Science World Journal Vol 13(No 1) 2018
www.scienceworldjournal.org
ISSN 1597-6343

Full Length Research Article

OPTIMIZATION OF OBJECT TRACKING BASED ON ENHANCED IMPERIALIST COMPETITIVE ALGORITHM

[1]Luhutyit Peter Damuut and [1]Jakada Dogara

[1]Department of Mathematical Sciences, Kaduna State University, Kaduna- Nigeria

ABSTRACT
Object tracking is one of the most challenging tasks in the field of computer vision. Tracking moving object(s) in video/image frame sequences in cluttered scenes usually results in complications and hence performance degradation. This is attributable to complexity in partial and full object occlusions and scene illumination changes which render object tracking complicated besides the delay in processing of moving images from frame to frame as well as the presence of multiple objects in the video frames under consideration. This paper explores the use of Enhanced Imperialist Competitive Algorithm (EICA) to track moving object(s) in video frames. The results obtained reveal the usefulness of this approach and provide the needed stimulus for further research in the problem domain.

Keywords: Imperialist, Optimization, Tracking, Colony, Object

1. INTRODUCTION

The development in the technology of programming languages is In political history, the term empire has over the years been used to explain the extending control of a stronger nation over weaker one(s). A typical example is the Roman Empire which had influence or control over most parts of the world from 27BC to 476BC. The most powerful nation or person is usually called an imperialist. Among imperialists with overriding authority and control over one or more nation states, there is usually fierce competition for more control and influence (a socio-political evolution process) serving as inspiration for the evolutionary algorithm called Imperialist Competitive Algorithm (Atashpaz & Lucas, 2007).

Initially, due to the quest for wealth, nations were focused on building empires based on availability of natural resources such as gold and silver. However, nowadays the new imperialism is focused on technological advances. The ultimate goal is to increase the number of their colonies and spreading their empires all over the world.

This paper therefore seeks to implement an enhanced version of Imperialist Competitive Algorithm (ICA) using frames generated from video files from where object(s) of interest would be effectively and efficiently tracked.

Equally important is the need to investigate the performance of the Enhance Imperialist Competitive Algorithm (EICA) in optimizing the generated video frames by comparative analysis of the key performance indicators.

The rest of the paper is organized as follows: Section 2 presents a review of related literature; Section 3 explains the problem formulations; Experimental results and discussions are presented in Section 4. The paper is concluded in section 5 with a summary of findings as well as recommendation for further research on the subject.

2.0 Literature Review

The researchers in (Barga & Dalton, 2014) examine the state of the art in tracking methods such as Mean-shift, Kalman filter, Particle filter among others. Specifically, this work is focused on feature-based object tracking. Here, object tracking is accomplished in three steps. The first step involves extracting the features of the object of interest. The second step deals with clustering the extracted features of the object while the third step matches the extracted features in successive image frames. Feature extraction and correspondence are the important steps in this tracking method. The challenge in feature-based tracking however, is feature correspondence because a feature point in one image may have many similar points in another image, hence resulting in ambiguity.

Similarly, (Dhara & Chintan, 2015) developed an intelligent framework for real-time object motion detection. This work combines the Gaussian Mixture Modeling (GMM) and Optical Flow (OF) object detection and tracking method. While GMM proves more useful in the context of complex environments, OF is more suitable for faster calculations. Here, the two methods are combined for a more accurate detection and tracking of moving objects

The authors of (Wu, Jongwoo, & Ming-Hsuan, 2015) identified that in several tracking algorithms, datasets often do not have common ground-truth object positions, resulting in difficulty when comparing the performances of the algorithms. In addition, the initial conditions or parameters of the evaluated tracking algorithms are not often identical. To address these issues, most of the publicly available tracking algorithms are often integrated into one code library with uniform input and output formats to facilitate large-scale performance evaluation. Based on the benchmark experiments conducted, some key components essential for improving the tracking performance are identified namely: background information, local models, motion models and state prediction models which can significantly reduce the search range and thus improve the tracking efficiency and robustness.

An algorithm is proposed by (Tripty, Sanju, & Bichu, 2014) for the detection of moving objects in highly secured environments. With the objective of increasing the efficiency of the detection of moving object in both offline and online video modes, the algorithm is tested with input AVI format video file of 320 x 240 frame size and frame rate of 15fps in offline mode. In online mode however, video file of 640 x 480 frame size and frame rate 2of 5fps is captured in real time using a mounted webcam.

Using F-tree method (Prakash & Shalini, 2011) proposed an algorithm for achieving better and faster frame object tracking based on video frame sequence. The method uses frame differentiation (i.e., sequential, algorithmic and random), segmentation and tracking based on mathematical models. Figures 2.1 and 2.2 respectively illustrate this concept.

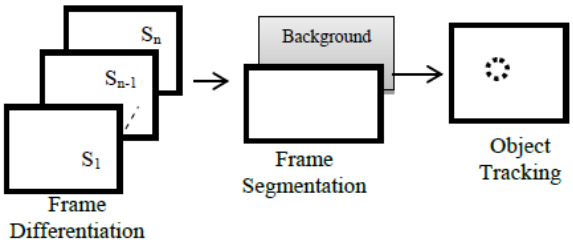

Figure 2.1: Basic Block- Object Video Tracking (Prakash & Shalini, 2011)

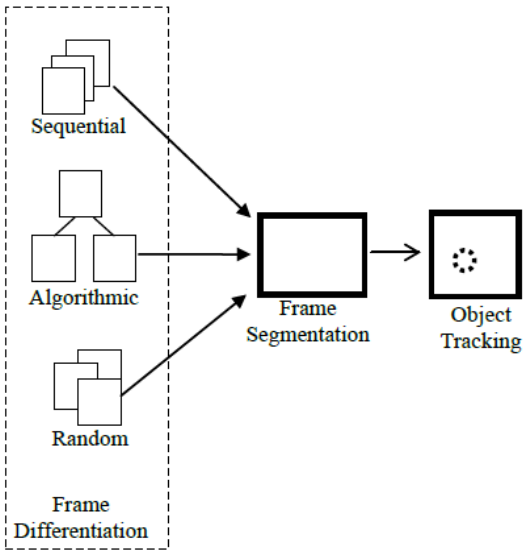

Figure 2.2: Frame Differentiation Methods (Prakash & Shalini, 2011)

Graph theory is used by (Hodais, Majid, & Mehrdad, 2013) for image segmentation in object tracking. Here, graph partitioning is carried out using Imperialist Competitive Algorithm (ICA).

In the work titled 'An Enhanced Imperialist Competitive Algorithm for Optimum Design of Steel Frames' (Mahmoud, Maheri, & Talezadeh, 2015), an algorithm is developed in order to optimize the search for an escape route from local optima. In this algorithm, added value is given to a slightly unfeasible solution, based on its distance from the relative imperialist. The algorithm performs well in terms of design quality in larger structures, indicating its usefulness in solving practical structural engineering problems.

There are primarily two sources of information that can be used for detection and tracking of objects in a video frame: visual features (e.g. color, texture and shape) and motion information. Robust approaches have been suggested by combining the statistical analysis of visual features and temporal analysis of motion information. An intuitive approach could be to first, segment the video frame into a number of regions based on visual features like color and texture, subsequently merging of regions with similar motion vectors can be performed subject to certain constraints such as spatial neighborhood of the pixels (Shaikh, Saeed, & Chaki, 2014).

3.0 Problem Formulation

In the basic Imperialist Competitive Algorithm (ICA), the countries represent the population which is basically divided into two groups (i.e., Colonies and Imperialists) based on their strength or fitness. Also, an empire is formed by one imperialist together with at least one colony. Furthermore, two operators namely, assimilation and revolution constitute the strategy called Imperialist Competition which is the key building block in basic ICA (Zaynab, Mehdi, & Seyyed, 2014).

The ICA algorithm is described as follows:

Initializing Phase:

i. Preparation of initial population. Each solution (i.e., country) in form of an array can be defined by equation (1):

$$\text{Country} = \{p_1, p_2, \ldots\ldots\ldots\ldots\ldots, p_N\} \quad (1)$$

Where p_i represents different variables based on various socio-political characteristics (such as culture, language, and economical policy), and N denotes the total number of the characteristics (i.e., n- dimension of the problems) to be optimized.

ii. Creating the cost function. In order to evaluate each country, the cost function can be defined using equation (2):

$$\text{Cost } t = f(\text{Country}) = f(p_1, p_2, \ldots\ldots\ldots\ldots, p_N) \quad (2)$$

iii. Initializing the empire: in general, the initial size of populations (Npop) involves two type of countries [i.e., colony (Ncol) and Imperialist (N_{imp})] which together form each empire. To form the initial empire proportionally, the normalized cost of an imperialist is defined through equation below

$$NC_n = C_n - \text{Maxi}\{C_i\}, \quad (3)$$

Where C_n is the cost of the nth imperialist, NC_n denotes its normalized cost. Normally, two methods can be used divide colonies among imperialist: (1) from the imperialist point of view which is based on the power of each imperialist and (2) from the colonies point of view which depends on its relationship with any of the imperialists (i.e., the colonies should be possessed by the imperialist according to their respective powers).

The EICA Pseudo-code for Tracking Moving Object(s) in Video Frames

1. Initialization;
2. Input video ;
3. Generate 50 frames (countries) from video;
4. Choose N_{Imp} fitter countries as Imperialists; // using (1)
5. Evaluate costs of all countries (10 Imperialist frames) // using (2)
6. Allocating colonies to empires based on empires' powers.
7. **While** stoppage criterion not satisfied **do**
8. Move colonies towards their corresponding Imperialists;
9. Evaluate costs of all colonies in the empires;
10. **If** a colony fitter than its corresponding Imperialist **then**

Swap the positions of the colony and the Imperialist (possession); // using (3)

11. End If

12. Randomly replace certain percentage of colonies in each empire;
13. Take a colony from the weakest empire and give it to a stronger one (imperialist competition);
14. **If** there **exists** more than one empire **then**
15. **P**erform revolution operation;
16. Update the imperialists in the empires;
17. Generate a random value μ - μ(0,1);
18. **If** an empire has no colonies **then**

Eliminate that empire;

19. End If

20. End If

21. **End While**

The Pseudo-code for Selecting Imperialist Frames

1. Sort all empire frames in sequential order of their costs (P_1, P_2, P_3,------ P_m);
2. Construct anchor frame Imperialist $P_1 = P_a$
3. **If** $f(P_a)<f(P_m)$ **then**
4. P_a replaces P_m as the Imperialist of empire m;
5. **Else if**
6. P_m is the final Imperialist of the sorted empire frames
7. **End if**

The Pseudo- code for Crossover of Imperialist frames

1. **For** $k=1$ *to* $[P^m]$ **do**
2. Randomly choose two Imperialists P_1 and P_2 from all Imperialists assuming $f(P_1)\leq f(P_2)$;
3. crossover P_1 and P_2 to generate new empire
4. Assign the best of P_1 , P_2 as the imperialist of the empire i;
5. **End for**

4. Results and Discussions

This section presents experimental setup, implementation results and performance comparisons. Using MATLAB R2014a as a simulation tool, the experiments are implemented using an Intel (R) Pentium (R) processor CPU clock speed of 2.20GHz, installed memory (RAM) of 4.00GB in a 32- bit windows operating system.

4.1. Frame Extraction

The frames extraction was carried out on three video files, one indoor and two outdoor as described in Table 1 below:

Table 1: Parameters and Values for the Experiments

Parameter	Outdoor	Indoor	Outdoor
Video Filename	dogara4.avi	Surprise.mp4	dogaraabu.mp4
Length	00:00:40	00:00:36	00:00:33
Size	3.08 MB	512KB	5.35 MB
Frame Rate	30 frames/second	25 frames/second	29 frames/second

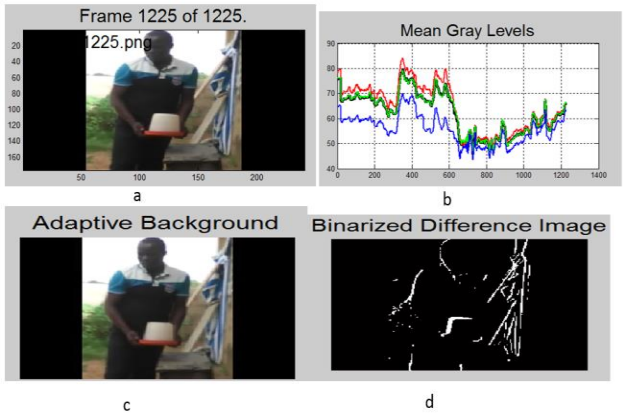

Figure 4.1: Extraction of video frames from the video file "dogara4.avi" (outdoor scenario).

(a) Original video frames
(b) Generated frames mean gray levels
(c) Adaptive background and
(d) Binarized difference image

Table 2: Summary of multimedia reader object for the video file 'dogara4.avi'

Video parameter	30.00 frames per second
RGB 24	240 x 176
Video frames	1225
Structural array	1225

- Total number of frames generated is 1225, and by presenting 1225 countries in EICA
- The structural array is 1225 x 1 one dimensional array meaning it generates the frames in sequential order.
- One movable object (a person) is tracked from one frame position to another. The change in object position and orientation is indicated by the mean gray levels (Figure 4.1 b).

Experiment 2

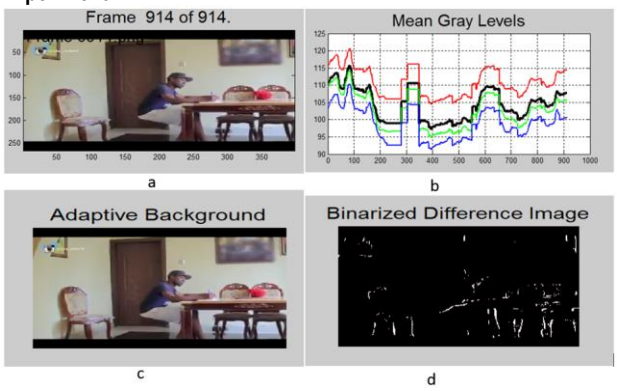

Figure 4.2: Extraction of Video Frames from video file "surprise.mp4" (indoor scenario)

(a) Original video frames
(b) Generated frames mean gray levels
(c) Adaptive background and
(d) Binarized difference image

Table 3: Summary of multimedia reader object for 'surprise.mp4'

Video parameter	25.00 frames per second
RGB 24	398 x 266
Video frames	914
Structural array	914 x 1

- Total number of frames generated is 914, presenting 914 countries.
- The structural array is 914 x 1 one dimensional array meaning it generates the frames in sequential order.
- Three movable objects (2 people and a chair) were tracked from one frame position to another. The change in object position and orientation is indicated by the mean gray levels (Figure 4.2 b).

Experiment 3

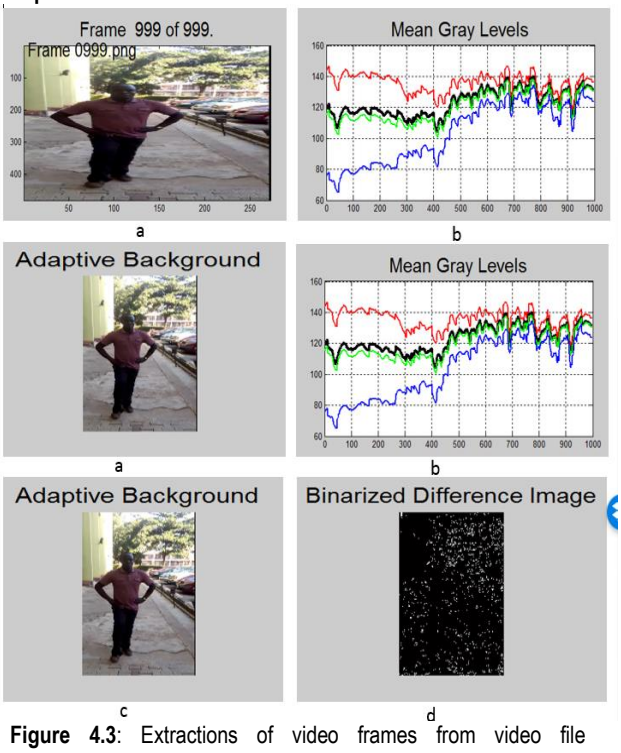

Figure 4.3: Extractions of video frames from video file "dogaraabu.mp4" (outdoor scenario).

(a) Original video generated to frames
(b) Generated frames mean gray levels
(c) Adaptive background and
(d) Binarized difference image

Table 4: Summary of multimedia reader object for video file "dogaraabu.mp4"

Video parameter	29.94 frames per second
RGB 24	272 x 480
Video frame available	999
Structural array	999x 1

- Total number of frames generated is 999, presenting 999 countries.
- The structural array is 999 x 1 one dimensional array meaning it generates the frames in sequential order.
- One movable objects (a person) is tracked from one frame position to another. The change in object position and orientation is indicated by the mean gray levels (Figure 4.3).

The same extraction operation was performed on all the three scenarios with noticeable differences in video parameters and in RGB values. In case of experiment 1, the video parameter is 30.00 frames per second and RGB 24 is 240 x 176. In experiment 2, the video parameter is 25.00 frames per second and RGB 24 is 398 x 266. In experiment 3, the video parameter is 29.94 frames per second and RGB 24 is 272 x 480. These variations in video parameter and RGB 24 have significant impact on the target object(s) of interest in a particular tracking scenario and also help to determine which tracking algorithm is most suited for a video shot. Therefore, RGB 24 is one of the major factors that affects the object orientation, appearance, shape and size and also can help in determining the effectiveness and efficiencies of a tracking procedure.

Furthermore, the processing speed and hence time taken to extract the video frames in object tracking were collated. Table 5 presents the summary of the comparative performances of the three experiments using the three video files whose parameters are shown in Table 1

Table 5: Summary of CPU time for frame extraction from the three video files

	dogara4.avi video			surprise.mp4 video			dogaraabu.mp4 video		
Function Name	**Calls**	**Total time (s)**	**Self-time (s)**	**Calls**	**Total time (s)**	**Self-time (s)**	**Calls**	**Total time (s)**	**Self-time (s)**
Extract Movie Frames	1	334.856	53.361	1	290.907	45.617	1	284.53	45.159
Image show	2450	90.379	19.441	1828	72.787	15.407	1998	64.350	12.385
Get frame	1225	64.779	62.694	914	44.458	42.888	999	46.968	45.406
New plot	11025	52.614	3.715	8226	41.673	2.831	8991	38.617	2.647

In Table 5, the Calls field indicates the number of times *ExtractMovieFrames, ImageShow, Getframe* and *Newplot* functions respectively were called by the CPU. Total time indicates the time duration spent in executing a function/procedure. Self-time is the time spent by a function excluding the time spent executing its child functions. Self-time also includes overhead resulting from profiling.

Experiment1: In the video file "dogara4.avi", the CPU spent 334.856s as total time to extract movie frames and 2450 *ImageShow* function calls were undertaken.

Experiment2: video file "surprise.mp4" the CPU spent 290.907s as total time to extract movie frames and 1828 *ImageShow* function calls were undertaken.

Experiment3: video file "dogara.mp4" the CPU spent 248.53s as total time to extract movie frames and 1998 *ImageShow* function calls were undertaken

Table 6: Comparison of the results of EICA based on variations in values for assimilation coefficients and revolution rates respectively.

		dogara4.avi		surprise.mp4		dogaraabu.mp4	
	Parameters and Values	**Running cost (s)**	**CPU times (s)**	**Running cost (s)**	**CPU times (s)**	**Running cost (s)**	**CPU times (s)**
1	*Revolution Rate = 0.3*	38.8162	46.966	$6.7832e^{-05}$	28.095	0.0000	29.428
	Assimilation coefficient = 2						
2	*Revolution Rate = 0.1*	77.6068	46.020	24.8739	28.474	1.9899	29.520
	Assimilation coefficient = 1						
3	*Revolution Rate = 0.1*	62.6820	46.739	2.9849	27.997	0.0000	29.420
	Assimilation coefficient = 1.5						
4	*Revolution Rate = 0.5*	6.0046	46.492	0.0573	28.495	0.0000	29.353
	Assimilation coefficient = 2						
5	*Revolution Rate = 0.5*	41.7882	46.504	3.9798	28.284	0.0000	28.994
	Assimilation coefficient = 1.5						

Table 6 shows the result of running EICA on video frames generated from the file ***dogara4.avi***, with the best result as 6.0046 when the revolution rate is 0.5 and assimilation coefficient is 2, representing the optimal result.
Figure 4.5 shows the plot of costs against the frame iteration runing time

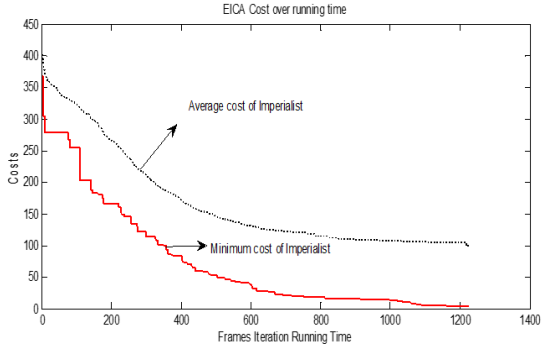

Figure 4. 5: The costs over running time with revolution rate = 0.5 and assimilation coefficient = 2

It is obvious from Figure 4.5 that based on the given parameters, it took EICA about 1200 iterations to return the minimum cost of the last imperialist indicating the termination of the algorithm.
Figure 5.6 shows the plot of mean cost agsinat running time for the video file ***surprise.mp4***.

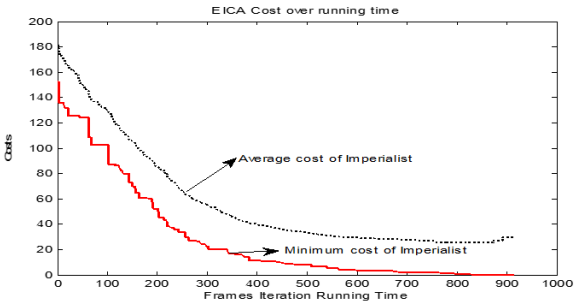

Figure 4. 6: The costs over running time with revolution rate = 0.1 and assimilation coefficient = 1.5

To meet the termination criteria, EICA iterated about 900 times (Figure 1.6).
Figure 4.7 shows the plot of the mean cost agsiant running time for the video file ***dogaraabu.mp4***

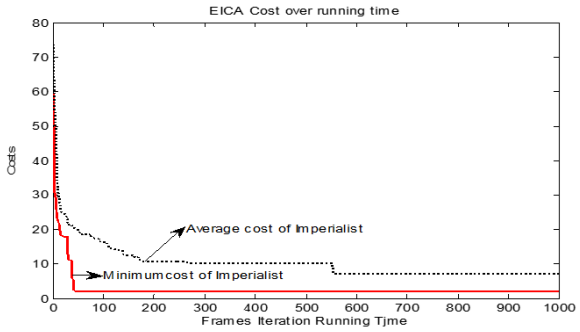

Figure 4.7: The mean costs plot agaisnt running time with revolution rate = 0.1 and assimilation coefficient = 1.00

5. Conclusions

In this paper, the first objective is to design and simulate the Enhanced Imperialist Competitive Algorithm model using assimilation and revolution operations. This objective was fully achieved where the generated frames gotten from the converted recorded video file into JPG frames representing countries. Systematic sampling method (K = N/n) was used in the selection of the Imperialist and colonies frames. The assimilation and revolution operations were performed on both the Imperialist and Colonies video frames respectively.
The second objective is to investigate the performance of the Enhance Imperialist Competitive Algorithm (EICA) in optimizing the generated video frames and to calculate the running time complexity of the algorithm. To achieve this objective the 10 selected Imperialist video frames were represented in memory as one-dimensional array because the frames were sequentially selected.
The third objective is to carry out comparative analysis, evaluation and measurement of the performance of EICA. In order to achieve this, Kalmar filter algorithm was employed in order to enable the algorithm optimizes the tracking of moving objects in both indoor and outdoor scenarios accordingly.
Specifically, the paper has demonstrated the optimization of object tracking using video frames by extending the original Imperialist Competitive Algorithm to use Kalman filter coding in order to significantly reduce noise in outdoor video settings.

Acknowledgements
Special appreciation goes to almighty God for providing the needed inspiration as well as the wherewithal to start and conclude this work. Equally worth mentioning is the invaluable contribution of family, friends and colleagues for positive criticism of the work.

REFERENCES

Atashpaz, G., & Lucas, C. (2007). Imperialist Competitive Algorithm for Optimization Inspired by Imperialist Competition.

Barga, D., & Dalton, T. (2014). A Suryer on Moving Object Tracking In Video. *International Journal on Information Theory(IJIT)*, 32-43.

Dhara, T., & Chintan, V. (2015). A Review on Moving Object Detection and Tracking Methods. *International Journal of Advance Engineering and Research Development (IJAERD)*.

Hodais, S., Majid, V., & Mehrdad, J. (2013). Graph-Based Image Segmentation Using Imperialist Competitive Algorithm. *Advances in Computing*, pp. 11-21.

Mahmoud, R., Maheri, & Talezadeh, M. (2015). An Enhanced Imperialist Competitive Algorithm For Optimum Design Of Steel Frames. *Implementing Innovative Ideas in Structural Engineering and Project Management*, pp. 520-524.

Prakash, K., & Shalini, S. (2011). Object Tracking in Video Sequences Based on Background Updation Using F-Tree Method. *International Conference on Computational Techniques and Artificial Intelligence.*

Shaikh, S. H., Saeed, K., & Chaki, N. (2014). Moving Object Detection Using Background Subtraction . *Briefs in Computer Science*.

Tripty, S., Sanju, S., & Bichu, V. (2014). A New Algorithm Designing for Detection of Moving Objects in Video.

International Journal of Computer Applications.

Wu, Y., Jongwoo, L., & Ming-Hsuan, Y. (2015, September). Object Tracking Benchmark. *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, pp. 1834-1846.

Zaynab, B., Mehdi, S., & Seyyed, J. (2014). Imperialist Competitive Algorithm for Improving Edge Detection. *International Journal of Advanced Research in Computer Science and Software Engineering*, 227-230.