Analysis of *Terrain-Adaptive Wheel Speed Control on the Curiosity Mars Rover: Algorithm and Flight Results*

Robert Vandemark Diane Ngo

2020
November

Contents

Abstract

Throughout recent decades, the topic of space exploration is desirable towards scientists and researchers. Mars is the nearest planet that is possible to explore in great detail. There are many challenges pertaining to exploring unknown territory, through long distance, and difficulty of remotely operating robots. NASAs approach to exploring and researching Mars was through the Sample Return Rover, created in 1997 by the JPL Jet Propulsion Laboratory. The scope of this project focuses on the Mars Rover, its wheel-slip detection (traction control), and path-planning towards a desired object.

Introduction

The Mars Rover is a Sample Return Rover designed by the JPL NASA Laboratory. It consists of a chassis with a central revolute joint on both the left and right sides of the robot. The central revolute joint is connected to two links (on each side), for the front and rear wheels. Each respective link is connected to another link downward that is connected to a wheel. The front links are revolute so that the robot can turn left and right. Note that the original paper is based off of the Curiosity Rover, which has six wheels. The focus on this paper will be on the Sample Return Rover, which has four wheels.

Primitive designs of this robot had a large amount of damage to the rovers wheels. This wheel damage reduced the longevity of the Mars Rover mission by a great amount. NASA had to counteract this damage through researching the cause of the wheel damage. The rover did not properly avoid terrain obstacles nor did it deal with traction loss. The goal for this project is to recreate the traction control algorithm and simulate the results. The algorithm for the traction control is a velocity-based algorithm. One wheel on the rover will be rotating much faster than the others. The traction control system then applies a brake to that wheel to reduce its slip and then reducing wheel slip.

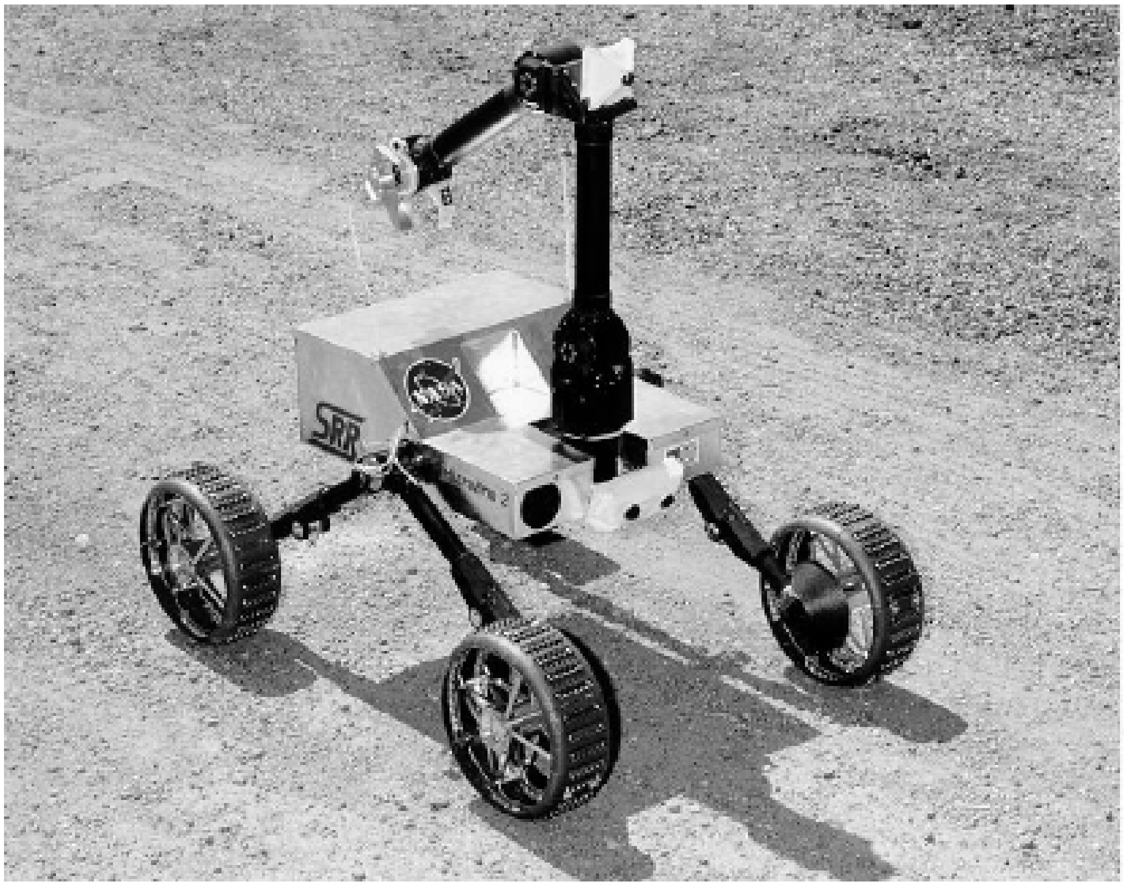

Figure 2.1: An Early Version of the Sample-Return Rover

Algorithms

Introduction

The *Traction Control Algorithm* (TCA) implementation for the *Curiosity Mars Rover* (CMR) is one that heavily utilizes geometry and a rigid-body model assumption, while also placing an emphasis on relying on minimal assumptions/data about the environment and avoiding computationally complex calculations on input data from sensors, cameras, etc. There were various reasons as to why some of these assumptions/choices were made and why the agreed upon implementation was deemed the best choice for the given scenario. [2] Figure 3.1 shows a testbed version of the CMR.

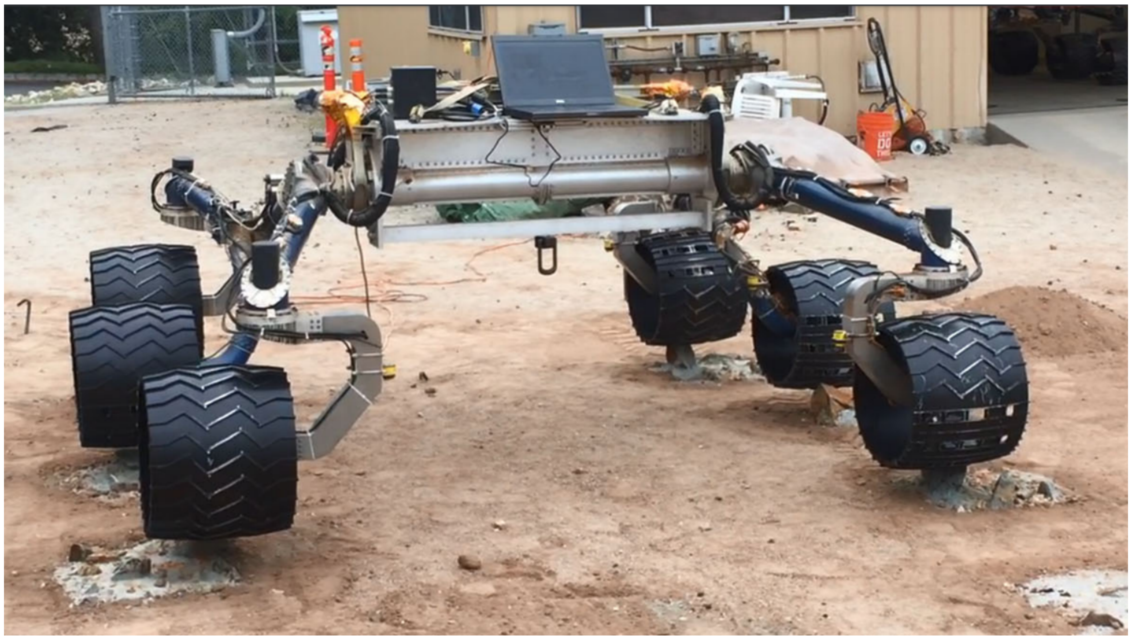

Figure 3.1: The Scarecrow Testbed Rover, a Rover Kinematically Similar to the *Curiosity Mars Rover* [2]

Design Limitations and Considerations

It is important to note that the need for this TCA was not discovered until after the flight system was actively engaging in its mission on the Martian surface, while ground control was observing telemetry of its use. Therefore, it was impossible to make any physical modifications to the CMR, only software could be remotely flashed to it.

However, there are still ramifications to having this additional routine run on the CMR. The limited computational resources available to do so must be considered carefully, so as to not interfere with existing processes being ran, and the implementation chosen has to have enough resources to perform the task it needs to as well. The team responsible for solving the problem at hand clarified some of these issues and how it restricted their choices for strategies to solve the problem. For example, the rover “does not include force or torque sensors on the mobility subsystem, nor can it measure slip with high enough frequency to be able to react to it.” [2]

Some of the characteristics and assumptions of the CMR and its TCA implementation are discussed in the following sections.

Ackermann Steering Model

Modeling vehicles using the Ackermann steering model is a common practice, including for the CMR. Even thought it adds slightly more complex geometric modeling, the mechanical steering system can be implemented relatively easily, and there are benefits to doing so.

Figure 3.2 illustrates a basic vehicle that is modeled using Ackermann steering, where the left image has the wheels positioned such that the vehicle will move straight, while the right image would cause the vehicle to turn counter-clockwise. Important to note is that when in a turning position, the front wheels are not turned to the same angle as one another, because of the nonzero distance between them. They are positioned such that the direction that both wheels are pointing are normal to a common center point called the *Instantaneous Center of Curvature* (ICC), so that when the vehicle turns, the left and right wheels follow two different circular arcs, but both of their centers are at the ICC.

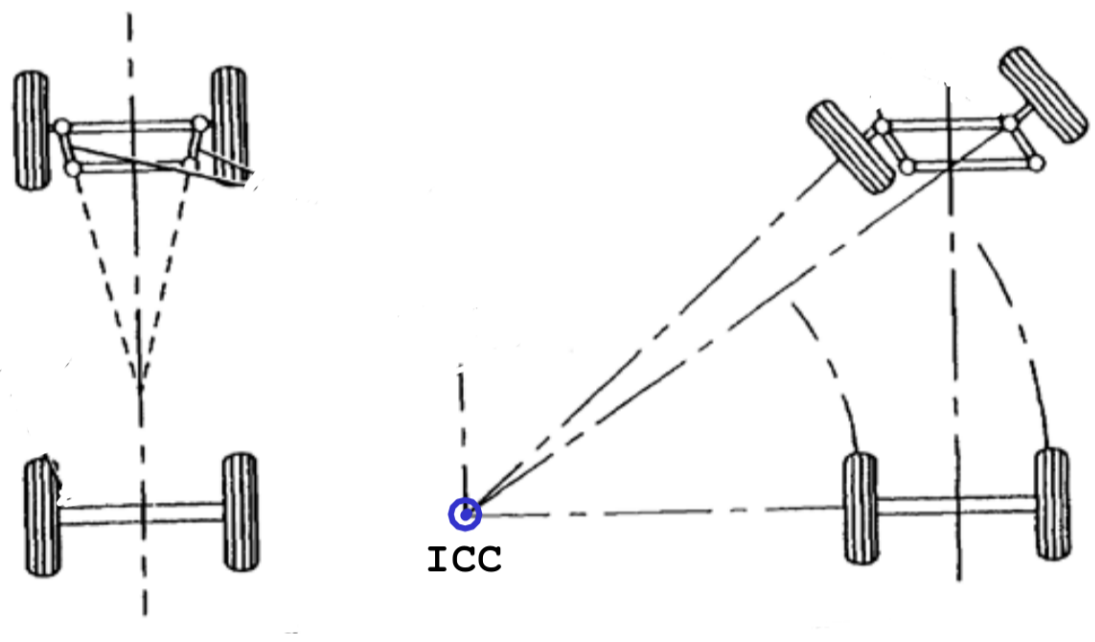

Figure 3.2: A Vehicle with Ackermann Steering Going Straight (Left) and in a Turning Position(Right)

The benefit to this is that, assuming all the wheels' velocities are coordinated and in an ideal world, at no point while traveling in these flat, circular arcs will a wheel slip. Less slippage means less unpredictable behavior, such as a wheel slipping over a rough surface. However, the assumption that the CMR would only be traversing terrain that could be modeled as flat is what led to the need for the TCA, as excessive slippage occurred due to the reality that the terrain was non-negligible. As one wheel would traverse over a rock and then back down, it traveled a longer distance.

The Ackermann steering model by itself does not account for this extra distance and this caused the wheel slippage, which damaged them at an alarming rate. This is how the TCA can be used to improve the model, so that when rough terrain is being traversed, the wheel(s) doing so can be sped up accordingly.

Rigid-Body Model Assumption

The rigid-body model assumption says that a body of some sturdy material can be assumed to maintain it shape, and that a force/torque that it can expect to encounter will not deform it to a degree that needs to be accounted for. This is not true in reality, but in scenarios like these, the deformation can be negligible.

Making this assumption can be extremely beneficial when implementing an algorithm like TCA because it heavily simplifies the math needed to represent vectors in different coordinate frames.

Coordinate Frame Notation

TO-DO, add:
1.) description of what a coordinate frame is and why they're used

2.) CMR frame notation
3.) mention differences in *Sample-Return Rover* (SRR) frames
4.) maybe more

Example rotation matrix between some generic frames A and B equation:

$$ {}^{B}_{A}R = \begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{bmatrix} \tag{3.1} $$

The rover, like any robot, has coordinate frames to define its Denavit-Hartenburg parameters (DH) and rotation matrices. Figure 3.3 shows these frames from the left side of the robot. A top view is also provided as Figure 3.4.

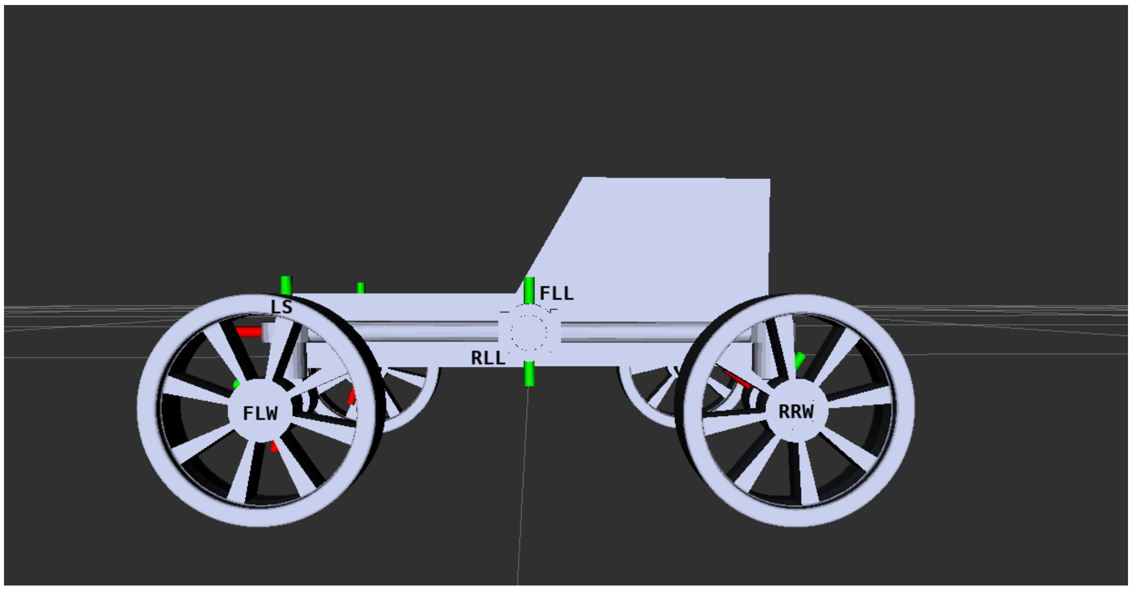

Figure 3.3: Side View of the Robot Model and its Coordinate Frames

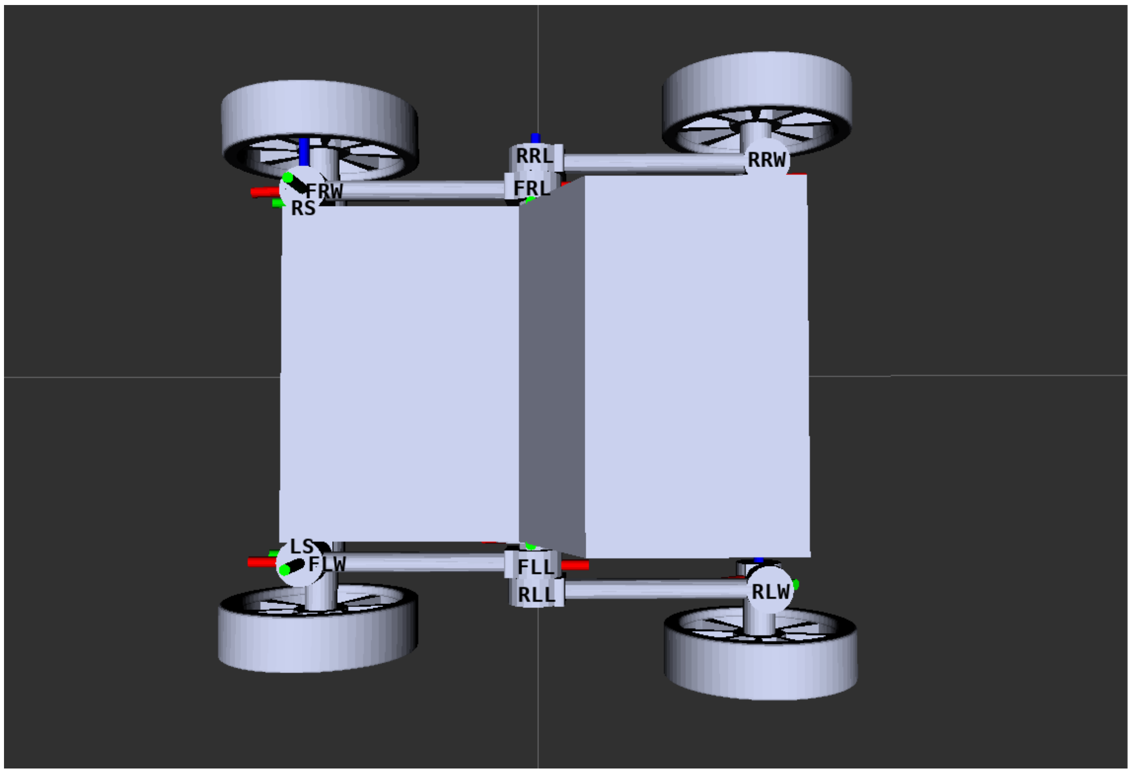

Figure 3.4: Side View of the Robot Model and its Coordinate Frames

A summary of the acronyms for the coordinate frames used in Figures 3.3 and 3.4 are seen below in Table 3.5.

FLL	Front Left Leg
FRL	Front Right Leg
RLL	Rear Left Leg
RRL	Rear Right Leg
FLW	Front Left Wheel
FRW	Front Right Wheel
RLW	Rear Left Wheel
RRW	Rear Right Wheel
LS	Left Steering
RS	Right Steering

Calculations

Overview

For this algorithm on the CMR, there are a number of configuration parameters (values that do not change throughout the lifetime of the system), a few inputs (values that are updated for each iteration of the control loop), and one output. The configuration inputs are related to the geometry of the vehicle, as that is obviously unchanging. The inputs are values such as the joint positions of the vehicle and kinematic values. The output are the ideal wheel speeds to maximize the rover's velocity while avoiding wheel slippage. These values are summarized in detail in Tables 3.1 and 3.2.

Input Value	**Description**
$[\omega_x, \omega_y, \omega_z]^T$	The 3x1 angular velocity vector around the base coordinate frame of the vehicle
$[\beta, \rho_1, \rho_2]^T$	The 3x1 vector of joint positions describing the rotation of the bogie joint and each rocker joint
$\left[\dot{\beta}, \dot{\rho_1}, \dot{\rho_2}\right]^T$	The 3x1 vector of angular velocities describing the rotational speed of the bogie joint and each rocker joint
$\vec{\Psi}$	The 6x1 vector of joint positions describing the steering angle of each wheel

Table 3.1: Summary of the Inputs for the Traction Control Algorithm Onboard the Curiosity Mars Rover

Output Value	**Description**
$\dot{\theta}$	The 6x1 angular velocity vector describing the ideal wheel speeds to command to the *Curiosity Mars Rover*

Table 3.2: Summary of the Inputs for the Traction Control Algorithm Onboard the Curiosity Mars Rover

When considering this algorithm for the SRR, the inputs and outputs will only change because the number of suspension joints and the number of wheels differs. The inputs and outputs for an SRR implementation are summarized in Tables 3.3 and 3.4.

Input Value	**Description**
$[\omega_x, \omega_y, \omega_z]^T$	The 3x1 angular velocity vector around the base coordinate frame of the vehicle
$\vec{\delta}$	The 4x1 vector of joint positions describing the rotation of each joint corresponding to the four legs' pivot points
$\dot{\vec{\delta}}$	The 4x1 vector of angular velocities describing the speed of rotation of each joint corresponding to the four legs' pivot points
$\vec{\Psi}$	The 2x1 vector of joint positions describing the steering angle of each wheel

Table 3.3: Summary of the Inputs for the Traction Control Algorithm Onboard the Curiosity Mars Rover

Output Value	**Description**
$\dot{\theta}$	The 4x1 angular velocity vector describing the ideal wheel speeds to command to the *Sample-Return Rover*

Table 3.4: Summary of the Inputs for the Traction Control Algorithm Onboard the Curiosity Mars Rover

Measuring the Inputs

As mentioned previously, one of the advantages to this implementation for traction control is that there is relatively less input data to process. Nonetheless, there are still the few inputs described in Tables 3.1 and 3.3 that must be measured.

Each of the joints on the CMR that appear in Table 3.1 have encoders that measure the current absolute position of the joint, including β, $\rho_{1,2}$, and Ψ_{1-6}. Some encoders are capable of measuring the current velocity of the corresponding motor as well, but the CMR was not equipped with these. The team behind the solution in [2] proposed instead manually computing the time derivative of the input positions to estimate each angular velocity ($\dot{\beta}$, $\dot{\rho}_1$, and $\dot{\rho}_2$). In other words, given two contiguous samples of the position of each joint, the average angular velocity over the length of time between these two samples Δt can be calculated for some angle α as:

$$\dot{\alpha} \approx \dot{\alpha}_{ave} = \frac{\alpha_1 - \alpha_0}{\Delta t} \tag{3.2}$$

The angular velocity vector $\dot{\omega}$ is measured with an *Inertial Measurement Unit* (IMU), and more specifically with the gyroscope(s) within it. This mechanical device is a mechanism with three degrees of freedom, whose components rotate about their axes as it's turned itself. These rotations are measured as the angular velocities about those three principal axes. See Figure 3.5 for a 3D rendering of a gyroscope.

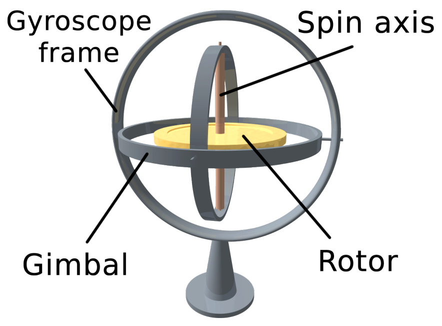

Figure 3.5: A 3D Rendering of a Gyroscope

Transforming Velocities Between Coordinate Frames

To solve the problem at hand, the velocity of each wheel must be calculated in their own respective coordinate frames, which would be due to the linear and angular velocities of the vehicle, as well as then angular velocities of each of the joints of the links that connect the wheel to the chassis of the vehicle. An estimate for this velocity can be calculated once the angle between the wheel and the surface it is on is found, called the contact angle. Figure 3.6 illustrates the velocity $\vec{v}$ and contact angle η of a generic wheel A.

In general, the linear velocity of some point Q resolved in some coordinate frame H (which is attached to some rigid body) can be expressed as a function of some other frame G (which is attached to some other rigid body), as follows [1]:

$$^{G}\vec{V}_Q = {^{G}\vec{V}_{H_0}} + ({^{G}_{H}R} \cdot {^{H}\vec{V}_Q}), \text{when } {^{G}_{H}\dot{R}} = I_3 \tag{3.3}$$

In other words, Equation 3.3 states that when the rotational offset between frames G and H is constant, the linear velocity of point Q resolved in G is the sum of the linear velocity of the origin of H with respect to G and the linear velocity of Q with respect to H rotated to be resolved in G. However, it is not accurate to assume the orientation between two bodies will be static as time

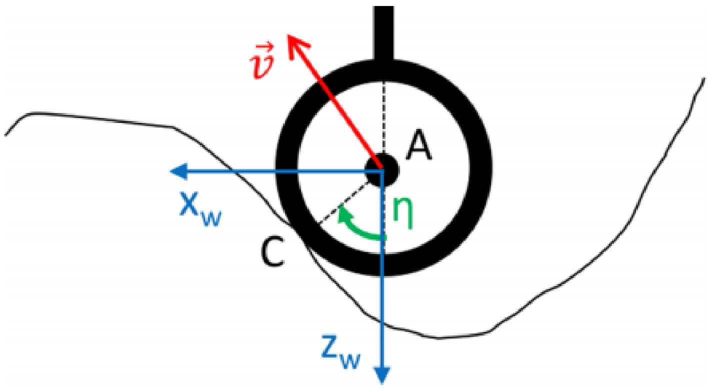

Figure 3.6: The Coordinate Frame Attached to the Wheel, Illustrating the Wheel's Velocity and Contact Angle to the Surface [2]

passes. The linear velocity of a point induced by a rotational offset that changes with time can be calculated as follows [1]:

$$^{G}\vec{V}_Q = {}^{G}\vec{\omega}_H \times ({}^{G}_{H}R \cdot {}^{B}Q), \text{when } {}^{G}\vec{V}_{H_0} = {}^{H}\vec{V}_Q = \vec{0}_{3x1} \tag{3.4}$$

Equation 3.4 states that when the origins of frames G and H are coincident but the orientation between them changes as time passes, the linear velocity of Q resolved in frame H equals the cross product of the angular velocity between frames G and H with point Q resolved in frame G.

Equations 3.3 and 3.4 can be combined to eliminate their prior assumptions, such that:

$$^{G}\vec{V}_Q = {}^{G}\vec{V}_{H_0} + ({}^{G}_{H}R \cdot {}^{H}\vec{V}_Q) + ({}^{G}\vec{\omega}_H \times ({}^{G}_{H}R \cdot {}^{B}Q)) \tag{3.5}$$

Equation 3.5 describes how to calculate the linear velocity of a point resolved to one frame attached to a rigid body that was originally resolved to a frame attached to another rigid body. Now consider the implications of these frames being attached to a single rigid body instead. The orientation between frames G and H cannot change, because the frames rotate as the body does, implying that rotating one rotates the other in the same manner.

$$^{G}\vec{V}_Q = DummyEqn. \tag{3.6}$$

XXXXXXX
FINISH ME LATER
XXXXXXX

Bibliography

[1] John J. Craig. *Introduction to Robotics: Mechanics and Control*. 2nd ed. Reading, Massachusetts: Addison-Wesley, 1989. ISBN: 0201095289.

[2] Arturo Rankin Amanda Steffy Gareth Meirion-Griffith Dan Levine Maximilian Schadegg Mark Maimone Olivier Toupet Jeffrey Biesiadecki. "Terrain-Adaptive Wheel Speed Control on the Curiosity Mars Rover: Algorithm and Flight Results". In: *Journal of Field Robotics* (2018), pp. 699–728. DOI: `http://dx.doi.org/10.1002/rob.21903`.

[3] T. L. Huntsberger P. S. Schenker and G. T. McKee University of Reading (UK) P. Pirjanian Jet Propulsion Laboratory (USA). "Robotic Autonomy for Space: Cooperative and Reconfigurable Mobile Surface Systems". In: i-SAIRAS 2001. Canadian Space Agency, St. Hubert, Quebec, Canada, 2001.