On-demand printable robots

Ankur Mehta
mehtank@csail.mit.edu

Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Computational problem?

There’s an App For That
Physical problem?

- “There’s a robot for that.”
Robot compiler

**Vision:**
Autonomously generate designs for personal robots

**Big picture goal:**

```
$ vim myrobot.rbt
  “I want a robot to make me breakfast”
$ make myrobot
  Parsing specification ...done.
  Determining behaviors ...done.
  Generating mechanisms ...done.
  Assembling components ...done.
  Printing ...done.
  Success!
```
Challenges

- Subsystem integration
Application space

- Users

- Constraints
Related work

- **Rapid fabrication**
  - 3D printing: Mavroidis et al., J. Mech. Des. ’01; Lipson et al., J. Mech. Des. ’05; Rossiter et al., SPIE Smart Struct. and Mat. ’09; and others
  - 2D printing: Shimoyama et al., Control Systems ’93, Hoover et al., ICRA ’08; Onal et al., Trans. Mechatronics ’13; and others
  - Requires domain specific expertise and CAD tools
Related work

• Modular design of robots
  – Farritor et al., ICRA ’96; Hornby et al., IEEE Trans. Robotics and Automation ’03; Davey et al., IROS ’12; Romanishin et al., IROS ’13; and others.

  – Employs expensive custom components with limited configurations
Goal

• Enable the on-demand creation of custom printable electromechanical systems
  – Design intuitively
  – Fabricate cheaply
  – Iterate rapidly
An Expedition in Computing

... for compiling printable programmable machines
Robot compiler: design flow

- **Input**: high level task specification
  “I want a robot to play chess”

- **Output**: functional specification
  - Mobility across a 50cm x 50cm square with obstacles
  - Ability to move pieces
  - Knowledge of chess rules
Robot compiler: design flow

- **Input**: functional specification
- **Output**: modular definition
  - Chassis
  - Motorized legs
  - Gripper
  - Power, processing, communication
  - Chess logic
Robot compiler: design flow

- **Input**: modular definition

- **Output**: parameterized robot design
  - Mechanical template
  - Electromechanical transducers
  - Electrical connectivity
  - Algorithms
Robot compiler: design flow

- **Input**: parameterized robot design

- **Output**: fabrication plan
  - Fold pattern
  - Circuit layout and specific components
  - Executable drivers, software, and user interface
Robot compiler: design flow

- **Input**: fabrication plan
- **Output**: physical device
  - Folded robot with circuit, firmware, and software

- Task definition + decomposition
  - Modular composition
    - Unified mechanism
      - Co-design implementation
        - Parameterized model
          - Realization
            - Fully specified design
              - Fabrication

Robot
Robot compiler: design flow

- **Input:** physical device
- **Output:** mission accomplished!
Current system

- **Input**: structural specification
- **Output**: robot designs
  - Directly fabricable mechanical drawings
  - Wiring diagram
  - Arduino firmware
  - Android UI
A real world example

- **Request:** “We need a robot …
  In ascii art form it is:
  
  \[
  \begin{array}{c}
  \text{x} \\
  \hline
  \hline
  \hline
  \text{t}
  \end{array}
  \]

  where \(|\) is a rotational axis, and \(t\) is the tool.”

- “rotating joint” $\rightarrow$ ActuatedHinge
- “the tool” $\rightarrow$ ActuatedGripper
- **Arm =** ActuatedHinge + ActuatedHinge + ActuatedGripper

~1 hour later…

Ankur Mehta
mehtank@csail.mit.edu
INTEGRATED CO-DESIGN
On-demand printed robots

Compile, print and run!
Software defined hardware

- Parameterized, process independent representation
- Object oriented code abstraction of design modules
- Algorithms to generate fabricable drawings
Library of components

- Mechanical building blocks
- Electrical building blocks
- Software building blocks
- UI elements

```c
bool setSpeed(int servoNum, int speed);
void calibrateServo(int servoNum);
```

![Mechanical building blocks](image1)

![Electrical building blocks](image2)

![Software building blocks](image3)

![UI elements](image4)
Composition algorithms

Hierarchical composition across subsystems
Computation for combining functionality of modules
Hierarchical design composition

• { Subcomponents }

• { Parameters }
  – Inherited by default
  – Can be specified by imposing constraints
Hierarchical design composition

```python
from api.Component import Component
from api.Edge import Flex
from Beam import Beam

finger = Component()
finger.addSubComponent("beam1", Beam)
finger.addSubComponent("beam2", Beam)

finger.setSubParameter("beam1", "length", 60)
finger.setSubParameter("beam1", "beamwidth", 10)
finger.setSubParameter("beam1", "shape", 3)
finger.setSubParameter("beam1", "angle", 45)

finger.setSubParameter("beam2", "length", 40)
finger.setSubParameter("beam2", "angle", 45)

finger.add("beam1")
finger.connect(("beam1", "topedge"),
               ("beam2", "botedge"),
               Flex())
```
Full system
Let's make robots!
Design paradigm

- User selects desired geometries with degrees of freedom and their ranges.
  - Standard user: assembles building blocks (GUI)
  - Expert user: can generate building blocks (Python)
ROS Lab design environment

- Graphical programming language
- Automatic modular code generation

```python
self.attach(('brain', 'core', 'botright'),
            ('legpair', 'front', 'topright'), Fold(-180))
```
Robot design

\[ \text{Seg} = \text{left wheel} + \text{core} + \text{right wheel} + \text{tail}. \]
Designed robots
Designed robots

Ant

Brain
- EBrain
  - Firmware
  - Arduino
  - Mount
- Beam
- Cutout

LegPair
- FixedLegs
  - Beam
  - Leg
- MovingLegs
  - Motor
  - FourBar
  - Leg
    - UI element
    - Firmware
    - Servomotor
    - Cutout
    - Beam
FABRICATION-INDEPENDENT DESIGN
Robot fabrication: 3D print
Robot fabrication: 3D print
Robot fabrication: 3D print

Pros: Minimal user assembly / post fabrication processing
      Strong rigid bodies

Cons: Long fabrication time
      Minimal compliance
Robot fabrication: cut-and-fold
Robot fabrication: cut-and-fold
Robot fabrication: cut-and-fold
Robot fabrication: cut-and-fold

Pros:  Cheap and universally fabricable  
          Controlled compliance

Cons:  Requires post-process assembly  
          Weak structures

Ankur Mehta  mehtank@csail.mit.edu
Robot fabrication: self-folding
Robot fabrication: self-folding
Robot fabrication: self-folding

Pros:  Fast fabrication time with low user-driven assembly  
        Both rigid and flexible elements

Cons:  Large tolerances  
        Design compromises
One design, four robots
New hybrid processes
APPLICATION PROGRAMMING
Robot programming

- Autonomous behaviors
  - Feedback controllers
  - Task driven programs
Autogenerated user interface

Motor mount

Motor

PWM driver

UI Joystick

Drive unit

Left wheel

Right wheel

Ankur Mehta
mehtank@csail.mit.edu
Programmed autonomy

- Light sensor
  - Left speed
    - Left wheel
  - Right speed
    - Right wheel
Current work

- Verification & validation
  - Kinematic simulations
  - REACT programming
  - Derive behavior from designs
  - Ensure that the output matches the desired input
Algorithmic questions

• Provably correct composition for integrated designs

• Optimality bounds for composed designs
  – Functional decomposition
  – Fabrication specifications

• Efficiency metrics
  – Computation
  – Task completion
Future work

- How do we identify which mechanisms will accomplish desired functions?

Task definition + decomposition

- Modular composition
  - Unified mechanism
  - Co-design implementation

- Structural constraints
  - Parameterized model
  - Realization

- Behavioral constraints
  - Fully specified design
  - Fabrication

Operation

Robot
Future work

- How do we identify which mechanisms will accomplish desired functions?
- How do we identify what functions are required for a given task?
- Robot programming language
Robots!