Differential Evolution Algorithms Comparison Used to Tune a Visual Control Law

PABLO SÁNCHEZ-SÁNCHEZ1, JOSE GUILLERMO CEBADA-REYES2, (Member, IEEE), AGUSTÍN RUIZ-GARCÍA3, AIDEÉ MONTIEL-MARTÍNEZ4, AND FERNANDO REYES-CORTÉS1

1Facultad de Ciencias de la Electrónica, BUAP, Puebla 72530, Mexico
2Tecnológicos Nacionales de México/ITES de Teziutlán, Teziutlán 73960, Mexico
3Departamento Ingeniería Agrícola, UACH, Texcoco 56230, Mexico

Corresponding author: Jose Guillermo Cebada-Reyes (jose.cr@teziutlan.tecnm.mx)

This work was supported in part by the Departamento de Posgrado en Sistemas Computacionales, Instituto Tecnológico Superior de Teziutlán, through the Consejo Nacional de Ciencia y Tecnología (CONACYT) under Grant Estancias Posdoctorales por México: Modalidad 1.

ABSTRACT This paper presents the results of the comparative study of differential evolution algorithms used as an alternative for tuning the gains of a PD+ type visual control structure, since tuning PD or PID type control structures is not trivial. We made the quantitative comparison using the performance index ($L_2$-norm) and we applied the algorithm of best performance in a case of study (detection and cutting of leaves using a Cartesian robot). We also present the stability analysis according to Lyapunov theory and experimental results.

INDEX TERMS Differential evolution algorithms, PD+ type control structure, performance index, tuning control structures, centroid, visual-servoing, plant tissue analysis, leaves cutting.

I. INTRODUCTION

Tuning control structures like PD or PID are not trivial, and although there are several studies to tune these controllers, we have not been solved the problem in order to have a technique that we can implement in various systems, for example, SISO systems are limited to Classical tuners because of its linear nature, while computational resources limit intelligent methods such as neural networks, [1]. The PID controller is not the most suitable for certain non-linear systems. However, it is the preferred control law by the industry for its ease of implementation [2].

We must keep in mind that, for a controller, it is not enough to maintain the desired response, since it is also necessary that converge to zero in a finite time, so by changing gains, we can change the system’s response. With a set of gains, we get unique features, so the goal is to look for the set of gains that produce the desired output defined by the design requirements [1]. However, adjusting gains is a complicated, and to some extent empirical, procedure that requires much of the designer’s experience.

Although we widely used the PID controllers in industrial processes, their effectiveness is often limited because of bad tuning. Manual tuning of a controller’s gains is a time-consuming task [3].

It has made many proposals to tune PID control over the years, for example, Chang, in [4], proposed a modification of the recombination formula of a genetic algorithm, and this method was used to determine the gains of PID controllers in multivariable processes, where it is ensured that the genetic algorithm is a convincing method of searching optimal for solving control problems. Sung-Kwun, in [5], proposed fuzzy controllers to stabilize the Beam-Ball system, compared with PD controllers, and he tuned the parameters of these controllers using a genetic algorithm based on hierarchical fair competition (HFCGA). Helon, in [6], tuned a PID controller using a multi-target genetic algorithm for a manipulator robot. In [7], it is presented a novel adaptive neural-based control design for a robot with incomplete dynamical model and facing disturbances based on a simple structured PID-like control. Radial basis function neural networks are used to estimate uncertainties, and to determine PID gains through a direct Lyapunov method. The paper provides proof that all signals in the closed-loop system are bounded while it does not violate the constraints.

In [8], it is presented a novel method called self-adaptive proportional integral derivative (PID) of radial basis function neural network (RBFNN-PID), this method shown
an improvement longitudinal speed tracking accuracy for autonomous vehicles in comparison with the traditional PID and fuzzy control methods, as we can see in the simulation results.

Lee et al. [9], have proposed an adaptive PID control that is simple, model-free, and robust against payload variations. The proposed PID control has inherited these advantages, already verified by through the equivalence relationship between the two controls. As a result, the proposed PID shares the simplicity, robustness, and the model free property, as well as the high levels of stability, and performance with the adaptive TDC. In particular, the selection of its gains becomes especially simple, while the adaptation becomes efficient under substantial payload variations.

In [10], an adaptive PID control for wind turbines was developed, which allows an automatic change of gains with no trial-and-error processes. In [11], the principal aim was to decide the gains of the torque-based PID controller with the help of a neural network trained by using nature-inspired optimization algorithms. The adaptiveness of the algorithm lies in modifying the gains of the controller based on the magnitude of the error in the angular displacement received at the input.

Because of the increasing complexity of optimization problems, distributed differential evolution (DDE) has become a promising approach for global optimization. However, similar to the centralized algorithms, DDE also faces the difficulty of strategies’ selection and parameters’ setting. In order to deal with such problems effectively, in [12], it is proposed an adaptive DDE (ADDE) to relieve the sensitivity of strategies and parameters. On the other hand, in [13], it is proposed an adaptive variant of the differential evolution algorithm DE using both historical experience, and heuristic information for adaptation. This helps the algorithm select a more appropriate mutation strategy and determine better parameters for each individual at different evolution stages.

As with PID control, it has developed many proposals to tune the gains of PD control, for example, in [14], it is proposed a PD+ controllers to be applied in regulation tasks, and tracking of trajectories, in this work it is developed a proportional control of the position error, velocity, and the complete dynamics of the manipulator robot, which provides robustness. In [15] and [16], a strict Lyapunov function is proposed in order to perform the overall asymptotic stability analysis of the PD+ control. However, the problem of model-based controllers is of practical origin, as they require accurate numerical knowledge of dynamics and system parameters. Therefore, differential evolution algorithms provide an alternative for solving this type of problem.

We should note that differential evolution algorithms are direct and simple population search algorithms, which allow us to find the overall optimal value of multimodal functions, not differential and non-linear. There are several differential evolution algorithms, those in which their parameters, population sizes, probability of crossing, and mutation are constant, and those whose parameters are adaptive (SaDE, JDE and JADE), [17]–[19].

We can see, there are many proposals for tuning control structures such as PD or PID, so we presents a comparative study, through the performance index, of differential evolution algorithms as an option to adjust the gains of the PD+ type control. The aim of this article is to compare the most used differential evolution algorithms in the agricultural engineering and to use the algorithm with the best performance in the detection and cutting of leaves using a Cartesian robot.

We organized the article as follows: in Section II, we defined the dynamic model of the robot, as well as some properties and characteristics of the vision system and the PD+ type control, we also define the main characteristics of the differential evolution algorithms, which allows us to make the comparison of the algorithms. In section III, we define the case of study, and we show the experimental results, and finally, we present the conclusions in section IV.

II. MODELING
A. DYNAMIC MODEL
We define the dynamic model of a robot manipulator with n degrees of freedom (DOF) as

\[ H(q) \ddot{q} + C(q, \dot{q}) \dot{q} + g(q) = \tau \]  

(1)

where \( q \in \mathbb{R}^n \) is the generalized coordinate vector; \( H(q) \in \mathbb{R}^{n \times n} \) is the mass and inertia matrix, which is symmetric and positive definite; \( C(q, \dot{q}) \dot{q} \) is the Coriolis vector and centripetal force; \( g(q) \in \mathbb{R}^n \) is the gravitational torque; and \( \tau \in \mathbb{R}^n \) is the applied torque. We based the dynamic model on Euler-Lagrange equation of motion defined as

\[ \frac{d}{dt} \left( \frac{\partial L(q, \dot{q})}{\partial \dot{q}} \right) - \frac{\partial L(q, \dot{q})}{\partial q} + \frac{\partial D(q)}{\partial \dot{q}} = \tau, \]  

(2)

where \( L(q, \dot{q}) \) is the Lagrangian defined as the difference between the kinetic energy \( K(q, \dot{q}) \), and the potential energy \( U(q) \),

\[ L(q, \dot{q}) = K(q, \dot{q}) - U(q) \]  

(3)

and \( D(q) \) is the dissipation energy.

For simplicity, we consider that all robots have rotational joints, which allow us to state the following properties:

Property I (20, Sect. 2, pp. 474): \( H(q) \) satisfies

\[ \lambda_h \| x \|^2 \leq \mathbf{x}^T H(q) \mathbf{x} \leq \lambda_H \| x \|^2 \quad \forall \mathbf{x} \in \mathbb{R}^n \]  

(4)

where

\[ \lambda_h \triangleq \min_{q \in \mathbb{R}^n} \lambda_{\min} \{ H(q) \} \]  

(5)

\[ \lambda_H \triangleq \max_{q \in \mathbb{R}^n} \lambda_{\max} \{ H(q) \} \]  

(6)

and \( 0 < \lambda_h \leq \lambda_H < \infty \).
Property 2 (20, Sect. 2, pp. 474): With the appropriate definition of \( C(q, \dot{q}) \), the matrix
\[
H(q) - 2C(q, \dot{q})
\]
is antisymmetric.

The robot manipulator’s end-effector interacts with a rigid environment, so it restricted its movement to a \((n - m)\) dimensional space \( \varphi \) defined by \( \varphi(q) = 0 \) where the function \( \varphi : \mathbb{R}^n \rightarrow \mathbb{R}^m \) is continuously differentiable, such that \( \varphi \in C^2 \). Due to the existence of the function \( \varphi \), we assume that there is an operating region \( \Omega \subseteq \mathbb{R}^n \) defined as \( \Omega = \Omega_1 \times \Omega_2 \), where \( \Omega_1 \) is a convex subset of \( \mathbb{R}^{n-m} \), \( \Omega_2 \) is an open subset of \( \mathbb{R}^m \). Because of the operating region \( \Omega \subseteq \mathbb{R}^n \), we assume the existence of a function \( k_1 : \Omega_1 \rightarrow \mathbb{R}^m \) that is continuously differentiable such that \( k_1 \in C^2 \), where \( \varphi(q_1, k(q_1)) = 0 \ \forall q^1 \in \Omega_1 \). Under these assumptions, \( q^2 \) can uniquely define \( q^1 \) so that \( q^2 = k(q^1) \ \forall q^1 \in \Omega_1 \). Under this assumption, the Jacobian matrix \( J(q) \) exists and is nonsingular \( \forall q \in \Omega_1 \), [21].

**B. VISION MODEL**

The goal of a vision system is to create a real-world model from images and using its two-dimensional projections to retrieve useful information from the scene.

We must keep in mind that this information recovery requires a lot of data from a plane. To accomplish this, we need to consider a reference system in the Cartesian space \( \mathbb{R}^n \) that is continuously differentiable such that \( k_1 \in C^2 \), whose origin is at the intersection formed by the optical and the center of the lens \( \Sigma_C \), [22], [23]. We express the coordinates of a point regarding this reference system as \( x_C \). The vector \( \alpha_C = [\alpha_{c1}, \alpha_{c2}, \alpha_{c3}]^T \) represents the relative location between the robot reference systems \( \Sigma_R \) and the camera \( \Sigma_C \). The target has a point in the Cartesian system \( \Sigma_T = \{i_1, i_2, i_3\} \), whose origin is located regarding its geometric center. We denote by \( \sigma_T = [\sigma_{t1}, \sigma_{t2}, \sigma_{t3}]^T \) the position of the object reference system regarding \( \Sigma_R \), [24]. We project the scene got by the camera in the CCD, which is associated with a reference system described by \( \Sigma_1 = \{i_1, i_2\} \), whose origin is in the geometric center of the CCD camera. The \( i_1 \) and \( i_2 \) axes are parallel and point in the same direction as the \( c_1 \) and \( c_2 \) axes, respectively. To get the coordinates of the image in the CCD plane, it is necessary to get a perspective transformation.

We digitize the scene image in the CCD and transfer it to the computer screen, and we define a new two-dimensional reference system \( \Sigma_D = \{u, v\} \), the origin of which is in the upper left corner of the monitor, [24]. We know the complete vision system that we use as a fixed camera configuration. This system expresses the image coordinates in pixels.

We describe the model of the vision system by
\[
\begin{bmatrix}
  u \\
  v \\
  x_c1 \\
  x_c2 \\
  x_c3
\end{bmatrix} = R^T(\theta)\begin{bmatrix}
  x_R - \theta_R^T
\end{bmatrix}
\]
where \( \alpha_u > 0 \), \( \alpha_v > 0 \) are the scale factors in [pixel/meter], \( \lambda > 0 \) is the camera focal length, and \( \frac{\lambda}{\lambda + x_c3} \), [24].

**C. CONTROL LAW**

The control structure used to track the desired trajectory is
\[
\tau_{ipD+} = K_p \psi(\tilde{q}) + K_v \psi(\tilde{\dot{q}}) + \tau
\]
where \( \psi(\cdot) = \text{sinh}(\cdot)\sqrt{1 - \text{tanh}^2(\cdot)}, \cdot \in \mathbb{R}^n \) represents the error of \( \cdot \), \( \tau \in \mathbb{R}^n \) is the dynamic model of the system defined in (1), \( K_p \in \mathbb{R}^{n \times n} \) is the proportional gain matrix and \( K_v \in \mathbb{R}^{n \times n} \) is the derivative gain matrix, both are symmetric and diagonal, [16]. The function \( \text{tanh}(\cdot) \) is a function continuous, monotonically increasing, and with derivative \( \text{sech}^2(\cdot) \). Also, satisfy \( \|\cdot\| \geq \|\text{tanh}(\cdot)\| \) and \( 1 \geq \|\text{tanh}(\cdot)\| \) for all \( \cdot \in \mathbb{R} \).

In this article, we use the following vectorial definition of the hyperbolic tangent function, [16].

**Definition 3** (16, Ch. 4, Sect. 4.5, pp. 109): We define the vectorial representation of the hyperbolic tangent function as
\[
f(x) = \begin{bmatrix}
  \text{tanh}(x_1) \\
  \vdots \\
  \text{tanh}(x_n)
\end{bmatrix}
\]
with \( x \in \mathbb{R}^n \). It satisfies the following properties for all \( x, \dot{x} \in \mathbb{R}^n \):
\[
\|f(x)\| \leq \alpha_1 \|x\| \tag{12}
\]
We derive (17) and we get 
\[
\begin{align*}
\|f(x)\| &\leq \alpha_2 \\
\|f(x)\|^2 &\leq \alpha_3 f^T(x)x \\
\|f(x)\| &\leq \alpha_4 |\dot{x}| 
\end{align*}
\] (13) (14) (15)

where \(\alpha_1, \ldots, \alpha_4 > 0\).

With \(f(x)\) defined as in (11), the various constants remain like \(\alpha_1 = 1, \alpha_2 = \sqrt{m}, \alpha_3 = 1\) and \(\alpha_4 = 1, [16]\).

By considering the robot manipulator equation described in (1), and the control equation defined in (10), we defined the closed loop system equation:
\[
\begin{bmatrix}
\ddot{q} \\
\dot{q}
\end{bmatrix} = \begin{bmatrix}
H(q)[K_p \psi(\dot{q}) - K_v \psi(\dot{q}) - C(q, \dot{q}, \ddot{q})]
\end{bmatrix}.
\] (16)

We use the following positive definite function:
\[
V(\ddot{q}, \dot{q}) = \frac{\ddot{q}^T H(q) \ddot{q}}{2} + \begin{bmatrix}
\sqrt{\ln (\cosh(q_1))} \\
\vdots \\
\sqrt{\ln (\cosh(q_n))}
\end{bmatrix}^T K_p \begin{bmatrix}
\frac{\tanh(q_1)}{\sqrt{\ln (\cosh(q_1))}} \\
\vdots \\
\frac{\tanh(q_n)}{\sqrt{\ln (\cosh(q_n))}}
\end{bmatrix} \dot{q} + \begin{bmatrix}
\frac{\tanh(q_1)}{\sqrt{\ln (\cosh(q_1))}} \\
\vdots \\
\frac{\tanh(q_n)}{\sqrt{\ln (\cosh(q_n))}}
\end{bmatrix}. \tag{17}
\]

As we can see, the function described in (17) fulfills the conditions
\[
\begin{align*}
V(0, 0) &= 0 \quad \forall \ddot{q}, \dot{q} = 0 \\
V(\ddot{q}, \dot{q}) &= 0 \quad \forall \ddot{q}, \dot{q} \neq 0
\end{align*} \tag{18} \tag{19}
\]

We derive (17) and we get
\[
\begin{align*}
\dot{V} &= \ddot{q}^T H(q) \ddot{q} + \frac{\ddot{q}^T \dot{q}}{2} + \begin{bmatrix}
\sqrt{\ln (\cosh(q_1))} \\
\vdots \\
\sqrt{\ln (\cosh(q_n))}
\end{bmatrix}^T K_p \begin{bmatrix}
\frac{\tanh(q_1)}{\sqrt{\ln (\cosh(q_1))}} \\
\vdots \\
\frac{\tanh(q_n)}{\sqrt{\ln (\cosh(q_n))}}
\end{bmatrix} \dot{q} \\
&\quad + \begin{bmatrix}
\frac{\tanh(q_1)}{\sqrt{\ln (\cosh(q_1))}} \\
\vdots \\
\frac{\tanh(q_n)}{\sqrt{\ln (\cosh(q_n))}}
\end{bmatrix}
\] (20)

where \(\begin{bmatrix}
\frac{\tanh(q_1)}{\sqrt{\ln (\cosh(q_1))}} \\
\vdots \\
\frac{\tanh(q_n)}{\sqrt{\ln (\cosh(q_n))}}
\end{bmatrix}\) is a diagonal matrix defined as
\[
\begin{bmatrix}
\frac{\tanh(q_1)}{\sqrt{\ln (\cosh(q_1))}} & 0 & 0 & 0 \\
0 & \frac{\tanh(q_2)}{\sqrt{\ln (\cosh(q_2))}} & 0 & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \frac{\tanh(q_n)}{\sqrt{\ln (\cosh(q_n))}}
\end{bmatrix}. \tag{21}
\]

After applying Property 2 and performing the operations, we get
\[
\dot{V} = -\ddot{q}^T K_v 
\begin{bmatrix}
\sinh(q_1) \sqrt{1 - \tanh^2(q_1)} \\
\vdots \\
\sinh(q_n) \sqrt{1 - \tanh^2(q_n)}
\end{bmatrix} \tag{22}
\]

Considering the following equality
\[
\sinh(\cdot) \sqrt{1 - \tanh^2(\cdot)} = \tanh(\cdot), \tag{23}
\]
we have
\[
\dot{V} = -\ddot{q}^T K_v \tanh(\dot{q}) \leq 0, \tag{24}
\]

where \(\dot{V}\) is negative semi-definite, this implies that the state \(x\), defined as
\[
x = \begin{bmatrix} q \\ \dot{q} \end{bmatrix} \tag{25}
\]
is bounded. This means that the equilibrium point of the system is stable.

We define the following relationship:
\[
K_v \leq \lambda_{\text{max}} \{K_v\}. \tag{26}
\]

By considering (12) and (26), we can define (24) as:
\[
\dot{V} = -\|\ddot{q}\| \lambda_{\text{max}} \{K_v\} \|\dot{q}\| \\
= -\|\ddot{q}\|^2 \lambda_{\text{max}} \{K_v\} \tag{27}
\]

(28)

where \(\alpha_1 = 1\).

In order to prove that \(\dot{V} \to 0\) and thus \(\ddot{q} \to 0\) and \(\dot{q} \to 0\), we use the following lemma.

Lemma 4 (Lyapunov-Like Lemma) [25, Chap. 4, Sect. 4.5, pp. 125]: If a scalar function \(V(x, t)\) satisfies the following conditions
\[
\begin{itemize}
\item V(x, t) is lower bounded
\item \dot{V}(x, t) is negative semi-definite
\item V(x, t) is uniformly continuous in time
\end{itemize}
\]
then \(\dot{V}(x, t) \to 0\) as \(t \to \infty\).

By Lemma 4 (Lyapunov-Like Lemma), \(\dot{V} \to 0\) when \(t \to \infty\), if (24) is uniformly continuous. A sufficient condition for (24) to be considered as uniformly continuous is that its derivative \(\dot{V}\) is bounded.

We derive (27), such that:
\[
\dot{V} = -2 \|\ddot{q}\| \lambda_{\text{max}} \{K_v\} \tag{28}
\]
as we can see (28) is bounded. This allows us to conclude that \(\dot{V} \to 0\) and thus \(\ddot{q} \to 0\) and \(\dot{q} \to 0\) when \(t \to \infty\).

D. TUNING

For the tuning of the control structure gains, we proposed in this article the use of differential evolution algorithms, the main aim being the realization of a comparative analysis that allows determining, through the performance index (\(L_2\)-norm), the evolution algorithm suitable for the proposed case of study. Recall that we describe in (10) the control to be used in the case of study, a PD+ control law type.

Tuning is a way to determine the control parameters according to a set of specifications for the system. However, the tune of the gains is an empirical procedure that, through trial and error, allows to get the numerical values that allow the system to fulfill the assigned task. Using evolution algorithms, we can generate an input vector \(v\) that covers the
parameters of the robot’s dynamic model, as well as the maximum and minimum limits of the torque. The resulting vector provides an output vector \( u_G \) that represents the change parameters to be implemented at the programming level in the robot.

We should consider that differential evolution algorithms are a class of evolution algorithms proposed as global optimization methods [26]. However, we presented the classical tuning of a PD+ control law to perform the comparison between algorithms.

1) CLASSICAL PD+ TYPE CONTROL LAW TUNING
The PD+ type control law defined in (10) has the characteristic of producing an bounded output because of \( \text{tanh}(\cdot) \), so we must calculate the gains \( K_p \) and \( K_v \) within the limits of the maximum motor torque, such that

\[
0 \leq K_p \leq 0.8 T_{\text{max}}
\]

\[
0 \leq K_v \leq 0.2 T_{\text{max}}.
\]

The PD+ type control has the disadvantage of requiring velocity information, which is got using numerical differentiation or observers. For this reason, the change made by (30) does not work accurately, so it is necessary to adjust the gains until we find the values to get an acceptable response without exceeding the torque limits, [15].

2) PD+ TYPE CONTROL LAW TUNING THROUGH DIFFERENTIAL EVOLUTION
Differential evolution algorithms are a class of algorithms proposed as methods of global optimization. The evolution algorithms generate an input vector, \( v_G \), containing the parameters of the system’s dynamic model, and the upper and lower torque limits, which are determined using (29) and (30). The input vector, \( v_G \), returns an output vector, \( u_G \), which represents the adjusted system parameters. The parameters that grow are the gains, \( K_p \) and \( K_v \), which depend on the characteristics of the motor torque.

The parameters established by the user are: mutation parameter \( (E) \), population size \( (NP) \), crossover factor \( (CR) \) and number of generations assigned \( (G) \), [27], [28].

In order to use differential evolution methods, it is important to establish the maximum and minimum search limits, and the mechanical characteristics of the system as initial conditions, such as dimensions, weight, mass, etc.

Another important aspect of these methods is the minimization of the target function, which in this case we use the \( L_2 \)-norm which, according to [23], if \( L_2 \rightarrow 0 \) then the position error of the links tends to zero, \( \dot{q} \rightarrow 0 \), which implies that the parameters found adjust and improve the behavior of the control scheme.

Next, we briefly describe the most used differential evolution algorithms in agricultural engineering, and we present the parameters, and gains got through this tuning process.

a: DE/RAND 1/BIN (DE1) ALGORITHM
In the literature, this algorithm has been widely used since its introduction in 1997 by Storn and Price, [29]. This tool is a search method that uses the population number \( (NP) \) to display the number of vectors. As a general rule, we assume a uniform distribution for random decision taken, so it has the characteristic of being a simultaneous operator of crossing and mutation.

The scheme of this algorithm involves a test vector \( v_i \), such that:

\[
v_i,G+1 = x_{i,1,G} + F[x_{i,2,G} - x_{i,3,G}]  \quad i = 1, 2, \ldots, NP
\]

The purpose of (31) is to generate a new vector of parameters \( v_i \), through the weight difference of two population member vectors \( (NP) \), represented by \( (x_{i,2,G} - x_{i,3,G}) \), and a third member vector \( x_{i,1,G} \) known as main member. We call this procedure a mutation. The mutation parameter \( F \in [0, 1] \) is a true and constant mutation factor that controls amplification in differential variation, [29].

The crossing combines the vector of mutated populations, \( v_{ji,G+1} \), with the chosen population vector, \( u_{ji,G+1} \), to generate a new vector of populations, \( u_{ji,G+1} \), such that (31) shows whether the ability of the mutated population, or resulting vector, is lower than the member of the chosen population, the new vector shall replace the vector with which it was compared, [29].

Otherwise, we maintain the population chosen for each generation \( (G) \),

\[
u_{ji,G+1} = \begin{cases} 
v_{ji,G+1} & \text{if } (\text{randb}(i) \leq CR) \text{ or } j = \text{rnbr}(i) \\
x_{ij,G} + 1 & \text{if } (\text{randb}(i) > CR) \text{ and } j \neq \text{rnbr}(i) 
\end{cases}
\]

where \( \text{randb}(j) \in [0, 1] \) is the j-th evaluation of uniformly generated random numbers, \( \text{randb}(i) \in [1, 2, \ldots, D] \) is a randomly chosen index, \( CR \in [0, 1] \) is the constant of crossover, a parameter that increases the diversity of individuals in the population, [29].

We note that the DE1 scheme has the disadvantage that we must find empirically the mutation and crossover factors. This has motivated several authors to improve its robustness.

b: SADE ALGORITHM
The SaDE algorithm is based on a learning period, [30]. The learning strategies used in this algorithm are

\* DE/rand/1/bin

\[
v_{i,G+1} = x_{i,1,G} + F[x_{i,2,G} - x_{i,3,G}]
\]

\* DE/current to best/2

\[
v_{i,G+1} = x_{i,G} + F[x_{\text{best},G} - x_{i,G}] + F[x_{i,1,G} - x_{i,2,G}] + F[x_{i,3,G} - x_{i,4,G}]
\]
We defined the test vector or mutation vector as:

\[ v_{i,G+1} = x_{r1,G} + F_i \left[ x_{r2,G} - x_{r3,G} \right] + F_i \left[ x_{r4,G} - x_{r5,G} \right] \]  

(35)

\[ v_{i,G+1} = x_{r0,G} + F_i \left[ x_{best_i,G} - x_{i,G} \right] + F_i \left[ x_{r1,G} - x_{r2,G} \right] \]  

(39)

where \( x_{r0,G} \) and \( x_{r1,G} \) are two vectors randomly selected from a population \( \mathcal{P} \), \( x_{r2,G} \) is a vector chosen from \( \mathcal{A} \cup \mathcal{A} \) where \( \mathcal{A} \) is the population of stored individuals that are replaced in the selection. Crossover operator \( CR \) is equal to the classic \( DE1 \) algorithm.

We defined the selection operator as:

\[ x_{i,G+1} = \begin{cases} u_{i,G}, & \text{if } f(u_{i,G}) < f(x_{i,G}) \\ x_{i,G}, & \text{otherwise.} \end{cases} \]  

(40)

This algorithm has a high performance compared to the first versions of the \( DE1 \) algorithm.

The \( JADE \) adaptive differential evolution algorithm can use \( DE/rand \) to best/1/bin (DE1) with the crossover operator

\[ v_i = F_i (u_{best} - u_i) + F_i (u_{r2} - u_3) \]  

(41)

or \( DE/current \) to best/1/bin (DE1), which has the operator crossover

\[ v_i = u_i + F_i (u_{best} - u_i) + F_i (u_{r2} - u_3) \]  

(42)

where \( u_i \) is the target vector and \( F_i \) is the mutation factor associated with \( u_i \), which changes with each generation.

An important modification occurs when an external file \( \mathcal{A} \) is used to save solutions lower than the current population \( \mathcal{P} \). So the mutation operator for the \( DE/rand \) to best/1/bin (DE1) algorithm is:

\[ v_i = u_{r1} + F_i (u_{best} - u_{r1}) + F_i (u_{r2} - u_3) \]  

(43)

where \( u_{r3} \) is a different vector of \( u_{r1} \) and we randomly chosen \( u_{r2} \) from the union of \( \mathcal{A} \) and \( \mathcal{P} \), and the mutation operator for the \( DE/rand \) to best/1/bin (DE1) algorithm is:

\[ v_i = u_{r1} + F_i (u_{best} - u_{r1}) + F_i (u_{r2} - u_2) \]  

(44)

where \( u_{r2} \) is a different vector of \( u_1 \) and \( u_{r1} \), randomly selected from the junction of \( \mathcal{A} \) and \( \mathcal{P} \).

The following equation

\[ F_i = \text{rand} \left( \mu_F, 0.1 \right) \]  

(45)

where \( \text{rand} \) \((\mu_F, 0.1)\) means a Cauchy distribution with the location parameter \( \mu_F \) and the parameter scale 0.1.

If \( F_i \geq 1 \) this means \( F_i \) is truncated at 1, and if \( F_i \leq 1 \), this means \( F_i \) is regenerated.

We initialize the \( \mu_F \) parameter at 0.5 and we update at the end of each generation using

\[ \mu_F = (1 - c) \mu_F + c \times \text{mean}_L \left( s_F \right) \]  

(46)
where $\mu_F$ is the set of all successful mutation factors $F_i$ that carry out the adaptation of the Mutation factor ($F_i$, $c \in [0, 1]$), and $\text{mean}_L(S_F)$ is the Lehmer mean, defined as

$$\text{mean}_L(S_F) = \frac{\sum_{F \in S_F} F^2}{\sum_{F \in S_F} F}. \quad (47)$$

We made the adaptation of the probability of crossing ($CR_i$) for each individual through a normal distribution with mean $\mu_{CR}$ and standard deviation 0.1 as

$$CR_i = \text{randn}_i(\mu_{CR}, 0.1). \quad (48)$$

Below, we present the pseudo-code Algorithm 1 that describes the differential evaluation algorithm. We should note that the pseudo-code shows the most important elements of the program, while we present the operators in the appendices.

**Algorithm 1** Differential Evaluation

**Require:** Generate a population of solutions ($\hat{v}(0)$).

**Require:** Evaluate each solution $g = 1$.

1: while we do not reach convergence do
2: for all $i = 1$ to $NP$ do
3: Apply the differential mutation operator. \hspace{1cm} (Appendix B)
4: Apply the differential crossover operator. \hspace{1cm} (Appendix C)
5: Evaluate the new solution using the objective function.
6: Apply the differential selection operator. \hspace{1cm} (Appendix D)
7: end for
8: $g = g + 1$
9: end while

**E. $L_2$-NORM**

Robot manipulator are very complex mechanical systems due to the non-linear nature of their dynamics. For this reason, there are no well-established criteria for proper evaluation of control structures in the robotic community. However, the $L_2$-norm is accepted to compare the performance of the controllers, as a numerical measure to compare the error curve [16]. We accept this criterion as an objective function for DE algorithms.

We define the $L_2$-norm as

$$L_2 = \sqrt{\frac{1}{T} \int_{t_0}^{t_1} \left\| \dot{q}(t) \right\|^2 dt} \quad (49)$$

where $T$ is the sampling period, $\dot{q} \in \mathbb{R}^n$ is the position error in joint coordinates, $t_0$ and $t_1$ are the start and end times, respectively.

A norm is a function from a real or complex vector space to the nonnegative real numbers that behaves in certain ways like the distance from the origin. The norm commutes with scaling, obeys a form of the triangle inequality, and is zero only at the origin.

In particular, the Euclidean distance of a vector $\hat{\cdot}$ from the origin is a norm, called the Euclidean norm, or 2-norm, which we defined as the square root of the inner product of a vector with itself:

$$\|\hat{\cdot}\|^2 = \sqrt{\langle \hat{\cdot}, \hat{\cdot} \rangle} = \langle \hat{\cdot}, \hat{\cdot} \rangle \quad (50)$$

The $L_2$-norm averages the position errors under the action of different tuning algorithms. For this, each Differential Evolution Algorithm takes the $L_2$-norm as an objective function to calculate the value of the gains. When considering the $L_2$-norm as the objective function if it tends to zero, $L_2 \to 0$, the solutions reached are the optimal ones.

**III. CASE OF STUDY: DETECTING AND CUTTING LEAVES USING A CARTESIAN ROBOT**

Since in vitro propagation is an important technique for the rapid clonal production of disease-free plants, the transformation of the plant genome, and the production of commercially valuable plant-derived metabolites, the use of robots is important to automate these processes, [33].

We use cultivated tissues for the propagation of plants, which allows us to create whole plants. The pieces of tissue or small organs used are called explants. The selection of the part of the plant from which we will obtain explant will depend on the plant species, the purpose and the type of crop to obtain, [33].

In some protocols, we use true leaves. So, tissue isolation is just one of the key steps for in vitro propagation. Tissue isolation consists of isolating a part of the plant using some tool (sterile blades, micro scissors, laser, etc.) and then cultivating the isolated tissue in the physical and chemical conditions necessary to express its intrinsic potential, [33].

Because this process is repetitive, robotics has received a lot of attention from agricultural engineering, as robots are programmable multifunctional manipulators that can perform multiple tasks autonomously and repetitively through programmed movements, [34], allowing agricultural engineers to make the leaf cutting process more efficient.

Typically, we make the cutting of the leaves on a flat surface inside a petri box using a cutting tool and a handling tool. We hold the leaves with precision tweezers and cut with a scalpel. Due to the characteristics of the task, we have selected the Cartesian robot of 3 DOF as the ideal to perform the task helped by a vision system.

**A. EXPERIMENTAL PLATFORM**

Figure 2 shows a 3-DOF Cartesian robot built of aluminum and nylamid with prismatic-type links.

The robot uses three 24 V to 2.1 A DC motors, has a speed of 0-122 rpm, an accuracy of 0.367°, and a maximum torque of 1.568 Nm.

The system has incremental encoders with 980 pulses per revolution. We implemented a 1/32 inch vertical cutter as
A cutting tool that allows to get leaf sections of 0.02 m². We attach the leaves to an adherent surface that holds the leaf, as the robot executes the cutting trajectory. We present the mechanical characteristics of the 3-DOF robot in Table 1.

Table 1. Mechanical characteristics.

| Link | Mass [kg] | Size [m] |
|------|-----------|----------|
| 1    | 3.80      | 0.40 × 0.07 |
| 2    | 2.70      | 0.41 × 0.39 |
| 3    | 2.35      | 0.21 × 0.21 |

We define the forward kinematics of the Cartesian robot as

\[
\mathbf{X} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix} \tag{51}
\]

where \( q_1, q_2 \) and \( q_3 \) are in meters. We define the dynamic model of the Cartesian robot as:

\[
\begin{bmatrix} m_1 & 0 & 0 \\ 0 & m_1 + m_2 & 0 \\ 0 & 0 & m \end{bmatrix} \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \\ \ddot{q}_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ m \end{bmatrix} \mathbf{g} = \begin{bmatrix} \tau_{m1} \\ \tau_{m2} \\ \tau_{m3} \end{bmatrix} \tag{52}
\]

where \( m = m_1 + m_2 + m_3, m_1, m_2 \) and \( m_3 \) are the masses of the links of the Cartesian robot. It is important to note that Cartesian robots do not have a Coriolis matrix and a centripetal force, [35].

Figure 3 describes the visual system block diagram. As we can see, the camera provides, after extraction, the centroid of the image.

We consider the centroid as the center of the desired trajectory \( \mathbf{q}_d \in \mathbb{R}^3 \), which is compared to the robot actual position \( \mathbf{q} \in \mathbb{R}^3 \) in order to generate the error vector \( \mathbf{q} - \mathbf{q}_d \) which is part of the input of the control structure in order to perform the tracking task, [24].

In order to clone the mother plant for bio-preservation purposes, to propagate the plant tissue, we must make a circular cut of 0.02 m² on the selected leaf.

For this reason, we must get the centroid through the camera, and define the center of the desired trajectory. In this case, we define the desired trajectory (circular trajectory) as

\[
\mathbf{X}_d = \begin{bmatrix} x_d \\ y_d \\ z_d \end{bmatrix} = \begin{bmatrix} x_c + r \sin (\omega t) \\ y_c + r \cos (\omega t) \\ 0 \end{bmatrix}, \tag{53}
\]

where \( x_c \) and \( y_c \in \mathbb{R} \) represent the coordinates of the center of the circle, \( r \in \mathbb{R} \) is the radius of the circle, \( \omega \in \mathbb{R} \) is the period in which the system has to draw the circle.

Information got from a vision sensor is used to control the movement of the robot. For leaf detection, we mount a Logitech C270 camera 0.4 meters away and perpendicular to the scene. We describe the camera features in Table 2.

Table 2. Logitech C270 camera features.

| Characteristics     | Value        |
|---------------------|--------------|
| Focal distance      | 0.04 m       |
| Pixel size          | 2.8 x 2.8 \( \mu \)m |
| Image resolution    | 640 x 480 pixels |

It is important to note that a digital camera forms a discrete representation of the plane of the object. The number of photographic sensors in the CCD matrix does not match the number of elements in the digital image because of analog processing and digitization.

The coordinates of the centre of the image plane also do not match the coordinates of the centre of the digital image, [24]. In the system the camera is perpendicular to the scene being analyzed, therefore, the following equation relates the pixels \((u, v)\) with the real coordinates of the scene \((x_m, y_m)\):

\[
\begin{bmatrix} u \\ v \end{bmatrix} = \frac{f \alpha}{z} \begin{bmatrix} x_m \\ y_m \end{bmatrix}, \tag{54}
\]

where \( f \) is the focal length, \( z \) is the distance from the camera to the scene, and \( \alpha \) is the scale factor.

So that the robot does not collide with the structure of the camera, it is 0.4 meters from the scene, so we have:

\[
\begin{bmatrix} u \\ v \end{bmatrix} = 3663 \begin{bmatrix} x_m \\ y_m \end{bmatrix}. \tag{55}
\]

We use the function developed in Matlab by Bhargav Anand to detect the leaf by considering its green color, which
allows us to discriminate objects that are not of interest, [36] (Appendix A). This function allows the manipulation of the RGB planes in such a way that:

$$ W = R_{\text{adj}} + G_{\text{adj}} + B_{\text{adj}} $$  \hspace{1cm} (56)

where $R_{\text{adj}}$, $G_{\text{adj}}$ and $B_{\text{adj}}$ are the attenuation factors for the red, green and blue planes. The user must assign the attenuation values. This is important since the leaves have the characteristic of being loaded with reds representing warm greens. When the leaves are young or cold, it loaded them onto the blue plane, [36].

We found the set-point required for the control as the pixel closest to the leaf centroid, which we previously detected and located in the image as the green region of interest.

If $u_{x,i}, v_{x,i}$ with $i = 1, \ldots m$ denote the coordinates of the pixels on the leaf, then its centroid $(u, v)$, also called the center of mass or point of gravitational center, is given by

$$ u = \frac{1}{m} \sum_{i=1}^{m} u_{x,i} \hspace{1cm} (57) $$

$$ v = \frac{1}{m} \sum_{i=1}^{m} v_{x,i} \hspace{1cm} (58) $$

**B. TUNING ALGORITHMS EVALUATION**

The differential evolution algorithms presented in Section II-D are used to find the $K_p \in \mathbb{R}^{n \times n}$ and $K_v \in \mathbb{R}^{n \times n}$ gains of the control law defined in (10). In order to evaluate the algorithms, we set a test point $X_{\text{test}}$ as reported in [37], and programmed the tuning algorithms to minimize $v_i$, repeating the search 10 times until we get the adjusted parameters’ mean value. By using the Petri dish’s center in which will place the leaf, Figure 4, we define the next test point (in meters)

$$ X_{\text{test}} = \begin{bmatrix} x_{\text{test}} \\ y_{\text{test}} \\ z_{\text{test}} \end{bmatrix} = \begin{bmatrix} 0.065 \\ 0.145 \\ 0.065 \end{bmatrix} \hspace{1cm} (59) $$

We used in the control law the adjusted parameters that are got by minimizing $v_i$, so that the Cartesian robot moves from the initial position to the test point. In addition to the value of the gains to be used in the control law, it is important to consider the estimated time to obtain the parameters. In Table 3, we present the gains got using the tuning algorithms described in Section II-D, and in Table 4, we present the parameters used in each algorithm.

![FIGURE 4. Test point.](image)

**TABLE 4. Parameters used in tuning algorithms.**

| Algorithm | (F) | (CR) | (NP) | (G) |
|-----------|-----|------|------|-----|
| DEI      | 0.1 | 0.5  | 45   | 150 |
| SaDE     | 0.1 | 0.9  | 50   | 150 |
| JDE      | 0.9 | 0.9  | 50   | 150 |
| JADE     | 0.7 | 0.7  | 18   | 150 |

As shown in Table 5, the best performs algorithm is the JADE algorithm. We will tune the case of study using this algorithm.

After we getting the gain values, we apply these to the control law defined in (10) in order to move the robot to the test point. We evaluate the position error that generates each pair of gains using the $L_2$-norm, so that we can determine the algorithm that generates the best gain values in the shortest time. In Table 5, we present the values got by solving (49).

In Table 6 we present mean standard deviation

As shown in Table 5, the best performs algorithm is the JADE algorithm. We will tune the case of study using this algorithm.

In Figure 5, we present the results obtained from the evaluation of the $L_2$-norm of each of the algorithms used.
In Figure 8, we show the positions of the three links of the Cartesian robot in order to reach the test point using the different gain values. We observe in Figures 8(a) to Figure 8(d) that for link 1 there is not much difference between the use of classical tuning and differential evolution tuning methods. However, we observed in Figures 9(a) to Figure 9(d) that with the tuning proposed by [15], [16] the link 2 does not present stabilization in the steady state, while with the change of the differential evolution algorithms we achieve a stabilization below of the 0.5 seconds. For link 3, we get a high overshoot using the tuning proposed in [15], [16], while with the differential evolution algorithms, we improved the response in a transient state, reducing overshoots and getting a settling time in less than 0.31 seconds as seen in Figures 10(a) to Figure 10(d).

C. EXPERIMENTAL RESULTS
The principal aim of the case of study is to perform centroid detection on a leaf, and then perform the cut using a Cartesian robot, Figure 2.

In order to perform the cutting task, we used a PD+ type control, equation (10), whose gains we tune using the JADE algorithm, which has the best performance, according to the evaluation carried out in the previous section.

As we have explained, the desired trajectory is a circle, equation (53), whose center must be located using the centroid provided by the vision system, Figure 3. From the centroid, we defined the circle of 0.02 m of diameter, and we make the cut.

We can perform this task repetitively thanks to the use of the Cartesian robot.

We perform the tracking of the trajectory using the PD+ type control using the tuning performed with the classical method and the JADE algorithm and we perform the comparison of the trajectories. We present the experimental results in Figure 6, where we observe the tracking of the desired trajectory that the robot performs.

The several resulting cut are shown in Figure 7.
FIGURE 8. Positions of the Link 1 produced by the control law using each tuning algorithm to arrive at $X_{\text{test}}$ (x component).

FIGURE 9. Positions of the Link 2 produced by the control law using each tuning algorithm to arrive at $X_{\text{test}}$ (y component).
IV. CONCLUSION

This article presents the study of four global optimization methods where the DE1 and JADE methods showed an outstanding performance for the tuning of the PD+ type controller. Through the $L_2$-norm, it is determined that the algorithm with the best performance is the JADE algorithm (0.02236 meters).

We carried the performance index evaluation procedure by considering the test point $X_{\text{test}}$ as the desired position.

Because of performing the JADE algorithm, for the case of study, it was used to tune the control and fulfill cutting a circle on a leaf from its centroid, which is got using the visual system.

APPENDIX A BHARGAV ANAND FUNCTION

```
diff_im = bwareaopen(diff_im, 300);  
bw = bwlabel(diff_im, 8);  
stats = regionprops(bw, 'BoundingBox', ...  
', 'Centroid');  
imageshow(data)  
hold on  
for object = 1:length(stats)  
    bb = stats(object).BoundingBox;  
    bc = stats(object).Centroid;  
    rectangle('Position', bb, 'EdgeColor', ...  
', 'r', 'LineWidth', 2);  
    plot([bc(1), bc(2), '-m+'])  
    a=text(bc(1)+15, bc(2), strcat('X: ', ...  
', num2str(round(bc(1))), ', Y...  
', num2str(round(bc(2)))));  
    set(a, 'FontName', 'Arial', ', '  
', FontWeight', 'bold', ', FontSize...  
', 12, 'Color', 'yellow');  
end  
hold off  
end  
stop(vid);  
flushdata(vid);  
clear all  
sprintf('hs', ',That was all about Image ...  
', tracking, Guess that was pretty easy :)...  
')
```

APPENDIX B MUTATION OPERATOR

The mutation operator generates ($NP$) new vectors $v_i$ according to the equation:

$$v_i = u_{r1} + F(u_{r2} - u_{r3})$$ (60)
where \( r_1, r_2, r_3 \in [1, 2, \ldots, NP] \) are mutually indices different for there to be a mutation and also different from \( i \) index, so that (60) does not become an operator of crossing. We can guarantee this by sampling three vectors in randomly, from the number of individuals in the population (\( NP \)) and verifying that we satisfied the conditions \( r_1 \neq i, r_2 \neq r_1, r_2 \neq i, r_3 \neq r_1, r_3 \neq r_2, r_3 \neq i \). Also, \( F \) is a scalar constant that affects the differential variation between two vectors.

**APPENDIX C CROSSOVER OPERATOR**
The crossing comprises combining the vector

\[
v_i = [v_{i1}, v_{i2}, \ldots, v_{id}],
\]

previously mutated, with another vector called the white vector

\[
u_i = [u_{i1}, u_{i2}, \ldots, u_{id}],
\]

which we considered as an individual parent of the previous generation. Thus, we generated a new descending vector

\[
\bar{u}_i = [\bar{u}_{i1}, \bar{u}_{i2}, \ldots, \bar{u}_{id}],
\]

called test vector. The following equation implements the binomial crossover:

\[
\bar{u}_{ij} = \begin{cases} 
u_{ij} & \text{if } (\text{randb}(j)) \leq \text{CR} \ o j = \text{rnbr}(i) \\ u_{ij} & \text{if } (\text{randb}(j)) > \text{CR} \ o j \neq \text{rnbr}(i) \end{cases}
\]

for \( i = 1, 2, \ldots, NP \) and \( j = 1, 2, \ldots, d \); where \( (\text{randb}(j)) \in [0, 1] \) is the \( j \)-th evaluation of a number generator uniform random; \( \text{rnbr}(i) \in 1, 2, \ldots, d \) is a generated index randomly.

\( \text{CR} \in [0, 1] \) represents the probability of crossbreeding, which increases the diversity of the population. Large values of \( \text{CR} \) generate a child vector \( \bar{u}_i \) more similar to the mutated vector \( \bar{v}_i \) and vice versa. Therefore, with a value closer to unity, we increased the velocity by convergence of the algorithm. Even with a probability of zero crossover, the crossover operator ensures the descendant vector differs from the parent vector, minus one value.

**APPENDIX D SELECTION OPERATOR**
It is a very simple operator since it only comprises comparing the value of the cost function \( f(u_i) \) of the target vector \( u_i \) against the corresponding value of the aim function \( f(\bar{u}_i) \) of the test vector \( \bar{u}_i \). The vector with the lowest value of the cost function becomes a member of the population in the next generation.
% Trial(jj) = Pop{i, jj} + F*(Pop{BestIndividual, jj} + Pop{a, jj} - Pop{b, jj});
else
Trial(jj) = Pop{i, jj};
end
end
end
% DE/best/1/bin Strategy
function Trial = DE/best(bin, varargin)
% Crossover factor randomization
j = floor(rand*NVAR);
jj = j + 1;
for k = 1:NVAR
if((rand < CR) jj (k=1:NVAR))
Trial(jj) = Pop{BestIndividual, jj} + F*(Pop{a, jj} - Pop{b, jj});
else
Trial(jj) = Pop{i, jj};
end
end
% Target function through DE/rand/1/bin
function Trial = RandBin(Pop, varargin)
Trial = zeros(1, NVAR+2);
% Mutation factor adaptation function
if (rand<tau)
F = F1 + rand*Fu;
else
F = Pop{i, NVAR+1};
end
if (rand<Tau2)
CR = rand;
else
CR = Pop{i, NVAR+2};
end
% Apply the randomized mutation factor
j = floor(rand*+NVAR);
jj = j + 1;
for k = 1:NVAR
if((rand < CR) jj (k=1:NVAR))
Trial(jj) = Pop{i, jj} + Kr*(Pop{c, jj} ... Pop{i, jj}) + F*(Pop{a, jj} - Pop{b, jj});
else
Trial(jj) = Pop{i, jj};
end
end
REFERENCES
[1] D. Bedolla-Martínez, E. Lugo-González, F. Trujillo-Romero, and H. R. Kamili-Leyva, “Offline tuner of a discrete PID controller using a multi-objective genetic algorithm,” J. Comput. Sci. Comput. Eng., vol. 113, pp. 171–180, Dec. 2016.
[2] K. J. Aström and T. Hägglund, “The future of PID control,” Control Eng. Pract., vol. 9, no. 11, pp. 1163–1175, Apr. 2001, doi: 10.1016/S0967-0661(01)00062-4.
[3] N. J. Killingsworth and M. Krešić, “PID tuning using extremum seeking: Online, model-free performance optimization,” IEEE Control Syst. Mag., vol. 26, no. 1, pp. 70–79, Feb. 2006, doi: 10.1109/MCS.2006.1580155.
[4] W.-D. Chang, “A multi-crossover genetic approach to multivariable PID controllers tuning,” Expert Syst. Appl., vol. 33, no. 3, pp. 620–626, Oct. 2007, doi: 10.1016/j.eswa.2006.06.003.
[5] S.-K. Oh, H.-J. Jang, and W. Pedrycz, “The design of a fuzzy cascade controller for ball and beam system: A study in optimization with the use of parallel genetic algorithms,” Eng. Appl. Artif. Intell., vol. 22, no. 2, pp. 261–271, Mar. 2009, doi: 10.1016/j.engappai.2008.07.003.
[6] H. V. H. Ayala and L. dos Santos Coelho, “Tuning of PID controller based on a multiobjective genetic algorithm applied to a robotic manipulator,” Expert Syst. Appl., vol. 39, no. 10, pp. 8968–8974, Aug. 2012, doi: 10.1016/j.eswa.2012.02.027.
P. Sánchez-Sánchez et al.: Differential Evolution Algorithms Comparison Used to Tune Visual Control Law

[31] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer, “Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems,” IEEE Trans. Evol. Comput., vol. 10, no. 6, pp. 646–657, Dec. 2006.

[32] J. Zhang and A. C. Sanderson, Adaptive Differential Evolution. Berlin, Germany: Springer, 2009, pp. 39–82.

[33] E. F. George, M. A. Hall, and G. J. De Klerk, Plant Propagation by Tissue Culture, 3rd ed. Dordrecht, The Netherlands: Springer, 2008.

[34] M. W. Spong and M. Vidyasagar, Robot Dynamics and Control. Hoboken, NJ, USA: Wiley, 2005.

[35] P. Sánchez-Sánchez, F. Reyes-Cortés, and R. Reyes-Ruiz, “Cartesian controllers for robot manipulators,” in Proc. Int. Symp. Robot. Automat., Querétaro, México, 2004, pp. 347–351.

[36] A. B. Anand. (2022). Tracking red color objects using MATLAB. MATLAB Central File Exchange. Accessed: Mar. 16, 2022. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/28757-tracking-red-color-objects-using-MATLAB

[37] J. G. C. Reyes, F. F. H. Schlam, A. R. Garcia, E. R. Kriuchkova, and A. M. Camarillo, “Design of a position control based on cuckoo search tuning for a cutter leaves robot,” IEEE Latin Amer. Trans., vol. 14, no. 5, pp. 2085–2092, May 2016.

PABLO SÁNCHEZ-SÁNCHEZ received the Doctor degree in electronic engineering from the National Autonomous University of Mexico. He is currently a full-time Research Professor at the Faculty of Electronics Sciences, Benemérita Universidad Autónoma de Puebla. His research interests include mechanical systems modeling, vision, force control, robotics, and simulator design.

JOSE GUILLERMO CEBADA-REYES (Member, IEEE) received the degree in electronics from the Benemérita Universidad Autónoma de Puebla and the Ph.D. degree in agricultural engineering from the Universidad Autónoma Chapingo. He is currently a full-time Professor at the Instituto Tecnológico Superior de Teziutlán attached to the Mechatronics Department. His research interests include robotics, biological and industrial process automation, meta-heuristic optimization, visual servoing, applied electronics in bio-processes, and industrial processes.

AGUSTÍN RUIZ-GARCÍA received the Ph.D. degree in agricultural engineering from the Department of Agricultural Engineering and Integrated Water Use, Universidad Autónoma Chapingo. He is currently a full-time Professor with the Irrigation Department, Universidad Autónoma Chapingo. His research interests include dynamic and meta-heuristic optimization, biosystems modeling, and Kalman filtering.

AIDÉE MONTIEL-MARTÍNEZ received the B.S. degree in biological sciences from the Benemérita Universidad Autónoma de Puebla and the Ph.D. degree in biological sciences from the Universidad Nacional Autónoma de México. She is currently a full-time Professor at the Faculty of Electronics Sciences, Benemérita Universidad Autónoma de Puebla. Her research interests include urban ecology, sustainable development, and blue-green infrastructure.

FERNANDO REYES-CORTÉS was born in Puebla, Mexico, in March 1962. He received the B.S. degree in electronic engineering from the Benemérita Universidad Autónoma de Puebla, Puebla, in 1984, the M.S. degree from the Instituto Nacional de Astrofísica, Óptica y Electrónica, Puebla, in 1989, and the Ph.D. degree in electronics from the Centro de Investigación Científica y de Educación Superior de Ensenada (CICESE), Ensenada, Mexico, in 1997. In 1980, he joined the Benemérita Universidad Autónoma de Puebla, where he is currently a Professor and a Researcher. He has published four books and over 250 scientific papers in national and international conferences and journals. His research interest includes the fields of control of robot manipulators with special emphasis on practical applications.