A 3D Fax Machine based on Claytronics

Padmanabhan Pillai, Jason Campbell
Intel Research Pittsburgh
Pittsburgh, PA 15213

Gautam Kedia, Shishir Moudgal, Kaushik Sheth
Carnegie Mellon University
Pittsburgh, PA 15213

***Abstract*— This paper presents a novel application of modular robotic technology. Many researchers expect manufacturing technology will allow robot modules to be built at smaller and smaller scales, but movement and actuation are increasingly difficult as dimensions shrink. We describe an application — a 3D fax machine — which exploits inter-module communication and computation without requiring self-reconfiguration. As a result, this application may be feasible sooner than applications which depend upon modules being able to move themselves.**

In our new approach to 3D faxing, a large number of sub-millimeter robot modules form an intelligent "clay" which can be reshaped via the external application of mechanical forces. This clay can act as a novel input device, using intermodule localization techniques to acquire the shape of a 3D object by casting. We describe software for such digital clay. We also describe how, when equipped with simple inter-module latches, such clay can be used as a 3D output device. Finally, we evaluate results from simulations which test how well our approach can replicate particular objects.

I. Introduction

Consider a hypothetical scenario in the near future, where a paleontologist is searching in a remote location for interesting fossil remains. Having found an interesting bone fragment, she records the site information, using GPS and local surveying instruments, and takes numerous digital photographs to record the context of the find. She then takes an impression of the fragment in clay or plaster to facilitate its later reproduction for wider study as well as display in museums.

Given present technologies, the latency between discovery and access to fossil and object replicas will generally be measured in months. However, imagine in this future world, that the clay-like material used to take the impression is electronic in nature, comprised of a myriad of tiny modules which are capable of inter-module communication and computation. This intelligent clay measures its own shape and, by reflection, the shape of the embedded fossil fragment, and generates a digital representation of the fossil's three-dimensional structure.

Like other digital representations, the shape can then can be readily stored, manipulated, annotated, and transmitted. Within moments, the field scientist can share her find with colleagues across the globe. Those colleagues, using either similar clay-like material or traditional rapid-prototyping equipment, can then reproduce a facsimile of the fossil from the digital representation. This scenario is illustrated in Fig. 1.

Can such intelligent clay be created? Key considerations in answering this question include the physical and the electronic properties of the components:

1) Physical Characteristics: Suppose this material is composed of tiny, sub-millimeter particles that stick together, e.g.,

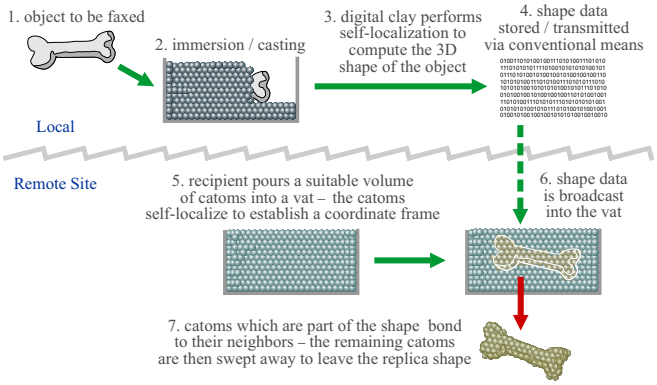

Fig. 1. An overview of the 3D fax scenario

spheres covered with a self-cleaning nanofiber adhesive [1], [2]. The combination of discrete parts and adhesion would permit the required malleability for this application. In an alternative implementation, the spheres could be covered with thin insulated plates, permitting the spheres to adhere to each other under software control through the establishment of an appropriate electric field on each plate.

2) Electronic properties: Suppose further that each of these particles is actually a micro-fabricated silicon sphere, its surface covered with electronic circuitry. A 300 micron (radius) sphere has a surface area of 1.13mm^2. Current embedded microprocessors can be fabricated in only 0.26mm^2 using mature process technology [3]. With specialized design and modest process improvements it is feasible that an entire system can be embedded on such a sphere, including microprocessor, memory, communications, power distribution, and sensor circuits.

An ensemble of 1000 to 100,000 of such spheres can be viewed as a modular robotic system taken to an extreme scale — both in terms of the minuscule size of each module, and the enormous number of cooperating units within the system. The realization of such high-module-count robotic systems is the goal of several major research endeavors, including our own work on claytronics [4]. In that broader effort we are working with many collaborators to develop hardware that can be scaled into the sub-millimeter range, mechanisms for actuation, communication, and power, and the software and programming methodologies required to control and handle distributed systems consisting of millions of units. Our long-term goal is general purpose programmable matter, which can serve as a new medium for human-computer interaction

and enable a vast realm of new applications (telepresence; immersive, tactile virtual environments; variable form input devices; and new forms of dynamic art and entertainment).

This paper concerns itself with a much less grandiose vision, and in particular with an application that may be feasible much sooner than general purpose programmable matter. Actuation and motion are challenging aspects of modular robotic systems, not only because of the size and required strength of the actuators but also because of the complexity of planning and controlling their use. For this paper we imagine programmable matter *minus such actuation capacity*, i.e., without the ability to self-reconfigure or move. Even without motion, inter-module communication still permits modules to localize themselves relative to their neighbors, and by extension to establish the geometry of the boundaries of the ensemble. In this paper we present techniques for doing so and describe their use for a 3D fax machine.

Although other technologies have been proposed to implement 3D faxing [5]–[7], the modular microrobot approach has certain distinct advantages, notably size and speed. In contrast to 3D fax machine approaches using serial (raster) input and output devices such as a laser scanner [8], [9] and 3D printer [10], programmable matter would acquire the 3D input shape and generate the 3D output shape in parallel. Thus, rather than taking hours to days the process could take seconds to minutes. Laser scanners and 3D printers also remain many times bulkier than the object being scanned/reproduced despite years of commercial development. Similar results could be achieved with a much smaller volume of programmable matter. The resulting increase in portability and speed could make 3D faxing more attractive than at present. Finally, depending upon implementation details (nanofiber adhesives vs. electrostatic attraction), the same robots may be capable of both input and output at different times, as well as frequent reuse.

Demonstrating operation of 3D faxing via our method requires thousands of functioning microrobot modules. Unfortunately our hardware prototyping efforts will be unable to produce such quantities in the near to medium term. Thus, in this paper we evaluate our 3D fax algorithms using a variety of simulation techniques. While simulations do not reveal the unexpected challenges and effects that physical testing can uncover, simulation results are nonetheless meaningful for validating high-level operation, verifying algorithmic feasibility, and providing useful data for those in the community studying software environments for programmable matter modules.

II. Overview

The process of remotely reproducing a facsimile of an object requires three phases: acquisition, transmission, and reproduction. In the first phase, the system senses the object and creates a digital representation of the visible, external structure. The shape information is then transmitted to the remote site. Finally, using the transmitted data, a facsimile of the object is reconstructed at the remote site.

We describe below how each of these phases can work in the context of programmable matter. In order to make our language more specific in the following, we adopt the module design, capabilities, and nomenclature envisioned by the Claytronics Project [4]. In this nomenclature, a connected volume of programmable matter is termed an *ensemble*, a word we use interchangeably here with *mass*. Individual units are termed *catoms*, and in this paper we also use *module* and *particle* to mean the same thing as catom.

A. Shape Acquisition

A variety of existing technologies can serve as the input mechanism to a 3D fax machine. In particular, a variety of structured light approaches, most based on scanning lasers, are capable of producing medium to high resolution digital representations of the 3D external structure of an object [8], [9]. Multicamera stereo systems can also capture dense shape information, though with a variety of limitations imposed by non-Lambertian surfaces and the unsolved nature of the correspondence problem. In applying programmable matter to this task, several key differences are apparent.

1) Contact vs. non-contact sensing: Programmable matter can read the shape of an object by direct contact with its surface. Structured light (laser) and stereo approaches work at some distance and hence impose constraints on object curvature and self-shadowing [11]. While both at-a-distance and programmable-matter-based techniques will likely have difficultly with deep concavities, programmable matter is less likely to misread moderate surface textures and simple overhanging shapes.

2) Simplified geometries: Because of the relatively limited geometric possibilities for sphere packing and the absence of large unclosed loops or featureless spaces, the localization problem for a programmable-matter ensemble can be significantly easier than solving, for instance, a dense stereo reconstruction or a sparse SLAM problem.

3) Parallel vs. serial read-in: Raster scanning techniques face a resolution vs. speed trade off in which high spatial sampling rates can only be achieved by substantially slowing the scanning process and making more passes. Programmable matter localization in contrast is a highly parallelizable operation. Resolution is a function of the catom size rather than scan rate or the spatial frequency of scanning.

A claytronic ensemble performs self-localization by a multiphase peer-to-peer communication process between the individual catoms in the ensemble. Each catom's surface is covered with contact patches permitting communication between neighbors. The particular surface site used to communicate with a given neighbor identifies the relative geometry of that neighbor to within a known tolerance for successful communication. The high degree of interconnectedness offered in a packed or near-close-packed lattice allows quick convergence for robust location estimation techniques [12].

A dense mesh of catom locations permits software running on the ensemble to reason about its own shape and perimeter. In particular, shapes of interior voids can be readily modeled by detecting the absence of catoms. If an object is completely embedded in a claytronic ensemble, then these regions without

particles will correspond directly to the volume occupied by the object. If the material is carefully and tightly applied against the object, then the details of its surface contours will be faithfully reflected down to single-catom scale. Given the known positions of the modules in a consistent coordinate frame, a digital representation of the object's surface contours can be produced.

Unlike digital sampling in PCM systems where the Nyquist frequency offers a sharp bound on sampling precision, catoms can pack spaces down to the width of a single catom. However, because reliable sampling requires high probability that catoms will actually fill each concavity — an effect dependent upon the pressures applied to manipulate the ensemble as well as the chance incidence of any lattices — the single-catom bound has a soft character. The likelihood of accurately modeling a particular surface feature decreases as it approaches single catom scale.

In the full vision of claytronics, the requirement that catoms accurately engage with every contour would be easier to achieve since the ensemble could self-reconfigure to improve its fit against the surface. In this paper we assume the catoms involved are not capable of self-reconfiguration. The user of the system applies the claytronic material on the object as if they are taking a traditional cast or making a mold. Alternatively, if the catoms involved are constructed to have low self-adhesion, we can expect the ensemble to act more like an aggregate of discrete particles, e.g., a pile of sand. In this case, the user can place the object in a container, and pour the electronic particles around the object, shaking as necessary to reduce trapped bubbles and completely cover the object.

B. Remote Transmission

After the 3D structure has been determined by digital shape acquisition, many well known techniques can be used to store, manipulate, and transmit it. A radio or optical bridge would likely be used to extract the shape information from the ensemble and transfer it to a traditional computing device such as a laptop for storage or remote transmission.

C. Shape Reconstruction

In the final stage of a 3D fax system, the transmitted data is used to reconstruct the structure of the object at the remote site. One method of converting the digital description to a physical replica is to use a 3D printer based on rapid prototyping techniques such as fused deposition modeling or stereolithography [10]. Such a device can create a physical object from a digital representation by building up the structure layer by layer or line by line. Typical spatial resolutions are 50-300 microns and the machines often require costly consumables and caustic solvents as part of the process.

With programmable matter, shape reconstruction can be implemented in at least two different ways:

First, with a fully-functional claytronic ensemble, i.e., one capable of self-reconfiguration, we could imagine the ensemble reshaping itself on command to conform to the desired shape. Because many catoms can move simultaneously this process would be substantially faster than a raster or planar deposition process. However, this constructive approach can be very complex, and scalable methods for such self-reconfiguration motion planning are an open research question.

Second, with catoms incapable of self-reconfiguration but equipped with simple inter-catom latches which can selectively bond one module to another, a shape can be formed by what we term *digital sand casting*. In digital sand casting we start with a large mass and remove unneeded material, as in wood or stone carving. First, an appropriate volume of catoms is used to fill a closed structure such as a bucket. Power is supplied to the ensemble and the desired shape is transmitted to it. The catoms in the ensemble carry out self-localization to identify the coordinate structure within which they sit. Then, each catom evaluates whether it is or is not part of the target shape. Those catoms which are part of the target shape bond themselves together, while the other catoms simply switch themselves off. The user then pours or sweeps off the unbonded catoms to reveal the reconstructed shape. For all but the smallest volume shapes this process should also be substantially faster than incremental deposition as is typical in rapid prototyping.

The first and second techniques are duals of one another, though one may require a smaller volume of catoms and the other less computation.

III. Design

In this section, we look at the details of the software to implement a 3D fax machine on a large collection of modular robots acting as programmable matter. Critical to the success of the system is the scalability of the distributed techniques employed, as the number of units in the system can be very large — running into many thousands or even millions of units.

A. Localization

The most critical algorithm in a programmable-matter-based 3D fax machine is that for self localization of the robotic modules. The specifics of the algorithm will depend on the characteristics of the modular robot hardware. Prior research [12], also based on claytronics, has already described a localization algorithm based on local broadcast communication to immediate neighbors, a capability that very short range RF communication mechanisms such as near-field coupling could enable. The localization technique described in [12] works for highly connected lattices — in particular, hexagonal planar (2D), hexagonal close-packed (3D) and face-centered cubic (3D), but is unable to cope with less densely interconnected lattices such as cubic packing.

In contrast to [12], we seek to perform localization in both hexagonal and cubic lattice structures, as well as in ensembles with mixed regions of varying lattice types and parameters. Our approach is also motivated by a different set of assumptions about the catom hardware, reflecting two more years of work in our catom prototyping efforts. In particular, we no longer expect local-neighborhood broadcast communication, but rather envision catoms which are equipped

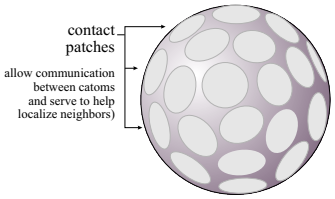

Fig. 2. Contact points on the surface of a module / catom

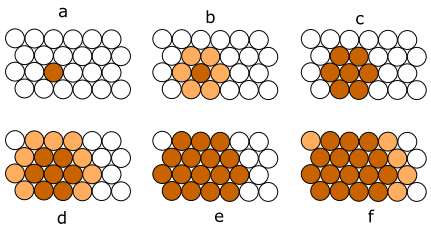

Fig. 3. Localization in a hexagonal lattice (2D shown for clarity): (a) initial random seed sends positions to neighbors; (b) partially configured set of neighbors know their positions (but not orientations); (c) with info from 2 or more neighbors, the partially configured modules determine their orientations and change to the fully configured state; (d) all fully configured modules propagate position information to each of their neighbors; (e) the next wave of neighbors determines their orientations; (f) continues

for directional communication with each of their immediate neighbors, but have no other communication facilities. These characteristics could be satisfied by short range optical, wired, or capacitively coupled communication mechanisms. This peer-to-peer approach sacrifices broadcast and long range transmission but offers higher reliability as well as directional attribution for messages received and directional control for outbound messages.

We model each catom as a spherical module with a collection of contact points spaced across its surface (see Fig. 2). Each contact point serves as a communications interface between modules. Only those contact points which actually touch other catoms are active at any given time. Contact points may also serve to distribute power [13], or implement intercatom actuation mechanisms via, for instance, magnetic or electric fields [14], but these techniques are beyond the scope of this paper.

Each contact point is treated as a separate communication channel. A catom knows *a priori* the positions of the contact points on its surface and can compute the angle between and the relative positions of pairs of its neighbors. We assume that catoms are randomly oriented, but that adjacent pairs are always aligned at a pair of surface contacts. That is, a pair of neighboring catoms always touch precisely at the center of two contact points. (In future work we plan to lift this aligned-contact requirement.)

Localization Algorithm: Given this inter-module communication model, we have devised a fully distributed algorithm to establish a consistent coordinate frame across a connected ensemble and localize individual modules within it. This algorithm proceeds as follows: Initially, all modules power up unassigned to a location or coordinate frame. A module is selected to be the *seed* (origin) of a new coordinate frame. The seed sets its own position to be the origin and selects an arbitrary orientation.

Modules knowing their location and orientation are said to be *fully configured*. Since all modules have the same radius (assumed to be 1 without loss of generality), every module can infer the position of each adjacent module from its *a priori* map of its surface contacts and the identity of the specific contact through which the catom and corresponding neighbor touch. Every fully configured module sends each of its neighbors a message indicating the neighbor's computed position and the configured module's own position. Each unconfigured neighbor receiving such a message becomes a *partially configured* module.

A partially configured module knows its position in the coordinate frame, but does not know its orientation. It shares its position information with its neighbors, but cannot definitively tell the neighbors their positions. A partially configured module, using its own position and the position of one neighbor, can partly constrain its orientation relative to the coordinate frame to one degree of rotational freedom. The position information from a second, non-collinear neighbor suffices to fully determine the partially configured module's orientation. Upon establishing its orientation thusly, the partially configured module marks itself as fully configured and can compute and propagate the positions of all its neighbors.

For hexagonal lattices (2D hex planar as well as the 3D forms, HCP and FCC), this set of rules is sufficient to localize all modules in a distributed manner. However, for cubic lattices, modules have fewer neighbors yet six degrees of freedom, and situations will arise where a partially configured module has only one neighbor with a known position. In such a case, the module will be unable to determine its orientation and progress will stop. To treat the cubic case successfully, we introduce an additional state transition: if an unconfigured module receives the positions of two neighbors, a and b, on orthogonal contacts (i.e., vectors from the center to the contact points form a right angle), it assumes a cubic packing relationship applies to those two neighbors and proceeds to determine its position as follows. First, the module requests the positions of all neighbors of a and b. In the case of square planar or cubic packing, a and b will have one additional neighbor in common, c. Using simple vector math, the module computes its own position as $a + b - c$, and advances to the partially configured state. This additional rule is sufficient to fix the distributed localization algorithm for cubic packed lattices (see examples in Figures 3 and 4).

B. Selecting Seeds

For the localization algorithm outlined above, we need a single module to serve as the seed that determines the coordinate frame. To this end, we use a simple stochastic approach: each unconfigured module randomly decides with some small probability p to become a seed of a new coordinate frame. This random election is repeated periodically by modules

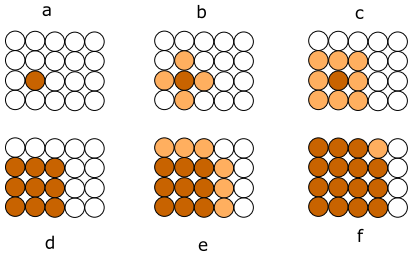

Fig. 4. Localization in a cubic lattice (2D shown for clarity): (a) initial seed, sends locations to neighbors; (b) partially configured neighbors cannot determine their orientations; (c) unconfigured modules determine their positions from two neighbors at right angles; (d) orientation determined from two or more neighbors; (e) propagate positions; (f) continues

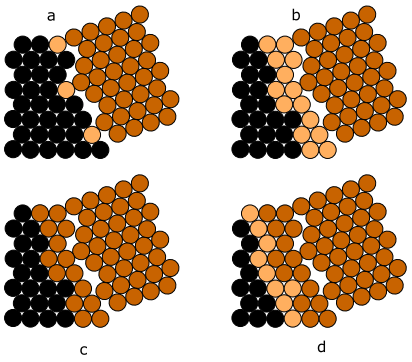

Fig. 5. Dealing with grain boundaries: (a) different frame of reference on each side of the boundary; the higher priority frame (brown / medium grey) extends across bridge points but these modules lack sufficient information to determine their orientations and progress halts; (b) using correspondences from multiple bridge points, a transform between coordinate frames is determined, and modules neighboring the bridge points switch frames; (c) orientations can be determined when positions of two or more neighbors are known, allowing the bridge modules and their neighbors to become fully configured; (d) propagation of higher priority coordinate frame continues on other side of the grain boundary

remaining in the unconfigured state to ensure at least one seed is eventually formed even in very small systems. Each seed picks a unique (or probabilistically unique) id, either from a factory-programmed serial number or via a random number generator. They initiate the localization process described above, prepending the messages in the basic algorithm with the id of the seed module for the coordinate frame. We deal with multiple seeds and coordinate frames by prioritizing based on the id of the seed. Higher id's take precedence, and when two regions of partially and fully configured modules grow into each other, the modules configured in the coordinate frame with the lower id are simply treated as unconfigured and relocalized in the higher id coordinate frame. The seed with the highest id ultimately wins, and all modules will be localized in its coordinate frame.

C. Termination Detection

It is useful to know when the localization process is complete so we can initiate the next step in the 3D fax application. To detect termination in a scalable manner, we generate a spanning tree over the entire ensemble of modules in parallel to the localization algorithm.

The seed is treated as the root of the tree. When the seed or any module turns into a fully configured module and sends position information to its neighbors, we treat this message as a request for the recipient to become a child node of the module in the spanning tree. Each module receiving such a message for the first time sets its parent to the sender and returns an acknowledgment (ACK) message. If the module already has a different parent, it returns a negative acknowledgment, or NACK. Each module responding with ACK is added to the children list of the parent module. If a module receives a negative acknowledgment from all of its neighbors, it is a leaf node, and it sends a subtree complete message to its parent. Likewise, any module which has received a subtree complete message from each of its children, sends a subtree complete message to its parent.

As each module can send the completion message only after hearing from all of its neighbors, all reachable modules will be covered by the generated spanning tree. Furthermore, as modules attempt to add neighbors as children only after they are fully configured, each complete subtree will consist entirely of localized modules. Hence, by the time the seed has received subtree complete messages from all of its children, the termination of both the spanning tree algorithm and the localization of all modules can be guaranteed, and reliably determined.

D. Dealing with Grain Boundaries

When two regions with different lattice parameters abut, a grain boundary is formed. Along this boundary, a slight gap between the modules occurs, bridged by a few, widely spaced points of contact. Each bridge point is typically a singleton contact between two modules. As a result, a fully configured module on one side of the grain boundary can create no more than one partially configured module on the other side, and the coordinate frame cannot grow across the grain boundary. Using the algorithm from Section III-A without modification, we will eventually end up with a different coordinate frame on each side of the boundary.

At the bridge points, this situation can be detected (see Fig. 5). If a fully configured module in coordinate space a has a neighbor in coordinate space b that remains in a partially configured state for a number of rounds, and id of $b >$ id of a, the module determines that it is at a grain boundary bridge point. It computes this neighbor's position in its own coordinate space a, and sends this with the position for coordinate space b up the spanning tree to the root node, seed a. This seed node collects several such correspondences, and uses them to compute matrix T_{ab}, the transformation from coordinate space a to b. T_{ab} is multicast down the spanning tree to all modules in coordinate space a. We note that a similar mechanism is employed in [12] to cross grain boundaries, but availability of a broadcast medium, and lack of the spanning tree algorithm change the implementation details.

Although we can apply the transform at each module to immediately convert coordinate frames, we take a differ-

ent approach to ensure compatibility with the spanning tree generation and termination detection for seed b. Here, each fully configured module in a that is adjacent to a partially configured module in b applies the transform and becomes a fully configured module in coordinate space b. It then waits for the partially configured neighbor to become fully configured, and joins b's spanning tree as a child of that neighbor. Only after this does it begin to propagate the coordinate frame as in the basic algorithm. Once this crossing of the bridge points is accomplished, the higher priority coordinate frame takes over just as in the case without grain boundaries.

E. Surface Detection

Once all of the modules in the ensemble have been localized to a consistent coordinate frame, and termination of the algorithm has been detected by the seed module, we begin the next phase of the application to determine the surface of an object embedded in the ensemble. In this phase, each module determines locally whether the immediate neighborhood is complete with respect to the lattice, or if one or more neighbors are missing (see Fig. 6). A module with missing neighbors is considered a *surface-adjacent* module. It computes the positions of its missing neighbors, which potentially correspond to points on or close to the surface of the embedded object. The complete set of these missing neighbor positions, with duplicates removed, is retrieved from the ensemble to a laptop that can post process this information, and store it or transmit it to a remote site.

Efficient extraction of this data can be facilitated as follows. First, we pivot the root of the global spanning tree from the seed module to a module that can communicate to the external interface device. Next, we create a spanning tree on each connected set of surface-adjacent modules, with a random root. Missing neighbor positions are sent up the trees, to the root. As duplicates can only occur locally, each surface-adjacent module checks with its neighbors to ensure any points in common are reported only once. The position data is streamed from the roots of the surface adjacent spanning trees along the global spanning tree, and to the external laptop.

If one places a module at each of the locations indicated by the extracted data, a rendition of all of the detected surface is formed. However, this includes more than just the surfaces corresponding to the embedded object. In particular, the external surface, any bubbles or voids, and grain boundaries are also included. We assume in this work that the modules were densely packed and no voids were present. We note that along grain boundaries, modules on both sides of the boundary report missing neighbors, and, under the assumption of no voids or bubbles, that modules placed at these points will overlap. Using an efficient data structure for 3D proximity, e.g., octtree, we can post-process the data to eliminate such positions that would result in overlap, and thus remove the false surfaces due to grain boundaries.

Finally, the same data structure is used to eliminate the external surface. We start with a position known to be part of the external surface, e.g., it has the largest value in some component axis. From this point, we grow a set of these data points, iteratively adding those within 3 module radii of any point within the set. Given that all grain boundaries that can bridge between the external and object surface have already been removed, and that the object surface itself is sufficiently separated from the outside surface, this set of points corresponds to just the external surface, and is removed from the data set.

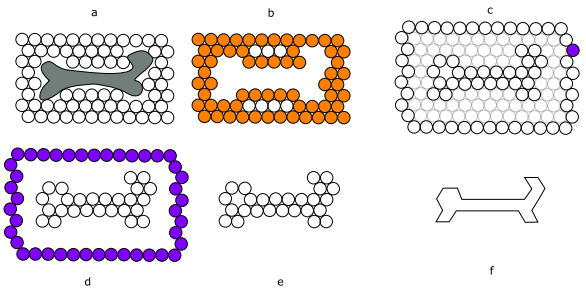

Fig. 6. Surface detection: (a) modules surrounding original object perform localization; (b) modules marked as surface-adjacent (shaded) have one or more missing neighbors; (c) missing neighbors (dotted outlines) are candidates for membership in the object surface; select one location known to be an extremity (shaded); (d) grow connected components from that extreme point – these correspond to the external surface and are removed, (e) revealing the acquired positions for the object surface; (f) view of these points as a mesh

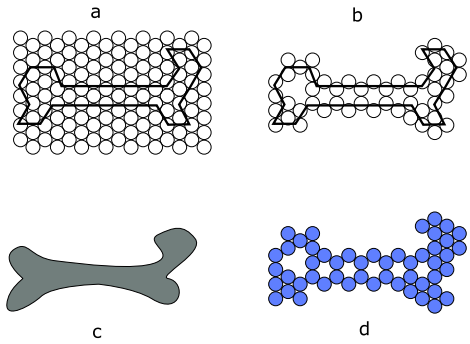

Fig. 7. Shape Reconstruction: (a) overlay mesh of output points over a full lattice; (b) select modules that intersect mesh; these latch on to each other; remove all other modules revealing replica (d); compare to original (c)

These processing steps, since they deal with the entire set of surface points, are best done on an external computer, rather than on the memory- and processing-limited claytronic modules. This external machine may also be able to compress the representation of the surface prior to storage or transmission. Efficient encoding of 3D surface models has been the subject of considerable research and is beyond the scope of this paper.

F. Shape Reconstruction

In the shape reconstruction step, we start with solid volume of claytronic material, with a single lattice structure and a known frame of reference. Fig. 7 illustrates this process. We take the set of surface positions from the input phase, and perform translation, rotation, and scaling as needed to fit all of the points to the new coordinate frame and within the destination volume. This set of surface points will not, in general, correspond to module positions in the output

ensemble. So, as a preprocessing step, we consider the set of points as vertices of connected triangular regions that form a surface mesh. By looking at the known lattice locations in the target ensemble, we can determine which modules will intersect with the mesh. The positions of these surface modules are fed to the target mass. The modules corresponding to those positions effect bonds to any of their neighbors which are also members of the position set. All other modules simply remain inert, or actively repel one another. The user then removes the unwanted modules (e.g., by wiping, shaking, or sweeping), revealing a replica of the original object, subject to reorientation and scaling to fit within the volume of the target ensemble.

IV. Experiments

The algorithms described above have been implemented on top of a software simulation environment for claytronics [15]. This environment supports both the execution of code and communication between claytronic modules in a simulated ensemble. In this section, we evaluate how well our system acquires and reproduces the shape of various trial objects.

A. Simulation set up

The simulator takes as an input the code that must be run on the modules, and a file specifying the state of the world. For our purposes, this state consists of the actual positions of the modules in the ensemble. This information is used only by the simulator and is not accessible to the code running on the simulated modules. The simulator does not support objects other than the modules themselves in the world.

To simulate a mass of catoms wrapped around an object, we construct a large cubic packed lattice of spherical modules. The object to be modeled is represented as an arbitrary volume described by equations. Using these equations, we eliminate modules from the lattice that would overlap with the object. This results in a hollowed out region that corresponds to the modules displaced by the object that would be embedded in the mass. Although the object shapes can be arbitrary, for simplicity of evaluation we mainly consider rectangular prisms and spheres here.

We execute our algorithms on the simulated modules, and retrieve the points that they compute as the object surface. We produce the claytronic replica of object by placing modules at the computed locations, and then evaluate how closely these match with the original objects.

To quantify replication fidelity, we define a mean squared error (MSE) metric between the computed set of points and the object surface, rotated and translated to match the coordinate frame of the output. For each point in the output set, we find the closest point on the object surface, and compute the distance between the two points. The mean of the squared distances for all points in the output set gives the MSE. This metric indicates how close the computed points conform to the target surface, and quickly explodes if the system outputs points that are far away.

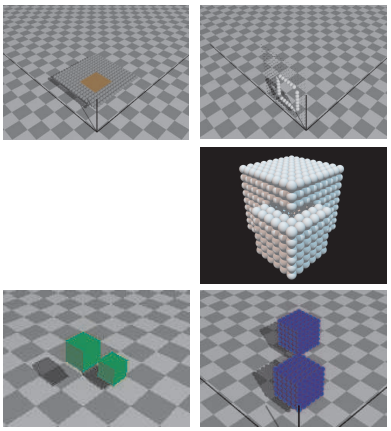

Fig. 8. Examples of 3D faxing input and output

We also compute the MSE from the object surface to the output points. Taking uniform random samples of points on the surface, we compute the distance between the surface samples and the corresponding nearest points in the output set. The MSE is computed for a large number of random samples on the object surface. This metric ensures coverage of the object surface, and will become large very quickly if parts of the object surface do not have a nearby point in the output set.

B. Simulation Results

1) Qualitative: All aspects of the algorithms described have been observed to function correctly in simulation (at least for the scenarios developed so far). Localization succeeds, even in the presence of multiple seeds, and achieves its goal of determining the positions of all modules in a consistent coordinate frame. Termination of the localization phase is correctly detected. The surfaces are correctly detected and the interior surface is successfully distinguished from the exterior surface. Finally, the system successfully generates a list of points it believes define the surface of the object.

2) MSE analysis: Fig. 9 shows the MSE computed for multiple runs of the simulation, with varying sized cubes as the input objects. The radius of the catoms is fixed to 1. The first plot shows MSE from output points to the object surface, as a function of the object size (in terms of volume). This remains in a tight range, mainly between 0 and 1, indicating that the system does indeed produce a list of points that correspond closely to the object surface. The second plot, of MSE from surface samples to the output points, also remains in a very tight range around 1, indicating that we get good coverage of the object surface with the output points.

Fig. 10 shows results for experiments in which the radius of the catoms has changed. Here, both metrics indicate that the error quickly diminishes as the size of the module decreases. Very small modules can produce very accurate replicas of an object's surface.

3) Time complexity: The localization algorithm requires what is essentially flooding of the initial ensemble of modules

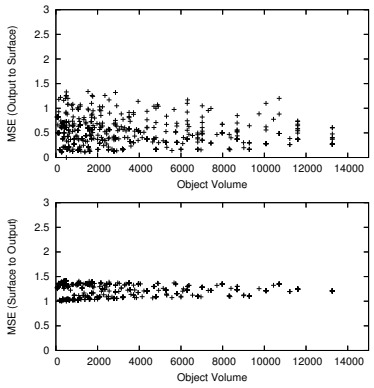

Fig. 9. MSE for rectilinear objects of varying volume, catom radius=1

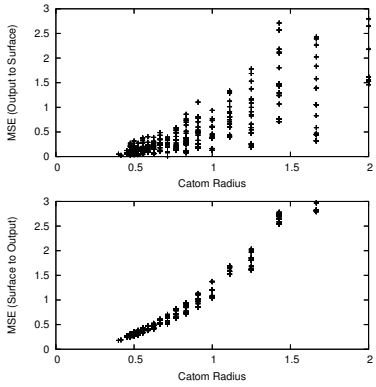

Fig. 10. MSE for various rectilinear objects, varying catom radius

to communicate a consistent coordinate frame. The time to accomplish this increases roughly with the diameter of the network formed by the modules. This is reflected in Fig. 11, which shows the runtime (in number of simulator steps, roughly corresponding to message passing intervals) of our algorithms for various sized input configurations. (Please note that these numbers do not include the time it would take to serially extract all of the generated output points from the ensemble. This time will scale proportionally to the object surface area.)

V. Conclusions

In this paper we have described novel 3D input and output devices constructed around an intelligent clay formed of a myriad of tiny modular microrobots. We also presented an algorithm for digital casting, i.e., acquisition of a 3D shape from the inverse of a modular robot ensemble's perimeter.

Two key limitations of this paper are the reliance of our evaluations on regular lattices, and the absence of real hardware testing. In the very near future we plan to extend the simulation analysis to study the impact of grain boundaries, uncertainty in orientation and alignment, and amorphous, non-lattice ensembles.

We are also working toward the design and construction of larger numbers of suitable robot modules but do not anticipate completing a quantity sufficient to demonstrate digital casting

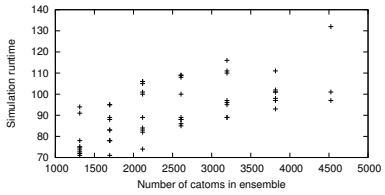

Fig. 11. Run time of system for various ensemble sizes

or 3D output for several years. (Indeed, few modular robot research efforts have ever built more than 100 units.)

Areas for future work include consideration of variable module sizes as well as techniques to better model the object surface from module positions. In the absence of self-reconfigurability, variable module size may confer few advantages to digital casting and 3D output since the application will be unable to directly choose the distribution of modules. However a multi-step 3D output strategy in which a rough skeleton is formed out of larger modules, and the skeleton is subsequently surrounded by smaller modules which can selectively latch themselves onto the skeleton could improve fidelity versus a single-stage approach.

Acknowledgment

The authors would like to thank M. Satyanarayanan, Seth Copen Goldstein, Todd Mowry, Brian Kirby, Rahul Sukthankar, and Piyush Kedia for their suggestions and insights.

References

[1] M. Sitti and R. Fearing, "Synthetic gecko foot-hair micro/nano-structures as dry ad hesives," *Journal of Adhesion Science and Technology*, vol. 17, no. 8, pp. 1055–1074, 2003.

[2] A. Geim, S. Dubonos, Grigorieva, I.V., K. Novose lov, A. Zhukov, and S. Shapoval, "Microfabricated adhesive mimicking gecko foot-hair," *Nature Materials*, vol. 2, pp. 461–63, 2003.

[3] Advanced RISC Machines Ltd. (ARM), "Arm7tdmi core datasheet," http://www.arm.com/products/CPUs/ARM7TDMI.html.

[4] "Claytronics project website," http://www.cs.cmu.edu/~claytronics/.

[5] "Project to build a 3d fax machine," http://graphics.stanford.edu/projects/faxing/.

[6] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *Proc. SIGGRAPH 1996*, vol. 30, 1996, pp. 303–312.

[7] Reyes, "World's first 3d fax machine," http://www.ices.utexas.edu/~reyes/self/3D_fax.html.

[8] Cyberware, "3d laser scanner," http://www.cyberware.com/.

[9] XYZ RGB Inc., "3d scanning white paper," http://www.xyzrgb.com/html/whitepaper.html.

[10] A. Novac, "Rapid prototyping homepage," http://www.cc.utah.edu/~asn8200/rapid.html.

[11] G. Turk and M. Levoy, "Zippered polygon meshes from range images," in *Proc. SIGGRAPH 1994*, 1994, pp. 311–318.

[12] G. Reshko, "Localization techniques for synthetic reality," Master's thesis, Carnegie Mellon University, 2004.

[13] J. Campbell, P. Pillai, and S. C. Goldstein, "The robot is the tether: Active, adaptive power routing for modular robots with unary inter-robot connectors," in *IEEE IROS*, 2005.

[14] B. Kirby, J. Campbell, B. Aksak, P. Pillai, J. Hoburg, T. Mowry, and S. C. Goldstein, "Catoms: Moving robots without moving parts," in *AAAI (Robot Exhibition Abstract)*, 2005.

[15] "Dynamic physical rendering simulator (unpublished)," http://www.pittsburgh.intel-research.net/dprweb/.