A Novel Computational Intelligence-based Approach for Medical Image Artifacts Detection

Beibei Cheng[a], R. Joe Stanley[a], Sameer Antani[b], George R. Thoma[b]
[a]*Department of Electrical and Computer Engineering*
Missouri University of Science and Technology
Rolla, MO 65409-0040
[b]*Lister Hill National Center for Biomedical Communications*
National Library of Medicine, National Institutes of Health, DHHS
Bethesda, MD 20894, USA
bcx93@mst.edu, stanleyj@mst.edu, santani@mail.nih.gov, gthoma@mail.nih.gov

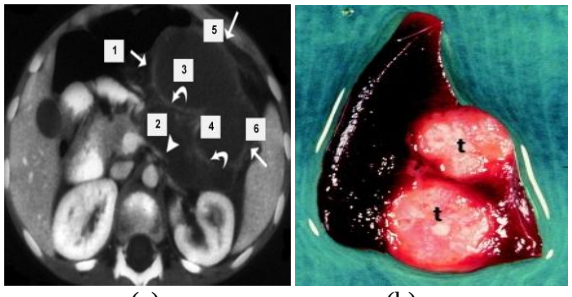
Abstract

In this research, a novel computational intelligence-based algorithm to detect artifacts, specifically arrows, in medical images is presented. Image analyses techniques are developed to find the symbols and text automatically. Features are computed from the shape of arrow for the discrimination of arrows from other artifacts. We investigate a biologically-inspired reinforcement learning (RL) approach in an adaptive critic design (ACD) framework to apply Action Dependent Heuristic Dynamic Programming (ADHDP) for arrow discrimination based on the computed features. Experimental results for ADHDP are compared with feed forward multi-layer perception (MLP) back-propagation artificial neural networks (BP-ANN), particle swarm optimization (PSO) for training of a MLP neural network, genetic algorithm (GA) for training of a MLP neural network, k-nearest neighbor (KNN), and support vector machine (SVM).

1. Introduction

The detection of medical image artifacts such as arrows is important to highlighting supplemental and context-based information which may be helpful in understanding medical images. It is necessary to have an accurate algorithm in discriminating arrows from other characters and symbols. There are several methodologies that have been implemented to find arrows in previous research. Laurent Wendling and Salvatore Tabbone [1] proposed a method in recognizing arrows based on the aggregation of geometric criteria using the choquet interal; J.E. den Hartogtz and T.K. ten Katet [2] gave a solution of finding arrows in utility maps using a neural network; Jongan Park, Waqas Rasheed, and Junguk Beak [3] proposed a way of identifying arrow signs for Robot Navigation using a camera-based method.

Extending techniques from previous research, the arrow symbols present in the medical images in our experimental data set have variety in shape and size. Arrows do not necessarily have to be straight (arrow 3, arrow 4) and the shape of the arrows can change (arrow 2) as you can see in Figure 1(a). Furthermore, a noise example can include characters and symbols, which may be of similar size to arrows as shown in Figure 1 (b). Therefore, a general and robust arrow detection algorithm is needed for discrimination from other medical image artifacts.



(a) (b)

Figure 1. Medical image examples.
(a) Arrow image example. (b) Noise image example.

Since both text and symbol objects are white or black, they can be segmented by some image analysis techniques. After generating the binary image containing only text-like and symbol-like objects, features sets are used as input to classifiers. An overview of the algorithm investigated is shown in Figure 2. This study uses 144 medical images annotated by modality (radiological, photo, etc.) selected from 2004-2005 issues of the British Journal of Oral and Maxillofacial Surgery, including 79 images with one or more arrows and 65 images with no arrows. The image analysis techniques are follows:

1) Convert RGB images into gray images and inverted gray images.
2) Use Otsu's method [4], which chooses the threshold to minimize the intra-class variance of the black and white pixels to convert gray images into binary images.

3) Remove objects that are considered small and objects that are considered short.
4) Get the edge of object with gray drop. The value of gray drop is 30. If the absolute value of center pixel minus NW, N, NE, W, E, SW, S, SE is greater than the gray drop, this pixel will be marked. (Figure 3)
5) Compare the edge image getting after step 4) with the image getting after step 3), keep the objects with the same bounding box size.
6) Or image result with inverted image result.

Figure 4 presents an image example of the image processing steps for the original image to generate the binary mask for feature calculations.

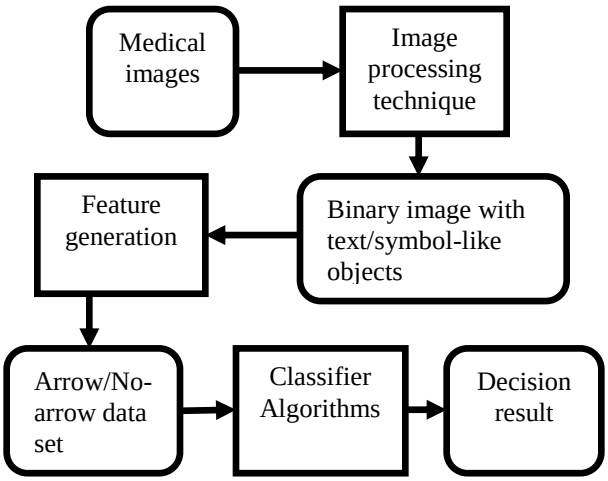


Figure 2. Overview of arrow detection and discrimination process.

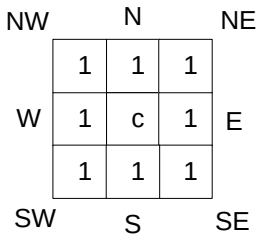
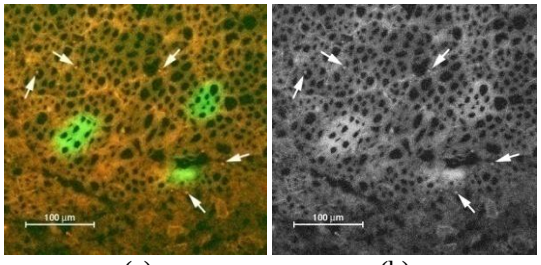
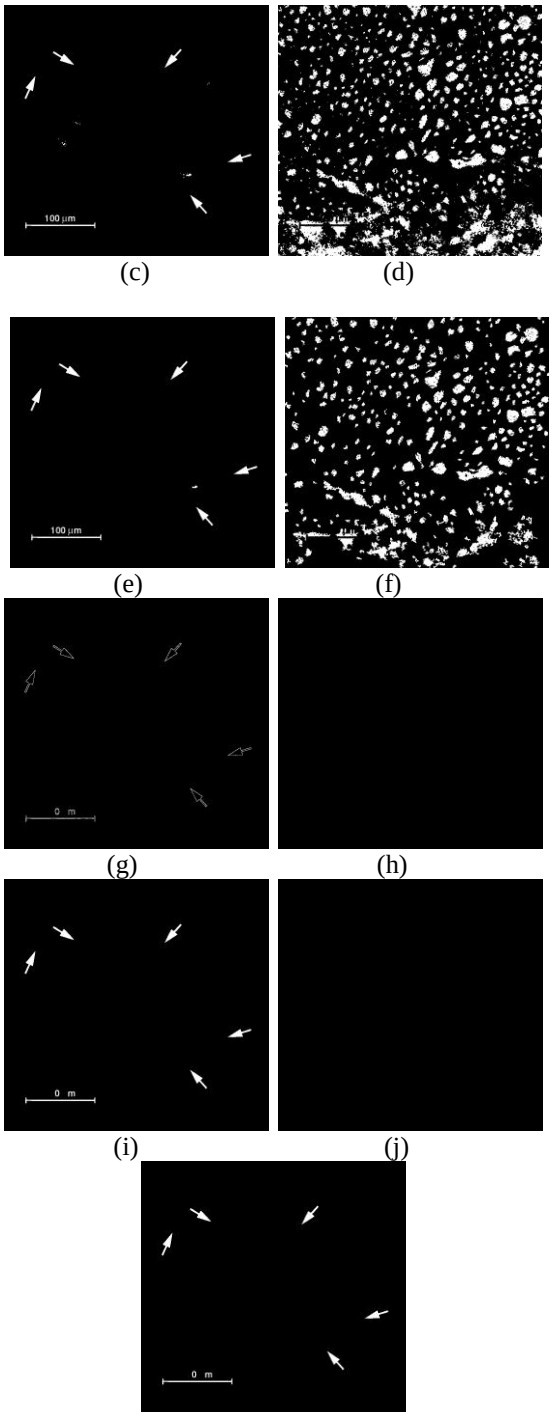


Figure 3. Edge detection.



(a) (b)



(c) (d) (e) (f) (g) (h) (i) (j) (k)

Figure 4: Image processing algorithm example. (a) Original image. (b) Grey image. (c) Image by using threshold. (d) Inverted image by using threshold. (e) Image after noise removal. (f) Inverted image after noise removal. (g) Image by using edge detection. (h) Invert image by using edge detection. (i) Image by comparing (e) to (g) with the bounding box size. (j) Image by comparing (f) to (h) with the bounding box size. (k) Final image by or (i) and (j).

The remaining sections of this paper include: 1) Feature Generation, 2) Adaptive Critic Designs Methodology, 3) Other Classification Methodologies, 4) Results and Discussion, 5) Acknowledgement, and 6) Conclusions and Future Work.

2. Feature generation

After generating the binary image containing only text-like and symbol-like objects by using above techniques, certain features are selected that will help in the identification of arrows from the rest of objects. This selection process is very important, because the identification stage depends heavily upon this process. We select twenty-two features that will help us in distinguishing the arrows. These characteristic features and their explanations are shown as follows:

MajorAxisLength: length (in pixels) of the major axis of the ellipse that has the same normalized second central moments as the region.

MinorAxislength: length (in pixels) of the minor axis of the ellipse that has the same normalized second central moments as the region.

Axis Ratio: ratio of MajorAxisLength with MinorAxislength.

Normalized area: area of the region divided by the whole image.

Solidity: area of the region divided by the Convexhull Area.

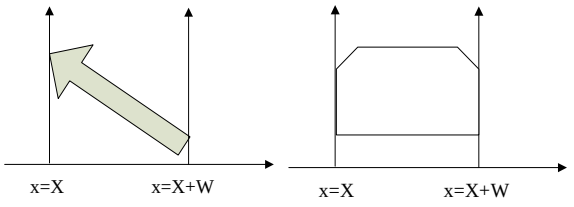
EulerNumber: equal to the number of objects in the region minus the number of holes in those objects.

EquiDiam: the diameter of a circle with the same area as the region.

Extent: ratio of area to bounding box area.

AvgSkelDist: average width of object.

MinPixelNo: for two line x=X and x=X+W (X is the top left point of bounding box horizontal value; W is the width of bounding box), the minimum number of pixels of intersection for region and each line as shown in Figure 5.



(a) (b)

Figure 5: MinPixelNo feature. (a)Arrow. (b) Noise.

Weighted Density Distribution features: other arrow features are extracted by correlating the shape samples of the arrow with weighted density distribution functions (WDD) [5-6]. Let $E = \{E(s_1), E(s_2), \dots, E(s_m)\}$ be the sequential of shape samples collected, where m is the number of samples collected at a constant rate and $E(s_i) \in R_n$, where $1 \le i \le m$. Figure 6 shows the WDD functions used in the experiments. Twelve WDD-based features are computed. Each of the WDD function is decomposed into 12 discrete points for WDD feature calculations. Let W_1 denote the WDD function in figure 6(a), W_2 denote the WDD function in figure 6(b) and so on. The gray horizontal position marker for each WDD function shown in figure 6 points to the reference position for which WDD features is computed. The reference position corresponds to the current sample s. Six WDD features ($f_1(s), \dots, f_6(s)$) corresponding to the measurement at sample s are computed according to the following expression

$$f_k(s) = \sum_{i=s-10}^{s+1} E(i) W_k\,(i - s + 11) \qquad (1)$$

For $k = 1,2,\dots,6$. Six additional features ($f_7(s), \dots, f_{12}(s)$) are computed by correlating the six WDD functions with the sequence of absolute differences between samples value as follows

$$f_k(s) = \sum_{i=s-10}^{s+1} |E(i) - E(i-1)| W_k\,(i - s + 11) \qquad (2)$$

In this research, the shape samples of arrow are computed in the following way: divide the height of the bounding box of arrow by twenty and get line $y=Y_1$ $y=Y_2 \dots$, $y=Y_{20}$. The 20 samples is the length of line $y=Y_1$ to $y=Y_{20}$ intersected with this arrow, as you can see in figure 7, the red lines. After all, twelve WDD features could be generated according to these samples.

Therefore, with the 144 medical images as the input, after image processing and feature extraction, there are 154 arrow objects and 276 text/noise objects generated. They are manually catalogued with class 1 (arrow objects) and class 0 (other objects). To evaluate these features, the attribute selection criteria are information gain [7] and chi-square [8] by Weka®. Table 1 shows information gain value and chi-square value for each feature.

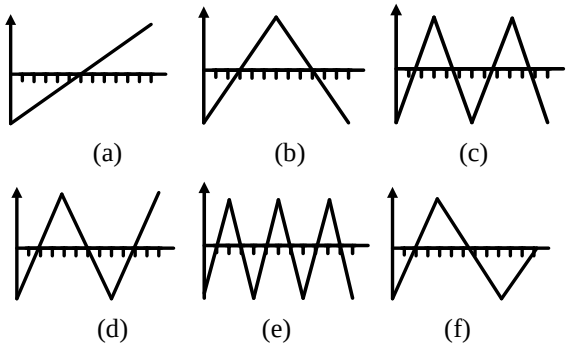


Figure 6: The WDD functions used to compute arrow features [6].

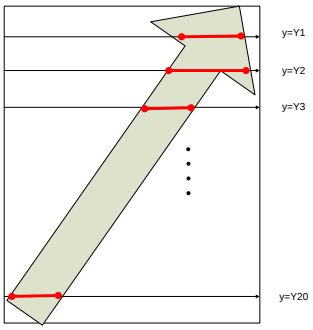


Figure 7: Samples for generating WDD features.

The orientation of each object can impact feature generation. For example, the feature MinPixelNo is supposed to be smaller for arrow objects than for others because of the accurate point of arrow. But since the various shapes and sizes of arrows, as can be seen in Figure 8, this value can still be quite large. This arrow is falsely identified as noise by the MLP backpropgation neural network.

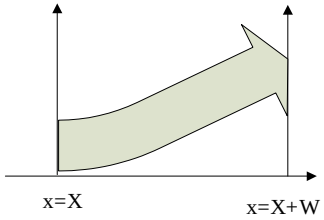


Figure 8: Falsely identified arrow.

To improve the outcome, rotating the region is an effective solution. Figure 9 shows the arrow image output with different rotation values (0 degree, 15 degree, 30 degree, 45 degree, 60 degree, 75 degree, 90 degree). Therefore, seven data sets will be generated after applying feature extraction to these seven image sets.

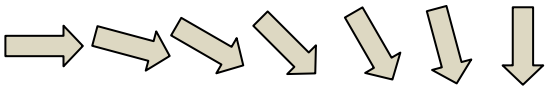


Figure 9: Arrows with different rotation value.

An ACD-based arrow discrimination methodology is then applied in the presence of these seven data sets. So the decision of arrow/no-arrow is made by the global data set but not the single data set. The input for the neural network is the features generated from those seven different orientations based on rotation.

Table 1: Information gain and chi-square value.

Feature_Name	Info_gain_value	Chi_square_value
MajorAxisLength	0.5875	287.0769
MinorAxislength	0.1924	106.3261
Axis Ratio	0.4380	217.0534
Normalized area	0.5052	248.0459
Solidity	0.1038	58.169
eulerNumber	0.1587	73.3162
EquiDiam	0.5628	278.9818
Extent	0.2260	112.2445
AvgSkelDist	0.4566	230.6459
MinPixelNo	0.1555	75.0847
f1	0.1837	109.0502
f2	0.2504	141.9929
f3	0.2768	155.7666
f4	0.1658	95.7217
f5	0.0457	27.9648
f6	0.4004	213.7316
f7	0.3361	187.9629
f8	0.1205	71.3723
f9	0.1350	81.8844
f10	0.0538	32.9075
f11	0.1837	25.5232
f12	0.2768	159.0519

3. Adaptive critic design methodology

Reinforcement learning is the problem faced by an agent that must learn behavior through trial-and-error interactions with a dynamic environment. It is a computational approach to learning whereby an agent tries to maximize the total amount of reward it receives when interacting with a complex, uncertain environment [9]. The RL has been developed in various applications such as neuro-computing [10], and multi-resolution object recognition [11].

The Adaptive Critic Design provides a workstation for implementing RL. An ACD approximates the neuro-dynamic programming by using an action and a critic network, respectively [12]. This model employs reinforcement learning (RL) through direct neural

dynamic programming (Direct NDP) [13]. The term "direct" is influenced by the adaptive control literature where "direct adaptive control" means no plant model, and thus no plant parameter estimation takes place but instead certain plant information is used directly to find appropriate and convergent control laws and control parameters, which is required in this research. Direct NDP is a model independent approach to action dependent heuristic programming (ADHDP).

Figure 10 shows the model of ADHDP used in this study, which is based on the model in [12]. In the current problem setting, let the discounted total reward to go R(t) at time t be given by

$$R(t) = r(t+1) + \alpha r(t+1) + \cdots = \sum_{k=1}^{\infty} \alpha^{k-1} r(t+k) \quad (3)$$

Where the function of r(t) is the reinforcement value at time t, and α is a discount factor between 0 and 1.

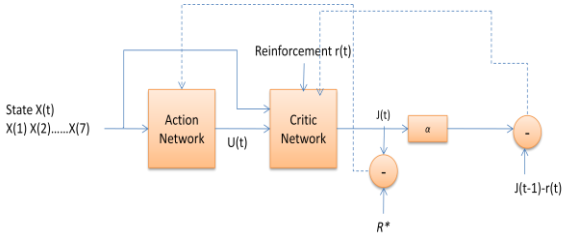


Figure 10: Schematic diagram of ADHDP.

The critic network is used to provide an output J(t), which is an approximation for R(t), the weighted total future reward to go. The reward function R(t) at time t is given by Eq. (3).

We define the prediction error, and consequently the Bellman error, for the critic element as

$$e_c(t) = \alpha J(t) - [J(t-1) - r(t)] \quad (4)$$

and the objective function to be minimized in the critic network is

$$E_c(t) = \frac{1}{2} e_c^{\,2}(t) \quad (5)$$

In the action network, the weight update in the action network can be formulated as follows.

$$e_a(t) = J(t) - R^* \quad (6)$$

The principle in adapting the action network is to backpropagate the error between the desired ultimate performance objective, denoted by R*, and the approximate function J from the critic network. Since r_s has been defined as the reinforcement signal for "success," R* is set to r_s/(1-α) has in the direct NDP design paradigm and in subsequent case studies. In this paper, r_s is set to be zero for simplification.

An artificial neural network is chosen for implementation of the action and critic networks. The structure of the neural networks for both the action and critic networks are implemented as a multi-layer feed forward (MLP) neural network. It consists of the input layer, the hidden layer and the output layer. The hidden layer neurons have a sigmoid transfer function while other layers have linear neurons. For the action network, the architecture is 23x5x1, with twenty-two features and a bias in the input layer, five nodes in the hidden layer and one output layer. For the critic network, the architecture is 24x5x1, with twenty-two features, a bias and the output from action network in the input layer, five nodes in the hidden layer and one output layer.

A ten-fold cross validation methodology is used for training/test set generation for the neural network [14]. The neural networks are trained up to 1000 epochs, using online (Stochastic/Delta) learning. In this case, the next input pattern is selected randomly from the training set, to prevent any bias that may occur due to the sequences in which patterns occur in the training set. For each training feature, 7 different data states (original image feature data set and its six different rotated orientation feature data sets) are applied as the input one by one for both the action network and the critic network to update the weights. If the difference between action network output u(t) and the target is less than 0.5, the reinforcement signal r(t) takes the reward "0", otherwise, r(t) takes the punishment "-1". The learning rates for both critic and action network are set to be 0.001. The discount factor α is set to be 0.1. The test set is the original image feature data set only.

With the target value for the arrow data set to 1 and the no-arrow data set to 0, action network outputs after testing are between -1 and 1. Receiver operating characteristic (ROC) curves are generated for classification results based on the neural network outputs obtained for the ten-fold cross cases [15]. The ROC curve is a plot of the sensitivity for a binary classifier system as its discrimination threshold is varied. The ROC curve represents equivalently the fraction of true positives versus false negatives rate.

4. Other classification algorithms

4.1. MLP backpropagation neural network

A multilayer perception backpropagation neural network is investigated for arrow discrimination [16]. Sigmoid transfer functions are used in the hidden layers, and a linear transfer function is used in the input and output layer, the neural network architecture is 23x5x1. The neural networks are trained up to 1000 epochs, using

online (Stochastic/Delta) learning, ROC curves are generated by the ten-fold cross strategy.

4.2. Particle swarm optimization (PSO) for training of a MLP neural network

In swarm intelligence algorithm [17], each particle has random velocity and memory that keeps track of previous best position and corresponding fitness. The previous best value of the particle position is called the 'pbest'. It has another value called 'gbest', which is the best value of all the 'pbest' positions in the swarm. The basic concept of PSO is that each particle in the swarm move toward its pbest and gbest locations at each time step. The basic concept of PSO is that each particle in the swarm move toward its pbest and gbest locations at each time step. (Figure 11)

In this research, PSO is used to train a MLP neural network, which has the same architectures and training epochs as the last one. To train this neural network, all neuron weights are together to contribute one of the particles. Each particle is updated toward the global best position, which will minimize the difference between the neural network output and target value. Detail of this algorithm is described in [18].

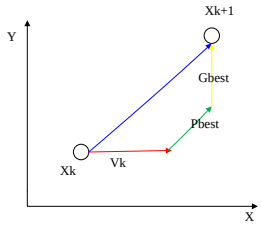


Figure 11: Basic concept of PSO.

4.3. Genetic algorithm (GA) for training of a MLP neural network

Genetic Algorithm is a kind of optimization by using selection, crossover, mutation and elitism operators [19]. This MLP neural network still has the same architectures and training epochs as usual. In the training procedure, all neuron weights are put together as the parents firstly, after applying the selection, crossover and mutation operators, offspring could be generated. The next offspring is chosen based on whether parent or its offspring minimizes the difference between the neural network output and target value.

5. Result and discussion

Adaptive Critic Designs is applied for the arrow classification. The output is compared with MLP backpropgation neural network, PSO for training of a MLP neural network, GA for training of a MLP neural network. Ten-fold cross testing strategy is implemented for all of them. ADHDP and other three neural networks are all build on the same twenty-two input features, use the same number of training epochs (1000). Since the output of the neural network is not an accurate class number, the ROC curve is generated as the comparison criteria. AUC represent the area under the ROC curve. Figure 12 shows the ROC curves and AUC result for these four algorithms.

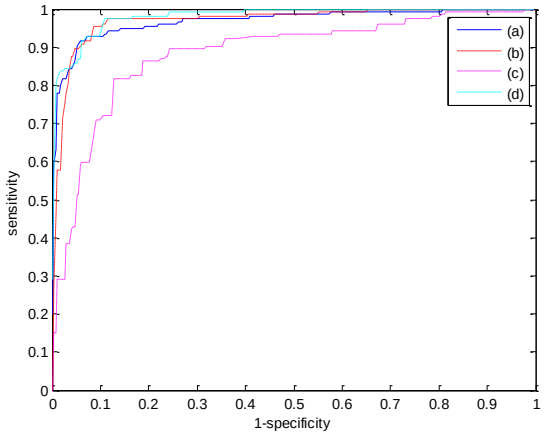


Figure 12: ROC curve and AUC (area under curve) for neural networks. (a) MLP backpropgation NN. AUC=0.9672. (b) PSO based NN. AUC=0.9681. (c) GA based NN. AUC=0.8839. (d) Direct NDP. AUC=0.9790.

In addition to comparing the discrimination results from the different classifiers using AUC, as given in Figure 12, we used a highest true positive rate and highest true negative rate with minimum difference between them from the ROC curves. Therefore, the true positive rate and true negative rate for (a), (b), (c), (d) are 91.53% and 91.56%, 91.53% and 92.77%, 87.66% and 82.97%, 92.86% and 92.03%.

Based on the AUC and the true positive and true negative comparison, we found that adaptive critic design achieved the best results among the different classifiers investigated. In direct NDP, the critic network is used as the performance evaluation; the reward of reinforcement learning is helping to update its weight and the output J(t). Therefore, the action network could update its weight based on total seven state feature data instead of only one state feature for other techniques. The global feature data improves the accuracy of discriminating arrow or no-arrow.

Furthermore, K-nearest neighbor (KNN) [20] and Support Vector Machine (SVM) [21] are also applied to the same data set. In KNN algorithm, generally Euclidean distance function is used to calculate distance between two points. K is set to be fifteen. In SVM algorithm, polynomial is used as kernel function value to train the data. The true positive rate and true negative rate for

KNN and SVM are 93.50% and 88.76%, 90.90% and 93.03%, respectively.

6. Acknowledgment

This work was supported by NLM under contract number 276200800413P and the Intramural Research Program of the National Institutes of Health (NIH), NLM, and Lister Hill National Center for Biomedical Communications (LHNCBC).

7. Conclusions and future work

This paper introduces the ACD design to image recognition using ADHDP algorithm. The result is very promising. ADHDP demonstrates greater strength and superiority than existing methods. In addition, the features extracted as the input of different classifiers are approved to be significantly useful.

The future work will include finding the exact orientation of each arrow. In addition, other adaptive critic design algorithms such as action dependent dual heuristic dynamic programming (ADDHP) or action dependent globalized dual heuristic dynamic programming (ADGHP) could be implemented as comparison.

References

[1] L. Wendling, S. Tabbone, “Recognition of arrows in line drawings based on the aggregation of geometric criteria using the Choquet integral”, Document Analysis and Recognition, Proceedings of Seventh International Conference on, 2003, vol. 1, pp. 299-303.

[2] J.E. den Hartogtz, T.K. ten Katet, “Finding arrows in utility maps using a neural network”, Proceedings of the 12th IAPR International Conference on Pattern Recognition, Conference B: Computer Vision & Image Processing, 1994, vol. 2, pp. 190-194,

[3] J. Park, W. Rasheed, and J. Beak, “Robot Navigation Using Camera by Identifying Arrow Signs”, Grid and Pervasive Computing Workshops, GPC Workshops '08, The 3rd International Conference on Grid and Pervasive Computing, Kunming, May 2008, pp. 382-386.

[4] M. Sezgin, B. Sankur, "Survey over image thresholding techniques and quantitative performance evaluation", Journal of Electronic Imaging, 2003, vol. 13, no. 1, pp. 146-165.

[5] J. Piper, E. Granum, “On fully automatic feature measurement for banded chromosome classification”, Cytometry, 1989, vol. 10, no. 3, pp. 242-255.

[6] R.J. Stanley, W.V. Stoecker, R.H. Moss, H.S. Rabinovitz, A.B. Cognetta, G. Argenziano, and H.P. Soyer, “A basis function feature-based approach for skin lesion discrimination in dermatology dermoscopy images”, Skin Research and Technology, 2008, vol. 14, no. 4, pp. 425-435.

[7] T.M. Mitchell, *Machine Learning*, the Mc-Graw-Hill Companies, Inc, 1997.

[8] A. Mood, A.G. Franklin, and C.B. Duane, *Introduction to the Theory of Statistics*, McGraw-Hill Companies, Inc, 1974.

[9] R.S. Sutton, A.G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, 1998.

[10] R.S. Sutton, A.G. Barto, and R.J. Williams, “reinforcement learning is direct adaptive optimal control”, Proceedings of IEEE American Control Conference, Boston, 1991, pp. 2143-2146.

[11] K.M. Iftekharuddin, Y. Li, “A biologically-inspired computational model for transformation invariant target recognition”, IEEE International Joint Conference on Neural Networks, Hong Kong, 2008, pp.1049 – 1056.

[12] D.V. Prokhorov, D.C. Wunsch, “Adaptive Critic Designs”, IEEE Transactions on Neural Networks, 1997, vol. 8, no. 5, pp. 997-1007.

[13] J. Si, A.G. Barto, W.B Powell, D.C. Wunsch, *Handbook of Learning and Approximate Dynamic Programming*, Wiley-IEEE Press, 2004.

[14] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection", Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, 1995, vol.14, pp. 1137–1143.

[15] J. Fogarty, R.S. Baker, and S.E. Hudson, “Case studies in the use of ROC curve analysis for sensor-based estimates in human computer interaction”, Proceedings of Graphics interface, Canadian Human-Computer Communications Society, School of Computer Science, University of Waterloo, Waterloo, Ontario Victoria, British Columbia, 2005, vol. 112, pp. 129-136.

[16] S. Haykin, *Neural Networks: A Comprehensive Foundation*, Prentice Hall, 1998.

[17] J. Kennedy, R. Eberhart, “Particle swarm optimization”, Proceedings of the IEEE International Conference on Neural Networks, Piscataway, NJ, 1995, pp. 1942-1948.

[18] G. Guidse, G. K. Venayagamoorthy, “Comparison of Particle Swarm Optimization and Backpropagation as Training Algorithms for Neural Networks”, Swarm Intelligence Symposium, 2003, pp. 110-117

[19] J.H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, 1975.

[20] T.M. Cover, P.E. Hart, "Nearest neighbor pattern classification," IEEE Transactions on Information Theory, 1967, vol. 13, no. 1, pp. 21-27.

[21] D. Meyer, F. Leisch, and K. Hornik, “The support vector machine under test”, Neurocomputing, 2003, vol. 55, pp. 169-186.