Active Control and Guidance-Aided Propulsive Landing Research for Small-Scale Vehicles

Cheng Liu*
Georgia Institute of Technology, Atlanta, GA, 30332, United States of America

In recent decades, rocket launches have been primarily single-use, with an average cost of over 1 billion dollars. With the breakthrough from SpaceX, reusable systems started to be widely adopted. This study aims to apply reusable technology on a smaller scale by propulsively landing model-scale rockets using active thrust vectoring and GNSS guidance. The study involves developing a custom flight computer and software which conducts sensor fusion and various methods of data filtering, such as PID and Kalman Filtering. Through communication protocols such as I2C, the flight computer is able to conduct accurate data acquisition from the onboard IMU and barometer. In addition, a 2 degrees of freedom thrust vectoring module was implemented to receive PWM output from the flight computer and perform accurate attitude correction of the vehicle by changing the direction of the thrust. To tackle the dynamic error caused by the structural deficit, the study have constructed a matrix to address dynamic error during the actuation of the thrust vectoring module. This study provides valuable insight into the feasibility of conducting propulsive launching and landing of model-scale vehicles using GNC guidance and custom avionics.

Nomenclature

$u(t)$	=	PID control variabler
K_p	=	proportional gain
$e(t)$	=	error value
K_i	=	integral gain
$\mathrm{d}e$	=	change in error value
$\mathrm{d}t$	=	time step
ω	=	angular velocity
$\vec{d}$	=	vector pointing from the center of mass to the point of application of the thrust vector
$\vec{F_H}$	=	vector component of thrust vector which passes through the vehicle's center of mass
$\vec{F_N}$	=	vector component of thrust vector which is normal to the line of action of $\vec{F_H}$
$\vec{\tau}$	=	torque vector

I. Introduction

Inherently, most modern space launch vehicles are aerodynamically unstable – their centers of pressure are ahead of, or relatively close to, their centers of gravity. In order to counter the issues regarding instability, the idea of active Thrust Vector Control (TVC) was introduced [1].

During the process of active TVC stabilization, a vehicle's Thrust Vectoring Module (TVM) actuates with the effort of redirecting the thrust vector, creating a vector component $\vec{F_N}$. As a result, $\vec{\tau}$ can be represented by the following equation:

$$\vec{\tau} = \vec{d} \times \vec{F_N} \tag{1}$$

Following Newton's Second Law for Rotation, $\vec{\tau}$ induces a rotational motion of the vehicle about its center of gravity, forcing the heading of the vehicle to change [2]. Based on the principles of TVC, this study applies the theory by scaling down the space-shot vehicles.

*Undergraduate Student, Department of Aerospace Engineering, AIAA Student Member (1600100), cliu794@gatech.edu

II. Overview of Testing Platform

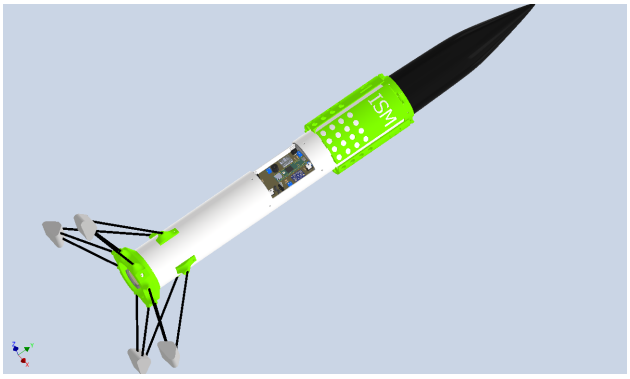


Fig. 1 3D model of the testing platform

The study constructed and designed a customized rocket with various avionics components, landing legs, and a gimballing module, allowing it to serve as a testing platform (Fig 1) to validate the active control capabilities of vehicles in small scale. The rocket is propelled by two commercial black powder rocket motors, Estes F15-0 and Estes E16.

A. Flight Computer

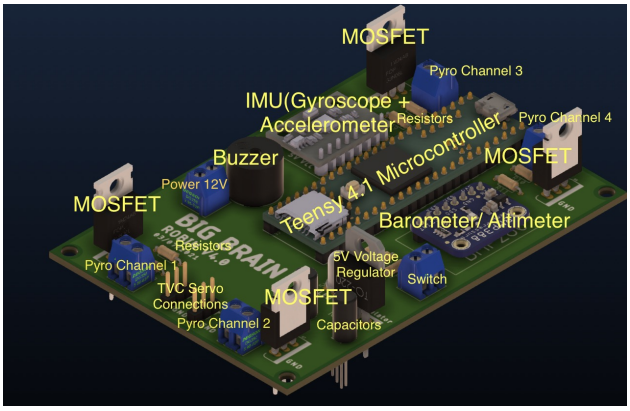


Fig. 2 Flight Computer with Various Components Annotated

1. Flight Computer (Fig. 2) Motherboard

In order to house various sensors and reduce wiring difficulties within the test vehicle, the study designed and manufactured a flight computer circuit board.

Preparing for more versatile testing applications in the long run, the flight computer was equipped with additional features such as spare PWM ports for analog output, Inter-Integrated Circuit (I2C), and Serial Peripheral Interface (SPI) ports for digital communication, and VCC power ports.

2. Main Processing Unit (MPU)

The main processing power of the flight computer originates from the Micro Controller Unit (MCU) iMXRT1060. With a nominal clock speed of 600 MHz, the MCU processes data from sensors quickly while being responsive to changes in flight. The onboard IMU and barometer accurately read the position of the rocket.

3. Attitude Sensors

The onboard MPU6050 IMU is capable of operating under ±16g. With the Digital Motion Processor integrated into the sensor, MPU6050 is able to create a stream of data of acceleration on the 3 axes as well as relatively accurate yaw, pitch, roll data while taking computational stress off from the MPU [3].

Along with the IMU, the BMP280 barometric pressure sensor is also selected to measure altitude by detecting the surrounding air pressure inside the vehicle. Communicating with the MCU through the I2C protocol, BMP280 provides valuable data to the MPU by sending Barometric pressure, Temperature, and Humidity information while only taking up two data lines.

The ZED-F9P-02B-00 GPS module provides location accuracy within 17.0 x 22.0 x 2.4 mm. Therefore, it acts as a reliable source of position when the vehicle obtains GPS lock from the satellites.

4. Telemetry

The onboard XBee telemetry module enables wireless data communication between the ground mission control station and the testing platform under the 2.4GHz band. Communicating using the ZigBee protocol, the telemetry module onboard is able to provide mission control with real-time feedback of the vehicle status such as GPS location, sensor data output, and the powering status of the flight computer. The addition of General Purpose Input/Output (GPIO) pins on the XBee module allows remote ignition from the mission control station by sending a digital `HIGH` signal to the vehicle, turning on the pyrotechnic channel configured to ignite the ascending motor.

B. Thrust Vectoring Module (TVM)

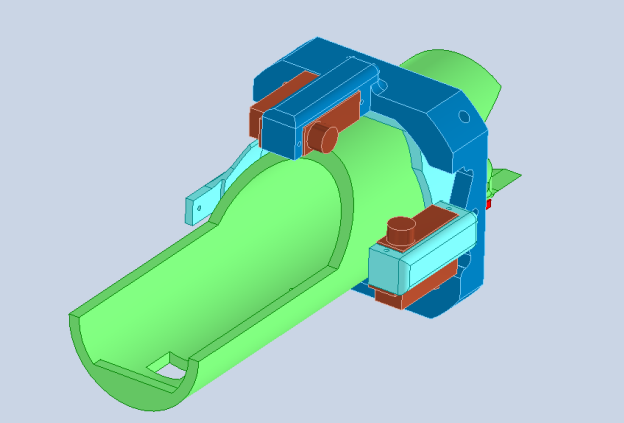


Fig. 3 3D Model of TVM with servos, motor housing, inner gimbal ring, and outer gimbal ring colored in gold, green, light blue, and dark blue, respectively.

The study designed the TVM (Fig. 3) with the intention of gimbaling motors to control the heading of the test vehicle. With two KST-X08 servos communicating through Pulse Width Modulation (PWM) signals with the flight computer, the TVM is able to pivot on the x and y axes up to ±7°. Upon reaching apogee, the TVM ejects the empty ascending motor and loads the descending motor.

C. Flight Software

Being executed on the MPU of the flight computer, the flight software, written in C and C++, runs the sensor outputs through various filters to reduce noise and improve accuracy. After determining the vehicle's orientation across the stages of flight, the flight software decides on the critical execution of the vehicle.

1. Real-Time Operating System

To improve readability and decrease the complexity of the flight software, the software architecture is written within an open-source Real-Time Operating System (RTOS) framework – HeliOS [4]. The RTOS flight software architecture includes independent tasks, such as reading data from the attitude sensors, and a scheduler that is in charge of setting

priorities and timing for tasks to be executed. When the flight software is initiated, the scheduler sets the execution frequency for the tasks from predefined constants within the code. The following code snippet illustrates the creation of IMU and Barometer tasks in the HeliOS environment.

```
    xTask xTask_YPR_Update = xTaskCreate("YPR", getYPR_main, NULL);
    xTask xTask_Altitude_Update = xTaskCreate("ALT", getAltitude_main, NULL);
```

2. Data Filtering

Applying an open-source Kalman Filter library SimpleKalmanFilter [5], the flight software uses Kalman filtering to process data from the BMP280 sensor to reduce the noise in the barometric data. The coefficient of the Kalman filter algorithm is derived by finding the standard deviation of the barometric sensor data while leaving the sensor running in oversampling mode in a motionless and undisturbed environment. The standard deviation determines the Measurement Uncertainty constant within the filter algorithm [6].

D. PID Control Loop

$$u(t) = K_p e(t) + K_i \int e(t)\,dt + K_d \frac{de}{dt} \tag{2}$$

To achieve smoother control feedback of the vehicle while avoiding significantly increasing the complexity of the software, an ordinary Proportional, Integral, and Derivative (PID) Control Loop (Eq. 2) is added to the flight software. Since the flight duration is short, the software omits the 'I' term by setting the constant K_i equal to 0. During flight, two PID loops are declared for gimbal actuation on both axes. After filtering data from the IMU with the PID loop, the flight computer converts the angle of correction ($u(t)$) into PWM signals and sends them to the respective gimbal servos. The following code snippet illustrates the declaration of the PID class in the flight software.

```
class PID {
private:
        double lastTime = millis();
        double integral = 0;
        double lastErr = 90;
public:
        double p = 0.3;
        double i = 0;
        double d = 0.01;

        double UPDATE(double curErr, double rate) { //curErr should be in degrees/s
                double curTime = millis()/1000;
                double dx = curErr - lastErr;
                lastErr=curErr;
                integral += curErr*0.005;//Riemann sum
                return p*curErr + i*integral + d*rate;
        }
};
```

E. State Estimation

The flight software for the test platform includes state estimation functions for all stages of flight (Armed, Liftoff, Powered Ascent, Unpowered Ascent, Apogee, Freefall, Landing Ignition, Landed) and changes configurations of the vehicle accordingly.

After executing the launch ignition sequence, the flight software will sample for acceleration spikes of the test vehicle caused by the ignition of the motor for the following five seconds. If successful liftoff is not detected during the 5-second launch-detect period, the flight software will enter abort mode and start disarming the flight computer and ignition channels of both motors for safety purposes.

If liftoff was successful, the rocket will switch to the Powered Ascent Mode, which will last for the duration of the burn time for the F15-0 motor. Throughout the Powered Ascent Stage, PID-controlled gimballing will be applied to the vehicle. Since the vehicle will most likely lose GPS lock during this high acceleration period, the altitude feedback will be obtained mainly from operating a dead-reckoning algorithm. In other words, the flight software will double integrate the flight acceleration to obtain the position of the vehicle.

Upon entering the Unpowered Ascent stage, the software uses GPS and barometric data for apogee detection. If the software detects that the vehicle altitude increased during the previous sampling interval and decreases during the current sample interval, the software will mark successful apogee detection and use the TVM to eject the ascent motor and load the new descent motor for landing. After the motor ejection sequence, the state machine will switch itself to freefall mode.

During the freefall stage, the flight software will closely monitor the current altitude of the vehicle and calculate the gravitational potential energy of the vehicle. Once the gravitational potential energy is equal to the energy produced by the E-16 landing motor (27853 J), the flight software will command the ignition of the landing motor and switch the state machine to Landing mode. If the apogee altitude is already lower than the desired altitude, the landing motor will be ignited right away to attempt landing.

During the landing period, the TVC will be activated again with PID controls to actively keep the rocket vertical to the ground. After no further changes in altitude and the reading is close to the ground, the flight software will switch the state of the test vehicle to the ground, disarming all the electronic components onboard, and saving the flight data.

The following code snippet illustrates the general layout of the state machine inside the flight software:

```
if(flightState == LAUNCH && block3) {
                        flightState ++;
                        block3 = false;
                }else if(flightState == ASCENDING && block4) {
                        TVC.ASCENDING();
                        Serial.println(mpu6050.RWAcc);
                        flightState ++;
                }else if(flightState == PWRLSASCENDING && block1){
                        if(realAcc=1.0){
                                apogee = baro.UPDATE_ALTITUDE();
                                flightState ++;
                                block1 = false;
                        }
                }else if(flightState == APOGEE && block2) {
                        TVC.MOTOR_EJECTION();
                        pyro.FLAP_DEPLOY();
                        digitalWrite(10,HIGH);
                        delay(50);
                        digitalWrite(10,LOW);
                        delay(50);
                        flightState ++;
                        block2 = false;
                }else if(flightState == DESCENDING){
                        if(Ep.GetPotentialEnergy(baro.UPDATE_ALTITUDE()-baro.LAUNCHALTITUDE, MASS
    ) - E16 == 50, block6){//give of margin of error
                                pyro.DSCENDING_IGNITION();
                                block6 = false;
                                block5 = false;
                        }else if(Ep.GetPotentialEnergy(apogee, MASS)<E16,block5){
                                block5 = false;
                                pyro.DSCENDING_IGNITION();
                        }
                        TVC.DESCENDING();
                }
```

III. Testings and Modeling

A. TVM Static Fire

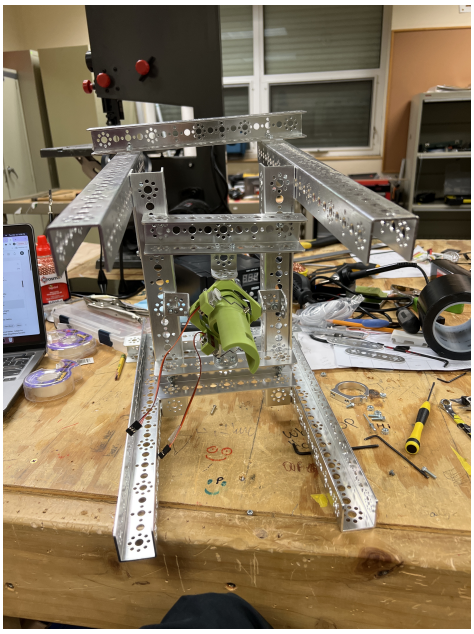
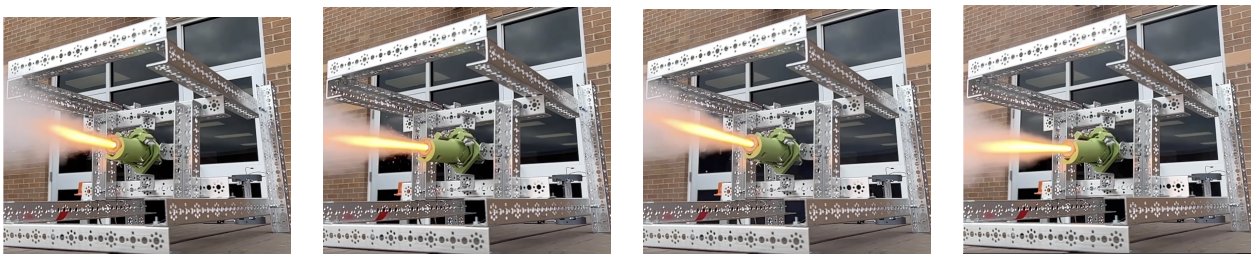


Fig. 4 Assembly of the TVM static fire test stand

As a means to validate the structural rigidity and full-range gimballing capabilities of the TVM under the full thrust of the F15-0, the study assembled a rigid static fire test stand from aluminum C-Channels for mounting the TVM (Fig. 4).

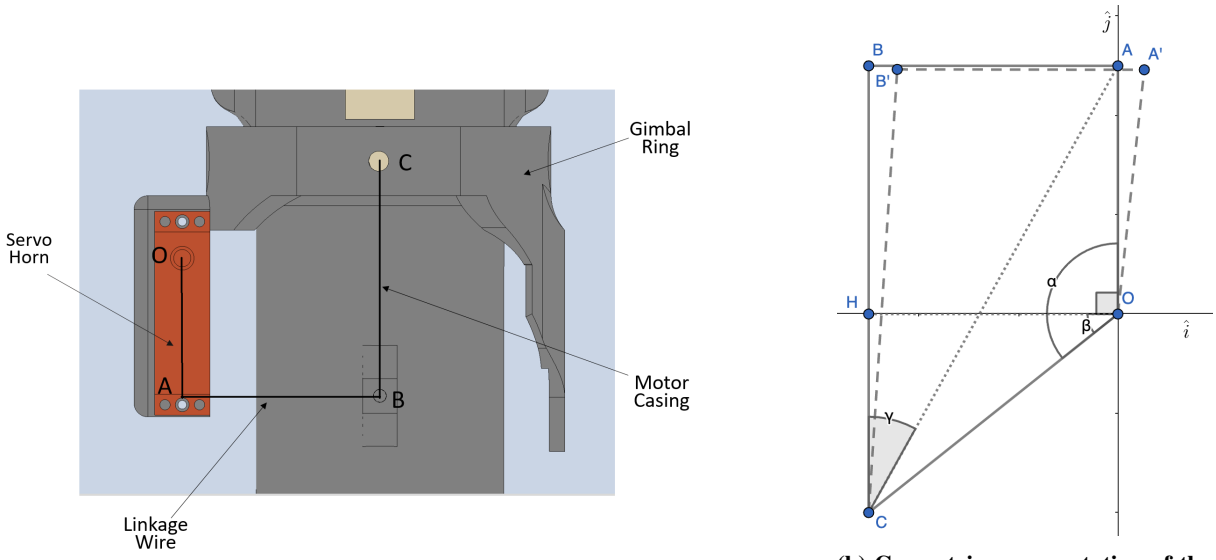
During the static fire test, the TVM is mounted onto the test stand with the servos connecting with the flight computer on the side. The flight computer runs the testing software which pivots the motor in its full $\pm 7^\circ$ range of operation on both axes while the motor is exerting full thrust on the TVM.



(a) $+7^\circ$ **on x-axis** **(b)** -7° **on x-axis** **(c)** $+7^\circ$ **on y-axis** **(d)** -7° **on y-axis**

Fig. 5 TVM pivoting in the within the maximum range of operation during static fire

B. TVM Dynamic Error Correction Model



(a) TVM Linkage With letters

(b) Geometric representation of the same mechanism

Fig. 6 Real and Geometric representation of the 4-bar linkage problem on TVM

One of the concerns identified during the test fire was the inability of the TVM (Thrust Vectoring Mechanism) to achieve precise angular adjustment as commanded by the flight computer. For instance, when the servo's horn pivots 10 degrees, the motor casing pivots less than 10 degrees. Further analysis of this matter determined that the root cause of this discrepancy was a structural design deficiency, resulting in a phenomenon commonly recognized as the "four-bar linkage problem" (Fig. 6). The following mathematical computation and code snippet will derive the solution model to this problem, which will not only apply to this particular study but to all.

```
double alpha = M_PI/2+beta-OMEGA_AO*t;
double AC = oppoLengthLawOfCosine(AO, CO, alpha);
double gamma = oppoAngleLawOfCosine(BC, AC, AB);
double BCO = gamma + oppoAngleLawOfCosine(CO, AC, AO);
double BCB = BCO - atan(AB/HC);
double AOA = OMEGA_AO*t;

double X_B = -BC*sin(BCB)+X_C;
double Y_B = BC*cos(BCB)+Y_C;

double X_A = -sin(OMEGA_AO*t)*AO;
double Y_A = cos(OMEGA_AO*t)*AO;

double BA_i = X_A - X_B;
double BA_j = Y_A - Y_B;

double OA_i = X_A;
double OA_j = Y_A;

double CB_i = X_B-X_C;
double CB_j = Y_B-Y_C;

//cramer's rule
double OMEGA_AB = det({{OA_i*OMEGA_AO, CB_i},{OA_j*OMEGA_AO,CB_j}})/det({{BA_i,CB_i},{BA_j,CB_j
    }});
double OMEGA_BC = det({{BA_i,OA_i*OMEGA_AO},{BA_j,OA_j*OMEGA_AO}})/det({{BA_i,CB_i},{BA_j,CB_j}})
    ;
```

$$\because \beta = \tan^{-1}\left(\frac{HC}{HO}\right) \tag{3}$$

$$\alpha = (90 + \beta) + \omega_{AO} \cdot t \tag{4}$$

$$\therefore AC^2 = AO^2 + OC^2 - 2 \cdot AO \cdot OC \cdot cos(\alpha) \tag{5}$$

$$\cos^{-1}\left(\frac{AC^2 - AO^2 - OC^2}{-2 \cdot AO \cdot OC}\right) = \alpha \tag{6}$$

$$AB^2 = BC^2 + AO^2 + OC^2 - 2 \cdot AO \cdot OC \cdot \cos(\alpha) - 2 \cdot BC \cdot AC \cdot \cos(\gamma) \tag{7}$$

$$\therefore \gamma = \cos^{-1}\left(\frac{AB^2 - BC^2 - AO^2 - OC^2 + 2 \cdot AO \cdot OC \cdot \cos(\alpha)}{-2 \cdot BC \cdot AC}\right) \tag{8}$$

$$\therefore \angle BCO = \gamma + \angle ACO \tag{9}$$

$$\because AO^2 = AC^2 + OC^2 - 2 \cdot AC \cdot OC \cdot \cos(\angle ACO) \tag{10}$$

$$\angle ACO = \cos^{-1}\left(\frac{AO^2 - AC^2 - OC^2}{-2 \cdot AC \cdot OC}\right) \tag{11}$$

$$\therefore \angle BCO = \gamma + cos^{-1}\left(\frac{AO^2 - AC^2 - OC^2}{-2 \cdot AC \cdot OC}\right) \tag{12}$$

$$\because V_{A|B} = V_A - V_B \tag{13}$$

$$\therefore \vec{V_A} = \vec{\omega_{AO}} \times \vec{r_{A|O}} \tag{14}$$

$$\vec{V_A} = OA_{\hat{i}}\omega_{AO}\hat{j} - OA_{\hat{j}}\omega_{AO}\hat{i} \tag{15}$$

$$\therefore \vec{V_B} = \vec{\omega_{BC}} \times \vec{r_{B|C}} \tag{16}$$

$$\vec{V_B} = CB_{\hat{i}} \cdot \omega_{BC}\hat{j} - CB_{\hat{j}} \cdot \omega_{BC}\hat{i} \tag{17}$$

$$\therefore \vec{V_{AB}} = \vec{\omega_{AB}} \times \vec{r_{A|B}} \tag{18}$$

$$\vec{V_{AB}} = BA_{\hat{i}}\omega_{AB}\hat{j} - BA_{\hat{j}}\omega_{AB}\hat{i} \tag{19}$$

$$\therefore \begin{cases} BA_{\hat{i}}\omega_{AB} = OA_{\hat{i}}\omega_{AO} - CB_{\hat{i}}\omega_{BC} \\ BA_{\hat{i}}\omega_{AB} = OA_{\hat{j}}\omega_{AO} - CB_{\hat{j}}\omega_{BC} \end{cases} \tag{20}$$

Applying Cramer's Rule of solving system of first order equations, we can arrive at the matrix:

$$\omega_{AB} = \frac{det\begin{bmatrix} CB_i & OA_i \cdot \omega_{AO} \\ CB_j & OA_j \cdot \omega_{AO} \end{bmatrix}}{det\begin{bmatrix} BA_i & CB_i \\ BA_j & CB_j \end{bmatrix}} \tag{21}$$

$$\omega_{BC} = \frac{det\begin{bmatrix} BA_i & OA_i \cdot \omega_{AO} \\ BA_j & OA_j \cdot \omega_{AO} \end{bmatrix}}{det\begin{bmatrix} BA_i & CB_i \\ BA_j & CB_j \end{bmatrix}} \tag{22}$$

The result showcase the angular speed of the TVM and relative to the angular speed of the servo. In this case, AB is the linkage arm between servo and TVM, while BC is the TVM and AO is the servo arm.

IV. Results and Discussion

A. TVC Response

During testing, vehicle exemplifies signs of active error correction during the powered ascent of the flight by changing the direction of thrust(Fig. 7).

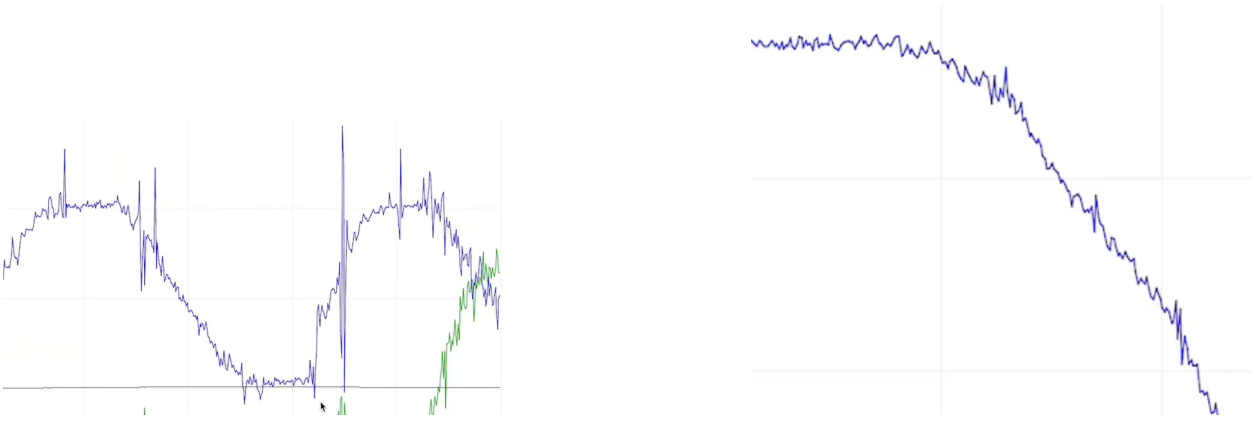
(a) Liftoff **(b) TVM Correction** **(c) TVM Correction** **(d) TVM Over-correction**

Fig. 7 TMV reacting to the deviation detected by the IMU sensor onboard, last picture shows an over-correction of TVM

While the flight computer effectively detected positional deviations in the rocket's trajectory through the application of PID filtering, observations indicate that the TVM system may be exhibiting excessive responsiveness, potentially leading to over-correction and unnecessary oscillation of the vehicle's heading (Fig. 7d).

To address this concern, adjustments will be made to the proportional (P) and derivative (D) coefficients within the control software. Fine-tuning these coefficients can serve to reduce the sensitivity of the TVM system, promoting more stable and controlled flight dynamics.

B. Sensor Noise Reduction



(a) Data before applying Kalman Filter **(b) Data After applying Kalman Filter**

Fig. 8 Barometric sensor output before and after Kalman Filtering

Following the implementation of the Kalman filter for data filtering, a notable reduction in the noise within the output signals from both the Inertial Measurement Unit (IMU) and barometer has been achieved. Without extensive filtering, the unfiltered input signals exhibit spikes, thereby inducing abrupt and undesirable perturbations in the flight dynamics.

V. Conclusion

This study verified that Thrust Vector Control Theory is applicable on a smaller scale by using a hobby-size rocket as a testing platform. However, there are still limitations and room for improvement. For example, PID gains should still be tuned so that the vehicle doesn't overcorrect. Additionally, active controlled landing is still yet to be proven to be executable due to mission abort of the flight testing procedure caused by unexpected yaw-pitch gain during the flight. However, it is still evident that the thrust vectoring logic and methodology show effect on a smaller scale. Furthermore, the dynamic error correction model discussed in Section III.B can act as a helpful reference for any other gimbaling rocketry.

The study aims to introduce a physical throttling mechanism in addition to the existing throttling algorithm. This strategic enhancement will contribute to a more comprehensive and adaptable control system, further enhancing the precision and reliability of our vehicle's performance.

Acknowledgments

The author would like to acknowledge the generous funding from the ISM Parents Connection. The contents are solely the responsibility of the author and do not necessarily represent the official views of the funder. Additionally, the author would like to acknowledge Manny Peterson, an open-source developer, for his assistance with the study.

References

[1] YAĞMUR, H., ŞEN, S., BAYAR, C., and SERBEST, K., "Design of A 3-DOF Thrust Control System for Rocket Engines," *Journal of Smart Systems Research (*, Vol. 30, 2022, p. 30–48.

[2] Sutton, G. P., and Biblarz, O., "Rocket Propulsion Elements," 2016.

[3] Rowberg, J., *CJROWBERG/i2cdevlib: I2C Device Library Collection for AVR/Arduino or Other c++-Based Mcus.*, GitHub, https://github.com/jrowberg/i2cdevlib, 2021.

[4] Peterson, M., *heliosproj/HeliOS: A community delivered, open source embedded operating system project.*, GitHub, https://github.com/heliosproj/HeliOS, 2021.

[5] Sene, D., *denyssene/SimpleKalmanFilter: A basic implementation of Kalman Filter for single variable models.*, GitHub, https://github.com/denyssene/SimpleKalmanFilter, 2021.

[6] Siouris, G. M., "Missile Guidance and Control Systems," 2003.