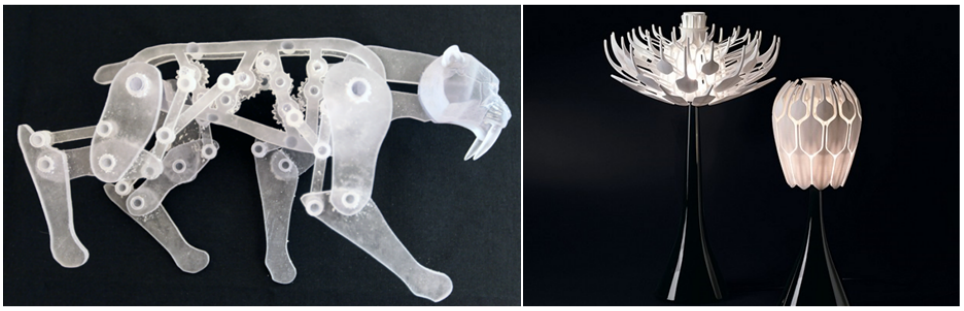
Thesis Title



Jonas Stolle

Bachelor Thesis
May 2021

Supervisors:
Moritz Geilinger
Prof. Dr. Stelian Coros

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

CRL

Abstract

This thesis addresses the development of a novel sample thesis. We analyze the requirements of a general template, as it can be used with the LaTeX text processing system. (And so on...) The abstract should not exceed half a page in size!

Contents

List of Figures

List of Tables

1

Introduction

also state here that we are dealing with mechanical systems in particular

2

Related Work

Sample references are [**?**] and [**?**].

2.1. Appearance Modeling

2.1.1. Taxonomy

2.1.2. Appearance Models

2.2. Human Skin Rendering

3

Your Central Work

3.1. Definitions and Problem Formulation

In this section describes the relvant physical definitions from which the robustness measure is derived and which will be referred to in the later code implementation.

Examples will be given n terms of laikag quadruped robot as most of the testing of the framework was done with that.

3.1.1. Dynamical Systems

Dynamical systems are distinguished by an evolution of their state $\mathbf{x}(t)$ through time. This evolution can be fully described by a set of ordinary differential equations of the form $\dot{\mathbf{x}}(t) = \mathbf{F}(\mathbf{x}(t), t)$ which simplifies to $\dot{\mathbf{x}}(t) = \mathbf{F}(\mathbf{x}(t))$ under the assumption that the dynamical system is autonomous, i.e. is not explicitly dependent on time.

When solving for the explicit solution $\mathbf{x}(t)$, an initial condition $\mathbf{x_0} = \mathbf{x}(t_0)$ is required, which is a system state at an initial time. For simplicity and without loss of generality for autonomous systems, we set $t_0 = 0$. With this the Initial Value Problem (IVP) can be formulated:

$$find\ \mathbf{x}(t) \tag{3.1}$$

$$s.t.\ \dot{\mathbf{x}}(t) = \mathbf{F}(\mathbf{x}(t)) \tag{3.2}$$

$$and\ \mathbf{x}(0) = \mathbf{x}_0 \tag{3.3}$$

We denote the solution to the IVP as $\mathbf{x}(t, \mathbf{x}_0)$. It represents trajectory of the system state through time given an initial condition. The space in which this trajectory lies is spanned by all possible

states x and termed the state space. Note that any future state of the trajectoy $\mathbf{x}(\tau, \mathbf{x}_0)$ at time τ can be interpreted as an initial condition of the IVP itself. It turns out that the new solution coincides with the initial one, i.e. $\mathbf{x}(t, \mathbf{x}_0) = \mathbf{x}(t, \mathbf{x}(\tau, \mathbf{x}_0))$, which illustrates how any state of a trajectory $\mathbf{x}(t) \in \mathbf{x}(t, \mathbf{x}_0)$ is suffitient to describe trajectory itself.

Mechanical systems (which are the focus of this project) tend to be described in terms of so called generalized coordinates $\mathbf{q}(t) = \begin{pmatrix} q_1(t) & q_2(t) & \dots & q_n(t) \end{pmatrix}^T$. They are the minimal set of coordinates needed to fully describe the position and orientation of all of the systems elements. Their dimension n cooincides with the number of degrees of freedom of the system. The related differential equations are of second order, depending on $\ddot{\mathbf{q}}(t)$ in addition to $\dot{\mathbf{q}}(t)$ and $\mathbf{q}(t)$. Simply put, this is due to their derivation by Newton's second method, where forces acting on the system are related to the second time derivative via $F = ma$.

Through an order reduction (Appendix) these differential equations can be cast into the reviously mentioned general form 3.1, where $\mathbf{x}(t) = \begin{pmatrix} q_1(t) & q_2(t) & \dots & q_n(t) & \dot{q}_1(t) & \dot{q}_2 & \dots & \dot{q}_n \end{pmatrix}^T$ (maybe just use vectorial forms of q and qdot). When discussing high level concepts we will refer to the system state $\mathbf{x}(t)$ for simplicity, while in the code implementation the generalized coordinates $\mathbf{q}(t)$ and its derivatives will be more relevant. Keep in mind that both are equivalent.

This implies that in order to solve the IVP, initial conditions $\mathbf{q}_0$ and $\dot{\mathbf{q}}_0$ are required. We can also state that for a system with n degrees of freedom, $\mathbf{x}(t) \in \mathbb{R}^{2n}$. The state space spanned by the generalized coordinates and velocities is termed the phase space $\mathbf{x}(t) = \begin{pmatrix} \mathbf{q}(t) \\ \dot{\mathbf{q}}(t) \end{pmatrix}$. It can be used for a generalizable qualitative analysis of the behaviour of nonlinear dynamical systems.

qdot = f(q,qdot) qddot = g(q,qdot)

order reduction:

x = [q,qdot] => xdot = [qdot, qddot] = F(x) = [f(x), g(x)]

q,qdot,qddot

Explic The dynamic of a system can often be described by a system on differential equations, The dynamical systems we dealt were autonomous and nonlinear in nature. Autonomous as we did not state, derivatives, trajectory

state ivp generalized coordinates and their derivatives trajectories

3.1.2. Attractors and Convergence

From analysis of nonliner dynamics we know that in the phase space of a system, there usually exist sets of states which show an attracting behaviour. Once a trajectory reaches an element of such a set, all future states will also be part of it. (insert math formula)

Define an Attractor $A = \{ \{\mathbf{x} \in \mathbb{R}^{2n} \mid$

These so called attractors come in two variants. If the attracting set consists of only one state, it is called a fixed point. Fixed points come with the condition of $\dot{\mathbf{x}}(t) = \mathbf{0} \forall t \in \mathbb{R}$. This means the

state of the fixed point is unchanging and a trajectory with the fixed point as an initial condition is reduced to a point in the phase space. A classical example of this is the stable bottom position of a pendulum, where if it starts with zero velocity, it won't leave the stable position. This is not the case for any other position as gravity will act on the mass (except for the inverted position, however this is practically not realizable).

In the general case with A containing of more than one state, (reference above eq) is not true. Rather the trajectory is moving through the sets of A, visiting every element at some point in time and returning to it repeatedly in the future in a periodic fashion. These sets are called limit cycles. This name is derived from what happens to states that are close to A, but not part of it. An intuitive example for this are the compliant linkages described in (ref strandbeest compliant version), where the end effector follows a cyclic trajectory, i.e. set of states if undisturbed.

In the code implementation section, we provide methods for dealing with both cases.

define basin of attraction.

Around any attractor, there exists a set of states in the phase space, the trajectories of which all tend towards the attractor. This set is called the basin of attraction. (mathematical formulation) Formally, a trajectory converges to an attractor A iff $\lim_{t\to\infty} \mathbf{x}(t) \in A$ We denote an initial condition which ends up at the attractor of choice as converging towards the attractor. Any initial condition for which this is not true is said to diverge. Note that there may exist any number of attractors in the phase space (reference paper with multitude of attractors) towards the which is why a particular attractor needs to be specified. If the attractor converges a different attractor, it is still defined as diverging from the attractor of interest. The attractor of interest is can be found by simulation, but often it already clear a priori, mostly in the form of the undisturbed state.

The size of this set can be interpreted as a measure of robustness, which was done in (REF) and is further described in (next section).

Determining the

examples for attractors (strandbeest for periodic orbit)

When referring to a system being robust to a disturbance, it means that it's trajectory in the phase space will return to the specified attractor under the influence of that particular disturbance.

Underline that this concept of convergene or divergence is a very general way of saying wheather a system works as intended or doesn't.

3.1.3. Parameter space and Disturbance space

I feel like this should be part of the robustness measure

separate from "dynamics", focus on

dimension disturbance space equals d

In reference they did it with the phase space

We actively separate phase space and disturbance space for the concept to be more versatile

The disturbance space is defined as the set containing all combinations of disturbances, against which the robustness of the system is evaluated. The choice of disturbances in "REFERENCE" are the initial conditions in the phase space (reference to the section in related work or copy that here) as they can be interpreted as the direct result of external forces. In general however, the disturbances may be chosen arbitrarily and the resulting robustness measure will quantify robustness against these. Examples of this were initial tilting of a quadruped over pitch and roll axes and oscillations with the disturbance space spanning the amplitudes and frequencies. Tuple combinations are favoured because of their ease of visualization and mild computational power requirements.

"REFERENCE" chose the DS to coincide with the phase space, as elements of it can be directly interpreted as the result of external forces on the system. This choice is not always feasible nor necessary, as will be shown in the following sections.

It may coincide with the phase space as implemented in "REFERENCE", in which case disturbances can be interpreted as the effects of external forces acting on the system. This moves the system onto a different trajectory , however it will be shown in the following sections, that this is not always feasible nor necesseary. Especially in systems with a large number of degrees of freedom n, the resulting disturbance space is 2n dimensional.

case DS = phase space: d

dimension parameter space = p The two main variable spaces of relevance move are the parameter space (PS), where each dimension represents the value of a chosen system parameter. The goal of finding a robustness measure can be though of as $RM : \mathbb{R}^p \to \mathbb{R}$, i.e. a mapping from the p-dimensional parameter space to a scalar value. (really it also depends on the disturbance space) Finding the best system parameters with respect to the resulting robustness can be formulated as an optimization problem ...

state space v phase space

3.1.4. robustness Measure

The basic idea of robustness

First explain the basics of the REF implementation Then expand to general disturbance spaces with a pointer to the meaning of DS = phase space

The goal aim of the robustness measure is to quantify the robustness of a system with a particular parameter constellation with respect to a set of distrubances. In REF, these disturbances were chosen to be initial conditions sampled from the phase space. They have a nice physical interpretation. However instantaneous forces are not the only kind of disturbance. Here we will apply the concept of the minimal Radius in REF described to some general disturbance space. This may concide with or be a subspace of the phase space, but it must not necessarily be the case. At the same time, evolution of the state trajectory and convergence will still be evaluated in the phase space.

As in "REFERENCE", the basic idea is that the more disturbances the system can endure, i.e. it converges to the attractor, the more robust it is. Discretizing the DS and simply counting

the total number of disturbances the system is robust to is impractiacal because of the possibly fractal nature of the boundary of the basin of attraction (ref) and the fact that the number of evaluations is $O((1/h)^d)$, meaning that higher resolution and a larger DS will drastically increase computational time. To remedy this issue, the in "REF" proposed minimal Radius is found which denotes the radius of the largest hypersphere that sill fits completely within the basin of attraction.

In other words, any combination of disturbances which lies on the surface of the hypersphere with the minimal radius in the DS will result in a trajectory that converges to the attractor in the phase space.

Taking sampling initial conditions form the phase space as disturbances gives a nice physical interpretation and in "REF" this was denoted as general robustness (or was it?), however as a counterexample, periodic or constant disturbances are cleary not covered by this. This is why we worked with general disturbance spaces that can be chosen to work for the particular application at hand. Note that the trajectory will still lie in the phase space and convergence evaluation

3.2. Code Implementation

3.2.1. Framework Overview

differential equations are implicitly represented as simulation objects in dde. Their parameters can be set. The state x as well. Use the solver to compute the trajectory under a disturbance for a given amount of timesteps. check if trajectory converges or diverges. Do this repeatedly with initial conditions sample talk about how knowing the exact shape of the basin of attraction is not necessary for optimization of the robustness, with this and because of the additional work needed to implement the cell mapping algorithms, the method with full trajectories was chosen. For this we need a solver. boom. Great segue!

3.2.2. Solver

Finding an explicit analytical solution to the IVP is possible for simple cases, but very hard if not impossible for more complex systems. The solution to the IVP can be found using numerical solvers. This is not an explicit solution but the trajectory must be iteratively computed by integrating the differential equations over small time steps.

Here we used dde ... write some stuff about dde.

Describe what dde puts out DDE provides q, qdot and qddot at every timestep when computing the trajectories.

Starting with a sytem of ODE's and as corresponding set of initial states, the first step towards computing robustness measure is finding the resulting trajectories. This is achieved by numerically integrating the differential equations over small timesteps until either convergence is detected or the t_max is reached. With the chosen solver it is imperative to keep this timestep

constant when comparing different robustness measures. The time step has a direct influence on the solvers accuracy and by that onto the system dynamics. It is advised to find a value that is a sufficient compromise between accuracy and cumputational time and keeping this fixed from there on.

trajectories here are not continuous but sets of states for discrete timepoints $[t_1, t_2, \ldots, t_n]$.

3.2.3. Detecting Attractors

In order to evaluate wheather a trajectory converges

General idea with similarity of states to attractorstate

Fixed points

fixed points are defined states where $\dot{\mathbf{x}}(t) = \mathbf{0} \quad \forall\, t \;>\; t_0$, i.e. $\mathbf{q}(t) = \mathbf{q}_{fp}, \dot{\mathbf{q}}(t) = \ddot{\mathbf{q}}(t) \forall t > t_0$

Periodic Orbits assume that period of attractor is the same as forcing frequency. Think controller forcing a gait given period, then the leg will also move at that period. This is not generally true and needs to be verified. Poincare => issues with high computational effort and issues with systems that exhibit oscillations with a multitude of periods.

Issue with wrong attractor

Simplification

3.2.4. "Evaluating" Convergence

Cleary it is impractical to let t go to infitiy, which is why some t_n, max needs to be defined at which the simulation stops. quite similar to

3.2.5. minRad algorithm

maybe not go into too much detail on how it works (look at ref)

tuning

visualizations

3.2.6. boundarys PS DS

elephant in the room changing even the unit changes the scaling and therefore the robustness value.

explain and give examples on how boundaries were chosen.

was noted in REF that it can be an ellipse as well, but they didn't know how to choose it's parameters. We do this by analysis and finding reasonable boundaries in the DS.

order of complexity depending on PS and DS

3.2.7. Multithreading

computational time can be reduced by pseudocode

3.2.8. Application to specific systems

need to decide on PS and DS plus boundaries. Need to find or define attractor.

analysis needed to find parameter space and disturbance space boundaries and good initial guesses. analysis of high dof motion tricky (bad 3d image), rather plot every coordinate over time (image).

applyParameters

simAndEval

3.2.9. Optimization and Complexity reduction

find out how the order of computational effort when making discretization smaller vs how much one bisection algorithm iteration can do. cmaes given robustness value, any optimizer that does not depend on derivatives can be used. Of course one can also just explore the entire parameter space, howeve that is quite costly. Useful for debugging though (rough estimate if optimizer converges).

complexity reductions should probably be noted where they are implemented (i.e. not in this section)

3.3. Tests

Introduction to laikago high dof rather long computational time. optimization of robustness examples

Laikago Droptest Laikago Swingtest

4

Conclusion and Outlook

A

Information For The Few (Appendix)

A.1. Foo Bar Baz

A.2. Barontes

A.3. A Long Table with Booktabs

Table A.1.: *A sample list of words.*

ID	Word	Word Length	WD	ETL	PTL	WDplus
1	Eis	3	4	0.42	1.83	0.19
2	Mai	3	5	0.49	1.92	0.19
3	Art	3	5	0.27	1.67	0.14
4	Uhr	3	5	0.57	1.87	0.36
5	Rat	3	5	0.36	1.71	0.14
6	weit	4	6	0.21	1.65	0.25
7	eins	4	6	0.38	1.79	0.26
8	Wort	4	6	0.30	1.62	0.20
9	Wolf	4	6	0.18	1.54	0.19
10	Wald	4	6	0.31	1.63	0.19
11	Amt	3	6	0.30	1.67	0.14

continued on next page

Table A.1.: *(Continued)*

ID	Word	Word Length	WD	ETL	PTL	WDplus
12	Wahl	4	7	0.36	1.77	0.42
13	Volk	4	7	0.45	1.81	0.20
14	Ziel	4	7	0.48	1.78	0.42
15	vier	4	7	0.38	1.81	0.42
16	Kreis	5	7	0.26	1.62	0.33
17	Preis	5	7	0.28	1.51	0.33
18	Re-de	4	7	0.22	1.56	0.33
19	Saal	4	7	0.75	2.10	0.43
20	voll	4	7	0.48	1.82	0.24
21	weiss	5	7	0.21	1.59	0.36
22	-ger	5	7	1.16	2.69	0.59
23	bald	4	7	0.18	1.56	0.19
24	hier	4	7	0.40	1.70	0.43
25	neun	4	7	0.17	1.52	0.26
26	sehr	4	7	0.36	1.85	0.43
27	Jahr	4	7	0.50	1.82	0.43
28	Gold	4	7	0.04	1.35	0.20
29	Ter	5	8	0.15	1.39	0.59
30	Tei-le	5	8	0.30	1.71	0.46
31	Na-tur	5	8	0.18	1.59	0.41
32	Feu-er	5	8	0.30	1.71	0.45
33	Rol-le	5	8	0.15	1.46	0.45
34	Rock	4	8	0.29	1.68	0.25
35	Spass	5	8	0.28	1.64	0.32
36	Gte	5	8	0.49	1.75	0.66
37	En-de	4	8	0.36	1.72	0.33
38	Kunst	5	8	0.26	1.59	0.35
39	Li-nie	5	8	0.45	1.88	0.63
40	Bme	5	8	0.48	1.92	0.45
41	Bh-ne	5	9	0.94	2.48	0.62
42	Bahn	4	9	0.21	1.62	0.42
43	Br-ger	6	9	0.38	1.70	0.65
44	Druck	5	9	0.60	2.03	0.31
45	zehn	4	9	0.41	1.84	0.42
46	Va-ter	5	9	0.36	1.78	0.40
47	Angst	5	9	0.29	1.56	0.35
48	lei-der	6	9	0.13	1.47	0.52
49	hfig	6	9	0.82	2.31	0.52
50	le-ben	5	9	0.38	1.85	0.40
51	aus-ser	6	9	1.20	2.26	0.57
52	be-vor	5	9	1.28	2.75	0.39

continued on next page

Table A.1.: *(Continued)*

ID	Word	Word Length	WD	ETL	PTL	WDplus
53	Kai-ser	6	9	0.92	2.37	0.53
54	Markt	5	9	0.23	1.58	0.28
55	Os-ten	5	9	0.21	1.54	0.48
56	Krieg	5	9	0.33	1.67	0.50
57	Mann	4	9	0.31	1.47	0.25
58	Hal-le	5	9	0.24	1.65	0.45
59	heu-te	5	9	0.44	1.87	0.46
60	in-nen	5	10	0.36	1.80	0.45
61	Na-men	5	10	0.28	1.72	0.41
62	jetzt	5	10	0.70	2.07	0.32
63	kei-ner	6	10	0.28	1.62	0.53
64	Schu-le	6	10	1.02	2.12	0.48
65	Ar-beit	6	10	0.34	1.70	0.52
66	An-teil	6	10	0.27	1.63	0.53
67	di-rekt	6	10	0.67	2.04	0.47
68	vor-her	6	10	0.78	2.25	0.47
69	wol-len	6	10	0.44	1.85	0.51
70	Kampf	5	10	0.70	1.96	0.27
71	dern	6	10	1.18	2.62	0.65
72	lau-fen	6	10	0.21	1.64	0.52
73	Eu-ro-pa	6	10	0.23	1.53	0.66
74	statt	5	10	1.61	2.86	0.39
75	Wes-ten	6	10	0.29	1.60	0.54