Molding a Shape-Memory Polymer with Programmable Matter

Florian Pescher[1], Benoît Piranda[1], Stephane Delalande[2], and Julien Bourgeois[1]

[1] FEMTO-ST Institute, Univ. Bourgogne Franche-Comté, CNRS,
1 Cours Leprince-Ringuet - 25200 Montbéliard, France
{florian.pescher, benoit.piranda, julien.bourgeois}@femto-st.fr
[2] PSA Groupe, Scientific Direction
2 Route de Gizy - 78943 Vélizy-Villacoublay, France
stephane.delalande@mpsa.com

Abstract. The design phase of a car development is a long and tedious process requiring a lot of trials and errors. In this paper, we introduce a new concept aiming at making this process easier and more interactive. Our solution mixes self-reconfigurable autonomous robots forming programmable matter and a shape-memory polymer surface that produces an interactive model of the desired object. We propose a global algorithm to manage the interactions with the users and the self-reconfiguration of programmable matter to mold the polymer surface. We detail the technical aspects used to define the new shape of the programmable matter to better approach a goal surface described by a Non-Uniform Rational Basis Splines (NURBS) using a dichotomy algorithm.

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. While realizing a Computer Aided Design (CAD) of the shape, handmade physical objects made from clay are still used in order to design the most crucial parts of a car. To reduce and to optimize this fastidious work, we would like to replace the 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 times 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 modules is going to be necessary. To simplify the concept and to have a smooth surface, we 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). SMP will be heated to mold the

object represented by the robots and then cooled down so that it will keep its shape and could be used as a real molded object. This cycle, heating, molding, cooling down, can be made as many times as needed, until the molded object has the right shape. An overview of the complete process can be seen in Figure 1 and some simulation results of the different steps are shown in Figure 2.

2 Context

This work takes place in the context of the programmable matter project[3] [2]. In this section, we will first detail the robotic context before explaining the shape description choice of Non-Uniform Rational Basis Splines (NURBS), how to obtain them and finally presenting the shape-memory polymer.

2.1 Robots and development context

Modular robots geometry, called *3D Catoms* proposed by Piranda and Bourgeois in [12] are quasi-spherical robots that can be placed in a regular Face-Centered Cubic lattice (FCC). The proposed geometry allows each robot to connect and to exchange messages with up to 12 neighbors, and one robot to move to one neighboring cell of the FCC lattice by rotating around a static neighboring robot. Figure 3a shows an example of a configuration made from connected 3D Catoms in a FCC lattice.

In the present paper, we propose to use these *3D Catoms* to define a dynamical mold. This mold is then used to shape the polymer surface as shown on Figure 3b. The displacement capability of the *3D Catoms* is used to update the shape of the mold depending on an external request.

The 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. Each vertical column is placed on a bottom connector (drawn in grey in Figure 3), 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. This new orientation implies to adapt the coordinate system in the FCC lattice. Each horizontal plane $(\overrightarrow{x}, \overrightarrow{y})$ is made of staggered lines of modules, aligned along the $\overrightarrow{x}$ axis. In order to get the same vector to access each neighboring cell from a module, we define a new lattice coordinate system $(\overrightarrow{i}, \overrightarrow{j}, \overrightarrow{k})_L$. With this coordinate system, we define M as the homogeneous transformation matrix to convert lattice coordinates to world coordinates:

$$M = \begin{pmatrix} 2 \times r & r & 0 & x_0 \\ 0 & \sqrt{2} \times r & 0 & y_0 \\ 0 & r & 2 \times r & z_0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (1)$$

[3] http://projects.femto-st.fr/programmable-matter/

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.

The *3D Catom* robots are not available yet but geometrical configurations and distributed programs running in these robots can be evaluated in the *VisibleSim* simulator [11]. *VisibleSim* is able to simulate large scale configurations of tens of thousand of robots of several shapes including *3D Catoms* geometry. The distributed program that managed robots behavior is written in C++, it allows to exchange messages between modules, actuates rotations and get events associated to physical interactions.

2.2 Surface Description and choice of the NURBS solution

Describing the shape of a configuration using a minimum of memory usage is a huge problem. Vectorization of the shape may be an efficient solution as proposed in [16]. The class of volumes that can be created by our method are defined by a planar floor and a surface placed over this floor. In this context we propose to vectorize the goal surface and deduce the shape of the configuration of robots placed below. It requires much less memory than describing the whole 3D configuration. In this section, we will study the advantages and drawbacks of several shape description solutions and explain our choice of using NURBS taking into account the tradeoff between the precision and the data size needed.

The shape description problem We first need to describe, in the most efficient way, the shape we want our robots to achieve. In cartesian coordinates, the shape description problem can be seen as a binary classification problem: We are searching for a function $f(x, y, z)$, indicating that the cartesian position (x, y, z) is in the desired shape or not. In our application, we are limiting the scope of shapes to surfaces, which means that at each couple (x, y) there is a single highest point z_{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 such that $g(x, y)$ returns z_{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 a solution that solves the regression problem.

NURBS method NURBS [10] or Non-Uniform Rational Basis Spline is a classical 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 which are defined by a set of control points and two knot vectors. Control points are represented with their homogeneous coordinates (x, y, z, w).

Knot vectors define how the different control points affect the final shape. The number of knots m is defined by $m = n + d + 1$ where n is 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

vectors with respectively m_u and m_v knots. If we note the control points $P_{i,j}$ with a weight $w_{i,j}$ the NURBS surface $S(u,v)$ can be computed as follow:

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

Where components $N_j^d(t)$ are computed using the Cox-De Boor formula:

$$\begin{cases} N_j^0(t) = \begin{cases} 1 & if \quad t_j \leq t < t_{j+1} \\ 0 & otherwise \end{cases} \\ N_j^d(t) = \begin{cases} 0 & if \quad a = 0 \wedge b = 0 \\ \frac{t-t_j}{a}N_j^{d-1}(t) & if \quad a \neq 0 \wedge b = 0 \\ \frac{t_{j+d+1}-t}{b}N_{j+1}^{d-1}(t) & if \quad a = 0 \wedge b \neq 0 \\ \frac{t-t_j}{a}N_j^{d-1}(t) + \frac{t_{j+d+1}-t}{b}N_{j+1}^{d-1}(t) & otherwise \end{cases} \end{cases} \quad (3)$$

Where t_j are components of the knot vector, $a = t_{j+d} - t_j$ and $b = t_{j+d+1} - t_{j+1}$.

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.

Multivariate polynomial interpolation In [14], Saniee presents a generalization of Lagrange polynomial interpolation in the multivariate case. To obtain a polynomial of m variables and of degree n with this method, $p = \binom{n+m}{n}$ points are needed. As a generalization of the Lagrange polynomial interpolation this solution presents the same advantages and drawbacks: The polynomial will fit the given points, however, a precise approximation is only reached with a high degree polynomial, implying lots of points to describe the shape.

Analogy with interpolation in image processing In image processing, interpolation methods [13] are often used to reduce the size of an image. 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 scaling are: the nearest neighbor, bilinear and bicubic interpolations. While having relatively good results in the image processing context, those methods require a lot of initial data in order to correctly reconstruct the wanted shape making it less efficient in our context.

Our choice Table 1 shows the studied shape description methods. 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 (48 NURBS control points for the car mirror, whereas other evaluated methods failed to achieve complex shapes with approximately the same number of control points).

Table 1. Comparison between the different shape description methods

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

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 two ways: either from a 3D Model created with a CAD software or by interacting with the robots giving back an ensemble of positions, in other words a point cloud. In [5], Lankinen describes a way of obtaining a NURBS from an unorganized point cloud by first transforming it into a 3D model and then transforming it into a NURBS description.

2.3 The Polymer surface

Shape memory materials have the ability to change their shape and recover their original shape upon application of an external stimulus [4][7][8][9][17]. One possible way to trigger shape memory effect is to change and increase system temperature. These material 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 $1-3\ MPa$ [1][3][6][15]. 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 developped and will eventually become the one used by our system.

3 Contribution

We are now going to detail some elements of our system. Left part of Figure 1 shows the complete system divided into 4 fields (A. NURBS definition, B. Distributed reconfiguration, C. User interaction and D. Polymer molding) and the

right part divides the "Distributed reconfiguration" process into several sub-functions. In this section, we detail the "Reconfiguration algorithm" and the "Polymer molding" sections of the complete system.

3.1 The Reconfiguration Algorithm

We consider a set of vertical columns of modules placed over the $(x, y, 0)_L$ position (lattice coordinates), as shown in Figure 3. Each column is made of a vertical line of connected *3D Catoms* that deduces their z world position from their order in the column. All the elements placed at the origin of the column are connected to a central system. The first module (called origin of the column) is placed at $M \times (x, y, 0)_L$ (where M is defined by Equation 1), then the next module is placed at $M \times (x, y, 1)_L$, and so on.

The structure of our system implies that all the actions of the "B" area of Figure 1 can be done in going through the list of connected modules of a column. It highly simplifies flooding processes as it avoids loops. For example, the first action: "Transmit NURBS parameters to all robots" is initiated by the origin module of the column. It first gets the NURBS data from the central system, then sends a "Data Message", embedding NURBS data to its top neighbor and waits for an acknowledgement. When other modules receive "Data Message", they memorize the NURBS parameters and re-transmits the message to their top neighbor. At the end of the flooding, the highest module of the column sends the acknowledgement to the origin module.

We now detail the "Reconfiguration algorithm" proposed on the right side of Figure 1. This distributed algorithm is executed on each module simultaneously.

For the first action ("Get robot coordinates (x, y, z)") the module obtains its current Cartesian coordinates, asking the origin module of its column.

Then, "Calculate z_{max} from NURBS (dichotomy)" makes the link between the NURBS parameters and the Cartesian coordinates delimiting our shape using a dichotomy process detailed in Algorithm 1. z_{max} is then computed locally using its local position and the NURBS description.

The "Find current height $h(x, y)$" function works in a similar way as the "Transmit NURBS parameters to all robots" described earlier. Each module requests its top neighbor for the height h (the z coordinate of the highest module of the column).

The z_{max} needed for obtaining the shape described by the NURBS is then compared to the current height of the column h. If there is not enough *3D Catoms* in the column to reach the NURBS position ($z_{max} \leq h$), then, the "Ask robot in $(x, y, h + d)$" function is called, requesting a new robot being the top of the column. This request is sent to the origin module of the column, to express that a module in excess in another column must move to this place.

If a sufficient number of *3D Catoms* are present in the column, each robot's z coordinate is compared to z_{max} and if $z_{max} \leq z$ then the "Move robot" function is called. This function creates a request which is sent to the origin module to inform it that modules are available at the top of this column and can be used to

Algorithm 1 NURBS Dichotomy Algorithm

```
u_0 = 0; u_1 = 1; v_0 = 0; v_1 = 1;
ε = 0.01;
d = +infinity
d_max = 3DCatom radius
while (d > d_max) do
  u = (u_0 + u_1)/2;
  v = (v_0 + v_1)/2;
  (x_1, y_1, z_1) = S(u − ε, v − ε);
  (x_2, y_2, z_2) = S(u + ε, v − ε);
  (x_3, y_3, z_3) = S(u − ε, v + ε);
  (x_4, y_4, z_4) = S(u + ε, v + ε);
  for i = 1 to 4 do
    d' = (x − x_i)^2 + (y − y_i)^2
    if (d' < d^2) then
      quadrant = i
      d = √d'
    end if
  end for
  switch (quadrant)
  case 1:
    u_1 = u; v_1 = v; z = z_1;
  case 2:
    u_0 = u; v_1 = v; z = z_2;
  case 3:
    u_1 = u; v_0 = v; z = z_3;
  case 4:
    u_0 = u; v_0 = v; z = z_4;
  end switch
end while
return z;
```

fill other columns. More precisely the robot movement will be realized in future work using a minimum cost flow algorithm for path planning.

The "Synchronization" phase consists in waiting for other modules that are moving to complete all the previous constraints. When all robots have finished the algorithm then the robots at the top of columns approximate the shape described by the NURBS.

3.2 Polymer Simulation

In order to simulate the polymer, we slice it into an ensemble of small cubes of side length dl equal to its thickness. We consider that each of those small cubes is submitted to the same forces as the macroscopic object itself:

- Its weight: $F = -m * g$ with g the constant of gravity and m the mass of the object
- The elastic deformation defined as $\frac{F}{dl^2} = \frac{l-dl}{dl} * E$ with E Young's modulus and l the length of the contracted/extended side.
- A collision stopping the fall of the polymer when it hits an other robot or the ground.

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

- The volumetric expression of gravity is $dF = -\rho * g$ with ρ the volumetric mass of the polymer.
- The volumetric expression of elastic deformation is $dF = \frac{l-dl}{dl^2} * E$.

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 take their shape.

4 Experiments

In this section, we test the accuracy of our system as well as the influence of the size of a *3D Catom* on the relative precision of the created shape.

We choose 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. Moreover 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.

We realize these experiments by implementing the "dichotomy" algorithm of the "Reconfiguration algorithm" presented Section 3 in our simulator *VisibleSim*.

An initial configuration of *3D Catom* is constructed in order to fill the complete volume of the scene. Then for each column of module, each module receives the NURBS description and calculates locally if it is placed under the NURBS or not. If it is placed over the surface, it is removed from the interface. (cf. Figure 2c). Then this configuration is used as an obstacle for the simulation of the molding of the Polymer surface using the iterative algorithm presented Section 3. (cf. Figure 2d)

4.1 Shape accuracy

In order to evaluate the shape accuracy of our system we plotted three graphs in *Octave*: The exported data of the simulated polymer, the mathematical NURBS model and the difference between those two graphs.

We can see in Figure 4 that the NURBS mathematical model seems correctly approximated by the *3D Catoms* and polymer, except on bigger altitude differences on one of the sides of the object. This error can be explained by the stiffness of the polymer. We also need to consider that in our simulation we only applied gravity force to the polymer, we expect better results similar to the theoretical result presented in Figure 3 if we add other forces (for example creating a negative pressure under the surface) in order to attract the polymer closer to the *3D Catoms*.

However, a simple plot analysis is not enough to correctly judge the accuracy of our method 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 Figure 5.

In this experiment, the diameter d of a module is equal to 10 mm. Figure 5 shows that we can get a good approximation of the mathematical model on some points (really low minimum error), that the maximal error is of around 4 modules on the side of the structure. The study of the quartiles shows that 75% of the modules shows an error inferior to $2 \times d$, 50% of modules approximate it with an error inferior to d, and that 25% of the modules approximates the shape with an error inferior to $\frac{d}{2}$.

Considering a structure contained in a $8 \times 12 \times 14$ box, the analysis gives an error close to d along the z axis, that means an error inferior to 10% along z. That leads us to wonder if this error will grow or stay constant as the structure become larger (equivalent to the 3D Catoms becoming smaller).

4.2 Scale influence

We now study the influence of the size of the *3D Catoms* relatively to the size of the desired shape. Considering the previous experiment of the rear-view mirror in a $8 \times 12 \times 14$ box as the reference scale, we perform the same statistical analysis on different scale of the mirror to show the influence of the module size to the precision of the approximation of the NURBS by the polymer surface.

Figure 6 shows that when the size of a *3D Catom* become smaller, even though the absolute error remains around the same, the relative error is decreased. As could be expected when the size of a *3D Catom* is divided by 2, so is the surface approximation error which encourage us to pursue the development of smaller modules.

5 Conclusion

In this paper, we introduced a new way to help the design of objects in the car industry using a combination of Modular Robots and Shape-Memory Polymer. We present a global algorithm that integrates the shape Analysis, the Self-Reconfiguration process and the human/system interactions. We mainly focused our work 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 *3D Catoms* organized in the shape we want to design. Future works will focus on creating several parts of the full system and more precisely on the way to plan the *3D Catoms* movement to reach the desired shape.

Acknowledgement This work was partially supported by the ANR (ANR-16-CE33-0022-02), the French Investissements d'Avenir program, ISITE-BFC project (ANR-15-IDEX-03), Labex ACTION program (ANR-11-LABX-01-01) and the Mobilitech project.

References

1. Behl, M., Lendlein, A.: Actively moving polymers. Soft Matter 3(1), 58–67 (2007)
2. Bourgeois, J., Piranda, B., Naz, A., Boillot, N., Mabed, H., Dhoutaut, D., Knychala Tucci, T., Lakhlef, H.: Programmable matter as a cyber-physical conjugation. In: 2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC 2016). IEEE (Oct 2016), https://publiweb.femto-st.fr/tntnet/entries/13257/documents/author/data
3. Han, X.J., Dong, Z.Q., Fan, M.M., Liu, Y., li, J.H., Wang, Y.F., Yuan, Q.J., Li, B.J., Zhang, S.: ph-induced shape-memory polymers. Macromolecular rapid communications 33(12), 1055–1060 (2012)
4. Hu, J., Zhu, Y., Huang, H., Lu, J.: Recent advances in shape–memory polymers: Structure, mechanism, functionality, modeling and applications. Progress in Polymer Science 37(12), 1720–1763 (2012)
5. Lankinen, J.: 3D surface modeling using point clouds (2009)

6. Lendlein, A., Jiang, H., Jünger, O., Langer, R.: Light-induced shape-memory polymers. Nature 434(7035), 879 (2005)
7. Lendlein, A., Kelch, S.: Shape-memory polymers. Angewandte Chemie International Edition 41(12), 2034–2057 (2002)
8. Liu, C., Qin, H., Mather, P.: Review of progress in shape-memory polymers. Journal of Materials Chemistry 17(16), 1543–1558 (2007)
9. Mather, P.T., Luo, X., Rousseau, I.A.: Shape memory polymer research. Annual Review of Materials Research 39, 445–471 (2009)
10. Piegl, L., Tiller, W.: The NURBS Book. Springer-Verlag, New York, NY, USA, second edn. (1996)
11. Piranda, B.: Visiblesim: Your simulator for programmable matter. In: Algorithmic Foundations of Programmable Matter (Dagstuhl Seminar 16271) (Jul 2016)
12. Piranda, B., Bourgeois, J.: Geometrical study of a quasi-spherical module for building programmable matter. In: DARS 2016, 13th International Symposium on Distributed Autonomous Robotic Systems. pp. –. London, United Kingdom (Nov 2016)
13. Prasantha, H., Shashidhara, H., Balasubramanya Murthy, K.: Comparative analysis of different interpolation schemes in image processing. In: International Conference on Advanced Communication Systems (ICACS), India. pp. 17–24 (2007)
14. Saniee, K.: A simple expression for multivariate lagrange interpolation. In: SIAM Undergraduate Research Online (SIURO) (Jul 2008)
15. Sun, L., Huang, W.M., Ding, Z., Zhao, Y., Wang, C.C., Purnawali, H., Tang, C.: Stimulus-responsive shape memory materials: a review. Materials & Design 33, 577–640 (2012)
16. Tucci, T.K., Piranda, B., Bourgeois, J.: Coding 3d scene for distributed self-reconfiguration algorithms. In: 32nd ACM SIGAPP Symposium On Applied Computing (SAC17), Marrakesh, Morocco, Apr. 6-9. pp. 256–261. ACM (2017)
17. Zhao, Q., Zou, W., Luo, Y., Xie, T.: Shape memory polymer network with thermally distinct elasticity and plasticity. Science advances 2(1), e1501297 (2016)

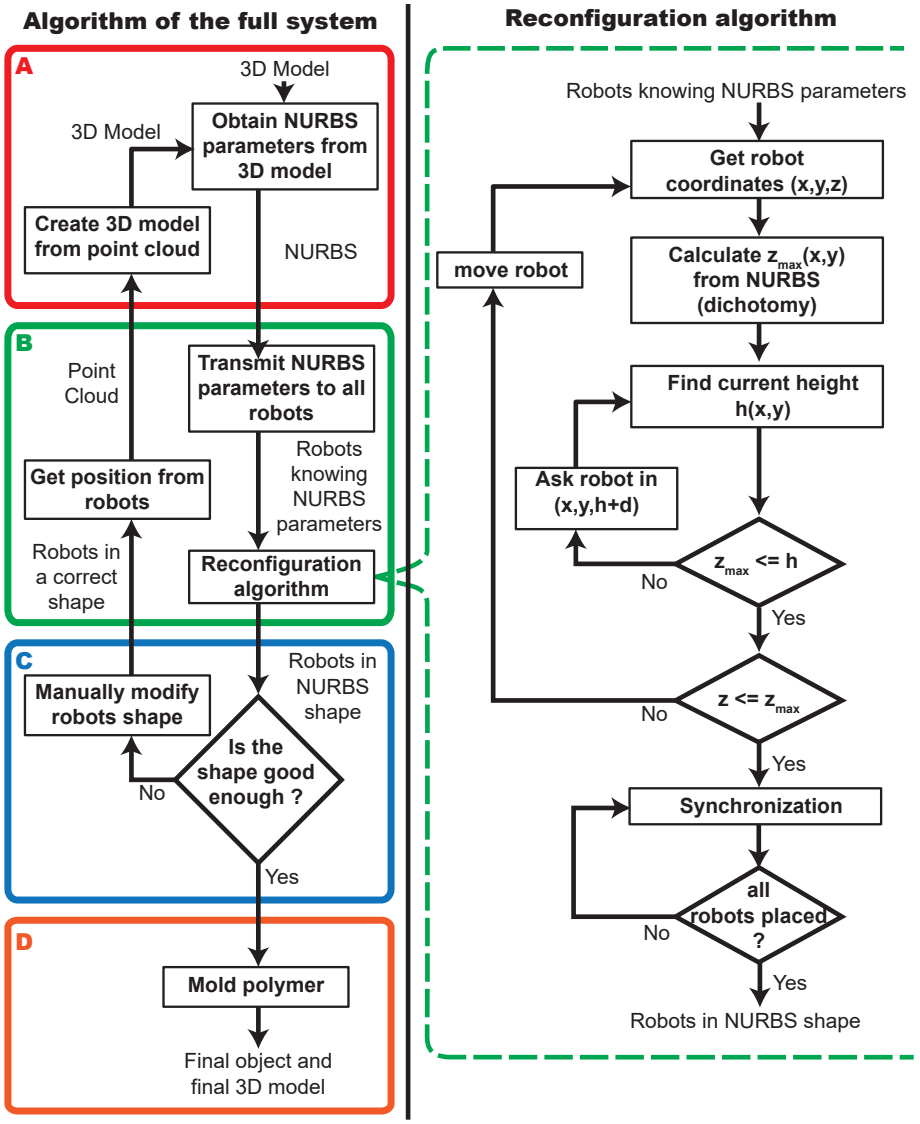

Fig. 1. The full system: Red area A is the CAD part of the system, some ideas on how to obtain Non-Uniform Rational Basis Splines (NURBS) from point cloud and/or 3D model can be found in Section 2 of the paper. Green area B is the distributed reconfiguration part of the system and detailed on the right part of the figure. Explanations can be found in the Section 3. Blue area C refers to choices made by the final user, as such no further explanation is required. And orange area D is the polymer part of the system, details on its characteristics can be found in the Section 2 and the simulation implementation idea can be found in the Section 3.

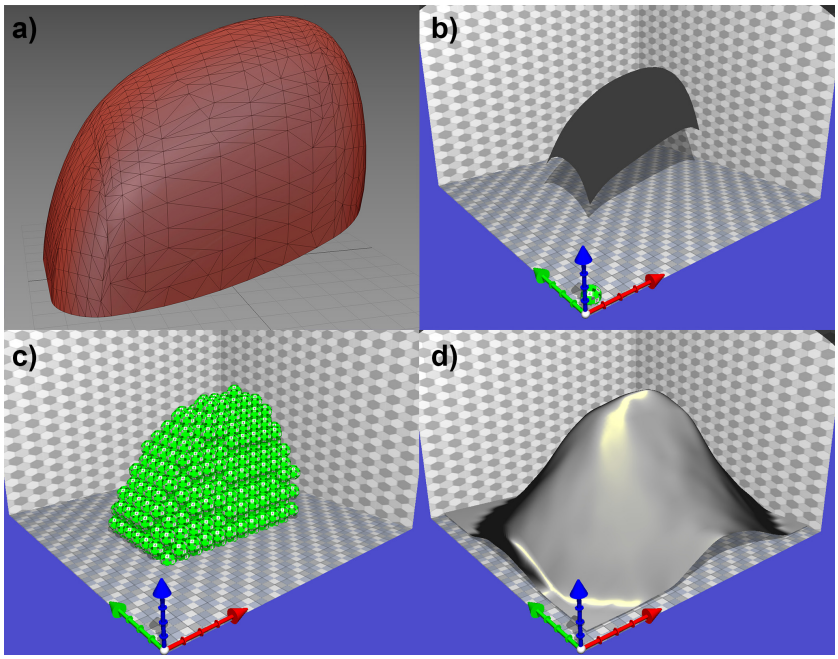

Fig. 2. The different steps of the design of a part using our system. a) The 3D model of a car rear-view mirror. b) The NURBS mathematical model representing our object. c) An ensemble of 3D Catoms taking the shape of the mirror in our simulator VisibleSim. d) The simulation of the polymer covering the ensemble of 3D Catoms.

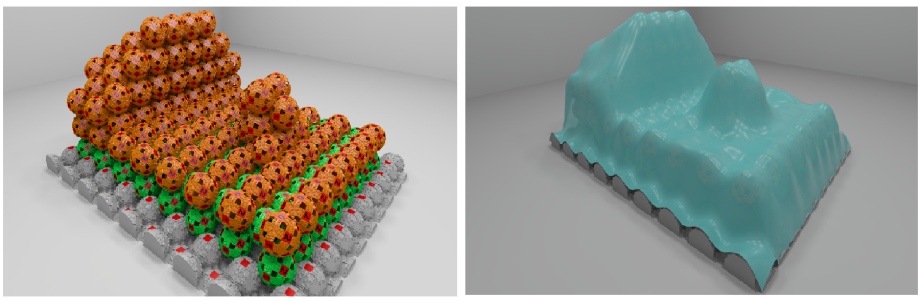

Fig. 3. Left (a): Organization of 3D Catoms along a FCC lattice. Right (b): Polymer surface placed over 3D Catoms configuration.

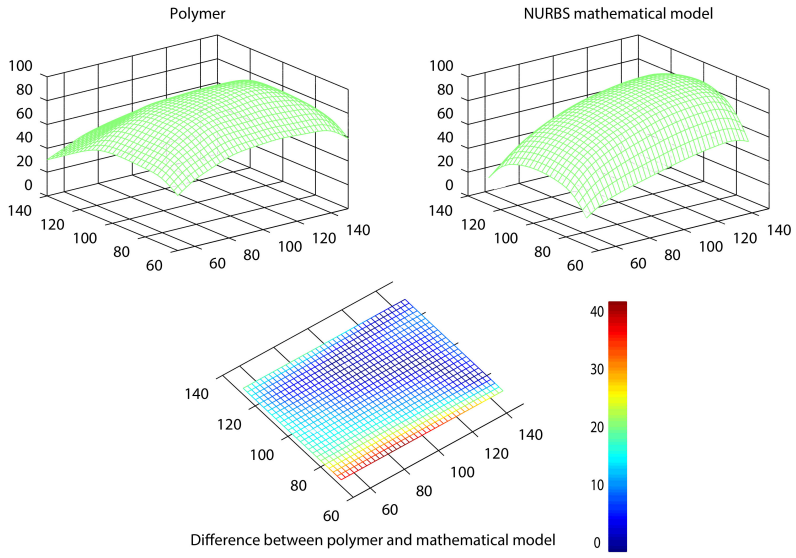

Fig. 4. Comparison between polymer and NURBS mathematical model. All values are in mm

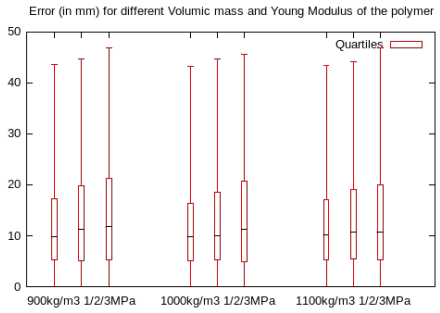

Fig. 5. Shape accuracy statistical analysis

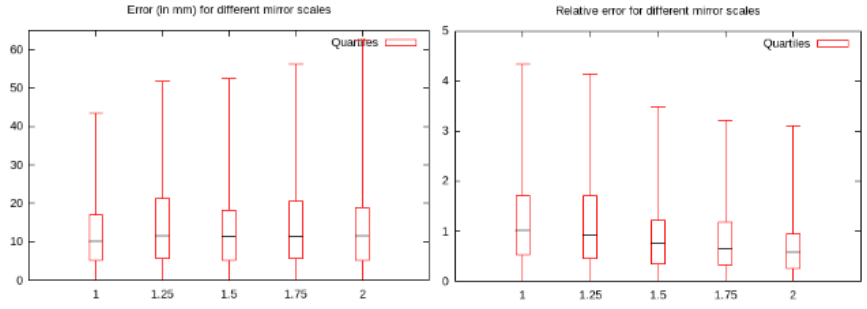

Fig. 6. Left: Error (mm) for different mirror scales. Right: Relative error for different mirror scales.