Fast and Robust QEF Minimization using Probabilistic Quadrics

P. Trettner\textsuperscript{1} and L. Kobbelt\textsuperscript{2}

\textsuperscript{1}RWTH Aachen University, Germany

\begin{abstract}
Error quadrics are a fundamental and powerful building block in many geometry processing algorithms. However, finding the minimizer of a given quadric is in many cases not robust and requires a singular value decomposition or some ad-hoc regularization. While classical error quadrics measure the squared deviation from a set of ground truth planes or polygons, we treat the input data as genuinely uncertain information and embed error quadrics in a probabilistic setting (“probabilistic quadrics”) where the optimal point minimizes the expected squared error. We derive closed form solutions for the popular plane and triangle quadrics subject to (spatially varying, anisotropic) Gaussian noise. Probabilistic quadrics can be minimized robustly by solving a simple linear system — 50× faster than SVD. We show that probabilistic quadrics have superior properties in tasks like decimation and isosurface extraction since they favor more uniform triangulations and are more tolerant to noise while still maintaining feature sensitivity. A broad spectrum of applications can directly benefit from our new quadrics as a drop-in replacement which we demonstrate with mesh smoothing via filtered quadrics and non-linear subdivision surfaces.
\end{abstract}

\section*{CCS Concepts}
\begin{itemize}
\item Computing methodologies \rightarrow Mesh models; Mesh geometry models;
\end{itemize}
1. Introduction

Quadratic error functions (QEFs) are a popular choice for defining distance, loss, energy, or fitness functions in various optimization settings. In the three-dimensional case, a quadric can be represented by a symmetric matrix \( Q \in \mathbb{R}^{4 \times 4} \) and the corresponding QEF is defined as \( f(x) = x^T Q x \) for \( x \) in homogeneous coordinates. \( Q \) has the following structure:

\[
Q = \begin{bmatrix}
A & -b \\
-b^T & c
\end{bmatrix}
\]  

(1)

Thus, \( f \) can also be written as \( x^T A x - 2x^T b + c \), with a symmetric positive semidefinite matrix \( A \in S^3 \), \( b \in \mathbb{R}^3 \), and \( c \in \mathbb{R} \). The point minimizing the quadratic error (the minimizer of \( Q \)) can be computed via

\[
x_{\text{min}} = A^{-1}b
\]

(2)

Often, individual quadrics are formulated for certain objects or sub-problems. The final function to be optimized consists of the sum of various quadratic functions. Due to the linearity of the matrix product, instead of optimizing a sum of quadratic functions, one can sum the individual quadric matrices (e.g. \( Q_a \) and \( Q_b \)) into a new quadric \( Q' \) and optimize \( x^T Q' x \).

\[
f(x) = x^T Q_a x + x^T Q_b x = x^T (Q_a + Q_b) x = x^T Q' x
\]

(3)

This insight lies at the heart of many quadric-based optimizations such as incremental decimation [LT98, GH97], vertex clustering [Lin00], isosurface extraction [KBSS01, JLSW02, SW04], and mesh-filtered quadrics [VNMC10, LTB19]. While often interpreted geometrically, quadrics are a versatile tool and can be used for basically any continuous attribute [Hop99].

While many geometric problems are intuitively formulated and optimized by QEFs, this process is not always robust. Many quadrics correspond to singular matrices, most notably the quadric describing the quadratic distance to a plane. To uniquely determine the minimizer of \( f(x) \), it requires at least the sum of three linearly independent plane quadrics. Even then, the result is not robust if any planes are close to being coplanar, a problem often noted in decimation and isosurface extraction. This problem is magnified if the quadrics are affected by aliasing or noise, e.g. if they originate from sampled measurements or other forms of discretization.

The conceptual idea behind classical error quadrics is to consider the input geometry (e.g. input planes and triangles) as ground truth and measure the deviation of a point from it by the sum of the squared distances. The optimal point is then the least squares minimizer of the quadratic error function. For noisy (or uncertain) input data, the least squares solution is expected to implicitly “average out” the noise.

The biggest issue with classical error quadrics is that (near) degenerate (i.e. rank deficient) configurations are quite common such that pseudo-inverses have to be computed via SVD in order to guarantee a robust solution of the least squares problem.

In our approach we follow a different rationale. We assume that the input quadrics are samples drawn from a certain, potentially complex distribution. In particular, we will examine the case of planes built from point and normal, both subject to Gaussian noise, and triangles built from three points, each subject to Gaussian noise. The optimal point is then the minimizer of the expected quadratic error or, equivalently, the minimizer of the expected error quadric. We call this expected error quadric the “probabilistic error quadric”.

1.1. Contribution

In summary, our method is a principled and robust way to incorporate arbitrarily distributed noise directly into the creation of error quadrics and thus let any quadric-based technique benefit from this probabilistic approach.

We contribute:

- the insight that the expected value over a distribution of quadrics can be recast into an ordinary quadratic form using a special “probabilistic” quadric.
- closed-form solutions for the popular plane and triangle quadrics under (potentially anisotropic) Gaussian noise.
- a theoretical foundation for the validity of common regularization techniques that can be applied to minimize quadrics robustly.

After describing our method in Section 3, Section 4 examines the properties of our quadrics in various experiments. Probabilistic quadrics are useful in a broad spectrum of applications which we demonstrate with isosurface extraction, mesh smoothing via filtered quadrics, and subdivision surfaces in Section 5.

A reference implementation of our probabilistic quadrics is provided as a C++ header under the permissive MIT license at https://graphics.rwth-aachen.de/probabilistic-quadrics.

2. Related Work

Quadratic error functions are a popular tool with a long history in geometry processing.

One of the first use of error quadrics was for mesh decimation. Garland and Heckbert [GH97] present an incremental decimation scheme using arbitrary vertex-pair collapses. They store per-vertex quadrics that are summed up on each contraction, choosing the new vertex position as the minimizer of the quadric sum. The quadrics are constructed as the sum of the quadratic distance from the planes defined by the adjacent triangles. Salinas et al. [SLA15] take additional plane proxies into account to better preserve planar structures. Li and Zhu [LZ08] further incorporate local curvature.

Vertex clustering also benefits greatly from quadrics. Lindstrom [Lin00] presents an out-of-core simplification system based on vertex clustering where the space complexity only depends on the output complexity. The representative points per cell are computed by minimizing an error quadric. They also propose the singular value decomposition as a robust way to minimize quadrics.

Error quadrics are not restricted to 3D geometry: Hoppe [Hop99] presents quadrics for appearance attributes and Garland and Zhou [GZ05] generalize quadric-based simplification to higher dimension.
A common problem in isosurface extraction is the preservation of sharp features. This can be done by incorporating plane-based error quadrics. Based on this idea, Kobbelt et al. extend the Marching Cubes algorithm. Ju et al. [JLSW04] introduce Dual Contouring which stores Hermite data on each edge of a voxel grid and the vertices of the extracted geometry are found by minimizing an error quadric associated with each voxel. Schaefer and Warren [SW04] present Dual Marching Cubes where they apply a bilateral filtering to the dual graph of an octree, informing the subdivision level by the quadratic error.

Recently, averaging and filtering quadrics across the surface of a mesh to perform tasks like smoothing and clustering gained in popularity. Vieira et al. [VNMC10] compute averaged quadrics over some surface area and move each vertex to its quadric minimizer to perform denoising. Legrand et al. [LTB19] use a bilateral filtering of quadrics for smoothing and clustering.

Quadrics are also popular in shape approximation. Yan et al. [YWLY12] directly fit quadrics to the surface to perform variational mesh segmentation. Thierry et al. [TGB13] introduce Sphere-Meshes for mesh approximation based on spherical error quadrics. They later use these meshes for animation [TGBE16]. Calderon and Boubekeur use quadrics as part of their bounding proxy generation [CB17].

3. Method

We will start by deriving probabilistic quadrics for planes and triangles under Gaussian noise. These quadrics are formed by taking a classical quadric, choosing a distribution of input parameters, and then computing the expected value of $Q$ in the form of $E[A]$, $E[b]$, and $E[c]$. Afterwards we discuss the stability of these quadrics and present the general case.

3.1. Planes under Gaussian Noise

Given a plane defined by position $q$ and normal $n$, we compute the quadratic distance to this plane via

$$d(x) = (x-q)^Tn$$

$$= x^Tnn^Tn - 2q^Tnn^Tx + q^Tnn^Tq. \quad (4)$$

This is a quadric with $A = nn^T$, $b = qn^T$, and $c = q^Tnn^Tq$.

If normal and position are estimates from measurements, they could be modeled by independent multivariate Gaussian distributions with means $\bar{n}, \bar{q} \in \mathbb{R}^3$ and variances $\Sigma_n, \Sigma_q \in \mathbb{R}^{3x3}$:

$$n \sim N(\bar{n}, \Sigma_n), \quad q \sim N(\bar{q}, \Sigma_q) \quad (5)$$

Recalling that the second moment of a Gaussian is given by $E[e_{xx}xx^T] = xx^T + \Sigma_x$, we can derive the probabilistic quadric $Q_{p(n,q)}$:

$$E[A] = E_{n,q}[nn^T] = \bar{n}\bar{n}^T + \Sigma_n \quad (6)$$

Because $n$ and $q$ are independent, $E[b]$ can be computed as

$$E[b] = E_{n,q}[qn^T] = E_n[qn^T]E_q[q] = (\bar{n}\bar{n}^T + \Sigma_n)\bar{q} \quad (7)$$

For quadratic forms it can be shown that $E_{x}[x^TMx] = \bar{x}^TM\bar{x} + \text{Tr}[\Sigma_x]$ where $\text{Tr}[\cdot]$ is the trace of a matrix. Finally, using $\text{Tr}[xx^TM] = x^TMx$ we can compute $E[c]$ as:

$$E[c] = E_{n,q}[q^Tnn^Tq] = E_q[q^T\Sigma_n[q]$$

$$= \bar{q}^T(\bar{n}\bar{n}^T + \Sigma_n)\bar{q} + \text{Tr}[(\bar{n}\bar{n}^T + \Sigma_n)\Sigma_q]$$

$$= \bar{q}^T(\bar{n}\bar{n}^T + \Sigma_n)\bar{q} + \text{Tr}[\Sigma_n\Sigma_q]$$

Compared to the original quadric, we add $\Sigma_n$ to $A$ and $\Sigma_nq$ to $b$. Two insightful interpretations of this probabilistic quadric are:

1. The probabilistic plane quadric is the sum of the classical plane quadric and the quadric for $||x-q||^2$, weighted by $\Sigma_q$.

2. The probabilistic plane quadric is the classical plane quadric, solved with generalized Tikhonov regularization in the form of $||Ax-b||^2 + ||x-q||^2$.

3.2. Triangles under Gaussian Noise

Lindstrom et al. [LT98, Lin00] proposed the following error quadric for triangles $t = (p,q,r)$ with $p, q, r \in \mathbb{R}^3$:

$$Q = \begin{bmatrix} A & -b & c \\ -b^T & d & e \\ c^T & e & f \end{bmatrix} = mn^T \quad (9)$$

$$n = \begin{bmatrix} p \times q + q \times r + r \times p \\ -|p,q,r| \end{bmatrix}$$

In this formulation, $n$ is a 4-vector and $|p,q,r|$ the scalar triple product. These quadrics are automatically area-weighted. Small or highly anisotropic triangles are often problematic. Even with area weights they lead to problems, especially when the normals deviate strongly or in the presence of fold-overs (cf. Figure 2).
For the probabilistic quadric we will again model the uncertainty as a multivariate Gaussian, this time as three independently distributed positions:  

\[ p \sim \mathcal{N}(\bar{p}, \Sigma_p), \quad q \sim \mathcal{N}(\bar{q}, \Sigma_q), \quad r \sim \mathcal{N}(\bar{r}, \Sigma_r) \]  

(10)

The full derivation is in Appendix A (Theorem A.8, A.9, and A.10). There we show that:

\[
\mathbb{E}[A] = \mathbb{E}(p \times q + q \times r + r \times p) \cdot (p \times q + q \times r + r \times p)^T
\]

\[
= (\bar{p} \times \bar{q} + \bar{q} \times \bar{r} + \bar{r} \times \bar{p}) (\bar{p} \times \bar{q} + \bar{q} \times \bar{r} + \bar{r} \times \bar{p})^T
\]

\[
\mathbb{E}[B] = 2\sigma^4 \cdot I
\]

(11)

(12)

For \( b \), we can derive the following:

\[
\mathbb{E}[b] = \mathbb{E}(p \times q + q \times r + r \times p) \cdot (p \times q + q \times r + r \times p)
\]

\[
= (\bar{p} \times \bar{q} + \bar{q} \times \bar{r} + \bar{r} \times \bar{p}) (\bar{p} \times \bar{q} + \bar{q} \times \bar{r} + \bar{r} \times \bar{p})
\]

\[
\mathbb{E}[c] = \mathbb{E}[(p \times q)^2]
\]

\[
= (\bar{p} \times \bar{q})^T \Sigma_p \exp(\bar{p} \times \bar{q})
\]

\[
+ (\bar{q} \times \bar{r})^T \Sigma_p \exp(\bar{q} \times \bar{r})
\]

\[
+ (\bar{r} \times \bar{p})^T \Sigma_p \exp(\bar{r} \times \bar{p})
\]

\[
+ p^T \Sigma_p \exp(p) \Sigma_p p
\]

\[
+ q^T \Sigma_q \exp(q) \Sigma_q q
\]

\[
+ r^T \Sigma_r \exp(r) \Sigma_r r
\]

\[
+ \text{Tr} \Sigma_p \Sigma_q \Sigma_r
\]

(13)

(14)

(15)

This quadric is more sophisticated than the plane quadric and harder to interpret. However, there is at least one interesting pattern. \( \mathbb{E}[A] \) and \( \mathbb{E}[b] \) both consist of \( O(1) \), \( O(\Sigma) \), and \( O(\Sigma^2) \) terms, and \( \mathbb{E}[c] \) additionally of an \( O(\Sigma^4) \) term:

- The \( O(1) \) term corresponds to the classical quadric formed by the means of the positions.
- The \( O(\Sigma) \) terms are linear in \( \Sigma_p, \Sigma_q, \Sigma_r \) and capture the interaction between the covariance of one vertex with the means of the other two vertices, i.e., the opposing edge.
- The \( O(\Sigma^2) \) terms contain the interaction between two covariance matrices (in the shape of the \( \Sigma \) matrices).
- Finally, \( \mathbb{E}[c] \) contains the \( O(\Sigma^4) \) term quantifying the combined uncertainty of all covariances.

As expected, all terms have an obvious symmetric structure. The non-obvious part is \( \text{Tr}[\Sigma_p \Sigma_q \Sigma_r] \) which can be shown to be commutative in all \( \Sigma \).

3.3. Stability

A common problem with classical quadrics is that they are not invertible in planar regions. \( A = nn^T \) has always rank one and thus a two-dimensional kernel. Trying to compute \( x_{\text{min}} = A^{-1} b \) leads to undefined results which is why using an SVD is recommended. This problem is not unique to planar regions but occurs in any rank-deficient quadric.

Our probabilistic quadrics do not suffer from this problem. For any positive definite covariance, the probabilistic plane quadric \( A = nn^T + \Sigma \) is full rank and invertible (see Theorem A.11). Similarly, the probabilistic triangle quadrics are also positive definite and thus invertible given non-degenerate covariance matrices (see Theorems A.12 and A.13).

3.4. General Probabilistic Quadric

The derivations in Section 3.1 and 3.2 can easily be generalized to arbitrary configurations. Let \( Q(s) \in \mathbb{R}^{4 \times 4} \) be the individual quadric corresponding to object \( s \in S \) (for example, \( s \) could be a plane and \( S \) the set of all planes). Given a probability distribution \( p(s) \) over objects, we define the expected value formulation of QEF minimization:

\[
f(x) = \mathbb{E}_{p(s)}[Q(s)x] = \mathcal{E}_{p(s)}[Q(s)x]
\]

\[
= \int_{s \in S} x^T Q(s)x \cdot p(s) ds
\]

\[
= x^T \mathbb{E}[Q(s)]x
\]

\[
= x^T \mathbb{E}[Q(s)]x
\]

(16)

This provides the foundation of our method: The expected quadratic error is equal to the quadratic error of the expected quadric. We call \( Q_p(s) \) the probabilistic quadric corresponding to the distribution \( p(s) \). Since \( Q_p(s) \) is just another \( 4 \times 4 \) matrix (independent of \( x \)), we can apply our technique to any geometry processing task involving quadrics, handling data uncertainty in a more principled and robust manner.
For many distributions, \( Q_{p}(s) \) can be computed analytically. The plane and triangle quadric only contain multi-variate second order polynomials when fully expanded. Thus, in theory, it is simple to plug in any distributions as long as second moments and covariances are known. The practical challenge arises from re-assembling the terms into an efficient form.

Even for involved probability distributions without closed-form solutions for their expected values, the fact that \( Q_{p}(s) \) can be pre-computed means that numerical integration methods can be applied. In a sense, methods that average or sum classical quadrics over large regions are already computing a numerical approximation of the expected quadric.

One might wonder which distribution to use for a given problem. A good rule of thumb is that the distribution should roughly model how the data was obtained or what form of uncertainty can be expected. For example, points from a laser scan have a somewhat quantifiable positional uncertainty making the triangle quadric a good choice while it is not clear how this uncertainty translates to a \( \Sigma \) for a plane quadric. However, it is not critical to exactly match the input distribution as can be seen in Figure 2 where the Gaussian quadrics perform well, even for spiky non-Gaussian noise.

### 4. Experiments

#### 4.1. Performance

The benchmarks were performed on a 3.60 GHz Intel Core i9-9900K (4.8 GHz single core turbo) using a single CPU thread. All algorithms were implemented in C++, compiled with Clang 7 using the flags -O3, -march=native, and -ffast-math.

Table 1 summarizes the central performance metrics. Constructing classical quadrics is pretty close to being limited by memory bandwidth on a modern CPU (processing more than 10 GB/s). The same can be observed for the probabilistic plane quadrics. Our probabilistic triangle quadric requires considerably more computation due to the various occurrences of the covariances. It is about 10 \( \times \) slower than the classical version, 5 \( \times \) when assuming isotropic noise. However, with about 20 million matrix operations per second on a single core, it is rarely the bottleneck.

<table>
<thead>
<tr>
<th>operation</th>
<th>throughput</th>
<th>cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>constructing plane quadrics</td>
<td></td>
<td></td>
</tr>
<tr>
<td>classical ((A, b))</td>
<td>399000000 / s</td>
<td>12 / op</td>
</tr>
<tr>
<td>classical ((A, b, c))</td>
<td>368000000 / s</td>
<td>13 / op</td>
</tr>
<tr>
<td>probabilistic ((A, b))</td>
<td>273000000 / s</td>
<td>17 / op</td>
</tr>
<tr>
<td>probabilistic ((A, b, c))</td>
<td>137000000 / s</td>
<td>35 / op</td>
</tr>
<tr>
<td>probabilistic ((A, b, isotropic))</td>
<td>399000000 / s</td>
<td>12 / op</td>
</tr>
<tr>
<td>probabilistic ((A, b, c, isotropic))</td>
<td>310000000 / s</td>
<td>15 / op</td>
</tr>
<tr>
<td>constructing triangle quadrics</td>
<td></td>
<td></td>
</tr>
<tr>
<td>classical ((A, b))</td>
<td>236000000 / s</td>
<td>20 / op</td>
</tr>
<tr>
<td>classical ((A, b, c))</td>
<td>221000000 / s</td>
<td>21 / op</td>
</tr>
<tr>
<td>probabilistic ((A, b))</td>
<td>220000000 / s</td>
<td>216 / op</td>
</tr>
<tr>
<td>probabilistic ((A, b, c))</td>
<td>187000000 / s</td>
<td>257 / op</td>
</tr>
<tr>
<td>probabilistic ((A, b, isotropic))</td>
<td>554000000 / s</td>
<td>86 / op</td>
</tr>
<tr>
<td>probabilistic ((A, b, c, isotropic))</td>
<td>495000000 / s</td>
<td>96 / op</td>
</tr>
<tr>
<td>computing (x_{\text{min}})</td>
<td></td>
<td></td>
</tr>
<tr>
<td>solving (A^{-1}b)</td>
<td>261000000 / s</td>
<td>17 / op</td>
</tr>
<tr>
<td>Eigen::JacobiSVD [GJ*10]</td>
<td>444500000 / s</td>
<td>1168 / op</td>
</tr>
<tr>
<td>Eigen::BDCSVD [GJ*10]</td>
<td>307000000 / s</td>
<td>1560 / op</td>
</tr>
<tr>
<td>svd3 [Jan14, MST*11]</td>
<td>328000000 / s</td>
<td>1460 / op</td>
</tr>
</tbody>
</table>

#### 4.2. Properties

Figure 2 shows the results of various experiments performed on the different quadrics. In each scenario, fixed percentage of the vertices were removed via incremental decimation using edge collapses. We exemplarily depict 70% as a compromise to show a high level of decimation but not too high, which would limit the degrees of freedom too much. Given an edge with vertices \( v_1 \) and \( v_2 \), the new vertex position is chosen as \( x_{\text{min}} \) of the quadric \( Q' = Q_{v_1} + Q_{v_2} \). Collapses are performed in descending order of \( x_{\text{min}}Q'x_{\text{min}} \), skipping those that would lead to flips. The classical quadrics were minimized with an SVN using the edge center as reference for the least-norm solution.

Point quadrics refer to \( \|x - v\|^2 \) for vertex position \( v \). Their minimizer is the center of gravity of all summed up vertices. These quadrics are always stable and favor regular triangulations but obviously do not preserve sharp features.

The regularly triangulated cube with no noise demonstrates the feature preservation of plane-based quadrics. However, it also shows the instability of classical quadrics: every collapse has zero error leading to a highly irregular triangulation. In contrast, our probabilistic quadrics lead to quite regular triangulations while preserving features.

In the second experiment the cube has a small amount of vertex noise and a non-uniform triangulation. The vertex noise amplifies the uncertainty in the normal in regions of smaller triangles. Plane quadrics are not area-weighted and sensitive to the noise in the normal. Thus, they are more easily affected by irregular triangulation and tend to preserve regions of smaller triangles even if there is no geometric reason to do so. The area-weighted triangle quadrics lead to more regular triangulations.
Figure 2: Various experiments demonstrating the properties of our probabilistic quadrics in comparison to classical quadrics under different conditions. The top row shows the input mesh. The other rows show the results of removing 70% of the vertices via incremental decimation using different quadrics. Our probabilistic quadrics naturally preserve features, favor regular triangulations, and can adapt to noise. \( \Sigma \) were chosen to correspond to 5% of normal/edge length, except in the middle case where it went up to 50% on the noisy end. See Section 4.2 for a detailed discussion. The rendering uses an outline shader based on normal to improve readability. Best viewed in the digital version. Centrifuge model from [ZJ16].
The third experiment has spatially varying noise levels. We vary \( \Sigma_n \) and \( \Sigma_p \) with the same intensity to show that our probabilistic quadrics can exploit prior information about the data uncertainty if available. Given the magnitude of the noise they are unable to fully smooth the mesh while preserving the features but still perform better than the classical quadrics. Furthermore, the probabilistic triangle quadrics outperform the plane quadrics as they model the noise source more accurately (noise in the vertices vs. noise in the normal).

In the fourth experiment a spiky noise was applied, similar to salt-and-pepper noise in images. It has a small magnitude but because the perturbed vertices were also moved towards a neighbor, the normal can change drastically. The classical quadrics are unable to remove these spikes as they consider them features, regardless of how small they are. Our probabilistic quadrics have no difficulties smoothing over this noise even though they were designed for Gaussian noise. In particular, the triangle quadric is very resilient against unreliable normals of small triangles.

The final experiment shows a CAD object with irregular triangulation, curved and flat regions, and some features. While both preserve features, the resulting triangulation caused by the classical quadrics is more irregular compared to the ones produced by our probabilistic quadrics.

5. Applications

Our probabilistic quadrics, although specially constructed, are still simple 4 x 4 matrices and thus can be used in any quadric-based application as a drop-in replacement for classical quadrics. In the previous section we already show their effect on incremental decimation. Vertex clustering benefits similarly except that the output topology is fixed by the method. In the following we showcase a few other applications.

5.1. Isosurface Extraction

Isosurface extraction is an important tool, especially in medical imaging. A sizeable amount of research has been devoted to using quadrics to preserve sharp features in the extraction of surfaces from volumetric input data [KBSS01, JLSW02, SW04]. Figure 3 shows the effect of probabilistic plane quadrics when used in dual contouring. Using \( \sigma_n \) and \( \sigma_p \) with the same intensity to show that our probabilistic quadrics can exploit prior information about the data uncertainty if available. Given the magnitude of the noise they are unable to fully smooth the mesh while preserving the features but still perform better than the classical quadrics. Furthermore, the probabilistic triangle quadrics outperform the plane quadrics as they model the noise source more accurately (noise in the vertices vs. noise in the normal).

In the fourth experiment a spiky noise was applied, similar to salt-and-pepper noise in images. It has a small magnitude but because the perturbed vertices were also moved towards a neighbor, the normal can change drastically. The classical quadrics are unable to remove these spikes as they consider them features, regardless of how small they are. Our probabilistic quadrics have no difficulties smoothing over this noise even though they were designed for Gaussian noise. In particular, the triangle quadric is very resilient against unreliable normals of small triangles.

The final experiment shows a CAD object with irregular triangulation, curved and flat regions, and some features. While both preserve features, the resulting triangulation caused by the classical quadrics is more irregular compared to the ones produced by our probabilistic quadrics.

In the example we chose a mesh with spatially strongly varying noise. Classical quadrics are unable to remove noise above a certain level and start to treat the peaks as features to be preserved. Using a global \( \Sigma \) is not granular enough to account for the spatially varying nature of the noise: the value must be high enough to smooth over the noise but that negatively affects features at the same time.

If the amount of noise is known a priori (even if only approximately), then this can be taken into account to construct probabilistic quadrics with a spatially varying \( \Sigma \). Even without this information we can make an “educated guess”. In this example we chose a simple heuristic: the per-vertex absolute angle defect. Lower angle defect indicates less noise. While not perfect, this already lets us smooth over the noisy parts while still reasonably preserving features.

5.2. Adaptive Mesh Smoothing

In Figure 4 we demonstrate mesh smoothing and denoising in the spirit of [VNMC10] and [LTB19]. We initialize each face with its probabilistic triangle quadric. These quadrics are then smoothed over multiple iterations by averaging them with their neighboring faces. Finally, for each vertex we compute the sum of adjacent face quadrics and move the vertex into the minimizer of the quadric.

In the example we chose a mesh with spatially strongly varying noise. Classical quadrics are unable to remove noise above a certain level and start to treat the peaks as features to be preserved. Using a global \( \Sigma \) is not granular enough to account for the spatially varying nature of the noise: the value must be high enough to smooth over the noise but that negatively affects features at the same time.

If the amount of noise is known a priori (even if only approximately), then this can be taken into account to construct probabilistic quadrics with a spatially varying \( \Sigma \). Even without this information we can make an “educated guess”. In this example we chose a simple heuristic: the per-vertex absolute angle defect. Lower angle defect indicates less noise. While not perfect, this already lets us smooth over the noisy parts while still reasonably preserving features.

5.3. Subdivision and Interpolation

Inspired by Gaussian-product Subdivision Surfaces [PBW19], we use quadrics to define an experimental non-linear surface interpolation scheme, shown in Figure 5. Instead of interpolating surface positions or applying a subdivision stencil on vertex positions, we
interpolate error quadrics. More concretely, we annotate each vertex with a probabilistic plane quadric with normal $n$ and covariance $\Sigma_n$. We then interpolate the quadrics and compute new vertex positions as the minimizer of the interpolated quadrics. This scheme permits a spectrum of surface properties, from smooth with low curvature (high $\Sigma$) to sharp features (low $\Sigma$). Anisotropic covariances can be used to apply these properties direction-dependently. This approach leads to an equivalent formulation as [PBW19] and can be seen as an alternative interpretation of their method.

6. Future Work

We presented the derivation of the probabilistic plane and triangle quadric assuming independent, potentially anisotropic Gaussian noise for each input parameter. An obvious future avenue of research is to investigate various other probabilistic quadrics and their feasibility. This includes different types of noise such as uniform, chi-square, gamma, or pareto distributed inputs but also cases where the linear independence assumption does not hold.

Even when analytical solutions are intractable, we mentioned that probabilistic quadrics can still be computed via numerical integration. In a sense, averaging a large number of classical quadrics is already a numerical integration, although a spatial one. If the input distribution can be simulated it is also possible to construct local quadrics via Monte Carlo integration.

In Section 5.2 we used the angle defect as a simple measure for local noise. This method can be refined by investigating more sophisticated measures that could for example quantify the anisotropy of the noise. Our subdivision and interpolation scheme of Section 5.3 is a simple proof-of-concept and deserves a proper investigation. One particular issue that arose was that weighting interpolated quadrics behaves non-linearly. Finding a mapping that makes it more linear would help for intuitive control.

7. Conclusion

Classical quadrics treat the input geometry as ground truth. Any noise and uncertainty present in the data can only be incorporated in an ad hoc fashion by averaging the quadrics over large regions. In contrast, our probabilistic quadrics embrace the uncertain nature of most inputs and model the expected quadratic error over the input distribution. We show how to efficiently construct probabilistic versions for two popular quadrics: the plane quadric built by position and normal under Gaussian noise and the triangle quadric built from three positions, each under Gaussian noise. This probabilistic framework lets us deal with uncertainty in a granular and principled manner.

A welcome side effect is that all our quadrics are full rank by construction (for non-zero covariance). This means that finding the minimizer of a quadric can be done robustly by solving a simple $3 \times 3$ linear system instead of a singular value decomposition which is faster by a factor of 50 in our benchmarks.

Our experiments also show that the probabilistic quadrics naturally favor regular triangulations while still being feature preserving. Their ability to incorporate local information about the shape of the noise enables us to deal with spatially varying noise.
Because our probabilistic quadrics are still quadratic functions in form of a symmetric positive definite $4 \times 4$ matrix, they can be used as a drop-in replacement in any application using classical quadrics. We demonstrate their versatility by showing how they improve decimation, isosurface extraction, and smoothing. More exploratory, we construct a non-linear subdivision and interpolation scheme based on quadric minimization.

References


[MST+11] MCADEMS A., SELLE A., TAMSTORF R., TERAN J., SIFAKIS E.: Computing the singular value decomposition of 3x3 matrices with minimal branching and elementary floating point operations. 5


Appendix A: Derivation of Probabilistic Quadric for Triangles under Gaussian Noise

All derivations were checked with a custom program that performs vector arithmetic symbolically using polynomials in canonical form (in which equality is decidable and reasonably efficient).

Definition A.1 (Cross product matrix) $[\mathbf{I}^T_x]$ denotes the skew-symmetric matrix corresponding to the cross product, i.e.

$$\mathbf{a} \times \mathbf{b} = [\mathbf{a}^T]_{x} \mathbf{b} = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \quad (17)$$

Due to its skew-symmetry, $[\mathbf{a}^T]_{x} = -[\mathbf{a}^T]_{x}^T$.

Definition A.2 (Cross-interference matrix) $[\mathbf{C}_i] : S_3^3 \times S_3^3 \rightarrow S_3^3$ is a symmetric matrix capturing second-order terms in the probabilistic quadrics:

$$\mathbf{C}_i[A, B]_{xy} = A_{xy}B_{zz} - 2A_{xz}B_{yz} + A_{yz}B_{xz}$$

$$\mathbf{C}_i[A, B]_{yx} = -A_{yx}B_{zz} + 2A_{zx}B_{yz} - A_{zy}B_{xz}$$

$$\mathbf{C}_i[A, B]_{zx} = A_{zy}B_{yy} - A_{yz}B_{yy} - A_{yx}B_{yy} + A_{xy}B_{yy}$$

$$\mathbf{C}_i[A, B]_{yz} = A_{zx}B_{xx} - 2A_{xy}B_{xx} + A_{yz}B_{xx}$$

$$\mathbf{C}_i[A, B]_{zy} = -A_{yx}B_{xx} + A_{xy}B_{xx} + A_{zx}B_{xx} - A_{zy}B_{xx}$$

$$\mathbf{C}_i[A, B]_{zz} = A_{xx}B_{yy} - 2A_{xy}B_{xy} + A_{yy}B_{xx}$$

**Lemma A.1** Given $x \sim N(\bar{x}, \Sigma_x)$, it holds that

$$\mathbf{E}[\mathbf{x}^T \mathbf{T}] = \bar{x}^T \Sigma_x + \Sigma_{xx} \mathbf{T}$$

(19) because $\mathbf{E}[\mathbf{x}_i \mathbf{v}] = \bar{x}_i \mathbf{v} + \Sigma_{x_i \mathbf{v}}$ (second moment of Gaussian).

**Lemma A.2** Given $M \in \mathbb{R}^{3 \times 3}$ and $\mathbf{v} \in \mathbb{R}^3$, it holds that

$$\mathbf{Tr}[M \mathbf{v} \mathbf{v}^T] = \mathbf{v}^T M \mathbf{v}$$

(20) because $\mathbf{Tr}[ba^T] = a^T b$ for all $a, b \in \mathbb{R}^3$.

**Lemma A.3** Given $x \sim N(\bar{x}, \Sigma_x)$ and a symmetric matrix $A$, [MP92] show that

$$\mathbf{E}[\mathbf{x}^T \mathbf{A} \mathbf{x}] = \bar{x}^T \mathbf{A} \bar{x} + \mathbf{Tr}[\mathbf{A} \Sigma_x]$$

(21)
Lemma A.4 Given $a \sim N(\bar{a}, \Sigma_a)$ and $b \sim N(\bar{b}, \Sigma_b)$, it holds that

$$\mathbb{E}_a[[a]_x \Sigma_a[b]_x^T] = [\bar{a}]_x \Sigma_a[\bar{b}]_x^T + \text{Ci}[\Sigma_a, \Sigma_b].$$ (22)

This follows from using $\mathbb{E}_a[a_i a_j] = a_i a_j + \Sigma_{a,i,j}$ and expanding all terms. Everything without $\Sigma_a$ can be reassembled into $[\bar{a}]_x \Sigma_a[\bar{b}]_x^T$ and the rest forms $\text{Ci}[\Sigma_a, \Sigma_b]$.

Lemma A.5 Given $x \in \mathbb{R}^3$ and $A, B \in S_+^3$, it holds that

$$\text{Tr}[A \cdot [x]_x \cdot B \cdot [x]_x^T] = x^T \text{Ci}[A, B] \cdot x.$$ (23)

We simply used our symbolic computation tool to verify equality of both sides.

Lemma A.6 Given $a \sim N(\bar{a}, \Sigma_a)$ and $b \sim N(\bar{b}, \Sigma_b)$, it holds that

$$\mathbb{E}_{a,b}[[a \times b](a \times b)^T] = (\bar{a} \times \bar{b})(\bar{a} \times \bar{b})^T + [\bar{a}]_x \Sigma_a[\bar{b}]_x^T + [\bar{b}]_x \Sigma_b[\bar{a}]_x^T + \text{Ci}[\Sigma_a, \Sigma_b].$$ (24)

Proof Using Lemma A.1 and Lemma A.4:

$$\mathbb{E}_{a,b}[[a \times b](a \times b)^T] = \mathbb{E}_{a,b}[[a]_x \times [b]_x^T [a]_x] = \mathbb{E}_{a,b}[[a]_x \times [b]_x^T [a]_x] \Sigma_a[\bar{b}]_x^T + [\bar{a}]_x \Sigma_a[\bar{b}]_x^T + [\bar{b}]_x \Sigma_b[\bar{a}]_x^T + \text{Ci}[\Sigma_a, \Sigma_b]$$

(25)

Lemma A.7 Given $a \sim N(\bar{a}, \Sigma_a)$, $b \sim N(\bar{b}, \Sigma_b)$, and $c \sim N(\bar{c}, \Sigma_c)$, it holds that

$$\mathbb{E}_{a,b,c}[[a \times b \times c]^T] = (\bar{a} \times \bar{b}) \Sigma_c[\bar{c}]_x^T + [\bar{a}]_x \Sigma_a[\bar{b}]_x^T + [\bar{b}]_x \Sigma_b[\bar{a}]_x^T + \text{Ci}[\Sigma_a, \Sigma_b, \Sigma_c].$$ (26)

Proof Using Lemma A.1:

$$\mathbb{E}_{a,b,c}[[a \times b \times c]^T] = \mathbb{E}_{a,b,c}[[a]_x \times [b]_x^T [c]_x^T] = \mathbb{E}_{a,c}[[a]_x \times (\bar{b} \Sigma_b + [\bar{b}]_x \Sigma_c [\bar{c}]_x^T)$$

(27)

$$= (\bar{a} \times \bar{b}) \Sigma_c[\bar{c}]_x^T + [\bar{a}]_x \Sigma_a[\bar{b}]_x^T + [\bar{b}]_x \Sigma_b[\bar{a}]_x^T + \text{Ci}[\Sigma_a, \Sigma_b, \Sigma_c]$$

Theorem A.9 Given $p \sim N(\bar{p}, \Sigma_p)$, $q \sim N(\bar{q}, \Sigma_q)$, and $r \sim N(\bar{r}, \Sigma_r)$, it holds that

$$\mathbb{E}[b] = \mathbb{E}[(p \times q + r + q \times r) \cdot [p, q,r]] = (\bar{p} \times \bar{q} + \bar{r} \times \bar{r} + \bar{r} \times \bar{r}) \cdot [\bar{p}, \bar{q}, \bar{r}] + (\bar{q} \times \bar{p}) \Sigma_r(\bar{p} \times \bar{q})$$

(29)

$$+ (\bar{r} \times \bar{q}) \Sigma_p(\bar{q} \times \bar{r}) + (\bar{r} \times \bar{p}) \Sigma_p(\bar{r} \times \bar{p})$$

$$+ \text{Ci}[\Sigma_p, \Sigma_q, \Sigma_r] + \text{Ci}[\Sigma_q, \Sigma_r] + \text{Ci}[\Sigma_r, \Sigma_p]$$

Proof The determinant can be written as a triple product $[p, q,r] \times (p \times q) = (p \times q)^T r = (q \times r)^T p = (r \times p)^T q$. Expanding the product and choosing a suitable triple product yields three terms of form $(a \times b)(a \times b)^T$. The claim results from applying Lemma A.6 and re-assembling some cross products.

Theorem A.10 Given $p \sim N(\bar{p}, \Sigma_p)$, $q \sim N(\bar{q}, \Sigma_q)$, and $r \sim N(\bar{r}, \Sigma_r)$, it holds that

$$\mathbb{E}[c] = \mathbb{E}[(p \times q, r)]^2 = |\bar{p}, \bar{q}, \bar{r}|^2$$

$$+ (\bar{p} \times \bar{q}) \Sigma_r(\bar{p} \times \bar{q}) + (\bar{q} \times \bar{r}) \Sigma_p(\bar{q} \times \bar{r}) + (\bar{r} \times \bar{p}) \Sigma_p(\bar{r} \times \bar{p})$$

$$+ p^T \Sigma_q \Sigma_r + q^T \Sigma_p \Sigma_r + r^T \Sigma_p \Sigma_r + \text{Tr}[\Sigma_p, \Sigma_r, \Sigma_q]$$

(30)

Proof

$$\mathbb{E}_{p,q,r}[[p, q, r]^2] = \mathbb{E}_{p,q,r}[[p, q, r]^T (p \times q)(p \times q)^T] r$$

$$= \mathbb{E}_{r}[r^T \mathbb{E}_{p,q}[[p, q, r]^T] r]$$

(31)

Set $M = \mathbb{E}_{p,q}[[p, q, r]^T] r$ and apply Lemma A.3:

$$\mathbb{E}_r[r^T \mathbb{E}_{p,q}[[p, q, r]^T] r] = \mathbb{E}_r[r^T M r]$$

$$= r^T M F + \text{Tr}[MF]$$

(32)

The claim now results from expanding $M$ (Lemma A.6), using the linearity of the trace, and applying Lemma A.2 and A.5.

Theorem A.11 For any positive definite matrix $\Sigma \in S_+^3$ and any vector $v \in \mathbb{R}^3$ the matrix $A = m \Sigma + \Sigma$ is invertible.

Proof $m \Sigma$ is a positive semidefinite matrix because $v^T m \Sigma v \geq 0$ for all $v \in \mathbb{R}$. The sum of a positive semidefinite and a positive definite matrix is positive definite. Any positive definite matrix is invertible. Thus $m \Sigma + \Sigma$ is invertible.

Theorem A.12 For any positive definite matrix $\Sigma \in S_+^3$ and any vector $n \in \mathbb{R}^3$ the matrix $A = [n]_x \Sigma [n]_x^T$ is positive definite and thus invertible.

Proof Let $v$ be a vector in $\mathbb{R}^3 \setminus 0$. Then $v^T A v = v^T [n]_x \Sigma [n]_x^T v = (v \times n)^T \Sigma (v \times n) > 0$ due to $\Sigma$ being positive definite.

Theorem A.13 For any two positive definite matrix $A, B \in S_+^3$ the matrix $\text{Ci}[A, B]$ is also positive definite.

Proof Let $v$ be a vector in $\mathbb{R}^3 \setminus 0$. Then $v^T \text{Ci}[A, B] v = \text{Tr}[A \cdot [v]_x \cdot B \cdot [v]_x^T]$ via Lemma A.5. From Theorem A.12 we know that $M := [v]_x \cdot B \cdot [v]_x^T$ is positive definite. The square root of a positive definite matrix is also positive definite. Thus $\text{Tr}[AM] = \text{Tr}[A \sqrt{M} \sqrt{V}] = \text{Tr}[\sqrt{M} A \sqrt{V} M] > 0$, because $\sqrt{M} A \sqrt{V}$ is positive definite and the trace of a positive definite matrix is positive.