

# Digital Calculus Frameworks and Comparative Evaluation of their Laplace-Beltrami operators<sup>\*</sup>

Colin Weill-Duflos<sup>1</sup> [0009-0009-9553-0779], David  
Coeurjolly<sup>2</sup> [0000-0003-3164-8697], and Jacques-Olivier  
Lachaud<sup>1</sup> [0000-0003-4236-2133]

<sup>1</sup> Université Savoie Mont Blanc, CNRS, LAMA, F-73000 Chambéry, France

{colin.weill-duflos|jacques-olivier.lachaud}@univ-smb.fr

<sup>2</sup> Univ Lyon, CNRS, Lyon1, INSA, LIRIS, France

david.coeurjolly@cnrs.fr

**Abstract.** Defining consistent calculus frameworks on discrete meshes is useful for processing the geometry of meshes or model numerical simulations and variational problems onto them. However digital surfaces (boundary of voxels) cannot benefit directly from the classical mesh calculus frameworks, since their vertex and face geometry is too poor to capture the geometry of the underlying smooth Euclidean surface well enough. This paper proposes two new calculus frameworks dedicated to digital surfaces, which exploit a *corrected normal field*, in a manner similar to the recent digital calculus of [3]. First we build a *corrected interpolated calculus* by defining inner products with position and normal interpolation in the Grassmannian. Second we present a *corrected finite element method* which adapts the standard Finite Element Method with a corrected metric per element. Experiments show that these digital calculus frameworks seem to converge toward the continuous calculus, offer a valid alternative to classical mesh calculus, and induce effective tools for digital surface processing tasks.

**Keywords:** Digital calculus · Laplacian operator · Differential operators

## 1 Introduction

When solving differential equations on a mesh, it is often required to build a set of differential operators for this mesh. Perhaps the most commonly found is the Laplace-Beltrami operator as it is used in a wide variety of applications such as mesh editing [17,14], mesh smoothing [15] or geodesic path approximation [5]. Building a simple graph Laplacian or discrete Laplacian does not suffice, since the mesh geometry must be taken into account. Using a subdivision scheme and building the operators on it (as done in [7]) do not suffice either, as the limit surface does not solve the metric issues (staircase effects induced by the grid). On triangular and polygonal surfaces, several calculus frameworks produce these differential operators, such as the Finite Element Method (FEM) [16], Discrete

---

<sup>\*</sup> This work was partly funded by STABLEPROXIES ANR-22-CE46-0006 research grant.

Exterior Calculus (DEC) [9], the Virtual Element Method [18], etc (see [1] for a comparative evaluation).

Usually these frameworks operate under the assumption that the mesh interpolates the underlying "true" smooth geometry. In the case of digital surfaces made of surfels (boundary of voxels), which are frequent when processing 3D images, this assumption is false, and these frameworks fail at yielding convergent operators. However several geometric quantities can be evaluated with convergence properties, such as surface area [13], or the normal field and the curvature tensor [10,11] on digital surfaces. We are aware of only two digital analogues to differential operators: Caissard *et al.* [2] proposed a digital Laplacian based on the heat kernel, while two of the authors have adapted in [3] the polygonal calculus of [6], by correcting its normal vector field. The digital "Heat kernel" Laplacian of [2] is the only one that is proven convergent and its convergence is observed through experiments. The digital "Projected PolyDEC" Laplacian of [3] is not pointwise convergent, but yet provide meaningful results in variational problems.

This paper proposes two new digital calculus frameworks, that are constructed with a tangent space corrected by a prescribed normal vector field (e.g., the II normal estimator [10]). Tangent space correction has proven to be effective for tasks such as estimating curvatures [11] and reconstructing a piecewise smooth surface from a digital surface [4]. The first one, called "interpolated corrected calculus", embeds the digital surface into the Grassmannian with a vertex-interpolated corrected normal vector field: the resulting surface is thus continuous in positions and normals. It is thus more consistent than the Projected PolyDEC, whose embedding is discontinuous between surfels. The second one, called "corrected FEM", adapts the Finite Element Method with metrics tailored to a constant corrected normal vector per element. Both constructions are consistent with classical calculus constructions, and we hope they will allow to prove the convergence of operators. For now, we conducted experiments which show that these frameworks build a consistent Laplacian, convergent when slightly diffused. We achieve results on par with [2] while retaining the ease of build and sparsity from [3].

## 2 Digital calculus with corrected tangent space

We demonstrate here that the same approach of corrected lengths and areas used in [3] can be used to build differential operators with other methods. The approach can be summarized as a correction of lengths and areas based on how orthogonal they are to the true normal. Assuming we have a vector  $\mathbf{v}$  and a normal  $\mathbf{u}$ , the corrected length of  $\mathbf{v}$  is given by  $\|\mathbf{v} \times \mathbf{u}\|$ . To correct the area of a parallelogram defined by two vectors  $\mathbf{v}$  and  $\mathbf{w}$  with normal  $\mathbf{u}$  is given by  $\det(\mathbf{u}, \mathbf{v}, \mathbf{w})$ . These can be seen as the length/areas of the projected vector/parallelogram onto the tangent plane.

Our data here will be defined by values at vertices, meaning that each face has 4 degrees of freedom. We use these degrees of freedom to build a base of

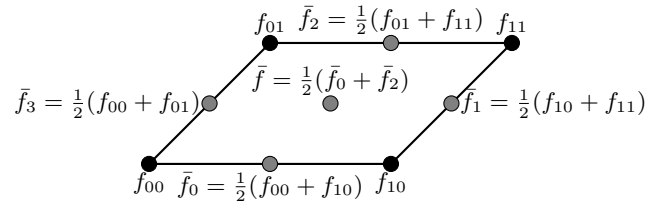


Fig. 1: Notations for the interpolations of function  $f$  values on a surfel.

functions on the mesh. These functions are bilinear on mesh elements here but other basis functions are possible (e.g. the Virtual Elements Method [18] requires solely the behavior of functions on edges). The methods we present, similarly to [3], use a per face construction of sparse operators.

*Notation* The parameter space of each surfel is a unit square  $\square := [0, 1]^2$  parameterized by  $s$  and  $t$ . We denote by  $\mathbf{n}$  the natural or naive normal of a surfel  $\sigma$ , that can be computed with a cross product of two consequent edges. The corrected normal field will be denoted  $\mathbf{u}$ . Inside a surfel, we can decompose this surfel in the natural base of the surfel into  $\mathbf{u} = (\mathbf{u}^x, \mathbf{u}^y, \mathbf{u}^z)$ .

A function  $f$  in a surfel  $\sigma$  is assumed to be bilinearly interpolated. We denote then by  $[f_{\square}(\sigma)] := [f_{00}(\sigma), f_{10}(\sigma), f_{11}(\sigma), f_{01}(\sigma)]^T$  the degrees of freedom of  $f$ , corresponding to its values at each vertex when circulating around  $\sigma$ . We will often write simply  $[f_{\square}]$  when the surfel is obvious from the context. We sometimes use averages of these values, whose notations are illustrated in Figure 1.

## 2.1 Interpolated corrected calculus

We propose here a calculus where the corrected normal vector field  $\mathbf{u}(x)$  is continuous over the mesh: corrected normal vectors are given at vertices; these vectors are bilinearly interpolated within each face. Hence, within a surfel,  $\mathbf{u}(s, t) = \mathbf{u}_{00}(1-s)(1-t) + \mathbf{u}_{10}s(1-t) + \mathbf{u}_{01}(1-s)t + \mathbf{u}_{11}st$ . Although this naive bilinear interpolation does not respect the condition that normals need to be unitary vectors, it yields much simpler formulas in calculation. Furthermore, experiments show that a more complex interpolation yielding almost unit normals does not improve the results, while increasing the complexity of formulas.

The construction of the calculus is similar to the polygonal calculus of [6], building inner products, sharp and flat operators on a per face basis. However the correction of the geometry does not follow [3], but instead use an embedding of the mesh into the Grassmannian to correct the area/length measures. The Grassmannian is a way to represent affine subspaces, hence tangent spaces here. Within this space, one can define differential forms that are invariant to rigid motions (Lipschitz-Killing forms). We exploit here the *corrected area 2-form* (see [12, 11]):  $\omega_0^{\mathbf{u}}(\mathbf{x})(\mathbf{v}, \mathbf{w}) := \det(\mathbf{u}(\mathbf{x}), \mathbf{v}, \mathbf{w})$ , for  $\mathbf{v}$  and  $\mathbf{w}$  tangent vectors. As one can see, thanks to the embedding in the Grassmannian, the corrected area form

can be expressed as a simple volume form (i.e. a determinant). Note that it falls back to the usual area measure  $\|\mathbf{v} \times \mathbf{w}\|$  when  $\mathbf{v}$  and  $\mathbf{w}$  are indeed orthogonal to a unit normal vector  $\mathbf{u}(\mathbf{x})$ , while it gets smaller if there is a mismatch between tangent and normal information.

We first define how we integrate a quantity  $g$  defined at vertices. In the case of a surfel  $\sigma$  with constant normal  $\mathbf{n}$  aligned with  $z$ -axis wlog, and with  $\mathbf{v} = \frac{\partial \mathbf{x}}{\partial s}$  and  $\mathbf{w} = \frac{\partial \mathbf{x}}{\partial t}$ , the corrected area form reduced on  $\square$  to  $\omega_0^{\mathbf{u}}(s, t) = \langle \mathbf{n} \mid \mathbf{u}(s, t) \rangle = \mathbf{u}^z(s, t)$ . We can now compute the integral of  $g$  inside a surfel:

$$\iint_{\square} g \omega_0^{\mathbf{u}} := \iint_{\square} g(s, t) \mathbf{u}^z(s, t) ds dt = [\mathbf{u}_{\square}^z]^{\top} \frac{1}{36} \begin{bmatrix} 4 & 2 & 1 & 2 \\ 2 & 4 & 2 & 1 \\ 1 & 2 & 4 & 2 \\ 2 & 1 & 2 & 4 \end{bmatrix} [\phi_{\square}].$$

We study now the integral quantity  $\iint \nabla \Phi \omega_0^{\mathbf{u}}$ , which is an integrated gradient corrected by the normal vector field  $\mathbf{u}$ . First of all, the scalar field  $\Phi$  will generally be defined as the bilinear interpolation of a scalar field  $\phi$  defined over the domain. Thus  $\phi(s, t) = \Phi(\mathbf{x}(s, t))$ . We relate the gradient of  $\Phi$  with the partial derivatives of  $\phi$  by writing the standard chain rule with Jacobian matrices:

$$J_{\phi}(s, t) = J_{\Phi}(\mathbf{x}(s, t)) J_{\mathbf{x}}(s, t) \Leftrightarrow \left[ \frac{\partial \phi}{\partial s} \quad \frac{\partial \phi}{\partial t} \right] (s, t) = (\nabla \Phi)^T(\mathbf{x}(s, t)) \left[ \frac{\partial \mathbf{x}}{\partial s} \quad \frac{\partial \mathbf{x}}{\partial t} \right] (s, t).$$

We quite naturally extend  $\phi$  as constant along the  $\mathbf{u}$  direction. The preceding relation can now be inverted given that  $(\frac{\partial \mathbf{x}}{\partial s} = [1 \ 0 \ 0]^T, \frac{\partial \mathbf{x}}{\partial t} = [0 \ 1 \ 0]^T, \mathbf{u} = [\mathbf{u}^x \ \mathbf{u}^y \ \mathbf{u}^z]^T)$  forms a basis ( $(s, t)$  is omitted for conciseness):

$$\nabla \Phi(\mathbf{x}) = \underbrace{\begin{bmatrix} \mathbf{u}^z & 0 & 0 \\ 0 & \mathbf{u}^z & 0 \\ -\mathbf{u}^x & -\mathbf{u}^y & 1 \end{bmatrix}}_C \begin{bmatrix} \frac{\partial \phi}{\partial s} \\ \frac{\partial \phi}{\partial t} \\ 0 \end{bmatrix}.$$

It follows that  $\iint_{\square} \nabla \Phi \omega_0^{\mathbf{u}} = \iint_{\square} C \left[ \frac{\partial \phi}{\partial s} \quad \frac{\partial \phi}{\partial t} \quad 0 \right]^{\top} \mathbf{u}^z ds dt$ . Below, we explicit the vector  $\left[ \frac{\partial \phi}{\partial s} \quad \frac{\partial \phi}{\partial t} \quad 0 \right]^{\top}$  involving derivatives of  $\phi$  as

$$\begin{bmatrix} (1-t)(\phi_{10} - \phi_{00}) + t(\phi_{11} - \phi_{01}) \\ (1-s)(\phi_{01} - \phi_{00}) + s(\phi_{11} - \phi_{10}) \\ 0 \end{bmatrix} = \underbrace{\begin{bmatrix} 1-t & 0 & -t & 0 \\ 0 & s & 0 & s-1 \\ 0 & 0 & 0 & 0 \end{bmatrix}}_B \underbrace{\begin{bmatrix} -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \\ 1 & 0 & 0 & -1 \end{bmatrix}}_{D_0} [\phi_{\square}].$$

Matrix  $D_0$  is the differential operator, and is common to all quad faces. We get:

$$\iint_{\square} \nabla \Phi \omega_0^{\mathbf{u}} = \underbrace{\iint_{\square} C B \mathbf{u}^z ds dt}_{\mathcal{G}_{\sigma}} D_0 [\phi_{\square}],$$

where  $\mathcal{G}_\sigma$  is a  $3 \times 4$  matrix whose expression is (note the use of averages):

$$\mathcal{G}_\sigma = \frac{1}{3} \begin{bmatrix} \bar{\mathbf{u}}^z & 0 & -\bar{\mathbf{u}}^z & 0 \\ 0 & \bar{\mathbf{u}}^z & 0 & -\bar{\mathbf{u}}^z \\ -\bar{\mathbf{u}}^x & -\bar{\mathbf{u}}^y & \bar{\mathbf{u}}^x & \bar{\mathbf{u}}^y \end{bmatrix} + \frac{1}{6} \begin{bmatrix} \bar{\mathbf{u}}_0^z & 0 & -\bar{\mathbf{u}}_2^z & 0 \\ 0 & \bar{\mathbf{u}}_1^z & 0 & -\bar{\mathbf{u}}_3^z \\ -\bar{\mathbf{u}}_0^x & -\bar{\mathbf{u}}_1^y & \bar{\mathbf{u}}_2^x & \bar{\mathbf{u}}_3^y \end{bmatrix}.$$

The (corrected) area  $a_\sigma$  of such a surfel  $\sigma$  has a simple expression, while a pointwise expression of the gradient  $\mathbf{G}_\sigma$  is obtained by normalizing  $\mathcal{G}_\sigma$  by the corrected area leading to:

$$a_\sigma := \iint_{\square} \omega_0^{\mathbf{u}} = \iint_{\square} \mathbf{u}^z dsdt = \bar{\mathbf{u}}^z, \quad \mathbf{G}_\sigma := \frac{1}{a_\sigma} \mathcal{G}_\sigma D_0.$$

*Sharp and flat operators.* The sharp operator transform a 1-form into a vector field. We use the expression of the pointwise gradient to raise any 1-form as a representative vector per surfel. Within a surfel, a 1-form associates a scalar value to each (oriented) edge. Let  $\beta$  be 1-form, and  $[\beta_{\mathfrak{E}}(\sigma)] := [\beta_0 \beta_1 \beta_2 \beta_3]^\top$  its values on the 4 edges of  $\sigma$ . Omitting the differential operator  $D_0$  in the pointwise gradient gives the representative 3D vector of  $\beta$  on surfel  $\sigma$ :

$$\beta^\sharp(\sigma) := \frac{1}{a_\sigma} \mathcal{G}_\sigma [\beta_{\mathfrak{E}}(\sigma)].$$

The discrete sharp operator on  $\sigma$  is thus the  $3 \times 4$  matrix  $U_\sigma := \frac{1}{a_\sigma} \mathcal{G}_\sigma$ .

The flat operator projects a vector field onto the tangent plane and computes its circulation along each edge. The 1-form  $\mathbf{v}^\flat$  associated with vector  $\mathbf{v}$  is thus:

$$[\mathbf{v}_{\mathfrak{E}}^\flat] := \oint_{\partial f} \mathbf{t}^T (I - \mathbf{u}\mathbf{u}^T) \mathbf{v} = \int_0^1 \begin{bmatrix} \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} (I - \mathbf{u}(r, 0)\mathbf{u}^T(r, 0)) \mathbf{v} \\ \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} (I - \mathbf{u}(1, r)\mathbf{u}^T(1, r)) \mathbf{v} \\ \begin{bmatrix} -1 & 0 & 0 \end{bmatrix} (I - \mathbf{u}(r, 1)\mathbf{u}^T(r, 1)) \mathbf{v} \\ \begin{bmatrix} 0 & -1 & 0 \end{bmatrix} (I - \mathbf{u}(0, r)\mathbf{u}^T(0, r)) \mathbf{v} \end{bmatrix} dr.$$

By linearity, the flat operator  $V_\sigma$  is a  $4 \times 3$  matrix (see appendix for details).

*Inner products for discrete forms (i.e metrics).* The inner product between 0-forms is simply the integration of their product on the surfel  $\sigma$ . For any bilinearly interpolated functions  $\phi, \psi$ , we obtain on the surfel  $\sigma$  the scalar:

$$\langle \phi | \psi \rangle_0(\sigma) := \iint_{\sigma} \phi \psi \omega_0^{\mathbf{u}} = [\phi_{\mathfrak{E}}(\sigma)]^\top M_{0,\sigma} [\phi_{\mathfrak{E}}(\sigma)].$$

The associated metric matrix is a  $4 \times 4$  symmetric matrix, called *mass matrix*, whose expression is given in the appendix. If the corrected normal vector  $\mathbf{u}$  is consistent with the naive surfel normal  $\mathbf{n}$  (i.e.  $\langle \mathbf{u}(s, t) | \mathbf{n} \rangle > 0$ ), then  $M_{0,\sigma}$  is positive definite.

We would like the inner product between 1-forms  $\beta$  and  $\gamma$  to be defined by emulating the continuous case. We integrate the scalar product between the vectors associated with the 1-forms on the surfel  $\sigma$ :

$$\langle \beta | \gamma \rangle_1(\sigma) := \iint_{\square} \langle \beta^\sharp | \gamma^\sharp \rangle \omega_0^{\mathbf{u}} = [\beta_{\mathfrak{E}}(\sigma)]^\top M_{1,\sigma}^{\text{naive}} [\gamma_{\mathfrak{E}}(\sigma)].$$

Using above relations we have:

$$\langle \beta | \gamma \rangle_1(\sigma) = a_\sigma (U_\sigma [\beta_{\mathfrak{F}}(\sigma)])^\top (U_\sigma [\gamma_{\mathfrak{F}}(\sigma)] \gamma) = [\beta_{\mathfrak{F}}(\sigma)]^\top \left( \frac{1}{a_\sigma} \mathcal{G}_\sigma^\top \mathcal{G}_\sigma \right) [\gamma_{\mathfrak{F}}(\sigma)].$$

Hence  $M_{1,\sigma}^{\text{naive}} = \frac{1}{a_\sigma} \mathcal{G}_\sigma^\top \mathcal{G}_\sigma$ ; it is a symmetric matrix. It can be verified that, if  $\mathbf{u}$  is a unit constant vector over the surfel  $\sigma$  and  $\langle \mathbf{u} | \mathbf{n} \rangle > 0$ , then this matrix is symmetric positive. However, it is not definite. To remedy this, we follow [6] and complement the definition to get the *stiffness matrix* as

$$M_{1,\sigma} := \frac{1}{a_\sigma} \mathcal{G}_\sigma^\top \mathcal{G}_\sigma + \lambda(I - U_\sigma V_\sigma). \quad (1)$$

*Calculus on the whole mesh.* Let  $n$ ,  $m$  and  $k$  be respectively the number of vertices, edges and faces of the mesh. Let  $\mathcal{V}$  be the space of all sampled functions (an  $n$ -dimensional vector space), and  $\mathcal{E}$  be the space of all discrete 1-forms (an  $m$ -dimensional vector space). Global operators sharp  $U$  (size  $3k \times m$ ), flat  $V$  (size  $m \times 3k$ ), mass matrix  $M_0$  (size  $n \times n$ ) and stiffness matrix  $M_1$  (size  $m \times m$ ), differential  $D_0$  (size  $m \times n$ ) are obtained by merging the corresponding local operators  $U_\sigma, V_\sigma, M_{0,\sigma}, M_{1,\sigma}, D_0$  on the corresponding rows and columns.

*Codifferentials and Laplacian.* We build the 1-codifferential  $\delta_1 : \mathcal{E} \rightarrow \mathcal{V}$  by adjointness in our inner products.

$$\forall f \in \mathcal{V}, \forall \alpha \in \mathcal{E}, \langle D_0 f | \alpha \rangle_1 = -\langle f | \delta_1 \alpha \rangle_0 \Leftrightarrow (D_0 f)^\top M_1 \alpha = -f^\top M_0 \delta_1 \alpha.$$

Being true for all pairs  $(f, \alpha)$ , it follows that  $\delta_1 := -M_0^{-1} D_0^\top M_1$ . The *Laplacian operator*  $\Delta_0$  is the composition of the codifferential and the differential, i.e.

$$\Delta_0 := \delta_1 D_0 = -M_0^{-1} D_0^\top M_1 D_0.$$

Since it is very costly to build matrix  $M_0^{-1}$ , we will generally not use the two operators  $\delta_1$  and  $\Delta_0$  as is when solving numerical problems, but we will rather work with their “integrated” version ( $M_0 \delta_1$  and  $M_0 \Delta_0$ ). We can now see another approach to compute a Laplace-Beltrami operator coming from the Finite Elements framework.

## 2.2 Generalization to Finite Element Method

We show here how to adapt the standard Finite Element Method (FEM), e.g. see [16], in order to solve a Poisson problem. The method builds a stiffness matrix  $L$  and a mass matrix  $M$  to transform the Poisson problem into a linear problem. We will see also that a Laplace operator can be obtained with the same method. Our adaptation consist in correcting the metric used, changing the formulas used for derivatives and dot products. While we only demonstrate here how to correct FEM on a Poisson problem, other problems can also be corrected with the same metric.

The Poisson problem is formulated as solving for  $g$  in  $\Delta g = f$ , with a given border constraint for  $g$  if the domain has a boundary, or with a fixed value somewhere if the domain has no boundary. The weak formulation of this problem is given by: solve for  $f$

$$\int_{\Omega} \nabla g \cdot \nabla \Phi = - \int_{\Omega} f \Phi + \int_{\partial\Omega} \Phi \langle \nabla g, \mathbf{n} \rangle, \quad (2)$$

for any  $\Phi$ . In our case, we will evaluate against  $\Phi$  the functions locally bilinear inside each element. The third term is dependent on the boundary condition, and we will make it vanish here for now.

The FEM approach consists in discretizing the problem at nodes and splitting the domain into elements bordered by nodes (quads here): functions  $g$  (say) are discretized at these nodes as vectors  $\mathbf{g}$  of their values at nodes. FEM assumes bilinear interpolation of functions within elements. It builds a stiffness matrix  $L$  and a mass matrix  $M$  such that  $L\mathbf{g} = M\mathbf{b}$ . This corresponds to the first two terms in (2). Boundary constraints are integrated in this linear problem, either by removing rows and columns or by setting equalities. We can then solve the Poisson problem by solving the linear system  $L\mathbf{g} = M\mathbf{b}$ , but we can also deduce a Laplacian operator  $\Delta := M^{-1}L$ .

The matrices are built quad by quad, so here per surfel. We start by defining a metric  $G$  per surfel, since it depends on the corrected normal  $\mathbf{u}$ , then using this metric in the formulas for derivatives and scalar products when building the matrices. Our reference element is a unit square in the plane  $\square$ . We obtain:

$$G = \begin{bmatrix} 1 - (\mathbf{u}^x)^2 & -\mathbf{u}^x \mathbf{u}^y \\ -\mathbf{u}^x \mathbf{u}^y & 1 - (\mathbf{u}^y)^2 \end{bmatrix}.$$

Since we assume now that our corrected normal field is constant on the surfel, the metric is also constant. This is an arbitrary choice we make in order to keep formulas simple. It becomes easy to compute the gradient and Laplacian. We use the formula  $df(\mathbf{w}) = \langle \nabla f, \mathbf{w} \rangle_G$  for any vector  $\mathbf{w}$ , with  $\langle \cdot, \cdot \rangle_G$  the inner product. It follows that  $\nabla f = G^{-1} \left[ \frac{\partial f}{\partial s} \quad \frac{\partial f}{\partial t} \right]^T$ . For the Laplacian, since the metric is constant, we use  $\Delta f = \nabla \cdot \nabla f$ . We write them more explicitly as:

$$\nabla f = \frac{1}{(\mathbf{u}^z)^2} \begin{bmatrix} (1 - (\mathbf{u}^y)^2) \frac{\partial f}{\partial s} + \mathbf{u}^x \mathbf{u}^y \frac{\partial f}{\partial t} \\ \mathbf{u}^x \mathbf{u}^y \frac{\partial f}{\partial s} + (1 - (\mathbf{u}^x)^2) \frac{\partial f}{\partial t} \end{bmatrix} \quad (3)$$

$$\Delta f = \frac{1}{(\mathbf{u}^z)^2} \left( (1 - (\mathbf{u}^y)^2) \frac{\partial^2 f}{\partial s^2} + 2\mathbf{u}^x \mathbf{u}^y \frac{\partial^2 f}{\partial s \partial t} + (1 - (\mathbf{u}^x)^2) \frac{\partial^2 f}{\partial t^2} \right) \quad (4)$$

We choose a basis of bilinear functions on the square as our basis functions. This choice can be disputed: while linear functions are still harmonic regarding to the Laplacian in (4), bilinear functions are no longer harmonics in this setting. However, finding a way to build hat functions that stay harmonic in this setting is not obvious. Yet bilinear functions are still used on quad meshes that are not rectangular and where the same reasoning can be applied to show that they are not harmonic. We define the four basis functions as  $f_0 = (1-s)(1-t)$ ,  $f_1 = s(1-t)$ ,  $f_2 = st$ ,  $f_3 = (1-s)t$ .

In order to build our stiffness matrix we evaluate:  $\int_{\square} \langle \nabla f, \nabla p \rangle_G$  for any  $f$  and  $p$  bilinear. The local stiffness matrix is then:

$$L_M = \frac{1}{6\mathbf{u}^z} \begin{bmatrix} 3\mathbf{u}^x \mathbf{u}^y + 2 + 2(\mathbf{u}^z)^2 & 2(\mathbf{u}^y)^2 - 1 - (\mathbf{u}^x)^2 & 1 - 3\mathbf{u}^x \mathbf{u}^y - (\mathbf{u}^z)^2 & 2(\mathbf{u}^x)^2 - 1 - (\mathbf{u}^y)^2 \\ 2(\mathbf{u}^y)^2 - 1 - (\mathbf{u}^x)^2 & 2 - 3\mathbf{u}^x \mathbf{u}^y + 2(\mathbf{u}^z)^2 & 2(\mathbf{u}^x)^2 - 1 - (\mathbf{u}^y)^2 & 3\mathbf{u}^x \mathbf{u}^y + 1 - (\mathbf{u}^z)^2 \\ 2 - 3\mathbf{u}^x \mathbf{u}^y - (\mathbf{u}^z)^2 & 2(\mathbf{u}^x)^2 - 1 - (\mathbf{u}^y)^2 & 3\mathbf{u}^x \mathbf{u}^y + 2 + 2(\mathbf{u}^z)^2 & 2(\mathbf{u}^y)^2 - 1 - (\mathbf{u}^x)^2 \\ 2(\mathbf{u}^x)^2 - 1 - (\mathbf{u}^y)^2 & 3\mathbf{u}^x \mathbf{u}^y + 1 + (\mathbf{u}^z)^2 & 2(\mathbf{u}^y)^2 - 1 - (\mathbf{u}^x)^2 & 2 - 3\mathbf{u}^x \mathbf{u}^y + 2(\mathbf{u}^z)^2 \end{bmatrix}. \quad (5)$$

The global stiffness matrix is then obtained by summing over all the local stiffness matrices. The mass matrix is computed from  $\int_{\Omega} f p$ , with  $f$  and  $p$  bilinear:

$$M_M = \frac{\mathbf{u}^z}{36} \begin{bmatrix} 4 & 2 & 1 & 2 \\ 2 & 4 & 2 & 1 \\ 1 & 2 & 4 & 2 \\ 2 & 1 & 2 & 4 \end{bmatrix}. \quad (6)$$

We recognize the standard mass matrix for quad mesh with a factor correcting the area of the surfel. It is the same as  $M_0$  for constant  $\mathbf{u}$ . We now have two ways of building a sparse Laplace-Beltrami operator. We see now how they compare against operators from previous works.

### 3 Evaluations and comparisons

We compare the resulting operators and the ones from previous works on several use cases : first on the sphere, with forward evaluation (compute the laplacian of a function), backward evaluation (solve a Poisson problem to get a function back from its laplacian), eigenvalue comparisons, and then on a standard mesh by comparing with the results obtained on an underlying triangle mesh. Plots related to digitized spheres are the means of the results of 32 computations for each step, each conducted with a different center to better take into account the variability in sphere discretizations.

#### 3.1 Forward evaluation

Several previous works tried to evaluate the quality and convergence of the Laplacian operator when used in a forward manner: from  $f$  defined on the mesh, we compute  $\Delta f$  both analytically and with a discrete Laplacian, then compare the two results. In other words, if our stiffness matrix is called  $L$  and our mass matrix  $M$ , we solve the equation  $LF = MX$  where  $X$  is the unknown.

A naive approach consists in computing  $X = M^{-1}LF$ . This is the one that was used for evaluation in previous works, and was not convergent when using sparse operators. We reproduce this behavior by computing the Laplacian of  $f(x) = e^x$  using various methods, none of which seem to converge (see figure 2). This is disappointing since we expect the Laplacian operator to have linear convergence when evaluated in forward manner, as observed in [2] and proven for the mesh Laplacian on triangle mesh.

Our idea for improving the convergence consists of adding a small diffusion step to the result. It suffices to replace the mass matrix  $M$  by  $M - dtL$ . In other



words, instead of evaluating  $X = M^{-1}LF$ , we evaluate  $X = (M - dtL)^{-1}LF$ . The result depends on the choice of parameter  $dt$ : we found that we approach linear convergence when  $dt$  is in the order of  $h$ , and the best quality for  $dt = 0.035h$ . This means that we add a diffusion with a characteristic length of order  $h^{\frac{1}{2}}$ , which is coherent with results from other works. Using this method, we achieve what seems to be linear convergence on different functions (Figure 2), with results comparable or even higher quality than in [2]. We run the same experiment as figure 2, with evaluated normals (using Integral Invariant [10]) instead of ground truth. Results are shown in figure 3, and also approach linear convergence.

### 3.2 Backward evaluation

A Laplacian is often built to solve a Poisson problem. We evaluate a function on our digital surface, we also evaluate its Laplacian using an exact formula, then we compute an approximation of the original function that we compare to the exact original. It is a criterion used for Laplacian evaluation (see [1]), which has not yet been done for digital Laplacians. It also makes more sense to evaluate the Finite Element Methods in this case than in forward evaluation, as this is the problem the operator is built for and is proven (in the case of standard

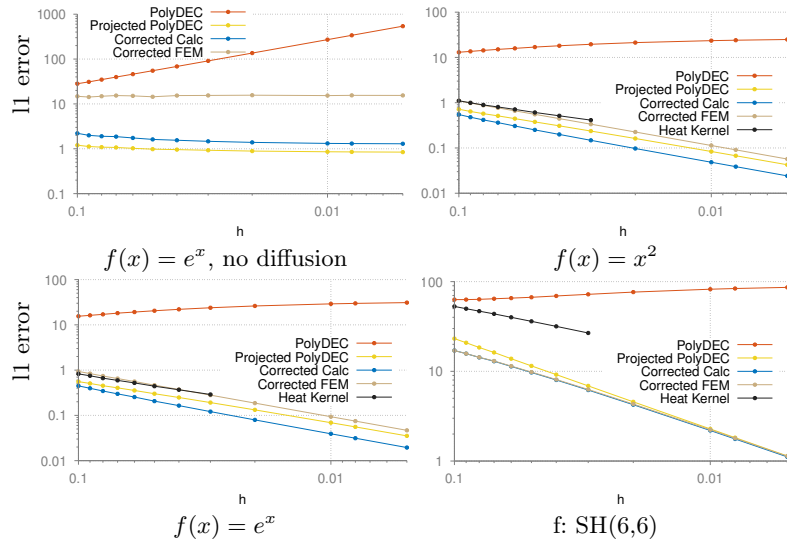


Fig. 2: Forward evaluation of the mesh Laplacian on quadratic, exponential functions and the sixth spherical harmonic. Adding diffusion significantly improves the results, achieving linear convergence on the sphere. We achieve similar rates of convergence as Heat Kernel [2], with a better quality on less smooth functions. Note that the Heat Kernel method is limited to a gridstep of  $h \geq 0.03$ , due to its enormous memory usage.

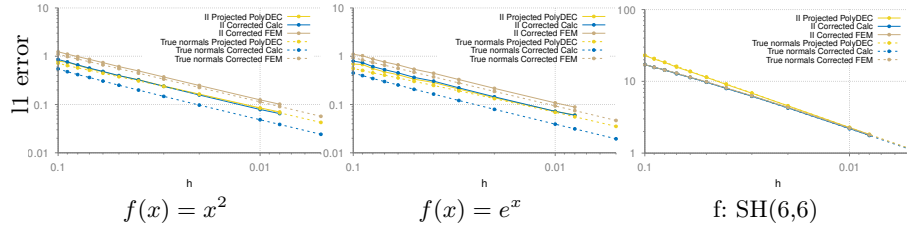


Fig. 3: Comparison on evaluation of the Laplacian using the true normals and using the Integral Invariant estimators. Estimated normals give slightly worse results than with true normals, but still seem to converge. The estimated normals values are limited to a gridstep of  $h \geq 0.008$  due to their time to evaluate.

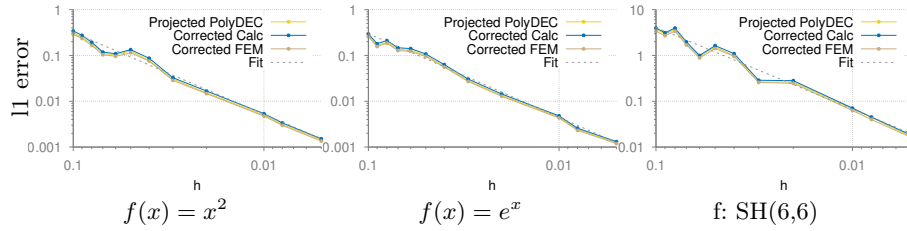


Fig. 4: Error when solving a Poisson problem with different Laplacians. We approach a quadratic convergence rate.

regular meshes) to converge. We find that all methods give roughly the same results (figure 4). They seem to be convergent, with a rate around  $h^{1.9}$ , which is coherent with the theoretical quadratic rate. Again, we run the experiment using the Integral Invariant estimators and approach similar rate of convergence. (figure 5)

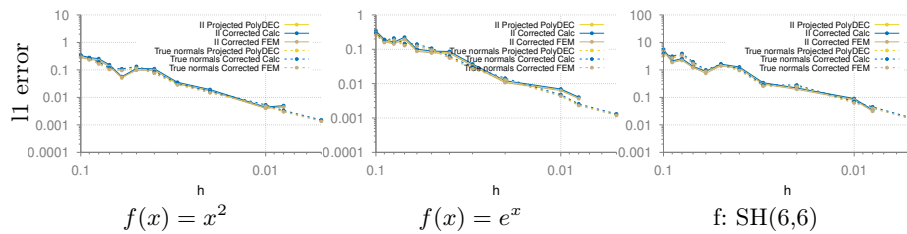


Fig. 5: Comparison of results using true normals and estimated normals for solving a Poisson problem.

### 3.3 Eigenvalues

We follow the evaluation of eigenvalues on the spherical harmonics used in [1]. Since the spherical harmonics have analytic expressions, we can compare the eigenvalues of our operator to the exact eigenvalues of the Laplace-Beltrami operator on the sphere. To obtain these eigenvalues, we solve for  $\lambda$  in the following generalized eigenvalue problem  $L\mathbf{u} = \lambda M\mathbf{u}$ . Figure 6 shows the first eigenvalues of our Laplacians on the unit sphere with discretization steps  $h = 0.1$  and  $h = 0.01$ . The PolyDEC method [6] is not accurate, but corrected methods are, with accuracy increased at finer resolutions.

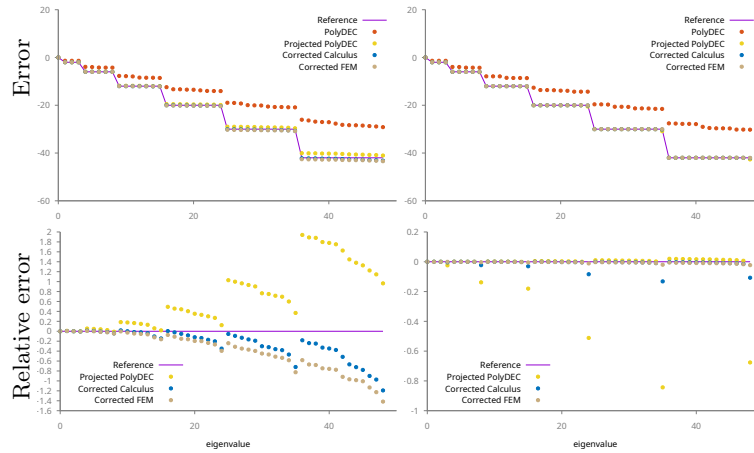


Fig. 6: Smallest 49 eigenvalues of the Laplacian on unit sphere with discretization step 0.1 (left) and 0.01 (right). Relative error is given by  $\hat{\lambda} - \lambda$ , where  $\hat{\lambda}$  is the approximated eigenvalue and  $\lambda$  the correct value.

### 3.4 Comparison to the cotan Laplacian

Until now all our comparisons were made on a digitized sphere: this is because there are some closed form expressions of Laplacians, and its eigen decomposition is well studied. However, the sphere is a very specific case, and our evaluations may not reflect well more general cases. We compare here our operators to the results obtained on a regular, high quality triangle mesh with the standard cotan Laplacian. To do so, we use a refined version of a triangle mesh (at 100000 vertices), and equivalent digital surfaces at different resolutions ( $128^3$ ,  $256^3$ ,  $512^3$ ). Then we built a projection operator allowing us to map values on the high resolution mesh to the digital one (orthogonal projection and linear interpolation). We also use this projection operator to map the normal vector field computed on the mesh to the surfels or vertices in the calculus frameworks. We then compute

a function on the triangle mesh as well as its Laplacian using the cotan Laplacian on the triangle mesh [14], and then their projection on the digital surface, which we use as "ground truth". Forward evaluation results are shown on figure 7. We use the same diffusion constant as previously ( $0.035h$ ). Error is significantly reduced with the discretization step of the digital surface, suggesting that our operator is convergent toward the result given by the cotan Laplacian.






















	resolution: 128	$l1$ error	resolution: 256	$l1$ error	resolution: 512	$l1$ error
<i>Ground truth</i>						
<i>FEM</i>						
		0.0484979		0.0299988		0.0194369
<i>Corrected Calculus</i>						
		0.0473037		0.0311042		0.0197104
<i>Poly Proj</i>						
		0.0474556		0.0301771		0.0195026

Fig. 7: Comparison between classical cotan Laplacian of a function defined on a triangle mesh and the digital Laplacian operators. All three operators have comparable performances. The error decreases as the resolution increases, suggesting that all three operators are convergent.

## 4 Conclusion

We show that, similarly to the corrected PolyDEC method [3], a corrected normal field can be inserted within discrete calculus frameworks yielding different Laplace-Beltrami operators. All these operators seem to converge when solving Poisson problems, and when used in a forward evaluation, the addition of a slight diffusion also seems to make them convergent. A limit of our study is that these results are only experimental: only the Heat kernel Laplacian of [2] is yet proven to converge on digital surfaces (strong consistency). However, since results on a common framework (the finite element method) seem promising, it may be interesting to see if the proof of convergence from this framework can be adapted

to digital surfaces. The same type of calculus construction could also be tested outside digital surfaces, for instance on a triangle mesh with a corrected normal field or a normal field of much higher resolution such as a normal map (as done in [12]). The idea of adding diffusion and modifying the mass matrix can be seen as similar to the approach of [8]. However Caissard *et al.* [2] were not able to reproduce their experiments and expected convergence, probably because digital surfaces do not have the mesh regularity required by the proof. Indeed, from our metric  $G$  it is easy to find that mesh regularity means that  $\frac{1}{|u^2|}$  is bounded. Such a condition can be fulfilled for some specific meshes (such as a digital plane), but is not guaranteed on surface digitizations in general (such as a sphere).

*Acknowledgments* This work is supported by the French National Research Agency in the framework of the « France 2030 » program (ANR-15-IDEX-0002), by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) and by the StableProxies project (ANR-22-CE46-0006).

## References

1. Bunge, A., Botsch, M.: A survey on discrete Laplacians for general polygonal meshes. *Computer Graphics Forum* **42**(2), 521–544 (2023)
2. Caissard, T., Coeurjolly, D., Lachaud, J.O., Roussillon, T.: Laplace–Beltrami operator on digital surfaces. *Journal of Mathematical Imaging and Vision* **61**(3), 359–379 (2019)
3. Coeurjolly, D., Lachaud, J.O.: A simple discrete calculus for digital surfaces. In: Étienne Baudrier, Naegel, B., Krähenbühl, A., Tajine, M. (eds.) *IAPR Second International Conference on Discrete Geometry and Mathematical Morphology*. Springer, LNCS (October 2022)
4. Coeurjolly, D., Lachaud, J.O., Gueth, P.: Digital surface regularization with guarantees. *IEEE Trans. Vis. Comput. Graph.* **27**(6), 2896–2907 (2021)
5. Crane, K., Weischedel, C., Wardetzky, M.: The heat method for distance computation. *Commun. ACM* **60**(11), 90–99 (Oct 2017)
6. De Goes, F., Butts, A., Desbrun, M.: Discrete differential operators on polygonal meshes. *ACM Transactions on Graphics (TOG)* **39**(4), 110–1 (2020)
7. de Goes, F., Desbrun, M., Meyer, M., DeRose, T.: Subdivision exterior calculus for geometry processing. *ACM Trans. Graph.* **35**(4) (jul 2016)
8. Hildebrandt, K., Polthier, K.: On approximation of the Laplace-Beltrami operator and the Willmore energy of surfaces. *Computer Graphics Forum* (2011)
9. Hirani, A.N.: *Discrete Exterior Calculus*. Ph.D. thesis, USA (2003), aAI3086864
10. Lachaud, J.O., Coeurjolly, D., Levallois, J.: Robust and convergent curvature and normal estimators with digital integral invariants. In: Laurent Najman, P.R. (ed.) *Modern Approaches to Discrete Curvature*, *Lecture Notes in Mathematics*, vol. 2184. Springer-Verlag (2017)
11. Lachaud, J.O., Romon, P., Thibert, B.: Corrected curvature measures. *Discrete & Computational Geometry* **68**(2), 477–524 (2022)
12. Lachaud, J.O., Romon, P., Thibert, B., Coeurjolly, D.: Interpolated corrected curvature measures for polygonal surfaces. *Comput. Graph. Forum* **39**(5), 41–54 (2020)

13. Lachaud, J.O., Thibert, B.: Properties of Gauss digitized shapes and digital surface integration. *Journal of Mathematical Imaging and Vision* **54**(2), 162–180 (2016)
14. Lévy, B., Zhang, H.: Spectral mesh processing. In: *ACM SIGGRAPH 2010 Courses*, pp. 1–312 (2010)
15. Nealen, A., Igarashi, T., Sorkine, O., Alexa, M.: Laplacian mesh optimization. In: *Proceedings of the 4th International Conference on Computer Graphics and Interactive Techniques in Australasia and Southeast Asia*. p. 381–389. GRAPHITE '06, Association for Computing Machinery, New York, NY, USA (2006)
16. Reddy, J.N.: *Introduction to the finite element method*. McGraw-Hill Education (2019)
17. Sorkine, O.: Laplacian mesh processing. In: Chrysanthou, Y., Magnor, M. (eds.) *Eurographics 2005 - State of the Art Reports*. The Eurographics Association (2005)
18. Veiga, L., Brezzi, F., Cangiani, A., Manzini, G., Marini, L., Russo, A.: Basic principles of virtual element methods. *Mathematical Models and Methods in Applied Sciences* **23** (11 2012)

## A Details on the interpolated corrected calculus

Let  $\sigma$  be a surfel aligned with  $x$  and  $y$  and with normal aligned with  $z$ . The flat operator has the following expression:

$$V_\sigma := \frac{1}{6} \begin{bmatrix} 6 - 2((\mathbf{u}_{00}^x)^2 + \mathbf{u}_{00}^x \mathbf{u}_{10}^x + (\mathbf{u}_{10}^x)^2) & -(2\mathbf{u}_{00}^x + \mathbf{u}_{10}^x) \mathbf{u}_{00}^y - (\mathbf{u}_{00}^x + 2\mathbf{u}_{10}^x) \mathbf{u}_{10}^y & -(2\mathbf{u}_{00}^x + \mathbf{u}_{10}^x) \mathbf{u}_{00}^z - (\mathbf{u}_{00}^x + 2\mathbf{u}_{10}^x) \mathbf{u}_{10}^z \\ -(2\mathbf{u}_{10}^x + \mathbf{u}_{11}^x) \mathbf{u}_{10}^y - (\mathbf{u}_{10}^x + 2\mathbf{u}_{11}^x) \mathbf{u}_{11}^y & 6 - 2((\mathbf{u}_{10}^y)^2 + \mathbf{u}_{10}^y \mathbf{u}_{11}^y + (\mathbf{u}_{11}^y)^2) & (2\mathbf{u}_{10}^y + \mathbf{u}_{11}^y) \mathbf{u}_{10}^z - (\mathbf{u}_{10}^y + 2\mathbf{u}_{11}^y) \mathbf{u}_{11}^z \\ 2((\mathbf{u}_{01}^x)^2 + \mathbf{u}_{01}^x \mathbf{u}_{11}^x + (\mathbf{u}_{11}^x)^2) - 6 & (2\mathbf{u}_{01}^x + \mathbf{u}_{11}^x) \mathbf{u}_{01}^y + (\mathbf{u}_{01}^x + 2\mathbf{u}_{11}^x) \mathbf{u}_{11}^y & (2\mathbf{u}_{01}^x + \mathbf{u}_{11}^x) \mathbf{u}_{01}^z + (\mathbf{u}_{01}^x + 2\mathbf{u}_{11}^x) \mathbf{u}_{11}^z \\ (2\mathbf{u}_{00}^x + \mathbf{u}_{01}^x) \mathbf{u}_{00}^y + (\mathbf{u}_{00}^x + 2\mathbf{u}_{01}^x) \mathbf{u}_{01}^y & 2((\mathbf{u}_{00}^y)^2 + \mathbf{u}_{00}^y \mathbf{u}_{01}^y + (\mathbf{u}_{01}^y)^2) - 6 & (2\mathbf{u}_{00}^y + \mathbf{u}_{01}^y) \mathbf{u}_{00}^z + (\mathbf{u}_{00}^y + 2\mathbf{u}_{01}^y) \mathbf{u}_{01}^z \end{bmatrix}$$

The metric matrix for 0-forms is defined as the matrix such that, for any bilinearly interpolated functions  $\phi, \psi$ , we obtain on surfel  $\sigma$  the scalar:

$$\langle \phi | \psi \rangle_0(\sigma) := \iint_\sigma \phi \psi \omega_0^{\mathbf{u}} = [\phi_{\square}(\sigma)]^\top M_0 [\phi_{\square}(\sigma)].$$

Let us now define weighted sums for components of  $\mathbf{u}$  over the quad. We number the edges when turning along the boundary of the surfel  $\sigma$  from 0 to 3, such that edges 0,1,2,3 connect vertex pairs  $(\mathbf{x}_{00}, \mathbf{x}_{10})$ ,  $(\mathbf{x}_{10}, \mathbf{x}_{11})$ ,  $(\mathbf{x}_{01}, \mathbf{x}_{11})$ ,  $(\mathbf{x}_{01}, \mathbf{x}_{00})$ , respectively. We define

$$\begin{aligned} \bar{\mathbf{u}}_{00} &:= 9\mathbf{u}_{00} + 3\mathbf{u}_{10} + \mathbf{u}_{11} + 3\mathbf{u}_{01} & \bar{\mathbf{u}}_{10} &:= 3\mathbf{u}_{00} + 9\mathbf{u}_{10} + 3\mathbf{u}_{11} + \mathbf{u}_{01} \\ \bar{\mathbf{u}}_{11} &:= \mathbf{u}_{00} + 3\mathbf{u}_{10} + 9\mathbf{u}_{11} + 3\mathbf{u}_{01} & \bar{\mathbf{u}}_{01} &:= 3\mathbf{u}_{00} + \mathbf{u}_{10} + 3\mathbf{u}_{11} + 9\mathbf{u}_{01} \\ \bar{\mathbf{u}}_{00,10} &:= 3\mathbf{u}_{00} + 3\mathbf{u}_{10} + \mathbf{u}_{11} + \mathbf{u}_{01} & \bar{\mathbf{u}}_{10,11} &:= \mathbf{u}_{00} + 3\mathbf{u}_{10} + 3\mathbf{u}_{11} + \mathbf{u}_{01} \\ \bar{\mathbf{u}}_{11,01} &:= \mathbf{u}_{00} + \mathbf{u}_{10} + 3\mathbf{u}_{11} + 3\mathbf{u}_{01} & \bar{\mathbf{u}}_{01,00} &:= 3\mathbf{u}_{00} + \mathbf{u}_{10} + \mathbf{u}_{11} + 3\mathbf{u}_{01} \end{aligned}$$

By integration of left-hand side, we obtain for a surfel with normal  $z$ :

$$\mathbf{M}_0 = \frac{1}{144} \begin{bmatrix} \bar{\mathbf{u}}_{00}^z & \bar{\mathbf{u}}_{00,10}^z & 4\bar{\mathbf{u}}^z & \bar{\mathbf{u}}_{01,00}^z \\ \bar{\mathbf{u}}_{00,10}^z & \bar{\mathbf{u}}_{10}^z & \bar{\mathbf{u}}_{10,11}^z & 4\bar{\mathbf{u}}^z \\ 4\bar{\mathbf{u}}^z & \bar{\mathbf{u}}_{10,11}^z & \bar{\mathbf{u}}_{11}^z & \bar{\mathbf{u}}_{11,01}^z \\ \bar{\mathbf{u}}_{11,01}^z & 4\bar{\mathbf{u}}^z & \bar{\mathbf{u}}_{11,01}^z & \bar{\mathbf{u}}_{01}^z \end{bmatrix}.$$