# Policy Blending for Deep Reinforcement Learning via Facial Gesture Recognition

Jonas Tjomsland
Department of Computer Science, University of Cambridge
Mini-Project in Affective Computing

***Abstract*— In this work we implement and test a Deep Reinforcement Learning (DRL) agent capable of determining whether a human UAV-operator is in control, by analysing the operator's facial gestures. Building on the term of policy blending, the DRL-agent is trained to change the mode between autonomous and human control, solely by looking facial gestures. Our results show that the agent is able to learn this, ultimately allowing an autonomous agent to take over when the human operator looses track of the UAV, and letting the human operator have control when no disturbance is present.**

## I. Introduction

Learning to recognise faces is something we as humans do from a young age and substantial amount of information is conveyed through facial expression during human interactions [1]. Giving machines the same ability to detect and recognise emotions on the same level as humans is a well researched field, but there are still unexplored avenues with respect to how that ability can be leveraged in human-robot interaction and shared control.

The term shared autonomy is often used for control scenarios where a human operator, together with an artificial agent, control a system. The idea is that the the agent will assist the human in accomplishing the task, either by stepping in when the human makes mistakes or slightly altering the human's commands if they are sub-optimal. What is often difficult to define in these systems is the level of intervention from the artificial agent, i.e. when should it assist and when should it allow the human full control. This problem, often labeled policy blending, is well researched for applications like teleoperation [2]. However, there is still no trivial solution to how one should balance shared human and artificial agent control.

A situation where such a challenge may occur is a shared-control drone flying scenario. If it is a sunny day with good visual sight of the drone from the operators perspective, we would like to give the drone operator full control, as we assume they can solve the task optimally. However, given a sudden change in weather, fog might appear and the operator loose sight of the drone. When this happens the artificial agent would ideally intervene and take over control. It is not obvious to a drone that the operator is not in control, however, it is likely that the weather transition, i.e. the change of how in control the human operator feels, can be reflected in their facial expression. This work aims to leverage that information by giving a Deep Reinforcement Learning agent access to the human operator's emotions through their facial expression. The main contribution of the work is a proof-of-concept implementation showing that with only minutes of real-world training with an operator, such an agent can learn to differ the facial expressions that indicates an operator in control from the ones of someone not in control and directly use that information for policy blending.

## II. Related Work

Determining the level of autonomy in a shared control context is a well researched field and maybe the most used paradigm is the use of pre-defined discrete levels autonomy. Previous work has introduced terms as adjustable autonomy [3] and sliding autonomy [4], where robots operate autonomously most of the time and asks humans to intervene when necessary. This approach works well when the task at hand can be solved optimally by an autopilot. However, this is not always the case and in some situations a human operator is needed for optimal performance. Our work does therefore take the opposite approach, allowing human control most of the time and encouraging an autonomous agent to intervene when needed.

To determine when this intervention is needed we leverage the facial expressions of the human operator. This is not a novel approach as there are examples of work where facial expression recognition (FER) has been used for driver drowsiness detection [5] or fatigue recognition [6]. Although this has been explored previously, most of the work has been focused on the supervised learning part, i.e. the detection and recognition, and less work has been done on how to effectively use that information in an end-to-end solution like the one proposed in this work.

Combining robot learning with human facial expressions has also been explored in previous work. Within the Reinforcement Learning (RL) framework, this has often been done by using the facial expressions of a human interactor to reward the agent and determine the value of different actions [7], [8]. Our work is different in that the agent does not have access to the state of the environment, but have to infer that from the human operator's face and is rewarded based on the human's performance. This differs from the other way around where it has access to the environment's state and gets rewarded based on the human feedback through facial gestures.

When training with humans in the loop from scratch, without any pretraining, high demands must be set to the sample-efficiency of the training process. It is not feasible for a

human subject to conduct the thousands of interaction steps that RL methods traditionally requires. Nonetheless, there are methods that tackle this challenge and allows for sample efficient training. One example is the Probabilistic Inference for Learning Control (PILCO) method [9], developed by Deisenroth and Rasmussen. PILCO leverages Gaussian Processes to build dynamic models without out needing vast amount of data. With only seconds of interaction with a physical control system, PILCO can learn to optimise control. However, this comes with the drawback of significant computation time and would not be feasible for the real-time system this work implements. More promising is the SAC algorithm proposed by Haarnoja et al. Running with a continuous action and state space, their SAC implementation with automatic entropy tuning was used to teach a four-legged robot to walk without any training in simulation. The authors reported that no need for substantial parameter tuning was critical to the sample-efficient performance. Building on this work, Shafti et al. showed that the low number of samples needed to learn fascilitated for human-in-the-loop training [10].

## III. Method

In this section, the several steps needed to implement the proposed system will be explained. Overall, the complete system consist of a facial gesture recognition model that gives the state of the first DRL-agent which again outputs and action that determines whether the human operator or the second DRL-agent should be allowed control over the UAV. The second DRL-agent is referred to below as the autonomous agent, the main contribution of this work lays in the construction and implementation of the first DRL-agent, referred to as the facial expression agent. See the figure below for an overview of the system.

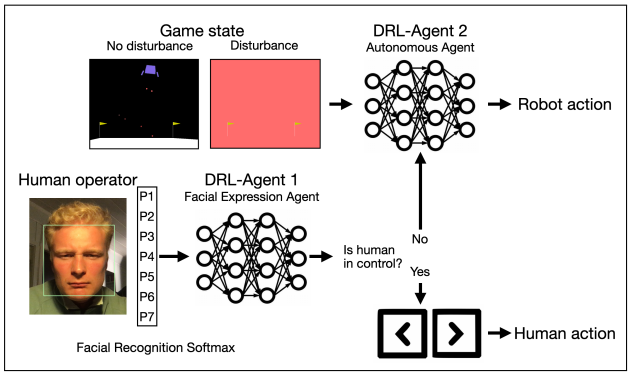


Fig. 1. Overview over the complete system and the involved agents.

Briefly explained, the facial expression agent is only given the softmax outputs of a facial expression recognition model and uses this to decide whether to use the human operator's or the autonomous agent's action. The facial expression agent is not given access to the game's state, only rewarded based on the performance in the game.

### A. Reinforcement Learning Preliminaries

In the field of Reinforcement Learning (RL), an artificial agent operates in a simulated or real environment, executing actions and receiving rewards based on the outcomes of those actions. By exploring the environment, the agent develops a policy that maps the state of the system to specific actions, with the intention of maximizing the long-term cumulative reward [11]. The term Deep Reinforcement Learning comes from the use of Deep Neural Net as function approximators between states and actions or states and the values of states. In our system, we consider both the autonomous agent as well as the facial expression agent as Markov Decision Processes (MDPs). At every time step they are defined as a tuple, $(S, A, P_{sa}, \gamma, R)$, where $S$ and $A$ represents the state and action respectively, $P_{sa}$ the transition probability, $\gamma$ the discount factor and $R$ the reward. When exploring the environment, state-action-reward transitions are generated by a policy, $\pi(s, a)$, that tries to maximize its objective.

### B. Environment and Autonomous DRL-Agent

The implemented DRL-agents operate in the OpenAI Gym's Lunar Lander environment [12]. In the environment the objective is to safely land the UAV between two flags without unnecessary fuel consumption. The traditional reward function penalises distance from goal position, translational and rotational velocity, fuel consumption as well as potential crashes. The agent is rewarded for a successful landing. See Fig. 3 for a screenshot of the environment.

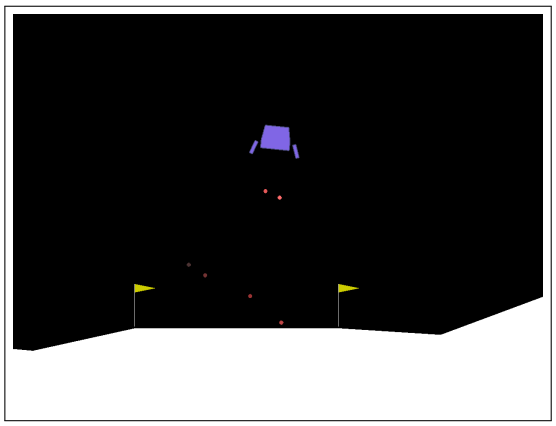

Fig. 2. Screenshot of the OpenAI Gym's Lunar Lander environment.

This environment sets the arena for our experiments and both the autonomous agent that takes over control when needed as well as the agent that determines when that is the case, are trained here. To facilitate for our policy blending experiments, we start by training the agent for autonomous control. It is important that it performs on a reasonable level, i.e. does not crash, but at the same time it should not be as optimal as a human operator in good conditions. Therefore, the traditional reward function for the Lunar Lander environment is slightly modified to encourage hovering instead of landing. By doing so, when the trained autonomous agent takes over, the lander will hover in the air staying roughly in the same spot. This means it will receive a negative reward for fuel consumption and distance to goal at every time step, therefore performing worse than a human operator that can land the craft. However, it will

be better than a operator that crashes because they lost track of the lander. The hovering behaviour was easily obtained by implementing the Advantage Actor Critic A2C method [13] from Stable Baselines [14]. With around 20 minutes of training on a CPU the autonomous agent was able to hover at roughly the same position for as long as necessary.

### C. Facial Gesture Recognition

The next step is to extract the facial gestures of the human operator. The initial idea was to use the well-known OpenFace software [15], to extract Facial Action Units (FAU) activation and use these as a direct input to the second DRL-agent's state. However, OpenFace runs in Python 2 while OpenAI's Gym requires Python 3. This challenge led to the use of a different approach and a different Open Source solution. Since no other off-the-shelf software for FAU recognition was available, one of the many pretrained emotion recognition models out there was used, with the assumption that they were trained to make emotion predictions based on changes in facial expressions. Pham et al. [16] implemented such a model called Residual Masking Network (RMN) with a reported accuracy of 76.82% on the FER2013 dataset. After some testing, their pretrained model was deemed to perform reasonably well and chosen for this project. Like other similar models, the RMN model outputs softmax values for 7 emotion classes; angry, disgust, fear, neutral, happy, sad and surprise. This output vector alone, represents the state of the first DRL-agent, allowing it to determine whether the human operator is in control.

Following this, a simple script was written using the OpenCV module in Python. At every time step, a frame is extracted from the webcam, edited for prediction and then given to the pretrained RMN model. Inference took approximately 0.3s per image and since the time step size in the actual Lunar Lander environment was shorter than this the facial expression model was kept in a separate script, writing its predictions to a file that could be accessed by the main script.

### D. Sample-efficient learning system — DRL-Agent 1

For the training of the facial expression agent the reward function was restored to its original form where landing between the flags is rewarded as well. Because of the slight delay that results from inference in the RMN model, the time step size of the facial expression agent has to be larger than the one in the running environment. The perception-action loop can be described as follows. For every 50 time step in the environment (Roughly 5s), the softmax output from the RMN-model is read and given as the state to the facial expression agent which then again outputs an action determining whether to use the human or the autonomous agent's commands. The outcome is then allowed to run for the next 50 time steps, i.e. the human might be controlling the UAV for that period, before a new evaluation is performed again. In other words, at approximately every 5 second the agent evaluates the facial expressions of the human operator and uses that information to decide whether he or she is in control or if the autonomous agent must intervene.

However, this implementation leads to another challenge, sparse amount of data. Since a new state-action transition is only obtained every 5 second and the system is trained with a human in the loop, high requirements must be set to the sample-efficiency of the implementation. Because of this, the Soft Actor Critic (SAC) algorithm with automatic entropy tuning is used [17]. The implemented models are based on work from the FiredUp team [18], modified for this project's purposes. SAC is an off-policy algorithm, a trait that is crucial for sample-efficiency since it allows for offline updates of the models knowledge asynchronously to the action-perception loop. The "Soft" part relates to how the method not only learns to maximize its long-term cumulative reward, but also at the same time maximizing the entropy of the policy. This leads to a more stochastic learning process which has been shown to increase performance with few data points as well as resulting in a more robust behaviour [19]. This stochasticity encourages the agent to not only exploit its current ideas about what the best action is, but to explore different options, ultimately increasing the probability of finding the optimal solution. The entropy term is included in the facial expression agent's reward function as follows,

$$J(\pi) = \sum_{t=0}^{T} {}_{\tau\sim p_\pi}[r(\boldsymbol{s_t}, \boldsymbol{a_t}) - \underbrace{\alpha_t log \pi_t(\boldsymbol{s_t}|\boldsymbol{a_t})}_{\text{Entropy term}}], \quad (1)$$

where the first term is received based on the performance in the Lunar Lander environment and the second term relates to the entropy of the policy. To balance between the impact of the two terms, the $\alpha$ parameter is used, often refereed to as the temperature parameter. $alpha$ determines the relationship between rewarding exploration or exploitation and acts as an important hyperparameter. Ideally, the same entropy should be kept throughout learning, but this is difficult due to the change in reward magnitude when the agent's performance improves. Haarnoja et al. solved this by implementing automatic entropy tuning, updating the temperature parameter at every step to keep roughly the same entropy [20].

As an Actor Critic method, SAC consists of an actor, $\pi(s, a)$, that executes actions and a critic, $Q(s, a)$, that evaluates the value of taking a specific action when in a certain state. Both the actor and the critic are represented by neural networks, where two networks are used for the critic to speed up training [21]. The actor network outputs a mean and standard deviation that represents a Gaussian distribution. During training, actions are sampled from this distribution and during testing the mean is used directly. The softmax output from the RMN-network sets the seven state dimensions and the action space is one-dimensional with a

value ranging from -1 to 1. A positive action means human command and a negative action gives the autonomous agent the command. A more detailed description of the algorithm implemented and how the SAC agent is trained can be found below.

**Algorithm 1** SAC for Policy Blending via Facial Expressions

Initiate Facial Recognition script (RMN)
Reset the Lunar Lander Environment
Set command to human action
**for** Every interaction step **do**
    Get Gym environment state $s_{Lunar}$ and reward $r$
    Add reward for facial expression agent, $r_{Face}$ += $r$
    **if** Human in control **then**
        Execute keyboard command
    **else**
        Get autonomous action, $a_{Auto}$, given $s_{Lunar}$
        Execute autonomous action $a_{Auto}$
    **end if**
    **if** Step modulo 50 = 0 **then**
        Get facial expression state $s_{Face}$
        Get facial expression action, $a_{Face}$ given $s_{Face}$
        **if** $a_{Face} > 0$ **then**
            Allow human control
        **else**
            Allow autonomous agent control
        **end if**
        Save transition from last state to replay buffer
        Load batch of transitions from replay buffer
        Perform gradient descent update of all ANNs
        Update temperature parameter $\alpha$
        Reset reward for facial expression, $r_{Face}$ = 0
    **end if**
**end for**

### *E. Experimental setup*

Unfortunately, due to time constrains and difficulties finding subjects during the current situation, the system has only been tested on the author. The experimental setup is explained in this section.
The human subject played with the environment following the algorithm described above. At roughly every 10 second a random generator determined whether noise should be added to the environment, making it impossible to see the Lander for the human operator. If noise was added it lasted for 10-20 seconds, a figure of the environment with added noise can be found below.

The facial expression agent had no other way to tell whether a disturbance was present than to analyze the operators face. The training process consisted of three trials of roughly 10 min of playing, with 2 min of offline updates of the networks in between. In total, the training procedure took around 30 min to achieve satisfactory performance.

Fig. 3. Screenshot of the OpenAI Gym's Lunar Lander environment when noise is added, the Lander is not possible to find.

## IV. Results

With only 30 minutes of training and 4 minutes of offline updates the facial expression agent was able to perform as expected, allowing the autonomous agent to take over control when noise disturbed the human operator. To validate the performance, two tests was conducted, one of the full system at every stage of training and one where different facial expression was mapped to actions. The results from the first test is presented in Table 1.

| Training time | Mean Reward |
|---|---|
| 10 min (First trial) | -5.2 |
| 20 min (Second trial) | 540.6 |
| 30 min (Third trial | 783.7 |

Between every stage of training, 10 trials of 1000 steps was conducted and the reward per trial saved. Table 1 shows the average reward per trial and a substantial increase in performance with respect to training time is clear.

The author also observed a clear difference in behaviour from the agent throughout training. After the first stage, actions were still quite random, leading to the human being allowed control even when disturbed which resulted in several crashes. After the second stage, there was clear indications that different facial gestures led to different outputs. However, the agent still required quite substantial difference in expressions to change its decisions. In other words, the operator would have to show significant indicators of confusion/anger for the robot to take over vice versa almost smile to be allowed control. After the final stage of training, this improved and the operator was then allowed control even when keeping a neutral face and only slight indication of confusion/anger/disgust was needed to make the facial expression agent choose autonomous command.

In the second test, the facial expressions of the operator and the corresponding action from the agent was recorded while playing in the environment with the fully trained agent. Below, in Fig. 4, some of these are presented.

We observe from Fig. 4 that with the fully trained agent, a neutral face allowed the human to remain in control and

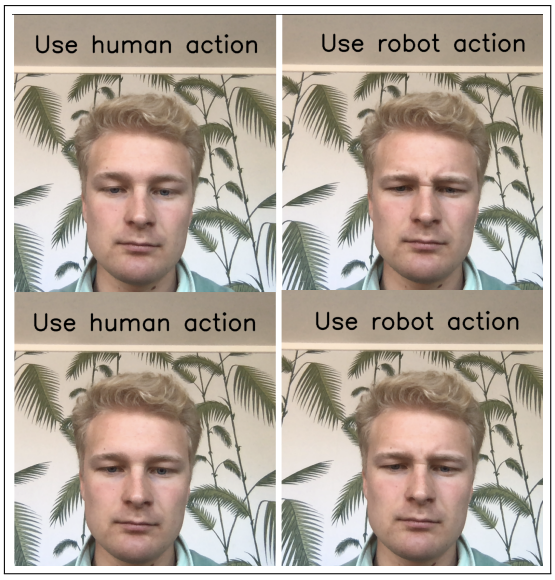


Fig. 4. Facial expressions of the human operator along with corresponding agent prediction.

when the eyebrows were slightly drawn together and the upper part of the nose wrinkled, the agent predicted that a disturbance was present and the autonomous control would take over.

## V. Conclusions

Overall this work successfully implement a DRL-agent that neatly solves the intended task. However, there are several limitations and also multiple interesting avenues for future work.

The main limitation of the work lays in the evaluation metrics of the results. Only using the author as subject is not good practice and a more thorough experimental procedure is needed to validate the results. It is not unlikely that a subject that is aware of how the system works will over-compensate their facial expressions to facilitate learning. A natural progression of this work would be to extend the experiments.

Moreover, it would be worthwhile to further investigate the impact of the facial recognition model that gives the input state to the first DRL-agent. With more time, a system that is compatible with Python 2, thus allowing for OpenFace to be used, could be implemented. If so, facial action units could be used as input instead, which would increase the dimensionality of the state and maybe allow for more interesting behaviour. The generalising abilities of the system could also be explored in further research, by for example training the system on one subject and testing on another, or perhaps use one type of disturbance during training and a different one when testing. After all, in a real-world scenario, no situation would be exactly as in a controlled environment and thorough testing of the system's robustness is necessary,

In summation, the DRL method implemented manages to learn in such a sample-efficient way that training with a human-in-the-loop is not too bothersome or exhausting. At the same time it solves the task at hand, showing that not only can a human operator's state of control be extracted from their facial expressions, but it is also feasible to do personal real-world training with human subjects.

## References

[1] L. F. Barrett, R. Adolphs, S. Marsella, A. M. Martinez, and S. D. Pollak, "Emotional expressions reconsidered: Challenges to inferring emotion from human facial movements," *Psychological Science in the Public Interest*, vol. 20, no. 1, pp. 1–68, 2019, pMID: 31313636.

[2] A. D. Dragan and S. S. Srinivasa, "A policy-blending formalism for shared control," *The International Journal of Robotics Research*, vol. 32, no. 7, pp. 790–805, 2013.

[3] D. Kortenkamp, D. Keirn-Schreckenghost, and R. P. Bonasso, "Adjustable control autonomy for manned space flight," in *2000 IEEE Aerospace Conference. Proceedings (Cat. No.00TH8484)*, vol. 7, 2000, pp. 629–640 vol.7.

[4] M. B. Dias, B. Kannan, B. Browning, E. Jones, B. Argall, M. F. Dias, M. Zinck, M. Veloso, and A. Stentz, "Sliding autonomy for peer-to-peer human-robot teams," in *Proceedings of the international conference on intelligent autonomous systems*, 2008, pp. 332–341.

[5] M. A. Assari and M. Rahmati, "Driver drowsiness detection using face expression recognition," in *2011 IEEE International Conference on Signal and Image Processing Applications (ICSIPA)*, 2011, pp. 337–341.

[6] Y. Zhang and C. Hua, "Driver fatigue recognition based on facial expression analysis using local binary patterns," *Optik*, vol. 126, no. 23, pp. 4501–4505, 2015.

[7] J. Broekens, "Emotion and reinforcement: affective facial expressions facilitate robot learning," in *Artifical intelligence for human computing*. Springer, 2007, pp. 113–132.

[8] A. H. Qureshi, Y. Nakamura, Y. Yoshikawa, and H. Ishiguro, "Robot gains social intelligence through multimodal deep reinforcement learning," in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2016, pp. 745–751.

[9] M. Deisenroth and C. E. Rasmussen, "Pilco: A model-based and data-efficient approach to policy search," in *Proceedings of the 28th International Conference on machine learning (ICML-11)*, 2011, pp. 465–472.

[10] A. Shafti, J. Tjomsland, W. Dudley, and A. A. Faisal, "Real-world human-robot collaborative reinforcement learning," *arXiv preprint arXiv:2003.01156*, 2020.

[11] R. S. Sutton, A. G. Barto *et al.*, *Introduction to reinforcement learning*. MIT press Cambridge, 1998, vol. 2, no. 4.

[12] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," 2016.

[13] J. X. Wang, Z. Kurth-Nelson, D. Tirumala, H. Soyer, J. Z. Leibo, R. Munos, C. Blundell, D. Kumaran, and M. Botvinick, "Learning to reinforcement learn," *arXiv preprint arXiv:1611.05763*, 2016.

[14] A. Hill, A. Raffin, M. Ernestus, A. Gleave, A. Kanervisto, R. Traore, P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, and Y. Wu, "Stable baselines," https://github.com/hill-a/stable-baselines, 2018.

[15] B. Amos, B. Ludwiczuk, and M. Satyanarayanan, "Openface: A general-purpose face recognition library with mobile applications," CMU-CS-16-118, CMU School of Computer Science, Tech. Rep., 2016.

[16] L. P. . T. A. Tran, "Facial expression recognition using residual masking network," 2020. [Online]. Available: https://github.com/phamquiluan/ResidualMaskingNetwork

[17] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," *arXiv preprint arXiv:1801.01290*, 2018.

[18] K. Rasul and J. Achiam, "Fired up," https://github.com/kashif/firedup/, 2019.

[19] B. D. Ziebart, A. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning," in *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, 2008.

[20] T. Haarnoja, S. Ha, A. Zhou, J. Tan, G. Tucker, and S. Levine, "Learning to walk via deep reinforcement learning," *arXiv preprint arXiv:1812.11103*, 2018.

[21] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International Conference on Machine Learning*, 2018, pp. 1582–1591.