MANCHESTER
1824
The University of Manchester

Thesis Title

A thesis submitted to the University of Manchester for the degree of
Doctor of Philosophy
in the Faculty of Science and Engineering

2022

Damian Crosby
Department of Mechanical, Aerospace and Civil Engineering

Contents

Word Count: X

List of figures

List of tables

List of publications

Publications go here.

List of abbreviations

AUJ Actuated Universal Joint

2D Two Dimensional

3D Three Dimensional

COM Centre of Mass

DLS Damped Least Squares

DOF Degree(s) of Freedom

PID Proportional Integral Derivative

SHTP Sensor Hub Transport Protocol

MSB Most Significant Bit

LSB Least Significant Bit

I^2C Inter-Integrated Circuit

SPI Serial Peripheral Interface

UART Universal Asynchronous Receiver-Transmitter

DIO Digital Input/Output

IMU Inertial Measurement Unit

GOOP Graphical Object Oriented Programming

CS Chip Select

Abstract

This is abstract text.

Declaration of originality

I hereby confirm that no portion of the work referred to in the thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Copyright statement

i The author of this thesis (including any appendices and/or schedules to this thesis) owns certain copyright or related rights in it (the "Copyright") and s/he has given The University of Manchester certain rights to use such Copyright, including for administrative purposes.

ii Copies of this thesis, either in full or in extracts and whether in hard or electronic copy, may be made *only* in accordance with the Copyright, Designs and Patents Act 1988 (as amended) and regulations issued under it or, where appropriate, in accordance with licensing agreements which the University has from time to time. This page must form part of any such copies made.

iii The ownership of certain Copyright, patents, designs, trademarks and other intellectual property (the "Intellectual Property") and any reproductions of copyright works in the thesis, for example graphs and tables ("Reproductions"), which may be described in this thesis, may not be owned by the author and may be owned by third parties. Such Intellectual Property and Reproductions cannot and must not be made available for use without the prior written permission of the owner(s) of the relevant Intellectual Property and/or Reproductions.

iv Further information on the conditions under which disclosure, publication and commercialisation of this thesis, the Copyright and any Intellectual Property and/or Reproductions described in it may take place is available in the University IP Policy (see `http://documents.manchester.ac.uk/DocuInfo.aspx?DocID=24420`), in any relevant Thesis restriction declarations deposited in the University Library, The University Library's regulations (see `http://www.library.manchester.ac.uk/about/regulations/`) and in The University's policy on Presentation of Theses.

Acknowledgements

Acknowledgements go here.

Chapter 1

Introduction

1.1 Mobile Robot Stability

Stability is a significant issue for mobile robot design. Loss of stability can mean the robot is unable to move and must be reorientated or retrieved, which maybe difficult or impossible in some extreme environments, such as in outer space or a nuclear fuel pool. In the worst case it can result in severe damage or destruction of the robot, and any objects it is carrying. This has become more of an issue as mobile robots have become increasingly fast and agile, often running, jumping and hopping around less controlled environments.

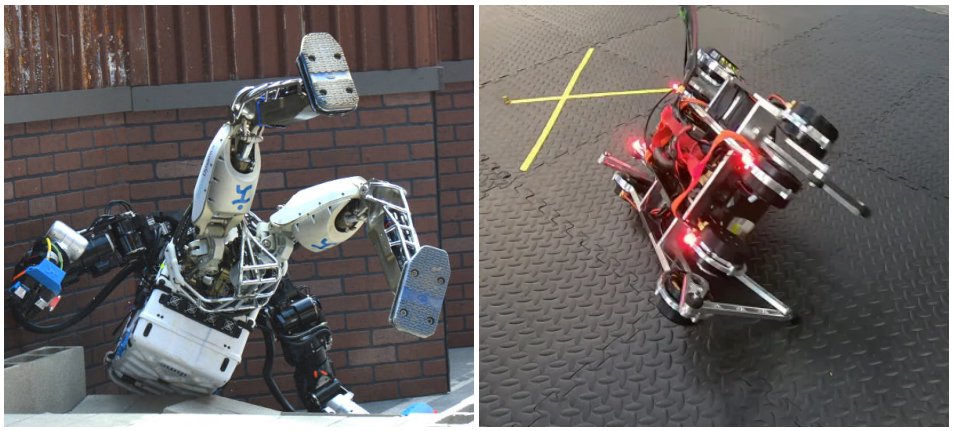


Figure 1.1: Examples of loss of stability in bipedal and quadrupedal robots.

In many ways the consequences of stability loss in mobile robots are analogous to other situations. A human that falls over has to pick themselves up before continuing on, or if they are infirm they may require assistance. Likewise they could also suffer injury, or if walking along the edge of a long drop, fall to cause severe injury or death. A forklift truck or other piece of heavy plant can topple, injuring the driver and causing the damage or destruction of vehicles and materials.

In general, stability from a biomechanical perspective can be divided into two different types, *static* and *dynamic*. While there are numerous ways to define the difference between the two, such as the maximum lyupanov exponent for dynamic stability, the following definitions will be used:

- **Static stability** only considers the uniform force of gravity and assumes no other forces are acting on the object.

- **Dynamic stability** considers other forces and torques on the object, both internal and external, as well as gravity.

A stationary object that has no external forces or torques being applied needs to be statically stable, a moving object, or an object that is having a force or torque applied to it other than gravity, needs to be dynamically stable.

Figure 1.2: Examples of loss of stability (static or dynamic) in a human and forklift truck.

1.1.1 Static Stability

To determine if the robot is statically stable, the gravity axis projection of the center of gravity needs to fall within a defined "support polygon" on the plane perpendicular to the gravity axis plane, as in figure 1.3. If gravity is treated as a constant force along vector g, then the center of mass and center of gravity are equivalent. The center of mass can be calculated for using equation 1.1 for n bodies of masses $m_{1...n}$ and COM positions $\boldsymbol{p_{1...n}}$. If the gravity vector is parallel to any of the basis vectors, then the perpendicular components of the COM/COG can be used to determine the static stability.

$$\text{COG} = \text{COM} = \frac{\sum_{i=1}^{n} m_i \boldsymbol{p_i}}{\sum_{i=1}^{n} m_i} \tag{1.1}$$

Fundamentally, there are two different methods to maintain static stability:

- Change the region or shape of the support polygon so the gravity axis projection of the center of gravity remains within the bounds of the support polygon.
- Move the gravity axis projection of the center of gravity so it remains within the bounds of the support polygon.

The former method can be considered equivalent to using a walking pole to steady yourself on uneven ground when hiking, the leg of the pole acts as a new vertex that is used to calculate the support polygon, expanding it sufficiently, or placing your foot in front of you when walking. The latter method can be considered equivalent to leaning back to remain upright when falling forward. Leaning back moves the centre of gravity to keep it within the support polygon.

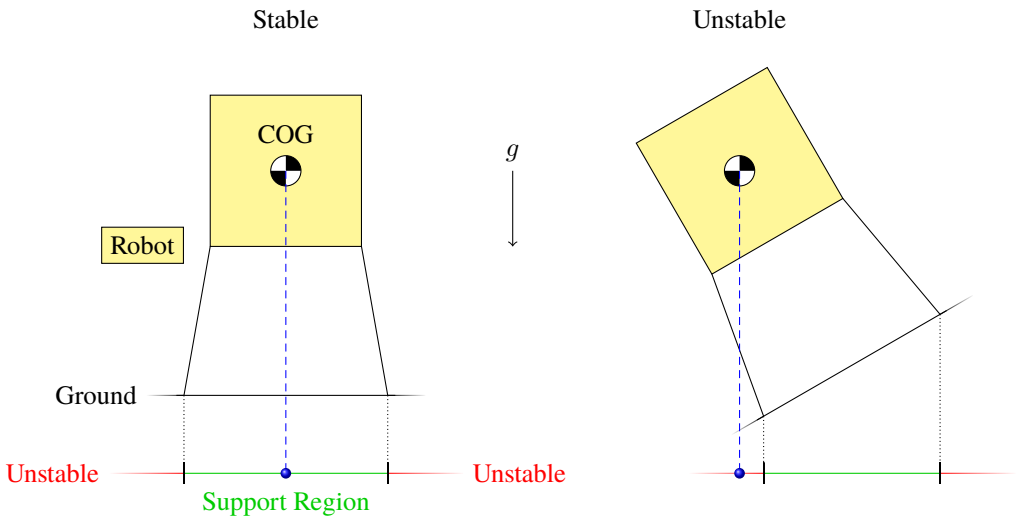


Figure 1.3: 2D representation of the static stability of a legged robot, with the support region defined by the contact points of the legs with the ground. Notice how the orientation of the robot with respect to the gravity axis can move it in and out of the static stability region.

However, just because the center of gravity falls outside of the support polygon does not mean that loss of stability is inevitable, as long as a force or torque applied to the body counteracts the forces and torques induced by the force of gravity in order to maintain stability. This is similar to applying a torque to your ankle when standing on one leg, or a strong wind keeping you upright when leaning forward. If this is the case, then *dynamic* stability is maintained while *static* stability is lost. If the force or torque is removed, then stability is lost. Conversely, forces and torques can also cause loss of stability even if the center of gravity does not fall outside of the support polygon. This is similar to being pushed over, or stopping too quickly and falling forward. So dynamic stability can maintain stability even if static stability is lost, but static stability cannot maintain stability if dynamic stability is lost.

Support Polygon

1.1.2 Dynamic Stability

To determine if the robot is dynamically stable,

The concept of the Zero Moment Point, first defined in, is useful for mobile robots to check if dynamic stability will be maintained. It extends the calculation used for static stability by including inertial forces caused by accelerations of the bodies, as shown in figure 1.4. Though it has mostly been utilised for bipedal robots to ensure stability while walking, it is also applicable to quadruped robots, and has even been investigated for the development of a stability warning system in road vehicles [1]. The ZMP is formally defined as the point at which the point where the total of horizontal inertia and gravity forces equals zero. It can be thought of as a *dynamically augmented* version of the gravity axis projection of the center of mass.

Equation 1.2 defines the position of the ZMP for a robot or vehicle with n bodies of masses

$m_{1...n}$, COM positions $\boldsymbol{p_{1...n}}$ and COM accelerations $\ddot{\boldsymbol{p}}_{\boldsymbol{1...n}}$, in contact with a planar surface of normal vector $\boldsymbol{n}$ (ZMP cannot be calculated for non-planar surfaces). $\boldsymbol{\tau_{i...n}}$ defines the torque acting on each COM, which can be calculated from .

$$\begin{aligned} \boldsymbol{\tau_i} &= \boldsymbol{R_i}\left(\boldsymbol{I_i}\ddot{\theta}_i - \left(\boldsymbol{I_i}\dot{\theta}_i\right) \times \dot{\theta}_i\right) \\ \text{ZMP} &= \frac{\boldsymbol{n} \times \sum_{i=1}^{n}\left(\boldsymbol{p_i} \times m_i\boldsymbol{g} - \boldsymbol{p_i} \times m_i\ddot{\boldsymbol{p}}_{\boldsymbol{i}} - \boldsymbol{\tau_i}\right)}{\boldsymbol{n} \cdot \left(\left(\sum_{i=1}^{n} m_i\boldsymbol{g}\right) - \left(\sum_{i=1}^{n} m_i\ddot{\boldsymbol{p}}_{\boldsymbol{i}}\right)\right)} \end{aligned} \tag{1.2}$$

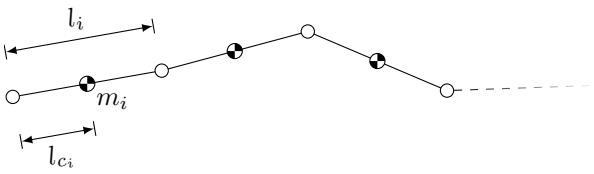


Figure 1.4: 2D representation of the ZMP of a legged robot under horizontal acceleration, with the support region defined by the contact points of the legs with the ground. Notice how increasing the horizontal acceleration of the robot can make it dynamically unstable.

1.1.3 Payload

When a payload is added to the robot, it is equivalent to instantaneously adding an extra body to the robot of mass m_p and position p_p thus changing the COM. Initially the payload will also create an extra contact point with the ground, changing the support polygon so the robot remains statically stable. However, as soon as contact between the payload and ground is severed, the robot can become statically unstable. This does not mean the robot will immediately lose stability, the dynamic forces created picking up the payload may keep the robot dynamically stable, but once the robot is stationary without other forces acting upon it, it may lose stability. The best way to compensate for this is to change the position of the other bodies so the COM remains within the support polygon. This is akin to leaning back when carrying something heavy. But leaning generally has limited range, so can only compensate for lighter payloads. An alternative is needed for heavy payloads.

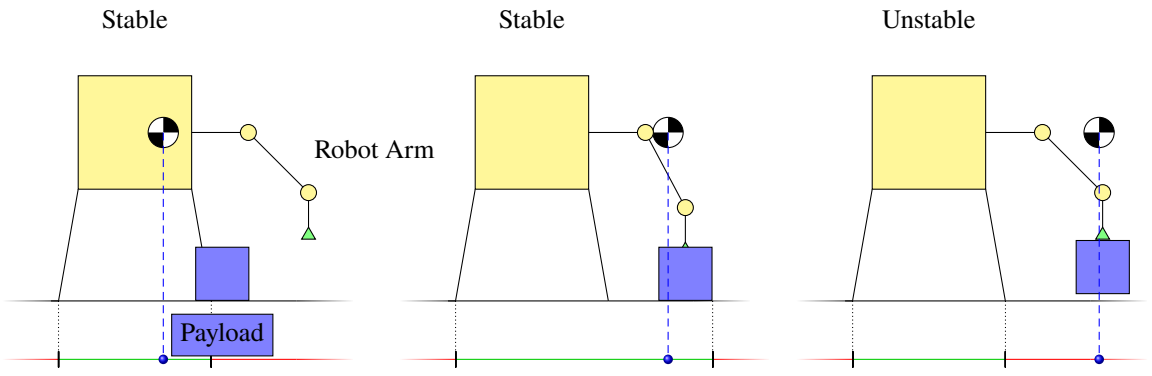


Figure 1.5: 2D representation of the static stability of a legged robot picking up a payload, with the support region defined by the contact points of the arm and legs with the ground. Notice how the payload acts as a contact point until it is lifted off the ground, preventing loss of stability even as the COM/COG is translated due to the mass of the payload.

1.2 Tails

1.2.1 Tails for Stability in the Animal Kingdom

Tails are a common sight in vertebrate animals, a natural extension of the spinal column. While some tails are used purely for grasping, locomotion, communication or decoration, many have some function in maintaining stability.

Case Study 1: Balance

[2] demonstrates how the domestic cat uses its tail for balance when walking along a narrow beam, which was shifted laterally at a certain velocity by 2.5 cm or 5 cm while they are traversing it. Four cats were trained to walk across the beam, before and after a surgical procedure that severed their spinal cord just above their tail, severely affecting its function. As can be seen from figure 1.6, this procedure caused the cats to fall from the beam far more often than before surgery.

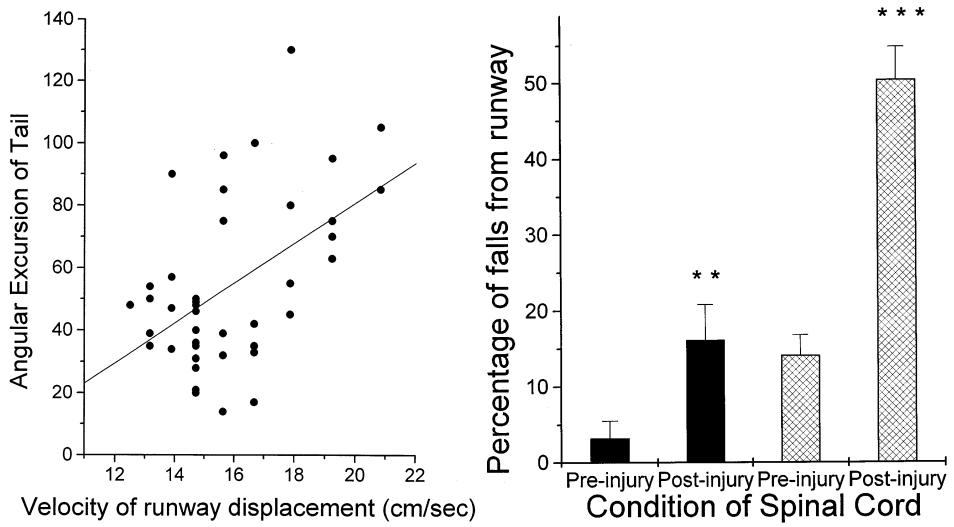


Figure 1.6: Charts from [2] showing how the cat's tail is used to maintain balance on the beam when it is shifted, and how impairing it causes a major loss of stability. Dark bars are a 2.5 cm displacement, cross-hatched bars are a 5 cm displacement.

Case Study 2: Inertial Force/Torque Compensation

Other animals use a tail to compensate for inertial forces and torques induced during a change in velocity, either in magnitude or direction. [3], [4] examines how the cheetah uses its tail to counteract centrifugal force when turning at high speed, and acceleration and braking forces when speeding up and slowing down. They then applied this to a robotic vehicle, which is discussed in section 1.2.2.

Figure 1.7: Images from [3], [4] of a cheetah using its tail during a turn and braking while chasing a lure.

Case Study 3: Aerial Reorientation

Finally, some animals use their tail to remain upright while airborne. **gillis604** examines the aerial stability of the arboreal lizard with an intact tail and with their tail removed. Lizards with the tail removed are unable to maintain their body orientation and do not land cleanly, as can bee seen in figure 1.8. [5] then applies this to a robotic vehicle, discussed in section 1.2.2.

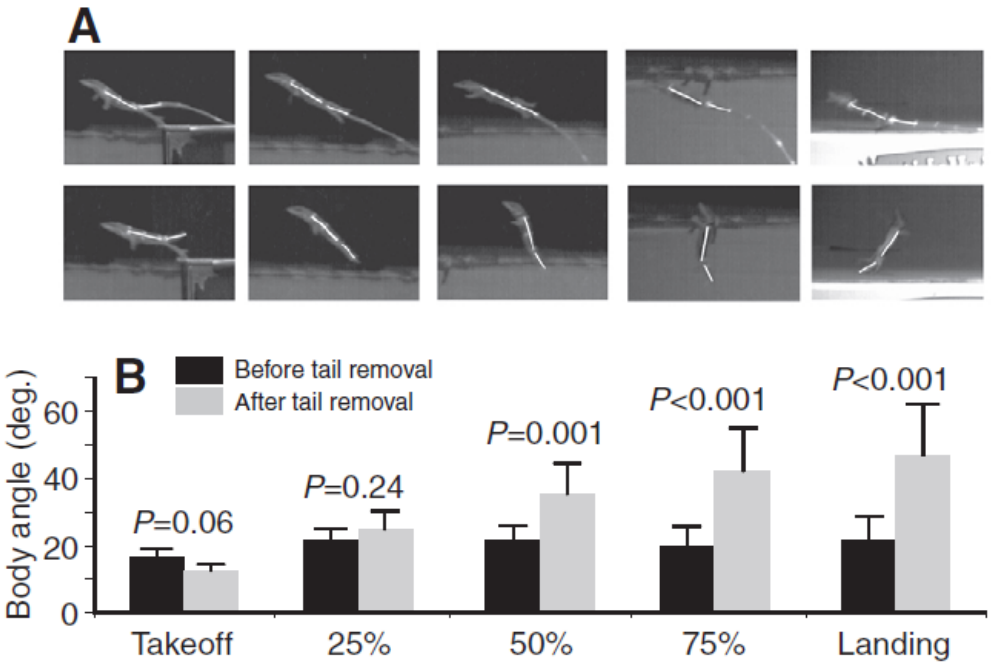


Figure 1.8: Image from **gillis604**, showing the body angle of a lizard during a jump, before and after tail removal.

1.2.2 State of the Art Mobile Robots

As many mobile robots have looked at animals for inspiration when it comes to solving various problems with dynamics and locomotion, tails have been used in the development of a significant selection of mobile robots, typically for similar functions as seen in the animal kingdom.

Case Study 1: Balance

[6] uses a tail to keep a legged robot, the "MIT Cheetah", from falling over when disturbed, in this case the disturbance being a "wrecking ball", slamming into its side. This can be seen as a similar reaction to the experiments by [2], though it again uses the example of the cheetah in the publication.

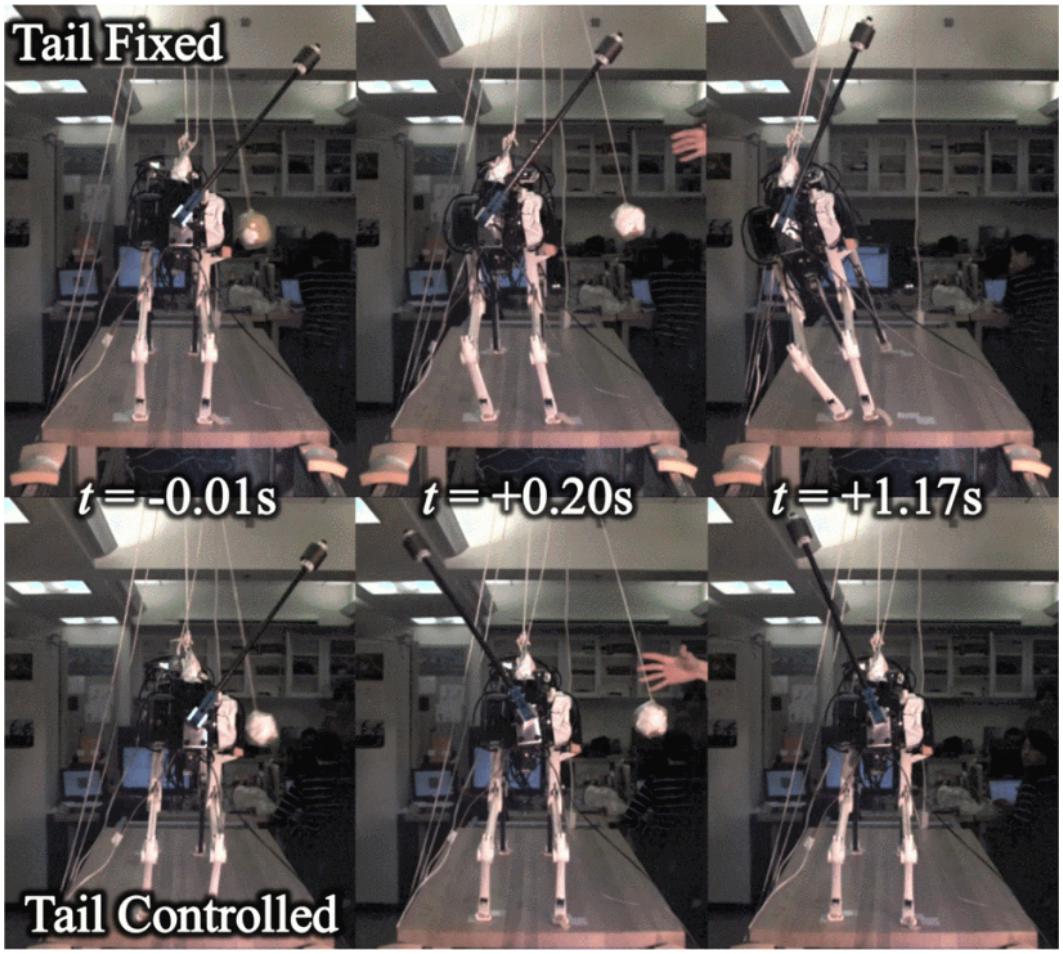


Figure 1.9: Image from [6] showing the difference in body compensation required with and without a moving tail when impacted by a "wrecking ball".

Case Study 2: Inertial Force/Torque Compensation

In [3], [4] which also details the functions of the cheetah tail, the robot "Dima" is fitted with a single segment 2-DoF tail which can replicate the motion of the cheetah when this wheeled robot makes a turn at high speed, or accelerates or brakes.

Case Study 3: Inducing Turning Torque

In [7]–[9] a roach like robot is fitted with a single-segment yawing tail. By swinging this tail rapidly to one side or the other during locomotion on a low-friction surface, a reaction torque causes the robot body to rotate and the legs to "skid" on the surface.

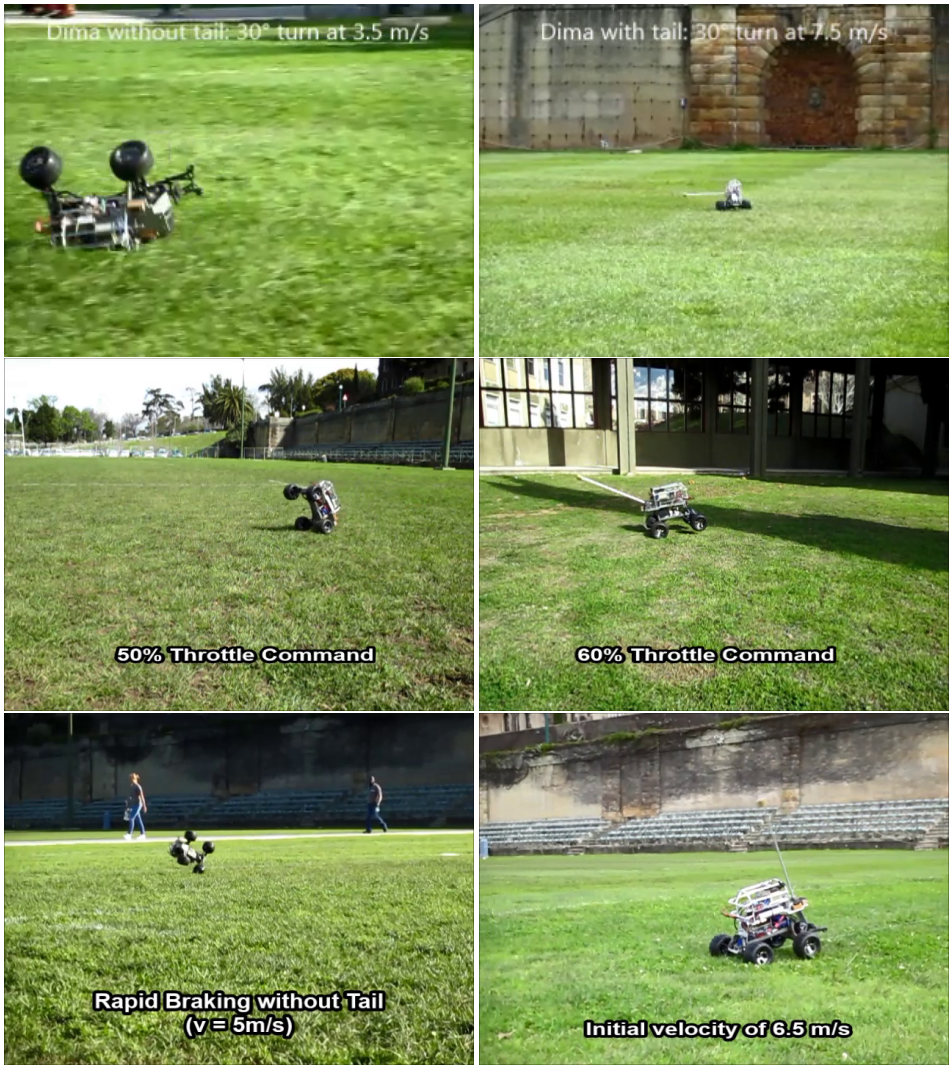


Figure 1.10: Video stills from [3], [4] of the "Dima" robot performing steering, acceleration and braking manoeuvres with and without a tail.

Case Study 4: Aerial Reorientation

In [10] a tail is used to maintain the pitch orientation of a robot when it is dropped from a height, by inducing torques on the body while airborne. They also consider the effect of tail contact against a surface (such as a wall) to increase the torque on the body. Similar studies in publications such as [11] also examine the same technique when a robot has forward momentum (when it drives off a ledge, for example).

References

[1] S. Lapapong, A. Brown, and S. Brennan, "Experimental validation of terrain-aware rollover prediction for ground vehicles using the zero-moment point method," p. 6, Oct. 2020.

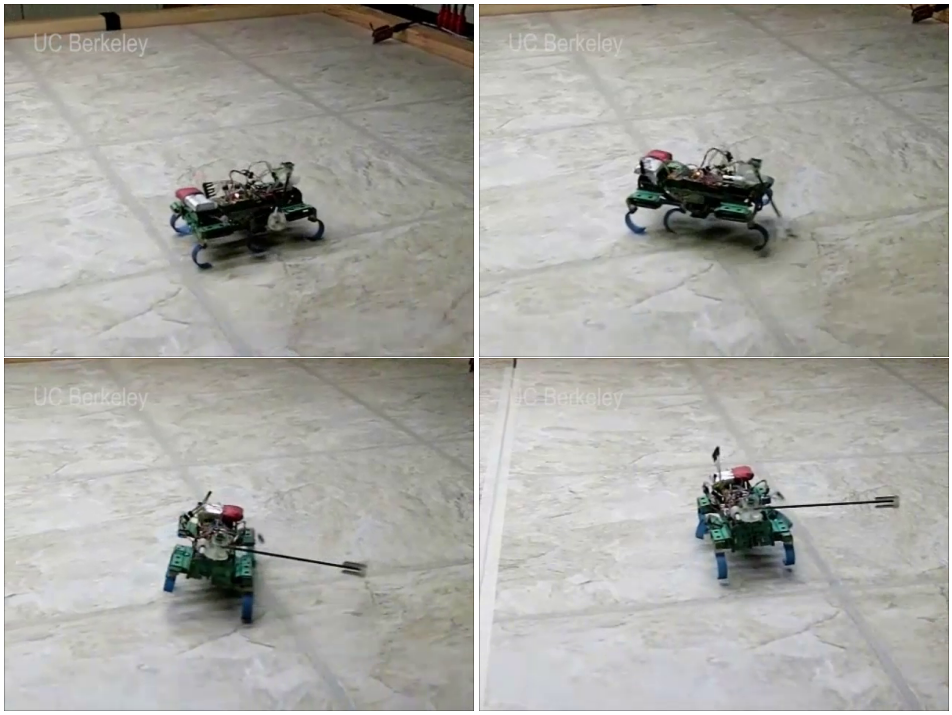


Figure 1.11: Video stills from [8] of the TalyRoACH robot using its tail to change its direction of motion.

[2] C. Walker, C. J. Vierck Jr, and L. A. Ritz, "Balance in the cat: Role of the tail and effects of sacrocaudal transection," *Behavioural brain research*, vol. 91, no. 1-2, pp. 41–47, 1998.

[3] A. Patel and M. Braae, "Rapid turning at high-speed: Inspirations from the cheetah's tail," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, IEEE, 2013, pp. 5506–5511. DOI: 10.1109/IROS.2013.6697154.

[4] ——, "Rapid acceleration and braking: Inspirations from the cheetah's tail," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, IEEE, 2014, pp. 793–799. DOI: 10.1109/ICRA.2014.6906945.

[5] T. Libby, T. Y. Moore, E. Chang-Siu, *et al.*, "Tail-assisted pitch control in lizards, robots and dinosaurs," *Nature*, vol. 481, no. 7380, pp. 181–184, 2012. DOI: doi : 10.1038/nature10710.

[6] R. Briggs, J.-W. Lee, M. Haberland, and S.-B. Kim, "Tails in biomimetic design: Analysis, simulation, and experiment," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, IEEE, 2012, pp. 1473–1480. DOI: 10.1109/IROS.2012.6386240.

[7] A. O. Pullin, N. J. Kohut, D. Zarrouk, and R. S. Fearing, "Dynamic turning of 13 cm robot comparing tail and differential drive," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, IEEE, 2012, pp. 5086–5093. DOI: 10.1109/ICRA.2012.6225261.

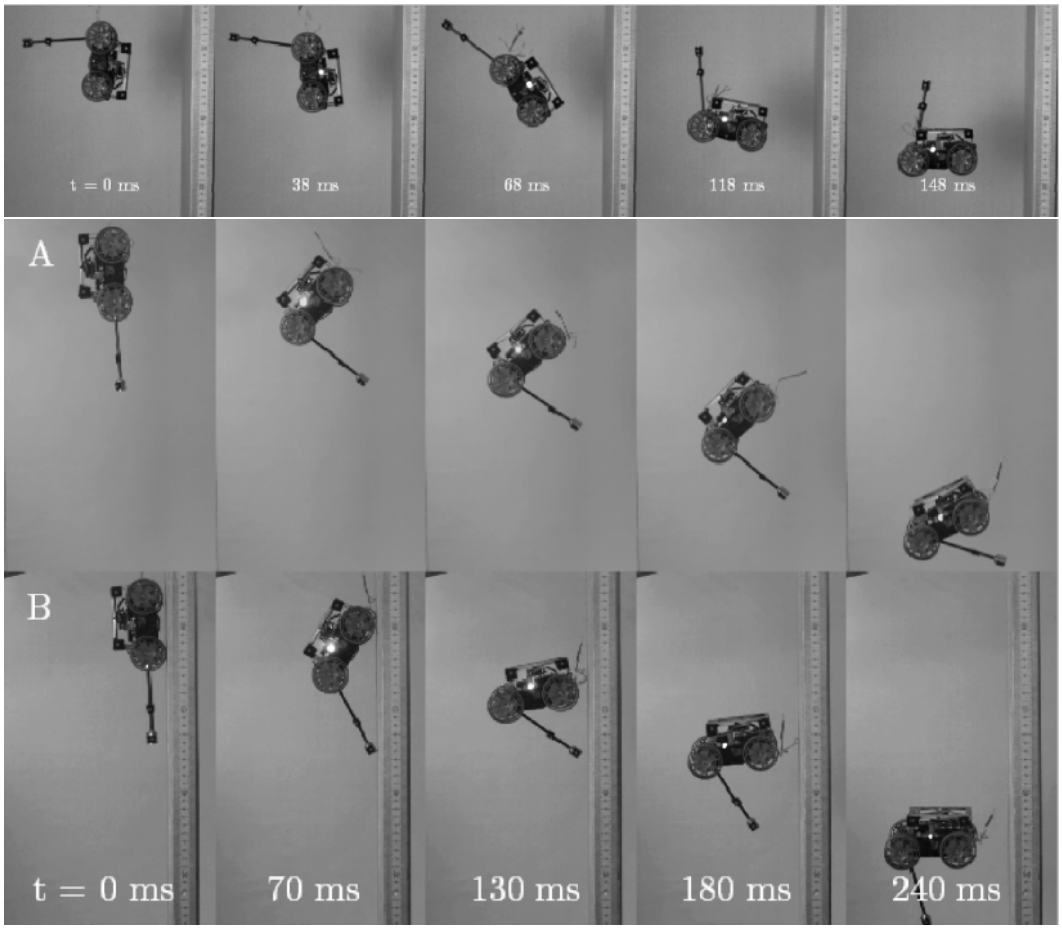


Figure 1.12: Images from [10] of the MSU Tailbot orienting itself after being dropped from a height, and how the tail contacting a wall can improve the orientation range.

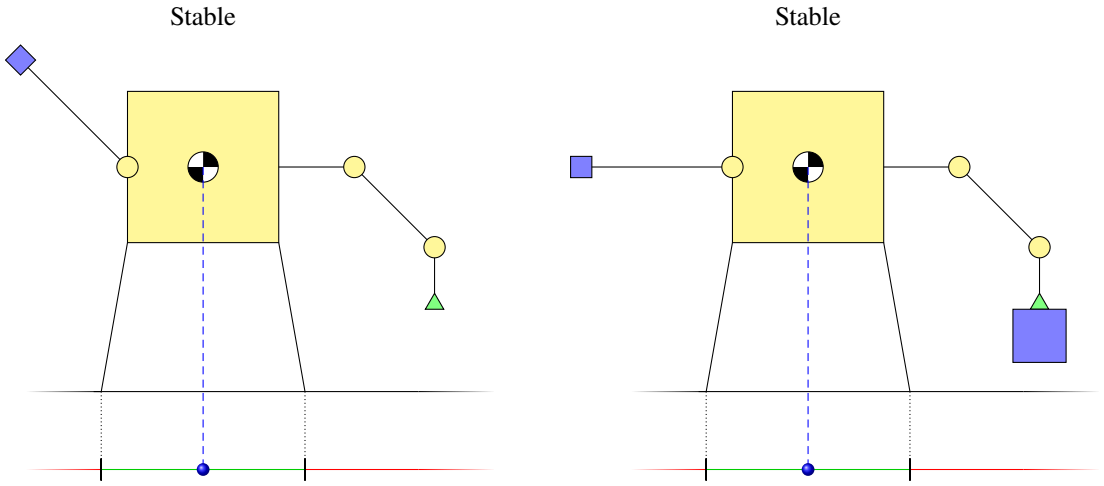


Figure 1.13: 2D representation of the static stability of a legged robot with a weighted tail picking up a payload. By changing the angle of the tail, the postion of its COG can be adjusted in order to compensate for the payload.

[8] N. J. Kohut, A. O. Pullin, D. W. Haldane, D. Zarrouk, and R. S. Fearing, "Precise dynamic turning of a 10 cm legged robot on a low friction surface using a tail," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, IEEE, 2013, pp. 3299–3306. DOI: 10.1109/ICRA.2013.6631037.

[9] N. Kohut, D. Haldane, D. Zarrouk, and R. Fearing, "Effect of inertial tail on yaw rate of 45 gram legged robot," in *Int. Conf. Climbing Walk. Robot. Support Technol. Mob. Mach*, 2012, pp. 157–164. DOI: 10.1142/9789814415958_0023.

[10] E. Chang-Siu, T. Libby, M. Tomizuka, and R. J. Full, "A lizard-inspired active tail enables rapid maneuvers and dynamic stabilization in a terrestrial robot," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, IEEE, 2011, pp. 1887–1894. DOI: 10.1109/IROS.2011.6094658.

[11] T. Libby, A. M. Johnson, E. Chang-Siu, R. J. Full, and D. E. Koditschek, "Comparative design, scaling, and control of appendages for inertial reorientation," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1380–1398, 2016. DOI: 10.1109/TRO.2016.2597316.

Chapter 2

A Literature Review of Terrestrial and Static Robots with Robotic Tails and Their Functions

2.1 Introduction

2.2 Research Methodology

2.3 Locomotion

2.4 Tail Function

2.4.1 Angular Stability during Locomotion

2.4.2 Angular Stability in Free Space

use a tail to change the orientation of a robot (by imparting a torque on the body) when the robot is in "free space" i.e. no part of the robot is in contact with the ground or other external surface. This is the most common use of a tail in terrestrial robots.

Hopping/Galloping Gait

wenger2012frontal use a tail when in the aerial phase of a hopping/galloping gait. This is to compensate for uneven terrain and/or slight differences in forces generated by each leg when in the leaping phase, imparting a pitch or roll moment on the body.

Single Jump

[2], [3] use a tail

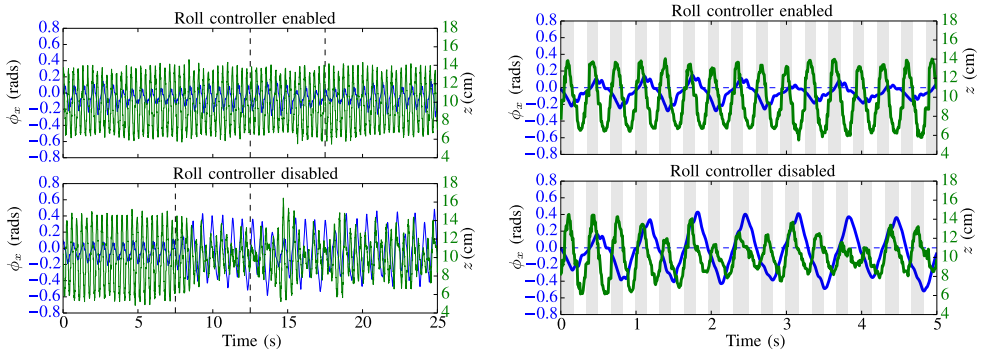


Figure 2.1: Graphs from [1] showing a steady state hopping gait with and without the tail roll controller. Body roll angle ϕ_x is in blue, and body height z is in green. These graphs demonstrate without a roll controller ϕ_x quickly increases in magnitude and becomes chaotic.

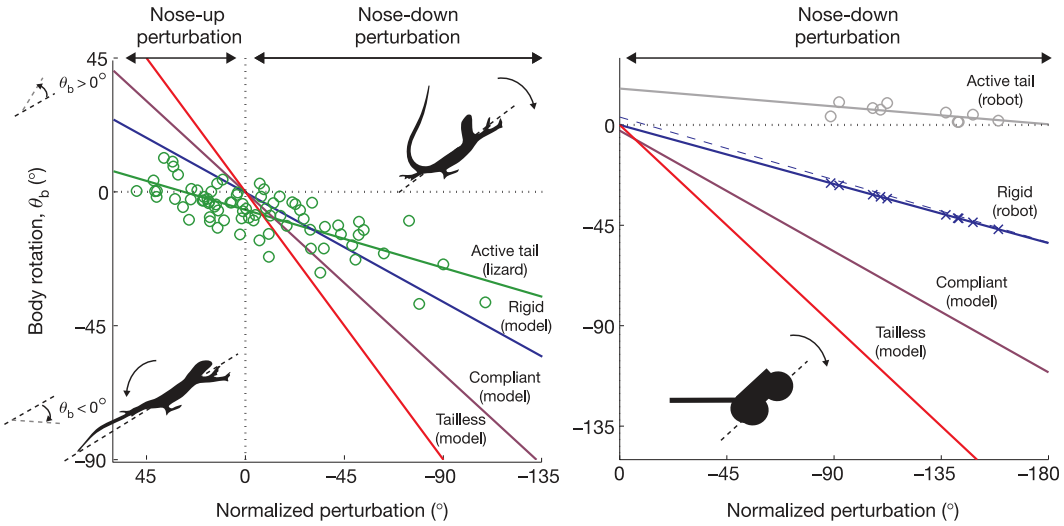


Figure 2.2: Graphs from [2] showing how a lizard and a robot can counteract a "normalised pertubation" disturbance, defined by how much the body would rotate without a tail.

In [2], a wheeled robot with a 1 DOF tail is compared to a lizard (*Agama agama*) that uses its tail to minimise pitch rotation after leaping off a surface. The robot outperformed the lizard, as can be seen in figure **??**.

In [3], a legged robot with a 1 DOF tail is able to change the body orientation so it lands on its side in a more stable configuration, then using the tail to self right.

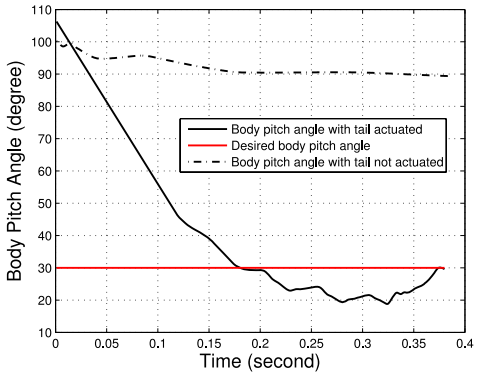


Figure 2.3: Graph from [2] showing the desired body pitch angle for landing on its side, and the trajectory of a jump with an actuated and unactuated tail.

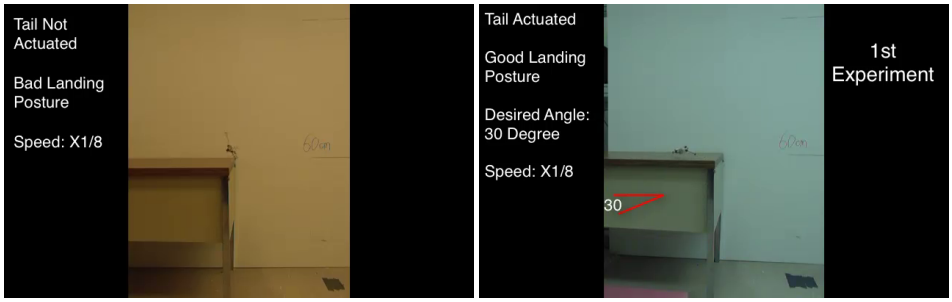


Figure 2.4: Still images from [2] showing an unstable landing on its feet, and a stable landing on its side.

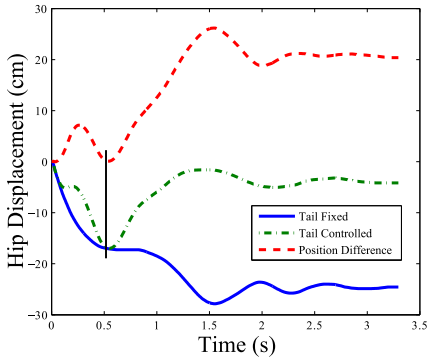


Figure 2.5: Graph from [4] showing how an active tail can minimise body sway when a disturbance force is imparted.

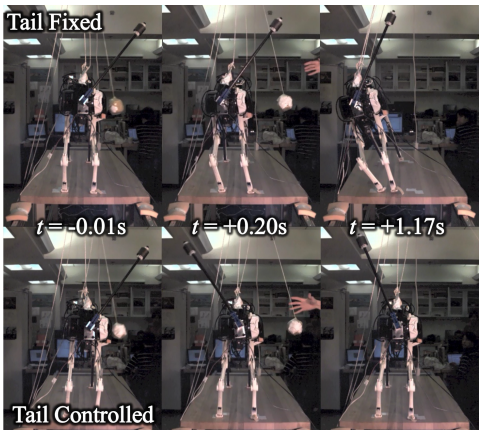


Figure 2.6: Still images from [4] showing the robot absorbing the impact of the wrecking ball with and without an active tail.

Free Fall

2.4.3 Angular Stability after Disturbance

[4], [5] use a tail to prevent a quadrupedal robot tipping when a disturbance force is imparted (using a "wrecking ball" hitting the body of the robot).

2.4.4 Path Drift

A number of papers with different kinds of locomotion

Inertial

[6] uses a quardrupedal robot with a 1 DOF head and tail. By swinging the tail on the roll axis in synchronisation with the leg motion, but in such a way to counteract the unwanted leg dynamics, the robots yaw angle (heading) was stabilised, as can be seen in figure **??**.

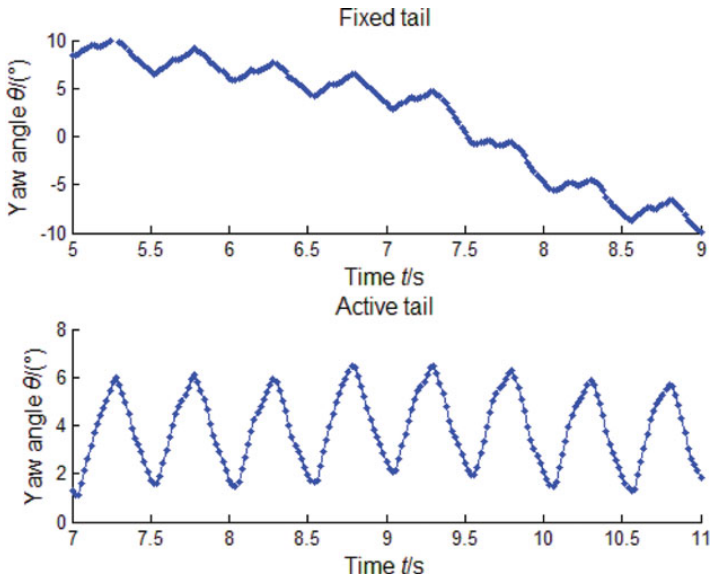


Figure 2.7: Data from [6] showing the effects of a static and dynamic tail on maintaining robot yaw angle in a walking gait.

Ground Contact

2.4.5 Jumping Height

use a tail to increase the apex of a jump. This can be done either by imparting a dynamic force on the robot body (*Inertial*) or striking the ground (*Ground Contact*). This is generally done in sync

Inertial

[7], [8] developed an open-loop system that synchronised with a hopping cycle using only a single actuator. [7] used a rotating disk to adjust tension on 4 wires terminating at each segment of a 4 segment nylon tail with an accordion pattern to allow deformation, while [8] used a single actuator at the base of the tail, and adjustable bidirectional spring joints on 5 additional segments to create a whip-like trajectory.

While [7] failed to increase the jumping height of the robot (attributed to a low tail mass), [8] was able to successfully prove a dynamic tail could increase the jumping height of a robot.

Ground Contact

[9] uses a tail to strike the ground in order to impart a lever action on the body, lifting it into the air.

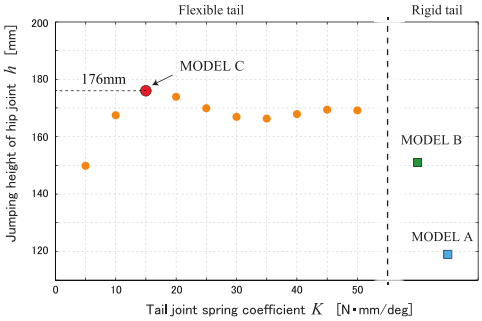


Figure 2.8: Graph from [8] showing how by optimising the spring constant for the spring joints a maximum jump height can be achieved.

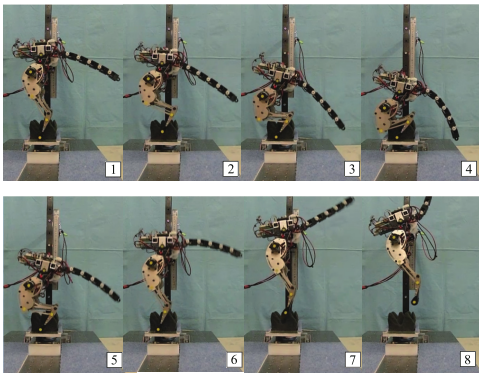


Figure 2.9: Still images from [8] showing the whip-like trajectory of the tail in flight.

2.4.6 Static Model

have developed a robotic in isolation, then measured the dynamics in order to apply it to a function for a mobile robot.

2.5 Bio-Inspiration

2.6 Conclusion and Discussion

References

[1] G. Wenger, A. De, and D. E. Koditschek, "Frontal plane stabilization and hopping with a 2DOF tail," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, IEEE, 2016, pp. 567–573. DOI: 10.1109/IROS.2016.7759110.

[2] T. Libby, T. Y. Moore, E. Chang-Siu, *et al.*, "Tail-assisted pitch control in lizards, robots and dinosaurs," *Nature*, vol. 481, no. 7380, pp. 181–184, 2012. DOI: doi:10.1038/nature10710.

[3] Z. Jianguo, Z. Tianyu, X. Ning, F. J. Cintrón, M. W. Mutka, and X. Li, "Controlling aerial maneuvering of a miniature jumping robot using its tail," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, IEEE, 2013, pp. 3802–3807. DOI: 10.1109/IROS.2013.6696900.

[4] R. Briggs, J.-W. Lee, M. Haberland, and S.-B. Kim, "Tails in biomimetic design: Analysis, simulation, and experiment," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, IEEE, 2012, pp. 1473–1480. DOI: 10.1109/IROS.2012.6386240.

[5] O. V. Borisova, I. I. Borisov, and S. A. Kolyubin, "Analysis and modeling of galloping robot with actuated tail for balance," in *2020 International Conference Nonlinearity, Information and Robotics (NIR)*, IEEE, 2020, pp. 1–6.

[6] Z. Xiuli, G. Jiaqing, and Y. Yanan, "Effects of head and tail as swinging appendages on the dynamic walking performance of a quadruped robot," *Robotica*, vol. 34, no. 12, pp. 2878–2891, 2016. DOI: 10.1017/S0263574716000011.

[7] R. Sato, S. Hashimoto, A. Ming, and M. Shimojo, "Development of a flexible tail for legged robot," in *Mechatronics and Automation (ICMA), 2016 IEEE International Conference on*, IEEE, 2016, pp. 683–688. DOI: 10.1109/ICMA.2016.7558645.

[8] B. Simon, R. Sato, J.-Y. Choley, and A. Ming, "Development of a bio-inspired flexible tail system," in *2018 12th France-Japan and 10th Europe-Asia Congress on Mechatronics*, IEEE, 2018, pp. 230–235.

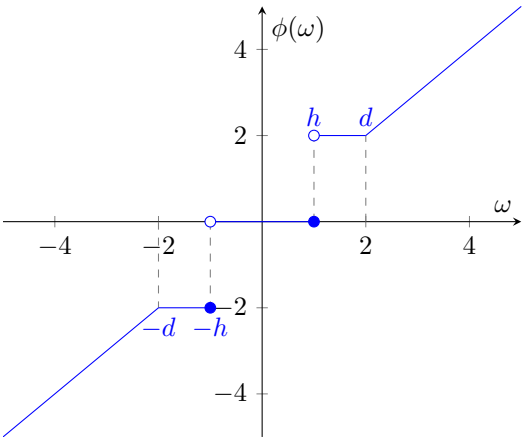
[9] A. L. Brill, A. De, A. M. Johnson, and D. E. Koditschek, "Tail-assisted rigid and compliant legged leaping," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, IEEE, 2015, pp. 6304–6311. DOI: 10.1109/IROS.2015.7354277.

Chapter 3

A Novel Triad TSA! as a Candidate for a 2 Degrees of Freedom Tail Joint

3.1 Introduction

3.2 TSA! (TSA!)



Chapter 4

Creating a Configurable Payload for Instability Experiments

4.1 Introduction

In order to generate a diverse set of test data for the experiments in chapter **??**, a configurable payload was conceived, an object that could be configured to have a wide range of masses and COM. A series of test points can then be generated which have a specific mass and COM, and a matching algorithm can be used to find the configuration that mostly closely matches these parameters. The experiments in chapter **??** can then be run with each of these test points to generate the test data.

4.2 Payload Design

The payload consists of a matrix of cubes of various materials packed tightly into a Three Dimensional (3D) printed container. The cubes are designed to be changed after each experimental run to alter the mass and COM of the payload. A lid on the container prevents the cubes from falling out during the experiment, and the exterior design of the box may accommodate additional features to improving the handling of the payload by the robot arm.

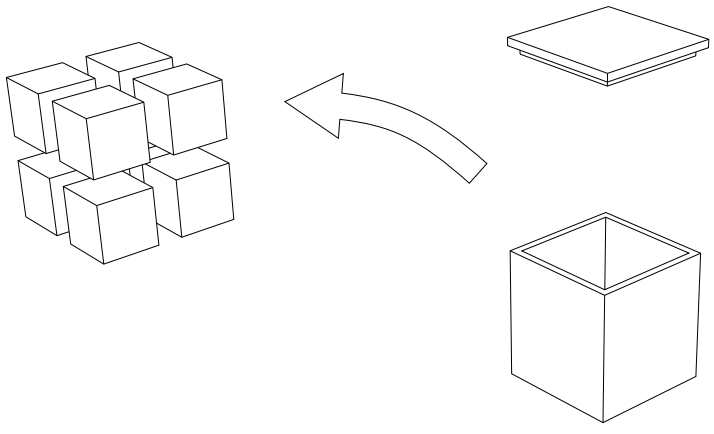


Figure 4.1: Concept drawing of the configurable payload.

Material	Variant	Density (kg m^{-3})
Wood	Pine	860
Plastic	Acrylic	1200
Aluminium	6082	2700
Steel	EN3B	8060

Table 4.1: The materials chosen for the cubes.

4.3 Payload Configuration Space

In order to find a configuration that closely matches a desired test point, first the payload has to be abstracted mathematically, so the mass and COM can be calculated for a given configuration. Firstly we can consider a positive real set of material densities $\mathcal{P} \in \mathbb{R}^+$, each element the density (in kg m^{-3}) of a material to be used:

$$\mathcal{P} = \{\rho_1, \rho_2 \dots \rho_n \mid \rho_i > 0\} \tag{4.1}$$

Each configuration can then be defined as an $n \times n \times n$ matrix $\boldsymbol{C}$, such that each element is an element of $\mathcal{P}$, where n^3 is the number of cubes in the matrix:

$$\boldsymbol{C} = (c_{ijk}) \in \mathbb{R}^{n^n} \mid (c_{ijk}) \in \mathcal{P} \tag{4.2}$$

4.3.1 Mass and COM functions

To calculate the mass of the configuration, we can take the sum of all the cube densities multiplied by their volume a^3, where a is the cube edge length, plus the container mass m_c:

$$M(\boldsymbol{C}) = \left(\sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} c_{ijk} a^3 \right) + m_c \tag{4.3}$$

To calculate the COM, we can take the sum of each cube mass multiplied by its position relative to the centroid of the center cube (c_{222}), which can be calculated from the cube indexes ijk, plus the container COM $\boldsymbol{r}_c$ if non-zero:

$$R(\boldsymbol{C}) = \frac{\left(\sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} c_{ijk} a^4 \left(\begin{bmatrix} i & j & k \end{bmatrix} - n + 1 \right) \right) + \boldsymbol{r}_c}{M(\boldsymbol{C})} \tag{4.4}$$

4.4 Test Points

With $\mathcal{Z}$ as the set of all permutations of $\boldsymbol{C}$, test points can be derived from subsets of $\mathcal{Z}$ defined either by logical expressions on $\mathcal{Z}$, or the nearest neighbours of $\mathcal{Z}$ from a set of co-

ordinates. A test point is defined as a $\mathbb{R}^4$ vector containing the target mass and COM concatenated as $\begin{bmatrix} m_i & \boldsymbol{r}_i \end{bmatrix}$.

4.4.1 Extrema Set ($\mathcal{E}$)

The extrema set is designed to test the extremas of the space of $\mathcal{Z}$ for both $M(\mathcal{Z})$ and $R(\mathcal{Z})$. The extrema set if defined from a set of logical constraints. The first two constraints of the set find the maximum and minimum values of the payload mass using $M(\mathcal{Z})$, and the next four constraints use the payload COM using $M(\mathcal{C})$ to get the maximum and minimum values of the x and y component of the COM. Finally, the last four constraints define the diagonal maximum and minimum values where the COM components match $x = y$ or $x = -y$.

$$\mathcal{E} = \left\{ \boldsymbol{x} \subset \mathcal{Z} \mid \left\{ \begin{array}{l} M(\boldsymbol{x}) = \max\{M(\mathcal{Z})\} \\ M(\boldsymbol{x}) = \min\{M(\mathcal{Z})\} \\ R(\boldsymbol{x})_x = \max\{R(\mathcal{Z})_x\} \\ R(\boldsymbol{x})_x = \min\{R(\mathcal{Z})_x\} \\ R(\boldsymbol{x})_y = \max\{R(\mathcal{Z})_y\} \\ R(\boldsymbol{x})_y = \min\{R(\mathcal{Z})_y\} \\ R(\boldsymbol{x})_x = \max\{R(\mathcal{Z})_x\} \wedge R(\boldsymbol{x})_x = R(\boldsymbol{x})_y \\ R(\boldsymbol{x})_x = \min\{R(\mathcal{Z})_x\} \wedge R(\boldsymbol{x})_x = R(\boldsymbol{x})_y \\ R(\boldsymbol{x})_x = \max\{R(\mathcal{Z})_x\} \wedge R(\boldsymbol{x})_x = -R(\boldsymbol{x})_y \\ R(\boldsymbol{x})_x = \min\{R(\mathcal{Z})_x\} \wedge R(\boldsymbol{x})_x = -R(\boldsymbol{x})_y \end{array} \right\} \right. \tag{4.5}$$

Mass Limited $\mathcal{E}$

When this set was generated, it was found that $M(\mathcal{E}_1)$ was greater than the chosen robot arm could safely lift. Therefore, $M(\boldsymbol{x}) = \max\{M(\mathcal{Z})\}$ in $\mathcal{E}$ was changed to $\begin{bmatrix} m_{max} & 0 & 0 & 0 \end{bmatrix}$ where m_{max} is the safe mass limit that the robot arm can lift. One of the search methods described in section 4.5 can then be used to find the nearest point in configuration space.

4.4.2 Cube Set ($\mathcal{C}$)

The cube set is defined by the vertices of a cube of size b centred around the COM origin $\begin{bmatrix} 0 & 0 & 0 \end{bmatrix}$.

$$\mathcal{C} = \left\{ \boldsymbol{x} \subset \mathcal{C} \mid \begin{bmatrix} \pm\frac{b}{2} & \pm\frac{b}{2} & \pm\frac{b}{2} \end{bmatrix} = \boldsymbol{x} \right\} \tag{4.6}$$

4.4.3 Balanced Set ($\mathcal{B}$)

The balanced set is defined by q points in $\mathcal{C}$ subject to the constraint $R(\boldsymbol{x})_x = 0 \wedge R(\boldsymbol{x})_y = 0$. This can be defined as a "balanced" set as the COM x and y components are both zero. The points are evenly spaced between the maximum and minimum mass as defined in section 4.4.1.

$$\begin{aligned} m_r &= \frac{\max\{M(\mathcal{Z})\} - \min\{M(\mathcal{Z})\}}{q+1} \\ \boldsymbol{z} &= \begin{bmatrix} m_r & 2m_r & \cdots & qm_r \end{bmatrix} \\ \mathcal{B} &= \{\boldsymbol{x} \subset \mathcal{C} \mid z_i = \boldsymbol{x} \wedge R(\boldsymbol{x})_x = 0 \wedge R(\boldsymbol{x})_y = 0\} \end{aligned} \tag{4.7}$$

4.5 Test Point Matching Search Methods

While we are guaranteed an exact result for elements of $\mathcal{E}$ as every element is defined by constraints on known configurations, for $\mathcal{C}$ and $\mathcal{B}$ as the elements are defined numerically, there is no guarantee that any element will have an exact match in solution space. The cardinality of $\mathcal{Z}$ is also important. It is defined as $|\mathcal{Z}| = |\mathcal{P}|^{n^n}$ which increases super exponentially with n. For example, when $|\mathcal{P}| = 4$, $n = 2$ results in 256 permutations and $n = 3$ results in approximately 1.8×10^{16} permutations. It's very clear that when $n > 2$ for non-trivial cardinalities of $\mathcal{P}$, any kind of brute-force method is not computationally tractable. The only exception is for $\mathcal{E}_1$ and $\mathcal{E}_2$, where the solution is trivial as $\mathcal{E}_1 = \boldsymbol{C} \mid \min(\mathcal{P}) \forall c_{ijk}$ and $\mathcal{E}_2 = \boldsymbol{C} \mid \max(\mathcal{P}) \forall c_{ijk}$. Therefore, we investigated both a brute-force nearest neighbour method for when $n = 2$, and a simulated annealing method for when $n = 3$.

4.5.1 Simulated Annealing ($n = 3$)

Simulated annealing [1] is a modification to a gradient descent optimisation that allows the algorithm the chance to "jump out" of local minima early on (even though the approximation becomes temporarily worse). However, as the number of remaining steps decreases, that probability becomes smaller, becoming more and more like gradient descent. First, like any gradient descent algorithm, two things need to be generated, the initial configuration $\boldsymbol{C_0}$, which can be random or manually selected, and the function $\mathcal{N}(\boldsymbol{C})$ which creates a set of all the "neighbours" of $\boldsymbol{C}$. In this case, this can be defined as the subset of $\mathcal{Z}$ where the difference between $\boldsymbol{C}$ and an element of $\mathcal{N}(\boldsymbol{C})$ is one and only one $c_{ijk} \neq c_{ijk}$:

$$\mathcal{N}(\boldsymbol{C}) = \{x \subset \mathcal{Z} \mid \exists!\,(x_{ijk} \neq c_{ijk})\} \tag{4.8}$$

Then the simulated annealing function can be described as follows:

1. Set $\boldsymbol{C}$ to the initial permutation $\boldsymbol{C_0}$.

2. For each of the optimisation steps:

   (a) Set the temperature value t with function $T\left(\frac{k_{max}}{k}\right)$ which takes into account the number of remaining steps.

   (b) Set $\boldsymbol{C}_{new}$ as a random element from the set of all neighbours of $\boldsymbol{C}$ as defined by $\mathcal{N}(\boldsymbol{C})$.

   (c) Use acceptance probability function $P\left(E\left(\boldsymbol{C}\right), E\left(\boldsymbol{C}_{new}\right), t\right)$ where $E\left(\boldsymbol{C}\right)$ is the energy function (in this case the absolute difference $E\left(\boldsymbol{C}, g\right) = \left|M\left(\boldsymbol{C}\right) - g\right|$ or $E\left(\boldsymbol{C}, g\right) = \left|R\left(\boldsymbol{C}\right) - g\right|$ where g is the target could be used to look for a lower energy state in either mass or COM) to generate a value. Note this function is dependent on the temperature t.

   (d) Compare that value with a random uniformly distributed real number between 0 and 1. If greater than or equal to, then replace $\boldsymbol{C}$ with $\boldsymbol{C}_{new}$. Otherwise, keep it the same.

   (e) Repeat with $\boldsymbol{C}$ until there are no remaining steps.

3. Return the approximated permutation $\boldsymbol{C}$.

$\boldsymbol{C} = \boldsymbol{C}_0$
for $k \leftarrow 1, k_{max}$ **do**
  $t = T\left(\frac{k_{max}}{k}\right)$
  $\boldsymbol{C}_{new} = \mathcal{N}\left(\boldsymbol{C}\right) \xleftarrow{R} x$
  if $P\left(E\left(\boldsymbol{C}\right), E\left(\boldsymbol{C}_{new}\right), t\right) \geq x \sim U\left([0,1]\right)$ **then**
    $\boldsymbol{C} = \boldsymbol{C}_{new}$
  end if
end for
return $\boldsymbol{C}$

Using this algorithm an array of approximate configurations can be easily generated from a desired array of test points. Simulated annealing can also be adapted for multi-objective optimisation [2], so it is possible to generate test points that approximate a desired mass and COM simultaneously.

Cooling Function

The function which controls the probability of exiting local minima (known as the *temperature*) is known as the *cooling function*. This function can be any function which monotonically decreases (except in adaptive simulated annealing where it is dependant on the accuracy of the current approximation). Different functions will result in a different cooling profile, generally decreasing quickly in the first few steps, and then slowing down after that.

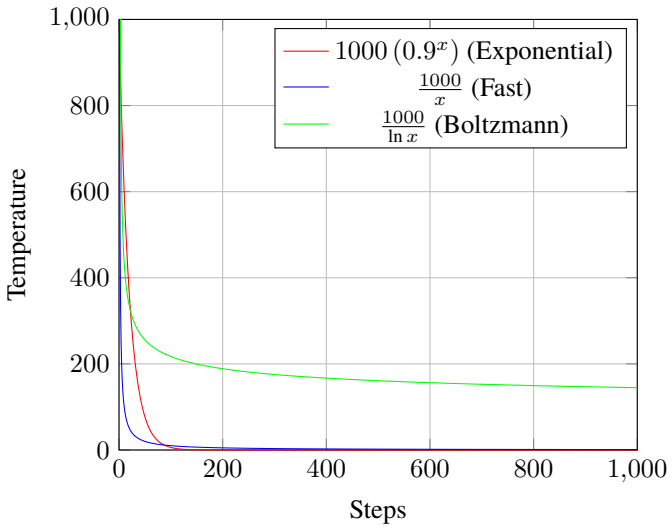


Figure 4.2: Various temperature cooling profiles for simulated annealing, assuming 1000 steps.

4.5.2 Brute-Force Nearest Neighbour ($n = 2$)

If $\mathcal{Z}$ is suitably small, as it is when $n = 2$ then a brute-force method can be used which is guaranteed to find the nearest element to the target within a finite time. This can be done by calculating the L2 norms between the target vector t and all the elements of $\mathcal{Z}$ and finding the minimum. If several elements of $\mathcal{Z}$ have the minimum norm, then one is chosen at random from this set.

$$NN\left(t, \mathcal{Z}\right) = \min\left\{\left\|t - x\right\|_2 \mid \forall x \in \mathcal{Z}\right\} \tag{4.9}$$

4.5.3 Selected Method

Initially an $n = 3$ configuration was used with the acceptance probability function *Rule M* from [2]. This is a weighted blend of two other algorithms defined in the paper, *Rule P* and *Rule W* with a weighting coefficient $\alpha \in (0, 1) \subset \mathbb{R}$. There is also a weighting vector for each element of the test point $\boldsymbol{w} \in \mathbb{R}^4 \mid w_i \in (0, 1)$.

$$P\left(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{w}, t\right) = \alpha \underbrace{\prod_{i=1}^{m} \min\left\{1, e^{\frac{w_i(x_i - y_i)}{t}}\right\}}_{\text{Rule P}} + (1 - \alpha) \underbrace{\min\left\{1, \max_{i=1,\ldots,m}\left\{1, e^{\frac{w_i(x_i - y_i)}{t}}\right\}\right\}}_{\text{Rule W}} \tag{4.10}$$

Unfortunately it was difficult to find a stable and consistent result even after a long time running the algorithm.

Therefore as an alternative, the $n = 2$ configuration was used, with larger cubes to compensate. This successfully produced all three test point sets, after some small adjustments of b in order to get closer to a single configuration for each point of $\mathcal{C}$.

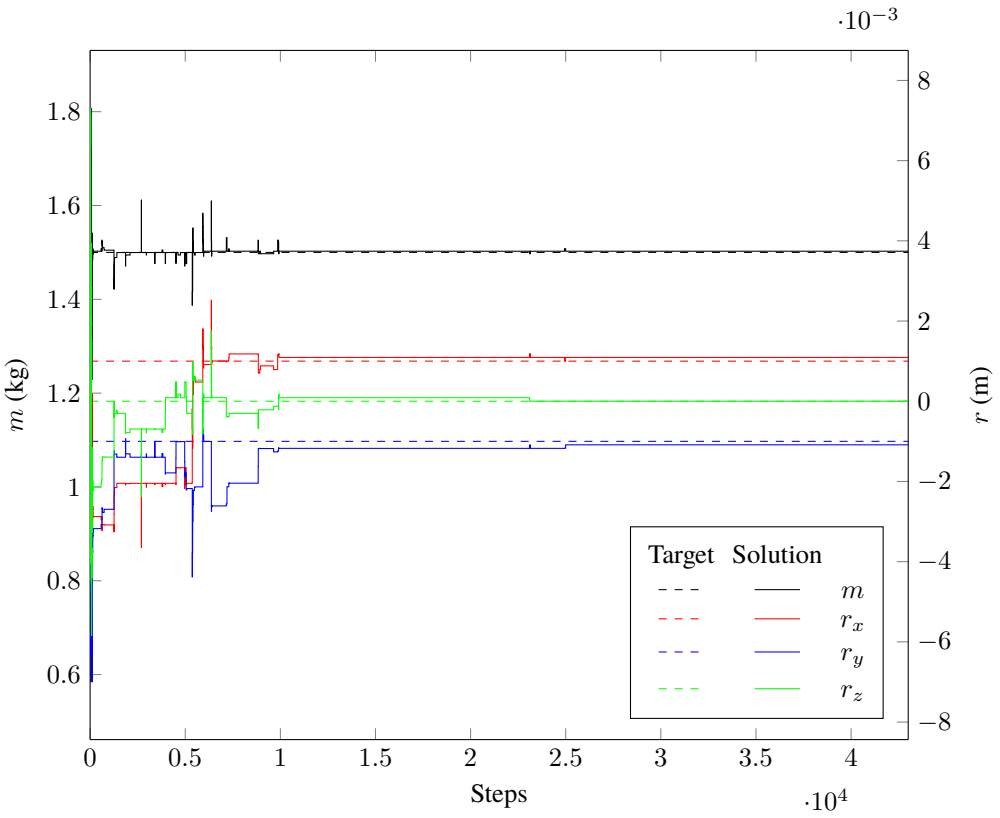


Figure 4.3: Simulated annealing output for the target $[1.5 \quad 0.001 \quad 0.001 \quad 0.001]$ with $\alpha = 0.997$ and even weighting $w_m, \boldsymbol{w_r} = 0.25$ for 4.3×10^4 steps.

m	$\boldsymbol{r}$
Extrema Set ($\mathcal{E}$)	
1.751	$[0.012 \quad 0.000 \quad 0.000]$
1.751	$[0.000 \quad 0.012 \quad 0.000]$
1.751	$[-0.012 \quad 0.000 \quad 0.000]$
1.751	$[-0.000 \quad -0.012 \quad 0.000]$
0.516	$[0.000 \quad 0.000 \quad 0.000]$
1.133	$[0.010 \quad 0.010 \quad 0.000]$
1.133	$[-0.010 \quad -0.010 \quad 0.000]$
1.133	$[0.010 \quad -0.010 \quad 0.000]$
1.133	$[-0.010 \quad 0.010 \quad 0.000]$

Table 4.2: Table of the vectors of $\mathcal{E}$, exclusing the mass-limited element.

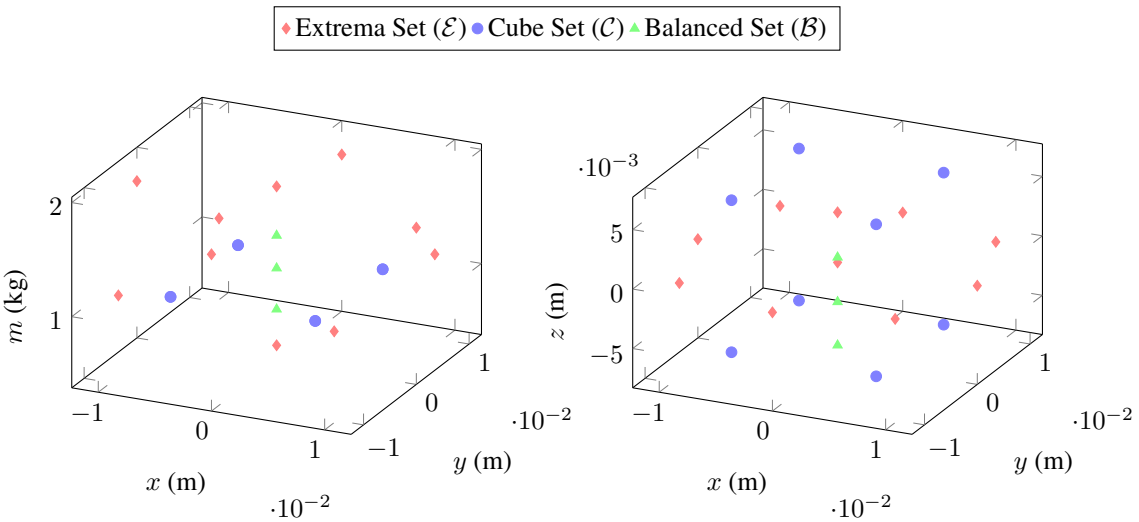


Figure 4.4: Mass and COM coordinates for each test point set.

Target		Nearest		L2 Norm Error
m	$\boldsymbol{r}$	m	$\boldsymbol{r}$	
Extrema Set ($\mathcal{E}$)				
2.000	$\begin{bmatrix}0.000 & 0.000 & 0.000\end{bmatrix}$	1.909	$\begin{bmatrix}0.000 & 0.000 & 0.004\end{bmatrix}$	9.154×10^{-2}
Cube Set ($\mathcal{C}$)				
$*$	$\begin{bmatrix}-0.007 & -0.007 & -0.007\end{bmatrix}$	1.061	$\begin{bmatrix}-0.006 & -0.006 & -0.006\end{bmatrix}$	1.055×10^{-3}
$*$	$\begin{bmatrix}-0.007 & -0.007 & 0.007\end{bmatrix}$	1.061	$\begin{bmatrix}-0.006 & 0.006 & -0.006\end{bmatrix}$	1.895×10^{-2}
$*$	$\begin{bmatrix}-0.007 & 0.007 & -0.007\end{bmatrix}$	1.061	$\begin{bmatrix}0.006 & -0.006 & -0.006\end{bmatrix}$	1.895×10^{-2}
$*$	$\begin{bmatrix}-0.007 & 0.007 & 0.007\end{bmatrix}$	1.061	$\begin{bmatrix}0.006 & 0.006 & -0.006\end{bmatrix}$	1.895×10^{-2}
$*$	$\begin{bmatrix}0.007 & -0.007 & -0.007\end{bmatrix}$	1.061	$\begin{bmatrix}-0.006 & -0.006 & 0.006\end{bmatrix}$	1.895×10^{-2}
$*$	$\begin{bmatrix}0.007 & -0.007 & 0.007\end{bmatrix}$	1.061	$\begin{bmatrix}-0.006 & 0.006 & 0.006\end{bmatrix}$	1.895×10^{-2}
$*$	$\begin{bmatrix}0.007 & 0.007 & -0.007\end{bmatrix}$	1.061	$\begin{bmatrix}0.006 & -0.006 & 0.006\end{bmatrix}$	1.895×10^{-2}
$*$	$\begin{bmatrix}0.007 & 0.007 & 0.007\end{bmatrix}$	1.061	$\begin{bmatrix}0.006 & 0.006 & 0.006\end{bmatrix}$	1.055×10^{-3}
Balanced Set ($\mathcal{B}$)				
0.516	$\begin{bmatrix}0.000 & 0.000 & *\end{bmatrix}$	0.832	$\begin{bmatrix}0.000 & 0.000 & -0.003\end{bmatrix}$	3.156×10^{-1}
1.258	$\begin{bmatrix}0.000 & 0.000 & *\end{bmatrix}$	1.192	$\begin{bmatrix}-0.000 & -0.000 & 0.000\end{bmatrix}$	6.630×10^{-2}
2.000	$\begin{bmatrix}0.000 & 0.000 & *\end{bmatrix}$	1.478	$\begin{bmatrix}-0.000 & -0.000 & -0.007\end{bmatrix}$	5.219×10^{-1}

Table 4.3: Table of the target and actual vectors for $\mathcal{C}$, $\mathcal{B}$ and the mass limited element of $\mathcal{E}$ with the L2 norm error. $*$ notation indicates "don't care" and is excluded from the search algorithm.

4.6 Conclusion and Discussion

References

[1] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *science*, vol. 220, no. 4598, pp. 671–680, 1983.

[2] P. Serafini, "Simulated annealing for multi objective optimization problems," in *Multiple criteria decision making*, Springer, 1994, pp. 283–292.

Chapter 5

Creating a LabVIEW™ Driver for the Bosch Sensortec BNO080 Inertial Measurment Unit

5.1 Introduction

In order to measure the pitch and roll of the Actuated Universal Joint (AUJ) in chapter **??**, and the pitch and roll of the tail in chapter **??**, a sensor was required that could measure orientation in at least two axes with a fair degree of precision. Considering cost, footprint and ease of integration, the *Bosch Sensortec BNO080 IMU* was chosen for this purpose. This is an IMU that is used in VR headsets and mobile phones, among other applications, and has a wide range of output data, from raw accelerometer data to step counters. It achieves this by combining data from 3 hardware sensors, a 3 axis accelerometer, gyroscope and magnetometer, to produce 9 streams of sensor data, which an integrated 32-bit *ARM® Cortex™-M0+* processor fuses to generate the output data.

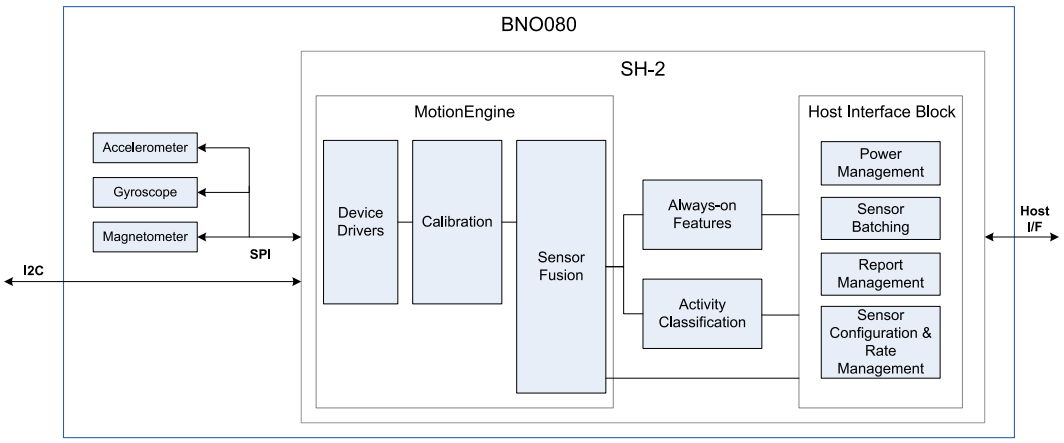


Figure 5.1: The BNO080 IC architecture [1]. Note the I^2C **IO!** is only for attaching specific additional sensors.

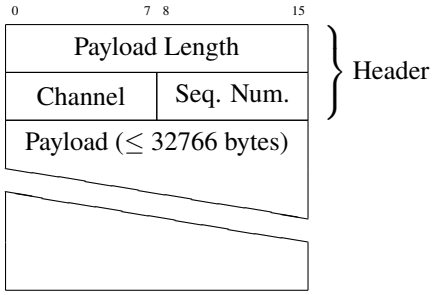


Figure 5.2: An SHTP packet [2].

5.2 Device Architecture

The BNO080 can be thought of as having a simple "layered architecture", made up of two layers, the *communication layer* and the *software layer*. The communication layer is a basic packet based communication protocol used to send messages, known as "payloads", between the host and the device. The software layer then reads the payload and uses the information to perform an action on the BNO080 or recieve data on the host, depending on the recipient.

5.2.1 Communication Layer: SHTP

The BNO080 uses a packet based communication system called the Sensor Hub Transport Protocol. This can be used over I^2C, SPI and UART interfaces. The SHTP packet consists of a header defining the byte length of the payload data, the channel the payload is to be sent to or has been received from, and a sequence number which monotonically increases which each complete payload sent (used to send payloads greater than the maximum byte length).

The BNO080 has 6 channels, though only 3 are currently used in this driver:

Channel	Function	Used?
0	SHTP Commands	✗
1	Sleep, Wake & "Soft Reset"	✓
2	Device Commands	✓
3	Sensor Data	✓
4	Sensor Data (Wake Trigger)	✗
5	Gyro Rotation Vector	✗

5.2.2 Software Layer: SH-2

SH-2 is the software that runs on the BNO080, and provides interfaces for configuring the device, requesting sensor data, and receiving sensor and configuration data.

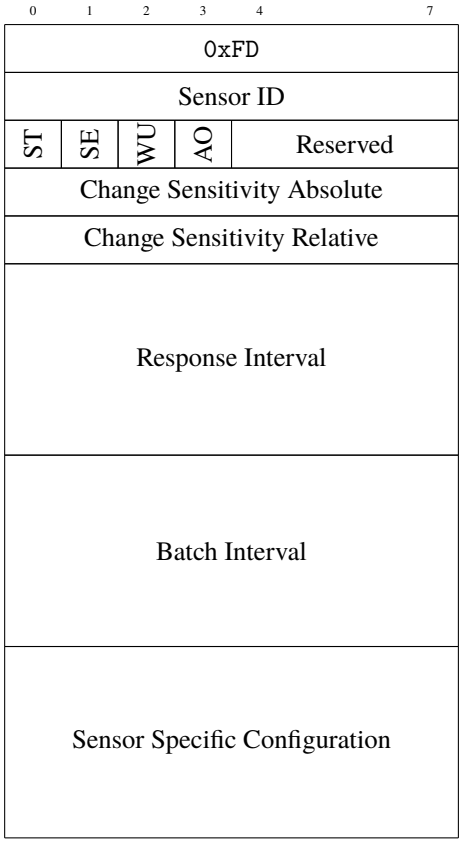


Figure 5.3: Set Feature Command [3]

Sensor Data

In order to receive sensor data from the BNO080, *Get Feature Command (0xFD)* command must be sent, which contains the following fields:

Field	Bit Length	Description
Sensor ID	8	The ID of the sensor to recieve data from.
Change Senstivity Type (ST)	1	Absolute (0) or relative (1). Only read if Senstivity Enable (SE) is enabled.
Senstivity Enable (SE)	1	Enables change sensitvity.
Wake up Enable (WU)	1	Allows the sensor to wake up the BNO080.
Always On (AO)	1	Sends data even if the BNO080 is asleep.
Change Sensitivity Absolute	8	Absolute change sensitivity value.
Change Sensitivity Relative	8	Relative change sensitivity value.
Response Interval	32	Frequency at which the BNO80 will take sensor readings.
Batch Interval	32	Frequency at which the BNO80 will send batched data to the controller (same as response interval if set to 0).
Sensor Specific Configuration	32	Unique configuration data for each sensor.

Table 5.1: Set Feature Command fields.

Once sent, this command will return a *Get Feature Command (0xFD)* once, then stream out batched *Input Report* responses at the interval specified in the request. Change sensitivity allows the sensor to only send data if it exceeds a certain threshold, either an absolute value or a relative value compared to the previous reading. All readings are sent to the controller in a "batch" format, where a list of samples are grouped together with a timestamp header

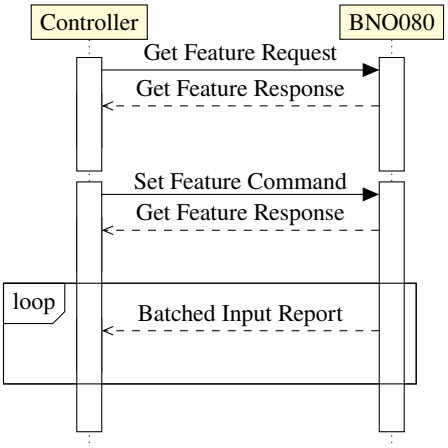


Figure 5.4: Communication diagram of the sensor data commands and responses.

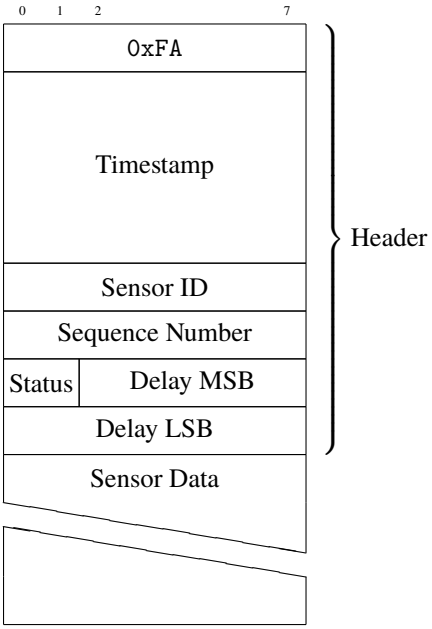


Figure 5.5: Generic batched *Input Report* with a single data record [3]

specified by the interval value. For a simple implementation, as in this driver, change sensitivity is disabled, so the samples are always sent regardless of value, and there is no batch interval, so a single sample is sent to the controller immediately once read by the BNO080.

Field	Bit Length	Description
Timestamp	32	Timestamp in µs.
Sensor ID	8	The ID of the sensor which the data comes from.
Sequence Number	8	A monotonically increasing number that increments with every report sent.
Status ID	8	The calibation accuracty of the sensor (0 - Unreliable, 1 - Low, 2 - Medium, 3 - High).
Delay	14	Delay from Timestamp in 100 µs units (useful when there are multiple samples).

Table 5.2: Batched *Input Report* header fields for a single sample.

The sensor data itself can take on a variety of formats, but for continuous sensor data as a result of one or more of the 3 hardware sensors has one of two formats, a quaternion for a calculated orientation, or a 3 element vector for plain sensor data. The quaternion can also

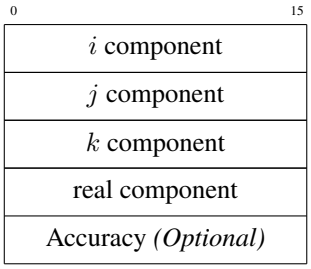


Figure 5.6: Quaternion report structure.

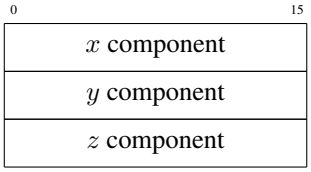


Figure 5.7: 3 element vector report structure.

optionally have an accuracy estimate.

5.3 LabVIEW GOOP Implementation

Since the BNO080 communication protocol is not stateless due to the increasing sequence numbers, a object oriented solution was chosen using the LabVIEW™ Graphical Object Oriented Programming (GOOP) library. This would construct a BNO080 object, which could then have methods to configure the device and get sensor data, with the sequence numbers as mutable properties. This also allowed flexibility in communication method (I^2C, SPI, UART), since each one could be a subclass of an inherited "communication" class which would be a property of the BNO080 object. Multiple BNO80 devices could also be easily implemented, simply by constructing another BNO80 class with a different communication configuration.

Each BNO080 device is represented by a *SH-2* object, which contains a *SHTP* object as a protocol, and a list of *Sensor* objects for each output record on the BNO080. There are also counters for sent and received *Commands*, and received *Input Reports* which hold the sequence numbers, plus a boolean flag for the *Calibration Status* of the device. This object contains all the methods and functions LabVIEW needs to interact with the device.

The *SHTP* object contains a list of *Channel* objects with packet send and receive counters and ID numbers, and the inheritable *Communication* class for the hardware protocol selected.

Each *Sensor* object has an ID which is the sensor ID,

Overall, the library was designed with. Large amounts of the structure were adapted from [4], which is an Arduino implementation for interacting with the BNO080.

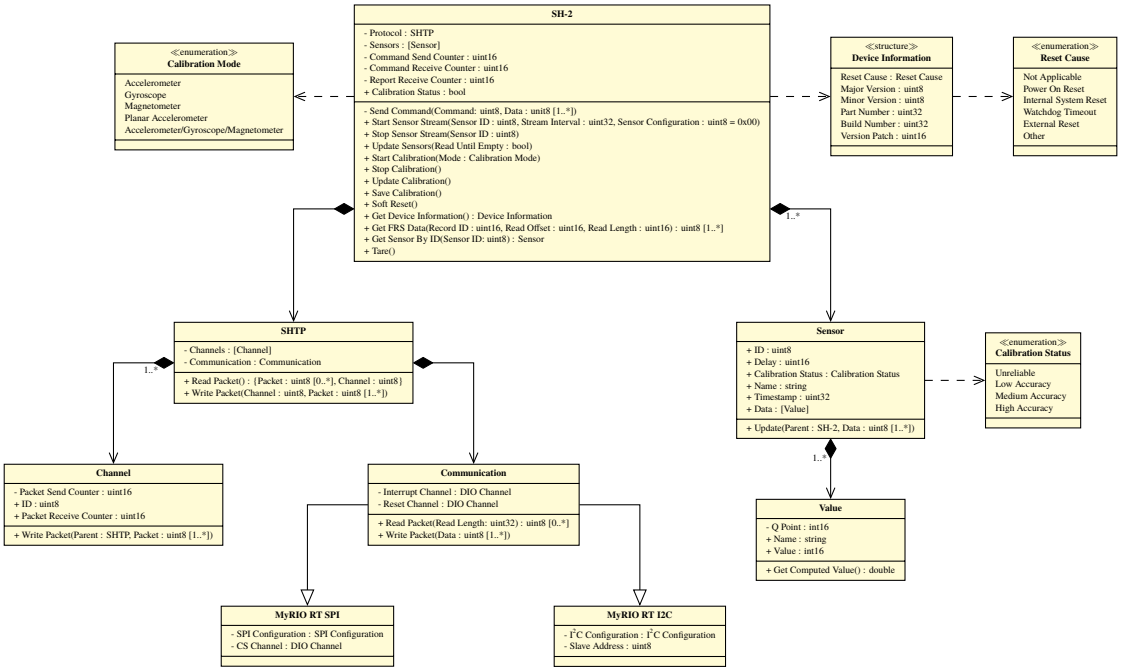


Figure 5.8: UML diagram of the BNO080 GOOP library.

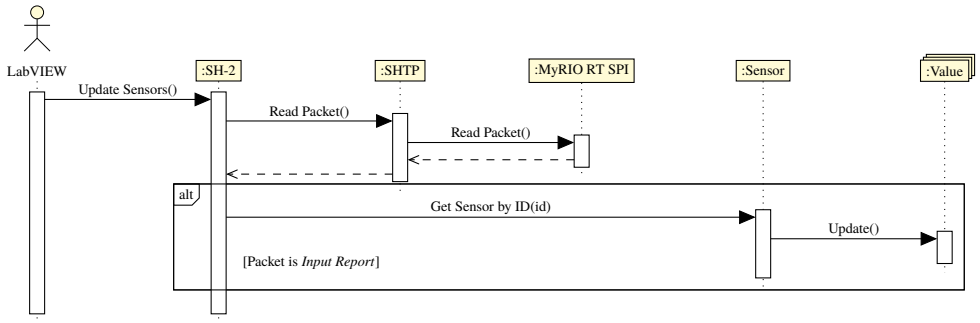


Figure 5.9: Communication diagram for updating a sensor record.

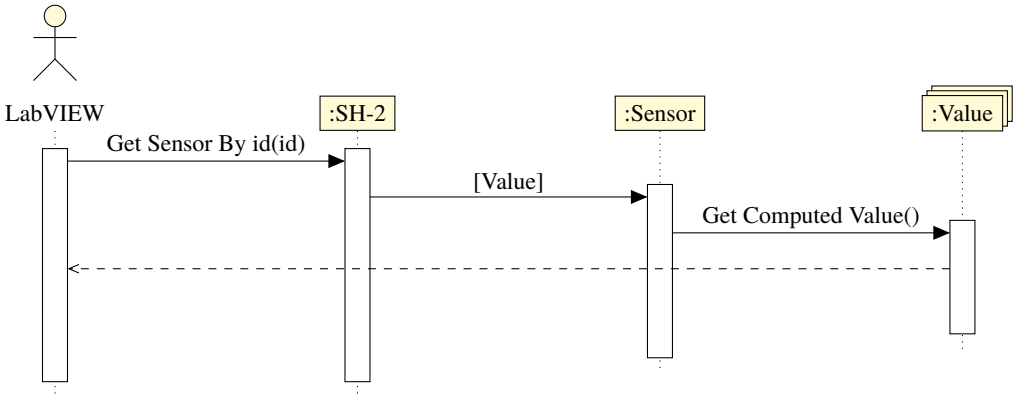


Figure 5.10: Communication diagram for reading a sensor record.

5.3.1 Library Structure

5.3.2 Implementation for Reading Sensor

5.4 Conclusion and Discussion

The driver in its current state is only a very partial implementation of the full feature set of the BNO080, however it is sufficient for the data required from the device within the scope of the research. The author plans to release the driver on GitHub as a public repository, where it could be further expanded by other users in the future.

References

[1] *BNO08X data sheet*, version 1.8, Hillcrest Laboratories, Inc., 2020.

[2] *Sensor hub transport protocol*, version 1.7, Hillcrest Laboratories, Inc., Feb. 2017.

[3] *SH-2 reference manual*, version 1.2, Hillcrest Laboratories, Inc., May 2017.

[4] SparkFun, *SparkFun BNO080 arduino library*, 2021. [Online]. Available: `https://github.com/sparkfun/SparkFun_BNO080_Arduino_Library`.

Chapter 6

Creating a Configurable Payload for Instability Experiments

6.1 Introduction

6.2 Control Experiments

6.2.1 Results

6.3 Conclusion and Discussion

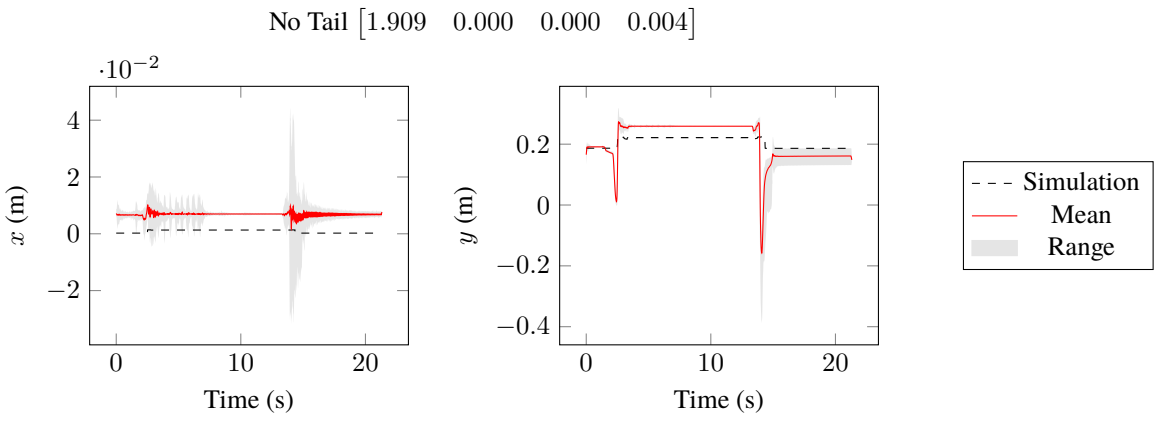


Figure 6.1: COM x and y position of the test rig along the test trajectory for the mass maximum element of the extrema set with no tail.

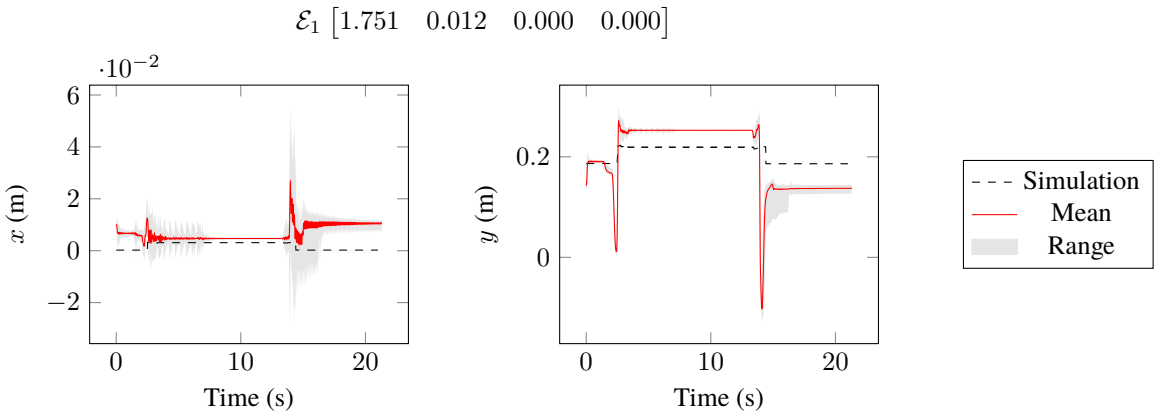


Figure 6.2: COM x and y position of the test rig along the test trajectory for the mass minimum element of the extrema set with no tail.

Chapter 7

Optimisation Study for Multi-Segment Tails for Centre of Mass Control

7.1 Introduction

One of the most noticeable differences between robotic and animal tails, as discovered in chapter **??**, is the far greater number of segments in most animal tails when compared to robot tails.

7.2 Model Definition

We can consider a tail with an arbitrary number of segments as a chain of bodies connected by revolute joints, where $\boldsymbol{l}$ are the lengths of the bodies, $\boldsymbol{m}$ are the masses of the bodies, $\boldsymbol{l_c}$ are the offsets of the COM from the origin of each body along the axis of the chain (assuming the COM remains on that axis for the sake of simplicity), and $\boldsymbol{\theta}$ are the angles of each revolute joint.

The transformation matrices for a given body's origin and COM can then be computed.

$$
\begin{aligned}
T_i &= \begin{bmatrix} \cos\theta_i & -\sin\theta_i & l_i \\ \sin\theta_i & \cos\theta_i & 0 \\ 0 & 0 & 1 \end{bmatrix} \\
T_{c_i} &= \begin{bmatrix} \cos\theta_i & -\sin\theta_i & l_{c_i} \\ \sin\theta_i & \cos\theta_i & 0 \\ 0 & 0 & 1 \end{bmatrix}
\end{aligned}
\tag{7.1}
$$

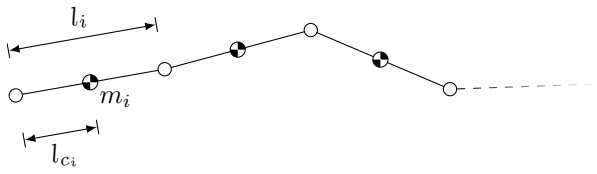


Figure 7.1: Diagram of a 2D tail, with all parameters annotated.

The forward kinematics for the tail is then computed. This will give us the position of the endpoint of the tail for a given set of joint angles $\boldsymbol{\theta}$.

$$T(\boldsymbol{\theta}) = \prod_{i=1}^{n} T_i \tag{7.2}$$

A similar equation can be used for the forward *kinetics*, with the addition of the body masses and COM offsets as parameters. This will give us the position of the COM for the entire tail.

$$R(\boldsymbol{\theta}) = \frac{\sum_{i=1}^{n} m_i \prod_{j=1}^{i-1} (T_j)\, T_{c_i}}{\sum_{i=1}^{n} m_i} \tag{7.3}$$

7.3 Inverse Kinetics

As we can compute the forward kinetics in a very similar fashion to the inverse kinematics, it follows that the inverse kinetics can be computed in a similar fashion. Firstly we define the Jacobian.

$$\boldsymbol{J}_R = \begin{bmatrix} \frac{\partial r_{13}}{\partial \theta_1} & \frac{\partial r_{13}}{\partial \theta_1} & \cdots & \frac{\partial r_{13}}{\partial \theta_n} \\ \frac{\partial r_{23}}{\partial \theta_1} & \frac{\partial r_{23}}{\partial \theta_1} & \cdots & \frac{\partial r_{23}}{\partial \theta_n} \end{bmatrix} \tag{7.4}$$

Then, assuming a COM position defined by $\begin{bmatrix} q_x & q_y \end{bmatrix}^\intercal$ the COM velocity can be calculated using the Jacobian.

$$\begin{bmatrix} \dot{q}_x \\ \dot{q}_y \end{bmatrix} = \boldsymbol{J}_R \cdot \dot{\theta} \tag{7.5}$$

To get the joint velocities (or torques) for a given COM velocity, we can use any inverse kinematic algorithm, treating the COM in a similar fashion to a 3-DOF end effector. The COM velocity can be calculated from a Proportional Integral Derivative (PID) controller which has the COM error as an input.

7.4 Inverse Kinematic Algorithms

Three different inverse kinematic algorithms were chosen for the experiment. Two used a weighting factor that allowed for both the joint velocity and torque to be minimised, the other only minimised the joint velocity but was much simpler to implement.

7.4.1 Damped Least Squares

Damped Least Squares (DLS), also known as Levenberg-Marquardt [1], is the simplest algorithm to implement. It is an improvement on the pseudoinverse or transpose method, which have issues with singularities [1].

$$\dot{\boldsymbol{\theta}} = \boldsymbol{J}^{\intercal}\left(\boldsymbol{J}\boldsymbol{J}^{\intercal}\lambda^{2}\boldsymbol{I}\right)\begin{bmatrix}\dot{q}_x\\ \dot{q}_y\end{bmatrix} \tag{7.6}$$

Where $\boldsymbol{I}$ is the identity matrix and λ is a suitable damping constant.

While it is easy to implement, DLS can only minimise the joint velocities, expressed as the euclidean norm $\|\dot{\boldsymbol{\theta}}\left(t\right)\|_2$.

7.4.2 A Balancing Technique to Stabilize Local Toque Optimization Solution of Redundant Manipulators [2]

This is an expansion on the pseudeoinverse method that allows for a weighted

7.4.3 Different-Level Simultaneous Minimization of Joint-Velocity and Joint-Torque for Redundant Robot Manipulators [3]

$$\begin{aligned}
\boldsymbol{\tau} &= D\left(\boldsymbol{\theta}\right)\ddot{\boldsymbol{\theta}} + C\left(\boldsymbol{\theta},\dot{\boldsymbol{\theta}}\right) + G\left(\boldsymbol{\theta}\right)\\
\boldsymbol{Q} &= \alpha\boldsymbol{I} + (1-\alpha)D\left(\boldsymbol{\theta}\right)^{2}\\
\boldsymbol{p} &= \alpha\lambda\dot{\boldsymbol{\theta}} + (1-\alpha)D\left(\boldsymbol{\theta}\right)^{\intercal}\left(C\left(\boldsymbol{\theta},\dot{\boldsymbol{\theta}}\right) + G\left(\boldsymbol{\theta}\right)\right)\\
\ddot{\boldsymbol{\theta}} &= \min_{\ddot{\boldsymbol{\theta}}}\frac{1}{2}\ddot{\boldsymbol{\theta}}^{\intercal}\boldsymbol{Q}\ddot{\boldsymbol{\theta}} + \boldsymbol{p}^{\intercal}\ddot{\boldsymbol{\theta}} \text{ s.t. } \left\{\boldsymbol{J}\ddot{\boldsymbol{\theta}} = \ddot{x}_1 - \dot{\boldsymbol{J}}\dot{\boldsymbol{\theta}}\right.
\end{aligned} \tag{7.7}$$

7.5 Results

7.6 Conclusion and Discussion

References

[1] S. R. Buss, "Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods," *IEEE Journal of Robotics and Automation*, vol. 17, no. 1-19, p. 16, 2004.

[2] S. Ma, "A balancing technique to stabilize local torque optimization solution of redundant manipulators," *Journal of Robotic Systems*, vol. 13, no. 3, pp. 177–185, 1996.

[3] Y. Zhang, D. Guo, and S. Ma, “Different-level simultaneous minimization of joint-velocity and joint-torque for redundant robot manipulators,” *Journal of Intelligent & Robotic Systems*, vol. 72, no. 3-4, pp. 301–323, 2013.

Appendices