
Peer reviewed version

Link to published version (if available):
10.1109/ICRA.2016.7487508

Link to publication record in Explore Bristol Research
PDF-document

This is the author accepted manuscript (AAM). The final published version (version of record) is available online via IEEE at http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=7487508. Please refer to any applicable terms of use of the publisher.

University of Bristol - Explore Bristol Research
General rights

This document is made available in accordance with publisher policies. Please cite only the published version using the reference above. Full terms of use are available:
http://www.bristol.ac.uk/pure/about/ebr-terms
Cubimorph: Designing Modular Interactive Devices for End-Users

Roudaut A.\(^1\), Krusteva D.\(^1\), McCoy M.\(^2\), Karnik A.\(^3\), Ramani K.\(^2\), Subramanian S.\(^1\)
\(^1\)University of Bristol, UK \(^2\)Purdue University, USA \(^3\)Lancaster University, UK

Abstract—We contribute toward the vision of programmable matter where interactive devices reconfigure in any shapes to fit a myriad of functionalities. To achieve this vision, we can benefit from the great breakthroughs that have been done in building modular robots. However, there is still a lack of understanding of user requirements in order to build devices that can not only reconfigure but also fully satisfy user needs. This is especially true in mobile contexts where many constraints need to be tackled. To address this issue, we present a design rationale that exposes eight user requirements, and we show that current modular designs to solve some requirements tend to violate some others. We then propose the concept of Cubimorph, a modular device in the form of a chain of cubes. We show an initial design and algorithm to address most user requirements and present three proof-of-concept prototypes demonstrating some of its aspects (turnable hinges, embedded touchscreens and miniature). We then discuss its limitations and how this work opens a more general discussion on the importance of creating more synergy between the field of Human Computer Interaction and Robotic.

I. INTRODUCTION

Human Computer Interaction (HCI) researchers have explored the benefit of exposing shape changing features in interactive devices [42][28][31]. This is useful in the case of mobile devices where the devices adapt their shape to the myriad of applications offered by these devices. For instance, when a user launches a game, the device transforms into a console-shape (Fig.1ab), thus creating a new affordance that satisfies the new functions and helps the user to interact.

Building such devices is however a hard challenge. Most devices consist in folding a material [14][31] but hardly reach high shape-resolution because the physical constraints of the material prevent folding the device in many ways [31]. Another way to achieve highly reconfigurable shape changing devices is to follow the vision of programmable matter [16][38] and to build devices made of self-actuated modules that reorganize themselves. In fact, many designs have been explored in the field of Robotic [25].

Unfortunately, using existing modular robots to create interactive devices poses significant problems when we add user experience to the equation: e.g. detachable modules work well on a large flat surface but not in mobile conditions [29][30]: modules cannot accommodate interactive elements on all the faces due to placement of actuators [27][33]; algorithms used for reconfiguration are not adapted for a mobile scenario as they consist in unfolding the device into a straight line before folding into a new shape [1][35].

There is a need for taking user experience into account when designing modular devices because user requirements dramatically change the way we tackle the design problem. Our paper demonstrates this by proposing a design rationale that exposes eight user requirements for designing modular handheld devices. We use the literature to demonstrate the difficulty of the design problem, and especially that trivial or existing solutions to address some of the requirements individually tend to be in conflict with others.

To address the user requirements, we present the concept of Cubimorph whose mechanical design consists of a chain of cubical modules linked together with a single hinge mounted on a turntable mechanism (to reposition it along the desired edge). Its algorithm, based on the probabilistic roadmap algorithm [20], consists in first creating an offline roadmap (a graph) of possible chains, and secondly searching the best path in this roadmap. Together our design and algorithm overcome most limitations of current designs.

We also present three proof-of-concept prototypes that demonstrate key aspects of Cubimorph: (1) the first has two 7.6x7.6cm modules which demonstrate the turntable hinge mechanism; (2) the second has two 7.6x7.6cm modules which demonstrates how to embed OLED touchscreens in the modules faces; and (3) the third had 16 2x2cm which shows how Cubimorph could be miniaturized in a near future.
While our work does not yet allow putting shape changing devices in end-users hands, we believe that it lays the foundation for it. We discuss how the current limitations of Cubimorph open new research questions and lay down a research agenda where HCI and Robotic researchers could benefit from their mutual skills.

II. DESIGN RATIONALE

Although there are many original modular devices, their designs do not always take into account the user experience. To demonstrate this, we considered an application scenario inspired by major related work on shape changing interactive devices [28][31][42]. From it, we derived eight requirements for creating modular handheld devices. We then show that embedding these requirements into a modular design is a complex task by showing how existing solutions to address some of the requirements tend to violate others.

Use-case: Shelly uses her phone to arrange a rendezvous. As she hangs up, she launches the map application. The device shape-shifts in a surface-like shape. Shelly uses the seamless surface to pinch and zoom the map and locate the rendezvous. As she closes the map, the device shape-shifts in a form easier to put in her pocket. In the train, Shelly launches a game and the device shape-shifts in a console-like shape, curling its edges to facilitate grasping and popping up joysticks to ease the manipulation of characters.

A. User requirements

R1. The device must be self-contained to avoid modules to fail or be lost, and to allow the device to reach any point in space without the user interventions.

R2. Modules must have free-faces to enable the placement of interactive equipment (e.g. displays, touch sensors).

R3. Modules must be dockable to create seamless interactive surfaces when placed next to each other. It also increases the device robustness as it forms a compact assembly.

R4. Modules should have minimum two degrees of freedom so that the assembly could transform into any 3D geometry. It means that a module should be able to rotate around its edges touching the adjacent module.

R5. The device must form a stable structure to avoid the device to disassemble once the transformation is done, especially if the user is then manipulating the device.

R6. The transformation should be safe, i.e. it should not hit the user’s hand. Additionally during the reconfiguration. Additionally the device should have a safe position in the hand to avoid falling during the transformation.

R7. The transformation should happen in a constrained-space, i.e. the size of the device must be minimized at any time to ensure a reconfiguration on the fly, directly in the user’s hand, and prevents adopting large shapes.

R8. The transformation should be performed in a reasonable amount of steps to enable fast interaction.

As one can observe, some requirements are more of a mechanical nature as they relate to the modules shapes and the way they assemble. In contrast, some requirements are more related to the transformation mechanism. In the following we present how previous related work have addressed these two types of requirements.

B. Mechanical requirements (R1 to R5)

R1 (self-contained): a powerful feature of most robot designs is the docking interface that allows modules to attach or detach to other modules. For instance, Changle [30] use magnets and M-Blocks [29], Pebbles [11], EmCube [2], Claytronic [13] or Catoms [8] use electromagnets for docking. However, this actuation mechanism violates R1 by draining power of the mobile device and allowing all hinges to break away when there is no power. Other solutions involve mechanical docking. This is the case of Conro [6], Polybot [46] or MTRAN [19] whose modules can attach or detach from another one through a connection mechanism made of pins and holes. In contrast, CKBot [27] and Superbot [33] rely on permanently attached modules.

R2 (free-faces): CKBot [27] and Superbot [33] offer two free faces that could fit interactive elements. Roombot [36] and Molecule [50] rely on cubical modules that rotate along their diagonals. Their faces are used for docking in order to create lattice assemblies, but used as a chain it would let four faces free among six to place touchscreens. Unfortunately for Conro [6], Polybot [46], MTRAN [19], or ATRON [18], the usable area is reduced drastically due to the hinge mechanism (clamps on each hemisphere sides).

R3 (dockable): a way to satisfy this requirement is to have cubical modules. This is the case of M-Blocks [29], Pebbles [11], Em-Cube [2], Roombot [36] and Molecule [50]. Some designs involve spherical (ATRON [18]) or cylindrical modules (Claytronic [13], Catoms [8] or Octabot [34]). However such geometry, even a lattice positioning of the modules, will result in the creation of gaps that prevent the creation of a seamless surface, which could be problematic for user interaction. Some geometry lends to more efficient packing, which results in continuous planar surfaces, e.g. hexagonal modules [4]. This is the case of Fracta [26], Gear-Type-Unit [40] and Metamorphic [8].

R4 (2DOF): achieving 2DOF can be done with a shaft between modules. Conro [6], Polybot [46], MTRAN [19] use this approach. The rotation requires a linkage between two modules centers, which need to pass through three of the module faces. It also requires the faces to be rounded for smooth rotations. A potential challenge is that the mechanisms should not protrude out of the modules, as this might prevent them from achieving 2DOF, have free-faces (R2) or being dockable (R3). An alternate approach is to move the rotation axis at the center of each module (on their diagonals) such as in Roombot [36] and Molecule [50].

R5 (stable): if modules are only connected (at most) to two modules, the resulting shape may not be structurally sound, as neighboring modules are not necessarily attached. A solution would be to use a latching mechanism such as in Conro [6], Polybot [46] or MTRAN [19] but as said earlier it would prevent to have free-faces (R2), except if the latching is done magnets like in Claytronic [13] or Catoms [8]. Another solution could be to entangle units to create structural strength, but this solution might increase the complexity of the algorithm to a great extend.

1 Modules could also find others, but this is impractical in a mobile context.

2 Diagonally connected squares or cube can create any 2D or 3D shapes by folding [7][15].
C. Transformation requirements (R6 to R8)

R6 (safety): existing algorithms can transform a chain into another while avoiding collisions without obstacles (e.g. [24]) or with obstacles (e.g. [44]). Obstacles could represent anything, and especially in our case the user’s hand or the other users in the immediate surrounding of the user. In this case, however, the links between modules are reduced to points and are allowed to pass through each other, and thus it does not model constraints of the real world. In fact, adding such constraints significantly impact the complexity of the algorithms as reported by Trinkle et al. [41]. Meanwhile, some researchers have proposed solution for discrete motion planning with obstacles that is known to be a hard problem scaling poorly with increasing numbers of modules [5][32].

R7 (constrained-space): most algorithms do not take into consideration a constrained space and are specific to some design such as MTRAN [48] Pebbles [12] or Catoms-like structures using hexagonal modules [47] [43]. More generic algorithms such as Mottein [35], protein-folding algorithm [1] or [32] for 2D chain, also fail to validate this requirement as they consist in applying repulsive forces to unfold the chain and thus generally tend to go through a straight line configuration of modules. A solution is to consider the problem of constrained space as a particular form of reconfiguring with dynamic obstacles [9], but this algorithm is specific to locomotion planning (e.g. to control cranes).

R8 (amount of steps): decreasing the amount of transformation steps is obviously a very common issue in Algorithmic, especially for path planning in high-dimensional configuration spaces [23]. Simple algorithms are fast if the chains are similar, but, as soon as they are different it must cope with multiple self-collisions that are increasing the amount of steps. This becomes even more complicated in our case, as we are also deal with collisions with the user’s hand and collisions with a constrained space. Other solutions are non-deterministic such as the simulated annealing algorithm [22] that is often employed to unfold protein structures. Instead of reconfiguring the chain into a final one, the algorithm finds an acceptably good solution in a fixed amount of time. It runs multiple iterations in which it considers a neighboring chain of the current one, and decides to move the system to the new state or not depending on some specific cost function. Hybrid methods, such as probabilistic roadmap, uses the power of probabilistic methods to precompute possible configurations, as well as the efficiency of deterministic algorithms that are only used to reconfigure nearby chains. This is a widely used method for modular reconfiguration [5] and the one we implemented.

III. CUBIMORPH MECHANICAL DESIGN AND PROTOTYPES

Cubimorph is a first attempt to fulfill our requirements. Its mechanical design answer R1, R2, R3, R4 and R5. We will show later how its algorithm addresses other requirements.

A. Principle

With all the conflicting requirements in our rationale, the ideal actuation design needs to allow rotation along the edges using reconfigurable hinges which keep the modules connected at all times. Figure 2 illustrates our Cubimorph design that answers these conflicting requirements.

A chain of modules (R1): Cubimorph is a chain of cubical modules. Modules have five possible positions relatively to their neighbor: straight, top, bottom, back or front. Relying on independent hinges per edge requires complex docking interfaces and does not fully answer R1. We eliminate this need by using a single actuated hinge, which keeps the modules always connected at all time.

Modules shape (R2, R3): each module is cubical, thus enabling the placement of interactive elements on them as well as allowing a flush docking when placed together.

Actuation mechanism (R4, R2, R3): the key feature that allows 2DOF is that the hinge connecting two neighbor modules is mounted on a turntable mechanism. This turntable repositions the hinge along the desired edge before actuation (Fig. 3). This mechanism also does not protrude on the side of the modules, thus increasing the real estate of each module face (R2) and allowing for flush docking (R3).

Chain stability (R5): the turntables are actuated by a worm drive so they lock when not powered. Any two cubes in a straight configuration (turntables face-to-face) will lock in place by simply positioning the turntable hinge along any diagonal of the cube face. This allows the device to avoid a slinky like feel without consuming power when inactive or needing a complex latching mechanism.

Figure 3 shows how to flip the right module of Figure 3b on the top (it can currently only flip backward): (1) rotating the hinge 180° to put the two modules to their default position (Fig.3.b); (2) rotating the internal assembly so that the hinge faces the desired edge; (3) rotating the hinge 180° (Fig.3.c). Note that, during the repositioning of the hinge, the external case of the module does not rotate along with the internal assembly. Thus, in order to place a module in a desired position, the module must first reach the straight position.

B. Mechanical design (Fig. 2)

The mechanical design is composed of cubical modules, each with a single hinge mounted on a turntable. The turntables are actuated by a worm drive so they lock when not powered. Any two cubes in a straight configuration (turntables face-to-face) will lock in place by simply positioning the turntable hinge along any diagonal of the cube face. This allows the device to avoid a slinky like feel without consuming power when inactive or needing a complex latching mechanism.

Figure 3 shows how to flip the right module of Figure 3b on the top (it can currently only flip backward): (1) rotating the hinge 180° to put the two modules to their default position (Fig.3.b); (2) rotating the internal assembly so that the hinge faces the desired edge; (3) rotating the hinge 180° (Fig.3.c). Note that, during the repositioning of the hinge, the external case of the module does not rotate along with the internal assembly. Thus, in order to place a module in a desired position, the module must first reach the straight position.

C. Transformation requirements (R6 to R8)

R6 (safety): existing algorithms can transform a chain into another while avoiding collisions without obstacles (e.g. [24]) or with obstacles (e.g. [44]). Obstacles could represent anything, and especially in our case the user’s hand or the other users in the immediate surrounding of the user. In this case, however, the links between modules are reduced to points and are allowed to pass through each other, and thus it does not model constraints of the real world. In fact, adding such constraints significantly impact the complexity of the algorithms as reported by Trinkle et al. [41]. Meanwhile, some researchers have proposed solution for discrete motion planning with obstacles that is known to be a hard problem scaling poorly with increasing numbers of modules [5][32].

R7 (constrained-space): most algorithms do not take into consideration a constrained space and are specific to some design such as MTRAN [48] Pebbles [12] or Catoms-like structures using hexagonal modules [47] [43]. More generic algorithms such as Mottein [35], protein-folding algorithm [1] or [32] for 2D chain, also fail to validate this requirement as they consist in applying repulsive forces to unfold the chain and thus generally tend to go through a straight line configuration of modules. A solution is to consider the problem of constrained space as a particular form of reconfiguring with dynamic obstacles [9], but this algorithm is specific to locomotion planning (e.g. to control cranes).

R8 (amount of steps): decreasing the amount of transformation steps is obviously a very common issue in Algorithmic, especially for path planning in high-dimensional configuration spaces [23]. Simple algorithms are fast if the chains are similar, but, as soon as they are different it must cope with multiple self-collisions that are increasing the amount of steps. This becomes even more complicated in our case, as we are also deal with collisions with the user’s hand and collisions with a constrained space. Other solutions are non-deterministic such as the simulated annealing algorithm [22] that is often employed to unfold protein structures. Instead of reconfiguring the chain into a final one, the algorithm finds an acceptably good solution in a fixed amount of time. It runs multiple iterations in which it considers a neighboring chain of the current one, and decides to move the system to the new state or not depending on some specific cost function. Hybrid methods, such as probabilistic roadmap, uses the power of probabilistic methods to precompute possible configurations, as well as the efficiency of deterministic algorithms that are only used to reconfigure nearby chains. This is a widely used method for modular reconfiguration [5] and the one we implemented.

III. CUBIMORPH MECHANICAL DESIGN AND PROTOTYPES

Cubimorph is a first attempt to fulfill our requirements. Its mechanical design answer R1, R2, R3, R4 and R5. We will show later how its algorithm addresses other requirements.

A. Principle

With all the conflicting requirements in our rationale, the ideal actuation design needs to allow rotation along the edges using reconfigurable hinges which keep the modules connected at all times. Figure 2 illustrates our Cubimorph design that answers these conflicting requirements.

A chain of modules (R1): Cubimorph is a chain of cubical modules. Modules have five possible positions relatively to their neighbor: straight, top, bottom, back or front. Relying on independent hinges per edge requires complex docking interfaces and does not fully answer R1. We eliminate this need by using a single actuated hinge, which keeps the modules always connected at all time.

Modules shape (R2, R3): each module is cubical, thus enabling the placement of interactive elements on them as well as allowing a flush docking when placed together.

Actuation mechanism (R4, R2, R3): the key feature that allows 2DOF is that the hinge connecting two neighbor modules is mounted on a turntable mechanism. This turntable repositions the hinge along the desired edge before actuation (Fig. 3). This mechanism also does not protrude on the side of the modules, thus increasing the real estate of each module face (R2) and allowing for flush docking (R3).

Chain stability (R5): the turntables are actuated by a worm drive so they lock when not powered. Any two cubes in a straight configuration (turntables face-to-face) will lock in place by simply positioning the turntable hinge along any diagonal of the cube face. This allows the device to avoid a slinky like feel without consuming power when inactive or needing a complex latching mechanism.

Figure 3 shows how to flip the right module of Figure 3b on the top (it can currently only flip backward): (1) rotating the hinge 180° to put the two modules to their default position (Fig.3.b); (2) rotating the internal assembly so that the hinge faces the desired edge; (3) rotating the hinge 180° (Fig.3.c). Note that, during the repositioning of the hinge, the external case of the module does not rotate along with the internal assembly. Thus, in order to place a module in a desired position, the module must first reach the straight position.
B. Proof-of-concept prototypes

We built three proof-of-concept prototypes that demonstrate specific aspect of Cubimorph. We used a ProJet 5000 multijet 3D printer to create the mechanical parts.

1. Hinge prototype (Fig.5.a): it is made of two modules of 7.6x7.6cm. Each module has two parts, the external case that is a hollowed out cube, and the internal assembly that is a cylindrical structure. This structure is shared between two adjacent modules and rotates when in default position (Fig.3.b). The internal assembly (Fig. 4) ensures that the chain is always connected and consists of:
   - 2 hinge motors (Futuba s3115), one in the right side module and one in the left to actuate the hinge gears.
   - 2 hinge gears, one for each hinge motors, transmit rotation to the hinge printed with a gear pattern.
   - 1 turntable motor that is responsible for rotating the turntable gear. We used ROB-08910 DC geared motors.
   - 1 turntable gear, placed on top of the turntable motor, transmits rotation to the turntable worm drives.
   - 1 turntable worm drive transmits rotation to the entire internal assembly.
   - 1 metal protrusion makes contact with the limit switches of the external case.
   - 4 limit switches on each internal sides of the external case to detect the rotation of the internal assembly.

   ![Figure 4. Cubimorph mechanical design.](image)

2. Touchscreens prototype (Fig.5.b): our prototype is made of two cubical modules of 7.6x7.6cm whose faces all accommodate a µOLED-128-G2. Note that the size of the OLED does not match exactly the size of each face. The reason is that the µOLED-128-G2 is the only display with embedded driver we could find, however it is possible to embed display that fit the exact size of the modules.

3. Miniature prototype (Fig.5.c): researchers have demonstrated that shrinking the size of modules is possible with progresses in miniaturization. In Peebles [11], each module measures 12mm and weight 4g and is capable to latch to neighbors using electromagnets. Yoshida et al. created modules capable of 1DOF that measure only 2cm and weight 15g each [49] using shape memory alloys. Recent advances in piezoelectric motors [37] could allow us creating 2DOF using our design. Such motors have a smaller torque but their reduced size and weight can lighten the entire assembly and allow lifting multiple modules. To demonstrate how such technology could be used, we 3D printed our design at a smaller scale in a way that it could accommodate these piezoelectric motors (16 2x2cm modules). This prototype, even though non-actuated, shows what we could build in a near future and strengthens our vision that future devices could be made of many modules.

Note that the hinge and turntable’s rotation are designed for running electrical connections between two cubes. The shown miniature prototypes already contain spaces for wiring between the cubes in parallel with the hinge.

![Figure 5. Proof-of-concept prototypes: (a) the hinge version demonstrates the turntable hinge mechanism; (b) the touchscreens version shows how to embed OLED in it and (c) the miniature version demonstrates a device with an estimation of module size to fit last advances in piezoelectric motors.](image)

IV. CUBIMORPH ALGORITHM

We show how our reconfiguration algorithm addresses the requirements R6, R7 and R8.

A. Principle

Our goal is to transform a chain made of n modules into a shape while answering to the user requirements. The output is a collection of ordered chains, which represents a discrete path from the initial chain to a chain representing the final shape. Note that we treat the problem in a discrete way (angle between modules is 0º or 180º except when a rotation is performed). Our algorithm is based on the probabilistic roadmap algorithm [20], which has been shown to compute in a relatively small amount of steps (R8):
1. Offline roadmap generation: it creates a set of chains containing the initial and final chain, and randomly generated non-self-colliding chains. It then creates a graph with chains as nodes. An edge is then added between two nearby nodes following this process: a simple local planner (simple but fast algorithm) computes the reconfiguration between the two chains. If the algorithm takes more than a certain amount of steps, it is stopped and no edge is added. Otherwise an edge is added.

2. Online graph search: to go from one chain to another, the algorithm then searches in the graph the shortest path between the node representing the device initially and the node representing the final shape.

B. Adaptation to fit requirements

- Prevent device from falling (R6): we create our algorithm with the requirement that the user holds an extremity of the chain at any time. This prevents the device to fall from the hand when reconfiguring. It is also possible to run the reconfiguration on a surface (e.g. a table). Note that the position of the user’s hand can be easily retrieved by the interactive sensor placed on each module faces.

- Prevent chains from colliding with user’s hand and constrained space (R6, R7): to create the roadmap, the algorithm generates non-self-colliding chains, and also takes into account the hand while eliminating chains that don’t fit within a bounding box around the hand or that collide with it. The size of this box can be parameterized.

- Detecting nearby chains: as with the classic algorithm, we check if an edge exists between two nodes only if those nodes are nearby. Such a case happens if the number of rotations needed to go from one to another is less than a certain value (parameter of the algorithm). To compute this difference, we encode our chains as a vector of number from 1 to 5 (1: straight, 2: top, 3: bottom, 4: back, 5: front) that represents the relative position of each module to its previous neighbor. E.g. a straight chain of 5 modules with the last module placed on the bottom of the previous one is [1,1,1,3]. By making the difference between two vectors and counting the non-zero values, we know how many rotations are needed to go from the first chain to the other one and thus decide whether or not the chains are nearby.

The local planner checks user’s hand collision and allows constrained space to inflate (R6, R7): to transform a node into another, the local planner rotates each module one by one in its correct position, thus creating a sequence of chains that represent the path between one chain to another. This simple planner computes in a constrained space. It means that each time a collision is detected with the bounding box or the user’s hand, the algorithm first resolves it. For example, if, when rotating the middle of a chain the last module collides with the hand, the algorithm first rotates the colliding module so that its new position will not create a collision when performing the original rotation. As resolving a collision can lead to collisions, the planner uses recursive calls. If the algorithm fails to compute efficiently (the number of reconfiguration reaches a certain value), the bounding box of the constrained space increases and the planner starts again, or abandon if the box size reach a value. In this case it does not result in creating an edge.

Each edge is tagged with a tuple: this tuple corresponds to the output properties of the local planner that consist of the number of reconfigurations necessary (number of reconfiguration), the overall torque required, and the area of the bounding box used to perform the reconfiguration.

- Parametrizable path search (R7, R8): because the edges are tagged, the algorithm can search the shortest path in the graph but also take into account other parameters. E.g. one could tradeoff number of steps over space. The algorithm can also be parameterized so that the search produces a path that fits the maximum torque that the design permits.

C. Removing invalid chains

We also implemented mechanisms to avoid deadlocks such as blocked modules or tangled chains.

- Blocked modules: a module is blocked when it directly faces a module to which it is not connected (i.e. this unconnected module occupies its “straight” position). In this case the module has no possible degrees of freedom. Figure 7 illustrates such a case in which the module j is blocked after the rotation. A way to avoid this situation is to search the pair of modules that are connected but not neighbors. For each of these pairs (mi,mj) if the segment [i−1, i] and the segment [i,j] are parallel then the module mi is facing module mj. We ensure this case never happens by eliminating blocked chains and forbidding any rotation that would lead to such a case.

- Tangled chains: a tangled chain can lead to deadlocks. E.g. in Figure 8 it could be problematic to move the first four modules of the chain after rotation of the module mi...
because most possible configurations are too large to pass through the knot. While there might be some cases where a tangle can be removed, untying a chain is a hard problem and we rather choose to ensure that tangling never happens.

We used the Khatib’s algorithm [21] (inspired by Taylor [39] used for detecting knots in proteins). It considers 3 consecutive module i−1, i, i+1 and checks if the triangle formed by these points is intersected by any segment constituting the chain (a segment starts and ends at the centers of two consecutive modules). When no line intersects the triangle i−1, i, i+1, the algorithm removes the point from the chain. After multiple iterations, if there are no tangles, the algorithm eventually reduces the chain to a single line between the first and last module. Otherwise, the tangled modules persist and prevent the formation of a line. Fig. 9 illustrates the result of the multiple iterations of our algorithm on a tangled and an untangled chain.

D. Possible optimizations

Collision detection: our method for collision detection is basic: we perform iterations that discretely rotate the moving subchain and check collisions. While this method is functional, it impacts the overall computation time. Other proven collision methods can accelerate this process such as the Gilbert–Johnson–Keerthi distance algorithm [10].

One chain fitting a given shape: finding a chain that fits into a shape is a surprisingly arduous challenge, especially in 3D, and our final and initial chains are currently hand written. The Motein algorithm however offers a solution [7]. It consists in voxelizing the shape and splitting each voxel in eight subvoxels. The authors then show that it is always possible to find a Hamiltonian path between those subvoxels (a path that connect all subvoxels) (Fig.10). Such method could be use to automate the creation of chains corresponding to multiple final and initial shapes.

Multiple chains fitting a shape: an idea to increase the chance of findings valid paths in the graph is to attribute several chains to a shape rather than just one. A method to achieve this would be to degenerate the original shape several times (e.g. blurring it) and use the Motein algorithm on each degenerated versions to find new chains that will populate the roadmap graph. The reasoning behind this idea is that, in our specific scenario, it is possible to sacrifice the accuracy of the shape to a certain extent. For instance it is possible to generate chains that are 90% accurate that would not change the overall user perception of the shape.

To achieve a measure of accuracy we need a measure of similarity between shapes. A lot of research has been focused on comparing shapes [45] and several methods have been proposed. Anders et al.’s method is a good candidate [3]. It consists in decomposing the space into concentric shells and sectors that emerge from the centroid of the shape. It then shoots rays from the center of the shape to points placed uniformly on the surface of a sphere. It then computes the radial distance and the polar angle to the ray intersections, and stores them into the appropriate bins (shells and sectors), thus creating a histogram of the points that immediately defines a decomposition of the shape. When only one sector is used, the histogram is made of concentric shells, in turn creating a rotation independent model that would be extremely useful in our case.

We implemented our algorithm using Open-Frameworks and Python as well as Maya library to perform transformation in 3D. We did not implement the possible optimization described in the previous sections but only the probabilistic roadmap core algorithm and thus we still have room for improvement. Figure 11 shows an example of output of our algorithm that reconfigures a rectangle shape to a sphere with a device of 48 modules.

V. LIMITATIONS AND FUTURE WORK

Although Cubimorph answers most user requirements, it still has some limitations, which need to be answered before creating high fidelity devices. We list them in the following and propose some solutions. We also present several research opportunities that could benefit for more synergy between the HCI research field and the Robotics.

A. Design and algorithm improvements

Continuous rotations (R3, R4): our design relies on discrete steps, i.e. the modules rotate at 180° but do not stop in an intermediary position. Allowing continuous movements could increase the number of possible geometry (R3). There are two challenges to allow continuous movements. The first is mechanical as this way of placing the modules decreases the robustness of the devices (R3). The second is algorithmic. Continuous motion planning with obstacles is a hard problem and existing solutions [44] do not take into account collisions (links between modules pass through each other) as adding real-world constraint significantly impacts algorithm complexity [41].
Chain vs. star (R5, R8): we have considered single chain devices but a problem is that the transformation complexity (R8) increases with the number of modules. It could become a serious issue when dealing with devices made of thousand of modules not only for the algorithm but also because of the increasing torque (R5). A way to fasten the process would be to use several chains emerging from a central unit.

Self-contained detachable modules (R1, R5): another way to address the problem of strength (R5) would be to allow module to detach from each other in order to balance the movable mass. As we said earlier, this could be unpractical in mobile scenario where the user is likely to lose some modules (R1). However, a solution could be to place the modules in an external envelope, which morph according to the configuration taken by the modules. Such extra layer could for instance benefit from recent advances in OLEDs technologies that allow creating screen of arbitrary shapes.

Number of steps/speed (R8): an obvious limitation related to requirement R8 is the speed of the reconfiguration. Although roadmaps algorithms tend minimize the number of step required to perform the transformation, it still requires too much step to be considered in a certain usage scenario, e.g. when the user requires fast reconfiguration. There are still improvements to do from an algorithmic point of view.

Actuator strength and structural integrity: servomotors with gear trains power the current prototypes but this can be simplified in future designs to strengthen the actuation. Advances in technologies like shape memory alloys could replace the two hinge motors with one drive. In addition, our prototypes have a current structural weak point at the double hinge (caused by needing to drive the device using small DC servomotors) but changing these motors to a more advanced alternative would afford twice as much room for the hinge pivots greatly reducing the stress on these joints.

B. Interactional improvements

The user interacts during the reconfiguration: our algorithm does not allow users to interact during the transformation but this is a promising extension. Because the faces of the modules can accommodate touchscreens it is possible to detect a hand position and to consequently adapt the reconfiguration. It is also possible to ask the user to adopt a specific grasp by displaying information on certain modules faces, e.g. to indicate where the device should be hold. In an algorithm point of view, certain modifications are required. In particular, our algorithm generates an offline roadmap that considers a static position of hand and the algorithm would have to regenerate the roadmap each time this one moves. This is feasible but might require more computation resources. A solution could be to perform the computation on a cloud in order to fasten the process.

The user chooses/models the shape: our current approach requires the designer of the device to decide which shapes the device can morph into. An extension would be to let the user decides of any shape. For instance, the user could capture a shape with the camera, e.g. using a depth camera in combination of a 3D reconstitution technique [17]. We could also let the user molds the device as he/she wishes, and to do that, there are more considerations to take for the requirements R5 and R6: we need actuation mechanisms that are strong enough to support user manipulations and we need safe manipulation. But to do this, more evaluation need to be performed in order to really understand how users are interacting with such devices, e.g. what forces and motions are applied to the devices, in order to better guide the design.

The designer chooses appropriate shapes: finally we believe that more investigations need to be done to understand what shapes make sense in certain scenarios. Doing so would help designers to choose shapes with better affordances for specific tasks. For instance there have been much work in Robotic to make actuated arms grab an object in a certain way, and it would be interesting to do the opposite, i.e. change the device shape so that it is grasped by the user in a certain way. Such a work would need deeper evaluations to understand the cognitive and psychomotor mechanisms behind the notion of affordance.

VI. CONCLUSION

We have presented Cubimorph, a concept of modular interactive device that changes its shape to fit functionalities required by end-users. We contributed a design analysis to create modular devices that can reconfigure but also fulfill user requirements. We made the first steps toward concretizing our concept with an initial mechanical design and its algorithm as well as three prototypes demonstrating key aspects of our design. Much work still needs to be achieved to put such devices in the end-user hands but we hope our research will create discussion between fields that could highly benefit from each other, HCI and Robotic.

REFERENCES

[1] Amato, N. M., Dill, K. A., Song, G. Using motion planning to map protein folding landscapes and analyze folding kinetics of known native structures. ACM Recomb02, 2-11


[28] Rasmussen, M. K., Pedersen, E. W., Pedersen, M. G., Hornbaek, K. Shape-changing interfaces: a review of the design space and open research questions CHI12, 735-744.

[29] Romanishin, J.W., Gilpin, K., Rus, D. 2013. M-blocks: Momentum-driven, magnetic modular robots. IROS’13, 3-7 Nov. 2013, 4288-4295,


[34] Shiu, M.C., Lee, H.T., Lian, F.L., Fu, L.C. Design of 2d modular robot based on magnetic force analysis. ICIT’08,1-6


[38] Sutherland, I. E. 1965. The ultimate display. IFIP Congress 506-508.


