A Computational Model of Bounded Developable Surfaces with Application to Image-Based 3D Reconstruction

Mathieu Perriollat[1] and Adrien Bartoli[2]

[1] VI-Technology, Grenoble, France

Mathieu.Perriollat@gmail.com

[2] Université d'Auvergne, Clermont-Ferrand, France

Adrien.Bartoli@gmail.com – isit.u-clermont1.fr/~ab

Abstract

Developable surfaces have been extensively studied in computer graphics since they are involved in a large body of applications. This type of surfaces has also been used in computer vision and document processing in the context of 3D reconstruction for book digitization and augmented reality. Indeed, the shape of a smoothly deformed piece of paper can be very well modeled by a developable surface.

Most of the existing developable surface parameterizations do not handle boundaries or are driven by overly large parameter sets. These two characteristics become issues in the context of developable surface reconstruction from real observations. Our main contribution is a generative model of bounded developable surfaces that solves these two issues. Our model is governed by intuitive parameters whose number depends on the actual deformation and including the 'flat shape boundary'.

A vast majority of the existing image-based paper 3D reconstruction methods either require a tightly controlled environment or restricts the set of possible deformations. We propose an algorithm for reconstructing our model's parameters from a general smooth 3D surface interpolating a sparse cloud of 3D points. The latter is assumed to be reconstructed from images of a static piece of paper or any other developable surface. Our 3D reconstruction

method is well adapted to the use of keypoint matches over multiple images. In this context the initial 3D point cloud is reconstructed by Structure-from-Motion for which mature and reliable algorithms now exist and the Thin-Plate Spline is used as a general smooth surface model. After initialization, our model's parameters are refined with model-based bundle adjustment.

We experimentally validated our model and 3D reconstruction algorithm for shape capture and augmented reality on seven real datasets. The first six datasets consist of multiple images or videos and a sparse set of 3D points obtained by Structure-from-Motion. The last dataset is a dense 3D point cloud acquired by structured light. Our implementation has been made publicly available on `http://isit.u-clermont1.fr/~ab/Research/St-DR_v1p0.rar`.

Keywords: developable, surface, Structure-from-Motion, 3D reconstruction, capture.

1 Introduction

Most of the knowledge has been handed down in written records until recently. Those are currently supplanted by electronic documents, natively generated by computers, and come with indexing tools, searching facilities and fast worldwide sharing means on the internet. These useful tools are however only available for electronic documents. This has created the need for software to link paper documents and digital data. The task is not only to digitize printed paper, but to explore the full relationship between paper documents and computers. This is a complex relationship exploiting the emerging concept of augmented reality. For instance, readers usually prefer looking at real paper documents or books instead of reading on a computer screen. Obviously, the former does not make it directly possible to take full advantage of the computer in terms of searching facilities or translation capabilities. Bridging the gap between the real paper made objects and computers would allow one to combine the advantages of both worlds. The mixing of real scenes and computers is the object of study of augmented reality, where the goal in this context is to provide user-friendly multimedia tools overlaid over real scenes. The 'magic book' (Billinghurst et al., 2001) is one example of such applications where 3D reconstruction is needed. New imaging devices such as webcams, digital cameras or cameras embedded in mobile phones seem to be appropriate to these applications, since they are now cheap and popular. This underlines the need for image-based 3D reconstruction methods dealing with surfaces of paper material.

With the goal of using commonly available low-cost cameras, Structure-from-Motion appears as the method of choice to get 3D structure. We here assume that Structure-from-Motion can be used to obtain an initial sparse cloud of 3D points. Augmented reality however requires a much more accurate and dense model able to finely capture the object's surface and self-occlusions. Fitting a general smooth model such as a Thin-Plate Spline provides a continuous and dense surface, but is not as accurate as a model which would explicitly capture true physical constraints: the fact that the underlying object's shape is developable. Our main contribution is a model of bounded developable surfaces

that extend our previous work in this area (Perriollat and Bartoli, 2007). Once initialized from the estimated general smooth surface and iteratively refined to fit the images, our model provides an accurate and stable estimate of the object's 3D surface.

Our model is governed by a set of parameters whose number depends on the actual deformation. In other words, the number of model parameters increases with the complexity of the actual deformation. For instance, a flat surface will have very few parameters while a 'creased surface' will have many more. The key idea behind our model is to generate the surface by bending a plane around rulings with some bending angles. Our model parameters are thus *(i)* a 2D shape that defines the boundaries of the flattened object being modeled, and *(ii)* a set of rulings and bending angles. This parameter space can be easily restricted or constrained to match special deformations, for instance for the page of a book which is strongly constrained by the binding. In comparison with most of the previously proposed models, ours naturally handles the object shape: it generates a surface *and* its boundary.

We validated our model by proposing algorithms to fit it to real data. The input data must include an estimate of the object's surface, which is generally not accurate enough for applications such as augmented reality, but allows us to initialize our model's parameters. In the specific case of 3D reconstruction from images, we use Structure-from-Motion to get a 3D point cloud (Hartley and Zisserman, 2004) and a Thin-Plate Spline as general smooth surface model. In a monocular framework the scene must thus be static while dynamic scenes can be handled by using multiple synchronized cameras. In our current implementation, the whole reconstruction pipeline is automatic. The user just indicates the part of the object of interest as visible in one of the input images. Most of the other developable surface reconstruction algorithms require dense and accurate 3D or range data (Chen et al., 1999; Peternell, 2004) or provide results from monocular images using assumptions on the lighting conditions (Courteille et al., 2004). In contrast, our method is easy to set up (there is no need for a complex imaging device or particular lighting conditions) and robust to phenomena such as paper self-occlusion.

Paper organization. We review previous work on developable surfaces and their 3D reconstruction in section 2 and give our notation in section 3. Some background on developable surfaces is given in section 4. Our bounded developable surface model is given in section 5. Our 3D fitting algorithm is described in section 6 and results on real images are reported in section 7. Eventually, we give our conclusions and discuss future work in section 8.

2 Preliminaries and Previous Work

2.1 Preliminaries

The first step in the modeling process is to study the type of deformation to be represented. It is highly dependent on the eventual use of the model and its targeted accuracy. Our goal is to represent the overall 3D shape of a deformed

paper: we do not attempt to model the elastic behavior of paper sheets which is more related to the study of internal forces (see *e.g.* (Houle and Sethna, 1996).) The paper is modeled by an inextensible object with zero thickness, *i.e.* by a surface. Within this setting, the set of possible deformations is still extremely wide and too complex to be embedded in one single model. We assume that the surface is smooth. This means that there is no strong fold on the object surface. This is a commonly used assumption, which holds for most cases of interest.

2.2 Previous Work on Developable Modeling

A paper-like surface is considered as a smooth and inextensible surface that can be isometrically mapped onto a plane. This is mathematically described by developable surfaces (simply called *developables* (Struik, 1961).) This kind of surfaces and its parameterization have been extensively studied in the computer graphics and computer aided design communities since it is well adapted to the design of many manufactured products, such as ship hulls and clothes. Indeed, the crafting process of such surfaces is simple: one has to compute and cut the flat boundary in the material and bend it to get the final object. It is also used in architectural design (Glaeser and Gruber, 2007; Liu et al., 2006), for instance to design rounded glass buildings. The main goal for the graphics community in this area is to provide intuitive design methods and possibly simulation algorithms. Estimating this kind of shapes from observations such as images or 3D data is more related to reverse engineering.

There are three main ways to describe a developable: the direct continuous form, the continuous form in the dual space and the discrete approximation. We review all of them below. From a mathematical point of view, a developable is a 3D surface, *i.e.* an unbounded manifold in $\mathbb{R}^3$. Since we target 3D object reconstruction applications, the surface *and* its boundary are needed. To make the distinction between these two notions, we refer to *developable* for the infinite mathematical surface and to *bounded developable* for the pair including the surface and a boundary.

2.2.1 The Direct Continuous Form

This formulation treats the developable as a constrained surface in $\mathbb{R}^3$. Previous work using this formulation is focused on combining common design tools and the constraints to provide intuitive developable surface design methods. One of the classical tools in computer graphics is the surface patch, for instance the Bézier and the B-spline patches. Several studies tackled the problem of constraining these patches to be developable. Actually, to define a developable, two 3D boundary curves are needed. Early methods were quite restrictive (Aumann, 1991; Lang and Röschel, 1992). For example they constrained these curves to lie in parallel planes (Aumann, 1991). It has been shown (Chu and Séquin, 2002) that given the first 3D curve, only five other degrees of freedom must be chosen to completely define the developable patch. In accordance with this property (Aumann, 2003) proposes an algorithm to build developable Bézier patches, extended to an arbitrary degree in (Aumann, 2004). Similar works have been done with B-spline surfaces (Fernández-Jambrina, 2007). More recently, (Bo and Wang, 2007) built a generative model using a prop-

erty of geodesics on a developable surface. This formulation is efficient and allows one, for instance, to flatten the deformed shape. Other direct models use several simple pieces of developable put together to form the global surface (Leopoldseder and Pottmann, 1998; Sun and Fiume, 1996). It makes sense to do that with, for instance cones since the cone of revolution is the osculating quadric of a developable. It thus gives the order two approximation to the surface, but lacks a parameterization with a minimal set of parameters. The continuous form is able to exactly represent developables and bounded developables. It is even possible to restrict the representation to its order two approximation. The problem with these methods comes from their parameterization, which might not be minimal or even physically meaningful.

2.2.2 The Continuous Form in the Dual Space

The second possibility to describe developables is based on the tangent plane property and is also continuous. A developable can be seen as the envelope of a one dimensional family of tangent planes (Hilbert and Cohn-Vossen, 1952). It means that it is equivalent to know the developable or the one dimensional family of tangent planes. This is called the dual representation. It has been mixed with NURBS, leading to developable NURBS surfaces in (Pottmann and Wallner, 1999). (Peternell, 2004) uses Laguerre geometry to represent the one dimensional family of planes on the Blaschke cylinder. Although these methods are powerful to represent developables, they do not allow one to directly parameterize bounded developables. They are thus not well adapted to the reconstruction of real objects.

2.2.3 The Discrete Approximation

The last category of models is the discrete model, so-called *strip approximation* of developables. The developable here is a strip of planar quads. The deformation is controlled by the rotation between quads along the shared edges. This model was presented in (Pottmann and Wallner, 2001). It has been embedded in a simulation framework based on internal physical constraints in (Kergosien et al., 1994). Recently, it has been used in (Liu et al., 2006) with a subdivision algorithm to smooth the generated surface. The basic parameterization of this formulation describes the shape of the quads and the rotations between them. It uses a minimal set of parameters to control the entire deformation but may lack smoothness. By contruction, this model is more dedicated to bounded developables than to developables.

2.3 Previous Work on Developable Surface Reconstruction

The 3D reconstruction of developable surfaces has been treated from several points of view. In computer graphics, (Chen et al., 1999) and (Peternell, 2004) come up with algorithms to estimate developables from 3D point clouds. There are strong limitations to use them directly for paper shape recovery from images. First of all, they deal with

3D points, whereas only 2D projections are available in images. More problematic is that they require quite dense and accurate sets of points which are not always possible to obtain with real images and passive vision.

Flattening paper images is an active research field and several methods were proposed. Most of them requires a highly controlled environment. A formulation of shape-from-shading for a book page was proposed in (Courteille et al., 2004). The page model is a generalized cylinder, so it does not handle complex deformations. The method uses shading and thus requires special lighting conditions, which makes it dedicated to book scanning. Moreover, real-time augmented reality applications are not possible since interacting with the paper creates cast shadows and defeats the algorithm. The results are however accurate and lead to a good correction of the skew in documents. One property of bounded developables is that they can be entirely determined from the contour of the sheet as shown in (Gumerov et al., 2004). In this paper this property is used to synthetically generate paper deformation and to flatten deformed paper. The method does not handle external- or self-occlusions since it needs the whole contour to be visible in the input image. A discrete paper model and an energy minimization process are used by (Liang et al., 2005). It detects rulings using textual information. A 3D scanner combined with a high resolution image is used in (Brown et al., 2007) leading to accurate flattening.

Although all these methods work with only one image, they are not compatible with augmented reality applications since either the imaging setup is highly constrained or they require a highly accurate general smooth surface from which to extract the developable, prohibiting the use of low-cost imagery.

3 Notation

The set of real numbers is written $\mathbb{R}$, $\frac{v}{\|\cdot\|}$ is vector normalization, $u \times v$ is vector cross product, sign is the sign function, $x'(t)$ is the first derivative and $x''(t)$ the second derivative of $x(t)$. Indices f, c and a refer to 'flat', 'continuous' and 'discrete' respectively. The arc length (al) coordinate is written s, $\mathcal{S}$ is a 3D curve, p a geodesic and d a ruling's direction. As an example of notation, p_c is a geodesic curve on a continuous surface and $\|p_c''(s)\|^2$ is the squared norm of its second derivatives.

4 Developable Surfaces

This section reviews the mathematical formulation of developables and their properties. The boundary problem, which is also important in the modeling process is not addressed here but in section 5 where our model is presented.

4.1 Background

Two surfaces can be isometrically mapped to each other if and only if the corresponding points share the same Gaussian curvature. The Gaussian curvature is defined as the product of the principal curvatures, which are the extremal curvatures of the set of curves passing through a surface point. Since a straight line has zero curvature, the Gaussian curvature of the plane vanishes everywhere. As we mention above, developables are isometric to the plane and therefore their Gaussian curvature vanishes everywhere (Hilbert and Cohn-Vossen, 1952). This explains why a sphere, whose Gaussian curvature is constant and positive, cannot be mapped to a plane without distortions. In other words, the sphere is not developable as opposed to a cylinder, which can be unrolled to a plane and is thus developable.

Obviously, a plane is developable. Excluding the plane, the other developables belong to the set of ruled surfaces and are characterized by an extra condition (Pressley, 2005), as described below. The ruled surfaces are the surfaces generated by the motion of a straight line in space. So for each surface point, there exist a straight line passing through this point and that lies on the surface. This line is called a *ruling*. The equation of a ruled surface is:

$$X(t, v) = \mathcal{S}(t) + v\, d_c(t), \quad (t, v) \in \mathbb{R}^2 \quad d_c(t) \neq 0. \tag{1}$$

The space curve $\mathcal{S}$ is called a directrix of the surface, d_c is a vector field that gives the direction of the rulings. It defines a line pencil: for each $t = t_0$, $X(t_0, v)$ is a 3D line. The Gaussian curvature vanishes everywhere on a ruled surface if and only if:

$$\det\left(\mathcal{S}'(t), d_c(t), d_c'(t)\right) = 0. \tag{2}$$

This equation constrains the tangent plane to be constant along ruling. This makes the ruling be one of the principal curves. Since it is a straight line and so has zero curvature, the Gaussian curvature vanishes.

These constrained ruled surfaces are developable. Actually the converse theorem is true, and all the developables can be expressed by these equations (Hilbert and Cohn-Vossen, 1952). Consecutive rulings intersect to form a curve, named the edge of regression (Struik, 1961). The rulings are tangent to this curve. This curve does not belong to the surface because it is the locus of its singular points. Conversely, the surface generated by the tangents of a space curve is a developable, the so-called tangent developable. There are two distinct parts of a tangent developable, placed on each side of the edge of regression. They meet on the edge of regression. The tangent planes of a developable are constant along the rulings. Actually they are the osculating planes of the edge of regression (Hilbert and Cohn-Vossen, 1952). Therefore a developable can be generated by the envelope of this one dimensional family of tangent planes. Conversely, the envelope of a one dimensional family of planes is a developable. This property leads to the dual representation of developable surfaces, *i.e.* instead of the direct parameterization of the surface in $\mathbb{R}^3$, the one dimensional family of planes is defined.

The previous paragraph links the ruled surfaces to the developables and gives other representations of developables, namely the tangent developable and the dual representation. There are special developable surfaces for which the edge of regression is a point (the generalized cones: all the rulings meet at the vertex of the cone) or is not defined (the generalized cylinder: all the rulings are parallel and thus meet at infinity.) However they can still be parameterized as a one dimensional family of tangent planes.

4.2 Some Properties

An isometry between two surfaces preserves the geodesic property of curves (Pressley, 2005): a geodesic curve onto the first surface is mapped to a geodesic curve onto the second surface with the same length. This is used in (Bo and Wang, 2007) to intuitively construct developables: the deformation of the flat shape is controlled by a 3D geodesic curve whose flat correspondence is known. The ruling directions $d_c(s)$ (s is the arc length along the geodesic) are computed from the 3D geodesic curve $p_c(s)$ using equation (3), and the object boundary is inferred from the flat shape, see figure 1 (middle):

$$d_c(s) = \frac{p_c''(s) \times p_c'''(s)}{\|p_c''(s)\|^2}. \tag{3}$$

In the context of modeling, developables are used to represent the object surface. However, it has been shown that some developables have singular points (the edge of regression is the locus of those.) Moreover, these singular points do not belong to the surface and therefore are not developable. Real objects of interest can be flatten onto the plane whatever the deformation, so their surface do not contain any singularity. It means that the edge of regression of the surface that models a real object must be outside the object boundary. The edge of regression of a bounded developable is thus outside the boundary. For instance, this constraint is enforced by the algorithm in (Bo and Wang, 2007). This problem is natively handled by the model we propose in the next section. This reveals a main drawback of dual space parameterization of developable surfaces in the context of object surface modeling: with this formulation, there is no simple expression of the object boundary.

Developable surfaces have been described by the set of surfaces that can be flatten onto the plane without distortion. This leads to the constrained ruled surface formulation. However, real deformations of flat shapes often involve several pieces of developable surfaces, for example the page of a book may have a global shape whereas the corners are independently bent, see figure 2. There are two ways to join developables: by defining a common directrix and by linking them via a planar region, as shown in (Do Carmo, 1976).

The last possibility to describe a developable is to use the strip approximation. It is composed of a strip of planar quads, see figure 1 (right). One could think of using triangles instead of quads, but this has been shown not to be a valid representation (Wang and Tang, 2005). In this model, the rulings are represented by the shared edges between quads. The edge of regression is a polygone.

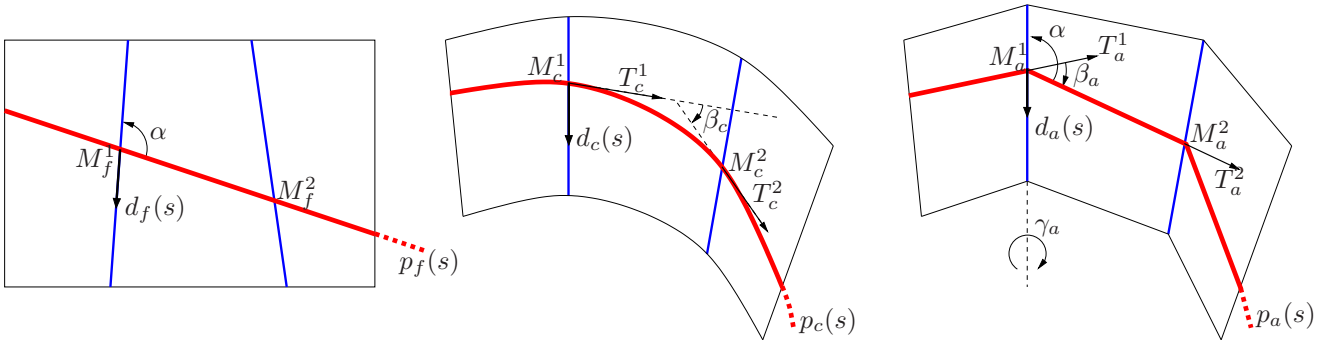


Figure 1: Curvature preserving continuous surface approximation. The red (thick) curve is a geodesic, the blue (thin) ones are rulings. (left) The flat shape: the geodesic is a straight line. (middle) The geodesic is a curve on the continuous model. (right) On the discrete strip, the geodesic is a polyline.

A relation between the continuous surface and its discrete approximation can be derived from the curvature of a geodesic curve. The deformations of the discrete strip can be parameterized by rotation angles along the rulings, the angle γ_a in figure 1 (right.) The problem is to relate these angles to the curvature of the geodesic on the continuous form such that if the number of quads of the strip increases, the discrete surface tends to the continuous one. This can be done by using the geometric interpretation of the curvature of a geodesic. Figure 1 illustrates the approximation method. First of all the knowledge of a geodesic on the surface allows one to compute the rulings (Bo and Wang, 2007). Thus the angles α between the rulings and the geodesic is known. The curvature κ at the point $M_a^1 = p_c(s_1)$ is the limit of the ratio between the angle of contingence β_c (it is the angle formed by the tangents of the first point T_c^1 and a second point T_c^2) and the length of the arc $M_c^1 M_c^2$ when $M_c^2 = p_c(s_2)$ tends to M_c^1 (for more details see (Struik, 1961)).

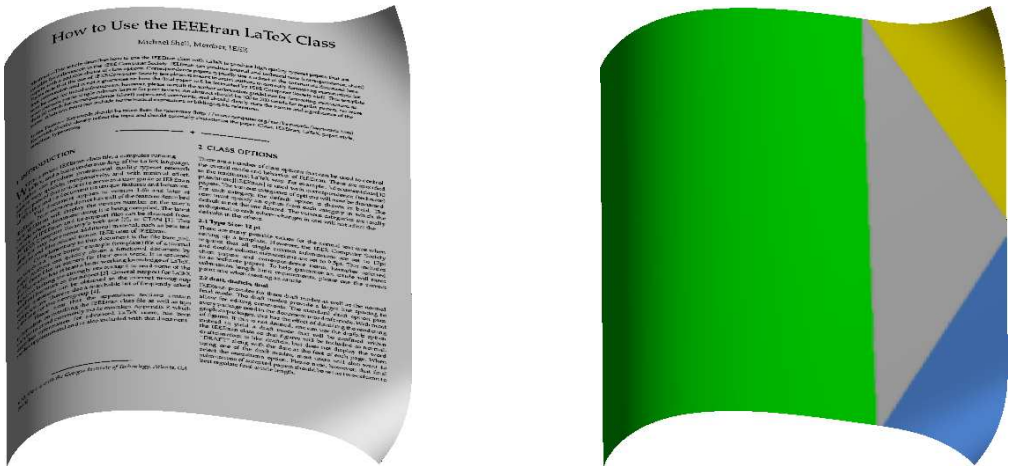


Figure 2: Synthetic deformation of a book page. (left) The textured view of the page. (right) Several pieces of developable surfaces are involved in this deformation (colored patches), linked together by the gray flat patch.

5 Modeling Bounded Developable Surfaces

One application goal for our model is fitting a developable to real data, in particular image-based 3D reconstruction. Our model must thus be comprenhensive (represent all possible bounded developables), with as few parameters as needed, and easily handle special cases by restricting the parameter space (for instance the constraints implies by the binding of a book.)

5.1 Principle

5.1.1 Ruling-Based Generation

The method we propose is based on the discrete model of developables (see figure 1.) The main idea is to use a discrete set of rulings instead of a continuous formulation. This leads to a piecewise planar surface. The constraint on Gaussian curvature in (2) turns into a formulation in terms of bending angles. The rulings are chosen such that they do not intersect each other to guarantee that the regression line is off the object shape.

Given the planar boundary shape, generating a surface using our model has two main steps, as figure 3 illustrates. First, we add extra rulings by interpolating the positions and the angles of the guiding rulings. Guiding rulings will be used in our model to specify where the flat shape has to be bent in the surface generation process. They are introduced more formally in the next section. The number of extra rulings directly controls the smoothness of the generated surface (see figure 4 for an example.) Second, the flat surface is bent along all the rulings. It is guaranteed to be admissible in the sense that the surface underlying the generated surface is necessarily a bounded developable.

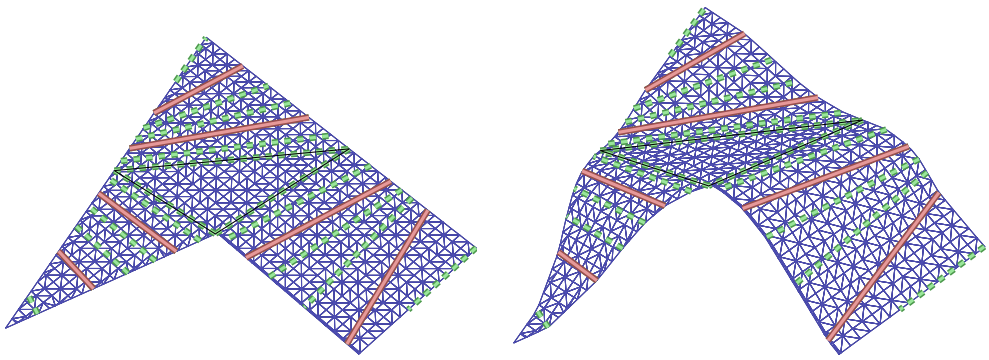


Figure 3: Surface generation. (left) Flat surface with guiding rulings (thick and pink) and extra rulings (thin and green.) (right) Surface folded along the guiding and extra rulings.

5.1.2 Balancing Model Complexity and Surface Smoothness

Wr can make two fundamental observations about the model behavior. The density of rulings influences:

(i) the smoothness of the surface: the higher the number of rulings, the smoother the surface.

(ii) the model complexity: the higher the number of rulings, the more complex the model.

Observation *(i)* leads us to consider a huge number of rulings so as to generate a smooth and accurate surface. To deal with observation *(ii)*, in order to avoid an overly large number of parameters, we propose to control a subset of the rulings and to interpolate the other ones. They are respectively called *guiding rulings* and *extra rulings*. This has the advantage to generate a smooth surface with a small set of parameters. The aspect of the final surface depends on both the guiding rulings and the interpolation process. Figure 4 illustrates the effect of the proportion between the number of guiding and extra rulings. The surface generated by 6 guiding rulings and 12 extra rulings is an interesting tradeoff: there are enough parameters to capture all deformations since the smoothness given by the extra rulings significantly decreases the error, and adding guiding rulings does not really improve the accuracy.

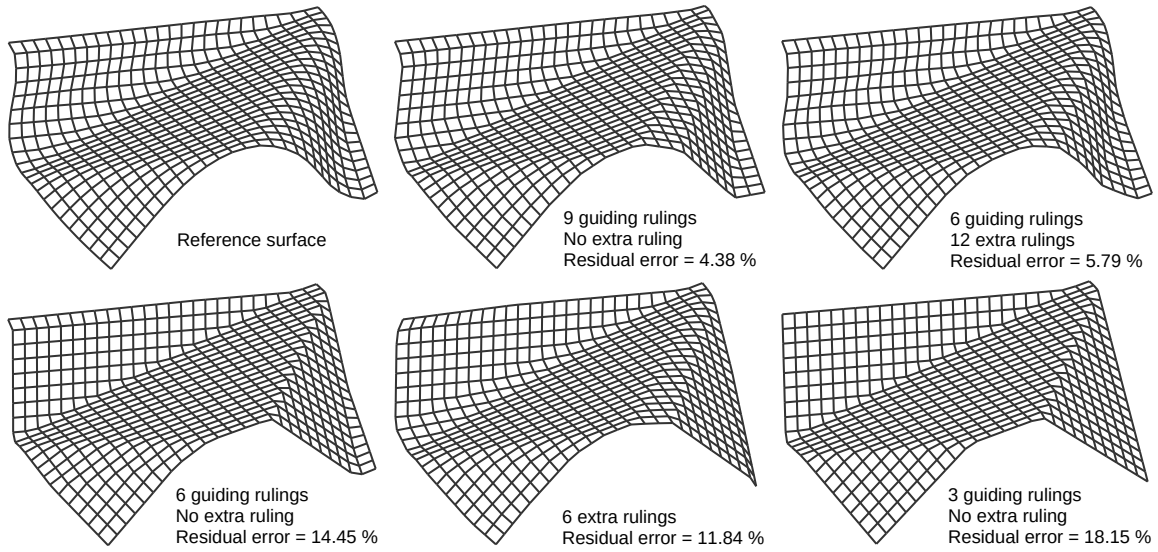


Figure 4: Surface generation behavior. The reference surface is estimated by our model with a varying number of parameters. The residual error represents the mean distance to the reference surface, it is given as a percentage of the meshgrid step.

5.1.3 Internal Consistency Constraints

To satisfy the edge of regression constraint of bounded developables, the rulings must not intersect each other inside the object boundary. Moreover, to make the interpolation process work with a non convex boundary, a ruling is valid if the segment joining the two boundary-ruling intersections is entirely on the surface. These two constraints are satisfied by our model, they are illustrated in figure 5.

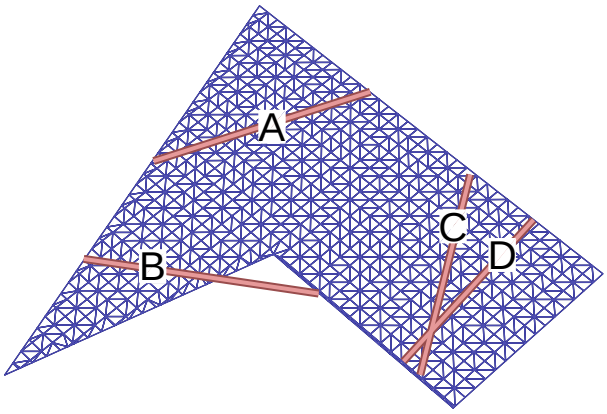


Figure 5: Ruling validity examples. Ruling A is valid. Ruling B is not valid since it gets outside the surface. Rulings C and D are not valid because they intersect each other on the surface.

5.2 Parameterization

Our model has a parameter set into which we distinguish two parts. The first part describes the shape of the flat surface. The second part controls the deformations. The shape is defined by a planar curve, often a planar polygon, and gives the boundary of the object. The deformations are parameterized by the guiding rulings and their bending angles. Since each ruling intersects the boundary curve at exactly two points, a minimal parameterization of the rulings is the arc length of these two points along the shape curve. To build a realistic surface, the rulings must not intersect each other on the surface. This is enforced by constraining the arc lengths of the rulings. More details are given in §5.3. The deformations are eventually defined by coupling each ruling with a bending angle, choosing the number of extra rulings and the interpolation functions. Table 1 summarizes the model parameters. The model has $2 + S + 3n$ parameters, with S the number of parameters describing the surface boundary (for instance, width and height in the case of a rectangular shape) and n the number of guiding rulings.

Parameters	Description	Size
n	number of guiding rulings	1
n_e	number of extra rulings	1
$\mathcal{S}$	surface boundary parameters	S
s_A	arc length of the guiding rulings	$2n$
θ	bending angles of the guiding rulings	n

Table 1: Summary of our model parameters. (top) Discrete parameters (kept fixed during nonlinear refinement.) (bottom) Continuous parameters.

5.3 Surface Generation

We bring together rulings that belong to the same 'bending region' on the paper. We define a region as a set of consecutive rulings. Two rulings are consecutive if both of their endpoints are. Figure 6 (top left) shows the labeled guiding rulings on the flat surface. We plot the arc lengths of guiding rulings onto a graph, represented in figure 6

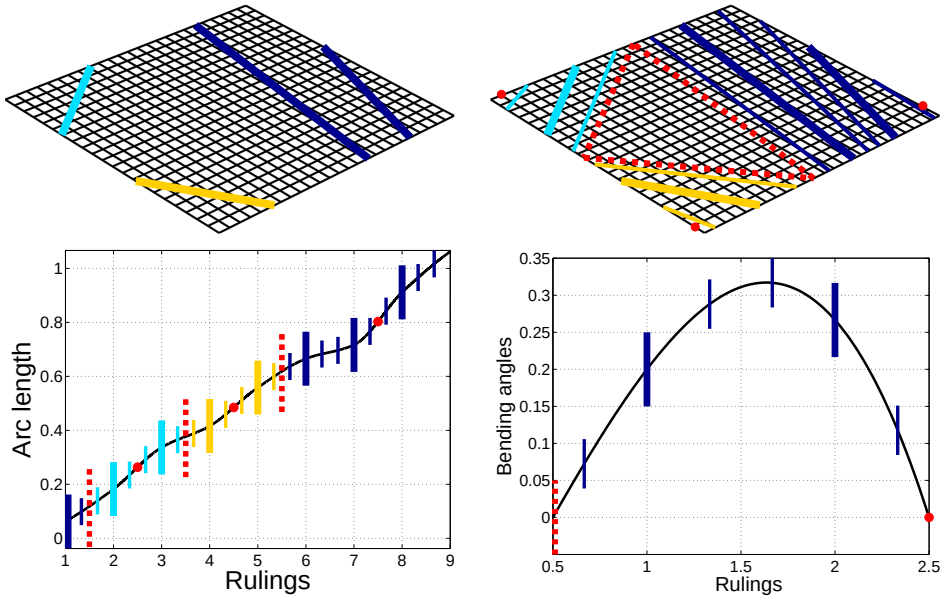


Figure 6: Interpolation process. (top left) Flat surface with the labeled guiding rulings. Three regions are defined. (bottom left) Ruling interpolation process. Each vertical line represents a ruling. The black curve is an increasing interpolation. (top right) Flat shape with rulings. (bottom right) Bending angles interpolation process. The black curve is an interpolation. The thick lines are the guiding rulings. The thin lines are the extra rulings. The dashed red lines are the region limits. The red dots are the region extremities.

(bottom left) and compute an increasing interpolation function passing through these points. The monotonicity constraint is important to guarantee that rulings do not intersect on the shape. We use a piecewise cubic Hermite interpolating polynomial as interpolation function (de Boor, 2001). This function is resampled to get extra rulings, limits and extremities of each regions. Region limits are chosen in the middle of two consecutive rulings having different labels. The result of resampling is visible in figure 6 (top right.) The interpolation of the bending angles is region-dependent. For each region, we represent the bending angles of the guiding rulings on a graph, see figure 6 (bottom right.) We compute an interpolation function (a piecewise cubic Hermite interpolating polynomial) passing through the bending angles with the following side conditions to ensure continuity between regions: the bending angles are null at the limit and the extremity of the region. We get the bending angles of extra rulings by resampling this curve, previously rescaled to take into consideration the increased number of bendings. This rescaling is nothing but a division by the ratio of the total number of rulings against the number of guiding rulings. Since all rulings have been computed, we split the shape into cells, each cell being a region between two consecutive rulings. With this representation, folding the flat surface is done by rotating and translating each cell. The rigid transformations are formed by composing those induced by each ruling starting from a reference cell. Figure 7 shows the result of this last step. Table 2 gives an overview of the surface generation process.

The asymptotic surface is the continuous surface obtained when the number of rulings tends toward infinity. It

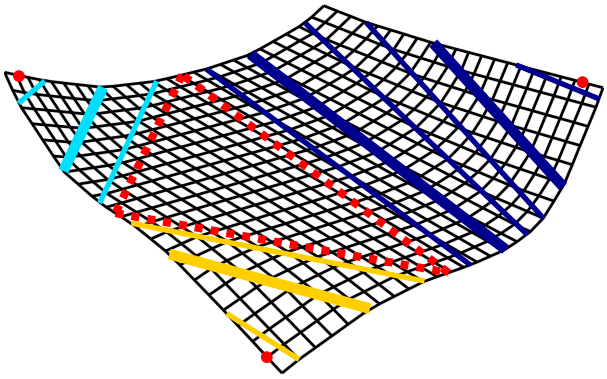


Figure 7: Bent shape with rulings. The thick lines are the guiding rulings. The thin lines are the extra rulings. The dashed red lines are the region limits. The red dots are the region extremities.

Surface generation process
1. Define the shape boundary on the flat surface
2. Gather the rulings into regions
3. Interpolate the ruling positions and their angles
4. Resample the interpolating functions to get the extra rulings
5. Fold the flat surface

Table 2: Overview of the surface generation process.

is possible to derive it for our model though it is not trivial. We describe here the main ideas to achieve this. As mentioned in (Bo and Wang, 2007), a developable is entirely defined by a geodesic crossing all the rulings. Using this property, it is sufficient to evaluate the continuous asymptotic curve of a geodesic to get the asymptotic surface. The *Fundamental Theorem* (Struik, 1961) states that a curve is defined, up to a rigid motion, by the knowledge of its curvature and its torsion. The idea is thus to evaluate the curvature and the torsion of the geodesic on the discrete model using their geometric interpretations and to get their continuous form by making the number of rulings tend toward infinity. The 3D curve is then computed by solving the so-called *natural equations* which give the curve from its curvature and its torsion. There is no simple equation to describe the asymptotic surface of our model.

5.4 The $al - al$ curve

The rulings are defined by the arc length of their intersections with the object boundary. Some interesting properties come from this representation. They are used in our reconstruction algorithm to detect the rulings and to ensure that the edge of regression is outside the object.

One can represent the rulings in $\mathbb{R}^2$ by a curve mapping both arc lengths of the rulings, the $x-$coordinate being the lowest arc length of the two intersections. We call this representation the $al - al$ curve. This curve is piecewise continuous, breakpoints appearing when there are several bending regions, see figures 13 for an illustration. To prevent

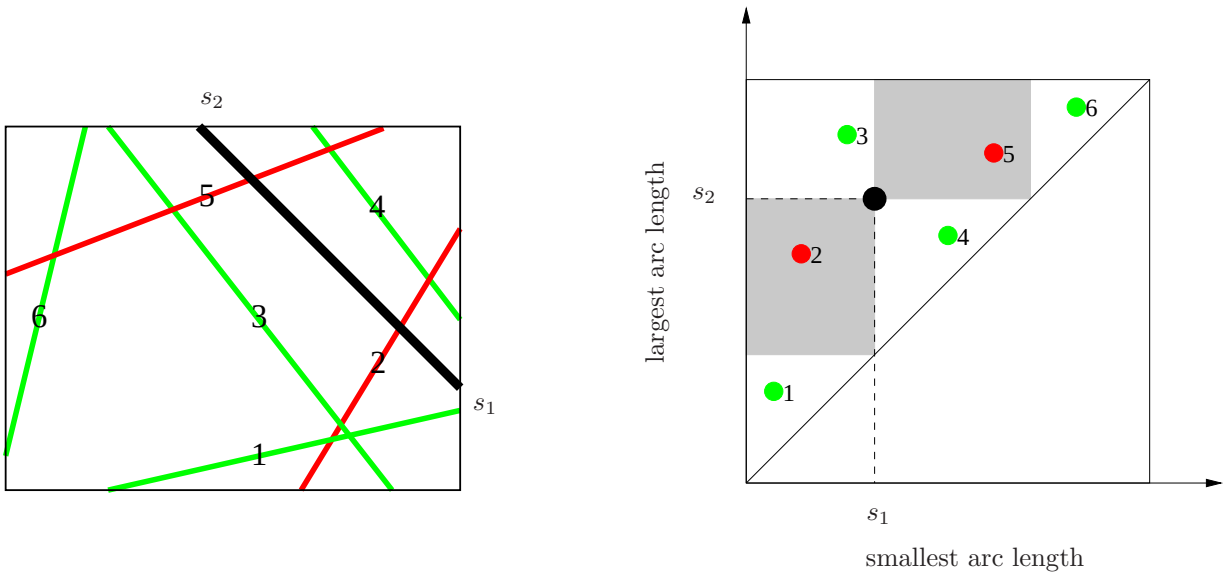


Figure 8: Constraints on the ruling positions in the $al - al$ curve. (left) The flat shape with the rulings. (right) The $al - al$ curve. The gray zones on the $al - al$ curve are dismissed zones to avoid the edge of regression to be into the shape. The green (thin light) rulings are valid and the red rulings (thin dark) violate the edge of regression contraint.

the edge of regression to be on the shape, it is intuitive that the curve must be decreasing, at least piecewise. The constraint on the edge of regression can be stated by the following question: given a ruling, where is the set of rulings that cross it on the shape? Figure 8 illustrates this property. Let s_1 and s_2 be the two arc lengths of a ruling such that $s_1 \leq s_2$. The rulings that cross this ruling are those which have one arc length between s_1 and s_2 and the other one either less than s_1 or greater than s_2. This leads to the property that the curve describing the position of the rulings in a region must be monotonic and decreasing.

5.5 Mapping the Flat to the Deformed Surface

With our discrete model, the flat shape is splitted into cells, each cell being a region between consecutive rulings. The first step is to identify in which cell lies the point to transfer. The second step is to apply the rotation and translation defined for that cell to the point.

6 Fitting Bounded Developable Surfaces

The goal in this section is to describe an algorithm to fit our bounded developable model to some real data. The model is initialized from some general smooth surface fitted to some sparse 3D point cloud. This general smooth surface need not be highly accurate. Our algorithm focuses on the case where the 3D point cloud was obtained from keypoint matches over multiple images by Structure-from-Motion, and uses a Thin-Plate Spline as a general smooth surface model. It can however accomodate general smooth surfaces obtained by other types of 3D sensors. In Structure-from-

Motion the optimal criterion being minimized in 3D reconstruction is the reprojection error. We iteratively refine our model's parameters by minimizing the reprojection error via model-based bundle adjustment.

6.1 Structure-from-Motion

The 3D point cloud is generated by triangulating point correspondences between several views (Hartley and Zisserman, 2004). The method depends on the number of views. In the stereo case, the camera pair is first calibrated. Keypoints are then detected into the images (Lowe, 2004) and the points are matched together, using the epipolar geometry. Eventually the points are triangulated to get the 3D point set. 3D reconstruction from videos where the camera moves around a static scene (here, the deformed paper) was studied in (Pollefeys et al., 2004). This method provides camera calibration and pose, correspondences between images and 3D points. At this stage, a sparse set of 3D points and calibrated cameras have been evaluated.

6.2 General Smooth Surface Reconstruction

For the initial surface reconstruction, a three step process is defined. Each step is explained in detail in a paragraph below. Steps 1 and 2 detect and select from the 3D point cloud those points which lie inside the object and can thus be used in the reconstruction process. They also provide a 2D parameterisation. Step 3 fits the general smooth surface.

A 3D surface can be viewed as a manifold of dimension 2 in the $\mathbb{R}^3$ space. An explicit representation f of the surface is a function that maps a 2D parameterization (u, v) to the 3D points of the surface:

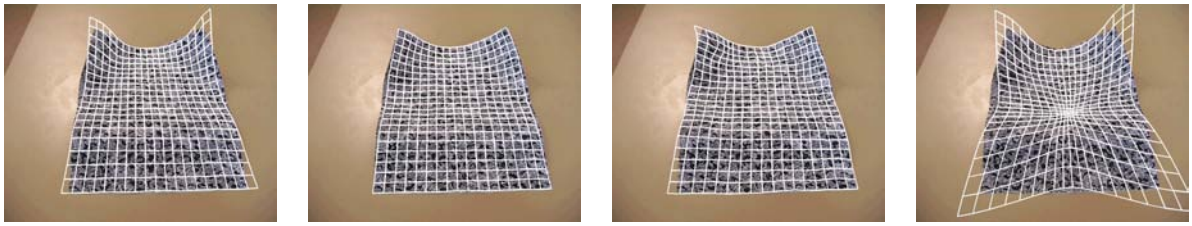
$$f(u,v) = \begin{pmatrix} f_x(u,v) \\ f_y(u,v) \\ f_z(u,v) \end{pmatrix}.$$

The first step is to evaluate the planar parameterization of the 3D points. Then the interpolating function from this planar parameterization to the space points gives the continuous function.

6.2.1 Dimensionality Reduction

The planar parameterization is given by a dimensionality reduction algorithm. Several algorithms exist. They differ by the quantity they minimize (or maximize) in the lower dimension space. For instance, *Curvilinear Component Analysis* (CCA) (Demartines and Hérault, 1997) and *Isometric feature Mapping* (Isomap) (Tenenbaum et al., 2000) are based on the preservation of the neighborhood and the Euclidean distances between points whereas *Maximum Variance Unfolding* (MVU) (Weinberger and Saul, 2005) maximizes the variance of the data in the lower dimensional

space. Linear methods have been proposed too, such as Locally Linear Embedding (LLE) (Roweis and Saul., 2000). The results obtained using these methods on a real example are shown in figure 9.



LLE CCA Isomap MVU

Figure 9: Flattening of the initial general smooth surface. LLE: the planar parameterization is not accurate leading to a bad boundary estimate. CCA: the boundary is correctly estimated and the surface is almost inextensible. Isomap: the generated surface is less smooth than the one obtained with the CCA algorithm due to a less accurate planar parametrization. MVU: the surface is extensible since the planar parameterization does not preserve distances.

The best results are achieved by CCA and Isomap. Actually these methods are well-suited to planar parameterization estimation due to the criterion they minimize. The neighborhood preservation expresses that the points belong to a surface: their respective position is preserved. Moreover the Euclidean distance preservation reflects the inextensibility of the surface. The other methods are less accurate. The examples presented in this paper are obtained with CCA.

6.2.2 Point Selection

This step detects the points that have to be removed. This is possible with a knowledge about the surface shape: if the surface boundary is known in the planar parameterization, the points out of the boundary are from the background.In our implementation, we let the user to mark a polygon in one of the images. Since each point in the planar surface parameterization has two coordinates, each vertex of the polygon can be readily transferred to the planar surface parameterization space. Figure 10 illustrates the point selection result.

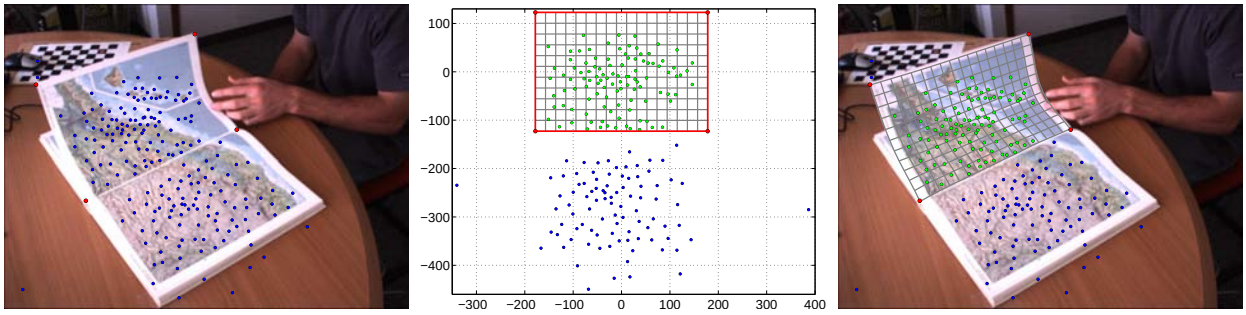


Figure 10: Initial interpolation and selection of points on the object. (left) Reprojection of the triangulated points in one image of the sequence, the red points are the user selected object's polygon vertices. (middle) Result of the nonlinear dimensionality algorithm: two sets of points appear, the vertices form a rectangle into which the points on the sheet lie. (right) Reprojection of the initial surface.

6.2.3 Surface Reconstruction

The initial surface is given by an interpolating function of the 3D points. The interpolation is composed of three 2D to 1D Thin-Plate Splines (Duchon, 1976)[1], the radial basis function that minimizes the bending energy. They are parameterized by a set of weights (w_k, a, b, c) evaluated from centre (c_k) to image point correspondences. The expression of a Thin-Plate Spline is:

$$f_\tau(u,v) = \left(\sum_{k=1}^{l} w_k^\tau \; \rho\left(d_k^2\right)\right) + a^\tau u + b^\tau v + c^\tau, \tag{4}$$

$$\text{with} \quad d_k = \left\| \begin{pmatrix} u \\ v \end{pmatrix} - c_k \right\|, \quad \rho(d) = d\ln(d) \quad \text{and} \quad \tau \in \{x, y, z\}.$$

The planar points computed by the dimensionality reduction algorithm are chosen as the centres of the Thin-Plate Splines, their corresponding 3D points being the reconstructed points. In order to visualize the surface, a regular surface is created on the planar representation and transferred to the 3D space. The boundary of this surface is given by the object's contour polygon on the flatten object's surface. Eventually it can be reprojected into the images as shown in figures 9 and 10.

6.3 Model Parameters Estimation

The model parameters are first initialized from the estimated general smooth surface and then refined by iteratively minimizing the reprojection error.

6.3.1 Initialization

First, the surface boundary parameters are easily initialized from the 3D surface. For instance, for a rectangular piece of paper, the side length is directly measured as the side length of the 3D surface. Second, the guiding rulings must be defined on the surface. This set of n rulings must represent the surface as accurately as possible. We thus detect a bunch of so-called weak rulings, compute their significance score and keep the most significant rulings as estimates for the model's guiding rulings. These three steps require a weak ruling detection process, a sensible significance score and a method to extract the most significant rulings. There are respectively explained in the next three paragraphs.

Detecting weak rulings. In order to efficiently detect rulings, the properties of developables are used. First of all a ruling starts and stops on the object boundary. The points along the boundary are parameterized by their arc length, so a ruling r is nothing but a couple of arc lengths: $r = (s_A, s_B)$. The set of rulings is thus a mapping m from

[1] This is similar to the TPS warps of Bookstein (Bookstein, 1989) which use two TPS.

the arc length to the arc length along the object boundary:

$$\begin{array}{rcccc} m & : & [0,1] & \longrightarrow & [0,1] \\ & & s_A & \longmapsto & s_B, \end{array},$$

where we normalized the length of the object boundary to one for convenience. A discrete form of this mapping is computed from the initial surface. The arc length along the boundary is evenly sampled. For each sampled arc length s_A, the other arc length s_B is found that leads to the line that stands the best chance of being a weak ruling, according to a weak ruling scalre sc:

$$s_B = \underset{s}{\operatorname{argmin}}\, sc(s_A, s)$$

The weak ruling score is built from the definition of developable surfaces: a ruling must lie on the surface and the tangent plane along a ruling must be constant. Obviously, these properties are most likely satisfied for short candidate rulings so a term to balance this effect is incorporated into the weak ruling score function sc, defined as:

$$sc(s_A, s_B) = \frac{A(s_A, s_B)\, P(s_A, s_B)}{L(s_A, s_B)},$$

where L is the length of the candidate ruling, A is the on-surface term and P is the tangent-plane term. The on-surface term is computed as follow. From the initial surface, we compute some 3D points M_i $(i = 1, \ldots, l)$ along the candidate ruling, and measure how well these 3D points are aligned. Let $\mathbf{v}_i = M_{i+1} - M_i$ and $\bar{\mathbf{v}}$ be the average of the vectors $\mathbf{v}_i$. Figure 11 illustrates the evaluation of this on-surface term:

$$A(s_A, s_B) = \operatorname{var}\left(\left\{\arccos\left(\frac{\mathbf{v}_i^{\mathsf{T}}}{\|\cdot\|}\,\frac{\bar{\mathbf{v}}}{\|\cdot\|}\right)\right\}_{i=1,\ldots,l}\right),$$

For the tangent-plane term, the initial surface allows us to compute the normals $\mathbf{n}_i$ to the surface at the 3D points and the same function as for the on-surface term is used to evaluate if the normals are colinear:

$$P(s_A, s_B) = \operatorname{var}\left(\left\{\arccos\left(\frac{\mathbf{n}_i^{\mathsf{T}}}{\|\cdot\|}\,\frac{\bar{\mathbf{n}}}{\|\cdot\|}\right)\right\}_{i=1,\ldots,l}\right),$$

where $\bar{\mathbf{n}}$ is the average of the normals $\mathbf{n}_i$. Note that the normals are analytically computed from the derivatives of the TPS:

$$\mathbf{n} = \frac{\partial f}{\partial u} \times \frac{\partial f}{\partial v},$$

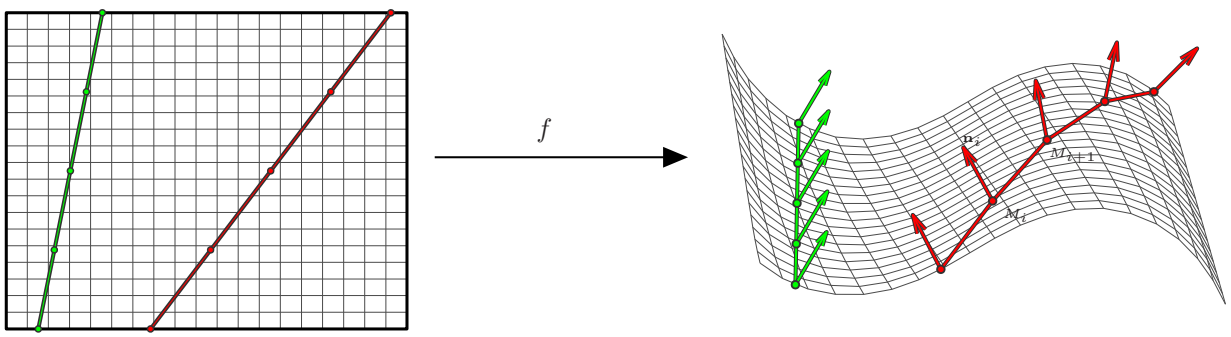


Figure 11: The ruling detection score. (left) The sampled candidate rulings on the flat shape. (right) The candidate rulings mapped to the initial surface with the TPS function f. For each pair of arc length, a ruling score is computed. The $\mathbf{v}_i$ ($\mathbf{v}_i = M_iM_{i+1}$) vectors are related to the on-surface term whereas the $\mathbf{n}_i$ vectors are the normal vectors and thus related to the tangent plane. The green (light) curve is most likely a ruling since the $\mathbf{v}_i$ and the $\mathbf{n}_i$ vectors are aligned as opposed to the ones of the red (dark) curve.

with:

$$\frac{\partial f_\tau}{\partial u} = 2\left(\sum_{k=1}^{l} w_k^\tau \left(u - u_k\right)\left(\ln\left(d_k^2\right) + 1\right)\right) + a^\tau, \quad \tau \in \{x, y, z\}$$

Our implementation takes advantage of the symmetry in the problem (the ruling (s_A, s_B) has the same score as the ruling (s_B, s_A).) Figure 12 shows the result of the detection on the flat shape and on the $al - al$ curve.

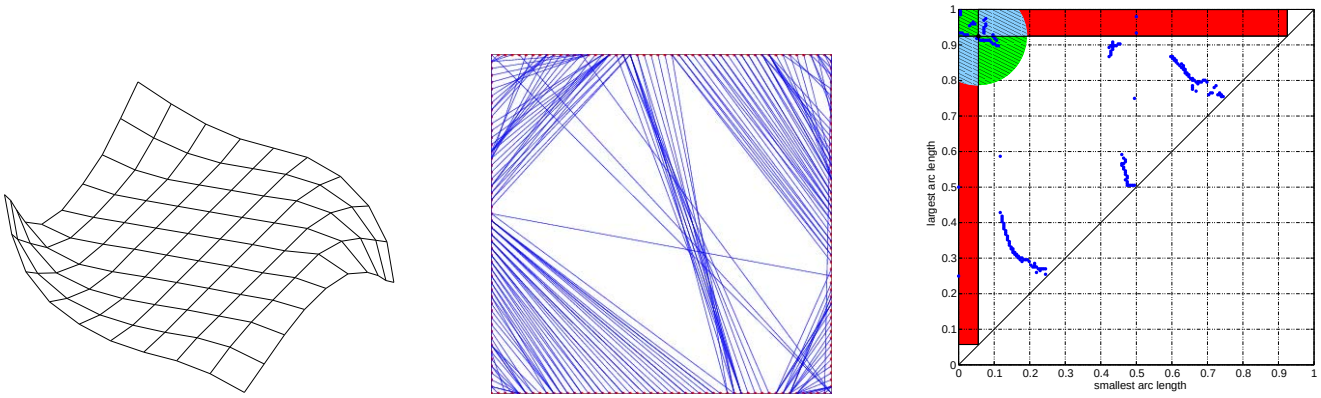


Figure 12: Weak ruling detectiong. (left) The 3D shape. (middle) The detected weak rulings on the flat shape. (right) The detected weak rulings on the $al - al$ curve (as blue points.) The black point is the actual ruling where the score is computed. The light blue (light) and red (dark) zones are the parts that are incompatible with the actual ruling. The green and light blue (hached) parts circle the near detected rulings.

Another method would be to detect rulings using differential geometry properties as in (Perriollat and Bartoli, 2007). This has two main drawbacks, the principal curvature is a local property whereas a ruling is global and this fails in the flat connecting regions producing outliers. These problems are naturally avoided by the method we propose.

Computing the rulings' significance score. The weak rulings are numerous and contain the desired guiding rulings. However, they may also contain erroneous rulings which must be discarded. We propose a ruling significance

score, which facilitates selecting the guiding rulings from the weak rulings. This score is based on three criteria, related to the number and distance between the detected weak rulings. The closeness $\mathcal{N}$ between two rulings r_i and r_j is given by:

$$\mathcal{N}(r_i, r_j) = \begin{cases} 1 \text{ if } d(r_i, r_j) < T \\ 0 \text{ otherwise,} \end{cases}$$

where the threshold T is fixed to 5% of the total arc length; changing this value between 1% and 10% did not affect much our experimental results. The last useful property is based on the edge of regression and states if the other rulings are in the dismissed zones. It is given by the following function:

$$\mathcal{D}(r_i, r_j) = \begin{cases} 1 \text{ if } r_i \text{ and } r_j \text{ cross on the shape} \\ 0 \text{ otherwise.} \end{cases}$$

The weak ruling r is likely to be a guiding ruling if:

1. There are rulings around it:
$$sc_1 = \frac{1}{n} \sum_{i=1}^{n} \mathcal{N}(r, r_i),$$
n being the total number of detected rulings.

2. There are no far rulings crossing it:
$$sc_2 = \frac{1}{\sum_{i=1}^{n} 1 - \mathcal{N}(r, r_i)} \sum_{i=1}^{n} \mathcal{D}(r, r_i).$$

3. The nearest rulings crossing it are well spread around it:
$$sc_3 = \sum_{i=1}^{n} \mathcal{D}(r, r_i)\mathcal{N}(r, r_i)\mathcal{P}(r, r_i),$$
with $\mathcal{P}(r_i, r_j)$ is 1 if r_j is on the right side of r_i and -1 if it is on the left side.

These three scores are used to build the significance map for the curves describing the surface by:

$$sc_C = sc_1 - sc_2 + |sc_3|.$$

An example of significance map is shown in figure 13 for the surface shown in figure 12. The valleys are the curves $\mathcal{C}_i$ that describe the shape (the minimum of the score):

$$\mathcal{C}_i = \underset{c}{\operatorname{argmin}} \int_c sc_C.$$

Non-maximum suppression and chaining are used to detect these valleys, giving the localization of a large set of rulings containing the guiding rulings. These curves are forced to be strictly decreasing. Since the ruling positions are now

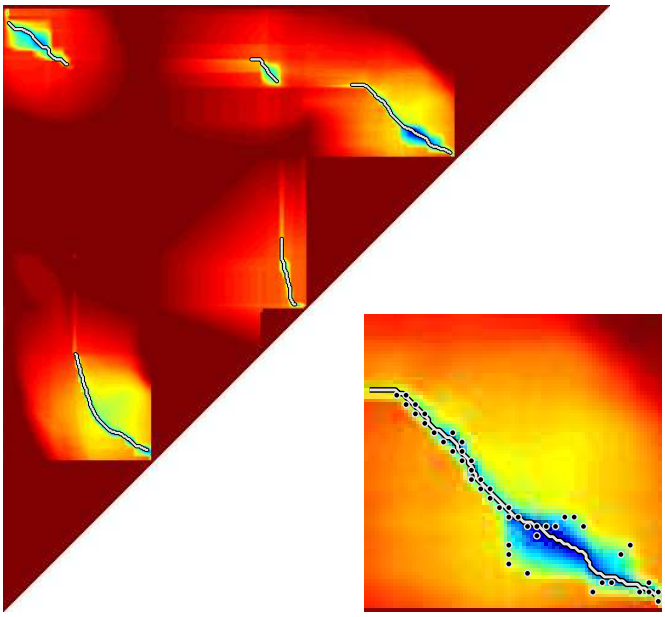


Figure 13: $al - al$ curve detection. The colors represent the ruling significance score: the bluer the color, the more likely the ruling. The white curve is the detected curve representing the ruling position. On the close up, the black points are the detected rulings.

estimated, the bending angles are evaluated directly on the 3D initial surface. Finally, these information are gathered to get the sought-after curves.

Extracting the guiding rulings. The proposed model uses guiding rulings, a small subset of all the possible rulings, to build a developable surface using few parameters thanks to an interpolation process. The initialization leads to the curves $\mathcal{C}_i$ describing the shape via all possible rulings, from which the current step will select guiding rulings. The problem is to optimally choose guiding rulings such that a minimum number describes the whole shape properly. This is cast as a parametric curve fitting problem taking each curves $\mathcal{C}_i$ on turn as input. The desired guiding rulings are recovered from the control points of the fitted parametric curves. A large body of methods were proposed to optimally fit a parametric curve to input data (for instance the spline fitting problem is tackled in (Lu and Milios, 1994; Mamic and Bennamoun, 2001; Pottmann et al., 2002; Yang et al., 2004).) Our problem is a little more complicated since we have to enforce the monotonicity constraint that prevents us using these methods directly. We use a simple dedicated algorithm based on the idea of (Lu and Milios, 1994). The fitting starts with two control points, one at each of the curve extremities, and then iteratively adds new control points where the error is maximum. This process is illustrated in figure 14. The fitting stops when the maximal distance between the data and the parameterized curve is below a user-defined threshold. Thanks to this process, the complexity of the model (the number of parameters) is

directly related to the actual modeled shape.

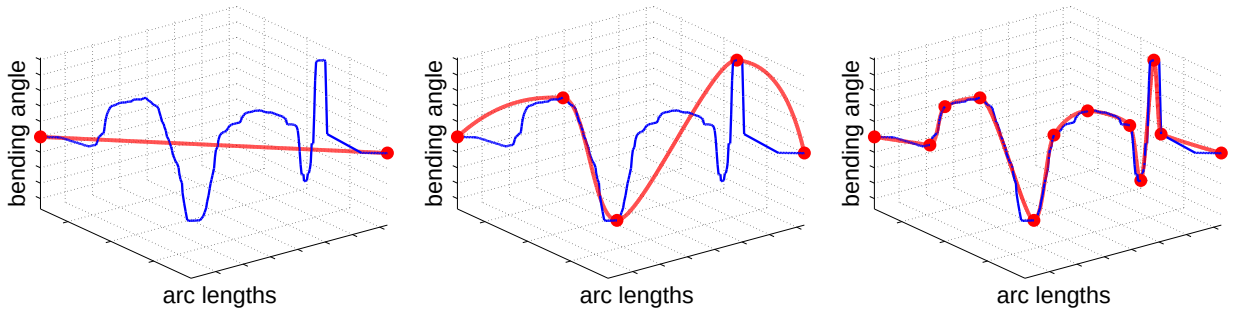


Figure 14: Curve fitting for guiding ruling extraction. The data points (blue, dark), the parameterized curve (red, light) and the control points (red points.) (Left) The two control points when the fitting starts. (Middle) An intermediate fit with five control points. (Right) The selected curve is parameterised by twelve control points.

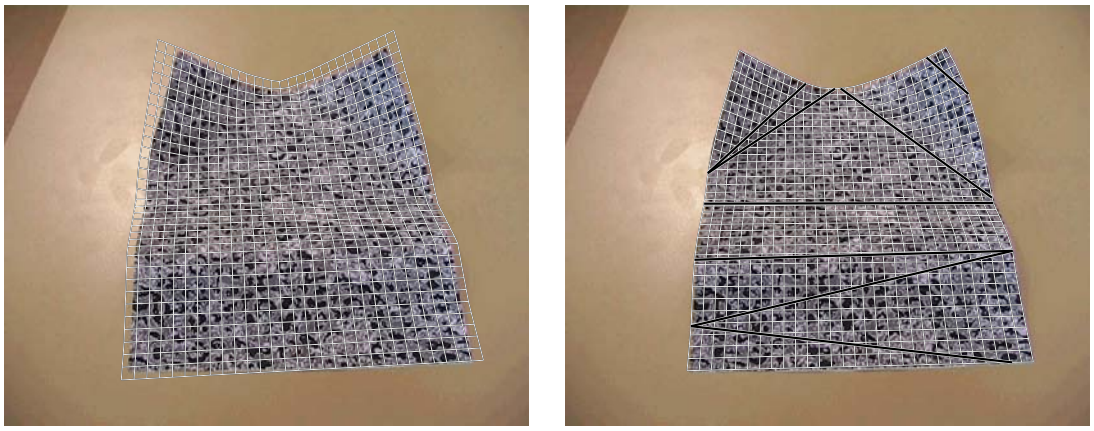


Figure 15: Paper sheet surface estimation with 8 guiding rulings and 16 smoothing rulings. (left) Initial model. (right) Refined model with guiding rulings.

6.3.2 Refinement

The reprojection error describes how well the model fits the actual data, namely the image feature points. Thus latent variables representing the position of each point onto the modeled surface with two parameters are introduced. Let L be the number of images and N the number of points, the reprojection error is:

$$e^2(S, \mathbf{x}, \mathbf{y}) = \sum_{i=1}^{N} \sum_{j=1}^{L} \delta_i^j \left(m_{j,i} - \Pi\left(C_j, M(S, x_i, y_i)\right)\right)^2. \tag{5}$$

In this equation, $m_{i,j}$ is the j-th keypoint in image i (δ_i^j is a binary visibility factor indicating whether point j is visible in image i), $\Pi(C, M)$ projects the 3D point M in the camera C and $M(S, x_i, y_i)$ is a two dimensional parameterization of the points lying on the surface, with S the surface parameters. The points on the surface $(\mathbf{x}, \mathbf{y})$ are initialized by

computing each (x_i, y_i) such that their individual reprojection error is minimized, using the initial surface model:

$$\{x_i, y_i\}_{i\in[1,N]} = \underset{x,y}{\operatorname{argmin}} \sum_{j=1}^{L} \delta_i^j \left(m_{j,i} - \Pi\left(C_j, M(S, x, y)\right)\right)^2.$$

In order to minimize the reprojection error, the following parameters are tuned: the surface parameters (the number of guiding and extra rulings is fixed), see table 1, the pose of the surface (rotation and translation) and the 3D point parameters:

$$\{S, \mathbf{x}, \mathbf{y}\} = \underset{s,x,y}{\operatorname{argmin}}\, e^2(s, x, y)$$

The Levenberg-Marquardt algorithm (Triggs et al., 2000) is used to minimize the reprojection error. Upon convergence, the solution is the Maximum Likelihood Estimate under the assumption of an additive *i.i.d.* Gaussian noise on the image keypoints. The implementation of this minimization algorithm can be optimized by using the sparse pattern of the Jacobian matrix. Solving the normal equations as required by the Levenberg-Marquardt algorithm is efficiently done using the block structure.

7 Results on Real Images

We demonstrated the representational power of our model and our fitting algorithm on seven datasets. For the first six datasets the 3D point clouds were generated by standard Structure-from-Motion, including camera self-calibration, pose estimation and point triangulation (Hartley and Zisserman, 2004). The last dataset is a 3D point cloud which was obtained by structured light.

Figure 16: **The paper dataset.** (left) the 3D reconstructed surface and cameras, (middle) example of retexturing and (right) example of synthetically generated view, including retexturing.

The paper dataset. The following results have been obtained from five views. We used a model with eight guiding rulings and sixteen extra rulings. Figures 15 and 17 show the reprojection of the 3D surfaces into the first image of

the sequence and the reprojection error distribution for the three main steps of our algorithm: Structure-from-Motion, initialization and refinement. Although the former one has the lowest reprojection error, the associated surface is not satisfying, since it is not regular enough and does not fit the actual boundary. The initialization makes the model more regular, but is not accurate enough to fit the boundary of the paper, and introduces large reprojection errors. Eventually, the refined model is visually satisfying and its reprojection error is very close to the unconstrained set of points obtained by Structure-from-Motion. It means that our model accurately fits the image points, while being governed by a much lower number of parameters than the initial set of independent 3D points. The reprojection error significantly decreases thanks to the refinement step, which validates its relevance. Comparing these errors in the object space leads to the same conclusions: the average distance between the triangulated points and the predicted points before (respectively after) the refinement step is 0.16 cm (respectively 0.06 cm), the paper size being estimated to 25 cm by 21 cm. The reconstruction paper and cameras are showed in figure 16.

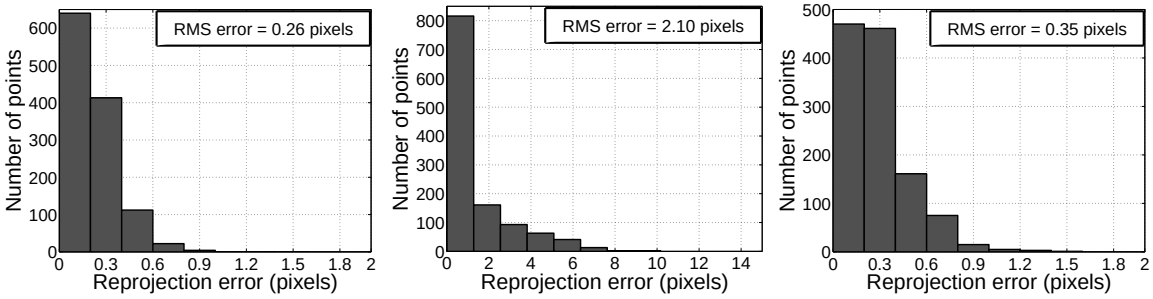


Figure 17: Reprojection errors distribution for the images shown in figure 15. (left) 3D point cloud. (middle) Initial model. (right) Refined model.

Since we have a 3D model of the paper sheet and its reprojection into the images, it is possible to overlay some pictures or to change the texture map. We use the augmentation process described in table 3 to change the whole texture map of the paper and to synthetically generate a view of the paper with the new texture. The results are shown in figure 16. A movie with the new texture and synthetically generated deformations and views is available as supplemental material.

Augmenting images

1. Run the proposed algorithm to fit the model to images
2. Choose an illumination model and light sources
3. For each image, automatically do
   (a) Transfer the new texture map
   (b) Add new 3D objects
   (c) Apply lighting changes

Table 3: Overview of the retexturing and full 3D augmentation process.

The book dataset. The second dataset is an image pair of a book. We estimate the page surface with two guiding rulings and eight extra rulings. Figure 18 shows the reprojection of the estimated surface and the 3D surface. The reprojection of the computed model is fine: the reprojection error of the 3D points is 0.26 pixels and the one for the refined model is 0.69 pixels, taking the triangulated points as ground truth, the final error in object space is 0.06 cm for a page size of 18 cm by 13 cm. It means that we accurately recover the page shape with a surface governed by only nine parameters. One application of the algorithm in the case of a written page is shown in figure 18: our surface estimate is used to unwarp the page's content and to get a rectified image of the text.

Figure 18: **The book dataset.** (left) The reconstructed 3D book model and cameras, (middle) the reprojected surface and guiding rulings and (right) unwrapped texture.

The map dataset. The third example is a sequence of a folded map shown in figure 19. Although all parts of the paper are seen in several images, the whole paper is never entirely seen in a single image. The fitting algorithm naturally deals with this kind of occlusions because the initialization is based on the reconstruction of 3D points, and the 3D point cloud is dense enough since all parts of the paper are visible in several views. The missing points do not perturb convergence because the bundle adjustment minimizes the distance between the actual image points and the reprojection of the 3D points. The reprojection of the model onto one of the original images is shown in figure 19. Since the 3D model of the surface and the position of the cameras are known it is possible to compute an occlusion map for each images. This is useful to unwarp the texture map from each image and to combine them to get the whole texture map. Some partial texture maps and the whole mosaic are shown in figure 19. The reprojection error of the model is 0.45 pixels, very close to the error of the initial triangulation (0.31 pixels), in object space the refined model error is 0.07 cm for a sheet size of 28 cm by 19 cm. We use the 3D information to virtually insert a teapot sliding on the paper surface. Some frames are shown in figure 19 and the full augmented sequence is available as supplemental material to this article.

The poster dataset. The former examples deal with small paper sheets where the developable constraints are always satisfied. A poster is a more challenging object because singularities may appear on the surface due to its

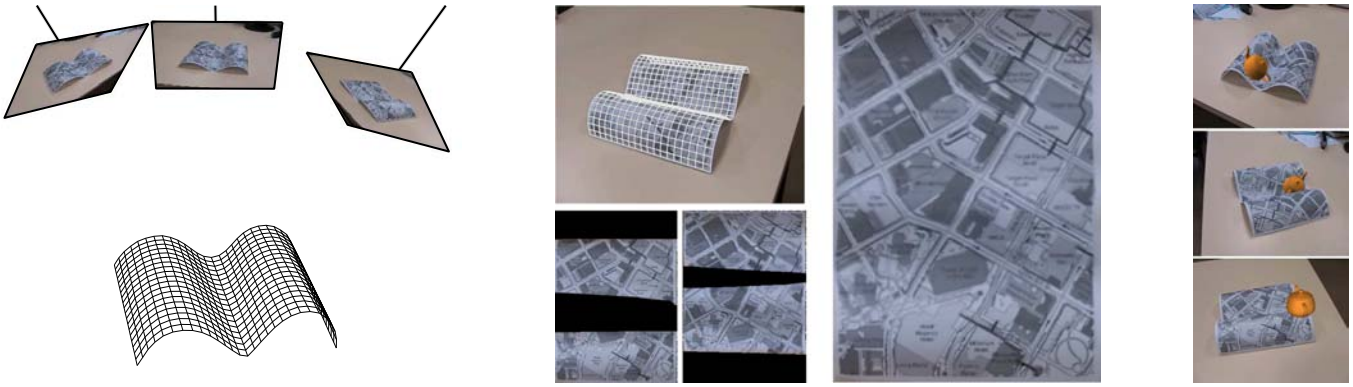


Figure 19: **The map dataset.** (left) Reconstructed shape paper and cameras, (middle) reprojected surface and estimated texture map (obtained by stitching partial texture maps obtained from the input images) and (right) augmented images.

larger size. The input data are two images of the poster obtained from a calibrated stereo system, see figure 20. The surface of the poster is smooth enough, enabling our model to capture the deformations: the RMS error of the triangulated 3D points is 0.35 pixels and the one for our model is 0.65 pixels.

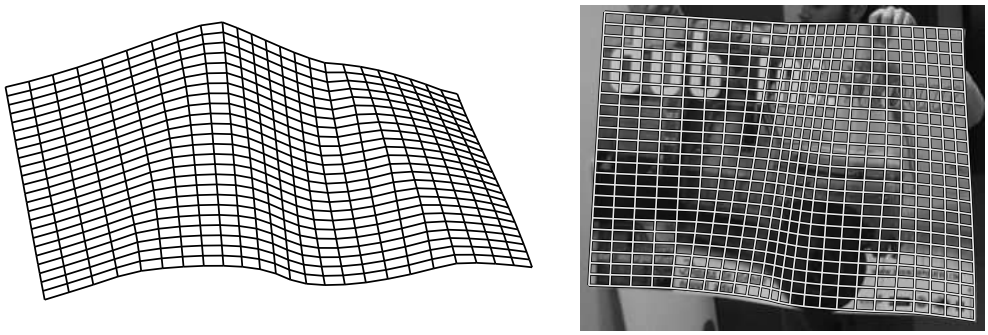


Figure 20: **The Poster dataset.** (left) Estimated surface and (right) reprojection into the first image.

The rug dataset. In this example the model is used to estimate a surface whose physical behavior does not satisfy the developable constraints except under special assumption, for example a suspended piece of fabric or in this case an hanged rug. Even though the results are slightly less accurate, the global shape is well-fitted. The difference between the errors of the triangulated points and the model is representative of the lack of accuracy: 0.34 pixels for the original points against 1.36 pixels for the model. This is mainly visible along the boundary of the rug in figure 21.

Monet's "Soleil couchant". This dataset is made of 14 views of the painting "Soleil couchant" (which means "sunset") visible at the "Musée de l'Orangerie" in Paris. Figure 22 shows two of these images. This painting lies on an elliptical shaped wall. It is not flat but developable and so we can estimate its shape using our algorithms and

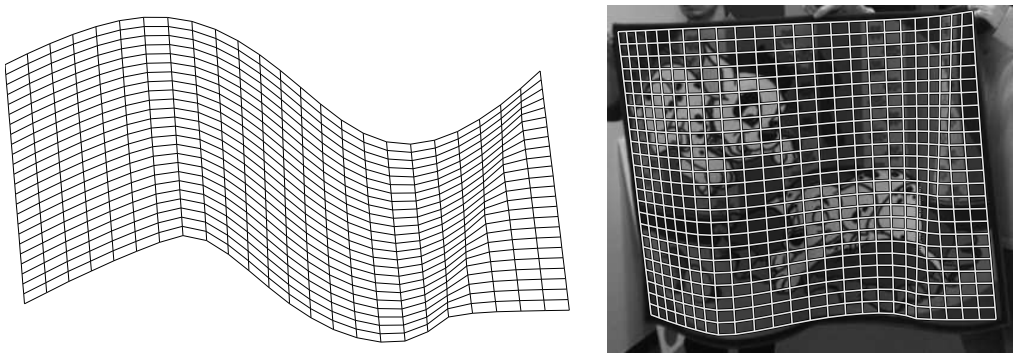


Figure 21: **The rug dataset.** (left) Estimated surface and (right) reprojection into the first image.

eventually flatten it. The painting's size is 200cm by 600cm. The ratio of the size computed by our algorithm is 3.002, meaning that the error on the dimensions is less than 3mm. For this reconstruction, we used 331 points. The point-based Structure-from-Motion algorithm has a reprojection error of 0.62 pixels while our developable Structure-from-Motion reaches 0.66 pixels. This shows that our model captures the 3D shape very accurately. From this 3D model it is then easy to take all the model's bending angles to 0 and get the flattened painting shown in figure 23.

Figure 22: **Monet's "Soleil couchant" dataset.** (left and centre) two images (out of 14) and (right) the reprojected 3D model.

The structured light dataset. Structured light is a technique for 3D reconstruction that gives dense and clean point clouds (Salvi et al., 2004) that combines a projector and a camera. A known pattern is projected to the scene and is then detected in the image. Given the relative camera to projector position and orientation the image points can be triangulated to recover their 3D position. Figure 24 shows a subset of the 162,000 points of a point cloud obtained from the method proposed in (Park and Kak, 2004) for one pose of a piece of paper.

We have seven such reconstructed sets of points for different camera pose and paper deformations. We first qualitatively evaluate our model and reconstruction method using these data, as follows. For each set of points we first randomly select a subset of 100 points. We then estimate the parameters of our paper model using only these 100 points. The results obtained for three points sets are shown in figure 24. We then compute a test error as the RMS (Root Means of Squares) distance between 3D points which were not use to reconstruct our model.

Figure 23: Monet's painting "Soleil couchant" flattened.

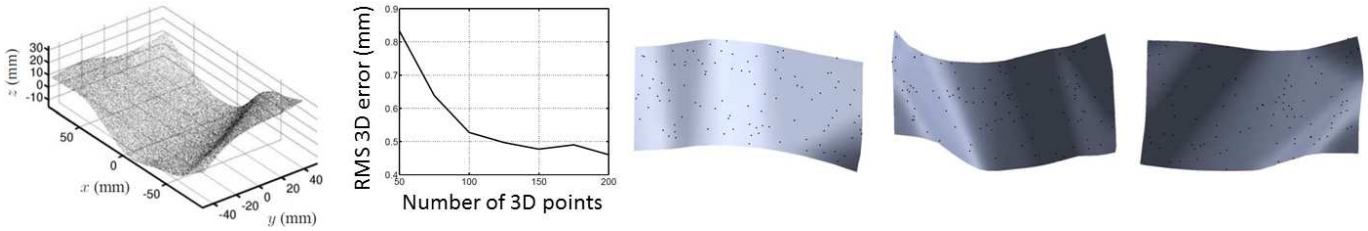


Figure 24: **The structured light dataset.** (left) A cloud of points obtained by a structured light sensor observing a piece of paper (about a third of the 162,000 points is shown, (middle) 3D error computed on an external set of points as a function of the number of points being used and (right) 3D surface obtained for three different point clouds, with an error of 0.14mm, 0.24mm and 0.30mm, respectively.

We also give some more quantitative results. We repeat the same model estimation and test error computation procedure, but averaging over the 7 points clouds, 100 trials, and for a varying number of sample points. The results are shown in figure 24. We clearly observe that the test error goes down with the number of sample points. It stabilizes beyond 100 points at around 0.5mm which is a very satisfying level of accuracy. Indeed the order of magnitude of the error is much lower than the total paper size (100mm by 180mm.) We did not observe overfitting on these data.

8 Conclusion

We addressed the problems of modeling and fitting bounded developable surfaces. To achieve this, we proposed a generative model and a fitting algorithm handling multiple images or dense surfaces. Our model is governed by the flat object's shape and the positions and bending angles of a set of guiding rulings. It guarantees that the generated surface is developable and lets one control the surface smoothness via a simple ruling interpolation process. Our fitting algorithm has two main steps: initialization and iterative refinement of the model's parameters. However, a preliminary step in the image-based case is to get some general smooth surface using Structure-from-Motion and Thin-Plate Spline interpolation. The former also provides a set of keypoint matches across the input images. The initialization step is then performed through a ruling detection process based on that fact that rulings are straight lines with a constant tangent plane. The refinement step minimizes the reprojection error by tuning the model's parameters and the positions of the 3D points constrained to lie exactly on the generated surface. Our model's accuracy was demonstrated by fitting it to several real datasets followed by flattening and augmented reality.

References

G. Aumann. A simple algorithm for designing developable Bézier surfaces. *Computer Aided Geometric Design*, 20: 601–619, 2003.

G. Aumann. Degree elevation and developable bézier surfaces. *Computer Aided Geometric Design*, 21:661–670, 2004.

G. Aumann. Interpolation with developable bézier patches. *Computer Aided Geometric Design*, 8:409–420, 1991.

M. Billinghurst, H. Kato, and I. Poupyrev. The magicbook: A transitional AR interface. *Computers & Graphics*, 25: 745–753, 2001.

P. Bo and W. Wang. Geodesic-controlled developable surfaces for modeling paper bending. In *Proceedings of the Eurographics*, 2007.

F. L. Bookstein. Principal warps: Thin-plate splines and the decomposition of deformations. IEEE *Transactions on Pattern Analysis and Machine Intelligence*, 11:567–585, 1989.

M. S. Brown, M. Sun, R. Yang, L. Yun, and W. B. Seales. Restoring 2d content from distorted documents. IEEE *Transactions on Pattern Analysis and Machine Intelligence*, 29:1904–1916, 2007.

H.-Y. Chen, I.-K. Lee, S. Leopoldseder, H. Pottmann, T. Randrup, and J. Wallner. On surface approximation using developable surfaces. *Graphical Models and Image Processing*, 61:110–124, 1999.

C.-H. Chu and C. H. Séquin. Developable bézier patches: Properties and design. *Computer-Aided Design*, 34:511–522, 2002.

F. Courteille, A. Crouzil, J.-D. Durou, and P. Gurdjos. Towards shape from shading under realistic photographic conditions. In *Proceedings of the International Conference on Pattern Recognition*, 2004.

C. de Boor. *A Practical Guide to Splines*. Springer, 2001.

P. Demartines and J. Hérault. Curvilinear component analysis: A self-organizing neural network for nonlinear mapping of data sets. IEEE *Transactions on Neural Networks*, 8:148–154, 1997.

M. Do Carmo. *Differential geometry of curves and surfaces*. Prentice-hall, 1976.

J. Duchon. Interpolation des fonctions de deux variables suivant le principe de la flexion des plaques minces. In *RAIRO Analyse numérique*, volume 10, pages 5–12, 1976.

L. Fernández-Jambrina. B-spline control nets for developable surfaces. *Computer Aided Geometric Design*, 24:189–199, 2007.

G. . Glaeser and F. Gruber. Developable surfaces in contemporary architecture. *Journal of Mathematics and the Arts*, 01:59–71, 2007.

N. A. Gumerov, A. Zandifar, R. Duraiswami, and L. S. Davis. Structure of applicable surfaces from single views. In *Proceedings of the European Conference on Computer Vision*, volume III, pages 482–496, 2004.

R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Number ISBN: 0521540518. Cambridge University Press, 2004.

D. Hilbert and S. Cohn-Vossen. *Geometry and the imagination*. New York, Chelsea, 1952.

P. A. Houle and J. P. Sethna. Acoustic emission from crumpling paper. *Physical Review E*, 54:278–283, 1996.

Y. Kergosien, H. Gotoda, and T. Kunii. Bending and creasing virtual paper. IEEE *Computer Graphics & Applications*, 14:40–48, 1994.

J. Lang and O. Röschel. Developable (1, n)-bézier surfaces. *Computer Aided Geometric Design*, 9:291–298, 1992.

S. Leopoldseder and H. Pottmann. Approximation of developable surfaces with cone spline surfaces. *Computer-Aided Design*, 30:571–582, 1998.

J. Liang, D. DeMenthon, and D. Doermann. Flattening curved documents in images. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, volume 2, pages 338–345, 2005.

Y. Liu, H. Pottmann, J. Wallner, Y.-L. Yang, and W. Wang. Geometric modeling with conical meshes and developable surfaces. In *Proceedings of* SIGGRAPH, 2006.

D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60: 91–110, 2004.

F. Lu and E. Milios. Optimal spline fitting to planar shape. *Signal Process*, 37:129–140, 1994.

G. Mamic and M. Bennamoun. Automatic bayesian knot placement for spline fitting. In *Proceedings of the* IEEE *International Conference on Image Processing*, 2001.

J. Park and A. C. Kak. Specularity elimination in range sensing for accurate 3D modeling of specular objects. In *International Symposium on 3D Data Processing, Visualization and Transmission*, 2004.

M. Perriollat and A. Bartoli. A quasi-minimal model for paper-like surfaces. In IEEE *Towards Benchmarking Automated Calibration, Orientation and Surface Reconstruction from Images*, 2007.

M. Peternell. Developable surface fitting to point clouds. *Computer Aided Geometric Design*, 21:785–803, 2004.

M. Pollefeys, L. V. Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, and R. Koch. Visual modeling with a hand-held camera. *International Journal of Computer Vision*, 59:207–232, 2004.

H. Pottmann and J. Wallner. *Computational Line Geometry*. Springer, 2001.

H. Pottmann and J. Wallner. Approximation algorithms for developable surfaces. *Computer Aided Geometric Design*, 16:539–556, 1999.

H. Pottmann, S. Leopoldseder, and M. Hofer. Approximation with active b-spline curves and surfaces. In *Proceedings of the 10th Pacific Conference on Computer Graphics and Applications*, 2002.

A. Pressley. *Elementary Differential Geometry*. Springer, 2005.

S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000. URL `http://www.cs.toronto.edu/~roweis/lle/`.

J. Salvi, J. Pagès, and J. Battle. Pattern codification strategies in structure light systems. *Pattern Recognition*, 37: 827–849, 2004.

D. Struik. *Lectures on classical differential geometry.* Addison-Wesley Press, 1961.

M. Sun and E. Fiume. A technique for constructing developable surfaces. In *Proceedings of Graphics Interface*, pages 176–185, May 1996.

J. B. Tenenbaum, V. D. Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, 2000.

B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment – A modern synthesis. In W. Triggs, A. Zisserman, and R. Szeliski, editors, *Vision Algorithms: Theory and Practice*, LNCS, pages 298–375. Springer Verlag, 2000. URL `citeseer.ist.psu.edu/triggs00bundle.html`.

C. Wang and K. Tang. Developable triangulations of a strip. *Computer-Aided Design & Applications*, 2:233–242, 2005.

K. Q. Weinberger and L. K. Saul. Unsupervised learning of image manifolds by semidefinite programming. *International Journal of Computer Vision*, 70:77–90, 2005.

H. Yang, W. Wang, and J. Sun. Control point adjustment for b-spline curve aproximation. *Computer-Aided Design*, 36:639–652, 2004.