State Estimation for Tensegrity Robots

Ken Caluwaerts[1,*], Jonathan Bruce[2,*], Jeffrey M. Friesen[3], and Vytas SunSpiral[4]

***Abstract*— Tensegrity robots are a class of compliant robots that have many desirable traits when designing mass efficient systems that must interact with uncertain environments. Various promising control approaches have been proposed for tensegrity systems in simulation. Unfortunately, state estimation methods for tensegrity robots have not yet been thoroughly studied. In this paper, we present the design and evaluation of a state estimator for tensegrity robots. This state estimator will enable existing and future control algorithms to transfer from simulation to hardware. Our approach is based on the unscented Kalman filter (UKF) and combines inertial measurements, ultra wideband time-of-flight ranging measurements, and actuator state information. We evaluate the effectiveness of our method on the SUPERball, a tensegrity based planetary exploration robotic prototype. In particular, we conduct tests for evaluating both the robot's success in estimating global position in relation to fixed ranging base stations during rolling maneuvers as well as local behavior due to small-amplitude deformations induced by cable actuation.**

I. Introduction

Tensegrity robotics is a relatively young field of research wherein a robot is structured according to tensegrity principles. We define a tensegrity as a structure with discontinuous compression elements suspended within a web of tension elements. In this class of robots, motion is often achieved through actuation of the tensile elements within the structure.

Tensegrity robots have highly coupled dynamics due to their interconnected network of compliant tensile elements. As such, an external force exerted at a single point will cause a global displacement in all nodal positions of the system. This property is beneficial in that it allows the system to passively adapt to external forces and redistribute loads effectively through the tension network. However, it also causes difficulties in determining the state of the system when only limited sensor information is available.

Various control approaches have been proposed for tensegrity systems in simulation. However, these algorithms often rely on full state or trajectory information [1][2][3][4]. In this paper, we present the design and evaluation of a state estimator for this class of robot which will allow the transfer of these existing and future control algorithms from simulation to hardware.

*These authors contributed equally to this work.

This work was performed while all authors were at the NASA Ames Research Center, Moffett Field CA, USA, with funding from NASAs NSTRF, NIAC and GCD Programs

[1]Oak Ridge Associated Universities (ORAU), Oak Ridge TN, USA ken.caluwaerts@nasa.gov

[2]University of California Santa Cruz, Santa Cruz CA, USA jebruce@ucsc.edu

[3]University of California San Diego, San Diego CA, USA jfriesen@ucsd.edu

[4]SGT Inc., Greenbelt MD, USA vytas.sunspiral@nasa.gov

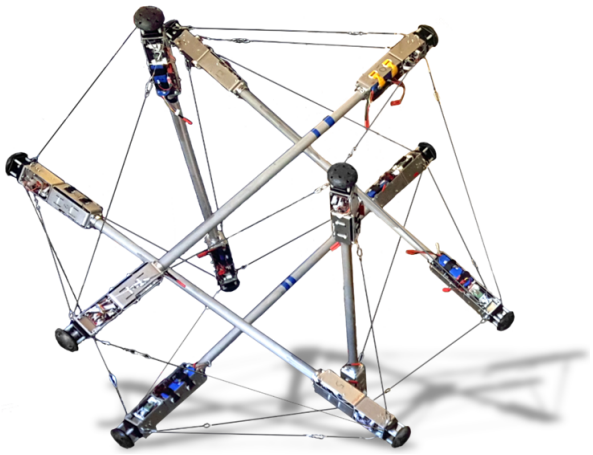

Fig. 1: The SUPERball prototype, a six strut tensegrity robot with 12 actuated cables and 12 passive cables. Our state estimation algorithm is evaluated on this platform.

We focus on the use of low-cost ranging modules and inertial measurement units mounted to the rods of a tensegrity robot as the sensor inputs to an unscented Kalman filter (UKF). These ranging sensors can be purchased off-the-shelf and do not rely on any user-designed mechanical infrastructure to operate. We also use motor encoders to sense change in cable rest length as control inputs into the dynamic model utilized by the UKF. For testing our approach, we use the SUPERball prototype, a six strut tensegrity robot designed to explore tensegrity systems for planetary exploration [5][6]. SUPERball is shown in Fig. 1.

This paper is organized as follows. We first present a detailed overview of the system's sensors, ranging method and calibration routine. We then describe the implementation of the unscented Kalman filter and the dynamics model. This is followed by our experimental results. We end this paper with our conclusions and future outlook.

II. SUPERball Overview

The Spherical Underactuated Planetary Exploration Robot (SUPERball) is a preliminary tensegrity robot prototype developed at the NASA Ames Research Center. The purpose the SUPERball project is to develop technologies for a new class of planetary exploration robot which is able to deploy from a compact launch volume, land at high speeds without the use of air-bags, and provide robust surface mobility. This broad set of functions can be enabled by utilizing the efficiency and structural compliance of tensegrity robots. Passive-structure drop tests have confirmed the analysis supporting the high-speed landing concept, and the current

prototype of SUPERball is intended to develop the foundational engineering approaches required to support surface locomotion by tensegrity robots. A full system overview is out of the scope of this paper, but the relevant details of the system will be discussed. Please refer to Bruce et. al. [5] and Sabelhaus and Bruce [6] for system details.

Relevant to this work are the sensors, actuators, and the ROS network implemented on SUPERball. SUPERball consists of 6 identical rods held together by 24 cables in-line with springs. As described in [5], each rod of SUPERball is comprised of two modular tensegrity robotic platforms (end caps). Each platform is equipped with inertial measurement units, a motor with encoders, and a fully enabled Robot Operating System (ROS) node which communicates to our ROS network via WiFi. SUPERball is underactuated and only 12 out of the 24 cables can be actuated (shortened) by spooling cable around a spindle (one per end cap). This allows the robot to roll through deformation. Since each modular tensegrity platform can be outfitted with sensors, ranging sensors were added for this paper to enable positioning. The ranging sensors are discussed in detail in Section III.

III. Ranging Setup and Calibration

This section introduces the hardware and software setup for a set of wireless ranging modules to enable position tracking of the robot both as an as internal distance measurements (end cap to end cap) an in an external (world) reference frame.

We equipped all end caps of SUPERball with a DWM1000 ranging module from DecaWave Ltd. By employing ultra wideband technology, the low-cost DWM1000 modules provide wireless data transfer and highly accurate timestamps of transmitted and received packets. This allows the distance between two DWM1000 modules to be estimated by computing the time-of-flight of exchanged messages without the need for synchronized clocks. We opted for this technology because it allows proprioceptive state estimation (distances between end caps), which cannot be easily tracked directly via motor encoders. [7] Furthermore, we placed eight more DWM1000 modules as "fixed anchors" around our testing area to provide a world reference frame for ground truth and generation of a reward signal for the machine learning algorithms that we will use to develop locomotion controllers for the robot. Our intention is that the fixed anchors will not be required in the final deployed version of the robot, and are primarily for use during algorithm development.

We first introduce the basic sensor operation and our approach to efficiently estimate distances between a large number of sensor modules. This is followed by a discussion of our ranging software and hardware setup. Finally, we provide a calibration routine similar to a common motion capture system that allows for quick set up of the sensor network.

A. Sensor Operation

1) Bidirectional Ranging: We operate the DWM1000 in the so-called *symmetric double-sided two-way ranging* mode. In this mode, the modules exchange 3 packets to estimate the time-of-flight between each other. While the time-of-flight of unsynchronized modules can be estimated with the exchange of only 2 packets, the employed mode can significantly reduce measurement noise [8].

The basic ranging packet exchange is shown in Fig. 2. One module sends out a *poll* message containing an emission timestamp (t_{SP}) using its local clock. A second module receives this message and timestamps the time of arrival using its local clock (t_{RP}). The second module then sends out a *response* packet at time t_{SR} (module 2's clock). The first module receives this packet at time t_{RR} (module 1's clock). Module 1 now sends out a final message containing t_{RR} and the emission time of the final message (t_{SF}, clock of module 1). Module 2 receives this information and timestamps it (t_{RF}).

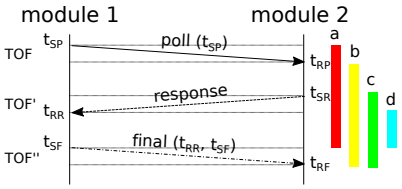

Fig. 2: Basic symmetric double-side two-way ranging packet exchange. Modules 1 and 2 exchange 3 packets (*poll*, *response*, and *final*). Module 2 then estimates the distance between the modules based on the local timestamps.

Module 2 can now estimate the time-of-flight and the distance between itself and module 1 based on the 6 timestampes. The basic equations to estimate the distance between module i and module j (module i initiates the ranging and module j computes the distance) are given by:

$$a_i = t^i_{SF} - t^i_{SP} \quad (1)$$
$$b_{j,i} = t^{j,i}_{RF} - t^{j,i}_{RP} \quad (2)$$
$$c_{j,i} = t^{j,i}_{RF} - t^{j}_{SR} \quad (3)$$
$$d_{i,j} = t^{i}_{SF} - t^{i,j}_{RR} \quad (4)$$

$$TOF_{j,i} \approx \frac{1}{2}\left(c_{j,i} - d_{i,j}\frac{b_{j,i}}{a_i}\right) - \delta_{j,i} \quad (5)$$
$$\|\boldsymbol{N}_j - \boldsymbol{N}_i\| \approx \frac{1}{2\boldsymbol{c}}\left(c_{j,i} - d_{i,j}\frac{b_{j,i}}{a_i}\right) - o_{j,i} \quad (6)$$
$$\doteq m_{j,i} - o_{j,i}. \quad (7)$$

The variables a, b, c, and d are also visualized in Fig. 2. The time-of-flight calculation between two modules i and j ($TOF_{j,i} = TOF_{i,j}$) is hindered by a fixed measurement offset ($\delta_{j,i} = \delta_{i,j}$). This offset is due to antenna delays and other discrepancies between the timestamps and actual packet reception or emission. Whereas this offset is expected to be unique to each module, we found that it is necessary to estimate this offset pairwise for closely located modules. Our hypothesis is that the proximity of the robot's motors and the sensor's position near the end cap's metal structure influence the antenna characteristics between pairs of modules.

Eq. 6 estimates the distances between the modules based on the time-of-flight calculation ($\boldsymbol{c}$ is the speed of light). We rewrite the time offset $\delta_{j,i}$ as a distance offset $o_{j,i}$ (with $o_{j,i} = o_{i,j}$). Here $\boldsymbol{N}_i$ and $\boldsymbol{N}_j$ refer to the positions of nodes i and j respectively (see Section IV). The variables $m_{j,i}$ represent the uncorrected distance estimates.

The DWM1000 requires careful configuration for optimal performance. We provide our main configuration settings in Table I. The ranging modules tend to measure non line-of-sight paths near reflective surfaces (e.g. floor, computer monitors), which may cause filter instability. Using the DWM1000's built-in signal power estimator, we reject such suspicious packets. In practice, between 30% and 70% of packets are rejected in our indoor test environment.

TABLE I: DWM1000 configuration

bitrate	channel	preamble	PRF	preamble code
$6.8\,\mathrm{Mbit\,s^{-1}}$	7	256	64 MHz	17

2) Broadcast Ranging: Due to the large number of exchanged packets (3 per pair) bidirectional ranging between pairs of modules quickly becomes inefficient when the number of modules grows. We propose a simple approach using timed broadcast messages that scales linearly in the number of modules (3 packets per module). In this setup one module periodically initiates a measurement sequence by sending out a *poll* message. When another module receives this message it emits its own *poll* message after a fixed delay based on its ID, followed by *response* and *final* messages after additional delays. Broadcast ranging is illustrated in Fig. 3.

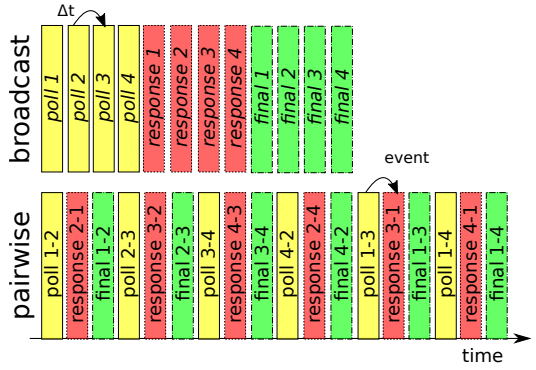

Fig. 3: Packet exchange between 4 modules for bidirectional pairwise and broadcast ranging. Timed broadcast messages allow for efficient ranging with a large number of modules.

One disadvantage of the broadcasting approach is that the total measurement time between a pair of modules takes longer (up to $60\,\mathrm{ms}$ in our experimental setup) than a single pairwise bidirectional measurement (approx. $3\,\mathrm{ms}$). However, broadcast ranging provides two measurements for each pair of modules per measurement iteration.

Note that each module now needs to keep track of the *poll* and *final* packet reception times of all other modules. The *final* packet becomes longer as each module needs to transmit the *response* reception time (t_{RR}) of all other modules.

B. Ranging Setup

Each end cap of SUPERball was fitted with a DWM1000 module located approximately $0.1\,\mathrm{m}$ from the end of the strut. To simplify the notation, we do not distinguish between end cap positions (ends of the struts) and the positions ranging sensor locations. In practice, we take this offset into account in the output function of our filter (see Section IV).

The broadcasting algorithm runs at $15\,\mathrm{Hz}$ and packet transmissions are spaced $1\,\mathrm{ms}$ apart. This allows for over 20 modules to range. After one ranging iteration, each end cap transmits its measurements over WiFi to the ROS network. A ROS node then combines measurements from all end caps into a single ROS message at $10\,\mathrm{Hz}$.

The fixed anchors operate in a similar way to the end caps, but are not connected to a ROS node and can not directly transmit data to the ROS network. This means that we obtain two measurements (one in each direction) for each pair of modules on the robot, but only a single measurement between the fixed anchors and the modules on the robot.

C. Calibration

One of the design goals of our state estimation method is quick deployment in new environments without significant manual calibration. To achieve this, we implemented an automatic calibration procedure to jointly estimate the constellation of fixed modules (anchors, defining an external reference frame) and the pairwise sensor offsets ($o_{i,j}$). Calibration is performed - similar to common motion capture systems - by moving the robot around, while recording the uncorrected distance measurements ($m_{j,i}$).

After recording a dataset, we minimize the reconstruction error L by optimizing over the offsets $\boldsymbol{o}$ ($o_{i,j}$ rearranged as a vector), the estimated anchor locations $\boldsymbol{N}^{est}$, and the estimated moving module locations $\boldsymbol{N}^{float}[1 \dots n_{samples}]$ (i.e. the module on the robot's end caps):

$$L\left(i,j,t\right) = \left(\|\boldsymbol{N}_i^{anchor} - \boldsymbol{N}_j^{float}\left[t\right]\| - o_{j,i} - m_{i,j}\left[t\right]\right)^2 \quad (8)$$

$$L\left(\boldsymbol{o}, \boldsymbol{N}^{anchor}, \boldsymbol{N}^{float}[1 \dots n_{samples}]\right) = \textstyle\sum_{i,j,t} \alpha_{j,t} L\left(i,j,t\right). \quad (9)$$

The brackets in $\boldsymbol{N}^{float}[1 \dots n_{samples}]$ indicate the moving module locations (end cap positions) at a specific timestep. For example $\boldsymbol{N}^{float}[5]$ contains the estimated end cap positions at timestep 5 in the recorded dataset. In Eq. 9, i iterates over anchors, j iterates over moving nodes and t iterates over samples. The indicator variables $\alpha_{j,t}$ are equal to 1 when for sample t there are at least 4 valid measurements to the fixed module for moving module j (i.e. the number of DOFs reduces).

In practice we also add constraints on the bar lengths, which take the same form as Eq. 8 with the offsets set to 0. We used BFGS to minimize Eq. 9 with a dataset containing approximately 400 timesteps selected randomly from a few minutes of movement of the robot. Although the algorithm works without prior knowledge, we noticed that providing the relative positions of 3 fixed nodes (3 manual measurements) significantly improves the success rate as there are no guarantees on global convergence.

Once the external offsets (between the anchors and moving nodes) and the module positions are known, we can estimate the offsets between moving nodes in a straightforward way by computing the difference between the estimated internal distances and the uncorrected distance measurements.

IV. Filter Design

Tensegrity systems are nonlinear and exhibit hybrid dynamics due to cable slack conditions and interactions with the environment that involve collision and friction. This warrants a robust filter design to track the robot's behavior.

The commonly used Extended Kalman Filter (EKF) does not perform well on highly nonlinear systems where first-order approximations offer poor representations of the propagation of uncertainties. Additionally the EKF requires computation of time-derivatives through system dynamics and output functions which is challenging for a model with complex hybrid dynamics.

The sigma point UKF does not require derivatives through the system dynamics and is third order accurate when propagating Gaussian Random Variables through nonlinear dynamics [9]. The computational cost of the UKF is comparable to that of the EKF, but for tensegrity systems which commonly have a large range of stiffnesses and a high number of state variables the time-update of the sigma points dominates computational cost. As such we first describe the methods used to reduce computational cost of dynamic simulation, then in the following section we outline the specific implementation of the UKF for the SUPERball prototype.

A. Dynamic Modelling

The UKF requires a dynamic model which balances model fidelity and computational efficiency since it requires a large number of simulations to be run in parallel. We model a tensegrity system as a spring-mass net and used the following incomplete list of simplifying assumptions:

- Only point masses located at each node point
- All internal and external forces are applied at nodes
- Members exert only linear stiffness and damping
- Unilateral forcing in cables
- Flat ground at a known height with Coulomb friction
- No bar or string collision modelling

For a tensegrity with n nodes and m members, the member force densities, $\boldsymbol{q} \in \mathbb{R}^m$, can be transformed into nodal forces, $\boldsymbol{F_m} \in \mathbb{R}^{n\times 3}$, by using the current Cartesian nodal positions, $\boldsymbol{N} \in \mathbb{R}^{n\times 3}$, and the connectivity matrix, $\boldsymbol{C} \in \mathbb{R}^{m\times n}$, as described in [10]. This operation is described by the equation:

$$\boldsymbol{F_m} = \boldsymbol{C}^T diag(\boldsymbol{q})\boldsymbol{C}\boldsymbol{N},$$

where $diag(\cdot)$ represents the creation of a diagonal matrix with the vector argument along its main diagonal. We first note that $\boldsymbol{CN}$ produces a matrix $\boldsymbol{U} \in \mathbb{R}^{m\times 3}$ where each row corresponds to a vector that points between the ith and jth nodes spanned by each member. Therefore, this first matrix multiplication can be replaced with vector indexing as $\boldsymbol{U}_k = \boldsymbol{N}_i - \boldsymbol{N}_j$, where we use the notation $\boldsymbol{U}_k$ to denote the kth row of matrix $\boldsymbol{U}$. If we then compute $\boldsymbol{V} = \boldsymbol{C}\frac{d\boldsymbol{N}}{dt}$ with the same method as $\boldsymbol{U}$, we obtain a matrix of relative member velocities. The matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ are used to calculate member lengths as $L_k = |\boldsymbol{U}_k|_2$ and member velocities as $\frac{d}{dt}(L_k) = \frac{\boldsymbol{U}_k(\boldsymbol{V}_k)^T}{L_k}$.

We can then use these values to calculate member force densities, $\boldsymbol{q}$, using Hooke's law and viscous damping as:

$$\boldsymbol{q}_k = K_k(1 - \frac{L_{0k}}{L_k}) - \frac{c_k}{L_k}\frac{d}{dt}(L_k).$$

Here K_k and c_k denote the kth member's stiffness and damping constants, respectively. Note that cables require some additional case handling to ensure unilateral forcing.

Scaling each $\boldsymbol{U}_k$ by $\boldsymbol{q}_k$ yields a matrix whose rows correspond to vector forces of the members. We denote this matrix as $\boldsymbol{U}^q \in \mathbb{R}^{m\times 3}$, and we note that $\boldsymbol{U}^q = diag(\boldsymbol{q})\boldsymbol{C}\boldsymbol{N}$. Thus this matrix of member forces can be easily applied to the nodes using:

$$\boldsymbol{F_m} = \boldsymbol{C}^T\boldsymbol{U}^q.$$

We now have a method for computing nodal forces exerted by the members and need only compute ground interaction forces, which we will denote as $\boldsymbol{F}_g$. We computed ground interaction forces using the numerical approach in [11]. The nodal accelerations can then be written as:

$$\frac{d^2\boldsymbol{N}}{dt^2} = \boldsymbol{M}^{-1}(\boldsymbol{F}_m + \boldsymbol{F}_g) - \boldsymbol{G},$$

where $\boldsymbol{M} \in \mathbb{R}^{n\times n}$ is a diagonal matrix whose diagonal entries are the masses of each node and $\boldsymbol{G} \in \mathbb{R}^{n\times 3}$ is matrix with identical rows equal to the vector acceleration due to gravity. It is then straightforward to simulate this second order ODE using traditional numerical methods.

Note also that it is possible to propagate many parallel simulations efficiently by concatenating multiple $\boldsymbol{N}$ matrices column wise to produce $\boldsymbol{N}_{\|} \in \mathbb{R}^{n\times 3l}$ for l parallel simulations. The resultant vectorization of many of the operations yields significant gains in computational speed with some careful handling of matrix dimensions.

B. UKF Implementation

We implement a traditional UKF as outlined in [9] with additive Gaussian noise for state variables and measurements.

Several parameters are defined for tuning the behavior of the UKF, namely α, β and κ, where α determines the spread of the sigma points generated by the unscented transformation, β is used to incorporate prior knowledge of distribution, and κ is a secondary scaling parameter. We hand tuned these parameters to the values $\alpha = 0.0139$, $\beta = 2$ for Gaussian distributions and $\kappa = 0$ and found this to yield an adequately stable filter.

We define our state variables as $\boldsymbol{N}$ and $\frac{d\boldsymbol{N}}{dt}$ stacked in a vector $\boldsymbol{y} \in \mathbb{R}^L$ where $L = 6n$ is the number of state variables. We assume independent state noise with variance $\lambda_y = 0.4$ thus with covariance $\boldsymbol{R} = \lambda_y \boldsymbol{I}_L$.

For measurements we take estimated orientation data from our IMUs using a gradient descent AHRS algorithm based

on [12], $\theta \in \mathbb{R}^b$ where b is the number of bar angles available at the given time step and all ranging measures, $\boldsymbol{r} \in \mathbb{R}^a$, where a is the number of ranging measures available at a given time step. We again assume independent noise with $\lambda_\theta = 0.1$ and $\lambda_r = 0.029$ the measurement covariance matrix is then defined as:

$$\boldsymbol{Q} = \begin{bmatrix} \lambda_\theta \boldsymbol{I}_b & \boldsymbol{0} \\ \boldsymbol{0} & \lambda_r \boldsymbol{I}_a \end{bmatrix}.$$

These user defined variables are then used within the framework of our UKF to forward propagate both the current expected value of the state as well as its covariance. Fig. 4 shows an overview of our complete state estimation setup.

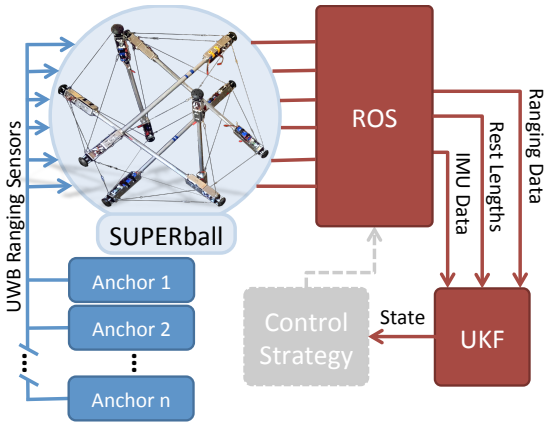

Fig. 4: Block diagram of data flow within the system. Red signals are passed as ROS messages and blue signals are passed using the ranging modules. Note that each rod contains two ranging sensors located at each end of the rod. The gray control strategy block represents a to-be-designed state-feedback control strategy.

V. Filter Evaluation

A. Experimental Setup

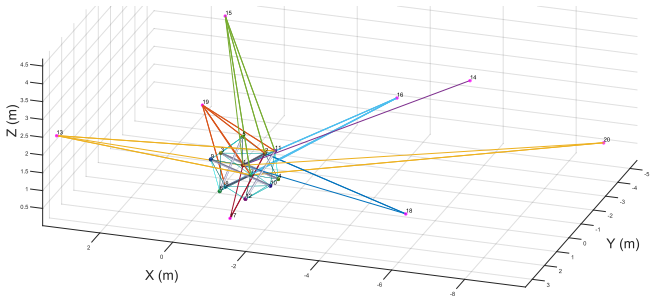

Fig. 5: Visualization of the UKF output. SUPERball sits in the middle of the plot surrounded by 8 ranging base stations. Lines between the robot and the base stations indicate valid ranging measures during this timestep.

To evaluate the performance of the UKF, we used the eight "fixed anchor" ranging base stations calibrated as detailed in Section III-C. Each end cap of SUPERball was then able to get a distance measurement to each base station. This information was sent over ROS along with IMU data (yaw,pitch,roll) and cable rest lengths to the UKF. The base stations were placed in a pattern to cover an area of approximately 91 m^2. Each base station's relative location to each other may be seen in Fig. 5. SUPERball and the base stations were then used to show the UKF tracking a local trajectory of end caps and a global trajectory of the robotic system. In each of these experiments, the UKF was allowed time to settle from initial conditions upon starting the filter. This ensured that any erroneous states due to poor initial conditioning did not affect the filter's overall performance.

B. Local Trajectory Tracking

In order to track a local trajectory, SUPERball remained stationary while two of its actuators tracked phase shifted stepwise sinusoidal patterns. During the period of actuation, two end cap trajectories were tracked on SUPERball and compared to the trajectory outputs of the UKF. One end cap was directly connected to an actuated cable (end cap 2), while the other end cap had no actuated cables affixed to it (end cap 1). To obtain a ground truth for the position trajectory, a camera that measured the position of each end cap was positioned next to the robot. Both end caps started at the same relative height and the majority of movement of both fell within the plane parallel to the camera. Fig. 6 shows the measured and UKF global positions of the two end caps through time.

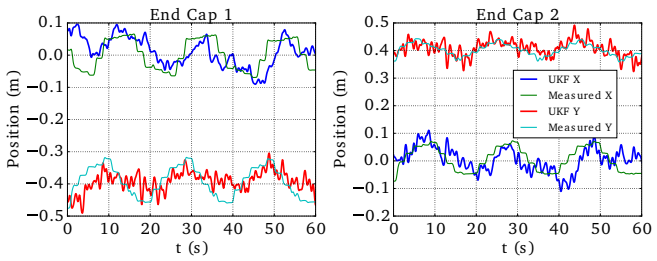

Fig. 6: Position plotted through time for both end cap 1 and end cap 2. The thin line represents the position output measured by the camera tracking system, and the bold line represents the position output from the UKF filter. As expected, there is a time domain lag between the measured and estimated positions.

C. Global Trajectory Tracking

For global trajectory tracking, SUPERball was actuated to induce a transition from one base triangle rolling through to another base triangle as presented in [6]. Ground truth for this experiment was ascertained by marking and measuring the positions of each base triangle's end caps before and after a face transition. We evaluate 4 settings of the state estimator. *Full*: The state estimator as described in Section IV with all IMU and ranging sensors. *no IMU*: Only the ranging sensors are enabled. *full w. cst. offset*: Same as *full*, but the offsets $\boldsymbol{o}$ are set to a constant instead of optimized individually. *4 base station ranging sensors*: 50% of the base station

ranging sensors are disabled. The results of this experiment are presented in Fig. 7 and 8.

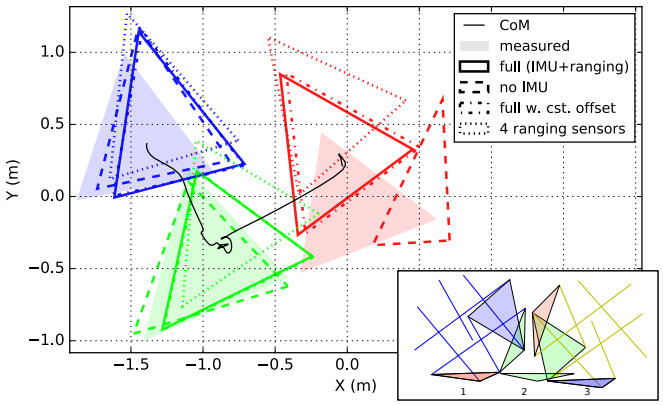

Fig. 7: Top down view of the triangular faces to which the robot transitions during the global trajectory tracking experiment for various setting of the state estimator. The small inset illustrates the movement of the robot. The line shows the estimated center of mass (CoM) using the *full* settings. Finding the initial position (origin) is hard for all settings, and without the IMUs the estimator does not find the correct initial face. After a first roll, tracking becomes more accurate. The offsets $\boldsymbol{o}$ have a minimal impact, which indicates that our calibration routine is sufficiently accurate.

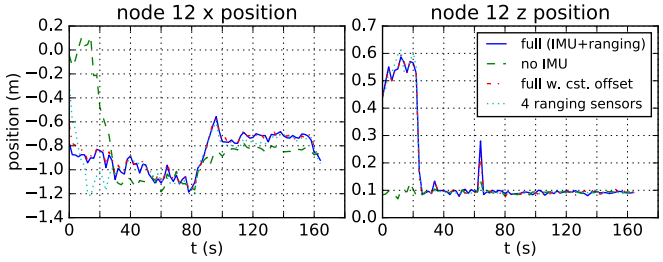

Fig. 8: X and Y position of end cap 12 as a function of time for the various estimator settings. The end cap was initially off the ground and touches the ground after the first roll. This is not tracked correctly when the IMUs are disabled. The system works as expected when 4 base stations ranging sensors are disabled, but with slower convergence and more noise on the robot's position. Around 60 s there's a spurious IMU value from which the state estimator recovers.

VI. Conclusion

We have introduced a state estimation approach for tensegrity robots based on ultra wideband ranging sensors, inertial measurements, and actuator states. An unscented Kalman filter was used to combine these sensor and state observations. While this is a fairly common approach to state estimation, our algorithm is robust to measurement noise and significant amounts of missing data. To verify these statements, we evaluated our method on two tasks: rolling and stationary deformations of SUPERball.

Most of the current control approaches for tensegrity robots rely on position tracking for either motion planning or performance evaluation. Hence, our main contribution to the field of tensegrity robotics is that this work will finally allow various proposed control algorithms for tensegrity robots to be transferred from the simulation domain to hardware.

We believe that - in this context - our setup is a viable alternative to more established external solutions, such as motion capture systems. In particular, our approach is low-cost, self-calibrating and only relies on autonomous anchors. These last two qualities are particularly attractive for future space missions. We do not yet achieve the accuracy provided by commercial motion capture systems. However, this is not a priority at this point as we mainly are focused on large displacements by rolling of SUPERball.

The logical next step is to test our state estimator when SUPERball is moving around in a larger space. The RoverScape at NASA Ames - a football field sized outdoor rover testbed - is the ideal candidate test area for this.

A related future goal is to add support for an incremental number of ranging anchors. This will allow SUPERball to explore uncharted terrain by dropping beacons when the uncertainty of its current position increases.

Acknowledgments

We appreciate the support, ideas, and feedback from members of the Dynamic Tensegrity Robotics Lab. We are also grateful to Terry Fong and the NASA Ames Intelligent Robotics Group.

References

[1] J. Rieffel, R. Stuk, F. Valero Cuevas, and H. Lipson, "Locomotion of a tensegrity robot via dynamically coupled modules," *Proceedings of the International Conference on Morphological Computation*, 2007.

[2] C. Paul, J. W. Roberts, H. Lipson, and F. J. Valero Cuevas, "Gait production in a tensegrity based robot," *International Conference on Advanced Robotics*, pp. 216–222, 2005.

[3] A. Graells Rovira and J. M. Mirats-Tur, "Control and simulation of a tensegrity-based mobile robot," *Robotics and Autonomous Systems*, vol. 57, no. 5, pp. 526–535, May 2009.

[4] C. Sultan, M. Corless, and R. Skelton, "Tensegrity flight simulator," *Journal of Guidance, Control, and Dynamics*, vol. 23, no. 6, pp. 1055–1064, 2000.

[5] J. Bruce, K. Caluwaerts, A. Iscen, A. P. Sabelhaus, and V. SunSpiral, "Design and evolution of a modular tensegrity robot platform," in *ICRA*, May 2014, pp. 3483–3489.

[6] A. P. Sabelhaus, J. Bruce, K. Caluwaerts, P. Manovi, R. F. Firoozi, S. Dobi, A. M. Agogino, and V. SunSpiral, "System design and locomotion of SUPERball, an untethered tensegrity robot," in *ICRA*, 2015, pp. 2867–2873.

[7] A. Ledergerber, M. Hamer, and R. D'Andrea, "A robot self-localization system using one-way ultra-wideband communication," in *IROS*, 2015.

[8] DecaWave, "APS011 application note: Sources of error in DW1000 based two-way ranging (TWR) schemes."

[9] E. Wan, R. Van Der Merwe, *et al.*, "The unscented Kalman filter for nonlinear estimation," in *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000*. IEEE, 2000, pp. 153–158.

[10] R. E. Skelton and M. C. Oliveira, *Tensegrity systems*. Springer, 2009.

[11] K. Yamane and Y. Nakamura, "Stable penalty-based model of frictional contacts," in *ICRA*, 2006, pp. 1904–1909.

[12] S. O. Madgwick, A. J. Harrison, and R. Vaidyanathan, "Estimation of imu and marg orientation using a gradient descent algorithm," in *IEEE International Conference on Rehabilitation Robotics (ICORR2011)*, 2011, pp. 1–7.