Radboud University Nijmegen

Faculty of Social Sciences

Context-Aware Active Inference for Robot Navigation

Thesis BSc Artificial Intelligence

Author:
Simon van der Wulp

Supervisor:
Pablo Lanillos

Second reader:
Sander Keemink

January 2022

Abstract

In neuroscience, perception-based pose estimation has given an insightful approach to further discovering human functions used for localization and navigation using sensory information. This paper seeks to draw inspiration from this by implementing an algorithm that performs sensory-based localization and navigation by minimizing variational free energy under the free energy principle. This is done by minimizing the difference between predicted sensory observations, obtained through a generative model of an environment, and actual observations. We expand on this algorithm by adding a classifier that maps sensory observations to environment types. This gives the ability to switch between different generative models for different types of environments which increases overall adaptability and flexibility. By implementing the algorithm in a virtual robot, we can evaluate the localization and navigation capabilities of the suggested model. Results show that the method posed in this paper is a proof-of-concept of context-aware robotic localization and navigation under the free energy principle.

1 Introduction

Localization and navigation are intuitive processes for humans. As infants, we are continuously presented with new knowledge regarding our state in the world, the sensory observations associated with these states and how we can perform actions that change the environment around us [3]. As we learn, a worldview is formed that informs us about the relation between our sensory input and our position in the world. This learning process is very adaptable in the sense that we are able to quickly update our worldview when presented with new knowledge. In contrast, robots generally operate in a limited environment and behave according to a predetermined model. When presented with an observation not present in this model, the robot underperforms or fails completely. Classically, this made robotic navigation a difficult task in dynamic, unpredictable environments such as a real world setting [17].

In this paper, we present an algorithm that enables a robotic agent to adaptively perform self-localization and navigation in an environment. This is achieved by the minimization of free energy under the free energy principle [6]. Free energy is the prediction error, or surprise, between predicted sensory observations and actual input. These predictions are obtained through a generative model that maps position to sensory input for an environment. Minimizing free energy entails that an agent minimizes this experienced surprise. An agent achieves this by simultaneously updating its internal belief of its current state (perceptual inference) and performing actions that change its current state (active inference). Both updating the belief and performing actions are done in a way that minimizes the difference between the predicted and actual observation. The approach of using raw visual input for state-estimation and navigation is inspired by the PixelAI algorithm [19] and attempts to move forward on the simulated robotic navigation approach posed in [2].

In recent works, pose estimation in the from of a regression approach has risen significantly in popularity [16, 13, 12]. These end-to-end methods provide a intuitive approach to localization by training a neural network in pose-observation mapping and obtaining pose estimations by a forward pass through the network. However, these approaches have also proven to be less accurate than structure-based methods [20]. This gives another incentive to approach localization problems from another angle than regression, which is in turn an added motivational factor to this thesis.

The generative model used for predicting observations is learned for a certain environment. However, possible surroundings in which humans operate differ greatly and

therefore ask for different generative models associated with them. For instance: when standing in our bedroom, we can accurately predict what we will see based on where we are standing. However, this mapping of location to observation for our bedroom serves us little to no purpose when we want to predict observations based on our location in a park. This means that our bedroom mapping differs from a park mapping and therefore the two types of environments require different generative models for performing inference. By adding the ability to switch between generative models, our agent can change its worldview according to the environment in which it is present, making it adaptable to a wider range of environments and thus more flexible as a whole. Using the definitions for 'context' and 'context-aware' posed by *Abowd et al.* [1], we can formulate that this switching is based on the context of the environment, thus making the system context-aware. The context-aware switching performed by the robot is conceptually based on [9].

Our suggested approach was evaluated by implementing the algorithm in a simulated robot provided by the Robotic Operating System framework (ROS) [18]. The robot operates in Gazebo [15], a simulator for robotic operation in self-made environments.

2 Methods

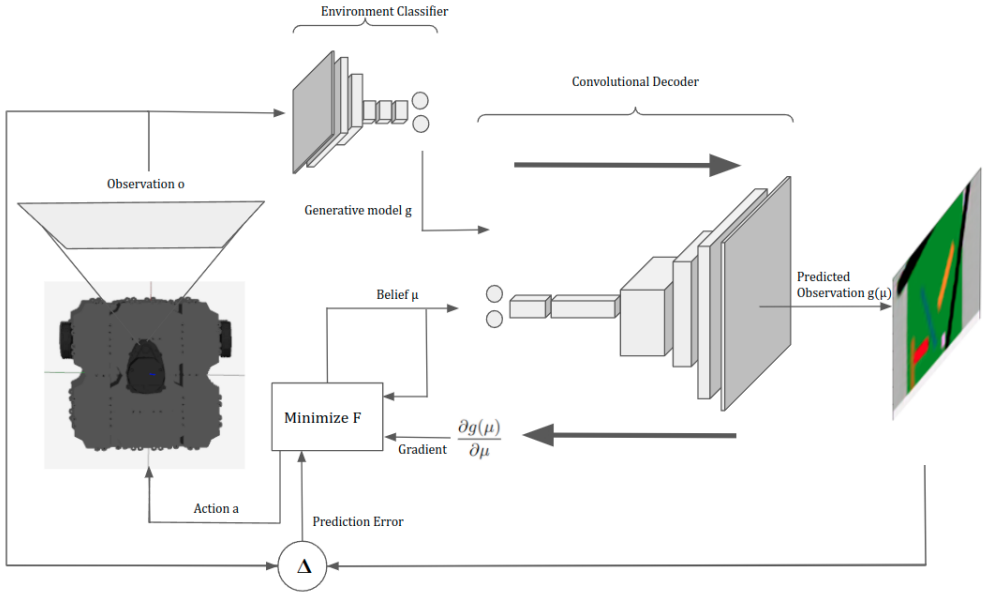

Figure 1: Proposed algorithm for robotic perception and action under the free energy principle. By minimizing the free energy F, the robot can update its internal state belief and perform actions in order to reach a desired goal state.

2.1 The free energy principle

Performing perceptual inference can be explained as inferring an agent's believed state x from sensory observations o. To obtain this posterior $p(x|o)$, we can use Bayes' formula:

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

.

Computing the posterior $p(x|o)$ exactly requires the marginal likelihood $p(o)$, which is obtained by integration over all states. For large state spaces, this integration becomes intractable which makes the exact posterior computation unsolvable. The free energy principle gives a solution to this problem by using a tractable distribution $q(x)$ that approximates the posterior $p(x|o)$. This approximation is obtained by minimizing the the free energy F:

$$F = D_{KL}(q(x) \,||\, p(x|o)) - ln\ p(o) \tag{2}$$

which is formulated as the Kullback-Leibler divergence between the approximated and real distributions minus the surprise $ln\ p(o)$. F is an upper bound on surprise, minimization of F can therefore be achieved by minimizing the surprise experienced by the agent. Surprise can be minimized by an agent changing its belief over its current state (perceptual inference) and performing actions that change its current state (active inference).

2.2 Perceptual inference

Inferring an agent's state x from sensory observations o is formalized as changing the latent state belief μ in a way that minimizes free energy. Under the Laplace approximation [7], this is achieved by gradient descent on the prediction error, multiplied by the inverse variance Σ_o^{-1}:

$$\dot{\mu} = \frac{\partial g(\mu)}{\partial \mu}\Sigma_o^{-1}(o - g(\mu)) \tag{3}$$

where $\frac{\partial g(\mu)}{\partial \mu}$ is the gradient with respect to the latent state and $(o - g(\mu))$ is the prediction error, or surprise, between the real and expected sensory input. The expected sensory input is obtained by a forward pass through a generative model in the form of a convolutional decoder that maps latent states to observations. The update rule for the state belief is defined as follows:

$$\mu_{t+1} = \mu_t + \Delta_t \dot{\mu} \tag{4}$$

where Δ_t is the step size of one iteration.

2.3 Active inference

In order to perform navigational tasks, the agent can perform actions that bring it closer towards a desired state. This desired state is presented as the desired sensory observation o_{goal}. This sensory observation acts as an attractor to the agent by creating an error between the current observation and the desired observation. With inclusion of this attractor-error, state inference is computed as follows:

$$\dot{\mu} = \frac{\partial g(\mu)}{\partial \mu}\Sigma_o^{-1}(o - g(\mu)) + \frac{\partial g(\mu)}{\partial \mu}\Sigma_\mu^{-1}\beta(o_{goal} - g(\mu)) \tag{5}$$

where β is a scalar that weighs the attractor part of the equation.

In trying to minimize this error, the agent will perform actions in order to change its observation to match that of the attractor observation. This results in the agent navigating towards this desired state. Choosing actions to perform is computed analogously to state inference:

$$\dot{a} = -\frac{\partial g(\mu)}{\partial \mu}\Sigma_o^{-1}(o - g(\mu)) \tag{6}$$

.

The update rule for action is formalized similarly to Eq.(4):

$$a_{t+1} = a_t + \Delta_t \dot{a} \tag{7}$$

.

2.4 Environment classification

The predicted observations used in Equations (3), (5) and (6) are obtained by a forward pass of μ through a generative model that maps a state to a sensory observation $g(\mu)$ for a specific environment. In doing so, the agent is restrained by this generative model and the environment which it represents. By adding the ability to switch between generative models, the agent can function in a wider range of environments.

We define the variable g as the generative model used for predicting observations. At time t, g is defined as follows:

$$g_t = mode([e_{t-n}, e_t]) \tag{8}$$

where e_t is the mapping of observation o_t to an environment class by means of a classifier:

$$e_t = c(o_t) \tag{9}$$

.

n is a hyperparameter that defines the memory of an agent. Changing n denotes the trade-off between how quickly the agent switches its generative model upon being presented with a new environment class and the degree of certainty it has in using the current generative model.

2.5 Applying deep learning

For performing perceptual and active inference, generative models are used to predict sensory observations. Moreover, a classifier is used to determine which generative model to use based on the environment in which an agent is operating. Both the generative model and the classifier are realized as deep neural networks that map state to observation and observation to environment class respectively.

2.5.1 Generative model

As generative model a Convolutional Neural Network (CNN) is used that takes an input of size 2, representing the x and y coordinate of the robot. This input goes through a sequence of 2 Fully Connected (FC) layers before being up-sampled by 4 Transposed Convolution (TConv) layers. Between the FC layers and TConv layers is a reshape function that changes the shape of the neurons to an input suitable for the TConv layers. The output of the model is a 3x80x80 tensor representing an 80x80 rgb image with pixel intensity values between 0 and 1. All TConv layers except the last are followed by a normal Convolution (Conv) layer to help in smoothing out checkerboard-artifacts that can occur as the result of the Transposed Convolution. A Dropout layer is added to prevent overfitting and all layers use a Rectified Linear Unit (ReLU) activation function to apply non-linearity.

2.5.2 Classifier

The classifier used for environment classification is a CNN that takes a 3x80x80 tensor representing an image as input. This input is passed through three Conv layers, each followed by a Max-Pooling (MP) layer for feature extraction from the input image. Following these Conv layers are three FC layers that ultimately give an output equal to the amount of environments for which the classifier is trained, which in the case of our model is two. By applying a Softmax function over this output, the most probable environment is classified. All layers except the last FC layer use a ReLU activation function.

2.6 Robotic experiment

The algorithm is implemented in a robot from the ROS framework which operates in the Gazebo simulator. The algorithm is tested by letting the robot perform localization and navigation tasks. The outcomes of these tasks are then evaluated by inspecting how the prediction error of the model changes throughout the run of the algorithm. For localization this consists of determining if the robot can succeed in matching its state belief to its real state whereas for navigation we inspect how the robot performs actions to minimize its surprise in order to reach a certain goal state.

By evaluating the prediction error of the robot throughout runs of the algorithm, we can determine if the robot succeeds in minimizing this prediction error and thus minimizes free energy. If the prediction error decreases throughout a run it can be argued that the robot succeeds in performing perceptual or active inference. The degree as to which the robot succeeds in its localization and navigation tasks depends on the value to which the prediction error converges and the speed of convergence. Perfect convergence would result in a final prediction error of 0, meaning that the robot has set its belief state equal to its true state when performing perceptual inference, or reaching its desired goal state in the context of active inference. In our results, we try to give meaningful insight into how our algorithm performs in relation to the above-mentioned criteria.

The implementation source code can be found in the online GitHub repository of this project[1].

2.6.1 Setup

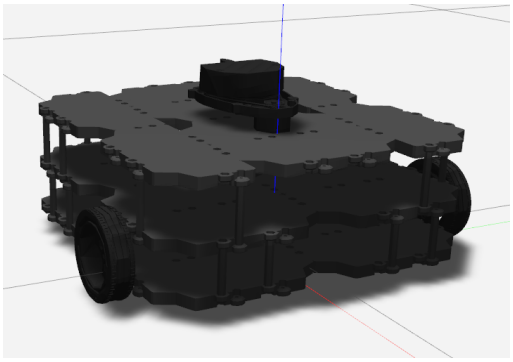

Figure 2: Simulated TurtleBot3 Waffle Pi robot

The ROS robot used for the experiments is the TurtleBot3 Waffle Pi. The robot is equipped with a Raspberry Pi camera for image capturing. The robot can move by means of two velocity controlled motors. However, for this experiment we will move the robot by instantly changing its location instead of changing motor velocities. This is done for simplification purposes as it removes handling movement dynamics.

The environments used for localization, navigation and environment classification tasks are each made up of a 8x8 meter grid on which the robot can move around. The robot's orientation is fixed towards a wall on the edge of the grid containing shapes that give a point of reference to the robot in order to perform inference. For this experiment, two environments are designed and used used which seen in Figures 3 and 4 respectively.

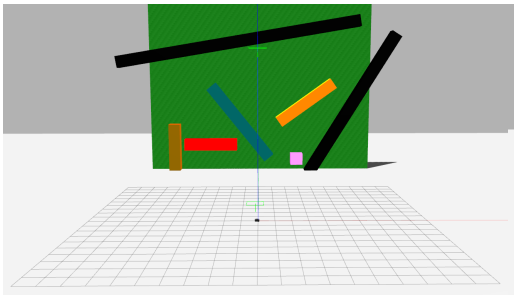

Figure 3: Squares environment

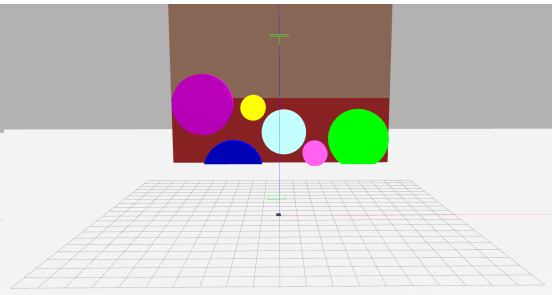

Figure 4: Circles environment

2.6.2 Data sampling

For training the Convolutional Decoders used for the agents' generative models 7000 samples are obtained per environment in a uniformly random manner. Each sample consists of a resized 80x80 rgb image captured by the robot's camera and the associated coordinate vector: $[x, y]$. The dataset used for training the classifier consists of the combined samples from each environment, which is 14000 for our experiment.

[1] https://github.com/SimonWulp/ActiveInferenceRobotics

2.6.3 Model training

For each environment a Convolutional Decoder is trained on the environments' respective dataset in batches of 200 samples. The training was repeated for 200 epochs and made use of an Adam Optimizer [14] with a learning rate of 0.001 and a learning rate-scheduler with step size 20 and gamma 0.95. The loss function used in training was the Mean Squared Error (squared L2 norm) loss.

The Classifier was trained on a dataset consisting of the combined samples from each environment. The training was done in batches of 100 samples using a Stochastic Gradient Descent optimizer with learning rate 0.001 and momentum 0.9.

3 Results

3.1 Sensory predictions

Figure 5 shows two visual observations for each environment (top row) and the associated predicted visual observation that is obtained by a forward pass of the respective state through the CNN belonging to that environment (bottom row).

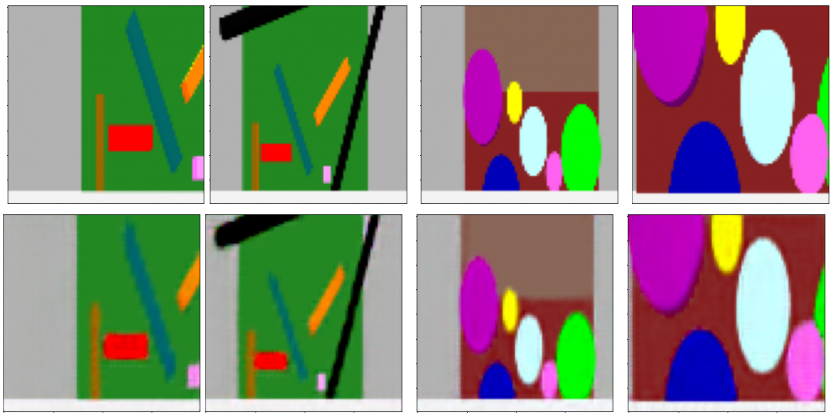

Figure 5: Real observations (top row) and their respective predictions made by the Convolutional Decoder (bottom row)

3.2 Localization

We evaluate localization performance of the model by assigning a uniform random belief to the robot within ranges x: $[-lim, lim]$ and y: $[-lim, lim]$. The sensory observation of the robot is an image obtained from a random location in the environment, with the true state being the state associated to this observation. The algorithm is run for 100 steps, at each step updating its belief to minimize surprise. Figure 6 shows the mean and standard deviation of the prediction error as distance in meters at each step as a result of 100 full runs of the algorithm. The plot for lim=4 shows that initializing the robot with a belief that is more central in the environment results in better approximation of the true state. Whereas lim=8 shows that the agent generally ends up with a belief further away from the true state since this higher limit creates the possibility of generating a initial belief and true state that are further apart from each other, making it harder for the algorithm to properly converge.

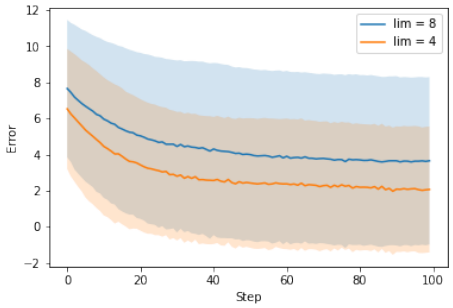

Figure 6: Prediction error per step with mean and 1 standard deviation over 100 algorithm runs. Results are shown for runs with starting positions uniformly random between ranges x: $[-lim, lim]$ and y: $[-lim, lim]$.

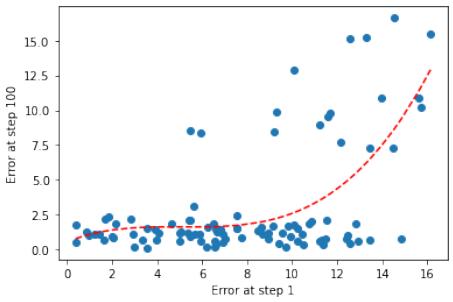

Figure 7: Prediction error as distance in meters at start of algorithm run (x-axis) against end of run (y-axis). The trend line shows that convergence to true state becomes harder when begin state and true state are further apart. The initial state belief is sampled with lim=8.

Figure 7 shows the prediction error as distance in meters at step 1 in algorithm plotted against the prediction error at step 100. The plot shows that if the distance from the initial belief to the true state is low, the agent is more likely to converge to a belief close to the true state after 100 steps. On the contrary, when the distance between the initial belief and the true state is larger, the agent has more difficulty in converging a belief state close to the true state.

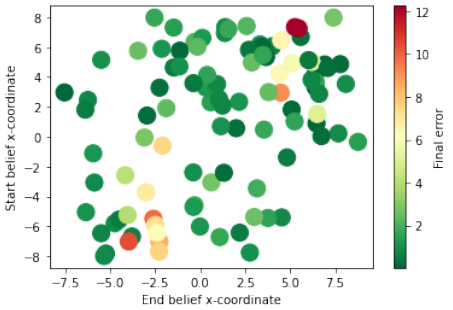

Figure 8: Start belief state x-coordinates of the algorithm against end belief state x-coordinates, the colormap displays the error as distance in meters between the end belief state x-coordinates state and the goal state x-coordinates

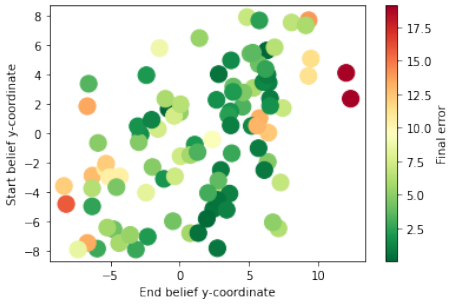

Figure 9: Start belief state y-coordinates of the algorithm against end belief state y-coordinates, the colormap displays the error as distance in meters between the end belief state y-coordinates and the goal state y-coordinates

Figures 8 and 9 show the start state belief of each of the 100 executed runs on the y-axis and the final state belief after 100 steps of each run on the x-axis for the x and y coordinates of the beliefs respectively. The colormap shows the error as distance in meters between the final state belief and the true state of the agent (i.e. the goal state to which the belief state would ideally converge).

3.3 Navigation

Navigation is evaluated by testing the robot's performance in moving towards a goal. This goal consists of a state in the environment and the observation associated to this state. This desired observation is used by the model as an attractor image, as posed in Eq. (5). The attractor image is obtained as a random sample from the environment. We evaluate the navigational performance in a similar manner to localization. The starting position of the robot is a randomly sampled location within ranges x: $[-lim, lim]$ and y: $[-lim, lim]$. The initial belief of the agent is set to this starting state. A full run of the algorithm consists of 100 steps, at each step the free energy is minimized by updating the internal belief and performing an action that minimizes the prediction error.

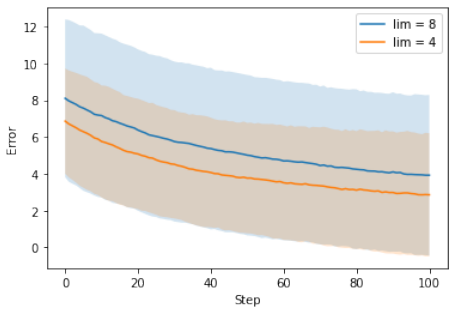

Figure 10: Prediction error per step with mean and 1 standard deviation over 50 algorithm runs. Results are shown for runs with starting positions uniformly random between ranges x: $[-lim, lim]$ and y: $[-lim, lim]$

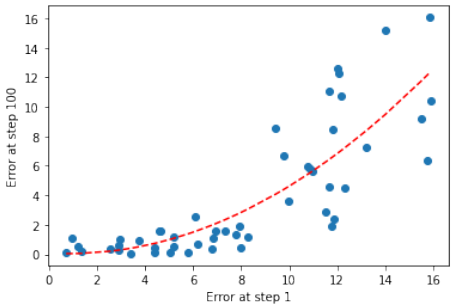

Figure 11: Prediction error as distance in meters at start of algorithm run (x-axis) against end of run (y-axis). The trend line shows that convergence to true state becomes harder when begin state and true state are further apart. The initial state is sampled with lim=8.

Similar plots as used for localization can also be used for evaluating navigation. Figure 11 shows the mean and standard deviation of the prediction error as distance in meters at each step as a result of 50 full runs of the algorithm. By looking at the different plots for lim=8 and lim=4, we can determine that a starting position closer to the center of the environment causes for generally better navigational performance to the goal state. Figure 11 shows that navigating to goal states that are further away from the start state than 8 meters is challenging for the agent.

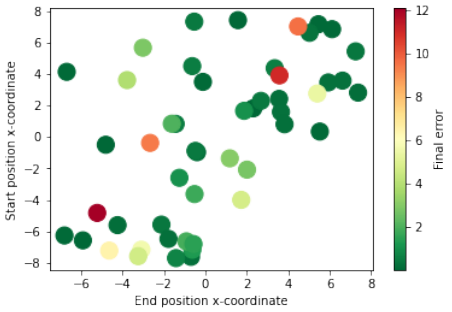

Figure 12: Start location x-coordinates of the algorithm against end location x-coordinates. The colormap displays the error as distance in meters between the end location x-coordinates state and the goal location x-coordinates.

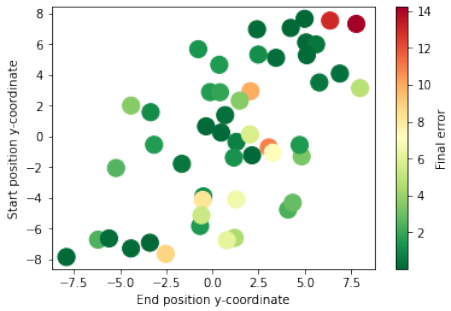

Figure 13: Start location y-coordinates of the algorithm against end location y-coordinates. The colormap displays the error as distance in meters between the end location y-coordinates and the goal location y-coordinates.

Figures 12 and 13 show the start state of each of the 50 executed runs on the y-axis and the final state belief after 100 steps of each run on the x-axis for the x and y coordinates of the locations respectively. The colormap shows the error as distance in meters between the final state and the true state of the agent (i.e. the goal state that the robot would ideally move towards).

3.4 Environment classification

The classifier accuracy is obtained by predicting environment classes on a test set of 6000 datasamples. These samples come from the same environments on which the model is trained but are kept separate from the training set. By comparing the predicted environment class for each observation to its ground truth, the accuracy of the classifier can be computed. The obtained accuracy is 100%. This high number can be explained by the fact that both environments are quite simple and distinctive and therefore easy to differentiate by the classifier.

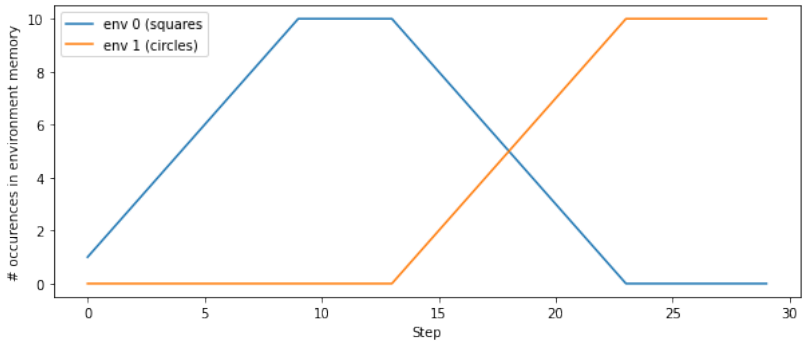

Figure 14: Change in ratio of classes represented in environment memory

Figure 14 shows the environment memory of a run of 30 steps where each of the two shape-filled walls from the shapes and circles environments were both present. At the start, the robot is oriented towards the wall representing environment 0 and will therefore

use the generative model associated to this environment. After step 14, the robot's camera orientation was changed from looking at the wall representing environment 0, to the wall representing environment 1. Following from this, the ratio of env 0 and env 1 represented in the environment memory of the agent also changes because the classifier labels the new observations by the robot as belonging to env 1. After step 18, the env 1 class ratio becomes greater than the env 0 class ratio. This makes the agent switch the generative model that it was using from the one associated with environment 0 to the one associated with environment 1.

4 Discussion

This work has shown that robotic state estimation and navigation can be achieved through the minimization of variational free energy. Generative models in the form of Convolutional Decoders provided a meaningful state-observation mapping used for computing the prediction error experienced by the model. It should be mentioned that the environments that the graphical models represent are very simple in their complexity. As a consequence, the Convolutional Decoder used for this work is also not comprised of a very elaborate architecture. For further work, it would be interesting to discover how more complex Decoder architectures, such as those used in Varitaional AutoEncoders [22], can offer support for more complex environments. Such architectures could furthermore contribute to a more complex state representation. The inclusion of rotation, which is not present in this work, would be a natural first choice.

Our algorithm handles navigation as performing an action that minimizes free energy ad-hoc. With the use of expected free energy [8], the agent can perform navigational tasks in a more planned out manner by minimizing the free energy of a policy of actions, rather that just a single action. In doing so, a more forward thinking approach is suggested that could contribute to more meaningful robotic behaviour in terms of navigation.

We have build upon the active inference framework by adding context-awareness in the form of environment classification. This adds flexibility to the model by increasing the range of environments in which it can successfully operate. Although a general proof-of-concept is shown, there is definitely room for further exploration of this premise. By adding more environments to the equation which are furthermore of an increased complexity, a more meaningful form of context-awareness can be achieved. This asks for an increase in complexity of the classifier or maybe even a different overall approach to the classification process. A lot of insightful work exists in which approaches are given for defining contextual information based on objects [11], semantic cues [4, 5] or other contextually informative elements in an environment [10, 21]. [23] suggests an hierarchical approach to generative modelling, a premise that could also be applied to the concept of context-awareness.

Approaches to obtaining additional information from an agents' contextual surrounding serve as an interesting addition to the concept of variational free energy minimization. In doing so, an agent can create a better understanding of its surroundings and perform more meaningful inference on it.

References

[1] Gregory D. Abowd, Anind K. Dey, Peter J. Brown, Nigel Davies, Mark T. Smith, and Pete Steggles. Towards a better understanding of context and context-awareness. In *HUC*, 1999.

[2] Daniel Burghardt and Pablo Lanillos. Robot localization and navigation through predictive processing using lidar. 09 2021.

[3] Fabian Chersi and Neil Burgess. The cognitive architecture of spatial navigation: Hippocampal and striatal contributions. *Neuron*, 88(1):64–77, 2015.

[4] Akansel Cosgun and Henrik I. Christensen. Context-aware robot navigation using interactively built semantic maps. *Paladyn, Journal of Behavioral Robotics*, 9(1):254–276, 2018.

[5] Jonathan Crespo, Clara Gómez, Alejandra Hernández, and Ramón Barber. A semantic labeling of the environment based on what people do. *Sensors*, 17(2), 2017.

[6] Karl Friston. Friston, k.j.: The free-energy principle: a unified brain theory? nat. rev. neurosci. 11, 127-138. *Nature reviews. Neuroscience*, 11:127–38, 02 2010.

[7] Karl Friston, Jérémie Mattout, Nelson Trujillo-Barreto, John Ashburner, and Will Penny. Variational free energy and the laplace approximation. *NeuroImage*, 34(1):220–234, January 2007.

[8] Karl Friston, Francesco Rigoli, Dimitri Ognibene, Christoph Mathys, Thomas FitzGerald, and Giovanni Pezzulo. Active inference and epistemic value. *Cognitive neuroscience*, 02 2015.

[9] Raul Guilherme, Francisco Marques, André Lourenço, Ricardo Mendonça, Pedro Santana, and José Barata. Context-aware switching between localisation methods for robust robot navigation: A self-supervised learning approach. In *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 004356–004361, 2016.

[10] Marcel Häselich, Marc Arends, Nicolai Wojke, Frank Neuhaus, and Dietrich Paulus. Probabilistic terrain classification in unstructured environments. *Robotics and Autonomous Systems*, 61(10):1051–1059, 2013. Selected Papers from the 5th European Conference on Mobile Robots (ECMR 2011).

[11] Georgios Kapidis, Ronald Poppe, Elsbeth van Dam, Lucas Noldus, and Remco Veltkamp. *Object Detection-Based Location and Activity Classification from Egocentric Videos: A Systematic Analysis*. 04 2019.

[12] Alex Kendall and Roberto Cipolla. Geometric loss functions for camera pose regression with deep learning. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6555–6564, 2017.

[13] Alex Kendall, Matthew Grimes, and Roberto Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. pages 2938–2946, 12 2015.

[14] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[15] N. Koenig and A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent*

Robots and Systems (IROS) (IEEE Cat. No.04CH37566), volume 3, pages 2149–2154 vol.3, 2004.

[16] Iaroslav Melekhov, Juho Kannala, and Esa Rahtu. Relative camera pose estimation using convolutional neural networks. *CoRR*, abs/1702.01381, 2017.

[17] Ulrich Nehmzow and Carl Owen. Robot navigation in the real world:: Experiments with manchester's fortytwo in unmodified, large environments. *Robotics and Autonomous Systems*, 33(4):223–242, 2000.

[18] Morgan Quigley, Brian Gerkey, Ken Conley, Josh Faust, Tully Foote, Jeremy Leibs, Eric Berger, Rob Wheeler, and Andrew Ng. Ros: an open-source robot operating system. In *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA) Workshop on Open Source Robotics*, Kobe, Japan, 2009.

[19] Cansu Sancaktar, Marcel Gerven, and Pablo Lanillos. End-to-end pixel-based deep active inference for body perception and action. 12 2019.

[20] Torsten Sattler, Qunjie Zhou, Marc Pollefeys, and Laura Leal-Taixé. Understanding the limitations of cnn-based absolute camera pose regression. pages 3297–3307, 06 2019.

[21] Erke Shang, Xiangjing An, Tao Wu, Tingbo hu, Qiping Yuan, and Hangen He. Lidar based negative obstacle detection for field autonomous land vehicles. *Journal of Field Robotics*, 33, 06 2015.

[22] Arash Vahdat and Jan Kautz. Nvae: A deep hierarchical variational autoencoder, 2021.

[23] Ozan Çatal, Tim Verbelen, Toon Van de Maele, Bart Dhoedt, and Adam Safron. Robot navigation as hierarchical active inference. *Neural Networks*, 142:192–204, 2021.

Appendix

A Model parameters

In testing, the following parameters were used in Equations (3), (4), (5), (6), (7) and (8):

$$\Sigma_o = 5000$$
$$\Delta_t = 0.02$$
$$\Sigma_\mu = 1000$$
$$\beta = 1$$
$$n = 10$$

.

B Neural network architectures

Figure 15 and figure 16 show the network architectures of the Convolutional Decoder and classifier respectively.

Operation	Input	Output	Activation	Kernel Size	Stride	Padding
FC	2	1024	ReLU	-	-	-
FC	1024	128*5*5	ReLU	-	-	-
Reshape	128*5*5	128x5x5	-	-	-	-
TConv	128x5x5	64x10x10	ReLU	4x4	2	1
Conv	64x10x10	64x10x10	ReLU	3x3	1	1
TConv	64x10x10	32x20x20	ReLU	4x4	2	1
Conv	32x20x20	32x20x20	ReLU	3x3	1	1
TConv	32x20x20	16x40x40	ReLU	4x4	2	1
Conv	16x40x40	16x40x40	ReLU	3x3	1	1
Dropout	16x40x40	16x40x40	-	-	-	-
TConv	16x40x40	3x80x80	ReLU	4x4	2	1

Figure 15: Architecture of the Convolutional Decoder representing a generative model

Operation	Input	Output	Activation	Kernel Size	Stride	Padding
Conv	3x80x80	6x76x76	ReLU	5x5	1	0
MP	6x76x76	6x38x38	-	2x2	2	0
Conv	6x38x38	12x34x34	ReLU	5x5	1	0
MP	12x34x34	12x17x17	-	2x2	2	0
Conv	12x17x17	18x13x13	ReLU	5x5	1	0
MP	18x13x13	18x6x6	-	2x2	2	0
Reshape	18x6x6	18*6*6	-	-	-	-
FC	18*6*6	128	ReLU	-	-	-
FC	128	64	ReLU	-	-	-
FC	64	2	ReLU	-	-	-

Figure 16: Architecture of the Convolutional Classifier used for environment classification