Developability of Heightfields via Rank Minimization

SILVIA SELLÁN, University of Toronto
NOAM AIGERMAN, Adobe Research
ALEC JACOBSON, University of Toronto and Adobe Research

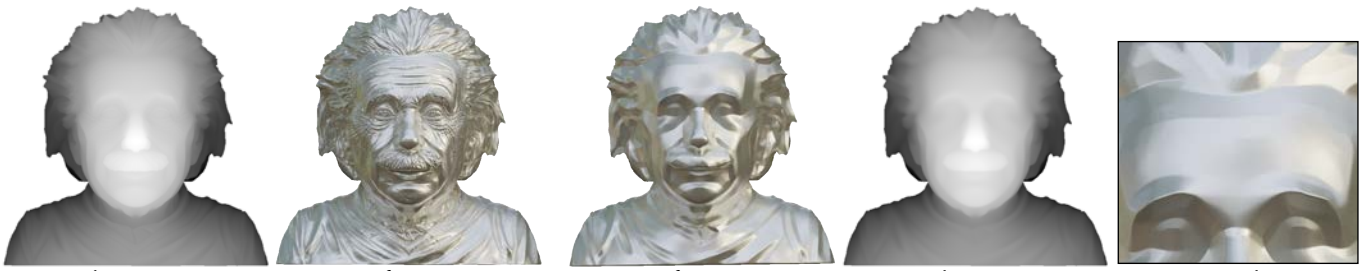


Fig. 1. Our method, inspired by compressed sensing and the problem of rank minimization, takes as input a heightfield describing a surface (left) and outputs a heightfield describing a piecewise developable surface (right) which approximates the input. 3D model by Lloyd Chidgzey under CC BY-SA 3.0.

This work concerns the computation and approximation of developable surfaces — surfaces that are locally isometric to the two-dimensional plane. These surfaces are heavily studied in differential geometry, and are also of great interest to fabrication, architecture and fashion. We focus specifically on developability of heightfields. Our main observation is that developability can be cast as a rank constraint, which can then be plugged into theoretically-grounded rank-minimization techniques from the field of compressed sensing. This leads to a convex semidefinite optimization problem, which receives an input heightfield and recovers a similar heightfield which is developable. Due to the sparsifying nature of compressed sensing, the recovered surface is piecewise developable, with creases emerging between connected developable pieces. The convex program includes one user-specified parameter, balancing adherence to the original surface with developability and number of patches. We moreover show, that in contrast to previous techniques, our discretization does not introduce a bias and the same results are achieved across resolutions and orientations, and with no limit on the number of creases and patches. We solve this convex semidefinite optimization problem efficiently, by devising a tailor-made ADMM solver which leverages matrix-projection observations unique to our problem. We employ our method on a plethora of experiments, from denoising 3D scans of developable geometry such as documents and buildings, through approximating general heightfields with developable ones, and up to interpolating sparse annotations with a developable heightfield.

Authors' addresses: Silvia Sellán, University of Toronto, 40 St. George Street, Toronto, Ontario, M5S2E4, sgsellan@cs.toronto.edu; Noam Aigerman, Adobe Research, 601 Townsend St., San Francisco, California, 94103, aigerman@adobe.com; Alec Jacobson, University of Toronto and Adobe Research, 40 St. George Street, Toronto, Ontario, M5S2E4, jacobson@cs.toronto.edu.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2020 Association for Computing Machinery.
0730-0301/2020/7-ART1 $15.00
https://doi.org/10.1145/3386569.3392419

CCS Concepts: • **Computing methodologies** → *Image processing*; *Shape analysis*; *Mesh geometry models*; • **Mathematics of computing** → *Convex optimization.*

Additional Key Words and Phrases: geometry processing, compressed sensing, rank minimization, developable surface, heightfield

ACM Reference Format:
Silvia Sellán, Noam Aigerman, and Alec Jacobson. 2020. Developability of Heightfields via Rank Minimization . *ACM Trans. Graph.* 39, 4, Article 1 (July 2020), 15 pages. https://doi.org/10.1145/3386569.3392419

1 INTRODUCTION

Developability determines an important subclass of surfaces in three-dimensions. A (piecewise) developable surface is one that can be constructed by folding, creasing, bending or welding planar surfaces without stretching. Piecewise developable surfaces are all around us: paper pages of a book, mechanical objects manufactured with a 5-axis CNC-mill, the wooden-plank hulls of boats, and the steel and glass panels of modern architecture.

While manufacturing techniques for developable surfaces enjoy a long history and ubiquitous use, computational methods for developable surfaces have been notoriously elusive. Mapping the curvature criteria of developability to common discrete surface representations can be tricky: for example, a triangle mesh is trivially piecewise developable; meanwhile, a quad mesh is in general non-planar. There has been a recent surge of advances building new discrete notions of developability for these and other common surface representations. Many if not most works focus on defining developability for a single smooth patch without crease or weld curves, while others require a small number of explicitly provided curves. These methods focus on forward simulation of bending planar patches into a design or surfacing provided boundary curves. Relatively few works consider the inverse problem: which piecewise developable surface best explains an input observation (see Fig. 1).

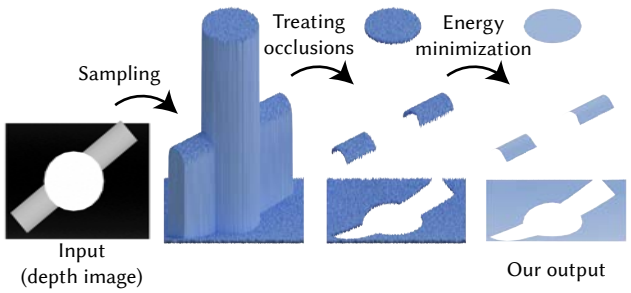


Fig. 2. Visual outline of our method: a (noisy) input depth image is densely sampled with our mesh. After detection and removal of occlusion edges, our convex energy is minimized by a tailor-made ADMM algorithm, outputting a piecewise developable surface that is close to the input.

Methods that do exist suffer from discretization dependence, poor robustness (e.g., getting stuck in local minima), sensitive parameters, or poor scalability.

In this paper, we show that if we restrict consideration to height-field surfaces, the problem of reconstructing a piecewise developable heightfield that best approximates an input heightfield has an efficient relaxation, which can be solved as a convex optimization problem. Our main contribution stems from a simple observation on the connection between two seemingly unrelated fields: developable surfaces and compressed sensing. We observe that developability can be formulated as a low-rank constraint on the second fundamental form of a surface, and that for a heightfield, there exists an efficient convex relaxation to that constraint, in the form of the nuclear norm of the Hessian (matrix of second derivatives) of the height function. This relaxation is rooted in compressed sensing theory, which shows it is rank-reducing and thus aligned with the unrelaxed objective. Minimizing this compressed-sensing-inspired developability term along with a data-fidelity term leads to a convex semidefinite program, which we efficiently solve using a tailor-made ADMM algorithm. As a direct consequence of rank minimization theory, creases and welds naturally emerge during optimization due to the sparse concentration of energy.

Our approach has several unique advantages. It is able to handle inputs with noise and arbitrarily high curvature. Our convex formulation yields a unique global minimum. This harmonizes with another quality: our method is controlled by a single parameter which balances total developability and data fidelty. Geometrically, our formulation poses no restriction on the area of the output surface and the number and orientation of developable patches and creases which may emerge via optimization. Our optimization is not affected by the orientation of the underlying mesh connectivity or the mesh's resolution.

Considering only heightfields strictly limits the scope of our method, but we argue the set of possible inputs remains vast, interesting, and important. For example, architecture is rich with examples of interesting heightfield surfaces [Vouga et al. 2012]. We also enjoy directly accepting as input the output of rasterized 3D scanning technologies.

We demonstrate the effectiveness of our method (see Fig. 2) on a variety of input heightfields. We highlight the advantages of our method through direct comparisons to the state of the art. Finally, we show prototypical lab experiments validating our developability reconstruction on raw, noisy 3D-scanner data (see Fig. 3).

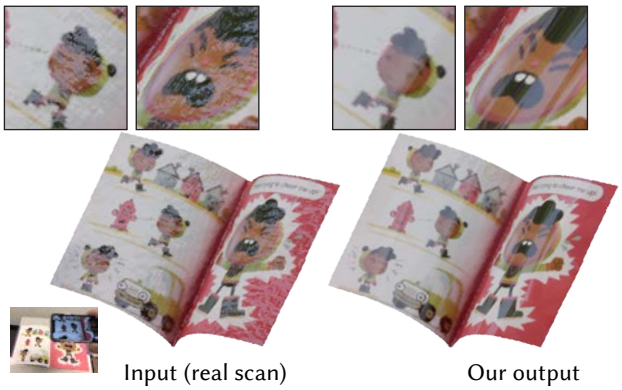


Fig. 3. Recovering the geometry of scanned pages from a book, captured using a 3D scan phone app. The noisy low resolution input is removed of artifacts by our method. Scan of *Ira Crumb Feels the Feelings*, written by Naseem Hrab and Illustrated by Josh Holinaty (Owlkids Books, 2018).

2 BACKGROUND & RELATED WORK

Our work draws on the theory of compressed sensing, convex optimization, and computational methods for developable surfaces. Each area has a broad literature so we focus this section on establishing necessary background mathematics and sufficient context with works most related to ours in terms of methodology or application.

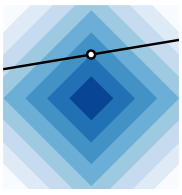
2.1 Compressed Sensing and Rank Minimization

The full history of sparse reconstruction and L_1 techniques is fascinating but outside the scope of this paper (see, e.g., [Boyd and Vandenberghe 2004; Candès and Wakin 2008]). The key insight is that convex relaxations of NP-hard sparsity problems work surprisingly well, often with provable optimality. For example, consider the problem of finding the vector $\mathbf{x}$ with the smallest number of non-zero entries which satisfies the linear equation $\mathbf{Ax} = \mathbf{b}$. This can be written in terms of the L_0 "norm", which counts the number of non-zeros in $\mathbf{x}$:

$$\min_{\mathbf{Ax}=\mathbf{b}} \|\mathbf{x}\|_0. \tag{1}$$

This problem is NP-hard [Boyd and Vandenberghe 2004], but its solution generally agrees with the solution to its convex relaxation using the L_1 norm, which sums the absolute values of entries:

$$\min_{\mathbf{Ax}=\mathbf{b}} \|\mathbf{x}\|_1. \tag{2}$$



The inset figure shows a topographic plot of $\|\mathbf{x}\|_1$ for $\mathbf{x} \in \mathbb{R}^2$, where the line represents the constraint $\mathbf{Ax} = \mathbf{b}$. The solution lands on the "sharp corner" of the isolines of the L_1 objective, which in turn lies on the coordinate axis implying exact sparsity.

As a result, L_1 techniques have been used to model a variety of problems in image processing (e.g., [Didas

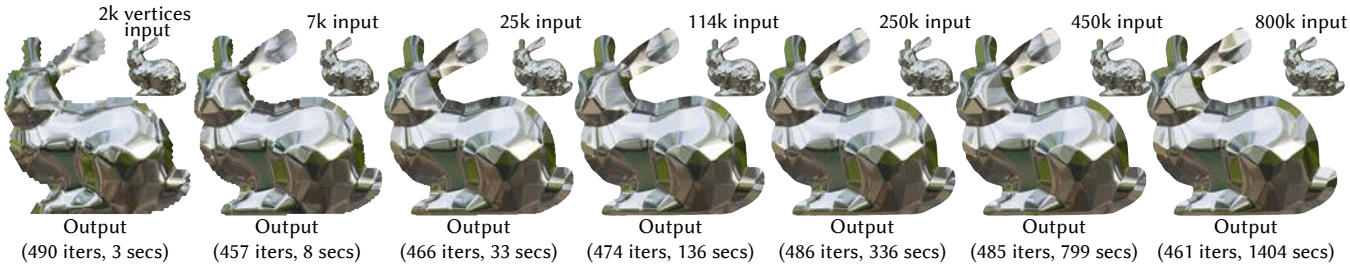


Fig. 4. Algorithm behavior for increasing mesh resolutions. Our output is practically identical for all resolutions. Empirically, the number of ADMM iterations is constant, and the runtime itself scales linearly.

and Weickert 2004; Rudin et al. 1992]) and — closer to our domain — geometry processing (e.g., [Achenbach et al. 2015; Bouaziz et al. 2013; Brandt and Hildebrandt 2017; Huang et al. 2014; Lipman et al. 2007; Rustamov 2011; Zhang et al. 2019, 2014]). Previous works have applied the sparsity-inducing L_1 norm to value or gradient objectives. Recently, Stein et al. [2018b] explored minimizing the L_1 norm of second derivatives to reconstruct piecewise planar surfaces. Similarly, Liu and Jacobson [2019] minimize the L_1 norm on the normals of a triangle mesh to produce cubic stylizations.

Indeed, L_1 minimization has become a ubiquitious method, applied whenever sparsity is desired. Towards the goal of this paper, it may be tempting to directly employ L_1 minimization to the Gaussian curvature itself, noting that vanishing curvature is equivalent to developability. However, while L_1 heuristics like this often work well in practice, in many cases they hold no guarantees of sparsity. As a simple example, consider a vector of unknowns $\mathbf{v} \in \mathbb{R}^n$ and a corresponding vector $\mathbf{u} \in \mathbb{R}^n$ composed of the squared entries of $\mathbf{v}$ (i.e., $\mathbf{u}_i = \mathbf{v}_i^2$). Minimizing $\|\mathbf{u}\|_1$ subject to $\mathbf{A}\mathbf{v} = \mathbf{b}$ with respect to $\mathbf{v}$ will *not* yield a sparse solution, as it is equivalent to minimizing the regular 2-norm $\|\mathbf{v}\|_2^2$ — note this lack of sparsity occurs even though this example is strictly convex.

Intuitively, one can visualize the same image as in the inset before, but now the isolines and the constraint-line are curved — it is no longer clear that they intersect at a sparse corner.

Thus, minimizing the L_1 norm of a vector does not immediately entail its sparsification. This is especially true for many practical methods, which apply the L_1 to *non-convex* functions, usually leading to non-convex optimization. This not only mitigates the guarantees of compressed sensing theory, but also entails dependence of the output on initialization, which may lead to a significantly suboptimal, non-sparse local minimum. Hence, we make a geometric observation which enables us to to employ a theoretically-founded convex optimization problem often used in rank minimization. Its crux is a well-studied sparsifying objective, discussed next.

While, in general, minimizing the L_1 norm of non-convex functions results in a non-convex (and not necessarily sparsifying) function, there is one important exception: applying it to the vector of singular values of the matrix $\mathbf{X} \in \mathbb{R}^{n \times m}$. This matrix function is called the *nuclear norm*, denoted

$$\|\mathbf{X}\|_* = \sum_{i=1}^{m} \sigma_i(\mathbf{X}), \tag{3}$$

with $\sigma_i(\mathbf{X}) \geq 0$ being the i^{th} singular value of $\mathbf{X}$. In case $\mathbf{X}$ is symmetric these are equal to the *absolute value* of the eigenvalues.

As its name implies, the nuclear norm is a true mathematical norm on the space of matrices [Fazel et al. 2001], even though the singular values are highly non-convex functions. Moreover, it is optimizable by standard convex programming techniques, namely semidefinite programming.

The nuclear norm has been deeply studied in compressed sensing, with theory supporting its applicability to *rank minimization*. In fact, it is known to be the convex envelope of the rank for matrices in the unit ball, and is often minimized to recover low-rank matrices.

The intuition for its connection to rank minimization can be derived by noting that the (NP-hard) problem of rank minimization can be equivalently written in the form of an L_0 minimization, analogous to Equation (1):

$$\min_{\mathbf{AX}=\mathbf{B}} \|\sigma(\mathbf{X})\|_0 \Leftrightarrow \min_{\mathbf{AX}=\mathbf{B}} \operatorname{rank}(\mathbf{X}) \tag{4}$$

where here $\mathbf{X}$ is considered as a column-stacked vector, and $\sigma(\mathbf{X})$ is the vector of singular values. The analogous L_1 relaxation follows, with the L_1 norm of the singular values being exactly the nuclear norm, leading to the (surprisingly) convex relaxation:

$$\min_{\mathbf{AX}=\mathbf{B}} \|\sigma(\mathbf{X})\|_1 \Leftrightarrow \min_{\mathbf{AX}=\mathbf{B}} \|\mathbf{X}\|_* \tag{5}$$

In image processing, nuclear norm minimization for low-rank approximation has enjoyed widespread use (e.g., [Candès et al. 2011]). In geometry processing, low-rank optimizations via the nuclear norm have appeared for mesh animation [Mukai and Kuriyama 2016], surface reconstruction [de Lima and Bíscaro 2015; Pan et al. 2016], normal estimation [Zhang et al. 2013] and geometric denoising [Arvanitis et al. 2019; Lu et al. 2018; Wei et al. 2019]. To our knowledge, no previous work has considered the nuclear norm as a convex relaxation of developability.

2.2 Computational Developability

A smooth surface (at least piecewise C^2 and possibly with boundaries) is developable if its Gaussian curvature is zero everywhere:

$$K = \kappa_1 \kappa_2 = 0 \Leftrightarrow \kappa_1 = 0 \text{ or } \kappa_2 = 0, \tag{6}$$

where K, κ_1, κ_2 are the Gaussian, maximum, and minimum curvature, respectively (see, e.g., [O'Neill 1966]). This condition implies not just that locally the surface is *ruled* (defined by sweeping a

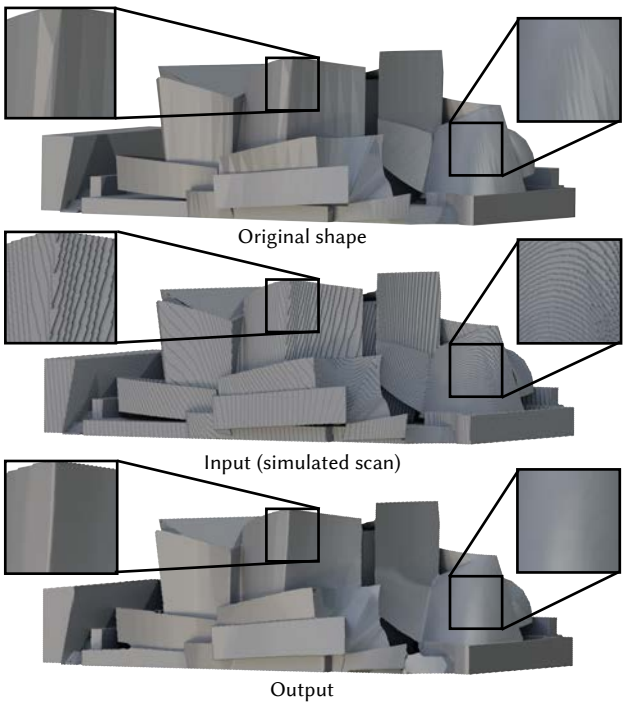


Fig. 5. A coarse mesh of the Walt Disney Concert Hall is further coarsely voxelized and then recovered by our method. Note the recovery of the sharp crease that is not well-approximated in the original coarse mesh. 3D model by Juan Rodríguez under CC BY-SA 3.0.

straight line), but further that the surface normals along any ruling line are constant.

Splines present an attractive and promising discrete representation [Leopoldseder and Pottmann 1998; Pottmann and Farin 1995; Pottmann and Wallner 1999]. More recently, Tang et al. [2016] iteratively project piecewise-spline surfaces onto the submanifold of developable splines for predetermined crease and fold patterns. Taking a discrete differential geometry approach, Liu et al. [2006] define a discrete notion of developability for planar quad meshes. Inputs to this method should be a principal-curvature aligned quad mesh, and the output is restricted by the connectivity of the input mesh. In a recent series of works by Rabinovich et al. [2018a; 2018b; 2019], this restriction was lifted by introducing discrete orthogonal quad meshes. Predetermined creases are incorporated by constraints between overlapping developable meshes.

Various surface design and paper folding approaches have also discretized a single developable surface using strips and ribbons [Kergosien et al. 1994; Kilian et al. 2008; Pottmann et al. 2008; Redont 1989; Wang et al. 2019] and piecewise developables using predetermined patch layouts [Bo and Wang 2007; Frey 2004; Solomon et al. 2012]. Treating folding as a forward simulation, it is possible to apply the finite-element method and optimize over triangle meshes [Bradley 2006; Dudte et al. 2016; Rose et al. 2007], utilizing remeshing [Narain et al. 2013; Schreck et al. 2015] to avoid locking and allow creases to emerge. We eventually discretize our problem

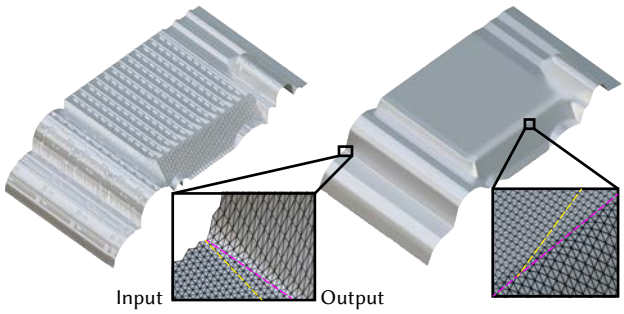


Fig. 6. Agnosticism of our method to the alignment of the underlying mesh: a harsh misalignment of 10°between the creases (purple line in blowup) and the edge lines (yellow line) does not hinder the ability of our method to recover a piecewise developable surface aligned with the input features, without bias to the edges. 3D model by Bold Machines under CC BY 4.0.

over a triangle mesh, but consider this choice incidental. Creases and smooth developable patches emerge regardless of our mesh resolution or orientation, without remeshing (see Fig. 4). Unlike these forward simulations designed to fold a flat design into 3D, our optimization finds a developable surface that closely matches an arbitrary input heightfield (see Fig. 5).

Closer to our reconstruction problem, previous works have considered segmentation of an input surface into (near-)developable patches [Julius et al. 2005; Lee and Bo 2016]. Jung et al. [2015] connect this task to curve-based modeling of 3D piecewise developable surfaces. Lee and Bo [2016] conduct point-cloud segmentation using a planarity prior, followed by independent developable patch fitting.

Prior works have shared our motivation to reverse-engineer existing physical objects and prepare virtual objects for manufacturing. They have considered fitting a single developable patch [Chen et al. 1999; Peternell 2004], finding (near-) developable patches between a predetermined boundary layout [Decaudin et al. 2006; Subag and Elber 2006], fitting developable templates [Hofer et al. 2005], or a sequential optimization that determines a segmentation then fits each developable to each patch [Lee and Bo 2016]. Perriollat and Bartoli [2013] share a similar experimental setup as ours for developable reconstruction from real scan data. Their method considers a single developable surface without sharp creases or welds.

Recently, Stein et al. [2018a] introduced the current state-of-the-art method for approximating an input surface with a piecewise developable surface. Rather than minimize Gaussian curvature directly (cf. [Bradley 2006]), Stein et al. flow the vertices of a mesh until edges align with creases and ruling directions. This method is heavily dependent on the initial triangulation, which is in turn treated as the user's interface to control the method. Creases emerge during the non-convex optimization, before reaching a local minimum. Getting stuck in a local minimum is both a disadvantage and an important criterion because their method does not include a notion of adherence to the input surface, aside from the flow initialization. We compare directly to this method in Figs. 22 and 20. Our method is restricted to heightfields, but enjoys convexity and stability with respect to the input triangulation (see Figs. 5 & 6).

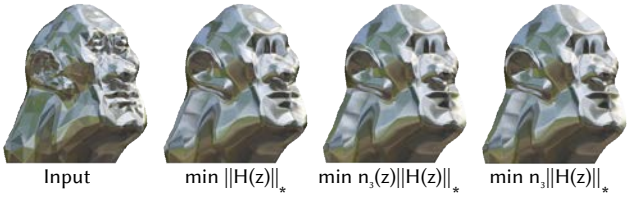


Fig. 7. Comparison of minimizing the Hessian's rank with minimizing the rank of possible proxies to the second fundamental form. The results are virtually identical, with the maximum distance between the three staying under 1% of the bounding box length.

3 DEVELOPABILITY AS HESSIAN RANK MINIMIZATION

Our goal is to design a *convex* objective function that will measure how developable an input heightfield surface is. Let us start by considering a sufficiently smooth surface and we will finish with our proposed discretization over a 2D lattice.

A surface is (piecewise) developable if its Gaussian curvature K is zero (almost) everywhere. Without mentioning principal curvatures explicitly, we can rewrite Equation (6) in terms of the determinant of the surface's second fundamental form $\mathbb{II} \in \mathbb{R}^{2\times 2}$:

$$K = \det \mathbb{II} = 0, \tag{7}$$

where we note that if $\mathbb{II}$ is transformed into the coordinate system of the principal curvature directions then it becomes a diagonal matrix made of κ_1 and κ_2, and the determinant is $\kappa_1\kappa_2$.

Requiring that $\det \mathbb{II} = 0$ is equivalent to requiring that the rank of the second fundamental form is less than one:

$$\operatorname{rank}(\mathbb{II}) \leq 1. \tag{8}$$

For a non-developable surface we can generally expect that $\operatorname{rank}(\mathbb{II}) = 2$. Moving the surface to decrease the rank of the second fundamental form will increase *developability*.

Integrating the rank of the second fundamental form would lead to an unwieldy objective function. Instead, we invoke the theory of rank minimization (see Eqs. (4-3)) and approximate rank integration with the sum of the nuclear norm of the second fundamental form to define our developability objective for a smooth surface S:

$$\int_S \|\mathbb{II}\|_* dA. \tag{9}$$

While this energy is convex in $\mathbb{II}$, $\mathbb{II}$ is still non-linear in the surface positions, leading to a non-convex energy. Our key insight is that we can use a *linear proxy* for $\mathbb{II}$ when the surface is a heightfield. In turn, Equation (9) becomes convex in the height values.

Consider now that the surface S can be described as a graph above the plane: $z(x, y) : \mathbb{R}^2 \to \mathbb{R}$, transforming the area element dA into $dxdy$. The Hessian $\mathbf{H} : \mathbb{R}^2 \to \mathbb{R}^{2\times 2}$ of this height function is the symmetric matrix of second partial derivatives:

$$\mathbf{H}(x, y) := \begin{pmatrix} \frac{\partial^2 z}{\partial x^2} & \frac{\partial^2 z}{\partial x \partial y} \\ \frac{\partial^2 z}{\partial x \partial y} & \frac{\partial^2 z}{\partial y^2} \end{pmatrix}. \tag{10}$$

In Appendix A, we show that the Hessian $\mathbf{H}$ of a heightfield z is proportional in some basis to the second fundamental form $\mathbb{II}$ of the surface defined as the graph of z. In particular, this means that:

Corollary 3.1. *$\mathbf{H}$ has low rank if and only if $\mathbb{II}$ has low rank.*

Thus, we can safely substitute $\mathbf{H}$ for $\mathbb{II}$ in Equation (9), resulting in our proposed developability objective for heightfields:

$$\iint \|\mathbf{H}(x, y)\|_* \, dxdy. \tag{11}$$

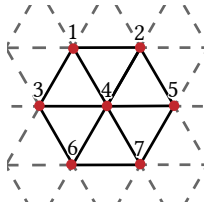
The Hessian is a linear operator in the height function z, and hence, combined with the convex nuclear norm this objective is convex in the height z. The scalar factor that relates $\mathbf{H}$ and $\mathbb{II}$ (see Appendix A) depends on the surface's geometry and is what makes Equation (9) non-convex. In Fig. 7, we compare the convex minimization of Equation (11) (center left) to the non-convex one of Equation (9) (center right). We also add a naive convexification of Equation (9) (right), with this factor being computed only in the input surface and made constant throughout the minimization. We carry out all three of these optimizations using the same ADMM formulation described in Section 4.1; in the non-convex case, we update the scalar factor at each ADMM iteration. We find the results to be qualitatively similar and choose to keep this simpler substitution by $\mathbf{H}$ which guarantees convexity.

3.1 Discretization

We discretize the Hessian nuclear-norm energy in Equation (11) over a regular hexagonal lattice with edge length h placed over the planar domain. For each vertex i, we associate its two-dimensional position $\mathbf{x}_i = [x_i\ y_i] \in \mathbb{R}^2$ and its height value $z_i \in \mathbb{R}$. Treating z as a smooth function, its Taylor expansion exposes its Hessian matrix:

$$z(\mathbf{x} + \Delta\mathbf{x}) = z(\mathbf{x}) + \nabla z^\top(\Delta\mathbf{x}) + \frac{1}{2}(\Delta\mathbf{x})^\top \mathbf{H}(\Delta\mathbf{x}) + \dots \tag{12}$$

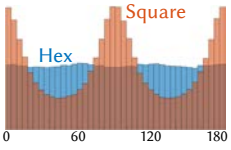
This suggests a best quadratic fit approximation of the Hessian (and gradient and constant).



The chosen hexagonal lattice supplies six neighbors that are equal distance and equally distributed radially to facilitate this approximation. Compared, e.g., to a regular square lattice, this leads to a less biased approximation. Looking locally at one vertex at position $\mathbf{x}_4$ surrounded by its six neighbors (w.l.o.g., indexed reading order), we can solve for the coefficients $\mathbf{c} = [h_{xx}, h_{yy}, h_{xy}, g_x, g_y, c]$ of the best fit quadratic function:

$$\min_{\substack{c\in\mathbb{R} \\ \mathbf{g}\in\mathbb{R}^2 \\ \mathbf{H}=\mathbf{H}^\top\in\mathbb{R}^{2\times 2}}} \frac{1}{2}\sum_{i=1}^{7} \left\| c_4 + \mathbf{g}^\top(\mathbf{x}_i - \mathbf{x}_4) + \frac{1}{2}(\mathbf{x}_i - \mathbf{x}_4)^\top \mathbf{H}(\mathbf{x}_i - \mathbf{x}_4) - z_i \right\|^2.$$

Fortunately, this can be written as a linear function of the vertex heights $\mathbf{c} = \mathbf{B}\mathbf{z}$ (see Appendix B for details) for a specific matrix $\mathbf{B}$. Applying the first three rows of this matrix locally at each vertex i reveals the entries of the best fit Hessian $h^i_{xx}, h^i_{yy}, h^i_{xy}$ as a linear function of the local height values.



It is now that we can justify our choice of a hexagonal lattice as oppossed to a square one. Experimentally, we observe that using a sampling on a square leads to approximated Hessians that

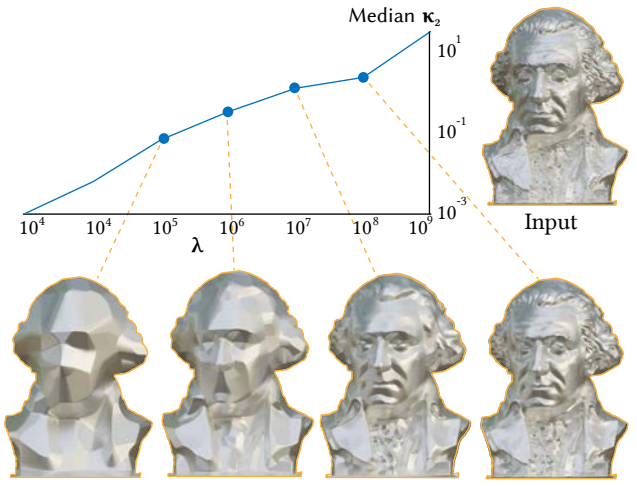


Fig. 8. The λ parameter directly controls the output developability. George Washington bust model by Brian Palmer under CC BY-SA 3.0.

are biased towards alignment with the grid edges (multiples of ninety degrees). This is in contraposition to our hexagonal one, where bias towards the grid directions is barely distinguishable. In the inset, we show the results of an experiment where we randomly generated samples on both square and hexagonal minimal grids to then obtain the angles of the principal directions of the least-squares Hessian. The preference for the hexagonal case is thus justified.

Let

$$\mathbf{H}^i = \begin{pmatrix} h^i_{xx} & h^i_{xy} \\ h^i_{yx} & h^i_{yy} \end{pmatrix} \in \mathbb{R}^{2\times 2} \tag{13}$$

be the Hessian approximation corresponding to vertex i. For a (possibly non-convex) planar domain with boundary, we may now summarize our discretized Equation (11) as sum over interior vertices $\mathcal{I}$ (i.e., set of vertices with a full set of neighbors inside the domain):

$$\sum_{i\in\mathcal{I}} \|\mathbf{H}\|_* . \tag{14}$$

The *boundary vertices* $\partial\mathcal{I}$ of the domain are then the set of non-interior vertices touched by any $\mathbf{B}$ stencil. In the absence of other constraints, minimizing this energy will lead to these vertices receiving discrete natural boundary conditions (cf. [Courant and Hilbert 2008; Stein et al. 2018b]).

This energy (and its smooth counterpart) has exactly affine functions in its null spaces. While this means it is non-zero for non-affine developable surfaces, it should not dissuade us from its use as a developability measure. This should be analogously comfortable to those familiar with the use of the total variation energy (see, e.g., [Boyd and Vandenberghe 2004]) as a smoothness regularizer despite only having constant functions in its null space. In the presence of a data-fitting term or non-trivial boundary conditions, the minimizer will be far from the null space and the gradient behavior of the energy will be the dominating effect.

4 PIECEWISE DEVELOPABLE FITTING

With our discretized Hessian's nuclear norm in hand, we can turn to the main problem that we wish to solve: fitting a piecewise developable surface to an input heightfield observation. We will assume that the observed heightfield data $\tilde{\mathbf{z}} \in \mathbb{R}^n$ arrives as values on a hexagonal lattice with n vertices as discussed in the previous section, or otherwise can be resampled accordingly.

Our fitting energy consists of an L_2 data fidelity term and our discrete Hessian nuclear norm energy from Equation (14), leading to the following optimization problem:

$$\min_{\mathbf{z},\mathbf{h}} \lambda \sum_{i\in\mathcal{I}\cup\partial\mathcal{I}} \|z_i - \tilde{z}_i\|^2 + \sum_{i\in\mathcal{I}} \|\mathbf{H}\|_* \tag{15}$$

$$\text{subject to } \mathbf{Cz} = \mathbf{h}$$

$$\text{and } h^i_{xy} = h^i_{yx} \ \forall i \in \mathcal{I} \tag{16}$$

where $\mathbf{h} \in \mathbb{R}^{4|\mathcal{I}|}$ stacks all of the Hessian coefficients of interior vertices and the sparse matrix $\mathbf{C} \in \mathbb{R}^{4|\mathcal{I}|\times n}$ linearly assembles them according to the local stencil $\mathbf{B}$ above. The scalar weighting parameter λ balances data fidelity and developability (see Fig. 8).

While the objective is convex in the unknown height values $\mathbf{z} \in \mathbb{R}^n$ it is more complicated than a simple quadratic program. Indeed, as Fazel et al. [2001] shows, nuclear norm minimizations like this can be reduced to a semi-definite programming problem (see Appendix C). While solving this SDP with standard convex optimization techniques would be a possibility (see [Boyd and Vandenberghe 2004; Fazel et al. 2001]) we can do significantly better by using the specifics of our energy to design a direct ADMM optimization.

4.1 Tailor-Made ADMM Optimization

The alternating direction method of multipliers (ADMM) has been a boon to convex optimization [Boyd et al. 2011]. This method is generally applicable for linearly constrained convex programs written as the sum of two convex objectives, and fortunately our problem in Equation (15) already has this form and is in fact already split so that the first and second terms depend only on the $\mathbf{z}$ and $\mathbf{h}$ variables, respectively.

Let us use the notation $\mathbf{C}^i \in \mathbb{R}^{4\times n}$ to be the four rows of $\mathbf{C}$ corresponding to vertex i. We introduce a set of dual variables $\mathbf{u}$ corresponding to $\mathbf{h}$ (with analogous notation $\mathbf{U}^i \in \mathbb{R}^{2\times 2}$ as in Equation (13)). We now follow the *scaled form* of the ADMM algorithm described by Boyd et al. [2011], which repeats three steps:

$$\mathbf{z} \leftarrow \operatorname*{argmin}_{\mathbf{z}} \lambda\|\mathbf{z} - \tilde{\mathbf{z}}\|^2 + \frac{\rho}{2}\|\mathbf{Cz} - \mathbf{h} + \mathbf{u}\|^2 , \tag{17}$$

$$\mathbf{H}^i \leftarrow \operatorname*{argmin}_{\mathbf{H}^i} \left\|\mathbf{H}_i\right\|_* + \frac{\rho}{2}\left\|2\text{x}2(\mathbf{C}^i\mathbf{z}) - \mathbf{H}^i + \mathbf{U}^i\right\|_F^2 , \ \forall i \in \mathcal{I}, \tag{18}$$

$$\mathbf{u} \leftarrow \mathbf{u} + \mathbf{Cz} - \mathbf{h}, \tag{19}$$

where

$$2\text{x}2(\mathbf{a}) := \begin{pmatrix} a_{xx} & a_{xy} \\ a_{yx} & a_{yy} \end{pmatrix} . \tag{20}$$

We can see immediately that the $\mathbf{z}$ update step in Equation (17) is a simple quadratic minimization solved via an $n \times n$ sparse linear system. The $\mathbf{h}$ update step in Equation (18) is *embarassingly* parallel across the vertices of the mesh, depending only on *local* data. Each local problem is a small semi-definite program in four

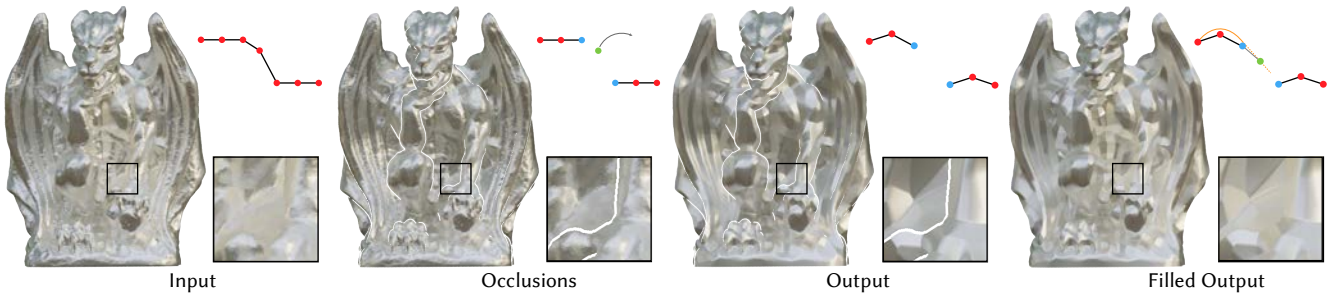
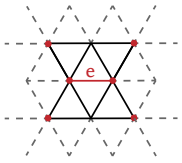


Fig. 9. Pre-processing and post-processing for handling occlusions, illustrated in 1D on top, and with a real input on the bottom. The input (left) is tesselated so that each vertex is connected to all its neighbours. During occlusion detection (second from left), vertices that lie across a steep change in height are set as boundary and their Hessians are ignored. The two blue vertices contribute to their red neighbours' Hessian, but the green vertex has no non-boundary vertices adjacent and hence removed. After running our method (second from right) on this modified input, the omitted green vertex is restored (right), with its height value set by extrapolating the quadric approximation of the closest non-boundary vertex. 3D model by 3DWP under CC BY-NC-SA 4.0.

variables ($h^i_{xx}, h^i_{yy}, h^i_{xy}, h^i_{yx}$). Rather than call a general-purpose algorithm, we can solve each in closed form and avoid auxiliary variables and other overheads (see Appendix D).

After each iteration through the three steps in Equations (17-19), we check for convergence (following [Boyd et al. 2011]) and update ρ using the rule given by [Boyd et al. 2011] in Section 3.4.1. Updating ρ changes the system matrix in the update of $\mathbf{z}$ in Equation (17) which invokes a new sparse Cholesky factorization. To avoid doing this too often we loosen the criteria on triggering a ρ-update as recommended by Stellato et al. [2017].

Occlusion boundaries. Our optimization so far will work well for (noisy) observations of a continuous heightfield. In many practical scenarios, there will be very large jumps in the value of the observations $\tilde{\mathbf{z}}$ at self-occlusion boundaries. In some scenarios, these boundaries are known *a priori* and we can accept them as input. In most cases, the occlusion boundaries will need to be detected automatically. Occlusion detection is not the main focus of our paper, but for reproducibility we describe the heuristic we used to create our examples and experiments.



We fit a quadratic function to the six vertex heights that are in the vicinity of *each edge* (see inset). If the largest eigenvalue of the quadratic function's Hessian is great than a threshold (in our results, this threshold is 4×10^4), then we declare the edge to be an occlusion boundary. As grid (i.e., "scanning") resolution increases, the curvature of the quadratic fit across these edges grows asymptotically, guaranteeing that for a fine-enough grid, all occlusions, and only them, will be detected. We reiterate that occlusion detection is not our focus; other more sophisticated methods may exist and our developability optimization would immediately benefit from them.

Once provided or detected, the vertices of each occlusion edge are removed from the set of interior vertices ($\mathcal{I}$) where the Hessian stencil will be evaluated (see Fig. 9). Prior to optimization, we omit any vertices that may end up not touched by any stencil (i.e., not in $\mathcal{I}$ or $\partial\mathcal{I}$). We then restore these vertices by setting its height value by extrapolating from a best-fit quadratic function to the solution, centered around the nearest interior vertex (see Fig. 9).

5 RESULTS

Our method's robustness and independence to discretization enables its employment for different goals, from in-the-wild capture of piecewise developable surfaces, through approximating non-developable heightfields with developable ones, and up to interpolation of given height constraints with a developable heightfield. We further evaluate various aspects of our method's performance, and compare to two rivaling techniques for developability and mesh denoising.

In producing the following results, we chose between three single options for λ: "big" or $\lambda = 10^7$, "medium" or $\lambda = 10^6$ and "small" or $\lambda = 10^5$. The heuristic for choosing between these three is simple. In cases where our input is close to developable but for a few artifacts (see Figs. 3, 11, 15, 17 top left, and 16), we choose the "big" λ since we expect little deviation from the input. In cases where the amount of noise is large in both amplitude and distribution (see Figs. 5, 12, 10, 17 top right and bottom, and 21), we choose the "small" λ. Finally, in the cases where the input is clearly non-developable and we wish to approximate it by developable patches, the choice of λ represents the stylistic decision of whether to have a higher number of patches but a higher input fidelity, or lower both.

We normalize the XY coordinates of every heightfield to fit the unit square in the interest of consistency. Unless specified otherwise, all our examples contain 50k-300k vertices. We have also made the conscious choice of rendering our results using a metallic material to emphasize the surface normals, which are key to visually analyzing developability. We have done this also in cases like Figs. 10 and 3, where the scanned material (paper) is considerably less reflective.

We implemented our method in MATLAB, relying on the libraries GPTOOLBOX [Jacobson et al. 2016] and LIBIGL [Jacobson et al. 2018]. We report timings conducted on a machine with Intel Xeon CPU E5-2637 v3 @ 3.50Hz (16 cores), Nvidia GTX 970 and 64 GB of RAM.

5.1 Recovering Scanned Developables

Depth scans of piecewise developable geometry are a natural fit to our method as they produce heightfields. Due to noise and blurring, the resulting scans contain significant artifacts like oscillating normals and over-smoothing of sharp creases. Our method can be used to recover the underlying piecewise developable geometry.

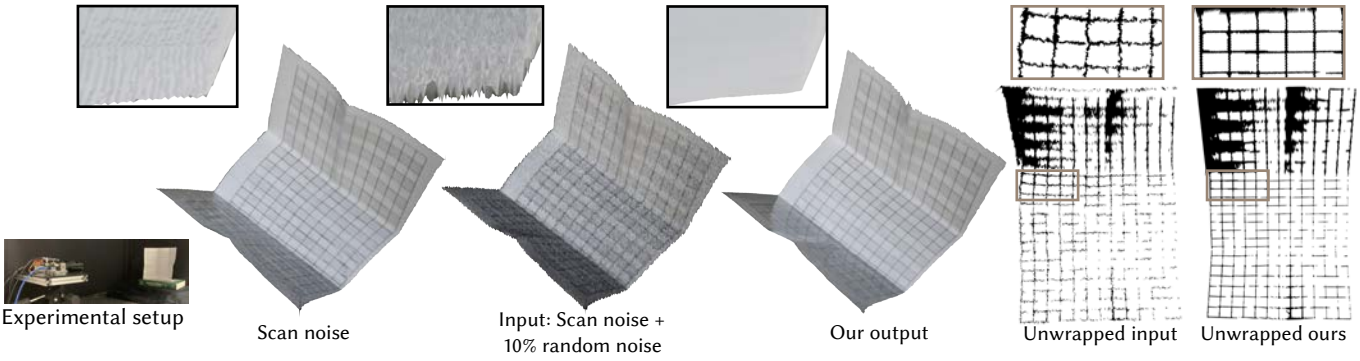


Fig. 10. Recovering the developable geometry of a scanned page. The scan's inherent noise is further amplified by additional synthetic noise, which is then run through our algorithm. On the right, the unwarping of the input and output reveals our method's removal of non-developable artifacts.

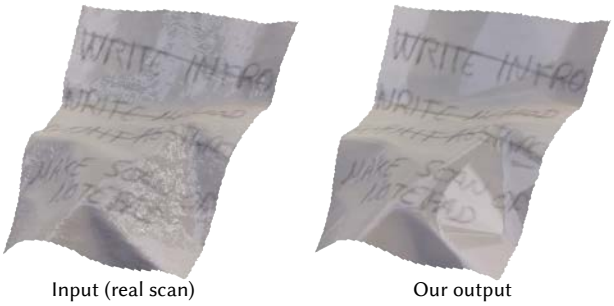


Fig. 11. Reconstructing a noisy phone-app scan of a crumpled notepad paper is cleaned by our method.

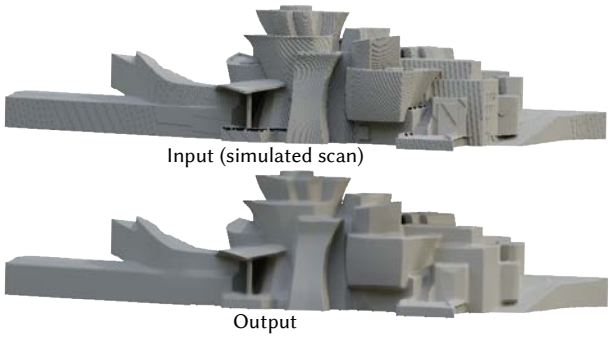


Fig. 12. Recovering the Bilbao Guggenheim Museum from a simulated depth scan.

Paper scans. To test our method's ability to recover developable geometry from scans, we printed a grid on an A4 paper and scanned it using a high-resolution depth scanner, as shown in Fig. 10. To further challenge our method, we added an additional random noise with amplitude of 10 percent of the full input's amplitude, and ran it through our method. To verify our result, we unwarped the geometry to 2D, by running an isometric parameterization algorithm [Liu et al. 2008], see Fig. 10, right . If the output is not developable, *i.e.*, is not isometric to a planar sheet, the unwarping will have distortion. Indeed, the unwarping of the noisy input is jagged and exhibits distortion, while our method's output is unwarped with smooth, straight lines.

A similar experiment is shown in Fig. 3; however, in this case we use a depth-scanning smartphone app, that even for this smooth input produces a low quality scan with oscillating normals. Our method succeeds in removing these artifacts and recovers the natural curving of the pages, while still adhering to the crease between the pages. The same process is followed in Fig. 11, where we recover the shape of our crumpled notepad from a noisy phone scan.

Simulated rasterized scans of large-scale objects. To simulate scans of real-world large-scale objects, we coarsely rasterize them and then sample a heightfield, to create a staircasing effect. This effect is especially challenging to our method, as it is piecewise-zero-curvature when sampled densely by our grid. However, since our method aims to strike a balance between developability and number of creases, it still optimizes to remove this artifact. In Figs. 5 and 12 we show two examples of our method applied to rasterizations of famous piecewise developable buildings – the Disney concert hall, and the Bilbao Guggenheim Museum, respectively. Note the Disney concert hall is initially a coarse surface mesh with triangulation artifacts (note recovered crease on left). We nonetheless manage to recover smooth developable surfaces in both cases and remove the staircasing artifact.

5.2 Developable Approximations to Heightfields

When the input heightfield is not assumed to be sampled from a developable surface, our method is still successful in approximating the input with a piecewise developable heightfield.

Developable stylization. We can approximate given heightfields with developable patches, which can be used as a stylization effect or for fabrication from bendable metal sheets. Here λ serves as an artist-chosen parameter balancing adherence to geometry with number of patches. In Fig. 13 we use our "small" λ to decompose the lion's scultpure into few developable pieces. In Fig. 14, our "big" λ leads to a better approximation of the heightfields with more developable patches; only small detail such as facial features are lost.

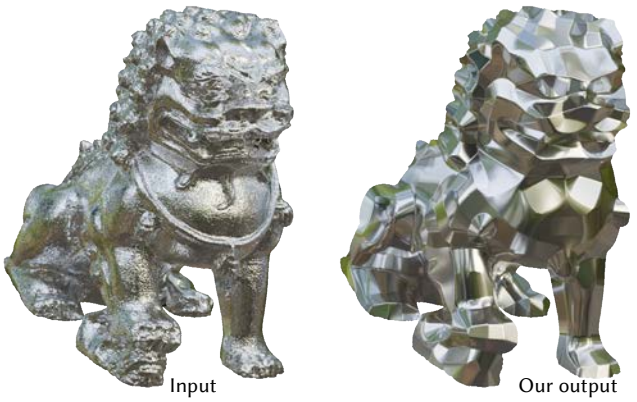


Fig. 13. Using our method we can reconstruct the lion from large, developable patches. Model from the AIM@SHAPE mesh repository.

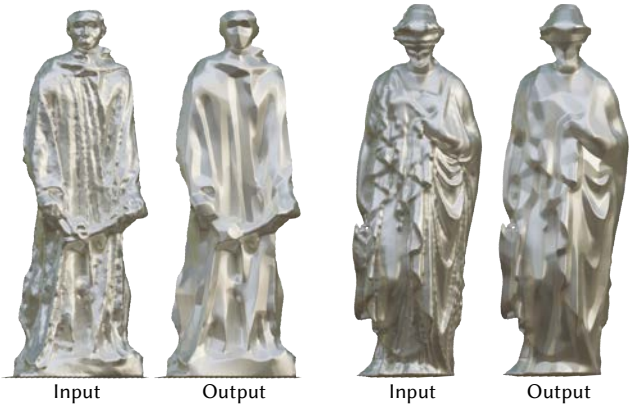


Fig. 14. A developable museum. Rodin's *Les Bourgeois de Calais* model (left) by Yasmine Afshar under CC BY-NC 4.0 and Bronze Greek Statue (right) model scanned by Matt Stultz under CC BY-SA 3.0.

Alleviating tessellation artifacts. A coarse mesh of a developable surface can also be sampled densely with our grid, to enable an ad-hoc "developable superesolution upsampling" that restores a fine developable heightfield from the sampling of the coarse mesh. In Fig. 15 we show a heightfield stemming from a mesh of Zaha Hadid's *Bench*. Our output recreates a perfectly smooth developable surface, removing the visible tessellation artifacts. Likewise, in Fig. 16 we recover the high-resolution, piecewise developable Fandisk from its low-resolution counterpart. Note how our method smooths the different pieces without affecting the creases.

Finally, we sample heightfields from three different triangulated outputs generated by [Rabinovich et al. 2018a] to illustrate the strength of our method. In Fig. 17, we densely upsample one of their geodesic nets (top left) to remove tessalation artifacts, using "big" λ. In the top right and bottom, we add synthetic noise and use our "small" λ to denoise them and recover the developable groundtruth.

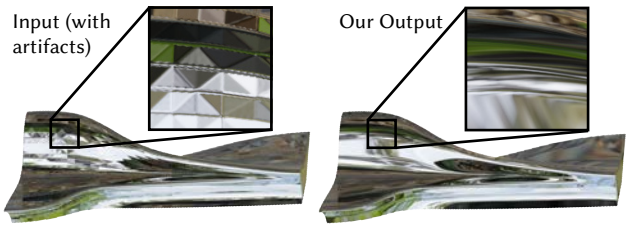


Fig. 15. Recovering the underlying developable surface while removing triangulation artifacts of Zaha Hadid's *Bench* (model by VSR under CC BY-SA 3.0).

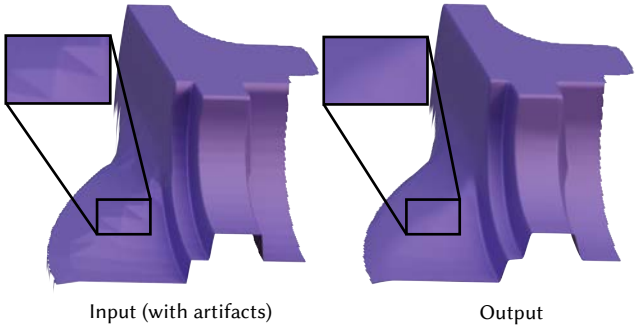


Fig. 16. Recovering the high-resolution Fandisk from a coarse mesh.

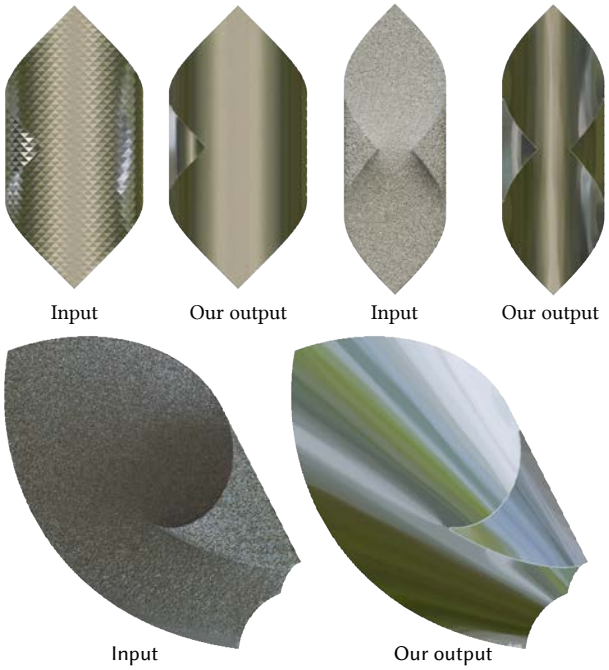


Fig. 17. Given coarse results from [Rabinovich et al. 2018a], our method can be used to either remove the discretization artifacts (top left) as well as to recover the underlying developable surface when synthetic noise is added (top right and bottom).

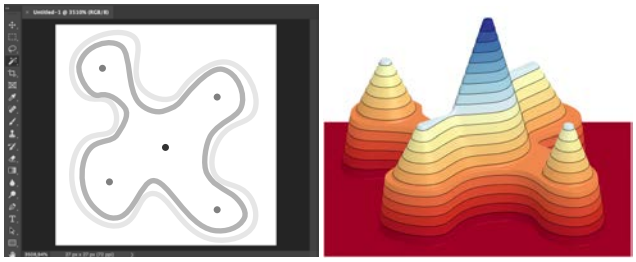


Fig. 18. We minimize the nuclear norm subject to fixed points constraints to apply our method to developable data interpolation.

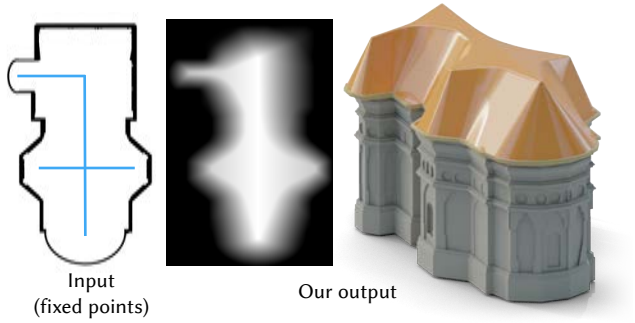
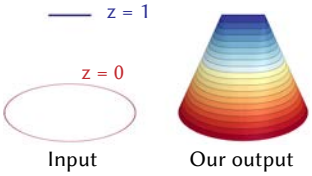


Fig. 19. We use fixed point constraints (left) to generate the heightfield of a cathedral roof using our nuclear norm minimization.

5.3 Design of Developable Heightfields



Our approach can also be used for *designing* heightfields that are developable. Given a sparse set of constraints of the form $z_i = b_i$, we fix them as hard constraints $Cz = b$ in Equation (15). We can then solve our optimization Equation (15) with only the nuclear norm term, to enforce developability of the output signal in the interpolated points (see inset). In Figs. 18 and 19 we present possible use cases, where the boundary conditions are drawn using a standard image editing software.

5.4 Comparisons and Evaluations

Comparison to Developability of Triangle Meshes [Stein et al. 2018a]. There exist many deep and elegant methods for the computation of developable surfaces, however most of them aim for user-driven developable modeling and hence cannot fit a developable to a given geometry. Of the few methods that aim for approximating an input geometry with a developable surface, [Stein et al. 2018a] is considered state of the art.

While their method is more general and can be applied directly to any mesh embedded in 3D, its output is designed to create creases along edges and hence is closely dependent on the mesh's tessellation and resolution. Fig. 20 illustrates this with a simple experiment:

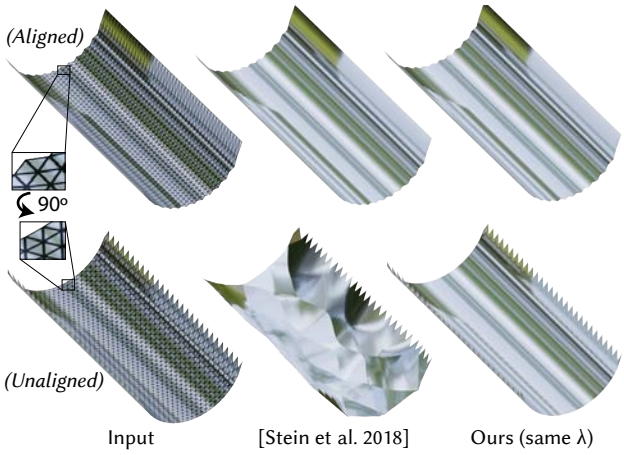


Fig. 20. Comparison to [Stein et al. 2018a] on detecting developability. Both methods reproduce the input cylinder when the mesh's edges align with the ruling. When the edges are misaligned with the ruling, their method produces artifacts. Ours yields the same identical output as before.

while their method reproduces the cylinder when the mesh's edges are aligned with the cylinder's principal curvature, once the cylinder is retesallated such that the mesh's edges are rotated by 90 degrees, [Stein et al. 2018a] introduce a large amount of creases and patches, as their energy does not detect non-edge-aligned rulings. Our method produces the same result for both alignments, and we explore this advantage further in Fig. 6.

The dependence on mesh tessellation is further exacerbated by the extremely non-convex optimization of their objective, which may lead to different local minima. In Fig. 22 we rotate the same input (a noisy heightfield of a cone and cylinder) to 3 different orientations and sample each one with our mesh, yielding different alignment of principal curvatures with mesh edges. While this has insignificant effect on our output, the output of [Stein et al. 2018a] changes, with different creases for each orientation.

Comparison to L_0 Mesh Denoising [He and Schaefer 2013]. Competing techniques for mesh denoising may be employed instead of ours. Similarly to us [He and Schaefer 2013] suggest a sparsity-inducing denoising technique for meshes. Their optimization problem is non-convex and hence heavily relies on the initialization. In Fig. 21, the oloid is rasterized and then sampled on our triangular grid. Our method manages to reproduce the ground truth, while [He and Schaefer 2013] yield a significantly different result.

Evaluations. Our method and discretization are extremely robust, as shown by the following stress tests. In Fig. 6 we intentionally misalign the underlying mesh's edge-lines with the input surface's principal curvatures. Nonetheless, our method reproduces the underlying surface without any artifacts, exhibiting its lack of bias to the edge's orientation.

In Fig. 21, other functions aside from the nuclear norm such as the Frobenius norm, or simply the nuclear norm squared, will not lead to a developable heightfield with sparse creases. We swap the nuclear norm with the above two options in our optimization

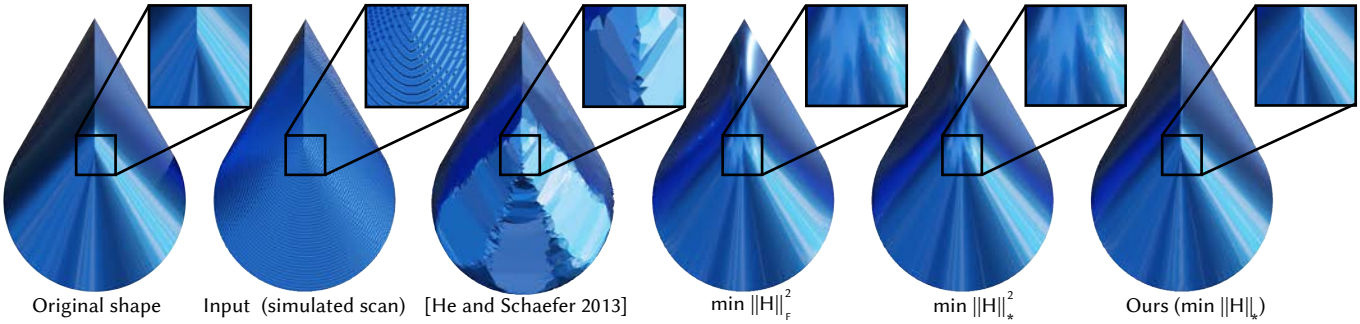


Fig. 21. Comparison between the nuclear norm, the squared nuclear norm, and the Frobenius norm of the Hessian. The nuclear norm encourages developability except in sparse creases, while the squared nuclear norm and the Frobenius norm smooth the shape uniformally, trading the sharp crease for a smooth non-developable region.

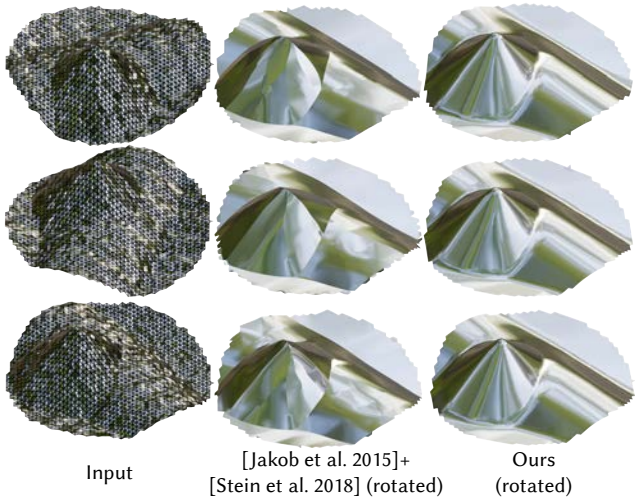


Fig. 22. Comparison of our method with [Stein et al. 2018a] in recovering piecewise developable geometry from a noisy developable heightfield. We sample the heightfield with our triangular grid in 3 different orientations, with [Stein et al. 2018a] producing different results for each one. Our output is unaffected.

problem Equation (15), and attempt to recover the oloid heightfield from a rasterized version of it. Both options opt to smooth out the crease as well as the entire heightfield, sacrificing developability.

In Fig. 23, we show the robustness of our discretization and optimization to extreme levels of curvature, by feeding in highly-oscillatory topographical heightfields with high frequency oscillations. Our convex optimization is unaffected, and reproduces a piecewise-smooth developable approximation to both inputs.

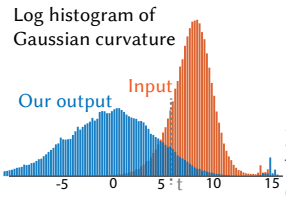
In Fig. 8 we show the effect of the one parameter of our method, λ, which controls the weight of the data fidelity term versus the nuclear norm term. The nuclear norm not only controls developabilty, but also the number of creases, due to an observation similar to one in [Stein et al. 2018a]: as resolution increases, the hexagonal stencil at the crease will produce a Hessian with increasing *maximal* curvature (*large* eigenvalue). Since the largest eigenvalue is included in the nuclear norm our energy naturally penalizes creases. As shown in Fig. 8, for low λ, the output is made up of a few developable pieces. As λ increases, so does the number of developable pieces and the deviation from developability, in favor of adherence to the input.

In Fig. 4 we show the effect of grid resolution on our method by approximating a heightfield of the Stanford Bunny with a developable one. Our output is nearly identical for the lowest resolutions and identical for the highest ones. This is due to 1) our convex optimization problem having a unique solution, and 2) our discretization, especially designed for agnosticism to tessellation parameters.

As shown in the statistics at the bottom of the image, our ADMM algorithm uses an almost-constant number of iterations for all resolutions, and our runtime scales linearly with mesh size. We find this to be representative of the behaviour we consistently witnessed.

An important question is whether our optimization indeed yields low-rank Hessians, and whether that indeed correlates to other measures of discrete developability. To that end, we measure developability using one of the most popular metrics for discrete developability – the angle deficit at each vertex. To get a consistent measure across mesh resolutions, we divide the angle deficit by the vertex's lumped area.

We visualize this discrete Gaussian curvature on Lucy in Fig. 25, with curvature intensity visualized in blue.



In Table 1 we further measure discrete gaussian curvature (g_k), as well as the direct underlying objective of our optimization (κ_2, the smaller eigenvalue of the Hessian). For both input and output, we show the percentage of vertices with g_k above an arbitrary threshold t (see "dragon"' histograms in the inset), and the median κ_2. In all cases the output has a very low κ_2 and only a small amount of vertices with g_k exceeding the threshold.

Finally, in Fig. 24 we empirically study the effect the boundaries have on the solution to our energy minimization problem. In this didactic example inspired by [Stein et al. 2018b]'s Figure 8, we observe that while our method is not completely independent of the boundaries, their effect appears to be small.

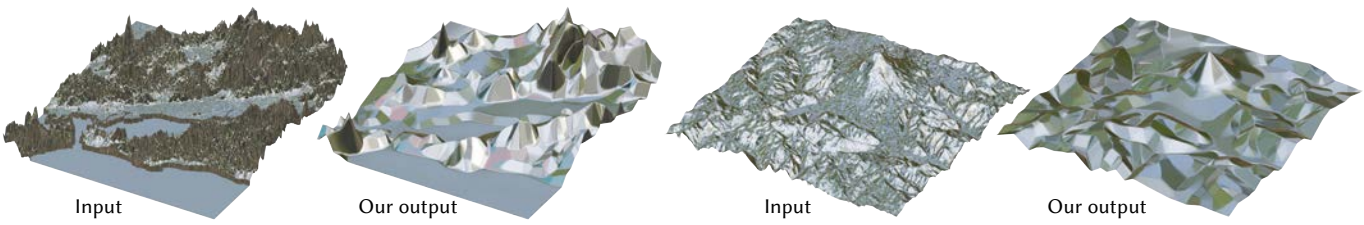


Fig. 23. Our method robustly handles highly-oscillatory data. San Francisco Bay Area topography model (left) by Waleed Kadous under CC BY 4.0 and pre-eruption Mt. Saint Helens model (right) by Jetty under CC BY-NC 4.0.

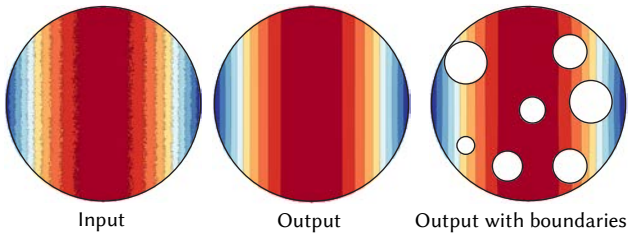


Fig. 24. Running our method on a noisy input (left) from which some holes have been removed adds little distorsion to our solution (right) compared to our method ran on the complete domain (middle).

Fig. 25. Our algorithm reduces Gaussian curvature, measured per each vertex as its angle deficit divided by its 1-ring area. Model from the Stanford 3D Scanning Repository.

6 LIMITATIONS AND FUTURE WORK

We have presented a method inspired by compressed sensing for recovering piecewise developable heightfields. Our convex optimization yields a global optimum, and is robust to input resolution, grid orientation, as well as noise. This enables us to perform developable denoising, as well as approximating arbitrary heightfields with developable ones. Our approach has many potential uses, from scanning of prints, approximating given models for metal-sheet construction, and designing developable structures. One geometrical application is to use the ruling lines - the eigenvectors of the second fundamental form - to segment a piecewise developable into its developable pieces.

Our main limitation is our restriction to heightfields. Moving to a 3D mesh, there is no longer a proxy for the second fundamental form which is linear in the variables, and we cannot formulate a convex problem. Thus our immediate future goal is to find alternative ways to transfer this method to 3D. One immediate approach to consider is to run our method simultaneously from multiple views.

A second limitation stems from the *density* of the space of piecewise developable surfaces; simply considering that *any* mesh is piecewise developable entails there is a piecewise developable an epsilon away from any given surface, and hence our recovery problem can seem as ill-posed. It is due to our optimization that penalizes creases that, *e.g.*, piecewise flat surfaces do not emerge. If the noise is too low-frequency with respect to the signal, we may recover a suboptimal developable reconstruction. This is the case in Fig. 26, in which the voxelization artifact is low frequency enough to cause our method to not reconstruct the smooth leg.

From the theoretical perspective, we note that our convex program is still a relaxation, which we have not proven is *exact*: it may be that there are piecewise developable height fields that are closer to the input than our output. To prove tightness requires compressed sensing machinery which is outside the scope of this paper, and we mark it as important future work.

Finally, preliminary segmentation results (see inset, which uses the output of Fig. 8) obtained with a flood-fill algorithm based on our rank proxy $\|H\|_*$ are promising towards making the application of our method to fabrication more direct.

We hope that our contribution of framing developability as a rank minimization problem can give a novel perspective to this familiar area of research in Computer Graphics, and that the many fabrication pipelines that rely on developability can be enriched by our work.

ACKNOWLEDGEMENTS

This project is funded in part by NSERC Discovery (RGPIN2017–05235, RGPAS–2017–507938), New Frontiers of Research Fund (NFRFE–201), the Ontario Early Research Award program, the Canada Research Chairs Program, the Fields Centre for Quantitative Analysis and Modelling and gifts by Adobe Systems, Autodesk and MESH Inc.

We thank Mirela Ben-Chen, Oded Stein and Derek Liu for their insightful conversation and advice; Herng Yi Cheng and Abhishek Madan for producing origami examples; Wenzheng Chen for his help with the 3D scanner setup in Fig. 10; Josh Holinaty, Naseem Hrab and Owlkids for allowing us to use their book in Fig. 3; and Honglin Chen, John Kanji, Ruiqi Wang and Rahul Arora for proofreading.

Fig. 26. Our method does not always recover the correct developable: the noise artifacts in the legs of the Tokyo Tower are preserved after running our method. 3D model by Michael Hill under CC BY 4.0.

Model	$K > t$ Input	$K > t$ Ours	$\tilde{\kappa}_2$ Input	$\tilde{\kappa}_2$ Ours
bergher	28.9 %	5.3 %	12.2	0.51
dragon	60.2 %	2.8 %	63.7	0.47
gargoyle	24.6 %	4.4 %	14.9	0.13
lucy	38.8 %	7.0 %	24.7	1.13
range	34.5 %	1.1 %	13.6	0.02
woman	18.2 %	9.5 %	17.1	0.05
bunny	7.9 %	1.1 %	2.1	0.06
einstein	35.1 %	2.9 %	68.4	0.57
fandisk	2.6 %	1.5 %	0.1	0.03
book	23.7 %	2.7 %	19.7	0.07
notepad	12.3 %	8.1 %	11.4	0.03
paper	95.9 %	1.9 %	899.7	0.04
disney	35.1 %	4.3 %	7373.3	0.20
tower	27.2 %	5.8 %	0.0	0.26
bench	27.4 %	2.1 %	0.0	0.33
oloid	27.4 %	1.3 %	0.0	0.04
bilbao	31.3 %	10.1 %	0.0	1.22

Table 1. Quanitative evaluation of our method. For input and output of each example in the paper, we show the median of κ_2, denoted $\tilde{\kappa}_2$ – the smaller-magnitude eigenvalue of the Hessiann and the target objective our convex optimization aims to minimize. Likewise, we inluce a developability metric we don't directly optimize – K – the discrete Gaussian curvature, measured by angle deficit over vertex area. We show the percentage of vertices with curvature below a threshold t (see inset histograms)

REFERENCES

Jascha Achenbach, Eduard Zell, and Mario Botsch. 2015. Accurate face reconstruction through anisotropic fitting and eye correction. (2015).

Noam Aigerman and Yaron Lipman. 2013. Injective and Bounded Distortion Mappings in 3D. *ACM Trans. Graph.* 32, 4, Article Article 106 (July 2013), 14 pages. https://doi.org/10.1145/2461912.2461931

Gerasimos Arvanitis, Aris S Lalos, and Konstantinos Moustakas. 2019. Denoising of dynamic 3D meshes via low-rank spectral analysis. *Computers & Graphics* (2019).

Pengbo Bo and Wenping Wang. 2007. Geodesic-Controlled Developable Surfaces for Modeling Paper Bending. *Comput. Graph. Forum* 26, 3 (2007), 365–374.

Sofien Bouaziz, Andrea Tagliasacchi, and Mark Pauly. 2013. Sparse Iterative Closest Point. *Comput. Graph. Forum* 32, 5 (2013), 1–11.

Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. 2011. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning* 3, 1 (2011), 1–122.

Stephen Boyd and Lieven Vandenberghe. 2004. *Convex optimization.*

Derek Bradley. 2006. Deforming Developable Surfaces.

Christopher Brandt and Klaus Hildebrandt. 2017. Compressed vibration modes of elastic bodies. *Computer Aided Geometric Design* 52 (2017), 297–312.

Emmanuel J. Candès, Xiaodong Li, Yi Ma, and John Wright. 2011. Robust principal component analysis? *J. ACM* 58, 3 (2011), 11:1–11:37.

Emmanuel J Candès and Michael B Wakin. 2008. An introduction to compressive sampling [a sensing/sampling paradigm that goes against the common knowledge in data acquisition]. *IEEE signal processing magazine* 25, 2 (2008), 21–30.

H.-Y. Chen, I.-K. Lee, Stefan Leopoldseder, Helmut Pottmann, Thomas Randrup, and Johannes Wallner. 1999. On Surface Approximation Using Developable Surfaces. *Graphical Models and Image Processing* 61, 2 (1999), 110–124.

Richard Courant and David Hilbert. 2008. *Methods of Mathematical Physics: Partial Differential Equations.* John Wiley & Sons.

José Paulo R. de Lima and Helton Hideraldo Bíscaro. 2015. Compressive Representation of Three-dimensional Models. *SBC Journal on Interactive Systems* (2015).

Philippe Decaudin, Dan Julius, Jamie Wither, Laurence Boissieux, Alla Sheffer, and Marie-Paule Cani. 2006. Virtual Garments: A Fully Geometric Approach for Clothing Design. *Comput. Graph. Forum* 25, 3 (2006), 625–634.

Stephan Didas and Joachim Weickert. 2004. Higher order variational methods for noise removal in signals and images. *Saarbrücken, Saarland Universitesi* (2004).

Levi Dudte, Etienne Vouga, Tomohiro Tachi, and Lakshminarayanan Mahadevan. 2016. Programming curvature using origami tessellations. *Nature Materials* 15 (01 2016).

Maryam Fazel, Haitham Hindi, Stephen P Boyd, et al. 2001. A rank minimization heuristic with application to minimum order system approximation. In *Proceedings of the American control conference*, Vol. 6. Citeseer, 4734–4739.

William H Frey. 2004. Modeling buckled developable surfaces by triangulation. *Computer-Aided Design* 36, 4 (2004), 299–313.

Lei He and Scott Schaefer. 2013. Mesh Denoising via L0 Minimization. *ACM Trans. Graph.* 32, 4, Article Article 64 (July 2013), 8 pages. https://doi.org/10.1145/2461912.2461965

Michael Hofer, Boris Odehnal, Helmut Pottmann, Tibor Steiner, and Johannes Wallner. 2005. 3D Shape Recognition and Reconstruction Based on Line Element Geometry. In *Proc. ICCV*. 1532–1538.

Jin Huang, Tengfei Jiang, Zeyun Shi, Yiying Tong, Hujun Bao, and Mathieu Desbrun. 2014. L1-Based Construction of Polycube Maps from Complex Shapes. *ACM Trans. Graph.* (2014).

Alec Jacobson et al. 2016. gptoolbox: Geometry Processing Toolbox. http://github.com/alecjacobson/gptoolbox.

Alec Jacobson, Daniele Panozzo, et al. 2018. libigl: A simple C++ geometry processing library. http://libigl.github.io/libigl/.

Dan Julius, Vladislav Kraevoy, and Alla Sheffer. 2005. D-charts: Quasi-developable mesh segmentation. In *Computer Graphics Forum*, Vol. 24. Wiley Online Library, 581–590.

Amaury Jung, Stefanie Hahmann, Damien Rohmer, Antoine Bégault, Laurence Boissieux, and Marie-Paule Cani. 2015. Sketching Folds: Developable Surfaces from Non-Planar Silhouettes. *ACM Trans. Graph.* 34, 5 (2015), 155:1–155:12.

Yannick L. Kergosien, Hironobu Gotoda, and Tosiyasu L. Kunii. 1994. Bending and creasing virtual paper. *IEEE Computer Graphics and Applications* 14, 1 (1994), 40–48.

Martin Kilian, Simon Flöry, Zhonggui Chen, Niloy J Mitra, Alla Sheffer, and Helmut Pottmann. 2008. Curved folding. In *ACM transactions on graphics (TOG)*, Vol. 27. ACM, 75.

Kai-Wah Lee and Pengbo Bo. 2016. Feature curve extraction from point clouds via developable strip intersection. *J. Computational Design and Engineering* 3, 2 (2016), 102–111.

Stefan Leopoldseder and Helmut Pottmann. 1998. Approximation of developable surfaces with cone spline surfaces. *Computer-Aided Design* 30, 7 (1998), 571–582.

Yaron Lipman, Daniel Cohen-Or, David Levin, and Hillel Tal-Ezer. 2007. Parameterization-free projection for geometry reconstruction. *ACM Trans. Graph.* 26, 3 (2007), 22.

Hsueh-Ti Derek Liu and Alec Jacobson. 2019. Cubic Stylization. *ACM Trans. Graph.* (2019).

Ligang Liu, Lei Zhang, Yin Xu, Craig Gotsman, and Steven J Gortler. 2008. A local/global approach to mesh parameterization. In *Computer Graphics Forum*, Vol. 27. Wiley Online Library, 1495–1504.

Yang Liu, Helmut Pottmann, Johannes Wallner, Yong-Liang Yang, and Wenping Wang. 2006. Geometric modeling with conical meshes and developable surfaces. *ACM Trans. Graph.* 25, 3 (2006), 681–689.

Xuequan Lu, Scott Schaefer, Jun Luo, Lizhuang Ma, and Ying He. 2018. Low Rank Matrix Approximation for Geometry Filtering. *CoRR* abs/1803.06783 (2018). arXiv:1803.06783 http://arxiv.org/abs/1803.06783

Tomohiko Mukai and Shigeru Kuriyama. 2016. Efficient dynamic skinning with low-rank helper bone controllers. *ACM Trans. Graph.* (2016).

Rahul Narain, Tobias Pfaff, and James F. O'Brien. 2013. Folding and crumpling adaptive sheets. *ACM Trans. Graph.* 32, 4 (2013), 51:1–51:8.

Barrett O'Neill. 1966. *Elementary differential geometry.*

Maodong Pan, Weihua Tong, and Falai Chen. 2016. Compact implicit surface reconstruction via low-rank tensor approximation. *Computer-Aided Design* 78 (2016), 158–167. https://doi.org/10.1016/j.cad.2016.05.007
Mathieu Perriollat and Adrien Bartoli. 2013. A computational model of bounded developable surfaces with application to image-based three-dimensional reconstruction. *Journal of Visualization and Computer Animation* 24, 5 (2013), 459–476.
Martin Peternell. 2004. Developable surface fitting to point clouds. *Computer Aided Geometric Design* 21 (2004), 785–803.
Helmut Pottmann and Gerald E. Farin. 1995. Developable rational Bézier and B-spline surfaces. *Computer Aided Geometric Design* 12, 5 (1995), 513–531.
Helmut Pottmann, Alexander Schiftner, Pengbo Bo, Heinz Schmiedhofer, Wenping Wang, Niccolo Baldassini, and Johannes Wallner. 2008. Freeform surfaces from single curved panels. *ACM Trans. Graph.* 27, 3 (2008), 76.
Helmut Pottmann and Johannes Wallner. 1999. Approximation algorithms for developable surfaces. *Computer Aided Geometric Design* 16, 6 (1999), 539–556.
Michael Rabinovich, Tim Hoffmann, and Olga Sorkine-Hornung. 2018a. Discrete geodesic nets for modeling developable surfaces. *ACM Transactions on Graphics (ToG)* 37, 2 (2018), 16.
Michael Rabinovich, Tim Hoffmann, and Olga Sorkine-Hornung. 2018b. The shape space of discrete orthogonal geodesic nets. In *SIGGRAPH Asia 2018 Technical Papers*. ACM, 228.
Michael Rabinovich, Tim Hoffmann, and Olga Sorkine-Hornung. 2019. Modeling curved folding with freeform deformations. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 170.
P Redont. 1989. Representation and deformation of developable surfaces. *Computer-Aided Design* 21, 1 (1989), 13 – 20.
Kenneth Rose, Alla Sheffer, Jamie Wither, Marie-Paule Cani, and Boris Thibert. 2007. Developable surfaces from arbitrary sketched boundaries. In *SGP'07-5th Eurographics Symposium on Geometry Processing*. Eurographics Association, 163–172.
Leonid I. Rudin, Stanley Osher, and Emad Fatemi. 1992. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena* 60, 1 (1992), 259 – 268.
Raif M. Rustamov. 2011. Multiscale Biharmonic Kernels. *Comput. Graph. Forum* 30, 5 (2011).
Camille Schreck, Damien Rohmer, Stefanie Hahmann, Marie-Paule Cani, Shuo Jin, Charlie C. L. Wang, and Jean-Francis Bloch. 2015. Nonsmooth Developable Geometry for Interactively Animating Paper Crumpling. *ACM Trans. Graph.* 35, 1 (2015), 10:1–10:18.
Justin Solomon, Etienne Vouga, Max Wardetzky, and Eitan Grinspun. 2012. Flexible developable surfaces. In *Computer Graphics Forum*, Vol. 31. Wiley Online Library, 1567–1576.
Oded Stein, Eitan Grinspun, and Keenan Crane. 2018a. Developability of Triangle Meshes. *ACM Trans. Graph.* 37, 4 (2018).
Oded Stein, Eitan Grinspun, Max Wardetzky, and Alec Jacobson. 2018b. Natural Boundary Conditions for Smoothing in Geometry Processing. *ACM Trans. Graph.* (2018).
B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd. 2017. OSQP: An Operator Splitting Solver for Quadratic Programs. *ArXiv e-prints* (Nov. 2017). arXiv:math.OC/1711.08013
Jacob Subag and Gershon Elber. 2006. Piecewise Developable Surface Approximation of General NURBS Surfaces, with Global Error Bounds. In *Proc. GMP*.
Chengcheng Tang, Pengbo Bo, Johannes Wallner, and Helmut Pottmann. 2016. Interactive design of developable surfaces. *ACM Transactions on Graphics (TOG)* 35, 2 (2016), 12.
Etienne Vouga, Mathias Höbinger, Johannes Wallner, and Helmut Pottmann. 2012. Design of Self-supporting Surfaces. *ACM Trans. Graph.* (2012).
Hui Wang, Davide Pellis, Florian Rist, Helmut Pottmann, and Christian Müller. 2019. Discrete geodesic parallel coordinates. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 173.
Mingqiang Wei, Jin Huang, Xingyu Xie, Ligang Liu, Jun Wang, and Jing Qin. 2019. Mesh Denoising Guided by Patch Normal Co-Filtering via Kernel Low-Rank Recovery. *IEEE TVCG* 25, 10 (2019).
Jie Zhang, Junjie Cao, Xiuping Liu, Jun Wang, Jian Liu, and Xiquan Shi. 2013. Point cloud normal estimation via low-rank subspace clustering. *Computers & Graphics* 37, 6 (2013), 697–706.
Juyong Zhang, Bailin Deng, Yang Hong, Yue Peng, Wenjie Qin, and Ligang Liu. 2019. Static/Dynamic Filtering for Mesh Geometry. *IEEE TVCG* (2019).
Juyong Zhang, Bailin Deng, Zishun Liu, Giuseppe Patanè, Sofien Bouaziz, Kai Hormann, and Ligang Liu. 2014. Local Barycentric Coordinates. *ACM Trans. Graph.* 33 (2014).

A RANK EQUIVALENCY PROOF

Lemma A.1. *Let $z : \mathbb{R}^2 \longrightarrow \mathbb{R}$ be a continuous, twice-differentiable function and let $\mathcal{M}$ be the two-dimensional surface defined by the graph of z. Let $p \in \mathcal{M}$ be a point of the form $(x_0, y_0, z(x_0, y_0))$. Then, the second fundamental form of $\mathcal{M}$ at p when seen as a matrix $\mathbb{I}_p$ is proportional to the Hessian matrix $H_z(x_0, y_0)$.*

Proof. $\mathcal{M}$ is parametrized by $\mathbf{r}(x, y) = (x, y, z(x, y))$. The second fundamental form in matrix form is given by

$$\mathbb{I} = \begin{pmatrix} L & M \\ M & N \end{pmatrix}, \tag{21}$$

where

$$L = \frac{\partial^2 \mathbf{r}}{\partial x^2} \cdot \mathbf{n}, \quad M = \frac{\partial^2 \mathbf{r}}{\partial x \partial y} \cdot \mathbf{n}, \quad N = \frac{\partial^2 \mathbf{r}}{\partial y^2} \cdot \mathbf{n}, \tag{22}$$

and $\mathbf{n} = (n_1, n_2, n_3)$ is the normal vector. In our case this becomes

$$\mathbb{I} = n_3 \begin{pmatrix} \frac{\partial^2 z}{\partial x^2} & \frac{\partial^2 z}{\partial x \partial y} \\ \frac{\partial^2 z}{\partial y \partial x} & \frac{\partial^2 z}{\partial y^2} \end{pmatrix}, \tag{23}$$

which is nothing but

$$\mathbb{I} = n_3 H_z \,. \tag{24}$$

Since n_3 cannot be zero on a surface described as a graph, this proves the lemma.

□

B BEST FIT HESSIAN AS A LINEAR TRANSFORMATION

The best fit coefficients of the quadratic functions minimize

$$\min_{\substack{c \in \mathbb{R} \\ \mathbf{g} \in \mathbb{R}^2 \\ \mathbf{H}=\mathbf{H}^\top \in \mathbb{R}^{2\times 2}}} \frac{1}{2} \sum_{i=1}^{7} \left\| c_4 + \mathbf{g}^\top (\mathbf{x}_i - \mathbf{x}_4) + \frac{1}{2} (\mathbf{x}_i - \mathbf{x}_4)^\top \mathbf{H} (\mathbf{x}_i - \mathbf{x}_4) - z_i \right\|^2,$$

which can be written as

$$\min_{\mathbf{c} \in \mathbb{R}^6} \frac{1}{2} \|\mathbf{A}\mathbf{c} - \mathbf{z}\|_F^2 \tag{25}$$

where $\mathbf{A} \in \mathbb{R}^{7\times 6}$ contains constant coefficients that ultimately do not even depend on which vertex is considered (only h), and $\mathbf{z}$ stack the seven involved height values in order. The solution is revealed via solving the normal equations:

$$\mathbf{c} = \underbrace{(\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}}_{\mathbf{B}} \mathbf{z}, \tag{26}$$

where the fixed local stencil matrix $\mathbf{B} \in \mathbb{R}^{6\times 7}$ is revealed to be:

$$\mathbf{B} = \frac{1}{6h^2} \begin{pmatrix} 0 & 0 & 3 & -6 & 3 & 0 & 0 \\ 2 & 2 & -1 & -6 & -1 & 2 & 2 \\ -2\sqrt{3} & 2\sqrt{3} & 0 & 0 & 0 & 2\sqrt{3} & -2\sqrt{3} \\ -h & h & -2h & 0 & 2h & -h & h \\ \sqrt{3}h & \sqrt{3}h & 0 & 0 & 0 & -\sqrt{3}h & -\sqrt{3}h \\ 0 & 0 & 0 & 6h^2 & 0 & 0 & 0 \end{pmatrix}. \tag{27}$$

C WRITING OUR MINIMIZATION AS AN SDP

Let us introduce the auxiliary symmetric matrix variables $\mathbf{X}_i = \mathbf{X}_i^\top, \mathbf{Y}_i = \mathbf{Y}_i^\top \in \mathbb{R}^{2\times 2}$ for each interior vertex and rewrite the problem as a semi-definite program:

$$\min_{\mathbf{z},\mathbf{X},\mathbf{Y}} \lambda \sum_{i\in\mathcal{I}\cup\partial\mathcal{I}} \|z_i - \tilde{z}_i\|^2 + \sum_{i\in\mathcal{I}} \operatorname{tr}\mathbf{X}_i + \operatorname{tr}\mathbf{Y}_i, \tag{28}$$

$$\text{subject to } \mathbf{Cz} = \mathbf{h},$$

$$\text{and } h^i_{xy} = h^i_{yx}\ \forall i \in \mathcal{I} \tag{29}$$

$$\text{and } \begin{bmatrix} \mathbf{X}_i & \begin{matrix} h^i_{xx} & h^i_{xy} \\ h^i_{xy} & h^i_{yy} \end{matrix} \\ \begin{matrix} h^i_{xx} & h^i_{xy} \\ h^i_{xy} & h^i_{yy} \end{matrix} & \mathbf{Y}_i \end{bmatrix} \succcurlyeq 0\ \forall\ i \in \mathcal{I},$$

where $\mathbf{M} \succcurlyeq 0$ indicates that the matrix $\mathbf{M}$ is positive semi-definite.

D CLOSED FORM SOLUTION TO THREE-VARIABLE SEMI-DEFINITE PROGRAM

In Equation (18), we need to solve (many) small semi-definite programs of the form:

$$\mathbf{H}^* = \operatorname*{argmin}_{\mathbf{H}\in\mathbb{R}^{2\times 2}} \|\mathbf{H}\|_* + \frac{\rho}{2}\|\mathbf{H} - \mathbf{G}\|_F^2, \tag{30}$$

where the matrix $\mathbf{G} \in \mathbb{R}^{2\times 2}$ (gathering the relevant terms involving $\mathbf{C}, \mathbf{z}, \mathbf{u}$ in Equation (18)) is constant with respect to the unknowns in $\mathbf{H}$.

Let $\mathbf{G} = U\Sigma V^T$ be the Singular Value Decomposition of $\mathbf{G}$, with $\Sigma = \operatorname{diag}(\sigma_1, \sigma_2) \in \mathbb{R}^{2\times 2}$. Let A be an optimum of the above program. Consider the equivalent (*i.e.*, has the same set of optimal solutions) problem

$$\mathbf{H}^* = \operatorname*{argmin}_{\mathbf{H}\in\mathbb{R}^{2\times 2},\, \|\mathbf{H}\|_* = \|A\|_*} \frac{\rho}{2}\|\mathbf{H} - \mathbf{G}\|_F^2. \tag{31}$$

Retracing the proof of Theorem 1 in [Aigerman and Lipman 2013], letting the space $\left\{\mathbf{H} \in \mathbb{R}^{2\times 2}, \|\mathbf{H}\|_* = \|\mathbf{H}^*\|_*\right\}$ replace the space denoted there as $\mathcal{T}_k$ (the proof applies to any space of matrices defined in terms of singular values), we get that A's singular vectors are the same as G's,

$$A = UDV^T, \tag{32}$$

for some unknown diagonal matrix D.

Plugging UDV^T instead of H in the above minimization, and using both norms' invariance to multiplication by orthogonal matrices, we get that the optimal D is the diagonal matrix minimizing

$$\mathbf{D}^* = \operatorname*{argmin}_{\mathbf{D}} \|\mathbf{D}\|_* + \frac{\rho}{2}\|\mathbf{D} - \Sigma\|_F^2. \tag{33}$$

Writing $\mathbf{D} = \operatorname{diag}(d_1, d_2)$, we consider each entry independently:

$$d_i^* = \operatorname*{argmin}_{d_i} |d_i| + \frac{\rho}{2}|d_i - \sigma_i|, \tag{34}$$

whose minimum is simply:

$$d_i^* = \max\left(\sigma_i - \tfrac{1}{\rho}, 0\right). \tag{35}$$

To summarize, the optimal $\mathbf{H}^*$ is found by computing the singular value decomposition of $\mathbf{G}$, computing the diagonal entries of $\mathbf{D}^*$ according to Equation (35), and then constructing $\mathbf{H}^* = \mathbf{U}\mathbf{D}^*\mathbf{V}^\top$. In our optimization, $\mathbf{G}$ will always be symmetric by construction, so $\mathbf{U} = \mathbf{V}$ and the resulting $\mathbf{H}^*$ will be symmetric, fulfilling the constraint that $h_{xy} = h_{yx}$.