The Balancing Cube

A DYNAMIC SCULPTURE AS TEST BED FOR DISTRIBUTED ESTIMATION AND CONTROL

SEBASTIAN TRIMPE and
RAFFAELLO D'ANDREA

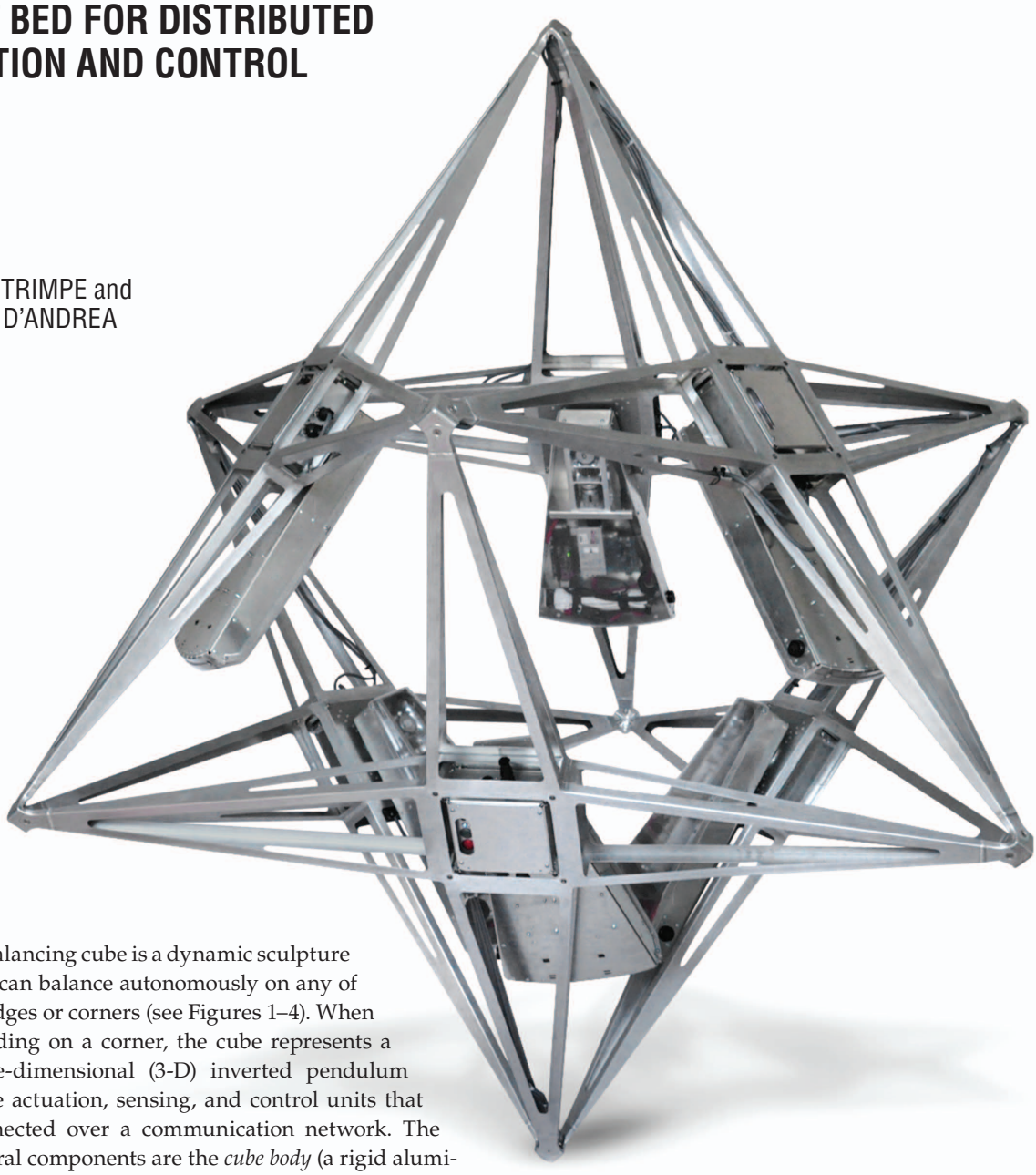


IMAGE COURTESY OF CAROLINA FLORES, ETH ZÜRICH

The balancing cube is a dynamic sculpture that can balance autonomously on any of its edges or corners (see Figures 1–4). When standing on a corner, the cube represents a three-dimensional (3-D) inverted pendulum with multiple actuation, sensing, and control units that are interconnected over a communication network. The main structural components are the *cube body* (a rigid aluminum structure with a cubic shape) and six identical rotating arms located on each of the cube's inner faces. The rotating arms are self-contained units carrying sensors, actuation, a computer, and a battery. Due to their modular design, these units are referred to as *modules*. As they rotate, they shift the overall center of mass of

Digital Object Identifier 10.1109/MCS.2012.2214135
Date of publication: 12 November 2012

1066-033X/12/$31.00©2012IEEE

the system, exert forces on the cube structure, and can, as a result, influence the cube's motion. The modules constitute the agents in the distributed and networked control system; their joint objective is the stabilization of the cube. A video of the cube can be found on the project Web site [1].

Inverted pendulum systems are popular in controls education and research; see, for example, [2]–[4] and references therein. In the most basic version, an inverted pendulum is a point mass on a massless link that is connected to a base (ground or moving platform) through a revolute joint with one degree of freedom (DOF, see Table 1 for acronyms). More generally, if the pendulum is considered as a general rigid body with three rotational degrees of freedom at its pivot, the pendulum is referred to as a 3-D pendulum [2]. A defining characteristic of all inverted pendulums is that the pendulum is pointing upward as seen from the pivot (that is, in opposite direction to the gravity vector). The corresponding equilibrium is hence unstable, which makes the system interesting for controls researchers and educators: to balance, the pendulum needs active stabilization by some actuation mechanism, such as actuated masses or a moving base.

Used as demonstrators for controls research since the 1950s (see [4] and references therein), inverted pendulum systems have remained popular experiments (simulation or physical) in various current research areas, such as learning control [5]–[7], networked control [8]–[12], adaptive control [13]–[15], model predictive control [8], [16], [17], decentralized control [18]–[20], and different branches of nonlinear control [21]–[28]. Common to most inverted pendulum systems is that the control algorithms are implemented on a single, central processing unit. In contrast, the balancing cube is stabilized by the joint action of six agents (the modules) with independent processing units, and the implementation of the feedback control system is distributed among the agents. The dynamics of the individual agents are coupled through the cube's rigid body.

Because data is exchanged between the agents over a digital communication network, the balancing cube qualifies as a *networked control system*, [29]. Other experimental test beds that have been developed to study distributed control and/or control of multiple agents over networks include modular robot systems [30], [31], a two-axis contouring system [32], a system for handling materials [33], a formation flight experiment [34], and multivehicle systems of various types [31], [35]–[37].

The cube is a 3-D inverted pendulum when balancing on one of its corners (denoted as *corner balancing*). When balancing on one of its edges (*edge balancing*) as shown in Figures 3 and 4, it becomes a 1-D inverted pendulum. Moreover, with the six modules on its inner faces, the cube is a multibody system and may therefore be qualified as a 1-D/3-D multibody inverted pendulum [2]. For both edge and corner balancing, different equilibrium configurations and, hence, different dynamics can be obtained by varying the nominal angle of the modules. Since the multibody system has fewer inputs than DOFs (each of the module DOFs is actuated, the cube body DOFs are not), it is an *underactuated mechanical system* [38]. Overall, the system combines the challenges of nonlinear unstable dynamics with distributed control and networked communication, making it a rich platform for research in dynamics and control.

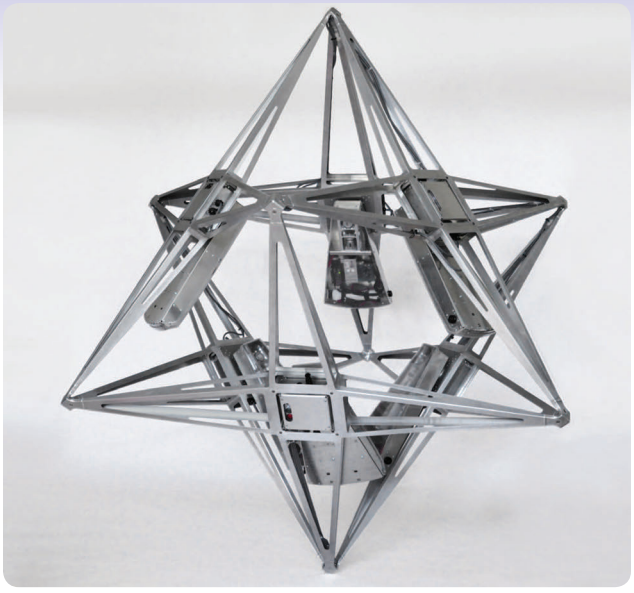


FIGURE 1 The cube balancing on one of its corners. The cube balances through the action of six rotating arms on the cube's inner faces. The diagram in Figure 2 is helpful for visualizing why the sculpture is called a cube: its tips are simply the corners of a cube. Alternative terms for this shape are the star tetrahedron and, less commonly used, the stellated octahedron. Self-contained with onboard sensing, actuation, computation, and communication, the rotating arms are called *modules* due to their modular design. The cube's height from tip to tip is 2.08 m.

Enabled by its control system, the balancing cube is a dynamic sculpture: the cube body is kept in balance through the slight corrective movements of the six modules. No external system is required for balancing. Set on one of its corners, the cube can balance as long as its batteries last (four or more hours)

TABLE 1 Acronyms used in this article.

Acronym	Meaning
CAN	Controller area network
CG	Center of gravity
dc	Direct current
DOF	Degree of freedom
IMC	Intermodule communication
IMU	Inertial measurement unit
LED	Light-emitting diode
LQR	Linear-quadratic regulator
MEMS	Microelectromechanical system
RMS	Root mean square
SBC	Single-board computer
SPI	Serial peripheral interface
WLAN	Wireless local area network

Enabled by its control system, the balancing cube is a dynamic sculpture: the cube body is kept in balance through the slight corrective movements of the six modules.

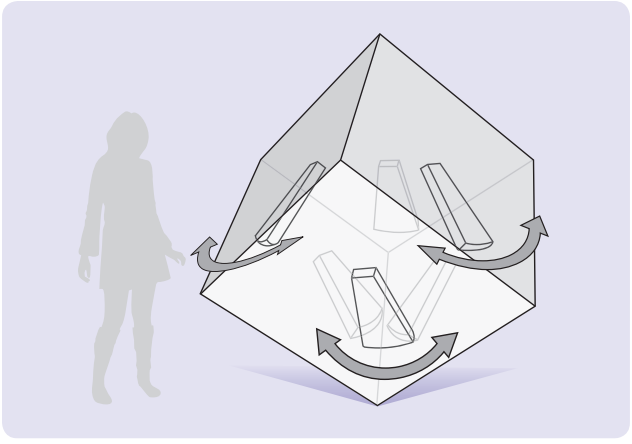


FIGURE 2 Corner balancing. The diagram visualizes the cubic shape and the six modules. It shows the cube in the same orientation as in Figure 1: the cube stands on a corner, and all modules are pointing down. Due to their position on the cube body, two types of modules are distinguished: the *bottom modules* and the *top modules*. The top modules are less effective for balancing the sculpture, as explained later in "Why Are the Top Modules Used Less?"

or until someone pushes it over. With many peoples' understanding of balancing, it makes an ideal device for communicating key concepts of control engineering such as stability, feedback control, and cooperation to the general public.

The balancing cube was built at the Institute for Dynamic Systems and Control (IDSC) at ETH Zurich and was completed in 2009. Since then, it has been demonstrated at public exhibitions and at an international control conference (see "Balancing Cube on Tour"). To the best of the authors' knowledge, the cube presented in this article is the only cube to date that can balance autonomously on a corner. Another cube, which can balance on a fixed edge, is sold by Quanser Inc. [39]. Quanser's cube uses a single actuation mechanism to stabilize the cube on its edge.

This article explains the design, modeling, and control of the balancing cube and demonstrates its balancing performance with experimental data. For the purpose of this article, all sensor data is exchanged between the agents. This way, the design of the control system can be posed as a centralized problem, which is addressed by separately designing a state estimator and a state-feedback controller. The resulting control and estimation algorithms are implemented in a distributed fashion; that is, they are running

Balancing Cube on Tour

Since its completion in fall 2009, the cube has been exhibited at public events such as the European researchers' night in Zurich, Switzerland, [S1], and the Festival della Scienza in Genoa, Italy, [S2]. More recently, it made an appearance at the triennial IFAC World Congress in Milan, Italy, [S3], as part of the interactive presentation of [40] (see Figure S1).

A large cube balancing on a corner in a public place is an unusual sight and tends to attract a good deal of interest. As passersby stop to push the cube and test its ability to maintain equilibrium, they engage in a unique opportunity to learn about control engineering and its limitations.

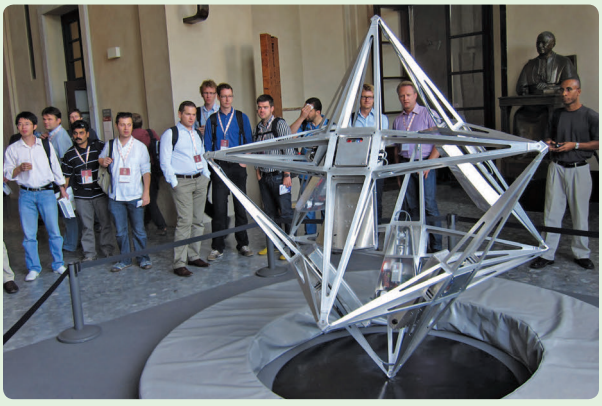


FIGURE S1 The balancing cube at the 2011 IFAC World Congress in Milan, Italy. The cube was shown in one of the interactive sessions.

REFERENCES

[S1] European Researchers' Night. (2009, Sep.). Zurich, Switzerland [Online]. Available: http://www.nachtderforschung.ethz.ch/en

[S2] Festival Della Scienza. (2009, Oct.). Genoa, Italy [Online]. Available: http://www.festivalscienza.eu

[S3] 18th World Congress International Federation of Automatic Control. (2011, Aug.). Milan, Italy [Online]. Available: http://www.ifac2011.org

on all six agents in parallel with no hierarchical distinction among the agents. The full communication case presented herein serves as the baseline for studies with constrained or reduced communication. In [40] and [41], two approaches are presented for the problem of state estimation and stabilization of the cube with reduced communication.

Whereas the state-feedback controller is designed using standard linear-quadratic regulator (LQR) design techniques, the state estimator is tailored to the specific problem. It exploits the facts that the cube has only rotational DOFs, and that measurements from multiple inertial sensors are available, to generate an estimate of the cube's tilt that is independent of the rigid body dynamics. In particular, the estimator provides a tilt estimate for whatever motion of the cube (slow or fast), and no assumption on near equilibrium configuration is made. The estimation algorithm only requires geometric system knowledge, namely the sensor locations on the cube. Since the algorithm does not rely on a dynamic system model, it is inherently robust to modeling errors or changes in the system (for example, in the mass or module configuration). The developed tilt estimation algorithm is applicable to any rigid body with only rotational degrees of freedom that is equipped with multiple inertial sensors.

This article focuses on the concepts and tools that were used to build and control the cube. The models that are presented herein include sensor models capturing the nonlinear dependency of the measurements on the system states (used for the state estimator design) and a linear model of the system dynamics (used for the design of the linear state-feedback controller). Nonlinear dynamic models and nonlinear controllers for similar 3-D pendulum systems (but with different actuation mechanisms) can be found in [21]–[24], for example.

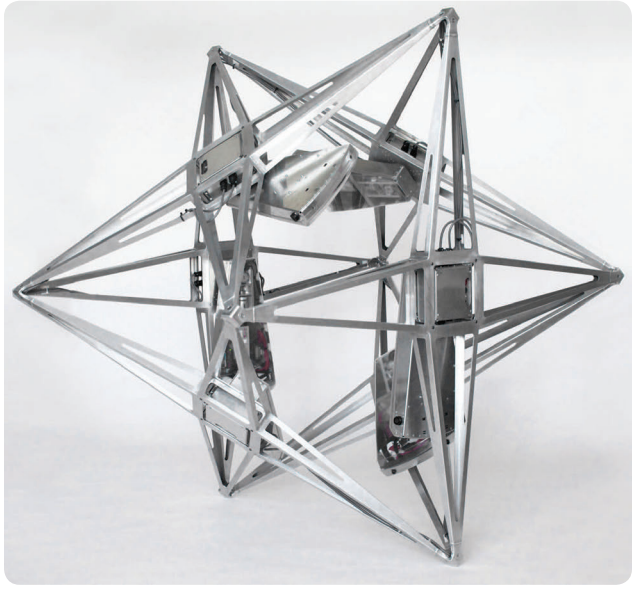


FIGURE 3 The cube balancing on one of its edges. When the cube body is placed on two of its tips, it has only one rotational degree of freedom left. This is called edge balancing (see Figure 4).

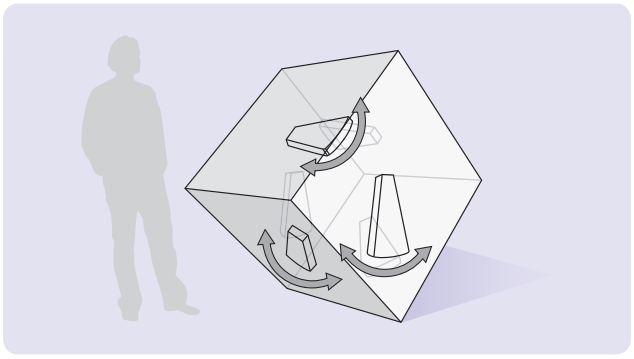


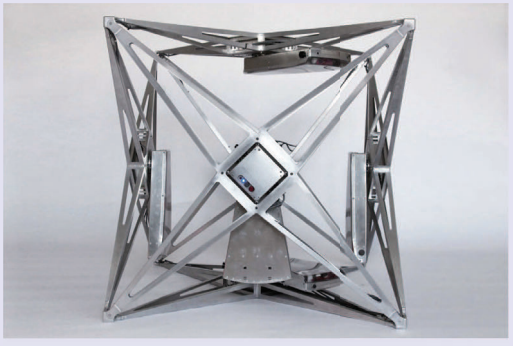
FIGURE 4 Edge balancing. The cube's orientation is the same as in Figure 3: the cube stands on an edge, the modules on the front and back face are pointing down, and the bottom and top modules are rotated away from the downward position.

DESIGN

In this section, the hardware design of the balancing cube's two key components is discussed: the cube body and the modules.

Cube Body

The cube body is formed by six sheet metal constructions (one for each face), and eight corner parts (to connect the



(a)

(b)

FIGURE 5 Detailed views of the cube's rigid body. The cube body has six faces and eight corners. The faces are formed by an X-shaped welded aluminum construction (a). Three adjacent faces join at right angles at one of the eight identical corner parts (b).

faces); see Figure 5. Each of the cube's faces is an X-shaped welded construction fabricated from 1.5-mm sheet metal. The particular shape provides support for the modules, which are mounted in a square slot in the center of each face. The cube's corners are CNC-machined aluminum parts. On each corner part, the three adjacent faces are attached at right angles. The result is a single rigid body in the shape of a cube with an edge length of 1.2 m. The modular construction of the cube body allows for easy disassembly and transportation.

The total mass of the cube body (without the modules) is 14 kg and represents a trade-off between weight and structural integrity. On the one hand, the structure must be light enough for the modules to manipulate the cube's overall center of gravity (CG). On the other hand, the structure must be strong enough to support the modules and to withstand repeated falls. The three principal moments of inertia of the cube body are roughly 5 kgm^2 (each of the principal axes goes through the center of two opposing cube faces).

Modules

The six modules carry the system's mechanical and electrical system components. At the same time, they constitute the actuation mechanisms that allow the cube body to balance. One of the cube's modules is shown in Figure 6. Each module is composed of two parts: 1) a square-shaped component that is fixed to the cube's rigid body and 2) a pie-shaped, eccentrically mounted "arm" that rotates relative to the cube body. Because of its motion relative to the rigid body, the former is called the *nonmoving part* and the latter the *moving part* of the module.

The cube is actuated by the rotating arms through: 1) gravitational moments caused by their displacement and 2) reaction moments caused by their acceleration or deceleration. Other actuation mechanisms are conceivable: in [2] and [21] for example, reaction wheels, proof masses, and fans are discussed as actuators to control a 3-D pendulum.

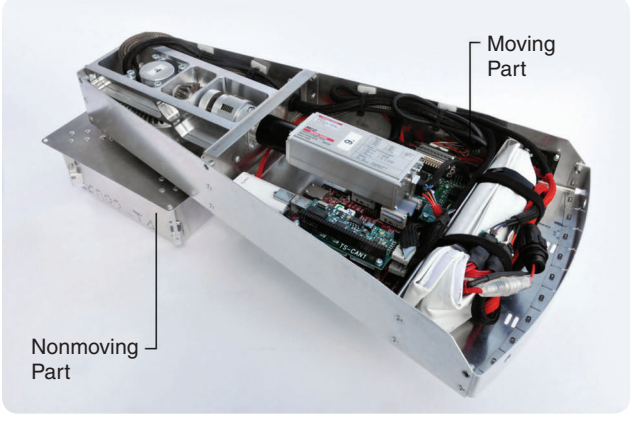


FIGURE 6 A module. Each module consists of a nonmoving part, which is rigidly mounted to the cube body, and a moving part (the pie-shaped arm), which rotates relative to the nonmoving part.

Rotating arms were chosen for the balancing cube mostly for aesthetic reasons.

Nonmoving Part

The nonmoving part connects the module's moving part to the cube body. It also houses inertial sensors and a user interface; see Figure 7. The nonmoving part is rigidly mounted to the center of the cube's face so that the user interface faces outward. The nonmoving part has a mass of about 1.2 kg. The six nonmoving parts, together with the cube body, constitute one rigid body that must be balanced through the action of the moving parts.

The nonmoving part houses an inertial measurement unit (IMU) (Analog Devices, ADIS16350) with triaxis accelerometer and triaxis rate gyroscope. A lowpass filter onboard the IMU results in accelerometer and gyro noise standard deviations of $\sigma_{\text{acc}} = 0.04$ m/s^2 and $\sigma_{\text{gyro}} = 0.0042$ rad/s, respectively. From the IMU measurements, the tilt of the rigid body and its rate of change are estimated as described in the section "State Estimation."

The user interface consists of two LEDs and two pushbuttons. Located on each face of the cube, the user interfaces are used to set the state of the system (for example, calibration mode or balance mode) and to turn the system on and off.

Connectors on two sides of the nonmoving part (see Figure 7) connect all modules through wires running along the

FIGURE 7 The module's nonmoving part. The user interface has two buttons and two LEDs indicating the status of the system. Cables running along the cube structure (not shown) connect the nonmoving parts and allow both the exchange of data and the synchronization of the power circuits. The inertial measurement unit observing the cube motion sits inside the nonmoving part (not shown).

cube structure. One set of wires connects the modules' power circuits so that they can all be turned on/off at once by pressing a button on any module. A second set of wires forms the data network, allowing communication between the modules.

Moving Part

The motion of the cube body is influenced by actuating the moving part of the modules. In addition to the actuation mechanism, the moving part carries an absolute encoder, a computer, and a battery. All components are shown in Figure 8. The moving part has a total mass of roughly 3.7 kg, however additional weights of up to 1.9 kg can be added to increase control authority. For the results presented in this article, the moving parts of the bottom modules (see Figure 2) are equipped with an extra mass of 1.9 kg each.

The frame of the moving part is made of sheet metal. A removable cover of semitransparent plastic protects the system components from dirt. Machined aluminum parts attach to the sheet metal frame and hold the actuation mechanism, which consists of a 60-W brushless dc motor with a planetary gear head (reduction 1:103) to drive the pinion of a bevel gear (reduction 1:3). The bevel gear is connected to the nonmoving part of the module and hence the cube body. The motor therefore rotates the moving part relative to the cube. A clutch is used in the drive train to protect the motor from mechanical damage (such as when the cube falls) and to protect users if they are unintentionally hit by a rotating module.

The dc motor and the gear head are part of a compact drive unit (Maxon, MCD EPOS 60 W) that also includes a motor shaft encoder and a digital position and velocity control unit. In normal operation, the control unit is used in velocity mode to control the motor shaft velocity and, hence the angular velocity of the module.

The motor receives commands from a single-board computer (SBC) over a dedicated controller area network (CAN). Through the same interface, the SBC can also read out motor data, such as the shaft velocity or the motor current. The SBC (embeddedARM, TS-7260) has a 200-MHz ARM9 processor and consumes fewer than 1 W of power. In addition to the motor, it interfaces with all local sensors, the local user interface, and all other SBCs.

An absolute encoder (Hengstler, AC 36) with 12-bit resolution is attached to the module's axis of rotation and hence allows for absolute positioning of the module's moving part relative to its nonmoving part. It is connected to the SBC's serial peripheral interface (SPI).

The wires connecting the components on the nonmoving part to the SBC are routed through a slip ring (Moog, AC6846). The IMU is connected to the SBC over SPI. The SBCs are connected with each other over a CAN separate from the CAN connecting the motor. The CAN used for the *intermodule communication* (IMC) is a broadcast network operating at a data rate of 500 kB/s. The network allows the reliable exchange of all absolute encoder and IMU measurements between all modules every 10 ms.

The SBC handles the IMC and runs the estimation and control algorithms that enable the cube to balance. For programming and monitoring purposes, each SBC also has a wireless local area network (WLAN) module that allows for a connection to an external computer. A Linux operating system on the SBC allows easy handling of data and external access.

All components on the module are powered by a six-cell lithium polymer battery (FlightPower, EVO 20 3700 mAh 22.2 V). Customized electronics manage each module's power locally, including safety shut-off. Each module's battery voltage is additionally monitored by the

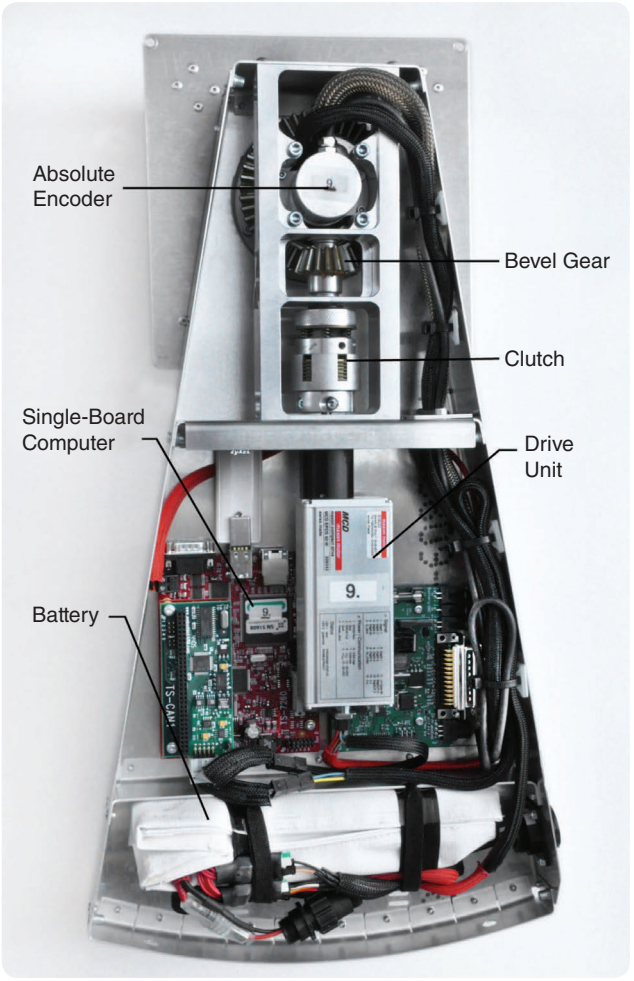


FIGURE 8 System components on the moving part of a module. The torque produced by the dc motor of the drive unit is applied between moving part and nonmoving part through the bevel gear. The clutch in the drive train protects the motor from mechanical damage. The absolute encoder measures the angle of the moving part relative to its mounting. All sensors are read by the single-board computer, which issues commands to the drive unit. The electronic connection to the components on the nonmoving part is through a slip ring (not shown). The module is powered by the lithium polymer battery.

(a)

(b)

(c)

FIGURE 9 Operation of the balancing cube. This photo sequence illustrates a typical operation sequence: (a) lifting the cube, (b) holding near equilibrium, and finally (c) balancing autonomously.

SBC through its analog input. When fully charged, the cube can balance on one of its corners for more than four hours. The batteries can be charged through a connector on one side of the moving part.

OPERATION

The usual operation of the system is illustrated in Figure 9. First, a human operator lifts the cube and brings it near the desired equilibrium. The cube recognizes its balancing mode, that is, which edge or corner it is standing on, and rotates its modules to the appropriate starting configuration. After a short calibration phase, the cube starts to balance autonomously. If the cube falls, for example, after it was pushed too hard, the operation cycle—lifting the cube, holding near equilibrium, balancing until disturbed—may be repeated.

The balancing cube is a standalone device (without external systems) that can operate anywhere there is solid ground. To lower the impact on the structure when the cube falls (such as when the batteries run low or a viewer pushes the structure too hard), the cube is usually balanced inside a foam ring. The complete setup is shown in Figure 9.

In normal operation mode, the operator controls the cube through the dedicated user interfaces on the cube's faces or by guiding or pushing the cube body (such as during the setup phase). The system uses its inertial sensors to respond. An external computer can be used for monitoring real-time data sent over WLAN.

MODELING

This section presents the models that are used in later sections for the design of the state estimation and control algorithms.

Linear dynamic models have proven sufficient for designing controllers that can stabilize the cube about an equilibrium. The system's nonlinear equations of motion are therefore omitted. Nonlinear dynamic models for similar 3-D pendulum systems (with more straightforward actuation mechanisms than the rotating arms on the cube) are discussed in [2] and [21]–[24]. The multibody system of the cube is complicated enough so that it is arguable how to best represent it in terms of being comprehensible, manageable, and not error-prone—as a system of nonlinear differential equations or as a computer-based symbolic model, for example. Herein, a 3-D multibody model in Matlab/Simulink is presented, which is used for nonlinear simulations and to extract linearized dynamic models. For a 1-D abstraction of the cube (an inverted pendulum being balanced by a single module, presented later in "Why Are the Top Modules Used Less?"), the modeling procedure was verified by comparing the computer-based models to the analytically derived equations of motion.

The design of the global and nonlinear state estimator does not rely on the linear dynamics model; in fact, it does not rely on any dynamics model. The sensor models that are the basis of the state-estimator design are algebraic equations expressing the sensor measurements as a (nonlinear) function of the system states. To state these sensor models, only geometric information about the system (namely the sensor locations on the cube body) is required.

Multibody System

The balancing cube is a multibody system, with the cube body and the modules as rigid bodies. The cube body and

The balancing cube is a standalone device that can operate almost anywhere there is solid ground.

the nonmoving parts of the modules are treated as a single rigid body, which stands with either one or two of its tips in contact with the ground. The modules' moving parts are connected to the cube body through revolute joints. Because of its large mass, it can be assumed that the supporting points of the cube body do not slip and that the cube, therefore, does not experience any translational motion. The support of the rigid body is hence modeled as a ball joint with three rotational DOFs for the case of corner balancing. When the cube balances on its edge, the second ground contact constrains two rotational DOFs. Hence the cube support is modeled as a revolute joint with one rotational DOF.

The cube's body is subject to gravity and moments generated by the actuation mechanisms (the rotating arms). The latter include gravitational moments due to displacement of the eccentric moving parts and reaction moments from accelerating or decelerating the arms. Centripetal forces from the rotation of the arms are negligible because of low angular rates in typical operation.

The coordinate frames shown in Figure 10 are used to describe the orientation of the cube: $\hat{O}$ denotes the inertial frame of reference, and $\hat{B}$ denotes the cube body-fixed coordinate frame. The origin of $\hat{B}$ lies on the cube's balancing corner and its axes are along the cube's edges as depicted in Figure 10. The origins of frames $\hat{O}$ and $\hat{B}$ coincide.

The rotation between the inertial frame $\hat{O}$ and the cube frame $\hat{B}$ is expressed by the rotation matrix ${}^{O}_{B}S$. For all rotation matrices, the notation from [42] is adopted, where the matrix ${}^{O}_{B}S$ describes the rotation of $\hat{B}$ relative to $\hat{O}$, and a vector quantity v given in frame $\hat{B}$, ${}^{B}v \in \mathbb{R}^3$, is expressed in frame $\hat{O}$ by ${}^{O}v = {}^{O}_{B}S\, v^{B}$.

In this article, the attitude of the rigid body is represented by Z-Y-X-Euler angles (yaw, pitch, roll) such that (see [42])

$$ {}^{O}_{B}S = S_Z(\alpha)\, S_Y(\beta)\, S_X(\gamma), \tag{1} $$

with

$$ S_Z(\alpha) := \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad S_Y(\beta) := \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix}, $$

$$ S_X(\gamma) := \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{bmatrix}, $$

where α, β, and γ are the yaw, pitch, and roll Euler angles, respectively. Notice that the yaw angle α (capturing rotations about the gravity axis) is irrelevant for balancing since gravity does not excite any yaw motion. Therefore, the yaw angle is not part of the system state that is considered later for the controller and estimator design. Because yaw represents a DOF of the rigid body, it is introduced here nonetheless. The particular choice of the Euler angle order (Z-Y-X) will result in the yaw angle naturally dropping out in the derivation of the state estimator later. The pair of pitch and roll angle (β, γ) is denoted the *tilt* of the rigid body. When the cube balances on its edge, the body Y-axis is the axis of rotation; that is, the rotation angle is β, and $\gamma = 0$.

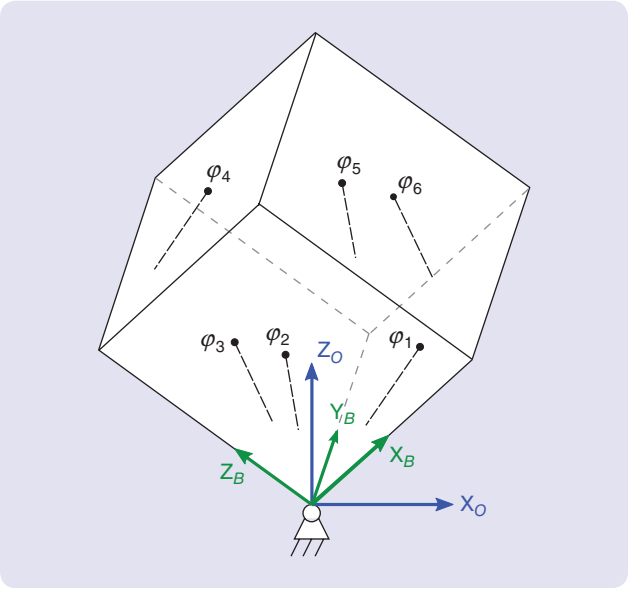


FIGURE 10 Coordinate frame definitions and zero module angles. Frame $\hat{O}$ (in blue with axes X_O–Y_O–Z_O, Y_O axis not shown and pointing inside the drawing plane) is the inertial frame of reference, and $\hat{B}$ (green, X_B–Y_B–Z_B) is the body-fixed frame. The cube is shown standing on its corner, where the corner coincides with the inertial frame origin. For edge balancing, the body frame Y-axis Y_B is the balancing axis. The zero directions of the module angles $\varphi_1, \varphi_2, \ldots, \varphi_6$ are indicated as the dashed black lines (pointing downward, toward the balancing corner), and the direction of positive rotation is such that the rotation vector points to the cube's center.

The modules' rotation vectors are orthogonal to the corresponding cube face and point to the cube's center. The angle φ_i is used to denote the rotation angle of module i relative to the cube body. The zero angles of φ_i are shown in Figure 10. The full configuration of the multibody system is described by the generalized coordinates

$$ q = (\varphi_1, \varphi_2, \varphi_3, \varphi_4, \varphi_5, \varphi_6, \alpha, \beta, \gamma). \tag{2} $$

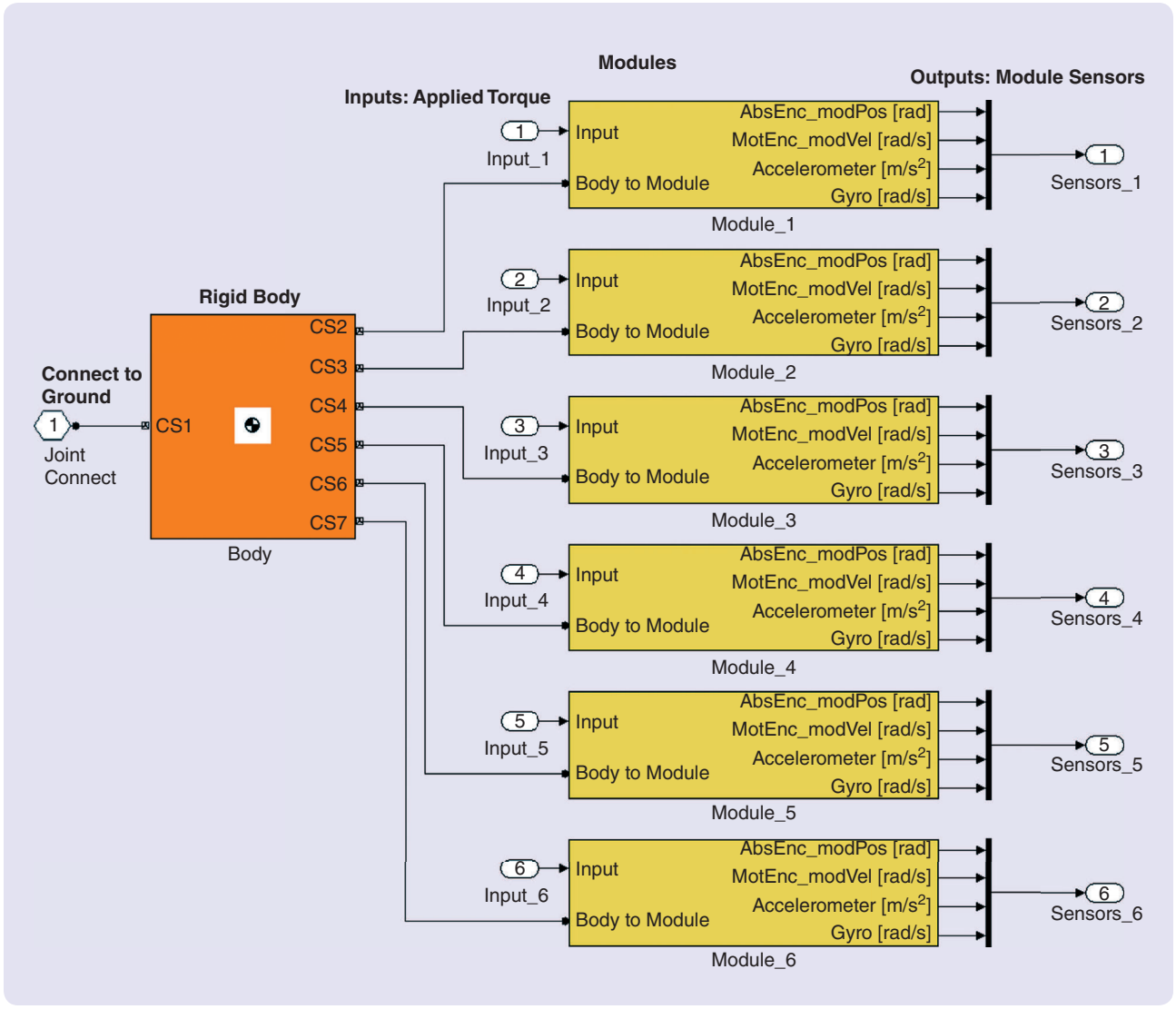


FIGURE 11 SimMechanics model. Shown is a screenshot of the highest model level with the cube body block (orange) and the six modules (yellow). Inputs to the modules are the motor torques, and the outputs are the sensor measurements. The cube's ground connection is either to a ball joint (for corner balancing) or a revolute joint (for edge balancing).

The acceleration that is sensed by an IMU depends on the IMU mounting position and orientation on the rigid body. Because the IMU is a component of the nonmoving part of the module, its position and orientation are constant in the body frame $\hat{B}$. The positions of the six IMUs on the cube body are denoted by ${}^{B}p_i$. Each sensor measures accelerations and rotations in its local sensor frame $\hat{A}_i$. The IMU orientation is captured by the rotation matrix ${}^{A_i}_{B}S$. Since the mounting of the IMUs is known, ${}^{B}p_i$ and ${}^{A_i}_{B}S$ are known for all sensors.

3-D Simulation Model

A multibody model capturing the nonlinear system dynamics in 3-D was built in SimMechanics, which is an extension of Matlab/Simulink for physics-oriented symbolic modeling of mechanical systems, [43]. A library was created that includes models of all system components, in particular, the cube body and the modules with their actuation mechanism and sensors. The library facilitates the assembly of models for different simulation scenarios (for example, corner or edge balancing) and allows for easy modification of the complete system or its subcomponents. A screenshot of a part of the model is shown in Figure 11.

The SimMechanics model provides a convenient simulation platform for design and verification of the control and estimation algorithms. Furthermore, it is used to automatically generate linearized models about different equilibrium configurations, as described below. An earlier version of this model was used for feasibility studies and to support design decisions early in the project's development.

Linear Dynamics Model

Different static equilibrium configurations can be obtained for both edge and corner balancing by varying the nominal angles of the modules. Herein, the two basic configurations in Figures 1 and 2 (for corner balancing) and in Figures 3 and 4 (for edge balancing) are considered. The modeling procedure does not, however, depend on the specific equilibrium and can readily be applied in the same way for other equilibria. Due to the particular mass configuration of the cube body and the modules, the range of equilibria is limited. The maximum that the cube can tilt and still remain in equilibrium is discussed in "What Is the Cube's Maximal Balancing Range?"

Expressed in generalized coordinates (2), the considered equilibrium for corner balancing is

$$q_0^c = \left(0, 0, 0, 0, 0, 0, \alpha, -\operatorname{atan}(1/\sqrt{2}), \frac{\pi}{4}\right). \tag{3}$$

It corresponds to the cube standing upright (its body diagonal being parallel to the gravity axis) and all modules pointing downward as shown in Figures 1 and 2. The yaw angle α is left unspecified and can be arbitrary. The edge balancing equilibrium configuration shown in Figures 3 and 4 is given by

$$q_0^e = \left(0, 0, \frac{\pi}{2}, -\frac{\pi}{4}, 0, -\frac{3\pi}{4}, \alpha, -\frac{\pi}{4}, 0\right). \tag{4}$$

What Is the Cube's Maximal Balancing Range?

This study investigates how much the cube body can tilt in static equilibrium by changing the module angles, as compared to the upright configurations (3) and (4). The maximal tilt calculated below is a theoretical upper bound on the feasible balancing range (for greater tilt, the multibody system has no static equilibrium for any configuration of the modules). The calculations are based on the SimMechanics multibody model.

To illustrate the dependency of the cube tilt on the module angles in static equilibrium for corner balancing, the equilibrium tilt is computed for the module angles parameterized by

$$(\varphi_1, \varphi_2, \varphi_3, \varphi_4, \varphi_5, \varphi_6) = (\phi, -\phi, 0, -\phi, \phi, 0), \tag{S1}$$

with parameter ϕ ranging from 0° to 180°. For $\phi = 0°$, the module configuration (S1) is the same as for the nominal "upright" equilibrium (3). The chosen parameterization affects the equilibrium only in pitch direction; the roll equilibrium angle is identical to the nominal γ_0 in (3) for all ϕ. The difference of the resulting equilibrium pitch (denoted by $\bar{\beta}$) from the nominal pitch β_0 in (3) is shown in Figure S2. Additionally, the resulting horizontal displacement d_{xy} of the cube tip is shown in Figure S2; it is given by

$$\begin{bmatrix} d_x \\ d_y \\ d_z \end{bmatrix} := \left({}^O_B S(\beta_0, \gamma_0) - {}^O_B S(\bar{\beta}, \bar{\gamma})\right) \begin{bmatrix} 1.2\,\text{m} \\ 1.2\,\text{m} \\ 1.2\,\text{m} \end{bmatrix}, \tag{S2}$$

$$d_{xy} := \sqrt{d_x^2 + d_y^2} \tag{S3}$$

where 1.2 m is the edge length of the cube. The maximal pitch angle difference $\bar{\beta} - \beta_0 = -6.74°$ corresponds to the maximal tip displacement of 24.4 cm; it is attained for $\phi = 102.5°$. As verified by an optimization over the module angles without the constraint (S1), the maximal displacement is, in fact, the global maximum for all possible module angles. The module configuration (S1) with $\phi = 102.5°$ is, however, not the only configuration that maximizes the cube tip displacement.

A similar analysis for edge balancing yields a maximal tilt from the upright standing cube [β_0 in (4)] of 8.15°, which corresponds to a tip displacement of $d_{xy} = 24.1$ cm.

The obtained maximal tip displacements represent the maximal range of static equilibria. In practice, the maximal range of equilibria that can be stabilized is smaller due to actuation limitations and sensor noise.

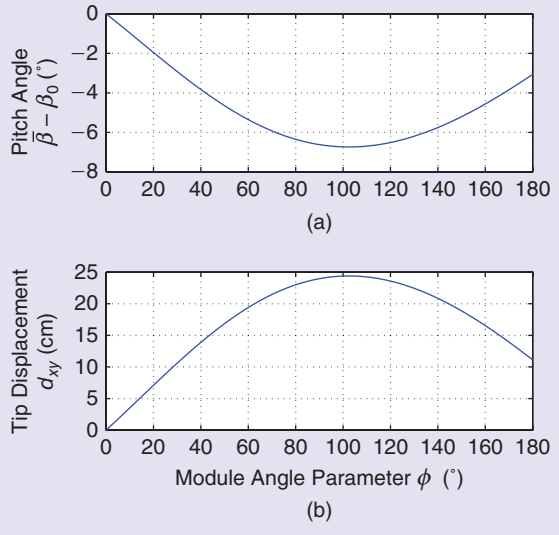


FIGURE S2 Cube tilt and tip displacement for different corner balancing equilibria. The pitch angle $\bar{\beta}$ in static equilibrium is shown in (a) as a function of the module angle parameter ϕ [the module angles are parametrized according to (S1)]. For better comparability, the nominal pitch β_0 is subtracted; that is, an angle of 0° corresponds to the nominal equilibrium (3), where the cube is standing upright. For the chosen parameterizations, the roll angle is equal to the nominal ($\bar{\gamma} = \gamma_0$) and therefore omitted here. The corresponding horizontal displacement d_{xy} of the cube tip from the upright configuration is shown in (b).

TABLE 2 The states of the cube model (5). Roll angle and roll rate, x_{15} and x_{16}, are relevant only for corner balancing.

State Variable	Physical Meaning
x_1	φ_1, angle module 1
x_2	φ_2, angle module 2
⋮	⋮
x_6	φ_6, angle module 6
x_7	$\dot{\varphi}_1$, angular velocity module 1
x_8	$\dot{\varphi}_2$, angular velocity module 2
⋮	⋮
x_{12}	$\dot{\varphi}_6$, angular velocity module 6
x_{13}	β, cube pitch angle
x_{14}	$\dot{\beta}$, cube pitch rate
x_{15}	γ, cube roll angle (corner balancing only)
x_{16}	$\dot{\gamma}$, cube roll rate (corner balancing only)

TABLE 3 The inputs of the cube model (5).

Input Variable	Physical Meaning
u_1	Torque at module 1
u_2	Torque at module 2
⋮	⋮
u_6	Torque at module 6

The dynamics of the multibody system about an equilibrium configuration (3) or (4) are described by the state-space model

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{5}$$

with the system inputs $u(t) \in \mathbb{R}^6$ being torques applied to the modules, and $x(t) \in \mathbb{R}^n$ the state vector. The system states are the generalized coordinates of the multibody system and their time derivatives; that is, the angles and angular velocities of the modules and the cube body. When the cube stands on an edge, it has one rotational DOF (the pitch angle β); when it is balancing on a corner, it has three rotational DOFs. The rotation about the gravitational axis can, however, be neglected for balancing and is removed from the state-space equations. The relevant state of the cube body is therefore fully characterized by the pitch and roll angles, β and γ, and their derivatives. Hence, $n = 14$ for edge balancing, and $n = 16$ for corner balancing. The physical meaning of states and inputs is summarized in Tables 2 and 3.

Built-in functions in Matlab/Simulink are used to compute numerical values for the state space model (A, B) from the SimMechanics model. Since for the controller design procedure it makes no difference whether edge or corner balancing is considered, the general state-space description (A, B) is used throughout the article. The open-loop poles of the system are listed in Table 4. The numerical values for the state space matrices (A, B) can be found in [44].

TABLE 4 Open-loop poles. The linear cube model (5) has 16 poles for corner balancing and 14 poles for edge balancing. The poles were computed numerically from the SimMechanics models. The two unstable poles for corner balancing correspond to the two dimensions of the cube that need to be stabilized. The pole roughly at zero for edge balancing is due to the fact that two modules are horizontal (the top modules in Figure 4). It corresponds to an eigenmode with equally directed displacement of these two modules.

Corner Balancing	Edge Balancing
2.92	3.09
2.92	0.0
−2.92	−3.11
−2.92	−1.04
−0.505 ± 3.8i	−0.294 ± 3.09i
−0.505 ± 3.8i	−0.294 ± 3.14i
−0.54 ± 3.97i	−0.516 ± 4.15i
−0.362 ± 4i	−0.315 ± 4.53i
−0.341 ± 4.14i	−0.559 ± 0.207i
−0.341 ± 4.14i	–

Nonlinear Sensor Model

Functional relations between measurements y and the system states x, which form the basis for developing the state estimation algorithms, are sought in this section. The obtained sensor model is linear for the absolute encoder and nonlinear for the accelerometer and rate gyroscope.

Accelerometer Model

The previous definitions of the inertial frame $\hat{O}$, the body frame $\hat{B}$, and the IMU frame $\hat{A}_i$ are used to express the acceleration ${}^{A_i}y_i^{\text{acc}}$ measured by the triaxis accelerometer on module i. The accelerometer located at the position ${}^{O}p_i$ measures the acceleration ${}^{O}\ddot{p}_i$ of the cube body at this position plus the gravity vector ${}^{O}g$ plus sensor noise, all in its local frame $\hat{A}_i$. That is,

$${}^{A_i}y_i^{\text{acc}} = {}^{A_i}_{B}S\,{}^{B}_{O}S\,({}^{O}\ddot{p}_i + {}^{O}g) + {}^{A_i}w_i^{\text{acc}}, \tag{6}$$

where ${}^{A_i}y_i^{\text{acc}}$ is module i's accelerometer measurement (in m/s^2), and ${}^{A_i}w_i^{\text{acc}}$ is measurement noise. The noise is assumed to be zero-mean, band-limited white noise with standard deviation σ_{acc}; that is, $\mathbb{E}[{}^{A_i}w_i^{\text{acc}}] = 0$, $\mathbb{E}[{}^{A_i}w_i^{\text{acc}}({}^{A_i}w_i^{\text{acc}})^T] = \sigma_{\text{acc}}^2 I$, where $\mathbb{E}[\,\cdot\,]$ denotes the expected value and I denotes the identity matrix of appropriate dimensions. This noise model is reasonable for many MEMS accelerometers once the bias has been removed and if scaling and axes cross coupling errors are neglected [45]. Biases in the state estimates resulting from sensor biases

are compensated by using integral action in the control algorithm. This is explained later in the section "Control."

From the identity ${}^{O}p_i = {}^{O}_{B}S\,{}^{B}p_i$ and the fact that ${}^{B}p_i$ is constant with time, it follows that

$$ {}^{O}\ddot{p}_i = {}^{O}_{B}\ddot{S}\,{}^{B}p_i, \tag{7} $$

where ${}^{O}_{B}\ddot{S}$ denotes the second derivative of the rotation matrix ${}^{O}_{B}S$ with respect to time. The matrix ${}^{O}_{B}\ddot{S}$ captures the rotational and centripetal acceleration terms of the cube rigid body motion. Using (7) and multiplying (6) with ${}^{B}_{A_i}S$ from the left yields

$$ {}^{B}y_i^{\text{acc}} = \tilde{S}\,{}^{B}p_i + {}^{B}g + {}^{B}w_i^{\text{acc}}, \tag{8} $$

where

$$ \tilde{S} := {}^{B}_{O}S\,{}^{O}_{B}\ddot{S} \tag{9} $$

combines the rotation and acceleration of the rigid body,

$$ {}^{B}g = {}^{B}_{O}S\,{}^{O}g \tag{10} $$

is the gravity vector in body coordinates, and

$$ {}^{B}w_i^{\text{acc}} = {}^{B}_{A_i}S\,{}^{A_i}w_i^{\text{acc}} \tag{11} $$

is the noise vector rotated to the body frame. The mean and variance of the noise still satisfy $\mathbb{E}[{}^{B}w_i^{\text{acc}}] = 0$ and $\mathbb{E}[{}^{B}w_i^{\text{acc}}({}^{B}w_i^{\text{acc}})^T] = \sigma_{\text{acc}}^2 I$.

Equation (8) expresses an accelerometer measurement as a function of the rigid body dynamics (captured in $\tilde{S}$), the gravity vector in the body frame ${}^{B}g$, and sensor noise. This relation is used in the section "State Estimation" to obtain an estimate of ${}^{B}g$ from multiple accelerometer measurements. By means of (10) and the representation (1) of ${}^{B}_{O}S$, an estimate of the cube tilt is then obtained.

Rate Gyroscope Model

The six rate gyros measure the angular rate vector of the cube body: expressed in the body frame of reference $\hat{B}$,

$$ {}^{B}y_i^{\text{gyro}} = {}^{B}\omega + {}^{B}w_i^{\text{gyro}}, \tag{12} $$

where ${}^{B}y_i^{\text{gyro}}$ is the ith rate gyro measurement (in rad/s) rotated to the body frame, ${}^{B}\omega$ is the angular rate vector in the body frame, and ${}^{B}w_i^{\text{gyro}}$ is sensor noise with zero mean and variance $\mathbb{E}[{}^{B}w_i^{\text{gyro}}({}^{B}w_i^{\text{gyro}})^T] = \sigma_{\text{gyro}}^2 I$.

The body rotation vector relates to the Euler angle rates $\dot{\alpha}, \dot{\beta}, \dot{\gamma}$ by (see [46], for example)

$$ \begin{bmatrix} \dot{\alpha} \\ \dot{\beta} \\ \dot{\gamma} \end{bmatrix} = T(\beta, \gamma)\,{}^{B}\omega, \tag{13} $$

with the nonlinear transformation

$$ T(\beta,\gamma) := \begin{bmatrix} 0 & \sin(\gamma)/\cos(\beta) & \cos(\gamma)/\cos(\beta) \\ 0 & \cos(\gamma) & -\sin(\gamma) \\ 1 & \sin(\gamma)\tan(\beta) & \cos(\gamma)\tan(\beta) \end{bmatrix}. \tag{14} $$

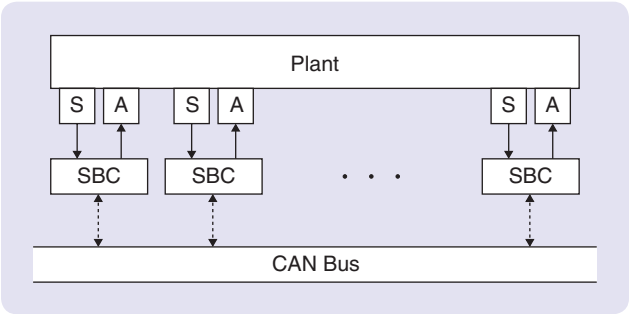


FIGURE 12 Abstraction of the networked control system. The blocks A and S denote actuator and sensor units. The single-board computer (SBC) runs estimation and control algorithms and manages the communication with the other modules over the controller area network (CAN) bus.

The transformation matrix (14) is nonsingular for the considered equilibria (3) and (4). Equations (12) and (13) are used later to obtain estimates of the pitch rate $\dot{\beta}$ and the roll rate $\dot{\gamma}$.

Absolute Encoder Model

The sensor model of the absolute encoder is straightforward: each absolute encoder measures the corresponding module angle. Hence, the measurement y_i^{enc} (in rad) of module i's absolute encoder is given by

$$ y_i^{\text{enc}} = \varphi_i + w_i^{\text{enc}}, \tag{15} $$

where w_i^{enc} is a random variable modeling the quantization error due to finite encoder resolution.

CONTROL SYSTEM ARCHITECTURE

The SBCs on the modules run the estimation and control algorithms, which enable the cube to balance. Based on its local sensor data and data communicated from the other modules over CAN, each module's SBC computes commands for its local actuator. The components of the networked control system are depicted in Figure 12.

The algorithms implemented on each module comprise a state estimator and a state-feedback controller. Each module maintains an estimate $\hat{x}$ of the full system state x, which it computes from all available sensor data (local and IMC data). The controller uses this estimate to compute the actuator command. The block diagram in Figure 13 represents the implementation of the feedback-control system.

A crucial question for the design of the estimation and control algorithms is what data is shared between the modules over CAN. Since CAN is a broadcast network, if one module sends data, the data can be received by all the others. For the results presented in this article, all modules share all their local sensor data (IMU and absolute encoder) over CAN. The capacity of the network is such that all modules can broadcast their sensor measurements every 10 ms, which is the time step of the feedback controllers K_i. Delays and losses in the transmission of sensor

data are not taken into account in the design of the algorithms in this article.

With this communication scheme, the available sensor information on each module is identical. Therefore, the modules can run a copy of the same state estimator (see Figure 13). Consequently, the inputs to the controllers K_i are also identical. Hence, state estimation and control design are treated as the centralized problem given in Figure 14: the state estimator has access to all measurements and the controller K computes all system inputs. The distributed implementation of the feedback system in Figure 13 is then straightforward: a copy of the estimator runs on each module, and the local controllers K_i are obtained from K by selecting the output corresponding to the local actuator.

The communication protocol considered herein represents the case of maximal information in terms of sensor data and serves as a baseline for distributed or event-based algorithms that cope with a reduced set of sensor data. Depending on what aspect of a distributed and networked control system is to be studied, different protocols and topologies can be implemented on the balancing cube by constraining the IMC (for example, reducing the average communication rate such as in [40] and [41]).

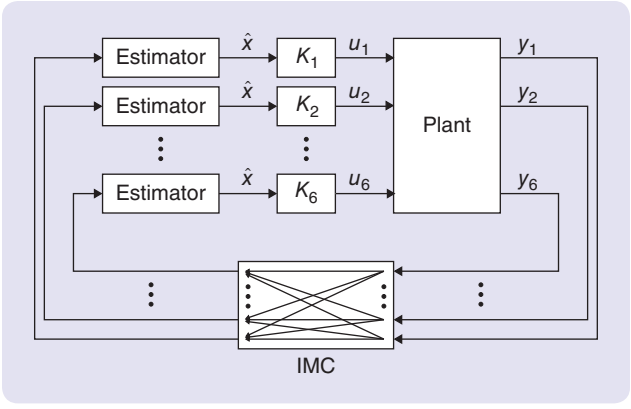


FIGURE 13 Distributed implementation of the control system. Each module i runs a state estimator and a controller K_i. The signal y_i combines all of module i's sensor measurements, and u_i denotes its control input. Different communication protocols can be implemented for the intermodule communication (IMC): for the design and results presented in this article, each module broadcasts its sensor measurements to all other modules.

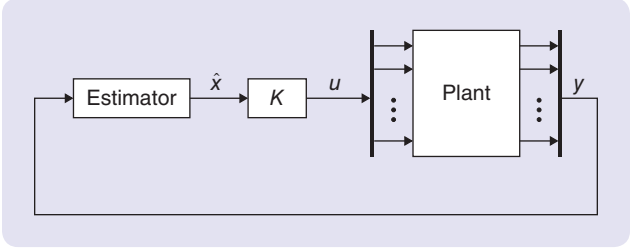


FIGURE 14 Centralized design problems. Since each module shares its sensor data with all other modules at every time step, the designs of the state estimator and the state-feedback controller K can be addressed as centralized problems. Therein, the state estimator has access to all sensor measurements $y = (y_1, \ldots, y_6)$, and the state-feedback controller K computes all system inputs $u = (u_1, \ldots, u_6)$.

STATE ESTIMATION

This section addresses the design of the centralized state estimator shown in Figure 14. Since the estimator is ultimately implemented on a digital computer, the estimator equations are expressed in discrete time. For this purpose, the discrete-time index $k \in \mathbb{N}$ is used: for a continuous-time signal $s(t)$, $s[k]$ denotes its value at time $t = kT_s$, where T_s is the sampling time; thus, $s[k] = s(kT_s)$. Measurements are assumed to be acquired at the discrete-time instants k. The objective of this section is to develop an algorithm that computes an estimate $\hat{x}[k]$ of the system state $x[k]$ based on all sensor measurements at time k.

In contrast to many standard methods for state estimation, such as the Luenberger observer [47] or the Kalman filter [48], the approach to state estimation presented herein requires neither the knowledge of a dynamic system model [for instance, in the form of (5)] nor knowledge of the system inputs $u[k]$. Instead, estimates of all states are computed from the sensor measurements only. In addition to reducing the modeling effort, an immediate favorable consequence of this approach is that the state estimator is robust to modeling errors in the system dynamics or their intentional modification (for example, when weights are added to the modules). Furthermore, the estimator works both for slow and fast motion, and irrespective of the operation mode (such as corner balancing, edge balancing, or the cube being moved into starting position by an operator).

The design of the state estimator is addressed below separately for the module and the cube states.

Module States

The module angle φ_i is measured by the absolute encoder on module i according to the sensor model (15). Since the quantization error is negligible for the balancing application presented herein, filtering of the encoder measurement is not necessary. The encoder measurement y_i^{enc} is hence directly used as the module angle estimate; that is,

$$\hat{x}_i[k] = y_i^{\text{enc}}[k], \quad i = 1, \ldots, 6. \tag{16}$$

An estimate of the module angular velocities $\dot{\varphi}_i$ may be readily obtained from the encoder measurements (15); for example, by numerical differentiation combined with appropriate lowpass filtering. Since the motors are operated with local velocity feedback ensuring fast tracking of velocity commands, actual estimates of the

module velocities are not required for the state feedback controller. This aspect is discussed in detail in the section "Control."

Cube States

Estimates of the cube's tilt (β, γ) and the tilt rates $(\dot{\beta}, \dot{\gamma})$ are obtained from the measurements of the six IMUs on the cube body, each of which includes a triaxis accelerometer and rate gyro. In a first step, the accelerometer measurements are used to generate an estimate of the tilt angles that is independent of the rigid body motion. This estimate is fused in a second step with an estimate of the tilt rates, which itself is obtained from the rate gyro measurements.

Tilt Estimate from Accelerometers

An estimate of the cube's tilt is obtained by the following approach: first, an unbiased estimate ${}^B\hat{g}$ of the gravity vector ${}^Bg$ in the body frame is derived as a linear combination of all accelerometer measurements based on the model (8). Second, the *accelerometer-based tilt estimate* $(\hat{\beta}^{\text{acc}}[k], \hat{\gamma}^{\text{acc}}[k])$ is constructed from the gravity vector estimate using (10) and the representation (1) for the rotation matrix ${}^B_O S$.

In static conditions, a single triaxis accelerometer mounted on a rigid body is enough to measure the gravity vector ${}^Bg$ in body frame. In fact, from (8) it can be seen that, for static conditions where ${}^O_B\ddot{S} = 0$ and hence $\tilde{S} = 0$, each individual accelerometer measurement ${}^By_i^{\text{acc}}$ is an unbiased estimate of the gravity vector ${}^Bg$. The difficulty in the context of the balancing cube arises from the fact that the rigid body moves during balancing, and therefore $\tilde{S} \neq 0$ in general.

The tilt estimation method presented herein makes use of multiple accelerometers mounted on the same rigid body. Exploiting the kinematics of the rigid body with only rotational DOFs and the knowledge of the different sensor locations allows one to compensate for the dynamic terms of the rigid body motion in the accelerometer measurements. The algorithm works for any rigid body that has only rotational DOFs and measurements from multiple triaxis accelerometers. To state the method for the general case, the constant N is used below to denote the number of sensors (for the cube, $N = 6$). The algorithm was first presented in [49].

Gravity Vector Estimate

The first objective is to obtain an estimate ${}^B\hat{g}$ of the gravity vector from all accelerometer measurements ${}^By_i^{\text{acc}}$, $i = 1, \ldots, N$ that is optimal in a least-squares sense.

For notational convenience, all N measurements (8) are combined into one matrix equation,

$$Y = XP + W, \tag{17}$$

$$Y := \begin{bmatrix} {}^By_1^{\text{acc}} & {}^By_2^{\text{acc}} & \cdots & {}^By_N^{\text{acc}} \end{bmatrix} \in \mathbb{R}^{3\times N}, \tag{18}$$

$$X := \begin{bmatrix} {}^Bg & \tilde{S} \end{bmatrix} \in \mathbb{R}^{3\times 4}, \tag{19}$$

$$P := \begin{bmatrix} 1 & 1 & \cdots & 1 \\ {}^Bp_1 & {}^Bp_2 & \cdots & {}^Bp_N \end{bmatrix} \in \mathbb{R}^{4\times N}, \tag{20}$$

$$W := \begin{bmatrix} {}^Bw_1^{\text{acc}} & {}^Bw_2^{\text{acc}} & \cdots & {}^Bw_N^{\text{acc}} \end{bmatrix} \in \mathbb{R}^{3\times N}, \tag{21}$$

where Y combines all accelerometer measurements, X is the matrix of unknown parameters, P is the matrix of known parameters (the sensor locations), and W combines all accelerometer noise vectors, with $\mathbb{E}[W] = 0$ and $\mathbb{E}[W^TW] = 3\sigma_{\text{acc}}^2 I$. Notice that Y, X, and W in (17) are time varying, whereas P is constant. The time index k is omitted for ease of notation.

In addition to the sought gravity vector ${}^Bg$, the unknown matrix X also contains the matrix $\tilde{S}$, which captures the second derivatives of the rigid body motion according to (9). In [49], a method is presented for optimally estimating the entire matrix X. To shorten the exposition, only the results for optimally estimating the gravity vector ${}^Bg$ are presented here.

An unbiased estimate ${}^B\hat{g}$ of the gravity vector ${}^Bg$ is sought such that the two-norm of the estimation error is minimized; that is,

$${}^B\hat{g} = \operatorname{argmin}_{{}^B\hat{g}} \mathbb{E}\left[\| {}^B\hat{g} - {}^Bg \|_2^2\right] \text{ subject to } \mathbb{E}[{}^B\hat{g}] = {}^Bg, \tag{22}$$

where $\|\cdot\|_2$ denotes the vector two-norm. The estimate ${}^B\hat{g}$ is restricted to linear combinations of the measurements Y; that is, a vector $\lambda \in \mathbb{R}^N$ is sought for ${}^B\hat{g} = Y\lambda$. This approach yields a straightforward implementation: at each time step, the estimate ${}^B\hat{g}$ is obtained by a single matrix-vector multiplication. The following proposition, the proof for which can be found in [49, Lemma 2.2], states the optimal unbiased linear estimate of the gravity vector.

Proposition 1

Let the matrices $P \in \mathbb{R}^{4\times N}$ and $Y \in \mathbb{R}^{3\times N}$ be given and satisfy $Y = XP + W$ with unknown matrix $X = [{}^Bg\ \tilde{S}] \in \mathbb{R}^{3\times 4}$ and the matrix random variable $W \in \mathbb{R}^{3\times N}$ with $\mathbb{E}[W] = 0$, $\mathbb{E}[W^TW] = \sigma_W^2 I$. Assuming P has full row rank, the (unique) minimizer $\lambda^* \in \mathbb{R}^{N\times 1}$ of

$$\min_\lambda \mathbb{E}\left[\| Y\lambda - {}^Bg \|_2^2\right] \text{ subject to } \mathbb{E}[Y\lambda] = {}^Bg \tag{23}$$

is given by $\Lambda^* = [\lambda^* \Lambda_2^*] = P^T(PP^T)^{-1}$.

Applying Proposition 1 and reintroducing time index k for all time-variant quantities yields the gravity vector estimate ${}^B\hat{g}[k]$,

$${}^B\hat{g}[k] = Y[k]\lambda^*. \tag{24}$$

The *optimal fusion vector* λ^* is constant and entirely defined by the geometry of the problem (through P).

In Proposition 1 it is assumed that P has full row rank. To obtain a physical interpretation of this assumption, consider the case where P does not have full row rank. Then, there exists a nontrivial linear combination of the rows of P; that is,

$$\begin{aligned} &\text{there exists } \xi \neq 0 \in \mathbb{R}^4 \\ &\quad \text{such that } \xi_1 p_X + \xi_2 p_Y + \xi_3 p_Z + \xi_4 \mathbf{1} = 0, \end{aligned} \tag{25}$$

where $p_X^T, p_Y^T, p_Z^T \in \mathbb{R}^{1\times N}$ are the last three rows of P (the vectors of X, Y, and Z-coordinates of all sensor locations) and $\mathbf{1}^T \in \mathbb{R}^{1\times N}$ is the vector of all ones. Expression (25) is equivalent to the statement

$$\begin{aligned} &\text{there exists } \xi \neq 0 \in \mathbb{R}^4 \\ &\quad \text{such that } \xi_1 {}^B p_{X,i} + \xi_2 {}^B p_{Y,i} + \xi_3 {}^B p_{Z,i} = -\xi_4 \\ &\qquad\qquad\qquad\qquad \text{for all } i = 1, \ldots, N \end{aligned} \tag{26}$$

where ${}^B p_{X,i}$, ${}^B p_{Y,i}$, ${}^B p_{Z,i}$ denote the X, Y, and Z-coordinate of the ith sensor location in the body frame. Since the equation $\xi_1 x + \xi_2 y + \xi_3 z = -\xi_4$ defines a plane in (x,y,z)-space, condition (26) is equivalent to *all* N sensors lying on the same plane. Therefore, the full row rank condition on P is satisfied if and only if *not* all sensors lie on the same plane. Moreover, since three points always lie on a plane, this result implies that at least four triaxis accelerometers are required for the method presented herein. For the cube, the full row rank assumption of P is satisfied.

The gravity vector estimate (24) can be shown to be independent of the rigid body dynamics (captured in $\tilde{S}$) as follows: from the singular value decomposition of the parameter matrix P,

$$P = U[\Sigma \;\; 0]\begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix} = U\Sigma V_1^T, \tag{27}$$

with $U \in \mathbb{R}^{4\times 4}$ unitary, $\Sigma \in \mathbb{R}^{4\times 4}$ diagonal, $V_1 \in \mathbb{R}^{N\times 4}$, $V_2 \in \mathbb{R}^{N\times(N-4)}$, and $V = [V_1 V_2]$ unitary, it can be verified that $\Lambda^* = V_1 \Sigma^{-1} U^T$. With this result and using the partition $U^T = [U_1^T U_2^T]$, $U_1^T \in \mathbb{R}^{4\times 1}$, $U_2^T \in \mathbb{R}^{4\times 3}$, the gravity vector estimate can be written as

$$\begin{aligned} {}^B\hat{g} &= Y\lambda^* = XP\lambda^* + W\lambda^* \\ &= [{}^B g \quad \tilde{S}]\underbrace{U\Sigma V_1^T}_{P}\underbrace{V_1\Sigma^{-1}U_1^T}_{\lambda^*} + W\lambda^* \\ &= [{}^B g \quad \tilde{S}]\begin{bmatrix} U_1 \\ U_2 \end{bmatrix} U_1^T + W\lambda^* \\ &= [{}^B g \quad \tilde{S}]\begin{bmatrix} 1 \\ 0 \end{bmatrix} + W\lambda^* = {}^B g + W\lambda^*. \end{aligned} \tag{28}$$

Obviously, the matrix $\tilde{S}$ does not appear in the estimate; that is, the gravity vector observation is not affected by any motion of the rigid body. As expected, the sensor noise W does enter the estimation equation.

Tilt (Pitch and Roll) Estimate

With the estimate ${}^B\hat{g}$ of the gravity vector in the body frame, (10) can be used to compute an estimate of the rigid body tilt (β,γ), since the direction of the gravity vector in the inertial frame is known. Inserting the representation (1) for ${}^O_B S$, and ${}^O g = [0\ 0\ g_0]^T$, with gravity constant g_0, (10) can be rewritten as

$${}^B g = S_X^T(\gamma) S_Y^T(\beta) S_Z^T(\alpha) {}^O g = g_0 \begin{bmatrix} -\sin(\beta) \\ \sin(\gamma)\cos(\beta) \\ \cos(\gamma)\cos(\beta) \end{bmatrix}. \tag{29}$$

Given the estimate of the gravity vector (24), *the accelerometer-based tilt estimate* at time k is

$$\hat{\beta}^{\text{acc}}[k] = \text{atan2}\left(-{}^B\hat{g}_x[k], \sqrt{{}^B\hat{g}_y^2[k] + {}^B\hat{g}_z^2[k]}\right), \tag{30}$$

$$\hat{\gamma}^{\text{acc}}[k] = \text{atan2}\left({}^B\hat{g}_y[k], {}^B\hat{g}_z[k]\right), \tag{31}$$

where atan2 is the four-quadrant inverse tangent. Note that the gravity constant g_0 does not need to be known and, hence, the estimator does not to be calibrated for it. The estimator requires only knowledge of the accelerometer locations on the cube body.

Tilt Rate Estimate from Rate Gyros

Estimates of the tilt rates are obtained from the rate gyro measurements (12) and the transformation (13). First, an estimate ${}^B\hat{\omega}[k]$ of the rotation vector ${}^B\omega[k]$ at time k is computed by averaging the available rate gyro measurements,

$${}^B\hat{\omega}[k] = \frac{1}{6}\sum_{i=1}^{6} {}^B y_i^{\text{gyro}}[k]. \tag{32}$$

Given that the rate gyro sensors have identical noise variance, the average represents the best unbiased linear estimate of ${}^B\omega[k]$ minimizing the mean squared error. Using the transformation (13), an estimate of yaw, pitch and roll rates is given by

$$\begin{bmatrix} \hat{\dot{\alpha}}[k] \\ \hat{x}_{14}[k] \\ \hat{x}_{16}[k] \end{bmatrix} = \begin{bmatrix} \hat{\dot{\alpha}}[k] \\ \hat{\dot{\beta}}[k] \\ \hat{\dot{\gamma}}[k] \end{bmatrix} = T(\hat{\beta}[k-1], \hat{\gamma}[k-1])\, {}^B\hat{\omega}[k], \tag{33}$$

where the transformation matrix $T(\cdot,\cdot)$ is evaluated at the yet undefined estimates $\hat{\beta}[k-1]$ and $\hat{\gamma}[k-1]$ of pitch and roll angle at the previous time step. They are made precise in the next subsection. The estimate of the yaw rate $\hat{\dot{\alpha}}[k]$ is not required for the balancing application presented in this article.

Sensor Fusion

To further reduce the noise level of the tilt estimate, the accelerometer-based tilt estimate (30), (31) is fused with the integrated estimate of the tilt rates (33). The *tilt estimate*

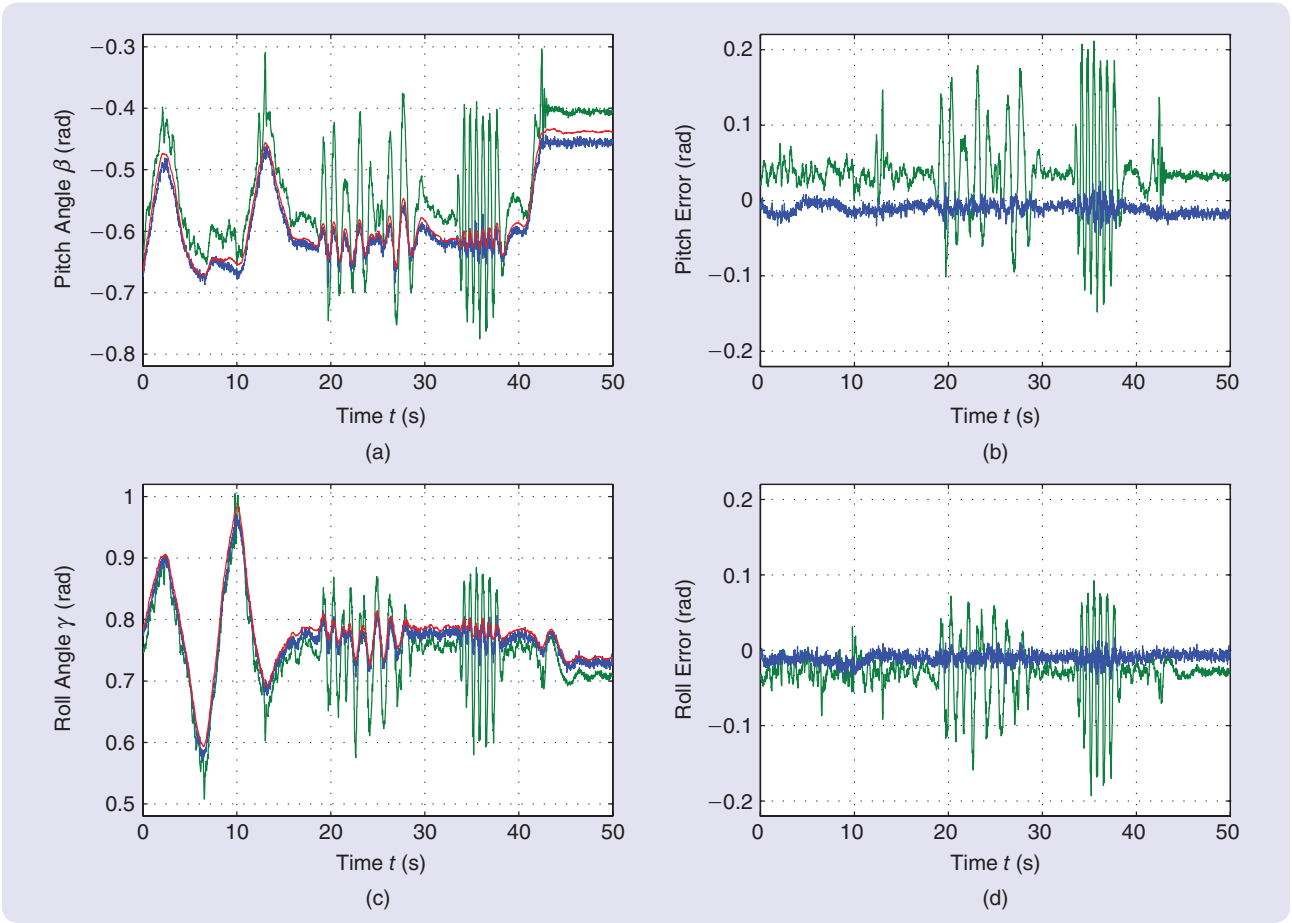


FIGURE 15 Verification of the accelerometer-based tilt estimator. For this experiment, the cube was moved manually around the nominal equilibrium (3) ($\beta_0^c = -0.615$ and $\gamma_0^c = 0.785$). The pitch and roll estimates $\hat{\beta}^{\text{acc}}$ and $\hat{\gamma}^{\text{acc}}$ are shown in the graphs on the left (in blue) with their camera-based reference (red). For comparison, the graph in green is the tilt estimate that results if a single triaxis accelerometer measurement is used directly as an estimate of the gravity vector [instead of the fusion equation (24)]. The corresponding error signals are shown on the right. Clearly, the multi-accelerometer-based estimator outperforms the single-accelerometer one, especially when the cube is being moved fast. The static biases visible in the estimation error signals result from biases in the accelerometers and/or in the camera system. The biases are, however, irrelevant for the balancing application, since biases in the state estimates are compensated by integral action in the controller. This effect is analyzed in "What Is the Effect of Integral Action in the Controller?" (The experimental data is taken from [49].)

$(\hat{\beta}[k], \hat{\gamma}[k])$ is obtained from a linear combination of accelerometer- and gyro-based estimates according to

$$\hat{x}_{13}[k] = \hat{\beta}[k] = \kappa_1 \hat{\beta}^{\text{acc}}[k] + (1-\kappa_1)\big(\hat{\beta}[k-1] + T_s \hat{\dot{\beta}}[k]\big), \quad (34)$$

$$\hat{x}_{15}[k] = \hat{\gamma}[k] = \kappa_2 \hat{\gamma}^{\text{acc}}[k] + (1-\kappa_2)\big(\hat{\gamma}[k-1] + T_s \hat{\dot{\gamma}}[k]\big), \quad (35)$$

where T_s is the sampling time and κ_1 and κ_2 are tuning parameters, which, given the noise specifications of accelerometers and rate gyros, can be chosen to minimize the estimation error variance.

Experimental Validation

The estimator for the cube states is validated in [49] using a camera-based global positioning system, which tracks the position and orientation of the cube body with submillimeter precision. The main experimental results from [49] are restated here. For further details, the reader is referred to the original publication.

Figure 15 shows the accelerometer-based tilt estimate (30), (31) in comparison to the camera-based reference measurement. For this experiment, the cube was moved manually about the nominal corner balancing equilibrium (3). Additionally, the tilt estimate is included that would result if only a single triaxis accelerometer was used to observe the gravity vector (instead of (24), the accelerometer measurement (8) is used directly as an estimate of the gravity vector). The data shows that, when the cube is relatively static (from 45 s to 50 s), the single-accelerometer-based estimate is satisfactory. However, when the cube body is moving fast, the estimate suffers from the dynamic terms that act as disturbances to the

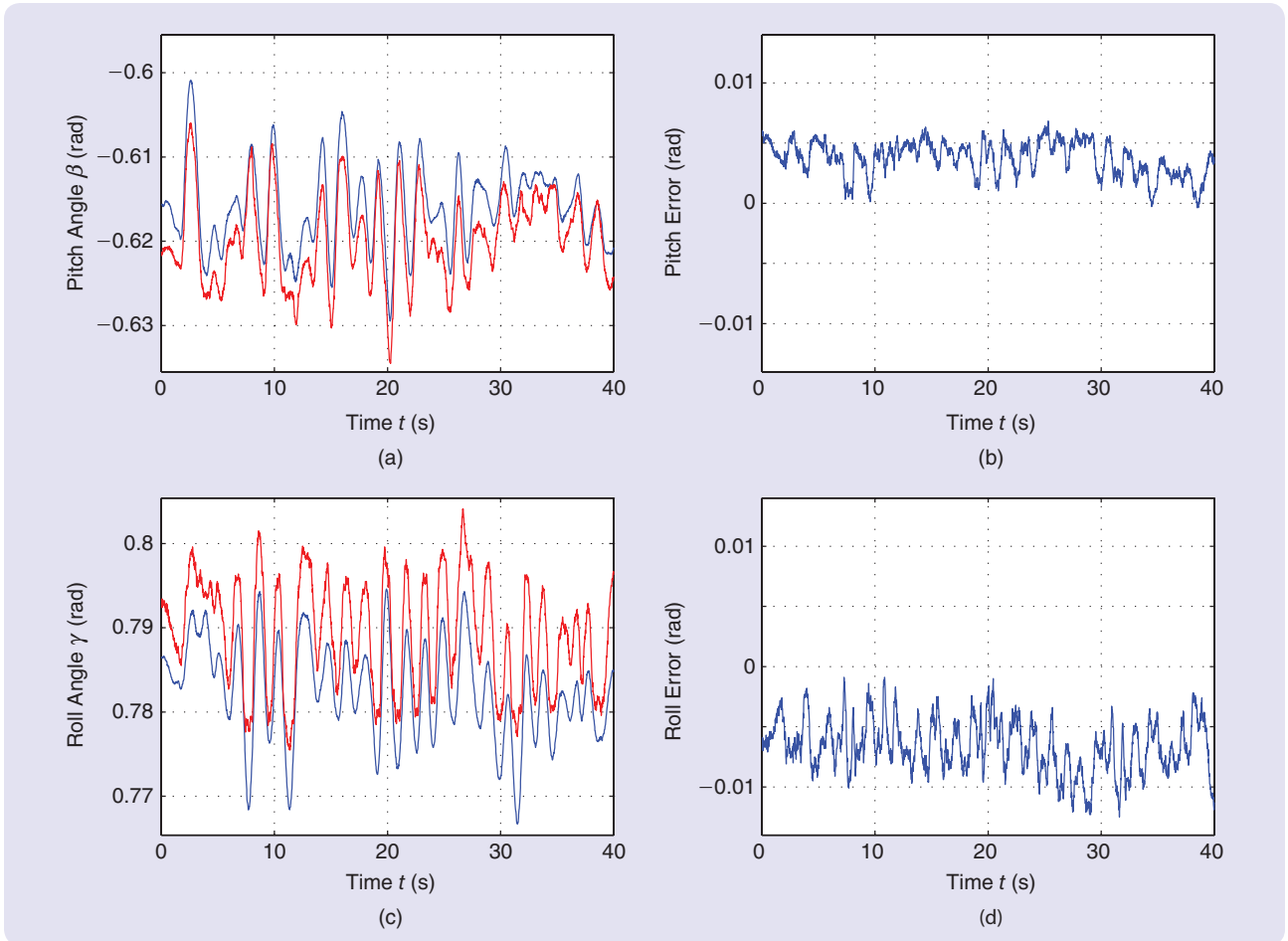


FIGURE 16 Verification of the tilt estimator. The data was taken while the cube was balancing about the nominal equilibrium (3) ($\beta_0^c = -0.615$ and $\gamma_0^c = 0.785$). The pitch and roll estimates $\hat{\beta}$ and $\hat{\gamma}$, resulting from fusing accelerometer and rate gyro measurements, are shown on the left (blue) with the camera-based reference (red). The graphs on the right show the corresponding error signals. Constant biases in the state estimates are compensated by integral action in the control algorithm. (The experimental data is taken from [49].)

single-accelerometer-based estimator. The experimental data illustrates the result discussed in (28), namely that the presented (multi) accelerometer-based tilt estimator compensates for the rigid body dynamics.

Figure 16 shows the comparison of the improved tilt estimates (34) and (35), which are based on accelerometer and rate gyro data, to the camera-based reference during autonomous balancing of the cube.

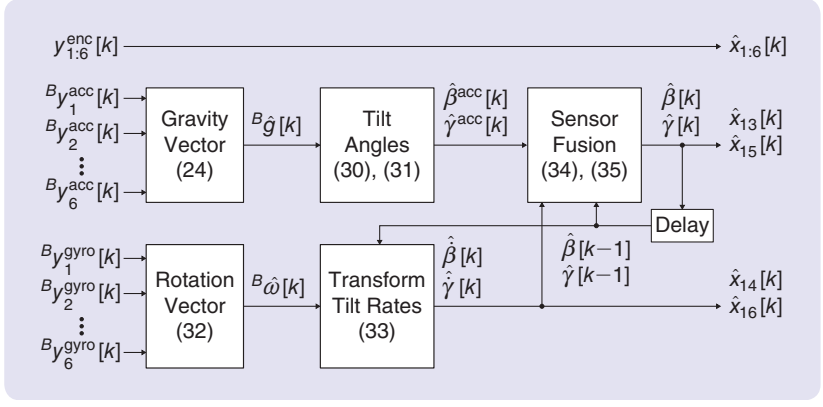


FIGURE 17 The state estimator. All encoder, accelerometer, and rate gyro measurements are input to the estimator; its outputs are the estimates of all states that are required for feedback control (estimates of the module velocities $x_{7:12}[k]$ are not required as is discussed in the section "Control").

Summary: The Complete State Estimator

The complete state estimator is given by (16), (24), and (30)–(35). It is summarized in the block diagram of Figure 17. Memory is present only in the sensor fusion in (34) and (35); hence, the estimator has only two states. The estimator is nonlinear because of the nonlinear transformations in (30), (31), and (33).

None of the estimator equations in Figure 17 depends on the system dynamics. The only assumption on the rigid body is that it is pivoting (it has only rotational DOFs). Hence, the estimator works irrespective of the system dynamics and, in particular, for slow and fast motion, for different mass configurations, and for both edge and corner balancing.

CONTROL

The design of the centralized controller K shown in Figure 14 is described in this section. The controller is designed as an LQR with additional integrator feedback on the module angles. The controller design is based on the model (5). The main control objective is the stabilization of (5); that is, to balance the cube about the corresponding equilibrium (3) or (4).

Time-Scale Separation Algorithm

The time-scale separation algorithm is a transformation of a continuous-time, state-space model (A,B) to a discrete-time model $(\tilde{A},\tilde{B})$ that represents (A,B) under high-gain feedback on some of its states through a controller K_{loc} (see Figure S3). The method allows one to obtain a simplified model of the feedback system without detailed knowledge of the controller K_{loc}, by approximating it as an ideal controller with infinite proportional gain. This approximation is legitimate as long as the time scales of the feedback loop dynamics are sufficiently smaller than those of the remaining system. The resulting model can be used, for example, to design an outer-loop controller that feeds references to the inner-loop controller K_{loc}. The algorithm was first presented in [51].

Consider the continuous-time, state-space representation

$$\begin{bmatrix}\dot{x}_{\text{f}}(t)\\ \dot{x}_{\text{s}}(t)\end{bmatrix}=\underbrace{\begin{bmatrix}A_{\text{ff}} & A_{\text{fs}}\\ A_{\text{sf}} & A_{\text{ss}}\end{bmatrix}}_{\tilde{A}}\begin{bmatrix}x_{\text{f}}(t)\\ x_{\text{s}}(t)\end{bmatrix}+\underbrace{\begin{bmatrix}B_{\text{f}}\\ B_{\text{s}}\end{bmatrix}}_{\tilde{B}}u(t), \tag{S4}$$

where the states $x(t)$ are separated into those on which local feedback is applied ($x_{\text{f}}(t)\in\mathbb{R}^{n_{\text{f}}}$, index f for "fast") and the remaining ones ($x_{\text{s}}(t)$, index s for "slow"). To model the controller K_{loc}, proportional feedback on the states x_{f} is assumed to be of the form

$$u(t)=B_{\text{f}}^{-1}F(v(t)-x_{\text{f}}(t)), \tag{S5}$$

where $v(t)\in\mathbb{R}^{n_{\text{f}}}$ is a piecewise constant reference signal changing at a rate T_{s}, $F=\text{diag}(f_1,f_2,\ldots,f_{n_{\text{f}}})$ is a diagonal matrix with entries f_j, and B_{f} is assumed invertible. The feedback (S5) means that individual loops are closed on the states $x_{\text{f}}(t)$.

To obtain the discrete-time representation $(\tilde{A},\tilde{B})$, the feedback system given by (S4) and (S5) is first discretized at the sampling rate T_{s}. To represent ideal feedback loops, the controller gains f_j in (S5) are chosen to approach infinity in the limit. The model $(\tilde{A},\tilde{B})$ is then obtained using a limiting property of the matrix exponential, which is derived in [51]. The resulting model has the structure

$$\begin{bmatrix}x_{\text{f}}[k+1]\\ x_{\text{s}}[k+1]\end{bmatrix}=\underbrace{\begin{bmatrix}0 & 0\\ \tilde{A}_{\text{sf}} & \tilde{A}_{\text{ss}}\end{bmatrix}}_{\tilde{A}}\begin{bmatrix}x_{\text{f}}[k]\\ x_{\text{s}}[k]\end{bmatrix}+\underbrace{\begin{bmatrix}I\\ \tilde{B}_{\text{s}}\end{bmatrix}}_{\tilde{B}}v[k]. \tag{S6}$$

The details of the derivation are omitted here, but can be found in [51]. Notice from (S6) that $x_{\text{f}}[k+1]=v[k]$; that is, the reference is achieved in one time step, which corresponds to the infinite gain assumption.

APPLICATION TO THE CUBE MODEL

The time-scale separation algorithm is used to compute a model for the block $(\tilde{A},\tilde{B})$ in Figure 18. For this purpose, the state-space model (5) is partitioned in blocks corresponding to module angles, module velocities, and cube states

$$\begin{bmatrix}\dot{x}_{1:6}(t)\\ \dot{x}_{7:12}(t)\\ \dot{x}_{13:n}(t)\end{bmatrix}=\begin{bmatrix}0 & I & 0\\ A_{21} & A_{22} & A_{23}\\ A_{31} & A_{32} & A_{33}\end{bmatrix}\begin{bmatrix}x_{1:6}(t)\\ x_{7:12}(t)\\ x_{13:n}(t)\end{bmatrix}+\begin{bmatrix}0\\ B_2\\ B_3\end{bmatrix}u(t). \tag{S7}$$

Through the controllers on each drive unit, local feedback is applied on the module velocities $x_{7:12}(t)$ (see Figure 18). Hence, the time-scale separation technique above is applied with $x_{\text{f}}(t)=x_{7:12}(t)$ and $x_{\text{s}}(t)=(x_{1:6}(t),x_{13:n}(t))$. Since each module has an individual torque input, $B_{\text{f}}=B_2$ is invertible. The reference signals $v[k]$ are the velocity commands sent to the motors at an update rate of $T_{\text{s}}=0.01$ s. The resulting model corresponding to (S6) is

$$\underbrace{\begin{bmatrix}x_{1:6}[k+1]\\ x_{7:12}[k+1]\\ x_{13:n}[k+1]\end{bmatrix}}_{x[k+1]}=\underbrace{\begin{bmatrix}I & 0 & 0\\ 0 & 0 & 0\\ \tilde{A}_{31} & \tilde{A}_{32} & \tilde{A}_{33}\end{bmatrix}}_{\tilde{A}}\underbrace{\begin{bmatrix}x_{1:6}[k]\\ x_{7:12}[k]\\ x_{13:n}[k]\end{bmatrix}}_{x[k]}+\underbrace{\begin{bmatrix}T_{\text{s}}I\\ I\\ \tilde{B}_3\end{bmatrix}}_{\tilde{B}}v[k]. \tag{S8}$$

The discrete-time model (S8) captures the dynamics of the cube including the local velocity feedback. The numerical values for the matrices $(\tilde{A},\tilde{B})$ may be found in [44].

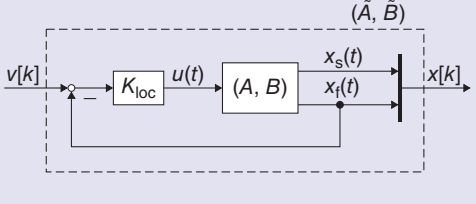


FIGURE S3 System with partial state feedback. The system is described by the continuous-time, state-space model (A,B). Feedback loops are closed on some states $x_{\text{f}}(t)$ through controller K_{loc} with reference input $v[k]$ changing at a rate T_{s}. The time-scale separation algorithm computes a discrete-time model $(\tilde{A},\tilde{B})$ with sampling time T_s under the assumption of an ideal feedback controller K_{loc} with infinitely high proportional gains.

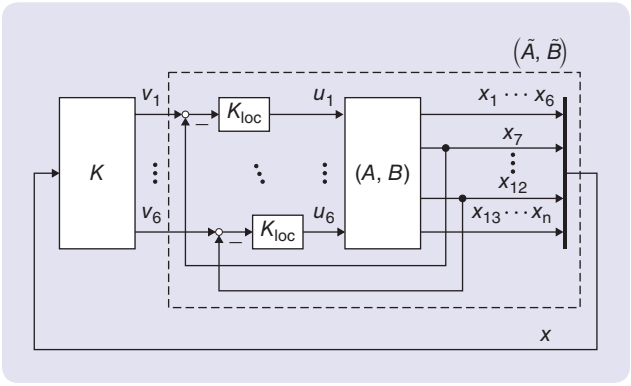


FIGURE 18 The cascaded control architecture. The block (A,B) denotes the cube model (5) and the blocks K_{loc} represent the local velocity controllers on each drive unit. The dashed block combines the plant with the local feedback loops. A simplified model $(\tilde{A},\tilde{B})$ for this block is derived in "Time-Scale Separation Algorithm" with the assumption of high-gain controllers K_{loc}. The controller K is the centralized state-feedback controller that is also shown in Figure 14. When comparing Figures 14 and 18, notice that the inner feedback loops are not shown in Figure 14; they can be thought of as included in the block *Plant*. Furthermore, the *Estimator* block is omitted here, since a plant with state output is assumed for the purpose of controller design.

System (5) captures the linearized pitch and roll dynamics of the cube; the controller thus locally stabilizes pitch and roll. Control of the yaw angle is not considered herein (the yaw motion is negligible in typical operation of the cube). The problem of simultaneously stabilizing yaw, pitch, and roll angles of a 3-D pendulum is addressed in [22]–[24].

The drive units are operated in velocity mode; that is, the motor shaft angular velocity is controlled locally on each drive unit. When neglecting gear backlash, the shaft velocity is proportional to the module velocity by the gear ratio; hence the feedback on the drive units is treated as feedback on the states $(x_7,\ldots,x_{12})$. The corresponding controllers are denoted by K_{loc}. Their reference input v is computed by the controller K. This cascaded control architecture is shown in Figure 18. It consists of the *inner-loop controllers* K_{loc} and the *outer-loop controller* K. The inner-loop controllers K_{loc} operate at an update rate of 1 kHz, whereas the outer loop runs at rate of 100 Hz. The parameters of K_{loc} are tuned using a software tool provided by the manufacturer of the drive unit.

In contrast to designing a controller for system (5) directly (that is, a controller that computes the torque inputs u), the cascaded architecture with fast local velocity feedback reduces the complexity of the controller design problem. In model (5), it is assumed that the torque at the module can be controlled directly. In reality, however, only the torque at the motor can be controlled. This is translated to the torque at the module in a nontrivial way through a transmission system, which involves nonlinearities such as kinetic and static friction and backlash. The application of high-gain velocity feedback mitigates the effect of the nonidealities in the actuation mechanism and allows one to abstract them away for the design of the controller K.

Simplified Model Incorporating Local Feedback Loops from Time-Scale Separation

The design of the outer-loop controller K requires a system model that incorporates the effect of the inner loops (represented by the dashed block $(\tilde{A},\tilde{B})$ in Figure 18). To avoid modeling the details of the local controllers K_{loc}, the motor and its local controller are abstracted as a system that achieves a commanded module velocity sufficiently fast. In fact, the ideal case of infinitely fast feedback is considered. This approximation is legitimate as long as the inner control loop operates sufficiently faster than the outer loop (the tracking performance of the inner loop is discussed in the section "Experiments"). The method described in "Time-Scale Separation Algorithm" is used to compute a model of system (5) that incorporates the feedback on the module velocities. The obtained model is a discrete-time model with the sampling time of the outer-loop controller, $T_s = 0.01$ s,

$$\underbrace{\begin{bmatrix} x_{1:6}[k+1] \\ x_{7:12}[k+1] \\ x_{13:n}[k+1] \end{bmatrix}}_{x[k+1]} = \underbrace{\begin{bmatrix} I & 0 & 0 \\ 0 & 0 & 0 \\ \tilde{A}_{31} & \tilde{A}_{32} & \tilde{A}_{33} \end{bmatrix}}_{\tilde{A}} \underbrace{\begin{bmatrix} x_{1:6}[k] \\ x_{7:12}[k] \\ x_{13:n}[k] \end{bmatrix}}_{x[k]} + \underbrace{\begin{bmatrix} T_s I \\ I \\ \tilde{B}_3 \end{bmatrix}}_{\tilde{B}} \upsilon[k] \qquad (36)$$

where $x_{i:j} := (x_i,\ldots,x_j)$. For easier reference below, the notation from "Time-Scale Separation Algorithm" is adopted; that is, the module velocity states $x_{7:12}$ are denoted by x_{f} (f for "fast") and the remaining states by x_s (s for "slow"). Notice from (36) that $x_{\text{f}}[k+1] = v[k]$ (the module velocities are equal to the previously commanded reference), which corresponds to the assumption of ideal feedback loops. By using this approximation in the controller, no measurements or estimates of the module velocities are required. Estimates for the states x_{s} are available from the state estimator shown in Figure 17.

State-Feedback Controller Design

The controller K is obtained from a discrete-time LQR design. In LQR design (see [50], for example), a quadratic cost function involving weights on system states x and system inputs υ is minimized, which allows one to trade off control performance with control effort. The resulting optimal controller is a static feedback gain.

To ensure zero steady-state error of the module angles, plant (36) is first augmented with integrator states on the module angles; that is, the system model used for the LQR design is

$$\begin{bmatrix} x[k+1] \\ x_{\text{int}}[k+1] \end{bmatrix} = \begin{bmatrix} \tilde{A} & 0 \\ [T_s I_{6\times 6}\ 0] & I \end{bmatrix} \begin{bmatrix} x[k] \\ x_{\text{int}}[k] \end{bmatrix} + \begin{bmatrix} \tilde{B} \\ 0 \end{bmatrix} v[k], \qquad (37)$$

where $x_{\text{int}}[k]$ are the augmented integrator states, and $I_{6\times 6}$ is the six-by-six identity matrix. The augmented integrator states

What Is the Effect of Integral Action in the Controller?

Integral control is typically used to improve the steady-state behavior of a feedback control system. In particular, the quantity whose integral is used in a feedback controller (for example, a system state) is forced to zero when the system reaches steady state.

The control algorithm described in the section "Control" includes integral feedback on the module angles. The analysis below shows that the integrators in the controller ensure average zero steady-state error not only for the module angles, but for *all* states—despite a possible bias on the estimates of the cube states (pitch, roll, and their rates). First, the analysis is presented for a general linear time-invariant system, and then the results are interpreted for the balancing cube.

Consider the discrete-time system

$$x[k+1]=\begin{bmatrix}x_1[k+1]\\ x_2[k+1]\end{bmatrix}=\begin{bmatrix}A_{11} & A_{12}\\ A_{21} & A_{22}\end{bmatrix}x[k]+\begin{bmatrix}B_1\\ B_2\end{bmatrix}u[k]. \tag{S9}$$

The state measurements

$$y[k]=\begin{bmatrix}x_1[k]+w_1[k]\\ x_2[k]+w_2[k]+d_2\end{bmatrix} \tag{S10}$$

are corrupted by zero-mean sensor noise $w[k]=(w_1[k], w_2[k])$ and a static bias d_2. Notice that the measurement of x_1 is bias-free. A feedback controller with integral action on the measurement of state x_1 is used,

$$x_{\text{int}}[k+1]=x_{\text{int}}[k]+T_s(x_1[k]+w_1[k]) \tag{S11}$$

$$u[k]=F\begin{bmatrix}y[k]\\ x_{\text{int}}[k]\end{bmatrix}=\begin{bmatrix}F_1 & F_2 & F_{\text{int}}\end{bmatrix}\begin{bmatrix}x_1[k]+w_1[k]\\ x_2[k]+w_2[k]+d_2\\ x_{\text{int}}[k]\end{bmatrix}, \tag{S12}$$

where F is a static gain matrix, and T_s is the sampling time. The following assumptions are made on the system and the controller:

(A1) B_1 has full column rank.

(A2) $I-A_{22}+B_2(B_1^TB_1)^{-1}B_1A_{12}$ is invertible.

(A3) The closed-loop system given by (S9), (S11), and (S12) is stable.

It is argued below that, under these assumptions, all states $x[k]$ of (S9) have zero steady-state mean,

$$\lim_{k\to\infty}\mathbb{E}[x[k]]=0, \tag{S13}$$

for any static disturbance d_2.

The closed loop system given by (S9), (S11), and (S12) reads

$$\begin{bmatrix}x_1[k+1]\\ x_2[k+1]\\ x_{\text{int}}[k+1]\end{bmatrix}=\underbrace{\begin{bmatrix}A_{11}+B_1F_1 & A_{12}+B_1F_2 & B_1F_{\text{int}}\\ A_{21}+B_2F_1 & A_{22}+B_2F_2 & B_2F_{\text{int}}\\ T_sI & 0 & I\end{bmatrix}}_{=:\bar{A}}\underbrace{\begin{bmatrix}x_1[k]\\ x_2[k]\\ x_{\text{int}}[k]\end{bmatrix}}_{=:\bar{x}[k]}+\underbrace{\begin{bmatrix}B_1F_2\\ B_2F_2\\ 0\end{bmatrix}}_{\bar{B}}d_2+\begin{bmatrix}B_1F_1 & B_1F_2\\ B_2F_1 & B_2F_2\\ T_SI & 0\end{bmatrix}w[k]. \tag{S14}$$

Since $w[k]$ has zero mean, taking expected values yields

$$\mathbb{E}[\bar{x}[k+1]]=\bar{A}\mathbb{E}[\bar{x}[k]]+\bar{B}d_2. \tag{S15}$$

From (A3), it follows that $\mathbb{E}[\bar{x}[k]]$ converges to a constant $\bar{x}$,

$$\lim_{k\to\infty}\mathbb{E}[\bar{x}[k]]=\bar{x}, \tag{S16}$$

which is the solution to

$$\bar{x}=\bar{A}\bar{x}+\bar{B}d_2. \tag{S17}$$

From the equation for $\bar{x}_{\text{int}}$ in (S17) [with $\bar{A}$ and $\bar{B}$ as defined in (S14)], it follows that

$$\bar{x}_{\text{int}}=T_s\bar{x}_1+\bar{x}_{\text{int}}\Leftrightarrow\bar{x}_1=0, \tag{S18}$$

which is the aforementioned typical effect of integral action on x_1. Using this result, the top rows of (S17) read

$$0=(A_{12}+B_1F_2)\bar{x}_2+B_1F_{\text{int}}\bar{x}_{\text{int}}+B_1F_2d_2, \tag{S19}$$

which, with assumption (A1), yields

$$F_{\text{int}}\bar{x}_{\text{int}}=-(B_1^TB_1)^{-1}B_1^T(A_{12}+B_1F_2)\bar{x}_2-F_2d_2. \tag{S20}$$

Using this result, (S17) can be solved for $\bar{x}_2$,

$$\bar{x}_2=(A_{22}+B_2F_2)\bar{x}_2-B_2(B_1^TB_1)^{-1}B_1^T(A_{12}+B_1F_2)\bar{x}_2 - B_2F_2d_2+B_2F_2d_2 \tag{S21}$$

$$\Leftrightarrow(I-A_{22}+B_2(B_1^TB_1)^{-1}B_1^TA_{12})\bar{x}_2=0, \tag{S22}$$

which, by assumption (A2), has the unique solution $\bar{x}_2=0$. Hence, (S13) holds as claimed. Furthermore, from (S20), it can be seen that the bias d_2 maps to steady-state offsets in the integrator states given by the relation

$$F_{\text{int}}\bar{x}_{\text{int}}=-F_2d_2. \tag{S23}$$

INTERPRETATION FOR THE BALANCING CUBE

The cube model (36) corresponds to (S9) with x_1 being the module angles and x_2 the remaining states. The absolute encoder measurements of the module angles can be considered bias free. The controller (40)–(41) is of the form (S11)–(S12), and it can be shown that the assumptions (A1)–(A3) hold.

Hence, the control system ensures that the deviation from the equilibrium is zero for all states—on average and in the long run—despite any static offset in the estimates of the cube states. Such offsets result from IMU sensor biases and the fact that the cube's physical equilibrium is not exactly at the model-based pitch and roll equilibrium angles in (3) and (4).

An intuitive explanation for this property of the feedback system is as follows: enforced by the integral feedback, the module angles are on average at the module equilibrium angles given in (3) or (4). These module angles define a unique equilibrium for the cube tilt parameterized by pitch and roll angles. Since the feedback control system is stable, the average pitch and roll angles must be equal to the equilibrium angles—otherwise the cube would not be in equilibrium on average and, hence, it would fall.

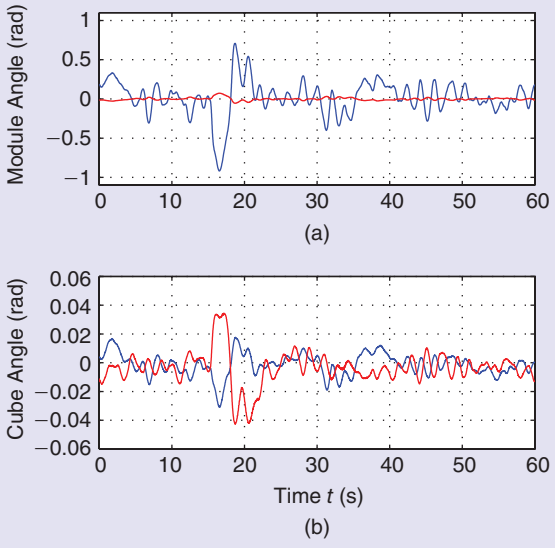


FIGURE 19 Corner balancing experiment. Part (a) shows module 2's angle $\hat{x}_2$ (blue) and module 5's angle $\hat{x}_5$ (red); part (b) shows the cube pitch $\hat{x}_{13}$ (blue) and roll $\hat{x}_{15}$ (red). All angles are about the nominal equilibrium (3). At roughly 15 s, the cube was disturbed by pushing one of its corners. From the data, it is obvious that module 2 moves more than module 5. The reason is that the bottom modules are more effective than the top modules and, hence, used more by the optimal state-feedback controller. This fact is investigated in "Why Are the Top Modules Used Less?"

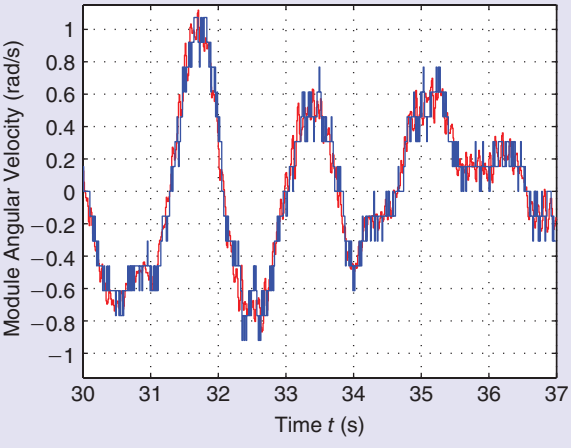


FIGURE 20 Command tracking for corner balancing experiment. The inner feedback loop on each module with controller K_{loc} tracks velocity commands from the outer-loop state-feedback controller K (compare Figure 18). As an example for the tracking performance of the inner loop, the angular velocity (blue) and the corresponding command (red) are shown for module 2. The module velocity is computed as the change of the absolute encoder angle signal per time step, $(y_2^{\text{enc}}[k] - y_2^{\text{enc}}[k-1])/T_s$. The module velocity signal shows a significant quantization error, which is caused by the quantization of the absolute encoder. The signal is, however, not used in feedback but only to demonstrate the tracking capabilities of the inner feedback loop here. The data is from the same experiment as in Figure 19.

are eventually implemented in the controller. In addition to ensuring that the module angles are, on average, at their set points, any offset in the cube tilt estimates (for example, due to sensor bias or imperfect calibration) is compensated by the integrators. This property of the control system is analyzed in detail in "What Is the Effect of Integral Action in the Controller?"

In addition to the usual weights on states and system inputs in LQR control, the difference in the control commands $v[k]-v[k-1]$ is also penalized. This is motivated by the special structure of the model (36): penalizing the difference in velocity commands corresponds to imposing a penalty on the applied module torque [the original system input in (5)] since change in velocity is proportional to acceleration, which itself relates to torque at a module. Therefore, the cost function

$$J = \sum_{k=0}^{\infty} \begin{bmatrix} x_s^T[k] & x_{\text{int}}^T[k] \end{bmatrix} Q \begin{bmatrix} x_s[k] \\ x_{\text{int}}[k] \end{bmatrix} + v^T[k]\, R\, v[k] + (v[k]-v[k-1])^T\, \Theta\, (v[k]-v[k-1]) \tag{38}$$

is used with suitable weighting matrices Q, R, and Θ (numerical values may be found in [44]). Since $v[k-1] = x_f[k]$, (38) can be reformulated as a standard LQR cost with nonzero weights on state and input cross terms,

$$J = \sum_{k=0}^{\infty} \tilde{x}^T[k] \begin{bmatrix} \Theta & 0 \\ 0 & Q \end{bmatrix} \tilde{x}[k] + v^T[k](R+\Theta)v[k] + 2\tilde{x}^T[k] \begin{bmatrix} -\Theta \\ 0 \end{bmatrix} v[k], \tag{39}$$

where $\tilde{x}[k] := (x_f[k],\ x_s[k],\ x_{\text{int}}[k])$. The discrete-time LQR design problem can be solved using standard tools [50] (such as the Matlab implementation dlqr). Let $F = [F_1\ F_2]$ be the resulting gain matrix with $F_1 \in \mathbb{R}^{6\times n}$ corresponding to the gains on $(x_f[k], A_s[k])$, and $F_2 \in \mathbb{R}^{6\times 6}$ corresponding to the gains on the integral states $x_{\text{int}}[k]$. Using the approximation $\hat{x}_f[k] = v[k-1]$ in addition, a representation of controller K then is

$$\zeta[k+1] = \zeta[k] + [T_s I_{6\times 6} \quad 0]\hat{x}_s[k] \tag{40}$$

$$v[k] = F_2\zeta[k] + F_1 \begin{bmatrix} v[k-1] \\ \hat{x}_s[k] \end{bmatrix}, \tag{41}$$

which has the state estimates $\hat{x}_s[k]$ from Figure 17 as input.

EXPERIMENTS

To compare the balancing performance in experiments, the root mean square (RMS) value of the state estimates,

$$x_i^{\text{RMS}} := \sqrt{\frac{1}{\bar{k}} \sum_{k=1}^{\bar{k}} \hat{x}_i^2[k]} \tag{42}$$

for data of length $\bar{k}$, is used (for the experimental evaluation, the motor shaft velocity measurement from the motor encoder scaled by the gear ratio is used as an estimate for the module velocity; estimates of all other states are as in

Figure 17). The RMS values for a sequence of 5 min of undisturbed balancing on a corner are given in Table 5 (middle column). The data shows that the RMS values for angular position and velocity differ by more than one order of magnitude for the bottom modules (1–3) versus the top ones (4–6) for corner balancing. The top modules are used significantly less because the corresponding inputs are much less effective than those of the bottom ones. This fact is explained by considering the analogy of a one dimensional abstraction of the balancing cube in "Why Are the Top Modules Used Less?"

In a different experiment, the cube was disturbed by pushing one of its corners. A 60-s balancing sequence with the disturbance applied at 15 s is shown in Figure 19. For a shorter sequence of the same experiment, the angular velocity command and the actual angular velocity of module 2 are shown in Figure 20. The data demonstrates the tracking capability of the inner velocity controllers, which was assumed for the derivation of model (36) in "Time-Scale Separation Algorithm."

From the data in Table 5 and Figures 19 and 20, it can be seen that the cube exhibits slight motion during balancing; that is, it is not perfectly steady. Some motion of the cube is inevitable due to excitation of the feedback system by sensor noise. Other effects such as network delays or gear backlash may be partially compensated for with a more sophisticated controller design. A discussion on the achievable balancing performance of the cube based on the $\mathcal{H}_2$ system norm is presented in "How Steady Can the Cube Balance?" The slight oscillations during balancing do, however, enable viewers to perceive the cube as a dynamic sculpture.

Experimental data for balancing the cube on one of its edges is shown in Table 5 (right column) and Figure 21. It can be seen that the cube exhibits less motion in edge balancing compared to corner balancing. This behavior is expected, since only one DOF needs to be stabilized when on edge (compared to two for corner balancing) with the same number of actuators.

TABLE 5 Balancing performance. The root mean square (RMS) value x_i^{RMS} according to (42) measures the average squared deviation of the state estimates from the equilibrium $x = 0$. It is used as a measure for the control performance. The data for corner balancing shows that the motion of the bottom modules (1–3) is an order of magnitude greater than the motion of the top ones (4–6). The reason for this is explained in "Why Are the Top Modules Used Less?" Also for edge balancing, the motion of modules 1–3 is greater than the motion of the modules 4–6. The difference however, is not as pronounced as for corner balancing. Due to the symmetrical arrangement of modules 1 and 3, and modules 4 and 6, their RMS values are almost the same. As expected, the balancing performance on edge is better than on corner since only one dimension needs to be stabilized (instead of two).

State	RMS Value x_i^{RMS} Corner Balancing	RMS Value x_i^{RMS} Edge Balancing
Angle module 1	0.1036	0.0434
Angle module 2	0.1214	0.0581
Angle module 3	0.1225	0.0433
Angle module 4	0.0080	0.0158
Angle module 5	0.0092	0.0255
Angle module 6	0.0098	0.0159
Angular velocity module 1	0.2688	0.1034
Angular velocity module 2	0.3266	0.1446
Angular velocity module 3	0.3227	0.1029
Angular velocity module 4	0.0275	0.0382
Angular velocity module 5	0.0316	0.0658
Angular velocity module 6	0.0281	0.0391
Cube pitch angle	0.0048	0.0034
Cube pitch rate	0.0103	0.0061
Cube roll angle	0.0066	—
Cube roll rate	0.0144	—

CONCLUDING REMARKS

This article presents the balancing cube and, in particular, the control system that enables the cube to live up to its name. The cube is a multiagent 3-D inverted pendulum system: stability is achieved through the coordination of six rotating bodies on the cube—each equipped with sensors, actuation, and a computer, and all communicating with each other over a digital network. With this architecture, the balancing cube combines the

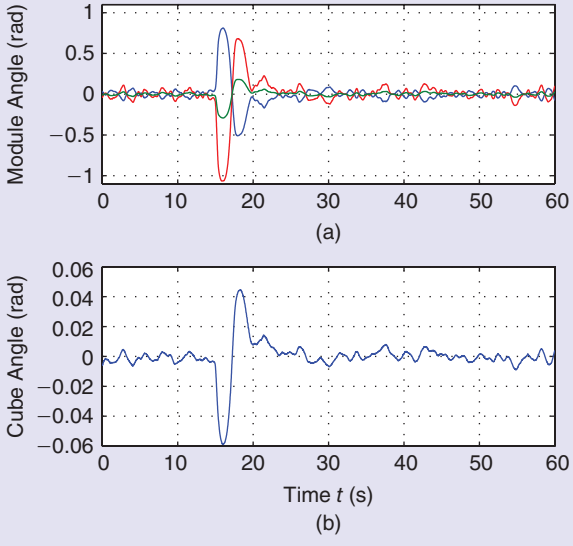


FIGURE 21 Edge balancing experiment. Shown in the top graph are the angles of modules 1, 2, and 6, $\hat{x}_1$ (blue), $\hat{x}_2$ (red), and $\hat{x}_6$ (green), respectively, and, in the bottom graph, the cube pitch angle $\hat{x}_{13}$. All angles are about the nominal equilibrium (4). At roughly 15 s, the system was disturbed by pushing the cube. The motion of module 6, which is one of the lighter modules, is less than the motion of modules 1 and 2.

Why Are the Top Modules Used Less?

When the cube balances on one of its corners, the top modules move considerably less than the bottom ones (see the section "Experiments"). To understand the impact of the mounting position on a module's ability to balance the cube, a one-dimensional (1-D) abstraction of the balancing problem is considered first. The controllability of an inverted pendulum that is balanced by a single module is analyzed as a function of the module's mounting height on the pendulum. The insights of the 1-D analysis are then interpreted for the cube.

An inverted pendulum of the above type is shown in Figure S4. The rotating arm that balances the planar inverted pendulum about the single rotational DOF is identical to the moving part of the modules on the cube. The system is essentially a double pendulum with torque applied at the joint linking the two bodies (see Figure S5). This type of double pendulum has also been termed an acrobot, and it is a typical example of an underactuated system [38], [S4]. For the analysis herein, the two bodies are approximated as point masses.

The equations of motion of the point-mass acrobot are, [S4],

$$H(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \begin{bmatrix} 0 \\ \tau \end{bmatrix}, \tag{S24}$$

where q are the generalized coordinates (angles of the two links, defined in Figure S5), τ is the torque applied at the second link, and the matrices are given by

$$H(q) = \begin{bmatrix} m_1 l_1^2 + m_2 l^2 + m_2 l_2^2 + 2m_2 l_2 l \cos(q_2) & m_2 l_2^2 + m_2 l_2 l \cos(q_2) \\ m_2 l_2^2 + m_2 l_2 l \cos(q_2) & m_2 l_2^2 \end{bmatrix}, \tag{S25}$$

$$C(q,\dot{q}) = \begin{bmatrix} -2m_2 l_2 l \sin(q_2)\dot{q}_2 & -m_2 l_2 l \sin(q_2)\dot{q}_2 \\ m_2 l_2 l \sin(q_2)\dot{q}_1 & 0 \end{bmatrix}, \tag{S26}$$

$$G(q) = \begin{bmatrix} (m_1 l_1 + m_2 l) g_0 \cos(q_1) + m_2 l_2 g_0 \cos(q_1 + q_2) \\ m_2 l_2 g_0 \cos(q_1 + q_2) \end{bmatrix}, \tag{S27}$$

with g_0 the gravity constant and all other parameters as given in Figure S5.

Next, the local controllability of the acrobot about the equilibrium configuration of an upright pendulum body and a downward pointing module is analyzed for different values of the link location l. For this purpose, the system (S24) is linearized about the equilibrium given by $\bar{q} = (\pi/2, \pi)$ and $\bar{\tau} = 0$. The obtained state-space representation with the state $x = (q, \dot{q}) - (\bar{q}, 0)$ reads

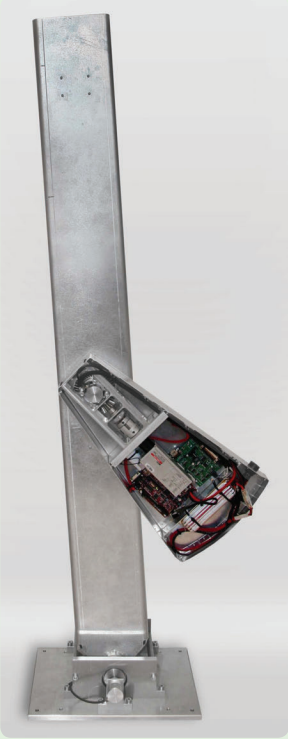


FIGURE S4 Inverted pendulum. The pendulum is a one-dimensional abstraction of the balancing cube: one module balances the pendulum body about one rotational degree of freedom. This system was built as a prototype prior to the cube.

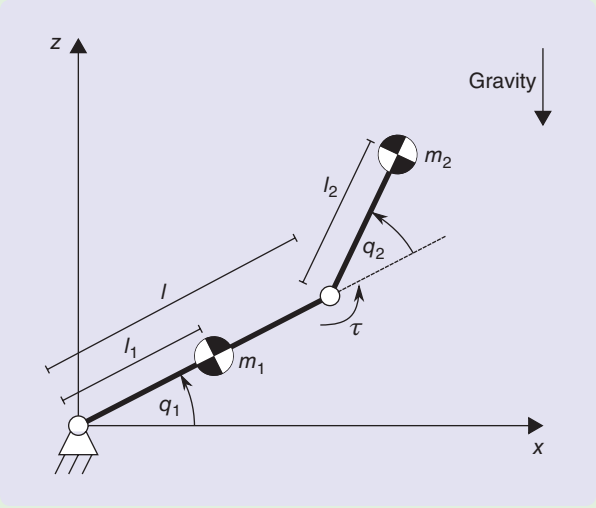


FIGURE S5 Acrobot. The acrobot is an abstraction of the pendulum in Figure S4. The module is represented by mass m_2, which is linked through a revolute joint to mass m_1 representing the pendulum body, which itself is linked to the ground. The acrobot is a typical example of an underactuated system: it has two degrees of freedom (described by the generalized coordinates $q = (q_1, q_2)$) with only one actuator (torque τ applied at the link between m_1 and m_2). The controllability of the system about the equilibrium given by $\bar{q} = (\pi/2, \pi)$ and $\bar{\tau} = 0$ is studied in this section for different location l of the module link on the pendulum.

$$\dot{x} = \begin{bmatrix} 0 & I \\ -H^{-1}(\bar{q})\dfrac{\partial G}{\partial q}(\bar{q}) & -H^{-1}(\bar{q})C(\bar{q},0) \end{bmatrix} x + \begin{bmatrix} 0 \\ H^{-1}(\bar{q}) \begin{bmatrix} 0 \\ \tau \end{bmatrix} \end{bmatrix}$$

$$= \underbrace{\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \dfrac{g_0}{l_1} & -\dfrac{g_0 l m_2}{m_1 l_1^2} & 0 & 0 \\ \dfrac{g_0(l-l_1-l_2)}{l_1 l_2} & -\dfrac{g_0(m_1 l_1^2 + m_2 l(l-l_2))}{m_1 l_1^2 l_2} & 0 & 0 \end{bmatrix}}_{=:\bar{A}} x$$

$$+ \underbrace{\begin{bmatrix} 0 \\ 0 \\ \dfrac{l-l_2}{m_1 l_1^2 l_2} \\ \dfrac{m_1 l_1^2 + m_2 (l-l_2)^2}{m_1 m_2 l_1^2 l_2^2} \end{bmatrix}}_{=:B} \tau. \qquad \text{(S28)}$$

For varying l and fixed parameter values $g_0 = 9.81\ \text{m/s}^2$, $l_1 = 0.66\ \text{m}$, $l_2 = 0.2\ \text{m}$, $m_1 = 5.3\ \text{kg}$, $m_2 = 3.7\ \text{kg}$, which are representative for the pendulum in Figure S4, the two smallest singular values of the controllability matrix $C = [\bar{B}, \bar{A}\bar{B}, \bar{A}^2\bar{B}, \bar{A}^3\bar{B}]$ are shown in Figure S6. It can be seen that the system becomes uncontrollable for $l = 0.66\ \text{m} + 0.2\ \text{m} = 0.86\ \text{m}$. More generally, it can be shown from (S28) that for $l = l_1 + l_2$ and arbitrary $l_1, l_2, m_1,$ *and* m_2, the rank of C drops from four to two. That is, for the configuration where the center of gravity (CG) of the module is at the same height as the CG of the pendulum, the system becomes locally uncontrollable.

How can this insight be interpreted for the cube? It can be expected that the location of a module relative to the CG of the cube body is a crucial factor in how effectively the module can act on it. The top modules' CG is above the CG of the cube body, whereas the bottom modules' CG is well below (especially since the lower modules are heavier and their CG is lower). In analogy to the 1-D analysis, it can therefore be expected that the top modules are less effective than the bottom ones. This is confirmed by the analysis of the singular values of the controllability matrix of the cube shown in Figure S7.

In conclusion, the top modules are used less, since they are less effective for control. The controller resulting from the LQR design seeks to minimize the total control effort and hence tends to use mainly the more effective bottom modules (equal weighting provided).

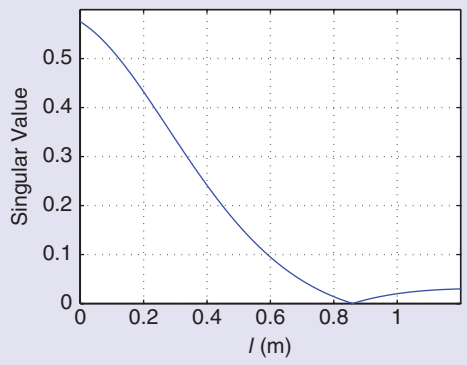


FIGURE S6 Controllability of the acrobot about the equilibrium $\bar{q} = (\pi/2, \pi)$. Shown are the two smallest (identical) singular values of the controllability matrix of the linearized model as a function of the link location l. For $l = l_1 + l_2 = 0.86$ m the system is uncontrollable. The pendulum in Figure S4 with $l = 0.69$ m is controllable.

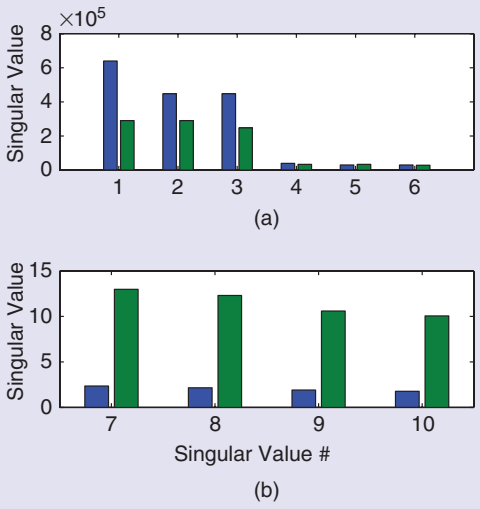


FIGURE S7 Singular values of the controllability matrix for corner balancing. The controllability of the cube is analyzed for the case that *only the top modules* (blue) and the case that *only the bottom modules* (green) are used. The controllability matrix C is computed from the model (5) after removing the states and inputs corresponding to the unused modules. The first six singular values (a) correspond mainly to motion of the modules, whereas the last four (b) correspond mainly to cube states. The fact that the last four singular values are smaller than the first six means that the cube states are "harder" to control than the module states (which is expected since the cube's degrees of freedom are not directly actuated). Furthermore, stabilizing the cube is harder when only the top modules are used compared to using only the bottom ones. This is indicated by the corresponding singular values being almost one order of magnitude smaller. Since the top modules are less effective in influencing the cube motion, they are used less by the optimal LQR controller for all six modules, which is designed in the section "Control."

REFERENCES

[S4] M. W. Spong, "The swing up control problem for the acrobot," *IEEE Control Syst. Mag.*, vol. 15, no. 1, pp. 49–55, Feb. 1995.

How Steady Can the Cube Balance?

The cube is not perfectly steady when it balances (see the section "Experiments"). Since the control system relies on sensory feedback for stabilization, sensor noise (mainly from the IMUs) inevitably excites the closed-loop system. In addition, unmodeled effects such as backlash or network-induced delays not taken into account in the control design can further deteriorate the balancing performance.

The question of how steady the cube can balance in principle with the current sensors is analyzed by employing tools from optimal $\mathcal{H}_2$ controller design. The $\mathcal{H}_2$ system norm is a measure of the overall gain of a dynamic system driven by noise: it equals the root mean square (RMS) value of the system output when the input is white noise of unit intensity [S5]. Hence, by appropriately scaling the input and output channels, the $\mathcal{H}_2$ norm is used to analyze the response of the cube (measured as the RMS of its state) to sensor noise excitation. It corresponds to the achievable balancing performance if no other nonidealities in the control system are present. For an introduction to $\mathcal{H}_2$ optimal controller design, the interested reader is referred to [S5].

For the $\mathcal{H}_2$ analysis and synthesis below, the discrete-time model (36) is assumed with noisy state measurements; that is,

$$x[k+1] = \tilde{A}x[k] + \tilde{B}v[k] \tag{S29}$$

$$\tilde{y}[k] = x[k] + w_1[k], \tag{S30}$$

where the artificial measurement noise $w_1[k]$ is assumed to have zero mean and variance R_n. The variance R_n is chosen below to match the variance of the state estimates resulting from the state estimation algorithms employed on the cube. Hence, the state estimation problem is abstracted away for the purpose of this analysis.

The $\mathcal{H}_2$ norm is computed for the closed-loop system given by (S29), (S30), and, first, the actual balancing controller K from (40), (41), and, second, an $\mathcal{H}_2$ optimal controller. The analysis allows one to estimate how well the control system would perform under ideal circumstances (that is, if the linear model captured the dynamics perfectly). It also allows one to determine how much could be gained from a more sophisticated controller design.

$\mathcal{H}_2$ ANALYSIS OF BALANCING CONTROLLER

The generalized plant shown in Figure S8 combines the cube model (36) with exogenous weighted inputs and outputs that are used to express the analysis objective. The $\mathcal{H}_2$ norm $\|G_{w_1 \to z_1}\|_2$ from measurement noise input $w_1[k]$ to the cube states $z_1[k] = C_z x[k]$ is a measure of the balancing performance (C_z is chosen to select a subset of the state vector $x[k]$). The norm corresponds to the RMS of the output $z_1[k]$ under noise excitation representative of the sensor noise on the balancing cube. The additional green blocks in Figure S8 are required for the $\mathcal{H}_2$ optimal controller design presented later.

To have a noise excitation representative of the actual cube hardware, the measurement noise variance R_n is chosen to be equal to the variance of the state estimates presented in the section "State Estimation." Hence, the noise variance on the cube states is set to be

$$R^{c}_{n,\text{cube}} = \text{diag}\left(\text{Var}\left[\hat{\beta}[k]\right],\ \text{Var}\left[\hat{\dot{\beta}}[k]\right],\ \text{Var}[\hat{\gamma}[k]],\ \text{Var}\left[\hat{\dot{\gamma}}[k]\right]\right) \tag{S31}$$

for corner balancing, and

$$R^{e}_{n,\text{cube}} = \text{diag}\left(\text{Var}\left[\hat{\beta}[k]\right],\ \text{Var}\left[\hat{\dot{\beta}}[k]\right]\right) \tag{S32}$$

for edge balancing, where $\hat{\beta}[k]$, $\hat{\dot{\beta}}[k]$, $\hat{\gamma}[k]$, and $\hat{\dot{\gamma}}[k]$ are the cube state estimates in (33)–(35). Given the noise variance of

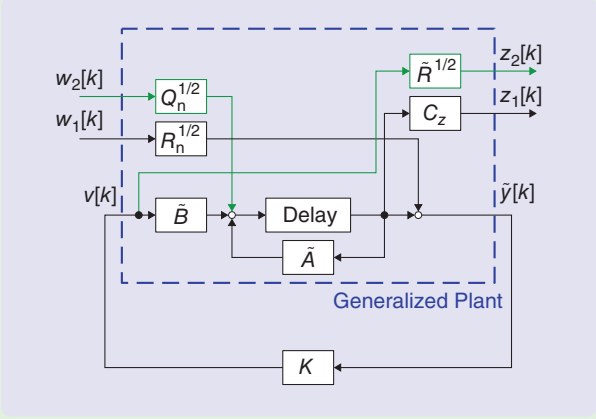


FIGURE S8 Generalized plant used to analyze the cube's balancing performance. The blocks $\tilde{A}$ and $\tilde{B}$ represent the cube model (36). The output $\tilde{y}[k]$ feeds to the controller K, which computes the system input $v[k]$. The exogenous input $w_1[k]$ and exogenous output $z_1[k]$ are used to express the design objective: input $w_1[k]$ is scaled with $R_n^{1/2}$ (the square root of the measurement noise variance), and $z_1[k]$ are the system states of interest, which are selected from $x[k]$ by means of the output matrix C_z. Hence, the $\mathcal{H}_2$ norm from input $w_1[k]$ to output $z_1[k]$, $\|G_{w_1 \to z_1}\|_2$, is the RMS of the selected state signals when the system is excited by noise of variance R_n. It is therefore a measure of the cube's balancing performance. The additional green signals and blocks (input $w_2[k]$ with weighting $Q_n^{1/2}$ and output $z_2[k]$ with weighting $\tilde{R}^{1/2}$) are required for the $\mathcal{H}_2$ controller synthesis problem to be well defined.

challenges of an unstable nonlinear system, a distributed control system, and a networked control system.

The concept of balancing a rigid body with multiple modules can also be used to balance other 3-D shapes. The modeling techniques as well as the developed state estimation and control algorithms are generalized from the concrete representation of the cube, and can be applied to other shapes with slight modification. See

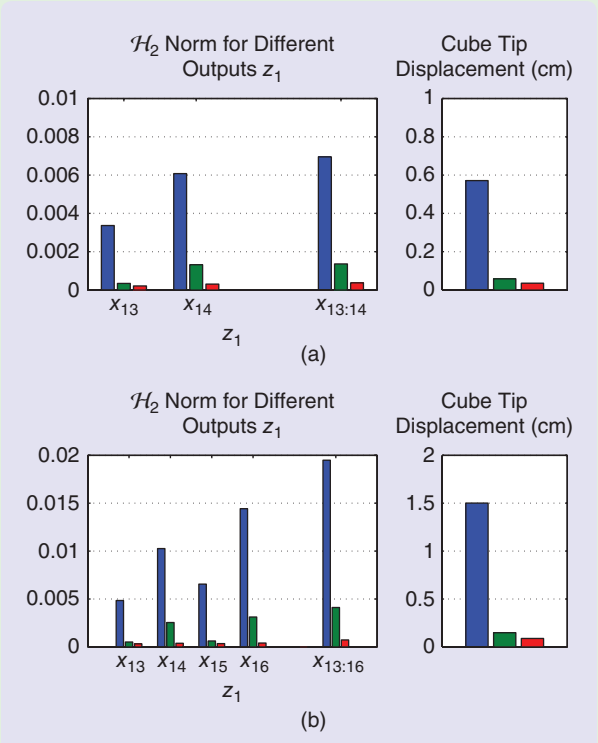


FIGURE S9 $\mathcal{H}_2$ analysis of balancing performance on (a) edge and (b) corner. The bar diagram on the left shows the $\mathcal{H}_2$ norm from state noise input w_1 to the cube states x_{13} through x_n, both individually and combined. These norms are obtained from experimental data (blue), from the linear model with the actual balancing controller (green), and with the $\mathcal{H}_2$ optimal controller (red). The $\mathcal{H}_2$ norm corresponds to the RMS value of the output signals under noise excitation of the system. The experimental data is the same as in Table 5. The diagram on the right shows the RMS displacement of the cube tip that is equivalent to the RMS of the cube tilt angles x_{13} and x_{15} (it is computed analogously to (S2), (S3) in "What Is the Cube's Maximal Balancing Range?").

the IMU sensors, first-order approximations of the variances in (S31) and (S32) can readily be computed from the involved estimator equations. Since the quantization error on the module encoders can be neglected, the noise variances for the modules states are chosen to be substantially lower than those for the cube states; hence, the overall noise variance is set to

$$R_n = \begin{bmatrix} 10^{-6}\bar{\sigma}(R_{n,cube})I & 0 \\ 0 & R_{n,cube} \end{bmatrix}, \qquad \text{(S33)}$$

where $\bar{\sigma}(\cdot)$ denotes the largest singular value.

With the weighting matrix (S33) and output matrix C_z chosen to select the cube state(s) of interest, the $\mathcal{H}_2$ norm of the generalized plant in Figure S8 from $w_1[k]$ to $z_1[k]$ can be evaluated for the balancing controller (40), (41). The results are shown in Figure S9 in green. The comparison to the experimental data (blue) reveals that the practically achieved balancing performance with the current LQR controller is lower than the theoretically achievable performance.

$\mathcal{H}_2$ OPTIMAL DESIGN

To design an $\mathcal{H}_2$ optimal controller for the generalized plant in Figure S8, some augmentations are necessary to make the $\mathcal{H}_2$ synthesis problem well defined. For this purpose, the generalized plant is augmented with the process noise input $w_2[k]$ (weight Q_n) and weighted control signal $z_2[k]$ (weight $\tilde{R}$). The process noise intensity is chosen to be considerably smaller than the measurement noise, to have a negligible effect on the controller design, $Q_n = 10^{-6}\bar{\sigma}(R_n)I$. For simplicity and to allow for a fair comparison in terms of control effort, diagonal weights $\tilde{R} = \tilde{\rho}I$ with parameter $\tilde{\rho}$ are used for the control input, where $\tilde{\rho} = 10^{-7}$ has been tuned such that the $\mathcal{H}_2$ gain from input $(w_1[k], w_2[k])$ to the output $z_2[k]$ is comparable to the respective gain when using the actual controller (40), (41). The output matrix C_z is chosen as $C_z = \text{diag}(0.01I_{12\times 12}, I)$ to express the primary design objective of minimizing the cube state variance.

The $\mathcal{H}_2$ optimal controller that minimizes the $\mathcal{H}_2$ norm from exogenous input $w[k] = (w_1[k], w_2[k])$ to exogenous output $z[k] = (z_1[k], z_2[k])$ can be obtained using standard $\mathcal{H}_2$ synthesis tools (for example, the Matlab implementation h2syn). The resulting $\mathcal{H}_2$ gains are shown in Figure S9 (red). The resulting smaller gains compared to the LQR design (green) are at the expense of a higher order controller. The $\mathcal{H}_2$ design presented herein is mainly of interest as a theoretical bound on the balancing performance. It does not take into account other design objectives that are of practical importance, such as steady-state behavior or actuator limitations.

In conclusion, the results of this section point to a potential improvement in the balancing performance of the cube. This may partly be achieved by taking currently unmodeled effects (such as gear backlash or communication delays) into account in the controller design.

REFERENCES

[S5] S. Skogestad and I. Postlethwaite, *Multivariable Feedback Control: Analysis and Design*, 2nd ed. New York: Wiley, Nov. 2005.

"Other Balancing Shapes" for a discussion and some conceptual ideas.

Problems addressed during the design and construction phase of the project have triggered unanticipated research results that are applicable beyond the cube itself. The "Time-Scale Separation Algorithm," first presented in [51], is an example. Without knowing the details of the feedback controllers, and yet taking their effects into account, this algorithm yields

This article focuses on the concepts and tools that were used to build and control the cube.

Other Balancing Shapes

Because the modules are self-contained, with onboard sensing, actuation, computation, and communication, they can be arranged into other "balancing" shapes. Conceptual drawings of such shapes are depicted in Figure S10.

The tilt estimation method presented in the section "State Estimation" requires at least four triaxis accelerometers. If fewer than four modules are used (such as for the wedge in Figure S10), additional sensors may be placed on the balanced body. Alternatively, a model-based state estimation technique (such as a Kalman filter) can be used to observe the tilt state from rate gyro sensors only. As far as the modeling process and the design of the control algorithms are concerned, the same strategy as presented in this article for the cube may be used for other shapes.

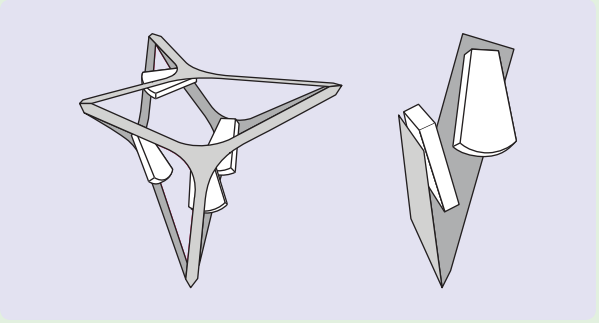


FIGURE S10 Other balancing shapes. Instead of balancing a cube, the modules could be used to balance a tetrahedron or a wedge (standing on a tip), for example.

a simplified discrete-time model of a continuous-time process under high-gain feedback on some of its states. A limiting property of the matrix exponential was also derived in the process. Another example is the algorithm for estimating the tilt of the cube (presented both in this article and in [49]). The algorithm can be used to estimate the tilt of any rigid body with only rotational degrees of freedom from measurements of multiple inertial sensors without requiring a dynamic system model.

For the control and estimation algorithms presented herein, full communication between the agents is assumed: each agent shares its sensory data with all its peers at the closed-loop rate. The state estimation and control design problems can hence be treated in a centralized fashion. In [40], [41], the problem of reduced communication state estimation is addressed. Therein, event-based communication protocols are used to reduce the amount of sensor data shared over the network while maintaining a certain estimation performance.

ACKNOWLEDGMENTS

The authors' special thanks goes to Matthew Donovan for his contributions to the concept, the mechanical design, and the realization of the balancing cube. Furthermore, the authors thank their colleagues at ETH Zurich—Daniel Burch, Sergei Lupashin, Hans Ulrich Honegger, and Gajamohan Mohanarajah—as well as all students who have participated in the project for making the balancing cube happen. Carolina Flores is gratefully acknowledged for drawings and photos of the cube in this article. The authors would finally like to thank Philipp Reist and Raymond Oung for their valuable comments and suggestions.

This work was supported by the Swiss National Science Foundation (SNSF).

AUTHOR INFORMATION

Sebastian Trimpe (strimpe@ethz.ch) received a B.Sc. degree in general engineering science in 2005 and an M.S. degree (Dipl.-Ing.) in electrical engineering in 2007 from the Hamburg University of Technology. He is currently a Ph.D. candidate at the Institute for Dynamic Systems and Control at ETH Zurich. In 2007, he spent eight months as a research scholar at the University of California at Berkeley. He is recipient of the General Engineering Award for the best undergraduate degree (2005), a scholarship from the German National Academic Foundation (2002–2007), and the triennial IFAC World Congress Interactive Paper Prize (2011). He can be contacted at ETH Zurich, Sonneggstr. 3, ML K33, 8092 Zurich, Switzerland.

Raffaello D'Andrea received a B.Sc. degree in engineering science from the University of Toronto in 1991 and M.S. and Ph.D. degrees in electrical engineering from the California Institute of Technology in 1992 and 1997, respectively. He was an assistant, and then an associate, professor at Cornell University from 1997 to 2007. While on leave from Cornell, from 2003 to 2007, he cofounded Kiva Systems, where he led the systems architecture, robot design, robot navigation and coordination, and control algorithms efforts. A creator of dynamic sculpture, his work has appeared at various international venues, including the National Gallery of Canada, the Venice Biennale, Ars Electronica, the Smithsonian, the FRAC Centre, and the Spoleto Festival. He is currently a professor of dynamic systems and control at ETH Zurich and chief technical advisor at Kiva Systems.

REFERENCES

[1] Balancing Cube Web site. (2012, Aug. 13). [Online]. Available: http://www.cube.ethz.ch

[2] J. Shen, A. K. Sanyal, N. A. Chaturvedi, D. S. Bernstein, and N. H. McClamroch, "Dynamics and control of a 3-D pendulum," in *Proc. 43rd IEEE Conf. Decision Control*, Paradise Island, Bahamas, Dec. 2004, pp. 323–328.

[3] P. Horáček, "Laboratory experiments for control theory courses: A survey," *Ann. Rev. Control*, vol. 24, pp. 151–162, Jan. 2000.

[4] K. Åström and K. Furuta, "Swinging up a pendulum by energy control," *Automatica*, vol. 36, no. 2, pp. 287–295, Feb. 2000.

[5] L. Buşoniu, D. Ernst, B. De Schutter, and R. Babuška, "Online least-squares policy iteration for reinforcement learning control," in *Proc. American Control Conf.*, Baltimore, MD, Jul. 2010, pp. 486–491.

[6] A. P. Schoellig and R. D'Andrea, "Optimization-based iterative learning control for trajectory tracking," in *Proc. European Control Conf.*, Budapest, Hungary, Aug. 2009, pp. 1505–1510.

[7] S. Jung and S. S. Kim, "Control experiment of a wheel-driven mobile inverted pendulum using neural network," *IEEE Trans. Control Syst. Technol.*, vol. 16, no. 2, pp. 297–303, Mar. 2008.

[8] R. Findeisen and P. Varutti, "Stabilizing nonlinear predictive control over nondeterministic communication networks," in *Nonlinear Model Predictive Control* (Lecture Notes in Control and Information Sciences), vol. 384, L. Magni, D. Raimondo, and F. Allgöwer, Eds. Berlin, Germany: Springer-Verlag, 2009, pp. 167–179.

[9] H. Gao, X. Meng, and T. Chen, "Stabilization of networked control systems with a new delay characterization," *IEEE Trans. Autom. Control*, vol. 53, no. 9, pp. 2142–2148, Oct. 2008.

[10] J. Colandairaj, G. W. Irwin, and W. G. Scanlon, "Wireless networked control systems with QoS-based sampling," *IET Control Theory Appl.*, vol. 1, no. 1, pp. 430–438, Jan. 2007.

[11] L. Zhang, Y. Shi, T. Chen, and B. Huang, "A new method for stabilization of networked control systems with random delays," *IEEE Trans. Autom. Control*, vol. 50, no. 8, pp. 1177–1181, Aug. 2005.

[12] X. Liu and A. Goldsmith, "Wireless network design for distributed control," in *Proc. 43rd IEEE Conf. Decision Control*, Paradise Island, Bahamas, Dec. 2004, pp. 2823–2829.

[13] D. V. Efimov and A. L. Fradkov, "Robust and adaptive observer-based partial stabilization for a class of nonlinear systems," *IEEE Trans. Autom. Control*, vol. 54, no. 7, pp. 1591–1595, Jul. 2009.

[14] R.-J. Wai, M.-A. Kuo, and J.-D. Lee, "Design of cascade adaptive fuzzy sliding-mode control for nonlinear two-axis inverted-pendulum servomechanism," *IEEE Trans. Fuzzy Syst.*, vol. 16, no. 5, pp. 1232–1244, Oct. 2008.

[15] R.-J. Wai and L.-J. Chang, "Adaptive stabilizing and tracking control for a nonlinear inverted-pendulum system via sliding-mode technique," *IEEE Trans. Ind. Electron.*, vol. 53, no. 2, pp. 674–692, Apr. 2006.

[16] A. Mills, A. Wills, and B. Ninness, "Nonlinear model predictive control of an inverted pendulum," in *Proc. American Control Conf.*, St. Louis, MO, June 2009, pp. 2335–2340.

[17] S. Jung and J. T. Wen, "Nonlinear model predictive control for the swing-up of a rotary inverted pendulum," *J. Dynamic Syst., Meas., Control*, vol. 126, no. 3, pp. 666–673, Sep. 2004.

[18] C. R. Magers and A. N. Gündeş, "Low order decentralized stabilizing controller design for a mobile inverted pendulum robot," in *Proc. American Control Conf.*, St. Louis, MO, June 2009, pp. 4233–4234.

[19] W. Chen and J. Li, "Decentralized output-feedback neural control for systems with unknown interconnections," *IEEE Trans. Syst., Man, Cybern. B*, vol. 38, no. 1, pp. 258–266, Feb. 2008.

[20] G. Liu, I. Mareels, and D. Nešić, "Decentralized control design of interconnected chains of integrators: A case study," *Automatica*, vol. 44, no. 8, pp. 2171–2178, 2008.

[21] S. Cho, J. Shen, and N. McClamroch, "Mathematical models for the triaxial attitude control testbed," *Math. Comput. Model. Dynamical Syst.*, vol. 9, no. 2, pp. 165–192, 2003.

[22] N. A. Chaturvedi, N. H. McClamroch, and D. S. Bernstein, "Stabilization of a 3D axially symmetric pendulum," *Automatica*, vol. 44, no. 9, pp. 2258–2265, 2008.

[23] N. A. Chaturvedi, N. H. McClamroch, and D. S. Bernstein, "Asymptotic smooth stabilization of the inverted 3-D pendulum," *IEEE Trans. Autom. Control*, vol. 54, no. 6, pp. 1204–1215, June 2009.

[24] N. Chaturvedi and H. McClamroch, "Asymptotic stabilization of the inverted equilibrium manifold of the 3-D pendulum using non-smooth feedback," *IEEE Trans. Autom. Control*, vol. 54, no. 11, pp. 2658–2662, Nov. 2009.

[25] B. Srinivasan, P. Huguenin, and D. Bonvin, "Global stabilization of an inverted pendulum – control strategy and experimental verification," *Automatica*, vol. 45, no. 1, pp. 265–269, 2009.

[26] S. Riachy, Y. Orlov, T. Floquet, R. Santiesteban, and J.-P. Richard, "Second-order sliding mode control of underactuated mechanical systems I: Local stabilization with application to an inverted pendulum," *Int. J. Robust Nonlinear Control*, vol. 18, no. 4–5, pp. 529–543, 2008.

[27] K. Pathak, J. Franch, and S. K. Agrawal, "Velocity and position control of a wheeled inverted pendulum by partial feedback linearization," *IEEE Trans. Robot.*, vol. 21, no. 3, pp. 505–513, Jun. 2005.

[28] P. Reist and R. Tedrake, "Simulation-based LQR-trees with input and state constraints," in *Proc. IEEE Int. Conf. Robotics Automation*, Anchorage, AK, May 2010, pp. 5504–5510.

[29] J. P. Hespanha, P. Naghshtabrizi, and Y. Xu, "A survey of recent results in networked control systems," *Proc. IEEE*, vol. 95, no. 1, pp. 138–162, Jan. 2007.

[30] K. Gilpin and D. Rus, "Modular robot systems," *IEEE Robot. Autom. Mag.*, vol. 17, no. 3, pp. 38–55, Sep. 2010.

[31] R. Oung and R. D'Andrea, "The distributed flight array," *Mechatronics*, vol. 21, no. 6, pp. 908–917, Sep. 2011.

[32] J. K. Yook, D. M. Tilbury, and N. R. Soparkar, "Trading computation for bandwidth: Reducing communication in distributed control systems using state estimators," *IEEE Trans. Control Syst. Technol.*, vol. 10, no. 4, pp. 503–518, July 2002.

[33] J. E. Luntz and W. Messner, "A distributed control system for flexible materials handling," *IEEE Control Syst. Mag.*, vol. 17, no. 1, pp. 22–28, Feb. 1997.

[34] J. M. Fowler and R. D'Andrea, "A formation flight experiment," *IEEE Control Syst. Mag.*, vol. 23, no. 5, pp. 35–43, Oct. 2003.

[35] A. Stubbs, V. Vladimerou, A. T. Fulford, D. King, J. Strick, and G. E. Dullerud, "Multivehicle systems control over networks: A hovercraft testbed for networked and decentralized control," *IEEE Control Syst. Mag.*, vol. 26, no. 3, pp. 56–69, June 2006.

[36] Z. Jin, S. Waydo, E. B. Wildanger, M. Lammers, H. Scholze, P. Foley, D. Held, and R. M. Murray, "MVWT-II: The second generation Caltech multivehicle wireless testbed," in *Proc. American Control Conf.*, Boston, MA, July 2004, pp. 5321–5326.

[37] N. Michael, J. Fink, and V. Kumar, "Experimental testbed for large multirobot teams," *IEEE Robot. Autom. Mag.*, vol. 15, no. 1, pp. 53–61, Mar. 2008.

[38] M. W. Spong, "Underactuated mechanical systems," in *Control Problems in Robotics and Automation* (Lecture Notes in Control and Information Sciences), vol. 230, B. Siciliano and K. Valavanis, Eds. Berlin, Germany: Springer-Verlag, 1998, pp. 135–150.

[39] Quanser Inc. Cube. (2012, Aug. 13). [Online]. Available: http://www.quanser.com

[40] S. Trimpe and R. D'Andrea, "An experimental demonstration of a distributed and event-based state estimation algorithm," in *Proc. 18th IFAC World Congr.*, Milan, Italy, Aug. 2011, pp. 8811–8818.

[41] S. Trimpe and R. D'Andrea, "Reduced communication state estimation for control of an unstable networked control system," in *Proc. 50th IEEE Conf. Decision Control European Control Conf.*, Orlando, FL, 2011, pp. 2361–2368.

[42] J. J. Craig, *Introduction to Robotics: Mechanics and Control*, 3rd ed. Englewood Cliffs, NJ: Prentice-Hall, 2005.

[43] MathWorks Inc. (2009, Sep.). *SimMechanics™ User's Guide* [Online]. Available: http://www.mathworks.com

[44] S. Trimpe and R. D'Andrea, "Numerical models and controller design parameters for the balancing cube," Institute for Dynamic Systems and Control, ETH Zürich, Zürich, Switzerland, Tech. Rep., 2012, DOI: 10.3929/ethz-a-007343301.

[45] D. H. Titterton and J. L. Weston, *Strapdown Inertial Navigation Technology*, 2nd ed. Stevenage, U.K.: IET, 2004.

[46] J. Farrell and M. Barth, *The Global Positioning System and Inertial Navigation*. New York: McGraw-Hill, 1999.

[47] D. Luenberger, "An introduction to observers," *IEEE Trans. Autom. Control*, vol. 16, no. 6, pp. 596–602, Dec. 1971.

[48] B. D. O. Anderson and J. B. Moore, *Optimal Filtering*. New York: Dover, 2005.

[49] S. Trimpe and R. D'Andrea, "Accelerometer-based tilt estimation of a rigid body with only rotational degrees of freedom," in *Proc. IEEE Int. Conf. Robotics Automation*, Anchorage, AK, May 2010, pp. 2630–2636.

[50] B. D. O. Anderson and J. B. Moore, *Optimal Control: Linear Quadratic Methods*. New York: Dover, 2007.

[51] S. Trimpe and R. D'Andrea, "A limiting property of the matrix exponential with application to multi-loop control," in *Proc. Joint 48th IEEE Conf. Decision Control Chinese Control Conf*, Shanghai, China, Dec. 2009, pp. 6419–6425.