Computational Multicopter Design

Tao Du
MIT CSAIL

Adriana Schulz
MIT CSAIL

Bo Zhu
MIT CSAIL

Bernd Bickel
IST Austria

Wojciech Matusik
MIT CSAIL

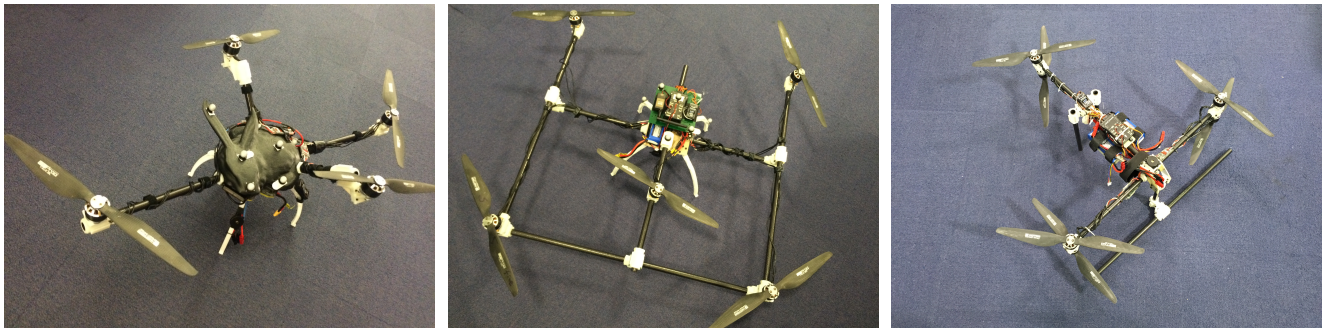

Figure 1: *We provide an interactive system for users to design, optimize and fabricate multicopters. We explore the design space to allow multicopter design with free-form geometry and nonstandard motor positions and directions. Based on user-specified metrics, our system optimizes the copter geometry and suggests a valid controller. Left: a multicopter example with free-form geometry, various motor heights and different propeller sizes. Middle: a pentacopter with optimized motor positions and orientations, allowing* 30% *increase in payload. Right: a classic hexacopter with three pairs of coaxial propellers.*

Abstract

We present an interactive system for computational design, optimization, and fabrication of multicopters. Our computational approach allows non-experts to design, explore, and evaluate a wide range of different multicopters. We provide users with an intuitive interface for assembling a multicopter from a collection of components (e.g., propellers, motors, and carbon fiber rods). Our algorithm interactively optimizes shape and controller parameters of the current design to ensure its proper operation. In addition, we allow incorporating a variety of other metrics (such as payload, battery usage, size, and cost) into the design process and exploring trade-offs between them. We show the efficacy of our method and system by designing, optimizing, fabricating, and operating multicopters with complex geometries and propeller configurations. We also demonstrate the ability of our optimization algorithm to improve the multicopter performance under different metrics.

Keywords: parametric modeling, optimization, fabrication

Concepts: •**Computing methodologies** → **Modeling and simulation;**

1 Introduction

Multicopters are aerial vehicles that are becoming more and more popular. They are mechanically simple and can be controlled manually or automatically to have a stable and accurate motion. For this reason, these vehicles are increasingly being used in many settings, including photography, disaster response, search and rescue operations, hazard mitigation, and geographical 3D mappings. Most current multicopter designs are fairly standard (e.g., symmetric quadcopters or hexacopters). Designing a non-standard multicopter that is optimized for a specific application is challenging since it requires expert knowledge in aerodynamics and control theory.

In this work we propose a design process that allows non-expert users to build custom multicopters that are optimized for specific design goals. Our system allows users to concentrate on high-level design while computation handles all the necessary elements to ensure the correct function of the resulting physical models. Our intuitive composition tool enables users to express their creativity and to explore trade-offs between different objectives in order to develop machines well suited for specific applications.

Our system also allows us to expand the design space of multicopters. Typical space encompasses a small set of standard designs with a symmetric distribution of rotors and all propellers oriented upright. In this design space, forces and torques induced by propellers are easily balanced and controllers can be easily adjusted. However, this does not allow the design of nonsymmetric multicopters that are more optimal for some specific tasks. For example, the field view of a camera in a hexacopter could be obstructed by the motors, so one might prefer removing a motor in front of the camera. Another example could be a multicopter carrying an irregularly shaped object like a wide-band antenna. In this case, an asymmetric design gives more freedom to specify mass distribution for better flying stability. Finally, a nonstandard design with extra motors can increase the payload and improve fault tolerance, which are key factors for product delivery. Our computational design employs parametric representations that capture variably in the general shape, rotor positions and orientations, and performs optimizations based on user-specified metrics. This enables users to explore the shape space and discover functional vehicles that significantly differ and outperform the standard models. For example, the pentacopter shown in Figure 8 has a performance increase of about 30% by a non-trivial variation of rotor position and orientation which would be difficult to be modeled even by a skilled engineer.

An immediate challenge after we expand the design space is to find a good controller for non-standard multicopter designs. Due to its uneven distribution of mass and inertia tensor as well as its random rotor position and orientation, applying a traditional controller from classic multicopters directly requires nontrivial and tedious parameter adjustments. Moreover, during flight, the performance of a multicopter is jointly influenced by its dynamics and control signals and therefore optimization has to include both shape variables and controller parameters. Finally, there is a tradeoff between allowing users to freely express metrics and still keeping the optimization problem tractable. In our system, we use Linear-Quadratic-Regulator (LQR), an optimal control method for nonstandard multicopter designs, and the controller is automatically determined to avoid tedious parameter adjustments. We formulate an optimization problem which includes both shape and control variables and propose an algorithm to effectively find the optimal shape as well as the

control parameters for a given design. We specify the user metrics as a bi-convex function, which is expressive enough to represent many useful metrics.

To summarize, our main contributions include:

- Providing a complete pipeline that allows users to design, optimize, and fabricate multicopters.
- Formulating an optimization problem that can jointly optimize the shape and control parameters according to different design metrics.
- Providing an efficient numerical scheme to solve the multicopter optimization problem and show its efficacy by optimizing various types of non-standard multicopter designs.

2 Related Work

Our work draws inspiration from dynamical system design and analysis, fabrication-oriented design, and multicopter design.

Dynamical System Optimization Optimizing dynamical systems has recently drawn great attention in computer graphics. These systems are governed by different physical models, ranging from walking robots [Wampler and Popović 2009; Coros et al. 2011], mechanical characters [Coros et al. 2013], swimming characters [Lentine et al. 2011; Tan et al. 2011], birds [Wu and Popović 2003; Ju et al. 2013], and gliders [Umetani et al. 2014]. Particularly, for the sake of efficiently generating artist-desired animations, a large variety of optimization algorithms have been developed in the effort to edit and control the dynamics of passive rigid bodies, including [Popović et al. 2000], [Twigg and James 2008], and [Jain and Liu 2009], to name a few. Two main categories of optimization problems were solved for these systems: optimizing the shape and the controller. Geometry optimization methods were mostly applied in passive dynamical systems, which receive control forces from the interaction between the body and the environment such as wind lift and buoyancy. Umetani et al. [2014] presented a system which allows the user to design the shape of free-form hand-ranching gliders that actually fly. Martin et al. [2015] presented a data-driven approach to capture parameters of omni-directional aerodynamics model, which is then used to design three-dimensional kites. Controller optimization methods were used in optimizing the locomotion and balance of self-propelling systems, such as bicycle riders [Tan et al. 2014] and swimming creatures [Lentine et al. 2011]. In this paper we are aiming at jointly optimizing both shape and control parameters for a dynamical system, which to our knowledge has not been explored in this literature.

Multicopter optimization has been studied recently in other engineering disciplines. Magnussen et al. [2014; 2015] presented a method for optimizing multicopter configurations to achieve longer flying time and faster motor response. While their design procedure assumed symmetric frames and their mixed-integer programming approach was limited to discrete motor/propeller selections, our method explores a larger design space that allows for nonstandard shape with continuous motor positions and orientations.

Fabrication-Oriented Design Design tools for fabrication have gained a lot of interest in the computer graphics community. Typically, these approaches optimize object properties to meet a specific functional goal, and often consist of a mix of interaction- and numerical optimization-based approaches. Recent work, for example, investigated the design of plush toys [Mori and Igarashi 2007], furniture [Saul et al. 2011; Umetani et al. 2012; Lau et al. 2011; Schulz et al. 2014], clothes [Umetani et al. 2011], inflatable structures [Skouras et al. 2014], wire meshes [Garg et al. 2014], mechanical objects [Koo et al. 2014; Zhu et al. 2012; Coros et al. 2013], and masonry structures [Whiting et al. 2012; Vouga et al. 2012].

While physical quantities such as center of mass and rotational inertia play a crucial role in our process (related work has addressed static [Prévost et al. 2013] and dynamic balancing [Bächer et al. 2014] of physical objects by internal hollowing and thereby redistributing its mass), our objective is significantly more complex. Probably works that have come close to our work are model airplanes [Umetani et al. 2014] and kites [Martin et al. 2015].

In the context of exploring stable motion, Bharaj et al. [2015] recently presented a method that leverages physical simulation and evolutionary optimization to refine the mechanical designs of automata such that the resulting toys are able to walk. However, while their design approach is an evolutionary, discrete offline optimization approach, ours is interactive and almost in real-time.

Probably the most important factor that distinguishes our work to the above mentioned work is that none of these contain any sensors, nor high-level controllers. To our knowledge, our work is the first to investigate the challenge of designing multicopters while taking into account the capabilities of the controller.

Multicopter Dynamics and Control Multicopters, especially quadcopters, have drawn attention in aerodynamics and robotics community for a long time due to its mechanical simplicity and ease of control. Quadcopter modeling and dynamics have been extensively studied [Hoffmann et al. 2007]. We extend quadcopter dynamics model and explore a larger design space where a multicopter can have free-form geometry and nonsymmetric motors.

Various control methods have been successfully applied in quadcopter, including PID [Tayebi and McGilvray 2004; Hoffmann et al. 2007], LQR [Bouabdallah et al. 2004; Hoffmann et al. 2004], nonlinear H_∞ controller [Raffo et al. 2010], sliding mode controller and reinforcement learning controller [Waslander et al. 2005]. However, in our work, controller design is still a new challenge as our multicopter dynamics is generally more complicated than a quadcopter, and directly applying existing control techniques usually requires nontrivial parameter tweaking. We simplify the controller design process by using LQR control and automatically solving its parameters from our optimization problem. To our knowledge our work is the first to provide a systematic way to select controller for nonstandard multicopters.

Recently, multicopters have been applied to computer graphics research problems. [Joubert et al. 2015; Roberts and Hanrahan 2016] optimize quadcopter trajectories for better camera shots. In their work an interactive design tool is provided for users to specify camera key frames, and a smooth camera trajectory is generated. Similarly, our work also provides an interactive tool for multicopter design, but we focus on optimizing the geometry and control for multiple metrics defined on the multicopter design itself.

3 System Overview

Figure 2 shows the overview of our system. Users start from designing a multicopter with our interactive design tool, which includes a database of standard parts and parameterized free-form body frames. The output of the design tool is a parametric multicopter with geometric constraints on shape parameters and is fed into our optimization algorithm, which optimizes both the shape and controller based on user-specified metrics and returns the results within seconds. Users can go back and adjust their design using the interactive tool or verify the optimization results in a real-time simulator. A real multicopter can be built based on the shape parameters and the controller suggested by our optimization. Finally, a real flight test is performed for evaluation.

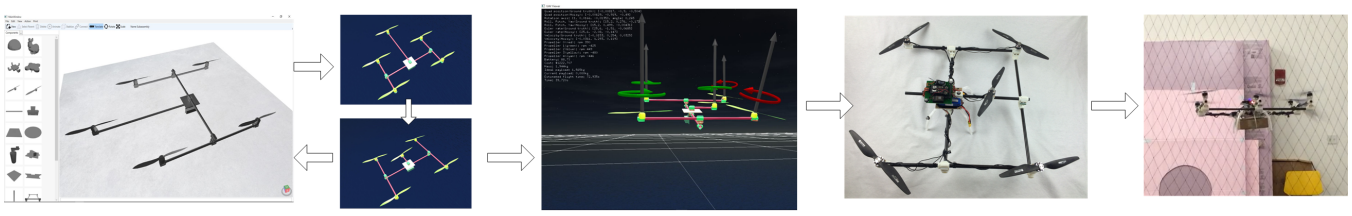

Figure 2: *An overview of our system. The pipeline consists of multicopter design (Section 4), optimization (Section 7), simulation (Section 8), fabrication (Section 9) and flight tests (Section 10).*

4 Geometry Design

Our interactive design tool allows users to compose new models from parts in a database, which contains standard parts like propellers, motors, carbon fiber rods, and a variety of free-form body frames. Though the database is relatively small, users can create a widely diverse set of designs by manipulating and composing them.

4.1 Part database

Following the ideas in [Schulz et al. 2014], both functional parts and free-form body frames in the database are *parametric*, represented by a feasible parameter set and a linear mapping function that returns different geometries for different parameter configurations. The use of parametric shapes allows geometric variations while guaranteeing that all shape manipulations preserve structure and manufacturability. In our model, all parameter configurations in the feasible set are guaranteed to be manufacturable geometries. Linear maps were chosen because they speed up computation without compromising too much on expressiveness. In this linear representation, geometries are represented as meshes where each vertex is a linear function of the parameters.

Each part in the database is annotated with connecting patches that indicate regions where parts can be attached to each other. Since every part is parametric, the position of each patch is also a function of the parameters. We define different patch types for different composition methods. For example, carbon tubes are annotated with circular patches while the bottom of a rotor has a flat patch (see Figure 3). Each patch type includes a parametric representation of its center and additional information for alignment (for example, circular patches include the radius and main axis and flat patches include the normal). Our collection was designed and parametrized by mechanical engineers.

4.2 Composition

After the user drags in a new part and calls the *connecting* operation, the system aligns the parts with respect to the working model and adds the appropriate connecting components. The two closest patches are used to create a connection between the components.

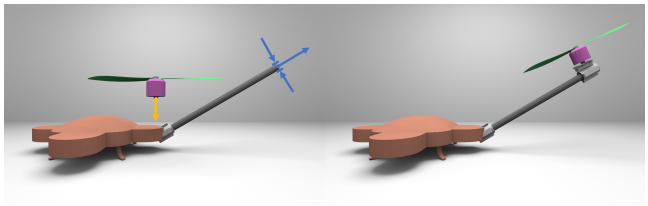

Figure 3: *Example of composition. Left: parts with highlighted patches (circular patch with an annotated main axis and a diameter in blue and flat patch with an annotated normal in orange). Right: composed design.*

We define a list of rules that designate appropriate connecting parts for each type of patch pair. Connecting parts are selected from a small list of standard components, e.g., patches of carbon fiber rods need circular adapters and plates need mounts with screws.

Next, we position parts and appropriate connectors relative to each other. Our goal is to preserve the parametric representation in the composed designs so that users can continue to manipulate the geometry at every step in a structure-preserving fashion. In the composed model the parameter set is the union of the parameters of the containing parts and the feasible set is the intersection of the feasible sets. Therefore, the alignment step must define constraints on part positions so that parts are correctly placed relative to each other in all feasible configurations. This is done in two steps: first the patch information is used to appropriately rotate the models relative to each other (in the example in Figure 3, the normal of the flat patch is made perpendicular to the main axis of the circular patch). Rotations are dealt with first because these are not linear operations and therefore cannot be represented by our linear parameter mapping. Since the position of each patch is represented as function of the parameters, alignment translations involve constraining these two functions to be equal. Therefore parts are aligned by adding constraints to the feasible set of the composed parametric design and solving for the closest feasible solution.

The parameters of the composed design are not only useful for allowing users to better explore the design space, but they also describe the possible variations for automatic design optimization. Since both parts and composition schemes are defined using linear models, linearity is preserved in the resulting composed designs, which include a set of equality and inequality constraints. Equality constraints can be removed by replacing the original shape parameters with free variables in the affine space defined by the linear equations. As a result, we will use $\mathbf{s}$ to denote the new shape parameters and $\mathbf{A}^{\text{ineq}}\mathbf{s} \leq \mathbf{b}^{\text{ineq}}$ to represent these constraints in later sections. The linear parameter variations allow parts of the copters to scale and move relative to each other but do not define local rotations. We therefore augment the feasible set by defining additional variables, $\mathbf{d}$, that represent the orientation of the propellers. These are used in the optimization algorithm, as will be discussed next.

5 Multicopter Dynamics

In this section we provide the background information about multicopter dynamics. We first introduce the motor and propeller model, then multicopter dynamics is provided based on Newton's law and Euler's equations. Table 1 lists all the variables in this paper.

5.1 Motor and propeller

Our motor and propeller model is based on measuring multiple properties (such as thrust, torque, voltage, and current) then fitting them with analytical functions. All measurement data and fitting results are provided in supplemental materials.

Symbol	Domain	Definition
n	N^+	Number of motors.
u	R^+	Propeller thrust.
τ	R^+	Propeller torque.
λ	R^+	Propeller torque-thrust ratio.
P_{ele}	R^+	Electric power consumed by motor.
I	R^+	Current supplied to a motor.
$\mathbf{p}$	R^3	Center of gravity in world frame.
$\mathbf{R}$	SO(3)	Body-to-world rotation matrix.
ω	R^3	Angular velocity in body frame.
$\mathbf{e}$	R^3	Euler angles: roll, pitch and yaw.
m	R^+	Mass.
$\mathbf{J}$	$R^{3\times3}$	Inertia tensor in body frame.
$\mathbf{r}$	R^3	Motor position in body frame, linear on shape parameters $\mathbf{s}$.
$\mathbf{d}$	unit sphere	Motor orientation in body frame.
b	$\{-1,1\}$	Motor spin direction, -1 means counterclockwise and 1 clockwise.
$\mathbf{g}$	R^3	Gravitational acceleration in world frame.
$\mathbf{u}$	R^n	A column vector representing all propeller thrusts.
$\mathbf{M}_f$	$R^{3\times n}$	Mapping from thrusts to net force. The i-th column is $\mathbf{d}_i$.
$\mathbf{M}_t$	$R^{3\times n}$	Mapping from thrusts to net torque. The i-th column is $b_i\lambda_i\mathbf{d}_i + \mathbf{r}_i \times \mathbf{d}_i$.
$\mathbf{x}$	R^{12}	Copter state. $\mathbf{x} = [\mathbf{p}, \mathbf{e}, \dot{\mathbf{p}}, \dot{\mathbf{e}}]$
$\mathbf{x}^*$	R^{12}	State part of the fixed point in the state-space model. Typically it contains arbitrary positions, fixed attitude and zero velocities.
$\mathbf{u}^*$	R^n	Thrust part of the fixed point in the state-space model, i.e., gravity is balanced and all torques are canceled out.
$\mathbf{s}$	R^k	Shape parameter satisfying $\mathbf{A}^{\text{ineq}}\mathbf{s} \le \mathbf{b}^{\text{bineq}}$

Table 1: *Variable definitions in copter dynamics and control. For a multicopter we use n to denote the number of motors, and k the dimension of its shape parameter. Subscript i refers to the variable corresponding to the i-th motor or propeller.*

Thrust and torque We use u and τ to denote the magnitude of thrust and torque developed by the propeller. At hover, the torque is known to be proportional to the thrust [Leishman 2006]:

$$\tau = \lambda u \tag{1}$$

where λ is a constant ratio determined by the blade geometry, and is acquired by fitting the thrust and torque measurement.

Motor control The motor spinning rate is controlled by sending desired Power-Width Modulation (PWM) signals to its Electric-Speed-Controller (ESC). PWM signals control the power supplied to the motor, and therefore its spinning rate, which further influences the thrust and torque induced by the motor. We measure the mapping between PWM values, battery voltage and thrust, then use its inverse mapping to convert the output thrust from the controller to PWM values sent to each motor.

Power consumption When a motor loaded with a propeller is powered on, the spinning motor converts electronic power P_{ele}, the product of voltage and current I, into mechanic power which rotates the propeller, which then pushes surrounding air to generate thrusts u. We directly measure u-P_{ele} and u-I curves, which can be well approximated by power functions, and use them to guide our optimization on flight time and max amperage.

5.2 Equations of motion

We use North-East-Down (NED) coordinates as our world frame to determine the position and orientation of the copter. Our body frame is fixed at the center of the copter, and initially its three axis are parallel to the axis of world frame. To use propellers efficiently, we require all propellers to thrust upwards, not downwards. This can be guaranteed by matching the propeller type and motor spin direction. The thrust and torque produced by the i-th motor are $u_i\mathbf{d}_i$ and $b_i\lambda_i u_i\mathbf{d}_i$, as explained in Table 1. From Newton's second law and Euler's equation the dynamics are:

$$\begin{aligned} m\ddot{\mathbf{p}} &= m\mathbf{g} + \mathbf{R}\sum_{i=1}^{n} u_i\mathbf{d}_i \\ \mathbf{J}\dot{\omega} + \omega \times \mathbf{J}\omega &= \sum_{i=1}^{n}(b_i\lambda_i u_i\mathbf{d}_i + \mathbf{r}_i \times u_i\mathbf{d}_i) \end{aligned} \tag{2}$$

Since the net thrust and torque on the right side are linear on u_i, we introduce matrices $\mathbf{M}_f$ and $\mathbf{M}_t$ for compact representation:

$$\begin{aligned} m\ddot{\mathbf{p}} &= m\mathbf{g} + \mathbf{R}\mathbf{M}_f\mathbf{u} \\ \mathbf{J}\dot{\omega} + \omega \times \mathbf{J}\omega &= \mathbf{M}_t\mathbf{u} \end{aligned} \tag{3}$$

The explicit definitions of $\mathbf{M}_f$ and $\mathbf{M}_t$ can be found in Table 1. Since the motor positions $\{\mathbf{r}_i\}$ can be computed from the shape parameter $\mathbf{s}$ by $\mathbf{r}_i = A_i\mathbf{s} + \mathbf{b}_i$, where all $\mathbf{A}_i$ and $\mathbf{b}_i$ are constant matrices and vectors from parametric shape representation, we will omit $\{\mathbf{r}_i\}$ and only use $\mathbf{s}$ in later sections.

6 Controller Design

In this section we introduce the controller used in our multicopters. We start from reformulating the equations of motion into the state-space model, then we use its linear approximation at a fixed point to design an LQR controller.

6.1 State-space model and linear approximation

To design a controller, we rewrite the equations of motion into the following nonlinear form, known as the *state-space representation*:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \tag{4}$$

where $\mathbf{f}$ is a nonlinear function determined by multicopter dynamics. Intuitively the state-space representation describes the fact that given the current state and the actuator output we can predict how the state will change in the future.

Designing a controller for a nonlinear system is not an easy task, so we linearize the state-space model at a fixed point $(\mathbf{x}^*, \mathbf{u}^*)$ and find a controller for the linear model approximation. A fixed point satisfies $\mathbf{f}(\mathbf{x}^*, \mathbf{u}^*) = \mathbf{0}$, which can be explained as a combination of state and thrust such that the copter can stay in this state forever, as long as the thrust does not change.

Define $\bar{\mathbf{x}} = \mathbf{x} - \mathbf{x}^*$, $\bar{\mathbf{u}} = \mathbf{u} - \mathbf{u}^*$, $\mathcal{A} = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_{\mathbf{x}^*,\mathbf{u}^*}$ and $\mathcal{B} = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{u}}\right|_{\mathbf{x}^*,\mathbf{u}^*}$, we can get the linear time-invariant (LTI) approximation of the state space model around $(\mathbf{x}^*, \mathbf{u}^*)$:

$$\dot{\bar{\mathbf{x}}} \approx \mathcal{A}\bar{\mathbf{x}} + \mathcal{B}\bar{\mathbf{u}} \tag{5}$$

For brevity we leave the complete derivation of $\mathbf{f}$, $\mathcal{A}$ and $\mathcal{B}$ in supplemental materials.

6.2 Linear Quadratic Regulator

The state-space model in the last section is different from the actual dynamics model because it is a linear approximation, and it assumes the state is in the neighborhood of the fixed point. As a result, we choose to use LQR because it is a robust controller with a phase margin of at least 60 degrees and an infinite gain margin [Aström and Murray 2010], which means it can remain stable even if the dynamics model deviates a lot from our expectation. Given equation (5), LQR generates the control policy $\bar{\mathbf{u}} = -\mathcal{K}\bar{\mathbf{x}}$ by minimizing the cost function $\int_0^\infty (\bar{\mathbf{x}}^\top \mathcal{Q}\bar{\mathbf{x}} + \bar{\mathbf{u}}^\top \mathcal{R}\bar{\mathbf{u}})dt$, where $\mathcal{Q}$ and $\mathcal{R}$ are user-specified weight matrices which are usually positive diagonal. The first quadratic term tries to shrink $\bar{\mathbf{x}}$ to $\mathbf{0}$, so it penalizes the deviation from stable states. Similarly, the second term discourages actuators from saturation. The matrix $\mathcal{K}$ is found by solving the *Continuous-time Algebraic Riccati Equation* (CARE) once $\mathcal{A}$, $\mathcal{B}$, $\mathcal{Q}$ and $\mathcal{R}$ are given, which has been well-studied and implemented in many linear algebra packages [Laub 1979; MATLAB 2016; SciPy 2014].

Given multicopter dynamics, designing an LQR controller consists of two steps: selecting a fixed point for linearization, and solving CARE. For traditional quadcopters the first step is not a problem because it has a unique fixed point, i.e, each motor provides thrust equal to $1/4$ of the gravity and the quadcopter stays completely level. However, we notice that general multicopter designs often have nonunique fixed points, and the performance of a flying multicopter is influenced by the choice of fixed points so arbitrarily picking a fixed point does not yield good results. To address this issue, we select the fixed point by including it in an optimization problem, which we will describe in the next section.

Although LQR is robust and straightforward to tweak, it is not well-suited for heading control in our hardware platform. As explained in the supplemental materials, applying LQR to control Euler angles requires much more computation because it needs to update $\mathcal{A}$ and $\mathcal{B}$ in every iteration. As a result, in our experiments we limit LQR to position control and trajectory following.

7 Optimization

Optimizing a multicopter is a challenging problem because the performance of a multicopter relies on both its geometry and controller, which are usually coupled with each other. For example, to make sure a multicopter can hover, one needs to find reasonable motor positions and orientations, as well as good output thrusts suggested by the controller. Another challenge comes from the metrics that users want to optimize. For example, it could be a nonconvex function with multiple local minimals, or a nonsmooth function so a gradient-based solver is not applicable.

In this section we introduce our solution to the two challenges above. We provide an algorithm to decouple geometry and control variables during optimization. The geometry variables are motor spin directions $\{b_i\}$, motor orientations $\{\mathbf{d}_i\}$ and shape parameter $\mathbf{s}$. The control variables are the fixed point $(\mathbf{x}^*, \mathbf{u}^*)$ and the control matrix $\mathcal{K}$, as explained in Section 6. As $\mathbf{x}^*$ can be easily determined once $\mathbf{u}^*$ and geometry variables are given, and $\mathcal{K}$ can be determined after geometry and fixed point are known, the optimization algorithm only focuses on finding the optimal $\mathbf{u}^*$ and we leave the derivation of $\mathbf{x}^*$ in supplemental materials. For the metrics, we formulate an objective function including bi-convex user-defined metrics, and we demonstrate that many useful metrics can be represented as bi-convex functions.

The pseudocode for the complete optimization process is provided in Algorithm 1. Our optimization starts from searching the discrete variables $\{b_i\}$ and selecting the combination that is most controllable, then in the main loop we solve three subproblems to optimize the control variable $\{\mathbf{u}^*\}$, shape parameter $\mathbf{s}$ and motor orientations $\{\mathbf{d}_i\}$ in every iteration. After that, we use the geometry variables to build the multicopter and control variables to implement the controller.

Input : Initial geometry variables, acquired from user design: $\{\mathbf{d}_i\}$, $\{\mathbf{r}_i\}$, $\mathbf{s}$
User-specified LQR weight matrices $\mathcal{Q}$, $\mathcal{R}$
Output : Optimized geometry variables: $\{b_i\}$, $\{\mathbf{d}_i\}$, $\{\mathbf{r}_i\}$, $\mathbf{s}$;
Optimized control variables(fixed point): $\mathbf{x}^*$, $\mathbf{u}^*$;
control matrix $\mathcal{K}$

// Preprocessing //
bestCondNumber $\leftarrow +\infty$
foreach $\mathbf{b} \in \{$*all* 2^n *assignments to* $\{b_i\}$ $\}$ **do**
  Compute $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$;
  if *cond*($\mathcal{C}$) $\leq$ *bestCondNumber* **then**
    bestCondNumber = *cond*($\mathcal{C}$);
    $\{b_i\} \leftarrow \mathbf{b}$;
  end
end
// Main loop of the our optimization method ////////////////////////////////
while *not converge* **do**
  Optimize $\mathbf{u}^*$ by solving a convex subproblem;
  Optimize $\mathbf{s}$ by solving a convex subproblem;
  Optimize $\{\mathbf{d}_i\}$ from QCQP relaxation;
  // Check controllability.
  Compute $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{C}$;
  if $\mathcal{C}$ *is singular* **then**
    Revert all variables to their values in the last iteration;
    break;
  end
end
// Postprocessing //
Compute $\mathbf{x}^*$ from $\mathbf{u}^*$;
Compute $\mathcal{A}$, $\mathcal{B}$ from $(\mathbf{x}^*, \mathbf{u}^*)$;
Compute $\mathcal{K}$ from $\mathcal{A}$, $\mathcal{B}$, $\mathcal{Q}$ and $\mathcal{R}$;

Algorithm 1: Optimization algorithm pseudocode.

7.1 Preprocessing

We first determine the spinning directions $\{b_i\}$ for each motor by checking *controllability*. A system is controllable if for any initial state $\mathbf{x}_0$ and final state $\mathbf{x}_1$ there exists a control signal such that the system can steer from $\mathbf{x}_0$ to $\mathbf{x}_1$ in finite time [Aström and Murray 2010]. An LTI system is controllable if the corresponding *controllability matrix* defined on the state-space model has full row rank. For equation (5), the controllability matrix $\mathcal{C}$ is defined as:

$$\mathcal{C} = \begin{bmatrix} \mathcal{B} & \mathcal{A}\mathcal{B} & \mathcal{A}^2\mathcal{B} & \cdots & \mathcal{A}^{11}\mathcal{B} \end{bmatrix} \tag{6}$$

where the maximal exponent is 11 because $\bar{\mathbf{x}}$ consists of 12 variables. We check all 2^n possible combinations to determine the best $\{b_i\}$: for each assignment to $\{b_i\}$ we do linear approximation at its fixed point(if multiple fixed points exist we use the one with minimal L2 norm)then pick the assignment with the smallest condition number. Since n is up to 6 in our case, this straightforward method does not cause any performance issue.

Once $\{b_i\}$ is determined we keep them fixed during the optimization. Our algorithm in later sections checks at the end of each iteration whether the multicopter is still controllable, and it terminates with the best solution so far if controllability is violated. However, in our experiments this check rarely fails and the controllability property is preserved most of the time.

7.2 Problem formulation

Here we give the formal definition of the optimization problem:

$$\begin{aligned} \min_{\mathbf{s},\mathbf{d}_i,\mathbf{u}^*} \quad & E(\mathbf{u}^*,\mathbf{s}) + \eta\|\mathbf{M}_f\mathbf{u}^* + m\mathbf{g}\|_2^2 + \mu\|\mathbf{M}_t\mathbf{u}^*\|_2^2 \\ s.t. \quad & \mathbf{A}^{\text{ineq}}\mathbf{s} \le \mathbf{b}^{\text{ineq}} \\ & \mathbf{d}_i^\top\mathbf{d}_i = 1 \\ & \mathbf{0} \le \mathbf{u}^* \le \mathbf{u}^{\max} \end{aligned} \tag{7}$$

where E is the user-selected metrics, which we will describe in the next section. We require that E rely on the shape parameters $\mathbf{s}$ and thrust $\mathbf{u}^*$ only, and be bi-convex. The second and third terms model soft constraints required by the fixed point with user specified weights η and μ: by definition at a fixed point the net thrust should try to balance the gravity, and the net torque should be zero. Compared with the equations of motion the rotational matrix $\mathbf{R}$ is removed to reduce the complexity, and to indicate that it is preferable for the copter to maintain its original attitude as much as possible, i.e., $\mathbf{R}$ is equal to an identity matrix, which makes the copter easier to take off. Note that $\mathbf{M}_f$ relies on $\mathbf{d}_i$ and $\mathbf{M}_t$ depends on $\mathbf{d}_i$ and $\mathbf{s}$.

The first constraint gives the feasible set of the shape parameter. Constraints on $\mathbf{d}_i$ require that each $\mathbf{d}_i$ should be a unit vector. The last constraint requires that no motor saturation should occur.

7.3 Metrics

Although we limit the energy function to be bi-convex, we demonstrate that a lot of metrics can fit into this representation either directly or after reasonable reformulation. Here we list some:

Payload We define the payload to be the maximal weight the copter can take at its mass center while hovering. Maximizing the payload directly results in a non-convex formulation with all variables closely coupled with each other. Instead, we minimize the following indirect metric:

$$E_{\text{payload}} = \max\left(\frac{u_1^*}{u_1^{\max}}, \frac{u_2^*}{u_2^{\max}}, \cdots, \frac{u_n^*}{u_n^{\max}}\right) \tag{8}$$

E can be explained as searching all the motors and finding the one that is most likely to become saturated. If a copter can hover with thrust equal to $\mathbf{u}$, scaling $\mathbf{u}$ uniformly allows a copter with same geometry but heavier weight to stay in the air. As a result, smaller E indicates possibly larger payload.

Max amperage The relation between thrust and current supplied to the motor can be well approximated by a power function $I = au^\alpha$. Since the power module unit in hardware platform distributes the current from the battery to all motors, for safety reasons we are interested in minimizing the total current so that we do not exceed the maximal amperage of the power module cable:

$$E_{\text{amp\_sum}} = \sum_i^n a_i u_i^{*\alpha_i} \tag{9}$$

where $a_i u_i^{*\alpha_i}$ represents the current supplied to i-th motor. Alternatively, we can minimize the max current supplied to a single motor so that it does not exceed the maximum amperage of the electric speed controller of each motor:

$$E_{\text{amp\_max}} = \max(a_1 u_1^{*\alpha_1}, a_2 u_2^{*\alpha_2}, \cdots, a_n u_n^{*\alpha_n}) \tag{10}$$

Size and flight time To minimize the copter size, we choose to minimize the total length of rods used in the copter, which is linear on the shape parameter $\mathbf{s}$:

$$E_{\text{size}} = \mathbf{c}_{\text{rod}}^\top\mathbf{s} + d_{\text{rod}} \tag{11}$$

We can also relate the copter size to its flight time. From our measurement we notice that larger propellers are more efficient than smaller ones with the motors we use. As a result, we can increase the flight time by maximizing the copter size so that it has enough space to install larger propellers:

$$E_{\text{time}} = -E_{\text{size}} = -(\mathbf{c}_{\text{rod}}^\top\mathbf{s} + d_{\text{rod}}) \tag{12}$$

The negative sign comes from the fact that we minimize the objective function. It should be pointed out that this energy function is based on the observation that larger propellers are more efficient than smaller ones with our motor, which is not always true for all combinations of motors and propellers. Careful measurement needs to be taken before applying this metric.

Cost For a multicopter without a free-form body frame the cost is fully determined by the shape parameter $\mathbf{s}$ in a linear form:

$$E_{\text{cost}} = \mathbf{c}_{\text{cost}}^\top\mathbf{s} + d_{\text{cost}} \tag{13}$$

where $\mathbf{c}_{\text{cost}}$ represents the linear cost like carbon fiber rods, and d_{cost} is the constant cost from components such as battery, controlling board and motors.

Mass For a multicopter without a free-form body frame the mass is linear on the shape parameter $\mathbf{s}$

$$E_{\text{mass}} = \mathbf{c}_{\text{mass}}^\top\mathbf{s} + d_{\text{mass}} \tag{14}$$

When applying this metric, m in the objective function is replaced with $m(\mathbf{s}) = E_{\text{mass}}$. However, this modification won't break our proposed algorithm as the mass is linear on the shape parameters, so it should not break the convexity of the subproblem.

Multi-objective metric Since a non-negative weighted sum preserves convexity, any nonnegative weighted combination of the metrics above meets our bi-convexity requirement. For examples, users may choose to define a metric mixed with both payload and mass, and use weights to express the tradeoff between them. Also note that although all metrics above are defined solely on $\mathbf{u}^*$ or $\mathbf{s}$, a mixed metric can include both of them.

7.4 Algorithm

Directly optimizing the objective function proposed in the previous section is challenging due to the fact that the product between $\mathbf{M}_f$, $\mathbf{M}_t$ and $\mathbf{u}^*$ couples all variables together, and it may contain non-smooth energy functions like the payload or max amperage. However, if we focus on $\mathbf{u}^*$ or $\mathbf{s}$ only it reduces to a simple convex problem. This motivates us to propose our algorithm which alternatively optimizes the thrust, shape parameters and motor directions, as described in Algorithm 1. It terminates when the energy converges within a given threshold (1e-6 in our setting) or it breaks the controllability. With the assumption that user metrics are bi-convex functions, the problem is convex when restricted to optimizing either $\mathbf{u}^*$ or $\mathbf{s}$, which can be efficiently solved in CVX [Grant and Boyd 2014; Grant and Boyd 2008], a package for specifying and solving convex programs.

Optimizing $\{\mathbf{d}_i\}$ is more subtle and bears some discussion. The objective function is quadratic and convex on $\{\mathbf{d}_i\}$ due to the fact that both $\mathbf{M}_f$ and $\mathbf{M}_t$ are linear on $\{\mathbf{d}_i\}$, but the unit-length constraints on $\{\mathbf{d}_i\}$ breaks the convexity. Because of its quadratic form in the objective function, we choose Sequential Quadratic Programming (SQP) [Nocedal and Wright 2006] to solve $\{\mathbf{d}_i\}$. The initial guess is acquired from relaxing the unit length constraints to $\mathbf{d}_i^\top\mathbf{d}_i \le 1$, and then solving the convex Quadratic Constrained Quadratic Programming (QCQP) problem [Boyd and Vandenberghe 2004].

We demonstrate that our algorithm is more suitable than three other general solvers for our optimization problem in Figure 4. In most of the time our algorithm manages to find a better solution, but with

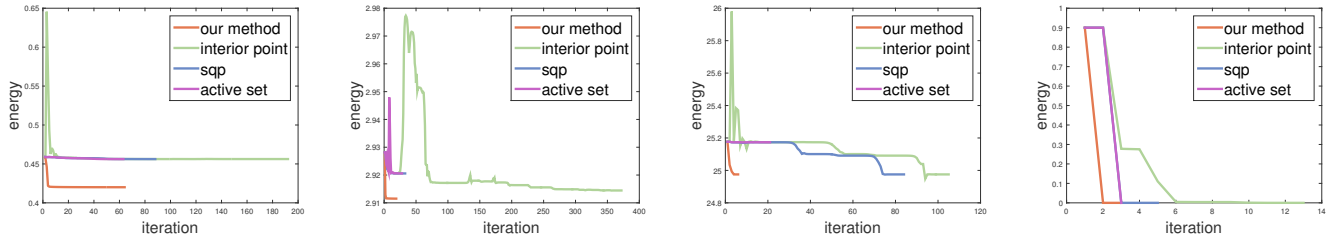

Figure 4: *Comparing our method in multiple copter examples with interior point, sequential quadratic programming and active set methods, all implemented in MATLAB's* `fmincon` *command. All examples have* $\eta = 0.3$ *and* $\mu = 0.7$*, and all methods share the same initial guess and termination conditions. The horizontal axis shows the iterations (note that it does not reflect the true running time as the time an iteration takes varies in different methods) and the vertical axis is the value of the objective function in Equation 7 after each iteration. Left: pentacopter with payload metric. Middle left: bunny with mixed metrics of payload and amperage. Middle right: pentacopter with mixed metrics of max amperage and cost. Right: quadcopter with size metric.*

the cost of longer time. However, this is not a big issue as our solver usually terminates within a few seconds.

In the first two examples, we run our method on two different multicopter designs with single and mixed metrics. For both examples our algorithm quickly finds a better solution after the first few iterations, while the other solvers either fail to make progress or get trapped into a worse local minimal solution. There exist some cases where our algorithm ends up with similar optimal values, in which case it is still acceptable to call our solver as the system is not sensitive to its running time.

Finally, as a sanity check we provide an example where the global minimizer is known. In this example a standard quadcopter is optimized to have minimal size. Without other constraints the copter shrinks to a single point and the energy function becomes zero. In this case all solvers agree on the global minimizer in the end.

8 Simulation

We provide a real-time physics simulator to help users verify the shape and controller design suggested by the optimization. Users can interactively change the input from a virtual RC transmitter and see the flight performance of the copter. Random noises are added to the sensor data to simulate real world environment. If the optimized shape and controller are not satisfactory, users can either manually tweak control parameters in simulation, or go back to the interactive design tool to change the shape representation.

9 Fabrication

Once the shape parameter and motor orientations are determined, we generate a fabrication plan by computing rod lengths, motor angles, and geometry meshes if it contains a free-form body frame (Figure 5). We use 3D printing to fabricate connectors and the body frame. Specifically, we design parametric compound angle clip pieces so that the connectors can support tilted motors.

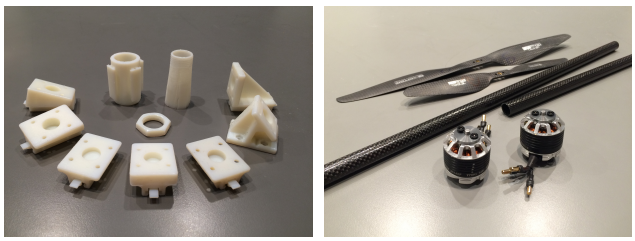

Figure 5: *Some multicopter components used in fabrication. Left: 3D printed connectors that support various motor orientations. Right: propellers, carbon fiber rods and motors.*

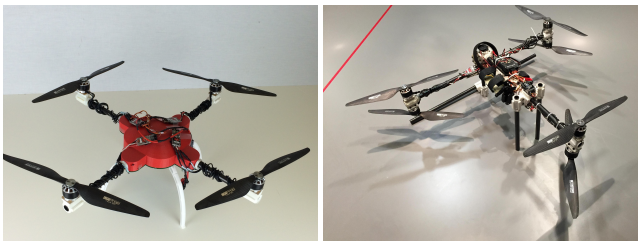

Figure 6: *Classic designs. Left: a quadcopter with a free-form body frame. Right: a hexacopter with coaxial propellers.*

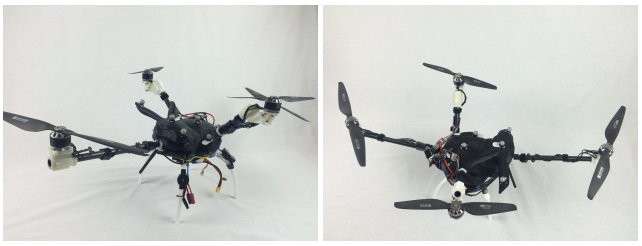

Figure 7: *Bunny copter (left) and its top-down view (right). Note that the positions of motors are not symmetric, their propeller sizes are different and are at various heights.*

Our hardware platform consists of a ground station laptop and a multicopter flight controller. The ground station subscribes the Vicon motion tracking system and sends out real-time position and orientation data to the copter. The flight controller runs our modified version of the open source software ArduPilot [APM 2015] on a Pixhawk [Meier et al. 2012] flight computer hardware.

10 Results

In this section we show multiple examples to demonstrate the effectiveness of our design system, physical simulator, control loop and optimization method. We start from two classic multicopters: an X-frame quadcopter and a Y6 hexacopter. We provide a bunny example to show the expressiveness of our design tool. A pentacopter demonstrates the correctness of our controller and its ability to handle nonstandard copters with odd number of motors. A more challenging pentacopter with tilted motors, which is optimized for the payload metric, shows the efficacy of our optimization method. Finally, we provide a rectangular quadcopter optimized for longer flight time and compare it with a standard quadcopter.

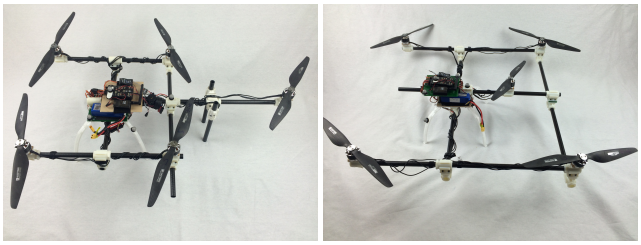

Figure 8: *Pentacopter pairs. Left: original pentacopter design. Right: optimized pentacopter for larger payload.*

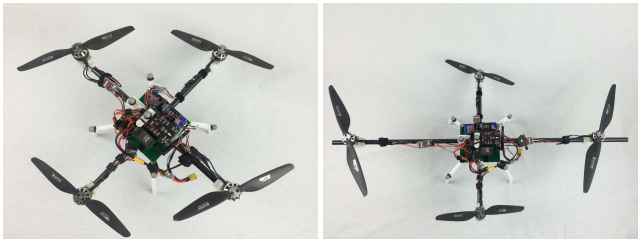

Figure 9: *Optimizing a quadcopter with flight time metric and geometry constraints. Left: a standard quadcopter. Right: optimized rectangular quadcopter.*

Quadcopter We designed a simple quadcopter with a 3D printed red body frame, shown in Figure 6. The red frame is a parametric shape so users can change its size by specifying different shape parameters in the interactive design tool. For simplicity the default quadcopter PID controller in ArduPilot is used here so Vicon is not needed, and no modification to ArduPilot firmware is required.

Hexacopter We designed and fabricated a classic Y6 hexacopter with three pairs of coaxial propellers (Figure 6). As in the previous example, the default Y6 PID controller in ArduPilot is applied and no additional hardware is needed. The hexacopter can change its heading, stabilize itself and fly to a target during the flight.

Bunny A bunny copter is designed and fabricated by using our system (Figure 7). The bunny copter is challenging to fly as the four propellers have different sizes, their positions are not symmetric and they are placed at different heights. Based on its dynamics we compute an LQR controller to control its position in the air. The bunny copter can take off, hover, fly to a target and land.

Unoptimized pentacopter Figure 8 shows a multicopter with five rotors all pointing upright. Flying a multicopter with odd number of rotors, even if it is symmetric, is challenging because there is no straightforward way to distribute thrust so that all motor torques can be balanced easily. However, with the LQR controller suggested by our system this pentacopter can reliably take off, land, hover, and carry over 1kg payload to the destination.

Figure 10 shows the real-time output of all the 5 motors when this pentacopter carries maximal payload from one place to another. Although by default the motor won't saturate until PWM reaches 2000, in this example and its optimized counterpart we clamp PWM at 1800 for safety reasons.

Pentacopter optimized for payload Given the initial unoptimized pentacopter, our optimization improves its payload by changing its geometry and tilting motors to balance thrust from all motors. Figure 10 shows the PWM values from all 5 motors when the optimized pentacopter carries its maximal payload, and Table 2 compares the specifications of two pentacopters. Our optimization result predicts the new pentacopter is able to take off with a 15.8% increase in the overall weight. Note that in theory the maximal

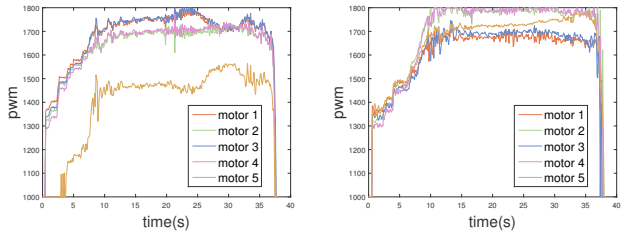

Figure 10: *Left: motor outputs of the unoptimized pentacopter with 1047g payload. Motor 1 and 3 reach saturation point (PWM=1800) at 23s. Increasing the payload will cause them to saturate constantly and therefore fail to balance the torques from other motors. Note that motor 5 is not fully exploited in this copter. Right: motor outputs of the optimized pentacopter with 1392g payload. Motor 2 and 4 reach saturation during the flight. Compared with the unoptimized pentacopter, all five motors are now well balanced, making it possible to take over* 30% *more payload.*

	Unoptimized	Optimized
Size (mm×mm×mm)	750×420×210	650×670×210
Weight (g)	2322	2353
Max payload (g)	1047	1392
Max overall weight (g)	3369	3745
Max motor angle (degree)	0	10.6

Table 2: *Pentacopter specifications. Motor angle is the angle between motor orientation and up direction.*

possible increase in the overall weight is less than 25% because even the unoptimized pentacopter outperforms a quadcopter whose maximal overall weight is $4u^{\max}$, and the maximal possible overall weight of a pentacopter is not greater than $5u^{\max}$. Table 2 shows that we get 11.1% actual gain in our experiments. The main reason for this loss is that we did not take into account the interference between propellers, which we leave as future work.

Quadcopter optimized for flight time Figure 9 shows a standard quadcopter and its optimized version which has longer flight time. Our optimization tries to increase the total length of rods so that it makes room for larger propellers. In this example we add an upper bound constraint on the copter width so it only scales in the other direction, allowing us to replace the propellers in the longer rod with larger ones. This geometry constraint is useful when a quadcopter is designed to fly into a tunnel. Both copters are controlled by LQR controllers computed with our system. In our experiments we let both copters hover for 5 minutes and record the battery voltage and current, shown in Figure 11.

11 Limitation and Future Work

One limitation in our pipeline is that the system responds passively to user inputs for design, optimization and simulation. A potential extension in the future is to have a system that can actively give design suggestions in this case, for example by suggesting to place additional motors and propellers, so the whole design process can be accelerated. In particular, our system fails when the initial geometric design is uncontrollable. For example, a quadcopter initialized with 4 rotors in a line is obviously not fully controllable. In this case, our algorithm gets trapped in assigning spinning directions (Section 7.1), and therefore fails to find a controllable solution, which clearly exists for a quadcopter.

In terms of design, our assembly based approach depends on the library of parts and is therefore limited by its size. In the future it would be nice to define ways to easily grow the database. In addi-

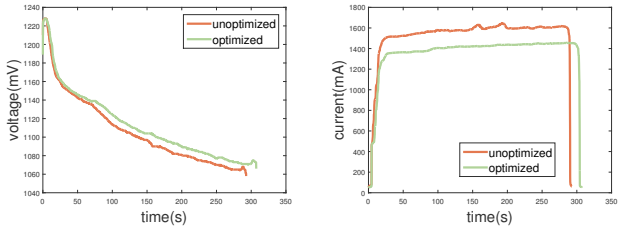

Figure 11: *Battery change when a quadcopter hovers. Left: battery voltage. Given the same amount of time the optimized quadcopter ends up having a larger voltage; Right: battery current. In steady state the optimized copter requires less current.*

tional, the parametrization is constrained to be linear which restricts geometry variability.

In terms of the optimization, the metrics we proposed are limited to functions defined on the fixed points, i.e., the steady states of the copter. While this simplifies the optimization by decoupling the geometry and control variables, it would be useful to extend the optimization so that dynamic metrics can be included and optimized, for example the responsive time to a control signal, or the maneuverability of the copter.

Another limitation is that our real-time physics simulation does not model aerodynamic effects. While a rigid-body simulation provides reasonable results for cases with slow velocity, aerodynamic effects such as interferences between propellers and ground effects need to be modeled to simulate a high speed copter correctly.

12 Conclusion

In this paper we proposed a new pipeline for users to efficiently design, optimize, and fabricate multicopters. Users can easily design a multicopter by interactively assembling components in a user interface. We proposed a new optimization algorithm that can jointly optimize the geometry and controller to improve the performance of a given multicopter design under different metrics, such as payload and battery usage, and can be further verified in a real-time simulator. We demonstrated the ability of our system by designing, fabricating, and flying multicopters with nonstandard designs including asymmetric motor configurations and free-form body frames.

13 Acknowledgments

We thank Nobuyuki Umetani for his insightful suggestions in our discussions. We thank Alan Schultz and his colleagues at NRL for building the hexacopter and for the valuable discussions. We thank Randall Davis, Boris Katz, and Howard Shrobe at MIT for their advice. We are grateful to Nick Bandiera for preprocessing mechanical parts and providing 3D printing technical support; Charles Blouin from RCBenchmark for dynamometer hardware support; Brian Saavedra for the composition UI; Yingzhe Yuan for data acquisition and video recording in the experiments; Michael Foshey and David Kim for their comments on the draft of the paper.

This work was partially supported by Air Force Research Laboratory's sponsorship of Julia: A Fresh Approach to Technical Computing and Data Processing (Sponsor Award ID FA8750-15-2-0272, MIT Award ID 024831-00003), and NSF Expedition project (Sponsor Award ID CCF-1138967, MIT Award ID 020610-00002). The views expressed herein are not endorsed by the sponsors. This project has also received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 645599.

References

APM, 2015. APM autopilot suite. http://ardupilot.org.

ASTRÖM, K. J., AND MURRAY, R. M. 2010. *Feedback systems: an introduction for scientists and engineers*. Princeton University Press.

BÄCHER, M., WHITING, E., BICKEL, B., AND SORKINE-HORNUNG, O. 2014. Spin-it: optimizing moment of inertia for spinnable objects. *ACM Trans. Graph.* 33, 4, (July), 96:1–96:10.

BHARAJ, G., COROS, S., THOMASZEWSKI, B., TOMPKIN, J., BICKEL, B., AND PFISTER, H. 2015. Computational design of walking automata. In *Proc. Symposium on Computer Animation*, 93–100.

BOUABDALLAH, S., NOTH, A., AND SIEGWART, R. 2004. PID vs LQ control techniques applied to an indoor micro quadrotor. In *Intelligent Robots and Systems (IROS) 2004*.

BOYD, S., AND VANDENBERGHE, L. 2004. *Convex optimization*. Cambridge University Press.

COROS, S., KARPATHY, A., JONES, B., REVERET, L., AND VAN DE PANNE, M. 2011. Locomotion skills for simulated quadrupeds. *ACM Trans. Graph.* 30, 4, (July), 59:1–59:12.

COROS, S., THOMASZEWSKI, B., NORIS, G., SUEDA, S., FORBERG, M., SUMNER, R. W., MATUSIK, W., AND BICKEL, B. 2013. Computational design of mechanical characters. *ACM Trans. Graph.* 32, 4, (July), 83:1–83:12.

GARG, A., SAGEMAN-FURNAS, A. O., DENG, B., YUE, Y., GRINSPUN, E., PAULY, M., AND WARDETZKY, M. 2014. Wire mesh design. *ACM Trans. Graph.* 33, 4, (July), 66:1–66:12.

GRANT, M., AND BOYD, S. 2008. Graph implementations for nonsmooth convex programs. In *Recent Advances in Learning and Control*. Springer, 95–110. http://stanford.edu/~boyd/graph_dcp.html.

GRANT, M., AND BOYD, S., 2014. CVX: MATLAB software for disciplined convex programming, version 2.1. http://cvxr.com/cvx.

HOFFMANN, G., RAJNARAYAN, D. G., WASLANDER, S. L., DOSTAL, D., JANG, J. S., AND TOMLIN, C. J. 2004. The Stanford testbed of autonomous rotorcraft for multi agent control (STARMAC). In *Digital Avionics Systems Conference (DASC) 2004*.

HOFFMANN, G. M., HUANG, H., WASLANDER, S. L., AND TOMLIN, C. J. 2007. Quadrotor helicopter flight dynamics and control: theory and experiment. In *Proc. AIAA Guidance, Navigation, and Control Conference*.

JAIN, S., AND LIU, C. K. 2009. Interactive synthesis of human-object interaction. In *Proc. Symposium on Computer Animation*, 47–53.

JOUBERT, N., ROBERTS, M., TRUONG, A., BERTHOUZOZ, F., AND HANRAHAN, P. 2015. An interactive tool for designing quadrotor camera shots. *ACM Trans. Graph.* 34, 6, (November), 238:1–238:11.

JU, E., WON, J., LEE, J., CHOI, B., NOH, J., AND CHOI, M. G. 2013. Data-driven control of flapping flight. *ACM Trans. Graph.* 32, 5, (September), 151:1–151:12.

KOO, B., LI, W., YAO, J., AGRAWALA, M., AND MITRA, N. J. 2014. Creating works-like prototypes of mechanical objects. *ACM Trans. Graph.* 33, 6, (November), 217:1–217:9.

LAU, M., OHGAWARA, A., MITANI, J., AND IGARASHI, T. 2011. Converting 3D furniture models to fabricatable parts and connectors. *ACM Trans. Graph.* 30, 4, (July), 85:1–85:6.

LAUB, A. 1979. A Schur method for solving algebraic Riccati equations. *IEEE Transactions on Automatic Control* 24, 6, 913–921.

LEISHMAN, J. G. 2006. *Principles of helicopter aerodynamics*. Cambridge University Press.

LENTINE, M., GRÉTARSSON, J. T., SCHROEDER, C., ROBINSON-MOSHER, A., AND FEDKIW, R. 2011. Creature control in a fluid environment. *IEEE TVCG* 17, 5, 682–693.

MAGNUSSEN, Ø., HOVLAND, G., AND OTTESTAD, M. 2014. Multicopter uav design optimization. In *IEEE/ASME 10th International Conference on Mechatronic and Embedded Systems and Applications (MESA)*, 1–6.

MAGNUSSEN, Ø., OTTESTAD, M., AND HOVLAND, G. 2015. Multicopter design optimization and validation. *Modeling, Identification and Control* 36, 2,, 67.

MARTIN, T., UMETANI, N., AND BICKEL, B. 2015. OmniAD: data-driven omni-directional aerodynamics. *ACM Trans. Graph.* 34, 4, (July), 113:1–113:12.

MATLAB, 2016. Linear-Quadratic Regulator (LQR) design. http://www.mathworks.com/help/control/ref/lqr.html.

MEIER, L., TANSKANEN, P., HENG, L., LEE, G. H., FRAUNDORFER, F., AND POLLEFEYS, M. 2012. PIXHAWK: a micro aerial vehicle design for autonomous flight using onboard computer vision. *Autonomous Robots* 33, 1-2.

MORI, Y., AND IGARASHI, T. 2007. Plushie: an interactive design system for plush toys. *ACM Trans. Graph.* 26, 3, (July), 45:1–45:8.

NOCEDAL, J., AND WRIGHT, S. 2006. *Numerical optimization*. Springer Science & Business Media.

POPOVIĆ, J., SEITZ, S. M., ERDMANN, M., POPOVIĆ, Z., AND WITKIN, A. 2000. Interactive manipulation of rigid body simulations. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., 209–217.

PRÉVOST, R., WHITING, E., LEFEBVRE, S., AND SORKINE-HORNUNG, O. 2013. Make it stand: balancing shapes for 3d fabrication. *ACM Trans. Graph.* 32, 4, (July), 81:1–81:10.

RAFFO, G. V., ORTEGA, M. G., AND RUBIO, F. R. 2010. An integral predictive/nonlinear H_∞ control structure for a quadrotor helicopter. *Automatica* 46, 1, 29–39.

RCBENCHMARK, 2016. Dynamometer. https://www.rcbenchmark.com/.

ROBERTS, M., AND HANRAHAN, P. 2016. Generating dynamically feasible trajectories for quadrotor cameras. *ACM Trans. Graph.* 35, 4, (July).

SAUL, G., LAU, M., MITANI, J., AND IGARASHI, T. 2011. Sketchchair: an all-in-one chair design system for end users. In *Proc. International Conference on Tangible, Embedded, and Embodied Interaction*, 73–80.

SCHULZ, A., SHAMIR, A., LEVIN, D. I. W., SITTHI-AMORN, P., AND MATUSIK, W. 2014. Design and fabrication by example. *ACM Trans. Graph.* 33, 4, (July), 62:1–62:11.

SCIPY, 2014. CARE solver. http://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.linalg.solve_continuous_are.html.

SKOURAS, M., THOMASZEWSKI, B., KAUFMANN, P., GARG, A., BICKEL, B., GRINSPUN, E., AND GROSS, M. 2014. Designing inflatable structures. *ACM Trans. Graph.* 33, 4, (July), 63:1–63:10.

TAN, J., GU, Y., TURK, G., AND LIU, C. K. 2011. Articulated swimming creatures. *ACM Trans. Graph.* 30, 4, (July), 58:1–58:12.

TAN, J., GU, Y., LIU, C. K., AND TURK, G. 2014. Learning bicycle stunts. *ACM Trans. Graph.* 33, 4, (July), 50:1–50:12.

TAYEBI, A., AND MCGILVRAY, S. 2004. Attitude stabilization of a four-rotor aerial robot. *In Proc. IEEE Conference on Decision and Control.* 2, 1216–1221.

TEDRAKE, R., 2014. Underactuated robotics: algorithms for walking, running, swimming, flying, and manipulation (course notes for MIT 6.832). http://underactuated.csail.mit.edu/underactuated.html.

TWIGG, C. D., AND JAMES, D. L. 2008. Backward steps in rigid body simulation. *ACM Trans. Graph.* 27, 3, (August), 25:1–25:10.

UMETANI, N., KAUFMAN, D. M., IGARASHI, T., AND GRINSPUN, E. 2011. Sensitive couture for interactive garment modeling and editing. *ACM Trans. Graph.* 30, 4, (July), 90:1–90:12.

UMETANI, N., IGARASHI, T., AND MITRA, N. J. 2012. Guided exploration of physically valid shapes for furniture design. *ACM Trans. Graph.* 31, 4, (July), 86:1–86:11.

UMETANI, N., KOYAMA, Y., SCHMIDT, R., AND IGARASHI, T. 2014. Pteromys: interactive design and optimization of free-formed free-flight model airplanes. *ACM Trans. Graph.* 33, 4, (July), 65:1–65:10.

VOUGA, E., HÖBINGER, M., WALLNER, J., AND POTTMANN, H. 2012. Design of self-supporting surfaces. *ACM Trans. Graph.* 31, 4, (July), 87:1–87:11.

WAMPLER, K., AND POPOVIĆ, Z. 2009. Optimal gait and form for animal locomotion. *ACM Trans. Graph.* 28, 3, (August), 60:1–60:8.

WASLANDER, S. L., HOFFMANN, G. M., JANG, J. S., AND TOMLIN, C. J. 2005. Multi-agent quadrotor testbed control design: integral sliding mode vs reinforcement learning. In *Intelligent Robots and Systems (IROS) 2005*.

WHITING, E., SHIN, H., WANG, R., OCHSENDORF, J., AND DURAND, F. 2012. Structural optimization of 3D masonry buildings. *ACM Trans. Graph.* 31, 6, (November), 159:1–159:11.

WU, J.-C., AND POPOVIĆ, Z. 2003. Realistic modeling of bird flight animations. *ACM Trans. Graph.* 22, 3, (July), 888–895.

ZHU, L., XU, W., SNYDER, J., LIU, Y., WANG, G., AND GUO, B. 2012. Motion-guided mechanical toy modeling. *ACM Trans. Graph.* 31, 6, (November), 127:1–127:10.