Resonance frequency of
Hopping Robots

Spring Term 2011
# Contents

Abstract iii
Symbols v

## Introduction

1 The 22-Robot
   1.1 Design 3
   1.2 Spring Stiffness of Double-Beam 5
   1.3 Trajectories of Radial and Rotational Mode 7
   1.4 Accelerometer and FFT 10
   1.5 Measuring Resonance Frequency with Accelerometer 11
   1.6 New Method: Design 2 of the 22-Robot 12
   1.7 New Method: Design 3 of the 22-Robot 13
   1.8 Cost of Transport of the 22-Robot 15
   1.9 Hopping Trajectory 15
   1.10 Conclusions of the 22-Robot 18
      1.10.1 Resonance Frequency of the 22-Robot (design 3) 18
      1.10.2 Cost of Transport 19

2 Two-Beam Mirror Robot 21
   2.1 Steering Control 22
   2.2 Synchronisation 23
   2.3 Conclusions of the Two-Beam Mirror Robot 23

3 Carbon Hopper 25
   3.1 Carbon Robot 25
   3.2 Conclusions of the Carbon Hopper 26

4 Conclusions and Recommendations for Future Work 27

A Codes 22-Robot 29
   A.1 MATLAB-code for gathering data from accelerometer 29
   A.2 MATLAB-code for the FFT and resonance frequency 30
   A.3 Arduino-code for gathering data from accelerometer 33

B Codes Two-Beam Mirror Robot 35
   B.1 MATLAB-code for controlling the robot with the keyboard 35
   B.2 Arduino-code for controlling the robot with the keyboard 36

C Carbon Beam and Plates 41

Bibliography 43
Abstract

This work is about manipulating resonance frequency of hopping robots and direction control of a two-beam hopping robot. First, an extended design of the 2-robot was developed which allowed to manipulate various parameters. Then, different methods for changing the resonance frequency were examined. One of these methods then was used for further investigation, analyzing the resonance frequency in different configurations as well measuring hopping height and the cost of transport. The results showed that the resonance frequency can be changed significantly and that the angle of the radial trajectory has great influence on the cost of transport. Then a steering control for a two-beam hopping robot was developed. Also the synchronisation problem of this robot is shortly described. At last, the key-points of making an energy efficient hopping robot are discussed together with experiments with a carbon hopper.
Acknowledgements

I would like to thank Prof. Dr. Fumiya Iida for giving me the opportunity to do my thesis at his Bio-Inspired Robotics Lab at the ETH Zurich. Many thanks to my supervisors Murat Reis, who always had inspiring ideas that gave me a fresh view, and Keith Gunura for many fruitful discussions and for the constant support. In addition, thanks to Nandan Maheshwari and all the other members of the BIRL lab.

A special thanks belongs to my father who helped me with the construction of my robots and who followed my work with great interest.
Symbols

Symbols

$\theta$ angle of the trajectory of the radial mode
$k$ stiffness of the spring
$k_{\text{radial}}$ radial spring stiffness
$k_{\text{rotational}}$ rotational spring stiffness
$\Delta d_{\text{top}}$ set distance of the top clamp
$\Delta d_{\text{mid}}$ set distance of the midcurve clamp
$\Delta d_{\text{foot}}$ distance between the beams at the attachment points at the foot

Acronyms and Abbreviations

CoT cost of transport
FFT fast fourier transform
CF carbon fibers
UD unidirectional carbon fibers
Introduction

As already mentioned in [2] and implemented in [3], a newer approach of controlling a robot is not working against its natural dynamic tendencies. One of the most feared phenomena, when controlling a robot, is resonance and vibration because the robot body dynamic behaviour changes drastically resulting in uncontrollability. However, by not avoiding these resonance frequencies but rather using them for creating a different kind of locomotion is the key idea behind the energy efficient hopping robot in [1]. Similar projects inspired from biological studies about energetics of animal locomotion (e.g. [4], [5], [6]) were done in the past, although most of them investigating simulated legged robots with idealized springs (e.g. [7], [8], [9], [10], [11]) with only a few successful attempts of hopping robots, demonstrated in [12], [13].

The robot presented in [1] is able to develop a steady-state motion without any sensors and control. Because it has no control abilities, it can only hop forward. This work examines the possibilities of manipulating the resonance frequency of such a robot and making it energy efficient and it also describes a robot which can be steered.

Because different projects are described in this work, it is organized as follows. First the 22-robot is presented together with all measurements and results. This starts with the very first experiments with the new robot, showing its further development and findings chronologically, finishing with some analysis of the cost of transport of the final design of the robot. The obtained results then are discussed shortly.

Second, the two-beam mirror robot is introduced with its special configuration. The work with this robot is described as well as the problems occured when dealing with multiple elastic beams.

Finally, the concept of the carbon hopper is shown together with the key points of creating an energy efficient hopping robot.
Chapter 1

The 22-Robot

1.1 Design

The first task was to change the design of the 2-Robot\[1\] in a way that the construction parameters could be changed. The idea was to be able to change properties of the construction online to achieve different configurations of the robot. Among all parameters, only the oscillating mass and the spring stiffness $k$ have an influence on the resonance frequency of the elastic beam according to the well-known formula of a spring-damper system (formula 1.1)

$$m \cdot \ddot{x} + d \cdot \dot{x} + c \cdot x = f(t)$$  \hspace{1cm} (1.1)

where

$$\omega_{\text{resonance}} = \sqrt{\frac{c}{m}}$$  \hspace{1cm} (1.2)

and where $m$ represent the mass, $d$ the damping factor and $c$ the spring stiffness of the system. Since the mass cannot be changed online in formula 1.2, the parameter to change was therefore chosen to be the spring stiffness $k$.

This was done by adding a second beam behind the first. By changing the distance between the beams either at the attachment points on the foot of the robot or somewhere along the beams, one can influence the spring stiffness of the spring formed by these two beams. In the first approach, only the attachment points of the two beams were variable and the beams joined each other on top of the robot, creating a form similar to a “22” (figure 1.1).

Deriving the eigen-values of such a system is not trivial (investigation on vibration of cylindrically curved and slightly curved beams can be found in [14], [15]) and since the testing of this method would require numerous experiments, one main aspect of the design was to easily be able to change the beams, position of the beams as well as the position of the motor. Clamps were used for fixing the beams to the foot and the motor to the beams. Also, the weight of the rotating masses should be interchangeable. Here, screws with different weight as the rotating masses could be added. This allowed a very fast adaptation to new configurations (figure 1.2 (a) and (b)).

In order to gain some understanding of the spring stiffness of the elastic beam and the rotating masses, first only one beam was implemented, copying the original 2-robot. A strong steel beam turned out to be suitable for a single-beam robot. But because the robot now used two beams as the spring, by adding the second beam the spring was expected to be stiffer than by just using a single beam. The first experiments looking at the hopping behaviour therefore were done using soft
steel beams. The beams turned out to be far too soft, the robot could not hop because the force generated by the spring could not lift the foot off the ground. Stronger aluminium beams showed a better performance. The robot was able to hop, although the stiffness was small. Finally, the same kind of steel beam used for single-beam hopping created the desired behaviour: The robot developed a steady-state hopping. This is a little surprising, the reason for this and the resulting consequences are shown later.

As shown in [1], such a construction develops two modes: The rotational and the radial mode. Both modes have individual resonance frequencies. While in the rotational mode the trajectory of the motor describes a circular movement, leading to a back and forth motion of the beams, the trajectory of the motor in the radial mode describes a linear up and down motion, making the robot lift its foot off the ground and therefore making it hop. Because only the radial mode can be used for hopping, the following experiments aimed at changing the radial spring stiffness $k_{\text{radial}}$. 
1.2 Spring Stiffness of Double-Beam

Having successfully developed a working robot with the design as shown in figure 1.1, the influence of the second beam had to be examined. In order to verify if the method of changing the distance between the beams at the foot is feasible, the radial spring stiffness and thus the influence of the second beam was measured.

A force sensor measured the force needed for a particular displacement of the motor on top of the robot. Because the real trajectories of the motor in radial vibration mode for the different configurations were not known yet, the measurements do not exactly represent the behavior of the spring due to the errors in the trajectories. The chosen trajectory for the measurements below is perpendicular to the foot (figure 1.3). Figure 1.4 shows the displacement versus the force. The slope of these graphs represent the “radial” (because of the errors) spring stiffness $k$ according to the
Chapter 1. The 22-Robot

Formula

\[ F = \hat{k} \cdot \Delta x \] (1.3)

where \( F \) is the force measured, \( \hat{k} \) the spring stiffness and \( \Delta x \) the displacement.

Figure 1.3: Chosen trajectory perpendicular to the foot

Figure 1.4: Force measurements of the system (design 1)
It can be inferred from figure 1.4 that the “radial” spring stiffness $\hat{k}$ almost shows no change in the different configurations. The change of distance $\Delta d_{\text{foot}}$ seems to have almost no influence on the radial spring stiffness. On the one hand, this was an unexpected result, on the other hand, the errors caused by the difference of the trajectories could possibly lead to this behaviour. In order to obtain more accurate measurements, the actual trajectories in both radial and rotational mode got analyzed in the next section.

1.3 Trajectories of Radial and Rotational Mode

Using the same technique as in [1], a LED was added on the motor. By setting the shutter time of a camera correctly, a full cycle of the hopping motion can be recorded. For these experiments, the robot was glued to a plate which simplified the recording. The robot then was set to resonance frequency in both modes for every configuration and a picture of one full cycle was taken. $\Delta d_{\text{foot}}$ ranges from 7cm to 28cm with steps of 3cm. Figure 1.5 shows the results for the rotational mode, figure 1.6 for the radial mode. The shutter time of the camera and the $\Delta d_{\text{foot}}$ are indicated below each picture.

Most of the pictures show a clear distinction of the two modes in almost every configuration. In figure 1.5 it can be seen that there is a gradual divergence from an almost single line trajectory to a curved figure 8 trajectory. This shows that the two modes overlap instead of increasing in amplitude as compared to figure 1.6. The shutter time however indicates that the founding of the previous experiment was probably correct. $\Delta d_{\text{foot}}$ influences the rotational stiffness but hardly the radial stiffness. The measurements contain a lot of errors, the shutter time only approximately indicates the duration of one full cycle.
(a) $\Delta d_{foot} = 7\text{cm}, T_{shutter}=0.9\text{s}$

(b) $\Delta d_{foot} = 10\text{cm}, T_{shutter}=0.9\text{s}$

(c) $\Delta d_{foot} = 13\text{cm}, T_{shutter}=0.9\text{s}$

(d) $\Delta d_{foot} = 16\text{cm}, T_{shutter}=1\text{s}$

(e) $\Delta d_{foot} = 19\text{cm}, T_{shutter}=1\text{s}$

(f) $\Delta d_{foot} = 22\text{cm}, T_{shutter}=1\text{s}$

(g) $\Delta d_{foot} = 25\text{cm}, T_{shutter}=1.1\text{s}$

(h) $\Delta d_{foot} = 28\text{cm}, T_{shutter}=1.1\text{s}$

Figure 1.5: Trajectories of the rotational mode
1.3. Trajectories of Radial and Rotational Mode

Figure 1.6: Trajectories of the radial mode

(a) $\Delta d_{foot} = 7\text{cm}, T_{shutter}=0.25\text{s}$
(b) $\Delta d_{foot} = 10\text{cm}, T_{shutter}=0.25\text{s}$
(c) $\Delta d_{foot} = 13\text{cm}, T_{shutter}=0.25\text{s}$
(d) $\Delta d_{foot} = 16\text{cm}, T_{shutter}=0.25\text{s}$
(e) $\Delta d_{foot} = 19\text{cm}, T_{shutter}=0.25\text{s}$
(f) $\Delta d_{foot} = 22\text{cm}, T_{shutter}=0.25\text{s}$
(g) $\Delta d_{foot} = 25\text{cm}, T_{shutter}=0.25\text{s}$
(h) $\Delta d_{foot} = 28\text{cm}, T_{shutter}=0.25\text{s}$
1.4 Accelerometer and FFT

To measure the exact changes in the resonance frequency, an accelerometer was implemented (figure 1.7).

![Accelerometer ACA302](image)

The sensor used is a 3-axis accelerometer\(^1\) which can measure acceleration in x-, y- and z-direction up to 2g. It runs at 5 volt. Table 1.1 shows some properties of the sensor. To establish a connection between the sensor and the computer, it was controlled by an Arduino Due\(^2\) board which itself communicates with MATLAB. The control was implemented as a master/slave-program: MATLAB sends a “read” command to the Arduino board which then gets the current acceleration data from the sensor. This data then is sent to the serial port, where MATLAB could get the data from (see code in appendix A).

<table>
<thead>
<tr>
<th>Parameter</th>
<th>ACA302</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acceleration Range</td>
<td>± 2G</td>
<td></td>
</tr>
<tr>
<td>Sensitivity [Vcc=3V]</td>
<td>300</td>
<td>mV/G</td>
</tr>
<tr>
<td>Supply Voltage</td>
<td>2.7 - 5.5</td>
<td>V</td>
</tr>
<tr>
<td>Linearity</td>
<td>6%</td>
<td></td>
</tr>
<tr>
<td>Mass</td>
<td>3.8 g</td>
<td></td>
</tr>
</tbody>
</table>

The following procedure has been used for converting the data from the sensor to actual acceleration: The arduino board has a 10-bit resolution, leading to values between 0 and 1023 for the voltage output of the sensor, ranging from 0V to 5.04V. Since the sensor can measure positive and negative accelerations, first the value had to be found for which the acceleration is zero. The sensitivity defines the output range of the sensor which now only have to get converted to volt by a certain factor (see MATLAB code in appendix A for detail). For calibration, the sensor data got calibrated by measuring gravitational acceleration seperately in x-, y- and z-axis. The sensor gathers 300 datapoints over approximately 10 seconds which results to a sampling frequency of 30Hz. After gathering the data, it got stored in MATLAB. In order to get the resonance frequency, a fast fourier transform (FFT)\(^3\) is used to split the signal into its corresponding periodic components. Since the motor performs a

---

\(^1\)http://www.servolfo.com/downloads/acx302.pdf
\(^2\)http://www.arduino.cc
\(^3\)http://en.wikipedia.org/wiki/Fast_Fourier_transform
stable, periodic movement when set to resonance frequency, the representation of the signal in the frequency domain should contain one large peak corrupted with noise. In fact, the FFT showed this peak and picking out the frequency causing this peak led to the resonance frequency.

1.5 Measuring Resonance Frequency with Accelerometer

As previously mentioned, the sensor measured the acceleration in all three directions. The sensor was mounted on top of the robot next to the motor so two axes of the sensor would lie in the plane the motor is moving. In this case, the y- and z-axis recorded the up and down as well as the forward and backward motion of the motor while the x-axis recorded the lateral side-to-side motion which in this case was not of interest (see figure 1.12). Since the motor develops a periodic motion in the y-z-plane, the FFT of these signals should reveal the same resonance frequency. The recorded data only was valid if both the peaks in the y- and z-axis led to the same resonance frequency. A second criteria was the shape of the peak itself: A sharp peak indicates the exact match of the induced frequency by the motor and the resonance frequency determined by the system while a flattened peak contains various frequencies around the actual resonance frequency which causes errors in the measurements.

Figures 1.8, 1.9 and 1.10 show an example of a set of data of all three axes. The signal obtained in figure 1.8 only consists of noise with no periodicity while figure 1.9 and 1.10 share the same periodic movement with the same frequency. In figure 1.11, the FFT of the acceleration of z-axis (figure 1.10) is shown.
1.6 New Method: Design 2 of the 22-Robot

With this accelerometer, precise measurements were taken. For the next experiments, a different approach was developed to change the resonance frequency because the first experiments did not lead to the desired effect. Instead of changing $\Delta d_{\text{foot}}$, this parameater was kept constant while the distance $\Delta d_{\text{mid}}$ now was changed midcurve (figure 1.12) because observations of the motion of the beams revealed that the part that move the most during the steady-state motion is the curve of the inner beam, meaning that the biggest change in distance occured to be in between the curvature of the two beams. Figure 1.13 show the results of these measurements.
Note that for these and all the following experiments, the mass on top of the robot was changed causing significant higher resonance frequency.

![Figure 1.12: Change distance midcurve (design 2), fix ∆d_{foot}](image)

Indeed there was a change in the resonance frequency, but it was relatively small and in the opposite direction than expected: Instead of increasing $k_{radial}$ and resonance frequency by increasing $\Delta d_{mid}$, the resonance frequency decreased.

### 1.7 New Method: Design 3 of the 22-Robot

After various experiments with different approaches, mostly changing the distance between the two beams at some point, a combined strategy was developed: The method of changing $\Delta d_{mid}$ remained, but instead of aligning both beams on top
of the robot and fixing them with the motor, the beams were separated on top by a certain distance $\Delta d_{\text{top}}$. Figure 1.14 (a) shows a schematic sketch of this method and figure 1.14 (b) the real robot.

![Sketch of design 3](image1.jpg)

(b) Real robot with the new design (design 2) without motor

Figure 1.14: New design with offset on top

With this method, it was possible to change the resonance frequency significantly. Two series of experiments were taken: The first with $\Delta d_{\text{top}} = 3\, \text{cm}$, the second with $\Delta d_{\text{top}} = 5\, \text{cm}$ (figure 1.15). These results are discussed in section 1.10.

![Resonance Frequency of the System (Design 3)](image2.png)

Figure 1.15: Resonance frequency of the system (design 3)
1.8 Cost of Transport of the 22-Robot

To complete the first project, a cost of transport (CoT, [16], [17]) analysis of these different configurations of the design 3 of the 22-robot was done. Here, only the version with $\Delta d_{\text{top}} = 5\text{cm}$ was examined. The cost of transport is given by

$$c_t = \frac{\text{(energy used)}}{\text{(weight x distance traveled)}} = \frac{U \cdot I \cdot t}{m_{\text{robot}} \cdot \Delta l} \quad (1.4)$$

where $U$ represents the voltage, $I$ the current, $t$ the time needed to travel $\Delta l$ and $m_{\text{robot}}$ the mass of the robot.

For the following six experiments, the robot first had to develop a steady-state motion. Then the robot had to hop $50\text{cm}$ ($\Delta l = 0.5\text{m}$), during which voltage, current and time was measured. The mass of the robot is given by $m_{\text{robot}} = 0.88\text{kg}$. Figure 1.16 shows the cost of transport for every configuration.

The measurements itself contain relatively big errors. Even if the robot developed the steady-state motion, the velocity and therefore the time was not the same for the same configuration as well as the current differs during the procedure. For every configuration, four runs were measured and the average of all parameters was taken.

![Figure 1.16: Cost of transport for the 22 robot, $\Delta d_{\text{top}} = 5\text{cm}$](image)

1.9 Hopping Trajectory

To verify and compare the graph obtained in figure 1.16, the hopping trajectory of the real robot was analyzed for every configuration. To record the hopping trajectory, a high speed camera was used to record the steady-state motion. The recordings were analyzed by the open source program OpenCV$^4$. This program tracks the movement of a point, specified with a special mark. The mark was attached between the two beams at the attachment points because the force making

the robot hop acts approximately at this point (figure 1.17), thus the movement of this point is most meaningful.

For each configuration, the movement of this point was tracked to make the steady-state behaviour clearly visible. In the following pictures, the red line shows this track. The record time was between 5 and 10 seconds to catch a few cycles.

![Figure 1.17: Robot with mark on the foot](image)

Table 1.2 gives an overview over the average hopping height for each configuration. These numbers were calculated by using a pixel-to-centimeter ratio, defined by the height of the frame the camera can record (in real world) divided by the number of pixels the picture made by the camera contains. By analyzing how many pixels the robot moved in the picture, the actual hopping height can be found by multiplying this number with the pixel-to-centimeter ratio. The table also indicates $\theta$ (see figure 1.20 for detail) as well as the absolute change of $\theta_{\text{diff}}$ which indicates the change of $\theta$ with respect to the last $\theta$ measured, e.g.

$$\theta_{\text{diff}} = \theta_{\text{previous}} - \theta_{\text{current}} \quad (1.5)$$

where $\theta_{\text{previous}}$ is the last $\theta$ measured and $\theta_{\text{current}}$ the current $\theta$. As will be seen in the next section, this angle is greatly affected by the change of $\Delta d_{\text{mid}}$ and it is one of the main reasons causing the behaviour of the graph in figure 1.16.

<table>
<thead>
<tr>
<th>$\Delta d_{\text{mid}}$ [cm]</th>
<th>average hopping height [cm]</th>
<th>$\theta$ [degree]</th>
<th>$\theta_{\text{diff}}$ [degree]</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>1.64</td>
<td>52</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>1.37</td>
<td>56</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>1.42</td>
<td>60</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>1.36</td>
<td>65</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>1.15</td>
<td>74</td>
<td>9</td>
</tr>
<tr>
<td>8</td>
<td>1.1</td>
<td>83</td>
<td>9</td>
</tr>
</tbody>
</table>
1.9. Hopping Trajectory

Figure 1.18: Hopping trajectories

(a) $\Delta d_{mid} = 3\text{cm}$

(b) $\Delta d_{mid} = 4\text{cm}$

(c) $\Delta d_{mid} = 5\text{cm}$

(d) $\Delta d_{mid} = 6\text{cm}$

(e) $\Delta d_{mid} = 7\text{cm}$

(f) $\Delta d_{mid} = 8\text{cm}$
Conclusions of the 22-Robot

1.10.1 Resonance Frequency of the 22-Robot (design 3)

After the experiments with the first two designs of the 22-robot, design 1 and design 2, an important characteristic was found: While almost every change affects the rotational stiffness $k_{\text{rotational}}$ of the spring, a change of the radial stiffness $k_{\text{radial}}$ can only be achieved by changing the shape of the curvature of the beams. Therefore, design 3 of the 22-robot was developed to achieve such a change by introducing an offset $\Delta d_{\text{top}}$ on top of the beams. Figure 1.19 schematically illustrates the effect caused by this offset, starting with a flattened curve when $\Delta d_{\text{mid}}$ is fully extended leading to a circular shape for small $\Delta d_{\text{mid}}$.

![Figure 1.19: Effect of the offset](image)

By looking at the graphs of figure 1.15, it can be seen that the resonance frequency increases with increasing $\Delta d_{\text{mid}}$. This corresponds with figure 1.19. Although the absolute change may seem small, it is significant due to the following reason:

- The system has two modes: the rotational vibration and the radial vibration
- Each of this mode has an infinite number of resonance frequencies (e.g. the first resonance frequency has 0 nodes, second frequency has 1 node etc.)
- These resonance frequencies are predefined by the system
- The two modes are not dependend on each other, meaning that one can change the resonance frequency for one mode by changing the respective spring stiffness almost without effect on the other mode. Sometimes both modes share the same resonance frequency, leading to a combined steady state motion
- The difference in frequency between the two modes may be large or small, dependend only on the shape of the robot

A closer look at the graphs reveals an initial resonance frequency of 3,8573 Hz for the case of $\Delta d_{\text{top}} = 3\text{cm}$ and $\Delta d_{\text{mid}} = 3\text{cm}$. Setting $\Delta d_{\text{mid}}$ to 8cm, the resonance frequency now is located at 5,2884 Hz which represents a relative change of 37.1% without changing the mode nor changing the number of nodes (which in this case is
1.10. Conclusions of the 22-Robot

Almost the same relative change of 36.5% can be observed for $\Delta d_{top} = 5cm$. This can be considered as a relatively big change which could be used for changing the hopping behaviour and therefore creating different gait patterns which is shortly discussed in chapter 4. Note that the model developed in [1] still can be used for the 22-robot, except that the radial spring stiffness $k_{radial}$ and the angle of the radial trajectory $\theta$ now are functions of the distances between the beams:

$$ k_{radial} = k_{radial}(\Delta d_{top}, \Delta d_{mid}, \Delta d_{foot}) \quad (1.6) $$

$$ \theta = \theta(\Delta d_{top}, \Delta d_{mid}, \Delta d_{foot}) \quad (1.7) $$

1.10.2 Cost of Transport

The graph in figure 1.16 shows the overall behaviour of the cost of transport. While the CoT increases in a linear manner in the first four configurations, it suddenly diverges for the last two configurations. This is related to the in the change of the curvature shown in figure 1.19. The change of the shape not only affects $k_{radial}$ but it also changes the angle $\theta$ of the radial trajectory which can be seen in table 1.2.

Figure 1.20 shows this effect on the real robot.

This angle plays a very important role in the forward velocity of the robot: A small angle results in a large stepsize, a large angle to a small stepsize ($0 \leq \theta \leq 90^\circ$ for which $\theta = 0^\circ$ indicates slipping only and for $\theta = 90^\circ$, the robot would hop without forward/backward velocity). Thus, the last column of table 1.2 explains the CoT of the last two configurations: Since the gain of $\theta$ from one configuration to the next configuration is not equal to the other gains for the last two measurements, the stepsize does not decrease linear with respect to all measurements, resulting in a sudden decrease of the velocity and increase of CoT (because of the time needed for hopping 0.5m). The linear correlation however between the resonance frequencies in figure 1.15 is obvious:

- The voltage increases with an increasing spring stiffness $k_{radial}$ to cope with the higher resonance frequency
• This leads to smaller hopping heights and stepsizes and thus to more steps which then causes a lower velocity because the robot only moves during flight phase (no movement in the stance phase)

• Because of the lower velocity, the time increases which altogether result in a higher cost of transport according to formula 1.4

Figure 1.18 illustrates these observations: In (a), the hopping shows a barely steady-state hopping with big jumps and large steps. This means a large velocity which results in a low CoT. In (b), (c) and (d), the stepsize is slowly decreasing, causing the linear increase of the CoT. (e) and (f) then show significant lower hopping heights and stepsize which result in these high CoTs.
Chapter 2

Two-Beam Mirror Robot

The two-beam mirror robot was developed to examine steering of a hopping robot. It also uses two beams, but the second beam is not placed behind the first. Instead, the first beam is fixed at one end of the foot and the second beam parallel to the first one at the other end, creating a symmetric robot (figure 2.1). A motor with a rotating mass is fixed on top of every beam.

Figure 2.1: Two-beam mirror robot
The robot focused on two questions: How can direction control be achieved and how do the beams synchronize when used at the same time?

2.1 Steering Control

In order to eliminate cable issues and to make the robot autonomous, an Arduino Bluetooth device was used instead of a normal Arduino board. This allowed communication over bluetooth from a laptop to the device. The board is shown in figure 2.2.

![Arduino Bluetooth device with supply voltage](image1)

The board is mounted in the middle of the robot together with its supply voltage of 3V to achieve an approximately symmetric distribution of the additional weight. Because the analog output of the arduino is a PWM-signal\(^1\), also motor drivers\(^2\) had to be implemented. These drivers use a separate supply voltage which then can be controlled by using the PWM-signal. In this case, their supply voltage is a 9V battery, therefore the output voltage for the motors range from 0 to 9V. Because every beam uses its own motor and motor driver and for reasons discussed in the next chapter, these drivers and batteries are located on top of each beam behind the motor (figure 2.3).

![Motor, motor driver and battery](image2)

\(^1\)http://en.wikipedia.org/wiki/Pulse-width_modulation
\(^2\)http://www.ixs.co.jp/en/download/manual/iMDs03_manual_e.pdf
The arduino board communicates - like in the last chapter - with MATLAB. To create an intuitive operation, the robot should be controlled by the arrow keys of the keyboard. The structure of the program is very simple: If MATLAB detects a press on an arrow key, it sends a “run”-command to the respective motor, otherwise it sends a “stop”-command. In addition, the speed and direction of rotation of each motor can be adjusted manually by setting the PWM-signal for each driver with other keys of the keyboard and by pressing “space”, Arduino sends the current speed of both motors to MATLAB (see code in appendix B). The steering control is simple. By running both motors at the same time, the robot hops forward and by running only one motor, the robot either turns right or left. Two interesting points were observed:

- When turning, the speed of the running motor needs to be higher since the hopping height decreases which must result in a higher resonance frequency
- It is not possible to make the robot hop backward. Even if the rotation directions of the motors are inversed, the robot performs the same motion

2.2 Synchronisation

Hopping forward requires both motors running at the same time. To create a steady-state motion, they must be perfectly synchronized, otherwise the robot would develop an unstable hopping motion. If possible, this synchronisation should happen without any sensors or control. Note that for a control of the synchronisation, at least a model of the flight and the stance phase and a model of the influence of each beam to the motion of the other beam is required. Even then, if the beams do not have matching properties such as different length, slightly different spring stiffness, different attachment points etc., it is almost impossible to control this synchronisation.

2.3 Conclusions of the Two-Beam Mirror Robot

With the presented design of the robot, an easy way for controlling the direction of the hopping is realized. Turning however requires a higher velocity of the rotating mass and a backward motion is not possible.

The solution to the synchronisation problem is a **passive synchronisation**. This means that the robot and the rotating mechanism are constructed in a way that the steady-state result are synchronized beams, given any initial condition. In this case, this is done by adjusting the weight of the rotating masses. Normally, if one increases the voltage of the motor, the rotation starts slowly, heading towards resonance frequency and then escaping resonance frequency. By increasing the weight of the rotating mass, this behaviour can be changed such that once in resonance frequency, the power of the motor is not enough to escape since this requires a large amount of energy: The energy of the rotating masses keeps the motion fixed to the resonance frequency even if the voltage of the motor increases (within a certain range of course). With this method, the beams automatically synchronize even with a 180° phase shift in the rotating arms. The correct weight has to be found experimentally.

---

Chapter 3

Carbon Hopper

The idea of the carbon hopper was to go towards the development of the most energy efficient hopping robot. According to the newest findings of recent works in the BIRL-lab, this includes the following points:

- The foot must be as light as possible so the energy needed for lifting the foot off the ground is small

- The beam also must be as light as possible for the same reason because the weight of the beam can approximately be split into two parts. Half of the weight adds to the weight of the foot, the other half to the weight on top of the beam

- The beam should have no damping

- The beam should have a high spring stiffness so it can carry a large mass

- The mass on top of the robot should be large. A large mass leads to a large compression of the beam which leads to a lower resonance frequency which consumes less energy since the voltage for running the motor is smaller

- The weight of the rotating masses as well as the length of the rotating arms then are determined experimentally

By searching for materials for these conditions, two suitable materials were found: Titanium and carbon fibre (CF). Because CF was not examined yet, it was decided to build a hopping robot completely out of carbon and analyze the properties of the material. The manufacturing process of this robot is described in appendix C.

3.1 Carbon Robot

A first test of the robot was made by just assembling one plate and the beam (figure 3.1). The robot was able to hop and it achieved a COT of

$$c_t = \frac{3.3V \cdot 0.1A \cdot 10s}{0.26kg \cdot 0.5m} \approx 25$$  \hspace{1cm} (3.1)

The total mass of the robot plays an important role. The robot itself is very light, weighting only 260 grams (with motor and rotating mass). This mass could be further reduced by applying a proper foot. However, to improve the COT, the total mass should be as heavy as possible. Because of the previously mentioned aspects, this additional mass needs to be applied on top of the robot, increasing
the total mass of the robot and at the same time decreasing the voltage needed by decreasing the resonance frequency. This possible additional weight only depends on the properties of the beam.

3.2 Conclusions of the Carbon Hopper

The concept of using only carbon fibers seems suitable, at least for the stiff part of the robot. Due to a lack of experience in dealing with CF, the beam built for this robot was at its limits so no additional weight could be added to improve the CoT. Although the beam showed good properties, e.g. light and little damping, further investigation would be needed to build an optimal beam.
Chapter 4

Conclusions and Recommendations for Future Work

In the first chapter, a method was found to change the stiffness of an elastic curved beam by using two beams in row and changing the distance between these beams. The results of this method show that a relative change of the resonance frequency of approximately 35% can be achieved. This can be regarded as a relatively large change since neither the mode nor the number of nodes change using the presented method. Because also the structure of the robot changes in the different configurations, recordings of the motion with a high speed camera revealed that the cost of transport of the robot is strongly connected to the angle of the radial trajectory, causing the CoT to diverge at certain angles.

The second chapter presents a robot which can control its hopping direction by using two beams in parallel side by side. A wireless bluetooth communication was developed to eliminate cable issues and to be able to control each beam separately. Running only one motor makes the robot turn while hopping forward is done by running both motors at the same time. Hopping backward however is not possible. In the third chapter, the key points towards the design of an energy efficient hopping robot are discussed. A robot was built completely out of carbon to examine the properties of this material and if it can fulfill the requirements. The material is suitable for the foot of the robot but building a proper beam out of carbon fibre is not trivial and needs a closer investigation. The beam built for this robot was too soft and due to the given time for this thesis, no more experiments could be taken with different carbon beams.

For future work, the actual online implementation of the presented method is realized. The clamps can be replaced with strong motors which could be controlled for example with an arduino board. Also, a first application of the method could be the two-beam mirror robot. Instead of changing the speed of the motors for either turning or hopping forward, the spring stiffness of each beam could be changed individually, making the speed of the motors constant. In general, the method possibly could be used for any joints where the stiffness of the joint needs to be adjusted. In addition, the oscillating mechanism could be revised. The rotating masses do not imitate nature, thus a more “natural” mechanism could be developed.
Appendix A

Codes 22-Robot

A.1 MATLAB-code for gathering data from accelerometer

clc
format compact
clear all

% butter-filter
wn=0.09;
order=20;
% moving average filter
moving_average=2;
% simulation-time
it=500;
t=(1:it);
% memory allocation
x_vector=zeros(it,1);
y_vector=zeros(it,1);
z_vector=zeros(it,1);
% value_vector=zeros(1:it)

s2=serial('COM4', 'BaudRate', 9600);
fopen(s2);
pause(2);

disp('measurement in progress')
tic;
for i=1:it
% get values from the accelerometer
fwrite(s2,'1');
value_vector{i}=fgetl(s2); % write values into cell array
end

time_it= toc;
disp('measurement finished')

% converting to numbers
for j=1:it
    temp=value_vector{j}; % store string in a temp variable
komma_index=strfind(temp,',');  \%find komma
\%split string into components
x_vector(j)=str2double(temp(1:komma_index(1)-1));
y_vector(j)=str2double(temp(komma_index(1)+1:komma_index(2)-1));
z_vector(j)=str2double(temp(komma_index(2)+1:length(temp)));
end

\% [B,A]=butter(order,wn);
\% x_vector_filter=filter(B,A,x_vector);
\% y_vector_filter=filter(B,A,y_vector);
\% z_vector_filter=filter(B,A,z_vector);

\%moving average filter
for n=moving_average:it
x_vector_filter(n)=sum(x_vector(n-(moving_average-1):n))/moving_average;
y_vector_filter(n)=sum(y_vector(n-(moving_average-1):n))/moving_average;
z_vector_filter(n)=sum(z_vector(n-(moving_average-1):n))/moving_average;
end
fclose(s2)
clear s2

A.2 MATLAB-code for the FFT and resonance frequency

clc
figure('units','normalized','position',[0 0 1 1]); \%set fullscreen

\%setting arduino outputs
x_zero=1.71;
y_zero=1.62;
z_zero=1.72;
max_acceleration=19.6;
min_acceleration=-19.6;

x_maxvolt=x_zero+0.6;
x_minvolt=x_zero-0.6;
y_maxvolt=y_zero+0.47; \%smaller sensitivity than 300/g
y_minvolt=y_zero-0.47;
z_maxvolt=z_zero+0.75; \%bigger sensitivity than 300/g
z_minvolt=z_zero-0.75;

volt_to_bit=1024/5.04;
bit_to_volt=1/volt_to_bit;

\% \%find scale bit
\% x_scale=x_zero*volt_to_bit;
\% y_scale=y_zero*volt_to_bit;
\% z_scale=z_zero*volt_to_bit;
%% scale data by zero value
    x_vector_scaled = x_vector - x_scale;
    y_vector_scaled = y_vector - y_scale;
    z_vector_scaled = z_vector - z_scale;

%% find parameters of linear volt_to_acceleration
    x_slope = (max_acceleration - min_acceleration) / (x_maxvolt - x_minvolt);
    x_offset = min_acceleration - x_slope * x_minvolt;
    x_volt = x_vector * bit_to_volt;
    x_acceleration = x_slope * x_volt + x_offset;

    y_slope = (max_acceleration - min_acceleration) / (y_maxvolt - y_minvolt);
    y_offset = min_acceleration - y_slope * y_minvolt;
    y_volt = y_vector * bit_to_volt;
    y_acceleration = y_slope * y_volt + y_offset;

    z_slope = (max_acceleration - min_acceleration) / (z_maxvolt - z_minvolt);
    z_offset = min_acceleration - z_slope * z_minvolt;
    z_volt = z_vector * bit_to_volt;
    z_acceleration = z_slope * z_volt + z_offset;

%% fft for every acceleration
    %x
    length_vector = length(x_acceleration); % length of intervall
    NFFT = 2^nextpow2(length_vector); % Next power of 2 from L
    X = fft(x_acceleration, NFFT) / length_vector; % FFT
    freq = sampling_freq / 2 * linspace(0, 1, NFFT/2+1); % frequency intervall

    %y
    Y = fft(y_acceleration, NFFT) / length_vector; % FFT

    %z
    Z = fft(z_acceleration, NFFT) / length_vector; % FFT

    time_abs = (0 + time_it/it:time_it/it:time_it)'; % convert iterations to absolute time

%% plot Amplitude Spectrum
    subplot(2,3,1)
    plot(freq, 2*abs(X(1:NFFT/2+1))); % plot fft of x_acceleration
    title('Ampl. Spec. of x acceleration')
    xlabel('Frequency (Hz)')
    ylabel('|X(f)|')

    subplot(2,3,2)
    plot(freq, 2*abs(Y(1:NFFT/2+1))); % plot fft of y_acceleration
    title('Ampl. Spec. of y acceleration')
    xlabel('Frequency (Hz)')
    ylabel('|Y(f)|')

    subplot(2,3,3)
    plot(freq, 2*abs(Z(1:NFFT/2+1))); % plot fft of z_acceleration
    title('Ampl. Spec. of z acceleration')
    xlabel('Frequency (Hz)')
Appendix A. Codes 22-Robot

ylabel('|Z(f)|')

% find max amplitudes
fft_offset=10;
% 5Hz -> freq(92)
five_Hz_offset=92;
% 10Hz -> freq(183)
ten_Hz_offset=183;
% find max amplitude index of max amplitude, skip the first 10 values
[x_maxamp,x_indexmaxamp]=max(2*abs(X(fft_offset:NFFT/2+1)));
[y_maxamp1,y_indexmaxamp1]=max(2*abs(Y(fft_offset:five_Hz_offset)));
[y_maxamp2,y_indexmaxamp2]=max(2*abs(Y(five_Hz_offset:ten_Hz_offset)));
[z_maxamp,z_indexmaxamp]=max(2*abs(Z(fft_offset:NFFT/2+1)));

x_omega=2*pi*freq(x_indexmaxamp+fft_offset-1);
y_omega1=2*pi*freq(y_indexmaxamp1+fft_offset-1);
y_omega2=2*pi*freq(y_indexmaxamp2+five_Hz_offset-2);
z_omega=2*pi*freq(z_indexmaxamp+fft_offset-1);

z_freq=freq(z_indexmaxamp+fft_offset-1)
y_freq=freq(y_indexmaxamp1+fft_offset-1)

% plot accelerations
x_ideal_acceleration=x_maxamp*sin(x_omega*time_abs);
y_ideal_acceleration1=y_maxamp1*sin(y_omega1*time_abs);
y_ideal_acceleration2=y_maxamp2*sin(y_omega2*time_abs);
z_ideal_acceleration=z_maxamp*sin(z_omega*time_abs+3)-5;

subplot(2,3,4) % plot x acceleration and fft ideal signal
line(time_abs,x_acceleration)
hold on
line(time_abs,x_ideal_acceleration,'Color','r')
title('Original x acceleration signal and ideal fft signal')
xlabel('Time [s]')
ylabel('Acceleration [m/s^2]')
legend('original signal','ideal signal','Location','Best')

subplot(2,3,5) % plot y acceleration and fft ideal signal
line(time_abs,y_acceleration)
hold on
line(time_abs,(y_ideal_acceleration1+y_ideal_acceleration2),'Color','r')
title('Original y acceleration signal and ideal fft signal')
xlabel('Time [s]')
ylabel('Acceleration [m/s^2]')
legend('original signal','ideal signal','Location','Best')

% hold on %plot +-1g
% line([0 12],[0 0],'Color','g')
% hold on
% line([0 12],[9.81 9.81],'Color','g')
A.3 Arduino-code for gathering data from accelerometer

```
/*
gathering data from acceleration sensor
awaits a read/send command from matlab
and returns sensor values
*/

int x_value = 0; // first analog sensor
int y_value = 0; // second analog sensor
int z_value = 0; // third analog sensor

void setup()
{
    Serial.begin(9600);
}

void loop()
{
    // if we get a valid byte, read analog ins:
    if (Serial.available() >0) {
        Serial.read();

        x_value = analogRead(0);
        //delay(10);
        y_value = analogRead(1);
        //delay(10);
        z_value = analogRead(2);

        Serial.print(x_value);
        Serial.print(',');
        Serial.print(y_value);
        Serial.print(',');
        Serial.println(z_value);
    }
}
```
Appendix B

Codes Two-Beam Mirror Robot

B.1 MATLAB-code for controlling the robot with the keyboard

clc
format compact
fprintf('w: \tincrease speed of the left motor by 1\n')
fprintf('s: \tdecrease speed of the left motor by 1\n')
fprintf('e: \tincrease speed of the right motor by 1\n')
fprintf('d: \tincrease speed of the right motor by 1\n')
fprintf('space: \tdecrease speed of both motors by 5\n')
fprintf('q/a: \tchange direction of the left motor\n')
fprintf('r/f: \tchange direction of the right motor\n')
fprintf('enter: \tget speed of both motors\n')
fprintf('connection establishing, wait for figure window to appear...\n')
fprintf('connection establishing, wait for figure window to appear...\n')

s2=serial('COM5', 'BaudRate', 9600);
open(s2);
pause(2);

%set motor speed as string
% motor_speed='1200';
% fwrite(s2,motor_speed);%send motor speed
% fwrite(s2,10,'char');%send newline comand
% fgetl(s2)

while 1
    w = waitforbuttonpress;
    p = get(gcf, 'CurrentCharacter');

    fwrite(s2,p);
    fwrite(s2,10,'char');

B.2 Arduino-code for controlling the robot with the keyboard

/*
   control of 2-beam robot by using bluetooth connection
   control either by serial monitor or matlab and keyboard
   press upload before releasing reset button
*/

String inString = "";// string to hold input
int inChar = 0;//incoming byte
int motor=0;//general motor variable
int channel_r=9;//analog channel of the right motor
int channel_l=10;//analog channel of the left motor
int speed_right=39;//setting initial speed
int speed_left=39;
int right_motor=channel_r;//renaming channels to motors
int left_motor=channel_l;

int right_direction = 5;//digital channel of right motor
int left_direction = 6;//digital channel of left motor

void setup() {
  // Initialize serial communications:
  Serial.begin(115200);//Arduino BT
  //Serial.begin(9600);
  pinMode(right_direction, OUTPUT);// sets the digital pin as output
  pinMode(left_direction, OUTPUT);// sets the digital pin as output
digitalWrite(right_direction, HIGH);//setting right digital pin to high
digitalWrite(left_direction, HIGH);//setting left digital pin to high
}

void loop() {

  //controlling robot with matlab and keyboard
  //read serial input
  if(Serial.available()
{ inChar = Serial.read();
if (inChar == 28)// left arrow
{
    analogWrite(right_motor, 48);// run right motor only
    analogWrite(left_motor, 0);
}
else if (inChar == 29)// right arrow
{
    analogWrite(right_motor, 0);// run left motor only
    analogWrite(left_motor, 48);
}
else if (inChar == 30)// up arrow
{
    analogWrite(right_motor, speed_right);// run both motors
    analogWrite(left_motor, speed_left);
}
else if (inChar == 31)// down arrow
{
    analogWrite(right_motor, 0);// switch off both motors
    analogWrite(left_motor, 0);
}
else if (inChar == 113)// if q
{
    digitalWrite(left_direction, HIGH);
    // sets the left digital pin high, change direction
}
else if (inChar == 97)// if a
{
    digitalWrite(left_direction, LOW);
    // sets the left digital pin low, change direction
}
else if (inChar == 114)// if r
{
    digitalWrite(right_direction, HIGH);
    // sets the right digital pin high, change direction
}
else if (inChar == 102)// if f
{
    digitalWrite(right_direction, LOW);
    // sets the right digital pin low, change direction
}
else if (inChar == 119)// if w
{
    speed_left = speed_left + 1;// increase left speed by 1
    analogWrite(left_motor, speed_left);
}
else if (inChar == 115)// if s
{
    speed_left = speed_left - 1;// decrease left speed by 1
    analogWrite(left_motor, speed_left);
}
else if (inChar == 101)// if e
{  
speed_right=speed_right+1; // increase right speed by 1  
analogWrite(right_motor, speed_right);  
}
else if(inChar == 100) // if d  
{  
speed_right=speed_right-1; // decrease right speed by 1  
analogWrite(right_motor, speed_right);  
}
else if(inChar == 32) // if space  
{  
speed_right=speed_right-5; // decrease right and left speed by 5  
speed_left=speed_left-5;  
analogWrite(right_motor, speed_right);  
analogWrite(left_motor, speed_left);  
}
else if(inChar == 13) // if enter  
{  
Serial.print("speed_right: "); // print right and left speed  
Serial.print(speed_right);  
Serial.print(" , speed_left: ");  
Serial.println(speed_left);  
}
}

////////////////////////////////////////////////////////////////////////////////
// controlling robot by using serial monitor
// read serial input:
//*while (Serial.available() > 0)  
{
  
inChar = Serial.read();  
Serial.print(inChar);  
Serial.print(", ");  
if(inChar == 'r') motor=channel_r;  
if(inChar == 'l') motor=channel_l;  
if(inChar == 'b') motor=2;  
if (isDigit(inChar))  
{  
  // convert the incoming byte to a char  
  // and add it to the string:  
inString += (char)inChar;  
}

// if you get a newline, print the string,  
// then the string's value:  
if (inChar == '\n')  
{  
  //Serial.print(inChar);  
  Serial.println(inString.toInt());  
  if(motor == 2)  
  {  
    analogWrite(right_motor, inString.toInt());  
    analogWrite(left_motor, inString.toInt());  
  }  
}
```c
}
else
{
    analogWrite(motor, inString.toInt());
}
inString = ""; //clear the string for new input
}
}*/

};

```

B.2. Arduino-code for controlling the robot with the keyboard
Appendix C

Carbon Beam and Plates

Carbon fabric is available in different styles. The two styles used for these components are unidirectional fabric (UD), where all the fibers lay parallel, and twill, where the fibers are interwoven. While UD only can withstand forces in one direction, twill has a quasi-isotropic behaviour. One layer of these fabrics is very thin, several layers need to be applied to achieve a stiff structure. The layers are held together by a special wax which in this case hardens in an oven.

Because the beam only has to withstand the compression, it mainly consists of UD. The top and bottom layer are out of twill to give the beam also stability in other directions. Inside these two layers, six layers of UD are applied. Because carbon structures cannot be shaped after hardening, first a negative form had to be created. Wood (MDF) was used to create this form. Figure C.1 shows this negative form and the carbon fibre wrapped around the wooden wheel.

The plates for the foot should be stiff against bending, here a core is used. This core is out of NOVEX, has a thickness of 3mm and is extremely light but creates a very stiff structure when used with carbon fiber. The core is enclosed by one layer of twill each on the bottom and on top of the core. In order to make the layers connect with each other, the structures must be placed in a vacuum bag and then in an oven, causing the wax to harden. In figure C.2, the plates and the wheel are visible.
Figure C.1: Carbon beam and wooden wheel

Figure C.2: Oven and structures in the vacuum bag
Bibliography


