ROBOTIC AUTOMATIC GENERATION OF PERFORMANCE MODEL FOR NON-UNIFORM LINEAR MATERIAL VIA DEEP LEARNING

DAN LUO[1], JINSONG WANG[2] and WEIGUO XU[3]
[1,2,3]*School of Architecture, Tsinghua University*
[1]*luo_dana@126.com* [2]*wangjs1992@163.com* [3]*xwg@tsinghua.edu.cn*

Abstract. In the following research, a systematic approach is developed to generate an experiment-based performance model that computes and customizes properties of non-uniform linear materials to accommodate the form of designated curve under bending and natural force. In this case, the test subject is an elastomer strip of non-uniform sections. A novel solution is provided to obtain sufficient training data required for deep learning with an automatic material testing mechanism combining robotic arm automation and image recognition. The collected training data are fed into a deep combination of neural networks to generate a material performance model. Unlike most traditional performance models that are only able to simulate the final form from the properties and initial conditions of the given materials, the trained neural network offers a two-way performance model that is also able to compute appropriate material properties of non-uniform materials from target curves. This network achieves complex forms with minimal and effective programmed materials with complicated nonlinear properties and behaving under natural forces.

Keywords. Material performance model; Deep Learning; Robotic automation; Material computation; Neural network.

1. Introduction

"Good structure form" is traditionally considered to be composed with pure tension or compression, minimizing bending and deformation in the system. This perspective partially results from the difficulty simulating or controlling the quality of bending and deformation due to its highly complicated nonlinear material performance mechanism rooted in the molecular level, as well as its strong dependency on individual material qualities. This phenomenon makes it impossible to generate an accurate mathematical performance model with adjustable variables. Additionally, as most typical construction materials, such as masonry, concrete, wood, and steel, best perform under pure tension or compression, to date, it has not been determined how to generate structure and form with the natural behaviour of bending and deformation, as well as how to programme the material to achieve the designed form with such force and behaviour.

T. Fukuda, W. Huang, P. Janssen, K. Crolla, S. Alhadidi (eds.), *Learning, Adapting and Prototyping, Proceedings of the 23rd International Conference of the Association for Computer-Aided Architectural Design Research in Asia (CAADRIA) 2018*, 1-11. © 2018, The Association for Computer-Aided Architectural Design Research in Asia (CAADRIA), Hong Kong.

Elastomer, or synthetic rubber, used in the following study is a polymer with viscoelasticity, which enables a linear strip of such material to be bent and to naturally settle into a curvature under interplay of internal and external force. Currently, most bending and deformation models only deal with the behaviour of a uniform ideal material with given material properties, resulting in several limited geometrical curvatures due to the natural form finding process. However, during earlier material experiments, different non-uniform materials with changing sectional properties are noticed to settle into a vast diversity of curves under bending with the potential to approximate most given convex curves and even several concave curves under gravity with equilibrium in all forces. However, the simulation of such a natural form finding process requires a combination of in-depth knowledge of classical definite deformation theorem, finite element analysis, and material properties. Due to the nature of such a geometrically nonlinear finite elastic deformation process, the mechanism simulation of such material and process is currently extremely difficult and restricted within ideal and simplified models.

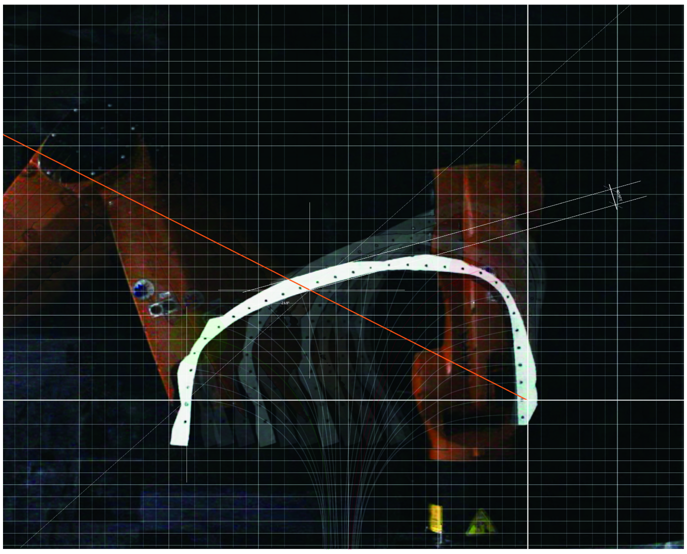


Figure 1. None-uniform elastomer material behaving under bending .

In the following research, a novel approach is developed to utilize the complexity of flexible materials with their bending and deformation features and generate a performance model that is capable of programming a non-uniform material to be bent and naturally relax into the designated form with the equilibrium of internal constraints and external forces. While most of the structural form finding process focuses on finding the most efficient form based on the ideal material model, in our research, the key to be able to "programme" the characteristic of the material according to formal intention is the use of deep learning combining robotic automation and image recognition.

Though most structure simulation software, such as Kangaroo and Rhino Vaults, begins with known material properties and constraints to simulate the final rational form, reversely computing customized material properties from given designated forms is highly difficult to achieve with current structure solutions. The ability to work backward from form to material properties would be a key feature of a performance model generated from a deep learning process compared to a typical simulation process. In our research, we used a combination of few different types of neural networks to generate the performance model from training data.

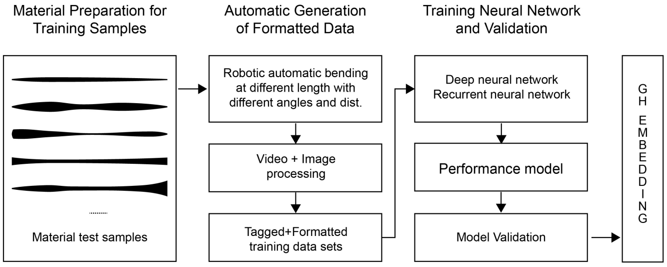

Material Preparation for Training Samples
Automatic Generation of Formatted Data
Training Neural Network and Validation
Robotic automatic bending at different length with different angles and dist.
Video + Image processing
Tagged+Formatted training data sets
Deep neural network Recurrent neural network
Performance model
Model Validation
Material test samples
GH EMBEDDING

Figure 2. Workflow of the generation process of the material performance model.

The process of the generating performance model begins with preparing featured testing material. Next, the material goes through an automatic testing process using robotic arms and image recognition. The massive amount of well-labelled training data containing information on material behaviour generated from previous processes is later fed into a combination of deep and recurrent neural networks to generate a trained performance model. The performance model is then validated before being baked and embedded into grasshopper and blended into the design workflow. A more specific explanation of each step is as follows:

2. Material preparation

In this experiment, 34 pieces of 1-m-long rubber are used to generate testing and validation sample pieces. The flat rubber sheet from which to fabricate the testing samples is 1 metre wide and 50 millimetres thick. Unites are cut from the sheet with a water jet to avoid the accumulation of heat and deformation that normal CNC causes during the drilling process. The section of testing pieces has a diversity of curvatures composed of different combinations of length and angles to ensure that the scope of training samples encompasses the main portion of variables. The material itself is matted black, while the front face is painted in white to add additional contrast for better image recognition for post-processing of the video clips.

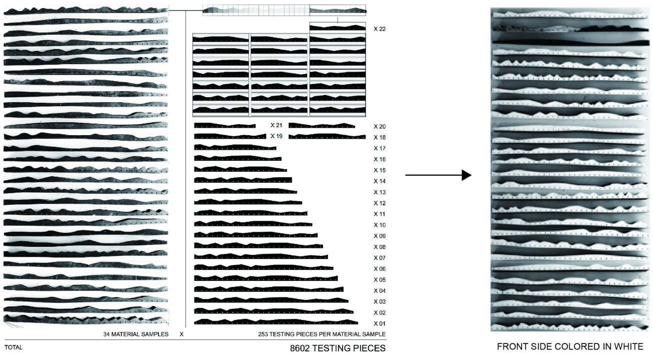


Figure 3. Preparation and cropping of testing samples.

Each rubber has a serious of fixture holes with a diameter of 5 mm evenly spaced at 31.4 mm. The rubber sample will be fixed to the end of the robotic arms using the holes. By fixing the linear rubber samples at different locations, the bending quality of rubber samples could be tested at different lengths with combinations of different curvatures. As the bending of elastomer materials is a highly non-linear process with multiple influencers, the process of testing a single material at different lengths and curvatures to increase the range of training data increases the accuracy of the trained model. From one rubber strip, we manage to generate video clips of 253 testing pieces under the best scenario.

3. Robotic automatic material testing

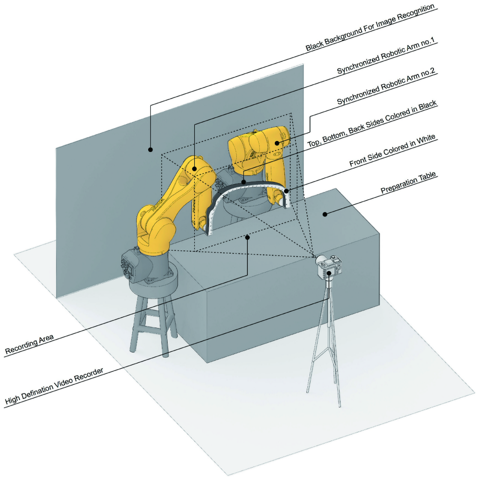

Black Background For Image Recognition
Synchronized Robotic Arm no.1
Synchronized Robotic Arm no.2
Top, Bottom, Back Sides Colored in Black
Front Side Colored in White
Preparation Table
Recording Area
High Defination Video Recorder

Figure 4. Set up of the material test.

After the testing material samples are prepared, the robotic automatic testing environment is set up. Two synchronized robotic arms are located against a black background. One arm is front facing and only rotates in the A6 axis, while the other is located on the side with the end plate facing front and moving horizontally at the same level and plane as the other end plate while rotating along the A6 axis.

In this case, we used KUKA|prc in GhRhino to plan the path and movement of the robots and run simulations of the process. The movements and path are further adjusted according to the length of each piece. In this case, the parameters of the movements are a result of multiple manual experiments to avoid tearing and failure of the materials. This step could be improved with the addition of a mechanical feedback system. A high definition camera located at the front of the set up facing the white front side of the testing material against the black background is used to record in real-time. The video clips captured are then processed and engineered for the training of the neural network.

4. Feature engineering

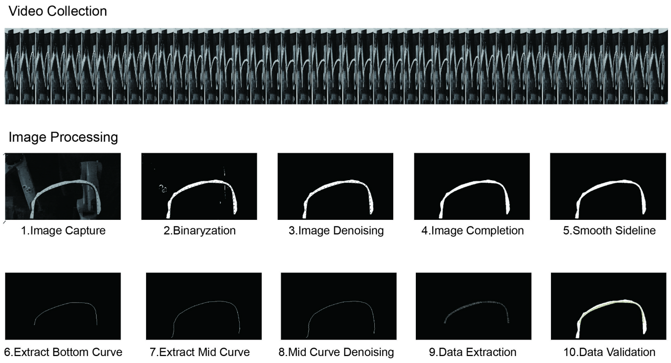


Figure 5. Process of feature engineering.

To prepare the data for machine learning, a crucial step is feature engineering. Particularly, feature engineering in this research means to process the camera-captured video and format it into usable data. This is a twelve-step process with four main phases. The first phase is extracting usable images from video clips including image extracting, greyscale, image crop, and binarization. The second phase is image editing including image denoising, image completion, and smooth sideline. The third phase entails curve extraction including extracting bottom and middle curves as well as mid curve denoising. Last, data validation was employed, including data extraction and data validation.

The goal of feature engineering is to extract the bottom and middle lines of each testing piece. Evenly spaced points are selected on the bottom line. The gradient and distance to the corresponding middle point are recorded and formatted. This distance is considered as the width of the material at each point. Later, the coordinates of the bottom points and corresponding middle points, gradient of the bottom point, and distance data will be used to train the neural network. Due to the length of the paper, we will not address the algorithm and logic of each step in detail.

5. Data formatting and feasibility study

The purpose of this phase is to confirm the relationship between material property and the formal result exist and could be achieved with the deep learning neural network model that uses the training data that we acquired. The intention is to create a material performance model that could programme the sectional properties of elastomer based on a given curve. The relationship to establish is between the height of the section at each selected point and final form of the curve after bending. This is a two-way model, which means we could not only infer the final form of the curved material from given sections of the material with the fixture condition at both ends given but also programme the section of the linear materials based on the final intended form under the force of nature.

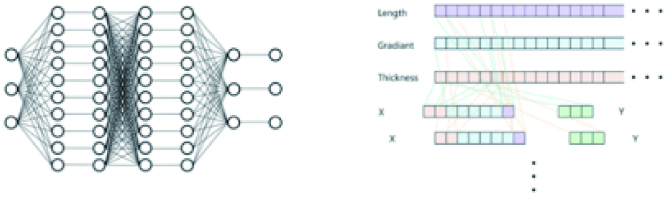


Figure 6. Neural network model structure and data structure.

Before training of the final neural network, data are pre-processed with whitening process, normalization, data wash, and data formatting to generate a ready-to-use data set. After that step, two simple foundation models are developed based on smaller data sets to ensure the existence of such a relationship. One neural network is to build a model that could create the non-linear mapping from given material property and adjacent fixing condition to the spatial curve, proving the capability of deep learning. The other is the inverse model that takes in all information around one point, including the coordinate and the gradient, to predict the sectional height at that point. This model proves that our intended mapping relationship exists. Even though those testing models are relatively simple, a series of parameters of the neural network is adjusted, and the results offer strong reference in the development of the final model.

6. Training the LSTM neural network model

After confirming the feasibility of the model and approach, we decided to use a more complicated and expressive model for training. In this case, recurrent neural network (RNN) is selected because it exceeds at learning the connecting between the horizontal connections in a sequence; for these testing materials, the condition of one point is strongly related to the conditions of its neighbourhood points.

The key feature of the RNN is that the connections between units form a directed cycle, which enabls it to store temporal information. The input of each layer comes from both current input and previous input, which would increase the dependency of individual points to its neighbours. Long Short Term Memory (LSTM) is a special improved type of the RNN model. Even though the RNN model is intended to resolve the problem of data storage, in practice, the model often faces the problem of vanishing gradient. LSTM with “forget gates” prevents such back-propagated problems from vanishing or exploding and can learn from events that occurred many steps earlier. In our case, the forget gates are activated with sigmoid function.

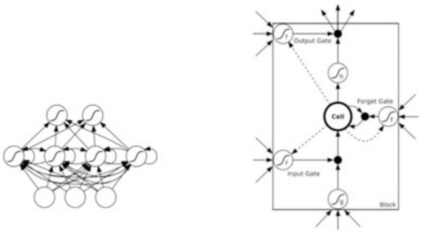


Figure 7. Non-Linear Fields Theories of Mechanics, Handbuch der Physik, Bd.□.

The training data used in this phase is a sequence of vectors stored in an array instead of a single vector. The exported result is a one-dimension array containing the height at each individual position. For a full-length training sample, the input is the gradient information at 80 evenly distributed points and the initial material height at both ends; the output information of the model is the material height at each corresponding point.

7. Evaluation of the generated performance model

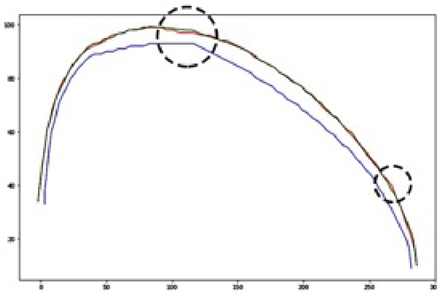


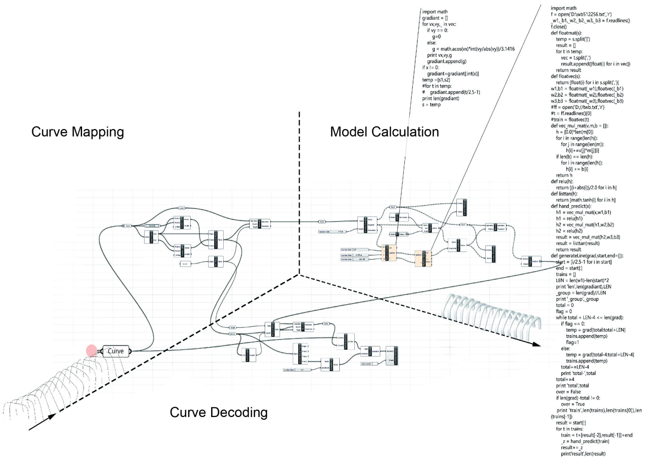
Figure 8. Validation of performance model.

Only 50% of all samples are used for training of the neural network; the other 50% is kept for the evaluation of the generated performance model and loss function. The overall loss is under 0.0015, which is approximately a 5% deviation in scale.

Additionally, we have visualized the prediction result of 54 randomly selected samples and compared it to the result collected from the experiment. In the visualization, the red line is the middle line extracted from the physical experiment, while the green line is the calculated middle line using the generated material model. The model successfully predicts all 54 randomly selected samples with an acceptable accuracy. The overall trend of the predicted outline effectively matches the real result. However, since the bottom line used for input is extracted from pixilated image recognition with some zigzag, the generated upper outline is also slightly zigzagged. The final green line is smoother than the original red lines. This difference is due to the large amount of noise during the extraction of lines from the image sampling and certain flaws on the material from the fabrication process. However, as the green line is the result of the deep learning model, it avoids flaw and problems during image processing and fabrication, resulting in a smoother output.

8. Model conversion

The trained neural network must be further converted to fit into the architectural workflow. In this case, we used Rhino's plugin grasshopper to mediate between the neural network model and the design process. The grasshopper definition is composed of three parts:



```
import math
gradiant = []
for vx,vy,_ in vec:
   if vy == 0:
      g=0
   else:
      g = math.acos(vx)*int(vy/abs(vy))/3.1416
   print vx,vy,g
   gradiant.append(g)
if x != 0:
   gradiant=gradiant[:int(x)]
temp =[s1,s2]
#for t in temp:
#    gradiant.append(t/2.5-1)
print len(gradiant)
s = temp
```

```
import math
f = open('D:\wb512256.txt','r')
_w1,_b1,_w2,_b2,_w3,_b3 = f.readlines()
f.close()
def floatmat(s):
   temp = s.split('|')
   result = []
   for t in temp:
      vec = t.split(',')
      result.append([float(i) for i in vec])
   return result
def floatvec(s):
   return [float(i) for i in s.split(',')]
w1,b1 = floatmat(_w1),floatvec(_b1)
w2,b2 = floatmat(_w2),floatvec(_b2)
w3,b3 = floatmat(_w3),floatvec(_b3)
#ff = open('D://twb.txt','r')
#t = ff.readlines()[0]
#train = floatvec(t)
def vec_mul_mat(v,m,b = []):
   h = [0.0]*len(m[0])
   for i in range(len(h)):
      for j in range(len(m)):
         h[i]+=v[j]*m[j][i]
   if len(b) == len(h):
      for i in range(len(h)):
         h[i] += b[i]
   return h
def relu(h):
   return [(i+abs(i))/2.0 for i in h]
def listtan(h):
   return [math.tanh(i) for i in h]
def hand_predict(x):
   h1 = vec_mul_mat(x,w1,b1)
   h1 = relu(h1)
   h2 = vec_mul_mat(h1,w2,b2)
   h2 = relu(h2)
   result = vec_mul_mat(h2,w3,b3)
   result = listtan(result)
   return result
def generateLine(grad,start,end=[]):
   start = [i/2.5-1 for i in start]
   end = start[:]
   trains = []
   LEN = len(w1)-len(start)*2
   print 'len',len(gradiant),LEN
   _group = len(grad)//LEN
   print '_group',_group
   total = 0
   flag = 0
   while total + LEN-4 <= len(grad):
      if flag == 0:
         temp = grad[total:total+LEN]
         trains.append(temp)
         flag=1
      else:
         temp = grad[total-4:total+LEN-4]
         trains.append(temp)
      total+=LEN-4
      print 'total-',total
   total+=4
   print 'total',total
   over = False
   if len(grad)-total != 0:
      over = True
    print 'train',len(trains),len(trains[0]),len
(trains[-1])
   result = start[:]
   for t in trains:
      train = t+[result[-2],result[-1]]+end
      _z = hand_predict(train)
      result+=_z
      print'result',len(result)
```

Figure 9. Structure of grasshopper definition.

- Curve mapping to transform the input rhino curve into the standard formatted data that the neural network model can process. The key to this step is to check the scale of the curve exported from the model to ensure that it matches the scale of the testing sample and is a planar curve. Then, the curve went through a serious of operation including normalization to become formatted input data for the neural network. In the end, we need to input the initial condition at both end, which will strongly influence the outcome.

- Model calculation is applied to use the trained neural network model to predict the outcome. The current Python module in grasshopper does not support the deep learning framework. Thus, the current solution is to re-write the trained neural network model into GhPython. The neural network model is essentially non-linear mapping and calculation of matrix. After completing the training process by printing out all layer information in Keras, saving it into a file, and extracting the weights of each layer, the weights can be put into GhPython layers, making them ready for forward propagating.
- Curve decoding is used to convert the output data to a 3D model that the designer can work with. In this process, the normalized data must be re-scaled to the original. The original coordinates and gradient provide ground for the translation of the material height at each point with the input of the original plane and coordinates

9. Modelling test

With the tool developed from the previous phase, we are able to translate a curve into the corresponding non-uniform linear piece. The final form is dependent on two factors: the target curve and the initial thickness at both ends. We used such a tool in several other design studies. For example, the tool was used to generate a sectional framework for soft cave forms. In this study, the fabrication of a 2D planar curve could be transferred into fabrication of linear materials with limited thickness. This transfer could easily be accomplished with common fabrication technology and far more material efficiency during fabrication than the 2D planar curve. Additionally, the linear material also occupies less space during transportation and would self-assemble into the designed form by simply bending and fixing both ends. Since we used a smooth curve directly generated from rhino, the output form is far smoother than the curve used for training, which is extracted from the pixilated image.

10. Review and conclusion

Reviewing the series of processes and experiments conducted, there are still limitations to the strategy employed and the performance model generated that could be further improved in the future. The primary emerging constraints are as follows:

- There is a strict limitation in scale and material properties. As we know, material behaviour radically changes as the scale increases, especially since we are utilizing the highly non-linear elasticity of materials. The model trained from smaller scale test samples is only valid for use at similar scales and cannot simply expand to the architecture scale, similar to most mathematical mechanical models. Additionally, it could not be expanded for use of other types of rubber with input of material qualities.
- The model lacks a clearly defined valid region. As this model is trained from a neural network, it is very hard to say where the limit lies. As the complexity, scale, and typology of the target curve deviate from the test samples, the accuracy of the model decreases. However, there is no way to determine the new deviation rate or know at which point the model fails. The only thing we

know is that the more variation and amount of training, the larger the valid region of the model.

- It is difficult to combine with other material and physical models for compound problems. As neural network is a kind of black-box solution complex non-linear problems. It is hard to extract any physical and mathematical rules for use in combination with other physic laws. For example, it is impossible to predict the behaviour of the linear elastomer material under point force with the current model nor the behaviour of a network of such material. New training data need to be added for each scenario. The amount of training data increase exponentially as the complexity increased.

In conclusion, the method combining robotic automation, image recognition, and deep learning neural networks offers an effective and novel approach to solving the behaviour of materials with nonlinear qualities and behaviour under complex forces. Such method and logic could be expanded for more scenarios with minor adjustments, such as replacing the normal camera with a depth camera to collect 3D information. Moreover, the method is a two-way material model, especially effective for material computation, achieving complex forms with minimum and effective programmed materials. However, as this approach is founded on entirely different logic than the typical material models, it has certain problems and restrictions of its own that are worthy of further investigation in the near future.

Acknowledgments

The authors gratefully acknowledge the financial support provided by the National Natural Science Foundation of China (NSFC) (Project No. 51538006)

References

Fleischmann, M., Knippers, J. and Lienhard, J.: 2012, Material behaviour: embedding physical properties in computational design processes, *Architectural Design*, **82**, 271-274.

Graves, A.: 2012, *Supervised Sequence Labelling with Recurrent Neural Networks*, Springer Berlin Heidelberg.

Li, H.: 2012, *Tong ji xue xi fang fa*, Tsinghua University Press, Beijing.

Martins, P.A.L.S., Natal Jorge, R.M. and Ferreira, A.J.M.: 2006, A Comparative Study of Several Material Models for Prediction of Hyperelastic Properties: Application to Silicone-Rubber and Soft Tissues, *Strain*, **42**, 135-147.

Menges, A. and Reichert, S.: 2012, Material capacity: embedded responsiveness, *Architectural Design*, **82**, 52-59.

Moritz, F., Knippers, J., Lienhard, J., Menges, A. and Schleicher, S.: 2012, Material Behaviour: Embedding Physical Properties in Computational Design Processes., *Architectural Design*, **82**, 44-51.

Oxman, N.: 2007, Material computation, *International Journal of Architectural Computing*, **1**, 21-44.

Parker, J.R.: 2011, *Algorithms for image processing and computer vision*, Wiley, Indianapolis, IN.

Russell, S.J. and Norvig, P.: 2002, *Artificial Intelligence: A Modern Approach*, Pearson.

Truesdell, C. and Noll, W. 1965, The Non-linear Fields Theories of□Mechanics, *in* S.S. Antman (ed.), *Handbuch□der□Physik□Bd.□*, Springer-Verlag,Berlin,Heidelberg, ,New York.