An Iterated Local Search Algorithm Based on Nonlinear Programming for the Irregular Strip Packing Problem

Takashi Imamichi\(^1\)  \quad \underline{Mutsunori Yagiura}^2  \quad Hiroshi Nagamochi\(^3\)

1) IBM Research - Tokyo, 2) Nagoya University, 3) Kyoto University

Reggio Emilia, 2010

Email: yagiura@nagoya-u.jp
Outline

1. The irregular strip packing problem

2. Algorithm
   - Overview of our algorithm
   - Overlap minimization by nonlinear programming
   - Swapping two polygons

3. Computational results

4. Conclusions
The Irregular Strip Packing Problem

- A rectangular container $C$ with a fixed width
- Place irregular polygons $\{P_1, \ldots, P_n\}$ without any overlap
- Minimize the length of the container

- Translations and rotations by fixed degrees.
- Applications of apparel industry, steel industry, etc.
- NP-hard
Existing Heuristics

- Burke et al. (2005)
  - Bottom-left-fill algorithm
  - Hill climbing and tabu search

- Gomes and Oliveira (2006)
  - Compaction and separation algorithms based on linear programming
  - Simulated annealing

- Egeblad et al. (2006)
  - Move a polygon into the position with the least overlap area on a line
  - Guided local search

SHAPES0/67.09%

ALBANO/87.44%

(Egeblad et al. 2006)
Compaction and Separation

Compaction
- Input: feasible layout of polygons
- Objective: translate polygons to shrink the container keeping the feasibility

Separation
- Input: infeasible layout of polygons
- Objective: translate polygons to obtain a feasible layout

- Algorithms to solve these problems: compaction/separation operators
- Heuristic algorithms are usually used
minimize the length of the container

separation constraints of polygons

\[ \mathbf{v}_{ij} \cdot \mathbf{p}_i \leq d_{ij}, \quad \forall \mathbf{p}_i \in P_i, \]
\[ \mathbf{v}_{ij} \cdot \mathbf{p}_j \geq d_{ij}, \quad \forall \mathbf{p}_j \in P_j. \]

generate the constraints of adjacent polygons

limit the size of the translations and repeat solving LPs
Specify an order of polygons and place them one by one
Find a feasible position on a line
Try several lines
Search for the best order of polygons
Basic Strategy

Repeat

- **Shrink** the length of the container
- **Relocate** polygons to minimize the total overlap
  (iterated local search)

![Diagram showing steps of the strategy: Initial solution, shrink, relocate, shrink, relocate.](image)
Iterated local search

- Incumbent solution: the best solution so far
- Repeat **perturbations** of the incumbent solution and **local searches**
Overlap Minimization

Iterated local search

- Incumbent solution: the best solution so far
- Repeat perturbations of the incumbent solution and local searches

Local search — quasi-Newton method
- Perturbation — swap two polygons and find their good positions
Pseudo Code

1. Generate an initial solution
2. While (within a time limit)
   3. **MINIMIZE OVERLAP**
   4. No overlap $\Rightarrow$ Shrink the length of the container ($r_{dec}$)
   5. Otherwise $\Rightarrow$ Expand the length of the container ($r_{inc}$)

\[ M I N I M I Z E \ O V E R L A P (x_{\text{best}}) \]

1. Until failing to update $x_{\text{best}}$ $N$ consecutive times ($N = 200$)
   2. $x_{\text{init}} := SWAPTWO POLYGONS(x_{\text{best}})$
   3. $x_{\text{lopt}} := SEPARATE(x_{\text{init}})$
   4. $x_{\text{lopt}}$ has less amount of overlap than $x_{\text{best}}$ $\Rightarrow$ $x_{\text{best}} := x_{\text{lopt}}$
Overlap Minimization Problem

- Fix the container; No rotation
- Minimize the total overlap of polygons
- Formulated as an unconstrained nonlinear program

Define the amount of overlap by using the penetration depth
- Propose a differentiable objective function
- Apply the quasi-Newton method to obtain a locally optimal solution (translate all polygons simultaneously)
Notation

Polygon

- Polygons: \( P_i \), \( 1 \leq i \leq n \)
- Translation vectors: \( x_i \), \( 1 \leq i \leq n \)
- \( P_i \) translated by \( x_i \): \( P_i \oplus x_i = \{ p + x_i \mid p \in P_i \} \)

Container

- The width of the container: \( W \)
- The length of the container: \( L \)
- Container: \( C = \{(x, y) \mid 0 \leq x \leq L, 0 \leq y \leq W\} \)
- The exterior region of \( C \): \( \overline{C} = \{(x, y) \mid (x, y) \not\in C\} \)
No-fit Polygon (NFP) (Bennell et al. 2001)

No-fit polygon: $\text{NFP}(P_i, P_j)$

- Fix $P_i$’s reference point at the origin
- The set of $x_j$ such that $P_i$ and $P_j \oplus x_j$ overlap
Lemma 1

- $x_j - x_i \in NFP(P_i, P_j) \iff P_i$ and $P_j$ overlap
- $x_j - x_i \in \partial NFP(P_i, P_j) \iff P_i$ touches $P_j$
- $x_j - x_i \notin cl(NFP(P_i, P_j)) \iff P_i$ and $P_j$ are separated
No-fit Polygon (NFP)

\[ \text{NFP}(\overline{C}, P_i) \]

- The set of \( x_j \) such that \( P_i \oplus x_i \) protrudes from \( C \)
Lemma 2

- \( x_i \in NFP(\overline{C}, P_i) \) \( \iff \) \( P_i \) protrudes from \( C \)
- \( x_i \in \partial NFP(\overline{C}, P_i) \) \( \iff \) \( P_i \) is inside \( C \) and touches \( C \)
- \( x_i \notin \text{cl}(NFP(\overline{C}, P_i)) \) \( \iff \) \( C \) includes \( P_i \)
Lemma 3

\[ \text{NFP}(P_i, P_j) = \text{int}(P_i) \oplus (-\text{int}(P_j)) \]
\[ = \{ p_i - p_j \mid p_i \in \text{int}(P_i), \ p_j \in \text{int}(P_j) \} \]

(Proof)
\[ x_j - x_i \in \text{int}(P_i) \oplus (-\text{int}(P_j)) \]
\[ \iff \exists p_i \in \text{int}(P_i), \ \exists p_j \in \text{int}(P_j), \ x_j - x_i = p_i - p_j \]
\[ \iff \exists p_i \in \text{int}(P_i), \ \exists p_j \in \text{int}(P_j), \ p_i + x_i = p_j + x_j \]
\[ \iff \exists q_i \in \text{int}(P_i \oplus x_i), \ \exists q_j \in \text{int}(P_j \oplus x_j), \ q_i = q_j \]
\[ \iff P_i \oplus x_i \text{ and } P_j \oplus x_j \text{ overlap} \]
Penetration depth: $\delta(P_i, P_j)$

The minimum translational distance to separate $P_i$ and $P_j$

Lemma

$\delta(P_i \oplus x_i, P_j \oplus x_j)$ is equal to the minimum distance from $x_j - x_i$ to the boundary of $\text{NFP}(P_i, P_j)$.

\[ \text{NFP}(P_i, P_j) \oplus x_i \]
Overlap Minimization Problem

Input: polygons $P_1, \ldots, P_n$; a fixed container $C$

minimize \[ \sum_{1 \leq i < j \leq n} f_{ij}(\mathbf{x}) + \sum_{1 \leq i \leq n} g_i(\mathbf{x}) \]
subject to \[ \mathbf{x} \in \mathbb{R}^{2n} \]

- $m(>0)$: a parameter
- $\mathbf{x} = (x_1, x_2, \ldots, x_n)$: $x_i$ is the position of $P_i$'s reference point.
- $f_{ij}(\mathbf{x}) = \delta(P_i \oplus x_i, P_j \oplus x_j)^m$: the amount of overlap of $P_i$ and $P_j$.
- $g_i(\mathbf{x}) = \delta(C, P_i \oplus x_i)^m$: the amount of protrusion of $P_i$ from $C$.
Overlap Minimization Problem

Input: polygons $P_1, \ldots, P_n$; a fixed container $C$

\[
\begin{align*}
\text{minimize} & \quad \sum_{1 \leq i < j \leq n} f_{ij}(\mathbf{x}) + \sum_{1 \leq i \leq n} g_i(\mathbf{x}) \\
\text{subject to} & \quad \mathbf{x} \in \mathbb{R}^{2n}
\end{align*}
\]

- $m(>0)$: a parameter
- $\mathbf{x} = (x_1, x_2, \ldots, x_n)$: $x_i$ is the position of $P_i$’s reference point.
- $f_{ij}(\mathbf{x}) = \delta(P_i \oplus x_i, P_j \oplus x_j)^m$: the amount of overlap of $P_i$ and $P_j$.
- $g_i(\mathbf{x}) = \delta(C, P_i \oplus x_i)^m$: the amount of protrusion of $P_i$ from $C$.

Invoke the quasi-Newton method to compute a locally optimal solution by using the current layout as the initial solution.
Penetration Penalty $f_{ij}$ and $\nabla f_{ij}$ (Case 1)

Case 1: $P_i$ and $P_j$ are tangent or separated

$$f_{ij}(\mathbf{x}) = \delta(P_i \oplus \mathbf{x}_i, P_j \oplus \mathbf{x}_j)^m = 0$$
Case 1: $P_i$ and $P_j$ are tangent or separated

\[ f_{ij}(x) = \delta(P_i \oplus x_i, P_j \oplus x_j)^m = 0 \]

\[ \nabla_k f_{ij}(x) = 0, \quad k \in \{1, \ldots, n\} \]

- $x = (x_1, \ldots, x_n); \quad x_k = (x_{k1}, x_{k2}), \quad 1 \leq k \leq n$
- $\nabla_k = (\partial/\partial x_{k1}, \partial/\partial x_{k2}), \quad 1 \leq k \leq n$
- $\nabla f_{ij}(x) = (\nabla_1 f_{ij}(x), \ldots, \nabla_n f_{ij}(x)), \quad 1 \leq k \leq n$

\[ P_j \]
\[ x_j \]
\[ P_i \]
\[ x_i \]
\[ NFP(P_i, P_j) \]
\[ x_j - x_i \]
Penetration Penalty $f_{ij}$ and $\nabla f_{ij}$ (Case 2)

**Case 2:** $x_j - x_i$ has a unique nearest point on $\partial \text{NFP}(P_i, P_j)$

$$f_{ij}(x) = \delta(P_i \oplus x_i, P_j \oplus x_j)^m = \|z\|^m$$

- $v = x_j - x_i$; $w$ = the nearest point from $v$ on $\partial \text{NFP}(P_i, P_j)$
- $z = w - v$ and the contour line of $f_{ij}(x)$ are orthogonal
Penetration Penalty $f_{ij}$ and $\nabla f_{ij}$ (Case 2)

Case 2: $x_j - x_i$ has a unique nearest point on $\partial NFP(P_i, P_j)$

\[
\begin{align*}
    f_{ij}(x) &= \delta(P_i \oplus x_i, P_j \oplus x_j)^m = \|z\|^m \\
    \nabla_i f_{ij}(x) &= -\nabla_j f_{ij}(x) = m\|z\|^{m-2}z \\
    \nabla_k f_{ij}(x) &= 0, \quad k \in \{1, \ldots, n\} \setminus \{i, j\}
\end{align*}
\]

- $v = x_j - x_i$; $w =$ the nearest point from $v$ on $\partial NFP(P_i, P_j)$
- $z = w - v$ and the contour line of $f_{ij}(x)$ are orthogonal
Penetration Penalty $f_{ij}$ and $\nabla f_{ij}$ (Case 3)

**Case 3:** $x_j - x_i$ has two or more nearest points on $\partial \text{NFP}(P_i, P_j)$

- Choose one of the nearest points and compute $\nabla f_{ij}$ as in the Case 2

$$v = x_j - x_i$$

- This case happens only if $v$ is on the medial axis of the NFP
Protrusion Penalty $g_i$ and $\nabla g_i$

- $x_i$ has a unique nearest point on $NFP(\overline{C}, P_i)$
- Compute $\nabla g_i$ as in the Case 2 of $\nabla f_{ij}$

- $w =$ the nearest point from $x_i$ on $NFP(\overline{C}, P_i)$
Let $m = 2$ in our experiments

- $\nabla f_{ij}$ is continuous on $\partial \text{NFP}(P_i, P_j)$
- The equation $\nabla f_{ij}$ of Case 2 is simple

$$(m \|z\|^{m-2} z \to 2z)$$
Separation Algorithm: SEPARATE

- Given an initial layout, SEPARATE applies the quasi-Newton method to the overlap minimization problem.
- Return a locally optimal layout.
- We adopt L-BFGS for the quasi-Newton method solver.

Demonstration of SEPARATE

- Fix the container.
- Place polygons randomly and apply SEPARATE.
- Move a blue polygon after SEPARATE ends.
- Apply SEPARATE again.
Swapping Two Polygons

**SWAPTWOPOLYGONS**

- Choose two polygons $P_i$ and $P_j$ randomly.
- Fix the other polygons.
- Move $P_i$ and $P_j$ to positions where overlap is small. (try all orientations)
Choose two polygons $P_i$ and $P_j$ randomly.
Fix the other polygons.
Move $P_i$ and $P_j$ to positions where overlap is small.
(try all orientations)
Swapping Two Polygons

**SWAP TWO POLYGONS**

- Choose two polygons $P_i$ and $P_j$ randomly.
- Fix the other polygons.
- Move $P_i$ and $P_j$ to positions where overlap is small. (try all orientations)
Choose two polygons $P_i$ and $P_j$ randomly.

Fix the other polygons.

Move $P_i$ and $P_j$ to positions where overlap is small. (try all orientations)
Move a polygon at a position with small overlap

FindBestPosition($P_i$)

- Place $\text{NFP}(P_k, P_i) \ (k \in \{1, \ldots, n\} \setminus \{i\})$ and $\text{NFP}(\overline{C}, P_i)$.
- Let $Q$ be a set of $K$ vertices chosen randomly from the NFPs.
- Find the minimum overlap position $t^*$ from $Q$.
- Try all orientations of $P_i$ and find the best rotation $r^*_i$.

($K = 800$ in our experiments)
Initial Solution

- Generate a sequence of polygons in descending order of area.
- Place polygons in the sequence one by one by FINDBESTPOSITION.
Pseudo Code

1. Generate an initial solution
2. While (within a time limit)
   3. **MINIMIZE OVERLAP**
      4. No overlap ⇒ Shrink the length of the container ($r_{\text{dec}}$)
      5. Otherwise ⇒ Expand the length of the container ($r_{\text{inc}}$)

**MINIMIZE OVERLAP($x_{\text{best}}$)**

1. Until failing to update $x_{\text{best}}$ $N$ consecutive times ($N = 200$)
2. $x_{\text{init}} := \text{SWAP TWO POLYGONS}(x_{\text{best}})$
3. $x_{\text{lopt}} := \text{SEPARATE}(x_{\text{init}})$
4. $x_{\text{lopt}}$ has less amount of overlap than $x_{\text{best}}$ ⇒ $x_{\text{best}} := x_{\text{lopt}}$
### Environment

<table>
<thead>
<tr>
<th>PC</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CPU:</strong> Xeon 2.8GHz (NetBurst)</td>
</tr>
<tr>
<td><strong>Memory:</strong> 1GB</td>
</tr>
<tr>
<td><strong>OS:</strong> FreeBSD 5.3-Release</td>
</tr>
<tr>
<td><strong>Compiler:</strong> GCC 4.02 (C++)</td>
</tr>
</tbody>
</table>

### Parameters

- $r_{\text{dec}} = 0.04$, $r_{\text{inc}} = 0.01$, $N = 200$
- Shrink the right side of the container
### Information of Benchmark Instances

<table>
<thead>
<tr>
<th>Instance</th>
<th>NDP</th>
<th>TNP</th>
<th>ANV</th>
<th>Orientations (°)</th>
<th>Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALBANO</td>
<td>8</td>
<td>24</td>
<td>7.25</td>
<td>180</td>
<td>4900</td>
</tr>
<tr>
<td>DAGLI</td>
<td>10</td>
<td>30</td>
<td>6.30</td>
<td>180</td>
<td>60</td>
</tr>
<tr>
<td>DIGHE1</td>
<td>16</td>
<td>16</td>
<td>3.87</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td>DIGHE2</td>
<td>10</td>
<td>10</td>
<td>4.70</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td>FU</td>
<td>12</td>
<td>12</td>
<td>3.58</td>
<td>90</td>
<td>38</td>
</tr>
<tr>
<td>JAKOBS1</td>
<td>25</td>
<td>25</td>
<td>5.60</td>
<td>90</td>
<td>40</td>
</tr>
<tr>
<td>JAKOBS2</td>
<td>25</td>
<td>25</td>
<td>5.36</td>
<td>90</td>
<td>70</td>
</tr>
<tr>
<td>MAO</td>
<td>9</td>
<td>20</td>
<td>9.22</td>
<td>90</td>
<td>2550</td>
</tr>
<tr>
<td>MARQUES</td>
<td>8</td>
<td>24</td>
<td>7.37</td>
<td>90</td>
<td>104</td>
</tr>
<tr>
<td>SHAPES0</td>
<td>4</td>
<td>43</td>
<td>8.75</td>
<td>0</td>
<td>40</td>
</tr>
<tr>
<td>SHAPES1</td>
<td>4</td>
<td>43</td>
<td>8.75</td>
<td>180</td>
<td>40</td>
</tr>
<tr>
<td>SHAPES2</td>
<td>7</td>
<td>28</td>
<td>6.29</td>
<td>180</td>
<td>15</td>
</tr>
<tr>
<td>SHIRTS</td>
<td>8</td>
<td>99</td>
<td>6.63</td>
<td>180</td>
<td>40</td>
</tr>
<tr>
<td>SWIM</td>
<td>10</td>
<td>48</td>
<td>21.90</td>
<td>180</td>
<td>5752</td>
</tr>
<tr>
<td>TROUSERS</td>
<td>17</td>
<td>64</td>
<td>5.06</td>
<td>180</td>
<td>79</td>
</tr>
</tbody>
</table>

- **NDP**: the number of different polygons
- **TNP**: the total number of polygons
- **ANV**: the average number of vertices
## Comparison of Computational Times (sec)

<table>
<thead>
<tr>
<th>Instance</th>
<th>Ours 2.8 GHz 10 runs</th>
<th>Gomes et al. 2.4 GHz 20 runs</th>
<th>Burke et al. 2.0 GHz 40 runs</th>
<th>Egeblad et al. 3.0 GHz 20 runs</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALBANO</td>
<td>1200</td>
<td>2257</td>
<td>–</td>
<td>600</td>
</tr>
<tr>
<td>DAGLI</td>
<td>1200</td>
<td>5110</td>
<td>188.80</td>
<td>600</td>
</tr>
<tr>
<td>DIGHE1</td>
<td>600</td>
<td>83</td>
<td>–</td>
<td>600</td>
</tr>
<tr>
<td>DIGHE2</td>
<td>600</td>
<td>22</td>
<td>–</td>
<td>600</td>
</tr>
<tr>
<td>FU</td>
<td>600</td>
<td>296</td>
<td>20.78</td>
<td>600</td>
</tr>
<tr>
<td>JAKOBS1</td>
<td>600</td>
<td>332</td>
<td>43.49</td>
<td>600</td>
</tr>
<tr>
<td>JAKOBS2</td>
<td>600</td>
<td>454</td>
<td>81.41</td>
<td>600</td>
</tr>
<tr>
<td>MAO</td>
<td>1200</td>
<td>8245</td>
<td>29.74</td>
<td>600</td>
</tr>
<tr>
<td>MARQUES</td>
<td>1200</td>
<td>7507</td>
<td>4.87</td>
<td>600</td>
</tr>
<tr>
<td>SHAPES0</td>
<td>1200</td>
<td>3914</td>
<td>21.33</td>
<td>600</td>
</tr>
<tr>
<td>SHAPES1</td>
<td>1200</td>
<td>10314</td>
<td>2.19</td>
<td>600</td>
</tr>
<tr>
<td>SHAPES2</td>
<td>1200</td>
<td>2136</td>
<td>21.00</td>
<td>600</td>
</tr>
<tr>
<td>SHIRTS</td>
<td>1200</td>
<td>10391</td>
<td>58.36</td>
<td>600</td>
</tr>
<tr>
<td>SWIM</td>
<td>1200</td>
<td>6937</td>
<td>607.37</td>
<td>600</td>
</tr>
<tr>
<td>TROUSERS</td>
<td>1200</td>
<td>8588</td>
<td>756.15</td>
<td>600</td>
</tr>
</tbody>
</table>
Comparison of Densities (%)

<table>
<thead>
<tr>
<th>Instance</th>
<th>Ours Average</th>
<th>Best</th>
<th>Gomes et al. Best</th>
<th>Burke et al. Best</th>
<th>Egeblad et al. Best</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALBANO</td>
<td>87.14</td>
<td>88.16</td>
<td>87.43</td>
<td>–</td>
<td>87.44</td>
</tr>
<tr>
<td>DAGLI</td>
<td>85.80</td>
<td>87.40</td>
<td>87.15</td>
<td>83.7</td>
<td>85.98</td>
</tr>
<tr>
<td>DIGHE1</td>
<td>90.49</td>
<td>99.89</td>
<td>100.00</td>
<td>–</td>
<td>99.86</td>
</tr>
<tr>
<td>DIGHE2</td>
<td>84.21</td>
<td>99.99</td>
<td>100.00</td>
<td>–</td>
<td>99.95</td>
</tr>
<tr>
<td>FU</td>
<td>87.57</td>
<td>90.67</td>
<td>90.96</td>
<td>86.9</td>
<td>91.84</td>
</tr>
<tr>
<td>JAKOBS1</td>
<td>84.78</td>
<td>86.89</td>
<td>78.89</td>
<td>82.6</td>
<td>89.07</td>
</tr>
<tr>
<td>JAKOBS2</td>
<td>80.50</td>
<td>82.51</td>
<td>77.28</td>
<td>74.8</td>
<td>80.41</td>
</tr>
<tr>
<td>MAO</td>
<td>81.31</td>
<td>83.44</td>
<td>82.54</td>
<td>79.5</td>
<td>85.15</td>
</tr>
<tr>
<td>MARQUES</td>
<td>86.81</td>
<td>89.03</td>
<td>88.14</td>
<td>86.5</td>
<td>89.17</td>
</tr>
<tr>
<td>SHAPES0</td>
<td>66.49</td>
<td>68.44</td>
<td>66.50</td>
<td>60.5</td>
<td>67.09</td>
</tr>
<tr>
<td>SHAPES1</td>
<td>72.83</td>
<td>73.84</td>
<td>71.25</td>
<td>66.5</td>
<td>73.84</td>
</tr>
<tr>
<td>SHAPES2</td>
<td>81.72</td>
<td>84.25</td>
<td>83.60</td>
<td>77.7</td>
<td>81.21</td>
</tr>
<tr>
<td>SHIRTS</td>
<td>88.12</td>
<td>88.78</td>
<td>86.79</td>
<td>84.6</td>
<td>86.33</td>
</tr>
<tr>
<td>SWIM</td>
<td>74.62</td>
<td>75.29</td>
<td>74.37</td>
<td>68.4</td>
<td>71.53</td>
</tr>
<tr>
<td>TROUSERS</td>
<td>88.69</td>
<td>89.79</td>
<td>89.96</td>
<td>88.5</td>
<td>89.84</td>
</tr>
</tbody>
</table>
Layouts

DAGLI / 87.40%

JAKOBS2 / 82.51%

ALBANO / 88.16%

SWIM / 75.29%
Layouts

SHAPES0 / 68.44%

SHAPES1 / 73.84%

SHAPES2 / 84.25%

SHIRTS / 88.78%
Original Instance

Japan $\times 10$

- $180^\circ$ rotations; avg. # vertices: 135
- Time limit: 1 hour; #runs: 5
- Density: 62.83%
Scandinavia $\times 10$

- $180^\circ$ rotations; avg. # vertices: 124.6
- Time limit: 3000 seconds; #runs: 2
- Density: 59.6%

$x10$
Original Instance

**Italy × 20**

- 90° rotations; # vertices: 148
- Time limit: 3000 seconds; # runs: 5
- Density: 53.95%
Conclusions

- Overlap minimization based on nonlinear programming
- Iterated local search algorithm incorporating nonlinear programming and the swapping operation
- We updated several best known results

Future work

- Circles or other shapes in 2D
- 3D objects