Engineering Journal of University of Qatar, Vol. 9, 1996, p.133-146

A SELF-ORGANIZING MULTISENSOR FUSION CLASSIFICATION ALGORITHM

Rustom Mamlook
Department of Electrical and Computer Engineering
Applied Science University, Jordan - 11931, Amman.

and

Wiley E. Thompson
Professor, Department of Electrical and Computer Engineering
New Mexico State University, New Mexico 88003, USA.

ABSTRACT

A self-organizing multisensor fusion algorithm to classify the inputs (data or images) into classes (targets, backgrounds) is presented. The algorithm forms clusters and is trained without supervision. The clustering is done on the basis of the statistical properties of the set of inputs. The algorithm is a self-organizing algorithm, since it has the ability to form and adjust the number of clusters without being given the correct number of clusters. This algorithm implements a clustering algorithm that is very similar to the simple sequential leader clustering algorithm and the Carpenter/Grossberg net algorithm (CGNA). The algorithm differs from CGNA in that (1) the data inputs and data pointers may take on real values, (2) it features an adaptive mechanism for selecting the number of clusters, and (3) it features an adaptive threshold. The algorithm does not require the number of classes been known apriori. The problem of threshold selection is considered and the convergence of the algorithm is shown. An example is given to show the application of the algorithm for multisensor fusion for classifying targets and backgrounds, and the results of using this algorithm is compared to the results of using K-nearest neighbor algorithm.

INTRODUCTION

One of the very important, fundamental problems in computer vision is the classification problem. Many classification algorithms have been developed (1-14) over the past several years using a variety of approaches and implementations. Some of the problems facing a number of the current classification algorithms include the requirement of supervision, threshold selection, and computation time.

A self-organizing multisensor fusion classification algorithm for classifying multitarget, multi-background images is presented here. The algorithm is a self-organizing algorithm, since it forms clusters and is trained without supervision. It uses multi-sensors to improve overall performance and provides a method for adaptively selecting the thresholds. The algorithm gives solutions to some of the problems facing the current classification algorithms include the number of classes being known apriori, the requirement of supervision, threshold selection, and computation time.

A Self-Organizing Multisensor Fusion Classification Algorithm

The multisensor fusion classifier (MFC) algorithm presented here is a self organizing algorithm that forms clusters and is trained without supervision (see Box 1). This clustering is done on the basis of the intrinsic statistical properties of the set of inputs. The algorithm implements a clustering algorithm that is very similar to the simple sequential leader clustering algorithm described in (1) and the Carpenter/ Grossberg net algorithm (CGNA) described in (5).

The algorithm starts by selecting the first cluster data pointer vector (FCDPV). This cluster data pointer (CDP) is a vector that has a size equal to the number of sensors which the input data comes from, and element i of (FCDPV) is related to the mean of the data coming from sensor i by a linear function. Element i of the input vector represents a data point from sensor i, and the size of the input vector is equal to the number of sensors used to collect the input data. The first input vector is compared to the FCDPV, and it is clustered with the first cluster if the distance to the FCDPV is less than a chosen threshold. Otherwise it is the data pointer for a new cluster. This process is repeated for all following input vectors. The algorithm differs from CGNA in that (1) the data inputs and data pointers may take on real values, (2) it features an adaptive mechanism for selecting the number of clusters, and (3) it features an adaptive threshold. These features and their implementation in the MFC algorithm are detailed in the following:

(1) The elements of both inputs and stored cluster data pointers take on not only the values 0 and 1 as in CGNA but also any positive real values.

(2) The number of clusters, which depends on both the distance metric used to compare inputs to cluster data pointers and the threshold, is adjusted so it does not grow with time. The adjustment of the number of clusters is done by adjusting the number of cluster data pointers which is adjusted as such

$$V_j(k) = g(threshold,\ d_{kj})\ V_j(k-1), \quad j=1,2,\ldots,nc \tag{1}$$

if

$$V_j(k) \geq n,\ then\ keep\ cluster\ data\ pointer\ \#\ j \tag{2}$$

else

$$disregard\ cluster\ data\ pointer\ \#\ j \tag{3}$$

where $V_j(k)$ denotes the number of input vectors in cluster j at stage k, d_{kj} denotes the distance between the input vector at stage k (which is in cluster j) and the cluster data pointer #j, n is a chosen integer number which represents the minimum number of input vectors that must be clustered within cluster j, g(.) is a nonlinear decreasing function, and nc is the number of clusters classified so far.

(3) The threshold of this algorithm ranges as such

$$\frac{1}{N}\sum_{i=1}^{N} var\ (i) \leq Threshold \leq \sum_{i=1}^{N} var\ (i) \tag{4}$$

where N is the number of the sensors used to collect the inputs data, var(i) is the variance of the data collected from sensor i. This range of the threshold was found by experience, and the best threshold is chosen Box 1.

Self-Organizing Multisensor Fusion Classification Algorithm

Step 1: Initialization

- Choose threshold (T)

$$\frac{1}{N}\sum_{i=1}^{N} var(i) \leq T \leq \sum_{i=1}^{N} var(i)$$

where var(i) denotes the variance of the data collected from sensor i and N is the number of sensors used to collect the data.

- Select the first cluster data pointer (DP_1)

$DP_1(i) = f(mean^i), \quad i=1,2,\ldots,N$

where f(.) is a linear function , $mean^i$ is the mean of the data from sensor i, $DP_1(i)$ is the element i of the first cluster data pointer, and N is the number of sensors.

Step 2: At stage k, apply a new input vector (INK).

Step 3: Compute the minimum distances between the cluster data pointers and new k input vector (INK).

$$d_j = \left\| DP_j - INK \right\|^2, \quad j = 1,2,\ldots,nc$$

where d_j is the distance between input vector k and the cluster data pointer # j, INK is the k input vector, nc is number of clusters classified so far, DP_j is the cluster data pointer #j, and

$$d_j^* = \min_j^{\{d_j\}}$$

Step 4: Decision making

If $d_j^* < T$ then $DP_j(k) = DP_j(k-1) + \frac{1}{V_j(k)} \left(DP_j(k) - INK \right)$

else $DP_{nc+1} = INK$

where $V_j(k)$ denotes the number of input vectors in cluster j at stage k.

Step 5: Repeat by going to step 2 if nc < M (M is a large chosen integer number) and not all inputs are clustered.

Step 6: Adjust the numbers of cluster data pointers (equations: 1-3).

Step 7: Go to step 2 if not all the inputs are clustered.

Step 8: If the optimal threshold (satisfied equation 5) is found, stop, else go to step 1. such that the square error e is as such that

$$e = \sum_{i=1}^{nc} \left(var(i) - var_1(i) \right)^2 < \varepsilon \qquad (5)$$

where e is a positive arbitrary small value, $var_1(i)$ is the variance of the cluster i which is classified using Kohonen's algorithm (5), var(i) is the variance of the cluster i that is classified using this algorithm, and nc is the number of clusters classified using this algorithm.

If at stage k the distance between the input vector (INK) and the cluster data pointer j (DP_j) is less than the chosen threshold, then DP_j will be adaptive as such

$$DP_j(k) = DP_j(k-1) + \frac{1}{V_j(k)}\left(DP_j(k) - INK\right),$$
$$j = 1,2,\ldots,nc$$

else

$$DP_{nc+1} = INK$$

where $V_j(k)$ denotes the number of input vectors in cluster j at stage k, and nc is number of clusters classified so far. Since the cluster data pointers DP_j, j=1,2,...,nc, converge to their optimal values, because the following Dvoretzky's conditions (15) are satisfied:

$$\sum_{i=1}^{\infty} \frac{1}{V_j(i)} = \infty, \quad \sum_{i=1}^{\infty} \left(\frac{1}{V_j(i)}\right)^2 < \infty, \quad \|initial\ DP_j\| < \infty, \quad j=1,2,\ldots,nc$$

then this algorithm converges.

APPLICATION

To illustrate the application of the MFC algorithm presented here, consider the problem of classifying an image into targets and backgrounds. The algorithm starts by selecting the first cluster image data pointer vector (FCIDPV). This cluster image data pointer (CIDP) is a vector that has a size equal to the number of sensors which the input image data comes from, and element i of (FCIDPV) is related to the mean of the image data coming from sensor i by a linear function. Element i of the input vector represents an image data point from sensor i, and the size of the input image vector is equal to the number of sensors used to collect the input image data. The first input image vector is compared to the FCIDPV, and it is clustered with the first cluster if the distance to the FCIDPV is less than a chosen threshold. Otherwise it is the image data pointer for a new cluster. This process is repeated for all following input image vectors. The elements of both input image data and stored cluster image

data pointers take on the values from 0 to 255. The number of clusters, which depends on both the distance metric used to compare input image data to cluster image data pointers and the threshold, is adjusted so it does not grow with time. The adjustment of the number of clusters is done by adjusting the number of cluster image data pointers which is adjusted as in equation 1, equation 2, and equation 3. The threshold is selected as a function of the variance of the image data collected from sensor i, and the threshold is adjusted according to equation 5. Two sensors are used to collect data from an airport, and the MFC algorithm is first used to learn the background of the airport without targets (figure 1). Then the MFC is used to classify the data collected from the airport with targets (figure 2). The results of this classification process are shown in figure 3. Data points that belong to the clusters that have been learned using MFC (backgrounds in figure 1) are represented by black, and the others are represented by white. From figure 3 one can note that six targets have been classified, three detected and two backgrounds, one the airport and the other the airport surroundings. The MFC algorithm was used to classify two targets and one background (figures 4 and 5) and to classify one target and one background (figures 6 and 7). This configuration was used in order to compare the

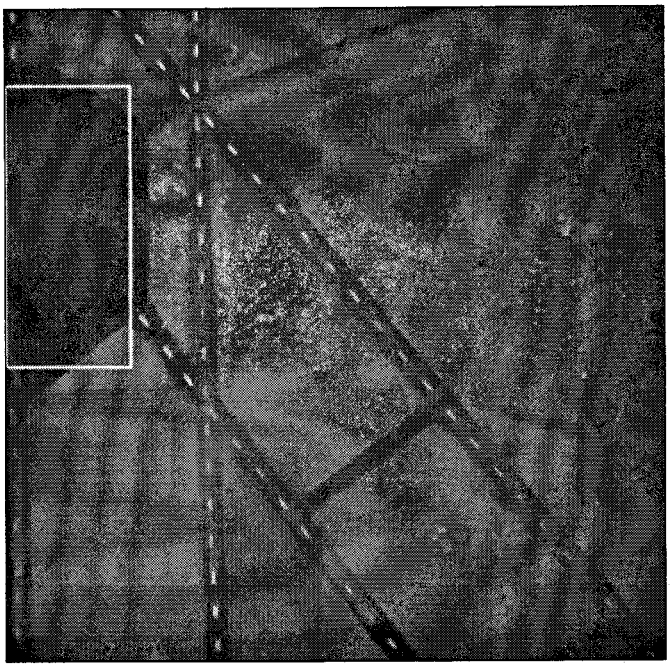

Fig. 1. The background of the airport to be learned using the MFC algorithm

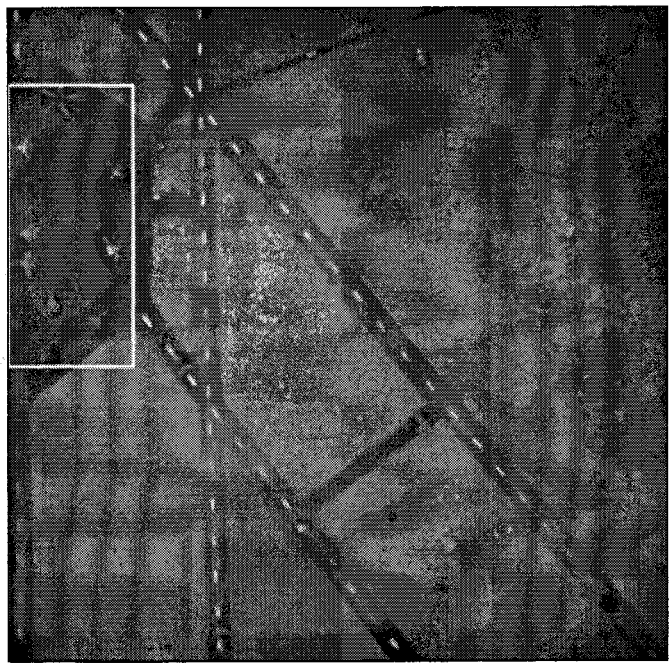

Fig. 2. Multi-target and multi-background to be classified using the MFC algorithm

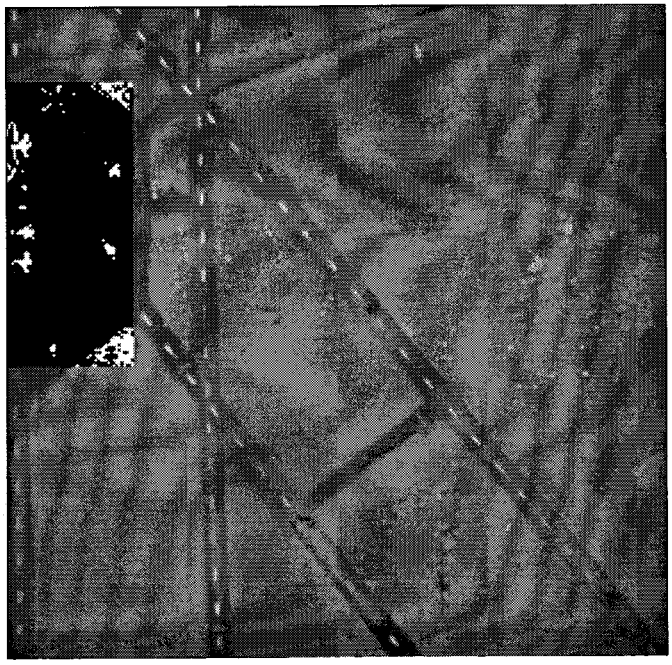

Fig. 3. The result of using the MFC algorithm to classify multi-target and multi-background image

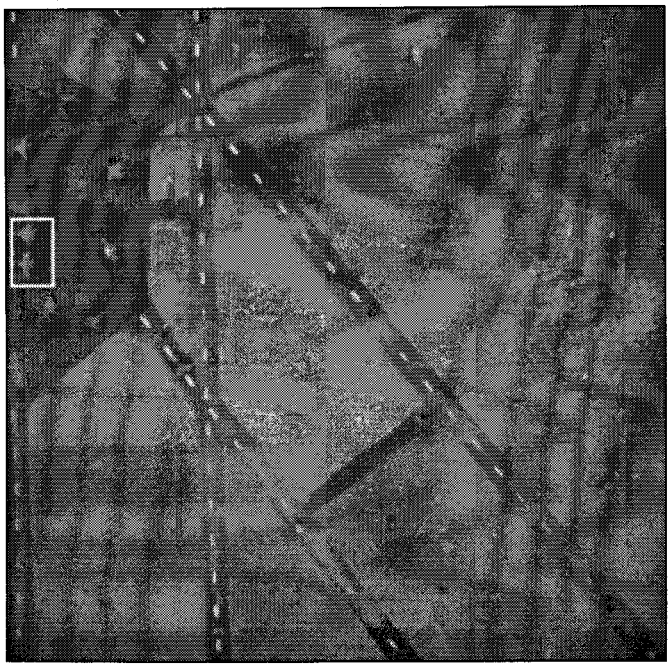

Fig. 4. Two targets and one background to be classified using the MFC algorithm

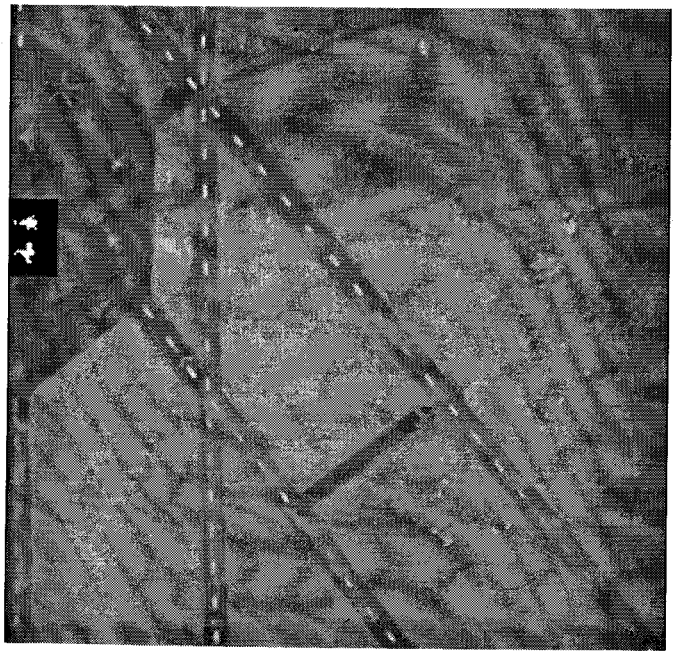

Fig. 5. The result of using the MFC algorithm to classify two targets and one background

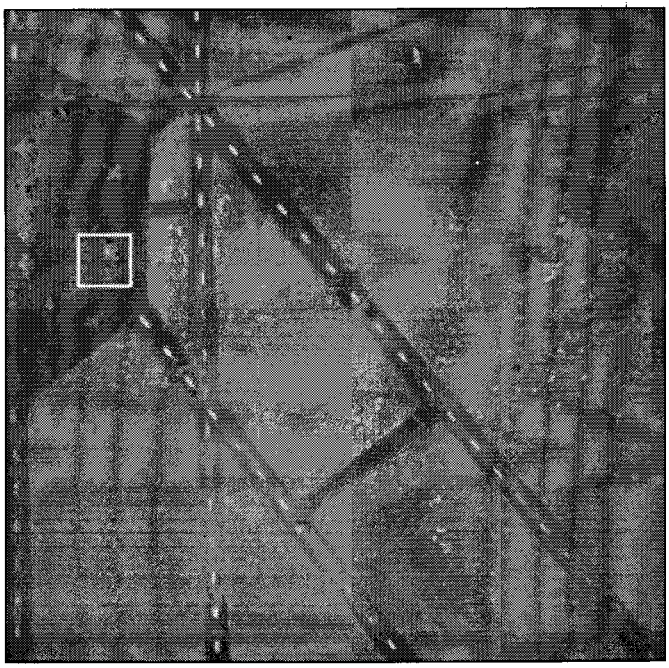

Fig. 6. One target and one background to be classified using the MFC algorithm

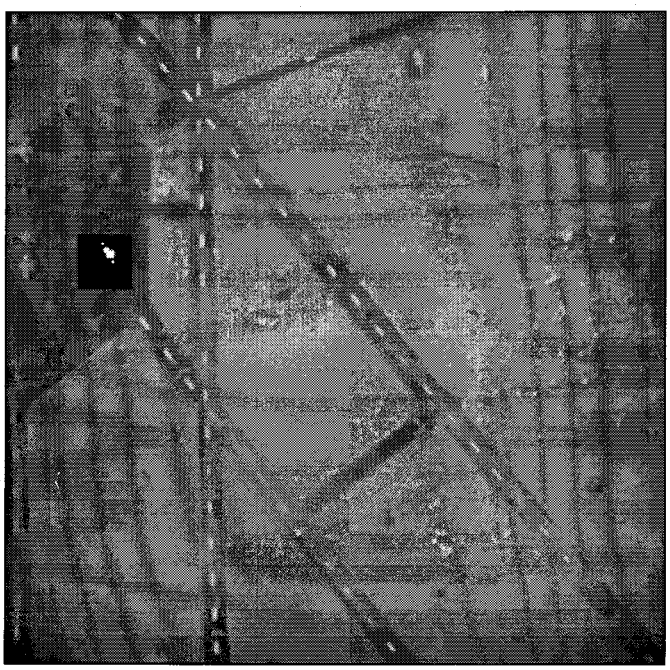

Fig. 7. The result of using the MFC algorithm to classify one target and one background

performance of the MFC algorithm with that of the K-NN algorithm (9). This K-NN algorithm is used with a double window filter in which the inner window surrounds the target, and the outer window contains the background. The K-NN was also used to classify the targets and backgrounds in the airport image in figure 2 and the results can be seen in figures 8 and 9 for two targets and one background and in figures 10 and 11 for one target and one background. Comparing the results of using K-NN algorithm and MFC algorithm, one can note that the results of using MFC algorithm are better, 2 to 3 times (computer time) faster than the K-NN algorithm and less noisy. One can also note that the MFC algorithm can be used to classify one target and one background, and it gives good results (figure 7); it also gives good results when it is used to classify two targets and one background (figure 5), and it gives reasonable results when it is used to classify multi-target and multi-background images (figure 3). Using K-NN algorithm to classify two targets and one background gives good results (figure 9), it gives very good results when it is used to classify one target and one background (figure 11), and when it is used to classify multi-target and multi-background images it gives very noisy results and to find K neatest neighbors is time consuming, particularly for a large number of image data points.

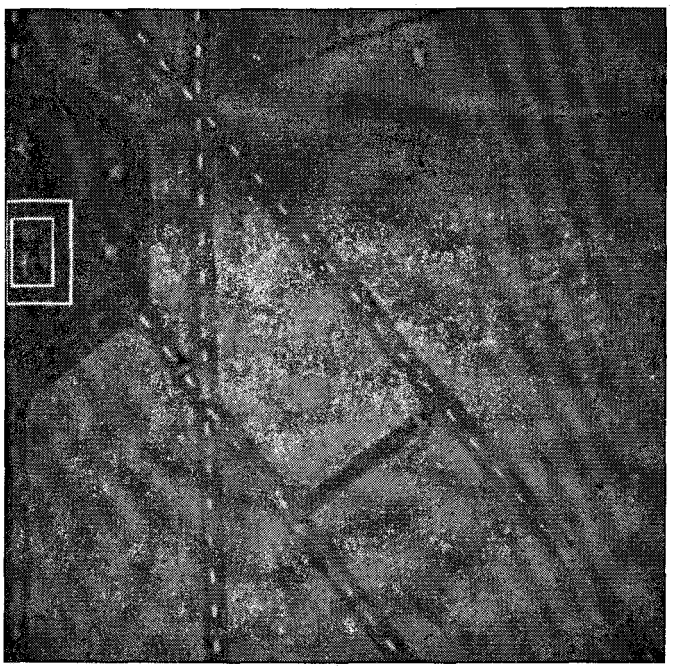

Fig. 8. Two targets and one background to be classified using the K-NN algorithm

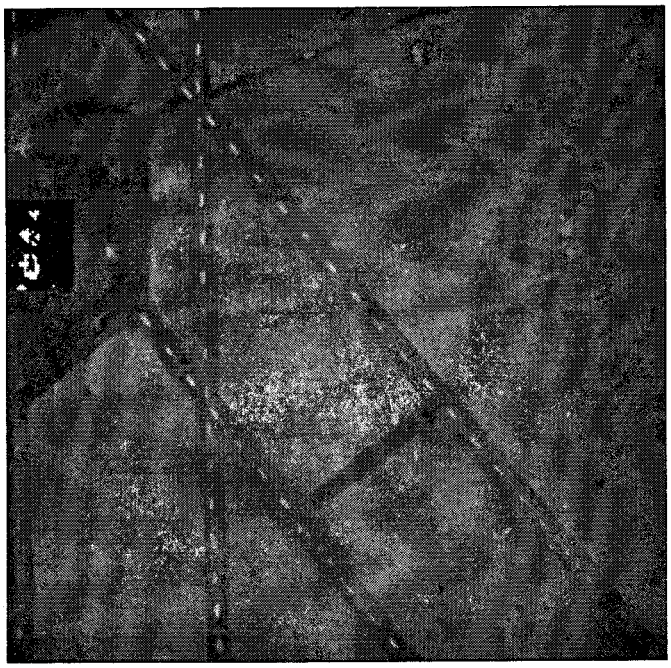

Fig. 9. The result of using the K-NN to classify two targets and one background (K=4)

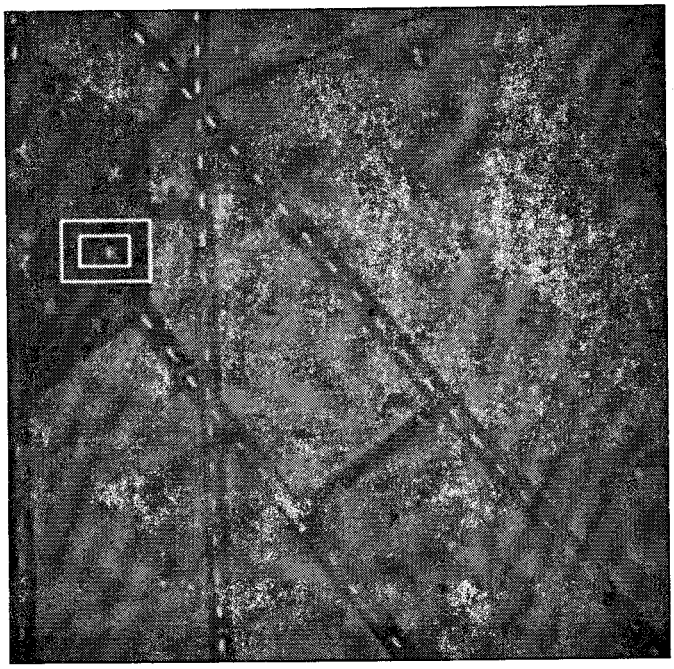

Fig. 10. One target and one background to be classified using the K-NN algorithm

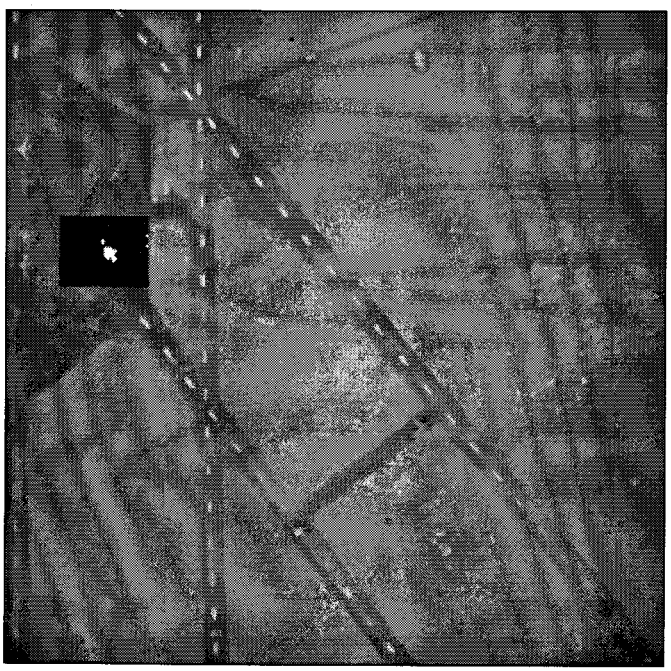

Fig. 11. The result of using the K-NN algorithm to classify one target and one background (K=3)

CONCLUSIONS

Some of the advantages of the self-organizing multisensor fusion classification algorithm (Box 1) are:

1. It does not require the number of classes being known apriori.
2. It is a self-organizing classifier algorithm, since it has the ability to form and adjust the number of clusters without being given the correct number of clusters.
3. It does clustering on the basis of the statistical properties of the set of inputs.
4. It gives a solution to the problem of threshold selection.
5. It adjusts the number of clusters so it does not grow with time.
6. It converges.
7. Its results are better 2 to 3 times (computer time) faster and less noisy than the results of using K-NN algorithm.

This algorithm still needs to be improved to deal with the issue of occlusion and obscuration. Comparing the result in figure 7 for one target and the result in figure 5 for two targets which are near each other, one can note that the separation of individual targets is not perfect when multiple targets in an image are near each other. Also, targets may be obscured by or be partially hidden in shadows, dust, and smoke.

REFERENCES

1. **Hartigan, J.A., 1975**. Clustering Algorithms. New York, New York: John Wiley & Sons.

2. **Mitchell, O.R. and Lutton, S.M., 1978**. Segmentation and Classification of Targets in FLIR Images: Image understanding systems and industrial applications. Proceedings of The International Society for Optical Engineering, Vol. 155, pp. 83 90.

3. **Dasarathy, B.V. and Sheela, B.V., 1979**. A Composite Classifier Design: Concepts and methodology. Proceedings of The Institute of Electrical and Electronics Engineers, Vol. 67, pp. 708-713.

4. **Helland, A.R., Willet, T.J. and Tisdale, G.E., 1981**. Application of Image Understanding to Automatic Tactical Target Acquisition: Techniques and applications of image understanding. Proceedings of The International Society for Optical Engineering, Vol. 281, pp. 26-31.

5. **Lippman, R.P., 1987**. An Introduction to Computing with Neural Net. The Institute of Electrical and Electronics Engineers ASSP Magazine, pp. 4-22.

6. **Dechter, Rina, Pearl, and Judea, 1989**. Tree Clustering for Constraint Networks. Artificial Intelligence, Vol. 38, No. 3, pp. 353-366.

7. **Parthasarathy, Guturu, and Chatterji, 1990**. Class of New KNN Method for Low Sample Problems. The Institute of Electrical and Electronics Engineers Transactions on Systems, Man and Cybernetics, Vol. 20, No. 3, pp. 715-718.

8. **Lowe, D. and Webb, A.R., 1991**. Optimized Feature Extraction and the Bayes Decision in Feed-Forward Classifier. The Institute of Electrical and Electronics Engineers Transactions on Pattern Analysis and Machine Intelligence, Vol. 13, No. 4, pp. 355-364.

9. **Scott, D.R., Flachs, G.M., Gaughan, and Patrick, T., 1991**. Sensor Fusion Using K-Nearest Neighbor Concepts. Proceedings of The International Society for Optical Engineering, Vol. 1383, pp. 367-378.

10. **Ramasubramanian, V. and Paliwal, K.K., 1992**. Fast K-Dimensional Tree Algorithms For Nearest Neighbor Search With Application To Vector Quantization Encoding. IEEE Transactions on Signal Processing, Vol. 40, No. 3, pp. 518-531.

11. **Barnes, C.F. and Frost, R.L., 1993**. Vector Quantizers with Direct Sum Codebooks. IEEE Trans. Inform. Theory. Vol. 2, No. 2, pp. 565-580.

12. **Torres, L. and Huguent, J., 1994**. An Improvement on Codebook Search for Vector Quantization. IEEE Transactions on Communication, Vol. 42, No. 2-4, pp. 208-210.

13. **Sahiner, B. and Yagle, A.E., 1995**. Region-of Interest Tomography Using Exponential Radial Sampling. IEEE Transactions on Image Processing, Vol. 4, No. 8, pp. 1120-1127.

14. **Wegmann, B. and Zetzsche, C., 1996**. Feature-Specific Vector Quantization of Images. IEEE Transactions on Image Processing, Vol. 5, No. 2, pp. 274-288.

15. **Saridis, G.N., 1978**. Self-Organizing Control of Stochastic Systems. New York, New York: Marcel Dekker.