MICRO-GRAVITY ROBOTS

Designing robots that can move in micro- and zero gravity environments

Parth Chaudhry

Table of Contents

1 ABSTRACT

There is increasing activity in space on a lot of fronts: launch, space stations, planetary rovers, satellite launch/ maintenance, asteroid prospecting, tourism etc. There is a lot of activity in the micro-gravity environment of Low Earth Orbit (LEO) and soon cislunar space: ISS, Hubble telescope, satellite maintenance, and other planned space stations. Today a lot of tasks in this micro-gravity environment, especially on the ISS, are done by humans (e.g. ISS construction, ISS battery replacement or Hubble telescope repair etc.). These tasks are ideally suited for robots, but current Earth based robots don't work in space because of lack of gravity. There is need for free flying robots that can operate in the microgravity environment of space and perform needed tasks.

There are several challenges with getting robots to work in the microgravity environment of space:

- 3D motion (translation and rotation): traditional methods to move like wheels, walking, crawling or sliding don't work in microgravity.
- Maintaining position and attitude due to conservation of linear and angular momentum
- 3D route calculation
- 3D map generation etc. to mention a few.

This paper proposes a robot design that can effectively move, orient and maintain its position in microgravity. The robot has a cube structure for symmetry. The robot uses cold gas propulsion to control both translation and rotation motions. It has twelve cold gas propulsion nozzles on four out of six faces of the robot. The paper provides mathematical equations relating distance traveled and angle of rotation to the needed fuel (an inert gas, either N_2 or CO_2). The paper includes equations to calculate 3D routes that are either speed or fuel optimized. The paper also discusses methods to maintain robot position and attitude when the robot expands its appendages or turns, which is a challenge due to conservation of linear and angular momentum.

Additional research work is needed to design autonomous control and operation of the robot. 3D mapping is another area which needs work for the robot to be effective. These topics are not addressed in this paper, but are areas of research for the future.

2 INTRODUCTION

There is a lot of interest and activity in Space these days. Several private companies like SpaceX and Rocket Lab are able to launch payloads to orbit around Earth. Many more companies like Relativity Space, Virgin Galactic, and Blue Origin are building launch capabilities. There is a lot of interest in Space tourism and building space stations in Low Earth Orbit (LEO) also. In the coming years there is likelihood of a lot of space station construction e.g. private space stations for tourism and the Lunar gateway for NASA's Artemis project.

2.1 **NEED**: ROBOTS THAT WORK IN MICRO-G

Most people are surprised to know that there are no robots in space and most work is done by humans. For example, the ISS was assembled by humans in 227 spacewalks over 1400 hours. The space station has robotic arms, but not standalone robots. Most experiments and maintenance work on the ISS is also done by humans. In 2019, NASA introduced Astrobees, robots that can work inside the ISS's atmosphere at 1 atm pressure, but won't work in the vacuum. It is a start, but what's needed are robots that can work in the vacuum and micro-gravity environment of space. Such robots can be used for a variety of tasks, e.g.

1) Perform maintenance and inspection tasks outside and inside the ISS or future space stations
2) Perform experiments on the ISS or future space stations
3) Build and manage future space stations or Unmanned orbital outposts
4) Prospect asteroids (microgravity environment)
5) Mine asteroids (microgravity environment)

The problem is that robots designed for operating on Earth won't work in the micro-g environment of space and a different approach is needed. This paper describes an approach to build robots that can work in microgravity.

3 CHALLENGES WITH MICRO-G ROBOTS

Robots designed for operating on Earth won't work in space due to many challenges related to microgravity:

1) Motion: Translation and Rotation
2) Maintaining Position and Orientation
3) 3-D route calculation
4) 3-D map generation

The following challenges are not unique to micro-g robots, but need to be addressed:

1) **Autonomous operation**: more than likely, the robots need to be able to operate autonomously, as there may be a delay in reaching the robots in space. Below are some roundtrip time durations for a signal from Earth:
    - **LEO**: LEO is probably fine for remote operation: round trip: 1/100th of a second. Also, there may be a need for intermediate satellites for routing the commands to the robots.
    - **Lunar Gateway**: one way trip is 1.255 seconds, round trip is 2.6 seconds.
    - **Asteroids**: It may take 10+ minutes for a roundtrip communication with a robot on an asteroid.
2) **Vision and image recognition**: The robot needs to have the ability to recognize images and act on them. This should be as much automated as possible.

Lastly, a big challenge is simulating microgravity to test the robot on Earth.

In this document the term micro-g refers to environments with gravitational acceleration in the range of 10^{-6} m/s^2.

3.1 ROBOT MOTION IN SPACE: 6 DEGREES OF FREEDOM

Robot motion in space needs 6 degrees of freedom as shown in the picture below.

1) Translation: linear motion across the 3-axes: X, Y and Z.
2) Rotation: rotation motion along the 3-axes: X, Y and Z, also known as pitch, yaw and roll.

There are several challenges with both of these:

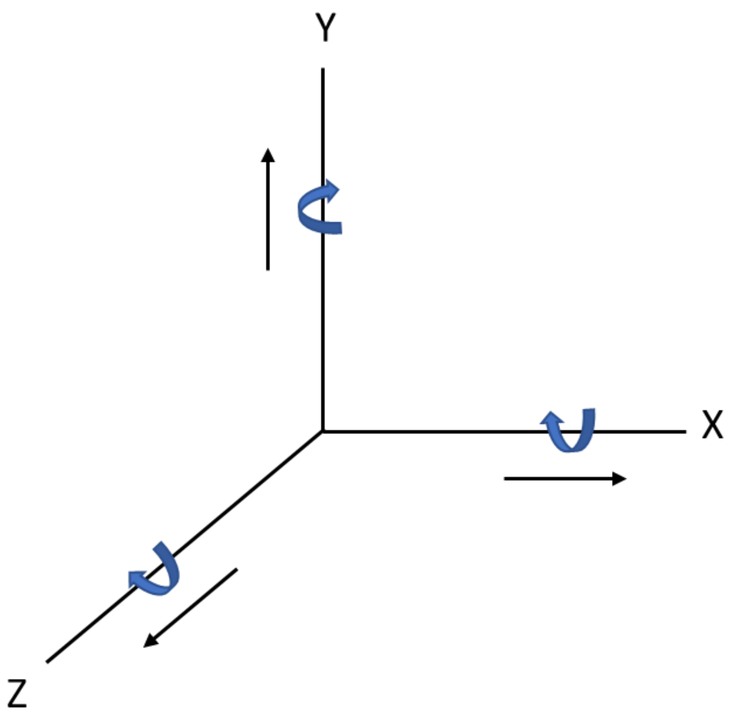


Figure 1: Robot motion in 6 degrees of freedom

3.1.1 TRANSLATION MOTION

Traditional robots on Earth and even rovers designed for the Moon and Mars won't work in space as their method of movement and braking will not work in micro-g due to absence of gravity.

1) **Initiating motion**: Traditional modes of robot motion include wheels and moving along a surface (walking, slide or hopping). All of these modes need gravity, friction and the ability to push off against another object, which are not present in the micro-g environment of space.
2) **Stopping motion**: The traditional braking methods which rely on friction don't work in micro-g environment. Since there is no gravity or friction, the robot can't come to a stop on its own either.

3.1.2 ROTATION/CHANGING ORIENTATION

Similar problems exist with rotation motion in micro-g:

1) **Initiating rotation**: Traditional robots use gravity or a push against another surface to apply torque for rotation/changing orientation. In microgravity those methods don't work.
2) **Stopping rotation**: Once the robot initiates the rotation in micro-g, the rotation doesn't stop automatically, it needs another torque in the opposite direction to stop the turning.

3.2 MAINTAINING POSITION AND ORIENTATION

In micro-g, the robot is simply "floating" and is not attached to any surface. This causes an additional challenge: maintaining the robot's position and orientation if the robot has to perform a task. Simple tasks like extending a robot's arm, or rotating a part of the robot will cause the robot's position and orientation to change.

3.2.1 MAINTAINING POSITION

Any operation by the robot which changes the center of mass of the robot, will cause the robot's position to change to ensure that the center of mass of the robot doesn't move from its original position. This is because internal forces can't shift the center of mass of an object. For example:

- Extending/retracting part of a robot: e.g. extending one arm, or the torso of the robot.
- Lifting/dropping something: this would also shift the center of mass.
- Changing robot shape

3.2.2 MAINTAINING ORIENTATION

If part of a robot rotates, for example a robot hand, or neck or another part, the robot orientation would change due to conservation of angular momentum.

Robots in micro-g need to be able to detect and handle changes in position and orientation as a result of these actions.

3.3 3-D ROUTE CALCULATION

Other than drones or robots that can fly, most Earth based robots/rovers move in 2 dimensions, because they're moving along a surface using gravity. Even drones perform most of their motion in 2 dimensions except for start and stop where elevation change is required.

In space, robots would need to routinely move in 3 dimensions and route calculation would need to be done in 3 dimensions. For example, for a robot that is servicing the solar panels on ISS a route calculation might involve calculating the optimum route from the airlock to the solar panels in 3 dimensions.

Another example is for the microgravity environment of an Asteroid, where the gravity could be 1/100,000th of Earth. An asteroid's surface is extremely uneven and rocky. A prospecting robot would need to be able to calculate the optimal route in all 3 dimensions.

3.4 3-D MAP GENERATION

For robots that need to operate autonomously, maps are important to plan routes from one place to another. A robot designed to work in the micro-g environment of space would need a map in 3 dimensions. Similarly, an asteroid prospecting robot would need to generate 3-D maps so that any mining robots can use those maps to navigate optimally.

3.5 AUTONOMOUS OPERATION

Other than for Low Earth Orbit, robots in space may need to operate mostly autonomously because the delay in getting the signal to the robot would make real time control impossible. Here are round trip times for a signal to travel to a few locations in space from Earth:

- ISS: 2.6×10^{-3} seconds
- Lunar Gateway: 2.6 seconds
- Asteroid Ryugu: 33.3 minutes (2,000 seconds)

It's clear that other than ISS (400 km altitude) or other LEO targets, robots in space would need to operate mostly autonomously. Even on ISS, for the robot to be really useful, it may be better to design robots that can receive commands from Earth or astronauts on the ISS, but can then perform those commands autonomously.

3.6 COMPUTER VISION, IMAGE RECOGNITION AND OTHER SENSORS

Robots in space would need to have computer vision and ability to recognize images and take actions based on them. This would be an essential component of autonomous operation. Additionally, robots would need have different types of sensors, e.g. LIDAR, gyroscope, spectrometer, temperature sensor etc.

4 MICRO-G ROBOT DESIGN

There are several factors that need to be considered for the robot design, some are unique to micro-g robots, while others are common to all robot designs:

1) **Propulsion system**
2) **Sensors**
3) **Control**
4) **Communication**
5) **Map/route calculation**
6) **Appendages**

This paper primarily addresses robot propulsion and movement which is unique to micro-g robots.

4.1 PROPULSION SYSTEM

The robot propulsion system is one of the most important factors for micro-g robots as they need a completely different method for translation and rotation/attitude vs robots that operate in regular gravity.

Following are some design considerations for the robot propulsion system:

- **Propulsion mechanism for translation**: For a robot to move in space, it needs a propulsion mechanism. The ideal method is cold gas propulsion. Using a pressurized gas like N_2 or CO_2 for robot propulsion is both cost effective and safe. The thrust generated by cold gas propulsion can easily be calculated using the Tsiolkovsky Rocket Equation.
- **Rotation mechanism**: While there are many mechanisms to control rotation or attitude in space e.g. reaction wheels, cold gas propulsion can be effective to control the orientation, especially if the robot already uses it for propulsion. The design proposed in this paper uses the same cold gas thrusters for both translation and rotation.
- **Fuel**: The ideal fuel for cold gas propulsion is an inert gas that is inexpensive, can be easily transported to space, and is safe to use both inside and outside the space station. Both Nitrogen and CO_2 are ideal candidates as both are inert, already present on the ISS and are reasonably safe to use inside the atmosphere of the ISS (in limited quantity).
- **Symmetrical shape**: for ease of determination of center of mass, moment of inertia and torque.
- **Cost**: use off-the-shelf or 3-D printed components

4.1.1 ROBOT PROPULSION SYSTEM DESIGN

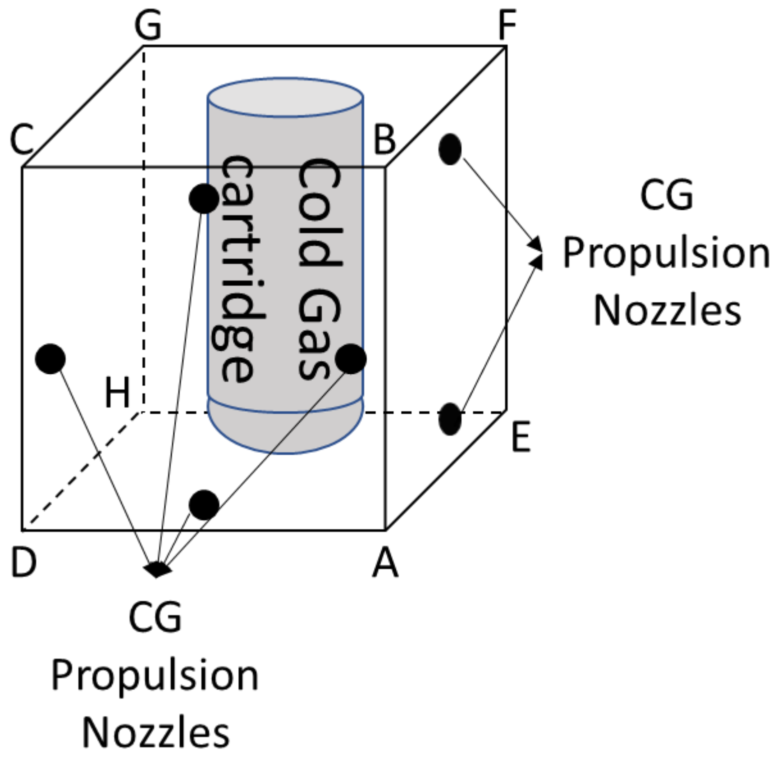


Figure 2: Robot propulsion system overview

The picture shows a simplified view of the robot which focuses on the main components of the robot's Cold Gas (CG) Propulsion system. The system has the following main components:

- **Chassis**: The robot chassis is shaped like a cube. This allows for a symmetrical shape and uniform distribution of mass. This also allows for easier calculation of moment of inertia and ability to calculate required torque to change the robot orientation or attitude. The chassis has the following specifications:
  - **Side of the cube**: s meters
  - **Mass of the cube**: m Kg
- **12 Nozzles**: The robot has 12 nozzles connected to the cold gas propulsion tank to allow for 6 degrees of freedom to the robot. The cube has 8 vertices ABCDEFGH and 6 faces. The distribution of nozzles is as follows:
  - **Face ABCD:** 4 nozzles
  - **Face EFGH:** 4 nozzles
  - **Face ABFE:** 2 nozzles
  - **Face CDHG:** 2 nozzles
  - **Face AEHD:** 0 nozzles
  - **Face BFGC:** 0 nozzles

- **Removable Cold Gas Tank**: The robot is designed so that the cold gas tank can be removed and replaced with another tank for easy refueling. The tank is designed so that the center of mass of the robot remains as close to the center of the cube as possible.

4.1.1.1 Cold Gas Propulsion System Characteristics

A cold gas propulsion system generates thrust by expelling gas particles at a high speed from a narrow opening to generate thrust in the opposite direction. The thrust generated by a cold gas system can be calculated using the following Equation:

Thrust $F = \dot{m} \times V_e + (P_e - P_o) \times A_e$, where

- $\dot{m}$ = mass flow rate
- V_e = exhaust velocity
- P_e = Pressure in the exhaust area of the regulator
- P_0 = ambient pressure
- A_e = Exhaust area

4.1.1.1.1 MASS FLOW RATE

The Mass Flow Rate is denoted by the symbol $\dot{m}$, and is a measure of the rate at which matter leaves the system. It is calculated using the Choked Flow Equation.

$$\dot{m} = C_d A \sqrt{\gamma \rho_0 P_0 \left(\frac{2}{\gamma+1}\right)^{\frac{\gamma+1}{\gamma-1}}}$$

- C_d = Discharge coefficient
- A = area of the orifice the propellant comes out of.
- ρ_0 = real gas density of the gas at pressure P_0 and Temperature T_0
- P_0 = pressure of the gas in the tank
- γ = specific heat ratio of the gas

4.1.1.1.2 EXHAUST VELOCITY

The Exhaust Velocity is a measure of the speed of the particles leaving the orifice.

$V_e = M_e\sqrt{k * R * Te}$ (Rocket Thrust Summary, n.d.)

M_e = Mach Number

k = Specific Heat Ratio of the gas

R = Specific Gas Constant for the gas = Universal gas constant divided by molar mass of the gas.

T_e = Gas temperature in exhaust region

4.2 ROBOT SENSORS

There are two types of sensors:

a) **Not specific to Robot Application**: These types of sensors are needed for the operation of base robot itself. The robot will need the following types of sensors:
   - **Distance**: The robot needs to be able to measure its distance from various sources: the destination, obstacles etc. The robot will have distance sensors on each of the 6 faces of the cube chassis.
   - **Orientation**: The robot will need to have a gyroscopic sensor to measure its orientation.
   - **Vision**: The robot should ideally have a camera on each of the 6 faces of the cube chassis.

b) **Specific to robot application**: Depending on the purpose of the robot additional application specific sensors may be needed, e.g. a IR spectrometer for a robot that will be used for asteroid prospecting. This paper doesn't address those sensors.

4.3 ROBOT CONTROL

The robot will need a control mechanism to control overall robot movement, orientations and any other tasks the robot may need to do. The design proposed in this paper uses an Arduino based controller.

4.4 COMMUNICATION

The communication method for the robot will depend on the distance of the robot from its operator. For example, for robots operating within a confined area like in and around the ISS WiFi or Bluetooth may suffice, but for robots that need to prospect an asteroid, specialized long-range communication will be needed.

4.5 MAP/ROUTE CALCULATION

The robot will need to operate in 2 modes:

- **Mapped**: In the mapped mode, the robot has access to the map of the area in which the robot is operating and the robot is able to use it to calculate routes of destinations which may not be visible to the robot via a line of sight. All maps would need to be 3D maps as the robot needs to operate in 3 dimensions.
- **Unmapped**: In this mode, the robot has 2 options:
  - **Line of sight operation**: In this mode, the robot can move from one point to another as long as it has a line of sight to the destination and is able to calculate the distance vector to it.
  - **Construct a map**: In this mode, the robot may have to construct the map of the area first and then operate in the mapped mode.

4.6 APPENDAGES

Depending on the task being performed, the robot may need one or more appendages e.g. arms, legs etc. This paper doesn't focus on which appendages the robot will use, but rather highlights the following guidelines that will apply to all appendages:

- **Symmetrical design and attachment**
- **Predictable center of mass**
- **Angular speed feedback**

5 ROBOT MOVEMENT

This section describes how the robot moves (rotation and translation) using a combination of the 12 nozzles on the robot.

5.1 ROTATION/ATTITUDE

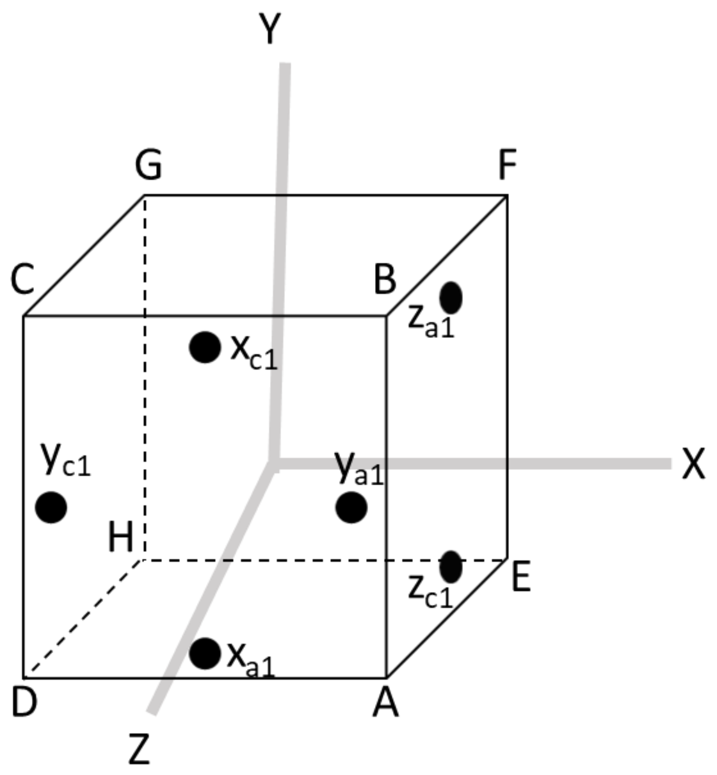


Figure 3: Robot chassis around robot centered coordinate system

This figure shows the robot chassis around a coordinate system whose origin is the center of the cube. Six out of twelve nozzles are shown. The nozzles are labeled to describe the axis along which they cause rotation.

5.1.1 ROTATION MECHANISM:

5.1.1.1 Paired nozzles

For rotation, the nozzles are always used in pairs. **A nozzle's pair is located on the opposite face of the cube, diagonally opposite to it**. For example, the nozzle x_{c1} along the edge BC would be used with the nozzle along the edge EH (x_{c2}, not shown). This generates a torque which causes the robot to rotate clockwise along the x-axis. To stop the rotation the opposite nozzles are used, e.g. if x_{c1} and x_{c2} were used to initiate rotation, x_{a1} and x_{a2} are used to stop the rotation.

Naming convention:

- The first letter of the nozzle (x, y or z) indicates the axis which the robot rotates along.
- The letter in the subscript indicates whether the rotation is either clockwise or counter-clockwise.
- The number in the subscript indicates the number of the nozzle (1 or 2).

Torque

The torque τ is given by the formula = n x F x $\frac{s}{2}$

- F = thrust generated by each nozzle (calculated previously)
- n = number of thrusters. Since we're using 2 thrusters, n = 2
- s = side length of the cube

5.1.1.2 Rotation Process

Rotation is a 3-step process:

1. **Start rotation**: by firing the appropriate pair of thrusters for a duration t_b.
2. **Coast**: After time t_b, the robot is turning and will continue to turn as in space there is no torque to stop it from turning. The coast time denoted by t_c.
3. **Stop rotation**: by firing the opposite pair of thrusters for the same duration as in step 1, i.e. t_b.

Example (anti-clockwise Rotation along Z axis)

1. **Start rotation**: Fire thrusters z_{a1} and z_{a2}
2. **Coast**: Let the rotation of the robot continue for as long as desired.
3. **Stop rotation**: Fire thrusters z_{c1} and z_{c2}

By controlling times t_b and t_c, the robot rotation can be optimized for either speed or fuel efficiency.

5.1.1.3 Calculating angle of rotation and duration

The angle θ of the robot rotation along a given axis is given by the following equation:

θ = $\frac{nFL}{I} t_b^2 + \frac{nFL}{I} t_b t_c$

n = Number of thrusters

F = Thrust generated by each nozzle

L = Distance to center of mass of robot

t_b = Duration of the burn

t_c = Duration of the coast

I = Moment of Inertia, cube = $\frac{1}{6}(ms^6)$, where s = side length of the cube, m = the mass.

5.1.1.4 Estimating the propellant used for rotation

We can estimate the propellant used for rotation by using the specific impulse of the gas we intend to use as the propellant.

Thrust F = $\dot{m} \times V_e$

$V_e = I_{sp} \times g$

Rearranging, $\dot{m} = \frac{F}{g\, I_{sp}}$ for 1 thruster

Total propellant consumed = 2 x t_b x $\dot{m}$

= $\frac{2t_bF}{g\, I_{sp}}$, for 1 thruster

5.2 TRANSLATION

Translation in space is moving from Current position to a destination. There are two types of translation:

1) **Single step, or Point to Point translation**: the destination can be reached in a single 3D straight line. This means there are no obstructions or any other reasons to not go in a straight line.
2) **Multi-step**: the path to the destination is not a single step, instead there are multiple single steps.

This section will focus on single step translation where the path between the source and destination is a single 3D straight line. The more complicated case of calculating a route via multiple intermediate points will be tackled separately using single step translation as a building block.

Point to point translation has the following steps:

1. Selecting a reference coordinate system
2. Calculate the distance vector from the robot's current position to the destination
3. Orient the robot along the distance vector between the two points

4. Determine the speed mode: fastest or most propellant efficient
5. Calculate thruster fire time t_b and coast time t_c based on the distance to be traveled.
6. Fire thruster for time t_b to initiate motion
7. Coast for time t_c
8. Fire thrusters for time t_b to brake

5.2.1 SELECTING A REFERENCE COORDINATE SYSTEM

There are a number of coordinate systems that can be used, but the following two are the most useful:

1) **Robot centered coordinate system**: a coordinate system with the origin at the center of the robot cube
2) **Space Body centered coordinate system**: a coordinate system with the origin on the space body (station or another microgravity body like an asteroid) where the robot is deployed.

5.2.1.1 Robot centered coordinate system

The advantage of using a robot centered coordinate system is that calculation of distance vector between the robot's current position and the destination becomes straightforward. It is simply the coordinates of the destination on the robot centered coordinate system. Two possible ways to calculate the vector distance are:

a) Using robot sensor instruments like LIDAR, calculate the distance and angle to the destination, using which the coordinates with respect to the robot can be calculated.
b) Given the coordinates of the robot and the destination in the Space Body coordinate system, we can calculate the coordinates of the destination in the robot centered coordinate system.

5.2.1.2 Space Body centered coordinate system

Using this coordinate system can be useful in the following situations:

a) The robot doesn't have an ability to determine the vector distance to the destination
b) The robot doesn't have a line of sight to the destination
c) The route to the destination is not one step, but rather involves multiple intermediate steps
d) The robot needs to follow a prescribed route to the destination

In these situations, the robot can be provided the coordinates of the final destination or the intermediate steps in the SpaceBody coordinate system. As long as the robot is aware of its starting coordinates in the SpaceBody coordinate system, it can determine the vector distance to the final destination.

5.2.2 CALCULATE THE DISTANCE VECTOR(S) TO THE DESTINATION

There are two scenarios to consider:

a) The path from the Robot to the destination can be represented by a single distance vector
b) The path from the Robot to the destination consists of a series of steps, each of which can be represented by a single distance vector

This section will only look at scenario a). Scenario b) is a combination scenario requiring both translation and rotation and will be looked at separately.

Once the robot has selected the appropriate coordinate system and it is aware of the coordinates of itself and the destination, it can calculate the distance vector to the destination using simple vector mathematics.

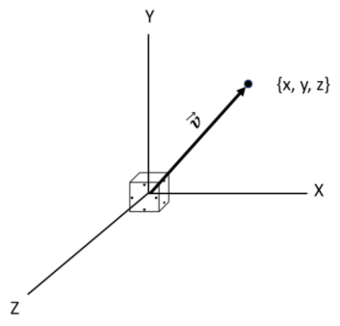


Figure 4: Destination coordinates in robot centered coordinate system

For example, in the picture above, the coordinates of the destination vector ($\vec{v}$) in the robot centered coordinate systems are {x, y, z}. The magnitude of the distance vector is given by $||v||$ = $\sqrt{x^2 + y^2 + z^2}$ and the direction unit vector is given by $\hat{v}$ = $\{\frac{x}{||v||}, \frac{y}{||v||}, \frac{z}{||v||}\}$.

5.2.3 ORIENT THE ROBOT ALONG THE DIRECTION VECTOR

Using the direction vector calculated in the previous step, we can calculate the rotation required along each axis to orient the robot's thrusters along that vector.

Let's say the robot needs to rotate along each of the axes by {α, β, γ} to orient along the destination vector. The relationship of these angles with the unit vector is given by the following:

- cos(α) = $\frac{x}{||v||}$
- cos(β) = $\frac{y}{||v||}$
- cos(γ) = $\frac{z}{||v||}$

Once we know the angles of rotation along the axes, we can rotate the robot using the rotation mechanism described before.

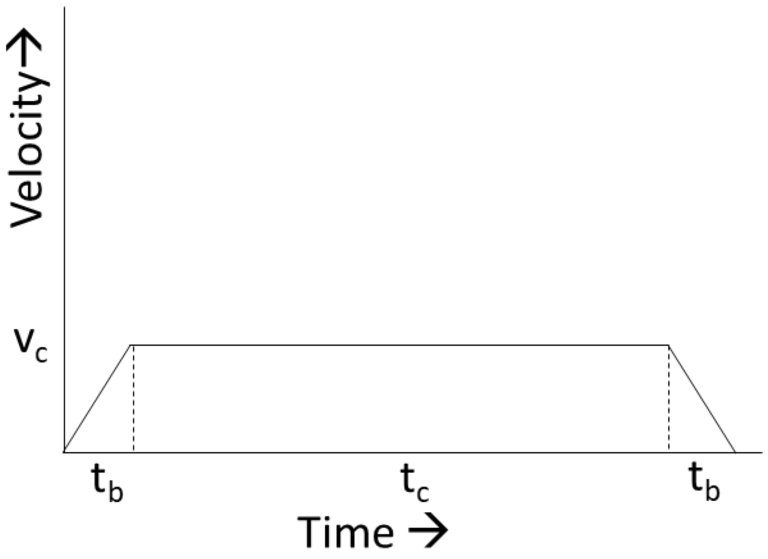
5.2.4 DETERMINE SPEED MODE

The robot can reach its destination in two modes:

1) **Fuel-efficient**: the robot fires its thrusters for a very short amount of time, then coasts for the vast majority of the distance before braking.
2) **Speed-optimized**: The main difference is that the robot doesn't coast at all. For the first half of the travel, the robot fires its thrusters in the direction of the motion and then in the opposite direction for the 2nd half.

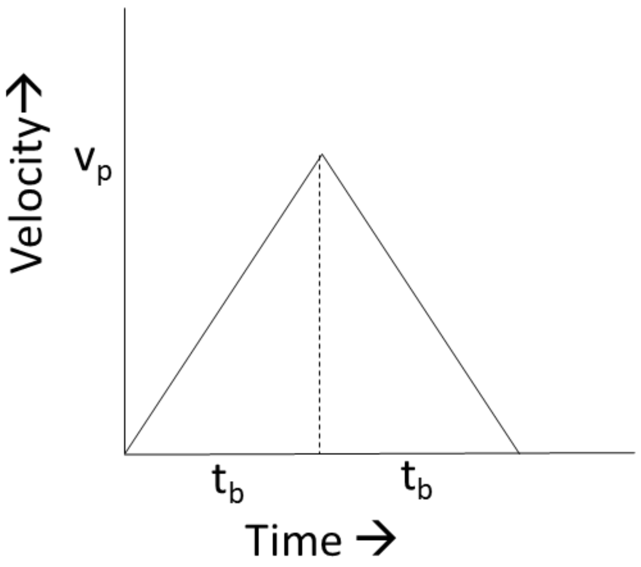
5.2.4.1 Fuel-efficient mode

The picture below shows the velocity of the robot in the fuel-efficient mode. The robot thrusters fire for duration t_b, causing the robot to achieve coast velocity v_c. The robot coasts for duration t_c and then brakes for time t_b to come to a stop at the destination.



5.2.4.2 Speed optimized mode

The picture below shows the velocity profile of the robot in the speed optimized mode. The main difference is that the robot doesn't coast at all, and coast time t_c = 0



5.2.5 CALCULATING COAST AND BURN TIMES

The time t_b can be specified or it can be the minimum duration for which the robot thrusters must fire. If we know the distance to be traveled and the duration t_b, we can calculate the coast time t_c.

The following are known, or we know how to calculate them as shown in previous sections:

- F = thrust generated by each nozzle
- n = number of thrusters used
- m = mass of the robot
- d = total distance traveled by the robot = $2d_b + d_c$

Based on above, acceleration 'a' experienced by the robot = $\frac{nF}{m}$

Robot coast velocity $v_c = v_0 + at$, since $v_0 = 0$ and $t = t_b$, we can write:

$v_c = at_b$

$d_b = \frac{1}{2}at_b^2$

$d_c = v_c \times t_c$, if we rearrange, we can write: $t_c = \frac{d_c}{v_c}$

if we substitute $d_c = d - 2d_b$, we get:

$$t_c = \frac{d-2d_b}{v_c} = \frac{d}{v_c} - \frac{2d_b}{v_c} = \frac{d}{at_b} - \frac{at_b^2}{at_b} = \frac{d}{at_b} - t_b$$

Coast time $t_c = \frac{d}{at_b} - t_b$

5.2.6 FIRE THRUSTERS TO INITIATE MOTION

The robot fires thrusters on the face ABCD as shown in Figure 3: Robot chassis around robot centered coordinate systemFigure 3 on page 13, for duration t_b. The robot can fire either just 2 thrusters or all 4 thrusters on the face ABCD.

5.2.7 COAST

After the thrusters are fired, the robot coasts for time t_c, the formula for which is given previously. During the coast period, the robot needs to continuously monitor its distance and orientation to make sure they are as predicted by the model. If the actual distance and orientation is different than what's predicted then the robot needs to take corrective action. This is handled by the control module of the robot.

5.2.8 FIRE THRUSTERS TO BRAKE

The robot fires thrusters on the face EFGH for duration tb to come to a stop. The number of thrusters fired should match the number of thrusters fired to initiate motion.

5.2.9 ADDITIONAL CONSIDERATIONS

Note: In reality, a couple of factors will need to be taken into account:

- **Safety buffer**: The robot will need to factor in a safety buffer in the amount of distance traveled to prevent collisions. This may mean that the robot will need to stop at a distance less than the actual distance.
- **Distance sensing**: The robot will need to use sensors to continuously monitor the distance from the destination to make sure the robot is moving at the predicted velocity.
- **Minimum maneuver duration:** Due to practical constraints with cold gas propulsion systems, the robot will have a minimum amount for the following maneuvers:
    - **Linear distance traveled:** The robot will have a minimum distance it must always travel every time the thrusters are fired. Let's say this distance is x meters. This means that if the robot is closer than x meters from the destination, it may have to first go backwards otherwise there will be a collision.
    - **Angle of rotation**: Similar to above, the robot also has a minimum angle of rotation, let's say that angle is θ. If the robot needs to rotate an angle ϕ less than θ, it can just turn $2\pi \pm \phi$.

6 MAINTAINING POSITION AND ORIENTATION

In micro-g, the robot is simply "floating" and is not attached to any surface. Therefore, maintaining the robot's position and orientation can be a challenge if the robot has to perform a task. Simple tasks like extending a robot's arm, or rotating a part of the robot will cause the robot's position and orientation to change.

6.1 MAINTAINING POSITION

Any operation by the robot which changes the center of mass of the robot, will cause the robot's position to change to ensure that the center of mass of the robot doesn't move from its original position. This is because internal forces can't shift the center of mass of an object. For example:

- Extending/retracting part of a robot: e.g. extending one arm, or the torso of the robot.
- Lifting/dropping something: this would also shift the center of mass.
- Changing robot shape

The robot can adopt the following strategies to reach the desired position:

a) **Stop short of destination**: pre-calculate the impact of the expected shift of the center of mass before the final maneuver and stop short of the destination by that amount.
b) **Get back in position**: Use the sensors to determine the shift in the center of mass from the desired position and use robot movement techniques described in the previous section to get back in position.

6.1.1 STOP SHORT OF DESTINATION

If the robot is aware of the distance vector by which its center of mass shifts when an appendage is expanded or contracted, it can stop short of the destination by that amount. For example, if the robot's center of mass shifts by a vector Δr_{com}, and the distance to the destination is represented by r, the distance the robot should move should be $r - \Delta r_{com}$.

This approach requires that the robot designers are aware of how much the center of mass shifts for expansion/contraction of each appendage.

6.1.2 GET BACK IN POSITION

An alternative approach is for the robot to use distance sensors to determine how much the center of mass has shifted by using the distance sensors. The robot can then move using the translation motion described earlier.

6.2 MAINTAINING ORIENTATION

If part of a robot rotates, for example a robot hand, or neck or another part, the robot orientation would change due to conservation of angular momentum. Similar to for maintaining linear position, the robot can use the following strategies:

Robots in micro-g need to be able to detect and handle changes in position and orientation as a result of these actions.