1. Konrad NIDERLA[1,2], 2. Grzegorz KŁOSOWSKI[1,3]

Centrum Badawczo Rozwojowe Netrix S.A.(1), Lubelska Akademia WSEI (2), Politechnika Lubelska (3)
ORCID: 1. 0000-0003-1280-0622; 2. 0000-0002-4678-4512

doi:10.15199/48.2024.04.46

Środowisko nauki ze wzmocnieniem do sterowania ramieniem robota przemysłowego

Streszczenie. *W ostatnich latach wzrasta zainteresowanie wykorzystaniem uczenia ze wzmocnieniem w dziedzinie sterowania robotyki. W tym kontekście istotne jest badanie i porównanie różnych algorytmów RL, które mogą być efektywnie zastosowane do zadań sterowania robotami. W tym artykule porównano trzy popularne algorytmy RL: Trust Region Policy Optimization (TRPO), Proximal Policy Optimization (PPO) i Advantage Actor Critic (A2C), koncentrując się na ich zastosowaniu w sterowaniu ramieniem robota. Eksperymenty przeprowadzono w środowisku z symulowanym ramieniem robota wykorzystując szereg bibliotek i struktur programistycznych tzw. frameworków, a wyniki działania poszczególnych algorytmów zaprezentowano.*

Abstract*. In recent years, there has been increasing interest in the use of reinforcement learning in the field of robotics control. In this context, it is important to study and compare different RL algorithms that can be effectively applied to robot control tasks. This article compares three popular RL algorithms: Trust Region Policy Optimization (TRPO), Proximal Policy Optimization (PPO), and Advantage Actor Critic (A2C), focusing on their application in robotic arm control. The experiments were carried out in an environment with a simulated robot arm using a number of libraries and programming structures, the so-called frameworks, and the results of individual algorithms were presented (**Learning environment with reinforcement for industrial robot arm control**).*

Słowa kluczowe: Reinforcement Learning, ramię robota, TRPO, PPO, A2C.
Keywords: Reinforcement Learning, robot arm, TRPO, PPO, A2C.

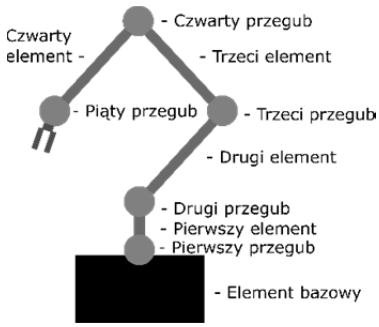
Wstęp

Sterowanie ramieniem robota przemysłowego z wykorzystaniem systemów sztucznej inteligencji jest problemem wielokrotnie poruszanym w różnych opracowaniach [1,2,3]. Mając na uwadze między innymi doświadczenia polegające na wykorzystaniu systemów uczenia ze wzmocnieniem (Reinforcement Learning, LR) do sterowania grami Atari [4,5] w których systemy głębokiego uczenia ze wzmocnieniem osiągają wyniki porównywalne z wynikami osiąganymi przez zawodników, można stwierdzić że w dziedzinie sterowania robotami przemysłowymi za pomocą systemów sztucznej inteligencji nadal są obszary do eksploracji i optymalizacji. Wykorzystanie systemów głębokiego uczenia ze wzmocnieniem niesie ze sobą potencjał uczenia logiki działania wybranych urządzeń za pomocą wielokrotnych prób, w których każde z podejść przynosi nową dawkę informacji, bez konieczności definiowania ścisłych reguł i zasad oraz wchodzenia w zawiłości teorii kontroli. System sterowania (Agent) w procesie uczenia sam optymalizuje nastawy w procesie trenowania tak aby maksymalizować nagrodę, nagroda z kolei jest zależna od osiąganych rezultatów. W artykule przedstawiono środowisko symulacji robota z siedmioma stopniami swobody oraz porównano działanie trzech algorytmów głębokiego uczenia ze wzmocnieniem: TRPO, PPO i A2C, które zyskały popularność ze względu na ich efektywność w różnorodnych zastosowaniach. Wskazane algorytmy były trenowane na stworzonym środowisku z wykorzystaniem biblioteki optymalizującej wybrane parametry, prowadząc do uzyskania możliwie najlepszych wyników.

Środowisko testowe

Wizualizacja środowiska testowego została przedstawiona na rysunku 1. Składa się ono z cyfrowego modelu robota KUKA iiwa 14 umieszczonego na płaszczyźnie stanowiącej (w zasięgu ruchu robota) przestrzeń roboczą urządzenia. W losowym położeniu przestrzeni roboczej umieszczano krążek będący celem dla ramienia robota. Zadaniem systemu sterownia (Agenta) jest doprowadzenie ramienia robota w bezpośrednie sąsiedztwo krążka. Model robota w formacie Unified Robot Description Format (URDF) [6] pozyskano z otwartego repozytorium ROS-Industrial [7]. URDF został opracowany przez członków zespołu Robot Operating System (ROS) [8], opisuje on robota jako drzewo elementów (link) połączonych przegubami (joint). Elementy reprezentują fizyczne komponenty robota, a przeguby przedstawiają sposób, w jaki jeden element porusza się względem drugiego, definiując położenie poszczególnych elementów w przestrzeni. Ideę takiej definicji w sposób graficzny przedstawiono na rysunku 2.

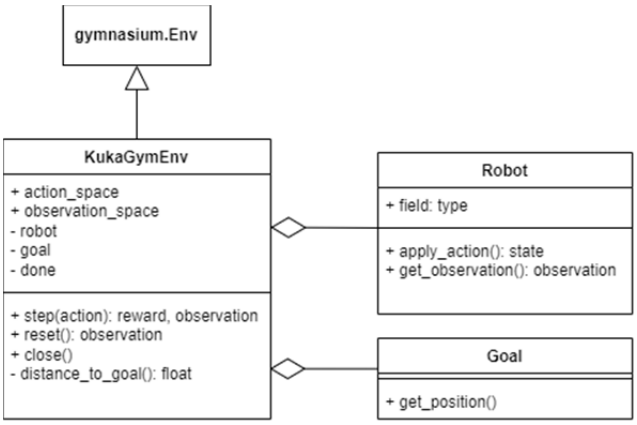
Rys. 1 Model środowiska testowego z ramieniem robota



Rys. 2 Idea definicji formatu URDF

W środowisku symulacyjnym od odwzorowania ruchów robota wykorzystano bibliotekę pybullet [9], za pomocą której została wczytana definicja robota do systemu. Biblioteka ta jest interfejsem Pythona dla Bullet Physics

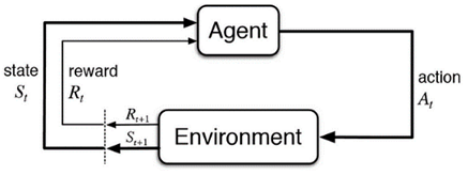
SDK [10], która to została stworzona głównie do badań w dziedzinie robotyki, detekcji kolizji, wirtualnej rzeczywistości, gier, efektów wizualnych, uczenia maszynowego itp. PyBullet oferuje zarówno narzędzia do symulacji fizyki, jak i narzędzia do wizualizacji, obsługuje dynamikę ciał sztywnych i kolizje w czasie rzeczywistym a zatem doskonale nadaje się do symulacji robotów i jest często wykorzystywana do tego celu. Zastosowanie biblioteki pybullet pozwoliło zarówno symulować i wizualizować działanie ramienia robota wytrenowanego ale również w robota czasie trenowania. Wizualizacja taka pomaga zrozumieć na jakim etapie nauki jest urządzenie, jakie ograniczenia występują w stworzonym środowisku, gdzie utknął proces nauki, a zatem daje dodatkowe możliwości eliminowania błędów i problemów. Wszystkie wyżej zdefiniowane elementy zostały zapakowane do środowiska zbudowanego z wykorzystaniem biblioteki gymnasium [11]. Gymnasium stworzone przez fundację Farama, jest to rozwijana wersja narzędzia OpenAI Gym [12], dostarczającego wiele środowisk testowych z jednolitym interfejsem środowisko-agent dla systemów uczenia ze wzmocnieniem. Ustandaryzowany interfejs pozwala z jednej strony uruchamiać istniejące algorytmy RL na wybranym środowisku, również własnym, z drugiej zaś strony umożliwia testowanie istniejących i własnych algorytmów na istniejących środowiskach, w łatwy sposób porównując ich cechy. Diagram klas UML dla stworzonego środowiska KukaGymEnv został przedstawiony na rys. 3.



Rys. 3 Diagram klas środowiska testowego

Warto zwrócić uwagę na dwie metody modelu środowiska testowego, czyli klasy KukaGymEnv: metodę reset() i step(action) z których pierwsza odpowiada za ustawienie środowiska do stanu początkowego i zwraca aktualny stan środowiska S_0 na początku każdej iteracji treningu, natomiast druga metoda aplikuje do wykonania w środowisku akcję At, zwracając stan środowiska S_t+1 oraz nagrodę R_t+1 po wykonaniu akcji. Nagroda za bieżącą akcję wykorzystana zostaje podczas trenowania agenta do poprawienia jego skuteczności zgodnie z zastosowanym algorytmem [13]. Ideowy proces trenowania agenta przedstawiono na rysunku 4.

Zadaniem Agenta w testowym środowisku jest, na podstawie obserwacji położenia celu i elementu wykonawczego robota w przestrzeni 3D (wektor sześciu wartości), taka regulacja siedmiu przegubów robota aby końcówka ramienia zbliżyła się do krążka na ustaloną, możliwie małą odległość. Funkcja nagrody została zdefiniowana w sposób wskazujący Agentowi kierunek podążenia ramieniem przy każdej akcji agenta. Nagroda stanowi odległość euklidesową ramienia od celu wyliczoną w poprzednim kroku, pomniejszoną o odległość wyliczoną w kroku bieżącym. W ten sposób każdy ruch ramienia robota przybliżający go do celu premiowany był dodatnią wartością nagrody, a dodatkowo po osiągnięciu celu nagroda jest powiększana o 10 punktów, natomiast po przekroczeniu limitu kroków nagroda jest pomniejszana o 10 punktów.



Rys. 4 Diagram ideowy systemu uczenia ze wzmocnieniem

Zdefiniowane środowisko służyło jako przestrzeń robocza wybranych algorytmów uczenia ze wzmocnieniem. Wybrano trzy algorytmy, zaimplementowane odpowiednio: PPO i A2C w bibliotece stable-baselines3 (SB3), TRPO w bibliotece stable-baselines3-contrib (SB3 Contrib). Biblioteka SB3 stanowi zbiór przetestowanych i udokumentowanych implementacji algorytmów uczenia ze wzmocnieniem realizowanych za pomocą popularnego frameworka PyTorch, natomiast SB3 Contrib jest eksperymentalnym rozszerzeniem biblioteki SB3, dostarczającym dodatkowych eksperymentalnych implementacji lub implementacji nowych algorytmów.

Algorytmy powyższe uruchamiane były na środowisku za pomocą narzędzia optymalizacji hiperparametrów optuna. Optuna odpowiedzialna jest za wielokrotne uruchamianie procesu uczenie za pomocą wybranego algorytmu, jednocześnie dostosowując hiperparametry procesu uczenia przy każdej próbie, w sposób umożliwiający możliwie szybką penetrację przestrzeni hiperparametrów. Optuna ma wbudowane kilka strategii doboru parametrów: losowy, przeszukujący hiperparametry w określonej siatce, wg algorytmu CMA-ES oraz domyślny TPESampler. Warto wiedzieć że narzędzie to oferuje również tzw. sekatory (pruners) których zadaniem jest przerwać proces nauki gdy próba nie rokuje na dostarczenie dobrych rezultatów. W opisywanych badaniach zrezygnowano z wykorzystania tej funkcjonalności. Dla zachowania równowagi zdecydowano o wykorzystaniu domyślnych parametrów dla każdego z algorytmów z wyjątkiem trzech hiperparametrów optymalizowanych podczas realizowanych prób.

1. Wskaźnik uczenia (learning rate) – określa jak duże kroki są podejmowane przy aktualizacji parametrów podczas nauki.
2. Współczynnik dyskontowania (gamma) – określa jak duży wpływ mają przyszłe nagrody w porównaniu z bieżącymi.
3. Współczynnik λ w GAE, tj. metodzie służącej do wyznaczania funkcji przewagi. Idea funkcji przewagi została opisana w dalszej części.

Każda próba zawierała 3000 epok po 1000 iteracji. Wynik każdej epoki stanowił średnią z ostatnich 100 iteracji, natomiast wynik każdej próby liczony był jako suma wyników ostatniego 1000 epok. Podczas eksperymentów wytrenowano 150 modeli, po 50 dla każdego algorytmu, w grupach po 5 sesji treningowych, oznacza to że każdy algorytm miał przeprowadzone 10 treningów po 5 różnych konfiguracji parametrów dobieranych przez optymalizator. Wybrane algorytmy domyślnie wykorzystują te same modele sieci neuronowej, składającej się z czterech warstw: warstwa wejściowa (6 neuronów), warstwa ukryta (64 neurony), kolejna warstwa ukryta (64 neurony), warstwa wyjściowa (7 neuronów akcji, jeden neuron wartości), gdzie warstwy ukryte używają funkcji aktywacji tangens hiperboliczny (tanh).

Podstawy teoretyczne wybranych algorytmów

Istnieje wiele metod do rozwiązywania zagadnień optymalizacyjnych [14-24]. Jak wspomniano wcześniej, trenowanie agenta polega na eksplorowaniu środowiska poprzez wielokrotne wykonywanie akcji, a na podstawie osiąganych wyników dostosowywaniu agenta za pomocą przyjętego algorytmu w taki sposób aby w kolejnych przebiegach procesu maksymalizować skumulowaną nagrodę. Istnieje wiele rodzajów algorytmów uczenia ze wzmocnieniem [25] jednak do prezentowanych badań wybrano trzy popularne algorytmy mające zastosowanie w środowiskach z ciągłymi przestrzeniami stanów i akcji TRPO, PPO i A2C. Wybrane algorytmy nie opierają się na modelu (model-free), charakteryzują się również wykorzystaniem metod gradientowych do optymalizacji tzw. strategii lub polityki (dalej używanej zamiennie), czyli funkcji określającej jakie działania powinien podjąć agent w aktualnym stanie środowiska. Algorytmy te, w porównaniu do algorytmów opartych na modelu (model–based) mają lepszą zbieżność, a algorytmy gradientowe zapewniają możliwość nauki działania dla środowisk stochastycznych.
Pierwotnym algorytmem gradientowym jest Policy Gradient Method [26,27], skupiający się na maksymalizowaniu nagrody $J(\pi_\theta)$ zgodnie ze wzorem (1) gdzie π_θ to polityka (w naszym przypadku sieć neuronowa, parametryzowana przez wagi θ), $\mathbb{E}_\tau$ to oczekiwana skumulowana nagroda natomiast R(τ) to nagroda dana wzorem (2), gdzie γ jest współczynnikiem zmniejszenia przyszłych nagród (discount).

(1) $$J(\pi_\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[R(\tau)]$$

(2) $$R(\tau) = \sum_{t=0}^{T} \gamma^t r_t$$

(3) $$\theta_{k+1} = \theta_k + \alpha\nabla_\theta J(\pi_\theta)|_{\theta_k}$$

(4) $$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta log\pi_\theta(a_t|s_t)R(\tau)\right]$$

Optymalizacja strategii π_θ polega na zmianie wag θ zgodnie ze wzorem (3) przy każdej iteracji algorytmu w kierunku zgodnym z gradientem (4). Algorytm ten nie jest szczególnie odporny na wahania, zbyt duże tempo nauki (learning rate) prowadzi do braku zbieżności, natomiast zbyt małe powodują że trening nie przynosi oczekiwanych rezultatów. Problemowi temu wychodzi naprzeciw algorytm Trust Region Policy Optimization (TRPO) [28], wprowadza on do algorytmu Policy Gradient Method dwie istotne koncepcje: Metodę Regionu Zaufania (Trust Region Method) oraz dywergencję Kullbacka-Leiblera. Funkcja $\mathcal{L}_\pi$ (5) określa jak dobra jest nowa strategia π' względem strategii poprzedniej, przed zmianą π.

(5) $$\mathcal{L}(\pi') = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{\infty} \frac{\pi'(s_t,a_t)}{\pi_\theta(s_t,a_t)} A(s_t,a_t)\right]$$

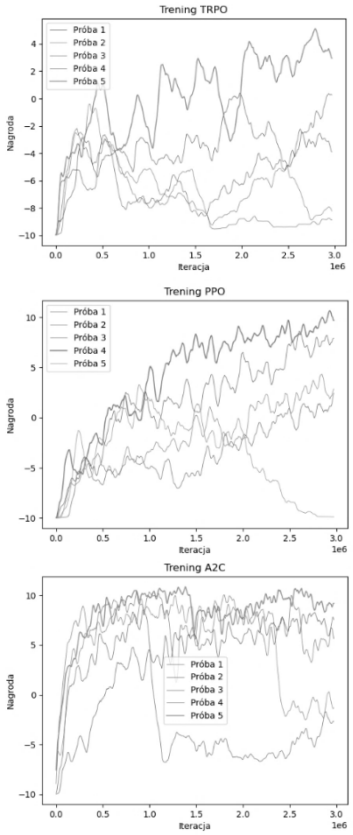
Definicja powyższa korzysta z funkcji przewagi (advantage function) $A(s_t,a_t)$ (6), określającej różnicę pomiędzy oczekiwaną wartością nagrody Q(s,a) dla wybranego działania a w aktualnym stanie s, a oczekiwaną wartością nagrody V(s) aktualnego stanu s przy realizowaniu bieżącej strategii.

(6) $$A(s,a) = Q(s,a) - V(s)$$

Ponieważ Q(s,a) jest równa sumie bieżącej nagrody i zdyskontowanych współczynnikiem γ przyszłych nagród (7), funkcję przewagi możemy również opisać wzorem (8).

(7) $$Q(s,a) = r + \gamma V(s')$$

(8) $$A(s,a) = r + \gamma V(s') - V(s)$$



Rys. 5 Nagrody dla pięciu sesji treningowych odpowiednio od góry: TRPO, PPO, A2C

Tabela 1. Wartości hiperparametrów

Próba	Wynik	GAE lambda	Gamma	LR
TRPO 1	-3513	0,9458	0,9872	4,46E-03
TRPO 2	-6809	0,9880	0,9837	9,16E-04
TRPO 3	-2947	0,9543	0,9552	4,07E-05
TRPO 4	-9069	0,9851	0,9783	1,83E-05
TRPO 5	3085	0,9725	0,9877	9,61E-04
PPO 1	-178	0,9807	0,9769	3,06E-05
PPO 2	1748	0,9837	0,9647	4,88E-05
PPO 3	6147	0,9121	0,9627	2,35E-05
PPO 4	8579	0,9460	0,9784	1,09E-04
PPO 5	-7488	0,9619	0,9859	5,36E-04
A2C 1	-5027	0,9544	0,9673	3,16E-05
A2C 2	2210	0,9338	0,9662	4,83E-05
A2C 3	6889	0,9797	0,9775	1,40E-05
A2C 4	6444	0,9584	0,9651	1,11E-04
A2C 5	9418	0,9494	0,9627	7,31E-05

Dywergencja Kullbacka-Leiblera wyrażona wzorem (9) opisuję miarę różnicy pomiędzy dwoma rozkładami prawdopodobieństwa, w analizowanym przypadku jest to miara różnicy rozkładu danych P i Q.

(9) $$D_{KL}(P||Q) = \sum_{x\in X} P(x)\log\frac{P(x)}{Q(x)}$$

Wykorzystując powyższe definicje, aktualizację strategii algorytmu TRPO opisujemy wyrażeniem (10) Gdzie δ jest promieniem regionu zaufania po przekroczeniu którego strategia nie jest aktualizowana.

(10) $$\pi_{k+1} = \underset{\pi'}{argmax}\,\mathcal{L}_{\pi_k}(\pi') \quad \text{przy założeniu że} \quad D_{KL}(\pi'||\pi_k) \le \delta$$

Metoda powyższa wymaga obliczania macierzy pochodnych drugiego rzędu tzw. macierzy hesjańskiej, a zatem złożoność obliczeniowa jest duża dla rzeczywistych zadań co skutkuje ograniczeniami wydajności przy systemach o dużej skali. Algorytm Proximal Policy Optimization (PPO) [29] nie narzuca twardych ograniczeń dla zaufanej strefy, w zamian aplikuje takie ograniczenie jako karę w funkcji celu, tym samym, aby osiągnąć rezultat optymalizacji może zostać użyty optymalizator pierwszego rzędu. Zmiana polega na wprowadzeniu współczynnika prawdopodobieństwa $r_t(\theta)$ (11) oraz zastępczej funkcji „przycinającej" (Clipped Surrogate Advantage Function) $\mathcal{L}^{CLIP}(\theta)$ (12) ograniczającej z dołu i góry parametrem ϵ wartość $r_t(\theta)$, tym samym ograniczając wielkość możliwej zmiany parametrów θ polityki.

(11) $$r(\theta) = \frac{\pi'(a|s)}{\pi_\theta(a|s)}$$

(12) $$\mathcal{L}^{CLIP}(\theta) = \mathbb{E}[min(r(\theta)A, clip(r(\theta), 1-\epsilon, 1+\epsilon)A)]$$

Koncepcja funkcji przewagi zastosowana została również w algorytmie Advantage Actor Critic (A2C) [30], w którym algorytm Actor Critic [13] został rozbudowany o dodatkową sieć neuronową. Jedna sieć dla krytyka odpowiedzialnego za estymację funkcji wartości V(s) wykorzystanej później do wyliczenia wartości funkcji przewagi A(s, a), natomiast druga sieć wykorzystana przez aktora do wyboru najlepszej w danej sytuacji akcji.

Wyniki eksperymentów i wnioski

Wyniki wybranych eksperymentów trenowania agenta z wykorzystaniem biblioteki optuna przedstawiono na wykresach na rysunku 5, odpowiednio dla trenowania algorytmem TRPO, PPO i A2C. Wykres przedstawia wartość średnią z 50 iteracji. Wartości parametrów dla każdej z sesji treningowych przedstawiono w Tabeli 1.

***Autorzy**: mgr inż. Konrad Niderla, Centrum Badawczo Rozwojowe Netrix S.A. ul Związkowa 26, 20-148 Lublin, Lubelska Akademia WSEI, ul. Projektowa 4, 20-209 Lublin, E-mail: konrad.niderla@netrix.com.pl; dr inż. Grzegorz Kłosowski, Politechnika Lubelska, ul. Nadbystrzycka 38, 20-618 Lublin, E-mail: g.klosowski@pollub.pl*

LITERATURA

[1] Maldonado-Ramirez A., Rios-Cabrera R., Lopez-Juarez I., A visual path-following learning approach for industrial robots using DRL, Robotics and Computer-Integrated Manufacturing, Volume 71(2021), 102130.
[2] Abdi A., Adhikari D., Park J.H., A novel hybrid path planning method based on q-learning and neural network for robot arm, Applied Sciences (2021), Vol. 11, No. 15.
[3] Abdi A., Ranjbar M.H.,H.Park J.H., Computer Vision-Based Path Planning for Robot Arms in Three-Dimensional Workspaces Using Q-Learning and Neural Networks. Sensors(2022), 22, 1697.
[4] Mnih V., Kavukcuoglu K., Silver D., Graves A., Antonoglou I., Wierstra D., Riedmiller M., Playing Atari with Deep Reinforcement Learning, arXiv e-prints (2013), arXiv.1312.5602.
[5] Mnih V., Kavukcuoglu K., Silver D., Rusu A.A., Veness J., Bellemare M.G., Graves A., Riedmiller M., Fidjeland A.K., Ostrovski G.,.Petersen S., Beattie C., Sadik A., Antonoglou I., King H., Kumaran D., Wierstra D., Legg S., Hassabis D., Human-level control through deep reinforcement learning, Nature 518 (2015), 529–533.
[6] http://wiki.ros.org/urdf/XML
[7] https://github.com/ros-industrial/kuka_experimental
[8] https://www.ros.org/
[9] https://pybullet.org/
[10] https://github.com/bulletphysics/bullet3
[11] https://gymnasium.farama.org/index.html
[12] Brockman G., Cheung V., Pettersson L., Schneider J., Schulman J., Tang J., Zaremba W., OpenAI Gym, arXiv preprint (2016) arXiv:1606.01540.
[13] Sutton R.S., Barto A.G., Reinforcement Learning: An Introduction Second Edition (2018). The MIT Press.
[14] Kania, W., Wajman, R., Ckript: a new scripting language for web applications, Informatyka, Automatyka, Pomiary W Gospodarce I Ochronie Środowiska, 12(2022), No. 2, 4-9.
[15] Styła, M., Adamkiewicz, P., Hybrid navigation system for indoor use. Informatyka, Automatyka, Pomiary W Gospodarce I Ochronie Środowiska, 12 (2022), No. 1, 10-14.
[16] Sikora R., Markiewicz P., Korzeniewska E., Using identification method to modelling short term luminous flux depreciation of LED luminaire to reducing electricity consumption, Scientific Reportst, 13 (2023), No. 1, 673.
[17] Lebioda, M., Korzeniewska, E., The Influence of Buffer Layer Type on the Electrical Properties of Metallic Layers Deposited on Composite Textile Substrates in the PVD Process, Materials, 16 (2023), No. 13, 4856.
[18] Rymarczyk T., Kozłowski E., Kłosowski G., Electrical impedance tomography in 3D flood embankments testing – elastic net approach, Transactions of the Institute of Measurement and Control, 42 (2020), No. 4, 680-690.
[19] Kłosowski G., Rymarczyk T., Niderla K., Rzemieniak M., Dmowski A., Maj M., Comparison of Machine Learning Methods for Image Reconstruction Using the LSTM Classifier in Industrial Electrical Tomography, Energies 2021, 14 (2021), No. 21, 7269.
[20] Rymarczyk T., Kłosowski G., Hoła A., Sikora J., Tchórzewski P., Skowron Ł., Optimising the Use of Machine Learning Algorithms in Electrical Tomography of Building Walls: Pixel Oriented Ensemble Approach, Measurement, 188 (2022), 110581.
[21] Koulountzios P., Rymarczyk T., Soleimani M., A triple-modality ultrasound computed tomography based on full-waveform data for industrial processes, IEEE Sensors Journal, 21 (2021), No. 18, 20896-20909.
[22] Koulountzios P., Aghajanian S., Rymarczyk T., Koiranen T., Soleimani M., An Ultrasound Tomography Method for Monitoring CO2 Capture Process Involving Stirring and CaCO3 Precipitation, Sensors, 21 (2021), No. 21, 6995.
[23] Kłosowski G, Rymarczyk T, Niderla K, Kulisz M, Skowron Ł, Soleimani M., Using an LSTM network to monitor industrial reactors using electrical capacitance and impedance tomography–a hybrid approach. Eksploatacja i Niezawodnosc – Maintenance and Reliability,25(2023), No.1,11.
[24] Kłosowski G., Rymarczyk T., Kania K., Świć A., Cieplak T., Maintenance of industrial reactors supported by deep learning driven ultrasound tomography, Eksploatacja i Niezawodnosc – Maintenance and Reliability; 22 (2020), No 1, 138–147.
[25] AlMahamid F., Grolinger K., Reinforcement Learning Algorithms: An Overview and Classification, arXiv e-prints (2022). arXiv:2209.14940
[26] Sutton R.S., McAllester D., Singh S., Mansour Y., Policy gradient methods for reinforcement learning with function approximation, Advances in Neural Information Processing Systems (2000), 12 (NIPS 1999)
[27] Peters J., Schaal S., Policy Gradient Methods for Robotics, IEEE/RSJ International Conference on Intelligent Robots and Systems, Beijing, China (2006), 2219-2225
[28] Schulman J., Levine S., Moritz P., Jordan M.I., Abbeel P., Trust Region Policy Optimization, arXiv e-prints (2015), arXiv.1502.05477
[29] Schulman J., Wolski F., Dhariwal P., Radford A., Klimov O., Proximal Policy Optimization Algorithms, arXiv e-prints (2017), arXiv.1707.06347
[30] Mnih V., Puigdomènech Badia A., Mirza M., Graves A., Lillicrap T.P., Harley T., Silver D., Kavukcuoglu K., Asynchronous Methods for Deep Reinforcement Learning, arXiv e-prints (2016), arXiv.1602.01783.