Sztuczne sieci neuronowe o radialnych funkcjach bazowych do śledzenia obiektów w obrazach wideo

J. SZYMONIK
szymonik.jacek@gmail.com

Instytut Systemów Informatycznych, Wydział Cybernetyki
Wojskowa Akademia Techniczna
ul. Gen. S. Kaliskiego 2, 00-908 Warszawa

W pracy przedstawiono opis sztucznej sieci neuronowej do lokalizacji i śledzenia obiektu w obrazach wideo z wykorzystaniem środowiska MATLAB oraz wyniki badań odporności algorytmu na mogące wystąpić zakłócenia. W artykule zaprezentowana została architektura sztucznej sieci neuronowej o radialnych funkcjach bazowych. Pokazany został zarówno algorytm śledzenia celu z wykorzystaniem powyższej architektury sieci, jak i metoda modelowania oraz lokalizacji celu. W podsumowaniu przedstawione zostały wyniki przeprowadzonych symulacji algorytmów śledzących opartych na sztucznych sieciach neuronowych.

Słowa kluczowe: śledzenie obiektów, sztuczne sieci neuronowe, radialne funkcje bazowe.

1. Wprowadzenie

Maszyny widzące i analizujące swoje otoczenie już istnieją, a ich ciągły rozwój jest przyspieszany poprzez postęp zarówno w dziedzinie mikroelektroniki, jak i w algorytmach analizy sygnału wideo. Proces estymacji pozycji interesującego nas obiektu w czasie, w sekwencji obrazów nazywamy śledzeniem obiektów. Jedną z fundamentalnych cech urządzeń przeznaczonych do obserwowania, rozumienia oraz reagowania na otaczające środowisko jest ich zdolność do detekcji oraz wcześniej wspomnianego śledzenia interesujących nas obiektów.

Główne problemy, które należy wziąć pod uwagę podczas projektowania trakera, są związane z podobieństwem wyglądu celu i innych obiektów otoczenia. Pozostałe przeszkody stojące na drodze efektywnego procesu śledzenia związane są ze zmianami wyglądu samego celu powodowanymi m.in.:

- zmianami pozycji
- światłem z otoczenia
- przesłonięciem celu (częściowym lub całościowym)
- szumem wprowadzanym przez urządzenia pomiarowe, sensory [5].

Wśród wielu podejść naukowych do problemu śledzenia, detekcji i rozpoznawania obiektów, ważną rolę odgrywają wielokryterialne algorytmy rozpoznawania wzorców [2], [3] bazujące na teorii podobieństwa [4], dynamiczne sieci Bayesa [7] oraz algorytmy oparte na sztucznych sieciach neuronowych [6]. Niniejszy artykuł jest poświęcony właśnie metodzie śledzenia adaptacyjnego *online* wykorzystującego sieci neuronowe o radialnych funkcjach bazowych [12].

2. Wybór architektury sztucznej sieci neuronowej

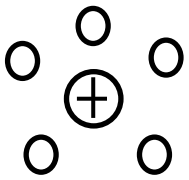
W przypadku algorytmów bazujących na sztucznych sieciach neuronowych, metodologia iteracyjnego szukania jest szeroko stosowana w celu aktualizacji parametrów sieci. Z tego powodu proces uczenia sieci jest bardzo intensywny obliczeniowo i często trenowanie sieci może zająć od kilku do kilkunastu godzin [1]. Kolejnym problemem przy korzystaniu z sieci neuronowych z propagacją wsteczną jest możliwość jej przetrenowania, co doprowadzi do utraty przez nią możliwości do generalizowania, czyli dostosowywania się sieci do nowych sytuacji. Co więcej algorytmy bazujące na uczeniu są zwykle projektowane do śledzenia wybranych obiektów, co wymaga każdorazowo długotrwałego procesu uczenia sieci *offline*.

Ze względu na powyższe czynniki w kolejnym punkcie zaprezentowana zostanie uniwersalna metoda śledzenia obiektów opierająca się na sieciach neuronowych o radialnych funkcjach bazowych i wykorzystująca uczenie sieci *online*.

Algorytm ten uwzględniać będzie również adaptację mogącego się zmieniać modelu obiektu wraz z każdym nowym obrazem sekwencji wideo.

3. Radialne funkcje bazowe

Specjalną odmianę sztucznych sieci neuronowych stanowią te o **radialnej funkcji bazowej** (ang. *radial basis function neural networks* – sieci neuronowe typu RBF), w której neuron ukryty realizuje funkcję zmieniającą się radialnie wokół wybranego centrum $\boldsymbol{c}$. Reprezentuje on hipersferę dokonującą podziału kołowego wokół punktu centralnego – rysunek 1 [10].



Rys. 1. Podział przestrzeni danych przez sieć radialną; Źródło: [10]

Niech $\boldsymbol{x}$ i $\boldsymbol{c}$ oznaczają dwa punkty leżące w przestrzeni R^d. Punkt $\boldsymbol{c}$ uważany jest za ustalony i nazywany punktem centrum. Punkt $\boldsymbol{x}$ uważany jest za zmienny.

Radialną funkcją bazową nazywa się funkcję G postaci:

$$G(\boldsymbol{x},\boldsymbol{c}) = G(r(\boldsymbol{x},\boldsymbol{c}), \tag{3.1}$$

gdzie:
$r(\boldsymbol{x},\boldsymbol{c})$ jest odległością między punktami $\boldsymbol{x}$ i $\boldsymbol{c}$. Centrum $\boldsymbol{c}$ w powyższej definicji gra rolę parametru funkcji. Funkcje radialne nazywane bywają czasami jądrami (ang. *kernels*).

Przy określaniu odległości między punktami $\boldsymbol{x}$ i $\boldsymbol{c}$ używa się najczęściej odległości euklidesowej [5].

$$r = r(\boldsymbol{x},\boldsymbol{c}) = ||\boldsymbol{x}-\boldsymbol{c}||_2 = \\ = \{(\boldsymbol{x}-\boldsymbol{c})^T(\boldsymbol{x}-\boldsymbol{c})\}^{1/2}. \tag{3.2}$$

Najczęściej stosowaną funkcją bazową jest radialna funkcja Gaussa określana wzorem:

$$G(\boldsymbol{x},\boldsymbol{c},\sigma^2) = \exp\{-\frac{||\boldsymbol{x}-\boldsymbol{c}||^2}{2\sigma^2}\}, \tag{3.3}$$

gdzie:
$\boldsymbol{x}$ jest wektorem wartości wejściowych, $\boldsymbol{c}$ to wektor określający centra funkcji bazowych. Wartość parametru σ kontroluje własności gładkości funkcji interpolującej [6].

4. Opis algorytmu śledzenia

Wiarygodne śledzenie obiektu może zostać osiągnięte, jeżeli możliwa jest separacja obszaru zawierającego obiekt od otaczającego tła w każdej chwili. Pierwszym krokiem będzie utworzenie funkcji gęstości prawdopodobieństwa (histogramów) dla obiektu (h_o) oraz tła (h_b). Do określenia, które z pikseli należą do obiektu, a które nie, wykorzystany został logarytm wskaźnika wiarygodności:

$$L_i = log\frac{\max\{h_o(i),\epsilon\}}{\max\{h_b(i),\epsilon\}}, \tag{4.1}$$

gdzie:
$h_o(i)$ i $h_b(i)$ są prawdopodobieństwami przynależności i-tego piksela odpowiednio do obiektu lub tła; ϵ jest małą niezerową wartością służącą do uniknięcia numerycznej niestabilności. Nieliniowy logarytm wiarygodności mapuje dystrybucję obiekt/tło jako wartości pozytywne dla barw związanych z pierwszym planem oraz jako wartości ujemne dla pikseli o barwach związanych z tłem [12].

Binaryzacja obrazu określona jest następującym wzorem:

$$M_i = \begin{cases} 1\ jeżeli\ L_i > \tau \\ 0\ \ jeżeli\ L_i \le \tau \end{cases}, \tag{4.2}$$

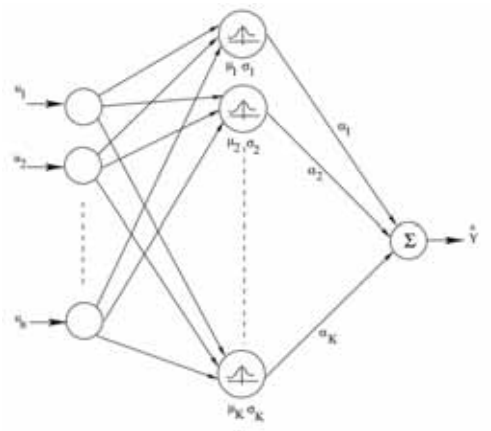
gdzie:
τ jest wartością progową decydującą o przynależności pikseli.

Kolejnym krokiem podczas tworzenia modelu obiektu zainteresowania jest wybór odpowiedniej przestrzeni cech. W tym celu dla każdego piksela określić można wartości reprezentujące informacje o jego barwie, bazując na różnych systemach określania barw (RGB, YCbCr, HSV). Wartością charakteryzującą piksel była również liczba określająca przynależność piksela do tła bądź obiektu (odpowiednio była to wartość −1 oraz +1).

Algorytm klasyfikacji oparty został na dwóch sieciach typu RBF:

- zadaniem pierwszej jest maksymalizacja poprawnej klasyfikacji pikseli należących do obiektu
- zadaniem drugiej jest maksymalizacja poprawnej klasyfikacji pikseli nienależących do obiektu (należących do tła).

Uproszczony schemat sztucznej sieci neuronowej zaprezentowany jest na rysunku 2.



Rys. 2. Uproszczona architektura sieci neuronowej typu RBF. Źródło: [12]

Problem klasyfikacji dwuklasowej może zostać przedstawiony w sposób następujący: para$(U_i, T_i), i = 1,2, \dots, N$ (N – liczba pikseli) przedstawia parę próbka–etykieta, gdzie U_i jest n-wymiarowym wektorem w przestrzeni cech a T_i jest etykietą jednej z dwóch klas. Zadaniem procesu klasyfikacji jest przypisanie etykiety dla nowego wektora w przestrzeni cech, z odpowiednią dokładnością.

Przy założeniu, że neurony ukryte realizują funkcję Gaussa o parametrach $\boldsymbol{\mu}_i$ (wektor wartości średnich w przestrzeni cech) oraz σ_i^2 (wariancja), wyjście z sieci można określić następującym wzorem:

$$\hat{y} = \sum_{j=1}^{K} \alpha_j \exp\left(-\frac{\left\|U-\boldsymbol{\mu}_j\right\|}{2\sigma_j^2}\right), \qquad (4.3)$$

gdzie α_j – wartości wag pomiędzy warstwą ukrytą a neuronami wyjściowymi, U – wektor wartości wejściowych z uwzględnieniem wag wejściowych. Postać macierzowa (4.3):

$$\hat{Y} = \Phi_K \alpha, \qquad (4.4)$$

gdzie:

$$\Phi_K(\boldsymbol{\mu}, \boldsymbol{\sigma}, \boldsymbol{U}) = \begin{bmatrix} \phi_1(\mu_1, \sigma_1, U_1) & \dots & \phi_K(\mu_K, \sigma_K, U_1) \\ \vdots & \ddots & \vdots \\ \phi_1(\mu_1, \sigma_1, U_N) & \dots & \phi_K(\mu_K, \sigma_K, U_N) \end{bmatrix}, \qquad (4.5)$$

ϕ – funkcja Gaussa, *i*-ty wiersz odpowiada *i*-temu ukrytemu neuronowi w odniesieniu do wartości wejściowych $U_1, U_2, \dots . U_N$. W praktycznych zastosowaniach liczba neuronów ukrytych jest znacznie mniejsza od ilości danych wejściowych.

Wartości parametrów funkcji Gaussa wybierane są losowo, natomiast wagi połączeń pomiędzy neuronami wyjściowymi oraz neuronami warstwy ukrytej wyznaczane są analitycznie. Jeżeli $\Phi_\alpha = T$, to wartości wag (α) obliczane są z wykorzystniem metody najmniejszych kwadratów jako [8]:

$$\alpha = \Phi_K^+ T, \qquad (4.6)$$

gdzie: $\Phi_K^+ = (\Phi_K^T \Phi_K)^{-1} \Phi_K^T$ jest pseudo-odwrotną macierzą Moore'a-Penrose'a; przyjmujemy $M_0 = (\Phi_K^T \Phi_K)^{-1}$.

Projektowanie klasyfikatora opartego na sieciach typu RBF składa się z dwóch faz. Postępowanie w pierwszej fazie, tzw. fazie rozwoju klasyfikatora, można zawrzeć w następujących krokach:

1. Przypisanie poszczególnym neuronom sieci typu RBF losowych wartości parametrów $\boldsymbol{\mu}_i$ oraz σ_i^2.
2. Obliczenie macierzy wyjściowej warstwy ukrytej Φ_K.
3. Estymacja wartości macierzy $\alpha = \Phi_K^+ T$.
4. Obliczenie dokładności klasyfikacji (w procentach) pikseli należących do obiektu oraz tła według wzorów:

$$\eta_o = 100 * \frac{Liczba\ pikseli\ poprawnie\ sklasyfikowana\ jako\ obiekt}{Liczba\ wszystkich\ pikseli} \qquad (4.7)$$

$$\eta_b = 100 * \frac{Liczba\ pikseli\ poprawnie\ sklasyfikowana\ jako\ tło}{Liczba\ wszystkich\ pikseli} \qquad (4.8)$$

5. Zapisanie wartości parametrów neuronów warstwy ukrytej, dla których η_o iη_b osiągnęły maksymalne wartości.

W przypadku fazy drugiej – fazy adaptacji *online*, dla każdej kolejnej pary (U_N, T_N) wykonywane są następujące czynności [12]:

1. Obliczony zostaje wektor wartości wyjściowych warstwy ukrytej:

$$\boldsymbol{\phi}_n = [\phi_1(\mu_1, \sigma_1, U_n)\ \ \phi_2(\mu_2, \sigma_2, U_n) \dots\ \phi_K(\mu_K, \sigma_K, U_n)]^T$$

2. Obliczone zostaje wyjście:

$$\hat{y} = \phi_n^T \alpha \qquad (4.9)$$

3. Następuje adaptacja wag wyjściowych α, korzystając z algorytmu najmniejszych kwadratów (ang. *RecursiveLeastSquares*).

$$M_0 = M_0 - \frac{M_0 \Phi_n \Phi_n^T M_0}{1 + \Phi_n^T M_0 \Phi_n}, \qquad (4.10)$$

$$\alpha = \alpha + M_0 \Phi_n (T_i - \hat{y}). \qquad (4.11)$$

Ze względu na to, iż szybkość działania algorytmu jest bardzo istotna, należy wybrać tylko nieliczne piksele z każdej kolejnej ramki w celu przeprowadzenia adaptacji wag (jedną z proponowanych metod jest wybranie pikseli tworzących spiralę Archimedesa).

5. Model obiektu i jego lokalizacja

Klasyfikatory oparte na sztucznych sieciach neuronowych są w stanie aproksymować prawdopodobieństwo *a posteriori* z dowolną dokładnością [11]. Etykiety dwóch klas oznaczone są jako +1 lub −1. Prawdopodobieństwo *a posteriori* piksela U_i, otrzymanego z wykorzystaniem klasyfikatora obiektu:

$$\hat{p}_o(U_i|c) = \frac{\max(\min(\hat{Y},1),-1)+1}{2}. \quad (5.1)$$

W podobny sposób obliczane jest prawdopodobieństwo *a posteriori* piksela U_i, otrzymanego z wykorzystaniem klasyfikatora pikseli nienależących do obiektu.

$$\hat{p}_b(U_i|c) = \frac{\max(\min(\hat{Y},1),-1)+1}{2}. \quad (5.2)$$

Model celu opracowywany jest przy użyciu tylko tych pikseli, które zostały sklasyfikowane poprawnie jako piksele należące do obiektu przez oba klasyfikatory (np. w klasyfikatorze obiektu piksele, które posiadają prawdopodobieństwo *a posteriori* większe niż 0.5, są zadeklarowane jako należące do klasy obiektu). W ten sposób otrzymujemy model celu w postaci:

$$\hat{p}_t(U_i|c) = \min(\hat{p}_o, 1.0 - \hat{p}_b). \quad (5.3)$$

Lokalizacja celu w poszczególnych ramkach sekwencji wideo jest osiągana poprzez iteracyjne poszukiwanie mody prawdopodobieństwa *a posteriori* estymowanego przez sieć neuronową typu RBF.

Położenie środka kandydata na cel w obecnej ramce jest początkowo takie samo jak to estymowane dla obiektu w poprzedniej ramce. Rozważmy *k*-tą iterację lokalizacji obiektu. Niech X_c^k będzie środkiem kandydata na cel, X_j^k – zbiorem pikseli kandydata na obiekt, $j = 1, \dots, N$, ze środkiem w punkcie X_c^k. W takim wypadku prawdopodobieństwo *a posteriori* *i*-tego piksela $(i \in j)\hat{p}_k(U_i|c)$ otrzymywane jest jako:

$$\hat{p}_k(U_i|c) = \min(\hat{p}_o(U_i|c), 1.0 - \hat{p}_b(U_i|c)). \quad (5.4)$$

Prawdopodobieństwo kandydata na cel podczas *k*-tej iteracji $(\hat{p}_k)$ otrzymywane jest poprzez testowanie cech otrzymanych z lokalizacji X_j^k. Nowa lokalizacja środka obiektu jest estymowana jako środek ciężkości prawdopodobieństw *a posteriori* $(\hat{p}_k)$ ważonych modelem celu $(\hat{p}_t)$, co przedstawia równanie (5.5):

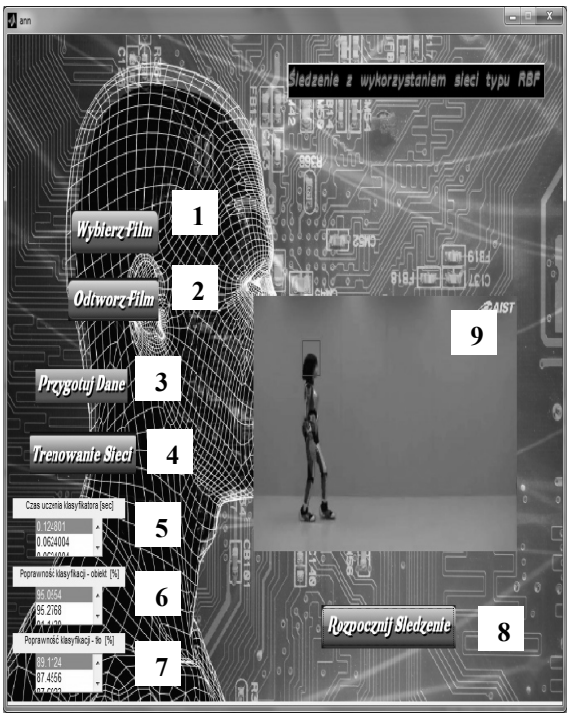
$$X_c^{k+1} = \frac{\sum_i X_i^k \hat{p}_k(U_i|c)\hat{p}_t(U_i|c)}{\sum_i \hat{p}_k(U_i|c)\hat{p}_t(U_i|c)}. \quad (5.5)$$

Operacja iteracji zostaje zakończona w momencie, gdy zmiana w lokalizacji środka ciężkości dla dwóch kolejnych iteracji spadnie poniżej określonej wartości t_s [12].

6. Graficzny interfejs użytkownika

W celu przeprowadzania symulacji przygotowany został autorski graficzny interfejs użytkownika (wykorzystujący środowisko programistyczne Matlab®) przedstawiony na rysunku 3. Poniżej opisane zostały poszczególne elementy okna oraz ich funkcje:

1. Przycisk *Wybierz Film* służy do wskazania ścieżki pliku wideo, z którego użytkownik będzie korzystał podczas testowania trakera.
2. Przycisk *Odtwórz Film* daje możliwość podglądu wybranego pliku.
3. Przycisk *Przygotuj Dane* wykonuje operacje oddzielenia wybranego obiektu od tła oraz przygotowuje odpowiednią macierz cech w celu zapisania obiektu w przestrzeni cech.
4. Przycisk *Trenowanie Sieci* służy do wytrenowania sztucznej sieci typu RBF.
5. Okienko wyświetlające czas, jaki był potrzebny do wytrenowania sieci z wykorzystaniem uprzednio przygotowanych danych w sekundach.
6. Okienko wyświetlające poprawność klasyfikacji pikseli jako należących do celu (%).
7. Okienko wyświetlające poprawność klasyfikacji pikseli jako należących do tła (%).
8. Przycisk *Rozpocznij Śledzenie* uruchamia proces śledzenia wybranego przez użytkownika obiektu.
9. Okno wyświetlające efekt działania trakera.



Rys. 3. Graficzny interfejs użytkownika.
Źródło: opracowanie własne

7. Podsumowanie

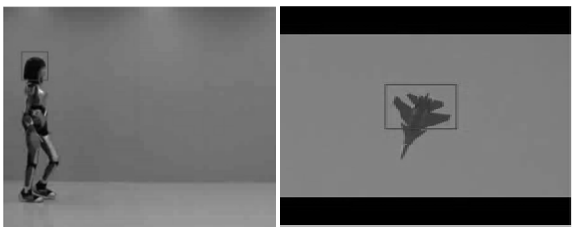
Podczas symulacji działania algorytmu porównaniu podlegały dwie jego konfiguracje:

- z adaptacją *online*
- bez adaptacji *online*.

W obu algorytmach liczba neuronów warstwy ukrytej sieci jest taka sama i wynosi 16. Wartość ta została wybrana eksperymentalnie – powyżej 16 neuronów w warstwie ukrytej nie zaobserwowano poprawy działania algorytmu. Symulacje przeprowadzone zostały pod kątem sprawdzenia odporności algorytmu na zakłócenia i problemy omówione we wprowadzeniu.

Symulacje działania algorytmu programu przeprowadzone zostały na dwóch sekwencjach wideo zaczerpniętych z portalu *youtube.com*:

- *robot.avi*, na którym śledzony będzie robot człekokształtny bądź jego część (rysunek 4a)
- *samolot.avi*, na którym śledzony będzie myśliwiec Su-30 (rysunek 4b).



a) b)

Rys. 4. Przykłady obiektów do śledzenia:
a) głowa robota człekokształtnego;
b) myśliwiec Su-30.
Źródło: opracowanie własne

Na podstawie sekwencji obrazów przygotowane zostały nowe, mające na celu dokładniejsze zbadanie właściwości algorytmów:

- oryginalny – w dobrej jakości
- zaszumiony – dodany został szum Gaussa o średniej 0 oraz wariancji 0.025
- zaszumiony i rozmazany – poszczególne klatki zaszumionej sekwencji wideo zostały poddane operacji rozmazania (ang. *blur*) z wykorzystaniem dolnoprzepustowego filtru Gaussa o rozmiarze 10 x 10 oraz standardowym odchyleniu równym 10.

Podczas badania działania obu algorytmów na tak przygotowanych sekwencjach obrazów nie było różnicy w przeprowadzanym procesie detekcji oraz śledzenia obiektów – żadna z metod nie miała problemów z poprawnym wykonaniem tych operacji.

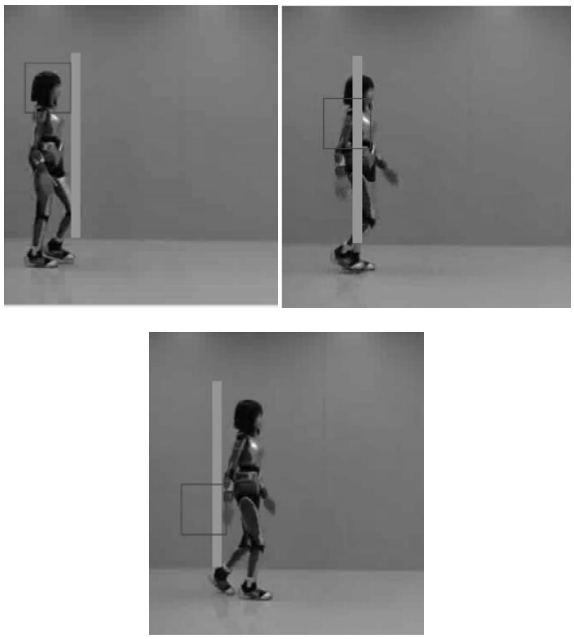
Różnica w działaniu pojawiła się w momencie przesłonięcia celu. Aby zbadać zachowania trakera w przypadku częściowego przesłonięcia śledzonego obiektu, do sekwencji obrazów przedstawiających ruch robota dodany został obiekt, który będzie wprowadzał zakłócenie. Obiekt ten to prostokąt o wymiarach 220 x 10 mający na celu przesłonięcie ok. 30% badanego obiektu. W tym przypadku algorytm bez adaptacji wykazał się niewielką poprawnością działania. Mimo dobrych parametrów wytrenowania sieci:

- poprawność klasyfikacji pikseli należących do obiektu na poziomie 92%
- poprawność klasyfikacji pikseli należących do tła na poziomie 93%,

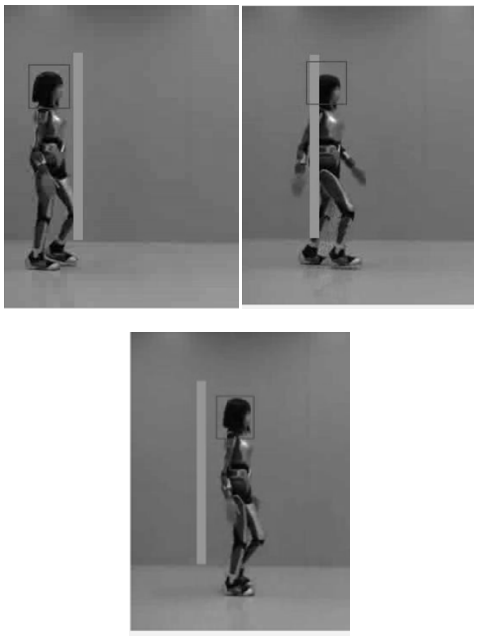
traker gubił cel po jego przejściu przez przesłonę (rysunek 5). Algorytm śledzący z adaptacją *online* po przejściu celu przez przesłonę śledził go dalej poprawnie (rysunek 6).

Dopiero w przypadku powiększenia dwukrotnie szerokości prostokąta–przesłony (do rozmiaru 220 x 20) algorytm z adaptacją nie podążał za obiektem po przejściu przez

przeszkodę. Przesłona zajmowała ponad 50% powierzchni śledzonego obiektu.



Rys. 5. Rezultat działania algorytmu bez adaptacji w przypadku wystąpienia częściowego przesłonięcia celu.
Źródło: opracowanie własne



Rys. 6. Rezultat działania algorytmu z adaptacją w przypadku częściowego przesłonięcia celu.
Źródło: opracowanie własne

Powyższe rozważania wskazują na wyższość trakera opartego na sztucznych sieciach neuronowych typu RBF z adaptacją *online* nad algorytmem niewykorzystującym procesu adaptacji. Pomimo poprawnego śledzenia obiektów w przypadku wystąpienia szumu Gaussa lub rozmycia obrazu z wykorzystaniem filtru dolnoprzepustowego Gaussa, prostszy z algorytmów nie był wstanie poradzić sobie z częściowym przesłonięciem.

Istotne jest spostrzeżenie, że algorytm oparty na sieciach neuronowych nie jest wrażliwy na zmiany pozycji (rotacji) obiektu śledzonego, jego oświetlenia (co pociąga za sobą zmianę barwy obiektu).

Co więcej, zaprezentowana sieć neuronowa, z racji niewielkich wymagań obliczeniowych i braku potrzeby długiego jej trenowania, może być stosowana w systemach czasu rzeczywistego. W połączeniu z odpowiednimi algorytmami rozpoznawania wzorców, badana metoda adaptacyjna śledzenia może zostać zaimplementowana np. w systemie detekcji oraz śledzenia zagrożeń (ang. *Video Threat Detection Systems*).

8. Bibliografia

[1] J. Ahmed, M.N. Jafri, J. Ahmad, M.I. Khan, ”Design and Implementation of a neural Network for Real-Time Object Tracking”, *World Academy of Science*, *Engineering and Technology*, 6/2005.

[2] A. Ameljańczyk, *Optymalizacja wielokryterialna w problemach sterowania i zarządzania*, Ossolineum, Wrocław, 1984.

[3] A. Ameljańczyk, „Matematyczne aspekty modelowania pajęczynowego obiektów”, *Biuletyn Instytutu Systemów Informatycznych*, Nr 4, str. 9–16 (2009).

[4] A. Ameljańczyk, ”Multicriteria similarity models in medical diagnostics support algorithms”, *Bio-algorithms and Med-Systems*, Vol. 21, No. 1 (2013).

[5] A. Bartkowiak, *Sieci Neuronowe. Notatki Rozszerzone ANN08*, Uniwersytet Wrocławski, grudzień 2008.

[6] C.M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, Oxford, 2005.

[7] Z. Ghahramani, *Learning Bayesian Networks*, Department of Computer Science, University of Toronto, Canada, October 1997.

[8] G.B. Huang, Q.Y. Zhu, C.K. Siew, ”Extreme learning machine: Theory and applications”, *Neurocomputing* 70 (2006).

[9] E. Maggio, A. Cavallaro, *Video Tracking Theory and Practice*, Wiley, 2011.

[10] S. Osowski, *Sieci Neuronowe w Ujęciu Algorytmicznym*, Wydawnictwa Naukowo-Techniczne, Warszawa 1997.

[11] R. Rojas, *A short proof of the posterior probability property of classifier neural networks*, Institut für Informatik, Freie Universität, Berlin 1996.
[12] R. Venkatesh Babu, S. Suresh, A. Makur, ”Online adaptive radial basis function networks for robust object tracking”, *Comput. Vis. Image Understand*., 2009.

Artificial neural networks with radial basis functions for video object tracking

J. SZYMONIK

The main problem considered in this article was the artificial neural network design for target localization and target tracking in video sequence, with the use of Matlab environment. What is more, the algorithm resistance to noise and disturbances that may occur was studied. The article presents the architecture of artificial neural network with radial basis functions. The algorithm for tracking as well as the method for target modeling and localization with the use of the above network architecture is shown. In the summary there are results of conducted simulations in Matlab of video trackers based on artificial neural networks.

Keywords: object tracking, artificial neural networks, radial basis functions.