International Journal of Computer Applications (0975 - 8887)
Volume 60 - No. 3, December 2012

A Fast-Multiplying PSO Algorithm for Real-Time Multiple Object Tracking

Fakheredine Keyrouz
Notre Dame University - Louaize
Zouk Mosbeh, Lebanon

ABSTRACT

The problem of real-time object tracking in live video sequences is of increasing importance today mainly due to higher security requirements for surveillance applications. In this study we present a novel particle swarm optimization (PSO) algorithm with additional new features. The basic idea of PSO is to use one swarm or one hierarchical swarm of particles to find the best estimate or the global optimum for the object location in a given search space. Particles fly around, share information with each other, and optimize their behavior to find the global optimum. Until today, PSO was used to track one pre-classified pattern of objects. The existing algorithms apply only one swarm of particles to track predefined patterns. The algorithm we present in this paper extended the PSO algorithm to track different objects having non-predefined patterns: n swarms are used to track n objects, i.e. to find n local maxima in different parts of the search space. The proposed algorithm introduces two new components to PSO. A self-adapting component, which is robust against drastic brightness changes of the image sequence, and a self-splitting component, which decides to track the scene as one connected object, or as more stand-alone objects.

General Terms:

Swarm Intelligence, Tracking.

Keywords:

Object Tracking, Particle Swarm Optimization, Real-time performance, Self-splitting.ifx

1. INTRODUCTION

Real-time object tracking is having an increased importance by a number of applications today: elderly people surveillance, human-computer interaction, medical imaging, as well as security and entertainment systems. An emerging necessity to extract and track people or objects in live videos is strongly demonstrated in our everyday life.
Today's tracking algorithms are able to find and follow single or multiple objects in a precise way under certain predefined conditions. However, if the brightness, intensity, and color conditions are continuously changing, or if the object's velocity is not constant and the objects are occluded by other barriers, most tracking algorithms fail to achieve a satisfying tracking performance. One common apporach to the multiple object tracking problem is the use of multiple trackers, in such a way that each tracker is tuned to follow one and only one object [6]. Other successful approaches use training sets and static templates to model the appearance of objects. While these techniques are usually computationally inexpensive and can be implemented for real-time tracking applications, their performance degrades considerably when the object appearance undergoes changes beyond the training set [1].
Acceptable experimental results for the special situations, where frequent oclusions and light variations are frequent, have been proposed in [5] where Hidden Markov Models (HMMs) provide a solution to the particular problems at hand. Additionally, a popular approach to object tracking is the Kalman filter [11], where the data from the last rounds is used to predict the current position of the tracked object with discrete variables. Techniques like mean shift [4] and edge detection [2] are normally used as a pre-processing step to object extraction and tracking.
A Particle Swarm Optimization (PSO) tracking approach has been suggested in [10]. The basic PSO idea is to use n particles or birds belonging to one swarm in order to find the best object position in a given parameter space. Every particle explores one part in the search space, and all the particles work concurrently. Experimental results showed that existing PSO algorithms outperform other mathematical models, though with certain limitations, i.e. tracking of multiple completely independent objects without predefined patterns is not yet achieved [13, 10]. Recent studies use multiple particle filter trackers to track independent objects which do not interact with each other. However, when interactions happen between multiple objects, it is difficult to distinguish and associate the correct image observations to the corresponding trackers, especially for those objects that are similar in appearance. This makes the independent trackers fails more often than not [12]. The method introduced in [12] formulates the multiple interaction problem as a general optimization problem and proposes a new hybrid PSO that incorporates a differential evolution mutation operation with a Gaussian based PSO. Using one swarm along with multiple particle filters, this method successfully handles frequent occlusions and object interactions.
In this study, we introduce a novel self-splitting PSO which is able to track completely stand-alone objects and their mutual interactions in real-time. this new PSO-based self-splitting and self-adapting algorithm is robust against changing light conditions, fast enough for real-time usage, and with the outstanding option to track two or more completely independent objects. When the objects are occluded by each other, only the swarm tracks all objects, ie.e only one tracking window will be assigned. When the objects start moving apart, the algorithm automatically splits the tracking window according to the number of objects in the image.

2. PARTICLE SWARM OPTIMIZATION

The PSO algorithm was first introduced in [8]. The term swarm is applied to fish, insects, birds, and microorganisms, and describes a behavior of a school of animals of similar size and body orientation, generally cruising in the same direction. These animals adjust their physical movement in order to avoid predators, seek

food and mates, optimize environmental parameters such as temperature.
The general idea of PSO is inspired by a flying swarm of birds searching for food: they will cooperatively search the area by continuously communicating their current positions and proximity to one another. Every bird knows where the other birds are and what they have found. The reliability of a recently found food location is defined by a static evaluation function, the so called fitness function. In every round of the algorithm, birds cogitate whether it is appropriate to change their position towards the higher amount of food or to stay at their current position. After many iterations, nearly all birds are at the peaks or close to the peaks of the search space where the food exists.
In comparison with other stochastic optimization techniques like genetic algorithms (GAs) [7], PSO has fewer complicated operations, fewer defining parameters, and can be coded in just a few lines. Because of these advantages, it has received increasing attention in a variety of applications in recent years. Training of artificial neural networks, finding optimal solutions for equations, and localizing mobile robots in video sequences [9, 3] are few examples. Basically, PSO is usable for every problem definable with an n-dimensional search space, where every position in this search space has a testable value. In PSO, the particles are initialized with certain positions of the objects to be tracked. The influence the particles exercise on each other is based on changing factors like the position of those particles, the direction they are flying to and their velocity.
The velocity of each particle is defined by

$$v_{x,y,z}(t) = \omega \cdot v_{x,y,z}(t-1) + c_1 \cdot \phi_1 \cdot (pbest_{x,y,z} - p_{x,y,z}(t-1)) + c_2 \cdot \phi_2 \cdot (gbest - p_{x,y,z}(t-1)) \quad (1)$$

where $v_{x,y,z}(t)$ denotes the velocity in the x, y, and z directions at time instant t, ω is a weighting factor, c_1 and c_2 are a-priori defined weighting factors, ϕ_1, ϕ_2 are random numbers between 0 and 1, p is the position of the particle, $pbest$ is the best position found by this particle and $gbest$ is the best position found by all particles.
The position at the current time t for a certain particle, $p_{x,y,z}(t)$, is updated as its previous position $p_{x,y,z}(t-1)$ augmented by its current velocity $v_{x,y}(t)$. This process is repeated after every iteration of the PSO algorithm.

$$p_{x,y,z}(t) = p_{x,y,z}(t-1) + v_{x,y,z}(t) \quad (2)$$

It is important that the search space shows convergent behavior, otherwise it is nearly impossible for the PSO to attain acceptable results. The PSO mechanism assumes that in most situations the peaks have smaller peaks around them. This concept is inspired by nature: If the highest mountain top is at a certain location, it is normally surrounded by lower mountain tops around that location. Thus, if one particle has found the current global best position, all the other particles drift towards this best particle. The velocity and the accurate direction of a drifting particle changes from one frame to the other, it's therefore not deterministic.
Figure 1 illustrates a typical search scenario for the global best position in a three-dimensional space. The term $z = f(x, y)$ denotes a randomly generated function with peaks distributed according to random variables x and y. In this context, (x, y) could pinpoint the position of the particle with respect to the object to be tracked, and $f(x, y)$ denotes the grey value of the object at this position.The best position (x, y) corresponds to the highest point in $f(x, y)$. All the N particles assigned with a certain object to be tracked are communicating their current position and trying to find the global best position.

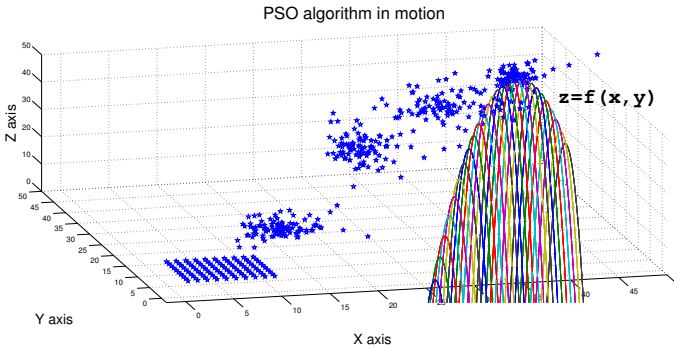

Fig. 1. Different time-snaps of one swarm of particles searching for the global best position in a three dimensional space.

3. PSO-BASED TRACKING FRAMEWORK

3.1 N Completely Independent Swarms

As previously mentioned, the basic idea in PSO is that one swarm is constantly searching for one certain kind of food. In this study, we have evolved the original idea by using n independent swarms searching for the same or for a different kind of food, depending on the scenario at hand. If two swarms are far enough from each other, they would not meet, i.e. they will keep concentrating their search for their own kind of food. Even if one bird in the first swarm has found a much bigger amount of food corresponding to the second swarm, the birds in the second swarm would not notice that as long as they stay far away from the first swarm. This first idea is very important for our framework: When swarms are far enough from each other, they cannot distract each other. Take for example two swarm of birds living in two different places on earth: if one swarm lives in California and the other lives in Munich and every bird knows only the positions and velocities of the members of his own swarm, they will never distract each other.
On the other hand, if swarms are very close to each other, they will mix together. Consequently, since every swarm is tracking one particular object, after the swarms mix together, the tracking behavior becomes totally unpredictable. To counter effect this tendency, we provide a solution completely inspired by nature. It takes into account the diversity between different species of birds living on earth. If the swarms don't like the same food, i.e. their corresponding fitness functions are not matching, they naturally do not mix together. This means that the distribution of the particles corresponding to every swarm has its peaks and valleys naturally located at different places in the search space. Imagine a swarm of ravens and a swarm of starlings: They search for a different kind of food and will never mix together even if they come very close to each other.
Thus, when the swarms are far enough from each other, where "far enough" depends on the size of the objects to be tracked, we will be successfully tracking multiple objects. If the two swarms are close to each other, the PSO has to adapt it's fitness function to the situation at hand. This reasoning applies to a large number of swarms as well.

3.2 Implementation

3.2.1 Prerequisites. As a preprocessing step, we need to filter out the movements in the video frames, without affecting the overall real-time performance of the PSO algorithm. A sufficient operation for this task is to subtract the grayscale images at time instants t and $t-1$. Doing this, we obtain a search space with n peaks corresponding to the objects in motion and showing a convergent behavior. Away from the moving objects is only a big ocean of zeros. Since the velocity of the particles is limited,

and considering that no drastic camera movements is happening, one swarm finds absolutely no way to track the wrong object.
To illustrate our idea using an example, consider having one red and one blue ball. In the initial round assuming that the balls are not moving, we make usage of an additional criteria like the hue value to initialize the tracking algorithm with the locations of the balls. The hue value is very sensitive to light changes, therefore it will only be used in the initial round or when the objects are very close to each other. Once both swarms have a starting position near the object they should track, the PSO algorithm uses only the grayscale values as long as a predefined minimum distance between the swarms is not undercut.
In the case where the two balls would have the same color, an edge detection operation along with a two-dimensional convolution to identify the number of connected components in the picture could be applied to initialize the PSO algorithm [2].

3.2.2 Self-Adapting Component. If objects are very close to each other, the grayscale values aren't enough to separate the objects, i.e. the swarms. In this case, the ocean of zeros between the two objects disappears and the particles could start to track the wrong object. Thus we introduced an adapting mechanism to decide for the next round whether the grayscale values are enough or additional information is needed. This adapting component checks the position of the global best of every swarm in the last round. If two or more global bests from different swarms undercut the predefined minimum distance, the fitness function must change over time. This is only possible after a complete search procedure happens, i.e. it's applicable to the next frame, otherwise the peaks would change their height in the search process and this is a very complex criteria for the PSO, comparable to drastic changes in the environment itself where the objects to be tracked are moving.

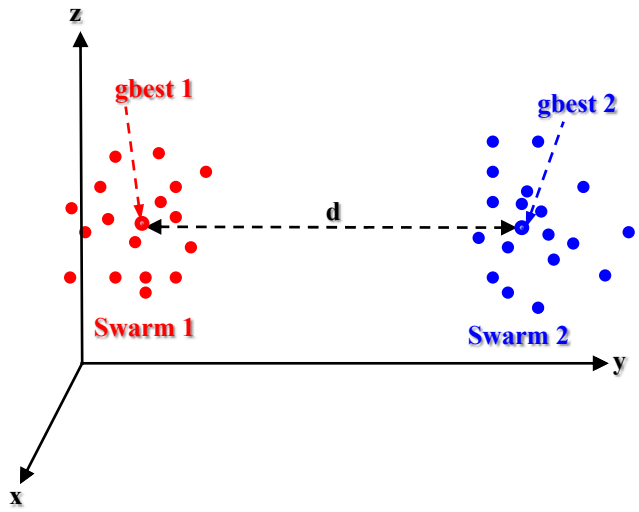

Fig. 2. Visualization example of a self-adapting scene.

It is important to notice the difference between search rounds and PSO epochs. The PSO swarms explore the search space in up to 2000 epochs to find peaks. After this procedure is complete, the next round with frame $x+1$, $x+2$ and $y+1$, $y+2$ can start. The algorithm requires the global best positions of all swarms from the last round and adaptively configures the fitness function for the next round, making it thus possible to react on different tracking situations.
Figure 2 shows how the self adapting process works: The particles of each swarm are around their global best. Depending on the distance between the two global best positions, the fitness function will be adapted for the next round.

3.2.3 Self-Splitting Component. We have presented a solution for the case the objects are far away and close to each other, now we introduce a solution for the more complex case: when one of the objects hides partly or completely behind the others. Figure *3.2.3 illustrates a self-splitting situation in billiard scene where two billiard balls move apart from each other.*
The self-splitting component checks the global best of every swarm after one round. In the case of a 2-dimensional search space, if the distance between two global bests is below m pixels, where m depends on the size of the objects, the two tracking windows will fuse into one window containing both objects. If the distance between the global best positions increases again, the search algorithm has to split the search window into two or more windows, depending on the number of moving objects in the video frames. The distance between two global bests is calculated using the following equations:

$$a = \mid gbest_x - gbest2_x \mid \quad (3)$$

$$b = \mid gbest_y - gbest2_y \mid \quad (4)$$

$$d_{gb1,gb2} = \sqrt{a^2 + b^2} \quad (5)$$

where a denotes the distance between the two global bests in the x direction, b is the distance in the y direction, $gbest_x$ and $gbest_y$ are the components from the global best from swarm 1, $gbest2_x$ and $gbest2_y$ are the x and y positions from the global best from swarm 2 and d is the distance between the global bests calculated with the Pythagorean theorem.

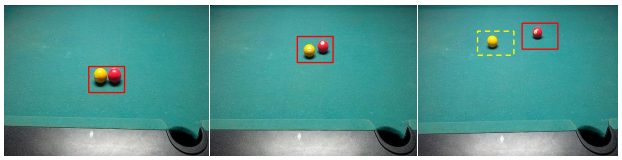

Fig. 3. Self-splitting situation in a billiard scene.

4. 1-SWARM VERSUS N-SWARM PSO

4.1 PSO With Only One Swarm

The video frames used were captured with a MD-9035 digital camera having 25 fps and a resolution of 720x576 pixel (PAL). At first we tried to implement the standard PSO with only one swarm. We subtracted two frames to get a convergent behavior, i.e. the tracked object has a continuous flow of gray values. It was possible to find and track one object, but if the starting points of the particles were very far away from the object, the PSO algorithm never found the object. We have repeated the same procedure with two objects. Although one of the two objects had much higher peaks in the two-dimensional search space, the particles had absolutely no chance to find it if they started too far away. Hence, we have forced a swarm into a near local maximum when the global maximum is far away.
Current state of the art PSO techniques use patterns or masks [10] and only one swarm may be successfully assigned a specific object describable with a certain pattern. However, these techniques are not efficient for tracking two or more independent objects, since the necessary pattern will continuously change after every time-step depending on the direction, velocity, and location of the tracked object in the environment. Many trials to track two independent objects using one swarm have failed due to the fact that this kind of experiments conflicts with the theory of PSO, which aims at finding a unique solution or tracking a single object using many particles.
Other techniques like Hidden Markov Models show acceptable results for concurrently tracking multiple objects [5]. However, these models require a well-conditioned problem and intensive

training experiments for every new tracking scenario. In addition, multiple-model Kalman filters and particle filtering techniques used for multiple object tracking necessitate a training phase and become computationally loaded when the number of tracking objects increase, or when the movement models are not linear [11].

4.2 N-Swarm PSO With Self-Adapting Fitness Function

In the initial round of the algorithm a median filter is used to reduce the noise of the first two video frames. All objects are extracted with standard image processing methods, a frame substraction along with a hue value is used in our case. After the initial round is executed every particle in the swarm is assigned a group of pixels corresponding to the object to be tracked. The size of this group of pixels depends on the size of the object, and on the distance between the tracked objects. The closer the objects are to each other the smaller is the number of pixels in the group. Therefore the number of pixels assigned to a certain particle changes from one round to the other. In our case, a total of 20 particles were sufficient for every swarm. The maximum number of training epochs per round was set to 2000. In every epoch every particle runs through (1) and (2) to get a new velocity and a new position. A total of 500 rounds were enough for a successful training.
Other PSO parameters were initialized as follows. The value for the parameter c_1, also called independent component, was set to 2 and c_2, so called social component, was set to 2 as well. The weighting factor ω was initially set to a value of 0.9 and was decreased every round according to the following equation:

$$\omega(i) = ((iw_2 - iw_1)/(iw_e - 1)) \cdot (i - 1) + iw_1 \qquad (6)$$

where $\omega(i)$ is the weighting factor at training round i, iw_2 and iw_1 are the starting and final values for the weighting factor, and iw_e is the number of rounds when the weight factor reaches its final value. The particle velocity was limited to 20 pixels per round. More about the PSO parameters can be found in [3].
It is not necessary to analyze every video frame, for most practical situations 12 fps should be enough. If the objects move around with extremely high velocity, the velocity of the particles has to be adjusted or a higher frame rate becomes necessary. When one object is partly occluded by another object, this will not affect the particle's tracking behavior since the shape of an object is not considered for tracking. The more complex case where one object is not visible anymore is dependent on the time the object is out of the picture: if the absence of the object is very short, the swarm finds the object again when it appears in the video sequence. This is a very complex issue with other tracking technologies like HMM or Kalman Filters.
In the case where both objects are far from each other, the fitness function takes the following form:

$$f_{x,y} = frame_{x,y} \qquad (7)$$

where $frame_{x,y}$ denotes the grayscale frame at coordinates x and y in the search space. When the objects start approaching each other, and the hue value of the pixels in the search space is consistent with the object to be tracked, the fitness function takes another form:

$$f_{x,y} = \alpha \cdot frame_{x,y} \qquad (8)$$

where α is a positive rational weighting factor greater than 1. In our case the value of α increases when hue value of the tracked object approaches the searched hue value. Thus if the point in the search space has the searched color or a color close to it, the fitness value will increase dramatically for this space. If two objects are very close to each other, the algorithm uses the color to separate the objects. A stand alone usage of the hue value is not recommended, as it is very sensitive to light changes. The combination of the grayscale and hue values makes sure that the wanted object is distinguishable from other objects.

4.3 Comparison of Results

To examine our new algorithm, we have assigned every tracked object a tracking window. We then measured how accurate does the tracking window fit and follow the object. After every round of the algorithm, we compared the tracked object with the actual object. If the object is completely inside the window, an assessment value of 10 is achieved. When the assessment value is 8.6, only 86 percent of the object are found inside the boundaries of the tracking window. The lowest assessment value is 0 and corresponds to the case where the object is totally outside the tracking window. We tested our n-swarm PSO algorithm with two billiard balls randomly moving on a table.

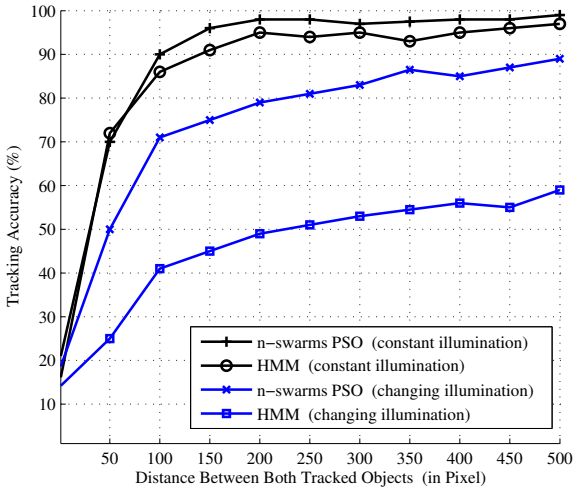

Fig. 4. Tracking accuracy of the n-swarm PSO algorithm as compared to HMMs under constant and changing light conditions.

In the first part of the experiment, the light conditions of the video recording did not change. The size of each ball in the video sequence was 40x40 pixels, thus if the distance between the 2 global best positions of the swarms is lower than 50 pixels, the tracking assessment is hard because one object hides behind the other. In such a case, we assessed only the visible part of the object. Hence, the tracking accuracy is lower when the distance between the two objects is less than 50 pixels. This is illustrated in the left part of the "+"-marked line in figure 4.
In the second part of the experiment, the light conditions were drastically changing from a video frame to the other. The n-swarm PSO algorithm was tested again under these conditions. While the general performance dropped down as seen in the "x"-marked line of figure 4, it's an ameliorated performance compared to the HMM method applied and trained for the same scenario and under the same conditions. It's worth mentioning here that in addition to the degraded performance of HMMs, they have required intensive training in an offline phase prior to the live experiment. On the other hand, existing PSO-based methods are not directly comparable with our technique because they are tuned to track only one object or a predefined pattern of objects.
In a further experiment, the n-swarm PSO algorithm was tested in three different scenarios. The first one consists of a billiard scene with two randomly moving balls. The second scenario is a scene where the two balls are forced to move up and down with different velocities like marionettes, and where the light conditions are constantly changing. The third scenario is a scene with two persons walking and only their heads were tracked. Table 1 illustrates the percentage of correct localization of the n-swarm

Fig. 5. Tracking of multiple subjects: The self-splitting PSO automatically generates new swarms to new or separating subjects in real-time.

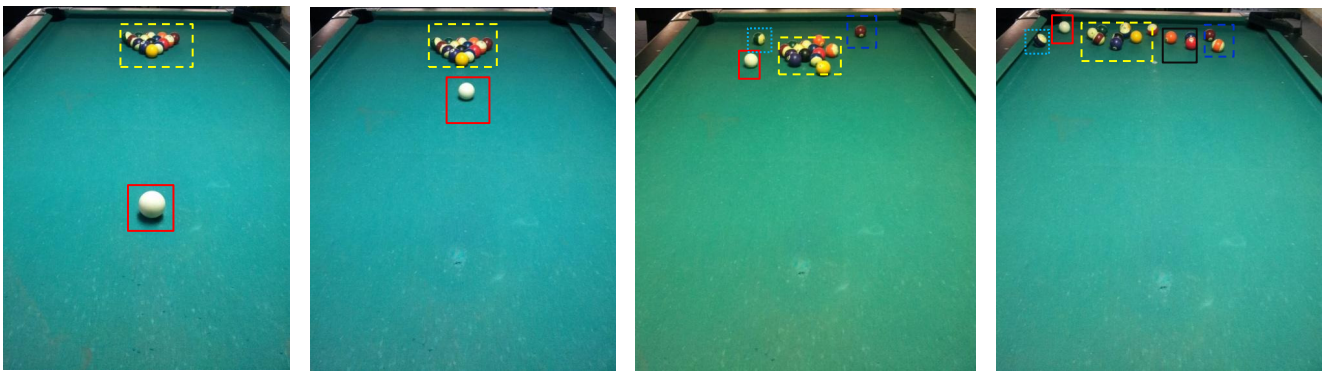

Fig. 6. Tracking of multiple billiard balls: The self-splitting PSO instantaneously assigns new swarms to those scattering balls just after break-in.

algorithm for every scenario. Compared to the HMM results for tracking parts of people's body in [5], our algorithm showed similar results for the head tracking experiment. Nevertheless, our algorithm was remarkably outperforming for the case where the light conditions and the velocity of the objects drastically changed.

Finally, we set our self-splitting n-swarm PSO to track several walking persons in a hallway, where occlusions and interactions occurs as illustrated in figure 5. It is seen that the tracker can robustly and accurately track the randomly moving subjects over all frames. One can observe how self-splitting is occuring when the two persons walking close to each other in frame 4 separate in frame 5. Furthermore, the tracker was applied to track the high-speed movement of different billiard balls on a pool table after break-in. The results are illustrated in figure 6. Even under high speed impact conditions, the n-swarm PSO algorithm was able to instentaneously create new swarms on-the-fly for each of those balls which separated from the assembly of the others balls immediatly after the break-in.

Table 1. Percentages of Correct Localization (%).

Sequence	Object 1	Object 2	HMM [5]
Billiard (constant illumination)	97.3	93.7	92.2
Marionette Balls (changing illumination)	94.1	92.8	86.7
People walking (changing illumination)	95.6	98.2	87.3

5. CONCLUSION

In this paper we introduced a new tracking algorithm based on particle swarm optimization. The algorithm outperforms state of the art techniques by making it possible to track n completely independent objects using PSO. In comparison with other tracking techniques like HMMs and Kalman filtering, the n-swarm PSO algorithm is more robust to changing light and velocity conditions, has fewer computational operations, fewer defining parameters, and can be coded in just a few lines, providing thus an easy real-time implementation on a DSP platform. Applications of the algorithm are projected to be in the fields of concur-

rent object recognition and tracking, medical imaging, as well as machine learning.
Based on our approach to multiple object tracking, many venues for future work are to be considered. A valuable step in this direction is to find a generalized form of an adaptive fitness function suitable for every dimensions and every color of the objects to be tracked in real-time.

6. REFERENCES

[1] *B. Babenko, M. Yang, and B. Belongie. Robust object tracking with online multiple instance learning.* IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(8):1619–1632, 2011.*

[2] *J. Canny. A computational approach for edge detection.* IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.*

[3] *A. Carlisle and G. Dozier. An off-the-shelf PSO. Proc. Workshop on Particle Swarm Optimization, Indianapolis, 2001.*

[4] *D. Comaniciu and P. Meer. Mean shift: a robust approach toward feature space analysis.* IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.*

[5] *D. Forsyth D. Ramanan and A. Zisserman. Tracking people by learning their appearance.* IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(1):65–81, 2007.*

[6] *X. Zhang et al. Multiple object tracking via species-based particle swarm optimization.* IEEE Transactions on Circuits and Systems for Video Technology*, 20(11):1590–1602, 2010.*

[7] *D. Goldberg.* Genetic algorithms in search, optimization and machine learning. *Reading, MA: Addison-Wesley, 1989.*

[8] *J. Kennedy and R. C. Eberhart. Particle swarm optimization.* Proc. IEEE International Conference Neural Networks*, 4:1942–1948, 1995.*

[9] *J. Pugh and A. Martinoli. Inspiring and modeling mulit-robot search with particle swarm optimization. In* Proc. IEEE Swarm Intelligence Symposium (SIS 2007)*, pages 332–339, 2007.*

[10] *M. Scheutz. Real-time hierarchical swarms for rapid adaptive multi-level pattern detection and tracking. In* Proc. IEEE Swarm Intelligence Symposium (SIS 2007)*, pages 234–241, 2007.*

[11] *G. Welch and G. Bishop. An introduction to Kalman filter. ACM SIGGRAPH 2001 Course #8., year =.*

[12] *Z. Zhang, H. Seah, and J. Sun. A hybrid particle swarm optimization with cooperative method for multi-object tracking.* Proc. IEEE world Congress on Evolutionary Computation (CEC)*, 4:1–6, June 2012.*

[13] *Y. Zheng and Y. Meng. The PSO-based adaptive window for people tracking. In* Proc. IEEE Swarm Intelligence Symposium (SIS 2007)*, pages 23–29, 2007.*