Classifier Swarms for Human Detection in Infrared Imagery

Yuri Owechko, Swarup Medasani, and Narayan Srinivasa
HRL Laboratories, LLC
3011 Malibu Canyon Road, Malibu, CA 90265
{owechko, smedasani, nsrinivasa}@hrl.com

Abstract- *In this paper, we describe a new method for visual recognition of objects in an image that combines feature-based object classification with efficient search mechanisms based on swarm intelligence. Our approach utilizes the particle swarm optimization algorithm (PSO), a population based evolutionary algorithm, which is effective for optimization of a wide range of functions. PSO searches a multi-dimensional solution space for a global optimum using a population of "particles" in which each particle has its own velocity vector. In our approach, we extend PSO using sequential niching methods to handle multiple minima. Also, in our approach, each particle in the swarm is actually a self-contained classifier that "flys" through the solution space seeking the most "object-like" regions. By performing this optimization, the classifier swarm simultaneously finds objects in the scene, determines their size, and optimizes the classifier parameters.*

I. INTRODUCTION

We describe a new method for visual recognition of objects in a scene which combines feature-based object classification with efficient search mechanisms based on swarm intelligence. Objects in a visual scene need to be located and classified so they can be tracked effectively for automotive safety, surveillance, perimeter protection, and a variety of other government, military, and commercial applications. Typically, classification of objects in an image is performed using features extracted from an analysis window that is scanned across the image. This brute force search can be very computationally intensive, especially if a small window is used since a classification must be performed at each window position. Conventional approaches to reducing the computational load are based on reducing the search space by using another sensor such as a scanning radar to cue the vision system and measure the range of the object. Limitations of the radar approach include high cost, false alarms, the need to associate radar tracks with visual objects, and overall system complexity. Alternatively, previous vision-only approaches have utilized motion-based segmentation using background estimation methods to reduce the search space by generating areas of interest (AOI) around moving objects and/or using stereo vision to estimate range in order to reduce searching in scale. These methods add cost and complexity by requiring additional cameras and computations. Motion-based segmentation is also problematic under challenging lighting conditions or if background motion exists as is the case for moving host platforms.

We describe a novel search mechanism that can efficiently find multiple instances of multiple object classes in a scene without the need for cueing sensors or scan-based searching. Our approach utilizes the particle swarm optimization (PSO) algorithm [1,2], a population based evolutionary algorithm, which is effective for optimization of a wide range of functions. The algorithm models the exploration of multi-dimensional solution space by a population of individuals where the success of each individual has an influence on the dynamics of other members of the swarm. Basic PSO has proved effective in exploring complicated fitness landscapes and converging populations of particles to a single global optimum, although it has been shown that the basic PSO is not guaranteed to converge to a local or global optimum. However, some optimization problems require the identification of global as well as local minima in a multi-modal framework. We extend the PSO algorithm using sequential niching methods [3,4] to enable it to locate multiple objects in the scene. Our approach is substantially different from the previous work in that each particle from the population is a unique classifier. As the population swarms around, the classifiers adjust parameters to best detect the objects in the scene.

Our method also differs from other vision algorithms that use swarm intelligence in that the other methods use swarms to build up features using ant colony pheromone-based ideas. In our method swarming is done at the classifier level in a space consisting of object location, scale, and classifier parameter dimensions and where each particle is a complete classifier. The particles swarm in this space in order to find the local optima that correspond to objects in the image. The classifier details are not visible at the abstraction level of the swarm.

To our knowledge, there are no existing methods that using particle swarms to implement object detection systems. However, there have been attempts to use Genetic Algorithms (GAs) and Evolutionary Algorithms for object detection [5]. In [5], the authors employ GA to detect and verify faces from images encoding only the position of the face. Object scale is handled by scaling the input image. Distances to eigen spaces are used as the fitness functions. Genetic algorithms have been used before for decreasing the search space in vision systems [6]. These systems employ a population of individual solutions that crossover and mutate in an effort to maximize the fitness function. Other efforts have used GAs for training and adapting neural networks to recognize objects [7].

The paper is organized as follows. In Section II, we briefly introduce the PSO algorithm. Details on using classifier swarms for object detection are presented in Section III. The Sequential Niching-based PSO (SNPSO) is introduced in detail in Section IV. In Section V, we

0-7695-2158-4/04 $20.00 (C) 2004 IEEE

present a variety of results on the human detection application. Finally, we summarize our conclusions in Section VI.

II. PARTICLE SWARM OPTIMIZATION

PSO is a relatively simple optimization method that has its roots in artificial life in general, and to bird flocking and swarming theory in particular [1,2]. Conceptually, it includes aspects of genetic algorithms and evolutionary programming. Each potential solution is assigned a randomized velocity vector and the potential solutions called particles then "fly" through the space in search of the function optima. Each particle keeps track of its coordinates in multi-dimensional space that are associated with the best solution (*pbest*) it has observed so far. A global best parameter (*gbest*) is used to store the best location among all particles. The velocity of each particle is then changed towards *pbest* and *gbest* in a probabilistic way according to

$$v^i(t) = wv^i(t-1) + c_1 * rand() * (pbest - x^i(t-1)) + c_2 * rand() * (gbest - x^i(t-1)) \quad (1)$$

$$x^i(t) = x^i(t-1) + v^i(t) \quad (2)$$

Where $x^i(t)$ and $v^i(t)$ are the position and velocity vectors at time t of the i-th particle and c_1 and c_2 are parameters that weight the influence of their respective terms in the velocity update equation. w is a decay constant which allows the swarm to converge to a solution more quickly. The rand() function generates a random number between 0 and 1 with a uniform distribution. The above dynamics reflect a socio-psychological model where individual particles change their beliefs in accordance with a combination of their own experience and the best experience of the group. (This is in contrast to other models of cognition where an individual changes his beliefs to become more consistent with his own experience only.) The random element introduces a source of noise, which enables an initial random search of the solution space. The search then becomes more directed after a few iterations as the swarm starts to concentrate on more favorable regions. This type of search is much more efficient than scanning or gradient based search methods. It is similar to genetic algorithms in that it can be used for discontinuous and noisy solution spaces since it only requires an evaluation of the function to be optimized at each particle position. No gradient information is used. Unlike GAs, the PSO particles are not modified at each iteration, they just travel to a different position, calculate the solution at that position, and compare it with their own and global best value in order to update their velocity vectors. PSO relies on the fact that in most practical problems the optimum solution usually has better than average solutions residing in a volume around it. These good solutions tend to attract the particles to the region where the optimum lies. The swarm becomes more and more concentrated on likely regions until the optimum is found, e.g. gbest no longer changes. PSO has been applied to a wide variety of optimization problems. It has been found experimentally that the number of particles and iterations required scale weakly with the dimensionality of the solution space. The total number of function evaluations is very small compared to the size of the solution space, as will be seen below. Basic PSO searches only for a single optimum in the solution space, but various approaches have been described for finding multiple local optima or "niches"[8,9]. We now introduce classifier swarms and describe them in detail.

III. CLASSIFIER SWARMS

Objects in a visual scene need to be located and classified so they can be tracked effectively for automotive safety, surveillance, perimeter protection, and a variety of other government, military, and commercial applications. Typically, classification of objects in an image is performed using features extracted from an analysis window that is scanned across the image. We propose an approach where a swarm of classifiers moves around in the search space looking for selected class of objects. One of the novel aspects of our approach is that two of the dimensions are used to locate objects in the image, while the rest of the dimensions are used to optimize the classifier parameters.

Our approach is a much more efficient method for finding objects in an image compared to searching based on scanning the image or using gradient information, especially if the scale of the object is not known beforehand. We have measured speedup factors of over 1000 relative to sequential scanning when searching in three dimensions. The number of false alarms per image is also greatly reduced, which is very important for practical applications. The speedup and false alarm advantages over sequential scanning increase as the number of dimensions is increased which makes it feasible to include object rotation angle as one of the search space dimensions. This approach will help increase the range of applications for vision systems by improving performance, reducing computational requirements dramatically, eliminating the need for cueing sensors such as radar, and reducing overall cost of practical systems.

The basic architecture for our approach is shown in Fig. 1. The objective is to find multiple instances of an object class in an input image. The PSO particles fly in a solution space where two of the dimensions represent the x and y coordinates in the input image. The key concept in our approach is that each particle in the PSO swarm is a self-contained object classifier which outputs a value

representing the classification confidence that the image distribution in the analysis window associated with that particle is or is not a member of the object class. All particles are instances of the same classifier and only the classifier parameters vary as the particle visits different positions in the solution space. As mentioned before, two of the solution space dimensions represent the location of the analysis window on the input image. A third dimension represents the size or scale of the analysis window in order to match the unknown size of objects in the image. Additional dimensions can be used to represent other classifier parameters such as, for example, the rotation angle of the object.

One can imagine a multidimensional surface of classifier confidence (or saliency map) that can be generated if the classifier is scanned across all of the dimensions. The saliency map for an image can be discontinuous and noisy, with many isolated false alarms where the classifier responds incorrectly to patterns in the image. Thus gradient-based methods cannot be used to find objects in the image, which is why brute force scanning is usually used. By generating saliency maps for many images, we have found experimentally that objects in the scene tend to have large "cores" of high confidence values. Many false alarms tend to be isolated with small cores. Since the probability of a particle passing near or through a larger core is greater for a larger core, the particles are attracted more to larger cores and the number of false alarms in a image are reduced using SNPSO compared to simple scanning. In simple scanning, all of the false alarms in an image will be detected so the classifier must be biased towards very low false alarm rates in order to keep the overall false alarm rate low, which also has the side effect of reducing the detection rate.

A comparison of the computational requirements of PSO classification compared to brute force scanning is given in Table. 2. The speedup factor for PSO is SM^2/PK where it is assumed that M scan positions are searched along x and y and S positions are searched along the scale dimension. P is the number of particles and K is the number of iterations. We have found for realistic images that P=80 and K=10 work well for 3 dimensions. If M=S=100, then the speedup factor is 1250. The speedup advantage of PSO increases further with the number of dimensions. The number of false alarms is also reduced relative to brute force scanning.

In our case, we developed a classifier to detect humans [8]. Given a 2:1 aspect ratio window in an image, pre-selected Haar wavelet features were computed along with a set of fuzzy edge-symmetry features. This combined 190D feature vector was then passed to a Non-linear Discriminant Analysis (NDA) neural network for detecting the presence of a human in the selected window region. Typically, to find all humans in the scene, the window needs to be scanned across the entire image. In conventional approaches, an estimate of the object height or depth from the camera needs to be known to determine the height of the scanning window. In our approach, the height of the classifier is also automatically obtained from the PPSO swarm dynamics. The performance of the trained human detection classifier on different infrared imagery is presented in Figures 2.

We now describe the sequential niching approach to classifier swarms that will be used to find multiple objects in the scene.

III. SEQUENTIAL NICHING PARTICLE SWARM OPTIMIZATION (SNPSO)

The flow chart for our sequential niching method for finding multiple objects in a scene is shown in Fig. 1. After an input image is received, a running list of object positions in the scene is cleared as well as the boundary flag table described below. The particle swarm of classifiers is then initialized in random positions in the solution space. The boundaries of the initialization volume can be set using various criteria. For example, if the flat ground constraint is appropriate, then we can utilize the fact that objects of a certain size will appear only in certain subregions of the image to reduce the search volume. After initialization, the swarm dynamics are iterated until the global best (gbest) exceeds a preset threshold or the number of iterations reaches a preset maximum value. If gbest does not exceed the threshold then it is assumed no targeted objects are present in the scene and the system waits for the next input image. If gbest does exceed the threshold, then a neighborhood check is performed around the gbest position to see if a certain number of neighboring positions in the image also exceed the threshold for that analysis window size. If the gbest position passes the neighborhood test, then gbest is added to the list of object positions. The image is then erased locally at the gbest position with a Gaussian whose width is proportional to the analysis window size. If gbest does not pass the neighborhood test, then the image is erased at gbest without adding gbest to the object list. The purpose of the local erasure step is to remove that object's influence on the swarm when the swarm is re-initialized to search for the next object. By erasing the image locally, the influence of that region on the swarm across all dimensions is eliminated. After the image erasure step, a boundary flag table is updated with the erasing Gaussian. This table has the same dimensions as the image and is in one-to-one correspondence with it. All table entries within the width of the erasing Gaussian are set ON. If a particle lands on a location whose flag is ON, the particle will not run its classifier since it is already known that an object is present at that location. Instead, the particle will keep its previous pbest value. This eliminates unnecessary classifier evaluations in regions where it has already been

0-7695-2158-4/04 $20.00 (C) 2004 IEEE

determined that an object exists which speeds up the swarm, especially if large objects are present in the scene. We do not implement any repelling forces at object locations to avoid isolating regions. After the Gaussian flag table is updated, the swarm is re-initialized to search for the next object. The boundary flag table is cleared when a new image is acquired.

This approach can be extended naturally to searching for members of different object classes in parallel. We simply run multiple swarms for multiple object classes. Each swarm maintains its own gbest, particle best (pbest), and object lists. The swarms interact indirectly only through the erasing Gaussians and boundary table which are common to all swarms. Thus if a swarm detects an object at a particular location in the image, the other swarms do not try to find other objects there.

IV. RESULTS

We have successfully implemented the sequential niching swarming classifiers method for detecting objects in a scene of unknown position and size. Each of the particles represented a classifier based on Haar wavelet and fuzzy symmetry features and a backpropagation neural network classification engine. The demonstration was done in Matlab with the particle classifiers implemented using a C++ dynamic link library called from Matlab. Due to the paucity of space we only show few relevant examples. The results are shown in Figs. 3 to 5. The swarm was programmed to search for pedestrians in the *x*, *y*, and *size* dimensions. In these examples we used infrared images, but the approach works equally well in the visible. In fact, the classifier was trained using visible light images only. The first test image was a single person walking outside between a building and a fence with hills in the background. Fig. 3 shows the swarm after a single iteration. The 3D solution space with the current positions of the classifier particles is shown in the upper left. At this early stage the particles are distributed widely in *x*, *y*, and *scale*. Particles that exceed the classification threshold but don't pass the neighborhood test are colored red, particles that pass both criteria are colored green, and all other particles are blue. The upper right image shows the particle positions projected on the x-y plane of the input image, including the effects of any erasing Gaussians. The analysis windows corresponding to the green particles are shown superimposed on the input image in the lower part of the figure. After only a single iteration, the particles are all blue and the pedestrian has not yet been found. Fig. 4 shows the swarm after 10 iterations. The particles are now centered on the pedestrian with a range of window sizes. Only one particle passes the classifier threshold and neighborhood tests. The corresponding erasing Gaussian and window are shown. It should be emphasized that it is not necessary to wait until the entire swarm condenses on an object before deciding an object is at that location. We label that location as an object as soon as a single particle passes the classification and neighborhood tests, at which point the image is erased locally, the boundary table is updated, the swarm is re-initialized, and search starts for the next object in the scene. In Figure 5, we show results for an infrared image with multiple pedestrians. After 13 iterations all 3 pedestrians were detected and it was determined that no more pedestrians were present. Note the 3 erasing Gaussians in the upper right centered on the pedestrians.

The same swarm parameters and classifier were used for both of these examples. The number of particles was 80 and the maximum number of iterations was limited to 10. The total number of classifier evaluations was 1280 for the single pedestrian and 1840 for the multi-pedestrian example. This is far less than the number of evaluations necessary to search by scanning in *x*, *y*, and *scale*. Pedestrians are often detected within 2 to 3 iterations. Most of the processing is spent on determining that no more pedestrians are present by propagating the swarm until the maximum allowed number of iterations is reached. The effect of this overhead can be reduced by using the time dimension or if the number of objects is known *a priori*. When the time dimension is used, the swarm searches a video cube consisting of a stack of image snapshots taken at discrete time intervals instead of a single image. Advantages of searching in time as well as space include automatically finding the space-time paths of objects, further reduction in false alarms because consistent objects will form extended saliency "tubes" in space-time, and the object sweep necessary for determining if all objects have been found needs to be done only once for each image stack rather than for each individual image.

V. CONCLUSIONS

In this paper, we introduced the concept of classifier swarms for effectively searching an image for multiple objects of interest. We first extended the particle swarm optimization algorithm using sequential niching methods to search for multiple minima. A human detection classifier that uses Haar wavelet and edge-symmetry features was designed. A human detection classifier was used in place of the usually used particle in the sequential niching PSO. From the variety of results presented, we can see that the proposed approach is an efficient and effective search mechanism. It is also shown to be very fast and can robustly detect multiple objects in the scene.

VI. REFERENCES

[1] Kennedy, J., Eberhart, R. C., and Shi, Y., *Swarm Intelligence* San Francisco: Morgan Kaufmann Publishers, 2001.

[2] R. C. Eberhart and Y. Shi, "Particle Swarm Optimization: Developments, Applications, and

0-7695-2158-4/04 $20.00 (C) 2004 IEEE

Resources," 2001.
[3] R. Brits, A. P. Engelbrecht, and F. van den Bergh, "A Niching Particle Swarm Optimizer," 2002.
[4] D. Beasley, D. R. Bull, and R. R. Martin, "A Sequential Niching Technique for Multimodal Function Optimization," Evolutionary Computation, 1(2), p. 101-125, 1993.
[5] G. Bebis, S. Uthiram, and M. Georgiopoulos, Face Detection and Verification Using Genetic Search, International Journal on Artificial Intelligence Tools, vol. 9, no 2, pp. 225-246, 2000.
[6] D. L. Swets, B. Punch, and J. Weng, Genetic Algorithms for Object Recognition in a complex scene, Proc. of Intl. Conference on Image Processing, vol. 2, Oct, pp 23-26, 1995.
[7] V. Ciesielski and M. Zhang, Using Genetic Algorithms to Improve the Accuracy of Object Detection, In *Proceedings of the third Pacific-Asia Knowledge Discovery and Data Mining Conference,* Ning Zhong and Lizhu Zhou (Eds.), Knowledge Discovery and Data Mining -- Research and Practical Experiences. Tsinghua University Press, page 19 - 24. Beijing, China, April 26-31, 1999.
[8] N. Srinivasa, S. Medasani, Y. Owechko, and D. Khosla, *Fuzzy edge-symmetry features for improved intruder detection*, The 12th IEEE International Conference on Fuzzy Systems, Vol: 2, 920-925, 2003.

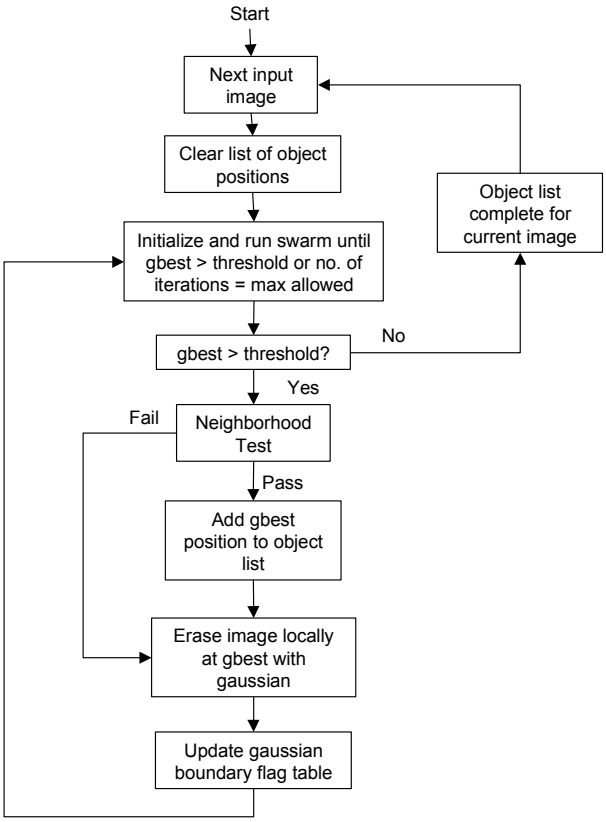

Figure 1. Flow chart for sequential niching PSO vision system.

Table. 1. Comparison of computational requirements of PSO vision system with brute force scanning.

Conventional Scanning	Particle Swarm
Wavelet Size = NxN Image Size = MxM No. of Features =F Operations per Feature = 2N(N-2) Total complexity per image = M^2*F*2*N(N-2) (Assumes that complete image is scanned to find objects of interest, stepsize = 1 pixel)	No. of Particles = P No. of Iterations = K No. of Features =F Operations per Feature = 2N(N-2) Total complexity per image = P*K*F*2*N(N-2) (Note that P*K << M^2)

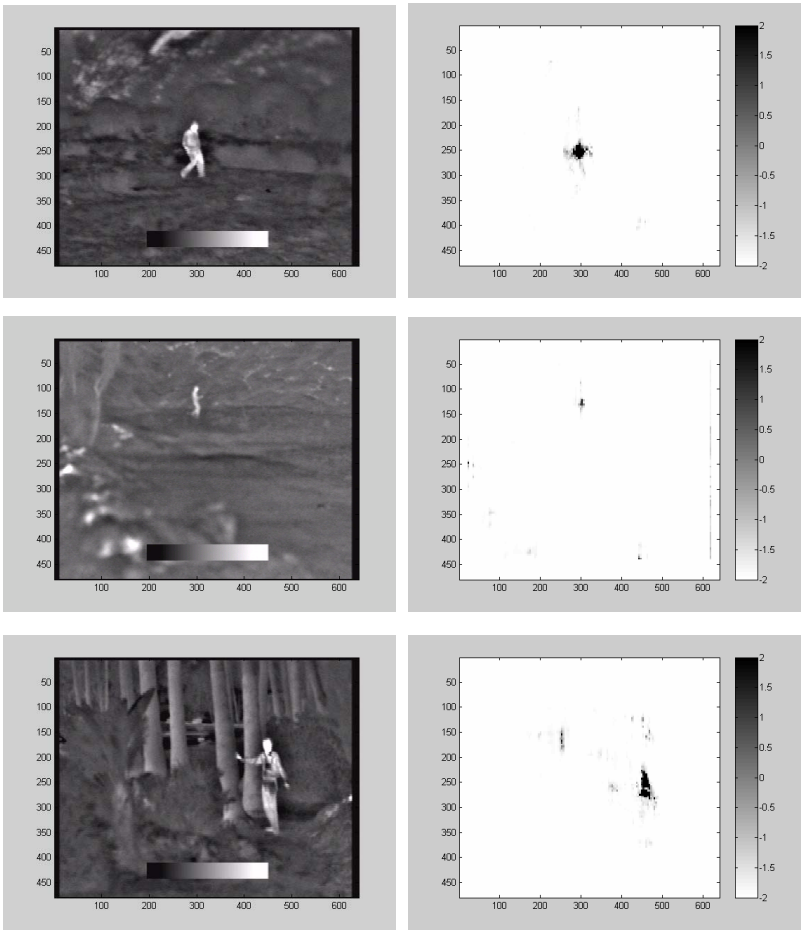

Figure 2. Saliency maps generated by human detection classifier for the IR images on the left. The saliency maps show the classifier response at each spatial location in the IR image. Black and white indicate the maximum and minimum classifier responses, respectively.

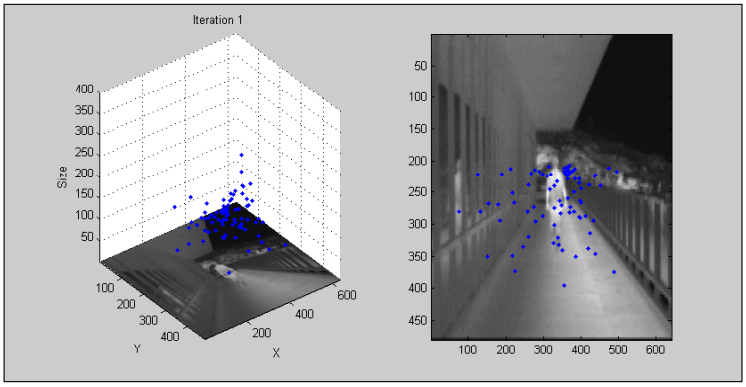

Figure. 3. Results for sequential niching PSO vision system for single pedestrian in the infrared. Results are shown after a single iteration of the swarm. The object is not yet detected.

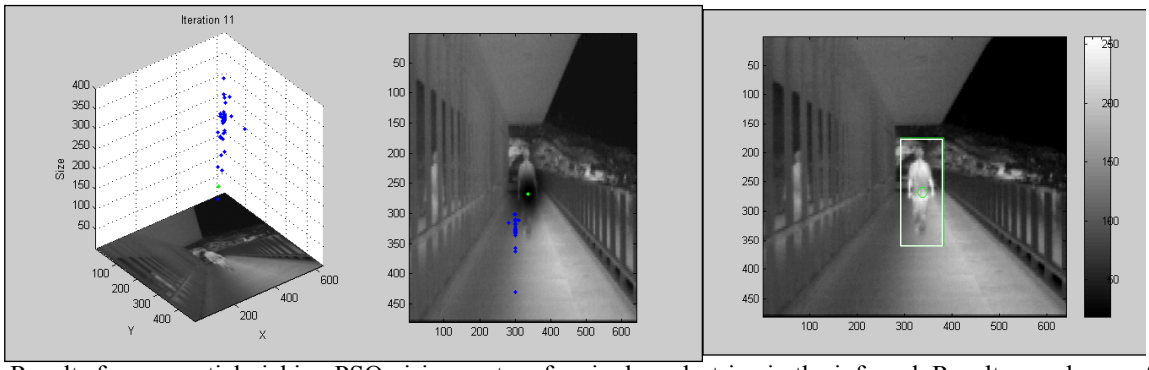

Fig. 4. Results for sequential niching PSO vision system for single pedestrian in the infrared. Results are shown after 10 iterations of the swarm. The object has been detected and it was determined that no other pedestrians are present in the image.

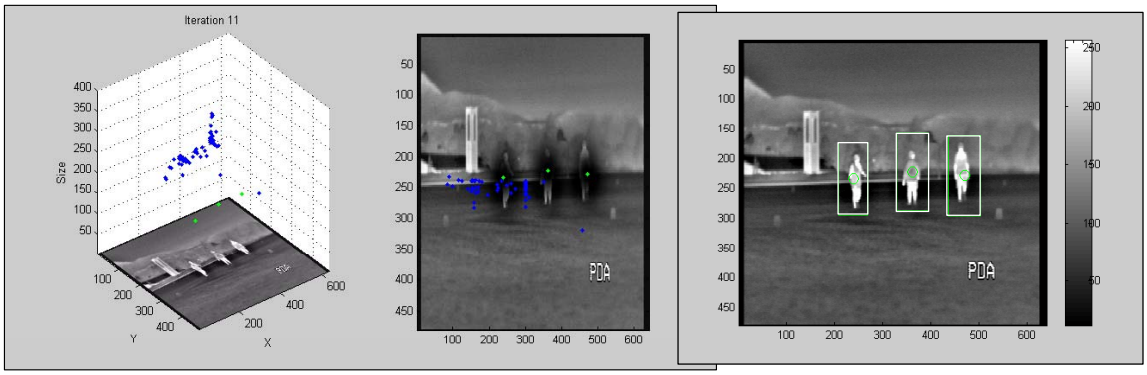

Fig. 5. Results for sequential niching PSO vision system for multiple pedestrians in the infrared. Results are shown after 10 iterations of the swarm. All three objects have been detected and it was determined that no other pedestrians are present in the image.