Solving Multiple Layer Containment Problems Using Iterative Methods

Nuno Marques, Pedro Capela
MIND, Software Multimédia e Industrial, S.A.,
Avenida Duque d'Ávila 23, 1000-138 Lisboa, Portugal
{nrm,pmc}@mind.pt

João Bernardo
Instituto Superior Técnico,
Avenida Rovisco Pais, 1049-001 Lisboa, Portugal
jcb@inesc.pt

ABSTRACT

The footwear industry's need for an automatic containment algorithm is becoming increasingly important within the manufacturing process. Irregular containers, such as hides, have many different quality regions and holes that must be taken into account when containment is done because they represent an important cost. Automatic containment processes should be aware of these factors and still perform in practical time. We present an iterative containment algorithm that uses Minkowski operators and can be applicable to such containment problems. Although the iterative solution is not the optimal one, it can reach a solution in practical running times and it can get results that approximate the human made containment process.

Keywords: Containment Problems, Minkowski Operators, Evaluators.

INTRODUCTION

Solutions for containment problems can be viewed as an automatic way of placing a set of shapes into another shape, called the container. The aim is to find good solutions, in the sense that they have to be competitive with the human made process and still spend approximately the same time as humans do. The quality - in terms of efficiency - of a solution can be measured in multiple ways: we can find the minimal container into which a given collection of shapes fit, or find the maximal collection of shapes which fit into a given container. We will address only one subset of the containment problem universe – the two dimension rotational containment problems.
In this paper, containment processes will be solved through an iterative method, that is, we will select only one shape from the set of all shapes to place, add it to the container and proceed to the next shape until the set of shapes is empty. Additional difficulties are found when we need to use multiple containers at the same time and/or when the containers have holes and quality layer differentiation. In our previous work, we have defined the use of optimal A* search trees to solve single layer, translational 2D containment problems. We will now formally define translational and rotational configurations, single and multiple layer configurations, single and multiple container configurations, and solve containment problems using only one framework for all types of configurations.

The first section will give formal description of 2D translational and rotational containment problems, using one or multiple containers and when shapes and/or containers have multiple layer quality regions. Then we will describe how positions are found and evaluated within the container, through the use of containment strategies and evaluators. A specific strategy that selects only valid contact positions within the container's boundary is formally defined and we will prove its correctness. We will also present an evaluator that can be computed recursively, which is based on the Minkowski difference operator. The algorithm will then be presented, separating the pre-processing step from the iteration steps. Finally some examples are shown, conclusions are drawn and we give an overview of our future research and development.

RELATED WORK

Related research about containment algorithms found in literature [Milenkovic97a] can be subdivided into four main groups:

- The Physics approach, which applies classical physics theory by adding potential energy to the shapes to place, that can be viewed as physical elements. However, solving these large equations takes a large amount of time, what makes it useless in containment applications for the footwear industry;

- The Computational Geometry approach, which latest development in multi-polygon rotational case [Cavalier96a] can place convex *m*-gon *P* into convex *n*-gon container *Q* by solving O(m^4n^4) linear programs;

- The Operational Research approach, which can lead to practical results, but tend to become more complex when rotations are part of the equation;

- The Meta-Heuristic approach, which can obtain interesting results if the selected heuristic is appropriate, but fail to be applicable when the input space increases substantially.

Heuristic search methods are independent from the geometric description of each set so present a significant advantage when we need to have tight running times bounds. Geometric algorithms tend to be more independent and capable of understanding specific problem irregularities. We will try to solve containment problems for irregular containers using an iterative algorithm, which simulate human made containment methods. This approach can lead us to results that stand below 6-8%, comparing with the efficiency of a human made process and that can reach running times that are acceptable for the footwear industry.

TRANSLATIONAL CONTAINMENT PROBLEM

The main goal of a translational containment algorithm is to find points $v_1,...,v_n \in \Re^2$ that translates each shape $P_1,...,P_n$ into the container *C*. Let $\Omega_n \subseteq 2^{\Re^2}$ be the finite set of size *n* containing all shapes to be placed. A *translational configuration* $z\ (\Omega_n)$ is a *(n+1)* tuple

$$z\ (\Omega_n) = \langle (v_0, P_0),...,(v_n, P_n) \rangle \tag{1}$$

where

1. $P_1,...,P_n \in \Omega_n$,
2. $v_1,...,v_n \in \Re^2$,
3. $P_0 = \overline{C}$, $v_0 = (0,0)$.

A translational configuration is *valid* if and only if the following condition is satisfied:

$$v_j - v_i \in \overline{P_i \oplus (-P_j)} \tag{2}$$

where $v_i, v_j \in \Re^2$ are translations of P_i and P_j [Daniels95a]. The previous condition just says that for each valid configuration we have *n* points $v_1,...,v_n \in \Re^2$ so that $P_1 + v_1,...,P_n + v_n$ fits into the container *C*.

ROTATING SETS

By adding a new degree of freedom we have to extend the previous concepts. Let $A \subseteq \Re^2$ and $a \in [0..2p[$. The *rotation* of a point $v \in \Re^2$ by a, $v\, fa$, is

$$v\, fa = (v_x \cos(a) - v_y \sin(a), v_x \sin(a) + v_y \cos(a)) \tag{3}$$

The *rotation* of a set *A* by a , $A\, fa$, is

$$A\, fa = \{ v\, fa : v \in A \} \tag{4}$$

The following properties are valid for the rotation of sets,

$$A\, f(a+p) = (-A)\, fa = -(A\, fa) \tag{5}$$

$$A\, f(a-b)\, fb = A\, fa \tag{6}$$

$$(A \oplus B) fa = (A fa) \oplus (B fa) \qquad (7)$$

Let's also introduce the *circular set,*

$$B(r) = \{(r\cos(b), r\sin(b)) : b \in [0,2p[\}, r \in \Re^{+} \qquad (8)$$

so that

$$B(r) = B(r) fa \qquad (9)$$

and by using Eq. (7) and Eq. (9)

$$(A fa) \oplus B(r) = (A \oplus B(r)) fa \qquad (10)$$

ROTATIONAL CONTAINMENT PROBLEM

We can now use rotated sets and extend the definition of translational containment. Let $\Gamma \in 2^{[0,2p[}$. *A rotational configuration* $z(\Omega_n)$ is a *(n+1)* tuple

$$z(\Omega_n) = \langle (t_0, P_0)\ldots,(t_n, P_n)\rangle \qquad (11)$$

where

1. $t_0 = (0,0,0)$,
2. $t_1,\ldots,t_n \in (\Re^2 \times \Gamma)$,
3. for all $t_i = (v_i, a_i), v_i \in \Re, a_i \in \Gamma$, $z(\Omega_n) = \langle (v_0, P_0 fa_0),\ldots,(v_n, P_n fa_n)\rangle$ is a translational configuration.

From the previous definition, a rotational configuration is *valid* if and only if the following condition is satisfied:

$$v_j - v_i \in \overline{P_i fa_i \oplus (-(P_j fa_j))} \qquad (12)$$

or, using Eq. (5)

$$v_j - v_i \in \overline{P_i fa_i \oplus (P_j f(a_j + p))} \qquad (13)$$

where $v_i, v_j \in \Re^2$ are translations for $P_i fa_i$ and $P_j fa_j$.

USING MULTIPLE CONTAINERS

Automatic containment using multiple containers can be achieved by applying the previous definitions.

If we aim to place *n* shapes $P_1,\ldots,P_n$ into *m* containers $C_1,\ldots,C_m$, we can apply the definition of translational or rotational configuration by building a composed container. Therefore we claim that the container and the shapes to be placed become

$$C' = \bigcup_{j=1..m} C_j \text{ and } P_i' = P_i, \forall i = 1..n \qquad (14)$$

VISIBILITY DECOMPOSITION

When we need to take into account holes within the container and when we have to assign quality differentiation and equivalence between shapes to place and containers, visibility decomposition must be performed on shapes and containers [Bernardo95a].
A *n-layer* set *A* is a finite union of *n* sets

$$A = \bigcup_{i=1}^{n} A^i$$

where for *i,j=1..n*

1. $A^i \subseteq \Re^2$,
2. $A^i \subseteq A$,
3. $A^i \cap A^j = \emptyset, i \neq j$ (15)

If a set *A* is *n-layer* then we say that $A \in L(n)$. The container and the shapes to be placed P_i can be written as *L(q)* sets

$$P_i = \bigcup_{l=1}^{q} P_i^l \qquad (16)$$

where $P_i^l \subseteq \Re^2$ is the subset of P_i visible at quality layer *l, l=1..q*. We will say that the quality of the layer is better as its index increases.

MULTIPLE LAYER CONFIGURATION

In this scenario, a configuration must be aware of inclusion or equivalence of quality layers of shapes to place and containers. For example, an *L(2)*-shape may be enclosed into an *L(*^{3}*2)*-container. Therefore a *multiple layer configuration* is a *(n+1)* tuple

$$z(\Omega_n) = \langle (v_0, P_0),\ldots,(v_n, P_n)\rangle \qquad (17)$$

which is valid if and only if the following conditions are always true:

$$v_j - v_i \in \overline{(\bigcup_{l=1..q} P_i^l) \oplus (-\bigcup_{l=1..q} P_j^l)}, \text{if } i \neq 0,\ j \neq 0 \qquad (18)$$

$$v_i \in \bigcup_{s=1..q} \overline{(\bigcup_{l=s..q} P_0^l) \oplus \left(-P_i^s\right)}, \text{if } i \neq 0 \qquad (19)$$

In practical terms, the former condition expresses the rules for visibility decomposition of multiple layer configurations.

DEFINING A STATE

Up to now we have formally defined how to understand the containment process using two degrees of freedom (2D translation) or three degrees of freedom (2D translation plus rotation) for the shapes, using single or multiple layer configurations.
Lets proceed by defining intermediate steps for the containment process. We shall call these steps states. A *state* is

$$State(\Omega_d, \Phi_k) = \langle z\ (\Omega_d), \Phi_k \rangle \qquad (20)$$

where $z\ (\Omega_d)$ is a valid configuration of size *d* and Φ_k is a collection of *k* shapes to be placed. The initial state

$$State(\Omega_0, \Phi_n) = \langle z\ (\Omega_0), \Phi_n \rangle \qquad (21)$$

where $\Omega_0 = \{\overline{C}\}$ and $\Phi_n = \{P_1, ..., P_n\}$ just sets the initial configuration, where only the container is part of it. When using A. I. methods [Marques98a] the state will be part of the tree node and should be expanded by using a set of operators. Using an iterative method, the node expansion remains as a valid concept. The main difference is that now we choose only one node and keep expanding without backtracking.

Given a shape, a state and nesting strategy, the nesting operator returns a new state. Formally the nesting operator is

$$Place(State(\Omega_n, \Phi_k), Q, t) = State(\Omega_{n+1}, \Phi_{k-1})$$

where

1. $Q \in \Phi_k$,
2. $\Omega_{n+1} = \Omega_n \cup \{Q\}$,
3. $\Phi_{k-1} = \Phi_k \setminus \{Q\}$. (22)

The new state is

$$State(\Omega_{n+1}, \Phi_{k-1}) = \langle z\ (\Omega_{n+1}), \Phi_{k-1} \rangle$$

where

$$\begin{cases} z_{\ i}(\Omega_{n+1}) = z_{\ i}(\Omega_n), \forall_{i=0..n} \\ z_{\ n+1}(\Omega_{n+1}) = (t, Q) \end{cases} \qquad (23)$$

for $t \in Strg(z\ (\Omega_n), Q)$, where *Strg* is any nesting strategy.

The *updated container* is the result of placing all shapes of a configuration into the container. For $z\ (\Omega_n) = \langle (t_0, P_0) ..., (t_n, P_n) \rangle$ we have

$$C^{(n)} = \bigcap_{i=0}^{n} \overline{(P_i + t_i)} \qquad (24)$$

Lets take a closer look on how to select strategies and evaluate the quality (in terms of containment) of the intermediate solution.

STRATEGIES

A *containment strategy* is an operator that selects valid positions for a given state and a shape to be placed.
Let $z\ (\Omega_n)$ be a configuration and $Q \subseteq \Re^2$. A containment strategy is

$$Strg(z\ (\Omega_n), Q) = T \qquad (25)$$

where $T \subseteq C^{(n)} \Theta (-Q)$. For example, if we want to select the point that is the most to the left of the container for each state, we would have

$$StrgLeftMost(z\ (\Omega_n), Q) = \{v\}$$

where $v \in T$ and

$$\neg \exists v' \in T : v'_x < v_x . \qquad (26)$$

It is important to note that any strategy selects only positions that lead to valid configurations because, by using the fundamental property of Θ [Marques98b]

$$t \in T \Rightarrow t \in C^{(n)}\Theta(-Q) \Rightarrow (Q+t) \cap C^{(n)} = \varnothing \qquad (27)$$

meaning that if we translate Q by t we don't intersect the updated container of the configuration.

EVALUATORS

The evaluator should measure the associated cost of placing a shape into the container. Therefore, a *containment evaluator* is

$$Eval(z\ (\Omega_n), Q, t) = x \qquad (28)$$

where $x \in \Re$ can take any value. Lets say that the lower the evaluation is, the seemingly better the containment is. For example, if we want to estimate the useful free area of the container for the shapes yet to be placed, we use the Opening evaluator [Marques98b]

$$EvalOpening(z\ (\Omega_d), Q, t) = \sum_{i=d}^{N} (\, d(z\ (\Omega_d), Q+t)) \qquad (29)$$

where $z\ (\Omega_d)$ belongs to $State(\Omega_d, \Phi_k)$ and $Q \in \Phi_k$. The d function gives us information about the possibility of placing each of the remaining shapes. If such nesting is possible, a weighted shape's area is returned. The cost rises if the amount of inaccessible regions increases. If not a maximum cost is returned. $d(z\ (\Omega_d), Q)$ is

$$\begin{cases} \dfrac{(\Delta(C^{(d)}) - \Delta(C^{(d)} oQ)) \cdot \Delta(Q)}{\Delta(C^{(d)})}, & \text{if } C^{(d)}\Theta(-Q) \neq \varnothing \\ \Delta(C^{(0)}), & \text{if } C^{(d)}\Theta(-Q) = \varnothing \end{cases}$$

where $(t_0, P_0), \ldots, (t_d, P_d) \in z\ (\Omega_d)$. (30)

PRE-PROCESSING INFORMATION

We aim to apply an algorithm that can select the seemingly best position and rotation of one of the shapes to place, put it inside the container, and proceed to the next iteration until it can no longer place any more shapes or when the set of shapes to place becomes empty. Please note that the following considerations will work on single or multiple layer configurations. The first step of the containment algorithm is to assemble information about the shapes and get the initial evaluation for all of them. Lets build the collection of all valid positions for all of the rotated sets of shapes to place, that is, if

1. $Q \subseteq \Re^2$,
2. $\Gamma \in 2^{[0,2p[}$,
3. $a \in \Gamma$,
4. $K(z\ (\Omega_n), Q, a) = C^{(n)}\Theta(Q\, f(a+p))$

then

$$K(z\ (\Omega_n), Q, a) \in \Im \qquad (31)$$

$\Im$ contain all the information on the valid positions for all shapes to be placed into the container at the beginning. We can make a first restriction by selecting only the points that belong to the boundary. Once we have gathered all the information we will generate the initial evaluation for all of the points. Let's build our containment strategy for a given $State(\Omega_d, \Phi_k)$ using the previous result.

$$StrgMk(z\ (\Omega_n), Q\, fa) = fr(K(z\ (\Omega_n), Q, a)) \qquad (32)$$

where $Q \in \Phi_k$, $a \in \Gamma$ and *fr(A)* denotes the topological border of the set *A*. The right choice of the evaluator will, obviously, determine the quality and efficiency of the final solution. One good evaluator could be the *EvalOpening*, which is known to be optimal [Marques98a]. However its computational costs are very high when we need to place several different shapes (>100) in practical time. Therefore we will try to approximate the *EvalOpening* by using a circle to substitute the Opening operator. We have

$$EvalCircle(z\ (\Omega_n), Q, t) = \Delta((C^{(n)} \cap (Q+t))\Theta B(r)) \qquad (33)$$

where *B(r)* is the circle of radius *r*.

Therefore will build the containment evaluator for a given state $State(\Omega_d, \Phi_k)$ by applying the circle evaluator

$$EvalMk(z\ (\Omega_n), Q\, fa, t) = \Delta(x(z\ (\Omega_n), Q, a, t)) \qquad (34)$$

where $Q \in \Phi_k$, $a \in \Gamma$, $t \in StrgMk(z\ (\Omega_n), Q\, fa)$ and

$$x(z\ (\Omega_n), Q, a, t) = (C^{(n)} \cap \overline{(Q\,fa + t)})\Theta B(r) \qquad (35)$$

using the invariance of translation over the Minkowski difference and the distributive property of the Minkowski difference [Marques98b], we have

$$x(z\ (\Omega_n), Q, a, t) = (C^{(n)}\Theta B(r)) \cap ((\overline{(Q\,fa) \oplus B(r))} + t) \qquad (36)$$

and using Eq.(10),

$$x(z\ (\Omega_n), Q, a, t) = (C^{(n)}\Theta B(r)) \cap (\overline{(Q \oplus B(r))\,fa} + t) \qquad (37)$$

This means that we only have to do the computation for the first operand once, because it is independent from the variables *Q*, *a* and *t*. Note that the second operand can be pre-processed because it depends only on the shapes to be placed.

ITERATING

Having collected the initial information of the containment, we just have to select a shape and a position, add it into the container and proceed. Selecting a shape and a position is trivial. All we have to do is to find

1. $S \in \Phi_k$
2. $b \in \Gamma$ for $\Gamma \in 2^{[0,2p[}$
3. $s \in StrgMk(z\ (\Omega_n), S\,fb)$

such that for all shapes to place $Q \in \Phi_k$, angles $a \in \Gamma$ and positions $t \in StrgMk(z\ (\Omega_n), Q\,fa)$

$$EvalMk(z\ (\Omega_n), S, b, s) \le EvalMk(z\ (\Omega_n), Q, a, t) \qquad (38)$$

Now that we have selected one shape, we should place it inside the container and update the information for the *K*'s and x's. Updating the state is just a matter of applying the *Place* operator - see Eq.(22). The new state is

$$State(\Omega_{n+1}, \Phi_{k-1}) \leftarrow Place(State(\Omega_n, \Phi_k), S\,fb, s) \qquad (39)$$

In the case of the *K*'s, by using (Eq.37), we have

$$K(z\ (\Omega_{n+1}), Q, a) = C^{(n+1)}\Theta(Q\,f(a+p)). \qquad (40)$$

By expanding the container we have

$$K(z\ (\Omega_{n+1}), Q, a) = (C^{(n)} \cap \overline{(S\,fb + s)})\Theta(Q\,f(a+p)) \qquad (41)$$

and by distributing intersection over the Minkowski difference [Marques98b] and rearranging the operands, we have

$$\begin{aligned} K(z\ (\Omega_{n+1}), Q, a) = (C^{(n)}\Theta(Q\,f(a+p)) \cap \\ \overline{((S\,fb) \oplus (Q\,f(a+p))} + s) \end{aligned} \qquad (42)$$

which can be rewritten as

$$\begin{aligned} K(z\ (\Omega_{n+1}), Q, a) = K(z\ (\Omega_n), Q, a) \cap \\ \overline{((S\,fb) \oplus (Q\,f(a+p))} + s) \end{aligned} \qquad (43)$$

We can do the same when computing the new maps for evaluation. Knowing that

$$\begin{aligned} C^{(n+1)}\Theta B(r) &= (C^{(n)} \cap (\overline{(S\,fb) + s)}))\Theta B(r) \\ &= C^{(n)}\Theta B(r) \cap (\overline{(S\,fb) \oplus B(r) + s}) \end{aligned} \qquad (44)$$

then

$$\begin{aligned} x(z\ (\Omega_{n+1}), Q, a, t) = x(z\ (\Omega_n), Q, a, t) \cap \\ (\overline{(S\,fb) \oplus B(r)} + s) \end{aligned} \qquad (45)$$

that, using Eq.(10) can be written as

$$\begin{aligned} x(z\ (\Omega_{n+1}), Q, a, t) = x(z\ (\Omega_n), Q, a, t) \cap \\ (\overline{(S \oplus B(r))\,fb} + s) \end{aligned} \qquad (46)$$

REACHING THE END

Until now we have seen how to compute initial information and how to choose shapes iteratively. The final step is to know when to stop. This happens when one of the following conditions are enabled:

1. There are no more shapes to be placed;
2. There is no free space left inside the container to place any remaining shape.

The first condition just means that the set of shapes to place is empty, that is,

$$\underset{k>0}{\exists} \Phi_k = \varnothing \tag{47}$$

The second condition means that there are no valid positions to evaluate, that is,

$$\underset{n>0}{\exists} \underset{Q\in\Phi_k}{\forall} \underset{a\in\Gamma}{\forall} StrgMk(z\ (\Omega_n), Q\, fa\ ) = \varnothing \tag{48}$$

RESULTS

Tests were performed using a PentiumII processor running at 350Mhz. Figures 1 and 2 show examples taken from footwear industry. In both cases, the container is a hide of 3m by 1.5m approximately. The goal is to minimise the waste on the container by selecting appropriate rotations and positioning for each of the parts to be cut.

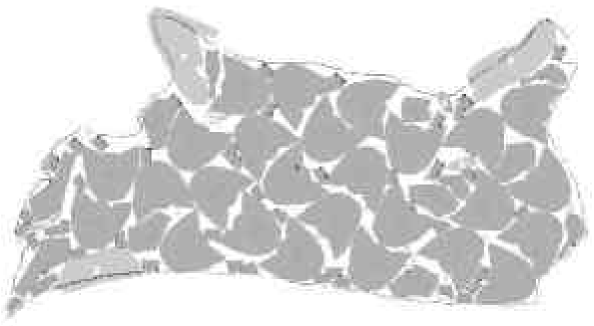

Figure 1: Leather nesting using the algorithm. Total waste is **19.6%**

In the first example, calculations took about 20 minutes for a waste of 19.6% of the container using 24 rotations (15º step) for each shape.

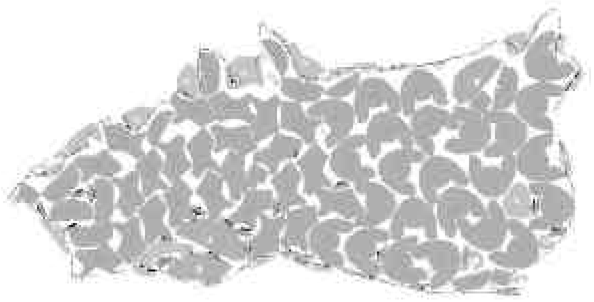

Figure 2: Leather nesting using the algorithm. Total waste is **21.0%**

The second example is also from the footwear industry and took 17 minutes to get 21.0% waste using 24 rotations for each shape. In general, the difference between manual containment (done by skilled persons) and the automatic one is 5% to 7%.

CONCLUSIONS

While researching for an optimal solution for 2D rotational containment problems is still being done, tight time requirements present an additional difficulty in implementing such a solution. The algorithm discussed in this article is intended to approximate with manual results, while maintaining practical execution times. At this time of development, practical results are very promising, and can be upgraded incorporating other automatic containment technologies, such as partial A.I. search.

FUTURE WORK

In the near future, we aim to optimise multiple layer containment, so it can compete with manual containment and apply partial A.I. search methods so that it can make “smarter” decisions. Time constraints are very restricting when we’re dealing with implementation issues. Future development on the geometric operators is already in place, which will allow a speed up of the calculations while maintaining approximation errors controlled.

REFERENCES

[Bernardo95a] Bernardo, J.: *Automatização Industrial do Corte de Modelos Bidimensionais e Irregulares - Uma Aproximação Baseada em Objectos*. I.S.T. Tese de Doutoramento, 1995.

[Cavalier96a] Cavalier, T., Grinde, R.: *A New Algorithm for the Two-Polygon Containment Problem.* Computers and Operations Research, 1996.

[Daniels95a] Daniels, K.: *Containment Algorithms for Nonconvex Polygons with Applications to Layout.* Harvard University; Cambridge, Massachusetts, 1997.

[Marques98a] Marques N., Capela P., Bernardo J.: *Implementing Optimal A.I. Informed Search Methods for 2D Containment Problems*. Presented at EURO XVI Conference 1998.

[Marques98b] Marques N., Capela P., Bernardo J.: *Heuristic Reasoning for 2D Containment Problems*. Presented at WSCG'99 Conference.

[Milenkovic97a] Milenkovic, V.J.: *Rotational Polygon Overlap Minimization*. Computational Geometry: Theory and Applications 1997.

[Serra82a] Serra, J. *Image Analysis and Mathematical Morphology*, Vol. 1. Academic Press, New York, 1982.