An Optimal Geometric Model for Clavels Delta Robot

Carlo Alberto Avizzano, Alessandro Filippeschi, Juan Manuel Jacinto Villegas, Emanuele Ruffaldi
PERCRO, TeCIP Institute
Scuola Superiore Sant'Anna
Pisa, ITALY
n.lastname@sssup.it

***Abstract*—This paper discusses the Clavel's Delta parallel robot and proposes an alternate solution to its kinematics/dynamic model. We meant to integrate these models into on a small electrical driving circuit that integrates an onboard microcontroller. We designed the solution by taking into account the reduced computing capability of small embedded systems. Direct kinematics (DK), differential kinematics, both direct (J) and inverse (invJ), and a simplified dynamic model will also be presented. The novelty of the approach relies in a series of geometric properties that allow to reduce the computational load. When the three kinematics are computed together (DK, J, invJ), their computations can be expressed in few lines of code. The accuracy of motion, as well as the reduced computing power, will be compared to classic algorithms . The proposed algorithms have been implemented in a working system in the context of a telemedicine project.**

Keywords–Clavel's Delta; Embedded Controller; Kinematics Optimization;

I. Introduction

The Delta parallel robot was first introduced by Clavel in 1989 [1], [2]. It is a pure 3D translational mechanism, completely made trough rotational joints. This type of kinematics has numerous advantages with respect other 3 Degree of Fredom Devices (DOF) devices. In particular it allows to ground all the motors, thus minimizing the load of moving masses; it is intrinsically parallel thus allowing high forces at the end-effector; it preserves a very good compromise of workspace/device size. This device kinematics has been applied in numerous cases of industrial and research products, such as the Omega [3] haptic interface, and industrial [4], [5] and medical [6] applications .

Since its original design the mechanism received much attention in literature to optimize and make the design more flexible. An example of the computational methods for direct and inverse kinematics/dynamics of the Delta Robot was presented by Lopez [7]. A trace of the computational algorithm was presented by Zsombor and Murray in 2004 [8]. A possible variant, based on a recursive method was proposed by Staicu that proposed a dynamic modelling of the robot based on the inverse dynamic problem[9]. A comparison of screw theory based solutions for the gravitational compensation was presented in [10]. Another design variant for the optimization of kinematics was presented by Liu [11] where he consider how to optimize the design as a function of the prescribed performances.

A good summary of the techniques used for the computation of direct/inverse kinematics and dynamics is summarized in the works of Olson [12] and Williams [13].

The existing solutions to the kinematics always propose unique solution method, that consist in two steps: first, finding the spatial positions of the device elbows; and second, to find the point of intersection of three different spheres whose centers are in the elbows. In 2009 Murray and Zsombor considered again how to reduce the computational load, but without changing the overall methodology that still consisted in finding the intersection of the three spheres[14].

While from one side the computational power of desktop and industrial computers is progressively increasing, from the other side, Human Robot Interfaces (HRI), Wearable Robots (WR) and Internet of Things (IoT) are at they dawn, and can only rely to use embedded systems, with reduced computational capabilities.

For instance, recently we integrated the control of a 3DOF moving planar platform, named MOTORE [15] using and onboard small power computing DSP from Texas Instruments (a TI-TMS320F28335). This type of device can provide performances at about 100MHz, with only a rough single precision floating point support that executes much slowly than the base controller frequency.

Our objective is to fully integrate the electrical and software control of a Delta-Like interface, in order to achieve a fully-portable bio-medical system that only needs power supply for being interactive. An overview of the system operation is given in[16].

For this purpose we choose a similar architecture to the TI's one, an STM32F407 device from ST Microelectronics, which has floating point capabilities while supporting a 32 bit integers thanks to the internal ARM cortexM4F architecture [17].

To integrate on this microcontroller a controller which includes Kinematics and Dynamics of the system, the complete computational process has been reviewed. In what follow we will present the geometrical ideas at the basis of the new computational strategy, the optimization during the numerical implementation, and an evaluation of the performances on the on-board target.

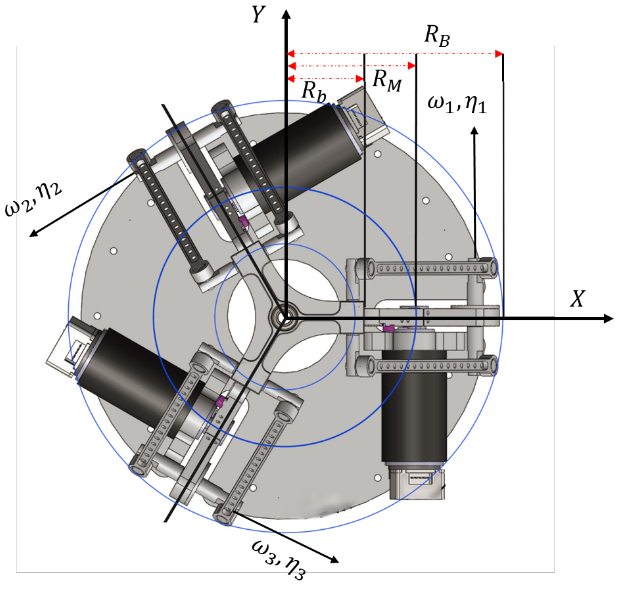

Figure 1. A top view of the device. Courtesy of ACCREA Ltd, adapted.

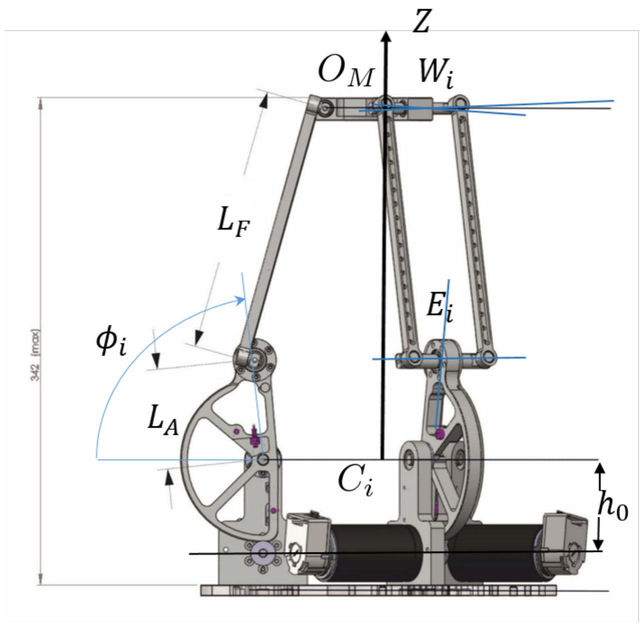

Figure 2. The side view of the device. Coutersy of ACCREA Ltd, adapted

II. Introduction to the Geometrical algorithm

The device we worked with has the mobile plate on its top side, and the fixed basis is on the bottom. This setup, upside-down with respect the original Clavel's delta, is being used with an handle on its mobile plate to allow operation of haptic exploration and teleoperation. With respect to the figure 1, the following relevant variables have been used. The three motors, capstans and legs have been numbered in a sequential order (1,2,3). We define a Cartesian base reference frame XYZ, in which XY are in the plane of the motor axes and the origin O_B is on the radial symetry axis of the device. The X axis is oriented from O_B to motor 1, and the Z axis is oriented upwards. We now define some geometrical quantities (see figure 1) that will be user later on: R_M is the radius of the circle tangent to the three capstans axes (Capstans' circle), we call C_i the points in which the circle is tangent to the i-th capstan. R_b is the radius of the moving plate, that is defined by the three centers of the pins which connect the forearms to the moving plate. Finally, R_B is the maximum radius of the Delta elbow joints that occurs when the elbows are horizontally aligned with the captan's rotation centers. We introduce three joint variables (namely ϕ_1, ϕ_2, and ϕ_3), whose zero value is set in this condition of maximum radius. The positiove direction of the three joint variables is highlighted in figure 2 by means of the velocity vectors (ω_1, ω_2, ω_3), and the related versors (η_1, η_2, η_3).

The description of the device is completed by the length of the lower limbs, the device arms indicated as L_A in figure 2 and the device forearms shown with the term L_F. Also we will define h_0 the vertical distance of the capstans-axes plane to the motors-axes plane; E_i (where $i \in \{1, 2, 3\}$) the position of the elbows' centers, and W_i the position of the wrists centers.

A summary of the construction parameters is summarized in Table I.

TABLE I
VALUES OF THE DEVICE GEOMETRIC PARAMETERS

Coupling plate radius (R_b)	42mm
Capstans' circle radius (R_M)	80mm
Maximum radius of encoumbrance (R_B)	150mm
Length of device arms (L_A)	70mm
Length of device forearms (L_F)	183mm

At first we focus on the Direct Kinematics (DK) that provides the pose of the moving plate with respect to the rigid base frame given the joint angles ϕ_i. This pose will be identified by a moving frame system whose axes are parallel to the axes of the base frame, and whose origin O_M is aligned to O_B along the Z axis when $\phi_1 = \phi_2 = \phi_3 = 0$.

To determine the position of the moving plate origin O_M, first we observe that this kinematics does not change if the coupling plate radius (R_b) and the capstans' circle radius are increased or diminished by the same quantity. Hence, to reduce the amount of variables in the realtime computation, we decided to work with an equivalent system in which the equivalent coupling plate radius is null, i.e. $\tilde{R}_b = 0$, and the equivalent capstans' circle radius is computed as $\tilde{R}_M = R_M - R_b$.

In order to minimize the number of quantities to be calculated in runtime, thus reducing the computational load, we initialized the η_i versors and the capstans' center positions

(CC$_\mathrm{i}$) as:

$$\eta_i = [\sin(\theta_i), \cos(\theta_i), 0]^T \quad (1)$$

$$CC_i = [\tilde{R}_M \cos(\theta_i), \tilde{R}_M \sin(\theta_i), 0]^T \quad (2)$$

where $\theta_{1..3} = \{0, 2\pi/3, -2\pi/3\}$.

The first part of our algorithm is quite standard and common to most existing direct kinematics solution. Using the geometrical information and the joint displacements, we compute the position of the three elbows as:

$$E_i = L_i + CC_i \quad (3)$$

where

$$L_i = [L_A \cos(\phi_i) cos(\theta_i), L_A \cos(\phi_i) sin(\theta_i), L_A \sin(\phi_i)]^T \quad (4)$$

provides also the arm direction.

Since most of the parameters are calculated offline, the realtime load is limited to the calculation of the goniometric functions of ϕ_i, the products in equation 4 and the sum in equation 3.

Also, having split the computation of the elbow position in two parts, we will benefit of the arm direction information that will reduce the amount of computation later. Given this result, classic solutions that can be found in literature require to find the position of the moving plate origin O_M as the intersection of three spheres, centered in E_1, E_2 and E_3 respectively, and having the same radius that is equal to the forearm length L_F. This procedure requires to write and solve a complete system made of three quadratic equations.

Instead we imagined to solve the same issue by considering that the three elbows belonging to a unique sphere centered in O_M and passing through all E_i (see figure 3. The search for this center is much easier thanks to the following properties. The intersection between this sphere and the plane define by the three elbow points is a circle. This circle is circumscribed to the elbow triangle. The position of the center should stay on the axis of the circle; and, finally, the offset on such axis can be easily determined by the Pitagora's theorem. As we will see, most of the information required to apply these steps will use the same numerical quantities, thus requiring minimal calculations.

Let us consider now the trinagle whose vertices are E_1, E_2 and E_3. We introduce a new non Cartesian reference system, whose origin is E_1, and whose axes are v_1, v_2 and v_3, that are respresented in figure 3 and defined as

$$\begin{array}{l} v_1 = E_2 - E_1 \\ v_2 = E_3 - E_1 \\ v_3 = v_1 \times v_2 \end{array} \quad (5)$$

Said O_c the position in the relative system of the circumscribed circle, we observe that is should be the intersection of the perpendicular bisectors of the triangle edges. Now, O_c solves equations 6.

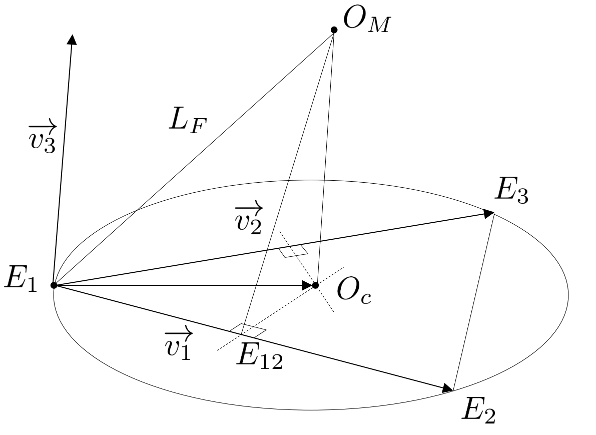

Figure 3. The geometry variables that we use to solve the device kinematics.

$$\begin{bmatrix} v_1^T \\ v_2^T \end{bmatrix} \begin{bmatrix} O_c - v_1/2 \\ O_c - v_2/2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad (6)$$

Therefore, after a simple algebraic manipulation and the use of the pseudoinverse, we obtain:

$$O_c = (v_1 C_{22}(C_{11} - C_{12}) + v_2 C_{11}(C_{22} - C_{12}))/(2\Delta) \quad (7)$$

where $C_{11} = |v_1|$, $C_{22} = |v_2|$, $C_{12} = v_1^T v_2$), and the determinant of the pseudoinverse was defined as $\Delta = C_{11}C_{22} - C_{12}C_{12}$.

The vertical offset may be determined on the v_3 axis by imposing the Pitagora's identity on the triangle described by (O_M, O_c, E_1). We have:

$$\gamma_3 = \sqrt{\frac{L_F^2 - O_c^T O_c}{v_3^T v_3}} \quad (8)$$

Where we expressed with γ_3 the coordinate along the v_3 versor of the moving plate in the reference system that we introduced in 5. This result leads us to:

$$O_M = E_1 + O_c + v_3\,\gamma_3 \quad (9)$$

III. Differential kinematics

Given the generic expression 10, we set the problem of the Direct Differential Kinematics (DDK) as solving equation 10 for Jd, whereas the goal of the Inverse Differential Kinematics is solving equation 11 for $Jinv$.

$$\dot{O}_M = Jd\,\omega \quad (10)$$

$$\omega = Jinv\,\dot{O}_M \quad (11)$$

being $\omega = [\omega_1 \omega_2 \omega_3]^T$.

The DDK and the DIK are computed together. Even in this case we determined the kinematics using a non conventional geometric derivation. First we considered that the forearm lengths do not change during motion, and hence, for each elbow (i), we have:

$$(O_M - E_i)^T(O_M - E_i) = L_F^2 \tag{12}$$

Which lead us to the following fundamental relationship:

$$(O_M - E_i)^T(\dot{O}_M - \dot{E}_i) = 0 \tag{13}$$

This formulation easily help us to describe the inverse differential kinematics. By replacing each $\dot{E}_i$ with its analytic representation,

$$\dot{E}_i = \eta_i \times L_i\omega_i \tag{14}$$

we have:

$$\begin{gathered}(O_M - E_i)^T\dot{O}_M = (O_M - E_i)^T(\eta_i \times L_i\omega_i) = 0 \\ \omega_i = \tfrac{(O_M - E_i)^T\dot{O}_M}{(O_M - E_i)^T(\eta_i \times L_i)}\end{gathered} \tag{15}$$

The equation 15 suggests us the generic row of $Jinv$ being computed as:

$$Jinv_i = \tfrac{(O_M - E_i)^T}{(O_M - E_i)^T(\eta_i \times L_i)} \tag{16}$$

The result of 16 is the only effective computation we need, all the terms in the equations were already computed offline and/or during the direct kinematics problem solution, therefore it does not requires additional new terms to be computed, and the overall computational load is minimal.

The direct kinematic my be then computed by superimposition of the effects. If we consider only one joint moving at a time, we can directly know the direction of motion related to this joint. If only joint three is moving, E_1 and E_2 will remain fixed. So the triangle (O_M, E_1, E_2) may only rotate along the axis described by v_1. This direction is computed as cross product between the v_1 axis and the line passing through the moving plate and the middle point between E_1 and E_2, hence

$$\tilde{J}d_3 = (O_M - E_{12}) \times v_1 \tag{17}$$

where

$$E_{12} = \frac{1}{2}(E_1 + E_2) \tag{18}$$

The tilde here highlight the fact that Jd only refers to the direction of the Jacobian and not the proper modulus.The same equation may be repeated for the first and the second row of the Jacobian. Once determined the Jacobian directions, we scale them by knowing the identity product between the direct and inverse Jacobian.

$$Jd \begin{bmatrix} k_1 & 0 & 0 \\ 0 & k_2 & 0 \\ 0 & 0 & k_3 \end{bmatrix} Jinv = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{19}$$

The equation 19 helps us to identify the scaling gains each row should be multiplied in order to find the exact direct Jacobian row:

$$Jd_i = \tilde{J}d_i/(Jinv_i\,\tilde{J}d_i) \tag{20}$$

Once again, we only use pre-computed values to determine the exact values of the direct Jacobian matrix.

The above computation can be implemented in about 20 lines of Matlab code.

IV. Simplified dynamic Model

In what follow we will produce a simplified dynamic models of the Delta Robot. In our simplified model, we will only neglect part of the inertial and gyroscopic and Coriolis effects due to the rotation of the device forearms. The numerical approximation is good for most cases in which the weight of the forearm is small and/or the speed of motion is limited.

The dynamic model can be split into two components: gravitational and inertial effects. In both case we proceed to the analysis of the dynamics by operating the following simplification. We considered the mass effects of the forearm, as being generated only by two lumped masses being placed on the elbow and on the wrist.

With such simplification (both masses equals exactly one half of the overall forearm mass), the overall weight is unchanged as well as the overall center of mass. As a result there would be no errors in computing the overall gravitational compensation.

To proceed in the dynamic model we off-line compute the following values:

- M_{UP} the overall mass of the moving plate, including the mass of the rotating joints and one half of the mass of the theee forearms;
- M_SH the overall mass of the shoulder joint which moves the arm, it includes the capstan mass, and one half of the mass of one forearm (for each joint);
- G_SH the position of the center of mass for the M_SH computed before;
- I_SH the overall inertia moment with respect the joint axis;
- τ_SH the (maximum) equivalent torque, produced on the capstan joint by the weight force when G_SH is vertically aligned with the capstan axis;
- ϕ_SH the displacement angle assumed by the capstan joint to achieve the vertical alignment described before.

All these values do not require any online computation. The gravitational compensation may be then assumed for each joint as the combined effect of the two torques applied by the moving plate ($\tau_M g$) and the "extended" capstan ($\tau_C g$) itself, we have:

$$\begin{aligned}\tau_M g &= Jd\,[0,0,M_{UP}g]^T \\ \tau_C g,i &= \tau_{SH}\cos(\phi_i - \phi_{SH})\end{aligned} \tag{21}$$

The Dynamic compensation can be computed by analyzing the inertia effects produced during motion. According to our simplification we have two types of torque effects, the inertia forces produced by the moving mass ($T_M d$) and the inertia forces produced by the capstan ($T_C d$).

$$\begin{aligned}\tau_M d &= Jd\,M_{UP}\ddot{O}_M \\ \tau_C g,i &= I_{SH}\ddot{\phi}_i\end{aligned} \tag{22}$$

Here the only simplification we introduced was replacing the forearm baricentral inertia (whose value for a uniformly distributed bar is $I_real = 1/12 M_F L_F^2$, with the inertia provided by two masses collocated at the end of the forearms:

$$I_{equiv} = 2\frac{M_F}{2}(\frac{L_F}{2})^2 = 1/4 M_F L_F^2$$

where we indicated with M_F the overall mass of the forearm. The relative error is therefore:

$$I_{err} = \frac{1}{6} M_F L_F^2$$

However considered the typical Delta structure we have the combination of three effects that made this approximation highly acceptable: first the parallel structure intrinsically limits the range of rotation of the forearms and therefore the value of their derivatives; secondly the forearm weights is usually much lighter than the capstans and the coupling plate value; third the mass distribution is not usually equally distributed since the presence of the bearings concentrate much of the weight in the elbow and wrist positions. The table II shows the case of the weight distribution in our Delta system.

TABLE II
SUMMARY OF THE WEIGHT PARAMETERS IN THE ACCREA'S DELTA

Coupling plate	274gr
Coupling plate bearings	3 x 26gr
Forearm weights	2 x 20gr (each)
Capstan weight	92gr (each)
Coupling capstan bearings	26 gr (each)

As we deduce from table II, the $1/6$ error is only in the forearm moving part (one sixth of 40gr). This error, when compared to the overall mass (40gr + 26gr +26gr = 92 g), provides an overall error close to 7%. If we compare this error to the mass of the moving mechanism (the moving plate), the resulting error is smaller than 2%.

In most applications, these errors are far below the noise provided by the estimation of the acceleration signal from the encoders position. Therefore it can be handled by traditional robust control techniques.

V. BENCHMARKING

First we tested accuracy of the provided algorithms. The code was validated our geometric implementation against the the one presented by Codourey in [18]. We choose about 45,000 points equally spaced in the whole range of the joint ranges. Both algorithms were implemented using numeric float representation. The averaged RMS error on the overall workspace was 15.24nm, with a maximum error of 51.84nm. Both values resulted well below the encoders resolution and close to the limit of representation for the single precision mathematics. Some manually checked cases revealed that this error was due to numerical approximation in the [18] algorithms than the our. For instance at input position $[0.1, 0.1, 0.1]$ for the joint coordinates, which is perfectly symmetric, we have:

- Codourey result: $[2.88379e-9, 0, 0.154976]$
- Our geometric result: $[0, 0, 0.154976]$

from which we concluded the error was introduced by numerical approximation in the three sphere intersection algorithm. A preliminary numerical estimation of the computational cost was performed on a core i7@3.3GHz machine. We executed 42M loops on the Kinematic computation all over the workspace. Tests showed the proposed Kinematic is about 15% more effective in the computation of the sole end effector position. [18] algorithms provided averaged computation time of 69nS against the average computing time of 59nS of the geometric algorithms.

We expect more benefits however from the differential kinematics and dynamics. These computations only require few more vector products, while the analytic approach requires again the computation of the lower legs velocities. An example is provided by Lopez [7] which determines the differential kinematics by computing the closure constraint between the base and the moving plate of the delta and then setting the differential to zero. A further simplification with an external product lead the Lopez approach to the same result as in 14. In our case however the introduction of the intermediate terms $(O_M - E_i)^T$ removes the need to perform computations of additional terms. A simple comparison between the estimated gravitational compensation load given by [4] against our model is 154 vs. 15 cycles (where we estimated 12 cycles per trigonometric function while we decomposed cosine in 21 into a MAC operation).

Our target platform is a STM32F407@168MHz. We estimated the computational load difference between the proposed algorithm and existing ones on the target processor, using the available cycle table from [17] and reported in table III.

We should note the ability of the M4F to execute in condensed cycles multiply, sign change and accumulation. Trigonometric-operation cycle have been neglected assuming they are in common between implementation and that developer and the compiler may process them using an ad-

TABLE III
CORTEX M4F, FPU INSTRUCTION SET CYCLES SUMMARY.

Operation	Cycles
Addition / Multiply	1
Mul.& Acc.	2
Divide / SQRT	14
Inverse Trig.	60

hoc lookup table. These operations were neglected because that would be the same in both approaches and would be ease through lookup table implementations. Moreover, the assignment operation have been neglected. Table IV reports the comparison, the last row of this table reports a weighted number of cycles using data in Table III. The table compares our algorithm with other two analytic methods available in literature [12], [4]. Inverse trigonometric is not available as an assembly instruction, its cost has been estimated as 60 cycles using as the template asin function available at *github.com/32bitmicro/newlib-nano-1.0*.

As we can see the small difference in computational load increases when we move to an embedded controller scenario were complex operations (sqrt and divide) are not possible at the a similar cost of elementary ones (multiply, accumulate and combination of them). The Total Cycle Count (TCC) is the estimated number of cycles and shows how the analytic method requires 269 cycles against 138 of the geometric method. The geometric algorithm is about twice more efficient than analytic one even only considering the forward kinematics only.

TABLE IV
COMPARISON OF CYCLES CONSUMED BY DIFFERENT APPROACHES: THE PROPOSED GEOMETRIC, AND EXISTING ANALYTIC METHODS.

Operation	Proposed method	method [12]	method [4]
Addition	8	3	35
Multiply	16	18	41
Divide	3	7	2
Mul.& Acc.	29	54	0
SQRT	1	3	1
Inv. Trig.	0	0	3
TTC	(138)	(269)	(298)

VI. CONCLUSION

In this paper we presented an alternative algorithm for the real-time computation of the Clavel's Delta parallel robot. The proposed algorithms reduces the amount of computation required while improving the robustness of the result even in presence of reduced precision mathematical engines.

The results have been tested on an embedded platform the relative controller has been embedded on. Overall results showed capabilities to manage the whole device from one single low cost microcontroller without introducing severe compromises in the control design.

ACKNOWLEDGMENT

This work has been carried out within the framework of the EU project REMEDI, grant number 610902, and the Tuscany Regional Project TAUM. The authors are grateful to the sponsor for the financial support received for the described development. Also the authors wish to acknowledge the project partners and the laboratory staff who kindly supported the conceptual discussion of the design and the technical work of development.

REFERENCES

[1] R. Clavel, "Une nouvelle structure de manipulateur parallèle pour la robotique légère," *Automatique-productique informatique industrielle*, vol. 23, no. 6, pp. 501–519, 1989.

[2] L. Rey and R. Clavel, "The delta parallel robot," in *Parallel Kinematic Machines*. Springer, 1999, pp. 401–417.

[3] E. J. Shahoian, B. M. Schena, and L. B. Rosenberg, "Haptic interface for laptop computers and other portable devices," Nov. 23 2004, uS Patent 6,822,635.

[4] F. Pierrot, C. Reynaud, and A. Fournier, "Delta: a simple and efficient parallel robot," *Robotica*, vol. 8, no. 02, pp. 105–109, 1990.

[5] T. Brogårdh, "Present and future robot control development—an industrial perspective," *Annual Reviews in Control*, vol. 31, no. 1, pp. 69–79, 2007.

[6] A. Frisoli, L. F. Borelli, C. Stasi, M. Bellini, C. Bianchi, E. Ruffaldi, G. Di Pietro, and M. Bergamasco, "Simulation of real-time deformable soft tissues for computer assisted surgery," *The International Journal of Medical Robotics and Computer Assisted Surgery*, vol. 1, no. 1, pp. 107–113, 2004. [Online]. Available: http://dx.doi.org/10.1002/rcs.12

[7] M. López, E. Castillo, G. García, and A. Bashir, "Delta robot: inverse, direct, and intermediate jacobians," *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, vol. 220, no. 1, pp. 103–109, 2006.

[8] P. Zsombor-Murray, "Descriptive geometric kinematic analysis of clavel's "delta" robot," *Centre of Intelligent Machines, McGill University, USA*, 2004.

[9] S. Staicu, "Recursive modelling in dynamics of delta parallel robot," *Robotica*, vol. 27, no. 02, pp. 199–207, 2009.

[10] D. Checcacci, E. Sotgiu, A. Frisoli, C. Avizzano, and M. Bergamasco, "Gravity compensation algorithms for parallel haptic interface," in *Robot and Human Interactive Communication, 2002. Proceedings. 11th IEEE International Workshop on*. IEEE, 2002, pp. 140–145.

[11] X.-J. Liu, J. Wang, K.-K. Oh, and J. Kim, "A new approach to the design of a delta robot with a desired workspace," *Journal of Intelligent and Robotic Systems*, vol. 39, no. 2, pp. 209–225, 2004.

[12] A. Olsson, *Modeling and control of a Delta-3 robot*. Citeseer, 2009.

[13] R. L. Williams, "The delta parallel robot: Kineatics solutions," online at http://www.ohio.edu/people/williar4/html/pdf/DeltaKin.pdf, April 2015.

[14] P. Zsombor-Murray, "An improved approach to the kinematics of clavel's delta robot," 2009.

[15] C. Avizzano, M. Satler, G. Cappiello, A. Scoglio, E. Ruffaldi, M. Bergamasco *et al.*, "Motore: A mobile haptic interface for neuro-rehabilitation," in *RO-MAN, 2011 IEEE*. IEEE, 2011, pp. 383–388.

[16] E. Ruffaldi, A. Filippeschi, F. Brizzi, J. M. Jacinto, and C. A. Avizzano, "Encountered haptic augmented reality interface for remote examination," in *3D User Interfaces (3DUI), 2015 IEEE Symposium on*. IEEE, 2015, pp. 179–180.

[17] J. Yiu, *The Definitive Guide to ARM® Cortex®-M3 and Cortex®-M4 Processors*. Newnes, 2013.

[18] A. Codourey, "Dynamic modeling of parallel robots for computed-torque control implementation," *The International Journal of Robotics Research*, vol. 17, no. 12, pp. 1325–1336, 1998.