Information Sciences 450 (2018) 246–266

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

A Hybrid Harmony search and Simulated Annealing algorithm for continuous optimization

Assif Assad*, Kusum Deep

Department of Mathematics, Indian Institute of Technology Roorkee, Roorkee, 247667, India

ARTICLE INFO

Article history:
Received 4 April 2017
Revised 15 March 2018
Accepted 16 March 2018
Available online 20 March 2018

Keywords:
Hybrid algorithms
Harmony search
Simulated annealing
Meta-heuristics
Evolutionary algorithms
Optimization

ABSTRACT

Harmony search is a powerful metaheuristic algorithm with excellent exploitation capabilities but suffers a very serious limitation of premature convergence if one or more initially generated solutions/harmonies are in the vicinity of local optimal. In order to remove this limitation this paper proposes a novel algorithm based on hybridization of Harmony search and Simulated Annealing called HS-SA to inherit their advantages in a complementary way. Taking the inspiration from Simulated Annealing the proposed HS-SA algorithm accepts even the inferior harmonies with a probability determined by parameter called Temperature. The Temperature parameter is initially kept high to favor exploration of search space and is linearly decreased to gradually shift focus to exploitation of promising search areas. The performance of HS-SA is tested on IEEE CEC 2014 benchmark functions and real life problem from computer vision called Camera Calibration problem. The numerical results demonstrate the superiority of the proposed algorithm.

1. Introduction

Optimization is defined as the process of selecting the best option from a set of available alternatives. Every process has the potential to be optimized and many challenging problems in business, economics, science and engineering can be formulated as an optimization process. For example, the objective of formulated optimization problems can be the maximization of profit and/or quality, or minimization of time, cost and risk.

Many real life optimization problems are complex and thus difficult to solve in an exact manner within reasonable amount of time. The classical optimization methods have the limitation of being highly sensitive to the initial guess and may frequently converge to a local optimum. Metaheuristic algorithms eradicate some of the afore-mentioned difficulties and are quickly replacing the classical methods in solving complex non linear optimization problems. Metaheuristic algorithms typically intend to find a reasonably good solution close to optimal in reasonable amount of computational time. During the last few decades, several metaheuristic algorithms have been proposed including Genetic Algorithms, Particle Swarm Optimization, Evolutionary Programming, Genetic Programming, Differential Evolution, Ant Colony Optimization, Evolutionary Strategies (ES). Developed by Geem et al. in 2001 the Harmony Search (HS) algorithm is the musicians inspired metaheuristic algorithm [1] that has found applications in diverse fields.

The efficiency of a metaheuristic algorithms depend on the extent of balance between diversification and intensification during the course of the search. Intensification also referred as exploitation is the ability of an algorithm to exploit the search

* Corresponding author.
E-mail addresses: assifassad@gmail.com (A. Assad), kusumdeep@gmail.com (K. Deep).

https://doi.org/10.1016/j.ins.2018.03.042
0020-0255/

space in the vicinity of the current good solution while diversification also called exploration is the process of exploring the new regions of search space thus allows dissemination of the new information. Proper balance between these two contradicting characteristics is a must to enhance the performance of an algorithm.

Based on the idea of balanced intensification and diversification this article describes a new variant of HS that synergistically incorporates some features of another very powerful optimization algorithm called Simulated Annealing with a view of improving the accuracy and robustness of Harmony Search. Hybridization of intelligent systems is a promising research field of modern artificial intelligence concerned with the development of next generation intelligent systems. Hybrid intelligent systems are becoming popular due to their capabilities in handling many complex real world problems involving uncertainty, imprecision and vagueness [2].

Taking the inspiration from Simulated Annealing the HS-SA algorithm accepts even the suboptimal solutions than the ones stored in its Harmony memory with some probability determined by parameter Temperature. The probability of accepting the suboptimal solutions is gradually decreased by decreasing Temperature parameter during execution so as to enhance the exploration capabilities of the algorithm in the earlier generations and exploitation towards the later generations.

To evaluate the performance of HS-SA comprehensively, it is tested on all the thirty IEEE CEC 2014 benchmark functions [3] and a real life problem from computer vision called Camera Calibration- a highly nonlinear twelve dimensional optimization problem. The numerical results obtained demonstrate the superiority of the proposed algorithm in terms of accuracy and robustness.

2. Harmony Search and Simulated Annealing

In this section an introduction to Harmony Search and Simulated Annealing is provided.

2.1. Harmony Search and its variants

Harmony Search (HS) is a musicians behavior inspired evolutionary algorithm developed in 2001 by Geem et al. [1], though it is a relatively new meta heuristic algorithm, its effectiveness and advantages have been demonstrated in various applications like traffic routing , multi objective optimization, design of municipal water distribution networks, load dispatch problem in electrical engineering, rostering problems, clustering, structural design, classification and feature selection to name a few. A detailed survey on applications of HS can be found in [4,5].

Weyland [6] raised an issue regarding the novelty of Harmony Search algorithm by declaring it a special case of $(\mu + 1)-$ES, however the pitch adjustment operator used in HS is entirely different than the mutation operator used in ES. Further HS utilizes the pitch adjustment operator (local search) probabilistically in contrast to ES's mutation operator and thus the two can't be considered same. Ample evidence has been provided by Saka et al. in [7] to show HS is not a special case of $(\mu + 1)-$ ES even though superficially they seem to be identical.

In order to explain the Harmony Search in detail, let us first idealize the improvisation process by a skilled musician. When a musician is improvising there are three possible choices:

1. Play any piece of music exactly from his memory.
2. Play something similar to a known piece.
3. Compose new or random notes.

Geem et al. [1] formalized these three options into quantitative optimization process and the three corresponding components become usage of harmony memory (HM), pitch adjusting, and randomization. The usage of HM is similar to the choice of the best fit individuals in genetic algorithms. In order to use this memory effectively, it is typically assigned a parameter called harmony memory considering rate (HMRC $\in$ [0, 1]). If HMRC is low best harmonies are not exploited well and if this rate is high solution space is not explored properly. Typically HMRC $\in$ [0.7, 0.95] is used as recommended in [8]. The second component is pitch adjustment determined by a pitch bandwidth (BW) and a pitch adjusting rate (PAR), it corresponds to generating a slightly different solution than the existing one. Pitch can be adjusted linearly or nonlinearly however most often linear adjustment is used. So we have

$$H_i^{new} = H_i^{old} \quad + \quad BW \ \times \ r_i \quad where \quad r_i \in [-1, 1] \quad and \quad 1 \leq i \leq D \tag{1}$$

Where H_i^{old} is the ith component of the existing harmony or solution and H_i^{new} is the ith component of new harmony after the pitch adjusting action. The Eq. (1) essentially produces a new solution around the existing solution by altering it slightly. Here r_i is a uniformly distributed random number generated in the range of $[-1, 1]$ and D is total number of components in the harmony. Yang [8] recommends the value of PAR $\in$ [0.1, 0.5]. The third component of the HS is the randomization, which is used to increase the exploration of the search space. Although pitch adjustment plays a some what similar role, but it is confined to close neighborhood of harmony and thus corresponds to local search. The use of randomization helps the algorithm to further explore diverse search areas to find the global optima. The pseudo code of harmony search is shown as Algorithm 1. In the pseudo code, H represents a potential solution or harmony, rand $\in$ [0, 1] is a uniformly distributed random number generator, rand_int(1,HMS) generates a uniformly distributed integer random number between 1 and HMS, where HMS is the size of harmony memory and D is the dimension of problem.

Algorithm 1 Harmony Search Algorithm (HSA).

```
1: Define Objective function f(H).
2: Define harmony memory consideration rate (HMCR).
3: Define pitch adjustment rate (PAR) and bandwidth(BW).
4: Define Harmony Memory Size (HMS).
5: Initialize Harmony Memory (HM).
6: while (Stopping Criteria Not Reached) do
7:    Find current Worst and Best harmony in HM.
8:    for i = 1 to D do
9:       if (rand ≤ HMCR) then
10:         H_i = HM_i^j where j=rand_int(1,HMS)
11:         if (rand ≤ PAR) then
12:            H_i = H_i ± rand × BW
13:         end if
14:      else
15:         Generate H_i randomly within the allowed bounds.
16:      end if
17:   end for
18:   if (H is better than worst Harmony in HM) then
19:      Update HM by replacing Worst harmony by H.
20:   end if
21: end while
22: print Best Harmony as obtained solution.
```

While generating the ith component of the new harmony referred as H_i either the ith component of some existing harmony from HM is assigned to it with probability HMCR (step 10) or a randomly generated value is assigned to it with probability 1-HMCR (step 15). In step no. 12, H_i is slightly modified with probability PAR.

In order to enhance the performance of the standard HS algorithm several variants of HS algorithm has been proposed in literature. A detailed survey on variants of HS can be found in [9].

Mahdavi et al. [10] proposed dynamic adaptation of both pitch adjustment rate (PAR) and bandwidth (BW), the algorithm became known as Improved Harmony Search (IHS). Inspired from Particle Swarm Optimization paradigm Omran and Mahdavi introduced another important modification to Harmony Search algorithm referred as Global-best Harmony Search (GHS) [11] algorithm. A new adaptation for HS was proposed in [12] by changing HMCR and PAR dynamically during the execution of Harmony Search and the algorithm became known as Adaptive Harmony Search. A Self adaptive Harmony Search was proposed by Wang et al. in [13]. Cheng et al. [14] developed Modified Harmony Search (MHS), which is based on the idea of selecting better harmony with higher probability. Explorative Harmony Search (EHS) [15] algorithm proposed by Das et al. eliminates the limitation of tuning the BW parameter by making it proportional to the population variance in HM. Another variant of HS based on the idea of increasing the PAR rather than decreasing it so as to favor exploration in the beginning of the algorithm has been proposed in [16]. Pan et.al modified the GHS algorithm to create Self-adaptive Global Best Harmony Search (SAGHS) algorithm [17] and Intelligent Tuned Harmony Search (ITHS) algorithm was proposed in [18]. The Improved Global-best harmony search (IGHS) introduced in [19] has been created by modifying pitch adjustment step of the standard HS.

Harmony Search has been successfully hybridized with other metaheuristic algorithms. The goal of hybridization is to improve the capabilities of the optimization algorithms to solve complex problems [20] by inheriting the good aspects of both algorithms which are being hybridized. The inception of the ability of HS algorithm to be integrated with other metaheuristic return to the relative ease and flexible structure of HS.

Geem [21] hybridized the Harmony search and Particle swarm optimization algorithm to create an efficient algorithm for solving water network design problem. In [22] three metaheuristic algorithms namely Simulated annealing (SA), GA and Artificial Immune System (AIS) were used to enhance the quality of the harmonies stored in HM and thus increase the convergence speed and at the same time preventing the HS from getting stuck in the local optimal. To enhance exploitation capabilities of HS it has been hybridized with Sequential Quadratic Programming (SQP) that acts as a local search operator in [23]. SQP is applied with a probability of P_c to improve the quality of the new improvised vector. Also as a final step once the HS meets the stopping criteria SQP is applied to the best harmony so that its quality can further be improved.

2.2. Simulated annealing

Simulated Annealing (SA) is an iterative meta-heuristic for solving nonlinear and non-convex optimization problems. It was introduced in [24] based on the Metropolis algorithm [25] and different variations were introduced later [26,27]. SA

has been extensively used in problems such as Traveling Salesman Problem, packing problem, supply chain management, vehicle routing, machine scheduling and timetabling.

A typical SA algorithm for a minimization problem is given as Algorithm 2. The algorithm starts from an initial randomly

Algorithm 2 Simulated Annealing (SA).

1: Let $f()$ be the given Objective function.
2: Initialize S, T_0 *and* L
3: $T \leftarrow T_0$
4: $Cost\_Current = f(S)$
5: **while** (Stopping Criteria Not Reached) **do**
6: $S' \leftarrow Neighbor(S)$
7: $Cost\_New = f(S')$
8: $\triangle Cost = Cost\_New - Cost\_Current$
9: **if** $(\triangle Cost < 0)$ **then**
10: $S = S'$ *and* $Cost\_Current = Cost\_New$
11: **else**
12: **if** $(rand(0,1) < e^{\frac{-\triangle Cost}{T}})$ **then**
13: $S \leftarrow S'$ *and* $Cost\_Current = Cost\_New$
14: **end if**
15: **end if**
16: After every L iterations Set $T = \alpha T$
17: **end while**

selected solution S (step 2) and temperature parameter denoted as T is set to its initial value T_0 (step 3). The cost of the initial solution S is calculated (step 4) and the following steps are repeated until the stopping criterion is satisfied. A neighbor solution S' of S is generated and its fitness is calculated (step 7). The Neighbor() function is used to generate a new solution of S by changing the value of one or more components. A better solution is always accepted and an inferior solution is also accepted with a probability determined by its fitness and current temperature T (step 12). The temperature is initially kept high so as to favor inferior moves and is gradually decreased by a factor α (step 16) to lower the probability of accepting inferior moves.

3. Proposed Hybrid Harmony Search and Simulated Annealing (HS-SA) algorithm

Harmony Search is a powerful metaheuristic algorithm with excellent exploitation capabilities, however it has a very serious limitation of getting stuck in local optimal usually referred as premature convergence if the initially selected harmonies are in the vicinity of local optimal. In order to remove this limitation HS algorithm is hybridized by proposing HS-SA algorithm so as to increase exploration particularly in the beginning of execution to escape local optima.

The success of a meta heuristic algorithm depends on the balance between exploration and exploitation. An ideal meta heuristic algorithm must have greater exploration capabilities in the earlier generations and enhanced exploitation capabilities towards the later generations. In HS-SA an attempt has been made to achieve this goal. Taking the inspiration from SA, the HS-SA algorithm accepts even inferior harmonies with probability determined by a parameter called Temperature (T). The Temperature parameter is initially kept high to favor inferior moves and hence increase capability of escaping local optima and is linearly decreased to gradually shift focus to exploitation of good harmonies. The pseudo code of HS-SA for a minimization problem is shown as Algorithm 3. The HS-SA is same as standard HS with the exception that even inferior harmonies are accepted as done in SA. Step 2 initializes the algorithmic parameters and it can be observed that three extra parameters of Simulated Annealing (T_0, α, L) are also required in HS-SA. Step number 7–16 generate a new harmony as in standard HS. In step 18 not only the superior harmonies (compared to worst harmony) are always accepted but the inferior harmonies are accepted with probability determined by the fitness of the new harmony and the current temperature. The temperature parameter is gradually decreased in step 21 so as to reduce the probability of accepting inferior harmonies and hence favour exploitation of good harmonies. In Algorithm 3 rand is a uniform random number generator generating random numbers between 0 and 1.

4. Numerical experiments on CEC 2014 benchmark suite

In this section, the performance of the proposed HS-SA algorithm is evaluated on IEEE CEC 2014 Benchmark functions [3]. The HS-SA algorithm has been compared with standard HS and Simulated Annealing. The experimentation has been carried out on all the IEEE CEC 2014 Benchmark functions using 10 and 30 dimensions. As per the instructions of test suite every problem is tested with 51 independent runs.

The parameter setting adopted for standard HS has been taken from [19] and is shown in Table 1 along with the parameter setting of SA and HS-SA algorithm. Obtaining an optimal parameter setting for a metaheuristic algorithm is a hyper

Algorithm 3 Hybrid Harmony Search Simulated Annealing (HS-SA).

Let $f()$ be the given Objective function
Initialize Parameters HMCR, PAR, BW, HMS, T_0, α, L
Initialize Harmony Memory (HM)
Set $T = T_0, \quad L = 3 \times D$
while (Stopping Criteria Not Reached) **do**
  Find current **Worst** harmony and **Best** harmony in HM
  for $i = 1$ to D **do**
    if (rand $\leq$ HMCR) **then**
      $H_i = HM_i^j$ where j=rand_int(1,HMS)
      if (rand $\leq$ PAR) **then**
        $H_i = H_i \pm rand \times BW$
      end if
    else
      Generate H_i randomly within the allowed bounds
    end if
  end for
  $\triangle Cost = f(H) - f(Worst)$
  if ($\triangle Cost < 0 \quad OR \quad rand < e^{\frac{-\triangle Cost}{T}}$) **then**
    Update HM by replacing **Worst** harmony by H
  end if
  After every L iterations Set $T = \alpha T$
end while
print **Best** Harmony as obtained solution

Table 1
Parameter setting of algorithms used in this study.

Algorithm	HMS	HMCR	PAR	BW	T_0	α	L
HS [1]	5	0.9	0.3	0.001	–	–	–
SA [24]	–	–	–	–	*	0.99	3 × D
HS-SA	5	0.9	0.3	0.001	*	0.99	3 × D

* indicates that T_0 has been calculated as explained in section titled Numerical Experiments and - indicated the parameter is not applicable.

optimization problem, however the parameter setting shown in Table 1 was found out to be appropriate for most, if not all the problem instances. Parameter T_0 for every function has been calculated by using Eq. (2). Where Worst and Best in Eq. (2) respectively refer to the worst and best function value obtained after evaluating the function for ten random vectors, β is the initial acceptance rate and is taken as 0.95.

$$T_0 = \frac{Worst - Best}{log(\beta)} \tag{2}$$

All the algorithms have been implemented in Dev C++ 5.0 and the experimentation has been carried out on a laptop with Windows 10 operating system, intel core i3 processor and 4GB of RAM.

4.1. IEEE CEC 2014 Benchmark suite

The IEEE CEC 2014 Benchmark suite is a collection of 30 unconstrained continuous optimization problems with varying difficulty levels. The functions 1 through 3 are unimodal, 4 through 16 are simple multimodal functions, 17 through 22 are hybrid functions and 23 through 30 are composition functions. The search range for each function is $[-100, 100]^D$ where D is the dimension of the problem.

Each problem is tested with 51 independent runs and the error values obtained in 51 runs are sorted from the smallest (best) to the largest (worst) and then best, worst, mean, median and standard variance of error values for each function is presented. The stopping criteria is performing MaxFES function evaluations or the error value is smaller than 10^{-8}. MaxFES is the maximum number of function evaluations allowed and is equal to $10^4 \times D$.

4.2. Analysis of results

The results reported in this paper are in the format as specified and required in IEEE CEC 2014 benchmark suit. Tables 2 and 3 show the results obtained by the three algorithms on 10 dimensional problems whereas Tables 4 and 5 show the results on 30 dimensional problems. The recorded results are the minimum, maximum, mean, median, and standard deviation

Table 2
Minimum, Maximum, Mean, Standard Deviation and Median of error value obtained by HS, SA and HS-SA on Function number 1 through 16 (Dimension 10) IEEE CEC 2014 benchmark problems. The better results are highlighted by bold.

Function	Algorithm	Best	Worst	Mean	Std Dev	Median
1	HS	2024.615373	1264336.972	101529.6576	193869.1394	46799.8972
	SA	30589.53061	592733.5247	255511.0682	154226.5451	237059.6142
	HS-SA	**706.645778**	**533634.1253**	**82505.10611**	**106852.3056**	**44781.00061**
2	HS	**14.409848**	11613.01593	5000.373956	3702.779672	4729.521976
	SA	25.205122	**10557.49275**	**3021.497966**	**2869.916983**	**2467.052241**
	HS-SA	78.651056	11108.24605	4124.633742	3222.309443	3348.860726
3	HS	99.495782	**17051.67611**	**5956.628969**	**4861.313866**	5029.526225
	SA	2666.311377	28400.22796	10766.82428	5603.77009	9449.395612
	HS-SA	**2.043561**	28059.60112	6556.588379	6387.374065	**4597.865761**
4	HS	0.001596	66.233851	12.254274333	10.63294081	4.060043
	SA	**0.000089**	**34.780283**	16.88307506	15.64891511	4.335434
	HS-SA	0.001546	70.04005	**7.27661416**	**9.2563402**	**0.504308**
5	HS	**0.000825**	**20.000001**	**19.25513465**	3.692047086	**19.999999**
	SA	19.994767	20.000046	19.99986643	**0.000735542**	20.000006
	HS-SA	1.646242	20.004775	19.32544773	3.339333837	20
6	HS	0.649689	6.33008	3.371964627	1.373186248	**3.234556**
	SA	8.287427	19.517817	12.70115988	2.413300672	12.628526
	HS-SA	**0.592986**	**5.3489**	**3.244567569**	**1.186168858**	3.730403
7	HS	0.039363	0.924989	0.226605961	0.168619442	0.192106
	SA	0.076241	3.391778	0.718100431	0.775297207	0.415714
	HS-SA	**0.017241**	**0.605639**	**0.207120216**	**0.14555276**	**0.153249**
8	HS	**0**	**0**	**0**	**0**	**0**
	SA	67.656853	299.477398	141.1911161	44.67702091	127.353671
	HS-SA	**0**	**0**	**0**	**0**	**0**
9	HS	2.984877	19.899161	10.76895931	4.229083295	10.944545
	SA	76.611194	277.588331	162.4190345	42.47033666	159.255593
	HS-SA	**2.979836**	**18.884018**	**10.30074475**	**3.839946226**	**9.949586**
10	HS	**0**	0.124913	**0.02082551**	**0.034247554**	**0.000002**
	SA	603.981304	2406.552283	1422.867199	406.959849	1371.94992
	HS-SA	**0**	**0.124909**	0.025720176	0.037464501	**0.000002**
11	HS	11.954435	649.560499	252.0211735	153.0642669	267.318394
	SA	497.699637	2353.318469	1473.48928	420.6869217	1480.639257
	HS-SA	**7.017285**	**620.792715**	**248.9707464**	**147.8597231**	**240.416618**
12	HS	0.01975	0.413027	0.126400078	0.077628815	0.113384
	SA	**0.001071**	**0.249401**	**0.054336235**	**0.057476106**	**0.034222**
	HS-SA	0.005709	0.346812	0.120865824	0.077689221	0.113097
13	HS	0.112451	0.969596	**0.370780078**	**0.152623758**	**0.360704**
	SA	0.176336	1.521441	0.740451725	0.291024172	0.730998
	HS-SA	**0.109073**	**0.690244**	0.376753059	0.154537144	0.363048
14	HS	**0.086034**	**0.955533**	0.362337059	0.235769455	**0.267156**
	SA	0.167679	1.665544	0.524960137	0.319697016	0.380781
	HS-SA	0.104591	1.033047	**0.337595863**	**0.232820522**	0.267941
15	HS	0.680723	8.356495	2.116615	1.380586478	**1.802614**
	SA	0.731106	**5.024347**	2.299249706	**1.090216117**	2.098397
	HS-SA	**0.605344**	7.702636	**2.040062745**	1.198450325	1.992706
16	HS	1.626887	**3.189286**	2.471638098	0.364040843	2.495824
	SA	3.601968	4.967019	4.541465588	**0.309790258**	4.56857
	HS-SA	**1.540465**	3.36388	**2.455255059**	0.404660696	**2.491922**

of the error value obtained as specified in IEEE 2014 Benchmark suite. The error value is the absolute difference between obtained objective function value by the algorithm and the known function value. In all the four Tables 2–5 the best result obtained are highlighted by bold font.

This paragraph analyze the results of 10 dimensional problems. For unimodal functions (1 through 3) all the three algorithms produced best mean results for one instance whereas HS-SA produced best results for two problems and HS produced best result for one instance. In case of 13 simple multimodal functions (4 through 16) HS-SA, HS and SA reported the best results in 9, 4 and 2 instances respectively, and reported the best mean results in 9, 4 and 1 instances respectively. Thus HS-SA is the winner both in terms of best and mean results for simple multimodal functions. In case of hybrid multimodal functions (17 through 22) HS-SA produced the best result in 5 instances and best mean result in 4 instances, HS produced the best result and best mean result in 1 instance and SA produced the best mean result in 1 instance. In case of composition multimodal functions (23 through 30) HS-SA, HS and SA produced the best results in 6, 1, 1 instances respectively and the best mean results in 5, 2, 1 instances respectively. Thus HS-SA impressively outperformed HS and SA algorithms both in terms of Best and Mean results on multimodal functions.

Table 3
Minimum, Maximum, Mean, Standard Deviation and Median of error value obtained by HS, SA and HS-SA on Function number 17 through 30 (Dimension 10) IEEE CEC 2014 benchmark problems. The better results are highlighted by bold.

Function	Algorithm	Best	Worst	Mean	Std Dev	Median
17	HS	1112.154516	1923953.535	278226.6652	441872.6584	81604.39526
	SA	9914.145009	**1338471.724**	514202.8929	416653.332	405312.3023
	HS-SA	**49.224508**	1417804.966	**242703.2866**	**359004.7428**	**74670.60139**
18	HS	**6.538502**	36495.97201	10600.21042	11550.28915	7037.217453
	SA	74.068969	36038.22559	10365.40483	9826.277288	7680.457396
	HS-SA	991.709978	**13925.77975**	**4787.499182**	**1441.927698**	**4622.295028**
19	HS	0.184631	1.98528	1.056482725	0.475149066	1.082683
	SA	1.633301	8.071913	4.307609314	1.378145771	4.473488
	HS-SA	**0.175909**	**1.974482**	**0.979982843**	**0.459262927**	**1.075747**
20	HS	11.853075	28921.40768	7622.383374	8132.303804	3998.495253
	SA	31.970007	**16876.9091**	**2197.306081**	**3146.277633**	**1047.754067**
	HS-SA	**11.565039**	30442.71659	6343.198212	7154.610183	3545.421293
21	HS	19.826384	**21349.45563**	5143.984255	**6115.284453**	2074.558552
	SA	1118.400124	196203.5134	57738.07133	52894.75528	46742.88126
	HS-SA	**0.730229**	83728.23117	**4633.704853**	12619.874	**983.655747**
22	HS	0.025513	**118.740482**	**6.55471398**	**17.18629139**	**2.239985**
	SA	20.82623	838.859735	449.5447088	202.8219162	454.400515
	HS-SA	**0.015668**	145.214208	7.767828275	20.61423183	2.757135
23	HS	329.457475	**329.457475**	329.457475	**0**	**329.457475**
	SA	**100.018326**	329.457484	**317.9218396**	47.53783363	329.457477
	HS-SA	329.457475	**329.457475**	329.457475	**0**	**329.457475**
24	HS	111.256008	148.215613	127.6186052	8.366729056	127.781746
	SA	178.08647	998.17782	285.0496245	141.6291828	245.30588
	HS-SA	**110.194039**	**145.086124**	**127.0232095**	**7.401734942**	**125.431697**
25	HS	124.378738	204.302622	182.5107444	29.73186373	201.63295
	SA	200.510831	219.830555	203.1991974	2.695844722	202.555764
	HS-SA	**122.602773**	**203.13363**	**177.9977278**	**28.83152328**	**201.549579**
26	HS	100.113825	**100.611758**	100.3390896	**0.124415488**	**100.333253**
	SA	100.132608	479.192931	197.6457507	110.2554355	200.015451
	HS-SA	**100.086504**	100.808842	**100.3145724**	0.138725758	100.350447
27	HS	**3.523765**	**493.817917**	**369.7159748**	97.08113655	**391.636258**
	SA	9.546533	776.610853	519.4394248	148.5510233	506.171271
	HS-SA	5.267504	498.974972	382.2942878	**83.05671614**	400.826206
28	HS	371.29863	**656.42116**	495.0411432	**69.67835223**	504.107264
	SA	579.136317	6351.830493	2927.983495	1669.664806	2527.40591
	HS-SA	**358.923943**	674.923835	**485.1494779**	77.72388726	**488.76135**
29	HS	232.452395	**774088.2979**	**45728.29453**	**180938.6683**	**467.830769**
	SA	293.737306	4998659.881	1312746.54	1520638.564	1505.478623
	HS-SA	**217.953004**	1724538.229	101986.4773	405638.0331	527.905095
30	HS	515.472685	1323.271353	811.610144	216.4346733	764.910337
	SA	594.845857	2015.114766	1253.290361	333.2477215	1264.564243
	HS-SA	**488.410632**	**1313.004365**	**785.2163952**	**215.0907877**	**741.567823**

This paragraph analysis the results reported in Tables 4 and 5 for 30 dimensional problems. For unimodal functions SA reports the best results in two instances and HS obtained the best result in one instance, SA and HS reported the best mean results in two and one instance respectively. In case of 13 simple multimodal functions HS-SA, HS and SA reported the best results in 7, 1 and 5 instances respectively, and reported the best mean results in 7, 2 and 4 instances respectively. In case of six instances of hybrid multimodal functions HS-SA produced the best result in 5 instances and best mean result in 3 instances, SA produced the best result in 1 instance and best mean result in 3 instances whereas HS failed to produce the best or best mean result. In case of composition multimodal functions HS-SA and SA produced the best mean results in 6 and 2 instances respectively whereas HS completely failed to produce the best mean result in any instance.

From the above discussion it can be concluded that HS-SA algorithm outperforms its competitors on multimodal functions, the superior performance of HS-SA is particularly evident on composition functions, which are the shifted, rotated, expanded, and combined variants of the classical functions and hence offer the greatest complexity.

4.3. t-Test Analysis

A pair wise two tail t-Test at 5% level of significance is used to statistically compare the performance of the three competing algorithms on 30 dimensional functions and the t-Test results are shown as Table 6. The sampling data used for applying t-Test has been obtained by performing 51 independent runs of each algorithm.

The t-Test value for a function has been set as A if HS-SA performs significantly better than other two algorithms in the same way it is set as B and C for HS and SA respectively. The multiple entries for some functions in Table 6 indicate

Table 4
Minimum, Maximum, Mean, Standard Deviation and Median of error value obtained by HS, SA and HS-SA on Function number 1 through 16 (Dimension 30) IEEE CEC 2014 benchmark problems. The better results are highlighted by bold.

Function	Algorithm	Best	Worst	Mean	Std Dev	Median
1	HS	1137553.817	50947967.78	11900193.63	10530736.33	8683954.305
	SA	**112623.0222**	**1950985.289**	**654970.8475**	**418183.1181**	**565898.3785**
	HS-SA	1583840.798	65767900.31	11566439.89	10957063.54	8274451.372
2	HS	117.276847	34622.44263	13078.29611	11045.32756	12302.3083
	SA	**16.227512**	**29534.05532**	**6804.175364**	**7503.948376**	**4320.025786**
	HS-SA	99.629031	33361.84522	13830.86645	11345.06017	12521.50254
3	HS	**27.82727**	38095.05077	7161.160686	8201.461481	4592.582959
	SA	7858.241951	44693.07392	27893.19536	9270.138384	27114.91921
	HS-SA	41.077712	**28496.82519**	**6308.100191**	**6063.310659**	**4266.338042**
4	HS	1.51277	321.371175	126.6562774	48.50248267	140.762176
	SA	**0.066404**	**69.998227**	**2.384142745**	**10.17011871**	**0.444018**
	HS-SA	4.295379	205.57011	111.2191262	40.06056413	117.330991
5	HS	20.000117	20.005036	20.00031196	0.000729137	20.000181
	SA	**19.999992**	20.007439	20.00040724	0.001484153	**20.000023**
	HS-SA	20.000129	**20.002087**	**20.00024565**	**0.000310287**	20.000176
6	HS	10.778884	19.705962	14.472374	2.197131778	14.681947
	SA	26.36189	45.259821	35.21119965	4.048046126	35.751437
	HS-SA	**10.294357**	**18.995425**	**13.06356149**	**2.121506054**	**13.208033**
7	HS	0.000162	0.235892	0.016120235	0.034934665	**0.000396**
	SA	**0.000083**	0.088494	0.022008373	0.019566975	0.015028
	HS-SA	0.000185	**0.06161**	**0.015190784**	**0.01634209**	0.015016
8	HS	0.000025	0.000069	4.23922E-05	8.15554E-06	0.000042
	SA	242.768536	587.016521	398.925054	80.11096675	395.989017
	HS-SA	**0.000024**	**0.000057**	**4.10392E-05**	**8.13363E-06**	**0.000041**
9	HS	37.80848	121.384474	69.15421551	18.46879347	69.64705
	SA	279.498235	898.420187	562.0957507	124.7102428	567.114319
	HS-SA	**36.808507**	**101.48525**	**67.11695837**	**15.19582083**	**68.63674**
10	HS	0.086872	**0.292876**	**0.197545529**	**0.043335239**	**0.189705**
	SA	3410.840097	6032.034321	4614.230302	632.3350939	4579.579595
	HS-SA	**0.085307**	0.293873	0.201482922	0.046565775	0.190465
11	HS	1032.789971	**2840.775628**	**1983.508689**	**400.2670571**	**1962.115514**
	SA	3277.702168	6076.857866	4579.110398	540.2888439	4550.591288
	HS-SA	**969.811972**	2855.690353	1988.987007	434.3879434	2060.895611
12	HS	0.086941	0.228864	0.154727216	0.037850082	0.150219
	SA	**0.00841**	**0.064771**	**0.024637235**	**0.012607429**	**0.020439**
	HS-SA	0.078137	0.307013	0.163992137	0.050046608	0.150544
13	HS	**0.27309**	0.939628	0.573187078	0.128081445	0.549634
	SA	0.363635	**0.759847**	0.542204196	0.106545743	0.53973
	HS-SA	0.315457	0.761266	**0.524117157**	**0.104315528**	**0.504629**
14	HS	0.191445	1.005696	0.392323	0.174392951	0.335945
	SA	0.209117	**0.397182**	**0.296813882**	**0.046953556**	**0.285568**
	HS-SA	**0.168112**	1.043144	0.414546549	0.228596981	0.336389
15	HS	5.469191	38.544269	14.50728757	7.192048877	11.526468
	SA	**3.948945**	**15.875609**	**9.195183784**	**2.605588358**	**8.451837**
	HS-SA	5.615548	74.032661	16.39189741	11.68770848	13.225481
16	HS	7.571243	11.214736	9.477624549	0.790252484	9.521612
	SA	12.596621	14.848375	14.19227733	**0.41428379**	14.272418
	HS-SA	**7.53601**	**10.862589**	**9.263387588**	0.782855716	**9.321862**

more than one algorithm have performed better than others however there is no significant difference between the listed algorithms e.g. for function 7 the A, B t-Test value indicates HS-SA and HS performed significantly better than SA, how ever there is no significant difference between HS-SA and HS. The - for function 5 and 8 indicate that there is no statistically significant difference in the performance of all the algorithms on these functions.

For unimodal functions SA is the winner as it has found significantly better results in two instances out of three, however for multimodal functions HS-SA is the winner as it has found significantly better results in 16 instances, followed by SA and HS that have managed to find significantly better results in 8 and 4 instances respectively.

The performance of proposed HS-SA algorithm is particularly prominent on Composition functions where it has significantly outperformed other competing algorithms in 6 out of 8 instances followed by SA that managed to outperform in just one instances. The composition functions are generally very challenging test beds for metaheuristic algorithms as exploration and exploitation can be simultaneously benchmarked by the composite functions. Moreover, the local optima avoidance of an algorithm can be examined due to the massive number of local optima in such test functions.

Table 5
Minimum, Maximum, Mean, Standard Deviation and Median of error value obtained by HS, SA and HS-SA on Function number 17 through 30 (Dimension 30) IEEE CEC 2014 benchmark problems. The better results are highlighted by bold.

Function	Algorithm	Best	Worst	Mean	Std Dev	Median
17	HS	167671.0695	4993363.463	2010103.437	1307086.136	1604010.008
	SA	**50462.0761**	**947572.0297**	**372702.1986**	**196332.2283**	**327751.0743**
	HS-SA	113300.8338	5606170.944	2087899.974	1310003.563	1931384.506
18	HS	49.987352	25577.35678	4818.267709	4943.224214	3136.943628
	SA	136.941019	**21540.48646**	**3288.736495**	**3977.140374**	**1861.863231**
	HS-SA	**42.053633**	24809.34018	6157.931933	6221.394287	4735.435313
19	HS	4.796791	124.932091	19.75680531	26.19771326	9.425505
	SA	9.032103	173.727163	21.44655616	**21.83374557**	12.026571
	HS-SA	**4.467052**	**114.584636**	**18.87371978**	24.60404528	**9.415907**
20	HS	1041.305793	30068.36986	6803.7555	5576.33862	**5132.107334**
	SA	5643.337282	100632.6617	40537.23538	20853.13142	40253.19965
	HS-SA	**918.059592**	**22559.08898**	**6769.699014**	**5085.369665**	5574.27143
21	HS	15884.36921	1659230.934	536300.4922	361098.8482	489114.3203
	SA	20709.63192	**1030796.032**	**451600.4115**	239456.7995	452941.1874
	HS-SA	**15420.2457**	1604103.404	526989.0989	**359489.0477**	**447868.313**
22	HS	157.031925	**856.366871**	507.2656545	178.3224078	484.434498
	SA	295.348217	1580.491214	1100.357173	268.8626019	1105.42779
	HS-SA	**53.92134**	944.007353	**487.9589202**	**172.8226814**	**482.691429**
23	HS	315.246625	317.148638	315.6074174	0.388650579	315.514061
	SA	**315.244108**	**315.244988**	**315.2442739**	**0.000179826**	**315.24421**
	HS-SA	315.244534	317.666757	315.7293326	0.573794473	315.454036
24	HS	227.735194	248.600915	233.6608904	5.215800199	231.701834
	SA	248.260311	821.81851	355.197295	120.7704009	293.13022
	HS-SA	**227.687986**	**238.086362**	**230.6527992**	**5.171124998**	**230.436568**
25	HS	205.107535	213.439246	207.8779291	1.924703278	207.29941
	SA	203.216176	253.404059	220.8529842	13.04710784	220.644622
	HS-SA	**202.182801**	**211.21568**	**205.1772248**	**1.511765482**	**206.097493**
26	HS	100.241332	340.065953	140.5971678	55.33574375	100.714249
	SA	100.301174	1047.860919	163.9885292	134.3079487	100.696608
	HS-SA	**100.177063**	**201.335676**	**137.8569406**	**48.38274733**	**100.659406**
27	HS	402.333727	903.157941	673.3748665	176.5585803	730.452986
	SA	**401.371978**	1515.346606	1017.461581	467.3351416	1224.120466
	HS-SA	402.873404	**902.909693**	**669.416932**	**162.9863584**	**702.081548**
28	HS	831.007532	2146.9474	1076.497372	261.2520787	1012.061673
	SA	2634.55868	11299.59152	6382.22538	1865.979286	6366.225811
	HS-SA	**760.224649**	**1416.928016**	**1028.368876**	**121.492247**	**1000.801337**
29	HS	616.225827	2719.629464	1459.54097	433.6824481	1364.202599
	SA	806.814774	18308333.98	1186166.307	4159086.233	1440.847711
	HS-SA	**608.601545**	**2443.1858**	**1398.190023**	**427.343221**	**1239.779371**
30	HS	1759.397611	9386.965218	4344.545107	1492.198205	4241.530517
	SA	1738.250693	**8058.768301**	**3078.985568**	**1047.48648**	**2916.689532**
	HS-SA	**1600.269512**	12392.77651	4630.836653	2318.985736	3807.421406

Table 6
Paired two tail t-Test results at 5% significance level and 50 degree of freedom for 30 dimensional problems.

Function	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
t-Test	C	C	A	C	–	A	A,B	A,B	A	A,B	B	C	A	C	C
Function	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
t-Test	A	C	C	A	A	C	A	–	A	A	A	A	A	A	C

4.4. Algorithm complexity

The time complexity of algorithms is computed as per the requirements laid down in IEEE CEC 2014 Benchmark suite. The complexity is represented in terms of three parameters T0, T1 and T2. T0 is the time complexity of a specified test program provided in IEEE CEC 2014 Benchmark suite reproduced as Algorithm 4. T1 is the computing time for 2×10^5 function evaluations of function 18 with dimension D and T2 is the computing time taken by the algorithm when the stopping criteria is 2×10^5 function evaluations of function 18 with dimension D. The time complexity for 10 and 30 dimensional problem is shown as Table 7.

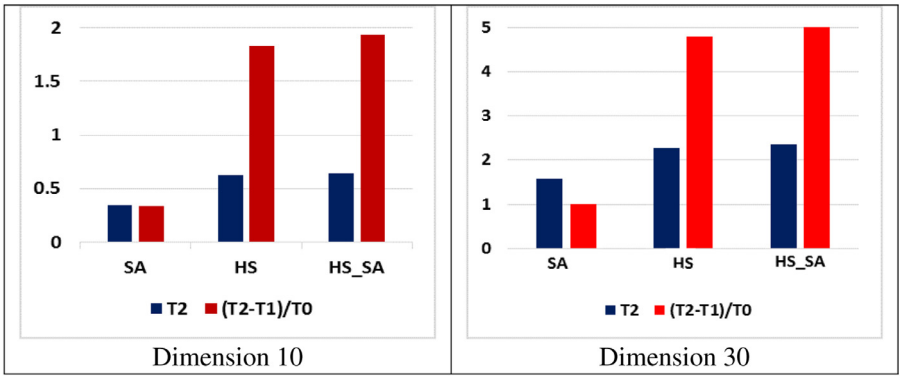
It is evident from Table 7 the time complexity of the algorithms shows the following order for both 10 and 30 dimensional problems.

$$SA < HS < HS - SA$$

Algorithm 4 TEST PROGRAM (T0).

```
for i = 1 to 1000000 do
    x=0.5+(double) i
    x=x + x; x=x/2; x=x*x; x=sqrt(x); x=log(x); x=exp(x); x=x/(x+2);
end for
```

Table 7
Complexity of Algorithms in seconds for 10 and 30 Dimensional problem.



HS-SA has to re adjust the temperature parameter during execution resulting in slightly higher time complexity compared to HS.

5. Comparison with other meta heuristic algorithms

In this section proposed HS-SA algorithm is compared with several state-of-the-art meta heuristic algorithms including Adaptive particle swarm optimization (APSO) [28], Social Spider Optimization algorithm (SSO) [29], Differential Evolution (DE) [30], Differential Evolution with a successful parent selecting framework (DE-SPS) [30], Symbiotic organisms search (SOS) [31], Gravitational Search Algorithm (GSA) [32], Cuckoo Search Algorithm (CS) [33], Evolution Strategy with Covariance Matrix Adaptation (CMA-ES) [34], Standard Cuckoo Search (CS2) [35], MOCS [36] and CS-VSF [37] on 30 dimensional IEEE CEC 2014 benchmark problems and the results are reported in Tables 8–10. The column W in all the tables represents the t-Test results conducted at 5% level of significance. The cases are marked with +, −, and = when the performance of HS-SA is significantly better than, worse than, or similar to the competing algorithm. In all the three Tables 8–10 the mean results of all algorithm are in bold face if the corresponding algorithm performs better than HS-SA algorithm.

On all the three instances of unimodal functions, all the thirteen instances of simple multimodal functions and all the six instances of hybrid multimodal functions the proposed HS-SA algorithm performed significantly better than Adaptive particle swarm optimization (APSO). On seven instances of composition functions APSO outperformed HS-SA whereas latter outperformed the former on just one instance. Thus HS-SA significantly outperformed APSO on all categories of functions except composition functions.

The proposed HS-SA algorithm significantly outperformed Social Spider Optimization algorithm (SSO) [29] on all the thirty instances.

HS-SA significantly outperformed Differential Evolution (DE) [30] on two instances of unimodal functions whereas latter outperformed the former on just one instance. HS-SA significantly outperformed DE on eleven instances of simple multimodal functions whereas DE is the winner in the remaining two instances. In case of hybrid multimodal functions HS-SA is the winner on two instances whereas DE performed better in the remaining four instances. In case of composition functions HS-SA is the winner on three instances whereas DE is the winner in four instances and the difference in performance is not statistically significant on the remaining one instance.

HS-SA significantly outperformed DE-SPS [30] on two instances of unimodal functions whereas latter outperformed the former on just one instance. HS-SA significantly outperformed DE-SPS on eight instances of simple multimodal functions whereas DE-SPS is the winner in the remaining five instances. In case of hybrid multimodal functions DE-SPS outperformed HS-SA on all the six instances. In case of composition functions HS-SA is the winner on two instances whereas DE-SPS is the winner in five instances and the differen in performance is not statistically significant on the remaining one instance.

The proposed HS-SA significantly outperformed Symbiotic organisms search (SOS) [31] on all instances of unimodal, simple multimodal and hybrid multimodal functions. In case of composition functions HS-SA outperformed SOS on six instances whereas latter outperformed former in the remaining two instances.

Table 8
Mean and Standard Deviation of error value obtained by HS-SA, APSO, SSO, DE, and DE-SPS on all functions of IEEE CEC 2014 (Dimension 30) benchmark problems.

	HS-SA		APSO			SSO			DE			DE-SPS		
Function	Mean	Std Dev	Mean	Std Dev	w	Mean	Std Dev	w	Mean	Std Dev	w	Mean	Std Dev	w
1	1.1566E+07	1.0957E+07	2.6900E+09	3.2800E+08	+	3.8700E+09	1.2100E+09	+	8.8069E+07	1.8777E+07	+	3.2830E+07	9.5544E+06	+
2	1.3831E+04	1.1345E+04	1.0200E+11	2.2900E+09	+	1.4200E+11	2.2100E+10	+	**1.7236E+03**	4.9340E+02	−	**1.5489E+03**	4.1780E+02	−
3	6.3081E+03	6.0633E+03	1.1900E+06	1.2500E+06	+	1.1200E+07	1.5700E+07	+	2.5487E+01	5.0444E+00	+	1.9002E+01	2.9699E+00	+
4	1.1122E+02	4.0061E+01	2.4900E+04	1.5400E+03	+	4.4000E+04	1.0300E+04	+	1.2268E+02	5.7729E+00	+	**8.9727E+01**	6.7012E+00	−
5	2.0000E+01	3.1029E-04	2.1300E+01	5.6100E-02	+	2.1400E+01	7.9800E-02	+	2.0897E+01	5.7695E-02	+	2.0888E+01	5.5126E-02	+
6	1.3064E+01	2.1215E+00	4.8000E+01	1.7900E+00	+	4.9200E+01	2.7700E+00	+	3.0312E+01	1.0790E+00	+	**4.5857E+00**	1.3963E+00	−
7	1.5191E-02	1.6342E-02	1.0600E+03	3.8500E+01	+	1.2900E+03	1.8900E+02	+	4.3384E-02	7.9261E-02	+	**1.0475E-03**	7.4062E-04	−
8	4.1039E-05	8.1336E-06	5.0300E+02	3.0200E+01	+	5.2200E+02	3.2100E+01	+	1.1981E+02	6.5322E+00	+	8.6523E+01	1.1314E+01	+
9	6.7117E+01	1.5196E+01	4.7800E+02	6.3000E+00	+	6.4100E+02	3.8400E+01	+	1.9501E+02	8.9552E+00	+	1.7766E+02	1.1591E+01	+
10	2.0148E-01	4.6566E-02	9.3000E+03	5.6800E+02	+	9.4200E+03	4.2700E+02	+	3.9102E+03	2.3839E+02	+	2.4244E+03	2.6569E+02	+
11	1.9890E+03	4.3439E+02	9.2400E+03	4.8600E+02	+	9.6000E+03	5.1100E+02	+	6.5465E+03	2.4765E+02	+	6.1906E+03	2.7124E+02	+
12	1.6399E-01	5.0047E-02	5.9100E+00	1.3200E+00	+	6.7400E+00	1.4200E+00	+	2.0819E+00	2.0466E-01	+	9.3610E-01	3.5208E-01	+
13	5.2412E-01	1.0432E-01	1.0300E+01	7.5300E-01	+	1.1100E+01	1.4400E+00	+	**4.8848E-01**	4.5645E-02	−	**4.1517E-01**	5.0633E-02	−
14	4.1455E-01	2.2860E-01	3.9500E+02	2.2200E+01	+	4.3800E+02	6.6200E+01	+	**2.9400E-01**	3.7750E-02	−	**2.6819E-01**	2.9785E-02	−
15	1.6392E+01	1.1688E+01	1.0500E+06	0.0000E+00	+	3.7400E+07	2.2600E+07	+	1.8975E+01	1.1252E+00	+	1.7327E+01	9.0671E-01	+
16	9.2634E+00	7.8286E-01	1.4200E+01	2.3700E-01	+	1.4300E+01	2.1300E-01	+	1.2474E+01	2.2289E-01	+	1.1956E+01	2.5889E-01	+
17	2.0879E+06	1.3100E+06	2.8600E+08	1.2800E+08	+	4.1500E+08	2.1000E+08	+	2.3978E+06	5.6934E+05	+	**7.7105E+05**	2.9092E+05	−
18	6.1579E+03	6.2214E+03	8.7500E+09	3.1100E+09	+	1.0400E+10	3.4300E+09	+	2.8801E+04	1.5262E+04	+	**1.5136E+03**	1.6635E+03	−
19	1.8874E+01	2.4604E+01	8.4500E+02	1.1500E+02	+	1.5300E+03	5.7500E+02	+	**1.0619E+01**	5.8059E-01	−	**5.7717E+00**	2.4342E-01	−
20	6.7697E+03	5.0854E+03	1.5900E+07	1.3700E+07	+	2.4200E+07	2.5300E+07	+	**4.5792E+02**	8.3335E+01	−	**2.0106E+02**	2.8480E+01	−
21	5.2699E+05	3.5949E+05	1.3300E+08	7.5000E+07	+	2.2200E+08	1.1300E+08	+	**1.8856E+05**	6.8832E+04	−	**3.9275E+04**	1.7385E+04	−
22	4.8796E+02	1.7282E+02	1.3100E+04	9.3800E+03	+	4.2600E+04	4.9000E+04	+	**2.1011E+02**	7.2336E+01	−	**1.0011E+02**	8.5275E+01	−
23	3.1573E+02	5.7379E-01	**2.0000E+02**	0.0000E+00	−	2.2700E+03	5.9100E+02	+	3.1524E+02	8.0845E-05	=	3.1524E+02	8.3143E-05	=
24	2.3065E+02	5.1711E+00	**2.0000E+02**	0.0000E+00	−	5.5900E+02	3.6300E+01	+	**2.0891E+02**	3.6346E+00	−	**2.0412E+02**	4.9267E-01	−
25	2.0518E+02	1.5118E+00	**2.0000E+02**	0.0000E+00	−	4.9700E+02	7.1400E+01	+	2.2324E+02	2.7951E+00	+	2.0973E+02	1.6944E+00	+
26	1.3786E+02	4.8383E+01	1.8600E+02	2.6800E+01	+	3.6700E+02	1.4100E+02	+	**1.0048E+02**	4.7725E-02	−	**1.0039E+02**	3.8711E-02	-
27	6.6942E+02	1.6299E+02	**2.0000E+02**	0.0000E+00	−	2.0900E+03	3.6500E+02	+	**3.8924E+02**	3.7274E+01	−	**3.3256E+02**	3.7449E+00	−
28	1.0284E+03	1.2149E+02	**2.0000E+02**	0.0000E+00	−	8.7900E+03	7.6100E+02	+	**9.7674E+02**	2.7797E+01	−	**7.9899E+02**	1.8681E+01	−
29	1.3982E+03	4.2734E+02	**2.0000E+02**	0.0000E+00	−	8.9600E+08	2.5300E+08	+	1.1761E+04	3.0527E+03	+	2.4632E+03	2.8207E+02	+
30	1.2393E+04	4.6308E+03	**2.0000E+02**	0.0000E+00	−	1.4400E+07	8.9300E+06	+	5.5898E+03	8.7287E+02	+	**2.5514E+03**	5.2341E+02	−

Table 9
Mean and Standard Deviation of error value obtained by HS-SA, SOS, GSA, CS, and CMA-ES on all functions of IEEE CEC 2014 (Dimension 30) benchmark problems. The better results are highlighted by bold.

	HS-SA		SOS			GSA			CS			CMA-ES		
Function	Mean	Std Dev	Mean	Std Dev	w	Mean	Std Dev	w	Mean	Std Dev	w	Mean	Std Dev	w
1	1.1566E+07	1.0957E+07	6.9464E+07	2.9152E+07	+	1.5688E+07	3.5776E+06	=	**2.2857E+05**	7.8599E+02	−	**9.4219E+04**	7.8851E+04	−
2	1.3831E+04	1.1345E+04	3.6133E+09	7.7578E+08	+	8.8850E+03	1.6241E+03	+	**1.3853E+02**	4.7145E+01	−	2.5541E+10	3.8502E+09	+
3	6.3081E+03	6.0633E+03	2.3789E+04	1.3521E+04	+	7.3158E+04	4.1273E+03	+	1.0476E+04	5.5381E-01	+	1.4469E+04	5.6612E+03	+
4	1.1122E+02	4.0061E+01	4.1439E+02	7.3228E+01	+	3.6846E+02	2.9801E+01	+	**7.0723E+01**	2.0152E+01	+	2.0000E+01	2.6288E-05	=
6	1.3064E+01	2.1215E+00	2.4822E+01	1.9335E+00	+	2.8755E+01	2.4932E+00	+	2.4762E+01	4.1864E+00	+	4.0855E+01	2.1285E+00	+
7	1.5191E-02	1.6342E-02	3.3820E+01	8.0225E+00	+	**1.9852E-04**	2.0184E-05	−	7.4485E-02	6.4093E-02	+	2.3115E+02	2.8258E+01	+
8	4.1039E-05	8.1336E-06	8.0952E+01	7.8952E+00	+	1.4012E+02	8.4125E+00	+	7.1183E+01	1.1201E+01	+	2.8308E+02	2.2058E+01	+
9	6.7117E+01	1.5196E+01	1.7262E+02	1.8996E+01	+	1.6489E+02	1.2005E+01	+	1.7805E+02	3.4720E+01	+	3.2810E+02	7.6521E+01	+
10	2.0148E-01	4.6566E-02	2.3950E+03	1.1958E+02	+	3.3531E+03	3.1885E+02	+	2.0203E+03	1.9374E+02	+	2.6105E+02	1.0589E+02	+
11	1.9890E+03	4.3439E+02	4.4831E+03	4.0843E+02	+	4.0567E+03	4.2429E+02	+	4.4904E+03	3.3801E+02	+	1.6864E+02	1.9833E+02	=
12	1.6399E-01	5.0047E-02	7.3655E-01	2.8584E-01	+	**9.0965E-02**	1.2252E-03	−	8.1058E-01	2.7821E-01	+	3.0284E-01	2.1796E+00	+
13	5.2412E-01	1.0432E-01	6.8510E-01	2.0744E+00	+	4.0259E-01	3.5520E-02	=	**4.1731E-01**	4.4655E-02	−	5.5079E+00	3.0711E-01	+
14	4.1455E-01	2.2860E-01	8.3855E+00	4.2866E+00	+	**2.3001E-01**	2.1930E-02	−	5.1779E-01	2.6524E-02	+	7.5318E+01	8.0755E+00	+
15	1.6392E+01	1.1688E+01	2.5625E+02	2.1086E+02	+	**1.2559E+01**	1.9874E+00	−	1.3177E+01	1.8683E+00	+	1.0215E+04	3.2423E+04	+
16	9.2634E+00	7.8286E-01	1.2023E+01	3.6985E-01	+	1.4804E+01	2.4403E-01	+	1.2314E+01	1.5985E-01	+	1.3785E+01	5.3125E-01	+
17	2.0879E+06	1.3100E+06	5.5865E+06	3.6851E+06	+	7.2880E+05	1.2488E+05	=	**1.2439E+05**	3.0053E+03	−	**5.4855E+03**	3.6247E+03	−
18	6.1579E+03	6.2214E+03	4.8625E+05	2.2476E+05	+	**3.8621E+02**	1.1468E+02	−	**1.4215E+03**	2.4209E+01	−	1.5185E+09	3.9258E+08	+
19	1.8874E+01	2.4604E+01	4.0660E+01	2.2358E+01	+	1.5678E+02	2.4896E+01	+	**1.1816E+01**	5.3125E-01	−	2.9845E+02	4.2523E+01	+
20	6.7697E+03	5.0854E+03	1.5949E+04	1.0025E+04	+	8.2447E+04	1.3421E+04	+	**1.3593E+02**	3.9858E+01	−	4.6121E+03	3.8802E+03	=
21	5.2699E+05	3.5949E+05	7.8598E+05	6.0551E+05	+	**1.7879E+05**	3.1904E+04	−	**1.6696E+03**	1.8149E+02	−	**6.8602E+03**	2.7562E+03	−
22	4.8796E+02	1.7282E+02	5.4507E+02	1.8229E+02	+	9.5109E+02	1.8234E+02	+	**3.1138E+02**	9.1532E+01	−	1.6104E+03	2.9209E+02	+
23	3.1573E+02	5.7379E-01	3.4236E+02	2.1852E+01	+	**2.0000E+02**	2.3411E-5	−	3.4374E+02	8.1932E-02	+	5.7912E+02	4.9411E+01	+
24	2.3065E+02	5.1711E+00	2.3617E+02	9.8953E+00	+	**2.0008E+02**	7.0555E-02	−	**2.2139E+02**	1.4508E+00	−	2.1203E+02	7.4908E+00	=
25	2.0518E+02	1.5118E+00	2.1495E+02	3.4960E+00	+	**2.0000E+02**	4.9228E-07	−	2.0904E+02	6.1156E-01	+	2.1207E+02	2.9699E+00	+
26	1.3786E+02	4.8383E+01	**1.0093E+02**	2.9009E-02	−	1.6935E+02	2.9081E+01	+	**1.0087E+02**	4.1891E-02	−	1.2533E+02	5.5098E+01	=
27	6.6942E+02	1.6299E+02	**4.9605E+02**	1.5103E+02	−	7.6900E+02	5.6821E+02	+	**4.1820E+02**	5.6852E+00	−	1.0692E+03	2.3008E+02	+
28	1.0284E+03	1.2149E+02	1.3208E+03	1.0582E+02	+	**7.6532E+02**	3.0818E+02	−	**9.1293E+02**	3.9422E+01	−	2.7948E+03	5.9168E+02	+
29	1.3982E+03	4.2734E+02	2.1568E+04	1.1852E+03	+	**2.0005E+02**	4.5785E-02	−	1.6858E+03	2.2585E+02	+	3.5228E+04	5.3369E+03	+
30	1.2393E+04	4.6308E+03	3.7319E+04	2.1242E+04	+	2.3115E+04	2.4398E+04	+	**3.5887E+03**	6.5226E+02	−	6.4777E+05	1.3141E+05	+

Table 10
Mean and Standard Deviation of error value obtained by HS-SA, CS2, MOCS and CS-VSF on all functions of IEEE CEC 2014 (Dimension 30) benchmark problems. The better results are highlighted by bold.

	HS-SA		CS2			MOCS			CS-VSF		
Function	Mean	Std Dev	Mean	Std Dev	w	Mean	Std Dev	w	Mean	Std Dev	w
1	1.1566E+07	1.0957E+07	3.5000E+07	2.4900E+07	+	2.9300E+06	1.5100E+06	=	9.2900E+07	2.3600E+07	+
2	1.3831E+04	1.1345E+04	1.9500E+07	5.4900E+07	+	8.1800E+03	4.7700E+03	+	6.7200E+08	1.4100E+08	+
3	6.3081E+03	6.0633E+03	3.1000E+04	1.3600E+04	+	5.6100E+02	5.8600E+02	−	3.6200E+04	8.4700E+03	+
4	1.1122E+02	4.0061E+01	2.0300E+02	6.6900E+01	+	**9.4100E+01**	4.3300E+01	−	2.8800E+02	3.5800E+01	+
5	2.0000E+01	3.1029E-04	2.0000E+01	2.2800E-03	=	2.0600E+01	1.4500E+00	+	2.0800E+01	6.6900E-02	+
6	1.3064E+01	2.1215E+00	3.2300E+01	3.2700E+00	+	2.1600E+01	8.8800E+00	+	2.9000E+01	1.6300E+00	+
7	1.5191E-02	1.6342E-02	1.7900E+00	2.1900E+00	+	3.9100E+00	9.8900E+00	+	7.0100E+00	1.3900E+00	+
8	4.1039E-05	8.1336E-06	1.7100E+02	3.4600E+01	+	3.8000E+01	2.5800E+01	+	1.6800E+02	1.2400E+01	+
9	6.7117E+01	1.5196E+01	2.8000E+02	5.1600E+01	+	1.3900E+02	1.0900E+02	+	2.0900E+02	8.4600E+00	+
10	2.0148E-01	4.6566E-02	2.6600E+03	5.3400E+02	+	7.9800E+02	1.1100E+03	+	4.6100E+03	2.9900E+02	+
11	1.9890E+03	4.3439E+02	4.1300E+03	5.3500E+02	+	2.7700E+03	1.2800E+03	+	6.2800E+03	2.4800E+02	+
12	1.6399E-01	5.0047E-02	5.1100E-01	2.5600E-01	+	6.1500E-01	2.9100E-01	+	3.5800E-01	5.5600E-01	+
13	5.2412E-01	1.0432E-01	4.8100E-01	1.1700E-01	=	**3.2400E-01**	7.2700E-02	−	**4.7400E-01**	6.3200E-02	−
14	4.1455E-01	2.2860E-01	**3.0800E-01**	5.6400E-02	−	6.3100E+00	1.5800E+01	+	3.6600E-01	7.7200E-02	=
15	1.6392E+01	1.1688E+01	9.8000E+01	3.0200E+01	+	4.1100E+01	1.6000E+01	+	5.1100E+01	1.1800E+01	+
16	9.2634E+00	7.8286E-01	1.2700E+01	5.0100E-01	+	3.9700E+04	1.1400E+05	+	1.2500E+01	1.8800E-01	+
17	2.0879E+06	1.3100E+06	**1.4800E+06**	1.2100E+06	−	1.7400E+05	1.3400E+05	=	**1.7200E+06**	6.4900E+05	−
18	6.1579E+03	6.2214E+03	7.6700E+03	6.7000E+03	+	**6.9400E+02**	9.5100E+02	−	**2.4700E+03**	2.6600E+03	−
19	1.8874E+01	2.4604E+01	5.3300E+01	3.6300E+01	+	5.8800E+02	1.5500E+03	+	**1.3900E+01**	2.0600E+00	−
20	6.7697E+03	5.0854E+03	3.9300E+04	2.2000E+04	+	2.3100E+04	5.5500E+04	+	7.4300E+03	3.3100E+03	+
21	5.2699E+05	3.5949E+05	3.5400E+05	3.4800E+05	=	**7.5800E+04**	6.2800E+04	-	2.4300E+05	1.0800E+05	=
22	4.8796E+02	1.7282E+02	9.4700E+02	3.3100E+02	+	8.0800E+02	3.0500E+02	+	3.0300E+02	1.0600E+02	=
23	3.1573E+02	5.7379E-01	3.2900E+02	7.5100E+00	+	**3.0300E+02**	3.0400E+01	-	3.2100E+02	9.7500E-01	+
24	2.3065E+02	5.1711E+00	2.7800E+02	3.1100E+01	+	2.2500E+02	5.9000E+00	=	2.5200E+02	2.9100E+00	+
25	2.0518E+02	1.5118E+00	2.2300E+02	9.3900E+00	+	2.0800E+02	1.6000E+01	+	2.2100E+02	2.1900E+00	+
26	1.3786E+02	4.8383E+01	**1.0000E+02**	1.6300E-01	-	2.4300E+02	1.6600E+02	+	**1.0000E+02**	6.6400E-02	-
27	6.6942E+02	1.6299E+02	**4.2700E+02**	1.9600E+01	-	1.3200E+03	1.0700E+03	+	**5.1800E+02**	4.0000E+01	-
28	1.0284E+03	1.2149E+02	3.4900E+03	5.4800E+02	+	3.3400E+03	8.0200E+02	+	**9.0000E+02**	7.8800E+01	-
29	1.3982E+03	4.2734E+02	5.4400E+05	2.6100E+06	+	2.1100E+03	4.7000E+02	+	**2.2800E+02**	2.9000E+00	-
30	1.2393E+04	4.6308E+03	2.4900E+04	2.2600E+04	+	**2.9600E+03**	8.4600E+02	-	**7.1000E+02**	1.1400E+02	-

HS-SA significantly outperformed Gravitational Search Algorithm (GSA) [32] on two instances of unimodal functions whereas the results are not statistically significant in the remaining one instance. HS-SA is the winner on seven instances of simple multimodal functions, GSA is the winner on four instances and the difference in performance is not statistically significant on one instances. In case of hybrid multimodal functions HS-SA outperformed GSA on three instances whereas latter is the winner on two instances. In case of composition functions HS-SA outperformed GSA on three instances whereas latter outperformed former in the remaining five instances

HS-SA significantly outperformed Cuckoo Search Algorithm (CS) [33] on two instances whereas latter is the winner in the remaining one instance. In case of simple multimodal functions HS-SA outperformed CS on twelve instances whereas latter is the winner in the remaining one instance. CS outperformed HS-SA on all instances of hybrid multimodal functions. In case of composition functions HS-SA outperformed CS on three instances whereas latter outperformed former in the remaining five instances.

HS-SA significantly outperformed Evolution Strategy with Covariance Matrix Adaptation (CMA-ES) [34] on two instances whereas latter is the winner in the remaining one instance. In case of simple multimodal functions HS-SA outperformed CS on eleven instances and in the remaining two instances the results are not statistically significant. On three instances of hybrid multimodal functions HS-SA is the winner whereas on the remaining two instances CMA-ES is the winner, the difference is not statistically significant in one instance.

HS-SA significantly outperformed Standard Cuckoo Search (CS2) [35] on all the three instances of unimodal functions. In case of simple multimodal functions HS-SA outperformed CS2 on ten instances whereas latter is the winner in one instance, the difference is not significant in remaining two instances. HS-SA outperformed CS2 on four instances of hybrid multimodal functions whereas latter is the winner on just one instance. In case of composition functions HS-SA outperformed CS2 on six instances whereas latter outperformed former in the remaining two instances.

Both HS-SA and MOCS [36] are winners in one instance of unimodal functions and the difference is statistically insignificant in the remaining one instance. In case of simple multimodal functions HS-SA outperformed MOCS on eleven instances whereas latter is the winner in only two instances. HS-SA outperformed MOCS on three instances of hybrid multimodal functions whereas latter is the winner on two instances and the difference is statistically insignificant in the remaining one instance. In case of composition functions HS-SA outperformed MOCS on five instances whereas latter outperformed former on two instances the difference in performance is statistically insignificant in the remaining one instance.

HS-SA significantly outperformed CS-VSF [37] on all the three instances of unimodal functions. In case of simple multimodal functions HS-SA outperformed CS-VSF on eleven instances whereas latter is the winner in just one instance, the difference is not statistically significant in remaining one instance. HS-SA outperformed CS-VSF on one instance of hybrid multimodal functions whereas latter is the winner on three instances, the difference in performance is statistically insignificant in the remaining two instances. In case of composition functions HS-SA outperformed CS-VSF on three instances whereas latter outperformed former in the remaining five instances.

6. Camera calibration

The problem of camera calibration has been studied extensively in photogrammetry and computer vision community because of its important applications such as vehicle guidance, robotic navigation and 3D-reconstruction. Camera calibration problem deals with finding the geometrical relationship between the 3D scene and its 2D images taken by the camera(s). It defines exactly how the scene has been projected by the camera to result in the given image(s). Camera calibration involves:

1. Determination of the orientation and position of the camera with respect to the scene which is specified by extrinsic parameters.
2. Determination of the internal geometric and optimal characteristics of the camera, which is specified by intrinsic parameters.

Given a point in a scene extrinsic parameters transform its 3-D world coordinates into 2-D camera coordinates, which are then transformed to 2-D image coordinates using intrinsic parameters.

The existing methods for camera calibration can be broadly categorized as linear [38] and non-linear [39] approaches. In linear methods lens distortion is not taken into consideration, thus calibration accuracy is very low to meet the requirements of commercial machine vision application. Even though nonlinear approaches provide more precise solution, however they have the limitation of being computationally extortionate and require precise initial estimates. Other standard procedure for camera calibration is “two-step” method [40], in the first step linear approach is used to generates an approximate solution, which is improved by using a non linear iterative process in the second step. The first step (linear approach) is key to the success of this procedure. Approximate solution provided by the linear technique must be precise enough for the subsequent nonlinear techniques to converge correctly . The existing linear techniques are notorious for their lack of accuracy and robustness because of being susceptible to noise in image coordinates [41]. Haralick et al. [42] showed that when the noise level exceeds a threshold or the number of control points is low these methods become extremely unstable and are error prone. This problem can be alleviated by use of more control points, however fabrication of more control points is an expensive, difficult and time consuming process. In case of applications with small number of control points (close to minimum required), it is not possible for linear procedures to consistently provide good initial solution for the subsequent non linear methods to find the optimal solution.

Another limitation is that virtually all nonlinear techniques use variants of conventional optimization techniques like conjugate gradient, gradient descent or Newton method. They therefore all inherent well known problems associated with these traditional methods of optimization namely sensitivity of getting trapped in local extrema and slow convergence. The problem is more prominent if objective function landscape contains isolated valleys or broken ergodicity. Camera calibration objective function involves 12 parameters and thus leads to complex error surface with the global minimum hidden among numerous local extrema. Therefore the risk of obtaining local rather than global optimum is very high with conventional methods.

To alleviate the problem with the conventional camera calibration techniques many metaheuristc algorithms have been proposed. Hati et al. [43] has used a binary coded genetic algorithm for solving camera calibration, whereas real coded genetic algorithm has been used in [44,45]. Even though impressive results have been shown in [45] both in terms of parameter values and pixel error, however the authors have not taken lenes distortion into consideration.

In this article music inspired Harmony Search (HS) algorithm [1] is used for camera calibration. The reason for choosing Harmony Search is: it requires only few free parameters to adjust, converges quickly and has been proven to be robust in solving a variety of non linear optimization problems. The algorithm proposed in this paper takes lenes distortion into consideration and hence results in 12 dimensional highly non linear search space with a number of local optima.

Researchers have used Artificial neural networks to solve the problem of calibration in [46], however the main drawback of employing neural networks to solve this problem is estimation of good initial value of weight vector of network. Particle swarm algorithm has been utilized to solve camera calibration problem in [47].

6.1. Perspective geometry

In this section the camera model defining the geometry of the image formation is presented. Fig. 1 shows the pinhole model for two cameras. The stereo camera system consists of five coordinate frames: a world reference frame (X, Y, Z), two camera frames (X_{ci}, Y_{ci}, Z_{ci}) and two image frames $(u_i, v_i), i = 1, 2$. The origins O_1 and O_2 of the cameras coordinate systems coincide with their corresponding optical centers and their Z coordinate axes are collinear with corresponding optical axes, which are perpendicular to corresponding image planes and intersect them in their principal points. The image plane of

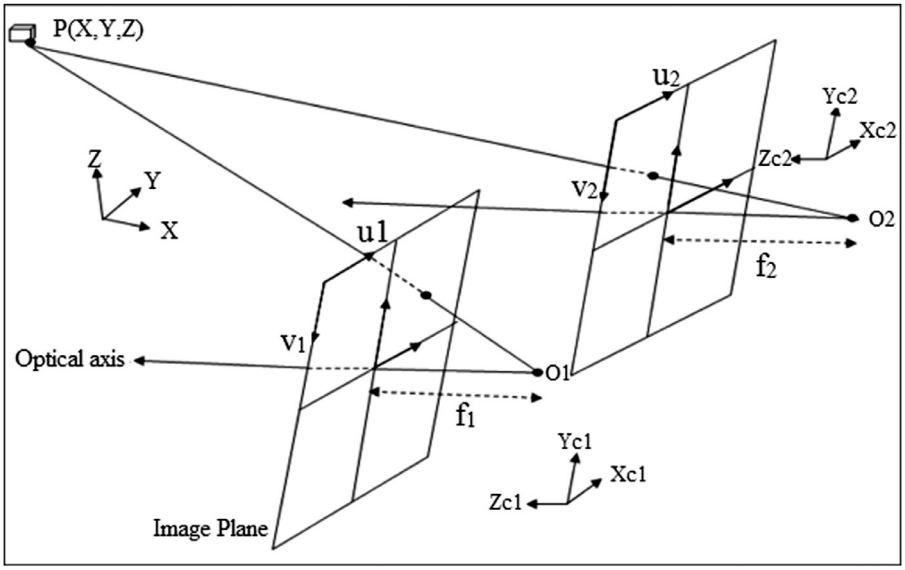


Fig. 1. Stereo camera calibration model reproduced from Deep et al. [48].

each camera is at a distance f_i, $i = 1, 2$ (its focal length) apart from the corresponding optical center. The transformation of 3-D world coordinates into 2-D image coordinates by a camera involves following four steps.

1. Eq. (3) transforms the 3-D world coordinates to 2-D camera coordinates:

$$\begin{pmatrix} X_c \\ Y_c \\ Z_c \end{pmatrix} = R.\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} + T, \tag{3}$$

R is a 3×3 rotation matrix used to represents the orientation of the camera relative to the world coordinate system and T is the translation vector used to represent position of the camera relative to world coordinate system. Further, R and T may be expressed as

$$R = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix}, \quad T = \begin{pmatrix} T_x \\ T_y \\ T_z \end{pmatrix}$$

The elements r_{ij} of the matrix R can further be expressed in terms of swing, tilt and pan angle (α, β, γ) as

$$\begin{aligned} r_{11} &= cos\alpha\ cos\beta \\ r_{12} &= sin\alpha\ cos\gamma + cos\alpha\ sin\beta\ sin\gamma \\ r_{13} &= sin\alpha\ sin\gamma - cos\alpha\ sin\beta\ cos\gamma \\ r_{21} &= -sin\alpha\ cos\beta \\ r_{22} &= cos\alpha\ cos\gamma - cos\alpha\ sin\beta\ sin\gamma \\ r_{23} &= cos\alpha\ sin\gamma - sin\alpha\ sin\beta\ cos\gamma \\ r_{31} &= sin\beta \\ r_{32} &= -cos\beta\ sin\gamma \\ r_{33} &= cos\beta\ cos\gamma \end{aligned}$$

2. The 3-D camera coordinates (X_c, Y_c, Z_c) are then transformed into ideal retinal coordinates (X_u, Y_u) of the camera using perspective projection equations:

$$X_u = f\frac{X_c}{Z_c}, \quad Y_u = f\frac{Y_c}{Z_c} \tag{4}$$

3. The ideal retinal coordinates (X_u, Y_u) are then transformed into actual retinal coordinates (X_d, Y_d) by considering lens radial distortion coefficient k:

$$X_d = X_u(1 + kr^2)^{-1}, \quad Y_d = Y_u(1 + kr^2)^{-1} \tag{5}$$

The distance between the image center and the actual retinal coordinates is denoted by r.

4. Then the actual retinal coordinates (X_d, Y_d) are transformed into pixel coordinates (u,v):

$$u = X_d N_x + u_0, \quad v = Y_d N_y + v_0 \tag{6}$$

where (u_0, v_0) represents the pixel coordinates of the image center; N_x and N_y is the number of pixels that are contained in the unit distance of the image plane along the X and Y axes, respectively.

Combining the above four steps the image formation process of a 3-D point P(X,Y,Z) can be written as:

$$N_s \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = Q \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{7}$$

where N_s is a 3×3 matrix representing scaling factor and matrix Q is decomposable into two matrices as: $Q = AD$ with $D = [RT]$, and A being a 3×3 matrix called the intrinsic matrix and can be expressed fully in terms of the intrinsic parameters of camera (f, k, N_x, N_y, u_0, v_0). The matrix Q can be formulated as follows:

$$Q = [q_1^T \; q_2^T \; q_3^T] \tag{8}$$

where q_i^T, $(i = 1, 2, 3)$ represent the rows of Q. Using Eqs. (7) and (8),the pixel coordinates (u,v) can be represented as

$$u = \frac{q_1^T P}{q_3^T P}, \quad v = \frac{q_2^T P}{q_3^T P} \tag{9}$$

Using Eq. (9), the 3-D world coordinates of a point P(X, Y, Z) can be transformed into pixel coordinates (u, v) for a single camera. This process being irreversible i.e. the 3-D position of point P cannot be uniquely determined from its pixel coordinates (u, v). However, from its stereo images, $p_l(u_l, v_l)$ and $p_r(u_r, v_r)$, taken by two cameras (or a single camera in two different positions) a point P(X, Y, Z) in the 3-D world can be uniquely determined.

6.2. Mathematical formation of camera calibration problem

The camera calibration problem is mathematically expressed as a nonlinear optimization problem in which the objective is to minimize the square root of the sum of squares of Euclidean distances between the observed pixel positions and their corresponding calculated pixel positions. Thus, the camera calibration problem can be stated as follows:

Given N number of control points whose world coordinates $P_i(X_i, Y_i, Z_i)$ and observed pixel positions are known with high precision, the problem is to determine the optimal values of 12 decision variables (u_o, v_o, N_x, N_y, f, k, α, β, γ, T_x, T_y, T_z) that minimize the objective function given by Eq. (10):

$$F = \sqrt{\sum_{i=1}^{N} \left[\left(\frac{q_1^T P}{q_3^T P} - u \right)^2 + \left(\frac{q_2^T P}{q_3^T P} - v \right)^2 \right]} \tag{10}$$

Based on the knowledge of camera any suitable range for decision variables can be chosen, as an example we can choose $\beta \in [-\pi, \pi]$.

6.3. Solving camera calibration problem with proposed HS-SA algorithm

For solving the camera calibration problem with HS-SA algorithm, let H denote a vector consisting of the unknown intrinsic and extrinsic parameters of the camera, that is

$$H = (u_o, v_o, N_x, N_y, f, k, \alpha, \beta, \gamma, T_x, T_y, T_z) \tag{11}$$

For convenience we may write $H = (h_1, h_2, h_3, \ldots, h_{12})$, with h_i, $i = 1, 2, \ldots, 12$ representing the camera parameters in the same order as given in Eq. (11). Then vector H with all its components within their bounds represents a potential solution to the problem.

The HS-SA algorithm starts by initializing all the harmonies in the Harmony memory, each harmony in the HM represents a potential solution and has the same structure as defined for H in Eq. (11). Every component of the harmony is randomly initialized with in the allowed bounds. Once the initialization phase is over potential solutions in HM are evaluated using fitness function defined as Eq. (10). Eq. (10) gives the error value (i.e. difference between the actual and obtained coordinated) of the control points and thus the objective of the algorithm is to minimize Eq. (10).

While generating the ith component of the new harmony say H i.e. H_i, $1 \le i \le 12$ either the ith component of some existing harmony from HM is selected with probability HMCR (step 7 and 8 of Algorithm 3) or it is randomly generated with in the allowed bounds (step 14). In case the ith component of new harmony is selected from HM, it is further adjusted with in the allowed Band width (BW) range with probability PAR (step 10, 11).

The new harmony H is evaluated using Eq. (10) and replaces the worst harmony of HM, if it is better than it. Incase H is inferior than the worst harmony, it is accepted with a probability determined by parameter Temperature (T). The parameter T is linearly decreased during the execution of algorithm (step 21) so as to decrease the probability of favoring inferior solutions and thus the algorithm gradually shift focus from exploration of search space to exploitation of promising search areas.

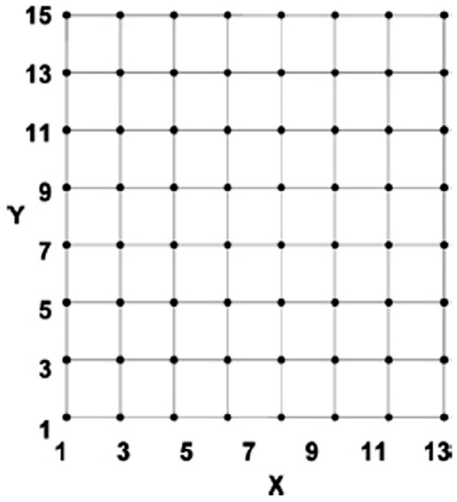


Fig. 2. Calibration chart.

Table 11
Camera parameter bounds and Ground truth values for camera.

Parameter	Ground truth	Bounds
f	10	[5, 15]
N_x	200	[170, 230]
N_y	200	[170, 230]
u_0	20	[15, 25]
v_0	19	[15, 25]
K	0.15152	[0, 0.5]
T_x	60	[20, 80]
T_y	35	[25, 45]
T_z	1210	[1000, 1400]
α	0	$[-\pi/2,\ \pi/2]$
β	0	$[-\pi/2,\ \pi/2]$
γ	0	$[-\pi/2,\ \pi/2]$

7. Experimentation on camera calibration

This section presents extensive experiments with HS-SA algorithm so as to evaluates its performance on camera calibration problem. The simulations has been carried out using varying number of control points. The accuracy is evaluated by calculating camera errors and pixel errors. Pixel error is defined as the root mean square Euclidean distance between the observed pixel positions and the corresponding re projected pixel positions calculated using estimated parameters values, whereas camera error is defined as the mean Euclidean distance between the ground truth and the estimated value of that parameter.

The parameter setting given in Table 1 has been adopted for all the competing algorithms . The stopping criteria for all the algorithm is 100,000 function evaluations of objective function (Eq. (10)).

The algorithms has been implemented in Dev C++ 5.0 and the experimentation has been carried out on a laptop with Windows 10 operating system, intel core i3 processor and 8GB of RAM.

7.1. Generation of synthetic data

In order to generate the synthetic data a calibration chart shown as Fig. 2, having 8×8 grids in X and Y directions has been utilized to obtain 64 grid points. This calibration chart is shifted on eleven different positions along Z direction to get a total of 704 points $P_i(X_i, Y_i, Z_i)$ in 3-D space. The values of intrinsic and extrinsic parameters that were used for calculating the 2-D image coordinates from the 3-D grid points are called the ground truth values. The ground truth values for camera along with the parameter bounds used in the experimentation is shown as Table 11.

Table 12
Best Parameter values obtained in 50 runs (HS-SA).

Parameters	N=5	N=10	N=50	N=100	N=200	N=300	N=400	N=500
f	8.253	7.06	9.924	12.397	15	12.314	11.861	7.643
N_x	217.986	209.671	214.286	212.658	184.347	188.408	220.348	221.912
N_y	230	206.564	216.72	213.266	186.513	188.791	225.93	222.688
u_0	20.927	15.656	15.812	19.831	24.769	21.664	22.64	15.313
v_0	22.181	23.051	24.212	18.475	21.495	17.959	18.754	19.42
K	0.174	0.024	0.191	0.379	0.374	0.214	0.297	0.103
T_x	48.173	79.541	76.334	66.077	79.399	65.158	60.469	61.69
T_y	39.664	31.662	41.447	25	27.465	41.67	40.733	31.985
T_z	1002.058	1000.33	1226.186	1004.9	1157.501	1331.794	1335.615	1004.411
α	0.00104	0.0008	−0.00065	−0.00152	−0.00193	0.00056	0.0016	−0.00082
β	−0.00934	0.00836	0.00525	0.00352	0.01813	0.00533	0.00283	−0.00453
γ	−0.00663	−0.0026	−0.01049	0.00731	0.002	−0.00317	−0.00372	0.00139

Table 13
Statistics of pixel error for varying number of control points (HS-SA).

Control points (N)	Mean pixel error	Min. pixel error	Max. pixel error	SD pixel error	Calibration time(s)
5	2.428	0.106	3.553	0.83	0.549
10	1.752	0.327	3.473	0.885	0.58
50	1.45	0.216	3.104	0.639	0.779
100	1.347	0.07	3.104	0.642	1.056
200	1.198	0.153	2.921	0.525	1.585
300	1.093	0.223	3.105	0.716	2.111
400	0.85	0.066	2.559	0.71	2.652
500	0.802	0.023	2.592	0.679	3.179

7.2. Results and discussion

7.2.1. Varying number of control points

The simulations are performed for 5, 10, 50, 100, 200, 300, 400, 500 control points and the experiment is repeated 50 times for each control point. It is not possible to show the results of parameter values of all the experiments in tabular form or graphically, so the result of best run has only been shown in terms of parameter values (Table 12) and the result matrices of all runs have been shown in terms of pixel error (Table 13) for varying number of control points (N). Table 13 shows the mean, maximum, minimum and standard deviation of pixel error for various control points along with average calibration time in seconds.

Most of the parameter values obtained in Table 12 seem close to ground values. However no specific observation can be made regarding the effect of number of control points on accuracy of parameter values. Table 13 reveals with the increase in number of control points the average pixel error decreases. Calibrating with control points as few as 5, the proposed algorithm shows very small pixel error (minimum 0.106) and thus is effective even at smaller number of control points. This is an important achievement because creating the control points is difficult and time consuming job in practice. Increase in number of control points results in decrease of pixel error. The minimum and mean of pixel error is respectively 0.106 and 2.428 when the number of control points is 5 with the increase in number of control points it gets reduced and finally becomes 0.023 and 0.802 when the number of control points is 500. With the increase in number of control points the complexity of objective function increases (Eq. (10)) and hence the execution time also increases. HS-SA is very efficient in terms of time complexity as the execution time is only few second even for large number of control points.

7.2.2. Varying levels of image noise

The purpose of simulations with synthetic data is to test the applicability of the algorithm for calibration with real images. Because in real images there are various causes of noise, HS-SA has been tested with noisy synthetic data. Independent Gaussian noise with zero mean and standard deviation of 0.02 and 0.05 is added to ideal pixel positions and the effect of varying image noise on estimated pixel error is respectively shown in Tables 14 and 15 for varying number of control points. The addition of noise causes the system to become highly unstable for smaller number of control points (5 and 10) and the obtained calibration values are simply unacceptable. However once the number of control points is 50 the system becomes stable and the average pixel error is approximately 1.5 . Increase in number of control points causes reduction in pixel error. Thus the proposed algorithm can be efficiently used in calibrating real images if the number of control points is chosen judicially.

Table 14
Statistics of pixel error for noise data (standard deviation of Gaussian noise equal to 0.02).

Control points (N)	Mean pixel error	Min. pixel error	Max. pixel error	SD pixel error	Calibration time(s)
5	4434781.517	4434779.5	4434783.5	1.087	0.527
10	3216495.577	3216494.25	3216497.25	0.735	0.574
50	1.506	0.203	2.942	0.774	0.766
100	1.362	0.045	2.897	0.696	1.038
200	1.136	0.12	3.205	0.689	1.586
300	1.057	0.057	3.08	0.73	2.143
400	0.858	0.045	2.931	0.711	2.595
500	1.096	0.088	3.12	0.795	3.192

Table 15
Statistics of pixel error for noise data (standard deviation of Gaussian noise equal to 0.05).

Control points (N)	Mean pixel error	Min. pixel error	Max. pixel error	SD pixel error	Calibration time(s)
5	5731818.827	5731816.5	5731820	0.919	0.553
10	4004632.209	4004631	4004634	0.814	0.576
50	1.68	0.099	3.039	0.733	0.791
100	1.484	0.321	3.131	0.692	1.053
200	1.352	0.123	2.532	0.701	1.591
300	1.306	0.078	2.846	0.829	2.137
400	1.059	0.088	2.485	0.683	2.653
500	1.049	0.091	2.736	0.737	3.148

Table 16
Statistics of pixel error for varying number of control points (HS).

Control points (N)	Mean pixel error	Min. pixel error	Max. pixel error	SD pixel error	Calibration time(s)
5	2.81	0.499	3.997	0.756	0.519
10	1.995	0.233	3.471	0.801	0.554
50	1.761	0.369	3.216	0.711	0.746
100	1.51	0.341	3.05	0.73	1.006
200	1.32	0.192	3.04	0.709	1.507
300	1.213	0.164	3.09	0.742	2.014
400	0.917	0.018	2.76	0.795	2.52
500	0.876	0.03	2.788	0.729	3.028

Table 17
Statistics of pixel error for varying number of control points (SA).

Control points (N)	Mean pixel error	Min. pixel error	Max. pixel error	SD pixel error	Calibration time(s)
5	2.658	0.158	4.062	0.921	0.217
10	1.775	0.221	3.512	0.67	0.229
50	1.652	0.011	3.076	0.788	0.333
100	1.38	0.095	3.299	0.728	0.46
200	1.252	0.17	2.887	0.713	0.717
300	1.116	0.032	2.822	0.768	0.973
400	0.945	0.104	2.548	0.717	1.233
500	1.016	0.101	2.663	0.647	1.531

7.3. Comparison of HS-SA with standard Harmony search and Simulated Annealing

The section evaluates the performance of standard HS and Simulated Annealing on Camera calibration problem and compares it with HS-SA Algorithm. Tables 16 and 17 respectively show the statistics in terms of pixel error for 5, 10, 50, 100, 200, 300, 400 and 500 control points. Comparing the mean pixel error from Tables 13, 16 and 17 it can be observed, HS-SA algorithm has consistently outperformed both HS and SA in terms of mean pixel error irrespective of number of control points used for calibration. Due to the relatively simple structure of SA its time complexity is lower than both HS and HS-SA whereas HS and HS-SA have almost same time complexity.

8. Conclusion

This manuscript introduces a hybrid variant of Harmony Search algorithm for continuous optimization problems with the aim to exhibited the desired behavior of exploring the search space at the earlier iterations and exploiting good solutions towards the later iterations. This is achieved by initially setting the parameter Temperature to a high value so as to favor in-

ferior moves. The Temperature parameter is linearly reduced to gradually shift the focus from exploration of search space to exploitation of promising search areas. The performance of proposed HS-SA algorithm is evaluated as per the specifications laid down in IEEE CEC 2014 benchmark suit and on camera calibration problem. It is established by statistical tests that the proposed algorithm is highly efficient.

References

[1] Z.W. Geem, J.H. Kim, G. Loganathan, A new heuristic optimization algorithm: harmony search, Simulation 76 (2) (2001) 60–68.
[2] A. Abraham, C. Grosan, H. Ishibuchi, Hybrid Evolutionary Algorithms, Springer-Verlag Berlin Heidelberg, 2007.
[3] J. Liang, B. Qu, P. Suganthan, Problem definitions and evaluation criteria for the cec 2014 special session and competition on single objective real-parameter numerical optimization, Comput. Intell. Lab. (2013).
[4] D. Manjarres, I. Landa-Torres, S. Gil-Lopez, J. Del Ser, M.N. Bilbao, S. Salcedo-Sanz, Z.W. Geem, A survey on applications of the harmony search algorithm, Eng. Appl. Artif. Intell. 26 (8) (2013) 1818–1831.
[5] A. Assad, K. Deep, Applications of harmony search algorithm in data mining: A survey, in: Proceedings of Fifth International Conference on Soft Computing for Problem Solving, Springer, 2016, pp. 863–874.
[6] D. Weyland, A rigorous analysis of the harmony search algorithm: how the research community can be, Model. Anal. Appl. Metaheuristic Comput. 72 (2012).
[7] M.P. Saka, O. Hasançebi, Z.W. Geem, Metaheuristics in structural optimization and discussions on harmony search algorithm, Swarm Evol. Comput. 28 (2016) 88–97.
[8] X.-S. Yang, Harmony Search as a Metaheuristic Algorithm, in: Music-inspired harmony search algorithm, Springer, 2009, pp. 1–14.
[9] O. Mohd Alia, R. Mandava, The variants of the harmony search algorithm: an overview, Artif. Intell. Rev. 36 (1) (2011) 49–68.
[10] M. Mahdavi, M. Fesanghary, E. Damangir, An improved harmony search algorithm for solving optimization problems, Appl. Math. Comput. 188 (2) (2007) 1567–1579.
[11] M.G. Omran, M. Mahdavi, Global-best harmony search, Appl. Math. Comput. 198 (2) (2008) 643–656.
[12] P. Saka M, O. Hasancebi, Adaptive harmony search algorithm for design code optimization of steel structures, in: Harmony Search Algorithms for Structural Design Optimization, Springer, 2009, pp. 79–120.
[13] C.-M. Wang, Y.-F. Huang, Self-adaptive harmony search algorithm for optimization, Expert Syst. Appl. 37 (4) (2010) 2826–2837.
[14] Y. Cheng, L. Li, T. Lansivaara, S. Chi, Y. Sun, An improved harmony search minimization algorithm using different slip surface generation methods for slope stability analysis, Eng. Optim. 40 (2) (2008) 95–115.
[15] S. Das, A. Mukhopadhyay, A. Roy, A. Abraham, B.K. Panigrahi, Exploratory power of the harmony search algorithm: analysis and improvements for global numerical optimization, IEEE Trans. Syst. Man Cyber. Part B (Cybernetics) 41 (1) (2011) 89–106.
[16] N. Taherinejad, Highly reliable harmony search algorithm, in: Circuit Theory and Design, 2009. ECCTD 2009. European Conference on, IEEE, 2009, pp. 818–822.
[17] Q.-K. Pan, P.N. Suganthan, M.F. Tasgetiren, J.J. Liang, A self-adaptive global best harmony search algorithm for continuous optimization problems, Appl. Math. Comput. 216 (3) (2010) 830–848.
[18] P. Yadav, R. Kumar, S.K. Panda, C. Chang, An intelligent tuned harmony search algorithm for optimisation, Inf. Sci. 196 (2012) 47–72.
[19] M. El-Abd, An improved global-best harmony search algorithm, Appl. Math. Comput. 222 (2013) 94–106.
[20] C. Blum, A. Roli, Hybrid metaheuristics: an introduction, in: Hybrid Metaheuristics, Springer, 2008, pp. 1–30.
[21] Z.W. Geem, Particle-swarm harmony search for water network design, Eng. Optim. 41 (4) (2009) 297–311.
[22] Y.C. Lee, A.Y. Zomaya, Interweaving heterogeneous metaheuristics using harmony search, in: Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on, IEEE, 2009, pp. 1–8.
[23] M. Fesanghary, M. Mahdavi, M. Minary-Jolandan, Y. Alizadeh, Hybridizing harmony search algorithm with sequential quadratic programming for engineering optimization problems, Comput. Methods Appl. Mech. Eng. 197 (33) (2008) 3080–3091.
[24] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, et al., Optimization by simmulated annealing, Science 220 (4598) (1983) 671–680.
[25] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, E. Teller, Equation of state calculations by fast computing machines, J. Chem. Phys. 21 (6) (1953) 1087–1092.
[26] Z. Xinchao, Simulated annealing algorithm with adaptive neighborhood, Appl. Soft Comput. 11 (2) (2011) 1827–1836.
[27] C. García-Martínez, M. Lozano, F.J. Rodríguez-Díaz, A simulated annealing method based on a specialised evolutionary algorithm, Appl. Soft Comput. 12 (2) (2012) 573–588.
[28] Z.-H. Zhan, J. Zhang, Y. Li, H.S.-H. Chung, Adaptive particle swarm optimization, IEEE Trans. Syst. Man, Cybern. Part B (Cybernetics) 39 (6) (2009) 1362–1381.
[29] E. Cuevas, M. Cienfuegos, D. Zaldívar, M. Pérez-Cisneros, A swarm optimization algorithm inspired in the behavior of the social-spider, Expert Syst. Appl. 40 (16) (2013) 6374–6384.
[30] S.-M. Guo, C.-C. Yang, P.-H. Hsu, J.S.-H. Tsai, Improving differential evolution with a successful-parent-selecting framework, IEEE Trans. Evol. Comput. 19 (5) (2015) 717–730.
[31] M.-Y. Cheng, D. Prayogo, Symbiotic organisms search: a new metaheuristic optimization algorithm, Comput. Struct. 139 (2014) 98–112.
[32] E. Rashedi, H. Nezamabadi-Pour, S. Saryazdi, Gsa: a gravitational search algorithm, Inf. Sci. 179 (13) (2009) 2232–2248.
[33] J. Huang, L. Gao, X. Li, An effective teaching-learning-based cuckoo search algorithm for parameter optimization problems in structure designing and machining processes, Appl. Soft Comput. 36 (2015) 349–356.
[34] N. Hansen, S.D. Müller, P. Koumoutsakos, Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es), Evol. Comput. 11 (1) (2003) 1–18.
[35] X.-S. Yang, S. Deb, Cuckoo search via lévy flights, in: Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on, IEEE, 2009, pp. 210–214.
[36] S. Walton, O. Hassan, K. Morgan, M. Brown, Modified cuckoo search: a new gradient free optimisation algorithm, Chaos, Solitons Fractals 44 (9) (2011) 710–718.
[37] L. Wang, Y. Yin, Y. Zhong, Cuckoo search with varied scaling factor, Front. Comput. Sci. 9 (4) (2015) 623–635.
[38] W. Faig, Calibration of close-range photogrammetric systems: mathematical formulation, Photogramm. Eng. Remote Sensing 41 (12) (1975).
[39] I. Sobel, On calibrating computer controlled cameras for perceiving 3-d scenes, Artif. Intell. 5 (2) (1974) 185–198.
[40] J. Weng, P. Cohen, M. Herniou, Camera calibration with distortion models and accuracy evaluation, IEEE Trans. Pattern Anal. Mach. Intell. (10) (1992) 965–980.
[41] X. Wan, G. Xu, Camera parameters estimation and evaluation in active vision system, Pattern Recognit. 29 (3) (1996) 439–447.
[42] R.M. Haralick, H. Joo, C.-N. Lee, X. Zhuang, V.G. Vaidya, M.B. Kim, Pose estimation from corresponding point data, Syst. Man Cybern. IEEE Trans. 19 (6) (1989) 1426–1446.
[43] S. Hati, S. Sengupta, Robust camera parameter estimation using genetic algorithm, Pattern Recognit. Lett. 22 (3) (2001) 289–298.
[44] S. Kumar, M. Thakur, B. Raman, N. Sukavanam, Stereo camera calibration using real coded genetic algorithm, in: TENCON 2008-2008 IEEE Region 10 Conference, IEEE, 2008, pp. 1–5.
[45] Q. Ji, Y. Zhang, Camera calibration with genetic algorithms, Syst. Man Cybern. Part A 31 (2) (2001) 120–130.

[46] Y. Xing, J. Sun, Z. Chen, Analyzing and improving of neural networks used in stereo calibration, in: Natural Computation, 2007. ICNC 2007. Third International Conference on, 1, IEEE, 2007, pp. 745–749.
[47] J.-d. Zhang, J.-g. Lu, H.-l. Li, M. Xie, Particle swarm optimisation algorithm for non-linear camera calibration, Int. J. Innovative Comput. Appl. 4 (2) (2012) 92–99.
[48] K. Deep, M. Arya, M. Thakur, B. Raman, Stereo camera calibration using particle swarm optimization, Appl. Artif. Intell. 27 (7) (2013) 618–634.