Control analysis of an active inference controller for a skid-steer mobile robot

An analysis on the performance of the active inference controller focusing on convergence time

B.D. van der Meer

Studentnumber: 4479653

Master of science thesis

TU Delft Delft University of Technology

ALTEN

Control analysis of an active inference controller for a skid-steer mobile robot

An analysis on the performance of the active inference controller focusing on convergence time

Master of science thesis

B.D. van der Meer

Studentnumber: 4479653

December 2020 - February 2022

Supervisors: Prof. Dr. Ir. M. Wisse
A. Vatsyayan

Graduation committee: Prof. Dr. Ir. M. Wisse
Dr. L. Peternel
Y. Tang
A. Vatsyayan

Faculty of Mechanical, Maritime and Materials Engineering (3ME)
Delft University of Technology


TU Delft
Delft
University of
Technology


ALTEN

Preface

Normally I am not the kind of person that would write a preface. However, in this case I would like to write it to be able to thank Aashish Vatsyayan for his help throughout the thesis. You helped me a lot with with the discussions that we had, and offered a fresh view in difficult times. I would also like to thank my supervisor Martijn Wisse, I found our collaboration quite pleasant and I would like thank you for always making time available for me if I needed it. Lastly, I would like to put a some focus on the student group, consisting of Erik, Mitchel and Sjoerd, with whom I developed the active inference controller. I found it a pleasant period and enjoyed working together.

Abstract

Active Inference control is a novel control method based on the free energy principle, which combines action, perception and learning [1][2]. The first Active Inference controller showed promising results on a 7-DOF robot arm for a pick and placing task, however it took nearly six seconds to converge which is too slow [2]. This thesis aims to reduce the convergence time of an Active Inference controller. Therefore an Active Inference controller that is used to control the velocity of a Jackal robot was developed. It was compared to the standard differential drive Controller and it was found that the standard controller outperformed the Active Inference controller on both rise and settling time.

A state space model was derived to obtain a better understanding of the Active Inference controller, for this model the robot was modelled as a point mass. This assumption was confirmed by a step response test of the robot, placing it both on the box and on the ground. It was found that both behaved similar, and thus the wheel ground interactions and internal dynamics of the robot did not affect the convergence time. The state space model was used to find three methods to reduce the convergence time of the Active Inference controller. As a first method the update frequency of the entire controller was increased, for the second method only the update frequency of the inner belief update loop was increased. The last method used an increased update frequency of the belief update loop and the parameters of the system were tuned.

The methods were tested using step response tests and a square wave test, first in a simulated Gazebo environment and later with the actual robot. It was found that the third method reduced the converging time of the Active Inference controller, and reduced it from 0.88s to 0.51s. The improved controller was also tested against the differential drive controller outperformed it. It was found that due to the increased update frequency the tuning parameters could be changed to a wider range of values, this resulted in a shorter convergence time.

List of Figures

List of Tables

Table of Contents

Introduction

Using inference mechanisms in the brain humans can easily adapt to changing or even new circumstances, for robots this is more complicated. To solve this, bio-inspired design is used in combination with techniques such as machine learning, predictive coding and neural networks [4][5]. However, these methods still lack adaptation to completely new circumstances as they are generally trained for specific data sets or situations. A novel method in robotics focuses on Active Inference, which is a neuro-scientific theory that tries to explain the working principle of the brain [6]. The theory if correctly adapted to robotics, could potentially lead to true artificial intelligence.

Active Inference is based on the free energy principle and accounts for action, perception and learning [1]. The theory is founded by the fact that all living organisms try to minimize the free energy to be able to minimize surprise. The free energy is a measure of the difference between the organisms internal model of the world and the observed real world. It can either be minimized by adapting the internal model to the world (perception), or changing the world to be more like the internal model (action) [6]. More information about the free energy formula as used by Active Inference can be found in [6][7]. This novel way of combining action and perception could be a powerful tool in robotics and therefore [8] explored the implementation of an Active Inference controller. Founded by this research, [2] created an Active Inference controller that controlled a robot arm.

In the previously created Active Inference controllers it was noted that it took approximately seven seconds to pick up a bottle[1], which is more than necessary [2]. This thesis aims to find a method to decrease the convergence time of an Active Inference controller. Therefore in collaboration with three other students an Active Inference velocity controller for a mobile robot was created. The controller was compared to the original controller of the mobile robot to compare the performance for a step response and it was found to be slower, which understates the problem.

[1] This research will be analyzed in chapter 2

1-1 Research goal

In the previous section Active Inference control was introduced, it was noted that the convergence time of an Active Inference controller is insufficient. This leads to the main research question for this thesis: **How can the response time of an Active Inference controller be decreased, while remaining an accurate controller?**

To be able to achieve this goal a list of sub goals was created:

- Develop an Active Inference velocity controller to control a Jackal robot[2].
- Can a state space model be used to analyze and simulate an Active Inference controller?
- Can the Gazebo environment[3] be used to simulate the Active Inference controller for a Jackal robot?
- Does increasing the update frequency of the entire system, or a part of the system, have an effect on the accurateness of the controller?
- Does increasing the update frequency of the system or part of the system affect the tuning parameters of the Active Inference controller?
- What are the maximum values that can be used for the tuning parameters?

1-2 Structure of the thesis

This thesis is structured in the following way: In chapter 2 Active Inference control for robotics will be discussed, followed by a short description of the convergence time problem of an Active Inference controller. Chapter 3 introduces the Jackal Robot, Gazebo environment and implemented Active Inference, finally the active inference controller will be compared to the differential drive controller of the Jackal. In Chapter 4 a state space model of the Active Inference controller will be derived. Based on the state space model different methods to decrease the convergence time will be proposed. Chapter 5 will show the results from the performed experiments followed by an analysis of these results. Lastly in chapter 6 the answers on the research questions will be given and some future work will be discussed.

[2]The Jackal robot will be used throughout this thesis and will be introduced in chapter 3
[3]The Gazebo environment will be introduced in chapter 3

Active Inference control

In this chapter Active Inference control for robotics will be introduced, starting with an overview in section 2-1. Followed by the introduction of the free energy formula in section 2-2 and the generative models in section 2-3. Sections 2-4 and 2-5 will discuss the generalized motions and update rules respectively. Lastly in section 2-6 the convergence time issue as found in literature will be highlighted.

2-1 Overview of Active Inference control for robotics

A short introduction on Active Inference was given in the introduction, this section will provide an overview of the Active Inference controller as implemented by Pezzato et al. [2]. This controller was used to control a 7-DOF robot arm for a pick and place task, a block diagram based on this control scheme can be found in figure 2-1. The closed loop system consists of the plant and the controller, the output of the plant is generalized in the generalization step where higher order derivatives of the measured states are estimated. The generative model of the sensory data, which estimates the measured states, is compared to the output of the plant to obtain an error term ε_y. In the free energy function a second error term is used, ε_μ, which follows form the generative model of the desired states. This model is used to evaluate the state dynamics, and to observe whether the desired state is reached. In the update step one iteration of a gradient descent is used on the free energy formula to compute the state update (belief update) and the action update. The belief update is used to update the believed states, which are used in the previously described generative models. The action update is used to update the control action update, which is used as input for the plant. The belief and action update have to be integrated to obtain the actual beliefs and actions, which is done using a forward Euler method.

2-2 The free energy principle for Active Inference control

This section aims to introduce the free energy formula (equation 2-1) for Active Inference control, for a full derivation of this formula [2] can be used. The free energy (F) is a summation of the error terms, ε_y and ε_μ which will be discussed later, up to the n_d^{th} order of generalized motions. The precision matrices, $\Sigma^{-1}_{y^{(i)}}$ and $\Sigma^{-1}_{\mu^{(i)}}$ can be used to adjust the confidence of the

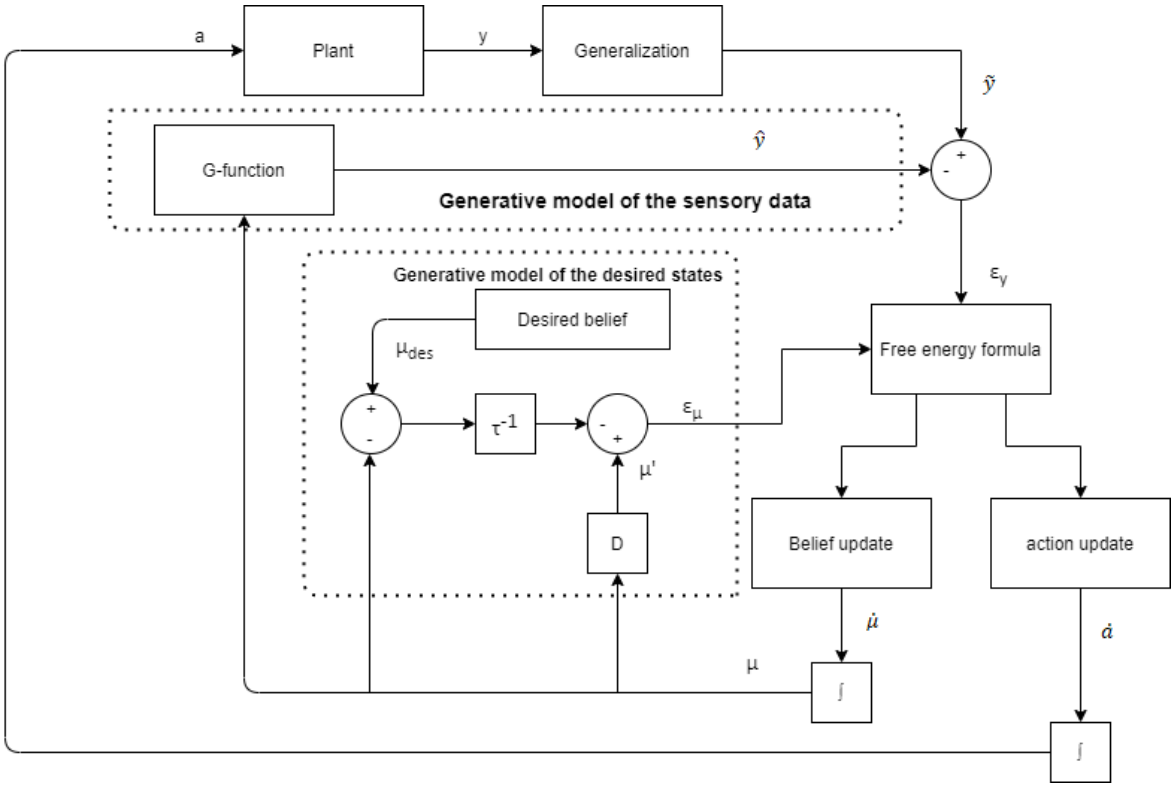


Figure 2-1: The closed loop system of an Active Inference controller and plant is shown. The Active Inference controller provides a control input, a, for the plant, this control input must be sampled. The measurements (y) are the output of the plant these are generalized to obtain the generalized measurements $\widetilde{y}$. The generative model of the sensory data is used to predict the states ($\hat{y}$), these will be subtracted from $\widetilde{y}$ to obtain the error term ε_y. In the generative model of the sensory data an error term (ε_μ) that represents the difference between the desired state (or beliefs) and the estimated state is calculated. This error term is multiplied by a scaling parameter τ^{-1}. The error terms are combined in the free energy formula and a gradient descent is used to calculate the belief and action updates, $\dot{\mu}$ and $\dot{a}$ respectively.

controller in either the estimated states (beliefs) or the measurements. K is a constant and will be neglected for this analysis, as it does not affect the minimization of free energy.

$$F = -\frac{1}{2}\sum_{i=0}^{n_d}[\varepsilon_y^{(i)}\Sigma_{y^{(i)}}^{-1}\varepsilon_y^{(i)} + \varepsilon_\mu^{(i)}\Sigma_{\mu^{(i)}}^{-1}\varepsilon_\mu^{(i)}] + K \tag{2-1}$$

2-3 Generative model

The generative model in an Active Inference controller consists of two sub models, the generative model of the desired states that compares the estimated state to the desired state and the generative model of the sensory data that is used to predict the new state. Both will be discussed throughout this section.

2-3-1 Generative model of the desired states

The generative model of the state dynamics (equation 2-2), models the evolution of the states (μ) towards the desired state (μ_d). Gaussian white (w) noise with a mean of zero and a variance of Σ was introduced [7]. In the remainder of the thesis this variance will be called Σ_μ to denote that it is the variance of the generative model of the desired states. Current Active Inference controllers assume an attractor dynamics model (equation 2-3) for $f(\mu)$, assuming that the states evolve towards the desired state [2][9][10]. The temporal parameter τ^{-1} was introduced by Baioumy et al. to decrease the rise and convergence time of the controller [11]. This parameter is in earlier implementations essentially set to 1, however Baioumy found that decreasing this parameter sped up the controller.

$$\mu' = f(\mu) + w \tag{2-2}$$

$$f(\mu) = \tau^{-1}(\mu_d - \mu) \tag{2-3}$$

Substituting equation 2-3 into equation 2-2 results in the generative model for the state dynamics (equation 2-4). From this equation a prediction error term ε_μ can be found, 2-5, for a derivation see [7]. The error term contains information about the dynamics of the system, if the derivatives of the beliefs are equal to the term $\tau^{-1}(\mu_d - \mu)$ the system is converged and the error term is zero. Interestingly, decreasing τ increases the value of the model $f(\mu)$ and creates an inaccurate model. By increasing $f(\mu)$ the error term, ε_μ, will be increased as well. This results in a larger contribution to the free energy formula which is counter-intuitive as the goal is to minimize the free energy. However, the larger free energy will result in a larger belief and control action step, which will help the system to converge faster.

$$\mu' = \tau^{-1}(\mu_d - \mu) + w \tag{2-4}$$

$$\varepsilon_\mu = \mu' - \tau^{-1}(\mu_d - \mu) \tag{2-5}$$

2-3-2 Generative model of the sensory data

State estimation in Active Inference is performed using the generative model of the sensory data as expressed in equation 2-6, z is Gaussian noise with a mean of zero and a variance of $(0, \Sigma)$, this variance will be denoted with Σ_y [7]. The model $g(\mu)$ is used to predict the evolution of the sensory data and maps the believed variables to the measured variables. In the first implementations $g(\mu)$ is chosen to be a linear mapping, this can however be changed to a non linear mapping to obtain a more accurate model.

$$\hat{y} = g(\mu) + z \tag{2-6}$$

From equation 2-6 equation 2-7 can be found, with the prediction error ε_y [7]. This error compares the measured states with the estimated, or believed, states if the error is zero the both are equal.

$$\varepsilon_y = \hat{y} - g(\mu) \tag{2-7}$$

2-4 Generalized motions

Friston introduced generalized motions to describe the beliefs about the dynamics of the states [12]. With the introduction of generalized motions, higher order derivatives of the states are used which contain information about the state dynamics. For example if one would estimate both the velocity, acceleration and jerk then jerk would give insights on the evolution of the acceleration and acceleration on the evolution of velocity. Generalized motions will be denoted by a tilde over a state, for example $\widetilde{y}$ (equation 2-6) which is further specified in equation 2-8 for one generalized order [2].

$$\begin{aligned} y &= g(\mu) + z \\ y' &= \frac{\delta g}{\delta \mu}\mu' + z' \\ \widetilde{y} &= \begin{bmatrix} y \\ y' \end{bmatrix} \end{aligned} \tag{2-8}$$

Sensors measure specific quantities such as, acceleration or velocity, other quantities such as jerk have to be estimated. A method was proposed by Heijne et al. [13], for this method the time difference between two measurements h is used to estimate higher generalized orders.

2-5 Update rules

Active Inference minimizes the free energy by using two gradient descents, the first to update the beliefs (equation 2-9) and the second to update the actions (equation 2-10) [6][14]. The action update consists of two parts, $\frac{\delta y}{\delta \widetilde{a}}$ and $\frac{\delta F}{\delta y}$, the former is the change in sensory input with respect to the control action the latter the change of the free energy with respect to the sensory input. In the earlier Active Inference implementations the former is approximated, in chapter 3 this will be done for the implemented Active Inference controller [2]. Two scaling parameters k_μ and k_a are introduced which can be used to change the step size [2][8].

$$\dot{\widetilde{\mu}} = \frac{d}{dt}\widetilde{\mu} - k_\mu \frac{\delta F}{\delta \widetilde{\mu}} \tag{2-9}$$

$$\dot{a} = -k_a \frac{\delta F}{\delta \widetilde{a}} = -k_a \frac{\delta y}{\delta \widetilde{a}} \frac{\delta F}{\delta y} \tag{2-10}$$

The calculated belief and action values have to be integrated to be able to use them as control inputs, this is done using the Forward Euler method [2][3]. Equation 2-11 shows the general formula that is used for the integration of the action step, the beliefs are integrated in a similar fashion [15]. In the equation h is the time difference between the steps, which depends on the update frequency of the system.

$$\widetilde{a}_t = \widetilde{a}_{t-1} + h * \dot{\widetilde{a}} \tag{2-11}$$

2-6 Convergence time of an Active Inference controller

The first implemented Active Inference controller was developed for a pick and place task for a 7-DOF robot arm [2], figure 2-2 displays the results. For this experiment the robot arm moved between two points, q_a and q_b defined below, it can be seen that it takes approximately 6 seconds to converge to the desired positions. Considering the fact that the positions are relatively close to each other this takes too long.

- q_a = [1, 0.5, 0, -2, 0, 2.5, 0] [rad]
- q_b = [0, 0.2, 0, -1, 0, 1.2, 0] [rad]

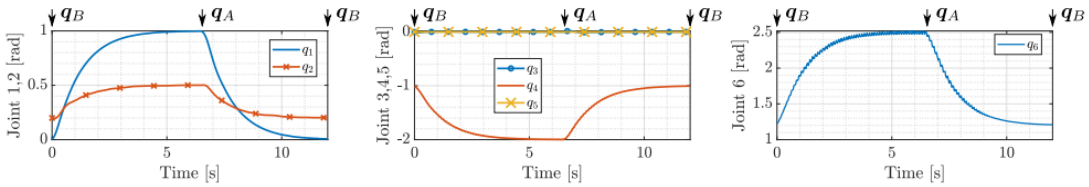


Figure 2-2: Evolution of the joint states during the pick and place task [2].

Based on the work of Pezzato et al. [2] the work of Baioumy et al. was initiated [3]. They introduced the temporal parameter to reduce the settling time of the robot arm and this resulted in the results of figure 2-3. Note that slightly different positions were used however, the same robot arm was used for a pick and placing task. It can be seen that the settling time is reduced yet the accuracy decreased as ringing occurred in the system. The settling time was decreased by 50%, however the system converged in approximately 3 seconds which is still to slow. The increase of speed introduced speed both ringing and overshoot into the system, which negatively affects the performance.

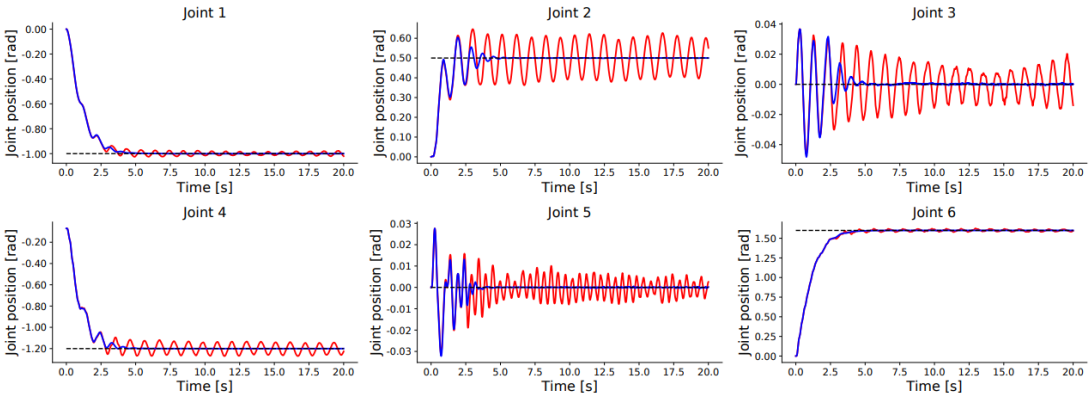


Figure 2-3: Evolution of the joint states during the pick and place task using the controller as implemented by Baioumy et al. [3]

Experimental setup

This chapter will elaborate on the experimental setup starting with a short introduction on the used robot in section 3-1. This will be followed by an introduction of the simulation environment, Gazebo, in section 3-2. An overview of the implemented active inference controller will be given in section 3-3 and in section 3-4 this controller will be compared to the standard controller of the Jackal.

3-1 Jackal robot

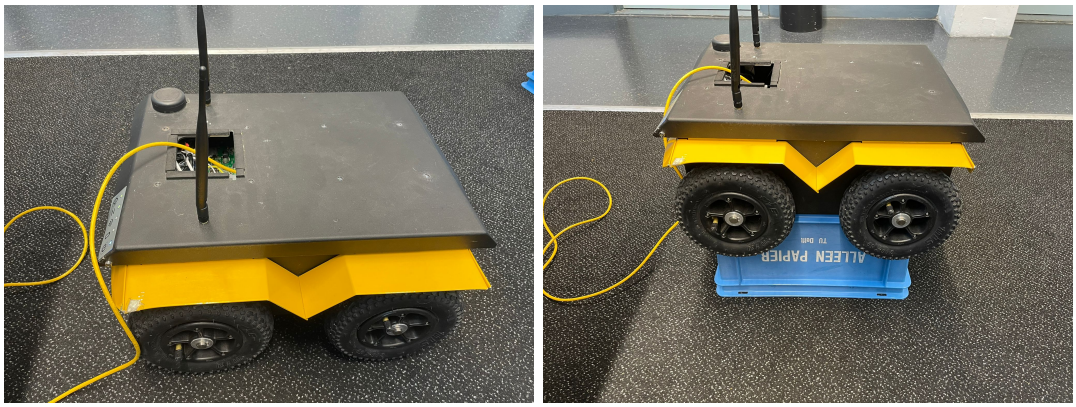
During this thesis a Jackal robot from the TU Delft cognitive robotics lab will be used. the Jackal is a four wheeled skid steering autonomous robot developed by Clearpath. In figure 3-1 the Jackal is shown, both as it is placed on the ground and as placed on a box. To communicate with the Jackal the ROS 1[1] environment is used, and the robot is connected to a computer using a LAN cable. An extensive Github containing all information of the Jackal and ROS can be found in[2].

The linear and angular velocity of the Jackal is measured using the internal IMU and the wheel encoders and is published at a frequency of 50 Hz. The Jackal currently makes use of the standard differential drive controller and has a maximum forward and backward speed of 2.5 m/s. More information on the Jackal can be found on the website of clearpath.[3]

3-2 Gazebo simulation environment

The Gazebo environment will be used to simulate the Jackal in combination with the active inference controller. In figure 3-2 a screenshot of the Jackal in the Gazebo environment can be found. The Gazebo simulation makes use of the ODE physics engine and the model of the Jackal was provided by Clearpath[4]. In the Gazebo environment step response tests will be performed to see whether it can be used to simulate the Jackal.

[1] https://www.ros.org/
[2] https://github.com/jackal
[3] https://clearpathrobotics.com/jackal-small-unmanned-ground-vehicle/
[4] https://www.clearpathrobotics.com/assets/guides/kinetic/jackal/simulation.html



(a) The Jackal robot **(b)** The Jackal robot as it is placed on a box

Figure 3-1: The Jackal robot that was used for the experiments throughout this thesis. The left figure shows the Jackal on the ground, the right figure shows the Jackal on the box for the tests on the box. The jackal is connected to the PC using the yellow LAN cable.

3-3 Implemented active inference controller

In collaboration with three other students an active inference controller that controlled the velocity of the Jackal was developed. During this process I focused on section 3-3-3 and later the implementation in C++ and the synchronization of the incoming data streams. The following section were written in collaboration with the other students and presents an overview of the controller and elaborates on the choice for the generative model of the sensory data. The full code used for the controller can be found in the following Gitlab[5].

3-3-1 Overview of the active inference controller

The controller for the Jackal robot needs to send and receive information to and from the plant. This communication is handled by ROS, which is inherently present on the Jackal robot. There are many topics available, but for the controller only a few are needed. An overview is shown in Figure 3-3, the block diagram in the Active Inference controller part can be found in appendix B. Sensory input from the wheel speeds are published on the topic /joint_states, while the speed of the center of mass μ is published on /imu/data, containing $\ddot{x}$, $\ddot{y}$ and $\dot{\theta}$. The sensory data are read by a generalization node that calculates the derivatives of the wheel speeds. These derivatives are then published on an intermediate topic: /sensor_generalized. These generalized measurements, combined with the desired velocity of the center of mass that is published on /jackal_AIC/set_point, are the actual input for the Active Inference controller. The Active Inference controller calculates the action required to bring down the Free Energy and publishes drive commands on /cmd_drive when using the robot and /jackal_left_wheel/command and /jackal_right_wheel/command when using Gazebo. These topics contain the velocities that the left- and right wheel of the Jackal should have: ω_L and ω_R. Currently an internal lower-level controller transforms the drive commands to actuator voltages.

[5] https://gitlab.tudelft.nl/active-inference-drive-control/active-inference-drive-control

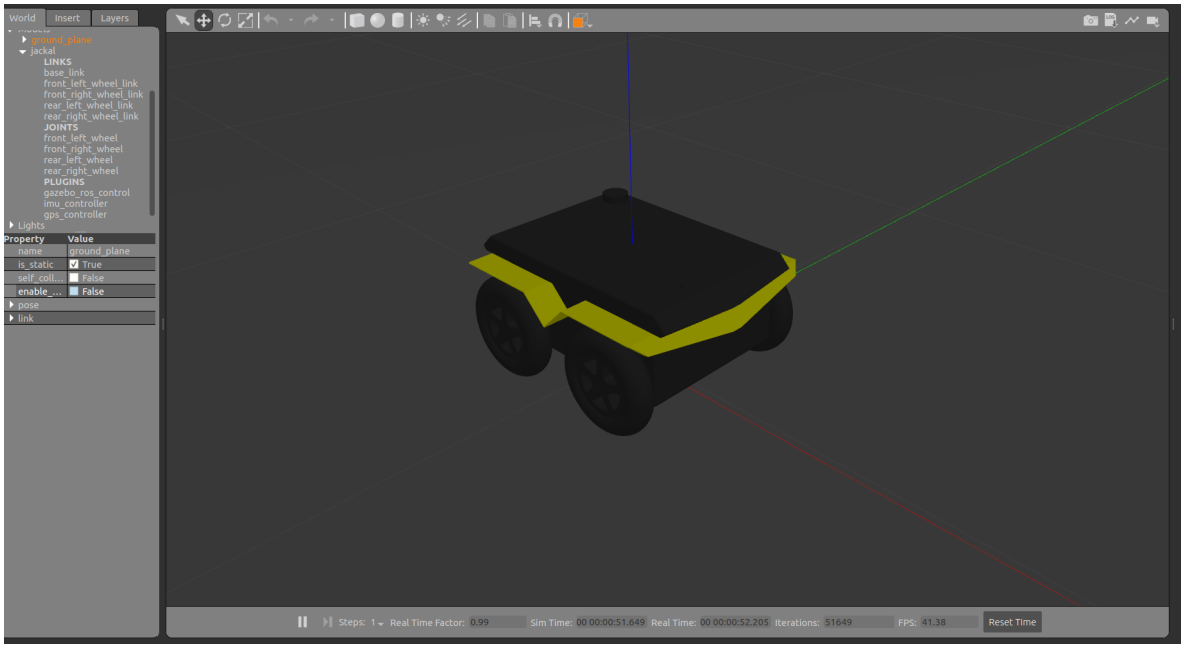


Figure 3-2: The Jackal in the Gazebo environment.

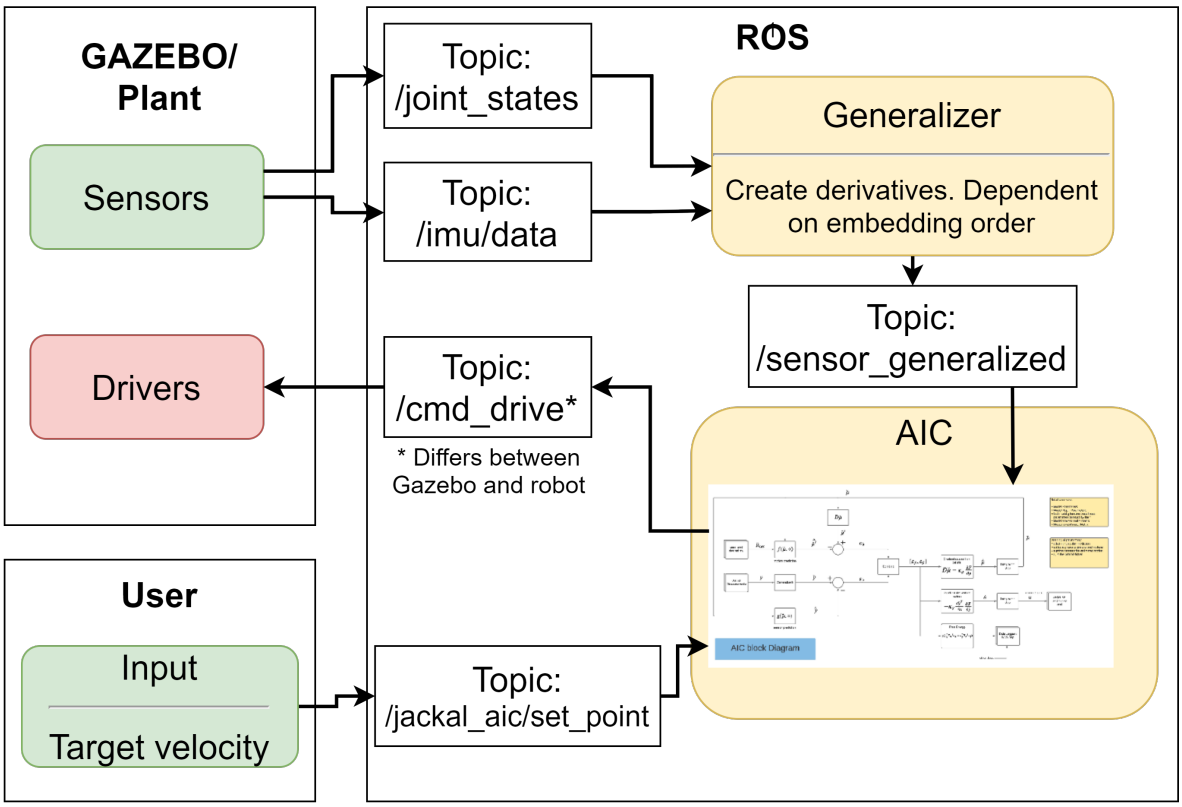


Figure 3-3: An overview of all components involved and through what ROS topics they communicate.

3-3-2 Generative model for the sensory data

The generative model for the sensory data was discussed in section 2-3-2, this section will focus on the choice of the model $g(\mu)$. For this model there can be chosen between a simple kinematic mapping or a dynamic model. The kinematic mapping maps the desired body velocity input by the user (linear and angular) to corresponding wheel speed inputs to the wheels and is often preferred [16][17][18][19][20]. The dynamic model takes takes the forces and accelerations on the vehicle into account. This will generally lead to an increase in control performance because the vehicle is modeled on a deeper level. However, this type of modeling requires an elaborate understanding on the ground-wheel interactions of the robot. As the main goal of this implementation was to implement a first active inference controller for the Jackal it was decided to keep the system as simple as possible, thus it was opted to use a simple kinematics mapping. The extended differential drive model introduced by [16] was used, as this is the simplest and most used kinematic mapping available, while still giving sufficient performance:

$$\mu = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = r\alpha \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ 0 & 0 \\ -\frac{1}{\hat{b}} & \frac{1}{\hat{b}} \end{bmatrix} \begin{bmatrix} \omega_L \\ \omega_R \end{bmatrix} \tag{3-1}$$

$$G = r\alpha \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ 0 & 0 \\ -\frac{1}{\hat{b}} & \frac{1}{\hat{b}} \end{bmatrix} \tag{3-2}$$

Where α is a slip parameter and $\hat{b}$ is the virtual width of the vehicle (as if it had two wheels), these are both trained empirically and where found to be 1 and 0.5621 respectively. The radius of the wheels, r, is 0.098. The generative model $g(\mu)$ maps the states to the measurements. The used implementation has access to the gyroscope, wheel encoders and accelerometer. The used model can be found in equation 3-3, where G^{-1} is the pseudo-inverse of the kinematic mapping.

$$y = \begin{bmatrix} \dot{\theta} \\ \omega_L \\ \omega_R \\ \ddot{x} \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ G_{11}^{-1} & G_{12}^{-1} & G_{13}^{-1} \\ G_{21}^{-1} & G_{22}^{-1} & G_{23}^{-1} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} \tag{3-3}$$

3-3-3 Free energy formulation

This section will focus on the definition of the free energy formula (as introduced earlier in equation 2-1) for the active inference velocity controller of the Jackal robot. In the previous paragraph the generative model of the sensory data and of the state dynamics are defined. The error terms, ε_y and ε_μ, can be derived based on the generative models shown before, these can be found in equations 3-4 and 3-5. To find the first generalized order these equations are

differentiated once to obtain equations 3-6 and 3-7. Note that in this example only the first derivative is shown, in the actual application the derivatives can go up to the 5th order.

$$\varepsilon_y = y - g(\mu) = y - R\mu = \begin{bmatrix} \dot{\theta} \\ \omega_L \\ \omega_R \\ \ddot{x} \\ \ddot{y} \end{bmatrix} - \begin{bmatrix} 0 & 0 & 1 \\ \frac{1}{r\alpha} & 0 & \frac{-\hat{b}}{2r\alpha} \\ \frac{1}{r\alpha} & 0 & \frac{\hat{b}}{2r\alpha} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} \tag{3-4}$$

$$\varepsilon_\mu = \mu' - f(\mu) \tag{3-5}$$

$$\begin{aligned} \varepsilon_y &= \tilde{y} - R\mu \\ \varepsilon_y' &= \tilde{y}' - R\mu' \\ &\vdots \\ \varepsilon_y^{(5)} &= \tilde{y}^{(5)} - R\mu^{(5)} \end{aligned} \tag{3-6}$$

$$\begin{aligned} \varepsilon_\mu = \mu' - f(\mu) &= \mu' - (\mu_d - \mu)\tau^{-1} \\ \varepsilon_\mu' &= \mu'' + \mu'\tau^{-1} \\ &\vdots \\ \varepsilon_\mu^{(5)} &= \mu^{(6)} + \mu^{(5)}\tau^{-1} \end{aligned} \tag{3-7}$$

For readability, equations 3-6 and 3-7 can be written in matrix form (equation 3-8). Furthermore, the same can be done for the precision matrices, which is done in equation 3-9. Note that in this case $\widetilde{\varepsilon}$ and $\widetilde{\Pi}$ consist of all generalized orders.

$$\widetilde{\varepsilon} = \begin{bmatrix} \widetilde{\varepsilon}_y \\ \widetilde{\varepsilon}_\mu \end{bmatrix} \tag{3-8}$$

$$\widetilde{\Pi} = \begin{bmatrix} \widetilde{\Sigma}_y & 0 \\ 0 & \widetilde{\Sigma}_\mu \end{bmatrix} \tag{3-9}$$

The free energy formula was defined in equation 2-1, if one makes use of equations 3-8 and 3-9 the free energy function can be more compactly written to equation 3-10. Note that $\widetilde{\varepsilon}$ contains all error terms and $\widetilde{\Pi}$ all precision matrices.

$$F = \frac{1}{2}\widetilde{\varepsilon}^T\widetilde{\Pi}\widetilde{\varepsilon} \tag{3-10}$$

From the free energy formula the belief update can be found, the belief update is defined in equation 3-11. In equation 3-12 this has been written out for 2 generalized orders.

$$\dot{\tilde{\mu}} = D\tilde{\mu} - k_\mu(\frac{\delta\tilde{\varepsilon}_\mu}{\delta\tilde{\mu}}\frac{\delta F}{\delta\tilde{\varepsilon}_\mu} + \frac{\delta\tilde{\varepsilon}_y}{\delta\tilde{\mu}}^T\frac{\delta F}{\delta\tilde{\varepsilon}_y}) \tag{3-11}$$

$$\begin{bmatrix}\dot{\mu}\\ \dot{\mu}'\\ \dot{\mu}''\end{bmatrix} = \begin{bmatrix}0 & I & 0\\ 0 & 0 & I\\ 0 & 0 & 0\end{bmatrix}\begin{bmatrix}\mu\\ \mu'\\ \mu''\end{bmatrix} - k_\mu(\begin{bmatrix}\frac{I}{\tau} & 0 & 0\\ I & \frac{I}{\tau} & 0\\ 0 & I & \frac{I}{\tau}\end{bmatrix}\begin{bmatrix}\Pi_\mu\varepsilon_\mu\\ \Pi_{\mu'}\varepsilon_{\mu'}\\ \Pi_{\mu''}\varepsilon_{\mu''}\end{bmatrix} - \begin{bmatrix}R^T & 0 & 0\\ 0 & R^T & 0\\ 0 & 0 & R^T\end{bmatrix}\begin{bmatrix}\Pi_y\varepsilon_y\\ \Pi_{y'}\varepsilon_{y'}\\ \Pi_{y''}\varepsilon_{y''}\end{bmatrix}) \tag{3-12}$$

Lastly the action update is defined in equation 3-13. This has been written out for 2 generalized orders of motion in equation 3-15. Note that the term $\frac{\delta\tilde{\varepsilon}_o}{\delta a}$ is replaced by the forward model (K) as defined in equation 3-14.

$$\dot{a} = -\kappa_a\frac{\delta\tilde{\varepsilon}_y}{\delta a}\frac{\delta F}{\delta\tilde{\varepsilon}_y} \tag{3-13}$$

$$K = \begin{bmatrix}\frac{-r\alpha}{\hat{b}} & 1 & 0 & 0 & 0\\ \frac{r\alpha}{\hat{b}} & 0 & 1 & 0 & 0\end{bmatrix} \tag{3-14}$$

$$\dot{a} = -\kappa_a\begin{bmatrix}K & 0 & 0\\ 0 & K & 0\\ 0 & 0 & K\end{bmatrix}\begin{bmatrix}\Pi_y\varepsilon_y\\ \Pi_{y'}\varepsilon_{y'}\\ \Pi_{y''}\varepsilon_{y''}\end{bmatrix} \tag{3-15}$$

The actual belief and action values are calculated using the forward euler method:

$$\mu_k = \mu_{k-1} + h\dot{\mu}_k \tag{3-16}$$

3-3-4 Generalization

Active Inference requires the measurements to be generalized, the states of μ are generalized by taking multiple derivatives. However, this is not as straight forward for generalizing measurements, there is only a limited amount of derivatives sensors can measure. For example if one has a position sensor, a velocity sensor and an acceleration sensor this would mean that one can only measure up to an embedding order of two. If one needs higher order measurements, these synthesized from the available measurements. To do so an estimation method based off the Master's thesis of I.L. Hijne [21] will be used. A robot receives measurement samples in discrete time, which make it possible to use finite differences.

Finite differences

It is possible to extract higher order derivatives from a Taylor expansion. The Taylor expansion computes signal y_{k+j} from y_k and a weighted sum of y_k's higher order derivatives and is defined as:

$$y_{k+j} = \sum_{i=0}^{i_{max}} \frac{(jh)^i}{i!} y_k^{(i)} \tag{3-17}$$

where (i) denotes the i-th order derivative of y_k and h is an interval step that is sufficiently small. Note that i_{max} goes to infinity and j is used for looking j steps forward/backwards and note that $0^0 = 1$.

The goal here is to extract every order of derivatives up to a defined i_{max}. So for the 1st order we can write the Taylor expansion as:

$$\begin{aligned} y_{k-1} &= y_k - h y_k^{(1)} + \mathcal{O}(h^2) \\ y_k^{(1)} &= \frac{1}{h}(y_k - y_{k-1}) + \mathcal{O}(h) \end{aligned} \tag{3-18}$$

Here, $\mathcal{O}(h^n)$ denotes the order of magnitude of the error terms. Similarly, we can do this approximation for the second derivative by eliminating the terms until we only end with the second order terms on the right side.

When

$$\begin{aligned} y_k &= y_k \\ y_{k-1} &= y_k - h y_k^{(1)} + \frac{1}{2} h^2 y_k^{(2)} + \mathcal{O}(h^3) \\ y_{k-2} &= y_k - 2h y_k^{(1)} + 2h^2 y_k^{(2)} + \mathcal{O}(h^3) \end{aligned}$$

are combined linearly to eliminate all but the second order terms we find.

$$\frac{1}{2} y_k - 1 y_{k-1} + \frac{1}{2} y_{k-2} = \frac{1}{2} h^2 y_k^{(2)} + \mathcal{O}(h^3) \tag{3-19}$$

This can then be solved for $y_k^{(2)}$:

$$y_k^{(2)} = \frac{1}{h^2}(y_{k-2} - 2y_{k-1} + y_k) + \mathcal{O}(h) \tag{3-20}$$

and has the coefficients $c = [1, -2, 1]$.

The error term still scales linearly with the step h, this error term will be the same for higher order derivatives as well. However, this error term can be reduced by taking more past samples into consideration. For example if we took 4 past samples for calculating $y_k^{(2)}$, then the error term scales quadratic with the interval step h: $\mathcal{O}(h^2)$.

The problem turns into finding the appropriate coefficients of the linear combinations of Taylor series. I.L. Hijne [21] tabulated these coefficient up to the 5th order derivative for an accuracy of $O(h)$. The coefficients for zero to higher order derivatives can be written in a 6×6 matrix and are shown in eq (3-21).

Matrix forms

In Active Inference the generalized coordinates of the sensory input for an embedding order p are:

$$\tilde{\mathbf{y}} = [y_k^{(0)}, y_k^{(1)}, ..., y_k^{(p)}]$$

We can approximate $\tilde{\mathbf{y}}$ by using backwards differentiation of numerous previous samples of $y_k^{(0)}$. This can be written as:

$$\tilde{\mathbf{y}} = E\check{\mathbf{y}}, \qquad \check{\mathbf{y}} = [y_{k-s}, y_{k-s+1}, ..., y_k]$$

In this equation s is the amount of samples backwards. For $p = 5$, $s = 5$ and an error term of $\mathcal{O}(h)$ the system of equations looks as follows:

$$\begin{bmatrix} y_k^{(0)} \\ y_k^{(1)} \\ y_k^{(2)} \\ y_k^{(3)} \\ y_k^{(4)} \\ y_k^{(5)} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -\frac{1}{h} & \frac{1}{h} \\ 0 & 0 & 0 & \frac{1}{h^2} & -\frac{2}{h^2} & \frac{1}{h^2} \\ 0 & 0 & -\frac{1}{h^3} & \frac{3}{h^3} & -\frac{3}{h^3} & \frac{1}{h^3} \\ 0 & \frac{1}{h^4} & -\frac{4}{h^4} & \frac{6}{h^4} & -\frac{4}{h^4} & \frac{1}{h^4} \\ -\frac{1}{h^5} & \frac{5}{h^5} & -\frac{10}{h^5} & \frac{10}{h^5} & -\frac{5}{h^5} & \frac{1}{h^5} \end{bmatrix} \begin{bmatrix} y_{k-5} \\ y_{k-4} \\ y_{k-3} \\ y_{k-2} \\ y_{k-1} \\ y_k \end{bmatrix} \tag{3-21}$$

This matrix does solve the problem and we get the generalized coordinates of the sensory input with a constant h. However, to be able to implement this it is required to keep track of old data samples, which causes the information to overlap. The equation can be rewritten to only contain data of the samples of interest and previous samples plus its derivatives. A special vector $\check{\tilde{\mathbf{y}}}$ can be constructed that contains the current sample and the previous sample with higher order derivatives of the previous sample. This form is initialized with the higher order derivatives at zero.

$$\tilde{\mathbf{y}} = Q\check{\tilde{\mathbf{y}}}, \qquad \check{\tilde{\mathbf{y}}} = [\mathbf{y}_k, \mathbf{y}_{k-1}^\top]^\top$$

$$\begin{bmatrix} y_k^{(0)} \\ y_k^{(1)} \\ y_k^{(2)} \\ y_k^{(3)} \\ y_k^{(4)} \\ y_k^{(5)} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{h} & -\frac{1}{h} & 0 & 0 & 0 & 0 \\ \frac{1}{h^2} & -\frac{1}{h^2} & -\frac{1}{h} & 0 & 0 & 0 \\ \frac{1}{h^3} & -\frac{1}{h^3} & -\frac{1}{h^2} & -\frac{1}{h} & 0 & 0 \\ \frac{1}{h^4} & -\frac{1}{h^4} & -\frac{1}{h^3} & -\frac{1}{h^2} & -\frac{1}{h} & 0 \\ \frac{1}{h^5} & -\frac{1}{h^5} & -\frac{1}{h^4} & -\frac{1}{h^3} & -\frac{1}{h^2} & -\frac{1}{h} \end{bmatrix} \begin{bmatrix} y_k \\ y_{k-1}^{(0)} \\ y_{k-1}^{(1)} \\ y_{k-1}^{(2)} \\ y_{k-1}^{(3)} \\ y_{k-1}^{(4)} \end{bmatrix} \tag{3-22}$$

Implementation Summary

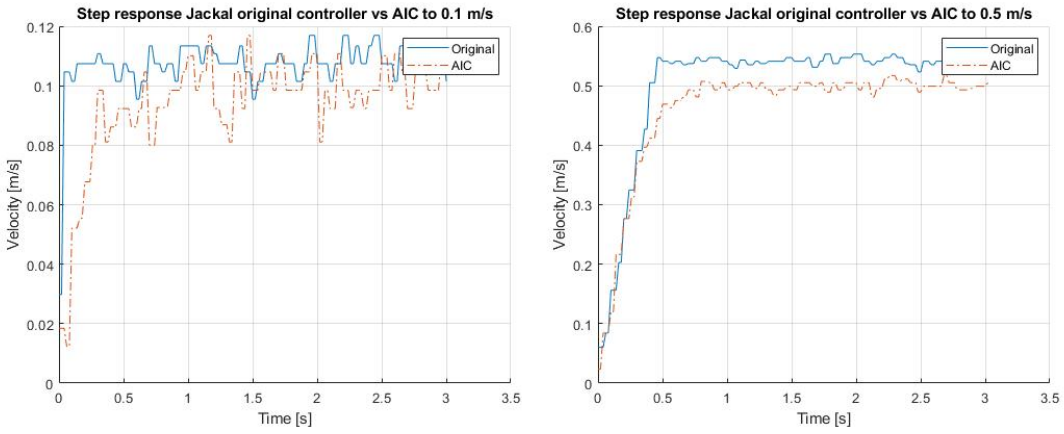
The form eq (3-22) is used in the Jackal robot. The sensor measurements are synchronized so that h is the same for all measurements during a time step. Their derivatives are computed based on eq (3-22). However instead of making one large matrix that matches the size of $\check{\check{\mathbf{y}}}$, it is more efficient to split this $\check{\check{\mathbf{y}}}$ into several single vectors of a measurement $\check{y}_{k,i} = [y_{k,i}, y^{(0)}_{k-1,i}, y^{(1)}_{k-1,i}, y^{(2)}_{k-1,i}, y^{(3)}_{k-1,i}, y^{(4)}_{k-1,i}]$, where each $\check{y}_{k,i}$ is a unique sensor measurement denoted by i. Then use these smaller vectors and the 6×6 matrix Q to calculate its derivatives as:

$$\tilde{y}_{k,i} = Q\check{y}_{k,i} \qquad \check{y}_{k,i} = [y_{k,i}, y^{(0)}_{k-1,i}, y^{(1)}_{k-1,i}, y^{(2)}_{k-1,i}, y^{(3)}_{k-1,i}, y^{(4)}_{k-1,i}]$$

This way Q only has to be calculated once and can be reused for every vector of $\check{y}_{k,i}$.

3-4 differential drive control vs active inference control a step response analysis

This thesis makes use of the controller described above, which controls the velocity of a Jackal robot. This controller will therefore also be tested to see how it's settling time is compared with the standard (differential drive[6]) controller implemented in the Jackal, which is the goal of the following experiment. Therefore three different step responses were performed, for each experiment the jackal was driven on the ground. The resulting velocities for these steps can be found in figures 3-4 and 3-5. In this experiment the basic active inference controller was used, using the parameters as written in appendix C.



(a) Step to 0.1 m/s **(b)** Step to 0.5 m/s

Figure 3-4: Step responses of the Jackal while using the original and the active inference controller. It can be seen that in general the original controller outperforms the active inference controller regarding rise and settling time.

[6] http://wiki.ros.org/diff_drive_controller

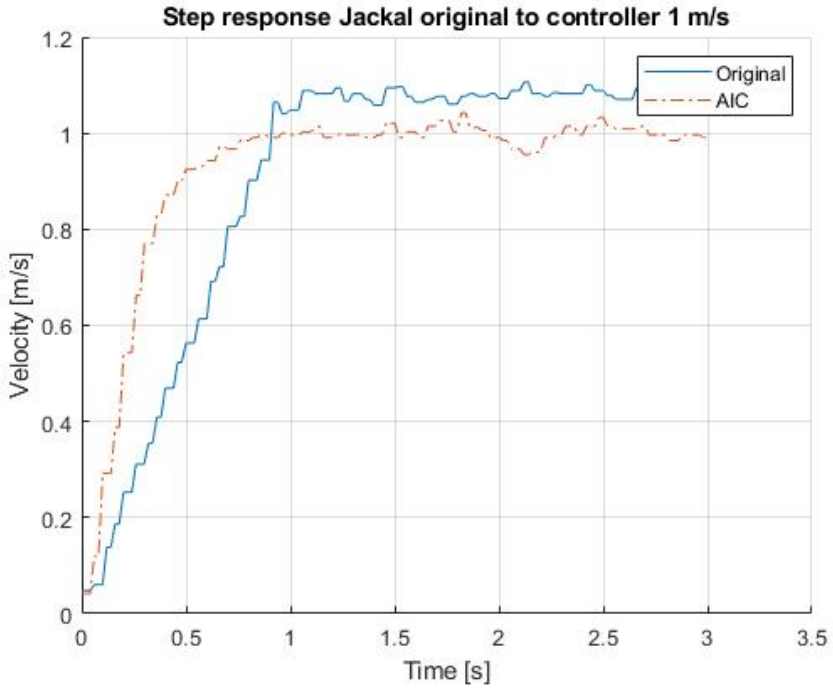


Figure 3-5: Step response to 1 m/s of the Jackal for an Active Inference controller and for the standard differential drive controller. In this case the Active Inference controller converges and rises faster.

In tables 3-1a and 3-1b the settling time (t_{set}) and rise time (t_{rise}) of the active inference controller and original controller can be found. It can be seen that for all, but the 1 m/s step responses, the original controller built in the Jackal outperforms the active inference controller. The active inference controller does converge to the desired goals, whereas the original controller settles at higher values. For this experiment the main goal was to observe the settling and rise times of the controllers, thus it can be said that for those metrics the original controller performs better. Note that later, tuning of the parameter τ lead to a shorter rise time, however compared to the original controller it was still slower.

Table 3-1: Settling time and rise time for different step responses. It can be seen that for the 0.1 m/s and 0.5 m/s, the rise and settling times of the original controller are lower. For the 1 m/s step the the settling times are similar, yet the rise time is lower for the original controller.

(a) Active inference controller.

Step size [m/s]	t_{set} [s]	t_{rise} [s]
0.1	0.67	0.30
0.5	0.87	0.49
1	1.01	0.45

(b) Jackal original controller.

Step size [m/s]	t_{set} [s]	t_rise [s]
0.1	0.14	0.04
0.5	0.54	0.35
1	1.05	0.62

Active Inference control analysis

This chapter will start with the derivation of a state space model of an Active Inference controller in section 4-1. Based on this state space controller three possible methods to decrease the settling time of the Active Inference controller will be shown in section 4-2.

4-1 State space modeling of an Active Inference controller

To get a better understanding of the Active Inference controller it was decided to build a simplified state space model. This section will cover the assumptions and simplifications of the model as well as the derivation of the state space model.

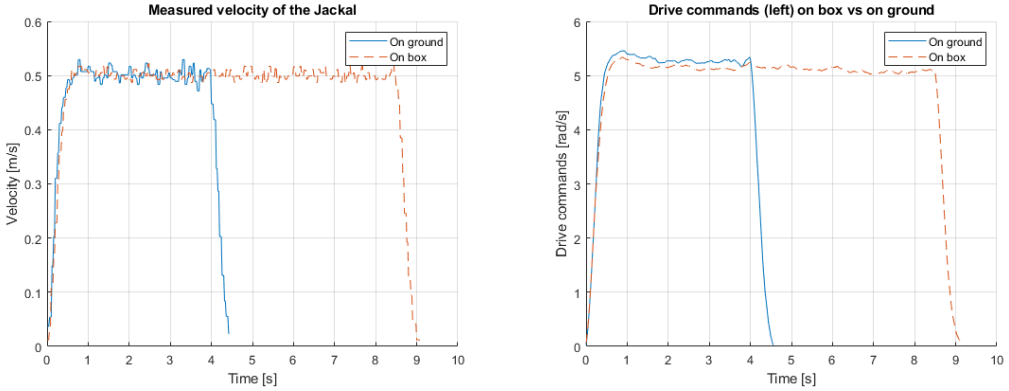
4-1-1 Assumptions for the state space Active Inference controller

A state space Active Inference controller was created to get a better understanding of the fundamentals of the Active Inference controller. The goal was to understand how the convergence time of an Active Inference controller can be decreased. It was decided to neglect the wheel ground interactions and the internal dynamics of the robot (Jackal) such as friction inside the robot. Therefore the robot is modelled as a point mass, making only the control action of interest.

Jackal on ground vs on box experiment

To verify the assumption above a step response test was performed placing the Jackal both on the ground and on a box. figure 4-1 shows the results of these tests, it can be seen that the measured velocity is similar for both cases. A small difference between both can be seen in the drive commands, figure 4-1b, it can be seen that the drive commands for the Jackal on the ground converge to a higher value. This can be due to friction between the wheels and the ground which could result in a higher value needed to converge to the same speed. The difference between both cases is quite small, and the convergence time and rise time is the same. Therefore it was decided that the assumption above can be made as the wheel ground dynamics do not affect the step response. Based on this test it was also decided that following tests can be performed by placing the Jackal on a box, as the step response is not affected. It is more complicated to verify that the internal dynamics of the robot can

be neglected, however later in this chapter the state space model will be tested against the Jackal to compare its behavior.



(a) Measured velocity of the Jackal for a step response to 0.5 m/s. The Jackal was both placed on a box and on the ground.

(b) Drive commands as computed by the Active Inference controller, both for the jackal on the box and on the ground

Figure 4-1: Results of a step response test to 0.5 m/s. The Jackal was for these tests once placed on the ground and once placed on a box. The measured velocity is similar for both cases, the drive commands are different.

4-1-2 Derivation of the state space Active Inference controller

If the dynamics of the controlled system are neglected the sensory model of the desired state ($g(\mu)$) is equal to 1. Furthermore the input of the plant is the same as the output of the plant. Due to this simplification the prediction errors simplify to equations 4-1 and 4-2, the prediction errors are written for 1 level of generalized order. As the plants input equals the plants output it can be said that $y = a$, where y are the measurements and a are the control actions.

$$\begin{aligned} \epsilon_\mu &= \dot{\mu} - \tau^{-1}(\mu_{des} - \mu) \\ \epsilon_{\dot{\mu}} &= \ddot{\mu} + \dot{\mu}\tau^{-1} \end{aligned} \tag{4-1}$$

$$\begin{aligned} \epsilon_y &= y - \mu = a - \mu \\ \epsilon_{\dot{y}} &= \dot{a} - \dot{\mu} \end{aligned} \tag{4-2}$$

Using these prediction models the belief update, equation 4-3, can be rewritten towards equation 4-4.

$$\begin{bmatrix} \dot{\mu} \\ \ddot{\mu} \end{bmatrix} = \begin{bmatrix} 0 & I \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \mu \\ \dot{\mu} \end{bmatrix} - k_\mu \left(\begin{bmatrix} \frac{I}{\tau} & 0 \\ I & \frac{I}{\tau} \end{bmatrix} \begin{bmatrix} \Sigma_\mu^{-1}\varepsilon_\mu \\ \Sigma_{\dot{\mu}}^{-1}\varepsilon_{\dot{\mu}} \end{bmatrix} - \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \Sigma_y^{-1}\varepsilon_y \\ \Sigma_{\dot{y}}^{-1}\varepsilon_{\dot{y}} \end{bmatrix} \right) \tag{4-3}$$

$$\begin{bmatrix}\dot{\mu}\\ \ddot{\mu}\end{bmatrix} = \begin{bmatrix}-k_\mu\Sigma_\mu\tau^{-2} - k_\mu\Sigma_y & 1 - k_\mu\Sigma_\mu\tau^{-1}\\ -k_\mu\Sigma_\mu\tau^{-1} & -k_\mu\Sigma_\mu - k_\mu\Sigma_{\dot{\mu}}\tau^{-2} - k_\mu\Sigma_{\dot{y}}\end{bmatrix}\begin{bmatrix}\mu\\ \dot{\mu}\end{bmatrix} + \begin{bmatrix}k_\mu\Sigma_y & 0\\ 0 & k_\mu\Sigma_{\dot{y}}\end{bmatrix}\begin{bmatrix}a\\ \dot{a}\end{bmatrix} + \begin{bmatrix}k_\mu\Sigma_\mu\tau^{-2}\\ k_\mu\Sigma_\mu\tau^{-}1\end{bmatrix}\mu_{des} \tag{4-4}$$

Similarly the action update step, defined in equation 4-5 can be rewritten to equation 4-6.

$$\begin{bmatrix}\dot{a}\\ \ddot{a}\end{bmatrix} = \begin{bmatrix}-k_a\Sigma_y(a-\mu)\\ -k_a\Sigma_y(\dot{a}-\dot{\mu})\end{bmatrix} \tag{4-5}$$

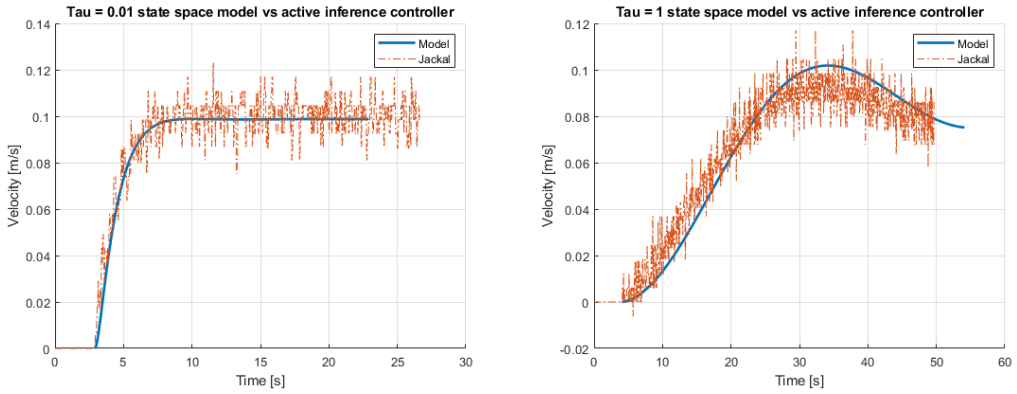
$$\begin{bmatrix}\dot{a}\\ \ddot{a}\end{bmatrix} = \begin{bmatrix}-k_a\Sigma_y & 0\\ 0 & -k_a\Sigma_{\dot{y}}\end{bmatrix}\begin{bmatrix}a\\ a'\end{bmatrix} + \begin{bmatrix}k_a\Sigma_y & 0\\ 0 & k_a\Sigma_{\dot{y}}\end{bmatrix}\begin{bmatrix}\mu\\ \dot{\mu}\end{bmatrix} \tag{4-6}$$

For the state space Active Inference controller μ and a are used as state variables, using these the state space Active Inference controller can be written (equation 4-7). Note that once again this equation assumes one level of generalized orders.

$$\begin{aligned}\begin{bmatrix}\dot{\mu}\\ \ddot{\mu}\\ \dot{a}\\ \ddot{a}\end{bmatrix} &= A * \begin{bmatrix}\mu\\ \dot{\mu}\\ a\\ \dot{a}\end{bmatrix} + B * \mu_{desired}\\ y &= C\begin{bmatrix}\mu\\ \dot{\mu}\\ a\\ \dot{a}\end{bmatrix}\end{aligned} \tag{4-7}$$

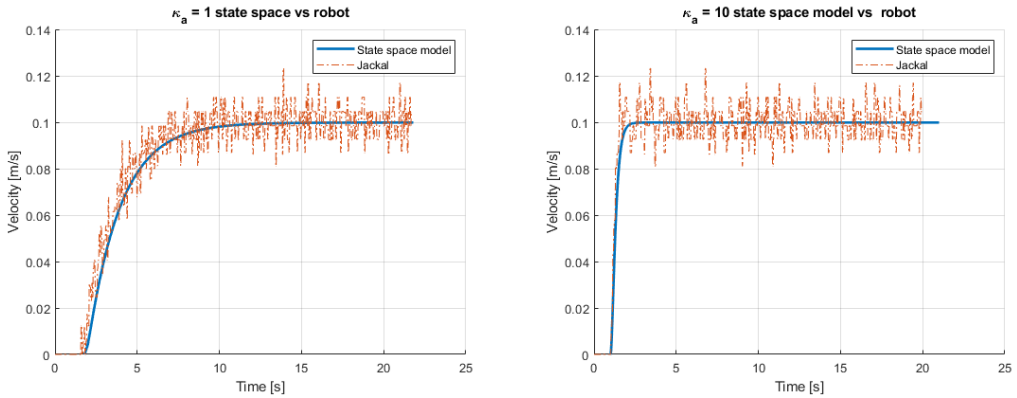
4-1-3 Results of the state space Active Inference controller

Previously a simplified state space Active Inference controller was described, a MATLAB implementation can be found in appendix A. In this section the state space model is compared to the actual robot to compare the performance. To compare the performance multiple step responses were performed, during the test some of the parameter were changed. The goal of the experiments is to see whether the settling time and rise time are comparable. In figure 4-2 the parameter τ is changed and the resulting responses for the state space model and the actual response on the Jackal are plotted. It can be seen that changing τ results in a similar change in response for the model and the controller. In figure 4-3 κ_a was changed, once again the results are similar.



(a) $\tau = 0.01$ **(b)** $\tau = 1$

Figure 4-2: Results of the tests where the Active Inference controller is compared for the response on the robot and as modelled by the state space model. During the tests the parameter τ was changed to see whether the model behaved in similar fashion as the actual response.



(a) $\kappa_a = 1$ **(b)** $\kappa_a = 10$

Figure 4-3: Results of the tests where the Active Inference controller is compared for the response on the robot and as modelled by the state space model. During the tests the parameter κ_a was changed to see whether the model behaved in similar fashion as the actual response. It can be seen that changing the parameter leads to similar results in the model and on the robot.

4-1-4 Limitations of the state space model

The previous results imply that the state space model can be used to see how the Active Inference controller would perform on the actual robot. However, it can be seen from the graphs that the actual response on the robot is prone to noise which is not modeled by the state space model. This could lead to inaccuracies for higher control gains, which is illustrated in figure 4-4. In this case the response on the robot became unstable while the state space model remains stable. The point of instability is different for the model and the actual response, this makes it more complicated to use the model.

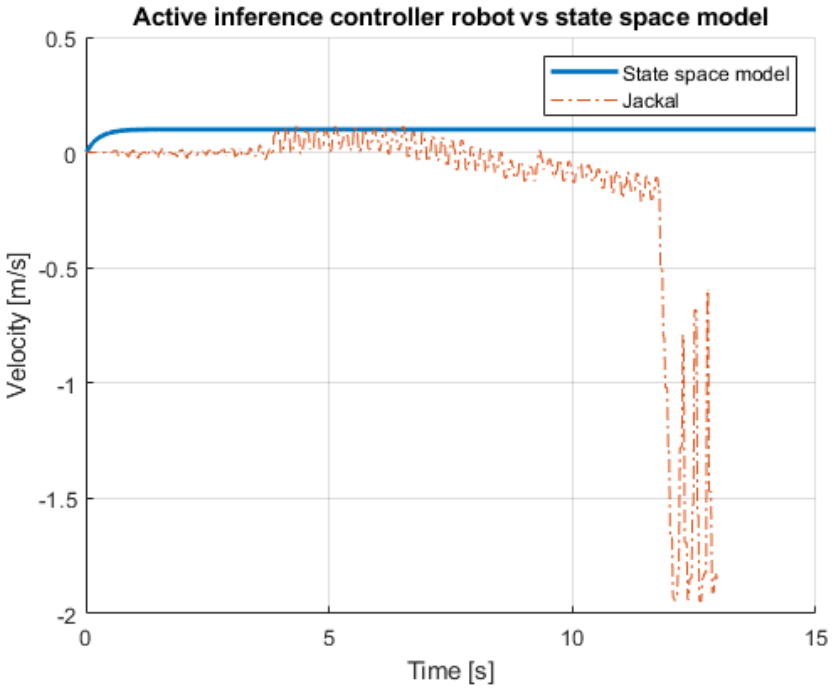


Figure 4-4: Step response of the state space model vs the actual response on the robot. In this case the robot became unstable, while the response of the state space model was stable.

4-1-5 State space Active Inference step response

In figure 4-5 a step response to 1 m/s as calculated by the state space model is shown. The update frequency for this system has been set relatively low at 10Hz. There are two points of interest in this graph, firstly it can be seen that the actions starts after a time step, dt (1/f), after the beliefs. This is due to the control layout as for the first step the action will always remain zero, unless a constant would be introduced. Secondly, the action trails the beliefs, this is counter-intuitive as this means that a wrong state estimation leads to an action in the right direction. In essence, the believed velocity first converges which leads to a correct response from the action. To speed up the Active Inference controller it could be beneficial to decrease the convergence time of the beliefs and thereby of the actions as well. This idea will be explored in the following chapters.

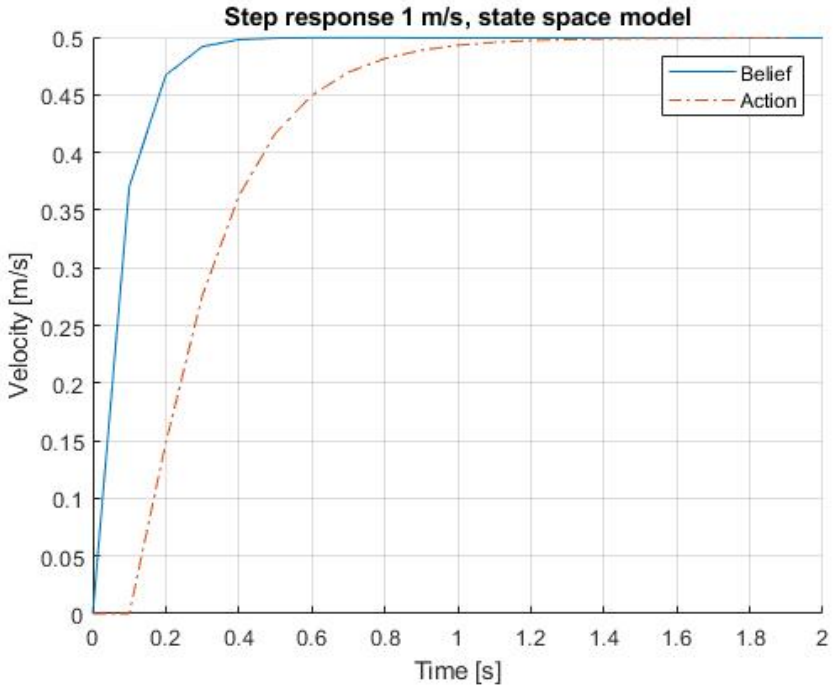


Figure 4-5: Step response of the state space model for an Active Inference controller with an update frequency of 10Hz. The believed velocity and action, in this case the velocity output, are shown. It can be seen that the actions trail the beliefs, and that the action start a time step later then the beliefs.

4-2 Methods to decrease the convergence time of an Active Inference controller

The previous sections introduced a state space Active Inference controller. Based on this controller different methods to decrease the convergence time of an Active Inference controller were found. This section will introduce the methods and elaborate on the expected behavior.

4-2-1 Increasing the update frequency of the entire controller

The first method uses the update frequency of the entire controller to decrease the convergence time. In the previous sections it was found that the actions follow the beliefs. This implies that a faster belief convergence results in a faster control action convergence. Increasing the update frequency of the entire system, means that the beliefs and actions are calculated at a higher frequency.

Expected behavior

Increasing the update frequency of the entire system, increases the frequency of the belief update step and the action update step. By increasing the update frequency the actions and

beliefs will be calculated at a higher frequency. Therefore it is expected that both converge faster, which should help the system converge faster.

4-2-2 Increasing the belief update frequency

In the previous sections it was found that the action trails the beliefs, which implies that a faster believe convergence leads to a faster action convergence. The second method to accomplish this is by increasing the update frequency of the belief update step. In the previous section the update frequency of the entire controller was increased, however it seems that increasing the update frequency of the beliefs should be enough.

Expected behavior

The Active Inference controller consists of two loops, the belief update loop which estimates the believes and the action update loop that calculates the action updates. If the belief update frequency is increased the belief update (equation 2-9), the beliefs (μ) and the prediction errors (ε_y and ε_μ) will be calculated at a higher frequency. The free energy is minimized for each update step, i.e. the prediction errors are minimized, implying that the believed measurements converge towards the actual measurement. Due to a more accurate belief update the action update becomes more accurate, which should result in a faster convergence. All in all, it is expected that increasing the belief update frequency leads to a faster converging of the beliefs. During the state space analysis it was found that the action follow the beliefs and thus the actions should converge faster.

4-2-3 Adapting the Active Inference controller to change the belief update frequency

To decrease the convergence time it was opted to increase the update frequency of the belief update loop. In figure 4-6 the belief update loop is highlighted. The frequency of this loop will be increased and subsequently the frequency of the believed measurements ($\hat{y}$) is increased. The frequency of the generalized measurements ($\tilde{y}$) will not be changed and in the free energy formula the old measurement will be reused. To conclude, the update step of the believes and actions will be calculated at their respective frequencies. This changes lead to a novel Active Inference controller which code can be found in [1].

4-2-4 Parameter tuning for the increased belief update frequency

The last method that will be explored combines the approach of the previous section with parameter tuning. For an Active Inference controller different tuning parameters are available, namely τ, κ_a, κ_y and the values in the precision matrices. In this thesis only the parameters τ, κ_a and κ_y will be tuned to show the working principle of the method.

[1] https://gitlab.tudelft.nl/active-inference-drive-control/active-inference-drive-control/-/tree/double_loop

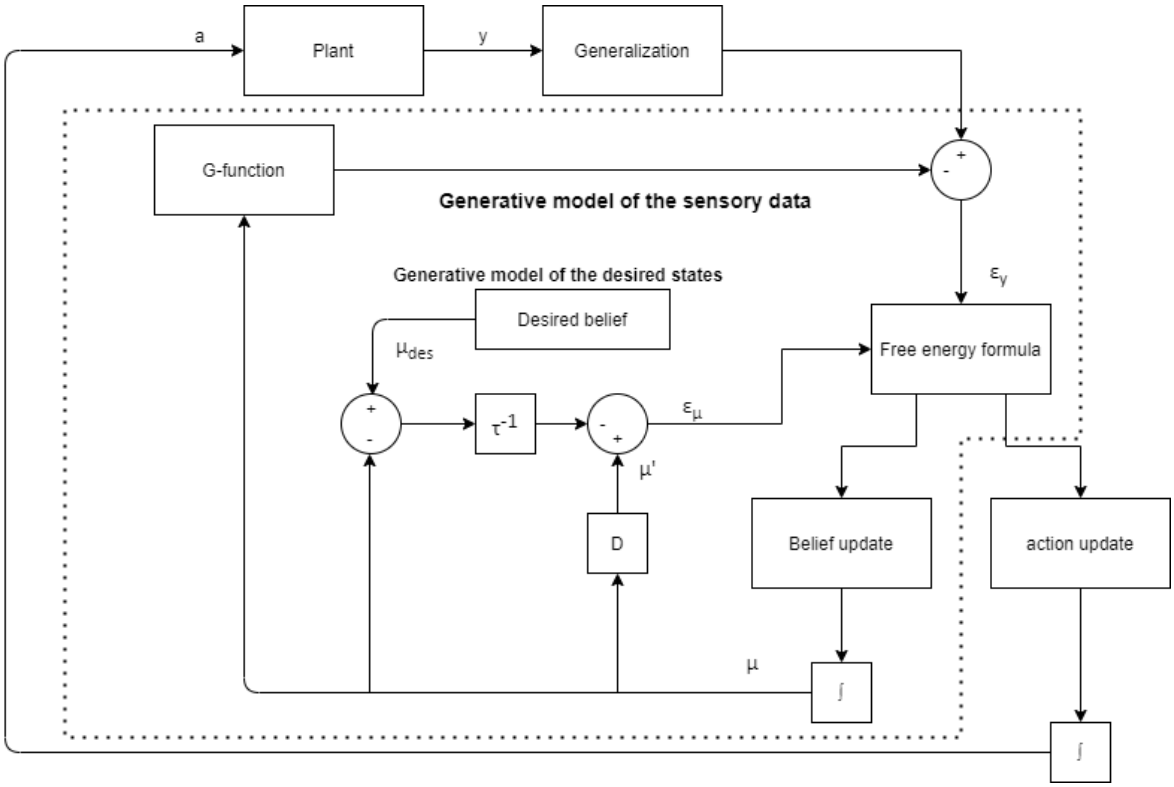


Figure 4-6: Overview of the Active Inference controller. The belief update loop is highlighted, the frequency of this loop will be increased to try and reduce the convergence time.

Expected behavior

If the update frequency of the beliefs is increased, a more accurate belief estimation is expected. The forward Euler method is used to integrate the belief ($\dot{\mu}$) and action ($\dot{a}$) updates as was shown in equation 2-11. In this method the time step h is used, if the update frequency of the system is changed this time step changes as well. Increasing the update frequency means that the time step h decreases, the impact of $\dot{\mu}$ and $\dot{a}$ on μ and a will therefore decrease. This would mean that a larger range of tuning parameters can be used, since the update steps can be larger without causing instability.

Results

This thesis aims to decrease the convergence time of an Active Inference controller. The first tests that were performed tested the robot versus the simulated robot in Gazebo, these results will be shown in section 5-1. Previously three methods that potentially decrease the convergence time of the controller where discussed. In this chapter the results will be shown and discussed. Starting with the results of the changing update frequency for the entire controller in section 5-2. Followed by the results of a changing belief update frequency in section 5-3 and concluded with the parameter tuning in section 5-4. The chapter will be concluded in section 5-5-3 with a comparison of the improved Active Inference controller versus the differential drive controller.

Definitions and control setup

Throughout this section the settling time and rise time will be used, for the settling time a 5% interval was used meaning that the system needed to converge within 5 % of its final value. For the rise time the 10% to 90% rise time is used, which is the time to rise from 10% of the set point to 90%. For most of the experiments the measured velocity, believed velocity and drive commands will be reported. The drive commands are the control actions in this case. The parameters used in the experiments are always as reported in appendix C unless stated otherwise.

5-1 Robot versus Gazebo

The first experiments conducted where both on the robot and in the simulation environment Gazebo. It was decided to start with 2 step response tests, one of an unstable system and one of a stable system. The goal of these experiments was to find out whether the responses would be similar for the robot and the simulated robot in Gazebo. If the results are similar the Gazebo environment can be used, this would make testing easier.

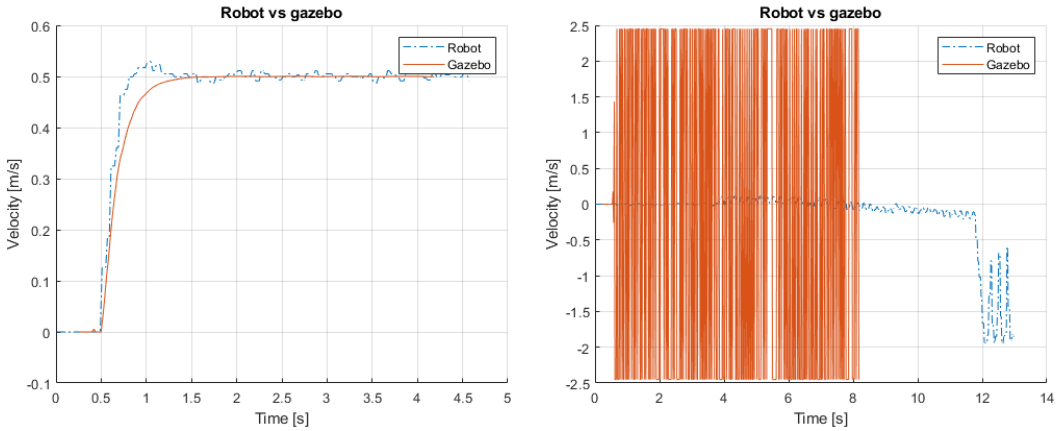
figure 5-1 shows the results of a step response to 0.5 m/s for an Active Inference controller (τ = 0.001, $f_{update} = 1kHz$), and for an unstable controller (τ=0.001, f_{update} = 50Hz). figure 5-1a displays the Active Inference controller and it can be seen that its settling time is similar for both the simulation and the actual controller. However, in the simulated case there is no overshoot which is present in the actual robot. It can also be seen that the actual robot is

subjected to noise whereas the simulated robot is not. The noise poses an extra challenge for the controller, as the measurements are affected by it. This could cause the error values ε_y and ε_μ) to be higher, which in its turn can affect the belief and action updates.

figure 5-2 shows the drive commands for this step response, it can be seen that the drive commands settle on a higher value for the actual robot. The drive commands in the Gazebo environment are smoother than the actual robot which can be explained by the noise in the system. There are two possibilities that explain the difference in steady state value for the drive commands, firstly this could be due to the physics engine of Gazebo and secondly this could be due to the model of the Jackal. The physics engine models the friction between the ground and the wheels, if this is different than the actual case this results in a difference in value. The second, models the dynamics in the robot if these are different this can result in a different value.

The second figure, 5-1b, shows the measured velocity for the unstable controller. The step response is both for the simulation and the robot unstable. There is a difference in response between the two, the Gazebo simulation became immediately unstable and switched between the maximum velocities of the robot (-2.5 m/s and 2.5 m/s). The actual robot on the other hand started ringing around its initial position and later around -1.5 m/s.

The goal of these experiments was to see whether Gazebo can be used instead of the actual robot. It can be said that there is a difference between the two, however the settling time is similar. Therefore Gazebo can be used to gain information about the effect of changing parameters on the settling time of the system. The overshoot on the actual robot could lead to instability or a delay in settling time which is why usage of the actual robot is preferred.



(a) Stable 1kHz **(b)** Unstable 50Hz.

Figure 5-1: This figure shows two graphs, 5-1a in which the results of a step response to 0.5 m/s for an Active Inference controller is compared to that of the robot. In this case the update frequency was set at 1kHz and the parameters τ was set to 0.001 the other parameters were the same as the original controller. It can be seen that the settling time is similar, although the rise towards this point is quite different. In the second figure the same controller is tested at a frequency of 50Hz. It can be seen that similarly to the robot, the simulated robot in gazebo became unstable for this case.

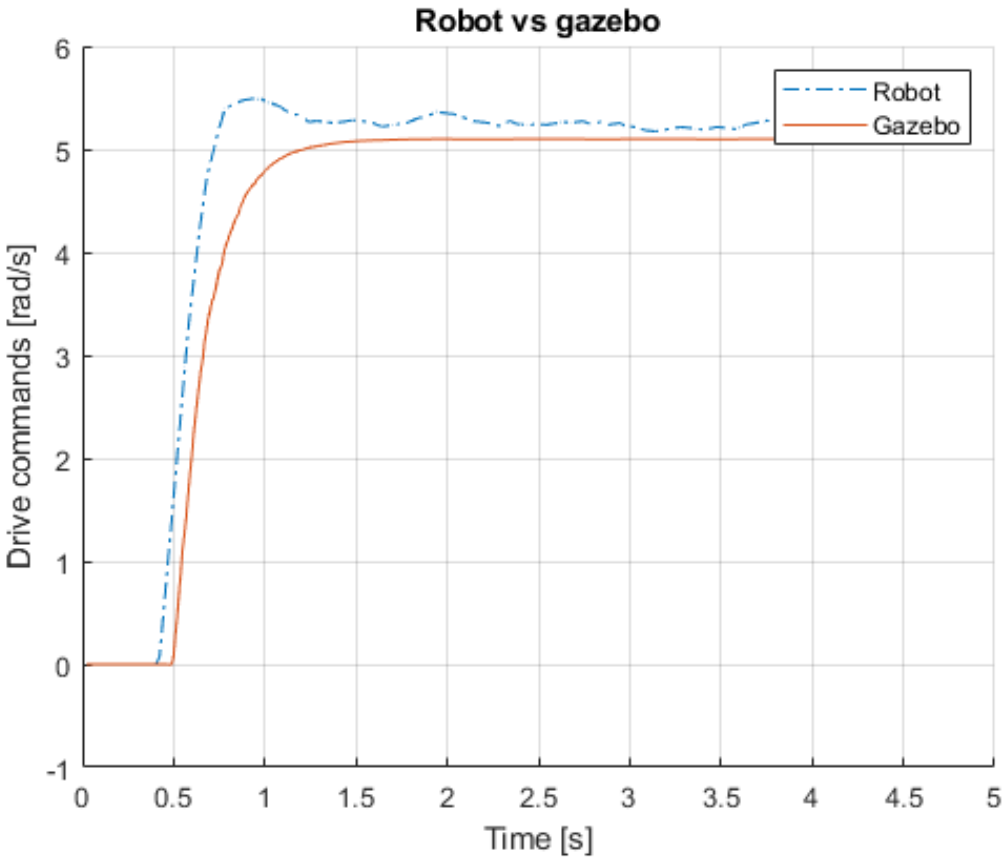


Figure 5-2: Robot vs Gazebo, drive commands. It can be seen that the drive commands send in real life are less smooth then in the simulation. The drive commands settle at a higher values in the actual robot then in Gazebo, which implies that the physics engine models friction different then it actually is.

5-2 Changing update frequency for the controller

In chapter 4 three methods that aimed to decrease the settling time of the Active Inference controller were introduced. This section will show the results of the first method: increasing the update frequency of the entire control system. In chapter 3 it was mentioned that the velocity is measured at a frequency of 50Hz, it was therefore decided to use this as the lowest value for the update frequency. To observe whether changing the frequency helps, it was decided to increase the frequency to 100Hz and 1kHz. The former was chosen as it is twice the measurement frequency, the latter as it is reasonably higher than the measurement frequency. It is decided to perform three different step response, the first to 0.3 m/s the second to 0.5 m/s and the last to 1 m/s.

figure 5-3, 5-4 and 5-5 show both the measured velocities and drive commands for the step response to 0.3 m/s, 0.5 m/s and 1 m/s respectively. Additionally in table 5-1 the measured settling and rise times for the different step responses are reported. It can be seen that changing the update frequency did not result in shorter rise or settling times. The drive commands remain similar for all frequencies, although the 50Hz case does seem to have a bit

more overshoot compared to the others. The believed velocities are reported in figure 5-6, and these are similar for all cases as well.

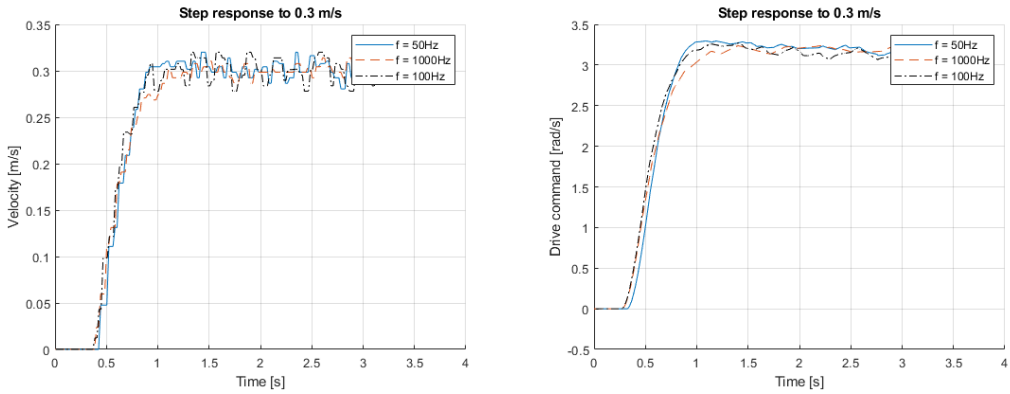
The differences in rise and settling times found in the table seem negligible as there is quite some noise in the system. A closer look at the update scheme of the controller explains the results, the integration scheme to find the actual control action a and belief μ. To find these values a Forward Euler method was used which results in equation 5-1 and 5-2. By changing the update frequency, the time step (h) is changed and the controller essentially takes smaller steps. Due to this changing the update frequency does not affect the rise or settling time.

$$a_t = a_{t-1} + h\dot{a}_t \tag{5-1}$$

$$\mu_t = \mu_{t-1} + h\dot{\mu}_t \tag{5-2}$$

Table 5-1: Table that shows the settling and rise times for the different update frequencies of the system. Three different step responses were performed, and the settling times were measured from both the measured velocity and drive commands. It can be seen that in general the resulting rise and settling times are quite similar.

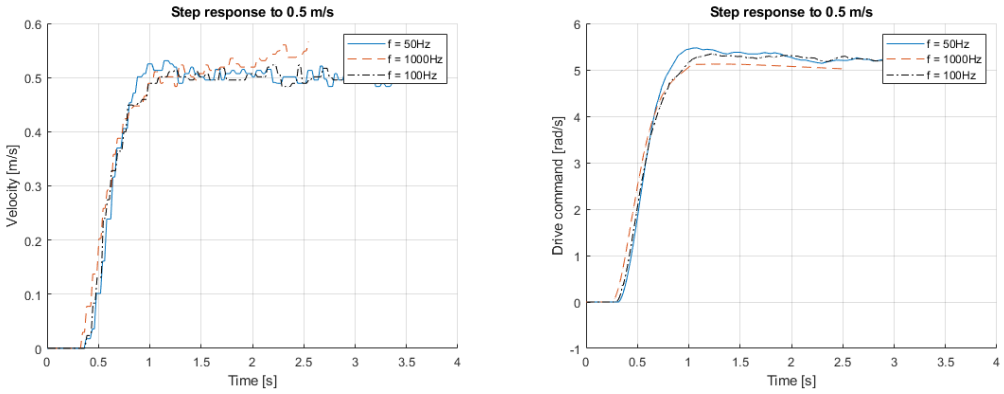
	f = 50Hz		f = 100Hz		f = 1kHz	
Step size [m/s]	t_{set} [s]	t_{rise} [s]	t_{set} [s]	t_{rise} [s]	t_{set} [s]	t_{rise} [s]
0.3	0.84	0.37	0.77	0.40	0.81	0.40
0.5	0.76	0.32	0.8	0.55	0.84	0.39
1	0.83	0.40	0.77	0.46	0.81	0.44



(a) Measured wheel velocity

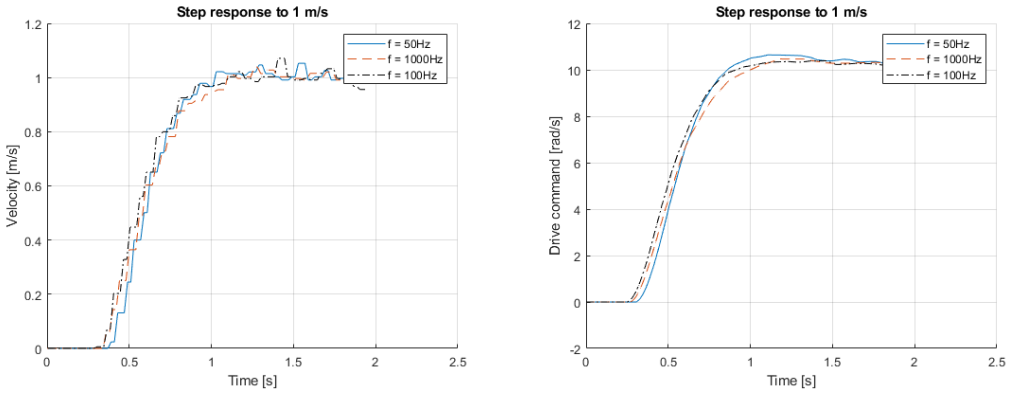
(b) Drive commands.

Figure 5-3: Results for a changing update frequency of the entire system for a step response of 0.3 m/s. Changing the update frequency results in similar step responses. A small difference can be observed as the 50Hz case starts with a bit of a delay.



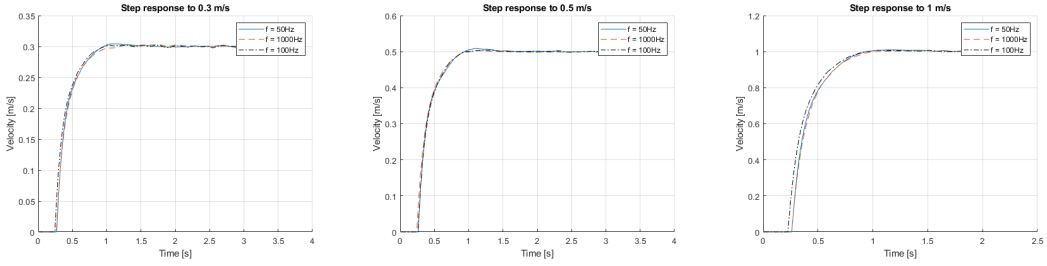
(a) Measured wheel velocity **(b)** Drive commands.

Figure 5-4: Results for a changing update frequency of the entire system for a step response of 0.5 m/s. Once again the step responses are similar for a changing update frequency. The 1kHz case increases a bit in the later stage of the step response, this appears to be measurement error as the drive commands keep constant.



(a) Measured wheel velocity **(b)** Drive commands.

Figure 5-5: Results for a changing update frequency of the entire system for a step response of 1 m/s. The step responses are similar, the drive commands for the 50Hz case show that a these were a bit larger.



(a) Beliefs step 0.3 m/s **(b)** Beliefs step 0.5 m/s. **(c)** Beliefs step 1m/s.

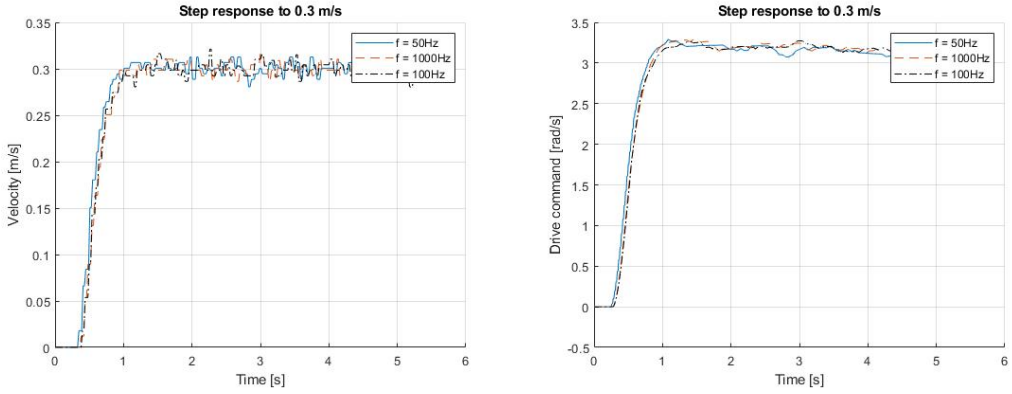
Figure 5-6: Believed velocities for the different step responses. It can be seen that these remain similar for all the tests.

5-3 Increased update frequency for the belief update step

The second method that will be explored to decrease the settling time is changing the update frequency of the belief update step. For these experiments the same update frequencies (50Hz, 100Hz and 1kHz) as in the previous section were used, however this time the action update was kept constant at 50Hz. figures 5-7 till **??** show the results of the step responses tests to 0.3 m/s, 0.5m/s and 1m/s respectively. In table 5-2 the resulting settling and rise times are shown, it can be seen that these remain similar for a changing update frequency. In figure 5-10 the believed velocities are shown, these are similar for all frequencies. This can be explained in a similar fashion as the previous section, however this time the focus is on equation 5-2 as only that time step changes. The principle is the same, as due to the integration scheme the system will not converge faster.

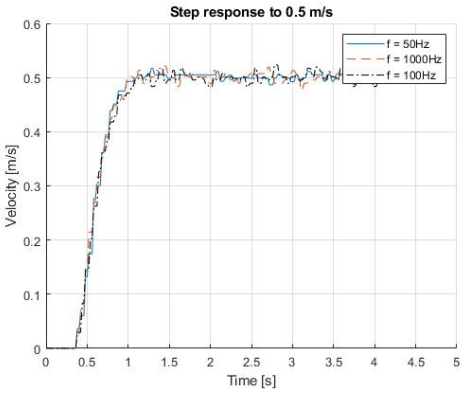
Table 5-2: The settling and rise times for different step responses are shown, for each step response the update frequency of the belief update loop was changed between 50Hz, 100Hz and 1kHz. It can be seen that increasing the update frequency does not necessarily decrease the settling or rise times, they remain similar.

	f = 50Hz		f = 100Hz		f = 1kHz	
Step size [m/s]	t_{set} [s]	t_{rise} [s]	t_{set} [s]	t_{rise} [s]	t_{set} [s]	t_{rise} [s]
0.3	0.77	0.40	0.73	0.43	0.73	0.40
0.5	0.75	0.40	0.77	0.45	0.79	0.43
1	0.79	0.39	0.76	0.39	0.80	0.46

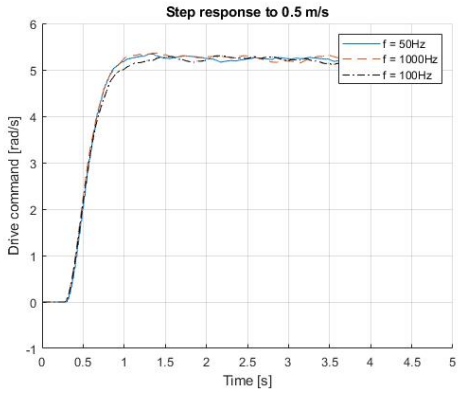


(a) Measured wheel velocity **(b)** Drive commands.

Figure 5-7: The update frequency of the belief update loop was changed, a step response to 0.3 m/s was performed. It can be seen that the step responses are similar, increasing the update frequency does not help.

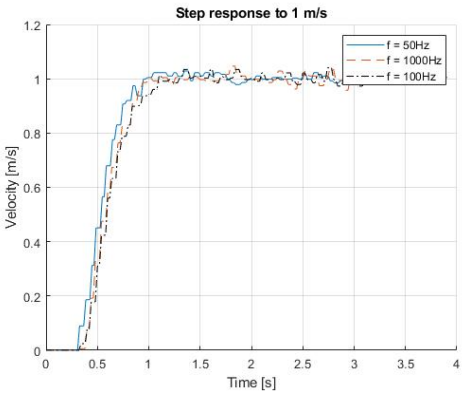


(a) Measured wheel velocity

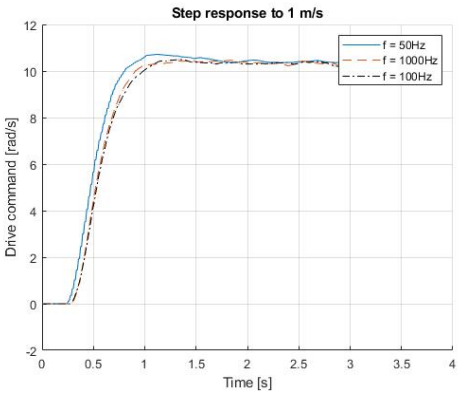


(b) Drive commands.

Figure 5-8: The update frequency of the belief update loop was changed, a step response to 0.5 m/s was performed. It can be seen that a changing update frequency does not lead to a difference in the step response.

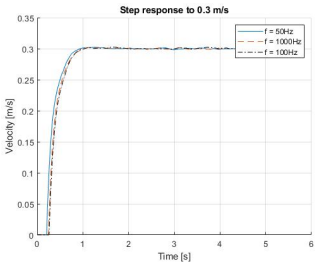


(a) Measured wheel velocity

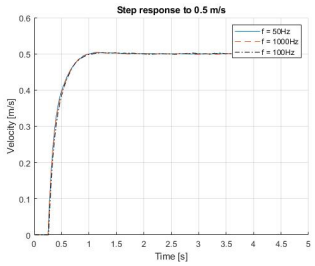


(b) Drive commands.

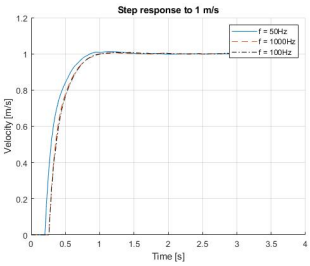
Figure 5-9: The update frequency of the belief update loop was changed, a step response to 1 m/s was performed. In similar fashion as the previous step responses it can be seen that changing the belief update frequency does not help the system converge faster.



(a) Beliefs step 0.3 m/s



(b) Beliefs step 0.5 m/s.



(c) Beliefs step 1m/s.

Figure 5-10: Believed velocities for the different step responses. It can be seen that these remain similar for all the tests.

5-4 Stability regions for different ranges of tuning parameters

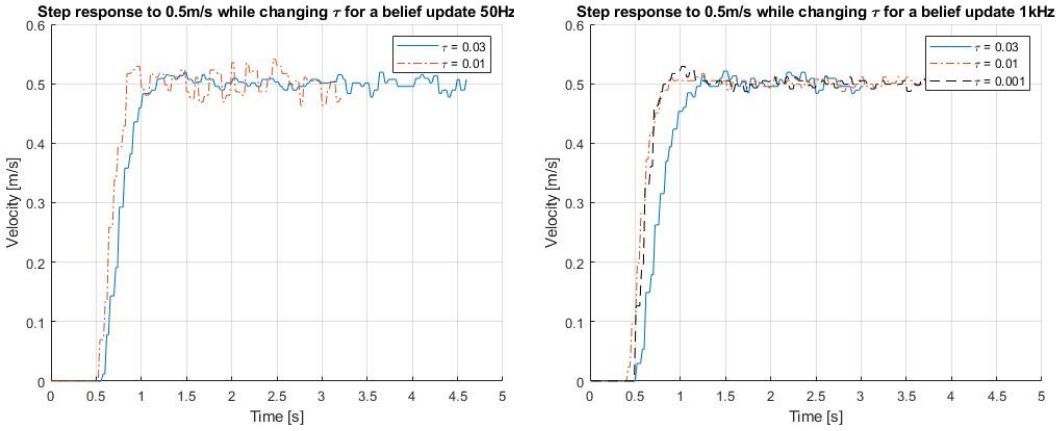
The last explored method is a combination of the previous method, increasing the update frequency of the belief update step and tuning of the parameters. In the previous experiment it was observed that changing the update frequency made it possible to use a wider spectrum of values for the tuning parameters. In this thesis the effect of three tuning parameters τ, κ_μ and κ_a, will be explored. The parameters were changed into different values, these can be found in tableC-1.

Table 5-3: Ranges of the parameters that were tested.

Parameter	Values
τ	[0.03, 0.01, 0.001]
κ_μ	[0.01, 0.05, 0.1, 1]
κ_a	[8, 10, 15, 20]

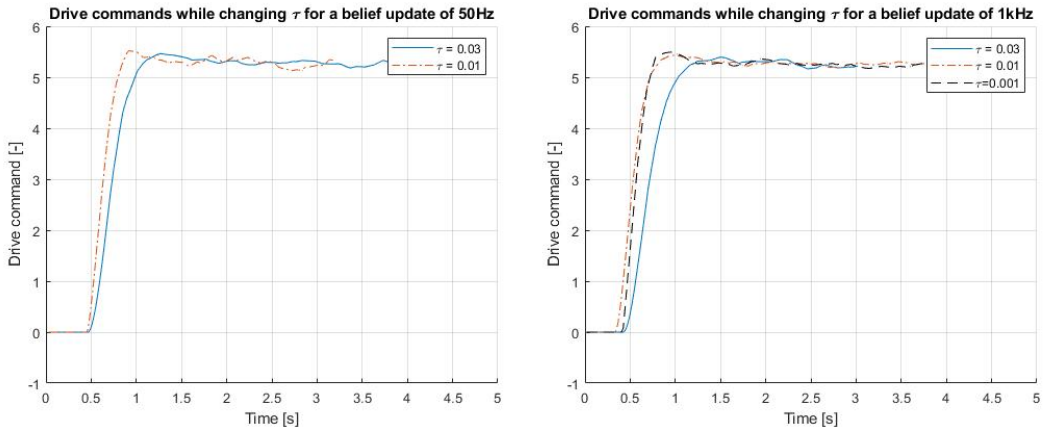
5-4-1 Tuning the temporal parameter

In figure 5-11 the measured velocity and in figure 5-12 the drive commands for changing τ was shown for both a belief update frequency of 50 Hz and 1kHz. The resulting graph for $\tau = 0.001$ using a belief update frequency of 50Hz is not reported in this graph as the system became unstable, it can be found in appendix D. It can be observed that the rise and convergence time decrease with a decrease of the parameter τ. This difference can be seen in the measured velocity, but more clearly in the drive commands. This is emphasized in table 5-4 where the convergence and rise times can be found.



(a) Belief update frequency of 50Hz. **(b)** Belief update frequency of 1kHz.

Figure 5-11: Measured velocities for a changing τ. The 1kHz case remains stable for lower values of τ compared to the 50Hz case. In the case of τ, decreasing the parameter increases the control performance.



(a) Belief update frequency of 50Hz. **(b)** Belief update frequency of 1kHz.

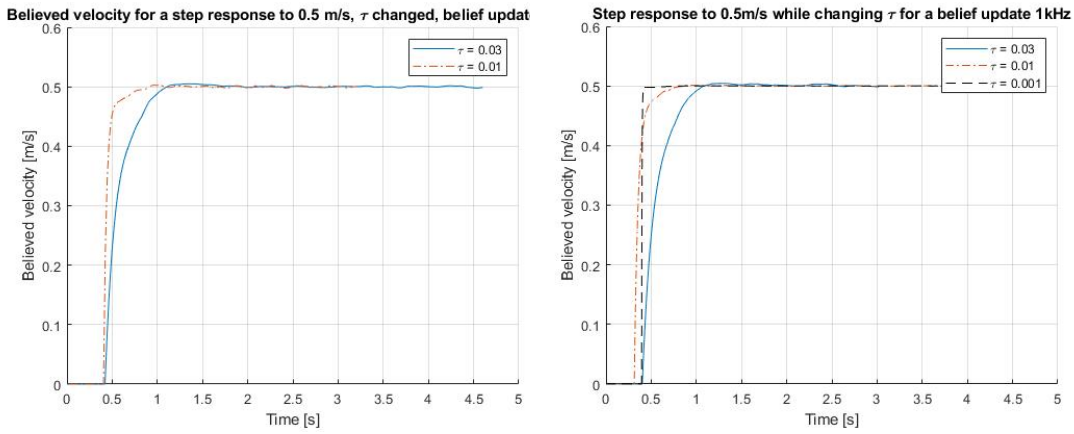
Figure 5-12: Drive commands for a changing τ. It can be seen that decreasing τ makes the drive commands converge faster, yet at the cost of a bit overshoot.

Table 5-4: Table that shows the settling time and rise time while changing τ. It can be seen that both the rise and settling times decrease with a decrease of τ.

	f = 50Hz		f = 1kHz	
τ	t_{set} [s]	t_{rise} [s]	t_{set} [s]	t_{rise} [s]
0.03	0.88	0.36	0.81	0.39
0.01	0.75	0.30	0.75	0.26
0.001	-	-	0.67	0.19

The believed velocity for this experiment can be found in figure 5-13. Decreasing the parameter τ leads to faster convergence of the believed velocity. This behavior can be explained by looking at equation 5-3, the definition of the belief update, which was derived in chapter 3. Decreasing τ leads to an increase of the second term both due to the $1/\tau$ in the matrix as the τ in the error term ε_μ. Increasing τ introduces a larger bias to ε_μ, which results in a faster convergence towards the desired state. Essentially, the beliefs will converge faster due to a lower value of τ and subsequently the entire system will converge faster.

$$\begin{bmatrix} \dot{\mu} \\ \dot{\mu}' \\ \dot{\mu}'' \end{bmatrix} = \begin{bmatrix} 0 & I & 0 \\ 0 & 0 & I \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mu \\ \mu' \\ \mu'' \end{bmatrix} - k_\mu \left(\begin{bmatrix} \frac{I}{\tau} & 0 & 0 \\ I & \frac{I}{\tau} & 0 \\ 0 & I & \frac{I}{\tau} \end{bmatrix} \begin{bmatrix} \Pi_\mu \varepsilon_\mu \\ \Pi_{\mu'} \varepsilon_{\mu'} \\ \Pi_{\mu''} \varepsilon_{\mu''} \end{bmatrix} - \begin{bmatrix} R^T & 0 & 0 \\ 0 & R^T & 0 \\ 0 & 0 & R^T \end{bmatrix} \begin{bmatrix} \Pi_y \varepsilon_y \\ \Pi_{y'} \varepsilon_{y'} \\ \Pi_{y''} \varepsilon_{y''} \end{bmatrix} \right) \quad (5\text{-}3)$$

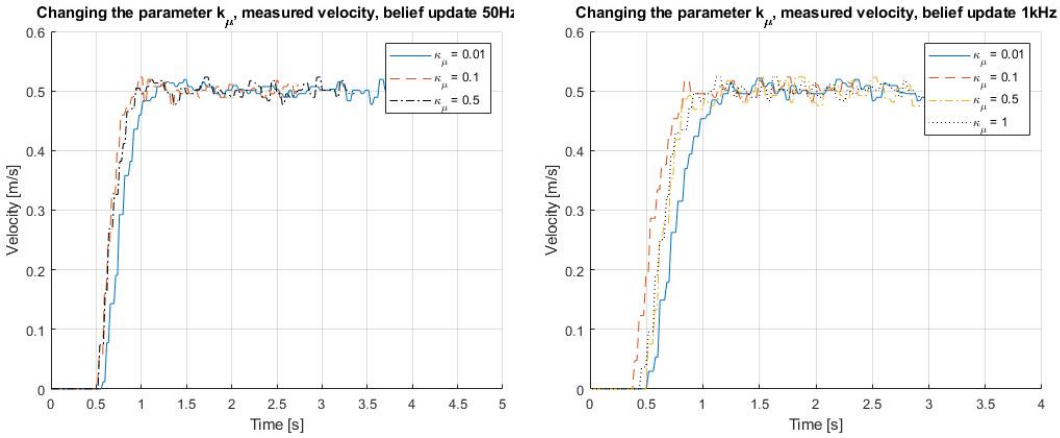


(a) Belief update frequency of 50Hz. **(b)** Belief update frequency of 1kHz.

Figure 5-13: Believed velocity for a changing τ parameter. It can be seen that decreasing τ leads to a faster convergence, this is especially observable for $\tau = 0.001$.

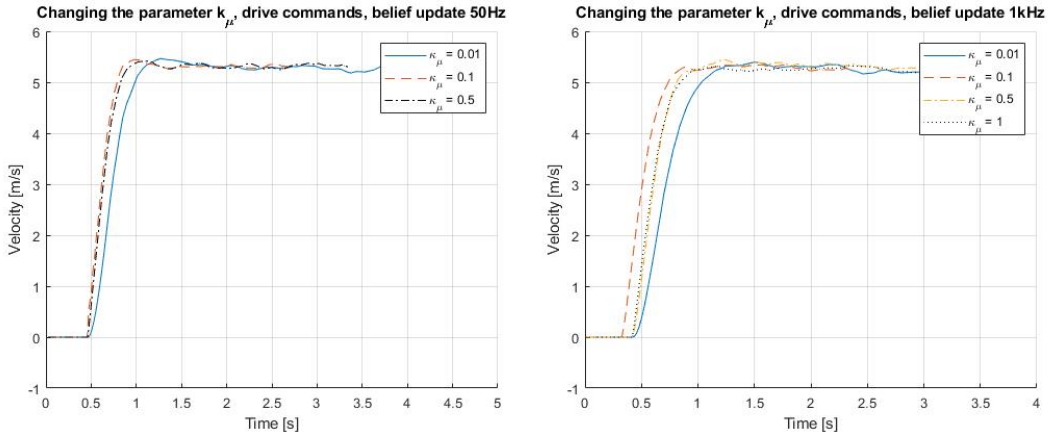
5-4-2 Changing the learning rate for the belief update

In figure 5-14 the measured velocity for different values of the learning rate for the belief update, κ_μ is shown. It can be seen that increasing the parameter leads to a decrease in rise and settling time. This is even more clearly visible in the drive commands in figure 5-15 which settle faster for higher values of the learning rate. Table 5-5 reports the measured rise and settling times, it can be seen that an increase of the parameter indeed results in an decrease of rise and settling time. Note that the learning rate for a belief update frequency of 1kHz can be changed to 1, however for the 50Hz cases this results in a unstable system (Graph can be found in appendix D).



(a) Belief update frequency of 50Hz. **(b)** Belief update frequency of 1kHz.

Figure 5-14: Measured velocities for a changing κ_μ. Increasing κ_μ decreases the settling time.



(a) Belief update frequency of 50Hz. **(b)** Belief update frequency of 1kHz.

Figure 5-15: Drive commands, increasing k_μ makes the drive command converge faster.

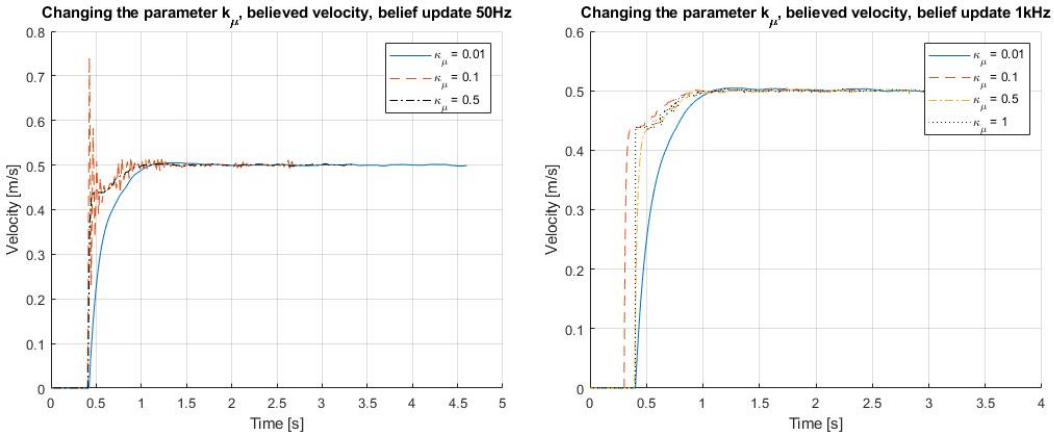
Table 5-5: Table that shows the settling time and rise time while changing κ_μ. It can be seen that the rise time and settling times decrease with an increase of κ_μ.

	f = 50Hz		f = 1kHz	
κ_μ	t_{set} [s]	t_{rise} [s]	t_{set} [s]	t_{rise} [s]
1	-	-	0.58	0.35
0.5	0.64	0.30	0.61	0.30
0.1	0.66	0.24	0.69	0.30
0.01	0.94	0.35	0.93	0.39

A closer look to figure 5-16 (the believed velocity) shows that the beliefs are already overshooting for $\kappa_\mu = 0.1$ and 0.5 for the systems with a belief update rate of 50Hz. This is a first indicator that the system becomes unstable, as the control action tends to follow the beliefs. To understand this behavior equation 5-3 can once again be used. Increasing κ_μ leads to a larger effect of the second term, which causes the system to take larger steps for the belief update. A difference compared to the parameter τ is that the term itself is multiplied by the parameter κ_μ.

5-4-3 Changing the learning rate of the action update

figures 5-17 and 5-18 show the measured velocity and the drive commands of the Active Inference controller for varying values of κ_a. In figure 5-17 it can be seen that increasing the learning rate leads to a shorter rise time, at the cost of overshoot and ringing. Similar behavior can be observed in the drive commands, where ringing and overshoot is observable for the higher values of κ_a. It can also be seen that the effect is observable for both the 50 Hz and 1kHz case, however it occurs earlier at the 50Hz case. The overshoot is also higher for the 50Hz case if one compares for example $\kappa_a = 20$. The ringing results in a longer settling time which can be observed in table 5-6, the rise time does decrease. To understands what is happening in this case, the action update (equation 5-4) can be used. Increasing κ_a directly affects the control action update and increases the control action. If the steps are too large,

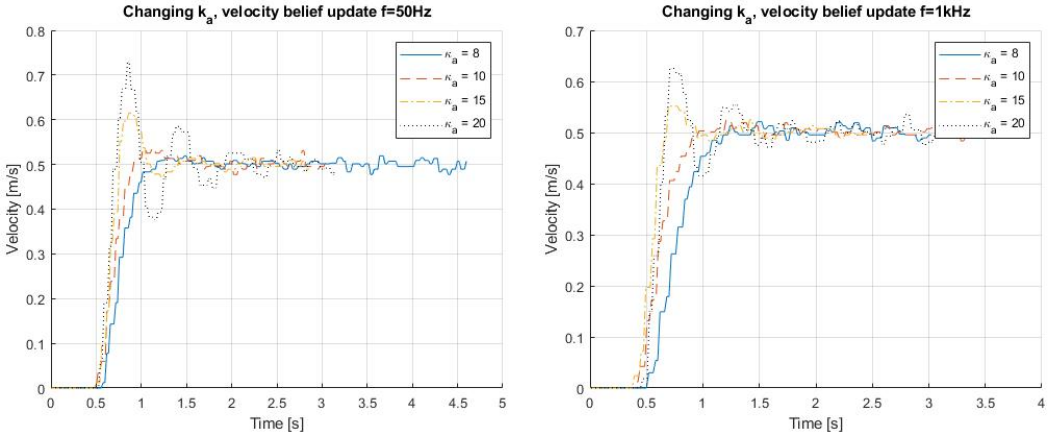


(a) Belief update frequency of 50Hz. **(b)** Belief update frequency of 1kHz.

Figure 5-16: Believed velocity for changing κ_μ. It can be seen that for the 50Hz case the beliefs become less accurate. For the 1kHz case the beliefs remain fine.

ringing will start to occur as the system overcompensates.

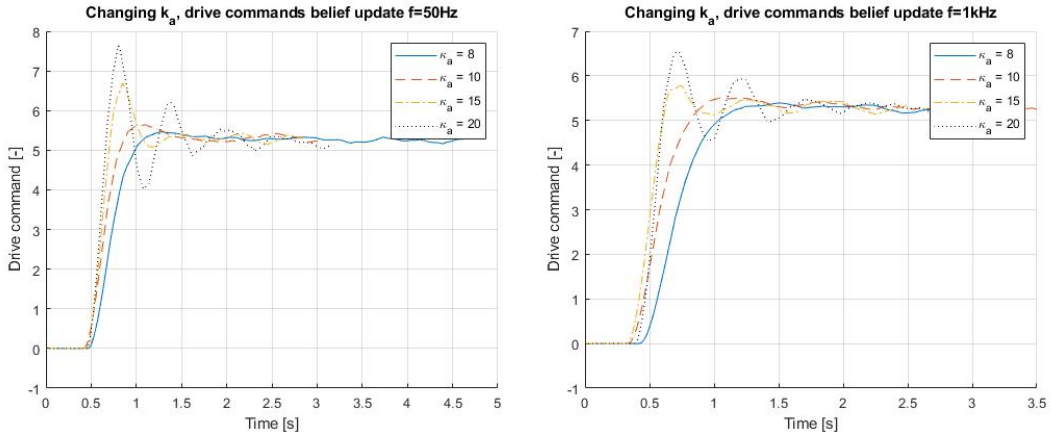
$$\dot{a} = -\kappa_a \begin{bmatrix} K & 0 & 0 \\ 0 & K & 0 \\ 0 & 0 & K \end{bmatrix} \begin{bmatrix} \Pi_y \varepsilon_y \\ \Pi_{y'} \varepsilon_{y'} \\ \Pi_{y''} \varepsilon_{y''} \end{bmatrix} \tag{5-4}$$



(a) Belief update frequency of 50Hz. **(b)** Belief update frequency of 1kHz.

Figure 5-17: Measured velocities for a changing κ_a. Increasing κ_a leads to ringing and overshoot, for a small increase to 10 it seems beneficial.

figure 5-19 shows the believed velocity once again ringing is observed. It can be seen that the ringing is less present in the believed velocity. This can be explained by setup of the controller, as the system tends to believe that it reaches the desired state. This means that the system expects to be closer towards the desired set point and thus basically estimates its actual position wrongly. This is emphasized by the chosen values for the precision matrices which are larger for the sensory input compared to the believed input. This results in a system that is biased towards the desired set point.

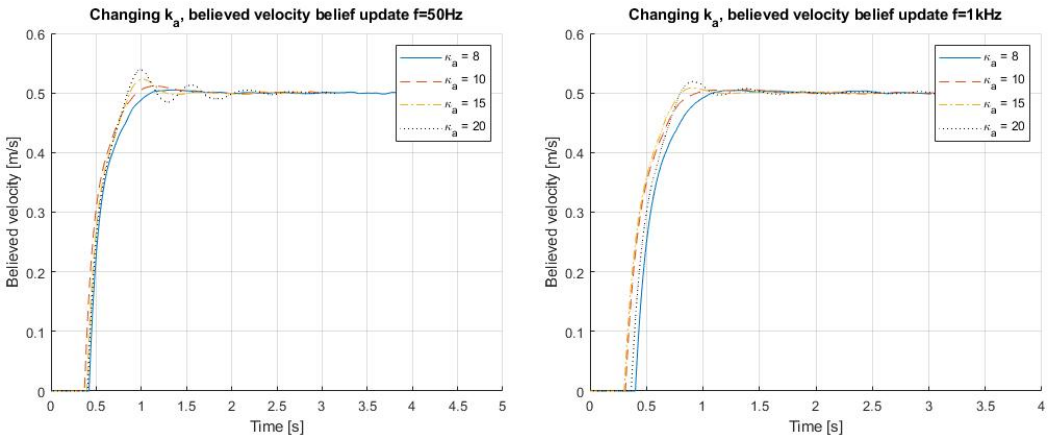


(a) Belief update frequency of 50Hz. **(b)** Belief update frequency of 1kHz.

Figure 5-18: Drive commands for a changing κ_a, similar behavior is observable as with the beliefs. The lower frequency suffers to ringing faster compared to the higher frequency.

Table 5-6: Table that shows the settling time and rise time while changing κ_a. The rise time decreases with an increase of the learning rate. The settling time decreases at first, but later ringing occurs which negatively affects the settling time.

	f = 50Hz		f = 1kHz	
κ_a	t_{set} [s]	t_{rise} [s]	t_{set} [s]	t_{rise} [s]
20	1.65	0.13	1.28	0.14
15	0.1.14	0.20	0.90	0.20
10	0.81	0.26	0.73	0.30
8	0.90	0.35	0.88	0.39



(a) Belief update frequency of 50Hz. **(b)** Belief update frequency of 1kHz.

Figure 5-19: Believed velocities, it can be seen that the beliefs remain smooth for the 1kHz case. For the 50Hz case ringing occurs for the higher values of κ_a, this could lead to instability for higher values.

5-4-4 Effect of the increased update frequency

In the previous sub sections the parameters τ, κ_μ and κ_a were tuned. A difference in the responses for the 50Hz and 1kHz systems was observed. To explain this the integration scheme will once again be used, equations 5-5 and 5-6. In the case of a higher frequency the time step, h, decreases and therefore the system takes smaller steps to calculate the actions and beliefs. This gives an advantage to higher frequency systems as the $\dot{a}$ and $\dot{\mu}$ can be larger while the system remains stable. This explains why the tuning parameters can be changed to larger values, as increasing the tuning parameters increases $\dot{a}$ and $\dot{\mu}$. This can be seen by for example looking at equation 5-3, where $\dot{\mu}$ was defined. Changing the parameters directly affects $\dot{\mu}$ and increasing them increases $\dot{\mu}$, and thus to estimate μ a larger value of $\dot{\mu}$ is used. Thus, the smaller time step for a system with a higher belief update frequency makes the larger ranges of tuning parameters possible.

$$a_t = a_{t-1} + h\dot{a}_t \tag{5-5}$$

$$\mu_t = \mu_{t-1} + h\dot{\mu}_t \tag{5-6}$$

Another advantage of the higher frequency system is that it calculates the beliefs more often, this makes it possible to react to changing circumstances. This is simply due to the fact that the higher frequency system, takes more steps in the same period of time. Thus if something changes, it is possibly able to faster detect it and adapt to it.

All in all, changing the beliefs update frequency alone does not help decrease the settling time of the Active Inference controller. It does make it possible to use a larger range of values for the tuning parameters which in its turn help the system to settle faster. Increasing the update parameters comes at a cost, for example increasing the parameter τ results in a shorter rise time yet it does introduce a bit of overshoot.

5-4-5 Ranges of the tuning parameter

In the beginning of this section the used values of the tuning parameters were shown (table C-1). This section aims to report the maximum usable values of the tuning parameters, as these differ for the 50Hz and 1kHz case both will be documented. Earlier in the section some of the maximum ranges for the 50Hz controller where already shown as the controller became unstable. The other values were found by using the robot on a box and evaluate the performance. The resulting maximum, or in the case of τ minimum values, can be found in table 5-7. Note that for κ_a the values are the same, it was decided to report the highest value that improved the system as was found in the previous section. The difference in maximum tuning parameters can be explained using the integration scheme as is done in the previous section.

Table 5-7: Maximum values usable for the active inference controller.

Parameter	50Hz	1kHz
τ	0.005	0.0002
κ_{mu}	0.1	3
κ_a	10	10

5-5 Improved Active Inference controller vs standard Active Inference controller

In the previous section it was found that increasing the belief update frequency in combination with tuning of the parameters could lead to an improved Active Inference controller. For a first improved Active Inference controller it was decided to only increase the temporal parameter which resulted in a controller with the parameters as found in table 5-8. To test whether the controller is indeed faster than the initial controller a step response test will be performed. There will also be a square wave test, this test is used to see whether the overshoot is an issue for the system.

Table 5-8: Parameters used for the improved and standard Active Inference controller.

Parameter	Improved AIC	Standard AIC
Belief update frequency	1kHz	50Hz
τ	0.001	0.03
κ_{mu}	0.1	0.1
κ_a	8	8

5-5-1 Step response

In figure 5-20 the velocity of the Jackal is plotted over time for the standard Active Inference controller and the improved Active Inference controller. It can be seen that the rise time is less for the improved controller and the settling time is shorter. This is more clearly visible in the drive commands of figure 5-21a. The believed velocity is shown in figure 5-21 and it can be seen that these converge a lot faster. Lastly in table 5-9 the measured rise and settling times are reported and it can be seen that the improved controller converges 0.21s faster.

Table 5-9: Resulting step response times for the standard Active Inference controller and the improved Active Inference controller.

	Standard		Improved	
Step [m/s]	t_{set} [s]	t_{rise} [s]	t_{set} [s]	t_{rise} [s]
0.5	0.88	0.36	0.51	0.19

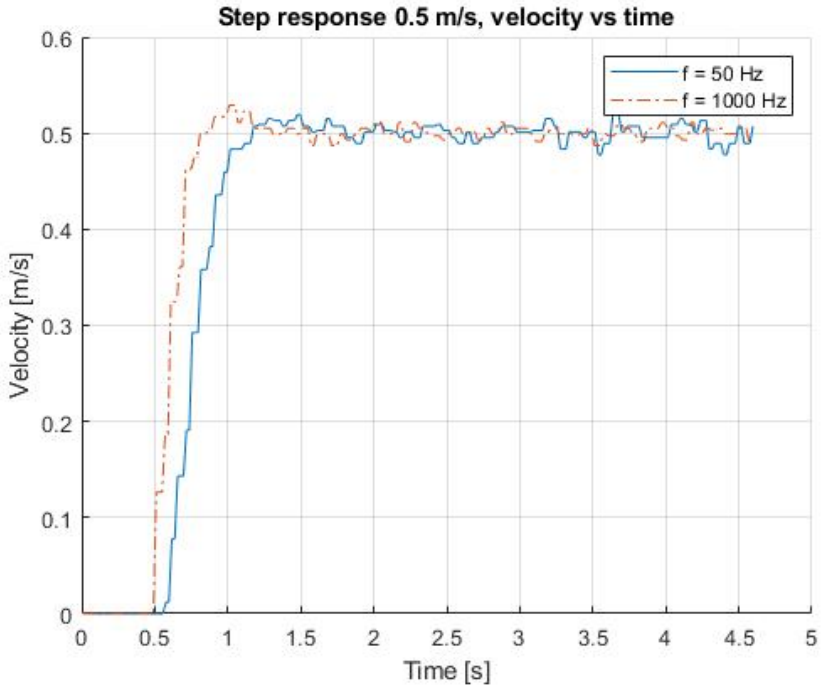
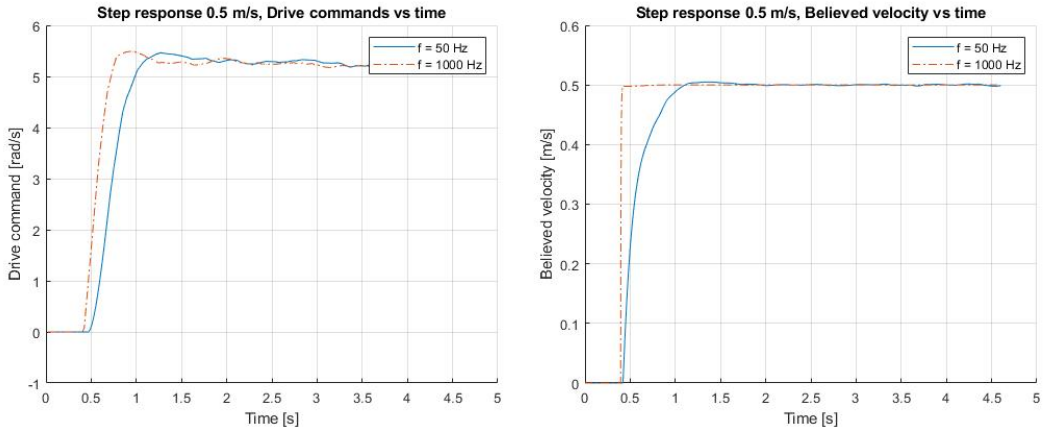


Figure 5-20: The measured velocity of the Jackal for the improved and standard Active Inference controller. It can be seen that the improved controller converges faster and that the rise time decreased as well.



(a) Drive commands. **(b)** Believed velocity.

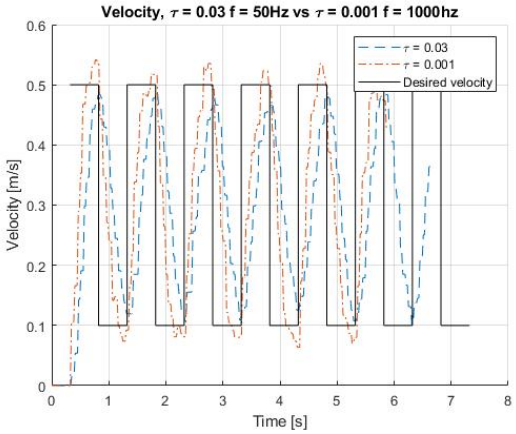
Figure 5-21: figures 5-21a and 5-21 display the drive commands and believed velocity of the Jackal for the standard and improved Active Inference controller respectively. It can be seen that the drive commands for the improved controller converge faster, yet have a bit more overshoot. The beliefs converge faster for the improved controller, which causes the system to converge faster.

5-5-2 Square wave tests

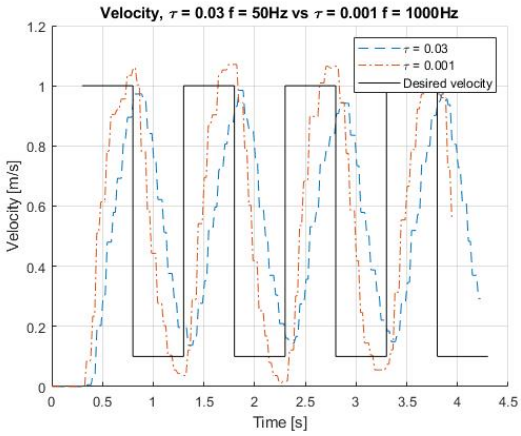
This section focuses on the results of the square wave test, two different square waves were used as inputs. The square wave test was performed to see whether the overshoot would make it impossible to track the square wave, or lead to instability for an altering velocity. In table 5-10 the minimum and maximum values of the square wave inputs can be found. In figures 5-22a to 5-23 the velocity of the Jackal for the different square waves is plotted for both the standard and the improved Active Inference controller. In the graphs the desired velocity is plotted as well, this is the set point given to the controller. It can be seen that in general the improved controller rises a bit faster than the original controller. The overshoot does not cause instability and makes it possible for the controller to reach the desired point, whereas the original controller fails in this task.

Table 5-10: Minimum and maximum values of the square wave inputs.

Case	Minimum	Maximum
1	0.1	0.5
2	0.1	1
3	0.2	1.2



(a) Square wave 1



(b) Square wave 2

Figure 5-22: Results of the square wave tests. The improved controller rises faster and has a bit of overshoot. The original controller does not converge to the desired set point. The goal was to see whether the system would remain stable and this is the case

5-5-3 Improved Active Inference vs differential drive

In chapter 3 the differential drive controller of the Jackal was compared to the basic Active Inference controller. At that point the Active Inference controller had a longer rise and settle time. The improved Active Inference controller is also tested against the original Jackal controller. In figure 5-24 the measured velocity and drive commands for this test are shown. The settle time of the Active Inference controller is now, 0.51s versus 0.54s of the differential drive controller. A clearer difference can be found in the rise time of the Active Inference controller this is 0.19s while the differential drive controller has a rise time of 0.35s. Therefore it can be said that the Active Inference controller now performs better than the differential drive controller for this test.

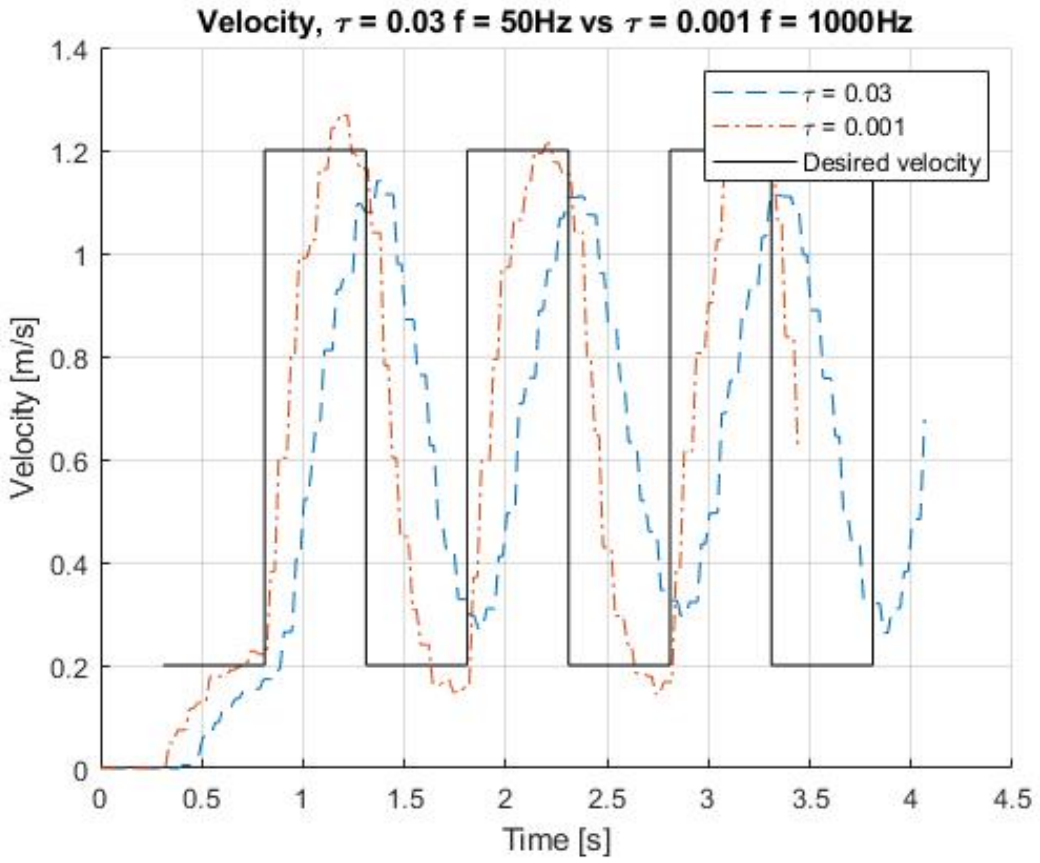
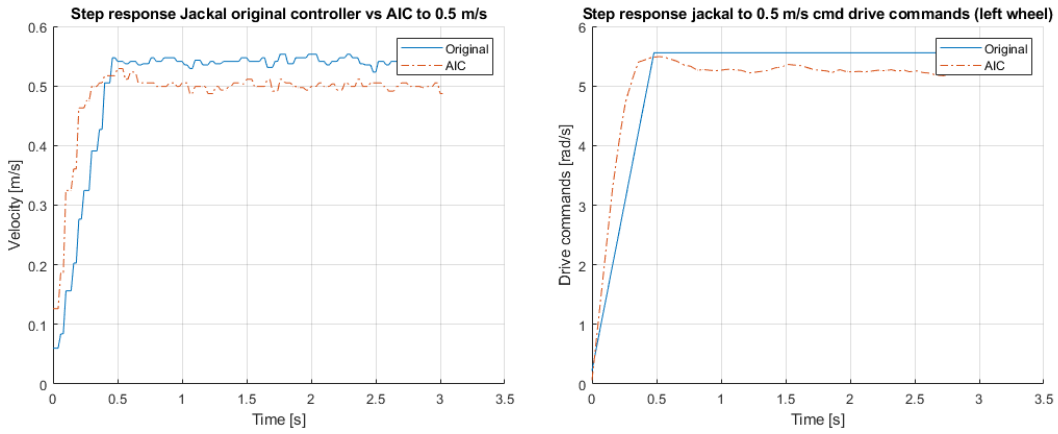


Figure 5-23: Square wave 3, this is the largest square wave. It can be seen that the new improved controller has some overshoot, the standard controller does not reach the desired set point.



(a) Measured velocity. **(b)** Drive commands.

Figure 5-24: In these figure the performance of the differential drive controller, which is the standard implemented controller for the Jackal, is compared to that of the Active Inference controller. It can be seen that the Active Inference controller is more accurate, as the differential drive controller settles at a higher velocity, and it has a shorter rise and settling time.

Conclusion and future work

The main goal of this thesis was to find a method to decrease the response time, or settling time, of an Active Inference controller, while maintaining the accuracy of the controller. To do so, a state space model of the Active Inference controller was created to gain a better understanding of the controller. From this state space controller it became clear that the belief update of the Active Inference controller is of interest if one wants to decrease the settling time. Three different methods to speed up the beliefs were explored, for the first method the update frequency of the entire controller was increased which did not help decrease the settling time. For the second method the update frequency of the belief update loop was increased, this resulted in a smoother controller and combined with the third method helped decrease the settling time. For the third method the belief update frequency was increased and the parameters were tuned, as it was found that these could be tuned to a larger range of values if the update frequency is increased. This last method proved efficient to decrease the settling time of the Active Inference controller, although it did introduce overshoot. Therefore the main research goal of this thesis, *how can the response time of an Active Inference controller be decreased, while maintaining the accuracy of the controller?*, is answered.

Throughout this process some sub-goals were set and answered. First of all, together with three other students an Active Inference velocity controller to control a Jackal robot was built. The controller was built and compared to the standard differential drive controller in section 3-4 and it was found that it initially performed worse than the the differential drive controller. After increasing the performance of the Active Inference controller it performed better than the differential drive controller.

A second goal was to find out *if a state space model can be used to analyze and simulate an Active Inference controller?* Therefore a state space Active Inference controller was derived in section 4-1, it was found that this model can be used to gain a better understanding of the working principles of the Active Inference controller. From this model it was found that the actions always trail the beliefs, and more importantly the actions are dependent on the convergence speed of the beliefs. A limitation in the model was found as the actual controller becomes unstable at a certain point whereas the state space model remains stable.

During the process the Gazebo environment was used to simulate the robot, therefore the following research goal can also be answered: *can the Gazebo environment be used to simulate an Active Inference controller for a Jackal robot?* It can be concluded that the Gazebo simulation is usable, however there are differences between the actual response and the response in Gazebo. This could potentially lead to issues if a controller is tuned in Gazebo and then used

on the real robot. The system can become unstable on the actual robot while it remained stable in Gazebo.

Is it beneficial for the accurateness of the controller to increase the update frequency of the entire system, or part of the system? It was found that increasing the update frequency of both the entire system and only the belief update does help the accurateness of the controller. It helps the controller to faster adapt to changes and it makes it able to better filter noise. This in it's turn helped with the research question, *does increasing the update frequency of the system or part of the system affect the tuning parameter of the Active Inference controller?* Changing the frequency of the system makes it possible to use a larger range of values for the parameters, which helps the Active Inference controller to converge faster. It was found that changing the belief update frequency helps most, changing the action frequency does not have an additional affect on the performance.

The last research question asked *What are the maximum values that can be used for the tuning parameters?* In table 5-7 the maximum, and in the case τ the minimum, values of the tuning parameters were shown. It was observed that the values differ for changing update frequencies, which was explained using the integration scheme.

6-1 Future work

In this thesis the first steps to a faster converging Active Inference controller were set, focusing on the update frequencies of the controller. Eventually it was found that by increasing the frequency a larger range of values could be used for the tuning parameters. A first step for future work would be to change more than one parameter at once, this could make the controller faster. A second possibility that could help decrease the settle time of the Active Inference controller would be to change the gradient descent method. Changing the update frequency essentially changes the step size for the gradient descent, which came with some benefits although it did not change the performance. Using a different method of gradient descent could help and would be an interesting field of further research. As a third possibility it would be interesting to change the amount of generalized orders and see whether the effect would be the same, or whether this would change. Even more so, it might be possible that the current changes only work for the used amount of generalized orders. A last possibility would be to explore the generative model of the state dynamics. In current applications an attractor dynamics model that assumes that the system converges towards a certain desired point is used for the state dynamics. This model could be changed towards a dynamic model that predicts the path towards the desired state, to gain a more accurate controller.

Bibliography

[1] K. Friston, "The free-energy principle: a unified brain theory?," *Nature reviews neuroscience*, vol. 11, no. 2, pp. 127–138, 2010.

[2] C. Pezzato, R. Ferrari, and C. H. Corbato, "A novel adaptive controller for robot manipulators based on active inference," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2973–2980, 2020.

[3] M. Baioumy, P. Duckworth, B. Lacerda, and N. Hawes, "Active inference for integrated state-estimation, control, and learning," *arXiv preprint arXiv:2005.05894*, 2020.

[4] N. Siddique and H. Adeli, "Nature inspired computing: an overview and some future directions," *Cognitive computation*, vol. 7, no. 6, pp. 706–714, 2015.

[5] K. K. Parhi and N. K. Unnikrishnan, "Brain-inspired computing: Models and architectures," *IEEE Open Journal of Circuits and Systems*, vol. 1, pp. 185–204, 2020.

[6] K. Friston, J. Kilner, and L. Harrison, "A free energy principle for the brain," *Journal of Physiology-Paris*, vol. 100, pp. 70–87, July 2006.

[7] C. L. Buckley, C. S. Kim, S. McGregor, and A. K. Seth, "The free energy principle for action and perception: A mathematical review," *Journal of Mathematical Psychology*, vol. 81, pp. 55–79, 2017.

[8] L. Pio-Lopez, A. Nizard, K. Friston, and G. Pezzulo, "Active inference and robot control: a case study," *Journal of The Royal Society Interface*, vol. 13, no. 122, p. 20160616, 2016.

[9] P. Lanillos and G. Cheng, "Active inference with function learning for robot body perception," in *Proc. Int. Workshop Continual Unsupervised Sensorimotor Learn.*, pp. 1–5, 2018.

[10] G. Oliver, P. Lanillos, and G. Cheng, "Active inference body perception and action for humanoid robots," *arXiv preprint arXiv:1906.03022*, 2019.

[11] M. Baioumy, C. Pezzato, R. Ferrari, C. H. Corbato, and N. Hawes, "Fault-tolerant control of robot manipulators with sensory faults using unbiased active inference," *arXiv preprint arXiv:2104.01817*, 2021.

[12] K. Friston, J. Mattout, N. Trujillo-Barreto, J. Ashburner, and W. Penny, "Variational free energy and the laplace approximation," *Neuroimage*, vol. 34, no. 1, pp. 220–234, 2007.

[13] I. Hijne, "Generalised motions in active inference by finite differences: Active inference in robotics," 2020.

[14] K. J. Friston, J. Daunizeau, and S. J. Kiebel, "Reinforcement learning or active inference?," *PloS one*, vol. 4, no. 7, p. e6421, 2009.

[15] F. V. C. Vuik, P. Van Beek and J. V. Kan, *Numerieke methoden voor differentiaal vergelijkingen*. VSSD, 2006.

[16] A. Mandow, J. Martínez, J. Morales, J. L. Blanco, A. Garcia, and J. González-Jiménez, "Experimental kinematics for wheeled skid-steer mobile robots," pp. 1222 – 1227, 12 2007.

[17] G. Anousaki and K. Kyriakopoulos, "A dead-reckoning scheme for skid-steered vehicles in outdoor environments.," pp. 580–585, 01 2004.

[18] S. Moosavian and A. Kalantari, "Experimental slip estimation for exact kinematics modeling and control of a tracked mobile robot," pp. 95 – 100, 10 2008.

[19] T. Wang, Y. Wu, J. Liang, C. Han, J. Chen, and Q. Zhao, "Analysis and experimental kinematics of a skid-steering wheeled robot based on a laser scanner sensor," *Sensors*, vol. 15, pp. 9681–9702, 05 2015.

[20] N. Seegmiller, *Dynamic Model Formulation and Calibration for Wheeled Mobile Robots*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, October 2014.

[21] I. Hijne, "Generalised Motions in Active Inference by Finite Differences," *TU Delft repository*, Aug. 2020.

State space simulator

```
function [y,x,tr, A, B, C,error_y, error_mu] =  ...
    aic_ss_simulator(pi_mu,pi_y, precision_mu, D, k_a, k_mu, tau, p, ...
    dt,T_end,mu_des,dgrd,errors_bool)
%This function accepts the parameters that are used for an active inference
%controller, time step dt, end time and desired beliefs. It outputs the y
%and x states of a simulated state space system. The goal is to obtain a
%better insight in the dynamics of the system.


%Inputs: pi_mu, precision matrix for the beliefs. [sigma_mu 0; 0 sigma_mu']
% pi_my, precision matrix for the measurements. [sigma_y 0; 0 sigma_y']
% precision_mu, vector that contains the precision sigma values of mu, ...
    needed to compute B [Sigma_mu, Sigma_mu'..]
% D, derivative operator. [0 1; 0 0]
%k_a learning rate for the action update
%k_mu learning rate fot he belief update
% tau temporal parameter, used in the attractor dynamics model
% p #of generalized coordinates
%dt time step
%T end time of the simulation
%mu_des, desired mu state i.e. states to which the system evolves


%Outputs Y velocity of the system, vector 1xn (n = T_end/dt)
%x = contains the beliefs and actions over time matrix 2*(p+1)Xn


%Create tau matrix, used in the determination of A
tau_mat = create_tau_matrices(p,tau);

%Definition of A,B and C matrices
A = [D - k_mu*tau_mat*pi_mu*D - k_mu*tau_mat*pi_mu/tau-k_mu*pi_y -k_mu*pi_y;
    k_a*pi_y -k_a*pi_y];

B = [k_mu*pi_mu(1,1)/(tau*tau);
     k_mu*pi_mu(1,1)/tau;
     zeros((2*p),1)];
C  = [zeros(1,p+1) 1 zeros(1,p)];

y = zeros(1,T_end/dt);
x = zeros(2*(p+1),T_end/dt);
```

```
39 error_y = zeros(p,T_end/dt);
40 error_mu = zeros(p,T_end/dt);
41
42
43 x_help = x(:,1);
44
45 for i = 1:T_end/dt
46     %simulation of the model
47     x_dot = A*x(:,i) + B*mu_des;
48     if dgrd
49         for j = 1:5
50             x_help = x_help+x_dot*dt;
51             x_dot = A*x_help+B*mu_des;
52         end
53         x(:,i+1) = x_help;
54         y(:,i) = C*x(:,i);
55     else
56     x(:,i+1) = x(:,i) + x_dot*dt;
57
58     y(:,i) = C*x(:,i);
59     end
60     if errors_bool
61     %error terms calculation
62     error_y(:,i) = x((p+3):end,i) - x(2:(p+1),i);
63     error_mu(1,i) = x(2,i)-(mu_des-x(1,i))/tau;
64     error_mu(2:p,i) = x((3:p+1),i) - x(2:p,1)/tau;
65     end
66 end
67
68 timeArray = 0:dt:T_end/dt;
69 tr10 = find(y>0.1*mu_des,1);
70 tr90 = find(y>0.9*mu_des,1);
71 tr = timeArray(tr90)-timeArray(tr10);
72
73 end
```

Active inference controller block diagram

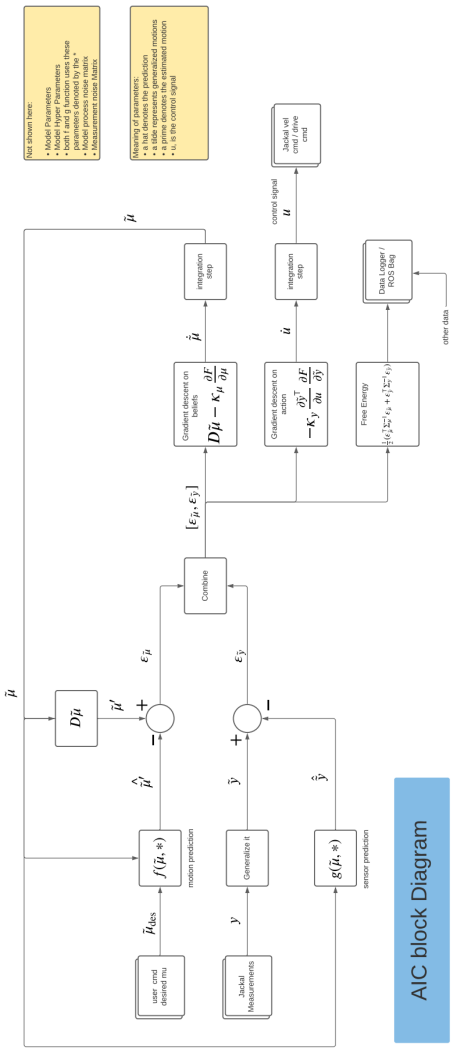


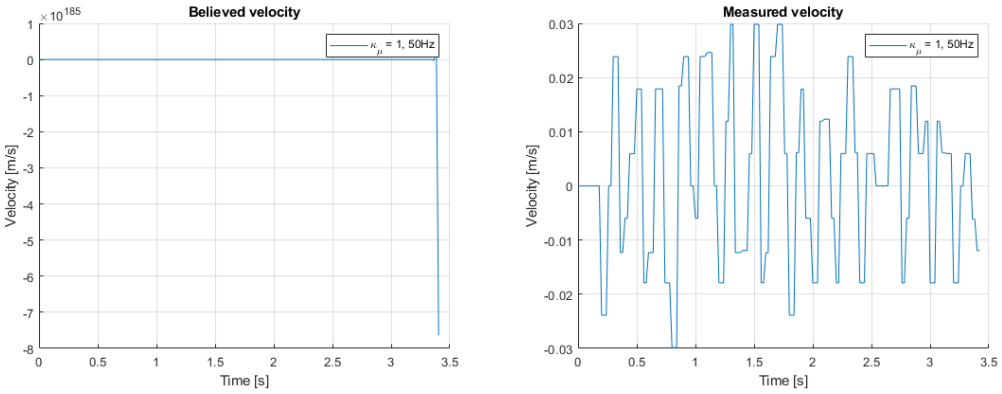
Figure B-1: Active inference control block diagram as it was introduced by the Jackal team

Parameters of the Active Inference controller

Table C-1: Parameters of the basic Active inference controller.

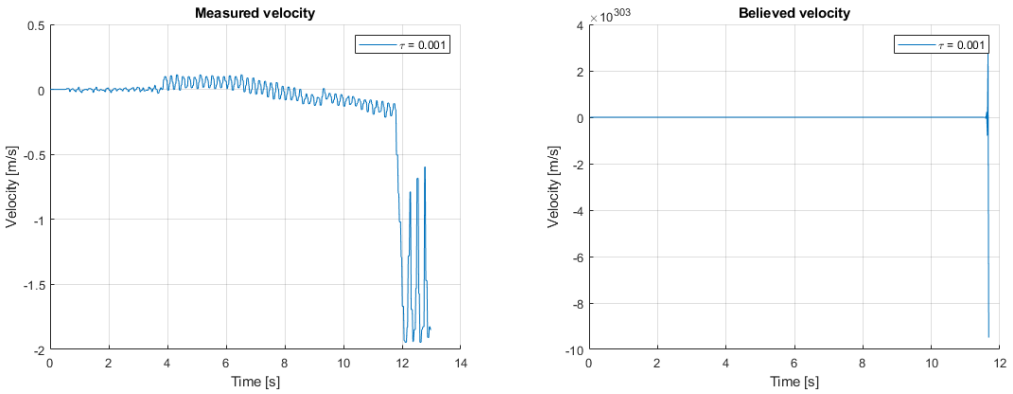
Parameter	Value
τ	0.03
κ_μ	0.01
κ_a	8
Generalized orders	3
Σ_μ	1.5
$\Sigma_{\mu'}$	2.0
$\Sigma_{\mu''}$	4.0
$\Sigma_{\mu'''}$	8.0
Σ_y	2
$\Sigma_{y'}$	2
$\Sigma_{y''}$	100000
$\Sigma_{y'''}$	100000
F_{update}	50Hz

Unstable step response cases



(a) Measured wheel velocity **(b)** Believed velocity.

Figure D-1: Step response results for the unstable case of κ_μ=1 for an update frequency of 50Hz. It is observable that the measured velocity does not converge to the desired point (0.5m/s) and the believed velocity becomes infinite.



(a) Measured wheel velocity **(b)** Believed velocity.

Figure D-2: Step response results for the unstable case of τ=0.001 for an update frequency of 50Hz. It is observable that the measured velocity and the believed velocity become unstable.