Surface Approximation by molding a shape-memory polymer on a modular robot

Florian Pescher, Benoît Piranda, Stephane Delalande, and Julien Bourgeois

Abstract The design phase of a car development is a long and tedious process that requires a lot of trial and error. In this article we introduce a new concept aiming to make this process easier and more interactive. Our solution consist in using a modular robot together with a shape memory polymer in order to create an interactive model of a car piece. We came up with an algorithm to approximate the surface of a piece using NURBS to describe the shape and the resulting molded polymer is simulated on our simulator VisibleSim, proving the accuracy of our system.

1 Introduction

Nowadays during the design phase of car development, a lot of time is used creating prototypes to prepare the final style of the new vehicle. Before realizing a Computer Aided Design (CAD) of the shape, physical objects, handmade with clay, are used in order to design the most interesting parts dreamed by designers. To reduce and to optimize this fastidious work, we would like to replace physical clay with a more interactive system based on programmable matter. The matter we have in mind will be a help for CAD tools, able to display on real matter an object being designed. Interactions with the real object or the CAD model will automatically be reproduced on the other one. Thus, a designer will be able to design an object by hand or with a CAD software and restart this process as many time as needed. This matter will
be composed of an ensemble of robots able to move by themselves around the others and equipped with a processing unit allowing them to perform calculations in order to plan their movements and achieve the desired shape. However, to realize this vision only with a modular robot, a huge amount of module is going to be necessary. To simplify the concept, we had the idea to cover the modules with a fabric able to modify its shape and recover its initial form. The most appropriate material to answer our needs is a shape memory polymer (SMP). An overview of how the complete process will work can be seen in Fig. 1 and some simulation results of the different steps are shown in Fig. 2.
2 Context

2.1 Robots and development context

Catoms 3D proposed by Piranda and Bourgeois in [4] are quasi-spherical robots that can be placed in a regular Face-Centered Cubic lattice (FCC). The proposed geometry allows to connect each robot to 1 to 12 neighbors and to displace a robot to one neighbor cell of the FCC lattice by rotating it around static neighbor modules.

In the present paper, we propose to use 3D catoms to define a dynamical surface mold. This mold is then used to shape the polymer surface. The displacement capabilities of 3D Catom are used to update the shape of the mold depending on an external request.

Lattice system gives a position of a module with a triplet of integer coordinates. For this application, we orientate the lattice in order to define vertical columns of modules, that implies to adapt the coordinate system in the FCC lattice. Each horizontal plane \((\vec{X}, \vec{Y})\) is made of staggered lines of modules, aligned along the \(\vec{X}\) axis. In order to get the same vector to access to each neighbor cell from a module, we define a lattice coordinate system \((\vec{i}, \vec{j}, \vec{k})_L\) where modules placed on \(\vec{i}\), \(\vec{j}\) and \(\vec{k}\).
axes are respectively drawn in red, green and blue. With this coordinate system, we define $M$ as the homogeneous transformation matrix to convert lattice coordinates to world coordinates:

\[
M = \begin{pmatrix}
2 \times r & r & 0 & x_0 \\
0 & \sqrt{2} \times r & 0 & y_0 \\
0 & r & 2 \times r & z_0 \\
0 & 0 & 0 & 1
\end{pmatrix}
\]  

(1)

Where $(x_0, y_0, z_0)$ is the position of the $(0, 0, 0)_L$ cell of the lattice expressed in the world coordinate system.

**Fig. 3** Left: Organization of 3D catoms along a FCC lattice. Right: Polymer surface placed over 3D catoms configuration.

Every 3D catom has 12 connectors (drawn in red in Figure 3) used to exchange messages with neighbors. Left part of Figure 3 shows the organization of 3D catoms in the FCC lattice. In this context, we orientate the lattice in order to vertically align modules. Each vertical column is placed on a bottom connector (drawn in grey), the first module drawn in green leads the treatments of the other modules of the column. The bottom connector allows to connect every green module to a common central neighbor. These connections will be used to transfer the shape of the goal surface to every module and to synchronize the distributed treatments.

The right part of Figure 3 shows the final polymer surface deformed by the 3D catoms configuration presented on the left of the figure.

3D catoms robots are not available yet but geometrical configuration and distributed program can be evaluated in VisibleSim simulator. VisibleSim [3] is able to simulate large scale configurations of tens of thousand of robots of several shapes including 3D Catoms. The distributed program that managed robots behavior is written in C++. It allows to exchange messages between modules, actuates rotations and get events of physical interactions.
2.2 Surface Description and choice of the NURBS solution

In this section we will study the advantages and drawbacks of several shape descriptions solutions and explain our choice of using NURBS taking into account the tradeoff between precision and data size needed.

2.2.1 The shape description problem

In our application we first need to describe the shape we want our robots to achieve in the most efficient way, in other words using the less data possible since our robots have a limited memory. In cartesian coordinates the shape description problem can be seen as a binary classification problem: We are searching a function \( f(x, y, z) \) such that \( f(x, y, z) \) return true if the cartesian position \((x, y, z)\) is in the desired shape and false otherwise. In our application we are limiting ourselves to surfaces, which means that at each couple \((x, y)\) there is a single highest point \( z_{\text{max}} \) that describes the surface.

With this assumption the shape description problem can be turned into a regression problem: We are searching for a function \( g(x, y) \) such that \( g(x, y) \) returns \( z_{\text{max}} \). We can then obtain \( f \) from \( g \) by doing the following: \( f(x, y, z) \) returns true if \( z \leq g(x, y) \) and false otherwise. In this section we will focus on solution that solves the regression problem.

2.2.2 NURBS method

NURBS [2] or Non-Uniform Rational Basis Splines are a mathematical model used to represent curves and surfaces. Nowadays they are widely used in computer-aided design (CAD). Here we will only consider the NURBS surfaces. They are defined by a set of control points and 2 knot vectors. The control points are represented with their homogeneous coordinates (Their 3D position \((x, y, z)\) and a weight \(w\)). The knot vectors then define how the different control points affect the final shape. The number of knots \( m \) is defined as follow: \( m = n + d + 1 \) with \( n \) the number of control points and \( d \) the degree of the NURBS. NURBS surfaces depends on two parameters \( u \) and \( v \), as such they have two degrees \( n_u \) and \( n_v \) and 2 knots vector with respectively \( m_u \) and \( m_v \) knots. If we note the control points \( P_{i,j} \) with respective weight \( w_{i,j} \) the NURBS surface \( S(u, v) \) can be computed as follow:

\[
S(u, v) = \frac{\sum_{i=0}^{m_u-n_u-1} \sum_{j=0}^{m_v-n_v-1} N_i^{m_u}(u)N_j^{m_v}(v)w_{i,j}P_{i,j}}{\sum_{i=0}^{m_u-n_u-1} \sum_{j=0}^{m_v-n_v-1} N_i^{m_u}(u)N_j^{m_v}(v)w_{i,j}}, (u, v) \in [0, 1]^2 \tag{2}
\]

In this equation the \( N_j^d(t) \) are computed using the Cox-De Boor formula:
\[
\begin{align*}
N^0_j(t) &= \begin{cases} 
1 & \text{if } t_j \leq t < t_{j+1} \\
0 & \text{otherwise}
\end{cases} \\
N^d_j(t) &= \frac{t - t_j}{t_{j+d} - t_j} N^{d-1}_j(t) + \frac{t_{j+d+1} - t}{t_{j+d+1} - t_{j+1}} N^{d-1}_{j+1}(t)
\end{align*}
\] (3)

In this equation the \( t_j \) are knots from the knot vector. In case some knots \( t_j \) are the same we suppose \( \frac{0}{0} = 0 \).

In order to solve our regression problem with this solution we need to do some changes to the method. To find the function \( g \) we need to be able to make a clear link between the parameter \((u,v)\) of the NURBS and the \((x,y)\) cartesian coordinates to be able to calculate \( g(x,y) \) at any \((x,y)\) wanted.

2.2.3 Multivariate polynomial interpolation

In [6] the author presents a generalization of Lagrange polynomial interpolation in multivariate case. With this method in order to obtain a polynom of \( m \) variable and of degree \( n \) you need \( p = \binom{n+m}{n} \) points. As a generalization of Lagrange polynomial interpolation this solution present the same advantages and drawbacks: The polynom will fit the given points, however you can not really achieve a precise shape without a high degree polynom, implying a lot of points to describe the shape.

2.2.4 Analogy with interpolation in image processing

In image processing interpolations method [5] are often used, in order to scale an image for example. Those methods could be adapted to our application by doing an analogy between the height in our application and the color in the image processing context. The most widely used method in image rescaling are nearest neighbor, bilinear and bicubic interpolation. While having relatively good results in an image processing context, those method requires a lot of initial data in order to correctly reconstruct the wanted shape making it less efficient in our context.

2.2.5 Our choice

Now that we have studied several shape description methods we compared them in Table 1. We decided to use NURBS to describe the shape in the next steps of our work since they are widely used in CAD and can describe a shape precisely with relatively few parameters.
Table 1 Comparison between the different shape description methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Data needed</th>
<th>Precision obtained</th>
<th>Advantages</th>
<th>Drawbacks</th>
</tr>
</thead>
<tbody>
<tr>
<td>NURBS</td>
<td>Polynomial order in 2 directions, 2 knot vectors, and a few control points</td>
<td>Good with only a few control points</td>
<td>Widely used in CAD: Good ratio precision/data needed</td>
<td>Parametric surface need to adapt to use on robots</td>
</tr>
<tr>
<td>Multivariate Polynomial Interpolation</td>
<td>For a polynomial of degree $n$ in both direction need for $(n+2)^2$ points</td>
<td>Depends on degree $n$. Fit the given points but bad generalization</td>
<td>Simple calculation of position</td>
<td>Need for a high degree polynomial to be precise</td>
</tr>
<tr>
<td>Nearest Neighbor Points</td>
<td>Depends on number of points. Appearance of a “stair effect”</td>
<td>Really simple to use</td>
<td>Need a lot of points to be precise</td>
<td></td>
</tr>
<tr>
<td>Bilinear Interpolation</td>
<td>Points organized in a regular grid</td>
<td>Depends on the grid. Smoother than nearest neighbor</td>
<td>Smoother than nearest neighbor</td>
<td>Need a lot of points. Regular grid on data is a big constraint</td>
</tr>
<tr>
<td>Bicubic Interpolation</td>
<td>Points organized in a regular grid</td>
<td>Depends on the grid. Smoother than bilinear interpolation</td>
<td>Smoother than bilinear interpolation</td>
<td>Need a lot of points. Regular grid on data is a big constraint</td>
</tr>
</tbody>
</table>

2.3 From 3D Model/Point Cloud to NURBS parameters

Now that we have chosen to use NURBS to describe the shape we want to create, we need to obtain those NURBS parameters. In our system there are 2 ways of obtaining those parameters: either from a 3D Model created with a CAD software or by interacting with the robots giving back an ensemble of position, in other words a point cloud. In his thesis [1] the author describe a way of obtaining NURBS from an unorganized point cloud by first transforming it into a 3D model and then transforming it into a NURBS description. Others solutions and/or simplification in our particular application might exist but will not be the focus of this article, in the following we just consider that it is possible to obtain usable NURBS parameters from either a 3D model or a point cloud.

2.4 Polymer

Shape memory materials have the ability to change their shape and recover their original shape upon application of an external stimulus. One possible way to trigger shape memory effect is to change and increase system temperature. These materials are called thermos-responsive. To obtain this ability, two conditions are required. First, switch domain as reversible thermal transition is necessary for temporary shape fixation and partial recovery. This shape memory transition allows to enable chain mobility to fix temporary shape and inversely recover permanent shape. Then, a cross-linking network determines the permanent shape to prevent chain slipping. The forming stage requires heat and the stabilization stage a temperature reduction. A common SMP presents an extent deformation up to 800%, a density between 0.9 and 1.1 g.cm$^{-3}$ and a required stress to be deformed around 1-3 MPa. Nevertheless, to limit the strength necessary to get the deformation by mems, the thickness of the foil will have to be limited consequently, a large number of transition could initiate cracks inside the material. For this reason a new material with healing properties is currently being developed and will eventually become the one used by our system.
3 Contribution

We are now going to detail some of the steps of our system. Left part of Fig. 1 show the complete system divided into 4 fields and the right part divide the reconfiguration process into several subfunctions. In this section we detail the distributed reconfiguration subfunctions and the polymer section of the complete system.

3.1 Reconfiguration: Flooding and robot motion functions

Transmitting the NURBS parameters, obtaining height \( h \) in \((x,y)\) and getting position from robots are simple functions and work in a similar way. You start from the ground of pseudo catom and send messages up with either the NURBS parameters or a request for height \( h \). Once you reach the top you send back down a message with either a "NURBS fully transmitted" response or the height \( h \). For the getting position function you can then artificially repopulate the point cloud on the computer by adding a point every \((x,y, h - n \ast d)\) with \( d \) the diameter of a catom while \( h - n \ast d \) is above ground height.

The move robots and ask for a robot in \((x,y, h + d)\) functions are used to get from one shape to another according to the shape description we have given to the robots. With the current work we are able to know how many robots are missing or too much in a vertical column compared to the goal shape thus creating some sinks and source of robots that we will use in future works to plan for the most efficient way to move the robots in order to achieve the goal shape.

3.2 Reconfiguration: Find wanted \( z_{\text{max}} \) in \((x,y)\) from NURBS

We chose to use the NURBS description method, however the usual NURBS calculation of a NURBS is a parametric surface \( S(u, v) = (x_s, y_s, z_s) \) with \((u, v) \in [0, 1]^2\) which makes it difficult to find our function. Since we limit our case to surfaces we can use a dichotomy method to find an approximation of \( g(x, y) \). The idea is to separate the surface into four quadrants, find the quadrant in which the couple \((x, y)\) where we want to evaluate \( g(x, y) \) is situated and take this quadrant as our new surface. Finally we have \( x_s \) almost equal to \( x \) and \( y_s \) almost equal to \( y \) so we can approximate \( g(x, y) = z_s \). The algorithm is described in Algorithm 1
Algorithm 1 NURBS Dichotomy Algorithm

\[ \begin{align*} u_0 &= 0; u_1 = 1; \\
v_0 &= 0; v_1 = 1; \\
\epsilon &= 0.01; \\
d &= +\infty; \\
d_{\text{max}} &= ? \\
\text{while } (d > d_{\text{max}}) \text{ do} \\
& \quad u = (u_0 + u_1) / 2; \\
& \quad v = (v_0 + v_1) / 2; \\
& \quad (x_1, y_1, z_1) = S(u - \epsilon, v - \epsilon); \\
& \quad (x_2, y_2, z_2) = S(u + \epsilon, v - \epsilon); \\
& \quad (x_3, y_3, z_3) = S(u - \epsilon, v + \epsilon); \\
& \quad (x_4, y_4, z_4) = S(u + \epsilon, v + \epsilon); \\
\text{for } i = 1 \text{ to } 4 \text{ do} \\
& \quad d' = (x - x_i)^2 + (y - y_i)^2 \\
& \quad \text{if } (d' < d^2) \text{ then} \\
& \quad \quad \text{quad} = i \\
\text{end switch} \\
\text{end while} \\
\text{return } z; \end{align*} \]

3.3 Polymer Simulation

In order to simulate the polymer we slice the polymer into an ensemble of small cubes of side length \( dl \). We consider that each of those small cubes is submitted to the same forces than the macroscopic object itself. The full polymer is submitted to the following forces:

- Its weight: \( F = -m \cdot g \) with \( g \) the constant of gravity and \( m \) the mass of the object
- The elastic deformation defined as \( F = l \cdot \frac{dl}{dr} \cdot E \) with \( E \) Young’s modulus and \( l \) the length of the contracted/extended side.
- An action of contact stopping the fall of the polymer when it hits a robot or the ground.

Since we are not considering the full polymer but an infinitesimal volume of it we can consider the volumic forces which leads to the following assumptions:

- The volumic expression of gravity is \( dF = -\rho \cdot g \) with \( \rho \) the volumic mass
- The volumic expression of elastic deformation is \( dF = \frac{l \cdot \frac{dl}{dr}}{E} \cdot E \)

We then apply Newton’s law of motion on those small cubes in order to simulate the movement of the polymer falling down on our robots in order to follow their shape.
4 Experiments

In this section we are going to test the accuracy of our system as well as the influence of the size of a catom. In order to test our system we are going to simulate the design of a car rear-view mirror with our simulator VisibleSim. We are going to run our NURBS dichotomy algorithm on the simulated catoms (see Fig. 2 c) and then simulate the polymer molding those catoms (see Fig. 2 d). The resulting polymer will then be exported into octave in order to be compared to the mathematical NURBS model. We chose to use a car rear-view mirror as a test object since this object is directly linked to the car industry and can be described as a single $z = f(x, y)$ function. Plus the required polymer deformation to mold this shape is of only 40% so the polymer would not be torn apart during a physical test with this same shape.

4.1 Shape accuracy test

In order to test the shape accuracy of our system we plotted 3 graphs in octave: The exported data of the simulated polymer, the mathematical NURBS model and the difference between those 2 graphs.

Fig. 4 Comparison between polymer and NURBS mathematical model
We can see in Fig. 4 that the NURBS mathematical model seems correctly approximated by the catoms and polymer, except on the huge altitude difference on one of the sides of the object. This error can be explained by the stiffness of the polymer. We also need to take into account that in our simulation we only applied gravity to the polymer, if we had another force to attract the polymer closer to the catoms on the sides this error can be reduced experimentally. However a simple plot analysis is not enough to judge the accuracy of our method correctly so we decided to perform a statistical analysis on the data of the difference between the mathematical NURBS and the polymer. The results of this analysis can be seen in Table 2.

<table>
<thead>
<tr>
<th>Volume mass</th>
<th>Young Modulus</th>
<th>min</th>
<th>max</th>
<th>q1</th>
<th>median</th>
<th>q3</th>
</tr>
</thead>
<tbody>
<tr>
<td>900 kg/m</td>
<td>0.024251</td>
<td>43.611</td>
<td>5.2080</td>
<td>9.8923</td>
<td>17.232</td>
<td></td>
</tr>
<tr>
<td>1000 kg/m</td>
<td>0.0033179</td>
<td>44.741</td>
<td>3.0415</td>
<td>11.297</td>
<td>19.739</td>
<td></td>
</tr>
<tr>
<td>1100 kg/m</td>
<td>0.034316</td>
<td>46.927</td>
<td>5.2032</td>
<td>11.794</td>
<td>21.235</td>
<td></td>
</tr>
</tbody>
</table>

Considering that in this experiment the diameter of a catom is 10, Table 2 shows that we can approximate really well the mathematical model on some points (really low minimum error), that the maximal error is of around 4 catoms on the side of the structure. The study of the quartiles shows that 25% of the catoms approximate the shape with an error inferior to half a catom, 50% of catoms approximate it with an error around a catom and that 75% of them shows an error of around two catoms.

This analysis show an error of around 1 catom altitude in a structure contained in a 8*12*14 catoms block meaning an altitude error inferior to 10%. That leads us to wonder if this error will grow or stay constant as the structure become larger (equivalent to the catoms becoming smaller).

### 4.2 Scale influence

We now study the influence of the size of a catom relatively to the size of the desired shape. In order to do so we consider the previous experiment of the rear-view mirror in the 8*12*14 block as the scale 1 model and we are going to perform the same statistical analysis on different scale to see the influence of the catom size.

Table 3 shows that when the size of a catom became smaller compared to the size of the desired shape, even though the maximal error increase the values of the
different quartiles remain approximately the same meaning that the relative error decrease. We can say that the smaller the catoms, the more precise our solution is.

### 5 Conclusion

In this article we introduced our new system to help the design of pieces in the car industry using a combination of modular robots and shape memory polymer. We mainly focused on finding a way to describe the desired shape introducing the NURBS dichotomy algorithm, and evaluating the accuracy of this method by simulating a polymer being molded over the catoms organized in the shape we want to design. This method proved itself to be accurate and this accuracy is improved as the catoms become smaller. Future works will focus on creating the full system and more precisely on the way to plan the catoms movement to get into the desired shape.

### References