Applied Soft Computing 68 (2018) 125–135

Contents lists available at ScienceDirect

Applied Soft Computing

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

A particle swarm approach for tuning washout algorithms in vehicle simulators

Sergio Casas [a,*], Cristina Portalés [b], Pedro Morillo [a], Marcos Fernández [a]

[a] Computer Science Department, ETSE, Universitat de València, Av. de la Universitat s/n, 46100, Burjassot, Valencia, Spain
[b] IRTIC, Universitat de València, C/Catedrático José Beltrán 2, 46980, Paterna, Valencia, Spain

ARTICLE INFO

Article history:
Received 27 June 2017
Received in revised form 7 January 2018
Accepted 27 March 2018
Available online 3 April 2018

Keywords:
Optimization
Particle swarm
Motion cueing
Washout
MCA
Tuning
Vehicle simulation
Motion platform

ABSTRACT

The MCA tuning problem involves finding the most appropriate values for the parameters (or coefficients) of Motion Cueing Algorithms (MCA), also known as *washout algorithms*. These algorithms are designed to control the movements of the robotic mechanisms, referred to as motion platforms, employed to generate inertial cues in vehicle simulators. This problem can be approached in several different ways. The traditional approach is to perform a manual *pilot-in-the-loop* subjective tuning, using the opinion of several pilots/drivers to guide the process. A more systematic approach is to use optimization techniques to explore the vast parameter space of the MCA, using objective *motion fidelity* indicators, so that the process can be automated. A genetic algorithm (GA) has been recently proposed to perform this process, with promising results. Following this approach, this paper proposes applying Particle Swarm Optimization (PSO) to solve the MCA tuning problem. The PSO-based proposed solution is assessed using the *classical washout* MCA, comparing its performance, convergence and correctness against the GA-based solution. Results show that a PSO-based tuning of MCA can provide better results and converges faster than a GA-based one. In addition, PSO is easier to set-up than GA, since only one parameter of the optimization algorithm itself (the number of particles) needs to be set-up, instead of a minimum of four in the case of the GA.

1. Introduction

The goal of any simulator is to provide its users with engaging sensations that make them forget about the reality they are living and truly believe they are experiencing an alternative/virtual reality, which is recreated by a computer by means of a synthetic, yet realistic, world. To accomplish this, the simulator needs to stimulate the perceptive skills of humans. In vehicle simulators, some of the most important perceptual cues are those related with the perception of motion. This kind of cue is usually generated by means of a robotic motion platform placed under the pilot/driver in the simulator set-up. These motion platforms have actuators that are able to move the user's seat, so that real motion sensations are experienced by pilots/drivers when they use the simulator. To control these devices, a specific kind of algorithm is designed and implemented. They are called, in the literature, Motion Cueing Algorithms (MCA), *washout algorithms* or even *washout filters*. They take the simulated physical state of the virtual vehicle (as input) and they provide the desired pose for the robotic mechanism (as output) in the form of translational and rotational degrees of freedom (DOF).

Fig. 1 shows the process diagram for the generation of motion cues in vehicle simulators with motion platforms. The core of the simulation system is the physics module. This module interacts with the pilot/driver (through the vehicle controls and displays) and calculates the physical state of the virtual vehicle (position, speed, acceleration, orientation, etc.). This physical state needs to be reproduced by the motion platform. The MCA is responsible for the calculation of the desired DOF for the robotic mechanism. Then, the desired DOF are first checked against the physical constraints of the motion platform and later transformed into actuators' positions by the Inverse Kinematics (IK) module. Then, the actuators are moved causing the simulator's seat to move, generating perceptual motion cues on the user. The actuators' control algorithm is placed in the lowest level of the process and is responsible for controlling individually each of the actuators of the robotic manipulator. This low-level control module is left aside of this study since we focus on the high-level control strategies (MCA) that deal with the much more challenging task of commanding jointly all the kinematic chains of the mechanism while enforcing the physical constraints of the simulator. There can be other modules in the

* Corresponding author.
E-mail address: Sergio.Casas@uv.es (S. Casas).

https://doi.org/10.1016/j.asoc.2018.03.044
1568-4946/

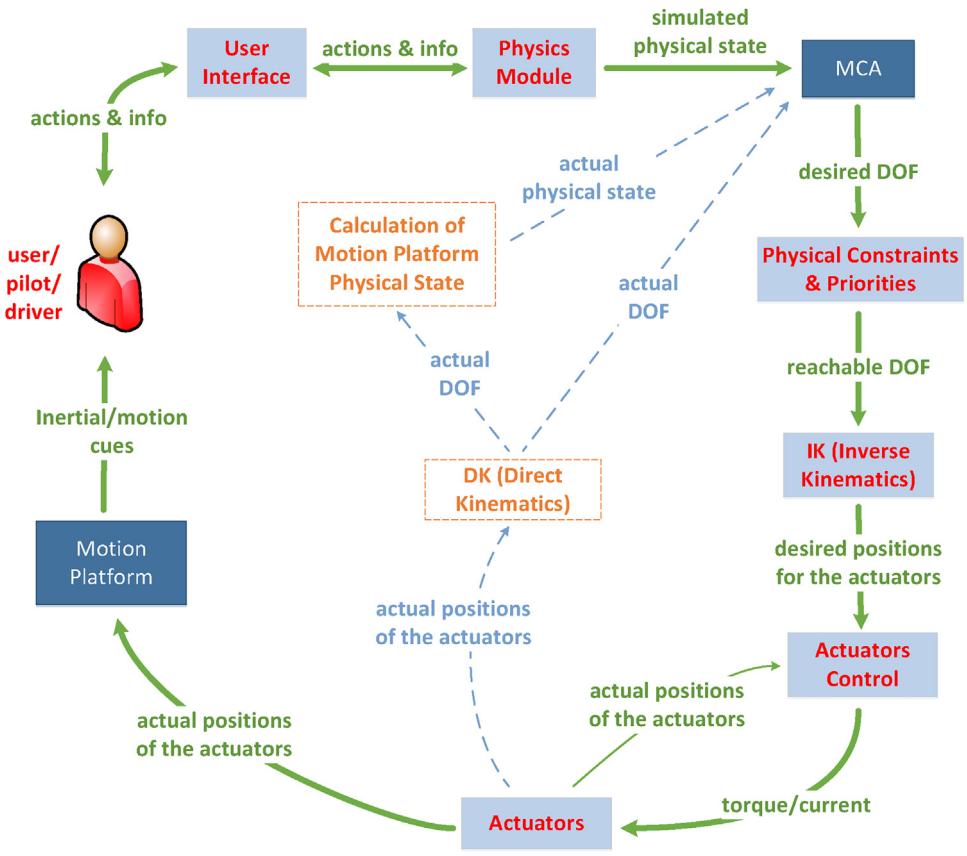

Fig. 1. Process diagram for the generation of motion cues in vehicle simulators.

process (highlighted with dashed lines in Fig. 1), although their use is optional.

One of the most important problems of this process is that MCA/washout algorithms are hard to evaluate and tune because there is not a universal criterion to evaluate this kind of algorithms. In addition, the high number of parameters that need to be tuned in most MCA make the tuning task very troubling and slow. The actuator's controllers (low-level control) are also hard to tune, but there are standardized mechanisms to assess this task. On the contrary, MCA are designed to satisfy pilots/drivers and, thus, their evaluation and tuning is not straightforward.

In this regard, the goal of this paper is to contribute to ease the MCA tuning process by proposing an automatic heuristic-based soft computing method that can improve or complement the existing methods. In particular, it will be proven that our solution outperforms the one recently proposed in [1].

The remainder of the paper is organized as follows. Section 2 reviews related and previous works on the topic. Section 3 describes the proposed solution. The computational experiments, their results and a discussion/interpretation of these results are described in Section 4. Finally, Section 5 summarizes the conclusions and outlines future research lines.

2. Related work

In this section, the literature related with the washout tuning problem is reviewed. First, we deal with the evaluation and tuning of MCA. Both problems are closely tied, since proper tuning is needed to perform meaningful evaluations and comparisons, and a reliable evaluation method is needed to achieve a systematic tuning method. Next, the few works dealing with the automatic tuning of MCA through optimization techniques are reviewed.

2.1. MCA: evaluation and tuning

As previously stated, the goal of an MCA is to generate the appropriate motion cues so that the pilot/driver of the simulator feels immersed in the virtual reality vehicle simulator. To achieve this, the MCA needs to provide, in real-time, the desired 6-component (surge, sway, heave, yaw, pitch, roll) pose for the motion platform. This 6-DOF pose represents the desired position and orientation of the robotic manipulator [2]. The inputs of washout algorithms may vary, but they typically take the linear acceleration (or specific force) and the angular speed of the simulated vehicle. These physical magnitudes are the ones that the human body is able to sense. Thus, the goal of an MCA is to generate outputs that match the expected perceived motion of the simulated vehicle, while keeping the physical constraints of the robotic manipulator, which has a limited motion envelope. These two goals are mutually opposed, so the simulated vehicle and the motion platform do not generally move in the exact same way. Thus, the MCA needs to find a trade-off between the fidelity of the reproduced motion and the spatial constraints of the robotic manipulator.

Several washout algorithms have been proposed in the literature [3–5]. Despite they present significant differences, they all share a series of basic ideas. The first principle is to reproduce a scaled version of the virtual vehicle motion, reducing its intensity by a fixed factor. The second idea is to high-pass filter the accelerations and angular velocities input to the MCA, to eliminate low-frequency movements that tend to create sustained displacements [6]. This reduces the fidelity of the reproduced motion but increases the probability of remaining within the spatial bounds of

the robotic manipulator. The last principle is to benefit from known perceptual illusions so that motion that is different to the expected one be perceived as similar. One example is the *somatogravic illusion* [7] by which linear motion can be confused with a controlled amount of tilt. This technique is known as *tilt coordination* [8] and can only be applied to linear frontal and lateral accelerations. More information about these techniques can be found in [9].

The evaluation of these algorithms is based on measures of *motion fidelity*. Motion fidelity [10] is defined as the ability of a simulator to generate the motion sensations that would be perceived in the real situation that it is being simulated. Different assessment methods for washout algorithms are used and mentioned, with different levels of detail, through vehicle simulation research. However, the literature dealing specifically with MCA evaluation is not very broad. Examples of research works dedicated to this topic are [11–15]. In summary, there are basically two ways of assessing the motion fidelity of a vehicle simulator: objective/quantitative evaluation and subjective/qualitative evaluation. In the former, certain magnitudes related with the execution of the MCA or with the performance of the pilot/driver are measured. In the subjective evaluation, pilots/drivers are prompted to rate the simulator or answer questionnaires about their impression of the generated motion, after expending some established time using the simulator. This is the most simple and natural way to assess the problem, because it can be considered a subjective problem. However, this evaluation system is slow, produces high inter-individual variations, it is non-systematic and non-repeatable. In addition, the evaluation of an MCA depends on how their parameters/coefficients have been previously tuned.

This leads us to the MCA tuning problem. Since the motion platform is almost always unable to replicate the motion of the simulated vehicle, the system needs to control what parts of motion are kept and what parts are eliminated. This is very often controlled by a series of parameters (or coefficients) of the MCA that need to be tuned before the algorithm is used to control a motion platform in the simulator. In this work, the authors do not focus in any particular MCA, so there is no point in explaining what the coefficients do for each washout algorithm. The meaning and effect of the parameters are different but the important message is that they substantially modify the behavior of the MCA, so it is crucial that a method to properly tune them be identified. In the case of the *classical washout* algorithm, which is the MCA used throughout this paper, the parameters and their meanings can be found in [16,17] and the number of parameters to be tuned can be 30 or more, although some of them are redundant or usually set to fixed values.

The tuning of MCA is often performed following a *pilot-in-the-loop* approach [5]. This method implies executing successive tests on the simulator with a series of pilots/drivers (or even only one) assessing the motion fidelity, and an expert changing (in real-time) the parameters of the MCA in reaction to the comments of the pilots/drivers. The process is repeated until the user is satisfied with the result. Grant [18] shows an example of this method, although using an expert system to remove some of the subjectibity of the process. This constitutes an improvement over a completely manual subjective process but still does not allow automatic tuning, as our work proposes.

2.2. MCA tuning with optimization algorithms

The subjective tuning, and the pilot-in-the-loop technique in particular, present some important disadvantages. The first problem is that pilots/drivers are human and they have difficulties in comparing different executions of the algorithms, especially if the two runs are distant in time. The second problem is that the expert subjectively decides what parameter(s) should be changed in response to the user's comments. This converts the process in a non-systematic procedure unless clear rules are defined, which is rarely done. Last, but certainly not least, one of the biggest problems is the lack of an objective criterion to stop the process. When should the process finish? The subjective evaluation makes sense, but the ending criterion should not be subjective too.

For these reasons, a systematic automatic solution is proposed in [1]. This proposal substitutes the subjective human evaluation by the calculation of objective motion fidelity indicators/metrics. These indicators are calculated upon the execution of a simulation of the robotic manipulator's movements. This way, it is not necessary to wear the expensive hardware of the motion platform performing hundreds of runs. The use of objective motion fidelity metrics opens also the possibility of using optimization algorithms to find the t-uple of parameters of the MCA that provides the best motion fidelity value. The details of the objective simulation-based assessment scheme can be found in [1].

Objective evaluation was not seriously considered a few years ago, but it is getting increasingly important in recent years, especially since the *Objective Motion Cueing Test* (OMCT) was established by the International Civil Aviation Organization (ICAO) [19] to certify flight simulators. OMCT is designed only for flight simulation and was not intended to be used to build automatic tuning methods upon it, since it does not provide a single-measure evaluation, although it is possible to perform an adaptation. In addition, OMCT focuses only on measuring magnitude attenuation (gain) and signal delay (phase shift) with sinusoidal signals. Therefore, it does not consider perceptual aspects over complex non-sinusoidal signals. In any case, several researchers have started to use this method to assess their simulators [20,21] because it represents a significant step forward in the evaluation of vehicle simulators. The application to other types of vehicles is also starting to be considered [22].

Not many researchers have proposed the use of objective evaluation systems in combination with optimization algorithms to solve the MCA tuning problem. The solution proposed by Thöndel [23] is one of the first works to reflect this idea. However, it is "partially automatic", and the details provided do not allow to reproduce the solution. Another tuning approach is presented in [24]. In this work, a subjective evaluation is performed, but a "fine tuning of the classical washout algorithm" is previously carried out using an Inertial Measurement Unit (IMU), a series of pre-recorded flights for a Cessna 172 flight simulator and a 3-DOF motion platform. However, the authors do not explain how this tuning is done, although the use of an IMU suggests the use of an automatic method. In [25] a rotorcraft simulator with a UH-60A Black Hawk helicopter is tuned with several methods, including an automatic one. The tuning methods were compared. Results show that their proposal for "quantitative tuning" (using an objective assessment method) does not provide good results, probably because their fitness function, as they acknowledge, is not adequate and still needs improvement. In addition, the paper does not explain the optimization method. A much more interesting and comprehensive approach is the one presented in [26]. In this case, OMCT is adapted to create an objective motion fitness function to tune MCA. The problem with this work is that it uses a kinematic approach by using a model of the motion platform that does not reflect its dynamics. Therefore, the assessment is incomplete since much of the false motion cues in vehicle simulators are caused by the inertial nature of the motion hardware that causes delayed motion. In addition, unlike our proposal, Roza does only tune a few parameters of the MCA, and the only reference to the optimization algorithm is that they use Matlab/Simulink optimization algorithms which "take about 12 h to find the optimal value on a regular desktop PC". In any case, heuristics strategies have been used in similar problems where the goal is to find the best parameters for the execution of some kind of

algorithm [27]. One key difference with the MCA tuning problem is that the evaluation function in the MCA tuning problem is very time consuming, as objective evaluations usually need to analyze motion signals of long duration (representative of the execution of the simulator during a certain amount of time).

The solution proposed in [1] represents a promising approach since it addresses many of the problems not dealt with in previously reviewed works, but it has a few disadvantages. First, the subjective evaluation has always been the traditional assessment method, and although the objective indicators are based on human perception [17], the proposed solution is still in a research phase, and it is only applicable to entertainment and non-regulated use, because the subjective assessment is preferred when the amount of time to perform the tuning task is high. Nevertheless, this objective approach is a useful tool when the available time is reduced and could also be used as a starting point for subjective fine tuning. The second problem, which is addressed in this paper, is that the proposed objective tuning method uses a genetic algorithm (GA) that needs to be tuned first. This optimization algorithm has four parameters to be set-up: population size, mutation probability, elitism rate and percentage of natural selection. As the optimization algorithm is used to tune and set-up an MCA, it would be preferable that the optimization algorithm itself be easier to set-up. In addition, the genetic algorithm was chosen because it was thought to be a good option for this optimization problem, since other search techniques such as greedy algorithms or exhaustive search are not applicable to this problem. Thus, it would be advisable to test if better optimization strategies can be found, a question that remains to be answered. This is also addressed in this paper.

GA is a heuristic optimization method [28]. It is well known that heuristic solutions are of potential benefit when the exact solution is not feasible or the analytical solution does not exist, when the optimal value for the solution is not really necessary and when the evaluation function is costly or unreliable [1,29]. All these circumstances apply here: (i) the exact solution cannot be expressed analytically and solved (it has many dimensions, and depends on many factors); (ii) the exact solution is not really necessary because the goal is to obtain satisfactory values for the parameters of an MCA that will never generate the perfect expected motion of the simulated vehicle; (iii) the evaluation function is not reliable because it is heuristic per-se and can be a matter of discussion.

However, there are several heuristics than can be applied to this problem other than a genetic algorithm, and in the few previous works using this approach, the problem of finding the best optimization heuristic for this application was not addressed. In fact, none of the previously cited works dive into the details and problems of the utilization of optimization algorithms for this particular use. In addition, unlike [1], we also use a 2-DOF motion platform to provide a broader performance comparison and study more in depth the suitability of soft computing techniques to the solution of this problem. For this reason, this paper proposes to use (and compare with the GA using several motion hardware configurations) a different heuristic that has also proven to be capable of solving optimization problems with vast and multi-dimensional search spaces: Particle Swarm Optimization (PSO) [30,31].

PSO has been successfully used for parameter tuning in other algorithms [32]. In [33] PSO is also used to adjust washout filters. This shows that the adaptation of this heuristic to this problem is feasible. Nevertheless, there are some important differences between [33] and this work. Unlike [1] and this proposal, PSO is used in that work to tune only the cut-off and damping of the high-pass and low-pass filters of the MCA. The tuning is performed in separate channels and only for the classical washout, whereas our proposal is to tune together all the parameters of any MCA, not just the classical MCA, albeit the classical washout is used for the assessment of the process. In addition, the evaluation function in [33] is very different: (i) it is heuristic and uses perception models of semicircular canals and otoliths that are both arguable (many different models and values for their parameters have been proposed [11,14,34]) and incomplete (not all the human motion perception comes from the inner ear sensors); (ii) it has several penalty weights (parameters) that would also need to be tuned. Moreover, the effectiveness of PSO against other optimization algorithms is not discussed, something that is addressed in this paper. In any case, [33] shows that PSO could be a very useful technique to improve motion cueing. In addition, PSO is easier to implement than evolutionary algorithms (EA), it tends to converge faster than EA [35] and it generally needs fewer parameters. These are the reasons why we choose this optimization solver to address and improve the MCA tuning problem.

3. Materials and methods

Particle Swarm Optimization can be applied and adapted to the MCA tuning problem. To do so, let us consider each t-uple t of parameters of the washout algorithm as a particle. The t-uple t represents a list of possible values for each of the parameters of the washout algorithm. For this research, the Reid-Nahon implementation of the classical washout algorithm was used [36] but other MCA can be used too. The number and meaning of the parameters can be changed but the method does not need to change.

Each particle can be represented by a sorted sequence of the MCA parameter values: $t[1], t[2], \ldots t[i], \ldots t[n]$. Each scalar in this vector represents a dimension in the parameter space. Thus, the value of the t-uple as a whole represents, in turn, the position of the particle in the n-dimensional parameter space in which the PSO algorithm will be searching. To perform the optimization process and identify the best set of values for the parameters of the MCA, an evaluation or fitness function $f(w, t)$ is needed, where w is the washout algorithm used, and t is the t-uple of parameters used to run the algorithm. Thus, the evaluation function returns a motion fidelity indicator given a washout algorithm, w, and a set of values for the parameters, t. Although the evaluation of the motion platform can be performed using objective indicators upon the actual hardware, the proposed method is to use a virtual motion platform instead of a physical one, as in [14,26,37] where models are used instead of the actual hardware. Thus, the motion fidelity indicators can be calculated faster since a virtual motion platform is able to simulate motion faster than real-time. In addition, this way, the hardware of the robotic device does not suffer unnecessary wear and potential human damage is avoided. The details of the motion platform simulator used throughout this work can be found in [38] and are not the focus of this paper. In fact, the method could operate using a different simulator or even upon the real hardware. In this latter case, however, it would be significantly slower.

The calculation of the motion fidelity indicators is performed by comparing the motion signals calculated by the physics module (this is the motion of the simulated vehicle and the expected motion of the robotic manipulator) with the motion actually generated on the pilot/driver by the motion platform. The comparison is performed on a signal-by-signal basis, comparing both the specific force and the angular velocity of the virtual vehicle with the motion actually generated on the user seat (this measures the physical or perceptual validity of the simulator [6], depending on how the indicators are designed). Six indicators are calculated for these six signals (three for the specific force and three for the angular velocity). These six indicators are later combined into a single objective number that represents the performance of the MCA for the given test, either additively or multiplicatively. The length of the signals depends on how long the evaluated sequence is. In driving simula-

tors, the usual procedure is to evaluate the washout algorithm for a circuit lap or a particular path. The evaluation process follows the scheme shown in [1].

The rationale and the design of these motion fidelity indicators is explained in [17] and are not included here for the sake of brevity. The indicators used in this work are: Normalized Average Absolute Difference (NAAD), Average Absolute Scale (AAS), Normalized Pearson Correlation (NPC), Estimated Delay (ED) and combinations of them. These indicators return a number greater or equal than unity, being this value the ideal one, so the goal of the optimization algorithm is to find the t-uple that is able to minimize the value of the objective motion fidelity indicator.

The PSO-based solution implements the optimization algorithm as described in [30]. It first evaluates each particle (t-uple) and then stores a series of values: (i) the position of the particle that provides the best global indicator; (ii) the indicator itself corresponding to this best global value; (iii) the best local indicator for each particle; (iv) the position of the best local indicator for each particle.

The most general expression of the canonical PSO algorithm, for each dimension d of the t-uple, is:

$$v_{id}^{k+1} = \omega^k v_{id}^k + c_1 U\left(0, \beta_1\right)\left(p_{id} - x_{id}^k\right) + c_2 U\left(0, \beta_2\right)\left(p_{gd} - x_{id}^k\right) \quad (1)$$

$$x_{id}^{k+1} = x_{id}^k + v_{id}^{k+1} \quad (2)$$

where x, v are n-component vectors representing the position and velocity of each particle respectively, i represents the index of the particle, k is the iteration (time) of the algorithm, p_i is the position of the local best, p_g is the position of the global best, $U(a, b)$ represents the uniform distribution between a and b, ω^k is an inertia weight function, whereas c_1,c_2, β_1, β_2 are constants. Different versions of PSO exist, depending on the terms and factors of Eq. (1) and in the neighborhood topology that defines how information is shared throughout the swarm [39]. The most simple version, which is implemented in this work, is [30]:

$$v_{id}^{k+1} = 2U(0, 1)\left(p_{id} - x_{id}^k\right) + 2U(0, 1)\left(p_{gd} - x_{id}^k\right) \quad (3)$$

$$x_{id}^{k+1} = x_{id}^k + v_{id}^{k+1} \quad (4)$$

In this form, the first term of the sum in (3), known as the *cognitive component*, represents the propensity of particles to return to their own previously found local best positions, whereas the last term, known as the *social component*, accounts for the desire of a particle to move towards the best position [31]. Regarding the neighborhood topology, *gbest* topology is used for the MCA-PSO algorithm, since the fitness function of this problem is very time consuming, and it is important that information is quickly shared by all the particles.

This way, the particle swarm "moves" towards the minimum and searches the optimal value for the MCA parameters. The full implementation of the PSO-based solution to solve the MCA tuning problem can be found in Appendix A. Fig. 2 depicts an example of how the MCA-PSO algorithm works, simplifying the problem to a 2-parameter space to visualize easily the operation of the algorithm. In each iteration, a particle with position t is updated and moved to another position t' depending on the position of the global best, the local best and two random coefficients.

Fig. 3 shows a block diagram of the workflow necessary to perform the whole process, highlighting the most important steps of the proposed tuning procedure. The main steps of the proposed tuning method are the following:

1) Choose an evaluation function for motion fidelity.
2) Run the vehicle simulator for a fixed time-span to obtain the physical state of the vehicle upon a particular task or use of the simulator (there is no need to enable the MCA in this step, although it could be used).
3) Record the signals representing the physical state of the simulated vehicle (typically the specific force and angular velocity vectors), which will be the inputs of the MCA.
4) Run the PSO optimization process. The optimization algorithm needs to run the vehicle simulator with an MCA (using the previous recorded inputs) varying the values of the MCA parameters. The fitness function is the evaluation function chosen in step 1 for motion fidelity and could be applied over a real motion platform or over a simulated version of the motion hardware. PSO loops to find the best parameters, until a termination condition is met. There may be different termination conditions: maximum time, maximum number of iterations, convergence threshold (in terms of average population fitness improvement, for instance), etc.

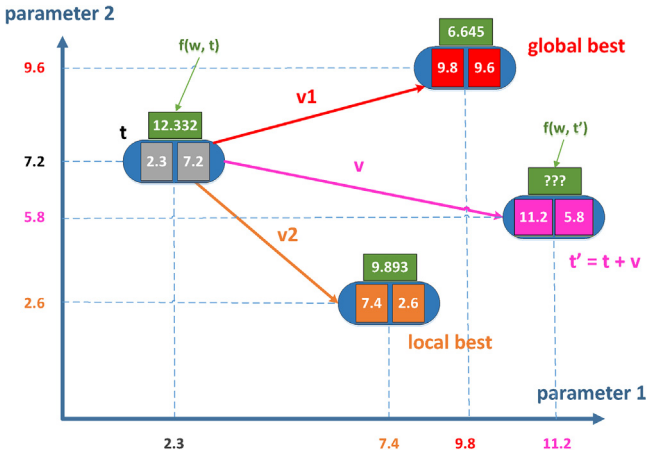

Fig. 2. Example of the MCA-PSO algorithm operation simplified to a 2-parameter space.

4. Experiments and results

In order to assess the proposed solution, a series of experiments and their results are presented in this section. The goal of these experiments is threefold: (i) to show the correctness of the solution; (ii) to analyze the effect of the PSO parameters on the final result; (iii) to compare the performance of the solution with other methods.

4.1. Assessment of correctness

Before analyzing and comparing the PSO-based solution performance, it is important to assess its correctness first, so that the algorithm can be confidently recognized as correct. As PSO is a heuristic approach, it is not possible to perform a formal demonstration to prove that the algorithm always provides the optimal value, because it does not. Instead, the algorithm provides satisfactory solutions that should be close to the (unknown) optimal value. Therefore, an empirical demonstration with a reduced problem is set-up to analyze if the PSO-based solution provides the expected output. For this experiment, a classical washout is used, setting all its inputs to zero except for the specific force in the vertical (Z) axis set to 9.8 m/s^2, and the specific force in longitudinal (Y) axis, where a 10-s sinusoidal signal, with an amplitude of 4 m/s^2 and a frequency of 1 Hz is used. This signal is used because it is the kind of test input used by OMCT and it can be representative of a typical acceleration-brake sequence when driving a vehicle.

To certify that the optimization algorithm works, only two parameters of the MCA are allowed to be varied and tuned: the

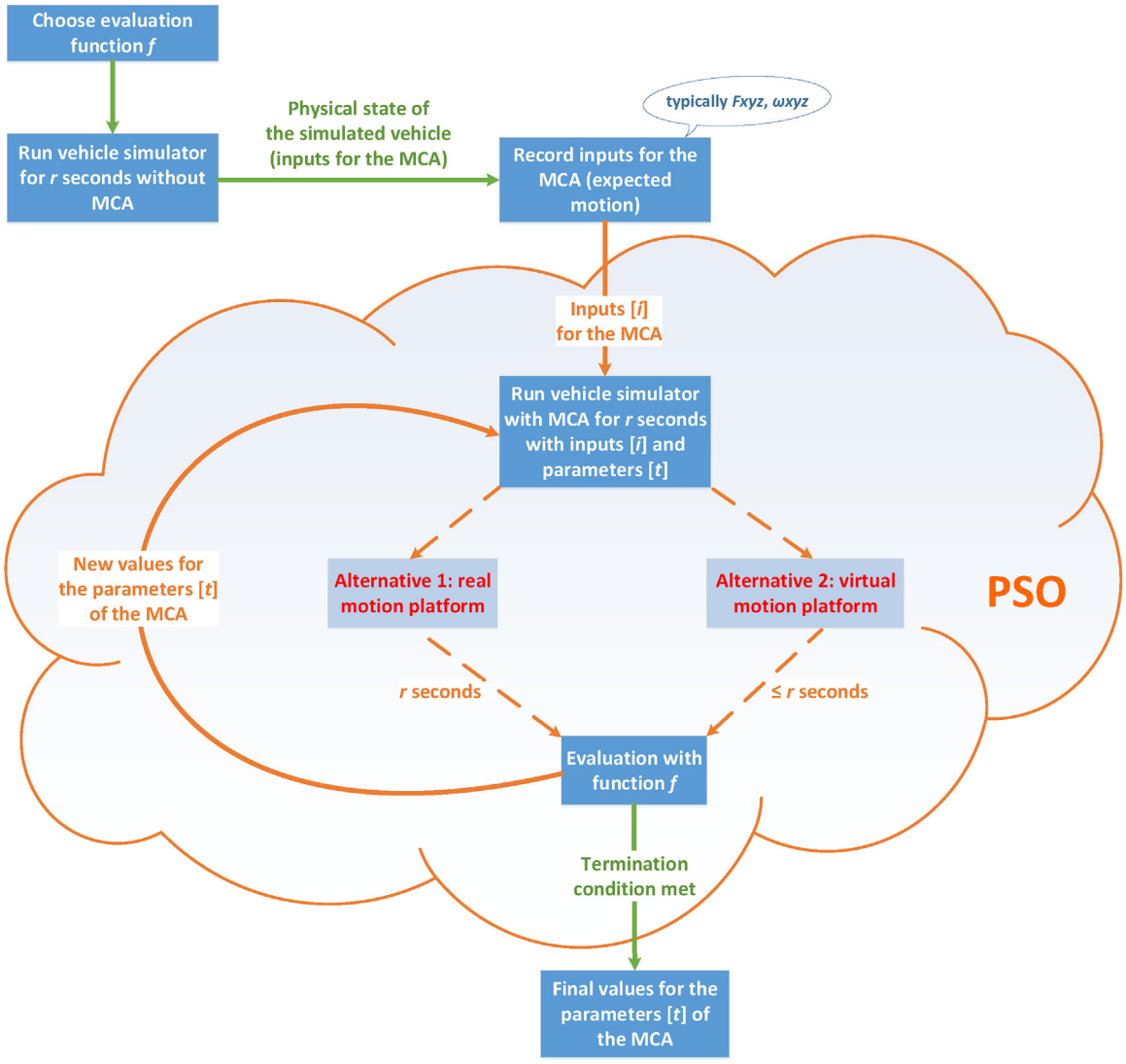

Fig. 3. Block diagram of the proposed MCA-PSO tuning method.

tilt limit of the Y coordination channel (between 0 and 20°) and the break frequency of the Y low-pass filter (between 0 and 10 Hz) [16]. The rest of the parameters of the MCA are set to 3 Hz in the case of break frequencies, 1 unit in the case of scale factors, and 10 units in the case of rate limiters. A simulated ideal 2-DOF motion platform (assuming instant motion and no delays) is used. The device used is a T0R2 (no translational DOF, two rotational DOF) motion platform with a (+38.60°, −20.65°) pitch range and a ±21.34° roll range, the architecture of which can be seen in [40]. The motion fidelity indicator used is NPC, calculated only for the specific force in the Y direction. With this simple setup and this reduced problem, the optimization algorithm should provide values near the upper limit of both intervals, since a higher break frequency and a higher tilt limit improves the correlation of the specific force with respect to the simulated motion in this particular set-up. Therefore, unlike in a general case, the expected solution can be considered known.

Tables 1 and 2 show the results using an Intel Pentium G840 at 2.80 GHz, with 8 GB of RAM and Windows 7 operating system. As it can be seen, both PSO and GA return the expected outputs so the implementation can be confidently considered correct. Moreover, PSO is able to identify the correct values faster than the GA, since it is able to perform just as well searching during 60 s as expending 600 s for the search task. Worst values found are also presented, although neither of these algorithms tries to find bad values.

Table 1
Assessment of the optimization algorithm – Search time = 60 s.

		Optimization Algorithm	
		GA	*PSO*
Best	*Cut-off (Hz)*	9.91327	10.0
	Tilt Limit (°)	18.35863	20.0
	NPC	1.24043	1.18267
Worst	*Cut-off (Hz)*	0.48854	0.44558
	Tilt Limit (°)	18.35863	19.65378
	NPC	15.77080	9.86634

It is necessary to emphasize the relevance of using a fixed time in the analysis of these algorithms for this particular problem. Although, in some specific cases the MCA may be tuned only once, in a general case it cannot be assumed that the MCA needs a single tuning. The simulator may be used for different tasks, simulated vehicles, tracks or even different configurations of the motion bases. Any change in the conditions under which the MCA is used may require a re-tuning. Since, in most of the applications, the amount of data necessary to perform the evaluation could be high, the time needed to perform an objective evaluation could also be high. As one of the goals of the process is to improve the slow manual tuning process that is typically used, time is a key factor and

Table 2
Assessment of the optimization algorithm – Search time = 600 s.

		Optimization Algorithm	
		GA	PSO
Best	Cut-off (Hz)	9.32855	9.87917
	Tilt Limit (°)	19.72081	19.97643
	NPC	1.18267	1.18267
Worst	Cut-off (Hz)	0.93369	1.40547
	Tilt Limit (°)	4.69742	6.71127
	NPC	25.14664	9.40380

Table 3
Analysis of population size for MCA-PSO. NPC values.

Population Size	Search Time (s)		
	100	500	1000
5	1.30851245	1.29147995	1.28477507
10	1.28989875	1.28746045	1.28487611
20	1.28895617	1.28590107	1.28335989
30	1.29392064	1.27895355	1.27585687
50	1.29476879	1.28829515	1.28525462
100	1.29694860	1.28939622	1.28539622
Max. Standard Deviation = 0.003913			

Table 4
2-DoF performance – Compared mean values.

Motion Fidelity Indicator	Optimization Algorithm		
	MC	GA	PSO
NPC	1.27782	1.26009	1.25537
NAAD	1.16899	1.16584	1.16593
AAS	18.14039	11.16844	10.31292
ED	1.19833	1.13667	1.13667
Combined	28.24064	18.81531	18.03238

the best way to compare the different optimization algorithms is to analyze the motion fidelity they can reach in a fixed time-span.

4.2. Tuning of the PSO parameters

As previously shown, the PSO-based solution is capable of finding the expected values for the MCA parameters/coefficients in a reduced and controlled set-up. However, the PSO algorithm itself needs to be set-up before using this optimization algorithm to look for satisfactory values for the coefficients of the MCA in a general case. In other words, the MCA tuner may need to be tuned first. Luckily, this PSO-based solution has only one parameter to be set-up: the number of particles of the swarm. Therefore, it is interesting to analyze if the value of this parameter has a significant influence in the results provided by the optimization algorithm.

For this reason, another experiment is prepared, varying the number of particles (the population size) in the PSO algorithm, while keeping the rest of the factors (motion platform, simulator, track, car, search time, motion fidelity indicator, etc.) fixed. The test is performed using a virtual 3-DOF motion platform, the same motion fidelity indicator (NPC), the same computer and MCA previously detailed, and 96-s input signals extracted from an open-wheel Formula 3 vehicle in one lap of the Monaco Grand Prix circuit using the rFactor [41] racing simulator. The 3-DOF motion platform is the one detailed in [17].

This time, the number of variable (tunable) parameters of the classical washout is set to 14, so the parameter space is much more complex than in the previous experiment, and the optimal values are completely unknown. Table 3 shows NPC values for different search times and different number of particles (population size), repeating the experiment 50 times to calculate average values. As the values are fairly similar, 8 decimal places are shown in the figures of Table 3.

Results suggest that a population size of 30 particles provides the best results when the available search time is high. When search time is reduced, population sizes between 10 and 20 particles are the preferred choice, because with bigger populations the number of evaluations of the MCA algorithm that are performed in one iteration of the PSO, increases, and the cost of each iteration increases too. Therefore, it is not possible to perform a significant number of iterations in such a reduced search time. On the contrary, if search time is higher, PSO can afford using larger populations since the number of iterations would be enough to converge to a satisfactory solution.

In any case, the differences are small and the PSO-based solution provides satisfactory results in almost all cases, so the tuning of the PSO algorithm should not be a problem, as the population size can be set to a fixed value of 20/30 particles. In this case, PSO can be treated almost as a parameter-less algorithm. This is an important improvement over other works, like [33], which requires several parameters and penalty weights that would also need to be tuned. Even, the solution shown in [1] needs more parameters than the one proposed here.

Table 5
2-DoF performance – Statistical tests (GA vs PSO).

Motion Fidelity Indicator	Standard Deviation GA	Standard Deviation PSO	F-test (Variances) P-value	T-test (Means) P-value
NPC	0.00489	0.00556	0.67256	0.00001
NAAD	0.00625	0.00695	0.64415	0.52718
AAS	0.13108	0.12956	0.48380	$< 10^{-5}$
ED	0.00487	0.00418	0.29745	0.49995
Combined	0.24761	0.25369	0.53366	$< 10^{-5}$

4.3. Compared performance

In order to draw conclusions about the performance of the PSO-based MCA tuning and analyze its behavior for different conditions, a comprehensive comparative test is presented. The test analyzes the performance of PSO with different motion fidelity indicators and several motion platforms, comparing the results with other optimization methods. For this experiment, the same computer, simulator, track, vehicle and MCA used in the previous test is employed. However, in this case, 18 parameters of the MCA are allowed to be varied/tuned, and the search time is set to 1000 s. The following motion fidelity indicators are used: NPC, NAAD, AAS, ED, and a multiplicative combination of AAS, NAAD and NPC. Three different motion platforms with 2, 3 and 6 DOF are used. The 2-DOF motion platform is the one used in Section 4.1, the 3-DOF motion platform is the one used in Section 4.2, while the 6-DOF motion platform capabilities can be seen in [17].

For this experiment, PSO is set-up with 30 particles, as suggested by the experiment in Section 4.2, and compared against the GA (implemented and set-up as in [1], where the GA parameters are tuned to offer the best possible performance) and a Monte-Carlo algorithm (MC), which has no parameters and operates generating random values on the parameter space of the MCA, keeping in memory only the values of the t-uple that induce the best motion fidelity indicator. The results of the experiment are depicted in Tables 4–9. Since the three optimization algorithms are probabilistic, 50 repetitions of each test are performed in order to calculate mean values and standard deviations. Tables 4, 6 and 8 show these mean values, which are also depicted in Fig. 4 for an easier interpretation. Results show that the PSO-based solution is the preferred choice in almost every case. Nevertheless, as the values obtained for the GA are in some cases only slightly worse (or even approx-

Table 6
3-DoF performance – Compared mean values.

Motion Fidelity Indicator	Optimization Algorithm		
	MC	GA	PSO
NPC	1.28565	1.26035	1.26180
NAAD	1.16963	1.16634	1.16648
AAS	14.95915	10.06724	9.26975
ED	1.19167	1.11833	1.11667
Combined	32.62618	18.39093	16.57539

Table 7
3-DoF performance – Statistical tests (GA vs PSO).

Motion Fidelity Indicator	Standard Deviation GA	Standard Deviation PSO	F-test (Variances) P-value	T-test (Means) P-value
NPC	0.00503	0.00548	0.61727	0.91416
NAAD	0.00695	0.00711	0.53159	0.53796
AAS	0.13045	0.12986	0.49370	$< 10^{-5}$
ED	0.00465	0.00440	0.42371	0.03433
Combined	0.24952	0.25273	0.51775	$< 10^{-5}$

Table 8
6-DoF performance – Compared mean values.

Motion Fidelity Indicator	Optimization Algorithm		
	MC	GA	PSO
NPC	1.29199	1.26586	1.26399
NAAD	1.17138	1.16872	1.16295
AAS	5.85252	4.69583	3.16952
ED	1.18167	1.11500	1.10000
Combined	9.63830	8.17163	5.84444

Table 9
6-DoF performance – Statistical tests (GA vs PSO).

Motion Fidelity Indicator	Standard Deviation GA	Standard Deviation PSO	F-test (Variances) P-value	T-test (Means) P-value
NPC	0.00501	0.00499	0.49444	0.03231
NAAD	0.00618	0.00694	0.65681	$< 10^{-5}$
AAS	0.13205	0.13068	0.48552	$< 10^{-5}$
ED	0.00496	0.00487	0.47458	$< 10^{-5}$
Combined	0.24931	0.25163	0.51287	$< 10^{-5}$

imately equal) than those of the PSO, *T*-tests were performed to statistically compare the results of GA and PSO (MC is clearly outperformed by these two). An *F*-test for variance equality between the GA and PSO populations is previously performed, in order to choose between an equal-variances *T*-test or an unequal-variances *T*-test. In all cases the null hypothesis of the *F*-tests (variances are different) is rejected, as shown by the p-values of the *F*-tests in Tables 5, 7 and 9. Therefore, we conclude that the variances are indeed equal and equal-variances *T*-tests are performed. The null hypothesis of these *T*-tests is that the indicators obtained with the GA are worse than those obtained with the PSO. This hypothesis was found true (using a 5% significance threshold for the p-value) for all the indicators when using the 6-DOF platform and for three of the five indicators when using the 2-DOF and 3-DOF devices. None of the experiments suggested that the opposite hypothesis (GA being better than PSO) could be accepted. Therefore, we can conclude that PSO represents an improvement over the GA-based tuning.

In addition, it is noticeable that the 2-DOF motion platform performs surprisingly well for the NPC and NAAD motion fidelity indicators (even better than the 3-DOF and the 6-DOF motion platforms). This is a consequence of two factors. First, the way these indicators are constructed: they reflect well similarity, penalize false cues, but do not penalize much the absence of motion. Second,

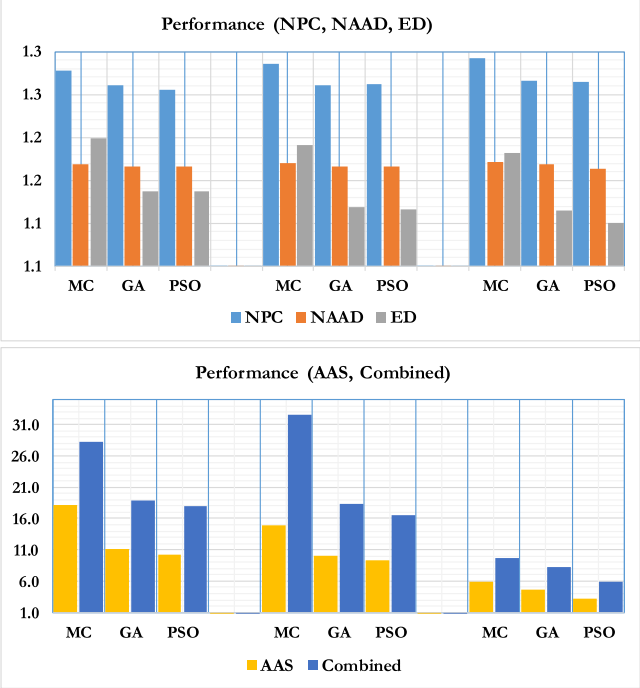

Fig. 4. Compared mean values for 2-DOF (left), 3-DOF (middle) and 6-DOF (right).

the design of this 2-DOF motion platform includes pitch and roll, which can be used to simulate forward and lateral linear accelerations using tilt coordination, reaching 4-DOF. In addition, the amount of pitch and roll motion available for this particular 2-DOF robotic manipulator is quite large (particularly the pitch range of motion).

AAS and ED indicators show a tendency to favor the 6-DOF motion platform against the smaller ones (especially in the case of AAS). This effect is confirmed in the combined indicator, which clearly shows that the 6-DOF motion platform is the preferred choice. When using this indicator, the PSO algorithm reveals itself as a much more efficient solution than the GA-based solution, something that could be said for some of the other motion fidelity indicators but gets clearer with this one. A possible explanation for this is that PSO has more memory than an EA, where only a few solutions (the good ones) are memorized. In PSO, knowledge about good solutions is retained by all particles. This is particularly important for a multi-dimensional problem with time-consuming evaluation functions, like the washout tuning problem. Although it is not possible to assert that PSO is better than any other algorithm for this problem, because the solution space of the motion fidelity function is unknown and could be changed, it can be said that PSO is particularly suited for this problem, since PSO is known for its ability to converge quickly to a reasonable acceptable solution with almost no need for meta-parameters, which is one of the objectives that we initially set.

It is also important to emphasize that neither the robotic motion platform nor the motion fidelity indicators seem to have influence in the performance of the PSO-based tuning algorithm, since the results are quite consistent for the different configurations tested.

4.4. Convergence

The last series of tests studies the convergence of the PSO algorithm comparing it with the GA. For this experiment, 10 parameters of the MCA are allowed to vary and be tuned. Using a multiplicative combination of NPC and AAS, the aforementioned 2-DOF motion platform, the same track, vehicle and simulator of previous tests,

Table 10
Convergence analysis – Mean values.

Algorithm	*Measure*	*Iterations*	*Time (s)*	*Indicator*
GA	*Mean*	7.20	159.13	1.2888906955
	Std. Dev.	4.61	101.26	0.0057859185
PSO	*Mean*	4.35	96.73	1.2877507686
	Std. Dev.	1.76	38.67	0.0049615949

Table 11
Convergence analysis – Statistical tests.

Measure	*Standard Deviation GA*	*Standard Deviation PSO*	*F-test (Variances) P-value*	*T-test (Means) P-value*
Iterations	4.61	1.76	$< 10^{-5}$	$< 10^{-5}$
Time (s)	101.26	38.67	$< 10^{-5}$	$< 10^{-5}$
Indicator	0.00579	0.00496	0.22229	0.0682

this experiment analyses the number of iterations, the required time and the indicator obtained when the improvement of the optimization algorithm reaches values lower than 10^{-3}. Thus, it studies the convergence of the two optimization algorithms. 100 repetitions of the test are performed in order to calculate average values. Tables 10 and 11 summarize the results.

Table 10 shows that the PSO algorithm converges much faster than the GA, needing fewer iterations (4.35 vs 7.20), and much less time (96.73 vs 159.13 s) to achieve the same results than the GA. Although the results of the comparison intuitively favor the PSO-based method, a statistical *T*-test for comparing the two time-measurement populations, shown in Table 11, is also performed to ensure statistical significance. As earlier, an *F*-test is performed to analyze the variances. In this case, the variances of the number of iterations and convergence times are different for the GA and PSO solutions. Therefore, unequal-variances *T*-tests are applied for those measures.

The *T*-test for the convergence times gives a t-value of 5.7569, with a p-value smaller than 10^{-5}. Thus, the mean time that is required for PSO to converge is indeed significantly lower than the time needed for GA to converge. A similar result is obtained when comparing the mean number of iterations (t-value 5.7756). The comparison for the resulting final indicator is much less meaningful since the amount of time to get the indicators is much lower in the case of the PSO because it converges faster. In any case, the p-value is very close to 0.05, although the test has to be rejected.

Moreover, PSO is more stable, since the values of standard deviation for both convergence time and number of iterations are significantly smaller for the PSO than those of the GA. This makes the PSO-based solution more reliable than the GA-based solution for this particular problem.

5. Conclusions and further work

As shown, the MCA tuning problem can be addressed and solved, in certain circumstances, by using optimization algorithms and heuristic techniques in combination with objective assessment methods, instead of the traditional subjective pilot-in-the-loop solution. The major benefit of this approach is that it is possible to provide an automatic and unattended method to obtain the best values for the parameters of an MCA. In previous works, it was shown that a genetic algorithm could provide satisfactory values for the solution of the MCA tuning problem. However, the genetic algorithm itself has several parameters that should be tuned first in order for the GA to be used to tune a motion cueing algorithm. This, somehow, breaks the premise to achieve an unattended solution. This paper deals with adapting the PSO optimization technique in order to provide an (almost) unattended solution to the MCA tuning problem.

A series of experiments have been performed in order to assess the correctness, performance and convergence of the proposed MCA-PSO tuning method. These experiments allow us to draw conclusions about the suitability of the proposed solution. First, we conclude that PSO provides better results than the genetic algorithm for this problem. It is true that the amount of improvement in terms of average indicators is relatively small, but the statistical tests show that this method performs better than previous ones. In addition, PSO is very easy to tune, as only one parameter (the number of particles of the particle swarm simulation) needs to be set. Moreover, this approach provides several other key advantages. First, PSO converges faster than the genetic algorithm for this problem, so it can provide a satisfactory solution much earlier. This result can be extremely important if the motion platform and the MCA are used for entertainment or unregulated training, where the MCA parameters may have to be adjusted or even retuned if a different track, vehicle or even simulator is used. Second, PSO provides satisfactory solutions regardless of the value of the number of particles used, and this parameter could even be set to a fixed value (a value between 20 and 30 seems the most appropriate one for the vehicle simulator used in the experiments). Last, but not least, the results provided by PSO are more consistent and predictable because the standard deviation of the convergence time is smaller than in the case of the GA. Therefore, it can be stated that PSO can fully substitute GA for a simulation-based automatic tuning of MCA, improving its performance and providing additional advantages. Another conclusion that can be drawn from the performance experiments is that 3-DOF and 2-DOF (especially this one) motion platforms perform surprisingly well for some of the indicators employed to measure motion fidelity. This is consistent with the results obtained by [42].

A number of improvements and future research lines can be derived from this study. Future work includes looking for different optimization strategies that may provide a good balance between exploration and exploitation for this problem, such as black-box and expensive black-box optimization techniques, different versions/topologies of PSO, or even new optimization schemes designed specifically for this problem. An in-depth study of the advantages of 6-DOF devices with respect to limited-DOF devices would be interesting as well, since it seems that the improvements are small. It is also possible to compare the performance of PSO for different simulators (including different vehicles), washout algorithms or even new objective indicators. In this regard, it would also be interesting to perform further tests to study the acceptance of pilots/users of this tuning method, although similar experiments have been performed in the past with the GA, and PSO works under the same principles.

Appendix A. MCA-PSO Algorithm

Inputs:
w : W // washout algorithm
n : N // number of parameters of w that can be modified
f : W, Rⁿ -------> R // evaluation/fitness function
li : vector [1..n] of R *// lower limits of the parameter ranges*
ls : vector [1..n] of R *// upper limits of the parameter ranges*

Parameters:
numParticles, maxIters : N
maxTime: R

Auxiliar:
particles: list of {*params: vector [1..n] of* R, *eval:* R}
localBests: list of {*params: vector [1..n] of* R, *eval:* R}
i, j, k : N
time, speed, localSpeed, globalSpeed: R

Outputs:
p : vector [1..n] of R *// optimum t-uple*
o : R *// fitness function value for p*

Algorithm:

```
i = 0;
o = ∞;
time = 0.0;
startClock();

for (j=1 to numParticles) do
   localBests[j].eval = ∞;

particles = generateRandomParticles(numParticles, li, ls);

do
{
   i = i + 1;

   // evaluate particles
   for (j=1 to numParticles) do
   {
      particles[j].eval = f(w, particles[j].params);

      if (particles[j].eval < localBests[j].eval) then
      {
         localBests[j].eval = particles[j].eval;
         localBests[j].params = particles[j].params;
      }

      if (particles[j].eval < o) then
      {
         o = particles[j].eval;
         p = particles[j].params;
      }
   }

   // update particles (speed and position)
   for (j=1 to numParticles) do
   {
      k = 0;
      foreach (x1 in particles[j].params, x2 in localBests[j].params, x3 in p) do
      {
         localSpeed =  x2 – x1;
         globalSpeed = x3 – x1;
         speed = 2.0*rand(0.0, 1.0)*localSpeed + 2.0*rand(0.0, 1.0)*globalSpeed;

         x1 = x1 + speed;
         clamp(x1, li[k], ls[k]);
         k++;
      }
   }

   time = getCurrentClockTime();
}
while ( (i < maxIters) ^ (time < maxTime) )
```

References

[1] S. Casas, I. Coma, C. Portalés, M. Fernández, Towards a simulation-based tuning of motion cueing algorithms, Simul. Modell. Pract. Theory 67 (2016) 137–154.
[2] J.M. Van Verth, L.M. Bishop, Essential Mathematics for Games and Interactive Applications: A Programmer's Guide, Morgan Kaufmann, Burlington, MA, USA, 2004.
[3] N.J.I. Garrett, M.C. Best, Driving simulator motion cueing algorithms—a survey of the state of the art, in: Proceedings of the 10th International Symposium on Advanced Vehicle Control (AVEC), Loughborough, UK, 2010, pp. 183–188.
[4] K. Stahl, G. Abdulsamad, K. Leimbach, Y.A. Vershinin, State of the art and simulation of motion cueing algorithms for a six degree of freedom driving simulator, in: 17th International Conference on Intelligent Transportation Systems (ITSC), IEEE, Qingdao, China, 2014, pp. 537–541.
[5] S. Casas, R. Olanda, N. Dey, Motion cueing algorithms: a review—algorithms, evaluation and tuning, Int. J. Virt. Augmented Real. 1 (2017) 90–106.
[6] G. Reymond, A. Kemeny, Motion cueing in the renault driving simulator, Veh. Syst. Dyn.: Int. J. Veh. Mech. Mobil. 34 (2000) 249–259.
[7] P.R. MacNeilage, M.S. Banks, D.R. Berger, H.H. Bulthoff, A Bayesian model of the disambiguation of gravitoinertial force by visual cues, Exp. Brain Res. 179 (2007) 263–290.
[8] E.L. Groen, W. Bles, How to use body tilt for the simulation of linear self motion, J. Vestib. Res. 14 (2004) 375–385.
[9] M.A. Nahon, L.D. Reid, Simulator motion-drive algorithms—a designer's perspective, J. Guid. Control Dyn. 13 (1990) 356–362.
[10] J.B. Sinacori, The Determination of Some Requirements for a Helicopter Flight Research Simulation Facility, 1977, Moffet Field, CA USA.
[11] M. Wentink, J. Bos, E. Groen, R. Hosman, Develpment of the motion perception toolbox, in: AIAA Modeling and Simulation Technologies Conference and Exhibit, Keystone, CO, USA, 2006.
[12] S. Advani, R. Hosman, M. Potter, Objective motion fidelity qualification in flight training simulators, in: AIAA (Ed.), AIAA Modeling and Simulation Technologies Conference and Exhibit, Hilton Head, SC, USA, 2007.
[13] R. Hosman, S. Advani, Are criteria for motion cueing and time delays possible? Part 2, in: AIAA (Ed.), AIAA Modeling and Simulation Technologies Conference, Boston, MA, USA, 2013.
[14] C. Onur, U. Türe, U. Zengin, Pilot perception and control behavior models as a tool to assess motion-Cueing algorithms, AIAA Modeling and Simulation Technologies Conference (2017), pp. 3475.
[15] D. Cleij, J. Venrooij, P. Pretto, D.M. Pool, M. Mulder, H.H. Bülthoff, Continuous subjective rating of perceived motion incongruence during driving simulation, IEEE Trans. Hum.-Mach. Syst. 48 (1) (2018) 17–29.
[16] L.D. Reid, M.A. Nahon, Flight Simulation Motion-Base Drive Algorithms: Part 2—Selecting the System Parameters, UTIAS, University of Toronto, 1986.
[17] S. Casas, I. Coma, J.V. Riera, M. Fernández, Motion-cuing algorithms characterization of users' perception, Hum. Fact.: J. Hum. Factors Ergon. Soc. 57 (2015) 144–162.
[18] P.R. Grant, L.D. Reid, PROTEST. An expert system for tuning simulator washout filters, J. Aircraft 34 (1997) 152–159.
[19] ICAO, Manual of Criteria for the Qualification of Flight Simulation Training Devices. Volume I—Aeroplanes, International Civil Aviation Organization, 2009.
[20] O. Stroosma, M.M. Van Paassen, M. Mulder, R. Hosman, S. Advani, Applying the objective motion cueing test to a classical washout algorithm, in: AIAA Modeling and Simulation Technologies (MST) Conference, Boston, MA, USA, 2013.
[21] P.M.T. Zaal, J.A. Schroeder, W.W.Y. Chung, Objective motion cueing criteria investigation based on three flight tasks, in: Challenges in Flight Simulation, Royal Aeronautical Society, London, UK, 2015.
[22] M. Fischer, A. Seefried, C. Seehof, Objective motion cueing test for driving simulators, Proceedings of the DSC 2016 Europe (2016) 41–50.
[23] E. Thöndel, Design and optimisation of a motion cueing algorithm for a truck simulator, in: European Simulation and Modelling Conference—ESM, Essen, Germany, 2012, pp. 165–170.
[24] J. Le Bouthillier, Y. Liang, P. Allard, Pilot evaluation of a low cost 3 degree-of-freedom flight simulator driven by the classical washout filter algorithm, Online J. Comput. Sci. Technol. 2 (2012) 102–106.
[25] S. Reardon, S. Beard, Evaluation of motion tuning methods on the vertical motion simulator, in: 71st American Helicopter Society Annual Forum, 2015, pp. 5–7.
[26] M. Roza, R. Meiland, J. Field, Experiences and perspectives in using OMCT for testing and optimizing motion drive algorithms, AIAA Modeling and Simulation Technologies (MST) Conference (2013).
[27] H. Cheng-Lung, W. Chieh-Jen, A GA-based feature selection and parameters optimization for support vector machines, Expert Syst. Appl. 31 (2006) 231–240.
[28] J.H. Holland, Genetic algorithms, Sci. Am. 267 (1992) 66–72.
[29] A. Díaz, F., Glover, H.M., Ghaziri, J.L. González, M., Laguna, P., Moscato, F.T. Tseng, Optimización Heurística y Redes Neuronales, Paraninfo, Madrid, 1996.
[30] J. Kennedy, R. Eberhart, Particle swarm optimization, in: IEEE International Conference on Neural Networks, IEEE Service Center, Piscataway, NJ, USA, Perth, WA, Australia, 1995, pp. 1942–1948.
[31] F. Marini, B. Walczak, Particle swarm optimization (PSO). A tutorial, Chemom. Intell. Lab. Syst. 149 (2015) 153–165.

[32] L. Shih-Wei, Y. Kuo-Ching, C. Shih-Chieh, L. Zne-Jung, Particle swarm optimization for parameter determination and feature selection of support vector machines, Expert Syst. Appl. 35 (2008) 1817–1824.
[33] H. Asadi, A. Mohammadi, S. Mohamed, C.P. Lim, A. Khatami, A. Khosravi, S. Nahavandi, A particle swarm optimization-based washout filter for improving simulator motion fidelity, in: Systems, Man, and Cybernetics (SMC), 2016 IEEE International Conference on, IEEE, 2016, pp. 001963–001968.
[34] M.C. Newman, B.D. Lawson, A.H. Rupert, B.J. McGrath, A.M. Hayes, L.S. Milam, in: USAARL (Ed.), A Model of Human Orientation and Self-Motion Perception During Body Acceleration: The Orientation Modeling System, U.S. Army Aeromedical Research Laboratory, 2016.
[35] Y.-J. Gong, J.-J. Li, Y. Zhou, Y. Li, H.S.-H. Chung, Y.-H. Shi, J. Zhang, Genetic learning particle swarm optimization, IEEE Trans. Cybernet. 46 (2016) 2277–2290.
[36] L.D. Reid, M.A. Nahon, Flight Simulation Motion-Base Drive Algorithms: Part 1—Developing and Testing the Equations, UTIAS, University of Toronto, 1985.
[37] K. De Ridder, M. Roza, Automatic optimization of motion drive algorithms using OMCT, AIAA Modeling and Simulation Technologies Conference (2015) 1139.
[38] S. Casas, J.M. Alcaraz, R. Olanda, I. Coma, M. Fernández, Towards an extensible simulator of real motion platforms, Simul. Modell. Pract. Theory 45 (2014) 50–61.
[39] K.-L. Du, M.N.S. Swamy, Particle swarm optimization, in: Search and Optimization by Metaheuristics, Springer, 2016, pp. 153–173.
[40] S. Casas, C. Portalés, S. Rueda, M. Fernández, On the simulation of robotic motion platforms with digital filters for virtual reality applications, Rev. Iberoamericana Autom. Inf. Ind. RIAI 14 (2017) 455–466.
[41] rFactor, http://www.rfactor.net/, in: rFactor Racing Simulation, 2018.
[42] N.A. Pouliot, C.M. Gosselin, M.A. Nahon, Motion simulation capabilities of three-degree-of-freedom flight simulators, J. Aircraft 35 (1998) 9–17.