Integral Action in First-Order Closed-Loop Inverse Kinematics. Application to Aerial Manipulators.

M. I. Sánchez, J.Á. Acosta* and A. Ollero

Abstract—The aim of this work is to design, analyze and test the behaviour of first-order CLIK (Closed-Loop Inverse Kinematics) manipulator algorithms under the influence of Cartesian integral error feedback. Although CLIK algorithms has been widely and successfully applied in robot manipulators with structured workspaces, in aerial manipulators the lack of structured workspaces is highly demanding for the control algorithm and some extra requirements are needed. Thus, in this work the standard proportional action in first-order CLIK algorithms is enhanced adding integral actions. Among others benefits, it provides a smoother behaviour needed for smart manipulation, zero steady-state tracking error, rejection to constant disturbances (like those of the numerical algorithm errors) and zero sensitivity to dc-gain uncertainties in stable systems. The theoretical achievement is corroborated with both simulations and experiments on a 7-DoF lightweight aerial manipulator. Experiments also include the integration with a SNS (Saturation in the Null Space) algorithm to deal with physical constraints, demonstrating a successful implementation and performance on a RTOS (Real-Time Operating System).

Index Terms—Inverse Kinematics, CLIK, aerial manipulation, redundant manipulator.

I. INTRODUCTION

Aerial Manipulators (AM) are a special class of flying machines formed by the join of Unmanned Aerial Vehicles (UAV) and manipulators capable of autonomous physical interaction with their environment (see e.g. [1], [2], [3], [4], [5] and [6]). The ARCAS project [7] is devoted to the design and development of a cooperating free-flying robot system for assembly and structure construction. In this project aerial robots with six and seven joint arms have been developed. One of the main differences between aerial manipulators and manipulators with fixed base is that in the former the manipulators workspace is changing over time. This fact is very demanding for the control algorithm, becoming a challenge to find a good trade-off between precision and computational requirements. Fig. 1 shows an outdoor example with an AM operating outdoors where one can figure out the expected performance and operation around manipulator workspace boundaries. Desired end-effector trajectories are generated during the manipulation tasks depending on sensed target position at each instant of time. In addition, obstacle avoidance makes the manipulation problem even more involved. Therefore, a proper inverse kinematics solution with singularity handling is necessary in these kind of applications. In fact, it is strongly needed smooth manipulator movements preventing significant platform perturbations and instability. In this work, it is assumed the AM is hovering with a multirotor controller as the one presented in [8]. The AM developed at University of SEville (AMUSE, Fig. 1) consists in a octoquad ( [8], [9]) mounting a Robai Cyton Gamma 1500 [10] 7-DoF lightweight manipulator. This arm has smart built-in Dynamixel [11] servos with their own electronics for a low-level PID position control. Depending on the model version, torque measurements and/or control are available, nevertheless, due to hardware limitations they are not accurate enough. However, in aerial manipulation the minimization of the flying platform weight is very important and hence despite the above mentioned drawbacks, these kind of servos are usually employed because their high specific torque (supplied torque over servo weight). Along the literature one can find many implementations of the CLIK algorithms with a proportional Cartesian error feedback in its first-order formulation (e.g.: [12], [13], [14], [15], [16], [17]) or proportional-derivative in its second-order formulation ( [13], [18], [19]). Moreover, the success of those algorithms is the general character of their formulations that takes into account from redundant/non-redundant manipulators to “random” Cartesian end-effector trajectories required in aerial manipulation tasks, in contrast with the typical predefined paths in the common structured industrial operation tasks. Therefore, the main goal of this paper is to show an enhanced inverse kinematics algorithm that solves the problem of manipulators control with smooth movements, accounting for manipulator limitations and Cartesian space operational constraints, all in absence of torque control capabilities. In fact, the hard physical constraints in AM make even more relevant the smooth tracking of the references. A schematic of the usual constraints is shown in the accompanying video where the base reference frame corresponds with the multirotor-arm linking point, the green planes to

Fig. 1. AMUSE performing outdoor grasping experiments.
the skids and lower rotor-blade planes, and blue surfaces represent design limits where end effector can operate. The paper is structured as follows: Section II briefly describes the CLIK algorithm and its problem formulation; Section III provides the algorithm with the integral action with its stability proof; Section IV presents a comparison with the standard CLIK approach showing the pros and cons; Section V shows its implementation and additional considerations to work in a real lightweight manipulator; finally, the paper is wrapped up with a conclusion section.

II. CLOSE-LOOP INVERSE KINEMATICS CONTROL

It is well known that the CLIK control can be formulated as an optimization problem. For the sake of completeness, in this section a brief description of this problem is included. Thus, let \( n \) and \( m \) be the joint space and Cartesian task space dimensions, respectively\(^1\). Thus, End-Effector (EE henceforth) Cartesian velocity vector \( \mathbf{v} \in \mathbb{R}^m \) is defined as

\[
\mathbf{v} := [x, y, z, \dot{\omega}_x, \dot{\omega}_y, \dot{\omega}_z]^T = [\mathbf{p}, \mathbf{w}]^T, \tag{1}
\]

where \( \dot{\mathbf{p}} \) represents the linear velocity vector and \( \mathbf{w} \) are its angular rate, both in EE frame with respect to base frame. The desired velocities are defined as \( \mathbf{v}_d = [\dot{x}_d, \dot{y}_d, \dot{z}_d]^T \). Let \( \mathbf{J} \in \mathbb{R}^{m \times n} \) denote the geometric Jacobian and \( \mathbf{\gamma} \in \mathbb{R}^n \) the joint angles vector, so that the direct kinematics equation for a general manipulator is governed by (see \cite{12, 20, 21})

\[
\mathbf{v} = \mathbf{J}\mathbf{\gamma}. \tag{2}
\]

The inverse kinematics for this redundant manipulator can be formulated as a (local) constrained linear optimization problem of an a priori given objective function. Thus, for a given end-effector velocity \( \mathbf{v} \) and a Jacobian \( \mathbf{J} \) we look for a solution on \( \mathbf{\gamma} \) as a result of the optimization problem defined by the quadratic cost functional on joint velocities given by

\[
\min_{\mathbf{\gamma}} \frac{1}{2}(\mathbf{\gamma} - \mathbf{\gamma}_0)^\top \Gamma(\mathbf{\gamma} - \mathbf{\gamma}_0), \quad \text{s.t.} \quad \mathbf{v} = \mathbf{J}\mathbf{\gamma}, \tag{3}
\]

where \( \mathbf{\gamma}_0 \) the homogeneous solution and \( \Gamma \in \mathbb{R}^{n \times n} \) is a positive definite weighting matrix. Necessary conditions for optimality together with Lagrange multipliers provide a solution of (3) (see \cite{12} for details) which is given by

\[
\mathbf{\dot{\gamma}} = \mathbf{J}^\top \mathbf{v} + \left( \mathbf{I}_n - \mathbf{J}^\top \mathbf{J} \right) \mathbf{\dot{\gamma}}_0 \tag{4}
\]

where \( \mathbf{J}^* := \Gamma^{-1} \mathbf{J}^\top [\mathbf{J}\Gamma^{-1}\mathbf{J}^\top]^{-1} \). Just for compactness and without any loss of generality, we choose \( \Gamma = \mathbf{I}_n \) and then \( \mathbf{J}^* \) becomes the Jacobian pseudoinverse, namely \( \mathbf{J}^\dagger \). Hence, the pseudo-inverse joint velocity (4) is the only one that minimizes the error joint velocity norm in (3). As it is well-known with (4), there is no guarantee that singularities are avoided during the task execution due to its local property, i.e. the equation (4) can be computed where the Jacobian \( \mathbf{J} \) is full rank and becoming useless at singular configurations. Among others, it is important to underscore that the computational problems become a serious drawback not only at singularities but also in the neighborhood of singularities. In fact, those Cartesian points (both position and orientation) can demand large joint velocities which might be unreachable by the manipulator (see \cite{20, 21}). The Damped Least Squares (DLS) pseudoinverse allows to overcome this problem with a good compromise between large joint velocities and Cartesian task accuracy, obtaining a robust behaviour close to singular configurations as shown in \cite{18, 22, 13, 12} or \cite{23}. The DLS algorithm is a modified pseudoinverse given by

\[
\mathbf{J}^\dagger_{DLS} := \mathbf{J}^\top (\mathbf{J}\mathbf{J}^\top + \kappa^2 \mathbf{I}_n)^{-1} \tag{5}
\]

where \( \kappa \) is a variable damping factor which has a maximum value, \( \kappa_0 \), at singular configurations and zero away from them. It is clear that \( \kappa \) must be a manipulability-like measure because \( \sqrt{\det(\mathbf{J}\mathbf{J}^\top)} \) is null at singular points and takes small values in its neighbourhood \cite{12}. To smooth that dependence including the neighbourhood the damping factor is shaped with a Gaussian function in the following way

\[
\kappa := \kappa_0 \exp\left(-\frac{\det(\mathbf{J}\mathbf{J}^\top)}{2 \varepsilon^2}\right),
\]

where the maximum value \( \kappa_0 > 0 \) and the shaping factor \( \varepsilon \in (0, 1) \) are design parameters. In the next section we first provide a design of the algorithm with the integral of the Cartesian error and then conditions for its stability and convergence are established.

III. MAIN STABILITY RESULT

The numerical implementation of (4) and (5) implies the inversion of \( \mathbf{J} \) evaluated at previous instants of time. This particular computer implementation provides numerical errors involving drift phenomena of the solution, which, in turn, causes a mismatch in the EE pose. To overcome this inconvenience, a numerical inverse kinematics algorithm that accounts for the Cartesian error and uses a feedback correction term is implemented. Therefore, it will be necessary a proper formulation of errors between EE desired –d sub-index– and current Cartesian positions and orientations. As it was highlighted by \cite{24} this problem is non-trivial since unlike the translational velocities, the angular rates are not orientation time derivative. Additionally, the non-uniqueness of standard angular representation, as the Euler angles, becomes also a drawback for numerical algorithms. It is well-known that a efficient way to overcome those difficulties is to select the unit quaternions as generalized coordinates and express the orientation error through them (see \cite{25}). Thus, let \( q = [q_0, \mathbf{q}]^\top \in \mathbb{R}^4 \) denote a quaternion with \( q_0 \in \mathbb{R} \) and \( \mathbf{q} \in \mathbb{R}^3 \) its scalar and vector components, respectively, and let \( q^* = [q_0, -\mathbf{q}]^\top \) be its conjugate which, in turn, for unit quaternions, i.e. \( ||q||^2 = 1 \), coincides with its inverse. Then the orientation error, namely \( \Delta q \), is expressed along the quaternion product \(^2\) as (see \cite{24, 26, 27})

\[
\Delta q := q_d \ast q^* = [\Delta q_0, \Delta \mathbf{q}]^\top
\]

\[
= [q_{0d} q_0 + \mathbf{q}_d \times \mathbf{q}_0, q_0 q_{0d} - q_{0d} q_0 - S'(\mathbf{q}_d) \mathbf{q}_0] , \tag{6}
\]

\(^1\)In particular, for our manipulator \( n = 7 \) and \( m = 6 \) (position and orientation).

\(^2\)Let \( q_i := [q_{i0}, \mathbf{q}_i]^\top, i = 1, 2, \) then \( q_1 \times q_2 = (q_{10} q_{20} - q_{20} q_{10}, q_{11} q_{20} - q_{20} q_{11}, q_{12} + q_{20} q_{12} + q_{12} q_{20} - q_{20} q_{12}) \) where \( \times \) stands for the cross product of vectors \( \mathbf{q}_i \).
where $S$ is the skew-symmetric cross-product matrix and $q_d := [q_{d_0}, q_d]^T$ the desired quaternion. Note that $\Delta q = [1, 0]^T \Rightarrow q_d = q$ providing the uniqueness property desired. In view of the unit quaternion norm property the orientation error, namely $e_o$, can be only defined with the vectorial part of (6), i.e. $\Delta q$ (see [25], [12], [27]). Hence, in what follows, we denote the Cartesian error vector as

$$e := \begin{bmatrix} e_p \\ e_o \end{bmatrix} := \begin{bmatrix} p_d - p \\ \Delta q \end{bmatrix} \in \mathbb{R}^6,$$

(7)

where $e_p$ and $e_o$ are the position and orientation error, respectively. Note that current orientation is joint-space dependant, i.e. $q = q(\gamma)$, and hence, for a given manipulator joint configuration the actual EE quaternion must be obtained through its rotation matrix, say $R \in \mathbb{R}^{3 \times 3}$, computed via direct kinematics.

The following proposition shows the error dynamics of (7) in terms of the orientation error and angular rates by making use of the relationship between angular rates and quaternions, that is, $\dot{e} = \dot{e}(\Delta q, \omega_d, \omega)$. 

**Proposition 1:** The dynamics of the error (7) yield

$$\dot{e} = F v_d - GJ \dot{\gamma},$$

(8)

with the $(6 \times 3)$-block matrices $F$ and $G$ defined as

$$F := \text{diag}\left(I_3, \frac{1}{2} A\right), \quad G := \text{diag}\left(I_3, \frac{1}{2} B\right),$$

(9)

where $A := \Delta q_0 I_3 - S(\Delta q)$, and $B := \Delta q_0 I_3 + S(\Delta q)$. 

**Proof:** First, we relate $e_o$, computed via quaternion product, with $\omega$ from (1). For that, let $\omega \in \mathbb{R}^3$ be represented by the pure quaternion $\Omega := [0, \omega]^T$ and then, the quaternion differential equation $\dot{\Omega} = 2q^* \dot{q}^*$ holds (see [27], [28]). With the latter equation and the quaternion product properties the error dynamics become

$$\dot{e}_o = \begin{bmatrix} 0_{3 \times 1} & I_3 \end{bmatrix} \left(\dot{q}^* \dot{q} + q^* \dot{q} \right)$$

$$= \frac{1}{2} \begin{bmatrix} 0_{3 \times 1} & I_3 \end{bmatrix} \left(\Omega_d \times \Delta q - \Delta q_d \times \Omega \right)$$

$$= \frac{1}{2} \left(A \omega_d - B \omega\right),$$

(10)

where matrices $A$ and $B$ were defined in the proposition. Finally, the error dynamics (8) is obtained lumping the position error dynamics $e_p = p_d - p$ together with the orientation error dynamics (10) accounting (1) and (2) along with the definition of the matrices $F$ and $G$ from (9).

**Proposition 1** allows to establish the main result below, where the integral action is included in the core algorithm (4).

**Proposition 2:** Let $\xi \in \mathbb{R}^m$ be defined as $\xi := \dot{e}$ and assume det$(JJ^T) \neq 0$. Consider the error system (8) along with $\dot{\gamma}$ given by (4) and $v$ from (2) defined as

$$v := G^{-1} \left[F v_d + K e + Q^T \dot{\xi} \right],$$

(11)

where $K, Q \in \mathbb{R}^{m \times m}$ are constant matrices. Then, the closed-loop inverse kinematic system becomes

$$\ddot{\xi} + K \ddot{\xi} + Q^T \dot{\xi} = 0.$$  

(12)

Moreover, fix $K := \text{diag}(K_p, K_o)$ and $Q := \text{diag}(Q_p, Q_o)$ with $K_p, K_o \in \mathbb{R}^{3 \times 3}$ positive definite and diagonal matrices and $Q_p := e_p K_p^2$ and $Q_o := e_o K_o^2$, for any constants $e_p, e_o \in (0, 1]$. Then, the zero equilibrium of (12) is (globally) asymptotically stable.

**Proof:** The first claim is straightforward by plugging (11) in (4) and the resulting $\dot{\gamma}$ in (8). For the second claim, let us rewrite the (12) as the first-order system given by

$$\dot{\xi} = \begin{bmatrix} 0 \\ -Q^T \dot{\xi} - K \end{bmatrix} \xi - \begin{bmatrix} e \\ e_o \end{bmatrix},$$

(13)

Define the Lyapunov function candidate as

$$V := \frac{1}{2} \begin{bmatrix} \xi^T \\ e^T \end{bmatrix} \begin{bmatrix} P + Q \end{bmatrix} \begin{bmatrix} \xi^T \\ e^T \end{bmatrix},$$

(14)

with $P \in \mathbb{R}^{m \times m}$ a positive definite matrix to be defined. The time derivative of (14) along the trajectories of (13) becomes

$$\dot{V} = -\xi^T Q^T \xi + \xi^T P (P - 2QK)e$$

(15)

Since $K, Q$ are diagonal and positive definite, we choose $P$ to cancel the cross term out in (15), i.e. $P := 2QK$, and then to force $V \geq 0$ the matrix $P$ must verify the inequality

$$0 < P < 4K^3.$$  

(16)

Additionally, to guarantee $\dot{V} \leq 0$ from (15) $P$ must verify

$$Q Q^T > 0 \Rightarrow PK^{-2}P > 0,$$

$$K^{-2}Q^T > 0 \Rightarrow P < 2K^3.$$  

(17)

Defining the matrix $P := \text{diag}(P_p, P_o)$ with $P_p := 2\epsilon_p K_p^3$ and $P_o := 2\epsilon_o K_o^3$ and $Q$ as stated in the proposition, the constraints (16) and (17) are satisfied. Hence, $V > 0$ and $\dot{V} < 0$ for $(\xi, e) \neq 0$ and then the zero equilibrium is (globally) asymptotically stable, concluding the proof.

**Remark 1:** The computation of (11) does not involve any matrix inverse calculations but the $\Delta q$ rotation matrix. Indeed, it is easy to check that $B^{-1}A = R^T(\Delta q)$ and, in a similar way, $B^{-1} = \frac{1}{2m} [R^T(\Delta q) + I_3].$

**Remark 2:** Notice that, the already known first-order CLIK algorithm is a particular case for $\epsilon_p = \epsilon_o = 0$ of the extended algorithm provided, (see [13], [12]).

The stability result of Proposition 2 relies on the assumption of a full-rank Jacobian in the whole work space, i.e. stay away from singularities. To overcome this assumption we compute the aforementioned DLS pseudoinverse algorithm instead. To this end, combining (4), (5) and (11) yield

$$\dot{\gamma} = J^T\text{DLS} (v_d + eK^2\dot{\xi} + Ke) + (I_n - J^T\text{DLS}J) \dot{\gamma}_0,$$

which can be rearranged as

$$\dot{\gamma} = \dot{\gamma}_0 + J^T\text{DLS} (v_d + eK^2\dot{\xi} + Ke - J\dot{\gamma}_0),$$

(18)

and where we have defined defining $e := \text{diag}(\epsilon_p I_3, \epsilon_o I_3)$. The latter algorithm avoids some waste of information due to the projection matrix $(I_n - J^T\text{DLS}J)$ has dimension $n \times n$ but only a maximum rank of $n - m$.
TABLE I
GAINS TABLE

<table>
<thead>
<tr>
<th>Gain</th>
<th>With I.A.*</th>
<th>Without I.A.</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K_p$</td>
<td>15</td>
<td>10</td>
</tr>
<tr>
<td>$K_o$</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>$e_p$</td>
<td>0.4</td>
<td>0.0</td>
</tr>
<tr>
<td>$e_o$</td>
<td>0.7</td>
<td>0.0</td>
</tr>
</tbody>
</table>

* “I.A.” stands for Integral Action.

IV. SIMULATIONS

In order to analyze and compare the behaviour of (18) with respect to the traditional approach, several simulation tests were performed. Matlab/Simulink [29] was employed both using a 200Hz and 50Hz fixed-step discrete solver. The algorithm implementation for the simulations followed a MBD (Model-Based Desing) method in order to export these algorithm in a embedded hardware running a RTOS for experiments. Choosing a position gain $K_p$, matrix $K_p$ can be built as $K_p := K_{p1}$ and the same goes for $K_o$, choosing an orientation gain $K_o$. Table I collects the numerical values employed. Both algorithms gains were tuned to ensure their best performance. For the sake of briefness, a representative test is presented were the EE must follow a continuous-time reference.

Additionally, for a redundant manipulator, the homogeneous solution can be used to optimize a secondary task following the projected gradient method (see [12], [30]), such a minimization of the distance from the mid points of the joint ranges. In particular, if the mid ranges are as the minimization of the distance from the mid points of the joint ranges. In particular, if the mid ranges are $[10, 4, 10, 4, 10, 1, 1]^T$ a weighting vector and $\gamma_{\text{max}}/\gamma_{\text{min}}$ each joint maximum/minimum bound angles.

Finally, the desired velocities can be computed to move the EE with linear paths at constant Cartesian linear and angular speeds, namely $V_d$ and $\omega_d$, as

$$\dot{\mathbf{p}} := V_d \frac{\mathbf{e}_p}{\|\mathbf{e}_p\|},$$

$$\dot{\mathbf{\omega}} := \omega_d \frac{\mathbf{e}_o}{\|\mathbf{e}_o\|},$$

with $V_d$ and $\omega_d$ fixed to 0.25m/s and 10deg/s respectively.

In the step case, both algorithms may not converge properly if the objective point is too far away from the starting point (about ~0.3m distance in this case) although the approach here presented seemed to be more robust. Nevertheless, this is not a problem since typically EE references are continuous trajectories. Fig. 2 shows both algorithms behaviour following a linear path in position and orientation accomplished in two seconds between

* Start: $x = [0.369m, 0.000m, 0.334m, 80^\circ, 0^\circ, 90^\circ]^T$
* End : $x = [0.000m, 0.000m, 0.700m, 0^\circ, 0^\circ, 0^\circ]^T$

IV. SIMULATIONS

In order to analyze and compare the behaviour of (18) with respect to the traditional approach, several simulation tests were performed. Matlab/Simulink [29] was employed both using a 200Hz and 50Hz fixed-step discrete solver. The algorithm implementation for the simulations followed a MBD (Model-Based Desing) method in order to export these algorithm in a embedded hardware running a RTOS for experiments. Choosing a position gain $K_p$, matrix $K_p$ can be built as $K_p := K_{p1}$ and the same goes for $K_o$, choosing an orientation gain $K_o$. Table I collects the numerical values employed. Both algorithms gains were tuned to ensure their best performance. For the sake of briefness, a representative test is presented were the EE must follow a continuous-time reference.

Additionally, for a redundant manipulator, the homogeneous solution can be used to optimize a secondary task following the projected gradient method (see [12], [30]), such as the minimization of the distance from the mid points of the joint ranges. In particular, if the mid ranges are $[10, 4, 10, 4, 10, 1, 1]^T$ a weighting vector and $\gamma_{\text{max}}/\gamma_{\text{min}}$ each joint maximum/minimum bound angles.

Finally, the desired velocities can be computed to move the EE with linear paths at constant Cartesian linear and angular speeds, namely $V_d$ and $\omega_d$, as

$$\dot{\mathbf{p}} := V_d \frac{\mathbf{e}_p}{\|\mathbf{e}_p\|},$$

$$\dot{\mathbf{\omega}} := \omega_d \frac{\mathbf{e}_o}{\|\mathbf{e}_o\|},$$

with $V_d$ and $\omega_d$ fixed to 0.25m/s and 10deg/s respectively.

In the step case, both algorithms may not converge properly if the objective point is too far away from the starting point (about ~0.3m distance in this case) although the approach here presented seemed to be more robust. Nevertheless, this is not a problem since typically EE references are continuous trajectories. Fig. 2 shows both algorithms behaviour following a linear path in position and orientation accomplished in two seconds between

* Start: $x = [0.369m, 0.000m, 0.334m, 80^\circ, 0^\circ, 90^\circ]^T$
* End : $x = [0.000m, 0.000m, 0.700m, 0^\circ, 0^\circ, 0^\circ]^T$

IV. EXPERIMENTS

Even though the integral action improves the classical first-order CLIK algorithm, there are some issues concerning the physical and operational properties of each manipulator. Indeed, the control of a manipulator –specially in aerial manipulation tasks– must take into account the maximum and minimum joint angles and rates, Cartesian constraints (obstacle avoidance), etc. Moreover, the secondary task accomplished following the projected gradient method does not guarantee the compliance of that task. This is exhibited in Fig. 2, where joints angles are far away from their mid range value (zero degrees) in order to minimize the Cartesian error, hence the secondary task expressed in (19) its not as effective as expected. Increase the $\mu_0$ values would be an option, but it would incur in a lose of accuracy in both algorithms, which should be solved increasing Table I gains. Once done that, (19) would be as inefficient as before. For that reason, the algorithm showed in (18) was modified as follows to apply the Saturation in the Null Space (SNS) method described in [31] and [32].

$$\dot{\gamma} = \dot{\gamma}_0 + (J_{DLS}^{-1})^T [s (v_d + \epsilon K_2 \xi + Ke) - J \dot{\gamma}_0]$$

where $W$ is the diagonal matrix with $W_{ii} = 1/0$ to specify which joints are enabled or disabled at each sample time and $s$ the scaling factor. $\dot{\gamma}_0$ is determined at each step time to accomplish the requirements listed before. Multiple Cartesian constraints –skids planes and bottom rotor blade planes– are also included as proposed in [31]. Since the arm operates on a multicopter, a state machine is also included to manage the manipulator in all the possible working modes. Other functionalities are added such as a EE workspace limiter –limits the operational space where the EE can move, blue surfaces in Fig. 2– and an equivalent rigid body properties computation –it computes arm centre of mass and equivalent inertia matrix to use it for feedback to the multirotor controller. The lightweight manipulator is actuated by Dynamixel servos, whose inputs/outputs are the joint angles and rates. As it was highlighted in the
introduction, control torque its not accurate enough, hence smooth movements generated by the kinematic arm contro
conformed to the physical and operational limits of the arm will contribute to better aerial manipulator control tasks. The control algorithm is programmed in Simulink for subsequent C code generation. This code is built in a RTOS QNX Neutrino [33] that will run as an embedded application on the AMUSE hardware system along a integral multirotor backstepping control ( [9], [8]). The Fig. 3 corresponds to an
eperiment showing the arm control behaviour representing the desired end-effector position and the commanded one to the manipulator. The Cartesian positions and orientations references are given using a joystick input. It can be shown how the manipulator follows the references smoothly and accordingly to the limitations and the requirements of the real platform since algorithm (green) and arm responses (red) are coincident. They differ only when arm operates in other state machine working mode (e.g. unfold/fold sequences, as
shown at the beginning and end of Fig. 3). In addition, notice on the workspace limiter effect when the desired reference is outside the established limits. Also, when the desired point is unreachable (e.g. outside of the arm range) can be shown the effect of the task scaling factor (s). The provided video shows these experiments both in real platform and virtual environments. Note AM vibrations due to its weight (~ 11kg) and the bench mounting available. Also, some preliminary flight tests are shown working this algorithm alongside an attitude back-stepping controller.

VI. CONCLUSIONS

In this paper, we have gone a step forward on CLIK algorithms including a pure integral action. We showed a formal treatment and provided a formal stability proof. A practical analysis of the benefits –widely used in other areas of Robotics– of an integral error feedback action in these algorithms has been presented –both in simulations and experiments– applied to the AM shown in Fig. 1, corroborating the theoretical claims. Experiments also include the integration with a SNS algorithm to deal with physical constraints, and its interaction with an attitude back-stepping controller demonstrating the successful implementation and performance on the AM embedded hardware. Aerial manipulation is highly demanding for control algorithms and, together with the lack of structured workspaces in outdoors, this control solution provides promising results in absence of torque control capabilities.

ACKNOWLEDGMENTS

This work is supported by the ARCAS project (Aerial Robotics Cooperative Assembly System) funded by the European Commission under the Seventh Framework Programme (http://www.arcas-project.eu/) and the CLEAR Project, funded by the Spanish government.

REFERENCES

[29] “Mathworks Matlab Simulink,”
http://www.mathworks.co.uk/products/simulink/.
[33] “QNX Neutrino RTOS,”