Proceedings of the ASME 2013 Dynamic Systems and Control Conference
DSCC2013
October 21-23, 2013, Palo Alto, California, USA

DSCC2013-3873

ROBUST STATE ESTIMATION WITH REDUNDANT PROPRIOCEPTIVE SENSORS

David Rollinson, Howie Choset and Stephen Tully
The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213
Email: drollins@cs.cmu.edu

ABSTRACT

We present a framework for robust estimation of the configuration of an articulated robot using a large number of redundant proprioceptive sensors (encoders, gyros, accelerometers) distributed throughout the robot. Our method uses an Unscented Kalman Filter (UKF) to fuse the robot's sensor measurements. The filter estimates the angle of each joint of the robot, enabling the accurate estimation of the robot's kinematics even if not all modules report sensor readings. Additionally, a novel outlier detection method allows the the filter to be robust to corrupted accelerometer and gyro data.

INTRODUCTION

Snake robots are a class of hyper-redundant mechanisms [1] consisting of kinematically constrained links chained together in series. The many degrees of freedom of these robots give them the potential to navigate a wide range of environments and provide unique challenges in the fields of robotic locomotion and control. There is a wide variety of snake-like robots and research from groups around the world which is thoroughly surveyed by Transeth and Pettersen in [2]. Our group has developed modular snake robots that rely solely on changing their shape to locomote through their environment [3]. These snake robots consist of single degree-of-freedom (DOF) modules which achieve 3D manipulation and mobility by alternately orienting the joints in the lateral and dorsal planes of the robot (Fig. 1).

Remotely operating these robots can be difficult, since the operator lacks a good overall view of the robot and its environment [4]. To help provide better feedback and improve an operator's situational awareness, we have integrated MEMS accelerometers and gyros into each of the robot's modules. Prior work from our group has already used an extended Kalman filter (EKF) to fuse these distributed sensors and achieve an estimate of the robot's pose [5]. By using knowledge of the robot's cyclic controller (gait) and an averaged body frame that we call the *virtual chassis*, we were able to estimate the robot's orientation, even when it undergoes highly dynamic motions.

Figure 1: The most recent iteration of the Carnegie Mellon modular snake robot, the *Unified Snake*. Typically, the robot consists of 16 single degree-of-freedom modules (shown here), with additional unactuated modules at the head and tail.

Unfortunately, our previous work has limitations in terms of its robustness in real-world field use. Frequent communication dropouts or corrupted data from the modules would sometimes cause the EKF to diverge. Additionally, the need for the state estimator to have explicit knowledge of the robot's gait equation means that it has to be tightly integrated with the gait framework that we use for control. This work addresses these issues with two contributions. First, we formulate the state estimation problem in a way that leverages redundancies in the proprioceptive

information provided by the robot's joint angle encoders and inertial sensors. Second, we are introducing a novel outlier detector that can identify corrupted measurement data with a minimal amount of tuning.

PRIOR WORK

Fusing redundant data in robotics systems is a topic with a wealth of prior work [6]. A common method of fusing redundant and complimentary sensor data is the Kalman filter [7]. However, the Kalman filter only applies to linear systems, and there are a number of extensions that allow it to be applied to non-linear systems. The work in this paper includes the implementation of three state estimators, an extended Kalman Filter (EKF), an unscented Kalman filter (UKF), and a spherical simplex unscented Kalman filter (SSUKF). The EKF extends the Kalman filter to non-linear systems by linearizing the system at the current state estimate at each timestep [8]. The UKF is a method that attempts to address the problems inherent in linearization. It uses a deterministic sampling technique that relies on *sigma points* to directly calculate the mean and covariance statistics that are necessary for the filter [9]. The SSUKF is a variant of the UKF that uses fewer sigma points, making it more computationally efficient [10, 11].

All forms of Kalman filters have problems in the presence of outliers, due to their underlying assumption of Gaussian noise in the state estimate and measurement observations. This is particularly problematic in robotics, where real-world effects like unmodelled disturbances, faulty sensors, or failed actuators can frequently produce outliers. Because of this, there have been a number of modifications to the Kalman filter to make it more robust to outliers at the cost of more computation and complexity. Some techniques require noise to be modeled as heavy-tailed distributions [12]. Others use a weighted least-squares approach learning the states and noise models online [13]. Ting et al [14] have developed a Kalman filter that is robust to outliers and requires very little tuning. However, their method relies on estimating the linear system dynamics, and in our case we have time-varying non-linear process and measurement models.

Perhaps the most widely used methods of outlier detection are ones that threshold on the Mahalanobis distance of the measurement residuals during the filter update [14, 15]. If the Mahalanobis distance is sufficiently large, the measurement vector at the current iteration is assumed to contain outliers, and the update step is skipped. Tuning this threshold can be difficult, especially in systems that are highly dynamic or modeled poorly.

STATE ESTIMATION

Snake robots are unique in both their locomotive capabilities as well as their challenges to estimation and control. Previous work from our group demonstrated accurate estimation of a snake robot's orientation using the robot's proprioceptive sensors [5] and an EKF. This work extends and improves those methods, so that they are more suitable for use in the field. To explore more state of the art techniques, we implemented a UKF and SSUKF in addition the conventional EKF. All three filters used the same process and measurement models, as well as the same values for process and measurement noises.

Kalman Filter

All of the filters presented in this paper extend the Kalman filter to non-linear systems. At the heart of the filter are the state estimate, $\mathbf{x}_k$, its covariance, $\mathbf{P}_k$, the robot's sensor measurements, $\mathbf{z}_k$, and the non-linear process and measurement models. The process model, f, is a function that predicts the state of the robot given the state at a previous timestep,

$$\mathbf{x}_k = f(\mathbf{x}_{k-1}, \Delta t). \tag{1}$$

The measurement model, h, is a function that predicts sensor measurements given the state of the robot,

$$\hat{\mathbf{z}}_k = h(\mathbf{x}_k). \tag{2}$$

In the EKF, the state estimate is propagated through these non-linear functions, and the functions are linearized at each iteration of the filter, resulting in the Jacobians $\mathbf{F}_k$ and $\mathbf{H}_k$ that correspond respectively to the functions f and h. This allows the propagation of the the state covariance between timesteps and the processing of measurements into the state estimate.

In the UKF and SSUKF, a deterministically sampled set of sigma points are chosen, propagated through the models, and then averaged in a way that directly calculates the state mean and covariance. For an n-dimensional state vector, the UKF uses $2n+1$ sigma points, and the SSUKF uses $n+2$ sigma points.

In all of these filters, a process noise matrix, $\mathbf{Q}$, is added to the state covariance at every prediction step. A measurement noise matrix, $\mathbf{R}$, is similarly used to tune the relative confidence in sensor measurements. The parameters in $\mathbf{Q}$ and $\mathbf{R}$ are used to tune the relative confidence of different states and measurements with respect to each other. The process model noise is assumed to be additive, and is incorporated before the state covariance is pushed through the process model at each iteration [16, 17].

Choice of Body Frame

A key assumption in all of the following sections is that all of the calculations of the robot's kinematics are carried out using an averaged body frame, that we call the *virtual chassis*. The virtual chassis is a body frame that is aligned with the principle components of the robot's overall shape, as shown in Fig. 2. The calculation of this body frame is performed as part of the measurement model at every iteration of the filter, using SVD to identify the principle component of the robot's shape [5].

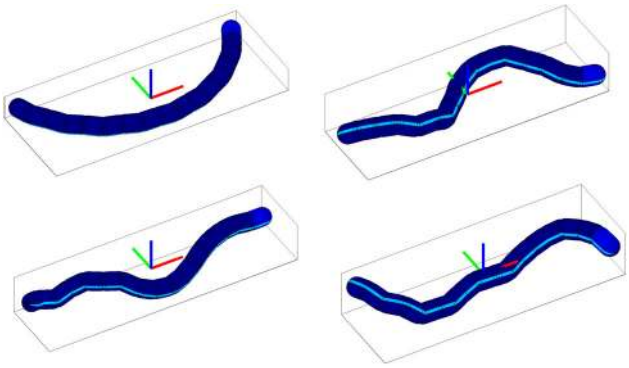

Figure 2: An example of the virtual chassis body frame for various shapes of the snake robot. Because the body frame is aligned with the principle components of the robot's shape, it helps separate the robot's internal shape changes from its external motions.

The virtual chassis body frame has the advantage that it approximately separates the robot's internal shape changes from its external motions in the world, enabling more accurate and stable state estimation with generic constant velocity process models. Additionally, the state of the snake robot in this body frame is more intuitive to the operator, since the notions of up-down and left-right are more aligned with the overall shape of the robot.

State Vector

The state of the filter tracks the robot's orientation, its inertial frame acceleration, and its shape variables,

$$\mathbf{x}_k = [\ \mathbf{a}_k\ \ \mathbf{q}_k\ \ \boldsymbol{\omega}_k\ \ \boldsymbol{\theta}_k\ \ \dot{\boldsymbol{\theta}}_k\ ]^T \tag{3}$$

The first part of the state vector describe the robot's inertial state and orientation: $\mathbf{a} = [\ a_x\ \ a_y\ \ a_z\ ]$ is robot's world frame acceleration, $\mathbf{q} = [\ q_1\ \ q_2\ \ q_3\ \ q_4\ ]$ is the world frame orientation, and $\boldsymbol{\omega} = [\ \omega_x\ \ \omega_y\ \ \omega_z\ ]$ are the robot's body frame angular velocities.

The second part describes the robot's kinematic state. Assuming a robot with m links, the vector joint angles, $\boldsymbol{\theta} = [\ \theta_1 \cdots \theta_m\ ]$ describe the shape of the robot, and the vector of joint velocities, $\dot{\boldsymbol{\theta}} = [\ \dot{\theta}_1 \cdots \dot{\theta}_m\ ]$ provide a first-order estimate of how the robot's shape changes over time.

Having an estimate of the robot's shape in the filter state means that we can always perform a prediction and an update, even if the robot does not report all (or even any) of its joint angles at a given timestep. Furthermore, this formulation of the state means the the joint angles are being redundantly estimated both directly, by observing the joint angles, and indirectly, by observing the inertial readings at adjacent modules. In the Results section, we show that this allows us to track the angles of modules that fail to provide feedback, even for extended periods of time.

Process Model

The process model evolves the robot's state over time. Overall, this model can be thought of as a generic constant velocity model that does not explicitly model the interaction of the world. In the following section, the hat notation indicates the states in the prediction step of the filter, before the update that incorporates measurement observations.

Acceleration is estimated in a world frame assumed to be damped according to the update

$$\hat{\mathbf{a}}_k = e^{-\tau \Delta t} \mathbf{a}_{k-1}. \tag{4}$$

In our experience acceleration must be strongly damped ($\tau > 20$) for the filter to remain stable, since the signal from the accelerometers is noisy and dominated by gravitational acceleration. Previous iterations of our state estimators [18] assumed zero world frame acceleration of the robot. This resulted in good performance at slow speeds, but became problematic when the snake robot falls or experiences sudden changes in pose.

The quaternion representing the orientation of the robot is updated based on the estimated angular velocities at that timestep. We perform a discrete-time update developed by van der Merwe et al. [19]

$$\hat{\mathbf{q}}_k = \exp(-\frac{1}{2}\boldsymbol{\Psi}\Delta t)\ \mathbf{q}_{k-1} \tag{5}$$

$$\boldsymbol{\Psi} = \begin{bmatrix} 0 & \omega_x & \omega_y & \omega_z \\ -\omega_x & 0 & -\omega_z & -\omega_y \\ -\omega_y & \omega_z & 0 & -\omega_x \\ -\omega_z & -\omega_y & \omega_x & 0 \end{bmatrix}. \tag{6}$$

The body frame angular velocities of the robot, ω, are assumed to be constant across timesteps,

$$\hat{\boldsymbol{\omega}}_k = \boldsymbol{\omega}_{k-1}. \tag{7}$$

The kinematic shape of the robot is also estimated in the state of the filter. This is achieved by estimating both the snake robot's joint angles and their angular velocities. Joint angles are updated by their estimated velocities,

$$\hat{\boldsymbol{\theta}}_k = \boldsymbol{\theta}_{k-1} + \dot{\boldsymbol{\theta}}_{k-1}\Delta t. \tag{8}$$

The angular velocities of the joints are estimated by a weighted combination of the estimated velocities from the last timestep, $\dot{\boldsymbol{\theta}}_{k-1}$, and the commanded angular velocities at the current timestep, $\dot{\boldsymbol{\theta}}_k^{cmd}$,

$$\dot{\boldsymbol{\theta}}_k = (1-\lambda)\dot{\boldsymbol{\theta}}_{k-1} + \lambda\dot{\boldsymbol{\theta}}_k^{cmd}. \tag{9}$$

The weighting parameter λ ranges from 0 to 1 and controls how much of the commanded angular velocity is mixed into the state. A value of 1 essentially overwrites the estimated joint angle velocities at each iteration, whereas a value of 0 turns the filter into a constant-velocity model that has no knowledge of the robot's controls. Since the robot's joints often deviate significantly from their commanded trajectories, this parameter is set relatively low, around 0.25.

Measurement Vector

Our latest snake robot provides feedback measurements from single-axis joint angle encoders, 3-axis accelerometers and a 3-axis gyros located in each module. This means that the vector of measurements in the filter has $7m$ dimensions, where m is the total number of modules in the robot,

$$\mathbf{z}_k = [\ \boldsymbol{\phi}_k\ \ \boldsymbol{\alpha}_k\ \ \boldsymbol{\gamma}_k\ ]^T. \tag{10}$$

In (10), each element is a vector containing the measurements of a corresponding sensor type for all the modules throughout the snake robot. $\boldsymbol{\phi}$ is the robot's joint angle measurements from its encoders, $\boldsymbol{\alpha}$ is the accelerometer measurements, and $\boldsymbol{\gamma}$ is the gyroscope measurements.

Measurement Model

The measurement model predicts measurements at the current timestep, given the prediction of the of the robot's state. The measurement model is a kinematic model that takes into account the robot's shape variables and its inertial state. In the following section, the superscript indicates the module for which a measurement is predicted and the hat operator denotes a predicted measurement, rather than a sensed measurement from the robot. The superscript i indicates that the measurements correspond the ith module in the robot.

Expected joint angle measurements are predicted directly from the estimated angles in the state vector (3),

$$\hat{\boldsymbol{\phi}}_k = \hat{\boldsymbol{\theta}}_k. \tag{11}$$

Using the estimated joint angles, $\boldsymbol{\theta}$, and joint angle velocities, $\dot{\boldsymbol{\theta}}$, accelerometer and gyro measurements for each module can be predicted using the finite time-differencing approach detailed below.

Accelerometers have the property that they measure an acceleration due to gravity in addition to lateral acceleration due to motion. For this reason our model treats these two sources of sensed acceleration separately and sums them to generate the predicted accelerometer measurement for each module,

$$\hat{\boldsymbol{\alpha}}_k^i = \hat{\mathbf{a}}^i_{\text{gravity}} + \hat{\mathbf{a}}^i_{\text{motion}}. \tag{12}$$

Acceleration due to gravity is predicted by transforming the estimated gravity vector $\mathbf{g}$ into the frame of each module

$$\hat{\mathbf{a}}^i_{\text{gravity}} = \mathbf{W}^i_k \mathbf{V}_k \mathbf{g} \tag{13}$$

where $\mathbf{W}$ is the rotation matrix that describes the orientation of module i in the body frame, and $\mathbf{V}$ is the rotation matrix representation of the quaternion pose $\mathbf{q}$ in the state vector (5) or (6).

Acceleration due to a module's motion is further split into two components,

$$\hat{\mathbf{a}}^i_{\text{motion}} = \hat{\mathbf{a}}^i_{\text{internal}} + \mathbf{W}^i_k \mathbf{V}_k \hat{\mathbf{a}} \tag{14}$$

Acceleration due to the robot's internal shape changes in the body frame, $\hat{\mathbf{a}}^i_{\text{internal}}$, is predicted by double-differentiating the position of the module in the virtual chassis body frame. Finally, the estimated world frame acceleration of the entire robot is incorporated by rotating the world frame acceleration $\hat{\mathbf{a}}$ from filter's state estimate into the frame of each module.

The predicted gyro measurements for each module are generated by differentiating the orientation of the robot at two nearby timesteps. If $\mathbf{W}^i_k$ and $\mathbf{W}^i_{k-1}$ are rotation matrices that describe the orientations of module i in the body frame at two timesteps, then gyro measurements due to the internal motion of the gait at two timesteps, k and $k-1$ can be approximated by

$$\begin{bmatrix} 1 & -\bar{\omega}^i_z & \bar{\omega}^i_y \\ \bar{\omega}^i_z & 1 & -\bar{\omega}^i_x \\ -\bar{\omega}^i_y & \bar{\omega}^i_x & 1 \end{bmatrix} \approx \frac{\mathbf{W}^i_k (\mathbf{W}^i_{k-1})^{-1}}{\Delta t}. \tag{15}$$

The complete prediction for each gyro is the angular velocity from (15) plus the robot's body frame angular velocity from the current state estimate, (7), rotated into the coordinate from of each module using $\mathbf{W}^i_{\text{t}}$

$$\hat{\boldsymbol{\gamma}}^i_k = \bar{\boldsymbol{\omega}}^i + (\mathbf{W}^i_k)^{-1} \hat{\boldsymbol{\omega}}_k \tag{16}$$

Using Partial Measurement Data

Due to noise in the robot's communications, around 2% of our sensor data is missing at a given update step. If intermittent connections are present in the robot this can increase even further. Furthermore, when using our robots aggressively in the

field, modules frequently reset due to their electrical protection circuitry. In these cases an individual module may drop out for 2-3 seconds before rebooting. During this time the module's joint is rotating freely with no direct measurement of its joint angle. Thus, it is desirable to have a method to identify and mitigate these problems.

To handle these issues, our communications are set up so that missing data is reported as `NaN`, or not a number. In the filter this measurement data is over-written with a value of 0, and that measurement's corresponding value in the additive measurement noise matrix, $\mathbf{R}$, is increased to 10^6 for that timestep. This causes the filter to effectively ignore the measurement during the update step and is simpler to implement than dynamically resizing the covariance and state at every iteration.

OUTLIER DETECTION

Because our robots are undergoing constant development, testing, and maintenance, it is not uncommon for sensors to become unresponsive or miscalibrated. And because erroneous measurements from an unresponsive sensor can severely disrupt the state estimate [14, 15], it is beneficial to detect such outliers from the observed sensor data.

Algorithm

The Mahalanobis distance for the residual error between the predicted measurement vector, $\hat{\mathbf{z}}_k = \mathbf{h}(\hat{\mathbf{x}}_{k|k-1})$, and the observed measurement vector, $\mathbf{z}_k$,

$$d_k = (\hat{\mathbf{z}}_k - \mathbf{z}_k)^T \mathbf{S}_k^{-1} (\hat{\mathbf{z}}_k - \mathbf{z}_k),$$

gives an indication of the likelihood of the measurement, where $\mathbf{S}_k$ is the innovation covariance.

To detect outliers, our method computes, for each sensor s, a Mahalanobis distance that excludes sensor s from the measurement vector. To exclude these elements, we define the following selection matrix,

$$\mathbf{Y}_s = \begin{bmatrix} \mathbf{I}_{MxM} & \mathbf{0}_{Mx3} & \mathbf{0}_{MxN} \\ \mathbf{0}_{NxM} & \mathbf{0}_{Nx3} & \mathbf{I}_{NxN} \end{bmatrix},$$

where M is the number of elements in the measurement vector $\mathbf{z}_k$ that precede sensor s, N is the number of elements in $\mathbf{z}_k$ that follow sensor s, and 3 is the number of elements in the measurement vector that correspond to sensor s.

The Mahalanobis distance associated with excluding sensor s can be written as follows,

$$d_k^s = (\hat{\mathbf{z}}_k^s - \mathbf{z}_k^s)^T \left[\mathbf{S}_k^s\right]^{-1} (\hat{\mathbf{z}}_k^s - \mathbf{z}_k^s), \tag{17}$$

where $\mathbf{z}_k^s = \mathbf{Y}_s \mathbf{z}_k$, $\hat{\mathbf{z}}_k^s = \mathbf{Y}_s \hat{\mathbf{z}}_k$, and $\mathbf{S}_k^s = \mathbf{Y}_s \mathbf{S}_k \mathbf{Y}_s^T$.

For each sensor s, a Mahalanobis distance d_k^s that is significantly smaller than d_k means that the excluded component is likely to be an outlier. For our implementation, we initially choose to discard a fixed number of sensors from the measurement vector (specifically the 4 that are more likely to be outliers based on d_k^s being small). After discarding the four most likely outliers, we calculate the mean, μ_k, and standard deviation (SD), σ_k, of the remaining Mahalanobis distances.

Finally, for each sensor s, a new metric is computed that compares the Mahalanobis distance d_k^s to the mean and variance of the Mahalanobis distances of the presumed inliers,

$$w_k^s = \frac{\left(d_k^s - \mu_k\right)^2}{\sigma_k^2}.$$

We then consider all sensors, including the initially discarded sensors, and decide whether each one is an inlier or outlier by thresholding the metric w_k^s at some level, ξ. When w_k^s is large, the sensor is likely to be an outlier and when this metric is small, the sensor measurement is likely to be an inlier. If the measurement is determined to be an outlier, the measurement's corresponding element in $\mathbf{R}$ is set to 10^6, just as is done for missing data.

In a sense, w_k^s is a Mahalanobis distance of Mahalanobis distances. By thresholding on this statistic, we are able to pick a single static threshold that is valid at all times. Compared to thresholding on the Mahalanobis distance alone, outliers tend to be extremely obvious, often greater than 100 standard deviations from the mean. Setting the detection threshold, ξ, to a value between 10 and 50 has been shown to work well for our system, regardless of sensor type or the robot's motion. Finally, this algorithm has the benefit that it ensures an upper limit on the number of sensors that can be flagged as outliers, as long as ξ is not set too low (setting $\xi > 3$).

Efficient Implementation

One drawback of this outlier detection algorithm, as written, is the need to invert $\mathbf{S}_k^s$ for each sensor in order to compute d_k^s as computed in (17). For our 16-link snake robot that has 2 inertial sensors per module, the accelerometer and the gyro, performing this for a typical 16-link robot would require 32 inversions of a 109-by-109 matrix. This is a significant computational expense during real-time operation.

Ideally, it would be beneficial to compute the inverse of the innovation covariance, $\mathbf{S}_k^{-1}$, only once and then to somehow efficiently infer, for each sensor s, the matrix $\left[\mathbf{S}_k^s\right]^{-1}$. By definition, $\left[\mathbf{S}_k^s\right]^{-1} = \left[\mathbf{Y}_s \mathbf{S}_k \mathbf{Y}_s^T\right]^{-1}$. But unfortunately, $\left[\mathbf{S}_k^s\right]^{-1} \neq \mathbf{Y}_s \mathbf{S}_k^{-1} \mathbf{Y}_s^T$, otherwise we would compute $\left[\mathbf{S}_k^s\right]^{-1}$ directly from $\mathbf{S}_k^{-1}$. This is not possible because $\mathbf{Y}_s$ is not an orthogonal matrix.

Instead, we can leverage the Woodbury matrix identity [20], to perform a low-rank correction to the relatively large matrix

$\mathbf{S}_k^{-1}$. This allows us to efficiently obtain the matrix $\left[\mathbf{S}_k^s\right]^{-1}$, which we require for computing the Mahalanobis distance in Eq. (17). First, we define the following matrix that can be used to rearrange the elements of the innovation so that the elements corresponding with sensor s are last,

$$\mathbf{G}_s = \begin{bmatrix} \longleftarrow & \mathbf{Y}_s & \longrightarrow \\ \mathbf{0}_{3xM} & \mathbf{I}_{3x3} & \mathbf{0}_{3xN} \end{bmatrix},$$

Using this matrix, we can rearrange the inverse of the innovation covariance matrix as follows,

$$[\mathbf{S}'_k]^{-1} = \left[\mathbf{G}_s\mathbf{S}_k\mathbf{G}_s^T\right]^{-1} = \mathbf{G}_s\mathbf{S}_k^{-1}\mathbf{G}_s^T = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^T & \mathbf{C} \end{bmatrix}.$$

Using the Woodbury matrix identity to invert the matrix $[\mathbf{S}'_k]^{-1}$, we obtain,

$$\mathbf{S}'_k = \begin{bmatrix} (\mathbf{A}-\mathbf{B}\mathbf{C}^{-1}\mathbf{B}^T)^{-1} & \sim \\ \sim & \sim \end{bmatrix}.$$

Above, the $\sim$ indicates regions of the matrix from the Woodbury matrix identity that are omitted for clarity. Since the upper left component of $\mathbf{S}'_k$ is equal to the matrix $\mathbf{S}_k^s$, due the existence of $\mathbf{Y}_s$ in $\mathbf{G}_s$, we can simply invert the upper left component of $\mathbf{S}'_k$,

$$[\mathbf{S}_k^s]^{-1} = \mathbf{A}-\mathbf{B}\mathbf{C}^{-1}\mathbf{B}^T. \tag{18}$$

In (18), $\mathbf{C}$ is the inverted covariance of the sensor being excluded. This form only requires the inversion of $\mathbf{C}$, which in our case is 3-by-3. Thus, we can efficiently calculate the inverse of the innovation covariance, $\left[\mathbf{S}_k^s\right]^{-1}$ for each test of a sensor s by performing a small update to the full inverse of the innovation covariance matrix $\mathbf{S}_k^{-1}$. A summary of the algorithm that we use for outlier detection is provided in Alg. 1.

Algorithm 1 Outlier Detection Procedure

for each s **do**
  $\mathbf{S}_k'^{-1} \leftarrow \mathbf{G}_s\mathbf{S}_k^{-1}\mathbf{G}_s^T$
  $[\mathbf{A},\mathbf{B},\mathbf{C}] \leftarrow \text{extractBlocks}\left(\mathbf{S}_k'^{-1}\right)$
  $\left[\mathbf{S}_k^s\right]^{-1} \leftarrow \mathbf{A}-\mathbf{B}\mathbf{C}^{-1}\mathbf{B}^T$
  $\mathbf{z}_k^s = \mathbf{Y}_s\mathbf{z}_k$
  $\hat{\mathbf{z}}_k^s = \mathbf{Y}_s\hat{\mathbf{z}}_k$
  $d_k^s = (\hat{\mathbf{z}}_k^s - \mathbf{z}_k^s)^T \left[\mathbf{S}_k^s\right]^{-1} (\hat{\mathbf{z}}_k^s - \mathbf{z}_k^s)$
end for
$[\mu_k, \sigma_k] \leftarrow \text{statistics\_of\_inliers}\left(d_k^s \text{ for all } s\right)$
for each s **do**
  $w_k^s \leftarrow \frac{\left(d_k^s-\mu_k\right)^2}{\sigma_k^2}$
  if $w_k^s > \xi$ **then**
    Mark s **as outlier**
  end if
end for

EXPERIMENT

To test the accuracy of the different state estimators, multiple trials of the snake robot were performed in a Vicon motion capture system. During these trials, the robot was remotely controlled through a wide variety of motions, some of which were quite fast and abrupt. The pose of the head module was tracked by the motion capture system to provide ground truth. This was then compared to the filter's estimate for pose of the head module, which is dependent on the estimate of the entire state of the robot. Figure 4 shows a montage of the robot during one of these trials.

To demonstrate the advantages of redundant state formulation, we simulated missing data and complete module dropouts. To test the outlier detection algorithm, we simulated corrupted data on the robot's inertial sensors similar to what is seen when a module is poorly calibrated or programmed incorrectly.

Filter Performance Comparison			
Euler Angle Errors (degrees)			
	Roll	Pitch	Yaw
EKF	2.9	3.4	36.3
UKF	2.9	3.4	34.7
SSUKF	3.1	3.3	24.3

Table 1: The accuracy of the SSUKF in predicting the euler angle orientation of the head module of the snake robot. The filter performs well even with half of the robot's data being excluded. Even when 75% of the data is excluded the filter continues to run, although accuracy begins to be degraded.

RESULTS

Overall, the EKF, UKF and SSUKF performed comparably. The filters were all able to run in real time on the feedback data coming from the robot, about 20 Hz. A comparison of the errors of the estimated head Euler angles for each filter is presented in Table 1. This is the averaged error for 3 different motion capture trials where the robot was driven in a wide variety of speeds and directions. The accuracy in yaw is significantly lower than pitch and roll because it is being dead-reckoned based on the filter's integration of estimated angular velocities.

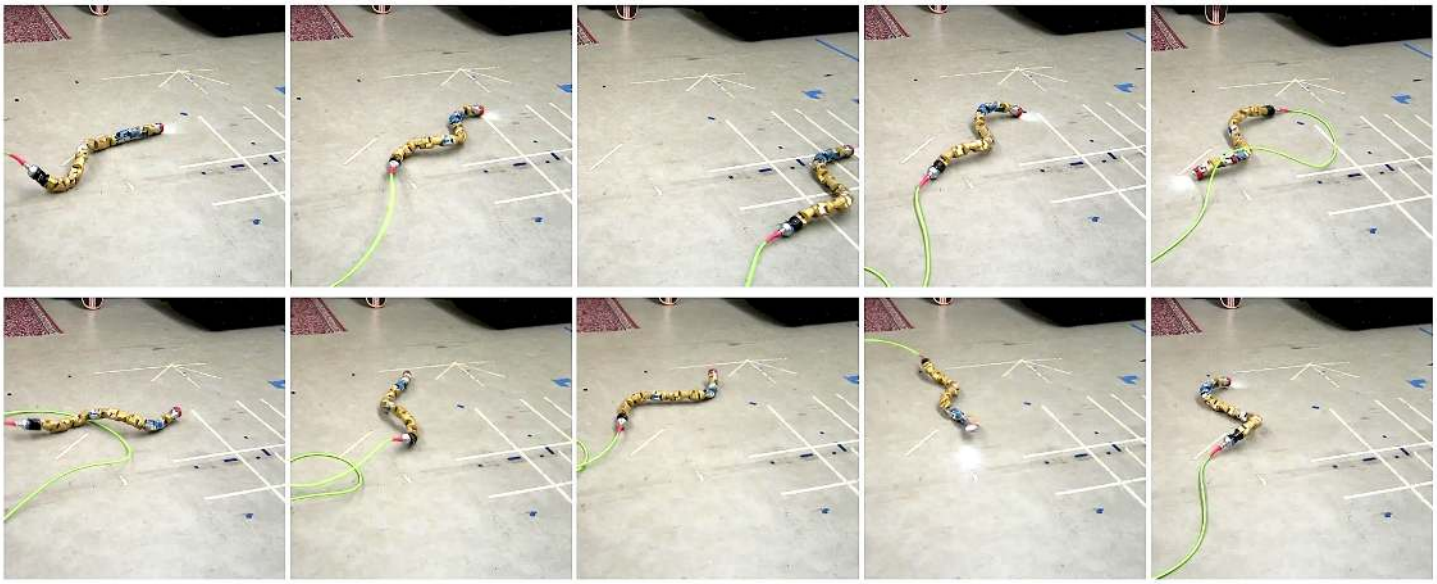

Figure 3: A montage of the snake robot's movements in one of the motion capture trials. The robot does a combination of motions that include slithering forward, sidewinding right and left, and turning in place clockwise and counter-clockwise. This montage corresponds to the plots that are presented in the results section.

SSUKF Performance - Missing Data			
Euler Angle Errors (degrees)			
	Roll	Pitch	Yaw
Baseline	3.1	3.3	24.3
25% Missing	3.7	4.1	31.8
50% Missing	4.5	5.3	27.6
75% Missing	18.4	11.7	84.9

Table 2: The accuracy of the SSUKF in predicting the Euler angle orientation of the head module of the snake robot. The filter performs well even with half of the robot's data being excluded. Even when 75% of the data is excluded the filter continues to run, although accuracy begins to be degraded.

SSUKF Performance - 4 Dropped Modules		
Euler Angle Errors (degrees)		
Roll	Pitch	Yaw
5.5	6.1	44.0

Table 3: The accuracy of the SSUKF in the presence of missing data from some modules for the entire run. These results are for the same trial as shown in Fig. 3. Feedback from a quarter of the snake robot (modules 3,6,7 and 12) was eliminated.

Figure 4 shows a comparison of the estimated head module orientation from the SSUKF compared the motion capture data for one of the trials. To provide a meaningful comparison the quaternion orientations of the head have been converted into Euler angles.

Partial Measurement Data

Under normal circumstances, our snake robot drops about 2% of its data due noise and errors in its communications. To simulate more adverse conditions, we randomly selected and removed higher proportions of the robot's feedback data. The results are summarized in Table 2.

We simulated prolonged module dropouts by eliminating all of the feedback data (joint angles, gyros, accelerometers) from a module in the robot for the entirety of the same data set shown in Fig. 4 and Fig. 5. For the data presented here, the joint angles and inertial sensors were unavailable for the entire run in modules 3, 6, 7 and 12. Even if up to 4 modules were eliminated the filter still converged and estimated the pose of the head reasonably well (Table 3).

Outlier Detection

To test our method of outlier detection, feedback data from the IMUs was corrupted by having its sign reversed. For the data presented in Table 4, modules 3, 6, 7 and 12 had their IMU data corrupted. When running the outlier detection, the filter performs on par with its normal baseline performance. However, it is worth noting that the redundant state formulation is robust enough to remain stable even without the outlier detection, albeit with degraded performance.

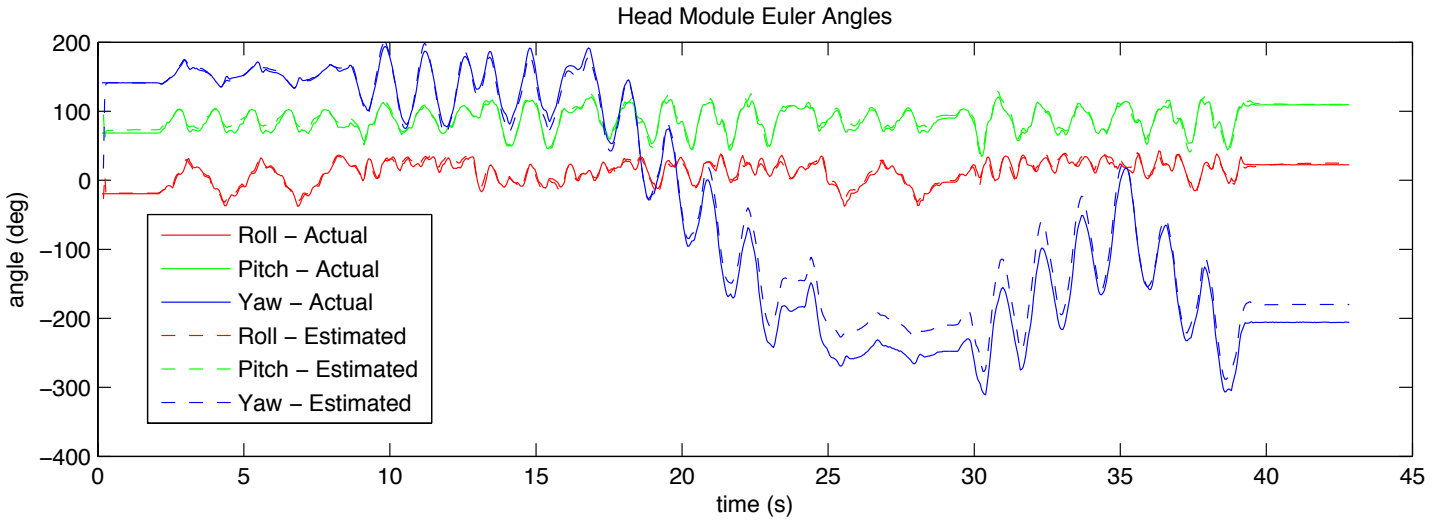

Figure 4: A comparison of the orientation of the head module from motion capture (solid line) compared to the state estimate of the filter (dashed line). The results presented are for the SSUKF, although the UKF and EKF performed similarly.

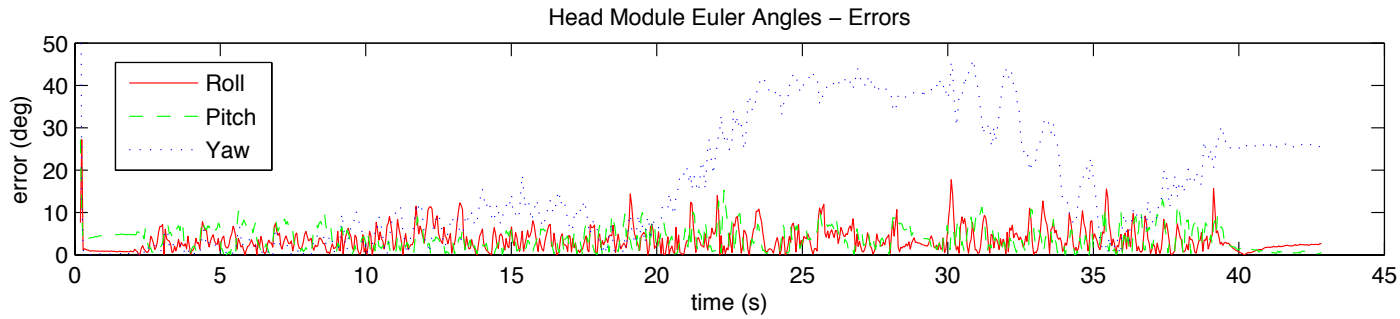

Figure 5: The error of the state estimate compared to the motion capture data for the SSUKF. Note that the error in yaw increases over time because it is being dead-reckoned from gyros.

SSUKF Performance - Corrupted IMU Data			
Euler Angle Errors (degrees)			
	Roll	Pitch	Yaw
Outlier Detection OFF	9.1	6.4	75.6
Outlier Detection ON	3.6	3.7	17.3

Table 4: The accuracy of the SSUKF when accelerometer and gyro feedback from quarter of the robot (modules 3,6,7 and 12) was corrupted. With the outlier detection, the filter performs almost as well as with clean data.

CONCLUSIONS AND FUTURE WORK

This paper implements and evaluates a state estimation framework that leverages the distributed redundant sensing of an articulated robot. It enables an accurate estimate of the robot's orientation even in the presence of extreme sensor degradation and corruption. This kind of state estimator is extremely valuable for snake robots, where the large numbers of modules and sensors increases the likelihood of failure among some of the sensors.

We evaluated the state estimators by comparing the robot's estimate of the orientation of its head module to ground truth data provided by motion capture system. Missing and corrupted measurements were simulated, demonstrating that filter performance degraded only slightly, even in the most severe situations. Three different varieties of Kalman filters were tested (EKF, UKF and SSUKF) all with similar results. For our system, it is our conclusion that the formulation of the estimation problem, filter tuning, and other system-specific considerations are more important factors than the choice of Kalman filter.

The most obvious improvement that can be made to the filter would be to handle the update of the quaternion orientation correctly. In all of the filter implementations presented here the quaternion is normalized after the Kalman update, but this is sub-optimal. There are estimators that have been developed that properly handle the multiplicative correction that quaternions require [17,21]. We plan to implement these to see if they improve the state estimate of the robot's orientation, particularly the drift

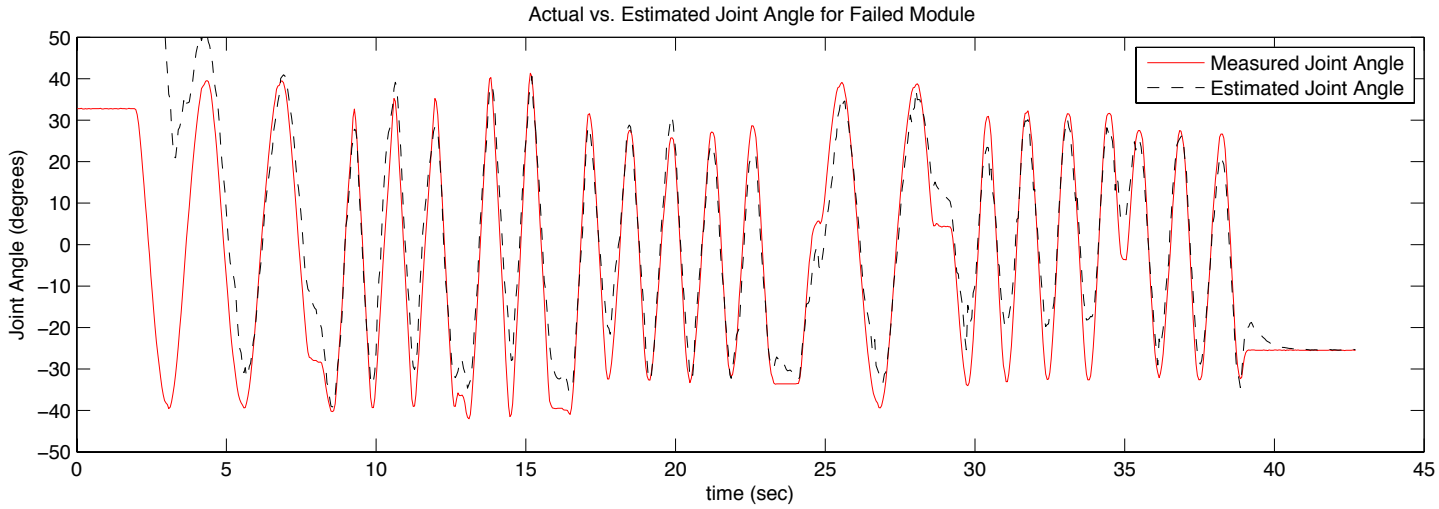

Figure 6: A comparison of the actual and estimated joint angle for module 7 in the snake robot. No feedback was available to the filter for modules 3,6,7 and 12 during the entire trial, but the joint angle is able to be estimated from the feedback from the remaining modules.

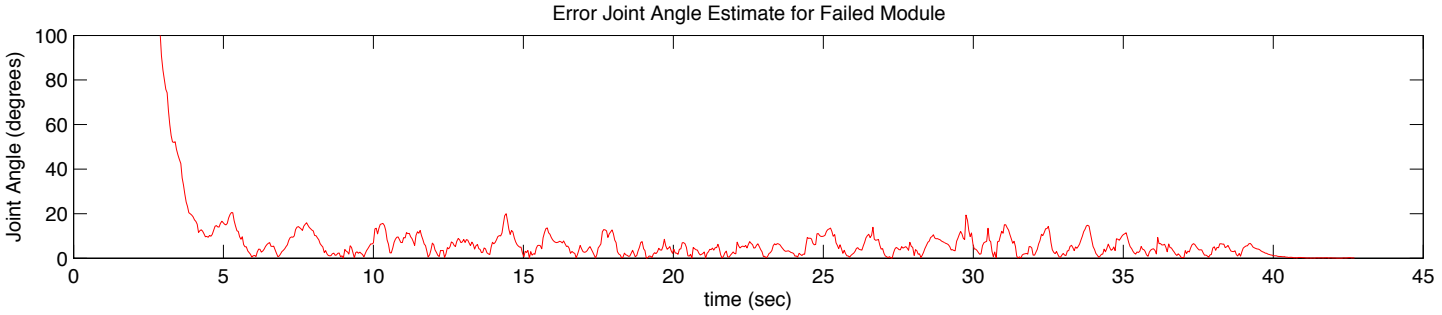

Figure 7: The error of the estimated joint angle. After the filter converges it tracks the joint angle reasonably well, with a mean error of 7 degrees.

in yaw due to it being dead-reckoned by the robot's gyros.

Like our previous work, this filter uses a simplified model that makes minimal assumptions about the way the robot interacts with the world. Perhaps the biggest assumption is embodied in our choice of body frame, using the virtual chassis to attempt to separate the robot's internal shape changes from its external motions. A more elaborate motion model that makes assumptions about ground contact would likely provide better results, as long as ground contact can be accurately estimated.

Incorporating additional sensors that observe the robot's yaw in an inertial frame is desirable. Unfortunately, the magnetic fields from the robot's distributed motors have thus far prevented the use of MEMs magnetometers in the modules. However, the video feed from the head module of the robot remains unused. Implementing various vision algorithms on this video and integrating this into the filter could aid greatly in pose estimation, as has been demonstrated in recent work that estimates motion with the camera and IMU of a smartphone [22, 23].

Finally, it is possible that this redundant state formulation and outlier detection method could be relevant to other modular robots, particularly those that use wireless protocols to communicate between modules.

ACKNOWLEDGEMENTS

The authors would like to Matt Tesch, Matt Travers, Florian Enner, Bruno Hexsel, Nathan Wood, Justine Rembisz and Justin MacEy. This work was supported by the DARPA M3 program.

REFERENCES

[1] Chirikjian, G., and Burdick, J., 1995. "The kinematics of hyper-redundant robot locomotion". *IEEE Transactions on Robotics and Automation,* **11**(6), pp. 781–793.

[2] Transeth, A. A., Pettersen, K. Y., and Liljebäck, P. l., 2009. "A survey on snake robot modeling and locomotion". *Robotica,* **27**(07), Mar., p. 999.

[3] Wright, C., Buchan, A., Brown, B., Geist, J., Schwerin, M., Rollinson, D., Tesch, M., and Choset, H., 2012. "Design and Architecture of the Unified Modular Snake Robot". In IEEE International Conference on Robotics and Automation (ICRA), pp. 4347–4354.

[4] Yanco, H., and Drury, J., 2004. ""Where am I?" Acquiring situation awareness using a remote robot platform". In IEEE International Conference on Systems, Man and Cybernetics, Vol. 3, pp. 2835–2840.

[5] Rollinson, D., Buchan, A., and Choset, H., 2012. "Virtual Chassis for Snake Robots: Definition and Applications". *Advanced Robotics*, Oct., pp. 1–22.

[6] Luo, R., Yih, C., and Su, K., 2002. "Multisensor fusion and integration: approaches, applications, and future research directions". *Sensors Journal, IEEE,* **2**(2), pp. 107–119.

[7] Kalman, R., 1960. "A new approach to linear filtering and prediction problems". *Journal of Basic Engineering,* **82**(1), pp. 35–45.

[8] Choset, H., Lynch, K. M., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L. E., and Thrun, S., 2005. *Principles of Robot Motion: Theory Algorithms, and Implementations*. MIT Press, Boston, USA.

[9] Julier, S., and Uhlmann, J., 2004. "Unscented Filtering and Nonlinear Estimation". *Proceedings of the IEEE,* **92**(3), Mar., pp. 401–422.

[10] Castrejon Lozano, J. G., Garca Carrillo, L. R., Dzul, A., and Lozano, R., 2008. "Spherical simplex sigma-point Kalman filters: A comparison in the inertial navigation of a terrestrial vehicle". *2008 American Control Conference*, June, pp. 3536–3541.

[11] Julier, S., 2003. "The spherical simplex unscented transformation". *Proceedings of the 2003 American Control Conference, 2003.*, pp. 2430–2434.

[12] Schick, I., and Mitter, S., 1994. "Robust Recursive Estimation in the Presence of Heavy-Tailed Observation Noise". *The Annals of Statistics,* **22**(2), pp. 1045–1080.

[13] Durovic, Z. M., and Kovacevic, B. D., 1999. "Robust estimation with unknown noise statistics". *Automatic Control, IEEE Transactions on,* **44**(6), June, pp. 1292–1296.

[14] Ting, J.-a., Theodorou, E., and Schaal, S., 2007. "Learning an Outlier-Robust Kalman Filter". In Machine Learning: ECML, University of Southern California, pp. 748–756.

[15] Thrun, S., Burgard, W., and Fox, D., 2005. *Probabilistic Robotics*. MIT Press.

[16] Zarchan, P., and Musoff, H., 2009. *Fundamentals of Kalman Filtering: A Practical Approach*, 3rd editio ed. AIAA (American Institute of Aeronautics & Astronautics).

[17] Kraft, E., 2003. "A quaternion-based unscented Kalman filter for orientation tracking". *Proceedings of the Sixth International Conference on Information Fusion,* **1**, pp. 47–54.

[18] Rollinson, D., Buchan, A., and Choset, H., 2011. "State Estimation for Snake Robots". In IEEE International Conference on Intelligent Robots and Systems (IROS), pp. 1075—-1080.

[19] Van Der Merwe, R., Wan, E., and Julier, S., 2004. "Sigma-Point Kalman Filters for Nonlinear Estimation and Sensor-Fusion". In Proceedings of the AIAA Guidance, Navigation & Control Conference, Citeseer, pp. 5120–5159.

[20] Press, W., Flannery, B., Teukolsky, S., and Vetterling, W., 1992. *Numerical Recipes: The Art of Scientific Computing*, 2nd ed. Cambridge University Press.

[21] Lefferts, E. J., Markley, F. L., and Shuster, M. D., 1982. "Kalman Filtering for Spacecraft Attitude Estimation". *Journal of Guidance, Control, and Dynamics,* **5**(5), Sept., pp. 417–429.

[22] Li, M., Kim, B. H., and Mourikis, A. I., 2013. "Real-time Motion Tracking on a Cellphone using Inertial Sensing and a Rolling-Shutter Camera". In IEEE International Conference on Robotics and Automation (ICRA), pp. 4697–4704.

[23] Li, M., and Mourikis, A. I., 2013. "3-D Motion Estimation and Online Temporal Calibration for Camera-IMU Systems". In IEEE International Conference on Robotics and Automation (ICRA), pp. 5689–5696.