Constraint-based simulation of carton folding operations

G. Mullineux[a,*], J. Matthews[a]

[a] *Innovative Design and Manufacturing Research Centre,*
Department of Mechanical Engineering,
University of Bath, Bath BA2 7AY, UK

Abstract

Cartons are a common means for packaging a variety of items. These are usually "erected" by dedicated machines which fold the carton into shape from a flat net. When there is a need to use new forms of carton, dedicated machinery may not be appropriate and reconfigurable systems may be used instead. There is a need to simulate the erection process itself to ensure that this behaves as expected. This paper investigates the use of constraint-based techniques to produce such a simulation. The necessary commands are generated from the geometry of the carton net. Geometric constraints are imposed to ensure that the net remains intact and so resolve cases where loops of faces are present.

Key words: transforms, simulation, cartons, carton erection, folding, packaging, origami

1. Introduction

Cartons represent a popular way of packaging a variety of consumer products including food items, electronic components and stationary supplies [1]. They are versatile and they are likely to increase in popularity as they have good environmental properties in regard to recycling and degradability.

Cartons are formed or erected from flat nets of carton-board which have been cut and printed. Additionally creases are normally pre-formed by pressing thin metal rules into the flat board. Typically the erection process is car-

*Corresponding author
Email addresses: `G.Mullineux@bath.ac.uk` (G. Mullineux), `J.Matthews2@bath.ac.uk` (J. Matthews)

ried out by dedicated packaging machines which fold along the appropriate creases [2]. The carton is usually held in its erected state by gluing or by tucking and locking of flaps.

For some ranges of products, there is a desire to adapt the carton to accommodate different sizes and quantities [3]. There is also interest in “innovative” designs of cartons [4, 5, 6] so as to provide greater appeal for the customer. These ideas can be accommodated by designing new dedicated machines or introducing reconfigurable processes [7]. Reconfigurability can be obtained by the use of robots and these have been used in various ways to fold cartons [8, 9, 10]. Additionally, there has been related interest in paper folding [11, 12] and in the use of robots to form origami models from paper [15].

When designing a packaging system, there is naturally a need to simulate the machinery and its interaction with the packaging material. There are difficulties when the material is carton-board as its properties are non-linear [16, 17] and can vary with the properties of the environment in which processing takes place [18]. Often numerical techniques such as finite element analysis, with a suitable representation of the material properties, are needed [19, 20, 21].

When dealing with reconfigurable systems, it is also important to simulate the erection process of the carton itself. This is to ensure that the process is feasible and no unwanted interferences occur between different parts of the carton. It also helps in checking that parts of the net which are to be “captured” by other parts are moved to appropriate positions. The simulation also provides data for the machine in terms of settings [22] and control parameters.

This paper looks at how to provide such a simulation of the motion of the carton during erection. It aims to do this based on data from the original flat net for the carton. Folding takes places along creases and these divide the net into faces. The next section discusses the face graph [23] which shows the adjacency between faces. When this has no loops (as can occur with simple cartons [9, 10, 11, 24]), the erection process is one of driving all the creases through the required angles, assuming that there are no cases of faces interfering with each other. The interference problem is certainly an issue. For simple cartons it is usually straightforward to see how interferences can be avoided, but this is not the case for more general folding operations [25].

When loops are present, “gussets” exist in which panels move because of the motion of neighbours. Section 3 discusses how transforms can be applied to simulate the motion of the faces of cartons and it shows how commands can be created to enable the angles for gusset folds to be found using a constraint-based approach. An example of a tray carton is discussed in the early sections. A second example of a skillet carton is given in section 4. Section 5 investigates whether the approach can be extended to more complex folding situations and looks at an example taken from origami. In particular this highlights the large angular changes that may be induced in some gusset folds by relatively small changes in the positions of neighbours. This means that care is required to ensure that constraint resolution finds the appropriate angular positions.

2. Graphs

Consider the tray carton [4] shown in figure 1. The figure gives various stages during the erection process. The original net is shown in figure 2. It has fifteen face panels in total. These are numbered with the panel for the base being numbered zero. This can be regarded as being fixed with the other faces being moved relative to it. It is likely that a mechanism or machine used to erect the tray acts principally on the main side walls. So panels 1, 4, 7 and 10 are turned through 90° and panels 13 and 14 are each turned through 180° relative to panels 4 and 10.

In the corners of the tray are four "gusset" arrangements. An example comprises the panels 2 and 3. These need to move inwards, as seen in figure 1, and they are finally captured and held inside the short double-wall end as panel 13 is folded over. The gussets do not require to be pushed explicitly; they move naturally as the main panels to which they are attached move. However, they may need to be guided during the initial stages of their motion to ensure they move inwards and not outwards.

Also shown in figure 2 is the face graph of the carton [23]. The nodes of this graph correspond to the panels of the carton, and two nodes are joined if, in the carton net, the two faces have a common edge. (If the face net is itself regarded as a planar graph, then the face graph is essentially the dual graph.)

Similar graphs are used to represent the interconnection of the links of mechanisms or robots [26], and they have also been used to describe the hierarchy of "model spaces" used to represent such systems [27]. A model space is a collection of geometric entities described with respect to a local coordinate system. It also has associated with it a transform which maps

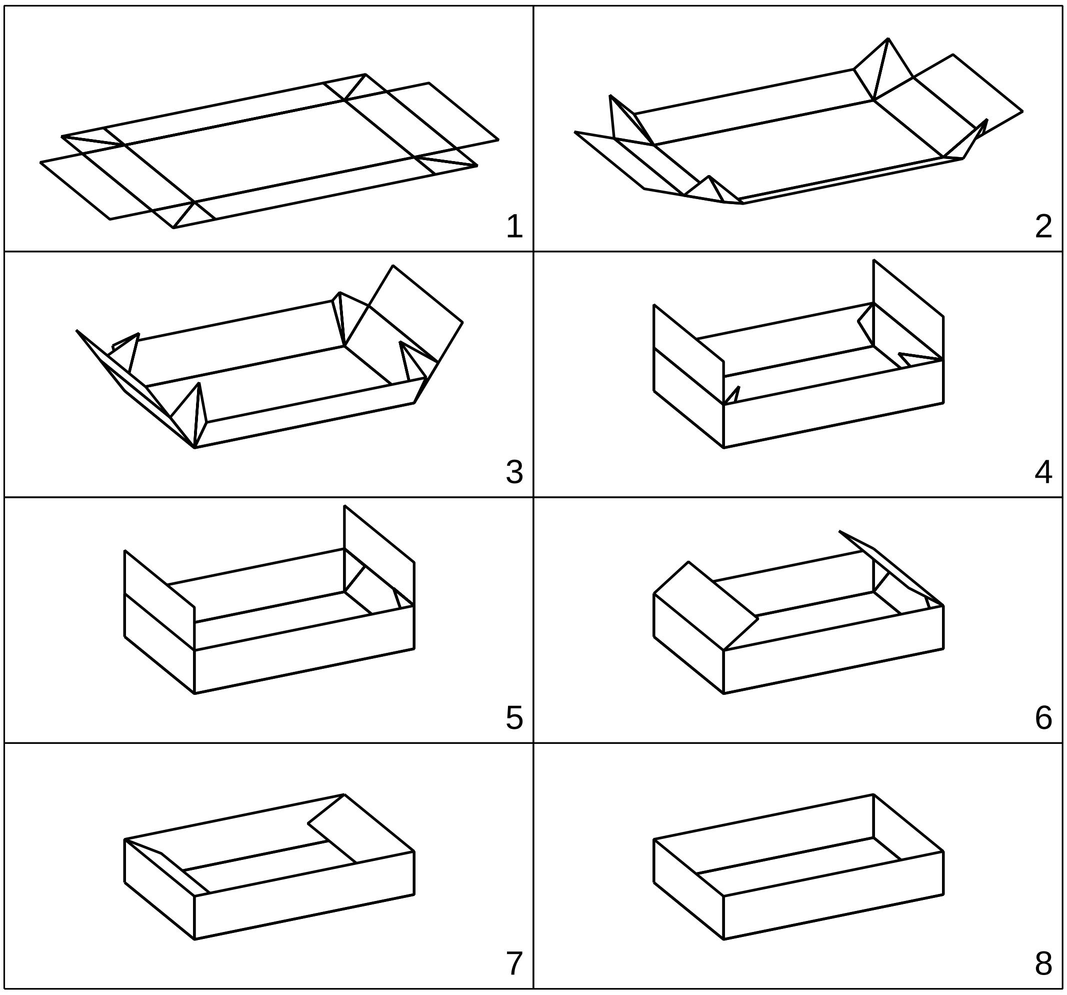


Figure 1: Folding of an origami tray carton

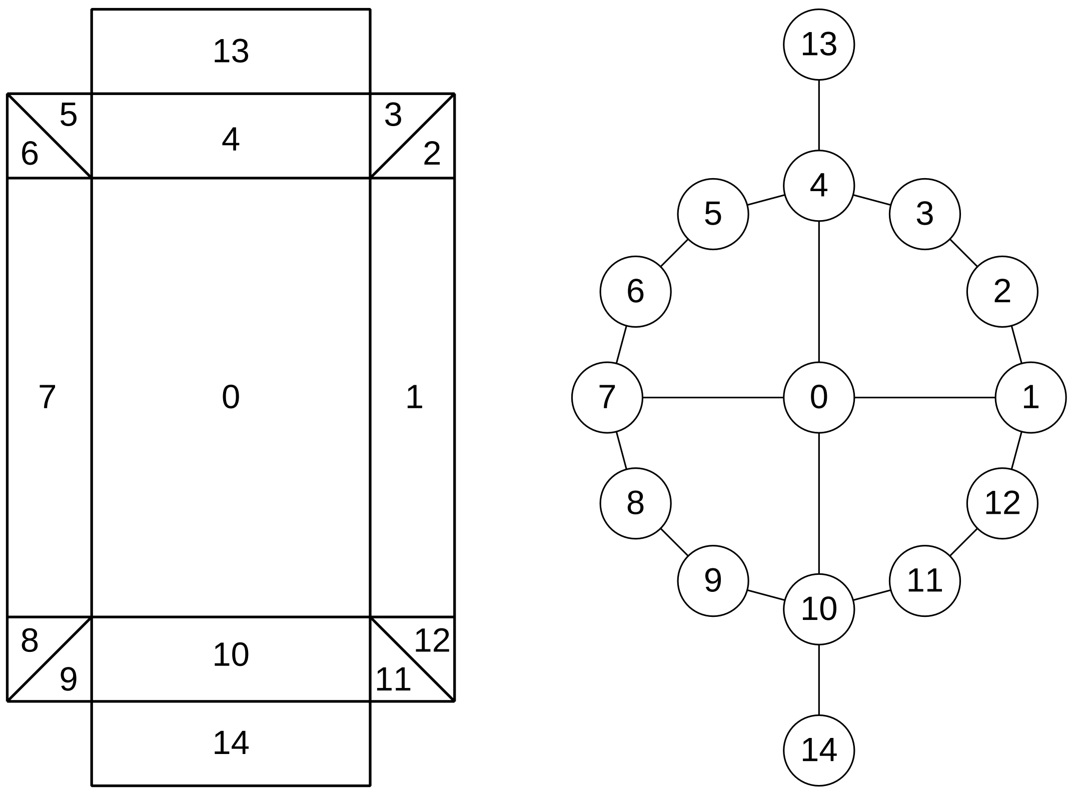


Figure 2: Tray carton and associated face graph

from the local space to another coordinate system. That second system can either be world space or another model space. A hierarchy is created in which the nodes are the spaces themselves and the edges can be regarded as the transforms mapping between these. The hierarchy forms a tree in which the root node is world space. To find how any other space relates to the world it is necessary to map it by each of the transforms along edges between it and world space. As the hierarchy is a tree, this sequence of transforms is unique.

If each of the transform is specified, then the positions of each of the model spaces is determined. So, for example, with a conventional robot, if each of the joint angles is known, then the position of the end-effector is established. However, in practice, one needs to take the end-effector to a given point. This effectively creates a new edge joining the end-effector directly to world space. This in turn creates a loop in the hierarchy and it is no longer a tree. Once there are loops, determining the transforms is more difficult as there is likely to be conflict depending on which way one passes around the loops.

An example is shown in figure 3 which shows the three moving links of a four bar mechanism. The coupler and driven links are not joined. The corresponding hierarchy is also shown in the figure. Joining the two links creates an additional edge (shown as a dashed line) and a loop is formed. One way to establish the assembly is to consider the angles of the coupler and driven links as being free. Two such links form a “dyad”. Forming the expressions for the ends of the links and setting them equal leads to two simultaneous non-linear equations for the angles. These can be solved numerically by a variety of techniques.

In the case of the tray carton, each of the four gusset corners can be regarded as a dyad. If each is thought of as being cut along its diagonal fold, as shown in exaggerated form on the left in figure 4, then the resultant face graph is that shown on the right of the figure. This graph is a now a tree, in fact a spanning tree of the original graph, and its edges represent transforms which involve rotations about the corresponding edges in the carton net. The cuts in the dyads represent non-linear equations which need to be solved.

The approach adopted here is to deal with each dyad in terms of a geometric constraint [28, 29, 30] requiring its two sides to join. Use is made of a constraint modelling environment [31, 32] in which the geometric entities and the constraints between them are specified via a user interface language. What is done is to create automatically the command file for the constraint

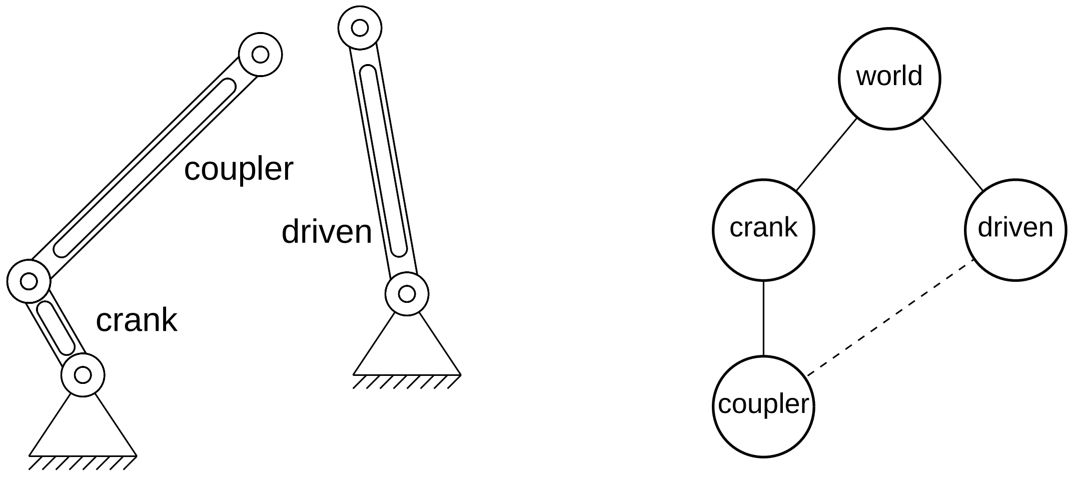
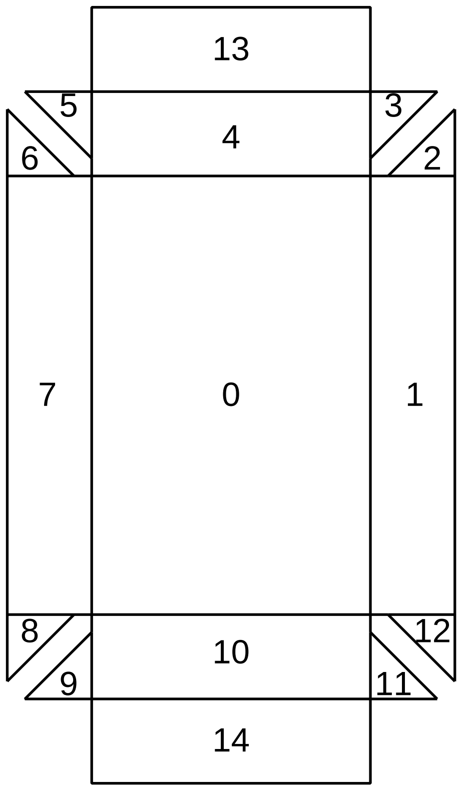


Figure 3: Four bar mechanism and graph of spaces

modeller from a description of the net for the carton (and its dyads). The command file specifies the constraints which need to be solved to assemble the carton at any stage in its erection. A simulation can be obtained by specifying a sequence of values for the rotations about the uncut edges. These can then be stepped through and at each stage the constraints resolved. The next section looks at the creation of the command file from the carton net.

Finally in this section, a brief overview is given of the constraint modelling approach which is used for the examples in this paper. A number of approaches to constraint-based design have appeared [33, 34, 35, 36, 37, 38] with applications including graphics, conceptual design, and embodiment design. Various means for resolving constraints have been used including computational reasoning and numerical solution of simultaneous equations. The constraint modelling environment [31, 32] uses optimisation techniques for constraint resolution. It has has an underlying language in which design parameters are declared. These can include graphical entities such as points, lines and arcs.

A constraint is specified as an expression involving some of the design parameters. This is deemed to be true when its value is zero; if it has a non-zero value (as a real number) then this is a measure of its falseness. For each set of constraints, the user specified which parameters can be changed in order to satisfy those constraints. The system then treats the sum of the



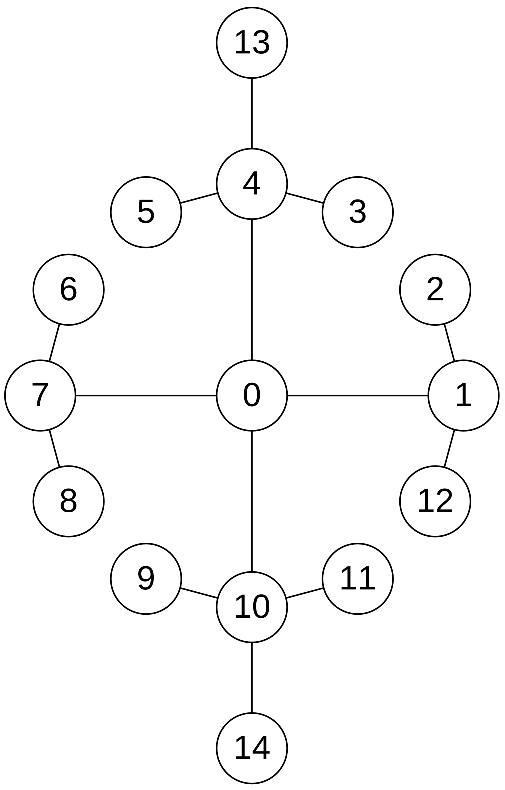


Figure 4: Tray carton with cuts and associated spanning tree of face graph

squares of the constraint expressions as a function of the parameters which can be varied and searches for a (local) minimum value using numerical optimisation techniques (such as Powell's direct search method [39]). If a minimum of zero is found, then it is known that the constraints have been resolved satisfactorily. A non-zero minimum suggests that the constraints are in conflict and cannot be fully resolved.

The aim is to find a single satisfactory configuration. This means that the initial values of the variable parameters is important as this is the starting point for the numerical search. The approach is intended as an aid to the designer in exploring the design space. In the early stages of the design process, the constraints are evolving as the designer gains understanding of the design task. Specifying the known constraints to the environment allows their implications to be assessed so that the constraints can be modified or extended as necessary.

As an example, part (a) of figure 5 shows a representation as a "stick model" of the four bar mechanism given in figure 3. The lines representing the coupler and driven links are `Lcoupler` and `Ldriven` and the constraint to bring their ends together is

```
rule( Lcoupler:e2 on Ldriven:e2 );
```

Here the extension `e2` denotes the second end point of the line and `on` is a binary function which returns the distance between two geometric objects; in the case of points this is zero when they coincide.

The system finds the solution in part (b) of figure 5, when the constraint is resolved allowing the angles of the two lines to vary (and keeping the crank fixed). This is starting from the configuration shown in part (a). There is of course another solution as shown in part (c). If this is the one that is desired, then either the two lines need to be rotated "close" to this configuration before the search is made, or additional constraints imposed to disallow the other solution.

3. Transforms

This section describes how a description of the geometry of the original carton net is used to create a simulation of the erection process. The description of the net is as a collection of points and faces supplied in a data file. This file is processed by a stand-alone pre-processor program to create

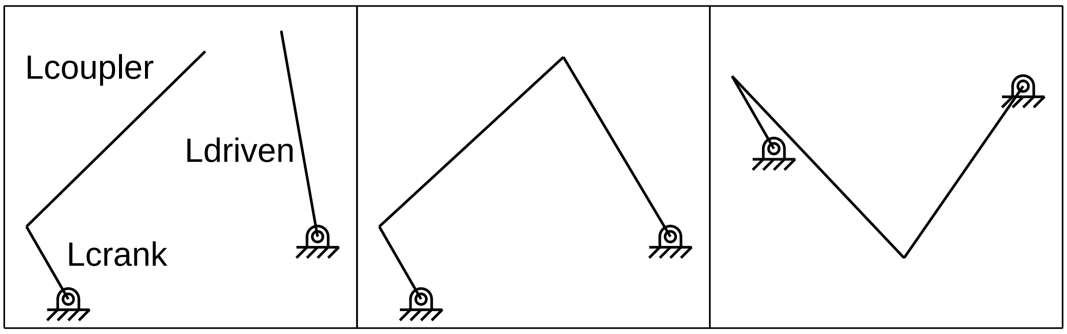


Figure 5: Constraint resolution for links forming a dyad

commands for the constraint modelling environment. It is within this that the simulation is performed.

Each point represents the vertex of one of the face panels. It is labelled with a name and its coordinates are specified using a suitable coordinate frame for the net. Each face is specified as the sequence of points around it (in an anticlockwise sense). The pre-processor program needs to determine the hierarchy of the faces corresponding to the spanning tree of the face graph that is to be used. This can be done automatically by declaring within the data file which edges have been cut and which remain; this gives sufficient information for the pre-processor to determine the hierarchy. It can also be achieved, more simply, by tagging each face with a reference to the next face in the hierarchy. The base face is used as the root node of the hierarchy.

The pre-processor program generates commands to create lines representing the boundary of each face. Each internal line is created twice: once for each face in which it lies. For convenience, call the "joining edge" of a face that edge which is common to it and the next face in the hierarchy nearer to the base. The lines are created in a local coordinate system for the face. This is chosen so that its joining edge forms the local x-axis. Figure 6 shows two adjacent faces f and F with f being nearer to the base. The local coordinates for face F use point (X_0, Y_0) as the origin, with the x-axis going towards point (X_1, Y_1). If the typical point in face F has coordinates (X, Y) with respect to the frame of the net, these can be expressed as a column vector of homogeneous coordinates $[X, Y, 0, 1]^T$. Then transforming by pre-multiplying by the matrix

$$\begin{bmatrix} \cos\beta & \sin\beta & 0 & -X_0\cos\beta - Y_0\sin\beta \\ -\sin\beta & \cos\beta & 0 & X_0\sin\beta - Y_0\cos\beta \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where β is the angle between the joining edge and the global x-axis, has the effect of transforming the point to the local coordinates of the face.

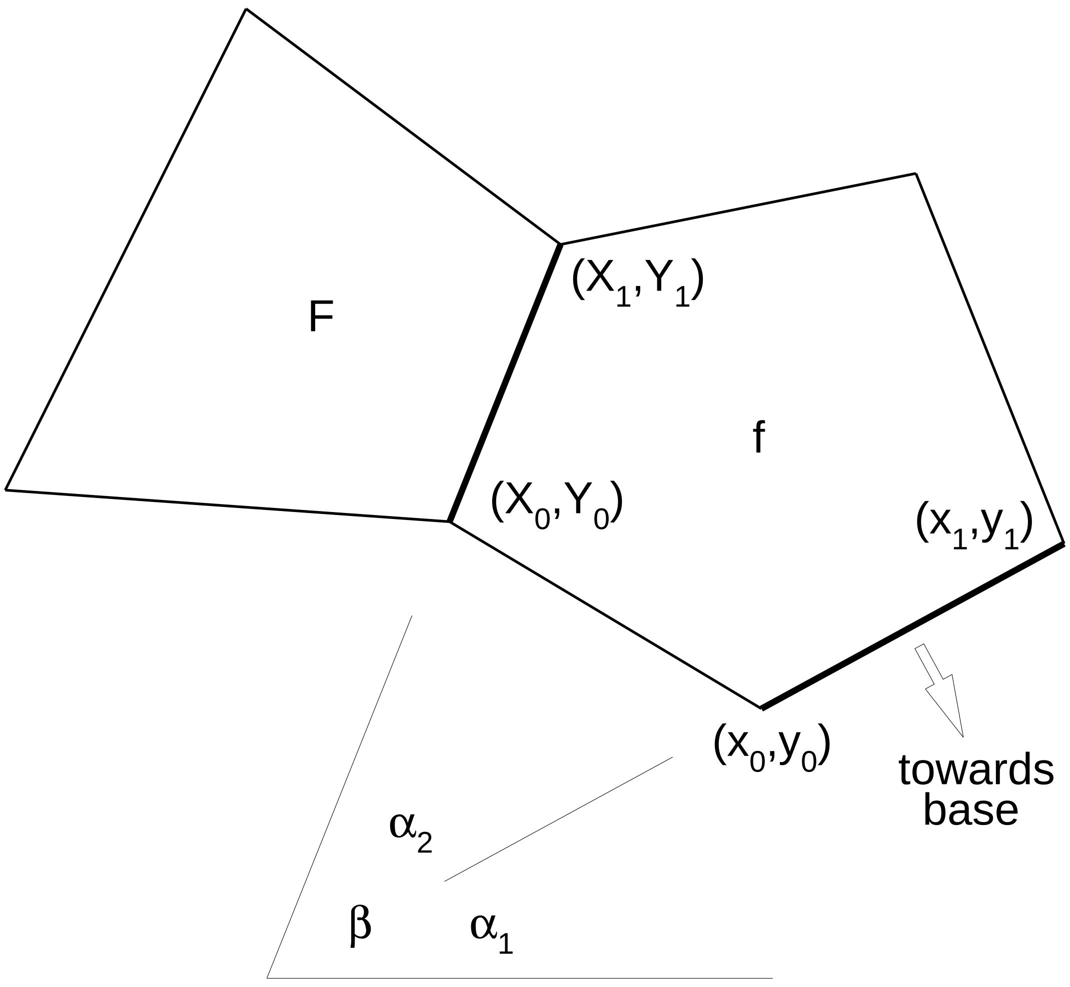


Figure 6: Transforms between two adjacent panel faces

As noted previously, the edges of the face graph can be regarded as transforms which relate one face to the next. For simplicity, the pre-processor sets up two transforms to be used one after another. A similar technique is used in [40]. Within the commands for the constraint modeller, this means creating two model spaces, one embedded in the other. The first transform that is applied is a rotation about the x-axis. Since locally this axis is the edge joining to the next face, this transform represents folding of the crease.

The second transform maps from one local space to the next with a translation in x and y and a rotation about an axis in the z-direction. Referring again to figure 6, the following matrix product

$$\begin{bmatrix} \cos\alpha_1 & \sin\alpha_1 & 0 & 0 \\ -\sin\alpha_1 & \cos\alpha_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & & & -x_0 \\ & 1 & & -y_0 \\ & & 1 & 0 \\ & & & 1 \end{bmatrix}$$

where α_1 is the angle between the joining edge for face f and the global x-axis axis, maps the space of face f so that its joining edge maps to the global x-axis with the end (x_0, y_0) mapping to the origin.

Let $\bar{X_0}$ and $\bar{Y_0}$ be the x and y coordinates of point (X_0, Y_0) under this map. Then the matrix

$$\begin{bmatrix} \cos\alpha_2 & -\sin\alpha_2 & 0 & \bar{X_0} \\ \sin\alpha_2 & \cos\alpha_2 & 0 & \bar{Y_0} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where α_2 is the angle between the two joining edges, creates the transform mapping the space of F to that of f. The pre-processor uses the values in this matrix to create commands to set up the second transform.

The pre-processor always creates the commands which specify the constraints which need to be solved for the dyads. The data file for the net declares these in terms of pairs of faces (between which a “cut” has been formed) and an indication of which points need to be brought together. As noted earlier, each gusset corner can move inwards or outwards. So it is also possible to specify bounds of the values for the angles of the folds so as to ensure motion in the appropriate direction.

Figure 7 shows one of the gussets corners for the tray carton with the cut between the two faces exaggerated. The following are the commands for the constraint modelling environment created by the pre-processor.

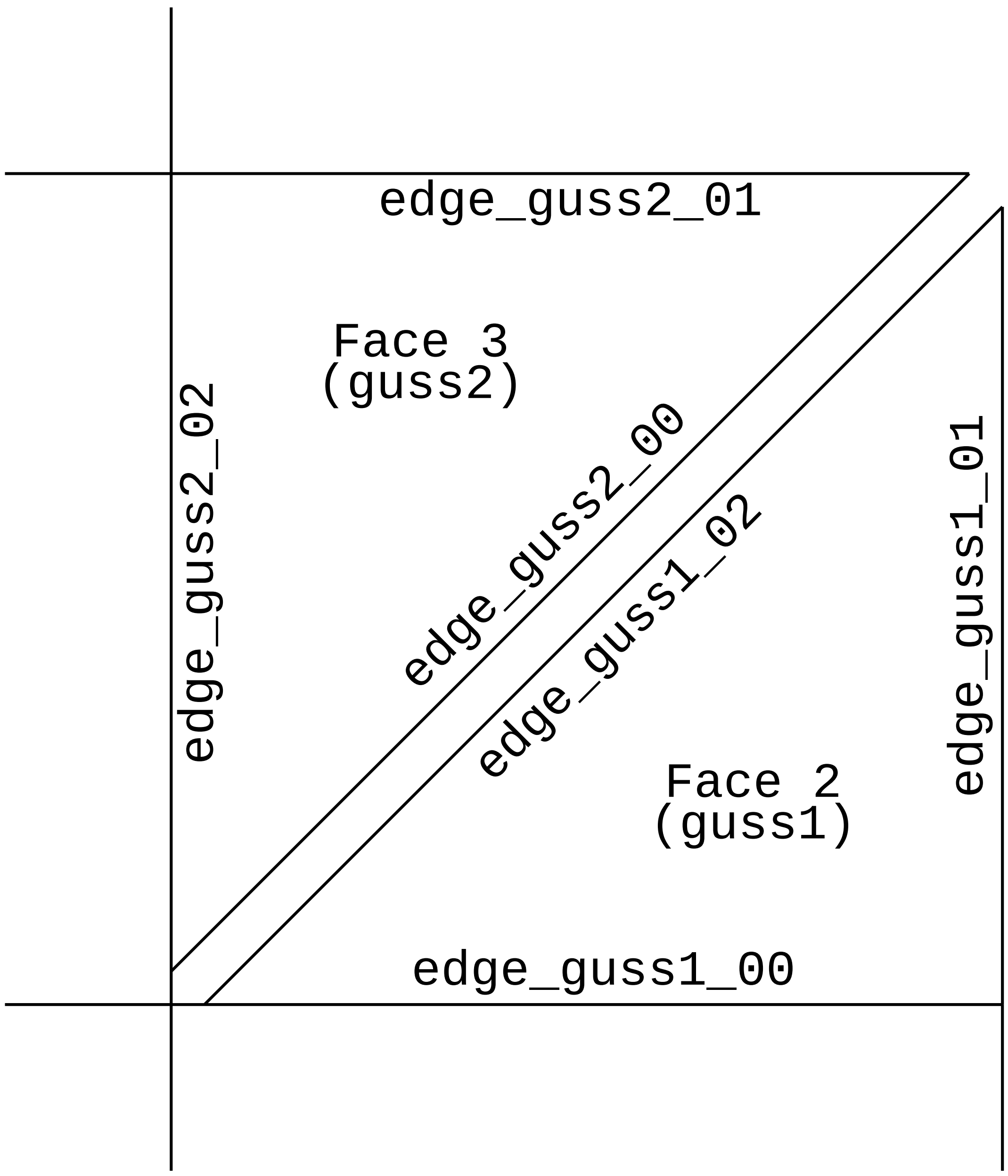


Figure 7: Dyad formed by gusset corner in tray carton

```
dec mod3 mspc_guss1_posn, mspc_guss1_turn;
dec geom edge_guss1_00, edge_guss1_01, edge_guss1_02;

dec mod3 mspc_guss2_posn, mspc_guss2_turn;
dec geom edge_guss2_00, edge_guss2_01, edge_guss2_02;

mspc_guss1_posn = mod3( 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, mspc_dsid1_turn );
mspc_guss1_turn = mod3( 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, mspc_guss1_posn );

edge_guss1_00 = lin( 0.0, 0.0, 0.0, 5.0, 0.0, 0.0, mspc_guss1_turn );
edge_guss1_01 = lin( 5.0, 0.0, 0.0, 5.0, 5.0, 0.0, mspc_guss1_turn );
edge_guss1_02 = lin( 5.0, 5.0, 0.0, 0.0, 0.0, 0.0, mspc_guss1_turn );

mspc_guss2_posn = mod3( 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, mspc_dsid2_turn );
mspc_guss2_turn = mod3( 0.0, 0.0, 0.0, .00, 0.0, 0.0, mspc_guss2_posn );

edge_guss2_00 = lin( 5.0, 0.0, 0.0, 0.0, 5.0, 0.0, mspc_guss2_turn );
edge_guss2_01 = lin( 0.0, 5.0, 0.0, 0.0, 0.0, 0.0, mspc_guss2_turn );
edge_guss2_02 = lin( 0.0, 0.0, 0.0, 5.0, 0.0, 0.0, mspc_guss2_turn );

function resolve_guss12
{
    var mspc_guss1_turn:ax;
    var mspc_guss2_turn:ax;

    rule( edge_guss1_01:e2 on edge_guss2_00:e2 );

    rule( mspc_guss1_turn:ax .ge.  0.0 );
    rule( mspc_guss1_turn:ax .le.  180.0 );
    rule( mspc_guss2_turn:ax .ge.  0.0 );
    rule( mspc_guss2_turn:ax .le.  180.0 );
}
```

The first four of these commands declare the names of the lines and the two pairs of model spaces used. These are then defined, with the model spaces embedded in each other appropriately and the lines similarly associated with their local spaces. The lines are defined in an anticlockwise sequence around each face. It is within the function that the constraints are specified. The rotations about the relevant local x-axes are specified as being variable when the constraints are resolved. The first constraint specifies that the second end-points of the two given lines should be coincident. The other four constraints limit the angles of rotations so that these lie between zero and 180°.

When the function is invoked within the constraint modeller, the system automatically adjusts the values of the two angles in a search for values which allow all the constraints to be satisfied. There is generally a unique solution to this problem. The exception is when the neighbouring side walls have been turned through 90°. Here the short edges of the gusset faces, those about which they rotate locally, are together and there are infinitely many angles which allow the above constraints to be satisfied. (The constraint modeller is set up to find just one solution and this is normally the same as that found in the previous step of a simulation.) From the point of view of simulating the process, this is inconvenient. So the net is modified by introducing "dummy" faces of zero width between each gusset face and its neighbouring side panel. This adds extra nodes and edges to the face graph and to its spanning tree, and allows the specification of additional angles for the simulation process. The angles of these faces are kept at zero as the side walls are rotated and the gussets move inwards. Then the angles of the dummy faces are varied over a range of 45°. The gusset constraints remain satisfied as this is done as their angular values do not change. This allows the simulation to show the gusset flaps being folded back against the double walls as in the passage between steps 4 and 5 in figure 1. Specifying the appropriate sequence of angles for the edges in the spanning tree and resolving the constraints at each stage allows the full simulation shown in that figure to be run.

The advantage of the constraint-based approach is that it allows investigation of different closing strategies by varying the driving angles while using the same commands to assemble the carton and perform the simulation. For example, the options of closing the side walls at different speeds can be investigated. In the simulation shown in figure 1, these are driven at the same speed. This leaves the gusset flaps pointing into the tray at 45° to the sides. In fact, even at this angle, it is possible for the double walls to fold over, capture and retain them. However driving the longer side wall more quickly than the shorter (double) walls has the effect of forcing the gussets flaps to lie closer to the short walls, thus making their capture more certain. A stage in the simulation of this is shown in figure 8. This simulation is obtained simply by modifying the sequence of driving angles defined for the edges in the spanning tree.

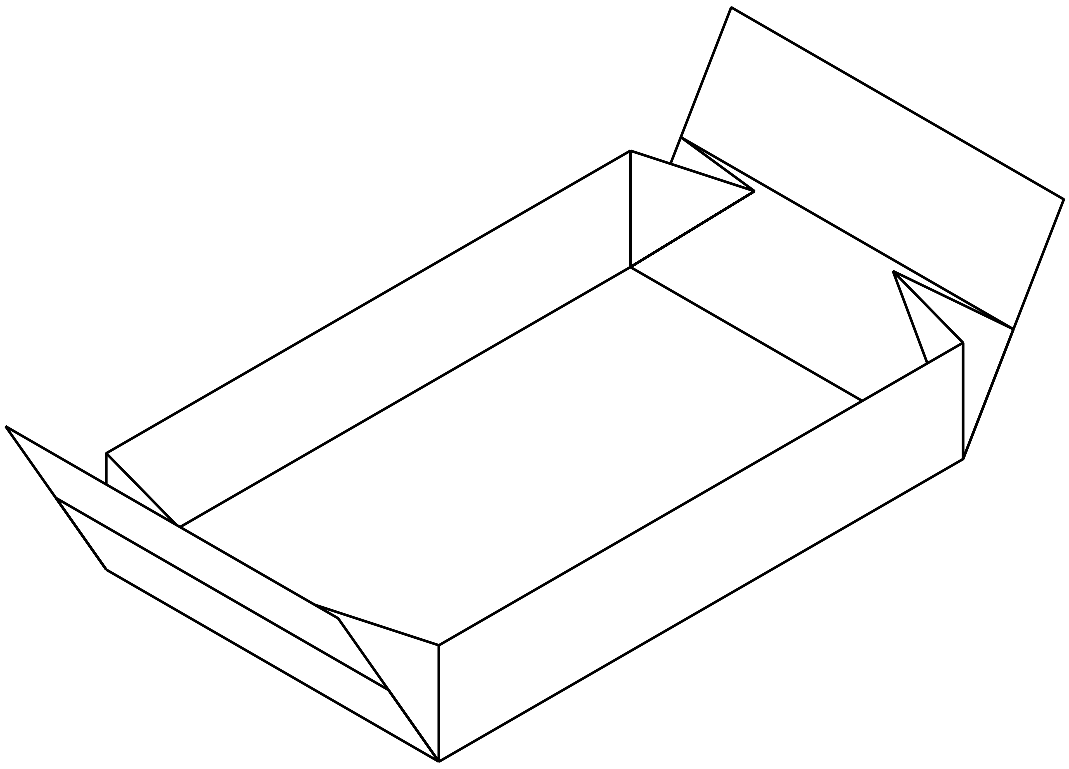


Figure 8: Closing the side faces at different speeds

4. Gable-end skillet

As a second example, consider the carton stages in whose erection are shown in figure 9. The form of the top of the carton is called a "gable-end" and it is also composed of a number of gussets.

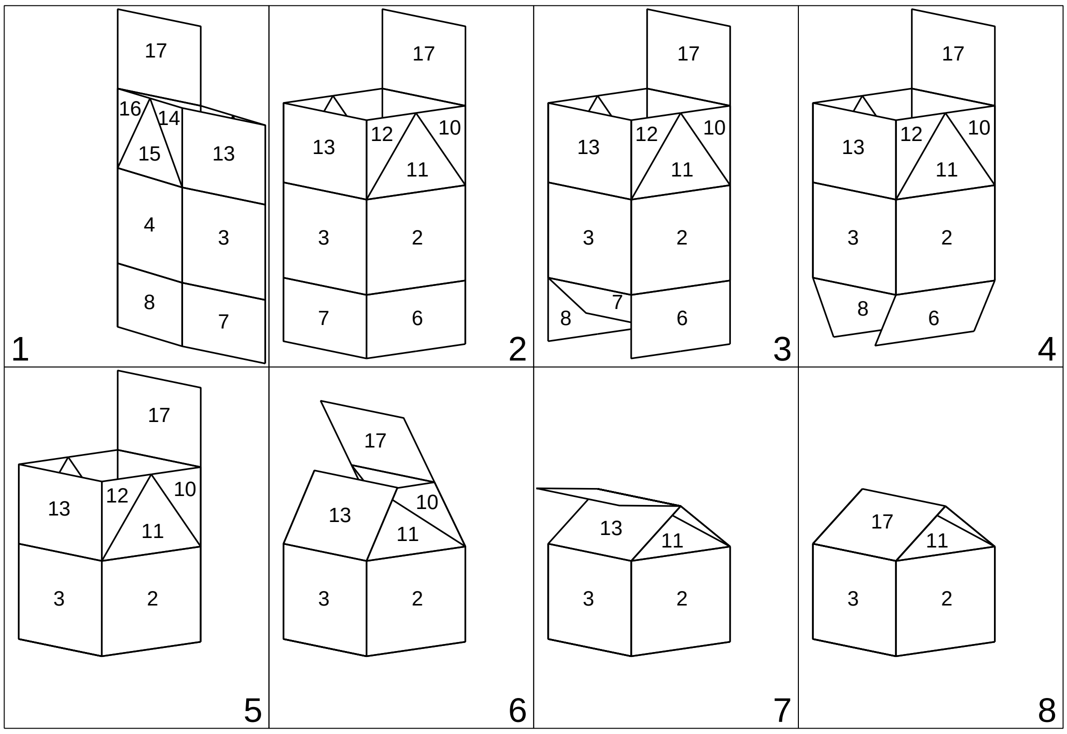


Figure 9: Folding of a skillet with a gable end

The net for the carton is shown in figure 10 along with its face graph. When the carton is produced, the initially flat net is folded along the edges between faces 2 and 3 and between faces 12 and 13. Then tabs along the edges of faces 4 and 16 are glued to faces 1 and 9. This means that the first stage of erection is to open out the carton so that faces 1, 2, 3 and 4 form a parallelogram and finally a square. It is this initial gluing that makes the carton a skillet [6].

Figure 11 shows two possible spanning trees for the face graph. Below each is the corresponding net with the appropriate faces separated. Also shown separated are the flaps at the bottom of the carton, faces 5-8. Note

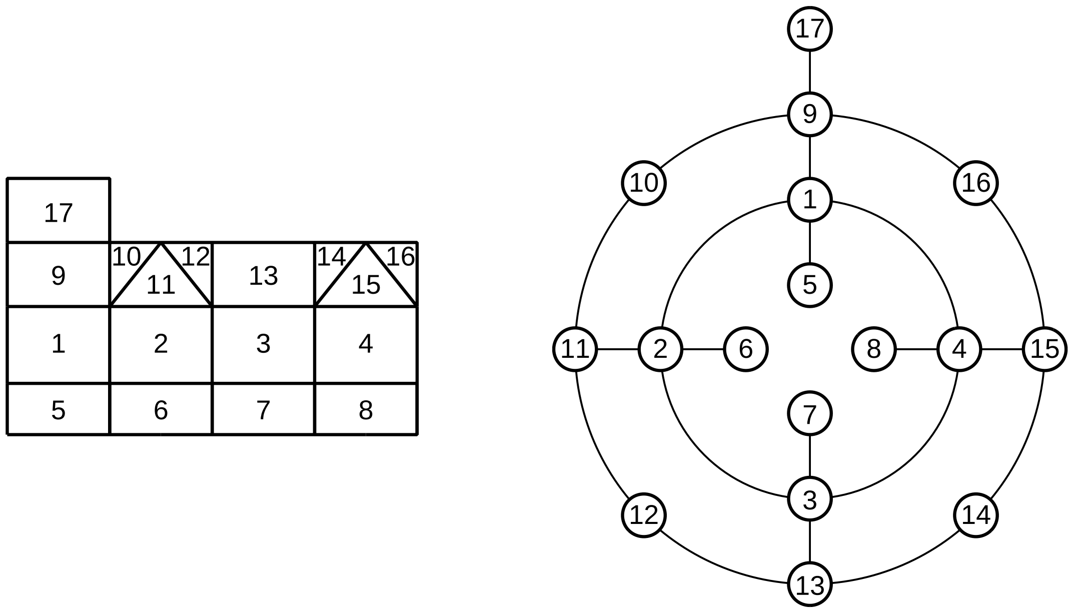


Figure 10: Net and face tree for gable-end skillet

that the nets do not fall into two pieces since there are joins between faces at the left and right sides as indicated by the arrows; for example faces 1 and 4 join. In both trees, the edge joining nodes 2 and 3 is removed. This is on the assumption that during the opening process, face 1 is held fixed and face 4 is driven through 90° as suggested in figure 9. To assemble the main walls of the carton, a constraint is required to bring faces 2 and 3 together.

A little more care is required with the gussets involved in the gable-end itself. This is because there are now cases of five faces with a common vertex. Consider the tree labelled (a). Assuming that face 9 is driven, faces 10 and 11 can be treated as a dyad. Once the position of face 11 is known, faces 12 and 13 can be dealt with as a second dyad; then faces 14 and 15. This leaves faces 16 as 9 as a final dyad. But the position of face 9 is already established. So the final dyad is handled by only adjusting face 16.

In practice, simply driving face 9 is unlikely to be successful in driving all the other faces because of the tendency of the panels to distort as they transmit the motion. Instead, both faces 9 and 13 are driven. This means that the second spanning tree, labelled (b) in figure 11, is more appropriate. It means that every face is at most distance 3 away (in terms of path-length

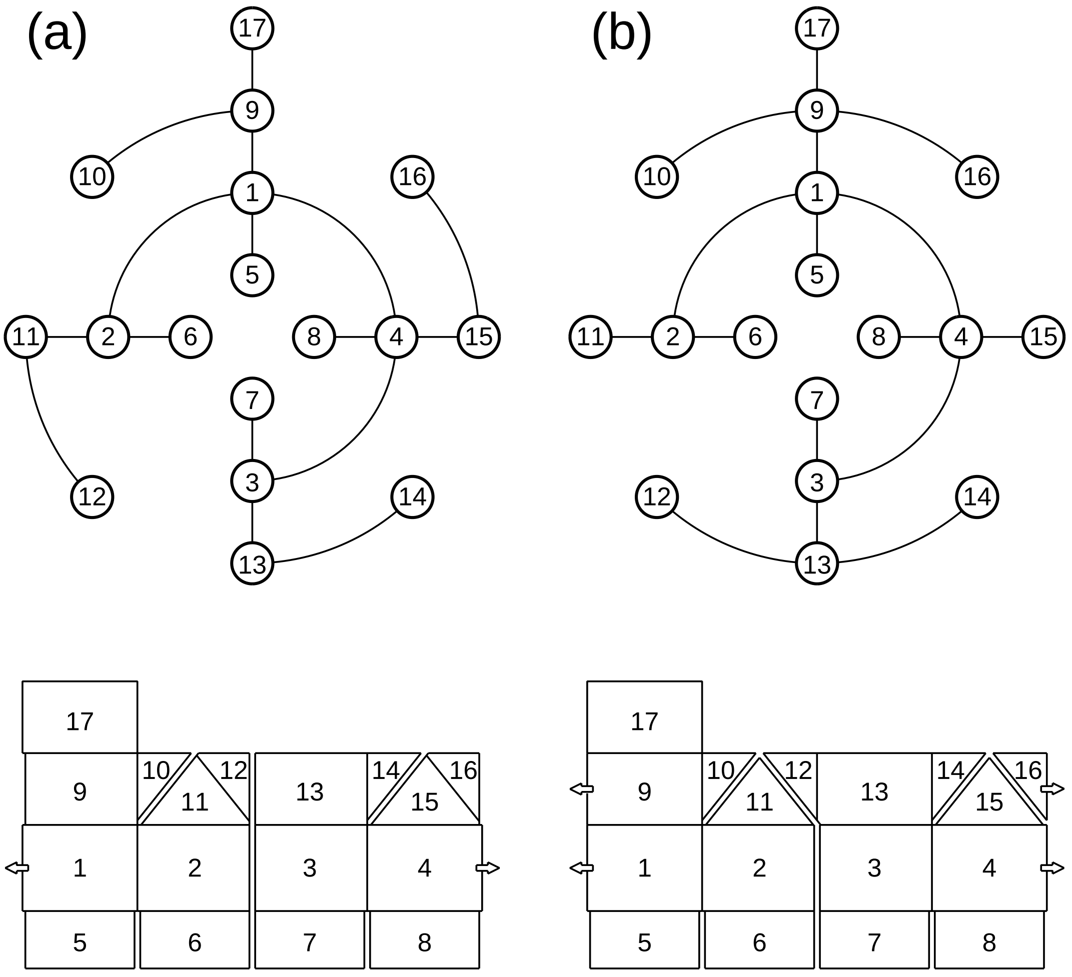


Figure 11: Possible spanning trees

within the tree) from a driven face. A general principle is that one wishes to choose the faces that are to be driven and the spanning tree so as to ensure that every face is close to a driven face and thus problems with distortion are reduced. Using spanning tree (b), faces 10 and 11 are solved as dyad. The dyad between faces 11 and 12 is then handled by adjusting face 12 alone. A similar approach is applied for the other faces around the gable-end.

From the point of view of the simulation, the pre-processor can set up the commands in the same way as before, with two angles declared as variable for each dyad. However, when only one angle is allowed to change, a "fix" command is given to ensure that the other is unaltered during constraint resolution.

5. Extension to origami

The previous sections have looked at the simulation of the erection of cartons. Although there is interest in more complicated packs [5], practical cartons tend to be fairly simple structures. This is partly because of the need to be able to manipulate them by machine or manually. In order to see if the previous simulation techniques work with added complexity, this section looks at their application to an example from origami. This is the "flapping bird" (e.g. [41]) which is a standard origami construction.

As with most origami models, the bird starts with a square of paper. The net used for the simulation is shown in 12. The upper right corner of the net contains the fold lines that form the head of the bird. Stages in the simulation are shown in figure 13.

As with the carton tray, there is a need to introduce a number of "dummy" faces to allow turning of parts once dyads have been fully folded. There are 17 dummy faces shown as shaded regions in figure 12. The 13 central dummy faces all have zero size and are coincident with the central point; showing them larger in the figure distorts the shape of neighbouring faces. The central face allows the simulation to show the parts folding symmetrical downwards and upwards about the mid-point of the square (as for example in passing between stages 1 and 3 in figure 13) . Its 12 neighbouring dummy faces allow the rotation of the main triangular parts about a common vertical axis (as in going between stages 3 and 4 of the simulation). The remaining four dummy faces are used to simulate the raising of the wings (as in stages 7 to 9).

The thick edges shown in figure 12 indicate which edges of the net are regarded as being cut so as to leave a spanning tree. In each case, the faces

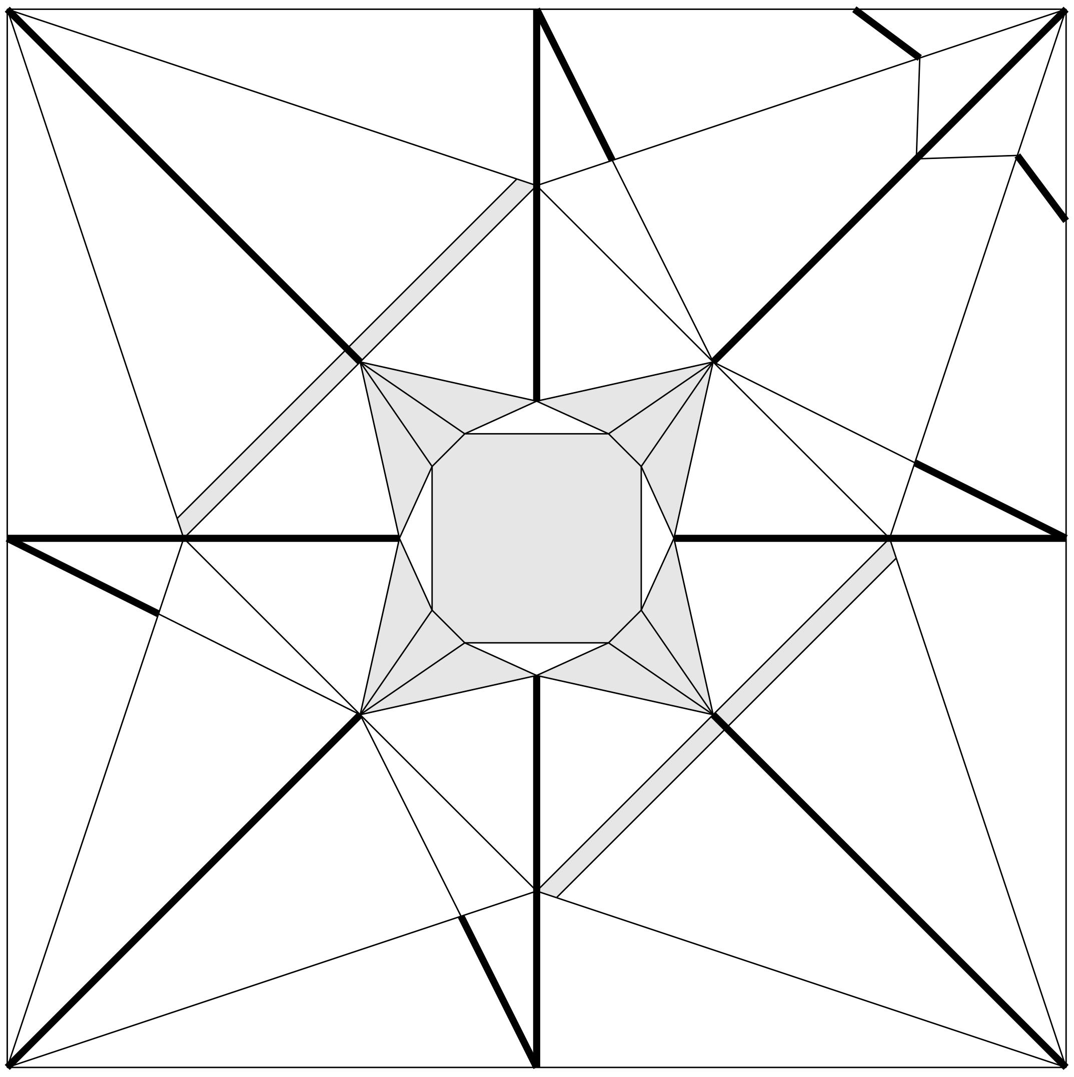


Figure 12: Starting net for origami bird

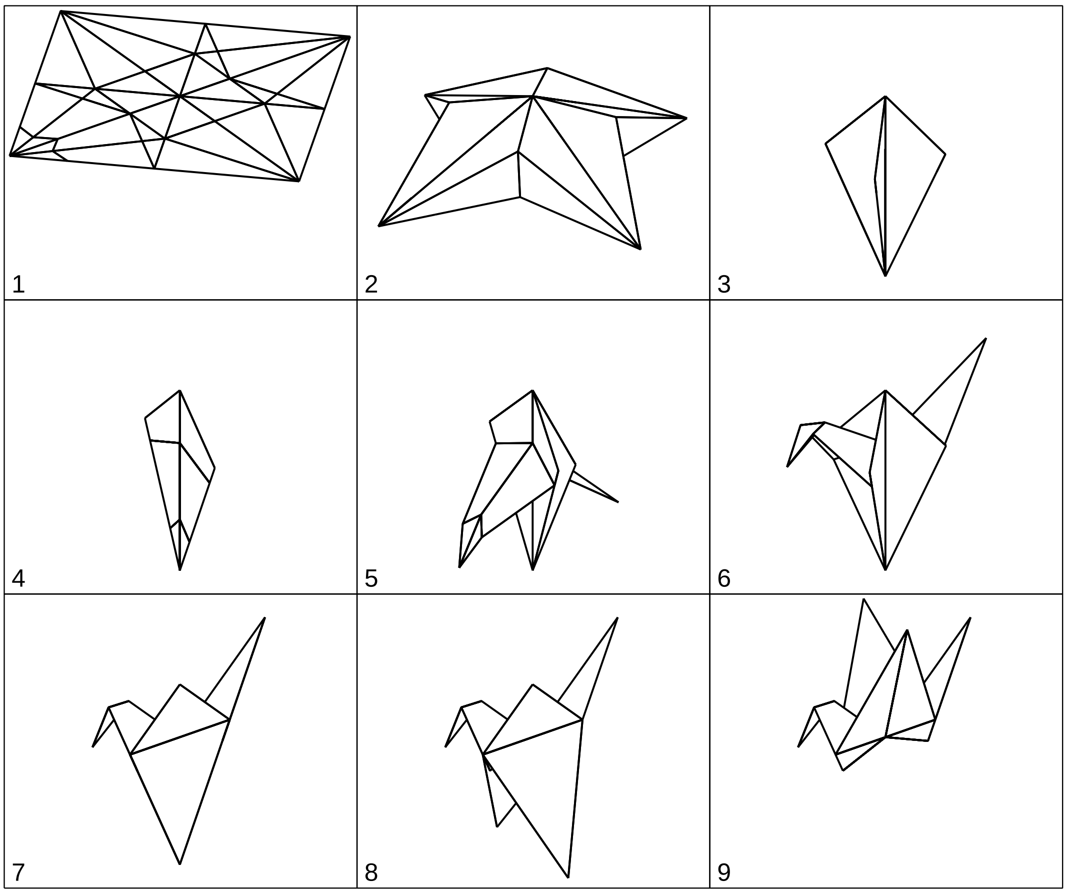


Figure 13: Folding of origami bird

on either side form a dyad, and these can be set in the correct orientation by setting up and resolving constraints. As with the skillet example, there is a need with some dyads to allow only one of the two faces to move. An example is the triangular face along the top edge of the net at the right hand corner. In the simulation, when this is aligned with the edge to its left, that face has already been positioned (with reference to the neighbour on its left), so only the triangular face can be changed.

As noted previously, there are typically two solutions for any dyad assembly and so there is a need to ensure that the appropriate one is found. In the constraint modelling environment, constraint resolution is carried out by a search process. The result achieved can be controlled by ensuring the search starts out in a configuration close to the one desired. Normally this is not a problem, and when it is, it can be overcome by applying constraints to bound the angle of rotation of a face. However it is found with the bird example that a little more care is required than with the cartons discussed previously (cf. also [42]).

To see the difficultly, consider the arrangement of faces shown in figure 14. The point O is taken as the origin of coordinates, with the x-axis along OA, the y-axis as shown, and the z-axis normal to the plane of the figure.

The two main triangles ABR and ACR are joined along line OR and separated along OA. Each is split into two faces along OP and OQ which lie at angle α to the y-axis shown in the figure. Folding around these lines is possible and the angle of rotation about each is assumed to be the same angle θ. Suppose that face $ORBP$ is rotated about OR through an angle ϕ, and face $ORCQ$ is rotated about OR through the same angle but in the opposite sense. Clearly the two points A can be kept together by taking θ to be zero. To find the other possibility, consider the transform matrix for face OAP. Since point O is fixed, the transform matrix is that of a rotation. It can be obtained by combining a rotation of α about the z-axis, a rotation of θ about the new y-axis, a reverse rotation of $-\alpha$ about the z-axis, and a rotation of ϕ about the x-axis. This is represented by the following product.

$$\begin{bmatrix} 1 & & \\ & \cos\phi & -\sin\phi \\ & \sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} \cos\alpha & \sin\alpha & \\ -\sin\alpha & \cos\alpha & \\ & & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & & -\sin\theta \\ & 1 & \\ \sin\theta & & \cos\theta \end{bmatrix} \begin{bmatrix} \cos\alpha & -\sin\alpha & \\ \sin\alpha & \cos\alpha & \\ & & 1 \end{bmatrix}$$

This product can be formed and used to transform, by pre-multiplication, the point $A = [a\ 0\ 0]^T$ regarded as part of face OAP. The y-component of

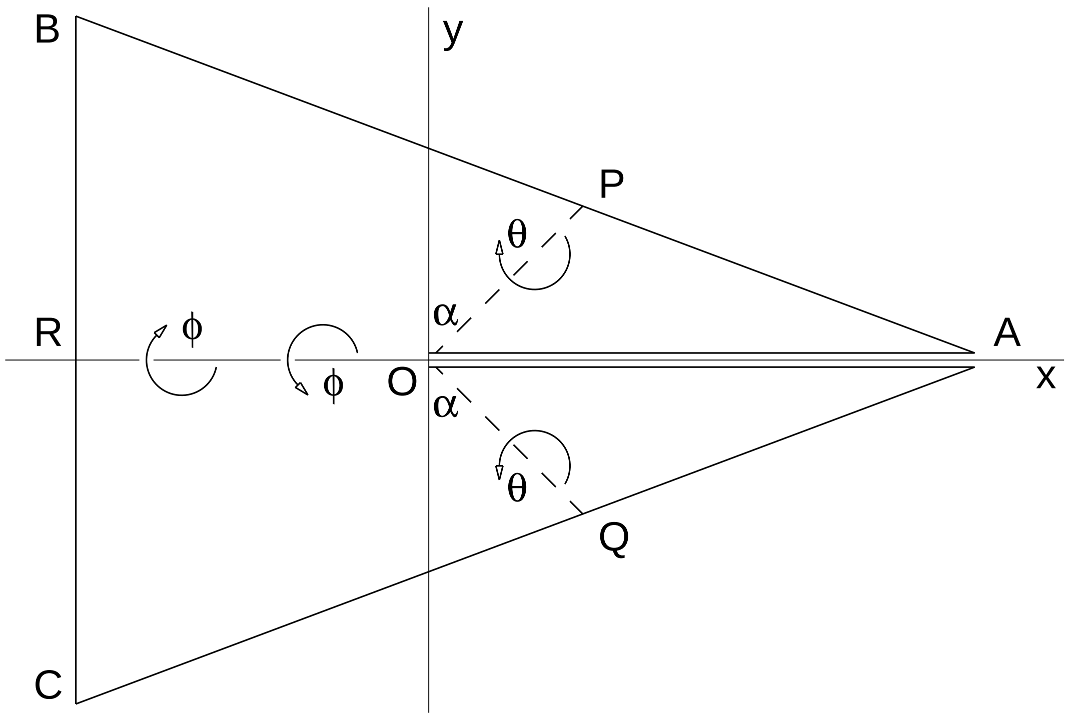


Figure 14: Two folding panels with a dyad

the transformed point is found to be

$$a\cos\alpha[-\sin\alpha\cos\phi\cos\theta \;+\; \sin\alpha\cos\phi \;-\; \sin\phi\sin\theta].$$

By symmetry, when the two points A coincide, the above component is zero and so, assuming that $\cos\alpha$ is non-zero,

$$\begin{aligned} 0 &= \sin\alpha\cos\phi\,(1-\cos\theta) \;-\; \sin\phi\sin\theta \\ &= \sin\alpha\cos\phi\,(2\sin^2\tfrac{1}{2}\theta) \;-\; 2\sin\phi\sin\tfrac{1}{2}\theta\cos\tfrac{1}{2}\theta. \end{aligned}$$

So either $\sin\frac{1}{2}\theta$ is zero, which is the trivial solution, or

$$\tan\tfrac{1}{2}\theta \;=\; \frac{\tan\phi}{\sin\alpha}$$

Figure 15 shows graphs of θ against ϕ as the latter varies between zero and 90° representing a complete fold. For small values of α the initial change in θ is seen to be large.

An instance of this arrangement of faces is present in the net of the bird. This is in the faces that form the neck. They lie on either of the cut along diagonal of the square in the top right of the net in figure 12. The large movement corresponding to the initial change in θ is seen in the movement between stages 4 and 5 in figure 13. To ensure that the correct solution is found in the simulation, the appropriate angles (those equivalent to θ) need to be reset (to around 30°) before the constraints for the dyad are resolved. Once the initial jump has been achieved, constraint resolution for subsequent stages can be undertaken starting with the previously found values of the angles.

6. Conclusions

When designing packaging machinery, there is a clear need to simulate the action of the machine itself to check that it functions correctly. There is also a need to simulate the motion of the carton itself during the various stages of the erection process. This is to ensure that unwanted interference between faces of the carton net does not occur. This is particularly the case when reconfigurable equipment is being used to check that the configuration is correct and to establish what motion control is required.

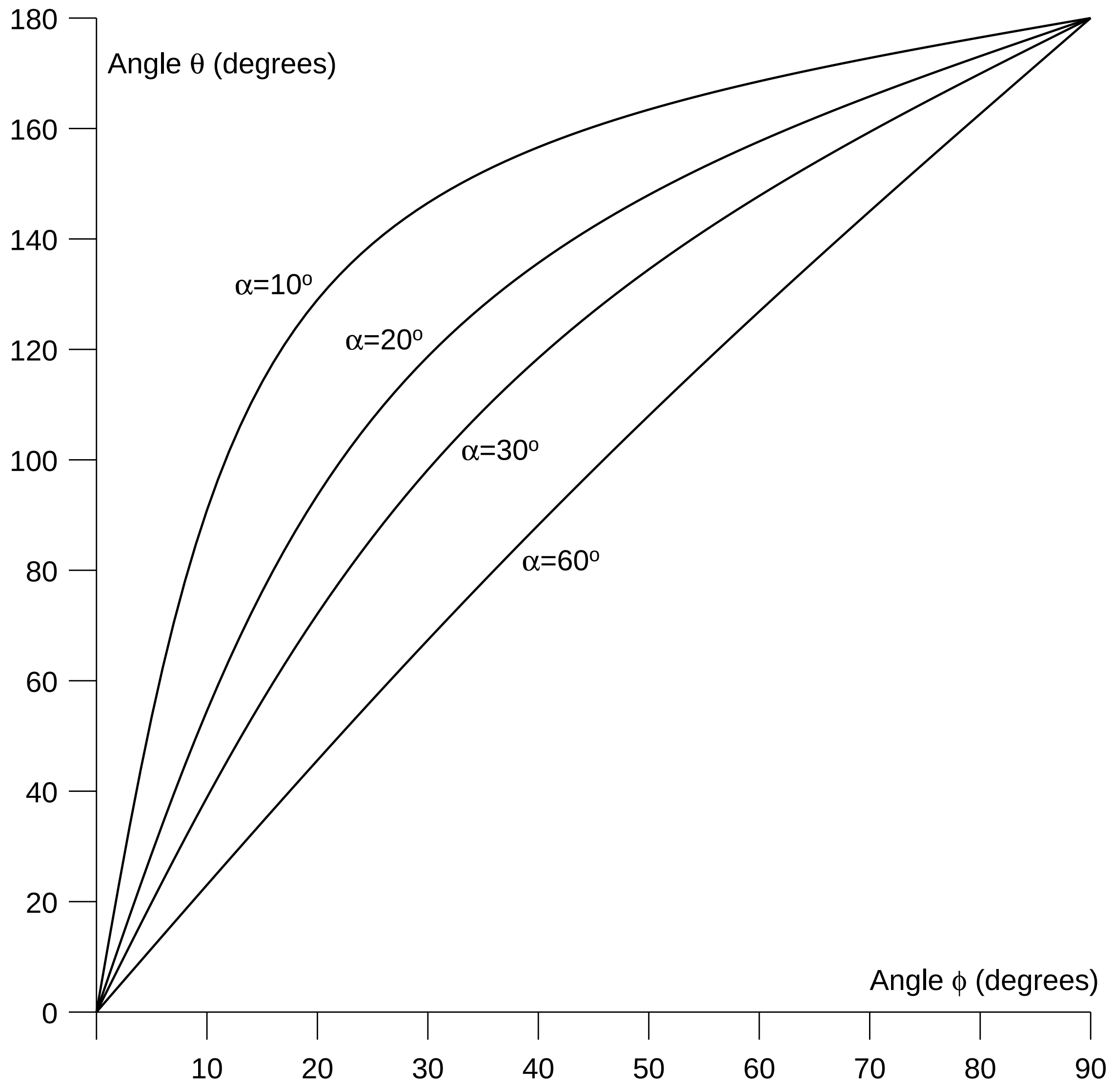


Figure 15: Variation of dyad angle θ with main fold angle ϕ

It has been seen that it is possible to obtain a simulation by handling the transforms applied to the carton faces. Reducing the face graph to a spanning tree imposes a hierarchy on the faces and allows the required driving motions to be established. The cuts effectively made in forming the tree correspond to dyads in which the angular positions of two adjoining faces need to be established. This can be done by imposing constraints which say that the edges on either of each cut must come together.

Such constraints can be dealt with within a constraint modelling environment. The necessary commands can be created automatically from a data file which defines the net of the carton (in terms of nodes and faces) and identifies where the cuts (and dyads) are deemed to lie. Since each dyad normally can have two solutions, care is needed to ensure that the correct one is found. This can be done by imposing bounds on the allowable angles for the faces or by imposing suitable values with which to start the search for the solution.

These ideas have been demonstrated successfully on two typical forms of carton, namely a tray and a skillet with a gable-end. In addition, the approach has been shown to work well with the more complicated folding pattern required for an origami bird.

Acknowledgements

The work reported in the paper was carried within the Innovative Design and Manufacturing Research Centre at the University of Bath. This is funded by the Engineering and Physical Sciences Research Council (EPSRC). Additionally the work relates to a project on reconfigurable packaging systems sponsored by the Department of the Environment, Food and Rural Affairs (DEFRA). This support is gratefully acknowledged.

References

[1] Hine D. Cartons and Cartoning. Leatherhead: Pira International; 1999.

[2] Hanlon JF, Kelsey RJ, Forcinio HE. Handbook of Package Engineering. Pittsburgh: Technomic Publishing Company; 1998.

[3] Leung SKS, Wong WK, Mok PY. Multiple-objective genetic optimization of the spatial design for packing and distribution carton boxes. Computers & Industrial Engineering 2008; 54: 899-902.

[4] van Roojen P, Baardman J, Hölscher J, Molenaar K. Special Packaging. Amsterdam: Pepin Press; 2004.

[5] Keay M. Cartoning machinery moves beyond the box. Machinery Update 2006; July/August: 31-37.

[6] Anon. Printed folding cartons. Huntingdon: Mainline Flatpacks Limited; http://www.mainlineflatpacks.co.uk/pdfs/cartons.pdf, accessed August 2008.

[7] Daniel J, Medland AJ, Mullineux G. Use of parametric modelling to understand the functional requirements for a reconfigurable packaging system. Proc. International Conference on Engineering Design (ICED) 07 2007; Ecole Centrale Paris: 9 pages on conference CD.

[8] Lu L, Akella S. Folding cartons with fixtures: a motion planning approach. IEEE Transactions on Robotics and Automation 2000; 16: 346-356.

[9] Khire MY, Madnaik SD. Folding cartons using low cost automation - a case study. Assembly Automation 2001; 21: 210-212.

[10] Liu H, Dai JS. An approach to carton-folding trajectory planning using dual robotic fingers. Robotics and Autonomous Systems 2003; 42: 47-63.

[11] Song G, Amato NM. A motion-planning approach to folding: from paper craft to protein folding. IEEE Transactions on Robotics and Automation 2004; 20: 60-71.

[12] Winder BG, Magleby SP, Howell LL. Kinematic representations of pop-up paper mechanisms. Proc. ASME 2007 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, IDETC/CIE 2007; Las Vegas; paper DETC2007-35505.

[13] Demaine ED, O'Rourke J. Geometric Folding Algorithms: Linkages, Origami, Polyhedra. Cambridge: Cambridge University Press; 2007.

[14] Matsushima K, Shimanuki H, Watanbe T. Computer-assisted paper wrapping with visualization. Lecture Notes in Computer Science 5093: Technologies for E-Learning and Digital Entertainment: Proc. 3rd International Conference, Edutainment 2008. Berlin: Springer; 2008; 114-125.

[15] Balkcom DJ, Mason MT. Robotic origami folding. Int. J. Robotics Research 2008; 27: 613-627.

[16] Stenberg N. A model for the through-thickness elastic-plastic behaviour of paper. Int, J. Solids and Structures 2003; 40: 7483-7498.

[17] Östlund M, Östlund S, Carlsson LA, Fellers C. Experimental determination of residual stresses in paperboard. Experimental Mechanics 2005; 45: 493-497.

[18] Berry C, Hicks BJ, McPherson CJ, Medland AJ, Mullineux G. Impact of environmental conditions on the performance of carton-board skillets. Packaging Science and Technology 2005; 18: 225-241.

[19] Hambli R, Richir S, Crubleau P, Taravel B. Modelling of sheet carton stapling using the finite element method. Int. J. Materials and Product Technology 2003; 19: 431-441.

[20] Hirano H, Kohuyashi T, Koduka I. Simulation for compressive behaviour of cartons. Japan Tappi Journal 2005; 59: 99-104.

[21] Sirkett DM, Hicks BJ, Berry C, Mullineux G, Medland AJ. Finite element simulation of folding carton erection failure. Proc. Instn Mech. Engrs - Part C: J. Mechanical Engineering Science 2007; 221: 753-767.

[22] Sirkett DM, Hicks BJ, Singh B, Mullineux G, Medland AJ. The role of simulation in predicting the effect of machine settings on performance in the packaging industry. Proc. Instn Mech. Engrs - Part E: J. Process Mechanical Engineering 2007; 221: 163-176.

[23] Liu H, Dai JS. Carton manipulation analysis using configuration transformation. Proc. Instn Mech. Engrs - Part C: J. Mechanical Engineering Science 2002; 216: 543-555.

[24] Nimnual R, Suksakulchai S. Virtual reality for packaging folding practice. Proc. International Conference on Control, Automation and Systems 2007, Seoul; 1011-1014.

[25] Demaine ED, Demaine ML, Hart V, Iacono J, Langerman S, O'Rourke J. Continuous blooming of convex polyhedra. Computing Research Repository (CoRR) 2009; 1-13; available at: http://arxiv.org/abs/0906.2461.

[26] Craig J. Introduction to Robotics: Mechanics and Control, 2nd edition. Englewood Cliffs: Prentice Hall; 2004.

[27] Leigh RD, Medland AJ, Mullineux G, Potts IRB. Model spaces and their use in mechanism simulation. Proc. Instn Mech. Engrs - Part B: J. Engineering Manufacture 1989; 203: 167-174.

[28] Mullineux G. Optimization scheme for assembling components. Computer-Aided Design 1987; 19: 35-40.

[29] Ge J-X, Chou S-C, Gao X-S. Geometric constraint satisfaction using optimization methods. Computer-Aided Design 1999; 31: 867-879.

[30] Michelucci D, Foufou S. Geometric constraint solving: the witness configuration method. Computer-Aided Design 2006; 38: 284-299.

[31] Mullineux G. Constraint resolution using optimisation techniques. Computers & Graphics 2001; 25: 483-492.

[32] Hicks BJ, Medland AJ, Mullineux G. The representation and handling of constraints for the design, analysis, optimization of high speed machinery. Artifical Intelligence for Engineering Design, Analysis and Manufacturing (AIEDAM) 2006; 20: 313-328.

[33] O'Sullivan B. Constraint-Aided Conceptual Design. London: Professional Engineering Publishing Limited; 2002.

[34] Au CK, Yuen MMF. A feature modeller for sculptured object modelling. Engineering with Computers 2003; 19: 1-8.

[35] Anderl R, Mendgen R. Modelling with constraints: theoretical foundation and application. Computer-Aided Design 1996; 28: 155-168.

[36] Wilhelms S. Function- and constraint-based conceptual design support using easily exchangeable, reusable principle solution elements. Artifical Intelligence for Engineering Design, Analysis and Manufacturing (AIEDAM) 2005; 19: 201-219.

[37] Hoffmann CM. Constraint-based computer-aided design. Journal of Computing and Information Science in Engineering 2005; 5: 182-187.

[38] Thornton AC, Johnson AL. CADET: a software support tool for constraint processes in embodiment design. Research in Engineering Design 1996; 8: 1-13.

[39] Fletcher R. Practical Methods of Optimization, 2nd edition. Chichester: Wiley; 1987.

[40] Perez-Garcia A, McCarthy JM. Kinematic synthesis of spatial serial chains using Clifford algebra exponentials. Proc. Instn Mech. Engrs - Part C: J. Mechanical Engineering Science 2006; 220: 953-968.

[41] Harbin R. Secrets of Origami: the Japanese Art of Paper Folding. Mineola: Dover Publications; 1997.

[42] Shimanuki H, Kato J, Watanbe T. Construction of 3-D paper-made objects from crease patterns. Proc. IAPR Conference on Machine Vision Applications (MVA IAPR), 2005; Tsukuba Science City; 35-38.