Adaptative Evolution Strategy for Robotic Manipulation

César Arismendi, Javier V. Gómez, Santiago Garrido, Luis Moreno

***Abstract*—Teaching a mobile robot to complete a task and to reproduce it is possible, but as the robot tries to replicate actions natural events as a wheel-slide would feed in inaccuracies on the localization of the robot mobile base, and it may be difficult to succeed replicating. Robot tasks can be represented as trajectories compound by a series of poses and movements. We propose an algorithm for adapting manipulation trajectories to different initial conditions from those of the learned assignment. The adaptation is achieve by optimizing in position, orientation and energy conservation. Manipulation paths generated can achieve optimal performance sometimes even improving original path smoothness. The approach is builded over the basis of Evolution Strategies(ES), and only uses forward kinematics permitting to avoid all the inconveniences that Inverse kinematics imply as well as convergence problems in singular kinematic configurations. Experimental results are presented to verify the algorithm.**

I. Introduction

Mobile manipulators combine a robotic manipulator with a mobile base and its purpose is to offer support in complicated tasks involving manipulation of tools or objects within human environment; to accomplish they must localize their base in a place where manipulation is possible, trigger the servomotors within manipulator links to place the end effector and finally perform a predefined task.

In order to accomplish a task, inverse kinematics are required to determine manipulator configurations given an endpoint position and orientation coordinates. Usually, approximated numerical methods are used to solve inverse kinematics due to the complexity and sometimes indeterminacy of analytical solutions. Many numerical methods have been proposed; the most common methods are based on the Newton-Raphson method [1], [2] and the damped least squares method [3], [4], but these methods have convergence problems in singular kinematic configurations because of their dependence on the Jacobian Matrix. Another manner of solving the problem consist on formulating the inverse kinematic as a constraint optimization problem. This way all singularities are avoided by only using the forward kinematics. In order to solve this optimization problem, several methods have been proposed, from classic methods like conjugate gradient and the Cyclic Coordinate Descent [5], to artificial intelligence methods as neural networks [6]–[11], Fuzzy Logic [12]–[16] and evolutionary algorithms [17]–[19].

Evolution Strategies (ES) have proven being robust and versatile by not depending on any continuity or derivability condition and being successfully applied in various disciplines. Never the less, as mentioned in [20], many works consider only the position error, while calculate orientation error using analytical methods. In [20] Differential Evolution is used to find an optimal path of manipulation, but the algorithm far from being real-time, requires a large amount of time and mobile base disturbance is not considered, thereby the robot base stays still at the same place where learned path was made while only arm pose changes. In our previous work [21], we proposed a time constrained algorithm with a minute period term, but manipulation path smoothness was not considered and computation time could be reduced.

The objective of this work is to adapt from a different location a mobile robot manipulation path using an Evolution Strategy. A system is said to be real-time if the total correctness of an operation depends not only upon its logical correctness, but also upon the time in which it is performed [22]. Tasks considered here include reaching tools and objects; we considered these tasks to have a five seconds tolerance term. This rule permit us to said that the algorithm is performed in real time as long as the tolerance time is not outperformed, operations completed after its deadline are considered useless.

Given a manipulation task in configuration space described by an end effector path in Cartesian space with a specific position and orientation of the robot base, a new path of manipulation is generated in real-time using a modified approach of the basic Evolution Strategies scheme, assuming different initial conditions from that in the learned path.

The modified Evolution Strategy (ES) here implemented uses criterions as simple as possible, minimizing computational burden, and reaching fast optimal results that coherently describe a manipulation task.

After a defined task manipulation path is obtained, the problem arrives when trying to replicate it. In mobile robotics, localization error is a recurring problem caused by imperfections present in floor surfaces and robot's hardware, making wheels slip among other errors [23]; in consequence robot bases rarely ever would situate in the exact same position and orientation where tasks were learned. In spite of that, determine correct robot placement it's possible through measure equipment such as lasers and 3D cameras. Once the new location is determined we proceed to apply an evolutionary algorithm to adapt a learned path to a new sequence that performs the original task.

This paper presents in section II the manipulation path adaptation problem. Section III describes the used evolution strategy with some modifications to the original scheme. Finally we show experimental results obtained by testing the algorithm on the robot MANFRED.

II. Path Evolutive Adaptation Problem

For a mobile manipulator a task may be defined as a sequence of points in the Cartesian space defining a path. This sequence of joint configurations is defined as [20]

$$\Omega_l = \{\vec{q}_k\}, \qquad k = 1, 2, \ldots, N, \tag{1}$$

where $\vec{q}_k \in \Re^n$ is a vector of joint variables $q_{k,j}$,

$$\vec{q}_{k,j} = (q_{k,1}, \ldots, q_{k,j}, \ldots, q_{k,n})^T.$$

We assume a given path Ω_l which describes the robot's goal or task, this path may be obtained by learning methods based on imitation, teleoperation or teaching techniques. After a displacement, the initial location of the robot base is different but near from that of Ω_l, this path must be adapted to the new location to complete the desired task.

By optimizing the end effector's position and orientations errors a new path is obtained for a predefined task. The trajectory is soften by considering the energy consumption, and it is determined by the sum of the manipulator joints displacements as

$$\zeta(\Omega) = \frac{1}{2_\pi} \sum_{k=1}^{N} |\vec{q}_k - \vec{q}_{k-1}| \tag{2}$$

In evolutive methods the step length is the disturbance introduced to design variables in order to change and evolve them from initial to optimum positions, this variation parameter is denoted by σ. In manipulation planing, the end-effector position can be changed in only one axe with a movement, while for orientation axes are hard coupled, varying all or at least two simultaneously when a rotation over an axis is executed. This makes orientation optimization harder and more computational time consuming than that of position, due to small changes in position could generate large variations in orientation. To overcome this problem, position minimization is first made with a large step length σ_{hi} approximated to that proposed by [24] and after position optimization target is reached, orientation minimization is added to optimization with a ten times smaller step length σ_{low}. This strategy results in improved convergence time of the algorithm.

Rechenberg's success rule is used for controlling σ size, [24]. After every N_b (number of design variables) iterations, the number of successes occurred over the preceding $10N_b$ mutations are revised. If this number is less than $2N_b$, step size is multiplied by a factor of 0.85, or divided by 0.85 if more than $2N_b$ successes occurred.

For the initial values of σ, Schwefel proposes to use the following estimation

$$\sigma_i^0 = \frac{\Delta b_i}{\sqrt{N_b}}$$

where Δb_i is the expected distance from the optimum for the corresponding design variable. Notwithstanding as the accuracy for this initial σ value is not critical because the law of success seems to quickly adapt the step size, a generalized form is deduced to approximate initial Schwefel values

$$\sigma_i^0 = \frac{\Delta d_{q_N}}{10(N_b + 1)} \tag{3}$$

where Δd_{q_N} is the last node distance error in millimeters. Value obtained here approximates experimental results average of Schwefel estimations that made no significant differences on convergence times respect to that of the exact estimation. When optimizing orientation, step length in (3) is reduced by a factor of ten.

To determine an appropriate size of the parent population μ there is no accurate rule. A good indicator is to have as many or a few times as many members as the number of design variables; parent population size used here is $N_b = 6$.

On the contrary, some theoretical studies have been realized on $(1, \lambda)$ strategies [24], where λ is the offspring population size, to estimate the optimal ratio between λ/μ. It has been shown that this ratio depends on the objective function and increases with its complexity. A ratio λ/μ equal to 5 can be considered as a good starting point, therefore an offspring population $\lambda = 30$ is chosen here.

Total error is the sum of the position error at each path point $k = 2, \ldots, N$, and the orientation error in the last two nodes $k = (N-1), N$, with weights $W_1 = 0.5$ and $W_2 = 1$ respectively, attaching greater importance to the last point where the robot is meant to perform the manipulation. Thereby, the joint configuration path must be transformed into end-effector position and orientation coordinates through the robot manipulator kinematic model. Position and orientation errors, denoted as E_P and E_0, are defined as [20]

$$E_P(\Omega) = \frac{1}{2R_{max}} \sum_{k=2}^{N} |p_{l_k} - p_k| \tag{4}$$

and

$$E_O(\Omega) = \frac{1}{2\pi} \sum_{k=N-1}^{N} |\varphi_{l_k} - \varphi_k|, \tag{5}$$

where $(\vec{(}p_l), \vec{(}\varphi_l))$ are the desired position and orientation coordinates calculated by Ω_l forward kinematics and R_{max} is the robot manipulator maximum reach suggested by [17] as a normalization value. The resulting optimal joint path $\Omega*$ minimizes the total deviation respect to Ω_l, and the optimization problem is conducted by the minimization of:

$$f(\Omega) = w_1 E_P(\Omega) + w_2 E_O(\Omega) + w_3 \zeta(\Omega). \tag{6}$$

subjetc to:

$$C = \{\Omega \mid g(\Omega) \leq 0 \wedge h(\Omega) \geq 0\},$$

where g and h are restrictions imposed by the mechanical joint limits of the robot manipulator, and w_1, w_2 and w_3 are weighting factors used according to task priorities.

III. Evolution Strategy Adaptation Algorithm

The path evolutive adaptation is accomplished with an implementation of the method denominated Evolution Strategies (ES) [25]. The algorithm used to adapt the manipulation path is illustrated in Algorithm 1, where P_g, and Po_g are respectively the father and offspring Population on generation g, and n_b is the number of design variables, which corresponds to the number of manipulator joints.

Algorithm 1 Evolution Strategy

```
initialization Pg = 0
evaluation Pg
while termination criterion ≠ true do
    Po_g ← Evolutionary mutation
    evaluation Po_g
    P_{g+1} ← selection(Po_g ⋃ P_g)
    if optimal position = true then
        reduce σ
    end if
    if g module(10 n_b) = 0 then
        step length control
    end if
    g ← g + 1
end while
```

The $(\mu + \lambda)$-EE presented in [25] is used with some modifications to address the optimal path adaptation problem. A known initial manipulator configuration vector $\overrightarrow{q_1}$ is assumed, as well as a robot base location and orientation at learned path p_l.

Consider an initial population of μ father individuals defined as in (1)

$$P_g = \{\Omega_{1,g}, \ldots, \Omega_{i,g}, \ldots, \Omega_{\mu,g}\},$$

where Ω_i represents a floating point vector with length size $T = N.n$ and $g = 0, \ldots, g_{max}$ the generation number. In the scheme $(\mu + \lambda)$-*EE* the initialization process generates a population of μ random individuals distributed within the vector parameter bounds. If the initial location is unknown, then the use of an uniform distribution would be advisable to ensure the diversity of the population. Further matter, if the initial joint configuration $\overrightarrow{q_1}$ is considered close enough to the learnt path initial node $(k = 1)$, we can intuitively assume that the optimal solution must be near learned path Ω_l. This first estimation is included in the initialization process $P_{g=0}$, as the learnt path perturbation with a Gaussian probability distribution at the configuration nodes $K = 2, \ldots, N$, reducing the convergence time. Therefore, the initialization process can be expressed as

$$\Omega_{i,g} = \begin{cases} \overrightarrow{q_1}, & \text{if } k = 1, \\ \overrightarrow{q_{l_k}} + rand_G(0, \sigma^2), & \text{if } k = 2, \ldots, N \end{cases} \quad (7)$$

Where $rand_G(0, \sigma)$ is a Gaussian distribution random number generator with zero mean and standard deviation σ, and $i = 1, \ldots, \mu$.

Once the population has been initialized, mutation is used to build a λ size offspring population. For each offspring, a parent is randomly selected, and each of its design variables b_i is mutated by adding a Gaussian random variable with zero mean and a standard deviation σ.

$$\Omega_{j,g} = \begin{cases} \overrightarrow{q_1}, & \text{if } k = 1, \\ \overrightarrow{q_{l_k}} + rand_G(0, \sigma^2), & \text{if } k = 2, \ldots, N \end{cases} \quad (8)$$

Where $j = 1, \ldots, \lambda$, the step length $\sigma = \sigma_{hi}$ during the position optimization phase and $\sigma = \sigma_{low}$ during the orientation optimization phase.

The objective evaluation function allows the assignment of a cost value to each member from parent and offspring populations: P_g and Po_g. The new parents P_{g+1} are selected from both populations as the μ individuals with the best fitness, that's to say with the lowest cost function.

$$\Omega_{i,g+1} = \begin{cases} \Omega_{j,g}, & if\ f(\Omega_{j,g}) \leq f(\Omega_{i,g}) \\ \Omega_{i,g}, & otherwise \end{cases} \quad (9)$$

The manipulator Direct kinematics defined by an homogeneous transformation matrix is calculated to obtain total cost function on (6). Homogeneous transform is a four by four elements matrix that contains end-effector location used for (4) and a rotation sub-matrix that is used to determine orientation error for (5), in this way reducing computational time is achieved by avoiding exact angles calculation.

Optimization loop begins by minimizing position error until a fitness value is reached. Then orientation error is added to the cost function, on first iterations the position error is incremented due to new added criterion influence, but then both criterion errors are gently improved as the generations evolve.

The only drawback in obtaining E_O from within the homogeneous transform is that calculated error in (5) is not directly proportional to angle error since it is the result of mathematical functions applied to the end-effector orientation angles; therefore it can't be used as a termination criterion. This issue is overcome by checking angles after position fitness is reached. Termination criterion takes into account only the position error until a fitness value is reached, then exact angles error is evaluated, if orientation fitness is not reached the position fitness value is reduced and minimization process continues. As both errors evolve together, orientation fitness is found eventually.

To ensure the generation of a feasible path, joint upper and lower limits need to be revised during optimization process. Joint limits are mechanical constraints that define the manipulator workspace, but also represent configuration values of reduced dexterity and hence should be avoided in the execution of the task. In the case of a boundary constraint violation, there are many solutions to replace the values that have exceeded their limits, [26]. Here a simple strategy is used, resetting the out-of-bound parameters with the exceeded bound value.

Finally mutation-selection process continues until convergence criterion is accomplished or a maximum generation reached.

IV. Simulations

The proposed methodology is tested in a simulation environment with a non-redundant mobile manipulator robot denominated MANFRED [27], which consist of a six degrees of freedom (n = 6) anthropomorphic arm mounted over a mobile base with two degrees of freedom (n=2). This robot was built in the Carlos III University of Madrid. Figure 1 shows the MANFRED robot in the developed 3D simulation environment; our laboratory was modeled with elements as doors and small tools to test grasping and manipulation tasks, simulations include bodies dynamics to increase realism and assure veracity.

The Denavit-Hartenberg parameters and joint limits for mobile manipulator MANFRED's robotic arm, are presented in Table I.

TABLE I
MANFRED robot Denavit-Hartenberg parameters and joint limits.

Art.	α_j	$a_j(m)$	θ_j	d_j	q_j^{ba}	q_j^{al}
1	90	0	0	0.25	-90	90
2	-90	0.4	0	0	0	180
3	-90	0	-90	0	-90	90
4	90	0	0	0.35	-90	90
5	-90	0	0	0	-90	90
6	0	0	0	0.25	-90	90

The Ω_l path describes our known task with $N = 6$ points, as shown in Figure 1. Forward Kinematic is calculated using [28], obtaining the end-effectors position and orientation $\{(x_k, y_k, z_k), (\phi_k, \theta_k, \psi_k)\}$ for each point $k = 1, 2, \ldots, 6$.

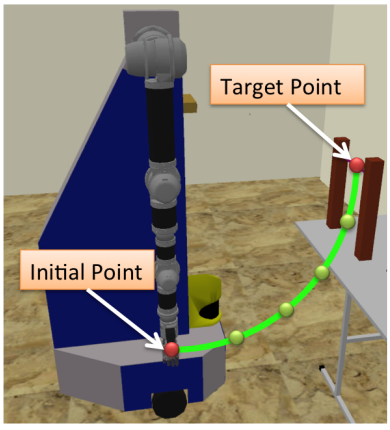

Fig. 1. Manipulation learned path Ω_l.

Table II shows the end-effector points coordinates for Ω_l. The robot location is given by the parameters (x_b, y_b, θ_b) referenced to a point predefined on the simulation map, where (x_b, y_b) determine the position coordinates in meters and θ_b the robot base orientation in degrees, position in z_b is not included since the robot base keeps the robotic arm at the same height all the time. For Ω_l, the location is $p_l = (-2.319, -2.138, 180°)$.

Learned path Ω_l is tested on the 3D simulation environment, results show how the robot reaches an experimental tool. Subsequently, this object is grabbed when the robotic hand is closed, and robot is capable to carry out by moving its base backwards.

TABLE II
End-effector learned path in Cartesian space.

k	X_k	Y_k	Z_k	ϕ_k	θ_k	ψ_k
1	250	147.63	-1000	$-180°$	$0°$	$-90°$
2	250	147.63	-980.62	$174.27°$	$17.09°$	$-108.86°$
3	250	284.83	-923.95	$156.88°$	$28.39°$	$-131.93°$
4	250	402.01	-834.35	$131.93°$	$28.39°$	$-156.88°$
5	250	491.23	-718.62	$108.86°$	$17.09°$	$-174.27°$
6	250	546.82	-585.47	$90°$	$0°$	$180°$

Two locations are used to verify the algorithm effectiveness: p_1 and p_2, these locations are different in position and orientation from that of the known path. Initial arm configuration $\vec{q_1} = \{0°, 0°, 0°, 0°, 0°, 0°\}$ is assumed for all cases, and the offspring population size is $\lambda = 30$ as proposed in Section II. The algorithm is executed 20 times for each location. Table III shows robot base locations for Ω_l, Ω_1 and Ω_2.

TABLE III
Position and orientation robot base coordinates for Ω_l, Ω_1 and Ω_2

Path	x(m)	y(m)	θ
Ω_l	-2.319	-2.138	$180.00°$
Ω_1	-2.294	-2.104	$181.48°$
Ω_2	-2.200	-2.207	$190.48°$

Once initial population members are generated, the execution of the algorithm starts looping to minimize (6). Position is optimized first with the configuration parameters: $F = 0.012$ and σ in accordance to (3). After fitness is reached, orientation is added to the objective function with σ reduced by a factor of ten; termination criterion is settled as end effector error less than 2.5 mm. Tests Results are shown in Table IV.

For an execution of the algorithm at p_1, a solution Ω_1 is found after $g = 120$ generations in $1,5$ seconds. In orientation terms an error of $2.98°$ is obtained on $N-1$ point, and $0.29°$ on last one. This makes sense when we recall the weighting factors for orientation optimization: $W_5 = 0.5$ and $W_6 = 1$. Lines described by the learned and evolutionary algorithm adapted path are shown on Figure 2, it can be seen that adapted path fits position closely with a soften adaptation curve when approaching to the known path; a position error of 1.78 mm is obtained in last node for this test execution. Results showed that intermediate points presented lower position errors and greater orientation errors than others because they are only optimized in position, while last two nodes presented minimal orientation error.

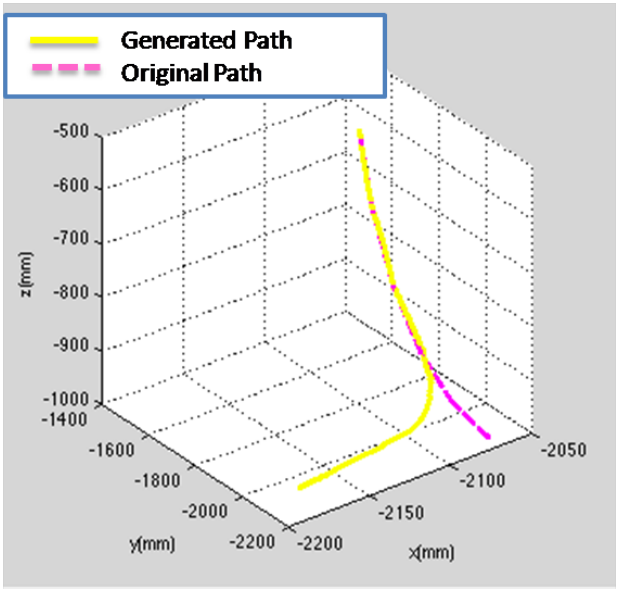

Fig. 2. Adapted and learned manipulation paths, Ω_1 and Ω_l.

All obtained manipulation paths are tested in the three-dimensional dynamic simulation environment where the robot executes the sequences correctly reaching the test tool every time. The object is carry out when additional steps are executed.

TABLE IV
PATH GENERARION STATISTICS FOR Ω_1 AND Ω_2.

	Generated paths:	
	Ω_1	Ω_2
Number of tests	20	20
Mean convergence generation	186.2	261.8
Best convergence generation	31	40
Worst convergence generation	500	500
Mean convergence time (seconds)	2.18	2.71
Best convergence time (seconds)	0.5867	3.92
Worst convergence time (seconds)	4.8736	4.8988
Mean position error (mm)	1.96	1.73
Min. position error (mm)	0.76	0.28
Max. position error (mm)	2.72	1.22
Mean orientation error	0.7831°	0.3703°
Min. orientation error	0.5679°	0.0059°
Max. orientation error	1.1180°	0.8448°

Experiment results shows that errors can be minimized so robot can achieve defined tasks. Time in worst-case scenario rose to $4,9$ seconds when reaching maximum generation $g_{max} = 500$, which stays within proposed real-time threshold. Further inquiries over our previous work [21] found that the forward kinematics calculus library consumed the most of the algorithm computational time. Therefore, an optimal computational complexity forward kinematics function was implemented reducing execution time significantly. When g_{max} is reached the fitness distance between the best and the worst of elements in population is taken as a convergence criterion, observe closely Figure 3 and 4.

On account of a reduced parent population size $\mu = 6$, lines on Figures 3 and 4 followed closely. Also an error peak when orientation optimization is began can be appreciated on Figure 4 around generation 20. This peak is less obvious on Figure 3 because robot base orientation error is smaller in that case.

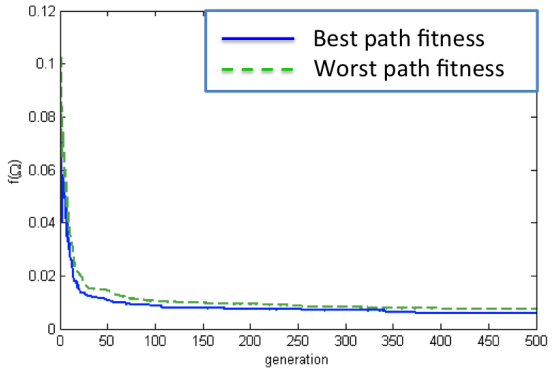

Fig. 3. Path fitness evolution for Ω_1 through g_{max} generations.

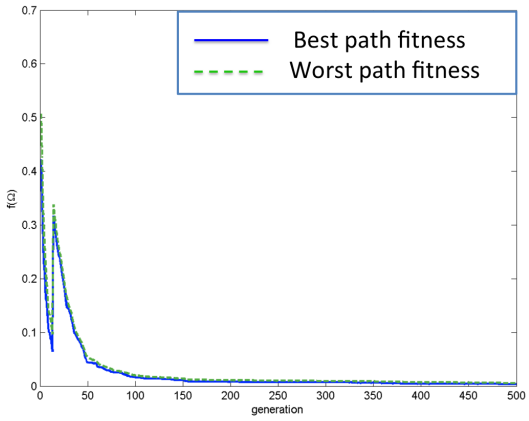

Fig. 4. Path fitness evolution for Ω_2 through g_{max} generations.

V. CONCLUSIONS

A methodology build over Evolution Strategies has been presented. The adaptation of manipulation paths for mobile manipulators is possible in real-time, achieving optimal manipulation due to position, orientation and energy consumption. Given a learned manipulation path a new one is calculated when robot base is in a different location from that of the learned path, minimal position and orientation end-effector errors are obtained within the evolutive process. Criterions and calculus are simplified to reduce computational time, overcoming Evolutionary algorithms time consumption problem. A forward kinematics function was implemented for reaching optimal computational time, it implied a significant time reduction for the execution of the algorithm. Granted

that the algorithm needs no inverse kinematics, singularities are avoided and convergence is guaranteed.

The experimental tests showed the ability to apply the algorithm in real-time for obtaining adapted manipulation paths, proving to be a feasible solution for mobile robots manipulation problems. A computational time improvement was obtained by first optimizing position until position error is minimized, then orientation error is added to the objective function with a reduced mutation step length until termination criterion is fulfilled at the end of the process. It is advisable to prioritize minimizing factors according to tasks because there is a proportional relationship between time and optimization parameters.

Additional development will be realized with other algorithms for performance comparison. Real robot tests will be conducted.

ACKNOWLEDGMENT

This work is funded by the project number DPI2010-17772 founded by the Spanish Ministry of Science and Innovation.

REFERENCES

[1] A. Goldenberg, B. Benhabib, and R. Fenton, "A complete generalized solution to the inverse kinematics of robots," *Robotics and Automation, IEEE Journal of*, vol. 1, no. 1, pp. 14–20, 1985. [Online]. Available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1086995

[2] K. Gupta and K. Kazerounian, "Improved numerical solutions of inverse kinematics of robots," *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, pp. 743–748, 1985. [Online]. Available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1087237

[3] R. Mayorga, A. Wong, and N. Milano, "A Fast Damped Least-squares Solution To Manipulator Inverse Kinematics And Singularities Prevention," *Intelligent Robots and Systems, 1992., Proceedings of the 1992 IEEE/RSJ International Conference on*, pp. 1177–1184, 1992. [Online]. Available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=594537

[4] S. Chiaverini, O. Egeland, and R. Kanestrom, *Achieving user-defined accuracy with damped least-squares inverse kinematics*. IEEE, 1991. [Online]. Available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=240676

[5] L.-C. Wang and C. Chen, "A combined optimization method for solving the inverse kinematics problems of mechanical manipulators," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 4, pp. 489–499, 1991. [Online]. Available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=86079

[6] E. Oyama, A. Agah, and T. Maeda, *Inverse kinematics learning by modular architecture neural networks with performance prediction networks*. IEEE, 2001. [Online]. Available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=932681

[7] L. Bao-Liang and K. Ito, "Regularization of inverse kinematics for redundant manipulators using neural network inversions," in *Proceedings of ICNN'95 - International Conference on Neural Networks*. IEEE, 1995, pp. 2726–2731. [Online]. Available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=488161

[8] A. S. Morris and A. Mansor, "Finding the inverse kinematics of manipulator arm using artificial neural network with lookup table," *Robotica*, vol. 15, no. May 2010, pp. 617 – 625, 1997.

[9] A. Guez and Z. Ahmad, *Solution to the inverse kinematics problem in robotics by neural networks*. IEEE, 1988. [Online]. Available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=23979

[10] S. Tejomurtula and S. Kak, "Inverse kinematics in robotics using neural networks," *Information Sciences*, vol. 116, no. 2-4, pp. 147–164, 1999. [Online]. Available: http://dx.doi.org/10.1016/S0020-0255(98)10098-1

[11] H. Thiang and R. Pangaldus, "Artificial Neural Network with Steepest Descent Backpropagation Training Algorithm for Modeling Inverse Kinematics of Manipulator," *World Academy of Science, Engineering and Technology*, pp. 671–674, 2009. [Online]. Available: http://fportfolio.petra.ac.id/user_files/97-031/v60-92.pdf

[12] S.-W. Kim, J.-J. Lee, and M. Sugisaka, *Inverse kinematics solution based on fuzzy logic for redundant manipulators*. IEEE, 1993. [Online]. Available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=583249

[13] A. Borboni, *Solution of the inverse kinematic problem of a serial manipulator by a fuzzy algorithm*. IEEE, 2001. [Online]. Available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1007317

[14] M. Yang, G. Lu, and J. Li, *An inverse kinematics solution for manipulators based on fuzzy logic*. IEEE, 2001. [Online]. Available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=983851

[15] K. Kumbla and M. Jamshidi, *Control of robotic manipulator using fuzzy logic*. IEEE, 1994. [Online]. Available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=343731

[16] W. Shen, J. Gu, and E. E. Milios, *Self-Configuration Fuzzy System for Inverse Kinematics of Robot Manipulators*. IEEE, Jun. 2006. [Online]. Available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4216772

[17] S. Tabandeh, C. Clark, and W. Melek, "A Genetic Algorithm Approach to solve for Multiple Solutions of Inverse Kinematics using Adaptive Niching and Clustering," *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, 2006. [Online]. Available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1688527

[18] J. Parker, A. Khoogar, and D. Goldberg, *Inverse kinematics of redundant robots using genetic algorithms*. IEEE Comput. Soc. Press, 1989. [Online]. Available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=100000

[19] W. Huapeng and H. Handroos, "Utilization of differential evolution in inverse kinematics solution of a parallel redundant manipulator," in *KES'2000. Fourth International Conference on Knowledge-Based Intelligent Engineering Systems and Allied Technologies. Proceedings (Cat. No.00TH8516)*. IEEE, 2000, pp. 812–815. [Online]. Available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=884170

[20] C. Gonzalez, D. Blanco, and L. Moreno, *Optimum robot manipulator path generation using Differential Evolution*. IEEE Congress on Evolutionary Computation, CEC, 2009. [Online]. Available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4983366

[21] C. Arismendi, M. Muñoz, D. Blanco, and L. Moreno, "Learning Evolutionary Strategy for a Mobile Manipulator in Imitation Learned Tasks," in *2010 International Conference on Technologies and Applications of Artificial Intelligence*. IEEE, Nov. 2010, pp. 203–209. [Online]. Available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=5695454

[22] P. A. Laplante, *Real-time systems design and analysis*, third edit ed. United States of America: Wiley-IEEE, 2004. [Online]. Available: http://books.google.com/books?id=1qDuaC17yxAC&pgis=1

[23] H. Xu and J. J. Collins, *Estimating the Odometry Error of a Mobile Robot by Neural Networks*. IEEE, Dec. 2009. [Online]. Available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=5381505

[24] H.-P. Schwefel, *Numerical Optimization of Computer Models*. John Wiley & Sons Ltd, 1981. [Online]. Available: http://www.amazon.com/Numerical-Optimization-Computer-Hans-Paul-Schwefel/dp/0471099880

[25] S. Datoussaid, O. Verlinden, and C. Conti, "Application of Evolutionary Strategies to Optimal Design of Multibody Systems," *Multibody System Dynamics*, vol. 8, no. 4, pp. 393–408, Nov. 2002. [Online]. Available: http://www.springerlink.com/content/p28831042v1rj641

[26] K. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization (Natural Computing Series)*. Springer, 2005. [Online]. Available: http://www.amazon.co.uk/Differential-Evolution-Practical-Optimization-Computing/dp/3540209506

[27] D. Blanco, S. A. Ansari, C. Castejón, B. López Boada, and L. E. Moreno, "MANFRED: Robot Antropomórfico De Servicio Fiable Y Seguro para operar En Entornos Humanos," *Revista Iberoamericana de Ingeniería Mecánica*, vol. 9, no. 3, pp. 33–48, 2005. [Online]. Available: http://www.uned.es/ribim/volumenes/Vol9N3Nov_2005/V9N3A03 Blanco.pdf

[28] J. Denavit and R. Hartenberg, "A kinematic notation for lower-pair mechanisms based on matrices," *Journal of Applied Mechanics*, vol. 23, no. June, pp. 215 – 221, 1955. [Online]. Available: http://www.citeulike.org/user/ghsalazar/article/1476529