5-2008

PHYSICS-BASED SHAPE MORPHING AND PACKING FOR LAYOUT DESIGN

Hong Dong
*Clemson University*, hdong@clemson.edu

Follow this and additional works at: [https://tigerprints.clemson.edu/all_dissertations](https://tigerprints.clemson.edu/all_dissertations)

Part of the [Engineering Mechanics Commons](https://tigerprints.clemson.edu/all_dissertations)

**Recommended Citation**

Dong, Hong. "PHYSICS-BASED SHAPE MORPHING AND PACKING FOR LAYOUT DESIGN" (2008). *All Dissertations*. 188.
[https://tigerprints.clemson.edu/all_dissertations/188](https://tigerprints.clemson.edu/all_dissertations/188)

This Dissertation is brought to you for free and open access by the Dissertations at TigerPrints. It has been accepted for inclusion in All Dissertations by an authorized administrator of TigerPrints. For more information, please contact kokkefe@clemson.edu.
PHYSICS-BASED SHAPE MORPHING AND PACKING FOR LAYOUT DESIGN

A Dissertation
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy
Mechanical Engineering

by
Hong Dong
May 2008

Accepted by:
Dr. Georges M. Fadel, Committee Chair
Dr. Vincent Y. Blouin
Dr. Mary E. Kurz
Dr. Lonny L. Thompson
Dr. Margaret M. Wiecek
ABSTRACT

The packing problem, also named layout design, has found wide applications in the mechanical engineering field. In most cases, the shapes of the objects do not change during the packing process. However, in some applications such as vehicle layout design, shape morphing may be required for some specific components (such as water and fuel reservoirs). The challenge is to fit a component of sufficient size in the available space in a crowded environment (such as the vehicle under-hood) while optimizing the overall performance objectives of the vehicle and improving design efficiency. This work is focused on incorporating component shape design into the layout design process, i.e. finding the optimal locations and orientations of all the components within a specified volume, as well as the suitable shapes of selected ones.

The first major research issue is to identify how to efficiently and accurately morph the shapes of components respecting the functional constraints. Morphing methods depend on the geometrical representation of the components. The traditional parametric representation may lend itself easily to modification, but it relies on the assumption that the final approximate shape of the object is known, and therefore, the morphing freedom is very limited. To morph objects whose shape can be changed arbitrarily in layout design, a mesh based morphing method based on a mass-spring physical model is developed. For this method, there is no need to explicitly specify the deformations and the shape morphing freedom is not confined.

The second research issue is how to incorporate component shape design into a layout design process. Handling the complete problem at once may be beyond our reach,
therefore decomposition and multilevel approaches are used. One bi-level approach is
tailored specially for the layout design with mass-spring physical model based shape
morphing. At the system level, a genetic algorithm (GA) is applied to find the
approximate positions and orientations of the objects, while at the sub-system or
component level, morphing is accomplished for select components. Adjacent objects
expanding optimization is performed to find how the morphable objects push away their
adjacent objects during the volume expansion. A gradient based local search is used for
local perturbation of the positions of the objects after shape morphing.

Although different packing applications may have different objectives and
constraints, they all share some common issues. These include CAD model preprocessing
for packing purpose, data format translation during the packing process if performance
evaluation and morphing use different representation methods, efficiency of collision
detection methods, etc. These common issues are all brought together under the
framework of a general methodology for layout design with shape morphing.

Finally, practical examples of vehicle under-hood/underbody layout design with the
mass-spring physical model based shape morphing are demonstrated to illustrate the
proposed approach before concluding and proposing continuing work.
DEDICATION

I dedicate this work to my parents, my brother and Guang.
ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Fadel, for his patient and high level advice and guidance. He gave me full backup in pursuing my own research topic. His high expectations and continuous support are greatly appreciated. It’s not enough only to say thank you to him. I am also grateful to my committee members: Dr. Blouin, Dr. Kurz, Dr. Thompson, and Dr. Wiecek. They are all very supportive and actively helped in the completion of this dissertation.

Special thanks to the former CREDO member Yi Miao and current member Santosh Tiwari. Their kind cooperation and support helped me during my study in Clemson.

I also want to thank the Automotive Research Center (ARC), a U.S. Army TACOM Center of Excellence for Modeling and Simulation of Ground Vehicles at the University of Michigan, for supporting this research.

Finally, I would like to thank all others who gave me help.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>ii</td>
</tr>
<tr>
<td>DEDICATION</td>
<td>iv</td>
</tr>
<tr>
<td>ACKNOWLEDGMENTS</td>
<td>v</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>ix</td>
</tr>
<tr>
<td>CHAPTER</td>
<td></td>
</tr>
<tr>
<td>1. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>Research Objectives</td>
<td>5</td>
</tr>
<tr>
<td>Outline</td>
<td>6</td>
</tr>
<tr>
<td>2. LITERATURE REVIEW</td>
<td>9</td>
</tr>
<tr>
<td>Optimization Algorithms for the Packing Problem</td>
<td>10</td>
</tr>
<tr>
<td>Geometric Representations</td>
<td>16</td>
</tr>
<tr>
<td>Layout Design with Shape Morphing</td>
<td>21</td>
</tr>
<tr>
<td>Morphing Methods</td>
<td>21</td>
</tr>
<tr>
<td>Mesh Based Morphing Software</td>
<td>29</td>
</tr>
<tr>
<td>Summary</td>
<td>33</td>
</tr>
<tr>
<td>3. A PHYSICAL MODEL BASED PACKING ALGORITHM FOR COMPACT PACKING -- RUBBER BAND ANALOGY</td>
<td>35</td>
</tr>
<tr>
<td>Methodology</td>
<td>36</td>
</tr>
<tr>
<td>CAD Model and Intersection Detection</td>
<td>37</td>
</tr>
<tr>
<td>Problem Formulation</td>
<td>38</td>
</tr>
<tr>
<td>Algorithm</td>
<td>39</td>
</tr>
<tr>
<td>Improving Compactness</td>
<td>46</td>
</tr>
<tr>
<td>Examples</td>
<td>50</td>
</tr>
<tr>
<td>Comparison with Gradient-based Methods</td>
<td>56</td>
</tr>
<tr>
<td>Extension of the Rubber-band Analogy Physical Model</td>
<td>57</td>
</tr>
<tr>
<td>Summary</td>
<td>58</td>
</tr>
<tr>
<td>4. CHARACTERISTICS OF MORPHING IN LAYOUT DESIGN</td>
<td>59</td>
</tr>
<tr>
<td>Types of Components</td>
<td>59</td>
</tr>
<tr>
<td>Characteristics of Shape Morphing in Layout Design</td>
<td>61</td>
</tr>
</tbody>
</table>
Table of contents (Continued)

Requirements on Morphing Methods for Layout Design.......................... 62

5. A PHYSICAL MODEL BASED SHAPE MORPHING METHODS FOR LAYOUT DESIGN — MASS-SPRING MESH BASED MORPHING..... 64

<table>
<thead>
<tr>
<th>Mass-spring Model</th>
<th>65</th>
</tr>
</thead>
<tbody>
<tr>
<td>Force Applied on Mass Points</td>
<td>66</td>
</tr>
<tr>
<td>Volume of the Object</td>
<td>70</td>
</tr>
<tr>
<td>Integrating Newton’s Equation with Euler method</td>
<td>72</td>
</tr>
<tr>
<td>Collision Detection and Response</td>
<td>74</td>
</tr>
<tr>
<td>Iteration Procedure</td>
<td>80</td>
</tr>
<tr>
<td>Example</td>
<td>81</td>
</tr>
<tr>
<td>Summary</td>
<td>84</td>
</tr>
</tbody>
</table>

6. BI-LEVEL APPROACH FOR LAYOUT DESIGN WITH MASS-SPRING PHYSICAL MODEL BASED SHAPE MORPHING................................. 85

| Decomposition of Large Scale Problem | 85 |
| Problem Formulation for Layout Design with Mass-spring Physical Model Based Shape Morphing | 87 |
| The Bi-level Iteration Process | 90 |
| Detailed Problem Formulation and Corresponding Optimization Algorithms | 93 |
| Two Important CAD Aspects in the Bi-level Iteration Process | 97 |
| Example | 99 |
| Summary | 115 |

7. GENERAL METHODOLOGY FOR LAYOUT DESIGN WITH SHAPE MORPHING ........................................................................... 116

| The General Layout Design with Shape Morphing Process | 116 |
| Generic Model for Layout Design with Shape Morphing | 118 |
| CAD Model Preprocessing | 122 |
| Preprocessing with Surface Wrapping | 122 |
| Re-mesh the Morphed Object | 128 |
| Collision Detection Efficiency | 130 |
| Summary | 132 |

8. VEHICLE LAYOUT DESIGN WITH SHAPE MORPHING .................... 133
Table of contents (Continued)

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ford Taurus Underbody Fuel Tank Design with Mass-spring Physical Model</td>
<td>134</td>
</tr>
<tr>
<td>Based Morphing</td>
<td></td>
</tr>
<tr>
<td>Ford Taurus Under-hood Layout Design with Mass-spring Physical Model</td>
<td>138</td>
</tr>
<tr>
<td>Based Morphing</td>
<td></td>
</tr>
<tr>
<td>Summary</td>
<td>161</td>
</tr>
<tr>
<td>9. CONCLUSION</td>
<td>162</td>
</tr>
<tr>
<td>Overview</td>
<td>162</td>
</tr>
<tr>
<td>Future Work</td>
<td>165</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>167</td>
</tr>
</tbody>
</table>
LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 Comparison of Different Morphing Methods</td>
<td>32</td>
</tr>
<tr>
<td>4.1 Qualification of Four Specific Morphing Methods</td>
<td>63</td>
</tr>
<tr>
<td>5.1 Parameters of Mass-spring Model</td>
<td>82</td>
</tr>
<tr>
<td>6.1 Objective Values after Each Sub-search of Solution A</td>
<td>105</td>
</tr>
<tr>
<td>6.2 Objective Values after Each sub-search of Solution B</td>
<td>110</td>
</tr>
<tr>
<td>6.3 Objective Values after Each Sub-search of Solution C</td>
<td>115</td>
</tr>
<tr>
<td>7.1 Computation Time for Collision Detection</td>
<td>131</td>
</tr>
<tr>
<td>8.1 Under-hood Components and Degree of Freedom</td>
<td>140</td>
</tr>
<tr>
<td>8.2 Weights of Accessibility</td>
<td>143</td>
</tr>
<tr>
<td>8.3 Objective Values after Each Sub-search of Solution A</td>
<td>148</td>
</tr>
<tr>
<td>8.4 Objective Values after Each Sub-search of Solution B</td>
<td>154</td>
</tr>
<tr>
<td>8.5 Objective Values after Each Sub-search of Solution C</td>
<td>159</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 Applications of packing problems in industry</td>
<td>2</td>
</tr>
<tr>
<td>1.2 Concurrent layout design and component shape design</td>
<td>5</td>
</tr>
<tr>
<td>2.1 Parametric beam design based on commercial CAD system</td>
<td>23</td>
</tr>
<tr>
<td>2.2 Morph the green circle by scaling and Boolean difference</td>
<td>25</td>
</tr>
<tr>
<td>2.3 Morph object A by scaling the cells (2D) along y-axis</td>
<td>26</td>
</tr>
<tr>
<td>2.4 Morphing use transformation operators such as swap, collapse or split</td>
<td>27</td>
</tr>
<tr>
<td>2.5 2D mesh of a circle and 3D mesh of a ball</td>
<td>28</td>
</tr>
<tr>
<td>2.6 Shape morphing in ANSA</td>
<td>30</td>
</tr>
<tr>
<td>3.1 Flow chart of the rubber band packing algorithm</td>
<td>39</td>
</tr>
<tr>
<td>3.2 (a) Convex polygon in 2D  (b) Convex polyhedron in 3D</td>
<td>40</td>
</tr>
<tr>
<td>3.3 Calculating translation vector for one object</td>
<td>41</td>
</tr>
<tr>
<td>3.4 Possible movements for surface-surface contact</td>
<td>43</td>
</tr>
<tr>
<td>3.5 Converting surface-vertex contact to surface-surface contact</td>
<td>45</td>
</tr>
<tr>
<td>3.6 Converting edge-edge contact to surface-surface contact</td>
<td>46</td>
</tr>
<tr>
<td>3.7 Object B rotates freely from position 1 to 5</td>
<td>48</td>
</tr>
<tr>
<td>3.8 Volume changes with respect to rotation positions</td>
<td>49</td>
</tr>
<tr>
<td>3.9 Packing 7 cubes with translation, rotation and sliding</td>
<td>51</td>
</tr>
<tr>
<td>3.10 Packing three polyhedrons without volume relaxation</td>
<td>53</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>3.11</td>
<td>Packing three polyhedrons with volume relaxation</td>
</tr>
<tr>
<td>3.12</td>
<td>Packing four polyhedrons without temporary retraction</td>
</tr>
<tr>
<td>3.13</td>
<td>Packing four polyhedrons with temporary retraction</td>
</tr>
<tr>
<td>4.1</td>
<td>Components morphable to any shape</td>
</tr>
<tr>
<td>4.2</td>
<td>Interpolation frames from initial shape to target shape</td>
</tr>
<tr>
<td>5.1</td>
<td>2D mesh of a circle and 3D mesh of a ball</td>
</tr>
<tr>
<td>5.2</td>
<td>Forces applied on mass points</td>
</tr>
<tr>
<td>5.3</td>
<td>A spring with ends located at vertices $V_i$ and $V_{i+1}$</td>
</tr>
<tr>
<td>5.4</td>
<td>Adding pressure inside the objects</td>
</tr>
<tr>
<td>5.5</td>
<td>The $i^{th}$ discrete triangular surface</td>
</tr>
<tr>
<td>5.6</td>
<td>Euler integration</td>
</tr>
<tr>
<td>5.7</td>
<td>The penetrating node</td>
</tr>
<tr>
<td>5.8</td>
<td>Retracting the penetrating node to the surface of the penetrated object</td>
</tr>
<tr>
<td>5.9</td>
<td>Velocity before and after collision</td>
</tr>
<tr>
<td>5.10</td>
<td>Reaction force</td>
</tr>
<tr>
<td>5.11</td>
<td>Iteration procedure of mass-spring physical model based morphing</td>
</tr>
<tr>
<td>5.12</td>
<td>Initial shape of the ball</td>
</tr>
<tr>
<td>5.13</td>
<td>One interval step during expansion</td>
</tr>
<tr>
<td>5.14</td>
<td>The final morphing result</td>
</tr>
</tbody>
</table>
List of Figures (Continued)

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1 Multilevel decomposition of a large scale problem</td>
<td>87</td>
</tr>
<tr>
<td>6.2 Flow chart of the bi-level iteration process</td>
<td>91</td>
</tr>
<tr>
<td>6.3 Identifying the status that the target volume cannot be reached</td>
<td>99</td>
</tr>
<tr>
<td>6.4 Packing components in the available packing space</td>
<td>100</td>
</tr>
<tr>
<td>6.5 Pareto front of system level global search</td>
<td>102</td>
</tr>
<tr>
<td>6.6 System level global search result of solution A</td>
<td>103</td>
</tr>
<tr>
<td>6.7 System level expanding optimization and subsequent component level volume expansion result of solution A</td>
<td>104</td>
</tr>
<tr>
<td>6.8 System level local search result of solution A</td>
<td>105</td>
</tr>
<tr>
<td>6.9 System level global search result of solution B</td>
<td>106</td>
</tr>
<tr>
<td>6.10 System level expanding optimization and subsequent component level volume expansion result of solution B</td>
<td>109</td>
</tr>
<tr>
<td>6.11 System level local search result of solution B</td>
<td>110</td>
</tr>
<tr>
<td>6.12 System level global search result of solution C</td>
<td>111</td>
</tr>
<tr>
<td>6.13 System level expanding optimization and subsequent component level volume expansion result of solution C</td>
<td>113</td>
</tr>
<tr>
<td>6.14 System level local search result of solution C</td>
<td>114</td>
</tr>
<tr>
<td>7.1 General layout design with shape morphing process</td>
<td>116</td>
</tr>
<tr>
<td>7.2 Generic model for layout design with shape morphing</td>
<td>118</td>
</tr>
<tr>
<td>7.3 Algorithm for converting tessellated model to ACIS solid model</td>
<td>121</td>
</tr>
<tr>
<td>7.4 Original detailed engine model</td>
<td>124</td>
</tr>
<tr>
<td>7.5 Outer shell of the original engine model with different precision control</td>
<td>126</td>
</tr>
</tbody>
</table>
List of Figures (Continued)

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.6</td>
<td>The original disconnected sheet metal model of a trunk</td>
<td>126</td>
</tr>
<tr>
<td>7.7</td>
<td>Outer wrap to close the hole with large element size</td>
<td>127</td>
</tr>
<tr>
<td>7.8</td>
<td>Packing space obtained by internal wrap function</td>
<td>128</td>
</tr>
<tr>
<td>7.9</td>
<td>Construct the Delaunay triangulation by flipping</td>
<td>129</td>
</tr>
<tr>
<td>7.10</td>
<td>Re-mesh with Delaunay triangulation</td>
<td>130</td>
</tr>
<tr>
<td>8.1</td>
<td>Ford Taurus under-hood system</td>
<td>133</td>
</tr>
<tr>
<td>8.2</td>
<td>Ford Taurus underbody system</td>
<td>135</td>
</tr>
<tr>
<td>8.3</td>
<td>Approximate packing space for fuel tank</td>
<td>135</td>
</tr>
<tr>
<td>8.4</td>
<td>Morphed fuel tank with a curved pipe</td>
<td>136</td>
</tr>
<tr>
<td>8.5</td>
<td>Morphed fuel tank with a straight pipe</td>
<td>137</td>
</tr>
<tr>
<td>8.6</td>
<td>Saddle-shaped fuel tank of Ford Mustang</td>
<td>138</td>
</tr>
<tr>
<td>8.7</td>
<td>Packing components in the under-hood compartment</td>
<td>140</td>
</tr>
<tr>
<td>8.8</td>
<td>Definition of accessibility along a specific direction</td>
<td>142</td>
</tr>
<tr>
<td>8.9</td>
<td>Pareto front of system level global search, maintainability = 10</td>
<td>144</td>
</tr>
<tr>
<td>8.10</td>
<td>System level global search result of solution A</td>
<td>145</td>
</tr>
<tr>
<td>8.11</td>
<td>System level expanding optimization and subsequent component level volume expansion result of solution A</td>
<td>147</td>
</tr>
<tr>
<td>8.12</td>
<td>System level local search result of solution A</td>
<td>149</td>
</tr>
<tr>
<td>8.13</td>
<td>System level global search result of solution B</td>
<td>150</td>
</tr>
<tr>
<td>8.14</td>
<td>System level expanding optimization and subsequent component level volume expansion result of solution B</td>
<td>153</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
<td>------</td>
</tr>
<tr>
<td>8.15</td>
<td>System level local search result of solution B</td>
<td>155</td>
</tr>
<tr>
<td>8.16</td>
<td>System level global search result of solution C</td>
<td>156</td>
</tr>
<tr>
<td>8.17</td>
<td>System level expanding optimization and subsequent component level volume expansion result of solution C</td>
<td>158</td>
</tr>
<tr>
<td>8.18</td>
<td>System level local search result of solution C</td>
<td>160</td>
</tr>
</tbody>
</table>
CHAPTER 1 INTRODUCTION

The packing problem is also named layout design, configuration design or packaging in the literature. There are various applications of packing problems (Fig.1.1) in industry such as stock cutting, luggage packing, payloads packing, electronics circuit board layout, factory layout/piping, mechanical component layout, vehicle/submarine/aircraft layout design, etc. Each application has its special focus. Generally speaking, packing optimization deals with how to arrange a set of components in available spaces such that given objectives can be optimized without violating any space or performance constraints.

Classified by the dimensions, packing problems can be divided into one, two, two and half or three-dimensional problems. The known basic packing problem, one-dimensional bin packing, consists of packing a given set of items of different sizes into a minimum number of equal size bins. Traditional circuit board layout, a two-dimensional problem, deals with how to place the micro-cell blocks on a planar site, where there is some defined connectivity between the blocks, such that the area of the planar site and the wire length can be minimized. Circuit board layout can today be considered 2½ dimensional since components can be placed on several planes and connected to each other. Many other applications belong to the three-dimensional problems such as luggage packing, payload packing and vehicle/submarine/aircraft layout design where the designer has the full freedom of placing objects anywhere in space as long as the overlap between components does not exist.
Figure 1.1 Applications of packing problems in industry

Classified by the objectives, packing problems can be divided into two groups: single objective and multi-objective. Bin packing, stock-cutting, and luggage packing are all single objective problems. The objective of these problems is to minimize the un-used space, i.e. maximize the space or material utilization. Circuit board layout, payloads packing, mechanical component layout, and vehicle/submarine/aircraft layout design are multi-objectives problems. Payload packing involves an additional objective besides mere maximizing compactness. It concerns how to place the center of inertia of the loaded containers as close as possible to a specified location [Wodziak and Fadel, 1994]. This is important for trucks, planes and boats, in order to reduce their fuel consumption and increase the safety of the trip by minimizing the propensity to roll over. Vehicle layout design deals with how to arrange components of a vehicle to achieve good dynamic performance, maintainability and survivability (military vehicles) besides...
packing the components compactly [Miao et al., 2003]. Mechanical component layout is concerned with arranging the components of a mechanical system such that the required functions can be performed well, while the components are packed tightly. For example in a heat pump design, the component locations are optimized such that the routing cost can be reduced and the vibration performance can be improved [Szykman et al., 1998].

In most of the above applications, the shapes of the objects do not change during the packing process. For example, the shapes of the components in bin packing, luggage packing or payload packing are fixed. However, in some other applications such as vehicle layout design, shape morphing may be required for some specific components (such as water and fuel reservoirs), to fit them in the available space of sufficient size in a crowded environment (such as the vehicle under-hood or underbody) while optimizing the overall performance objectives and improving design efficiency.

Vehicle layout design is a very important stage in the whole vehicle system design process. It is performed when the component design is frozen (by the parts supplier) therefore, the shapes of the components have already been determined. The component shape design and layout design are usually performed by two groups of designers with very limited collaboration. With this quasi-sequential approach, problems often surface late in the design cycle. On one hand, during the configuration design stage, the shapes of the components may need to be modified such that the components can be placed in optimal positions or be fitted into some available space. However, on the other hand, the component designer does not incorporate the requirements from the layout design stage during component design. Therefore, the solutions generated by the component designer
are not necessarily good solutions for the overall performance of the whole system. When
the layout designer cannot fit a component in a configuration, or if the target objectives
are not good enough with the current component shape, this component must be
redesigned. The lack of collaboration between these two stages reduces design efficiency,
increases design cost and prevents the search for a better design. Thus, it is imperative to
adopt a concurrent process instead of a sequential design approach [Syan and Menon,
1994], i.e. incorporate the component shape design into the layout design process (Fig
1.2). Thus the concept of layout design with shape morphing can be introduced. “Layout
design with shape morphing is finding the optimal locations and orientations of all the
components as well as the suitable shapes of selected ones such that specified
performance objectives can be optimized without violating any space or functionality
constraints”.

Currently, vehicle layout design is based on prior experience and usually carried
out manually. It is traditionally performed in a variant design mode, where past histories
provide starting points for incremental changes. The process is time consuming and it is
difficult to get an optimal design if such an optimum exists. However, with the
proliferation of hybrid concepts and completely new components systems, there is a need
to formalize the process to assist the engineer. The frequent changes in design of
automotive components and body require a tool that can rapidly and automatically
generate a series of alternative designs such that the whole design cycle could be shorter.
Based on the above presentation, the following research questions are posed, and identifying the responses is the goal of the dissertation.

The first research question is how can one efficiently and accurately morph the shapes of components respecting shape and functional constraints? Morphing methods depend on the geometrical representation of the components. A parametric representation may lend itself easily to modification, but how is the shape allowed to morph in an irregular manner to fit the available space? A mesh based method may be much more flexible in the morphing process, but to be able to control the huge number of nodal points may be too computationally costly. Therefore, an appropriate morphing method needs to be identified.
The second research question is how should component shape design be incorporated into a layout design process? A systematic approach is sought to solve such a complex problem. Handling the complete problem at once may be beyond our reach. Could decomposition and multilevel approaches be used to handle such a problem?

The third research question is related to the application at hand, i.e. can we solve a problem such as the vehicle under-hood/underbody layout design with shape morphing? What are the appropriate algorithms and which techniques are able to accomplish this task during the design process?

Outline

The dissertation is divided into the following chapters.

Chapter 2 is the literature review, which discusses some basic issues in packing optimization. First, different optimization algorithms such as gradient based methods, genetic algorithms, simulated annealing and their applications to the packing problem are reviewed. Then, the CAD representation methods and their impact on the related issues such as collision detection efficiency and shape morphing ability are discussed. In addition to these, the current status of layout design with shape morphing, as well as different morphing methods and their strengths and drawbacks are presented.

Chapter 3 presents a novel packing algorithm based on a rubber band analogy. This method solves packing problems by simulating the physical movements of a set of objects wrapped by a rubber band in the case of two-dimensional problems or by a rubber balloon in the case of three-dimensional problems. The method is implemented and applied for three-dimensional arbitrarily shaped objects. This elastic analogy technique can guarantee
locally optimal solutions, and displays a very intuitive behavior. This rubber-band model is further extended for morphing purpose in chapter 5.

Chapter 4 analyses the characteristic of shape morphing during layout design by comparing it with computer graphics morphing. The special function requirements for morphing methods suitable for packing purpose are derived. Furthermore, the components involved in layout design are classified according to their shape changing freedom. The mass-spring physical model based morphing is chosen for further investigation.

Chapter 5 focuses on the mass-spring physical model based shape morphing method. Mass-spring model is an extension of the rubber-band model in chapter 3. In the case of rubber-band model, the rubber band acts as springs to bring objects together to form a compact packing. The corner points of the objects form the nodes of the rubber band or balloon surface. While for the mass-spring model, the balloon surface (surface of the deformable object) is modeled as a collection of mass points (located on the nodes of the surface mesh) connected with springs (located on the edges of the surface mesh). It is proposed to inflate the rubber band or balloon to occupy the desired volume. The mass points forming the surface nodes may be subject to spring force and pressure force. By simulating movements of the mass points, which are nodes of the surface mesh of an object, the shape of the object is therefore directly morphed. A special collision detection procedure between deformable object and fixed shape object or enclosure is proposed. The response of the morphable object after collision is further investigated.

Chapter 6 presents the bi-level approach for the layout design with shape morphing problem. Vehicle layout design with shape morphing is a multi-objective
problem with a large number of design variables. To make the problem easier to solve, the original problem is decomposed into system and component level problems. The bi-level formulation is specially tailored for mass-spring physical model based morphing. An incremental shape morphing strategy is proposed.

Chapter 7 presents a general design methodology for the layout design with shape morphing problem. First, the general layout design with shape morphing process is described. The Generic model for layout design with shape morphing is presented, which includes four functional parts: performance evaluation, object layout, and shape morphing and data format translation. The packing related issues such as CAD data preprocessing, re-meshing after shape morphing, and how to improve collision detection efficiency are also discussed.

Chapter 8 focuses on the applications to vehicle layout design. To demonstrate the developed algorithms, the Ford Taurus underbody fuel tank design and under-hood layout design with mass-spring physical model based morphing are presented.

Chapter 9 summarizes this research, draws conclusions, and proposes directions for future work.
CHAPTER 2 LITERATURE REVIEW

This chapter reviews several important issues in packing optimization and covers the relevant literature review. The packing problem is an NP-hard problem [Coffman et al., 1997]. To find optimal solutions to this type of problem, typically a computationally-intensive, exhaustive analysis is required, in which all possible outcomes are tested. For engineering applications, obviously it is impossible and not wise to test all the possible solutions. Therefore, an appropriate algorithm needs to be identified to obtain an acceptable solution in a reasonable time frame. Different optimization algorithms such as gradient based methods, genetic algorithms, simulated annealing and their applications to the packing problem are reviewed. These algorithms are the actuators of the packing process and are therefore critical.

Second, an important issue is what kind of representation methods should be used to enable the tie to the geometry. Geometric representation methods directly affect the collision detection efficiency and the possible shape morphing methods. Therefore, several widely used CAD representation methods are compared in terms of collision detection efficiency, accuracy control ability, and shape morphing ability.

Next, changing the shapes of the objects during the packing process is a relatively new concept and only a few applications have incorporated this idea. Thus, the current research status of layout design with shape morphing is investigated. The most important aspect of this research is how to efficiently and accurately morph the shapes of components respecting the functional constraints during a layout process. Several
morphing methods: parameterization-based morphing, scaling coupled with Boolean
difference morphing method, Octree-based morphing, and mesh-based morphing are
investigated. Their strengths and drawbacks are compared.

The aim of this chapter is to identify the critical building blocks needed to solve the
problem at hand.

Optimization Algorithms for the Packing Problem

The packing problem is an NP-hard problem. The classification means that there
does not exist a known algorithm that produces an optimal packing in a polynomial
function of time. The solution of NP-hard problems requires (in the worst case) an
exhaustive search, i.e. every possible case has to be tried to find the solution. The
complexity of an exhaustive search is \( O(n!) \). As the size of the problem increases, the
complexity of the exhaustive algorithms increases exponentially. This prevents the use of
exhaustive search methods to find optimal solutions of problems of reasonable or large
size. The alternative approach to solve an NP-hard problem is to develop algorithms with
polynomial time complexity that generate near optimal solutions [Garey and Johnson,
1979]. By giving up solution quality, computational efficiency can be gained. Note that
the theory of NP-hard does not stipulate that it is hard to get close to the answer only that
it is hard to get to the optimal answer.

There are many known algorithms used to approximate the optimal solutions of the
packing problem. They can be divided into two classes: deterministic and
non-deterministic. Deterministic algorithms are algorithms with uniquely defined results.
Their output is predictable for a certain input. Gradient-based algorithms and heuristic
rule-based algorithms are deterministic algorithms. Non-deterministic algorithms are allowed to contain operations whose outcomes are limited to a given set of possibilities instead of being uniquely defined. The non-deterministic algorithms include genetic algorithms, simulated annealing, and some hybrid methods. In the following sections, a brief review of algorithms applied to packing problems is presented.

**Deterministic Methods**

*Gradient-based Algorithms*

Gradient-based algorithms use gradient information to guide the search for optimal solutions. However, for most packing problems, the explicit gradients of the objective functions or constraints are not available. They cannot be obtained by taking the derivatives of these functions with respect to all the design variables, mainly because the objectives are not explicit. In such cases, the gradients could be calculated numerically by the finite difference method. However, using finite differences may cause two problems. First, calculating one gradient value with respect to one design variable needs at least two evaluations of the function, which is not efficient with the large number of variables. Second, finite approximations are not accurate and cause numerical errors, therefore may mislead the search [Cagan et al, 2002]. Landon and Balling [1994] apply a gradient-based method to optimize a 3D packing problem according to a mass property criterion. They propose an explicit gradient calculation method to substitute for the numerical finite difference method. With the explicit gradients, an enormous amount of expensive analyses can be bypassed, and therefore, computational time is saved. However, the biggest issue for a gradient algorithm is that the solution quality is highly dependent on
the initial starting point. This type of algorithm converges to the nearest local optimum to the initial design. The objective space of the packing problem has been characterized as fractal like [Epstein et al., 2001] [Sorkin, 1991]. A fractal is an object or quantity that displays self-similarity on all scales. For the fractal-like objective space of packing problem, a deterministic algorithm is bound to frequently end up trapped in an inferior local optimum.

*Heuristic rule-based algorithms*

Heuristics are rules capable of finding a solution fast but with no guarantee that the solution is the best one. Heuristic rules usually come from commonsense. For example, First-fit, Next-fit and Best-fit rules for the 1D bin packing problem [Coffman et al., 1984] are commonsense rules that mimic human thinking. First-fit simply fits the current object into the first possible bin (the bins are checked in some fixed order). Best-fit, a little more sophisticated, tries to find the bin that will leave the smallest remaining space. Next-fit is to open a bin and place the items into it in the order they appear in the list. If an item on the list does not fit into the open bin, this bin is closed permanently. A new one is opened, and the remaining items in the list are packed sequentially. Heuristic rules are easy to implement and efficient for specific problems. However, they usually have special requirements on the geometry of the objects. The shape of the objects should be rectangular, cuboids or be of regular shape. Moreover it is not easy for this method to incorporate other optimization objectives besides compactness. This class of methods is not suitable for solving general packing problems.
Non-Deterministic Methods

Genetic Algorithms

The Genetic Algorithm (GA) uses a direct analogy to natural evolution behavior. It works with a population of individuals, each representing a possible solution to a given problem. Each individual is assigned a fitness score according to how good this solution is with respect to some objectives. The individuals with high fitness are given opportunities to reproduce, by cross breeding with other individuals in the population. Then, new individuals emerge as offspring, with features from each parent. The individuals with low fitness are less likely to be selected for reproduction and die out. Therefore, a whole new population of possible solutions is produced by selecting the best individuals from the current generation and by mating them. This new generation contains a higher proportion of the characteristic possessed by the good members of the previous generation. Over many generations, good characteristics are spread throughout the population. The most promising areas of the design space are explored by the mating of the more fit individuals and finally the population converges to an optimal solution if the GA is properly designed. The strength of the GA is that it is robust and can deal with a variety of problems that are difficult for other methods to solve, such as problems whose design space is fractal. The GA does not guarantee to find the global optimum solution to a problem, but it is good at finding acceptable good solutions in an acceptable amount of time. Also, GAs can handle discrete and continuous variables and can be tailored for the application at hand by judiciously manipulating the various operators and
the encoding of the design variables. Furthermore, GAs handle easily and efficiently multi-objective problems, a critical requirement in the general packing problem.

Wodziack and Fadel [1994] use a conventional GA to solve a 2½ dimensional truck-trailer packing problem where the goal is to obtain a specific center of gravity for the trailer where packing as many boxes as possible. Grignon and Fadel [1999] use a GA working with population sets instead of a population of individual points to solve a mechanical system packing problem with multiple objectives such as compactness, accessibility and maintainability. Miao et al. [2003] use a Non-dominated Sorting GA to solve a 3D midsize truck configuration design problem where compactness, dynamic behavior and maintainability are optimized. Miao and Fadel [2005] develop a Packing GA with a new encoding method and GA operators tailored specially for the packing problems, which improves average compactness and percentage of acceptable layouts significantly.

*Simulated Annealing*

Simulated Annealing is a stochastic algorithm that is based on the analogy between the annealing process, in which a metal cools and freezes into a minimum energy crystalline structure, and the search for a minimum in optimization problems [Cagan et al., 2002]. The optimization process with this algorithm can be described as follows. An initial state is randomly chosen from the design space, and the objective function is evaluated for this state. A new state is generated by applying a move or an operator, and the new state is evaluated. If the new state is better than the previous state, it is accepted; otherwise, it may still be accepted with a certain probability. The probability is a function
of a decreasing parameter called temperature, based on an analogy with the annealing of metals [Cagan et al., 2002], given by:

\[ P_{accept} = e^{\frac{-\Delta C}{T}} , \]

where \( \Delta C \) is the variation of objective function values, and \( T \) is the current temperature.

The search process is random during the initial stage, resulting in a broad exploration of the design space. The search process becomes more deterministic as the probability of accepting inferior states decreases, allowing the algorithm to converge to an optimal solution. This method can avoid trapping solutions in a local optimum. Szykman and Cagan [1995] apply simulated annealing to solve a 3D mechanical component layout problem. The move sets include translation, rotation constrained to multiples of 90 degrees and swap. An adaptive annealing schedule is used to control the parameter and a probabilistic selection strategy is used to choose moves. Campbell et al. [1997] use simulated annealing to perform three-dimensional electronic component layout while incorporating constraints related to thermal performance. A hierarchical heat transfer analysis is developed, which is used in conjunction with the simulated annealing algorithm to produce final layout configurations that are densely packed and operate within specified temperature ranges.

**Hybrid Algorithm**

There also exist some hybrid methods that may combine two or more search methods to exploit the relative strength of each method and undermine its weakness.
Usually stochastic algorithms are hybridized with some deterministic algorithms. The stochastic algorithm is used as a global searcher to get into the promising solution basins where the local deterministic searchers can do their job. The combination usually results in a more robust algorithm that provides faster and better solutions. Hopper and Turton [1999] combined GA and a Bottom-Left-Fill heuristic rule to solve the 2D rectangular packing problem. Smith et al. [1996] combined SA and a Knowledge-based system to solve a spatial layout problem with conflicting objectives.

In this research, the GA is chosen as the global search algorithm for packing optimization. The strengths of GA make itself a good candidate for solving multi-modal problems with discrete design variables such as the packing problem. Furthermore, many engineering applications such as vehicle layout design are multi-objective problems. GAs can handle easily and efficiently multi-objective problems.

In packing optimization, besides the optimization algorithms issue, another important aspect is what kind of geometric representation methods should be adopted to represent and enable interactions with the objects during a packing process. In the following section, several widely used geometric representation methods are reviewed.

Geometric Representations

For the three-dimensional packing problem, the geometric representation is a very important component of the whole algorithm. To explore the design space thoroughly, it is often desirable to allow objects to move around in the container and penalize the degree of overlap to drive the design to a feasible solution. Collision detection is
performed at each iteration between every pair of components and between every component and the enclosure, and usually a large number of iterations are necessary for the optimization algorithm to converge to a good solution. Therefore, an appropriate geometric representation method and its corresponding collision detection technique are critical for the success of the optimization process. The collision detection technique should be able to efficiently detect whether two objects intersect, how much the overlap is, and even where the intersection exists.

The more accurate the geometry model is, the more computational time the collision detection costs. For packing problems, only the external shape of the components is required and rough accuracy is desired. If the geometric representation technique is capable of accuracy control, it is desirable to begin with a coarse CAD model, and at the final stage of the optimization process, replace the coarse CAD model by a more accurate model to get a final accurate result. This approach allows trading off between accuracy of evaluation and time taken to calculate the overlap, but may also result in unacceptable designs.

Geometric representation methods directly affect the respective possible shape morphing methods. For those general packing problems where the components are of fixed shape, there is no requirement on the geometric shape morphing ability of the solid modeling technique. However, for our problem, optimizing locations of the components as well as the shapes of selected ones, shape morphing ability will be a very important aspect for choosing the appropriate modeling scheme.

The geometric representations widely used in packing optimization consist of
constructive solid geometry (CSG), boundary representation (B-rep), tessellated representation and octree. In the following sections, a brief review is given in terms of collision detection and accuracy control ability of each representation method. The shape morphing ability is addressed in the next section on morphing methods.

Constructive Solid Geometry (CSG)

Constructive Solid Geometry (CSG) represents a solid as “a set-theoretic Boolean expression of simple primitive solid objects” [Hoffmann and Rossignac, 1996]. Both the surface and the interior of the final solid are thereby implicitly defined. The CSG representation is valid if the primitives are valid. The traditional CSG primitives are block, sphere, cylinder, cone and torus. A general set of primitives is the set of algebraic half-spaces. Primitives may be transformed through rigid body motions (translation and rotation) or scaling. The Boolean operations consist of regularized union, intersection, and difference. The final geometry depends on the order of the operations performed and on the location of the objectives when the operations are performed. The order of operations is stored in a binary tree structure called a CSG tree.

Collision detection for two simple primitives is fast and easy. However, the Boolean operations of simple primitives can generate quite complex solids, and the CSG tree of a real mechanical part can include a very large number of primitives and operations. Therefore the efficiency of the collision detection deteriorates greatly with the increasing number of primitives and operations involved in a CSG tree. The accuracy control for CSG could be achieved by substituting smaller primitives with larger primitives. However no known CSG commercial software supports this function.
**Boundary Representation (B-rep)**

In Boundary Representation (B-rep), the solid surface is represented as a quilt of faces, edges, and vertices [Hoffmann and Rossignac, 1996]. The topological entities, vertex, edge, and face, are related to each other by incidence and adjacency. Geometrically, the entities in a B-rep must not intersect anywhere except in edges and vertices that are explicitly represented in the topology data structure. Boolean union, intersection and difference operations are usually implemented for B-rep systems. Both regularized and non-regularizing Boolean operations may occur [Hoffmann and Rossignac, 1996].

Collision detection could be realized by Boolean operation, which is supported by most commercial software. However it is slow and not efficient since calculating the intersected surface of two parts is quite complex. Accuracy control of the B-rep model is not available in commercial software.

**Tessellated Representation**

Tessellation is a triangulated representation of the surface geometry. The surface is tessellated logically into a series of small triangles (facets). Each facet is described by a normal vector and three points representing the vertices of each triangle.

Collision detection is realized by checking all of the intersected facets. Although collision detection for triangles is easy, for a part composed of thousands of triangles, the efficiency greatly decreases. Furthermore, no known software calculates the overlap volume between two tessellated parts, which is required in packing optimization to
evaluate how “good” a configuration is. Accuracy could be controlled by precision level when a tessellated model is generated from the original CAD model.

Octree

The octree belongs to the technique of spatial subdivision that decomposes a solid into cells, each with a simple geometric structure. An octree divides a cube into eight sub-cubes. Each sub-cube may be further subdivided recursively. Cubes and their subdivision are labeled white, black or grey. A grey cube is one that has been subdivided and contains both white and black sub-cubes. A sub-cube is black if it is totally located in the inside of the solid to be represented, white if it is on the outside.

Collision detection for octrees is very efficient. The overlap of two octree objects is calculated by counting common cubes occupied by both of them. The accuracy control is also very straightforward by precision levels, i.e. the size of the cells. The octree could be an ideal representation method for packing with fixed shape components. However, for packing with shape morphing, it is not considered since its morphing ability is very limited, as discussed in the following morphing methods section. Furthermore, the outer surface or an object represented using an octree will display a voxelization aspect directly related to the size of the smallest cube in the octree representation.

Since geometric representation methods directly affect the respective possible shape morphing methods, what kind of representations shall be chosen is further addressed in the following shape morphing methods section.
Layout Design with Shape Morphing

Changing the shapes of the objects during the packing process is a relatively new concept and only a few applications have incorporated this idea. One case is in VLSI layout design in which the sizes of the rectangular, L-shaped, and T-shaped micro–cells can change during the floor plan stage [Wu and Dai, 2000] [Kang and Dai, 1997] [Chu and Young, 2004]. This approach is called soft block or flexible block by the researchers. Another application is in layout design of the cross-section of an automatic transmission for a motor vehicle [Su et al., 2004]. The components include a system of clutches and planetary gear sets, where the voxel representation is used. The shapes of clutches are changeable. The sizes of the clutches are adjusted after each move of the components to achieve a compact package. Faulkenberg [2005] applies a bi-level programming approach to a 2D rectangular packing problem. At the system level, the rectangular blocks are moved to maximize the compactness, and at the component level, the rectangular blocks are resized, morphed parametrically, to minimize the overlap between them. All the above cases are 2D applications, and the shapes of the objects are simple geometry.

Morphing Methods

In general, morphing is a technique for transforming an object from its current shape to a target shape. Morphing is performed in many fields such as computer animation, structural design, aircraft/automobile external shape design, mechanical component design etc. For shape morphing in the packing process, the task is to find a shape that is optimal for some specified objectives while keeping its volume constant or increasing it. Based on different solid modeling representations, the generally used
morphing methods include parameterization based morphing, scaling coupled with Boolean difference morphing method, octree-based morphing and mesh-based morphing.

Parameterization Based Morphing

Parameterization based morphing is based on boundary representation, and it is realized by altering the parameters that define the object geometry. Usually a form of the solution shape, such as a set of splines, is derived or assumed, and the parameters associated with this form could be changed. Parametric design is supported by most of the commercial CAD systems. The following example [Kim, 2002] shows a parametric beam design example based on commercial CAD systems. In Fig.2.1(a), a set of size parameters are defined. Fig.2.1(b) is the initial shape, and Fig.2.1(c) shows the morphed shape by changing the predefined parameters.

Shape morphing is very convenient by changing those variables defining the object geometry. However, parametric approaches make strong assumptions on the form of the solutions, which may not necessarily be related to the optimal solutions. So this method is suitable for problems in which we already have an idea about what the final result should approximately look like [Kegl, 2005]. If the shape of the object is composed of spline surfaces, the volume cannot be written analytically. In such a case, keeping the volume constant has to be realized through an optimization procedure by adjusting the position of the control points. Parameterization based morphing is widely used in aircraft aerodynamic shape optimization [Fudge et al., 2005] [Samareh, 2001], mechanical component design and structural design [Alonso et al., 2003] [Zhang et al., 2005].
Figure 2.1 Parametric beam design based on commercial CAD system [Kim, 2002]
Scaling Coupled with Boolean Difference Morphing Method

The scaling coupled with Boolean difference morphing method relies on the assumption that it is possible to scale an object to achieve the desired volume. The basic idea is to scale the object by an arbitrary factor which produces an object large enough so that Boolean difference with overlapping objects results in a remaining volume which is still larger than the desired volume. The process is as follows:

(a) Assume that the initial size of the object corresponds to a scaling factor of 1.

(b) Scale the object by a factor of $f$ resulting in a volume significantly larger than the desired volume. An arbitrary large $f$ (assume 10) can be chosen such that the desired volume is achieved after removing the intersections with other objects. The desired scaling factor $f^*$ can be achieved by using the bisection method [Tiwari, 2007].

The main drawback of this method is that crazy shapes maybe obtained as shown in Fig.2.2, which may not be desirable from the point of manufacture. In the figure, the green circle at the bottom is increased in surface or volume by increasing its radius (Fig.2.2 (a)). Then the Boolean subtraction of the blue and red circles at the top of the first image in Fig.2.2 (a) results in the green shape on the right (Fig.2.2 (b)). Another drawback is Boolean operation is extremely computational expensive for objects that have complex shapes.
Octree based Morphing

In octree representation, octrees are generated from the base geometry of the morphable components. Each octree has a local coordinate system attached to it and a 3-D vector is associated with it. Morphing is realized by scaling the octree model along a local axis. Su et al., [2004] studied a 2-D transmission layout problem with shapeable components based on an octree representation. The shapes of the transmission components are morphed by scaling the cells. The following example shows morphing object A by scaling the cells along y-axis in 2D.

However, the shape morphing ability is very limited through only scaling operation. Furthermore it is hard to control the scaling operations since one object can be composed of a huge number of cells. There is also no direct control on the volume of an octree object.
Mesh based Morphing

Mesh-based morphing is supported by both the surface tessellated and volume mesh representations. According to the techniques used to morph the meshed objects, the mesh-based morphing methods can be divided into two groups: geometric methods and physical methods.

**Geometric methods**

Geometric methods employ the purely geometric technique. The following ways morph a tessellated surface: directly moving vertices of the mesh, warping the triangles [Lazarus and Verroust, 1998], or using mesh transformation operators such as swap, collapse or split [Welch and Witkin, 1994] (Fig.2.4). For the volumetric mesh, Yifan Chen and Tonshal [2005] use an extended direct surface manipulation [Chen et al., 2000] method to morph the CAE mesh of an automobile body and its internal structure. They introduce a depth function to handle volume morphing.
In mesh based morphing with geometric techniques, the design variables are coordinates of vertices or mesh nodes, which result in a great freedom to change the shape of an object when compared with parameterization based morphing. However, the morphing process is hard to automatically control since there are a large number of design variables (coordinates). The morphing process is usually performed interactively, and the designer must know how to transform the shape step by step. Another problem of
mesh-based morphing is that wiggly shapes may be obtained as optimal designs [Daoud et al., 2005]. A smoothing procedure is usually required to keep desired surface quality. Volume control ability is not available in mesh-based morphing with geometric techniques.

*Physical Methods*

![Figure 2.5 2D mesh of a circle and 3D mesh of a ball](image)

Instead of using purely geometric techniques, the computer graphics community has explored the physically based methods for deformation modeling since the 1980’s. Physically based methods are also based on a tessellated representation or FE mesh. The mass-spring system is one of the physical models that have been widely used for modeling deformable objects. In the mass-spring system, an object is modeled as a collection of mass points (located on the mesh nodes) connected with springs (located on the edges of the mesh) (Fig.2.5). The spring forces can be linear or non-linear according to what kind of behavior is to be simulated. The motion of the mass points observes Newton’s Second Law [Moore and Molloy, 2007]. There are a lot of applications that have successfully simulated the deformation of really objects using a mass-spring model,
such as cloth flag motion [Terzopoulos et al., 1987], facial simulation [Terzopoulos and Waters, 1990], and prediction of the tissue deformation in surgery [Koch et al., 1996].

Mass-spring based morphing methods are generally easy to implement and computationally efficient. For mass-spring model based morphing, there are no assumptions on the final shape, and there is no need to explicitly specify the deformations such as in the mesh-based morphing with geometric technique. One drawback of this method is that the system must be integrated over small time-steps to ensure stability, resulting in slow simulations. Another problem is that the mass-spring system tends to oscillate due to its iterative basis [Moore and Molloy, 2007]. Volume control can be achieved during the iteration by keeping specific physical properties constant such as a given pressure [Matyka and Ollila, 2003].

Mesh Based Morphing Software

As reviewed in the previous section, for mesh based morphing with a purely geometric technique, the generally used approaches include moving vertices (nodes), warping triangles, or using transformation operators (swap, collapse and split). Most of these techniques require interactive user action. There are several CAE preprocessing packages such as HyperMorph [Altair Corp.], ANSA [Beta CAE Systems Inc.], Meshworks/Morpher [DEP Inc.] which provide morphing functions capable of morphing the shape of the object in a parametric way. In the following part, the morphing process in ANSA is explained.
ANSA is a pre-processing tool for Finite Element Analysis. "Morphing Tool" is the morphing function ANSA provides. The Morphing Tool modifies the shape of a mesh model by creating and handling special entities called "Morphing Boxes" (see Fig. 2.6). The Morphing Box is the basic entity of the Morphing Tool. The Morphing Box can include line, shell or solid elements, or any combination of them. By changing the shape of this box, the included elements will change their shape and position accordingly. The shape of the morphing box is controlled by Control Points (red points shown in Fig. 2.6) residing at the corners and on the edges of Morphing Boxes. Thus, the parameters can be defined as translation distances of the Control Points along three coordinate directions. After defining parameters associated with actions of the Control Points, the objects can be morphed "parametrically".

![Morphing box](image1)

![Shape after morphing](image2)

Figure 2.6 Shape morphing in ANSA [Beta CAE Systems Inc.]

There are some limitations for parametric mesh morphing with these commercial morphing software. For example, for morphing with ANSA, users have to pre-process models (define control points, morphing boxes, morphing actions) in the visual mode.
interactively. The way of defining Morphing Box and Control Points directly affects the morphing results. That is to say, the shape morphing freedom is limited and determined by how to define control points and morphing actions. This morphing technique is only suited for infinitesimal changes in shape parameters. Dramatic change is not allowed because when any mesh element that is initially within a morphing box reside outside this morphing box after shape changing, the associated control points are frozen by system [Beta CAE Systems Inc.]. In the case of major changes, the mesh quality typically deteriorates significantly [Zimmer and Prabhu, 2005].

The comparison of the different morphing methods is summarized in the following table in terms of shape morphing, volume control, strengths, and drawbacks.
<table>
<thead>
<tr>
<th>Morphing Methods</th>
<th>Shape morphing</th>
<th>Volume Control</th>
<th>Strength</th>
<th>drawbacks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameterization based</td>
<td>Change parameters defining the geometry</td>
<td>Through analytical constraint solver for simple shapes or through optimization for complex shapes</td>
<td>Easy to control, supported by most commercial CAD systems</td>
<td>Strong assumptions on the form of the solution</td>
</tr>
<tr>
<td>Scaling coupled with Boolean difference</td>
<td>Scale object with appropriate factor $f$ and Boolean difference</td>
<td>The desired factor $f$ can be obtained through bi-section optimization</td>
<td>No assumptions on the form of the solution</td>
<td>Non-manufacturable shapes maybe obtained, computationally expensive for Boolean operation</td>
</tr>
<tr>
<td>Octree based</td>
<td>Scale the octree model along local coordinate axes</td>
<td>Through optimization</td>
<td>No assumptions on the form of the solution</td>
<td>Very limited shape morphing ability by scaling</td>
</tr>
<tr>
<td>Mesh based with geometric methods</td>
<td>Move vertices (nodes), warp triangles, transformation operators (swap, collapse, split)</td>
<td>Through optimization</td>
<td>Great shape morphing freedom</td>
<td>Large number of design variables, intensive user interactions, hard to control, wiggly shapes</td>
</tr>
<tr>
<td>Mesh based with physical methods</td>
<td>Solve motions of the mass points under spring forces, gravity or other physical forces</td>
<td>Through keeping physical properties such as pressure</td>
<td>No assumptions on the form of the solution, great shape morphing freedom</td>
<td>Small time steps to ensure stability</td>
</tr>
<tr>
<td>Parametric mesh with commercial morphing software</td>
<td>Change parameters associated with actions of the Control Points and Morphing Box</td>
<td>Through optimization</td>
<td>Automatic, with little user interactions</td>
<td>Morphing freedom limited and determined by how to define control points and morphing actions, only suitable for infinitesimal changes in shapes</td>
</tr>
</tbody>
</table>

Table 2.1 Comparison of Different Morphing Methods
Summary

In this chapter, several important aspects involved in layout design with shape morphing have been reviewed.

A Genetic Algorithm may be the most appropriate optimization approach for general multi-objective layout design problem. The biggest issue for a gradient algorithm is that the solution quality is highly dependent on the initial starting point. For highly multi-modal problems such as packing optimization, a gradient-based algorithm is bound to frequently end up trapped in an inferior local optimum. Simulated annealing can avoid the trapping of solutions in a local optimum. However, it is usually used for single objective optimization, while the vehicle layout design is typically multi-objective. Heuristic rules are easy to implement and efficient for specific problems. However, they usually have special requirements on the geometry of the objects. Moreover such methods incorporate with difficulty other optimization objectives than compactness. The GA has shown good performance in solving the 3D packing problem. Especially, the Packing GA [Miao and Fadel, 2005] with its new encoding method and with GA operators tailored for the packing problem. The latter, improves average compactness and percentage of acceptable layouts significantly. Because of all these issue, the Packing GA is chosen in this work, to solve the layout design with shape morphing problem.

CAD representation methods directly affect the collision detection efficiency and the respective possible shape morphing methods. Collision detection for octree is very efficient. Its accuracy control is also very straightforward using precision levels. Octree
may be an ideal representation method for packing with fixed shape components. However, for layout design with shape morphing, the octree method is not considered since its morphing ability is very limited and since its surface accuracy issues may be significant. In this research, the tessellated representation is chosen, with the overall consideration for collision detection, accuracy control and shape morphing.

Several morphing methods are investigated and compared. The parameterization based morphing method is straightforward and easy to control, but has strong assumptions on the form of the solution. It is suitable for morphing objects when their final approximate shape is known. The mesh based method has much more flexibility in the morphing process. When this method relies on a purely geometric technique to accomplish the morphing, the latter is performed interactively, and the huge number of control points is typically hard to control. Relying on a physical model, there is no need to explicitly specify the deformations of the nodal points such as in the mesh-based morphing based on the geometric technique. However, the physical system evolving and morphing must be integrated over small time-steps to ensure stability, which may result in very slow simulations. For the parametric mesh morphing with commercial software, the morphing freedom is confined by how to define the control points and morphing actions. Meanwhile, only infinitesimal changes are allowed. Obviously, no morphing technique outperforms all others on all aspects. The appropriate morphing method is chosen according to the characteristics of shape morphing in layout design in chapter 4.
CHAPTER 3  A PHYSICAL MODEL BASED PACKING ALGORITHM FOR COMPACT PACKING -- RUBBER BAND ANALOGY

In this chapter, a physical model based packing algorithm for compact packing -- the rubber band analogy method is presented. Fadel and Sinha [2001] first introduced this method in 2001. This method solves packing problems by simulating the physical movements of a set of objects wrapped by a rubber band. The rubber band analogy corresponds to stretching a rubber band (in two-dimensional problems) or a rubber balloon (in three-dimensional problems) around the set of objects (for the remainder of the dissertation, we will refer to the rubber band even in the three-dimensional case). The rubber band has a tendency to restore to its un-stretched status (lower elastic energy status), therefore the objects are brought together by the elastic forces applied by the rubber band resulting in a compact configuration (locally lowest elastic energy status). In other words, this method solves packing problems by simulating the physical movements of the objects subjected to elastic forces until maximum compactness is reached. Starting from different initial configurations will lead to different optimal packing results. To improve the compactness further, two new operators are introduced: volume relaxation and temporary retraction. The two operators allow temporary volume (elastic energy) increase such that possible better packing results can be obtained. The rubber band analogy packing algorithm is implemented and applied for three-dimensional arbitrary shaped objects. This elastic analogy technique can guarantee locally optimal solutions, and displays a very intuitive behavior that might lead to further advances in optimization for other problems.
after a judicious mapping. This rubber-band model is further extended for morphing purpose in chapter 5.

**Methodology**

This method solves packing problems by simulating the physical movements of the objects under the elastic forces applied by a rubber band until maximum compactness is reached. The following five assumptions are used.

1. Assume no friction forces since only the positions of the objects are important and no velocity or acceleration information is used.
2. The rubber band can be stretched indefinitely with constant stiffness.
3. The objects are distributed in space with no overlap, defining the initial configuration.
4. Objects will not switch positions or pass through each other during the packing process.
5. No bouncing back is allowed when contact occurs.

The packing process with the rubber band analogy can be described as follows: a rubber band is stretched around the objects, and the elastic energy is considered high initially. The stretched rubber band has the tendency to restore to its un-stretched status (lower energy status). Therefore the objects are brought together by the elastic forces exerted on them. Initially, the objects translate along the direction determined by the elastic forces. If contact occurs, the objects remain in contact without bounce. Then the objects can either rotate about the contact vertex/edge where the forces become moments, slide along the contact surface or cling together and translate further. The type of movement is
determined by the contact status. The translation and rotation processes iterate until no motion is possible to improve the compactness (i.e. minimize the elastic energy).

CAD Model and Intersection Detection

To implement the packing process, one important issue is how to represent the geometry of the objects. The tessellated representation is used. The data file of the CAD model is in “STL” format. As described in chapter 2, the surface is tessellated logically into a series of small triangles (facets). Each facet is described by a normal vector and three points representing the vertices of each triangle. The STL file is a standard format that all commercial CAD software can import and export to, which facilitates generating and displaying the objects’ CAD models.

Collision detection is another very important aspect in packing optimization. The efficiency and functionality of the collision detection technique highly affects the optimization process. In this research, collision detection is performed by invoking library functions in the collision detection package Swift++ (Speedy Walking via Improved Feature Testing for Non-Convex Objects) [UNC GAMMA Group]. Swift++ is a software package capable of intersection detection (detecting whether two objects intersect), tolerance verification (detecting whether two objects are closer than a given tolerance), approximate and exact distance (minimum distance between a pair of objects) computation, and contact normal calculation for arbitrary shaped objects. These functions provide necessary collision and contact information for implementing our algorithm, which will be described in following sections. This package requires the models with closed surface or boundary. The Oriented Bounding Boxes (OBB) are placed around each
object to prune unnecessary computation [UNC GAMMA Group]. It is claimed to be more robust and efficient than currently available packages such as RAPID (Robust and Accurate Polygon Interface Detection) and PQP (Proximity Query Package) [UNC GAMMA Group].

Problem Formulation

Considering the functions that the collision detection package can provide, the packing problem can be formulated as follows:

\[
\begin{align*}
\text{max} & \quad \text{Compactness} \\
\text{s. t.} & \quad \text{No overlap constraints} \\
& \quad \text{Tolerance constraints}
\end{align*}
\]

Compactness is measured as the ratio between the sum of the volumes of all objects and the volume of the convex hull enclosing all objects. The no overlap constraints impose no intersection between any two objects. The tolerance constraints define a maximum distance between two objects that can be considered “in contact”.
Algorithm

The flow chart of the algorithm is shown in Fig. 3.1. In this section, important issues pertaining the algorithm are discussed.

Figure 3.1 Flow chart of the rubber band packing algorithm
Analogy between the Convex Hull and a Rubber Band

The first task of the algorithm is to construct the convex hull as an analogy to the rubber band enclosing all objects, which is the basis of this methodology. The stretched rubber band has the tendency to restore to its original un-deformed status due to its elastic nature. Therefore the movements (translation/rotation direction and magnitude) of the objects in contact with the rubber band are determined by the elastic forces exerted on them and the positions of the contact points. To determine the contact points and compute the translation vector, the analogy between the convex hull and the rubber band is made. The convex hull is the smallest convex polygon or polyhedron that contains all of the points describing the vertices of all objects (see Fig. 3.2 (a) and (b)). Qhull package is used to construct the convex hull, which is based on the Quickhull algorithm [Barber et al., 1996]. Note that the objects that are not in contact with the rubber band are temporarily kept static if they also do not touch any other objects.

Figure 3.2

Figure 3.2 (a) Convex polygon in 2D  (b) Convex polyhedron in 3D
Translation

After generating the convex hull, the translation vectors can be calculated. The elastic forces of the rubber band are exerted on the vertices of the objects that are in contact with the convex hull (see Fig. 3.3). These contact vertices are also the extreme points of the convex hull. The vectors, whose tails are on a selected extreme point and heads are on the adjacent extreme points of the convex hull are computed. Note that in the 2D case one extreme point has two adjacent extreme points (Fig. 3.2(a)) and in the 3D case one extreme point can have multiple adjacent extreme points on the convex hull (Fig. 3.2(b)). The resultant vectors for each extreme point are calculated, and translated to the center of gravity of the respective objects. Then the resultant vector acting on each object is calculated by taking a vector sum of all those vectors acting on that object (Fig. 3.3). Finally, this vector is normalized to a unit vector, which indicates the translation direction.

![Figure 3.3 Calculating translation vector for one object](image)

Next, the problem is to determine the translation magnitude. Since it is assumed that objects neither exchange positions nor pass through each other, the initial translation magnitude should be no more than the diameter of the bounding sphere of the smallest
object. Let the initial translation magnitude be equal to this diameter. The objects translate with this magnitude along the pre-calculated direction until collision or contact occurs.

Collision Detection and Tolerance Check

In packing optimization, there are two important constraints that must be satisfied. One is no intersection between objects, and the other is the maximum distance between adjacent objects should be smaller than the predefined tolerance. If the latter constraint is satisfied for a pair of objects, these two objects are considered in contact. When intersection occurs, the intersected objects retract by an amount such that the collision constraints are satisfied. Here a one dimensional search algorithm is used to find how much the objects should retract. The entire process can be described as follows. First, the current non-intersected status is recorded, and the objects are translated in accordance to the applied translation vector. If intersection occurs, the intersected objects are retracted to the previous non-intersected positions, and the translation vectors are scaled with golden section numbers [Press, 1992]. Then the objects are translated with the new scaled vectors, and the collision status is checked. The above process iterates until no intersection occurs.

After the intersected objects are separated, a tolerance check is done to see whether the adjacent objects are in contact; if not, the objects are translated again until contact occurs (see the flow chart Fig. 3.1).
Rotation, Sliding, Clinging

Simple translation does not necessarily lead to an optimal packing configuration. In most cases, to get a better optimum, further movements are needed after contact occurs. The type of movements is determined by the contact status. For three dimensional objects, there are six possible contact states: surface-surface, surface-edge, surface-vertex, edge-edge, edge-vertex and vertex-vertex. Edge-vertex and vertex-vertex contacts seldom occur in a normal packing process, they are viewed as subsets of edge to edge contacts when considering the inaccuracies that are inherent in the computation of the geometry and intersection and therefore are not considered. In this research, the first four contact states are considered.

Figure 3.4 Possible movements for surface-surface contact

When objects contact, there are three possible movements: one object rotates relative to another object, objects slide along the contact surface, or objects cling together and translate further as a solid block. Fig.3.4 shows the three possible movements for surface-surface contact. The priority for the three types of movements is defined in the following order from high to low: rotation, sliding, and clinging. By this order, first the objects are rotated as long as the volume of the convex hull decreases (i.e. compactness
increases) and as long as the collision constraints are satisfied. If there is collision or volume increase, there will be no rotation. Second, one object tries to slide relative to another object. If neither rotation nor sliding can improve the compactness, then the need for two or more objects to cling together and translate with same magnitude and direction is checked.

The following cases for rotation occur according to the contact status between each contact pair. For surface-vertex contact (Fig. 3.5(a)), the object first rotates about an axis passing through the contact vertex and perpendicular to the plane determined by the translation vector placed at the center of gravity of the rotating object and the vector joining the contact vertex to the center of gravity. The aim is to covert the problem to a surface-edge contact (Fig. 3.5(b)), and at the same time to reduce the volume of the convex hull and avoid collision. Next, because of surface-edge contact, the object rotates about the contact edge until a surface-surface contact status is achieved (Fig. 3.5(c)) while the volume of the convex hull continues to decrease and the collision constraints are satisfied. Finally, for surface-surface contact there may be no need to rotate further (Fig. 3.4(b) and (c)). In some cases, one object rotates about the contacting edge of the other object only if this movement can reduce the volume of the convex hull while satisfying the collision constraints (Fig. 3.4(b)). Note that both objects of the contact pair can rotate as long as the rotation movement can reduce the volume of the convex hull as well as satisfy the collision constraints. From the translation vector, the rotation direction can be determined. The rotation angle is calculated using the same one dimensional search algorithm as the one described in the translation section.
Edge-edge contact is described in Fig. 3.6(a). First, the cross point of the two edges is calculated. The two edges are considered in contact with the predefined tolerance, so they do not cross in the strict mathematical sense. Therefore, the projection point of one edge on the other edge is identified as the “cross point”. The next step is finding the axis of rotation. The axis of rotation passes through the cross point and is perpendicular to the plane determined by the translation vector placed at the center of gravity of the rotating object and the vector joining the cross point to the center of gravity. The object first rotates about this axis to achieve surface-edge contact (Fig. 3.6(b)), and at the same time the volume of the convex hull should be reduced while avoiding collision. Next, according to surface-edge contact, the object rotates about the contact edge until the surface-surface contact status is achieved (Fig. 3.6(c)) while the volume of the convex hull decreases and the collision constraints are satisfied. Finally, for surface-surface contact, two objects slide along the contact surface to achieve the possible minimum volume.
In the algorithm, for each type of movement, volume increase is used as one of the stopping criteria. It is consistent with the fact that the stretched rubber band has the tendency to restore to its un-stretched status (lower energy status), leading to a smaller convex hull volume. Movements leading to smaller volume are considered favorable and accepted. The rotation, sliding, and clinging then translation are performed until collision occurs or the volume of the convex hull ceases to decrease.

Improving Compactness

The above rubber band analogy algorithm packs objects by simulating the movements of the objects under elastic forces applied by a rubber band. Since the rubber band analogy is a local search method, the packing result depends on the starting configurations. Some initial configurations may lead to good packing results, while others may lead to inferior solutions. However, those good initial configurations are unknown, and because of the tremendous size of the packing design space, an exhaustive search cannot be used to find such initial configurations. Concerning the cases that lead to inferior
solutions, the greedy nature of the rubber band analogy method allows rotation and translation of objects only if the volume of the convex hull decreases. Therefore, this may reduce the probability to obtain desirable packing results.

Upon close inspection of the process, it appears that some potentially beneficial rotations may be discarded because of a slight increase of the convex hull volume before a potentially significant decrease. Also, some promising movements may not be allowed by the algorithm because there is no enough room for further rotation or translation due to contact with other components. Therefore, it is necessary to add some flexibility and intelligence to the original rubber band analogy method so that these inferior solutions with potential can lead to more desirable configurations.

In what follows, two new operators are introduced as an extension to the original rubber band analogy algorithm. The two operators are volume relaxation for rotation and temporary retraction of the “obstacle” objects. These two operators both allow temporary volume increase (energy increase). The strategy is to allow the volume (energy) increase first. Then after several steps, if these operations can improve the packing, they are considered favorable and accepted. The process is analogous to the optimization process called simulated annealing where some solutions that do not necessarily lead to a decreased energy state are accepted with some probability in the hope of enabling the algorithm to escape local optima.

**Volume Relaxation**

Volume relaxation allows objects to rotate, although the volume of the convex hull increases. Note that the volume should not increase too much, and after some testing, we
converged on a method that requires the temporary volume increase to be no more than the volume of the bounding sphere of the rotated object. Without the effect of the rubber band, one object could rotate freely about the contacting edge/vertex from one extreme position to another extreme position. The change of the convex hull volume is not monotonic and can have one or multiple extreme values somewhere between two extreme positions.

Figure 3.7 Object B rotates freely from position 1 to 5

Here one simple case (Fig. 3.7) is illustrated to show that the volume change is not monotonic. To make the calculation easier, we do the following simplification: the set of objects is divided into two groups, one includes the rotating object B, and the other includes all of the other objects that are enclosed by the convex hull A. The effects of the shape of object B on the volume change are not considered, therefore object B is simplified as a line segment in the plane perpendicular to the rotation axis, and its length is equal to the diameter of the bounding sphere of object B. Then, we can easily see the approximate variation trend of the convex hull volume as shown in Fig. 3.8, which is a function of the rotation positions. At position 3, the convex hull volume reaches to its maximum value,
and at positions 1 and 5, the convex hull volumes are smallest. If Object B is currently at position 2 and the rotation direction is clockwise as dictated by the rubber band, using the original rubber band analogy, no rotation would be allowed since the convex hull volume increases in this direction. However, after passing through position 3, the volume decreases. So if a temporary volume increase is allowed to make this rotation happen, it is possible that a better packing can be achieved at positions 4 or 5.

Figure 3.8 Volume changes with respect to rotation positions

Thus considering that the change of convex hull volume can be non-monotonic during the rotation process, volume relaxation is performed and even if the convex hull volume increases, objects are rotated as long as no collision occurs. If this rotation leads to a smaller convex hull volume (lower energy) after the rotation finishes, it will be accepted; otherwise, discarded. The benefit of this operation is demonstrated in the examples section.
**Temporary Retraction**

Temporary retraction corresponds to retracting an object and keeping it static temporarily. Experience shows that sometimes the objects are crowded together too quickly to allow further rotation or translation. In other words, some objects are obstacles for the movements of others. Therefore, to improve the compactness further, the temporary retraction operator is introduced. First, the objects that are considered "obstacles" must be identified. Since rotation is very important for compactness, an object is considered to be an "obstacle" if it prevents the rotation of two or more objects. Then these "obstacles" are retracted along the reverse direction of the translation vector and temporarily stay static (note that they are still in the convex hull) to make room for the other objects’ movements. When the possible rotations of the other objects starting from the current new configuration are performed, the "obstacles" are unfrozen join again in the packing process. This criterion may not always lead to further improvement in compactness and there is a risk to worsen previous packing results by retraction. So, if after several packing steps, there are no improvements or the configuration worsens, this retraction is discarded. The packing process with temporary retraction is demonstrated in the examples section.

**Examples**

In this section, several examples are shown to demonstrate the packing process with the rubber band analogy.
Packing 7 Cubes with Translation, Rotation and Sliding

In this example, the 7 cubes are packed with translation, rotation and sliding as shown in Fig. 3.9. Starting from the initial configuration shown in Fig. 3.9(1), objects A, B, C, E, F and G are pushed toward object D by the rubber band until they come in contact with object D (Fig. 3.9(2)). The following movements are determined by the contact status.

Figure 3.9 Packing 7 cubes with translation, rotation and sliding

Objects B, C, E and F are in surface-surface contact with object D. They keep static since neither rotation nor sliding can improve the compactness. Objects A and G are in surface-vertex contact with object D. Rotation is needed. First, object A rotates about object D until they are in surface-edge contact as shown in Fig. 3.9(3). Next, Object A rotates further about object D until they are in surface-surface contact. Object A is also in
surface-surface contact with objects B and C (Fig. 3.9(4)). Object G has similar movements as object A (Fig. 3.9(5)). Finally, object A slides along the contacting surface with objects B, C and D, and G slides along the contacting surface with objects D, E and F until maximum compactness is obtained (Fig. 3.9(6)). This example illustrates that the rubber band applies forces on the objects in contact with it and brings them together. In this case, the object that is not in contact with the rubber band stays static during the packing process since the resultant of all the reaction forces exerted by other objects is zero.

**Packing Three Polyhedrons without and with Volume Relaxation**

In this example, two packing results without and with volume relaxation are compared. The polyhedrons have different sizes in the Z direction. Figure 3.10 shows the case where the objects are packed without volume relaxation. Starting from the initial configuration shown in Fig. 3.10(1), where objects A, D, and objects A and B are in contact, two rotations occur. Object D rotates clockwise about object A, and object B rotates counterclockwise about object A. This rotation leads to a smaller convex hull volume as shown in Fig. 3.10(2). However, at this point, the rotation stops because the convex hull volume begins to increase. It is obvious that if the rotation is continued by volume relaxation, a more compact packing can be obtained immediately as shown in Fig. 3.11. With volume relaxation, the objects B and D will continue to rotate about object A until surface contact happens (Fig. 3.11(2), Fig. 3.11(3)). Then objects B and D slide along the contacting surface with object A until objects B and D are in surface-surface contact as shown in Fig. 3.11(4). Now the packing result is much more compact than the configuration shown in Fig. 3.10(2).
Figure 3.10 Packing three polyhedrons without volume relaxation

Figure 3.11 Packing three polyhedrons with volume relaxation
Figure 3.12 Packing four polyhedrons without temporary retraction

Packing Four Polyhedrons without and with Temporary Retraction

In this example, two packing solutions with and without temporary retraction are compared. The polyhedrons have different sizes in the Z direction. Fig. 3.12 shows the case where the four polyhedrons are packed without temporary retraction. Starting from the initial configuration shown in Fig. 3.12(1), the objects translate first until objects C and D are in contact as shown in Fig. 3.12(2). Then object C rotates clockwise about object D until surface-surface contact occurs, and object D rotates counterclockwise about object C until D contacts with object A. For the new contact edge, object D cannot rotate clockwise about object A since object C is the “obstruction”, but object A can rotate counterclockwise
about object D until A contacts with B as shown in Fig. 3.12(3). For this new contact pair, object B rotates counterclockwise about object A until B contacts with C as shown in Fig. 3.12(4). We can see that object C rotate clockwise further with a new axis different from before until it contacts with B. At this point, packing stops because all of the objects are crowded together and no motion is possible to improve the compactness.

However, the packing result is not very good. The objects have crowed together too quickly and no significant rotation or other movements such as sliding can occur. Therefore, temporary retraction may be needed to improve the compactness further. For both objects B and D, object C is the obstacle for their further rotation, so retracting object C may lead to a better packing result. Starting from the last crowded configuration, we retract object C to make enough room for other objects’ movements (as shown in Fig. 3.13(2)). Now, there is enough room for objects B and D to rotate. They rotate until they are in surface-surface contact with object A. Now object C in introduced in the packing. Objects D and B slide along the contacting surface with A, and object C translates until it contacts with object B again as shown in Fig. 3.13(3). Now object C rotates further until it is in surface-surface contact with object B and then slides. At this point, the compactness has already been improved greatly, so this temporary retraction operation is accepted. Next, objects slide along each contact surface to improve the compactness further as shown in Fig. 3.13(4), (5) and (6). Note that the objects also slide along the Z direction since they are of different sizes in the Z direction.
Comparison with Gradient-based Methods

Gradient-based algorithms use gradient information to guide the search for optimal solutions. For instance, sequential quadratic programming could be used to determine the optimum location and orientation of the set of objects for maximum compactness. Compared with gradient-based methods, the enhanced rubber band analogy method has two advantages.

For most packing problems, the explicit gradients of the objective functions are not available, and cannot be obtained by computing the derivatives of the objective functions directly. In this case, the gradients can be calculated numerically by the finite difference method. However, finite difference approximation is not efficient and may mislead the search [Cagan et al., 2002]. For the rubber band analogy method, there is no need to
evaluate gradients. Instead, the rubber band analogy method calculates the translation vector and rotation direction directly from physical principles. This is very explicit and more accurate. Therefore, compared with a gradient method, the rubber band analogy method is more efficient for packing problems.

It is expected that gradient-based methods would lead to a local optimum and would heavily depend on the initial configuration. For a highly multi-modal design space, the gradient-based algorithm would frequently be trapped in inferior local optima. Although the rubber band analogy is also a local search algorithm, it is more flexible and intelligent with the extension of the two new operators to the original rubber band analogy algorithm. Therefore it has a better chance to avoid some inferior local optima.

Extension of the Rubber-band Analogy Physical Model

This rubber band analogy method solves packing problems by simulating the physical movements of a set of objects subjected to elastic forces applied by a rubber band (spring) until the maximum compactness is reached.

Instead of exploit the elastic forces of the springs to bring objects together for compact packing, this method could be extended for shape morphing of deformable objects whose surface is an elastic rubber band or balloon. The balloon surface (surface of the deformable object) is modeled as a collection of mass points (located on the nodes of the surface mesh) connected with springs (located on the edges of the surface mesh), which is named as mass-spring model. It is proposed to inflate the rubber band or balloon to occupy the desired volume. The mass points forming the surface nodes may be subject to spring force and pressure force. By simulating movements of the mass points, which are nodes of the
surface mesh of an object, the shape of the object is therefore directly morphed. The detail of mass-spring model will be addressed in chapter 5.

Summary

This work builds on initial developments of the rubber band analogy packing method. One improvement is the implementation of the rubber band analogy method for three-dimensional arbitrary shape objects. One important aspect concerns how the objects’ movements are handled when contact occurs and four contact cases are investigated. Another improvement is the enhancement of the original rubber band analogy method by introducing two new operators, volume relaxation and temporary retraction. These operators reduce the greedy nature of the method by allowing temporary volume increase of the convex hull to obtain more desirable packing solutions. The rubber band analogy method is intuitive and very efficient in getting local optimal solutions. The dependency of the final result on the initial configuration could be alleviated by coupling the rubber band analogy method to some global search algorithm such as Genetic Algorithms, Simulated Annealing, or others, and would therefore be used as a local optimal operator to accelerate convergence. This physical based rubber-band analogy model could be extended further for shape morphing purpose, which will be discussed in chapter 5.
CHAPTER 4 CHARACTERISTICS OF MORPHING IN LAYOUT DESIGN

This chapter analyzes the characteristics of shape morphing during layout design, comparing it with other type of morphing such as computer graphic morphing. The special functional requirements for morphing methods used in layout design are discussed. Furthermore, the components involved in layout design are classified according to their shape changing freedom since different types of components may require different appropriate morphing methods.

Types of Components

The components involved in layout design can be divided into three categories according to their desired shape changing freedom:

Fixed shape components

The shapes of this type of components (such as engine, transmission, axles) do not change during the packing process. They are dictated by their functional characteristics and do not need to be increased or modified unless their design is reconsidered. Since a redesign of such a component is a major undertaking, it is performed off-line and not in tandem with the layout design process.

Components morphable only through some size parameters

The shapes of these types of components (such as radiator, filter) have only limited morphing freedom and cannot be changed arbitrarily. Their shape can be morphed through some predefined size parameters. The objects do not have to be redesigned off-line, the size has to be changed parametrically to better
accomplish some function.

Components morphable to any shape

The shape of this type of components (such as reservoirs and fuel tank shown in Fig. 4.1) can be morphed to any shape as long as the shape is easy to manufacture and the object’s functional requirements are satisfied. This is the most interesting type for our particular application.

Fuel tank

Coolant reservoirs

Figure 4.1 Components morphable to any shape
Having reviewed possible representation methods and morphing methods in Chapter 2, the objective of this chapter is to better assess the most appropriate method to accomplish the task at hand. For components morphable only through some size parameters, the parameterization based method should be sufficient and convenient. For example, the size of a radiator can be altered by modifying its height, width or depth. The shape will not change, the component does not have to adapt to available space except in very special circumstances. For components morphable to any shape, such as reservoirs and fuel tanks, the morphing task is usually to grow them into the available space in a layout. Since there is no assumption on the final shape of the components, the mesh-based morphing method is the most appropriate.

Characteristics of Shape Morphing in Layout Design

In applications such as computer animation, the target shapes or the target positions of some nodes of the objects are known, and the task of the designer is to be able to morph the object from its current shape to a target shape with interpolation frames (Fig.4.2) [Alexa, 2002]. Intensive interactive actions are required during the morphing process. For morphing in layout design, especially for components that can have arbitrary forms, the target shape is usually unknown. The process of morphing the objects to fit available space has to be realized through collision detection where the boundary constraints are checked. However, the available space is usually unknown. Therefore, it is hard to make an assumption on the final shape of the object, which probably has nothing to do with the optimal shape.
Figure 4.2 Interpolation frames from initial shape to target shape by [Alexa, 2002]

Since the layout process is performed automatically with optimization algorithms, it is desirable that the shape morphing be done automatically through programming or scripting. The morphing techniques with intensive interactive actions are not suitable for shape morphing in layout design.

Packing problems deal with how to arrange components in an available limited space. Therefore, the volume of the object is a very important design criterion in packing problems. When the object is morphed, there should be a way to maintain control of the volume of that object (either constant or a target value) no matter how the shape is changed.

Requirements on Morphing Methods for Layout Design

According to the above analysis, several special requirements for the morphing methods used in layout design can be derived.

1. There should be no assumption on the final shape of the morphable objects that can have arbitrary shapes.

2. The morphing method should be able to morph the object from one shape to another shape with dramatic changes, not just infinitesimal adjustments.
3. The morphing process should be automatic or with little user interaction, and should be applied to a minimum number of control variables for easier control.

4. The volume of the objects must be controllable during the morphing process.

Table 4.1 Qualification of Four Specific Morphing Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Requirement 1</th>
<th>Requirement 2</th>
<th>Requirement 3</th>
<th>Requirement 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameterization based with commercial CAD software</td>
<td>×</td>
<td>×</td>
<td>√</td>
<td>×</td>
</tr>
<tr>
<td>Mesh based with geometric techniques such as moving vertices, transformation operators</td>
<td>×</td>
<td>√</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Parametric mesh with commercial morphing software</td>
<td>×</td>
<td>×</td>
<td>√</td>
<td>×</td>
</tr>
<tr>
<td>Mesh based with mass-spring physical model</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
</tbody>
</table>

According to the above requirements, Table 4.1 shows the qualification of four specific morphing methods for shape morphing in layout design. The mass-spring physical model is the most appropriate one for layout design application from the aspects of shape morphing freedom, automation and volume control. Therefore, it is chosen and tested for the purpose of layout design. In the following chapter, the mass-spring model is implemented.
CHAPTER 5  A PHYSICAL MODEL BASED SHAPE MORPHING METHODS FOR LAYOUT DESIGN — MASS-SPRING MESH BASED MORPHING

This chapter focuses on the mass-spring physical model based shape morphing method for layout design. First, the concept of mass-spring model is introduced, an extension of the rubber-band model in chapter 3. In the previous work, the rubber band acted as springs, bringing objects together to form a compact packing. In this particular case, we propose to inflate the balloon or rubber band to occupy the required volume. In the case of the rubber band, the node points were the corner points of the objects to pack which also coincided with the convex hull nodes. In the balloon case, the nodes forming the surface are mass points connected by springs, which may be subject to gravity force, spring force and internal pressure force because of inflation. The implementation is based on two physical laws: the Ideal Gas Law for pressure calculation at each specific state and Newton’s second law for the motion of mass points. After integration of Newton’s second law, the positions of the mass points are obtained. If these mass points are considered to be the node points on the surface of an object, its shape is therefore directly morphed and obtained. A special collision detection procedure between deformable object and fixed shape object or enclosure is proposed, and the response of the deformable object after collision is studied. Finally, an example shows how an object is morphed to fill the available space with this morphing method.
Mass-spring Model

The mass-spring system is one physical model that has been widely used for modeling deformable objects [Terzopoulos and Waters, 1990] [Koch et al., 1996] [Matyka and Ollila, 2003] [Moore and Molloy, 2007]. In the mass-spring system, an object is modeled as a collection of mass points (located on the mesh nodes) and connected with springs (located on the edges of the mesh) as shown in Fig. 5.1. The connection only exists between neighboring Mass Points. The spring forces can be linear or non-linear according to the type of behavior to be simulated. A linear spring is used to model an elastic object, while non-linear springs are usually used to model an object such as human skin that exhibits inelastic behavior. Since the objects involved in packing usually have a constant volume or a specified volume, the linear spring is appropriate to simulate their behavior when morphed. The shape of the meshed object will change and

![Figure 5.1 2D mesh of a circle and 3D mesh of a ball](image)
the object will be reduced in size if the mass points are only subjected to spring force. This behavior was used in the rubber-band approach described earlier. To keep a constant volume, a pressure is added inside the object [Matyka and Ollila, 2003]. For our case, it is proposed to continuing increase the pressure until a specific target volume is achieved. The motion of the mass points obeys Newton’s Second Law. The details of this method are described in the following sections.

Force Applied on Mass Points

The forces applied on the Mass Points include gravity force $\vec{G}$, elastic force coming from springs $\vec{f}_s$, and an internal pressure force $\vec{F}_p$ (Fig.5.2). Calculations of these forces are shown as follows.

![Figure 5.2 Forces applied on mass points](image)

**Gravity Force**

The Mass points in a gravity field are subject to gravity force. However, this force is not relevant for packing purposes since the actual object is a rigid body and its
morphing shape should not be affected by the gravity force. So it is assumed that there is no gravity force acting on the components during packing in our morphing problem. Note that the vehicle dynamic simulation that is conducted as case study later does consider the location of the center of gravity to simulate the dynamics of the vehicle. However, for the purpose of morphing only, gravity can be ignored.

\[ \vec{G} = mg = 0 \]  

(5-1)

Where \( g \) is the gravity field, and it is assumed to be 0 in this application.

**Elastic Force**

![Figure 5.3 A spring with ends located at vertices \( V_i \) and \( V_{i+1} \)](image)

In this implementation, the linear spring force is used. It can be derived from Hooke’s Law:

\[ \vec{f}_s = -(k_s(|d| - s) + k_d \frac{d \cdot d}{|d|}) \frac{d}{|d|} \]  

[Pfenning, 2002] (5-2)

Where \( \vec{f}_s \) is the spring force, \( k_s \) is the constant elasticity factor, \( k_d \) is the damping term. Notation \( s \) is the rest length of the spring, \( |d| \) is the deformed length which can be calculated from equation 5-3. Notation \( d \) is the position vector of the two ends of the spring, which is derived from equation 5-4.

\[ d = V_{i+1} - V_i \]  

(5-3)
\[ \dot{d} = \dot{V}_{i+1} - \dot{V}_i \]  
\[ |d| = |V_{i+1} - V_i| \]  

**Pressure Force**

To maintain a constant volume or achieve a specific given volume, a pressure is added inside the object. The pressure force acts on the triangular surface elements, and is parallel to the normal vector of the triangular facets (Fig. 5.4).

\[ \vec{F}_p = \vec{P} \cdot \hat{n} \left[ \frac{N}{m^2} \right] \]  

Where \( \vec{P} \) is the pressure value and \( \hat{n} \) is the normal vector of the facet surface. To calculate the pressure force \( \vec{F}_p \) on a specified area \( A \), the following equation is used.

\[ \vec{F}_p = \vec{P} \cdot A \ [N] \]  

In the following section, the procedure to calculate the pressure value \( P \) is explained.

**The Pressure Value**

How to obtain the pressure value is critical for the success of the algorithm. Ideal
Gas Approximation is used in this implementation [Matyka and Ollila, 2003]. Since only the macroscopic level effects of gas presence are useful, it is assumed that there is no interaction between the gas particles inside the object. The Ideal Gas Law [Callen, 1985] gives the simple relationship between pressure value, temperature of gas and the volume of the object.

\[ PV = nRT \quad (5-8) \]

Where \( P \) is the pressure value, \( V \) is the volume of the object, \( n \) is the number of gas moles (abbreviated as mol), \( R \) is the ideal gas constant, and \( T \) is the temperature. From equation 5-8, if the volume of the object is known, the pressure value can be expressed as:

\[ P = V^{-1}nRT \quad (5-9) \]

For a given number of moles, assuming the temperature \( T \) is constant, if \( P \) is constant, \( V \) (Volume of the object) should be constant too. This is useful for simulating the deformation of the object when an external force is applied and the internal pressure is kept constant.

In our case, the purpose is to expand the volume of the object to fit in the available space. For a given volume of container, the internal pressure of the container increases proportionally to the number of moles. Assume the current state is as follows:

\[ P_1 = V^{-1}n_1RT \quad (5-10) \]

If the number of moles is increased to \( n_2 \), the internal pressure is increased to \( P_{1,2} \) with the current volume \( V \).
To keep an equilibrium state, the volume of the object expands. During expanding process, for each specific volume \( V' \), the respective pressure is given as follows:

\[
P' = \frac{P_2 V}{V'}
\]  

(5-12)

From equation 5-12, the volume of the object is required for calculating the new pressure value. The following section shows an efficient approximation of the volume of a meshed object.

**Volume of the Object**

The volume of the object can be calculated with the Gauss Theorem. With a closed surface, integration over volume can be replaced with an integration over the surface of the object. The original form of Gauss’s Theorem is

\[
\int_S C \cdot \vec{n} da = \int_V \nabla \cdot C dv
\]  

(5-13)

Where \( C \) is the vector field, \( n \) is the normal to the differential surface area \( da \), \( dv \) is the differential volume, and \( \nabla \) is the divergence operator.

Assuming the vector field \( C \) is as follows:

\[
C = x\vec{i} + 0\vec{j} + 0\vec{k}
\]  

(5-14)

Therefore the divergence of the vector field \( C \) is

\[
\nabla \cdot C = \frac{\partial C}{\partial x} + \frac{\partial C}{\partial y} + \frac{\partial C}{\partial z} = 1
\]  

(5-15)

Equation (5-13) can be written as
\[ \int_S C \cdot \vec{n} da = \int_V d\nu \]  \hspace{1cm} (5-16)

Then

\[ \int_S C \cdot \vec{n} da = V \]  \hspace{1cm} (5-17)

Where \( V \) is the volume of the object.

For a tessellated surface, equation 5-17 can be written as

\[ V = \sum_i \int_{S_i} C \cdot \vec{n}_i da \]  \hspace{1cm} (5-18)

Where \( S_i \) is the \( i^{th} \) triangular surface, and \( \vec{n}_i \) is the normal to that triangle.

The \( i^{th} \) discrete triangular surface is shown in Fig.5.5. \( V_{i1}, V_{i2}, V_{i3} \) are the vertices of the triangle. \( \vec{n}_i \) is the normal to the triangular surface.

![Figure 5.5 The \( i^{th} \) discrete triangular surface](image)

The triangle \( \{ V_{i1}, V_{i2}, V_{i3} \} \) can be parameterized as

\[ \vec{e}_{i1} = V_{i2} - V_{i1} \]
\[ \vec{e}_{i2} = V_{i3} - V_{i1} \]  \hspace{1cm} (5-19)

Then any point on the triangle can be expressed with the following equation

\[ s(u,v) = V_{i1} + u\vec{e}_{i1} + v\vec{e}_{i2} \]  \hspace{1cm} (5-20)
Where \( u \in [0,1] \) and \( v \in [0,1-u] \)

Then

\[
\bar{n}_i \, da = (\bar{e}_{i1} \times \bar{e}_{i2}) \, dvdu
\]  

(5-21)

Substituting equations 5-20 and 5-21 into equation 5-18, we obtain

\[
V_i = \int_{S_i} C(s(u,v)) \cdot (\bar{e}_{i1} \times \bar{e}_{i2}) \, dvdu
\]

\[
V_i = \int_{S_i} (V_{i1x} + u e_{i1x} + v e_{i2x})(e_{i1y} e_{i2z} - e_{i1z} e_{i2y}) \, dvdu
\]  

(5-22)

Where the integral is evaluated through \( u \in [0,1] \) and \( v \in [0,1-u] \). The integration result is

\[
V_i = \frac{1}{6}\left((V_{i2y} - V_{i1y})(V_{i3z} - V_{i1z}) - (V_{i2z} - V_{i1z})(V_{i3y} - V_{i1y})\right)(V_{i1x} + V_{i2x} + V_{i3x})
\]  

(5-23)

And the total volume of the closed surface is

\[
V = \frac{1}{6} \sum_i \left((V_{i2y} - V_{i1y})(V_{i3z} - V_{i1z}) - (V_{i2z} - V_{i1z})(V_{i3y} - V_{i1y})\right)(V_{i1x} + V_{i2x} + V_{i3x})
\]  

(5-24)

For a tessellated surface, all the coordinates of the vertices are known, thus it is easy to calculate the approximate volume of the closed object with the discrete form of the Gauss-theorem (equation 5-24).

Integrating Newton’s Equation with Euler method

The motion of the mass points observes Newton’s Second Law.

\[
\vec{F}_i = m_i \, \vec{a}_i = m_i \, \frac{d^2 \vec{r}_i}{dt^2}
\]  

(5-25)

Where \( \vec{F}_i \) is the resulting force of the spring force and the pressure force applied
on the $i^{th}$ mass point, $r_i$ is the position vector and $m_i$ is the mass. From equation 5-25, the second order Newton’s equation can be further rearranged as

$$\frac{d^2 \vec{r}_i}{dt^2} = \frac{\vec{F}_i}{m_i}$$

(5-26)

Euler Method and the Accuracy

![Figure 5.6 Euler integration](image)

In mathematics, the Euler method is a first order numerical procedure for solving ordinary differential equations (ODEs) with a given initial value. It is the most basic kind of explicit method for numerical integration of ODEs. With Euler method, a function $r(t)$ can be expressed as equation 5-27

$$r(t + \Delta t) = r(t) + \Delta t \cdot \dot{r}(t) + O(\Delta t^2)$$

(5-27)

If $\Delta t$ is small, function $r(t)$ can be approximated as

$$r(t + \Delta t) \approx r(t) + \Delta t \cdot \dot{r}(t)$$

(5-28)

The dominant error per step is $O(\Delta t^2)$. Therefore, the accuracy depends on the step
size $\Delta t$. The accumulating error may cause numerical instability.

With Euler method, the velocity $V_i$ and position $r_i$ of the $i^{th}$ mass point can be obtained by integrating Newton’s second order differential equation.

$$
\begin{align*}
  a_i(t) &= \frac{F_i(t)}{m_i} \\
  V_i(t + \Delta t) &= V_i(t) + \Delta t a_i(t) \\
  r_i(t + \Delta t) &= r_i(t) + \Delta t V_i(t)
\end{align*}
$$

Collision Detection and Response

When the deformable object collides with other objects, the mass points that penetrate into another object or outside of the enclosure need to be identified and retracted. The following part gives a detailed description about collision detection algorithm and the response of the deformable object after collision.

Two-Step Collision Detection

Because all the current available collision detection packages for tessellated objects can only identify pairs of intersected triangles, further action is required to spot the nodes (vertices of triangles) that actually penetrate into other objects or reside outside of the boundary for each pair of intersected triangles. The following procedure is used for identifying these nodes.

**Step 1: detecting intersected pairs of triangles**

The collision detection package PQP (Proximity Query Package) [UNC GAMMA Group] is used for detecting all the intersected triangle pairs. PQP is a library that can detect whether two models composed of triangles overlap, and optionally, identify all of
the triangles that overlap. The library uses RSS (Rectangular Swept Sphere) bounding volumes for proximity queries, and OBBs (Oriented Bounding Box) [UNC GAMMA Group] for collision detection. One benefit of this collision detection package is that it works on triangle soups, which are an unordered collection of triangles. PQP does not require the object to be closed, and there is no need to know the inside or outside direction of a surface. This feature is very useful for layout design when dealing with enclosures since most of the collision detection algorithms operate uniquely on objects with closed boundaries and outward pointing normal. For a collision detection package such as Swift++ (introduced in chapter 3), for instance, if one object is completely inside another object, these two objects are automatically considered as colliding. PQP will treat these two objects as pair to identify collisions between them. Thus, the PQP approach is very convenient for detecting whether components collide with the enclosure in layout design, and for detecting intersections or collisions between the morphing object and any other one.

**Step 2: identifying the penetrating nodes**

Once the intersected triangles are identified, the following equation is used to check which node is penetrating for each pair of triangles.

\[
(X-P) \cdot N < 0
\]  

(5-30)
Figure 5.7 The penetrating node

Where $X$ is the position of the node (color blue) of the morphable object, $P$ is one of the vertices on the triangular facet of fixed shape object or enclosure, $N$ is the normal to that triangular facet, and $V$ is the velocity of the node which can be obtained from equation 5-29. From Fig. 5.7, it is straightforward that if the equation 5-30 is satisfied, then this node is penetrating. By iterating this procedure for three vertices of each pair of overlapping triangles, all the penetrating nodes can be identified.

Collison Response

Once the penetrating nodes are identified, the next step is retracting the colliding nodes and calculating the velocity of the node after collision. After retracting the node right onto the contact surface, the node is subject to a new force, the reaction force which prevents the node from further penetrating the object. Therefore, the resultant force applied on the node needs to be updated.
Retraction distance

For shape morphing in layout design, an object is morphed to fit the contour of the enclosure or of other objects. Therefore, the penetrating node must be retracted right onto the surface of the penetrated object as shown in Fig. 5.8 (from the red dot inside the object to the yellow dot on its surface).

![Figure 5.8 Retracting the penetrating node to the surface of the penetrated object](image)

Figure 5.8 Retracting the penetrating node to the surface of the penetrated object

The retraction distance $d$ is shown in Fig. 5.9, where $P$ is the penetrating node and $n$ is the normal vector of the triangular facet. To retract $P$ to the triangular facet (yellow

![Figure 5.9 Retraction distance](image)
dot), the retraction distance is the distance from $P$ to the plane where this triangular facet is located on. Assuming the coordinate of $P$ is $(x, y, z)$, the corrected position becomes:

$$
x' = x + \vec{n}_s \cdot d \\
y' = y + \vec{n}_s \cdot d \\
z' = z + \vec{n}_s \cdot d
$$

(5-31)

*Velocity of nodes after collision*

![Diagram showing velocity before and after collision]

The velocity after collision $V'$ is scaled as shown in Fig. 5.10. Where $V_N$ is the normal component velocity, $V_T$ is the tangent component of the original velocity $V$, and $K_r$ is the energy loss coefficient which is less than 1. The velocity after collision can be calculated using the following equations:

$$
V'_T = V_T \\
V'_N = -K_r \cdot V_N
$$

(5-32)

In this application, $K_r$ is set to be 0 since it is desirable for the node to stay on the contact surface after the collision without any rebounding effect.
Reaction force

After retracting the node to the contact surface, to prevent it from further penetrating the object, the node is subject to a new force -- a reaction force. The reaction force \( \vec{N} \) is normal to the contact triangular facet (as shown in Fig. 5.11).

![Diagram showing reaction force](image)

**Figure 5.11 Reaction force**

Assuming the normal of the contract triangular facet is \( \vec{n} \), the resultant of the spring and pressure forces is \( \vec{F} \), then the reaction force can be calculated using the following equation:

\[
N = (\vec{F} \cdot \vec{n})\vec{n}
\]

Therefore, the updated combination force for nodes that are on the contact surface should be:

\[
F' = \vec{F} - (\vec{F} \cdot \vec{n})\vec{n}
\]
Iteration Procedure

The whole procedure of the morphing process is shown in the flow chart of Fig.5.12.

![Flow chart of morphing process](image)

Figure 5.12 Iteration procedure of mass-spring physical model based morphing
Example

The following example (Fig. 5.12, Fig. 5.13 and Fig. 5.14) shows a ball expanding and filling the available space of an arbitrary shaped enclosure which represents the trunk of a car.

Figure 5.13 Initial shape of the ball
Figure 5.14 One interval step during expansion

Table 5.1 Parameters of Mass-spring Model

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of nodes</td>
<td>380</td>
</tr>
<tr>
<td>Number of springs</td>
<td>757</td>
</tr>
<tr>
<td>the constant elasticity factor</td>
<td>755</td>
</tr>
<tr>
<td>the damping term</td>
<td>35</td>
</tr>
<tr>
<td>is the energy loss coefficient</td>
<td>0</td>
</tr>
<tr>
<td>Integration step</td>
<td>0.0001</td>
</tr>
</tbody>
</table>

\[
\Delta t \quad \text{Integration step}
\]
Figure 5.15 The final morphing result

The parameter set up for this implementation is shown in Table.5.1.

The time taken is around 12 minutes on a computer with configuration of dual core 3.61 Hz and 3 GB RAM. The morphing speed becomes slow as the resolution of the mesh increases. The time taken also increases if the volume change is significant as shown in this case. The mesh quality after morphing deteriorates because of the dramatic shape change. Therefore it is necessary to re-mesh the object. This issue is discussed in chapter 7.
Summary

In this chapter, the mass-spring physical model based morphing is studied and implemented. There is no need to explicitly specify the deformations for this method, therefore the shape morphing freedom is not confined. The mass-spring physical model based morphing is a real time method. With the great morphing freedom and reasonable morphing speed, the mass-spring physical model based method is suitable for morphing objects (such as fuel tank and reservoirs) whose shape can be changed arbitrarily in layout design.
CHAPTER 6 BI-LEVEL APPROACH FOR LAYOUT DESIGN WITH MASS-SPRING PHYSICAL MODEL BASED SHAPE MORPHING

Vehicle layout design with shape morphing is a multi-objective problem with a large number of design variables. For large scale problems, decomposition is typically used to make the problem solvable or easier to solve. To solve this complex multi-objective problem, the original problem is decomposed into system level and component level optimizations. The bi-level formulation, the algorithms for system level optimization, the morphing strategy at the component level, and the iteration process between these two levels are studied. This bi-level approach is tailored specially for layout design with mass-spring physical model based shape morphing.

Decomposition of Large Scale Problem

Real world engineering problems are often large scale and quite complex. Layout design with shape morphing belongs to this type of problem. Vehicle layout design such as the under-hood layout is a multi-objective problem with a large number of design variables. The problem consists in placing the engine, radiator, fan, battery, coolant reservoir, air filter, etc inside the tight volume of the under-hood. The design variables include positions ($x, y, z$) and orientations ($\alpha, \beta, \gamma$) of every one of the components. The complexity of the packing problem increases with the number of components. The objectives could consist of dynamic performance, accessibility, maintainability, thermal management, etc. The constraint evaluation such as overlap calculation is very expensive, especially when the shapes of the components are complex. The performance or objective
evaluation could also be very expensive, for example, the multi-body dynamic analysis with ADAMS that predicts the dynamic behavior may take several minutes to hours. Additionally, shape morphing itself is not an easy problem. For example, for the explicit shape morphing method such as parametric mesh based morphing, each control point may have 3 degrees of freedom. To morph an object that could have an arbitrary shape, a very large number of control points would be required. Therefore, the number of design variables could be unmanageable. Incorporating the component shape design (shape morphing) into the layout design process makes the original problem even more complex and difficult to solve. Handling the complete problem all at once is typically beyond our reach with our current knowledge and capabilities.

For complex systems, decomposition is typically used to make the problem solvable or easier to solve. Multilevel multi-criteria programming is a technique used to solve multi-criteria problems with multiple decision makers at various levels [Faulkenberg, 2005]. Instead of optimizing all the objectives at once, the original problem is decomposed into a system level problem with overall objectives $F$, and $N$ component level problems with objectives $f_i$ for each individual decision maker $i$. The combination of component level problems $f_i$ forms the subsystem level problem $f$. Figure 6.1 shows the multilevel decomposition of a large scale problem. There may be some interactions between components, which are not shown in the figure.
Problem Formulation for Layout Design with Mass-spring Physical Model Based Shape Morphing

All-At-Once (AAO) Formulation

The All-At-Once (AAO) formulation of the vehicle layout design with mass-spring physical model based shape morphing is a multi-objective single level optimization problem. The position and orientation of the components and the shape of the morphable components are design variables. The AAO formulation is given as follows:
\[
\min \ F(X, S) \\
\text{s.t.} \ \sum_{i=1}^{n} \sum_{j=1}^{i-1} O_{ij}(X, S) + \sum_{i=1}^{n} C_i(X, S) = 0 \\
g_1(X) \leq 0 \\
Lb \leq X \leq Ub \\
g_2(S) \leq 0 \\
V_i(S) \geq \bar{V}_i \quad i = 1, ..., k
\]

where \( X \) is the vector containing the position coordinates and orientation angles of the components. For the mass-spring physical model based morphing, the shape parameters of the morphable components are implicit. Here, \( S \) represents these implicit shape parameters. \( F(X, S) \) are the given objectives such as dynamic performance, accessibility, maintainability of the vehicles, etc. The first set of constraints includes spatial constraints that require no overlap between any two components \( O_{ij}(X, S) \) and no overlap between components and the enclosure \( C_i(X, S) \). The letter \( n \) represents the number of components. The constraints \( g_1(X) \) incorporate the functionality constraints such as radiator should be put near the front side of the under-hood compartment, transmission axle should be coaxial with the output axle of engine. \( Lb \) and \( Ub \) are the bounds on the position and orientation of the components. \( g_2(S) \) are the constraints on the shape parameters. The last set of constraints consists of the volume constraints that demand the volume of the morphable components \( V_i(S) \) equal to or exceed some targets. The index \( k \) is the total number of morphable components.
Bi-level Formulation

The bi-level formulation is a decomposition of the AAO formulation into a system level and a component level. At the system level, in our case, the design variables are only the position and orientation of the components. The objectives \( F(X, S^*) \) are the same as given in the AAO formulation but with respect to the positions and orientations only. \( S^* \) represents the initial shape of morphable components. The shape of the morphable components is not changed at the system level optimization, therefore \( S^* \) is fixed. The spatial constraints and functionality constraints at this level are the same as those of the AAO formulation. The system level formulation is described as follows:

\[
\begin{align*}
\min & \quad F(X, S^*) \\
\text{s.t.} & \quad \sum_{i=1}^{n} \sum_{j=1}^{i-1} O_{ij}(X, S^*) + \sum_{i=1}^{n} C_i(X, S^*) = 0 \\
\ & \quad g_1(X, S^*) \leq 0 \\
\ & \quad Lb \leq X \leq Ub
\end{align*}
\]

For the given morphable component \( i \), the component level formulation is given as follows:

\[
\begin{align*}
\min & \quad |V_i(S) - \bar{V}_i| \\
\text{s.t.} & \quad \text{no collision between any components or enclosure} \\
\ & \quad g_2(S) \leq 0
\end{align*}
\]

The design variables of the component level is the implicit shape parameters \( S \) of the morphable components. At this level, for each morphable component, the objective is to maximize its volume \( V_i \) or reach a target volume \( \bar{V}_i \). At the component level, the geometrical representation of components is a triangular tessellation because the
morphing method is mesh based as we have chosen. The spatial constraints specify that there is no collision between any components or enclosure. Since there is no known software that is capable of calculating overlap for tessellated models, “no collision” is used as the spatial constraint instead of “no overlap” in the system level. The overlap evaluation would have returned the degree of overlap and could then be used to estimate how much to retract the components. The no collision evaluation can only return the information that whether or not there is a collision. \( g_2(S) \) are the constraints on the shape parameters.

The Bi-level Iteration Process

With this bi-level formulation, the original problem is described more clearly. At each level a simpler problem is presented. The above formulation is the general formulation of the bi-level decomposition. However, this formulation is not sufficient for solving our problems. The problem formulations in both system and component levels are further expanded as shown in Fig. 6.2. The system level has three phases: global search, local search and adjacent objects expanding optimization. The component level has one phase – volume expansion for each of the morphable components. The overall optimization strategy of the bi-level iteration process is as follows:

At the system level

The purpose of system level search is to find the optimal locations and orientations of fixed components with respect to the system performance objectives. Morphable components are considered fixed at that level.
At the component level

The purpose of component level search is to increase the volume of each morphable component to reach its target volume while satisfying the no collision constraints.

Figure 6.2 Flow chart of the bi-level iteration process

The optimization process starts from the system level global search, and goes to the component level. The iteration process is described as follows.

Step 1: The system level global search optimizes the performance objectives with respect to the position and orientation of the components only. When the
global search is finished, a set of solutions is obtained for multi-objective problems.

Step 2: Select a solution according to the designer’s preference, and convert the multi-objective problem into single objective problem using for instance the weighted-sum approach. The weights are assigned according to the preference.

Step 3: For each select solution, the optimization goes to the component level. At the component level, the morphing algorithm tries to increase the volume of each morphable component to reach its target volume while satisfying the no collision constraints.

Step 4: If the target volume cannot be reached, then the optimization returns to the system level. The adjacent objects expanding optimization is performed. The positions of those objects adjacent to the morphable components are adjusted along the contact normal direction. The details are described in the following part.

Step 5: After the adjacent objects are moved, the optimization goes down to the component level for further volume expansion. Steps 4 and 5 are iterative incremental processes. They are repeated until the target volume is obtained.

Step 6: When the target volume is obtained, the optimization goes up to the system level for local search to fine tune positions. The performance objectives are optimized with the new shapes of the morphable components and the new bounds on positions and orientations. The details are described in the following part.
Step 7: If the system level local search result is desirable, the whole iteration stops. Otherwise, a new system level global search needs to be restarted.

For each of the select solutions, the above described procedures are repeated. In the following sections, the detailed formulations for each sub-search and the corresponding optimization algorithms are presented to further explain this bi-level iteration process.

Detailed Problem Formulation and Corresponding Optimization Algorithms

System Level Problem Formulation

The system level optimization has three phases: global search, local search and adjacent objects expanding optimization.

System level global search

The global search is performed to find the approximate positions and orientations of the components with the fixed shapes $S^*$. The first phase can be viewed as finding the “pattern” of one configuration. The formulation is as follows.

$$\min \ F(X, \ S^*)$$

$$s.t. \ \sum_{i=1}^{n} \sum_{j=1}^{i-1} O_{ij}(X, \ S^*) + \sum_{i=1}^{n} C_i(X, \ S^*) = 0$$

$$g_1(X, \ S^*) \leq 0$$

$$Lb \leq X \leq Ub$$

Where $F(X, \ S^*)$ are the overall system performance objectives. The design variables $X$ are the positions and orientations of the components. The constraints consist of the no overlap constraint $\sum_{i=1}^{n} \sum_{j=1}^{i-1} O_{ij}(X, \ S^*) + \sum_{i=1}^{n} C_i(X, \ S^*) = 0$, the functionality
constraints \(g_1(X, S^*)\), and the bounds (\(Lb\) and \(Ub\)) on the positions and orientations of the components. The optimization result of the global search is a Pareto front for multi-objective problems. Designers can choose solutions according to their preferences for further optimization. The variables passing down to the component level are the current optimal positions and orientations (\(X\)) of the components, while the constants passing down to component level are the current fixed shapes of the morphable components \(S^*\). For the system level global search, the NSGAII is chosen as the optimization algorithm.

**System level local search**

The system local search determines the exact positions and orientations of the components with the new shapes (\(S^N\)) for morphable components and new bounds on positions and orientations. This phase can be viewed more as the local perturbation of the global “pattern”.

\[
\min \sum w_k * F_k(X, S^N)
\]

\[\text{s.t.} \sum_{i=1}^{n} \sum_{j=1}^{n} O_{ij}(X, S^N) + \sum_{i=1}^{n} C_i(X, S^N) = 0\]

\[g_1(X, S^N) \leq 0\]

\[Lb - l \leq X \leq Ub - l\]

Where \(\sum w_k * F_k(X, S^N)\) is the system performance objective converted from the original multi-objective problem by the weighted-sum approach for instance, and \(w_k\) is the assigned weight for each objective. The sum of weights should be 1. \(k\) is the number of system performance objectives. The design variables \(X\) are the positions and orientations of the components. The constraints consist of the no overlap constraints.
\[ \sum_{i=1}^{n} \sum_{j=1}^{i-1} O_{ij}(X, S^N) + \sum_{i=1}^{n} C_i(X, S^N) = 0, \]
the functionality constraints \( g_1(X, S^N) \), and the
new local bounds (\( Lb\_l \) and \( Ub\_l \)) on the positions and orientations of the components.

The optimization result of the local search is a single solution for the converted
performance objective. The variables passing down to the component level are the current
optimal positions and orientations (\( X \)) of the components. The constants passing down to
component level are the current shapes (\( S^N \)) of the morphable components. For the
system level local search, the gradient based method FSQP is chosen as optimization
algorithm.

**System level adjacent objects expanding optimization**

In the case where there is no sufficient room for the expanding of the morphable
components, the positions of those objects that are adjacent to one morphable component
need to be adjusted. Adjacent objects expanding optimization determines the optimal
moving magnitudes of those objects adjacent to the morphable components along the
contact normal direction, while optimizing the system performance objectives. The
expanding is an incremental process. For each expanding step, the following optimization
is performed.

\[
\min \sum_{k} w_k \cdot F_k(\delta, S^N)
\]
\[s.t. \quad \sum_{i=1}^{n} \sum_{j=1}^{i-1} O_{ij}(\delta, S^N) + \sum_{i=1}^{n} C_i(\delta, S^N) = 0
\]
\( g_1(\delta, S^N) \leq 0 \)
\( Lb\_l \leq X(\delta) \leq Ub\_l \)
\( lb \leq \delta \leq ub \)
\( \sum \delta_m \geq \Delta_{\min} \)
Similar as the local search, \( \sum w_k \cdot F_k(\delta, S^N) \) is the system performance objective converted from the original multi-objective problem by the weighted-sum approach. \( w_k \) are the assigned weights for each objective, and the sum of the weights should be 1. The design variables \( \delta \) are the moving magnitudes of the components adjacent to the morphable components for each incremental step. The moving direction is along the contact normal direction, which is dictated by the expanding action of the morphable components. The constraints include the no overlap constraints \( \sum_{i=1}^{n} \sum_{j=1}^{i} O_{ij}(\delta, S^N) + \sum_{i=1}^{n} C_{i}(\delta, S^N) = 0 \), the functionality constraints \( g_i(\delta, S^N) \), and the new local bounds (\( Lb_\_l \) and \( Ub_\_l \)) on the positions and orientations (\( X(\delta) \)) of the components. Now the positions (\( X(\delta) \)) of the components are function of the moving magnitudes \( \delta \). \( lb \) and \( ub \) are the bounds on these moving magnitudes. \( \Delta_{\text{min}} \) is the minimal sum of the moving magnitudes, which is used to ensure there is sufficient room for expansion. The selection of \( \Delta_{\text{min}} \) shall be proportional to the size of the morphable object. The optimization result is a single solution for the converted performance objective. The variables passing down to the component level are the current optimal positions and orientations (\( X \)) of the components. The constants passing down to the component level are the current shapes (\( S^N \)) of the morphable components. For the system level adjacent object expanding optimization, the gradient based method FSQP is chosen as the optimization algorithm.
Component Level Problem Formulation

For the component based optimization, the major issue is to morph the component to the desired volume, thus, the optimization can be stated as:

$$
\text{min} \ |V_i(S) - \bar{V}_i|
$$

s.t. no collision between any components or enclosure
$$
g_2(S) \leq 0
$$

Where the design variables $S$ are the implicit shape parameters for the morphable components. The constraints include the no collision constraints between any components and enclosure, and the constraints on shape parameters $g_2(S)$. The variables passing up to the system level are the morphed shapes of the components $S^N$, and the new center of gravity of the morphed objects. The morphing method is the physical based mass-spring shape morphing method.

With the detailed problem formulations for each sub-search, the general bi-level optimization is tailored specially for our problem. To implement this bi-level optimization process, there are still two important CAD related issues need to be addressed.

Two Important CAD Aspects in the Bi-level Iteration Process

In this bi-level iteration process, there are two important aspects that need to be further addressed. One is how to check the status that the target volume cannot be reached with the current components configuration, the one resulting from the top level optimization. The procedure is shown in Fig.6.3. While increasing the pressure inside the morphable component to increase its volume, if the volume difference between two subsequent iterations is less than a tolerance (0.00001 in this implementation), and
continues for $M$ iterations (20 iterations in this implementation), then the algorithm considers that the target volume cannot be obtained.

Another CAD issue is how to calculate the contact normal between a morphable component and its adjacent components. In this application, it is assumed that when the target volume cannot be obtained, the morphable component will continue to increase in size and adaptively push its adjacent components outward from its center of mass along the contact normal directions if possible. The contact normal is calculated using the contact query functions provided by Swift++ [UNC GAMMA Group]. Note that the magnitude of motion is controlled and kept within limits in order to avoid the situation where an object can move past its adjacent object. These limits are arbitrarily chosen, and depend on the problem at hand.
Figure 6.3 Identifying the status that the target volume cannot be reached

Example

In this section, an example is given to illustrate the above bi-level approach. The whole system includes four objects as shown in Fig 6.4. The ball is morphable.
Figure 6.4 Packing components in the available packing space

The general problem formulation is as follows:

*System level problem formulation*

**Objectives:**

\[ \text{Min.} \quad I, \quad \text{Moment of Inertia of the system with respect to axis (0, y, -0.5)} \]

\[ d, \quad \text{Ground clearance, } d = 0 - \text{bottom position of the lowest object} \]

**Constraints:**

No overlap

Bounds on the position range of the objects
Design Variables:

Positions of objects

*Components level problem formulation*

Objective:

Expand the volume of the ball to a target value 8.5 units

Constraint:

No collision

Design Variables:

Shape of the ball

The system level global search result is shown in Fig.6.5. Since the two objectives are conflicting, a Pareto front is obtained. The solutions A, B and C are chosen to illustrate the proposed bi-level approach.
The system level global search result of solution A is shown in Fig. 6.5. For solution A, it is assumed that the minimal moment of inertia is preferred; therefore the weights assigned to moment of inertia and ground clearance are chosen as 0.8 and 0.2 respectively. The problem formulations for component level search and system level local search are presented as follows.
Component level problem formulation

Target:

Expanding volume of the ball to a target value 8.5 units

Constraint:

No collision

In this case, the target volume is reached without pushing the adjacent objects away.

The result is shown in Fig 6.7. Next, the optimization goes up to the system level local search.

Figure 6.6 System level global search result of solution A
Figure 6.7 System level expanding optimization and subsequent component level volume expansion result of solution A

_System level local search formulation_

Objective:

\[ \text{Min. } 0.8I + 0.2d \]

Constraints:

No overlap

Local bounds on the position range of the objects

Design Variables:

Positions of objects

The optimization result is shown in Fig 6.8.
Figure 6.8 System level local search result of solution A

The moment of inertia and ground clearance after each search are shown in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Moment of Inertia</th>
<th>Ground Clearance</th>
</tr>
</thead>
<tbody>
<tr>
<td>System level global search</td>
<td>103.293000</td>
<td>1.063730</td>
</tr>
<tr>
<td>System level expanding optimization and component level volume expansion</td>
<td>139.676319</td>
<td>1.063730</td>
</tr>
<tr>
<td>System level local search</td>
<td>139.601311</td>
<td>1.063536</td>
</tr>
</tbody>
</table>
Solution B

The system level global search result of solution B is shown in Fig.6.9. For solution B, assume minimal ground clearance is preferred, therefore the weights assigned to moment of inertia and ground clearance are chosen as 0.2 and 0.8 respectively. The problem formulations for component level search, system level expanding optimization and system level local search are presented as follows:

Figure 6.9 System level global search result of solution B
Component level problem formulation

Target:

Expanding volume of the ball to a target value 8.5 units

Constraint:

No collision

In this case, the target volume cannot be reached without moving the objects adjacent to the morphing component D. Therefore, the algorithm proceeds to the system level adjacent objects expanding optimization.

System level expanding optimization

Objective:

\[ \text{Min. } 0.2*I + 0.8*d \]

Constraints:

Bounds on the translation magnitudes along the contact normal direction

\[ 0.0 \leq \delta \leq 0.3 \]

Minimal sum of moving magnitudes

\[ \sum \delta_m \geq 0.3 \]

Bounds on the position range of objects

No overlap

Design Variables:

Translation magnitudes along the contact normal directions \( \delta \)

Optimization result:

Moving magnitudes along the contact normal between the ball and its
adjacent objects:

\[ \delta_A = 0.249471 \]
\[ \delta_B = 0.101769 \]
\[ \delta_C = 0.0 \]

After adjusting the positions of the adjacent objects, the algorithm proceeds to the component level optimization for further volume expansion. This time, the target volume is obtained (Fig 6.10), and there is no need to further move the adjacent objects. If the target volume could not be reached at that step, another system level expanding optimization would have had to be performed. Now, the algorithm goes up to the system level local search.
Figure 6.10 System level expanding optimization and subsequent component level volume expansion result of solution B

*System level local search formulation*

Objective:

\[ Min. \ 0.2*I+0.8*d \]

Constraints:

No overlap

Local bounds on the position range of the objects

Design Variables:

Positions of objects

The optimization result is shown in Fig 6.11.
The moment of inertia and ground clearance after each search are shown in Table 6.2.

Table 6.2 Objective Values after Each sub-search of Solution B

<table>
<thead>
<tr>
<th>Value</th>
<th>Moment of Inertia</th>
<th>Ground Clearace</th>
</tr>
</thead>
<tbody>
<tr>
<td>System level global search</td>
<td>245.384000</td>
<td>0.492368</td>
</tr>
<tr>
<td>System level expanding optimization and component level volume expansion</td>
<td>273.245018</td>
<td>0.492368</td>
</tr>
<tr>
<td>System level local search</td>
<td>259.638373</td>
<td>0.492977</td>
</tr>
</tbody>
</table>
Solution C

The system level global search result of solution C is shown in Fig. 6.12. For solution C, assume both aspects of ground clearance and inertia are equally important, so the same weight 0.5 is assigned to them. The problem formulations for component level search, system level expanding optimization and system level local search are presented as follows:

Figure 6.12 System level global search result of solution C
Component level problem formulation

Target:

Expanding volume of the ball to a target value 8.5 units

Constraint:

No collision

In this case, the target volume cannot be reached without moving the objects adjacent to D. Therefore, the algorithm branches to the system level expanding optimization.

System level expanding optimization

Objective:

Min. \(0.5I + 0.5d\)

Constraints:

Bounds on the translation magnitudes along the contact normal direction

\[0.0 \leq \delta \leq 0.3\]

Minimal sum of moving magnitudes

\[\sum \delta_m \geq 0.3\]

Bounds on the position range of objects

No overlap

Design Variables:

Translation magnitudes along the contact normal directions \(\delta\)

Optimization result:

Moving magnitudes along the contact normal between the ball and its
adjacent objects

\[ \delta_A = 0.10 \]
\[ \delta_B = 0.10 \]
\[ \delta_C = 0.10 \]

After adjusting the positions of the adjacent objects, the algorithm goes down to the component level for further volume expansion. This time, the target volume can be obtained (Fig 6.13) without further moving the adjacent objects. Then the optimization goes up to system level local search.

Figure 6.13 System level expanding optimization and subsequent component level volume expansion result of solution C

*System level local search formulation*

Objective:
\[ \text{Min. } 0.5*I + 0.5*d \]

Constraints:

No overlap

Local bounds on the position range of the objects

Design Variables:

Position of objects

The optimization result is shown in Fig 6.14.

Figure 6.14 System level local search result of solution C

The moment of inertia and ground clearance after each search are shown in the following Table 6.3.
<table>
<thead>
<tr>
<th>Value</th>
<th>Moment of Inertia</th>
<th>Ground Clearance</th>
</tr>
</thead>
<tbody>
<tr>
<td>System level global search</td>
<td>166.631000</td>
<td>0.683883</td>
</tr>
<tr>
<td>System level expanding optimization and component level volume expansion</td>
<td>218.624989</td>
<td>0.783883</td>
</tr>
<tr>
<td>System level local search</td>
<td>212.334565</td>
<td>0.676976</td>
</tr>
</tbody>
</table>

Summary

For large scale problems, decomposition is typically adopted to make the problem solvable or easier to solve. In this chapter, a bi-level approach for layout design with the mass-spring physical model based morphing is studied. By decomposing the original problem into system and component levels, at each level, a simpler problem is formed and solved. The system level optimization is further classified into system level global search, system level local search and system level adjacent objects expanding optimization. For each sub-search, the problem formulation and corresponding optimization algorithm are studied. The whole problem is solved by iterating between system level and component level. Finally, an example is given to illustrate the proposed bi-level approach.
CHAPTER 7 GENERAL METHODOLOGY FOR LAYOUT DESIGN WITH SHAPE MORPHING

There are various applications of packing with shape morphing problems in industry as reviewed in chapter two. Although these applications may have different objectives and constraints, they share some common issues such as CAD model preprocessing for packing purpose, data format translation during the packing process if performance evaluation and morphing use different representation methods, efficiency of collision detection methods, etc. This chapter studies several common issues under the framework of a general methodology for layout design with shape morphing.

The General Layout Design with Shape Morphing Process

Figure 7.1 General layout design with shape morphing process

The general layout design with shape morphing process is shown in Fig. 7.1. It includes three main stages. The first stage is CAD model preprocessing. Many components such as an engine, transmission or radiator, are assemblies of a multitude of
small parts. The internal structures of these components are very complex. Since only the external shape of the object is relevant in the layout design process, using the detailed CAD model results in a huge waste of computation time and may totally hinder the ability to find acceptable solutions. Therefore, it is necessary to simplify the CAD models for packing purposes. A detailed CAD model can be simplified by extracting the external shells of each relevant component. With the extracted shells, precision control may be needed to simplify the model further for computational efficiency. Besides the simplification of components, the packing space needs to be identified since the space may not be obvious or closed. For example, the automotive under-hood is usually composed of many pieces of sheet metal parts, which are not connected. The under-hood is also not closed in the bottom, but ground clearance in effect establishes a virtual bottom plane. Therefore, the available and closed packing spaces must be identified before packing starts. The techniques for shell extraction, precision control and identification of packing space are discussed in the following sections.

The second stage is the layout design with shape morphing process. It includes the set up of the computation model for packing (analysis) and the coupling of the packing model with the optimizer for optimization. The packing model provides the necessary function evaluation information for the optimizer. The generic packing model includes four functional parts: object layout, shape morphing, performance evaluation and data format translation. The details of the generic model for layout design with shape morphing are presented in the following sections.

The last stage of a layout design process is validation of the layout with the
detailed CAD models. When presenting the final optimized layout, it is preferred to use the original CAD models instead of the simplified ones. However, due to simplification, the inconsistency between the original models and the simplified ones may cause interferences. At this stage, these interferences need to be identified and the layout or CAD models may need to be modified interactively by the designer.

**Generic Model for Layout Design with Shape Morphing**

![Diagram of Generic Model for Layout Design with Shape Morphing](image)

**Figure 7.2 Generic model for layout design with shape morphing**

The generic packing model is a computation model to simulate the layout design with shape morphing process, which provides the necessary function evaluation information for the optimizer. To make the computational model easy to be extended for different applications, the model is decomposed into four parts. These are: object layout, shape morphing, performance evaluation and data format translation. The functions of these modules are relatively independent. The relationships of the four functional parts are
shown in Fig. 7.2. The generic packing model for layout design with shape morphing can be viewed as a layered system. The arrows show the dependency relationships, where the functional parts at the start of the arrows invoke the services of the parts at the end of the arrow. The bottom layer is the fundamental CAD systems layer. If the shape morphing and performance evaluation are based on the same CAD representation, CAD system 2 will be the same as CAD system 1 and no data format translation would be required. Otherwise, the data format has to be converted through the data format translator. In our application, the CAD system 1 is a B-rep/CSG based CAD system -- ACIS, while CAD system 2 is based on triangular tessellation. The detailed functions and relationships are presented in the following sections.

Performance Evaluation

The function of the performance evaluation module is to evaluate the objectives and constraints of the packing problem. For a complex problem such as vehicle layout design, the objectives can include dynamic performance, maintainability, and others, and the constraints can include spatial constraints (no overlap) and functionality constraints. The performance evaluation part uses Boolean operation functions supplied by the CAD system to evaluate constraints such as overlap. When evaluating the objectives related to geometry information such as maintainability, the functions of the CAD systems are also invoked. The performance evaluation part uses the object layout part to place the components at the specified locations. The shape morphing function is invoked if the spatial constraints cannot be satisfied.

In our application, geometry related function evaluations such as overlap
calculation are implemented using the ACIS 3D modeling library [Spatial Corp.]. ACIS uses B-rep and CSG to create and represent 3D CAD models. ACIS does not provide a direct function to calculate overlap between two objects. The overlap calculation has to be implemented using two primitive functions supplied by ACIS: Volume and Unite. The Volume function calculates the volume of a solid object, while the Boolean function Unite returns a unification of two objects. Therefore, the overlap of object 1 and object 2 can be calculated as follows:

\[
\text{Overlap} = \text{Volume (obj1)} + \text{Volume (obj2)} - \text{Volume (Unite (obj1, obj2))}
\]

Object Layout

The object layout function places the components at the locations and in the orientations specified by the optimizer. It uses the transformation functions provided by the CAD system to transform the position and orientation of the objects. Besides placing the objects according to the global coordinates, it can also place the objects relative to the coordinates of other components. This is useful to place components that have mechanical or functional links such as object 2 should be coaxial with object 1.

Shape Morphing

The shape morphing function part morphs the shape of the components according to spatial constraints (no overlap), volume constraints and possibly some shape constraints. It is invoked by the performance evaluation module for components with shape morphing freedom. The shape morphing methods determine which kind of CAD
representation should be used. For example, mesh-based morphing uses the tessellation model, while the parameterization-based morphing requires a solid representation (B-rep or CSG) CAD system.

Data Format Translation

Figure 7.3 Algorithm for converting tessellated model to ACIS solid model

If the shape morphing and performance evaluation are based on different CAD representations, the data format has to be converted between these two function parts. For
example, in our applications, the performance evaluation is based on the B-rep/CSG CAD system ACIS, while the mass-spring physical model based shape morphing uses a tessellated model. In this case, to evaluate the overall performance objectives with morphed objects, the data format of morphable components needs to be translated from mesh to solid model. Meanwhile, the solid models must be tessellated for morphing purposes.

ACIS does not provide a direct function to convert tessellated model to ACIS solid model. To facilitate this data format translation process, a data format translator based on the ACIS CAD system is developed. The following ACIS functions are used: compose wire from points (api_make_wire), compose face from wire (api_cover_wires), and compose closed ACIS solid object by stitching faces (api_stitch). The algorithm for this conversion is shown in Fig.7.3.

**CAD Model Preprocessing**

CAD model preprocessing includes two aspects: simplify the detailed CAD models by extracting the external shells with controlled precision levels and identify the available and closed packing space. In this application, the CAD model preprocessing uses the surface wrapping function of commercial software ANSA [Beta CAE Systems Inc.].

**Preprocessing with Surface Wrapping**

There are several software packages such as ANSA, Fluent TGrid [ANSYS Inc.] and STAR-CD [CD-adapco Inc.], which provide surface wrapping function. Users can
wrap multiple disconnected geometries into a single, connected and high-quality surface mesh. In this application, the wrapping function in ANSA is tested.

The wrapping function has two options: wrapping from the outside or wrapping from the inside of the geometry. The outer surface wrapping can be used for extracting the outer boundary of complicated components. At the early stage of a layout design process, the coarser models are preferred for packing purpose. However, at the final stage to validate the layout, the more detailed models are required. By giving different element sizes, the precision of the extracted shells can be controlled. If a large enough element size is given, the outer wrap can close large gaps between objects or surfaces (as shown in Fig.7.6). This helps obtaining a closed surface. If a proper element size is given, the inner wrapping can extract the inner volumes from complex parts. The following example shows how to extract the outer shell of an engine (Fig.7.4) with different element sizes (Fig. 7.5(a), (b) and (c)) in ANSA.
Figure 7.4 Original detailed engine model
Figure 7.5 Outer shell of the original engine model with different precision control

Figure 7.6 The original disconnected sheet metal model of a trunk
The following procedure shows how to obtain the packing space of a trunk as shown in Fig. 7.6. There are two steps. First, an outer wrapping with big element size to cover the big holes (Fig. 7.7) is performed. Once the outer wrap is accomplished, an inner wrap with smaller element size can be executed. To find the element size that can extract the desired volume, several trials are necessary. Note that this process has to be done at the initial stage of the process and does not have to be repeated, thus its computational cost is not too important. Fig. 7.8 shows the final packing space obtained.

Figure 7.7 Outer wrap to close the hole with large element size
The mesh quality of the morphed object deteriorates especially with dramatic shape changes. Therefore it is necessary to re-mesh the object after and sometime during morphing. In this application, the Delaunay triangulation is used for mesh regeneration.

A Delaunay triangulation for a set of points $P$ in the plane is a triangulation $DT(P)$ such that no point in $P$ is inside the circum-circle (circum-sphere in 3D) of any triangle (tetrahedron in 3D) in $DT(P)$ [O’rourke, 1998]. Delaunay triangulations maximize the minimum angle of all the angles of the triangles in the triangulation, which tend to avoid skew triangles. The following picture shows how to construct the Delaunay triangulation by flipping an edge. For two triangles $ABD$ and $BCD$ with the common edge $BD$ (Fig. 7.9), if the sum of the angles $\alpha$ and $\gamma$ is greater than $180^\circ$, substitute the common edge $BD$
with the common edge AC.

In this implementation, the Hull [Clarkson, 1995] is integrated into the morphing code for re-meshing the object after morphing. Hull is based on an incremental algorithm [O’rourke, 1998] for Delaunay computation. The incremental algorithm computes the Delaunay triangulation by repeatedly adding one vertex at a time and re-triangulating the

![Diagram of Delaunay triangulation](image)

**Figure 7.9** Construct the Delaunay triangulation by flipping affected parts. When a vertex is added, a search is performed for all triangles (tetrahedrons in 3D) whose circum-circles (circum-spheres in 3D) contain this vertex. Then, those triangles (tetrahedrons in 3D) are removed and the affected parts are re-triangulated. The following example shows the re-meshing of a morphed object with the Delaunay triangulation.
Figure 7.10 Re-mesh with Delaunay triangulation

Collision Detection Efficiency
In this implementation, the Boolean operations of ACIS 3D CAD system are used to calculate the overlap. This is very expensive, especially when the complexity of the geometric models increases. In the packing process, the overlap is calculated for each pair of objects even if they do not intersect, which is a great waste of computational time. For a problem with \( n \) objects to be packed, \( \frac{(n-1)n}{2} \) calculations have to be performed. When \( n \) becomes large, the computational cost becomes very high. Therefore, instead of directly applying Boolean operations for objects that even do not overlap, the intersection query function [Spatial Corp.] is used first. The intersection query function checks if the bounding boxes of two objects intersect. A bounding box is a minimum size box containing the actual object. The following table shows a test comparing the computation time in two cases based on the ACIS CAD system, where the under-hood and the engine do not intersect actually.

### Table 7.1 Computation Time for Collision Detection

<table>
<thead>
<tr>
<th>Case</th>
<th>Overlap Calculation Time (milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Using Boolean operations to calculate the overlap of under-hood and engine</td>
<td>381</td>
</tr>
<tr>
<td>Using intersection query function to check if bounding box of under-hood and bounding box of engine intersect</td>
<td>3</td>
</tr>
</tbody>
</table>
Summary

This chapter presented the general layout design with shape morphing process, which includes three stages: CAD model preprocessing, layout design by coupling the generic packing model with the optimizer, and validation layout with the detailed model. The generic model for layout design with shape morphing process is composed of four functional parts: performance evaluation, object layout, shape morphing, and the data format translation. The functions of those modules and the relationships between them are explained. A data format translator is developed for layout design with mesh-based morphing. The CAD model preprocessing with surface wrapping functions of commercial software ANSA are investigated. Finally, an improvement of the collision detection efficiency is obtained by utilizing the intersection query function (based on ACIS) for checking whether the bounding boxes of two objects intersect before calculating their possible overlap with Boolean operations.
CHAPTER 8  VEHICLE LAYOUT DESIGN WITH SHAPE MORPHING

With all the components of the layout design process have been developed and explained, this chapter implements the process and applies it to the vehicle under-hood/underbody layout design problem. There are more and more parts that need to be packed under the hood or under the body of a vehicle with the increase in complexity and the need to reduce energy consumption. Thus, the space available under the hood or the body continues to be reduced and overcrowded. How to fully and efficiently exploit the limited space is a critical issue for a successful layout design. The under-hood/underbody layout design with shape morphing is a multi-objective optimization problem. It can be formulated as the process to find the optimal locations and orientations and the suitable

Figure 8.1 Ford Taurus under-hood system
shapes of select morphable components such that the specified performance objectives are optimized without violating any space or functionality constraints.

In this chapter, two examples are shown to illustrate the layout design with mass-spring physical model based morphing. One case is to design a fuel tank according to the available underbody space of a Ford Taurus. Another case shows a Ford Taurus under-hood layout design with the objectives of minimizing the moment of inertia, ground clearance, and maintainability, while at the same time increasing in the volume of the coolant reservoir.

Ford Taurus Underbody Fuel Tank Design with Mass-spring Physical Model Based Morphing

The following example consists of designing fuel tanks according to the available underbody space with two different pipe layouts. Fig.8.2 shows the original underbody model. The available packing space for the fuel tank is given approximately by the vehicle underbody profile, some necessary frame boundaries, the exhaust pipe (as shown in Fig.8.3), and the ground clearance of the vehicle. The fuel tank is obtained by expanding a sphere into the available space as shown in Fig.8.4.

The original underbody layout can be improved using a straight pipe instead of the original curved ones. The benefit is to reduce the length of pipe and the number of bends (location of points where the pipe changes its direction), which are the objectives of an optimal pipe layout [Sandurkar and Chen, 1999]. The morphing result of the fuel tank is shown Fig.8.5 in the shape of saddle. The driveshaft on AWD (all wheel drive) and RWD (rear wheel drive) vehicles also demands saddle shaped tanks as shown in Fig.8.6.
Figure 8.2 Ford Taurus underbody system

Figure 8.3 Approximate packing space for fuel tank
Figure 8.4 Morphed fuel tank with a curved pipe
Figure 8.5 Morphed fuel tank with a straight pipe
This example shows the flexibility of combining the component shape design with different under-body layouts. The time taken for each case is around 16 minutes on a computer with configuration of dual core 3.61 Hz and 3 GB RAM. The final volume is 120000000 units for both cases as chosen. It can be seen from Fig.8.6, in the second case, there are still space for further expansion if needed.

Figure 8.6 Saddle-shaped fuel tank of Ford Mustang

Ford Taurus Under-hood Layout Design with Mass-spring Physical Model Based Morphing

In this example, there are six components that need to be packed into the
under-hood compartment (Fig.8.7). These components include engine block, radiator, battery, coolant reservoir, air filter, and master brake cylinder. The degree of freedom of each component is shown in Table 8.1. The coolant reservoir is morphable, and the shapes of the other objects are fixed. The objectives consist of minimizing the moment of inertia, ground clearance, and maintainability of the under-hood system, as well as increasing the volume of the coolant reservoir.

At the system level, the ACIS 3D modeling library, which is based on B-rep and CSG representations, is used to handle the 3D vehicle components. Through ACIS, the overlap calculation, evaluation of the moment of inertia, ground clearance and maintainability, and the data format translation between B-rep to tessellation are implemented. At the component level, the triangular representation (tessellation) is used. The collision detection is performed using the PQP program described earlier, which is a collision detection package for objects represented in a triangulated format [UNC GAMMA Group].
Figure 8.7 Packing components in the under-hood compartment

Table 8.1 Under-hood Components and Degree of Freedom

<table>
<thead>
<tr>
<th>Component</th>
<th>Degree of freedom</th>
</tr>
</thead>
<tbody>
<tr>
<td>Engine block</td>
<td>$x, y, z$</td>
</tr>
<tr>
<td>radiator</td>
<td>$x, z$</td>
</tr>
<tr>
<td>battery</td>
<td>$x, y, z$</td>
</tr>
<tr>
<td>Coolant reservoir</td>
<td>$x, y, z$</td>
</tr>
<tr>
<td>Air filter</td>
<td>$x, y, z$</td>
</tr>
<tr>
<td>Brake cylinder</td>
<td>$x$</td>
</tr>
</tbody>
</table>
The general bi-level problem formulation of this problem and the detailed formulations are presented in the following sections.

The General Problem Formulation

System Level Problem Formulation

Objectives:

\[ \text{Min. } I, \text{ Moment of Inertia of the system with respect to axis (0, y, -100.0) and } \]
\[ \text{axis (x, 0, -100.0)} \]
\[ d, \text{ Ground clearance, } d = 0 - \text{bottom position of the lowest object} \]

Maintainability (described in the following part)

Constraints:

No overlap

Bounds on the position range of the objects

Design Variables:

Position of objects

Components Level Problem Formulation

Objective:

Expand the volume of the coolant reservoir to a target value 12000000 units

Constraint:

No collision

Constraints on the shape of the coolant reservoir (explained below).

Design Variables:
Shape of the coolant reservoir

The Definition of Maintainability of the System

The accessibility of a component along a specific direction is defined as the number of objects to be removed before this given object can be removed along this direction. For example, as shown in Fig.8.8 [Miao, 2005], the object 2 has an accessibility of 2 because the object 3 has to be removed first before object 2 can be removed. Considering the difficult levels of removing different components, a weight is assigned for each component in a system. Therefore, the maintainability of the whole system is defined as weighted sum of accessibility of all components in the system. The weights of accessibility of the under-hood components are chosen as shown in Table 8.2.

![Accessibility Diagram](image)

Figure 8.8 Definition of accessibility along a specific direction
Table 8.2 Weights of Accessibility

<table>
<thead>
<tr>
<th>Component</th>
<th>Weights of accessibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Engine block</td>
<td>10</td>
</tr>
<tr>
<td>Radiator</td>
<td>5</td>
</tr>
<tr>
<td>Battery</td>
<td>1</td>
</tr>
<tr>
<td>Coolant reservoir</td>
<td>1</td>
</tr>
<tr>
<td>Air filter</td>
<td>5</td>
</tr>
<tr>
<td>Master brake cylinder</td>
<td>10</td>
</tr>
</tbody>
</table>

The system level global search result is shown in Fig. 8.9. Since the moment of inertia and ground clearance are conflicting, a Pareto front is obtained for these two objectives. The maintainability is 10 for all of the solutions shown in the Pareto front. The solutions A, B and C are chosen for illustrating the proposed bi-level approach.
Solution A

The system level global search result of solution A is shown in Fig. 8.10. For solution A, assume the minimal moment of inertia is the preferred objective, therefore the weights assigned for moment of inertia and ground clearance are chosen as 0.8 and 0.2 respectively. The problem formulations for component level search, system level expanding optimization and system level local search are presented as follows.
Component Level Problem Formulation

Target:

Expanding volume of the coolant reservoir to 12000000 units

Constraints:

No collision

Shape constraints of coolant reservoir
\[ x_{\text{max}} - x_{\text{min}} \leq 240 \]
\[ z_{\text{max}} - z_{\text{min}} \leq 260 \]

In this case, the target volume cannot be reached without moving the adjacent objects. Therefore, the algorithm proceeds to the system level adjacent object expanding optimization.

**System level expanding optimization**

Objective:

\[ \text{Min. } 0.8*I + 0.2*d \]

Constraints:

- Bounds on the translation magnitudes along the contact normal direction
  \[ 0.0 \leq \delta_m \leq 2.0 \]
- Minimal sum of moving magnitudes
  \[ \sum \delta_m \geq 2.0 \]
- Bounds on the position range of objects
- No overlap

Design Variables:

- Translation magnitudes of adjacent objects (engine block for this example) along the contact normal directions \( \delta \)

Optimization result:

- Moving magnitude along the contact normal between the coolant reservoir and the engine block
  \[ \delta_{\text{engine}} = 2.0 \]
After adjusting the positions of adjacent objects, the optimization goes down to component level for further volume expansion. However, the volume still cannot reach the target value. Another system level expanding optimization is evoked. For this particular case, there are total 29 iterations of system level expanding search. The final result after system level adjacent objects expanding optimization and corresponding component level volume expansion is shown in Fig. 8.11.

![Figure 8.11 System level expanding optimization and subsequent component level volume expansion result of solution A](image)

Figure 8.11 System level expanding optimization and subsequent component level volume expansion result of solution A
Now, the algorithm goes up to the system level local search.

**System level local search formulation**

**Objective:**

\[ \text{Min. } 0.8*I + 0.2*d \]

**Constraints:**

- No overlap
- Local bounds on the position range of the objects

**Design Variables:**

- Positions of objects

The optimization result is shown in Fig 8.12.

The moment of inertia and ground clearance after each search are shown in the following Table 8.3.

<table>
<thead>
<tr>
<th>Value</th>
<th>Moment of Inertia</th>
<th>Ground Clearance</th>
</tr>
</thead>
<tbody>
<tr>
<td>System level global search</td>
<td>259057000.000000</td>
<td>59.223200</td>
</tr>
<tr>
<td>System level expanding optimization and component level volume expansion</td>
<td>267728634.273267</td>
<td>53.395761</td>
</tr>
<tr>
<td>System level local search</td>
<td>266501800.493513</td>
<td>54.201331</td>
</tr>
</tbody>
</table>
Solution B

The system level global search result of solution B is shown in Fig.8.13. For
solution B, assume the ground clearance is more important, therefore the weights assigned to moment of inertia and ground clearance are chosen as 0.2 and 0.8 respectively.

The problem formulations for component level search, system level expanding optimization and system level local search are presented as follows.
Component level problem formulation

Target:

Expanding volume of the coolant reservoir to 12000000 units

Constraints:

No collision

Shape constraints of coolant reservoir

\[ x_{\text{max}} - x_{\text{min}} \leq 240 \]
\[ z_{\text{max}} - z_{\text{min}} \leq 260 \]

In this case, the target volume also cannot be reached without moving the adjacent objects. Therefore, the algorithm goes up to the system level adjacent objects expanding optimization.

System level expanding optimization

Objective:

Min. 0.2*I + 0.8*d

Constraints:

Bounds on the translation magnitudes of adjacent objects along the contact normal direction

\[ 0.0 \leq \delta_m \leq 2.0 \]

Minimal sum of moving magnitudes

\[ \sum \delta_m \geq 2.0 \]

Bounds on the position range of objects

No overlap
Design Variables:

Translation magnitudes along the contact normal directions $\delta$

Optimization result:

Moving magnitude along the contact normal between the coolant reservoir and the engine block $\delta_{engine} = 2.0$
With the positions of adjacent object adjusted, the algorithm goes down to the component level step for further volume expansion. However, the volume still cannot reach the target value. Thus another system level expanding optimization is required. For this particular case, there are total 7 iterations of the system level expanding search. The final result after system level adjacent objects expanding optimization and corresponding component level volume expansion is shown in Fig.8.14. Now, the algorithm goes up to
the system level local search.

*System level local search formulation*

Objective:

\[
\text{Min. } 0.2*I + 0.8*d
\]

Constraints:

- No overlap
- Local bounds on the position range of the objects

Design Variables:

- Positions of objects

The system level local search optimization result is shown in Fig 8.15.

The moment of inertia and ground clearance after each search are shown in Table 8.4.

**Table 8.4 Objective Values after Each Sub-search of Solution B**

<table>
<thead>
<tr>
<th>Value</th>
<th>Moment of Inertia</th>
<th>Ground Clearance</th>
</tr>
</thead>
<tbody>
<tr>
<td>System level global search</td>
<td>262435000.000000</td>
<td>50.005300</td>
</tr>
<tr>
<td>System level expanding optimization and component level volume expansion</td>
<td>267787807.561912</td>
<td>53.983431</td>
</tr>
<tr>
<td>System level local search</td>
<td>267281706.787590</td>
<td>52.498305</td>
</tr>
</tbody>
</table>
Figure 8.15 System level local search result of solution B

Solution C

The system level global search result of solution C is shown in Fig.8.16. For solution C, assume both moment of inertia and ground clearance are equally important, so the same weight of 0.5 is assigned both entities. The problem formulations for each sub-search are given as follows.
Figure 8.16 System level global search result of solution C

Component level problem formulation

Target:

Expanding volume of the coolant reservoir to 12000000 units

Constraints:

No collision

Shape constraints of coolant reservoir
Similarly to the previous two cases, the target volume cannot be reached without moving adjacent objects. Therefore, the algorithm goes up to the system level expanding optimization.

*System level expanding optimization*

**Objective:**

\[
\text{Min. } 0.5*I + 0.5*d
\]

**Constraints:**

- Bounds on the translation magnitudes along the contact normal direction
  \[
  0.0 \leq \delta_m \leq 2.0
  \]

- Minimal sum of moving magnitudes
  \[
  \sum \delta_m \geq 2.0
  \]

- Bounds on the position range of objects

- No overlap

**Design Variables:**

- Translation magnitudes along the contact normal directions \( \delta \)

**Optimization result:**

- Moving magnitude along the contact normal between the coolant reservoir and the engine block
  \[
  \delta_{\text{engine}} = 2.0
  \]
Figure 8.17 System level expanding optimization and subsequent component level volume expansion result of solution C

Once the positions of adjacent objects are adjusted through system level expanding optimization, the optimization goes down to component level for further volume expansion. Similarly, the volume still cannot reach the target value. More system level expanding optimization are performed. For this particular case, there are a total of 15 iterations of the system level expanding search. The final result after system level adjacent objects expanding optimization and corresponding component level volume
expansion is shown in Fig. 8.17. Now, the algorithm goes up to the system level local search.

**System level local search formulation**

*Objective:*

\[
\text{Min. } 0.5*I + 0.5*d
\]

*Constraints:*

- No overlap
- Local bounds on the position range of the objects

*Design Variables:*

- Positions of objects

The optimization result is shown in Fig 8.18.

The moment of inertia and ground clearance after each search are shown in the following table 8.5.

<table>
<thead>
<tr>
<th>Value</th>
<th>Moment of Inertia</th>
<th>Ground Clearance</th>
</tr>
</thead>
<tbody>
<tr>
<td>System level global search</td>
<td>260986000.000000</td>
<td>53.742300</td>
</tr>
<tr>
<td>System level expanding optimization and component level volume expansion</td>
<td>266830503.883053</td>
<td>50.720431</td>
</tr>
<tr>
<td>System level local search</td>
<td>265712382.610178</td>
<td>51.709394</td>
</tr>
</tbody>
</table>
Figure 8.18 System level local search result of solution C
Summary

This chapter presents an underbody fuel tank design and a Ford Taurus under-hood layout design with the mass-spring physical model based morphing. The results show that the mass-spring physical model based morphing successfully performs as expected even with objects of complex shape and multi-object interactions. The results also show the flexibility of this morphing method, which has no assumptions on the final shape of the morphing component. This morphing method is suitable for morphing objects with arbitrary shapes in a crowded environment such as the under-hood/underbody. The proposed bi-level approach is applied to a Ford Taurus under-hood layout design, where three cases with different designer preferences as to the importance of the objectives are illustrated.
CHAPTER 9 CONCLUSION

Overview

Packing problems have found wide applications in the engineering field. Recently, another aspect has attracted strong research attention — component shape morphing in layout design. For example, in the automotive under-hood packing, with the addition of new components, the space available under the hood continues to be reduced and overcrowded. Shape morphing is required for meeting the challenge to fit a component of sufficient size in the limited space while optimizing the overall performance objectives of the vehicle and improving design efficiency.

This work is focused on using physical based models to solve layout design problem, especially for the case where shape morphing is required for fitting components in a crowded environment. A novel packing algorithm for compact packing based on the rubber band analogy is proposed and implemented. This method solves packing problems by simulating the physical movements of a set of objects subjected to elastic forces applied by a rubber band until maximum compactness is reached. Starting from different initial configurations will lead to different optimal packing results. This elastic analogy technique can guarantee locally optimal solutions, and displays a very intuitive behavior. The search direction -- translation vectors and rotation directions are obtained in a straightforward manner from physical principles. Compared with gradient based optimization, this direct search method does not require any gradient information. This is a great advantage especially for packing problems where the gradient of objective functions is not explicit. The rubber band analogy algorithm is implemented and applied
for three-dimensional arbitrary shaped tessellated objects.

To morph components that could have arbitrary shapes in layout design, the mass-spring physical model based morphing method is proposed and implemented. It is an extension of the rubber band model. For the packing with rubber band analogy, the rubber band acted as a series of springs, bringing objects together to form a compact packing. While for the mass-spring physical model based morphing, a balloon or rubber band is inflated to occupy the desired volume. The mass points form the surface nodes, which may be subject to spring forces and pressure force. By simulating movements of those mass points with interaction with the boundary constraints in a layout, the shape of the object is therefore morphed and directly obtained. For the mesh-spring physical based layout design, one of the most important constraints is the collision detection between nodes of the deformable objects and other triangular objects. However, no available collision detection package can achieve this task. Therefore, a special collision detection procedure between deformable object and fixed shape object or enclosure is proposed, which is effective for preventing collisions. Compared with the traditional morphing methods such as parameterization based and parametric mesh based, the mass-spring physical model offers greater morphing freedom. There is no need to explicitly specify the deformations, therefore the shape morphing freedom is not confined. Compared with mesh based with purely geometric techniques, this method does not require any user interactions. Therefore it can be easily integrated into an automatic layout design process.

Real world engineering problems are often large scale and quite complex. Layout design with shape morphing belongs to this type of problem. The general bi-level
problem formulation is not sufficient for solving our special problem. Therefore, a
bi-level approach is tailored specially for the layout design with mass-spring physical
model based shape morphing. At the system level, the overall performance objectives
are optimized with respect to locations and orientations of components (deformable
objects remain at their original shapes during this phase). The system level search is
further classified into global search, local search, and adjacent objects expanding
optimization. While at the component level, deformable objects are inflated and morphed
to fit in the available space. By decomposition, for each sub-search, a simpler problem is
solved.

Besides the physical based models for packing and shape morphing, this work also
investigates the general layout design process, generic packing model, and the key issues
involved such as CAD model preprocessing and re-meshing after shape morphing, as
well as collision detection efficiency. The generic packing model includes four functional
parts: object layout, shape morphing, performance evaluation and data format translation.
The function of each part is analyzed. In this research, at the system level, the geometric
representation of components is the B-rep/CSG, while at the component level, the
geometric representation for morphing is the tessellation. Therefore, a data format
translator is developed for transferring tessellation models into solid models. Furthermore,
some packing related issues are addressed. One issue is CAD model preprocessing. To
obtain the outer shell of components with complex internal features, or obtain the packing
space, surface wrapping is investigated. For the re-meshing of morphed object, the
Delaunay triangulation is used.
Finally, the proposed methods are illustrated with two cases in vehicle layout design. One case is to fit a fuel tank in a given under-body layout with two different exhaust pipe layouts. Another case is under-hood layout design with performance objective of minimizing the moment of inertia, ground clearance and maintainability, as well as increasing volume of coolant reservoir. The examples show the great morphing capability of the mass-spring physical model based method and how it is applied in a layout design process.

In summary, this work presented the methodology of using physical based models for solving layout design with shape morphing problem, especially for the crowded packing environment.

Future Work

The work has left several issues that need further investigation in the future.

The rubber band analogy method is intuitive and very efficient in getting local optimal solutions. The solutions depend on the initial configurations. However, the packing problem is multi-modal. To extend the usage of this method, it is possible to couple it with some global search algorithm such as Genetic Algorithms, Simulated Annealing, etc. The rubber band analogy may be used as a local optimal operator to accelerate convergence.

The second issue is adopting a multi-resolution mesh during the morphing process. The mesh quality deteriorates when an object is inflated and morphed, especially when dramatic change happens. Currently, only at the last step, the object is re-meshed with Delaunay triangulations to get an optimal mesh. The finer the mesh, the more accurate the
shape of object is able to fit a contour. Therefore, it is desired to refine the mesh during the iterations.

    Currently, for the mass-spring physical model based shape morphing, only the interactions between deformable object and fixed shape object (enclosure) are studied. However, with the development of the concept of soft object packing, the interaction between two or more deformable object needs to be investigated further.

    Finally, the bi-level iteration needs further investigation to improve the design efficiency and robustness. The robustness consist both CAD aspect and optimization algorithm aspect. The current adopted approximation query package Swift++ [UNC Gamma] can achieve the required functions. However, for complex geometry, the robustness needs to be improved. The robustness of gradient based algorithm used for local search and adjacent objects expanding optimization also needs to be improved.
REFERENCES


