Active Inference for Robot Control: A Factor Graph Approach

Mohamed Baioumy* Daan van der Lans* Rens de Rooij* Mees Vanderbroeck* Tiis van der Werf*

Abstract— **Active Inference provides a framework for perception, action and learning, where the optimization is done by minimizing the Free-Energy of a system. This paper explores whether active inference can be used for closed-loop control of a 1 degree of freedom robot arm. This is done by implementing variational message passing on Forney-style factor graphs; a probabilistic programming framework. By comparing the created active inference controller to a PID-controller, we show that an active inference controller with variational message passing can perform state estimation and control at the same time, without loss of performance. The active inference controller outperforms the PID-controller in handling impulse disturbances, inaccurate system models and observations with major noise. With the use of variational message passing, computation time is sufficiently small to reach high controller frequencies.**

Keywords: Forney-style factor graphs, free-energy principle, active inference, closed-loop control, variational message passing

I. INTRODUCTION

The Free Energy Principle (FEP) has been proposed to provide a unified theory of the brain [1], providing an explanation for how cognitive functions such as perception, action and model learning are achieved [2] [3] [4]. It claims that in order for an intelligent agent to persist in a time-varying environment, it must minimize 'surprise' (the atypicality of an event). This is done by minimizing an upper bound called 'Free Energy', as organisms can not directly minimize surprisal.

Active Inference, corollary to the FEP, states that biological agents act to fulfill prior beliefs about preferred future observations. Desired behaviour is then achieved by minimizing Free Energy with respect to a generative model of the environment [5]. This generative model is an internal model of the environment that is used to produce actions and estimate posterior states.

Recently, Active Inference has been shown to be successful in computational and systems neuroscience [6]. It is argued that the framework could be adopted in robotics. This would provide a unified framework for robot state estimation (perception), action selection (control) and model learning [7]. However, there are no implementations of the full active inference construct for online control at the moment. So the benefit of perception, control and model learning combined remain unproven.

There have been several attempts of implementing active inference in robot control. In [8], a PR2 robot, simulated in ROS, is controlled by open-loop Active Inference. It is used to display arm behaviour previously obtained in a simplified human arm simulation in [9]. Lanillos et al. [10] use concepts from Active Inference for sensor data fusion for an interactive robot. Additionally, challenges in computational time of generation of control actions have been reported in [11].

The main contribution of this paper is to provide an *online* Active Inference implementation for control of a 1 degree of freedom (DoF) robot arm. The system is modeled as a stochastic State-Space Model implemented as Forney-Style Factor Graphs (FFG) [12].

In order to verify the validity of our Active Inference controller (AIC), it is tested to deal with sensor noise, force disturbances, incorrect initial beliefs and changes in the environment variables. A comparison to a PID controller is made to benchmark the AIC.

This paper is structured as follows. Section II contains the necessary background on the Free Energy Principle, Active Inference and Forney-Style Factor graphs. Section III introduces the implementation of the AIC on the simulated robot arm. Section IV provides the results of the Active Inference based control sequence compared to other, more conventional control strategies. Section IV provides the results of performance tests on the controller. In section V, the results are analysed and suggestions for future work are provided. Lastly, in section VI conclusions about the AIC are drawn.

II. Active Inference and Factor Graphs

This section offers a short technical recap of Active Inference and Factor graphs. The concepts will be introduced with the application, provided in III, in mind: a torque controlled robot arm equipped with proprioceptive sensors.

A. Variational free energy

The Free Energy Principle postulates that well-adapted biological agents maintain a probabilistic model of their typical environment (including their bodies). In order then to persist in a time-varying environment, these agents attempt to minimize the occurrence of events which are atypical in such an environment as estimated by their internal model. The atypicality of an event can be quantified by the negative logarithm of the probability of its sensory data $P(x)$, also known as 'surprise'. It is argued in the FEP that organisms can not minimize surprisal directly [13]. Instead they achieve this by minimizing an upper bound called the 'Free Energy'. In order not to confuse this with thermodynamic free energy, this quantity will be referred to as the *Variational Free Energy*, as done in [13].

* Authors are with the Faculty of 3mE, Delft University of Technology, The Netherlands.

To see this into practice, let us consider a robot arm which receives sensor data x about the value of its state and it tries to infer its actual hidden state z. This could be formulated by Bayes' theorem as:

$$p(z|x) = \frac{p(z)p(x|z)}{p(x)} \tag{1}$$

In equation 1, $p(x|z)$ is the sensory consequence of being in a physical state z and $p(z)$ is the prior belief of the environmental states. The marginalization of the likelihood over all the possible states is computationally tough. In this case, we can apply Variational Free Energy approximation [13] [14]. The core idea is to minimize Kullback-Leibler (KL) divergence between a distribution $q(z)$ that is encoded in the agent and the true posterior $p(z|x)$. Meaning, the true posterior distribution ($p(z|x)$) is often intractable for all different values so it is approximated with a distribution $q(z)$ that has a standard shape (for example a normal distribution).

$$\begin{aligned} KL(q(z), p(z|x)) &= \int q(z) \ln \frac{q(z)}{p(z|x)} dz \\ &= \int q(z)(\ln q(z) - \ln p(z,x))dz \\ &+ \ln p(x) \\ &= F + \ln p(x) \end{aligned} \tag{2}$$

The Kullback-Leibler is in essence a measure for dissimilarity between two distributions. Note that if $q(z)$ and $p(z|x)$ were identical their ratio would be 1 and its logarithm would be 0. This would mean the integral evaluates to 0 and thus there is no dissimilarity.

From equation 2, it is noted that minimizing F directly reduces KL since the term $\ln p(x)$ does not depend on $q(z)$. Instead of approximating $p(z|x)$ with the whole $q(z)$ distribution, the agent model is a delta distribution $\delta(z - s)$ that makes s the mean of the approximating density as done in section 3 of [5]. So z is the actual physical state of the robot and s is the estimation of this physical state which is encoded in the brain dynamics. Thus, we can remove the integrals simplifying the Variational Free Energy equation to:

$$F = -\ln p(s,x) = -\ln p(x|s) - \ln p(s) \tag{3}$$

For further reading and full derivations see [5], [10], [13], [11].

B. Active Inference

The minimization of VFE does not just account for perceptual inference but also for actions within the same framework. This means that while the beliefs are updated to better predict sensory data, actions are simultaneously executed on the environment to alter the sensory input and make these coincide with the sensory predictions [13]. This would mean that the true posterior $p(z|x)$ is extended to $p(z,u|x)$ where u are the actions. Subsequently, $q(z)$ is extended to $q(z,u)$.

This process can be represented by the schematic in figure 1. The agent and the environment are two statistically separated nodes, shown in blue and red respectively. These can interact with each other through a Markov Blanket [15]. A Markov blanket only contains the variables that a node uses to interact with the outside world, isolating the dynamics within that node from the outside. In our case, the agent represents the Active Inference controller and the environment represents the robot and corresponding environmental physics.

Instead of knowing the dynamics of the environment and its states exactly, the agent estimates the states and dynamics using an internal recognition model q. The agent furthermore observes the state output y from the environment imprecisely to some extent in the triangular node. Secondly, the control signal u and the environmental action a are linked in the square node. This control signal and the corresponding environmental action are generated by the agent, using observations of the states. To obtain this control signal, the calculation of the Variational Free Energy is done based on the sensory data and the generative model as explained in the previous paragraph.

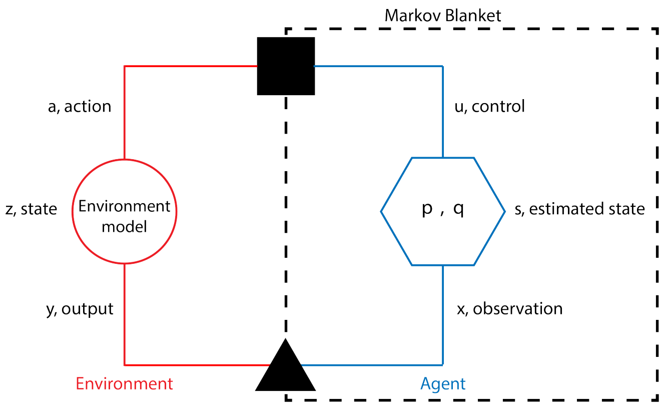

Fig. 1: Schematic of agent-environment interaction

C. Factor graphs

A Factor Graph is essentially a bipartite graph used to represent the factorization of a probability distribution function. It shares similarities with other methods such as Bayesian networks [15] and Markov random fields [16] and have similar notations. However, in this paper we utilize Forney-Style Factor Graphs (FFG) [17] because it is well-suited for representing dynamical models [18] and better suited for inference [19].

A FFG is built using two components, factors (represented as nodes in the graphs) and variables (represented as edges). A FFG can be used for stochastic models. For instance, let X be a real-valued random variable and let Y and Z be two independent real-valued noisy observations of X. These variables are randomly chosen for this example and are not related to the variables used in the method. This could resemble a robot trying to estimate its position based on information coming from two sensors. With factors 'f' corresponding to probability distributions, the factorized joint probability density of these variables is (with factors 'f' corresponding to probability distributions):

$$f(x, y, z) = f_a(x) f_b(x|y) f_c(x|z) \tag{4}$$

Now equation 4 can represented by the FFG in Figure 2a.

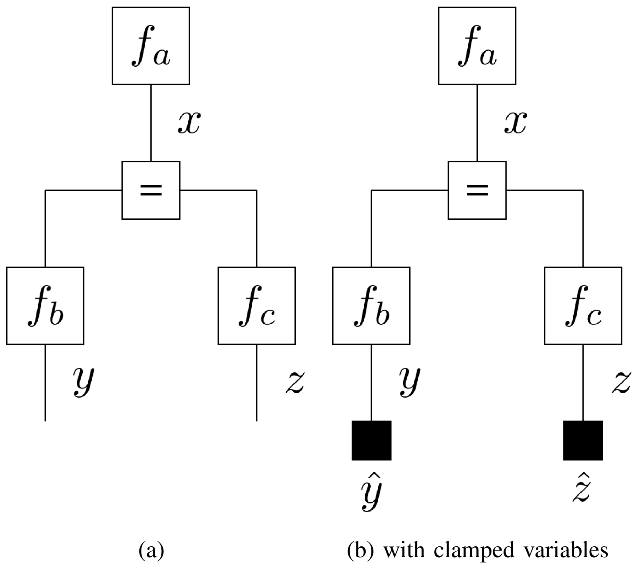

Fig. 2: Forney-style Factor Graph representing equation 4

In this example the node f_a is only connected to the variable x since it is only dependent on x. The $=$ resembles an equality node, meaning all edges connected to that node correspond to the same variable, in this case x. Finally, both f_b and f_c are connected to two variables since their factors depend on both, as seen in equation 4.

Now assume we observe $y = \hat{y}$ and $z = \hat{z}$, and are interested in computing a posterior distribution for x. The previous two observations impose two additional constraints $\delta(y-\hat{y})$ and $\delta(z-\hat{z})$ on the model, which clamp this variables to their observed values. Following [20] and [21], we indicate observations by a small solid node, see Figure 2b.

Computing the posterior distribution involves solving several integrals, see Equation 5. Solving this equation could be tedious and error-prone (examples shown in [18] and [21]). Using well-known message passing algorithms such as belief propagation [17] and variational message passing (VMP) [22] this problem could be solved.

$$\begin{aligned} f(x|y = \hat{y}, z = \hat{z}) \\ &\propto \iint f(x, y, z)\delta(y - \hat{y})\delta(z - \hat{z})dydz \\ &= f_a(x) \iint f_b(x|y)\delta(y - \hat{y}) f_c(x|z)\delta(z - \hat{z})dydz \end{aligned} \tag{5}$$

For our application, ForneyLab [23] is used. ForneyLab is a tool for automated derivation of message passing algorithms. It supports exible specications of factorized probabilistic dynamic models and generates high-performance inference algorithms on these models. It is implemented in Julia, a programming language designed to have the compiling speed of C, with the open-source libraries similar to Python [24].

III. Method

To research the feasibility of closed-loop AIC with FFG, a simulation of a 1 DoF robotic arm is set up. The robotic arm that is simulated can be seen in figure 3.

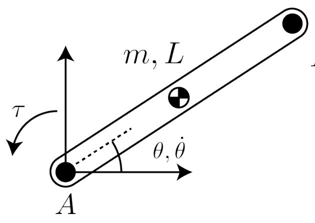

Fig. 3: Schematic of simulated robotic arm

The link can move relative to the fixed world in point A. Point B marks the end-effector position. The state of the robotic arm can be changed by the motor in A with torque τ. The sensor used is a proprioceptive sensor in A. The proprioceptive sensor outputs measurements of the angle and angular velocity between the robot arm and the horizontal axis (ϕ, $\dot{\phi}$). ϕ and $\dot{\phi}$ are sensor data that are captured in a probability function $p(\mathbf{x_t}|\mathbf{s_t})$. Be aware that ϕ and $\dot{\phi}$ are not necessarily identical to the real angle and angular velocity ($\theta, \dot{\theta}$), as they include sensor noise. All the variables used in this paper are summarized in table I.

TABLE I: Mathematical objects used in this paper

Symbol	**Name**
$\mathbf{z}$	Hidden environment state
$\mathbf{y}$	Output vector of environment, containing θ and $\dot{\theta}$
$\mathbf{s}$	Estimated environment state
$\mathbf{x}$	Observe vector, containing ϕ and $\dot{\phi}$
τ	Motor torque
a	Action caused by control signal
u	Control signal
s_0	Mean prior state
$p(x, s, u)$	Generative probabilistic model, estimate of environmental dynamics
$q(x, s, u)$	Local version of the generative probabilistic model $p(x, s, u)$
$F(p(x|s), p(s))$	Variational Free Energy (VFE)
Γ	State transition variance
α	Observation variance
$\mathbf{s_0}$	State prior
T	Lookahead

Since previous work listed challenges in computational time of generation of control actions [11], initially a relatively low control frequency was chosen (5 Hz). For further elaboration, see the paragraph on computation time in subsection IV-B.

A. Environment setup

The robotic arm operates in the horizontal plane, therefore the gravitational force has no influence on the motion dynamics. Friction is assumed to be negligible and thus has no influence on the motion dynamics as well. To simulate the saturation of the motor, the torque applied to the robot arm is limited to 10 Nm in both directions, for all possible control signals by equation 6.

$$\tau(a_t) = 10 * tanh(a_t) \tag{6}$$

The environment is updated by Euler integration, as in equation 7. Equation 7b is the short vector notation of 7a.

$$\begin{bmatrix}\dot{\theta}_t \\ \theta_t\end{bmatrix} = \begin{bmatrix}1 & 0 \\ dt & 1\end{bmatrix}\begin{bmatrix}\dot{\theta}_{t-1} \\ \theta_{t-1}\end{bmatrix} + \begin{bmatrix}\frac{\tau(a_t)*dt}{I_A} \\ 0\end{bmatrix} \tag{7a}$$

$$\mathbf{z_t} = \mathbf{g} * \mathbf{z_{t-1}} + \mathbf{h}(a_t) \tag{7b}$$

Where I_A is the mass moment of inertia around point A. To simulate the sensor data used in the AIC, Gaussian white noise is added to the real robot state values, with standard deviations σ_θ, $\sigma_{\dot{\theta}}$. The output vector y_t will contain the real (hidden) angle and angular velocity values θ, $\dot{\theta}$ similar to figure 1. The torque τ is given as a function of the action a_t. The control signal u_t from the agent generates these actions.

B. Internal (agent) model

In order to estimate the posterior states, the agent uses a stochastic state-space model. The execution of the generative model p_t is done with a factor graph represented in figure 4. The factorization of this factor graph is given by:

$$\begin{aligned} &p_t(\mathbf{x}, \mathbf{s}, \mathbf{u}) \propto \\ &p(\mathbf{s_{t-1}}) \prod_{n=t}^{t+T} \underbrace{p(\mathbf{x}_n|\mathbf{s}_n)}_{\text{observation}} \underbrace{p(\mathbf{s}_n|\mathbf{s_{n-1}}, u_n}_{\text{state transition}} \underbrace{p(u_n)}_{\text{control}} \underbrace{\tilde{p}(\mathbf{x_n})}_{\text{target}} \end{aligned} \tag{8}$$

Where the probability functions represent the model for the observation, state transition, control and target; which will be explained one by one. Note that now n is used to indicate a specific timestep, all variables that vary for each timestep will therefore be denoted with an n as subscript.

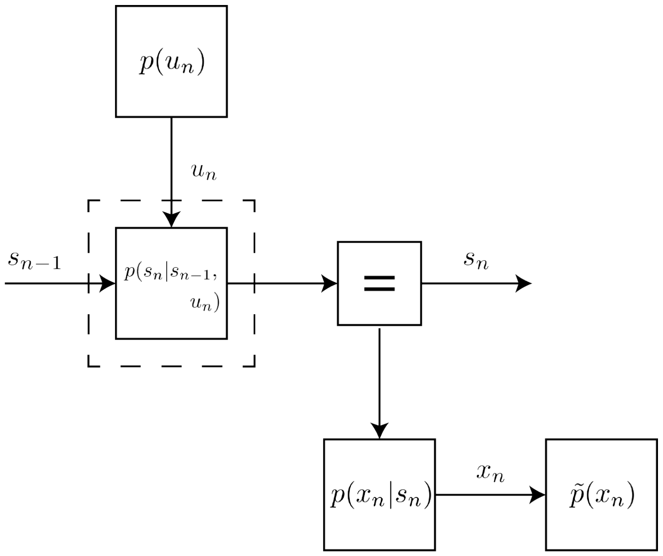

Fig. 4: Schematic of agent-environment interaction

The agent's target state is defined as $\mathbf{x_{ref}} = (\theta_{ref},\dot{\theta}_{ref}) = (2.0, 0.0)$. T is the number of steps that the algorithm will look ahead ("lookahead" parameter), so the factor graph will be generated for T timesteps in the future on each iteration. The target prior is defined as a normal distribution, marked as $\tilde{p}(\mathbf{x_n})$, to clearly distinguish it from the observation model notation. For timesteps before the first lookahead, this target prior has a large variance of 10^{12} to make a change in state have almost no effect on the Free Energy, allowing for unrestricted state changes, because the controller is not yet expected to have reached $\mathbf{x_{ref}}$. For $t \geqslant Tdt$, The target prior is set to $\mathbf{x_{ref}}$, with a small variance, 10^{-4} in this case, to make sure the robot arm reaches this state with high precision (equation 9).

$$\tilde{p}(\mathbf{x_n}) = \begin{cases}\mathcal{N}(\mathbf{x_n}|0, 10^{12}) & \text{if } t < Tdt \\ \mathcal{N}(\mathbf{x_n}|\mathbf{x_{ref}}, 10^{-4}) & \text{otherwise}\end{cases} \tag{9}$$

Furthermore, the agent has a transition model that captures the state dynamics for the estimated robot states s. The AIC uses this transition model, as defined in equation 10 to infer the environmental dynamics.

$$p(\mathbf{s_n}|\mathbf{s_{n-1}}, u_n) = \mathcal{N}(\mathbf{s_n}|p(\mathbf{s_{n-1}}, u_n), \Gamma) \tag{10}$$

Here Γ is the state transition variance. The value for Γ is chosen according to the belief of the accuracy of the generative model $p(\mathbf{s}, \mathbf{x}, u)$. For instance if a non-linear system is approximated by a linear model, you would resort to a higher variance. For the presented results, Γ was chosen to be 10^-4.

The generative model is represented by $p(\mathbf{s_{n-1}}, u_n)$ and closely resembles the environmental dynamics. This is indicated in figure 6 within the dashed box. In this case $\mathbf{g}$ and $\mathbf{h}$ are the same functions as in 7b.
The internal model uses an observation model, defined in equation 11, to predict the sensory outputs.

$$p(\mathbf{x_n}|\mathbf{s_n}) = \mathcal{N}(\mathbf{x_n}|\mathbf{s_n}, \alpha) \tag{11}$$

Here, α is the observation variance. The value of the observation variance is encoded in the agent, which resembles its belief about how noisy the observations of the environment are. Like Γ, the effect of α is elaborated in section IV-B.6. For the results presented in this paper, α was set to 10^{-4}.

Finally, the internal model has a state prior and control prior as defined in equations 12a and 12b respectively.

$$p(\mathbf{s_0}) = \mathcal{N}(\mathbf{s_0}|(0.0, 0.0), \sigma^2_{\mathbf{s_0}}) \tag{12a}$$

$$p(u_n) = \mathcal{N}(u_n|0.0, 10^{12}) \tag{12b}$$

Where $\sigma^2_{s_0}$ is defined dependent on how well the initial state of the robot arm is known. In our experiments, the AIC is given the assumption that the brain state perfectly matches the initial hidden state (i.e. the initial state is known), so the variance is small: $\sigma^2_{s_0} = 10^{-12}$. The control prior is given a large variance of 10^{12} to allow the controller to conduct a large range of control signals.

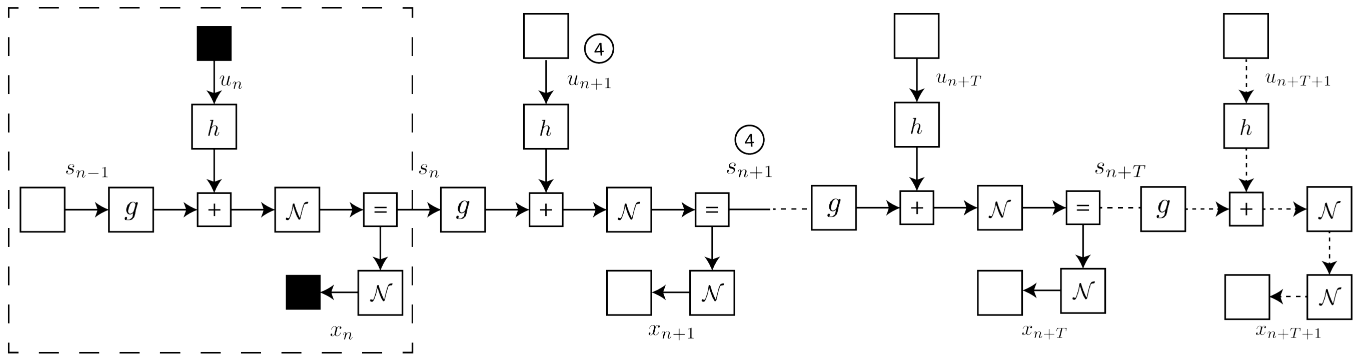

Fig. 5: Factor graph showing the slide function of the algorithm. The infer function is indicated with a 4.

C. Control loop

The algorithm loops through the functions act, execute, observe, infer and slide respectively, similar to [21], assuming we start at step n and simulate T timesteps, as in fig. 5. These functions are defined as follows:

1) Act: The act-function simply converts the control signal u, generated by the agent, into the action a, making their values equivalent. This action a will be applied to the environment. This is illustrated in figure 6 by the label 1.

2) Execute: Next up, the action a is used to calculate a certain torque, which will move the robot arm. The angle and angular speed of the robot are updated as described in equation 7. This way, the action is used to change the environment. So Execute happens in the environment as labelled by 2 in figure 6.

3) Observe: As the environment changes in the execute step, the angle and angular speed of the robot arm change. These new angles will be observed by the sensors as $\mathbf{x}$. This value is clamped as indicated by 3 in figure 6.

4) Infer: In the infer-function, VMP is used to calculate a new control signal u by minimizing VFE. The input for the VMP is the action a and observed $\mathbf{x}$, generated in the steps before. Subsequently, the next state and control action are inferred. This is shown in figure 5 by 4.

5) Slide: The last part of the algorithm is to remove the first time step from the factor graph, and to add one time step at $n + t + 1$. This is shown in figure 5, where the first step (encircled) is removed and the last step is added.

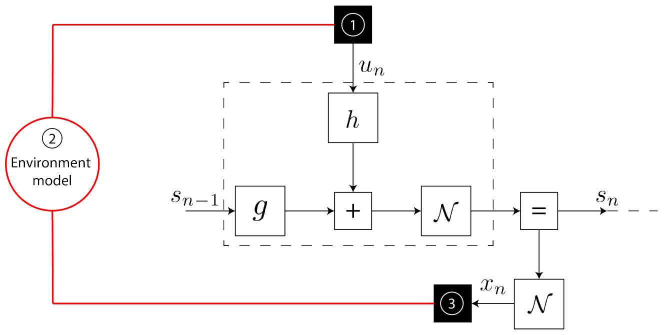

Fig. 6: Factor graph showing one timestep of the algorithm. The clamp block of the action is indicated with 1, the execute step is indicated with 2 and the controller observes the robot state with its sensors at 3.

IV. Experimental Results

The AIC is tested for its performance and robustness by a number of experiments. All experiments are executed with the same parameters and the same environment, unless stated otherwise. One by one, the parameters are changed and their influence is observed. Keep in mind that the agent can not directly observe the (real) environment. Sensor deviations will disturb this observation. However, the torque function $\mathbf{h}(a_t)$ is know by the agent. To analyze the AIC performance, a PID-controller is made for the same manipulator and is used as a benchmark. In other words: the environment is the same for both controllers.

A. Tuning the lookahead parameter

Before presenting the controller performance, we analyze the lookahead parameter. The lookahead (T) is a measure for the number of timesteps that the algorithm computes in advance on each iteration. As the lookahead is increased, the algorithm takes significantly longer to execute, which is evinced in subsection IV-B. For the experiments that are introduced later in this paper, there was no direct significant benefit to a high lookahead. The lookahead nonetheless needs to be large enough to initialize the algorithm appropriately, which requires an absolute minimum lookahead of 3. For lower lookahead, no control signal is generated in *ForneyLab*. The lookahead was set to 5 timesteps for this controller since higher values showed little to no improvement at the cost of significant compromises in computation. This value was chosen as it displayed good results at a computation speed comparable to lower lookahead values. In other cases planning further ahead may be required, such as the Mountain Car problem [25]. This problem is solved using *ForneyLab* in [21], where a lookahead of 20 is minimally required.

B. Controller performance

To observe the performance of the AIC, a number of experiments were conducted. To measure the general performance of the controller and show a proof of concept, the controller was instructed to go from an initial state to a target state, in this case from $\theta_{start} = 0$ rad to $\theta_{ref} = 2$ rad. Secondly, noise was added to the sensors. In a third experiment, angular velocity is changed suddenly to measure the influence of impulse force disturbances. Next, the response was observed when the initial beliefs of the agent did not match the starting position. In all experiments listed below, a timestep of 0.2 seconds and a lookahead of 5 timesteps are used.
To be able to draw conclusions about the level of performance of the AIC, a comparison to a benchmark PID controller is made for each experiment. To enable a fair comparison between the controllers, they are both tuned to a similar settling time and zero overshoot. Other performance qualities are then analyzed.

1) No noise: In figure 7, it can be seen that the controller is able to converge smoothly towards its target state. The results show that the rise time and settling time of the controller are both approximately 15 seconds for an error band of 5 %. Those are equal in this case, as the controller does not have any overshoot with the current settings. This response is used as a benchmark to test the effect of noise and disturbances.
Figure 8 shows output of the PID controller when it is tuned to similar performance as the AIC (similar rise time and settling time). The maximum used torque for the AIC is higher than the maximum torque output of the PID.

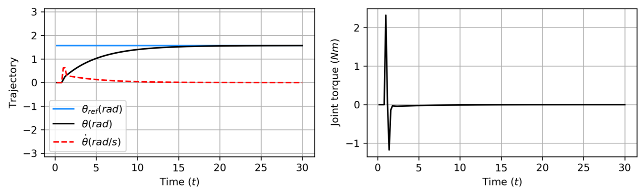

Fig. 7: Performance of the AIC with the standard settings and without any noise.

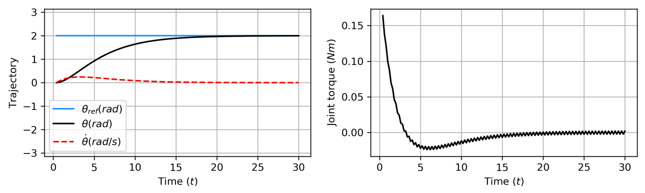

Fig. 8: Performance of the PID, tuned to performance similar to the AIC, without any noise.

2) With noise: Figures 9 and 10 show the controllers' response with Gaussian white noise on the sensors. The minor noise experiment is done with Gaussian white noise with a variance $\sigma_\theta^2 = 0.02$ rad. The major noise experiment is done with a variance of $\sigma_\theta^2 = 0.5$ rad. The response shows that the AIC is robust enough to withstand quite heavy noise. The rise time and overshoot are comparable to the no-noise experiment. The required input torque is quite acceptable which we will benchmark with the PID controller. The controller manages to create a remarkably smooth and steady trajectory for the robot arm from the noisy output of the sensors.

Figure 11 shows output of the PID controller when noise is added to the sensor output. The added white noise had the same variance as described before for the AIC; $\sigma_\theta^2 = 0.02$ rad and $\sigma_\theta^2 = 0.5$ rad respectively. Where there were hardly any differences observed for no noise performance, adding Gaussian white noise with variance $\sigma_\theta^2 = 0.02$ rad causes an increase in torque oscillations up to 1.5 Nm for the PID. This undesired behaviour increases the motor load. The oscillations for AIC are significantly smaller, as can be seen in figure 9. As noise with variance $\sigma_\theta^2 = 0.5$ rad is added, similar to the major noise that was applied to the AIC, the oscillations in PID torque increase, as can be seen in fig. 12. Also the oscillations in $theta$ around the reference value are increased with respect to the minor noise plot.

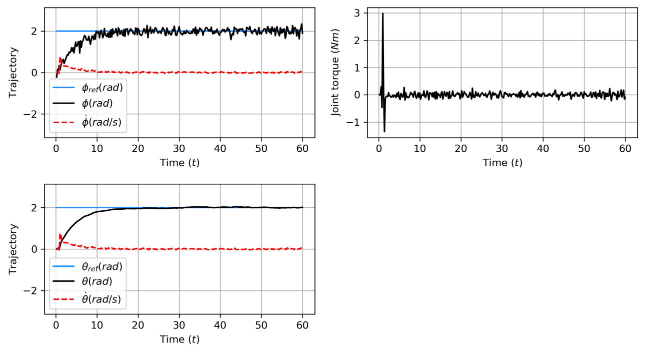

Fig. 9: Performance of the AIC with minor Gaussian white noise ($\sigma_\theta^2 = 0.02$) on the sensors.

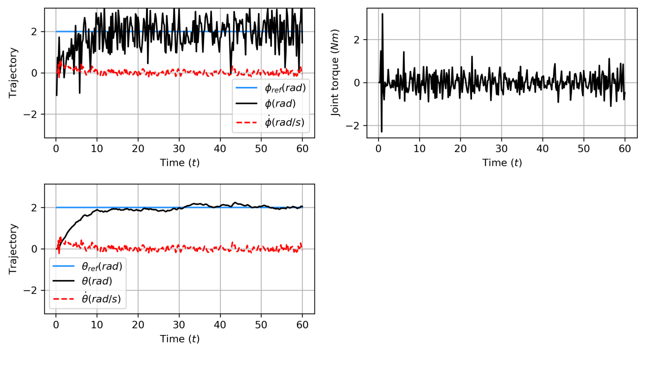
Fig. 10: Performance of the AIC with major Gaussian white noise ($\sigma_\theta^2 = 0.5$) on the sensors.

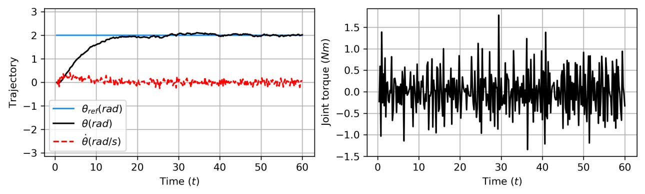
Fig. 11: Performance of the PID with minor Gaussian white noise ($\sigma_\theta^2 = 0.02$ rad) on the sensors.

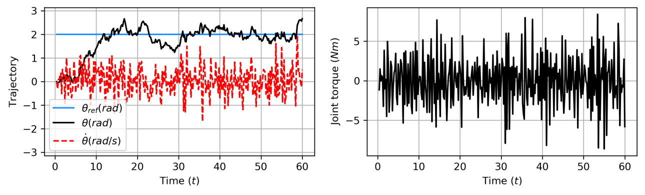
Fig. 12: Performance of the PID with major Gaussian white noise ($\sigma_\theta^2 = 0.5$ rad) on the sensors.

3) Impulse Disturbance: Figure 13 contains a plot of the response of the AIC to disturbance impulse forces put on the robot arm at $t = 20$ s and $t = 40$ s. These are modelled by a sudden velocity change of 0.6 rad/s downwards and 1.0 rad/s upwards respectively. The results show steady and predictable response to the disturbances. The robot moves back to the desired position in a similar manner as its initial response to a change in target position.
In figure 14 the response is shown when adding the same disturbances to the PID. Both controllers are able to return the robot arm to the target state. However, the AIC does this in significantly less time than the PID.

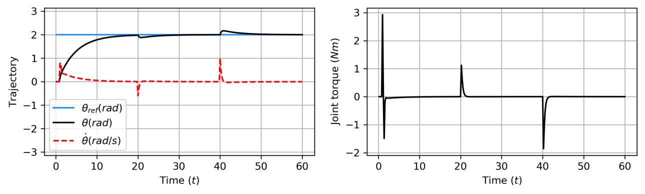
Fig. 13: Response of the AIC to impulse disturbances at $t = 20$ s and $t = 40$ s.

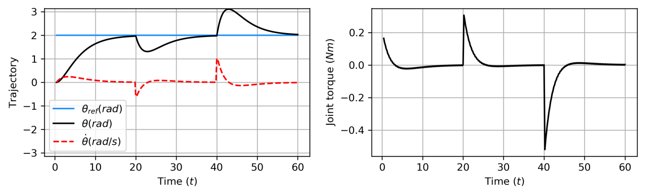
Fig. 14: Response of the PID to impulse disturbances at $t = 20$ s and $t = 40$ s.

4) Wrong initial beliefs: As stated before in section II, an active inference controller will minimize VFE by simultaneously adjusting its beliefs about the environment (state estimation) and the action on the environment (control). An incorrect internal belief would be rectified to a correct belief in order to minimize VFE. In figure 15, the response of the AIC is shown for a wrong initial belief. While the robot starts at $\theta = 0$ rad, the initial belief for the agent is set on $\phi = 4$ rad, so the robot should initially move down in order to achieve its desired position, given this wrong belief. The figure shows that the robot moves down for a very short time, before converging to the correct value.

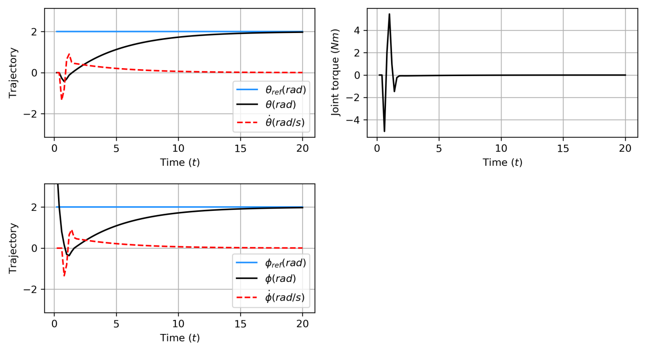
Fig. 15: Response of the AIC, starting with an incorrect initial belief

5) Changing environment model parameters: Both the AIC and the PID controller are tuned to act on a certain environment, in our case a robot arm with model parameters length and mass. To measure the robustness to a changing environment, the performance of both controllers was observed when different environmental variables were applied, while keeping the controllers tuned for the default model parameter values. This way, the performance of both controllers can be compared, when a wrong internal model is used. Decreasing the mass moment of inertia results in instability for both controllers, but this is mainly due to the discretization of the simulation with a large timestep of $dt = 0.2s$. An increase in moment of inertia results in a slower response for the PID. The AIC responds slower for large increases in the moment of inertia (e.g. factor of 50 or higher). For lower factors however, the rise time remains comparable, but the response shows some overshoot, as the robot arm decelerates slower than the AIC predicts in its internal model. This response is superior to the PID, as the system settles quicker and the overshoot is lower. This is visualized in 16 and 17, where the response of both controllers is shown when the moment of inertia is increased by a factor of 10.

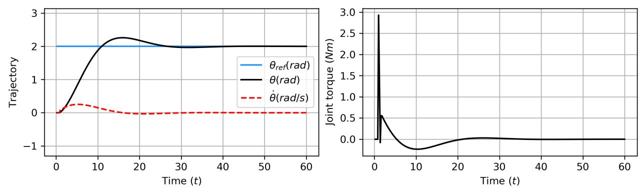

Fig. 16: Performance of the AIC when the mass moment of inertia is increased by a factor 10

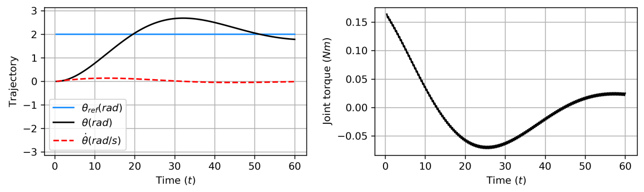

Fig. 17: Performance of the PID when the mass moment of inertia is increased by a factor 10

6) Adjusting agent model parameters: As discussed in section III, the state transition variance is chosen according to the agent's belief of the accuracy of generative model. This means that if the agent believes it has a proper model, a lower variance is chosen, which will cause the system to predict the next steps with a higher accuracy, resulting in faster responses.

Figure 18 shows a system with an accurate generative model and a transition variance set to 10^{-8} instead of 10^{-4}. The systems has a much faster response compared to previous results. However, if the generative model differs too much from the real dynamics, a low transition variance may cause the system to become unstable or fail to reach the target position. Note the minimal deviations for the angle and angular velocity even under noisy measurements. Additionally, the torque output settles for a value of 0 without oscillations.

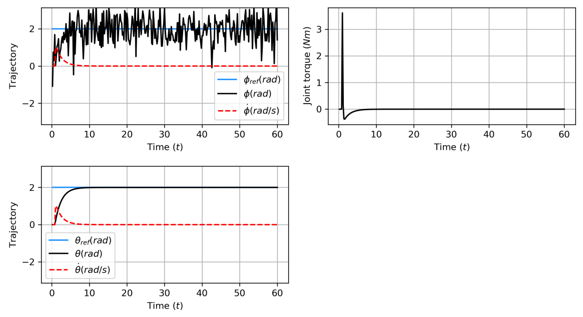

Fig. 18: System response in case of a correct generative model and a high state transition variance.

Additionally, the observation variance resembles the agent's belief about how noisy its measurements are. So a larger variance (10^{-1} instead of 10^{-4}) is required to perform well major noises ($\sigma_{\theta}^{2} = 0.5$). Figure 19 demonstrates this property.

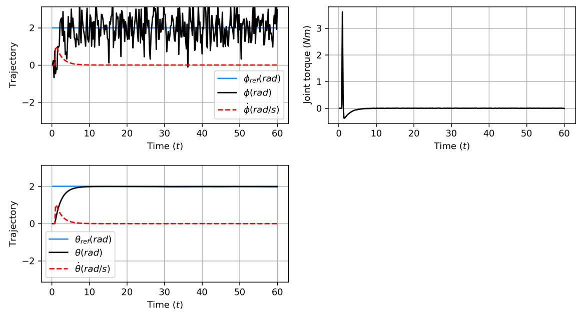

Fig. 19: System response in case of a correct generative model and a high observation variance.

7) Computation time: To determine the maximum controller frequency at which the AIC can work reliably, computation time for each iteration of the loop is measured. This test has been done with different values for the lookahead parameter (T). Results are shown in figure 20.

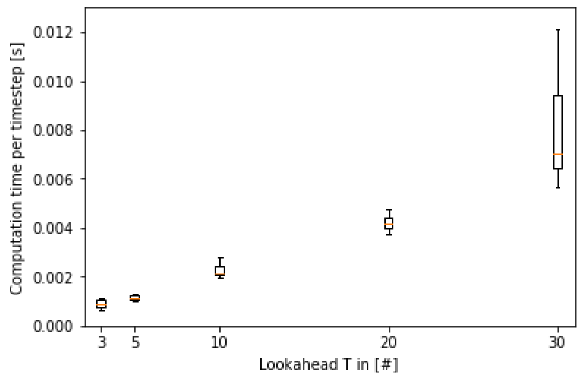

Fig. 20: Computation time per timestep for different T

These tests were conducted in Jupyter Notebook. The results show that the AIC can achieve high controller frequencies ($1/dt$), close to 1 kHz for T = 3. This is a significantly higher frequency than the initially used 5 Hz to obtain the results presented. The computation time per timestep increases significantly as the lookahead T is increased. This is caused by the fact that as the lookahead increases, the factor graph to be computed on each iteration becomes larger. The initialization of the algorithm also takes substantially longer as the lookahead is increased, but this does not directly affect the maximum controller frequency of the AIC.

V. Discussion

In this section, the results will be evaluated to get a better understanding of the meaning of the research done. Thereafter, the limitations of the controller are presented. Insights that are not fully understood are stated. Lastly, future work is proposed.

A. Benefits of the AIC

The controller presented in this paper shows to perform well in the experiments that were conducted. Despite the presented noise, impulse disturbances and initial belief imperfections, the controller behaves sufficiently for basic robot control. Moreover, the AIC is more robust to noise than the PID-contoller. Another benefit of the AIC is that the torque needed to control the arm is lower for the AIC than the PID when noise is introduced. This induces energy savings. Despite the computational challenges stated earlier, the controller executes the algorithm sufficiently fast. The behaviour of the controller is already good for a timestep of $dt = 0.2$ s, i.e. a control frequency of 5 Hz, as can be seen in 20. Under certain conditions the controller is able to reach a control frequency close to 1kHz. The limit of this control frequency is not investigated.

The AIC deals better with model imperfections than the PID as well. The PID-response becomes significantly slower, when the mass moment of inertia is increased. The AIC handles the increased mass moment of inertia better. The overshoot of the AIC increases, but the rise time is comparable to the response of the AIC for an environmental model that perfectly resembles the real environment. Furthermore, the settling time is quicker and the overshoot is lower for the AIC than for the PID.

Overall, the AIC shows more robust responses and requires less torque compared to the PID.

B. Limitations

Although this Active Inference controller is quite robust in the results obtained in this paper, the controller still has some limitations. The AIC has difficulties to deal with constant forces and the lookahead parameter has turned out to cause some issues, which will be further discussed below.

1) Lookahead: The AIC waits for a few timesteps before acting on its environment. This is due to the fact that the controller initializes a certain amount of time steps to build a factor graph and sets the mean prior input for those steps to zero. Changing the target after the robot arm has converged does not obviate this problem. It was tried to set the mean prior input for the first timesteps equal the desired state, but this can only be done for the first target state, keeping the same response for a change in target. Future work should be done to find ways to diminish the this behaviour of the AIC during the first timesteps.

2) Constant force (gravity): Attempts on dealing with a constant force; or gravity, have been only partially successful. The AIC is able to solve the inverted pendulum problem for equilibrium points, but no consistent tuning method was found to solve this problem for different environments. Solving these problems was simply done by adjusting parameters through trial-and-error. The controller seems to want to converge to a position where no torque is needed to hold that position. Since torque is needed to hold a non-equilibrium position, the Variational Free Energy is not minimized entirely and thus the controller will not converge to that position.
This could be solved by implementing prioritization in the controller, as proposed in [26]. This way the target state could be made more important to create a global Variational Free Energy minimum at that target. We tried to implement this prioritization by changing the variance on the target prior and the control prior. By doing so, the Variational Free Energy distribution is adjusted to allow for different controller response. An increased variance on the target prior turned out to allow the controller to solve a larger set of inverted pendulum problems (e.g. for larger mass), but this is limited to small values for the constant force. Different approaches might be needed to solve this. Methods to allow the controller to converge to a position with unknown, non-zero torque are yet to be contrived. This is pushed forward for future work.

C. Future work

In addition to further investigating the current limitations, additional future work to understand the potential of active inference based on variational message passing for control have been identified.

1) Model learning: Our current controller does not include generative model learning. The agent updates its beliefs about the environmental states, but uses an accurate internal model to realize this. It is unknown if the model learning is possible simultaneously with the perception and action updating in the *ForneyLab* library. However, we believe this is achievable in future work.

2) Scaling up to more degrees of freedom: Scaling the current controller to control a 2 degree of freedom robot arm has been attempted, but did not yet succeed. As the *ForneyLab* library is not specifically built to use for robot control, the implementation of higher order systems proves to be difficult. The inference of data of higher dimensions often induces errors. With more time and better understanding of the functions in the *ForneyLab* library, it should be possible to implement an AIC for a multiple degree of freedom robot arm in the future.

3) Tuning: To obtain stable and robust behaviour in the experiments, we tuned the AIC as described in section IV-A. Many predefined variances and parameters can be altered to increase the performance of the controller. In short, the method we used was based on trial and error. However, more tuning methods and more structured tuning may be possible to further improve the performance, mainly to decrease the rise time. This is subject for future work.

VI. Conclusions

In this paper we have presented a closed-loop Active Inference controlled (AIC) robot simulation, using a Factor Graph implementation. This controller is able to simultaneously update its beliefs about its state and perform actions on this environment, based on sensory input. It uses Free Energy minimization to calculate the control signals. The AIC was used to control a 1DoF robot arm in the horizontal plane. The results have shown that the controller is able to steer the robot arm to an arbitrary target position. The controller shows the ability to reach and maintain the desired position when Gaussian white noise and impulse disturbances are applied. Incorrect initial beliefs did not affect the rise time substantially. In the experiments, a control frequency of 5 Hz was used, which was sufficient to obtain the desired results. This frequency can be increased up to nearly 1 kHz to improve performance. A PID controller is presented as a benchmark, that is tuned to perform similar to our AIC for the simplest case. This results in similar behaviour in more complicated cases: adding disturbance forces and slightly changing the environmental variables. The results show that the AIC is able to handle sensor data with heavy noise, in contrast to the PID, which shows increased oscillatory behaviour for equal levels of noise. In addition, the response of the AIC deteriorates significantly less with model imperfections compared to the response of the PID. However, an implementation with more complicated disturbances such as gravitational forces has not yet been successful. Further research into the tuning methods for the AIC is required to fully understand and improve this control method. Another recommendation for future work is to fully implement online model learning. Altogether, we can conclude that the proposed Active Inference controller provides a way to control a robot that is robust to various disturbances and, with some extensions, shows promise for the future of robot control.

ACKNOWLEDGMENT

The authors thank Carlos Hernandez Corbato for supervising the research process and providing valuable feedback. We also thank Thijs van der Laar for answering questions about the *ForneyLab* library. Our gratitude also goes to Corrado Pezzato for giving us the first insights into Active Inference. Lastly, we thank the Cognitive Robotics department of the 3mE faculty of TU Delft for providing the assignment.

References

[1] Karl Friston. The free-energy principle: a unified brain theory? *Nature Reviews Neuroscience*, 11:127 EP –, 01 2010.

[2] Karl Friston. Friston k. the free-energy principle: a rough guide to the brain? trends cogn sci 13: 293-301. *Trends in cognitive sciences*, 13:293–301, 07 2009.

[3] Karl Friston, Philipp Schwartenbeck, Thomas FitzGerald, Michael Moutoussis, Tim Behrens, and Raymond J Dolan. The anatomy of choice: active inference and agency. *Frontiers in human neuroscience*, 7:598, 2013.

[4] Thomas HB FitzGerald, Philipp Schwartenbeck, Michael Moutoussis, Raymond J Dolan, and Karl Friston. Active inference, evidence accumulation, and the urn task. *Neural computation*, 27(2):306–328, 2015.

[5] Rafal Bogacz. A tutorial on the free-energy framework for modelling perception and learning. *Journal of mathematical psychology*, 76:198–211, 2017.

[6] Karl Friston and Christopher Frith. Active inference, communication and hermeneutics. *Cortex*, 68:129–143, 2015.

[7] Christoph Mathys, Jean Daunizeau, Karl Friston, and Klaas Stephan. A bayesian foundation for individual learning under uncertainty. *Frontiers in Human Neuroscience*, 5:39, 2011.

[8] Léo Pio-Lopez, Ange Nizard, Karl Friston, and Giovanni Pezzulo. Active inference and robot control: a case study. *Journal of The Royal Society Interface*, 13(122):20160616, 2016.

[9] Karl J Friston, Jean Daunizeau, James Kilner, and Stefan J Kiebel. Action and behavior: a free-energy formulation. *Biological cybernetics*, 102(3):227–260, 2010.

[10] Pablo Lanillos and Gordon Cheng. Adaptive robot body learning and estimation through predictive coding. 05 2018.

[11] A Cibiach Mercade. Robot manipulator control under the active inference framework. Master's thesis, TU Delft, Delft University of Technology, 2018.

[12] H. . Loeliger. An introduction to factor graphs. *IEEE Signal Processing Magazine*, 21(1):28–41, Jan 2004.

[13] Christopher L Buckley, Chang Sub Kim, Simon McGregor, and Anil K Seth. The free energy principle for action and perception: A mathematical review. *Journal of Mathematical Psychology*, 81:55–79, 2017.

[14] Karl Friston. Hierarchical models in the brain. *PLoS computational biology*, 4(11):e1000211, 2008.

[15] Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Elsevier, 2014.

[16] Ross Kindermann. Markov random fields and their applications. *American mathematical society*, 1980.

[17] G David Forney. Codes on graphs: Normal realizations. *IEEE Transactions on Information Theory*, 47(2):520–548, 2001.

[18] Hans-Andrea Loeliger, Justin Dauwels, Junli Hu, Sascha Korl, Li Ping, and Frank R Kschischang. The factor graph approach to model-based signal processing. *Proceedings of the IEEE*, 95(6):1295–1322, 2007.

[19] Frank Dellaert, Michael Kaess, et al. Factor graphs for robot perception. *Foundations and Trends® in Robotics*, 6(1-2):1–139, 2017.

[20] Christoph Reller. *State-space methods in statistical signal processing: New ideas and applications*, volume 23. ETH Zurich, 2013.

[21] Thijs W van de Laar and Bert de Vries. Simulating active inference processes by message passing. *Frontiers in Robotics and AI*, 6(20), 2019.

[22] Justin Dauwels. On variational message passing on factor graphs. In *2007 IEEE International Symposium on Information Theory*, pages 2546–2550. IEEE, 2007.

[23] TW van de Laar, MGH Cox, and A de Vries. Forneylab. jl: a julia toolbox for factor graph-based probabilistic programming. In *ForneyLab. jl: a Julia Toolbox for Factor Graph-based Probabilistic Programming*. 2018.

[24] J. Bezanson, A. Edelman, S. Karpinski, and V. Shah. Julia: A fresh approach to numerical computing. *SIAM Review*, 59(1):65–98, 2017.

[25] Andrew William Moore. Efficient memory-based learning for robot control. 1990.

[26] Giovanni Pezzulo, Francesco Rigoli, and Karl J. Friston. Hierarchical active inference: A theory of motivated control. *Trends in Cognitive Sciences*, 22(4):294 – 306, 2018.