosen to calculate an iterative improvement on $u, a$ is sequential quadratic programming (SQP) [6]. In an effort to make the resulting equation more understandable, it is derived by applying Newton's method to the optimality conditions, instead of directly trough the quadratic programming subproblem. These optimality conditions are the here relevant parts of the Karush-Kuhn-Tucker (KKT) conditions, namely:

- **Primal Feasibility:** $G = 0$, stating that the constraint must be satisfied at the solution.
- **Stationarity:** $\nabla O + \nabla G^T \lambda = 0$, stating that the gradient of the objective must be a linear combination of the gradients of every constraint at the solution. This ensures that there is no direction that is orthogonal to every constraint gradient and at the same time is not orthogonal to the objective gradient, which would allow the objective to be lowered while maintaining constraint satisfaction.

These two equations are packed into one and the derivatives split into parts with respect to $u$ and $a$.

$$\begin{bmatrix} O_u + {G_u}^T\lambda \\ O_a + {G_a}^T\lambda \\ G \end{bmatrix}_{\substack{u=u^* \\ a=a^* \\ \lambda=\lambda^*}} = 0 \tag{8}$$

This equation is then solved iteratively with Newton's method, by taking the linear approximation at the current best guess $(u, a, \lambda)$ to find an improvement step $(\delta u, \delta a, \delta\lambda)$ which solves the approximated equation.

$$H = \begin{bmatrix} O_{uu} + G_{uu} : \lambda & O_{ua} + G_{ua} : \lambda & {G_u}^T \\ O_{au} + G_{au} : \lambda & O_{aa} + G_{aa} : \lambda & {G_a}^T \\ G_u & G_a & 0 \end{bmatrix} \tag{9}$$

$$\left( \; (G_{\star\bullet} : \lambda)_{ij} = (\partial^2 G / \partial \star_i \partial \bullet_j)^T \lambda \; \right)$$

$$K = \begin{bmatrix} O_u + {G_u}^T\lambda \\ O_a + {G_a}^T\lambda \\ G \end{bmatrix} \tag{10}$$

$$\hookrightarrow \begin{bmatrix} \delta u \\ \delta a \\ \delta\lambda \end{bmatrix} = H^{-1}K \tag{11}$$

Usually at this point a line-search method is required to determine how much of the step should be taken, as to not leave the region where the linear approximation is accurate. In this use-case it's found to be sufficient to simply scale the step by $\alpha \in (0, 1)$ such that no single angle nor throttle step exceeds a fixed limit.

$$\alpha = \min(1.0, \frac{\delta a_{lim}}{\max(\delta a)}, \frac{\delta u_{lim}}{\max(\delta u)}) \tag{12}$$

**Algorithm 1** SQP algorithm

**Input:** $q, F, M$
**Output:** $u, a, \lambda$
  $u, a, \lambda \leftarrow$ result from previous run
  **repeat**
    $H,\ K \leftarrow$ calculate with (9), (10)
    $(\delta u, \delta a, \delta\lambda) \leftarrow$ solve $H^{-1}K$
    $\alpha \leftarrow$ calculate with (12)
    $(u, a, \lambda) \leftarrow (u, a, \lambda) + \alpha(\delta u, \delta a, \delta\lambda)$
    $O,\ G \leftarrow$ calculate with (3), (7)
  **until** $|O - O_{prev}| \,/\, O <$ tol. **and** $\|G\| <$ tol.

In practice, the entire algorithm is implemented in C++ and run on a *Raspberry Pi 4B*, serving as the flight controller. The *PartialPivLU* solver from the *Eigen* library is used to solve the linear system. This setup manages to converge to a solution reliably within 5 ms (200 Hz) or less. It manages to do so even during very fast orientation changes, when the solution is furthest from the initial guess.

Proposed future developments could involve more complex objective terms, nonlinear motor models, or even accounting for arm & motor dynamics and gyroscopic effects.

## V. Testflights

The three approaches presented in this paper are all tested together in a purpose built drone. The goal of the testflights is neither to quantify the effects of these approaches, nor to compare them to other existing solutions. Instead, the flights are simply used to show that these approaches can in principle produce a drone capable of stable flight. There are a few reasons for this very conservative objective. All are related to the inability of the author to continue working on this project.

- For safety reasons a tether is needed on the drone. It's current implementation greatly limits the achievable orientations, as it would intersect the propellers. For this reason only level flight is tested.
- No high-level translation control is implemented, as the drone does not have any means to measure its position. Instead, translation is controlled manually with a RC radio transmitter acting directly on the force setpoint. This causes large drifts in position of 1-2 m.
- A simple PD controller is implemented for high-level orientation control, which uses the current/desired orientation to calculate a torque setpoint. A PID controller or other more sophisticated control methods could prove beneficial.
- A limited amount of effort is put into the tuning and identification of all the various parameters in the low and high-level controllers.

The results of a hover test (Fig. 8) show acceptable orientation tracking and stable flight. Concluding, the author hopes to soon be able to resume work on this project and to solve the remaining issues.

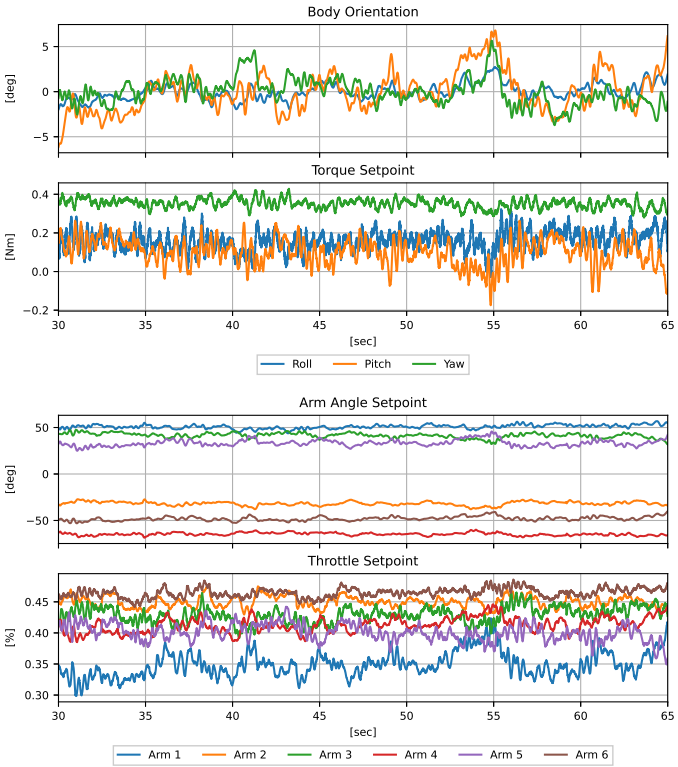


Fig. 8: Results of a hover test flight

## References


[1] Brett Beauregard. Introducing proportional on measurement. http://brettbeauregard.com/blog/2017/06/introducing-proportional-on-measurement/, 2017.

[2] Karen Bodie, Maximilian Brunner, Michael Pantic, Stefan Walser, Patrick Pfändler, Ueli Angst, Roland Siegwart, and Juan Nieto. Active interaction force control for contact-based inspection with a fully actuated aerial vehicle. *IEEE Transactions on Robotics*, 37(3):709–722, 2021. doi: 10.1109/TRO.2020.3036623.

[3] Fabio Ruggiero, Vincenzo Lippiello, and Anibal Ollero. Aerial manipulation: A literature review. *IEEE Robotics and Automation Letters*, 3(3):1957–1964, 2018. doi: 10.1109/LRA.2018.2808541.

[4] Marco Tognon, Hermes A. Tello Chávez, Enrico Gasparin, Quentin Sablé, Davide Bicego, Anthony Mallet, Marc Lany, Gilles Santi, Bernard Revaz, Juan Cortés, and Antonio Franchi. A truly-redundant aerial manipulator system with application to push-and-slide inspection in industrial plants. *IEEE Robotics and Automation Letters*, 4(2):1846–1851, 2019. doi: 10.1109/LRA.2019.2895880.

[5] Miguel Angel Trujillo, José Ramiro Martínez-de Dios, Carlos Martín, Antidio Viguria, and Aníbal Ollero. Novel aerial manipulator for accurate and robust industrial ndt contact inspection: A new tool for the oil and gas inspection industry. *Sensors*, 19(6), 2019. ISSN 1424-8220. doi: 10.3390/s19061305.

[6] Stephen Wright, Jorge Nocedal, et al. Numerical optimization. *Springer Science*, 35(67-68):7, 1999.

[7] DING Xilun, GUO Pin, XU Kun, and YU Yushu. A review of aerial manipulation of small-scale rotorcraft unmanned robotic systems. *Chinese Journal of Aeronautics*, 32(1):200–214, 2019.


## Appendix

### A. Drone Components

| | |
|---|---|
| **Core:** | |
| CPU | Raspberry Pi 4B |
| IMU | XSens MTi-3 |
| Battery | 2x Swaytronic 6S 2800mAh |
| **Arm Assembly:** | |
| Servo | KST MS325 |
| MCU | Microchip ATSAMD21 |
| H-Bridge | TI DRV8870 |
| Brushes | RS550 DC Motor Brushes |
| **Propulsion:** | |
| Motor | T-Motor F100 |
| ESC | T-Motor MINI F45A 6S 4In1 |
| Propeller | DJI Mavic Pro 8331 (3-blade-mod) |
| **Structural:** | |
| Sheets | Swiss-Composite 1.5mm CFRP Sheet |
| Arm Tubes | Swiss-Composite 8x10mm CFRP Tube |
| 3D Parts | Dutch Filament PET-G with Carbon |

### B. Drone Characteristics

| | |
|---|---|
| Weight (with battery) | 2.35 kg |
| Weight (without battery) | 1.55 kg |
| Diagonal (with propellers) | 590 mm |
| Diagonal (without propellers) | 412 mm |
| Thrust (abs. max. up) | 8.21 kg |
| Flight Time (hover) | 14 min |

### C. Derivatives Required in SQP

$$\frac{\partial \boldsymbol{O}}{\partial \boldsymbol{u_i}} = 2w_u u_i + 2w_p((u_i - u_{pl})_{u_i<u_{pl}} + (u_i - u_{ph})_{u_i>u_{ph}})$$

$$\frac{\partial^2 \boldsymbol{O}}{\partial \boldsymbol{u_i}\partial \boldsymbol{u_j}} = \begin{cases} i = j: & 2w_u + (2w_p)_{u_i<u_{pl}} + (2w_p)_{u_i>u_{ph}} \\ i \neq j: & 0 \end{cases}$$

$$\frac{\partial \boldsymbol{O}}{\partial \boldsymbol{a_i}} = \frac{2w_a}{\Delta t}(a_i - a_{i,prev}) \qquad \frac{\partial^2 \boldsymbol{O}}{\partial \boldsymbol{a_i}\partial \boldsymbol{a_j}} = \begin{cases} i = j: & \frac{2w_a}{\Delta t} \\ i \neq j: & 0 \end{cases}$$

$$\frac{\partial^2 \boldsymbol{O}}{\partial \boldsymbol{u_i}\partial \boldsymbol{a_j}} = \frac{\partial^2 \boldsymbol{O}}{\partial \boldsymbol{a_j}\partial \boldsymbol{u_i}} = 0$$

$$\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{u_i}} = \begin{bmatrix} \partial f_i/\partial u_i \\ \partial m_i/\partial u_i \end{bmatrix} \qquad \frac{\partial^2 \boldsymbol{G}}{\partial \boldsymbol{u_i}\partial \boldsymbol{u_j}} = \begin{cases} i = j: & \begin{bmatrix} \partial^2 f_i/\partial {u_i}^2 \\ \partial^2 m_i/\partial {u_i}^2 \end{bmatrix} \\ i \neq j: & 0 \end{cases}$$

$$\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{a_i}} = \begin{bmatrix} \partial f_i/\partial a_i \\ \partial m_i/\partial a_i \end{bmatrix} \qquad \frac{\partial^2 \boldsymbol{G}}{\partial \boldsymbol{a_i}\partial \boldsymbol{a_j}} = \begin{cases} i = j: & \begin{bmatrix} \partial^2 f_i/\partial {a_i}^2 \\ \partial^2 m_i/\partial {a_i}^2 \end{bmatrix} \\ i \neq j: & 0 \end{cases}$$

$$\frac{\partial^2 \boldsymbol{G}}{\partial \boldsymbol{u_i}\partial \boldsymbol{a_j}} = \frac{\partial^2 \boldsymbol{G}}{\partial \boldsymbol{a_j}\partial \boldsymbol{u_i}} = \begin{cases} i = j: & \begin{bmatrix} \partial^2 f_i/\partial u_i \partial a_i \\ \partial^2 m_i/\partial u_i \partial a_i \end{bmatrix} \\ i \neq j: & 0 \end{cases}$$

$$\frac{\partial \boldsymbol{f_i}}{\partial \boldsymbol{u_i}} = \mu n_i \qquad \frac{\partial^2 \boldsymbol{f_i}}{\partial \boldsymbol{u_i^2}} = 0 \qquad \frac{\partial \boldsymbol{f_i}}{\partial \boldsymbol{a_i}} = \mu u_i \frac{\partial n_i}{\partial a_i}$$

$$\frac{\partial^2 \boldsymbol{f_i}}{\partial \boldsymbol{a_i^2}} = \mu u_i \frac{\partial^2 n_i}{\partial {a_i}^2} \qquad \frac{\partial^2 \boldsymbol{f_i}}{\partial \boldsymbol{u_i}\partial \boldsymbol{a_i}} = \mu \frac{\partial n_i}{\partial a_i}$$

$$\frac{\partial \boldsymbol{m_i}}{\partial \boldsymbol{u_i}} = \mu(r_i \times n_i) + \tau s_i n_i \qquad \frac{\partial^2 \boldsymbol{m_i}}{\partial \boldsymbol{u_i^2}} = 0$$

$$\frac{\partial \boldsymbol{m_i}}{\partial \boldsymbol{a_i}} = \mu u_i (r_i \times \frac{\partial n_i}{\partial a_i}) + \tau u_i s_i \frac{\partial n_i}{\partial a_i}$$

$$\frac{\partial^2 \boldsymbol{m_i}}{\partial \boldsymbol{a_i^2}} = \mu u_i (r_i \times \frac{\partial^2 n_i}{\partial {a_i}^2}) + \tau u_i s_i \frac{\partial^2 n_i}{\partial {a_i}^2}$$

$$\frac{\partial^2 \boldsymbol{m_i}}{\partial \boldsymbol{u_i}\partial \boldsymbol{a_i}} = \mu (r_i \times \frac{\partial n_i}{\partial a_i}) + \tau s_i \frac{\partial n_i}{\partial a_i}$$

$$\frac{\partial \boldsymbol{n_i}}{\partial \boldsymbol{a_i}} = x_i \times n_i \qquad \frac{\partial^2 \boldsymbol{n_i}}{\partial \boldsymbol{a_i^2}} = x_i \times (x_i \times n_i)$$