Post-print of: Z. Kowalczuk, W. Jędruch and K. Szymański, "The Use of an Autoencoder in the Problem of Shepherding," 2018 23rd International Conference on Methods & Models in Automation & Robotics (MMAR), Miedzyzdroje, 2018, pp. 947-952, doi: 10.1109/MMAR.2018.8486067.

The use of an autoencoder in the problem of shepherding

Zdzisław Kowalczuk*, Wojciech Jędruch†, Karol Szymański*
*Department of Robotics and Decision Systems
Faculty of Electronics, Telecommunications and Informatics
Gdansk University of Technology
Narutowicza 11/12, 80-233 Gdansk
Email: kova@pg.edu.pl, wjed@pg.edu.pl, karszyma@pg.edu.pl
†Faculty of Navigation and Naval Weapon, Polish Naval Academy
Śmidowicza 69, 81-127 Gdynia
Email: w.jedruch@amw.gdynia.pl

***Abstract*—This paper refers to the problem of shepherding clusters of passive agents consisting of a large number of objects by a team of active agents. The problem of shepherding and the difficulties that arise with the increasing number of data describing the location of agents have been described. Several methods for reducing the dimensionality of data are presented. Selected autoencoding method using a Restricted Boltzmann Machine is then discussed. Autoencoding is deployed to reduce the dimensionality of graphic representation of clusters. Reduced data is used to train the neural network which determine movements of the active agents. Genetic algorithms are used in optimization of the parameters of this network.**

***Keywords*—shepherding, autoencoder, Restricted Boltzmann Machine, Genetic Algorithm**

I. Introduction

A. Shepherding

Shepherding consists in forcing movement of objects into the desired direction. The most common example of such a process is shepherding of sheep by a shepherd dog on pasture. The dog makes sure that the herd does not get distracted, but stays in a compact group. A similar mechanism can be observed in many areas of human activity, such as superintending the movement of crowd by the police, limiting the effects of spreading epidemics, extinguishing fire, and removing oil stains in water reservoirs. In times of intensive development of computer control systems, it is extremely important to construct algorithms which are to optimally perform the tasks mentioned above.

The problem of shepherding can be presented as interaction of two types of agents: active and passive. Active agents, by moving in a proper direction, force movement or eliminate passive agents. There are many versions of this problem depending on the task performed by active agents. In the case of relocation of passive agents, typical tasks are: focusing, driving, patrolling, keeping passive agents on a target surface or relocating them toward another position [11]. If the task is to eliminate passive agents, the problem can be reduced to passing through the areas they occupy. In this article, to check the legitimacy of the chosen approach, a case of shepherding is considered, in which passive agents are eliminated.

There are many ways in which active agents acquire information about the position of passive agents. There are shepherding problems, in which active agents track the location of passive agents. It may also happen that in addition to active and passive agents, there is an observer of the working area which provides the active agents with the needed information.

The literature suggests many models of shepherding process. Several heuristic algorithms are presented [11], [12], which are based on observation of real shepherding process, for example considering the behavior of shepherd dogs [20]. A simple algorithm of shepherding and successful testing of this algorithm on a real robot that herds a flock of geese to the point indicated in space is discussed in the paper [22].

Optimal algorithms for a few agents can be considered in continuous [18] or discrete [8] time domain, in which the solution is obtained via dynamic programming and using the Dijkstra algorithm.

There are very few papers describing the direct creation of a shepherding algorithm using machine learning. Optimization of autonomous agents' activities by teaching Bayesian networks proposed in [16] was applied to the discrete problem of shepherding of one sheep by one agent on a 4×4 mesh. Using a genetic algorithm for a similar problem of one sheep being driven by one agent is described in [17]. Determining the mode of operation of active agents controlled by neural networks, taught by genetic algorithms, is described in [13].

In the above-mentioned works, in which the control algorithm is set by learning or optimization, the number of agents must be small, except for heuristic algorithms. In the simplest case, the system includes only one active agent and one passive agent, although sometimes even a dozen agents can be considered. A small number of agents is applied obviously to limit the complexity of optimizing calculations. More agents appear only with heuristic algorithms. The constrains of optimal control algorithms only for small sets of agents is a significant limitation for their practical applications.

The problem of shepherding can be extended to the situation when passive agents are represented by a continuous area. Figure 1 illustrates three cases regarding the number of agents.

The case of passive agents forming continuous area occurs

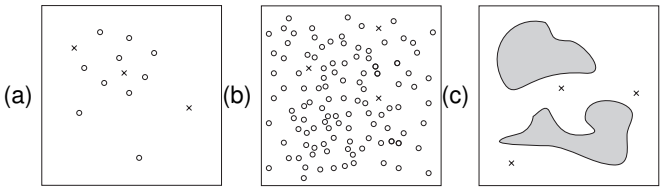

Fig. 1. Typical problems related to the number of agents in the process of shepherding: (a) small number of discrete agents – suitable for optimization by various methods; (b) large number of discrete agents – solved by heuristic methods; (c) big number of agents represented by continuous areas – as discussed in this paper (crosses represent active agents; and circles and continuous areas - stands for passive agents)

in nature e.g. during extinguishing fires or removing the effects of leaks. The area occupied by passive agents can be described as a set of elementary cells, which in the case of digital images are pixels. Unfortunately, for such a representation of passive agents, the number of data that the shepherding algorithm must operate to perform its task is unacceptably large. The novelty of this work consists in creating a system based on learning that can implement a simple process of shepherding objects occupying continues area. Striving to create such a system, it is necessary to reduce the dimensionality of data describing the location of the continuous passive agents.

II. Reducing dimension

Dimension reduction is simply a process of transforming multidimensional data into a space with fewer dimensions. It is often used in statistics [15], machine learning (to determine a model of a phenomenon) [5], pattern recognition and image processing [1]. Dimension reduction can be aimed at:

- shorting the processing time,
- reducing the amount of memory needed to store data
- getting rid of data linearly dependent on other data (improves the effectiveness of learning algorithms),
- data visualization in two-dimensional or three-dimensional space (for better understanding),
- extracting the features of data decisive for classification of patterns (improving generality performance).

There are two basic approaches to this problem: feature selection and feature extraction.

Feature selection consists in separating a subset of the original data dimensions (variables) [3]. The methods of feature selection are used when data contains many redundant or irrelevant variables. Search methods are used to select an appropriate subset. The simulated annealing algorithms, evolutionary methods and Particle Swarm Optimization are applied most frequently.

Feature extraction rely on the determination of new features describing these data (useful for a specific problem). In the case considered in this work, the data (from which the features should be extracted) are the numbers describing the brightness of subsequent pixels of a graphic image. Therefore, only the features that can be attributed to this type of data will be considered later in this subsection.

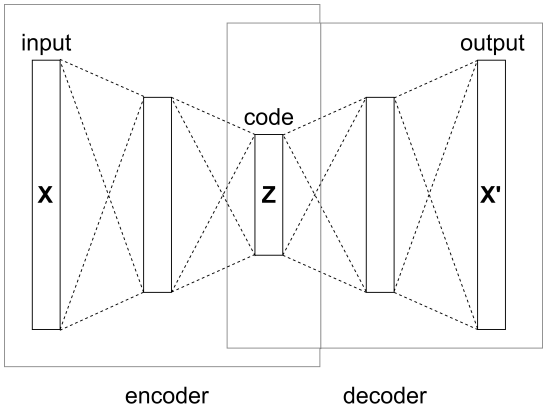

Fig. 2. Structure of the autoencoder

There are plenty of methods for extracting features of graphic patterns that can be used here. Among all of the possible solutions, the following methods can be listed:

- Heuristic (e.g. approximating the shape with a set of simpler shapes)
- Mathematical transformation
- Image processing (e.g skeletonization)
- Quadruple or map edges
- Compresses (e.g JPG, LZW)
- Analysis of the main components (PCA)
- Autoencoding.

In this paper the autoencoding method was chosen to reduce the dimension of graphic images of areas occupied by passive agents. The use of this method for the above-mentioned purpose has been discussed in [21].

III. Autoencoding

Autoencoder represents an artificial neural network used in the field of unsupervised learning for efficient data encoding [5], [9], [10], [14]. The aim of the autoencoder is to learn a certain representation of input data, which make it possible to reproduce the data again with a possibly greatest accuracy. Structurally, in the simplest form an autoencoder is a feedforward network with the same number of inputs and outputs, see Fig. 2. The autoencoder consists of two main parts: an encoder and a decoder. The purpose of the encoder is to extract features from the input data. The decoder has the reverse task that the well-reconstructed input data is presented at the output layer of the network. A code layer is a common layer of neurons for the encoder and decoder. The values of this neurons' activation are the taught (new) representation of data. If there are fewer neurons in this layer than in the network inputs, the dimension reduction of the data is achieved.

In practice it is difficult to train, i.e. to determine the optimal values of weights of the deep autoencoder, if classical methods of the weights initialization are applied. This is because in the learning process it is not possible to assess the real impact of weights in the first layer of the network on the output of the network. In this case, the phenomenon

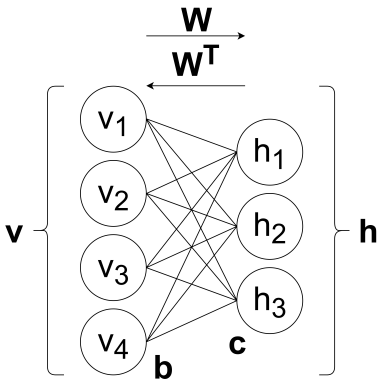

Fig. 3. Example of the Restricted Boltzmann Machine

of vanishing or exploding gradient is involved. Vanishing gradients result in a very low learning rate of the network, while exploding gradients pose the risk of getting stuck in a local minimum, i.e. leading to a suboptimal solution. One solution to this problem is pretraining proposed in 2006 by Hinton and Salakhutdinov [5]. It allows to initialize network weights with values that extract the desired features from the data. The selection of initial weights significantly improves the results achieved during a fine-tuning stage by the standard methods based on backward propagation.

Pretraining involves the iterative, layer by layer unsupervised neural network training. During the process of determining weights between two successive layers of the network, these two layers are treated as the Restricted Boltzmann Machine (RBM) [2]. Calculated weights are later used in finetuning the autoencoder.

RBM is a generative (i.e. used to generate data), stochastic neural network which generates random data, whose probability distribution is shaped by appropriate selection of weights and biases. A template of the Restricted Boltzmann Machine is shown in Fig. 3. It consists of two layers, which are called the visible and hidden. In RBM, neurons are binary units, i.e. they take one of the two values: 0 or 1. The state of the visible layer can be represented by an N-dimensional vector $\mathbf{v}$, where N is the number of neurons in the visible layer. The hidden layer state can be represented by an M-dimensional vector $\mathbf{h}$, where M is the number of neurons in the hidden layer. As can be seen in Fig. 3 each of the visible unit is connected to each hidden unit via a gain (weight). All weights are integrated into a matrix $\mathbf{W}$ of dimension $M \times N$. In addition, each neuron also has its own bias. Biases for visible layer are described by an N- dimensional vector $\mathbf{b}$, and the biases of the hidden layer are represented as the M- dimensional vector $\mathbf{c}$. The states of hidden and visible layers can be updated (i.e. the current value of the $\mathbf{v}$ and $\mathbf{h}$ vector can be replaced with new value). The states of visible units are set to one with probability $\sigma(\mathbf{W}^T\mathbf{h} + \mathbf{b})$, where $\sigma(x)$ is a logistic function and they are set to zero otherwise. Analogically, the new states of hidden units is set to one with probability $\sigma(\mathbf{W}\mathbf{v} + \mathbf{c})$.

The purpose of training RBM is to determine such weights and biases that will maximize the probability of generating training data in the visible layer of RMB. As can be deduced [7] [19], the probability can be described as

$$p(v) = \frac{1}{Z}\sum_{h} e^{-E(v,h)}$$

where $Z = \sum_{v,h} e^{-E(v,h)}$ and E, called the energy, is computed as follows:

$$E(\mathbf{v}, \mathbf{h}) = -\sum_{i \in visible} b_i v_i - \sum_{j \in hidden} c_j h_j - \sum_{i,j} v_i h_j w_{ij}$$

The probability that the state of the visible units will be set to the training data can be increased by adjusting weights and biases according to the derivative of the logarithm of this probability with respect to every weight or bias. As can be shown [4], these derivatives have the following form:

$$\Delta w_{i,j} \propto \frac{\partial \log p(\mathbf{v})}{\partial w_{i,j}} = \mathcal{E}[v_i h_j]_{data} - \mathcal{E}[v_i h_j]_{recon}$$

$$\Delta b_i \propto \frac{\partial \log p(\mathbf{v})}{\partial b_i} = \mathcal{E}[v_i]_{data} - \mathcal{E}[v_i]_{recon}$$

$$\Delta c_j \propto \frac{\partial \log p(\mathbf{v})}{\partial c_j} = \mathcal{E}[h_j]_{data} - \mathcal{E}[h_j]_{recon}$$

where $\mathcal{E}[\cdot]_{data}$ means expected value when in visible layer is data and hidden layer is driven by it, while $\mathcal{E}[\cdot]_{recon}$ means the expected value when RBM is in equilibrium (after infinitely many updates of both layers).

In practice, the calculation of $\mathcal{E}[\cdot]_{recon}$ is very computationally costly. Therefore a Contrastive Divergence algorithm [4] was proposed, which simplifies the computation of the term $[\cdot]_{recon}$, to adjust the weights and biases of RBM. Using this algorithm for pre-training deep autoencoder, firstly the inputs of the autoencoder are treated as the visible layer of RBM, and the first hidden layer of the autoencoder as the hidden layer of RBM. Once the weights and biases of the first RBM have been trained, the activations of the hidden units are written as new data for training the second RBM. The first hidden layer of the autoencoder is then treated as a new RBM visible layer and the second hidden layer of neurons as a new RBM hidden layer. In the same way, all subsequent layers of the encoder are trained.

In the fine-tuning stage, adjusted matrices $\mathbf{W}$ and vectors $\mathbf{c}$ are used as the initial encoder weights and biases and the matrices $\mathbf{W}^T$ and the vectors $\mathbf{b}$ are applied as the initial decoder weights and biases. Due to such an initialization the weights and biases are much closer to the optimal values than in case of their random initialization. As a result, a good quality training result can be achieved in acceptable time.

IV. Shepherding Algorithm

The reduced data of the position of passive agents are used to control the movement of active agents. The displacements of active agents are determined by neural networks.

In order to confirm the proposed approach, we choose a variant of shepherding, in which passive agents are not pushed, but eliminated when one of the active agents is close

enough and falls within one of the four neighboring cells. Similar process occurs, for example, during fire fighting. In addition, active agents are provided with all information about the location of active agents. At each step of the algorithm, the active agents are able to move to one of their eight neighboring cells in the workspace.

The inputs of the neural network are the transformed data of the location of active and passive agents. The data transformation process is illustrated in Fig. 4. First, the data is centered so that the center of mass of the passive agent is in the center of the work area. Passive and active agents are rotated so that the active agent, whose move is to be determined, is exactly under the mass center of the passive agents. This operation provide a useful rotational and displacement invariance to the observed agents. The prepared data of the location of passive agents is applied to the input of the autoencoder. The 2D image determining location of passive agents is treated by the autoencoder as a one-dimensional data, so the resultant feature vector is also one-dimensional.

Neural network determining the shifts of active agents is trained using a genetic algorithm. Individuals in this algorithm are sets of network weights and biases. Their fitness is evaluated on the basis of the number of passive agents which have not been eliminated in the evolutionary process. The fewer passive agents are left, the greater the chance that this individual is worth reproducing.

V. Experiments and results

Let us separately consider problems of shepherding for the herds of passive agents forming one coherent area and for herds that are represented as a few isolated areas. Figure 5(a) shows 5 examples of generated images of coherent herds (stains) and Figure 6(a) illustrates 6 examples of generated images of isolated herds (stains).

A. Dataset

Data for the conducted experiments were generated as stains of random shapes in the image of the size of 28 by 28 pixels. A stain means the surface area occupied by passive agents in the work area. We decided that in every analyzed image of the dataset the stain occupies 50 pixels. 30000 images have been generated containing coherent stains and other 30000, which may contain isolated stains. From each dataset, 5000 images were used as test data. Remaining images served as training data.

B. Autoencoding

A deep autoencoder with 7 fully connected hidden layers was used for autoencoding. One model was trained for dataset containing images of coherent stains and the other for dataset containing images of isolated stains. In subsequent layers there were 600, 300, 150, 30, 150, 300, 600 neurons, respectively. At the pre-trening stage, a mini-batch gradient descent with the batch size of 100 was applied. The momentum method was used to update the parameters. In addition, after each parameter update, they were scaled by 0.99998. Such a regularization was used to achieve better generalization. The length of the pre-training phase was confirmed to 50 epochs. During the fine-tuning phase, the same mini-batch gradient descent was applied and the parameters were updated using the RMSPROP method [6]. In this way, two models of the deep autoencoder converting images of size 28×28 pixels into 30 real numbers have been identified.

The outputs of the trained models of the autoencoder are shown in Fig. 5b and 6b. As can be seen the data have been reconstructed in a very precise way. The results of the autoencoder have been compared to the effects of the PCA method. By means of the PCA algorithm, the input data has been reduced to a vector of the same size as the one of the generated by the autoencoder (i.e. to a 30-dimensional vector for coherent stains and to a 50-dimensional vector for isolated stains). Then, the input data has been reconstructed using the reduced data. Obtained results are presented in Fig. 5c and 6c. It has also been checked how many dimensions the resulting PCA vector must have to obtain the same mean square error of the reconstruction process. Namely, it has been verified, that it is 50 features in the case of coherent stains, and 80 for the isolated stains. The reconstruction results for the analyzed vectors are presented in the Fig. 5d and 6d. As can be seen there the blurring effects are clearly different. In the case of autoencoding some information (small details) about the stain is lost, but the stain has sharp edges. The stains obtained with the use of PCA method retain information about details, but the stains are more blurred, making it difficult to decide which pixel belongs to the spot and which doesn't.

C. Shepherding

Let us consider experiments with 3 active agents participating in the process of shepherding. Their movements are controlled by three identical fully connected networks. The network consists of two layers of neurons (30-2) with a sigmoid activation function. At the input of the network, 30 code coefficients are given for coherent stains and 50 for isolated stains (obtained from the autoencoder) where as 6 coordinates define the location of 3 active agents. One of the output neurons determines whether the active agent is to move towards the mass center of the passive agents or, on the contrary, it should move away from it. The second output neuron decides if the agent should move left or right relative to the center of mass of the passive agents.

Training of the network was performed using a genetic algorithm with the population size of 50 individuals. Each individual was represented as a vector of 1172 weights (30 neurons in input layer $\times$ (30 parameters of the stains $+$ 6 parameters defining position of active agents) $+$ 30 biases + 2 neurons in output layer $\times$ 30 inputs $+$ 2 biases) of the neural network determining the shifts of active agents for coherent stains. For isolated stains the vectors had 1772 coordinates. Each epoch of the genetic algorithm consisted 4 stages: assessment of the fitness of individuals, selection, crossover and mutation.

Assessment of fitness was determined based on 20 full training scenarios, where a scenario means one multi-step shepherding process. Each scenario was selected from a set of 25000 training images. For each scenario, the starting positions of 3 active agents were selected randomly from within the workspace.

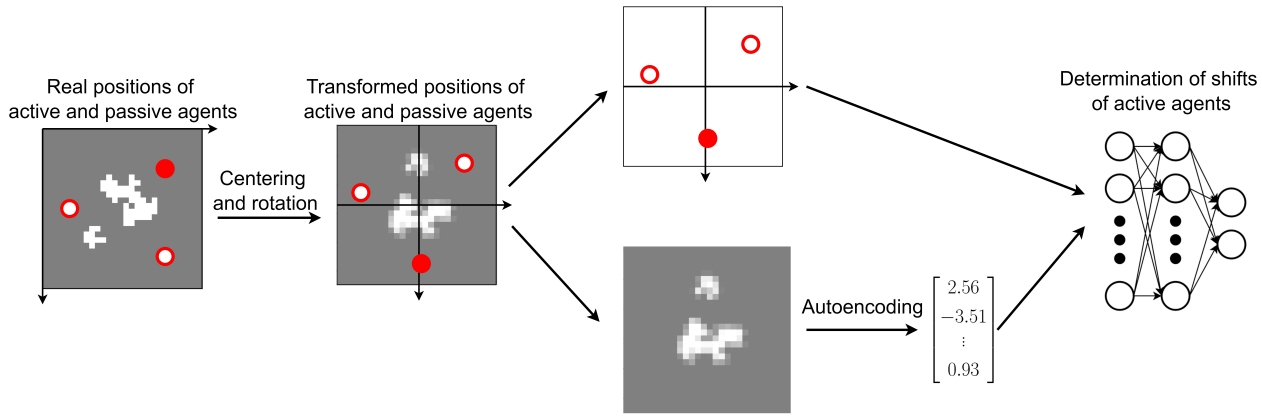

Fig. 4. A scheme for determining the shift of active agents. Note that the vector obtained from autoencoder represent the feature of the stain.

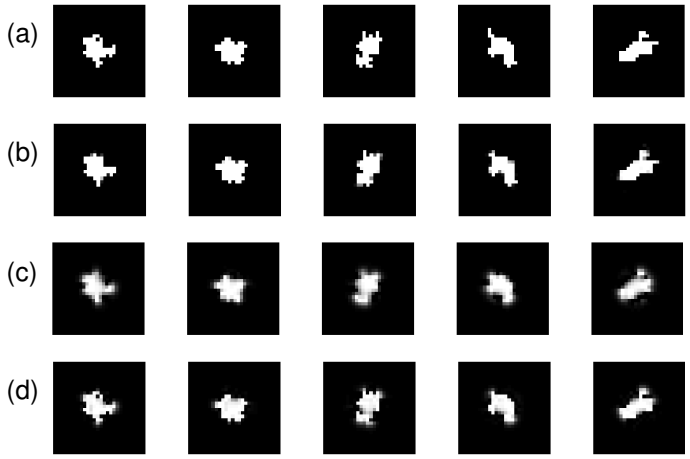

Fig. 5. Illustration of the effects of dimensional reduction of images of coherent patches: (a) randomly generated input images; (b) reconstruction of input images resulting from 30 real numbers generated by an autoencoder; (c) reconstruction of input images resulting from 30-dimensional vector obtained using the PCA method; (d) reconstruction of input images resulting from 50-dimensional vector obtained using the PCA method;

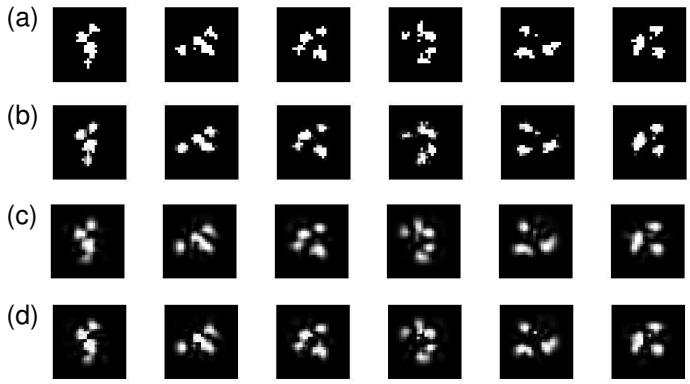

Fig. 6. Illustration of the effects of dimensional reduction of images of isolated patches: (a) randomly generated input images; (b) reconstruction of input images resulting from 50 real numbers generated by an autoencoder; (c) reconstruction of input images resulting from 50-dimensional vector obtained using the PCA method; (d) reconstruction of input images resulting from 80-dimensional vector obtained using the PCA method;

The process of shepherding took 50 iterations (i.e. each active agent performed 50 moves in any direction). After evaluating 20 training shepherding scenarios, the fitness of individuals was assessed on the basis of the number of pixels still occupied by passive agents after finishing shepherding. The selection process was carried out using the roulette method.

The crossover of randomly selected pairs took place with a probability of 0.7 and was carried out in accordance with the following formula:

$$\hat{o}_k^i = \beta_k o_k^i + (1-\beta_k) o_k^j$$
$$\hat{o}_k^j = (1-\beta_k) o_k^i + \beta_k o_k^j$$

where $\hat{o}_s^r$ and $\hat{o}_s^r$ are the n-th element of the weight vector of the m-th individual before and after crossover, and β_k represents a realization of a random variable with a uniform distribution which takes its value ranging from 0 to 1.

The mutation of individuals was carried out by adding (with a probability equal to 0.01) to each element of the individual's vector a random variable having normal distribution with a zero expected value and a variance equal to 2.

For the considered problem of shepherding passive agents forming a coherent shapes, the genetic algorithm (run using the mechanism of elitism) was performed for 100 epochs. After completing the algorithm, the fitness of individuals was again assessed on a set of 100 test scenarios, while the individual with the highest efficiency was chosen as the winner.

The total number of pixels left after 100 tests was equal to 77 (the maximum value that could be achieved is equal to 100 $\times$ 50). Hence, this results in the winning unit being 98.46% effective. Note that on average, only one pixel consisting of a passive agent was left after shepherding.

In Fig. 7 three trajectories of active agents are presented. The initial stain is displayed in the background, in gray. As you can see, the agents learned to move spirally towards the center of the stain and circulate in a clockwise direction. In this example, the active agents managed to eliminate the whole stain of passive agents.

For the considered problem of shepherding passive agents forming isolated shapes (herds), the genetic algorithm was performed for 2500 epochs. The total number of pixels left

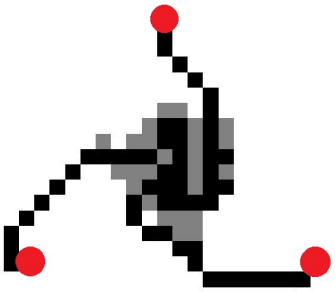

Fig. 7. Trajectories of active agents in the process of shepherding coherent stains. The initial positions of the active agents are marked as circles

after 100 tests was equal to 400 (the maximum number which is equal to 100×50). Thus, the winning individual has the effectiveness of 92.00%. On average, only four pixels representing passive agents were left after shepherding.

VI. Summary and future research

This article has implemented the task of shepherding a large number of passive agents creating a continuous area. Two types of data have been generated, representing situations in which passive agents form a coherent shape or isolated (disjoint) herds. Reduction of the dimensionality of the data describing the positions of passive agents has been obtained using a deep autoencoder.

A pretraining technique based on the training of Restricted Boltzmann Machine has been used to train the autoencoder. Then the reduced data has been used for genetic training of a neural network, which determines the movements of active agents. The preliminary results have shown that this approach can be extended to other, more complex problems of competing.

Further research in this area may focus on the type of criterion functions and variants of genetic algorithms and data processing, in order to speed up the learning process.

During the process of eliminating passive agents, it may happen that coherent stains become a group of smaller isolated stains. In this case, it would be worth replacing the type of neural network models so that the behavior of active agents would be optimal for such cases.

In this work, only the simplest variant of the shepherding problem has been considered, in which active agents are eliminating static passive agents. When considering a real shepherding problem (like fire fighting), one should model the passive agents as expanding stains.

You should also take into account additional factors affecting the fire development, such as wind direction, material flammability near the fire and ambient temperature. In such cases, it would be useful to teach active agents to keep mutual distances (from each other) to reduce the effects of fire spread.

References

[1] Ella Bingham and Heikki Mannila. Random projection in dimensionality reduction: applications to image and text data. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 245–250. ACM, 2001.

[2] Asja Fischer and Christian Igel. *An Introduction to Restricted Boltzmann Machines*, pages 14–36. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[3] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182, 2003.

[4] G. E. Hinton. Training products of experts by minimizing contrastive divergence. In *Neural Computations*, 2002.

[5] G E Hinton and R R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, July 2006.

[6] Swersky N. Hinton G., Srivastava N. Overview of mini-batch gradient descent. In *Neural Networks for Machine Learning*, 2014.

[7] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. In *Proceedings of the National Academy od Sciences of the United States of America*, volume 79, 1982.

[8] Pushkin Kachroo, Samy A Shedied, John S Bay, and Hugh Vanlandingham. Dynamic programming solution for a class of pursuit evasion problems: the herding problem. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 31(1):35–41, 2001.

[9] James David Keeler, Eric Jon Hartman, and Ralph Bruce Ferguson. Method for operating a neural network with missing and/or incomplete data, November 24 1998. US Patent 5,842,189.

[10] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[11] Jyh-Ming Lien, O. Burçhan Bayazit, Ross T. Sowell, Samuel Rodríguez, and Nancy M. Amato. Shepherding behaviors. In *ICRA*, pages 4159–4164. IEEE, 2004.

[12] Jyh-Ming Lien, Samuel Rodríguez, Jean-Phillipe Malric, and Nancy M. Amato. Shepherding behaviors with multiple shepherds. In *ICRA*, pages 3402–3407. IEEE, 2005.

[13] Romanowski M. and Jedruch W. Neuronowo sterowane agenty w procesie zespołowego zaganiania. In *Zaawansowane Systemy Automatyki i Diagnostyki*, pages 193–204, Gdańsk, 2015. PWNT.

[14] Andrew L Maas, Quoc V Le, Tyler M O'Neil, Oriol Vinyals, Patrick Nguyen, and Andrew Y Ng. Recurrent neural networks for noise reduction in robust asr. In *Interspeech*, pages 22–25, 2012.

[15] Sam T. Roweis and Lawrence K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *SCIENCE*, 290:2323–2326, 2000.

[16] Ferat Sahin and John S Bay. Structural bayesian network learning in a biological decision-theoretic intelligent agent and its application to a herding problem in the context of distributed multi-agent systems. In *Systems, Man, and Cybernetics, 2001 IEEE International Conference on*, volume 3, pages 1606–1611. IEEE, 2001.

[17] Alan Schultz, John J. Grefenstette, and William Adams. Robo-shepherd: Learning complex robotic behaviors. In *In Robotics and Manufacturing: Recent Trends in Research and Applications, Volume 6*, pages 763–768. ASME Press, 1996.

[18] Samy A. Shedied. Optimal trajectory planning for the herding problem: a continuous time model. *Int. J. Machine Learning & Cybernetics*, 4(1):25–30, 2013.

[19] P. Smolenski. Information processing in dynamical systems: Foundations of harmony theory. In *Parallel Distributed Processing: Volume 1: Foundations*, pages 194–281. MIT Press, Cambridge, 1986.

[20] Daniel Strömbom, Richard P. Mann, Alan M. Wilson, Stephen Hailes, A. Jennifer Morton, David J. T. Sumpter, and Andrew J. King. Solving the shepherding problem: heuristics for herding autonomous, interacting agents. *Journal of The Royal Society Interface*, 11(100), 2014.

[21] K. Szymanski, W. Jedruch, and Z. Kowalczuk. Learning by autoencoding reducing the size of images for the tasks of shepherding. In *http://dps17.mchtr.pw.edu.pl/*, 2017.

[22] Richard T. Vaughan, Neil Sumpter, Jane Henderson, Andy Frost, and Stephen Cameron. Experiments in automatic flock control. *Robotics and Autonomous Systems*, 31(1-2):109–117, 2000.