2019 International Conference on Robotics and Automation (ICRA)
Palais des congres de Montreal, Montreal, Canada, May 20-24, 2019

Pose and Posture Estimation of Aerial Skeleton Systems for Outdoor Flying

Sangyul Park, Yonghan Lee, Jinuk Heo and Dongjun Lee

***Abstract*— We present a novel pose and posture estimation framework of aerial skeleton system for outdoor flying. To exploit redundant/independent sensing while rendering the system "modular", we attach an IMU (inertial measurement unit) sensor and a GNSS (global navigation satellite system) module on each link and perform SE(3)-motion EKF (extended Kalman filtering). We then apply the kinematic constraints of the aerial skeleton system to these EKF estimates of all the links through SCKF (smoothly constrained Kalman filtering), thereby, enforcing the kinematic coherency of the skeleton system and, consequently, significantly enhancing the estimation accuracy and the control performance/stability of the aerial skeleton system. A semi-distributed version of the obtained estimation framework is also presented to address the issue of scalability. The theory is then verified/demonstrated with real outdoor flying experiments and simulation studies of a three-link aerial skeleton system.**

I. Introduction

Multi-rotor UAVs (unmanned aerial vehicles) or, often simply dubbed as drones, have received substantial interests from research community, industry and general public alike. Some of its representative/promising applications include aerial photography and surveying [1], pesticide spraying [2], entertainment [3] and aerial manipulation/operation [4]–[9]. Recently, a very unique and versatile drone-based aerial robotic system has been emerging, namely, *aerial skeleton system*, which consists of multiple articulated links, each purely actuated by multiple asymmetrically-attached distributed rotors; or with some motors to rotate the rotors w.r.t. the link and/or to directly produce inter-link relative motion. The LASDRA (large-size aerial skeleton with distributed rotor actuation) system [10], which adopts the ODAR (omni-directional aerial rotor) [7], [11] as its constituent link fully-actuated in SE(3) and connects them via cables with some compliance, is included in the former class (see Fig, 1), also containing the Hiryu-I system [12] which is constructed with 1-DOF (degree-of-freedom) parallel links actuated by two thrusters attached on each link. On the other hand, the DRAGON (dual-rotor multi-link robot with ability of multi-DOF aerial transformation) system [13] utilizes servo-motors

Research supported by the Basic Science Research Program (2015R1A2A1A15055616) of the National Research Foundation (NRF) funded by MSIT; by the Industrial Strategic Technology Development Program (10060070) funded by MOTIE; and also supported by a grant to Bio-Mimetic Robot Research Center funded by Defense Acquisition Program Administration, and by Agency for Defense Development (UD130070ID), Korea

The authors are all with Department of Mechanical & Aerospace Engineering and IAMD, Seoul National University, Seoul, Republic of Korea. {sangyul,ldragonfly,pooki09,djlee}@snu.ac.kr. Corresponding author: Dongjun Lee.

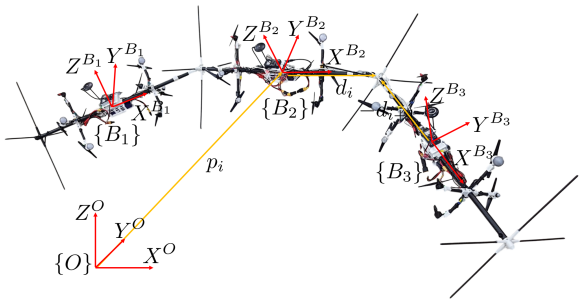

Fig. 1. Aerial skeleton system: three link LASDRA system for outdoor flying. Also shown are the inertial and body coordinate frames, $\{O\} := \{X^O, Y^O, Z^O\}$, $\{B_i\} := \{X^{B_i}, Y^{B_i}, Z^{B_i}\}$.

to generate rotor-link and link-link motions, belonging to the latter class. These aerial skeleton systems are envisioned to realize such new applications as mechanical operations at high-rise building or in a narrow/long space; or articulated flying characters in amusement parks.

In this paper, we consider the onboard pose and posture estimation problem of this LASDRA system [10] particularly for its outdoor flying (see Fig. 1). This problem is challenging for the following reason. First, to maximize system dexterity, the LASDRA system connects two links via a cable. This then allows for full 3-DOF inter-link rotation, which is difficult to measure by (accurate, yet, axis-demanding) encoders, and instead, IMUs (inertial measurement units) are more suitable (or often only viable) option for that as so for our LASDRA system. This IMU however exhibits non-negligible (yet, still bounded) link absolute attitude estimation error, which can be accumulated to a rather very large link position estimation error as propagated toward the end of the aerial skeleton system. This can pose a serious problem, since, e.g., if the skeleton is supposed to fly with a certain posture, this desired behavior essentially needs to be decoded into the position and attitude controls of each link, and, with the (accumulated) link position estimation error becoming very large (e.g., skeleton with large number of links), the link position control can be excessively erroneous, which may contradict/conflict with the attitude/position control of its own or other neighboring links and possibly result in collapsing and falling-down of the skeleton (with excessive internal force at some joints) particularly with the typical rotors prone to saturate. We may attach a GNSS (global navigation satellite systems) module on each link to correct its position estimation error, which is still not so promising either, since the accuracy of typical GNSS is too poor (i.e., meter-level accuracy) to reclaim the kinematic coherency of the aerial skeleton system as stated above.

To overcome this challenge, in this paper, we propose a novel pose and posture estimation framework for the aerial skeleton system based on IMU and GNSS sensors for its outdoor flying. More precisely, we attach an IMU sensor and a GNSS module on each link and estimate their state via standard SE(3)-motion EKF (extended Kalman filtering) [14], [15]. We choose this (distributed) sensor configuration to render the skeleton system "modular" while also significantly enhancing position sensing accuracy with redundant/independent GNSS sensors (via the constraints - see Sec. V). We then apply the kinematic constraints of the aerial skeleton system (i.e., point-constraint between two links via the cable) to these EKF-estimates of all the links to enforce the kinematic coherency and, consequently, (significantly) improve the estimation accuracy. For this, we adopt the framework of smoothly constrained Kalman filtering (SCKF) [16], where nonlinear constraint is linearized and applied repeatedly as measurement updates with some artificial noise. We choose SCKF here is because it is known of its superior performance of constraint error convergence as compared to other constrained KF techniques (see [17]), which is crucial for this paper as it can directly translate to the performance of enforcing the kinematic coherency of the skeleton system. We also extend the standard SCKF in this paper to incorporate the multi-dimensional kinematic constraints, error-state formulation and some suitable manifold structure common in SE(3)-motion estimation. We further devise a scalable semi-distributed version of the estimation algorithm, which can substantially speed up the computation speed by dividing the skeleton into several groups, locally-performing full-SCKF for each group, and globally-performing partial-SCKF among the groups. The presented estimation frameworks are then verified with real outdoor flying experiments and simulation studies of our LASDRA system. To our knowledge, this paper presents the very first result on the onboard estimation framework of the aerial skeleton system and its real outdoor flying demonstration.

The rest of the paper is organized as follows. In Sec. II, some preliminary information is explained for the description of main results including the modeling and controller for the system, in Sec. III-A, pose and posture estimation framework enforcing kinematic coherency is presented, and also its semi-distributed version is described. Simulation results to verify the scalability issue regarding the devised algorithm is introduced in Sec. IV, and in Sec. V, experiment result of outdoor flight using 3 link LASDRA system is presented.

II. Preliminary

A general aerial skeleton system can be modelled with Newton-Euler dynamics as following,

$$m_i\ddot{p}_i + m_i g e_3 = R_i u_i + R_i f_i - R_{i+1} f_{i+1}$$
$$J_i\dot{\omega}_i + S(\omega_i)J_i\omega_i = \tau_i + S(r_{ci,i})f_i - S(r_{ci,i+1})R_{i+1}f_{i+1}$$

where $m_i \in \Re, J_i \in \Re^{3\times3}$ are the mass and inertia matrix of i-th link, $p_i, \omega_i \in \Re^3$ are the position and angular velocity vector of i-th link expressed in inertial and body frame respectively, $R_i \in$ SO(3) is the rotation matrix of i-th link, $u_i, \tau_i \in \Re^3$ are force and torque input applied to i-th link in body frame, $f_i, f_{i+1} \in \Re^3$ are the force applied at left and right side joint of the i-th link (here, right side means positive x direction in link body frame), $g \in \Re$ is the gravitational constant with $e_3 := [0; 0; 1]$, and $S(\star)$ is the skew-symmetric matrix mapping.

In this paper, the LASDRA system [10] is exploited as an aerial skeleton, each link module of which is comprised of ODAR [7], [11] system. Each link modules can generate omni-directional force and torque with the non-aligned bi-directional rotors, and IMU, GPS modules are attached on the center of each link for the pose estimation of the system. The link modules of the system are connected each other using compliant cable, enabling to be acting as a 3DOF joint with wide range of motion. As the system has fully-actuated link modules, various control methods are available and one of the possible controller would be a decentralized impedance controller [10] as following

$$\begin{aligned} u_i &= R_i^T(m_i g e_3 + m_i\ddot{p}_{i,d} + k_d\dot{e}_{p,i} + k_p e_{p,i}) \\ \tau_i &= S(\omega_i)J_i\omega_i - k_R e_{R,i} - k_\omega e_{\omega,i} - k_I e_{I,i} \\ &\quad - J_i(S(\omega_i)R_{i,d}\omega_{i,d} - R_{i,d}\dot{\omega}_{i,d}) \end{aligned} \tag{1}$$

where $\ddot{p}_{i,d}, \omega_{i,d}, \dot{\omega}_{i,d} \in \Re^3$ are desired acceleration, angular velocity and angular acceleration, $e_{p,i} := p_i - p_{i,d} \in \Re^3$,$\dot{e}_{p,i} := \dot{p}_i - \dot{p}_{i,d} \in \Re^3$ are position and velocity error, $e_{R,i} := (R_{i,d}^T R_i - R_i^T R_{i,d})^\vee \in \Re^3$, $e_{\omega,i} := \omega_i - R_i^T R_{i,d}\omega_{i,d} \in \Re^3$, $e_{I,i} := \int e_{\omega,i} + \gamma_I e_{R,i} dt \in \Re^3$ are rotation, angular velocity and integral error, $(\star)^\vee$ is the mapping from skew-symmetric matrix to a vector, and $k_d, k_p, k_R, k_\omega, k_I, \gamma_I \in \Re$ are control gains. Here, notice that the position estimation is taking an important role for the control, which also leads to the motivation of the estimation framework in this work. Also, the controller provides an error-tolerant property, since it is an impedance control having some extent of compliancy.

For the LASDRA system, IMU sensors and GNSS modules are exploited for the pose estimation, and using these sensing modules for an aerial skeleton brings us the issues of position error accumulation propagated toward the link modules and kinematic coherency, as mentioned in the previous section. To deal with these issues, we come up with a novel estimation framework that enforce the estimated pose and posture to obey the given kinematic constraint using the SCKF algorithm. Also, to overcome the scalability issue when the number of links of the skeleton gets much larger, we devise a semi-distributed version of the proposed estimation framework, and all details on the proposed framework will be delineated in the following sections.

III. Pose and Posture Estimation of an Aerial Skeleton System

A. Estimation Algorithm via SCKF

In this paper, the SCKF algorithm in [16] is modified to be suitably used for aerial skeleton systems. The original algorithm is extended to be used for multi dimensional constraint, and the constraint is applied using error-state kinematics to deal with quaternion states. The developed

pose estimation algorithm is composed of two steps, a standard EKF and constraint application step, and both steps are described respectively in the following paragraphs.

1) Extended Kalman Filter: In the standard EKF step, the EKF state and covariance is updated with the measurement which can be expressed as

$$\hat{x}^E_{i,k}, P^E_{i,k} \leftarrow \text{EKF}(\hat{x}^E_{i,k-1}, P^E_{i,k-1}, y_{i,k}) \tag{2}$$

where we denote that the subscript $i \in \{1, ..., n\}$ is the link index, k is the step number of the filter, $\hat{x}_{i,k} \in \Re^{10}$ is the estimate of the state defined as

$$x_{i,k} := [p_{i,k}; v_{i,k}; q_{i,k}] \in \Re^{10}$$

where $p_{i,k}, v_{i,k}, q_{i,k} \in \Re^3$ are position, velocity and quaternion vector, and the superscript E means the variable resulting from the EKF. Also, $P_{i,k} \in \Re^{9\times9}$ is the covariance matrix of $\tilde{x}_{i,k}$, where $\tilde{x}_{i,k} := [\tilde{p}_{i,k}; \tilde{v}_{i,k}; \delta\theta_{i,k}] \in \Re^9$ is the error state for the EKF [14], [18], and $\tilde{p}_{i,k} := p_{i,k} - \hat{p}_{i,k}$, $\tilde{v}_{i,k} := v_{i,k} - \hat{v}_{i,k}$, and $\delta\theta_{i,k} \in \Re^3$ is the attitude error represented with angle vector which has relation of $q_{i,k} \otimes \hat{q}^{-1}_{i,k} \simeq [1; \frac{1}{2}\delta\theta_{i,k}]$, $\otimes$ is a quaternion multiplication operator, and $y_{i,k}$ is the measurement (e.g., GPS and IMU). Although there can be more state variables for the EKF state in (2) such as sensor biases, magnetic field vector, and so on, here we only consider position, velocity, and quaternion vectors as those are the variables to be updated by the constraint applying process.

2) Constraint Application: In this step, the estimate result of EKF and the constraint information is used to obtain the estimation result coherent with the kinematic constraint given by the system configuration. The constraint application process can be described with pseudo-code as following where

Algorithm 1: Constraint application process

$j \leftarrow 0,\ \hat{\mathbf{x}}^C_{k,j} \leftarrow \hat{\mathbf{x}}^E_k,\ \mathbf{P}^C_{k,j} \leftarrow \mathbf{P}^E_k$
while $||\mathbf{c}_{k,j}(\tilde{\mathbf{x}}^C_{k,j})||_2 > \epsilon$ **do**
  $(\hat{\mathbf{x}}^C_{k,j+1}, \mathbf{P}^C_{k,j+1}) \leftarrow \text{CA}(\hat{\mathbf{x}}^C_{k,j}, \mathbf{P}^C_{k,j}, \mathbf{c}_{k,j}(\hat{\mathbf{x}}^C_{k,j}))$
  $j \leftarrow j + 1$

$$\hat{\mathbf{x}}_k := [\hat{x}_{1,k}; \hat{x}_{2,k}; \cdots; \hat{x}_{n,k}] \in \Re^{10n}$$

is the stacked vector, n is the number of links of the system, $\mathbf{P}_k \in \Re^{9n\times9n}$ is the stacked block diagonal matrix of covariance matrices $P_{i,k}$, $\mathbf{c}_k(\hat{\mathbf{x}}^E_k) \in \Re^{6(n-1)}$ is the constraint error with the state estimate $\hat{\mathbf{x}}^E_k$, CA() is the abbreviation of "Constraint Apply", the superscript C means the state and covariance resulting from the constraint application process, and the subscript j is the step index that is initialized with 0 and added 1 after every loop. For the aerial skeleton system, the constraint error $\mathbf{c}_k(\hat{\mathbf{x}}_k)$ is defined as

$$\mathbf{c}_k(\hat{\mathbf{x}}_k) = [c_{1,k}; c_{2,k}; \cdots; c_{n-1,k}] \tag{3}$$

where the element $c_{i,k}$ is meaning the difference of position and velocity estimate at the point of joint between i-th and $(i+1)$-th links, and can be expressed as

$$c_{i,k}(\hat{x}_{i,k}, \hat{x}_{i+1,k}) = \begin{bmatrix} \hat{p}_{i,k} + \hat{R}_{i,k}d_i \\ \hat{v}_{i,k} + \hat{R}_{i,k}S(\hat{\omega}_{i,k})d_i \end{bmatrix} - \begin{bmatrix} \hat{p}_{i+1,k} - \hat{R}_{i+1,k}d_{i+1} \\ \hat{v}_{i+1,k} - \hat{R}_{i+1,k}S(\hat{\omega}_{i+1,k})d_{i+1} \end{bmatrix}$$

where $i \in \{1, ..., n-1\}$, $\hat{p}_{i,k}, \hat{v}_{i,k}, \hat{\omega}_{i,k} \in \Re^3$ are position, velocity, and angular velocity estimate of i-th link at k-th step, $\hat{R}_{i,k} \in$ SO(3) is the rotation matrix estimate, and $d_i \in \Re^3$ is the position vector from the i-th link center of mass to the right side joint. With an assumption of the symmetry of each link, the position vector to the left side joint can also be denoted as $-d_i$.

For the constraint application process, we also exploit the error state and error covariance as in usual EKF algorithms. The main advantage of using the error state in this work is the simple calculation of the constraint Jacobian which is described below the (4). By subtracting $\mathbf{c}_k(\hat{\mathbf{x}}_k)$ to the constraint error with nominal state $\mathbf{c}_k(\mathbf{x}_k)$, and with the small angle approximation of $\delta\theta_{i,k}$, the constraint equation can be expressed with the error state as following,

$$\tilde{\mathbf{c}}_k(\tilde{\mathbf{x}}_k) = [\tilde{c}_{1,k}; \tilde{c}_{2,k}; \cdots; \tilde{c}_{n-1,k}] \tag{4}$$

dimension of which is $\tilde{\mathbf{c}}_k(\tilde{\mathbf{x}}_k) \in \Re^{6(n-1)}$ and the each element $\tilde{c}_{i,k}$ is defined as

$$\tilde{c}_{i,k}(\tilde{x}_{i,k}, \tilde{x}_{i+1,k}) = \begin{bmatrix} \tilde{p}_{i,k} - S(\hat{R}_{i,k}d_i)\delta\theta_{i,k} \\ \tilde{v}_{i,k} - S(\hat{R}_{i,k}S(\hat{\omega}_{i,k})d_i)\delta\theta_{i,k} \end{bmatrix} - \begin{bmatrix} \tilde{p}_{i+1,k} + S(\hat{R}_{i+1,k}d_{i+1})\delta\theta_{i+1,k} \\ \tilde{v}_{i+1,k} + S(\hat{R}_{i+1,k}S(\hat{\omega}_{i+1,k})d_{i+1})\delta\theta_{i+1,k} \end{bmatrix}$$

where $i \in \{1, ..., n-1\}$, $\tilde{\mathbf{x}}_k := [\tilde{x}_{1,k}; \tilde{x}_{2,k}; \cdots; \tilde{x}_{n,k}] \in \Re^{9n}$ is the stacked error state vector, and it can be expressed again as $\frac{\partial\tilde{\mathbf{c}}_k}{\partial\tilde{\mathbf{x}}_k} \cdot \tilde{\mathbf{x}}_k = 0$. Here, we define the constraint Jacobian $\tilde{\mathbf{C}}_k(\hat{\mathbf{x}}_k) := \frac{\partial\tilde{\mathbf{c}}_k}{\partial\tilde{\mathbf{x}}_k} \in \Re^{6(n-1)\times9n}$ which can be easily obtained as the $\tilde{\mathbf{c}}_k$ has a form of analytic product with the error states.

We can now describe the function "ConstraintApply()" in the pseudo-code as following equations.

$$\begin{aligned}
\mathbf{P}^w_{k,j} &= \alpha e^{-j}\tilde{\mathbf{C}}_{k,j}\mathbf{P}^E_k\tilde{\mathbf{C}}^T_{k,j} \\
\mathbf{K}_{k,j} &= \mathbf{P}^C_{k,j}\tilde{\mathbf{C}}^T_{k,j}(\tilde{\mathbf{C}}_{k,j}\mathbf{P}^C_{k,j}\tilde{\mathbf{C}}^T_{k,j} + \mathbf{P}^w_{k,j})^{-1} \\
\tilde{\mathbf{x}}^C_{k,j+1} &= -\mathbf{K}_{k,j}\mathbf{c}_{k,j}(\hat{\mathbf{x}}^C_{k,j}) \\
\mathbf{P}^C_{k,j+1} &= (\mathbf{I} - \mathbf{K}_{k,j}\tilde{\mathbf{C}}_{k,j})\mathbf{P}^C_{k,j}(\mathbf{I} - \mathbf{K}_{k,j}\tilde{\mathbf{C}}_{k,j})^T \\
&\quad + \mathbf{K}_{k,j}\mathbf{P}^w_{k,j}\mathbf{K}^T_{k,j}
\end{aligned}$$

where $\mathbf{P}^w_{k,j} \in \Re^{6(n-1)\times6(n-1)}$ is the weakening covariance, an artificial noise for the constraint application that designed to decrease exponentially as the loop goes on, $\mathbf{K}_{k,j} \in \Re^{9n\times6(n-1)}$ is the kalman gain, $\tilde{\mathbf{C}}_{k,j} := \tilde{\mathbf{C}}_k(\hat{\mathbf{x}}_{k,j})$, α is a constant parameter which is set to be 0.01 as in [16]. Then the error state is update by the product of kalman gain and the constraint error, and the covariance matrix is also updated to be used for the next loop. Then, the state $\hat{\mathbf{x}}_{k,j}$ is updated

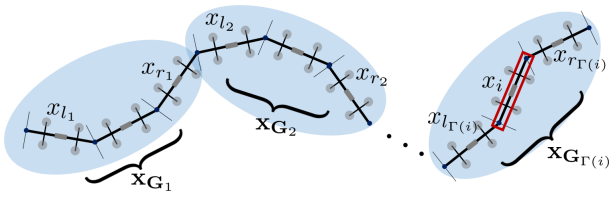

Fig. 2. Illustrative figure of the notations for the semi-distributed version algorithm.

with the error state as

$$\hat{p}^C_{i,k,j+1} = \hat{p}^C_{i,k,j} + \tilde{p}^C_{i,k,j+1}$$
$$\hat{v}^C_{i,k,j+1} = \hat{v}^C_{i,k,j} + \tilde{v}^C_{i,k,j+1}$$
$$\hat{q}^C_{i,k,j+1} = \hat{q}^C_{i,k,j} \otimes \delta q_{i,k,j+1}$$

where $\delta q_{i,k,j+1} \simeq [1; \frac{1}{2}\delta\theta_{i,k,j+1}]$ is a quaternion error vector and normalized before the multiplication. The loop is continued while the constraint error satisfies $||\mathbf{c}_{k,j}(\tilde{\mathbf{x}}^C_{k,j})||_2 < \epsilon$ where ϵ is a small enough number. After the termination of the constraint application loop (suppose that $j = t$), the resultant state estimate $\hat{\mathbf{x}}^C_{k,t}$ can be exploited as the final pose estimation $\hat{\mathbf{x}}_k$ used for the control.

In this work, the result of constraint application is not fed back to the next step EKF state again, since with the original algorithm: 1) when the measurement data has large deviation from the constraint, measurement update and the constraint application process can conflict each other causing state updates with large magnitude and also can harm the stability of the estimation; 2) there exists an implementation issue that the EKF and constraint application for multiple links need to be synchronized at every time step, and it is not simply applicable for the system with distributed computing modules as the case of our system.

B. Semi-Distributed Version of Algorithm

In this work, a semi-distributed version of the algorithm in Sec. III-A is also presented to deal with the scalability issue when the number links is extremely large. In the semi-distributed algorithm, the entire links of the system are divided into several groups which contains some number of links where the number depends on the computing performance of the on-board PC used for the system. Then, the constraint application process is performed for each group of links and the kinematic constraint can be obeyed at least among the links in a same group. To describe the proposed estimation scheme, let us define some notations (also refer to Fig. 2). First of all, each group of links are defined as a set and the γ-th group can be written as $\mathbf{G}_\gamma = \mathbf{G}_{\Gamma(i)} = \{l_{\Gamma(i)}, \cdots, i, i+1, \cdots, r_{\Gamma(i)}\}$ where γ is the index of the group increasing through the positive x direction in body frame of each link, $\Gamma(i)$ is the index of the group that contains the i-th link, $l_{\Gamma(i)}, r_{\Gamma(i)}$ are the indices of the most left and the most right link of the γ-th group which contains i-th link, the number of elements in the set $\mathbf{G}_\gamma$ is denoted as $|\mathbf{G}_\gamma| := n_\gamma$, and the number of the groups in the entire system is denoted as s.

Then, the semi-distributed pose estimation algorithm, as a substitute for the constraint application process in Sec. III-A, can be divided into local and global estimation steps, each will be explained in the paragraphs below.

1) Local Constraint Application: The local constraint application step shares the same function "CA()" in Algorithm 1, whereas the input for the function is changed as in Algorithm 2, where $\gamma = \Gamma(i) \in \{1, ..., s\}$, $\hat{\mathbf{x}}^E_{\mathbf{G}_\gamma,k} := [\hat{x}^E_{l_\gamma,k}; \cdots; \hat{x}^E_{r_\gamma,k}] \in \Re^{10n_\gamma}$ is the stacked vector of EKF pose estimates of links in γ-th group, $\mathbf{P}^E_{\mathbf{G}_\gamma,k} \in \Re^{9n_\gamma \times 9n_\gamma}$ is the covariance matrix of the error state $\tilde{\mathbf{x}}^E_{\mathbf{G}_\gamma,k} := [\tilde{x}^E_{l_\gamma,k}; \cdots; \tilde{x}^E_{r_\gamma,k}]$, similarly $\hat{\mathbf{x}}^C_{\mathbf{G}_\gamma,k} \in \Re^{10n_\gamma}$ and $\mathbf{P}^C_{\mathbf{G}_\gamma,k} \in \Re^{9n_\gamma \times 9n_\gamma}$ are the stacked state and covariance of pose estimates after constraint application, $\mathbf{c}_{\mathbf{G}_\gamma,k}(\hat{\mathbf{x}}^E_{\mathbf{G}_\gamma,k}) \in \Re^{6(n_\gamma-1)}$ is the kinematic constraint error among the links in γ-th group, $\mathbf{c}^G_k(\hat{\mathbf{x}}^C_k) \in \Re^{6(s-1)}$ is the constraint error vector defined as

Algorithm 2: Local constraint application process

foreach $\gamma \in \{1, \cdots, s\}$ **do**
  $j \leftarrow 0$, $\hat{\mathbf{x}}^C_{\mathbf{G}_\gamma,k,j} \leftarrow \hat{\mathbf{x}}^E_{\mathbf{G}_\gamma,k}$, $\mathbf{P}^C_{\mathbf{G}_\gamma,k,j} \leftarrow \mathbf{P}^E_{\mathbf{G}_\gamma,k}$
  while $||\mathbf{c}_{\mathbf{G}_\gamma,k,j}(\tilde{\mathbf{x}}^C_{\mathbf{G}_\gamma,k,j})||_2 > \epsilon$ **do**
    $(\hat{\mathbf{x}}^C_{\mathbf{G}_\gamma,k,j+1}, \mathbf{P}^C_{\mathbf{G}_\gamma,k,j+1}) \leftarrow$
      $\mathrm{CA}(\hat{\mathbf{x}}^C_{\mathbf{G}_\gamma,k,j}, \mathbf{P}^C_{\mathbf{G}_\gamma,k,j}, \mathbf{c}_{\mathbf{G}_\gamma,k,j}(\hat{\mathbf{x}}^C_{\mathbf{G}_\gamma,k,j}))$
    $j \leftarrow j + 1$

$$\mathbf{c}^G_k(\hat{\mathbf{x}}^C_k) := [c^G_{1,k}(\hat{x}_{r_1,k}, \hat{x}_{l_2,k}); \cdots; c^G_{s-1,k}(\hat{x}_{r_{s-1},k}, \hat{x}_{l_s,k})]$$

which is required to be zero, and $c^G_{\gamma,k}(\hat{x}_{r_\gamma,k}, \hat{x}_{l_{\gamma+1},k})$ is the position and velocity difference between the right side tip of the most right link in the γ-th group and left side tip of the most left link in the $(\gamma+1)$-th group, similarly defined as $c_{i,k}(\hat{x}_{i,k}, \hat{x}_{i+1,k})$ in (3).

2) Global Constraint Application: After the local constraint application process in Algorithm 2, the kinematic constraint is satisfied within a group, but there still exist inconsistencies between the tips of the neighbouring group of links. To resolve these inconsistencies, the global constraint application process is executed which is expressed as following

$$\hat{\mathbf{x}}_k \leftarrow \mathrm{GCA}(\hat{\mathbf{x}}^C_k, \mathbf{P}^C_k, \mathbf{c}^G_k(\hat{\mathbf{x}}^C_k)) \tag{5}$$

where GCA() is the abbreviation of the "Global Constraint Application".

Since the computation amount of the global constraint application process is directly related with the number of link of the system, the highest priority of this process would be reducing the computation to achieve the system scalability. As the main cause of computation load in the constraint application process is the nonlinear kinematic constraint coming from the attitude vectors, in this process, we consider the attitude estimates $\hat{q}^C_{i,k}$ as given values and only update the position and velocity estimate. Then, the global constraint application process can be thought of as shifting the position and velocity vectors of each group to match the given kinematic constraint, that is, $\mathbf{c}^G_k(\hat{\mathbf{x}}^C_k) = 0$. Let us define the shifting vector $\Delta x_{\mathbf{G}_{\Gamma(i)}} := [\Delta p_{\mathbf{G}_{\Gamma(i)}}; \Delta v_{\mathbf{G}_{\Gamma(i)}}] \in \Re^6$ for

TABLE I

PARAMETERS USED FOR THE SIMULATION

Descriptions	Values
Link length	1 [m]
Process noise	$\sim \mathcal{N}(0, \text{diag}[0.1I_3, 0.1I_3, 0.01I_3])$
Measurement noise	$\sim \mathcal{N}(0, \text{diag}[0.1I_3, 0.1I_3, 0.01I_4])$
Constraint apply stop condition	$\|\|\mathbf{c}_{k,j}(\tilde{\mathbf{x}}^C_{k,j})\|\|_2 < 0.01$ [m]

each link, and the vector share same values if the links are in the same group. To decide these shifting vectors, we solve for following constrained optimization

$$\min_{\Delta x_{\mathbf{G}_{\Gamma(i)},k}} \sum_{i=1}^{n} (\Delta x_{\mathbf{G}_{\Gamma(i)},k})^T \bar{P}^{-1}_{i,k} (\Delta x_{\mathbf{G}_{\Gamma(i)},k}) \quad (6)$$
$$\text{s.t. } \mathbf{c}^G_k(\hat{\mathbf{x}}_k) = 0$$

where $\bar{P}_{i,k} \in \Re^{6\times 6}$ is the covariance of $[p^C_{i,k}; v^C_{i,k}]$ which discarded quaternion vector from $x^C_{i,k}$, $\hat{\mathbf{x}}_k$ is the stacked vector of the final estimate element of which is calculated as $\hat{x}_{i,k} = \hat{x}^C_{i,k} + [\Delta x_{\mathbf{G}_{\Gamma(i)}}; 0] \in \Re^{10}$. The objective function means minimizing the Mahalanobis distance from the estimation $\hat{\mathbf{x}}^C_k$ to the $\hat{\mathbf{x}}_k$. Since (6) is the quadratic program with linear constraint, closed form solution can be easily obtained. Although this scheme is less optimal than the process in Sec. III-A as the attitude is not updated with the constraint apply, it does not need multiple loops of constraint application and can obtain updated estimate with a closed form solution.

IV. SIMULATION

In this paper, we perform simulation to verify the scalability issue of the devised estimation framework. In the simulation, 12 link aerial skeleton system is simulated with artificial measurements of IMU and GNSS. The parameters used for the artificial measurements, EKF, and SCKF are summarized in Table I. In addition to a white Gaussian measurement noise, we also applied bias with low frequency oscillation (< 0.1Hz) for position and quaternion measurement with amplitude of 0.5m and $5°$ in axis angle, respectively, to emulate the GNSS drift, biases of IMU.

Then, first of all, computing time increase with respect to the link number increase is checked with simulation, while comparing the semi-distributed version of the algorithm with non-applied algorithm. For both cases, simulation is conducted for 20 seconds, size of the group n_γ is set to be 2 for the semi-distributed algorithm, and the information of the whole links are insulted to the "CA()" function in Algorithm 1. The result is depicted in Fig. 3 and we can clearly see that the computation time of the semi-distributed algorithm grows much slower than the other, implying that the advantage of the algorithm further increases for the aerial skeleton with very large number of links.

For the second simulation, computation time and the accuracy of the estimation result are compared together, for the semi-distributed algorithm with different group sizes $n_\gamma \in \{2, 3, 4, 6, 12\}$. An illustrative figure of 12 link aerial skeleton during the algorithm is shown in Fig. 4 at simulation

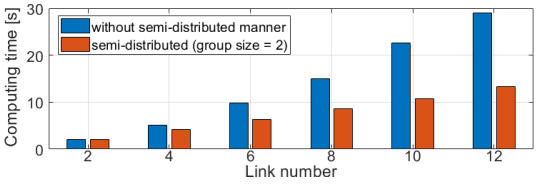

Fig. 3. Comparison of computing time according to the link number increase and the usage of semi-distributed algorithm

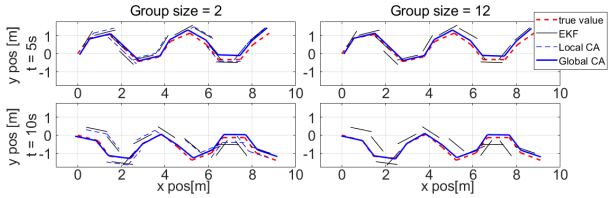

Fig. 4. Illustrative figure of the process of the semi-distributed version of the constraint application algorithm.

time $5, 10, 15$s, where each link module is drawn with a line segment. The figure shows the entire process of the proposed estimation framework from EKF to the final estimate after the global estimation process, and each case of $n_\gamma = 2, 12$ shows distinguishable difference of local estimation result (blue narrow dotted line), where two combined links are shown in the left column of figures. Then, the results of computation time and the accuracy of estimation are presented in Table II. Position and quaternion error in the table are meaning the two norm of the error (from the true value) average throughout the all links and all time steps in the simulation. The result shows that a slight increase of the estimation accuracy (about 15% and 12% for position and quaternion error respectively, from $n_\gamma = 2$ to $n_\gamma = 12$) is obtained with the increase of the group size, while sacrificing the computation time (more than twice from $n_\gamma = 2$ to $n_\gamma = 12$). This trade-off relation can be a reference for selecting the group size n_γ for the semi-distributed algorithm, considering the computing performance of the system and the desired extent of estimation accuracy.

V. EXPERIMENT

To verify the proposed estimation framework, we implement the outdoor flight using the LASDRA system [10] containing three ODAR [7] system (1m length, 1.8kg weight for each) as link modules. The link modules are connected each other using a compliant PVC (polyvinyl chloride) cable which can provide high operation range. For the actuation of

TABLE II

COMPUTATION TIME & ESTIMATION ACCURACY ACCORDING TO THE GROUP SIZE INCREASE

Group Size (n_γ)	Computing time [s]	Pos. error [m]	Quat. error
2	13.753	0.0377	0.0026
3	15.496	0.0370	0.0025
4	16.356	0.0369	0.0025
6	20.900	0.0360	0.0023
12	29.041	0.0322	0.0023

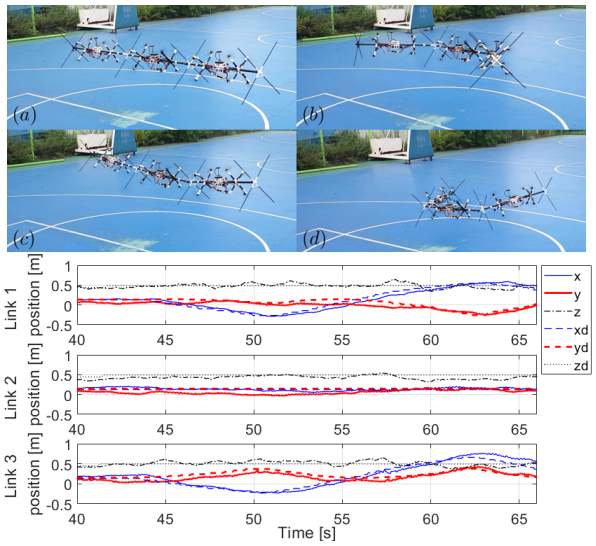

Fig. 5. Snapshots and position tracking result of 3 link LASDRA outdoor flying experiment. Solid line: estimated position, dashed line: desired position.

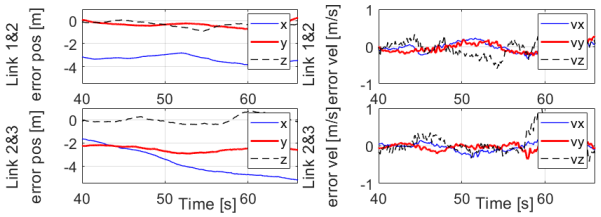

Fig. 6. Constraint error before the constraint application process.

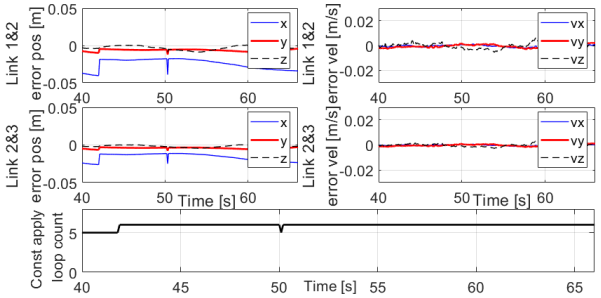

Fig. 7. Constraint error after the constraint application process, and the number of the constraint application loop run.

the system, DJI snail propulsion system is exploited with 6048-3D propeller that can generate bi-directional thrust upto about 8N. In the LASDRA system, both Raspberry Pi 3 and Pixhawk 2.4.8 are used as computing modules, one Raspberry Pi mounted on the middle link and three Pixhawks are attached on the center of every link modules. The Raspberry Pi takes a role of the main PC of the system, sending desired pose command, and collecting all the state and covariance information from the EKF running on each Pixhawks for the computation of SCKF algorithm. Due to the limited computing power of the onboard PC, the update rate is set to be 20Hz, yet, for the result of SCKF algorithm, only a difference from the EKF result is transferred to the Pixhawk so that the fast update rate of the EKF is not harmed while relatively slowly applying the constraint information. Then, on each Pixhawk, desired pose command and the updated state estimate with constraint application are received, EKF and controller are calculated with 500Hz and finally PWM signal is generated to run the rotors. For the power supply of motors, 4S LiPo battery attached on each link is exploited and the battery is also used to supply power for all the computing modules after stepdowning the voltage to 5V using the battery eliminator circuit (BEC). For the sensing modules of the system, IMU's inside the Pixhawk and GPS modules (U-blox NEO-M8) are used and those are mounted on every links of the system. Here, RTK (real-time kinematic) GPS can be another option for a sensing module which provides much more accurate position measurement than a general GNSS, yet, as the main purpose of the system prototype is verifying the performance of the proposed estimator, not achieving the best flight performance, we do not consider using of it. Exploiting the RTK-GPS and implementing the aerial skeleton with further accurate and agile motion would be one of our future work.

Then, with the constructed system above, outdoor flight experiment is performed as depicted in Fig. 5. After hovering for a few seconds, a pose trajectory is given to the system so that the system behaves like bending its configuration in forward and backward direction. The result of the flight is shown in the bottom of the Fig. 5. The RMS error between the desired and estimated position of each link is 11.61cm, 13.40cm, 12.57cm respectively. Although the position tracking error is not small, there was no issue at generating desired pose and posture, due to the error-tolerant property of the impedance control. Also in Fig. 6 and 7, the constraint errors before and after the SCKF algorithm are described, and these show that the constraint error is well regulated under 5cm and 2cm/s while there exist large constraint errors before the algorithm due to the inaccuracy of the GNSS, and gyro noise. There are small jump and peak in the position constraint error near $42, 50$s and this happened due to the change of the loop number of constraint application process as shown in the bottom plot in Fig. 7. This can be alleviated if the loop stop condition $||\mathbf{c}_{k,j}(\tilde{\mathbf{x}}^C_{k,j})||_2 < \epsilon$ is set much tightly, where the ϵ in the experiment is set to be 0.05 to avoid an excessive computation.

VI. Conclusion

In this paper, we present a novel pose and posture estimation framework of aerial skeleton system for outdoor flying. To enforce the kinematic coherency of the individual EKF estimates, we apply the kinematic constraints of the aerial skeleton system to the EKF estimates of all the links through SCKF, thereby, enforcing the kinematic coherency of the skeleton system and, a semi-distributed version of the obtained estimation framework is also presented to address the issue of scalability. The proposed estimation framework is then verified with real outdoor flying experiments and simulation studies. Some possible future works are as follows: motion generation for the system endowed with naturalness; teleoperation of the system; experiment verification of the semi-distributed version of the algorithm.

REFERENCES

[1] N. Michael, S. Shen, K. Mohta, V. Kumar, et al. Collaborative mapping of an earthquake damaged building via ground and aerial robots. In *Proc. Field and Service Robotics*, pages 33–47, 2014.

[2] DJI. Agras mg-1. https://www.dji.com/mg-1. Accessed: 2018-09-15.

[3] M. Waibel, B. Keays, and F. Augugliaro. Drone shows: creative potential and best practices. Technical report, ETH Zurich, 2017.

[4] H. Yang and D. J. Lee. Dynamics and control of quadrotor with robotic manipulator. In *Proc. IEEE Int'l Conference on Robotics & Automation*, pages 5544–5549, 2014.

[5] H-N. Nguyen, C. Ha, and D. J. Lee. Mechanics, control and internal dynamics of quadrotor tool operation. *Automatica*, 61:289–301, 2015.

[6] H-N. Nguyen, S. Park, J. Park, and D. J. Lee. A novel robotic platform for aerial manipulation using quadrotors as rotating thrust generators. *IEEE Transactions on Robotics*, 34(2):353–369, 2018.

[7] S. Park, J. Lee, J. Ahn, M. Kim, J. Her, G-H. Yang, and D. J. Lee. Odar: aerial manipulation platform enabling omnidirectional wrench generation. *IEEE/ASME Transactions on Mechatronics*, 23(4):1907–1918, 2018.

[8] N. Staub, D. Bicego, Q. Sablé, V. Arellano, S. Mishra, and A. Franchi. Towards a flying assistant paradigm: the othex. In *Proc. IEEE Int'l Conference on Robotics and Automation*, pages 6997–7002, 2018.

[9] B. Gabrich, D. Saldana, V. Kumar, and M. Yim. A flying gripper based on cuboid modular robots. In *Proc. IEEE Int'l Conference on Robotics and Automation*, pages 7024–7030, 2018.

[10] H. Yang, S. Park, J. Lee, J. Ahn, D. Son, and D. J. Lee. Large-size aerial skeleton system with distributed rotor actuation. In *Proc. IEEE Int'l Conference on Robotics & Automation*, pages 7017–7023, 2018.

[11] S. Park, J. Her, J. Kim, and D. J. Lee. Design, modeling and control of omni-directional aerial robot. In *Proc. IEEE/RSJ Int'l Conf. on Intelligent Robots & Systems*, pages 1570–1575, 2016.

[12] G. Endo, T. Hagiwara, Y. Nakamura, H. Nabae, and K. Suzumori. A proposal of super long reach articulated manipulator with gravity compensation using thrusters. In *Proc. IEEE/ASME Int'l Conference on Advanced Intelligent Mechatronics*, pages 1414–1419, 2018.

[13] M. Zhao, T. Anzai, F. Shi, X. Chen, K. Okada, and M. Inaba. Design, modeling, and control of an aerial robot dragon: A dual-rotor-embedded multilink robot with the ability of multi-degree-of-freedom aerial transformation. *IEEE Robotics and Automation Letters*, 3(2):1176–1183, 2018.

[14] Y. Lee, J. Yoon, H. Yang, C. Kim, and D.J. Lee. Camera-gps-imu sensor fusion for autonomous flying. In *Proc. Int'l Conference on Ubiquitous and Future Networks*, pages 85–88, 2016.

[15] L. Meier, P. Tanskanen, L. Heng, G. H. Lee, F. Fraundorfer, and M. Pollefeys. Pixhawk: A micro aerial vehicle design for autonomous flight using onboard computer vision. *Autonomous Robots*, 33(1-2):21–39, 2012.

[16] J. D. Geeter, H. V. Brussel, J. D. Schutter, and M. Decréton. A smoothly constrained kalman filter. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (10):1171–1177, 1997.

[17] D. Simon. Kalman filtering with state constraints: a survey of linear and nonlinear algorithms. *IET Control Theory & Applications*, 4(8):1303–1318, 2010.

[18] J. Sola. Quaternion kinematics for the error-state kalman filter. *arXiv preprint arXiv:1711.02508*, 2017.