A Scripted Printable Quadrotor: Rapid Design and Fabrication of a Folded MAV

Ankur M. Mehta and Daniela Rus; Kartik Mohta, Yash Mulgaonkar, Matthew Piccoli, and Vijay Kumar

Abstract Robotic systems hold great promise to assist with household, educational, and research tasks, but the difficulties of designing and building such robots often are an inhibitive barrier preventing their development. This paper presents a framework in which simple robots can be easily designed and then rapidly fabricated and tested, paving the way for greater proliferation of robot designs. The Python package presented in this work allows for the scripted generation of mechanical elements, using the principles of hierarchical structure and modular reuse to simplify the design process. These structures are then manufactured using an origami-inspired method in which precision cut sheets of plastic film are folded to achieve desired geometries. Using these processes, lightweight, low cost, rapidly built quadrotors were designed and fabricated. Flight tests compared the resulting robots against similar micro air vehicles (MAVs) generated using other processes. Despite lower tolerance and precision, robots generated using the process presented in this work took significantly less time and cost to design and build, and yielded lighter, lower power MAVs.

1 Introduction

Robotic systems by their very nature can be highly capable and versatile, providing enhanced actuation and automation abilities to enable otherwise untenable activities. Robots can be useful by themselves [1], or can assist in conducting unrelated research [2]. Robots are also useful in furthering educational goals [3].

However, the tightly coupled nature of mechanical, electrical, and software subsystems in a robot often demand sophisticated engineering skills to design and build an appropriate device. Computer-aided design (CAD) packages can be expensive and arcane, and numerous such tools are often required. The varied toolsets and de-

Massachusetts Institute of Technology, {mehtank, rus}@csail.mit.edu;
University of Pennsylvania, {kmohta, yashm, piccoli, kumar}@seas.upenn.edu

sign environments cause unique specialized robots to be created from the ground up for each application, with little design reuse across projects.

High prototyping costs and turnaround times are also impediments to robot development. Conventional fabrication techniques for mechanical and electrical components typically require trained technicians operating sophisticated machinery, and thus take significant investments of time and money. Recent developments in 3D printing technologies have done much to ameliorate such effects, allowing a vast variety of solid parts to be fabricated in a home environment. Nonetheless, such 3D printers still take hours to produce parts, making incremental improvements and rapid turnaround untenable.

Ultimately, robot creation currently tends to remain an expert's domain. In order for robotics to become prevalent, then, the entire process from conception to construction needs to be modified.

A key element to simplify the design process is to enable modular design. By breaking up a robot into functional subsystems, new designs can be made from previously developed and tested components of other robots. These modules can span disciplines – robotic subsystems necessarily involve mechanical, electrical, and software components. As more robots get designed in the system, the corpus of available components will grow; eventually robot development can transition to making high level functionality decisions and designing by connecting components from libraries.

Alternate fabrication processes are also necessary to ease robot development. As design often involves repeated testing with incremental improvement, a quick and cheap method of rapidly iterating prototypes can greatly increase the number of build-test-refine cycles, resulting in more varied and successful robots.

With these goals in mind, a new process was developed for rapid robot development. A quadrotor micro air vehicle (MAV) system was selected as a simple but nontrivial instance of a robotic specification. Its mechanical structure is basic enough to concisely demonstrate the modular code-based design, while its resulting behavior and performance is rich enough to demonstrate the relevance of this process.

Mechanical airframes were designed using a set of scripts in the Python programming language, then fabricated by folding a sheet of cut plastic. The flight control board and software were designed using existing library modules. The completed MAV was then flown to analyze it's performance. Data from these flights were compared across similar robots created using alternate design and fabrication techniques to characterize the scripted printable process.

The contributions presented in this paper include:

- the new scripted design platform and printed folding fabrication process for producing mechanical structures,
- a specific lightweight, low cost, and rapidly designed and manufactured MAV generated using them, and
- a comparison of these to other conventional processes and MAVs.

In particular, this process outshines other robot design methodologies by:

- allowing extensive *modular design*, to the level of simply connecting existing modules to generate a final robot,
- requiring easily available *low cost* software, hardware, and raw materials,
- fabricating the designs in considerably *shorter time*,
- producing *lighter* yet robust mechanical structures,
- enabling *rapid iteration* through the use of scripted design.

This paper begins in section 2 with a discussion on the various rapid fabrication methods available for creating mechanical structures, and compares them to the folded plastic process that is the focus of this work. Section 3 follows by outlining methods for designing mechanical structures – existing CAD programs are compared against the scripted design methodology developed herein. The modular design of the control system, now common in hardware and software development, is explained in section 4. Section 5 describes the setup in place to characterize the robots and presents the data from those flight tests. The conclusion in section 6 examines the results presented in this work.

2 Rapid fabrication processes

A wide variety of conventional manufacturing processes can be used to generate custom parts to meet arbitrary size, weight, and strength requirements of robotic components. However, these generally require considerable time, expense, and expertise. Rapid fabrication processes trade off precision and specificity for ease of use and build speed; some of these are described below. In particular, a process is proposed in which precision cut plastic sheets are folded to realize desired 3D geometries. This has the benefit of producing extremely light structures in a small fraction of the time required by other methods.

2.1 3D printing

3D printing is an additive manufacturing technique that has gained widespread popularity for producing rapid prototypes. It owes this popularity in large part to the relatively low cost of raw materials and hobbyist tabletop machines and to the ease of producing a part from a CAD specification when compared to traditional machining practices. However, 3D printing can still take on the order of hours per part. Furthermore, 3D printing is not always cost effective.

This method was used to produce a number of quadrotor frames, some of which are shown in figure 1. An industrial-grade printer was used to fabricate a frame out of acrylonitrile butadiene styrene (ABS), a common thermoplastic, a hobbyist grade printer was used to make frames out of polylactic acid (PLA), another common source material, while a desktop printer was used to make frames with a photopolymer. Fabrication of these structures took between one to three hours. The

parts ranged in weight from 6 g to upwards of 15 g, based on the configuration options set during manufacture. The lightest part was nothing more than a hollow shell, with a single layer of plastic forming the surface of the geometry (at 0% infill).

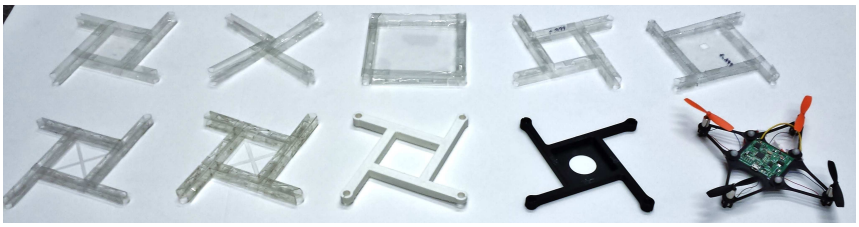


Fig. 1 A fleet of quadrotor frames weighing 4 g – 15 g were manufactured using 3D printing technologies and folding. Source materials included ABS, PLA, photopolymer plastics, and polyester

2.2 Plastic sheet cutting

Precision cutting can be used to generate arbitrary 2D parts from potentially inexpensive solid plastic sheets. This method has distinct benefits over other machining methods; though laser cutters are more expensive than the cheapest 3D printers, higher precision is achievable at a lower cost than professional grade printers or conventional machining. Alternately, desktop paper or vinyl cutters can accomplish a similar task for a fraction of the price. Design and machining requires simpler CAD tools and demands less skill from a designer. And fabrication is fast – parts generally take on the order of minutes to cut. However, precision cutting is limited to directly producing strictly 2D designs.

2.3 Origami-inspired folding

Inspired by the traditional Japanese art of origami, folding is an efficient method of creating 3D structures from planar fabrication processes such as the sheet cutting described above. Using 2D processes such as cutting or laser machining, folding patterns can be formed on a thin flat substrate similar to creasing a sheet of paper to create a tendency to fold along these creases. The resulting perforation lines can be manually or automatically [4] folded in both mountain and valley directions. This approach produces printed 2D precursors that are subsequently folded into rapidly and easily fabricated 3D robots [5–7]. Quadrotor frames designed using the Python package described below can be seen in figures 2 and 3. These designs were cut out of sheets of polyester (PE) film using a commercial laser cutter. Arbitrary thickness film could be used: the lightest frames, made using 0.005” thick stock with a latticed

design, weighed less than 3g; the strongest structures, made with unbroken 0.010" film, weighed just under 8g.

3 Mechanical design

Robot mechanical design involves generating 3D geometries that meet structural, kinematic, and dynamic requirements for specific robot abilities. There are a number of software tools that engineers can use to aid in this process; some programs are compared below. The new work presented in section 3.3 is a Python-based scripted design package, developed to provide a more beginner-friendly tool that can allow for powerful automation, modular design, and widespread sharing and reuse of components and scripts.

3.1 Commercial CAD software

Among professionals in the robotics community, mechanical design is most often done using commercial CAD packages such as SolidWorks and AutoCAD. SolidWorks is known for its user-friendly interface and 3D capabilities, while AutoCAD is often used for its command-line power at 2D drawings. Both programs are highly featureful when it comes to making sophisticated designs, but are very expensive. They also require significant processing power and graphics hardware to operate.

In addition to their high cost, the underlying proprietary nature of these tools inhibits widespread sharing or collaborative design. Binary source files inhibit the effective use of sophisticated distributed version control systems to fork and merge designs and changes. Furthermore, though scripting options exist in such programs, they are not well developed; rather than being able to automate repetitive design decisions, most tasks are left to the user to implement.

For this work, SolidWorks was used to design quadrotor frames to be 3D printed. The design was quickly implemented from scratch by an experienced designer, and the entire geometry was manually drawn and each dimension was individually constrained. Though dimensional changes were easy to implement, larger scale configuration modifications required starting a completely new design.

3.2 OpenSCAD

OpenSCAD is a free open source cross-platform 3D CAD program. It takes C-style text-based programs as input to generate 3D geometries by hierarchically applying primitive operations to basic shapes. With a low cost to entry given both the free program and minimal hardware requirements, it has become popular among hob-

byist designers. The text-based source enables widespread sharing and modification, as demonstrated in online communities such as Thingiverse [8]. It also enables simple reconfiguration by rearranging the modular definitions of constituent elements. However, though scripted, it is primarily focused on object generation, with only basic automation options. Nonetheless, again due to its text-based input, other scripting languages can be used to automate OpenSCAD program generation.

3.3 Python

Python is a user-friendly scripting language with an extensive collection of modules. With a large community of users, it has been used to interface to CAD programs, as well as generate CAD designs directly. In this work, a Python package was developed to directly generate 2D design files for the origami-inspired folded plastic sheet fabrication process.

This package leverages hierarchical design principles to build a library of building blocks from which to generate mechanical designs. Much like OpenSCAD, operations on basic geometries are included in the package, from which scripts can assemble more complicated geometries. However, because Python is a fully featured programming language, and the internal representation of the designed geometries are fully available in userspace, many design tasks can be fully automated. This greatly simplifies the modification and extension of existing designs, even by non-expert designers.

The basic unit of design for folded robots is the face, a polygon that represents a flat continuous section of the source sheet. Faces can be joined at folded edges to make 3D structural elements. For example, a beam is formed by an array of rectangular faces as shown in listing 1. These elements form the basic building blocks in the package's library. Other building blocks include additional structural elements such as various polyhedra to form bodies, wheels, etc.; as well as combining forms such as tabs and slots, hinges, and pleats.

Listing 1 A simple beam is formed by joining a number of rectangles along folded edges

```
1 class Beam(Drawing):
2   def __init__(self, length, thickness, shape=3):
3     Drawing.__init__(self)
4
5     r = Rectangle(thickness, length)
6     self.append(r.times(shape, 'e3', 'e1', 'r', FOLD))
```

Complicated mechanical design can then be reduced to hierarchically combining simpler building blocks with appropriate joints. Though this package focuses on the folded plastic process, a similar approach can include other fabrication methods.

A quadrotor frame is a simple illustrative example that can clearly demonstrate this design process. The frame itself is hierarchically composed of submodules:

- a belt comprises a rectangular face with tabs and slots for mounting,

- a motor mount includes a belt along with notches for the motor to sit against,
- an arm is created by attaching the motor mount to the end of a beam.

The quadrotor frame can then be generated by symmetrically joining four such motor arms. Example code demonstrating this composition can be seen in listing 2.

Listing 2 Modular hierarchical design of the constituent motor arms for a quadrotor

```
class Belt(Rectangle):
  def __init__(self, thickness, length):
    Rectangle.__init__(self, thickness, length)

    t = tabs.BeamTabSlot(thickness, thickness)
    self.attach('e2', t.tabs.times(2, 'e0', 'e2', 'mt', FLAT),
        'mt0.e0', 'mt0', FLAT)
    self.slots = t.slots.times(2, 'e0', 'e2', 'ms', FLAT)

class MotorMount():
  def __init__(self, thickness, motor_diameter):
    self.belt = Belt(thickness, pi * motor_diameter/2.)
    self.notch = Face(((thickness, 0), (thickness/2.,
        thickness/2.))).flip()

  def connect(self, beam):
    beam.attach('r0.e2', self.notch, 'e0', 'm0', FLAT)
    beam.attach('r1.e2', self.belt.slots, 'ms0.e0', 's0', CUT)
    beam.attach('r2.e2', self.notch, 'e0', 'm1', FLAT)
    beam.attach('r3.e2', self.belt, 'e0', 'mt', FLAT)

class MotorArm(Beam):
  def __init__(self, length, thickness, motor_diameter):
    Beam.__init__(self, length, thickness, 4)

    m = MotorMount(thickness, motor_diameter)
    m.connect(self)
```

This code is straightforward to generate – a user-friendly interface can automatically synthesize code like this from intuitive graphical input. Furthermore, once such components have been designed and committed to the library of parts, the details of their implementation are no longer relevant. New designs can be formed through combinations of these underlying building blocks. Existing designs can be easily reconfigured by adjusting the composition parameters of the constituent modules.

In addition to the modularity, another benefit of this method of design comes from its scripting abilities. An automated script was able to generate matching tabs and slots to attach the faces for each beam based on overall geometry. Another script was used to perforate each solid face with lattice-like speed holes, as shown in figure 3. Though such scripts take some time and expertise to develop initially, they can then be applied at will. In contrast to manual design tools that require repetitive placement of each feature, this process enables future designs to apply a short snippet of code to modify the entire design, regardless of complexity, as seen in listing 3.

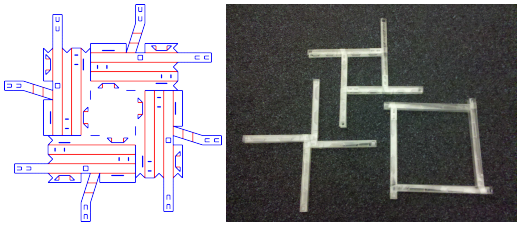


Fig. 2 The folded frames on the right are made from design files generated by the developed Python package like the one on the left. A laser cutter or desktop vinyl cutter operates on a sheet of plastic, cutting along the blue lines and perforating along the red lines.

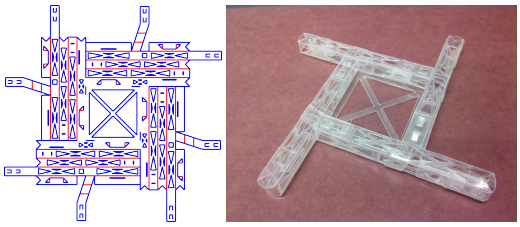


Fig. 3 Automated scripts can quickly and easily apply complex modifications to the generated designs, such as the addition of lattice shaped speed holes on each solid wall of the quadrotor frame.

Listing 3 Scripting capabilities allow repetitive or complex modifications to be easily encapsulated and shared

```
1 import quad
2 import lattice
3
4 q = quad.Quad(radius = 75)
5 lattice.latticify(q)
```

4 Control system

4.1 Electronic hardware design

Similar to the Python package for mechanical design, the controller for these vehicles employed modular design to quickly develop highly customized circuits from an electrical design library of component modules. Expert schematic designers create self contained schematic sheets for various electronic subsystems using EAGLE PCB Design software; experienced PCB designers can create accompanying board layouts for these schematics. Complex boards can then be compiled from these libraries by inexperienced users by stitching the desired components together. The generated hardware designs can be sent out to commercial foundries to get manufac-

tured. The result is indistinguishable from an expert designer's board yet consumes less time and skill to compile.

The quadrotor hardware comprises six library modules. An ARM Cortex M4 STM32F373 microprocessor forms the central controller, interfacing with an Atmel AT86RF212 900MHz Zigbee transceiver for wireless communication and an InvenSense MPU-6050 six degree of freedom MEMS gyro plus accelerometer for inertial sensing. A voltage regulator module regulates power to these components. Four instances of DC brushed motor drivers complete the flight control board. A USB connection to the microprocessor allows for programming and direct computer communication. Because both the USB and wireless blocks are used, an identical board can connect to a host computer, enabling bi-directional wireless communication without needing special basestation hardware. The final board is less than 15 cm^2 and can be made in a single layer process.

4.2 Software

The software driving a quadrotor controller needs to stabilize both its attitude dynamics as well as its position. The attitude controller needs a high update rate to handle the fast dynamics of the small frames; this then allows a position controller to run at a slower rate.

4.2.1 Onboard Controller

The low level software package complements the electronic hardware. A software library for each electronic module exists in the database. Their inclusion is done via preprocessor statements indicating the hardware's pin location. Future versions will extract this data from the schematic files automatically. Additional libraries are included in the same manner and range from LED manipulation to timers to brushless motor vector control. The quadrotor uses the Zigbee radio, IMU, serial packetization, and LED manipulation libraries, while the basestation uses the same libraries plus the USB and minus the IMU.

The microcontroller estimates its attitude at 2 kHz, rate limited by the IMU output data rate. A PD attitude controller runs on the microcontroller with target attitudes received over the radio from a basestation computer. Motor speeds are controlled by pulse-width modulation (PWM) commands to the motor drivers. In addition to maintaining stable hover, the autopilot wirelessly reports the measured inertial rates, calculated orientation, and commanded control inputs for data logging, post processing, and analysis.

4.2.2 Base-station Software

The outer loop position control is implemented using a pose and yaw estimate generated at 100 Hz by an external motion capture system from Vicon Motion Systems [9]. The position controller is a simple PID controller as described in [10] with an integral terms added for the 3 axes.

5 Flight testing

In order to evaluate the relative merits of the fabrication methods, a common design for a quadrotor frame was realized in the different processes. An additional frame of similar dimensions, but a more traditional design was used as a control. These frames were then outfitted with the same controller and actuators and flown through a series of tests. The flight tests consisted of both autonomous hovering and controlled flight using the control system described above. A full summary of the frames tested in this work is presented in tables 1 and 2. These fabricated devices were compared against a commercially available option, the NanoQuad from KMel Robotics [11].

Data collection for the flight parameters of the various models was conducted in two phases. Power consumption during hover was measured while tethered to a power supply, yet still carrying its battery, as seen in figure 4. To measure dynamic parameters, the power supply was disconnected and the battery was used. The data logged during the untethered flights included the position of the quadrotor and all feedback indicated in section 4.2.1.

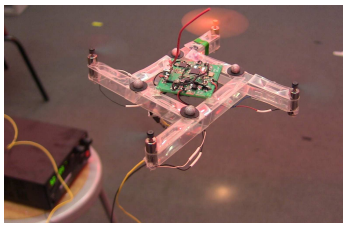


Fig. 4 A folded plastic quadrotor is tethered to a laboratory supply to measure power consumption during controlled hover.

To ensure reasonable comparisons between the different frames, the vehicle controllers were modified for each frame. Frequently, quadrotors are described as a fourth order system (although they can go above fifth order), and are divided into two second order PD controllers. Separation of time scales allows us to do this if the inner controller is significantly faster than the outer. The inner, high speed, onboard PD controller controls vehicle attitude and outputs motor PWM commands. The outer, lower speed, offboard controller controls vehicle position and commands

vehicle attitude. In both cases, there is negligible natural damping (from wind resistance) and no returning force around the hover condition in the horizontal directions. Thus, all proportional and derivative effects come from the controllers alone. Examining the vehicle's attitude behavior along one of the principal axis, after taking the Laplace transform, the simplified transfer function becomes:

$$\frac{Y(s)}{U(s)} = \frac{K_p + K_d s}{Is^2 + K_d s + K_p}$$

where K_p and K_d are the proportional and derivative gains, and I is the vehicle's moment of inertia along that axis.

Solving the for the poles:

$$s = \frac{-K_d \pm \sqrt{K_d^2 - 4IK_p}}{2I}$$

we see that the poles converge at $K_d = \sqrt{4K_pI}$. This corresponds to critical damping, which was the target for the experiments. In reality, the coefficient of 4 does not yield critical damping due to sensor error, time delays, etc. The coefficient and K_p for the inner control loop were found manually while K_d was adjusted according to the formula above for each frame. The same process was repeated in the outer control loop, replacing I for the mass.

5.1 Comparison of airframe properties

Airframe comparisons were broken down into three categories: production, frame characteristics, and assembly. Production comparisons include fabrication time, labor time, material costs, and machine costs. Frame characteristics are mass, stiffness, and brittleness.

The fabrication time of 2D designs are markedly lower than their 3D counterparts. These range from 1 to 2 minutes using a laser cutter, and 4 minutes using a desktop electronic crafts cutter. Only frame (A) from table 1 required no manual labor to construct the frame. The foldable frames can be folded by an inexperienced person without directions or tools in less than 30 minutes, while an experienced person can complete a frame in as little as 6 minutes. Depending on the 3D printer type, the removal of support material (scaffolding used during the printing process) takes a comparable time to folding, and can require more equipment like acid baths or high powered water guns. The folded frames are the cheapest material-wise. The 0.005" polyester frames, (B) and (D), cost roughly $0.13 USD when bought in quantity and the 0.010" frames, (C) are $0.25. They were cut on both $35,000 laser cutters and $250 electronic cutting machines with similar results. The 1/16" acrylic frame uses $0.75 of material, but cannot be cut with the low cost electronic cutter. The 3D printed ABS, (E), or PLA frames are $0.75 not including support material, which

Table 1 A comparison of the design of the various quadrotor frames built and flown

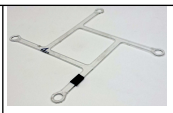
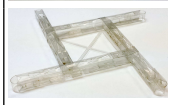
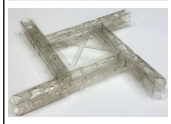
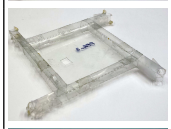
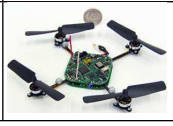
		Design process	Fabrication method	Material
(A)		SolidWorks	Laser Cut	1/16” Acrylic
(B)		Python script	Laser Cut + Fold	0.005” PE
(C)		Python script	Laser Cut + Fold	0.010” PE
(D)		Python script	Electronic Cut + Fold	0.005” PE
(E)		SolidWorks	Fused Deposition Modeling	ABS
(F)		SolidWorks	Stereolithography	Photopolymer
(G)		(N/A)	Purchased	Carbon fiber

Table 2 A comparison of the performance metrics of the various quadrotor frames

	Fabrication time (min)	Frame mass (g)	Total mass (g)	Hover power (W)	Power to mass (W/g)	Hover STD X (cm)	Hover STD Y (cm)	Hover STD Z (cm)	Rise Time (s)
(A)	**1**	**3.9**	**38.9**	**9.9**	0.254	1.08	1.26	**0.05**	**0.344**
(B)	8	4.0	39.0	10.2	0.262	1.13	0.98	0.11	0.352
(C)	8	7.6	43.0	10.4	0.242	1.12	1.93	0.11	0.374
(D)	10	4.0	39.0	10.2	0.262	**0.66**	**0.66**	0.08	0.400
(E)	58	15.2	51.0	11.2	**0.220**	2.22	4.94	0.16	0.480
(F)	150	7.4	42.4	10.4	0.245	1.23	1.11	0.12	0.366
(G)	(N/A)		74.9	17.2	0.230	1.50	1.42	0.21	

could bump it up to $1.00, and were printed on $30,000 machines. Hobby versions of this machine are now in the low $1000 range. The control frame, (F), cost $11 to print on a $20,000 machine. In contrast, the commercial option (G) costs on the order of $4000.

The frames' masses vary greatly, ranging from 3.9 g to 15.2 g. While in general, frame mass corresponds to frame rigidity, the folded frames have a small amount of play due to the clearances required to feed tabs through slots. The ideal solution appears to be to 3D print extremely light frames like the white frames found in figure 1. Unfortunately, 3D printers have a minimum thickness on the order of the polyester sheets used for folding. At the minimum thickness, each wall of the vehicle is a single layer thick, which is very brittle in the FDM style of 3D printing. Note that none of these frames were tested since their layers delaminated before any significant testing could be completed. Furthermore, frame (A) broke in a hard landing after flight testing was complete. Because it is flat, it is particularly weak in the vertical direction. Tubular or rod shapes with uniform materials like the folded polyester and the photopolymer did not experience this problem. These frames are also compliant without being brittle, contributing to their high tolerance to damage.

The stiff, 3D printed frames were quick and simple to assemble. Features like press fits for the motors not only kept the motors in place, but also aiming the motors in the proper direction. Utilizing the belts mentioned in section 3.3, the folded frames easily accepted the motors. Care was required during assembly, however, since the play in the frames could cause the belt to grab the motor on an angle, resulting in motors facing several degrees off of vertical. Frame (A) had no mechanism for aligning motors as it is a thin, 2D design. Motors are aligned by eye and held in place using hot melt adhesive. The first flight of this frame resulted in uncontrollable yawing. Only after careful remounting was it a competitive frame.

5.2 Comparison of flight characteristics

A plot of the hover performance of the folded frames (frame B and frame C) and a laser-cut frame (frame A) is shown in figure 5. The deviation of the position from the desired setpoint, summarized in table 2, demonstrates hover performance better than or on par with that of research grade quadrotors [10, 12].

Though the full system mass was largely dominated by the control electronics and battery, the lighter frames generated by the cutting processes consumed notably less power in hover. With a lighter control platform (such as the 2g controller from [13]), this benefit would be further magnified.

However, the folded frames were also less efficient, consuming more power per unit mass than those produced by conventional technologies. The relaxed tolerances of the folded structures resulted in misaligned motors; these then generated antagonistic thrust, increasing the required ratio of power consumption to effective lift. Furthermore, the more compliant frames leached energy from lift into the bending modes of the frame.

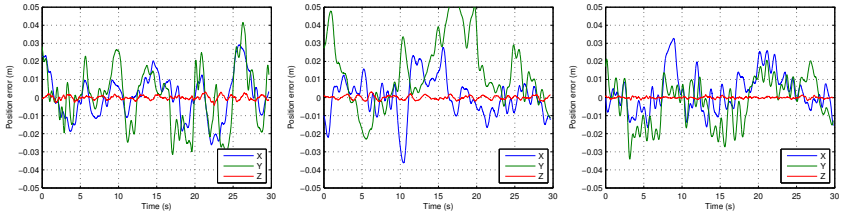


Fig. 5 The position error during hover for folded quadrotors, frame B (left), frame C (middle) and the laser-cut quadrotor, frame A (right) demonstrate performance comparable to commercial options.

The impact of the of motor misalignment is further evident from the control signals to the four quadrotor motors. Figure 6 shows the motor PWM values during hover for the same quads as in figure 5. Though the position errors are similar for these quadrotors, the relative motor PWM values vary significantly. The large difference in commanded output between the four motors is due to the torques required to compensate for the parasitic moments generated by off-axis alignment.

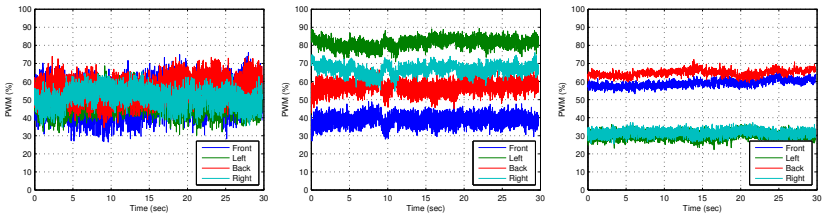


Fig. 6 Motor PWM during hover for folded quadrotors, frame B (left), frame C (middle) and the laser-cut quadrotor, frame A (right)

To compare the dynamic response of the different frames, we gave a step input of 0.15 m in the X-direction and looked at the resulting position response. These are shown in figure 7 and summarized in table 2. The lighter frames are more agile – the constant control effort exerted by the motors has a greater effect given lower mass. Frame A, being made of a more flexible 2D sheet, displays greater amplitude oscillations than the stiffer folded structures.

6 Conclusions and future work

This paper introduced a new system to simplify the process of robot design. To evaluate the system, both the process and the resulting robot must be considered. The data presented above (and summarized in table 2) displayed notable advantages of this system along both fronts.

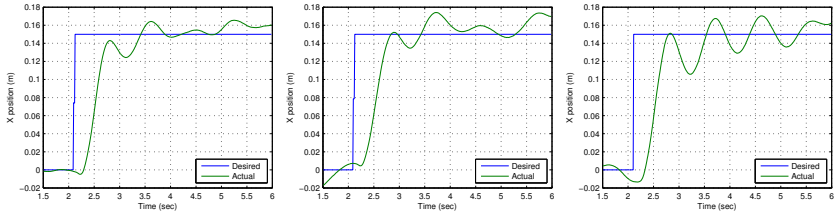


Fig. 7 The step response for a 0.15 m step in X-direction for frame B (left), frame C (middle) and frame A (right) demonstrates the agility and accuracy of the airframes. The position error about the setpoint can be further improved with a custom designed trajectory for each frame.

The combination of the folded plastic fabrication method with the Python script-based modular design environment significantly reduced both design cost (both of raw materials and necessary hardware and software tools) and time (to both generate and fabricate complete designs). Combined, these benefits greatly ease the viability of iterated design-build-test cycles, critical for robot design optimization. Along with the simpler design environment, this system can bring the process of robot creation into the hands of non-experts.

This work identified areas where this system can be improved. Most notably, future work will focus on simplifying the user interface. This system reduces complicated robot design down to assembling component modules; a user-friendly interface can confine engineering expertise exclusively to module generation, thus eliminating any specialized skills required to synthesize robots. The system will also be expanded to combine the modular design of the both mechanical and electrical subsystems into a combined framework, further simplifying the process.

The robots generated by this system display benefits over those generated by current processes. In the case of mobile robots and especially MAVs, minimizing system weight while maintaining robustness is a key design goal. It is in this that the folded plastic process particularly shines. The high strength, low weight mechanical frames enabled reliable, high performance MAVs comparable to existing technologies, though at the nominal expense of efficiency due to lower overall stiffness. These parameters can be tuned by system design and material selection, and so future design effort can work towards ameliorating compliance in the structure.

In addition to robot design, the system presented in this paper is also valuable in determining avenues of future robotics research. Conventional quadrotor control algorithms assume symmetric parallel motor axes; imperfect alignment and folding tolerances can lead to inefficient or poorly controlled flight. Though mechanical design effort can attempt to lessen such variations, an important orthogonal direction for future research revealed through this work is to develop robust control algorithms that allow robots to identify their dynamics and self-tune controller gains.

Supplemental multimedia

A video of the work presented in this paper can be accessed from:
`http://people.csail.mit.edu/mehtank/ISRR2013/ISRR2013.mp4`

Acknowledgments

This work was funded in part by NSF grants 1240383 and 1138967, for which the authors express thanks.

References

1. Jodi Forlizzi and Carl DiSalvo. Service robots in the domestic environment: a study of the roomba vacuum in the home. In *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, pages 258–265. ACM, 2006.
2. George V Lauder. Flight of the robofly. *Nature*, 412(16):688–689, 2001.
3. AFRON design challenges - african robotics network (AFRON). `http://robotics-africa.org/afron-design-challenges.html`. [Online; accessed 01-Feb-2014].
4. Samuel M Felton, Michael T Tolley, Cagdas D Onal, Daniela Rus, and Robert J Wood. Robot self-assembly by folding: A printed inchworm robot. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 277–282. IEEE, 2013.
5. Cagdas D. Onal, Robert J Wood, and Daniela Rus. Towards printable robotics: Origami-inspired planar fabrication of three-dimensional mechanisms. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 4608–4613. IEEE, 2011.
6. C. D. Onal, M. T. Tolley, K. Koyanagi, R. J. Wood, and D. Rus. Shape memory alloy actuation of a folded bio-inspired hexapod. In *in ATBio Workshop, IROS*, 2012.
7. C. D. Onal, R. J. Wood, and D. Rus. An origami-inspired approach to worm robots. *Mechatronics, IEEE/ASME Transactions on*, 18(2):430–438, 2013.
8. Thingiverse - digital design for physical objects. `http://www.thingiverse.com/`. [Online; accessed 01-Feb-2014].
9. Vicon. `http://www.vicon.com/`. [Online; accessed 01-Feb-2014].
10. Nathan Michael, Daniel Mellinger, Quentin Lindsey, and Vijay Kumar. The GRASP Multiple Micro-UAV Testbed. *IEEE Robotics & Automation Magazine*, 17(3):56, September 2010.
11. Kmel robotics. `http://www.kmelrobotics.com`. [Online; accessed 01-Feb-2014].
12. Alex Kushleyev, Daniel Mellinger, and Vijay Kumar. Towards A Swarm of Agile Micro Quadrotors. In *Proceedings of Robotics: Science and Systems*, 2012.
13. Ankur M Mehta and Kristofer SJ Pister. Warpwing: A complete open source control platform for miniature robots. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 5169–5174. IEEE, 2010.